**Vijay R**

**Superset ID: 5371616**

**Saveetha Engineering College**

**TechShop, an electronic gadgets shop**

**Task:1. Database Design:**

1.Creating Database:

CREATE DATABASE TechGadgetShop;

USE TechGadgetShop;

2.Tables:

CREATE TABLE Customers (

CustomerID INT AUTO\_INCREMENT PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL,

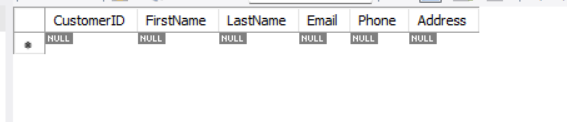
LastName VARCHAR(50) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL,

Phone VARCHAR(15) NOT NULL,

Address TEXT NOT NULL

);



CREATE TABLE Products (

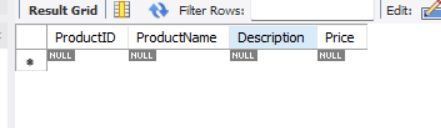
ProductID INT AUTO\_INCREMENT PRIMARY KEY,

ProductName VARCHAR(100) NOT NULL,

Description TEXT,

Price DECIMAL(10,2) NOT NULL

);



CREATE TABLE Orders (

OrderID INT AUTO\_INCREMENT PRIMARY KEY,

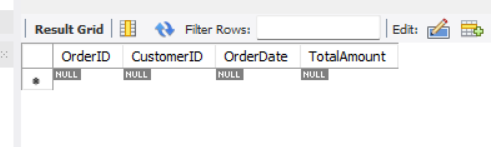
CustomerID INT NOT NULL,

OrderDate DATE NOT NULL,

TotalAmount DECIMAL(10,2) NOT NULL,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE CASCADE

);



CREATE TABLE OrderDetails (

OrderDetailID INT AUTO\_INCREMENT PRIMARY KEY,

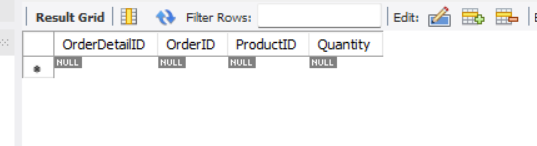
OrderID INT NOT NULL,

ProductID INT NOT NULL,

Quantity INT NOT NULL CHECK (Quantity > 0),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID) ON DELETE CASCADE,

FOREIGN KEY (ProductID) REFERENCES Products(ProductID) ON DELETE CASCADE);



CREATE TABLE Inventory (

InventoryID INT AUTO\_INCREMENT PRIMARY KEY,

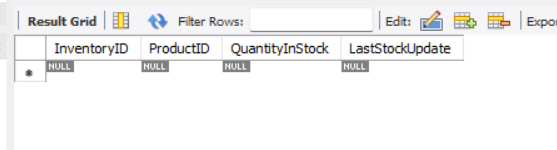
ProductID INT NOT NULL,

QuantityInStock INT NOT NULL CHECK (QuantityInStock >= 0),

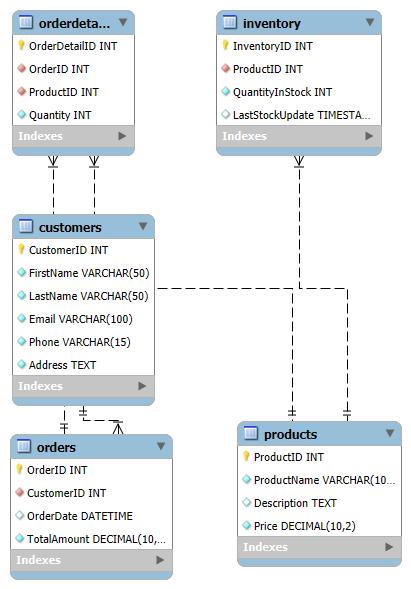
LastStockUpdate DATE NOT NULL,

FOREIGN KEY (ProductID) REFERENCES Products(ProductID) ON DELETE CASCADE

);



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



4. Insert at least 10 sample records into each of the following tables.

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

('Steve', 'John', 'steve.john@email.com', '9876543210', '123 Main St, NY'),

('Mark', 'Smith', 'mark.smith@email.com', '9876543211', '456 Last St, LA'),

('Kelly', 'Brown', 'kelly.brown@email.com', '9876543212', '789 Pine St, SF'),

('Mohamad', 'Sherif', 'sherif.m345@email.com', '9876543213', '101 Broadway St, TX'),

('David', 'Raj', 'david.rr45@email.com', '9876543214', '202 Paris St, FL'),

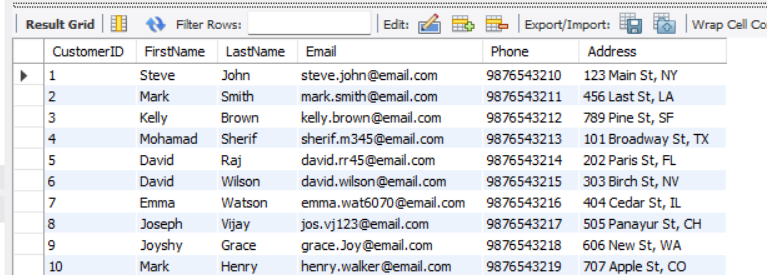
('David', 'Wilson', 'david.wilson@email.com', '9876543215', '303 Birch St, NV'),

('Emma', 'Watson', 'emma.wat6070@email.com', '9876543216', '404 Cedar St, IL'),

('Joseph', 'Vijay', 'jos.vj123@email.com', '9876543217', '505 Panayur St, CH'),

('Joyshy', 'Grace', 'grace.Joy@email.com', '9876543218', '606 New St, WA'),

('Mark', 'Henry', 'henry.walker@email.com', '9876543219', '707 Apple St, CO');



INSERT INTO Products (ProductName, Description, Price) VALUES

('Smartphone', 'Latest Android smartphone', 699.99),

('Laptop', '15-inch gaming laptop', 1299.49),

('Tablet', '10-inch display tablet', 399.99),

('Smartwatch', 'Water-resistant smartwatch', 199.99),

('Bluetooth Speaker', 'Portable speaker', 99.99),

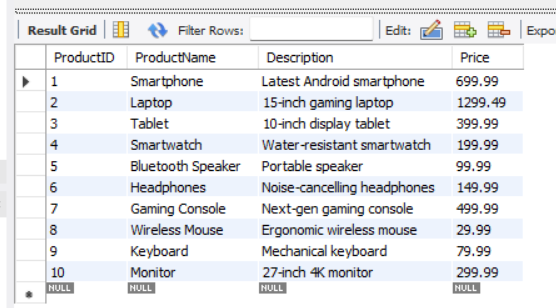
('Headphones', 'Noise-cancelling headphones', 149.99),

('Gaming Console', 'Next-gen gaming console', 499.99),

('Wireless Mouse', 'Ergonomic wireless mouse', 29.99),

('Keyboard', 'Mechanical keyboard', 79.99),

('Monitor', '27-inch 4K monitor', 299.99);



INSERT INTO Orders (CustomerID, OrderDate, TotalAmount) VALUES

(1, '2024-01-01', 999.99),

(2, '2024-01-05', 699.99),

(3, '2024-02-10', 1299.49),

(4, '2024-02-12', 1499.99),

(5, '2024-04-15', 399.99),

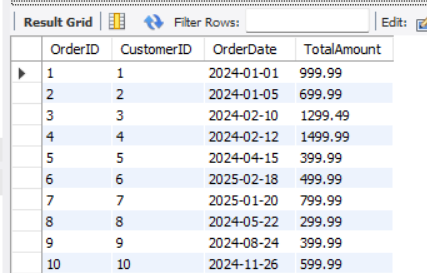
(6, '2025-02-18', 499.99),

(7, '2025-01-20', 799.99),

(8, '2024-05-22', 299.99),

(9, '2024-08-24', 399.99),

(10, '2024-11-26', 599.99);



INSERT INTO OrderDetails (OrderID, ProductID, Quantity) VALUES

(1, 1, 1),

(2, 2, 1),

(3, 3, 5),

(4, 4, 1),

(5, 5, 3),

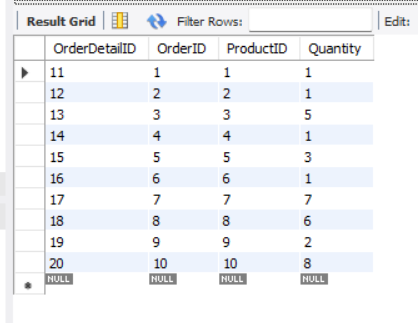
(6, 6, 1),

(7, 7, 7),

(8, 8, 6),

(9, 9, 2),

(10, 10, 8);



INSERT INTO Inventory (ProductID, QuantityInStock, LastStockUpdate) VALUES

(1, 50, '2024-08-30'),

(2, 30, '2024-03-29'),

(3, 40, '2024-01-28'),

(4, 25, '2024-06-27'),

(5, 60, '2025-01-26'),

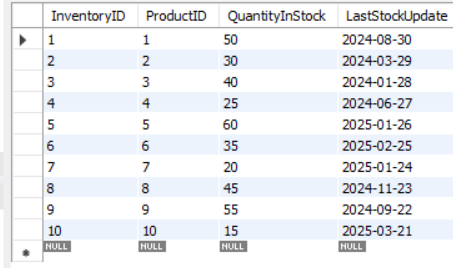
(6, 35, '2025-02-25'),

(7, 20, '2025-01-24'),

(8, 45, '2024-11-23'),

(9, 55, '2024-09-22'),

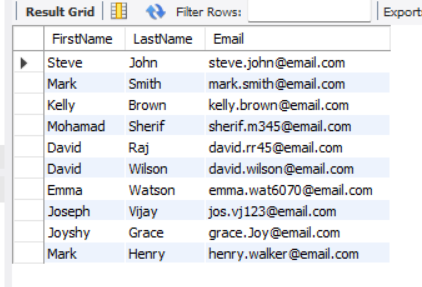
(10, 15, '2025-03-21');



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

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

SELECT FirstName, LastName, Email FROM Customers;

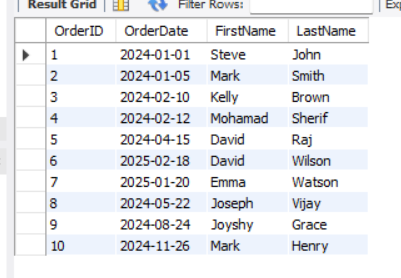


2. Write an SQL query to list all orders with their order dates and corresponding customer names.

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

FROM Orders

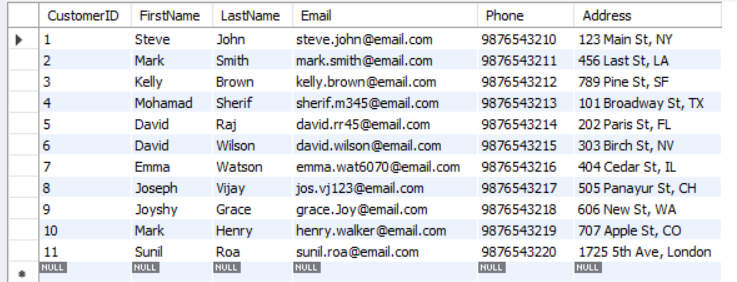
JOIN Customers ON Orders.CustomerID = Customers.CustomerID;



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

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

VALUES ('Sunil', 'Roa', 'sunil.roa@email.com', '9876543220', '1725 5th Ave, London');



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

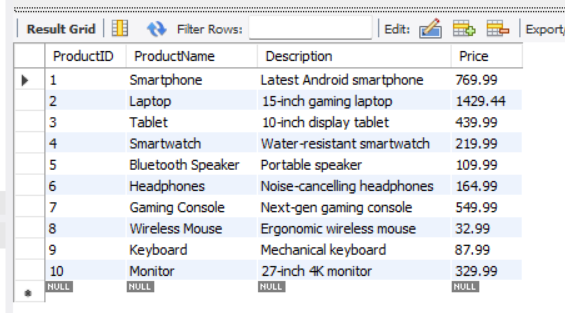
ALTER TABLE Products MODIFY COLUMN Price DECIMAL(10,2);

SET SQL\_SAFE\_UPDATES = 0;

UPDATE Products

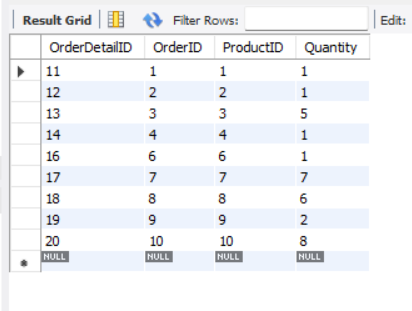
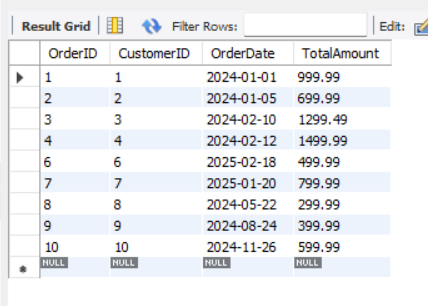
SET Price = ROUND(Price \* 1.10, 2)

WHERE EXISTS (SELECT 1 FROM (SELECT ProductID FROM Products) AS tmp WHERE tmp.ProductID = Products.ProductID);



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.

DELETE FROM Orders WHERE OrderID = 5;

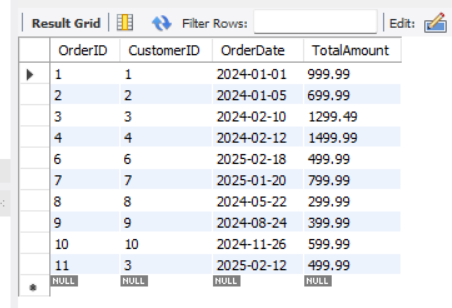
DELETE FROM OrderDetails WHERE OrderID = 5;

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

INSERT INTO Orders (CustomerID, OrderDate, TotalAmount)

VALUES (3, '2025-02-12', 499.99);

SELECT \* from Orders;



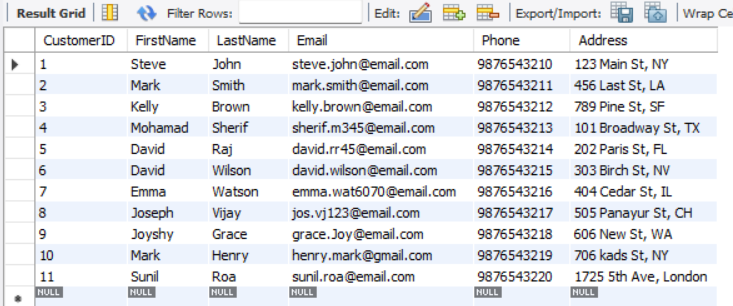
7. Write an SQL query to update the contact information (e.g., email and address) of a specific customer in the "Customers" table. Allow users to input the customer ID and new contact information.

UPDATE Customers

SET Email = 'henry.mark@gmail.com', Address = '706 kads St, NY'

WHERE CustomerID = 10;

SELECT \* from Customers;



8. 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.

UPDATE Orders o

SET o.TotalAmount = (

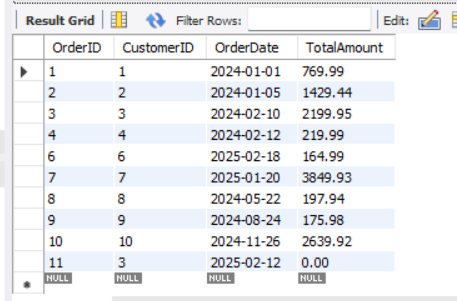
SELECT IFNULL(SUM(od.Quantity \* p.Price), 0)

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

WHERE od.OrderID = o.OrderID

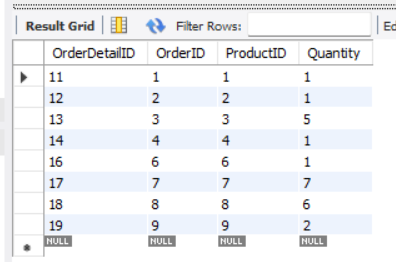
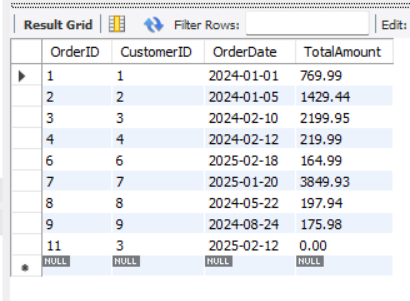
);



9. 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.

DELETE FROM OrderDetails WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = 10);

DELETE FROM Orders WHERE CustomerID = 10;



10. 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.

INSERT INTO Products (ProductName, Description, Price)

VALUES ('Power Bank', '20,000 mah Battery', 738.99);



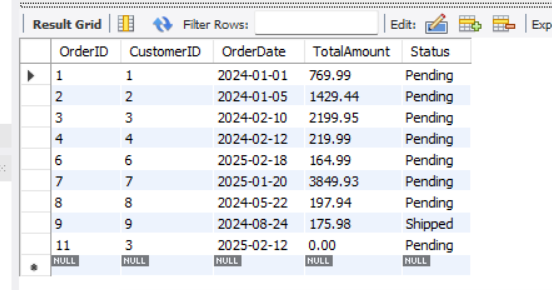
11. Write an SQL query to update the status of a specific order in the "Orders" table (e.g., from "Pending" to "Shipped"). Allow users to input the order ID and the new status.

ALTER TABLE Orders ADD COLUMN Status VARCHAR(20) DEFAULT 'Pending';

UPDATE Orders

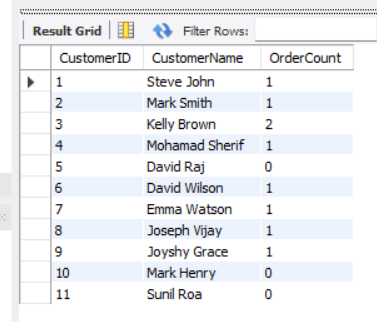
SET Status = 'Shipped'

WHERE OrderID = 9;



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.

SELECT CustomerID, CONCAT(FirstName, ' ', LastName) AS CustomerName, OrderCount FROM Customers;



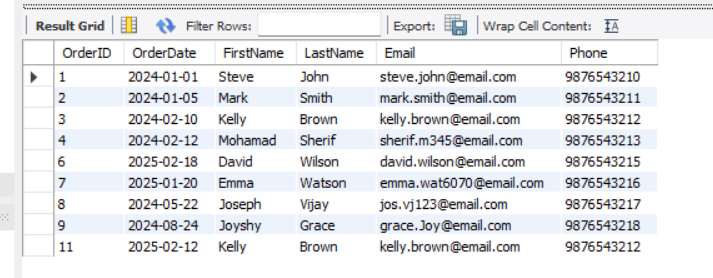
**Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:**

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

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

FROM Orders o

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



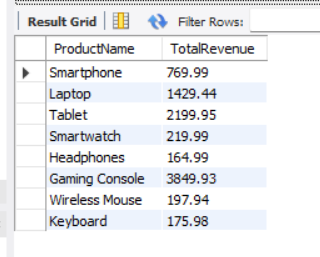
2. Write an SQL query to find the total revenue generated by each electronic gadget product. Include the product name and the total revenue.

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;

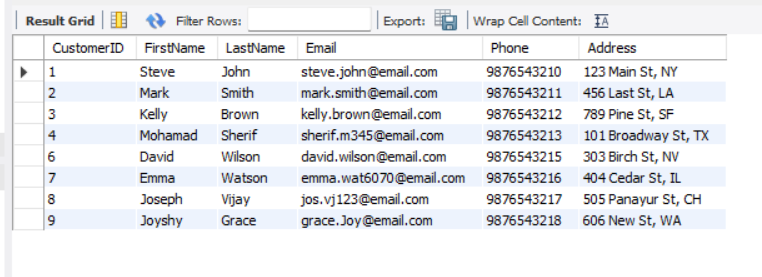


3. Write an SQL query to list all customers who have made at least one purchase. Include their names and contact information.

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

FROM Customers c

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



4. 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.

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

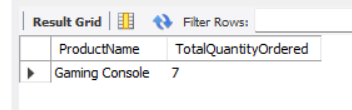
FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.ProductName

ORDER BY TotalQuantityOrdered DESC

LIMIT 1;



5. Write an SQL query to retrieve a list of electronic gadgets along with their corresponding categories.

SELECT ProductName, Description FROM Products;



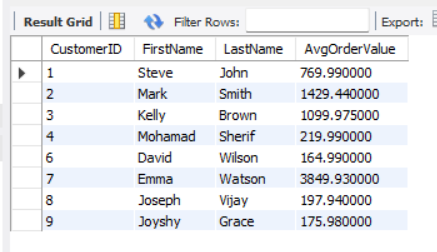
6. Write an SQL query to calculate the average order value for each customer. Include the customer's name and their average order value.

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

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName;



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

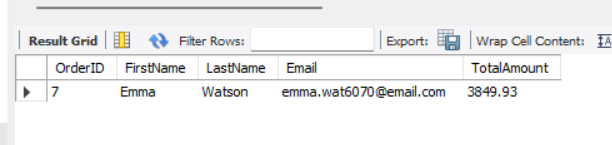
SELECT o.OrderID, c.FirstName, c.LastName, c.Email, o.TotalAmount

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

ORDER BY o.TotalAmount DESC

LIMIT 1;



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

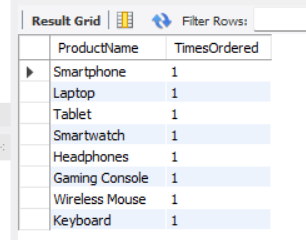
SELECT p.ProductName, COUNT(od.OrderDetailID) AS TimesOrdered

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.ProductName

ORDER BY TimesOrdered DESC;



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.

SELECT DISTINCT c.CustomerID, c.FirstName, c.LastName, c.Email, c.Phone

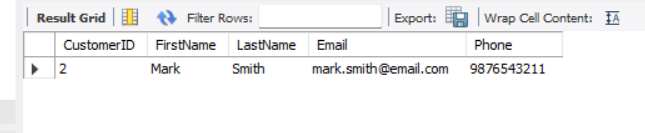
FROM Customers c

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 = 'Laptop';

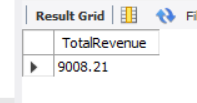


10. Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period. Allow users to input the start and end dates as parameters.

SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders

WHERE OrderDate BETWEEN '2024-01-01' AND '2025-03-30';



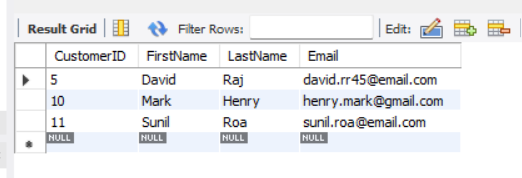
**Task 4. Subquery and its type:**

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

SELECT c.CustomerID, c.FirstName, c.LastName, c.Email

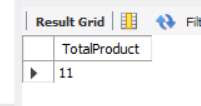
FROM Customers c

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



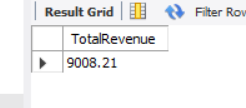
2. Write an SQL query to find the total number of products available for sale.

SELECT COUNT(\*) AS TotalProduct FROM Products;



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

SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders;



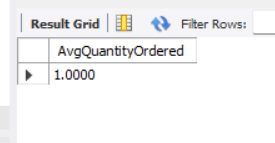
4. Write an SQL query to calculate the average quantity ordered for products in a specific category. Allow users to input the category name as a parameter.

SELECT AVG(od.Quantity) AS AvgQuantityOrdered

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

WHERE p.Productname = 'Laptop';

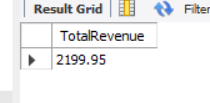


5. Write an SQL query to calculate the total revenue generated by a specific customer. Allow users to input the customer ID as a parameter.

SELECT SUM(o.TotalAmount) AS TotalRevenue

FROM Orders o

WHERE o.CustomerID = 3;



6. 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.

SELECT c.CustomerID, c.FirstName, c.Lastname, COUNT(o.OrderID) AS TotalOders

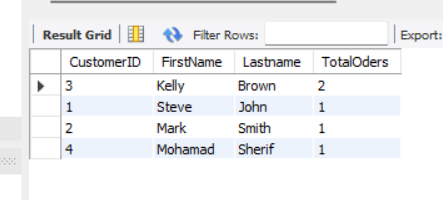
FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName

ORDER BY TotalOders DESC

LIMIT 4;



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

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

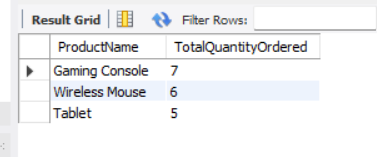
FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.ProductName

ORDER BY TotalQuantityOrdered DESC

LIMIT 3;



8. 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.

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

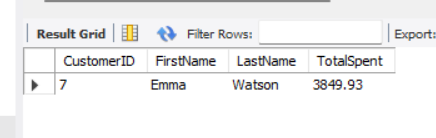
FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName

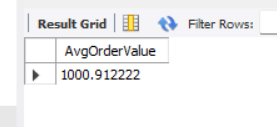
ORDER BY TotalSpent DESC

LIMIT 1;



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

SELECT AVG(TotalAmount) AS AvgOrderValue FROM Orders;



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.

SELECT c.CustomerID, c.FirstName, c.LastName, COUNT(o.OrderID) AS OrderCount

FROM Customers c

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

GROUP BY c.CustomerID, c.FirstName, c.LastName

ORDER BY OrderCount DESC;

