**E-commerce Data Platform - Architecture & Workflow**

Project Path: /media/softsuave/DATA-HDD/DataEngineering/Apache\_Airflow

# 1. Overview

This document describes the folder structure and the end-to-end data workflow for the e-commerce data platform. The system generates synthetic data, ingests it into SQL Server, processes data via a PySpark ETL pipeline, and orchestrates schedules with Apache Airflow.

# 2. Folder Structure

* dags/: Airflow DAG definitions (e.g., ecom\_schedules.py).
* src/: Application source code (ETL and synthetic data generators).
* - ecom\_etl\_pipeline.py: PySpark ETL pipeline.
* - synth\_ecom\_mssql\_stream.py: Synthetic data generator streaming to SQL Server.
* scripts/: Executable scripts for running ETL and synthetic streams.
* - etl/run\_etl.sh, etl/run\_etl.bat: Submit ETL job with Spark.
* - synth/run\_synth\_stream.sh, synth/run\_synth\_stream\_config.sh, synth/run\_synth\_stream.bat: Run synthetic stream.
* libs/jars/: External libraries (e.g., mssql JDBC driver).
* logs/: Log output (Spark/Airflow logs as configured).
* data/: Optional local data or exports.
* airflow\_venv/: Python virtual environment used by scripts (if present).

# 3. Technology Stack

* SQL Server: Transactional and analytics databases (ecom\_db, ecom\_dwh).
* PySpark: Data processing, cleansing, aggregations, analytics.
* Apache Airflow: Orchestration and scheduling.
* Python (Faker, SQLAlchemy/pyodbc): Synthetic data generation and connectivity.

# 4. End-to-End Data Flow

* Synthetic Generation: scripts/synth/run\_synth\_stream.sh executes src/synth\_ecom\_mssql\_stream.py to insert new customers, products, orders, order\_items, inventory, and clickstream events into SQL Server (ecom\_db).
* Scheduling: Airflow DAG synth\_stream\_hourly triggers the synthetic stream every hour to simulate continuous activity.
* ETL Extraction: src/ecom\_etl\_pipeline.py reads raw tables from source DB (ecom\_db) via JDBC.
* Data Cleaning: Customers, products, orders, order\_items, inventory, and clickstream are validated and standardized.
* Feature Views: Customer 360 and Product Analytics views are created for downstream analytics.
* Loading: Cleaned and analytical tables are written to the target DB (ecom\_dwh) using JDBC.
* Data Quality: Metrics are recorded into etl\_data\_quality\_log for monitoring.

# 5. Scheduling with Airflow

Defined in dags/ecom\_schedules.py:

* synth\_stream\_hourly: Runs scripts/synth/run\_synth\_stream.sh at the top of every hour (cron: 0 \* \* \* \*).
* ecom\_etl\_every\_2\_hours: Runs scripts/etl/run\_etl.sh every two hours (cron: 0 \*/2 \* \* \*).

# 6. Scripts & Parameters

Synthetic Stream (Linux): scripts/synth/run\_synth\_stream.sh

* DB\_URL: SQLAlchemy URL for SQL Server (sa / Gova#ss123).
* WEBHOOK\_URL: Optional webhook for notifications.
* INTERVAL\_MINUTES: Batch interval for synthetic inserts (default 15).
* ORDERS\_PER\_INTERVAL: Orders generated per interval.
* CLICKS\_PER\_INTERVAL: Click events generated per interval.

ETL Runner (Linux): scripts/etl/run\_etl.sh

* Resolves src/ and libs/jars/ paths dynamically.
* Builds JDBC URLs for source (ecom\_db) and target (ecom\_dwh).
* Submits the Spark job with performance-focused configs.

# 7. Databases & Tables

Source: ecom\_db (raw transactional data). Target: ecom\_dwh (modeled analytics).

* Source tables: customers, products, orders, order\_items, inventory, clickstream.
* Target tables: dim\_customers, dim\_products, fact\_orders, fact\_order\_items, dim\_inventory, fact\_clickstream, analytics\_customer\_360, analytics\_product\_performance, etl\_data\_quality\_log.

# 8. Operational Notes

* Ensure libs/jars/mssql-jdbc-\*.jar exists; update versions if needed.
* Set AIRFLOW\_HOME to the project root before running Airflow services.
* Virtual environment activation is handled in scripts if airflow\_venv exists.
* Array columns are flattened/cast before JDBC writes to avoid type errors.

# 9. Quickstart Commands

* Run synthetic stream (Linux): bash scripts/synth/run\_synth\_stream.sh
* Run ETL (Linux): bash scripts/etl/run\_etl.sh
* Start Airflow: export AIRFLOW\_HOME=... && airflow db init && airflow webserver -p 8080 & && airflow scheduler &