

WAY TO DATA QUALITY AND DATA OBSERVABILITY



DataKitchen

1 / Copyright DataKitchen.io

The DataOps Way To Data Quality and Data Observability

Christopher Bergh, Charles Bloche, and Gil Benghiat

Table of Contents

Data Quality Overview

[Data Quality Testing Is At The Core of Four Critical Data Team Processes](#)

['The Mystery Box Full of Data Errors'](#)

[Data Quality vs. Data Observability: The Pets and Cattle of Your Data Estate](#)

[Stop Paying The Data Quality Tax](#)

The Seven Deadly Sins of Data Quality

[The Seven Deadly Sins of Data Quality](#)

[The Data Errors Shame Game: Why Data Engineers Avoid Harsh Truths](#)

[Is Your Team in Denial about Data Quality? Here's How to Tell](#)

[The Ostrich Problem: Your Data Team Thinks Their Job Ends at Deployment.](#)

[The Data Quality Revolution Starts with You](#)

[Why Data Quality Isn't Worth The Effort: Data Quality Coffee With Uncle Chip #1](#)

[Data Quality The DataOps Way](#)

[Data Quality When You Don't Understand the Data: Data Quality Coffee With Uncle Chip #3](#)

The Five Organizational Sins of Data Quality

[The Five Organizational Sins of Data Teams](#)

[Warring Tribes into Winning Teams: Improving Teamwork in Your Data Organization With DataOps and Relational Coordination](#)

[The Art of Data Buck-Passing 101: Mastering the Blame Game in Data and Analytic Teams](#)

[Unlocking Data Team Success: Are You Process-Centric or Data-Centric?](#)

[You're Thinking About Data Products All Wrong](#)

[Process Guardianship: The Most Valuable Data Engineering Work You're Probably Not Doing](#)

Core Data Quality Concepts

[Flip the Script on Data Quality: Shift Left, Shift Down, and Take Control](#)

[Scaling Data Reliability: The Definitive Guide to Test Coverage for Data Engineers](#)

[The Syntax, Semantics, and Pragmatics Gap in Data Quality Validation Testing](#)

[Data Quality Testing: A Shared Resource for Modern Data Teams](#)

[Peter Piper on the Four Ps of AI Data Quality: Purge, Patch, Push Back, or Pass](#)

Data Architecture and Governance

[Data Quality Test Coverage In a Medallion Data Architecture](#)

[The Race For Data Quality in a Medallion Architecture](#)

[Critical Data Elements: Your Shortcut to Data Governance That Actually Works](#)

Team Performance and Culture

[Data Quality Circles: The Key to Elevating Data and Analytics Team Performance](#)

[How Data Quality Leaders Can Gain Influence And Avoid The Tragedy of the Commons](#)

[War Rooms Suck](#)

[Navigating the Storm: How Data Engineering Teams Can Overcome a Data Quality Crisis](#)

[Why Not Hearing About Data Errors Should Worry Your Data Team](#)

Data Quality Dashboards and Assessments

[Relieve yourself from the Burden of Traditional Data Quality Assessments.](#)

[A Guide to the Six Types of Data Quality Dashboards](#)

Tools and Technology Landscape

[The 2026 Open-Source Data Quality and Data Observability Landscape](#)

[The 2026 Data Quality and Data Observability Commercial Software Landscape](#)

[No Python, No SQL Templates, No YAML: Why Your Open Source Data Quality Tool Should Generate 80% Of Your Data Quality Tests Automatically](#)

[The Data Consultant's Growth Playbook: Accelerating Client Acquisition and Retention with DataKitchen's Open Source TestGen](#)

Real-World Stories and Case Studies

[How I Broke Our SLA and Delighted Our Customer](#)

[When Timing Goes Wrong: How Latency Issues Cascade Into Data Quality Nightmares](#)

[What Does Flossing One Tooth Have to Do with Data Quality?](#)

[We Got Roasted On Reddit For Asking 'Why Data Engineers Don't Test?'](#)

[The Data Consultant's Growth Playbook: Accelerating Client Acquisition and Retention with DataKitchen's Open Source TestGen](#)

Data Observability With Data Quality

[DataOps Observability: Taming the Chaos](#)

[The Five Use Cases in Data Observability: Overview](#)

[The Five Use Cases in Data Observability: Data Quality in New Data Sources](#)

[The Five Use Cases in Data Observability: Effective Data Anomaly Monitoring](#)

[The Five Use Cases in Data Observability: Mastering Data Production](#)

[The Five Use Cases in Data Observability: Fast, Safe Development and Deployment](#)

[The Five Use Cases in Data Observability: Ensuring Accuracy in Data Migration](#)

[Be The First To Know: Smart, Continuous Table Monitoring Has Arrived In TestGen](#)

Introduction

Data quality is not a technology problem. It never has been. The data industry has spent years chasing the next tool, platform, or vendor promise, yet data errors still dominate the daily work of data engineers. Teams keep blaming each other when pipelines break, and business stakeholders quietly distrust the numbers they see. After working with data teams across many industries, we believe the real barriers to data quality are cultural, organizational, and process-related. Teams get stuck in reactive habits—scrambling to fix errors after they happen, pointing fingers across silos, and treating data quality as someone else's issue. The articles in this book aim to honestly identify these patterns and offer a better way.

That better way is DataOps, which applies Agile, DevOps, and lean manufacturing ideas to data and analytics work. DataOps isn't a product you can buy; it's a set of habits and disciplines that, when followed consistently, turn data quality from a constant crisis into a managed, measurable, and ever-improving part of your data environment. The chapters here cover everything from the organizational issues that hurt data quality efforts—like warring tribes, the data shame game, and the ostrich problem—to technical practices that give teams real control: shift-left testing, systematic test coverage, continuous monitoring, and using data observability to spot and fix failures before they affect customers. Together, these chapters offer data teams a clear, honest, and practical guide to doing this work well.

Now, why did we open-source our tools? DataKitchen builds software like DataOps TestGen for data quality testing and profiling, and DataOps Observability for pipeline monitoring, but we've never believed that software alone solves the problem. The main reason data teams struggle with data quality isn't a lack of technology; it's a lack of knowledge, vocabulary, and process frameworks to use that technology well. By releasing TestGen and DataOps Observability as free, open-source products, we're removing the financial barrier that has kept smaller teams, independent consultants, and budget-conscious organizations from accessing enterprise-grade data quality tools. We want every data team, regardless of budget, to have tools that automatically generate tests, continuously monitor tables, and spot anomalies before they cause problems.

We also open-sourced our tools because we believe trust comes from transparency. The data quality and observability market is full of vendors who hide their pricing, lock customers into proprietary systems, and charge per-table fees that make full coverage too expensive. We believe that's wrong. Data quality testing should be available to everyone, not limited. Observability should cover every pipeline and dataset, not just the ones a team can afford. This book and these tools share the same belief: better data quality is achievable for every organization and worth pursuing. We hope you find not just ideas that resonate, but practical steps you can start using today.

Chris, Chip & Gil

Data Quality Testing Is At The Core of Four Critical Data Team Processes

What Is The Common Thread Running Through Four Critical Data Team Processes? Data Testing!

Every data and analytics team juggles multiple responsibilities. You are expected to ensure your data is accurate and fit for purpose. You need to prevent problematic data from entering your data environment. You must stop your data pipelines from delivering embarrassing errors to your stakeholders. And you have to deploy new code and data sets into production without breaking what already works. These four activities might seem distinct at first glance, but they share a fundamental characteristic: they all depend on robust data quality testing to succeed.

Data Quality Testing: The Core of 4 Critical Processes

A Field Guide for Data Engineering Teams



The challenge is that most organizations treat these activities as separate concerns, with different tools, different teams, and different approaches. This siloed thinking obscures a crucial insight. Data quality testing is not just one activity among many. It is the foundational capability that enables all four of these processes. When you understand this connection, you can build a more coherent and effective approach to managing your data estate.

The Four Processes That Define Data Team Operations

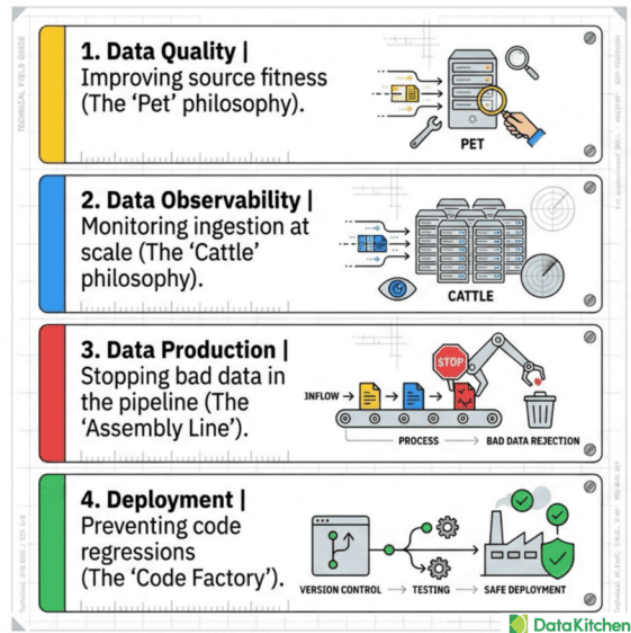
Before diving into how testing connects these activities, let us establish what each process entails and why it matters to your organization.

One Term, Four Distinct Missions



'Data Quality' is often used as a catch-all term, but efficient engineering requires distinct workflows.

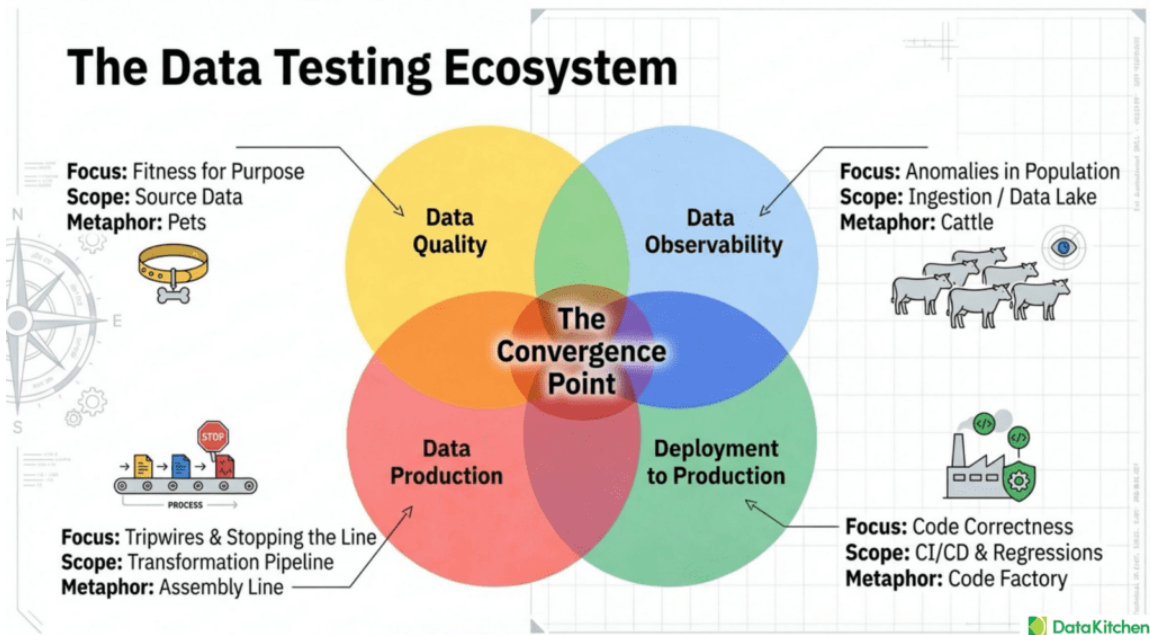
We are effectively managing four different ecosystems, all bound by a single foundational capability: Data Testing.



The first process is Data Quality, which focuses on improving the quality of your data sources and ensuring they are fit for purpose. This is the most traditional understanding of data quality work. Teams engaged in this process spend their time examining specific tables and datasets, running detailed checks against data quality dimensions, building dashboards to track quality scores, and working to influence upstream data sources to fix problems at their origin.

The second process is Data Observability, which has emerged as a critical discipline for teams managing large numbers of tables and datasets. The core problem here is avoiding blame for obvious data problems that slip through unnoticed. When you have hundreds or thousands of tables, you cannot manually inspect each one. Data observability solutions poll your databases looking for anomalies in freshness, volume, schema changes, and statistical patterns. The goal is to identify data ingestion errors and reduce data downtime.

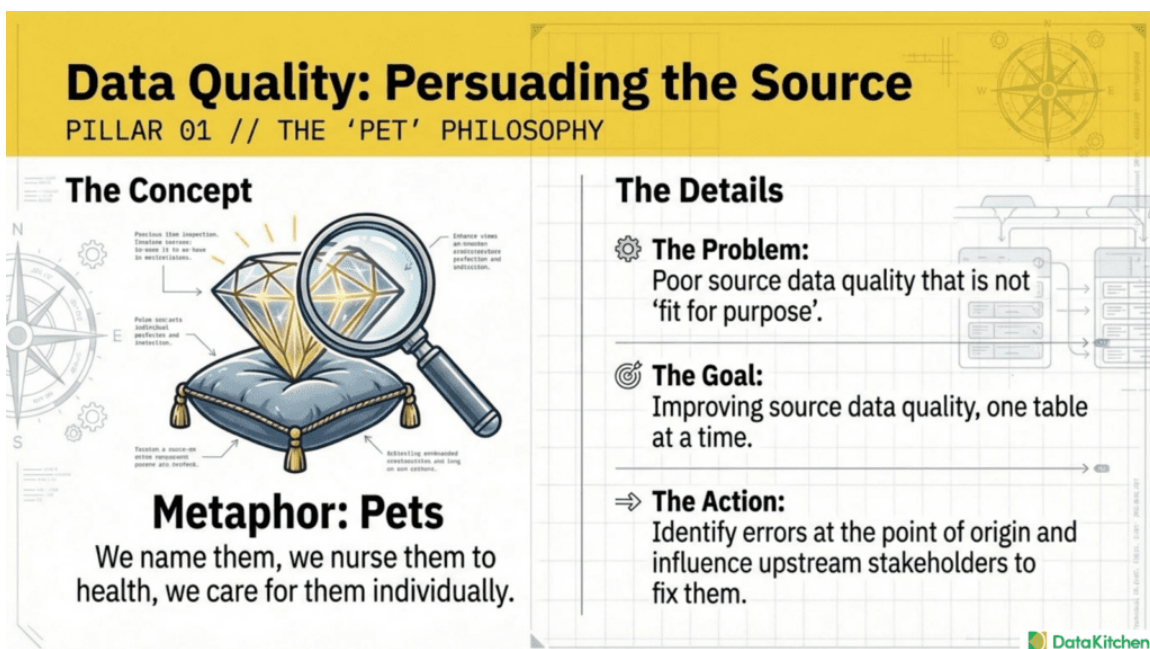
The third process is Data Production, which we might also call Data Journey Quality. This goes beyond simply observing data at rest to monitoring the entire journey that data takes through your production systems. The fundamental question here is how to stop errors in data and deliverables before they reach production. This requires placing tests and monitors at every step of the data production process, with tripwires that can halt processing when problems are detected.



The fourth process is *Deployment to Production*, specifically the regression and impact testing of new code. Your team of data engineers, scientists, and analysts is constantly writing and modifying code. Before that code reaches production, you need to ensure it has been adequately tested. This means going beyond simple unit tests to include full regression testing and functional testing in development environments during CI/CD.

The Data Quality Process: Treating Your Critical Data Like Pets

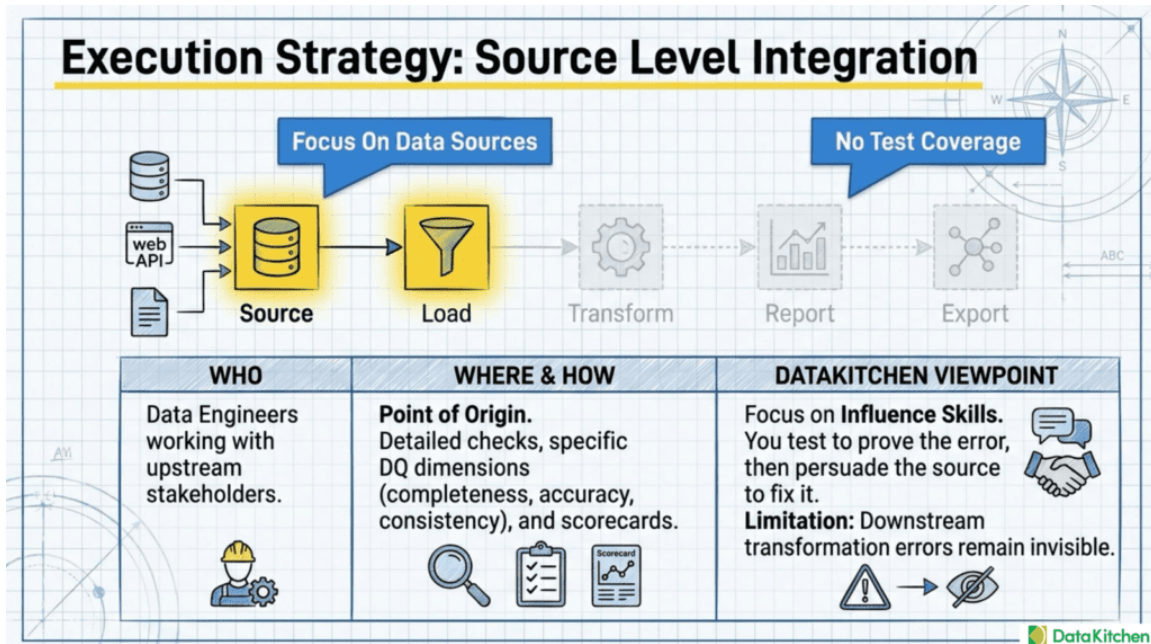
The metaphor of [“pets versus cattle”](#) has become common in infrastructure discussions, but it applies equally well to data quality work. When you treat data like pets, you give individual datasets names, you care for them individually, and you invest significant effort in keeping each one healthy. This approach makes sense for your most critical data assets.



Data quality work in this mode involves persuading the source to improve its data quality. You are not just detecting problems; you are building the case for why upstream systems and processes need to change. This requires finding errors at the point of origin, as close to where data enters your systems as possible. The work involves creating

detailed data quality dashboards that track specific tests across data quality dimensions like completeness, accuracy, consistency, validity, timeliness, and uniqueness.

The testing approach for traditional data quality work tends to be intensive and specific. You write lots of tests for each table, covering every aspect of the data that matters to your business. You score and track results over time. You use those results to influence your data sources, whether those are internal systems, external vendors, or manual data entry processes.



This focused approach has real benefits. It keeps things relatively simple at the data level and allows you to build deep expertise in specific datasets. However, it also has significant limitations. When you focus exclusively on source data quality, errors that occur in downstream tools and deliverables remain invisible to you. Your customers might receive incorrect analytic results even though your source data quality scores look excellent.

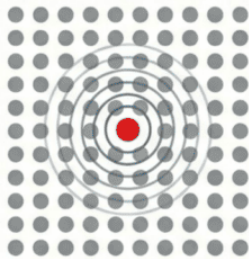
The Data Observability Process: Managing Data Scaled Like Cattle

When you have thousands of tables spread across multiple databases and data platforms, the pet approach simply does not scale. You cannot write hundreds of specific tests for every table. You cannot manually review quality dashboards for each dataset. You need a fundamentally different approach.

Data Observability: Polling for Perversity

PILLAR 02 // THE 'CATTLE' PHILOSOPHY

The Concept



Metaphor: Cattle

Managed in herds. We don't name them; we look for statistical anomalies in the population.

The Details

The Problem: Unfound errors in thousands of tables; getting blamed for "obvious" issues (freshness/volume).

The Goal: Find errors before the customer does. Reduce Data Downtime.

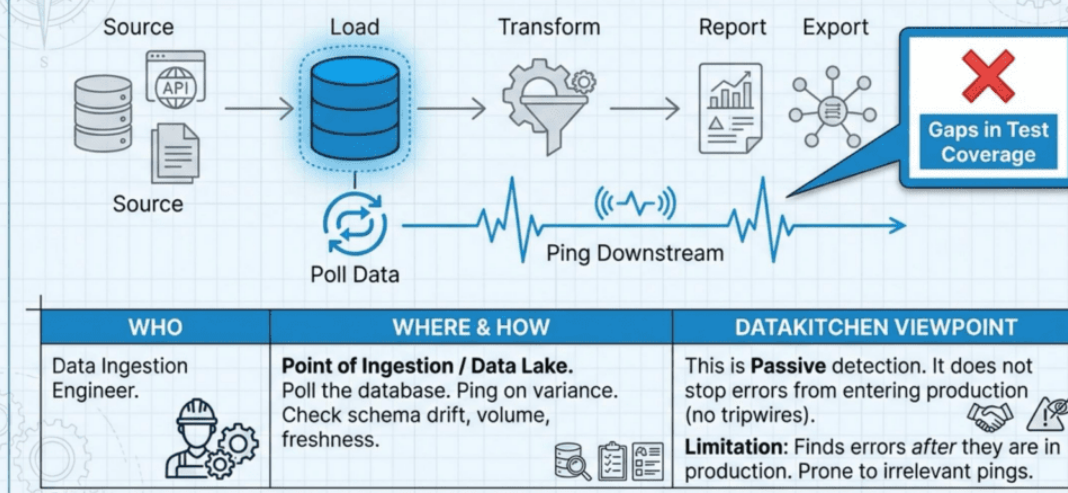
The Action: Poll the database. Ping on variance. Monitor the herd.

DataKitchen

Data observability treats data [more like cattle](#). Instead of naming and caring for individual animals, ranchers manage herds at scale using statistical sampling and pattern recognition. Similarly, data observability solutions monitor your databases and alert you to errors. They look for variance across large populations of tables, checking for things like unexpected changes in row counts, unusual freshness delays, schema modifications, and statistical drift that might indicate data problems.

The target audience for data observability is typically the data ingestion engineer responsible for reliably getting data into the platform. These engineers need to find errors in large databases before customers do. The primary goal is to reduce data downtime, which is the period when data is unavailable, inaccurate, or otherwise unusable.

Execution Strategy: Ingestion Level Integration



DataKitchen

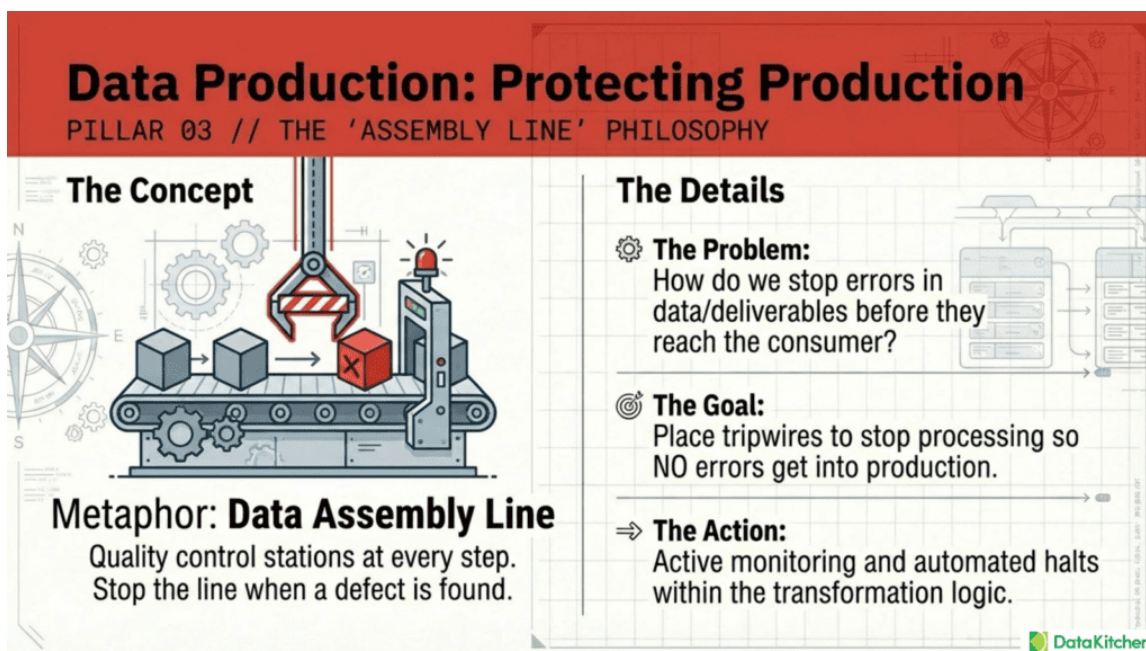
This polling approach enables coverage across far more tables than would be possible with hand-crafted tests. It also provides visibility into data lineage, helping you understand how problems propagate through your systems. However, the observability approach has its own limitations. It finds data errors after they have already gotten into production systems. The automated anomaly detection can generate irrelevant pings that waste time and erode trust. And the

anomaly testing, while broad, is often shallow compared to what you would achieve with specific hand-crafted tests for critical tables.

Perhaps most importantly, data observability alone does not prevent errors from reaching production. It detects problems after the fact rather than preventing them from reaching your stakeholders.

The Data Production Process: Protecting Production with Data Journey Quality

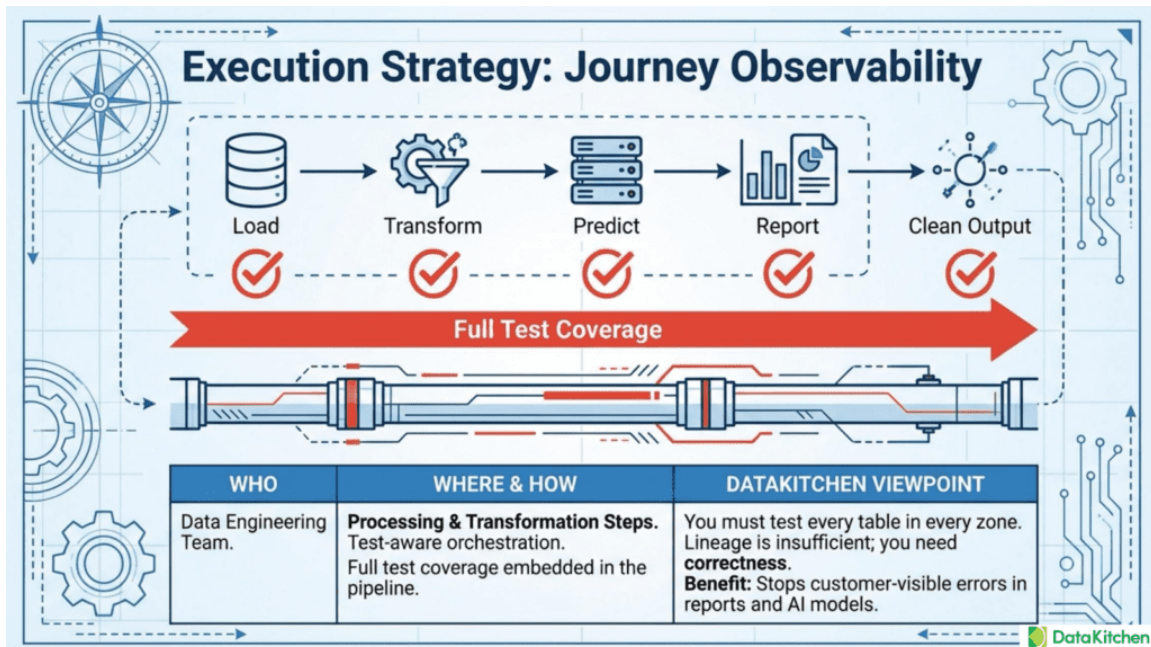
The third process addresses a fundamental gap in the first two approaches. Traditional data quality focuses on sources, and data observability polls data at rest. Neither adequately addresses what happens during the data production process itself. This is the data journey from raw inputs through transformation, enrichment, and aggregation to final deliverables.



[Data Journey](#) Quality operates on a different metaphor: the assembly line. In manufacturing, quality control does not just inspect raw materials at the loading dock or finished goods in the warehouse. Inspections happen at every step of the production process, with the ability to 'stop the line' when problems are detected. This prevents defective work from moving forward and wasting resources on products that will ultimately fail.

The core question for data production is straightforward but challenging: how do you stop errors in data and deliverables before they reach production? This requires placing tests and monitors within the data production processing steps themselves, not as an afterthought. You need either a test-aware orchestrator that can respond to test failures or a dedicated data journey observability tool that integrates with your existing infrastructure.

The goal is full test coverage across your production process, with the ability to find all errors during data production and place tripwires that stop processing. When an error is detected, the system should halt before the problem propagates further downstream or reaches customer-facing systems.

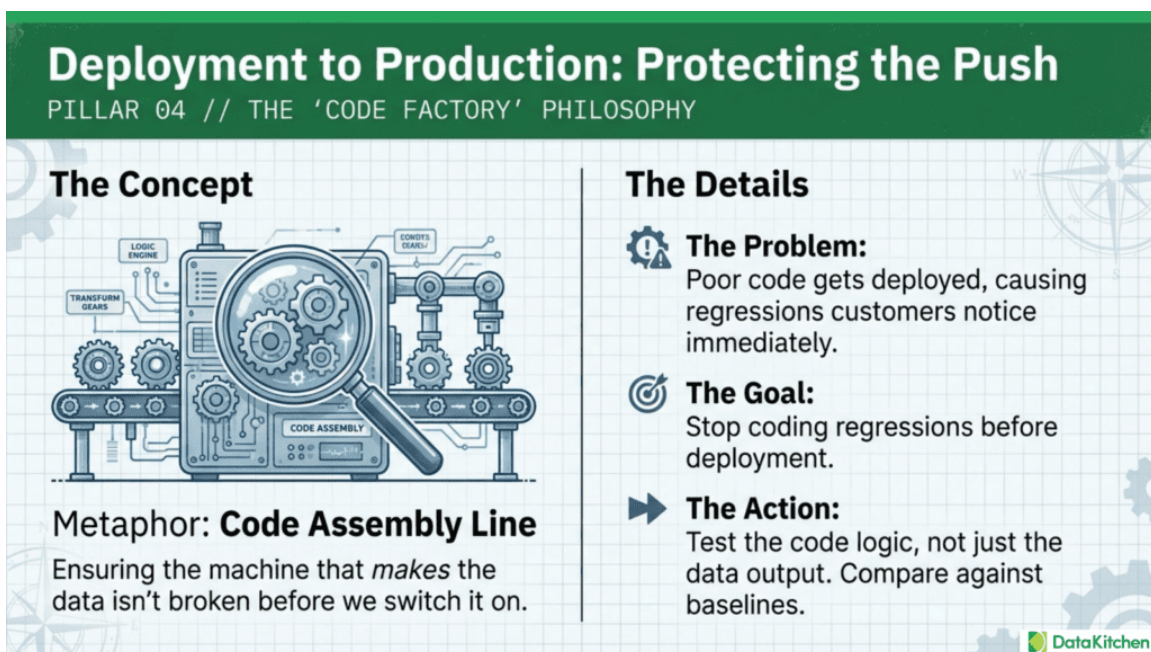


This approach requires testing every table in every zone of your data platform. It means monitoring every tool involved in your data production process and integrating them into a single unified view. Data lineage alone is insufficient because it tells you how data flows but not whether that data is correct at each step.

The testing requirements for data production are substantial. You need both data integration and tool integration, which makes implementation more complex than either traditional data quality or data observability alone. But the benefit is significant: you can actually stop customer-visible errors in data, reports, and models before they cause damage.

The Deployment to Production Process: Regression and Impact Testing

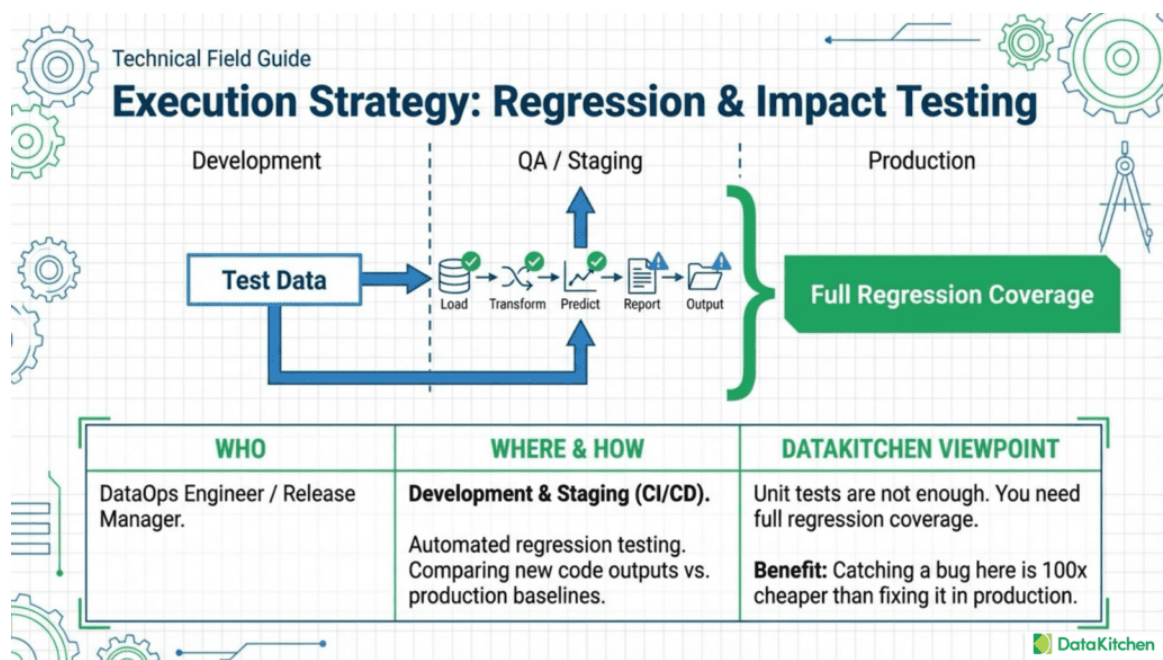
The fourth process shifts focus from data to code. Your data engineering team is constantly writing new transformations, building new pipelines, updating existing logic, and deploying changes to production. Each of these changes carries risk. A modification that works perfectly in development might break something in production. A new feature might introduce a regression in existing functionality.



The metaphor here is another assembly line, but this one produces code rather than data. Just as the data production assembly line needs quality gates, the code production process needs its own checkpoints to ensure that poor code does not reach production, where customers will notice the problems.

The traditional approach to code testing relies heavily on unit tests and manual business reviews. Unit tests verify that individual functions work correctly in isolation, and business reviews ensure that outputs look reasonable to human experts. These approaches are necessary but not sufficient. Unit tests cannot catch problems that only emerge when components interact in production environments. Manual reviews cannot scale to cover every possible scenario and are prone to human error and fatigue.

What data teams need is a fully automated system with [adequate test coverage](#). This includes regression testing that compares the outputs of new code against baseline results from the existing production system. It also includes impact testing that identifies which downstream assets might be affected by a change. And it means integrating these tests into your CI/CD pipeline so they run automatically before any code can be deployed. Great test data is key. Some teams say: *today's code, yesterday's data* for deployment testing.



The testing limitations here involve practical concerns: you need appropriate test data that reflects production scenarios, tools to automate comparison, and CI/CD integration that enforces testing requirements without creating bottlenecks for your development team.

The benefit of getting this right is substantial. Every regression error that slips into production costs far more to fix than one caught during development. The commonly cited figure is that production bugs cost one hundred times more than bugs caught earlier in the development process. This multiplier reflects not only the technical cost of fixing the bug but also the customer impact, the loss of trust, and the emergency response required by production incidents.

The Common Thread: Testing at the Core

When you examine these four processes carefully, a pattern emerges. Each one depends fundamentally on data quality testing, but each applies testing in a different context and for a different purpose.

	Data Quality	Data Observability	Data Production	Deployment to Production
Metaphor	Pets	Cattle	Assembly Line	Code Delivery Line
Primary Goal	Source improvement	Reduce data downtime	Prevent production errors	Prevent code regressions
Action	Persuading the Source	Polling/Pinging	Protecting Production	Protecting the Push
Where	Point of Origin	Data Ingestion	Every step (Tripwires)	Development / CI-CD
Benefit	Improves source data quality	Finds obvious data load issues	Stops customer-visible errors	Reduces fix costs by 100x

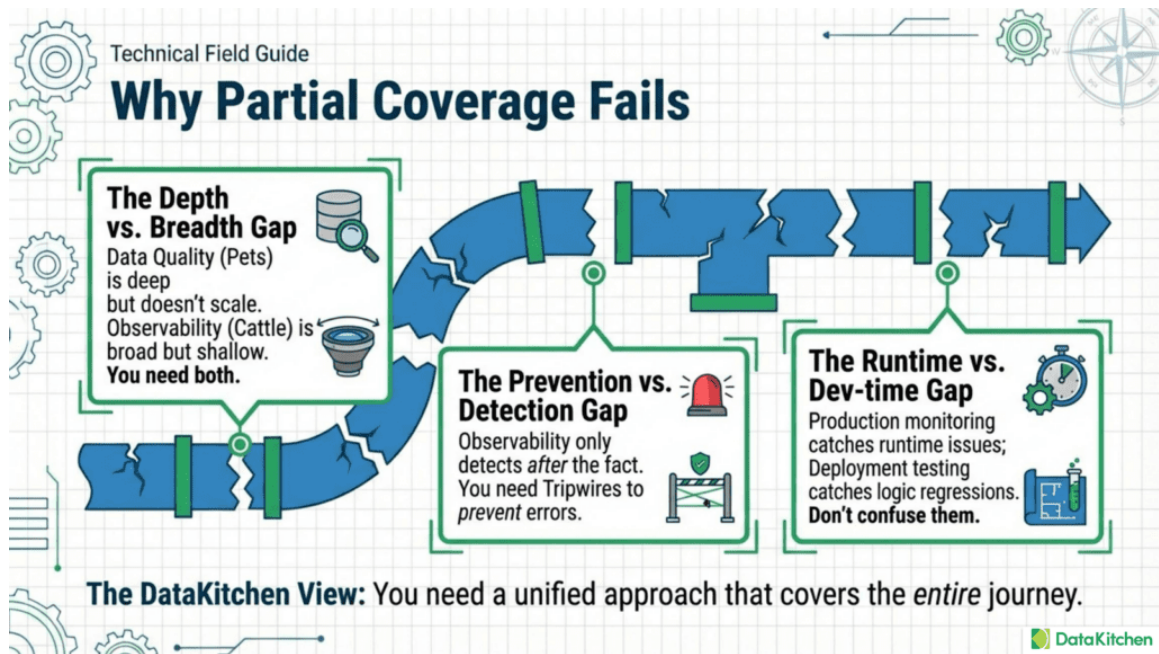
Testing techniques overlap significantly across these domains. A test that checks for null values in a critical column might serve data quality in one context, data observability in another, and deployment validation in a third. The difference lies not in the test itself but in when it runs, what triggers it, and what happens when it fails.

This insight has practical implications for how you structure your testing capabilities. Rather than building separate testing infrastructure for each of these four processes, you can create a unified testing foundation that serves them all. A comprehensive test suite can feed data quality dashboards for source improvement work while simultaneously providing the tripwires needed for data production protection and the regression baselines needed for deployment validation.

Bridging the Gaps Between Processes

Understanding how these four processes relate reveals several gaps that many organizations struggle with.

The gap between data quality and data observability is a matter of depth versus breadth. Data quality provides deep testing for critical tables, but cannot scale to cover everything. Data observability provides broad coverage but often misses problems that only detailed tests would catch. Organizations need both approaches, with clear criteria for which tables get the pet treatment and which are managed as cattle.



The gap between data observability and data production lies between prevention and detection. Observability tells you when something has gone wrong. Data journey observability with tripwires can prevent errors from propagating. Organizations that rely solely on after-the-fact detection will always be playing catch-up, fixing problems after customers have already been affected.

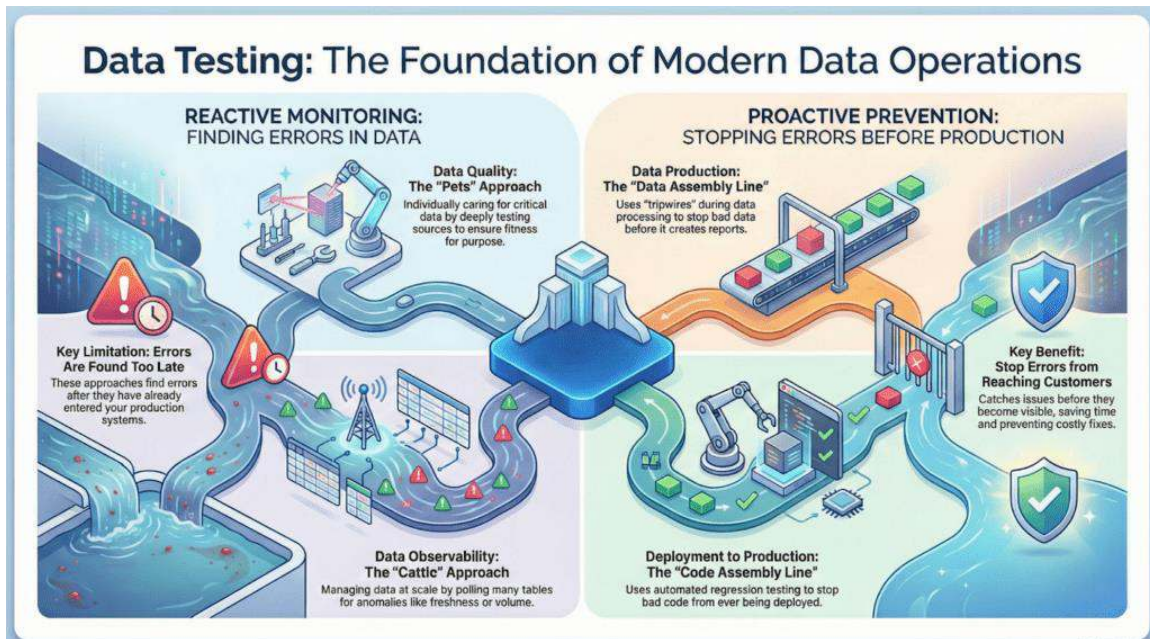
The gap between data production and deployment testing is about runtime versus development time. Data production monitoring catches problems during actual production runs. Deployment testing catches problems before code even reaches production. Both are necessary because some problems manifest only at scale in production environments, while others can and should be caught earlier in development.

Building a Comprehensive Testing Strategy

Given that testing underlies all four of these critical processes, how should data teams approach building their testing capabilities?

The first step is recognizing that testing is not a one-time project but an ongoing capability that requires investment and maintenance. Your test suite will need to evolve as your data assets, pipelines, and business requirements change. Building tests is not enough; you also need processes for reviewing, updating, and retiring tests as circumstances warrant.

The second step is prioritizing test coverage based on business criticality. Not every table needs the same level of testing attention. Your most critical datasets, the ones that drive key business decisions or feed customer-facing applications, deserve the intensive pet treatment. Less critical tables can be managed with a lighter-touch cattle-style observability approach. The key is making this prioritization explicit rather than letting it happen by accident.



The third step is to integrate testing into your production processes rather than treat it as a separate activity. Tests should run automatically as part of your data pipelines, with clear tripwires that halt processing when critical thresholds are breached. This requires both technical integration with your orchestration tools and organizational agreement on which severity of test failure should stop production.

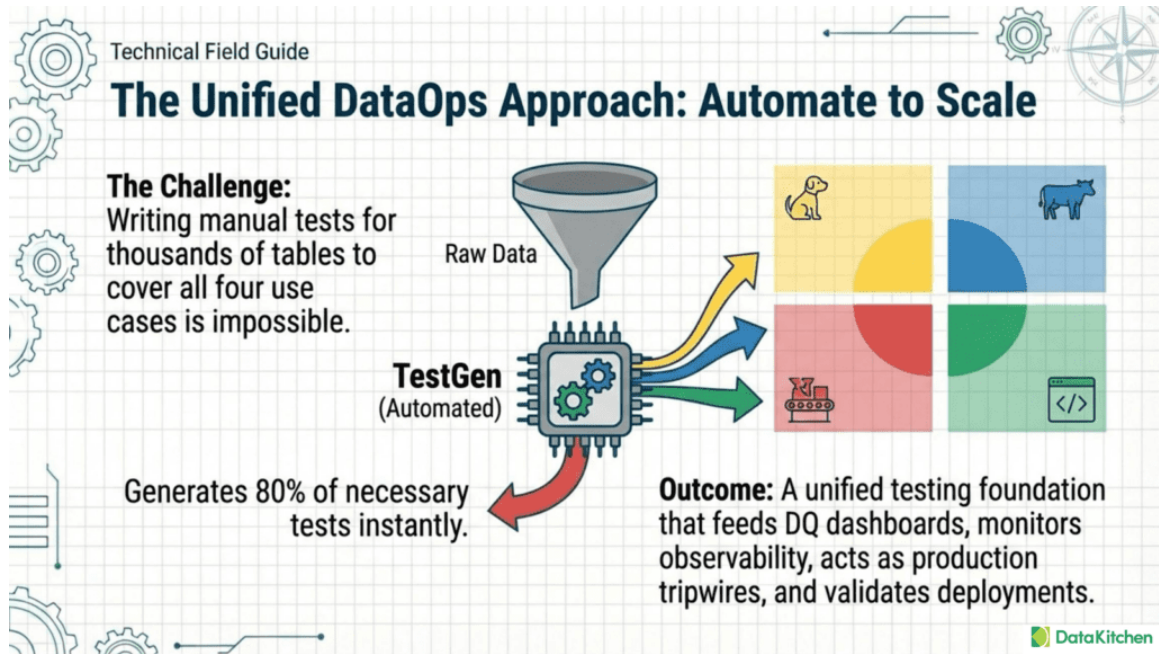
The fourth step is connecting testing to your deployment workflow. Every code change should trigger appropriate regression and impact tests before it can reach production. This requires investment in test automation and CI/CD integration, but the payoff in prevented production incidents is substantial.

The fifth step is to build feedback loops that use test results to drive improvement. Test failures should not just trigger alerts; they should inform root-cause analysis and drive permanent fixes. Data quality scores should drive conversations with upstream data sources. Deployment test failures should prompt an investigation into why the problem was not caught earlier in development.

The Cost of Inadequate Testing

Organizations that underinvest in data quality testing pay the price across all four of these processes. The 1:10:100 rule applies: find problems early to avoid paying the high price of failure.

Without adequate source data quality testing, problems at the origin propagate through your entire data estate, multiplying their impact at every step. You might clean up the symptoms downstream, but the root cause continues generating new problems.



Without adequate data observability testing, embarrassing errors slip through to your customers. You get blamed for obvious problems that automated checks should have caught. Your credibility with business stakeholders erodes over time.

Without adequate data production testing, errors in your transformation logic or pipeline configurations reach production systems. You might not even know about the problems until a customer reports incorrect numbers in a dashboard or a model starts making bad predictions.

Without adequate deployment testing, code changes introduce regressions that break things that used to work. Your team spends increasing amounts of time on emergency fixes and production support, leaving less time for the new development work that the business is waiting for.

The Scale Challenge: You Cannot Hand-Code Your Way to Safety

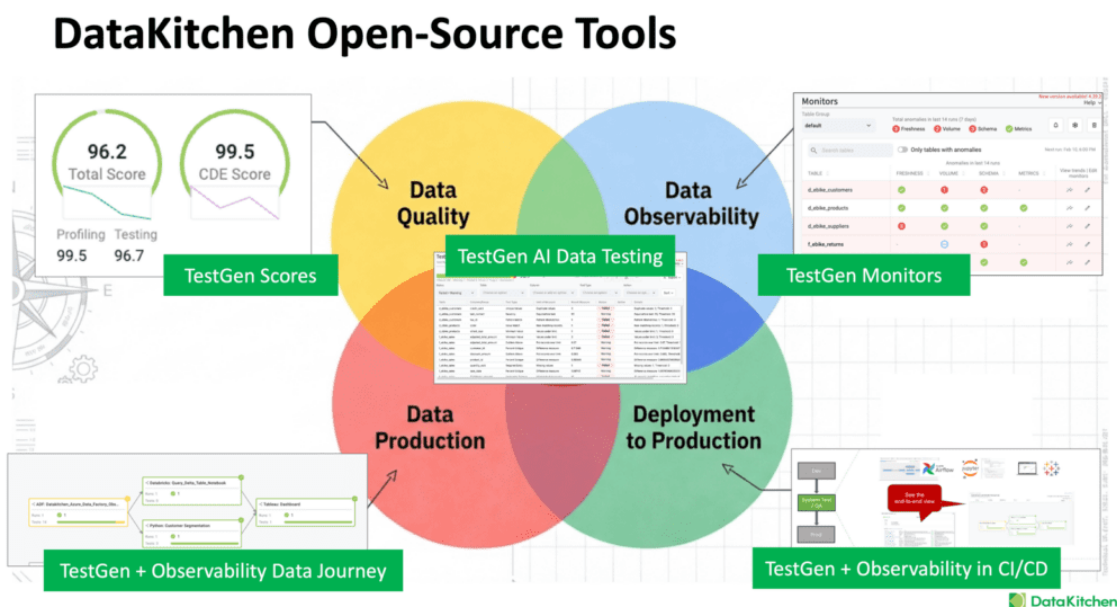
$$\begin{array}{ccccccc} 1,000 & \times & 20 & \times & 4 & = & 80,000+ \\ \text{Tables} & & \text{Tests per Table} & & \text{Environments} & & \text{Tests to write and maintain.} \\ & & (\text{Dimensions +} & & & & \\ & & \text{Anomalies +} & & & & \\ & & \text{Tripwires}) & & & & \end{array}$$

The Result: Teams are forced to cherry-pick a few 'Pets' to test, leaving the vast majority of the data estate exposed.

The cumulative effect of these testing gaps is a data platform that becomes increasingly unreliable and a data team that becomes increasingly reactive. Instead of building new capabilities and delivering business value, you spend your time fighting fires and apologizing for problems.

The Testing Challenge and a Practical Solution

By now, the central role of data quality testing should be clear. Whether you are improving source data quality, implementing data observability across hundreds of tables, building tripwires into your data production pipelines, or validating code deployments, you need tests. Lots of tests. Tests that cover your data comprehensively and catch problems before they reach your customers.



Here is the problem: achieving adequate test coverage across all four of these use cases is extraordinarily time-consuming. Writing tests manually does not scale. A single table might need dozens of tests to cover all relevant data quality dimensions, detect anomalies, serve as a production tripwire, and provide a regression baseline. Multiply that by hundreds or thousands of tables, and you are looking at months of work just to configure or code the tests you need. Most teams never get there. They write tests for a handful of critical tables and hope for the best with everything else.

The remaining twenty percent, the tests that encode specific business rules and domain knowledge unique to your organization, still require human input. But starting with a solid foundation of automatically generated tests means your team can focus their limited time on those high-value custom tests rather than writing boilerplate checks that a tool can handle. This is how you achieve the comprehensive test coverage that makes all four of these critical processes work effectively. This is how you move from reactive firefighting to proactive data quality management.

The Common Thread: Testing is the Core DNA

These four distinct processes share a single foundation. A single test (e.g., 'Check for Nulls') serves different strategic purposes depending on when and where it runs.



What is the Summary of the Data Quality Testing At The Core of 4 Very Important Processes?

Having adequate data quality test coverage is very important to four key activities that every data and analytics team does:

Data Observability: This is where you try to ensure you don't have data errors when you have lots of tables and get blamed for obvious data problems like freshness.

Data Production: Monitoring the data journey and stopping it with tripwires before errors reach production.

Deployment to Production: You have a team of data engineers, scientists, and others who are writing code. You want to ensure the code is adequately tested. Before it gets into production, this means going beyond unit tests into full regression and functional tests in development.

What is the Summary of the Data Quality Testing for Each of The Processes?

How: lots of specific tests, DQ dimensions, data quality dashboards, detailed checks. Just focus on testing and scoring, then influencing your data sources

DataKitchen Viewpoint: DataOps Way to Data Quality; focussed test driven data quality dashboards; influence skills

Testing Limitations: Errors in downstream tools and deliverables are invisible; customer-delivered data analytic results may be incorrect

Problem: Unfound data errors in large sets of data tables, getting blamed for 'obvious data problems'

How: Just Poll Your Database and Ping people when there may be an error; polling 100s of tables to find variance. frequency, volume, schema, drift, time series anomaly detection

Goals: find errors in large databases with 1000s of tables before your customer does. Reduce data downtime

DataKitchen Viewpoint: Does not stop errors from getting into production; no production tripwire, no Data Journey observability

Testing Limitation: Finds data errors after they get into production, irrelevant pings, and partial test coverage

How: Put tests and monitors as part of the data production processing steps. Use a test-aware orchestrator or a Data Journey Observability tool. Full test coverage

Goals: Find all errors during data production and place a tripwire to stop processing so that no errors get into production

DataKitchen Viewpoint: You need to test every table in every zone. Monitor every tool and integrate them into a single view of your process, with trip wires to stop production at every step of the Data Journey. Data Lineage is insufficient.

Deployment to Production Process: Regression & Impact Testing Of New Code Metaphor: Code production Assembly Line

Tagline: Protecting the Push to Production

Problem: Poor code gets into production, and your customers notice.

Who: Data Engineering team; DataOps Engineer who controls the deployment process

Where: Find errors in code during development and deployment to production

How: Regression and impact testing in development and CI/CD

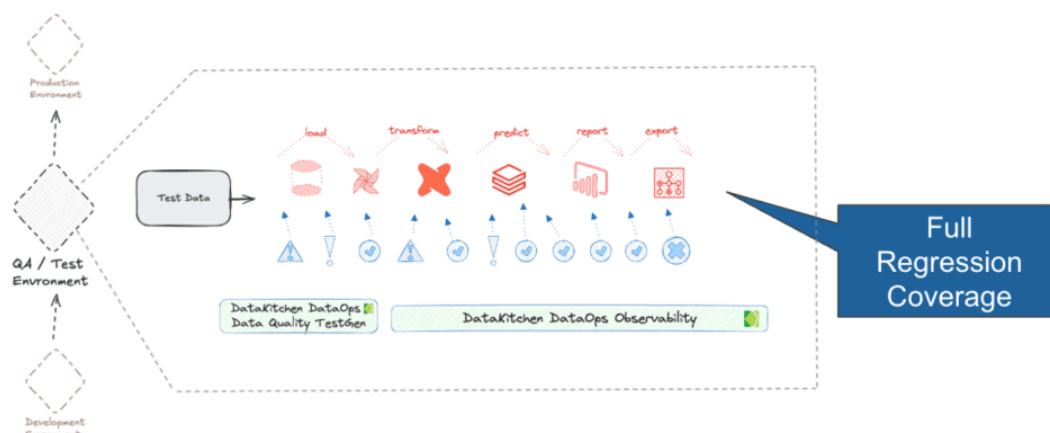
Goals: Stop coding regression before they get deployed to production

DataKitchen Viewpoint: Unit tests and manual business reviews are not enough; you need a fully automated system with adequate test coverage

Testing Limitation: Requires test data, tools, and CI/CD Integration

Benefit: Stop coding regression before it gets deployed to production; every regression error is 100x more expensive to fix in production

Benefit: Stop coding regression before it gets deployed to production; every regression error is 100x more expensive to fix in production



Author: Chris Bergh | Published: Feb 10, 2026

Original article: <https://datakitchen.io/data-quality-testing-is-at-the-core-of-four-critical-data-team-processes/>

‘The Mystery Box Full of Data Errors’

Do you have a dastardly dark database teaming with terrible data quality? Are the data sets around you a mystery box full of potential data errors? Is your data an enigma, wrapped in a pitch-black box, riddled with data irregularities? Or do you feel like the Forest Gump of data, where your data is like a box of chocolates, and you never know what will happen next?

Are you a member of a stressed-out, overworked data team that builds and runs data systems that produce embarrassing data errors regularly? Are you aware that only 22% of data engineers' time is spent on innovation, but 78% on errors and manual execution (Gartner 2022)? Are you aware that a survey of 700 data engineers by DataKitchen & data.world in 2022 found that 52% of Data Engineers said errors are a significant source of burnout?

Do you have embarrassing production data errors your customer found (or, luckily, did not find)? Data errors cause your customers not to trust the data, are hard to find, take time to identify, and waste your time on re-work. Data errors can cause compliance risk and have an opportunity cost for the business because of the downtime. And wrong data or wrong reports/models themselves can cause costly business mistakes.

Do you need more time or business knowledge to create data quality validation tests? Data engineers deal with hundreds of data sets and diverse customer needs. They have backlogs on their daily task lists. So they don't have time or energy to learn about each data set or customer to create robust production data validation tests. They need help creating tests that fit their data and customers' needs without taking too much time to set up or babysit once they go live. They also need help adding new types of tests for the business and adjusting tests when new data arrives. Can data engineers make data stewards their allies in improving data quality through testing?

What if these challenges could be fixed? What are the benefits of shining a light on data and reducing errors in production?

The following white paper discusses how to make embarrassing data errors a thing of the past. We will start with how data engineers are challenged to understand their data and, as a result, need help identifying problematic data records. We will also discuss how the vast majority of data engineers are so busy that they don't know or have time to write data tests to find data errors. We will then cover ways data engineers can create automatic data tests and a process to tweak existing production tests when new data comes in. And given the complexity of data, we will discuss the need for business-level data testing.

What Exactly Is The Problem? And Why Have We Not Solved It Already?

The data team may sometimes feel its work product is under a microscope. If the data looks “off,” users can often tell immediately. Business users are experts in their domain and will often see problems in data with only a glance. It's generally an unpleasant experience for the data team to learn about analytics errors from its internal and external customers. Finding issues before your internal customers is critically important for the data team. So, what is causing these errors?

Data changes continuously as it flows through the system. Data can drift out of statistical range, defy data ingestion and preparation algorithms, and not meet business requirements. Manual testing of data is performed step-by-step by a person. This process tends to be expensive as it requires a precious resource, such as a data engineer, to execute test queries one at a time. Manual testing can also be prone to human error and is often too cumbersome to run frequently. Checking production data manually or not at all is a recipe for failure. Hoping your customers will not find problems in production is not a strategy. Check data with software, not a person.

Your data providers occasionally give you insufficient and even incorrect or corrupted data. Get used to it. If you do not quickly find these problems in your providers' data and give them feedback, they will continually cause problems. Test your data sources as soon as they are ingested.

Data, especially new data, is a mystery. Data Engineers can't understand the characteristics of every data table in their organization. They can't identify obvious problems in various data sources just by looking at the data. A data engineer may deal with dozens of discrete data sets. They don't have a 'theory of the business' in their head to make judgments on data. Yet the data needs to be validated for use.

Data Teams often have too many things on their 'to-do' list. Customers are asking for new data, people need questions answered, and the tech stack is barely running – data engineers don't have time to create tests. They have a backlog full of new customer features or data requests, and they go to work every day knowing that they won't and can't meet customer expectations.

To understand the time crunch that data engineers face and how it prevents them from testing their data sufficiently, we can think about three categories of data tests, using linguistics as an analogy: data syntax vs. semantics vs. pragmatics. In linguistics, Syntax is the study of sentence structure and grammar rules. While people can do what they want with language (and many often do), syntax helps ordinary language users understand how to organize words to make the most sense.

On the other hand, semantics is the study of the meaning of sentences. The sentence "Colorless green ideas sleep furiously" makes syntactic sense but is meaningless. Pragmatics takes semantics one step further because it studies the meaning of sentences within a particular context.

Finally, data teams must start small and quickly add more complex data tests over time. Data is not static. As their teams grow and data conditions change, they must continually add and modify their existing database profiles and test suites.

Data engineers face another challenge: understanding what tests are needed and configuring the parameters of those tests. That knowledge is often in the heads of business users or Data Stewards. How do you get those individuals involved looped in early? How can you share the responsibility to edit data test definitions and configurations with people who know your business context the best? How can we empower these non-technical people to configure tests?

Data teams today often organize their data warehouse or data lake into layers: L1, L2, and L3. The layers generally refer to different stages of data processing, storage, and access. The naming and specifics of these layers can vary somewhat depending on the context and the specific data warehousing approach, but here is a generalized description:

Data Teams face specific challenges in understanding and testing data in each layer. First, they must ensure that each layer has not inherited lost or mangled data from previous processing. They must also ensure that data properties match business logic in each layer. For example, an application may expect 1 million rows of data to arrive in L1. Via testing, you need to verify that the correct quantity of data arrived in L1 and that the same quantity is present in layers L2 and L3.

Data Engineers adopt various development approaches. Some waterfall-driven teams need precise specifications and shareable test result artifacts to progress through various development phases. Some teams are the opposite; they are in full 'hero mode,' working night and day to add new features and data tests and rapidly fix any problems. Regardless of the approach, both teams must discover data problems rapidly and fix the errors to deliver value to their customers on time. Regardless of the approach, every data team needs a more effective and rapid method to test data.

Data teams are often concerned about running data tests during the production process. Will those tests cause performance problems? Will they cause me to miss my SLA? Data engineers also don't want to duplicate data in other systems just to run tests; they need the data tests to run in the production database quickly and with low impact. They want to understand the test queries clearly and see them in the database execution log so they can quickly take the best action.

Data Teams look, at times, to Data Lineage capabilities to help reduce production errors. Data lineage answers questions like, “Where is this data coming from, and where is it going?” Data Lineage is thus a way to describe the data assets in an organization. It is a description used to help data users understand where data came from and, with a data catalog, the content of specific data tables or files. These represent static blueprints for your data, which are valuable but insufficient. Think of it this way: if your house is on fire, you don’t want to go to town hall and rely on just the blueprints of your home to understand better how the fire could spread. You want detectors in every room so you can be alerted quickly to avoid damage. Data Lineage is the blueprint of the house; Data Testing is the set of fire detectors sending you signals in real time. You need both to provide the complete picture. Data Lineage alone is insufficient to produce error-free data.

It is time-consuming to write data tests from scratch. It’s much less work to refine a small number of pre-coded tests that miss the target. It’s much more work – if not impossible in the real world – to manually create a blanket of relevant tests that can alert you when anything goes wrong. Many data syntax and semantic tests can be configured instead of coded to give you the necessary test coverage. There will always be data tests that are unique to your industry, company, or domain. Wouldn’t you like time to focus on the tests where your team can add unique value?

TestGen produces and executes tests that validate data at rest and in motion during any production process. From data in its raw form to cleaned and prepared data sets, TestGen ensures rigorous testing, providing an extra layer of confidence in your data-driven decision-making.

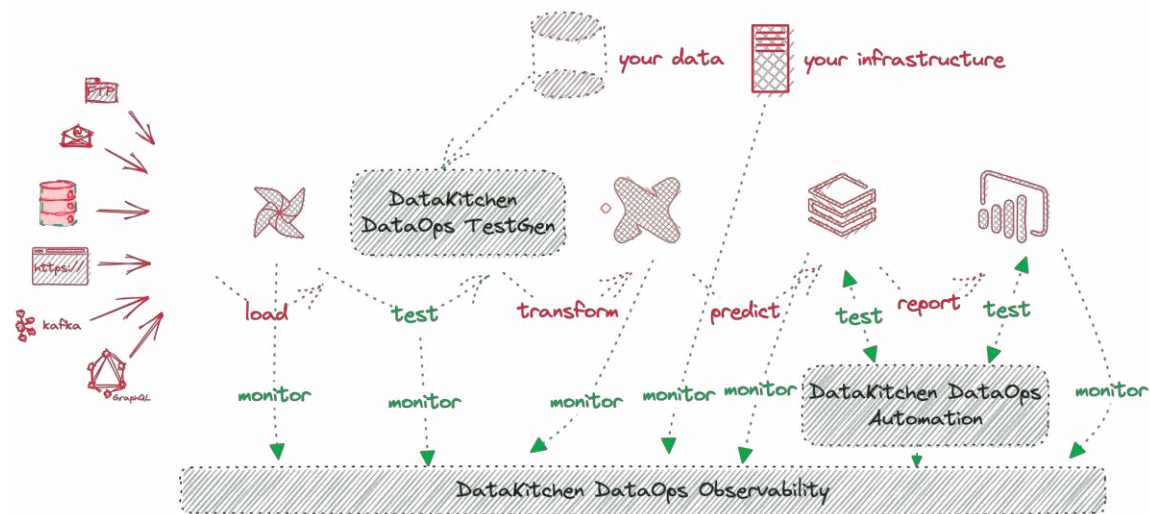
TestGen also offers 11 business rule data tests that, with minimal configuration, can be used for more customized tests. These tests allow users to customize testing protocols to fit specific business requirements with a “fill in the blank” model, offering a perfect blend of speed and robustness in data testing. These types of tests ensure your data not only meets general quality standards but also aligns with your unique business needs and rules. Data stewards, who may know more about the business than a data engineer, can quickly change a setting to adjust the parameters of a data test - without coding.

The ‘Mystery Box Full Of Data Errors’ Is Only One Part Of The “Data Journey”

We call that multi-tool data assembly line a ‘Data Journey.’ And just like data in your database, the data across the Data Journey, and the technologies that make up the Data Journey, all have to be observed, tested, and validated to ensure success. Successful Data Journeys track and monitor all levels of the data stack, from data to tools to servers to code across all critical dimensions. A Data Journey supplies real-time statuses and alerts on start times, processing durations, test results, technology and tool state, and infrastructure conditions, among other metrics. With this information, data teams will know if everything is running on time and without errors and immediately detect the parts that didn’t.

In the illustration above, we see a typical Data Journey and the status of what is working and not working across all of its components. For each component, it’s essential to compare the current state to the expected state, or the Data Journey’s “expectations,” at every step, from source to value delivery across multiple dimensions, including the [Five Pillars Of Data Journeys](#):

For complete ‘Data In Rest’ Data Quality Testing of databases across a Data Journey, two types of tests are needed. The first type includes Data Quality Measurement Tests. These tasks include data profiling, lineage, and ‘bad data’ detection tests. The second type features Active Data Tests for validating data in motion during production. These are operational data tests of data in raw, cleaned, and prepared forms within databases. As described above, DataKitchen’s TestGen product can help you create and execute both kinds of tests automatically with no coding.



To enable Data Quality Testing on 'Data In Use,' a solution should interact with data tools and their outputs and allow for customized, often domain-specific business tests. These include both business rule data tests and custom data tests. DataKitchen offers another test development and execution product, called DataOps Automation, that enables you to build and execute these types of tests quickly and easily.

Conclusion

This white paper discussed the need to automatically generate and execute data quality checks that identify errors in your data before your customer finds them. We discussed the need to test data during the production process directly in the database. We also discussed engineers' various approaches to understanding their data and identifying problematic data records. Finally, we talked about the challenge of enabling data engineers to expand from automatic data tests to a list of configurable 'fill in the blank' data tests.

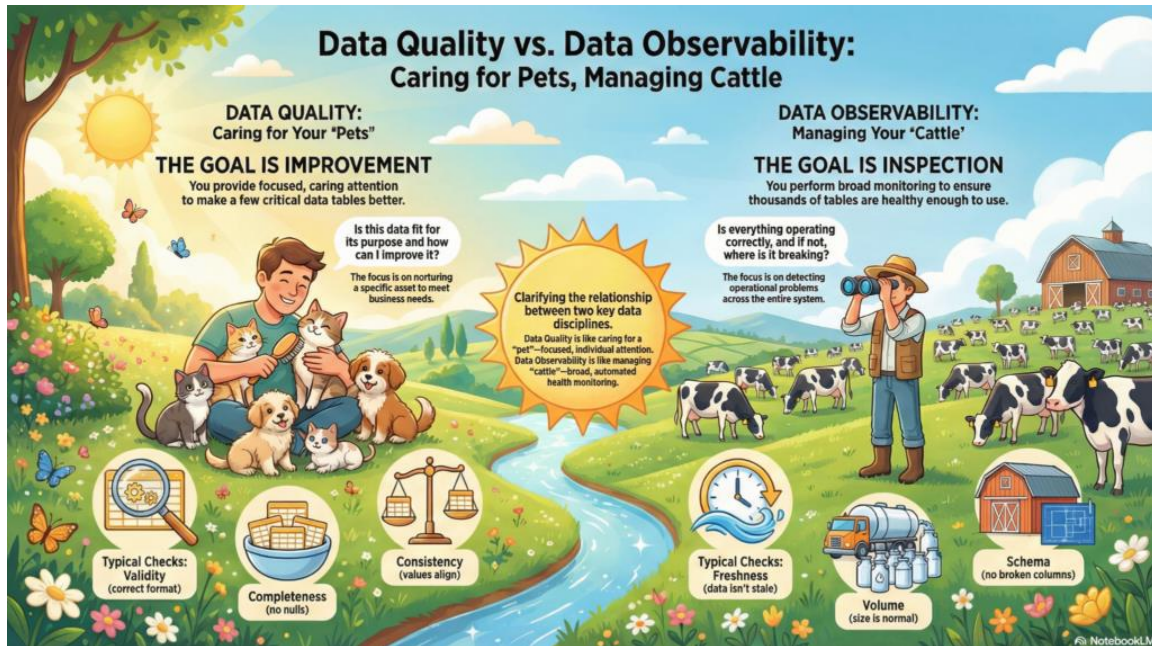
Additional Resources

Original article: <https://docs.google.com/document/d/1deyenEYKxFawDhP4V7Ntin48wCP8zslngL7Ky1c9T1o/export?format=html>

Data Quality vs. Data Observability: The Pets and Cattle of Your Data Estate

The same "pets vs cattle" philosophy that transformed DevOps explains exactly when to invest in data quality dashboards versus when to rely on automated anomaly detection—and why you need both.

If you've spent any time in DevOps circles, you've heard the phrase "[cattle, not pets](#)." Randy Bias popularized this analogy around 2012 to explain a fundamental shift in how we think about infrastructure. Pets are precious, unique systems you know intimately and nurture individually. Cattle are herds you manage collectively—you don't name them or know their personalities, you just need to know they're healthy enough to go to market.



As Bias put it: "In the old way of doing things, we treat our servers like pets, for example, Bob the mail server. If Bob goes down, it's all hands on deck. In the new way, servers are numbered, like cattle in a herd. When one server goes down, it's taken out back, shot, and replaced on the line."

This framework provides a useful lens for understanding two distinct but complementary disciplines that data engineering leaders must master: data quality and data observability. Both aim to increase trust in your data. Both require data quality checks. But they operate at different scales, serve different purposes, and demand different tooling.

Data Quality Is Caring for Your Precious Pets

Your pet tables are the ones you know and love—those lovingly hand-modeled star schemas, business-critical fact tables, and gold-layer marts. You know their names, their purpose, their refresh schedules, and their quirks. Like a cherished family pet, each one gets individual attention. You care about their well-being. You want them to thrive.



Data quality is the pet approach: focused, caring attention on a limited set of tables at a time, with the explicit goal of making them better.

When you have a precious pet, you don't just check that it's alive—you invest in its health and development. You track how it's doing over time. You notice when something's off and take action. You want it to flourish.

Data quality work is the same. You're not just monitoring—you're actively improving the data to ensure it meets the needs of downstream consumers. This requires dashboards that drive action, not vanity metrics that look impressive but change nothing.

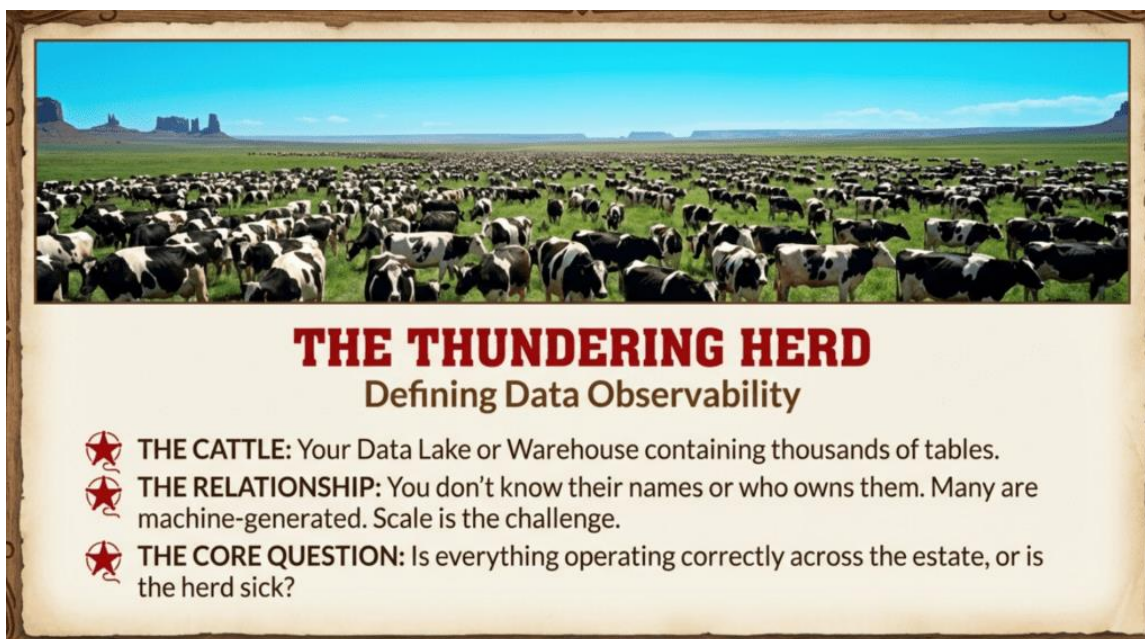
As we explore in our [Masterclass on the Six Types of Data Quality Dashboards](#), effective data quality programs need multiple dashboard types serving different audiences and purposes—from executive scorecards that drive accountability to operational views that help engineers prioritize remediation work. These dashboards track progress over time, connect quality issues to business outcomes, and motivate teams to improve data quality.



The goal isn't just detection—it's improvement. You define what "correct" means, track quality scores over time on dashboards, identify gaps, and drive remediation. You're making your precious pets healthier and stronger.

Data Observability Is Managing Your Cattle

Your cattle is a database—or data warehouse, or lakehouse—with thousands of tables. You don't know them all individually. You can't. No one has time to understand the purpose and semantics of every table in a modern data estate. Many are automatically generated, machine-produced, or simply not owned by any specific person.

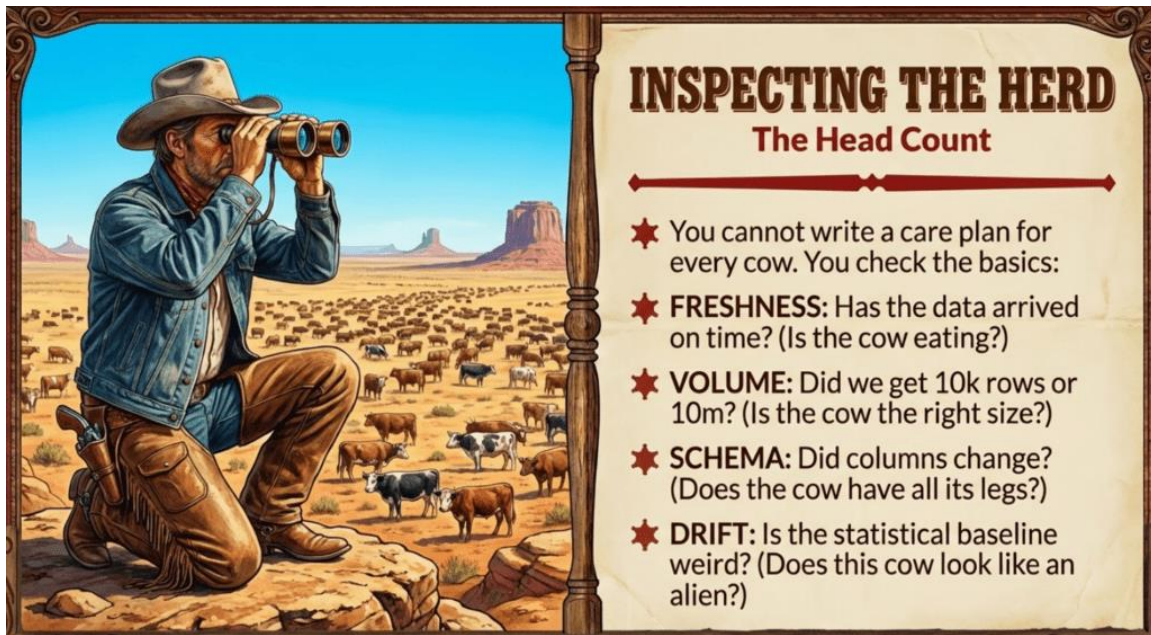


You're not trying to improve each one. You just need to know: are they healthy enough to go to market?

Data observability is the cattle approach: broad monitoring across your entire herd, checking that each animal is big enough, has all its legs, and is ready to be useful.

Data observability asks: Is everything operating correctly across my data estate, and if not, where is it breaking?

When you're running cattle, you don't write a unique care plan for each animal or track its personal development over time. You run them through a chute and check the basics: Are they the right size? Do they have all their legs? Any obvious problems? If they pass inspection, they go to market. If something's wrong, you deal with it.



Data observability works the same way. You're not trying to make each table excellent—you're checking that it meets basic health standards

This is cattle-scale inspection: time series anomaly detection and alerts that tell you when something in the herd isn't right. You're not trying to make each table perfect—you're making sure none of them are broken enough to cause problems when they reach their destination.

Both Disciplines Require Data Quality Tests

Here's the important nuance: both data observability and data quality rely on data quality checks. The difference is how those checks are generated, applied, and acted upon.



In data quality, checks encode business knowledge and specific requirements. You write checks because you understand what “correct” means for this particular table and use case. You care about this pet and want it to improve. When a check fails, it's tracked on a dashboard, scored, and prioritized for remediation.

In data observability, checks are largely automated and pattern-based. You profile data to establish baselines, then generate checks that detect deviation from those baselines. You don't need to understand what each table is for—you just need to know if it's healthy enough to pass through. When a check fails, you get an alert to investigate.

The checks may look similar—both might verify that null rates stay below a threshold—but the context differs. Quality checks ask, “Does this meet our standard for excellence?” Observability checks ask, “Is this healthy enough to go to market?”

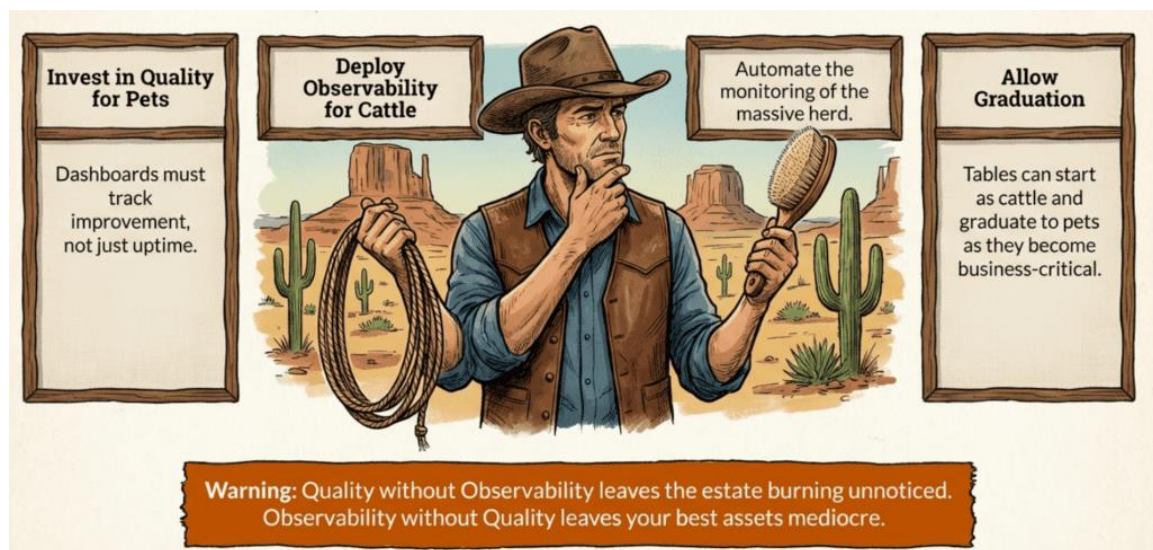
The Operational Models Differ

Quality failures require care and improvement. Someone understands that table's purpose. Someone knows the business impact. The failure isn't just “something changed”—it's “our precious data asset isn't meeting its potential.” The response is root cause analysis, remediation, and tracking improvement over time on your dashboards. You're nursing your pet back to health.

Observability alerts require triage. You may not know why the table exists or who depends on it. Your job is to confirm something actually looks wrong, assess whether it's serious, and either fix it quickly or route it to whoever owns that part of the pipeline. You're not trying to make this table excellent—you're just making sure it's not broken. Pull it out of the herd, deal with the problem, move on.

Building a Unified Strategy

Invest in data quality for your precious pets. Identify the tables that matter most—the ones driving decisions, the ones with clear consumers, the ones you want to make excellent. For these, define what “correct” means, implement comprehensive hygiene and quality checks, and track improvement on dashboards that drive action.



Deploy observability across your cattle. Use automated profiling to monitor all your tables—freshness, volume, schema, and statistical drift. This is your herd inspection. When something fails the basic health check, alerts fire, and you investigate.

Let tables graduate from cattle to pets. Many tables start as cattle—monitored generically through observability because you don't know their purpose. As you learn more about a table's importance, you can promote it by adding specific quality checks, tracking scores on dashboards, assigning ownership, and taking steps to improve it.

Recognize that most tables stay cattle. And that's fine. Not every table deserves the attention you give your pets. Generic observability coverage—checking that they're fresh enough, the right size, and structurally sound—is sufficient for tables that just need to be healthy enough to serve their purpose.

What This Means for Data Engineering Leaders

Understanding the distinction between data quality and data observability helps you allocate resources appropriately:



For your pets: Invest in dashboards that track improvement over time. Define quality standards and measure against them. Build remediation workflows. Make sure quality failures connect to business outcomes so stakeholders care about improving the data. These are your precious assets—treat them accordingly.

For your cattle: Invest in time series anomaly detection and alerting infrastructure. Automate baseline profiling. Build efficient triage workflows. Don't try to write bespoke checks for thousands of tables—just make sure they're healthy enough to pass inspection.

For your organization: Recognize that both disciplines are necessary. Quality without observability means your pet tables are excellent while unknown problems fester across your broader estate. Observability without quality means you catch broken tables but never truly make your critical data assets great.

The Cattle Drive Forward

The distinction between pets and cattle has helped a generation of infrastructure engineers understand modern operations. The same mental model clarifies the relationship between data quality and data observability.



Data quality is caring for your precious pets—focused improvement on specific tables you know and love, tracked through dashboards that drive action and accountability. You want these tables to be excellent.

Data observability is managing your cattle—checking thousands of tables you don't know individually to make sure they're fresh enough, the right size, and have all their parts. You just need them healthy enough to go to market.

Both require data quality checks with complete [test coverage](#). Both contribute to trust. The difference is between nurturing something precious and inspecting a herd.

Master both disciplines, and your data estate—the pets you cherish and the cattle you manage—will deliver the trust your organization needs.

DataKitchen pioneered the DataOps methodology. TestGen is our open-source data quality testing and observability tool that handles both pet-scale data quality and cattle-scale observability. [Use It Now](#) or talk to us about reasonably priced enterprise deployments.

What is the summary of the blog?

Based on the provided text, the main points center on using the DevOps “pets vs. cattle” analogy to distinguish between Data Quality and Data Observability. Here are the core arguments:

The Core Analogy*Pets vs. Cattle: The text adapts a DevOps concept: “pets” are unique systems nurtured individually, while “cattle” are herds managed collectively, with a focus on general health. Application to Data: This framework explains two necessary but different disciplines: Data Quality (pets) and Data Observability (cattle).*

Data Quality: The “Pets” Approach*Focus: This discipline focuses on specific, high-value tables (like star schemas or gold-layer marts) that data engineers know intimately. Goal: The objective is “excellence”—actively improving the data to ensure it is fit for its specific business purpose. Methodology: It relies on bespoke checks defined by business knowledge, such as validating business rules, consistency, completeness, and uniqueness. Operations: When issues arise, the response is “care”—conducting root cause analysis to nurse the asset back to health and using dashboards to track improvement over time.*

Data Observability: The “Cattle” Approach*Focus: This discipline monitors the entire “herd” (database or warehouse), often spanning thousands of tables, both auto-generated and not owned by a specific person. Goal: Verify that the data is “healthy enough to go to market,” rather than making it perfect. Methodology: It uses automated,*

pattern-based monitoring to establish baselines and detect anomalies. Key metrics include freshness, volume, schema changes, and statistical drift. Operations: When alerts fire, the response is “triage”—confirming the break, fixing it quickly, and moving on.

Strategic Recommendations for Leaders Complementary Disciplines: Data engineering leaders must master both approaches because quality without observability leaves the broader estate at risk, while observability without quality fails to maximize the value of critical assets. Resource Allocation: Leaders should invest in remediation workflows and deep checks for “pets,” while investing in alerting infrastructure and efficient triage for “cattle”. Lifecycle Management: Tables can “graduate” from cattle to pets; as a table’s importance becomes clear, it warrants the specific quality checks and ownership associated with the “pet” approach.

Are these the only to uses of data quality tests?

Having adequate data quality test coverage is very important to **four** key activities that every data and analytics team does:

Author: Gil Benghiat | Published: Feb 2, 2026

Original article: <https://datakitchen.io/data-quality-vs-data-observability-the-pets-and-cattle-of-your-data-estate/>

Stop Paying The Data Quality Tax

Why Data Observability costs a fortune, why it shouldn't, and why paying \$100,000/year for commodity algorithms is insane!



Why Data Quality And Data Observability Tools Cost So Much (And Why That's Ridiculous). You're paying enterprise prices for commodity algorithms. Here's what's really going on.

We understand the struggle. Data pipelines fail, dashboards go stale, and someone in finance spots the problem before you do. Suddenly, you're firefighting instead of building. Data observability tools promise to fix this. Sounds great, right?

Stop Paying the Data Quality Tax

Why data observability costs a fortune, why it shouldn't, and how to stop playing the 'VC Math' game.



Data pipelines fail.
Dashboards go stale.
You need observability.

But the price tags are off the rails: \$50,000, \$100,000, sometimes \$1M+ per year.

They've turned a basic necessity into a luxury good.

It's time to ask:
What exactly are you paying for?

For what, exactly? Let's talk about what you're actually paying for, and why the pricing in this market has gone completely off the rails.

Part 1: The Dirty Secret of Data Quality and Data Observability Pricing

Enterprise data quality and data observability vendors have figured out the perfect scam: charge you more every time you try to do the right thing.

Got 10,000 tables? Congratulations. You now need a six-figure budget just to find out whether your data is broken. So you don't test everything. You pick favorites. You cross your fingers. And when bad data sneaks through the tables you couldn't afford to monitor? That's your problem, not theirs.

Data quality isn't a premium feature. It's not a luxury. It's the bare minimum for trusting your analytics, your ML models, your dashboards, your decisions. The idea that teams should ration their data quality coverage like wartime bread is absurd.

It's All Commodity Code Under the Hood

Here's something vendors don't like to discuss: every data observability tool runs on the same basic time-series anomaly detection algorithms, e.g., Facebook's Prophet or ARIMA. Some combination of similar methods. There is no secret sauce. Time-series anomaly detection is still an active area of research in universities, and no vendor has cracked something the academic community hasn't already published.

You Are Paying Enterprise Prices for Commodity Code

There is no secret sauce.



The Dirty Secret: Most data observability tools run on the same standard time-series anomaly detection algorithms (like Facebook's Prophet or ARIMA).

The Reality: Vendors take open-source algorithms, wrap them in a UI, and call it a platform.

The Verdict: When a sales rep claims their 'AI-powered' magic justifies a six-figure bill, they are selling marketing copy, not technical differentiation.

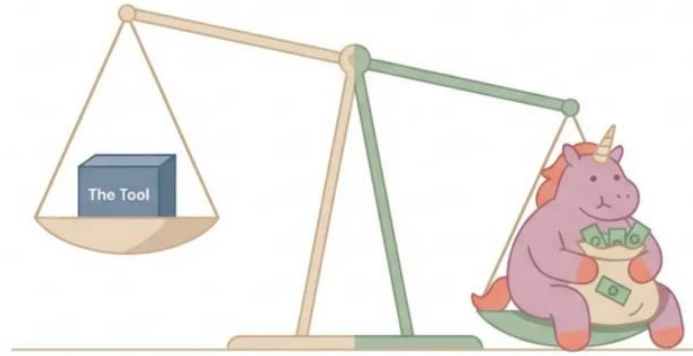
Most of these tools take open-source algorithms, wrap them in a nice UI, and call it a platform. That's the product. That's data observability.

Does it work? Sure, it can catch issues. But when a sales rep tells you their "AI-powered" platform justifies enterprise pricing because of proprietary magic, they're selling you marketing copy, not technical differentiation.

The Real Reason for the Price Tags: Venture Capital Math

If the technology is a commodity, why are prices so high? Because vendors need to charge hundreds of thousands of dollars a year. Their investors expect them to hit \$100 million or more in revenue. The pricing isn't based on cost to serve or value delivered. It's based on what VCs need to see on a spreadsheet.

You Aren't Paying for Value. You're Paying for Their Valuation.



The VC Math

Vendors charge hundreds of thousands because their investors demand \$100M+ revenue targets.

The Trend

Even if you get a discount today, price hikes are inevitable. Just look at dbt and Fivetran.

The Quote

"We are experiencing a huge spike (more than double) in monthly costs... and now with the January 2026 pricing updates." — Reddit User

© NotebookLM

Even if you're getting a discount today, a price hike is coming. Look at what happened with dbt and Fivetran. One Reddit user put it plainly: "We are experiencing a huge spike (more than double) in monthly costs following the March 2025 changes, and now with the January 2026 pricing updates." Every year brings a new price increase, and pay-per-usage models make it worse. Your costs rise automatically as your data grows.

The data engineering community has caught on. As one practitioner warned: "You need to keep an eye on how costs grow as your usage increases." This isn't paranoia. It's pattern recognition from people who've watched their bills double and triple.

Most vendors lure you in with friendly pricing, then jack up rates once your data pipelines depend on them. Just ask a Fivetran or dbt customer what their bill looked like in year one versus year three.

What It's Actually Like to Use These Tools

The gap between the sales pitch and operational reality comes up constantly when practitioners speak honestly. One data engineer shared their experience on Reddit: "My personal experience working at a fintech startup leveraging Monte Carlo is that it has a lot of beneficial features but it's overly complex for use across the organization (the DE team found it useful), prohibitively expensive if you want to cover your entire infrastructure, and takes a dedicated effort (at least a month for us) to get the alerting to a manageable state to avoid alert fatigue."

Think about that. Even at a well-funded fintech, the tool was too expensive to cover the full cost of their infrastructure. The team spent a month tuning alerts to avoid being buried in noise. Only the data engineering team found it useful. Not the analysts, not the business users, not the broader organization that was supposed to benefit from better data quality.

The 'Success Penalty'

Per-table and per-row pricing means that as you get better at data, you get punished with higher bills.



The Usage Trap

Vendors charge per table, per row, or per "credit."

The Math

A small team with 1,000 tables and 500 warehouse tables can face bills of \$150,000/year (e.g., Soda Data pricing models).

The Warning

"You need to keep an eye on how costs grow as your usage increases." — Data Practitioner

Another engineer was more direct: "It's expensive as golden shit. If your company has money to throw around, go with it. If not, there are better ways to spend cash."

These prices create a strategic problem. As one frustrated practitioner explained: "Boiling the ocean: we understand not to implement it slowly, but the DQ tool is priced too high to go slow on value realization. They want a proven main player tool that can be implemented below \$10K and ramps up when we need it. Our VPs would be all ears."

Best practice says start small, prove value, then expand. But these tools are priced for big, enterprise-wide commitments from day one. You can't start small and grow. You either overpay before proving ROI, or you walk away entirely.

Who Actually Needs Six-Figure Observability?

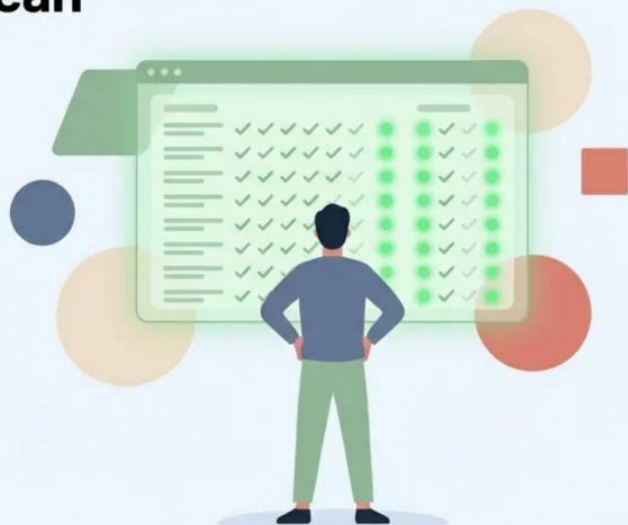
To be fair, there are scenarios where expensive tools might make sense. As one Redditor noted: "If your data team has more than 10ish people and you spend more than 30% of your time on quality issues, it's likely worth investigating."

Stop 'Boiling the Ocean'

The Trap: Competitors force you to make impossible tradeoffs between coverage and cost. You can't "start small" when the entry price is \$50k.

The Freedom: With flat pricing, you stop guessing which tables matter. You stop crossing your fingers. You sleep at night.

Quote: "Our VPs would be all ears for a tool that can be implemented below \$10K."

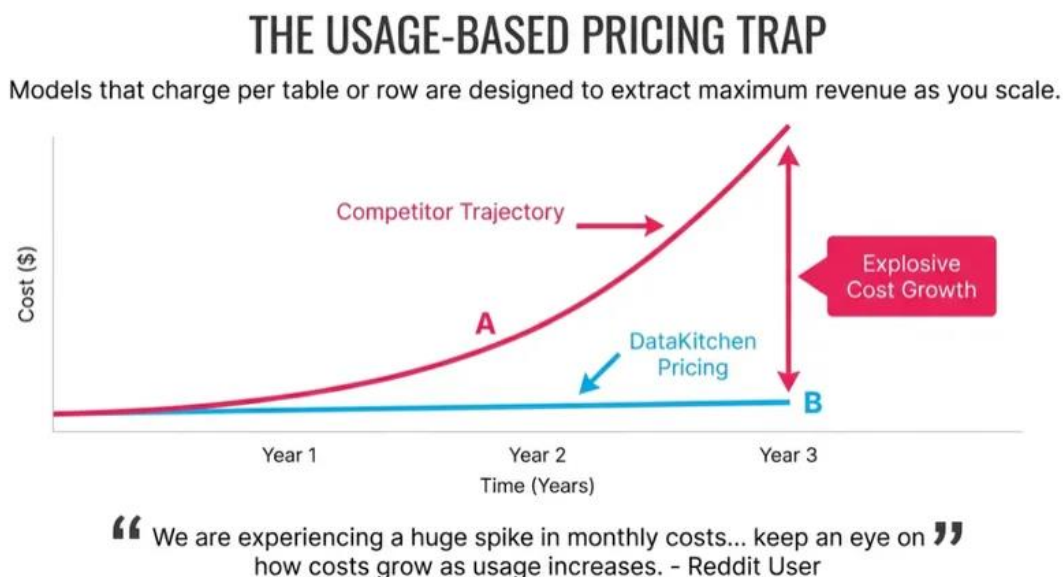


Notice those qualifications: more than ten people, spending over 30% of your time on quality fires. That describes a large team drowning in data problems.

What about the majority of teams? Four to eight people, real quality issues, but no way to justify \$80,000? The current market has essentially abandoned them.

Usage-Based Pricing Is a Disaster Waiting to Happen

One of our customers is a small company with 400 employees and a data team of six people. She has 1,000 tables in her main application and another 500 in a data warehouse. Soda Data would cost her \$750 per month base plus per-table fees that add up to nearly \$12,000 per month. That's over \$150,000 per year. DataKitchen costs her \$9,000.



Why the difference? [Soda just raised a new round of venture capital in April 2024](#). They have investors to satisfy. Prices will rise.

There are more than 50 companies offering data quality data observability tools, backed by over half a billion dollars in investment. They're all waiting for you to buy, then raise prices to hit their growth targets.

With per-table pricing, your bill grows linearly (or worse) with your data. With flat pricing, you can actually afford to monitor everything. You stop guessing which tables matter. You stop crossing your fingers. You sleep at night.

What Data Observability Should Actually Cost

So what's a reasonable price? Not \$100,000 a year. Not a million dollars. For a small team of 4 people with a few databases, it should cost \$0 or at most \$9,000- \$10,000 per year. That price should include support, updates, and a real partnership with your vendor. No extra charges for adding tables. No penalties for running more tests. Everything is included in a flat price.

1 YEAR OF TOOLS = 1 MONTH OF SALARY

Data observability for a team should cost the equivalent of one engineer's monthly salary, not their annual one.

If that sounds radical, it says more about how broken this market is than about us.

You should be able to detect broken pipelines for \$100 per month, not \$100,000 per year. The technology exists. The market just hasn't caught up.

Part 2: A Better Way: DataKitchen's Sane Pricing

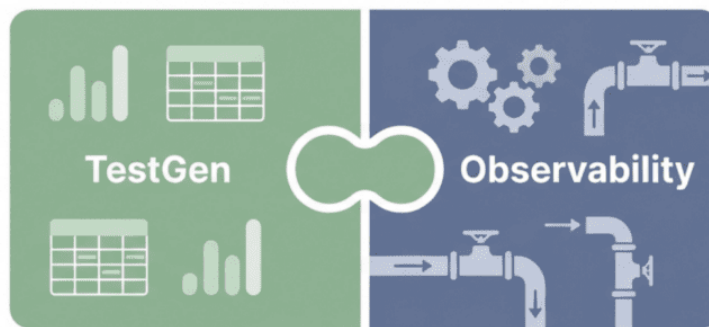
Here's what we believe: data quality and observability for a small team for an entire year should cost the equivalent of one data engineer's monthly salary.

Buy from a profitable, independent company. Don't pay the "VC squeeze." We've been profitable for 12 years. We built our tools from the profits of our data engineering consulting work because we needed something that actually worked. We don't have venture investors demanding growth at all costs, so there's no pressure to raise prices every year.

Two Tools. One Mission: Trust Your Data.

TestGen validates your actual data: schema changes, null rates, duplicates, referential integrity, and statistical anomalies. Observability monitors your pipelines, transformations, and orchestration tools to catch failures before they cascade downstream.

Complete Coverage: TestGen + Observability



1. DataOps Data Quality TestGen

Validates schema changes, null rates, duplicates, and statistical anomalies.
(Is my data correct?)

2. DataOps Observability

Monitors pipelines and orchestration tools to catch failures before they cascade.
(Did the process run properly?)

Both are built on Apache 2.0 open-source cores.

Together, they answer two questions every data team needs answered: Is my data correct? And did the processes that created it run properly?

Pricing That Doesn't Punish You for Scaling

Most vendors structure pricing to extract maximum revenue as you grow. More tables mean more money. More tests mean more money. That model forces you to make impossible tradeoffs between coverage and cost.

That's it. No per-table fees. No per-row fees. No credits that run out at the worst possible time. No "let's schedule a call to discuss pricing."

A team of four people with three databases and unlimited tables pays less than \$9,000 a year. Test every table. Run every check. Your price stays the same whether you have 100 tables or 100,000.

Why We Can Offer This When Competitors Charge 10x More

The answer is simple: DataKitchen has been profitable for 12 years. We developed TestGen and Observability for our consulting work because we needed solutions that were effective and reasonably priced. We don't have venture investors demanding we become a unicorn. We just need to build something useful and run a sustainable business.

Profitable. Independent. Sane.



12 Years Profitable

We built these tools for our own consulting work because we needed something that worked.

No VC Squeeze

We don't need to become a unicorn. We just need to run a sustainable business.

The Result

No pressure to raise prices every year. No 'per-table' fees to juice revenue.

That means no pressure to raise prices every year. No incentive to add per-table fees that punish you for growing. No need to optimize for investor returns instead of customer value.

Built by data engineers, for data engineers. Open-source core. Predictable enterprise pricing when your team needs it.

The Market Is Ready for Change

The data observability market is ripe for disruption because current pricing has lost touch with reality. Open-source alternatives are getting better. Teams are building lightweight in-house monitoring solutions. Sooner or later, someone was going to offer real observability at prices that don't require board approval.

Until the rest of the market catches up, question any vendor who can't explain their pricing simply. Ask about price increases for existing customers. Demand transparent pricing that doesn't penalize you for adding tables or running more tests. And remember what you're actually buying: commodity algorithms with a nice interface. Set your expectations accordingly.

Use TestGen In Production Today

1. Most observability tools use commodity algorithms. There's no technical reason for enterprise pricing. 2. High prices are driven by venture capital math, not actual value or cost to serve. 3. Usage-based pricing is designed to extract

more money as you grow. Flat pricing lets you actually cover your infrastructure.4. A year of data quality should cost a month of a data engineer's salary, not their annual salary.5. Affordable, open-source, low-risk alternatives exist. You don't have to play the enterprise game.

1. Most observability tools use commodity algorithms. There's no technical reason for enterprise pricing.

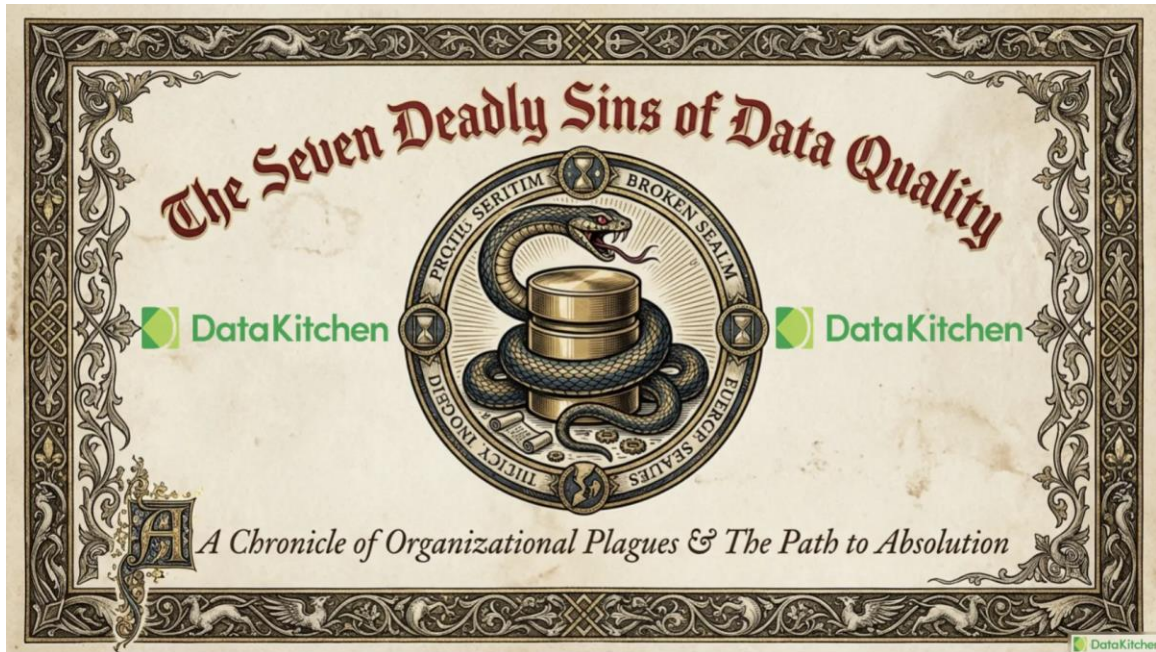
3. Usage-based pricing is designed to extract more money as you grow. Flat pricing lets you actually cover your infrastructure.

Author: Chris Bergh | Published: Feb 5, 2026

Original article: <https://datakitchen.io/stop-paying-the-data-quality-tax/>

The Seven Deadly Sins of Data Quality

Confess your data sins. Find redemption. Maybe even laugh about it.



Data quality problems don't just appear out of nowhere. They fester in organizational cultures where dysfunction becomes normalized, where warning signs are ignored, and where the path of least resistance leads to ever-growing data debt. After years of working with data teams across industries, we've identified seven behavioral patterns that consistently undermine data quality initiatives. Like the classical deadly sins, these aren't simply technical failures—they're deeply human tendencies that prevent teams from achieving the data quality their organizations desperately need.

Understanding these sins is the first step toward overcoming them. Each represents a common trap that even the most well-intentioned data teams fall into. The good news? Recognition is the beginning of recovery, and there are practical paths forward for teams willing to confront these challenges head-on.

1. Shame: The Fear of Finding Problems

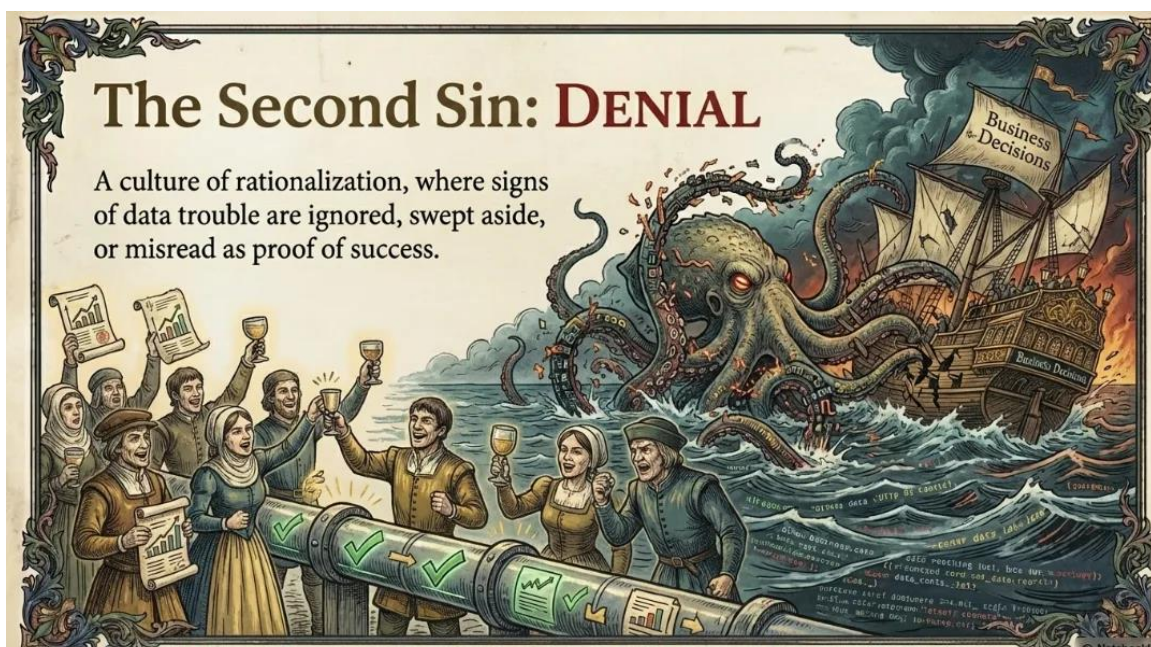
Here's something we don't talk about enough in data engineering: shame. The quiet, corrosive feeling that if something is wrong with the data, it must be your fault. And if you're the one who finds the problem, you're the one who'll have to answer for it. As explored in [The Data Errors Shame Game](#), this shame creates a strange and painful paradox: the very people best positioned to catch data quality issues often develop an unspoken preference not to look too closely. Not because they don't care, but because they've learned that discovering a problem means owning the blame for it.



When data quality responsibility lives entirely within the engineering team, every error feels personal. Engineers are expected to anticipate problems before they happen, catch them instantly when they do, and fix them without anyone noticing there was ever an issue. Over time, this creates a defensive posture: if I don't find the error, I can't be blamed for it. The antidote to shame is shared visibility. When data engineers and business users look at quality problems together, the dynamic shifts from “what did you miss?” to “what should we tackle first?” Building quality circles where stakeholders regularly review data issues together dissolves the isolation that makes shame thrive.

2. Denial: The Culture of Rationalization

In many organizations, data quality problems fester in the shadows—ignored, rationalized, or swept aside with confident-sounding statements that mask deeper dysfunction. As described in [Is Your Team in Denial about Data Quality?](#), teams often build a culture of rationalization where signs of trouble are misread as proof of success. A pipeline ran “all green”? That doesn't mean the data inside was correct. “I haven't heard a single complaint.”? That usually means users have stopped expecting anything better or have worked around the data team entirely.



The woman on the phone saying “The users know about that—they have a workaround!” might be trying to help, but she’s normalizing failure. Workarounds don’t fix broken data; they institutionalize it. Confidence is not a substitute for verification—manual processes, however well-intentioned, are rarely repeatable, scalable, or trustworthy. Breaking the cycle of denial starts with playing “Data Quality Denial Bingo” with your team: printing out common rationalizations and checking off every statement team members have heard—or said themselves—in the past month. Use the laughter and mild embarrassment that follows as a gateway to having a serious conversation about your data quality practices.

3. Avoidance: The Ostrich Mentality

Your team has gone full ostrich. Heads buried deep in the sand, convinced that if they can’t see the production data errors, those errors must not exist. The pipeline is deployed. The tests passed. Job done, right? As examined in [The Ostrich Problem: Your Data Team Thinks Their Job Ends at Deployment](#), this “deploy and forget” mindset is one of the most common—and most corrosive—patterns in data engineering teams. The team sees their job as done once pipeline code is in production. They don’t feel responsible for data quality problems that pop up after deployment



The frustrating part? Your team isn’t lazy or incompetent—they’re just overwhelmed with development work and genuinely don’t realize how bad things have gotten downstream. The ostrich approach feels safe when you’re already buried in sprint work. The fix starts with gathering the facts: survey end customers about their trust in the data, count recent data errors from tickets and Slack threads, and use tools to score your data quality. Share the results with your team. Most engineers don’t want to ship bad data; they just don’t have visibility into what’s happening after deployment. Turn on the lights, and the ostriches will lift their heads.

4. Passivity: Waiting for Permission

Picture this: You’re sitting in yet another meeting where someone asks, “Can we trust this data?” and the room falls silent. If you’re nodding along, you’ve identified yourself as the perfect candidate to become your organization’s data quality champion. As we discussed in [The Data Quality Revolution Starts with You](#), data quality transformation doesn’t begin with company-wide initiatives or sweeping organizational changes. It starts with a single empowered individual who decides that enough is enough—and that person could be you, whether you’re a data engineer debugging pipeline issues at 2 AM or a data leader watching your team struggle.

The Fourth Sin: PASSIVITY

The belief that data quality transformation must begin with a sweeping, company-wide initiative, causing individuals to wait for permission rather than leveraging their personal influence to start.



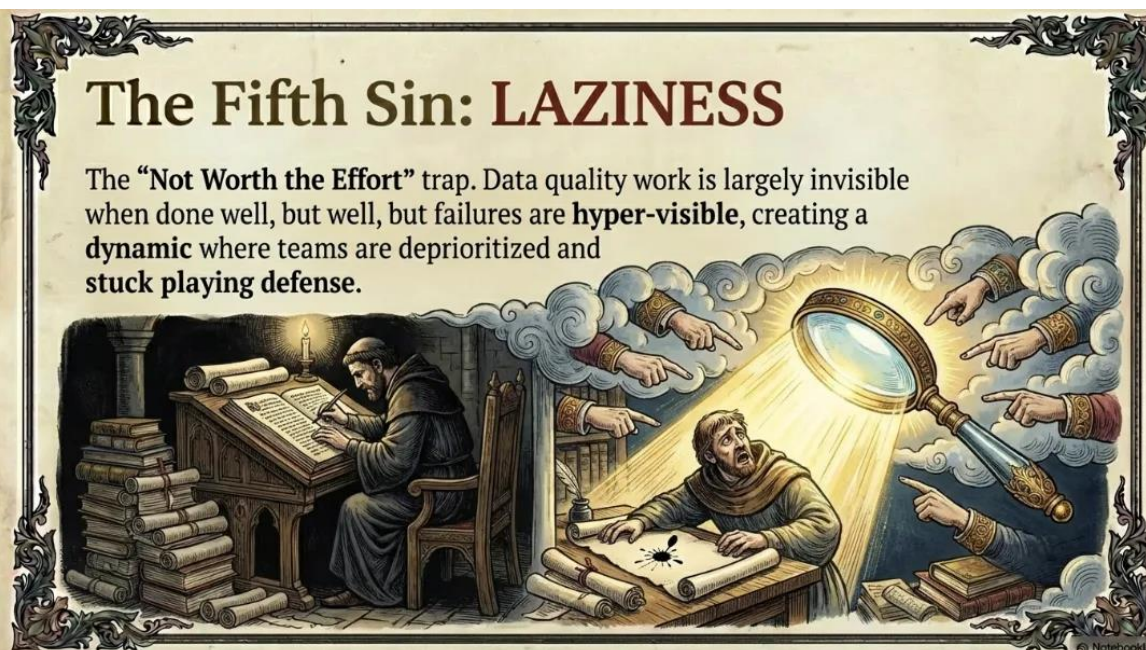
The magic happens when you stop waiting for permission and start leveraging your personal influence. You already have more power than you think. You know which datasets cause the most headaches, which stakeholders ask the toughest questions, and which processes break down most frequently. This insider knowledge is your superpower. Instead of attempting to fix every data problem in your organization, focus on a specific end customer with a specific pain point. Choose one customer, understand their world, and make their life measurably better. When you transform someone's work life by fixing their data headaches, you're recruiting a champion who will advocate for your initiatives.

5. Laziness: The “Not Worth the Effort” Trap

Data quality work is largely invisible. When everything is running smoothly, no one notices the effort behind it. But everyone notices the moment something breaks—when a report is wrong, or a dashboard fails. As discussed in [Why Data Quality Isn't Worth The Effort](#), this creates a dynamic where data teams are expected to prevent problems without being seen, making it hard to advocate for the time and resources required to do the job well. Data quality efforts are often deprioritized because they don't directly generate revenue.

The Fifth Sin: LAZINESS

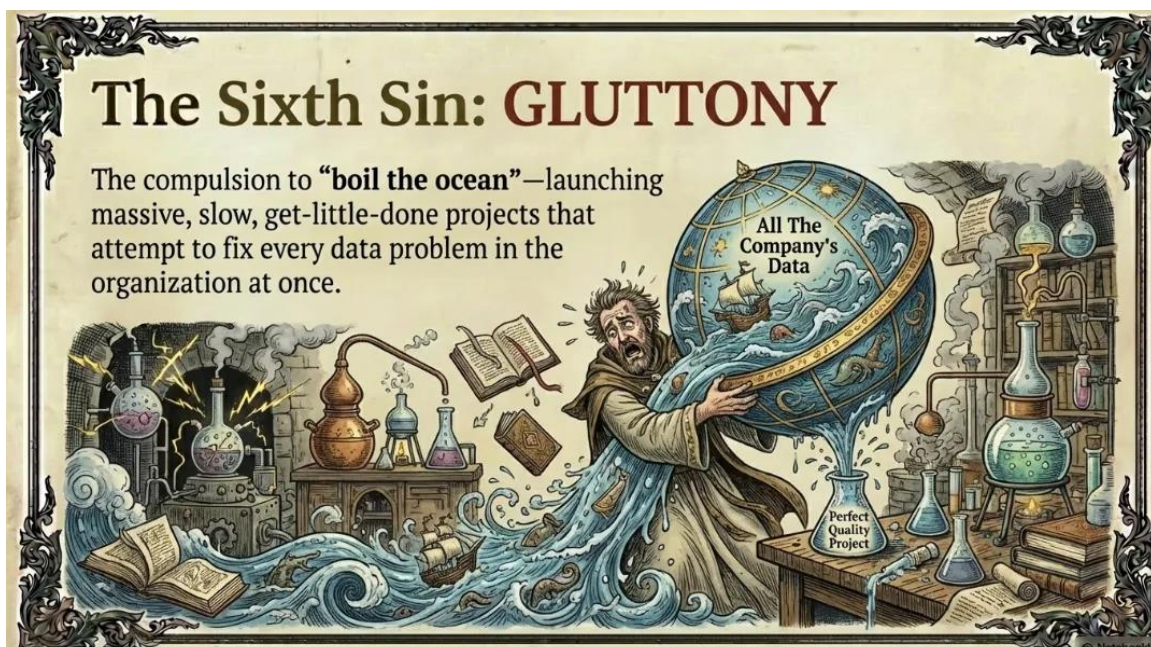
The “Not Worth the Effort” trap. Data quality work is largely invisible when done well, but well, but failures are **hyper-visible**, creating a dynamic where teams are deprioritized and stuck playing defense.



Data environments are constantly changing: new data sources are introduced, pipelines are updated, and business logic evolves. These changes often occur without proper cross-team communication, leaving data engineers and analysts to discover issues only after they've caused damage. The result is teams stuck playing defense, constantly reacting to problems rather than proactively preventing them. The key insight is that automation doesn't have to be overwhelming. Tools that automatically generate tests based on the structure and behavior of the data itself can dramatically simplify the process, shifting data quality from a reactive chore to a proactive, integrated part of the development process.

6. Gluttony: Trying to Boil the Ocean

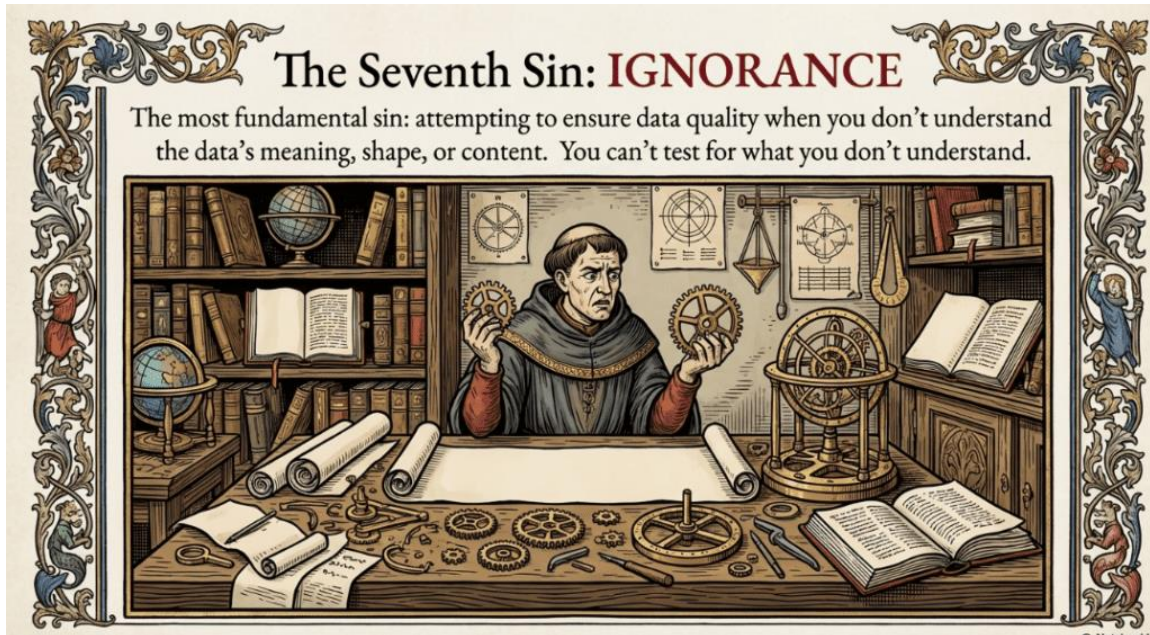
Here's where most data quality initiatives go wrong: they try to boil the ocean. Improving data quality can feel overwhelming—there are so many things to fix and so many processes to improve. Where do you even start? As outlined in [A DataOps Way To Data Quality](#), the answer lies in starting small, iterating quickly, and treating every improvement as a test or experiment. Instead of attempting to fix every data problem in your organization at once, focus on a specific pain point with a specific customer.



This laser focus does two things: it makes success achievable, and it creates visible wins that build momentum for bigger changes. Think of it like renovating a house—you wouldn't tear down every wall at once. Instead, you'd start with one room, perfect the process, then apply those lessons to the next space. DataOps principles provide the roadmap: start small, iterate quickly, launch a simple data quality check, measure its impact, learn from what works, then iterate. This approach feels less intimidating than traditional "big bang" implementations and delivers results faster.

7. Ignorance: Not Understanding Your Data

The most fundamental sin is simply not understanding the data you're responsible for. As explored in [Data Quality When You Don't Understand the Data](#), knowledge gaps—a lack of comprehensive understanding of the data being handled and the business context it serves—can lead to misinterpretations and incorrect data processing. This knowledge deficit can stem from insufficient training, high turnover rates, or poor knowledge transfer practices. When knowledge is concentrated among a few individuals rather than distributed across the team, it creates risky dependencies.



Poor documentation exacerbates this issue, making it difficult for new team members to get up to speed or for others to fill in during absences. In fast-paced environments, teams often prioritize immediate deliverables over long-term quality improvements. This lack of time for collaboration and learning can lead to shortcuts that compromise data quality. The path forward involves building institutional knowledge through data profiling, creating documentation that travels with the data, and establishing quality circles where cross-functional teams regularly discuss data issues and share context.

Acceptance And Recovery.

Understanding these sins is critical because recognition is the first step toward recovery. These behaviors explain why, despite the availability of tools and methodologies, data quality remains a persistent struggle for many organizations.

If the mechanics (data engineers) are afraid to open the hood because they might be blamed for what they find (Shame), or if they ignore a rattling noise because “the radio still works” (Denial), the car will eventually break down on the highway. If they believe their job ends the moment the car leaves the garage (Avoidance) or refuse to perform maintenance because no one applauds an oil change (Laziness), the engine will fail. The Seven Sins are the bad habits of the mechanics that guarantee the car will crash, regardless of how expensive or well-designed the vehicle originally was.

Breaking Free from the Seven Sins

The path from data quality dysfunction to data quality leadership isn’t as long as you think. It starts with one person making one small improvement for one specific customer. From there, it’s about building momentum, measuring impact, and expanding your influence one success at a time. The tools are available—including open-source options that remove budget barriers—the methodologies are proven, and the need is obvious.

The Cure: DataOps TestGen

We built TestGen to be the direct antidote to the seven sins. It's an open-source tool designed for teams looking to break the cycle of shame, surrender, and wishful thinking.

It was created from a simple, powerful belief: you don't have to know everything about your data to start improving its quality. You just have to *start*.



Getting your team to overcome these seven deadly sins isn't about shame or blame—it's about showing them there's a better way. When they can spot data problems before customers do, when the same issues stop breaking things week after week, and when those dreaded stakeholder calls turn into the occasional "hey, nice work"—suddenly, caring about data quality doesn't feel like a burden anymore. It feels like the thing that makes their jobs less stressful. Your organization's data quality transformation is waiting for someone to take the first step.

WANT TO LEARN MORE? Watch the webinar: <https://info.datakitchen.io/webinar-seven-deadly-sins-of-data-quality> Read the companion blog on team 'sins': <https://datakitchen.io/the-five-organizational-sins-of-data-teams/>

WANT TO LEARN MORE? Watch the webinar: <https://info.datakitchen.io/webinar-seven-deadly-sins-of-data-quality>

Read the companion blog on team 'sins': <https://datakitchen.io/the-five-organizational-sins-of-data-teams/>

Author: Chris Bergh | Published: Jan 7, 2026

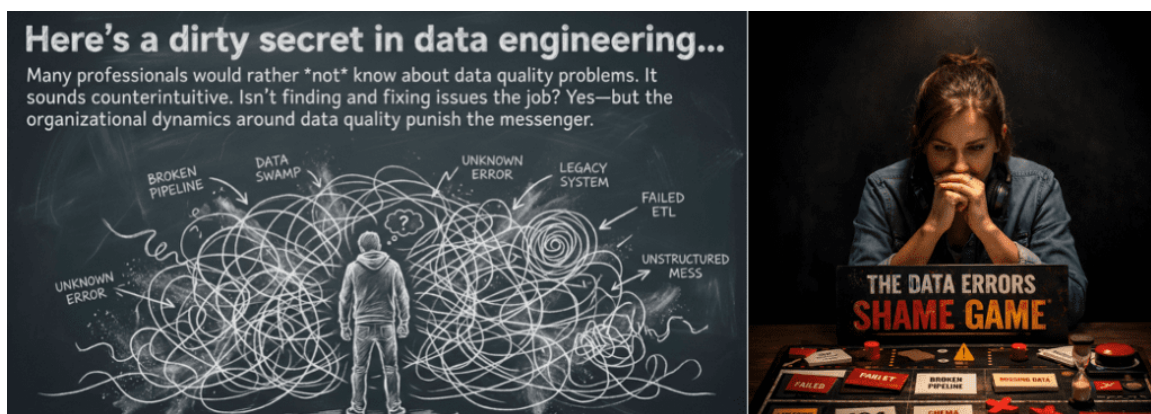
Original article: <https://datakitchen.io/the-seven-deadly-sins-of-data-quality/>

The Data Errors Shame Game: Why Data Engineers Avoid Harsh Truths

Many professionals would rather **not** know about data quality problems. Isn't finding and fixing issues the job? Yes ... but organizational dynamics around data errors punish the messenger. Here's how to fix that dynamic.



Here's something we don't talk about enough in data engineering: shame. The quiet, corrosive feeling that if something is wrong with the data, it must be your fault. And if you're the one who finds the problem, you're the one who'll have to answer for it.

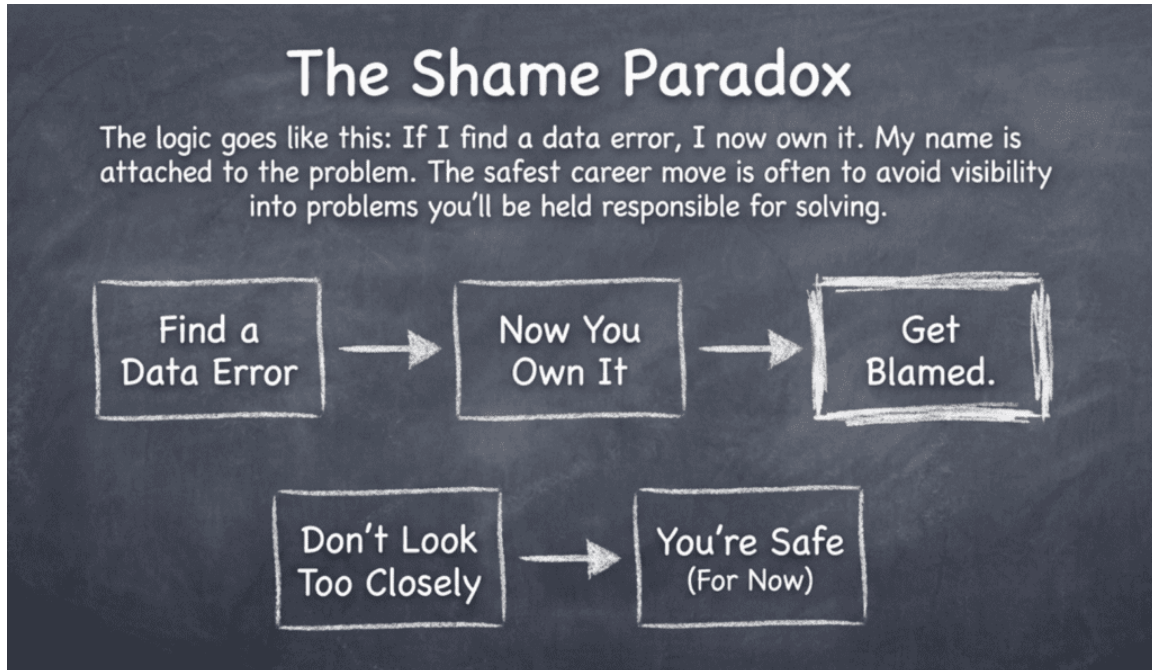


This shame creates a strange and painful paradox. The very people best positioned to catch data quality issues — the data engineers closest to the pipelines — often develop an unspoken preference not to look too closely. Not because they don't care, but because they've learned that discovering a problem means owning the blame for it.

The Real Problem: Engineers Holding the Bag Alone

When data quality responsibility lives entirely within the engineering team, every error feels personal. You're expected to anticipate problems before they happen, catch them instantly when they do, and fix them without anyone noticing there was ever an issue. The business sees clean dashboards and timely reports. They don't see the anxiety underneath.

And when something does slip through? The questions come. Why didn't you catch this? How long has this been happening? The unspoken accusation hangs in the air: you should have known.



Over time, this creates a defensive posture. If I don't find the error, I can't be blamed for it. If someone else discovers it later, at least I'm not the face of the failure. It's not a conscious strategy — it's an emotional survival mechanism. The shame of being blamed is so uncomfortable that avoiding knowledge becomes preferable to seeking it out.

The tragedy is that this makes everything worse. Problems grow in the dark. Small issues become big ones. And when they finally surface — as they always do — the blame lands harder than it would have if someone had felt safe enough to look.

The Fix: Make It a Team Sport

When data engineers and business users look at quality problems together, something remarkable happens. The engineer is no longer the sole bearer of bad news, nervously presenting failures to stakeholders who've been kept in the dark. Instead, everyone is looking at the same reality at the same time. The dynamic shifts from "what did you miss?" to "what should we tackle first?"

Shared Visibility Kills the Blame Reflex.

When a business stakeholder sees the same quality dashboard you do, you're no longer the bearer of bad news. The question stops being "why did *you* let this happen?" and becomes "what should *we* fix first?"



This isn't just a nice idea — it fundamentally changes the emotional texture of the work. When a business analyst sits beside you reviewing a list of anomalies, they bring context you don't have. They know instantly whether a flagged issue is a minor quirk or a decision-breaking catastrophe. They understand which upstream teams need to hear about source data problems. And critically, they become co-owners of the prioritization. The decision about what to fix isn't yours alone to defend.

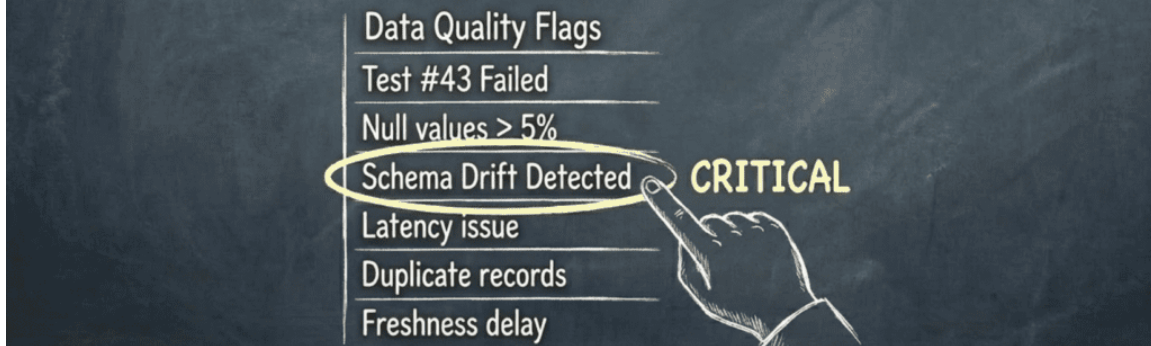
Shared visibility also creates shared accountability in the best sense. When stakeholders see the volume and complexity of data quality issues firsthand, they stop treating fixes as free. They understand the work. They become advocates rather than critics. And when problems originate in systems outside your control, you have allies who can push for fixes at the source — not just downstream workarounds that you'll be blamed for when they're not perfect.

Building a Culture of Collaborative Quality

The practical path forward is more straightforward than it might seem. Start by surfacing a manageable set of quality issues — perhaps fifty or a hundred flagged problems — and invite business stakeholders into a [regular review](#). This could be a weekly half-hour meeting, an async Slack thread, or a shared dashboard where people can comment and discuss. The format matters less than the principle: nobody looks at these problems alone.

Shared Prioritization Ends the Guessing Game.

Engineers often have no idea whether a failed test represents a minor annoyance or a decision-breaking catastrophe. Business users know instantly. This isn't extra work—it's work you were already doing, just with actual information.



In these sessions, we work together to identify what actually matters. Out of a hundred automated flags, maybe ten represent genuine business risk. The engineers can explain what the tests are catching; the business users can explain what the downstream impact would be. Together, you disposition each issue: this one is noise and can be tuned out, this one is critical and needs immediate attention, this one reflects a source system problem that needs to be escalated. Every decision is made collectively and documented transparently.

Over time, these collaborative reviews become a feedback loop that improves the entire system. You learn which tests generate false alarms and which catch real problems. You tune the automated checks to be smarter and quieter. You build institutional knowledge about what quality means for your specific data and your specific business. And perhaps most importantly, you build relationships. The business users who once felt like distant judges become partners who understand and appreciate what you're doing.

Tooling alone won't transform your culture, but the right tools make transformation possible. DataKitchen's TestGen is designed with collaborative quality in mind.

Shared Accountability Creates Allies, Not Adversaries.

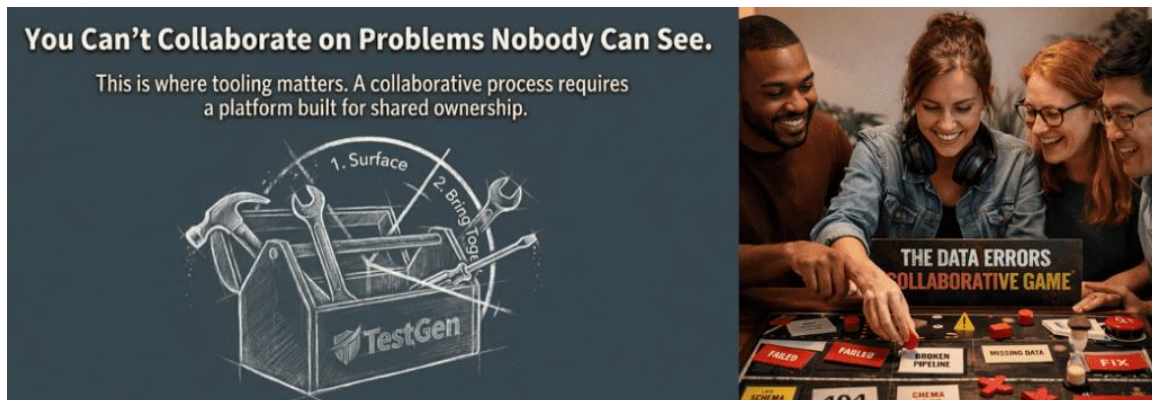
Many data quality problems originate in source systems. When it's just the data engineer complaining, nothing changes. When business stakeholders are visibly affected, there's organizational pressure to fix problems at the source.



The automated test generation means you get comprehensive coverage without having to anticipate every possible failure mode. You discover problems you didn't know to look for, and you're not shamed for missing tests you couldn't have predicted. The stakeholder-accessible dashboards mean business users can see quality status directly, removing the engineer from the uncomfortable role of sole messenger. Disposition workflows give your [quality circle](#) a structured way to review, prioritize, and resolve issues together, with decisions tracked and visible. And audit trails in the enterprise version document the collaborative process — not as surveillance, but as proof that quality decisions were thoughtful and shared.

From Shame to Confidence

The shame cycle in data quality isn't inevitable. It's a symptom of isolation, of engineers bearing impossible expectations alone while the people who could help remain unaware of the challenges.



The antidote is simple in concept and profound in effect: stop hiding the problems, and stop facing them alone. When you build a quality circle that includes both the people who build the data systems and the people who depend on them, you replace shame with solidarity. You replace defensiveness with curiosity. You replace the exhausting performance of perfection with the honest, [hopeful work](#) of continuous improvement.

You don't have to carry this alone. Invite your stakeholders in. Look at the problems together. And discover that the thing you were afraid of — visibility — is actually the thing that sets you free.

Surface a manageable list of issues. Start with 10-20 flagged quality problems — enough to be meaningful, not so many that it's overwhelming.

Bring engineers and business users together. This can be a weekly 30-minute review, an async Slack thread, or a shared dashboard with commenting. The format matters less than the participation.

Collaboratively pick what matters. Of 20 flags, only a few actually impact business decisions. The business users know which one. Without them in the room, engineers are guessing.

Disposition together. “This is noise, ignore it.” “This is critical, fix it now.” “This is a source system problem; escalate it.” Every decision is shared, documented, and defensible.

Learn and improve. Those disposition decisions become training data. Which tests cry wolf? Which catches real problems? Tune the system over time so it gets brighter and quieter.

Shame drives avoidance. Engineers learn that finding problems means owning the blame—so they stop looking.

Isolation makes it worse. When quality is one team's burden alone, every error feels personal and inescapable.

Collaboration breaks the cycle. Shared visibility turns “what did you miss?” into “what should we fix first?”

Build a quality circle. Bring stakeholders into regular reviews. Prioritize together. Document decisions collectively. Replace defensiveness with continuous improvement.

Second, show that adding data quality tests is not particularly time-consuming. Tools like TestGen [automatically generate several dozen data quality tests](#) with the push of a button.

Author: Chris Bergh | Published: Dec 31, 2025

Original article: <https://datakitchen.io/the-data-errors-shame-game-why-data-engineers-avoid-harsh-truths/>

Is Your Team in Denial about Data Quality? Here's How to Tell

Data quality problems are ignored, rationalized, or swept aside. How do you tell if your team is in denial about data quality? Uncle Chip has provided a bingo card ... play with your data team!



*In many organizations, data quality problems fester in the shadows—ignored, rationalized, or swept aside with confident-sounding statements that mask a deeper dysfunction. While the cost of poor data quality is well-documented—wasted time, lost trust, and flawed decisions—teams often operate in collective denial. The problem isn't that they don't care. They've built a **culture of rationalization**, where signs of trouble are misread as proof of success.*

SO MAYBE YOUR TEAM IS IN DENIAL ABOUT DATA QUALITY?

datakitchen.io/uncle-chip



Copyright DataKitchen 2025

Take a closer look at the illustration [Uncle Chip](https://datakitchen.io/uncle-chip) created above. It's a snapshot of common justifications that data teams use to **avoid grappling with the real condition of their data**. These aren't caricatures; they're alarmingly familiar echoes from meetings, Slack threads, and status updates across the industry. A pipeline ran "all green"? That

doesn't mean the data inside was correct. QA passed the code? Great, but that doesn't mean the business logic downstream was valid. "We passed the data exactly as we received it"? That's not a defense—that's a **surrender of responsibility**.

The woman on the phone saying, "The users know about that—they have a workaround!" might be trying to help, but she's **normalizing failure**. Workarounds don't fix broken data; they institutionalize it. When someone claims, "I haven't heard a single complaint," it usually means users have stopped expecting anything better or have worked around the data team entirely. And the engineer proudly pointing to COUNT(*) as validation? That's not testing; that's **wishful thinking**.

One of the most telling lines comes from the self-assured analyst who declares that metrics must be accurate because he calculated them in Excel. **Confidence is not a substitute for verification**. Manual processes, however well-intentioned, are rarely repeatable, scalable, or trustworthy. That's not data quality; that's data folklore.

Denial about data quality doesn't always look like laziness or malice. More often, it's baked into team culture, shaped by tools that don't make quality visible, roles that lack ownership, and a lack of shared language to even talk about what's broken. When success is measured by the **absence of red flags instead of the presence of verified data quality**, teams fool themselves into thinking everything is fine—until the report goes live. Someone important finds a problem in the data, and you sit in a ['war room'](#) all night looking at log files and writing random queries.

Start by playing **Data Quality Denial Bingo** with your team. Please print out the image above or share it during a team meeting. Ask each team member to check off every statement they've heard—or said themselves—in the past month. If someone gets five in a row, they win... or the whole team loses. Use the laughter and mild embarrassment that follows as a gateway to having a serious conversation about your data quality testing practices, data assessment, and data observability. Ask your team what it means to take ownership of data quality.

We built DataOps Data Quality TestGen as the first tool for data teams looking to break the cycle of surrender, failure normalization, wishful thinking, and unverified confidence. Those teams need an **open-source, intelligent data quality tool**. DataKitchen's TestGen lets you start data testing, profiling, and assessing thousands of tables immediately. It's AI-driven, so you can get valuable results in just a few button clicks.

It profiles your data, generates a catalog, **instantly generates hundreds of valuable data quality checks**, and allows you to build data quality assessment dashboards. Interested? Install here: <https://info.datakitchen.io/testgen>

Author: Chip Bloche | Published: May 2, 2025

Original article: <https://datakitchen.io/is-your-team-in-denial-of-data-quality/>

The Ostrich Problem: Your Data Team Thinks Their Job Ends at Deployment.

What to do when your team doesn't care about data errors in production? The "deploy and forget" ostrich mindset is one of the most corrosive patterns in data engineering teams. Here's How to Change That.



Your team has gone full ostrich. Heads buried deep in the sand, convinced that if they can't see the production data errors, those errors must not exist. The pipeline is deployed. The tests passed. Job done, right?

Meanwhile, you're the one fielding angry Slack messages from business users who found garbage in their reports, and you're starting to feel like the only person who actually cares about what happens after code hits production. The frustrating part? Your team isn't lazy or incompetent—they're just overwhelmed with development work and genuinely don't realize how bad things have gotten downstream. The ostrich approach feels safe when you're already buried in sprint work.

You can coax those heads out of the sand without turning it into a big confrontation or piling on more work. It starts with making the invisible impossible to ignore. The "deploy and forget" ostrich mindset is one of the most common—and most corrosive—patterns in data engineering teams. The good news? It's fixable, and it doesn't require a massive cultural overhaul or months of extra work. It starts with making the invisible visible.

Your team sees their job as done once the pipeline code is out in production, with all the unit and manual testing. They don't really feel responsible for any data quality problems that pop up after deployment—you know, in the staged data, dimension marts, or wherever else it's used. Finding that bad data, which can show up every hour or day, is just "not their job" in their eyes.



Your Team Has Gone Full Ostrich.

How to get their heads out of the sand and fix production data errors for good.

Your customers are getting hit with data errors, but the team either doesn't realize how big a deal it is or doesn't seem to care. They basically check out as soon as their deployment hits production. Since they're already swamped with development work, they're really pushing back against taking on any extra tasks. Customers are calling with questions and concerns about the data, which is also taking up team time. Adding data quality tests into the data production process – no one has the time or interest.

You're not going to win this argument with opinions or gut feelings. Your team will push back, insist everything is fine, and point to their passing unit tests as proof. The only way to get those ostrich heads above ground is with cold, hard evidence they can't ignore.

Start by talking to the people who actually use the data every day. Send a quick survey to your end customers—nothing fancy, just a few pointed questions. Do they trust the data? Have they built their own shadow QA processes because they don't trust the data? (Spoiler: they probably have, and they're not happy about it.) Next, do some archaeology on your own side. Count the data errors from the past month. Dig through tickets, bug reports, and Slack threads. Ask your team directly—you might be surprised what they remember once you start asking.

Step 1: Make the Invisible Impossible to Ignore.

Start by gathering the facts that prove there's a problem.



Survey Your Users:

Send a quick poll.
"Do you trust the data?"
"Have you built your own shadow QA processes?"



Count Past Errors:

Do some archaeology. Dig through tickets, bug reports, and Slack threads from the last month.



Get a Baseline Score:

Use an open-source tool like **TestGen** to profile your data and get a quality dashboard running in under an hour. No massive project required.

Present this evidence—survey results, error history, and quality scores—to your team. They don't **want** to ship bad data; they just lack visibility.

Then go a step further and actually score your data quality. Using an open-source tool like [TestGen](#), you can profile your data, get instant [data hygiene test results](#), and have a [data quality dashboard](#) up and running in less than an hour—no massive implementation project, no endless configuration—just real numbers showing the current state of things.

Once you've got all of this—the survey results, the error history, the quality scores, the hygiene test results—share it with your team. Lay it all out. I'm willing to bet they'll be genuinely surprised. Most engineers don't want to ship bad data; they just don't have visibility into what's happening after deployment. This is your chance to turn on the lights.

Here's where the resistance usually kicks in. Your team is already slammed with feature work, and the last thing they want is another "initiative" eating into their sprints. Fair enough. So don't make it a big, massive initiative—make it a small experiment.

Step 2: Demolish the "We Don't Have Time" Excuse.

Resistance kicks in here. The team is slammed. Don't propose a massive initiative; propose a small experiment.

The Plan

1. Pick a few high-pain tables.
2. Add data quality tests and let them run for a few few weeks.
3. Start a bi-weekly "quality circle" to review what the tests and customers found.



Weeks of Coding



Minutes to Generate

The Secret Weapon

Tools like **TestGen** automatically generate several dozen data quality tests with the push of a button, providing 80% test coverage instantly. The other 20% can be built from simple UI templates.

Pick a handful of tables that have been causing the most pain. Add some data quality tests and let them run for a few weeks. Watch what they catch. Then start a [quality circle](#)—a short meeting every two weeks where you review

production errors together. Look at what the tests found, what customers reported, and ask a simple question: Can we create a permanent fix so this doesn't keep happening?

Here's the secret weapon: tools like TestGen [automatically generate several dozen data quality tests](#) with the push of a button. That's not an exaggeration. TestGen handles the [80% test coverage](#) your team needs without anyone writing custom code. The remaining 20%—the tests that are unique to your business logic—can be built using simple templates right in the TestGen UI, often with input from your business partners who know the data best. The “we don't have time” excuse evaporates pretty quickly when you can generate meaningful test coverage in minutes instead of weeks.

This is where the mindset shift happens. A 2022 survey by Wakefield Research found that data engineers spend 40% of their time—two full days per week—firefighting data quality issues. That's not a typo. Nearly half their workweek goes to chasing down bad data instead of building new things. Worse, 74% of data quality issues are first surfaced by stakeholders, meaning most teams are stuck in reactive mode, responding to complaints rather than catching problems before anyone notices.

Step 3: Show Them Testing Isn't a Burden—It's Their Ticket Off the Treadmill

This is the mindset shift. Proactive testing doesn't add to the workload; it eliminates the most painful parts of it.

Without Production Tests	With Production Tests
	
Engineers spend 40% of their time firefighting bad data.	IDC found that improving data quality reduced time spent on defects by 53%.

This is how your team gets back to the **engineering work** they actually want to do.

The math is simple: teams that catch data errors proactively in production spend less time in firefighting mode. IDC research found that improving data quality reduced time spent on defects by 53%. That's when your team gets back to the work they actually want to do.

And the business case extends beyond engineering productivity. Gartner research shows that poor data quality costs organizations an average of \$12.9 million per year—and much of that cost comes from eroded customer trust. According to Accenture, only one-third of executives trust their data enough to derive value from it. When your stakeholders can't trust the numbers, they stop using them.

The flip side? Organizations that invest in data quality see real results. McKinsey research shows companies with mature data governance programs report 15-20% higher operational efficiency. One e-commerce retailer that implemented a data quality initiative saw a 20% reduction in customer churn and a 300% ROI within the first year.

That's the real pitch: caring about production data quality doesn't add to your team's workload. It's the thing that finally gets them off the treadmill.

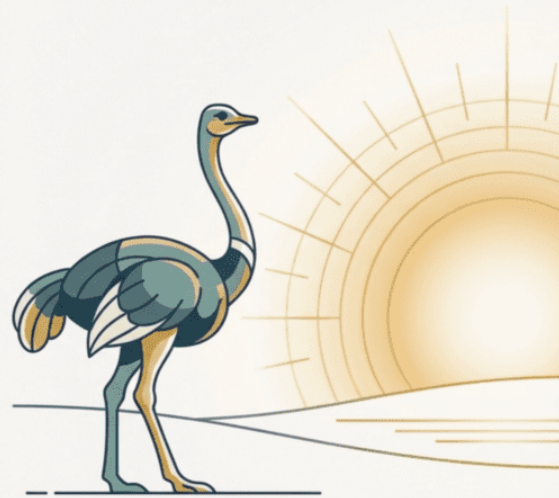
Getting your team to pull their heads out of the sand isn't about shame or blame—it's about showing them there's a better way. When they can spot data problems before customers do, when the same issues stop breaking things week

after week, and when those dreaded stakeholder calls turn into the occasional “hey, nice work”—suddenly, caring about production quality doesn’t feel like a burden anymore. It feels like the thing that makes their jobs less stressful.

They didn’t get into engineering to disappoint people. They just need a reason to look up.

Getting your team to pull their heads out of the sand isn’t about shame or blame—it’s about showing them a better, less stressful way to work. When they can spot problems before customers do, the same issues stop breaking things, and stakeholders start saying “nice work,” caring about quality stops feeling like a chore.

Start small, gather the evidence, and let the results do the convincing.



Start small, gather the evidence, and let the results do the convincing. Your team didn’t get into data engineering to disappoint people. They just need a reason to look up.

Author: Chris Bergh | Published: Dec 19, 2025

Original article: <https://datakitchen.io/the-ostrich-problem-your-data-team-thinks-their-job-ends-at-deployment/>

The Data Quality Revolution Starts with You

BlogCase StudiesWhite PapersWebinars / PodcastsDocumentation



Picture this: You're sitting in yet another meeting where someone asks, "Can we trust this data?" and the room falls silent. Sound familiar? If you're nodding along, congratulations—you've just identified yourself as the perfect candidate to become your organization's data quality champion. And here's the secret that might surprise you: you don't need a massive budget, a team of engineers, or executive approval to start making a real difference.

The Power of One (Empowered Individual)

Data quality transformation doesn't begin with company-wide initiatives or sweeping organizational changes. It starts with a single empowered individual who decides that enough is enough. That person could be you, whether you're a data engineer debugging pipeline issues at 2 AM, a data analyst tired of explaining why the numbers don't add up, or a data leader watching your team struggle with unreliable datasets.

The magic happens when you stop waiting for permission and start leveraging your personal influence. You already have more power than you think. You know which datasets cause the most headaches, which stakeholders ask the toughest questions, and which processes break down most frequently. This insider knowledge is your superpower.

The Data Quality Revolution Starts with One Person (Yes, That's You)



Here's where most data quality initiatives go wrong: they try to boil the ocean. Instead of attempting to fix every data problem in your organization, focus on a specific end customer with a specific pain point. Maybe it's the marketing

team that can't trust their campaign attribution data, or the finance department that spends hours reconciling conflicting revenue reports.

Choose one customer, understand their world, and make their life measurably better. This laser focus does two things: it makes success achievable, and it creates visible wins that build momentum for bigger changes. When Sarah from marketing can finally trust her conversion rates, she becomes your biggest advocate. When the CFO stops questioning every financial report, word spreads quickly.

Embrace the DataOps Way of Life

[DataOps principles](#) aren't just buzzwords—they're your roadmap to sustainable change. Start small, iterate quickly, and treat every improvement as a test or experiment. Launch a simple data quality check, measure its impact, learn from what works (and what doesn't), then iterate. This approach feels less intimidating than traditional "big bang" implementations and delivers results faster.

Think of it like renovating a house. You wouldn't tear down every wall at once. Instead, you'd start with one room, perfect the process, then apply those lessons to the next space. [Data quality improvement](#) works the same way.

Make It Concrete, Make It Count

Vague recommendations like "improve data governance" don't drive change. Specific, concrete remediation actions do. Instead of saying, "The customer data needs work," try "Customer email addresses in the CRM have a 23% invalid format rate, causing 1,200 bounced emails weekly. We need to implement email validation at the point of entry and create a cleanup process for existing records."

This specificity [transforms abstract problems into actionable tasks](#). Your stakeholders suddenly understand exactly what's broken, why it matters, and what needs to happen next. It's the difference between pointing at a symptom and diagnosing the disease.

Measure Everything (And Show Your Impact)

Improving data quality without measurement is merely wishful thinking. Every change you make needs to be continually measured and monitored. This isn't about creating busy work or generating reports that nobody reads. It's about proving value, identifying new problems before they explode, and building credibility for future initiatives.

The beautiful irony is that improving data quality measurement often reveals more data quality issues. This isn't a failure—it's progress. You're moving from unknown unknowns to known unknowns, which is exactly where you want to be.

Here's where the magic happens: showing improvement in data quality over time through charts becomes your most powerful tool for demonstrating real contribution. A simple line graph showing data quality scores climbing from 70% to 95% over six months tells a story that no amount of verbal explanation can match. When stakeholders see that upward trend, they're not just looking at numbers—they're seeing the tangible impact of your work. These visual proof points transform you from someone who "works with data" into someone who "delivers measurable business value."

Gather Your Allies

A significant challenge is, "How do I deal with all this data at scale?" You're drowning in datasets, struggling to understand what you have, and wondering how to prioritize your efforts when everything seems broken.

But most importantly, you should focus on getting allies., "How do I influence change to improve data quality?" You've identified problems, but getting people to care and take action feels like pushing a boulder uphill. You know what needs to be fixed, but creating organizational momentum feels impossible. The secret here is finding your allies in making change. Pick a good customer for change, focus on solving their specific problem, and they will become part of

your data quality army. When you transform someone's work life by fixing their data headaches, you're not just solving a technical problem—you're recruiting a champion who will advocate for your initiatives and help you influence others.

Your Data Quality Playbook

Ready to become your organization's data quality champion? Here's your action plan, distilled into five powerful strategies that you can start implementing today.

Get Going Quickly means embracing imperfection over inaction. Don't wait for the perfect tool, the ideal dataset, or organizational alignment. Pick one small problem and solve it this week. Success breeds success, and momentum matters more than perfection in the early stages of growth.

Focus on Specific Customer Needs transforms abstract data problems into human stories. Find one person whose work day gets measurably better when you fix a specific data issue. That person becomes your case study, your advocate, and your proof point for future initiatives.

Pick The Right Type Of [Data Quality Dashboard](#) recognizes that not all customers are created equal. Your executives require diverse perspectives from your data engineers. Your business stakeholders care about different metrics than your technical teams. Match your dashboard to your audience's needs and decision-making style.

Iterate, Improve, Influence acknowledges that data quality improvement is a cycle, not a destination. Every fix teaches you something new. Every success builds credibility. Every failure (yes, there will be failures) provides valuable learning that makes your next attempt stronger.

Give Concrete Actions moves beyond problem identification into solution implementation. Abstract recommendations gather dust in email threads. Specific, actionable steps with clear owners and deadlines create real change. Make it easy for the person doing the fix.

Your Data Quality Uprising Starts Now

The path from data quality frustration to data quality leadership isn't as long as you think. It starts with one person (you) making one small improvement for one specific customer. From there, it's about building momentum, measuring impact, and expanding your influence one success at a time.

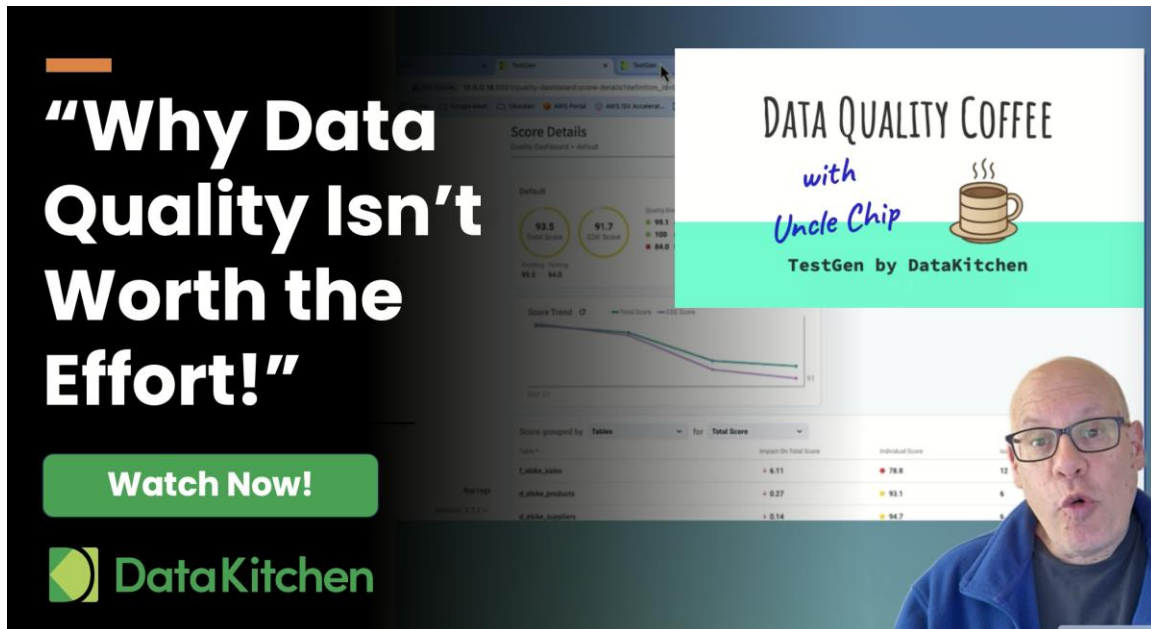
Your organization's data quality transformation is waiting for someone to take the first step. The tools are available, the methodologies are proven, and the need is obvious. The only question remaining is whether you're ready to be the empowered individual who makes it happen.

Author: Chris Bergh | Published: Jun 20, 2025

Original article: <https://datakitchen.io/the-data-quality-revolution-starts-with-you/>

Why Data Quality Isn't Worth The Effort: Data Quality Coffee With Uncle Chip #1

Data quality work is largely invisible. When everything is running smoothly, no one notices the effort behind it. But everyone notices the moment something breaks!



Data quality has become one of the most discussed challenges in modern data teams, yet it remains one of the most thankless and frustrating responsibilities. In the first of the ‘Data Quality Coffee With Uncle Chip’ series, he highlights the persistent tension between the need for clean, reliable data and its overwhelming complexity. For many data teams, maintaining data quality feels like an uphill battle that often goes unrecognized and unrewarded.

The issue’s core is that data quality work is largely invisible. When everything is running smoothly, no one notices the effort behind it. But everyone notices the moment something breaks—when a report is wrong, or a dashboard fails. This creates a dynamic where data teams are expected to prevent problems without being seen, making it hard to advocate for the time and resources required to do the job well.

Uncle Chip explains how data environments are constantly changing. New data sources are introduced, pipelines are updated, and business logic evolves. These changes often happen without proper communication across teams, leaving data engineers and analysts to discover issues only after they’ve caused damage. As a result, data teams are stuck playing defense, constantly reacting to problems rather than proactively preventing them.

Automation and monitoring tools can help but are not a silver bullet. Implementing these solutions still requires significant effort—setting up tests, defining thresholds, and maintaining alerting systems. Even with the best tools in place, a human layer of interpretation is still needed to understand what’s going wrong and how to fix it.

One of the most frustrating realities is that data quality efforts are often de-prioritized because they don’t directly generate revenue. Unlike shipping a new product feature, cleaning up a broken pipeline or fixing a data inconsistency has no immediate, visible impact. This leads to a lack of investment in foundational data health and a reliance on quick fixes that only temporarily patch deeper issues.

Ultimately, Uncle Chip underscores a shared experience across data teams: the feeling that data quality is essential but always treated as secondary. It’s a constant grind that demands vigilance, cross-functional alignment, and an

appreciation for work that, when done well, looks like nothing at all. For many teams, it's not that they don't care about data quality—it's just too much work with too little support.

TestGen helps reduce the overwhelming burden of data quality work by automating one of the most tedious and time-consuming parts: writing tests. In the traditional workflow, data engineers and analysts must manually define what “good” data looks like, write SQL or code-based tests to check for issues, and continuously maintain those tests as schemas, pipelines, and business rules change. This manual process is slow, error-prone, and often neglected in favor of more urgent tasks.

TestGen simplifies this by automatically generating tests based on the structure and behavior of the data itself. Instead of starting from scratch, data teams can use TestGen to instantly produce a baseline set of quality checks—null checks, value ranges, unique constraints, and expected distributions. These tests can be generated from existing tables, models, or past data behavior, giving teams a fast and scalable way to implement monitoring with minimal effort.

The real advantage is that TestGen shifts data quality from a reactive chore to a more proactive and integrated part of the development process. By making it easier to create and maintain tests, teams can catch issues earlier, reduce firefighting, and spend more time delivering insights rather than fixing broken pipelines. It also helps standardize data quality practices across teams, reducing tribal knowledge and improving collaboration.

In short, TestGen doesn't magically solve data quality—but it lowers the barrier to doing it well. It transforms what used to be an “extra task” into something teams can keep up with, which is precisely what's needed when data quality feels like just too much work.



Watch The 'Data Quality Coffee With Uncle Chip' Series

Author: Chip Bloche | Published: Apr 9, 2025

Original article: <https://datakitchen.io/why-data-quality-isnt-worth-the-effort/>



DataKitchen

Data quality is more critical than ever in the age of artificial intelligence (AI) and when organizations are 'competing with data.' As teams handle increasingly vast amounts of data, the demand for accuracy, consistency, and reliability in data sources has risen. Despite advancements in data management technologies, a significant gap persists in ensuring data quality. According to a 2024 survey by dbt Labs, 57% of respondents rated data quality as one of the top three most challenging aspects of the data preparation process—a sharp increase from 41% in prior years. This statistic highlights an important reality: data quality is not merely an operational problem but a strategic imperative.

Moreover, IDC reports that 73% of data practitioners do not fully trust their data, signaling deep-seated issues with the reliability of organizational data. This mistrust directly impacts decision-making processes across departments. Forrester's research adds another layer of urgency, estimating that millions were lost in 2023 due to poor data quality, with the potential for these losses to escalate into billions as AI becomes more integral to business operations. Without a concerted intervention, the symbiotic relationship between AI and data quality could become a global liability for organizations.

The severity and scale of these problems indicate that the traditional methods of addressing data quality must be revised. Many organizations have approached data quality issues through failed initiatives—avoiding fixing problems or through long, slow, get-little-done projects. However, data quality challenges are not static; they are dynamic and continuous, simultaneously impacting multiple aspects of an organization. Addressing these data quality challenges effectively requires more than avoidance or slow bureaucracy. It calls for an ongoing, adaptive methodology that can evolve with the data itself—a methodology found in DataOps.

Why a DataOps Approach to Data Quality: Leadership Challenges

While identifying and understanding data quality issues is crucial, data quality leaders face a different, more nuanced challenge: driving change at scale across their organizations. In many instances, the people responsible for the data quality often differ from those responsible for fixing it. Data quality problems often span multiple departments, crossing into different business units and functional areas. This creates an environment where achieving widespread improvement becomes incredibly complex, as it requires cooperation from a diverse range of stakeholders, many of whom might not see the immediate benefit of investing in data quality improvements.

One of the core issues faced by data quality leaders is that source system data is often deemed "good enough" for the operational systems where it originates. However, this data frequently needs to be revised when repurposed for analytical use or downstream processes. Satisfied with the performance of their data in its original context, source system owners often do not prioritize or even recognize the need for data quality improvements beyond their immediate domain. This scenario creates friction between those who rely on data for analysis and decision-making and those who maintain the systems generating the data.

Compounding this problem is that data quality leadership is often a role of influence rather than direct control. Data quality leaders frequently identify problems but need more authority, resources, or technical capacity to implement solutions directly. They must rely on others—teams and individuals who control the data sources—to execute changes. This dependence on others makes driving action a formidable challenge, as it involves convincing people across the organization to prioritize data quality improvements. The questions that data quality leaders must grapple with are not just technical but are social:



These questions underscore the leadership challenge at the heart of data quality improvement. More is needed to identify and understand the problem; data quality leaders must act as influencers, facilitators, and catalysts for change. The solution to this challenge lies in creating a structured, repeatable, and agile process that encourages continuous improvement and aligns the interests of all stakeholders—a solution that DataOps provides.

Critical Ideas in Data Quality The DataOps Way

The DataOps approach to data quality offers a compelling framework for addressing the leadership challenges and systemic issues outlined above. Drawing inspiration from Agile, Lean, and DevOps practices, DataOps applies these principles to data and analytics. It shifts the focus from a reactive, one-off project mindset to a proactive, iterative, continuous improvement process. At its core, DataOps enables individuals and teams to rapidly, collaboratively, and at scale manage and improve data quality. It encourages individuals to start small, iterate quickly, measure success, and ‘ship’ working results early and often.

DataOps emphasizes individual empowerment, recognizing that the people closest to the data are often best positioned to identify and address quality issues. Rather than waiting for top-down mandates, DataOps encourages organizations to equip individuals with the tools, autonomy, and authority to drive change within their respective domains. This empowerment enables faster problem identification and more agile responses.

In the DataOps approach, experimentation and iteration are central. A perfect data quality score or test is optional before taking action. Instead, teams are encouraged to start small, experiment with solutions, and iterate based on feedback and results. This iterative approach allows organizations to make incremental improvements quickly, learning from each cycle and refining their processes over time.

One of the critical principles of DataOps is leverage—the ability to maximize the impact of limited resources. In many cases, data quality improvements can be achieved not by scaling resources but by identifying the right leverage points within the organization. This could involve automating repetitive tasks, applying AI-driven insights, or tapping into the knowledge of subject matter experts at critical moments. Data Quality leaders should use scoring models tailored to distinct business needs to make data scoring more focused and impactful. This ensures scores are relevant to different departments or use cases, allowing data quality efforts to drive targeted improvements.

DataOps draws heavily from Agile methodologies, prioritizing adaptability, collaboration, and rapid iteration. A core tenet of DataOps is to "get a data quality process working right away" and then iterate and improve on it continuously. In practice, this means starting with a minimum viable data quality evaluation and scoring method that addresses the most immediate issues and then building on that foundation as more information becomes available. Documentation and analysis are seen not as obstacles to progress but as byproducts of the improvement process.

DataOps encourages organizations to measure and evaluate data quality before establishing formal standards. Early measurements provide a baseline against which future improvements can be gauged. Over time, these measurements help inform the creation of data quality standards, which can then be refined and improved through further iterations. By starting with measurable outcomes, data quality leaders can make data-driven decisions about where to focus their efforts and how to prioritize improvements.

*[G]ood quality and the proper uniformity have no meaning except with reference to the consumer's demands." Deming, W. Edwards. (1953). [*Statistical Techniques and International Trade*](#). *Journal of Marketing*, 17(4), 428-433. Sage Publications, Inc., page 428*

The DataOps approach is predicated on continuous learning. As teams cycle quickly through improvements, they learn more about their data, processes, and organizational needs. This rapid feedback loop enables teams to adapt quickly to changing requirements and evolving data landscapes, ensuring that the organization remains nimble in the face of new challenges.

Where to Start with Data Quality The DataOps Way

The first step in implementing DataOps for data quality is to start small, focusing on individual empowerment and personal leverage. Instead of attempting a large-scale, organization-wide rollout, DataOps Data Quality encourages organizations to begin with a single person who can gain influence. Still, it may need more authority over the data. This individual should be given a tool that enables them to identify meaningful data quality improvements with minimal ramp-up and at no significant cost.

Organizations can quickly see the benefits of data quality by starting small. This approach allows for learning and improvement over time, as each iteration provides valuable feedback that informs future efforts. Automation and Data Quality AI can play a significant role in this process, helping to streamline testing and analysis while reducing the manual burden on individuals. The goal is to make it easy to achieve impact, fit the solution to the organization's specific needs, and measure success along the way.

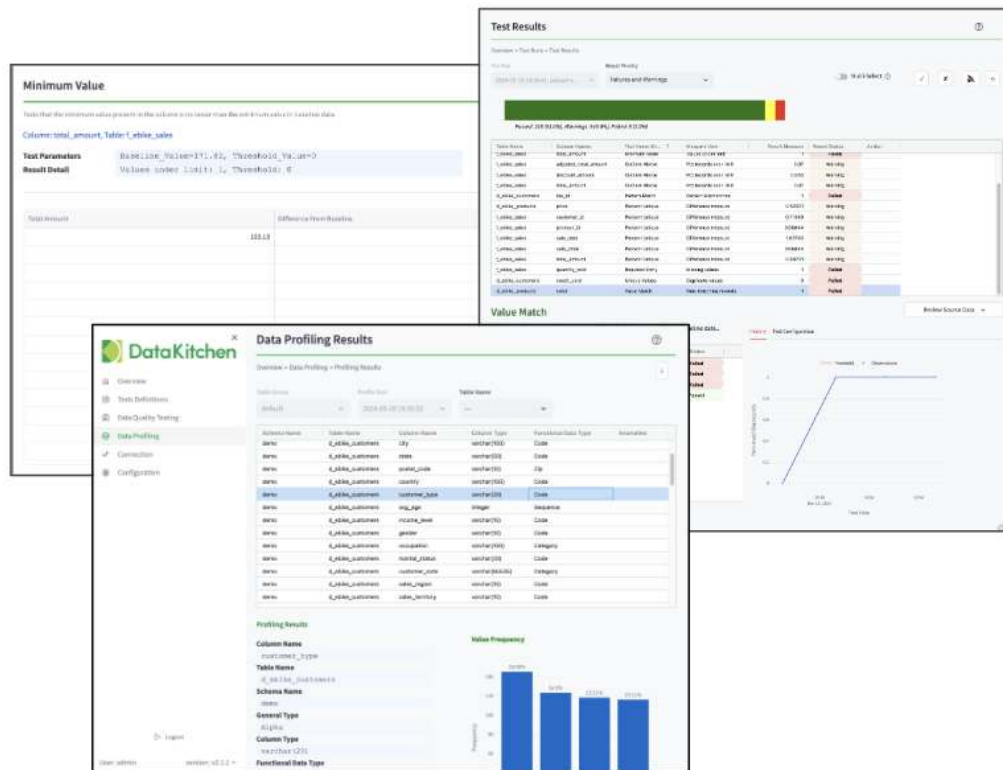
How to Do Data Quality The DataOps Way – Seven Steps

The following outlines seven steps to do Data Quality the DataOps way. It focuses on empowering one person with no power but has influence and enables them to start the virtuous Data Quality influence, action, and measure cycle immediately.



Finding individuals within the organization who are motivated to improve data quality is essential. These individuals, often tired of being the “data quality nags,” can play a pivotal role in driving change. Organizations can harness their passion and expertise to make meaningful improvements by giving them the necessary tools to influence.

The first step in the DataOps approach is to empower an individual who may not have direct authority but influence within the organization. This person should understand the importance of data quality and be motivated to drive change, even if they do not control the systems generating the data. Organizations can start improving data quality from the ground up by giving these individuals the tools and resources they need to effect change. To enable these individuals to start making improvements quickly, providing a tool that is easy to install and use is crucial. This tool should not require extensive IT support or additional infrastructure, allowing individuals to begin improving data quality within hours. By making the process as simple as possible, organizations can remove barriers to entry and accelerate the path to improvement.



Data profiling is a critical step in understanding the quality of data sources. It visually represents the data and checks for conformance with business rules. Profiling helps visualize data problems, such as missing values, inconsistencies, or patterns, and serves as an excellent starting point for data quality workshops. By enabling fact-based discussions about the causes and impacts of data problems, profiling helps foster collaboration and drive solutions.

Once data has been profiled, the next step is identifying specific data quality issues. Identifying data quality issues is critical to ensure that datasets are reliable, consistent, and ready for analysis. Data is complicated; coding hundreds of data quality tests for every table is time-consuming and never gets done. Data Quality leaders need an accelerator that automatically creates data quality tests. They need to manage and build company-specific tests with ease.

One major category of issues involves data hygiene problems. These hygiene issues can often be detected and mitigated through qualification and hygiene screening. Regularly conducting a periodic review of source data is essential to keep track of any degradation in data quality over time. Additionally, it's essential to surface data hygiene requirements early on to establish data cleanliness and reliability expectations. These requirements should be well-documented for collaborative review, ensuring that teams are aligned on the standards and any updates or concerns are shared.

A few example tests can help with hygiene management. These include confirming data typing, such as checking for numbers appearing in columns intended for text (alpha columns) and checking blank value representation to ensure no critical data is missing. It's also necessary to verify that distinct values are present where applicable and assess any string pattern inconsistencies that might suggest data formatting issues.

In addition to detecting hygiene issues, it is possible to automatically generate data quality scores based on data variations. One method for this is consistency testing, where anomalies from the baseline are identified. Baseline characteristics can be derived from data profiling, enabling teams to infer rules and documentation that establish acceptable data patterns automatically. These rules can then lead to the automatic generation of parameterized tests, providing a consistent mechanism for data validation over time.

Anomalies can arise in several forms. For instance, freshness anomaly checking ensures that the dataset is up-to-date, verifying that new data has been received. Similarly, volume anomaly checking confirms that the correct amount of data is being collected, while schema anomaly detection checks if the incoming data conforms to the expected schema. Data drift checks assess whether the received data makes sense compared to previous datasets. Deploying a wide net of simple tests—such as validating required values, calculating missing value percentages, and checking uniqueness—can help ensure data quality. These tests can also include list-of-values checks against baseline values, alpha and decimal truncation evaluations, and verifying that data recency is within acceptable ranges. Additional tests might involve comparing record counts within specific time windows, confirming min/max values against the baseline, and assessing distribution shifts over time.

Configurable business rules can also help address data quality concerns. These rules allow for tests that fit single-table and multi-table structures. Business rule testing leverages the knowledge of domain experts to establish accurate data validation rules. Once set, these rules double as documentation for the dataset. The system can then run parameterized tests to verify compliance with these rules, ensuring more signal and less noise in the results. Importantly, these evaluations should focus on the data in a stable state rather than in the middle of a batch process, as this ensures more accurate results.

Several example tests can be applied using business rules. For instance, a test might validate that list-of-values data aligns with expected entries, such as checking if a delivery status field contains only acceptable values. Another example could be ensuring that entity relationships are enforced, such as verifying that every SKU exists in the distributor list. Additional tests could confirm the absence of duplicate values in unique fields or perform referential integrity checks across related tables.

For more specific scenarios, custom data quality tests can be created. These tests can be tailored to specific use cases by allowing users to supply the SQL queries needed to validate the data. Vertical domains and industries often have unique requirements, making it necessary to design industry-specific tests. For example, in the healthcare industry, a rule might state that the number of medical practices should not exceed the number of doctors, or in logistics, it could be required that no shipments are made on Fridays. By reserving programming expertise for these complex, custom challenges, organizations can maximize the value of their technical resources without overburdening them with routine checks.

Finally, managing the people involved in ensuring data quality is critical to a successful data governance strategy. This can be done through a user interface (UI) or a command-line interface (CLI), allowing users to interact with the data quality process, run tests, review results, and make real-time decisions.

The final step is to implement a scoring system that measures data quality. Scoring allows data quality leaders to quantify the state of their data and communicate it effectively to stakeholders. These scores permit focus and be configurable, allowing organizations to tailor them to different use cases, whether assessing critical data elements (CDEs), DAMA quality dimensions, or machine learning model data. Organizations can track improvements over time by continuously monitoring data quality scores and adjusting their efforts.

By implementing targeted and relevant data-scoring models, data quality (DQ) leaders can significantly enhance their organizational influence. It is crucial to prioritize scoring only the data elements that matter most to the business, avoiding extraneous metrics like FAX numbers that add little value. Instead of adopting a broad approach to Data Quality scoring, which often results in marginal insights, leaders should focus on scoring data that aligns with the organization's needs. This means designing scoring models that address critical business requirements and customer priorities, ensuring meaningful and actionable results.

Data Quality leaders should develop multiple scoring models tailored to distinct business needs to make data scoring more impactful, rather than relying on a one-size-fits-all approach. This ensures scores are relevant to different departments or use cases, allowing data quality efforts to drive targeted improvements. Each scoring model should prioritize business-critical data elements, starting with a few metrics that genuinely resonate with the customer. As

confidence and stakeholder buy-in grow, these models can be expanded to include additional elements supporting the organization's goals. This approach keeps data scoring manageable and maximizes the influence of DQ initiatives by continually aligning data quality metrics with evolving business priorities. Focused data scoring creates a feedback loop where data quality directly contributes to operational success, cementing the DQ leader's role as a strategic partner.

It's crucial to make sharing data quality scores and test results simple and impactful. Leaders must communicate the significance of identified issues effectively, convincing others to take corrective measures. Providing a shareable summary package that clearly outlines the problem, its impact, and steps for resolution makes it easier for teams to understand and fix the issue. Additionally, simplifying the process of recreating the test conditions ensures transparency and helps others effectively verify and address the problem. To streamline this process, integrating data quality management with existing workflow tools ensures that issues can be tracked, assigned, and resolved within the organization's standard processes, making it easier for teams to take action and improve data quality.

It is crucial to track Data Quality (DQ) scores over time and consistently measure improvements to demonstrate a meaningful impact on data quality. By keeping a record of these scores, Data Quality Leaders can highlight tangible progress in data quality management and showcase their role in driving that improvement. Additionally, identifying related operational KPIs (Key Performance Indicators) that are influenced by data quality improvements can strengthen the case for ongoing quality initiatives. For example, enhanced data quality could lead to better decision-making, fewer errors, and improved operational efficiency, all visible through KPIs. Measuring progress consistently before setting rigid standards is essential, allowing for flexibility and continuous refinement. Adopting a pragmatic approach, where an initial 70% accuracy score is acceptable, enables teams to make improvements incrementally rather than waiting for perfection, which can cause delays.

As new data comes in, Data Quality scores are expected to evolve. This makes it crucial to continuously revisit and refine the data quality tests that generate those scores. Regularly refining DQ standards ensures that tests become more accurate and yield fewer false positives, providing a clearer picture of data quality issues. Additionally, real-world data scenarios offer valuable learning opportunities, allowing teams to adapt their testing processes and standards based on practical experience. Over time, this continuous learning process means that even less-skilled team members can be involved in maintaining and improving data quality as they build knowledge of the data and processes. Monitoring DQ score trends on a chart provides visibility into data quality improvements and the effectiveness of evaluation criteria. As standards and data improve, the chart will reflect the progress, reinforcing the impact of refined data quality efforts.

Traditional, Waterfall, Data Quality Projects Versus The DataOps Data Quality 'Influence and Action' Cycle

Traditional approaches are about requirements gathering, analysis, meetings, documentation, and planning before any action is taken to improve data quality. Waterfall project management approaches often fail because they rely on rigid, sequential processes that need more flexibility in accommodating changes. In traditional waterfall models, project phases such as requirements gathering, design, development, testing, and deployment occur one after the other, with little room for feedback or iteration until the end of the process. However, many projects, particularly in dynamic environments like data analytics, require continuous adaptation based on new information or changes in requirements. As a result, waterfall methodologies often result in delayed responses to evolving needs, causing misalignment with stakeholders' expectations and leading to a mismatch between the final deliverable and the users' needs.

Another major issue with the waterfall approach is that it assumes that all data quality requirements can be fully understood upfront. In reality, it is often difficult to define all aspects of a project in the initial stages. This assumption leads to a need for more iterative feedback loops, making it harder to correct mistakes or refine deliverables throughout the project. In cases where problems arise, they may only be discovered late in the process, resulting in costly delays or even project failure. Agile methodologies, in contrast, allow for more incremental development and

testing, enabling teams to identify issues earlier and make adjustments on the fly, which is essential for modern data quality environments where adaptability is critical.

The DataOps approach to improving data quality is a cycle that empowers an individual who may not have direct authority but has influence within the organization. Organizations can start improving data quality from the ground up by giving these individuals the tools and resources they need to effect change. This person is working in a continuous cycle of influence and action. This cycle involves the following key activities:

Advantages of Data Quality The DataOps Way

DataOps provides several distinct advantages when applied to data quality management. These advantages are particularly pronounced in organizations dealing with large volumes of complex data, where traditional approaches often fall short.

Handling Data at Scale: DataOps helps teams handle massive amounts of changes at scale. As organizations grow and their data requirements expand, the ability to manage and ensure the quality of that data becomes increasingly important. DataOps automatically and at scale applies universal quality principles—such as consistency, accuracy, and completeness—ensuring that large datasets can be managed efficiently without sacrificing quality.

Knowledge-Based Trade-offs: One key benefit of the DataOps approach is its ability to facilitate knowledge-based trade-offs. Rather than making assumptions about data quality or relying on hope, DataOps encourages organizations to use real-time data and insights to inform their decisions. This ensures that trade-offs between speed, cost, and quality are made based on actual knowledge, leading to better outcomes. Don't rely on your intuition; look at the facts on the ground with actual data.

Leveraging Expertise: Subject matter experts and skilled data engineers often need to improve data quality. By focusing their efforts where they are most needed, organizations can maximize the impact of their resources and ensure that expertise is applied where it counts.

Faster, Easier Implementation: The iterative, Agile nature of DataOps makes it faster and easier to implement real data quality improvements. Rather than waiting for a perfect solution, teams can begin implementing changes immediately, learning from each iteration and refining their processes over time. This iterative approach accelerates the implementation of data quality initiatives and ensures that improvements are continuously being made. Getting something 70% right working today is much better than a perfect solution sometime in the future.

More Iterations, Better Data Quality Standards: The more iterations a team goes through in its data quality processes, the more refined its standards become. Each cycle provides an opportunity to learn, improve, and adjust the standards to reflect the organization's needs and goals better. Over time, this leads to establishing more robust, actionable standards that drive fundamental improvements in data quality. Maximize your team's learning from experience, not second-hand anecdotes.

Conclusion: Data Quality The DataOps Way

Ultimately, the DataOps approach allows organizations to get data quality processes working quickly and then iterate and improve over time. Documentation and analysis become natural outcomes, not barriers to progress. By starting with testing and measurements, even before standards are fully established, organizations can build a foundation for continuous improvement. The faster the iteration, the more organizations learn, refine their processes, and elevate their data quality standards.

Data quality, the DataOps way, is about action, agility, and learning—making it the best way to drive meaningful, scalable, and lasting improvements in data quality. With Data Quality The DataOps Way, organizations can:

The DataOps approach to data quality provides a scalable, iterative, and flexible framework for improving data quality across an organization. By empowering individuals, leveraging expertise, and focusing on continuous improvement,

DataOps enables organizations to address their data quality challenges proactively and effectively. Rather than waiting for perfect solutions, DataOps encourages organizations to start small, iterate quickly, and learn from each improvement cycle.

Original article: https://docs.google.com/document/d/144z4o1Py_alFzBgV3k8ghEQbzHayJpjM1gXqLLu6niE/export?format=html

Data Quality When You Don't Understand the Data: Data Quality Coffee With Uncle Chip #3

You can't test for what you don't understand. And so, we don't. We skip it. We assume. Or worse, we wait until something breaks. What is the solution? Uncle Chip tells!

Data Quality When You Don't Understand The Data

Watch Now!

DataKitchen

Hygiene Issues
Profiling Run: Final Sales Star Schema | Apr 17, 6:09 PM

Hygiene Issues Potential PII Score: 99.0%

Issue Class	Issue Type	Column Name	Issue Likelihood	Act
d_kitchen_customers	address	address	Potential PII	
d_kitchen_customers	first_name	first_name	Potential PII	
d_kitchen_customers	gender	gender	Potential PII	
d_kitchen_customers	income_level	income_level	Potential PII	
d_kitchen_customers	last_name	last_name	Potential PII	
d_kitchen_customers	marital_status	marital_status	Potential PII	
d_kitchen_customers	postal_code	postal_code	Potential PII	
d_kitchen_products	tax_id	tax_id	Potential PII	
d_kitchen_products	frame_size	frame_size	Potential PII	
d_kitchen_products	product_name	product_name	Potential PII	

DATA QUALITY COFFEE with Uncle Chip
TestGen by DataKitchen

Let's be honest—data quality feels impossible when you don't understand the data. And in large organizations, that's not a rare problem. It's the norm. We've seen it firsthand: massive data estates maintained by teams who don't know what the numbers, strings, or categories in their tables really mean. This isn't because people are lazy or careless. It's structural. Domain experts and technical folks operate on parallel tracks—outsourced, offshored, remote, and siloed. They often don't even speak the same language, let alone share the same Slack channels.

That's the crux of the issue: you can't test for what you don't understand. And so, we don't. We skip it. We assume. Or worse, we wait until something breaks. Until a business leader notices that a dashboard shows the wrong customer segments or a machine learning model recommends absurd products. This reactive, fire-drill culture around data quality is like posting a guard at the front door while the back door is wide open. We're trying to protect our data, but we're guarding the wrong entry points—and often doing it way too late.

In most enterprises, documentation is stale or useless. People who once understood the data have moved on. Meanwhile, new projects are spun rapidly, and datasets are reused for purposes far removed from their original intent. AI, predictive modeling, real-time analytics—none of these cutting-edge initiatives can succeed without clean, reliable data. But how do you build confidence in the data if you have no context, no SME on speed dial, and no time to write detailed tests?

DataKitchen created TestGen because we were tired of the myth that you must know everything about your data before testing it. That's not true. You can start with common sense—and scale from there. TestGen is open-source software that runs a series of tests on your data without requiring you to write a single line of YAML, SQL, or Python. It begins with a profiling step: it reads your data column by column, learns its shape, content, and structure, and immediately flags issues obvious to any human, but usually overlooked by automated tools.

TestGen covers [27 kinds of data hygiene problems](#): from the usual suspects like missing values and bad formats to more subtle issues like inconsistent categories, skewed distributions, and internal contradictions. It also flags 24 different types of personally identifiable information (PII). This is especially important in today's privacy-conscious

world, where protecting sensitive information is not optional—it's mandatory. And let me tell you, most teams miss the obvious stuff. It's not just about encryption and access control; it's about knowing what you've got in the first place.

What's powerful about TestGen is how it uses profiling to infer rules. It builds a baseline of your data—what it looks like when it's healthy—and then checks future data against that baseline. Any deviation gets flagged, even if it's just a slight shift in category frequency or a sudden drop in data recency. Think about it: if a critical column suddenly has a dozen new codes that no one's seen before, wouldn't you want to know? That's not just hygiene—it's how you spot systemic breakdowns before they reach the business layer.

We also [baked in tests for transactional data freshness, statistical outliers, duplicate detection, uniqueness violations, and more](#). These tests don't need a PhD in data science to understand or run. They give you visibility into what's breaking and why, without needing perfect documentation or tribal knowledge. And that's the key: you don't have to be a business expert to start improving data quality. You just have to start.

The truth is, most people working in data are waiting for someone else to fix things. Don't wait. Be the one who turns on the lights. Start running tests. See what you find. Refine your rules over time. You don't need to be an expert to make a difference—you just need initiative, and maybe a little help from software built for this messy, complex world.

Because when we stop pretending we understand every data column and acknowledge that we don't, we can finally begin to see the problems and do something about them. And in our book, that's where real data quality begins.



Watch The 'Data Quality Coffee With Uncle Chip' Series

Author: Chip Bloche | Published: Apr 18, 2025

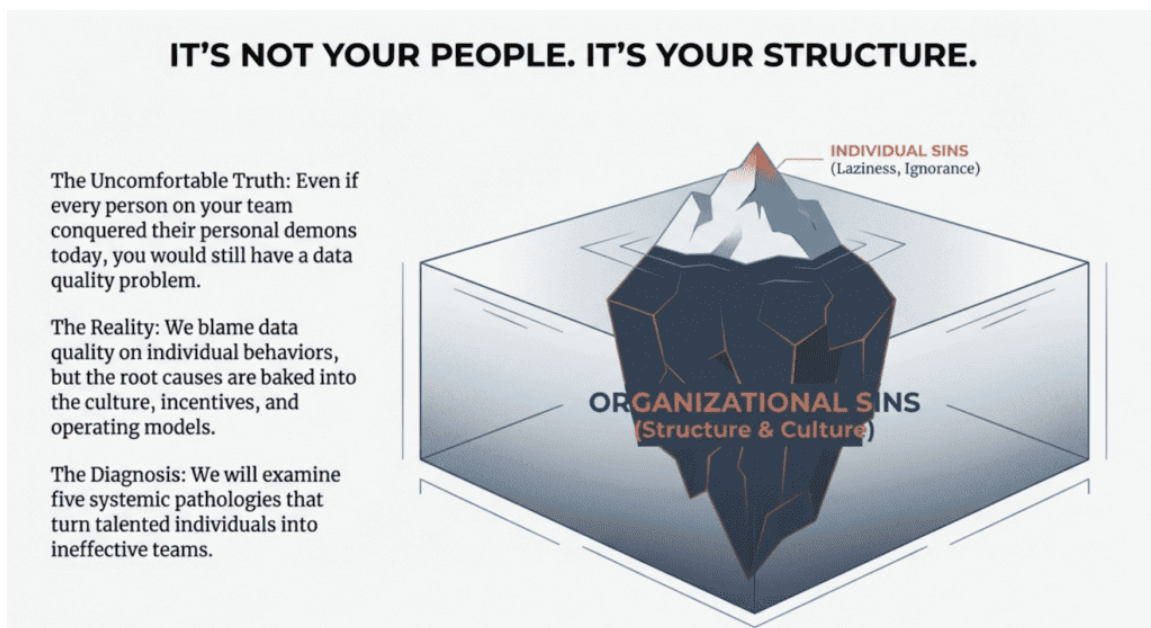
Original article: <https://datakitchen.io/data-quality-when-you-dont-understand-the-data-data-quality-coffee-with-uncle-chip-3/>

The Five Organizational Sins of Data Teams

The five organizational problems are systemic failures of culture and structure that lead to warring teams, a corrosive blame culture, "data blindness" regarding pipelines, a short-term project mindset, and the chaotic scattering of business logic. Learn how to identify and fix below.



In the first part of this series, we explored the [Seven Deadly Sins of Data Quality](#), focusing on the individual behaviors that sabotage data teams: shame, denial, avoidance, passivity, laziness, gluttony, and ignorance. But here is the uncomfortable truth that nobody wants to talk about: even if every person on your data team somehow conquered their personal demons tomorrow, you would still have a data quality problem. The issue is not just about people. It is about how organizations structure their teams, define success, and approach the very nature of data work itself.



Organizational sins are more challenging to diagnose and even more complex to fix because they are baked into the culture, the incentive structures, and the operating models that everyone takes for granted. These are the systemic

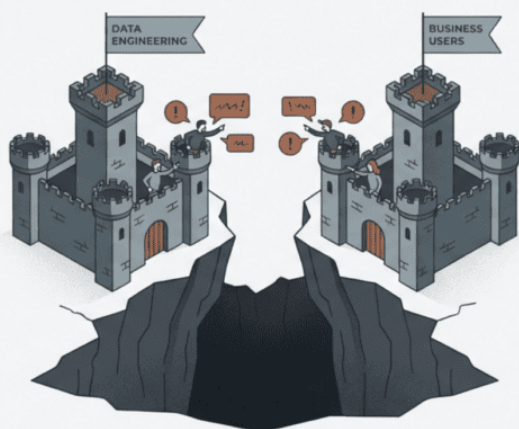
dysfunctions that turn talented individuals into ineffective teams and promising initiatives into expensive disappointments. In this follow-up post, we examine five organizational sins that plague data teams: low relational coordination, blame culture, data blindness, project thinking, and lack of process curation. Recognizing these patterns is the first step toward building data organizations that actually work.

Sin One: Low Relational Coordination

Data organizations face unique challenges in fostering positive interaction and communication, as tools and workflows often promote isolation. Groups that should be partners working toward shared objectives often behave more like warring tribes competing for resources and pointing fingers when things go wrong. As explored in [Warring Tribes into Winning Teams](#), geographical, cultural, and language barriers compound with technology-driven silos to create what management scientists call low relational coordination. The data engineers, analysts, scientists, and business users who need to collaborate seamlessly instead operate in isolated bubbles, communicating through emails and handoffs rather than a genuine partnership.

SIN #1: LOW RELATIONAL COORDINATION

Teams behaving like warring tribes rather than partners.



The Trap: Technology-driven silos and geographical barriers create isolation. Groups that should be partners operate as competitors.

The Behavior: Communication happens through ticket handoffs and “throwing requirements over the wall.”

The Result: “Us vs. Them” mentality. Engineers build pipelines without understanding context; business users create shadow IT.

The consequences ripple through every aspect of data operations. Analytics cycle times stretch as teams wait for responses from groups they barely know. Data quality suffers because no one has visibility into the entire pipeline. Governance becomes a bureaucratic exercise rather than a shared commitment. Employee retention plummets as talented people tire of fighting the same battles over and over. The solution requires more than better communication tools. Organizations must restructure their workflows to build strong role relationships and coordination that enable high performance. DataOps provides a framework for doing exactly that, using automation and process design to facilitate rather than hinder the communication between groups.

Sin Two: Blame Culture

In the world of data and analytics, one skill stands timeless and universal: the art of blaming someone else when things go sideways. As satirized in [The Art of Data Buck-Passing 101](#), data teams have elevated finger-pointing to a dark art form. When data issues surface, development teams blame QA. QA blames operations. Operations blames the data providers. Everyone invokes the sacred incantation of “garbage in, garbage out” while carefully positioning themselves outside the blast radius. The [Spiderman pointing meme](#) has become the unofficial mascot of data team retrospectives.

SIN #2: BLAME CULTURE

The art of data buck-passing.



The Trap: When data breaks, the organization asks “Who did this?” rather than “How do we fix this?”

The Behavior: “Garbage in, garbage out” becomes a shield to deflect responsibility. Dev blames QA. QA blames Ops. Ops blames Source.

The Symptom: The “Shame Game”—engineers avoid looking for errors because finding one means owning the blame.

This blame culture is not just demoralizing; it is organizationally corrosive. When people fear being blamed for problems, they stop looking for problems. They push untested code to production because once it is deployed, someone else becomes responsible. They craft emails that document decisions in ways designed to deflect future accountability rather than facilitate current collaboration. The energy that should flow into fixing root causes instead flows into constructing elaborate defenses. Breaking this cycle requires shared visibility into data pipelines, clear ownership models, and a cultural shift from “who broke it” to “how do we prevent it from breaking again.” Until organizations address the blame game directly, they will continue playing it.

Sin Three: Data Blinders

Most data teams are data-centric rather than process-centric, and this distinction has profound implications for their effectiveness. As described in [Are You Process-Centric or Data-Centric?](#), data-centric teams perceive complexity as directly proportional to the number of tables they manage. Their databases feel like mysterious, daunting black boxes full of tables updated at seemingly random frequencies by processes that are unknown or uncontrolled. Data engineers in these environments frequently respond to quality issues without direct control over the processes that cause the errors. They are held responsible for data problems but cannot actually fix or influence the pipelines that create them.

SIN #3: DATA BLINDNESS

Drowning in tables, starving for control.



- **The Trap:** Being Data-Centric instead of Process-Centric.
- **The Behavior:** Measuring complexity by the number of tables managed. Treating databases as mysterious “black boxes” that update randomly.
- **The Reality:** You are held responsible for errors in pipelines you cannot see or control. You fix the data, but not the process that broke it.

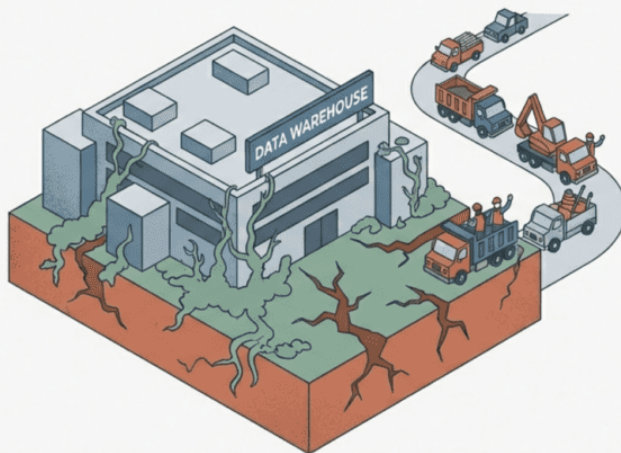
Process-centric teams operate completely differently. They focus on orchestrating and automating workflows, recognizing that robust, well-managed data processing pipelines inevitably yield reliable, high-quality data. Rather than focusing on individual tables, these teams allocate resources to ensure each pipeline is transparent, thoroughly tested, and easily deployable. They treat their pipelines as cattle, not pets, enabling them to quickly diagnose and rectify issues at the process level rather than getting lost in individual data table anomalies. The shift from data-centric to process-centric thinking is achievable, but it requires leadership commitment to unified orchestration, process transparency, and rigorous adherence to DataOps principles.

Sin Four: Project, Not Product Focus

When data team leaders hear “data products,” they immediately think of the stuff they produce: dashboards, datasets, models, and warehouses. But as argued in [You're Thinking About Data Products All Wrong](#), focusing on the “what” completely misses the massive shift that data products really represent. The truth is that data products are not about what you create but how you build, maintain, and continually improve your data assets to deliver value to customers. When organizations think about data products merely as deliverables, they fall into the same trap that has plagued data teams for decades: the project mindset.

SIN #4: PROJECT, NOT PRODUCT FOCUS

The “One-Off” Trap.



- **The Trap:** Treating data work as finite projects with a “done” date.
- **The Behavior:** Success is defined by delivery on time/budget, not by long-term value or reliability.
- **The Flaw:** Data is never “done.” It evolves, drifts, and breaks. The “Ostriches” bury their heads after deployment, ignoring downstream decay.

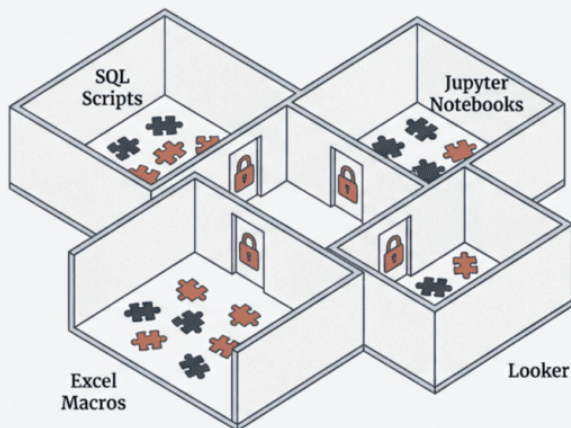
In the project world, workflows progress through stages in sequence. Teams are assigned to specific projects, creating silos and handoffs that slow delivery and degrade quality. Success is measured by whether the project was delivered on time and within budget, not by whether users actually found it valuable. The product approach transforms this dynamic entirely. Instead of significant, risky releases, teams deliver minor, incremental improvements continuously. Work is brought to stable teams who own their products end-to-end. Most importantly, the approach recognizes that data products are never “done.” This continuous improvement mindset yields measurable benefits: faster releases, increased team learning, better insights, and ultimately faster delivery of value from data investments.

Sin Five: Lack of Process Curation

There is a quiet crisis happening in data organizations everywhere. It occurs when brilliant analysts independently solve the same problem in three different ways, when data scientists struggle to reproduce each other’s work, and when a single “simple” metric like Monthly Recurring Revenue has five competing definitions scattered across the organization. As detailed in [Process Guardianship: The Most Valuable Data Engineering Work You’re Probably Not Doing](#), the absence of deliberate process curation means that business logic lives everywhere and nowhere. Metrics, models, calculations, and code are scattered across SQL scripts, Jupyter notebooks, Looker reports, Python models, and Excel macros, with no one taking responsibility for consolidating them into a unified, tested, production-grade system.

SIN #5: LACK OF PROCESS CURATION

Business logic: everywhere and nowhere.



- **The Trap:** Important logic is scattered across hidden corners and tribal knowledge.
- **The Behavior:** A simple metric like “Monthly Recurring Revenue” has five different definitions depending on who you ask (and which Excel sheet they use).
- **The Reality:** No deliberate consolidation or guardianship of the process.

The costs compound quickly. Duplicate work becomes the norm as teams waste countless hours reinventing the wheel because they do not know what already exists or cannot trust it enough to reuse. Multiple overlapping metrics create confusion and erode stakeholder trust. When leadership asks how many customers are in the high-value segment, they get three different answers. Embarrassing data errors slip through because corrections identified by one team are not shared with the rest of the enterprise. Data engineers have a unique opportunity to address this crisis through what might be called process guardianship: actively consolidating business logic, extracting it from hidden corners and tribal knowledge, and encoding it in a centralized, version-controlled, production-ready system. This work may seem unglamorous, but it is often the most valuable contribution a data team can make.

Moving From Organizational Sin to Organizational Strength

These five organizational sins share a common thread: they all represent failures of systems and structures rather than failures of individuals. You cannot shame individual data engineers into fixing low relational coordination. You cannot train analysts out of blame culture while the incentive structures reward blame. You cannot wish your way from data-centric to process-centric thinking without fundamentally changing how work is organized and measured. This is precisely why DataOps exists. DataOps is not just DevOps for data. It is a comprehensive methodology that applies Agile development, DevOps, and lean manufacturing practices to data analytics, enabling organizations to accelerate the development of new analytics, deploy with confidence, and reduce data errors to virtually zero.

The [DataOps Cookbook](#) provides a practical roadmap for implementing these principles. Now in its third edition, the Cookbook explains how DataOps draws on lessons from the software and manufacturing industries to transform data chaos into a single, smooth, efficient process that delivers fast, error-free insight. It covers everything from understanding Data Journeys and implementing observability to building the automated testing and unified orchestration that process-centric teams require. The methodology addresses each of the organizational sins we have discussed: it breaks down silos through shared visibility and collaboration frameworks, eliminates blame culture by making pipelines transparent and accountable, shifts teams from data-centric to process-centric thinking, enables product mindsets through continuous delivery practices, and provides the foundation for true process guardianship.

THE REVOLUTION STARTS WITH YOU

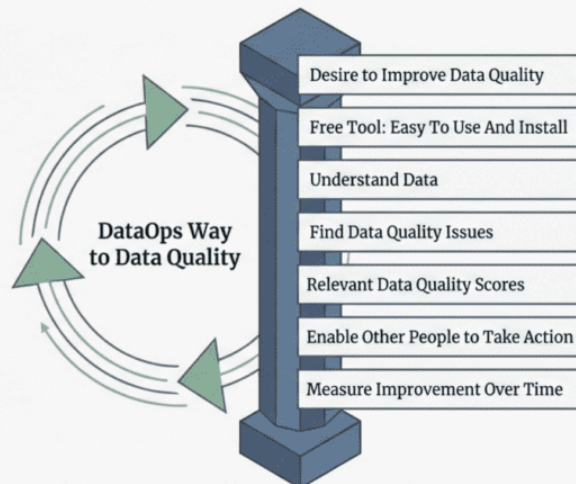
Don't wait for a culture shift.
Create one.

Step 1: Download DataOps TestGen
(It's Free & Open Source).

Step 2: Generate your first quality
report in minutes.

Step 3: Turn on the lights and see what
you've been missing.

<https://info.datakitchen.io/testgen>



DataKitchen: The DataOps Experts.

The journey from organizational dysfunction to data team excellence is not easy, but it is achievable. Thousands of data professionals have already formed study groups around the DataOps Cookbook and are beginning to transform their organizations. The key insight is that you do not need to fix everything at once. Start by focusing on understanding and observing the journey that data takes through your production environment. This monitoring process identifies data errors, tool problems, and timing issues, enabling a quick win for your DataOps implementation by driving immediate improvements. Lowering production errors increases the reliability of your data and gives your team more time to focus on automation. From there, you can systematically address each organizational sin, building the collaborative, process-centric, product-oriented data organization that actually delivers on the promise of data-driven decision making. Download the [DataOps Cookbook](#) and start your team's transformation today.

What is the summary of the blog The Five Organizational Sins of data teams?

While the "Seven Deadly Sins" previously focused on individual behaviors (like shame and denial), the Five Organizational Sins describe systemic dysfunctions. These failures are baked into a company's culture, incentive structures, and operating models. Even if every individual conquers their personal bad habits, these structural issues can still cause data teams to fail. Here are the five organizational sins of data teams:

The Problem: Technical, cultural, and language barriers separate data engineers, analysts, and business users. Instead of collaborating, they communicate through tickets and handoffs, behaving like "warring tribes" competing for resources.

The Consequence: This isolation creates low relational coordination, leading to stretched cycle times and poor data quality because no single group has visibility into the entire pipeline. It turns governance into a bureaucratic exercise rather than a shared commitment.

This is the systemic habit of pointing fingers when data issues arise, rather than fixing the root cause.

The Problem: When data breaks, teams immediately look for someone to blame: Development blames QA, QA blames Operations, and everyone blames the data providers.

The Consequence: This is corrosive because it discourages people from looking for problems. If finding an error means being blamed for it or having to fix it alone, engineers will "push untested code to production" to shift responsibility elsewhere. Energy that should be spent on prevention is instead spent on constructing defenses to stay outside the "blast radius" of failure.

This sin is the failure to manage the factory (the pipeline) because the team is too focused on the product (the tables).

The Problem: Most teams are data-centric, viewing complexity as the number of tables they manage. They treat databases like “black boxes” full of tables updated by mysterious processes they do not control.

The Consequence: Engineers are held responsible for data quality but lack control over the pipelines that cause errors. Successful organizations must shift to a process-centric approach, treating pipelines like “cattle, not pets,” and focus on orchestrating and automating the workflows that produce the data.

The Problem: Teams view success as delivering a project (like a dashboard or dataset) “on time and on budget.” Once delivered, the team moves on to the next project, creating silos and handoffs in the process.

The Consequence: This ignores the reality that data products are never truly “done.” A project mindset leads to risky “big bang” releases. A product mindset, by contrast, keeps stable teams on the asset to deliver continuous, incremental improvements and maintain value over time.

This is the chaos that ensues when business logic is scattered across the enterprise without ownership.

The Problem: Critical logic lives “everywhere and nowhere”—buried in SQL scripts, hidden in Excel macros, or locked in Tableau calculations. There is no “Process Guardian” responsible for consolidating this logic.

The Consequence: This leads to “metric confusion,” where a simple number like Monthly Recurring Revenue has five different definitions across the company. Teams waste time reinventing the wheel, and stakeholders lose trust because they get different answers to the same question from different departments.

Unlike individual sins, which can be addressed through personal initiative, organizational sins represent failures of systems. You cannot shame a team out of low relational coordination, nor train them out of a blame culture if the incentives reward it. Curing these requires a fundamental shift toward DataOps principles: shared visibility, automated orchestration, and continuous improvement.

Author: Chris Bergh | Published: Jan 19, 2026

Original article: <https://datakitchen.io/the-five-organizational-sins-of-data-teams/>

Warring Tribes into Winning Teams: Improving Teamwork in Your Data Organization With DataOps and Relational Coordination



If the groups in your data-analytics organization don't work together, it can impact analytics-cycle time, data quality, governance, employee retention and more. A variety of factors contribute to poor teamwork.

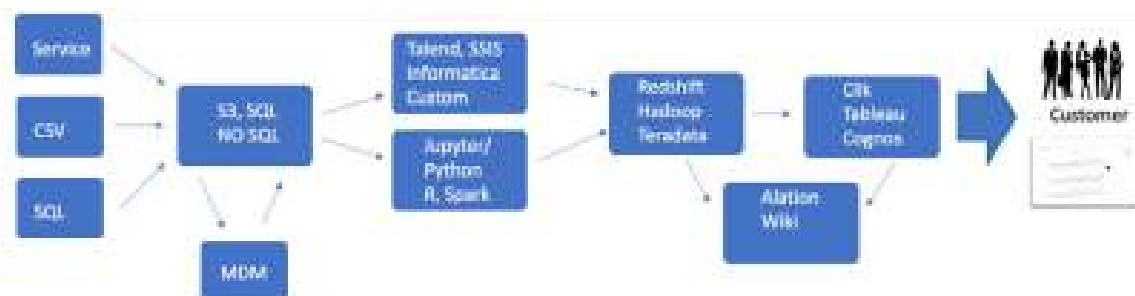
Sometimes geographical, cultural and language barriers hinder communication and trust. Technology-driven companies face additional barriers related to tools, technology integrations and workflows which tend to drive people into isolated silos.

The data organization shares a common objective; to create analytics for the (internal or external) customer. Execution of this mission requires the contribution of several groups shown in Figure 1. These groups might report to different management chains, compete for limited resources or reside in different locations. Sometimes they behave more like warring tribes than members of the same team.



Figure 1: Delivery of analytics (the value chain) to customers requires contributions from several groups in the data organization

Let's explore some of the factors that isolate the tribes from one another. For starters, the groups are often set apart from each other by the tools that they use. Figure 2 is the same value chain as above but reconstructed from the perspective of tools.



To be more specific, each of the roles mentioned above (figure 1) view the world through a preferred set of tools (figure 2):

- Data visualization, Preparation — Self Service tools, Tableau, Alteryx
- Data Governance/Catalog (Metadata management) Workflow — Alation, Collibra, Wikis

The day-to-day existence of a data engineer working on a master data management (MDM) platform is quite different than a data analyst working in Tableau. Tools influence their optimal iteration cycle time, e.g., months/weeks/days. Tools determine their approach to solving problems. Tools affect their risk tolerance. In short, they view the world through the lens of the tools that they use.

The division of each function into a tools silo creates a sense of isolation which prevents the tribes from contemplating their role in the end-to-end data pipeline. The less they understand about each other, the less compelling the need to communicate about actions taken which impact others. Communication between teams (people in roles) is critical to the organization's success. Most analytics requires contributions from all the teams. The work output of one team may be an input to another team. In the figure below, the data (and metadata) build as the work products compound through the value chain.

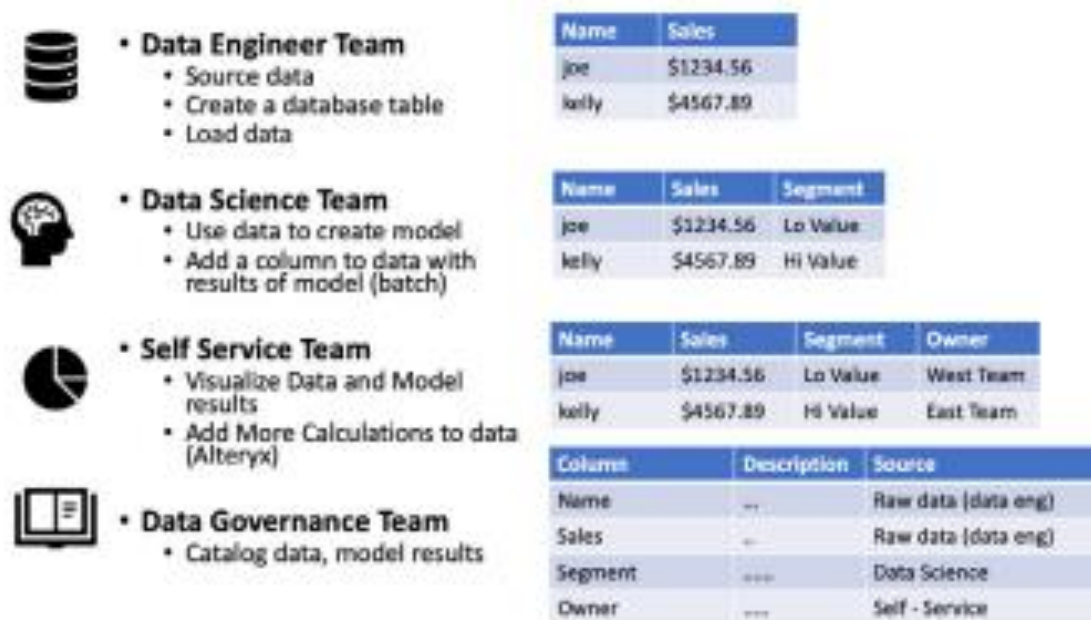
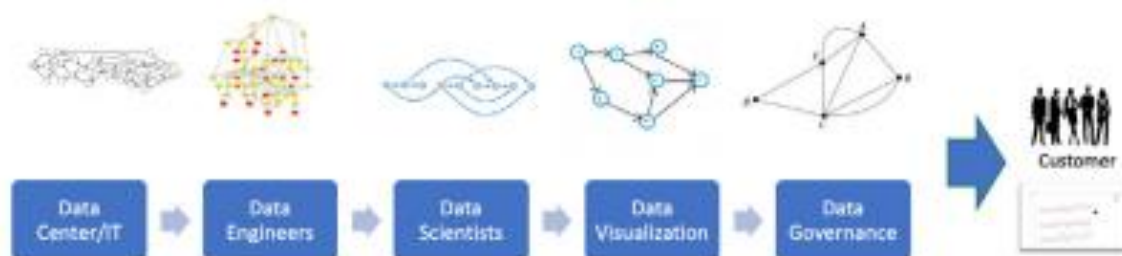


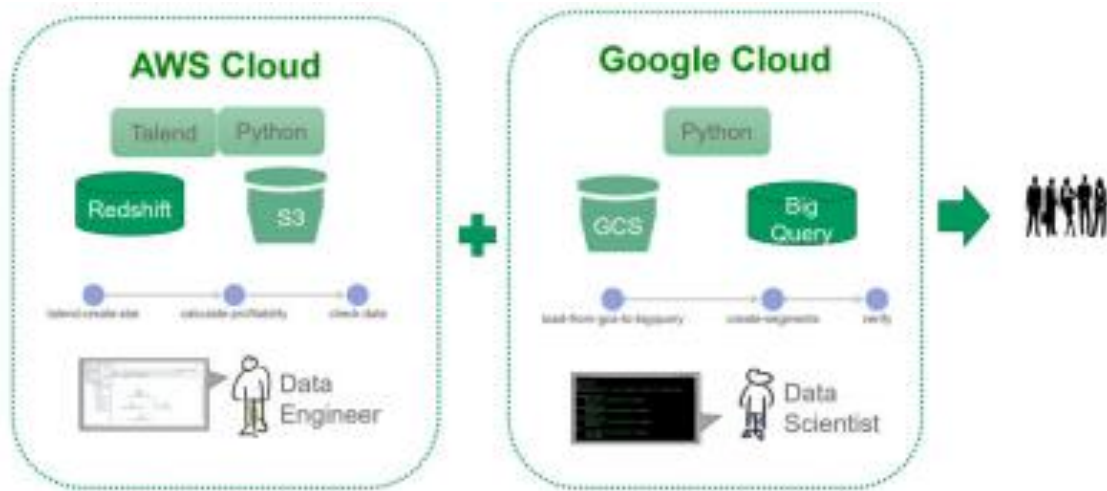
Figure 3: Each group adds unique value to analytics. In most cases, the work of one group is an input to the next group.

In many enterprises, there is a natural tendency for the groups to retreat into the complexity of their local workflow. In figure 4, we represent the local workflow of each tribe with a directed-acyclic graph (DAG).



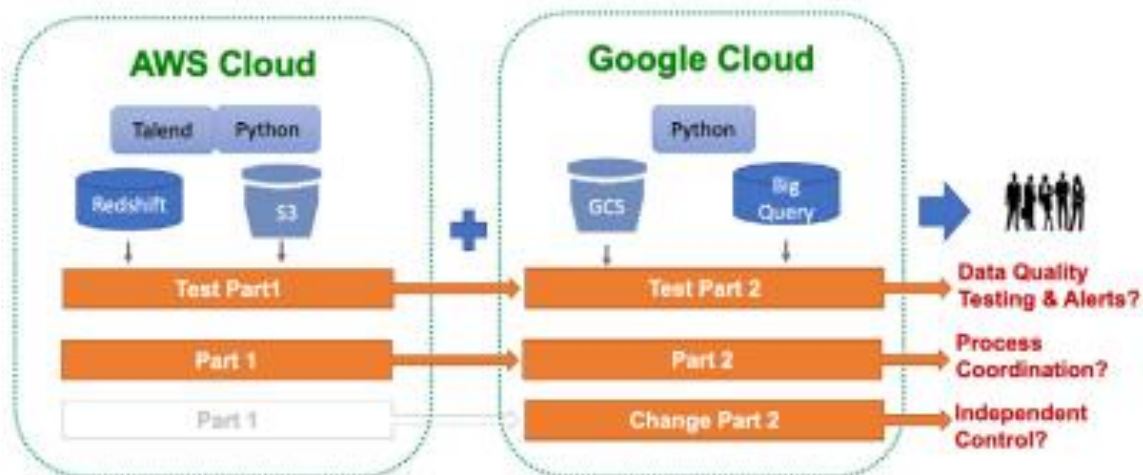
It is too easy to overlook the fact that the shared purpose of these local workflows is to work together to publish analytics for end-customers.

Group isolation is also induced by platforms, release cadence and geographic locations. The example below shows a multi-cloud or multi-data center pipeline with integration challenges.



The two groups managing the two halves of the solution have difficulty maintaining quality, coordinating their processes and maintaining independence (modularity). Group one tests part one of the system (figure 6). Group two validates part two.

Do the part one and two tests deliver a unified set of results (and alerts) to all stakeholders? Can tests one and two evolve independently without breaking each other? These issues repeatedly surface in data organizations.



In another example, assume that two groups are required to work together to deliver analytics to the VP of marketing. The home office in Boston handles data engineering and creates data marts. Their iteration period is weekly. The local team in New Jersey uses the data marts to create analytics for the VP of Marketing. Their iteration is daily (or hourly).



One day, the VP of Marketing requests new analytics (deadline ASAP) from the data analysts for a meeting later that day. The analysts jump into action, but face obstacles when they try to add a new data set. They contact data engineering in Boston. Boston has its own pressures and priorities and their workflow, organized around a weekly cadence, can't respond to these requests on an "ASAP" basis.

The home office team in Boston finally makes the needed changes, but they inadvertently break other critical reports (figure 8). Meanwhile, out of desperation, the New Jersey team adds the required data sets and updates their analytics. The new data sets are only available to New Jersey, so other sites are now a revision behind. New Jersey's reports are inconsistent with everyone else's. Misunderstandings ensue. It's not hard to imagine why the relationship between these groups could be strained.

These challenges may seem specific to data organizations, but at a high level, everything that we have discussed boils down to poor communication and lack of coordination between individuals and groups. As such, we can turn to management science to better understand the problem and explore solutions.

Strip away the technological artifacts from the situations described above and you are left with an organization that cannot foster strong role relationships and communication between employees. These

challenges are not unique to technology-driven organizations. Many enterprises across a wide variety of industries face similar issues.

For those who don't remember, the airline business in the 1980s and 1990s was brutally competitive, but during this same period, Southwest Airlines revolutionized air travel. By the early 2000s, they had experienced 31 straight years of profitability and had a market capitalization greater than all the other major US airlines combined. Brandeis management professor Jody Hoffer Gittel investigated the factors in Southwest Airlines' performance and, back in 2003, published a quantitative, data-driven analysis shedding light on Southwest's success.

Dr. Gittel surveyed the major players in the airline industry and found a correlation between key performance parameters (KPP) and something that she termed Relational Coordination (RC), the way that relationships influence task coordination, for better or worse. "Relational coordination is communicating and relating for the purpose of task integration — a powerful driver of performance when work is interdependent, uncertain and time constrained."

In her study, higher RC levels correlated with better performance on KPPs, even when comparing two sites within the same company. Since that time RC has been applied in industries ranging from healthcare to manufacturing across 22 countries.

One common misconception is that RC focuses on personal relationships. While personal relationships are important, RC is more concerned with the relationship of roles and workflows within the organization. RC studies how people interact and exchange information in executing their role-based relationships.

Members of the “Low-RC” organization express their goals solely in terms of their own function. They keep knowledge to themselves and there may be a tendency for one group to look down upon another group. Inter-group communication is inadequate, inaccurate and might be more concerned with finding blame than finding solutions. As expected, the “High-RC” organization embodies the exact opposite end of this spectrum.

“High-RC” team members understand the organization’s collective goal. They not only know what to do but why, based on a shared knowledge of the overall workflow. Everyone’s contribution is valued, and no one is taken for granted. There is constant communication, especially when a problem arises.

At this point you may be thinking: “OK fine, this is all touchy-feely stuff. I’ll try to smile more and I’ll organize a pizza party so everyone can get to know each other.” Maybe you should (smiling will make you feel good and parties are fun after all), but our experience is that the good feeling wears off once the last cupcake is gone and the mission-critical analytics are offline.

How do you keep people working independently and efficiently when their work product is a dependency for another team? How can one team reuse the data or artifacts or code that another team produces?

For most enterprises, improving RC requires foundational change. You need to examine your end-to-end data operations and analytics-creation workflow. Is it building up or tearing down the communication and relationships that are critical to your mission? Instead of allowing technology to be a barrier to Relational Coordination, how about utilizing automation and designing processes to improve and facilitate communication and coordination between the groups? In other words, you need to restructure your data analytics pipelines as services (or microservices) that create a robust, transparent, efficient, repeatable analytics process that unifies all your workflows.

BUILDING A HIGH-RC ENTERPRISE USING DATAOPS DataOps is a new approach to data analytics that applies lean manufacturing, DevOps and Agile development methods to data analytics. DataOps unifies your data operations pipeline with the publication of new analytics under one orchestrated workflow.

- **Robust** – Statistical process control (lean manufacturing) calls for tests at the inputs and outputs of each stage of the data operations pipeline. Tests also vet analytics deployments, like an impact review board, so new analytics don’t disrupt critical operations.
- **Transparent** – Dashboards display the status of new analytics development and the operational status of the data operations pipeline. Automated alerts communicate issues immediately to appropriate response teams. Team members can see a birds-eye-view of the end-to-end workflow as well as local workflows.
- **Efficient** – Automated orchestration of the end-to-end data pipeline (from data sources to published analytics) minimizes manual steps that tie up resources and introduce human error. Balance is maintained between centralization and decentralization; the need for fast-moving innovation, while supporting standardization of metrics, quality and governance.
- **Repeatable** – Revision control with built-in error detection and fault resilience is applied to the data operations pipeline.
- **Sharable and Chunkable** – Encourage reuse, by creating a services oriented architecture (SOA) for your team to use together.

It may help to provide further concrete examples of a DataOps implementation and how it impacts productivity. Some of these points are further explained in our blog [DataOps in Seven Steps](#):

Figure 9: DataOps is a task coordination and communication framework that uses technology to break down the barriers between the groups in the data organization.

- *DataSharing – data sources flow into a data lake which is used to create data warehouses and data marts. Bringing data under the control of the data organization decouples it from IT operations and enables it to be shared more easily.*
- *Deployment of code into an existing system – continuous integration and continuous delivery of new analytics, leveraging on-demand IT resources and automated orchestration of integration, test and deployment.*
- *Environment startup, shutdown – With computing and storage on-demand from cloud services (infrastructure as code), large data sets and applications (test environments) can be quickly and inexpensively copied or provisioned to reduce conflicts and dependencies.*
- *Testing of data and other artifacts – Testing of inputs, outputs, and business logic are applied at each stage of the data analytics pipeline. Tests catch potential errors and warnings before they are released so the quality remains high. Test alerts immediately inform team members of errors. Dashboards show the status of tests across the data pipeline. Manual testing is time-consuming and laborious so it can't be done in a timely way. A robust, automated test suite is a key element in continuous delivery.*
- *Reuse of a set of steps across multiple pipelines – Analytics reuse is a vast topic, but the basic idea is to componentize functionalities as services in*

ways that can be shared. Complex functions, with lots of individual parts, can be containerized using a container technology (like Docker).

Learn More About DataOps For more information about DataOps please refer to www.datakitchen.io. About DataKitchen

DataKitchen, Inc. helps organizations turn data into value by offering the world's first DataOps platform. With DataKitchen, data and analytic teams can orchestrate data to value and deploy features into production while automating quality. These teams benefit from delivering value quickly, with high quality, using the tools that they love. DataKitchen is leading the DataOps movement to incorporate Agile Software Development, DevOps, and manufacturing-based statistical process control into analytics and data management. DataKitchen is headquartered in Cambridge, Massachusetts.

© 2019 DataKitchen, Inc. All Rights Reserved. The information in this document is subject to change without notice and should not be construed as a commitment by DataKitchen. While reasonable precautions have been taken, DataKitchen assumes no responsibility for any errors that may appear in this document. All products shown or mentioned are trademarks or registered trademarks of their respective owners. | 910507A

Original article: <https://docs.google.com/document/d/12hBRlqjEA5KN1AoF2YDVvPKg5IfzCT-doU21-o3hIUQ/export?format=html>

The Art of Data Buck-Passing 101: Mastering the Blame Game in Data and Analytic Teams

In data and analytics, one skill stands timeless and universal: the art of blaming someone else when things go sideways. In this humorous blog, learn from the best!



Welcome, dear readers, to the hallowed halls of Data Buck-Passing University, where the motto is “Per Alios Culpa Transfertur” (Blame is Transferred to Others). In the world of data and analytics, one skill stands timeless and universal: the art of blaming someone else when things go sideways. Let’s embark on this ironic journey through the curriculum of advanced finger-pointing, where blame is not just for breakfast anymore. It’s a full-course meal.

In this foundational course, students will learn the subtle nuances of directing attention away from themselves with the grace of a ballet dancer avoiding a puddle. This course covers the basics, from the “I believe that was their responsibility” to the advanced “As per my last email...” techniques. With some practice, you’ll perfect that innocent shrug and bewildered look that says, “Me? Could never be!”

Who said blame had to be confined to the early morning scrum meetings? In this module, we expand your palate, allowing you to savor the flavors of blame throughout the day. From the morning email blitz to the late-night Slack message bombardment, discover the joy of sprinkling a little blame in every interaction. It’s the seasoning that makes the [data world](#) go round.

Why take responsibility when you can embark on the exhilarating hunt for the fall guy? This course teaches you ten innovative strategies to identify the perfect scapegoat for every occasion. From interns to the cloud service provider, nobody is safe. By the end, you can pin an ETL failure on the office plant with convincing evidence.

In the grand tradition of the Olympics, this course introduces you to the competitive world of Covering Your Analytics (CYA). Here, you'll learn to draft emails, document decisions, and log meetings with the precision of an athlete, all in the noble pursuit of ensuring that when the data hits the fan, your tracks are not just covered; they're practically invisible.

This module dives deep into the cultural phenomenon of the Spiderman finger-pointing meme, exploring its relevance to data teams worldwide. Through a comprehensive analysis, learn how to implement this timeless meme into your daily work life, ensuring that when accountability is called for, you're ready with a meme that says, "Why not them?"

Does the report contain errors? It must be those guys who supplied us with the data; they are constantly screwing up. They are always the real reason. Learn five easy ways to say 'Garbage In, Garbage Out' in calm, soothing business terms. Or ten ways to say, 'It's a Data Governance type problem.'

Ah, the crowning jewel of our curriculum. This advanced seminar encourages the reckless push of untested code into Production with the safety net of a well-crafted blame strategy. Why bother with regression testing and impact checking when you can deploy and deflect? After all, someone else is in charge of the deployment pipeline, ripe for the blaming.

By the end of your studies at Data Buck-Passing University, you'll have mastered the art of deflection and gained a PhD in Not My Fault-ology. Remember, in the fast-changing world of data and analytics, it's not about the mistakes made but about how creatively and convincingly you can attribute those mistakes to someone else.

So, embrace the chaos, enroll today, and enter a brighter future where accountability is just a word in the dictionary between "absurdity" and "accolade." Welcome to the art of blame—a skill that will serve you in data meetings, team collaborations, and, unfortunately, life itself.

Author: Chris Bergh | Published: Feb 22, 2024

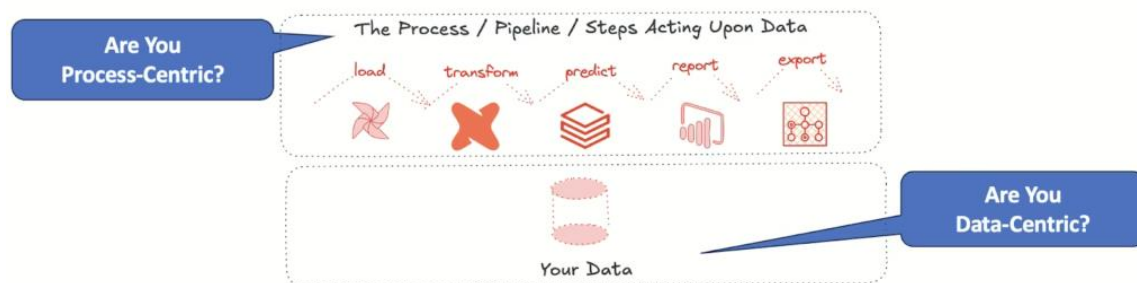
Original article: <https://datakitchen.io/the-art-of-data-buck-passing-101/>

Unlocking Data Team Success: Are You Process-Centric or Data-Centric?

We want to share our observations about data teams, how they work and think, and their challenges. We've identified two distinct types of data teams: process-centric and data-centric. Understanding this framework offers valuable insights into team efficiency, operational excellence, and data quality.

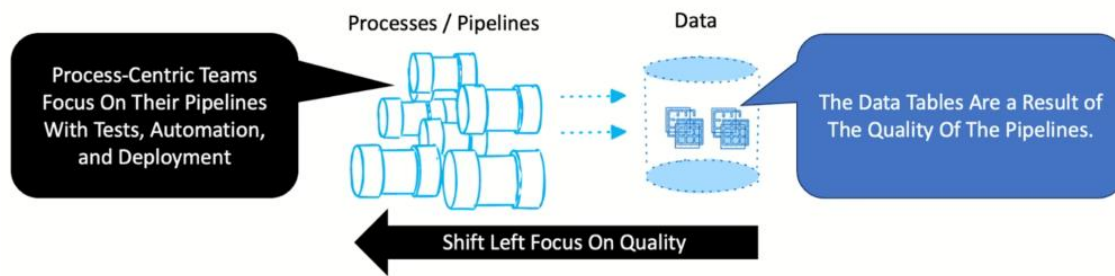


Over the years of working with data analytics teams in large and small companies, we have been fortunate enough to observe hundreds of companies. We want to share our observations about data teams, how they work and think, and their challenges. We've identified two distinct types of data teams: process-centric and data-centric. Understanding this framework offers valuable insights into team efficiency, operational excellence, and data quality.



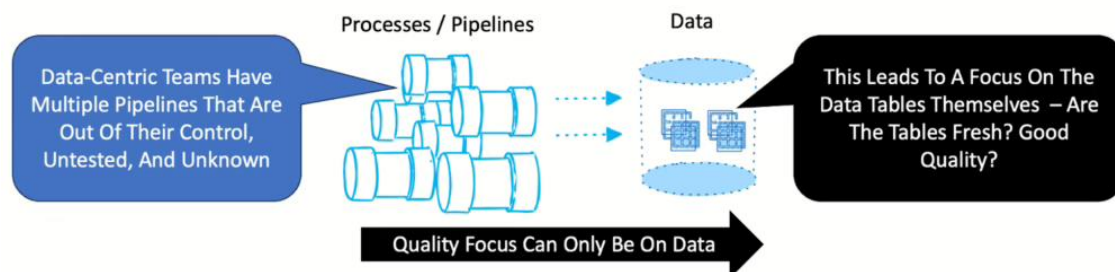
Process-centric data teams focus their energies predominantly on orchestrating and automating workflows. They have demonstrated that robust, well-managed data processing pipelines inevitably yield reliable, high-quality data. Rather than concentrating on individual tables, these teams devote their resources to ensuring each pipeline, workflow, or DAG (Directed Acyclic Graph) is transparent, thoroughly tested, and easily deployable through automation.

These teams excel because they embrace process visibility and control, believing firmly in the principles of DataOps. Their data tables become dependable by-products of meticulously crafted and managed workflows. As a result, complexity, in their view, stems from managing and orchestrating multiple processes, not from an abundance of tables. Each workflow is managed systematically, simplifying the integration of new data sources. Adding tables within an existing pipeline is manageable, posing minimal disruption. Every pipeline has embedded data quality tests, is version controlled, and is a sharable abstraction for the team to work within and deploy with low risk.



Data tables hold no emotional attachment for process-centric teams—they treat their pipelines as cattle, not pets. This mindset allows them to quickly diagnose and rectify issues at the process level rather than getting bogged down in individual data table anomalies. Their confidence comes from total control over when and how data updates occur, thanks to a unified orchestrator system. Process visibility is paramount, and data engineers are less concerned about the granular status of individual tables. Instead, their primary success metric is whether their processes run smoothly and without errors. They work in and on these pipelines. These teams, although rare, consistently achieve outstanding productivity and superior data quality.

In contrast, data-centric data teams face significant challenges due to a fundamentally different operational approach. Often burdened with numerous inherited, unmanaged, or unknown workflows, these teams are responsible for a sprawling network of data tables influenced by unknown or uncontrolled processes. Their databases feel like mysterious, daunting black boxes full of tables updated at seemingly random frequencies.



Data-centric data teams perceive complexity as directly proportional to the number of tables they manage. Each additional table intensifies their workload because visibility into data status, quality, and timeliness across hundreds or even thousands of tables is essential yet rarely achievable. Data engineers in these environments frequently react to quality issues without direct control over the processes causing the errors. This reactive stance hampers efficiency, forcing teams to patch or manually adjust data without resolving underlying process issues.

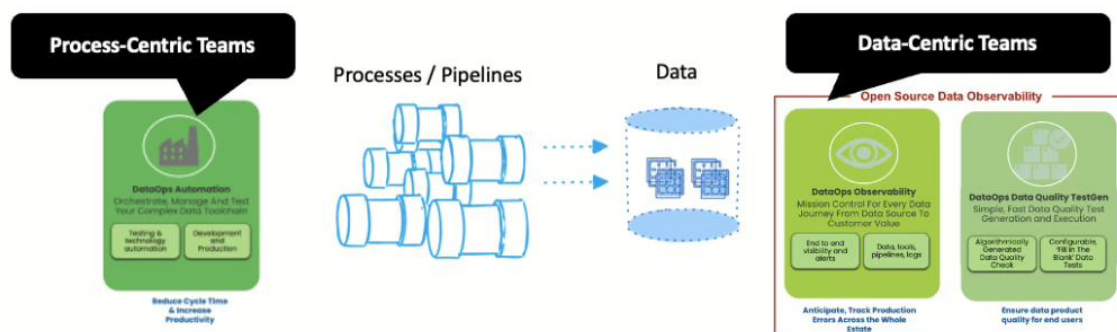
Unlike their process-focused counterparts, data-centric teams grapple with multiple orchestrators or job runners, exacerbating confusion and complicating efforts toward unified oversight. Many of these processes are out of visibility and control. They are held responsible for data errors but can't fix or influence the pipelines that made them. These teams may be familiar with DataOps practices but struggle to implement them effectively due to time constraints, resource limitations, and demanding customers. Consequently, productivity falters, data quality remains unreliable, and the team's morale and effectiveness decline.

Most data teams fall into this data-centric category, plagued by unclear data updates, inconsistent quality, and fragmented processes. These teams endure environments characterized by thousands of tables with uncertain update schedules and questionable accuracy. They may only control a few of their data input processes. The cumulative result is a pervasive inability to reliably answer the fundamental question: "Are our tables in good shape?" And it's harder still to answer the more important question: "Will they be in good shape tomorrow?"

Reflecting on these two paradigms provides clarity for data engineering leadership on how best to structure, empower, and optimize their teams. Though challenging, transitioning toward a process-centric approach is achievable, yielding significant benefits in data quality, operational efficiency, and team satisfaction. Leaders committed to this transformation must prioritize unified orchestration, process transparency, and rigorous adherence to DataOps principles.

Fortunately, no matter what type of team you currently belong to, there are actionable ways to adopt and integrate DataOps principles. Small, incremental steps such as consolidating orchestration tools, creating and implementing automated tests, and increasing process observability can drive meaningful progress. Embracing DataOps practices is a journey that will enhance your team's capabilities, improve data reliability, and significantly boost morale and productivity. The path to better data management is accessible and rewarding, regardless of your starting point. Together, you and your team can successfully transform your approach and enjoy the benefits of becoming more efficient, productive, and confident in your data operations.

DataKitchen started promoting DataOps by first building its DataOps Automation Product, a fantastic tool for process-centric data teams. We have also used it with DataKitchen's Managed customers for years, achieving excellent quality and fantastic data engineer productivity.

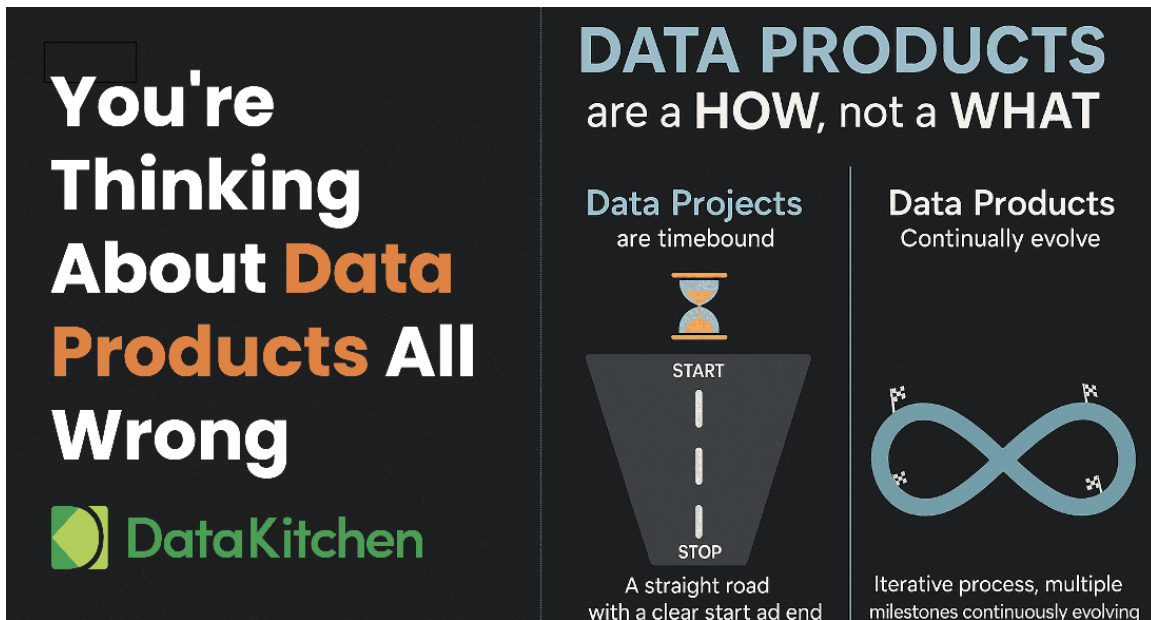


Author: Chris Bergh | Published: Mar 20, 2025

Original article: <https://datakitchen.io/unlocking-data-team-success-are-you-process-centric-or-data-centric/>

You're Thinking About Data Products All Wrong

When data team leaders hear "data products," they immediately think of the stuff their team creates. But focusing on the "what" completely misses the mark. Data products aren't about what you create, but "how" you build, maintain, and continually improve your data deliverables to your customers.



When data team leaders hear "data products," they immediately think of the stuff they produce: dashboards, datasets, models, and warehouses—but focusing on the "what" completely misses the massive shift that data products really represent. The truth is, data products aren't about what you create, but "how" you build, maintain, and continually improve your data assets to deliver value to customers.

The current discussion around data products, as reflected in countless articles and even AI-generated definitions, tends to describe them as "reusable, trusted, and well-defined data assets—such as a dataset, metric layer, feature set, or dashboard—that is designed, built, and managed like a product." This definition gets us halfway there, but it still emphasizes the artifact rather than the approach.

Current Discussion of Data Products ...

(From ChatGPT)

A data product is a reusable, trusted, and well-defined data asset—such as a dataset, metric layer, feature set, or dashboard

... that is designed, built, and managed like a product.

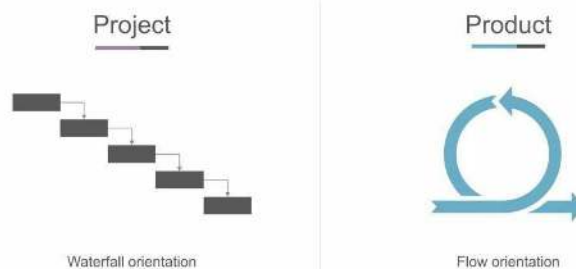
Copyright 2025 by DataKitchen, Inc. All Rights Reserved.

DataKitchen

When we think about data products merely as components or deliverables, we fall into the same trap that has plagued data teams for decades: the [project mindset](#). This perspective leads us to view our work as a series of discrete efforts with defined endpoints, budgets to meet, and boxes to check. However, this approach fundamentally misaligns with how modern organizations need to utilize data for a competitive advantage.

The shift from projects to products represents a fundamental change in how we approach data work. Unlike projects, which have a clear beginning and end, products are living entities that evolve continuously in response to user feedback and changing business needs. This isn't just semantics—it's a complete reimagining of how data teams operate.

Data Projects to Data Products



From Tasktop

Copyright 2025 by DataKitchen, Inc. All Rights Reserved.

DataKitchen

In the project world, we follow a waterfall orientation where work flows sequentially through stages. Teams are brought to specific projects, often creating silos and handoffs that slow down delivery and reduce quality. The focus becomes meeting deadlines and staying within budget, rather than delivering ongoing value to users.

The ‘thing’ that is in a data product could be any deliverable your team does. Defining the requisite components of data products is like counting angels on a pinhead.

In contrast, the product approach creates a continuous flow of value. Work is brought to stable teams who own their products end-to-end. These teams measure success not by whether they delivered on time, but by whether their users are satisfied and receiving value. The product mindset embraces iteration, learning, and continuous improvement.

This transformation in data isn’t happening in isolation. It mirrors a revolution that already transformed software engineering over the past two decades. Software teams have transitioned from DevOps to DataOps, from monolithic architectures to domain-driven design, and from project-based delivery to product-driven thinking. Now, these same principles are revolutionizing how we approach data and analytics.

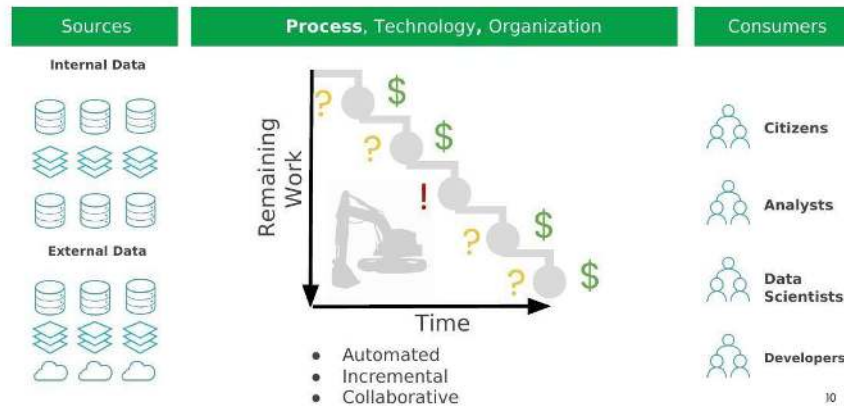
From Software Engineering to Data Analytics	
Software Engineering	Data & Analytics
• DevOps	-> DataOps
• Domain-Driven Design	-> Data Mesh
• Software Products	-> Data Products
• Test-driven Development	-> Data Testing
• Observability	-> Data Observability
• Event-Driven Architecture	-> Streaming; Kappa Architecture
• API	-> Data Contract
• Functional Programming	-> Functional Data Engineering

2025 by DataKitchen, Inc. All Rights Reserved. 

Consider how software teams transformed their practices: Test-driven development evolved into data-driven testing. Observability became data observability. Event-driven architectures evolved into streaming and Kappa architectures. APIs transformed into data contracts. Each of these transitions represents not just a technology change, but a fundamental shift in thinking about how we deliver value.

One of the most crucial aspects of the product mindset is understanding that data products are never “done.” This might feel uncomfortable for teams accustomed to celebrating project completions, but it’s actually liberating. Instead of rushing to meet an arbitrary deadline with a potentially suboptimal solution, teams can focus on iterative improvement, learning from each release to make the next one better.

Data Products Are Continuously Improved



Copyright 2025 by DataKitchen, Inc. All Rights Reserved.

DataKitchen

This continuous improvement approach yields measurable benefits: faster releases of analytics products, increased team learning, better insights, and ultimately, faster delivery of value from your data investments. Teams operating in this mode don't just deliver once and move on—they establish feedback loops with their users, monitor usage patterns, and continuously refine their deliverables. With the traditional project approach, you may invest heavily up front with uncertain returns. With the product approach, you can start small, prove value quickly, and scale based on actual user adoption and feedback. Each iteration builds on the last, creating compound returns on your data investments.

The ultimate goal of adopting a data product mindset is to maximize the flow of value to your customers. This isn't just about moving faster—it's about creating a sustainable and predictable delivery of high-quality data deliverables that users actually want and use.

Data Products Maximize Team **Flow**

Improving and measuring the 'flow' of value to your customer is the #1 goal of Data Products



- What is the 'flow' of value to your customer?
 - New data sets
 - Improved models/visualizations/data
 - New code/configuration
- Key measurements
 - Cycle time of deployment ('flow time')
 - Success Rates of Deployment ('flow efficiency')
 - Work done ('flow velocity')

Copyright 2025 by DataKitchen, Inc. All Rights Reserved.

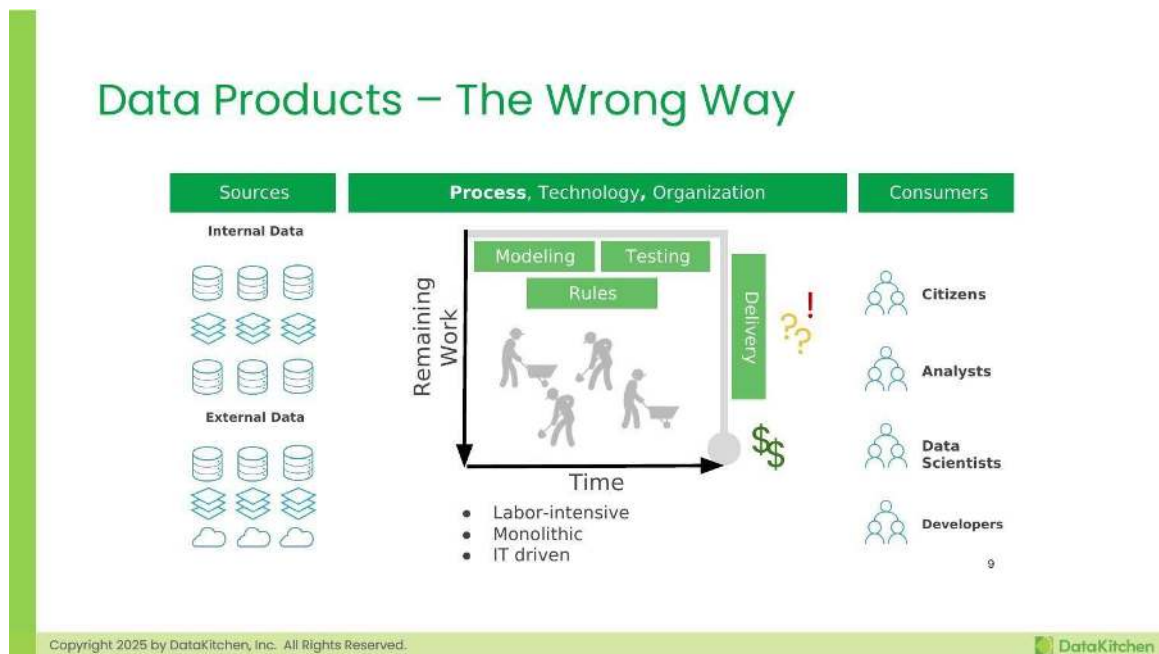
DataKitchen

Flow encompasses several key dimensions. Flow time measures how quickly you can deliver new capabilities from idea to production. Flow efficiency tracks your success rates, ensuring that the work you do actually reaches users

rather than getting stuck in approval processes or failing in production. Flow velocity measures the volume of valuable work completed, helping teams understand and improve their capacity over time.

These metrics matter because they directly correlate with business value. Reducing flow time means faster response to market changes. Improving flow efficiency means less wasted effort and more predictable delivery. Increasing flow velocity means more value delivered to users over time.

Many organizations still approach data products incorrectly, treating them as large, monolithic efforts driven primarily by IT teams. This approach is characterized by heavy manual processes, long delivery cycles, and a disconnect between what's built and what users actually need.



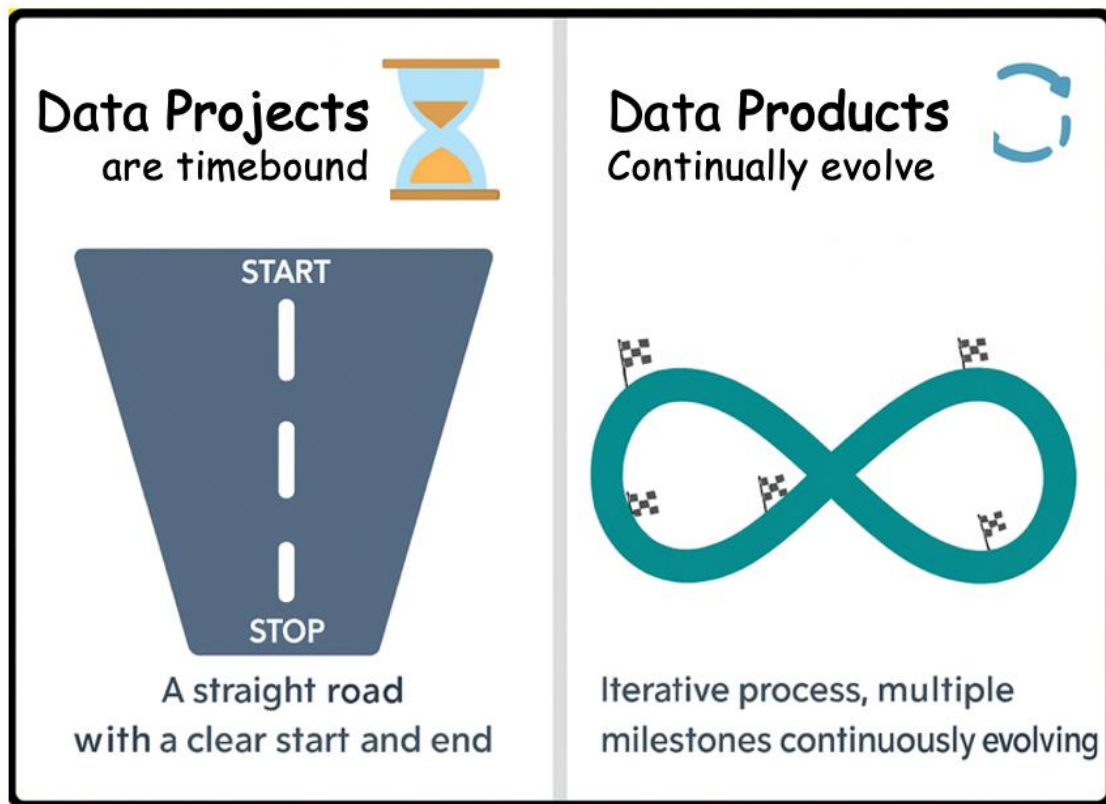
In this model, work flows slowly through various stages—modeling, rules definition, [testing](#)—with significant manual effort required at each step. The result is uncertainty about the timeline, cost, and whether the final product will actually meet user needs. Teams operating this way often find themselves in a reactive mode, constantly fighting fires rather than proactively improving their products. And many times, they miss the mark on what their users actually need.

“Our project was a success; we followed our SLDC and got an award.” Yeah, but did the customer use it? Did they find it valuable?

The product approach completely transforms this dynamic. Instead of significant, risky releases, teams deliver minor, incremental improvements continuously. Automation replaces manual processes wherever possible, freeing teams to focus on innovation rather than maintenance. Most importantly, the approach is inherently collaborative, bringing together data engineers, analysts, scientists, and business users to pursue shared outcomes.

This collaborative approach extends beyond just the data team. Product thinking requires close partnership with users, regular feedback sessions, and a willingness to pivot based on what you learn. It means treating your internal data consumers as valued customers whose satisfaction determines your success.

Transitioning from a project to a product mindset isn't just about adopting new tools or processes—it's about fundamentally changing how your organization thinks about data and information. This shift requires leadership commitment, cultural change, and sometimes, investment in new capabilities.



Start by identifying a small, high-value use case where you can demonstrate the product approach. Choose something with engaged users who can provide regular feedback. Build a cross-functional team that can own the deliverable from end to end. Establish clear metrics for success based on user outcomes rather than delivery milestones.

As you begin delivering value through this approach, use your success to build momentum. Share your learnings broadly and celebrate not just releases, but also improvements in your flow metrics. Help other teams understand how the product mindset can enable them to deliver more value with less stress and uncertainty.

While the product mindset is primarily about approach rather than technology, having the proper technical foundation certainly helps. This includes capabilities for continuous integration and deployment, automated testing, monitoring and observability, as well as robust data governance. Embracing a [DataOps approach](#), which applies agile and DevOps principles to data management, can significantly enhance this foundation. DataOps fosters collaboration between data engineers, data scientists, and other stakeholders, streamlining the entire data lifecycle from data ingestion to consumption. The goal is to create an environment where teams can safely and quickly iterate on their products without fear of breaking critical systems.

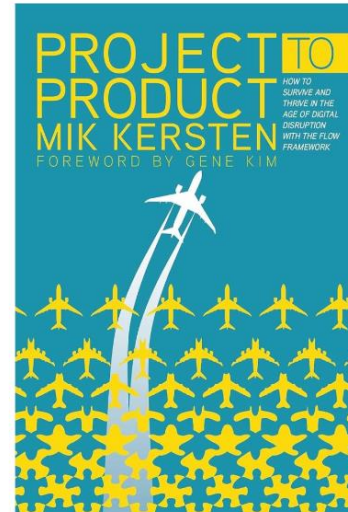
In the product world, success metrics shift dramatically from project metrics. Instead of asking “Did we deliver on time and on budget?” we ask “Are our users successful? Are they recommending our team to others?” This shift in metrics drives a shift in behavior, focusing teams on long-term value rather than short-term delivery.

[Establish metrics that matter.](#) This might include data quality scores or business-specific metrics, such as “time to insight” or “decisions influenced.” Make these metrics visible to your team and stakeholders, creating transparency around the value you deliver.

Making this transformation from projects to products, from “what” to “how,” isn’t a journey you have to take alone. DataKitchen specializes in helping organizations adopt DataOps practices that enable the product mindset for data teams. Our consulting services can help you assess your current state, design your transformation roadmap, and build the capabilities you need to succeed.

Data Products Are A 'How' Not A 'What'

- **Data Products are a method, not a component list**
 - They do not have an end (like a project)
 - They are iteratively improving
 - Are focussed on delivering value, not meeting budget
 - Outcome oriented, not data oriented
 - Can be any type of deliverable
 - Success is defined by customer satisfaction, not on time delivery or internal process followed.



Through our open-source tools and proven methodologies, including our [FITT \(Functional, Idempotent, Tested, Two-Stage\) Data Architecture framework](#), we provide the technical foundation and practical guidance needed to make this shift successfully. We've helped dozens of organizations transform their data teams from cost centers to value drivers, from project factories to product organizations.

The shift from thinking about data products as a “what” to understanding them as a “how” is more than just a philosophical exercise—it's a practical transformation that can dramatically improve your team's effectiveness and job satisfaction. By embracing the product mindset, you're not just changing how you deliver data assets; you're changing how your organization creates value from its data. The question isn't whether to make this shift, but how quickly you can begin the journey. [Give us a call!](#)

Author: Chris Bergh | Published: Oct 31, 2025

Original article: <https://datakitchen.io/youre-thinking-about-data-products-all-wrong/>

Process Guardianship: The Most Valuable Data Engineering Work You're Probably Not Doing

When people think of data engineers, the description usually stops at “building high-quality pipelines that deliver analyst-ready data.” That is true, but incomplete. In modern organizations, data engineers hold a deeper responsibility. They are not just the builders of pipelines—they are the curators of the business logic itself.



There's a quiet crisis happening in data organizations everywhere. Not the dramatic kind that makes headlines—no security breaches, no system failures. Instead, it's a slow erosion of value that occurs when brilliant analysts independently solve the same problem in three different ways, when data scientists struggle to reproduce each other's work, and when a single “simple” metric, such as Monthly Recurring Revenue, has five competing definitions scattered across the organization.

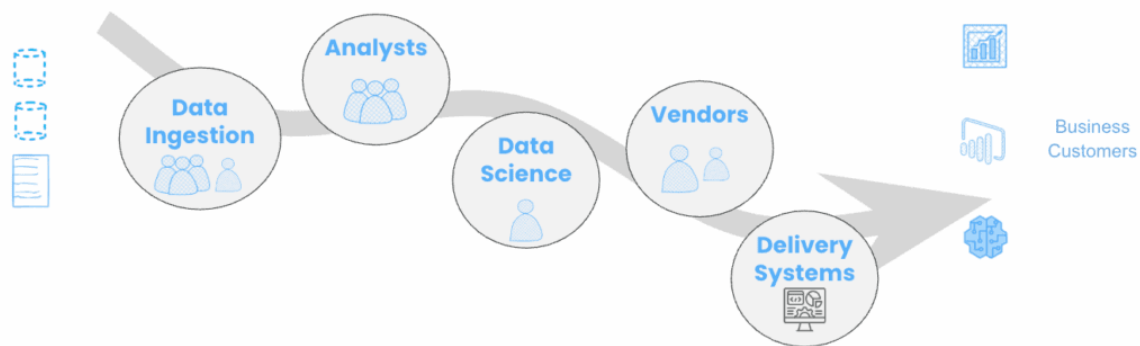
I've watched data teams double their headcount while their actual impact plateaus. The problem isn't talent or technology—it's the absence of what I call process guardianship: the deliberate, systematic work of consolidating how business logic lives and operates within an organization.

For data engineers reading this, I want to make a case that might initially feel uncomfortable: your job isn't finished when you deliver clean, analyst-ready data. In fact, that's where some of your most valuable work begins.

What Process Guardianship Actually Means

Process guardianship is the practice of curating and consolidating disparate metrics, models, calculations, and code into a unified, tested, and highly agile data pipeline. It means taking the business logic that naturally scatters itself across SQL scripts, Jupyter notebooks, Looker reports, Python models, and Excel macros, and deliberately centralizing it into a maintainable, version-controlled, production-grade system.

This isn't just about code hygiene or engineering best practices. It's about recognizing that the real intellectual property of your data team—the accumulated knowledge of how your business actually works—shouldn't live as tribal knowledge on the laptops of team members around the world.

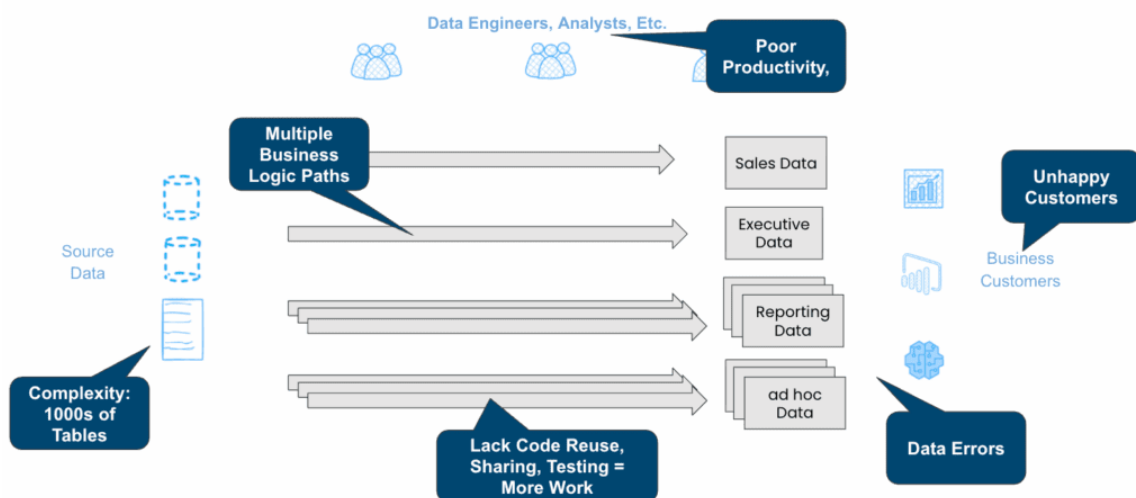


Think about the business logic in your organization right now. Where does the calculation for customer lifetime value actually live? Is it in the Excel macro on Bob's laptop on the finance team? Or in the segmentation model that the data science team built? Or in the SQL query your senior analyst wrote last quarter? The answer, more often than uncomfortable, is "all three places, and they don't quite match."

The Hidden Cost of Scattered Business Logic

Your marketing analyst builds a brilliant customer segmentation model. It works beautifully for their campaign targeting. Three months later, your data scientist builds another segmentation model for churn prediction—unaware that the first one exists. Six months after that, finance needs customer segments for their forecasting model and builds yet another version because they can't find or don't trust the existing two. Now you have three models that: use slightly different definitions of "active customer," handle edge cases differently, produce overlapping but inconsistent results, and require separate maintenance when business rules change

When leadership asks, "How many customers do we have in our high-value segment?", you get three different answers. This isn't a hypothetical scenario—this is Tuesday afternoon in most data organizations.



Duplicate work becomes the norm. Teams waste countless hours reinventing wheels because they don't know what already exists or can't trust it enough to reuse it.

Multiple overlapping metrics create confusion and erode trust. When "monthly active users" means something different in the product dashboard versus the board deck, stakeholders stop believing in your numbers.

Embarrassing data errors slip through. The marketing team launches a campaign targeting "high-value customers" based on one definition. At the same time, finance forecasts revenue using another method, and nobody notices until

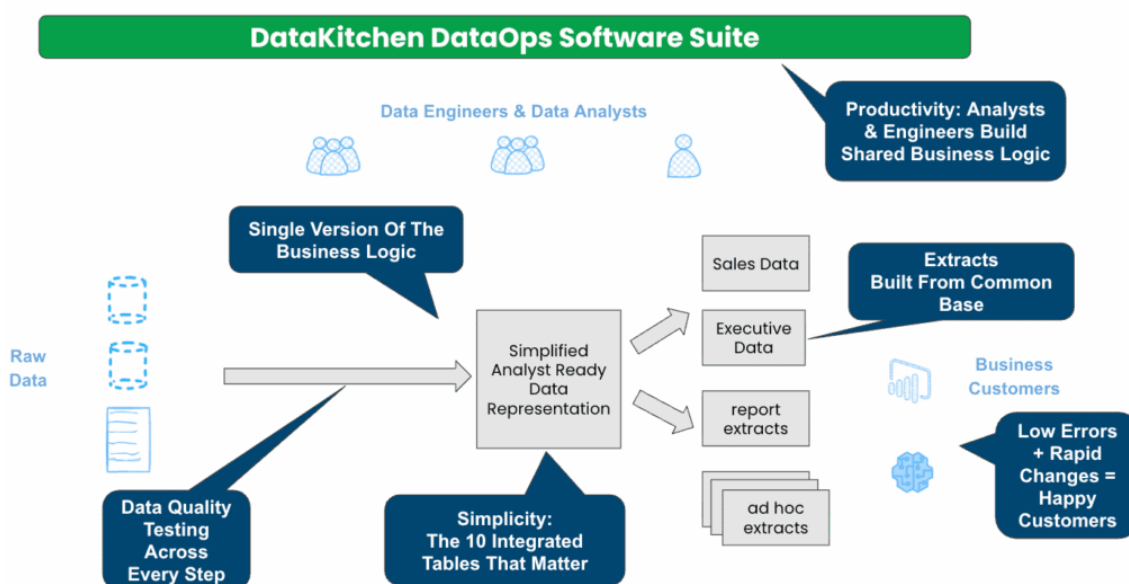
the numbers don't add up at the end of the quarter. Or legitimate corrections identified by one team aren't vetted or shared with the enterprise.

Lower data team productivity becomes the steady state. Your talented people spend more time reconciling discrepancies, explaining why numbers don't match, and rebuilding things that should already exist than they do creating new value. Newer employees or vendors often struggle to get up to speed quickly.

The Data Engineer as a Process Guardian

Data engineers have a unique position in the extended data team. You sit at the intersection of raw data and business consumption. You understand both the technical infrastructure and the analytical workflows. You see the patterns—you're the ones who notice when three different teams ask for "basically the same thing" in three different ways. Data engineers are the first to identify duplicate code and are ideally positioned to make recommendations.

Process guardianship means actively stepping into that space and saying: "This business logic is too important to live in scattered notebooks and ad-hoc queries. Let's consolidate it, test it, and make it production-grade."



Extracting business logic from its hiding places. That incredible revenue forecasting model your data scientist built in a notebook? That complex customer scoring logic buried in a Looker view? That critical date logic that exists only in your senior analyst's SQL queries? These need to be identified, documented, and brought into a centralized system.

Creating a unified, tested pipeline. Business logic should reside in version-controlled code, accompanied by thorough testing, comprehensive documentation, and rigorous review processes. When your churn model definition changes, it should change in exactly one place, with a clear history of what changed and why. With source code version control.

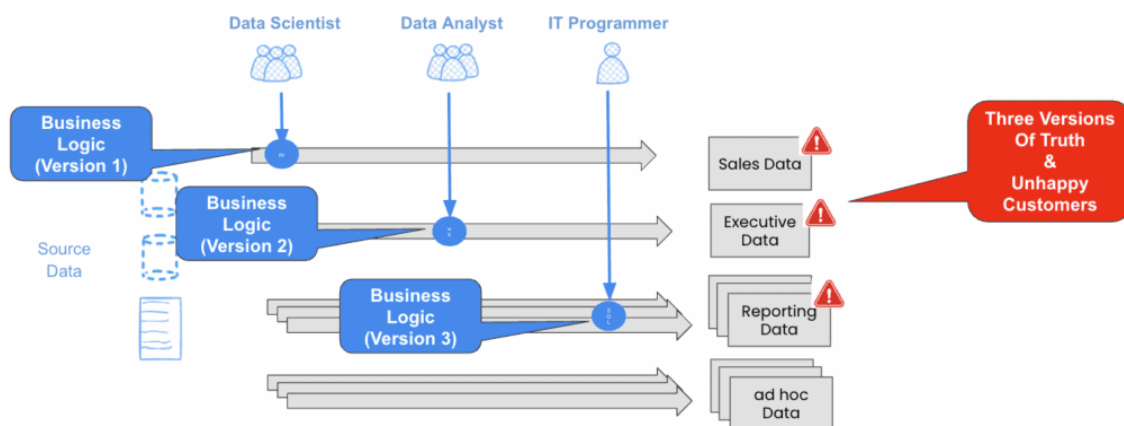
Building for agility and experimentation. Done right, process guardianship creates a foundation that enables experimentation to be faster and safer. When analysts and data scientists can trust the foundational business logic and metrics, they can focus on asking new questions rather than rebuilding the basics.

Enabling smooth transitions from experiment to production. When that experimental model or analysis proves valuable, moving it into production shouldn't require a complete rewrite. The path from notebook to production pipeline should be clear and well-traveled.

The Reality: Less Work Than You Think

Here's something that might surprise you: from a pure engineering hours perspective, this work is often a small fraction of your total effort. You're not rebuilding everything from scratch—you're consolidating what already exists and creating structure around it.

The actual technical work might look like: converting that critical SQL logic from a report into a dbt model with tests, refactoring a data science model in Python from a notebook into a versioned, parameterized pipeline, creating shared metric definitions in your semantic layer that replace dozens of slightly different calculations, building lightweight frameworks that make it easy for analysts to contribute tested, reusable logic, or building a new fact in a shared fact table.



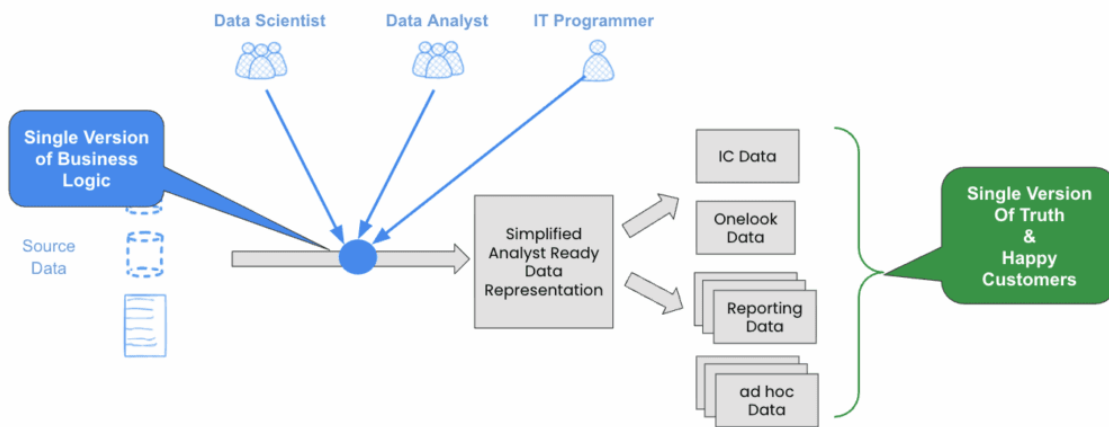
This isn't months of architectural redesign. It's often days or weeks of focused consolidation work. But the impact is wildly disproportionate to the time invested. Let me describe what this looks like in a well-functioning data organization.

When a data scientist wants to build a new churn prediction model, they don't start by defining what "churned" means—that definition already exists in the centralized data pipeline, tested and agreed upon by the business. They can focus their improvement on the actual working code with tests.

When finance needs to create a new forecast, they pull from the same customer segmentation, revenue recognition rules, and metric definitions that marketing and product are already using. The numbers match by default, not by accident.

When a critical business rule changes—such as how you handle returns in your revenue calculations—that change occurs in one place, propagates automatically, and every downstream consumer receives the update without manual intervention.

When an analyst builds something experimental that proves valuable, the path to productionizing it is straightforward because it was built on top of tested, production-grade foundations from the start.



When You Curate A Single Version Of The Business Logic, You Get A Single Version Of The Truth In Your Data

When You Curate A Single Version Of The Business Logic, You Get A Single Version Of The Truth In Your Data

The [data team's collective knowledge](#) isn't locked in individual heads or scattered across folders. It's codified, tested, documented, and accessible. New team members can onboard more quickly because they're learning from a centralized, maintained body of work rather than searching through old tickets and Slack messages.

Why This Matters

Stakeholder trust increases when numbers are consistent across different teams and tools. When executives see the same metrics in every dashboard, calculated the same way, they stop questioning the numbers and start acting on them.

Decision velocity improves because people are no longer waiting for analysts to rebuild basic metrics or reconcile conflicting definitions. The foundational layer is solid, allowing analysis to occur more quickly.

Experimentation becomes safer because there's a clear distinction between production logic (tested, reviewed, maintained) and experimental work (rapid, flexible, creative). You can move fast without breaking things.

Knowledge persists when key team members leave. The business logic they understood isn't lost—it's captured in the centralized system, so there's no more tribal knowledge.

Compliance and governance become manageable. When you need to audit how a metric is calculated or ensure specific data handling rules are followed, having logic centralized makes this dramatically simpler.

Streamlined compliance and governance. Auditing metric calculations or enforcing data handling rules is significantly simplified with centralized logic.

"We're too busy building new pipelines to worry about consolidating existing logic." This is the trap. You'll always be too busy if you never invest in consolidation. The duplicate work and scattered logic are what's making you busy. Process guardianship is how you break the cycle.

"Our analysts need flexibility—centralized logic will slow them down." Good process guardianship increases flexibility. When analysts can trust foundational metrics and logic, they spend less time on basic plumbing and more time on actual analysis. The goal is to liberate analysts from production drudgery.

"This sounds like a lot of bureaucratic overhead." It doesn't have to be. This isn't about creating approval workflows or committees. It's about making it easier to do the right thing than the wrong thing. When contributing to centralized,

tested logic is simpler than copying someone else's query and modifying it, people naturally gravitate toward it. It is about refactoring.

"Our business changes too fast for this to work." Actually, the opposite is true. When business logic is scattered, every change requires hunting down every instance and updating every one. When it's centralized, changes happen in one place and propagate automatically. Process curation makes you more agile, not less.

Ultimately, process guardianship is as much about mindset as it is about technical practice. It requires seeing your role as a data engineer not just as building pipelines, but as a steward of how your organization's business logic lives and evolves.

Typical "Scattered Processes"	With Process Guardianship
Customer Visible Data Errors	Infrequent Data Errors
Scattered, Independent Business Logic	Centralized 'Recipes' Of Business Logic, Git Based
Multiple Versions Of The Truth	Unified Version Of The Data Team's Collective Intelligence
Slow Development Of New Feature	Fast Development Of New Features
Poor Data Analytic Team Productivity	High Team Productivity, Less Waste
Slow Turn Around Time Of New Features/Data	Fast Iterations, Low Risk Of Breaking Existing Features

It means asking questions like: Does this calculation exist anywhere else? Should this logic live in a production pipeline instead of a notebook? How will we maintain this when business rules change? What happens when the person who wrote this leaves the company?

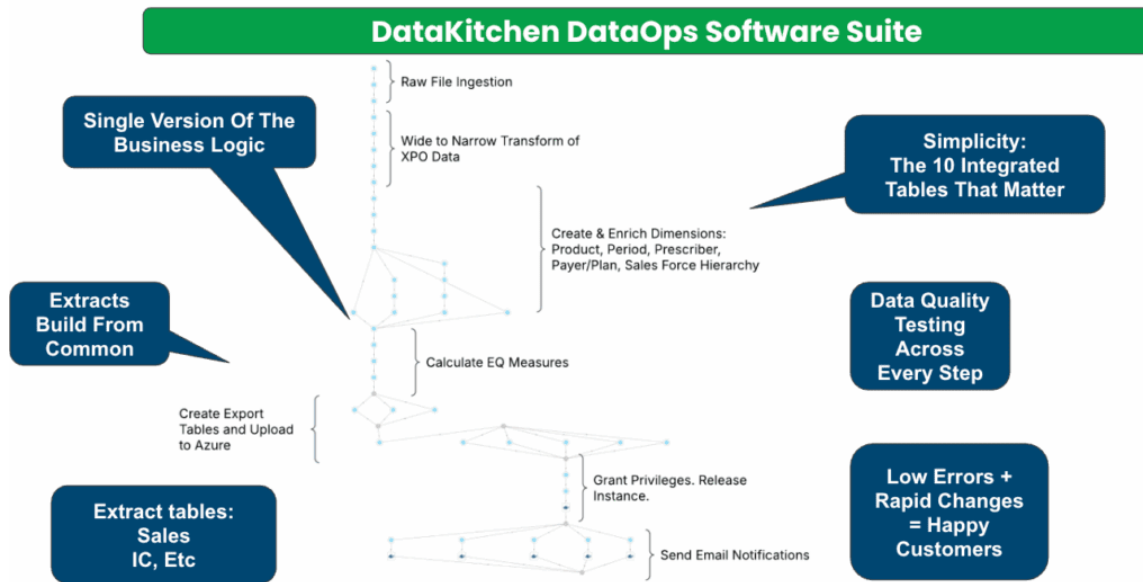
It means having the conviction to say "before we build this, let's consolidate what already exists," even when there's pressure to ship something new. It means advocating for the unsexy work of consolidation and refactoring, knowing that it's often more valuable than the flashy new project.

In Complex Organizations, This Becomes Invaluable

In small organizations with simple data needs, you can sometimes get away without deliberate process guardianship. Everyone knows everyone, communication is easy, and the surface area is small enough to keep in your head.

As organizations grow in size and complexity—adding more data sources, teams, stakeholders, and systems—the chaos caused by scattered business logic becomes exponentially more costly. The importance of having someone actively guard these processes shifts from "nice to have" to "absolutely critical."

Data engineers who practice process guardianship become force multipliers. The time investment is relatively small compared to other data engineering tasks. But the impact—on data quality, team productivity, stakeholder trust, and organizational agility—is enormous.



Be the Guardian Of Your Team's Collective Intelligence

When people think of data engineers, the description usually stops at “building high-quality pipelines that deliver analyst-ready data.” That is true, but incomplete. In modern organizations, data engineers hold a deeper responsibility. They are not just the builders of pipelines—they are the curators of the business logic itself.

This guardianship role involves consolidating fragmented definitions of the business—metrics scattered across dashboards, SQL scripts, notebooks, and models—into a unified, tested, performant, and agile data production line. It involves extracting business logic from hidden corners and tribal knowledge, then encoding it in a centralized, version-controlled, and production-ready system.

The payoff? More consistent decisions, fewer errors, and higher productivity across the extended data team.

Process guardianship means making it a core part of how your data team operates. It means recognizing that your job isn't done when the data is clean and accessible—it's done when the business logic that acts on that data is consolidated, tested, and maintainable. It means seeing yourself not just as a builder of data infrastructure, but as a guardian of your organization's collective intelligence about how to make sense of that data.

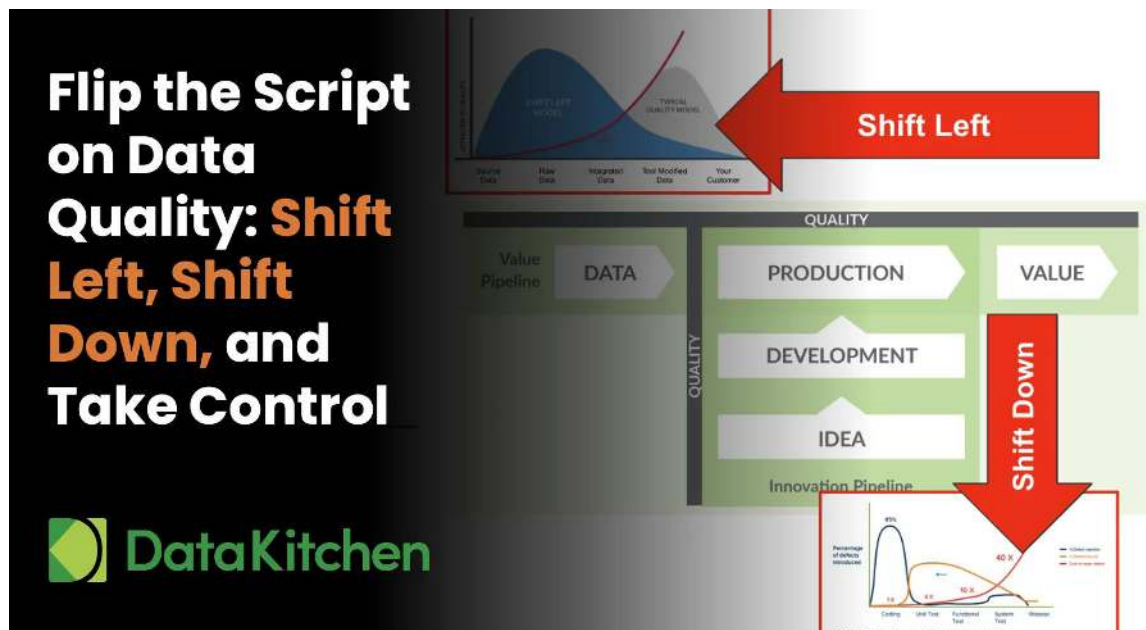
For data engineers, this is an opportunity to expand your impact beyond technical infrastructure into the strategic heart of how your organization operates. The IP of your data team—the accumulated knowledge of what metrics matter and how to calculate them, what models work, and how to apply them—is too valuable to scatter. It needs a guardian.

Author: Chris Bergh | Published: Oct 8, 2025

Original article: <https://datakitchen.io/process-guardianship-the-most-valuable-data-engineering-work-youre-probably-not-doing/>

Flip the Script on Data Quality: Shift Left, Shift Down, and Take Control

The manufacturing industry learned decades ago that catching defects early in the production process saves exponentially more money than fixing them after products ship. Today's data engineering teams face a strikingly similar challenge.

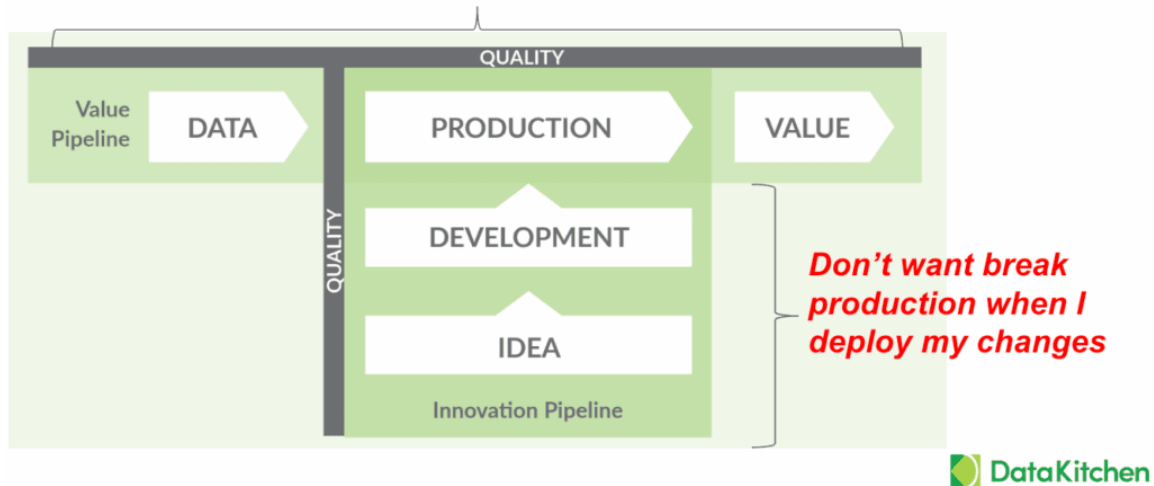


There has been considerable discussion lately about data contracts and [the shift-left](#) approach in data and analytics systems. The manufacturing industry learned decades ago that catching defects early in the production process saves exponentially more money than fixing them after products ship. Today's data engineering teams face a strikingly similar challenge, yet many organizations continue to discover data quality issues only after their analytics have already influenced critical business decisions. The concepts of “shift-left” and “shift-down” testing, borrowed from manufacturing and software development, respectively, offer a powerful framework for fundamentally rethinking how we approach quality in data and analytics systems.

The Assembly Line Parallel: Understanding Data as Manufacturing

Modern data pipelines bear a remarkable resemblance to manufacturing assembly lines. Raw materials enter the system in the form of source data. These materials flow through various transformation stages, involving Python code for data access, SQL and ETL processes for transformation, and R or Python for modeling, as well as visualization tools like Tableau for creating consumable outputs. At each stage, the data undergoes processing that adds value while also introducing potential points of failure.

Don't want to learn about problems from my customers



Just as a manufacturing line produces refined products from raw materials, data pipelines produce charts, graphs, models, and reports from raw data. The parallel extends further when we consider quality control. In manufacturing, quality checks at each stage of production prevent defective components from moving downstream. In data systems, similar checkpoints can prevent corrupted, incomplete, or incorrectly transformed data from propagating through the pipeline and ultimately reaching business users.

This manufacturing mindset reveals two critical dimensions along which we must shift our testing efforts. The horizontal dimension represents the flow of data through production pipelines, where “shifting left” means moving quality checks earlier in the data production lifecycle. The vertical dimension represents the flow of code and configuration changes from development to production, where “shifting down” refers to implementing comprehensive testing before code changes are deployed to production systems.

‘Shift Left’ executing data quality tests earlier in the data pipeline lifecycle—so issues are identified and resolved before they propagate downstream.



‘Shift Down’ use integration and end-to-end testing methodologies, validating not just isolated components but also the interactions and cumulative impacts within the entire data ecosystem.



Shift Left: The Economics of Early Detection in Production

The concept of shifting left in data quality testing draws inspiration from the Japanese manufacturing concept of the “[Andon Cord](#),” a mechanism that allows any worker on an assembly line to halt production when they detect a quality issue. In data systems, this translates to implementing automated data tests and tool checks at each stage of the pipeline, with the ability to stop data flow when errors are detected.

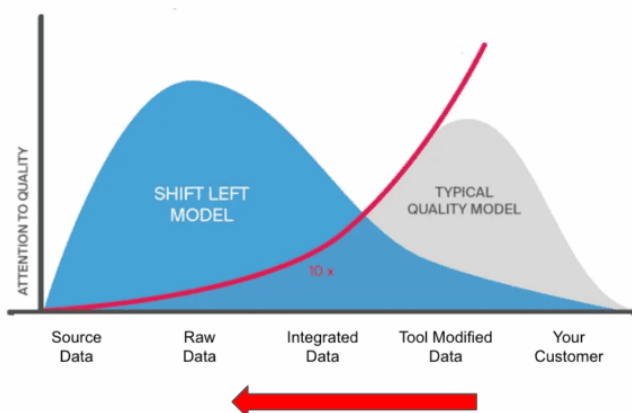
The economic argument for shift-left testing is compelling and aligns with the 1:10:100 rule, as identified by quality experts George Labovitz and Yu Sang Chang. In the context of data quality, this rule suggests that preventing poor data quality at the source costs approximately one dollar per record, remediating data quality issues after the data has been created costs ten dollars per record, and the cost of failure when poor quality data reaches production and influences decision-making approaches is one hundred dollars per record.

Shift Left Test Coverage Saves Time

It is 10x cheaper to find a problem in data at the source or raw layer.

Apply the 1:10:100 rule*:

- Cost of preventing poor data quality at source is \$1 per record
- Cost of remediation after it is created is \$10 per record
- Cost of failure (i.e. doing nothing) is \$100 per record



*developed by George Labovitz and Yu Sang Chang back in 1992

This exponential cost increase occurs because, as data moves through the pipeline, it becomes increasingly integrated with other datasets, embedded in downstream calculations, and ultimately influences business decisions. A simple data type mismatch caught at ingestion might take minutes to fix. The same issue discovered after it has corrupted aggregate calculations, been cached in multiple systems, and influenced executive reports could require days of investigation, correction, and communication to resolve.

Consider a practical scenario where a source system starts sending timestamps in a different timezone without warning. If caught at the ingestion point through shift-left testing, the fix involves updating a single transformation rule. If this issue goes unnoticed in the pipeline, it could lead to incorrect daily aggregations, skewed trend analyses, and fundamentally flawed business metrics. The remediation effort increases as teams must identify all affected downstream calculations, regenerate historical aggregates, and possibly explain incorrect reports that have already been distributed. Implementing shift-left testing requires a fundamental rethink of how we design data pipelines. Instead of viewing testing as an afterthought or a separate quality assurance phase, tests become essential parts of the data flow itself. This involves adding validation checks immediately after data ingestion to verify schema compliance, data type consistency, and adherence to business rules. It also includes performing statistical tests after transformation steps to ensure data distributions stay within expected ranges. Additionally, validating the outputs of machine learning models against baseline metrics before allowing predictions to influence downstream processes.

Shift Down: Preventing Deployment Disasters in Development

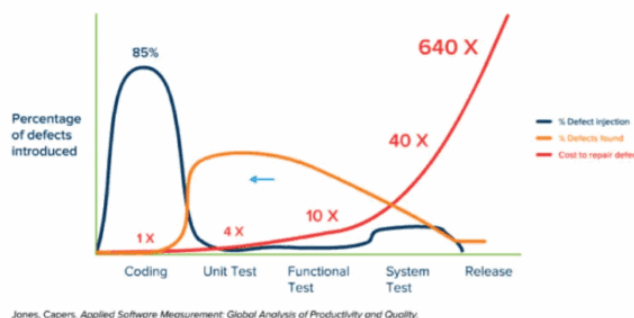
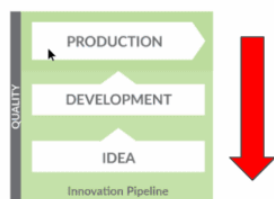
While 'shift left' addresses quality issues in the production data flow, 'shift down testing' focuses on preventing code and configuration changes from introducing new errors into production systems. This concept, borrowed from software development practices, emphasizes comprehensive testing during the development phase before changes are promoted to production.

The traditional approach in many data organizations involves what might charitably be called "hope-based testing." Developers make changes to SQL transformations, update Python scripts, or modify ETL configurations, perform cursory manual checks, and deploy to production with fingers crossed. This approach inevitably leads to a familiar

pattern: urgent calls about broken dashboards, frantic debugging sessions, and hasty rollbacks that disrupt business operations

Shift Down Test Coverage Saves Costs

The earlier your tests find a problem in development, the lower the cost and hassle



Shift down testing replaces hope with systematic validation. It includes various testing methods, each with a specific role in quality assurance. Integration tests confirm that multiple data processing steps work correctly together, ensuring data flows smoothly between different tools and transformations. End-to-end tests simulate complete data pipelines from ingestion to final output, confirming overall correctness across the entire system. These dynamic tests run on realistic test data that reflects production characteristics, executed in development environments that closely mimic production infrastructure.

The economic advantages of shift-down testing follow a similar exponential pattern to shift-left testing. Research consistently indicates that defects identified during coding are relatively inexpensive to fix, typically requiring only time to correct the code and rerun local tests. In contrast, defects detected during integration testing are significantly more costly, often requiring coordination between teams and necessitating changes to multiple components. Defects that reach production can be forty to six hundred times more expensive than those caught early, as they often necessitate emergency responses, hotfixes, data remediation, and damage control with stakeholders. Try [Data Architectures like FITT \(Functional, Idempotent, Tested, Two-Stage\)](#) to implement this with less hassle.

The Reality Gap: Where Teams Actually Focus

Despite the clear economic benefits of comprehensive testing, a stark reality gap exists between optimal testing practices and where data teams actually focus their efforts. Based on industry observations and DataKitchen's research, approximately eighty percent of data teams rely primarily on hope-based testing in development, with no automated validation before deploying changes to production. The remaining twenty percent might employ some static analysis tools, such as SQL linters, but dynamic testing that validates actual data transformations remains virtually nonexistent in most organizations.

Where **Should** Test Coverage Effort Focus?

		Production Testing		
Dev. Testing		'Avoid the Fact'	'After the Fact'	'Before the Fact'
	Hope	0%		
	Static Analysis		20%	
	Dynamic Testing			80%

Where Data Teams **Actually** Focus

		Production Testing		
Dev. Testing		'Avoid the Fact'	'After the Fact'	'Before the Fact'
	Hope	80%		
	Static Analysis		20%	
	Dynamic Testing			0%

On the production side, the situation is equally concerning. Most teams operate in what can be called “avoid the fact” mode, essentially waiting for customers or business users to report problems. Some organizations have implemented “after the fact” monitoring, continuously polling databases and checking data quality metrics; however, this approach still allows problems to reach production before they are detected. Only a small minority has implemented true “before the fact” testing that catches issues during the production process itself.

This misallocation of testing effort creates a vicious cycle. Teams spend the majority of their time fighting fires in production, leaving little bandwidth for implementing proper testing practices. The lack of testing leads to more production issues, which consume more time, further reducing the capacity for improvement. Breaking this cycle requires a deliberate shift in priorities and a recognition that investing in testing infrastructure pays dividends through reduced operational burden.

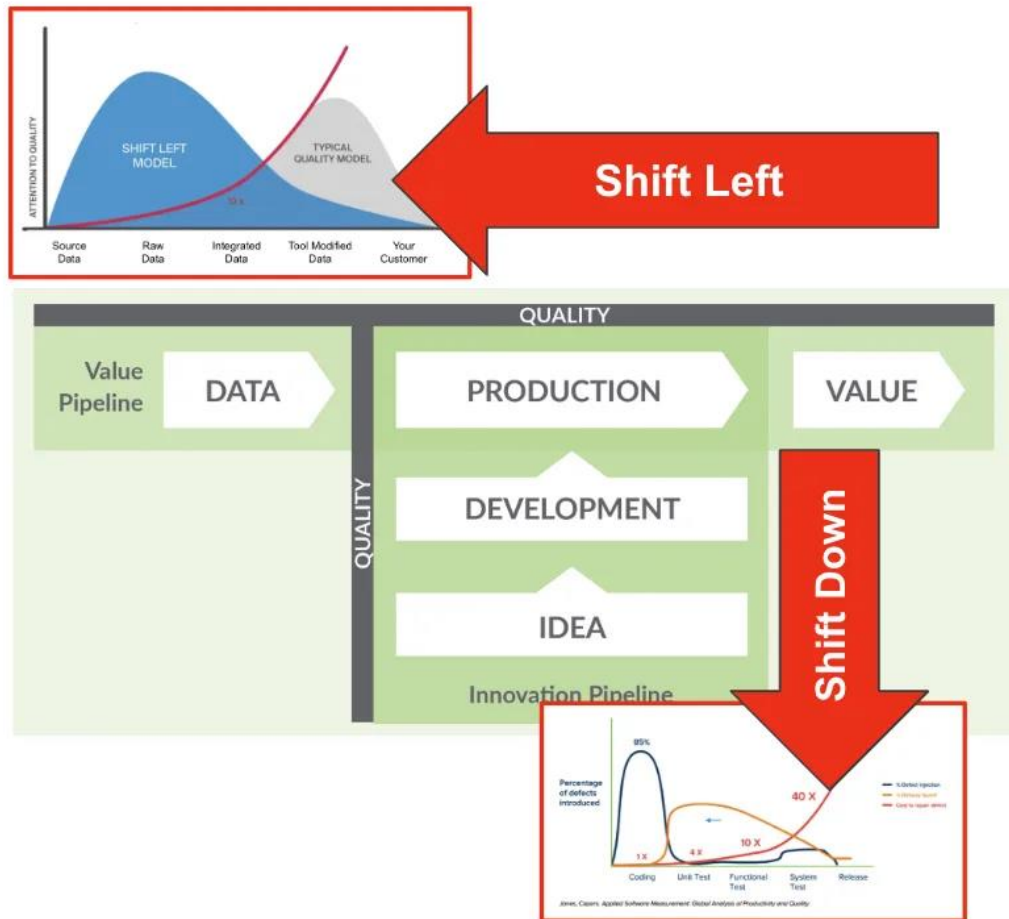
Implementation Strategies for Comprehensive Test Coverage

Implementing effective shift-left and shift-down testing requires more than just adding a few validation checks to existing pipelines. It demands a systematic approach to [test coverage](#) that addresses multiple dimensions of data quality and system reliability.

For shift left implementation, teams should begin by mapping their entire data pipeline and identifying critical quality checkpoints. At each checkpoint, implement tests. These tests should be automated and integrated into the pipeline orchestration, with the capability to stop processing if critical issues are found.

Input validation serves as the first line of defense, encompassing schema validation to ensure data conforms to expected formats, referential integrity checks to verify foreign key relationships, and business rule validation to confirm data values fall within acceptable ranges. For instance, a retail analytics pipeline might check that product prices are positive, order dates are not in the future, and customer IDs match existing records.

Validation of transformations occurs after each processing step. This involves row count reconciliation to prevent data loss, distribution checks to detect anomalies, and deterministic testing to ensure known inputs produce expected outputs. A financial data pipeline might verify that debits and credits balance, currency conversions are accurate, and moving averages align with expected results.



Output validation acts as the final quality gate before data reaches users. It includes consistency checks across metrics, monitoring of thresholds for key performance indicators, and anomaly detection for unusual patterns that indicate errors. For example, a marketing analytics system might ensure segment counts sum correctly, conversion rates stay within historical ranges, and no impossible values appear in calculated data.

For shift-down deployment, teams must set up development and testing environments that accurately replicate production systems, including the same tools, data volumes, performance characteristics, and security measures. Without realistic environments, issues that only appear under production conditions might go unnoticed.

The data strategy is especially important for shift-down testing. Production data can't be used directly due to privacy and security concerns, but synthetic data may lack real-world complexity. Effective teams employ data masking and sampling that preserve statistical properties and edge cases while safeguarding sensitive information. They maintain libraries of test datasets that represent various scenarios, including normal operations, peak loads, and known problematic patterns.

Cultural and Organizational Considerations

Technical solutions alone cannot solve the data quality challenge. The successful implementation of shift-left and shift-down testing requires cultural and organizational changes that prioritize quality as a shared responsibility, rather than an afterthought. Leadership must recognize that testing is not overhead but rather an investment in operational efficiency and business trust. This entails allocating time and resources for test development, maintaining the testing infrastructure, and continually improving quality processes. It means celebrating both prevented issues and resolved ones, recognizing that the absence of production failures represents successful quality engineering rather than luck.

Teams need to adopt a quality-first mindset where writing tests becomes as natural as writing transformation code.

This cultural shift often requires education and training, as many data professionals lack exposure to systematic testing practices. Organizations should invest in training programs that teach testing methodologies, provide hands-on experience with testing tools, and share best practices across teams to enhance overall testing effectiveness.

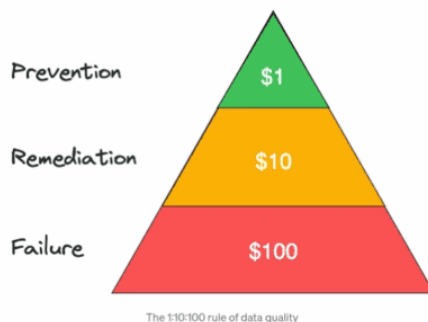
The organizational structure should support quality initiatives by clearly assigning ownership and accountability. While everyone shares responsibility for quality, dedicated roles such as data quality engineers or analytics reliability engineers can provide specialized expertise and drive systematic improvements. These specialists work across teams to establish standards, develop reusable testing frameworks, and promote best practices.

Metrics and incentives should align with quality goals. Rather than measuring success solely by feature delivery or project completion, organizations should track metrics like test coverage percentages, mean time to detection for issues, and the ratio of issues caught in development versus production. These metrics provide visibility into quality improvements and help justify continued investment in testing infrastructure.

Tools and Technologies for Shifting Down and Left

The evolution of data engineering tools has brought sophisticated testing capabilities within reach of teams at all scales. Modern orchestration platforms, such as Dagster and Apache Airflow, support integrated testing workflows where quality checks become first-class citizens in pipeline definitions. Transformation tools like dbt include built-in testing frameworks that make it easy to define and execute data quality tests alongside transformation logic.

Lack of Test Coverage Means Cost



Every Problem You Find Earlier is Cheaper!

- Tests are the gift you give to you future self
- The 1:10:100 rule applies
- Shift left and shift down testing
- Make sure every table is covered
- Make sure every tool is covered

Tools like Open Source DataOps Data Quality Testgen automate the creation of tests and make test coverage easy

Specialized data quality platforms have emerged to address the unique challenges of testing data pipelines. Open-source tools like [DataKitchen's DataOps Data Quality TestGen](#) automate the creation of data quality tests, reducing the manual effort required to achieve comprehensive coverage. These tools can profile existing data to suggest relevant tests, automatically generate test code, and integrate with existing orchestration and monitoring systems.

Commercial data observability platforms offer additional capabilities for teams that are ready to invest in enterprise-grade solutions. These platforms offer features such as automated anomaly detection, data lineage tracking for impact analysis, and sophisticated alerting systems that route issues to the relevant teams. While powerful, these tools work best when combined with systematic shift-left and shift-down practices rather than serving as a replacement for proper testing.

The key to successful tool adoption lies in selecting technologies that integrate smoothly with existing workflows rather than requiring wholesale process changes. Teams should start with tools that provide immediate value with minimal disruption, then gradually expand their testing toolkit as practices mature. A pragmatic approach might begin with

simple SQL-based tests in existing transformation tools, then add specialized data quality tools as test coverage grows, and finally implement comprehensive observability platforms once testing practices are well-established.

Implementing shift-left and shift-down testing is not a one-time project, but an ongoing journey of continuous improvement. Organizations require mechanisms to assess the effectiveness of their testing practices and pinpoint areas for improvement.

Regular retrospectives and post-incident reviews offer opportunities to identify testing gaps and areas for improvement. When production issues occur despite testing efforts, teams should analyze why existing tests failed to catch the problem and implement new tests to prevent recurrence. This creates a feedback loop where each incident strengthens the testing framework, gradually building comprehensive coverage through iterative improvement.

The Path Forward

The shift from reactive firefighting to proactive quality engineering marks a fundamental change in how organizations handle data and analytics systems. By shifting left to identify data issues early in the pipeline and shifting down to stop code defects from reaching production, teams can significantly reduce the cost and impact of quality problems while boosting the reliability and trustworthiness of their analytics.

The journey toward comprehensive test coverage requires technical investment, cultural transformation ([like Andons](#)), and organizational commitment. However, the alternative—continuing to operate in a mode where quality issues regularly disrupt business operations and weaken stakeholder trust—becomes increasingly unsustainable in a world where data drives critical decisions.

Organizations that successfully implement shift-left and shift-down testing will gain a significant competitive advantage. They will spend less time fixing problems in production and more time focusing on delivering value. They will rely on trustworthy data for making decisions, rather than relying on analytics that might be incorrect. They will build systems that scale reliably rather than becoming more fragile as complexity grows.

The tools, techniques, and knowledge needed to implement comprehensive testing are available today. What remains is for data engineering teams to embrace these practices, tailor them to their specific contexts, and commit to the ongoing effort necessary to maintain high-quality data systems. Although the investment may seem hefty, the costs of operating without proper testing grow higher with each production incident, each wrong business decision, and each loss of stakeholder trust.

The manufacturing industry learned these lessons decades ago, and software development has spent the last twenty years catching up. The time has come for data engineering to adopt the same quality-first mindset, implementing testing practices that will turn data pipelines from fragile hope chains into robust systems worthy of the critical business decisions they support. The way forward is clear: shift left, shift down, and embed quality into every part of our data and analytics systems.

Author: Chris Bergh | Published: Sep 30, 2025

Original article: <https://datakitchen.io/flip-the-script-on-data-quality-shift-left-shift-down-and-take-control/>

Scaling Data Reliability: The Definitive Guide to Test Coverage for Data Engineers

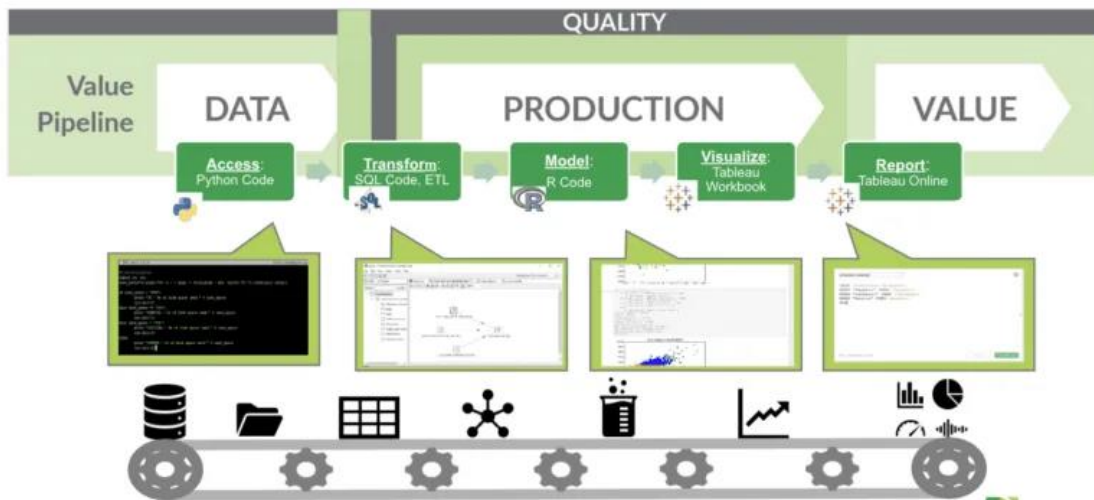
BlogCase StudiesWhite PapersWebinars / PodcastsDocumentation



The parallels between software development and data analytics have never been more apparent. Just as software teams would never dream of deploying code that has only been partially tested, data engineering teams must adopt comprehensive testing strategies to ensure the reliability, accuracy, and trustworthiness of their data products. This comprehensive guide explores the fundamental principles of test coverage in data and analytics systems, providing practical frameworks and strategies for implementing robust testing practices.

Not Just Software, But You're Also Running Data Manufacturing

It's not just software development that parallels data analytics, but manufacturing production. Data analytics processes fundamentally resemble manufacturing operations, where raw materials transform to produce refined outputs. In this context, data serves as the raw material, while the production outputs include refined datasets, visualizations, models, and reports. The analytic pipeline encompasses multiple stages, including data access through Python code, transformation via SQL and ETL processes, modeling using R or Python, visualization with tools like Tableau, and final reporting via various business intelligence platforms.

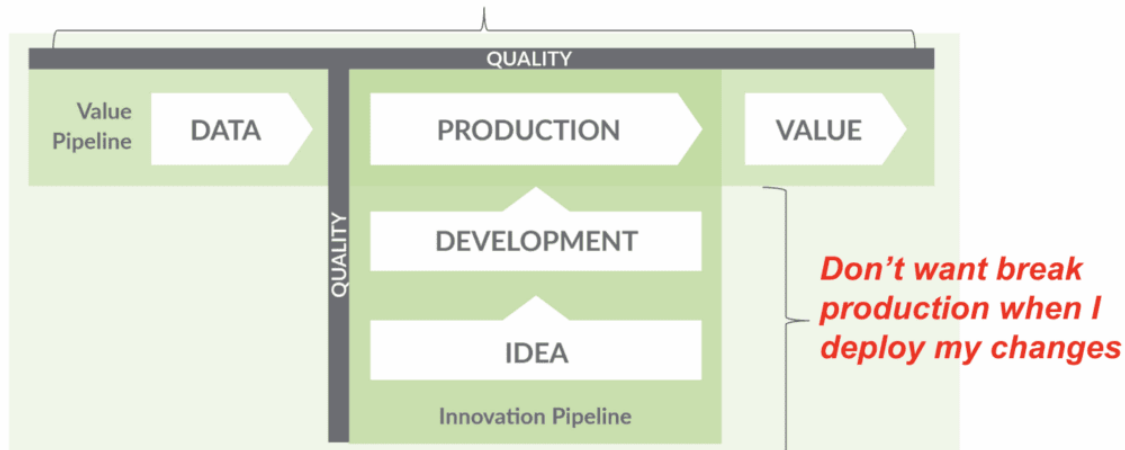


This manufacturing perspective highlights a critical truth: just as quality control is essential in manufacturing to prevent defective products from reaching customers, comprehensive testing is vital in data analytics to prevent erroneous insights from reaching business stakeholders. The consequences of inadequate testing can be severe, ranging from misguided business decisions to damaged customer relationships and regulatory compliance issues.

The Dual Challenge of Production and Development Testing

Test coverage in data and analytics operates across two distinct but interconnected dimensions: production testing and development testing. Production test coverage ensures that data quality remains high and error rates remain low throughout the value pipeline during live operations. Development test coverage provides confidence that changes made during the innovation pipeline will not break existing functionality when promoted to production.

Don't want to learn about problems from my customers



This dual nature creates an interesting dynamic where many tests serve double duty, functioning both as production monitors and development validators. Production testing focuses on maintaining data quality in live systems, while development testing ensures that new code changes do not introduce regressions or break existing functionality.

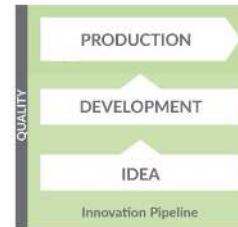
Shifting Left and Shifting Down

Two critical concepts underpin effective test coverage strategies: shifting left and shifting down. Shifting left involves executing data quality tests earlier in the data pipeline lifecycle, ensuring that issues are identified and resolved before they propagate downstream. This approach dramatically reduces the cost and complexity of addressing data quality issues.

'Shift Left' executing data quality tests earlier in the data pipeline lifecycle—so issues are identified and resolved before they propagate downstream.



'Shift Down' use integration and end-to-end testing methodologies, validating not just isolated components but also the interactions and cumulative impacts within the entire data ecosystem.



DataKitchen

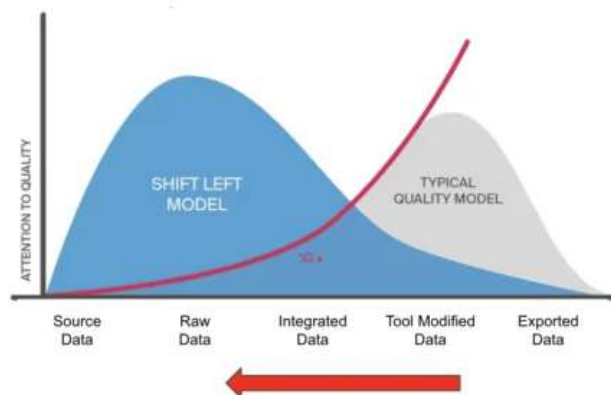
Shifting down emphasizes the use of integration and end-to-end testing methodologies, validating not just isolated components but also the interactions and cumulative impacts within the entire data ecosystem. This comprehensive approach ensures that the system functions correctly as a whole, not just in individual components.

The financial implications of these strategies are significant. Following the established 1:10:100 rule, the cost of preventing poor data quality at the source is approximately \$1 per record. In contrast, remediation after creation costs \$10 per record, and the cost of failure reaches \$100 per record. This exponential cost increase makes early detection and prevention not just a technical best practice but a business imperative.

It is 10x cheaper to find a problem in data at the source or raw layer.

Apply the 1:10:100 rule*:

- Cost of preventing poor data quality at source is \$1 per record
- Cost of remediation after it is created is \$10 per record
- Cost of failure (i.e. doing nothing) is \$100 per record



*developed by George Labovitz and Yu Sang Chang back in 1992

DataKitchen

Test coverage in data systems refers to the extent to which automated quality checks cover data itself, data pipelines, transformations, and outputs. Higher coverage levels correlate directly with increased confidence in data accuracy, reliability, and trustworthiness. Many organizations mistakenly assume that occasional spot checks or ad-hoc validations provide sufficient coverage, but this approach often leads to incomplete visibility and unexpected data quality issues.

The scope of testing in data systems extends beyond simple data validation to encompass the entire ecosystem of tools, processes, and outputs. This includes checking data inputs for quality issues, validating the correctness of business logic, ensuring output consistency, and monitoring the performance and reliability of every tool in the data pipeline.

Production Testing: “Before the Fact” – Testing is built into the production process to catch issues early. “After the Fact” – Testing occurs periodically. “Avoid the Fact” – No automated testing in production. Failures reach consumers. **Development Testing:** “Dynamic Testing” – Executes validations against real and code. “Static Analysis Testing” – Checks code, SQL, etc., without running real data. “Hope-Based Testing” – No formal tests. Reliance on intuition or trial-and-error.

Production Testing Strategies and Implementation

Production testing can be categorized into three distinct approaches, each with specific benefits and limitations.

“Before the fact” production testing integrates quality checks directly into the production process, stopping pipeline execution when issues are detected. This approach enables teams to identify problems before customers encounter them, but it requires the instrumentation of pipelines or the use of test-aware orchestration tools.

“After the fact” production testing involves continuous monitoring and polling of databases and data ingestion processes, using static data lineage to detect impacts based on test failures. While this approach does not require pipeline instrumentation, it allows problems to enter production before they are detected, necessitating scrambling to fix issues after they occur.

The third approach, “avoid the fact” production testing, relies on manual inspection and customer feedback to identify issues. This approach offers no proactive benefits and results in customers discovering and reporting problems, making it unsuitable for professional data operations.

Development Testing Methodologies

Development testing encompasses three primary methodologies, each serving different purposes in the testing strategy. Dynamic testing catches real-world data failures at runtime through integration tests that ensure multiple data processing steps interact correctly, and end-to-end tests that simulate complete data pipelines from ingestion through final analytics output.

Static analysis testing provides fast feedback by identifying problems before code commits, without requiring real data execution. This includes SQL linting tools like SQLFluff, unit testing frameworks, and data hygiene testing capabilities. These tools enforce coding standards, detect style issues, and validate code quality before deployment.

The third methodology, hope-based testing, relies on manual inspection and intuition, rather than formal automation. This approach provides no systematic validation and should be avoided in a professional data engineering environment.

Where **Should** Test Coverage Effort Focus?

		Production Testing		
Dev. Testing		‘Avoid the Fact’	‘After the Fact’	‘Before the Fact’
	Hope	0%		
	Static Analysis		20%	
	Dynamic Testing			80%

Where Data Teams **Actually** Focus

		Production Testing		
Dev. Testing		‘Avoid the Fact’	‘After the Fact’	‘Before the Fact’
	Hope	80%		
	Static Analysis		20%	
	Dynamic Testing			0%

Effective test coverage measurement requires a systematic application across all database levels and zones. Every table should have tests, every column in every table should have tests, and every significant business metric should have tests. Additionally, every tool that processes data should be monitored for errors and timing issues.

The minimum testing requirements include at least two tests per table, two tests per column, and one custom test per business metric. For tools and workflows, minimum requirements include one error log check per tool, one status check per job, and one timing check per job. These baseline requirements ensure comprehensive coverage while remaining manageable for implementation.



This Medallion Architecture Has **Three Database Levels** (L1, L2, L3), **Three Tools** (Streamsets, Airflow, dbt), And **Five Jobs/Workflows** to Monitor

- L1 - 100 tables + 10 Columns: Minimum 200 Table Tests and 2000 Column Tests
- L2 - 100 tables + 10 Columns: Minimum 200 Table Tests and 2000 Column Tests
- L3 - 10 tables + 30 Columns: Minimum 20 Table Tests and 600 Column Tests, Plus a handful of Domain Specific Business Metric Tests
- 3 Tools: 3 Checks For Errors
- 6 Jobs: 12 Checks For Problems



Modern test coverage strategies leverage automated consistency testing, which utilizes data profiling and semantic data type analysis to generate targeted tests. This approach automatically infers rules and documentation while generating parameterized tests that function like sensors in a comprehensive monitoring system.

The automated approach quickly and easily covers every table and column, automatically catching approximately 80% of errors through numerous small, fast tests that would be too numerous and tedious to code manually. This broad coverage enables monitoring across multiple data levels and provides wide-net detection of inconsistencies, shifts, and drift compared to previous data states.

Coverage: Automated Consistency Testing

Comprehensive test coverage encompasses multiple categories of validation, each addressing specific aspects of data quality. Freshness consistency tests ensure that new data arrives as expected, including recency tests that verify the latest dates fall within acceptable ranges, and date count tests that confirm the number of distinct dates has not dropped unexpectedly.

Volume consistency tests validate that sufficient new data is being received, including row count tests that ensure record counts have not decreased, and row range tests that verify counts fall within expected percentage ranges. These tests can be configured for different periods, from daily to monthly validation cycles.

Schema consistency tests ensure that the data structure remains consistent, including value match tests that verify all values align with the baseline data. These pattern match tests confirm alphanumeric patterns remain consistent, and format validation tests for specific data types, such as email addresses and geographic codes.

Data drift consistency tests identify statistically significant changes in data characteristics, utilizing algorithms such as Cohen's d for mean value differences, Cohen's h for ratio shifts, and various outlier detection methods. These tests provide early warning of gradual changes that might indicate upstream issues or evolving data sources.

Coverage: Custom and Domain-Specific Testing

While automated consistency testing provides broad coverage, custom domain-specific testing addresses the remaining 20% of validation requirements that are unique to specific business contexts. These tests encode business rules and industry-specific validations that automated systems cannot infer from data patterns alone.

Examples include verifying that medical practices do not exceed the number of doctors, ensuring that no shipments occur on specific days, or confirming that sales increases fall within expected ranges. These tests can be developed by data engineers, analysts, or data stewards, with templates and reuse patterns driving scalability.

Advanced testing strategies include referential tests, which validate relationships between multiple tables, and aggregate tests, which ensure data consistency across different groupings and time periods. These multi-table validations utilize sophisticated algorithms, such as the Jensen-Shannon divergence, to compare distributions across datasets that may not match exactly but should maintain similar characteristics.

Aggregate balance testing ensures that data totals remain consistent across different aggregation levels and periods, providing validation that transformations and calculations maintain mathematical consistency throughout the pipeline.

Writing Test Manually Does Not Scale

Writing data quality tests manually presents significant workload challenges, as demonstrated by the stark contrast between automated and manual approaches. The example shows that while TestGen can generate 2,500 comprehensive tests across 20 tables and 1,000 columns in just two simple steps (profile and generate tests), achieving the same coverage manually would require a trained Data Engineer to invest 7.2 months of dedicated work time. This translates to over 1,250 hours of continuous effort, working 8 hours per day for 156 days straight without accounting for meetings, breaks, or vacations.

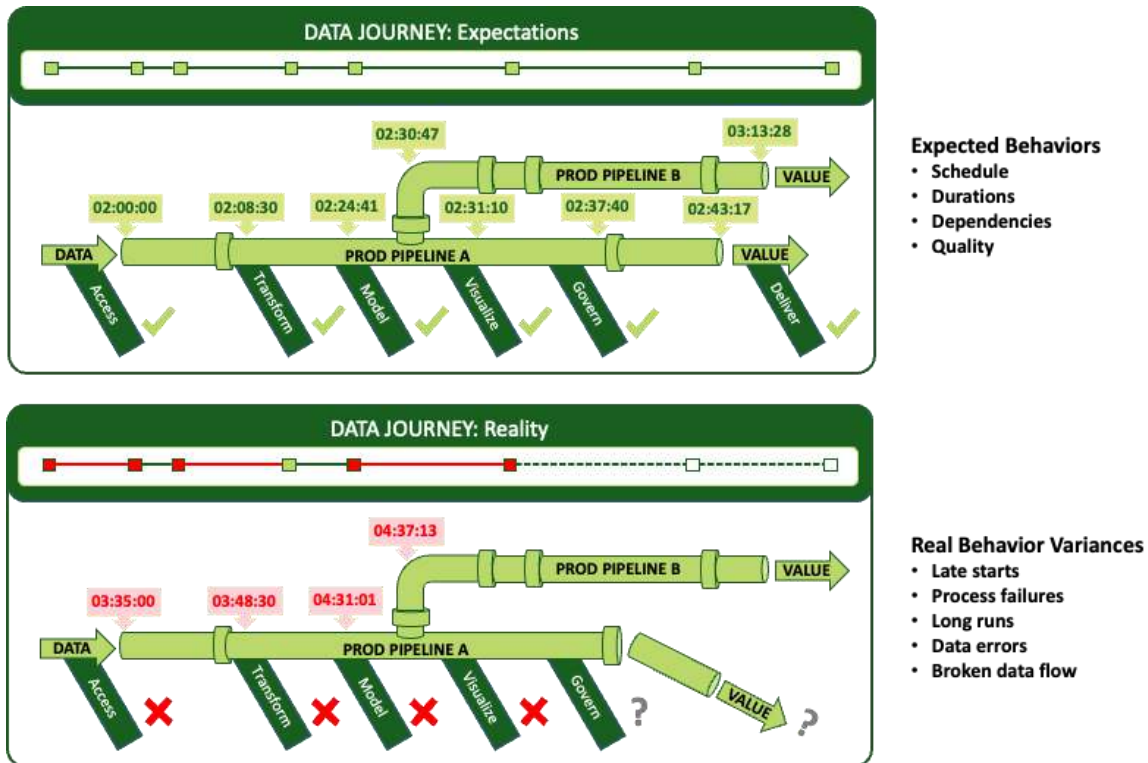
With TestGen, a junior operator can generate 2,500 tests with **two steps** (profile, generate tests)

It would take a trained Data Engineer **7.2 months** to achieve the same results – with no time for meetings, breaks, or vacations

Number of Tests	2,500
Number of Tables	20
Number of Columns	1,000
Minutes per Test	30
Hours	1,250
Days	156 8 hours/day
Weeks	31 5 days/week
Months	7.2 4.35 weeks/month
TestGen	2 steps

As data systems grow in complexity and scale, this manual approach becomes increasingly unsustainable – the time investment grows linearly with the number of tables and columns. At the same time, automated tools can handle exponentially larger datasets with minimal additional effort. The 30-minute manual effort required per test quickly compounds into months of work, making it impossible for organizations to maintain comprehensive data quality coverage across their entire data ecosystem without automation.

Comprehensive test coverage extends beyond data validation to include monitoring of the entire toolchain that processes data. Modern data environments typically include 50 or more tools across various categories, each requiring monitoring for errors, performance issues, and timing problems.



Development Environment Integration

Effective test coverage requires seamless integration between development and production environments. Development teams require access to representative test data and accurate replicas of production environments to facilitate meaningful testing and ensure a smooth deployment.

The development testing process should include data hygiene qualification screens for new datasets, regression testing for code changes, and comprehensive validation that all existing functionality remains intact after modifications. This controlled process ensures that only validated changes are deployed to production environments.

Quality Improvement and Reporting On Test Coverage

Sustainable test coverage requires supporting organizational processes, including data quality dashboards that provide visibility into system health, process reports that enable targeted improvements, and quality circles that foster a culture of continuous improvement.

[Quality circles](#) represent particularly important cultural elements, emphasizing periodic meetings and retrospectives with a “no shame, no blame” approach that focuses on learning from errors and identifying opportunities for improvement. These practices create organizational commitment to data quality that extends beyond technical implementation.

1				
2	Data Catalog Columns			
3	Exported	Jul 14, 3:06 PM		
4	Table group	default		
5				
6	Schema	Table	Column	Active tests
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				
40				
41				
42				
43				
44				
45				
46				
47				
48				
49				
50				
51				
52				
53				
54				
55				
56				
57				
58				
59				
60				
61				
62				
63				
64				
65				
66				
67				
68				
69				
70				
71				
72				
73				
74				
75				
76				
77				
78				
79				
80				
81				
82				
83				
84				
85				
86				
87				
88				
89				
90				
91				
92				
93				
94				
95				
96				
97				
98				
99				
100				

Measuring and reporting test coverage on data tables requires systematic tracking of test counts per column and table as a fundamental baseline metric. This quantitative approach serves two critical purposes: first, it provides concrete evidence to customers and management about the thoroughness and extent of your data quality efforts, transforming abstract concepts of “data testing” into tangible, measurable results that stakeholders can evaluate and trust. Second, this granular visibility into test distribution helps identify coverage gaps where certain columns or tables may be under-tested, potentially leaving critical data vulnerabilities undetected. By maintaining counts of tests per column and table, data teams can demonstrate due diligence in their quality assurance processes, justify resource allocation for data quality initiatives, and strategically prioritize areas that need additional testing attention. This coverage reporting also enables trend analysis over time, showing whether test coverage is improving or declining as the data ecosystem evolves, ultimately supporting more informed decisions about data reliability and risk management.

Organizations face three primary design choices when implementing comprehensive test coverage, each with specific benefits and limitations. Data quality-focused implementations concentrate on testing data sources but may lack visibility into downstream tools and processes, potentially allowing incorrect results to reach customers.

Data ingestion-focused implementations poll databases and alert teams to potential errors, but may still allow data errors into production while generating irrelevant alerts with partial coverage. Comprehensive data engineering implementations integrate tests and monitors into all data production processing steps, stopping errors before they reach production while enabling the same setup for development testing.

If you're a Data Quality Person: Just Focus on Testing Your Data Sources
Create one of six types of data quality dashboards
Limitation: Downstream tools and processes are invisible
Limitation: Customer-Delivered Data Analytic Results May Be Incorrect
Tool: DataOps Data Quality TestGen

If you're a Data Ingestion Engineer, or have no control beyond your database: Just Poll Your Database and Ping people when there may be an error
Limitation: Data errors can still get into production
Limitation: irrelevant pings, partial coverage
Tool: DataOps Data Quality TestGen+ data lineage based notifications

If you're a Data Engineering Team and you control your tools, go for complete coverage: Put tests and monitors as part of the data production processing steps (that is, the Data Journey)
Benefits: stop errors before they get into production, use the same setup for development and testing
DataKitchen Open Source Software uniquely provides this functionality

Comprehensive test coverage in data and analytics represents a fundamental requirement for professional data engineering practices. The 1:10:100 cost rule makes early detection and prevention a business imperative, while the complexity and scale of modern data environments make manual testing approaches unsustainable.

Organizations must adopt automated testing tools and frameworks that provide comprehensive coverage while maintaining rigorous validation capabilities. The combination of automated consistency testing for broad coverage and custom domain-specific testing for business rule validation provides a balanced approach that addresses both technical and business requirements.



Every Problem You Find Earlier is Cheaper!

- Tests are the gift you give to you future self
- The 1:10:100 rule applies
- Shift left and shift down
- Make sure every table is covered
- Tools like Open Source DataOps Data Quality Testgen automate the creation of tests and make test coverage easy

The future of data engineering depends on treating data quality with the same rigor and systematic approach that software engineering applies to code quality. By implementing comprehensive test coverage strategies, organizations can develop reliable and trustworthy data products that support informed business decision-making and maintain a competitive edge in data-driven markets.

Success requires not only technical implementation but also an organizational commitment to a culture of quality, continuous improvement processes, and systematic approaches to error prevention and detection. The investment in comprehensive test coverage pays dividends through reduced operational costs, improved customer satisfaction, and enhanced business confidence in data-driven insights and decisions.

Author: Chris Bergh | Published: Jul 8, 2025

Original article: <https://datakitchen.io/scaling-data-reliability-the-definitive-guide-to-test-coverage/>

The Syntax, Semantics, and Pragmatics Gap in Data Quality Validation Testing

What is the full range of data quality validation tests for data at rest and data in use? Linguistics provides an organizing principle: syntax, semantics, and pragmatics



*Data Teams often have too many things on their 'to-do' list. **Customers are asking for new data, people need questions answered, and the tech stack is barely running – data engineers don't have time to create tests.** They have a backlog full of new customer features or data requests, and they go to work every day knowing that they won't and can't meet customer expectations.*

To understand the time crunch that data engineers face and how it prevents them from testing their data sufficiently, we can think about three categories of data tests, using linguistics as an analogy: data syntax vs. semantics vs. pragmatics. In linguistics, Syntax is the study of sentence structure and grammar rules. While people can do what they want with language (and many often do), syntax helps ordinary language users understand how to organize words to make the most sense.

On the other hand, semantics is the study of the meaning of sentences. The sentence "Colorless green ideas sleep furiously" makes syntactic sense but is meaningless. Pragmatics takes semantics one step further because it studies the meaning of sentences within a particular context.

Syntax-Based Profiling and Testing: *By profiling the columns of data in a table, you can look at values in a column to understand and craft rules about what is allowed for a column. For instance, if a column is filled with US zip codes, a row should not have the word "bacon" in it. Data engineers need a quick way to profile data quickly and automatically cast a wide net to catch data anomalies. It's analogous to setting up a burglar alarm in your home by deploying sensors at all possible entrances to catch a burglar who may only try one window. Automatically creating tests from profile data allows teams to maintain maximum sensitivity to real problems while minimizing false positives that are not worth the follow-up.*

Semantics-Based Business Rule Testing. *What is a meaningful test for your business? Do you know as a data engineer? For example, you can compare current data to previous or expected values. These tests rely upon historical values as a reference to determine whether data values are reasonable (or within the range of reasonable). For example, a test can check the top fifty customers or suppliers. Did their values unexpectedly or unreasonably go up or down relative to historical values? What is the acceptable rate of change? 10%? 50%? Data engineers are only*

sometimes able to make these business judgments. They must thus rely on data stewards or business customers to 'fill in the blank' on various data testing rules.

Pragmatics-Based Custom Testing. Many companies have widely diverging business units. For example, a pharmaceutical company may be organized into Research and Development (R&D), Manufacturing, Marketing and Sales, Supply Chain and Logistics, Human Resources (HR), and Finance and Accounting. Each unit will have unique data sets with specific data quality test requirements. Drug discovery data is so different from manufacturing data that data test cases require unique domain knowledge or a specific, pragmatic business context based on each group's unique data and situation.

TestGen also offers 11 business rule data tests that, with minimal configuration, can be used for more customized tests. These tests allow users to customize testing protocols to fit specific business requirements with a "fill in the blank" model, offering a perfect blend of speed and robustness in data testing. These types of tests ensure your data not only meets general quality standards but also aligns with your unique business needs and rules. Data stewards, who may know more about the business than a data engineer, can quickly change a setting to adjust the parameters of a data test – without coding.

Every company is unique. Every company has complex data and tools within each team, product, and division. Monitoring and testing the data to ensure its reliability continually is crucial. It is crucial to build domain-specific data validation tests and, for example, the results of data models for accuracy and relevance, evaluate the effectiveness of data visualizations. Checking the result of a model, an API call, or data-in-use in a specific analysis tool is critical to ensure that data delivery mechanisms are operating optimally. DataOps Automation provides robust testing and evaluation processes **throughout the 'last mile'** of the Data Journey.

Author: Chris Bergh | Published: Jul 12, 2023

Original article: <https://datakitchen.io/the-syntax-semantics-and-pragmatics-gap-in-data-quality-validate-testing/>

Data Quality Testing: A Shared Resource for Modern Data Teams

Data quality is not a problem any single role can solve in isolation. The complexity and scale of modern data ecosystems necessitate a collaborative approach, where quality testing serves as a shared infrastructure across all data and analytics roles.



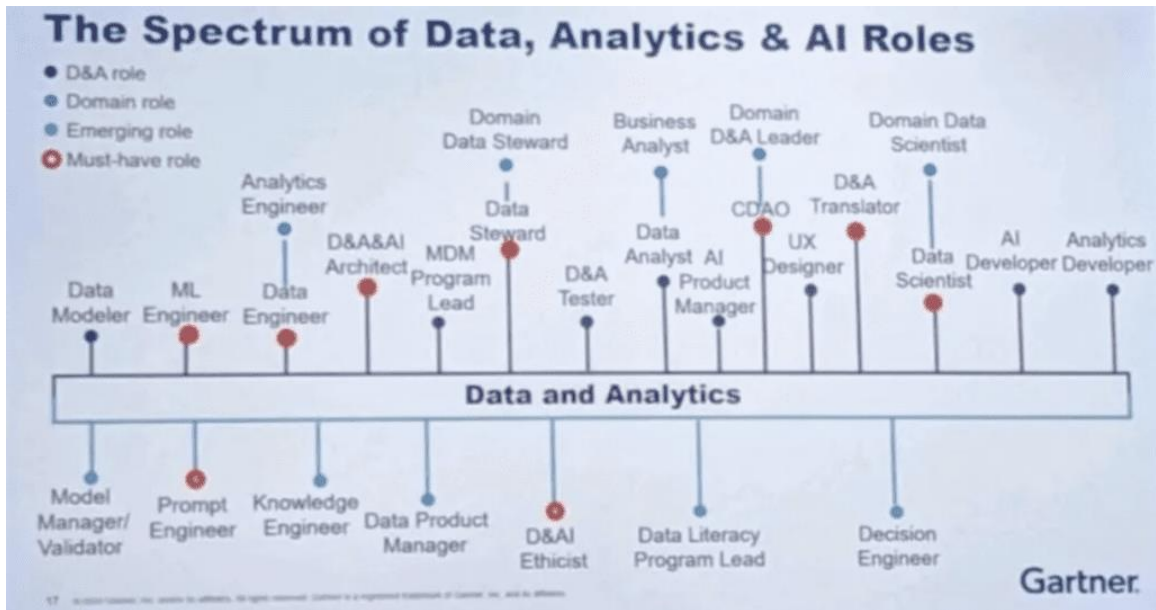
In today's AI-driven landscape, where data is king, every role in the modern data and analytics ecosystem shares one fundamental responsibility: ensuring that incorrect data never reaches business customers. Whether you're a Data Engineer building ETL pipelines, a Data Scientist developing predictive models, or a Data Steward ensuring compliance, we all want the same outcome: data that is trustworthy, accurate, and understandable.

The reality is stark: no one in your organization wants misleading data to reach business decision-makers. Yet ironically, data quality testing often remains isolated in the hands of a few engineers or specialists, disconnected from the broader collaborative framework that defines a high-functioning data team.

The Data Team Is Diverse—But Unified By the Need for Quality

A modern data team is a mosaic of specialized roles. According to [Gartner's breakdown of analytics and data roles](#), data teams now span far beyond traditional data engineering and business intelligence (BI) analysts. We have ingestion engineers, analytic engineers, stewards, governors, modelers, owners, scientists, product managers, compliance officers, and executives. This proliferation of roles is necessary—it reflects the growing complexity and business criticality of data. But it also introduces a problem.

Each role touches data differently. And each role encounters the impact of poor data quality in different ways.



Yet they are united by the same pain: quality issues are everyone’s problem, and solving them benefits everyone. Quality tests improve productivity, reduce stress, and restore the trust that’s often eroded by broken dashboards, missed alerts, or misaligned metrics.

So, how can a team with such varied responsibilities all participate in data quality without devolving into chaos or passing the buck?

While these roles have different day-to-day responsibilities and technical focuses, they are united by common pain points when data quality fails:

The solution isn’t for each role to develop isolated data quality processes. The answer: a centralized data quality testing engine—shared, visible, extensible, and integrated across your data lifecycle.

One Engine, Many Roles: How Different Roles Engage with Data Quality Testing

A centralized approach to data quality doesn’t mean central control—it means shared infrastructure. It provides everyone with a standard foundation, even as they utilize it in role-specific ways. Every data team member needs to interact with data quality testing, but in different ways depending on their responsibilities and expertise:

Data Ingestion Teams: Data ingestion specialists use quality tests to **identify errors in source data** before it propagates downstream. They configure tests to catch schema changes, missing data, and format inconsistencies at the earliest possible point in the data pipeline.

Data Engineering Teams: Data Engineers leverage quality testing to **find problems in data warehouse ETL jobs** both during development and in production environments. They write tests for data transformations, join operations, and aggregation logic to ensure their pipelines produce accurate results.

Data Quality Specialists: Dedicated Data Quality professionals use testing frameworks to **score data sources and influence source data improvement**. They establish quality metrics, set thresholds, and collaborate with upstream systems to identify and address the root causes of data issues.

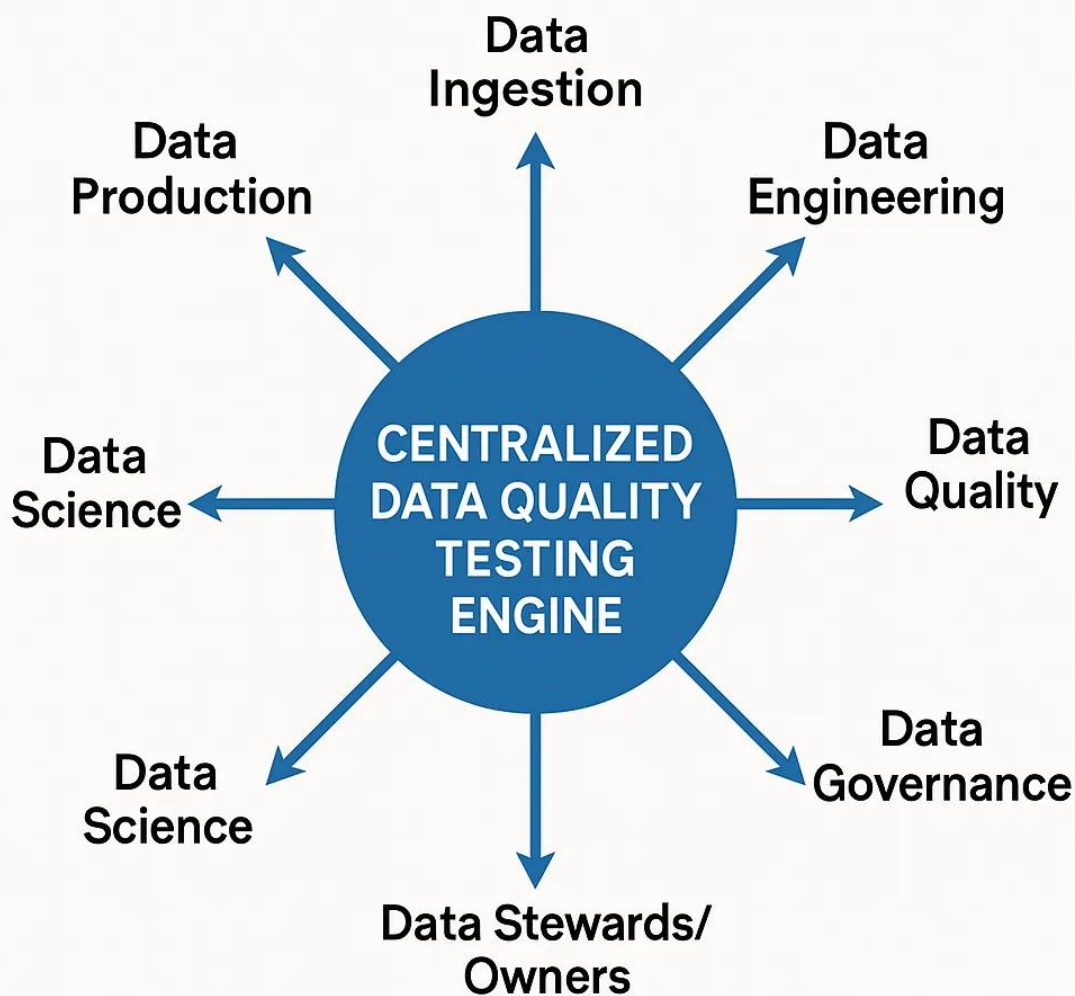
Data Governance Teams: Data Governance professionals employ quality testing as **a means to enhance data catalogs with high-quality metadata**. Test results provide objective measures of data fitness for use and help enforce data governance policies across the organization.

Data Stewards and Data Owners: Data Stewards and Owners need to **view and understand data quality test results** to fulfill their accountability for data accuracy. They use quality dashboards to monitor their domains and take corrective action when issues arise.

Data Science Teams: Data Scientists use quality testing as **a way to validate data for predictive models**. They run tests to check for data drift, feature completeness, and statistical properties that could impact model performance.

Data Production Teams: Production support teams use quality testing as **a monitoring tool for production data jobs**. They set up automated quality checks that alert them to issues before they impact business users.

Data Team Executives: Leadership uses quality testing metrics to **assess test coverage and organizational risk**. They track quality trends, coverage gaps, and the business impact of data quality initiatives.



A centralized data quality testing engine supports numerous use cases that different team members can leverage across the entire data lifecycle. Production support teams conduct continuous monitoring tests to detect problems as they occur and trigger alerts before business users experience any impact, while data ingestion specialists validate newly arrived data to ensure that source systems meet the expected standards and formats. Development teams integrate quality tests into their workflows as regression testing to prevent code changes from introducing new data issues, and data quality analysts generate objective quality scores that drive improvement initiatives in upstream

systems. Governance and compliance teams rely on standardized testing to demonstrate adherence to regulatory requirements and data policies. In contrast, operations teams execute comprehensive periodic health checks to identify gradual quality degradation that might otherwise go unnoticed until it becomes a critical business problem.

The Reality: Time and Resource Constraints

Despite the clear value of data quality testing, team members are often too busy with core responsibilities to write comprehensive tests. Data Engineers focus on building pipelines, Data Scientists on developing models, and Analysts on creating reports and dashboards.

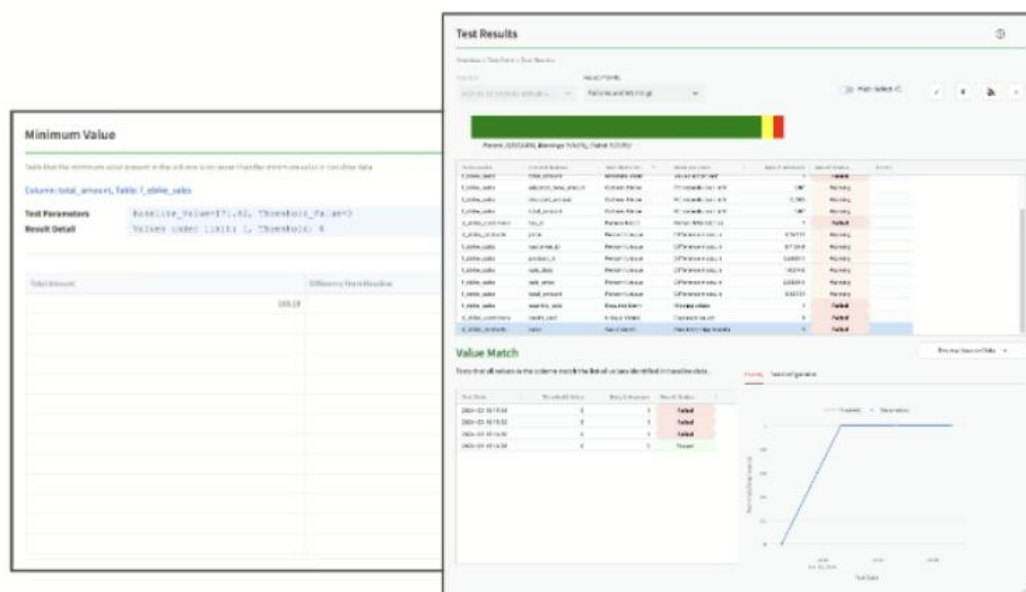
Even when team members recognize the importance of quality testing, they may feel disconnected from the business context needed to write meaningful tests. A Data Engineer might know how to test for null values, but may not understand the business rules that determine whether a customer record is valid.

This creates a gap between the need for comprehensive testing and the practical ability of busy data professionals to implement it.

Overcome That Reality: Automate the 70%, Focus on the 30% That Matters Most.

DataOps Data Quality TestGen addresses this challenge by providing a centralized testing engine designed for modern data teams. Rather than expecting every role to become a testing expert, TestGen automates the generation of approximately 70% of the tests most organizations need. TestGen functions as a complete DataOps tool supporting the entire quality testing lifecycle:

While automated generation addresses foundational needs, TestGen offers an intuitive user interface that enables team members with business knowledge to fine-tune tests without requiring code. Data Stewards can adjust acceptable ranges, Data Owners can modify validation rules based on domain expertise, and Analysts can configure business-specific logic.



Benefits Across the Organization

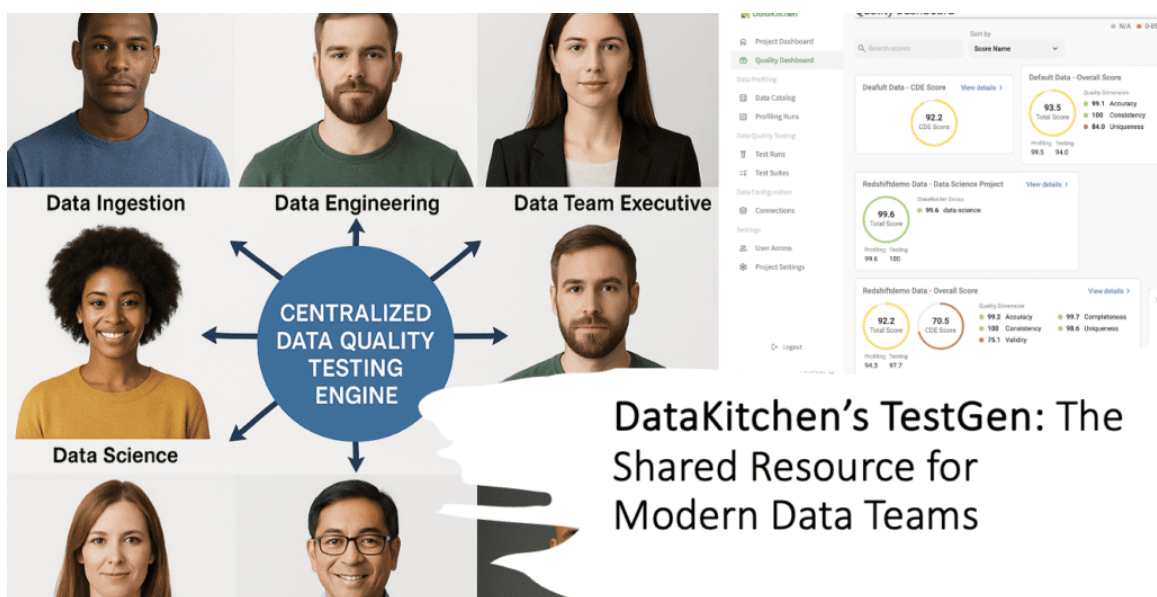
When implemented as a shared resource, data quality testing delivers transformative benefits across the entire organization. Improved productivity emerges as teams spend less time investigating issues and more time delivering business value, with automated test generation eliminating the overhead of writing basic tests from scratch. Data

professionals experience reduced stress as they gain confidence that quality issues will be caught systematically rather than discovered by frustrated business users in critical moments. Enhanced customer trust develops through consistent, proactive testing that builds stakeholder confidence in data accuracy and reliability. Better collaboration flourishes as a shared platform creates a common vocabulary and metrics for communicating about quality issues and improvements. Perhaps most importantly, centralized testing enables organizational learning by creating visibility into quality patterns that inform strategic process improvements and technology investments across the data ecosystem.

Conclusion: Data Quality Is a Team Sport—Let's Equip Everyone

Data quality is not a problem any single role can solve in isolation. The complexity and scale of modern data ecosystems necessitate a collaborative approach, where quality testing serves as a shared infrastructure across all data and analytics roles.

By implementing a centralized testing engine that automates foundational tests while providing intuitive customization capabilities, organizations ensure that every role has access to the necessary quality testing without requiring every professional to become a testing expert.



DataKitchen's TestGen: The Shared Resource for Modern Data Teams

The result is higher-quality data, more productive teams, and greater trust in the analytics that drive business decisions. In an era where data is increasingly central to competitive advantage, investing in shared data quality testing infrastructure is not just a technical necessity—it's a strategic imperative.

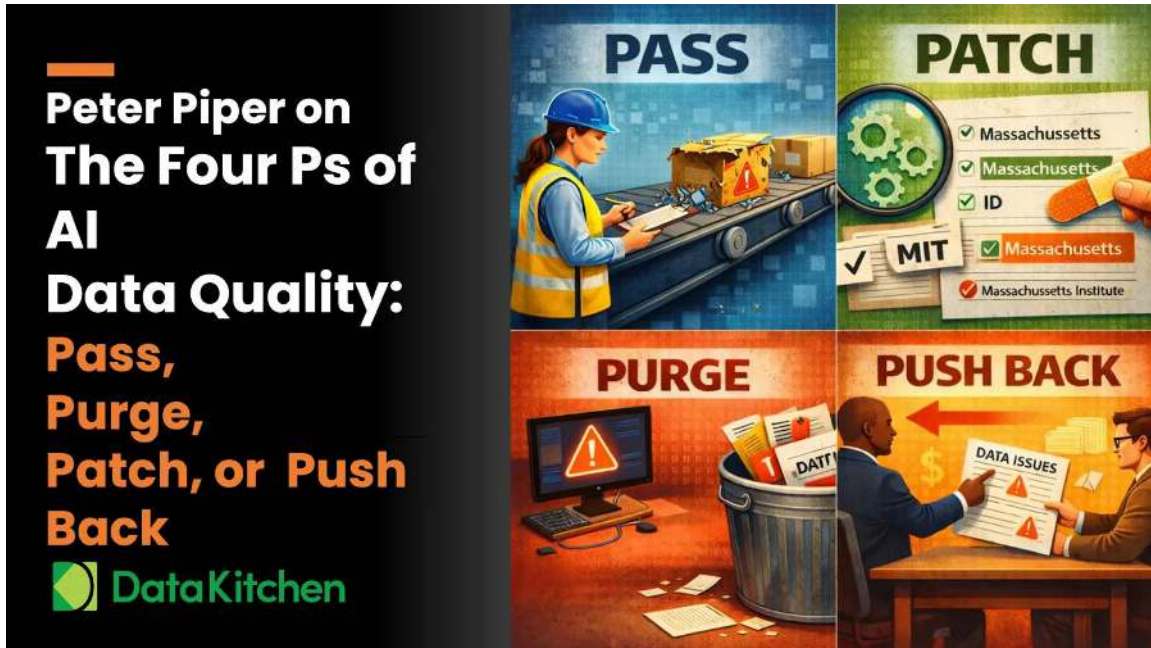
Data quality testing is everyone's job. Let's ensure that everyone has access to the shared infrastructure and tools necessary for success.

Author: Chris Bergh | Published: Jun 6, 2025

Original article: <https://datakitchen.io/data-quality-testing-a-shared-resource-for-modern-data-teams/>

Peter Piper on the Four Ps of AI Data Quality: Purge, Patch, Push Back, or Pass

How does a data team prevent poor data from poisoning AI when they have piles of raw and imperfect data?



How does a data team prevent poor data from poisoning AI when they have piles of raw and imperfect data?

Teams responsible for [data used to train AI models](#) (e.g., LLMs) face a persistent problem: piles of raw, imperfect data. Pressure builds to process quickly, publish promptly, and push data into pipelines. But passing problematic data into production-powered models can produce biased predictions, polluted patterns, and poor performance.

Before pipelines proceed, managers must pause and pick a path. In practice, there are four practical options for handling raw data: Pass, Purge, Patch, Push Back

1. Pass: The Path of Least Preparation

This path promises speed and simplicity. It is the least amount of work, but also the largest potential risk. Poorly prepared data propagates problems downstream, where models may:

2. Purge: Preventing Poisoned Patterns

Purging prevents polluted records from poisoning patterns learned by AI models. While purging reduces volume, it protects validity and preserves precision.

3. Patch: Precise, Programmatic Problem-Solving

4. Push Back: Partner Pressure for Proper Data

Pushing back promotes partnership, not punishment. It improves future feeds, reduces repeated patching, and produces more predictable pipelines.

Assess Data Quality with DataKitchen TestGen

Conclusion: Purposeful Preparation Produces Powerful Predictions

Passing poor data produces predictable problems. Purposeful preparation prevents polluted pipelines. By profiling proactively, purging problematic records, patching predictable problems, and pushing back on poor providers, teams gain control, confidence, and credibility.

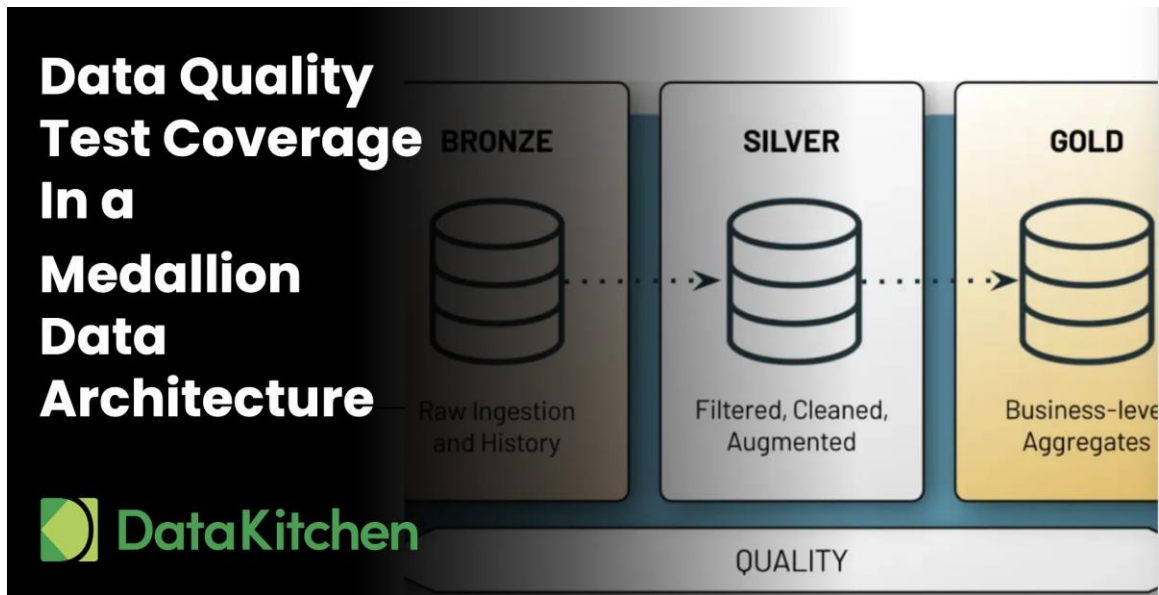
With precise profiling, principled processes, and practical platforms like TestGen, managers can protect pipelines, promote performance, and produce powerful, polished AI models — not by chance, but by plan.

Author: Gil Benghiat | Published: Dec 18, 2025

Original article: <https://datakitchen.io/peter-piper-on-the-four-ps-of-ai-data-quality-purge-patch-push-back-or-pass/>

Data Quality Test Coverage In a Medallion Data Architecture

For data engineering teams serious about delivering production-grade data products, implementing systematic test coverage across their Medallion architecture represents not only a technical improvement but a fundamental shift toward sustainable and trustworthy data operations.



Data quality test coverage has become one of the most critical challenges facing modern data engineering teams, particularly as organizations adopt the increasingly popular Medallion data architecture. While this multi-layered approach to data processing offers significant advantages in organizing and refining data, it also introduces complexity that demands rigorous testing strategies to ensure data integrity across all layers.

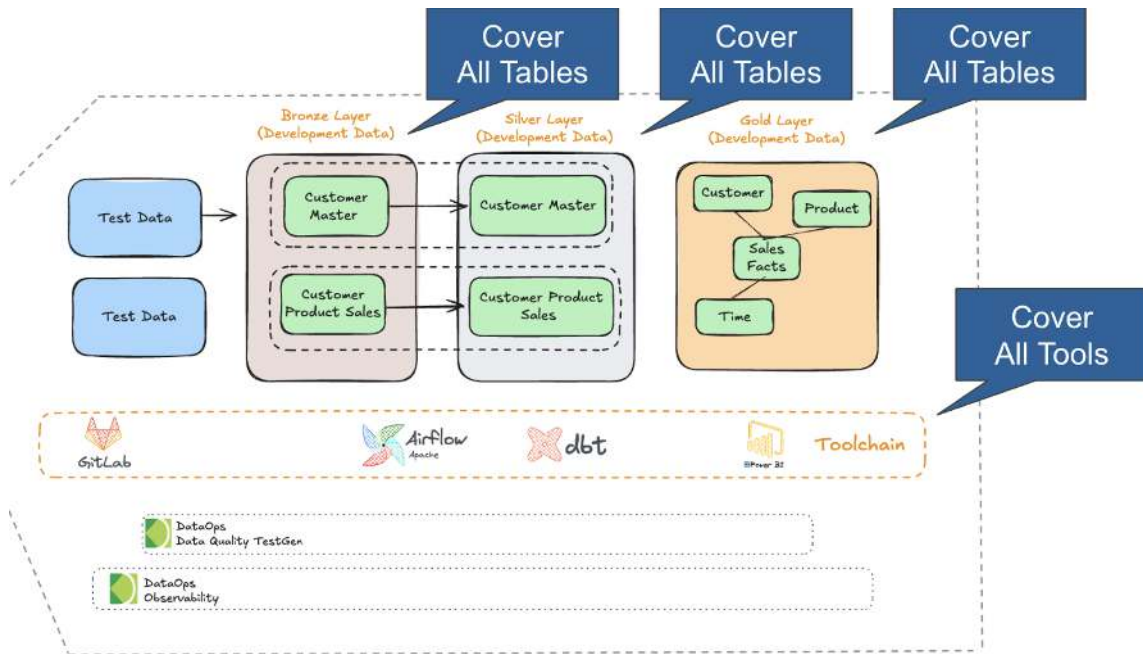
Understanding the Medallion Data Architecture

The Medallion architecture represents a structured approach to data processing, organizing data into distinct layers that serve specific purposes in the data transformation journey. This architecture has gained tremendous popularity in data lakehouse implementations, where organizations need to balance raw data storage with analytical readiness.

The Bronze layer serves as the initial landing zone for all incoming raw data, capturing it in its unprocessed, original form. This layer maintains the complete fidelity of source data, preserving every detail that might be needed for future analysis or compliance requirements. Data engineers often refer to this as the “source of truth” layer, where data remains unchanged from its original state.

Moving up the architecture, the Silver layer represents the processing stage where data undergoes “just enough” cleaning and transformation to provide a unified, enterprise-wide view of core business entities. This layer focuses on standardization, deduplication, and basic quality checks while maintaining the granular detail needed for various downstream applications. The Silver layer essentially creates a cleansed, normalized version of the Bronze data without heavy aggregation.

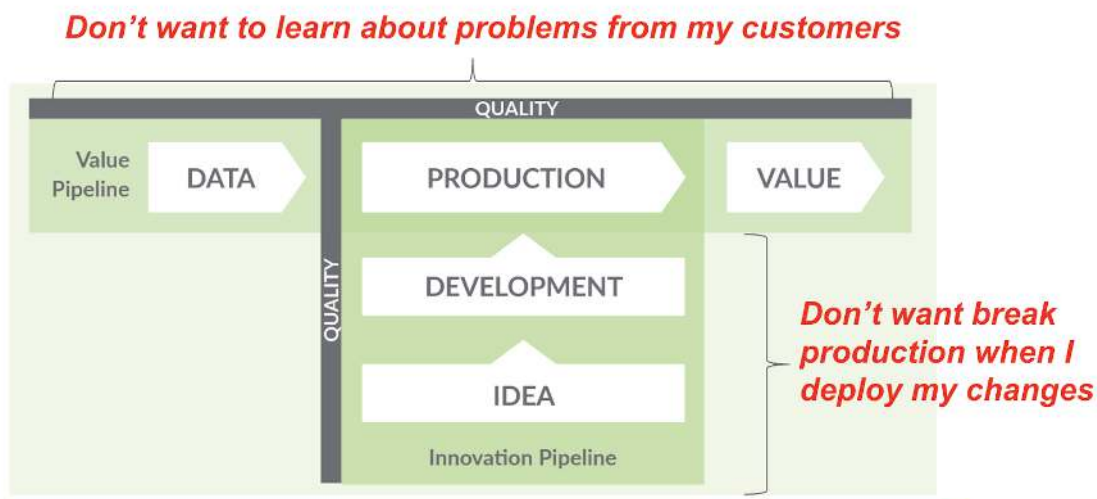
Finally, the Gold layer represents the pinnacle of the Medallion architecture, housing fully refined, aggregated, and analysis-ready data. Data in this layer is typically organized into project-specific schemas optimized for business intelligence and advanced analytics. The Gold layer delivers the final data products that business users interact with through dashboards, reports, and analytical applications.



Some organizations extend this pattern with a Platinum layer beyond Gold, while others prefer the simplicity of L1, L2, and L3 designations. Regardless of naming conventions, the fundamental challenge remains consistent across all implementations: data quality becomes increasingly complex as data moves through these layers, making comprehensive test coverage essential for maintaining trust and reliability.

The Critical Need for Test Coverage

Test coverage in data engineering shares fundamental principles with software testing but addresses unique challenges inherent to data systems. Just as software teams would never consider releasing partially tested code, data teams must adopt equally rigorous standards for their data products. The consequences of insufficient testing in data systems can be far-reaching, affecting business decisions, regulatory compliance, and organizational trust.



The primary motivation for comprehensive test coverage centers on two fundamental concerns that every data engineering team faces: preventing production breaks when deploying changes and avoiding the embarrassment of customers discovering data quality issues before internal teams do. These scenarios represent not just technical failures but organizational credibility challenges that can have lasting impacts on stakeholder confidence.

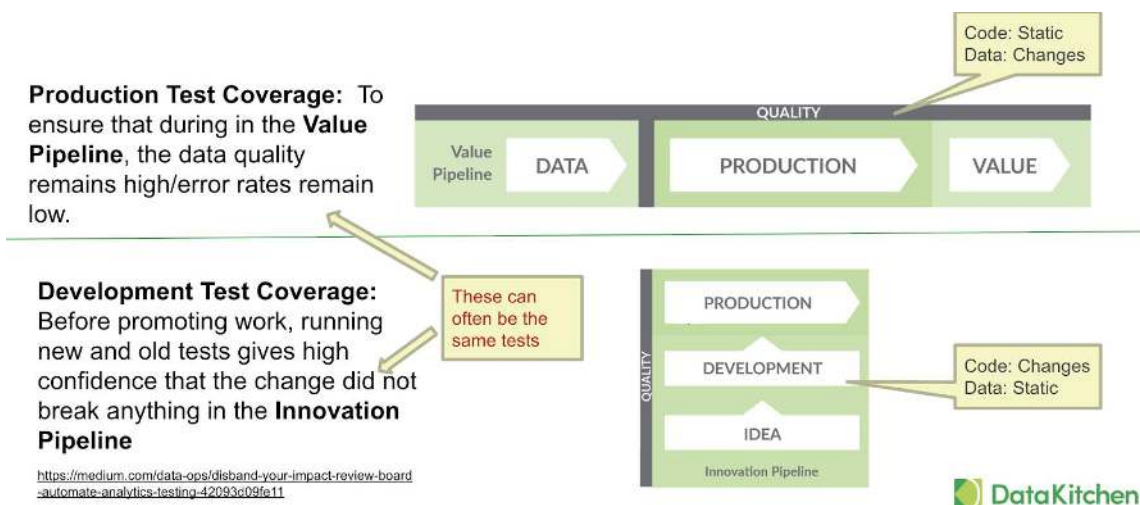
Data test coverage refers to the extent to which automated quality checks cover your data itself, data pipelines, transformations, and outputs. The higher your coverage, the more confident you can be that the data you produce is accurate, reliable, and trustworthy. Many organizations mistakenly assume that occasional spot checks or ad-hoc validations provide sufficient protection against data quality issues. Unfortunately, manual, partial testing often leads to incomplete visibility and unexpected problems that surface at the worst possible moments.

The analogy to software development proves particularly relevant here. Imagine releasing software that has only been partially tested—no development team would accept such risk. The same standards should apply to data systems, where incomplete testing can lead to faulty business decisions, compliance violations, and eroded trust in data-driven insights.

Strategic Placement of Test Coverage

Effective test coverage in a Medallion architecture requires understanding both where and when to implement testing strategies. The placement of tests follows two primary patterns: development test coverage and production test coverage, each serving distinct but complementary purposes in maintaining data quality.

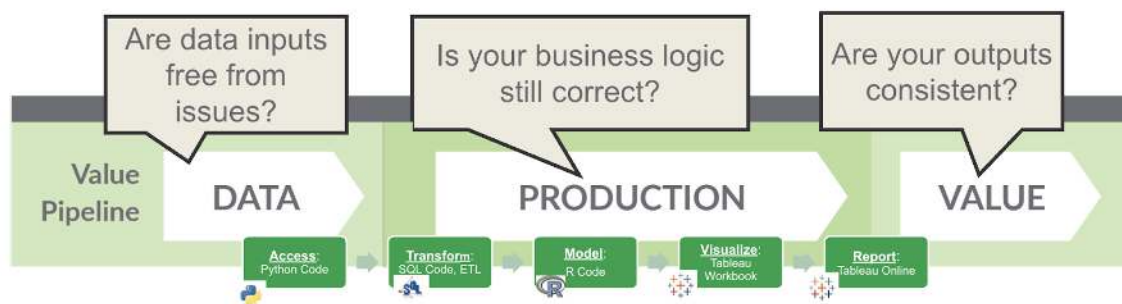
Development test coverage operates within what data engineers refer to as the Innovation Pipeline, where new features, transformations, and data sources are developed and validated before being promoted to production. During this phase, both new and existing tests run against changes to provide high confidence that modifications haven't broken existing functionality. This represents a critical checkpoint where code changes while data remains static, allowing teams to validate transformations against known datasets.



Production test coverage functions within the Value Pipeline, ensuring that data quality remains high and error rates stay low during active data processing. In this environment, code remains static while data constantly changes, requiring different testing approaches that focus on data consistency, freshness, and adherence to expected patterns. These production tests often mirror development tests but operate against live, flowing data rather than static test datasets.

The beauty of this dual approach lies in the fact that development and production tests can often be identical, providing consistency across environments while addressing the different dynamics of static versus flowing data. This consistency enables teams to build confidence in their testing strategies and ensures that validation logic remains synchronized across the entire data lifecycle.

Test Every Step And Every Tool in Your Value Pipeline



Comprehensive test coverage must extend beyond just data validation to encompass every step and tool in your value pipeline. Whether you're working with Python code for data access, SQL transformations in ETL processes, R code for modeling, Tableau workbooks for visualization, or online reporting platforms, each component requires specific validation approaches. The goal is to ensure that data inputs remain free from issues, business logic continues operating correctly, and outputs maintain consistency across all downstream applications.

Implementing Shift Left and Shift Down Testing

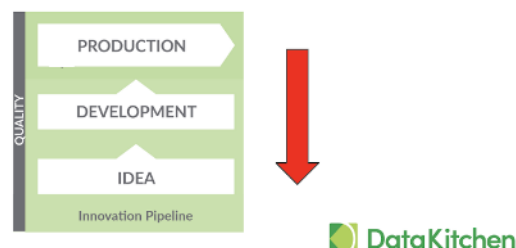
Modern data engineering practices emphasize two complementary testing philosophies that significantly improve data quality while reducing costs and complexity. Shift Left testing involves executing data quality tests earlier in the data pipeline lifecycle, ensuring that issues are identified and resolved before they propagate downstream. This approach recognizes that early detection dramatically reduces the effort and cost required to address data quality problems.

Shift Down testing embraces integration and end-to-end testing methodologies, validating not just isolated components but also the interactions and cumulative impacts within the entire data ecosystem. Rather than testing individual transformations in isolation, Shift Down approaches validate the complete data journey from source to consumption, ensuring that the entire system functions cohesively.

'Shift Left' executing data quality tests earlier in the data pipeline lifecycle—so issues are identified and resolved before they propagate downstream.



'Shift Down' use integration and end-to-end testing methodologies, validating not just isolated components but also the interactions and cumulative impacts within the entire data ecosystem.



The economic benefits of these approaches become clear when considering the cost implications of data quality issues discovered at different stages. The industry-standard 1:10:100 rule, developed by George Labovitz and Yu Sang Chang in 1992, demonstrates that the cost of preventing poor data quality at the source equals approximately one dollar per record, while remediation after creation costs ten dollars per record, and the cost of failure—doing nothing until customers discover issues—reaches one hundred dollars per record.

This cost structure makes a compelling case for comprehensive testing at the earliest possible stages. It becomes ten times cheaper to find problems in data at the source or raw layer compared to discovering them in integrated data or tool-modified outputs. The financial impact extends beyond immediate remediation costs to include reputation damage, regulatory penalties, and the opportunity cost of incorrect business decisions based on faulty data.

Measuring Test Coverage in Medallion Architecture

Effective measurement of test coverage requires systematic approaches that ensure comprehensive validation across all layers and components of your Medallion architecture. The measurement framework operates on multiple levels, each targeting different aspects of data quality and system reliability.

At the foundational level, every table within your architecture should have dedicated tests. This includes tables in Bronze, Silver, and Gold layers, ensuring that no data repository operates without quality validation. The minimum standard calls for at least two tests per table, typically focusing on consistency checks that validate volume, freshness, and schema adherence.

Diving deeper, every column in every table requires its own testing regime. Column-level testing extends beyond simple existence checks to validate data types, value ranges, null constraints, and adherence to patterns. The recommended minimum of two tests per column ensures that both basic integrity and business-specific validation occur for each data element.

Business-specific metrics represent another crucial dimension of test coverage. Every significant business or domain-specific metric should have custom tests that validate not just technical correctness but business logic accuracy. These tests often require in-depth domain knowledge and represent the highest-value validations in terms of business impact.

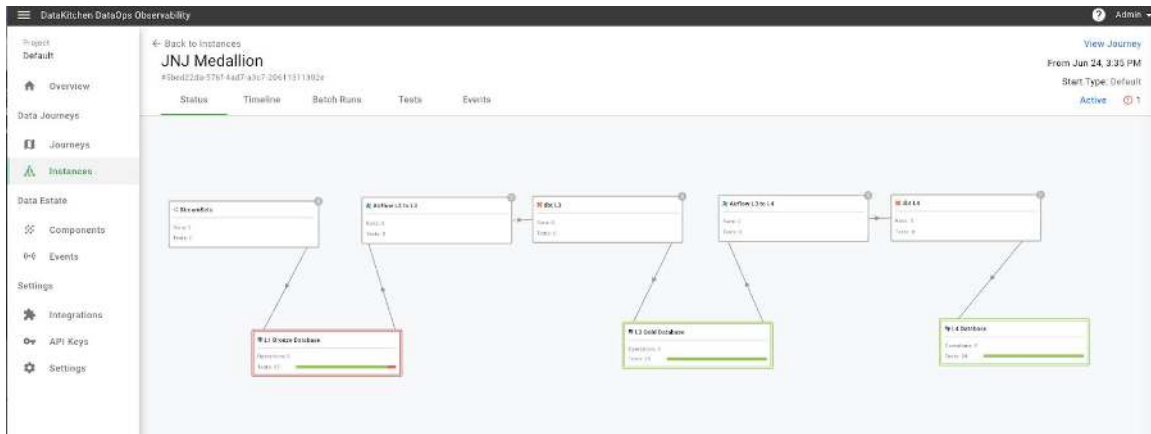
Treat data quality tests like first-class citizens. Tests aren't just QA – they're living proof of what the system should do. And by 'tests,' we mean numerous tests —hundreds, thousands, covering every table.

Tool-level testing completes the coverage framework by ensuring that every tool in your data processing pipeline operates correctly. This includes checking logs for errors, monitoring performance metrics, validating task status and substatus results, and tracking timing and duration patterns. Each tool should have at least one error check, with additional monitoring for metrics, status validation, and performance tracking.

Practical Test Coverage Implementation

To illustrate the scale of comprehensive test coverage, consider a typical Medallion architecture implementation with three database levels (L1, L2, L3), three primary tools (StreamSets, Airflow, dbt), and five distinct jobs or workflows requiring monitoring.

In the L1 Bronze layer, assuming 100 tables with an average of 10 columns each, the minimum testing requirement includes 200 table-level tests and 2,000 column-level tests. The L2 Silver layer, with similar table and column counts, requires another 200 table tests and 2,000 column tests. The L3 Gold layer, which contains fewer tables (perhaps 10) with more columns (averaging 30), requires 20 table tests and 600 column tests, in addition to domain-specific business metric tests.



Tool-level monitoring adds another layer of validation, with three tools requiring error checking and six jobs needing 12 separate checks for various problems, including status validation and performance monitoring. This example demonstrates how comprehensive test coverage quickly scales to thousands of individual tests across a moderately complex Medallion implementation.

The medallion architecture above has three database levels (I1, I2, I3), three tools (streamsets, airflow, dbt), and five jobs/workflows to monitor

Gold/L3: 10 tables + 30 Columns: Minimum 20 Table Tests and 600 Column Tests, plus a handful of Domain Specific Business Metric Tests

Manual Writing Data Quality Test Is Time Consuming

The economic argument for comprehensive test coverage becomes compelling when considering the actual cost of data quality issues. Every problem discovered earlier in the development and processing pipeline costs significantly less to resolve than issues found in production or by customers. Tests represent gifts that data engineers give to their future selves, providing early warning systems that prevent costly downstream problems.

Ensuring coverage for every table and tool creates a comprehensive safety net that protects both data quality and organizational resources. However, manually writing (or configuring) those tests can be time-consuming. Looking at the example below, it could take months to manually code all the necessary tests.

An example

- With TestGen, a junior operator can generate 50,20 tests with **two steps** (profile, generate tests)
- It would take a trained Data Engineer **14 months** to achieve the same results – with no time for meetings, breaks, or vacations

Manual Test Development Cost	
Number of Tests	5,020
Minutes Per Test	30
Hours	2,510
Days	314 8 hours/day
Weeks	63 5 days/week
Months	14 4.35 weeks/month
TestGen	minutes

Open-source tools for automated test generation make comprehensive coverage economically feasible, even for resource-constrained teams. Rather than requiring massive manual investment in test creation, automated tools enable junior operators to generate thousands of tests with minimal effort, democratizing access to enterprise-grade data quality assurance. This is why we created open-source [DataOps Data Quality TestGen](#).

Test Coverage + Data Journeys Provide Precision.

Understanding the distinction between data lineage and data journeys is crucial for implementing adequate test coverage. Data lineage provides valuable insights into where problems might spread throughout your system, much like a building blueprint shows potential paths for fire spread. However, data journeys with comprehensive test coverage reveal where problems are actually occurring, much like a fire alarm control panel indicating active emergencies.

Data lineage represents educated guesses about impact, showing dependencies and relationships between data elements. While valuable for understanding system architecture, lineage alone cannot definitively identify which components are experiencing actual problems. Complete test coverage, as displayed through data journeys, provides clarity on the real-time health of the system and the status of data quality.

When a Fireman walks into your building, which is more important? A Blueprint or a Fire Alarm Control Panel?



Data Lineage Shows Where Fire **Might** Spread—Data Journeys With Tests Show Where It's **Actually** Burning.

Data Lineage Is Guess On The Impact, Complete Test Coverage Shown In A Data Journey Is Clarity

Data Lineage May Show **Five Tables Impacted**: But **Only One Has An Error**.

The practical difference becomes apparent during incident response. Data lineage might indicate that an upstream issue could impact five tables, but comprehensive test coverage reveals that only one table actually contains errors. This precision enables faster resolution and prevents unnecessary investigation of unaffected components.

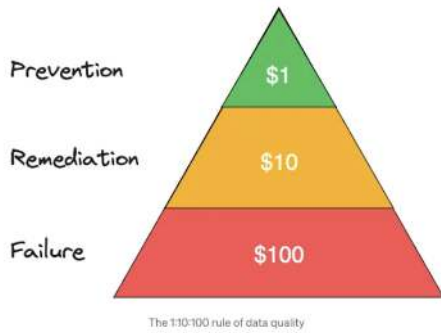
Conclusion and Future Considerations

Comprehensive test coverage in Medallion data architecture represents more than just technical best practice—it embodies a commitment to data reliability that enables organizational trust and effective decision-making. Having customers uncover data errors creates embarrassment and potential damage that rigorous quality testing can prevent.

Quality testing at every stage of the Medallion architecture—Bronze, Silver, and Gold—catches issues early while minimizing the risk of data inconsistencies surfacing in customer-facing insights. Prioritizing thorough automated testing and continuous monitoring enables data teams to uphold data integrity and foster trust across the organization.

The future of data engineering depends on treating data quality, at every level, with the same rigor applied to software quality.

Automatically testing both the data and the tools that act upon data creates comprehensive protection against quality degradation. Since nobody has time to write the thousands of tests required for comprehensive coverage manually, open-source enablement becomes crucial for the widespread adoption of best practices.



Every Problem You Find Earlier is Cheaper!

- Tests are the gift you give to you future self
- The 1:10:100 rule applies
- Shift left and shift down testing
- Make sure every table is covered
- Make sure every tool is covered

Tools like Open Source DataOps Data Quality Testgen automate the creation of tests and make test coverage easy

Test coverage and quality dashboards foster organizational best practices by providing visibility into data health and enabling proactive quality management. As Medallion architectures continue to gain popularity in data lakehouse implementations, teams that master comprehensive test coverage will deliver more reliable data products and maintain competitive advantages through superior data quality assurance.

The investment in [comprehensive test coverage](#) pays dividends through reduced firefighting, improved stakeholder confidence, and more reliable, data-driven decision-making. For data engineering teams serious about delivering production-grade data products, implementing systematic test coverage across their Medallion architecture represents not only a technical improvement but a fundamental shift toward sustainable and trustworthy data operations.

Author: Chris Bergh | Published: Aug 11, 2025

Original article: <https://datakitchen.io/data-quality-test-coverage-in-a-medallion-data-architecture/>

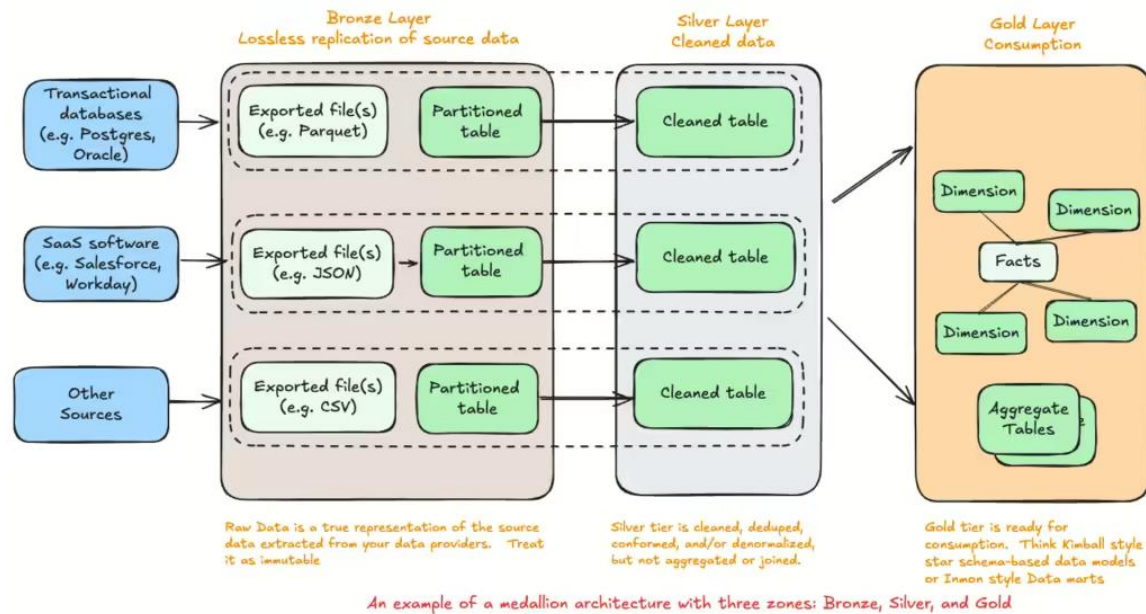
The Race For Data Quality in a Medallion Architecture

The Medallion Data Lakehouse Architecture Has A Unique Set Of Data Quality Challenges. Find Out How To Take The Gold In This Tough Data Quality Race!



In the Olympic quest for data quality, the Medallion Data Lakehouse Architecture is the ultimate marathon, where data teams strive to go beyond the Bronze and Silver to take the Gold. Each stage represents a new data accuracy and quality challenge; the Medallion architecture combines raw, cleansed, and analysis-ready data in a powerful three-tier system. However, ensuring data quality at every layer is a challenging feat. Just as elite athletes rely on rigorous training and constant feedback to reach peak performance, data teams need a flexible, iterative approach to spot and address quality issues early. Adopting a DataOps mindset transforms data quality into a dynamic, continuous process, allowing teams to improve incrementally through feedback loops and adapting to issues. This approach empowers individuals across the organization to champion data quality, enabling a fast-paced yet precise journey toward excellence. In this data quality race, DataOps makes data quality testing a natural, built-in practice, laying a foundation for constant refinement and ensuring that data teams remain on track to win Gold in accuracy, reliability, and organizational trust.

It sounds great, but how do you prove the data is correct at each layer? How do you ensure data quality in every layer? How do you get both speed of execution and accuracy of testing? Read more to find out!



Bronze, Silver, and Gold – The Data Architecture Olympics?

The [Medallion architecture pattern](#) is gaining traction among data teams. It is a layered approach to managing and transforming data. The Medallion architecture is a design pattern that helps data teams organize data processing and storage into three distinct layers, often called Bronze, Silver, and Gold. By systematically moving data through these layers, the Medallion architecture enhances the data structure in a [data lakehouse](#) environment. This architecture is valuable for organizations dealing with large volumes of diverse data sources, where maintaining accuracy and accessibility at every stage is a priority.

The Bronze layer is the initial landing zone for all incoming raw data, capturing it in its unprocessed, original form. This foundational layer is a repository for various data types, from transaction logs and sensor data to social media feeds and system logs. By storing data in its native state in cloud storage solutions such as AWS S3, Google Cloud Storage, or Azure ADLS, the Bronze layer preserves the full fidelity of the data. This raw, historical archive is essential for traceability and allows data engineers to load new data into the lakehouse incrementally through batch uploads or real-time streaming. While the Bronze layer doesn't offer the cleanliness or standardization necessary for analysis, it is a critical foundation upon which the Silver and Gold layers are built. Bronze layers can also be the raw database tables. Bronze layers should be immutable.

Next, data is processed in the Silver layer, which undergoes "just enough" cleaning and transformation to provide a unified, enterprise-wide view of core business entities. This stage involves validation, deduplication, and merging of data from different sources, ensuring that the data is in a more consistent and reliable format. For instance, records may be cleaned up to create unique, non-duplicated transaction logs, master customer records, and cross-reference tables. The Silver layer aims to create a structured, validated data source that multiple organizations can access. This intermediate layer strikes a balance by refining data enough to be useful for general analytics and reporting while still retaining flexibility for further transformations in the Gold layer. Expect the schema to be similar but not the same as the Bronze layer. Alternatively, the Data Vault methodology can be applied to the Silver layer, where data is transformed into Hubs, links, and satellites.

Finally, the Gold layer represents the pinnacle of the Medallion architecture, housing fully refined, aggregated, and analysis-ready data. Data is typically organized into project-specific schemas optimized for business intelligence (BI) applications, advanced analytics, and machine learning. Whether it's customer analytics, product quality assessments, or inventory insights, the Gold layer is tailored to support specific analytical use cases. This layer is often characterized by de-normalized, read-optimized tables that minimize the need for joins and are structured according to data

warehousing methodologies, such as Kimball's star schema or Inmon's data marts. When data reaches the Gold layer, it is highly curated and structured, offering a single version of the truth for decision-makers across the organization.

We have also seen a fourth layer, the Platinum layer, in companies' proposals that extend the Data pipeline to OneLake and Microsoft Fabric. We recently talked with one company whose Data Journey is a path from SAP & Teradata to ADLS (Bronze/Silver/Gold) and finally to Synapse for usage.

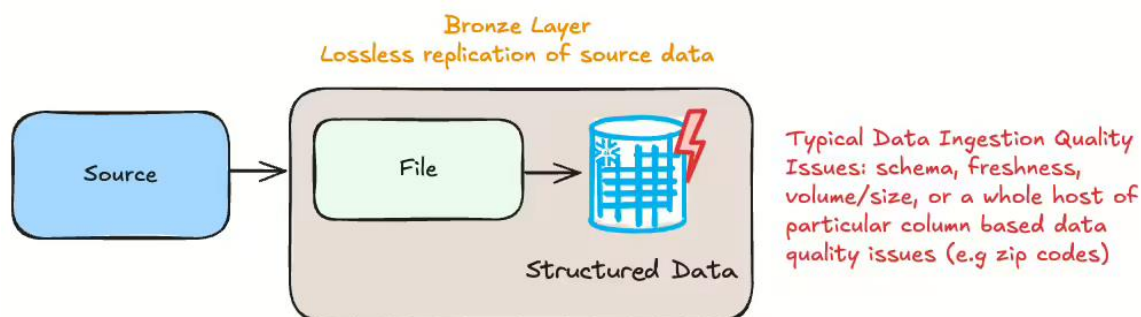
The Medallion architecture offers several benefits, making it an attractive choice for data engineering teams. By preserving raw data in the Bronze layer, organizations have a historical archive they can revert to, ensuring transparency and traceability. The structured flow through Silver to Gold allows flexibility in handling new transformations and aggregations. At the same time, the Gold layer's "single version of the truth" makes data accessible and reliable for reporting and analytics. However, this architecture is not without its challenges. The need to copy data across layers, manage different schemas, and address data latency issues can complicate data pipelines. For businesses requiring near-real-time insights, the time taken to traverse multiple layers may also introduce delays.

Moreover, implementing Medallion architecture involves sophisticated configuration and careful management of changes across all layers, which can complicate day-to-day operations. Finally, the challenge we are addressing in this document – is how to prove the data is correct at each layer.? How do you ensure data quality in every layer?

The Medallion architecture is a framework that allows data engineers to build organized and analysis-ready datasets in a lakehouse environment. By methodically processing data through Bronze, Silver, and Gold layers, this approach supports a variety of use cases. It provides a structured path from raw data to actionable insights, although it requires careful planning and consideration of specific business needs and data quality.

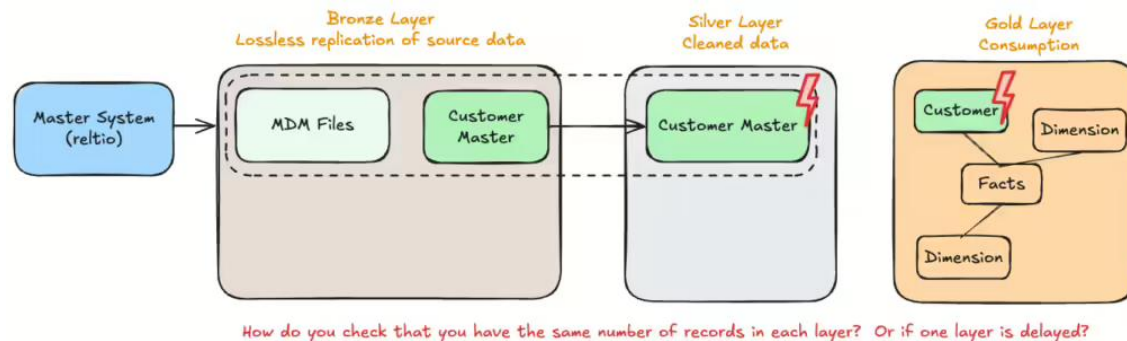
Data Quality Challenges That Cause You To Lose The Medallion Race

In a Medallion architecture, while the structured progression of data through Bronze, Silver, and Gold layers is designed to enhance organization, numerous data quality challenges can still arise, particularly once the architecture is in production. These "Day 2" production quality assurance (QA) issues can disrupt customer satisfaction, introduce inaccuracies, and ultimately undermine the value of analytics. Data engineering teams must understand these problems and incorporate mechanisms to detect and mitigate them. Key challenges include identifying the problem type and finding each issue's root cause and layer before your customer finds it for you and your team!



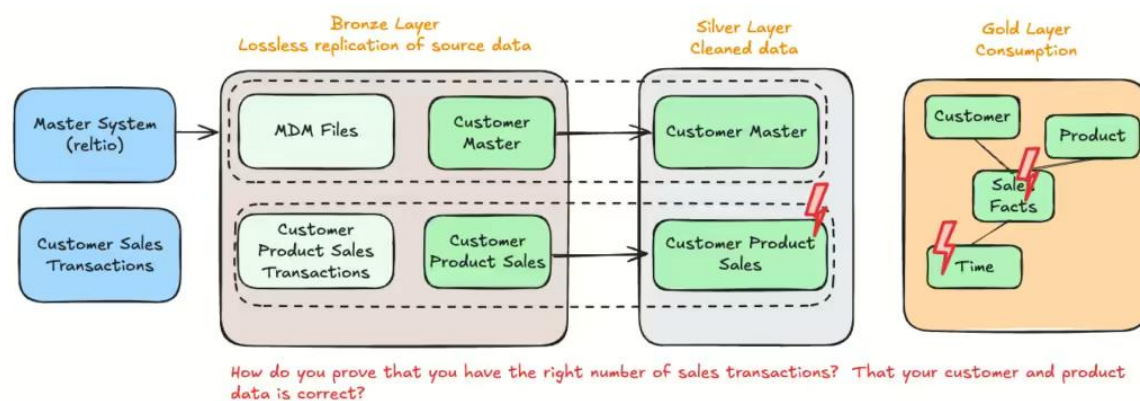
One common problem in production environments is when data fails to load correctly into one or more layers of the architecture, causing interruptions across the data pipeline. For instance, suppose a new dataset from an IoT device is meant to be ingested daily into the Bronze layer. Still, due to connectivity issues or file format mismatches, the load fails. This failure can cascade to the Silver and Gold layers, where downstream dependencies on the raw data prevent those layers from updating accurately. Such issues often go unnoticed until a user or analyst reports missing information in a dashboard or report, by which point the delay has already impacted business decision-making. You have typical data ingestion layer challenges in the bronze layer: lack of sufficient rows, delays, changes in schema, or more detailed structural/quality problems in the data.

Data missing or incomplete at various stages is another critical quality issue in the Medallion architecture. In a Silver layer, for example, merging and deduplication processes rely on complete records to create clean datasets for analysis. Suppose a batch from a transactional system only loads partial records into the Bronze layer due to truncation or data source issues. In that case, the Silver layer can inadvertently generate incomplete or inaccurate views. For instance, a table that shows customer purchase histories could display partial transaction data, leading analysts to underestimate sales or misinterpret customer behavior. Similarly, downstream business metrics in the Gold layer may appear skewed due to missing segments, which can impact high-stakes decisions. Routine checks for row counts, null values, and integrity constraints can help detect missing data early in the pipeline.



Another common issue is data arriving late, which is particularly challenging in real-time or near-real-time systems. When data from various sources does not reach the Bronze layer on time, it can lead to stale insights and missed opportunities in the Gold layer, especially for time-sensitive applications like inventory tracking or marketing campaigns. For example, if data about online customer interactions is delayed due to source system lags, the Gold layer's customer segmentation analysis may fail to reflect recent behavior, leading to irrelevant or poorly targeted campaigns. Late data can also create mismatches in time-based aggregations and reporting, where analytics teams might unknowingly operate on outdated information.

Detailed Data Quality issues are another recurring challenge in the Medallion architecture. In the Silver layer, the transformation processes required to standardize, deduplicate, and validate data can inadvertently introduce new quality problems, such as data type mismatches, improper joins, or invalid replacements. When moving data from the cleaned and enriched Silver layer to the ready-for-consumption Gold layer. An operation to merge customer data across multiple sources might incorrectly aggregate records due to mismatched keys, leading to inflated or deflated metrics in the Silver layer. Issues can propagate unnoticed to the Gold layer, where they can alter the final analytical outputs.



In production environments, these issues require proactive data quality monitoring and quick response capabilities to ensure that data flowing through the Medallion architecture remains accurate, timely, and complete. By incorporating automated alerting systems, regular QA checks, and well-defined SLAs for data loading, data engineering teams can better manage Day 2 production challenges, helping to preserve the integrity and reliability of their analytical outputs.

Get Off The Blocks Fast: Data Quality In The Bronze Layer

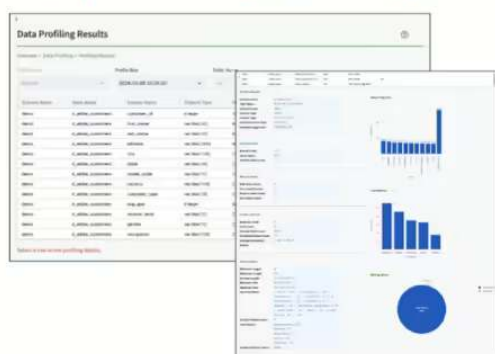
Effective Production QA techniques begin with rigorous automated testing at the Bronze layer, where raw data enters the lakehouse environment. The first part of Data Quality (DQ) training focuses on onboarding new datasets, which includes evaluating and cleansing them before they progress through the pipeline. This process involves detailed data profiling and column-by-column hygiene checks, which help detect hidden issues like null values, incorrect formats, and unexpected duplicates that can undermine data accuracy. During this phase, data teams must decide whether to patch problematic data, push back to data sources for corrections, or pass it through if it meets quality standards.

As data ingestion transitions to a continuous flow, the second part of DQ training equips engineers to monitor schema consistency, row counts, and data freshness, ensuring data integrity over time. This involves setting up automated, column-by-column quality tests to quickly identify deviations from expected values and catch emerging issues before they impact downstream layers. Four key areas for data quality check during continual data updates are:

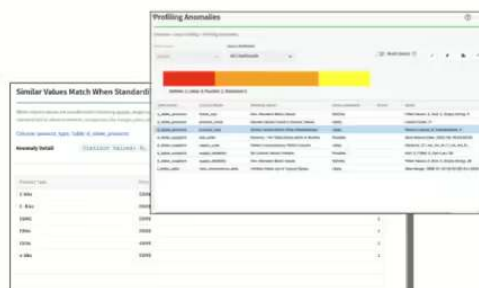
Bronze Layer QA: New Data Set Evaluation

	Use Case	When Does it Happen	Solution
Data Evaluation:	New data set evaluation and cleansing	Before New Data Sources Are Added To Production	Evaluate new data, find data hygiene issues, and communicate with your data providers.

DataOps TestGen DQ Data Profiling



DataOps TestGen DQ Data Hygiene Suggestions

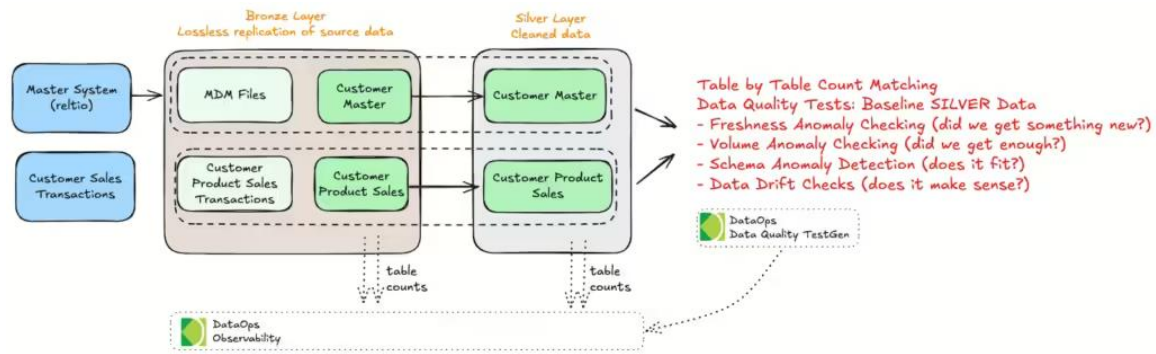


DataKitchen

Tools like DataOps Data Quality TestGen can be leveraged to automate test generation, making it easier for data teams to implement robust QA checks. For those looking to deepen their expertise, obtaining certification in data observability and data quality testing can further enhance their skills and proficiency. The certification program offered by DataKitchen provides a structured approach to mastering these concepts, enabling QA teams to uphold high data quality standards in production environments. For more information, check out the certification at [DataKitchen's website](https://datakitchen.io/certification).

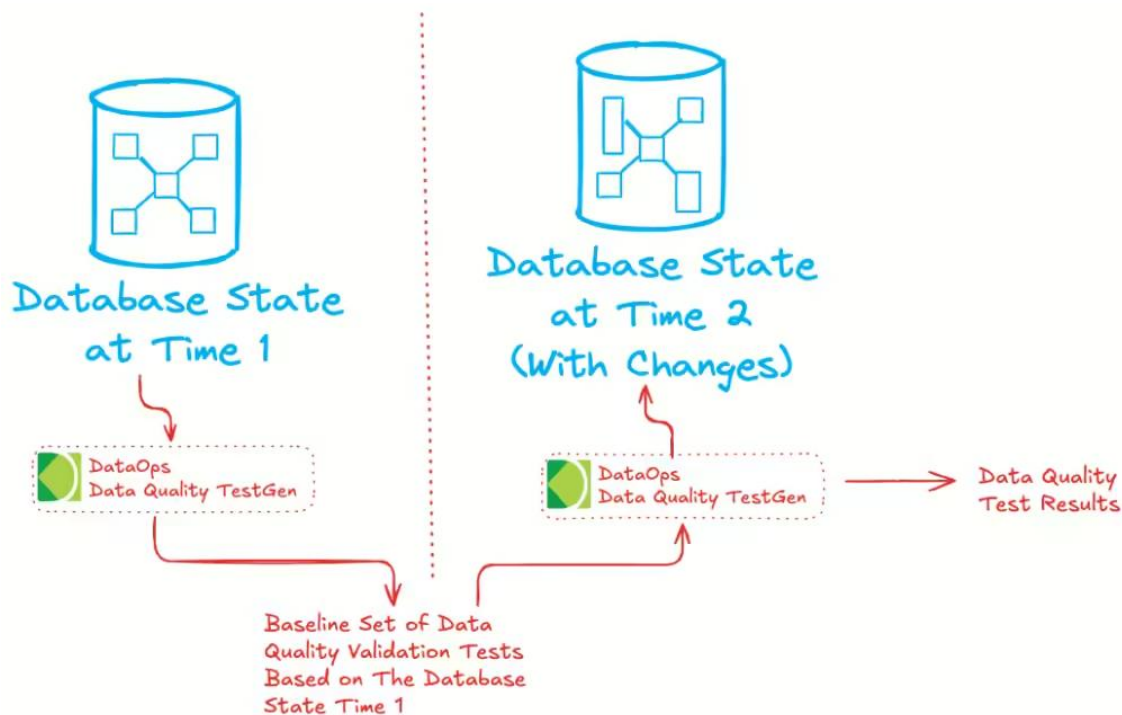
Run With Excellence: Data Quality in The Silver Layer

Testing the Silver layer in a Medallion architecture is crucial for maintaining high data quality, as this is the stage where raw data from the Bronze layer is cleansed, validated, and transformed into a standardized format for broader organizational use. Since this layer serves as a centralized source of trusted data, it's essential to perform comprehensive quality checks to ensure data consistency and reliability. One of the fundamental steps in validating Silver layer data is conducting table-by-table count matching. This involves checking that the number of records aligns with expectations and matches prior ingestion records, preventing discrepancies that could indicate data loss or duplication.



Beyond record counts, several vital data quality tests should be part of Silver layer validation. Freshness anomaly checks verify that the data is current by confirming recent updates or additions, ensuring that the Silver layer reflects the latest information from the Bronze layer. Volume anomaly checking ensures that enough data is being ingested, flagging instances where data might be missing due to issues in the upstream ingestion process. Schema anomaly detection helps verify that data structures align with expected formats, preventing schema mismatches that could cause issues during data processing. Finally, data drift checks examine whether the statistical properties of the data have significantly shifted compared to historical baselines, which can help identify unexpected changes or trends that may affect data accuracy or stability.

All these checks are similar to the Bronze layer, so you have two options. One option is to copy all these tests from the Bronze layer to the Silver layer and then run them in the Silver. This works if your schemas/ddl in each layer are precisely the same. Another option, if the Silver schema is not the same as Bronze, is to re-generate all your tests (using a tool like DataOps DQ TestGen) baselined in the Silver data. This creates a test set based on the silver schema, allowing you to catch data quality variance. Every time you receive an update of the data, running these checks can ensure the data is correct even if the schema differs slightly from the bronze layer.

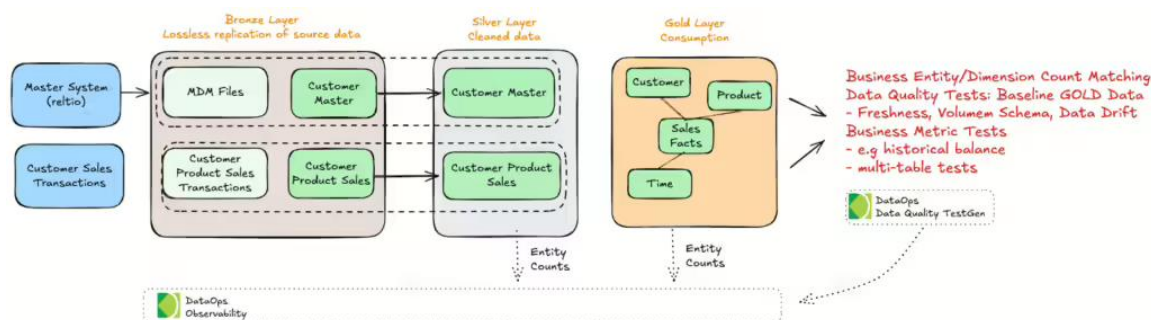


When implementing these tests in the Silver layer, two main approaches can be taken depending on the level of control over the ingestion process. If you can modify or control the ingestion code, data quality tests, and validation checks should ideally be integrated directly into the process. Running these tests immediately upon data arrival ensures that issues are detected early, minimizing the impact on downstream processes. Alternatively, suppose you

do not control the ingestion code. In that case, a practical approach is to set up periodic polling of the Silver layer database to run data quality tests and check for anomalies at scheduled intervals. Regular polling helps identify issues without direct integration into the ingestion code, providing a safeguard to maintain data quality across the Silver layer. This same choice works on any layer: Bronze, Silver or Gold.

Break The Ribbon With Gold Layer Data Quality

In the Gold layer of a Medallion architecture, data quality testing takes on a business-centric focus, as this layer is arranged explicitly in an analysis-ready schema design to support decision-making and reporting. Quality checks in the Gold layer ensure data consistency and integrity and validate that the data accurately reflects critical business metrics and entities. One of the primary testing methods involves count matching between the Gold and Silver layers for each essential business entity or dimension, such as the number of customers, transactions, or products. By verifying that the counts in the Gold layer align with those in the Silver layer, data quality teams can ensure that no data has been lost or inaccurately aggregated in the final transformation stages.



In addition to entity count matching, baseline data quality tests are performed to check for freshness, volume, schema consistency, and data drift in the Gold layer. The Gold layer is most likely to have a unique schema. So, like in the other layers, you need to baseline your test creation in that Gold schema. This creates a test set based on the Gold schema, allowing you to catch data quality variance. Running these checks every time you receive an update of the Gold data can ensure the data is correct.

Furthermore, business metric-specific tests are crucial in the Gold layer, as this layer often consolidates data across multiple tables and sources to produce metrics for business analysis. These tests validate that calculated metrics—such as the number of products sold per region, customer retention rates, or historical balances—are consistent with expectations and historical trends. Multi-table checks are also essential, verifying the consistency of relationships and aggregations across various tables and ensuring that complex business metrics are accurately represented.

Help The Referee With Data Quality Scores

Medallion Architecture and Data Quality Scoring introduce an approach to evaluate and improve data quality at each layer—Bronze, Silver, and Gold—enabling organizations to make data quality an actionable metric that drives better organizational outcomes. In a Medallion architecture, data progresses through these layers, with each stage representing a higher level of cleanliness, enrichment, and readiness for analysis. Data engineering and quality teams can continuously monitor, compare, and improve data quality by applying a scoring system to each layer. This ensures that the final, analysis-ready Gold layer meets the organization's highest accuracy, completeness, and reliability standards.



In the Bronze layer, data quality scoring focuses on foundational aspects, as this layer contains raw, unprocessed data. Scores are typically based on attributes like completeness (e.g., missing values), freshness (e.g., data recency), and schema conformity (e.g., alignment with baseline structures). A high score at this stage indicates that raw data is being captured accurately and is ready for cleansing and transformation in the Silver layer. Low scores here can highlight data source issues, prompting corrective actions upstream to ensure better data quality early in the pipeline. Comparing Silver layer scores with Bronze helps identify the efficiency of cleansing and transformation processes. If Silver scores do not significantly improve over Bronze, it may indicate gaps in data processing, pointing to areas where further data enrichment or error-handling routines are needed.

In the Gold layer, where data is fully prepared for analysis, quality scores primarily focus on business relevance, metrics accuracy, and decision-making readiness. Scoring in this layer includes checks on aggregated metrics, consistency in multi-table relationships, and stability of key business indicators like sales performance or customer segments. Since this layer is closest to end-users, a high score in the Gold layer is critical for building organizational trust in data-driven insights. By comparing Gold layer scores with Silver, teams can evaluate whether transformation processes have effectively prepared data for high-stakes analysis. Additionally, discrepancies between Silver and Gold scores can reveal issues like flawed aggregations or business logic misalignment.

Win the Gold With DataOps Data Quality

In a Medallion architecture, maintaining data quality must be a continuous and iterative process that can be challenging to address through traditional, upfront planning approaches—defining all data quality requirements upfront often results in missed issues that only surface later, causing costly delays or setbacks. By adopting a DataOps approach, data quality becomes a dynamic cycle that allows teams to improve and refine data through iterative feedback loops incrementally. This approach maximizes speed while not compromising accuracy, enabling data engineers and quality specialists to identify and correct issues early on. [DataOps Data Quality](#) in a Medallion architecture emphasizes empowering individuals within the organization who, while they may not have direct authority, can influence and drive change.

An iterative DataOps cycle starts with measuring data to establish a baseline, followed by evaluating data quality through scoring systems that assess key metrics like accuracy, completeness, and consistency. Using these scores, teams can generate or improve data quality tests tailored to specific issues, such as missing values or anomalies, making quality monitoring proactive and targeted. Continuous data quality scoring then provides ongoing insights, tracking improvements and highlighting areas that need attention. Ultimately, the DataOps approach allows organizations to get data quality processes working quickly and then iterate and improve over time. Documentation and analysis become natural outcomes, not barriers to progress. By starting with testing and measurements, even before standards are fully established, organizations can build a foundation for continuous improvement. The faster the iteration, the more organizations learn, refine their processes, and elevate their data quality standards.

No one likes their customer to find data errors. Data testing in a Medallion architecture is essential for maintaining high data quality across each layer, ensuring that data becomes progressively cleaner, more consistent, and analysis-ready. While this architecture provides a structured path from raw to refined data, pinpointing and troubleshooting errors can make it more challenging as data traverses multiple layers. Quality testing at every stage—Bronze, Silver, and Gold—not only helps catch issues early but also minimizes the risk of data inconsistencies surfacing in customer-facing insights. After all, having a customer uncover a data error is always embarrassing and potentially damaging, so rigorous quality assurance within the Medallion architecture is critical. By prioritizing thorough testing and continuous monitoring, data teams can uphold data integrity, fostering trust in data across the organization.

[DataOps Data Quality TestGen](#) is a data quality verification tool that does five main tasks: (1) data profiling, (2) new dataset screening and hygiene review, (3) algorithmic generation of data quality validation tests, (4) ongoing production testing of new data refreshes (5) continuous periodic monitoring of datasets for anomalies, and (6) Data Quality Scoring. ([GitHub](#))

Author: Chris Bergh | Published: Nov 5, 2024

Original article: <https://datakitchen.io/the-race-for-data-quality-in-a-medallion-architecture/>

Critical Data Elements: Your Shortcut to Data Governance That Actually Works

💥 80% of data governance initiatives fail. Not because of tools. Not because of frameworks. But because the business isn't involved, and no one agrees on what data truly matters. That's where Critical Data Elements (CDEs) change everything.



Critical Data Elements (CDEs) are specific data points that are vital for core business functions and processes. Unlike the traditional “all data matters” approach, CDEs recognize a key truth: some data elements are much more important than others. These aren’t just any data points—they are the backbone of your operations, the foundation of your decision-making, and often the difference between business success and failure.

Identifying CDEs is a vital step in data governance because it changes how organizations handle data quality. By focusing on the quality and accuracy of these critical elements, organizations can run more smoothly, reduce risks, and make better data-driven decisions.

What Defines a Data Element as “Critical”?

High Impact Errors: Errors in CDEs can lead to major issues like financial losses, dissatisfied customers, or legal complications. For instance, incorrect customer account numbers result in failed transactions, while inaccurate patient medication data can jeopardize lives. **Business Process Backbone:** Critical data elements are essential for key business functions, serving as the gears that enable smooth operations. **Regulatory Compliance:** Certain data must meet legal standards. Regulations such as GDPR, BCBS 239, and HIPAA not only suggest good data practices but also mandate them.

While specific CDEs differ across industries, here are some real-world examples. In finance, focus areas include customer account numbers, transaction details and amounts, security codes, authentication data, credit scores, and risk assessments. Healthcare prioritizes patient identifiers, medical history, diagnoses, medication details, dosages, and insurance and billing information. Retailers emphasize product data and SKUs, customer purchase history, pricing, promotional codes, inventory levels, and supplier data.

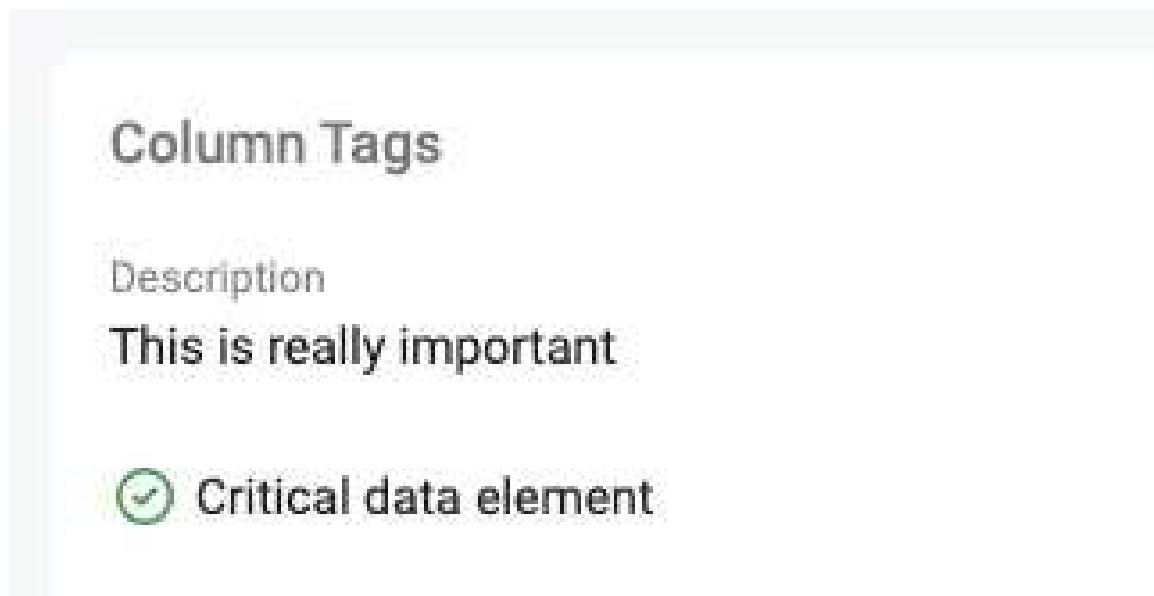
The Three-Step Framework for Operationalizing CDEs

Use a manual scoring model to assess which data elements impact customer experience or regulatory reporting, are used to calculate KPIs, appear in multiple systems or dashboards, are identified as critical by the business, or trigger downstream processes. Each criterion should be weighted based on your organization's specific priorities. As a rule of thumb, if a data element scores ≥ 5 points across these criteria, it's a CDE that deserves prioritization.

Automated CDE Discovery: While manual scoring provides business context, modern data platforms can accelerate CDE identification through intelligent automation. [DataOps Data Quality TestGen](#) uses data profiling and semantic models to automatically surface potential Critical Data Elements, dramatically reducing the time and effort required for discovery.

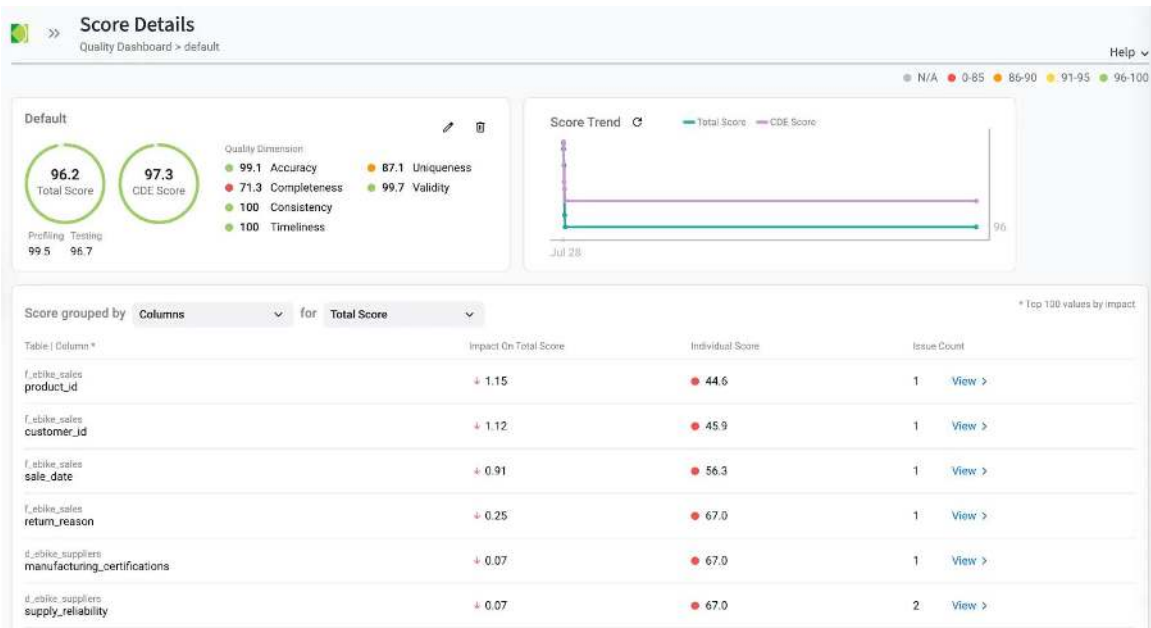
Data Profiling Intelligence With Semantic Model Recognition: Advanced data quality tools can recognize common business patterns and data types that typically represent CDEs. Customer identifiers, financial amounts, regulatory codes, and other semantically significant elements are automatically tagged for review, saving data teams from manually sifting through thousands of data points.

This automated approach doesn't replace business judgment—it enhances it. Your data teams can focus on validating and refining the algorithmically identified CDEs rather than starting from scratch, significantly accelerating your path to effective data governance.



Apply strict data quality controls to your CDEs: Implement rigorous data validation rules, ensure consistent naming conventions, track comprehensive metadata, and assign clear ownership. If you can't name the owner, it's not certifiable. Accountability is non-negotiable for CDEs.

[DataOps Data Quality TestGen](#) has revolutionized this certification process by automatically creating over 30 different test types for CDEs with just the click of a button. These automated tests include completeness checks, format validation, range verification, referential integrity tests, statistical anomaly detection, and business rule validation. What once required weeks of manual test development can now be accomplished in minutes, allowing data teams to focus on governance strategy rather than getting bogged down in technical implementation. The platform intelligently selects the most appropriate test types based on data profiling results and semantic understanding of each CDE, ensuring comprehensive coverage without overwhelming teams with unnecessary complexity.



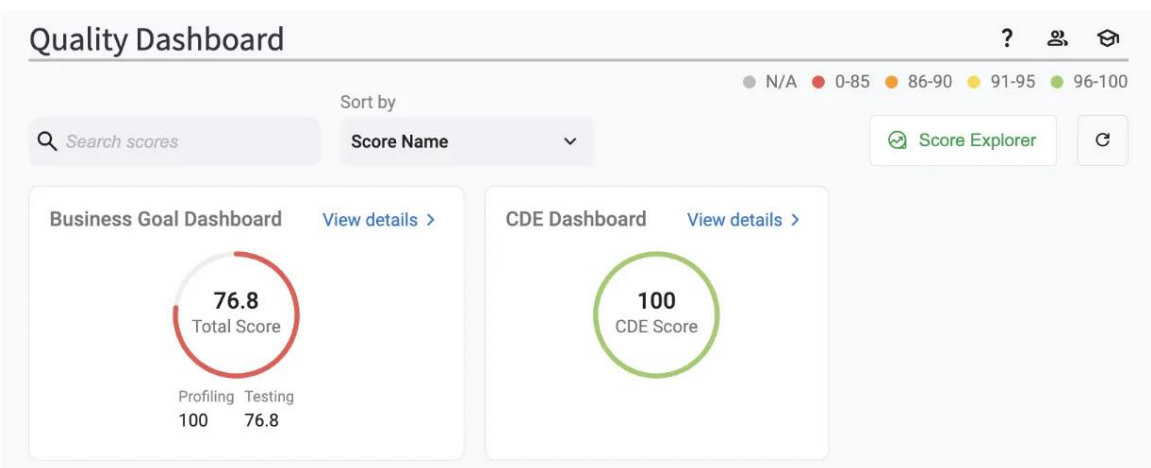
Don't just "fix data." Manage CDEs like products across their lifecycle: Define → Deliver → Monitor → Improve. This mirrors agile product development, treating your most important data with the same rigor you'd apply to your core products.

The "Monitor" phase of CDE lifecycle management is where data quality dashboards become indispensable. Just as product teams rely on analytics dashboards to track user engagement and performance metrics, data teams need comprehensive visibility into their Critical Data Elements.

Modern data quality platforms like [DataOps Data Quality TestGen](#) provide open-source solutions that make CDE monitoring accessible to organizations of all sizes. These dashboards offer:

Real-Time CDE Health Monitoring: Track the quality metrics of your most critical data elements in real-time, with alerts when thresholds are breached. Instead of discovering data quality issues during monthly reports, you catch them as they happen.

Historical Trend Analysis: Visualize how your CDEs perform over time. Are customer account numbers becoming less accurate? Is there a pattern to when product pricing data fails validation? These trends reveal systemic issues that need addressing.



Automated Quality Scoring: Implement the scoring models discussed in Step 1 directly into your dashboard. Watch as data elements automatically get flagged as critical based on their usage patterns, business impact, and quality metrics. Data Quality Tests show you precisely what is wrong.

Collaborative CDE Metadata: Modern DQ dashboards enable data stewards, business users, and IT teams to collaborate on quality issues directly within the platform. Comments and metadata tracking keep everyone aligned on CDE efforts.

The beauty of open-source solutions like TestGen is that they democratize data quality monitoring. You don't need enterprise-level budgets to implement sophisticated CDE tracking—you need the right approach and commitment to treating your critical data like the valuable assets they are.

The Focused Impact of CDEs on Data Quality Testing

CDEs transform data quality testing by offering: Prioritization of efforts—allowing organizations to concentrate resources on the most impactful data elements instead of trying to perfect everything. Targeted data quality rules—applying validation, cleansing, and standardization processes where they are needed most. Enhanced regulatory compliance—making it easier to meet legal requirements when knowing precisely which data elements are regulated. Improved decision-making—providing a dependable base for data-driven choices through high-quality CDEs. Risk mitigation—protecting critical data to help prevent operational, financial, and reputational damage.

Breaking the 80% Governance Failure Rate

CDEs address the root cause of data governance failure: lack of business alignment. When you can point to specific data elements that directly impact revenue, customer satisfaction, or compliance, suddenly everyone understands why data quality matters. This isn't about perfect data everywhere—it's about perfect data where it counts. Ask yourself and your team:

CDEs serve as your quick access to data value, enabling focused governance, simplifying compliance, and increasing trust in AI and BI projects. More significantly, they turn data governance from a cost center into a strategic advantage. The organizations that succeed will be those that identify their critical data elements and give them the attention they require.

Author: Chris Bergh | Published: Jul 31, 2025

Original article: <https://datakitchen.io/critical-data-elements-your-shortcut-to-data-governance-that-actually-works/>

Data Quality Circles: The Key to Elevating Data and Analytics Team Performance

DataOps Quality Circles are focused teams within data and analytics organizations that meet weekly or monthly to drive continuous improvement, quality automation, and operational efficiency. By leveraging the principles of DataOps, these circles ensure that data processes are error free, consistent, and aligned with business goals.

Got Errors? Start A Data Quality Circle



According to a study by HFS Research, 75 percent of business executives do not have a high level of trust in their data. High-quality data underpins the reliability of insights, models' accuracy, and decision-making processes' efficacy. Yet, ensuring this quality is often a complex challenge, requiring technical solutions and a cultural shift within teams. This is where the concept of Data Quality Circles comes into play—a structured yet collaborative approach modeled after the well-established [Quality Circle](#) method in manufacturing and service industries. It is a step to help avoid the [terrible war room](#) and the stress of a [data team crisis](#).

At the heart of Data Quality Circles is a culture that encourages learning from mistakes without fear of retribution. This “[no shame, no blame](#)” approach is crucial for fostering an environment where team members feel safe sharing and understanding incidents. The mantra “love your errors” is not about celebrating mistakes but viewing them as opportunities for improvement. This mindset is essential for continuously enhancing data operations, where the goal is to systematically identify and address the root causes of data quality issues.

The primary objective of a Data Quality Circle is to improve the quality and, therefore, the efficiency of data operations. This is achieved by analyzing individual production incidents or groups of incidents categorized by root cause, particularly when there is a high volume of issues. Whether it's a failed ETL job, a broken data pipeline, or a report with inaccurate figures, each incident provides a learning opportunity. The focus is not just on fixing the immediate problem but on understanding how to prevent similar issues from occurring in the future.

Data Quality Circles are akin to retrospectives—structured processes designed to help teams learn from past incidents. However, they go beyond mere analysis by actively driving actionable improvements. The process typically involves the following steps:

Compile a list of recent incidents, support calls, help desk tickets, or bugs. If the incidents are high, use Pareto analysis to identify the most frequent root causes, as 80% of issues typically stem from 20% of causes. Ensure that all attendees are equipped to contribute to the root cause analysis and generate preventive ideas.

Start by identifying the date and description of each incident, along with a timeline of events. Utilize the Five Whys technique to drill down to the root cause. For example: Problem: The vehicle will not start:

Brainstorm potential actions that could prevent similar incidents in the future. These actions should be practical, actionable, and focused on eliminating the root cause.

Create tickets or tasks for each identified action, ensuring a commitment to implementing at least one significant improvement. Document the facts and agreed-upon actions, avoiding any blame or negative language. The focus should remain on constructive problem-solving.

The result of a successful Data Quality Circle is a set of concrete actions aimed at preventing future incidents. These actions should be tracked, implemented, and reviewed to ensure they have the desired impact. Over time, this process leads to a gradual but sustained improvement in the quality of data operations, reducing the frequency and severity of incidents.

Implementing Data Quality Circles in data and analytics teams requires commitment from all levels of the organization. It demands a culture shift, where errors are seen not as failures but as opportunities to learn and improve. By regularly convening to discuss and address data quality incidents, teams can foster a culture of continuous improvement, leading to more reliable data and, ultimately, better business outcomes.

Learning from past mistakes and proactively addressing root causes is invaluable in stressful data and analytic teams. Data Quality Circles provide a structured yet flexible framework for achieving this, helping teams resolve current issues and prevent future ones. They ensure the data foundation remains strong, resilient, improving, and ready to support the business's needs.

DataOps Quality Circles are focused teams within data and analytics organizations that meet weekly or monthly to drive continuous improvement, quality automation, and operational efficiency. By leveraging the principles of DataOps—such as automation, testing, and iterative development—these circles ensure that data and the processes acting on data are error-free, consistent, and aligned with business goals. The goal is to create a culture of accountability and innovation where every team member contributes to improving data quality and operational performance.

Author: Gil Benghiat | Published: Aug 22, 2024

Original article: <https://datakitchen.io/data-quality-circles-the-key-to-elevating-data-and-analytics-team-performance/>

How Data Quality Leaders Can Gain Influence And Avoid The Tragedy of the Commons

Using the ecological idea of the 'Tragedy Of The Commons' as a metaphor for the eternal issue of data quality, we talk about how data quality leaders can leverage Dale Carnegie's 100-year-old ideas on influencing people and wrapping this improvement process with DataOps iterative improvement.

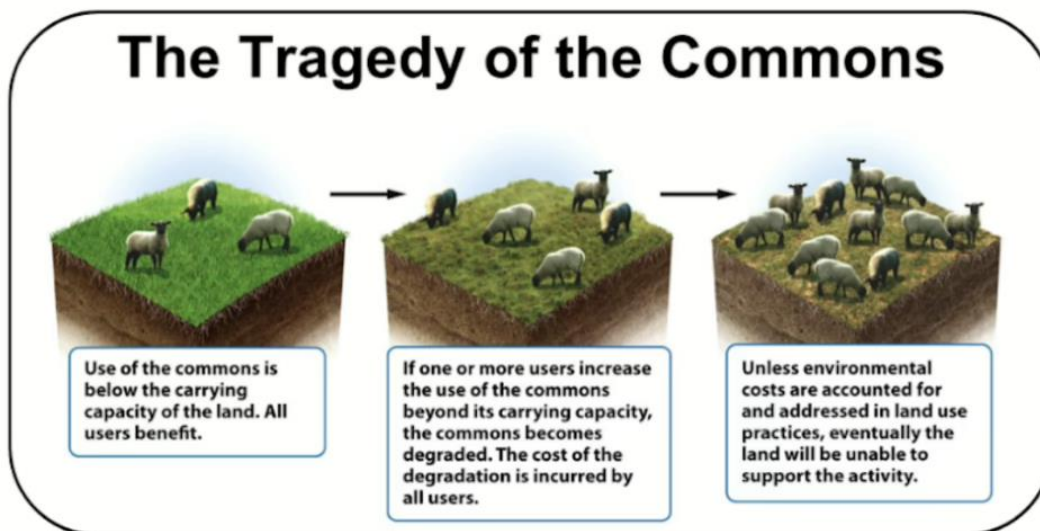


Data quality has long been essential for organizations striving for data-driven decision-making. Despite the best efforts of data teams, poor data quality remains a persistent challenge, leading to distrust in analytics, inefficiencies in operations, and costly errors. Many organizations struggle with incomplete, inconsistent, or outdated data, making it difficult to derive reliable insights. Furthermore, data quality leaders often face the challenge of limited influence within their organizations, making it hard to enforce necessary changes. Without proper visibility and control over data processes, these leaders are left addressing the same recurring problems without meaningful long-term solutions.

Addressing these issues requires a systematic approach that identifies and corrects errors, prevents them from occurring, and gives data quality leaders the influence they need to effect change. This article talks about three ideas: using the ecological idea of the 'Tragedy Of The Commons' as a metaphor for the eternal issue of data quality, how data quality leaders can leverage Dale Carnegie's 100-year-old ideas on how to influence people, and how to wrap this improvement process with DataOps iterative improvement.

Data Quality Is An Example Of The 'Tragedy Of The Commons'

Data quality issues often stem from fragmented ownership, leading to neglect. Organizations treat data as a shared resource, but without clear responsibility, errors accumulate, creating a "[Tragedy of the Commons](#)." The tragedy is that if many people enjoy unfettered access to a finite, valuable resource, they will tend to overuse it and may end up wrecking its value. According to industry surveys, 73% of data practitioners do not trust their data, 60% of data projects fail due to poor data quality, and billions of dollars are lost annually due to data errors. Data owners often focus only on immediate, high-priority issues while missing the systemic challenges of data quality, making it difficult to shift towards long-term improvements. They overgraze their standard data fields, leaving a bare, denuded data landscape.



A shared pasture is overgrazed because each person acts in their short-term interest rather than considering the collective good. Similarly, in data management, different teams use and create or modify data for their specific needs but often do not take responsibility for maintaining its quality for others' use. This results in degraded data integrity over time, with no single entity accountable for ensuring its reliability.

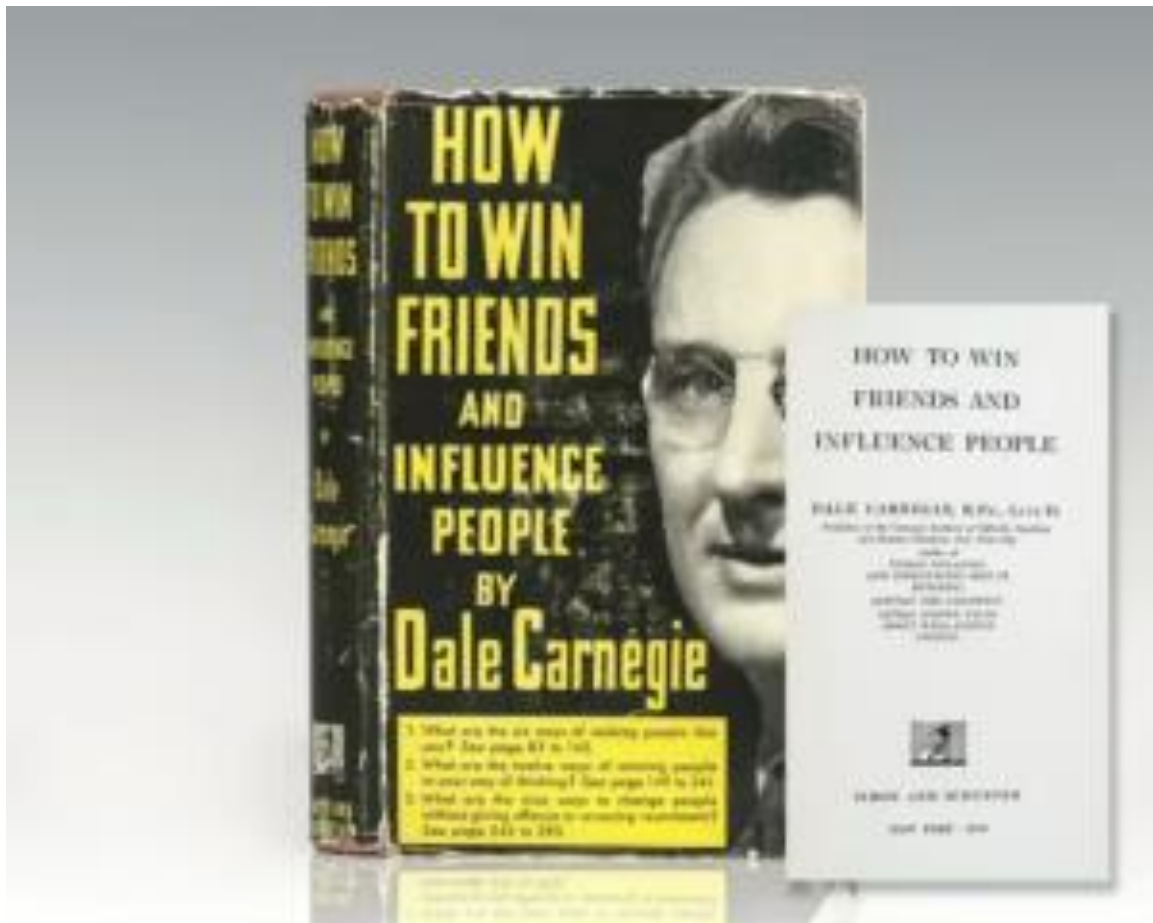
Misaligned incentives exacerbate this lack of ownership. Business teams prioritize speed and efficiency, often bypassing best practices to meet immediate goals. On the other hand, data engineers and analysts are responsible for maintaining data accuracy but struggle to get buy-in from stakeholders, who do not prioritize data quality until it becomes a crisis.

Compounding the issue, data governance policies tend to be unclear or poorly enforced. While organizations might have data quality guidelines, the absence of robust enforcement mechanisms and accountability means these policies fail to function effectively. Consequently, data quality professionals face a challenging situation—they are tasked with maintaining high data integrity, yet they lack the authority to implement necessary changes.

The Influence Factor: Overcoming Organizational Resistance With Persuasion

Data quality leaders often have influence but little power—they can identify problems but lack the authority to enforce change. Let's see how they can take inspiration from Dale Carnegie's principles on persuasion and leadership. In *How to Win Friends and Influence People*, Carnegie emphasizes that people are more likely to respond positively when approached with understanding rather than criticism. This is particularly relevant in data quality, where leaders often struggle to gain support for improvements. One of Carnegie's key principles is, "Don't criticize, condemn, or complain." Data professionals frequently encounter resistance when they approach teams with a list of problems rather than solutions. Instead of pointing fingers at errors in reports or data pipelines, they should frame data quality challenges as opportunities for collective improvement, highlighting the benefits of cleaner data for business decisions.

Another essential Carnegie principle is to "Talk in terms of the other person's interests." Different stakeholders view data through different lenses. A finance leader is concerned with accurate financial reporting, while a marketing executive is more interested in precise customer segmentation. Data quality leaders must tailor their messaging accordingly, demonstrating how improvements align with each team's objectives. They can foster greater collaboration and commitment across departments by presenting data quality as an enabler of business success rather than a technical burden.



Another key principle is, “If you are wrong, admit it quickly and emphatically.” Due to the complexity of data systems and the number of people handling them, errors are inevitable in data quality. Data quality professionals should not hesitate to acknowledge when data issues arise rather than attempting to downplay or obscure them. Transparency fosters trust among stakeholders and speeds up corrective actions. The team should avoid hiding or defending flaws by addressing issues openly and committing to resolutions to build credibility.

Furthermore, Carnegie’s advice to “Use encouragement and make the fault seem easy to correct” is especially relevant. Many data professionals struggle with getting buy-in from engineers and business teams when addressing data issues. Instead of overwhelming teams with exhaustive reports filled with errors, they should present issues to make the solution appear straightforward. This could involve breaking down problems into small, actionable steps and providing clear guidance on resolving them efficiently.

1. Don't Criticize, Condemn or Complain
Use Generative Data Quality To Automatically Identify Multiple Data Issues
8. Talk In Terms Of The Other Person's Interests
Build Data Quality Dashboards on Limited, Specific Data Items
12. If You Are Wrong Admit It Quickly And Emphatically
Start Quickly, Small, and Work Iteratively: DataOps Data Quality
18. Be Sympathetic With The Other Person's Ideas And Desires
Give Them Specific Actionable Problems – They Are Busy!
19. Appeal To The Nobler Motives
Make Multiple Data Quality Dashboards That Align To Organization Goals
29. Use Encouragement. Make The Fault Seem Easy To Correct
Package Data Quality Issues So They Are Easy to Fix
30. Make The Other Person Happy About Doing The Thing You Suggest
Measure And Show Data Quality Improvement Over Time

Finally, Carnegie's principle of "Make the other person happy about doing what you suggest" applies directly to securing collaboration across teams. Data quality improvements should benefit all stakeholders, whether they reduce inefficiencies for engineers, improve forecasting for business leaders, or ensure compliance for regulatory teams. By demonstrating how better data quality leads to smoother workflows, fewer operational disruptions, and more reliable insights, data professionals can motivate others to take ownership of improvements rather than see them as burdens.

A DataOps Approach To Data Quality: Iterative & Impactful

Traditional data quality initiatives often fail due to rigid, time-consuming processes. Embrace an agile, iterative approach that enables teams to start small and act quickly, establishing a working quality process within hours. Rather than spending months manually coding data quality rules, AI-driven test generation significantly reduces effort while maintaining high accuracy. Refining and iterating the tests based on real-world feedback facilitates continuous improvement instead of a static, one-time assessment—Automate data quality insights, providing clear and actionable reports that inform decision-makers. By tracking data quality scores over time, leaders can measure progress and demonstrate the impact of their initiatives. This approach ensures that data teams concentrate on small, high-impact data items, enabling them to gain momentum and quickly showcase the value of quality improvements. The key to success is not attempting to boil the ocean but selecting specific, meaningful data elements tied to business priorities and iterating toward better standards.

DataOps Process: Iterative Data Quality At Scale



- **(Re) Profile Data:**
 - Establish a baseline by measuring the current state of the data with data profiling
 - Get foundation for tracking progress and identifying areas for improvement, repeatedly
- **Generate Data Quality Tests:**
 - Generative Data Quality tests can streamline this process, making it easier to catch problems before they become more significant without coding
- **Data Quality Score Dashboard:**
 - Scoring provide a quantifiable way to assess the health of the data and communicate the results.
 - Scores should be focused on the data elements that matter and reflect top organization needs
- **Share Data Quality Issues Report**
 - Give actionable, specific changes to those who can do the technical work.
- **Influence Data Owners and Engineer:**
 - Once data quality issues have been identified, the next step is to drive the necessary changes within the organization.
 - Influence data owners, system administrators, or other stakeholders to address the identified issues.

The Path Forward: Building Influence Though Rapid, Focused Iterations to Improve Data Quality

Better tools and influence strategies are key to overcoming the ‘Tragedy of the Commons’ in data quality. Data quality professionals must find ways to communicate the importance of clean, reliable data in ways that resonate with different stakeholders.

One practical approach is to use Dale Carnegie’s principles of persuasion. For instance, instead of criticizing teams for poor data practices, data leaders should frame their message around how better data benefits everyone. Using language that aligns with different stakeholders’ interests—such as compliance for legal teams, accuracy for finance, and speed for marketing—helps build collaboration and overcome resistance.

Another strategy is to make data quality improvements feel manageable. If teams see data issues as insurmountable, they are less likely to act. Data leaders can gain incremental buy-in by breaking down problems into smaller, more actionable fixes and demonstrating quick wins. Rather than blaming, encouragement goes a long way in fostering a culture where data quality is seen as a shared responsibility rather than a burden.

Ultimately, the solution to data quality challenges lies in technology and data leaders’ ability to navigate organizational dynamics, build influence, and advocate for systemic changes. By understanding the behavioral and structural barriers to data quality, they can position themselves as essential in shaping data governance and organizational strategy.

DataKitchen has built an open-source tool, [DataOps Data Quality TestGen](#), specifically to address the ‘Tragedy Of The Commons’ in data quality. The tool provides AI-driven insights to leverage Dale Carnegie’s ideas on how to influence people and easy-to-use, full-featured data quality with built-in DataOps iterative improvement.

Author: Chris Bergh | Published: Mar 13, 2025

Original article: <https://datakitchen.io/how-data-quality-leaders-can-gain-influence-and-avoid-the-tragedy-of-the-commons/>

War Rooms Suck

Data analytic team war rooms, often convened for emergency problem-solving, epitomize inefficiency and detract from proactive, value-driven tasks. By leveraging data observability and rigorous testing, issues can be detected and resolved early, negating the need for such reactive measures in the modern era of DataOps.



The meeting room was spacious, adorned with charts, graphs, and several cups of hastily grabbed coffee. I distinctly remember an afternoon set to work with a promising prospect. As my team and I waited, a flurry of messages informed us that the key players from the other side wouldn't be joining. Why? Their entire data analytics team had been hastily yanked into an impromptu "war room" to tackle an emergency data issue. It was reminiscent of another time when I heard about a three-day-long marathon session or even that peculiar incident where a staggering 26 people were engrossed for over 8 hours only to find the root cause of a blank report.

War rooms steal productivity. They often end up being reactive pressure cookers, where resources are pooled in to fix an issue after it's spiraled out of control. A well-functioning data analytics team should proactively foresee and mitigate problems before they balloon into full-blown crises. Data analytic team war rooms represent a firefighting approach. They convine to just 'fix the issue' rather than provide retrospective learning. For every minute that a data professional is locked into these rooms, it's a minute taken away from value-driven, forward-thinking tasks.

Imagine a scenario where the need for such reactive war rooms is significantly diminished if not eradicated. Data observability and rigorous automatic data validation testing can pinpoint and address issues well before they necessitate an all-hands-on-deck emergency meeting. Observability provides the transparency required to monitor data health, ensuring integrity and accuracy at each step. Coupled with robust testing mechanisms, it acts as the first line of defense against anomalies.

In an era where DataOps is revolutionizing how we approach data management, leaning on archaic war room techniques feels incongruous. It's high time we bid farewell to the data analytic team war room!

Author: DataKitchen Marketing Team | Published: Oct 12, 2023
Original article: <https://datakitchen.io/war-rooms-suck/>

Navigating the Storm: How Data Engineering Teams Can Overcome a Data Quality Crisis

A data quality crisis in data engineering is more than a mere technical hiccup; it often signals deeper systemic issues within the team and organizational processes. Let's delve into the root causes, symptoms, and strategies for rapid intervention and long-term improvement.



Ah, the data quality crisis. It's that moment when your carefully crafted data pipelines start spewing out numbers that make as much sense as a cat trying to bark. You know you're in trouble when the finance team uses your reports as modern art installations rather than decision-making tools. But fear not, fellow data wranglers! We've all been there; there's a way out of this crisis of lousy data.

Let's face it: data problems are rarely just about the data. They're like onions (or ogres, if you're a Shrek fan) – they have layers. At the core, we often deal with collaboration hiccups and process potholes. It's like trying to bake a cake when half the team thinks we're making sushi, and the other half juggles the eggs. We've got siloed expertise that would make medieval castle builders proud, documentation so sparse it could win a minimalist art competition, and a reliance on "data heroes" that would make Marvel envious. Add a dash of unreliable source data, a pinch of outdated infrastructure, and a sprinkle of architectural challenges, and voila! You've got yourself a recipe for data disaster.

Data quality issues rarely exist in isolation. They are often symptomatic of broader organizational challenges that require a holistic approach. Let's delve into some of the root causes:

The symptoms of this crisis can feel like a bad case of data indigestion. Suddenly, you're playing a game of "Where's Waldo?" with your process flows, your error rates are higher than a kite on a windy day, and trust in your data evaporates faster than spilled coffee on a hot sidewalk. Before you know it, you're caught in a downward spiral that makes roller coasters look tame. The blame game becomes the new office sport, with fingers pointing in so many directions you could use them as a compass. Pressure mounts heroic all-nighters become the norm, and suddenly, your team starts eyeing the exit door like it's the last lifeboat on the Titanic.

Recognizing the signs of a data quality crisis is crucial for timely intervention. Here are the key indicators:

But don't panic! It's time for some rapid intervention, and no, that doesn't mean hiring a data exorcist (though, at this point, you might be tempted). Start by forming a cross-functional "Quality Circle" team – think of it as your data A-Team, minus the van and mohawks. Gather intel on your current issues, such as planning a heist, but instead of stealing diamonds, you're after those elusive root causes. Prioritize your problems and hunt for quick wins – they're like data-quality comfort food, giving you a momentary respite and a chance to catch your breath. Use tools like the trusty fishbone diagram or a riveting game of "5 Whys" (it's like 20 Questions, but for data nerds). The goal here is to fix something – anything! – to show that there's light at the end of this very messy tunnel.

When facing a data quality crisis, immediate action is crucial. Here's a step-by-step approach to rapid intervention:

As you emerge from the immediate fire-fighting phase, blinking in the sunlight of slightly less terrible data quality, it's time to think long-term. This is where you channel your inner data quality guru and build consensus for sustainable solutions. Dive deep into your data collection practices like Jacques Cousteau exploring the ocean depths. Conduct root-cause analysis with the tenacity of a terrier chasing a squirrel. Develop more sustainable solutions than your colleague's New Year's resolutions and more scalable than your grandma's cookie recipe. Remember, the journey to data quality nirvana is a marathon, not a sprint. By taking steady steps towards DataOps maturity and continuous improvement, you'll not only survive this crisis but come out the other side with a data engineering team that's more robust, more collaborative, and wiser. And who knows? The next time a data quality crisis looms, you might say, "It's an opportunity for improvement" rather than an "opportunity to blame."

While rapid intervention provides immediate relief, using this momentum to drive long-term, sustainable improvements is crucial. Here's how:

A data quality crisis can be pivotal for a data engineering team. While challenging, it presents an opportunity to reassess, realign, and rebuild stronger data management practices. By combining rapid intervention techniques with a commitment to long-term improvements, teams can resolve the immediate crisis and establish a foundation for sustained data quality excellence.

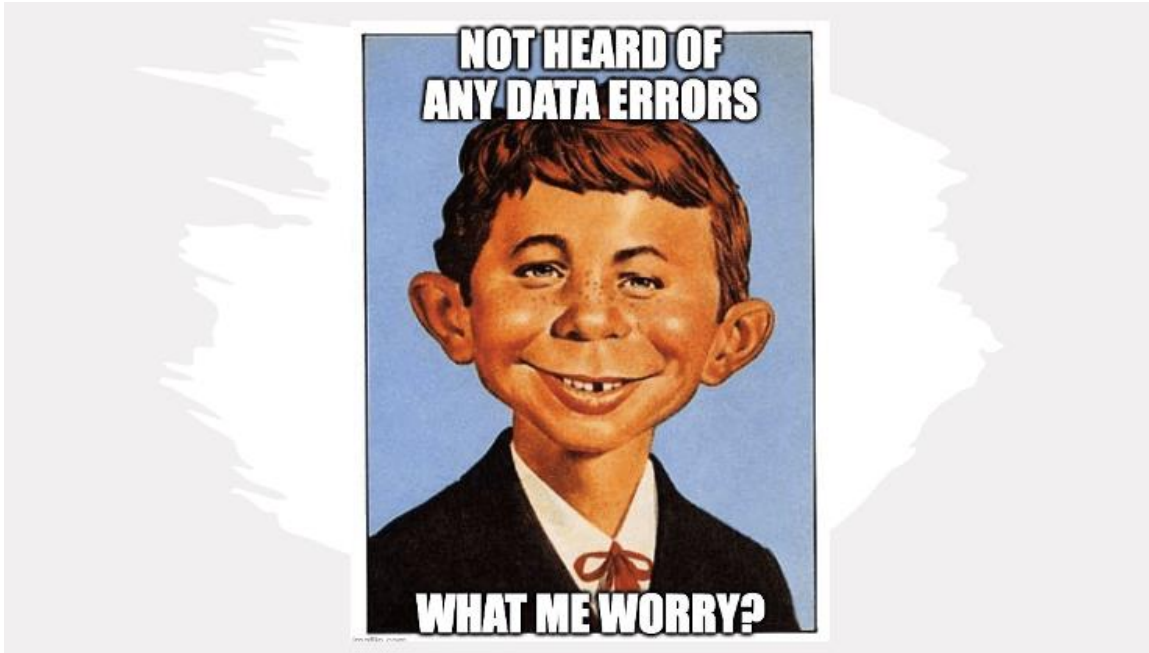
Navigating a data quality crisis requires a calm, empathetic, and systematic approach. By addressing immediate issues through rapid intervention and building consensus for long-term solutions, we can transform a crisis into an opportunity for growth and improvement. Remember, every data quality crisis is a chance to strengthen the foundations of our data processes and build a more resilient, reliable data ecosystem. So, let's steer this ship together, with confidence and a sense of humor, towards calmer waters.

Author: Chip Bloche | Published: Jun 21, 2024

Original article: <https://datakitchen.io/navigating-the-storm-how-data-engineering-teams-can-overcome-a-data-quality-crisis/>

Why Not Hearing About Data Errors Should Worry Your Data Team

Just because you're not hearing about data errors doesn't mean they don't exist. This silence could be a ticking time bomb for underlying issues yet to surface. Here are seven compelling reasons why you should care and be proactive, even when all seems well.



*In the chaotic lives of data & analytics teams, a day without hearing of any data-related errors is a blessing. Your team is on top of things, deliveries are on schedule (you think), and no major complaints are making their way to your desk. It's tempting to adopt the "What, me worry?" attitude under these seemingly calm conditions. But here's the catch: **just because you're not hearing about errors doesn't mean they don't exist**. This silence could be a ticking time bomb for underlying issues yet to surface. Here are seven compelling reasons why you should care and be proactive, even when all seems well.*

Firstly, not hearing about errors doesn't necessarily equate to their absence. Problems are often hidden, either unintentionally due to a lack of observability or deliberately as a cover-up. The absence of noise around data errors could indicate a culture where mistakes are silently corrected or ignored, leading to compounded issues down the line.

Another team may be picking up the slack, fixing and patching errors before they become visible to higher management or external stakeholders. This creates an imbalance in workload and resource allocation and prevents a holistic view of data system health and efficiency.

The most daunting are the errors that remain undetected – the unknown unknowns. You're not aware of these issues simply because you haven't found them yet, not because they don't exist. This blind spot in your operational oversight can lead to critical failures at the most inopportune moments.

A lack of reported problems doesn't guarantee a smooth sail ahead. It could be the calm before the storm, where underlying issues accumulate to the point of causing significant operational disruptions. Being proactive rather than reactive in seeking out potential data errors can save your team from future blow-ups. The worst thing I have said to a business sponsor is: 'Yes, there was an error. Yes, it was our fault, and the data was wrong for the past year.'

Regardless of current error reports, your business depends on data accuracy for making informed decisions. Even minor errors can have substantial consequences in industries where precision is non-negotiable. Assuming everything is accurate without rigorous verification is a gamble that most businesses cannot afford.

The analogy of an ostrich burying its head in the sand aptly describes some data teams' approach to handling data errors. This metaphor, often misunderstood in the animal kingdom, effectively captures the tendency of individuals or groups to ignore problematic situations, hoping they will resolve themselves or go unnoticed. In the context of data teams, this behavior manifests as a denial or avoidance of acknowledging and addressing data errors, coupled with a [readiness to shift blame onto other teams](#) when issues inevitably surface.



Relying on hope as a data accuracy and integrity strategy is fraught with risks. Stakeholders, from internal teams to external clients, seek confidence in the data they use and depend on. This confidence can only be built through transparency, proactive error detection, and a robust data quality and observability framework. Do you want to drive to work daily, expecting the following significant error to drop?

*Complacency is the enemy of progress. Not hearing about errors is not a sign to rest easy but a call to action for deeper investigation and proactive measures. You will be in trouble if you are not measuring data quality or delivering error rates and SLAs. It's just a matter of time. By embracing a culture of transparency, continuous monitoring, and rigorous testing, data teams can ensure their silence is truly golden, not just the calm before an inevitable storm. Let's shift from **hoping** things are correct to **knowing** they are, building a foundation of confidence and reliability in our data-driven decisions.*

Author: Chris Bergh | Published: Mar 5, 2024

Original article: <https://datakitchen.io/why-not-hearing-about-data-errors-should-worry-your-data-team/>

Relieve yourself from the Burden of Traditional Data Quality Assessments.



DataKitchen

Embracing a fundamentally different approach that prioritizes influence over enforcement, iteration over perfection, and advocacy over bureaucracy. This modern method empowers you to take control of data quality assessments and quickly deliver impressive results.

In most organizations, data quality is everyone's problem—yet paradoxically, it belongs to no one. It is a shared responsibility that often falls into the cracks of accountability. When marketing can't run a campaign, sales can't reach key customers, or finance can't reconcile reports. Fingers point to the data. But where, precisely, should that finger land? In a world where the stewards of data are not the same as its consumers, and where data quality teams lack the formal authority to demand change, traditional data quality assessment methods fall short. What's needed is not a new taxonomy or a better algorithm—it's a fundamentally different approach: one that prioritizes influence over enforcement, iteration over perfection, and advocacy over bureaucracy.

Conventional data quality assessments are built on a deeply flawed assumption: the way to improve quality is through exhaustive analysis. This leads to sprawling, multi-month efforts to categorize and classify issues, often using detailed frameworks derived from academic or regulatory sources. Terms like accuracy, validity, completeness, and consistency become ends rather than means to a business outcome. In the worst cases, teams spend months creating dashboards that no one uses or tracking issues no one fixes.

These methods tend to rely heavily on manual checks, require extensive technical effort (often in SQL or Python), and generate outputs that are hard to act upon. Worse still, they are usually divorced from the actual needs of business stakeholders. A dashboard might show that a column is 90% complete, but the metric is irrelevant if that missing 10% includes the most valuable customers for a campaign. The result is assessments that are technically rigorous but strategically impotent. They measure, but they don't move.

At the heart of the data quality conundrum is a misalignment between those who produce data and those who consume it. Data owners—such as IT, ERP teams, or operational units—optimize for their internal processes. They focus on the data fields that matter to them, and often consider their datasets “good enough” for those purposes. By contrast, data consumers—marketing, sales, finance, analytics—need integrated, comprehensive, and accurate information that spans domains. “Good enough” for the producer is not good enough for the consumer.

Take, for example, a marketing team eager to launch an upsell campaign. They need both street addresses and email addresses to personalize outreach. Yet they discover that half the records have reliable emails, the other half have complete addresses, and only 20% have both. The campaign stalls. The team turns to the data quality lead and says, “Fix it.” However, the lead cannot compel upstream data owners to prioritize the fix, especially if those owners don't experience the pain directly.

This disconnect mirrors the ecological concept of the [“tragedy of the commons,”](#) where shared resources are depleted because individuals act in their short-term interest. In data quality, that shared resource is clean, usable data, and no one wants to take responsibility for fixing problems that benefit others more than themselves. Like shared pastures where every farmer unknowingly overgrazes, shared data environments suffer when individual actors optimize for their

own needs, rather than the collective good. Without clear ownership or incentives, everyone contributes to degradation, and no one takes responsibility for repair.

Faced with this reality, data quality leaders must rethink their role. They are neither compliance officers nor gatekeepers of platonic data ideals. They are advocates. Using their language and metrics, they must campaign for change, build coalitions, and show stakeholders why quality matters.

This is not a theoretical shift; it is a practical one. In this new approach, the data quality assessment becomes a tool of persuasion and influence. It is no longer about documenting problems for posterity. It is about demonstrating how poor data hurts specific business outcomes and presenting a clear, actionable path to improvement. The number of issues identified does not define success, but rather the number resolved.

To support this approach, a new methodology has emerged—one that is small in scope, fast in execution, and laser-focused on stakeholder value. It doesn't begin with a taxonomy—it begins with a problem.

Practical assessments begin where the pain is sharpest. The triggers and motivators are a stalled marketing campaign due to incomplete customer data, a finance compliance risk caused by inaccurate reporting, or a high-priority sales initiative hampered by inconsistent customer information. The data quality leader begins by identifying a narrow slice of data critical to that goal. Rather than inventorying every field in a warehouse, they zoom in on the handful that truly matter.

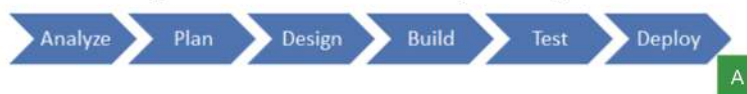
They then profile this data, not to build a catalog, but to understand what's broken meaningfully. Are the email fields malformed? Are the street addresses misaligned with customer IDs? What percentage of records are complete enough to support the campaign? These insights, when shared with business stakeholders, build credibility and urgency.

Most importantly, these tests produce scores, quantifiable metrics that track the quality of specific data elements over time. These scores form the backbone of a quality dashboard that speaks the language of business by presenting data quality in terms of its impact on business goals and outcomes, rather than technical jargon or abstract measures.

Once an initial dashboard is built, the next step is socializing it. This doesn't mean blasting an email. It means sitting down with data owners, showing them the impact of data issues on real business goals, and asking for help. The framing matters. It's not "You broke this," but "We can fix this together, and here's why it matters, and here is all the information you need to make the change."

In this model, the data quality leader becomes a salesperson—or perhaps more charitably, an evangelist. They frame problems as shared opportunities. They seek quick wins. They use each success to build trust and momentum. And they continue to iterate, adding more tests, refining dashboards, and expanding the scope of assessment based on stakeholder feedback. Spending your spare time reading Dale Carnegie's [How to Win Friends and Influence People](#) might help more than [reading ISO 8000](#).

Traditional Data Quality Assessment Waterfall (Months)



DataOps Data Quality Assessment



A = Assessment Delivered



Over time, a virtuous cycle emerges. Stakeholders see the value of improved data. Data owners become more responsive. The quality team is seen not as a nagging force but as a trusted partner. Most importantly, business outcomes improve.

This method works because it is aligned with human nature and organizational dynamics. People are more likely to act when they see a direct benefit, listen when problems are framed in their language, and support efforts that show results quickly.

It also works because it is inherently agile and iterative. Rather than waiting six months for a final report, this method delivers value in days or weeks. It encourages and rewards experimentation and tolerates imperfection. You don't need to fix everything; you just need to fix the right things. It's part of the [DataOps Approach to Data Quality](#).

Crucially, it works at scale. Automated test generation and scorecard dashboards mean that even a small team—or a single data quality advocate—can manage assessments across dozens or hundreds of data sources. The bottleneck shifts from technical labor to human alignment, where the data quality team can add the most value.

How it Works: The Data Quality Assessment and Improvement Process This process is built around a simple yet transformative idea: focus on what matters, deliver value quickly, and build momentum through influence rather than authority. The process begins by anchoring on a specific, urgent need from a real data consumer—someone who cannot achieve their business goal because of poor data. This is not an academic exercise. It's a targeted effort to remove a known blocker for a known person or team. Whether it's a marketing campaign that can't be launched, an incomplete compliance report, or a product analytics dashboard that no one trusts, the starting point is always the same: listen to the pain, and make it your purpose. Start with a specific data consumer's urgent need for quality data. Select a small number of data elements to improve. Identify the technologists and data owners who can effectively change those elements. Gain access and learn the data through data profiling. Develop draft data quality tests that identify specific data quality challenges. Gain alignment between data customers, data owner, and technologist on the problem, the need to improve, and the goals & timeline. Build a data quality score and dashboard for those data elements. Create specific, detailed actions that need to take place with data owners/technologists. Share detailed data quality issue reports. Monitor progress on improvement (and communicate). Improve data quality tests as learning takes place. Iterate until complete. Then start with another tranche of data elements, customers, owners, and technologists. The process repeats—iterating until the original goal is achieved, and then starting again with a new tranche of data elements. Each cycle adds value, builds relationships, and expands the culture of quality. What starts as a targeted fix evolves into a scalable, organization-wide improvement strategy—not through command-and-control, but through continuous, stakeholder-centered action.

This process is built around a simple yet transformative idea: focus on what matters, deliver value quickly, and build momentum through influence rather than authority. The process begins by anchoring on a specific, urgent need from a real data consumer—someone who cannot achieve their business goal because of poor data. This is not an academic exercise. It's a targeted effort to remove a known blocker for a known person or team. Whether it's a marketing

campaign that can't be launched, an incomplete compliance report, or a product analytics dashboard that no one trusts, the starting point is always the same: listen to the pain, and make it your purpose.

Gain alignment between data customers, data owner, and technologist on the problem, the need to improve, and the goals & timeline

The process repeats—iterating until the original goal is achieved, and then starting again with a new tranche of data elements. Each cycle adds value, builds relationships, and expands the culture of quality. What starts as a targeted fix evolves into a scalable, organization-wide improvement strategy—not through command-and-control, but through continuous, stakeholder-centered action.

For too long, data quality work has been mired in paralysis. When caught between technical rigor and business urgency, many assessments fail to gain traction. But the path forward is clear. Start small. Focus on specific business needs. Use tools that automate the grunt work. Most of all, they should act like advocates, not auditors.

Data quality is not just a technical challenge—it's a cultural one. Culture changes not through mandates but through influence. The new approach to data quality assessments puts that influence within reach. It's not about power. It's about persuasion. It's not about frameworks. It's about focus. And it's not about measuring everything. It's about improving what matters most.

[DataOps Data Quality TestGen](#) is uniquely equipped to bring the new, influence-first approach to data quality assessments to life. For data engineers and data quality practitioners tired of manual test creation and bloated governance exercises, TestGen offers a practical, fast, and scalable solution. It is built from the ground up to support the agile, stakeholder-driven methodology. It redefines how data quality assessments can deliver value, starting with small, urgent business needs and expanding based on real-world feedback.

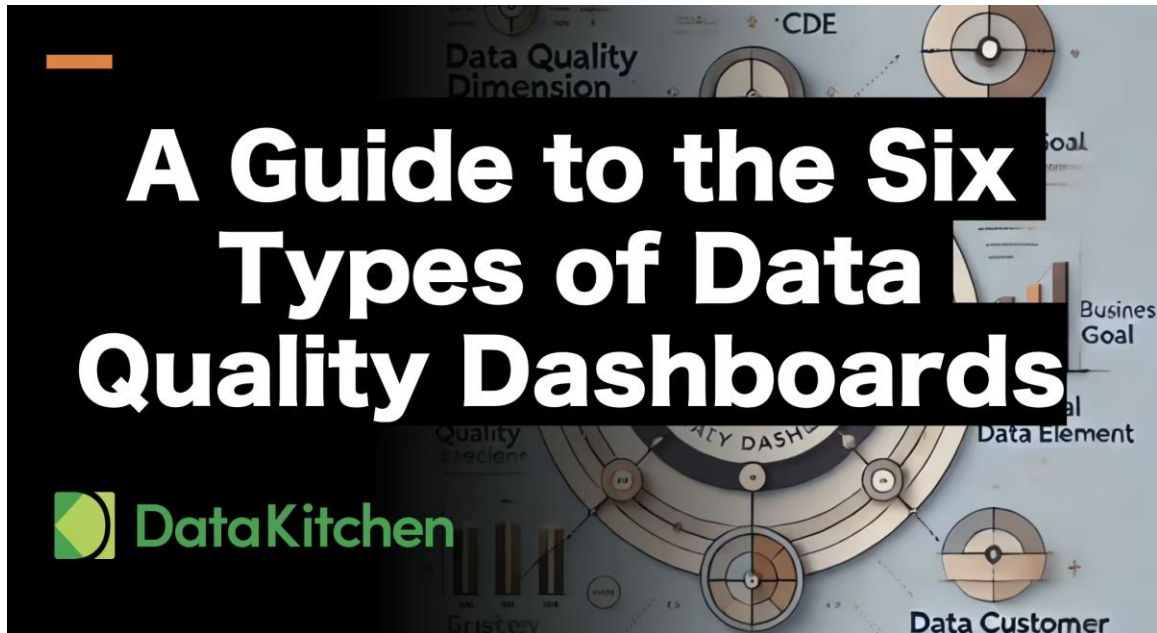
At the core of TestGen is a powerful generative engine that automatically creates dozens of meaningful data quality tests without requiring users to write a single line of SQL or Python. This is not a lightweight toy—it's a full-featured, open-source platform that profiles your data, identifies potential hygiene issues, and proposes comprehensive test suites based on profiling insights. With just a few clicks, you can connect to your database, define a scope of interest, and produce a set of tests that would otherwise take days or weeks to write manually. Within an hour, you can have a live dashboard scoring your data and highlighting exactly where and how quality problems undermine business goals.

Because TestGen is free and open-source, it lowers the barrier to entry for data quality work. You can install it locally and begin making a real impact today—no procurement, red tape, or licensing headaches. And for teams that want to scale their efforts or integrate more deeply into enterprise workflows, there is a path forward with enterprise-grade features, while keeping the core experience lightweight and focused. Whether you're a lone data engineer or part of a quality team embedded in a business unit, TestGen empowers you to act like an influencer: show results quickly, rally support, and iterate fast.

Original article: https://docs.google.com/document/d/1ZKohXVEym_rm1RMZv2QV7_nYJ-u7SVG4TQt-Wb6Eon8/export?format=html

A Guide to the Six Types of Data Quality Dashboards

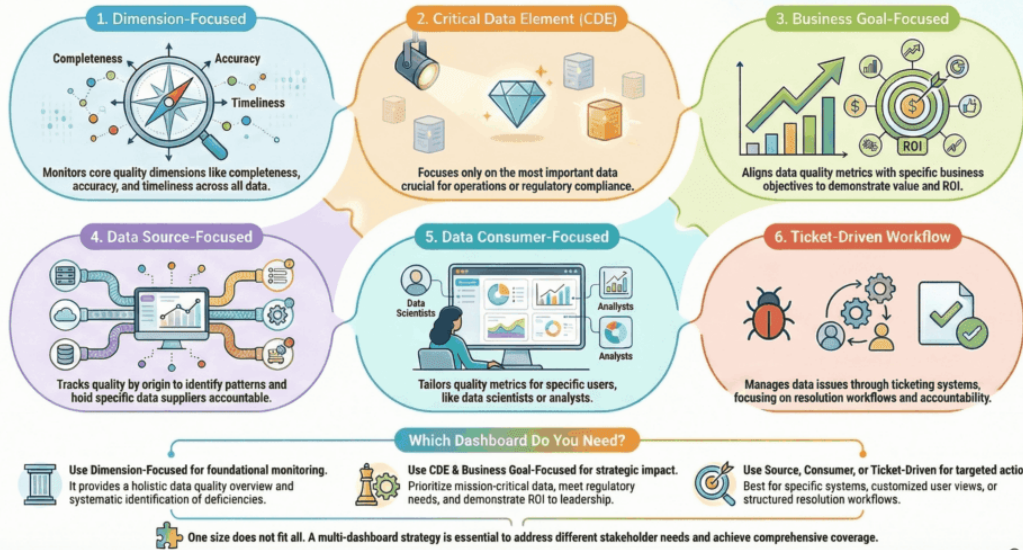
Not all data quality dashboards are created equal. Their design and focus vary significantly depending on an organization's unique goals, challenges, and data landscape. This blog delves into the six distinct types of data quality dashboards, examining how each fulfills a specific role in improving Data Quality.



Poor-quality data can derail operations, misguide strategies, and erode the trust of both customers and stakeholders. Data quality dashboards have emerged as indispensable tools, offering a clear window into the health of their data and enabling targeted, actionable improvements. However, not all data quality dashboards are created equal. Their design and focus vary significantly depending on an organization's unique goals, challenges, and data landscape. This blog delves into the six distinct types of data quality dashboards, examining how each fulfills a specific role in ensuring data excellence. By understanding and selecting the right dashboard, organizations can maximize their influence, streamline efforts, and drive meaningful, sustained improvements in data quality. Read more about how your data quality dashboard can be organized by data quality dimension, Critical Data Element (CDE), business goal, data source, and data customer, or it can be based on ticketed workflows. (updated December 2025)

A Guide to the Six Types of Data Quality Dashboards

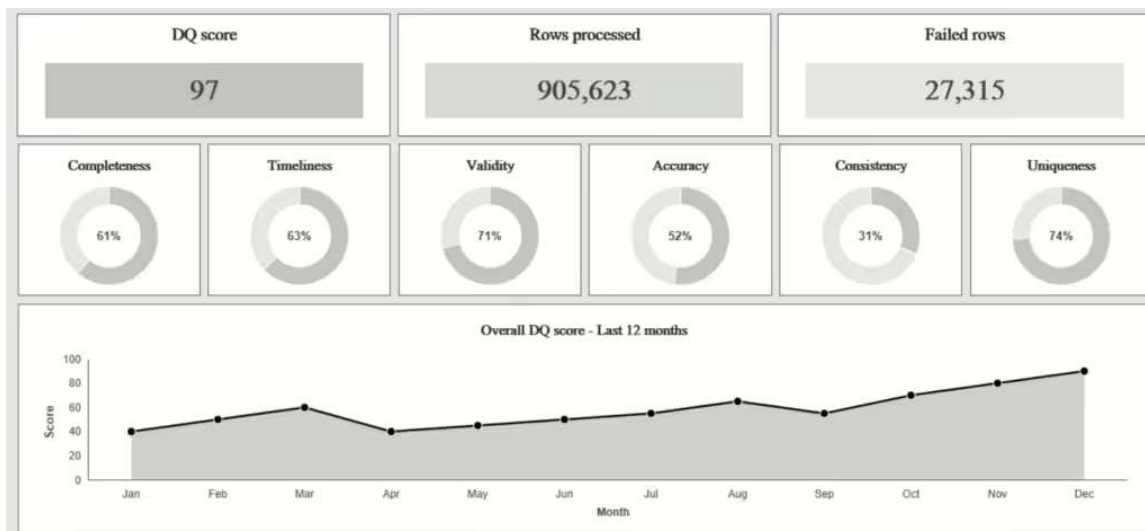
To explain the six distinct types of data quality dashboards and provide clear guidance on when to use each one.



© NotebookLM

The first type of data quality dashboard emphasizes data quality dimensions. Data Quality Dimension-Focused Dashboards are designed to evaluate data through fundamental quality dimensions, such as completeness, accuracy, timeliness, consistency, and uniqueness. These dimensions provide a best practice grouping for assessing data quality. By scoring each dimension individually, these dashboards offer a grouped view of data quality issues, enabling targeted interventions to address gaps.

For example, metrics such as the percentage of missing values help assess completeness, while deviations from authoritative sources gauge accuracy. Timeliness can be assessed by tracking the alignment of data updates with business timelines. These metrics are typically visualized through tools such as heatmaps, pie charts, or bar graphs, making it easy for stakeholders to understand compliance levels across different dimensions. A retail company, for instance, might use such a dashboard to monitor the completeness of customer profiles, ensuring marketing campaigns have enough data—such as email addresses and phone numbers—for effective targeting.



Data quality dimensions are often heralded as foundational principles for evaluating and improving data quality. While these dimensions provide a 'scientific' way to assess data, they can be overly general when applied to real-world data quality problems. While conceptually appealing, this generality can limit its practical utility in driving meaningful improvements in data quality.

The issue lies in the abstraction of these dimensions. For example, a metric like “completeness” is essential to data quality, but defining what “complete” means in a specific context can vary widely. Is completeness about filling every field in a record, or is it about having the fields critical to a particular business process? Similarly, “accuracy” often assumes the existence of an authoritative source for validation, which may not always be available or practical to implement. Without contextual specificity, these dimensions risk becoming check-the-box exercises rather than actionable frameworks that help organizations identify and address the root causes of data quality issues.

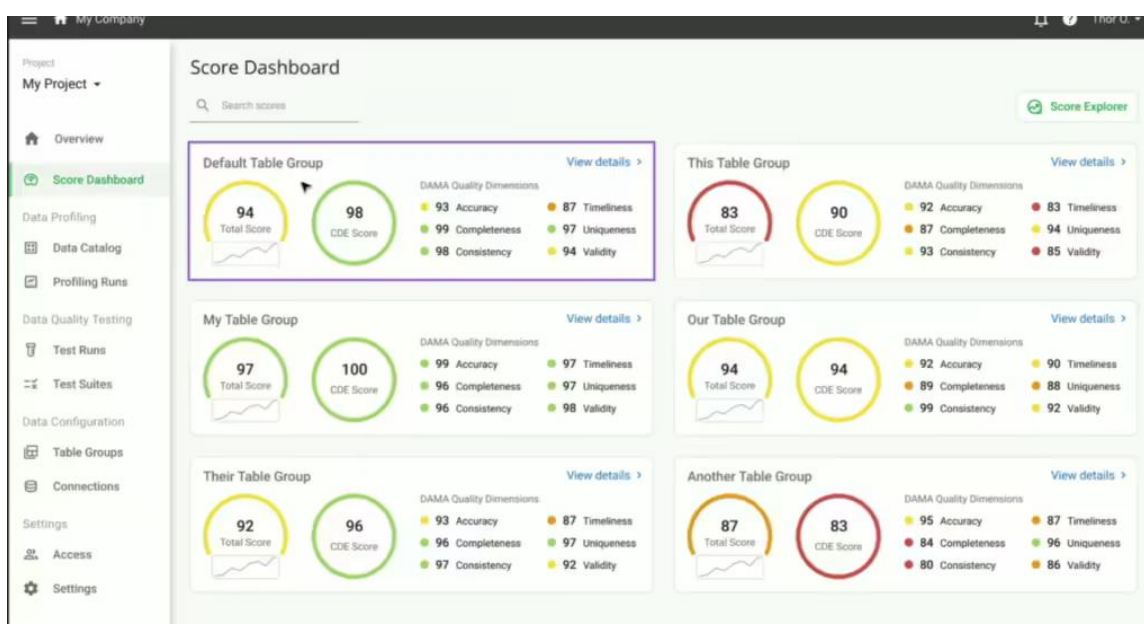
“The DAMA Data Quality Dimension dashboards are crap. They do nothing to motivate people to make improvements to their data.” – Senior Data Quality Person, Financial Services, DataKitchen Market Research 2024

Moreover, relying solely on DAMA dimensions tends to produce dashboards disconnected from an organization’s or its stakeholders’ unique needs. A dashboard filled with generic scores for completeness or consistency may look polished but needs to resonate with the teams responsible for acting on the insights. For example, a business executive might struggle to connect a “70% consistency score” with the operational goals they are trying to achieve. Similarly, data teams might struggle to determine actionable steps if the metrics do not highlight specific datasets, systems, or processes contributing to poor data quality.

While the DAMA dimensions provide a starting point, the key to impactful dashboards is translating these principles into concrete, tailored insights that directly address the organization’s challenges and objectives. By narrowing the scope and aligning metrics with actionable outcomes, organizations can ensure that their dashboards assess data quality and actively contribute to its enhancement.

2. Critical Data Element (CDE) Data Quality Dashboards

Critical Data Element (CDE) dashboards focus on essential data elements crucial for business operations or regulatory compliance. These dashboards prioritize high-impact fields such as financial values, key identifiers like customer IDs, or data required to meet regulatory standards. By narrowing their scope to these critical elements, CDE dashboards enable organizations to allocate resources efficiently and address data quality with the greatest impact. For instance, a bank might use a CDE dashboard to monitor the accuracy of customer credit scores, which are vital for regulatory compliance and effective risk modeling. Another example is an analytics team that wants to focus on the data that goes into the weekly report for the executive team. This targeted approach ensures that limited resources are directed toward improving the data elements that matter most to the organization.



Critical Data Element (CDE)–based data quality dashboards are highly effective in regulated industries such as finance, healthcare, and utilities, where organizations are mandated to monitor and report on the quality of specific data elements. CDEs are often clearly defined by regulatory bodies or industry standards in these industries. For example, a financial institution may need to ensure the accuracy and completeness of loan application data to comply with anti-money laundering (AML) regulations.. The specificity of these mandates makes it easier to identify CDEs and prioritize resources to ensure their quality. CDE-based dashboards in such contexts serve as powerful tools for compliance, risk mitigation, and operational oversight, providing clarity and focus on what truly matters.

“We have had amazing Data Quality Improvement, aligning everyone to improve data based on CDEs. We have corporate goals on CDE data quality. Why? Because we MUST do them for compliance reporting to the government. We focus our Data Quality reporting on CDEs.” – Senior Data Quality Leader, Large US Bank Services, DataKitchen Market Research 2024

However, identifying and gaining consensus on what constitutes a CDE is a significant challenge in industries without strict regulatory frameworks. Unlike regulated industries, where the importance of specific data elements is externally prescribed, non-regulated industries must internally determine which data elements are critical to their success. This process often involves aligning diverse stakeholders—from business leaders to data engineers—who may have varying priorities and definitions of “critical.” For instance, a marketing team might prioritize customer segmentation data, while operations teams focus on inventory data. The lack of predefined criteria can lead to prolonged debates, misalignment, and inconsistent interpretations of what constitutes a CDE.

Another challenge in non-regulated industries is the dynamic nature of business priorities. CDEs may shift as market conditions, organizational goals, or technologies evolve. A retail organization, for example, might initially consider transaction data as critical, only to pivot toward prioritizing clickstream data for e-commerce analytics. This fluidity requires an iterative approach to defining and managing CDEs, which can be resource-intensive and complicated to operationalize within a dashboard framework.

Despite these challenges, CDE-based dashboards remain valuable in non-regulated industries if implemented thoughtfully. Organizations can address these hurdles by establishing transparent governance processes for identifying CDEs, leveraging cross-functional collaboration to ensure alignment, and periodically revisiting and revising the CDE list to reflect changing priorities. While it requires effort and coordination, the result is a targeted data quality approach that delivers tangible business value, even in industries where regulations don’t dictate the terms.

3. Business Goal-Focused Data Quality Dashboards

Business goal-focused data quality dashboards link data quality metrics directly to specific organizational objectives, such as quarterly revenue goals, yearly cost savings targets, or specific revenue-generating marketing and sales programs. These dashboards highlight how data quality influences broader goals, such as customer retention, revenue growth, or regulatory compliance. For leadership teams, this type of dashboard serves as a critical tool to understand the operational and financial implications of data quality issues and to justify investments in remediation efforts. For example, an e-commerce company might use a business goal-focused dashboard to monitor how incomplete customer delivery information affects order fulfillment rates and timing, a crucial metric for customer satisfaction. By tying data quality improvements directly to business outcomes, these dashboards make the case for prioritizing data quality as a strategic initiative.



Business goal-focused data quality dashboards help connect technical data quality efforts to broader organizational objectives. By focusing on outcomes that matter to business stakeholders, these dashboards make data quality improvements more relevant and impactful, fostering cross-functional collaboration between technical teams and business units. This alignment also helps secure executive buy-in and funding for data initiatives by demonstrating the tangible value of addressing data quality issues.

Another significant advantage is that these dashboards provide ROI-driven insights. By highlighting the financial or operational impact of poor-quality data, business goal-focused dashboards allow organizations to prioritize remediation efforts that deliver the most significant value. For example, identifying how incomplete customer delivery information affects fulfillment rates can help an e-commerce company streamline operations, reduce costs, and enhance customer satisfaction.

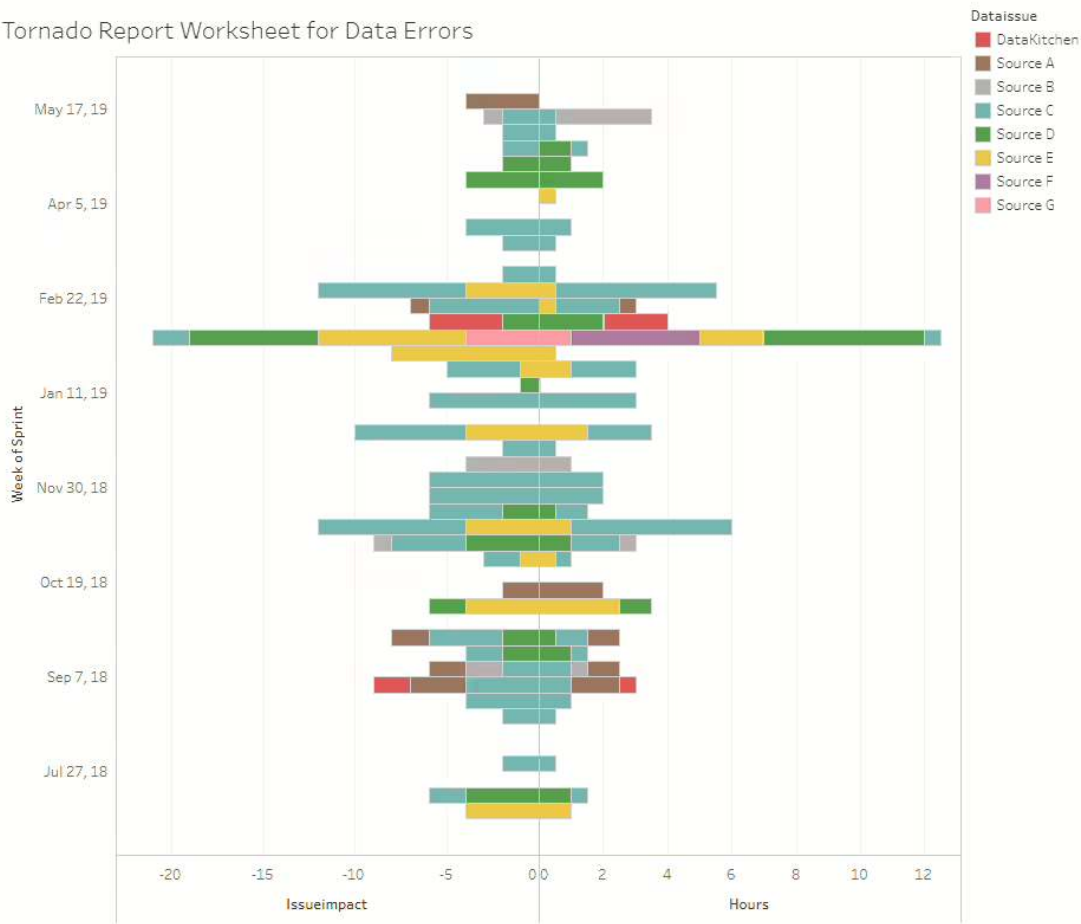
“How do you get the business to take action? By definition, they are inputting data that meets their data quality needs. With all their other problems, why should they care about data quality when providing data to other teams? The only way to motivate them is to pick data elements directly affecting their quarterly business goals. They care about that.” – Data Quality Consultant, DataKitchen Market Research 2024

Despite their advantages, business goal-focused dashboards come with notable challenges. One of the most significant challenges is defining and maintaining alignment between data quality metrics and business goals. Business objectives often evolve due to market dynamics, organizational restructuring, or technological changes, requiring dashboards to be continuously updated to stay relevant. This iterative process demands time, effort, and team collaboration, which can strain resources, especially in organizations with limited data governance capabilities.

4. Data Source-Focused Data Quality Dashboards

A data source-focused dashboard assesses data quality by analyzing its origin, enabling organizations to identify sources that consistently provide low-quality data and take corrective actions. This approach allows enterprises to hold data suppliers accountable or optimize their ingestion processes to ensure higher data integrity. By tracking source-level metrics such as error rates, duplication, and timeliness, these dashboards pinpoint the suppliers or systems responsible for data quality failures, fostering accountability and improvement. This is particularly valuable for organizations managing data from multiple external providers or systems, such as suppliers or third-party APIs. For example, a logistics company might use a source-focused dashboard to monitor the accuracy and timeliness of shipping data from various courier partners, ensuring reliable tracking information for customers and maintaining service quality.

Tornado Report Worksheet for Data Errors



Data source-focused dashboards assign accountability for data quality. Organizations can take targeted action by identifying which suppliers, systems, or processes are contributing to poor-quality data, whether that means renegotiating contracts with external vendors, improving internal workflows, or implementing better validation processes at the point of ingestion. This accountability encourages data providers to prioritize quality.

Another significant benefit is the actionable insights these dashboards provide for optimizing data ingestion processes. Organizations can track each source's error rates and timeliness. This enables them to refine pipelines and ensure data flows into the organization more accurately and reliably. For example, a logistics company might use these dashboards to monitor how courier partners provide shipping data, identify lagging contributors, and take steps to ensure timely updates.

“Our data suppliers don’t know that we exist, don’t care about quality, and will make arbitrary changes to our feeds at any time. It’s up to us to deal with it. We have no control over their data quality” – Data Engineer, Pharmaceuticals, DataKitchen Market Research 2024

Despite their advantages, data source-focused dashboards have notable challenges. One challenge lies in gaining cooperation from external data providers. While organizations can monitor and report on data quality issues, enforcing improvements depends on the willingness and capability of suppliers to address them. They often do not care. How do you motivate them to make a change? These dashboards can become less effective when multiple sources contribute to a single dataset, making it difficult to pinpoint the root cause of data quality issues. For example, if data from various suppliers is aggregated into a central system, determining whether errors originated with a specific supplier or during the integration process requires a root cause analysis.

5. Data Consumer-Focused Data Quality Dashboards

A data consumer-focused dashboard is designed to meet the specific needs of individuals and teams who rely on data to perform their roles, such as data scientists, analysts, or business intelligence professionals. These dashboards prioritize the quality and reliability of datasets and features critical to analytics models, reports, or decision-making tools, ensuring they are fit for purpose. They are often customized to address the unique requirements of different user personas, whether for predictive model inputs or operational reporting. For instance, a healthcare company might use such a dashboard to monitor the accuracy and consistency of patient demographic data used in predictive models for patient readmission. By catering directly to the needs of data consumers, these dashboards help their data customer use their influence to make changes to improve data quality.

One of the primary benefits of this approach is its emphasis on ensuring that the data used by analysts, data scientists, and other consumers is reliable, accurate, and error-free. Tracking the quality of specific datasets or features critical to their projects can give the project owner leverage to improve data quality. Another key advantage is the customization these dashboards provide, tailoring metrics and views to the needs of individual roles or teams. Different data consumers have unique requirements: a business intelligence analyst may need dashboards that track the completeness of reporting data. At the same time, a data scientist might need to focus on model input features such as timeliness or uniqueness. By aligning data quality metrics with the specific needs of end-users, these dashboards foster efficiency and confidence in analytics processes.

“Our data scientists are forever complaining about the quality of data. They have high-value predictions in production. They are motivated to drive improvements to data. We need to focus our data quality reporting on the data inputs to their models, specifically. – Data Quality Consultant, DataKitchen Market Research 2024

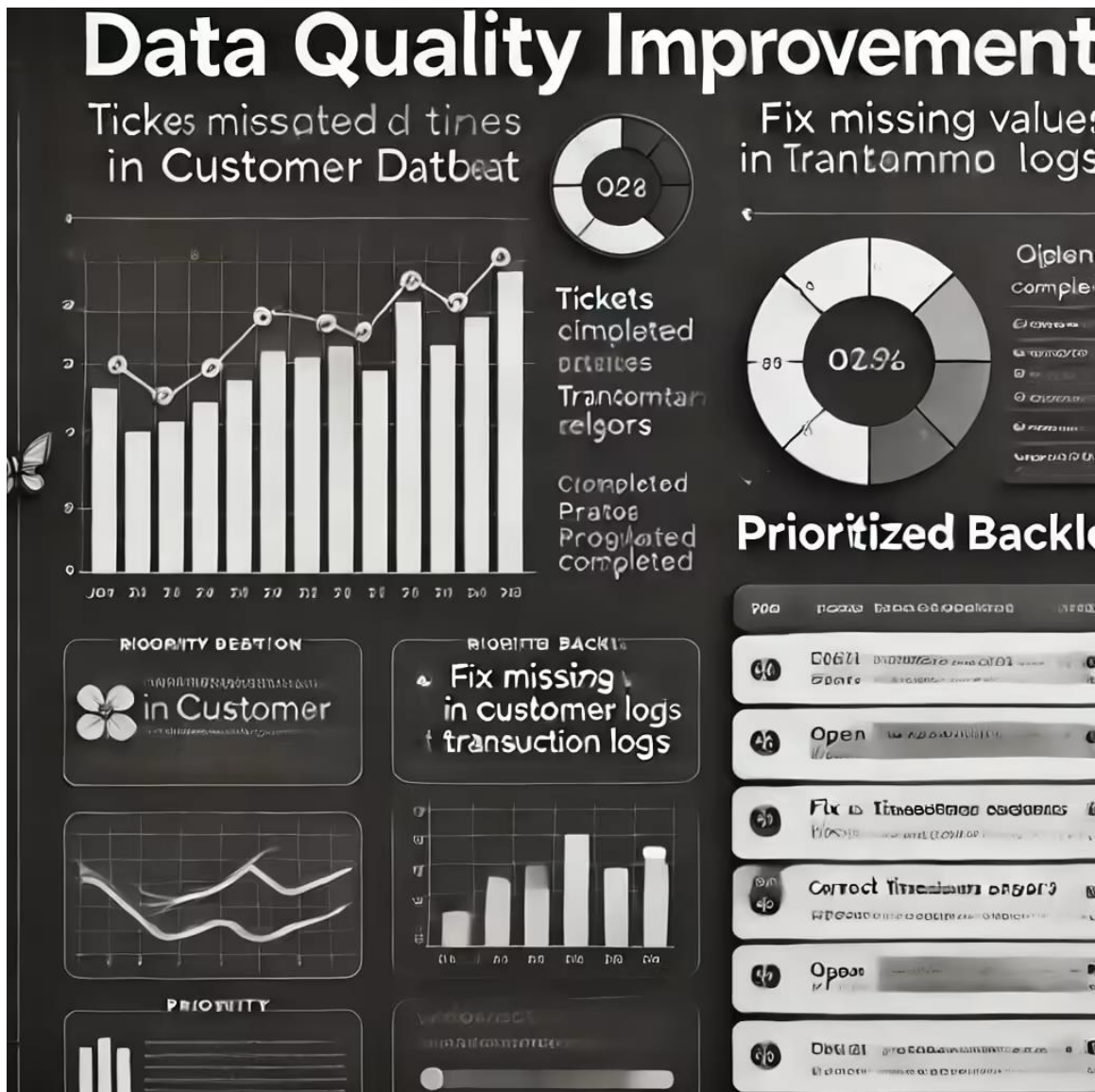
While highly effective, data consumer-focused dashboards present several challenges. One significant hurdle is defining and managing the diverse requirements of different user groups within an organization. Each team or individual may have distinct data needs, leading to a complex and time-intensive process of creating and maintaining customized dashboards. Balancing these needs while ensuring the scalability and manageability of the dashboard system is critical.

6. The Data Quality Ticket Focused Dashboard

Using tickets to drive data quality improvements is another approach to managing and resolving data issues. Tickets are change requests that break down data quality challenges into specific, detailed tasks targeting individual data elements or processes. Each ticket represents a well-defined issue—such as fixing missing values, correcting data inaccuracies, or standardizing inconsistent formats—making it easier for data engineering teams to prioritize, assign, and resolve problems systematically. This approach ensures that data quality initiatives are not abstract or overwhelming but are actionable, trackable, and aligned with organizational goals.

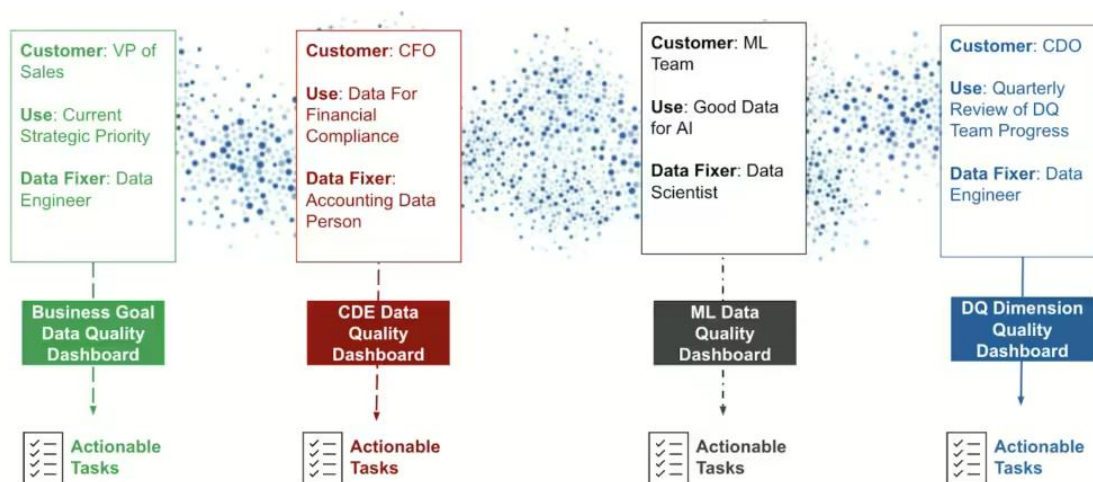
“We don’t use a dashboard per se; we use workflow. It is about the number of tickets created and fixed during a particular period. We have specific goals around tickets, and that works for us.” – Data Quality Leader, Regional Bank, DataKitchen Market Research 2024

In this context, the data quality dashboard shifts its focus from traditional quality metrics to operational progress, visualizing the state of the ticketing process. The dashboard provides two critical views: the count of tickets completed over time, which highlights progress and the team’s capacity to address issues, and a prioritized backlog of pending tickets, ensuring that high-impact problems are addressed first. By tracking ticket completion trends, organizations can measure the efficiency of their data quality efforts and identify bottlenecks in the remediation process. This ticket-driven model fosters accountability and aligns with agile principles, enabling continuous iteration and improvement in data quality over time.



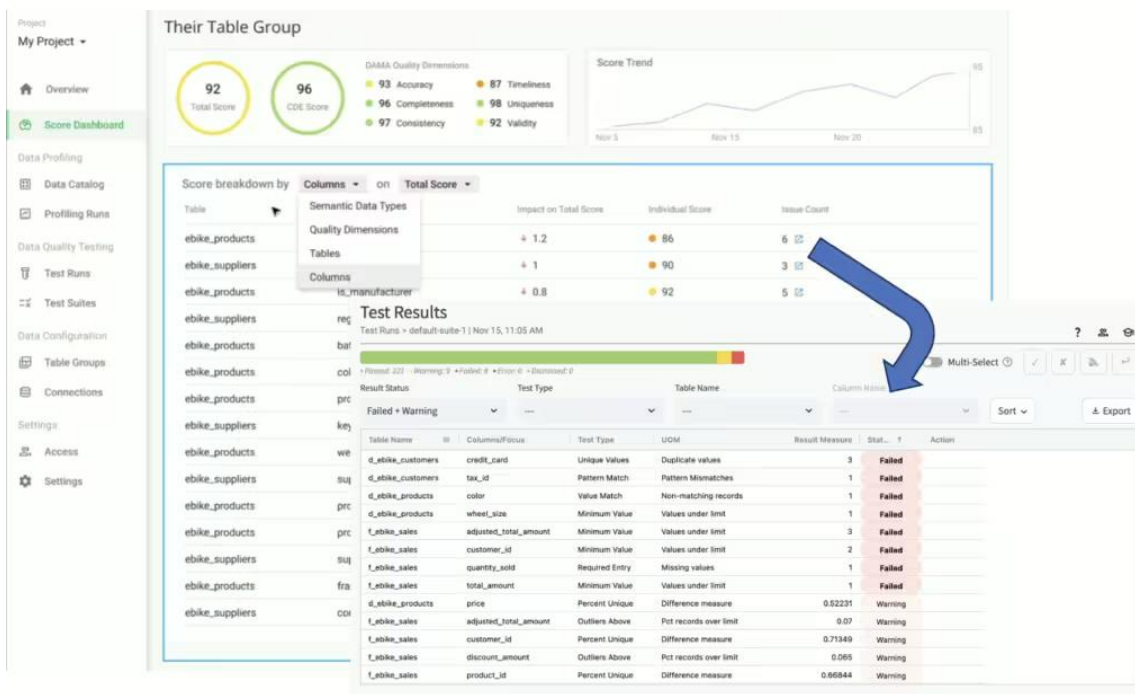
Why You Need a Multi-Dashboard Approach: Maximizing Your Influence

Adopting a multi-dashboard approach to data quality is essential for organizations aiming to understand their data landscape comprehensively. Each type of dashboard offers a unique lens through which to view and address data quality issues, and combining these perspectives allows for a more nuanced and effective strategy. By leveraging multiple dashboards, organizations can ensure that no aspect of data quality is overlooked, that priorities are aligned with business goals, and that all stakeholders have access to the data they need to succeed.



Dimension-focused dashboards are foundational in ensuring that every critical aspect of data quality—completeness, accuracy, timeliness, consistency, and uniqueness—is monitored and addressed. These dashboards provide a holistic overview of data quality, enabling organizations to identify and remediate deficiencies systematically. However, this broad approach can be complemented by Critical Data Element (CDE) dashboards, which prioritize the data elements that are most crucial for operational efficiency or regulatory compliance. These dashboards balance addressing general quality metrics and focusing on high-priority data elements.

Business goal-focused dashboards add another layer of alignment by tying data quality efforts directly to organizational objectives. These dashboards ensure that data quality initiatives are not just technical exercises but are integral to achieving key business outcomes such as customer retention, revenue growth, or compliance. At the same time, data source-focused dashboards bring accountability into the equation by identifying where data quality issues originate, whether from internal systems or external suppliers, and streamlining processes to mitigate risks at their source.

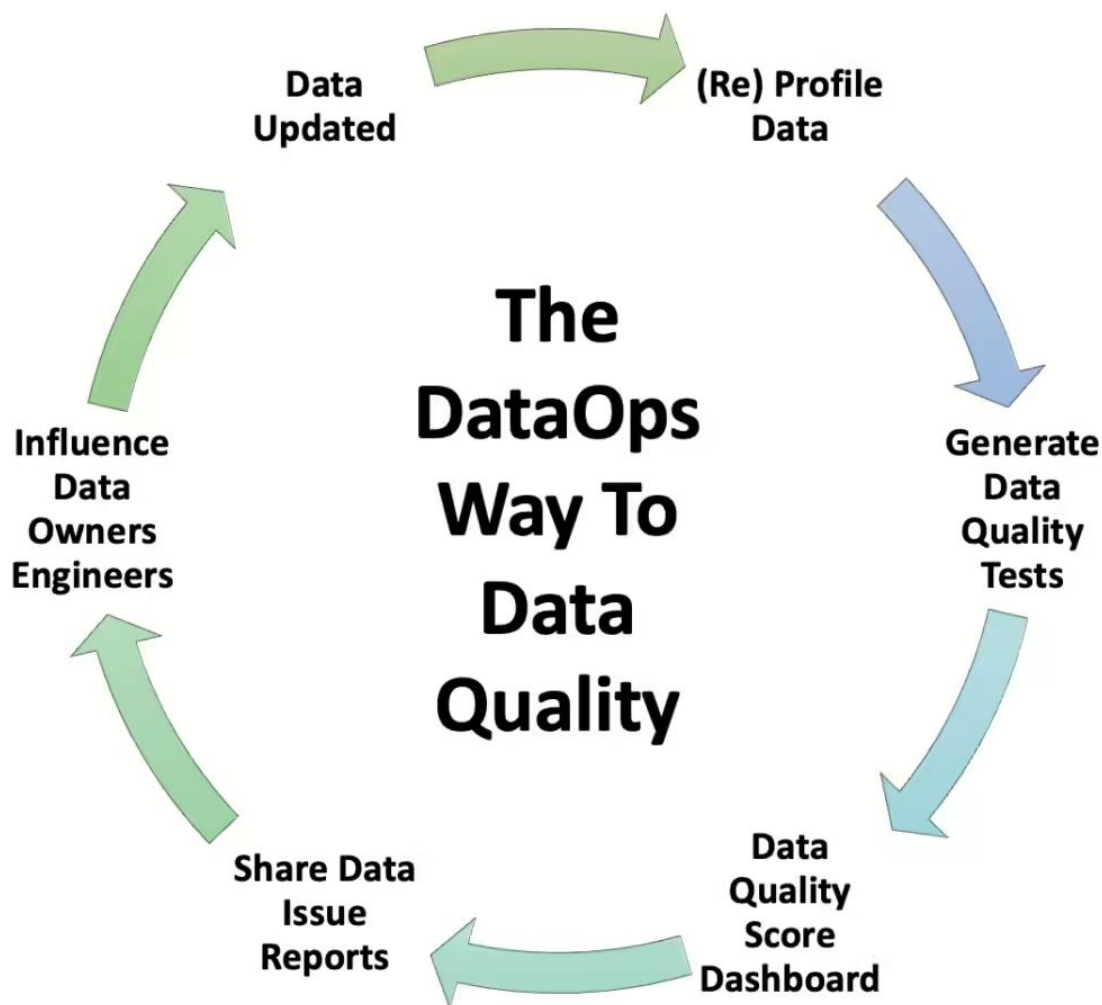


Finally, data consumer-focused dashboards ensure that end-users—data scientists, analysts, and business teams—can access reliable and fit-for-purpose data for their needs. Organizations can enhance decision-making, reduce

inefficiencies, and build trust in their data ecosystem by catering to these distinct roles. These six dashboards create a comprehensive framework that empowers organizations to address data quality while targeting specific pain points.

Best Practices for Building Effective Data Quality Dashboards

The first step is to define clear objectives for each dashboard. Understanding its specific purpose—whether to monitor compliance, support operational goals, or provide reliable insights to data consumers—helps tailor metrics to align with organizational priorities. A well-defined objective ensures that dashboards remain focused and deliver value to stakeholders without overwhelming them with irrelevant or extraneous data.



Automation is another critical element in maximizing the effectiveness of data quality dashboards. Fast updates ensure that dashboards remain responsive to the latest data conditions, reducing the need for manual intervention and improving decision-making speed. Automation also enhances scalability, making maintaining multiple dashboards across an organization easier without excessive overhead. By leveraging modern tools and frameworks like [DataKitchen's DataOps Data Quality TestGen](#), teams can ensure their dashboards remain dynamic and relevant, even as data sources and business requirements change.

Accessibility is critical to ensuring that dashboards are effective for technical and non-technical stakeholders. Dashboards should be intuitive, with user-friendly visualizations and interfaces catering to end-users' diverse needs. Clear, actionable metrics and insights allow executives, data scientists, and operations teams to understand and act on data quality issues without requiring extensive technical expertise. Dashboards must be paired with processes to

act on their insights to influence data quality outcomes. Dashboards alone are diagnostic tools—they highlight problems but cannot solve them. Organizations should integrate their dashboards with fast, actionable, motivating. ‘Here fix this’ ticket creation and workflows that enable rapid remediation of issues. This action-oriented approach ensures that data quality issues identified by dashboards are fixed.

DataOps Data Quality TestGen


TestGen Test Issue Report

Test	Minimum Value: All values at or above set minimum Tests that the minimum value present in the column is no lower than the minimum value in baseline data		Failed
Measured Value	3		
Threshold Value	0	Expected count of values under limit	
Test Run Date	Nov 15, 11:05 AM EST	Table Group	default
Database/Schema	demo	Test Suite	default-suite-1
Table	f_ebike_sales	Data Quality Dimension	Validity
Column	adjusted_total_amount	Disposition	No Decision

[View on TestGen >](#)

Usage Notes

This test is appropriate for a cumulative dataset only, assuming all prior values are still present. It is also appropriate for any measure that has an absolute, definable minimum value, or a heuristic that makes sense for valid data.

Result History

Test Date	Threshold Value	Measure Value	Status
Nov 15, 11:05 AM EST	0.0	3.0	Failed
Nov 15, 11:03 AM EST	0.0	3.0	Failed
Nov 15, 11:01 AM EST	0.0	3.0	Failed
Nov 15, 10:59 AM EST	0.0	0.0	Passed

Sample Data

Adjusted Total Amount	Difference From Baseline
102.1	-66.28
161.51	-6.87
164.95	-3.43

SQL Query

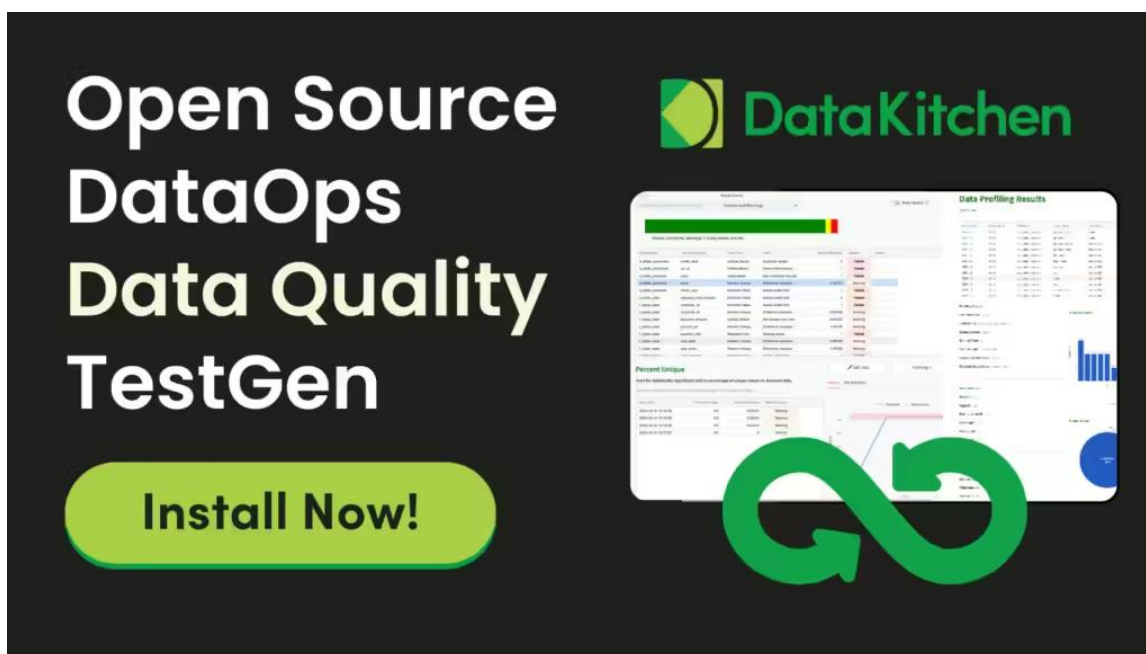
```
SELECT DISTINCT "adjusted_total_amount", (ABS("adjusted_total_amount") - ABS(168.38)) AS
difference_from_baseline FROM demo.f_ebike_sales WHERE "adjusted_total_amount" < 168.38 LIMIT 500;
```

Iteration and continuous improvement are essential to the long-term success of data quality dashboards. As organizational needs evolve, so must the dashboards and the metrics they track. A [DataOps approach to data quality](#) is needed, emphasizing collaboration, automation, and iterative development. DataOps enables teams to innovate on dashboards and metrics, ensuring they remain aligned with changing priorities and drive sustained improvements. By iterating and innovating, organizations can maximize their influence on stakeholders and foster a culture of continuous data quality enhancement.

Conclusion

Data quality dashboards are indispensable tools for managing and improving data quality. By leveraging the six dashboard types—dimension-focused, CDE, business goal-focused, data source-focused, data consumer-focused, and task list—organizations can take a strategic, targeted approach to address their unique data quality challenges. The goal is to achieve agile data quality at scale: automate data quality tests, make informed trade-offs, iterate quickly to maximize your influence, and deliver actionable improvements to those who can make change—fast and focused.

With the right dashboards, your organization can move confidently toward its data-driven goals and ensure that decisions are built on a solid foundation of high-quality data. And you, as a data quality leader, can gain influence and success in your organization!



The blog explains that there is no single “data quality dashboard.” Instead, dashboards vary widely because different stakeholders need different views of data quality. The article outlines six types of dashboards—KPI dashboards, data element dashboards, business goal dashboards, source-oriented dashboards, consumer-oriented dashboards, and operational dashboards—and shows how each reflects a different purpose, audience, and definition of “quality.” The central message: the best dashboard is the one aligned to the user’s goals, not the one with the most metrics.

The article does not offer downloadable templates, sample files, or prebuilt dashboards. What it does provide is a framework of six dashboard types, showing how they differ, why they differ, and which stakeholders each serves. The focus is conceptual, not template-based.

Real-world inspirations include:• Quality KPI dashboards showing completeness/accuracy metrics (e.g., missing values % or timeliness trends).• Operational monitoring dashboards that highlight failing tables or datasets.• Supplier quality dashboards tracking data source health across systems.

To make these more data-quality–focused, swap business metrics for key quality checks (completeness, validity, consistency, timeliness, uniqueness).

You can build data quality dashboards using TestGen directly on Databricks. If not, then build a data quality dashboard in Databricks using its SQL and visualization tools: In Databricks SQL, create queries to calculate data quality KPIs (e.g., percent nulls, schema changes, failed checks). Then surface those results in a Databricks dashboard that automatically refreshes as data pipelines update. External posts (e.g., Medium) show how to visualize data quality expectations (success/failure rates) using charts in Databricks.

Dashboard Examples & Templates• Power BI sample dashboards & templates from Microsoft Learn and community galleries (.pbix/.pbit files).• Online free dashboard template collections (e.g., Windsor.ai, community galleries) that you can adapt for data quality.

GitHub Projects• Sample dashboard repositories you can download and customize for data quality metrics.

Blogs & Tutorials• Community blogs and Medium posts on building data quality dashboards (e.g., with Databricks or other platforms) offer step-by-step guidance.

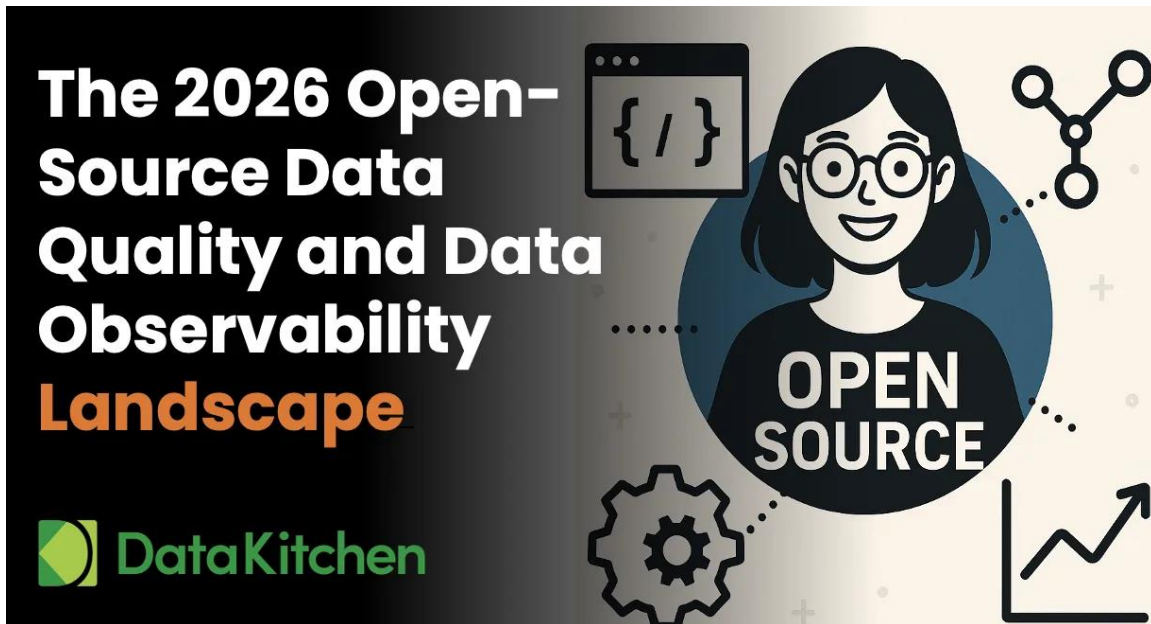
Watch our 2025 webinar at: <https://info.datakitchen.io/webinar-2025-12-video-form-masterclass-in-the-six-types-of-data-quality-dashboards>.

Author: Chris Bergh | Published: Nov 27, 2024

Original article: <https://datakitchen.io/the-six-types-of-data-quality-dashboards/>

The 2026 Open-Source Data Quality and Data Observability Landscape

BlogCase StudiesWhite PapersWebinars / PodcastsDocumentation



Your LLM just told the CEO that revenue is up 40% when it's actually down. Your analytic engineers are vibe coding late into the night. Your predictive models—once your pride and joy—are degrading faster than you can debug them. And your standard reports—no one trusts them. Welcome to 2026, where bad data doesn't just break dashboards anymore—it weaponizes AI against your business.

The terrifying truth is that AI amplifies data quality and data observability failures exponentially. A single schema drift that once meant a broken report now means thousands of incorrect predictions per second. That missing data validation you postponed? It just trained your model to be confidently wrong at scale. Your data engineers are in full panic mode, manually spot-checking tables while AI models consume data faster than any human can validate it. The executives who demanded “AI transformation” are now demanding answers for why their million-dollar models produce nonsense. And those expensive observability platforms that promised to solve everything? They're just telling you what broke after your AI has already made 10,000 bad decisions.

The cruel irony of the AI revolution is that it demands perfect data quality at the exact moment when data volumes, sources, and complexity have made quality impossible to achieve through traditional means. Modern data pipelines feed voracious AI systems that retrain hourly, consume from hundreds of sources, and make decisions in milliseconds—all while your team is still writing SQL tests like it's 2015. The old guard of enterprise solutions wants six figures to tell you what you already know: your data is broken. But in the age of DataOps, where deployment cycles are measured in hours, not months, you need tools that move at AI speed and don't require selling a kidney to afford.

We'll explore the new generation of solutions that use AI to police AI, automate test generation at scale, and provide the transparency and control that proprietary platforms can't match—all while keeping your CFO happy. Because in 2026, the question isn't whether you need data quality for AI; it's whether you'll solve it before bad data destroys everything you've built. This guide explores that landscape. It defines the categories, compares the tools, and explains how teams are combining them—often with DataOps practices in the age of AI—to create truly reliable, end-to-end systems.

Five years ago, most data teams relied on hope. They hoped the source systems behaved, that joins stayed aligned, and that nobody silently changed a schema. Testing, when it existed, was manual and reactive.

Then came the first wave of structured testing frameworks, such as Great Expectations, which brought the hand-writing (or configuration) of data quality test discipline from software into analytics. That was the beginning of a shift: from “data should work” to “let’s prove it does.”

By 2020, the conversation expanded to data observability—the continuous monitoring of freshness, volume, schema, and anomalies in production. Closed-source tools, such as Monte Carlo, Databand, and Acceldata, have made data observability mainstream.

Now, in 2025, the open-source community has accelerated this evolution. Projects like Soda Core, Elementary Data, and DBT Tests — alongside DataKitchen’s own DataOps Data QualityTestGen and DataOps Observability tools — are democratizing capabilities once locked behind expensive platforms. The convergence of data quality testing and data observability is creating a new expectation: reliable, transparent data pipelines built the same way modern software systems ensure reliability.

While open-source adoption brings its own challenges including fragmented tools, uneven test coverage, and limited governance layers, DataKitchen’s open-source DataOps tools addresses these precise pain points by unifying tests, monitoring, and [scorecards](#) under one consistent process, transforming the typical open-source complexity into an enterprise-ready solution that maintains all the flexibility and transparency benefits while adding the structure and governance that production data pipelines demand.

Mapping the Open-Source Data Quality and Observability Landscape

Open-source tools in this space generally fall into two categories: data quality software, data observability software.

Open Source Data Quality Tool	Description	Comment	Open Source – Command Line / DSL	Open Source – UI	Auto-Generate Data Quality Tests
Great Expectations — github.com/great-expectations/great_expectations	A Python-based framework to define “expectations” about your data (validations/tests/assertions), run them, document outcomes, and monitor data quality over time.	Remains the most mature option, offering a declarative YAML syntax and a strong validation library, though it can be heavy to maintain	yes	no	no
Soda Core — github.com/sodadata/soda-core	Open-source CLI + Python library for data quality testing: using SodaCL (checks language) to scan datasets for missing/invalid/unexpected values.	Really, a DSL (domain-specific language) to write data quality tests.	yes	no	no
Deequ — github.com/aws-labs/deequ	Scala library (Spark-based) by AWS Labs: define constraints/tests on DataFrames, compute metrics, detect anomalies/constraints violations at scale.	Very good for big-data / Spark environments; less so for lighter warehousing without Spark.	yes	no	no

DataQualityDashboard — github.com/OHDSI/DataQualityDashboard	<i>Focused on observational health data (OMOP CDM) but generalizable: runs systematic data quality checks (completeness, consistency) via defined check types.</i>	<i>DataQualityDashboard is an R package</i>	yes	yes	no
DQOps — github.com/dqops/dqops	<i>Data quality monitoring platform: ~150 built-in table/column checks, dashboards, incident grouping, notifications for freshness/timeliness.</i>	<i>Their Open Source version is functionally limited to a few tables—more like a demo of their enterprise product.</i>	yes	yes	no
SQLFluff — github.com/sqlfluff/sqlfluff	<i>A dialect-flexible SQL linter and auto-formatter designed for ELT/DBT codebases; supports templating (Jinja, dbt), multi-dialect SQL.</i>	<i>Linting of SQL is NOT data quality, more like code quality</i>	yes	no	no
CSVPath — www.csvpath.org	<i>A data file feeds a quality management framework centered around data identification, validation, and descriptive and lineage metadata production</i>	<i>A new player with a unique twist and a narrow focus on enterprise file feeds.</i>	yes	yes	no
DataOps Data Quality TestGen — https://github.com/DataKitchen/dataops-testgen	<i>DataOps Data Quality TestGen delivers simple, fast data quality test generation and execution by data profiling, new dataset hygiene review, AI generation of data quality validation tests, ongoing testing of data refreshes, & continuous anomaly monitoring</i>	<i>Data Engineers don't have time to manually write data quality tests; it's too slow. Get full data test coverage for all your tables and columns with one click.</i>	yes	yes	yes

A Python-based framework to define “expectations” about your data (validations/tests/assertions), run them, document outcomes, and monitor data quality over time.

Remains the most mature option, offering a declarative YAML syntax and a strong validation library, though it can be heavy to maintain

Open-source CLI + Python library for data quality testing: using SodaCL (checks language) to scan datasets for missing/invalid/unexpected values.

Scala library (Spark-based) by AWS Labs: define constraints/tests on DataFrames, compute metrics, detect anomalies/constraints violations at scale.

Focused on observational health data (OMOP CDM) but generalizable: runs systematic data quality checks (completeness, consistency) via defined check types.

Data quality monitoring platform: ~150 built-in table/column checks, dashboards, incident grouping, notifications for freshness/timeliness.

A dialect-flexible SQL linter and auto-formatter designed for ELT/DBT codebases; supports templating (Jinja, dbt), multi-dialect SQL.

A data file feeds a quality management framework centered around data identification, validation, and descriptive and lineage metadata production

Data Engineers don't have time to manually write data quality tests; it's too slow. Get full data test coverage for all your tables and columns with one click.

Open Source Data Observability Tool	Description	Comment	Open Source – Command Line / DSL	Open Source – UI	Auto-Generate Data Quality Tests
Elementary — github.com/elementary-data/elementary	“dbt-native” data observability solution: set up quickly, monitor pipelines, detect issues, send simple alerts; open-source package + managed offering.	Good for modern stack (dbt + warehouse) angle; lighter weight.	yes	no	no
OpenMetadata — github.com/open-metadata/OpenMetadata	A unified metadata platform: supports data discovery, lineage, governance, and includes data quality testing/test-suites, metrics, and dashboards.	Strong “observability + governance” angle; growing quickly. Good for enterprise contexts.	yes	no	no
ODD Platform — github.com/opendatadiscovery/odd-platform	Open-source data discovery and observability platform: includes a data quality dashboard, integration with DQ frameworks, and a metadata graph.	Less mature than some, but aligns with “observability” story vs purely DQ.	yes	no	no
Prometheus — github.com/prometheus/prometheus	Time-series monitoring and alerting system for metrics collection and analysis.	Very mature; only partial data observability (logs/metrics), no task, subtasks, no schedules, no process lineage	yes	no	no
Grafana — github.com/grafana/grafana	A visualization and dashboard platform for metrics/logs/traces integration. Used with Prometheus.	Extremely mature; default UI layer for IT observability stacks.	no	yes	no

DataOps Observability – https://github.com/DataKitchen/data-observability-installer	DataOps Observability monitors every data journey — from data source to customer value — from any team development environment to production, across every tool, team, environment, and customer, so that problems are detected, localized, and understood immediately.	Locate the root causes, understand the impact, and prevent future problems. Provides a dashboard to illustrate improvements in delivery speed and quality.	yes	yes	no (integrates with DataOps TestGen)
---	---	--	-----	-----	---

“dbt-native” data observability solution: set up quickly, monitor pipelines, detect issues, send simple alerts; open-source package + managed offering.

A unified metadata platform: supports data discovery, lineage, governance, and includes data quality testing/test-suites, metrics, and dashboards.

Open-source data discovery and observability platform: includes a data quality dashboard, integration with DQ frameworks, and a metadata graph.

Very mature; only partial data observability (logs/metrics), no task, subtasks, no schedules, no process lineage

Locate the root causes, understand the impact, and prevent future problems. Provides a dashboard to illustrate improvements in delivery speed and quality.

AI moves fast—dangerously fast—and its insatiable appetite for data has fundamentally broken traditional data quality approaches. While data engineers manually write test cases one SQL query at a time, bad data floods into models at unprecedented volumes, poisoning predictions, breaking deployments, and destroying stakeholder trust before anyone notices the damage. The math is brutal: modern AI pipelines require [comprehensive test coverage](#)—da2 tests per column, 3 tests per table—across hundreds or thousands of tables, updated continuously as schemas evolve and data patterns shift. Manual testing simply cannot scale to this reality; by the time you’ve written tests for table 10, tables 1 through 9 have already changed, and your LLMs are confidently hallucinating on corrupted data, amplifying minor quality issues into catastrophic business decisions.

The Future of Open-Source Data Reliability: DataOps

Looking ahead, automation and AI will push the boundary even further. Machine-learning models can already generate tests, detect anomalies, and predict failures. In the near future, AI-generated tests will expand coverage far beyond what humans can author manually. At the same time, [shift-left testing—embedding quality checks early in development—and shift-down development testing](#)—connecting quality results to leadership dashboards—will merge into a single continuous feedback loop.

The organizations that master this loop will no longer “check quality after the fact.” They will design for trust from the start. Open-source frameworks stitched together with the DataOps discipline form the foundation for that transformation.

Conclusion

The open-source revolution in data quality and data observability is not coming—it is already here. Engineers now have access to transparent, community-driven tools that can match or exceed proprietary systems. What remains is

coordination: linking those tools into a living, automated process that ensures every dataset is tested, monitored, and trusted.

*Looking for Commercial Data Quality or Data Observability Software? Read The 2026 Landscape:
<https://datakitchen.io/the-2026-data-quality-and-data-observability-commercial-software-landscape/>*

That data journey layer is what DataKitchen provides through its open-source DataOps toolkit. By combining automated test generation, continuous observability, and easy-to-read scorecards, DataKitchen helps data teams move from green pipelines to truly trusted data.

DataOps Data Quality TestGen stands out in the crowded field of data observability solutions by delivering enterprise-grade capabilities without enterprise-level costs. While other platforms charge premium prices for essential features, TestGen provides comprehensive data observability monitoring—including freshness, volume, schema, and drift detection—as part of its core open-source offering. This democratizes access to critical data quality tools that every organization needs but not every budget can accommodate. The platform's one-click test generation creates full coverage across all tables and columns, eliminating the tedious manual work that typically consumes valuable engineering hours. Rather than spending months building custom data quality dashboards from scratch, teams can configure professional-grade monitoring views in minutes, dramatically accelerating time-to-value.

What truly sets TestGen apart is its sustainable business model and transparent pricing structure. As a single-user open-source solution with an enterprise version at just \$100 per user per connection with unlimited data and events, it offers predictable costs that scale reasonably with your organization. This pricing philosophy comes from a profitable company with deep DataOps expertise that understands the importance of stable, reliable partnerships. Unlike venture-backed competitors who may suddenly pivot pricing models or sunset features, TestGen provides the confidence of working with a mature organization committed to customer success and consistent value delivery.

For data teams tired of choosing between incomplete coverage and overwhelming costs, TestGen delivers on its promise: all the checkmarks, none of the typical cost burden. By automating test creation, accelerating root cause analysis, and preventing embarrassing data errors before they reach stakeholders, it transforms data quality from a resource-intensive burden into a streamlined, confidence-building practice. The result is higher test coverage at lower cost—a combination that makes TestGen the superior choice for organizations serious about data quality without the serious price tag.

Author: Chris Bergh | Published: Oct 28, 2025

Original article: <https://datakitchen.io/the-2026-open-source-data-quality-and-data-observability-landscape/>

The 2026 Data Quality and Data Observability Commercial Software Landscape

With 50+ vendors to choose from, data quality and data observability software has never been more powerful, more plentiful, or more confusing—until now.



Let's be honest: the data quality and observability market has become a jungle—and it's consolidating fast. Datadog swallowed Metaplane to beef up its data observability play. Snowflake acquired Select Star. The message is clear: data quality and observability have become too strategic to ignore, and the big players are buying their way in. Meanwhile, between legacy enterprise giants bolting on new features, venture-backed startups promising AI-powered everything, and open-source projects gaining serious traction, choosing the right tool feels less like software selection and more like survival of the fittest.

The good news? You've never had more options. The bad news? You've never had more options. Whether you're a data engineer drowning in pipeline failures, a governance lead trying to prove ROI on data initiatives, or a CDO wondering why your dashboards still can't be trusted, this comprehensive 2026 vendor landscape will help you cut through the noise—from the Informaticas and Monte Carlos of the world to scrappy open-source alternatives that punch well above their weight.

Modern Data Observability & Data Quality Commercial Software **Catalog / Governance Platforms With Data Quality Features** **Specialist Contact / Reference Data Quality Vendors**

Primarily focused on customer/contact data, addresses, and identity, but still very much “data quality software”:

So where does this leave you? The data quality and observability market isn't going to get simpler anytime soon—expect more acquisitions, more feature overlap, and more vendors claiming to do everything. But here's the thing: the best tool isn't the one with the most features or the slickiest demo. It's the one your team will actually use. Start by getting ruthlessly clear on your real problem. Are you fighting fires from broken pipelines? Trying to build trust with business stakeholders? Meeting regulatory requirements? Then evaluate tools against that specific pain, not a generic checklist. Consider whether you need enterprise hand-holding or if your team can run with open-source. Think hard about vendor lock-in and what happens when that hot startup gets acquired or pivots. And remember: the fanciest

observability platform in the world won't save you if your data architecture is a mess to begin with. Tools are force multipliers—they amplify good practices, but they can't replace them.

Author: Chris Bergh | Published: Dec 1, 2025

Original article: <https://datakitchen.io/the-2026-data-quality-and-data-observability-commercial-software-landscape/>

No Python, No SQL Templates, No YAML: Why Your Open Source Data Quality Tool Should Generate 80% Of Your Data Quality Tests Automatically

The reality is that 80% of data quality tests can be generated automatically, eliminating the need for tedious manual coding. Learn how to do it today.



As a data engineer, ensuring data quality is both essential and overwhelming. The sheer volume of tables, the complexity of the data usage, and the volume of work make manual test writing an impossible task to get done. Everyone wants to write more tests, yet they somehow never get it done. Every customer we talk to has a considerable test debt. Current open-source frameworks like YAML-based Soda Core, Python-based Great Expectations, and dbt SQL are frameworks to help speed up the creation of data quality tests. They are all in the realm of software, domain-specific language to help you write data quality tests.

The data engineer's job is to ensure reliable, high-quality data pipelines that fuel analytics, machine learning, and operational use cases. But there's a growing problem—data quality testing is becoming an unsustainable burden. You don't have the time to write exhaustive data quality tests, and even if you did, you lack full context on business data usage. Meanwhile, business data stewards don't have the skills to program or even tweak these tests. The reality is that **80% of data quality tests can be generated automatically**, eliminating the need for tedious manual coding.

The solution? An open-source AI-driven data quality testing that learns from your data automatically while providing a simple UI, not a code-specific DSL, to review, improve, and manage your data quality test estate—a Test Generator.

The Challenge of Writing Manual Data Quality Testing

Organizations often have hundreds or thousands of tables. Writing comprehensive data quality tests across all datasets is **too costly and time-consuming**. Even if data engineers had the resources, they **lacked the full context of data use**. Business data stewards who understand data semantics should co-manage these tests. However, they do not have the programming skills required to define them in code.

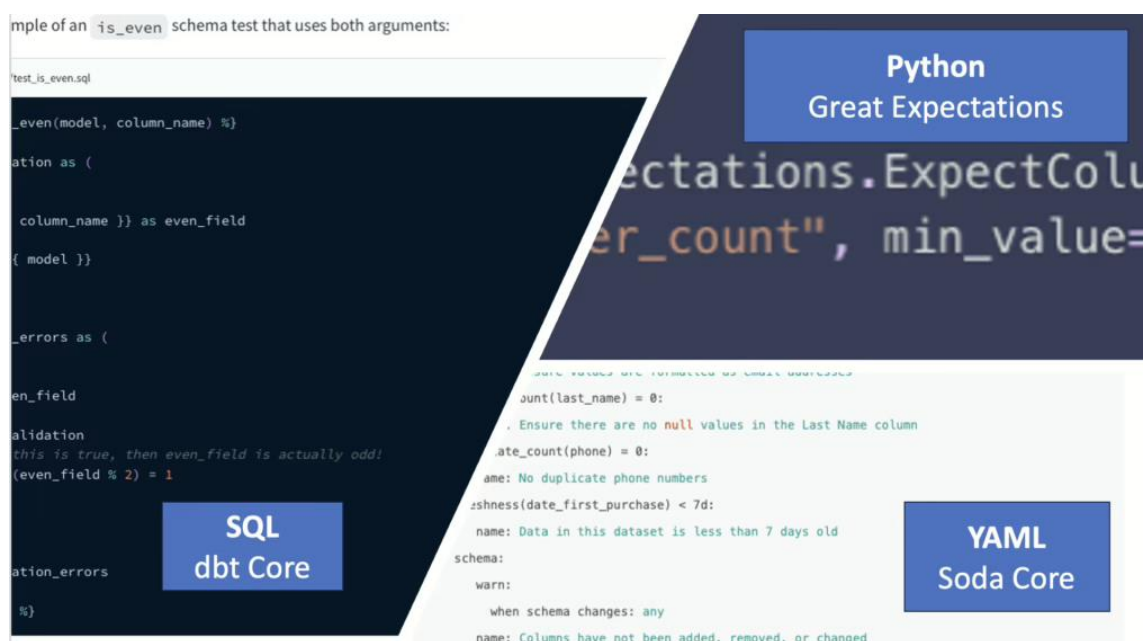
Data quality tests are critical to ensuring correct and trustworthy data but have significant challenges. One of the biggest hurdles is the sheer volume of tables in modern data environments. Organizations today manage hundreds or even thousands of tables, making it prohibitively expensive and time-consuming to write data quality tests manually for

each one. Even with frameworks like dbt tests, Great Expectations, or Soda Core, creating and managing custom tests at such a scale quickly becomes overwhelming.

Another challenge is the lack of full context surrounding the data. As a data engineer, you are responsible for managing pipelines, but you may not have deep domain expertise on how the data is used in the business. On the other hand, business users often have a better understanding of data semantics but lack the technical skills to write tests. This disconnect makes it difficult to ensure that data quality tests are comprehensive and relevant. Ideally, data quality tests should be co-managed by data engineers and business users, but current tooling does not support this collaboration effectively.

Furthermore, data quality tests serve multiple vital purposes beyond just catching issues in production. They are crucial for data quality scorecards, which help track the long-term health of an organization's data. These scorecards are essential for regulatory compliance, executive reporting, and decision-making. Manually writing tests limits the scope of what gets tested and can introduce biases, making it difficult to get a complete picture of data quality. Organizations that fail to prioritize data quality testing risk compromising their data integrity, affecting their ability to make informed business decisions.

Current leading open-source frameworks like YAML-based Soda Core, Python-based Great Expectations, and dbt SQL are frameworks to help speed up the creation of data quality tests. They are all in the realm of software, domain-specific language to help you write data quality tests.



The AI-Driven Solution To Data Quality Testing: You Can Get 80% Of The Way There Fast

AI-driven data quality testing presents a transformative shift in how organizations maintain the integrity of their data. Traditional methods rely heavily on manual rule-writing, requiring significant engineering effort. However, AI can analyze historical data, learn its natural constraints, and automatically generate tests for critical aspects such as schema drift detection, data volume fluctuations, missing values, null anomalies, referential integrity, and business logic validation based on past trends. These tests can be generated instantly with a single button, enabling data engineers to focus on ensuring pipeline reliability rather than spending time on custom validation rule development. AI-driven automation allows organizations to start fast and scale their data quality efforts effortlessly.

In The Land Of The Blind, The Data Engineer Who Has Data Quality Testing In Production Is King

Data engineers [experience burnout at alarming rates](#), with many considering leaving the industry or their current company within the following year. Surveys consistently reveal that the most significant sources of burnout include

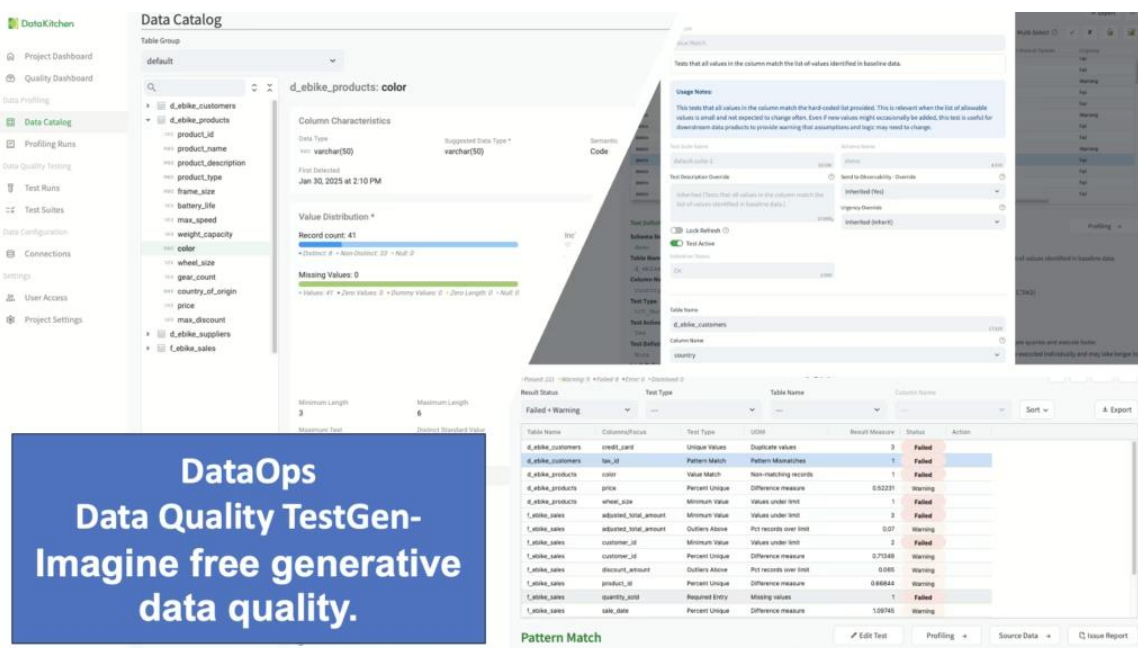
spending too much time fixing errors, repetitive manual tasks related to data preparation, and an endless stream of often unrealistic requests from colleagues. Data quality testing, unfortunately, is almost always treated as an afterthought—an overlooked necessity that only becomes urgent when something breaks. But testing is the gift that engineers give to their future selves. By proactively implementing automated and AI-driven testing, engineers can reduce their manual workload, prevent errors before they occur, and establish a more sustainable workflow. And let's not forget—most bosses are already asking about AI-driven solutions. Why not take charge of bringing AI to data quality testing and lighten your load?

Test Are A Shared Artifact: Business and Governance Users Need a UI, Not Code

Data quality is not merely a technical concern but a business imperative. Business users, who play a crucial role in defining and managing data governance, should be able to participate in data quality testing without the need for programming expertise. Writing SQL, Python, or YAML-based rules should not be a prerequisite for their involvement. Instead, a simple user interface should empower them to review and approve AI-generated tests, define additional rules using natural language inputs, and collaborate with engineers to refine data quality standards. Organizations can bridge the gap between business and technical teams by providing a UI-driven approach, ensuring a collaborative and more efficient data quality management process.

The Remaining 20% Of Domain-Specific Custom Tests Should Be Where You Focus Your Time.

Although AI can generate most data quality tests, some industry-specific validations require a level of customization that AI alone may not capture. These tests, however, should not be buried in complex code repositories that make them inaccessible or difficult to maintain. Instead, organizations should maintain a library of reusable test patterns, such as historical balance tests, transaction anomaly detection, and time-series trend analysis. Businesses can apply these custom tests flexibly across multiple datasets without reinventing validation logic for each use case by treating these custom tests as structured templates rather than hardcoded rules. This approach ensures that critical domain-specific data quality checks remain scalable and manageable while allowing organizations to benefit from automation wherever possible. This is where you can add real value to the business instead of just being a data plumber.



DataOps
Data Quality TestGen-
Imagine free generative
data quality.

Table Name	Column/Measure	Test Type	Result Measure	Status	Action
d_ebike_customers	credit_card	Unique Values	Duplicate values	3	Failed
d_ebike_customers	tax_id	Pattern Match	Pattern Mismatches	1	Failed
d_ebike_products	color	Value Match	Non-matching records	1	Failed
d_ebike_products	price	Percent Unique	Difference measure	0.9229	Warning
d_ebike_products	wheel_size	Minimum Value	Values under limit	1	Failed
d_ebike_sales	adjusted_total_amount	Minimum Value	Values under limit	3	Failed
d_ebike_sales	adjusted_total_amount	Outliers Above	Pct records over limit	0.07	Warning
d_ebike_sales	customer_id	Minimum Value	Values under limit	2	Failed
d_ebike_sales	customer_id	Percent Unique	Difference measure	0.71048	Warning
d_ebike_sales	discount_amount	Outliers Above	Pct records over limit	0.005	Warning
d_ebike_sales	product_id	Percent Unique	Difference measure	0.86844	Warning
d_ebike_sales	quantity_sold	Maximum Value	Missing values	1	Failed
d_ebike_sales	tax_id	Percent Unique	Difference measure	1.00746	Warning

The Future: AI-Powered, Generative Data Quality Management as a Standard Practice

AI is revolutionizing data management; data quality testing should be no exception. The current approach of manual rule-writing is unsustainable at scale, and the cost of poor data quality is too high to ignore. By leveraging AI:

*The next generation of data quality frameworks must be built around **AI-powered automation** and **human-in-the-loop validation**. The time has come to move beyond manual test writing and embrace intelligent, self-learning data quality monitoring.*

A robust AI-driven data quality platform eliminates the pain of manual test writing while providing governance teams with visibility and control. The future of data quality is automation-first, with AI-generated tests forming the baseline and domain-specific rules captured as reusable templates.

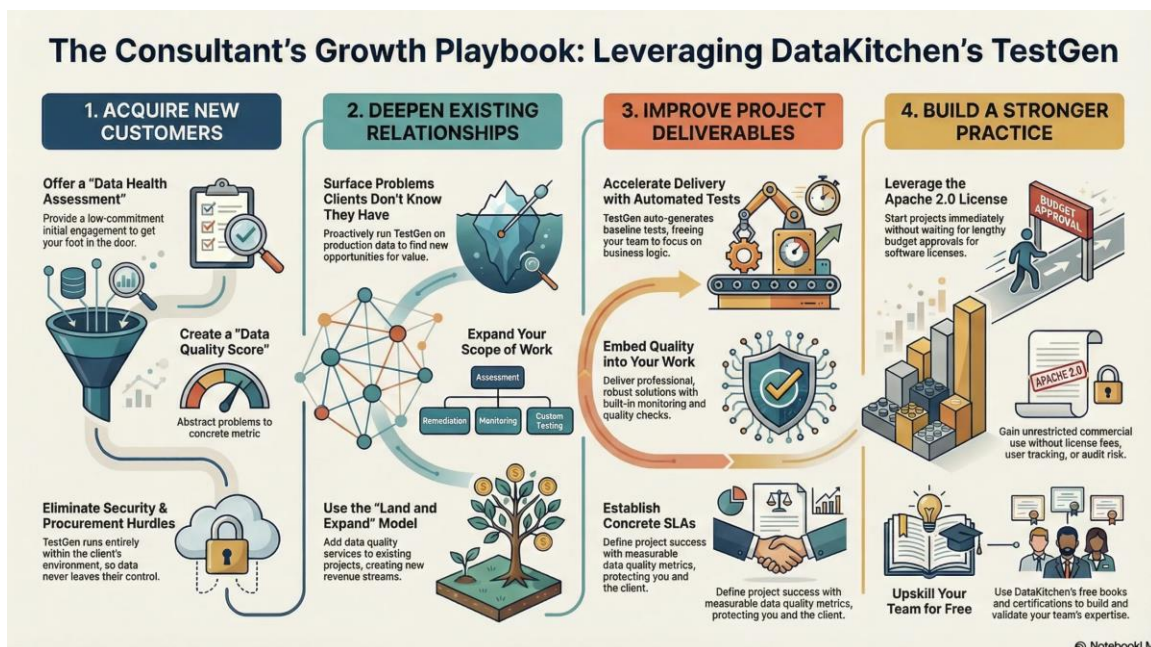
*The time has come to **move beyond manual test writing** and embrace intelligent, self-learning data quality monitoring that seamlessly integrates business and engineering needs.*

Author: Chris Bergh | Published: Feb 17, 2025

Original article: <https://datakitchen.io/no-python-no-sql-templates-no-yaml/>

The Data Consultant's Growth Playbook: Accelerating Client Acquisition and Retention with DataKitchen's Open Source TestGen

BlogCase StudiesWhite PapersWebinars / PodcastsDocumentation



If you've worked in data and analytics consulting, you know the hardest part isn't always the technical side. The real challenge is finding new clients, growing existing accounts, and proving your value to justify your fees.

The tools you choose make a difference. They impact the quality of your work and your ability to win and grow business. Expensive enterprise data quality platforms can be a barrier. Clients often hesitate at high licensing costs before seeing any value, and recommending pricey tools to budget-conscious organizations can stall deals for months.

DataKitchen's open-source data quality and observability tools, especially [DataOps Data Quality TestGen](#), give you a strong advantage. You can show your team's value and highlight the need for your services without asking clients to commit to a budget right away. By connecting to their data and uncovering hidden issues, you can start conversations that lead to real projects.

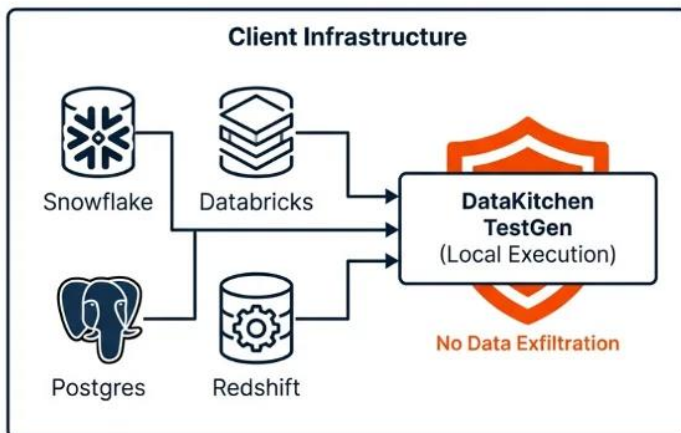
Goal 1: Acquire New Customers

Every consultant faces the same question: how do you get started with a prospect who doesn't know you? Cold emails and presentations only go so far. What really helps is showing you understand their specific problems, sometimes even better than they do. TestGen gives you a clear way to do this.

One approach that works is to offer prospects a simple "data health assessment" to start. Present it as a quick exercise where you connect TestGen to a sample of their data and create a detailed data quality report.

The mechanics are straightforward. TestGen connects to major database platforms (including Snowflake, Databricks, PostgreSQL, SQL Server, and Redshift). Once connected, it automatically profiles the data and generates a baseline set of data hygiene metrics and a data quality score without requiring custom logic. Within hours, you have concrete findings.

The 'Discovery Engagement' uses a low-risk Data Health Assessment to secure new business.



The Approach:

A 2-3 day high-impact exercise.

Process: Connect -> Auto-Profile
-> Generate Baseline Tests.

The Security Advantage:

Runs entirely in-house. Data never leaves the client's infrastructure.

Sidesteps CISO objections and lengthy security reviews.

TestGen is especially good for client assessments because it runs entirely in-house. You set it up in the client's environment, so their data never leaves their systems. This removes security worries that can slow down discussions about data tools. If a CISO asks where the data goes, you can say it stays put. TestGen checks the data in place and saves results locally. For consultants, this means you can suggest running TestGen on production data without long security reviews or procurement delays.

Since TestGen works on unlimited tables, you aren't limited in what you can check. Unlike commercial tools that charge per table or data source, you can profile and score all of a client's data if needed. This allows you to give a complete assessment and show the full range of data quality issues across the organization.

You can turn TestGen's results into a [data quality score](#)—a single number that gives executives something clear to consider. Most organizations don't know how to measure their data quality. They might say it's "messy" or that some reports "can't be trusted," but they don't have a consistent way to track it.

Converting technical hygiene issues into a 'Data Quality Score' creates executive urgency.

The Solution: A Synthesizable Metric

Executives lack a systematic way to measure "messy data."

This Report Card moves the discussion from abstract services to concrete evidence of a problem.

"You're presenting evidence of a problem and positioning yourself as the person who can fix it."

Data Quality Report Card			67/100
DATA QUALITY BREAKDOWN			
Data Asset	Grade	Issue / Status	
Customer_Master	D →	34% Null Values	
Sales_Transactions	B ✓	Key Integrity Valid	
Inventory_Log	C ⚠	Orphaned Records Detected	

DataKitchen
Data Health Assessment

If you tell a VP of Analytics that their customer data scores 67 out of 100, and show which tables and columns are lowering that score, you change the conversation. You're not just selling consulting services—you're showing real evidence of a problem and positioning yourself as the one who can solve it.

The assessment itself is a small task, but it reliably leads to larger conversations. Once you've shown a client their data quality issues, the natural next question is: what do we do about it? This opens doors to remediation work, data pipeline improvements, governance implementations, custom test development, and ongoing data quality monitoring arrangements.

Goal 2: Deepen an Existing Customer Relationship

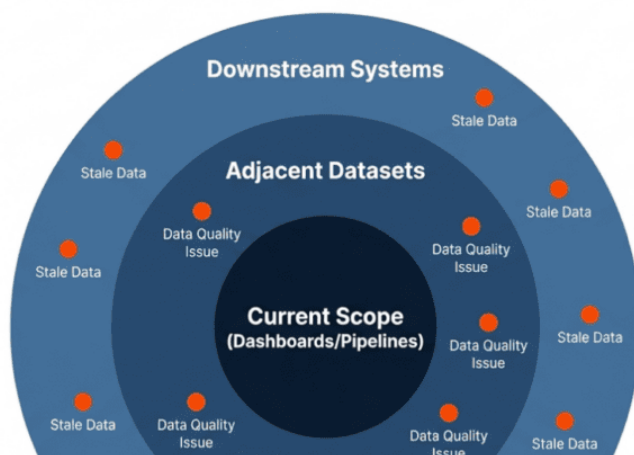
It's usually easier to grow a business with current clients rather than to find new ones. They already trust you, know how you work, and often have a budget set aside for your services. The key is to find new problems to solve and show value beyond what you're already doing.

Data quality work is often a missed opportunity in accounts where you're already handling analytics, pipeline development, or data science projects.

Even clients you know well may have data quality issues you haven't found yet. This happens because data problems often stay hidden until something breaks later on. Teams usually focus on building new features instead of checking existing data, and no one wants to point out problems without offering solutions.

TestGen helps you spot issues early and come prepared with both the problem and a solution. Try running it on data sources you haven't checked before. You can profile, test, and score datasets from other teams or systems that feed into your pipelines. You'll likely find data hygiene issues worth talking about.

Existing clients have invisible data problems that provide opportunities to expand scope.



Strategy: Proactive Diagnosis

- Run TestGen against data sources not currently in scope.
- Execute read-only queries against live production data, not sanitized staging copies.

Insight:

- Data quality issues are often transient or cumulative. Snapshots miss them.
- Identifies hygiene issues worth discussing without waiting for a system failure.

TestGen is built to run safely on production data, so you can use it on live systems. There's no need to make sample datasets or use old copies. The tool runs read-only queries on the real data your client relies on, giving you results that reflect the current situation, not just a cleaned-up sample.

TestGen can also monitor data for unusual changes over time. It automatically spots shifts in frequency, volume, schema, and other patterns. You can set up TestGen on a database, let it generate alerts for a week or a month, and then share those alerts with your client. For example, you might ask, "Did you know this data was late? Did you notice the schema changed? Are you missing data here?"

First, there's fixing the problems. When you find [data quality issues](#), someone has to address them. This could mean updating source system checks, adding logic to pipelines, or working with other teams to solve the root causes. Since you found the issues, you're in a good spot to lead this work.

Third, there's custom testing. While TestGen creates automated tests, clients often need custom tests for their specific business rules. For example, a retail client might want to check that inventory counts match physical audits, while a financial services client might need tests for regulatory metrics. Building these custom tests shows your deep understanding of their business. TestGen aims to cover about 80% of data quality tests automatically, but the remaining 20%—the unique ones—are often the most valuable and where your team's expertise really matters.

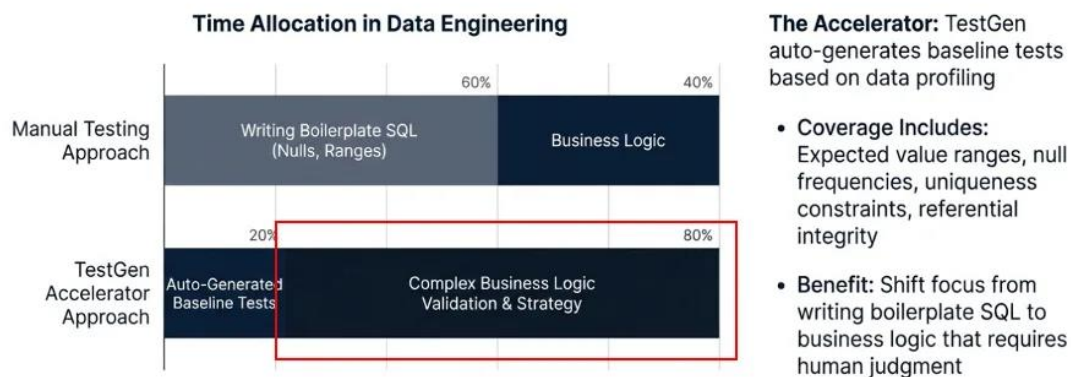
This approach is powerful because it adds value without taking away from existing work. You're not fighting for the same budget as other projects. Instead, you're finding new problems and creating new value. A client who hired you for dashboards might now want data quality help. A client who uses you for pipelines might add observability services. Each new skill opens up more opportunities.

Goal 3: Improve Project Deliverables

Beyond helping you win business, TestGen improves the quality of your technical work. In a competitive market, the quality of your deliverables sets you apart.

Writing tests is one of the most time-consuming parts of building production data pipelines. Sound engineers know that untested code is risky, but the push to deliver features often delays testing. This can lead to systems that work—until they suddenly don't, often at the worst times. TestGen automatically generates a basic set of data tests based on profiling. When you connect it to a data source, it analyzes the actual data and generates tests for expected value ranges, missing-value frequency, uniqueness, referential integrity, and more.

Automated accelerators free up consultant time for complex business logic



Automation test creation doesn't replace careful, domain-specific testing. Instead, it takes care of the basic tests, so you can focus on checking business logic that needs your expertise. The result is [better test coverage](#), delivered more quickly.

Since TestGen can run on unlimited tables with no licensing limits, you can create complete test suites across all your data platforms. Commercial tools that charge per table often limit test coverage. With TestGen, you can test everything—staging tables, transformations, final outputs—without worrying about extra costs for your client.

When you deliver a data pipeline or analytics solution, what sets professional work apart is how robust it is. Amateur work fixes the immediate problem. Professional work solves the problem and also plans for possible future issues.

Adding automated data quality tests and observability tools to your deliverables shows this level of professionalism. When you hand over a pipeline that checks its own health, alerts on problems, and shows data quality metrics, you give your client something they can trust to run smoothly.

Move clients from vague expectations to concrete Service Level Agreements (SLAs).

Professional Delivery = Self-Monitoring Pipelines.

The SLA Definition: What does “Working” mean?

- ✓ Data arrives within the time window.
- ✓ Row counts fall within expected ranges.
- ✓ Quality scores remain > 95.
- ✓ Key metrics do not deviate beyond thresholds.

The Shift:

From Amateur Delivery (“It works until it breaks”) to Professional Delivery (A pipeline that monitors its own health).

Outcome:

Shared expectations protect both the consultant and the client.

When clients are choosing between consultants, showing advanced testing and observability can set you apart. Many consultants only focus on features. By highlighting quality, reliability, and long-term success, you show you’re a partner who thinks beyond just the first delivery.

Goal 4: Improve Staff Skills

If you run a consulting practice with multiple team members, the skills of your staff directly impact your capacity to win and deliver work. Investing in training pays dividends through better delivery, higher client satisfaction, and the ability to command premium rates.

DataKitchen has published detailed, free books on [DataOps](#) and [DataOps change management](#). These resources give your team a solid foundation—not just in using the tools, but in understanding the bigger ideas behind good data operations. For consultants, this education helps in several ways. It builds technical skills, gives you a common language for talking about data challenges with clients, and shows that your team understands industry best practices.



Download

GitHub: DataKitchen
TestGen & DataOps
Observability
(Apache 2.0)



Learn

Free DataOps Books
& Methodology
Guides



Validate

DataOps & Data
Quality Certification
Programs

In addition to books, DataKitchen also offers free online certification programs in [DataOps](#) and [data quality](#). These certifications give your team a clear learning path and help them fully understand key concepts. Certifications are essential for consulting teams. They show outside proof of your team’s skills, give clients confidence in your expertise,

and set a clear standard for what “competent” means in your organization. This makes it easier to assess and grow your talent.

Using these free resources can become part of your company culture. New hires can complete certifications during onboarding. Teams can discuss book concepts in meetings, and delivery teams can apply best practices from these resources to client projects. Investing in skills pays off over time. Teams that keep learning and improving deliver better results, make clients happier, and earn more referrals and repeat business.

Enterprise software sales cycles are often long and unpredictable. If you recommend a tool that is a big investment, you add extra steps that you can’t control. Budget approvals can take months, procurement adds complexity, and other priorities can stop deals altogether.

TestGen’s Apache-2.0 license is ideal for consulting projects. It allows commercial use with no restrictions, lets you modify it without sharing changes, and you can distribute it without paying royalties. For clients, this means they can use TestGen across their whole organization, add it to their products, and build on it without legal headaches.

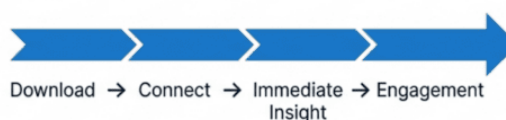
Open source eliminates the friction of procurement to accelerate time-to-value.

Traditional Enterprise Sales Model



Pain Point: Clients balk at six-figure price tags before value is demonstrated.
Result: Stalled deals.

The DataKitchen Open Source Model



Benefit: The only investment is consulting time.
Economics: Unlimited tables, no ‘per-table’ tax.

For consultants, this means you can recommend and use TestGen without worrying about license compliance. There’s no user counting, no usage tracking, and no risk of audits. You can build your own features on top, create custom add-ons for clients, and make it part of your regular toolkit.

A major economic benefit of TestGen is that it supports an unlimited number of tables. Most commercial data quality tools charge by table, data source, or data volume. These pricing models force you to choose between full coverage and saving money. They are too expensive.

With TestGen, you don’t have to make that tradeoff. You can profile and test every table in a client’s data warehouse at no extra cost. This lets you build truly comprehensive data quality programs rather than cutting corners due to budget constraints.

Because TestGen runs entirely in-house, it solves a common concern with data tools. Many organizations, especially in regulated industries, can’t send data to outside services. Even those without strict rules are often careful about letting data leave their control.

TestGen avoids this issue altogether. You install it in the client’s environment, it checks the data in place, and the results stay local. There’s no SaaS, no cloud dependency, and no risk of data leaving. This makes it a good fit for clients who wouldn’t approve a cloud-based data quality tool.

Most importantly, recommending open-source tools aligns your interests with your clients'. When they are ready to run production testing with multiple users, DataKitchen offers a low-cost enterprise version. DataKitchen has excellent support from experienced data people like you.

Clients notice and value this alignment. It builds trust, which is key to long-term consulting relationships.

Your Next Step

Begin by learning the tools yourself. Install TestGen, connect it to some data you can access, and see what it finds. Get to know its profiling, test generation, and reporting features. The better you know the tools, the more confidently you can show them to clients.

Next, look for opportunities with your current clients. Are there accounts with data quality issues you haven't tackled? Which clients could use a data health assessment? Which projects could benefit from better testing and observability? You probably have chances you haven't noticed yet.

Operationalize the "Frictionless Value" framework in four steps.



Then, define your service. How will you present data quality assessments? What's the scope, timeline, and price? What will you deliver? A clear offering makes it easier to talk with prospects and clients.

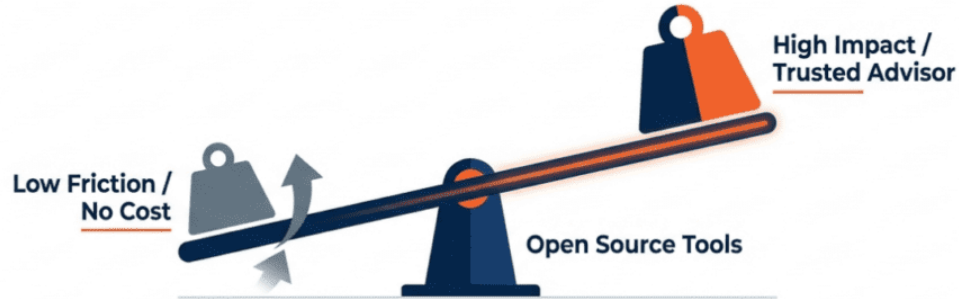
Finally, invest in your team's skills. Share the free DataKitchen resources with them. Encourage them to complete certifications. Make data quality and observability a core strength of your practice.

Conclusion

Consulting is all about creating value for clients and earning a share of that value as fees. The better you are at spotting client problems, showing solutions, and delivering results, the more successful your business will be.

DataKitchen's open-source data quality and observability tools give consultants strong capabilities at every stage of the client relationship. They help you win new clients by showing value right away, deepen relationships by uncovering new problems, and improve your deliverables with automated testing and advanced monitoring. The free educational resources also help your team grow over time.

Open Source Data Quality transforms engagements from "Time & Materials" to "Value-Based" partnerships.



- **Acquire:** Prove value before the contract.
- **Deepen:** Uncover new work continuously.
- **Deliver:** Automate quality for higher margins.

The tools are free; the revenue opportunity lies in the expertise you wrap around them.

TestGen helps data consultants and engineers by providing a low-friction, automated way to identify, measure, and resolve data quality issues while simultaneously acting as a business development engine.

Author: Chris Bergh | Published: Jan 18, 2026

Original article: <https://datakitchen.io/the-consultants-growth-playbook-accelerating-client-acquisition-and-retention-with-datakitchen-open-source-testgen/>

How I Broke Our SLA and Delighted Our Customer

Failing the SLA was the price we paid for trust. And it was worth every second.



I broke one of our most critical SLAs just last week, and it was the best thing that could have happened.

It was shaping up to be a major embarrassment. One of our key data warehouse refreshes had failed. No new data. No dashboard updates. The refresh was long past its deadline, the project's key data engineer was on vacation, and I was playing backup. Where was I? At the moment, I was flying home from a data quality conference. This was not good.

Naturally, the airline's Wi-Fi was terrible, but I managed to get through to a colleague, who knew little about the project. And then we started troubleshooting—one Slack message at a time. I slacked him SQL. He slacked back data. Back and forth we went. An embedded test had failed. Where had our process gone wrong? Everything I could see looked fine. And I was tempted, so tempted, as the clock kept ticking, to disable the test and let it go.

Then it dawned on me that this test wasn't even ours. It was one of a series of tests that had been added by one of our principal stakeholders in collaboration with his analytics team – esoteric referential integrity checks that had been placed at key points in the build, validating consistency between multiple data sources that fed hourly into our reporting layer.

These tests weren't easy to define or implement. They came from an understanding of the business rules that we, as data engineers, lacked. And they caught a major problem: the new records we received from one source were completely out of sync with our other data.

Something had gone wrong upstream—very wrong. And because our process was designed to fail fast when integrity was violated, it refused to publish anything at all.

We trusted stakeholders to define critical business rules that would test for major problems. They trusted us to implement their logic with every refresh. We trusted each other not to blame and shame, but to steadily build out better and better testing to benefit everyone.

Now, together, we had protected leadership from a serious integrity issue. We had prevented a flood of inaccurate reports from reaching decision makers. And when we explained what happened—why the data was late and what we had caught—they weren't frustrated. They were grateful.

The value of data quality is often invisible. When things go right, no one notices. But this was a moment where things went wrong exactly the way we hoped they would: the system raised a red flag, we caught the issue before it caused damage, and we preserved the integrity of our analytics.

The takeaway? If you're not building comprehensive, automated data quality checks into your production pipelines, you're one bad refresh away from losing stakeholder confidence and risking devastating business outcomes.

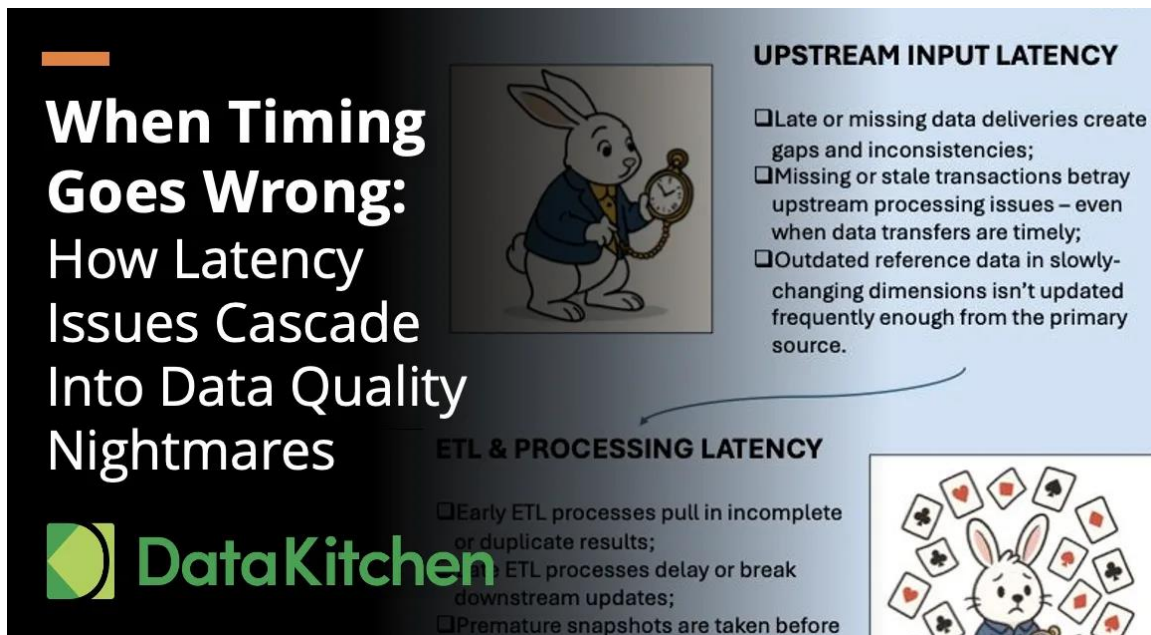
Your data processes shouldn't just be fast—they should be right. And sometimes, the best outcome is one where the system refuses to run.

Author: Chip Bloche | Published: May 17, 2025

Original article: <https://datakitchen.io/how-i-broke-our-sla-and-delighted-our-customer/>

When Timing Goes Wrong: How Latency Issues Cascade Into Data Quality Nightmares

Timing is EVERYTHING. How Latency Issues Spawn Data Quality Problems



As data engineers, we've all been there. A dashboard shows anomalous metrics, a machine learning model starts producing bizarre predictions, or stakeholders complain about inconsistent reports. We dive deep into data validation, check our transformations, and examine our schemas, only to discover the real culprit was something far more subtle: timing.

Process latency – the delays and misalignments when data moves through our systems – is one of the most underestimated threats to data quality in modern architectures. While we've gotten excellent at building robust pipelines and implementing data quality checks, we often treat timing as someone else's problem. This is a dangerous oversight.

Timing is EVERYTHING

How Latency Issues Spawn Data Quality Problems



UPSTREAM INPUT LATENCY

- ❑ Late or missing data deliveries create gaps and inconsistencies;
- ❑ Missing or stale transactions betray upstream processing issues – even when data transfers are timely;
- ❑ Outdated reference data in slowly-changing dimensions isn't updated frequently enough from the primary source.

ETL & PROCESSING LATENCY

- ❑ Early ETL processes pull in incomplete or duplicate results;
- ❑ Late ETL processes delay or break downstream updates;
- ❑ Premature snapshots are taken before upstream processing is complete or available;
- ❑ Processes intermingle new and old versions of data.



INTEGRATION DRIFT



- ❑ Silos are forced to maintain separate versions of the same entity records;
- ❑ Delayed data reconciliation between enterprise source systems hinders integration;
- ❑ Divergent upstream integration schedules

The Hidden Nature of Latency Problems

Unlike schema violations or null value issues that surface immediately in our monitoring dashboards, latency problems are insidious. They don't break your pipelines – they corrupt your insights. A report might run successfully every morning, but if it's pulling yesterday's transactions against today's customer dimensions, you're serving up beautifully formatted garbage.

The challenge intensifies because latency issues rarely manifest as obvious timing problems. Instead, they masquerade as:

The Four Faces of Latency-Induced Data Quality Issues

This is where data quality problems often begin. When source systems deliver data late or inconsistently, it creates gaps that propagate downstream. Missing transactions, stale reference data, and delayed dimension updates all stem from this root cause. Your perfectly architected data warehouse becomes unreliable not because of design flaws, but because the data it depends on arrives unpredictably.

Even when source data arrives on time, processing delays create their own quality issues. Early ETL jobs might pull incomplete datasets, while late-running processes create cascading delays throughout your pipeline. Premature snapshots capture partial states, and overlapping batch windows intermingle different versions of the same data, creating temporal chaos.

The most complex challenge emerges when different systems maintain separate versions of shared entities. Customer records updated in your CRM don't immediately reflect in your analytics warehouse. Product catalogs in your e-commerce platform are updated, while your recommendation engine still references the old version. These synchronization gaps create a fragmented view of reality across your data ecosystem.

The final manifestation occurs when downstream consumers expect fresh data that isn't available. Reports are run on schedule, but they reflect outdated information. Machine learning models retrain on outdated features. Business users make decisions based on metrics that are hours or days behind reality, and they often remain unaware of this discrepancy.

Why We Ignore the Timing Problem

There's a reason latency issues persist despite their impact: they're fundamentally collaboration problems disguised as technical ones. It's much easier to focus on what's within our direct control – our schemas, our transformations, our infrastructure – than to coordinate schedules across teams, systems, and vendors.

We build our monitoring around what we can measure easily: pipeline success rates, data volumes, and schema compliance. But measuring cross-system timing dependencies? That requires visibility into other teams' processes, understanding of business logic we didn't design, and ongoing coordination that extends far beyond our immediate technical domain.

The Data Mesh Dilemma

As organizations embrace data mesh architectures, latency challenges multiply exponentially. When data ownership becomes distributed across multiple product teams, timing coordination becomes everyone's responsibility, which often means it becomes no one's responsibility.

Each domain team optimizes its data products independently. Sales refreshes customer segments nightly, marketing updates campaign performance hourly, and finance reconciles transactions on a weekly basis. These decisions make perfect sense within each domain, but the interactions between these schedules create systematic quality issues that no single team owns or can easily detect.

Taking Ownership of Time

The solution isn't to abandon modern data architectures, but to explicitly own the timing aspects of data quality. This means:

Treating schedules as first-class design artifacts. Document not just what data moves where, but when it moves and what depends on that timing. Make schedule dependencies as visible as data lineage.

Building latency into your data quality framework. Monitor not just whether data arrived, but whether it arrived when expected relative to its dependencies. Alert on timing anomalies before they become data quality incidents.

Establishing cross-domain timing contracts. In data mesh implementations, service level agreements should cover temporal aspects: not just “we’ll provide customer data,” but “we’ll provide customer data by 2 AM EST, reflecting all transactions processed by midnight.” Make these explicit as part of your [Data Journey](#).

Designing for timing failures. Build graceful degradation into your systems. When fresh data isn’t available, how does your system behave? Can it serve stale data with appropriate caveats rather than serving incorrect data silently?

Beyond Technical Solutions

Ultimately, addressing latency-induced data quality issues requires us to expand our definition of data engineering beyond purely technical concerns. We need to become coordinators of temporal complexity, building systems that not only move data efficiently but also do so predictably and transparently.

This means having uncomfortable conversations about cross-team dependencies. It means pushing back when unrealistic timing expectations create quality trade-offs. And it means designing systems that make timing problems visible rather than hiding them behind successful pipeline metrics.

The next time you’re debugging a mysterious data quality issue, don’t just look at the data – look at the clock. The problem might not be in your transformations or your infrastructure. It might be in time itself.

Timing issues in data systems are often the invisible root cause of visible data quality problems. By treating [process latency as a first-class concern in our data architecture](#), we can build more reliable systems and prevent quality issues before they impact business decisions.

Author: Chip Bloche | Published: Jun 18, 2025

Original article: <https://datakitchen.io/when-timing-goes-wrong-how-latency-issues-cascade/>

What Does Flossing One Tooth Have to Do with Data Quality?

Surprisingly, the answer may come from an unusual place.



Improving data quality can feel overwhelming. There are so many things to fix and so many processes to improve. Where do you even start?

*In the book *Tiny Habits* by B.J. Fogg, the author suggests a simple way to build habits. If you want to start flossing your teeth, don't aim for a perfect routine right away. Instead, start with just one tooth.*

At first, this sounds silly. Why floss just one tooth? But the goal isn't perfect flossing right away—it's about building the habit. Once flossing one tooth becomes easy, you naturally start doing more.

I recently worked with a data team struggling with production data errors. They started holding [quality circle](#) meetings to reflect on problems and find solutions. But there was an issue—after listing their action items, they never followed through.

Instead of overwhelming themselves with a long to-do list, they could focus on one easy change and start forming a new organizational habit. Once the habit is formed, and the team is proficient at following up on actions, they can move to addressing more significant issues.

If your team struggles with follow-through, don't try to fix everything at once. Instead, start small. At your next quality circle, choose just one action item that's easy to complete. Once you build that habit, bigger improvements will follow.

Author: Gil Benghiat | Published: Apr 3, 2025

Original article: <https://datakitchen.io/what-does-flossing-one-tooth-have-to-do-with-data-quality/>

We Got Roasted On Reddit For Asking ‘Why Data Engineers Don’t Test?’

We asked the *r/dataengineering* community 'Why don't you test?' And we got roasted. Learn why



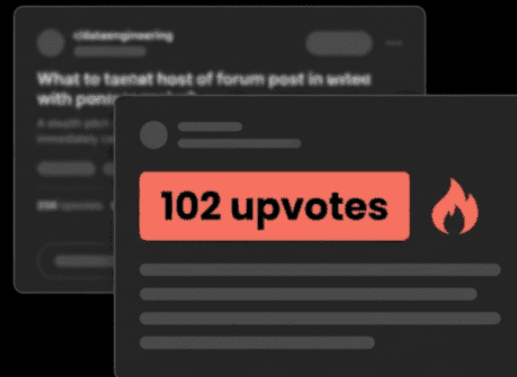
DataKitchen co-founder Gil Benghiat recently posted a question to [r/dataengineering](https://www.reddit.com/r/dataengineering) — one of the largest and most active data engineering communities on the internet. The question was genuine: “What is actually stopping teams from writing more data tests?”

The problem? He didn’t disclose who he was. Within hours, a lead data engineer named SearchAtlantis spotted that just one week earlier, Gil had posted in a separate thread and explicitly said: “full disclosure, I co-founded DataKitchen and we built TestGen for exactly this problem.” The callout got 102 upvotes. The replies were swift and merciless. “His picture tells a thousand words,” wrote another commenter. The community, as it tends to do, did not let it slide.

To Gil’s credit, he acknowledged it directly: “That is fair. I co-founded DataKitchen and should have said so here. The question is still genuine, though.”

We asked Reddit a genuine question and got roasted

- A stealth pitch to r/dataengineering was immediately called out.
- The community did not let it slide.
- But then, the thread exploded with 60+ brutally honest answers.



Here is what they said.

And it was genuine. The thread went on to generate over 60 thoughtful comments from working data engineers, and the insights that emerged are some of the most candid and practically useful we've seen on the topic. We're sharing them here — with full transparency about their source.

What's Actually Stopping Teams From Writing Data Tests?

The most upvoted response in the entire thread cut straight to the heart of it: writing tests that catch real bugs without being noisy is just difficult. This isn't false modesty — it's a real skill gap that takes time and painful experience to develop.

The problem isn't the mechanics of writing a test. It's knowing which tests are worth writing. Teams that skip this step and add tests just to "check a box" often end up worse off: tests that fire on irrelevant conditions, break jobs for illegitimate reasons, and eventually get deleted. Bad tests erode trust in the whole testing practice. Once engineers stop believing their alerts mean something, the entire system collapses.

One engineer described the downstream effect plainly: "I tried implementing a basic testing practice at my workplace... my manager insisted we need some version of unit testing on every table too as part of the policy. Just adding tests for the sake of claiming we test has been unnecessary extra work... then when those tests eventually fail and break a job it turns out we actually don't care or the dev never verified the rule was legitimate in the first place."

The compounding problem: DAG timing makes this worse. When pipelines have multiple arms refreshing on different cadences, false positives multiply. Engineers start ignoring alerts. And once alerts get ignored, you've lost the whole point.

02 No one has an operational playbook

Having 100 tests means nothing. Defaulting to not writing tests isn't laziness—it is rational risk avoidance.



One of the most insightful comments in the thread reframed the entire problem: "Writing tests isn't the hard part. It's deciding the operational playbook for each of the tests that gets tricky."

Without answers to these questions, [tests become noise](#). And in the absence of a clear playbook, teams default to not writing them at all. It's not laziness — it's rational risk avoidance.

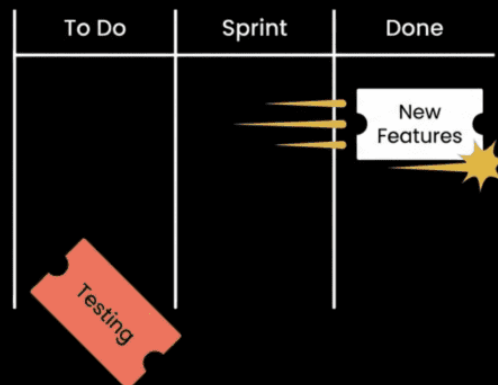
Gil's own response in the thread is worth noting here: "Not all tests are equal. Some should stop the pipeline, some should be warnings that are investigated later or documented as release notes for data consumers, and others are more metric tracking than tests — a way for the data engineer to stay calibrated on how the data behaves. If a test is constantly failing and nobody acts on it, then remove it."

03 Feature delivery always wins

Delivery has deadlines and visibility.
Testing gets cut silently.

No one cares, there's too much work and too little time to do it in. Just deploy and if there are bugs, take them up in subsequent sprints.

— CornflakesKid



At least four separate commenters called this out directly, and it's probably the most universal barrier on the list. During a sprint, tests lose. [Every time](#).

The dynamic is simple: delivery has a deadline, stakeholders, and visibility. Testing has none of those things — until something breaks in production and suddenly it has all three.

One senior data engineer summed it up bluntly: “From what I’ve seen, it is more of that last one. No one cares, there’s too much work and too little time to do it in. Just deploy and if there are bugs, take them up in subsequent sprints.”

This isn’t just an individual failure. It’s an organizational one. When teams are too small for the workload, testing is the first thing to get cut. And data testing gets cut silently, without anyone explicitly making that decision.

04 **Tests rot fast without a maintenance budget**
Upstream schemas shift. Business logic changes. Tests that were accurate six months ago now silently pass bad data.



A lot of teams don’t have stable expectations, schemas and upstream logic shift, so tests rot fast unless you budget time to maintain them.

— [calimovetips](#)

Even teams that invest in testing often find that investment decays. Upstream schemas change. Business logic shifts. A vendor updates their API without warning. Tests that were accurate six months ago now fire constantly — or, worse, silently pass bad data because the world has changed around them. One commenter described it well: “a lot of teams don’t have stable expectations, schemas and upstream logic shift, so tests rot fast unless you budget time to maintain them.”

Maintenance rarely gets prioritized. It doesn’t appear on the roadmap, doesn’t get sprint points, and doesn’t have a deadline. So tests slowly rot, confidence erodes, and eventually someone deletes the whole suite and starts over — or doesn’t bother starting over at all.

One commenter captured the fatigue well: teams that have been burned by upstream changes enough times learn to write schema validation and bail early rather than invest in elaborate fixtures, because fixtures require their own maintenance, which requires its own tests — a matryoshka doll of testing infrastructure that most teams don’t have the bandwidth to sustain.

05 You don't know what to test until it's already broken

Without experiential knowledge and deep business context, tests remain shallow. Row counts won't catch when revenue goes to zero.

Lack of experience. Those of us who have been burned like you were with that four-hour pipeline that did nothing write data tests.

— Atmosck



Data testing knowledge is largely reactive. Most engineers learn what to test by getting burned — a pipeline that produced zero rows, a join that silently dropped 40% of records, a schema change from upstream that nobody announced.

Until you've lived through a few of those, it's hard to know where the landmines are. And without that experiential knowledge, tests tend to be shallow — row counts, maybe nulls — and miss the things that actually matter.

One commenter put it plainly: "Lack of experience. Those of us who have been burned like you were with that four-hour pipeline that did nothing write data tests." Another echoed it: "You don't realize you need them until you've scrambled to get shit fixed cuz things are broken and the heat is on and it's embarrassing. That's when you realize oh wow it doesn't have to be like this."

The deeper version of this problem is domain knowledge. One commenter observed that they'd seen teams with hundreds of tests still miss "revenue went to 0"—because nobody on the data team understood the business well enough to recognize that it was a possible failure mode worth testing for. As Gil put it in response: "Many data teams just move data and don't understand the business." Technical competence and business context are both required, and the latter is often missing.

06

Data failures sneak through silently

A pipeline producing garbage rows completes successfully, turns green, and sits quietly in a dashboard.

CODE



DATA



"Code breaks loudly. A pipeline that silently produces garbage? That one sneaks through."

— Reddit Contributor

Code breaks loudly. A build fails, a deploy errors out, and an exception gets thrown. Data failures are different. A pipeline that produces garbage rows completes successfully. The job turns green. No alert fires. The wrong numbers sit quietly in a dashboard until someone notices them in a meeting — usually the wrong meeting, at the wrong time, in front of the wrong people.

One commenter nailed the asymmetry: "Code breaks loudly. A pipeline that silently produces garbage? That one sneaks through."

Because data failures are invisible until they're embarrassing, there's rarely organizational pressure to prevent them in advance. Leadership doesn't feel the cost of missing tests until a crisis makes it undeniable. And by then, the damage is done.

07

Catching the error makes it your problem

The incentive structure in many organizations actively punishes engineers who look too hard. Rational engineers respond accordingly.

If you caught an error, it turns into your problem.
If you delivered wrong data where the root cause is not the pipeline, it is not your problem.

— danryushin



This one is uncomfortable but widely recognized. One commenter articulated it clearly: "If you caught an error, it turns into your problem. If you delivered wrong data where the root cause is not the pipeline, it is not your problem."

The incentive structure in many organizations actively [punishes engineers who look too hard](#). Finding a problem means owning the investigation, the fix, the communication to stakeholders, and the post-mortem. Not finding it means the problem is the upstream's fault.

Gil's take: the only fix is leadership that actively rewards people who surface problems rather than penalizing them for finding them. This is ultimately a culture-and-values issue, not a tooling one.

08 At exabyte scale, we need a testing strategy that fits.

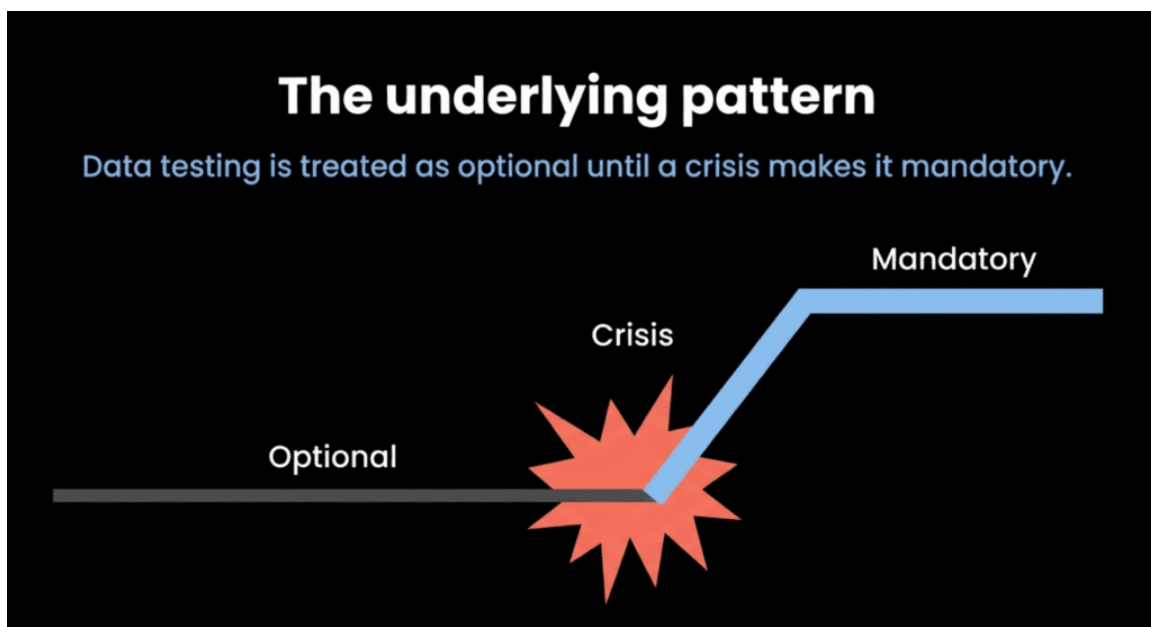
Instead of full validation at every stage, we must focus on smart, targeted testing. The goal is to find the right balance—testing what matters most, not everything.

Don't boil the ocean. Focus on the critical paths and high-risk areas. That's how you handle exabyte scale effectively.

— Data Architect

One commenter described processing close to an exabyte of data per day. At that volume, full validation at every stage is too slow, too expensive, and practically impossible. Their solution — sampling, anomaly detection, and back-testing on pipeline deployments — is the pragmatic answer to a genuine constraint. As they put it: “At our volume something that is 1 in a billion will happen several times a day, so if it can happen, it will happen — plan for the possibility.”

This is worth naming clearly: for some teams, the [barrier to comprehensive testing](#) isn't skill or culture. It's physics and economics. The right answer at hyperscale looks very different from the right answer at a mid-market company running 50 pipelines.



The Pattern Underneath All of It

Across every comment in the thread, one pattern emerges consistently: data testing is treated as optional until a crisis makes it mandatory.

Most teams build test coverage reactively — one incident at a time — rather than proactively as a standard engineering discipline. The barriers are a mix of technical difficulty, organizational culture, misaligned incentives, and resource constraints. Tooling is rarely the limiting factor.

These aren't problems that disappear when you hand a team a testing tool. They require investment in process, culture, and clear operational norms around what testing means and what happens when it fails.

Fixing the gaps requires open access

Tooling behind an enterprise paywall doesn't fix a cultural problem. That is why we open-sourced TestGen.

- Generates tests automatically to overcome the blank file barrier.
- Structured framework to separate critical stops from calibration signals.
- Flat-rate pricing for enterprise so it doesn't scale against you.

```
> pip install testgen-open-source
> testgen run --auto-profile
✓ Profiling data...
✓ Generating calibration tests...
```

The community was right. It's time to build better safety nets.

We're Data Engineers Too — This Is Why We Open-Sourced TestGen

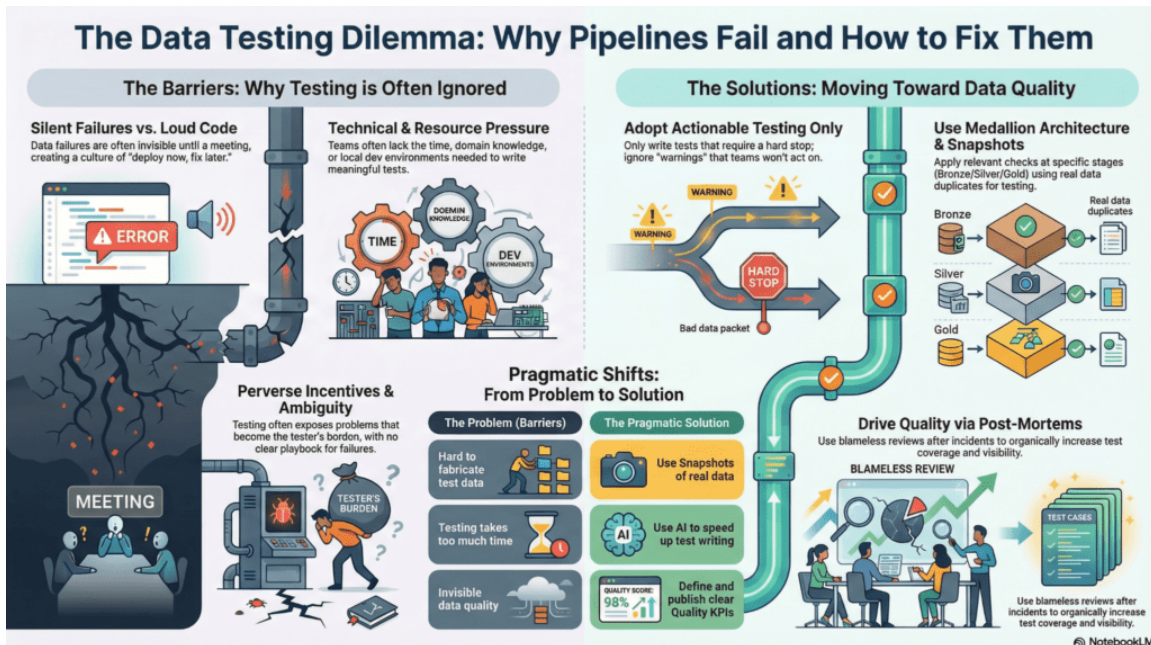
The most common reasons teams don't test aren't technical. They're about time, trust, incentives, and not knowing where to start. A tool sitting behind an enterprise paywall doesn't fix any of that. Giving engineers direct, no-commitment access to a production-ready testing tool — one that profiles your data and generates tests automatically, so you're not starting from a blank file — actually does.

TestGen is free to download, free to use, and built to address the specific gaps this community named: it reduces the expertise barrier for knowing what to test, cuts the time cost of writing tests from scratch, and gives teams a structured framework for deciding which tests are critical, which are warnings, and which are just calibration signals. We've been doing data engineering for a long time, and have run into every issue listed in the article. And we built TestGen as the free antidote.

For teams that want to go further — shared test suites, collaboration across engineers, and enterprise support — our flat-rate pricing means you're not getting hit with per-user or per-usage fees that scale against you as your team grows. One predictable number, regardless of how many tables, pipelines, or tests are involved.

Because the community was right: the problem is real, it's widespread, and it's been undersolved for too long.

Source: [r/dataengineering thread](#), "What is actually stopping teams from writing more data tests?" — community responses plus OP follow-up from DataKitchen co-founder Gil Benghiat.



What is the Summary of this article?

Data engineers know they should test more — but a candid Reddit thread reveals the real reasons they don't. The barriers are less about tooling and more about human and organizational dynamics: good tests are hard to write without getting burned first, there's no clear playbook for what to do when a test fails, and feature delivery always wins the sprint. Even teams that invest in testing watch their suites decay as upstream schemas shift, and maintenance never gets prioritized. Underneath it all sits a perverse incentive — finding a problem means owning it — and a culture in which data failures remain invisible until they're embarrassing. These are exactly the barriers DataKitchen built TestGen to address, and why they open-sourced it.

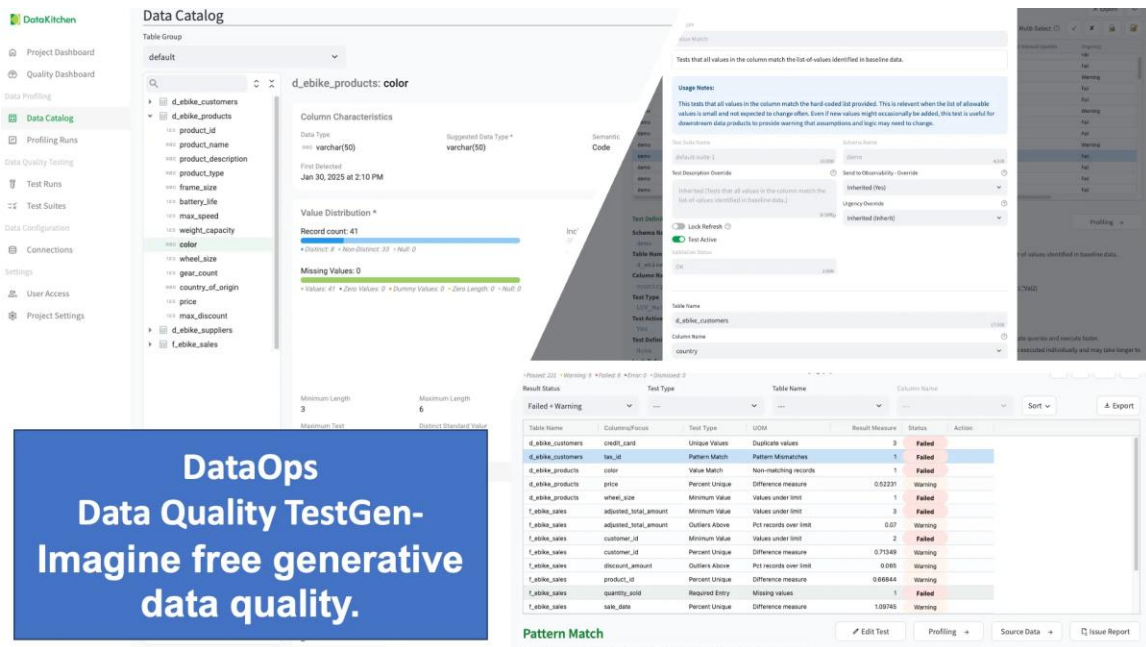
The *r/dataengineering* community named eight specific reasons data engineers don't write more tests. Here is how TestGen was built to address every one of them.

TestGen Fixes Barrier 1: Writing Good Tests Is Genuinely Hard

The community's top-voted concern was that writing tests that catch real bugs without generating noise requires deep expertise that most teams simply haven't had time to develop. TestGen removes this barrier at the source through what it calls one-button data quality checks: point TestGen at your data, and it generates tests automatically without requiring deep knowledge of the underlying dataset, custom coding, or sprawling YAML configuration.

The mechanism behind this is data profiling. TestGen performs a column-level X-ray of every table, gathering extensive information about what each column actually contains — averages, min and max values, unique value counts, date characteristics, percentiles, data types, and more. Based on those profiling results, the algorithm generates over 120 AI-driven tests covering data integrity, hygiene, and quality. These include checks for things like alpha truncation, average shift, constant value presence, future dates, distinct value changes, and daily record counts — tests that experienced engineers write after getting burned, but that TestGen generates proactively before a single pipeline runs.

Underpinning all of this is a semantic data model that captures the meaning and relationships of your data assets, not just their structure. This means TestGen doesn't just know that a column exists — it understands what the column represents, which allows generated tests to be meaningfully calibrated to the data's actual purpose rather than its raw technical characteristics alone.



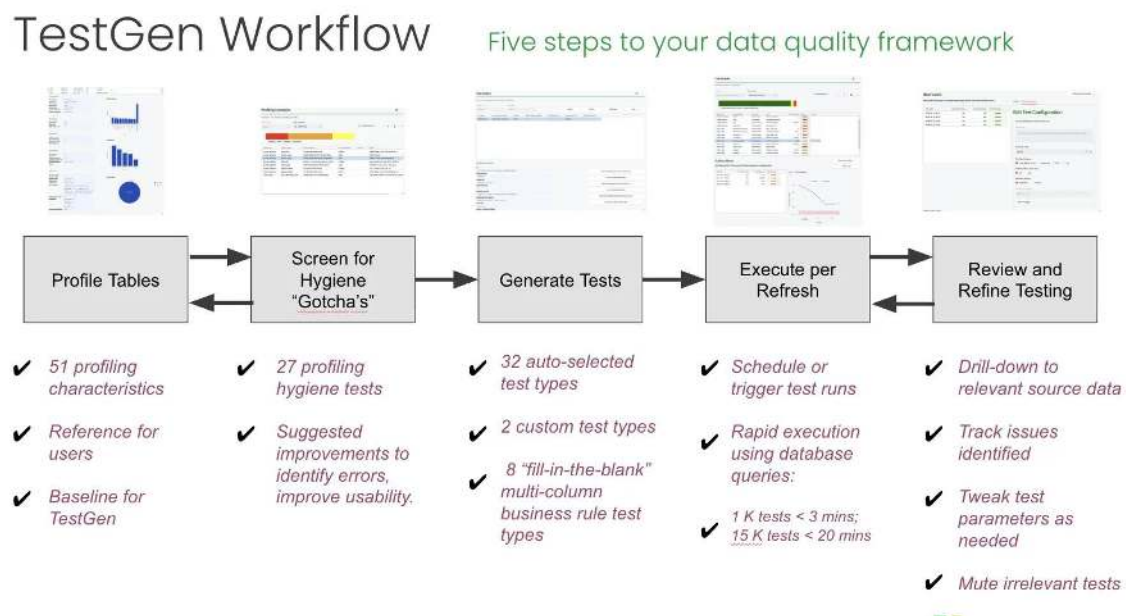
DataOps
Data Quality TestGen-
Imagine free generative
data quality.

The explicit design goal is to cast a wide net for problems that can't be predicted by targeted testing alone. TestGen describes it as the equivalent of setting up a burglar alarm by covering all possible entrances rather than guessing which window someone will try.

TestGen Helps Barrier 2: No One Has a Playbook for When Tests Fail

The thread's second most-discussed barrier was operational: teams don't know which failures should stop a pipeline, which should trigger a warning, and which are just signals to monitor. Without answers to those questions, writing tests feel premature.

TestGen addresses this structurally with customizable quality scoring and dashboards. Every test result feeds into automated scorecards that can be configured by DAMA data quality categories, by Critical Data Element designation, by a specific data science model, or by current business goals. Teams can drill down from a high-level score into the specific tables and columns driving quality issues, which turns a wall of test results into a ranked, prioritized view of what actually needs attention.



Critically, this isn't just a tool for engineers. Because the open-source version ships with a full UI out of the box, business partners and data consumers can participate directly in testing alongside the engineers who build the pipelines. Tests can be reviewed, annotated, and managed collaboratively without anyone needing to touch the command line. Business rules can be configured and shared with non-technical stakeholders rather than requiring a database programmer to translate requirements into SQL. This shared interface is what makes it possible to align on which test failures are critical, which are warnings, and which are simply signals to monitor over time — turning what is usually a solo engineering judgment call into a team decision.

On top of that, TestGen's shareable issue reports let engineers get influence and action on data quality problems with a single click, pushing findings to the people who need to act on them rather than leaving results buried in a dashboard nobody opens.

TestGen Helps Barrier 3: Feature Delivery Always Wins

Time is the most universal barrier in the thread. Testing loses to delivery every sprint because manual test authorship is genuinely time-consuming — weeks to cover a fraction of the tables, as TestGen's own positioning acknowledges.

TestGen ROI: 7.2 Months of Engineering vs. 2 Steps of Automation



THE MANUAL GRIND

- 30 Minutes per test
- 1,250 Total Hours coding
- 156 Days (8 hrs/day)
- 31 Weeks (5 days/week)

7.2 MONTHS

With no time for meetings, breaks, or vacations.



THE DATAOPS ADVANTAGE

Profile Data
Generate Tests

2 STEPS

2,500 tests generated immediately.



TestGen compresses **156 days** of senior engineering work into **minutes**.

DataKitchen

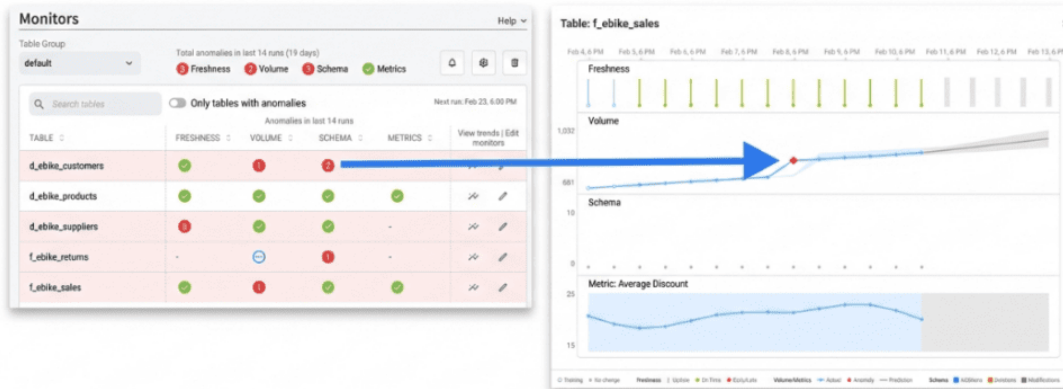
TestGen's answer is speed through automation. Because test generation flows directly from profiling, an engineer can go from connecting a database to running thousands of tests covering every table and column in under an hour — without writing a single test by hand. What typically takes data teams weeks to build manually, and still covers only a fraction of their tables, TestGen delivers complete test coverage in a single session. TestGen also runs those tests with blazing-fast in-database execution, pushing queries directly into the data warehouse rather than extracting data first, which means the performance cost of running tests is low enough not to become another argument against doing it. The time investment drops sufficiently that testing no longer competes with delivery and becomes something that happens alongside it.

TestGen Fixes Barrier 4: Tests Rot When Upstream Data Changes

Even teams that invest in testing watch their suites decay as upstream schemas shift. Tests written against yesterday's data structure start firing on irrelevant conditions or silently passing on newly invalid data.

Drill down into anomalies with historical context.

View anomalies alongside table history and schema changes—no digging through log files.
View anomalies alongside table history and schema changes—no digging through log files.

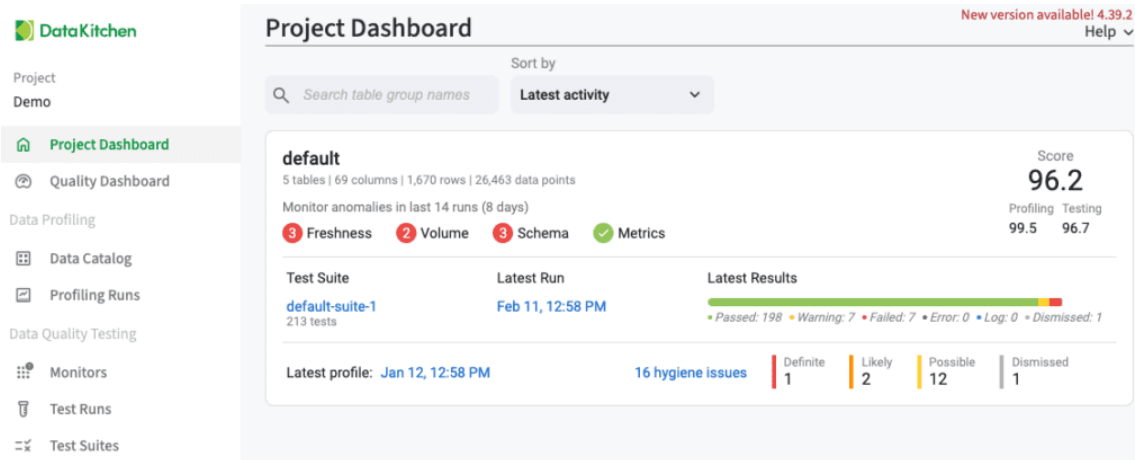


TestGen's anomaly detection is built specifically to catch this. It continuously monitors for schema changes, data drift, freshness degradation, and volume anomalies, generating automated alerts when something deviates from expected behavior. But the more important answer to test rot is a practice TestGen is designed to support directly: regular re-profiling and test regeneration. Rather than treating a test suite as a one-time artifact that slowly decays, TestGen encourages teams to re-run profiling on a scheduled basis and regenerate tests from the updated profile. When upstream data changes, the new profile reflects those changes, and the regenerated tests are calibrated to what the data actually looks like now rather than what it looked like six months ago. This turns test maintenance from a manual, easily-skipped chore into a repeatable, automated part of the data pipeline lifecycle. The data catalog adds another layer: it provides a 360-degree view of every table's metadata, profile results, hygiene issues, and test results, so engineers can see when the ground has shifted rather than discovering it after a silent failure.

TestGen Fixes Barrier 5: You Don't Know What to Test Until It's Already Broken

Most data testing knowledge is reactive. Engineers add tests after failures, which means every test was preceded by a production incident.

TestGen's dataset screening feature addresses this directly. Before production testing begins, TestGen performs an initial assessment of new data sources, running 27 data hygiene detector tests that automatically identify structural problems: invalid zip code formats, leading spaces, multiple data types in a single column name, non-standard blank values, columns with no values, and more. These are the kinds of issues that cause silent failures downstream but are invisible until something breaks. TestGen surfaces them upfront, during the screening phase, before data moves further into the pipeline.



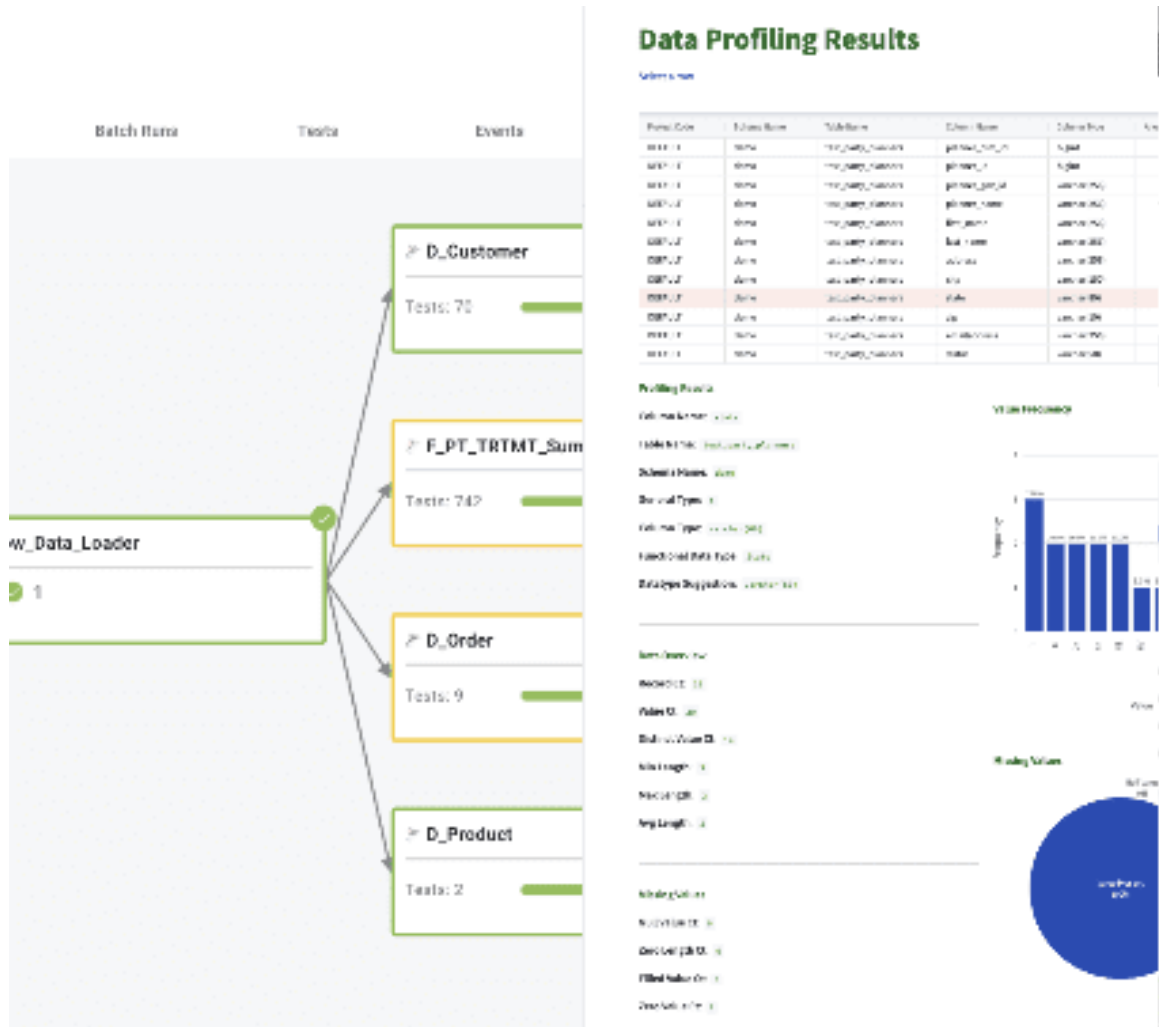
From that same profiling and screening pass, TestGen automatically generates a full suite of production-ready validation tests — no manual authoring required. The auto-generation doesn't just create generic checks; it creates tests calibrated to the actual characteristics of each column, using the semantic data model and profiling results to derive thresholds, expected value ranges, and format rules that would take an engineer hours to specify by hand. For teams that have historically relied on experience and incident history to know what to test, this is the most direct possible answer: TestGen builds the institutional knowledge into the generation process itself.

Combined with the 51 column-level profiling characteristics, engineers get a complete picture of what a dataset actually contains before they write a single transformation. The data catalog then surfaces potential PII risks and Critical Data Elements within the same view, thereby partially filling domain knowledge gaps with what the data itself reveals.

TestGen Fixes Barrier 6: Data Failures Are Invisible Until They're Embarrassing

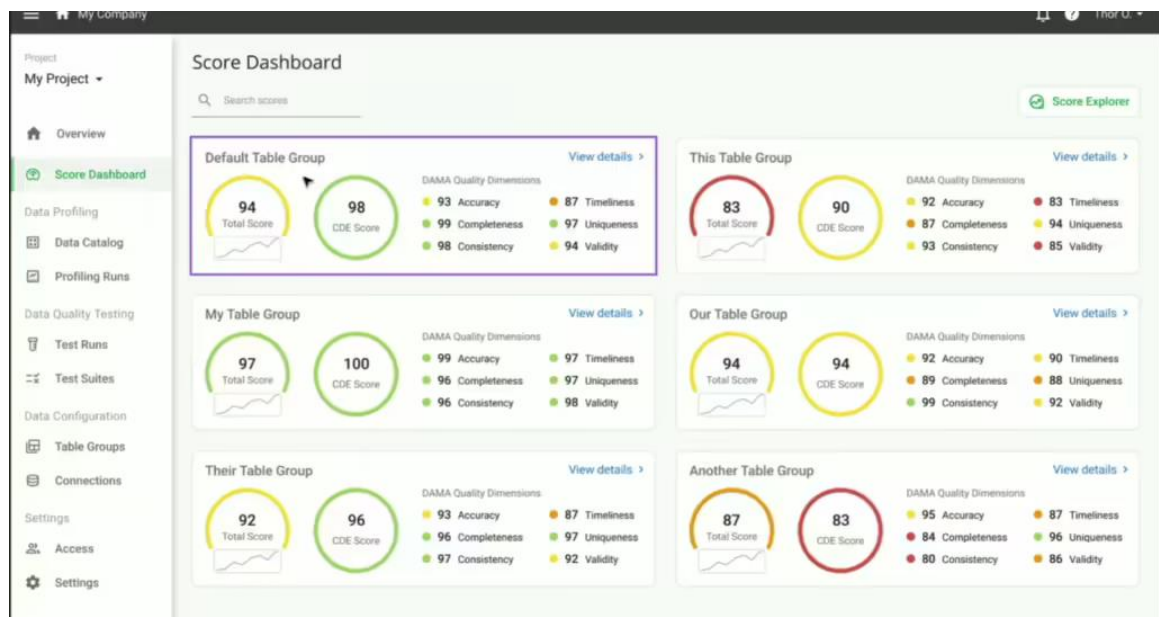
Code breaks loudly. Pipelines that produce garbage data complete successfully and turn green. The wrong numbers sit quietly in a dashboard until they surface in the wrong meeting.

TestGen's ML-based anomaly detection is the direct answer to this. It continuously monitors data for freshness issues (did the data update when expected?), volume anomalies (is row count trending in the right direction?), schema changes (has something been added, removed, or altered upstream?), and data drift (have the statistical characteristics of key columns shifted?). Crucially, the system is designed to alert without overwhelming — the explicit goal is to get alerted without being bothered by every transient issue, which means the signal stays meaningful rather than becoming noise.



TestGen Helps Barrier 7: Catching the Error Makes It Your Problem

One of the sharpest observations in the thread was that in many organizations, the engineer who finds a problem owns it. That perverse incentive actively discourages engineers from looking too hard.



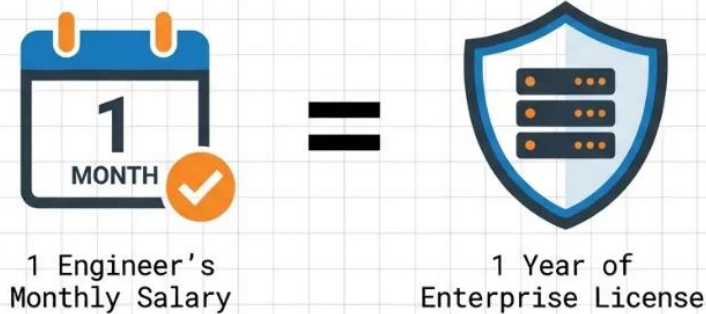
TestGen Helps Barrier 8: At Scale, Comprehensive Testing Isn't Viable

For some teams, the barrier is not skill or culture but physics and economics. At hyperscale, exhaustive row-level validation at every stage is too slow and too expensive.

TestGen handles this in two ways. The in-database execution model is central here: rather than pulling data out of the warehouse to test it, TestGen pushes queries directly into the database. This makes testing dramatically faster and more cost-efficient than approaches that require data movement, and it keeps data within the warehouse's security boundary rather than exposing it to external systems. For environments where even in-database full scans are prohibitive, TestGen supports configurable sampling — teams can set a sample percentage and minimum count per table group, so coverage stays broad without the compute cost of scanning billions of rows on every run.

Flat pricing designed for scale, not exploitation.

\$100/user/month. No "Table Tax." Monitor every table in every schema without the bill growing.



On pricing, the flat-rate model of \$100 per month per user and per database connection with unlimited tables means there is no per-table tax that grows as the data estate expands. Teams at scale are not penalized for having more data, which removes the economic argument against expanding coverage.

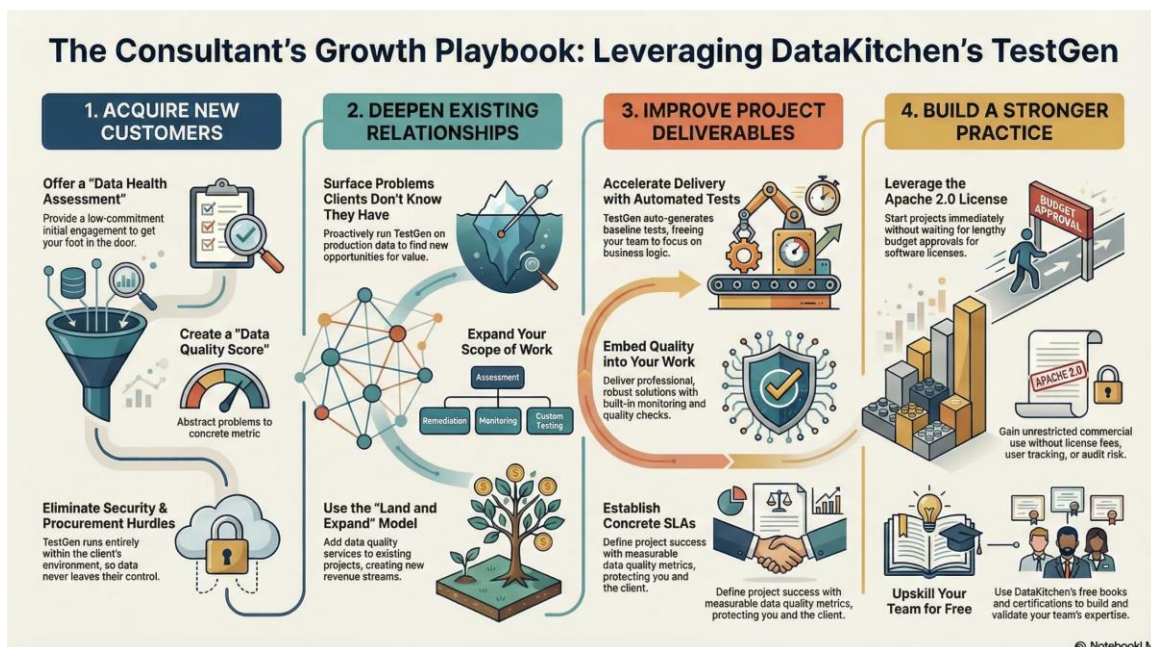
The Bottom Line

Author: Gil Benghiat | Published: Mar 4, 2026

Original article: <https://datakitchen.io/we-got-roasted-on-reddit-for-asking-why-data-engineers-dont-test/>

The Data Consultant's Growth Playbook: Accelerating Client Acquisition and Retention with DataKitchen's Open Source TestGen

BlogCase StudiesWhite PapersWebinars / PodcastsDocumentation



If you've worked in data and analytics consulting, you know the hardest part isn't always the technical side. The real challenge is finding new clients, growing existing accounts, and proving your value to justify your fees.

The tools you choose make a difference. They impact the quality of your work and your ability to win and grow business. Expensive enterprise data quality platforms can be a barrier. Clients often hesitate at high licensing costs before seeing any value, and recommending pricey tools to budget-conscious organizations can stall deals for months.

DataKitchen's open-source data quality and observability tools, especially [DataOps Data Quality TestGen](#), give you a strong advantage. You can show your team's value and highlight the need for your services without asking clients to commit to a budget right away. By connecting to their data and uncovering hidden issues, you can start conversations that lead to real projects.

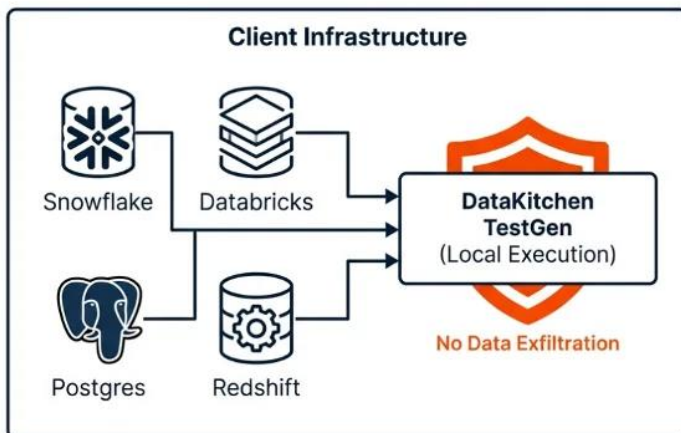
Goal 1: Acquire New Customers

Every consultant faces the same question: how do you get started with a prospect who doesn't know you? Cold emails and presentations only go so far. What really helps is showing you understand their specific problems, sometimes even better than they do. TestGen gives you a clear way to do this.

One approach that works is to offer prospects a simple "data health assessment" to start. Present it as a quick exercise where you connect TestGen to a sample of their data and create a detailed data quality report.

The mechanics are straightforward. TestGen connects to major database platforms (including Snowflake, Databricks, PostgreSQL, SQL Server, and Redshift). Once connected, it automatically profiles the data and generates a baseline set of data hygiene metrics and a data quality score without requiring custom logic. Within hours, you have concrete findings.

The 'Discovery Engagement' uses a low-risk Data Health Assessment to secure new business.



The Approach:

A 2-3 day high-impact exercise.

Process: Connect -> Auto-Profile
-> Generate Baseline Tests.

The Security Advantage:

Runs entirely in-house. Data never leaves the client's infrastructure.

Sidesteps CISO objections and lengthy security reviews.

TestGen is especially good for client assessments because it runs entirely in-house. You set it up in the client's environment, so their data never leaves their systems. This removes security worries that can slow down discussions about data tools. If a CISO asks where the data goes, you can say it stays put. TestGen checks the data in place and saves results locally. For consultants, this means you can suggest running TestGen on production data without long security reviews or procurement delays.

Since TestGen works on unlimited tables, you aren't limited in what you can check. Unlike commercial tools that charge per table or data source, you can profile and score all of a client's data if needed. This allows you to give a complete assessment and show the full range of data quality issues across the organization.

You can turn TestGen's results into a [data quality score](#)—a single number that gives executives something clear to consider. Most organizations don't know how to measure their data quality. They might say it's "messy" or that some reports "can't be trusted," but they don't have a consistent way to track it.

Converting technical hygiene issues into a 'Data Quality Score' creates executive urgency.

The Solution: A Synthesizable Metric

Executives lack a systematic way to measure "messy data."

This Report Card moves the discussion from abstract services to concrete evidence of a problem.

"You're presenting evidence of a problem and positioning yourself as the person who can fix it."

Data Quality Report Card			67/100
DATA QUALITY BREAKDOWN			
Data Asset	Grade	Issue / Status	
Customer_Master	D →	34% Null Values	
Sales_Transactions	B ✓	Key Integrity Valid	
Inventory_Log	C ⚠	Orphaned Records Detected	

DataKitchen
Data Health Assessment

If you tell a VP of Analytics that their customer data scores 67 out of 100, and show which tables and columns are lowering that score, you change the conversation. You're not just selling consulting services—you're showing real evidence of a problem and positioning yourself as the one who can solve it.

The assessment itself is a small task, but it reliably leads to larger conversations. Once you've shown a client their data quality issues, the natural next question is: what do we do about it? This opens doors to remediation work, data pipeline improvements, governance implementations, custom test development, and ongoing data quality monitoring arrangements.

Goal 2: Deepen an Existing Customer Relationship

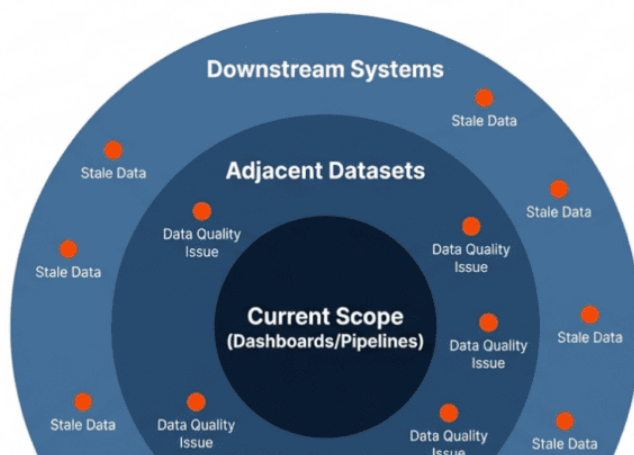
It's usually easier to grow a business with current clients rather than to find new ones. They already trust you, know how you work, and often have a budget set aside for your services. The key is to find new problems to solve and show value beyond what you're already doing.

Data quality work is often a missed opportunity in accounts where you're already handling analytics, pipeline development, or data science projects.

Even clients you know well may have data quality issues you haven't found yet. This happens because data problems often stay hidden until something breaks later on. Teams usually focus on building new features instead of checking existing data, and no one wants to point out problems without offering solutions.

TestGen helps you spot issues early and come prepared with both the problem and a solution. Try running it on data sources you haven't checked before. You can profile, test, and score datasets from other teams or systems that feed into your pipelines. You'll likely find data hygiene issues worth talking about.

Existing clients have invisible data problems that provide opportunities to expand scope.



Strategy: Proactive Diagnosis

- Run TestGen against data sources not currently in scope.
- Execute read-only queries against live production data, not sanitized staging copies.

Insight:

- Data quality issues are often transient or cumulative. Snapshots miss them.
- Identifies hygiene issues worth discussing without waiting for a system failure.

TestGen is built to run safely on production data, so you can use it on live systems. There's no need to make sample datasets or use old copies. The tool runs read-only queries on the real data your client relies on, giving you results that reflect the current situation, not just a cleaned-up sample.

TestGen can also monitor data for unusual changes over time. It automatically spots shifts in frequency, volume, schema, and other patterns. You can set up TestGen on a database, let it generate alerts for a week or a month, and then share those alerts with your client. For example, you might ask, "Did you know this data was late? Did you notice the schema changed? Are you missing data here?"

First, there's fixing the problems. When you find [data quality issues](#), someone has to address them. This could mean updating source system checks, adding logic to pipelines, or working with other teams to solve the root causes. Since you found the issues, you're in a good spot to lead this work.

Third, there's custom testing. While TestGen creates automated tests, clients often need custom tests for their specific business rules. For example, a retail client might want to check that inventory counts match physical audits, while a financial services client might need tests for regulatory metrics. Building these custom tests shows your deep understanding of their business. TestGen aims to cover about 80% of data quality tests automatically, but the remaining 20%—the unique ones—are often the most valuable and where your team's expertise really matters.

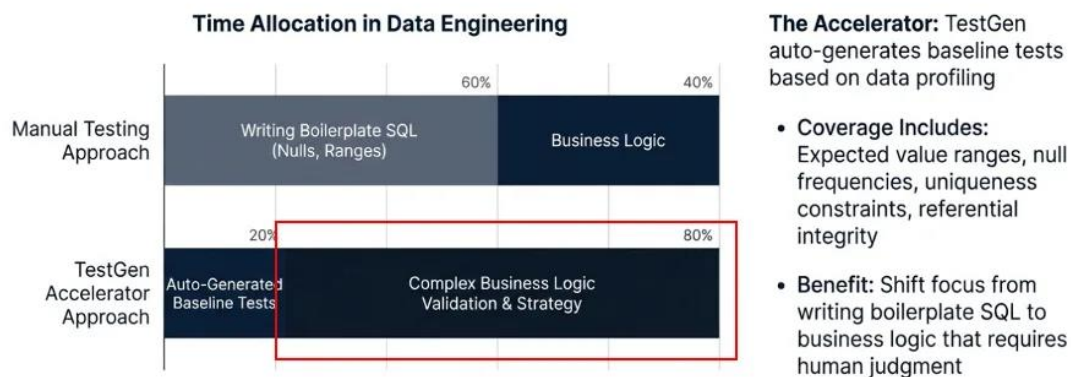
This approach is powerful because it adds value without taking away from existing work. You're not fighting for the same budget as other projects. Instead, you're finding new problems and creating new value. A client who hired you for dashboards might now want data quality help. A client who uses you for pipelines might add observability services. Each new skill opens up more opportunities.

Goal 3: Improve Project Deliverables

Beyond helping you win business, TestGen improves the quality of your technical work. In a competitive market, the quality of your deliverables sets you apart.

Writing tests is one of the most time-consuming parts of building production data pipelines. Sound engineers know that untested code is risky, but the push to deliver features often delays testing. This can lead to systems that work—until they suddenly don't, often at the worst times. TestGen automatically generates a basic set of data tests based on profiling. When you connect it to a data source, it analyzes the actual data and generates tests for expected value ranges, missing-value frequency, uniqueness, referential integrity, and more.

Automated accelerators free up consultant time for complex business logic



Automation test creation doesn't replace careful, domain-specific testing. Instead, it takes care of the basic tests, so you can focus on checking business logic that needs your expertise. The result is [better test coverage](#), delivered more quickly.

Since TestGen can run on unlimited tables with no licensing limits, you can create complete test suites across all your data platforms. Commercial tools that charge per table often limit test coverage. With TestGen, you can test everything—staging tables, transformations, final outputs—without worrying about extra costs for your client.

When you deliver a data pipeline or analytics solution, what sets professional work apart is how robust it is. Amateur work fixes the immediate problem. Professional work solves the problem and also plans for possible future issues.

Adding automated data quality tests and observability tools to your deliverables shows this level of professionalism. When you hand over a pipeline that checks its own health, alerts on problems, and shows data quality metrics, you give your client something they can trust to run smoothly.

Move clients from vague expectations to concrete Service Level Agreements (SLAs).

Professional Delivery = Self-Monitoring Pipelines.

The SLA Definition: What does “Working” mean?

- ✓ Data arrives within the time window.
- ✓ Row counts fall within expected ranges.
- ✓ Quality scores remain > 95.
- ✓ Key metrics do not deviate beyond thresholds.

The Shift:

From Amateur Delivery (“It works until it breaks”) to Professional Delivery (A pipeline that monitors its own health).

Outcome:

Shared expectations protect both the consultant and the client.

When clients are choosing between consultants, showing advanced testing and observability can set you apart. Many consultants only focus on features. By highlighting quality, reliability, and long-term success, you show you’re a partner who thinks beyond just the first delivery.

Goal 4: Improve Staff Skills

If you run a consulting practice with multiple team members, the skills of your staff directly impact your capacity to win and deliver work. Investing in training pays dividends through better delivery, higher client satisfaction, and the ability to command premium rates.

DataKitchen has published detailed, free books on [DataOps](#) and [DataOps change management](#). These resources give your team a solid foundation—not just in using the tools, but in understanding the bigger ideas behind good data operations. For consultants, this education helps in several ways. It builds technical skills, gives you a common language for talking about data challenges with clients, and shows that your team understands industry best practices.



Download

GitHub: DataKitchen
TestGen & DataOps
Observability
(Apache 2.0)



Learn

Free DataOps Books
& Methodology
Guides



Validate

DataOps & Data
Quality Certification
Programs

In addition to books, DataKitchen also offers free online certification programs in [DataOps](#) and [data quality](#). These certifications give your team a clear learning path and help them fully understand key concepts. Certifications are essential for consulting teams. They show outside proof of your team’s skills, give clients confidence in your expertise,

and set a clear standard for what “competent” means in your organization. This makes it easier to assess and grow your talent.

Using these free resources can become part of your company culture. New hires can complete certifications during onboarding. Teams can discuss book concepts in meetings, and delivery teams can apply best practices from these resources to client projects. Investing in skills pays off over time. Teams that keep learning and improving deliver better results, make clients happier, and earn more referrals and repeat business.

Enterprise software sales cycles are often long and unpredictable. If you recommend a tool that is a big investment, you add extra steps that you can’t control. Budget approvals can take months, procurement adds complexity, and other priorities can stop deals altogether.

TestGen’s Apache-2.0 license is ideal for consulting projects. It allows commercial use with no restrictions, lets you modify it without sharing changes, and you can distribute it without paying royalties. For clients, this means they can use TestGen across their whole organization, add it to their products, and build on it without legal headaches.

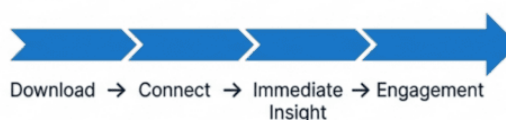
Open source eliminates the friction of procurement to accelerate time-to-value.

Traditional Enterprise Sales Model



Pain Point: Clients balk at six-figure price tags before value is demonstrated.
Result: Stalled deals.

The DataKitchen Open Source Model



Benefit: The only investment is consulting time.
Economics: Unlimited tables, no ‘per-table’ tax.

For consultants, this means you can recommend and use TestGen without worrying about license compliance. There’s no user counting, no usage tracking, and no risk of audits. You can build your own features on top, create custom add-ons for clients, and make it part of your regular toolkit.

A major economic benefit of TestGen is that it supports an unlimited number of tables. Most commercial data quality tools charge by table, data source, or data volume. These pricing models force you to choose between full coverage and saving money. They are too expensive.

With TestGen, you don’t have to make that tradeoff. You can profile and test every table in a client’s data warehouse at no extra cost. This lets you build truly comprehensive data quality programs rather than cutting corners due to budget constraints.

Because TestGen runs entirely in-house, it solves a common concern with data tools. Many organizations, especially in regulated industries, can’t send data to outside services. Even those without strict rules are often careful about letting data leave their control.

TestGen avoids this issue altogether. You install it in the client’s environment, it checks the data in place, and the results stay local. There’s no SaaS, no cloud dependency, and no risk of data leaving. This makes it a good fit for clients who wouldn’t approve a cloud-based data quality tool.

Most importantly, recommending open-source tools aligns your interests with your clients'. When they are ready to run production testing with multiple users, DataKitchen offers a low-cost enterprise version. DataKitchen has excellent support from experienced data people like you.

Clients notice and value this alignment. It builds trust, which is key to long-term consulting relationships.

Your Next Step

Begin by learning the tools yourself. Install TestGen, connect it to some data you can access, and see what it finds. Get to know its profiling, test generation, and reporting features. The better you know the tools, the more confidently you can show them to clients.

Next, look for opportunities with your current clients. Are there accounts with data quality issues you haven't tackled? Which clients could use a data health assessment? Which projects could benefit from better testing and observability? You probably have chances you haven't noticed yet.

Operationalize the "Frictionless Value" framework in four steps.



Then, define your service. How will you present data quality assessments? What's the scope, timeline, and price? What will you deliver? A clear offering makes it easier to talk with prospects and clients.

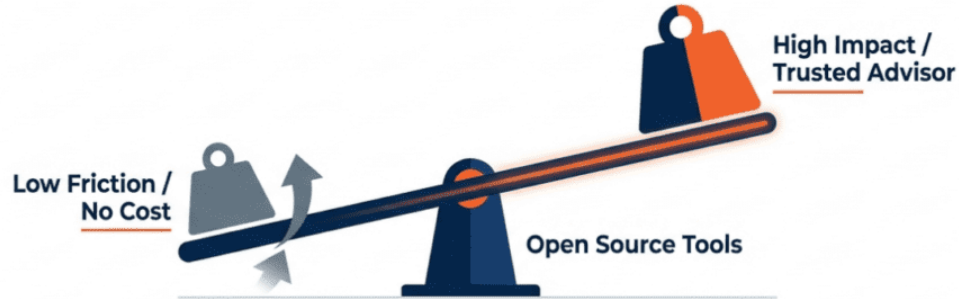
Finally, invest in your team's skills. Share the free DataKitchen resources with them. Encourage them to complete certifications. Make data quality and observability a core strength of your practice.

Conclusion

Consulting is all about creating value for clients and earning a share of that value as fees. The better you are at spotting client problems, showing solutions, and delivering results, the more successful your business will be.

DataKitchen's open-source data quality and observability tools give consultants strong capabilities at every stage of the client relationship. They help you win new clients by showing value right away, deepen relationships by uncovering new problems, and improve your deliverables with automated testing and advanced monitoring. The free educational resources also help your team grow over time.

Open Source Data Quality transforms engagements from "Time & Materials" to "Value-Based" partnerships.



- **Acquire:** Prove value before the contract.
- **Deepen:** Uncover new work continuously.
- **Deliver:** Automate quality for higher margins.

The tools are free; the revenue opportunity lies in the expertise you wrap around them.

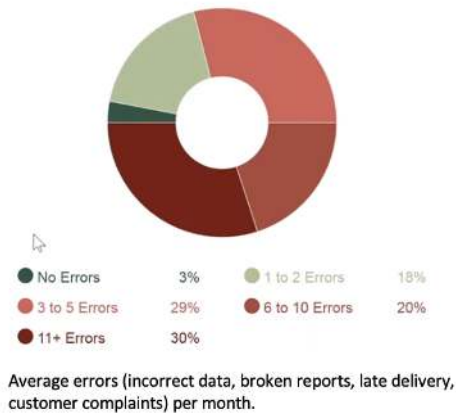
TestGen helps data consultants and engineers by providing a low-friction, automated way to identify, measure, and resolve data quality issues while simultaneously acting as a business development engine.

Author: Chris Bergh | Published: Jan 18, 2026

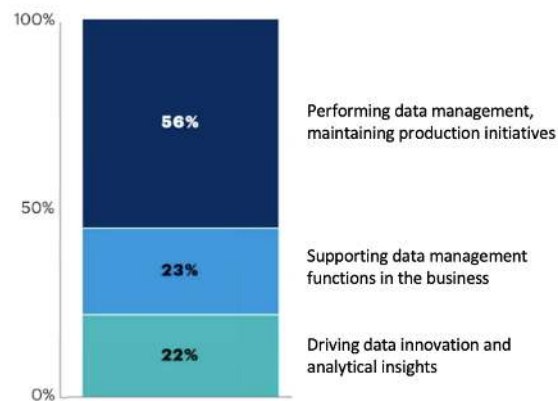
Original article: <https://datakitchen.io/the-consultants-growth-playbook-accelerating-client-acquisition-and-retention-with-datakitchen-open-source-testgen/>

DataOps Observability: Taming the Chaos

2019 Eckerson Survey
Errors per Month, Medium to Large Companies



2020 Gartner Survey
Proportion of Time Spent on Digital Management Tasks



Jason is the CDO at Company X, responsible for six data professional teams across several locations. The breadth of his team's work and the technologies they use present a significant challenge to his main goal: to deliver new and useful analytics solutions to the business. He has 1000 data pipelines running with 10,000 users relying on the data insights they produce—and no way to know if they are working properly. And, his teams are overworked maintaining these systems and firefighting; they cannot deliver on new requests.

Like Jason, you have a significant investment in your data and the infrastructure and tools your teams use to create value. Do you know for sure that it's all working properly, or do you hope and pray that a source change, code fix, or new integration won't break things? If something does break, can you find and fix the problem quickly, or does your team spend days diagnosing the issue? Do you fear that phone call or email from an angry customer who finds the problem before you do? How can you be confident that nothing will go wrong, and your customers will continue to trust your deliverables?

DataOps Observability can help you ensure that your complex data pipelines and processes are accurate and that they deliver as designed. Observability also validates that your data transformations, models, and reports are performing as expected. This solution sits on top of your existing infrastructure—without replacing staff or systems—to monitor your data operations. And it can alert you to problems before anyone else sees them.

Errors Happen; Do You React or Prevent?

Sure enough, Jason gets the call he's been dreading, but it's worse than ever! This call is from the CEO—his boss's boss—and it's only 7:30 a.m. He breaks into a sweat and answers the phone. Through the yelling, he learns that a compliance report, sent to critical business partners, was empty! He makes promises and cancels the rest of his day. With 26 of his best engineers on a call, they begin troubleshooting. At 1:00 p.m., the ad hoc team has the root cause: a change to a source file passed a blank field through the pipeline. At 2:45, they have a fix implemented on one machine, but it will take six weeks to get that fix deployed to the production pipeline. At 3:00, he takes a deep breath and calls the CEO to break the news.

Sound familiar? When was the last time this happened to you? Today's data and analytic systems are complex, made up of any number of disparate tools and data stores. Crisis happens, errors are inevitable. They cause huge embarrassment and affect business performance.

Jason's experience is not a unique one. A 2019 DataKitchen/Eckerson survey found that 79% of companies have more than three data-related errors in their pipelines per month. A 2020 Gartner survey revealed that 56% of engineer time is spent on operational tasks and addressing errors, not on innovations and customer requests.

Many companies are stuck in a culture of "hope and pray" that their changes and integrations won't break anything and a practice of "firefighting" when things inevitably go wrong. They wait for customers to find problems. They blindly trust their providers to deliver good, consistent data on time. They interrupt the daily work of their best minds to chase and fix a single error in a specific pipeline, all the while not knowing if any of the other thousands of pipelines are failing.

Bad data reaches the customer because companies haven't invested enough, or at all, in testing, automation, and monitoring. They have no way to catch errors early. And with engineers jumping around to put out the fires, their already demanding workloads are deprioritized. Customer delivery dates are missed, new features cannot be integrated, and innovations stop coming. As customer confidence in the data and analytics decreases, so does customer satisfaction.

Another result is rampant frustration within data teams. A survey of data engineers conducted by DataKitchen in 2022 revealed some shocking statistics. Data engineers are the people building pipelines, as well as doing data processing and data production. And they are suffering: 78% feel they need to see a therapist, and a similar number have considered quitting or switching careers. The industry is experiencing a shortage of people who do data work due to a supply problem as well as the exodus of people leaving the field because it's so stressful.

Why Haven't They Solved It?

Jason closely monitors the progress of the production fix and updates the CEO at every stage. Naturally, the CEO wants to know why it takes so long to resolve the issue. "It's just code; it's what you guys do every day! Can't you patch it or something?" When the fix is fully tested and deployed to the production pipeline, Jason has time to reflect. He'll conduct a post-mortem with his team on the specific issue, but he wants to find a solution to the bigger problem. Why has the company never addressed this kind of risk? What can he introduce into his data infrastructure and development cycle to ensure these things don't happen again?

You're just too busy. You'll get to it next year...right? The situation is so bad that your team is stressed, and things are breaking left and right. So, why hasn't your company—and many others—solved these problems? There are several reasons.

Teams are obviously very busy. They have backpacks full of customer requests, and they go to work every day knowing that they won't and can't meet customer expectations. The work they complete may sit for a while because any change to the complex architecture or fragile pipeline could break something. When a change finally deploys, the teams have no way to see the whole landscape to ensure everything continues to run smoothly. Even if they could look at everything across their complex data operations, they don't know what or where to check for dependencies, failures, and data inconsistencies.

Obstacles to Fixing the Problems in Your Data Operations	
No time and resources	Teams are already busy and stressed, and are not meeting customer expectations.
Low appetite for change	Teams have complicated in-place data architectures and tools, and they fear changes to what is already running.
No single pane of glass	Teams have no ability to see across all tools, pipelines, jobs, processes, datasets, and people.
No insight into testing	Teams don't know what, where, or how to check operations to ensure that the outputs are right.
Lots of blame and shame	Teams are panicked when customers find problems and spend time, without a shared context, trying to find out who is responsible.

Teams have complicated in-place data architectures and tools, and they fear changes to what is already running.

Teams are panicked when customers find problems and spend time, without a shared context, trying to find out who is responsible.

And then there are the considerable numbers of teams and data processes involved. When something goes wrong, there is a lot of finger-pointing and blame to go around. Each team knows its small part of the landscape, but there is no shared context among them of the larger picture—its infrastructure and problem areas—to adequately and efficiently address the issues.

It's a big risk if your teams can't answer basic questions about production processes, or if the answers are scattered across pockets of your organization, with teams that have no way to communicate broadly. Do they know whether the data will arrive on time? Do they know whether the integrations with other datasets are correct? Without a single source of truth about what's running, how can you pinpoint the failures? How can you find problems before your customers do?

There's a similar risk when teams do not understand the status of development processes and deployments to production. Do your teams know how and when deployments are happening? Do they know what processes and environments are affected by a deployment? Do they know what changes are going into production? Not knowing these answers makes it very hard to ensure that everything is going well and even harder to find problems when they occur.

The industry is replete with data tools, automation tools, and IT monitoring tools. But none of these tools fully addresses the problem. You have to monitor your entire data estate and all the reasons why pipelines may succeed or fail, not just server performance, load testing, and tool and transaction testing. A complete Observability solution is required.

Part 2: Introducing Data Journeys

Jason knows he needs to implement changes, but what and where? The more he reads about DataOps, the more he knows his company needs to make the transition. He thinks he can sell his boss and the CEO on this idea, but his pitch won't go over well when they still have more than six major data errors every month. He wonders what it would take to create a dashboard that could monitor his pipelines and alert him to potential problems or track negative trends before he gets the next dreaded call.

When considering how organizations handle serious risk, you could look to NASA. The space agency created and still uses "mission control," where many screens share detailed data about all aspects of a space flight. That shared information is the basis for monitoring mission status, making decisions and changes, and then communicating to all people involved. It is the context for people to understand what's going on in the moment and to review later for improvements and root cause analysis.

The goal of DataOps Observability is to provide visibility of every journey that data takes from source to customer value across every tool, environment, data store, data and analytics team, and customer so that problems are detected, localized, and raised immediately.

Data journey observability is the first step in implementing DataOps. It tackles the immediate challenges in your data operations by providing detailed information about what's going on right now.

Jason considers his dashboard idea but quickly realizes the complexity of building such a system. It's not just a fear of change. It's because it's a hard thing to accomplish when there are so many teams, locales, data sources, pipelines, dependencies, data transformations, models, visualizations, tests, internal customers, and external customers. "Too many moving parts," he thinks, "but there has to be some way to prove things work before my customers see them!"

And when these stakeholders can see a shared view of a single pipeline, they know everything that's happening during a run of that pipeline. The journey reveals all of the complex steps, toolchains, and actions within the data workflow—most importantly, where things go wrong.

The journey tracks all levels of the stack from data to tools to code to tests across all critical dimensions. It supplies real-time statuses and alerts on start times, processing durations, test results, and infrastructure events, among other metrics. And if you're armed with this information, you can know if everything ran on time and without errors and immediately identify the specific parts that didn't.

As valuable as this visibility is to this single pipeline and its users, most organizations have large numbers of pipelines that need this level of observation.

A Journey Maps and Observes Everything

Every data operation is a kind of factory, some with hundreds of pipelines, each made up of a combination of tools, technologies, and data stores. As a result, there are thousands of tools, and each tool has code that's acting on the data in some way.

Even more challenging is the fact that these pipelines are anything but uniform. Some are batched, some are streaming, some are scheduled or triggered from a dependency, while some are entirely manual. And they all may have different deployment methods and environments. Further, the pipelines are owned and operated by very different teams in different departments within your company.

The good news is that a single data journey can span multiple, related pipelines to track all of the data and analytic infrastructure within them.

A data journey can "see" all of this because it is a meta-structure that goes beyond typical test automation, application performance monitoring, and IT infrastructure monitoring software. While quite valuable, these solutions all produce lagging indicators. With these tools you may know that you are approaching limits on disk space, but you can't know if the data on that disk is correct. You may know that a particular process has completed but you can't know if it completed on time or with the correct output. You can't quality-control your data integrations or reports with only some of the details.

Since data errors happen with more frequency than resource failures, data journeys provide crucial additional context for pipeline jobs and tools and the products they produce. They observe and collect information, then synthesize it into coherent views, alerts, and analytics for people to predict, prevent, and react to problems.

No Journey Exists in a Vacuum

Jason takes a well-deserved break and meets up with his colleague Maria, Manager of Data Science, at the local coffee shop. He decides to run his data journey map idea by his friend. Maria listens carefully, nodding in agreement, then asks a question. "Will this help me know when my scientists are going to get new feeds of sales data from your engineers?" Jason frowns. What data feed? How did he not know about this downstream use?

This lack of visibility is why data journeys track and collect information across all levels of your organization. Starting small, they simplify the complex steps within pipelines. You can define journeys to represent the process "chunks" you truly care about, ignoring the noise of more granular details. Then expanding their scope, journeys also represent and track complex relationships among all your pipelines where no connections are currently coded. To accomplish this, you can set up relationships among the journeys themselves.

Imagine being able to pull together contexts from across the organization where data is shared or processes are dependent on inputs and transformations.

So often these relationships among data pipelines are tribal knowledge, rather than concrete and actionable connections. But when there's a problem, you need to know how these relationships could amplify it.

By identifying how pipelines actually connect via data dependencies and relate through shared data use across your data estate, you can build relationships among the representational journeys to reflect them.

Part 3: Considering the Elements of Data Journeys **Constructing Effective Journeys**

As he thinks through the various journeys that data take in his company, Jason sees that his dashboard idea would require extracting or testing for events along the way. So, the only way for a data journey to truly observe what's actually happening is to get his tools and pipelines to auto-report events.

With this information in a shared context, your analyst working on a data lake will know if the 15 datasets she is viewing are accurate, the most recent, or of the same date range. And she'll know when newer data will arrive. She can work more efficiently knowing when to conduct her analyses and what delivery date to communicate to her customers.

With this actionable data combined with an understanding of the bigger context of your data estate, you can empower your employees to stop production pipelines before problems escalate.

You want to track more than the schedule and duration of your runs. You want to check that the contents of your pipelines—the data, the models, the integrations, the reports, the outputs that your customers see—are as accurate, complete, and up to date as expected. You want to find errors early. To do that, your journeys need to include automated tests.

Tests that evaluate your data and its artifacts work by checking things like data inputs, transformation results, model predictions, and report consistency. They run the gamut from typical software development tests (unit, functional, regression tests, etc.) to custom data tests (location balance, historical balance, data conformity, data consistency, business logic tests, and statistical process control). Read more about DataOps testing in [A Guide to DataOps Tests](#) or [Add DataOps Tests for Error-Free Analytics](#).

Your organization already has many sources of tests, which can transmit vital information to data journeys with rules for taking action. You may have a database that has its own test framework, your data engineers may have written some SQL tests, your data scientists may have written Python tests, you may have test engineers writing Selenium test suites, you may be using DataKitchen DataOps Automation for testing. All usable information.

Together, the journeys, the expectations, the data store, the event engine, the rules and alerts, the integrated tools, the tests, and a robust interface all enable complete observability. All of these elements come together to inform you of where the problems are, minimize the risk to customer value, provide a shared context for all constituents to know what's happening and what's going to happen, and alert the right people when something goes wrong.

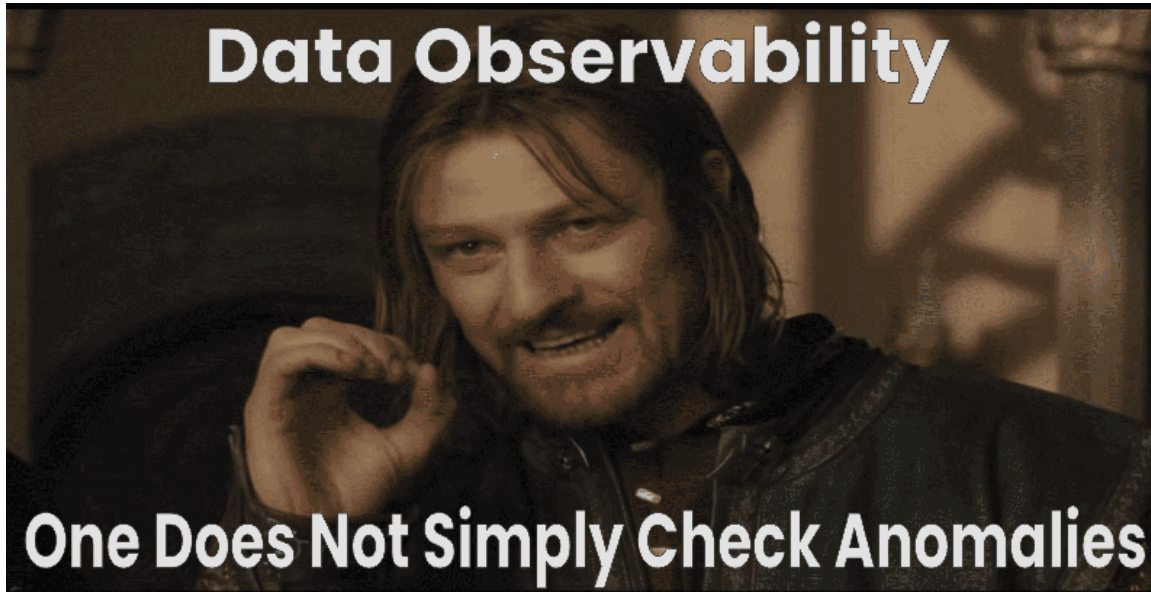
Part 4: Reviewing the Benefits

In a world of complexity, failure, and frustration, data and analytics teams need to deliver insight to their customers with no errors and a high rate of change. As in Jason's experience, the stress and embarrassment of breaking things crowd out the ability to create new insights. In a 2021 survey of 600 data professionals, responses suggested an overwhelming majority are calling for relief. In the survey, 97% reported experiencing burnout, 91% reported frequent requests for analytics with unrealistic or unreasonable expectations, and 87% reported getting blamed when things go wrong.

Spend less time worrying about what may go wrong and gain more time to create by observing the entire data journey and taking early action to stay on track.

The Five Use Cases in Data Observability: Overview

Data observability extends beyond simple anomaly checking, offering deep insights into data health, dependencies, and the performance of data-intensive applications. This blog post introduces five critical use cases for data observability, each pivotal in maintaining the integrity and usability of data throughout its journey in any enterprise.



The ability to monitor, validate, and ensure data accuracy across its lifecycle is not just a luxury—it's a necessity. Data observability extends beyond simple anomaly checking, offering deep insights into data health, dependencies, and the performance of data-intensive applications. This blog post introduces five critical use cases for data observability, each pivotal in maintaining the integrity and usability of data throughout its journey in any enterprise.

Use Cases: Why Should You Care?

	Data Observability Use Case	Benefits of Fixing
Data Evaluation:	New data set evaluation and cleansing	Save time, lower errors, improve data quality
Data Ingestion:	Updates to existing data sources; Data ingestion monitoring	Find problem data quickly, save time, lower errors
Production:	Monitoring of multi-tool, multi-data sets, multi-hop, data analytic production processes.	Stop embarrassing customer errors, gain customer data trust, lower errors, improve team productivity
Development:	Development Unit, Regression Tests, and Impact Assessment.	Improve the speed and lower the risk of changes to production, less wasted time, improve productivity
Data Migration	Checking data accuracy during Data Migration projects: 'Does It Match'?	Save rework, improve schedule accuracy, productivity

Before new data sets are introduced into production environments, they must be thoroughly evaluated and cleaned. This initial stage of data observability ensures that data quality is maintained from the start, preventing errors that

could affect downstream processes and decisions. This use case is vital for organizations that rely on accurate data to drive business operations and strategic decisions.

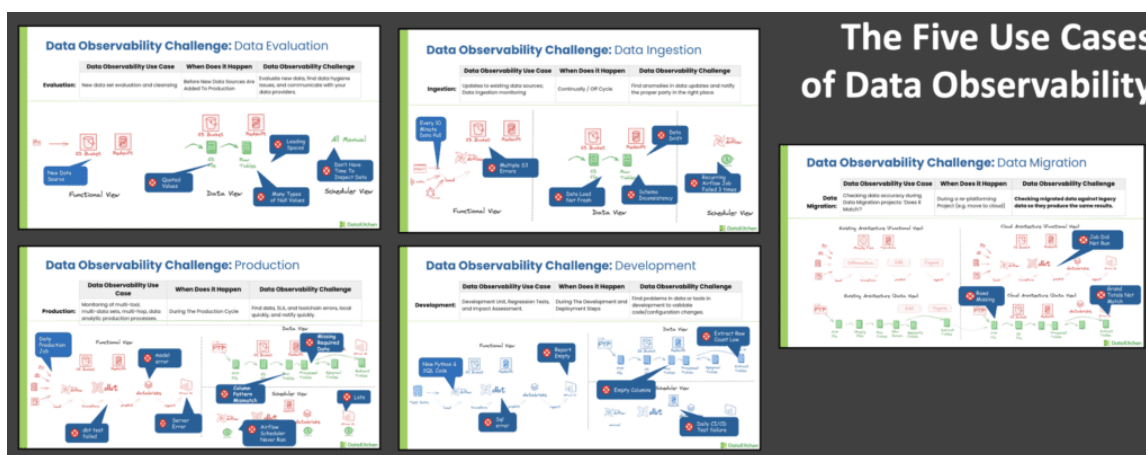
Continuous monitoring during data ingestion ensures that updates to existing data sources are accurate and consistent. Finding Anomalies in regular tasks, such as loading CRM data, is critical to catching issues early. Data ingestion observability helps organizations avoid costly mistakes when poor-quality data is used in critical business processes.

During the production cycle, it's crucial to oversee processes involving multiple tools and datasets, such as dashboard production or warehouse building. Observability in this phase ensures that all components function correctly and that the correct data is delivered to end-users, maintaining the integrity of delivered insights and operational outputs.

The development phase involves integrating new code, tools, or configurations. Data observability here includes conducting regression tests and assessing the impact of these changes. This ensures that any new integration or code alteration does not adversely affect the existing systems, thus safeguarding the overall system integrity.

Data migration, especially during projects like cloud transitions, requires meticulous checking to ensure that migrated data matches the legacy data in terms of output and functionality. Observability during migration verifies data accuracy and functionality in the new environment, ensuring seamless transitions with minimal data loss or corruption risk.

Each use case presents unique challenges and requires specific strategies to ensure data health. By [following our detailed blog posts](#) on each use case, readers can better understand how to implement effective observability strategies tailored to different data lifecycle stages. These posts will explore practical approaches, tools, and tips for enhancing data observability in your operations. We invite you to explore each of these use cases in further detail in our subsequent posts, where we will provide comprehensive insights and actionable advice on implementing data observability effectively. Stay tuned to transform your data management practices with enhanced observability!



Author: Chris Bergh | Published: May 10, 2024

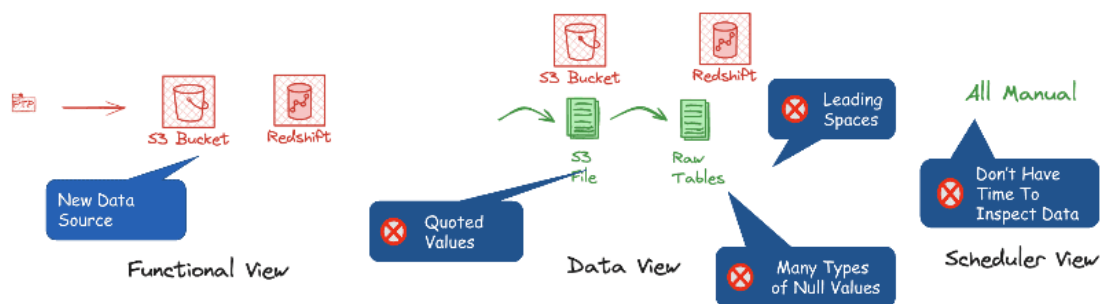
Original article: <https://datakitchen.io/the-five-use-cases-in-data-observability-overview/>

The Five Use Cases in Data Observability: Data Quality in New Data Sources

Chris Bergh CEO, Head Chef Chris is the CEO and Head Chef at DataKitchen. He is a leader of the DataOps movement and is the co-author of the DataOps Cookbook and the DataOps Manifesto. See Full Bio

Data Observability Challenge: Data Evaluation

	Data Observability Use Case	When Does it Happen	Data Observability Challenge
Evaluation:	New data set evaluation and cleansing	Before New Data Sources Are Added To Production	Evaluate new data, find data hygiene issues, and communicate with your data providers.



Ensuring their quality and integrity before incorporating new data sources into production is paramount. Data evaluation serves as a safeguard, ensuring that only cleansed and reliable data makes its way into your systems, thus maintaining the overall health of your data ecosystem. When looking at new data, does one patch the data? Or push back on the data provider to improve the data itself? And how can a data engineer give their provider a 'score' on the data based on fact?

Data Evaluation: This involves evaluating and cleansing new datasets before being added to production. This process is critical as it ensures data quality from the onset.

Data Ingestion: Continuous monitoring of data ingestion ensures that updates to existing data sources are consistent and accurate. Examples include regular loading of CRM data and anomaly detection.

Production: During the production cycle, oversee multi-tool and multi-data set processes, such as dashboard production and warehouse building, ensuring that all components function correctly and the correct data is delivered to your customers.

Development: Observability in development includes conducting regression tests and impact assessments when new code, tools, or configurations are introduced, helping maintain system integrity as new code or data sets are introduced into production.

Data Migration: This use case focuses on verifying data accuracy during migration projects, such as cloud transitions, to ensure that migrated data matches the legacy data regarding output and functionality.

Adding new data sets to production environments without proper evaluation can lead to significant issues, such as data corruption, analytics based on faulty data, and decisions that may harm the business. To avoid these pitfalls, it is

crucial to assess new data sources meticulously for hygiene and consistency before they are integrated into live environments.

Author: Chris Bergh | Published: May 10, 2024

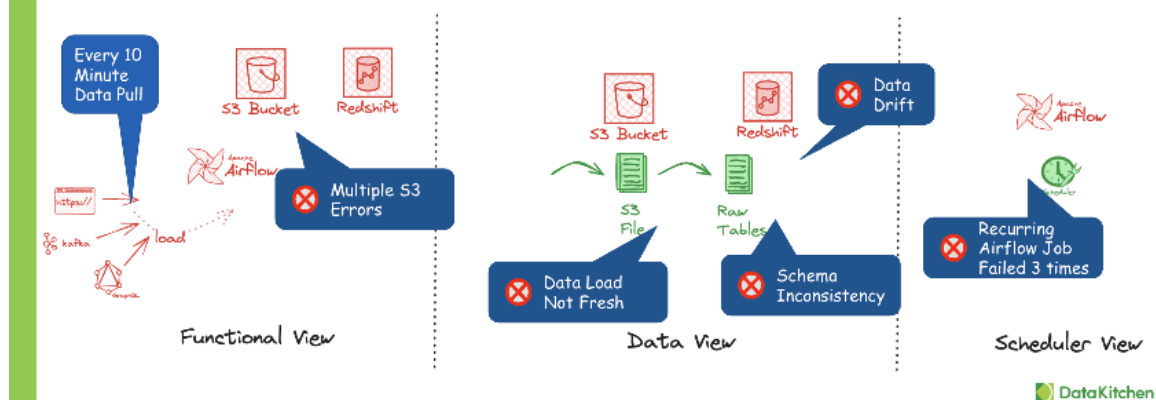
Original article: <https://datakitchen.io/the-five-use-cases-in-data-observability-part-1/>

The Five Use Cases in Data Observability: Effective Data Anomaly Monitoring

Chris Bergh CEO, Head Chef Chris is the CEO and Head Chef at DataKitchen. He is a leader of the DataOps movement and is the co-author of the DataOps Cookbook and the DataOps Manifesto. See Full Bio

Data Observability Challenge: Data Ingestion

	Data Observability Use Case	When Does it Happen	Data Observability Challenge
Ingestion:	Updates to existing data sources; Data ingestion monitoring	Continually / Off Cycle	Find anomalies in data updates and notify the proper party in the right place.



Data Evaluation: This involves evaluating and cleansing new datasets before being added to production. This process is critical as it ensures data quality from the onset.

Data Ingestion: Continuous monitoring of data ingestion ensures that updates to existing data sources are consistent and accurate. Examples include regular loading of CRM data and anomaly detection.

Production: During the production cycle, oversee multi-tool and multi-data set processes, such as dashboard production and warehouse building, ensuring that all components function correctly and the correct data is delivered to your customers.

Development: Observability in development includes conducting regression tests and impact assessments when new code, tools, or configurations are introduced, helping maintain system integrity as new code of data sets are introduced into production.

Data Migration: This use case focuses on verifying data accuracy during migration projects, such as cloud transitions, to ensure that migrated data matches the legacy data regarding output and functionality.

Data ingestion refers to transporting data from various sources into a system where users can store, analyze, and access it. This process must be continually monitored to detect and address any potential anomalies. These anomalies can include delays in data arrival, unexpected changes in data volume or format, and errors in data loading.

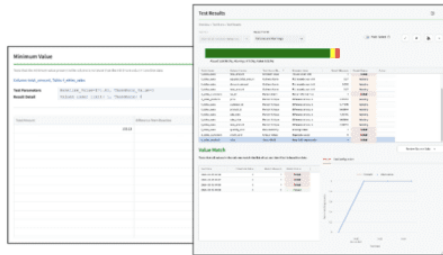
For instance, consider a typical scenario where CRM data is loaded every 10 minutes or comprehensive data change captures (CDC) that track and integrate data updates. Any disruptions in these processes can lead to significant data quality issues and operational inefficiencies. The key challenges here involve:

Effective data ingestion anomaly monitoring should address several critical questions to ensure data integrity:

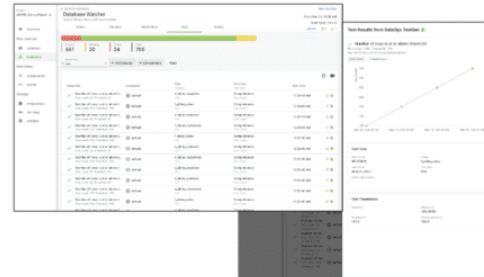
Tools for Data Ingestion Observability

	Data Observability Use Case	When Does it Happen	Data Observability Solution
Ingestion:	Updates to existing data sources; Data ingestion monitoring	Continually / Off Cycle	Find anomalies in data updates and notify the proper party in the right place.

DataOps TestGen Data Freshness, Volume, Schema, And Data Drift Anomaly Detection Test



DataOps Observability Anomaly Alerts



Author: Chris Bergh | Published: May 10, 2024

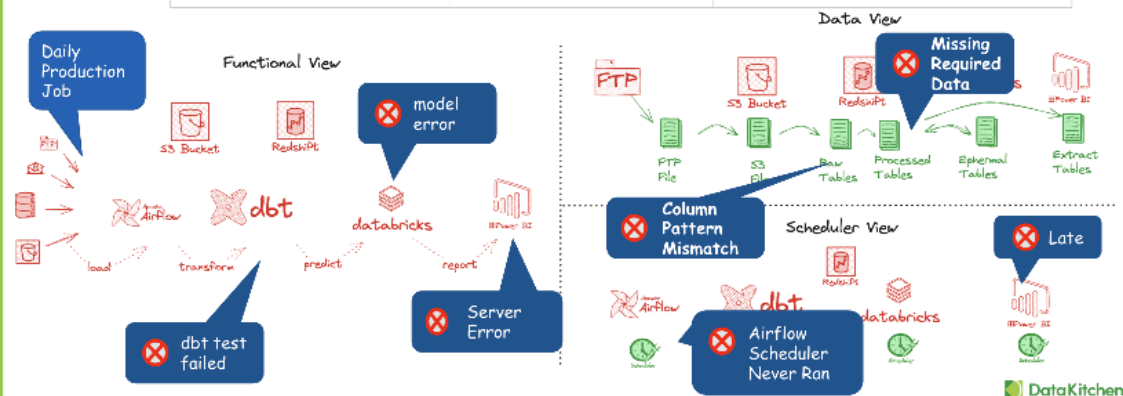
Original article: <https://datakitchen.io/the-five-use-cases-in-data-observability-part-2/>

The Five Use Cases in Data Observability: Mastering Data Production

Chris Bergh CEO, Head Chef Chris is the CEO and Head Chef at DataKitchen. He is a leader of the DataOps movement and is the co-author of the DataOps Cookbook and the DataOps Manifesto. See Full Bio

Data Observability Challenge: Production

	Data Observability Use Case	When Does it Happen	Data Observability Challenge
Production:	Monitoring of multi-tool, multi-data sets, multi-hop, data analytic production processes.	During The Production Cycle	Find data, SLA, and toolchain errors, local quickly, and notify quickly.



Managing the production phase of data analytics is a daunting challenge. Overseeing multi-tool, multi-dataset, and multi-hop data processes ensures high-quality outputs. This blog explores the third of five critical use cases for Data Observability and Quality Validation—data Production—highlighting how DataKitchen’s Open-Source Data Observability solutions empower organizations to manage this critical stage effectively.

Data Evaluation: This involves evaluating and cleansing new datasets before being added to production. This process is critical as it ensures data quality from the onset.

Data Ingestion: Continuous monitoring of data ingestion ensures that updates to existing data sources are consistent and accurate. Examples include regular loading of CRM data and anomaly detection.

Production: During the production cycle, oversee multi-tool and multi-data set processes, such as dashboard production and warehouse building, ensuring that all components function correctly and the correct data is delivered to your customers.

Development: Observability in development includes conducting regression tests and impact assessments when new code, tools, or configurations are introduced, helping maintain system integrity as new code of data sets are introduced into production.

Data Migration: This use case focuses on verifying data accuracy during migration projects, such as cloud transitions, to ensure that migrated data matches the legacy data regarding output and functionality.

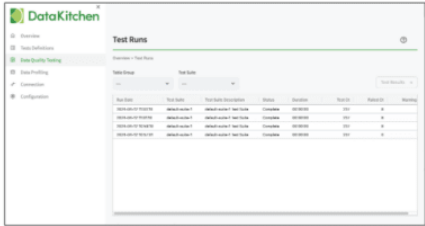
Data production encompasses processing and refining raw data into valuable insights, including dashboard production, warehouse constructions, and model refreshes. During these processes, monitoring and validating data at each step of the production process is vital to detect any discrepancies, errors, or inefficiencies that might compromise the final products. The challenges in data production are multi-faceted:

Effective data observability in production requires answers to several critical questions to ensure data integrity and operational efficiency:

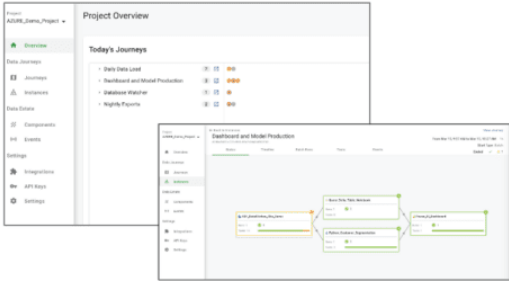
Tools for Production Data Observability

	Data Observability Use Case	When Does it Happen	Data Observability Solution
Production:	Monitoring of multi-tool, multi-data sets, multi-hop, data analytic production processes.	During The Production Cycle	Find datas, SLA, and toolchain errors, local quickly, and notify quickly.

DataOps TestGen Auto Generated Data Quality Validation Test



DataOps Observability Data Journey



Implementing DataKitchen’s observability solutions during the data production phase offers numerous benefits:

Author: Chris Bergh | Published: May 10, 2024
Original article: <https://datakitchen.io/the-five-use-cases-in-data-observability-part-3/>

The Five Use Cases in Data Observability: Fast, Safe Development and Deployment

Chris Bergh CEO, Head Chef Chris is the CEO and Head Chef at DataKitchen. He is a leader of the DataOps movement and is the co-author of the DataOps Cookbook and the DataOps Manifesto. See Full Bio

Data Observability Challenge: Development

	Data Observability Use Case	When Does it Happen	Data Observability Challenge
Development:	Development Unit, Regression Tests, and Impact Assessment.	During The Development and Deployment Steps	Find problems in data or tools in development to validate code/configuration changes.



The integrity and functionality of new code, tools, and configurations during the development and deployment stages are crucial. This blog post delves into the third critical use case for Data Observation and Data Quality Validation: development and Deployment. It highlights how DataKitchen's Data Observation solutions equip organizations to enhance their development practices, reduce deployment risks, and increase overall productivity.

Data Evaluation: This involves evaluating and cleansing new datasets before being added to production. This process is critical as it ensures data quality from the onset.

Data Ingestion: Continuous monitoring of data ingestion ensures that updates to existing data sources are consistent and accurate. Examples include regular loading of CRM data and anomaly detection.

Production: During the production cycle, oversee multi-tool and multi-data set processes, such as dashboard production and warehouse building, ensuring that all components function correctly and the correct data is delivered to your customers.

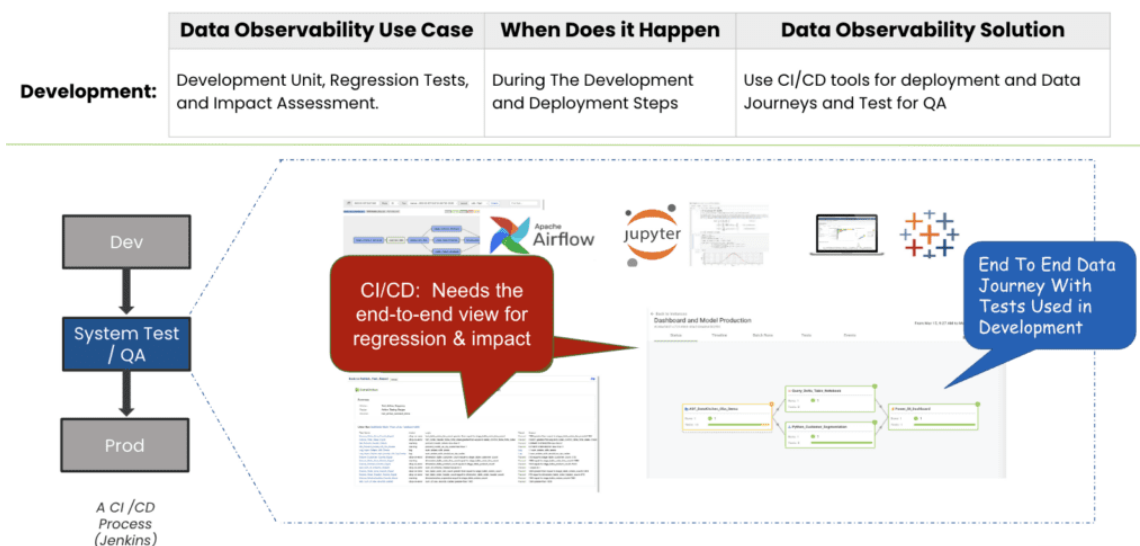
Development: Observability in development includes conducting regression tests and impact assessments when new code, tools, or configurations are introduced, helping maintain system integrity as new code of data sets are introduced into production.

Data Migration: This use case focuses on verifying data accuracy during migration projects, such as cloud transitions, to ensure that migrated data matches the legacy data regarding output and functionality.

The development phase often involves integrating new SQL scripts, Python modules, Yaml configurations, or datasets into existing systems. Each addition or modification poses potential risks that could propagate errors into production environments. The primary challenge is identifying and resolving these issues early in the development cycle to prevent failed deployments and operational disruptions.

A robust data observability strategy addresses several critical questions to ensure the stability and reliability of development activities:

Tools for Regression/Impact Testing



DataKitchen's solution offers an end-to-end Data Journey visualization that covers the complex data estate necessary for thorough development testing. This feature ensures developers have a comprehensive view of how new code or data sets integrate with existing systems, enhancing understanding and reducing integration errors.

Implementing DataKitchen's observability tools during the development and deployment phases brings substantial benefits:

For organizations aiming to enhance their development processes and ensure successful deployments, adopting DataKitchen's Data Observability solutions is an excellent strategy. By integrating comprehensive testing and real-time monitoring into the development lifecycle, companies can prevent costly errors and accelerate their time-to-market for new data sets and features.

Author: Chris Bergh | Published: May 10, 2024

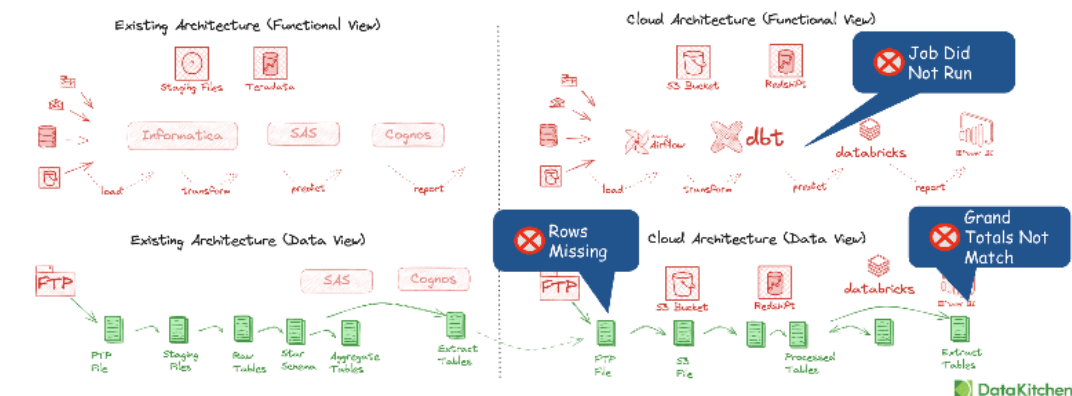
Original article: <https://datakitchen.io/the-five-use-cases-in-data-observability-part-4/>

The Five Use Cases in Data Observability: Ensuring Accuracy in Data Migration

Chris Bergh CEO, Head Chef Chris is the CEO and Head Chef at DataKitchen. He is a leader of the DataOps movement and is the co-author of the DataOps Cookbook and the DataOps Manifesto. See Full Bio

Data Observability Challenge: Data Migration

	Data Observability Use Case	When Does it Happen	Data Observability Challenge
Data Migration:	Checking data accuracy during Data Migration projects: 'Does It Match?'	During a re-platforming Project (e.g. move to cloud)	Checking migrated data against legacy data so they produce the same results.



Data migration projects, such as moving from on-premises infrastructure to the cloud, are critical and complex projects that involve transferring data across different systems while ensuring data integrity and consistency. This blog post explores the fifth use case for Data Observability and Data Quality Validation—data Migration—focusing on how DataKitchen's Open-Source Data Observation software ensures these migrations are successful and error-free.

Data Evaluation: This involves evaluating and cleansing new datasets before being added to production. This process is critical as it ensures data quality from the onset.

Data Ingestion: Continuous monitoring of data ingestion ensures that updates to existing data sources are consistent and accurate. Examples include regular loading of CRM data and anomaly detection.

Production: During the production cycle, oversee multi-tool and multi-data set processes, such as dashboard production and warehouse building, ensuring that all components function correctly and the correct data is delivered to your customers.

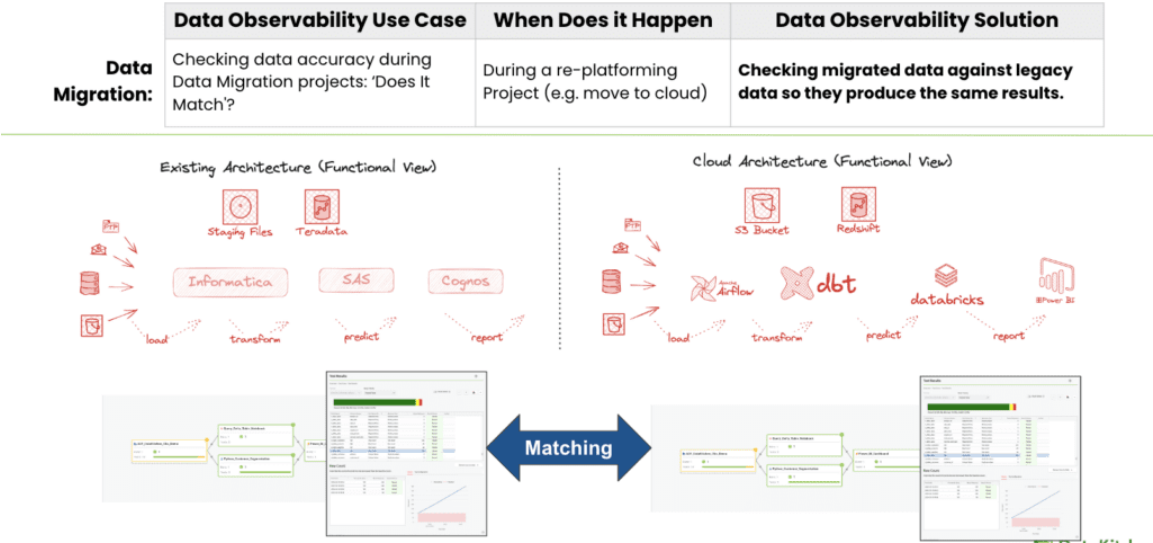
Development: Observability in development includes conducting regression tests and impact assessments when new code, tools, or configurations are introduced, helping maintain system integrity as new code or data sets are introduced into production.

Data Migration: This use case focuses on verifying data accuracy during migration projects, such as cloud transitions, to ensure that migrated data matches the legacy data regarding output and functionality.

Data migration is more than just moving data; it's about ensuring that the migrated data functions identically in a new environment without any loss or corruption. The key challenge in data migration is verifying that the data remains consistent before and after the move. This involves ensuring that:

A thorough data migration strategy must address several critical questions to confirm the success of the migration:

Tools For Data Migration



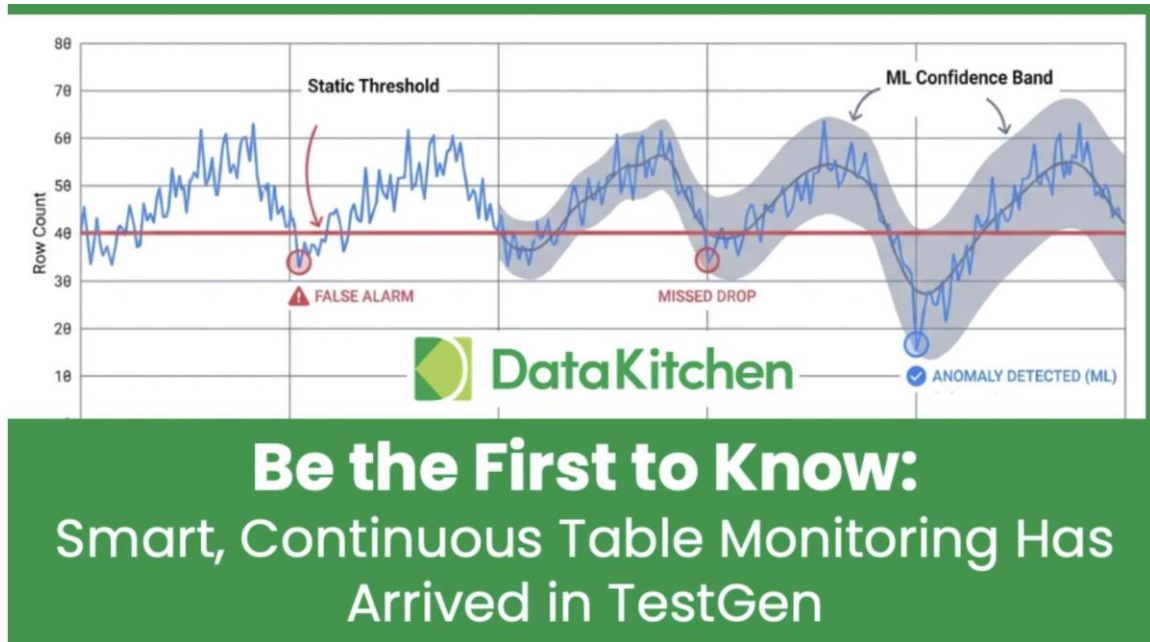
DataKitchen’s Data Observability solutions provide powerful tools to tackle the complexities of data migration:

Data Migration problems can be a thankless project, particularly those that involve moving to a cloud-based environment. With its robust testing and monitoring capabilities, DataKitchen provides the tools to ensure data migrations are successful, accurate, and efficient. By leveraging these advanced observability tools, companies can ensure their data remains robust and reliable, no matter where it resides.

Author: Chris Bergh | Published: May 10, 2024
Original article: <https://datakitchen.io/the-five-use-cases-in-data-observability-part-5/>

Be The First To Know: Smart, Continuous Table Monitoring Has Arrived In TestGen

The days of learning about data problems from an angry business analyst should be over. With TestGen's table monitoring, you're watching everything automatically and continuously.



If you're a data engineer tired of being the last to hear when something goes wrong, this is for you.

You know the feeling: a business stakeholder messages you at 2 PM on a Tuesday saying, "Hey, the revenue numbers look off." You open your laptop and spend the next three hours manually querying tables, checking row counts, and trying to figure out what happened to a pipeline that ingested data six hours earlier. Multiply that by a few incidents a month, and you could be losing upwards of a dozen analyst-hours every week just on detective work. By the time you find the cause—a simple upstream schema change no one told you about—the damage is already done.



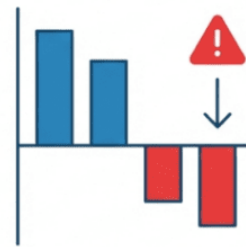
Wasted Time

Dozens of analyst-hours lost weekly on detective work.



The Silent Killer

Upstream schema changes go unnoticed until reports break.



Reputation Hit

You are the "last to know" instead of the "first to fix".

What Is TestGen's New Table Monitoring Feature?

TestGen's monitoring is based on one simple idea: continuous, automated watching of your data tables without needing you to write any SQL, Python, or YAML. After you set up the tables to monitor, TestGen runs regular checks and uses machine learning to spot problems before they turn into disasters.

The Monitor Tables workflow in TestGen helps you start monitoring quickly and easily. Just pick the tables you want to watch, set the schedule, and TestGen handles the rest. You don't have to create thresholds or baseline stats yourself, since TestGen learns your data's normal patterns and spots anything unusual. For teams that want more control, you can also define custom monitors, set manual thresholds, or override the machine learning-based checks for specific cases. This flexibility lets you cover both the everyday and the edge cases, so you get the level of monitoring that fits your needs.

Four dimensions of health are monitored automatically.



Freshness

Is data arriving on time?
Tracks arrival patterns to flag stale tables.



Volume

Are row counts consistent?
Catches duplicate ingestions or partial load failures.



Schema

Did structure change?
Detects renamed columns or deleted fields immediately.



Data Drift

Are statistical properties holding? Checks value distributions for anomalies.

The monitors TestGen generates cover the four dimensions that matter most to data engineers in production:

Freshness — Is your data arriving on time? If a table that normally refreshes every hour suddenly hasn't received new rows in four hours, you want to know immediately. TestGen tracks data arrival patterns and flags when a table goes stale.

Volume — Are the right number of records showing up? Unexpected spikes or drops in row counts are among the most common signals that something went wrong upstream. TestGen determines whether it's a duplicate ingestion, a partial load failure, or a row-count issue. TestGen monitors row count trends and alerts you when volume deviates from expected norms.

Schema — Did someone alter a column without telling you? Schema drift is a silent killer in data pipelines. A column that gets renamed, a data type that gets widened, or a field that quietly disappears can break downstream models and reports. TestGen watches your schemas and catches changes the moment they happen.

Data Drift — Are the statistical properties of your data changing in ways that suggest a quality problem? TestGen looks at the actual distributions and characteristics of column values. If a numeric field that normally has values between 0 and 100 suddenly contains values in the thousands, that's a signal worth investigating. You can add any metric that can be expressed in SQL to monitor for anomalies. Track sales volume, manufacturing output, user conversions, anything your data can tell you about, automatically.

Shift from reactive scripting to automated ML monitoring

TestGen provides continuous, automated watching of data tables without requiring SQL, Python, or YAML scripts. It transforms you from a “Data Detective” into a “Reliability Champion.”



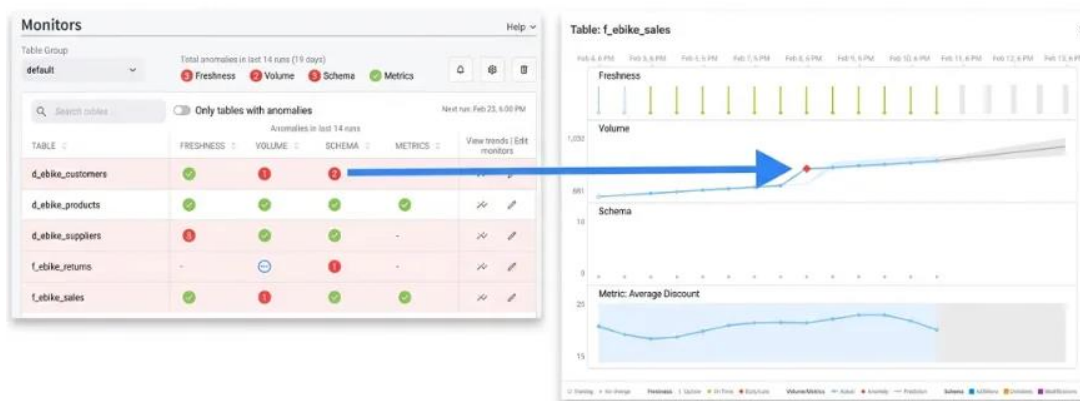
Viewing Monitor Results: A 360° View of Table Health

Once your monitors are running, the View Monitor Results screen gives you a consolidated, actionable view of everything TestGen has observed.

Instead of digging through log files or running random queries to find problems, you get a dedicated UI that shows anomalies in context. The results screen tells you which tables are healthy, which have alerts, and the history of each metric. This is especially helpful when you want to know, “When did this problem actually start?” because the trend data is all right there.

Drill down into anomalies with historical context.

View anomalies alongside table history and schema changes—no digging through log files.



From the monitor results, you can dig into specific anomalies to see how big they are and what caused them. Did row counts drop by 10% or 90%? Did the schema change today or three days ago? This detail helps make monitoring useful rather than noisy alerts you learn to ignore.

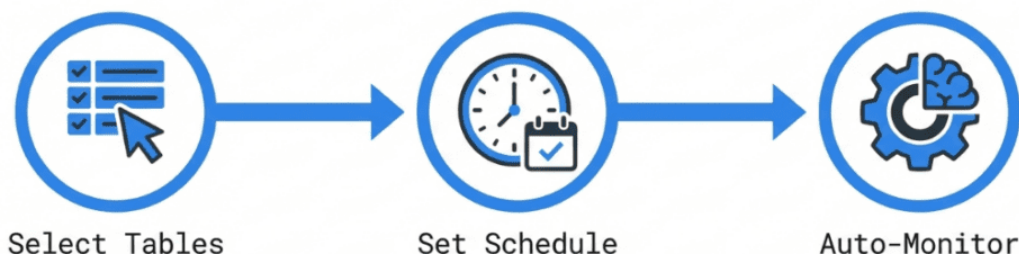
The results also feed into TestGen’s broader quality scoring and dashboard capabilities, so monitoring anomalies don’t live in a silo — they connect to the bigger picture of your data estate’s health.

Why This Matters for Data Engineering Sanity

You no longer have to manually check whether a nightly batch loaded correctly. You won't have to wait for users to tell you a dashboard looks off. You don't need to build and maintain a custom monitoring system or patch together Great Expectations configs and cron jobs. TestGen watches your data so you can focus on building. IT takes minutes to set up, comes with a great UI, and runs on-item for security.

Zero-friction setup with auto-learning capabilities

Simpler than you think. No manual baselines needed.
TestGen learns normal patterns automatically.



The monitoring works inside your database. TestGen runs queries directly there instead of pulling data out. That means no data copies, no security issues, and no unexpected slowdowns.

Since monitoring is integrated with TestGen's profiling, test generation, data catalog, and quality dashboards, an anomaly won't leave you staring at a number alone. You get the full context in one place: the table's history, column details, and quality test results all in one place.

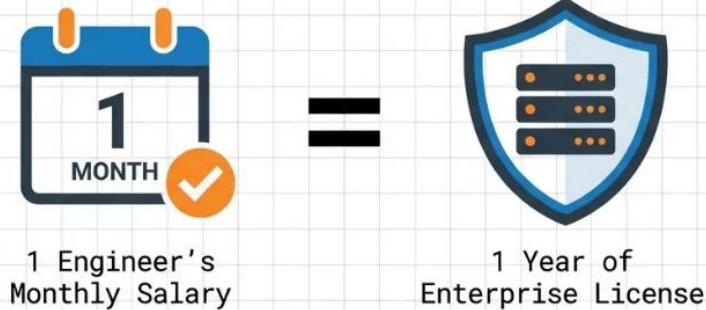
The Pricing Story: Flat \$100 — No Table Tax. Finally, Something That Makes Sense

Here's something refreshing for teams who have checked prices for other data observability tools and felt sticker shock.

DataKitchen's Enterprise version of TestGen uses a flat-rate, predictable pricing model: \$100 per month, per user, per database connection — and that's it. Unlimited tables. Unlimited data volume. All features included.

Flat pricing designed for scale, not exploitation.

\$100/user/month. No “Table Tax.” Monitor every table
in every schema without the bill growing.



This pricing rejects the “per-table tax” common in data observability. Many vendors charge based on the number of tables you monitor, so the more data you want to protect, the more you pay. That forces teams to pick and choose which tables to watch, hoping they’re right. That’s the opposite of how monitoring should work.

With TestGen’s pricing, there’s no reason to leave any tables unmonitored. You can cover your entire data estate — every table in every schema — without your bill growing out of control. A typical mid-size data team pays just a fraction of what similar closed-source, VC-backed platforms charge for the same coverage.

And if you want to start for free? The open-source version of TestGen is Apache 2.0 licensed and fully functional for individual users. You get one user, one database connection, and the complete feature set running locally or in your own Kubernetes or Docker environment. There’s no feature-gated free tier that forces you to upgrade to the paid plan. It’s a genuine open-source product.

[Security and privacy](#) are top priorities for teams adopting new data tools, especially in regulated industries. TestGen securely manages credentials, following best practices for encryption at rest and in transit. Your database credentials are encrypted and never shared outside your configured environment. Since TestGen runs queries directly inside your database and does not move or extract your raw data, your sensitive information stays protected behind your own firewall. TestGen is built with privacy and compliance in mind, making it suitable for environments with strict regulatory requirements. Audit logs and role-based access controls help you monitor and manage who can access monitoring features and data.

From Open Source individual use to Enterprise scale.

Open Source (Free)

- ✓ Fully functional
- ✓ Runs locally / Docker / K8s
- ✓ Apache 2.0 Licensed
- ✓ No feature-gating

Enterprise (\$100/mo)

- ✓ Unlimited Projects
- ✓ Proprietary DB Support
- ✓ Dedicated Support
- ✓ Team Management

Both Versions: No Table Tax.

For teams, the Enterprise tier adds unlimited projects, support for proprietary databases, DataKitchen's dedicated support, and the latest releases — still at that straightforward \$100/user/connection rate.

DataKitchen is a profitable, independent company, which means we are not racing to satisfy VC investors by jacking up prices or sunseting features on a whim. That stability matters when you're building production monitoring infrastructure.

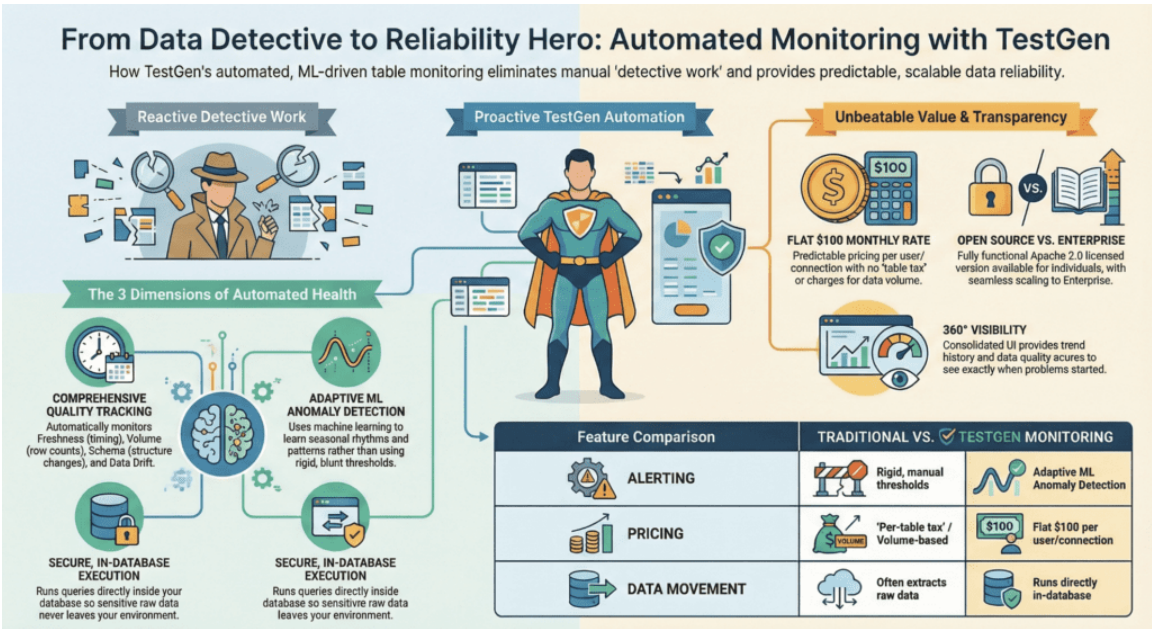
Getting Started

If you're ready to stop being the last to know about data problems, TestGen's monitoring is worth exploring.

Regain your sanity and focus on building.

- ✓ No more waiting for users to find bugs
- ✓ No more patchwork configs of Great Expectations
- ✓ No more pricing uncertainty
- ✓ Integrated profiling, testing, and cataloging

The days of learning about data problems from an angry business analyst should be over. With TestGen's table monitoring, you're watching everything automatically and continuously — all for a price your boss will appreciate.



Author: Chris Bergh | Published: Feb 24, 2026
Original article: <https://datakitchen.io/be-the-first-to-know-smart-continuous-table-monitoring-has-arrived-in-testgen/>