ETL Project Overview

Project Overview

This project involves designing and implementing a scalable ETL (Extract, Transform, Load) pipeline to simulate the transformation of data and its subsequent analysis.

Purpose

The primary purpose of this ETL project is to:

1. **Simulate Data Transformation**: Focus on transforming data by selecting only the necessary tables and columns. This approach ensures that only relevant data is processed and loaded into the destination database, optimizing performance and storage efficiency.
2. **Analyze Transformed Data**: Use Jupyter Notebooks to perform real-time analysis and reporting on the transformed data, providing insights and actionable information.
3. **Manage Scheduling and Users**: Utilize a scheduler service to handle user management and automate the ETL process, ensuring the destination database is updated with any changes from the source database.

Components

1. **ETL Pipeline**:
    - **Extract**: Data is extracted from the source database, focusing on essential tables and columns.
    - **Transform**: The extracted data is transformed to meet the specific requirements of the destination database. Only necessary data is loaded, enhancing efficiency.
    - **Load**: The transformed data is loaded into the destination database, ready for analysis.
2. **Jupyter Notebooks**:
    - **Analysis**: Jupyter Notebooks are used to analyze the transformed data. This provides a flexible environment for performing detailed analysis and generating reports.
3. **Scheduler Service**:
    - **User Management**: Manages user access and permissions for interacting with the destination database.
    - **ETL Process Scheduling**: Automates the ETL process to simulate updates to the destination database whenever there are changes in the source database. This is achieved through Apache Airflow, which schedules and monitors the ETL tasks.

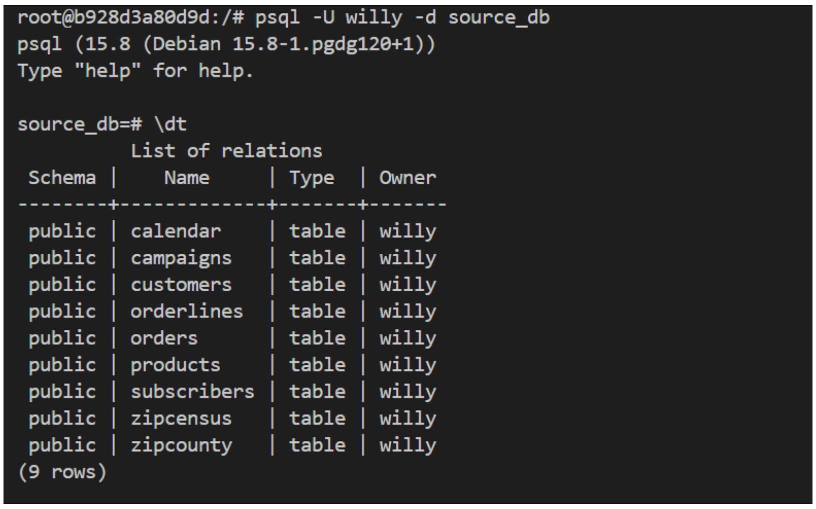
Technical Details

- **Technologies Used**:
    - **Flask**: For building the API and managing interactions with the ETL process.

- **PostgreSQL**: As both source and destination databases for storing and processing data.
- **PySpark**: For handling large-scale data transformations.
- **Apache Airflow**: For scheduling and managing the ETL process.
- **Jupyter Notebooks**: For data analysis and reporting.

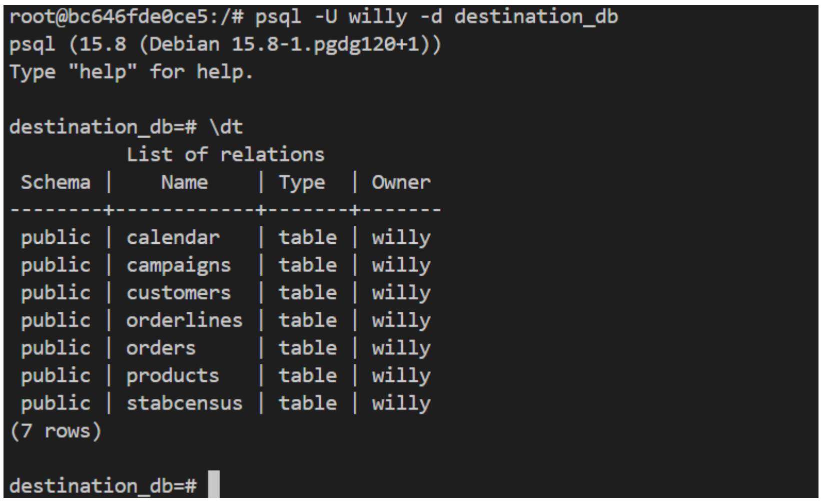
Results

The source database contains the following tables:



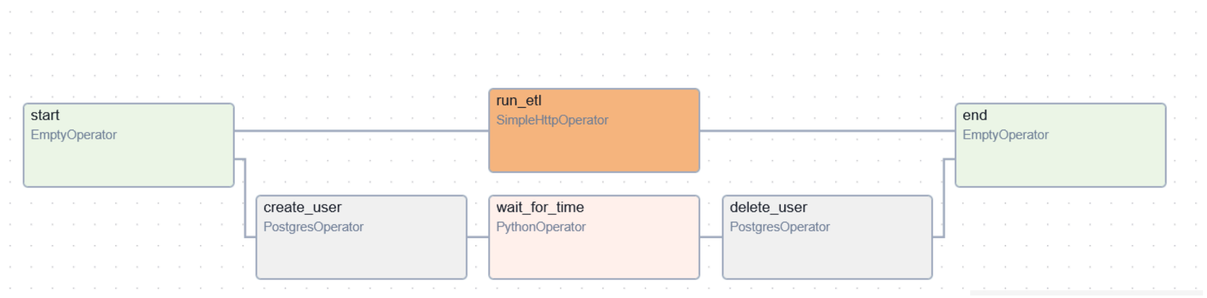
```
root@b928d3a80d9d:/# psql -U willy -d source_db
psql (15.8 (Debian 15.8-1.pgdg120+1))
Type "help" for help.

source_db=# \dt
          List of relations
 Schema |    Name     | Type  | Owner
--------+-------------+-------+-------
 public | calendar    | table | willy
 public | campaigns   | table | willy
 public | customers   | table | willy
 public | orderlines  | table | willy
 public | orders      | table | willy
 public | products    | table | willy
 public | subscribers | table | willy
 public | zipcensus   | table | willy
 public | zipcounty   | table | willy
(9 rows)
```

After careful consideration, I selected certain tables and columns to be included in the destination database. This process simulates the selection and loading of only the necessary data. The chosen tables and columns are as follows:

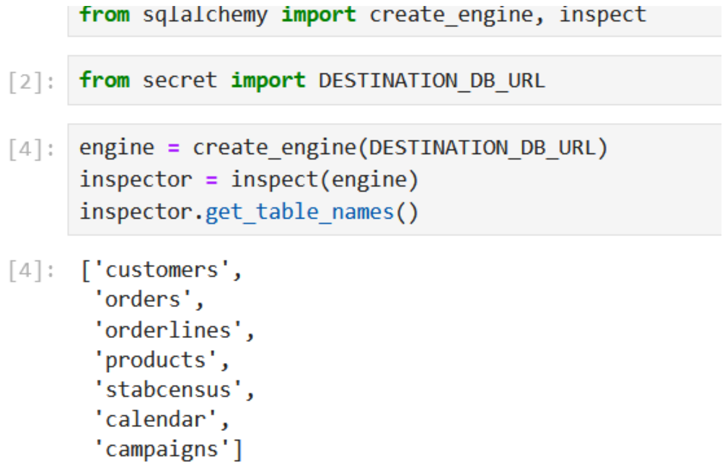


```
root@bc646fde0ce5:/# psql -U willy -d destination_db
psql (15.8 (Debian 15.8-1.pgdg120+1))
Type "help" for help.

destination_db=# \dt
          List of relations
 Schema |    Name    | Type  | Owner
--------+------------+-------+-------
 public | calendar   | table | willy
 public | campaigns  | table | willy
 public | customers  | table | willy
 public | orderlines | table | willy
 public | orders     | table | willy
 public | products   | table | willy
 public | stabcensus | table | willy
(7 rows)

destination_db=#
```

The purpose of the scheduler service (Apache Airflow) is to automatically manage the workflow of the ETL project and monitor the process for any issues. The figure below illustrates the workflow as displayed in the Airflow webserver UI:



After the data is loaded into the destination database, you can connect Jupyter Notebooks to the database using `psycopg2` and `SQLAlchemy`. The figure below shows all the tables available in the destination database:



```
      from sqlalchemy import create_engine, inspect

[2]:  from secret import DESTINATION_DB_URL

[4]:  engine = create_engine(DESTINATION_DB_URL)
      inspector = inspect(engine)
      inspector.get_table_names()

[4]:  ['customers',
       'orders',
       'orderlines',
       'products',
       'stabcensus',
       'calendar',
       'campaigns']
```

The `DESTINATION_DB_URL` is the connection string to the destination database, following this format: `postgresql+psycopg2://{db_user}:{db_password}@{db_host}:{db_port}/{db_name}`, where `db_user` is the `new_user` managed by the scheduler service. The scheduler service grants read privileges to the `new_user` for the database tables only for a specified duration (5 hours).