TECHSHOP, AN ELECTRONIC GADGETS SHOP

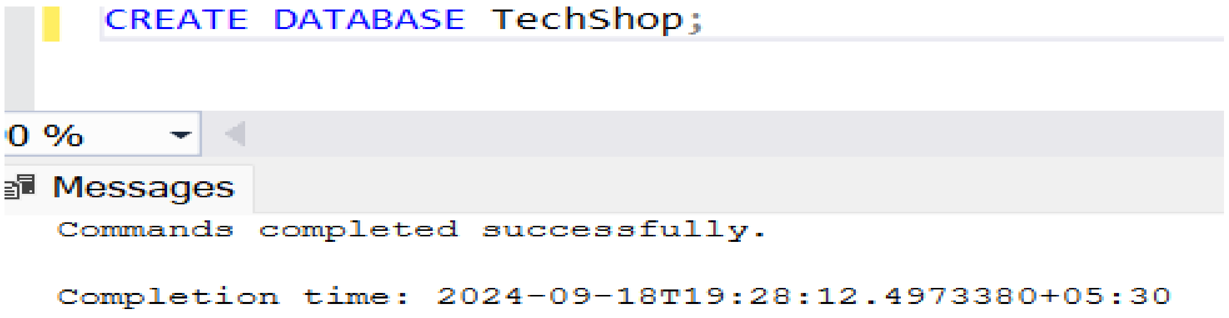
PYTHON ASSIGNMENT

HARSHINI V

TASK:1. DATABASE DESIGN:

1. Create the database named "TechShop"

CREATE DATABASE TechShop;

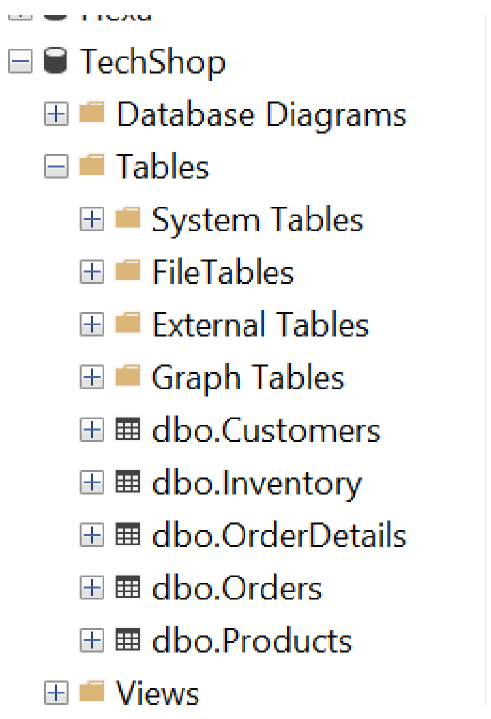


```
CREATE DATABASE TechShop;
```

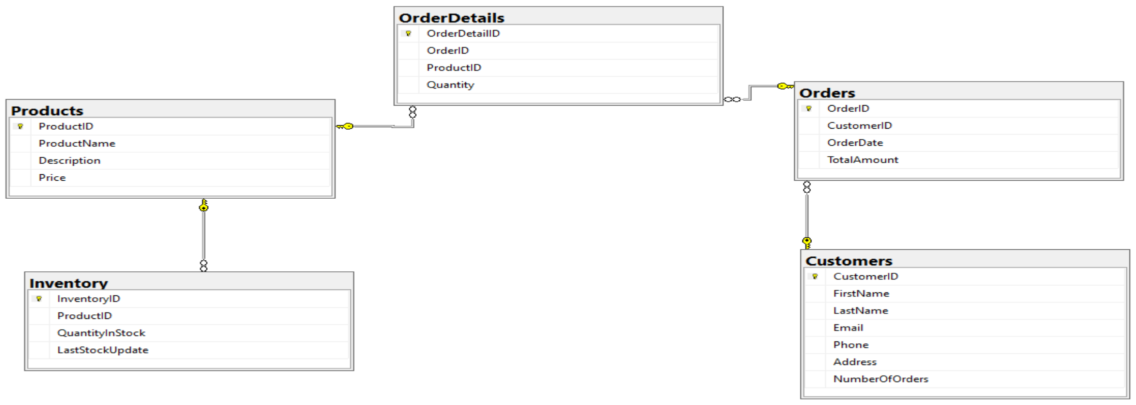
```
Commands completed successfully.

Completion time: 2024-09-18T19:28:12.4973380+05:30
```

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

```
CREATE TABLE Customers (
    CustomerID INT IDENTITY PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    Phone VARCHAR(15),
    Address VARCHAR(255)
);
CREATE TABLE Products (
    ProductID INT IDENTITY PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL,
    Description TEXT,
    Price DECIMAL(10, 2) NOT NULL
);
CREATE TABLE Orders (
    OrderID INT IDENTITY PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    TotalAmount DECIMAL(10, 2),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
CREATE TABLE OrderDetails (
    OrderDetailID INT IDENTITY 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 IDENTITY PRIMARY KEY,
    ProductID INT,
    QuantityInStock INT,
    LastStockUpdate DATE,
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```



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



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

Primary keys: CustomerID, ProductID, OrderID, OrderDetailID, InventoryID

Foreign keys:

- Orders.CustomerID references Customers.CustomerID
- OrderDetails.OrderID references Orders.OrderID
- OrderDetails.ProductID references Products.ProductID
- Inventory.ProductID references Products.ProductID

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

a. Customers

```
INSERT INTO Customers VALUES
('John', 'Doe', 'john.doe@example.com', '1234567890', '123 Main St'),
('Jane', 'Smith', 'jane.smith@example.com', '0987654321', '456 Maple Ave'),
('Alice', 'Johnson', 'alice.johnson@example.com', '5555555555', '789 Oak St'),
('Bob', 'Brown', 'bob.brown@example.com', '6666666666', '101 Pine St'),
('Charlie', 'Davis', 'charlie.davis@example.com', '7777777777', '102 Cedar Ave'),
('Emily', 'Clark', 'emily.clark@example.com', '8888888888', '202 Birch Blvd'),
('David', 'Garcia', 'david.garcia@example.com', '9999999999', '303 Spruce Lane'),
('Sophia', 'Martinez', 'sophia.martinez@example.com', '2222222222', '404 Walnut Way'),
('Liam', 'Miller', 'liam.miller@example.com', '3333333333', '505 Elm Dr'),
('Mia', 'Wilson', 'mia.wilson@example.com', '4444444444', '606 Cherry Ct');
```

b. Products

```
INSERT INTO Products VALUES
('Laptop', 'High-end gaming laptop', 1500.00),
('Smartphone', 'Latest model smartphone', 800.00),
('Tablet', '10-inch tablet', 400.00),
('Smartwatch', 'Fitness tracking smartwatch', 200.00),
('Headphones', 'Noise-cancelling headphones', 150.00),
('Keyboard', 'Mechanical keyboard', 100.00),
('Monitor', '27-inch 4K monitor', 300.00),
('Mouse', 'Wireless gaming mouse', 50.00),
('Printer', 'Laser printer', 250.00),
('Camera', 'Digital SLR camera', 1200.00);
```

c. Orders

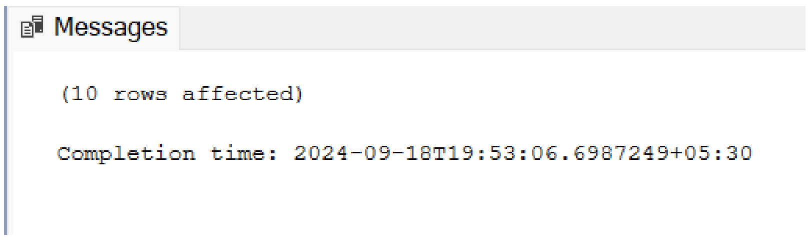
```
INSERT INTO Orders VALUES
(1, '2024-09-10', 2300.00),
(2, '2024-09-11', 950.00),
(3, '2024-09-12', 600.00),
(4, '2024-09-13', 200.00),
(5, '2024-09-14', 1550.00),
(6, '2024-09-15', 450.00),
(7, '2024-09-16', 500.00),
(8, '2024-09-17', 1200.00),
(9, '2024-09-18', 3000.00),
(10, '2024-09-19', 1350.00);
```

d. OrderDetails

```
INSERT INTO OrderDetails VALUES
(1, 1, 1),
(1, 2, 2),
(2, 3, 1),
(3, 4, 1),
(4, 5, 1),
(5, 6, 2),
(6, 7, 1),
(7, 8, 3),
(8, 9, 1),
(9, 10, 1);
```

e. Inventory

```
INSERT INTO Inventory VALUES
(1, 50, '2024-09-01'),
(2, 100, '2024-09-01'),
(3, 200, '2024-09-01'),
(4, 150, '2024-09-01'),
(5, 75, '2024-09-01'),
(6, 80, '2024-09-01'),
(7, 120, '2024-09-01'),
(8, 60, '2024-09-01'),
(9, 30, '2024-09-01'),
(10, 40, '2024-09-01');
```

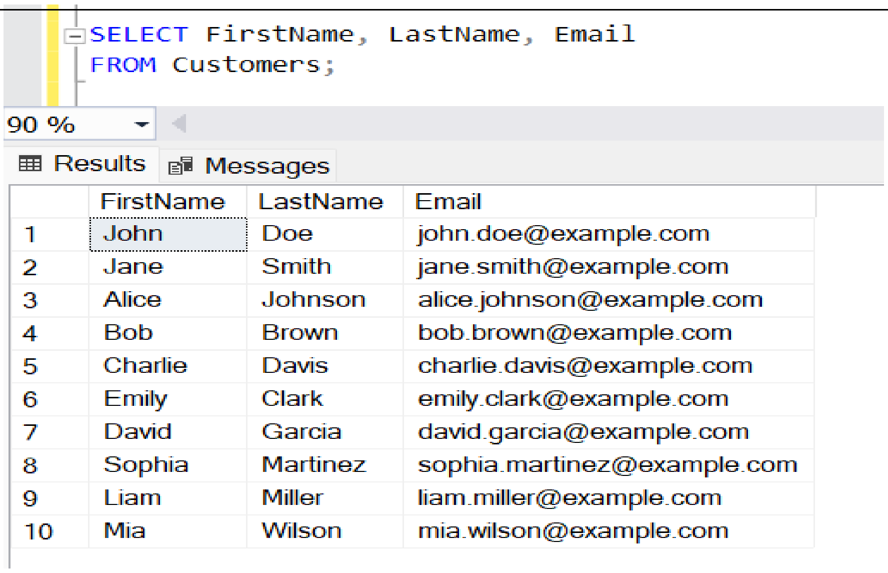


```
(10 rows affected)

Completion time: 2024-09-18T19:53:06.6987249+05:30
```

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;
```



```
SELECT FirstName, LastName, Email
FROM Customers;
```

90 %

Results | Messages

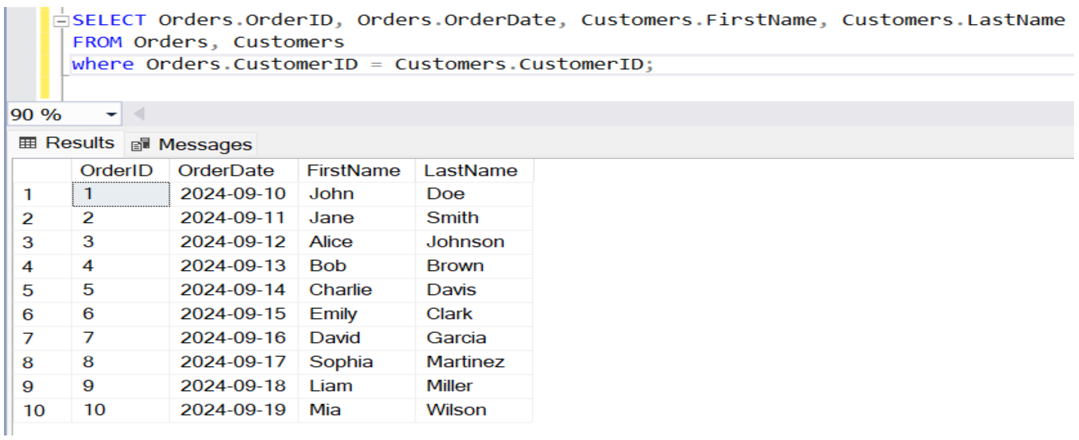
	FirstName	LastName	Email
1	John	Doe	john.doe@example.com
2	Jane	Smith	jane.smith@example.com
3	Alice	Johnson	alice.johnson@example.com
4	Bob	Brown	bob.brown@example.com
5	Charlie	Davis	charlie.davis@example.com
6	Emily	Clark	emily.clark@example.com
7	David	Garcia	david.garcia@example.com
8	Sophia	Martinez	sophia.martinez@example.com
9	Liam	Miller	liam.miller@example.com
10	Mia	Wilson	mia.wilson@example.com

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, Customers

where Orders.CustomerID = Customers.CustomerID;



```
SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName
FROM Orders, Customers
where Orders.CustomerID = Customers.CustomerID;
```

90 %

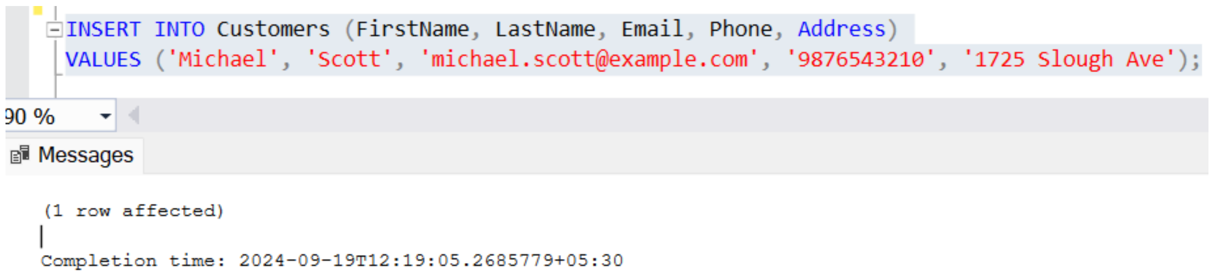
Results | Messages

	OrderID	OrderDate	FirstName	LastName
1	1	2024-09-10	John	Doe
2	2	2024-09-11	Jane	Smith
3	3	2024-09-12	Alice	Johnson
4	4	2024-09-13	Bob	Brown
5	5	2024-09-14	Charlie	Davis
6	6	2024-09-15	Emily	Clark
7	7	2024-09-16	David	Garcia
8	8	2024-09-17	Sophia	Martinez
9	9	2024-09-18	Liam	Miller
10	10	2024-09-19	Mia	Wilson

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 ('Michael', 'Scott', 'michael.scott@example.com', '9876543210', '1725 Slough Ave');

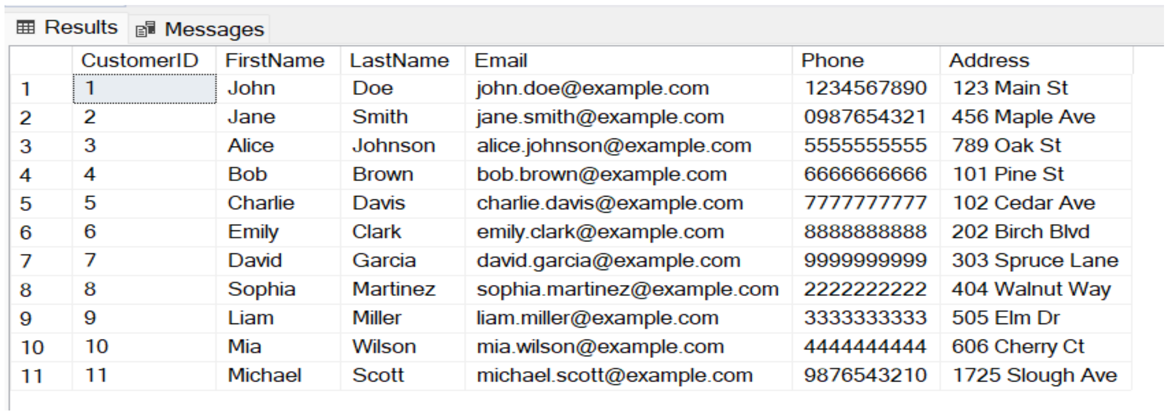


```
INSERT INTO Customers (FirstName, LastName, Email, Phone, Address)
VALUES ('Michael', 'Scott', 'michael.scott@example.com', '9876543210', '1725 Slough Ave');
```

90 %

Messages

```
(1 row affected)

Completion time: 2024-09-19T12:19:05.2685779+05:30
```

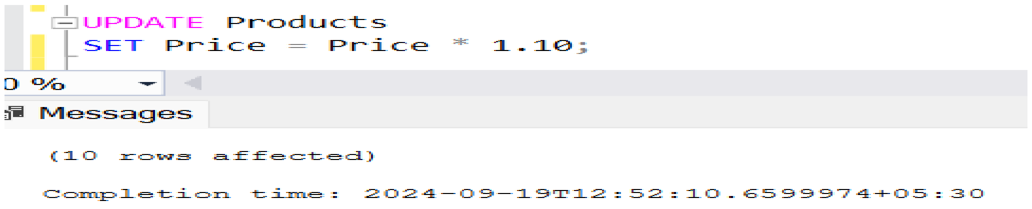


	CustomerID	FirstName	LastName	Email	Phone	Address
1	1	John	Doe	john.doe@example.com	1234567890	123 Main St
2	2	Jane	Smith	jane.smith@example.com	0987654321	456 Maple Ave
3	3	Alice	Johnson	alice.johnson@example.com	5555555555	789 Oak St
4	4	Bob	Brown	bob.brown@example.com	6666666666	101 Pine St
5	5	Charlie	Davis	charlie.davis@example.com	7777777777	102 Cedar Ave
6	6	Emily	Clark	emily.clark@example.com	8888888888	202 Birch Blvd
7	7	David	Garcia	david.garcia@example.com	9999999999	303 Spruce Lane
8	8	Sophia	Martinez	sophia.martinez@example.com	2222222222	404 Walnut Way
9	9	Liam	Miller	liam.miller@example.com	3333333333	505 Elm Dr
10	10	Mia	Wilson	mia.wilson@example.com	4444444444	606 Cherry Ct
11	11	Michael	Scott	michael.scott@example.com	9876543210	1725 Slough Ave

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

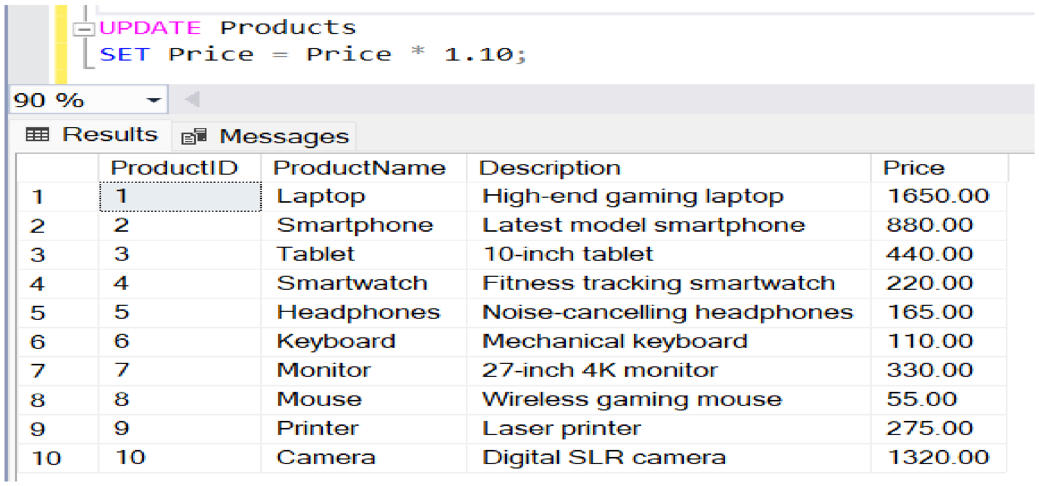
UPDATE Products

SET Price = Price * 1.10;



```
UPDATE Products
SET Price = Price * 1.10;
```

```
(10 rows affected)

Completion time: 2024-09-19T12:52:10.6599974+05:30
```



```
UPDATE Products
SET Price = Price * 1.10;
```

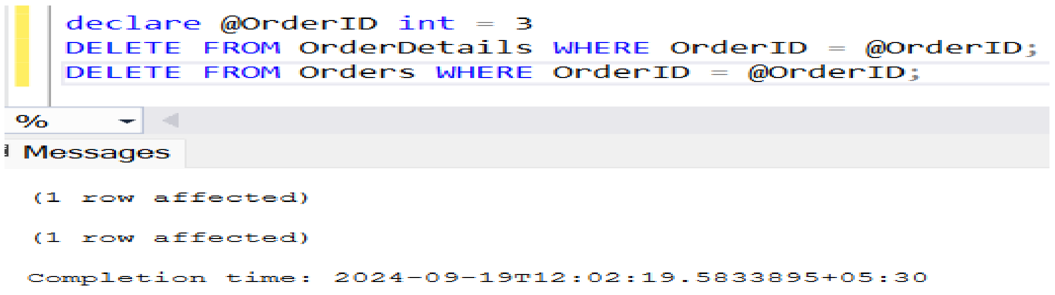
	ProductID	ProductName	Description	Price
1	1	Laptop	High-end gaming laptop	1650.00
2	2	Smartphone	Latest model smartphone	880.00
3	3	Tablet	10-inch tablet	440.00
4	4	Smartwatch	Fitness tracking smartwatch	220.00
5	5	Headphones	Noise-cancelling headphones	165.00
6	6	Keyboard	Mechanical keyboard	110.00
7	7	Monitor	27-inch 4K monitor	330.00
8	8	Mouse	Wireless gaming mouse	55.00
9	9	Printer	Laser printer	275.00
10	10	Camera	Digital SLR camera	1320.00

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.

declare @OrderID int = 3

DELETE FROM OrderDetails WHERE OrderID = @OrderID;

DELETE FROM Orders WHERE OrderID = @OrderID;



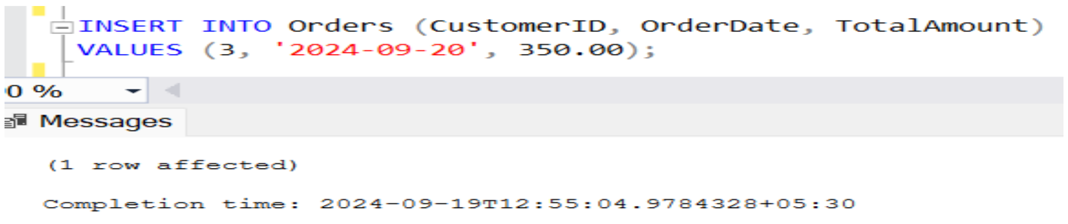
```
declare @OrderID int = 3
DELETE FROM OrderDetails WHERE OrderID = @OrderID;
DELETE FROM Orders WHERE OrderID = @OrderID;
```

```
(1 row affected)

(1 row affected)

Completion time: 2024-09-19T12:02:19.5833895+05:30
```

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.

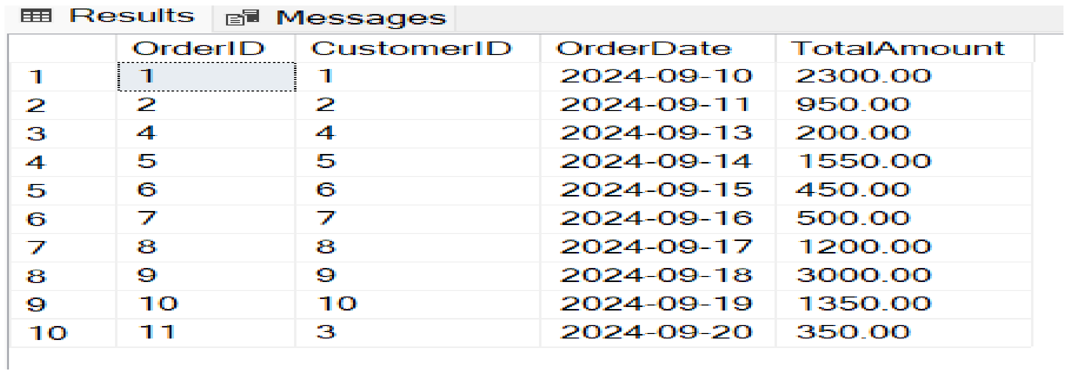
INSERT INTO Orders (CustomerID, OrderDate, TotalAmount)

VALUES (3, '2024-09-20', 350.00);



```
INSERT INTO Orders (CustomerID, OrderDate, TotalAmount)
VALUES (3, '2024-09-20', 350.00);
```

```
(1 row affected)

Completion time: 2024-09-19T12:55:04.9784328+05:30
```



	OrderID	CustomerID	OrderDate	TotalAmount
1	1	1	2024-09-10	2300.00
2	2	2	2024-09-11	950.00
3	4	4	2024-09-13	200.00
4	5	5	2024-09-14	1550.00
5	6	6	2024-09-15	450.00
6	7	7	2024-09-16	500.00
7	8	8	2024-09-17	1200.00
8	9	9	2024-09-18	3000.00
9	10	10	2024-09-19	1350.00
10	11	3	2024-09-20	350.00

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.

declare @NewAddress varchar(20) = '555 Great Avenue'

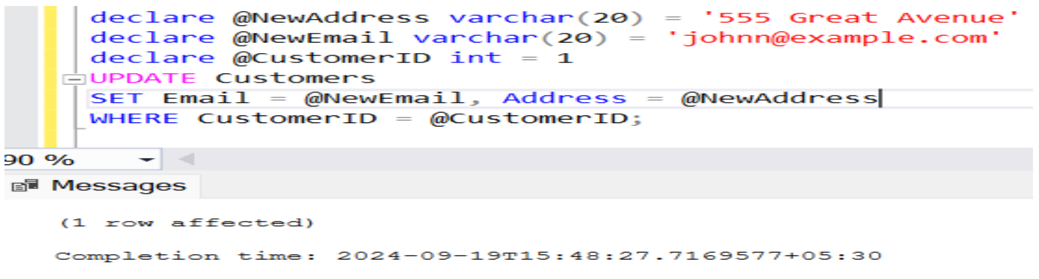
declare @NewEmail varchar(20) = 'johnn@example.com'

declare @CustomerID int = 1

UPDATE Customers

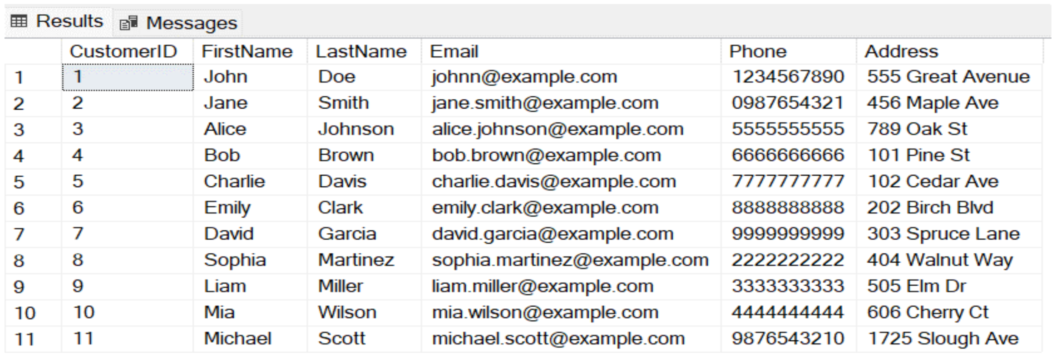
SET Email = @NewEmail, Address = @NewAddress

WHERE CustomerID = @CustomerID;



```
declare @NewAddress varchar(20) = '555 Great Avenue'
declare @NewEmail varchar(20) = 'johnn@example.com'
declare @CustomerID int = 1
UPDATE Customers
SET Email = @NewEmail, Address = @NewAddress
WHERE CustomerID = @CustomerID;
```

```
(1 row affected)

Completion time: 2024-09-19T15:48:27.7169577+05:30
```



	CustomerID	FirstName	LastName	Email	Phone	Address
1	1	John	Doe	johnn@example.com	1234567890	555 Great Avenue
2	2	Jane	Smith	jane.smith@example.com	0987654321	456 Maple Ave
3	3	Alice	Johnson	alice.johnson@example.com	5555555555	789 Oak St
4	4	Bob	Brown	bob.brown@example.com	6666666666	101 Pine St
5	5	Charlie	Davis	charlie.davis@example.com	7777777777	102 Cedar Ave
6	6	Emily	Clark	emily.clark@example.com	8888888888	202 Birch Blvd
7	7	David	Garcia	david.garcia@example.com	9999999999	303 Spruce Lane
8	8	Sophia	Martinez	sophia.martinez@example.com	2222222222	404 Walnut Way
9	9	Liam	Miller	liam.miller@example.com	3333333333	505 Elm Dr
10	10	Mia	Wilson	mia.wilson@example.com	4444444444	606 Cherry Ct
11	11	Michael	Scott	michael.scott@example.com	9876543210	1725 Slough Ave

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

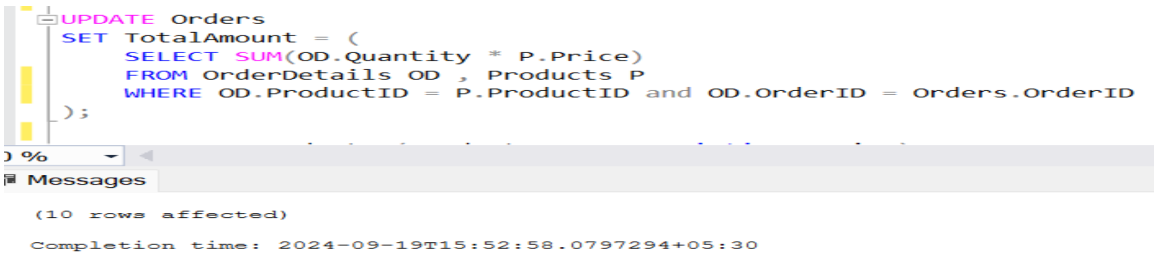
SET TotalAmount = (

  SELECT SUM(OD.Quantity * P.Price)

  FROM OrderDetails OD , Products P

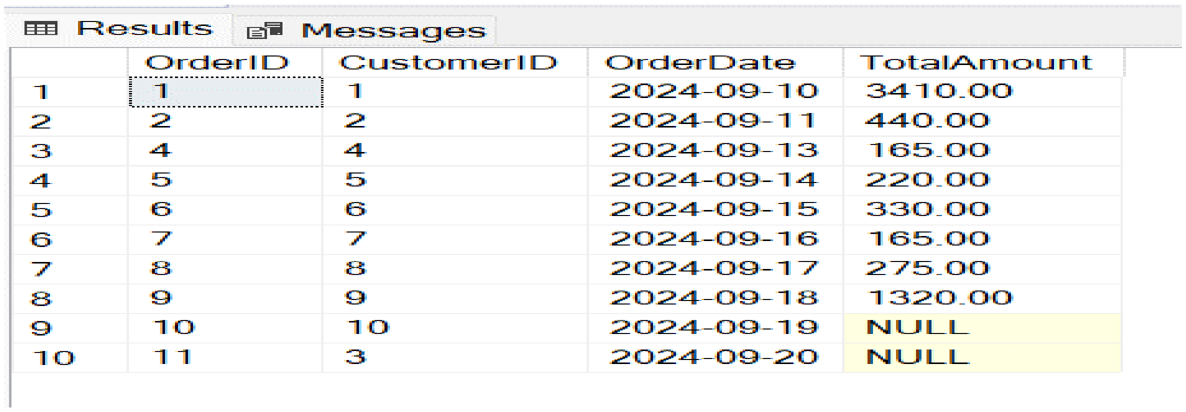
  WHERE OD.ProductID = P.ProductID and OD.OrderID = Orders.OrderID

);



```
UPDATE Orders
SET TotalAmount = (
    SELECT SUM(OD.Quantity * P.Price)
    FROM OrderDetails OD , Products P
    WHERE OD.ProductID = P.ProductID and OD.OrderID = Orders.OrderID
);
```

```
(10 rows affected)

Completion time: 2024-09-19T15:52:58.0797294+05:30
```



	OrderID	CustomerID	OrderDate	TotalAmount
1	1	1	2024-09-10	3410.00
2	2	2	2024-09-11	440.00
3	4	4	2024-09-13	165.00
4	5	5	2024-09-14	220.00
5	6	6	2024-09-15	330.00
6	7	7	2024-09-16	165.00
7	8	8	2024-09-17	275.00
8	9	9	2024-09-18	1320.00
9	10	10	2024-09-19	NULL
10	11	3	2024-09-20	NULL

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.

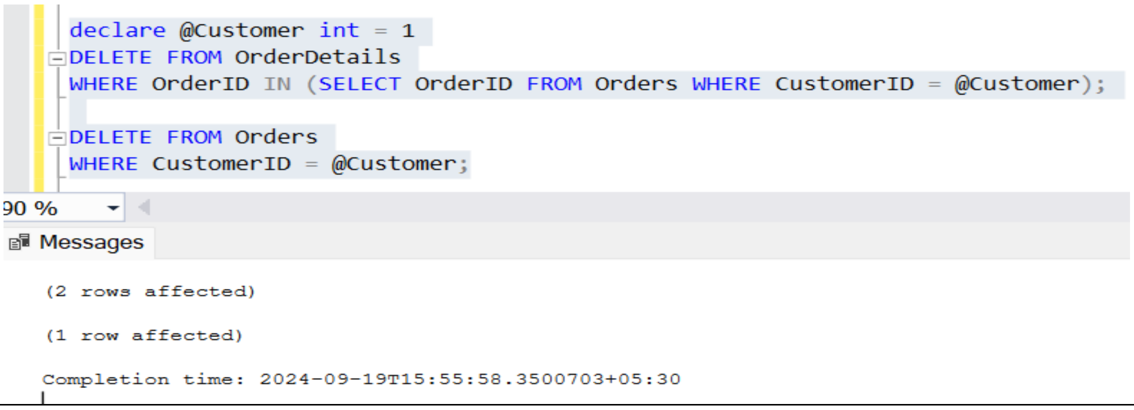
declare @Customer int = 1

DELETE FROM OrderDetails

WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = @Customer);

DELETE FROM Orders

WHERE CustomerID = @Customer;



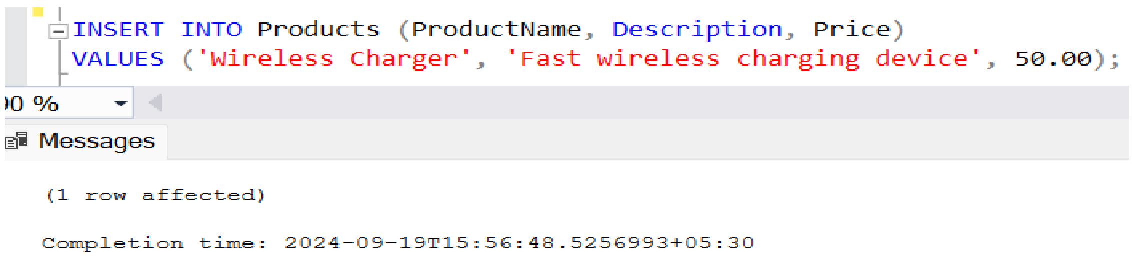
```
declare @Customer int = 1
DELETE FROM OrderDetails
WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = @Customer);

DELETE FROM Orders
WHERE CustomerID = @Customer;
```

```
(2 rows affected)

(1 row affected)

Completion time: 2024-09-19T15:55:58.3500703+05:30
```

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.

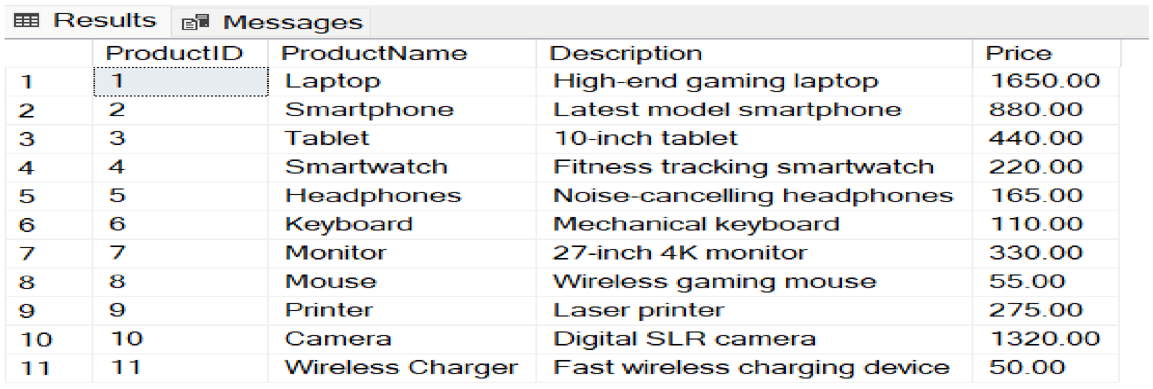
INSERT INTO Products (ProductName, Description, Price)

VALUES ('Wireless Charger', 'Fast wireless charging device', 50.00);



```
INSERT INTO Products (ProductName, Description, Price)
VALUES ('Wireless Charger', 'Fast wireless charging device', 50.00);
```

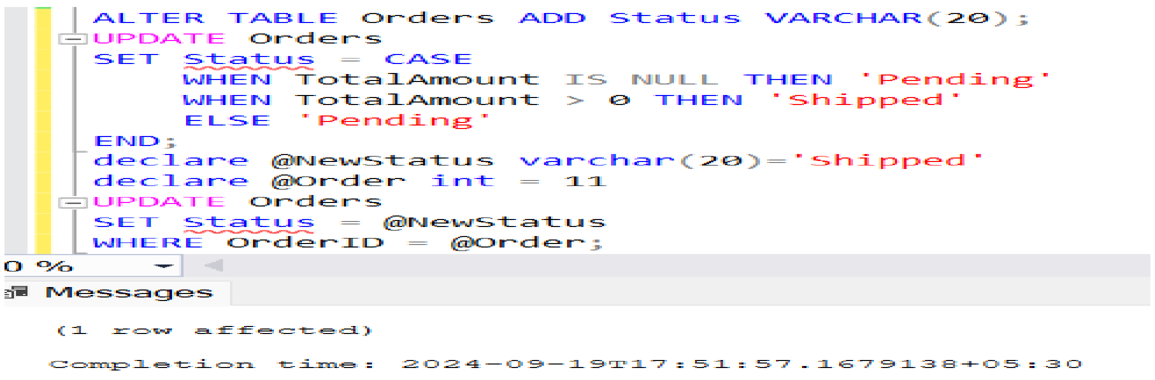
```
(1 row affected)

Completion time: 2024-09-19T15:56:48.5256993+05:30
```



	ProductID	ProductName	Description	Price
1	1	Laptop	High-end gaming laptop	1650.00
2	2	Smartphone	Latest model smartphone	880.00
3	3	Tablet	10-inch tablet	440.00
4	4	Smartwatch	Fitness tracking smartwatch	220.00
5	5	Headphones	Noise-cancelling headphones	165.00
6	6	Keyboard	Mechanical keyboard	110.00
7	7	Monitor	27-inch 4K monitor	330.00
8	8	Mouse	Wireless gaming mouse	55.00
9	9	Printer	Laser printer	275.00
10	10	Camera	Digital SLR camera	1320.00
11	11	Wireless Charger	Fast wireless charging device	50.00

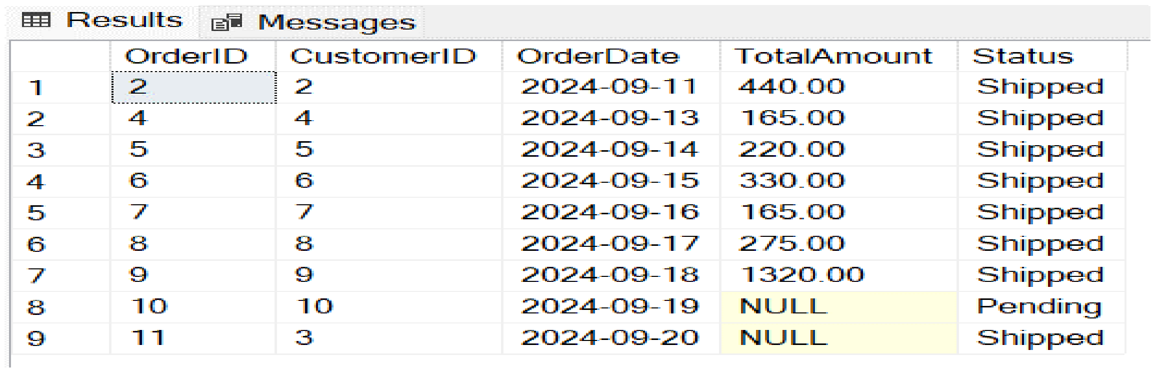
11. Write an SQL query to update the status of a specific order in the "Orders" table (e.g., from "Pending" to "Shipped"). Allow users to input the order ID and the new status.

```
ALTER TABLE Orders ADD Status VARCHAR(20);

UPDATE Orders

SET Status = CASE

    WHEN TotalAmount IS NULL THEN 'Pending'

    WHEN TotalAmount > 0 THEN 'Shipped'

    ELSE 'Pending'

END;

declare @NewStatus varchar(20)='Shipped'

declare @Order int = 11

UPDATE Orders

SET Status = @NewStatus

WHERE OrderID = @Order;
```



```
ALTER TABLE Orders ADD Status VARCHAR(20);
UPDATE Orders
SET Status = CASE
        WHEN TotalAmount IS NULL THEN 'Pending'
        WHEN TotalAmount > 0 THEN 'Shipped'
        ELSE 'Pending'
END;
declare @NewStatus varchar(20)='Shipped'
declare @Order int = 11
UPDATE Orders
SET Status = @NewStatus
WHERE OrderID = @Order;
```

```
(1 row affected)

Completion time: 2024-09-19T17:51:57.1679138+05:30
```



	OrderID	CustomerID	OrderDate	TotalAmount	Status
1	2	2	2024-09-11	440.00	Shipped
2	4	4	2024-09-13	165.00	Shipped
3	5	5	2024-09-14	220.00	Shipped
4	6	6	2024-09-15	330.00	Shipped
5	7	7	2024-09-16	165.00	Shipped
6	8	8	2024-09-17	275.00	Shipped
7	9	9	2024-09-18	1320.00	Shipped
8	10	10	2024-09-19	NULL	Pending
9	11	3	2024-09-20	NULL	Shipped

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.

ALTER TABLE Customers ADD NumberOfOrders INT;

UPDATE Customers

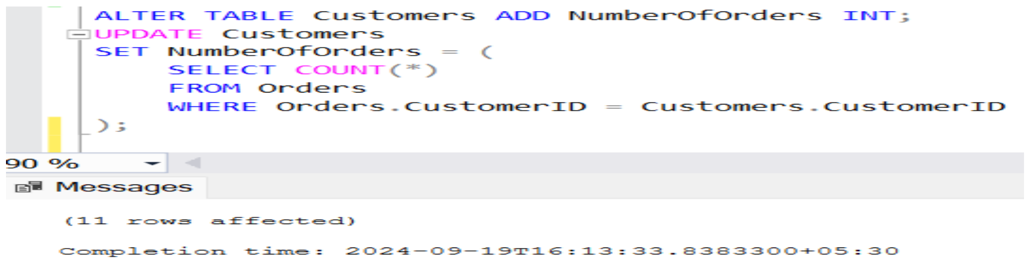
SET NumberOfOrders = (

SELECT COUNT(*)

FROM Orders

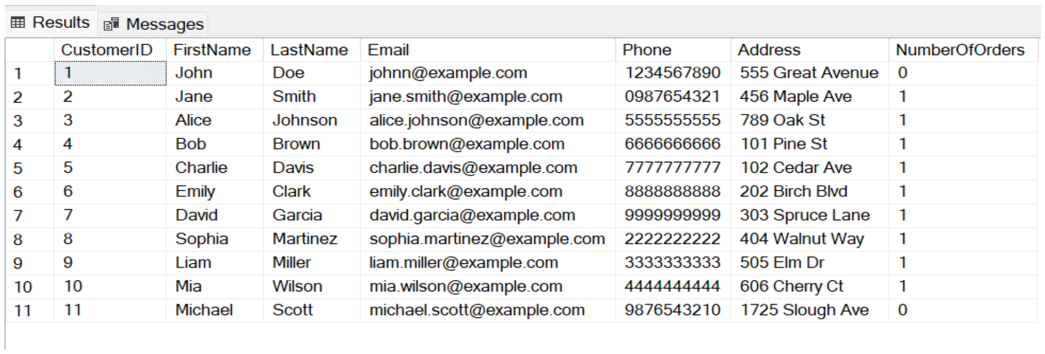
WHERE Orders.CustomerID = Customers.CustomerID

);



```
ALTER TABLE Customers ADD NumberOfOrders INT;
UPDATE Customers
SET NumberOfOrders = (
        SELECT COUNT(*)
        FROM Orders
        WHERE Orders.CustomerID = Customers.CustomerID
);
```

```
(11 rows affected)

Completion time: 2024-09-19T16:13:33.8383300+05:30
```



	CustomerID	FirstName	LastName	Email	Phone	Address	NumberOfOrders
1	1	John	Doe	johnn@example.com	1234567890	555 Great Avenue	0
2	2	Jane	Smith	jane.smith@example.com	0987654321	456 Maple Ave	1
3	3	Alice	Johnson	alice.johnson@example.com	5555555555	789 Oak St	1
4	4	Bob	Brown	bob.brown@example.com	6666666666	101 Pine St	1
5	5	Charlie	Davis	charlie.davis@example.com	7777777777	102 Cedar Ave	1
6	6	Emily	Clark	emily.clark@example.com	8888888888	202 Birch Blvd	1
7	7	David	Garcia	david.garcia@example.com	9999999999	303 Spruce Lane	1
8	8	Sophia	Martinez	sophia.martinez@example.com	2222222222	404 Walnut Way	1
9	9	Liam	Miller	liam.miller@example.com	3333333333	505 Elm Dr	1
10	10	Mia	Wilson	mia.wilson@example.com	4444444444	606 Cherry Ct	1
11	11	Michael	Scott	michael.scott@example.com	9876543210	1725 Slough Ave	0

TASK 3. AGGREGATE FUNCTIONS, HAVING, ORDER BY, GROUPBY AND JOINS:

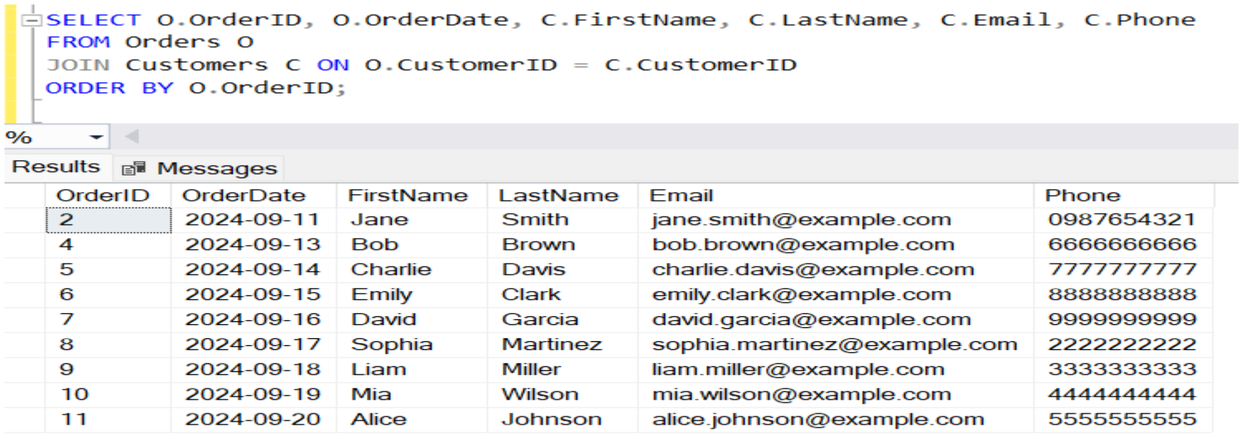
1. Write an SQL query to retrieve a list of all orders along with customer information (e.g., customer name) for each order.

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

FROM Orders O

JOIN Customers C ON O.CustomerID = C.CustomerID

ORDER BY O.OrderID;



```
SELECT O.OrderID, O.OrderDate, C.FirstName, C.LastName, C.Email, C.Phone
FROM Orders O
JOIN Customers C ON O.CustomerID = C.CustomerID
ORDER BY O.OrderID;
```

OrderID	OrderDate	FirstName	LastName	Email	Phone
2	2024-09-11	Jane	Smith	jane.smith@example.com	0987654321
4	2024-09-13	Bob	Brown	bob.brown@example.com	6666666666
5	2024-09-14	Charlie	Davis	charlie.davis@example.com	7777777777
6	2024-09-15	Emily	Clark	emily.clark@example.com	8888888888
7	2024-09-16	David	Garcia	david.garcia@example.com	9999999999
8	2024-09-17	Sophia	Martinez	sophia.martinez@example.com	2222222222
9	2024-09-18	Liam	Miller	liam.miller@example.com	3333333333
10	2024-09-19	Mia	Wilson	mia.wilson@example.com	4444444444
11	2024-09-20	Alice	Johnson	alice.johnson@example.com	5555555555

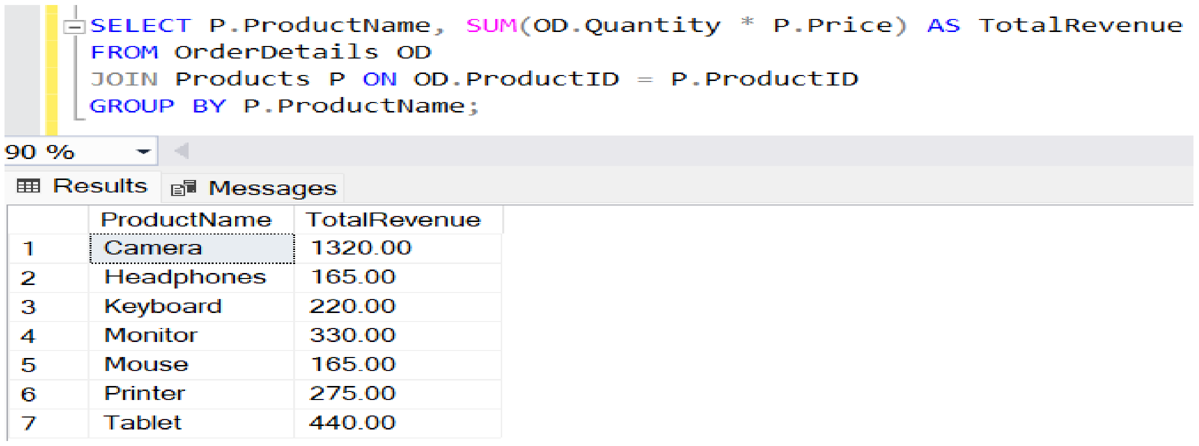
2. Write an SQL query to find the total revenue generated by each electronic gadget product. Include the product name and the total revenue.

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;



```
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;
```

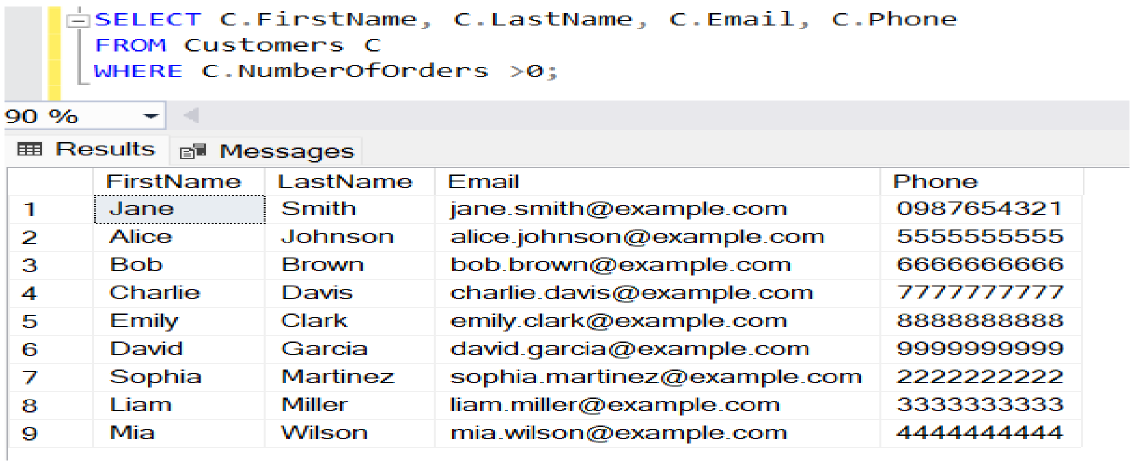
	ProductName	TotalRevenue
1	Camera	1320.00
2	Headphones	165.00
3	Keyboard	220.00
4	Monitor	330.00
5	Mouse	165.00
6	Printer	275.00
7	Tablet	440.00

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

SELECT C.FirstName, C.LastName, C.Email, C.Phone

FROM Customers C

WHERE C.NumberOfOrders >0;



```
SELECT C.FirstName, C.LastName, C.Email, C.Phone
FROM Customers C
WHERE C.NumberOfOrders >0;
```

	FirstName	LastName	Email	Phone
1	Jane	Smith	jane.smith@example.com	0987654321
2	Alice	Johnson	alice.johnson@example.com	5555555555
3	Bob	Brown	bob.brown@example.com	6666666666
4	Charlie	Davis	charlie.davis@example.com	7777777777
5	Emily	Clark	emily.clark@example.com	8888888888
6	David	Garcia	david.garcia@example.com	9999999999
7	Sophia	Martinez	sophia.martinez@example.com	2222222222
8	Liam	Miller	liam.miller@example.com	3333333333
9	Mia	Wilson	mia.wilson@example.com	4444444444

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

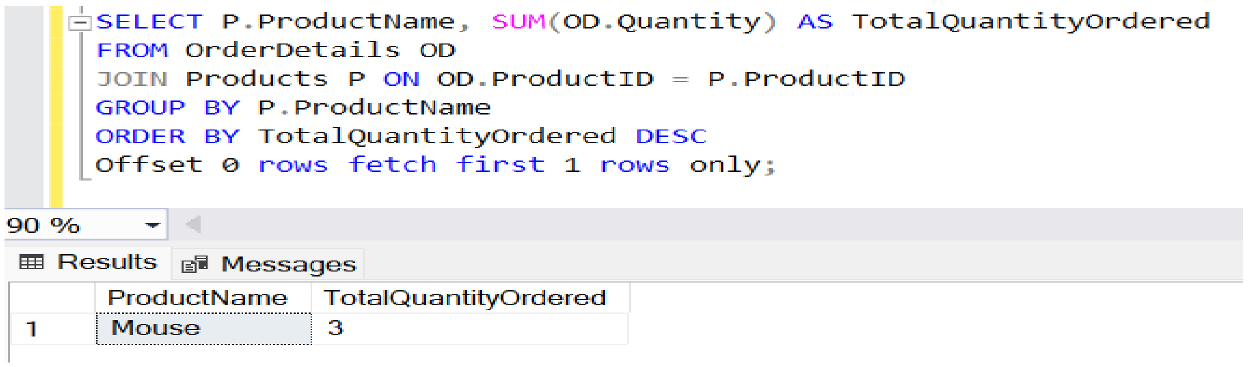
FROM OrderDetails OD

JOIN Products P ON OD.ProductID = P.ProductID

GROUP BY P.ProductName

ORDER BY TotalQuantityOrdered DESC

Offset 0 rows fetch first 1 rows only;



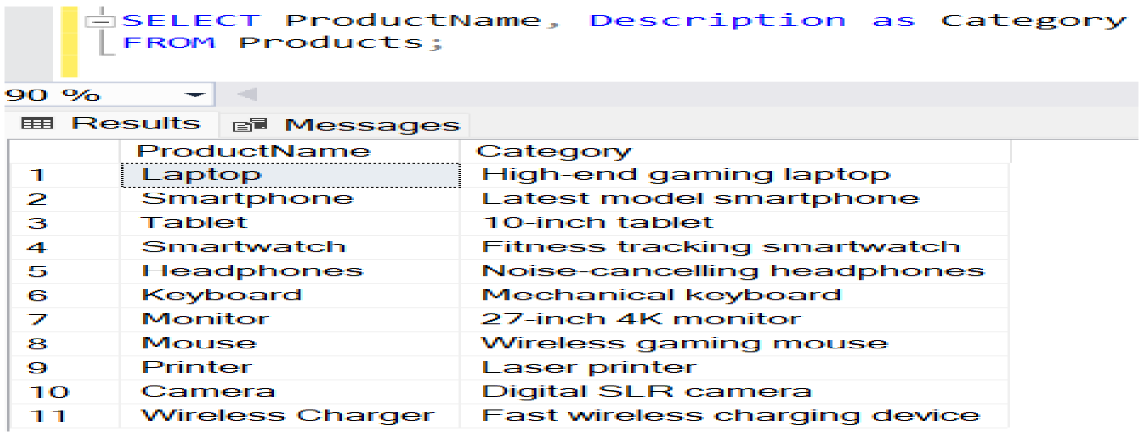
```
SELECT P.ProductName, SUM(OD.Quantity) AS TotalQuantityOrdered
FROM OrderDetails OD
JOIN Products P ON OD.ProductID = P.ProductID
GROUP BY P.ProductName
ORDER BY TotalQuantityOrdered DESC
Offset 0 rows fetch first 1 rows only;
```

	ProductName	TotalQuantityOrdered
1	Mouse	3

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

SELECT ProductName, Description as Category

FROM Products;



```
SELECT ProductName, Description as Category
FROM Products;
```

	ProductName	Category
1	Laptop	High-end gaming laptop
2	Smartphone	Latest model smartphone
3	Tablet	10-inch tablet
4	Smartwatch	Fitness tracking smartwatch
5	Headphones	Noise-cancelling headphones
6	Keyboard	Mechanical keyboard
7	Monitor	27-inch 4K monitor
8	Mouse	Wireless gaming mouse
9	Printer	Laser printer
10	Camera	Digital SLR camera
11	Wireless Charger	Fast wireless charging device

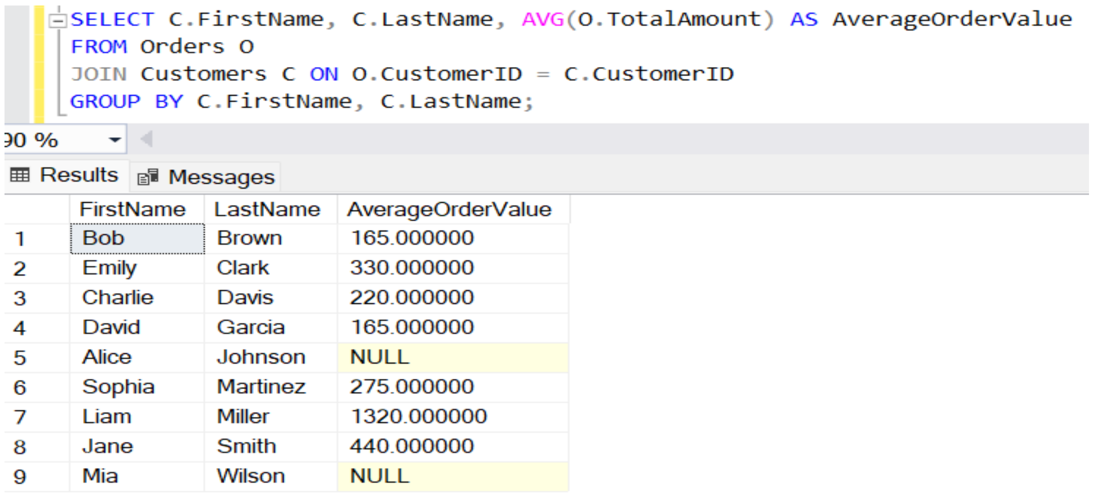
6. Write an SQL query to calculate the average order value for each customer. Include the customer's name and their average order value.

SELECT C.FirstName, C.LastName, AVG(O.TotalAmount) AS AverageOrderValue

FROM Orders O

JOIN Customers C ON O.CustomerID = C.CustomerID

GROUP BY C.FirstName, C.LastName;



```
SELECT C.FirstName, C.LastName, AVG(O.TotalAmount) AS AverageOrderValue
FROM Orders O
JOIN Customers C ON O.CustomerID = C.CustomerID
GROUP BY C.FirstName, C.LastName;
```

	FirstName	LastName	AverageOrderValue
1	Bob	Brown	165.000000
2	Emily	Clark	330.000000
3	Charlie	Davis	220.000000
4	David	Garcia	165.000000
5	Alice	Johnson	NULL
6	Sophia	Martinez	275.000000
7	Liam	Miller	1320.000000
8	Jane	Smith	440.000000
9	Mia	Wilson	NULL

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

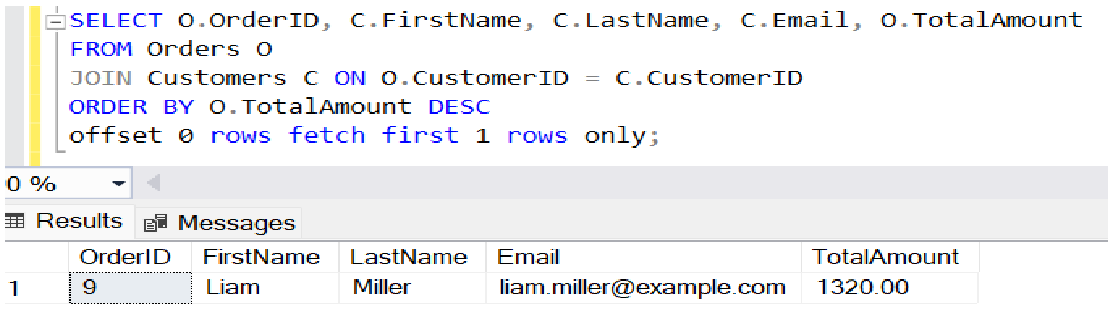
SELECT O.OrderID, C.FirstName, C.LastName, C.Email, O.TotalAmount

FROM Orders O

JOIN Customers C ON O.CustomerID = C.CustomerID

ORDER BY O.TotalAmount DESC

offset 0 rows fetch first 1 rows only;



```
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
offset 0 rows fetch first 1 rows only;
```

	OrderID	FirstName	LastName	Email	TotalAmount
1	9	Liam	Miller	liam.miller@example.com	1320.00

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

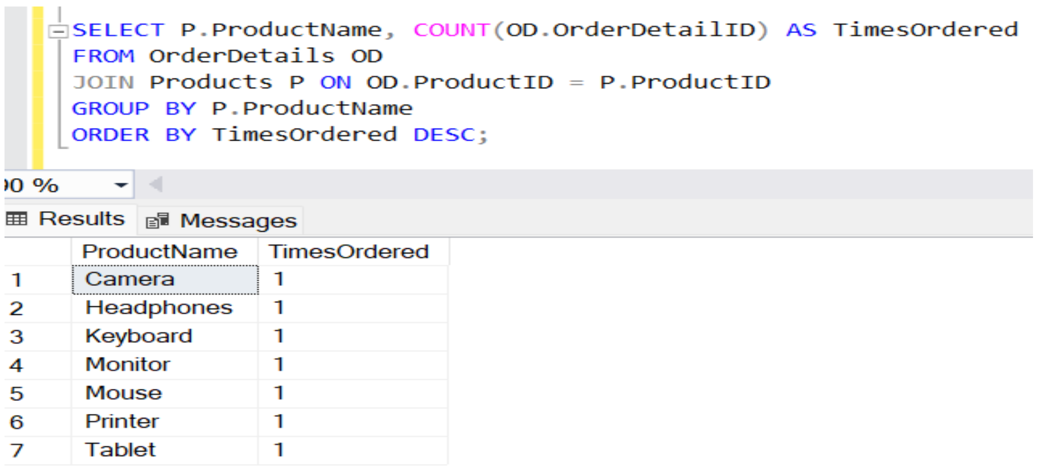
SELECT P.ProductName, COUNT(OD.OrderDetailID) AS TimesOrdered

FROM OrderDetails OD

JOIN Products P ON OD.ProductID = P.ProductID

GROUP BY P.ProductName

ORDER BY TimesOrdered DESC;

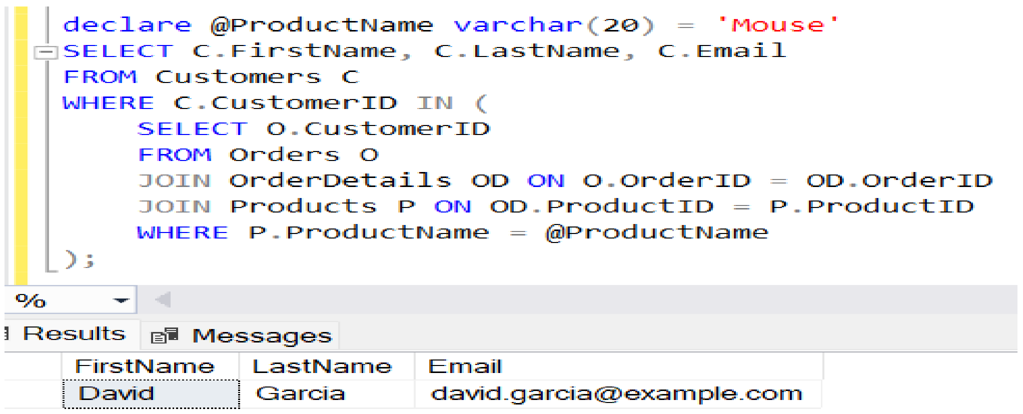


```
SELECT P.ProductName, COUNT(OD.OrderDetailID) AS TimesOrdered
FROM OrderDetails OD
JOIN Products P ON OD.ProductID = P.ProductID
GROUP BY P.ProductName
ORDER BY TimesOrdered DESC;
```

	ProductName	TimesOrdered
1	Camera	1
2	Headphones	1
3	Keyboard	1
4	Monitor	1
5	Mouse	1
6	Printer	1
7	Tablet	1

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.

```
declare @ProductName varchar(20) = 'Mouse'

SELECT C.FirstName, C.LastName, C.Email

FROM Customers C

WHERE C.CustomerID IN (

   SELECT O.CustomerID

   FROM Orders O

   JOIN OrderDetails OD ON O.OrderID = OD.OrderID

   JOIN Products P ON OD.ProductID = P.ProductID

   WHERE P.ProductName = @ProductName

);
```



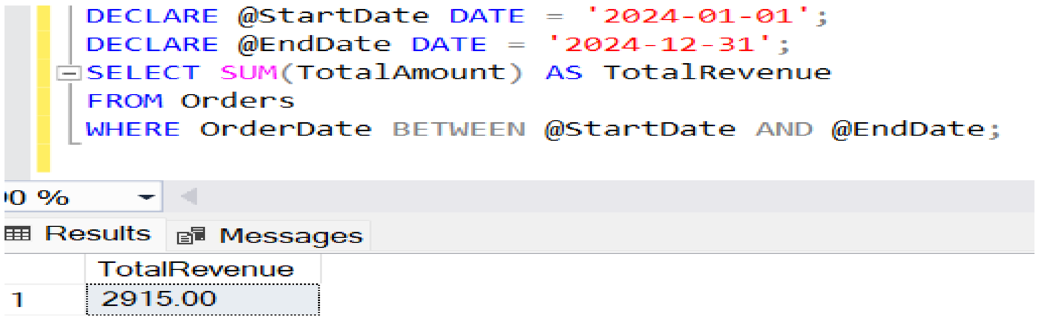
```
declare @ProductName varchar(20) = 'Mouse'
SELECT C.FirstName, C.LastName, C.Email
FROM Customers C
WHERE C.CustomerID IN (
    SELECT O.CustomerID
    FROM Orders O
    JOIN OrderDetails OD ON O.OrderID = OD.OrderID
    JOIN Products P ON OD.ProductID = P.ProductID
    WHERE P.ProductName = @ProductName
);
```

FirstName	LastName	Email
David	Garcia	david.garcia@example.com

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.

```
DECLARE @StartDate DATE = '2024-01-01';

DECLARE @EndDate DATE = '2024-12-31';

SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders
```

WHERE OrderDate BETWEEN @StartDate AND @EndDate;



```
DECLARE @StartDate DATE = '2024-01-01';
DECLARE @EndDate DATE = '2024-12-31';
SELECT SUM(TotalAmount) AS TotalRevenue
FROM Orders
WHERE OrderDate BETWEEN @StartDate AND @EndDate;
```

```
   TotalRevenue
1  2915.00
```

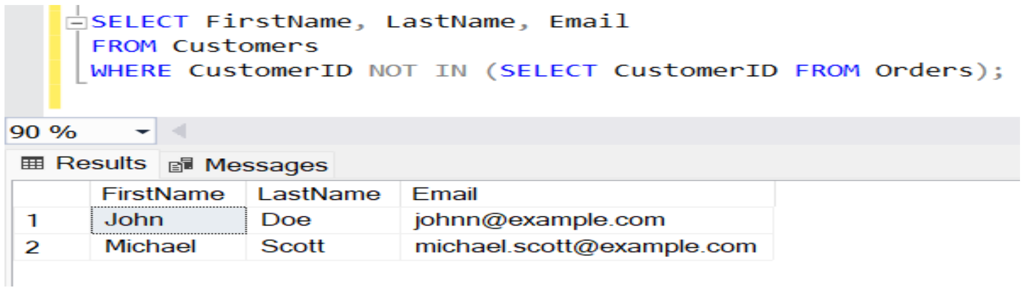
TASK 4. SUBQUERY AND ITS TYPE:

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

SELECT FirstName, LastName, Email

FROM Customers

WHERE CustomerID NOT IN (SELECT CustomerID FROM Orders);



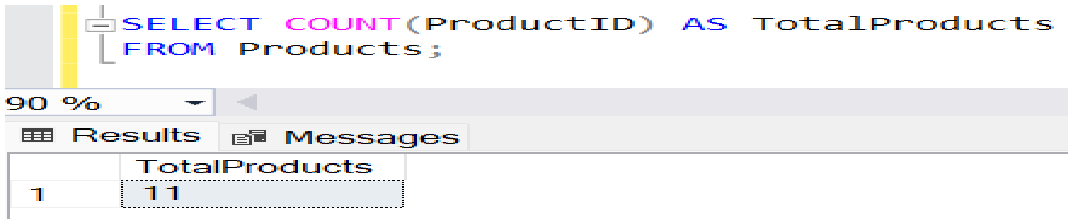
```
SELECT FirstName, LastName, Email
FROM Customers
WHERE CustomerID NOT IN (SELECT CustomerID FROM Orders);
```

```
   FirstName  LastName  Email
1  John       Doe       johnn@example.com
2  Michael    Scott     michael.scott@example.com
```

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

SELECT COUNT(ProductID) AS TotalProducts

FROM Products;



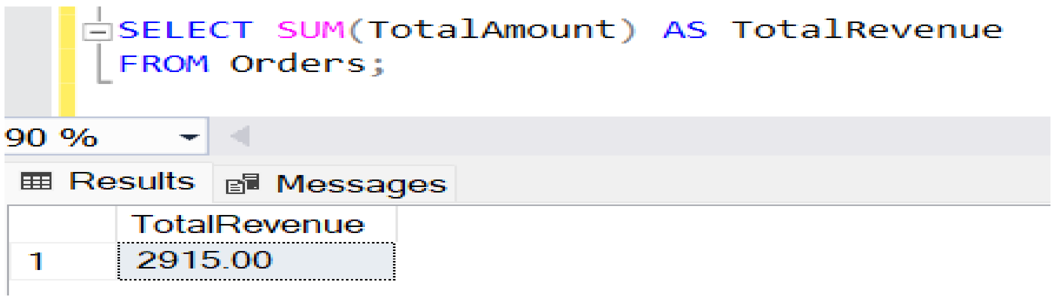
```
SELECT COUNT(ProductID) AS TotalProducts
FROM Products;
```

```
   TotalProducts
1  11
```

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

SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders;



```
SELECT SUM(TotalAmount) AS TotalRevenue
FROM Orders;
```

```
   TotalRevenue
1  2915.00
```

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.

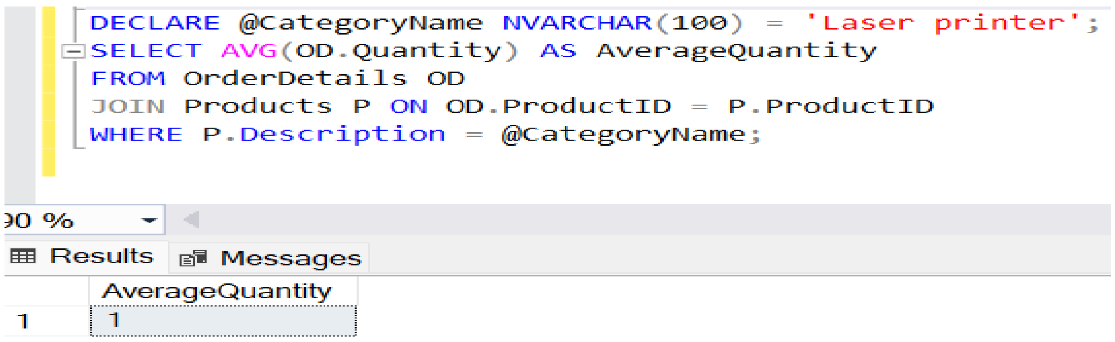
DECLARE @CategoryName NVARCHAR(100) = 'Laser printer';

SELECT AVG(OD.Quantity) AS AverageQuantity

FROM OrderDetails OD

JOIN Products P ON OD.ProductID = P.ProductID

WHERE P.Description = @CategoryName;



```
DECLARE @CategoryName NVARCHAR(100) = 'Laser printer';
SELECT AVG(OD.Quantity) AS AverageQuantity
FROM OrderDetails OD
JOIN Products P ON OD.ProductID = P.ProductID
WHERE P.Description = @CategoryName;
```

```
   AverageQuantity
1  1
```

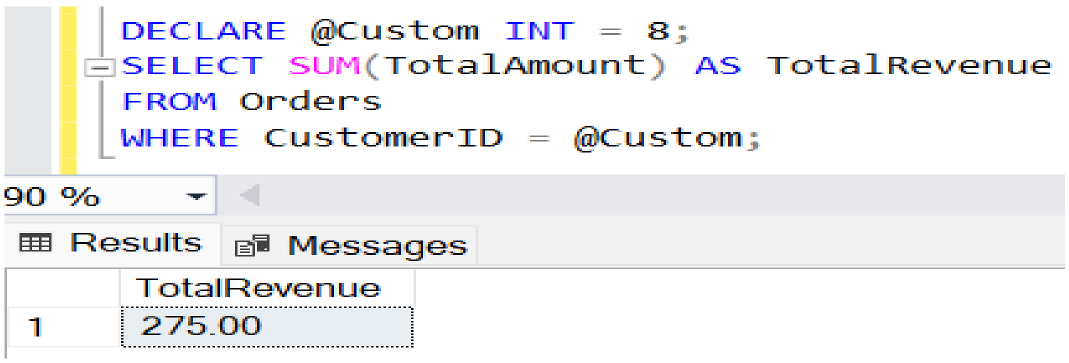
5. Write an SQL query to calculate the total revenue generated by a specific customer. Allow users to input the customer ID as a parameter.

DECLARE @Custom INT = 8;

SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders

WHERE CustomerID = @Custom;



```
DECLARE @Custom INT = 8;
SELECT SUM(TotalAmount) AS TotalRevenue
FROM Orders
WHERE CustomerID = @Custom;
```

```
   TotalRevenue
1  275.00
```

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.

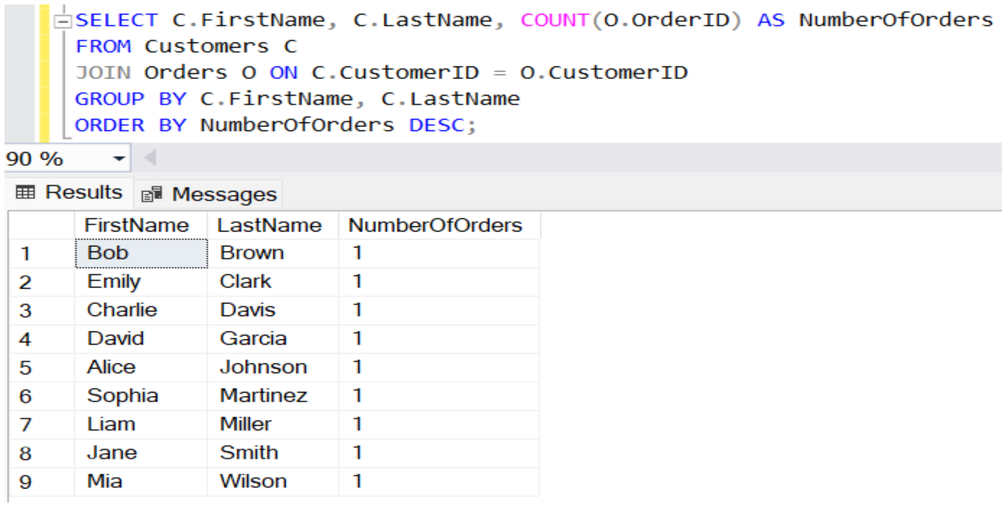
SELECT C.FirstName, C.LastName, COUNT(O.OrderID) AS NumberOfOrders

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

GROUP BY C.FirstName, C.LastName

ORDER BY NumberOfOrders DESC;



```
SELECT C.FirstName, C.LastName, COUNT(O.OrderID) AS NumberOfOrders
FROM Customers C
JOIN Orders O ON C.CustomerID = O.CustomerID
GROUP BY C.FirstName, C.LastName
ORDER BY NumberOfOrders DESC;
```

```
90 %
Results   Messages
```

	FirstName	LastName	NumberOfOrders
1	Bob	Brown	1
2	Emily	Clark	1
3	Charlie	Davis	1
4	David	Garcia	1
5	Alice	Johnson	1
6	Sophia	Martinez	1
7	Liam	Miller	1
8	Jane	Smith	1
9	Mia	Wilson	1

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.Description, SUM(OD.Quantity) AS TotalQuantityOrdered

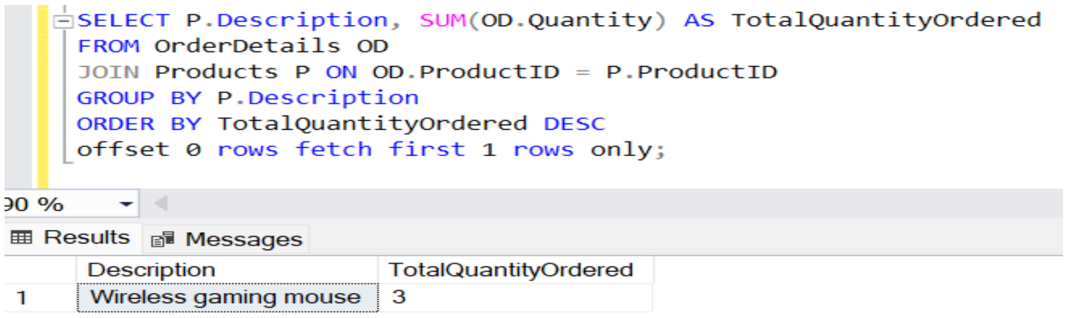
FROM OrderDetails OD

JOIN Products P ON OD.ProductID = P.ProductID

GROUP BY P.Description

ORDER BY TotalQuantityOrdered DESC

offset 0 rows fetch first 1 rows only;



```
SELECT P.Description, SUM(OD.Quantity) AS TotalQuantityOrdered
FROM OrderDetails OD
JOIN Products P ON OD.ProductID = P.ProductID
GROUP BY P.Description
ORDER BY TotalQuantityOrdered DESC
offset 0 rows fetch first 1 rows only;
```

```
90 %
Results   Messages
```

	Description	TotalQuantityOrdered
1	Wireless gaming mouse	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.FirstName, C.LastName, SUM(OD.Quantity * P.Price) AS TotalSpent

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

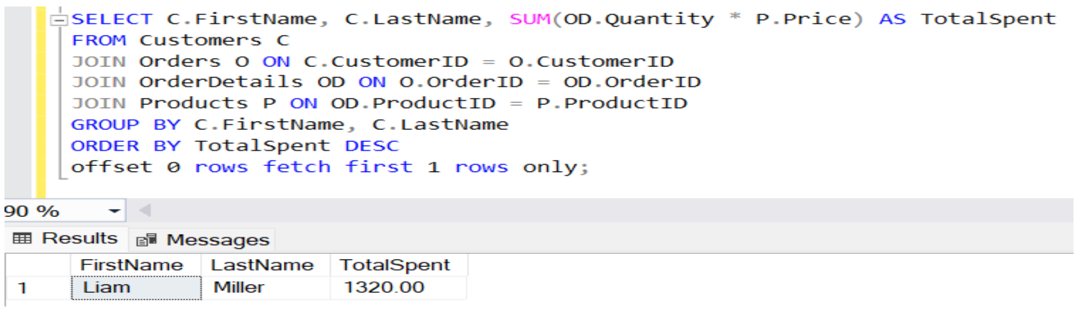
JOIN OrderDetails OD ON O.OrderID = OD.OrderID

JOIN Products P ON OD.ProductID = P.ProductID

GROUP BY C.FirstName, C.LastName

ORDER BY TotalSpent DESC

offset 0 rows fetch first 1 rows only;



```sql
SELECT C.FirstName, C.LastName, SUM(OD.Quantity * P.Price) AS TotalSpent
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
GROUP BY C.FirstName, C.LastName
ORDER BY TotalSpent DESC
offset 0 rows fetch first 1 rows only;
```

	FirstName	LastName	TotalSpent
1	Liam	Miller	1320.00

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

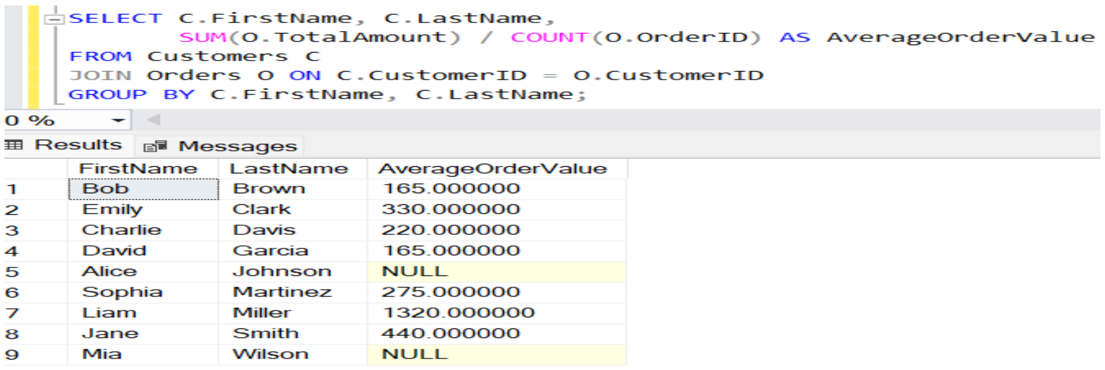
SELECT C.FirstName, C.LastName,

SUM(O.TotalAmount) / COUNT(O.OrderID) AS AverageOrderValue

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

GROUP BY C.FirstName, C.LastName;



```sql
SELECT C.FirstName, C.LastName,
       SUM(O.TotalAmount) / COUNT(O.OrderID) AS AverageOrderValue
FROM Customers C
JOIN Orders O ON C.CustomerID = O.CustomerID
GROUP BY C.FirstName, C.LastName;
```

	FirstName	LastName	AverageOrderValue
1	Bob	Brown	165.000000
2	Emily	Clark	330.000000
3	Charlie	Davis	220.000000
4	David	Garcia	165.000000
5	Alice	Johnson	NULL
6	Sophia	Martinez	275.000000
7	Liam	Miller	1320.000000
8	Jane	Smith	440.000000
9	Mia	Wilson	NULL

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.

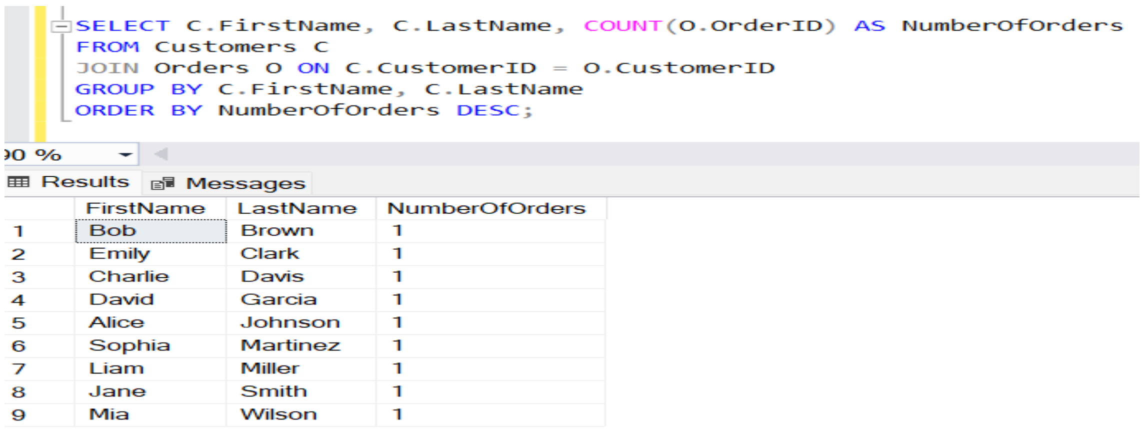
SELECT C.FirstName, C.LastName, COUNT(O.OrderID) AS NumberOfOrders

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

GROUP BY C.FirstName, C.LastName

ORDER BY NumberOfOrders DESC;



```sql
SELECT C.FirstName, C.LastName, COUNT(O.OrderID) AS NumberOfOrders
FROM Customers C
JOIN Orders O ON C.CustomerID = O.CustomerID
GROUP BY C.FirstName, C.LastName
ORDER BY NumberOfOrders DESC;
```

	FirstName	LastName	NumberOfOrders
1	Bob	Brown	1
2	Emily	Clark	1
3	Charlie	Davis	1
4	David	Garcia	1
5	Alice	Johnson	1
6	Sophia	Martinez	1
7	Liam	Miller	1
8	Jane	Smith	1
9	Mia	Wilson	1