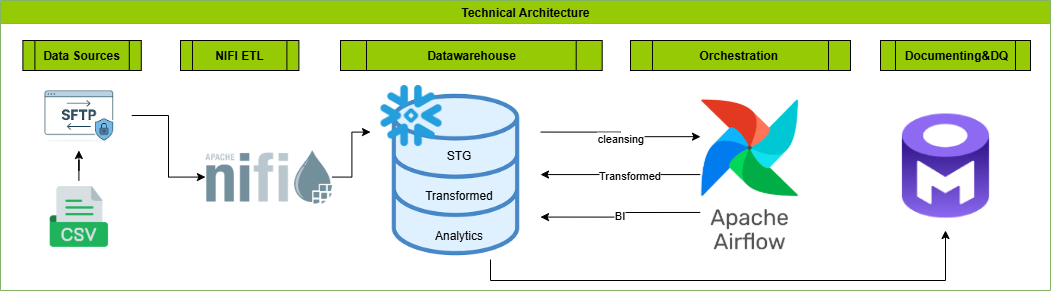
# **ETL Pipeline Documentation**

## **1. Project Overview**

* **Objective**: *Automate ETL pipeline to extract financial order data (CSV) from source system and load into Snowflake data warehouse.*
* **Scope**: The primary goal of this pipeline is to:
* **Data Extraction**: Extract CSV files using Apache NiFi.
* **Data Transformation**: NiFi: Add enrichment metadata during extraction.  
   Snowflake: Cleanse and transform data from staging to transformed layer.
* **Data Loading**: Load raw data into Snowflake staging layer, then load cleansed data into transformed layer.
* **Expected Outcome**: *Order data available in Snowflake for reporting and analysis.*

## **2. System Architecture**



* **Components Overview**:
* **Data Source**: *Flat files (CSV) containing financial order data from source system.*
* **Apache NiFi**: *Extracts CSV files,Adds enrichment columns and loads raw data into Snowflake staging layer.*
* ***Snowflake Data Warehouse:***

1. *Staging Layer: Stores raw data as ingested.*
2. *Transformed Layer: SQL-based data cleansing and transformation.*
3. *Analytical Layer: Final, curated data for analytics and data marts.*

* **Apache Airflow**: *Automates SQL transformations from staging to transformed layer.*
* **OpenMetadata**: Runs data quality checks and documentation.

## **3. Prerequisites and Setup**

* **Apache NiFi**:Install and configure **Apache NiFi**
* **Apache Airflow**: Install and set up **Apache Airflow**.
* **Snowflake:** Create and configure **Snowflake** account.
* **Openmetadata**:Install **OpenMetadata** and connect to **Snowflake**.

## **4. Apache NIFI Integration**

This section outlines the detailed steps taken to implement the data ingestion and enrichment pipeline using **Apache NiFi**, followed by loading the data into **Snowflake** for further processing.

### **1. Apache NiFi Installation and Setup**

#### **1.1. Download and Install**

* **Source**:  
   Downloaded Apache NiFi binary from:  
  <https://www.apache.org/dyn/closer.lua?path=/nifi/2.6.0/nifi-2.6.0-bin.zip>
* **Java Configuration**:  
   Configured the JAVA\_HOME environment variable to point to a compatible Java version.

#### **1.2. Run NiFi**

* **Commands Used**:

nifi.cmd start # Start NiFi in background

nifi.cmd status # Check NiFi status

nifi.cmd run # Run NiFi in foreground (optional)

* **Access Interface**:  
   NiFi UI available at:<https://localhost:8443/nifi>
* **Login Credentials** (extracted from application logs):
  + **Username**: ddb8f1c5-5683-4a87-bdf3-2d811280e477
  + **Password**: Z40Wzw9XIFrN6C6E8G6T9AeiWPSCoHxG

### **2. NiFi Data Pipeline Implementation**

#### **2.1. Ingest and Archive CSV Files**

##### **Goal:** Read CSV files from a local directory and archive them after ingestion.

* **Input Directory**: C:\Incoming\Orders
* **Archive Directory**: C:\Archived\Orders

##### **NiFi Processors Used:**

| **Processor** | **Purpose** |
| --- | --- |
| ListFile | Lists files in the input directory without consuming them |
| FetchFile | Reads file content and moves original files to archive |
| LogAttribute | Logs FlowFile metadata for debugging and verification |

##### **Outcome:**

* Files are successfully ingested into NiFi.
* FlowFiles are created for each file content.
* Original files are archived safely to C:\Archived\Orders.

#### **2.2. Enrich CSV Files**

##### **Goal:**Enhance each record with additional metadata fields:

| **Column** | **Description** |
| --- | --- |
| file\_name | Original file name |
| ingested\_at | Timestamp of ingestion |
| tbl\_dt | ${filename:replaceAll('.\*\_(\\d{4}\_\\d{2}\_\\d{2})\_.\*','$1'):replaceAll('\_','')} |
| offset\_id | Unique UUID per record |

##### 

##### 

##### 

##### 

##### **NiFi Processors Used:**

| **Processor** | **Purpose** |
| --- | --- |
| UpdateAttribute | Adds FlowFile-level attributes: FILE\_NAME, INGESTED\_AT, TBL\_DT |
| UpdateRecord | Adds attributes as new fields to each CSV record |

##### **Outcome:**Each row in the CSV is enriched with the following new columns:

* file\_name: e.g., 201903\_Orders\_2019\_04\_01\_20\_39\_17.csv
* ingested\_at: e.g., 2025-09-26 14:05:00
* tbl\_dt: e.g., 2019-04-11
* offset\_id: A unique UUID

#### **2.3. Load Data into Snowflake**

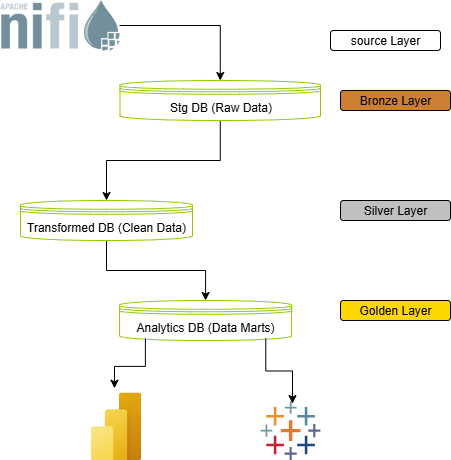
##### **Goal:**Load enriched records into a staging table in Snowflake.

##### **NiFi Processors Used:**

| **Processor** | **Purpose** |
| --- | --- |
| PutDatabaseRecord | Inserts records into Snowflake staging table |

**5. Snowflake Datawarehuse Implementation**

### **1.Snowflake structure**



### **1. Bronze Layer (Staging DB)**

* Raw ingestion from NiFi
* All columns (denormalized)
* No normalization
* Table: stg.raw\_data.orders

##### **Target Table: STG.RAW\_DATA.ORDERS**

****CREATE OR REPLACE TABLE STG.RAW\_DATA.ORDERS (

ROWID NUMBER(38,0),

ORDERID VARCHAR(16777216),

ORDERDATE VARCHAR(16777216),

SHIPDATE VARCHAR(16777216),

SHIPMODE VARCHAR(16777216),

CUSTOMERID VARCHAR(16777216),

CUSTOMERNAME VARCHAR(16777216),

SEGMENT VARCHAR(16777216),

COUNTRY VARCHAR(16777216),

CITY VARCHAR(16777216),

STATE VARCHAR(16777216),

POSTALCODE VARCHAR(16777216),

REGION VARCHAR(16777216),

PRODUCTID VARCHAR(16777216),

CATEGORY VARCHAR(16777216),

SUBCATEGORY VARCHAR(16777216),

PRODUCTNAME VARCHAR(16777216),

SALES NUMBER(18,4),

QUANTITY NUMBER(38,0),

DISCOUNT NUMBER(9,4),

PROFIT NUMBER(18,4),

TBL\_DT NUMBER(38,0),

INGESTED\_AT TIMESTAMP\_NTZ(9),

FILE\_NAME VARCHAR(16777216),

OFFSET\_ID VARCHAR(16777216)

);



### **2. Silver Layer (Transformed DB) — Dimension & Fact Tables**

Here is where you:

* Normalize the stg.orders data
* Transform the orders table
* Create & populate dimension and fact tables:

**Dimension Tables (in Silver Layer)**

* transformed.public.customers
* transformed.public.products
* transformed.public.geography

##### 

##### 

##### 

##### 

##### **Target Table: transformed.public.customers**

****create or replace TABLE TRANSFORMED.PUBLIC.CUSTOMERS (

CUSTOMERID VARCHAR(16777216),

CUSTOMERNAME VARCHAR(16777216),

SEGMENT VARCHAR(16777216)

);

##### **Target Table: transformed.public.products**

****create or replace TABLE TRANSFORMED.PUBLIC.PRODUCTS (

PRODUCTID VARCHAR(16777216),

CATEGORY VARCHAR(16777216),

SUBCATEGORY VARCHAR(16777216),

PRODUCTNAME VARCHAR(16777216)

);

##### **Target Table: TRANSFORMED.PUBLIC.GEOGRAPHY**

****create or replace TABLE TRANSFORMED.PUBLIC.GEOGRAPHY (

GEOGRAPHY\_ID NUMBER(38,0) NOT NULL DEFAULT TRANSFORMED.PUBLIC.GEOGRAPHY\_SEQ.NEXTVAL,

COUNTRY VARCHAR(16777216),

CITY VARCHAR(16777216),

STATE VARCHAR(16777216),

POSTALCODE VARCHAR(16777216),

REGION VARCHAR(16777216),

primary key (GEOGRAPHY\_ID));



#### **Fact Table (in Silver Layer)**

##### **Target Table: TRANSFORMED.PUBLIC.FACT\_ORDERS**

****create or replace TABLE TRANSFORMED.PUBLIC.FACT\_ORDERS (

ORDERID VARCHAR(16777216) NOT NULL,

ORDERDATE DATE,

SHIPDATE DATE,

SHIPMODE VARCHAR(16777216),

SALES NUMBER(18,4),

QUANTITY NUMBER(38,0),

DISCOUNT NUMBER(9,4),

PROFIT NUMBER(18,4),

CUSTOMERID VARCHAR(16777216),

PRODUCTID VARCHAR(16777216),

GEOID NUMBER(38,0),

primary key (ORDERID)

);

This is your **clean, structured, and normalized model** (3NF).

**3. Gold Layer (Analytics DB) — Final data marts**

* Join dimensions + facts
* Apply business logic, KPI calculation, aggregations

##### **Target Table: ANALYTICS.PUBLIC.DM\_BESTSELLERS\_PRICE\_DYNAMICS**

****create or replace TABLE ANALYTICS.PUBLIC.DM\_BESTSELLERS\_PRICE\_DYNAMICS (

MONTH DATE,

PRODUCTID VARCHAR(16777216),

PRODUCTNAME VARCHAR(16777216),

AVG\_PRICE NUMBER(36,10) ) ;

##### 

##### 

##### 

##### 

##### **Target Table: ANALYTICS.PUBLIC.DM\_SEGMENT\_TOP\_CUSTOMERS**

****create or replace TABLE ANALYTICS.PUBLIC.DM\_SEGMENT\_TOP\_CUSTOMERS (

SEGMENT VARCHAR(16777216),

CUSTOMERID VARCHAR(16777216),

CUSTOMERNAME VARCHAR(16777216),

TOTAL\_SALES NUMBER(30,4),

RANK\_IN\_SEGMENT NUMBER(18,0)

);

##### **Target Table: ANALYTICS.PUBLIC.DM\_STATE\_DELIVERY\_RANKS**

****create or replace TABLE ANALYTICS.PUBLIC.DM\_STATE\_DELIVERY\_RANKS (

MONTH DATE,

STATE VARCHAR(16777216),

PRODUCTID VARCHAR(16777216),

PRODUCTNAME VARCHAR(16777216),

AVG\_DELIVERY\_DAYS NUMBER(27,6),

FASTEST\_RANK NUMBER(18,0),

SLOWEST\_RANK NUMBER(18,0)

);

##### **Target Table: ANALYTICS.PUBLIC.DM\_STATE\_PRODUCT\_SALES\_RANKS**

****create or replace TABLE ANALYTICS.PUBLIC.DM\_STATE\_PRODUCT\_SALES\_RANKS (

MONTH DATE,

STATE VARCHAR(16777216),

PRODUCTID VARCHAR(16777216),

PRODUCTNAME VARCHAR(16777216),

TOTAL\_SALES NUMBER(30,4),

TOP\_RANK NUMBER(18,0),

BOTTOM\_RANK NUMBER(18,0)

);

##### 

##### 

##### 

##### 

##### **Target Table: ANALYTICS.PUBLIC.DM\_STATE\_SALES\_DYNAMICS**

****create or replace TABLE ANALYTICS.PUBLIC.DM\_STATE\_SALES\_DYNAMICS (

MONTH DATE,

STATE VARCHAR(16777216),

TOTAL\_SALES NUMBER(30,4)

);

## **6. Apache Airflow implementation**

**DAG Functionality:**

* Reads data from Snowflake **staging tables**.
* Extracts and loads **dimension tables**: Customers, Products, Geography.
* Populates **Fact\_Order** table.

**Orchestration:**

* Managed with **Apache Airflow DAG**.
* Scheduled execution: **monthly**, first day of each month.

## **7. Openmetadata integration**

**Setup:**

* Installed and configured **OpenMetadata**.
* Connected to **Snowflake Data Warehouse**.

**Metadata Ingestion:**

* Retrieved databases, schemas, and tables.
* Captured table structures and relationships.

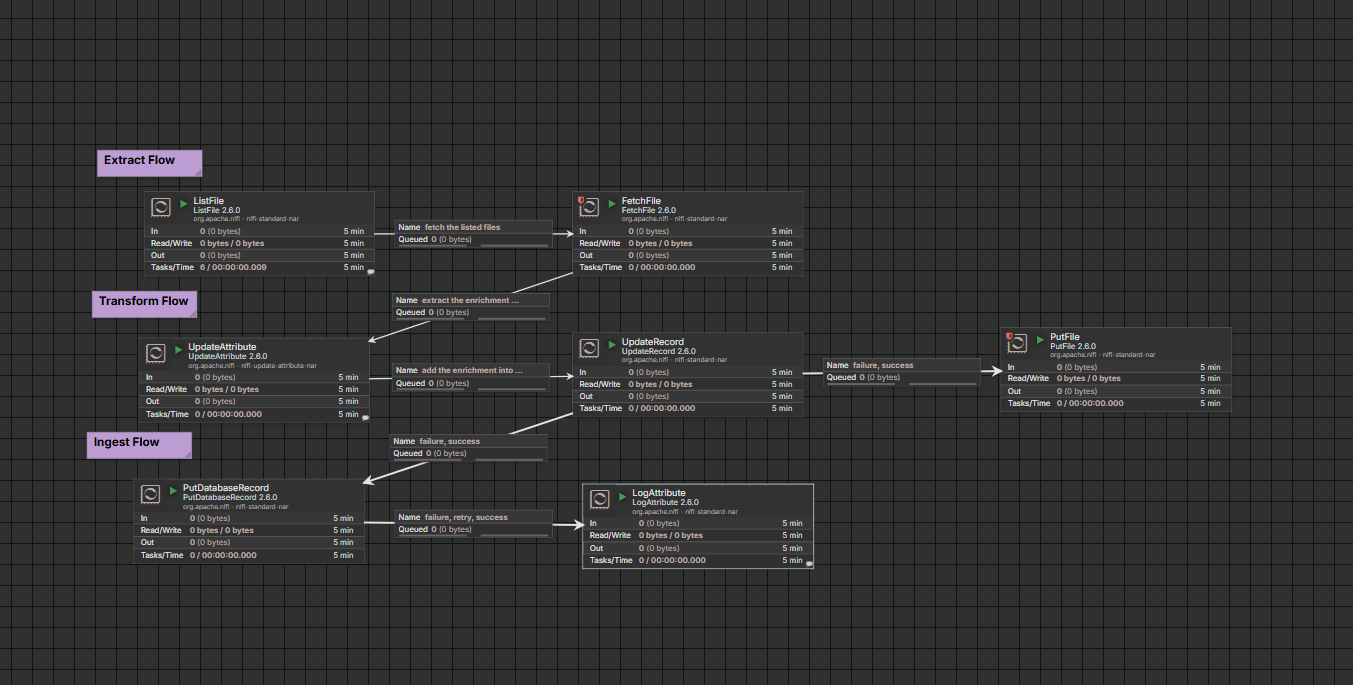
**Capabilities:**

* **Data Profiling:** Generates statistics for data quality assessment.
* **Documentation:** Add descriptions, criticality levels, and business context.
* **Lineage & Dependencies:** Visualize table and pipeline dependencies.
* **Data Quality Checks:** Define and monitor DQ rules across layers.

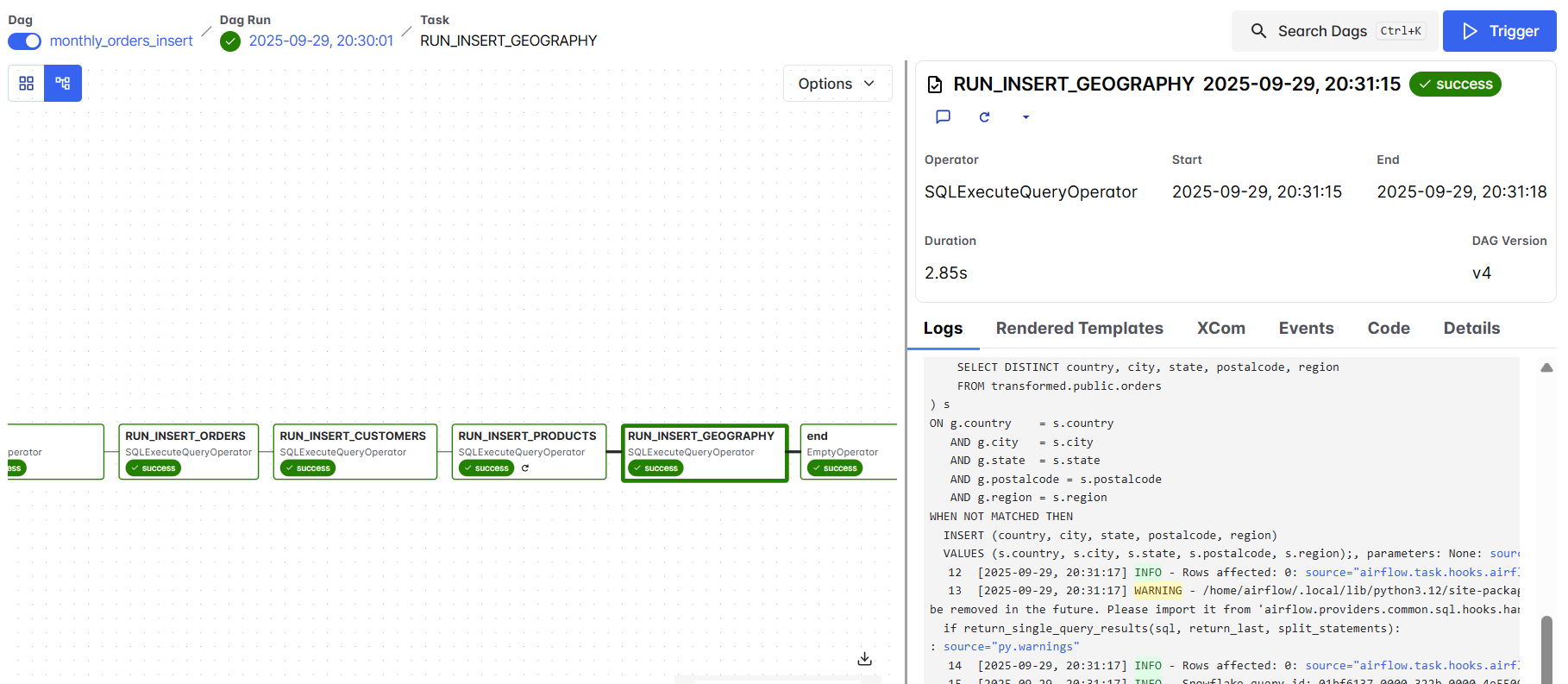
## 

## **Figures**

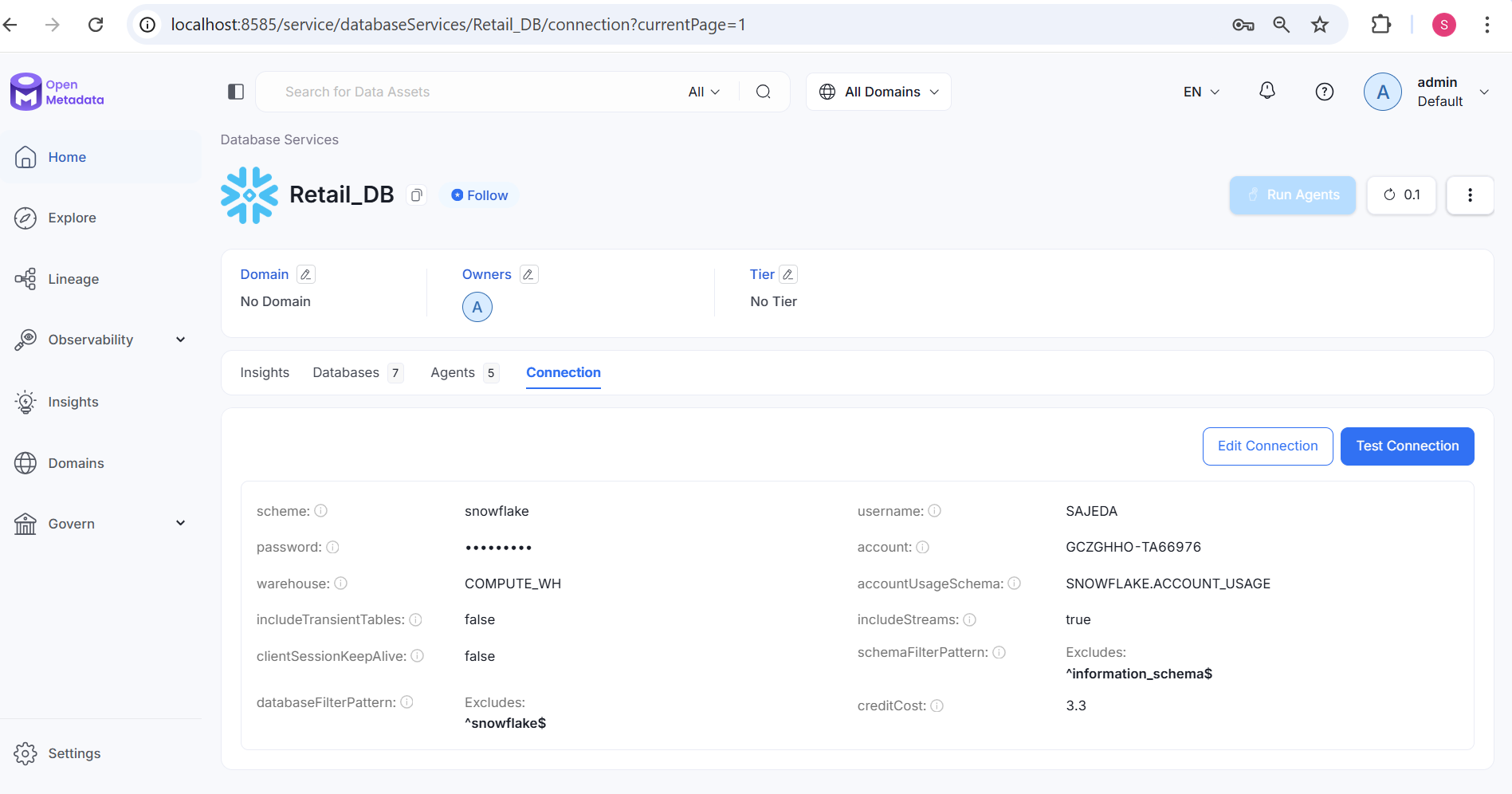
*Figure 1:*  – NiFi Flow: Ingest & Archive CSV Files



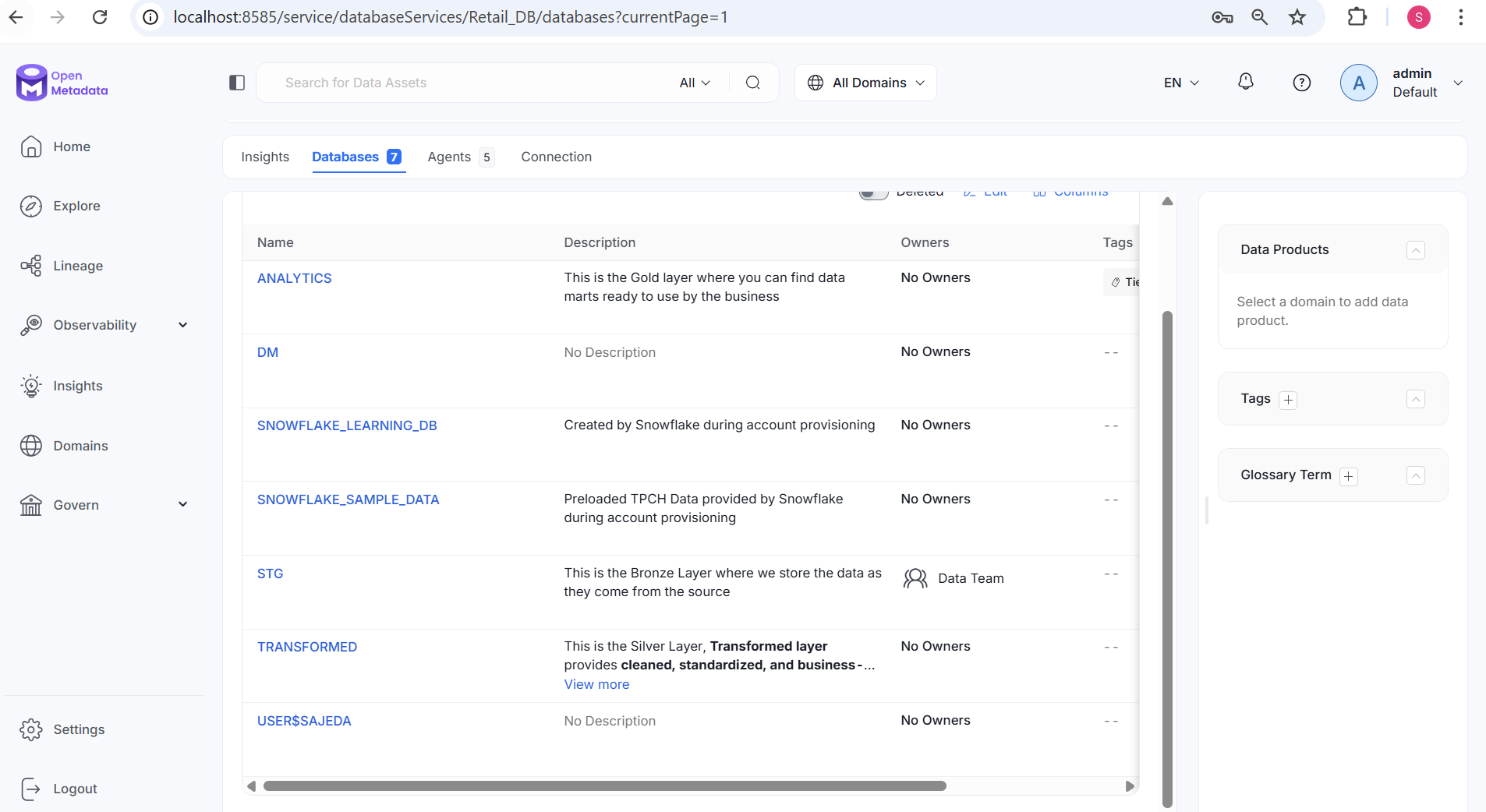
*Figure 2:*  – Apache Airflow dag



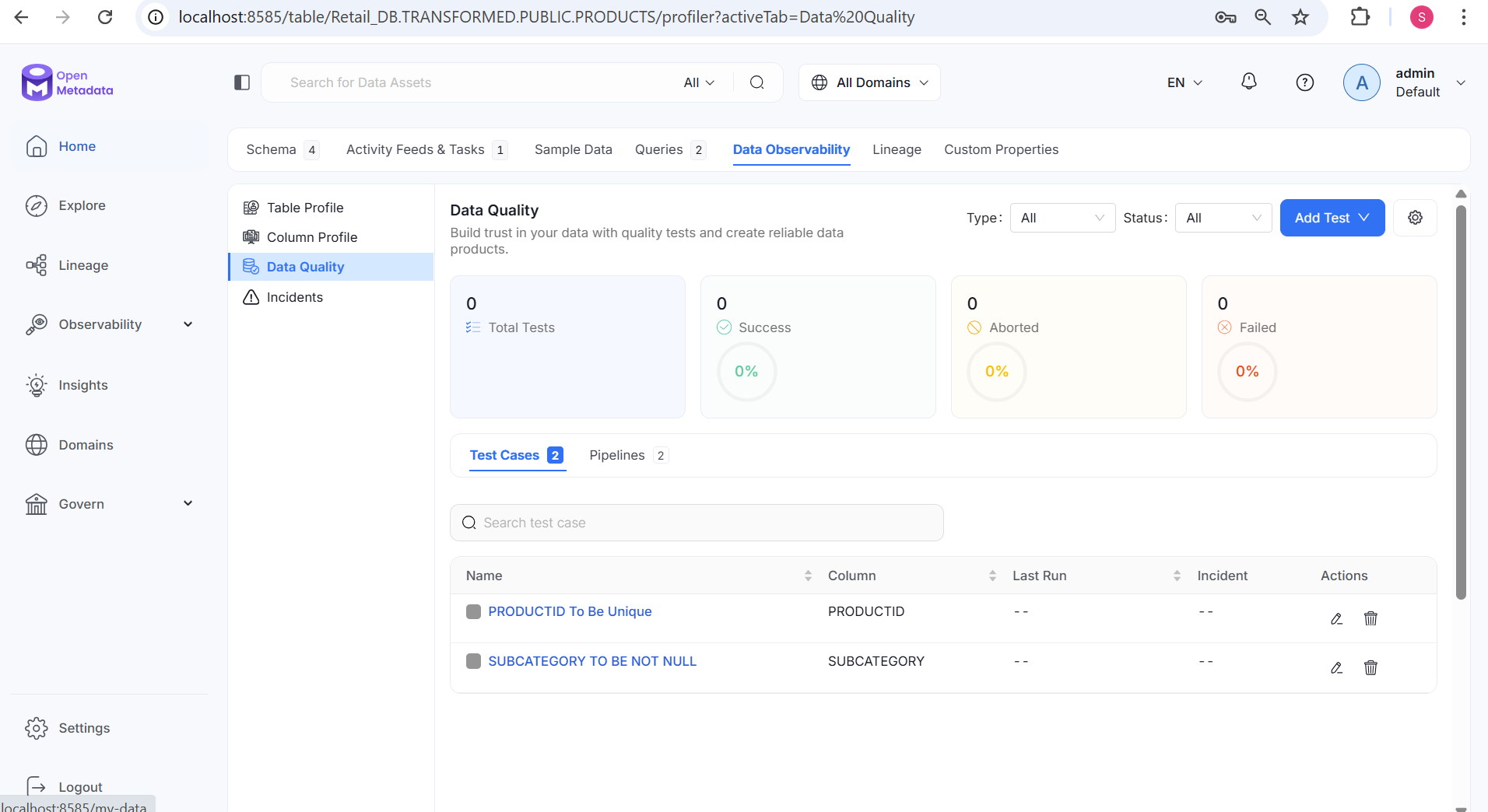
*Figure 3:*  – Openmetadata service integration



*Figure 4:* – Openmetadata Databases



*Figure 5:* – Openmetadata test case example



*Figure 6:* – Openmetadata lineage example

