Project Documentation: ETL Pipeline Setup, Data Pipeline Architecture, and Code Modifications

Data Genius

Contents

1 Overview

This assignment aims to build the ETL pipeline.

- The pipeline would process the raw csv data file and transform it into structured data.
- Load the data into the data warehouse for further analysis using Apache Superset.
- Project involves generating environment, ETL data and ensure data-quality/ integrity through testing.

2 Setup Process

2.1 Environment Setup

- Install Docker on the system by following the official Docker installation instructions.
- Navigate to the root folder where the `docker-compose.yml` file is located.
- Initiate the build and launch the necessary containers using the following command:

```
docker compose up --build
```

2.2 Verify Data Location

- Verify the data are correctly located at the import/csv folder
- Ensure all the configurations are properly set.

2.3 ETL Orchestration with Airflow

- Access the Airflow GUI at `http://localhost:8080`
- Login using default credentials `user: airflow`, `password: airflow`.
- Trigger `initialize_etl_environment` to create the necessary objects in PostgreSQL.
- Trigger `import_data` to import the data from CSV files inside the `import/csv` folder.

2.4 Data Transformation

- Trigger `run_dbt` to transform the original data and load the data warehouse.

2.5 Data Analysis

- Access the superset GUI at `http://localhost:8088`.
- Login using default credentials: `user: admin`, `password: admin`.
- Create visualization for data analysis purposes using the data stored in the data warehouse.

2.6 Challenges Faced

- Table Relationship Issues
  - While designing tables' relationships it was challenging because some of the attributes are relate to each other.
  - The attributes : `age`, `anchor_age`, and `anchor_year` relationship's necessary calculations were missing.
- Table Configuration Mismatching
  - Some of the table names/definitions typo caused some problems.
- Airflow Task Dependencies
  - While modifying the dags, some of the task dependencies were misconfigured and caused severe errors.
- Database Connection Timeout
  - We faced difficulty of task timeout issue, which is solved by optimizing the dag configuration and increasing the timeout limit.

3 Data Pipeline Architecture

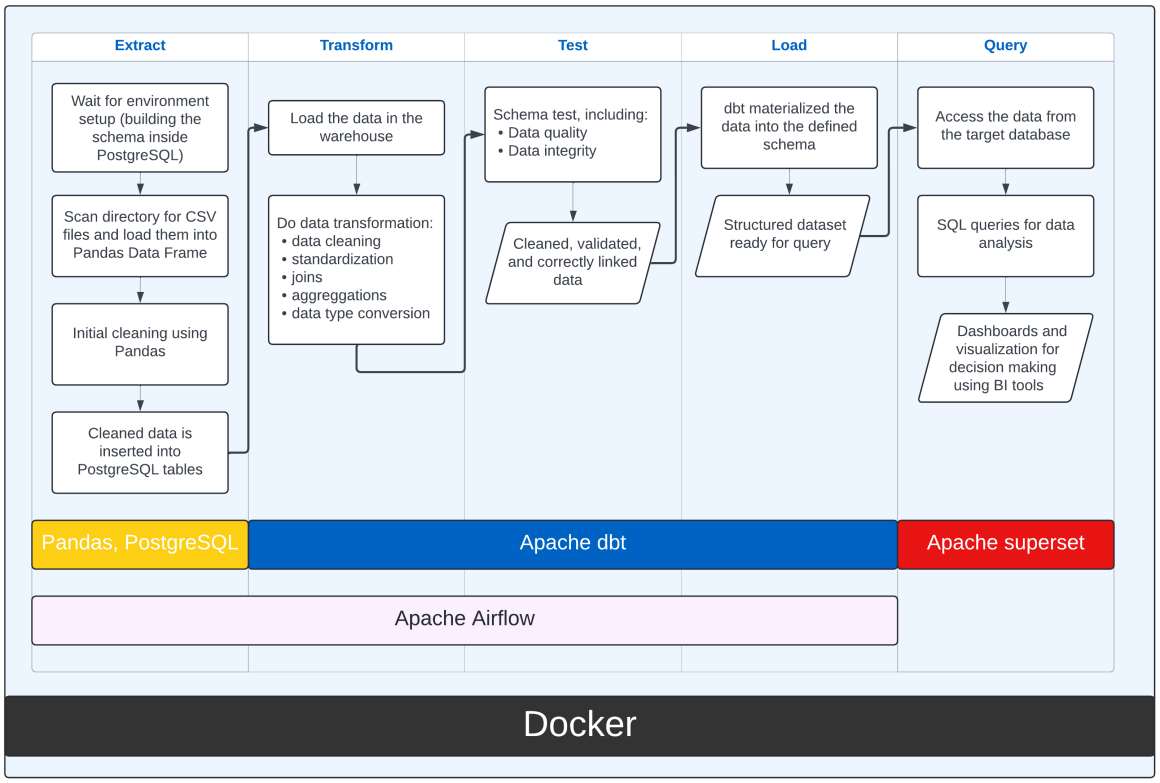

Figure 1: Data Pipeline Architecture

3.1 Extract: Pandas and PostgreSQL

Pandas and PostgreSQL are used to extract the data because its ease-of-use and simple syntax. Our dataset is small and structured, making it perfect to use these tools to run locally. Furthermore, SQL is the industry standard for database querying, making PostgreSQL compatible with other tools.

3.2 Transform, Test, and Load: dbt

dbt is used as a transformation-focused tool that operates directly in our database using SQL with seamless-integration with PostgreSQL. Which benefit for Automation-data-quality checks every time the pipeline executes, ensuring data quality.

3.3 Orchestrate: Apache Airflow

These Python based ETL process was orchestrated using Apache airflow, allowing us to define workflows as DAGs, making it easy to schedule, monitor, and manage dependencies between tasks, with highly flexibility for customization.

3.4 Query: Apache Superset

Superset is used as it can connect directly to our PostgreSQL database without needing to do data moving. This direct connection helps minimize latency and keeps our pipeline efficient.

All of the tools we use are open source and free to use, so we can use them without any licensing fees. Docker allows us to run everything on local servers, saving significant costs.

4 Airflow DAGs

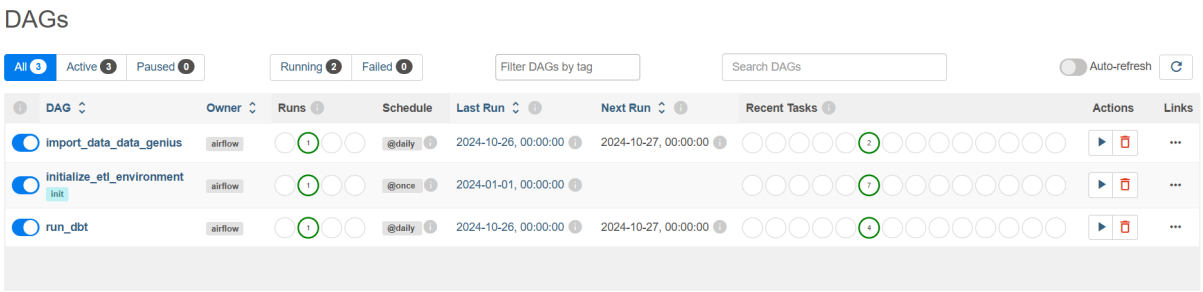

Figure 2: Airflow DAG GUI

4.1 initialize_etl_environment

This task is done to initialize the schema and tables inside PostgreSQL. It includes:

- Creating empty schema for import, warehouse, ops, and staging.
- Creating a registry table inside the ops schema.
- Creating patients, hospital admissions, ed visits, ed hospital admissions, and ecg recordings table within the import schema.

4.2 import_data

`import_data` is used to import the raw data from CSV files stored in the import/csv folder and do initial data preparations.

- Ensuring the data types of columns match the PostgreSQL table definitions.
- Converting the Pandas' array format into a format supported by PostgreSQL TEXT[].
- Replacing all pd.NA values with None to align with PostgreSQL's handling of null values.
- Splitting the data into multiple DataFrames corresponding to the PostgreSQL tables and removing any duplicate rows, then inserting it into the database.
- Record the name of the processed CSV file in the registry table for easy tracking of processed files.

4.3 run_dbt

`run_dbt` executes the dbt pipeline:

- Extracts data from the original source and stages it.
- Creates dimension tables and combines them into the final fact table.
- Ensures data integrity through the dbt test process.

5 Code Modifications

Changes	Description
Root directory structure	Delete generator and seed folder.
Airflow DAGs	Delete DAG for xml data loading, now only left with 3 DAGs.
Initial data processing	Some changes needed to convert data type from pandas to SQL and deduplication.
Data testing	Added a schema.yml file inside the dbt folder to check data integrity of our final dimension table during the dbt test, and ops folder to check the unnested data.
dbt_project.yml	Modify dbt/dbt comp5339/dbt project.yml to store data in the correct schemas: models in 'staging' go to the 'staging' schema, models in 'dim' go to the 'warehouse' schema, and models in 'ops' go to the 'ops' schema.

Figure 3: Code Modifications

6 dbt Models

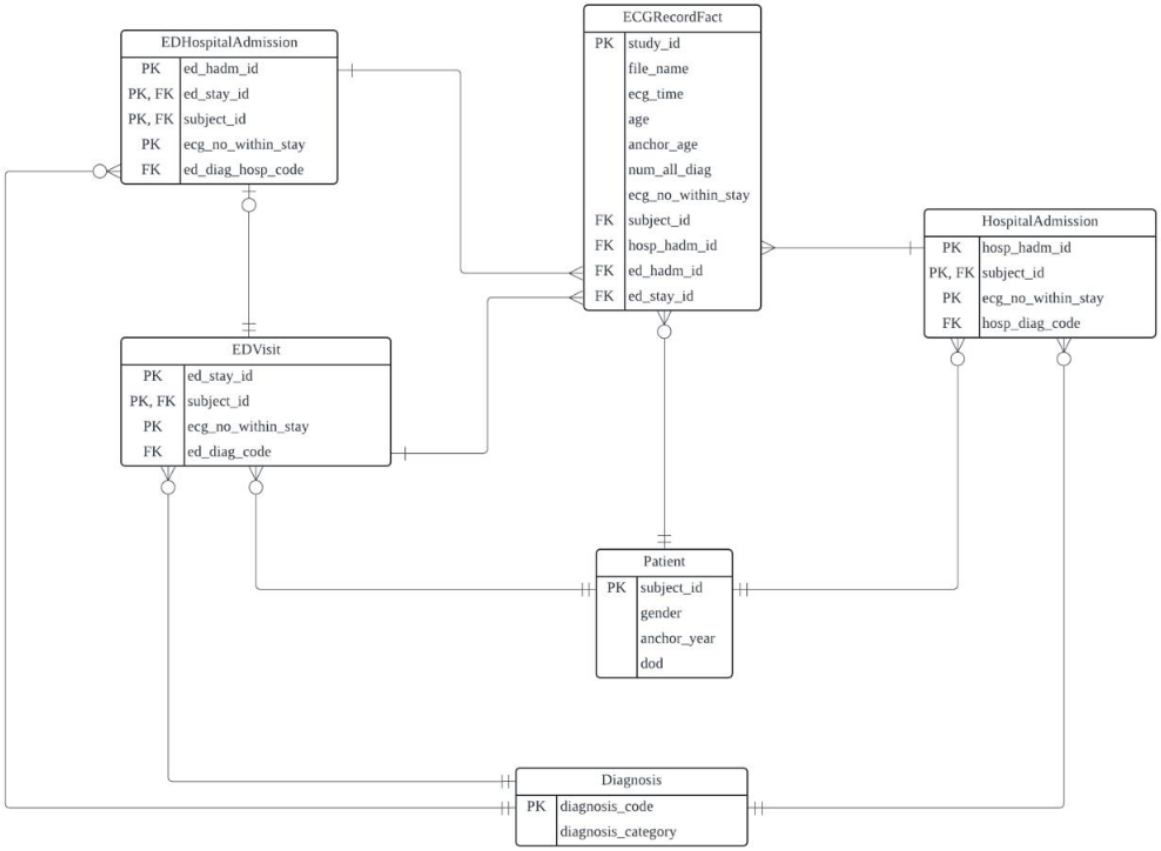

Figure 4: ERD

```
models
+---dim
        dim_diagnosis.sql
        dim_ed_hospital_admissions.sql
        dim_ed_visits.sql
        dim_hospital_admissions.sql
        dim_patients.sql
+---fact
        fact_ecg.sql
+---ops
        data_validation.sql
+---src
        src_ecg_recordings.sql
        src_ed_hospital_admissions.sql
        src_ed_visits.sql
        src_hospital_admissions.sql
        src_patients.sql
+---staging
        staging_diagnosis.sql
        staging_ecg_recordings.sql
        staging_ed_hospital_admissions.sql
        staging_ed_visits.sql
        staging_hospital_admissions.sql
        staging_patients.sql
```

Figure 5: dbt Directory

- src/ folder is used to draw the data from the data warehouse as specified inside the sources.yml.
- staging/ folder handles data preprocessing. Including:
  - Unnesting the array data of diagnosis results from the ed_hospital_admissions, ed_visits, and hospital_admissions tables.
  - Standardizing the gender values in the patients table by converting them to uppercase.
  - Filling missing values in the age, anchor_age columns with -1, and creating new column num_all_diags in the ecg_recordings table to count the total number of final-diagnoses.
  - Creating a new diagnosis table for more detailed information. Diagnosis codes mapped to broader categories (e.g., nervous-system, blood-disorders) based on the ICD-10-CM standard [1].
- dim/ folder takes the staged data from the staging folder and selects the columns needed. This is the final form of our tables.
- fact/ folder combines all the dimension tables into one fact table, linking the data from different sources.
- ops/ folder is used for validation and debugging. Performing checks for data integrity by verifying the number of diagnosis codes in the original arrays matches the number of unnested entries.

7 Dashboard

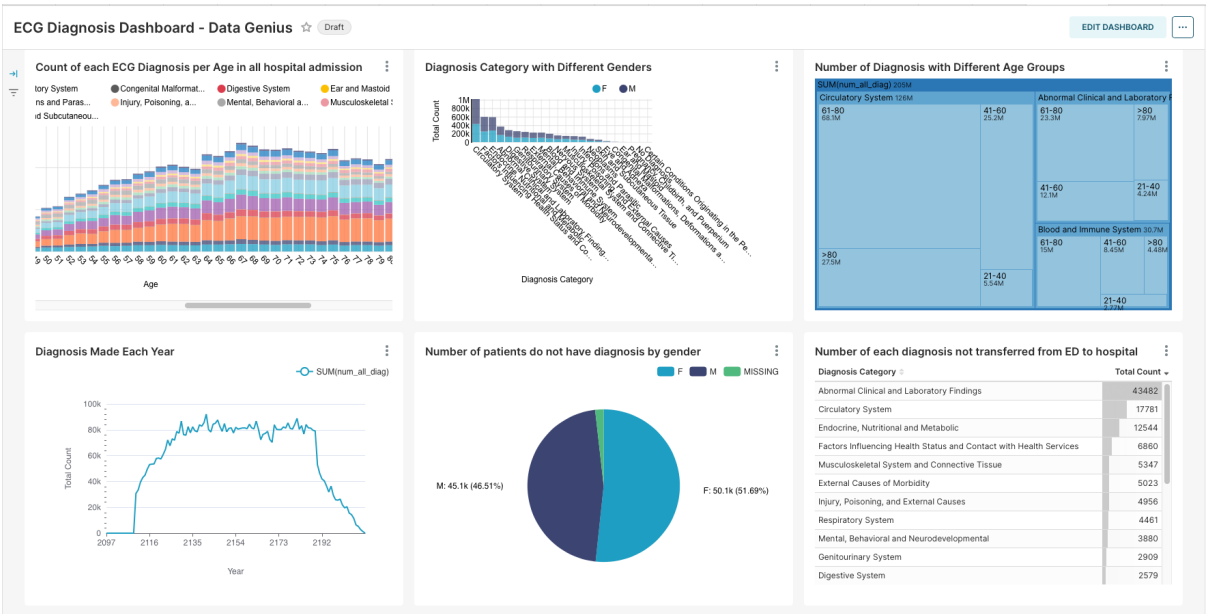

Figure 6: ECG Diagnosis Dashboard

The data visualization dashboard illustrates various aspects of ECG diagnoses across different attributes.

- The first chart shows the distribution of ECG diagnoses by age, with a notably larger amount among older populations.
- The second pie chart highlights the gender breakdown of patients lacking a diagnosis, showing a higher percentage of females.
- The third table presents diagnosis counts that were not transferred to the hospital, with "Abnormal Clinical and Laboratory Findings" being the most frequent category.
- The fourth line chart tracks annual diagnosis trends against anchor years, showing an increase followed by a recent decline.
- The fifth bar chart depicts diagnosis counts by gender and category, revealing the highest counts in the circulatory system.
- In the sixth chart, the treemap visualizes diagnosis volume by category and age, showing that patients between 61-80 took more diagnoses in every category.

8 Limitations and Improvements

Issue	**Proposed Improvement**
Pipeline only accepts CSV files	Expanding the file format support and optimize extra schema validation to ensure compatibility
Difficulty managing increasing schema	Deploy dynamic schema tool (Alembic) to provide smoother data base migrations without interruptions
Lack of errors monitoring	Deploying logging and monitoring application (ELK Stack) for tracking logs/ runtime issues/ assisting debugging

Table 1: Key Points for Pipeline Improvements

References

[1] Medical Billing and Coding. *ICD-10-CM Codes List and Guide*. Accessed: 2024-09-28. URL: `https://www.medicalbillingandcoding.org/icd-10-cm/`.