Lead Data Engineer Task

Scenario: E-commerce Sales Data Processing with Databricks

You've been assigned to design and implement a data processing system using Databricks for an e-commerce platform. This platform generates a lot of sales data, including details about orders, products, customers, and transactions. Your goal is to use Databricks to create a scalable, efficient, and reliable data engineering solution. This solution should process and analyze the data to provide valuable insights for business stakeholders.

Source Datasets:

You are required to download the datasets from the below drive
https://drive.google.com/drive/folders/1eWxfGcFwJJKAK0Nj4zZeCVx6gagPEEVc?usp=sharing

Data Transformation and Processing:

Your task is to process the raw sales data using **Databricks notebooks** and **PySpark**. You need to clean up the data and transform it into structured formats suitable for analysis. Specifically, you should **create a master table** and **perform aggregations** based on the requirements provided.

Note: Write appropriate Test Cases (Unit Tests) to ensure the correctness for the given scenarios. Use PySpark (not SQL) for this task.

Task

1) **Create raw tables for each source dataset**
2) **Create an enriched table for customers and products**
3) **Create an enriched table which has**
    a) **order information**
        i) **Profit rounded to 2 decimal places**
    b) **Customer name and country**
    c) **Product category and sub category**
4) **Create an aggregate table that shows profit by**

    a) **Year**
    b) **Product Category**
    c) **Product Sub Category**
    d) **Customer**

5) **Using SQL output the following aggregates**
    a) **Profit by Year**
    b) **Profit by Year + Product Category**
    c) **Profit by Customer**
    d) **Profit by Customer + Year**

Notes:

- Ensure you understand the task requirements thoroughly before starting.
- Pay attention to specific details and expectations outlined in the task descriptions.
- Use a test-driven development approach to validate the correctness of your implementations.
- Write comprehensive test cases to cover different scenarios and edge cases.
- Ensure your solution handles data quality issues and implements robust error-handling mechanisms.
- Document your code and assumptions clearly to make it understandable for others.
- Consider performance implications and optimize your code for efficiency and scalability.