Delta Live Tables
Hands-on Workshop

Glenn Wiebe, Sr. Solution Architect

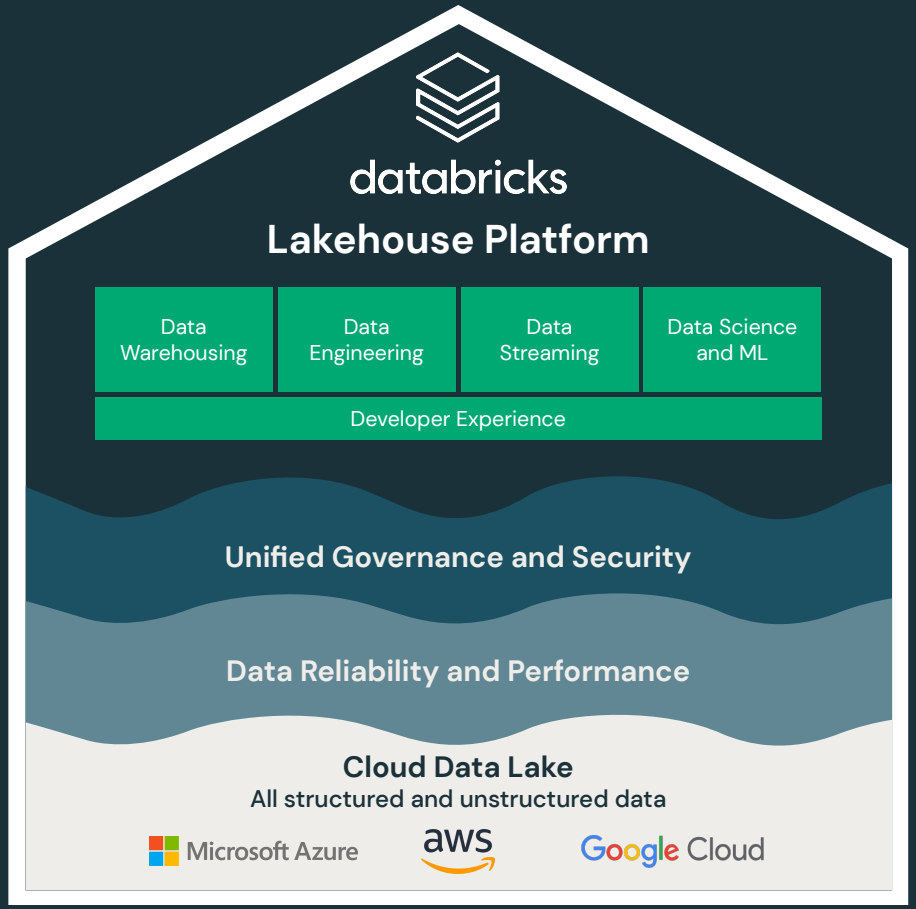
Alex Desroche, Sr. Customer Success Engineer

2022-06-16

databricks

Agenda

- Delta Live Tables within Databricks Lakehouse
- DLT Workshop
  - DB SQL & Data Profiling, Simplified Ingestion
  - Pipeline 1 - Sales “Channel” Reference Data Load
    - Simple → Medium → Enhanced DLT definition & iteration
    - Data lineage using built-in functions & capabilities
    - Transition from reference (as is) to master (governed) data loading
  - Pipeline 2 - “Customer” Master Data Load
    - Joins, External Data, Views & Advanced CDC
    - Troubleshooting
  - Event Log - Queries & Dashboards
  - Workflow - MTJ: SalesChannel → Customer Load
- Q&A



Databricks Lakehouse Platform

Simple

Unify your data warehousing and AI use cases on a single platform

Open

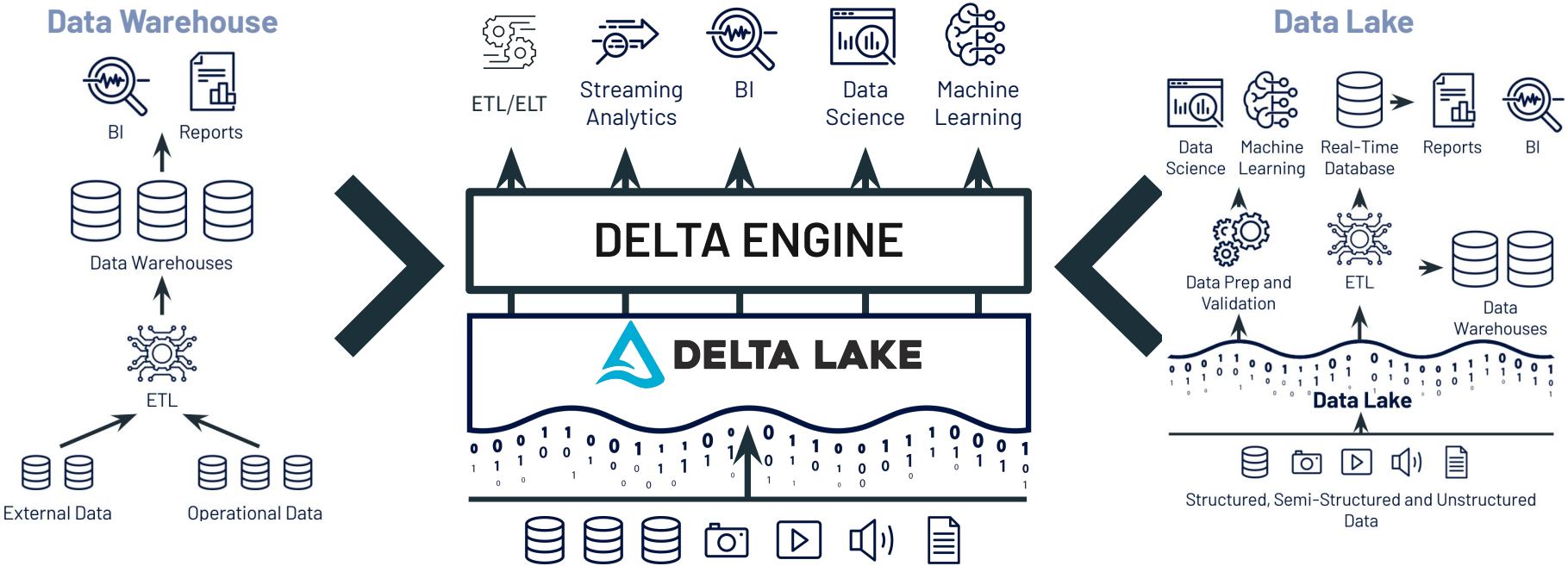
Built on open source and open standards

Multicloud

One consistent data platform across clouds

Delivered by the Lakehouse Architecture

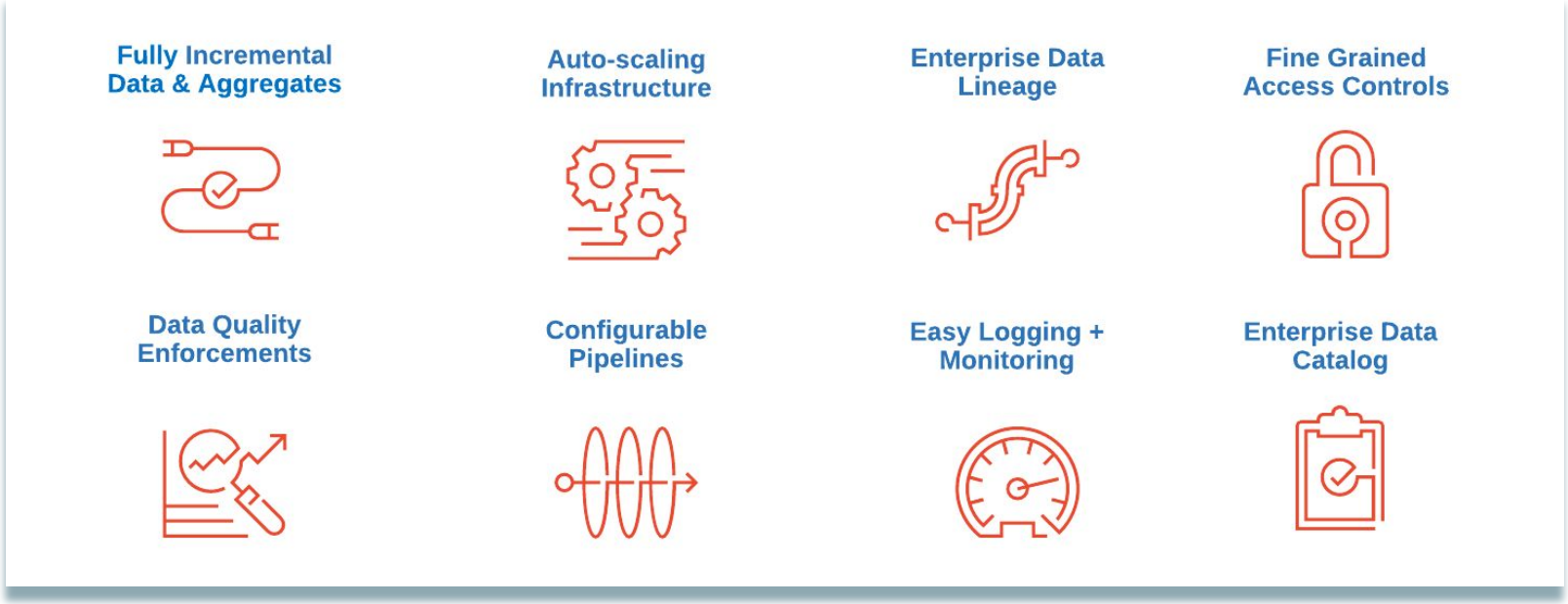
The best of Data Lakes and Data Warehouses



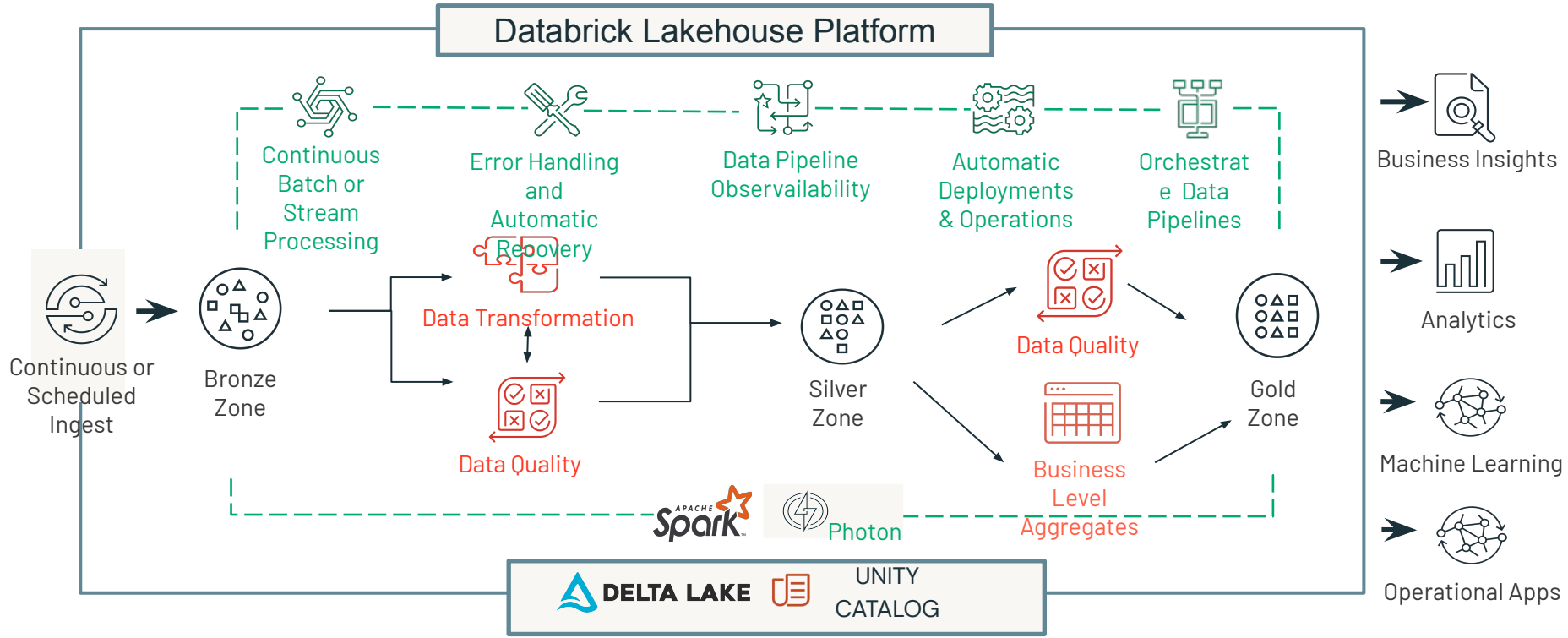
Delta Live Tables

Delta Live Tables

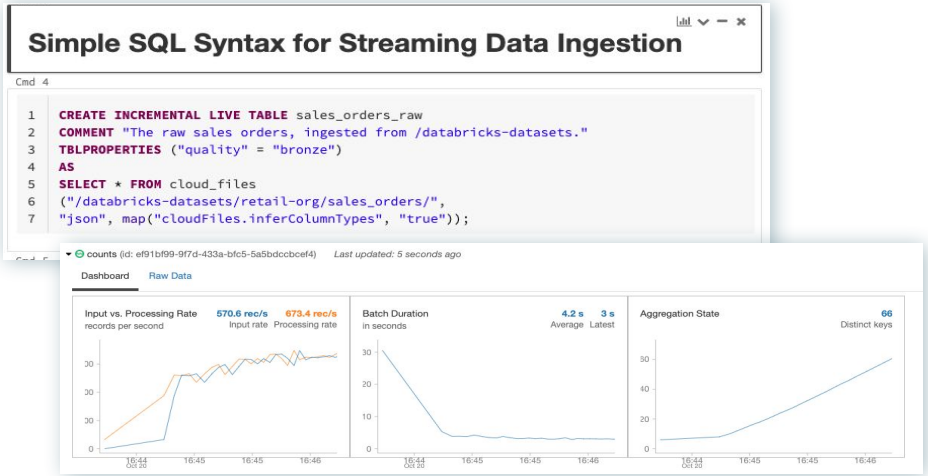
Push-to-start Enterprise Pipelines



Build Production ETL Pipelines with DLT



Continuous or scheduled data ingestion



- **Incrementally and efficiently** process new data files as they arrive in cloud storage using Auto Loader
- Automatically **infer schema** of incoming files or superimpose what you know with **Schema Hints**
- Automatic **schema evolution**
- **Rescue data column** – no data loss again

Schema Evolution

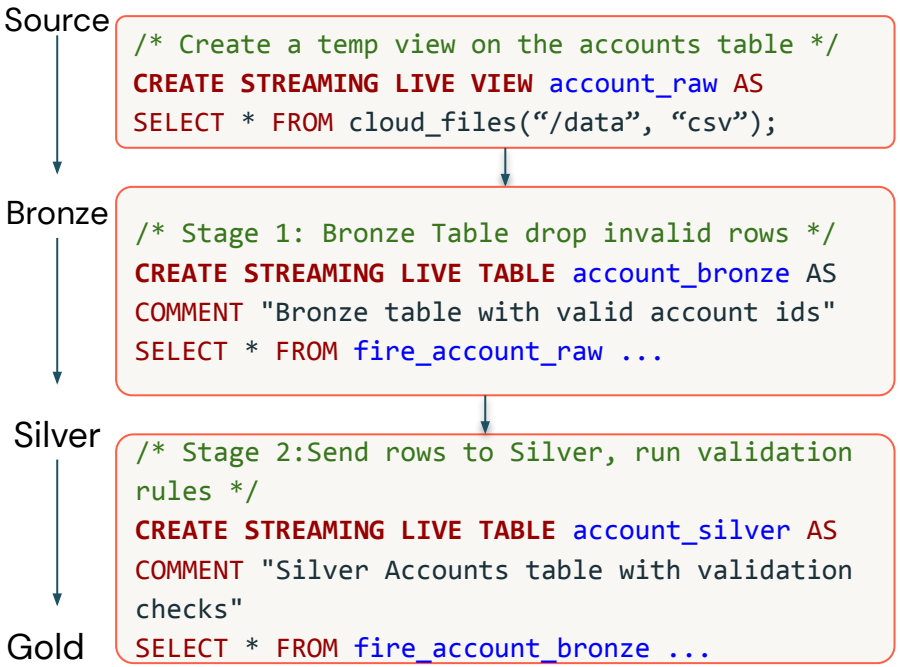
✅ JSON ✅ CSV ✅ AVRO

Coming Soon

PARQUET

databricks

Declarative SQL & Python APIs

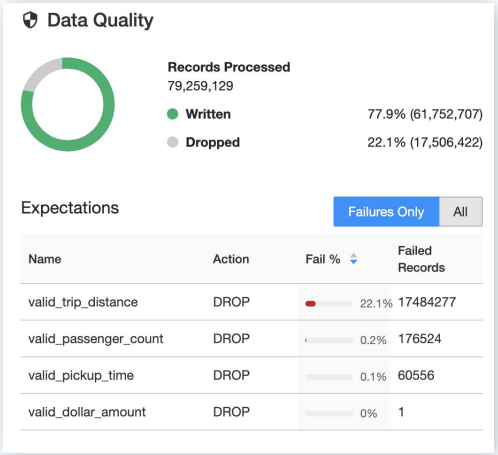


- Use intent-driven declarative development to abstract away the **"how"** and define **"what"** to solve
- Automatically generate **lineage** based on table dependencies across the data pipeline
- Automatically checks for errors, missing dependencies and syntax errors

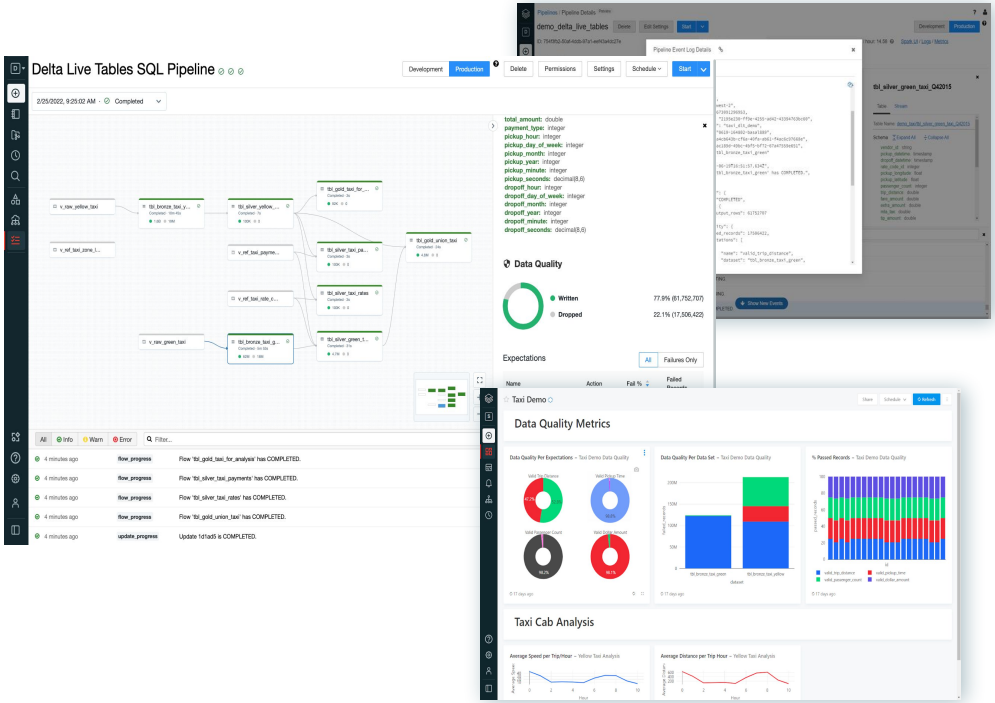
Data quality validation and monitoring

Load and transform at any scale with high quality data pipelines

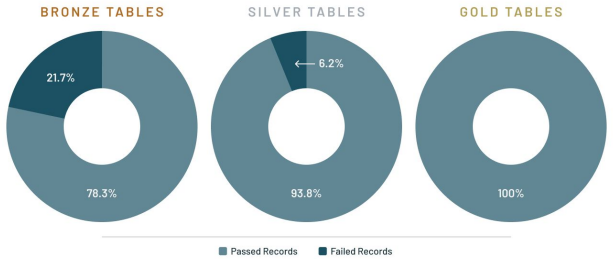
- **Define data quality & integrity controls** within the pipeline with data expectations
- **Address data quality errors** with flexible policies: fail, drop, alert, quarantine(future)
- All data pipeline runs & quality metrics are captured, tracked and reported



Data pipeline observability



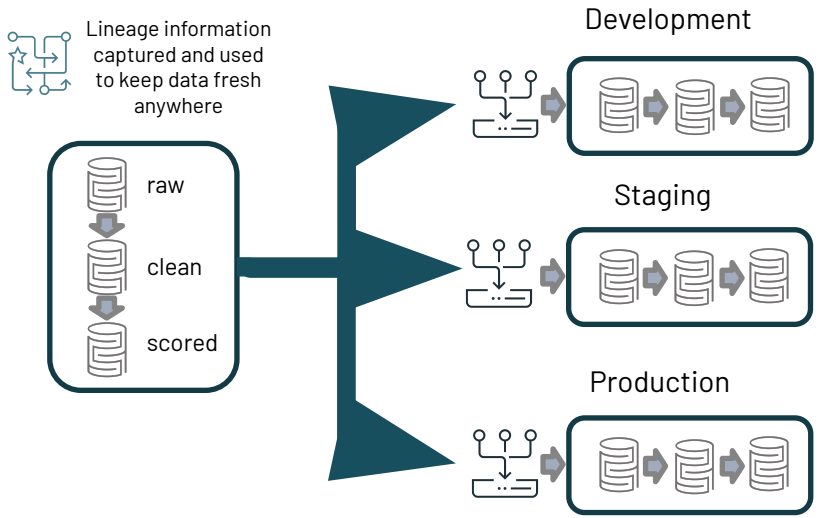
- High-quality, high-fidelity lineage diagram that provides visibility into how data flows for impact analysis
- Granular logging for operational, governance, quality and status of the data pipeline at a row level
- Continuously monitor data pipeline jobs to ensure continued operation
- Notifications using Databricks SQL



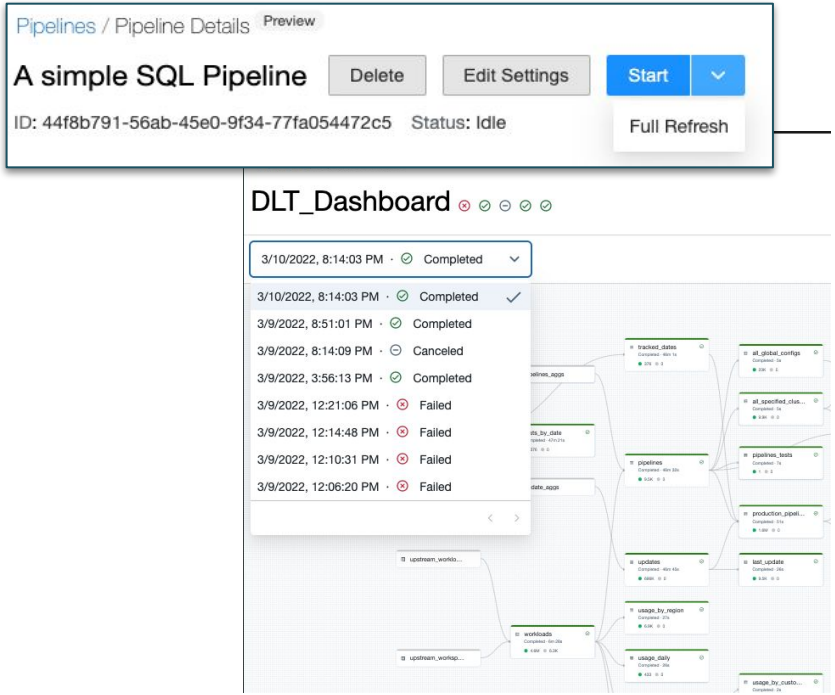
databricks

Automated ETL development lifecycle

- Develop in environment(s) separate from production with the ability to easily test it before deploying - entirely in SQL
- Deploy and manage environments using parameterization
- Unit testing and documentation
- Enables metadata-driven ability to programatically scale to 100s of tables/pipelines dynamically

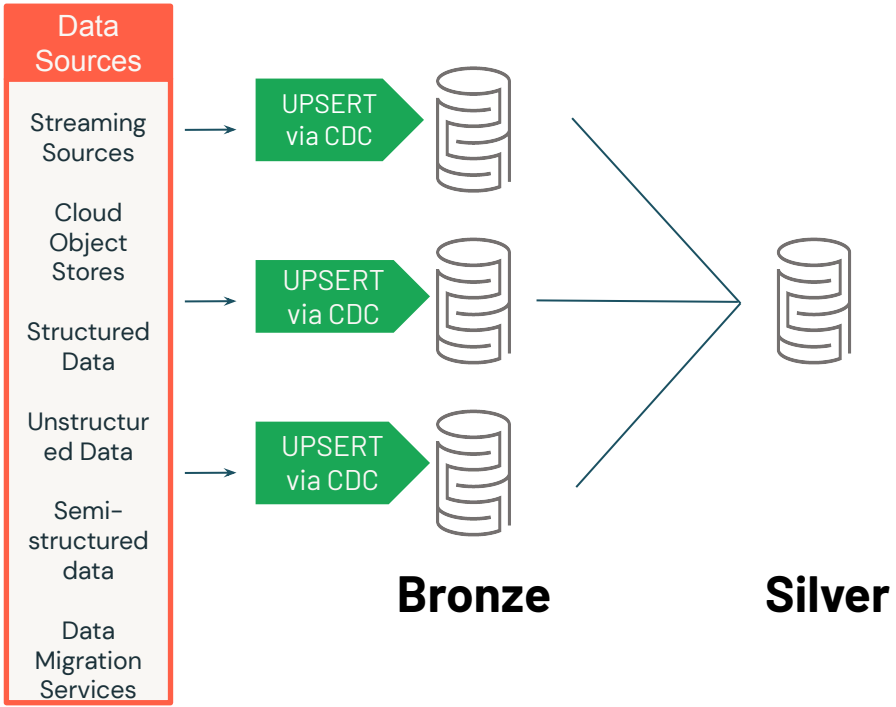


Automated ETL operations



- Reduce down time with automatic error handling and easy replay
- Eliminate maintenance with automatic optimizations of all Delta Live Tables
- **Auto-scaling** adds more resources automatically when needed.

Change data capture (CDC)



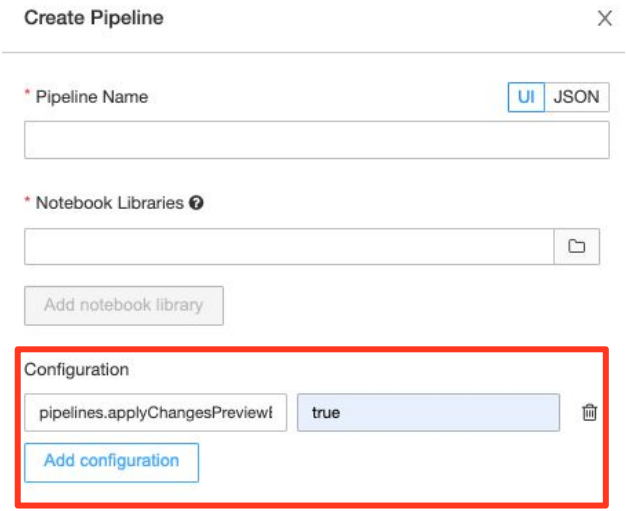
- Stream change records (inserts, updates, deletes) from any data source supported by DBR, cloud storage, or DBFS
- Simple, declarative "APPLY CHANGES INTO" API for SQL or Python
- Handles out-of-order events
- Schema evolution

Getting started with Change Data Capture in DLT

Enable **DLT APPLY CHANGES INTO in DLT Pipeline**

```
"configuration": {

"pipelines.applyChangesPreviewEnabled":
"true"
    }
```



Getting started with Change Data Capture in DLT

Delete unwanted clients records - Silver Table - DLT SQL

```
1  CREATE INCREMENTAL LIVE TABLE retail_client_cdc_silver;
2
3  APPLY CHANGES INTO live.retail_client_cdc_silver
4  FROM stream(live.retail_client_cdc_raw)
5    KEYS (Id)
6    APPLY AS DELETE WHEN operation = "DELETE"
7    SEQUENCE BY operation_date --primary key, auto-incrementing ID of any kind that can be used to identity order of events, or timestamp
8    COLUMNS * EXCEPT (operation, operation_date);
```

APPLY CHANGES INTO to propagate inserts, updates and deletes to your target table.

Specifies when a CDC event should be treated as a DELETE rather than an upsert.

To remove columns needed for APPLY CHANGES INTO computation, but not needed in final table state.

"Delta Live Tables has helped our teams save time and effort in managing data at this scale. With this capability augmenting the existing lakehouse architecture, Databricks is **disrupting the ETL and data warehouse markets**, which is important for companies like ours. We are excited to continue to work with Databricks as an innovation partner." - Dan Jeavons, GM Data Science

Use Case + Challenge

- 70+ use cases impacting supply chain, operations, product development, marketing, customer exp
- Large volumes of IoT data from millions of sensors difficult to harness for actionable insights and ML due to operational load created by complex data pipelines

Why Databricks + DLT?

- Lakehouse for unified data warehousing, BI, & ML — enabling new use cases not possible before
- DLT enables Shell to build reliable and scalable data pipelines - automatic job maintenance and deep pipeline visibility saves time and resources

Impact of DLT

- Process **1.3 trillion rows of sensor data** with ease
- Simplifying ETL development and management for **faster insights and ML innovation**

Agenda

- Delta Live Tables within Databricks Lakehouse
- DLT Workshop
  - DB SQL & Data Profiling, Simplified Ingestion
  - Pipeline 1 - Sales “Channel” Reference Data Load
    - Simple → Medium → Enhanced DLT definition & iteration
    - Data lineage using built-in functions & capabilities
    - Transition from reference (as is) to master (governed) data loading
  - Pipeline 2 - “Customer” Master Data Load
    - Joins, External Data, Views & Advanced CDC
    - Troubleshooting
  - Event Log - Queries & Dashboards
  - Workflow - MTJ: SalesChannel → Customer Load
- Q&A

Reference Materials/Links

Delta Live Tables is a framework for building reliable, maintainable, and testable data processing pipelines. You define the transformations to perform on your data, and Delta Live Tables manages task orchestration, cluster management, monitoring, data quality, and error handling.

Instead of defining your data pipelines using a series of separate Apache Spark tasks, Delta Live Tables manages how your data is transformed based on a target schema you define for each processing step. You can also enforce data quality with Delta Live Tables *expectations*. Expectations allow you to define expected data quality and specify how to handle records that fail those expectations.

To get started with Delta Live Tables:

- Develop your first Delta Live Tables pipeline with the quickstart.
- Learn about fundamental Delta Live Tables concepts.
- Learn how to create, run, and manage pipelines with the Delta Live Tables user interface.
- Learn how to develop Delta Live Tables pipelines with Python or SQL.
- Learn how to manage data quality in your Delta Live Tables pipelines with expectations.

Learn more about Delta Live Tables:

- Use external data sources in your Delta Live Tables pipelines: Data sources
- Use the data produced by your Delta Live Tables pipelines: Publish data
- Efficiently process continually arriving data in your Delta Live Tables pipelines: Streaming data processing
- Use change data capture (CDC) processing in your Delta Live Tables pipelines: Change data capture with Delta Live Tables
- Use the Delta Live Tables API: API guide
- Configure your Delta Live Tables pipelines: Pipeline settings
- Analyze and report on your Delta Live Tables pipelines: Querying the event log
- Run your Delta Live Tables pipelines with popular workflow orchestration tools: Workflow tool integration
- Learn how to use access control lists (ACLs) to configure permissions on your Delta Live Tables pipelines: Access control

Find answers and solutions for Delta Live Tables:

- Implement common tasks in your Delta Live Tables pipelines: Cookbook
- Review frequently asked questions and issues: FAQ

Delta Live Tables is a framework for building reliable, maintainable, and testable data processing pipelines. You define the transformations to perform on your data, and Delta Live Tables manages task orchestration, cluster management, monitoring, data quality, and error handling.

Instead of defining your data pipelines using a series of separate Apache Spark tasks, Delta Live Tables manages how your data is transformed based on a target schema you define for each processing step. You can also enforce data quality with Delta Live Tables *expectations*. Expectations allow you to define expected data quality and specify how to handle records that fail those expectations.

To get started with Delta Live Tables:

- Develop your first Delta Live Tables pipeline with the quickstart.
- Learn about fundamental Delta Live Tables concepts.
- Learn how to create, run, and manage pipelines with the Delta Live Tables user interface.
- Learn how to develop Delta Live Tables pipelines with Python or SQL.
- Learn how to manage data quality in your Delta Live Tables pipelines with expectations.

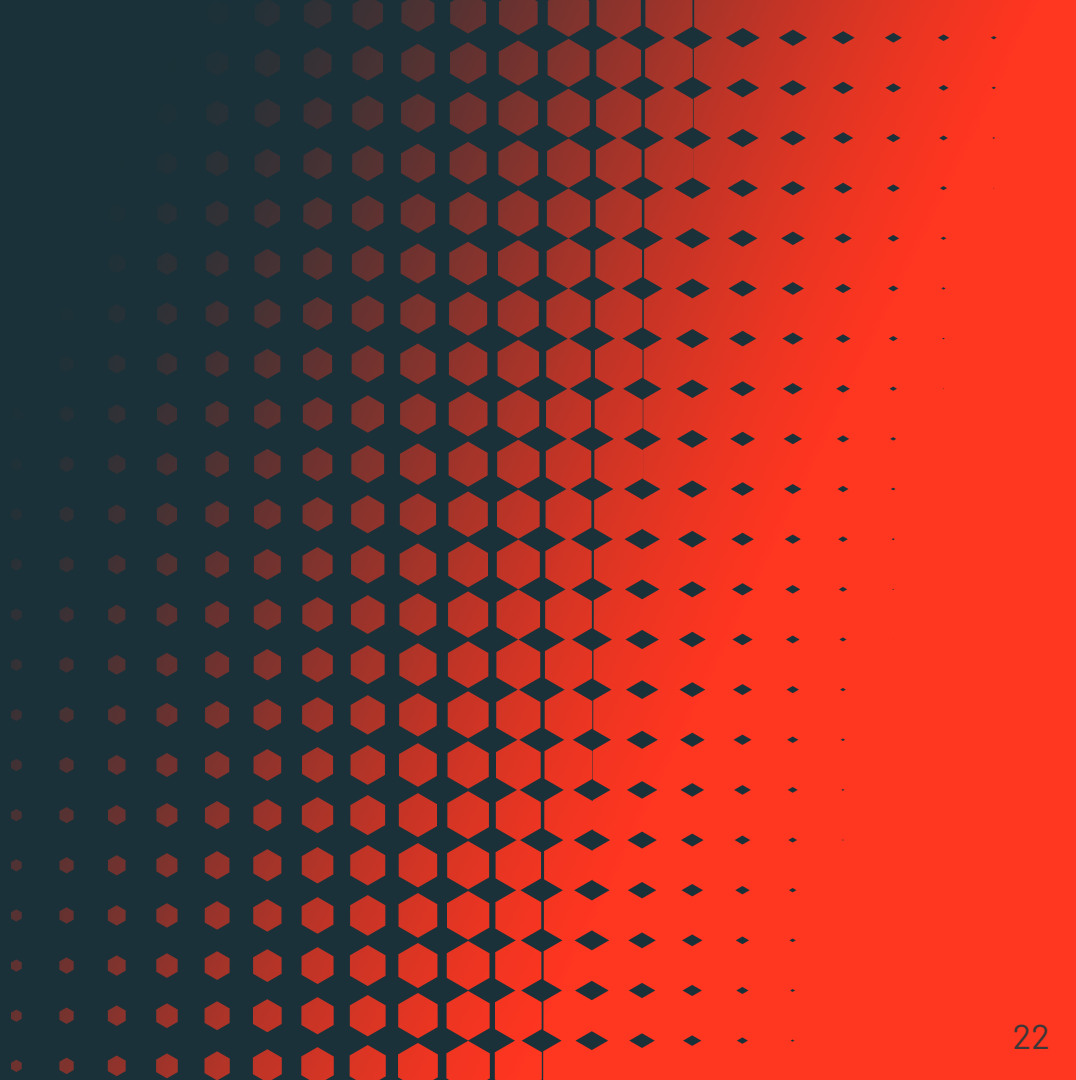
Learn more about Delta Live Tables:

- Use external data sources in your Delta Live Tables pipelines: Data sources
- Use the data produced by your Delta Live Tables pipelines: Publish data
- Efficiently process continually arriving data in your Delta Live Tables pipelines: Streaming data processing
- Use change data capture (CDC) processing in your Delta Live Tables pipelines: Change data capture with Delta Live Tables
- Use the Delta Live Tables API: API guide
- Configure your Delta Live Tables pipelines: Pipeline settings
- Analyze and report on your Delta Live Tables pipelines: Querying the event log
- Run your Delta Live Tables pipelines with popular workflow orchestration tools: Workflow tool integration
- Learn how to use access control lists (ACLs) to configure permissions on your Delta Live Tables pipelines: Access control

Find answers and solutions for Delta Live Tables:

- Implement common tasks in your Delta Live Tables pipelines: Cookbook
- Review frequently asked questions and issues: FAQ

Questions?



Q&A and Post-workshop Discussion

Thank you for joining us. We're here to answer any questions you have about the content, Databricks, or anything else. We'll send out a copy of this presentation after we wrap up.

Workshop Git Repo: https://github.com/ggwiebe/db_dlt_workshop.git

Let us know what you liked and didn't like so that we can continue to improve.

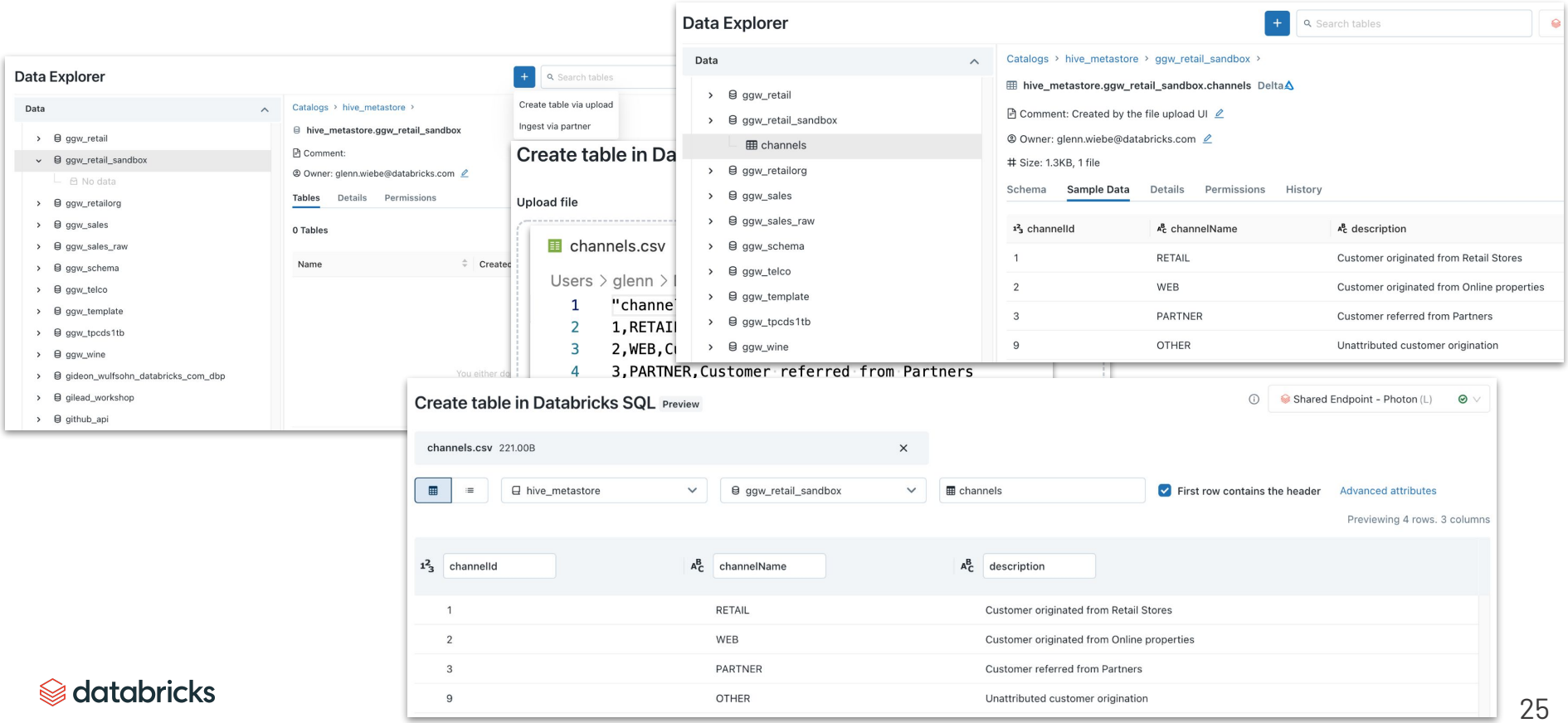
If you're not currently in contact with your Databricks account team and would like to be, let the presenters know and we can assist.

Contact info for presenters

glenn.wiebe@databricks.com , alex.desroches@databricks.com

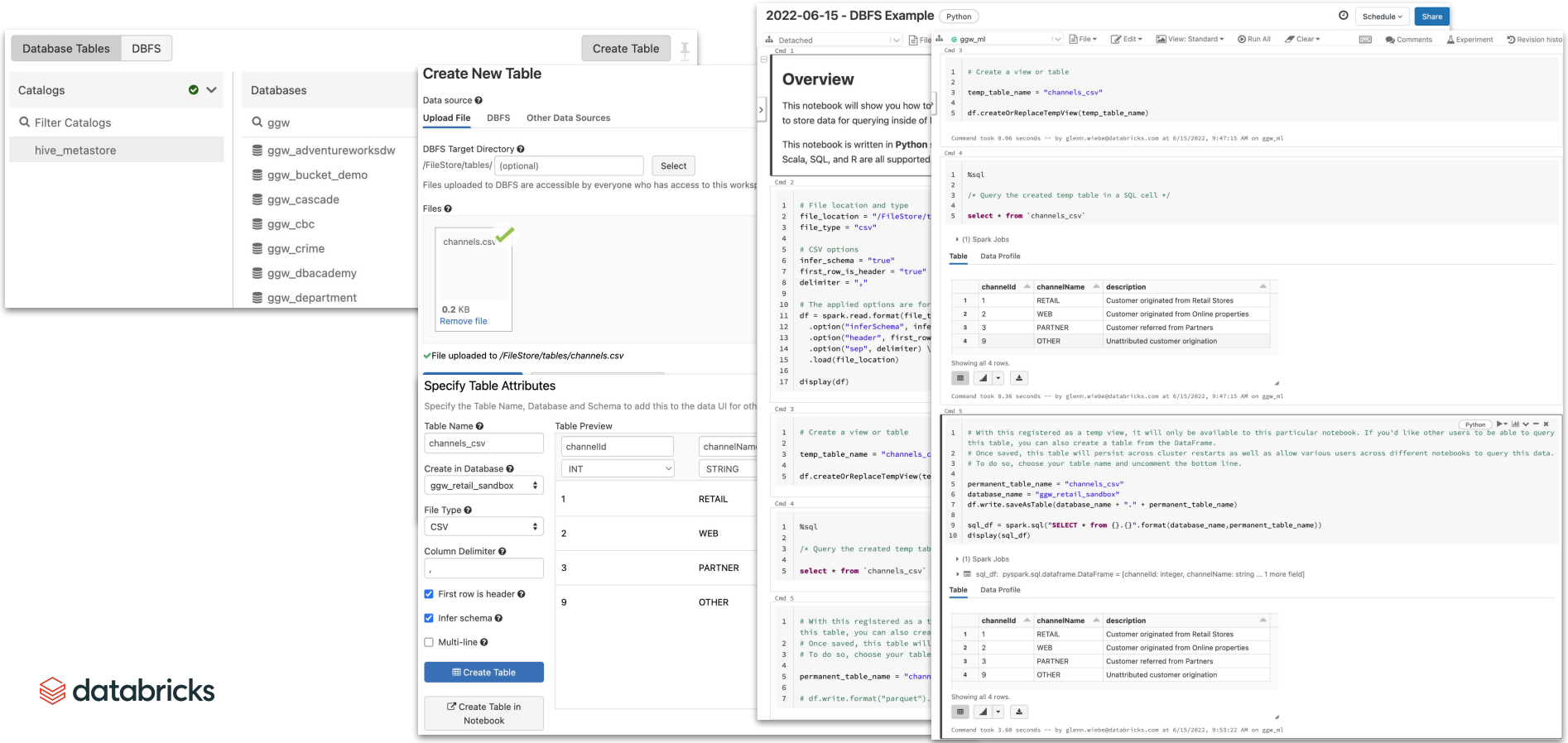
Workshop Screenshots

DLT Lab Screenshots – Ingest Data – Explorer Wizard



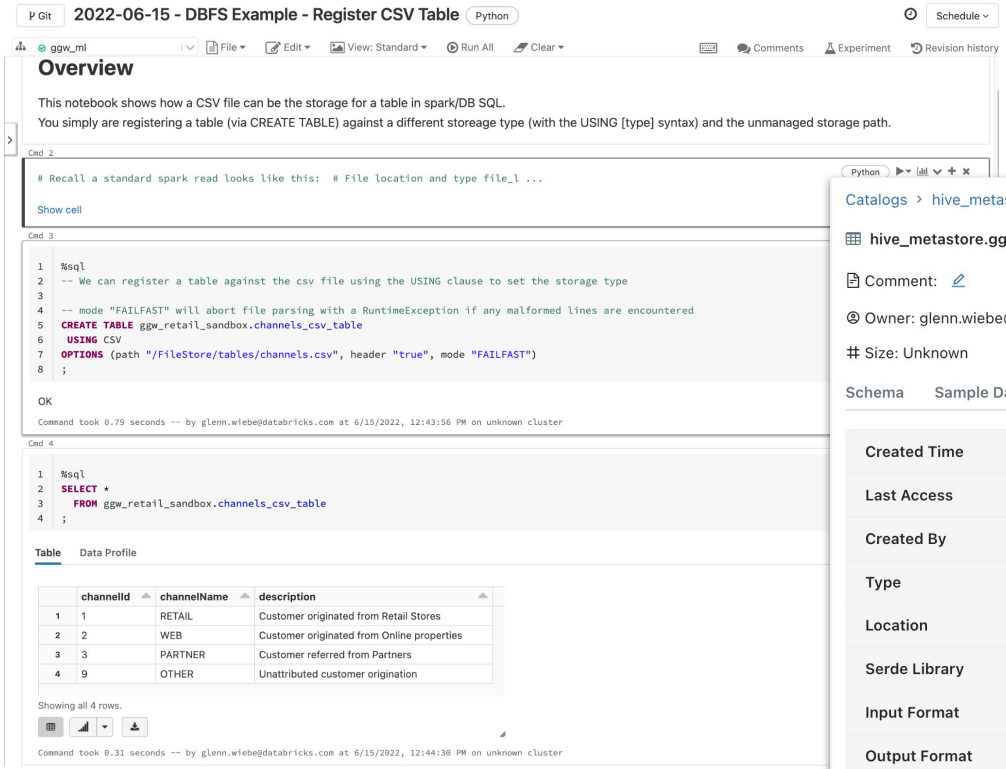
databricks

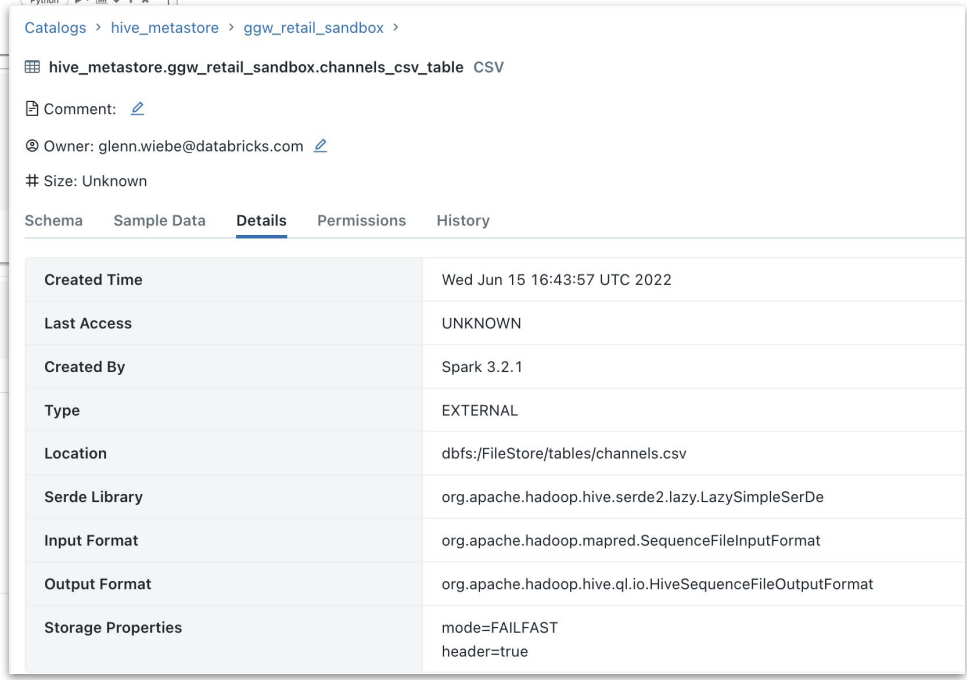
DLT Lab Screenshots - Ingest Data - Notebook Wizard



databricks

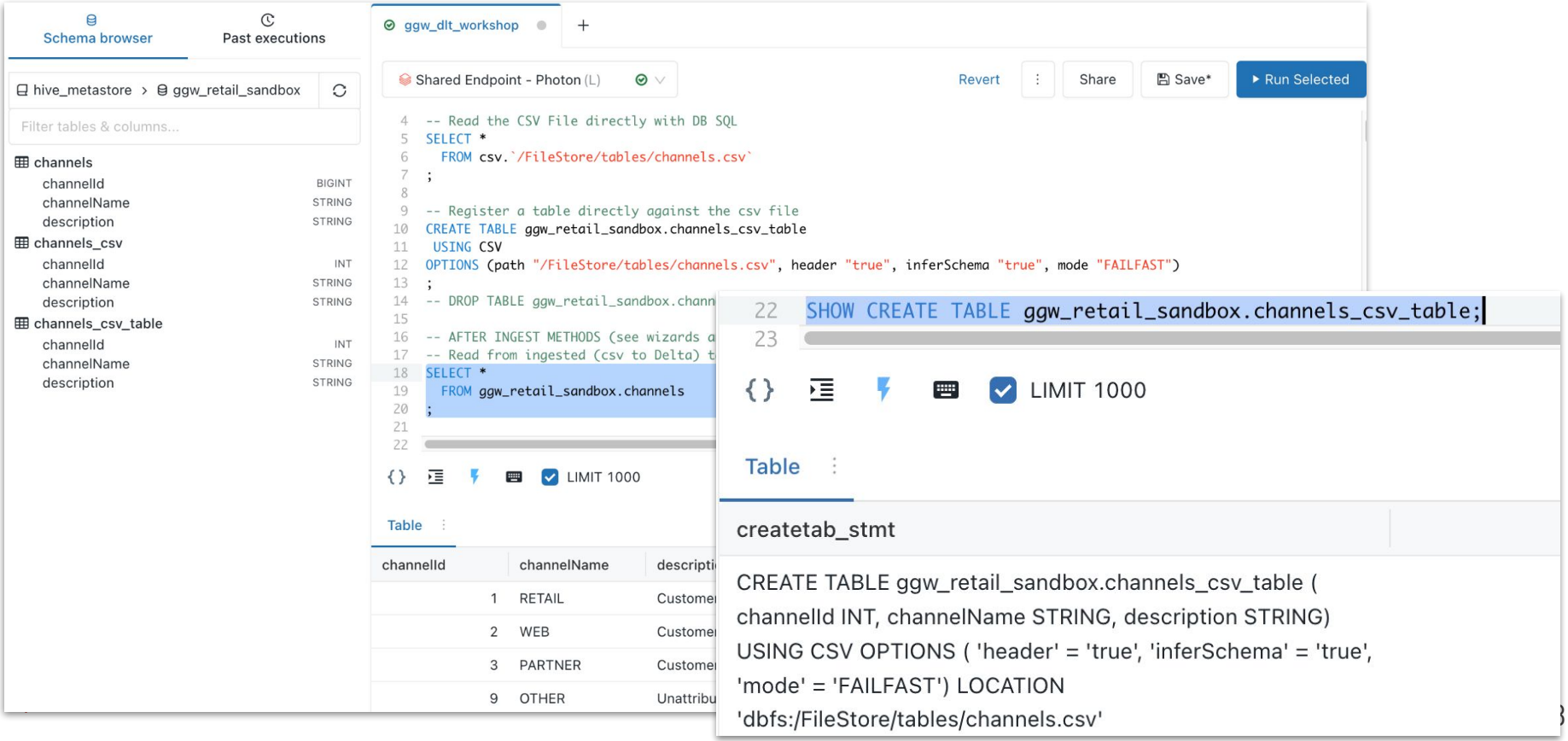
DLT Lab Screenshots - Ingest Data - Notebook Wizard #2





databricks

DLT Lab Screenshots - Build DLT Pipeline - Query Source



DLT Lab Screenshots - Create DLT Pipeline - Reference

```
"channelId","channelName","description"
1,RETAIL,Customer originated from Retail
2,WEB,Customer originated from Online pr
3,PARTNER,Customer referred from Partners
9,OTHER,Unattributed customer originatio
```

Step 0 - Get Table info from selec

-- This is the syntax from the CSV Table

-- We will not run this, but it serves to direct our table name

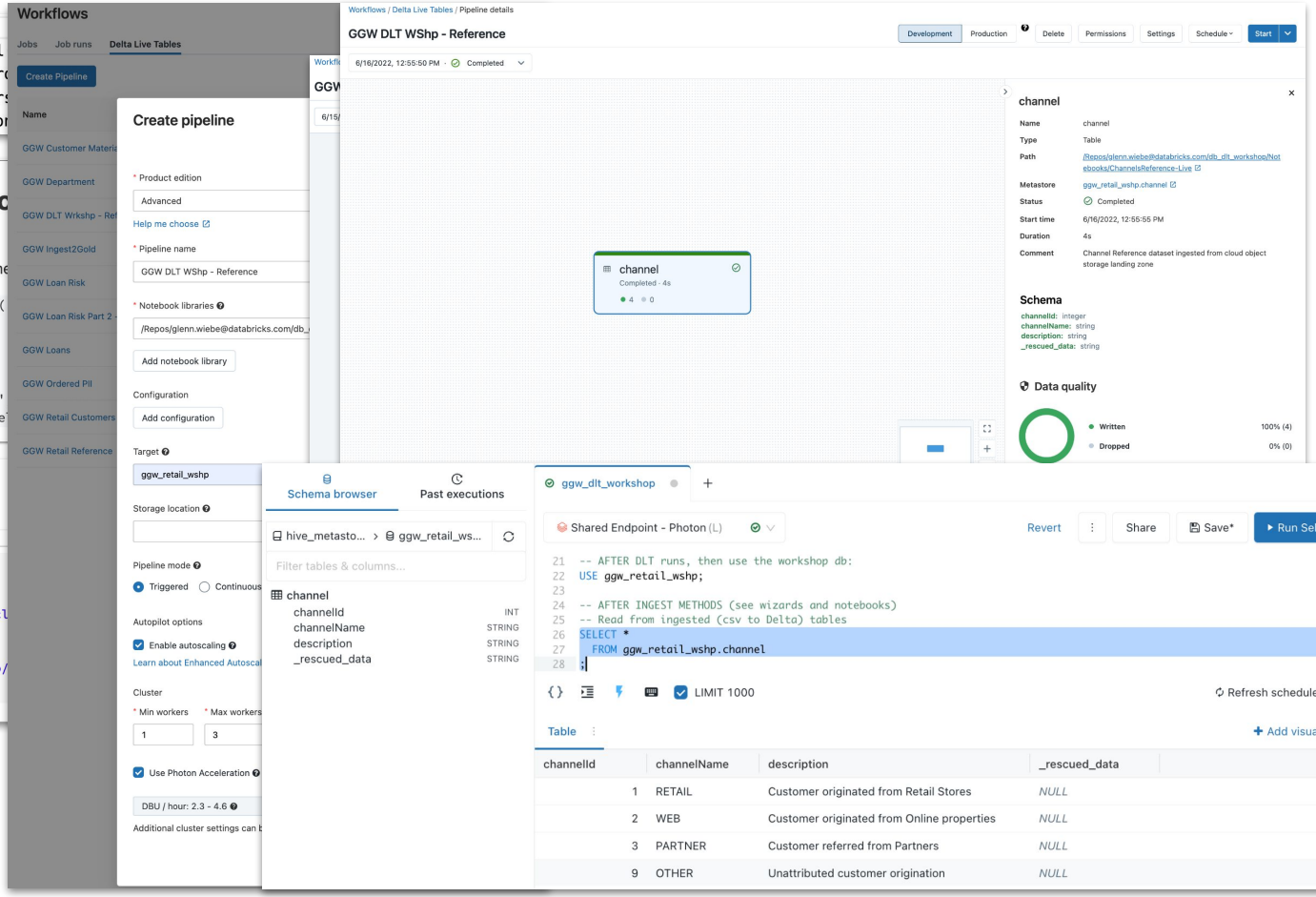
```
CREATE TABLE ggw_retail_sandbox.channels_csv_table (
  channelId INT,
  channelName STRING,
  description STRING)
   USING CSV
 OPTIONS ( 'header' = 'true', 'inferSchema' = 'true'
LOCATION 'dbfs:/FileStore/tables/ggw_dlt_wshp/channe
```

Cmd 4

Step 1-a - simplest query

Cmd 5

```
CREATE STREAMING LIVE TABLE channel
TBLPROPERTIES ("quality" = "reference")
COMMENT "Channel Reference dataset ingested from cl
AS
SELECT *
  FROM cloud_files('/FileStore/tables/ggw_dlt_wshp/
;
```



databricks

DLT Lab Screenshots - DLT Master with SCD Tracking

```
SELECT *
  FROM ggw_retail_wshp.channel_master
;
```

LIMIT 1000

Table

channelId	channelName	description	input_file_name	dlt_ingest_dt	dlt_ing
1	RETAIL	Customer originated from Retail Stores	/FileStore/tables/ggw_dlt_wshp/channels.csv	2022-06-16 16:29:57.125	RetailR
2	WEB	Customer originated from Online properties	/File		
3	PARTNER	Customer referred from Partners	/File		
9	OTHER	Unattributed customer origination	/File		

Step 4 - Add SCD Type Tracking to a Silver Master Cha

Cmd 11

```
-- SILVER - View against Bronze that will be used to load silver incrementally with AP
-- CREATE TEMPORARY [STREAMING] LIVE VIEW view_name
CREATE TEMPORARY STREAMING LIVE VIEW channel_silver_v (
  CONSTRAINT valid_file      EXPECT (input_file_name IS NOT NULL) ON VIOLATION DROP
  CONSTRAINT valid_procedure EXPECT (dlt_ingest_procedure IS NOT NULL) ON VIOLATION
)
COMMENT "View of cleansed Channel (bronze tier) for loading into / mastering in Silver
AS SELECT channelId,
          channelName,
          description,
          input_file_name,
          dlt_ingest_dt,
          dlt_ingest_procedure,
          dlt_ingest_principal
  FROM STREAM(LIVE.channel) c
;
```

Cmd 12

```
CREATE STREAMING LIVE TABLE channel_master;

APPLY CHANGES INTO LIVE.channel_master
FROM STREAM(LIVE.channel_silver_v)
KEYS (channelId)
SEQUENCE BY dlt_ingest_dt
STORED AS SCD TYPE 2
;
```

Workflows / Delta Live Tables / Pipeline details

GGW DLT WShp - Reference

Development | Production | Delete | Permissions | Settings | Schedule | Start

6/16/2022, 1:24:49 PM · Completed

channel → channel_silver_v → channel_master

Metastore: ggw_retail_wshp.channel
Status: Completed
Start time: 6/16/2022, 1:24:56 PM
Duration: 5s
Comment: Channel Reference dataset ingested from cloud object storage landing zone

Schema
channelId: integer
channelName: string
description: string
input_file_name: string
dlt_ingest_dt: timestamp
dlt_ingest_procedure: string
dlt_ingest_principal: string

Data quality
Written 100% (4)
Dropped 0% (0)

Expectations

Name	Action	Fail %	Failed records
valid_channel_name	DROP	0%	0

52 seconds ago	flow_progress	Flow 'channel' has COMPLETED.
51 seconds ago	flow_progress	Flow 'channel_master' is STARTING.
50 seconds ago	flow_progress	Flow 'channel_master' is RUNNING.
47 seconds ago	flow_progress	Flow 'channel_master' has COMPLETED.
45 seconds ago	update_progress	Update 56a310 is COMPLETED.

```
"channelId","channelName","description"
3,PARTNER,Customer originated OR referred from Partners
4,UNAFFILIATED,Customers that request anonymity
9,OTHER,Unattributed or unknown customer origination
```

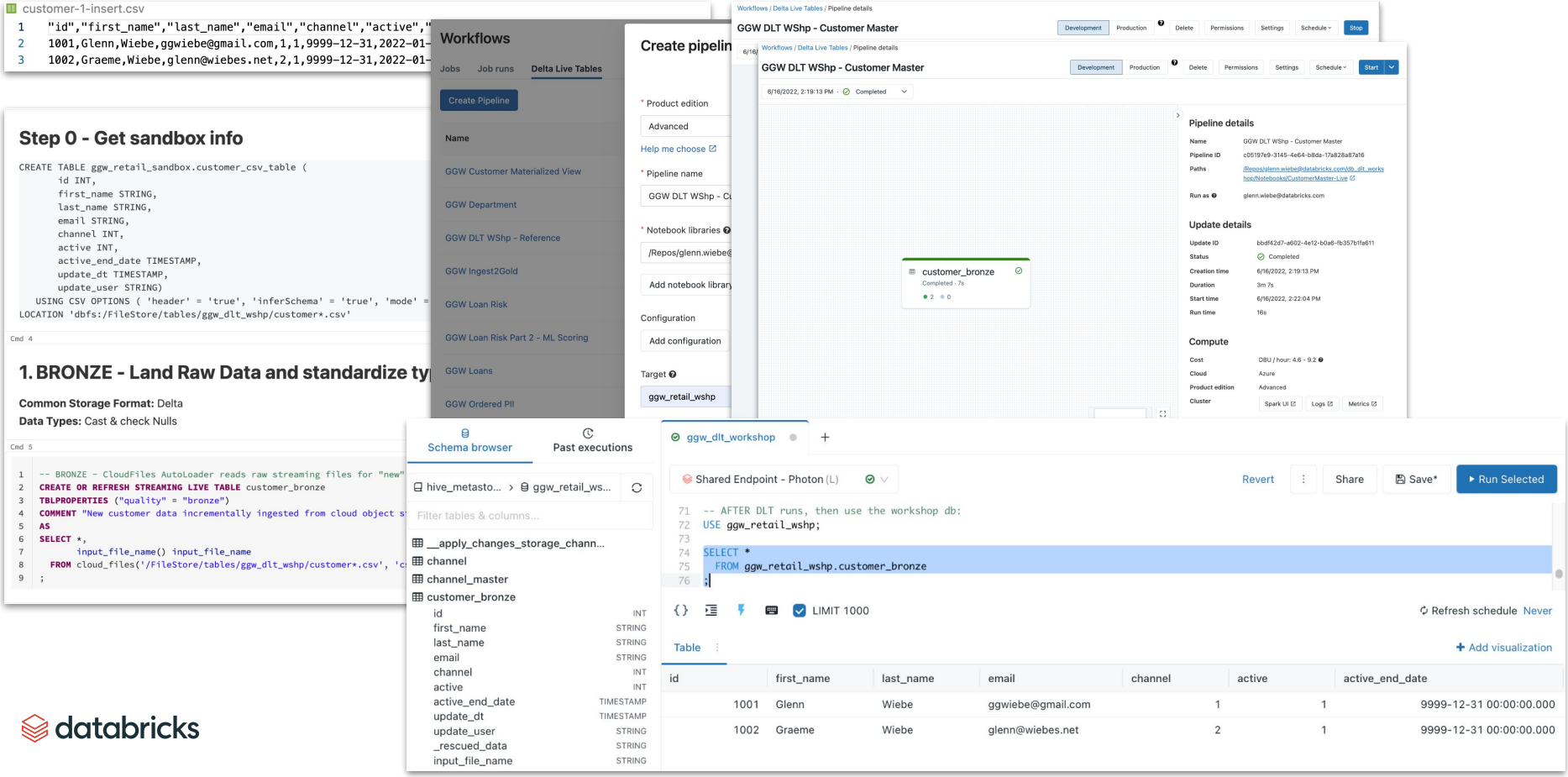
```
SELECT *
  FROM ggw_retail_wshp.channel_master
;
```

LIMIT 1000

Table

channelId	channelName	description	input_file_name	dlt_ingest_dt	dlt_ingest_procedure	dlt_ingest_principal	START_AT	END_AT
1	RETAIL	Customer originated from Retail Stores	/FileStore/tables/ggw_dlt_wshp/channels.csv	2022-06-16 16:29:57.125	RetailReference_Live	glenn.wiebe@databricks.com	2022-06-16 16:29:57.125	NULL
2	WEB	Customer originated from Online properties	/FileStore/tables/ggw_dlt_wshp/channels.csv	2022-06-16 16:29:57.125	RetailReference_Live	glenn.wiebe@databricks.com	2022-06-16 16:29:57.125	NULL
3	PARTNER	Customer referred from Partners	/FileStore/tables/ggw_dlt_wshp/channels.csv	2022-06-16 16:29:57.125	RetailReference_Live	glenn.wiebe@databricks.com	2022-06-16 16:29:57.125	2022-06-16 16:36:17.685
3	PARTNER	Customer originated OR referred from Partners	/FileStore/tables/ggw_dlt_wshp/channels_update1.csv	2022-06-16 16:36:17.685	RetailReference_Live	glenn.wiebe@databricks.com	2022-06-16 16:36:17.685	NULL
4	UNAFFILIATED	Customers that request anonymity	/FileStore/tables/ggw_dlt_wshp/channels_update1.csv	2022-06-16 16:36:17.685	RetailReference_Live	glenn.wiebe@databricks.com	2022-06-16 16:36:17.685	NULL
9	OTHER	Unattributed customer origination	/FileStore/tables/ggw_dlt_wshp/channels.csv	2022-06-16 16:29:57.125	RetailReference_Live	glenn.wiebe@databricks.com	2022-06-16 16:29:57.125	2022-06-16 16:36:17.685
9	OTHER	Unattributed or unknown customer origination	/FileStore/tables/ggw_dlt_wshp/channels_update1.csv	2022-06-16 16:36:17.685	RetailReference_Live	glenn.wiebe@databricks.com	2022-06-16 16:36:17.685	NULL

DLT Lab Screenshots - DLT Pipeline - Customer Master



databricks

DLT Lab Screenshots - DLT Pipeline - Customer Master - 2

customer-1-insert.csv

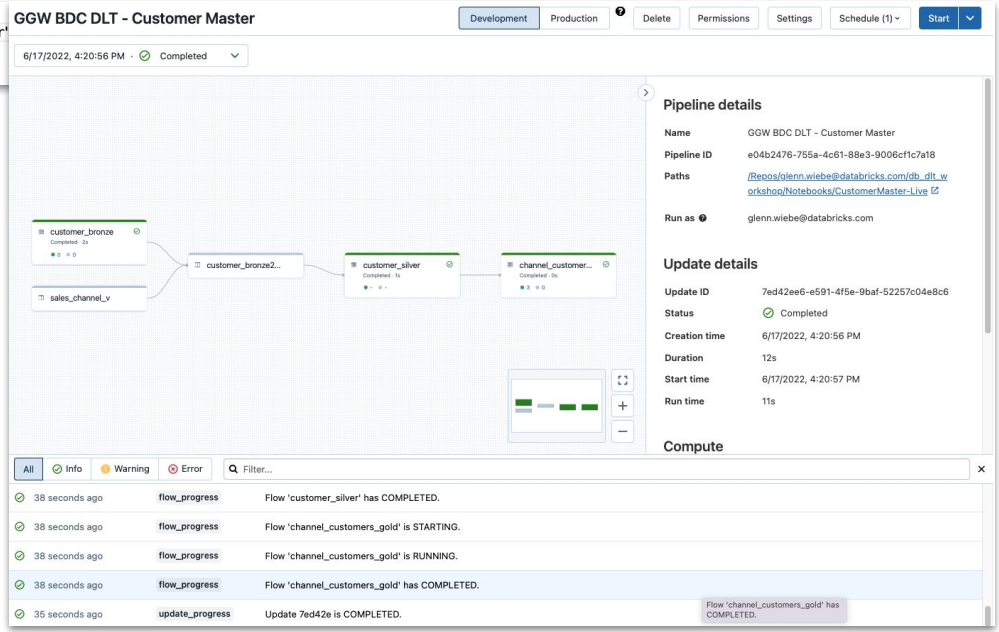
```
1  "id","first_name","last_name","email","channel","active","active_end_date","update_dt","update_user"
2  1001,Glenn,Wiebe,ggwiebe@gmail.com,1,1,9999-12-31,2022-01-13 16:12:42.453,glenn@wiebes.net
3  1002,Graeme,Wiebe,glenn@wiebes.net,2,1,9999-12-31,2022-01-13 16:12:42.453,glenn@wiebes.net
```

```
-- SILVER - View against Bronze that will be used to load silver incrementally with APPLY CHANGES INTO
CREATE TEMPORARY STREAMING LIVE VIEW customer_bronze2silver_v (
  CONSTRAINT valid_id           EXPECT (id IS NOT NULL) ON VIOLATION DROP ROW,
  CONSTRAINT valid_active       EXPECT (active BETWEEN 0 AND 1) ON VIOLATION DROP ROW,
  CONSTRAINT valid_channel      EXPECT (sales_channel IS NOT NULL),
  CONSTRAINT valid_first_name   EXPECT (first_name IS NOT NULL),
  CONSTRAINT valid_last_name    EXPECT (last_name IS NOT NULL)
)
COMMENT "View of cleansed Bronze Customer for loading into Silver."
AS SELECT c.id,
          UPPER(c.first_name) as first_name,
          UPPER(c.last_name) as last_name,
          c.email,
          sc.channelName sales_channel,
          c.active,
          c.active_end_date,
          c.update_dt,
          c.update_user,
          current_timestamp() dlt_ingest_dt,
          "CustomerMaster-Live" dlt_ingest_procedure,
          current_user() dlt_ingest_principal
     FROM STREAM(live.customer_bronze) c
     LEFT JOIN live.sales_channel_v sc
       ON c.channel = sc.channelId
;
```

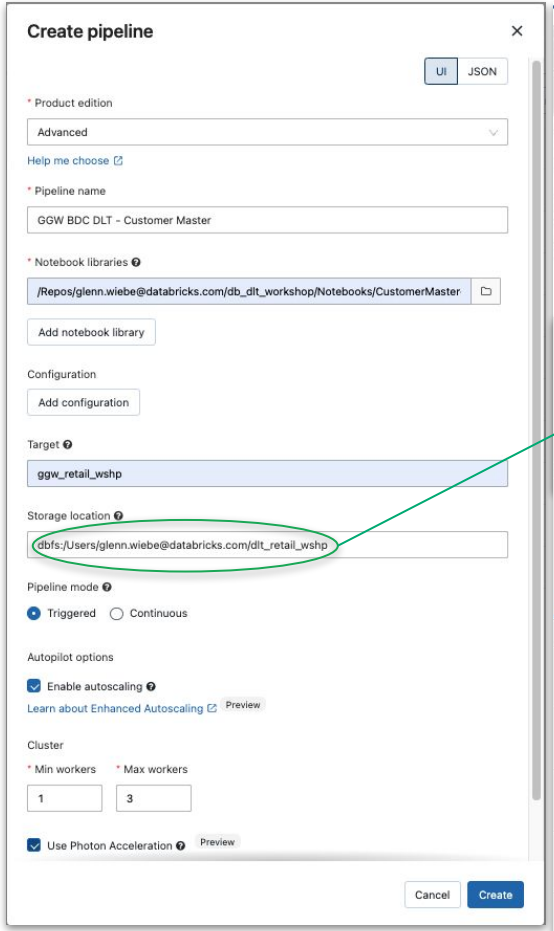
```
-- SILVER - Incremental Customer table with APPLY CHANGES INTO change handling
CREATE INCREMENTAL LIVE TABLE customer_silver
TBLPROPERTIES ("quality" = "silver")
COMMENT "Clean, merged customers"
;
```

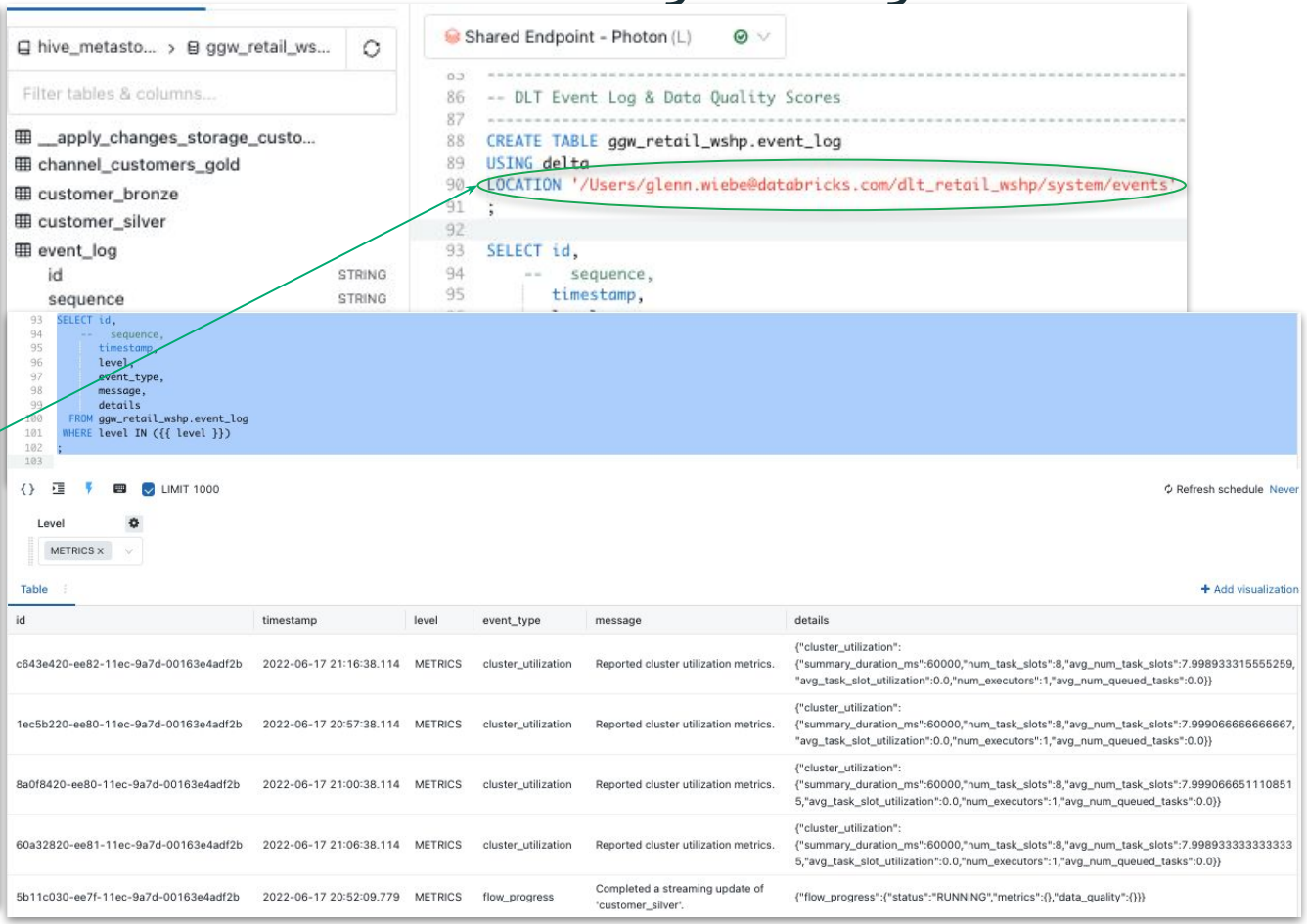
Cmd 12

```
APPLY CHANGES INTO live.customer_silver
FROM stream(live.customer_bronze2silver_v)
  KEYS (id)
  APPLY AS DELETE WHEN active = 0
  SEQUENCE BY update_dt
;
```



DLT Lab Screenshots - DLT Event Log Config





DLT Lab Screenshots - DLT Event Queries

```
-- Lineage
SELECT details:flow_definition.output_dataset,
       details:flow_definition.input_datasets,
       details:flow_definition.flow_type,
       details:flow_definition.schema
  --  , details:flow_definition.explain_text,
  --    details:flow_definition
  FROM ggw_retail_wshp.event_log
 WHERE details:flow_definition IS NOT NULL
 ORDER BY timestamp
```

LIMIT 1000 — Refresh schedule: Never

Table — + Add visualization

output_dataset	input_datasets	flow_type	schema
sales_channel_v	*NULL*	COMPLETE	[{"name":"channelId","path":["channelId"],"data_type":"INTEGER"},{"name":"channelName","path":["channelName"],"data_type":"STRING"},{"name":"description","path":["description"],"data_type":"STRING"}]
customer_bronze	*NULL*	INCREMENTAL	[{"name":"id","path":["id"],"data_type":"INTEGER"},{"name":"first_name","path":["first_name"],"data_type":"STRING"},{"name":"last_name","path":["last_name"],"data_type":"STRING"},{"name":"email","path":["email"],"data_type":"STRING"},{"name":"channel","path":["channel"],"data_type":"INTEGER"},{"name":"active","path":["active"],"data_type":"INT… {"name":"update_user","pa…
customer_bronze2silver_v	["sales_channel_v","customer_bronze"]	INCREMENTAL	[{"name":"id","path":["id"], ["last_name"],"data_type":… {"name":"active","path":["… {"name":"update_dt","path… ["dlt_ingest_dt"],"data_typ… ["dlt_ingest_principal"],"da…
customer_silver	["customer_bronze2silver_v"]	CHANGE	[{"name":"id","path":["id"], ["last_name"],"data_type":… {"name":"active","path":["… {"name":"update_dt","path… ["dlt_ingest_dt"],"data_typ… ["dlt_ingest_principal"],"da…
channel_customers_gold	["customer_silver"]	COMPLETE	[{"name":"sales_channel",… {"name":"most_recent_cus…

```
-- Flow Progress & Data Quality Results
SELECT
  id,
  details:flow_progress,
  details:flow_progress.status,
  details:flow_progress.metrics.num_output_rows,
  details:flow_progress.metrics.backlog_bytes,
  details:flow_progress.data_quality.dropped_records,
  details:flow_progress:data_quality:expectations
--   explode(from_json(details:flow_progress:data_quality:expectations
--           ,schema_of_json("[{'name':'str', 'dataset':'str', 'passed_records':42, 'failed_records':42}]"))) expectations,
FROM ggw_retail_wshp.event_log
WHERE details:flow_progress.metrics IS NOT NULL
ORDER BY timestamp
```

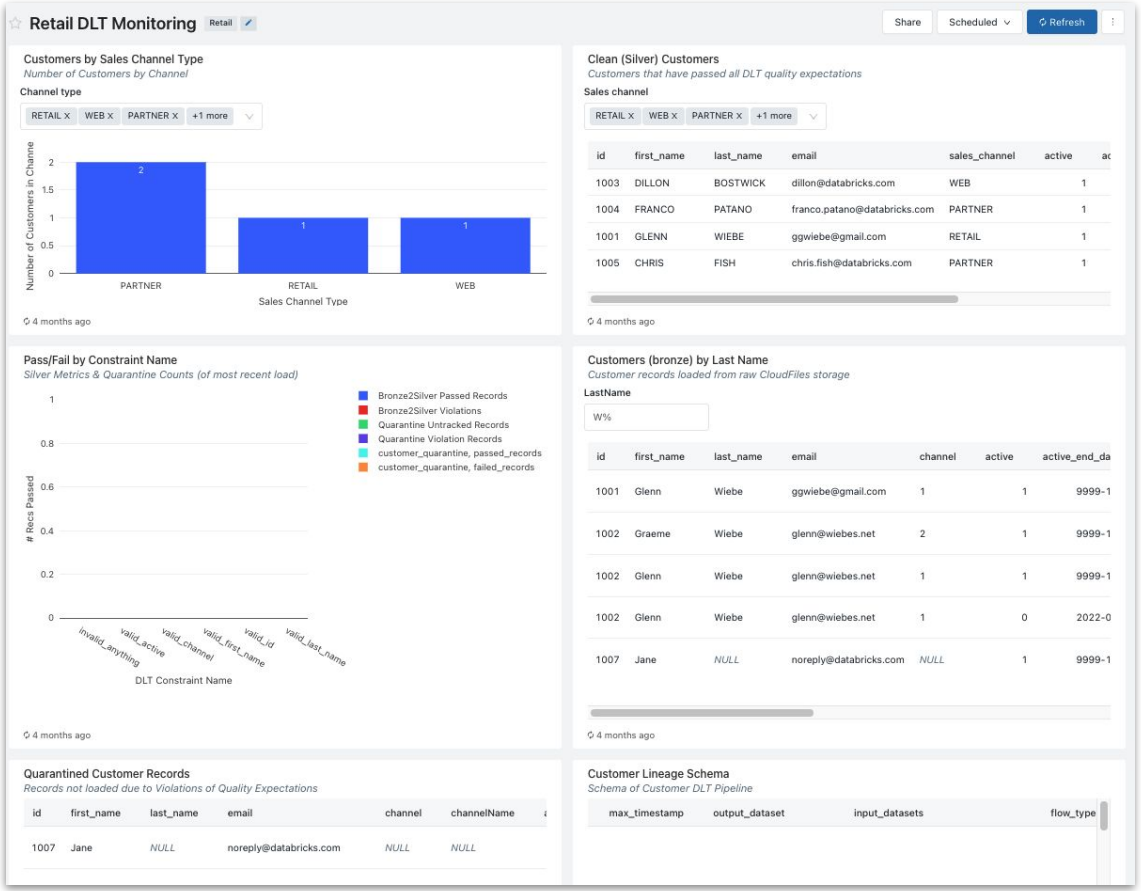
LIMIT 1000

Table

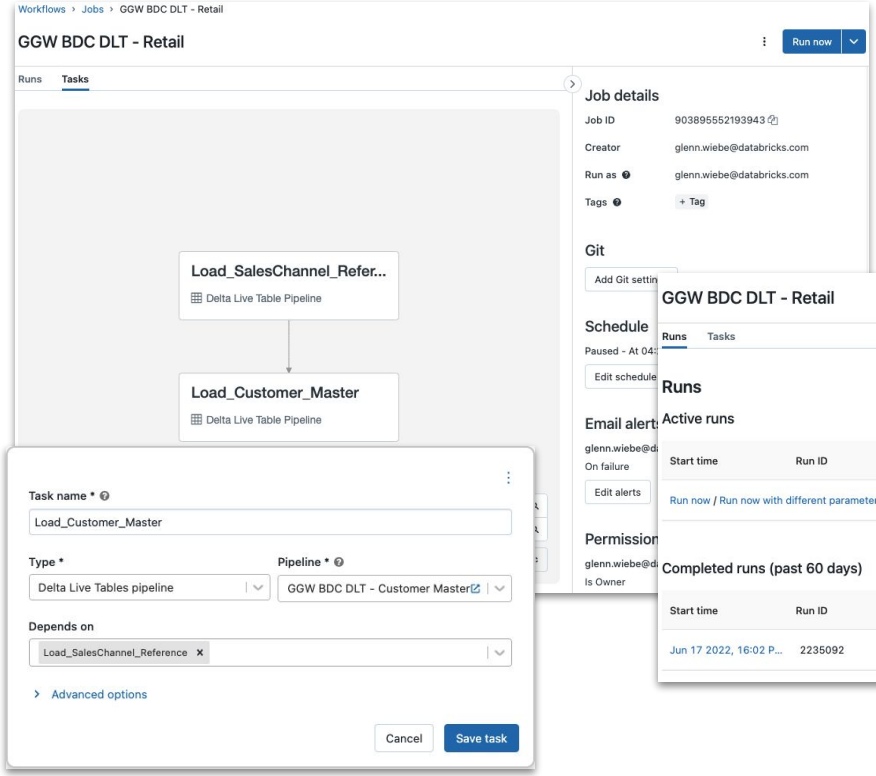
id	flow_progress	status	num_output_ro	backlog_bytes	dropped_recor	expectations
5589bc30-ee7f-11ec-9a7d-00163e4adf2b	{"status":"RUNNING","metrics":{"num_output_rows":6},"data_quality":{"dropped_records":0}}	RUNNING	6	*NULL*	0	*NULL*
56269870-ee7f-11ec-9a7d-00163e4adf2b	{"status":"COMPLETED","metrics":{"num_output_rows":6},"data_quality":{"dropped_records":0}}	COMPLETED	6	*NULL*	0	*NULL*
5b11c030-ee7f-11ec-9a7d-00163e4adf2b	{"status":"RUNNING","metrics":{},"data_quality":{}}	RUNNING	*NULL*	*NULL*	*NULL*	*NULL*
5b167b20-ee7f-11ec-9a7d-00163e4adf2b	{"status":"COMPLETED","metrics":{},"data_quality":{}}	COMPLETED	*NULL*	*NULL*	*NULL*	*NULL*
5b57f0f0-ee7f-11ec-9a7d-00163e4adf2b	{"status":"RUNNING","metrics":{"backlog_bytes":4432}}	RUNNING	*NULL*	4432	*NULL*	*NULL*
5ba6fb50-ee7f-11ec-9a7d-00163e4adf2b	{"status":"COMPLETED","metrics":{"num_output_rows":3},"data_quality":{"dropped_records":0}}	COMPLETED	3	*NULL*	0	*NULL*

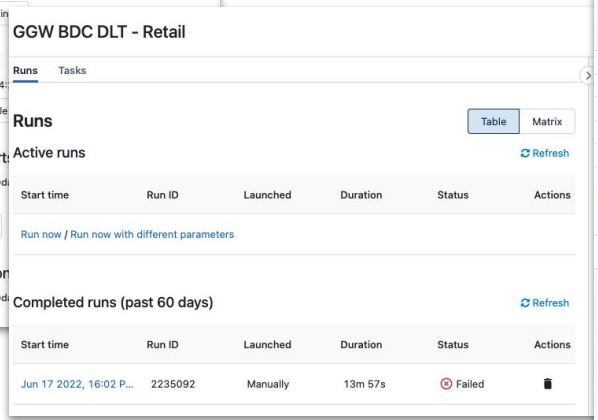
databricks

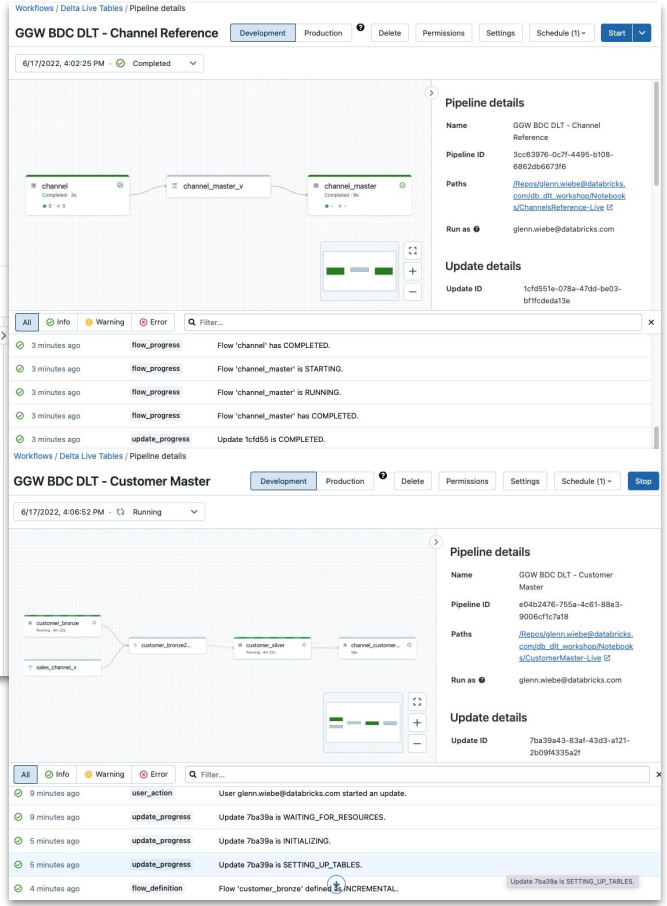
DLT Lab Screenshots - DLT Event Queries & Dashboards



DLT Lab Screenshots - Pipeline Workflows







databricks