Assignment 1

You are working as a database administrator for a fictional company named "TechShop," which sells electronic gadgets. TechShop maintains data related to their products, customers, and orders. Your task is to design and implement a database for TechShop based on the following requirements.

Database Tables:

1. Customers:

* CustomerID (Primary Key)
* FirstName
* LastName
* Email
* Phone
* Address

2. Products:

* ProductID (Primary Key)
* ProductName
* Description
* Price

3. Orders:

* OrderID (Primary Key)
* CustomerID (Foreign Key referencing Customers)
* OrderDate
* TotalAmount

4. OrderDetails:

* OrderDetailID (Primary Key)
* OrderID (Foreign Key referencing Orders)
* ProductID (Foreign Key referencing Products)
* Quantity

5. Inventory

* InventoryID (Primary Key)
* ProductID (Foreign Key referencing Products)
* QuantityInStock
* LastStockUpdate

Task:1. Database Design:

1. Create the database named "TechShop" CREATE DATABASE TechShop;

USE TechShop;

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

CREATE TABLE Customers(

CustomerId INT PRIMARY KEY,

FirstName VARCHAR(20),

LastName VARCHAR(20),

Email VARCHAR(40),

Phone VARCHAR(10),

Address VARCHAR(30),

# );

SELECT \* FROM Customers;

CREATE TABLE Products(

ProductID INT PRIMARY KEY,

ProductName VARCHAR(30),

Description VARCHAR(50),

Price DECIMAL(10,2)

# );

CREATE TABLE Orders(

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

TotalAmount DECIMAL(10,2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerId)

# );

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

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,

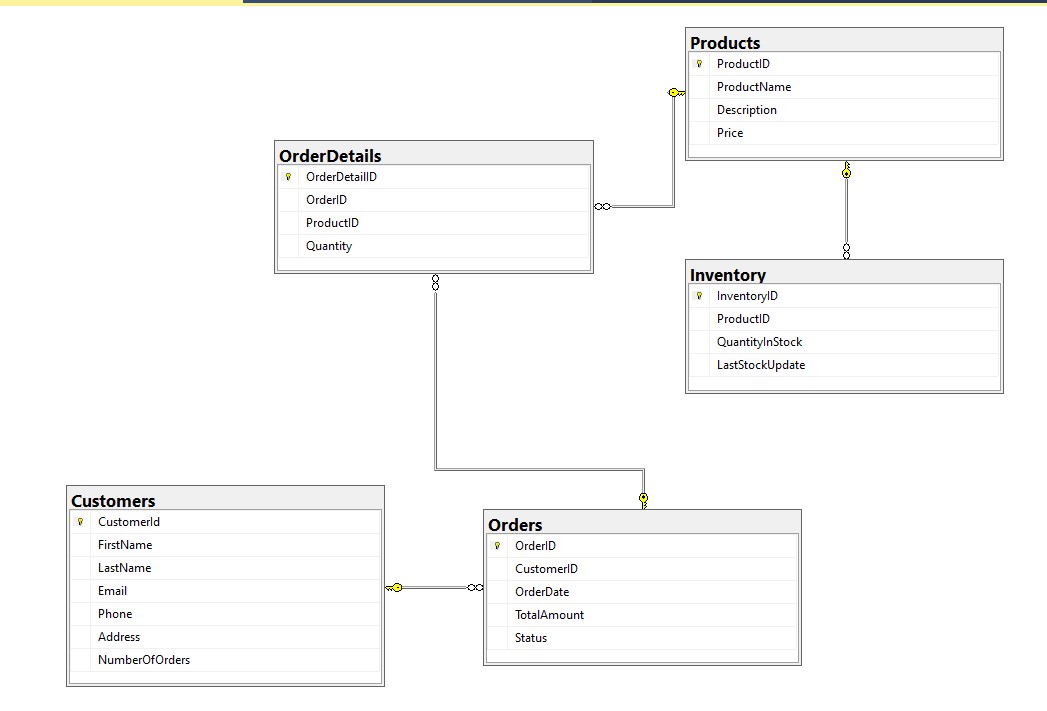
ProductID INT,

QuantityInStock INT,

LastStockUpdate DATE,

FOREIGN KEY (ProductID) REFERENCES Products(ProductID) );

1. Define the schema for the Customers, Products, Orders, OrderDetails and Inventory tables based on the provided schema.
2. Create an ERD (Entity Relationship Diagram) for the database.
3. Create appropriate Primary Key and Foreign Key constraints for referential integrity.



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

INSERT INTO Customers VALUES

(1, 'John', 'Doe', 'john.doe@email.com', '123-456-7890', '123 Main St'),

(2, 'Jane', 'Smith', 'jane.smith@email.com', '987-654-3210', '456 Oak St'), -- ... (Insert 8 more records)

-- Step 8: Insert sample records into Products table

INSERT INTO Products VALUES

(1, 'Laptop', 'High-performance laptop', 999.99),

(2, 'Smartphone', 'Flagship smartphone', 699.99),

-- ... (Insert 8 more records)

-- Step 9: Insert sample records into Orders table

INSERT INTO Orders VALUES

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

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

-- ... (Insert 8 more records)

-- Step 10: Insert sample records into OrderDetails table

INSERT INTO OrderDetails VALUES

(1, 1, 1, 2),

(2, 1, 2, 1),

-- ... (Insert 8 more records)

-- Step 11: Insert sample records into Inventory table

INSERT INTO Inventory VALUES

(1, 1, 50, '2023-01-01'),

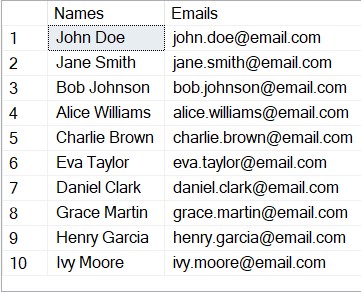
(2, 2, 100, '2023-02-01'),

-- ... (Insert 8 more records)

Tasks 2: Select, Where, Between, AND LIKE

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

SELECT FirstName + ' ' + LastName AS Names, Email as Emails from Customers;



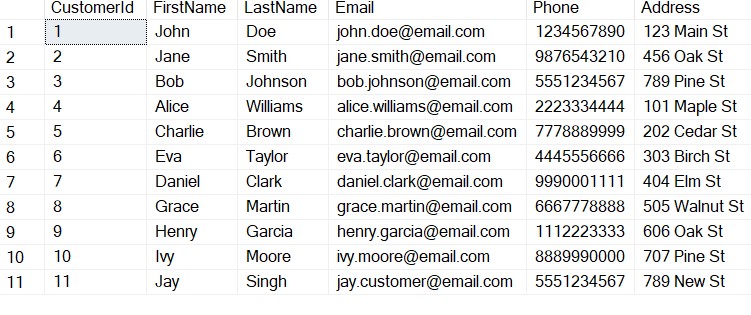
1. Write an SQL query to list all orders with their order dates and corresponding customer names.

SELECT O.OrderID, O.OrderDate, C.FirstName, C.LastName from Orders O INNER JOIN Customers C ON O.OrderID= C.CustomerId;



1. 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 VALUES (11,'Jay', 'Singh', 'jay.customer@email.com', '5551234567', '789 New St');



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

UPDATE Products

SET Price = Price \* 1.1;



1. 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.

DECLARE @OrderID INT; -- Declare the parameter

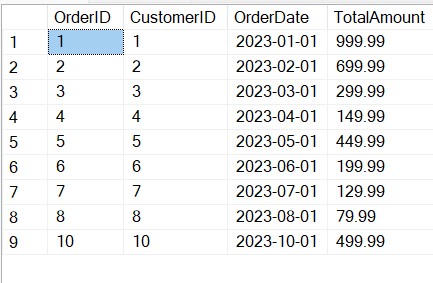
SET @OrderID = 9;

DELETE FROM OrderDetails

WHERE OrderID = @OrderID;

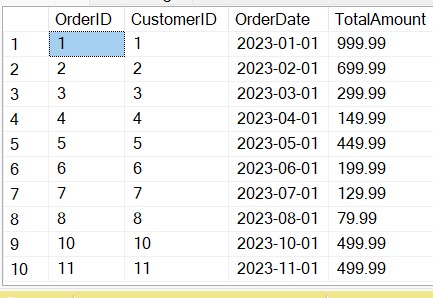
DELETE FROM Orders

WHERE OrderID = @OrderID;



1. 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 VALUES (11, 11, '2023-11-01', 499.99);



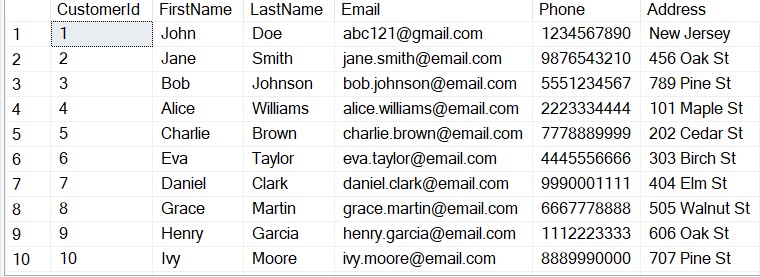
1. Write an SQL query to update the contact information (e.g., email and address) of a specific customer in the "Customers" table. Allow users to input the customer ID and new contact information.

DECLARE @CustomerID INT = 1;

UPDATE Customers

SET Email='abc121@gmail.com', Address = 'New Jersey'

WHERE CustomerId = @CustomerID;



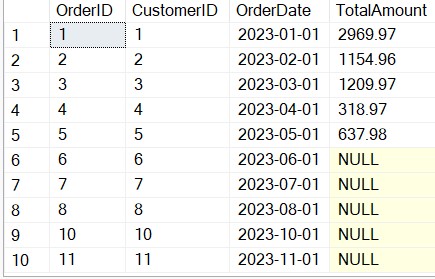
1. Write an SQL query to recalculate and update the total cost of each order in the "Orders" table based on the prices and quantities in the "OrderDetails" table.

UPDATE Orders

SET TotalAmount =(

SELECT SUM(P.Price\*O.Quantity)From OrderDetails O JOIN Products P

ON O.ProductID = P.ProductID WHERE O.OrderID = Orders.OrderID);



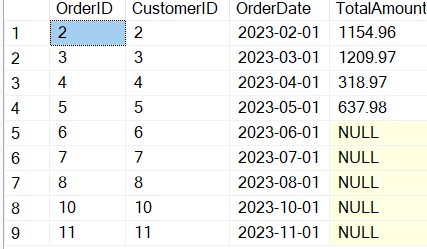
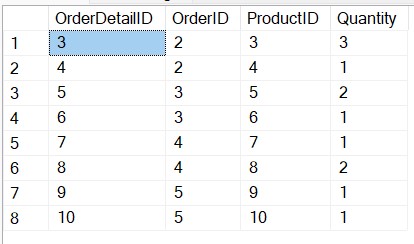
1. Write an SQL query to delete all orders and their associated order details for a specific customer from the "Orders" and "OrderDetails" tables. Allow users to input the customer ID as a parameter.

DECLARE @CustomerID INT = 1;

DELETE FROM OrderDetails WHERE OrderID = (

SELECT OrderID from Orders WHERE CustomerID = @CustomerID)

DELETE FROM Orders WHERE Orders.CustomerID = @CustomerID;



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.

INSERT INTO Products

VALUES (11,'Fridge', 'Multi-functional', 299.99);



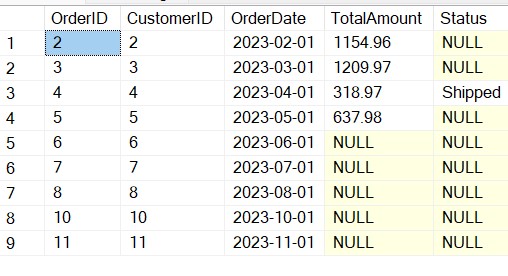
1. Write an SQL query to update the status of a specific order in the "Orders" table (e.g., from "Pending" to "Shipped"). Allow users to input the order ID and the new status.

DECLARE @OrderID INT = 4; -- Replace with the actual order ID

UPDATE Orders

SET Status = 'Shipped'

WHERE OrderID = @OrderID;



1. 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.

UPDATE Customers

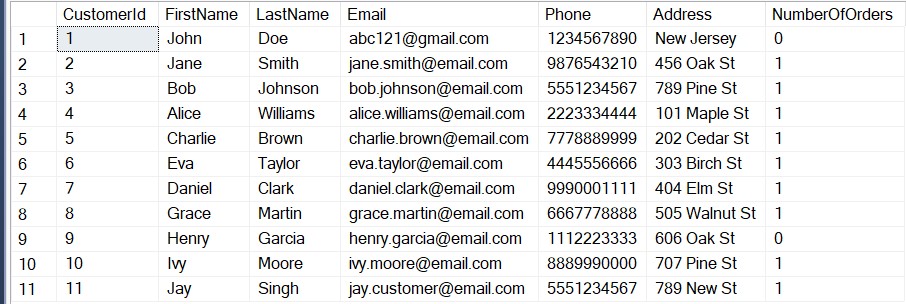
SET NumberOfOrders = (

SELECT COUNT(\*)

FROM Orders

WHERE Orders.CustomerID = Customers.CustomerId

# );



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

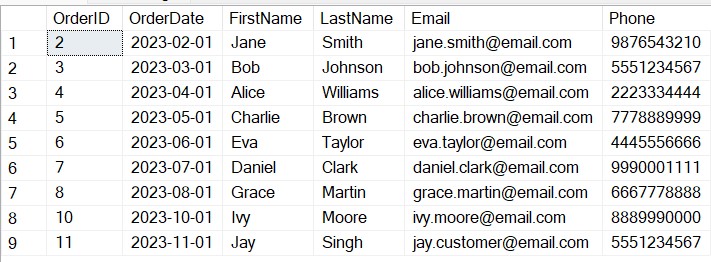
1. Write an SQL query to retrieve a list of all orders along with customer information (e.g., customer name) for each order.

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

Customers.LastName, Customers.Email, Customers.Phone

FROM Orders

JOIN Customers ON Orders.CustomerID = Customers.CustomerId;



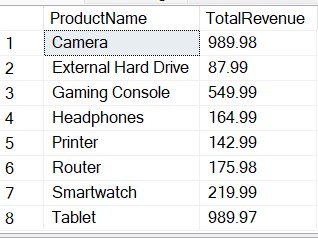
1. Write an SQL query to find the total revenue generated by each electronic gadget product. Include the product name and the total revenue.

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;



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

SELECT DISTINCT

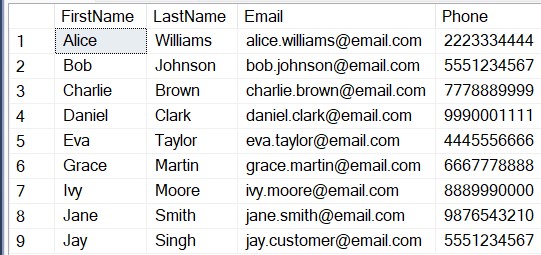
c.FirstName,

c.LastName,

c.Email,

c.Phone FROM Customers c

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

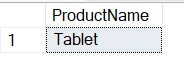


1. Write an SQL query to find the most popular electronic gadget, which is the one with the highest total quantity ordered. Include the product name and the total quantity ordered.

SELECT ProductName FROM Products WHERE ProductID = (

SELECT ProductID FROM OrderDetails O WHERE Quantity =

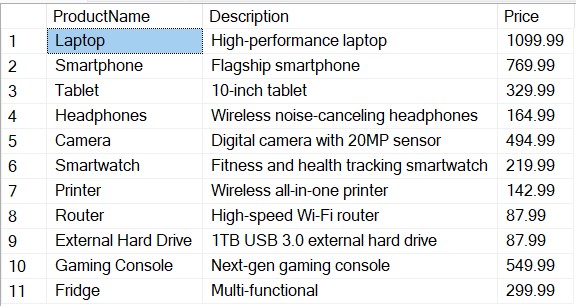
(SELECT MAX(Quantity) FROM OrderDetails) GROUP BY ProductID );



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

SELECT p.ProductName, p.Description, p.Price

FROM Products p;



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

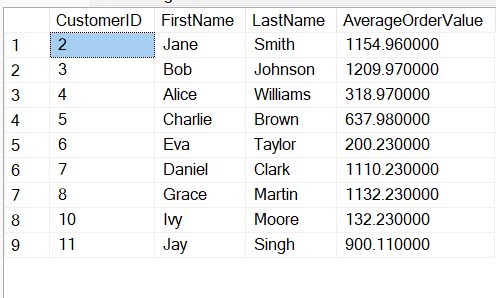
SELECT c.CustomerID, c.FirstName, c.LastName, AVG(o.TotalAmount) AS

AverageOrderValue

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

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

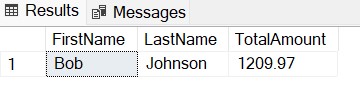


1. Write an SQL query to find the order with the highest total revenue. Include the order ID, customer information, and the total revenue.

SELECT TOP 1 C.FirstName, C.LastName, O.TotalAmount From Orders O

JOIN Customers C ON O.CustomerID = C.CustomerId

ORDER BY TotalAmount DESC;



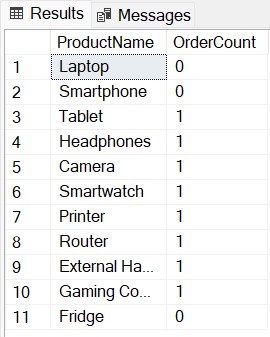
1. Write an SQL query to list electronic gadgets and the number of times each product has been ordered.

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

FROM Products p

LEFT JOIN OrderDetails od ON p.ProductID = od.ProductID

GROUP BY p.ProductID, p.ProductName;



1. 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 c.FirstName, c.LastName, c.Email, c.Phone

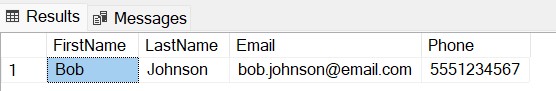
FROM Customers c

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 = @ProductName;



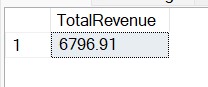
1. Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period. Allow users to input the start and end dates as parameters.

DECLARE @StartDate DATE = '2023-01-01';

DECLARE @EndDate DATE = '2023-12-31';

SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders

WHERE Orders.OrderDate BETWEEN @StartDate AND @EndDate;



Task 4: Subquery and its type:

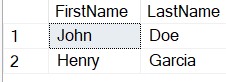
1. Write an SQL query to find out which customers have not placed any orders.

SELECT C.FirstName, C.LastName FROM Customers C

WHERE C.CustomerId NOT IN(

SELECT O.CustomerID FROM Orders O

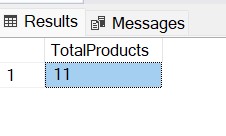
# );



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

SELECT COUNT(\*) AS TotalProducts

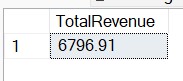
FROM Products;



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

SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders;

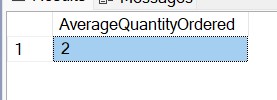


1. Write an SQL query to calculate the average quantity ordered for products in a specific category. Allow users to input the category name as a parameter.

DECLARE @PRODUCTNAME VARCHAR(20) = 'Camera';

SELECT AVG(O.Quantity) AS AverageQunantity FROM OrderDetails O WHERE O.ProductID IN

(SELECT P.ProductID FROM Products P WHERE P.ProductName = @PRODUCTNAME)

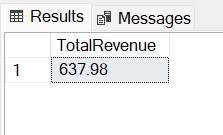


1. Write an SQL query to calculate the total revenue generated by a specific customer. Allow users to input the customer ID as a parameter.

DECLARE @CUSTID INT = 5;

SELECT SUM(O.TotalAmount) AS TotalRevenue FROM Orders O

WHERE O.CustomerID = @CUSTID;



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.

SELECT TOP 1 FirstName, LastName, OrderCount

FROM (

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

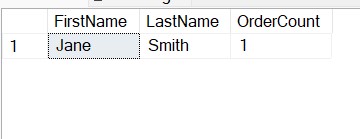
RANK() OVER (ORDER BY COUNT(o.OrderID) DESC) AS CustomerRank

FROM Customers c

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

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

) AS RankedCustomers WHERE CustomerRank = 1;



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.

SELECT p.ProductName, od.quantity FROM Products p

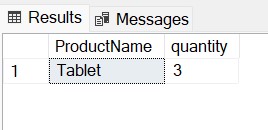
JOIN OrderDetails od ON p.ProductID = od.ProductID

WHERE Quantity = (

SELECT TOP 1 Quantity FROM OrderDetails

ORDER BY Quantity DESC

)



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.

SELECT C.FirstName, C.LastName, TotalSpending

FROM Customers C

# JOIN (

SELECT TOP 1 O.CustomerID, (O.TotalAmount \* Od.Quantity) AS

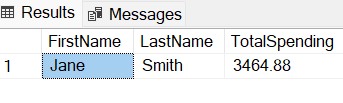
TotalSpending

FROM Orders O

JOIN OrderDetails Od ON Od.OrderID = O.OrderID

ORDER BY (O.TotalAmount \* Od.Quantity) DESC

) Orders ON C.CustomerID = Orders.CustomerID;



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

SELECT c.FirstName, c.LastName, AVG(OrderValue) AS AverageOrderValue

FROM Customers c

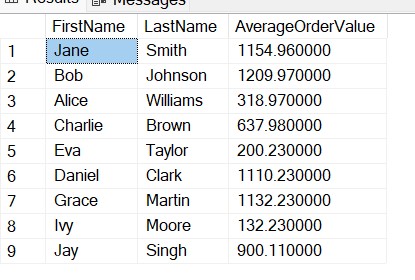
# JOIN (

SELECT o.CustomerID, SUM(o.TotalAmount) AS OrderValue

FROM Orders o

GROUP BY o.CustomerID

) AS CustomerOrderValues ON c.CustomerID = CustomerOrderValues.CustomerID GROUP BY c.CustomerID, c.FirstName, c.LastName;



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 FirstName, LastName, OrderCount

FROM Customers c

# LEFT JOIN (

SELECT CustomerID, COUNT(OrderID) AS OrderCount

FROM Orders

GROUP BY CustomerID

) AS CustomerOrderCount ON c.CustomerID = CustomerOrderCount.CustomerID;

