CODING CHALLENGE

SCHEMA

```
-- DATABASE CREATION

CREATE DATABASE Ecommerce;

USE Ecommerce;


-- CREATING TABLES


-- CUSTOMER TABLE

CREATE TABLE Customers(customer_ID INT PRIMARY KEY, name
VARCHAR(200), email VARCHAR(200), password VARCHAR(200));


-- PRODUCTS TABLE

CREATE TABLE Products(product_ID INT PRIMARY KEY, name
VARCHAR(200), price DECIMAL(10,2), description TEXT, stockQuantity
INT );


-- CARTT TABLE

CREATE TABLE Cart (cart_ID INT PRIMARY KEY, customer_ID INT,
product_ID INT, quantity INT, FOREIGN KEY (customer_ID) REFERENCES
Customers (customer_ID), FOREIGN KEY (product_ID) REFERENCES
Products(product_ID) );
```

-- ORDER TABLE

CREATE TABLE Orders(order_ID int PRIMARY KEY , customer_ID INT, order_date DATE, total_price DECIMAL(10,2), shipping_address TEXT, FOREIGN KEY(customer_ID) REFERENCES Customers(customer_ID));

-- Order items

CREATE TABLE Order_items(orderItem_ID int PRIMARY KEY, order_ID INT, product_ID INT, quantity INT,itemAmount DECIMAL(10,2), FOREIGN KEY(order_ID) REFERENCES Orders(order_ID), FOREIGN KEY(product_ID) REFERENCES Products(product_ID));

-- DATA INSERTION

-- product table

INSERT INTO Products (product_ID, name, description, price, stockQuantity) VALUES

(1, 'Laptop', 'High-performance laptop', 800.00, 10),

(2, 'Smartphone', 'Latest smartphone', 600.00, 15),

(3, 'Tablet', 'Portable tablet', 300.00, 20),

(4, 'Headphones', 'Noise-canceling', 150.00, 30),

(5, 'TV', '4K Smart TV', 900.00, 5),

(6, 'Coffee Maker', 'Automatic coffee maker', 50.00, 25),

```
(7, 'Refrigerator', 'Energy-efficient', 700.00, 10),
(8, 'Microwave Oven', 'Countertop microwave', 80.00, 15),
(9, 'Blender', 'High-speed blender', 70.00, 20),
(10, 'Vacuum Cleaner', 'Bagless vacuum cleaner', 120.00, 10);

-- CUSTOMER TABLE
ALTER TABLE Customers ADD address TEXT;
INSERT INTO Customers (customer_ID, name, email, password) VALUES
(1, 'John Doe', 'johndoe@example.com', 'password1'),
(2, 'Jane Smith', 'janesmith@example.com', 'password2'),
(3, 'Robert Johnson', 'robert@example.com', 'password3'),
(4, 'Sarah Brown', 'sarah@example.com', 'password4'),
(5, 'David Lee', 'david@example.com', 'password5'),
(6, 'Laura Hall', 'laura@example.com', 'password6'),
(7, 'Michael Davis', 'michael@example.com', 'password7'),
(8, 'Emma Wilson', 'emma@example.com', 'password8'),
(9, 'William Taylor', 'william@example.com', 'password9'),
(10, 'Olivia Adams', 'olivia@example.com', 'password10');

UPDATE Customers
SET address = '123 Main St, City'
```

```
WHERE customer_ID = 1;

UPDATE Customers
SET address = '456 Elm St, Town'
WHERE customer_ID = 2;

UPDATE Customers
SET address = '789 Oak St, Village'
WHERE customer_ID= 3;

UPDATE Customers
SET address = '101 Pine St, Suburb'
WHERE customer_ID = 4;

UPDATE Customers
SET address = '234 Cedar St, District'
WHERE customer_ID = 5;

UPDATE Customers
SET address = '567 Birch St, County'
WHERE customer_ID = 6;
```

```
UPDATE Customers
SET address = '890 Maple St, State'
WHERE customer_ID = 7;

UPDATE Customers
SET address = '321 Redwood St, Country'
WHERE customer_ID = 8;

UPDATE Customers
SET address = '432 Spruce St, Province'
WHERE customer_ID = 9;

UPDATE Customers
SET address = '765 Fir St, Territory'
WHERE customer_ID = 10;

select * from Customers;

-- ORDER TABLE
```

```
INSERT INTO Orders (order_id, customer_id, order_date, total_price)
VALUES
(1, 1, '2023-01-05', 1200.00),
(2, 2, '2023-02-10', 900.00),
(3, 3, '2023-03-15', 300.00),
(4, 4, '2023-04-20', 150.00),
(5, 5, '2023-05-25', 1800.00),
(6, 6, '2023-06-30', 400.00),
(7, 7, '2023-07-05', 700.00),
(8, 8, '2023-08-10', 160.00),
(9, 9, '2023-09-15', 140.00),
(10, 10, '2023-10-20', 1400.00);

-- ORDER ITEM TABLE
INSERT INTO Order_items (orderItem_id, order_id, product_id,
quantity, itemAmount) VALUES
(1, 1, 1, 2, 1600.00),
(2, 1, 3, 1, 300.00),
(3, 2, 2, 3, 1800.00),
(4, 3, 5, 2, 1800.00),
(5, 4, 4, 4, 600.00),
(6, 4, 6, 1, 50.00),
```

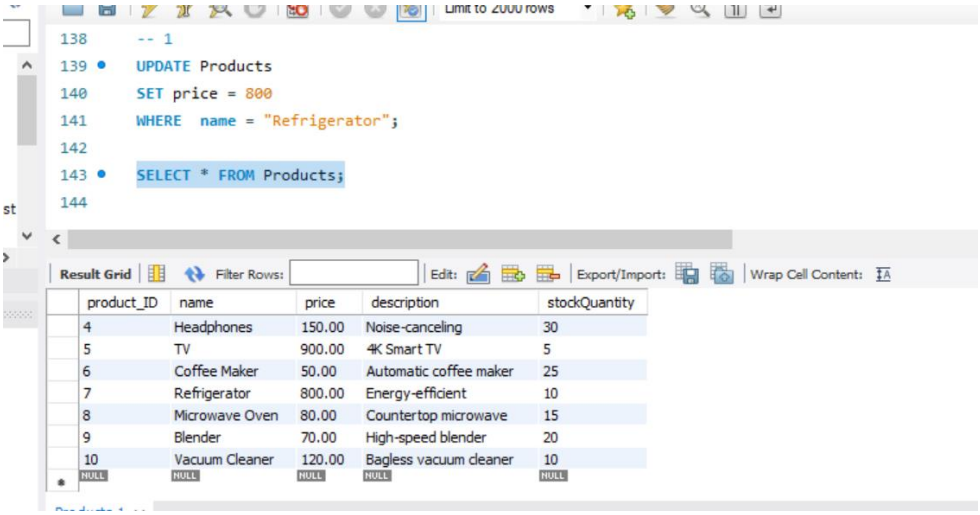
```
(7, 5, 1, 1, 800.00),
(8, 5, 2, 2, 1200.00),
(9, 6, 10, 2, 240.00),
(10, 6, 9, 3, 210.00);


-- CART TABLE
INSERT INTO Cart (cart_ID, customer_ID, product_ID, quantity) VALUES
(1, 1, 1, 2),
(2, 1, 3, 1),
(3, 2, 2, 3),
(4, 3, 4, 4),
(5, 3, 5, 2),
(6, 4, 6, 1),
(7, 5, 1, 1),
(8, 6, 10, 2),
(9, 6, 9, 3),
(10, 7, 7, 2);
```

1. Update refrigerator product price to 800.

```
UPDATE Products
```

SET price = 800

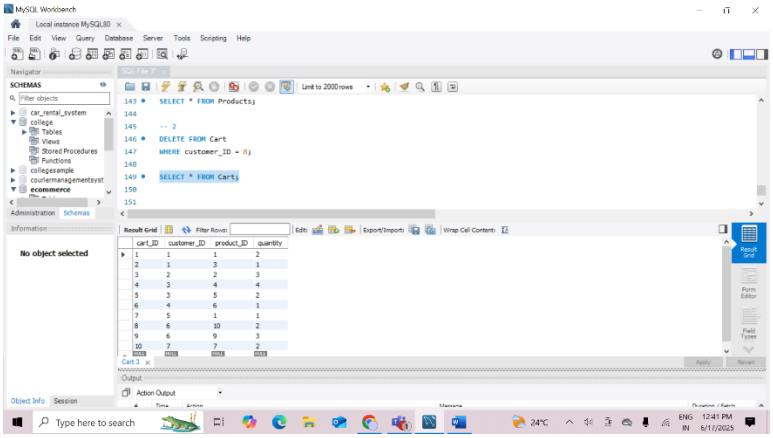
WHERE name = "Refrigerator";



2. Remove all cart items for a specific customer.

DELETE FROM Cart

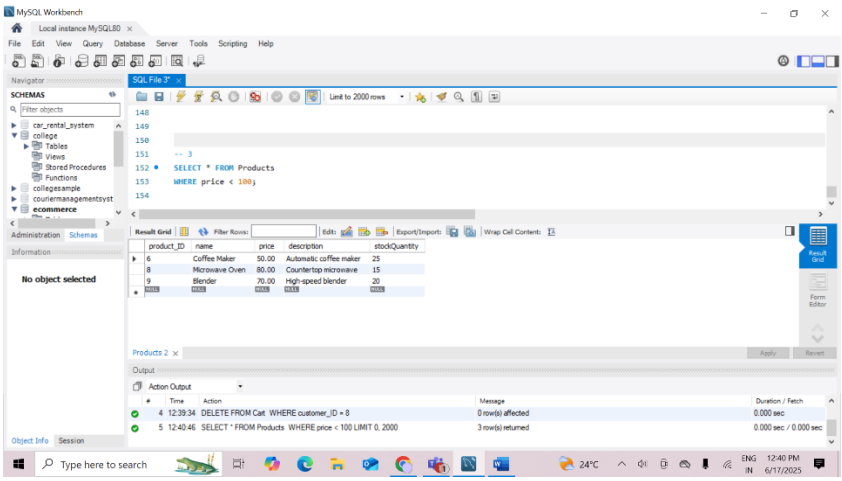
WHERE customer_ID = 8;



3. Retrieve Products Priced Below $100.

SELECT * FROM Products

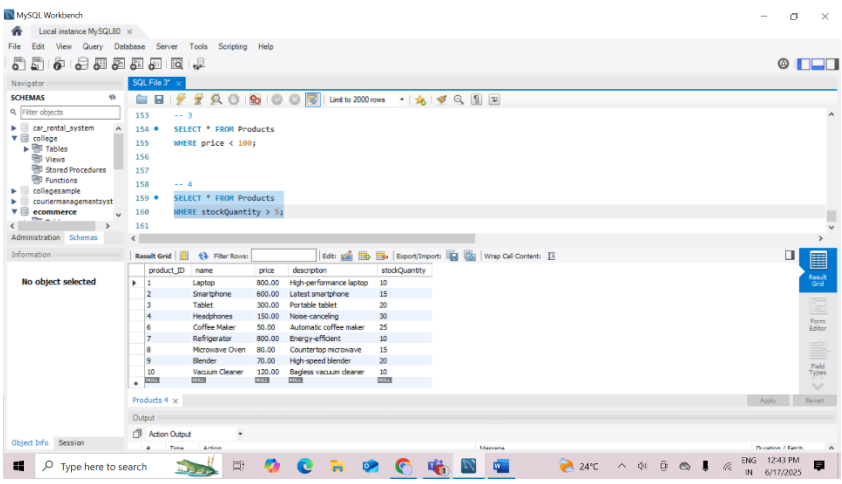
WHERE price < 100;



4. Find Products with Stock Quantity Greater Than 5.

SELECT * FROM Products

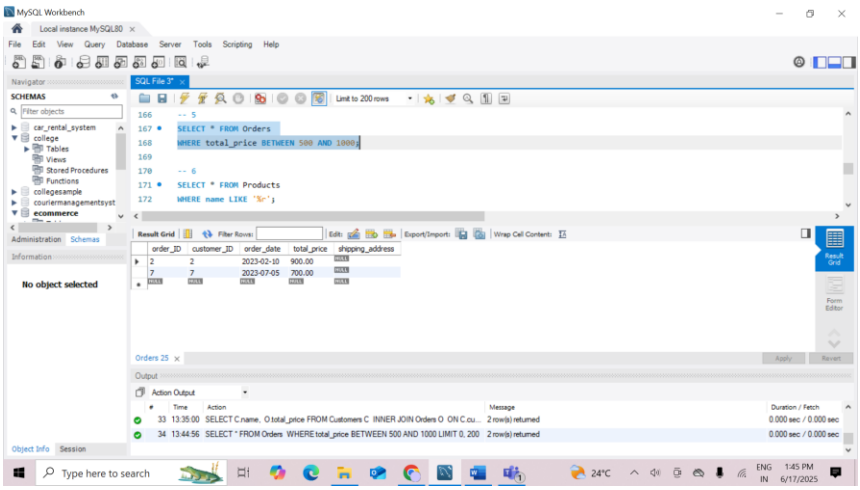
WHERE stockQuantity > 5;



5. Retrieve Orders with Total Amount Between $500 and $1000.

SELECT * FROM Orders

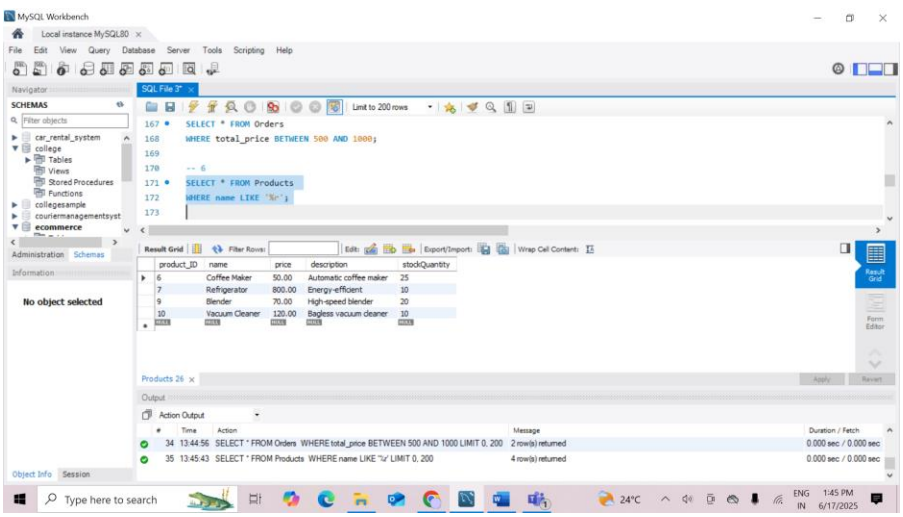
WHERE total_price BETWEEN 500 AND 1000;



6. Find Products which name end with letter ‘r’.

SELECT * FROM Products

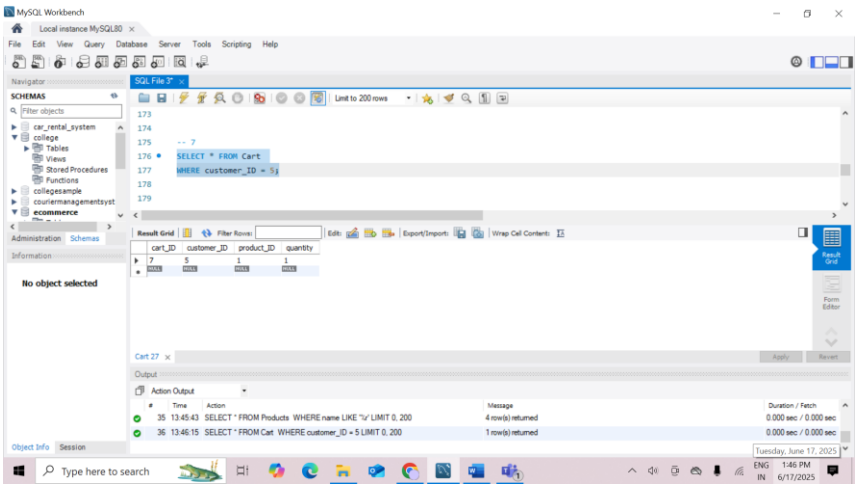
WHERE name LIKE '%r';



7. Retrieve Cart Items for Customer 5.

SELECT * FROM Cart

WHERE customer_ID = 5;



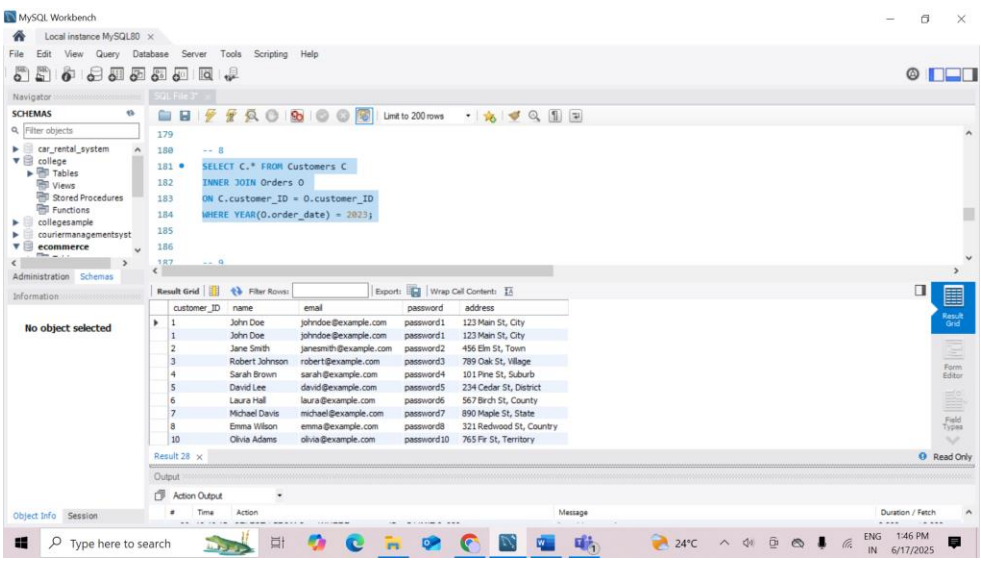
8. Find Customers Who Placed Orders in 2023.

SELECT C.* FROM Customers C

INNER JOIN Orders O

ON C.customer_ID = O.customer_ID

WHERE YEAR(O.order_date) = 2023;



9. Determine the Minimum Stock Quantity for Each Product Category.

10. Calculate the Total Amount Spent by Each Customer.

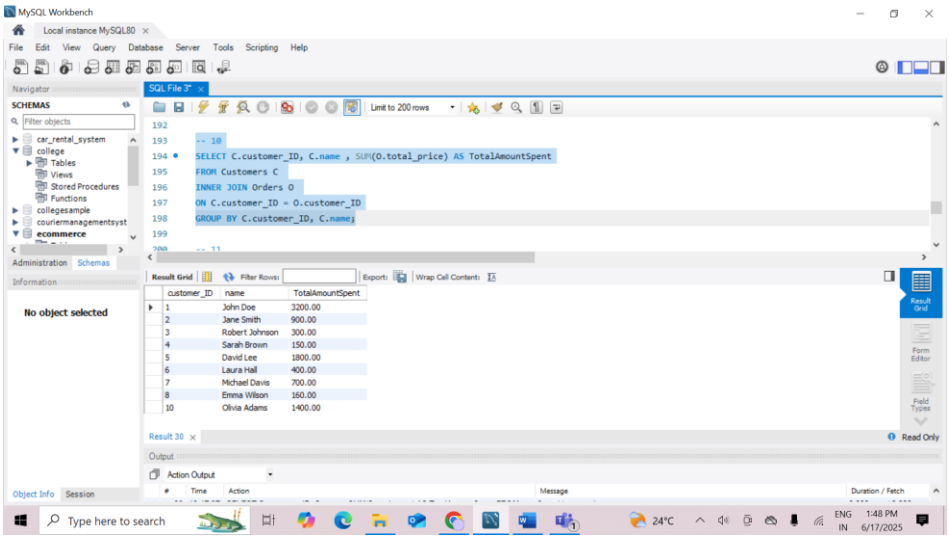
SELECT C.customer_ID, C.name , SUM(O.total_price) AS TotalAmountSpent

FROM Customers C

INNER JOIN Orders O

ON C.customer_ID = O.customer_ID

GROUP BY C.customer_ID, C.name;



11. Find the Average Order Amount for Each Customer.

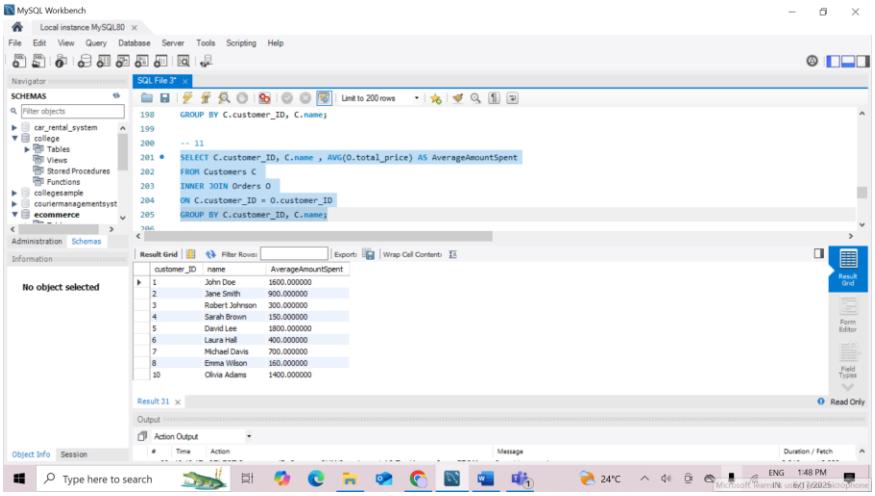
SELECT C.customer_ID, C.name , AVG(O.total_price) AS AverageAmountSpent

FROM Customers C

INNER JOIN Orders O

ON C.customer_ID = O.customer_ID

GROUP BY C.customer_ID, C.name;



12. Count the Number of Orders Placed by Each Customer.

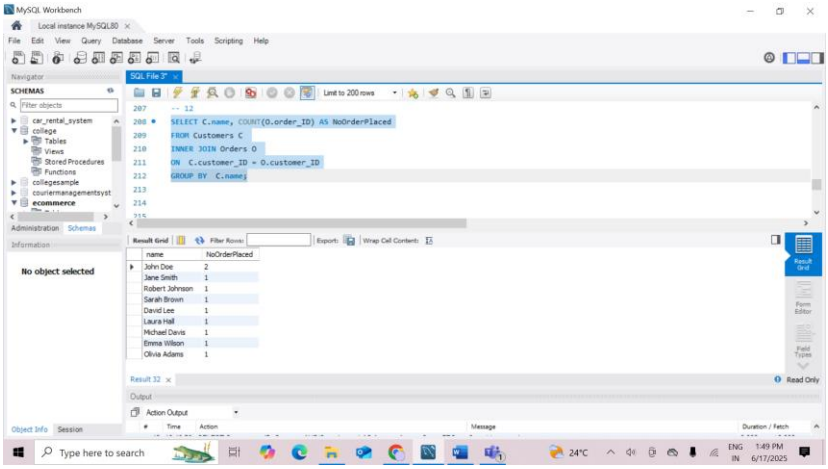
SELECT C.name, COUNT(O.order_ID) AS NoOrderPlaced

FROM Customers C

INNER JOIN Orders O

ON C.customer_ID = O.customer_ID

GROUP BY C.name;



13. Find the Maximum Order Amount for Each Customer.

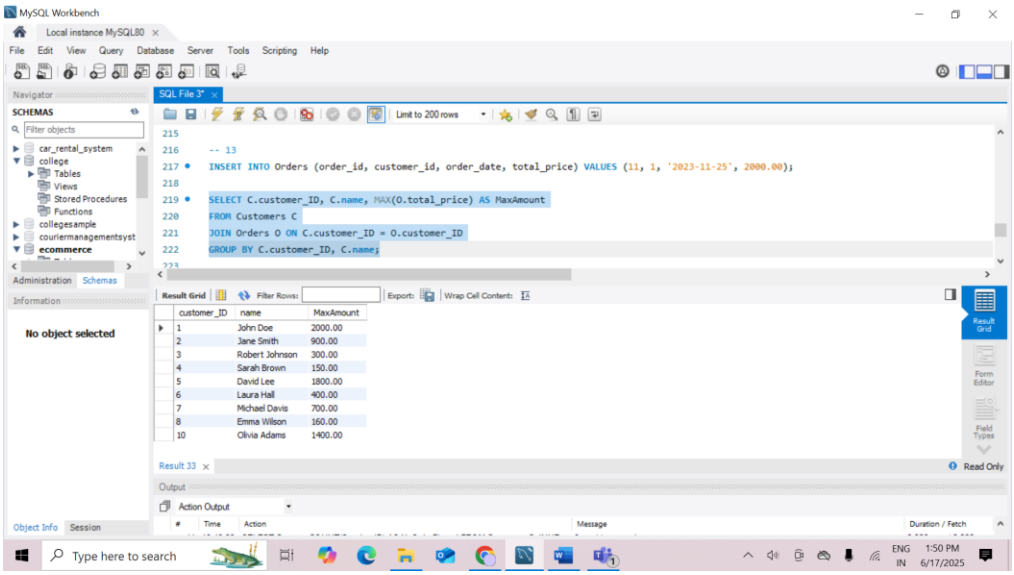
INSERT INTO Orders (order_id, customer_id, order_date, total_price) VALUES (11, 1, '2023-11-25', 2000.00);

SELECT C.customer_ID, C.name, MAX(O.total_price) AS MaxAmount

FROM Customers C

JOIN Orders O ON C.customer_ID = O.customer_ID

GROUP BY C.customer_ID, C.name;



14. Get Customers Who Placed Orders Totaling Over $1000.

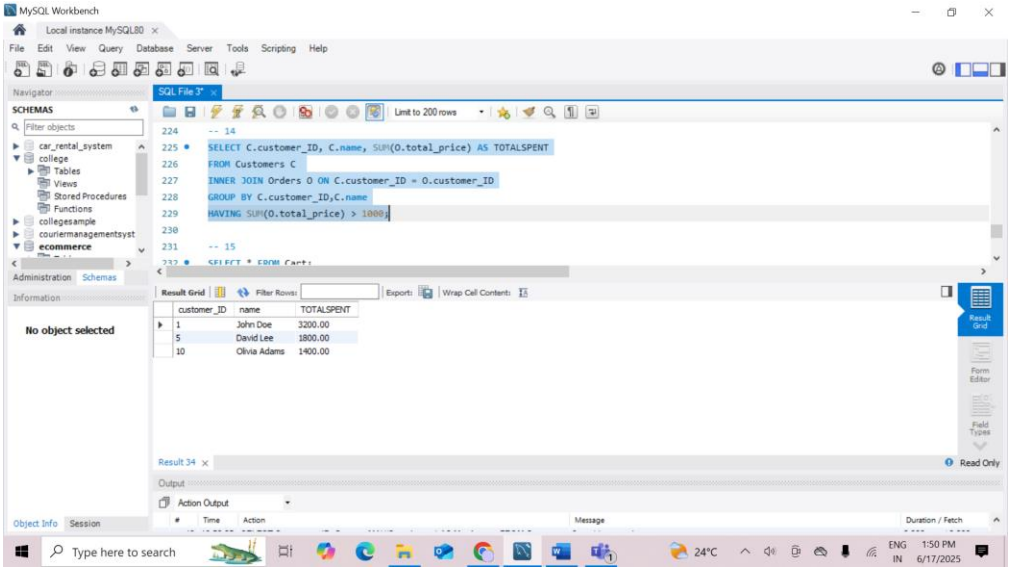
SELECT C.customer_ID, C.name, SUM(O.total_price) AS TOTALSPENT

FROM Customers C

INNER JOIN Orders O ON C.customer_ID = O.customer_ID

GROUP BY C.customer_ID,C.name

HAVING SUM(O.total_price) > 1000;

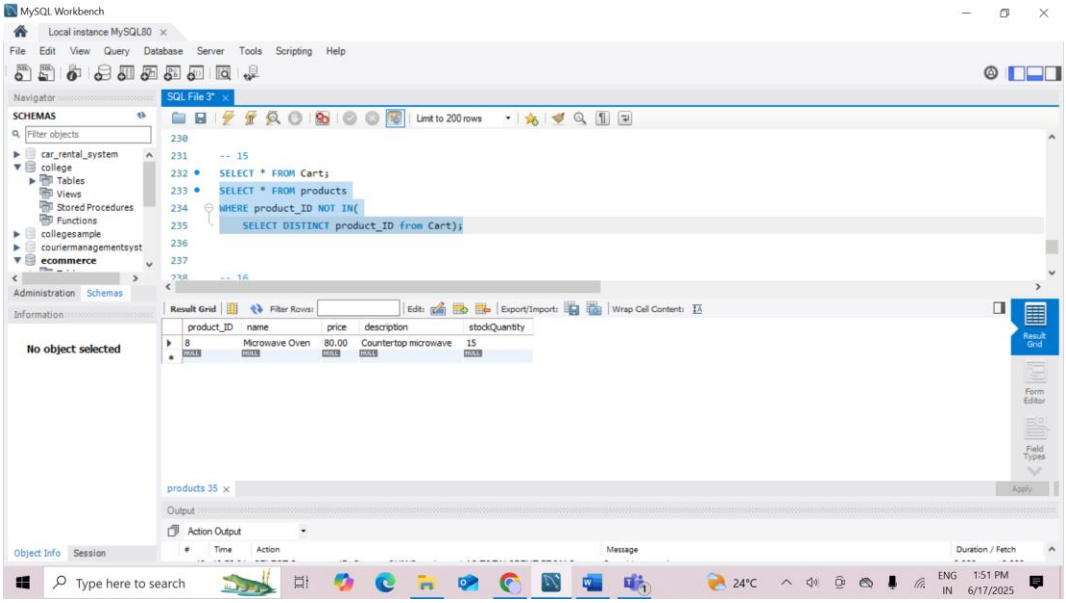


15. Subquery to Find Products Not in the Cart.

SELECT * FROM products

WHERE product_ID NOT IN(

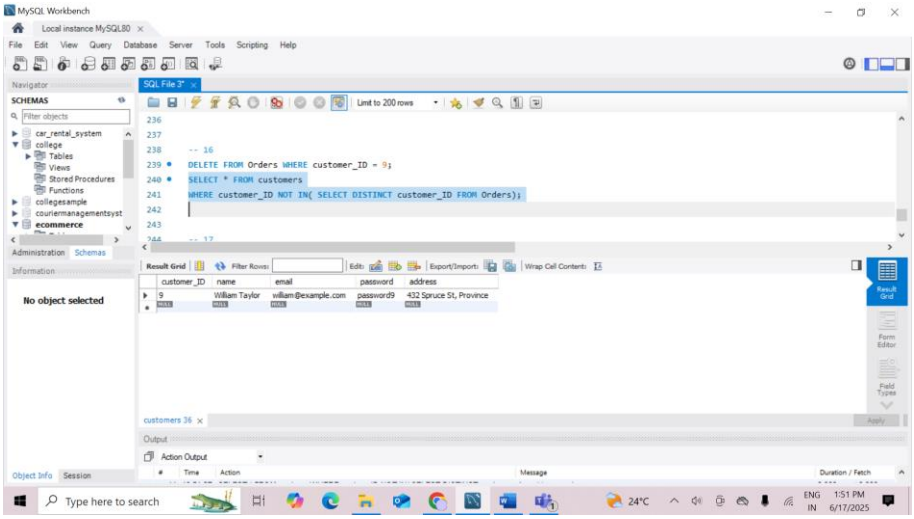
SELECT DISTINCT product_ID from Cart);



16. Subquery to Find Customers Who Haven't Placed Orders.

SELECT * FROM customers

WHERE customer_ID NOT IN(SELECT DISTINCT customer_ID FROM Orders);

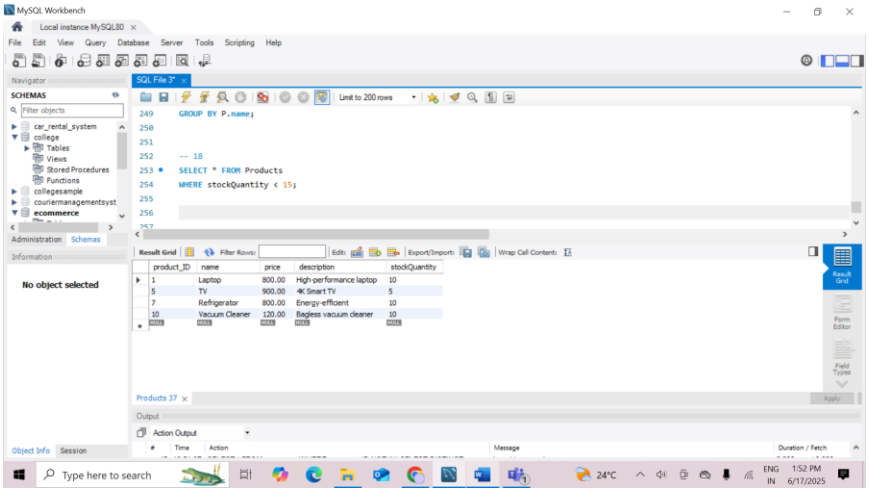


17. Subquery to Calculate the Percentage of Total Revenue for a Product.

```
SELECT P.name,
    SUM( ) * 100 / (SELECT SUM( ) FROM) AS RevenuePercentage
FROM Products P
INNER JOIN
GROUP BY P.name;
```

18. Subquery to Find Products with Low Stock.

```
SELECT * FROM Products
WHERE stockQuantity < 15;
```



19. Subquery to Find Customers Who Placed High-Value Orders.

```
SELECT C.name,  O.total_price

FROM Customers C

INNER JOIN Orders O

ON C.customer_ID = O.customer_ID

WHERE O.total_price > 1500;
```