Assignment Case Studies on Transactions in SQL Server

Problem Statement: E-Commerce Transaction Handling Using SQL Server

Scenario Overview:

An e-commerce platform enables users to purchase products. When a customer places an order, the following operations must occur **atomically** and **consistently**:

1. The product stock must be **reduced** in the Products table.
2. A new order record must be **inserted** into the Orders table.
3. The total amount must be **deducted** from the customer's UserWallet.

All these operations must succeed as a single **transactional unit**. If any one of the steps fails (e.g., insufficient stock, insufficient balance), **the entire transaction must be rolled back**, ensuring no partial changes are saved to the database.

Objectives:

- Demonstrate SQL Server's **transaction control** using BEGIN TRANSACTION, COMMIT, and ROLLBACK.
- Maintain **data consistency** and **integrity** using **ACID properties**.
- Prevent common transaction anomalies (e.g., partial orders, negative stock, overdraft wallets).
- Apply proper **error handling** and **logging** mechanisms.

Tables Involved:

1. Products(ProductID, ProductName, Price, Stock)
2. UserWallet(UserID, Balance)
3. Orders(OrderID, UserID, ProductID, Quantity, TotalAmount, OrderDate)

Business Rules:

- A customer **cannot** place an order if:
    - The product **stock** is insufficient.
    - The user's **wallet balance** is insufficient.
- All steps must be **wrapped in a single SQL Server transaction**.
- If any step fails, **rollback** all changes.

- If successful, **commit** the transaction and optionally log the order.

Expected Outcome:

- Data in all related tables is updated **only if all checks pass**.
- No record of the order is saved if an error occurs.
- System maintains **consistency, isolation**, and **durability**.
- Demonstrates SQL Server's capability to manage real-world transactions securely.
