LAB 07: Lakeflow Declarative Pipeline

Duration: ~45 min
Day: 3
After module: M07: Medallion & Lakeflow Pipelines
Difficulty: Advanced

Scenario

> *"Time to formalize the RetailHub data flow into a production-grade Medallion pipeline using Lakeflow Declarative Pipelines. You'll define Bronze (raw ingestion), Silver (cleaned/validated), and Gold (aggregates) layers with data quality expectations and SCD Type 2 for customer dimension tracking."*

Objectives

After completing this lab you will be able to: - Write `CREATE STREAMING TABLE` declarations for Bronze layer - Write `CREATE MATERIALIZED VIEW` declarations for Gold layer - Define data quality expectations (CONSTRAINT … EXPECT) - Configure a Lakeflow pipeline via the UI - Run and monitor the pipeline DAG

Part 1: Write Pipeline SQL Files (~20 min)

Task 1: Bronze Layer – Streaming Tables

Create SQL declarations for Bronze layer ingestion:

```sql
CREATE OR REFRESH STREAMING TABLE bronze_orders
AS SELECT * FROM STREAM read_files('/path/to/orders', format => 'json');
```

> **Exam Tip:** `STREAMING TABLE` processes data incrementally. Each run picks up only new files. Use `STREAM read_files()` for file-based sources.

Task 2: Silver Layer – Validated Tables

Create Silver layer with data quality expectations:

```sql
CREATE OR REFRESH STREAMING TABLE silver_orders (
    CONSTRAINT valid_order_id EXPECT (order_id IS NOT NULL) ON VIOLATION
        DROP ROW,
    CONSTRAINT valid_amount EXPECT (total_price > 0) ON VIOLATION DROP
        ROW
)
AS SELECT * FROM STREAM(LIVE.bronze_orders);
```

> **Exam Tip:** Know the three expectation actions: - `ON VIOLATION DROP ROW` – drops invalid rows - `ON VIOLATION FAIL UPDATE` – fails the pipeline - (no action) – records metric only, rows pass through

Task 3: Gold Layer – Materialized Views

Create Gold aggregation tables:

```sql
CREATE OR REFRESH MATERIALIZED VIEW gold_daily_revenue
AS SELECT order_date, SUM(total_price) AS revenue, COUNT(*) AS orders
   FROM LIVE.silver_orders
   GROUP BY order_date;
```

> **Exam Tip:** `MATERIALIZED VIEW` recomputes fully on each run (not incremental). Use for aggregations and joins on Silver data.

Part 2: Configure Pipeline in UI (~10 min)

Task 4: Create the Pipeline

<screen = Lakeflow Pipelines page showing “Create pipeline” button in the top right>

1. Go to **Lakeflow Pipelines** (left sidebar > Pipelines)
2. Click **Create pipeline**
3. Configure:

Setting	Value
Pipeline name	`retailhub_{your_name}_pipeline`
Source code	Select the SQL files from your workspace
Target catalog	Your catalog (`retailhub_{your_name}`)
Target schema	`bronze` / `silver` / `gold` (multi-schema)

<screen = Pipeline configuration dialog with name, source code path, and target catalog/schema filled in>

Task 5: Add Pipeline Variables

Add configuration variables used in your SQL files:

<screen = Pipeline Settings panel showing configuration variables section with key-value pairs>

Part 3: Run & Monitor (~15 min)

Task 6: Run the Pipeline

1. Click **Start** to trigger a Development run
2. Watch the DAG visualization
3. Check the data quality metrics in the UI

<screen = Lakeflow pipeline DAG showing bronze/silver/gold tables connected with arrows, green checkmarks for completed steps>

Task 7: Verify Results

Open the notebook and query the pipeline results:

<screen = Query results showing gold_daily_revenue table with dates and revenue amounts>

Task 8: Check Expectations

Review the data quality metrics: - How many rows passed each expectation? - How many were dropped?

<screen = Pipeline run details showing expectations metrics: rows passed, rows dropped, percentage>

Summary

In this lab you: - Wrote SQL declarations for a full Medallion pipeline (Bronze/Silver/Gold) - Added data quality expectations (DROP ROW, FAIL UPDATE) - Configured and ran a Lakeflow pipeline via the UI - Monitored the DAG and verified data quality metrics

> **Exam Tip:** Lakeflow (formerly DLT) pipelines are **declarative** – you define WHAT, not HOW. The engine handles orchestration, checkpointing, error handling, and data lineage.

Feature	STREAMING TABLE	MATERIALIZED VIEW
Processing	Incremental	Full recompute
Use case	Raw ingestion, append-only	Aggregations, joins
In `LIVE.` ref	`STREAM(LIVE.x)`	`LIVE.x`

> **Next:** LAB 08 - Lakeflow Jobs: Triggers, Dependencies & Orchestration