01 Databricks Lakeflow Declarative

Comprehensive Step-by-Step Guide: Databricks Lakeflow Declarative Pipeline with Medallion Architecture

Project Overview

This guide walks you through building an end-to-end data engineering solution using Databricks Lakeflow Declarative Pipelines, implementing a Medallion Architecture (Bronze, Silver, Gold) with automated incremental data loading.

---

1. Prerequisites and Data Preparation

Data Files Required

You need the following CSV files:

Fact Table:

- `fact_sales.csv` - Contains transaction data with columns:
  - sale_id, order_date, customer_id, product_id, quantity, discount, region_id, channel, promo_code
  - Approximately 50,000 records
  - Note: Contains null values in quantity, channel, and promo_code columns

Dimension Tables:

- `customer.csv` - Customer details:
  - customer_id, first_name, last_name, email, join_date, VIP
  - Contains some missing email values
- `product.csv` - Product information:
  - product_id, product_name, category, price, in_stock
- `region.csv` - Region details:
  - region_id, region_name, country

Additional File for Testing:

- `fact_sales_2025.csv` - For testing incremental load (approximately 2,526 records)

Account Setup

- Sign up for Databricks (free version available)
- Ensure you have access to Azure Databricks workspace

---

2. Unity Catalog and Schema Setup

Step 2.1: Create Unity Catalog

1. Navigate to **Catalog** in Databricks workspace
2. Click **Create Catalog**
3. Name: `lakeflow_dlt`
4. Click **Create**

Step 2.2: Create Four Schemas

Create the following schemas within the `lakeflow_dlt` catalog:

a) Landing Zone Schema:

1. Click on catalog name `lakeflow_dlt`
2. Click **Create Schema** (top right)
3. Name: `landing_zone`
4. Click **Create**

b) Bronze Schema:

1. Click **Create Schema**
2. Name: `bronze`
3. Click **Create**

c) Silver Schema:

1. Click **Create Schema**
2. Name: `silver`
3. Click **Create**

d) Gold Schema:

1. Click **Create Schema**
2. Name: `gold`
3. Click **Create**

Result: You should now have:

- `lakeflow_dlt.landing_zone`
- `lakeflow_dlt.bronze`
- `lakeflow_dlt.silver`
- `lakeflow_dlt.gold`

---

3. Volume Creation and Data Ingestion

Step 3.1: Create Managed Volume

1. Navigate to `lakeflow_dlt` catalog
2. Select `landing_zone` schema
3. Click **Create Volume**
4. Name: `fact_and_dimension_files`
5. Type: **Managed Volume**
6. Click **Create**

Step 3.2: Create Directory Structure

Create separate directories for each file to avoid conflicts:

a) Fact Sales Directory:

1. Inside the volume, click **Create Directory**
2. Name: `fact_sales`
3. Click **Create**
4. Double-click to open the directory
5. Click **Upload**
6. Select `fact_sales.csv`
7. Upload the file

b) Customers Directory:

1. Navigate back to main volume directory
2. Click **Create Directory**
3. Name: `dim_customers`
4. Click **Create**
5. Open directory and upload `customer.csv`

c) Products Directory:

1. Navigate back to main volume directory
2. Click **Create Directory**
3. Name: `dim_products`
4. Click **Create**
5. Open directory and upload `product.csv`

d) Regions Directory:

1. Navigate back to main volume directory
2. Click **Create Directory**
3. Name: `dim_regions`
4. Click **Create**

5. Open directory and upload `region.csv`

Important: Each file must be in its own directory for proper AutoLoader functionality.

---

4. Lakeflow Pipeline Creation

Step 4.1: Initialize Pipeline

1. Navigate to **Jobs and Pipelines** (or **Workflows**)
2. Click **Build ETL Pipeline**
3. Pipeline Name: `databricks_declarative_lakeflow_pipeline`
4. Press **Enter** to save

Step 4.2: Configure Pipeline Settings

1. **Unity Catalog:** Select `lakeflow_dlt`
2. **Schema:** Select `landing_zone`
3. **Start Option:** Choose **Empty File** (not sample code)
4. **Language:** Select **SQL**
5. Click **Select**

Step 4.3: Understand Pipeline Interface

- **Left Panel:** Pipeline assets and transformations
- **Center:** Code editor
- **Right Panel:** Pipeline graph visualization
- **Bottom:** Tables, performance metrics, and results
- **Top Right:** Settings, dry run, and run pipeline buttons

Step 4.4: Enable New Pipeline Editor

1. Toggle **Lakeflow pipelines editor** ON (recommended)
2. This provides better visualization and debugging capabilities

---

5. Bronze Layer - Data Ingestion

Step 5.1: Create Bronze Transformation File

1. In the transformations folder, you'll see `my_transformation.sql`
2. Right-click or click kebab menu
3. Select **Rename**
4. New name: `bronze_transformations.sql`
5. Click **OK**

Step 5.2: Write Bronze Layer Code

Fact Sales Ingestion:

```sql
-- Ingest data from volume to bronze schema

CREATE STREAMING LIVE TABLE bronze.fact_sales
COMMENT "Raw fact sales from volume to bronze schema"
AS SELECT * FROM CLOUD_FILES(
  '/Volumes/lakeflow_dlt/landing_zone/fact_and_dimension_files/fact_sales',
  'csv',
  map('header', 'true')
);
```

Products Ingestion:

```sql
CREATE STREAMING LIVE TABLE bronze.products
COMMENT "Raw products from volume to bronze schema"
AS SELECT * FROM CLOUD_FILES(
  '/Volumes/lakeflow_dlt/landing_zone/fact_and_dimension_files/dim_products',
  'csv',
  map('header', 'true')
);
```

Customers Ingestion:

```sql
CREATE STREAMING LIVE TABLE bronze.customers
COMMENT "Raw customers from volume to bronze schema"
AS SELECT * FROM CLOUD_FILES(
  '/Volumes/lakeflow_dlt/landing_zone/fact_and_dimension_files/dim_customers',
  'csv',
  map('header', 'true')
);
```

Regions Ingestion:

```sql
CREATE STREAMING LIVE TABLE bronze.regions
COMMENT "Raw regions from volume to bronze schema"
AS SELECT * FROM CLOUD_FILES(
  '/Volumes/lakeflow_dlt/landing_zone/fact_and_dimension_files/dim_regions',
  'csv',
  map('header', 'true')
);
```

Step 5.3: Get Volume Paths

Instead of typing paths manually:

1. Open **Catalog** panel on the right
2. Navigate to: `lakeflow_dlt` > `landing_zone` > `fact_and_dimension_files`
3. Expand to see directories
4. Click the **double chevron (>>)** icon next to directory name
5. Path auto-populates in your query
6. Remove trailing forward slash if present

Step 5.4: Validate with Dry Run

1. Click **Dry Run** button (top right)
2. This validates schema and syntax without executing
3. Check for errors in the output panel
4. Fix any issues (common: duplicate names, typos)
5. Wait for "Run completed" success message

Step 5.5: Execute Pipeline

1. Click **Run** button
2. Wait for pipeline execution
3. Monitor progress in pipeline graph
4. Green checkmarks indicate success
5. View tables in **All Tables** section

Step 5.6: Verify Results

1. Click **Maximize** to view full pipeline graph
2. Verify all four streaming tables created:
   - `bronze.fact_sales` (50,000 records)
   - `bronze.customers` (5,000 records)
   - `bronze.products` (500 records)
   - `bronze.regions` (10 records)
3. Click on any table to view:
   - **Preview:** Sample data
   - **Table Metrics:** Row counts, duration
   - **Columns:** Data types (all string initially)

---

6. Silver Layer - Data Cleaning

Step 6.1: Create Silver Transformation File

1. Click **Create File** in transformations folder
2. Language: **SQL**
3. Name: `silver_transformations.sql`
4. Dataset Type: **Streaming Table**
5. Click **Create**

Step 6.2: Write Data Cleaning Code

Fact Sales Cleaning:

```sql
-- Fact sales data cleaning

CREATE STREAMING LIVE TABLE silver.fact_sales
COMMENT "Fact sales data cleaning"
AS SELECT
  CAST(sale_id AS INT) AS sale_id,
  TO_DATE(order_date, 'dd/MM/yyyy') AS order_date,
  CAST(customer_id AS INT) AS customer_id,
  CAST(product_id AS INT) AS product_id,
  CAST(quantity AS INT) AS quantity,
  CAST(discount AS INT) AS discount,
  CAST(region_id AS INT) AS region_id,
  CAST(channel AS STRING) AS channel,
  CAST(promo_code AS STRING) AS promo_code
FROM STREAM(lakeflow_dlt.bronze.fact_sales);
```

Products Cleaning:

```sql
CREATE STREAMING LIVE TABLE silver.products
COMMENT "Products data cleaning"
AS SELECT
  CAST(product_id AS INT) AS product_id,
  CAST(product_name AS STRING) AS product_name,
  CAST(category AS STRING) AS category,
  CAST(price AS INT) AS price,
  CAST(in_stock AS INT) AS in_stock
FROM STREAM(lakeflow_dlt.bronze.products);
```

Customers Cleaning:

```sql
CREATE STREAMING LIVE TABLE silver.customers
COMMENT "Customers data cleaning"
AS SELECT
  CAST(customer_id AS INT) AS customer_id,
  CAST(first_name AS STRING) AS first_name,
  CAST(last_name AS STRING) AS last_name,
  CAST(email AS STRING) AS email,
  TO_DATE(join_date, 'dd/MM/yyyy') AS join_date,
  CAST(VIP AS STRING) AS VIP
FROM STREAM(lakeflow_dlt.bronze.customers);
```

Regions Cleaning:

```sql
CREATE STREAMING LIVE TABLE silver.regions
COMMENT "Regions data cleaning"
AS SELECT
  CAST(region_id AS INT) AS region_id,
  CAST(region_name AS STRING) AS region_name,
  CAST(country AS STRING) AS country
FROM STREAM(lakeflow_dlt.bronze.regions);
```

Step 6.3: Important - Add STREAM() Wrapper

Critical Fix: Wrap the source path with `STREAM()` to enable incremental processing:

```sql
FROM STREAM(lakeflow_dlt.bronze.fact_sales)
```

This prevents the error: "Cannot create a streaming table append flow from a batch query"

Step 6.4: Run and Verify

1. Click **Dry Run** to validate
2. Fix any errors (common: duplicate table names, missing STREAM wrapper)
3. Click **Run**
4. Verify 8 total streaming tables (4 bronze + 4 silver)
5. Check data types in silver tables are properly converted

---

7. Silver Layer - Materialized Views with Constraints

Step 7.1: Create Cleaned Sales Data File

1. Click **Create File**
2. Language: **SQL**
3. Name: `cleaned_sales_data.sql`
4. Dataset Type: **Materialized View**
5. Click **Create**

Step 7.2: Write Materialized View with Constraints

```sql
CREATE OR REFRESH MATERIALIZED VIEW silver.cleaned_sales_data
(
  -- Define data quality constraints
  CONSTRAINT sales_quantity_check EXPECT (quantity IS NOT NULL) ON VIOLATION DROP ROW,
  CONSTRAINT sales_channel_check EXPECT (channel IS NOT NULL) ON VIOLATION DROP ROW,
  CONSTRAINT sales_promo_code_check EXPECT (promo_code IS NOT NULL) ON VIOLATION DROP ROW,
  CONSTRAINT customer_email_check EXPECT (email IS NOT NULL) ON VIOLATION DROP ROW
)
COMMENT "Cleaned sales data"
AS SELECT
  -- Fact sales columns
  FS.sale_id,
  FS.order_date,
  FS.customer_id,
  FS.product_id,
  FS.quantity,
  FS.discount,
  FS.region_id,
  FS.channel,
  FS.promo_code,

  -- Customer columns (exclude primary key)
  C.first_name,
  C.last_name,
  C.email,
  C.join_date,
  C.VIP,

  -- Product columns (exclude primary key)
  P.product_name,
  P.category,
  P.price,
  P.in_stock,

  -- Region columns (exclude primary key)
  R.region_name,
  R.country,

  -- Calculated column
  FS.quantity * P.price AS revenue
```

```
FROM lakeflow_dlt.silver.fact_sales FS

LEFT JOIN lakeflow_dlt.silver.customers C
  ON FS.customer_id = C.customer_id

LEFT JOIN lakeflow_dlt.silver.products P
  ON FS.product_id = P.product_id

LEFT JOIN lakeflow_dlt.silver.regions R
  ON FS.region_id = R.region_id;
```

Step 7.3: Understanding Constraints

Constraint Syntax:

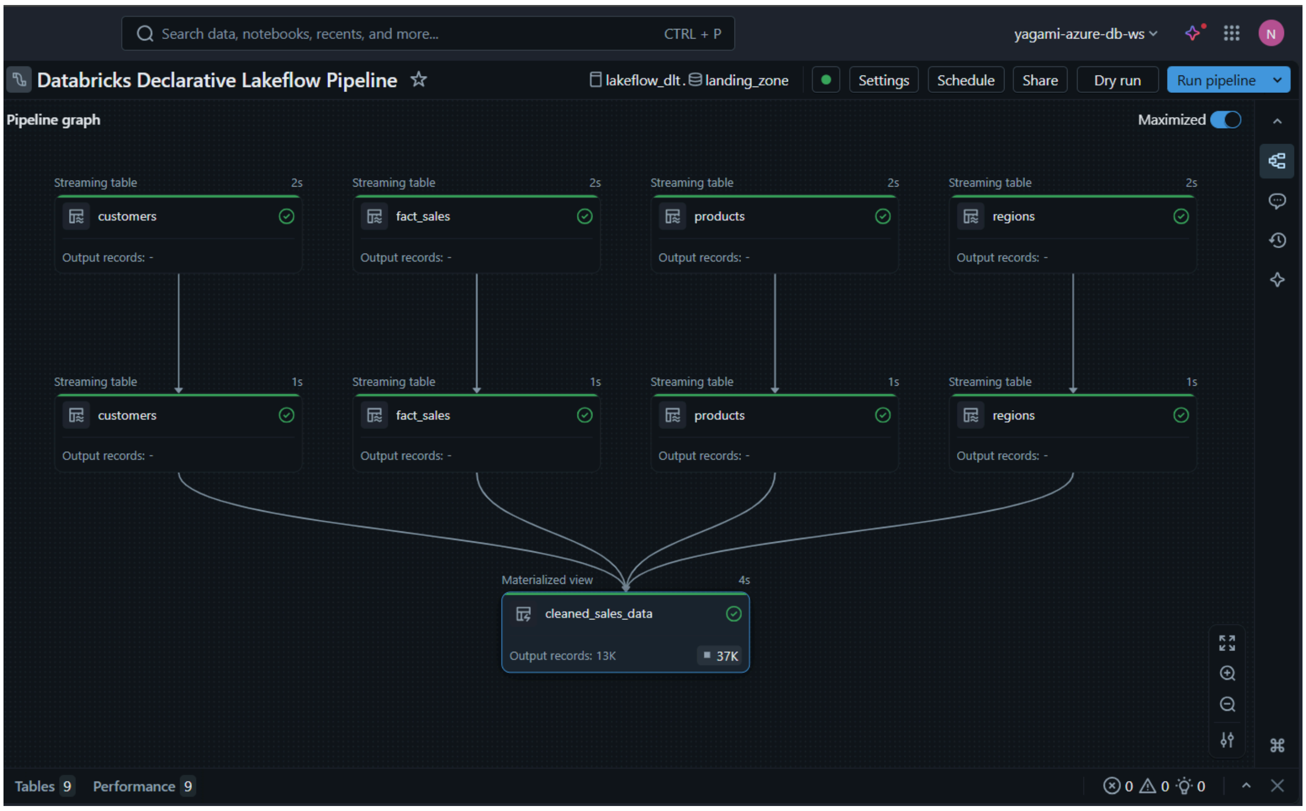
```sql
CONSTRAINT <constraint_name> EXPECT (<condition>) ON VIOLATION <action>
```

Actions:

- `DROP ROW` - Removes rows that violate constraint
- `FAIL` - Stops pipeline on violation
- `QUARANTINE` - Moves bad data to separate table

Step 7.4: Run and Validate

1. Click **Dry Run**
2. Click **Run**
3. View pipeline graph - materialized view shown with turbo icon
4. Check **Table Metrics** to see:
   - Total records written: ~12,935 (25.9%)
   - Total records dropped: ~37,065 (74.1%)
   - Breakdown by constraint


Search data, notebooks, recents, and more...
CTRL + P
yagami-azure-db-ws
Databricks Declarative Lakeflow Pipeline
lakeflow_dlt. landing_zone
Settings
Schedule
Share
Dry run
Run pipeline
Pipeline graph
Maximized
Streaming table 2s customers Output records: -
Streaming table 2s fact_sales Output records: -
Streaming table 2s products Output records: -
Streaming table 2s regions Output records: -
Streaming table 1s customers Output records: -
Streaming table 1s fact_sales Output records: -
Streaming table 1s products Output records: -
Streaming table 1s regions Output records: -
Materialized view 4s cleaned_sales_data Output records: 13K 37K
Tables 9
Performance 9
0 0 0

Step 7.5: Analyze Data Quality Results

Click on table metrics to see:

- **sales_promo_code_check:** 36,533 rows dropped
- **sales_channel_check:** 999 rows dropped (2%)
- **customer_email_check:** 971 rows dropped
- **sales_quantity_check:** 44 rows dropped (<1%)

Note: In production, 74% drop rate is high. This demonstrates the feature; adjust constraints to business needs.

---

8. Gold Layer - Business Aggregations

Step 8.1: Create Gold Transformations File

1. Click **Create File**
2. Language: **SQL**
3. Name: `gold_transformations.sql`
4. Dataset Type: **Materialized View**
5. Click **Create**

Step 8.2: Write Business Aggregate Queries

Aggregate 1: Category Sales Summary

```sql
CREATE OR REFRESH MATERIALIZED VIEW gold.category_sales_summary
AS SELECT
  category,
  YEAR(order_date) AS year,
  SUM(revenue) AS total_revenue
FROM lakeflow_dlt.silver.cleaned_sales_data
GROUP BY category, YEAR(order_date)
ORDER BY category, year;
```

Aggregate 2: Revenue by Customer in Each Region (Ranked)

```sql
CREATE OR REFRESH MATERIALIZED VIEW
gold.revenue_by_customers_in_each_region_by_ranking
AS SELECT
  region_name,
  customer_id,
  first_name,
  last_name,
  SUM(revenue) AS total_revenue,
  RANK() OVER (PARTITION BY region_name ORDER BY SUM(revenue) DESC) AS revenue_rank
FROM lakeflow_dlt.silver.cleaned_sales_data
GROUP BY region_name, customer_id, first_name, last_name
ORDER BY region_name, revenue_rank;
```

Aggregate 3: Customer Lifetime Value Estimation

```sql
CREATE OR REFRESH MATERIALIZED VIEW gold.customer_lifetime_value_estimation
AS SELECT
  customer_id,
  first_name,
  last_name,
  email,
  VIP,
  COUNT(DISTINCT sale_id) AS total_orders,
  SUM(quantity) AS total_items_purchased,
  SUM(revenue) AS lifetime_value,
  AVG(revenue) AS average_order_value,
  MIN(order_date) AS first_purchase_date,
  MAX(order_date) AS last_purchase_date,
  DATEDIFF(MAX(order_date), MIN(order_date)) AS customer_tenure_days
FROM lakeflow_dlt.silver.cleaned_sales_data
GROUP BY customer_id, first_name, last_name, email, VIP
ORDER BY lifetime_value DESC;
```

Step 8.3: Execute and Verify

1. Click **Dry Run**
2. Click **Run**
3. View complete pipeline graph showing full medallion architecture
4. Verify three gold layer materialized views:
   - `gold.category_sales_summary` (~56 records)
   - `gold.revenue_by_customers_in_each_region_by_ranking` (~52 records)
   - `gold.customer_lifetime_value_estimation` (~4,600 records)

Step 8.4: Customize Pipeline Graph View

1. Click **Maximize** for full view
2. Use **Fit to View** to adjust zoom
3. Change **Graph Orientation:**
   - Horizontal Layout (default)
   - **Vertical Layout** (recommended for medallion flow)
4. Toggle **Mini Map** on/off as needed
5. Click **Refresh** in catalog to see all tables

---

9. Scheduling and Automation

Step 9.1: Configure Pipeline Schedule

1. Click **Schedule** button in pipeline interface
2. **Schedule Type:** Choose frequency
   - **Option A:** Advanced - Every 3 minutes (for demo/testing)
   - **Option B:** Daily, Weekly, or Custom cron
3. **Schedule Name:** Keep default or customize
4. **Time Zone:** Verify correct timezone
5. Click **Create Schedule**

Step 9.2: Verify Schedule

1. Schedule status shows **Active**
2. Pipeline will run automatically at specified intervals
3. Monitor execution in **Jobs and Pipelines**

Step 9.3: Compute Configuration (Optional)

By default, pipeline uses:

- **Serverless Compute** (recommended)
- Auto-scaling based on workload
- No manual cluster management required

To modify:

1. Click **Settings** (gear icon)
2. Adjust compute settings if needed
3. Review JSON/YAML configuration

---

10. Dashboard Creation

Step 10.1: Create New Dashboard

1. Navigate to **Dashboards**
2. Click **Create Dashboard**
3. Name: `Category Sales Summary` (or your preference)

Step 10.2: Add Data Source

1. Click **Add** > **Visualization**
2. Select **Query Data**
3. Navigate to: `lakeflow_dlt` > `gold` > `category_sales_summary`
4. Click table to select
5. Wait for data to load (serverless warehouse starts automatically)

Step 10.3: Create Visualizations

Visualization 1: Total Revenue Counter

1. Click **Add Visualization**
2. In AI assistant prompt: "Show total revenue as counter"
3. Press **Enter**
4. AI generates counter showing total revenue (e.g., £2.06M)

Visualization 2: Revenue by Category Bar Chart

1. Click **Add Visualization**
2. Prompt: "Revenue by category using bar chart"
3. Press **Enter**
4. AI generates horizontal or vertical bar chart

Visualization 3: Year Filter

1. Click **Add Filter**
2. Select **Add Field**
3. Choose **year** column
4. Shows available years (2018-2024)
5. Users can filter dashboard by year

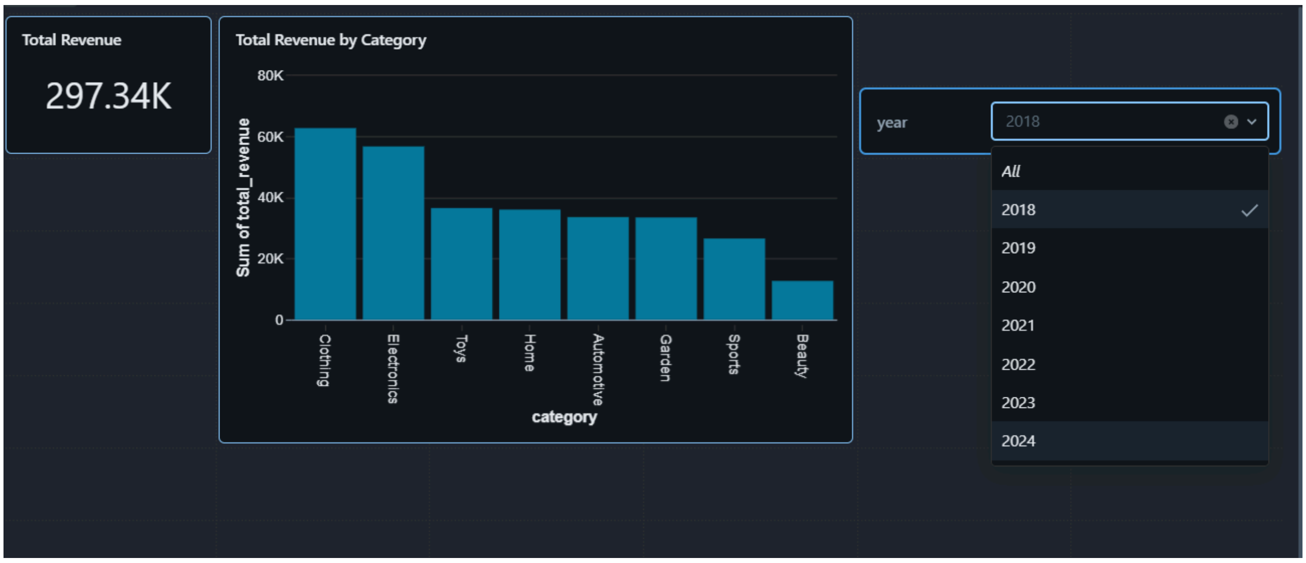
Step 10.4: Arrange Dashboard

1. Drag and resize visualizations
2. Position counter at top

3. Place bar chart prominently
4. Add year filter for interactivity

Step 10.5: Publish Dashboard

1. Click **Publish** button
2. Dashboard is now live and accessible
3. Share URL with stakeholders



---

11. Testing Incremental Load

Step 11.1: Prepare Test Data

You should have `fact_sales_2025.csv` ready with:

- Approximately 2,526 records
- Same schema as original fact_sales.csv
- Data for year 2025

Step 11.2: Upload New Data

1. Navigate to **Catalog**
2. Go to: `lakeflow_dlt` > `landing_zone` > `fact_and_dimension_files`
3. Open `fact_sales` directory
4. Click **Upload to this volume**
5. Click **Browse**
6. Select `fact_sales_2025.csv`
7. Click **Upload**

Step 11.3: Monitor Scheduled Execution

1. Navigate to **Jobs and Pipelines**
2. Find your scheduled pipeline
3. Wait for next scheduled run (3 minutes if using that interval)

4. Monitor execution status
5. Pipeline automatically detects new file via AutoLoader

Step 11.4: Verify Incremental Load

Check Pipeline Graph:

1. Open pipeline after execution
2. Record counts should increase
3. `bronze.fact_sales`: ~52,500 records (50,000 + 2,526 - duplicates removed)
4. Downstream tables update accordingly

Check Dashboard:

1. Refresh your dashboard
2. Year filter now shows **2025** option
3. Select 2025 to see new data
4. Total revenue includes 2025 transactions (£935 for 2025)
5. Bar chart shows 2025 category breakdown

Verify in Catalog:

1. Go to `gold.category_sales_summary`
2. Click **Sample Data**
3. Scroll to find 2025 records
4. Confirm new year data is present

Step 11.5: Validate AutoLoader Functionality

AutoLoader automatically:

- ✅ Detects new files in volume directories
- ✅ Processes only new/changed files (incremental)
- ✅ Maintains state to avoid reprocessing
- ✅ Handles schema evolution (if configured)
- ✅ Provides exactly-once processing guarantees

Additional Features and Best Practices

Pipeline Management

Version History:

- Click **Version History** to see all changes
- Track who made changes and when
- Revert to previous versions if needed

Collaboration:

- Click **Comments** icon to add annotations
- Team members can review and discuss
- Tag colleagues for feedback

Error Handling:

- View **Flows** tab to see execution details
- Check **Expectations** section for constraint violations
- Review **Failed** records for debugging

Performance Optimization

1. Partition Strategy:

```sql
CREATE STREAMING LIVE TABLE bronze.fact_sales
PARTITIONED BY (YEAR(order_date))
AS SELECT * FROM ...
```

2. Z-Ordering:

```sql
CREATE OR REFRESH MATERIALIZED VIEW gold.customer_lifetime_value_estimation
CLUSTER BY (lifetime_value)
AS SELECT ...
```

3. Auto Optimize: Enable in pipeline settings for automatic file compaction

Data Quality Enhancements

Custom Expectations:

```sql
CONSTRAINT price_validation
EXPECT (price > 0)
ON VIOLATION DROP ROW

CONSTRAINT date_range_check
EXPECT (order_date BETWEEN '2018-01-01' AND CURRENT_DATE())
ON VIOLATION QUARANTINE
```

Quarantine Tables: Use `ON VIOLATION QUARANTINE` to review bad data later instead of dropping

Security Configuration

1. Table ACLs:

```sql
GRANT SELECT ON TABLE gold.category_sales_summary TO `analysts@company.com`;
```

2. Row-Level Security:

```sql
CREATE OR REFRESH MATERIALIZED VIEW gold.regional_sales
AS SELECT * FROM cleaned_sales_data
WHERE region_name = current_user_region();
```

Troubleshooting Common Issues

Issue 1: "Cannot redefine dataset"

Cause: Duplicate table names in same schema
Solution: Ensure unique names for each streaming table/materialized view

Issue 2: "Cannot create streaming table from batch query"

Cause: Missing `STREAM()` wrapper
Solution: Wrap source with `STREAM(source_table)`

Issue 3: AutoLoader not picking up new files

Cause: File format or path mismatch
Solution:

- Verify file format matches (csv, parquet, etc.)
- Check directory path is correct
- Ensure file structure matches original

Issue 4: High constraint violation rate

Cause: Overly strict constraints or data quality issues
Solution:

- Review constraint logic
- Analyze source data quality
- Consider `QUARANTINE` instead of `DROP ROW`

Issue 5: Dashboard not updating

Cause: Cached results or warehouse stopped
Solution:

- Refresh dashboard manually
- Check if serverless warehouse is running
- Verify pipeline executed successfully

Project Architecture Summary

```
┌─────────────────────────────────────────────────────────────┐
│                       SOURCE FILES                          │
│  fact_sales.csv, customer.csv, product.csv,               │
│  region.csv                                                 │
└─────────────────────────────────────────────────────────────┘
                  │
                  │
                  ▼
┌─────────────────────────────────────────────────────────────┐
│                 LANDING ZONE (Volumes)                      │
│  Managed volumes with directory structure                   │
└─────────────────────────────────────────────────────────────┘
                  │
                  │ AutoLoader (CLOUD_FILES)
                  ▼
┌─────────────────────────────────────────────────────────────┐
│              BRONZE LAYER (Raw Data)                        │
│  Streaming Live Tables - No transformations                 │
│  All columns as STRING data type                            │
└─────────────────────────────────────────────────────────────┘
                  │
                  │ Data Type Casting
                  ▼
┌─────────────────────────────────────────────────────────────┐
│             SILVER LAYER (Cleaned Data)                     │
│  Streaming Live Tables - Proper data types                  │
│  Materialized Views - With constraints                      │
│  Data quality rules enforced                                │
└─────────────────────────────────────────────────────────────┘
                  │
                  │ Business Logic
                  ▼
┌─────────────────────────────────────────────────────────────┐
│           GOLD LAYER (Business Aggregates)                  │
│  Materialized Views - Analytics ready                       │
│  KPIs, summaries, and reporting tables                      │
└─────────────────────────────────────────────────────────────┘
                  │
                  │
                  ▼
┌─────────────────────────────────────────────────────────────┐
│                 DASHBOARDS & BI                             │
│  Interactive visualizations                                 │
│  Real-time insights                                         │
└─────────────────────────────────────────────────────────────┘
```

Key Learnings and Takeaways

1. **Declarative Pipelines:** Define what you want, not how to do it
2. **AutoLoader:** Automatically handles incremental data ingestion
3. **Medallion Architecture:** Clear separation of raw, cleaned, and aggregated data
4. **Data Quality:** Built-in constraints ensure data integrity
5. **Streaming Tables:** Enable real-time data processing
6. **Materialized Views:** Pre-computed results for faster queries
7. **Serverless Compute:** No cluster management required
8. **End-to-End Automation:** From ingestion to visualization

---

Next Steps and Enhancements

1. **Add more data sources:** Kafka, Kinesis, S3 events
2. **Implement Change Data Capture (CDC):** Track data changes
3. **Add data lineage tracking:** Understand data flow
4. **Integrate with ML workflows:** Use gold tables for training
5. **Set up alerting:** Notify on pipeline failures or data quality issues
6. **Implement slowly changing dimensions (SCD):** Track historical changes
7. **Add data retention policies:** Manage storage costs
8. **Create additional dashboards:** For different business units

---

Resources

- **Databricks Documentation:** https://docs.databricks.com
- **Unity Catalog Guide:** https://docs.databricks.com/unity-catalog
- **Delta Live Tables:** https://docs.databricks.com/delta-live-tables
- **AutoLoader:** https://docs.databricks.com/ingestion/auto-loader

---

Congratulations! You've completed a comprehensive, production-ready data engineering pipeline using Databricks Lakeflow Declarative Pipelines with automated incremental loading and full medallion architecture implementation.