**CASE STUDY**

Let's consider a case study related to a fictional e-commerce database. Suppose you are tasked with designing a database for an online store that sells products, manages customers, and handles orders. Below is a simplified version of the case study along with some questions to guide you:

**Case Study: E-Commerce Database**

**Entities:**

1] Customers:

CustomerID (Primary Key)

FirstName

LastName

Email

Phone

2] Products:

ProductID (Primary Key)

ProductName

Price

StockQuantity

3] Orders:

OrderID (Primary Key)

CustomerID (Foreign Key referencing Customers)

OrderDate

TotalAmount

4] OrderDetails:

OrderDetailID (Primary Key)

OrderID (Foreign Key referencing Orders)

ProductID (Foreign Key referencing Products)

Quantity

Subtotal

**Questions:**

**Create the Database Schema:**

Design the SQL statements to create tables for Customers, Products, Orders, and OrderDetails. Define appropriate data types, constraints, and relationships.

**Insert Sample Data:**

Populate the tables with sample data. Create at least two customers, five products, and a few orders with corresponding order details.

**Retrieve Customer Information:**

Write a SQL query to retrieve the details of a specific customer (e.g., by FirstName or CustomerID).

**Check Product Availability:**

Write a query to check the availability of a specific product based on the StockQuantity.

**Calculate Order Total:**

Write a query to calculate the total amount spent by a customer in all their orders.

**Update Product Stock:**

Create a query to update the stock quantity of a product after processing an order.

**Join Tables:**

Write a query to retrieve a list of orders with details including customer names, order date, and total amount.

**Aggregate Functions:**

Use aggregate functions to find the average order total and the maximum quantity of a product ordered.

**Delete Records:**

Write a query to delete a specific order and its associated details.

**Bonus: Transaction Handling:**

Implement a transaction that deducts the ordered quantity from the product stock only if the product is available. Rollback the transaction if any part of the process fails.

**1]** create database ECommerce DB;

**2]**

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

Phone VARCHAR(20)

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Price DECIMAL(10, 2),

StockQuantity INT

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT REFERENCES Customers(CustomerID),

OrderDate DATE,

TotalAmount DECIMAL(10, 2)

);

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT REFERENCES Orders(OrderID),

ProductID INT REFERENCES Products(ProductID),

Quantity INT,

Subtotal DECIMAL(10, 2)

);

**3]**

INSERT INTO Customers VALUES (1, 'John', 'Doe', 'john.doe@email.com', '123-456-7890');

INSERT INTO Customers VALUES (2, 'Jane', 'Smith', 'jane.smith@email.com', '987-654-3210');

INSERT INTO Customers VALUES (3, 'Jason', 'Dsouza','jason.dsouza@email.com', '453-486-7870');

INSERT INTO Customers VALUES (4, 'Micky', 'Dias', 'micky.dias@email.com', '987-484-3260');

INSERT INTO Customers VALUES (5, 'Minnie', 'Fernandes', 'minnie.fernandes@email.com', '485-746-7890');

INSERT INTO Customers VALUES (6, 'Bob', 'Lucas', 'bob.lucas@email.com', '687-144-8750');

INSERT INTO Products VALUES (1, 'Product A', 19.99, 100);

INSERT INTO Products VALUES (2, 'Product B', 29.99, 50);

INSERT INTO Products VALUES (3, 'Product C', 51.99, 150);

INSERT INTO Products VALUES (4, 'Product D', 64.99, 500);

INSERT INTO Products VALUES (5, 'Product E', 84.99, 200);

INSERT INTO Orders VALUES (101, 1, '2023-01-15', 69.97);

INSERT INTO Orders VALUES (102, 2, '2023-02-20', 155.97);

INSERT INTO Orders VALUES (103, 3, '2023-03-12', 119.96);

INSERT INTO Orders VALUES (104, 4, '2023-03-21', 169.98);

INSERT INTO Orders VALUES (105, 5, '2023-04-11', 51.99);

INSERT INTO Orders VALUES (106, 6, '2023-05-2', 69.98);

INSERT INTO OrderDetails VALUES (1001, 101, 1, 2, 39.98);

INSERT INTO OrderDetails VALUES (1002, 101, 2, 1, 29.99);

INSERT INTO OrderDetails VALUES (1003, 102, 3, 3, 155.97);

INSERT INTO OrderDetails VALUES (1004, 103, 2, 4, 119.96);

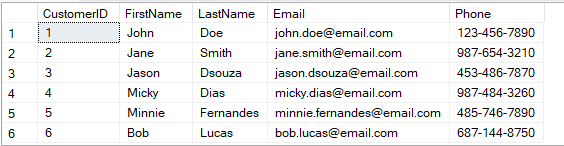
INSERT INTO OrderDetails VALUES (1005, 104, 5, 2, 169.98);

INSERT INTO OrderDetails VALUES (1006, 105, 3, 1, 51.99);

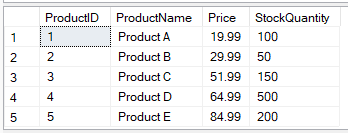
INSERT INTO OrderDetails VALUES (1007, 106, 4, 1, 64.99);

**4]**

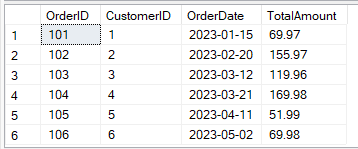
select \* from Customers;



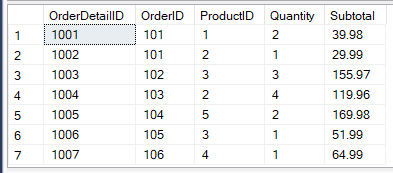
select \* from Products;



select \* from Orders;



select \* from OrderDetails;



**5]**

-- Retrieve Customer Information by FirstName

SELECT \* FROM Customers WHERE FirstName = 'John';



-- Retrieve Customer Information by CustomerID

SELECT \* FROM Customers WHERE CustomerID = 2;



**6]**

-- Check Product Availability by ProductID

SELECT ProductName, StockQuantity FROM Products WHERE ProductID = 1;



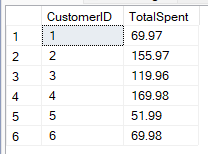
**7]**

-- Calculate Total Amount Spent by a Customer

SELECT CustomerID, SUM(TotalAmount) AS TotalSpent

FROM Orders

GROUP BY CustomerID;



**8]**

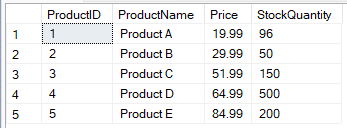
-- Update Product Stock after Processing an Order

UPDATE Products

SET StockQuantity = StockQuantity - 2

WHERE ProductID = 1;

select \* from Products



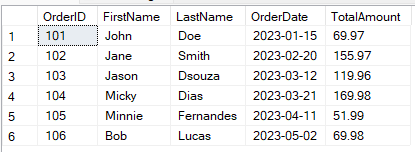
**9]**

-- Retrieve List of Orders with Customer Names, Order Date, and Total Amount

SELECT O.OrderID, C.FirstName, C.LastName, O.OrderDate, O.TotalAmount

FROM Orders O

JOIN Customers C ON O.CustomerID = C.CustomerID;



**10]**

-- Find Average Order Total

SELECT AVG(TotalAmount) AS AvgOrderTotal FROM Orders;



-- Find Maximum Quantity of a Product Ordered

SELECT MAX(Quantity) AS MaxQuantityOrdered FROM OrderDetails;



**11]**

BEGIN TRANSACTION;

-- Check Product Availability

DECLARE @AvailableStock INT;

SELECT @AvailableStock = StockQuantity FROM Products WHERE ProductID = 1;

-- Deduct Ordered Quantity from Stock if Available

IF @AvailableStock >= 2

BEGIN

UPDATE Products SET StockQuantity = StockQuantity - 2 WHERE ProductID = 1;

-- Commit the Transaction

COMMIT;

PRINT 'Transaction committed successfully!';

END

ELSE

BEGIN

-- Rollback the Transaction

ROLLBACK;

PRINT 'Transaction rolled back due to insufficient stock!';

END

