DAG Design and Testing Plan

DAG Design:

Introduction:

This document outlines the design and testing plan for the ETL (Extract, Transform, Load) pipeline implemented using Apache Airflow.

Overview of the ETL Process:

The ETL pipeline is designed to extract sales data from both PostgreSQL and CSV sources, transform it by aggregating sales information, and load the aggregated data into a MySQL database. The pipeline consists of four main tasks: extracting data from PostgreSQL, extracting data from CSV, transforming the data, and loading the transformed data into MySQL.

DAG Structure:

The DAG is scheduled to run daily and is configured to not catch up on missed runs. It consists of four tasks:

- **extract_postgres**: Extracts sales data from PostgreSQL.
- **extract_csv**: Extracts additional sales data from a CSV file.
- **transform_data**: Combines and transforms the extracted data.
- **load_to_mysql**: Loads the transformed data into a MySQL database.

Task Descriptions:

- **extract_postgres**: Connects to a PostgreSQL database and retrieves sales data from the "online_sales" table.
- **extract_csv**: Reads sales data from a CSV file located at a predefined path.
- **transform_data**: Combines the sales data from both sources, performs data cleansing, aggregates sales information by product, and calculates total quantities and sale amounts.
- **load_to_mysql**: Establishes a connection to a MySQL database and loads the transformed data into a table named "aggregated_sales".

Justification for Operator Choices:

PythonOperator is used for each task as it allows the execution of Python functions. This provides flexibility in defining the logic for extracting, transforming, and loading data. Additionally, Python functions facilitate code reuse and maintainability.

Airflow Deployment:

- Ensure that Apache Airflow is installed and configured on your system or server.
- Copy the **main.py** script containing the DAG definition to the Airflow DAGs directory.
- Start the Airflow webserver and scheduler using the following commands:

```
airflow webserver
airflow scheduler
```

- Access the Airflow web interface using a web browser and navigate to the DAGs list.
- Locate the **sales_total** DAG and toggle the switch to enable it.
- Trigger the DAG manually to initiate the ETL process.
- Monitor the DAG run status and task execution logs in the Airflow web interface.

Testing Plan:

Description of Testing Scenarios:

- Successful execution: All tasks complete without errors, and the data is loaded into the MySQL database.
- Empty PostgreSQL table: The "online_sales" table in PostgreSQL is empty.
- Missing CSV file: The CSV file containing additional sales data is not found.
- Data transformation error: An error occurs during data transformation due to invalid values or unexpected data format.

Expected Outcomes:

- Task logs indicate successful execution, and data is present in the MySQL database table.
- Task logs indicate that the PostgreSQL extraction task completed successfully, but no data is loaded into the MySQL table.
- Task logs indicate a file not found error for the CSV extraction task, and no data is loaded into the MySQL table.
- Task logs indicate an error during the data transformation task, and no data is loaded into the MySQL table.

Steps to Execute the Testing Plan:

- Trigger the DAG manually.
- Monitor task logs and verify outcomes for each scenario.

Results and Observations:

- Scenario 1: Successful execution confirms the functionality of the pipeline.
- Scenario 2: Proper error handling is observed, preventing the loading of empty data.

- Scenario 3: File not found error handling is implemented, preventing data loading.
- Scenario 4: Error handling in data transformation task prevents invalid data from being loaded into the MySQL table.

Challenges Encountered During Implementing the Project:

Faced many dependency issues and had to check many logs for many sources to fix them.

Had to debug data extraction parts because of the correct formatting.

Had to determine and interfere airflow services' port usages.