datapath

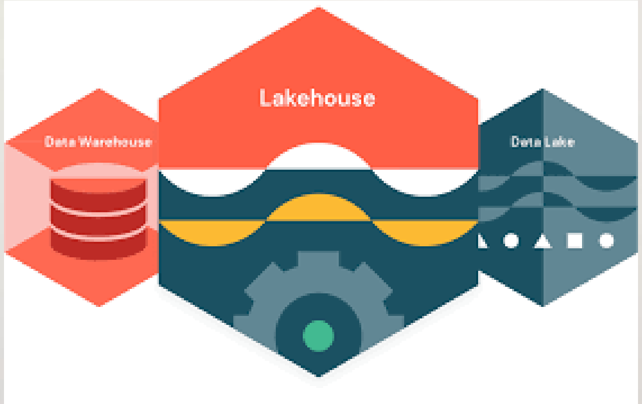
Administrando los datos con Delta Lake

www.datapath.ai

Introducción

www.datapath.ai

Delta Lake

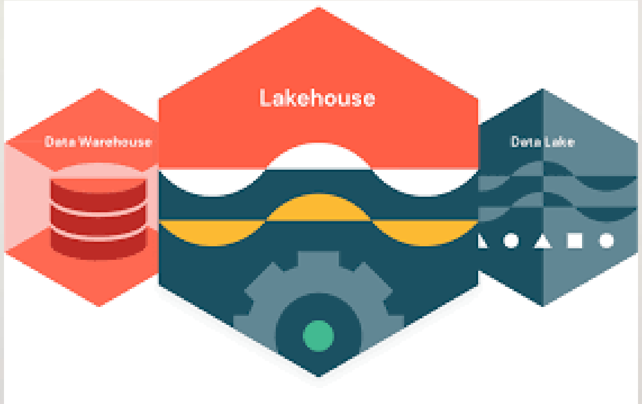


- Es un proyecto de código abierto que permite construir un Data Lakehouse sobre los sistemas de almacenamiento actuales.

Delta Lake no es:

- Tecnología propietaria
- Formato de almacenamiento
- Un medio de almacenamiento
- Servicio de base de datos o Data Warehouse.

Delta Lake



Delta Lake es:

- Codigo Abierto
- Construido sobre formatos de dato estándar.
- Optimizado para el almacenamiento de objectos en la nube.
- Construido para el manejo escalable de metadata.

Transacciones ACID

- Atomicity
- Consistency
- Isolation
- Durability

DELTA LAKE

Problemas resueltos por ACID

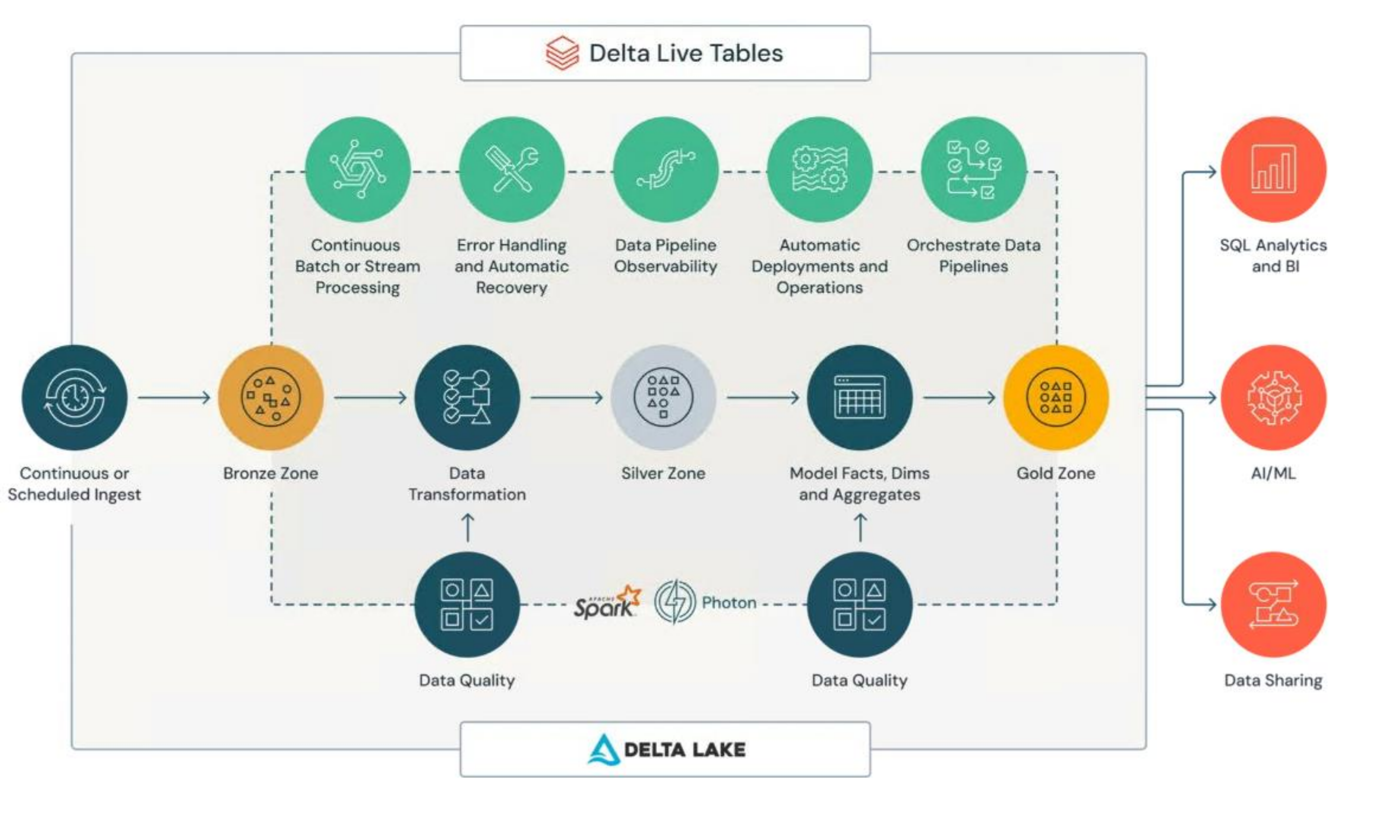
1. **Datos difíciles de apilar.**
2. **Modificación de datos difíciles de modificar.**
3. **Jobs que se caen a mitad de procesamiento.**
4. **Operaciones Real-Time**
5. **Data Histórica costosa de mantener.**

Viene por defecto

datapath

Delta Lake is the default for all tables created in Databricks

Delta Live Tables



www.datapath.ai

Delta Live Tables

Efficient data ingestion

Building production-ready ETL pipelines on the lakehouse begins with ingestion. DLT powers easy, efficient ingestion for your entire team — from data engineers and Python developers to data scientists and SQL analysts. With DLT, load data from any data source supported by Apache Spark™ on Databricks.

- Use Auto Loader and streaming tables to incrementally land data into the Bronze layer for DLT pipelines or Databricks SQL queries
- Ingest from cloud storage, message buses and external systems
- Use change data capture (CDC) in DLT to update tables based on changes in source data

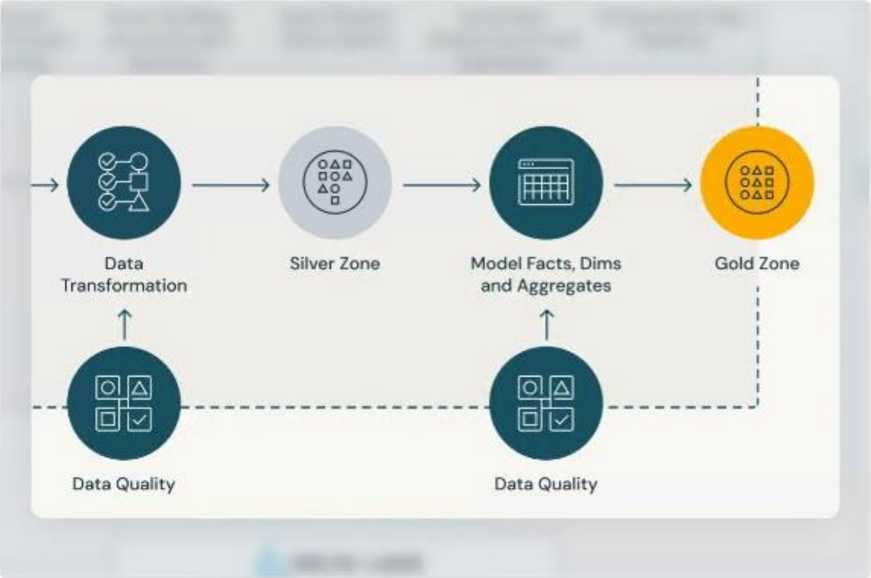
Delta Live Tables

datapath

Intelligent, cost-effective data transformation

From just a few lines of code, DLT determines the most efficient way to build and execute your streaming or batch data pipelines, optimizing for price/performance (nearly 4x Databricks baseline) while minimizing complexity.

- Instantly implement a streamlined medallion architecture with streaming tables and materialized views
- Optimize data quality for maximum business value with features like expectations
- Refresh pipelines in continuous or triggered mode to fit your data freshness needs



www.datapath.ai

"Delta Live Tables has helped our teams save time and effort in managing data at the multitrillion-record scale and continuously improves our AI engineering capability . . . Databricks is disrupting the ETL and data warehouse markets."

Delta Live Tables

Simple pipeline setup and maintenance

DLT pipelines simplify ETL development by automating away virtually all the inherent operational complexity. With DLT pipelines, engineers can focus on delivering high-quality data rather than operating and maintaining pipelines. DLT automatically handles:

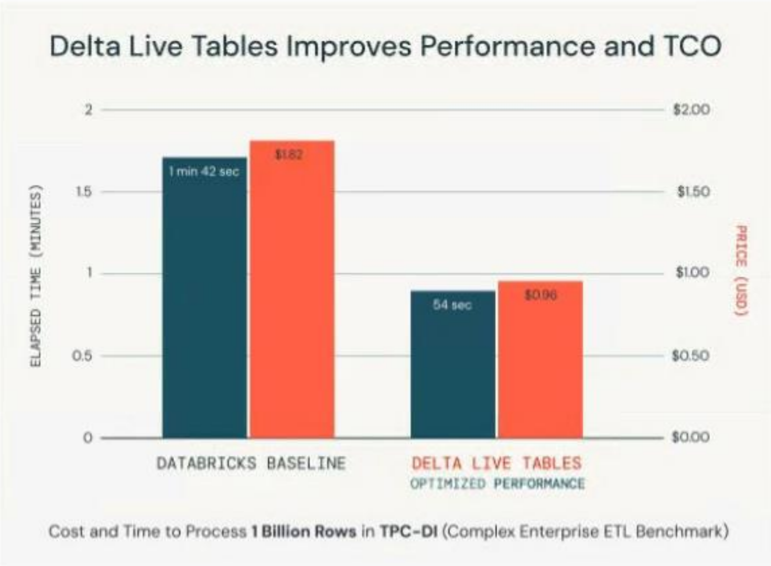
- Task orchestration
- CI/CD and version control
- Autoscaling compute infrastructure for cost savings
- Monitoring via metrics in the event log
- Error handling and failure recovery

Delta Live Tables

datapath

Next-gen stream processing engine

Spark Structured Streaming is the core technology that unlocks streaming DLT pipelines, providing a unified API for batch and stream processing. DLT pipelines leverage the inherent subsecond latency of Spark Structured Streaming, and record-breaking price/performance. Although you can manually build your own performant streaming pipelines with Spark Structured Streaming, DLT pipelines may provide faster time-to-value, better ongoing development velocity, and lower TCO because of the operational overhead they automatically manage.



www.datapath.ai

Comparition

Delta Live Tables pipelines vs. “build your own” Spark Structured Streaming pipelines

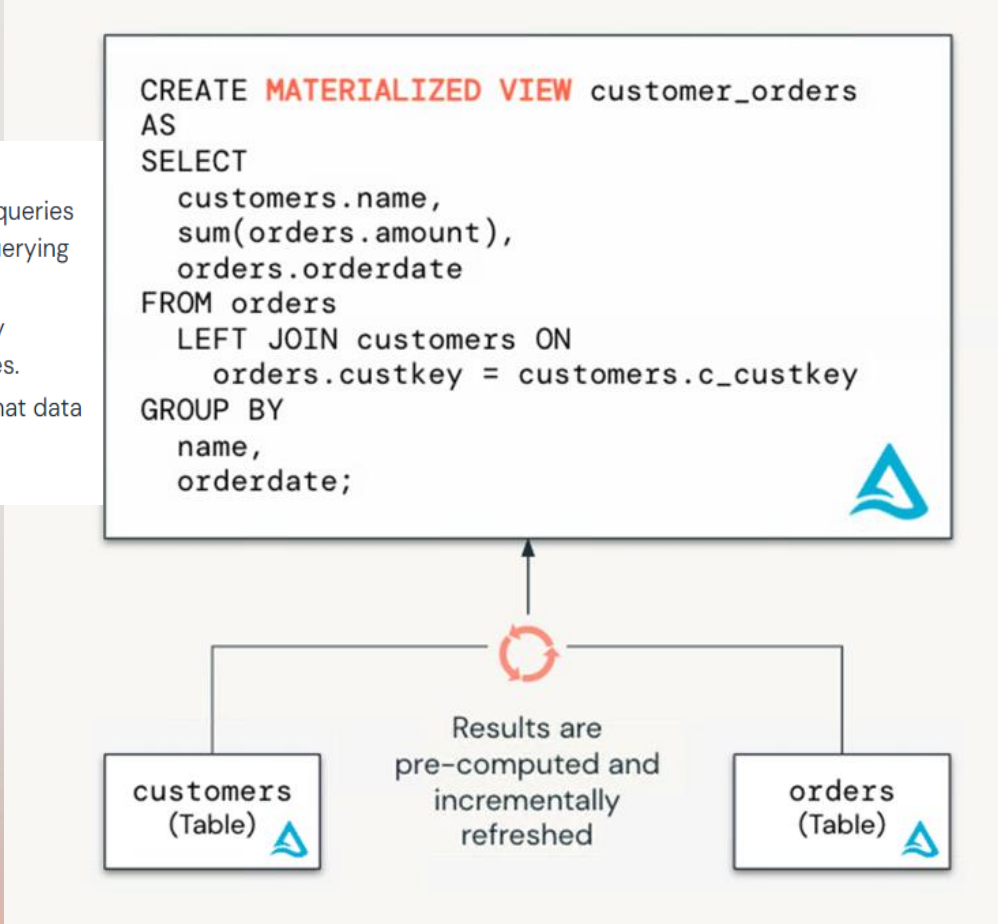
	Spark Structured Streaming pipelines	DLT pipelines
Run on the Databricks Lakehouse Platform	✔	✔
Powered by Spark Structured Streaming engine	✔	✔
Unity Catalog integration	✔	✔
Orchestrate with Databricks Workflows	✔	✔
Ingest from dozens of sources — from cloud storage to message buses	✔	✔
Dataflow orchestration	Manual	Automated
Data quality checks and assurance	Manual	Automated
Error handling and failure recovery	Manual	Automated
CI/CD and version control	Manual	Automated
Compute autoscaling	Basic	Enhanced

www.datapath.ai

Materialized View

Benefits of materialized views:

- Accelerate BI dashboards. Because MVs precompute data, end users' queries are much faster because they don't have to re-process the data by querying the base tables directly.
- Reduce data processing costs. MVs results are refreshed incrementally avoiding the need to completely rebuild the view when new data arrives.
- Improve data access control for secure sharing. More tightly govern what data can be seen by consumers by controlling access to base tables.

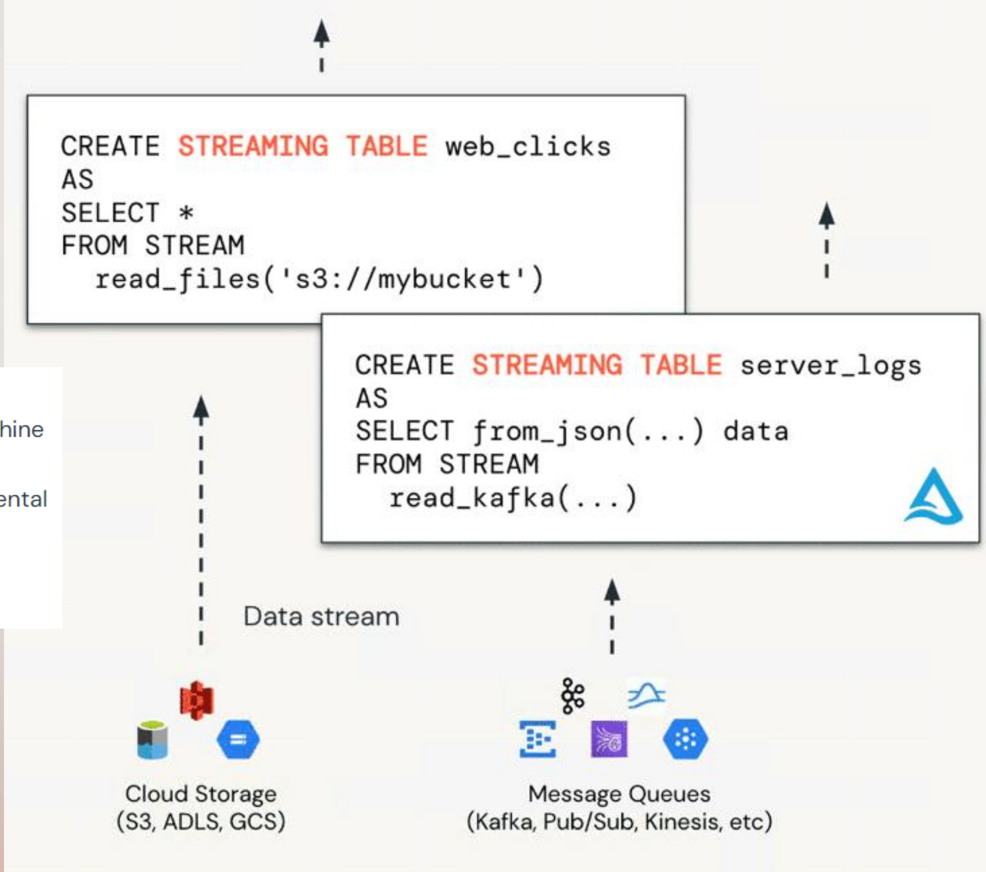


www.datapath.ai

Streaming Table

Benefits of streaming tables:

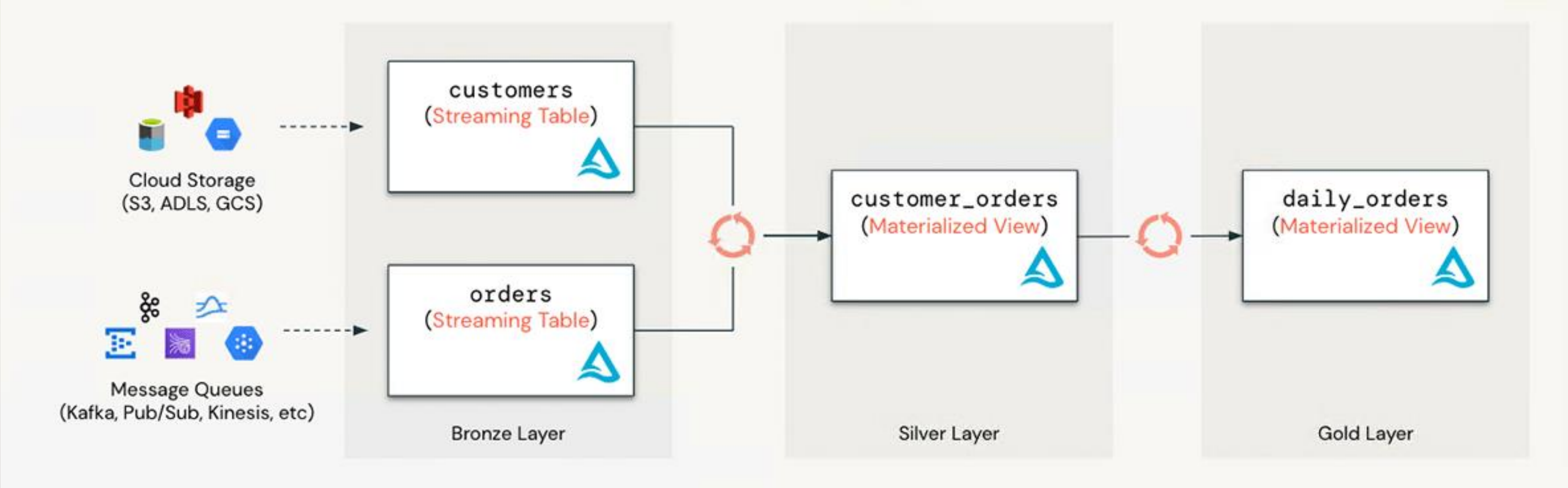
- Unlock real-time use cases. Ability to support real-time analytics/BI, machine learning, and operational use cases with streaming data.
- Better scalability. More efficiently handle high volumes of data via incremental processing vs large batches.
- Enable more practitioners. Simple SQL syntax makes data streaming accessible to all data engineers and analysts.



www.datapath.ai

Arquitecture

datapath



www.datapath.ai

Comparative

datapath

Declarative Programming with DLT

Declarative programs say what should be done, not how to do it

Spark imperative program

```
date = current_date()
spark.read.table("orders")
   .where(s"date = $date")
   .select("sum(sales)")
   .write
   .mode("overwrite")
   .replaceWhere(s"date = $date")
   .table("sales")
```

DLT Declarative Program

```
CREATE MATERIALZED VIEW sales
AS SELECT date, sum(sales)
FROM orders
GROUP BY date
```

The DLT runtime figures out the best way to create or update this table

Comparative

datapath

The core abstractions of DLT

You define datasets, and DLT automatically keeps them up to date

Streaming Tables

A delta table with stream(s) writing to it.

Used for:
- Ingestion
- Low latency transformations
- Huge scale

Materialized View

The result of a query, stored in a delta table.

Used for:
- Transforming data
- Building aggregate tables
- Speeding up BI queries and reports

Comparative

datapath

What is a Streaming Table?

A delta table that has structured streaming writing to it.

```
CREATE STREAMING TABLE report
AS SELECT *
FROM cloud_files("/mydata/", "json")
```

- Streaming tables read from append-only data sources such as Kafka, Kinesis, or Auto Loader (files on cloud storage)
- Streaming tables allow you to reduce costs and latency by reading each input record only once.
- Streaming tables support DML (UPDATE, DELETE, MERGE) for ad-hoc data manipulation (i.e. GDPR, etc)

Comparative

datapath

What is a Materialized View?

The result of a query, precomputed and stored in Delta

```
CREATE MATERIALIZED VIEW report
AS SELECT sum(profit)
FROM prod.sales
GROUP BY date
```

- A materialized view will always return the result of the the defining query, at the moment it was last updated (i.e. a snapshot)
- You cannot modify the data in a materialized view, you can change its query.

Comparative

datapath

Workflows Or DLT?

Often Both: Workflows can orchestrate anything, including DLT

Use Workflows to run any task

- At some schedule
- After other tasks have completed
- When a file arrives
- When another table is updated

Use DLT for managing dataflow

- Creating/updating delta tables
- Running Structured Streaming

Data Quality

datapath

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

Data Quality

datapath

Expectations using the power of SQL

Use SQL aggregates and joins to perform complex validations

```
-- Make sure a primary key is always unique.
CREATE MATERIALIZED VIEW report_pk_tests(
  CONSTRAINT unique_pk EXPECT (num_entries = 1)
)
AS SELECT pk, count(*) as num_entries
FROM LIVE.report
GROUP BY pk
```

Data Quality

datapath

Expectations using the power of SQL

Use SQL aggregates and joins to perform complex validations

```
-- Compare records between two tables,
-- or validate foreign key constraints.
CREATE MATERIALIZED report_compare_tests(
  CONSTRAINT no_missing EXPECT (r.key IS NOT NULL)
)
AS SELECT * FROM LIVE.validation_copy v
LEFT OUTER JOIN LIVE.report r ON v.key = r.key
```

Data Quality

datapath

Expectations using the power of SQL

Use SQL aggregates and joins to perform complex validations

```sql
-- Compare records between two tables,
-- or validate foreign key constraints.
CREATE MATERIALIZED report_compare_tests(
  CONSTRAINT no_missing EXPECT (r.key IS NOT NULL)
)
AS SELECT * FROM LIVE.validation_copy v
LEFT OUTER JOIN LIVE.report r ON v.key = r.key
```

Delta Live Tables

www.datapath.ai

Laboratorio Schemas y Tablas

www.datapath.ai

Laboratorio Version and Optimize Delta Tables

www.datapath.ai

datapath

Laboratorio Set Up Delta Tables

Laboratorio Load Data into Delta Lake

www.datapath.ai

¿PREGUNTAS?

Aprende, aplica y crece