COMSATS University Islamabad, Lahore Campus
Block–B, Department of Computer Engineering
COMSATS University Islamabad, Lahore Campus 1.5KM Defence Road, Off Raiwind Road, Lahore

Assignment

Course Title:	Database Systems				Course Code:	CSC270	Credit Hours: 4(3,*1*)
Course Instructor/s:	Modassir Ishfaq				Programme Name:	BCE	
Semester:	4th	Batch:	Fa23	Section:	A&B	Date:	
Time Allowed:	**Within 3 days**				**Maximum Marks:**		**10**
Student's Name:	Muhammad Ahmad				Reg. No.	FA23-BCE-113	

Question:

You are managing the backend of an online store. Customers place orders, and the system must deduct inventory and record the transaction safely. The following tables are in use:

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Stock INT,
    Price DECIMAL(10,2), flag BOOL
);

CREATE TABLE Customer (
    customerID INT PRIMARY KEY,
    CustomerName VARCHAR(100),
    Balance INT,
    DateofJoining DATETIME
);

CREATE TABLE Orders (
    OrderID INT IDENTITY(1,1) PRIMARY KEY,
    ProductID INT,
    CustomerID INT,
    Quantity INT,
    TotalAmount DECIMAL(12,2),
    OrderDate DATETIME DEFAULT GETDATE()
);
```

Write a script for SQL Transaction to handle customers' orders ensuring the following:

- Product stock validation (enough quantity available to meet the order requirements)
- Customer balance validation (enough balance to process the whole order)
- Products get flagged when the stock is less than 10. Such items will not be available for newer customer (base this on DateofJoining in customer's table)
- Customers who joined 5 years or more ago can order low stock items with a limit of 2 per order.
- Any situation where order is not processable will result in the rollback of the whole transaction and generate relevant prompt.
- On success of order placement, all tables must be updated accordingly.

```sql
CREATE OR ALTER PROCEDURE sp_PlaceOrder
    @customerId INT,
    @productId INT,
    @quantity INT
AS
BEGIN
    -- Automatically rollback transaction on runtime errors
    SET XACT_ABORT ON;

    -- Prevent unnecessary messages for each statement
    SET NOCOUNT ON;

    DECLARE
        @stock INT,
        @price DECIMAL(10,2),
        @balance INT,
        @totalAmount DECIMAL(12,2),
        @dateOfJoining DATETIME,
        @today DATETIME = GETDATE();

    BEGIN TRY
        BEGIN TRANSACTION;

        -- • Product stock validation (enough quantity available to meet the order requirements)
        SELECT @stock = Stock, @price = Price
        FROM Products
        WHERE ProductID = @productId;

        IF @stock IS NULL
        BEGIN
            THROW 50001, 'Product does not exist.', 1;
        END

        IF @stock < @quantity
        BEGIN
            THROW 50002, 'Not enough stock available.', 1;
        END

        -- • Customer balance validation (enough balance to process the whole order)
        SELECT @balance = Balance, @dateOfJoining = DateOfJoining
        FROM Customer
        WHERE CustomerID = @customerId;

        IF @balance IS NULL
        BEGIN
            THROW 50003, 'Customer does not exist.', 1;
        END

        SET @totalAmount = @price * @quantity;

        IF @balance < @totalAmount
        BEGIN
            THROW 50004, 'Customer has insufficient balance.', 1;
        END

        -- • Products get flagged when the stock is less than 10.
        --   Such items will not be available for newer customers (based on DateOfJoining)
        IF @stock < 10
        BEGIN
            IF DATEDIFF(YEAR, @dateOfJoining, @today) < 5
            BEGIN
                THROW 50005, 'New customers cannot order low stock items.', 1;
            END

            -- • Customers who joined 5 years or more ago can order low stock items with a limit of
2 per order.

            ELSE IF @quantity > 2
            BEGIN
```

```
                THROW 50006, 'Max 2 low-stock items allowed per order for old customers.', 1;
            END
        END


        -- • On success of order placement, all tables must be updated accordingly.


        -- Insert order
        INSERT INTO Orders (ProductID, CustomerID, Quantity, TotalAmount)
        VALUES (@productId, @customerId, @quantity, @totalAmount);

        -- Update product stock
        UPDATE Products
        SET Stock = Stock - @quantity
        WHERE ProductID = @productId;

        -- Update customer balance
        UPDATE Customer
        SET Balance = Balance - @totalAmount
        WHERE CustomerID = @customerId;

        -- Flag product if stock falls below 10
        UPDATE Products
        SET flag = 1
        WHERE ProductID = @productId AND Stock < 10;


        -- • Any situation where order is not processable will result in the
        --   rollback of the whole transaction and generate relevant prompt.
        COMMIT;
        PRINT 'Order placed successfully.';
    END TRY
    BEGIN CATCH
        -- Rollback only if a transaction is active
        IF @@TRANCOUNT > 0 -- Check open transaction count
            ROLLBACK;
       THROW; -- To display thy error messages
    END CATCH
END;
```

Testing Edge Cases

Test Case 1: Valid Order

```
154  --
155  EXEC sp_PlaceOrder @CustomerID = 1, @ProductID = 102, @Quantity = 2;
156
```

```
Messages
Order placed successfully.
```

Test Case 2: Not Enough Stock

```
159  -- Stock of ProductID 104 is 4
160  EXEC sp_PlaceOrder @CustomerID = 1, @ProductID = 104, @Quantity = 10;
```

```
Messages
Msg 50002, Level 16, State 1, Procedure sp_PlaceOrder, Line 36 [Batch Start Line 158]
Not enough stock available.
```

Test Case 3: New Customer Cannot Order Low-Stock Item

```
162  -- Sara Khan (CustomerID 2) joined in 2023 (< 5 years ago)
163  -- ProductID 104 has stock < 10
164  EXEC sp_PlaceOrder @CustomerID = 2, @ProductID = 104, @Quantity = 1;
```

```
Messages
Msg 50005, Level 16, State 1, Procedure sp_PlaceOrder, Line 62 [Batch Start Line 162]
New customers cannot order low stock items.
```

Test Case 4: Low-Stock Quantity Limit for Old Customers

```
166  -- Ahmed Butt (CustomerID 3) joined in 2017 (> 5 years ago)
167  -- ProductID 104 has stock < 10
168  EXEC sp_PlaceOrder @CustomerID = 3, @ProductID = 104, @Quantity = 3;
169
```

1 %

Messages

```
Msg 50006, Level 16, State 1, Procedure sp_PlaceOrder, Line 69 [Batch Start Line 167]
Max 2 low-stock items allowed per order for old customers.
```

Test Case 5: Customer Has Insufficient Balance

```
154  -- Customer have insufficient amount
155  EXEC sp_PlaceOrder @CustomerID = 2, @ProductID = 102, @Quantity = 2;
156
```

1 %

Messages

```
Msg 50004, Level 16, State 1, Procedure sp_PlaceOrder, Line 53 [Batch Start Line 154]
Customer has insufficient balance.
```