Task: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

# 1. Create the database named "TechShop"

# 2. Define the schema for the Customers, Products, Orders, OrderDetails and Inventory tables

# based on the provided schema.

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

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

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

# a. Customers

# b. Products

# c. Orders

# d. OrderDetails

USE TechShop;

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),

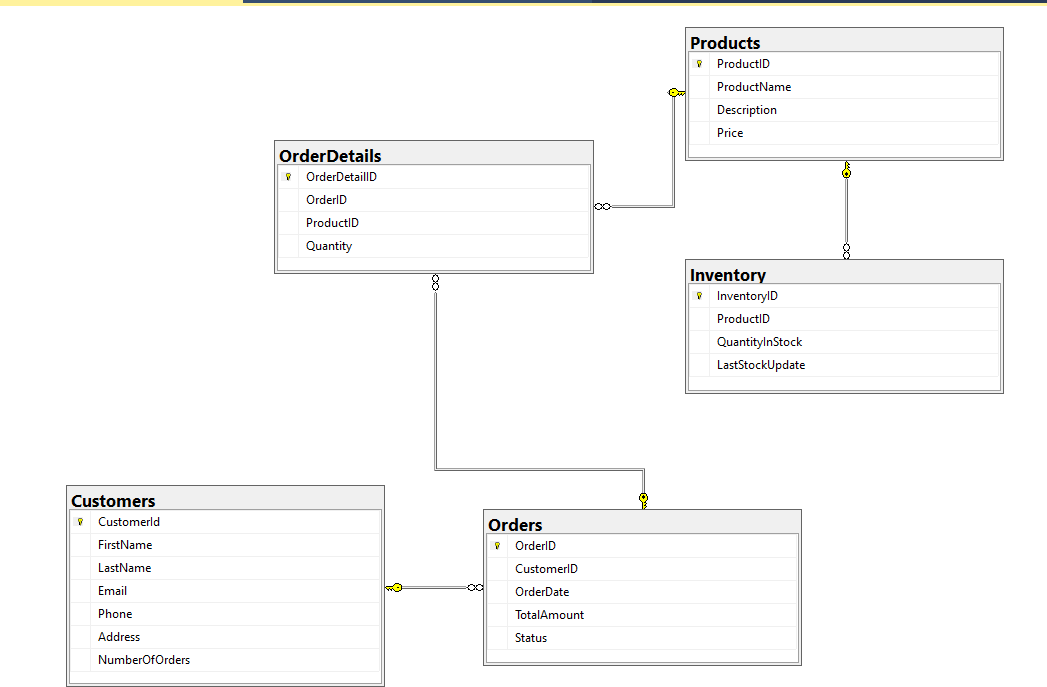
FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

CREATE TABLE Inventory ( InventoryID INT PRIMARY KEY, ProductID INT, QuantityInStock INT, LastStockUpdate DATE,

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);



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

INSERT INTO Customers VALUES

(1, 'John', 'Doe', ['john.doe@email.com',](mailto:%27john.doe@email.com) '123-456-7890', '123 Main St'),

(2, 'Jane', 'Smith', ['jane.smith@email.com',](mailto:%27jane.smith@email.com) '987-654-3210', '456 Oak St'),

INSERT INTO Products VALUES

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

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

INSERT INTO Orders VALUES

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

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

INSERT INTO OrderDetails VALUES

(1, 1, 1, 2),

(2, 1, 2, 1),

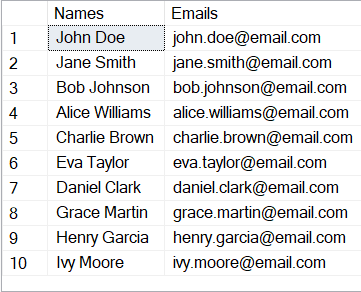
INSERT INTO Inventory VALUES

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

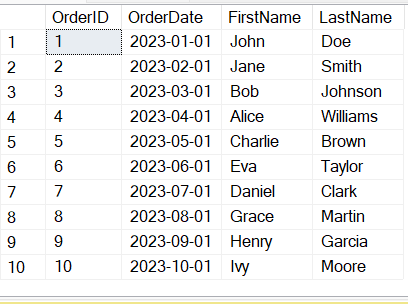
(2, 2, 100, '2023-02-01'),

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

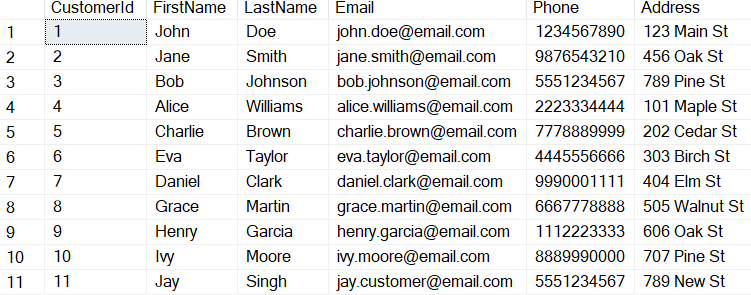
SELECT FirstName + ' ' + LastName AS Names, Email as Emails from Customers;



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

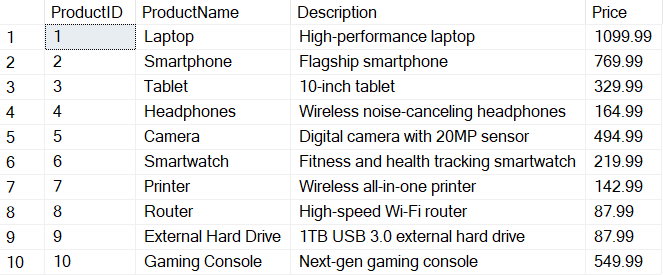


INSERT INTO Customers VALUES (11,'Jay', 'Singh', ['jay.customer@email.com',](mailto:%27jay.customer@email.com) '5551234567', '789 New St');



UPDATE Products

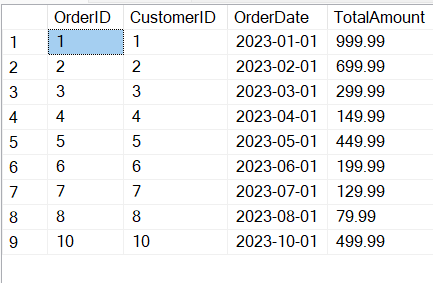
SET Price = Price \* 1.1;



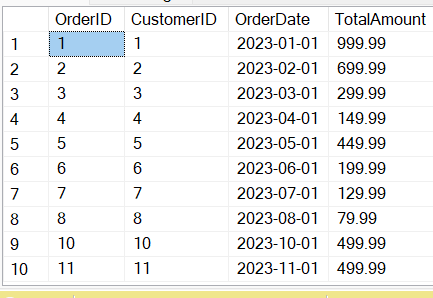
DECLARE @OrderID INT; -- Declare the parameter SET @OrderID = 9;

DELETE FROM OrderDetails WHERE OrderID = @OrderID; DELETE FROM Orders

WHERE OrderID = @OrderID;

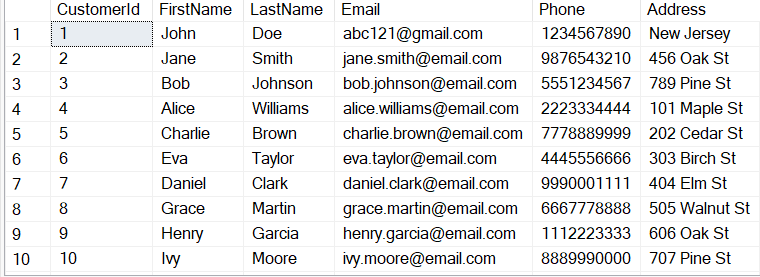


INSERT INTO Orders VALUES (11, 11, '2023-11-01', 499.99);



DECLARE @CustomerID INT = 1; UPDATE Customers

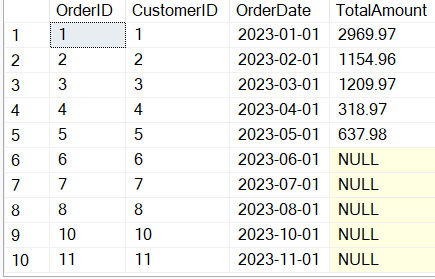
SET [Email='abc121@gmail.com',](mailto:Email%3D%27abc121@gmail.com) Address = 'New Jersey' WHERE CustomerId = @CustomerID;



UPDATE Orders

SET TotalAmount =(

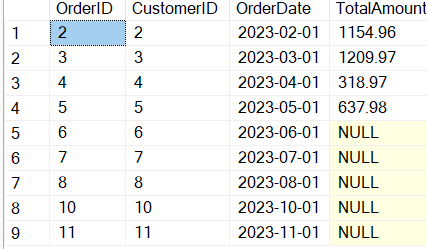
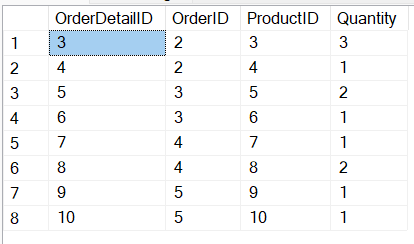
SELECT SUM(P.Price\*O.Quantity)From OrderDetails O JOIN Products P ON O.ProductID = P.ProductID WHERE O.OrderID = Orders.OrderID);



DECLARE @CustomerID INT = 1;

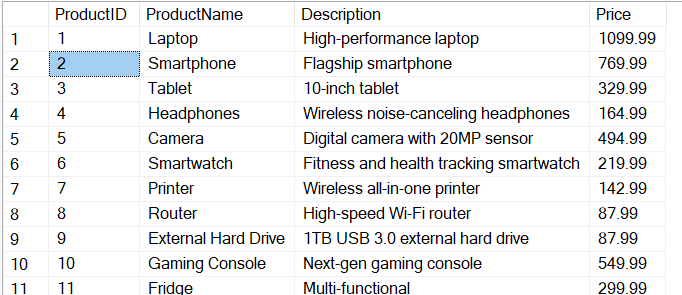
DELETE FROM OrderDetails WHERE OrderID = (

SELECT OrderID from Orders WHERE CustomerID = @CustomerID) DELETE FROM Orders WHERE Orders.CustomerID = @CustomerID;



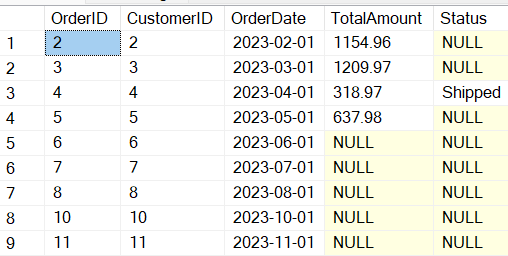
INSERT INTO Products

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



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

SET Status = 'Shipped' WHERE OrderID = @OrderID;



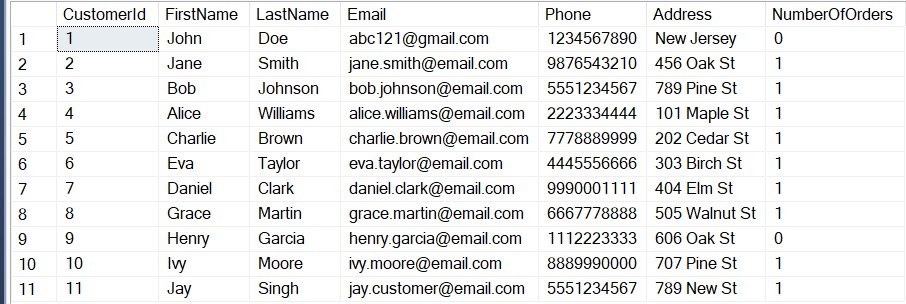
UPDATE Customers

SET NumberOfOrders = ( SELECT COUNT(\*)

FROM Orders

WHERE Orders.CustomerID = Customers.CustomerId

);

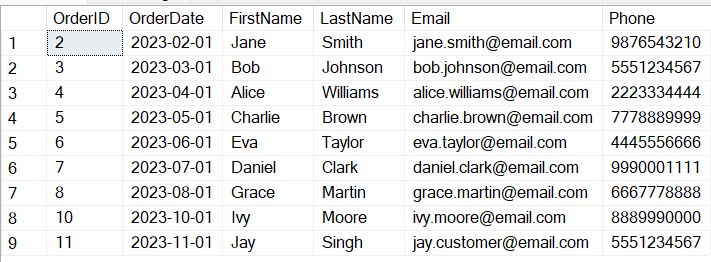


# Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

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

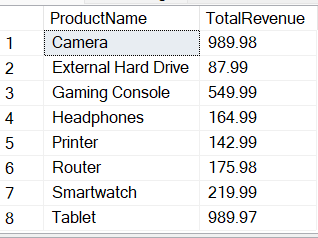
FROM Orders

JOIN Customers ON Orders.CustomerID = Customers.CustomerId;



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;

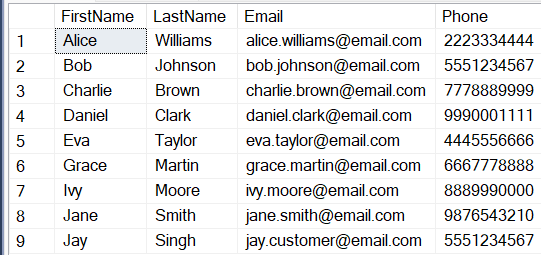


SELECT DISTINCT

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

FROM Customers c

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

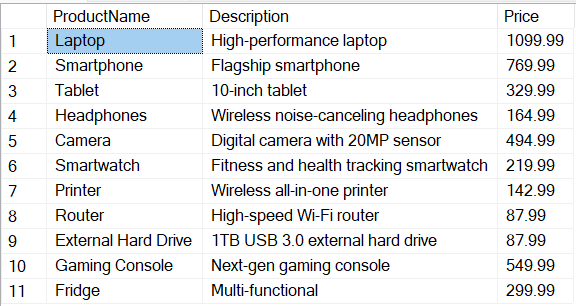


SELECT ProductName FROM Products WHERE ProductID = ( SELECT ProductID FROM OrderDetails O WHERE Quantity =

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



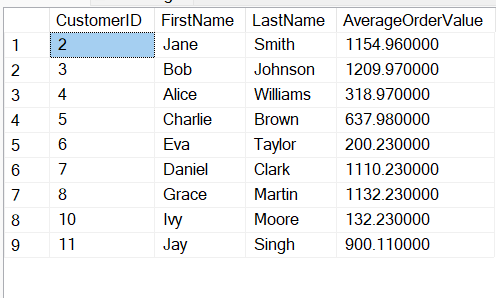
SELECT p.ProductName, p.Description, p.Price FROM Products p;



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

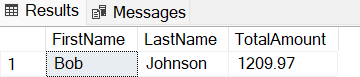
FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID GROUP BY c.CustomerID, c.FirstName, c.LastName;



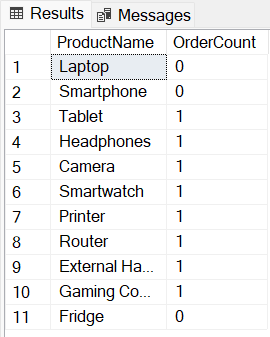
SELECT TOP 1 C.FirstName, C.LastName, O.TotalAmount From Orders O JOIN Customers C ON O.CustomerID = C.CustomerId

ORDER BY TotalAmount DESC;



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

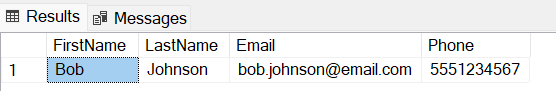
LEFT JOIN OrderDetails od ON p.ProductID = od.ProductID GROUP BY p.ProductID, p.ProductName;



SELECT c.FirstName, c.LastName, c.Email, c.Phone FROM Customers c

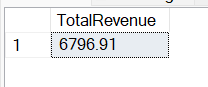
JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID JOIN Products p ON od.ProductID = p.ProductID WHERE p.ProductName = @ProductName;



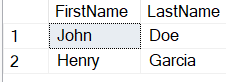
DECLARE @StartDate DATE = '2023-01-01'; DECLARE @EndDate DATE = '2023-12-31';

SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders WHERE Orders.OrderDate BETWEEN @StartDate AND @EndDate;



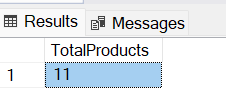
# Task 4: Subquery and its type:

SELECT C.FirstName, C.LastName FROM Customers C WHERE C.CustomerId NOT IN(

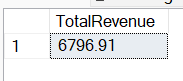
SELECT O.CustomerID FROM Orders O

);

SELECT COUNT(\*) AS TotalProducts FROM Products;



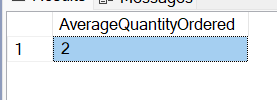
SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders;

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

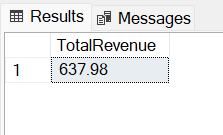
SELECT AVG(O.Quantity) AS AverageQunantity FROM OrderDetails O WHERE O.ProductID IN

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



DECLARE @CUSTID INT = 5;

SELECT SUM(O.TotalAmount) AS TotalRevenue FROM Orders O WHERE O.CustomerID = @CUSTID;



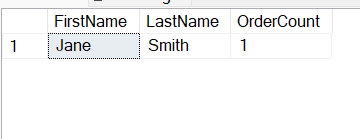
SELECT TOP 1 FirstName, LastName, OrderCount FROM (

SELECT c.FirstName, c.LastName, COUNT(o.OrderID) AS OrderCount, RANK() OVER (ORDER BY COUNT(o.OrderID) DESC) AS CustomerRank

FROM Customers c

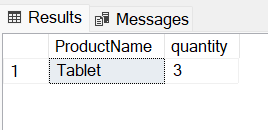
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID GROUP BY c.CustomerID, c.FirstName, c.LastName

) AS RankedCustomers WHERE CustomerRank = 1;



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

)

SELECT C.FirstName, C.LastName, TotalSpending FROM Customers C

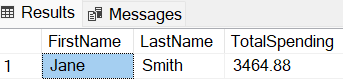
JOIN (

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;



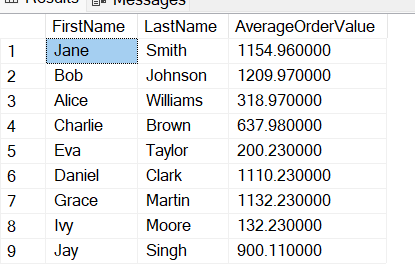
SELECT c.FirstName, c.LastName, AVG(OrderValue) AS AverageOrderValue FROM Customers c

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;

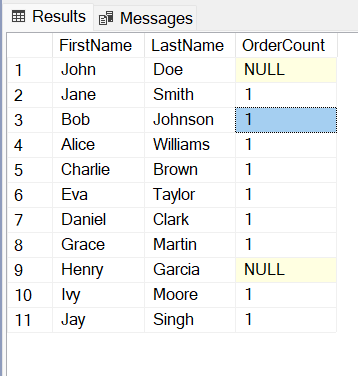


SELECT FirstName, LastName, OrderCount FROM Customers c

LEFT JOIN (

SELECT CustomerID, COUNT(OrderID) AS OrderCount FROM Orders

GROUP BY CustomerID

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