DATA PIPELINE DESIGN

Ingestion

- For flat files, used Pandas library to read the CSV files into dataframe.
- For APIs, used requests module to fetch data from both internal and external APIs and converted JSON data to dataframes. Testing done through mock server created using Postman tool.
- For Postgres tables, used sqlalchemy ORM to read the records from database tables into dataframe.

Standardization

- To transform and merge data from different sources, column names and data types conversions(object to datetime) done to ingest into standardized database after reading from the sources.
- Export(.sql file) created after inserting data into the standardized database.
- Changes in data types and column names are done looking into the sample files shared. Edge cases in handling date, column name conversions may prevail if deviations seen from sample data.

Data Preprocessing Pipeline

- Data preprocessing script is written using Python which covers handling missing data, duplicate records and abnormal values.
- For handling abnormal values, took absolute values for the columns with integer(quantity, stock) and float(prices, total_amount) values.
- For handling missing data in total_amount, calculation made using prices given in products table and default values inserted in other date, integer and float columns.

The preprocessing methods are implemented on the sample data only and can be extended/improved for more similar datasets.

Cloud Architecture

- Cloud-based architecture designed to deploy the prepared data processing pipeline ensuring it is scalable and resilient.
- AWS services, the selection choices & their brief to explain data flow diagram for each of the data source.
- Further alternative/optional services can be utilized for cost-reduction and low latency as per scaling requirements like AWS DMS, AWS EMR can be employed.
- AWS API Gateway can be used to expose the few APIs on top of the prepared database
- Airflow tasks or Step Function workflow can be used for orchestration.