ShopVerse Daily ETL Pipeline – Apache Airflow Project

Table of Contents

1. Project Overview

This project implements an end-to-end daily batch data pipeline for the e-commerce company ShopVerse. The pipeline ingests customer, product, and order datasets daily using Apache Airflow, loads them into a PostgreSQL data warehouse, validates data, handles transformations, and applies quality checks.

2. Folder & Environment Structure

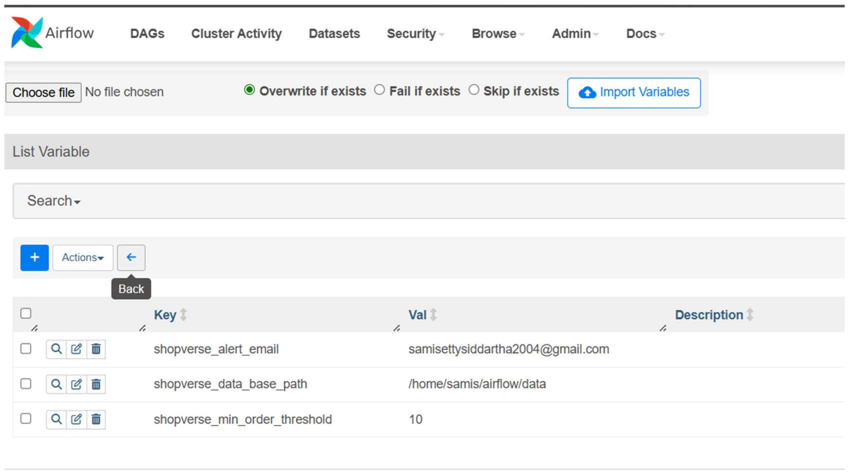
The following directory structure is used to organize Airflow DAGs and datasets:

```
~/airflow/
  dags/
    shopverse_daily_pipeline.py
  data/
    landing/
      customers/
      products/
      orders/
    anomalies/
```

3. Airflow Variables

Before running the pipeline, configure these Airflow Variables:

- shopverse_data_base_path = /home/samis/airflow/data
- shopverse_min_order_threshold = 10
- shopverse_alert_email = samisettysiddartha2004@gmail.com



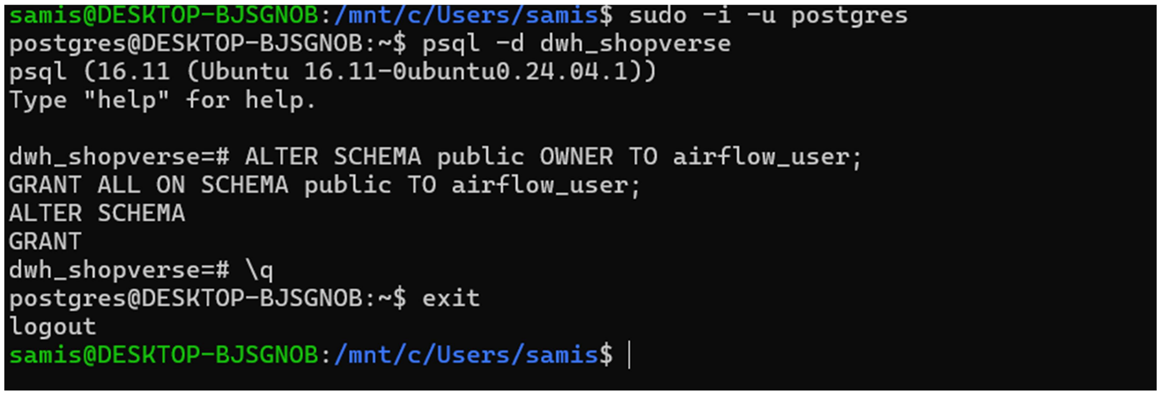
No file chosen

[✓ Overwrite if exists] Fail if exists Skip if exists

Key	Val	Description
shopverse_alert_email	samisettysiddartha2004@gmail.com	
shopverse_data_base_path	/home/samis/airflow/data	
shopverse_min_order_threshold	10	

4. Airflow Connections

Create a Postgres connection with the following details:
Connection ID: postgres_dwh
Type: Postgres
Host: localhost
Database: dwh_shopverse
User: airflow_user
Password: root
Port: 5432

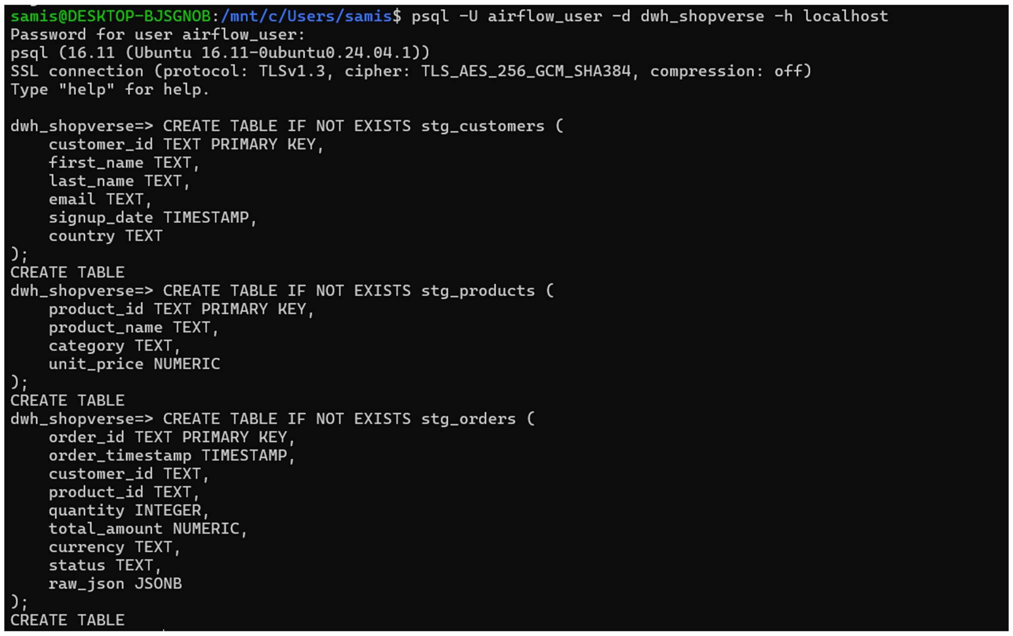


```
samis@DESKTOP-BJSGNOB:/mnt/c/Users/samis$ sudo -i -u postgres
postgres@DESKTOP-BJSGNOB:~$ psql -d dwh_shopverse
psql (16.11 (Ubuntu 16.11-0ubuntu0.24.04.1))
Type "help" for help.

dwh_shopverse=# ALTER SCHEMA public OWNER TO airflow_user;
GRANT ALL ON SCHEMA public TO airflow_user;
ALTER SCHEMA
GRANT
dwh_shopverse=# \q
postgres@DESKTOP-BJSGNOB:~$ exit
logout
samis@DESKTOP-BJSGNOB:/mnt/c/Users/samis$ |
```

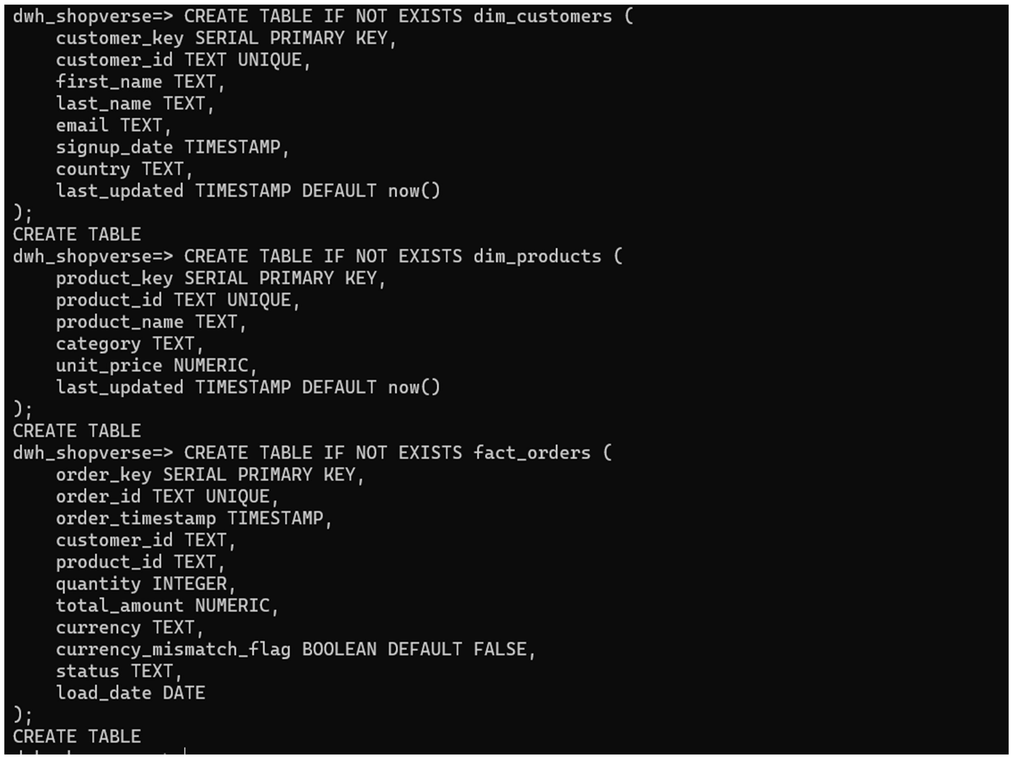
5. Database Tables (DDL)

The following tables must exist before running the pipeline:
Staging: stg_customers, stg_products, stg_orders
Warehouse: dim_customers, dim_products, fact_orders



```
samis@DESKTOP-BJSGNOB:/mnt/c/Users/samis$ psql -U airflow_user -d dwh_shopverse -h localhost
Password for user airflow_user:
psql (16.11 (Ubuntu 16.11-0ubuntu0.24.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

dwh_shopverse=> CREATE TABLE IF NOT EXISTS stg_customers (
    customer_id TEXT PRIMARY KEY,
    first_name TEXT,
    last_name TEXT,
    email TEXT,
    signup_date TIMESTAMP,
    country TEXT
);
CREATE TABLE
dwh_shopverse=> CREATE TABLE IF NOT EXISTS stg_products (
    product_id TEXT PRIMARY KEY,
    product_name TEXT,
    category TEXT,
    unit_price NUMERIC
);
CREATE TABLE
dwh_shopverse=> CREATE TABLE IF NOT EXISTS stg_orders (
    order_id TEXT PRIMARY KEY,
    order_timestamp TIMESTAMP,
    customer_id TEXT,
    product_id TEXT,
    quantity INTEGER,
    total_amount NUMERIC,
    currency TEXT,
    status TEXT,
    raw_json JSONB
);
CREATE TABLE
```



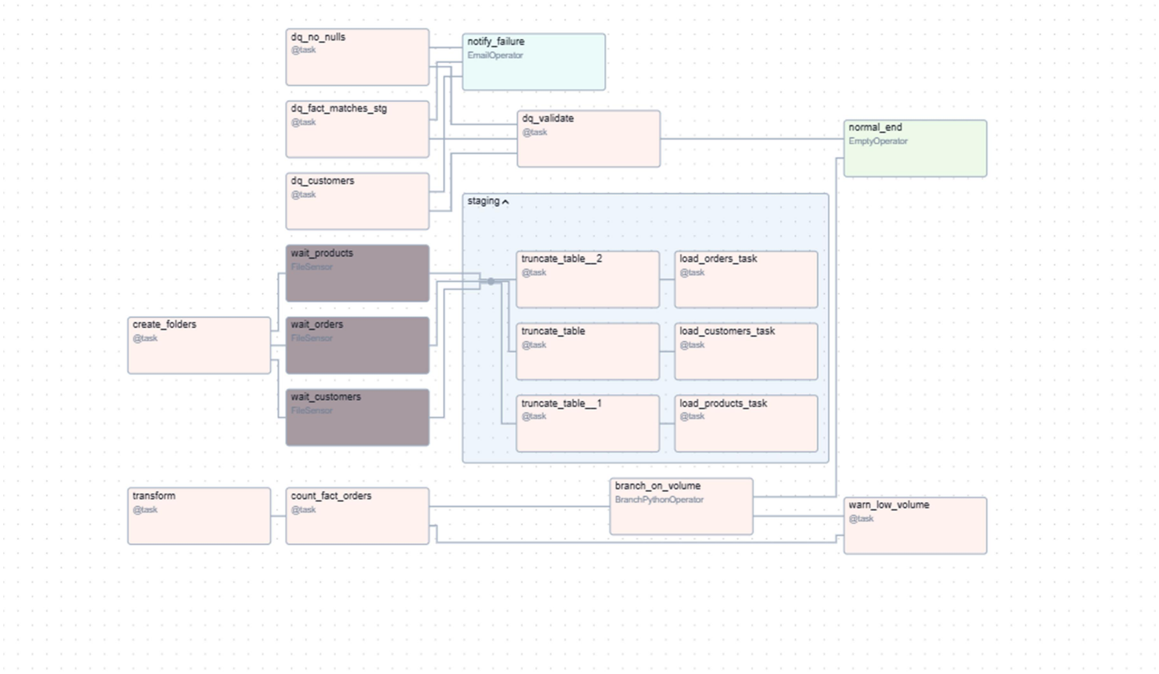
```
dwh_shopverse=> CREATE TABLE IF NOT EXISTS dim_customers (
    customer_key SERIAL PRIMARY KEY,
    customer_id TEXT UNIQUE,
    first_name TEXT,
    last_name TEXT,
    email TEXT,
    signup_date TIMESTAMP,
    country TEXT,
    last_updated TIMESTAMP DEFAULT now()
);
CREATE TABLE
dwh_shopverse=> CREATE TABLE IF NOT EXISTS dim_products (
    product_key SERIAL PRIMARY KEY,
    product_id TEXT UNIQUE,
    product_name TEXT,
    category TEXT,
    unit_price NUMERIC,
    last_updated TIMESTAMP DEFAULT now()
);
CREATE TABLE
dwh_shopverse=> CREATE TABLE IF NOT EXISTS fact_orders (
    order_key SERIAL PRIMARY KEY,
    order_id TEXT UNIQUE,
    order_timestamp TIMESTAMP,
    customer_id TEXT,
    product_id TEXT,
    quantity INTEGER,
    total_amount NUMERIC,
    currency TEXT,
    currency_mismatch_flag BOOLEAN DEFAULT FALSE,
    status TEXT,
    load_date DATE
);
CREATE TABLE
```

6. DAG Overview

The Airflow DAG 'shopverse_daily_pipeline' runs daily at 01:00 and processes data for the previous day. It uses TaskFlow API, TaskGroups, FileSensors, branching, and data quality checks.

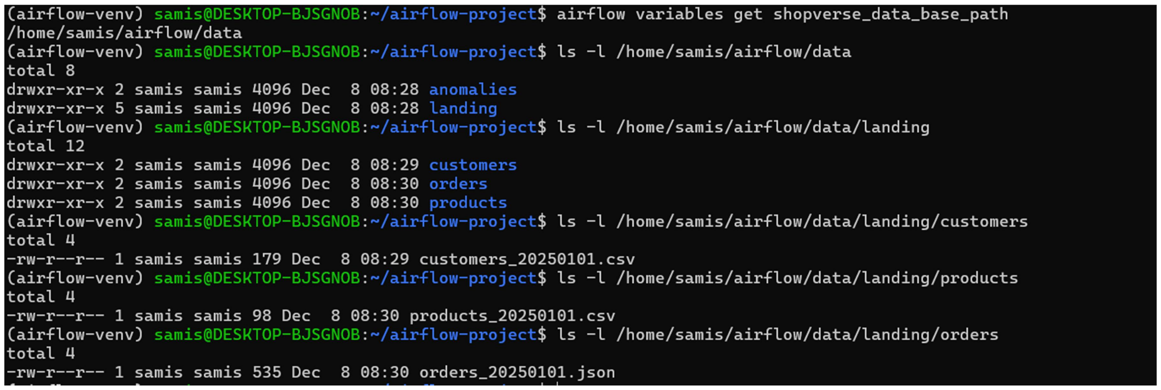
7. Pipeline Workflow

- 1. Wait for files using FileSensor.
- 2. Stage datasets into Postgres (truncate + load).
- 3. Transform and load warehouse tables.
- 4. Apply data quality checks.
- 5. Branch if order volume is below threshold.
- 6. Send alert email if failures occur.



8. How to Place Input Data

Files must follow the naming pattern customers_YYYYMMDD.csv, products_YYYYMMDD.csv, and orders_YYYYMMDD.json in the following directories:
/home/samis/airflow/data/landing/customers/
/home/samis/airflow/data/landing/products/
/home/samis/airflow/data/landing/orders/



```
(airflow-venv) samis@DESKTOP-BJSGNOB:~/airflow-project$ airflow variables get shopverse_data_base_path
/home/samis/airflow/data
(airflow-venv) samis@DESKTOP-BJSGNOB:~/airflow-project$ ls -l /home/samis/airflow/data
total 8
drwxr-xr-x 2 samis samis 4096 Dec  8 08:28 anomalies
drwxr-xr-x 5 samis samis 4096 Dec  8 08:28 landing
(airflow-venv) samis@DESKTOP-BJSGNOB:~/airflow-project$ ls -l /home/samis/airflow/data/landing
total 12
drwxr-xr-x 2 samis samis 4096 Dec  8 08:29 customers
drwxr-xr-x 2 samis samis 4096 Dec  8 08:30 orders
drwxr-xr-x 2 samis samis 4096 Dec  8 08:30 products
(airflow-venv) samis@DESKTOP-BJSGNOB:~/airflow-project$ ls -l /home/samis/airflow/data/landing/customers
total 4
-rw-r--r-- 1 samis samis 179 Dec  8 08:29 customers_20250101.csv
(airflow-venv) samis@DESKTOP-BJSGNOB:~/airflow-project$ ls -l /home/samis/airflow/data/landing/products
total 4
-rw-r--r-- 1 samis samis 98 Dec  8 08:30 products_20250101.csv
(airflow-venv) samis@DESKTOP-BJSGNOB:~/airflow-project$ ls -l /home/samis/airflow/data/landing/orders
total 4
-rw-r--r-- 1 samis samis 535 Dec  8 08:30 orders_20250101.json
```

9. How to Trigger the DAG

Manual trigger:
airflow dags trigger shopverse_daily_pipeline --execution-date 2025-01-01

Backfill:
airflow dags backfill shopverse_daily_pipeline -s 2025-01-01 -e 2025-01-05

10. Testing Sequence

1. Place test files for a specific date.
2. Trigger the DAG and confirm sensors detect files.
3. Verify staging tables populate correctly.
4. Confirm warehouse tables update.
5. Validate that data quality checks pass.
6. Test branching behavior with low-volume data.
7. Perform a backfill test.

11. Summary

This Airflow project implements a complete daily ETL workflow, including ingestion, staging, transformation, data quality enforcement, anomaly detection, and notifications. The DAG meets all assignment requirements and is built using best practices for modularity and maintainability.