```
--SQL E-Commerce Mini Project

CREATE TABLE Users (
   user_id INT PRIMARY KEY IDENTITY(1,1),
   name VARCHAR(50) NOT NULL,
   email VARCHAR(50) UNIQUE NOT NULL,
   phone VARCHAR(15),
   address VARCHAR(100));

CREATE TABLE Products (
   product_id INT PRIMARY KEY IDENTITY(1,1),
   name VARCHAR(50) NOT NULL,
   category VARCHAR(30),
   price DECIMAL(10,2) NOT NULL,
   stock_quantity INT DEFAULT 0);

Drop Table orders;
CREATE TABLE Orders (
   order_id INT PRIMARY KEY IDENTITY(1,1),
   user_id INT FOREIGN KEY REFERENCES Users(user_id),
   order_date DATE DEFAULT GETDATE(),
   total_amount DECIMAL(10,2));

CREATE TABLE Order_Items (
   order_item_id INT PRIMARY KEY IDENTITY(1,1),
   order_id INT FOREIGN KEY REFERENCES Orders(order_id),
   product_id INT FOREIGN KEY REFERENCES Products(product_id),
   quantity INT NOT NULL,
   price DECIMAL(10,2) NOT NULL);


--Inserting Sample Data

INSERT INTO Users (name, email, phone, address)
VALUES ('Mahi', 'mahi@example.com', '1234567890', 'Chittorgarh');

INSERT INTO Products (name, category, price, stock_quantity)
VALUES ('Laptop', 'Electronics', 50000, 10),
      ('Headphones', 'Electronics', 1500, 25);

INSERT INTO Orders (user_id, total_amount)
VALUES (1, 51500);

INSERT INTO Order_Items (order_id, product_id, quantity, price)
```

```
VALUES (1, 1, 1, 50000),
       (1, 2, 1, 1500);


--Geting all orders of a user

SELECT o.order_id, o.order_date, o.total_amount
FROM Orders o
JOIN Users u ON o.user_id = u.user_id
WHERE u.name = 'Mahi';

--Show products with low stock (<5)
SELECT * FROM Products
WHERE stock_quantity < 5;

--Total sales per product
SELECT p.name, SUM(oi.quantity) AS total_sold
FROM Order_Items oi
JOIN Products p ON oi.product_id = p.product_id
GROUP BY p.name;

--Most expensive order
SELECT TOP 1 * FROM Orders
ORDER BY total_amount DESC;

--Add new order
BEGIN TRANSACTION;

INSERT INTO Orders (user_id, total_amount)
VALUES (1, 6500);

INSERT INTO Order_Items (order_id, product_id, quantity, price)
VALUES (SCOPE_IDENTITY(), 2, 2, 1500);

COMMIT;

--Showing all orders with customer names
SELECT o.order_id, u.name AS customer_name, o.order_date, o.total_amount
FROM Orders o
JOIN Users u ON o.user_id = u.user_id;

--Showing products with stock less than 5
SELECT *
FROM Products
```

```
WHERE stock_quantity < 5;

--Total sales per product
SELECT p.name AS product_name, SUM(oi.quantity) AS total_sold, SUM(oi.price * oi.quantity)
AS total_revenue
FROM Order_Items oi
JOIN Products p ON oi.product_id = p.product_id
GROUP BY p.name;

--Most expensive order
SELECT TOP 1 *
FROM Orders
ORDER BY total_amount DESC;

--Orders of a specific customer
SELECT o.order_id, o.order_date, o.total_amount
FROM Orders o
JOIN Users u ON o.user_id = u.user_id
WHERE u.name = 'Mahi';

--Updating stock after a new order
UPDATE Products
SET stock_quantity = stock_quantity - 2
WHERE product_id = 2;

--Deleting an order
DELETE FROM Order_Items WHERE order_id = 1;
DELETE FROM Orders WHERE order_id = 1;

--Top selling product
SELECT TOP 1 p.name AS product_name, SUM(oi.quantity) AS total_sold
FROM Order_Items oi
JOIN Products p ON oi.product_id = p.product_id
GROUP BY p.name
ORDER BY total_sold DESC;

--Total revenue per customer
SELECT u.name AS customer_name, SUM(oi.price * oi.quantity) AS total_spent
FROM Users u
JOIN Orders o ON u.user_id = o.user_id
JOIN Order_Items oi ON o.order_id = oi.order_id
GROUP BY u.name;

--Orders in last 7 days
```

```
SELECT *
FROM Orders
WHERE order_date >= DATEADD(DAY, -7, GETDATE());

--Products not sold yet
SELECT *
FROM Products
WHERE product_id NOT IN (SELECT product_id FROM Order_Items);

--Average order value
SELECT AVG(total_amount) AS avg_order_value
FROM Orders;

--Customer with max orders
SELECT TOP 1 u.name, COUNT(o.order_id) AS total_orders
FROM Users u
JOIN Orders o ON u.user_id = o.user_id
GROUP BY u.name
ORDER BY total_orders DESC;

--Updating stock after orders
UPDATE P
SET P.stock_quantity = P.stock_quantity - OI.quantity
FROM Products P
JOIN Order_Items OI ON P.product_id = OI.product_id;
```