SQL ASSIGNMENT

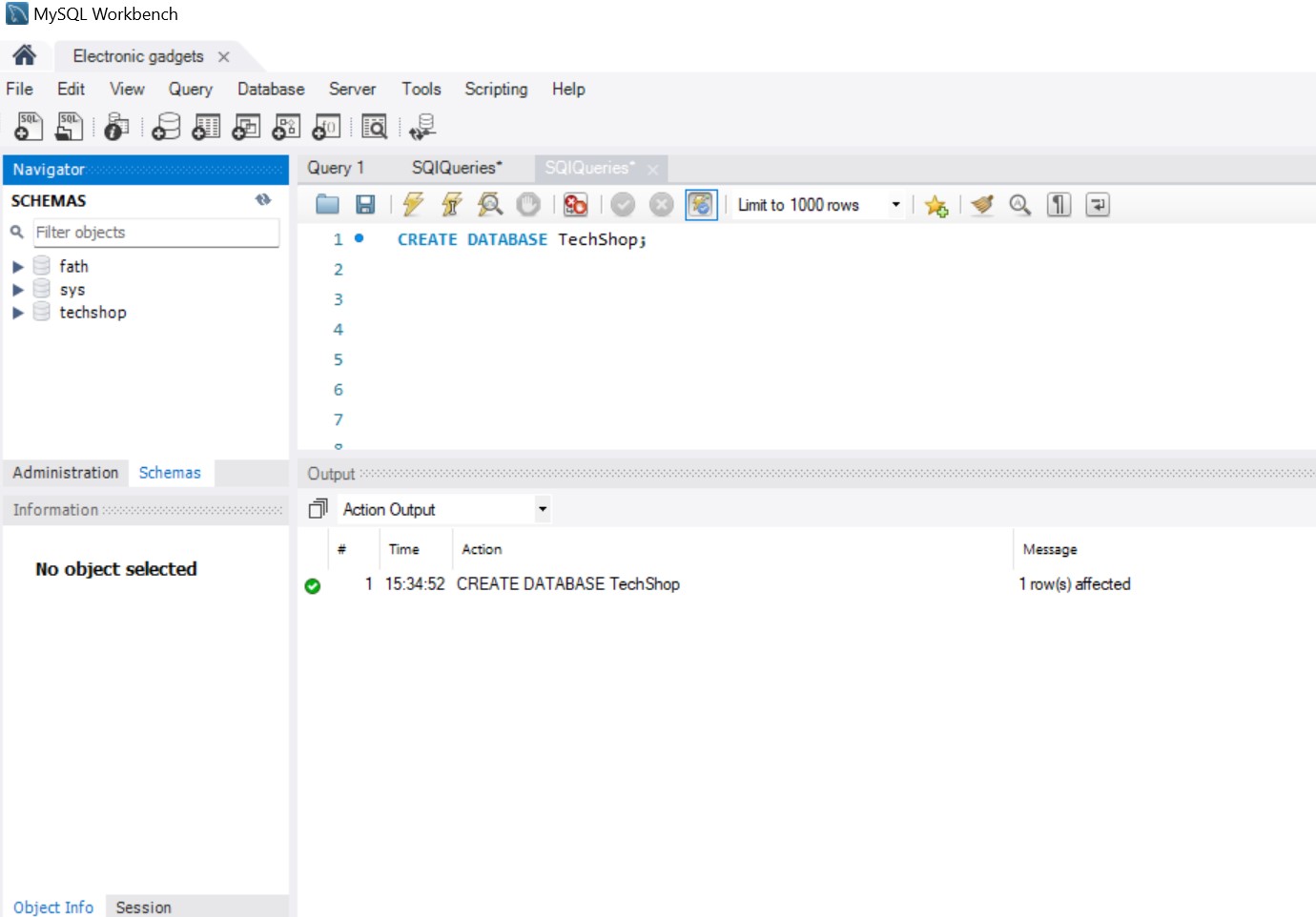
Task:1. Database Design:

1. **Create the database named "TechShop"**

**QUERY:**

CREATE DATABASE TechShop;

USE TechShop;



1. **Define the schema for the Customers, Products, Orders, OrderDetails and Inventory tables based on the provided schema.**

**and**

**4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.**

**QUERY:**

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY AUTO\_INCREMENT,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

Phone VARCHAR(20),

Address VARCHAR(255)

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY AUTO\_INCREMENT,

ProductName VARCHAR(100),

Description TEXT,

Price DECIMAL(10, 2)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT,

OrderDate DATE,

TotalAmount DECIMAL(10, 2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY AUTO\_INCREMENT,

OrderID INT,

ProductID INT,

Quantity INT,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

CREATE TABLE Inventory (

InventoryID INT PRIMARY KEY AUTO\_INCREMENT,

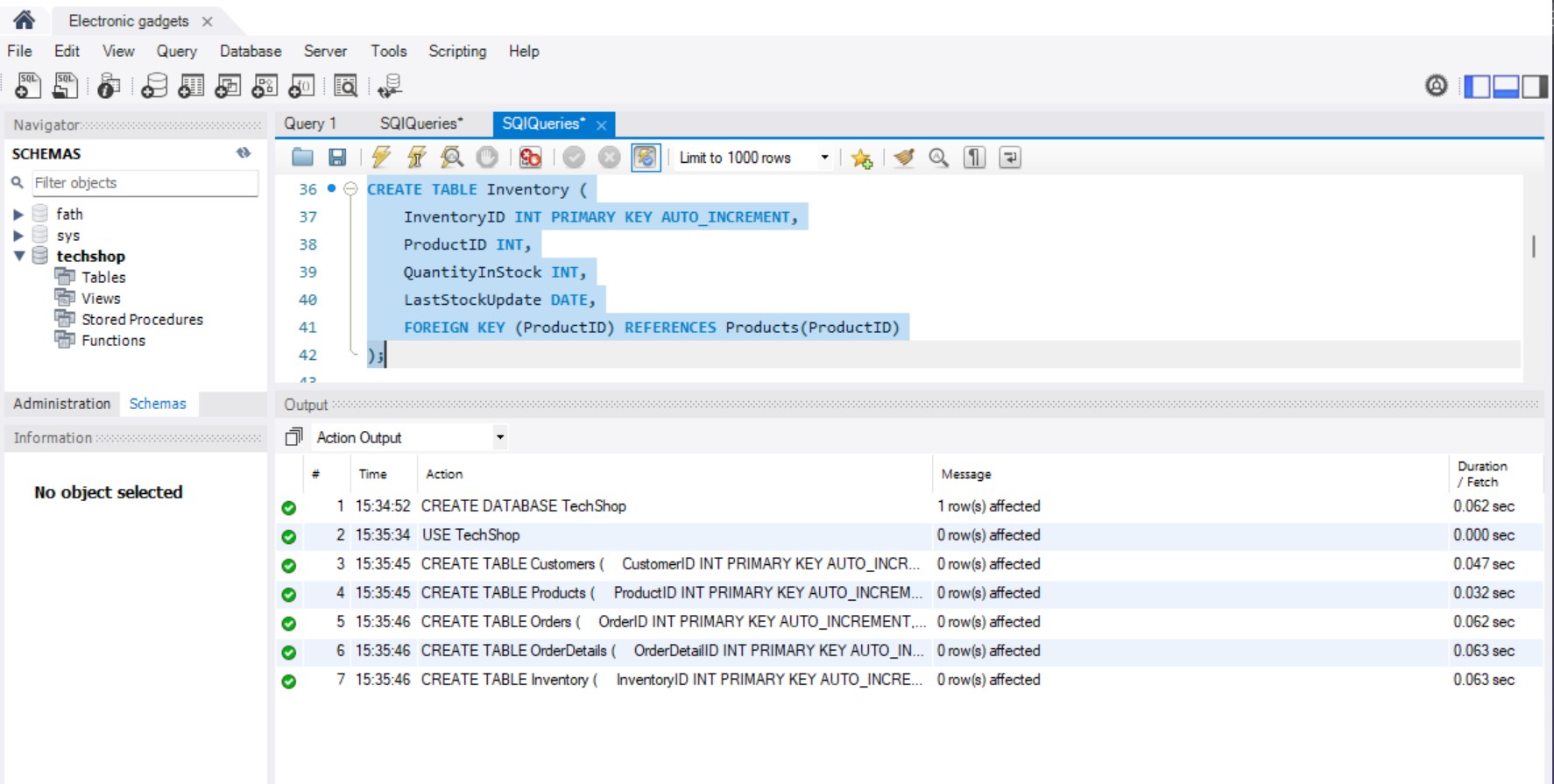
ProductID INT,

QuantityInStock INT,

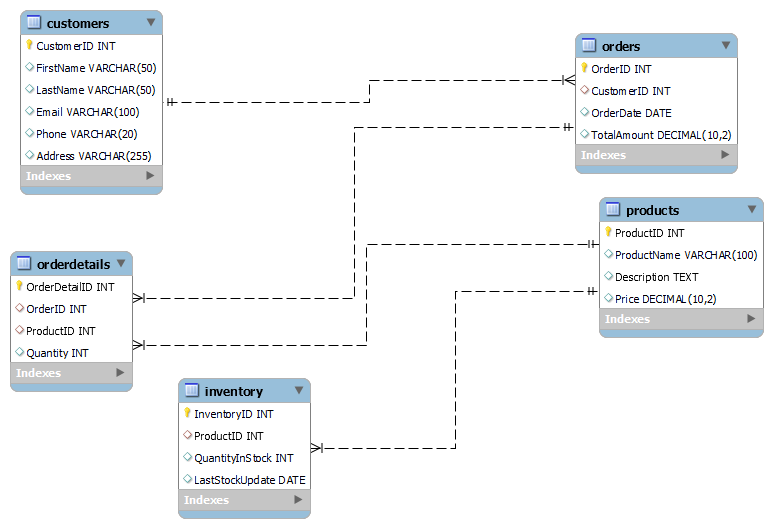
LastStockUpdate DATE,

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);



**3. Create an ERD (Entity Relationship Diagram) for the database.**



1. **Insert at least 10 sample records into each of the following tables.**
2. **Customers**
3. **Products**
4. **Orders**
5. **OrderDetails**

**e. Inventory**

**QUERY**

INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone, Address)

VALUES

(1, 'Johnson', 'Doe', 'doejohn@email.com', '1230987632', '123 Main St'),

(2, 'Jane', 'Smith', 'smith@email.com', '9876543210', '456 Elm St'),

(3, 'Michael', 'John', 'michaelJ@email.com', '1234567890', '789 Oak St'),

(4, 'Emily', 'Brown', 'emilyBrown@email.com', '2345678901', '101 Pine St'),

(5, 'David', 'Wilson', 'wilson@email.com', '4567890123', '234 Maple St'),

(6, 'Sarah', 'Warner', 'sarah@email.com', '7890123456', '567 Cedar St'),

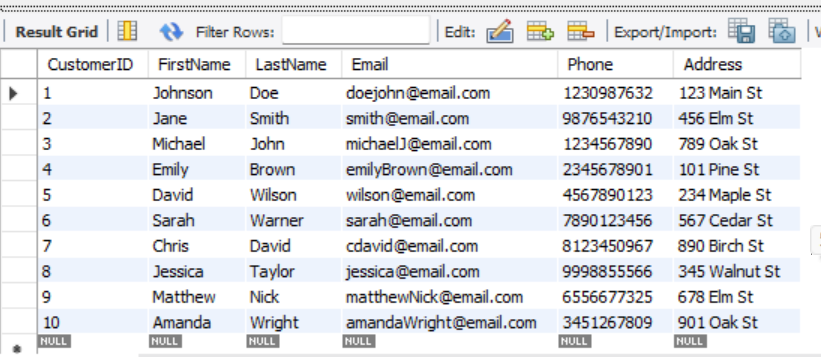
(7, 'Chris', 'David', 'cdavid@email.com', '8123450967', '890 Birch St'),

(8, 'Jessica', 'Taylor', 'jessica@email.com', '9998855566', '345 Walnut St'),

(9, 'Matthew', 'Nick', 'matthewNick@email.com', '6556677325', '678 Elm St'),

(10, 'Amanda', 'Wright', 'amandaWright@email.com', '3451267809', '901 Oak St');

**RESULT:**



INSERT INTO Products (ProductID, ProductName, Description, Price) VALUES

(1, 'Laptop', 'High-performance laptop with SSD storage', 999.99),

(2, 'Smartphone', 'Latest smartphone model with OLED display', 699.99),

(3, 'Tablet', 'Portable tablet device with long battery life', 399.99),

(4, 'Headphones', 'Wireless noise-canceling headphones with 30-hour battery', 149.99),

(5, 'Smartwatch', 'Fitness and health tracking smartwatch with heart rate monitor',299.99),

(6, 'Camera', 'Digital camera with advanced features and 4K video recording', 599.99),

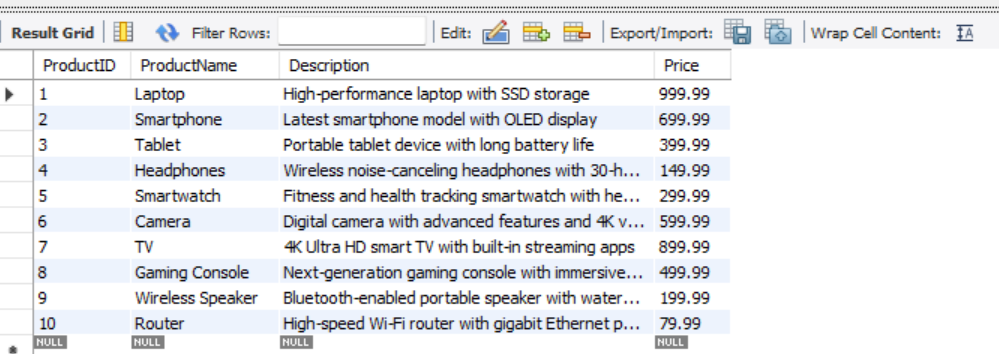
(7, 'TV', '4K Ultra HD smart TV with built-in streaming apps', 899.99),

(8, 'Gaming Console', 'Next-generation gaming console with immersive graphics', 499.99),

(9, 'Wireless Speaker', 'Bluetooth-enabled portable speaker with waterproof design', 199.99),

(10, 'Router', 'High-speed Wi-Fi router with gigabit Ethernet ports', 79.99);

**RESULT:**



INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)

VALUES

(1, 1, '2023-01-03', 999.99),

(2, 2, '2023-02-01', 699.99),

(3, 3, '2024-02-09', 399.99),

(4, 4, '2024-03-17', 149.99),

(5, 5, '2024-04-16', 299.99),

(6, 6, '2024-04-21', 599.99),

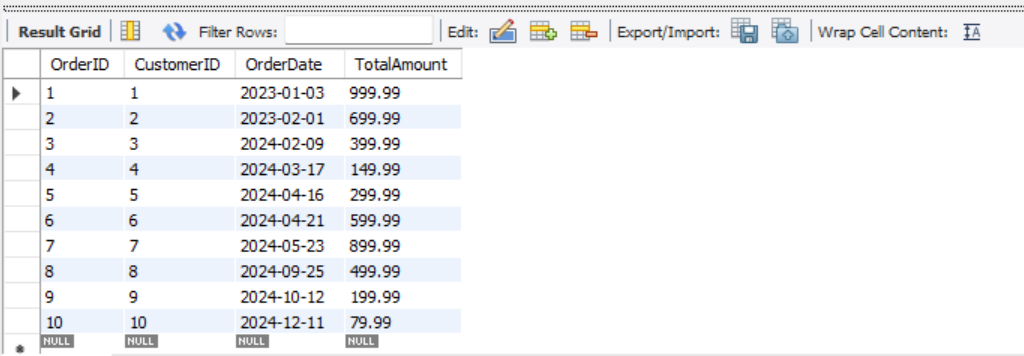
(7, 7, '2024-05-23', 899.99),

(8, 8, '2024-09-25', 499.99),

(9, 9, '2024-10-12', 199.99),

(10, 10, '2024-12-11', 79.99);

**RESULT:**



INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity) VALUES

(1, 1, 1, 1),

(2, 2, 2, 1),

(3, 3, 3, 2),

(4, 4, 4, 3),

(5, 5, 5, 1),

(6, 6, 6, 1),

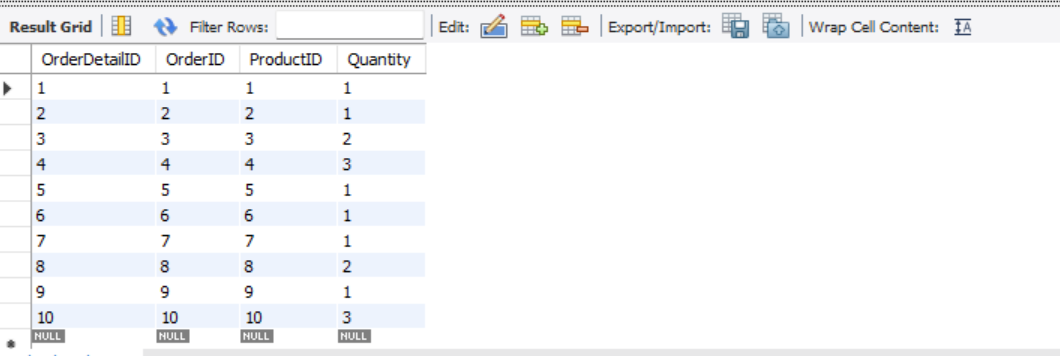
(7, 7, 7, 1),

(8, 8, 8, 2),

(9, 9, 9, 1),

(10, 10, 10, 3);

**RESULT:**



INSERT INTO Inventory (InventoryID, ProductID, QuantityInStock, LastStockUpdate) VALUES

(101, 1, 10, '2024-01-20'),

(102, 2, 20, '2024-01-20'),

(103, 3, 15, '2024-01-20'),

(144, 4, 5, '2024-01-20'),

(105, 5, 25, '2024-01-20'),

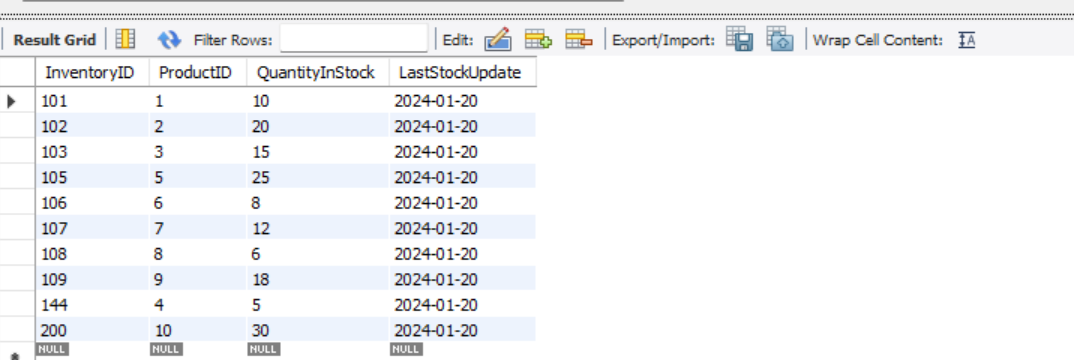
(106, 6, 8, '2024-01-20'),

(107, 7, 12, '2024-01-20'),

(108, 8, 6, '2024-01-20'),

(109, 9, 18, '2024-01-20'),

(200, 10, 30, '2024-01-20');

**RESULT:**

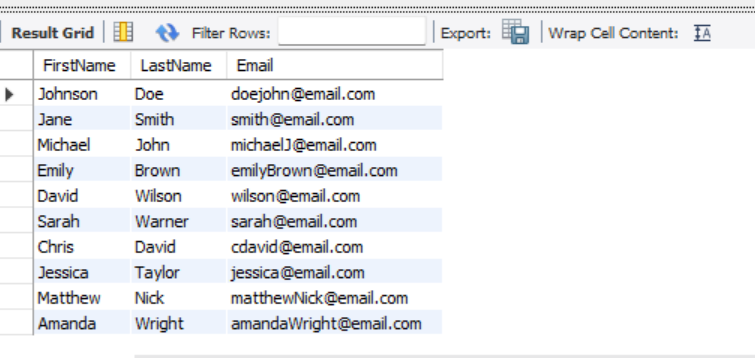
Tasks 2: Select, Where, Between, AND, LIKE:

**1. Write an SQL query to retrieve the names and emails of all customers.**

**QUERY**

SELECT FirstName, LastName, Email FROM Customers;

**RESULT:**



**2. Write an SQL query to list all orders with their order dates and corresponding customer names.**

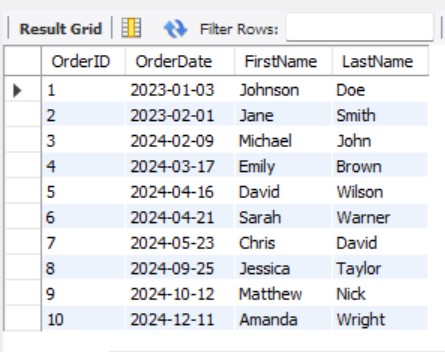
**QUERY:**

SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName

FROM Orders

INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

**RESULT:**



1. **Write an SQL query to insert a new customer record into the "Customers" table. Include customer information such as name, email, and address.**

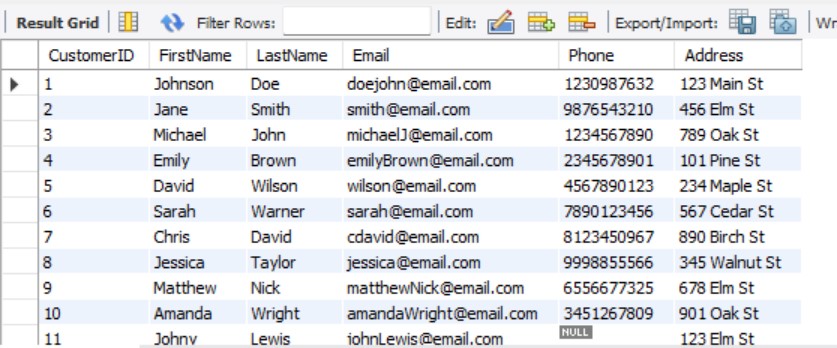
**QUERY:**

INSERT INTO Customers (FirstName, LastName, Email, Address)

VALUES ('Johny', 'Lewis', 'johnLewis@email.com', '123 Elm St');

Select \* from Customers;

**RESULT:**



ALTER TABLE Products MODIFY Price DECIMAL(12, 2);

1. **Write an SQL query to update the prices of all electronic gadgets in the "Products" table by increasing them by 10%.**

**QUERY:**

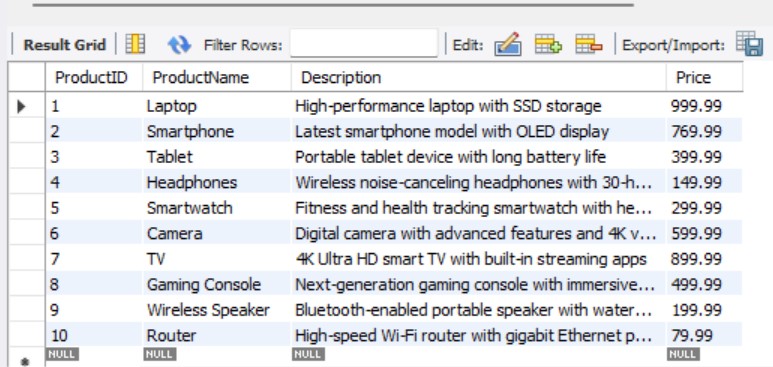
UPDATE Products

SET Price = Price \* 1.10

WHERE ProductID=2;

select \* from Products;

**RESULT:**



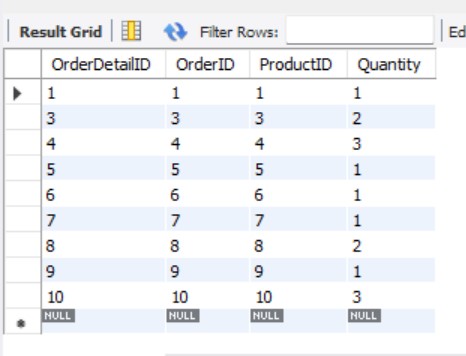
**5.Write an SQL query to delete a specific order and its associated order details from the "Orders" and "OrderDetails" tables. Allow users to input the order ID as a parameter.**

**QUERY:**

DELETE FROM OrderDetails WHERE OrderID = 2;

select \* from OrderDetails;

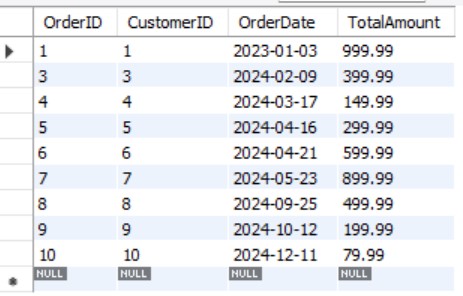
**RESULT:**



DELETE FROM Orders WHERE OrderID = 2;

select \* from Orders;

**RESULT:**



**6.Write an SQL query to insert a new order into the "Orders" table. Include the customer ID, order date, and any other necessary information.**

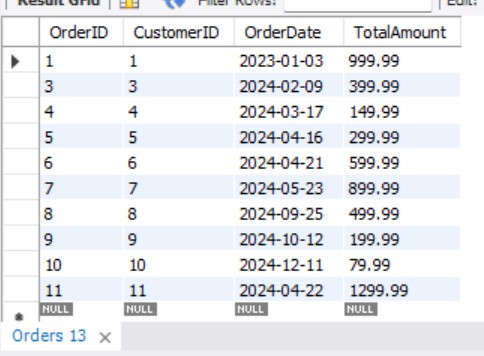
**QUERY:**

INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)

VALUES(11,11, '2024-04-22', 1299.99);

select \* from Orders;

**RESULT:**



**7.Write an SQL query to update the contact information (e.g., email and address) of a specific customer in the "Customers" table. Allow users to input the customer ID and new contact information.**

**QUERY:**

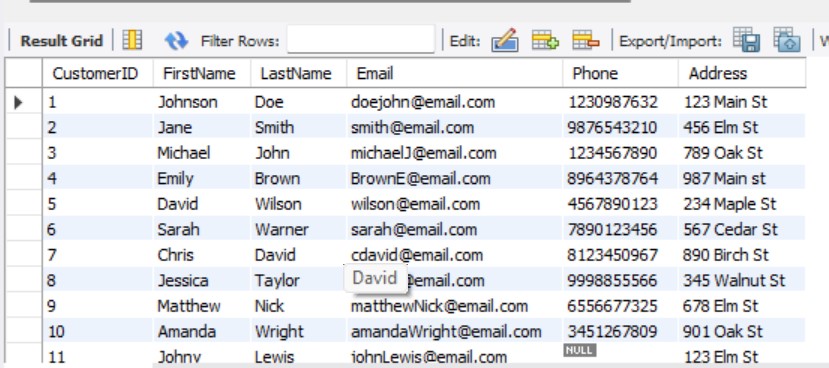
UPDATE Customers

SET Email = 'BrownE@email.com',Phone='8964378764',Address = '987 Main st'

WHERE CustomerID = 4;

select \* from Customers;

**RESULT:**



1. **Write an SQL query to recalculate and update the total cost of each order in the "Orders" table based on the prices and quantities in the "OrderDetails" table.**

**QUERY:**

UPDATE Orders AS o

SET TotalAmount = (

SELECT SUM(od.Quantity \* p.Price)

FROM OrderDetails AS od

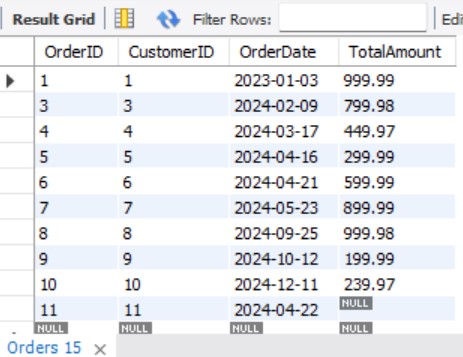
JOIN Products AS p ON od.ProductID = p.ProductID

WHERE od.OrderID = o.OrderID

);

SELECT \* FROM Orders;

**RESULT:**



1. **Write an SQL query to delete all orders and their associated order details for a specific customer from the "Orders" and "OrderDetails" tables. Allow users to input the customer ID as a parameter.**

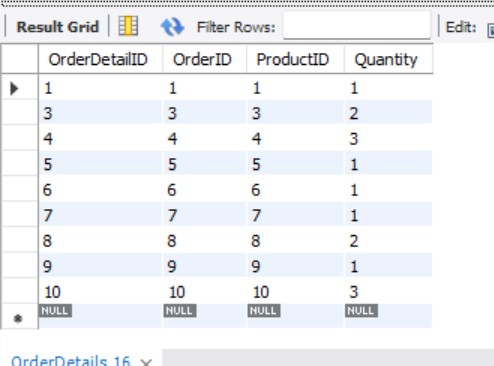
**QUERY:**

DELETE FROM OrderDetails

WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = 2);

select \* from OrderDetails;

**RESULT:**

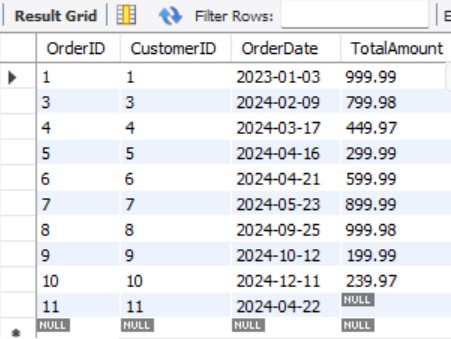


DELETE FROM Orders

WHERE CustomerID = 2;

select \* from Orders;

**RESULT:**



1. **Write an SQL query to insert a new electronic gadget product into the "Products" table, including product name, category, price, and any other relevant details.**

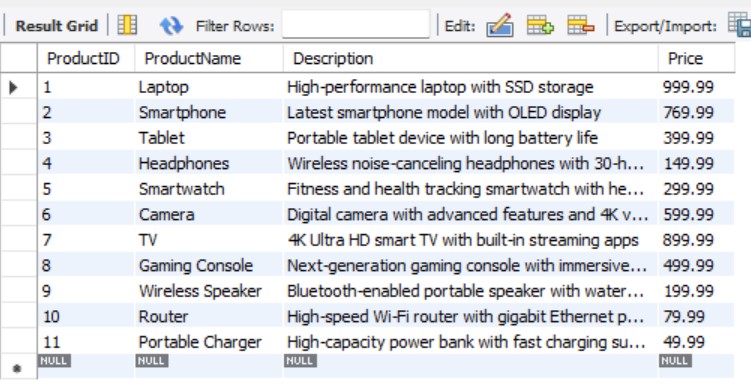
**QUERY:**

INSERT INTO Products (ProductID,ProductName, Description, Price)

VALUES (11,'Portable Charger', 'High-capacity power bank with fast charging support', 49.99);

select \* from Products;

**RESULT:**



1. **Write an SQL query to update the status of a specific order in the "Orders" table (e.g., from "Pending" to "Shipped"). Allow users to input the order ID and the new status.**

**QUERY:**

ALTER TABLE Orders

ADD COLUMN Status VARCHAR(255);

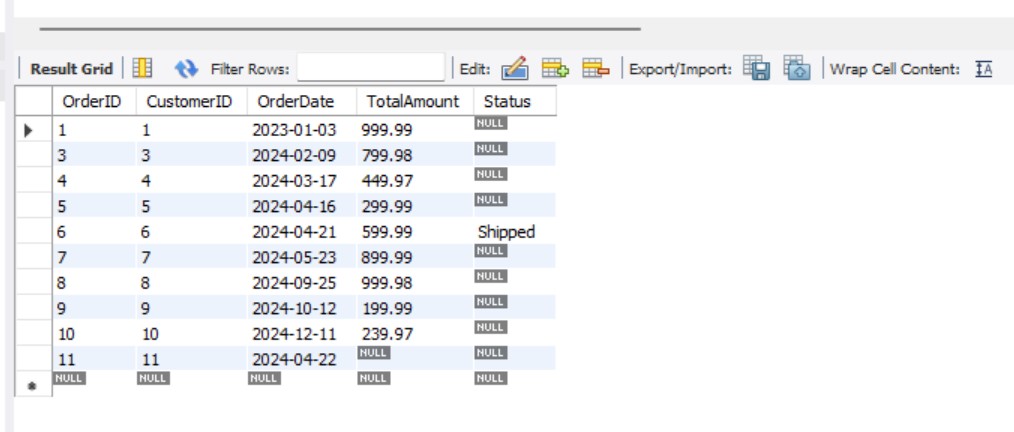
UPDATE Orders

SET Status = 'Shipped'

WHERE OrderID = 6;

SELECT \* FROM Orders;

**RESULT:**



**12.Write an SQL query to calculate and update the number of orders placed by each customer in the "Customers" table based on the data in the "Orders" table.**

**QUERY:**

ALTER TABLE Customers

ADD COLUMN NumberOfOrders INT;

UPDATE Customers AS c

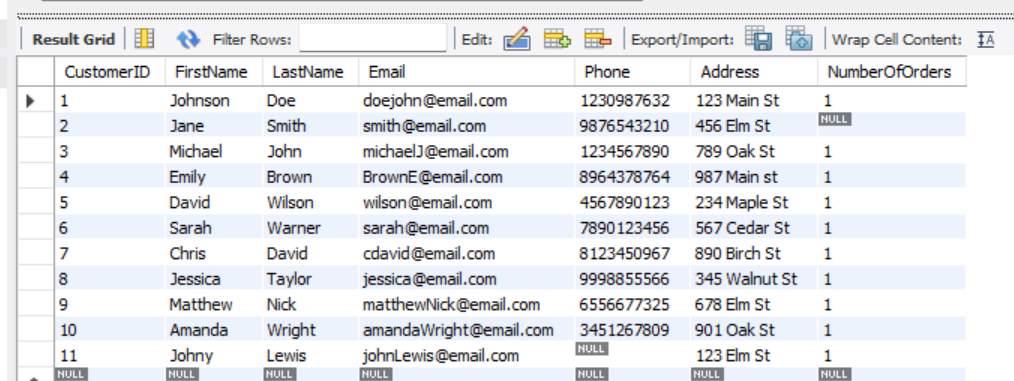
JOIN (SELECT CustomerID, COUNT(\*) AS order\_count FROM Orders GROUP BY

CustomerID) AS o

ON c.CustomerID = o.CustomerID

SET c.NumberOfOrders = o.order\_count;

**RESULT:**



Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

1. **Write an SQL query to retrieve a list of all orders along with customer information (e.g., customer name) for each order.**

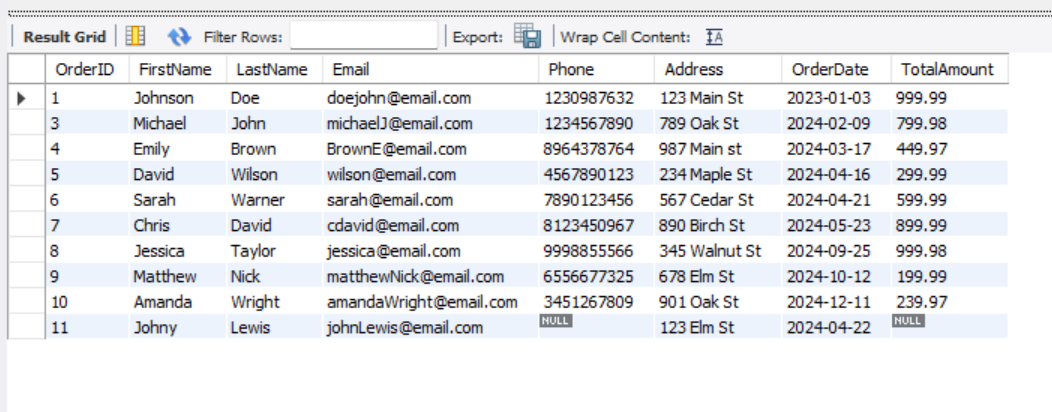
**QUERY:**

SELECT o.OrderID, c.FirstName, c.LastName, c.Email, c.Phone, c.Address, o.OrderDate, o.TotalAmount

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID;

**RESULT:**



1. **Write an SQL query to find the total revenue generated by each electronic gadget product. Include the product name and the total revenue.**

**QUERY:**

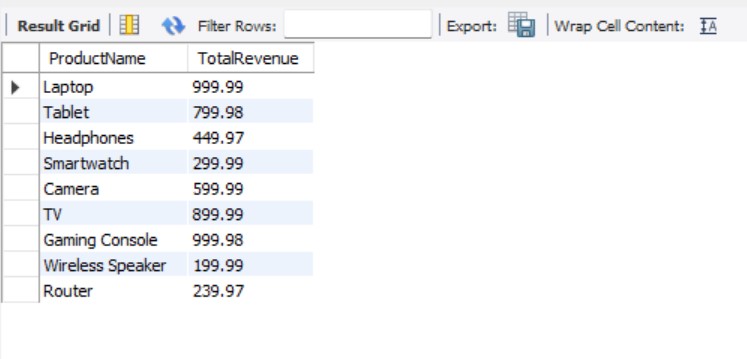
SELECT p.ProductName, SUM(od.Quantity \* p.Price) AS TotalRevenue

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.ProductName;

**RESULT:**



1. **Write an SQL query to list all customers who have made at least one purchase. Include their names and contact information.**

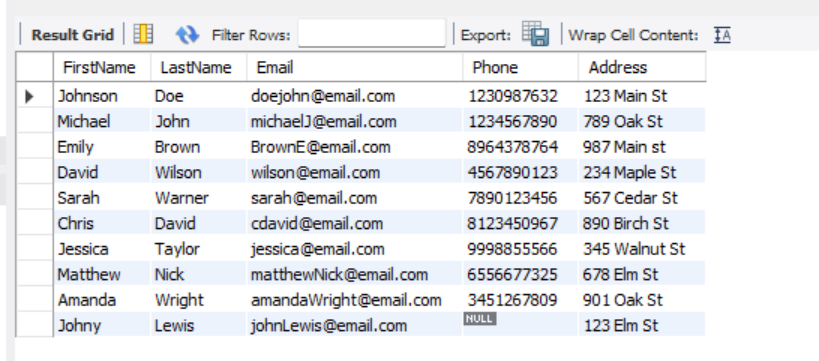
**QUERY:**

SELECT FirstName, LastName, Email, Phone, Address

FROM Customers

WHERE CustomerID IN (SELECT DISTINCT CustomerID FROM Orders);

**RESULT:**



1. **Write an SQL query to find the most popular electronic gadget, which is the one with the highest total quantity ordered. Include the product name and the total quantity ordered.**

**QUERY:**

SELECT p.ProductName, SUM(od.Quantity) AS TotalQuantityOrdered

FROM OrderDetails od

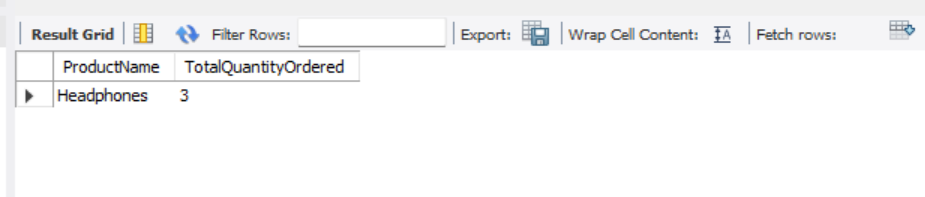
JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.ProductName

ORDER BY TotalQuantityOrdered DESC

LIMIT 1;

**RESULT:**



1. **Write an SQL query to retrieve a list of electronic gadgets along with their corresponding categories.**
2. **Write an SQL query to calculate the average order value for each customer. Include the customer's name and their average order value.**

**QUERY:**

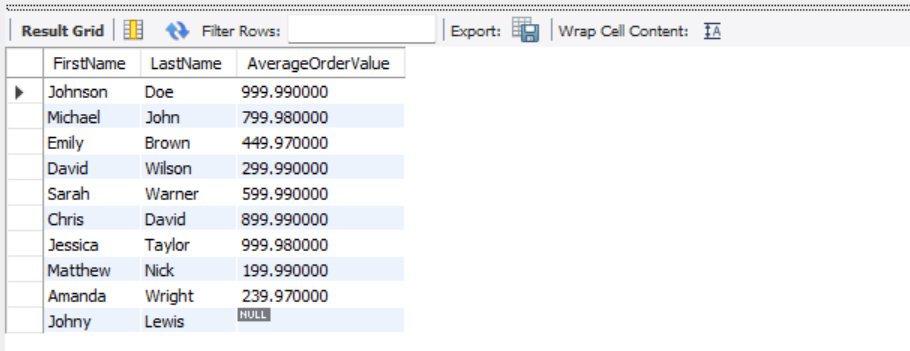
SELECT c.FirstName, c.LastName, AVG(o.TotalAmount) AS AverageOrderValue

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID;

**RESULT:**



**7. Write an SQL query to find the order with the highest total revenue. Include the order ID, customer information, and the total revenue.**

**QUERY:**

SELECT o.OrderID, c.FirstName, c.LastName, c.Email, c.Phone, c.Address, SUM(od.Quantity \* p.Price) AS TotalRevenue

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

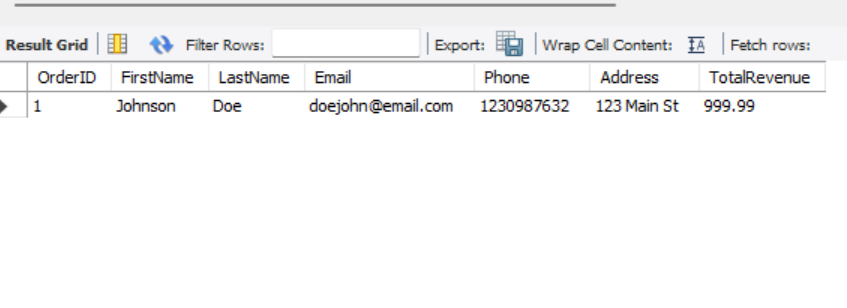
JOIN Products p ON od.ProductID = p.ProductID

GROUP BY o.OrderID

ORDER BY TotalRevenue DESC

LIMIT 1;

**RESULT:**



**8.Write an SQL query to list electronic gadgets and the number of times each product has been ordered.**

**QUERY:**

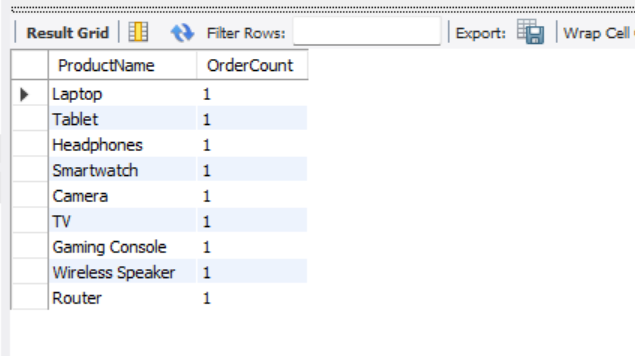
SELECT p.ProductName, COUNT(\*) AS OrderCount

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.ProductName;

**RESULT:**



**9.Write an SQL query to find customers who have purchased a specific electronic gadget product. Allow users to input the product name as a parameter.**

**QUERY:**

SELECT c.FirstName, c.LastName, c.Email, c.Phone, c.Address

FROM Customers c

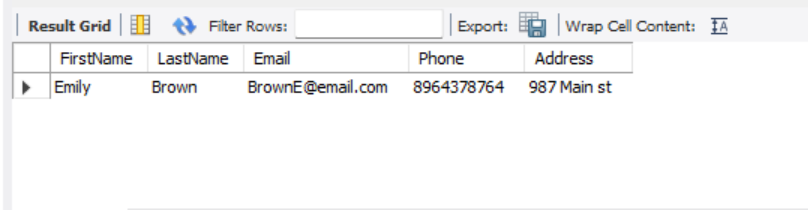
JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

WHERE p.ProductName = 'Headphones';

**RESULT:**



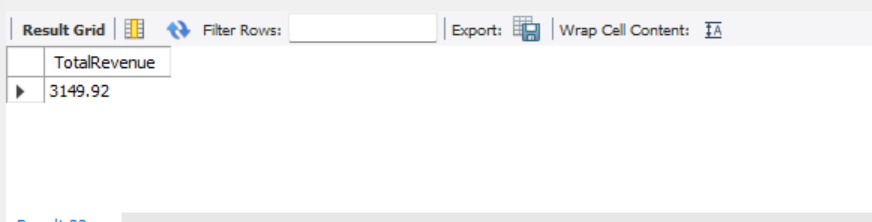
**10.Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period. Allow users to input the start and end dates as parameters.**

**QUERY**

SELECT SUM(o.TotalAmount) AS TotalRevenue

FROM Orders o

WHERE o.OrderDate BETWEEN '2023-01-03' AND '2024-04-21';

**RESULT:**

Task 4. Subquery and its type:

1. **Write an SQL query to find out which customers have not placed any orders.**

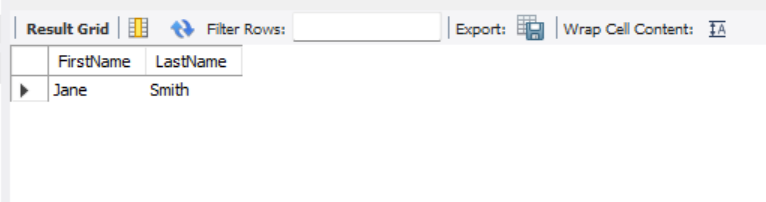
**QUERY:**

SELECT FirstName, LastName

FROM Customers

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

**RESULT:**



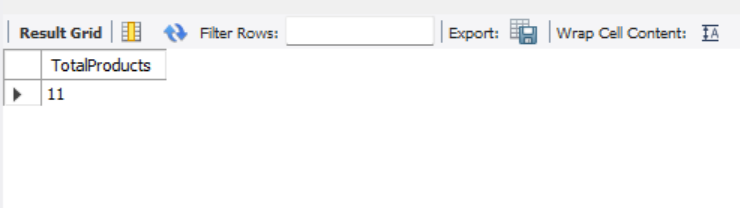
1. **Write an SQL query to find the total number of products available for sale.**

**QUERY:**

SELECT COUNT(\*) AS TotalProducts

FROM Products;

**RESULT:**



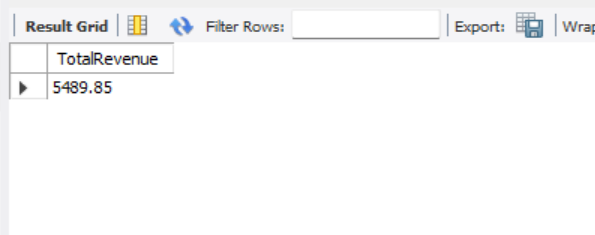
1. **Write an SQL query to calculate the total revenue generated by TechShop.**

**QUERY:**

SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders;

**RESULT:**



1. **Write an SQL query to calculate the average quantity ordered for products in a specific category. Allow users to input the category name as a parameter.**

**QUERY:**

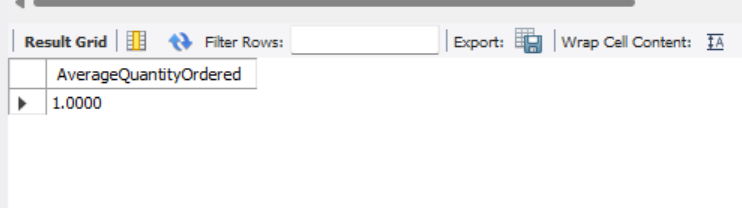
SELECT AVG(od.Quantity) AS AverageQuantityOrdered

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

WHERE p.Productname = 'Laptop';

**RESULT:**



1. **Write an SQL query to calculate the total revenue generated by a specific customer. Allow users to input the customer ID as a parameter.**

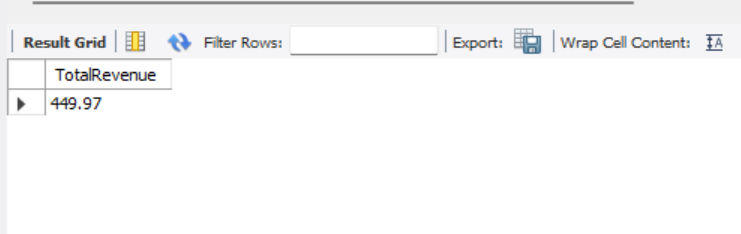
**QUERY:**

5.SELECT SUM(o.TotalAmount) AS TotalRevenue

FROM Orders o

WHERE o.CustomerID = '4';

**RESULT:**



1. **Write an SQL query to find the customers who have placed the most orders. List their names and the number of orders they've placed.**

**QUERY:**

SELECT c.FirstName, c.LastName, COUNT(\*) AS OrderCount

FROM Customers c

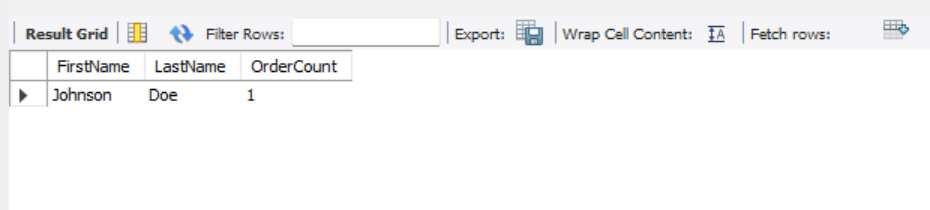
JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID

ORDER BY OrderCount DESC

LIMIT 1;

**RESULT:**



1. **Write an SQL query to find the most popular product category, which is the one with the highest total quantity ordered across all orders.**

**QUERY:**

SELECT p.Productname, SUM(od.Quantity) AS TotalQuantityOrdered

FROM OrderDetails od

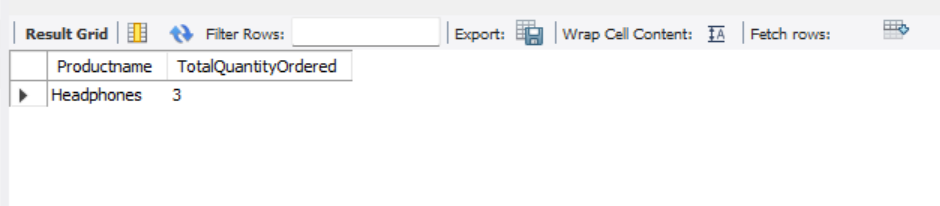
JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.Productname

ORDER BY TotalQuantityOrdered DESC

LIMIT 1;

**RESULT:**



1. **Write an SQL query to find the customer who has spent the most money (highest total revenue) on electronic gadgets. List their name and total spending.**

**QUERY:**

SELECT c.FirstName, c.LastName, SUM(o.TotalAmount) AS TotalSpending

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

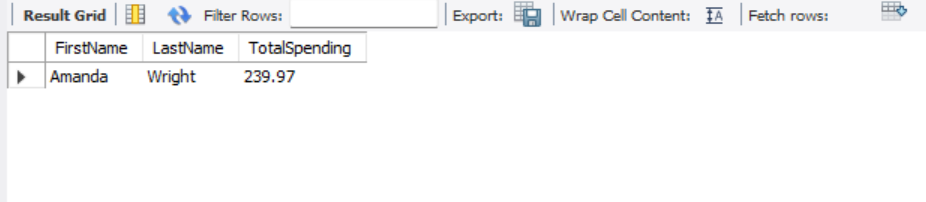
WHERE p.Productname = 'Productname'

GROUP BY c.CustomerID

ORDER BY TotalSpending DESC

LIMIT 1;

**RESULT:**



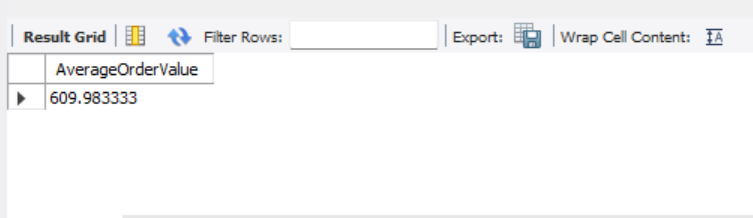
**9.Write an SQL query to calculate the average order value (total revenue divided by the number of orders) for all customers.**

**QUERY:**

SELECT AVG(TotalAmount) AS AverageOrderValue

FROM Orders;

**RESULT:**



**10.Write an SQL query to find the total number of orders placed by each customer and list their names along with the order count.**

**QUERY:**

SELECT c.FirstName, c.LastName, COUNT(\*) AS OrderCount

FROM Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID;

**RESULT:**

