Assignment Case Studies on Triggers in SQL Server

Problem Statement: Design and Implement SQL Server Triggers to Automate and Enforce Business Logic in an E-Commerce System

---

Context:

In a typical **e-commerce application**, customers can browse products, add them to a cart, and place orders. When an order is placed, the following backend operations occur:

- The product's available **stock** must be **reduced**.
- A record of the purchase is added to the **Orders** table.
- The customer's **wallet balance** is **deducted** by the total order amount.

While these operations can be performed via application code, it is important to **enforce business rules and maintain data integrity at the database level** to prevent inconsistencies, especially in multi-user environments.

Objective:

Your task is to implement **SQL Server Triggers** that automatically perform actions or validations when data is modified in the following tables:

- Products
- UserWallet
- Orders

You will use **AFTER** and **INSTEAD OF** triggers to:

- Enforce **business rules**
- Maintain **audit logs**
- Prevent invalid data changes
- Automatically generate **alerts** for critical events (e.g., low stock)

Note: Create required tables to implement this problem.

Tasks: You are required to write **SQL Server triggers** for the following business requirements:

---

1. Stock Change Auditing

When the stock of any product is **updated**, automatically insert a record into the StockAudit table capturing:

- Product ID
- Previous stock
- Updated stock
- Date/time of change

Use an AFTER UPDATE trigger and DELETED/INSERTED pseudo-tables.

---

2. Wallet Balance Auditing

Whenever a user's wallet balance is **updated**, insert an entry in the WalletAudit table indicating:

- User ID
- Old balance
- New balance
- Type of transaction: Debit or Credit
- Timestamp

Use an AFTER UPDATE trigger on the UserWallet table.

---

3. Prevent Order Placement on Zero Stock

Disallow order placement (insertion into Orders) if the related product has Stock = 0.

Use an INSTEAD OF INSERT trigger on Orders.
If stock is zero, raise an error and prevent insertion.

---

4. Alert on Low Stock

After an order is inserted, check if the ordered product's stock has dropped **below 5**. If so, display a message using PRINT.

Use an AFTER INSERT trigger on Orders.
(Simulate a system notification that would be used by warehouse staff.)

Expected Learning Outcomes:

By completing this assignment, learners will:

- Understand the practical use of **triggers** to automate database behavior
- Learn to use INSERTED and DELETED pseudo-tables
- Handle **auditing**, **validation**, and **alerts** inside triggers
- Apply **business logic** at the **database level** rather than relying solely on application code
- Improve understanding of **data integrity**, **referential rules**, and **concurrency handling**