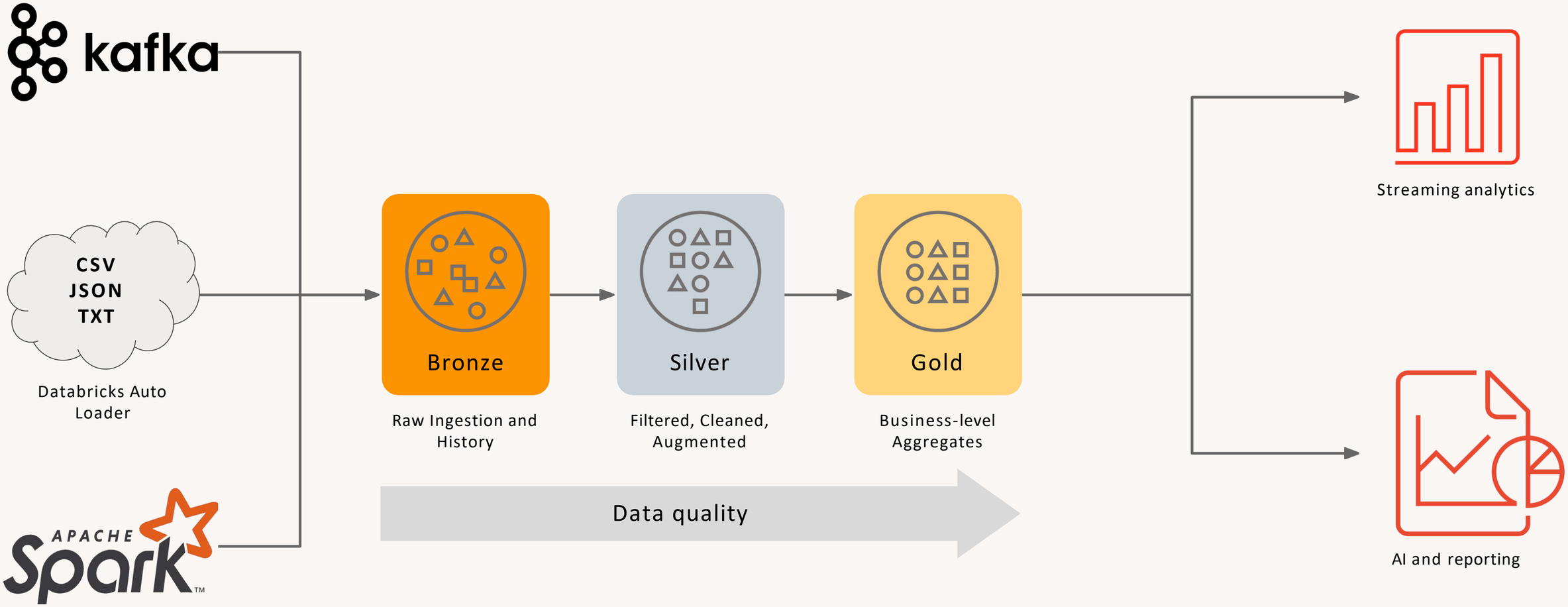
databricks

Build Data Pipelines with Delta Live Tables

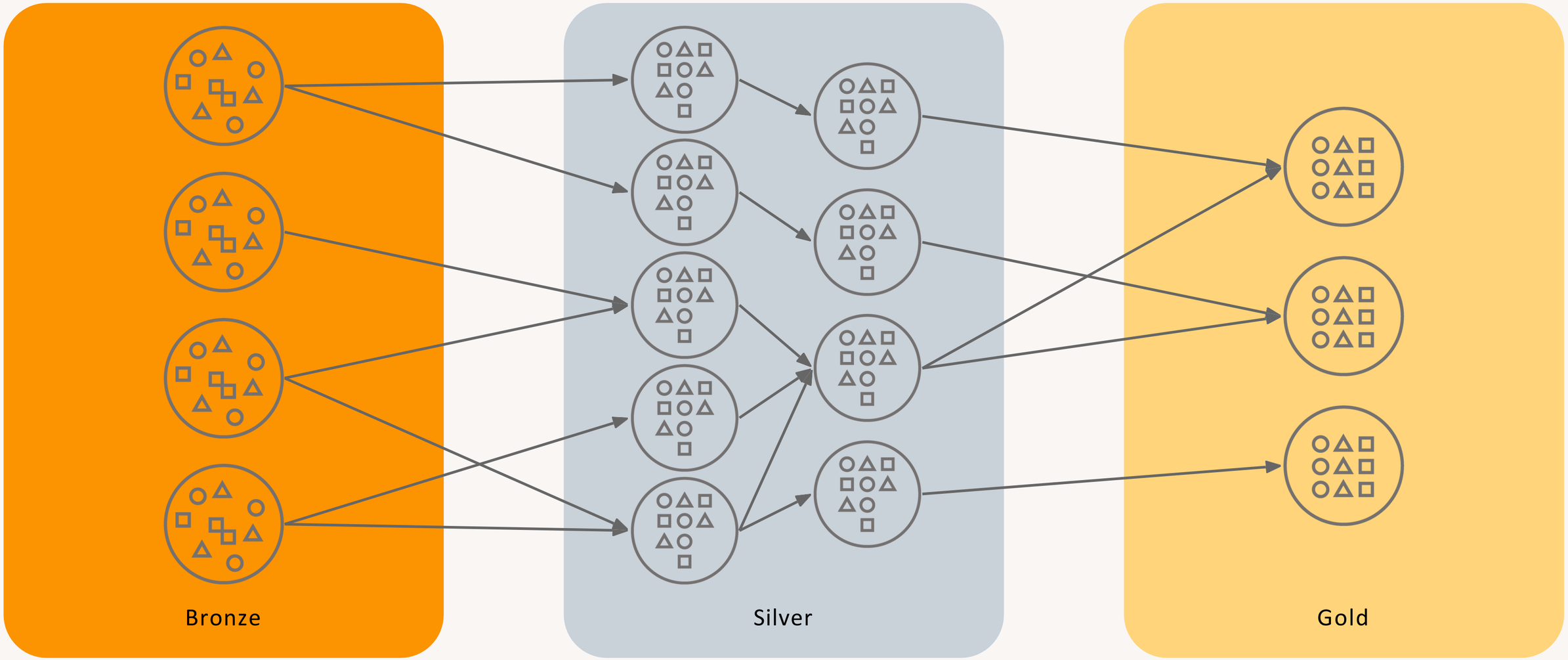
databricks

Introduction to Delta Live Tables

Multi-Hop in the Lakehouse



The Reality is Not so Simple



Large scale ETL is complex and brittle

Complex pipeline development

Hard to build and maintain table **dependencies**

Difficult to switch between **batch** and **stream** processing

Data quality and governance

Difficult to monitor and enforce **data quality**

Impossible to trace data **lineage**

Difficult pipeline operations

Poor **observability** at granular, data level

Error handling and **recovery** is laborious

Introducing Delta Live Tables

Make reliable ETL easy on Delta Lake

Operate with agility

Declarative tools to build batch and streaming data pipelines

Trust your data

DLT has built-in declarative quality controls

Declare quality expectations and actions to take

Scale with reliability

Easily scale infrastructure alongside your data

What is a LIVE TABLE?

What is a Live Table?

Live Tables are materialized views for the lakehouse.

A live table is:

- Defined by a SQL query
- Created and kept up-to-date by a pipeline

```
CREATE OR REPLACE LIVE TABLE report
AS SELECT sum(profit)
FROM prod.sales
```

Live tables provides tools to:

- Manage dependencies
- Control quality
- Automate operations
- Simplify collaboration
- Save costs
- Reduce latency

What is a Streaming Live Table?

Based on Spark™ Structured Streaming

A streaming live table is "stateful":

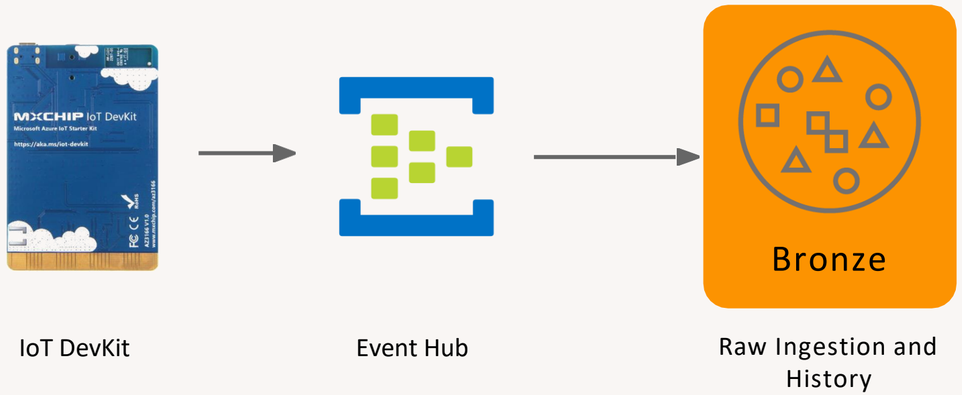
- Ensures exactly-once processing of input rows
- Inputs are only read once

```
CREATE STREAMING LIVE TABLE report
AS SELECT sum(profit)
FROM cloud_files(prod.sales)
```

- Streaming Live tables compute results over append-only streams such as Kafka, Kinesis, or Auto Loader (files on cloud storage)
- Streaming live tables allow you to reduce costs and latency by avoiding reprocessing of old data.

Demo Delta Live Tables

Demo: Live Tables



IoT DevKit

Event Hub

Raw Ingestion and History

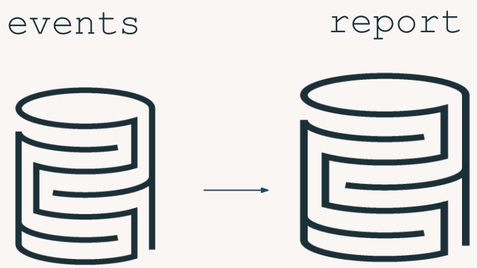
What if I have many tables?

Declare LIVE Dependencies

Using the LIVE virtual schema.

```
CREATE LIVE TABLE events
AS SELECT … FROM raw_data
```

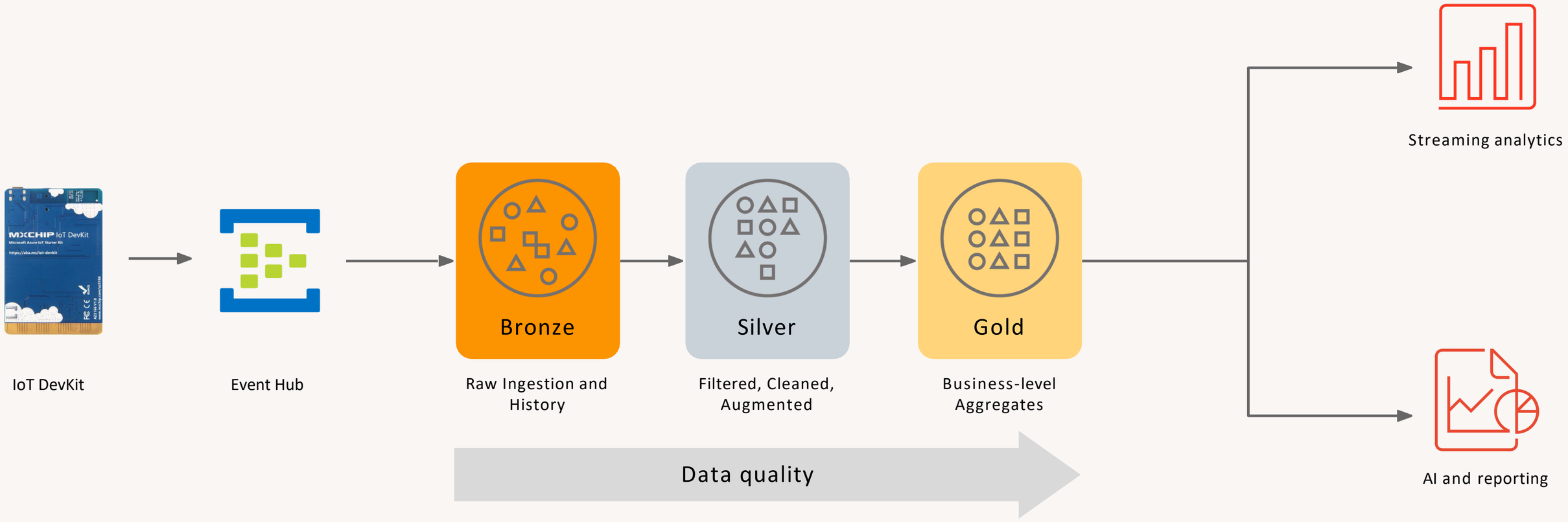
```
CREATE LIVE TABLE report
AS SELECT … FROM LIVE.events
```



- Dependencies owned by other producers are just read from the catalog or spark data source as normal.
- LIVE dependencies, from the **same pipeline**, are read from the LIVE schema.
- DLT detects LIVE dependencies and executes all operations in **correct order**.
- DLT handles parallelism and captures the lineage of the data.

Demo Delta Live Tables

Demo: Live Tables



How do I know my results are correct?

BEST PRACTICE

Ensure correctness with Expectations

Expectations are tests that ensure data quality in production

```
CONSTRAINT valid_timestamp
EXPECT (timestamp > '2012-01-01')
ON VIOLATION DROP
```

```
@dlt.expect_or_drop(
  "valid_timestamp",
  col("timestamp") > '2012-01-01')
```

Expectations are true/false expressions that are used to validate each row during processing.

DLT offers flexible policies on how to handle records that violate expectations:

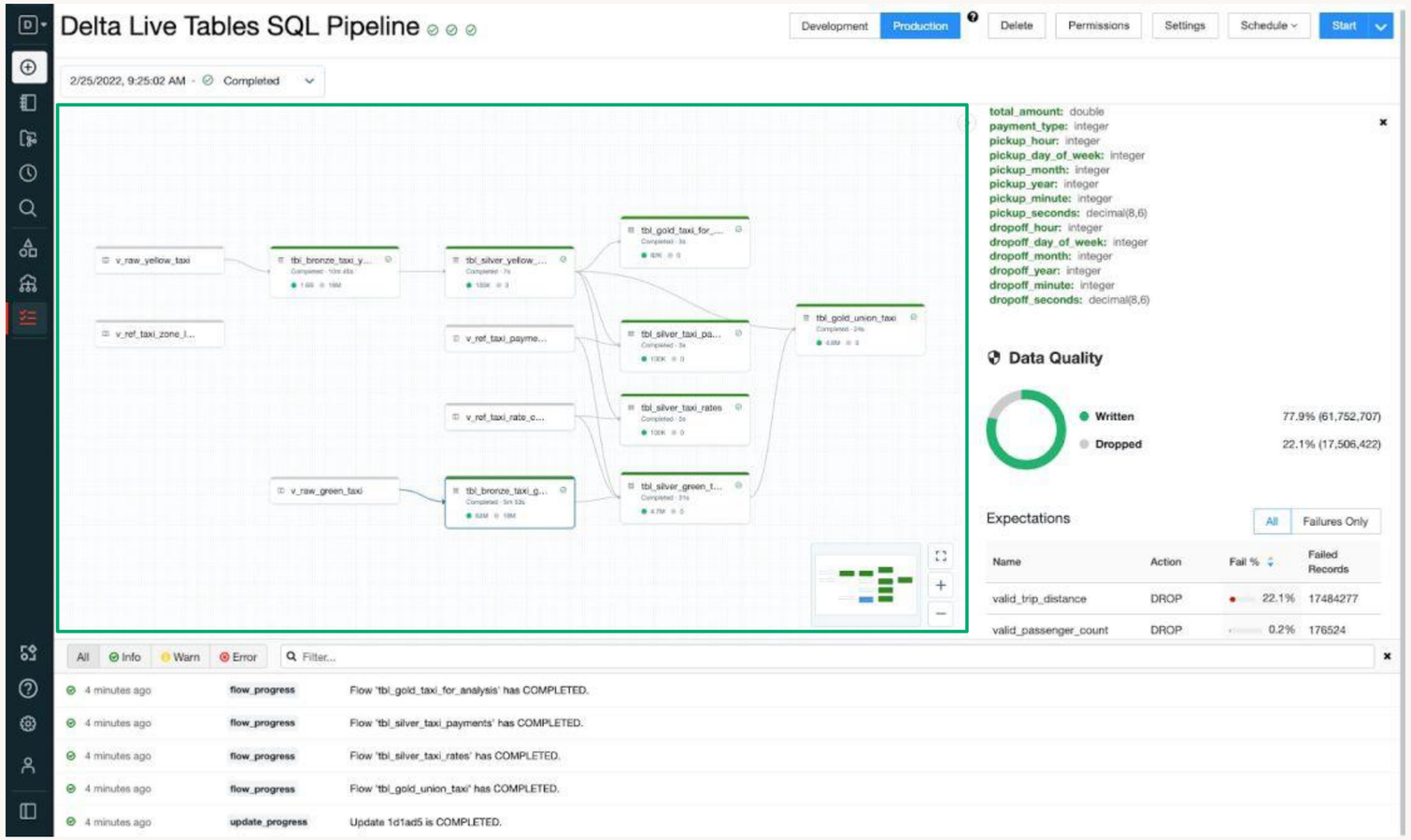
- Track number of bad records
- Drop bad records
- Abort processing for a single bad record

What about operations?

Pipelines UI

A one stop shop for ETL debugging and operations

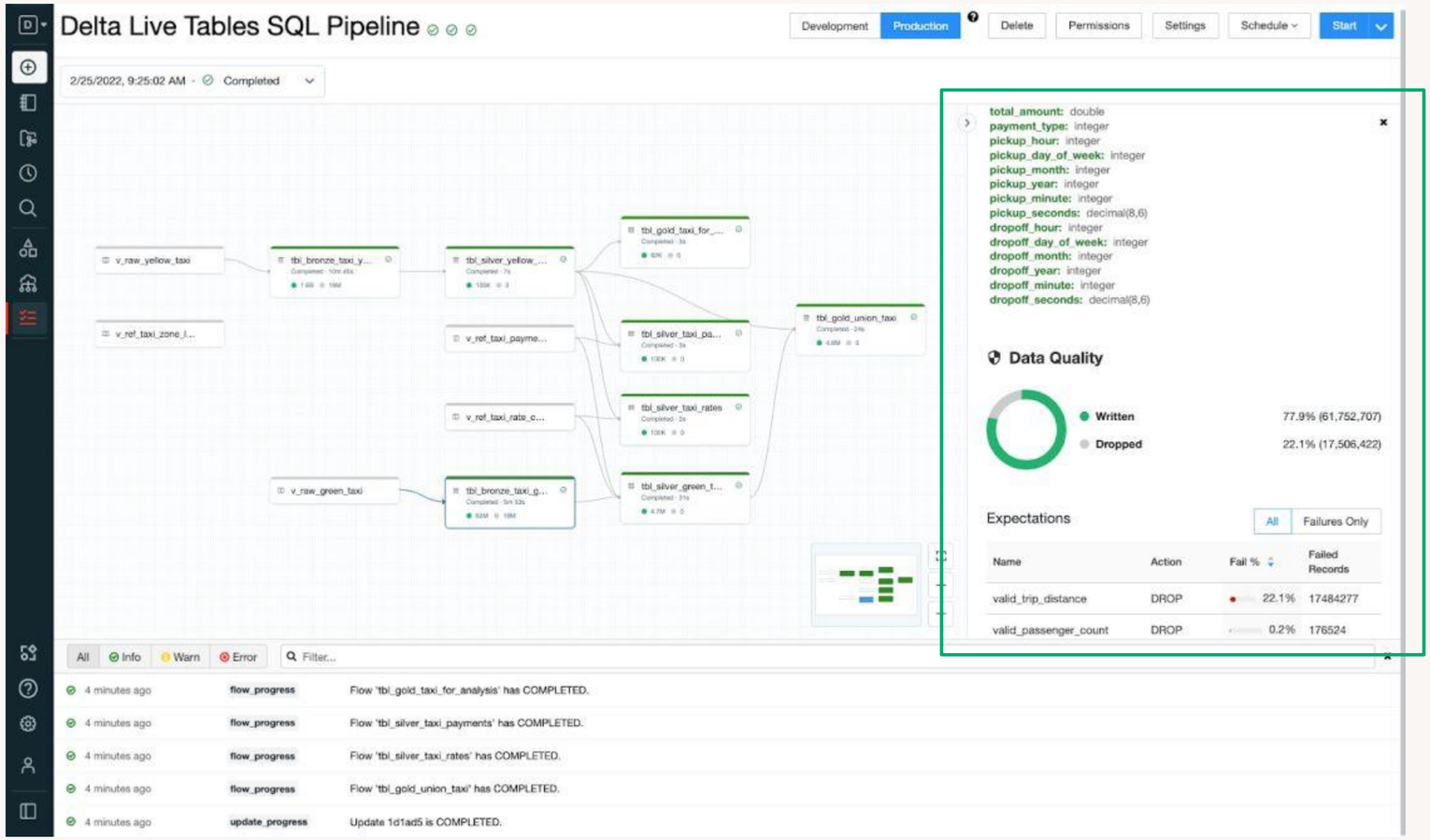
- Visualize data flows between tables



Pipelines UI

A one stop shop for ETL debugging and operations

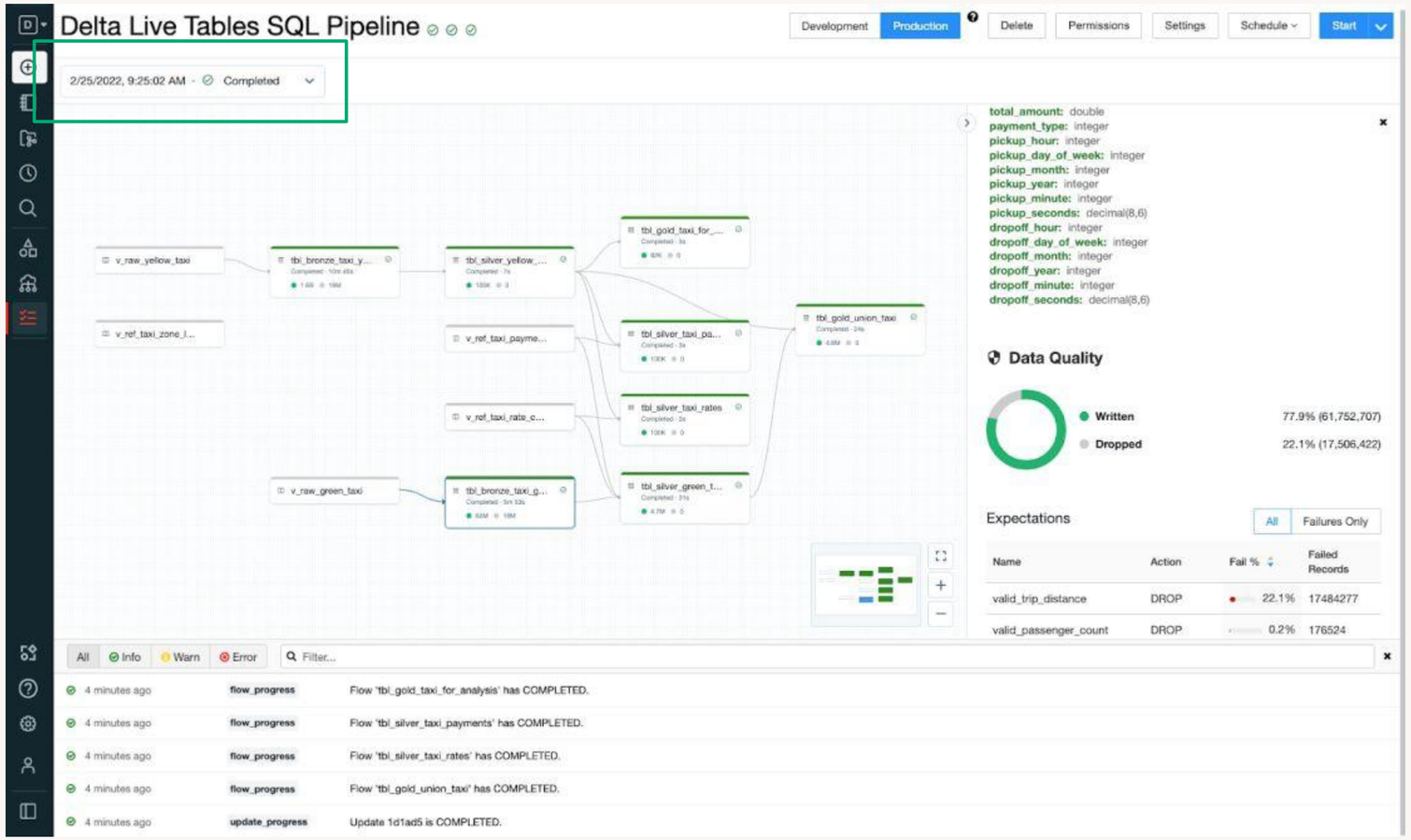
- Visualize data flows between tables
- Discover metadata and quality of each table



Pipelines UI

A one stop shop for ETL debugging and operations

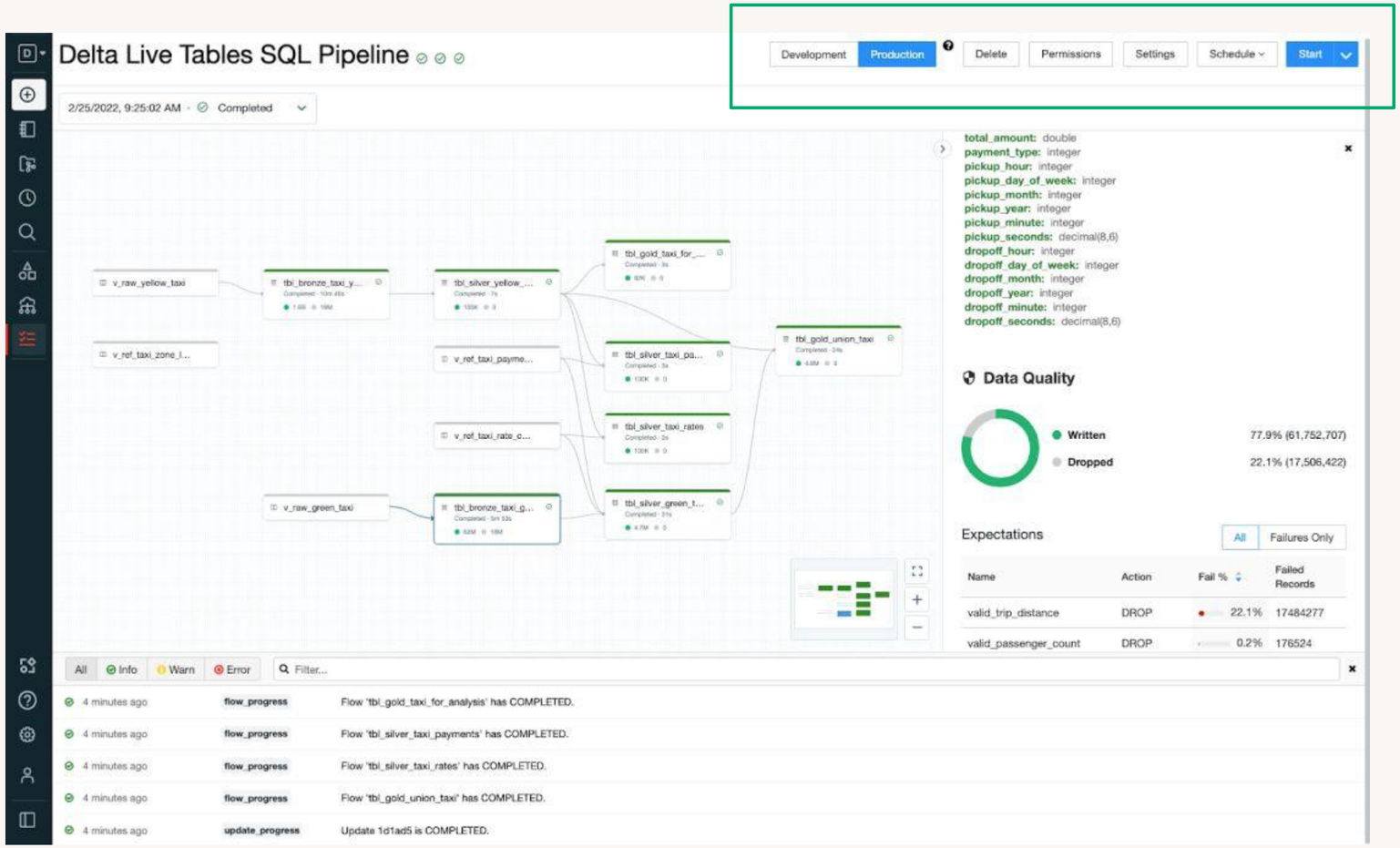
- Visualize data flows between tables
- Discover metadata and quality of each table
- Access to historical updates



Pipelines UI

A one stop shop for ETL debugging and operations

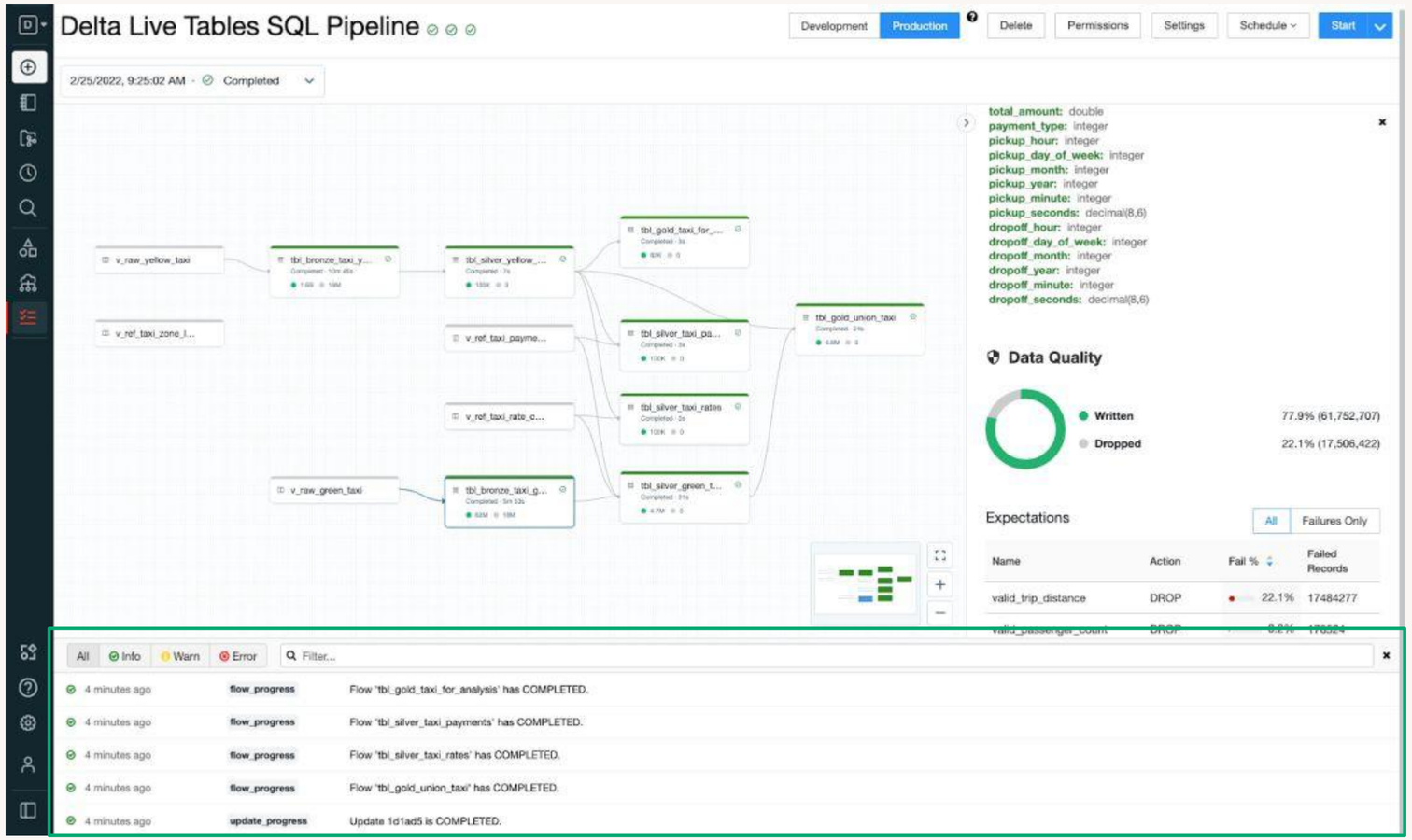
- Visualize data flows between tables
- Discover metadata and quality of each table
- Access to historical updates
- Control operations



Pipelines UI

A one stop shop for ETL debugging and operations

- Visualize data flows between tables
- Discover metadata and quality of each table
- Access to historical updates
- Control operations
- Dive deep into events



The Event Log

The event log automatically records all pipelines operations.

Operational Statistics

Time and current status, for all operations

Pipeline and cluster configurations

Row counts

Provenance

Table schemas, definitions, and declared properties

Table-level lineage

Query plans used to update tables

Data Quality

Expectation pass / failure / drop statistics

Input/Output rows that caused expectation failures

What do I *no longer* need to manage with DLT?

Automated Data Management

DLT automatically optimizes data for performance & ease-of-use

Best Practices

What:

DLT encodes Delta best practices automatically when creating DLT tables.

How:

DLT sets the following properties:

- `optimizeWrite`
- `autoCompact`
- `tuneFileSizesForRewrites`

Physical Data

What:

DLT automatically manages your physical data to minimize cost and optimize performance.

How:

- runs vacuum daily
- runs optimize daily

You still can tell us how you want it organized (ie ZORDER)

Schema Evolution

What:

Schema evolution is handled for you

How:

Modifying a live table transformation to add/remove/rename a column will automatically do the right thing.

When removing a column in a streaming live table, old values are preserved.