SAFERTECK BACKEND SQL QUERIES ALONG OUTPUT SCREEN SHOTS

Name: M.Mani Deep

Id:2100030898

1.Creating of database:



2.Creation and Insertion of Customer table:

-- > Create Customers table

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

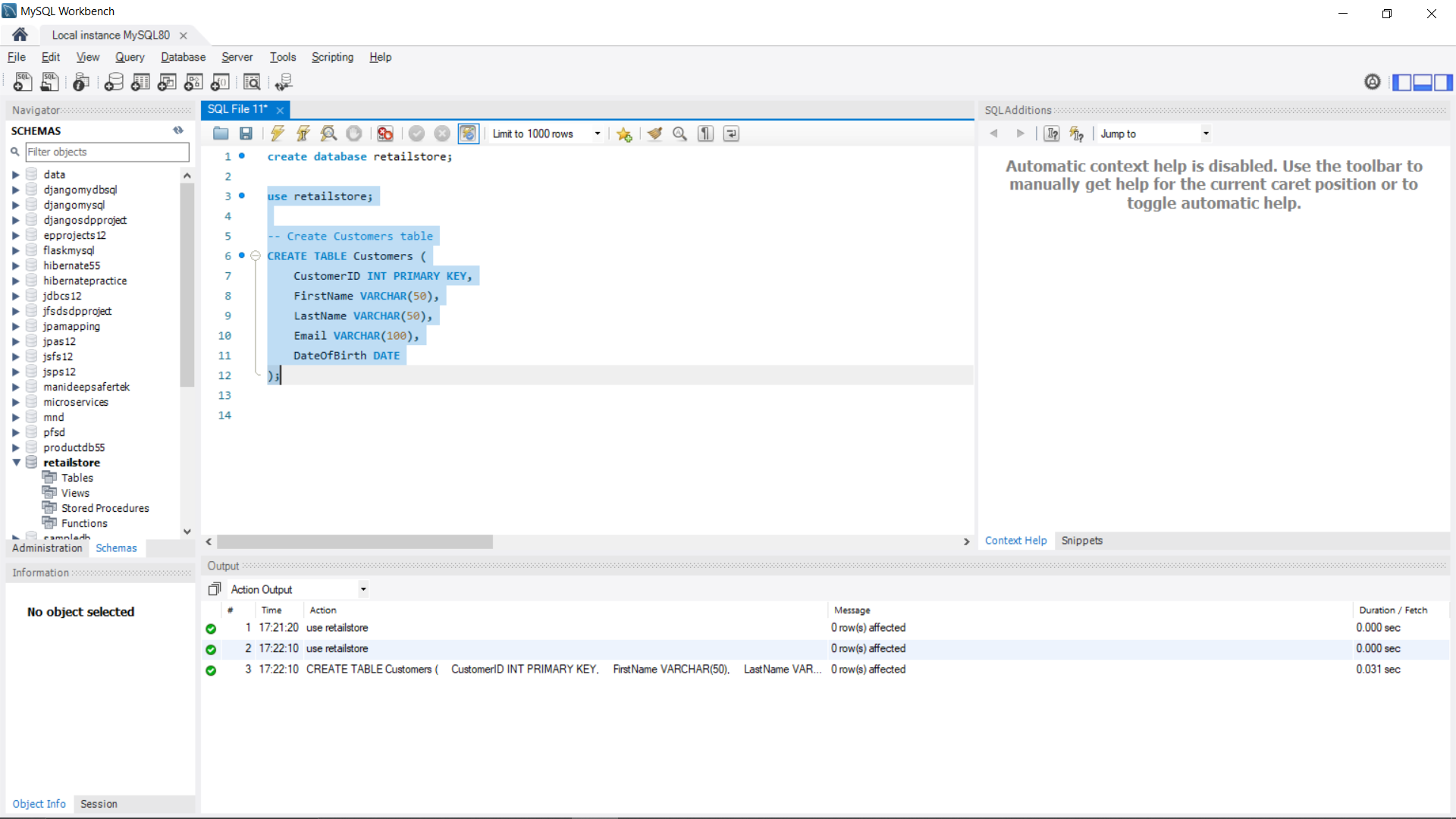
FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

DateOfBirth DATE

);



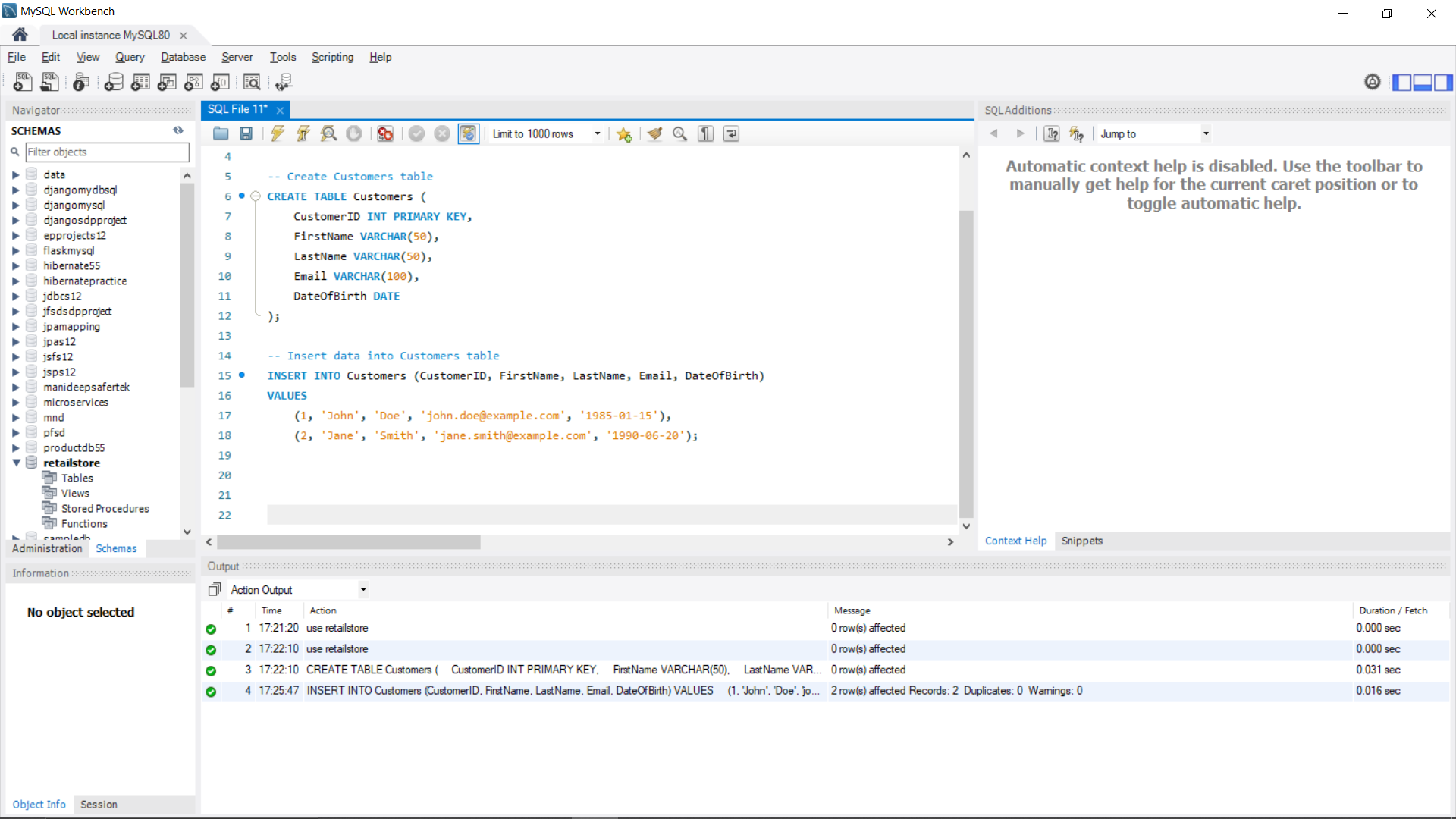
-- Insert data into Customers table

INSERT INTO Customers (CustomerID, FirstName, LastName, Email, DateOfBirth)

VALUES

(1, 'John', 'Doe', 'john.doe@example.com', '1985-01-15'),

(2, 'Jane', 'Smith', 'jane.smith@example.com', '1990-06-20');



3.Creation and Insertion of Product table:

-- Create Products table

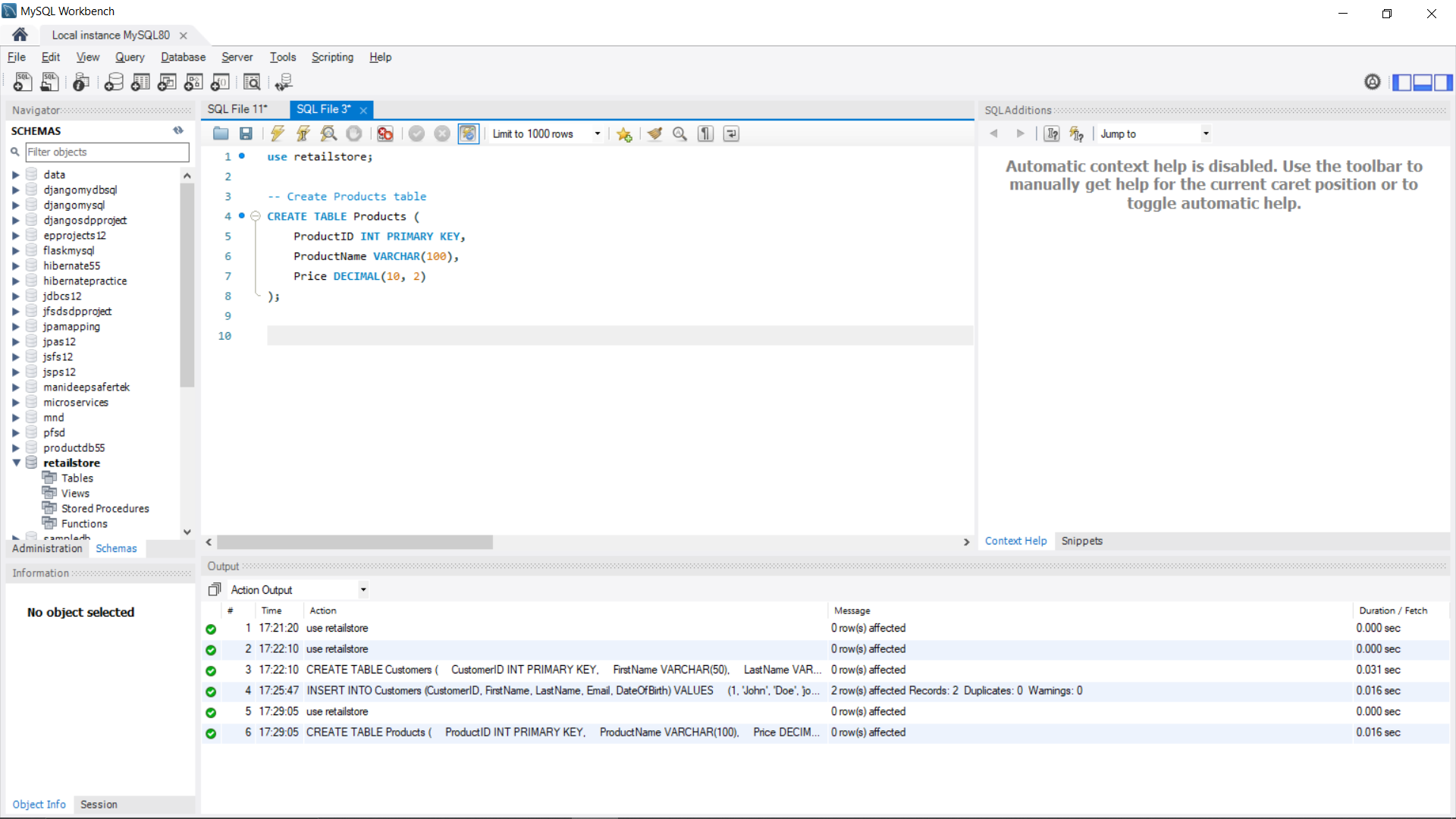
CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Price DECIMAL(10, 2)

);



-- Insert data into Products table

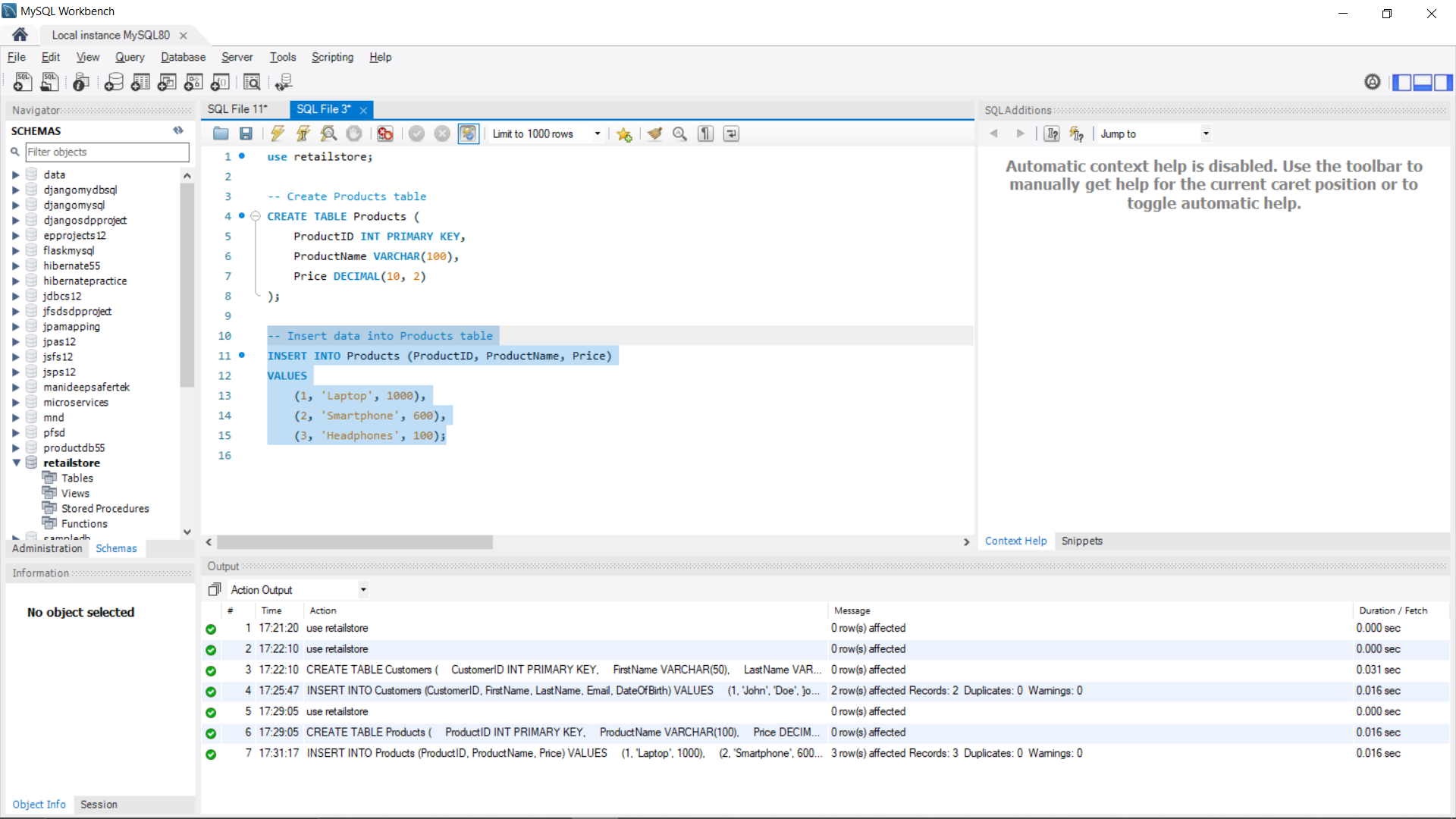
INSERT INTO Products (ProductID, ProductName, Price)

VALUES

(1, 'Laptop', 1000),

(2, 'Smartphone', 600),

(3, 'Headphones', 100);



4.Creation and Insertion of Orders table:

-- Create Orders table

CREATE TABLE Orders (

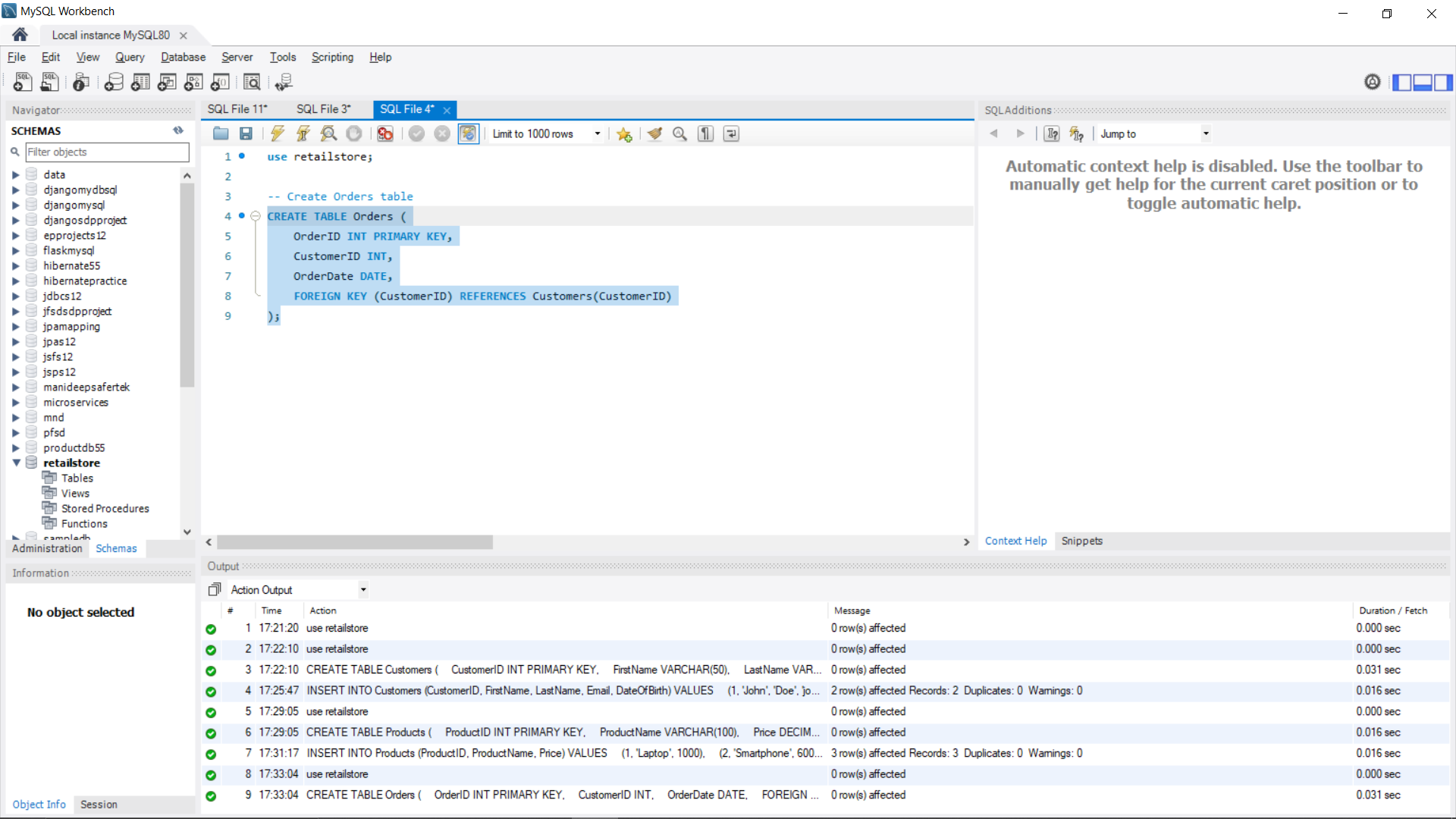
OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);



-- Insert data into Orders table

INSERT INTO Orders (OrderID, CustomerID, OrderDate)

VALUES

(1, 1, '2023-01-10'),

(2, 2, '2023-01-12');



5.Creation and Insertion of OrderItems table:

-- Create OrderItems table

CREATE TABLE OrderItems (

OrderItemID INT PRIMARY KEY,

OrderID INT,

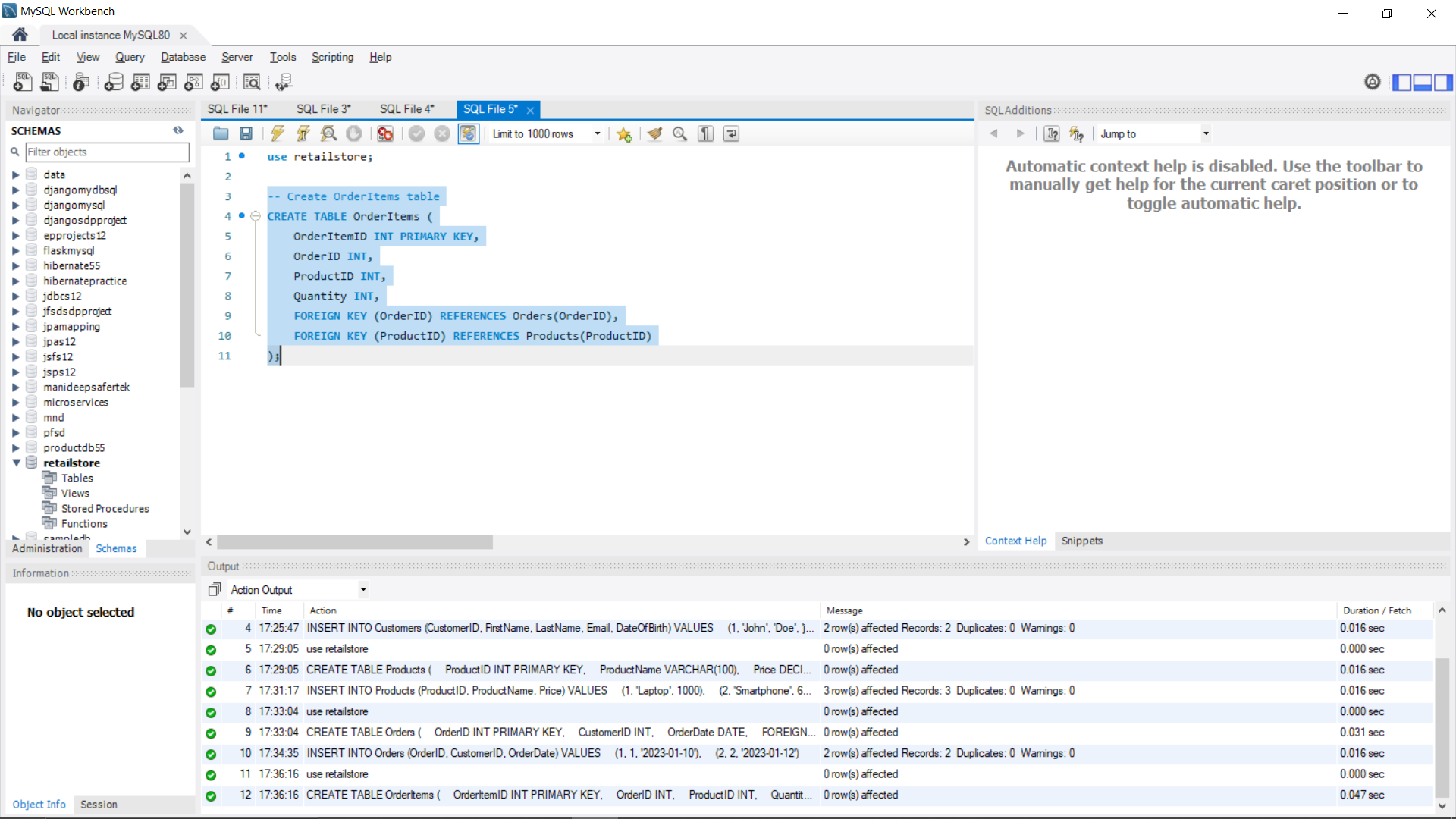
ProductID INT,

Quantity INT,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);



-- Insert data into OrderItems table

INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity)

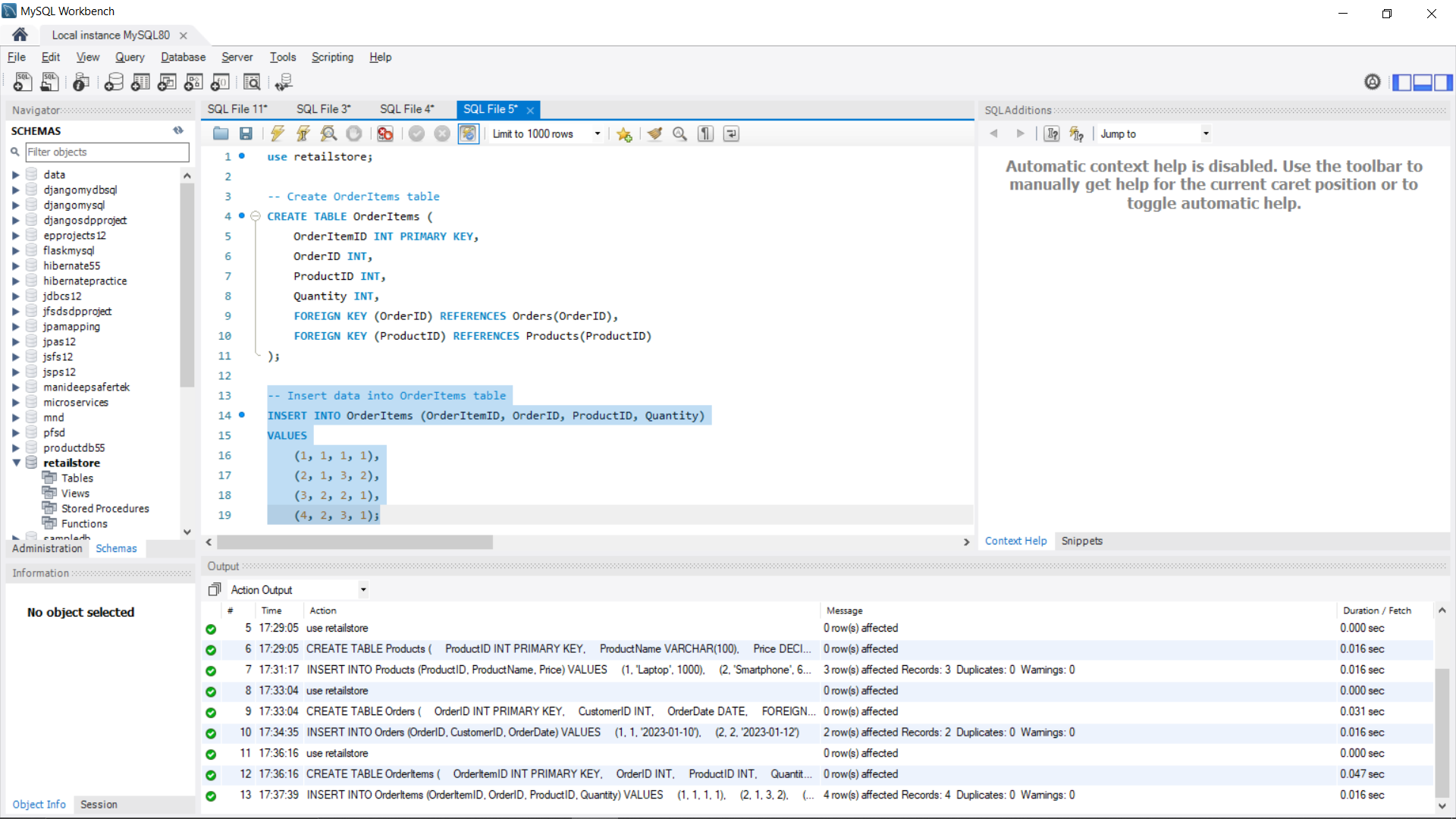
VALUES

(1, 1, 1, 1),

(2, 1, 3, 2),

(3, 2, 2, 1),

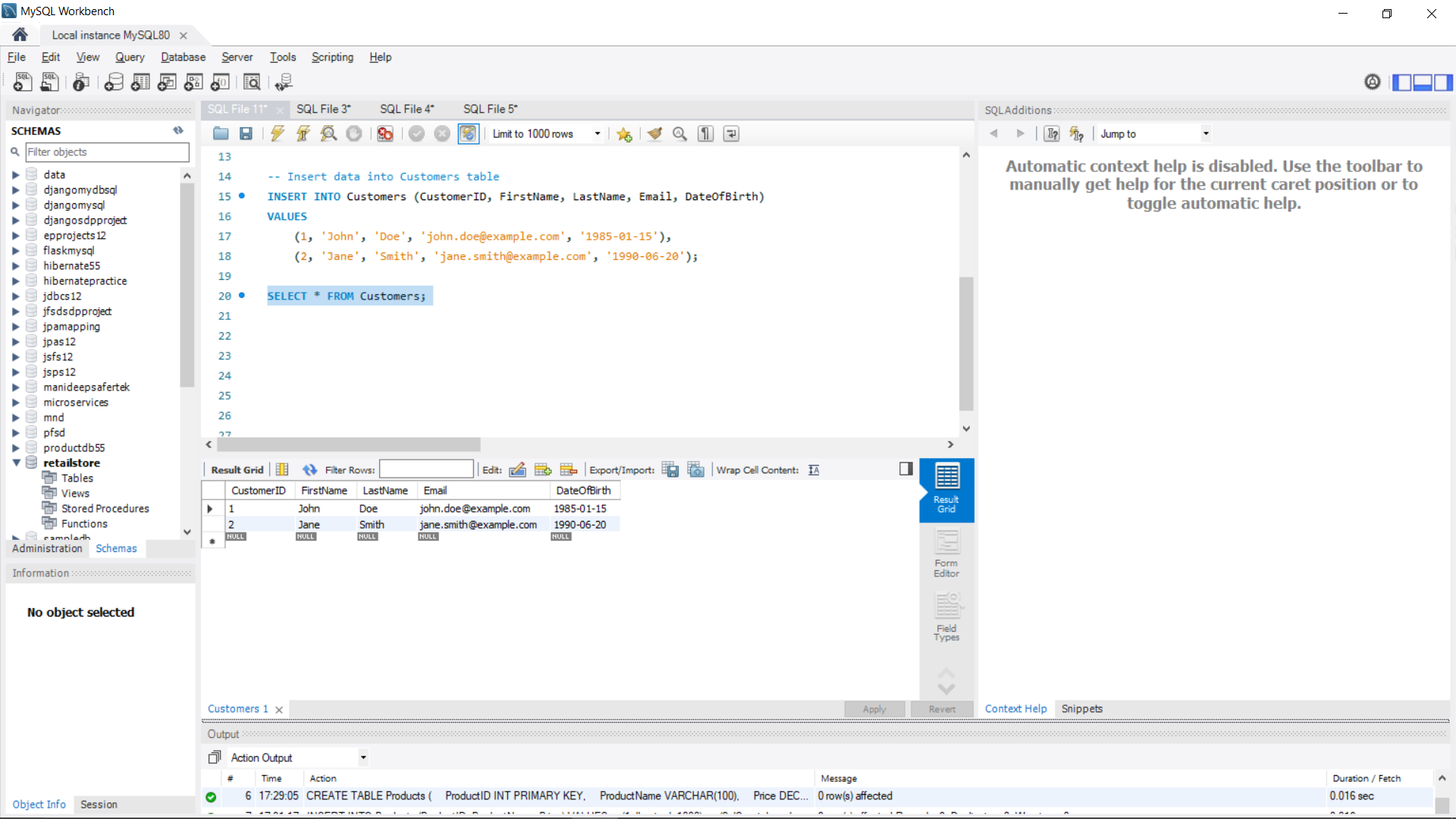
(4, 2, 3, 1);



QUERIES:

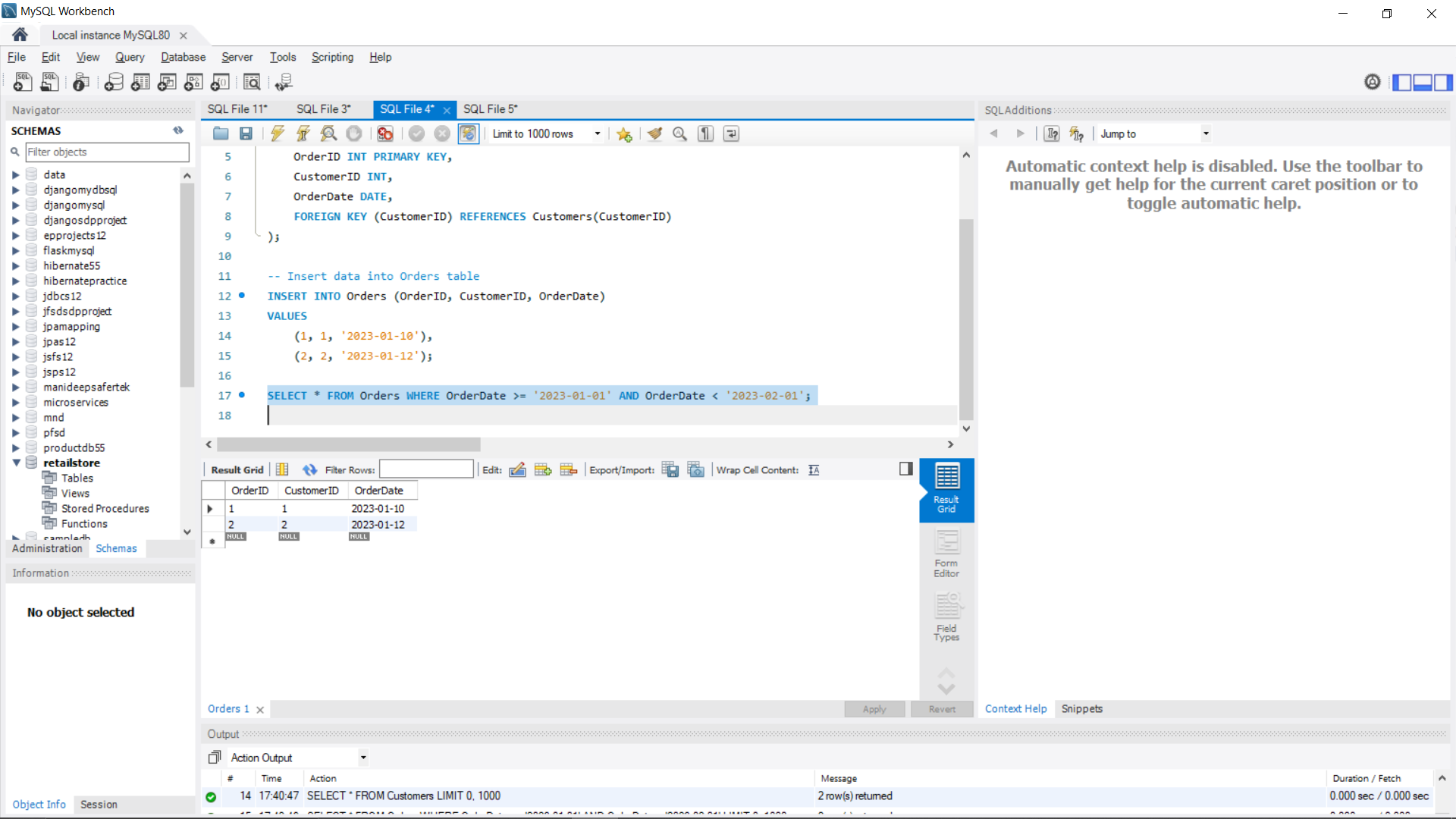
1. List all customers.

SELECT \* FROM Customers;



1. Find all orders placed in January 2023.

SELECT \* FROM Orders WHERE OrderDate >= '2023-01-01' AND OrderDate < '2023-02-01';

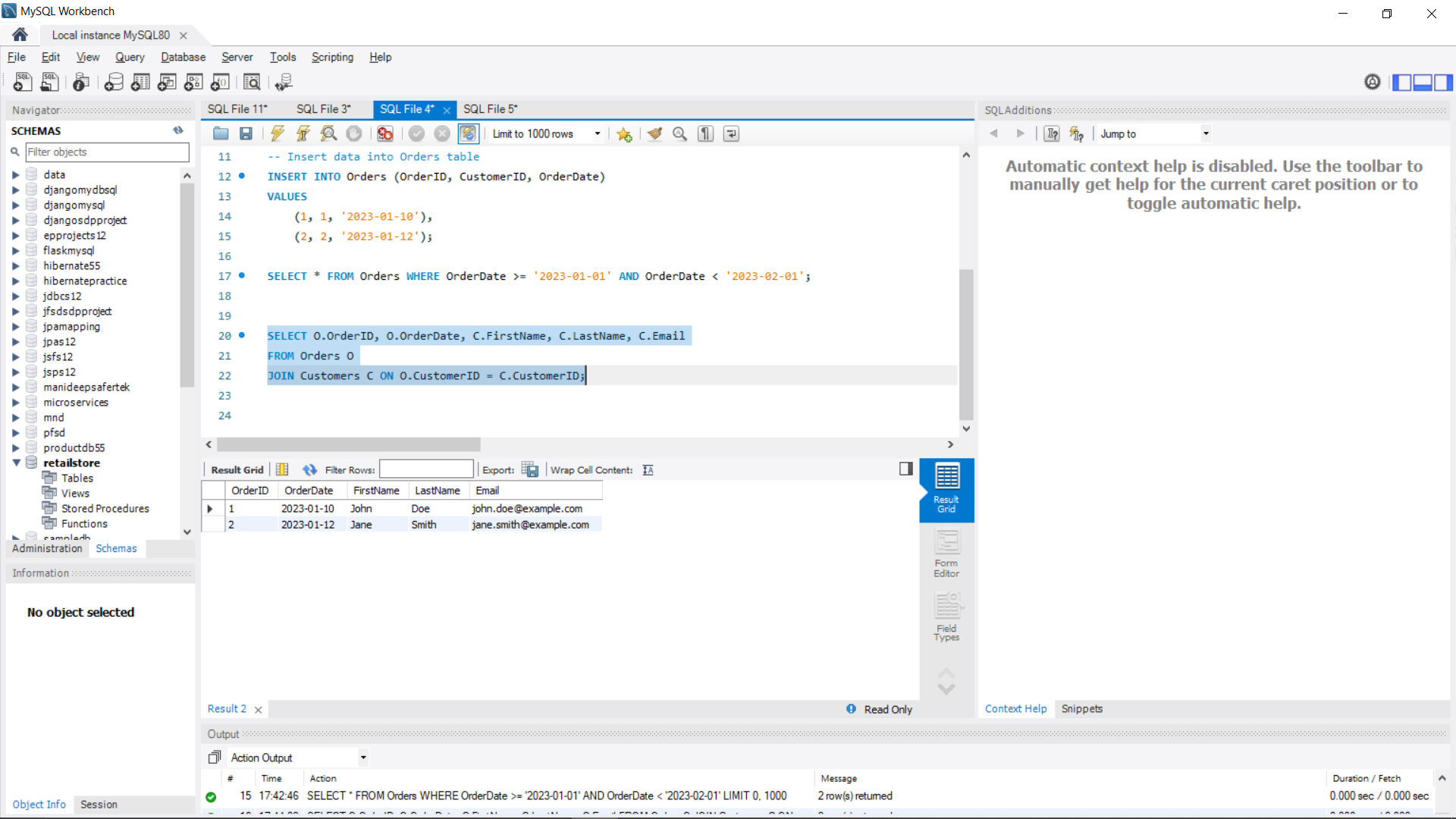


3. Get the details of each order, including the customer name and email.

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

FROM Orders O

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



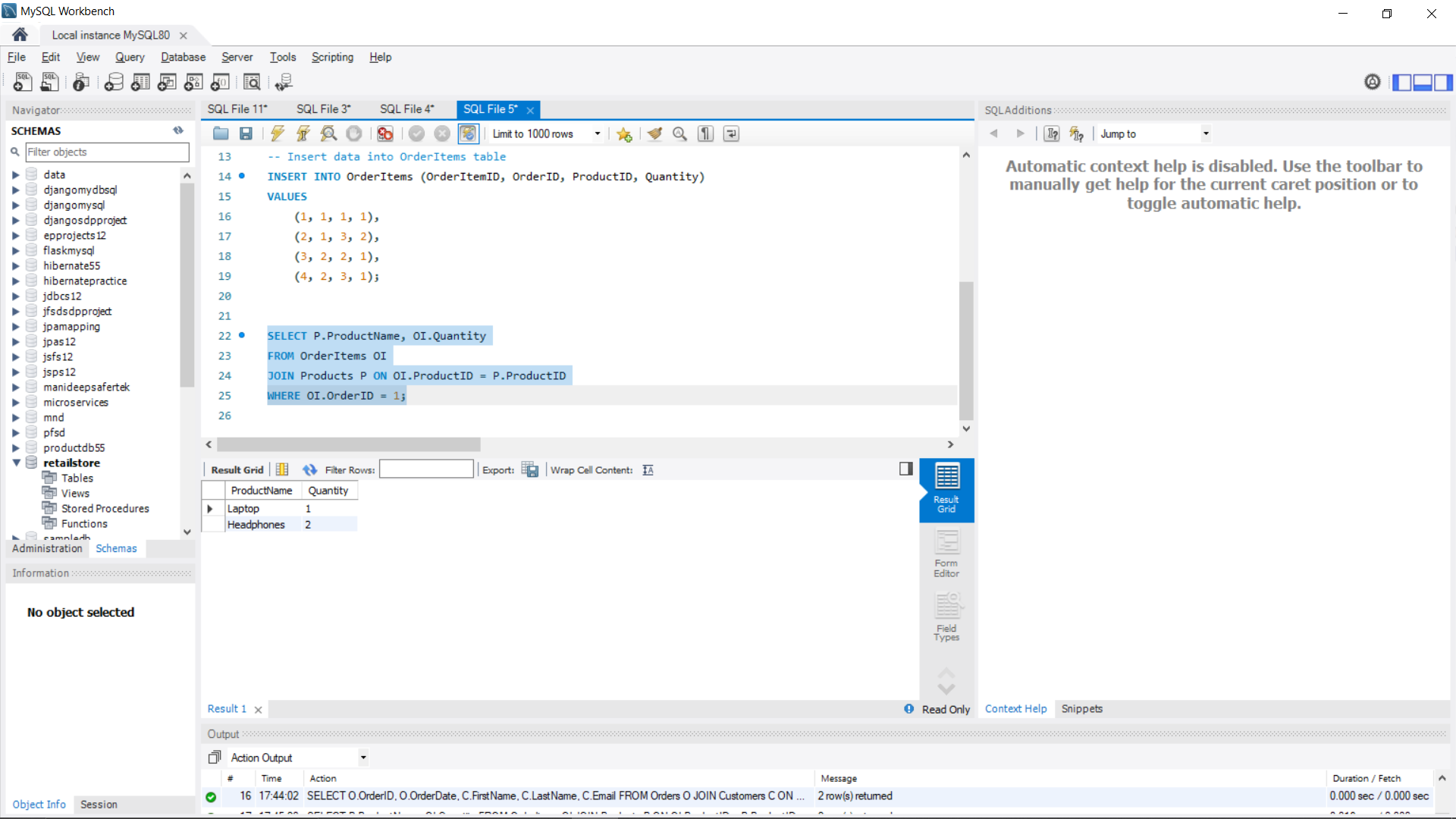
4.List the products purchased in a specific order (e.g., OrderID = 1).

SELECT P.ProductName, OI.Quantity

FROM OrderItems OI

JOIN Products P ON OI.ProductID = P.ProductID

WHERE OI.OrderID = 1;



5. Calculate the total amount spent by each customer.

SELECT C.FirstName, C.LastName, SUM(P.Price \* OI.Quantity) AS TotalSpent

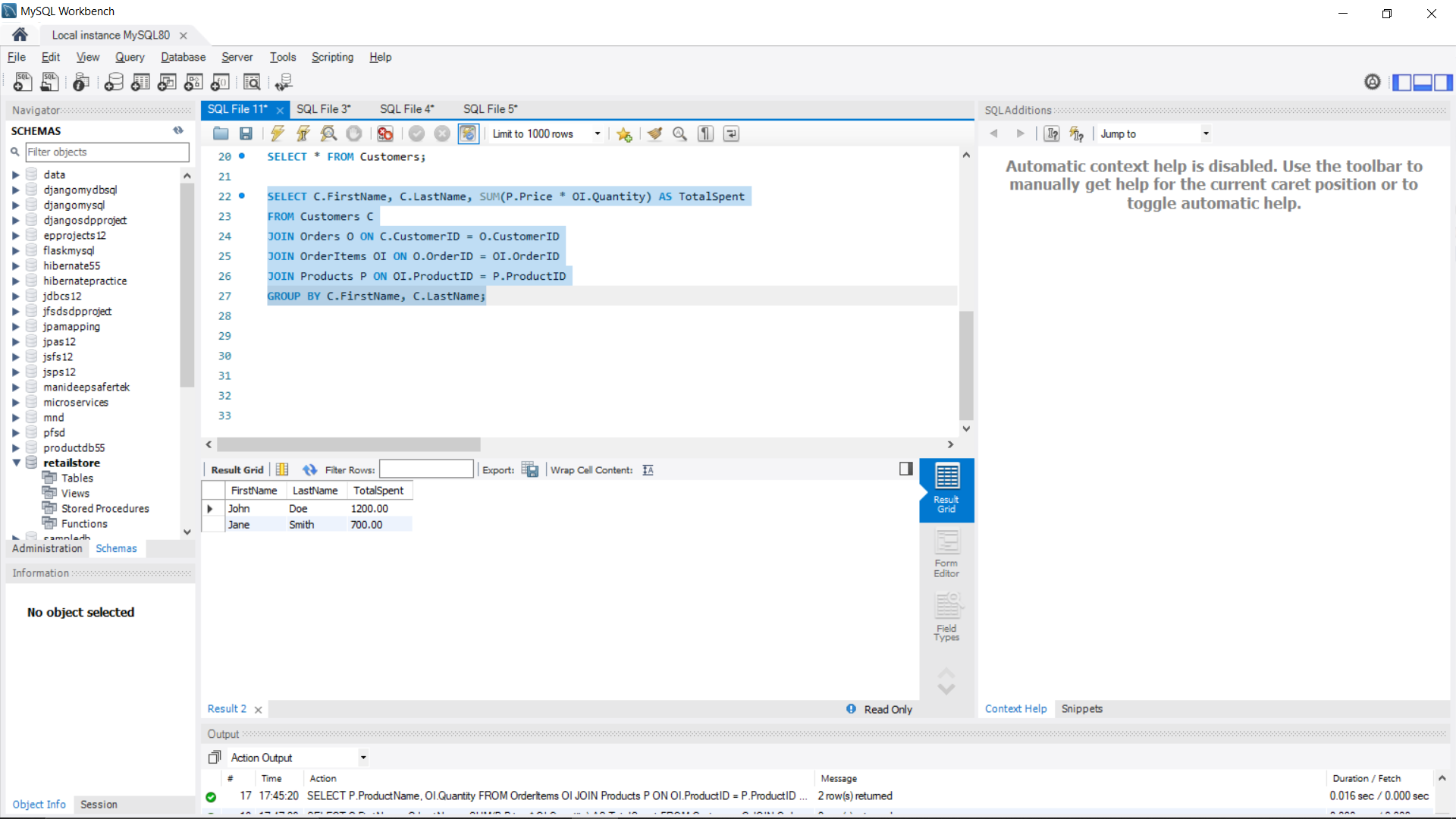
FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

JOIN OrderItems OI ON O.OrderID = OI.OrderID

JOIN Products P ON OI.ProductID = P.ProductID

GROUP BY C.FirstName, C.LastName;



6. Find the most popular product (the one that has been ordered the most).

SELECT P.ProductName, SUM(OI.Quantity) AS TotalOrdered

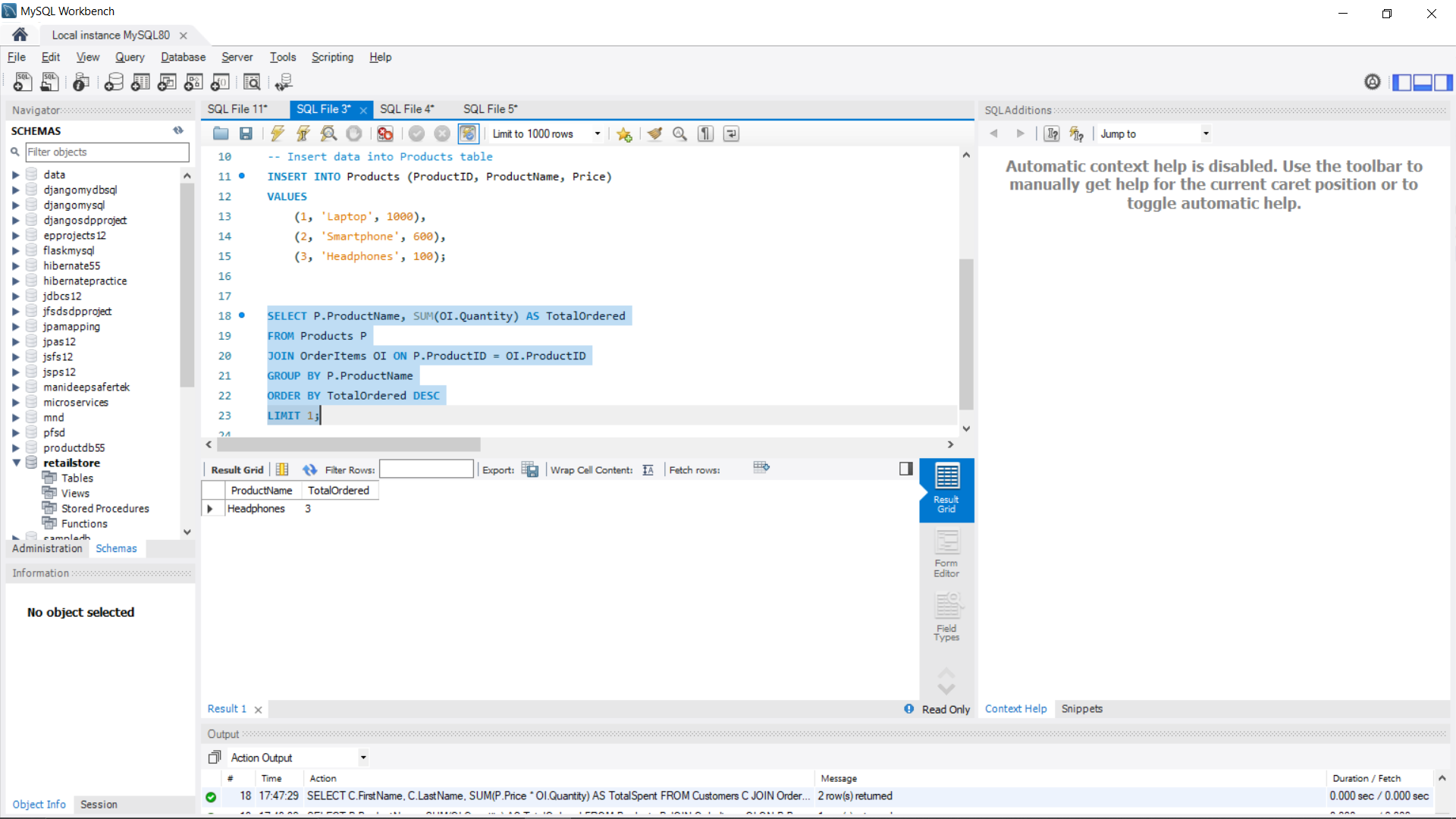
FROM Products P

JOIN OrderItems OI ON P.ProductID = OI.ProductID

GROUP BY P.ProductName

ORDER BY TotalOrdered DESC

LIMIT 1;



7. Get the total number of orders and the total sales amount for each month in 2023.

SELECT

EXTRACT(MONTH FROM OrderDate) AS Month,

COUNT(\*) AS TotalOrders,

SUM(P.Price \* OI.Quantity) AS TotalSalesAmount

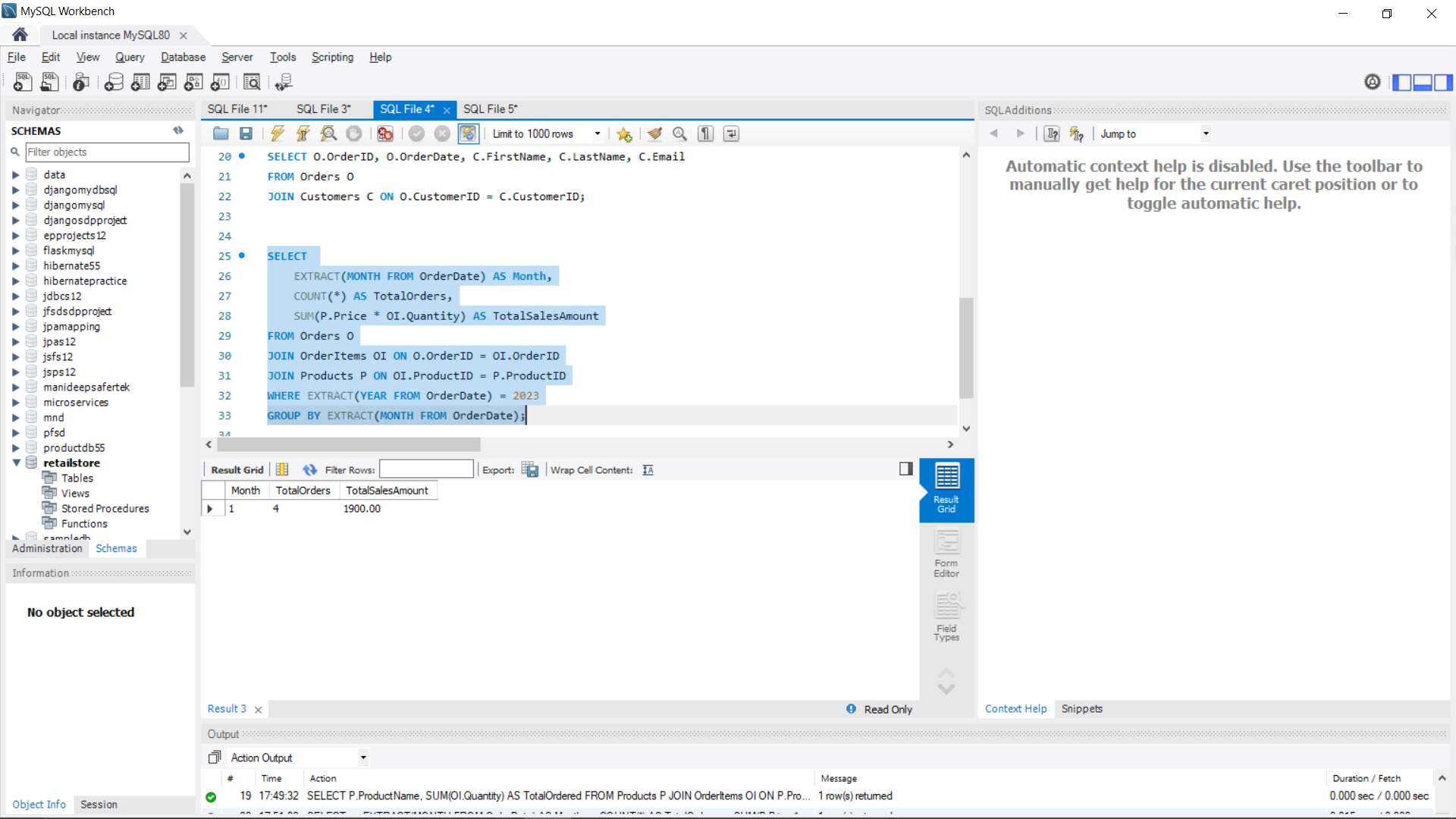
FROM Orders O

JOIN OrderItems OI ON O.OrderID = OI.OrderID

JOIN Products P ON OI.ProductID = P.ProductID

WHERE EXTRACT(YEAR FROM OrderDate) = 2023

GROUP BY EXTRACT(MONTH FROM OrderDate);



8.Find customers who have spent more than $1000.

SELECT C.FirstName, C.LastName

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

JOIN OrderItems OI ON O.OrderID = OI.OrderID

JOIN Products P ON OI.ProductID = P.ProductID

GROUP BY C.FirstName, C.LastName

HAVING SUM(P.Price \* OI.Quantity) > 1000;

