

O'REILLY®

Data Quality Fundamentals

A Practitioner's Guide to Building
More Trustworthy Data Pipelines



**Early
Release**

Raw & Unedited

Compliments of



MONTE CARLO

Barr Moses,
Lior Gavish &
Molly Vorwerck

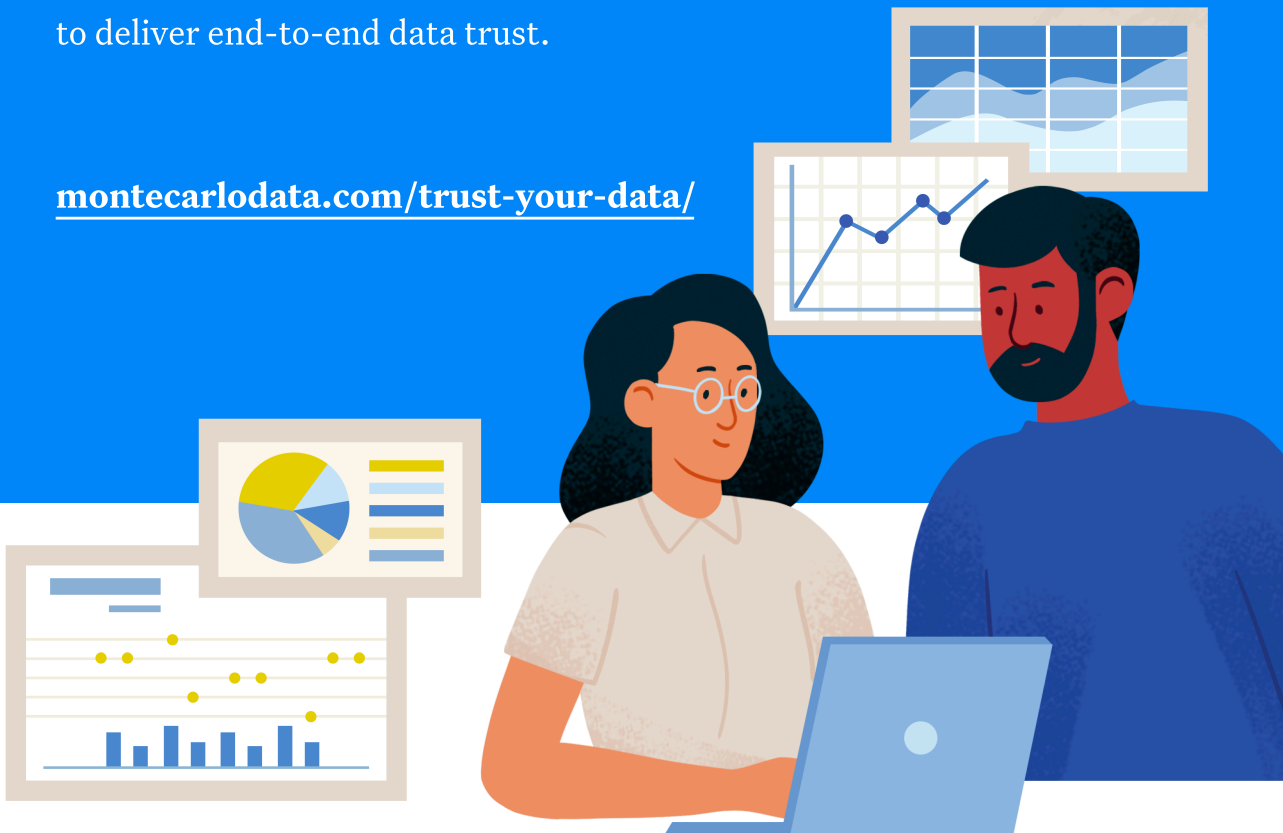
MC

MONTE CARLO

Trust your data.

Learn how modern data teams rely on
Monte Carlo's Data Observability Platform
to deliver end-to-end data trust.

montecarlodata.com/trust-your-data/



Data Quality Fundamentals

*A Practitioner's Guide to Building
More Trustworthy Data Pipelines*

With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

Barr Moses, Lior Gavish, and Molly Vorwerck

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

Data Quality Fundamentals

by Barr Moses, Lior Gavish, and Molly Vorwerck

Copyright © 2022 Monte Carlo Data, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://oreilly.com>). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

Acquisitions Editor: Jessica Haberman

Development Editor: Jill Leonard

Production Editor: Gregory Hyman

Interior Designer: David Futato

Cover Designer: Karen Montgomery

Illustrator: Kate Dullea

September 2022: First Edition

Revision History for the First Edition

2021-10-14: First Release

See <http://oreilly.com/catalog/errata.csp?isbn=9781098112042> for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. Data Quality Fundamentals, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

This work is part of a collaboration between O'Reilly and Monte Carlo Data. See our [statement of editorial independence](#).

The views expressed in this work are those of the author(s), and do not represent the publisher's views. While the publisher and the author(s) have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author(s) disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

Table of Contents

1. Why Data Quality Deserves Attention—Now.....	1
What Is Data Quality?	4
Framing the Current Moment	5
Understanding the “Rise of Data Downtime”	6
Other Industry Trends Contributing to the Current Moment	8
Summary	11
2. Architecting for Data Reliability.....	13
Measuring and Maintaining High Data Reliability at Ingestion	14
Measuring and Maintaining Data Quality in the Pipeline	17
Understanding Data Quality Downstream	20
Building Your Data Platform	22
Data Ingestion	24
Data Storage and Processing	24
Data Transformation and Modeling	25
Business Intelligence and Analytics	25
Data Discovery and Governance	26
Developing Trust in Your Data	27
Data Observability	27
Measure the Cost of Broken Data	28
How to Set SLAs, SLOs, and SLIs for Your Data	31
Case Study: Blinkist	34
Summary	35
3. Prospective Table of Contents (Subject to Change).....	37

Why Data Quality Deserves Attention—Now

A note for Early Release readers

With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the first chapter of the final book. Please note that the GitHub repo will be made active later on.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at jleonard@oreilly.com.

Raise your hand (or spit out your coffee, sigh deeply, and shake your head) if this scenario rings a bell.

Data is a priority for your CEO, as it often is for digital-first companies, and she is fluent in the latest and greatest business intelligence tools. Your CTO is excited about migrating to the cloud, and constantly sends your team articles highlighting performance measurements against some of the latest technologies. Your downstream data consumers including product analysts, marketing leaders, and sales teams rely on data-driven tools like CRMs, CXPs, CMSs, and any other acronym under the sun to do their jobs quickly and effectively.

As the data analyst or engineer responsible for managing this data and making it usable, accessible, and trustworthy, rarely a day goes by without having to field some request from your stakeholders. But what happens when the data is wrong?

Have you ever been about to sign off after a long day running queries or building data pipelines only to get pinged by your Head of Marketing that “the data is missing” from a critical report? What about a frantic email from your CTO about “duplicate data” in a business intelligence dashboard? Or a memo from your CEO, the same one who is so bullish on data, about a confusing or inaccurate number in his latest board deck?

If any of these situations hit home for you, you’re not alone.

This problem, often referred to as “data downtime,” happens to even the most innovative and data-first companies, and, in our opinion, it’s one of the biggest challenges facing businesses in the 21st century. Data downtime refers to periods of time where data is missing, inaccurate, or otherwise erroneous, and it manifests in stale dashboards, inaccurate reports, and even poor decision making.

The root of data downtime? Unreliable data, and lots of it.

Data downtime can cost companies **upwards of millions of dollars per year**, not to mention customer trust. In fact, ZoomInfo found in 2019 that 1 in 5 companies lost a customer due to a data quality issue.

As you’re likely aware, your company’s bottomline isn’t the only thing that’s suffering from data downtime. Handling data quality issues consumes upwards of 40 percent of your team’s time that could otherwise be spent working on more interesting projects or actually innovating for the business.

This statistic probably comes as no surprise. It certainly didn’t to us.

In a former life, Barr Moses, served as VP of Operations at a customer success software company. Her team was responsible for managing reporting for the broader business, from generating dashboards for her CEO to use during All Hands meetings to setting strategy to reduce customer churn based on user metrics. She was responsible for managing her company’s data operations and making sure stakeholders were set up for success when working with data.

Barr will never forget the day she came back to her desk from a grueling, hours-long planning session to find a sticky note with the words “The data is wrong” on her computer monitor. Not only was this revelation embarrassing, but unfortunately, it wasn’t uncommon. Time and again her and her team would encounter these silent and small, but potentially detrimental, issues with their data.

There had to be a better way.

Poor data quality and unreliable data have been a problem for organizations for decades, whether it’s caused by poor reporting, false information, or technical errors. And as organizations increasingly leverage data and build more and more complex data ecosystems and infrastructure, this problem is only slated to increase.

The concept of “bad data” and poor data quality has been around nearly as long as humans have existed, albeit in different forms. With Captain Robert Falcon Scott and other early Antarctic explorers, poor data quality (or rather, data-uninformed decision making) led them to inaccurately forecast where and how long it would take to get to the South Pole, their target destination.

There are several in more recent memory that stick out, too. Take the infamous Mars Climate Orbiter crash in 1999. A NASA space probe, the Mars Climate Orbiter crashed as a result of a data entry error that produced outputs in non-SI (International System) units versus SI units, bringing it too close to the planet. This crash cost NASA a whopping \$125 million dollars. Like spacecraft, analytic pipelines can be extremely vulnerable to the most innocent changes at any stage of the process. Another example comes from the 2008 financial crash, which was spurred in part by inaccurate data that overstated how much mortgage-backed securities and other derivatives.

And this just scratches the surface.

Barr’s unfortunate sticky note incident got her thinking: “I can’t be alone!”

Alongside Lior Gavish, Barr set out to get to the root cause of the “data downtime” issue. Together, they interviewed hundreds of data teams about their biggest problems, and time and again, data quality sprang to the top of the list. From e-commerce to healthcare, companies across industries were facing similar problems: schema changes were causing data pipelines to break, row or column duplicates were surfacing on business critical reports, and data would go missing in dashboards, causing them significant time, money, and resources to fix. We also realized that there needed to be a better way to communicate and address data quality issues as part of an iterative cycle of improving data reliability -- and building a culture around driving data trust.

These conversations inspired us to write this book to convey some of the best practices we’ve learned and developed related to managing data quality at each stage of the data pipeline, from ingestion to analytics, and share how data teams in similar situations may be able to prevent their own data downtime.

For the purpose of this book, “data in production” refers to data from source systems (like CRMs, CSMs, and databases from any of the other analogies previously mentioned) that has been ingested by your warehouse, data lake, or other data storage and processing solutions and flows through your data pipeline (ETL) and surfaced by the analytics layer to business users. Data pipelines can handle both batch and streaming data, and at a high-level, the methods for measuring data quality for either type of asset are much the same.

Data downtime draws corollaries to software engineering and developer operations, a world in which application uptime or downtime (meaning, how frequently your soft-

ware or service was “available” or “up”, or “unavailable” or “down”) is measured scrupulously to ensure that software is accessible and performant. Many site reliability engineers use “uptime” as a measurement because it correlates directly to the impact of poor software performance on the business. In a world where “five nines” (in other words, 99.999% uptime) of reliability is becoming the industry standard, how can we apply this to data?

In this book, we will address how modern data teams can build more resilient technologies, teams, and processes to ensure high data quality and reliability across their organizations.

In Chapter 1, we’ll start by defining what data quality means in the context of this book. Next, we’ll frame the current moment to better understand why data quality is more important for data leaders than ever before. And finally, we’ll take a closer look at how best-in-class teams can achieve high data quality at each stage of the data pipeline and what it takes to maintain data trust at scale. This book focuses primarily on data quality as a function of powering data pipelines and production systems, as opposed to data science platforms or other more research-focused work.

What Is Data Quality?

Data quality as a concept is not novel—“data quality” has been around as long as humans have been collecting data!

Over the past few decades, however, the definition of data quality has started to crystallize as a function of measuring the reliability, completeness, and accuracy of data as it relates to the state of what is being reported on. As they say, you can’t manage what you don’t measure, and high data quality is the first stage of any robust analytics program. Data quality is also an extremely powerful way to understand whether your data fits the needs of your business.

For the purpose of this book, we define data quality as the health of data at any stage in its life cycle. Data quality can be impacted at any stage of the data pipeline, before ingestion, in production, or even during analysis.

In our opinion, data quality frequently gets a bad rep. Data teams know they need to prioritize it, but it doesn’t roll off the tongue the same way “machine learning,” “data science,” or even “analytics does,” and many teams don’t have the bandwidth or resources to bring on someone full-time to manage it. Instead, resource-strapped companies rely on the data analysts and engineers themselves to manage it, diverting them away from projects that are conceived to be more interesting or innovative.

But if you can’t trust the data and the data products it powers, then how can data users trust your team to deliver value? The phrase, “no data is better than bad data” is

one that gets thrown around a lot by professionals in the space, and while it certainly holds merit, this often isn't a reality.

Data quality issues (or, data downtime) are practically unavoidable given the rate of growth and data consumption of most companies. But by understanding how we define data quality, it becomes much easier to measure and prevent it from causing issues downstream.

Framing the Current Moment

Technical teams have been tracking—and seeking to improve—data quality for as long as they've been tracking analytical data, but only in the 2020s has data quality become a top-line priority for many businesses. As data becomes not just an output but a financial commodity for many organizations, it's important that this information can be trusted.

As a result, companies are increasingly treating their data like code, applying frameworks and paradigms long-standard among software engineering teams to their data organizations and architectures. Developer Operations (DevOps), a technical field dedicated to shortening the systems development life cycle, spawned industry-leading best practices such as Site Reliability Engineering (SRE), CI/CD (continuous integration / continuous deployment), and microservices-based architectures. In short, the goal of DevOps is to release more reliable and performant software through automation.

Over the past few years, more and more companies have been applying these concepts to data in the form of “DataOps.” DataOps refers to the process of improving the reliability and performance of your data through automation, reducing data silos and fostering quicker, more fault-tolerant analytics.

Since 2019, companies such as [Intuit](#), [Airbnb](#), [Uber](#), and [Netflix](#) have written prolifically about their commitment to ensuring reliable, highly available data for stakeholders across the business by applying DataOps best practices. In addition to powering analytics-based decision-making (i.e., product strategy, financial models, growth marketing, etc.), data produced by these companies powers their applications and digital services. Inaccurate, missing, or erroneous data can cost them time, money, and the trust of their customers.

As these tech behemoths increasingly shed light on the importance and challenges of achieving high data quality, other companies of all sizes and industries are starting to take note and replicate these efforts, from implementing more robust testing to investing in DataOps best practices like monitoring and data observability.

But what has led to this need for higher data quality? What about the data landscape has changed to facilitate the rise of DataOps, and as such the rise of data quality? We'll dig into these questions next.

Understanding the “Rise of Data Downtime”

With a greater focus on monetizing data coupled with the ever present desire to increase data accuracy, we need to better understand some of the factors that can lead to data downtime. We'll take a closer look at variables that can impact your data next.

Migration to the cloud

Twenty years ago, your data warehouse (a place to transform and store structured data) probably would have lived in an office basement, not on AWS or Azure. Now, with the rise of data-driven analytics, cross-functional data teams, and most importantly, the cloud, cloud data warehousing solutions such as Amazon Redshift, Snowflake, and Google BigQuery have become increasingly popular options for companies bullish on data. In many ways, the cloud makes data easier to manage, more accessible to a wider variety of users, and far faster to process.

Not long after data warehouses moved to the cloud, so too did data lakes (a place to transform and store unstructured data), giving data teams even greater flexibility when it comes to managing their data assets. As companies and their data moved to the cloud, analytics-based decision making (and the need for high quality data) became a greater priority for businesses.

More data sources

Nowadays, companies use anywhere from dozens to hundreds of internal and external data sources to produce analytics and ML models. Any one of these sources can change in unexpected ways and without notice, compromising the data the company uses to make decisions.

For example, an engineering team might make a change to the company's website, thereby modifying the output of a data set that is key to marketing analytics. As a result, key marketing metrics may be wrong, leading the company to make poor decisions about ad campaigns, sales targets, and other important, revenue-driving projects.

Increasingly complex data pipelines

Data pipelines have become increasingly complex with multiple stages of processing and non-trivial dependencies between various data assets as a result of more advanced (and disparate) tooling, more data sources, and increasing diligence afforded to data by executive leadership. Without visibility into these dependencies, how-

ever, any change made to one data set can have unintended consequences impacting the correctness of dependent data assets.

In short, there's a lot that goes on in a data pipeline. Source data is extracted, ingested, transformed, loaded, stored, processed, and delivered, among other possible steps, with many APIs and integrations between different stages of the pipeline. At each juncture, there's an opportunity for data downtime, just like there's an opportunity for application downtime whenever code is merged. Additionally, things can go wrong even when data isn't at a critical juncture, for instance, when data is migrated between warehouses or manually entered in a source system.

More specialized data teams

As companies increasingly rely on data to drive smart decision making, they are hiring more and more data analysts, data scientists, and data engineers to build and maintain the data pipelines, analytics, and ML models that power their services and products, as well as their business operations.

While data analysts are primarily responsible for gathering, cleaning, and querying data sets to help functional stakeholders produce rich, actionable insights about the business, data engineers are responsible for ensuring that the underlying technologies and systems powering this analytics is performant, fast, and reliable. In industry, data scientists typically collect, wrangle, augment, and make sense of unstructured data to improve the business. The distinction between data analysts and data scientists can be a little vague, and titles and responsibilities often vary depending on the needs of the company. For instance, in the late 2010s, Uber changed all data analysts' titles to data scientists after an organizational restructure.

As data becomes more and more foundational to business, data teams will only grow. In fact, larger companies may support additional roles including data stewards, data governance leaders, operations analysts, and even analytics engineers (a hybrid data engineer-analyst role popular with startups and mid-sized companies who may not have the resources to support a large data team).

With all of these different users touching the data, miscommunication or insufficient coordination is inevitable, and will cause these complex systems to break as changes are made. For example, a new field added to a data table by one team may cause another team's pipeline to fail, resulting in missing or partial data. Downstream, this bad data can lead to millions of dollars in lost revenue, erosion of customer trust, and even compliance risk.

Decentralized data teams

As data becomes central to business operations, more and more functional teams across the company have gotten involved in data management and analytics to streamline and speed up the insights gathering process. Consequently, more and

more data teams are adopting a distributed, decentralized model that mimics the industry-wide migration from monolithic to microservice architectures that took the software engineering world by storm the mid-2010s.

What is a decentralized data architecture? Not to be confused with **the data mesh**, which is an organizational paradigm that leverages a distributed, domain-oriented design, a decentralized data architecture entails that data teams work from a centralized data infrastructure (including storage and data extraction, transformation, and loading, or ETL), managed by a data platform team, with analytical and data science teams distributed across the business. Increasingly, we're finding that more and more teams leaning into the embedded data analyst model are relying on this type of architecture.

For instance, your 200-person company may support a team of 3 data engineers and 10 data analysts, with analysts distributed across functional teams to better support the needs of the business. Either these analysts will report into operational teams or centralized data teams but own specific data sets and reporting functions. Multiple domains will generate and leverage data, leading to the inevitability that data sets used by multiple teams become duplicated, go missing, or go stale over time. If you're reading this book, you're probably no stranger to the experience of using a data set that's no longer relevant, unbeknownst to you!

Other Industry Trends Contributing to the Current Moment

In addition to the aforementioned factors that frequently lead to data downtime, there are also several industry shifts occurring as a result of technological innovation that are driving transformation of the data landscape. These shifts are all contributors to this heightened attention to data quality.

Data mesh

Much in the same way that software engineering teams transitioned from monolithic applications to microservice architectures, the data mesh is, in many ways, the data platform version of microservices. It's important to note that the concept of data mesh is nascent, and there is much discussion in the data community regarding how (or whether it makes sense) to execute on one at both a cultural and technical level.

As first defined by Zhamak Dehghani, a ThoughtWorks consultant and the original architect of the term, **a data mesh**, **Figure 1-1**, is a type of data platform architecture that embraces the ubiquity of data in the enterprise by leveraging a domain-oriented, self-serve design. Borrowing Eric Evans's theory of domain-driven design, a flexible, scalable software development paradigm that matches the structure and language of your code with its corresponding business domain.

Unlike traditional monolithic data infrastructures that handle the consumption, storage, transformation, and output of data in one central data lake, a data mesh supports distributed, domain-specific data consumers and views “data-as-a-product,” with each domain handling their own data pipelines. The tissue connecting these domains and their associated data assets is a universal interoperability layer that applies the same syntax and data standards.

Data meshes federate data ownership among domain data owners who are held accountable for providing their data as products, while also facilitating communication between distributed data across different locations.

While the data infrastructure is responsible for providing each domain with the solutions with which to process it, domains are tasked with managing ingestion, cleaning, and aggregation to the data to generate assets that can be used by business intelligence applications. Each domain is responsible for owning their pipelines, but a set of capabilities applied to all domains that stores, catalogs, and maintains access controls for the raw data. Once data has been served to and transformed by a given domain, the domain owners can then leverage the data for their analytics or operational needs.

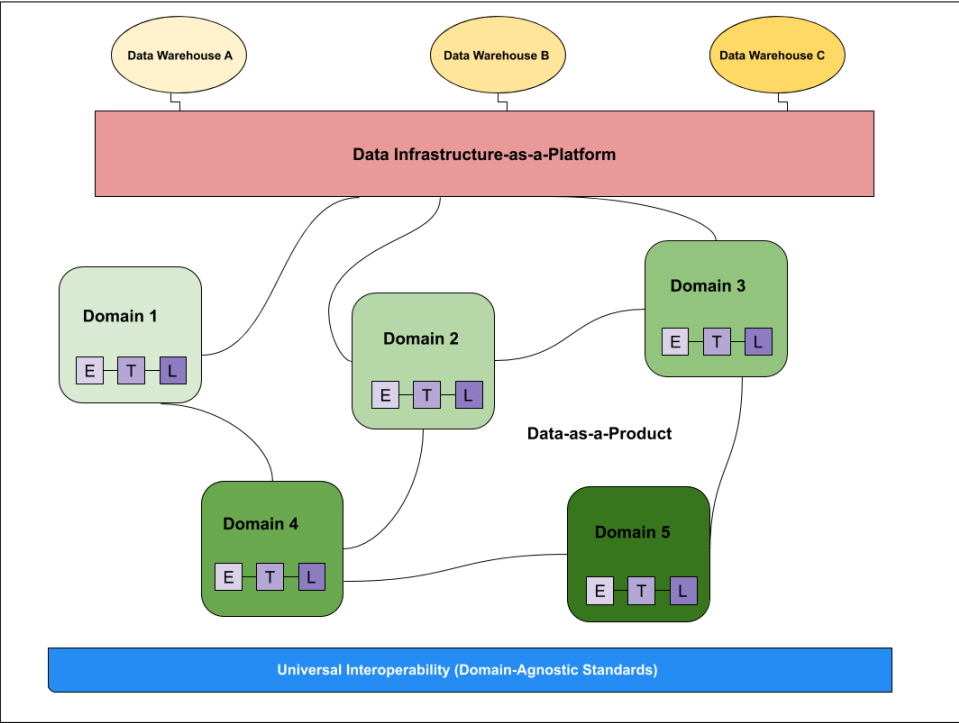


Figure 1-1. The data mesh, as pioneered by Zhamak Dehghani, pushes for a decentralized, domain-oriented data architecture that relies on high quality reliable data and universal governance

The data mesh paradigm is only successful if the data is reliable and trustworthy, and if this “universal interoperability layer” is applied across domains. The only way data can be reliable and trustworthy? A close attention to data quality through testing, monitoring, and observability.

Many companies are adopting the data mesh paradigm, particularly larger organizations with the need for multiple data domains. For instance, in a [January 2021 blog article](#) written by Intuit’s former VP of Data Engineering, Mammad Zadeh, and Raji Arasu, Intuit’s SVP of Core Services & Experiences, Intuit positions itself as a “AI-driven expert platform company,” whose platform “collects, processes, and transforms a steady stream of data into a connected mesh of high quality data.” Another example is [JPMorgan Chase](#), which built a data mesh architecture to help them delineate data ownership between discrete analytics functions and improve visibility into data sharing across the enterprise.

Regardless of your perspective on the data mesh, it’s certainly taken the data community by storm and surfaced great conversation ([and blog articles](#)) on the future of our distributed data architectures and team structures.

Streaming data

Streaming data refers to the process of transmitting a continuous flow of data into your pipeline to quickly generate real-time insights. Traditionally, data quality was enforced via testing batch data before it entered production pipelines, but increasingly, businesses are seeking more real-time analysis. While this has the potential to make insights faster, it also opens up greater questions and challenges related to data quality since streaming data is data “in motion.”

Increasingly, organizations are adopting both batch processing and stream processing, which forces data teams to rethink their approach to testing and observing their data.

Rise of data lakehouse

Data warehouse or data lake? That is the question - at least if you ask a data engineer. Data warehouses, a structured data repository, and data lakes, a pool of raw, unstructured data, both rely on high quality data for processing and transformation. Increasingly, data teams are opting to use both data warehouses and data lakes to accommodate the growing data needs of their business. Meet: the data lakehouse.

Data lakehouses first came onto the scene when cloud warehouse providers began adding features that offer lake-style benefits, such as Redshift Spectrum or Delta Lake. Similarly, data lakes have been adding technologies that offer warehouse-style features, such as SQL functionality and schema. Today, the historical differences between warehouses and lakes are narrowing so you can access the best of both worlds in one package.

This migration to the “lakehouse” model suggests that pipelines are growing more and more complex, and while some might choose one dedicated vendor to tackle both, others are migrating data to multiple storage and processing layers, leading to more opportunities for pipeline data to break even with ample testing.

Summary

The rise of the cloud, distributed data architectures and teams, as well as the move towards data productization have put the onus on data leaders to help their companies drive towards more trustworthy analytics and the data powering them. Achieving reliable data is a marathon, not a sprint, and involves many stages of your data pipeline. Further, committing to improving data quality is much more than a technical challenge; it's very much organizational and cultural, too. In the next chapter, we'll discuss some best practices your team can apply to build repeatable, iterative processes and frameworks with which to better communicate, address, and even prevent data downtime - and in the process, achieve more trustworthy data.

Architecting for Data Reliability

A note for Early Release readers

With Early Release ebooks, you get books in their earliest form—the authors’ raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the second chapter of the final book. Please note that the GitHub repo will be made active later on.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at jleonard@oreilly.com.

Airbnb, the global online vacation marketplace, wrote in a [2020 post](#) on their engineering blog that “leadership [set] high expectations for data timeliness and quality,” leading to the need to make significant investment in their data quality and governance efforts. Meanwhile, Krishna Puttaswamy and Suresh Srinivas, former engineers at Uber, wrote in a [2021 Uber Engineering Blog article](#) that high quality big data is “at the heart of this massive transformation platform.”

It’s no secret: data quality is top of mind for some of the best data teams. Still, it’s one thing to write about it: how do we actually achieve this in practice?

Data reliability—an organization’s ability to deliver high data availability and health throughout the entire data lifecycle—is the outcome of high data quality. As companies ingest more operational and third-party data than ever before, with employees from across the organization interacting with that data at all stages of its lifecycle, it’s become increasingly important for that data to be reliable.

Data reliability has to be intentionally built into every level of your organization, from the processes and technologies you leverage to build and manage your data stack to the way you communicate and triage data issues further downstream. In this chapter, we'll explore how to architect for data reliability at each stage of the pipeline—and data engineering experience.

Measuring and Maintaining High Data Reliability at Ingestion

Now that we have a better understanding of the state of data quality, let's look at what all of this means in practice, starting with ingestion. However, we would be remiss not to discuss some of the fundamental best practices that ensure high data quality before ingestion to the data warehouse or lake.

It is common for organizations to ingest data from both internal and external, third-party sources depending on the needs of the business. Your decision-making is only as good as the data you are using for insights and analysis: garbage in is garbage out. It's important to make sure your organization has the right set of practices in place to ensure data quality.

To achieve this, organizations have begun to establish rigorous data quality control standards for all data that enters their data ecosystem. While data quality issues can occur at any point in your data pipeline, most organizations will agree that catching and fixing data quality issues at the point of ingestion will help minimize the chances of poor quality data working its way downstream.

Best practices such as data cleaning, data wrangling (the process of structuring and enriching your data into a desired format), and data testing are ways organizations are ensuring data quality is up to their organizations' needs. And with technology advancing over the years in the data industry, an abundance of tools have emerged in the space to automate this process for companies.

Such tools allow organizations to automatically examine aspects of data such as its format, consistency, completeness, freshness, and uniqueness. By automating this process, organizations not only save time and resources when data cleaning but ensure that the quality of incoming data is constantly controlled and managed whenever data enters their ecosystem.

Data cleaning (also commonly referred to as cleansing) involves preparing and modifying data for future analysis by removing incomplete, irrelevant, incorrect, incorrectly formatted, or duplicate data from a data set. While the data cleaning process can be tedious, it is a very important responsibility for a data team. In fact, the data cleaning process in most organizations is owned by the data engineering and or data science team. However, it is important to educate the rest of the organization on the

importance of data cleaning as everyone in the company plays a key role in ensuring the integrity of data.

When dealing with missing or inaccurate data many companies turn to a process called data enrichment which is a process where organizations are able to merge and add either first or third-party data to datasets they already are working with. By enriching data, organizations are able to add more value to their datasets, which in the end makes data more useful and reliable.

After data cleaning, data testing is your best line of defense against data quality before ingestion.

Data testing is the process of validating your organizations' assumptions about the data, either before or during production. Writing basic tests that check for things such as uniqueness and `not_null` are ways organizations can test out the basic assumptions they make about their source data. It is also common for organizations to ensure that data is in the correct format for their team to work with and that the data meets their business needs.

There are a few basic types of data quality tests, each of which we'll address later in this book:

Unit testing

Unit tests check that a line of code (SQL) does what it is supposed to do; they can be used with very, very small snippets of data. When unit testing data, you must separate the business logic from the “glue code.”

Functional testing

Functional tests are used with large data sets, and are often separated in data validation, integrity, ingestion, processing, storage, and ETL. This type of testing frequently occurs in the pipeline (pre-analytics layer).

Integration testing

Integration tests are used to ensure that your data pipeline meets your criteria for validity (i.e., within expected ranges); generally, teams will run fake data through the pipeline using these tests before leveraging production data.

Some common data quality checks include:

Null values

Are any values unknown (NULL)?

Freshness

How up to date is my data? Was it updated an hour ago or two months ago?

Volume

How much data is represented by this data set? Did 200 rows turn into 2,000?

Distribution

Is my data within an accepted range? Are my units the same within a given column?

Missing values

Are any values missing from my data set?

How would you write these? In a later chapter, we'll go through a list of common data tests in SQL that can be applied to many open source languages (with varying syntax and glue code, of course), but for explanation's sake, let's walk through an example data set.

Let's assume you're a media distributor working with a data set that tracks your global customer base, including location (CITY) and how much they're paying for a subscription (PRICE) of your services. There are 500 entries in this data set, and 5 columns: CITY, COUNTRY, PRICE, CUSTOMER NAME, PRODUCT.

If you wanted to test the data to ensure that you're only running a pipeline on customers who live in BERLIN, you might run a SQL command that says:

```
SELECT * FROM Customers WHERE City = "Berlin";
```

And if you wanted to understand if there are any null values in CITY, you might query:

```
SELECT * FROM Customers WHERE City IS NULL;
```

If you wanted to understand if any product is more expensive than \$4.50 and less expensive than \$8.50, you might run:

```
SELECT * FROM Products WHERE Price > 4.50 AND Price < 8.50;
```

And this just scratches the surface of the types of tests you could run to better understand the health of your data.

Based on the above examples, you can tell that data testing can be quite tedious. Before setting your tests, you have to have a clear understanding of the data, what to expect from it, and what “bad data” looks like. We often refer to these expectations as “assertions,” derived from the world of unit test-driven development in software engineering.

As a result, data team members often split testing responsibilities over data sets, with individual analysts and engineers responsible for creating and maintaining tests for the data sets they're building pipelines for and interacting with daily. And some companies hire entire Data Quality Assurance teams to handle data testing, with responsibilities including creating tests for business use cases and maintaining existing tests.

In the last few years, tools, including open source solutions like Apache Griffin and Great Expectations have emerged in the data testing category to help data engineers

and analysts automate the data testing process at different stages of the pipeline. dbt (data build tool) is another solution in the data space that has narrow testing capabilities. These tools also help data team members document important information about data sets (in other words, metadata), such as what the data represents, how to use the data in reporting as well as what other data a given asset relies on and feeds into. More on these later.

We cannot emphasize enough the importance of testing your data before production; just as a software engineer would never (purposefully) push code to production without testing it first, a data engineer should never run a pipeline with untested data. But to test your data well, you need a clear understanding of your data health prior to running your pipelines—we'll talk about ensuring data health and observability later.

Keep in mind that data testing only catches expected data quality issues, and does not have the scalability or knowledge to account for “unknown” data quality issues. Data changes a lot, even during production, so it's important to supplement testing with reactive monitoring and anomaly detection. More on that later.

Measuring and Maintaining Data Quality in the Pipeline

In the 1990s, when your website was down, most people wouldn't notice by the time you were able to get it up and running again given the low volume of users to most websites (after all not everyone was using the worldwide web!). Now, in the 2020s, everyone notices when your service or application is down, for instance, [Slack's infamous outage](#) in January 2021. Slack, a popular enterprise communication management platform with [over 12 million daily active users](#) at the time, went down on Monday, January 4, 2021, the first workday after the New Year's Eve holiday, causing many companies to be without their primary means of inter-office communication. Consider the fact that many workers were homebound as a result of the [COVID-19 pandemic](#), and you can imagine the frustration that ensued.

Nowadays, nearly every business that hosts software relies on Site Reliability Engineers (SRE) to ensure that applications in production are reliable at all times. As organizations grow and the underlying tech stacks powering them become more complicated (think: moving from a monolith to a microservice architecture), it's important for SRE teams to maintain a constant pulse on the health of their systems. Observability, a more recent addition to the engineering lexicon, speaks to this need, and refers to the monitoring, tracking, and triaging of incidents to prevent downtime.

As a result of this industry-wide shift to distributed systems, site reliability engineering has emerged as a fast-growing engineering discipline. At its core, site reliability engineering is broken into three major pillars:

Metrics

Refer to a numeric representation of data measured over time

Logs

Records of an event that took place at a given timestamp; they also provide valuable context regarding when a specific event occurred

Traces

Represent causally related events in a distributed environment

Increasingly, data teams are coming to rely on similar principles of observability and monitoring to track data quality in production pipelines, with companies developing their own unique methodology for how to measure it depending on the needs of the business.

Similarly, data observability (i.e., ensuring data quality in the pipeline) can be broken down into five major pillars as shown in **Figure 2-1**.

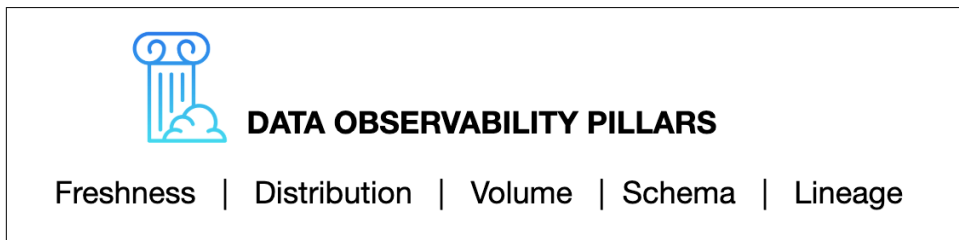


Figure 2-1. The five pillars of data observability, like the three pillars of SRE, highlight the elements of data health that should be closely monitored as an indicator of high data quality

Freshness

Is the data recent? When was the last time it was generated? What upstream data is included/omitted?

Distribution

Is the data within accepted ranges? Is it properly formatted? Is it complete?

Volume

Has all the data arrived?

Schema

What is the schema, and how has it changed? Who has made these changes and for what reasons?

Lineage

For a given data asset, what are the upstream sources and downstream assets which are impacted by it? Who are the people generating this data, and who is relying on it for decision making?

The five pillars of data observability serve as key measurements for understanding the health of your data at each stage in its lifecycle, and provide a fresh (no pun intended) lens with which to view the quality of your data.

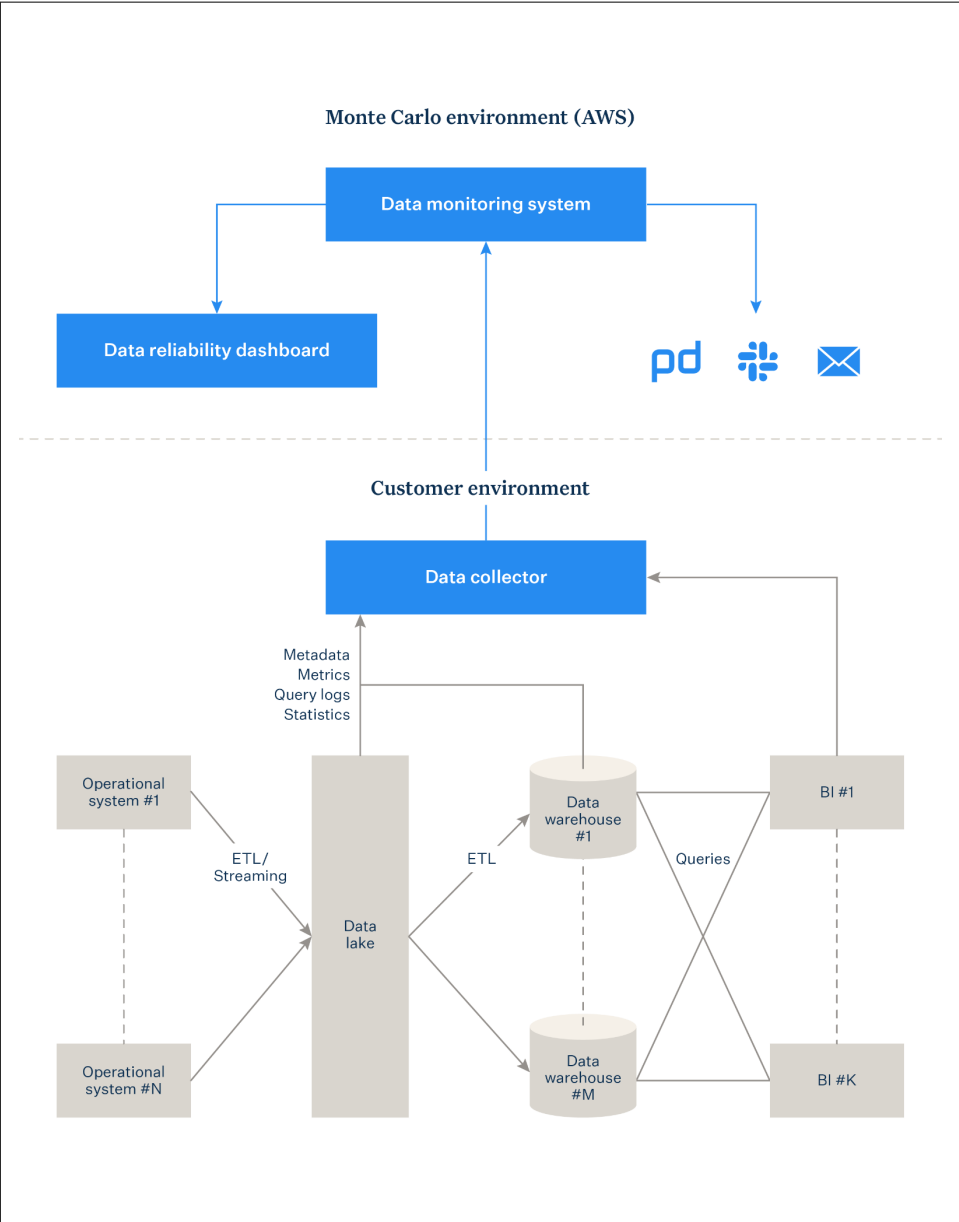


Figure 2-2. A data pipeline refers to “data in production” and is composed of a data warehouse / lake (or both), ETL, and an analytics layer

As previously mentioned, data downtime refers to periods of time where data is missing, erroneous, or otherwise inaccurate, and often suggests a broken data pipeline. By measuring data downtime, you can determine the reliability of your data and ensure the confidence necessary to use it. While SRE measures application downtime as a function of time, we can similarly measure data downtime.

As data becomes increasingly tied to business outcomes, we're observing a sea shift to less subjective, more quantifiable metrics, and for many teams, measuring uptime and downtime for data is broadly applicable and provides a good starting point for understanding data health.

Understanding Data Quality Downstream

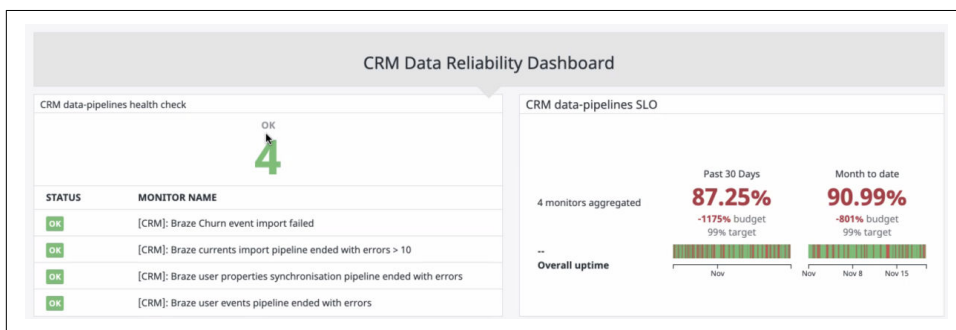


Figure 2-3. A data reliability dashboard (in this case, rendered in Datadog and Grafana) can help your data team and stakeholders keep a pulse on the quality and reliability of your data

Chances are, you won't realize that your data is "bad" until it reaches the analytics layer—or even beyond, when data is piped back to the applications and services you collect it from (i.e., the list of acronyms we shared earlier). As previously mentioned, teams can leverage monitoring and observability tools to catch data quality issues, or even set up a sequence of tests based on assumptions about your data.

Once data is in the analytics layer, teams can track quality and reliability in a few different ways, including:

- Data Reliability Dashboard that tracks the TTR, TDD, and other data quality metrics after data lands in the dashboard
- Service-level agreements (SLAs), service-level objectives (SLOs), and service-level indicators (SLIs) regarding whether your data meets the standards outlined by your business

- NPS score measuring how satisfied your stakeholders are with the data (i.e., was it delivered on time and do I trust it).

However, when understanding data quality in the dashboard, the most important step is to align with your stakeholders around how they intend to use the data and what high quality, reliable data looks like to them. An easy way to do this is to understand what they'll be using the data for, and which data should be prioritized. It's nearly impossible to write data tests or even monitor for all critical data assets, but aligning on which data matters and to who will cover many of your bases.

Traditionally, data quality is measured by data stewards and data governance leaders by a few defining characteristics. According to the [Data Management Association UK](#), companies have measured data quality based on six key dimensions:

Completeness

How complete is my data?

Timeliness

Did my data arrive on time?

Validity

Does my data meet all syntax requirements (i.e., format, type, or range)?

Accuracy

Does data describe the real-world environment it's trying to represent?

Consistency

Is data consistent against well-understood and accepted definitions?

Uniqueness

Is an individual data point recorded more than once?

In the world of data engineering and data analytics, these measurements are useful, but they aren't always directly applicable (i.e., accuracy). As a data engineer, you're not usually the one working with the end result (clean, reliable data) in the context of the business; you're just notified when something breaks, and trusted to apply testing and monitoring at each step of the process.

For data engineers, measuring data quality in the dashboard might boil down to tracking:

- The ratio of data to irrelevant or erroneous data (in other words, if you have 1TB of data, how much of that data is missing, inaccurate, or stale)
- The number of null or missing values in a given data set, or the completeness of data (which won't account for "accuracy," given that "inaccurate values" can skew this metric)

- The timeliness of data (in other words, was data late?)
- The percent of duplicated values (which only accounts for uniqueness of data, and not any of the other possible ways data can break)
- The consistency of data (i.e., does each value in this row or column have the same format and size?)
- The number of functional teams who consistently access and use your data (this is applicable when applying distributed data architectures, like the data mesh, for which data quality is of the utmost importance)

And the list goes on.

Building Your Data Platform

In addition to monitoring and alerting for data issues at all stages of the data pipeline, delivering reliable data requires a thoughtful data platform—a combination of technologies that enable you to manage data holistically, from ingestion to analytics.

Data platform requirements change with your business. The “right” data platform for a 2,000-person e-commerce company will look quite different from a 20-person Fin-Tech startup, but there are still a few core layers that all data platforms require. We think about the data stack in six layers: ingestion, storage and processing, transformation and modeling, business intelligence and analytics, discovery and governance, and quality and observability, as shown in [Figure 2-4](#). It’s important to note that “layers” is used in the figurative sense; each of these elements are interconnected (vs. stacked), and are not listed in order of priority or importance. But we’ve found that best-in-class data teams invest in each of them, sometimes leveraging the same tools or technologies to account for 2–3 at a time.

We’ll cover each of these layers in detail as we explore how to build your data stack.

The 6 Must-Have Layers of Your Data Platform

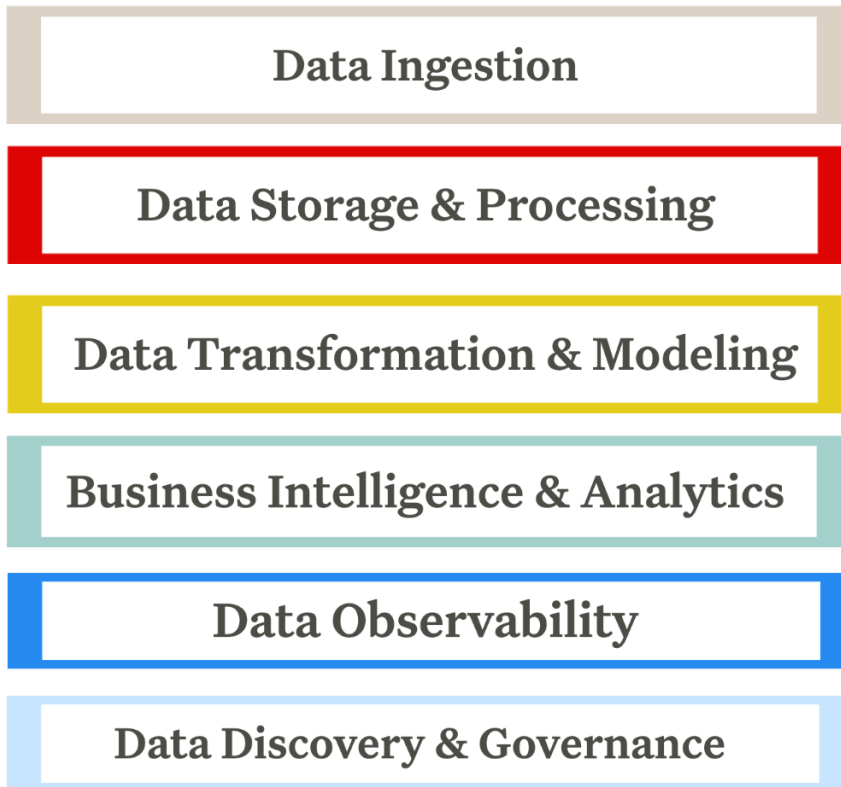


Figure 2-4. While building a data platform, it's important to account for six foundational, interconnected layers: data ingestion; data storage and processing; data transformation and modeling; business intelligence and analytics; data observability; and data discovery and governance. Of course, as architectures grow to accommodate more advanced use cases, the number of layers will increase depending on the needs of your data team.

Data Ingestion

Modern data ingestion is complex, usually involving the collection of structured and unstructured data from a wide variety of sources. This is also known as the extraction and loading stage of Extract Transform Load (ETL) and Extract Load Transform (ELT).

Most ETL tools *extract* data from external sources or internal systems, *transform* it within a staging area into an acceptable (usually relational) format for storage, and *load* it into databases. With the advent of modern cloud-based data warehouses that can store untransformed data, however, data teams can adopt the newer integration architecture of ELT—extracting raw data from a source, loading it directly into a data warehouse, and transforming it at the end of the process.

There are numerous ingestion tools available on today’s market—both off-the-shelf and open-source—although some data teams choose to use custom code and build custom frameworks to handle ingestion.

Orchestration and workflow automation are often folded into the ingestion layer—taking siloed data, combining it with other sources, and making it available for analysis. However, we would argue that orchestration can, and should, be weaved into the platform after you handle the storage, processing, and business intelligence layers. After all, orchestration requires an orchestra of functioning data!

Keep in mind: it’s best practice to test your data at each step of the data pipeline and make the proper assertions to help concretize what data quality at each step looks like. Data tested at ingestion will not necessarily stay reliable as it evolves through the pipeline.

Data Storage and Processing

The storage layer is the workhorse of your data stack—it’s where your newly ingested data is stored and processed. Data storage today looks unrecognizable compared to the on-premises computing clusters of a decade ago, thanks to the evolution of cloud-native data storage solutions. These tools make it much more accessible and affordable for companies to store and process massive amounts of data at scale.

There are three primary types of data storage solutions: data warehouses, data lakes, and data lakehouses. *Data warehouses* are fully managed solutions that typically require data to be structured according to specific schema—often forcing stricter data hygiene from the moment of ingestion. *Data lakes*, on the other hand, are often custom-built by data teams with a combination of open-source and off-the-shelf technologies, supporting raw, unstructured data and decoupled, distributed computing. *Data lakehouses* are an emerging hybrid, adding warehouse-style features like SQL functionality and schema to data lakes or providing more flexibility to traditional warehouses.

The “right” solution will be different for every company, and even for the same company at different stages—evolving along with the number of data sources you leverage or the skillsets of the primary users of your data platform.

Data Transformation and Modeling

The terms “data transformation” and “data modeling” may be used interchangeably, but they are very distinct processes. Data transformation encompasses preparing raw data for analysis and reporting. Data modeling is the process of identifying the key concepts and relationships in your data that encapsulates your business logical, and then modeling these in the form of tables and the relationships between them.

Data transformation usually includes exploratory data analysis (EDA, in other words, profiling data to understand its structure and characteristics), data mapping (defining how individual fields are formatted to produce the final output), code generation (producing executable code based on those defined rules or metadata), code execution (applying the generated code to produce the desired output), and data review (ensuring the transformed data meets requirements).

Traditionally, data transformation has been performed by specialized engineers using scripting languages like Python, R, or SQL and time-consuming work cycles. Today, some data transformation can be accomplished by end-users—like business analysts—using cloud-based tools and technologies. This modern, self-service approach allows the business users literate in SQL (and often closest to the data) to maintain more control in setting requirements and speed up the time to actionable insights, and no-code or low-code approaches make this transition possible. That being said, transformation is still very much a data engineer-owned process that incorporates Python and languages outside of SQL.

Data transformation can happen in batch or bulk processes, or even streaming, which is a less common but promising approach to handling transformation and modeling in real time for the rare use case (i.e., when having access to fresh data is more important than ensuring that data is accurate) in which it makes sense.

Business Intelligence and Analytics

Once data is collected, transformed, and stored, it must be made available to business users—after all, the best data in the world won’t do any good if your employees can’t use it.

This highly visible layer of the data stack is known as business intelligence and analytics. If your data platform is a book, your BI and analytics layer is its cover, complete with a descriptive title, engaging visuals, and a summary of what the data is trying to tell you. The BI layer makes data actionable, and without it, your data lacks meaning.

Analytics tools retrieve, analyze, and surface data through dashboards and data visualizations, allowing users to leverage data for actionable insights. Charts, graphs, maps, and other data visualization tools bring your data to life, giving employees an accessible way to explore and understand patterns and trends in your data. Without visualizations, your data remains virtually inaccessible—millions of rows on a spreadsheet that may be accurate, but aren't easily understood.

Experiencing data through visualizations empowers data storytelling, or the ability to convey data as a narrative that humans can comprehend and, therefore, act on. Data storytelling goes a step beyond visualizations by communicating the context around changes in data and sharing the *why* behind data trends. The discipline of data storytelling must be practiced and honed over time, but your team can't begin to develop those skills without access to self-serve business intelligence and analytics tooling.

Data Discovery and Governance

Data teams need a scalable way to document and understand critical data assets. Historically, this has been accomplished through data catalogs, which serve as an inventory of metadata and provide an understanding of your data's accessibility, health, and location. Data catalogs make it easy to keep track of where personally identifiable information is housed, as well as who within your organization has permission to access it across the pipeline—making them an integral part of data governance and regulatory compliance.

Modern data teams, however, are encountering the limitations of traditional data catalogs. As data ecosystems grow increasingly complex and leverage large amounts of unstructured and schemaless data, traditional catalogs can fall short due to their lack of automation and inability to scale with the growth and diversity of modern data stacks. They tend to require data teams to do the heavy lifting of manual data entry, including updating the catalog as data assets evolve, and often don't support the dynamic nature of unstructured data.

Data discovery is a new approach increasingly applied to data cataloging that provides a domain-specific, dynamic understanding of your data based on how it's being ingested, stored, aggregated, and used by a set of specific consumers. With data discovery, governance standards should remain federated across domains, but unlike more traditional approaches, data discovery enables a real-time understanding of the data's current state—not its ideal or “cataloged” state.

Data discovery can answer these questions not just for the data's ideal state but for the current state of the data across each domain:

- What data set is most recent? Which data sets can be deprecated?
- When was the last time this table was updated?

- What is the meaning of a given field in my domain?
- Who has access to this data? When was the last time this data was used? By who?
- What are the upstream and downstream dependencies of this data?
- Is this production-quality data?
- What data matters for my domain's business requirements?
- What are my assumptions about this data, and are they being met?

Data discovery makes it possible for data teams to trust that their assumptions about data match reality, empowering dynamic discovery and a high degree of reliability across your data infrastructure.

Developing Trust in Your Data

Now that you know what steps to take to ensure data quality pre, during, and post-production as well as which technologies you need to build a robust data platform, the next step is to develop trust in your data through the right processes and culture. After all, the most advanced data stack in the world is useless unless that data you're using can be trusted to deliver reliable insights to your business. Data for the sake of data is about as useful as a fish riding a bicycle.

When it comes to building reliable and trustworthy data systems, the first step is to understand the health of your data in its current state. In the same way that software engineering teams develop trust in their software applications through observability and DevOps, data teams must embrace similar best practices when it comes to building trust in their data. Data Observability is a good first step.

Data Observability

The sixth layer of the modern data stack isn't a final step per se, but rather, an interconnected approach that weaves throughout your entire data lifecycle: observability.

Over the last two decades, Developer Operations (DevOps) engineers have developed best practices of observability to ensure applications stay up, running, and reliable. And just as application observability includes monitoring, tracking, and triaging of incidents to prevent downtime, modern data engineers are applying the same principles to data.

Data observability refers to an organization's ability to fully understand the health of the data in their system at every stage of the lifecycle.

As mentioned in Chapter 1, data observability applies DevOps practices of automated monitoring, alerting, and triaging across five pillars: freshness, distribution, volume, schema, and lineage.

End-to-end data observability is crucial for ensuring data quality. Effective observability tooling will connect to your existing data stack, providing end-to-end lineage that allows you to surface downstream dependencies and automatically monitor your data-at-rest—without extracting data from your data store and risking your security or compliance. Having observability makes audits, breach investigations, and other possible data disasters much easier to understand and resolve while keeping your CTO from having an ulcer!

Measure the Cost of Broken Data

Unreliable data can lead to wasted time, lost revenue, compliance risk, and erosion of customer trust. Many data leaders tell us their data scientists and engineers spend **40 percent** or more of their time troubleshooting or firefighting data problems. Gartner estimates companies spend upwards of **\$15 million annually** on data downtime, while over **88 percent of U.S. businesses** have lost money because of data quality issues. And **1 in 5 companies** have lost a customer due to data quality issues.

Before you can improve your data quality, it's important to measure the impact of poor data quality and delineate which data sets matter most to your organization. As you're likely aware, all data is not created equal, but having a sense of the cost of data downtime on your business for key assets will be foundational to communicating the impact of data quality to your stakeholders.

Measuring the ROI on data quality

Quantifying and communicating the value of data quality is a complex endeavor. We've found that the following metrics, borrowed from DevOps practitioners, provide a good start: Time to Detection and Time to Resolution.

Time to Detection (TTD) describes the length of time it takes for your data team to surface a data quality issue of any kind, from freshness anomalies to schema changes that break entire pipelines. For many teams, TTD is measured in days, weeks, or even months—because most often, data outages are first detected by downstream consumers when a dashboard or report “looks off.”

These periods of time are incredibly costly because the more time that passes, the harder it becomes to recover data through re-processing or backfilling source data. Additionally, every business decision, marketing campaign, or product roadmap update that relied on the incorrect data needs to be re-validated or communicated to stakeholders.

Time to Resolution (TTR) refers to how quickly your team is able to resolve a data incident once alerted. This can be minutes, hours, or days, depending on the complexity of the incident, the availability of your data lineage, the robustness of your data discovery or catalog, and the resources available. TTR metrics allow you to

understand the severity of your data issue and track the amount of time it takes to resolve it. By converting to dollars—that is, articulating how much money is spent or saved as a result of TTR—it becomes considerably easier to communicate the impact of this broken data to your stakeholders.

$$\begin{array}{c} \text{(TTD hours + TTR} \\ \text{hours)} \\ * \\ \text{downtime hourly cost} \\ = \\ \text{cost of data downtime} \end{array}$$

Figure 2-5. Data downtime is an effective measurement to understand how poor your data quality is as a representation of how long it takes you to fix it

Downtime hourly cost is a generalized metric to represent the engineering time spent per downtime hour and the impact of data downtime on data consumers and business decisions.

Engineering time spent can be calculated as a factor of downtime hours. For example, we can estimate that 1 data engineer spends 1/4 of every downtime hour monitoring for and investigating issues, which contributes ~\$14.75 per downtime hour (avg \$59/hr salary + benefits for data engineer, a back-of-the-envelope calculation based on U.S. data engineering salary data from [ZipRecruiter](#)). Over time and as technical debt accrues, the cost of downtime only grows.

Impact of data downtime cost varies significantly depending on the potential impact of a downtime hour on your business. If for example, you rely on data to report earnings to Wall Street, a downtime hour resulting in mis-reporting data is catastrophic, likely contributing \$1000s/hr to the downtime cost. Additionally, you can add the cost of downtime on your analytics team. If, for example, you have 10 analysts, the

cost of them sitting idle during a downtime incident is significant (avg \$75/hr salary * 10 = \$750/hr). Assuming not all analysts will be impacted by a downtime hour, we can conservatively reduce this by 75% to \$175/hr.

In this scenario, then, we can estimate a downtime hour to cost our business between \$500/hr.

Assuming you have ~100 downtime hours a month, the cost to your business could easily exceed \$600,000/year (100hrs/month * \$500/hr * 12 months).

Often, the impact of data quality and reliability goes unnoticed (in fact, many of these issues often go unnoticed until it's too late!), and it can be difficult to proactively justify budget and resources with executives and other stakeholders who aren't on the data team. By calculating baseline TTD and TTR, it becomes much easier to then communicate exactly what impact you expect to generate on the business. Without this baseline, it's much harder to get operational buy-in from the powers that be to grow your team, up-level your tech stack, and scale out the data quality program of your dreams.

Calculating the cost of broken data

Your annual cost of broken data can be approximated by the engineering or resources you must spend to resolve the problem. We believe the right equation factors in the cost of labor to tackle these issues, your compliance risk (we can use average General Data Protection Regulation, or GDPR, fines to quantify this risk), and the opportunity cost of losing stakeholder trust in your data.

Based on available data as well as interviews and surveys conducted with over 150 different data teams across industries, we estimate that data teams spend 30-40 percent of their time handling data quality issues instead of working on revenue-generating activities.

Bringing this together, we can use the following equation to calculate the cost of broken data:

Labor cost: ([Number of data engineers] x [Annual salary of data engineer]) x 30%

+

Compliance risk: [4% of your annual revenue]

+

Opportunity cost: [Revenue you could have generated if you moved faster, releasing X new products, and acquired Y new customers]

= Annual cost of broken data

This framework is a starting point, but measuring the cost is the first step towards fully understanding the implications of broken data at your company—and, ultimately, preventing them altogether.

How to Set SLAs, SLOs, and SLIs for Your Data

Again, we can look to our DevOps counterparts for inspiration on architecting reliability into our data systems. Site reliability engineers use frameworks such as Service Level Agreements (SLAs), Service Level Objectives (SLOs), and Service Level Indicators (SLIs) to reduce application downtime and ensure reliability. Several of the data teams interviewed for this book have begun to implement these frameworks across their organizations to prioritize, standardize, and measure data reliability.

Essentially, companies use SLAs to define and measure the level of service a given product, internal team, or vendor will deliver, along with potential remedies if those SLAs are not met. For example, Slack promises its customers on Plus plans and above 99.99% uptime every quarter—and if they fall short, Slack will provide service credits on their accounts for future use.

Many software teams develop internal SLAs to help engineering, product, and business teams align on what matters most about their applications and prioritize incoming requests. The very practice of codifying SLAs—rather than counting on everyone to do their best and shoot for as close to 100% uptime as possible—helps set clear expectations. With these SLAs in place, engineering teams and their stakeholders can be confident they’re paying attention to the same metrics and speaking the same language.

And setting those non-100% expectations leaves space for growth. Without some tolerance for minimal downtime, there’s zero room for innovation—and seasoned engineers know that even with all the best practices in place, systems will still break occasionally. But with solid SLAs in place, engineers know exactly how and when to intervene once something does go wrong.

Similarly, SLAs can help data teams and their consumers define, measure, and track data reliability across its lifecycle. Setting data reliability SLAs builds trust between your data, your data team, and downstream consumers. Without agreed-upon metrics, consumers can make inaccurate assumptions or look to anecdotal evidence about the reliability of your data. With SLAs in place, your organization can become more “data-driven” about data.

Additionally, by formalizing communication and prioritization processes, data reliability SLAs help your data team have a clearer grasp on business priorities and make it easier to respond swiftly when incidents occur.

Still, setting SLAs in and of themselves is meaningless: you need alignment from data producers, engineers, analysts, and consumers on what these SLAs should be and how much attention and resources should be devoted to maintaining them.

How to create data reliability SLAs

Setting data SLAs requires specificity and collaboration, and clear, upfront alignment with everyone whom this SLA affects (data producers, data engineers, data analysts, business developers, data consumers, etc.). In fact, just setting SLAs for the sake of setting them can often leave your team in a poor position if there's no investment or accountability in meeting them. Instead, teams should create and evangelize SLAs the same way they set key performance indicators (KPIs) to larger strategic projects:

1. Take stock of business priorities
2. Assess how these business priorities are enabled or tied to data analytics
3. Understand your consumer's need for high data quality / tolerance for poor data quality
4. Set SLAs accordingly, and seek stakeholder feedback and alignment
5. Measure SLAs

So, how do we get started?

According to Google's highly influential SRE Handbook, service level agreements (SLAs) require well-defined service level indicators (SLIs), quantitative measures of service quality, and agreed-upon service level objectives (SLOs), the ideal values or ranges each indicator should meet. For example, most engineering teams use availability as a site reliability indicator, and set an objective to maintain that availability at least 99% of the time.

For data teams, setting reliability SLAs usually includes three steps: defining, measuring, and tracking.

1. Defining data reliability with SLAs. Setting SLAs first requires agreeing upon and clearly defining what reliable data means to your business. We recommend starting this process by conducting an inventory of your data, how it's being used, and by whom—assessing the historical performance of your data to get a baseline metric of reliability.

Data teams should also gather feedback from their consumers on what reliability looks like to them. Data engineers can be removed from their colleagues' daily workflows, but it's crucial to understand how consumers interact with data, what data matters most, and which potential issues require immediate attention. All relevant stakeholders, including data leaders or business consumers, should weigh in—and buy in—on the definitions of reliability you're developing.

After all, powerful technologies and workflows can facilitate proper incident response, but it can't replace a poor culture. Data teams, partners, and consumers must align on SLAs before they are useful to the business.

2. Measuring data reliability with SLIs. With a baseline in place and thorough understanding of your data consumers' needs, you can begin to target the metrics that will become your service-level indicators of reliability.

Generally speaking, data SLIs should reflect the agreed-upon state of data you defined in the first step, providing boundaries of how data is and isn't used and describing what data downtime looks like. Scenarios here could include missing, duplicative, or stale data.

SLIs will vary based on your specific use case, but here are a few examples of metrics often used to quantify data health:

The number of data incidents for a particular data asset (N). This may be beyond your control for external data sources, but is still a key driver of data downtime and typically worth measuring.

3. Tracking data reliability with SLOs. Once SLIs are identified, you can set objectives, or ranges of acceptable downtime for your data. These SLOs should be based on your real-world circumstances—for example, if you decide to track TTD but don't use automated monitoring tools, your SLO should be more generous than a mature organization with robust data observability tooling.

Setting these ranges makes it possible to create a uniform framework that rates incidents by level of severity, and makes it easy to respond swiftly when issues occur. With these objectives set and incorporated into your SLAs, you can build dashboards that track and report on progress—either custom, ad-hoc solutions or using dedicated data observability tools.

These measurements can be quite useful for data teams when understanding the health of their data at an operational level, but when it comes to measuring the impact of data quality on the business, we suggest revisiting data downtime.

Setting SLAs, SLOs, and SLIs for data is only the first piece of the puzzle. When data incidents occur, we also need a way to triage and manage incidents before they become a massive headache for downstream consumers.

For this, we can again turn to our friends in DevOps for inspiration. Most engineering organizations allocate entire Site Reliability teams to identifying, resolving, and preventing downtime. In today's modern data organization, data engineers often bear the brunt of the pain when pipelines break and dashboards turn wonky.

To make the incident resolution process easier and more seamless, we can take a page out of the SRE's handbook to effectively communicate and triage data issues as they arise.

For example, let's say one of your executive's critical reports is surfacing stale data. From the outset, you're not sure how this pipeline broke, but you need to communicate that it HAS broken and that your team is on the case. And as you're resolving this issue, you need to consistently update not just your fellow data downtime sleuths but also your key stakeholders on the incident resolution process.

While what it takes to achieve reliable data is ultimately up to the needs of your business, having a great communications strategy in place will make it that much easier to execute on your SLAs.

Let's shift gears from talking about principles and take a look at how all of these concepts were applied when an eBook subscription service was struggling with a lack of real-time data.

Case Study: Blinkist

With over 16 million users worldwide, Blinkist helps time-strapped readers fit learning into their lives through their ebook subscription service. Gopi Krishnamurthy, Director of Engineering, led the team responsible for data engineering, infrastructure, cloud center-of-excellence, growth, and monetization. For Blinkist, having trustworthy and reliable data is foundational to the success of their business.

Lack of real-time data tracking caused marketing spend to decrease across critical distribution channels. Image courtesy of Blinkist. As a high-growth company, Blinkist leveraged paid performance marketing to fuel customer acquisition. Their 2020 strategy—with a 40 percent growth target—included a significant investment in channels like Facebook and Google, which would auto-optimize campaigns based on behavioral data shared between the Blinkist app and the channels themselves.

Of course, like so many companies in 2020, the COVID-19 pandemic changed everything. Now, historic data didn't reflect the current reality of their audience's daily lives, and real-time data became essential—not just for determining advertising spend, but for understanding the current state of how users were interacting with the Blinkist app and content across the web.

Any inaccuracies in this data could impact decision-making, from campaign spending to updating the product roadmap. It was crucial that no opportunities to innovate were missed, from adding new features to simplifying onboarding to testing new advertisements—because a campaign around “improving your commute” just wasn't relevant anymore.

As C-level execs and campaign managers grew increasingly dependent on real-time insights to drive marketing strategy, budget spend, and ROI, Gopi and his team were struggling with data downtime—issues with data quality, dashboard update delays, and broken pipelines.

“Every Monday, we had executive calls,” said Gopi. “And almost every Monday, I was on this call trying to answer why we are not able to scale, what were the issues, how many problems we face in terms of tracking data...trying to explain the severity of the problem and trying to boost confidence with executive stakeholders.”

Gopi estimates his team was spending 50 percent of their working hours firefighting data drills, trying to resolve data downtime issues while rebuilding trust with the rest of the organization. It wasn’t sustainable and something had to change.

Foundational to achieving data reliability was a focus on data governance, data quality, and refactoring systems. Gopi and his team implemented a regimented approach to data testing and observability that tracked key data SLAs and SLIs.

“At the core of this framework is data reliability engineering—that we treat data reliability as a first-class citizen, the same way engineering teams in the last decade have started to treat DevOps and site reliability engineering,” said Gopi.

By investing in testing and data observability and setting clear data reliability SLAs to measure data reliability, Blinkist was able to remediate data downtime before it affected downstream consumers. As Gopi and his team worked to rebuild broken trust along with broken pipelines, they partnered with company leaders to build a shared understanding of data reliability principles and set concrete data SLAs.

Outcome: Time savings of 120 hours per week for a team of 6 data engineers through testing, observability, and SLA-alignment.

“The scale of growth that we’ve seen this year is overwhelming,” Gopi said. “Although the data teams can’t take full credit, I definitely think the things we were able to do—in terms of data observability and bringing transparency into data operations—improved how we target our audience and channels.”

Summary

For those beginning their data quality journeys, architecting for data reliability requires a three-pronged approach:

- Invest in DevOps-inspired processes (testing and observability) upfront—and across functional domains
- Build a resilient and performant data platform
- Set and align on cross-organizational data SLAs

Without these steps, data teams will have a challenging time achieving any semblance of reliable, high-quality data. Still, taking your data quality strategy from a siloed experience managed solely by data engineers and other upstream roles to something prioritized by your broader company is a gradual process. At the risk of sounding cliché: Rome wasn't built in a day, and neither is your data quality strategy.

In future chapters, we'll dive into the meat of how teams can invest in a more proactive and collaborative approach to data reliability with technology, processes, and culture in mind. Stay tuned!

Prospective Table of Contents (Subject to Change)

A note for Early Release readers

With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at jleonard@oreilly.com.

1. Why Data Quality Deserves Attention—Now
2. Architecting for Data Reliability
3. Conducting Root Cause Analysis on Your Data Pipelines
4. Preventing Broken Data Pipelines
5. Democratizing Data Quality
6. Tools of the Trade: Building Trustworthy Data Pipelines
7. Conclusion: Building for Data Trust

About the Authors

Barr Moses is the CEO and co-founder of Monte Carlo, a data reliability company. In her decade-long career in data, Barr has served as commander of a data intelligence unit in the Israeli Air Force, a consultant at Bain & Company, and VP of Operations at Gainsight, where she built and led their data and analytics team. The instructor of O'Reilly's first course on Data Observability, an emerging discipline in data engineering, Barr has worked with hundreds of data teams struggling with these problems. Inspired by her time in the analytics trenches, she is building a product literally dedicated to identifying, resolving, and preventing what she calls "data downtime," periods of time when data is missing, erroneous, or otherwise inaccurate. In other words: bad data. In this book, she shares her experiences and learnings on how today's data organizations can achieve high data quality at scale through technological, organization, and cultural best practices.

Lior Gavish is CTO and co-founder of Monte Carlo, a data reliability company backed by Accel, Redpoint, GGV, and other top Silicon Valley investors. Prior to Monte Carlo, Lior co-founded cybersecurity startup Sookasa, which was acquired by Barracuda in 2016. At Barracuda, Lior was SVP of Engineering, launching award-winning ML products for fraud prevention. Lior holds an MBA from Stanford and an MSC in Computer Science from Tel-Aviv University.

Molly Vorwerck is the Head of Content at Monte Carlo, a data reliability company. Prior to joining Monte Carlo, Molly served as editor-in-chief of the Uber Engineering Blog and lead program manager for Uber's Technical Brand team, where she spent countless hours helping engineers, data scientists, and analysts write and edit content about their technical work and experiences. She also led internal communications for Uber's Chief Technology Officer and strategy for Uber AI's Research Review Program. In her spare time, she freelances for USA Today, reads up on all the latest trends in data, and volunteers for the California Historical Society.