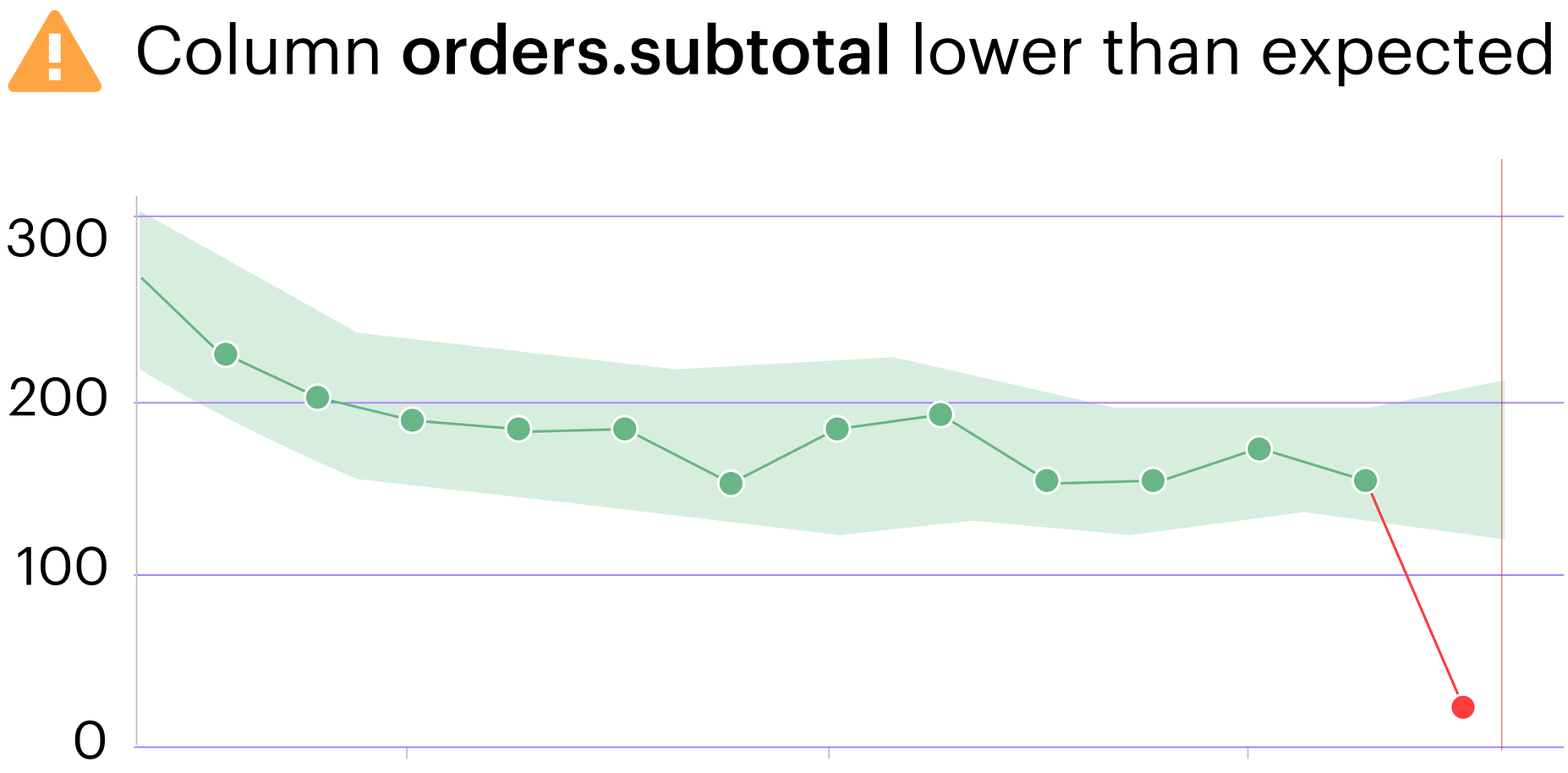
metaplane

Getting started with data observability



Improve data integrity, save engineering time

How to get started with data observability

Data observability may be a new buzz word, but problems related to data quality and losing trust in data have existed for thousands of years. One way to think about data observability is that it represents a technology that helps solve these problems.

By continuously monitoring the warehouse for data consistency and completeness, a data observability platform can empower data teams to proactively fix data quality issues and build trust in data so they can focus on creating value for their organization, rather than fighting data quality fires.

We break down getting started with data observability in a few steps:

0. The following guide assumes that you have a data warehouse and are actively using your data (e.g. analytics or data feeds back into business applications).

1. Add monitors across your warehouse to identify where your data quality issues are.

2. Use context from the platform to identify where and how to fix data quality issues.

3. Incorporate monitors into your continuous integration/deployment (CI/CD) transformation (modeling) workflow to prevent future incidents.

Stay aware of Schema Drift – The number one culprit

You've probably experienced this for yourself. You find a broken dashboard and laboriously query the reference objects only to discover that your null values weren't introduced by a data collection error, but rather, a simple column name change.

Particularly as teams get bigger and more tools are adopted, schema changes can occur from:

- **Upstream engineering changes**: A field that "no one is using" might be accidentally deprecated
- **Data loading tools**: Some tools merge schema changes into your warehouse by default, and may rename columns or change types altogether.
- **3rd party APIs**: 3rd party applications typically update field names and types
- **User input**: Custom field deprecations and simple typos can cause unintended downstream effects.

```
Tables (1 added, 1 deleted)
+ METAPLANE_DB.METAPLANE_SCHEMA.new_table
- METAPLANE_DB.METAPLANE_SCHEMA.old_table

Columns (1 added, 1 type changed, 1 deleted)
+ METAPLANE_DB.METAPLANE_SCHEMA.METAPLANE_TABLE.CITY
Δ METAPLANE_DB.METAPLANE_SCHEMA.METAPLANE_TABLE.PHONENUMBER: "NUMBER" → "TEXT"
- METAPLANE_DB.METAPLANE_SCHEMA.METAPLANE_TABLE.COLUMN_NAME
```

Tip: It's important to be aware of schema changes from your raw tables down to your modeled tables. Depending on your modeling strategy and adoption of self-service data usage, schema changes can impact entire business units.

Finding data quality incidents with machine learning based monitors

Adding tests to your data can be daunting. It involves coding, test type selection, acceptance threshold scoping, and ongoing maintenance.

Metaplane suggests a machine learning (ML) based approach, where hundreds of monitors can be programmatically added. Manual threshold setting is unnecessary as ML models monitor data based on its historical and current behavior, and can be done in as few as 3 days. This approach increases coverage and improves understanding of data behavior over time.

We recommend starting with baseline monitors for data volume and freshness to catch silent data bugs[1]. Then add more advanced monitors based on data characteristics.

Adding early indicator monitors

Two of the most common issues caused by silent data bugs are inconsistent data volumes and delayed data. When either of these problems occur, your data consumers are using inaccurate or out-of-date data.

Rather than receive questions from your teammates like “why does this dashboard look wrong?” or “why are our marketing automations broken?”, a data observability platform monitors these issues and helps to resolve them quickly to preserve trust in your data.

To change from being reactive to proactive, we recommend adding baseline monitors for data volume and freshness.

[1] **Silent data bugs** occur when source and ETL systems **appear to be operating, but the underlying data is broken**. For example, your ETL systems have finished landing data in the warehouse, but the volume of data is incorrect, schemas have changed, or the distribution of the data has drifted, to name a few common scenarios.

Data freshness

One of the most common issues data teams experience is stale data - there is data in the warehouse, but it has not been updated and is therefore inaccurate. Metaplane can automatically monitor the frequency at which a table has been updated and ensure that the data is updated at a consistent cadence.

Proactively catches: delayed Fivetran or Stitch jobs, broken dbt models that do not run, source API outages

Data volume (row count)

Data teams work hard to move data from source systems into the warehouse and beyond. It is common for inconsistent volumes of data to land or be used in the warehouse due to replication issues, transformation logic, or transient source issues. The majority of ingestion and transformation pipelines reference consistent source “objects” (ie files in s3 or tables) in regularly used scripts or queries, so a change in volume may indicate a change in the referenced object and potential incident.

Proactively catches: a table name change causing a transformation error, source systems incorrectly replicated to the warehouse, ETL systems incorrectly transforming or moving data

Adding advanced monitors

After adding baseline monitors, Metaplane recommends adding more advanced monitors on the characteristics of the data itself. These monitors should be applied to data that is well understood in regards to data types or downstream usage.

Numeric distribution

Storing critical metrics for operational teams like sales, marketing, customer success, and support is one of the most impactful use cases for a data warehouse.

The best way to ensure these metrics are not drifting over time, and therefore not breaking downstream automations and decisions, is to monitor the distribution of the data. Metaplane can automatically monitor the mean, minimum, maximum, and standard deviation of data with the click of a button.

Proactively catches: incorrect manual entry in an upstream sales tool (e.g. a deal was closed for $10m instead of $1m), bugs introduced into upstream product databases (e.g. total sale price becomes $0), transformation logic that incorrectly manipulates or casts data.

Type	Description	Example Use Case
Mean	Calculates the arithmetic average of a column	A dramatic change in mean for average order price over one day could indicate massive success for the business or a data incident caused by upstream data entry errors.
Minimum	Tracks the lowest value of a numeric column	In an e-commerce orders table, track the minimum order value to ensure no bugs have been introduced into a checkout flow.
Maximum	Tracks the highest value of a numeric column	In a sales deals table, track the maximum deal value to identify the root cause of anomalous changes in revenue.
Sum	Tracks the sum of a numeric column and alerts when the value is higher or lower than expected.	In a product events table, sum important product usage metrics over time. In a revenue table, sum revenue over a rolling window to ensure consistent daily, weekly, and monthly growth.
Percent Zero	Percent zero tracks the percentage of values equal to zero in a given numeric column.	This may be important for tracking daily generated sales leads, where an increase in this value over time might indicate a failing sales pipelines.
Percent Negative	Percent negative tracks the percentage of values less than zero in a given numeric column.	Some values, such as a purchase amount for an e-commerce site should ideally never be negative. You can additionally track an anomalous increase in returns or customer churn.
Standard Deviation	Tracks standard deviation in a numeric column.	Let's assume historically you've received 5 (out of 5) star customer reviews, an uptick in 1 star reviews would change the variance and might indicate a segment of customers that ..are unhappy with a new product.

Nullness

An increase in null values is one of the most common silent data bugs - your ETL tool or transformations successfully landed some data in your warehouse, but a particular field contains an abundance of null values, which could be an error with source application database storage or just a pixel that isn't capturing records correctly. Whether you have data that should never be null, or data occasionally allowed to be null, Metaplane can automatically monitor this over time.

Proactively catches: transformation bugs that fail to successfully join data, upstream product bugs that cause data to be missing, ETL syncs that miss data.

Uniqueness

Some values that you randomly generate, such as session IDs, user IDs, and/or values used for primary keys, should always be unique (or mostly unique). An increase in the percentage of unique values within a column can cause problems with joins downstream and indicate an issue with value generation for that column or data generation overall.

Proactively catches: duplicate primary keys, duplicate data, data drift.

Cardinality

Warehouses often store categorical data such as locations of business, dates, product brands, and other important enumerations of data. To protect against unacceptable values, add cardinality monitors to ensure that the data consistently has an acceptable set of values.

Proactively catches: the incorrect addition of new enum or acceptable values, software bugs that introduce misspelled values or differing capitalization

---

How to monitor for your business requirements

Every business, over time, develops use cases in data quality monitoring unique to them. To address use cases for focusing monitors on certain segments of the data, or any other edge case you may have, Metaplane offers: Partitioned Monitors and Custom SQL monitors.

For example, you may have a revenue recognition method that requires data to be merged from two disparate systems, and then cross referenced with a third system that has projections. You may also want to just focus monitoring on the

Partition monitors

By default, Metaplane applies monitors to tables and columns in the aggregate, helpful for identifying anomalous behavior across your entire warehouse. For your most critical data, it's important to monitor specific segments; for example:

- **Sales/Revenue**: Monitor changes in revenue by important properties like regions or sales reps, depending on your business. This helps identify insights and track performance accurately.

- **Marketing**: Monitor volume, freshness, and distribution of data based on marketing channels to identify replication issues or changes in marketing performance. For example, accurately replicating data from Facebook but not LinkedIn can cause downstream conversion reporting mistakes.

- **Product**: Ensure consistent and complete data for every product type, especially for product events. Segmenting and monitoring this data helps identify silent data bugs or changes in product behavior.

We recommend using these monitors for any raw tables used in the analytics use cases above.

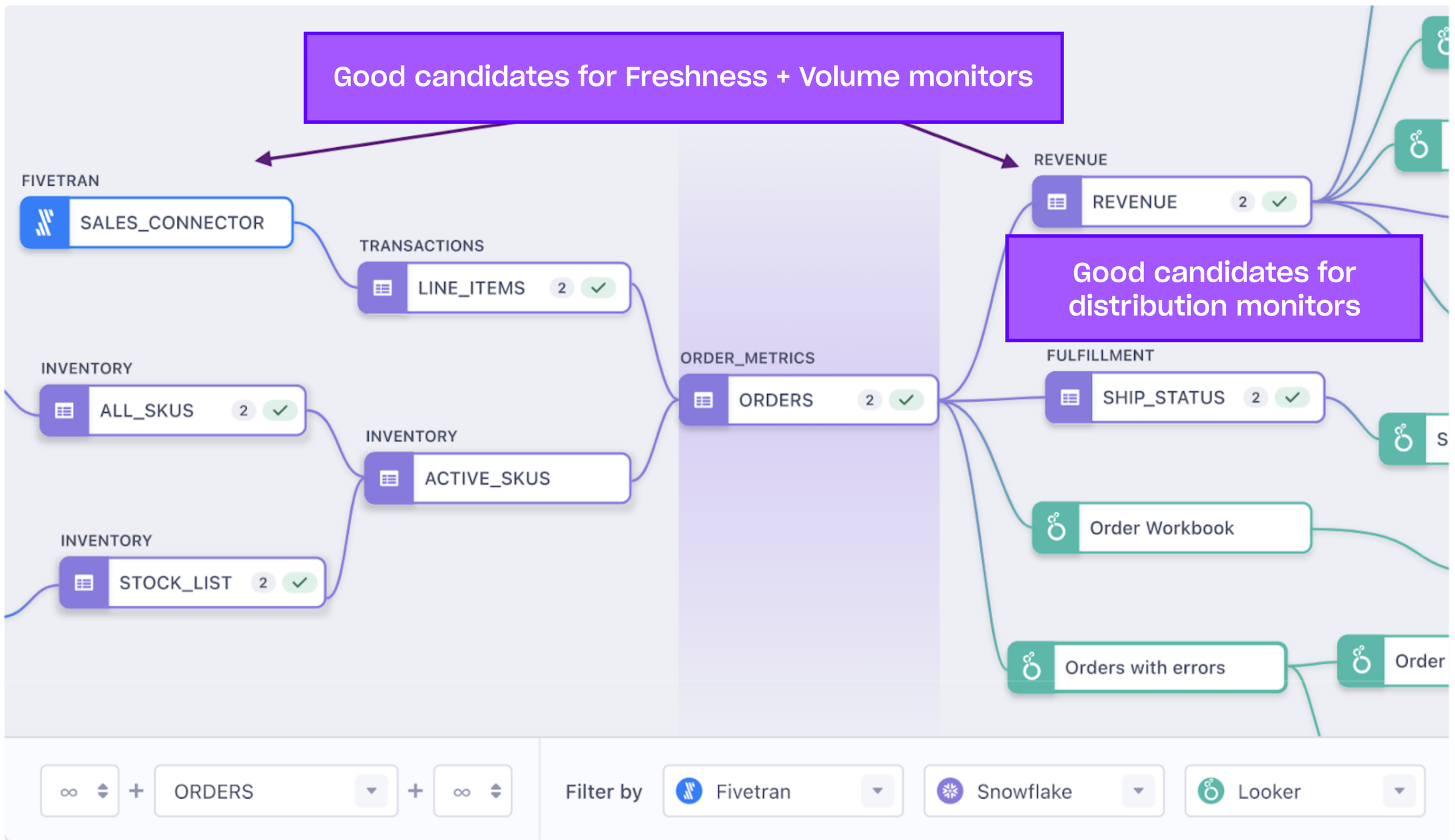
Custom SQL Monitors

Metaplane can apply the same automated machine learning capabilities to any query that runs in your warehouse and returns a scalar value or a set of values. Use custom SQL monitors to observe anything in the warehouse that is not covered by our one-click enabled monitors, such as:

- **Metrics that aren't represented by one column:** if you want to monitor a business metric, such as a conversion rate, that is not materialized in one column, simply write a SQL query that calculates this metric.
- **Data reconciliation across multiple tables:** ou may want to ensure that a 1:1 relationship of two tables is maintained, with simple count(*) statements.

Proactively catches: metric drift, acceptable value failures, warehouse cost increases, warehouse queue backlogs, your unique business logic.

Where to add monitors



In some cases, you may want coverage across every single object within your warehouse, but there may be some setups, such as when you're not actively using every object, that you may only want to focus on the objects that are most relevant to your business.

The most obvious place to deploy monitors is wherever you've most recently had a fire. If you haven't had any recent fires OR are looking for your next target to place monitors, here are a few steps that you can take:

1. **Find an important "object":** Find a dashboard or table that's regularly checked, either with query usage insights from Metaplane, or by asking your data team.

2. **Go upstream:** Using the lineage graphs offered by a Data Observability platform like Metaplane, you'll be able to visually find a schema with raw data tables that are heavily referenced by your important object from Step 1.

3. **Monitor selection:** Apply freshness and volume monitors to your raw data tables from Step 2. Depending on their data types and expected values, you may also want to add distribution and other monitors.

4. **Expand** to more tables and monitor types

How to manage data incidents

Incident interactivity

Every organization's data behaves in different ways. That's why a data observability tool needs to use machine learning, trained on your organization's data, to learn about typical data behavior, and predict how the data should behave in the future.

Data inevitably changes due to seasonality, backfilled data, or SLA variations. Static thresholds create monitoring nightmares, while ML-based calculations shine with minor human inputs. Metaplane, a data observability platform, offers user-friendly ways to provide instant feedback for immediate model adjustments, eliminating manual threshold calculations and updates. Metaplane provides the following options:

Mark as normal so that the most recent data point, and data points within a similar range, will be included into future models.

Mark as "Disabled" to turn it off. It will retain past training data in the case you decide to turn it on again.

Reset monitor: In extreme cases, such as significantly changing the volume or distribution of data in a table, or refactoring entire DAG workflows, you can retrain the monitor on new behavior.

How to manage data incidents

After setting up baseline and advanced tests, a data observability platform helps organizations manage data incidents.

A data incident is any data related issue that results in degraded data quality, causing downstream data consumers to use incorrect or out-of-date data.

Data incidents need to be communicated in a timely manner, provide enough context to understand priority, help loop in teammates and data consumers, and be interactive so that your data monitoring can evolve with your data over time.

Incident context helps you answer questions like what caused this issue? What downstream tables and dashboards are impacted? Should I drop what I'm doing and fix the problem?

A collaborative incident empowers data teams to pull in data engineers to triage issues and makes it simple to broadcast updates to data consumers and stakeholders.

Every incident needs to be interactive so that you can easily give feedback to the machine learning models, mute transient issues that are being fixed, or integrate with workflow automations.

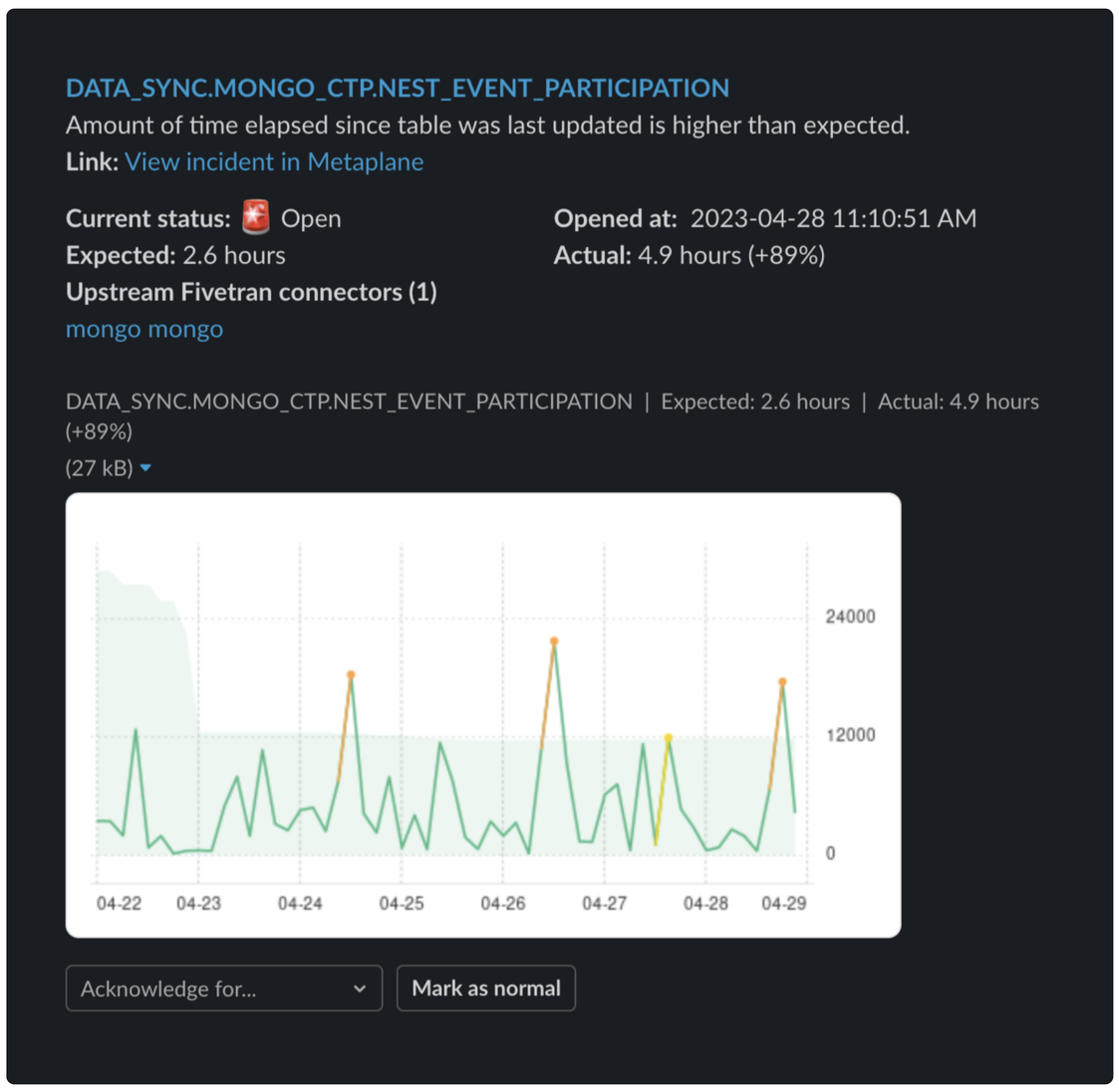
How to manage data incidents

Incident context

Every data incident needs to provide enough context so the data team knows what caused the incident and what is impacted downstream. It’s not enough to just find out about these issues - a data engineer needs context so they can prioritize incidents relative to the other work on their plate.

To help prioritize incidents, a data observability platform can provide context like:

- What is upstream from the impacted data? Could this be the root cause?
- What related data is also broken? How widespread is this issue?
- What is downstream from the impacted data? Are there dashboards or marketing automations that are now broken?



Metaplane tracks your team’s history of incidents over time. An incident can group together related issues, impacted BI dashboards, and provide a way to interact with the machine learning models.

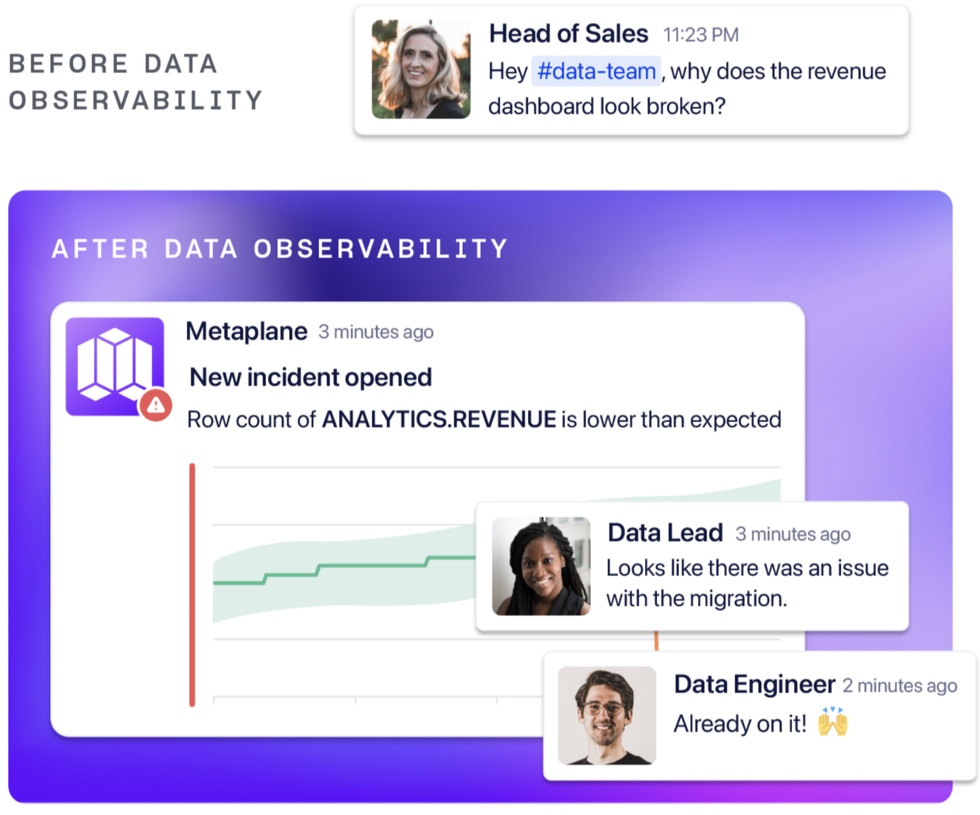
How to manage data incidents

Incident collaboration

Every incident is an opportunity to preserve trust in your data. Rather than being the last to find out about data quality issues, data observability tools help your team flip the script and proactively let teammates and stakeholders know about incidents.

A data observability platform can make collaborating on incidents easier by:

- Sending you alerts where your team already works
- Allowing data teams to configure incident alert destinations
- Providing a way to acknowledge incidents and easily tag teammates
- Making it easy to broadcast messages to teams that rely on the data impacted by the incident



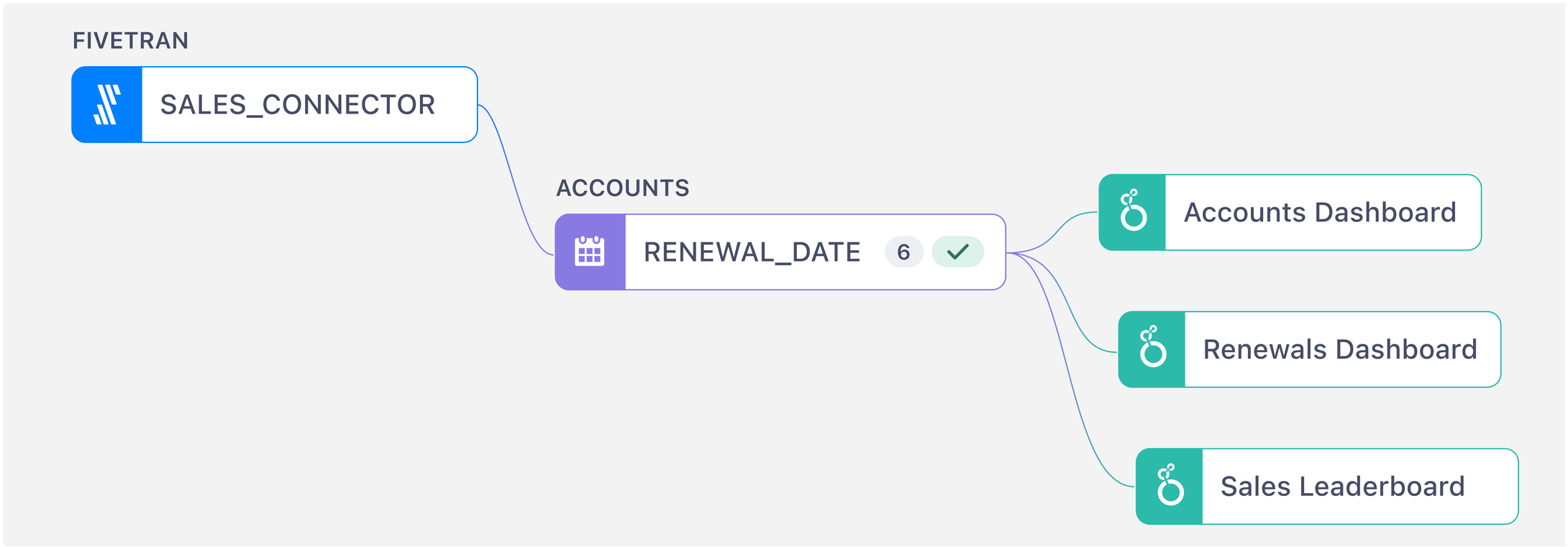
Configuring incident alerting channels and workflows is vital as your data team scales. It ensures relevant notifications are seen and handled promptly. Two commons ways to do this are:

- **Team-based tag example**: You can tag objects used in engineering analytics as "eng", so that you can easily apply similar monitors across the entire set.
- **Importance based tagging example**: You could also tag key objects as "critical" or assign severity levels unique to each schema. Each incident severity level can then be routed to its own channel for faster prioritization.

TIP Invite at least 2 stakeholders per grouping of objects to a slack channel

Proactive incident prevention

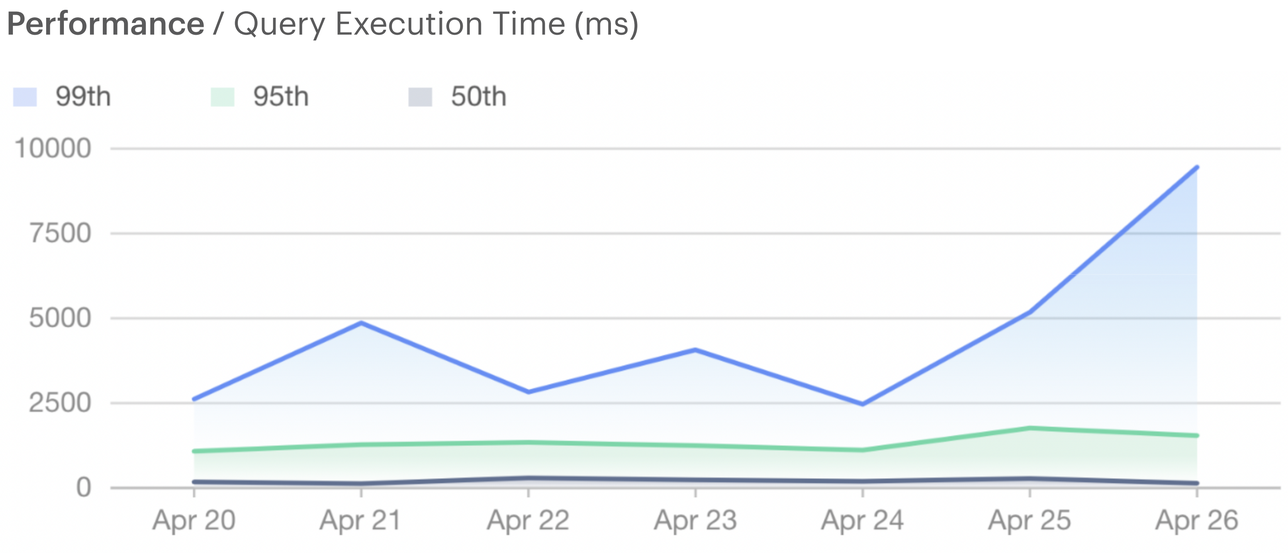
Lineage graphs for schema change planning

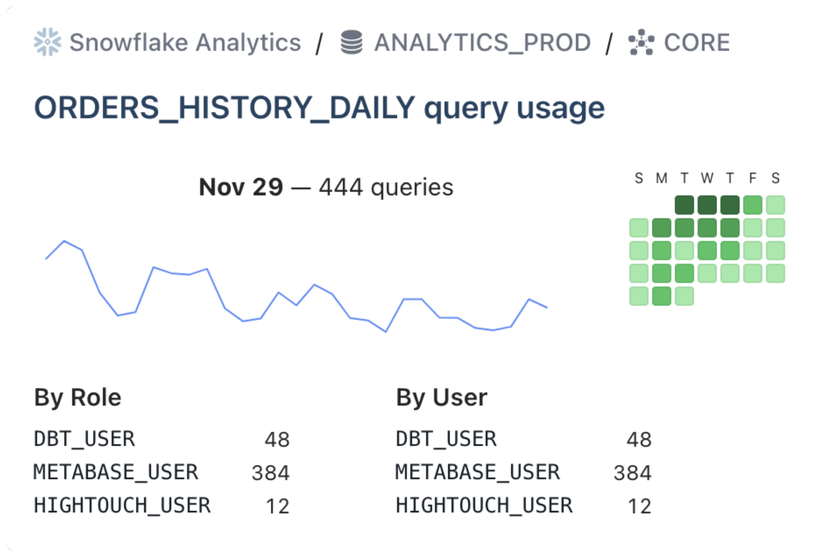


Column and table level lineage graphs have multiple uses, including understanding monitor deployment and preventing data incidents. These graphs show where tables and columns are referenced downstream, helping to avoid schema changes and identify queries that need updating.

Improve warehouse speed while reducing consumption costs

In addition to monitoring your data, monitoring your queries acting on your data is another key functionality to help you avoid incidents. Over time, queries can degrade in performance as their reference objects grow in size while optimization falls to the wayside. In the worst case, downstream job dependencies can cause failures when queries don't execute in time. With Data Observability, you'll be able to identify long query runtimes to help you focus your query cleanup efforts, **saving both (execution) time and money.**



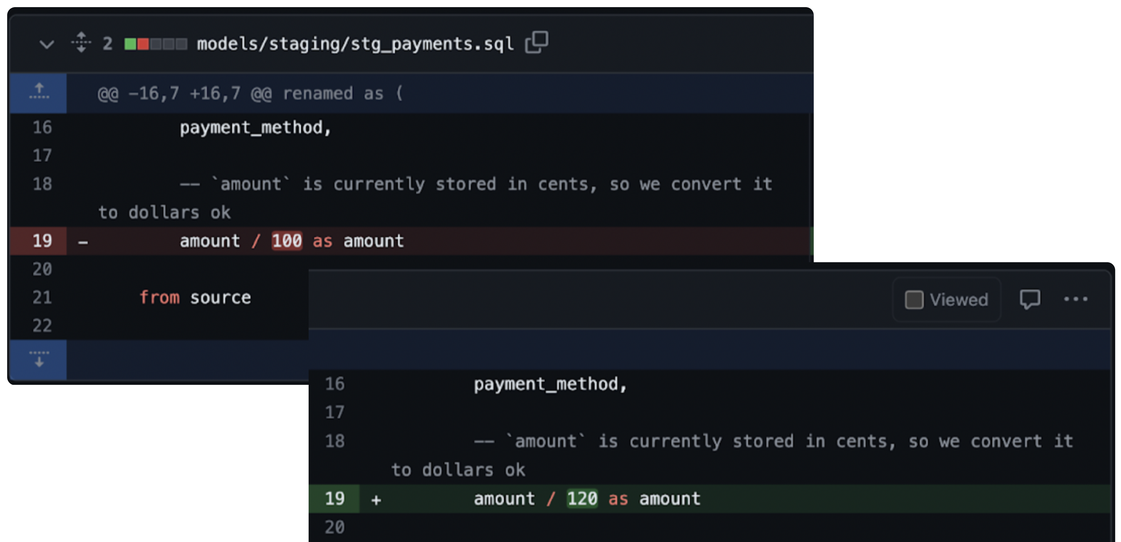


Avoid breaking reports and models with Data CI/CD

With our Github integration, you'll be able to understand the impact of any pending Pull Requests on your data.

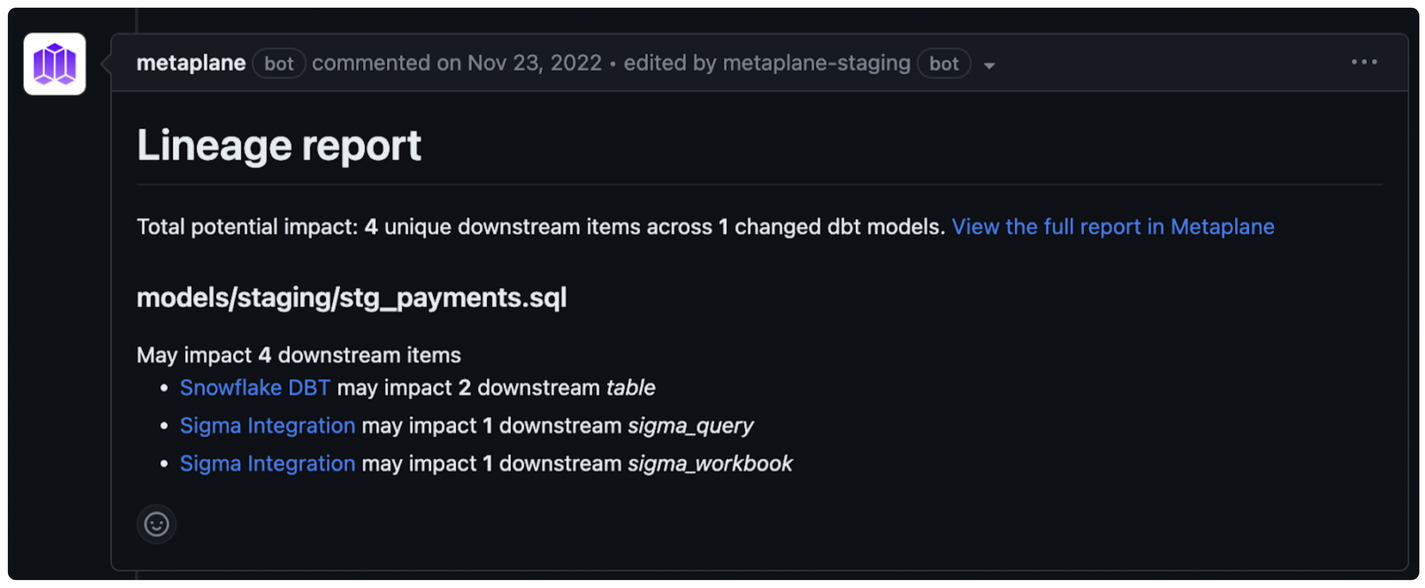
Query / Model Change

In this example, we've changed our calculation for converting a smaller currency denomination to a larger one (i.e. dividing by 100 → dividing by 120)



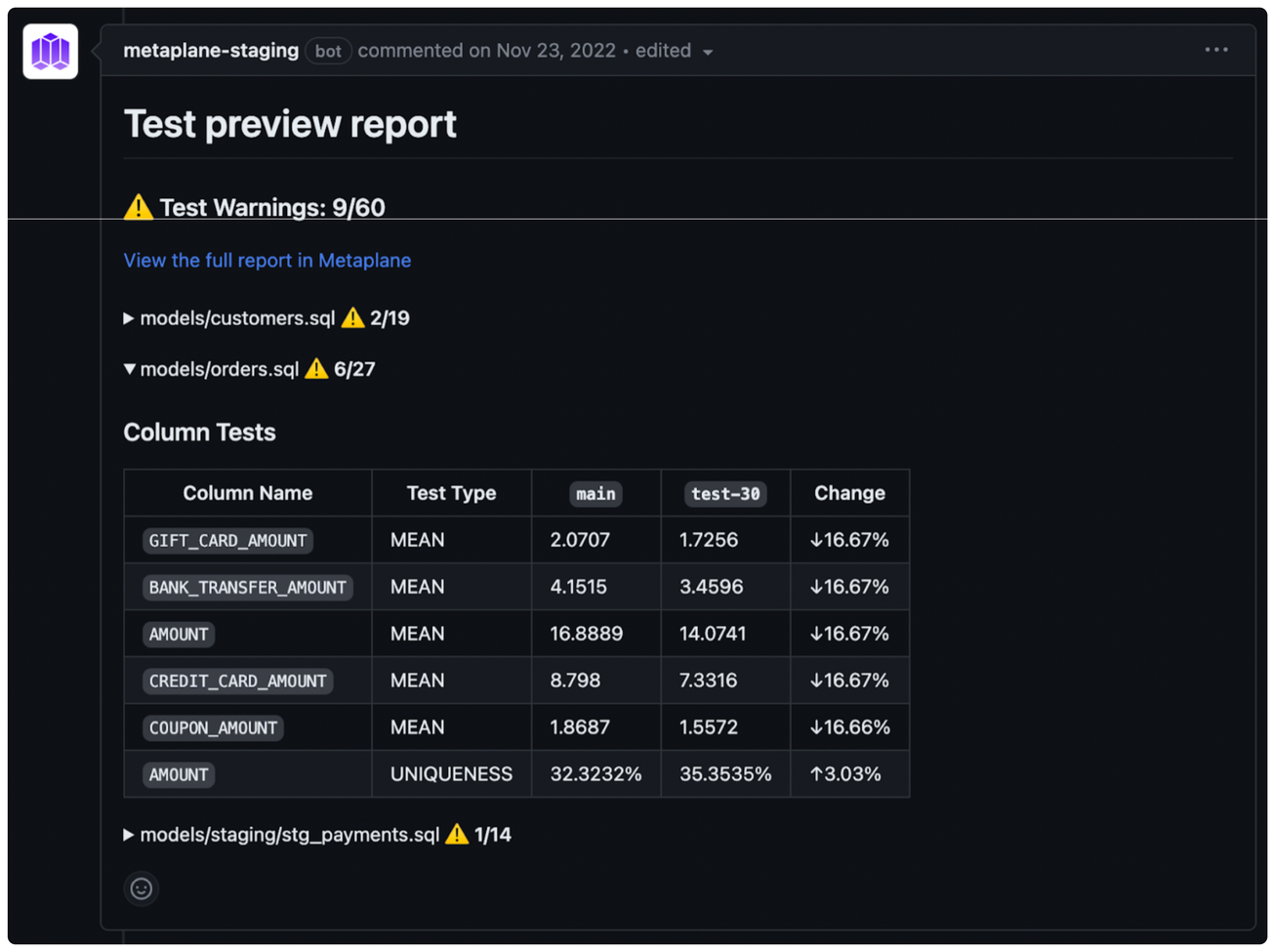
What objects are impacted?

You'll be able to immediately see that a few **other tables in the warehouse and workbooks (dashboards) in the business intelligence tool** will be impacted.



Does this affect any other models?

In this example, the `orders` model/table also depends on our `stg_payments` model, so we'll surface monitor breaches that will occur as a result of our pull request.



metaplane

metaplane.dev/signup

Implementation in **hours**,
high quality data **forever.**