ETL, Reporting & PDF Export Pipeline

Objective

The objective of this task is to evaluate your ability to design a **scalable ETL pipeline**, process and normalize data, store it in a structured database, and generate automated **reporting with both CSV and PDF exports**.

Task Requirements

Step 1: Data Ingestion (ETL Pipeline - CSV & API)

- Write a **Python script** that extracts data from **two sources**:
  i. A **CSV file** (`sales_data.csv`) containing sales data.
  ii. A **REST API endpoint** (e.g., exchange rates)
- Process the data by:
  - Converting all **currency values to USD** using the exchange rate API.
  - Standardizing **date formats** and handling missing values.
  - Removing **duplicate records** and ensuring data integrity.
  - Storing the cleaned data in a **PostgreSQL or SQLite database**.

Step 2: Database Schema & Optimization

- Create a well-structured **relational database schema** that optimizes storage and query performance.
- Use **appropriate data types and indexing** to ensure efficient data retrieval.
- Ensure **referential integrity** by properly linking transactions with exchange rates.

Step 3: Data Querying & Automated Report Generation

- Write **SQL queries** to:
  - Convert all sales values to **USD** based on exchange rates.
  - Aggregate **total sales by affiliate and category**.
  - Generate a **monthly sales summary** showing trends and report-ready insights.

Step 4: PDF & CSV Report Generation

- **Export the final report in both CSV and PDF formats** using Python.
- The **PDF report** should include:
  - A **title, summary statistics, and tabular representation** of the aggregated data.
  - A **bar chart or line graph** visualizing sales trends (using Matplotlib/Seaborn).
  - A professional layout formatted with tools like `ReportLab` or `WeasyPrint`.
- Ensure both CSV and PDF reports are **automatically saved** to a designated folder.

Bonus (Optional Enhancements)

✅ **Logging & Error Handling**: Implement logging to track the ETL process and log anomalies.
✅ **Unit Testing**: Use `pytest` to validate data transformations and ensure no errors in processing.