Retail Sales ETL (Airflow → Spark → Postgres)

**Project Overview**

This project contains a small, production style pipeline that ingests CSV batches of retail transactions and returns, cleans and transforms them with PySpark, and loads curated tables into Postgres. The pipeline is orchestrated by Apache Airflow and packaged with Docker Compose so it runs locally with one command.

**Highlights**

* Containerized Airflow (webserver, scheduler, worker), Postgres, and Redis.
* A single daily DAG that picks up the next CSV batch, processes it with Spark, loads to Postgres, and archives the file.
* Separate staging vs clean tables for traceability.

**Architecture**

* Airflow 2.9 - orchestration
* Spark 3.5 - transform
* Postgres 14 - warehouse
* Redis – celery broker

**Flow:**

A diagram of a sparking process

AI-generated content may be incorrect.

**Data Model**

Two subject areas, each with **staging** and **clean**:

**retail.sales\_clean**

* order\_id (PK from hash of invoice + sku or source key)
* invoiceno (e.g., INV5005)
* stockcode (SKU…)
* description
* quantity (positive)
* invoicedatets (timestamp)
* unitprice (decimal)
* customerid (int)
* country (text)
* **Derived**:
  + totalprice = quantity \* unitprice
  + month = date\_trunc('month', invoicedatets)::date

**retail.returns\_clean**

* return\_id (PK)
* invoiceno, stockcode, description
* quantity (negative)
* invoicedatets, unitprice, customerid, country
* totalprice, month (same derivations)

**Staging tables** mirror the clean schema and hold a single batch before upsert.

**Repo Layout:**

├─ dags/

│ └─ retail\_project/

│ ├─ RetailToPostgres.py # PySpark job

│ └─ dag\_retail\_etl\_daily.py # Airflow DAG (DockerOperator)

├─ data/ # drop CSV batches here

├─ archive/ # processed files moved here

├─ docker-compose.yml

└─ README.md

**Running Locally**

**Prereqs**

* **Docker Desktop (or Docker Engine) + Compose**
* **~4 GB RAM available for containers**

**Steps**

1. Start the stack
   1. **docker compose up -d –build**
2. Drop a CSV and trigger the DAG
   1. Place a file like data/retail\_batch\_2025\_03\_01.csv. Then in Airflow trigger the DAG **retail\_etl\_daily**
3. Verify in Postgres.

**What Airflow DAG does.**

Airflow DAG (retail\_etl\_dag.py):  
The Airflow DAG orchestrates the ETL workflow for retail sales data. It automates the process of loading CSV batches, running Spark jobs to transform and clean the data, and upserting the results into a PostgreSQL database. The DAG ensures that each batch is processed in sequence, manages dependencies, and provides monitoring and scheduling capabilities for the entire ETL pipeline.

RetailToPostgres (RetailToPostgres.py):  
This script performs the ETL logic for a single retail sales CSV file. It loads the data using Spark, applies transformations (such as timestamp parsing, total price calculation, and filtering), generates stable IDs, and normalizes column names. The cleaned sales and returns data are then upserted into PostgreSQL tables, ensuring data integrity and avoiding duplicates. The script is designed to be run as part of the Airflow DAG for batch processing.

**The Spark Job**

processes a retail sales CSV file by performing the following steps:

* Loads the raw data into a Spark DataFrame using a predefined schema.
* Transforms the data by parsing timestamps, calculating total prices, and filtering out invalid records.
* Separates sales and returns based on quantity.
* Generates stable, hashed IDs for each record to ensure uniqueness.
* Normalizes column names to lowercase for compatibility with PostgreSQL.
* Writes the cleaned sales and returns data to staging tables in PostgreSQL, then upserts them into the final target tables, avoiding duplicates.