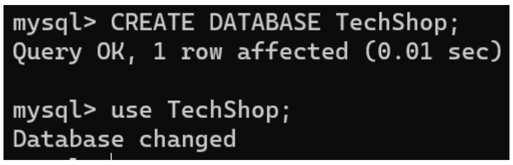
Assignment – 01

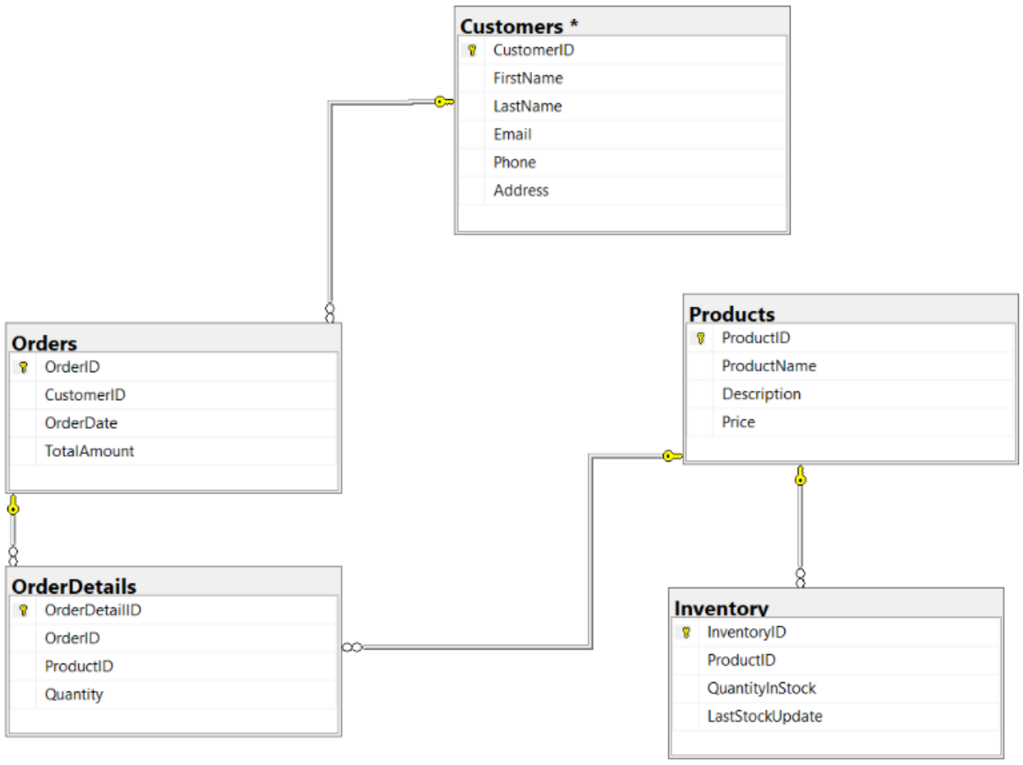
Task-1 Database Design:

1. Create the database name "TechShop".

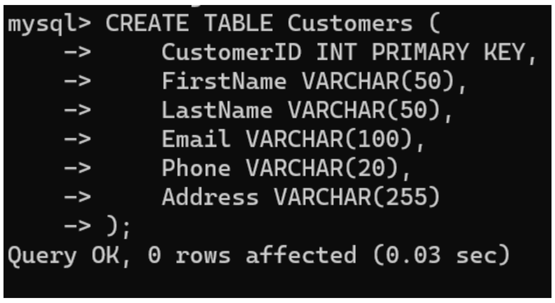


```
mysql> CREATE DATABASE TechShop;
Query OK, 1 row affected (0.01 sec)

mysql> use TechShop;
Database changed
```

2. ERD.

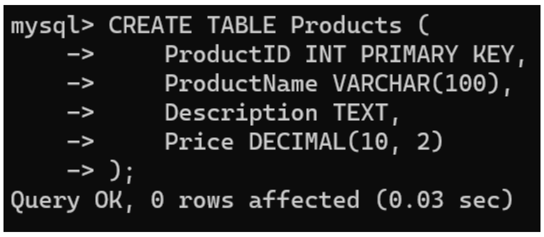


3. Define the schema for the Customers, Products, Orders, OrderDetails and Inventory tables based on the provided schema. Create appropriate Primary Key and Foreign Key constraints for referential integrity. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.
    a) Customers:



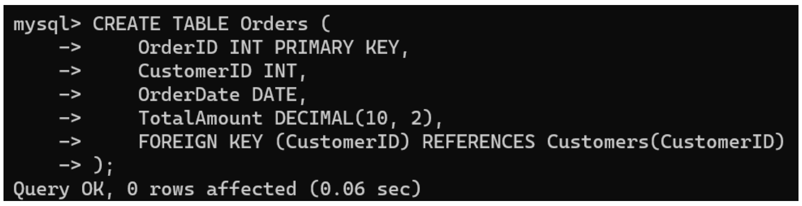
```
mysql> CREATE TABLE Customers (
    ->     CustomerID INT PRIMARY KEY,
    ->     FirstName VARCHAR(50),
    ->     LastName VARCHAR(50),
    ->     Email VARCHAR(100),
    ->     Phone VARCHAR(20),
    ->     Address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

b) Products:



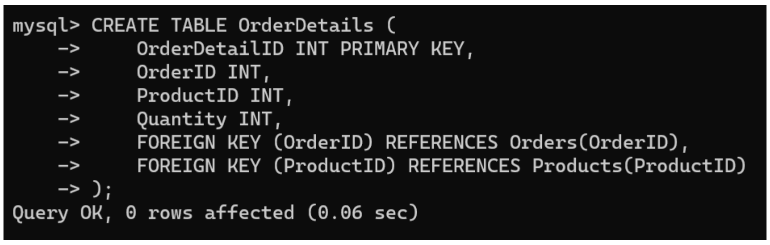
```
mysql> CREATE TABLE Products (
    ->     ProductID INT PRIMARY KEY,
    ->     ProductName VARCHAR(100),
    ->     Description TEXT,
    ->     Price DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

c) Orders:



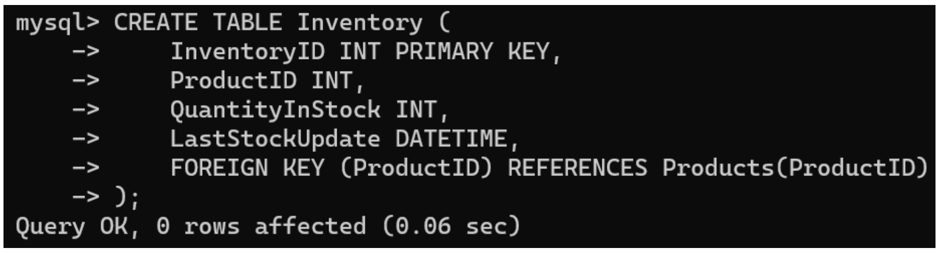
```
mysql> CREATE TABLE Orders (
    ->     OrderID INT PRIMARY KEY,
    ->     CustomerID INT,
    ->     OrderDate DATE,
    ->     TotalAmount DECIMAL(10, 2),
    ->     FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

d) OrderDetails:



```
mysql> 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)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

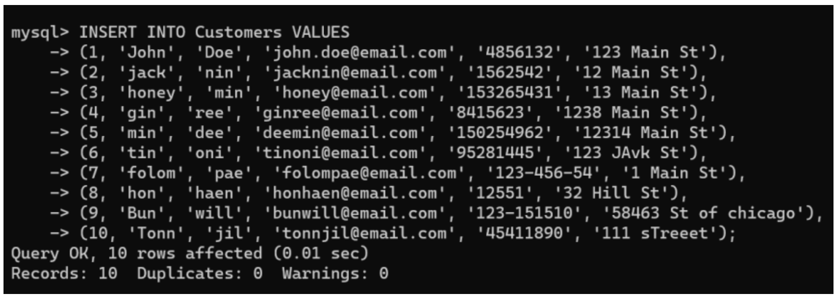
e) Inventory:



```
mysql> CREATE TABLE Inventory (
    ->     InventoryID INT PRIMARY KEY,
    ->     ProductID INT,
    ->     QuantityInStock INT,
    ->     LastStockUpdate DATETIME,
    ->     FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

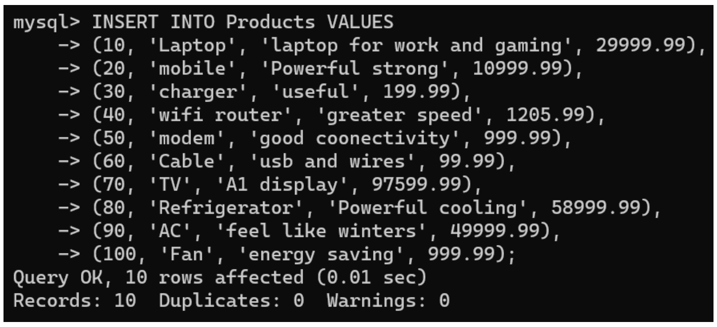
4. Insert at least 10 sample records into each of the following tables.

a) Customers:



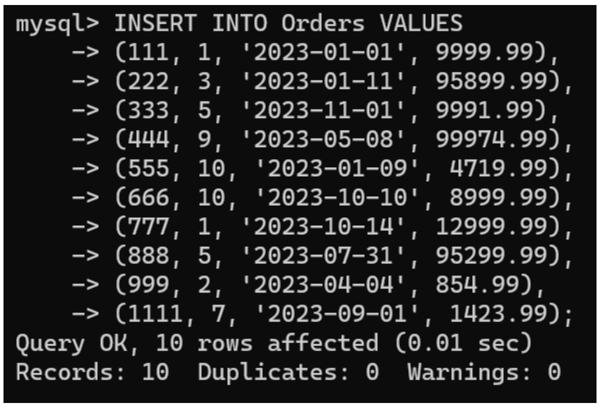
```
mysql> INSERT INTO Customers VALUES
    -> (1, 'John', 'Doe', 'john.doe@email.com', '4856132', '123 Main St'),
    -> (2, 'jack', 'nin', 'jacknin@email.com', '1562542', '12 Main St'),
    -> (3, 'honey', 'min', 'honey@email.com', '153265431', '13 Main St'),
    -> (4, 'gin', 'ree', 'ginree@email.com', '8415623', '1238 Main St'),
    -> (5, 'min', 'dee', 'deemin@email.com', '150254962', '12314 Main St'),
    -> (6, 'tin', 'oni', 'tinoni@email.com', '95281445', '123 JAvk St'),
    -> (7, 'folom', 'pae', 'folompae@email.com', '123-456-54', '1 Main St'),
    -> (8, 'hon', 'haen', 'honhaen@email.com', '12551', '32 Hill St'),
    -> (9, 'Bun', 'will', 'bunwill@email.com', '123-151510', '58463 St of chicago'),
    -> (10, 'Tonn', 'jil', 'tonnjil@email.com', '45411890', '111 sTreeet');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

b) Products:



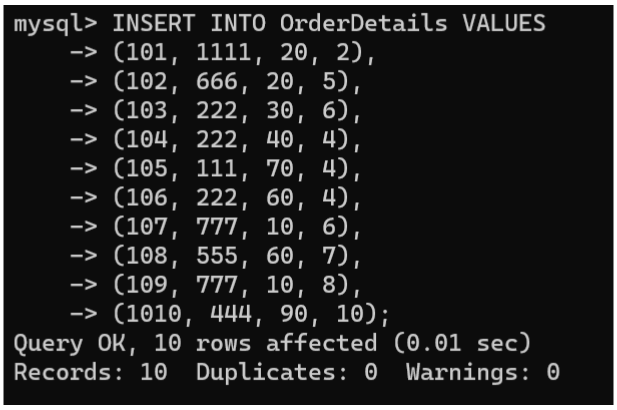
```
mysql> INSERT INTO Products VALUES
    -> (10, 'Laptop', 'laptop for work and gaming', 29999.99),
    -> (20, 'mobile', 'Powerful strong', 10999.99),
    -> (30, 'charger', 'useful', 199.99),
    -> (40, 'wifi router', 'greater speed', 1205.99),
    -> (50, 'modem', 'good coonectivity', 999.99),
    -> (60, 'Cable', 'usb and wires', 99.99),
    -> (70, 'TV', 'A1 display', 97599.99),
    -> (80, 'Refrigerator', 'Powerful cooling', 58999.99),
    -> (90, 'AC', 'feel like winters', 49999.99),
    -> (100, 'Fan', 'energy saving', 999.99);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

c) Orders:



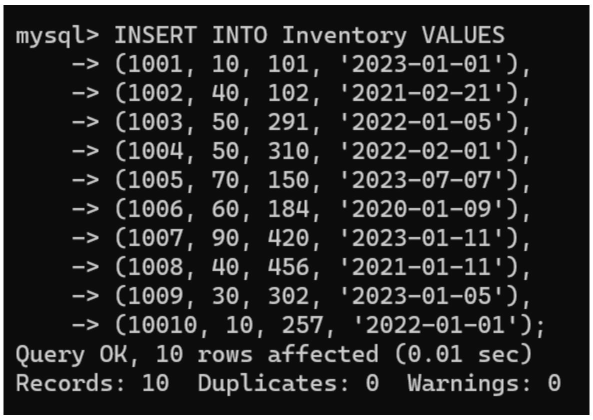
```
mysql> INSERT INTO Orders VALUES
    -> (111, 1, '2023-01-01', 9999.99),
    -> (222, 3, '2023-01-11', 95899.99),
    -> (333, 5, '2023-11-01', 9991.99),
    -> (444, 9, '2023-05-08', 99974.99),
    -> (555, 10, '2023-01-09', 4719.99),
    -> (666, 10, '2023-10-10', 8999.99),
    -> (777, 1, '2023-10-14', 12999.99),
    -> (888, 5, '2023-07-31', 95299.99),
    -> (999, 2, '2023-04-04', 854.99),
    -> (1111, 7, '2023-09-01', 1423.99);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

d) OrderDetails:



```
mysql> INSERT INTO OrderDetails VALUES
    -> (101, 1111, 20, 2),
    -> (102, 666, 20, 5),
    -> (103, 222, 30, 6),
    -> (104, 222, 40, 4),
    -> (105, 111, 70, 4),
    -> (106, 222, 60, 4),
    -> (107, 777, 10, 6),
    -> (108, 555, 60, 7),
    -> (109, 777, 10, 8),
    -> (1010, 444, 90, 10);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

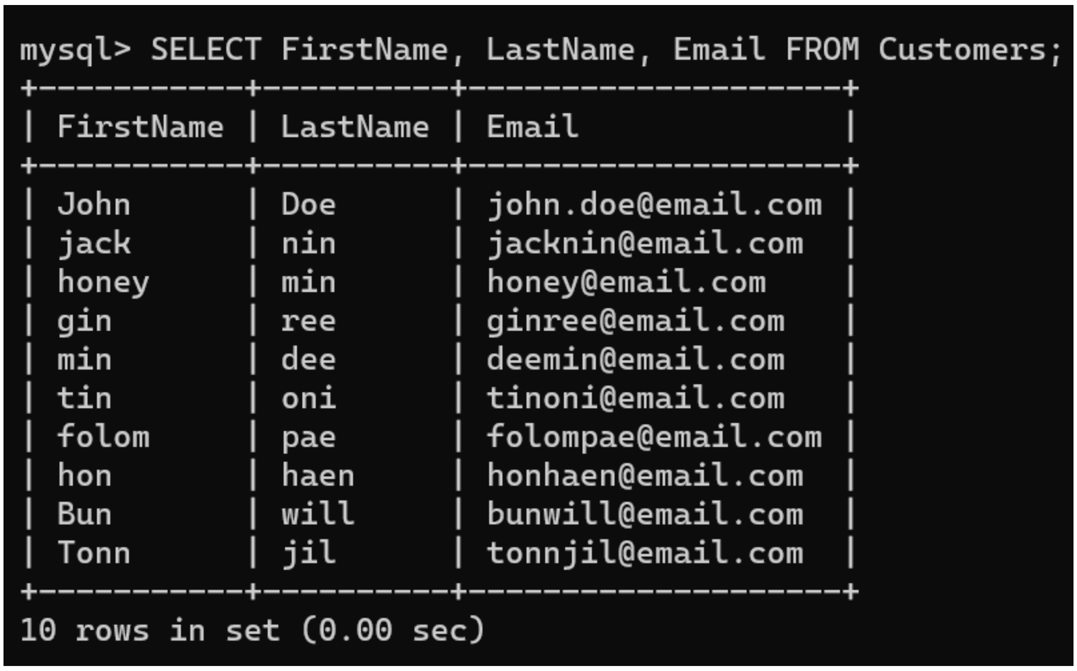
e) Inventory:



```
mysql> INSERT INTO Inventory VALUES
    -> (1001, 10, 101, '2023-01-01'),
    -> (1002, 40, 102, '2021-02-21'),
    -> (1003, 50, 291, '2022-01-05'),
    -> (1004, 50, 310, '2022-02-01'),
    -> (1005, 70, 150, '2023-07-07'),
    -> (1006, 60, 184, '2020-01-09'),
    -> (1007, 90, 420, '2023-01-11'),
    -> (1008, 40, 456, '2021-01-11'),
    -> (1009, 30, 302, '2023-01-05'),
    -> (10010, 10, 257, '2022-01-01');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

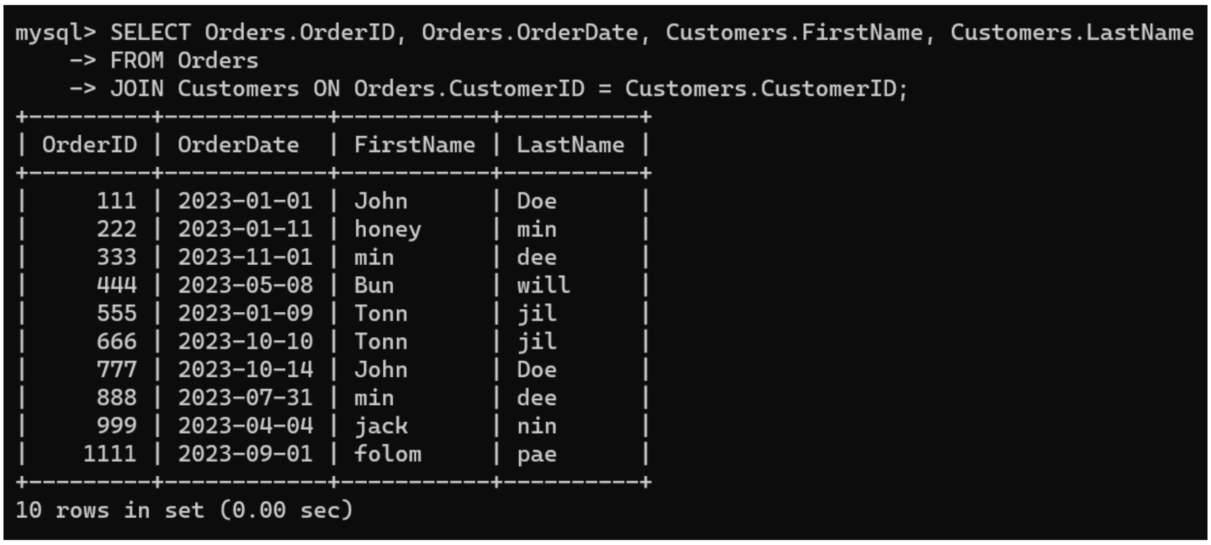
Task-2 Select, Where, Between, AND, Like:

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



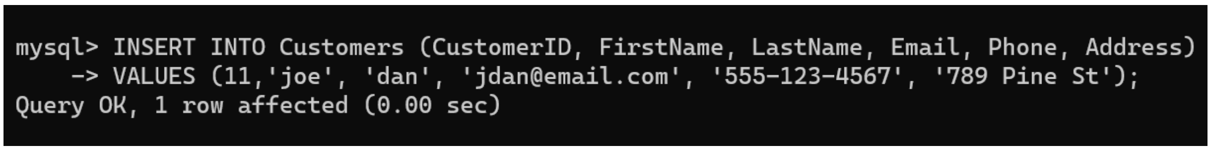
```
mysql> SELECT FirstName, LastName, Email FROM Customers;
+-----------+----------+--------------------+
| FirstName | LastName | Email              |
+-----------+----------+--------------------+
| John      | Doe      | john.doe@email.com |
| jack      | nin      | jacknin@email.com  |
| honey     | min      | honey@email.com    |
| gin       | ree      | ginree@email.com   |
| min       | dee      | deemin@email.com   |
| tin       | oni      | tinoni@email.com   |
| folom     | pae      | folompae@email.com |
| hon       | haen     | honhaen@email.com  |
| Bun       | will     | bunwill@email.com  |
| Tonn      | jil      | tonnjil@email.com  |
+-----------+----------+--------------------+
10 rows in set (0.00 sec)
```

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



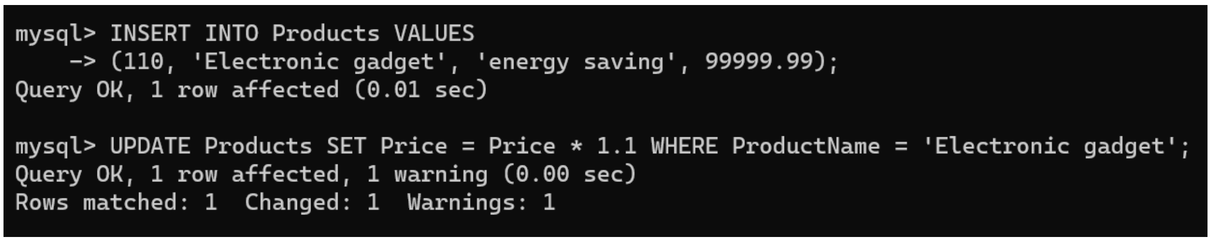
```
mysql> SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName
    -> FROM Orders
    -> JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
+---------+------------+-----------+----------+
| OrderID | OrderDate  | FirstName | LastName |
+---------+------------+-----------+----------+
|     111 | 2023-01-01 | John      | Doe      |
|     222 | 2023-01-11 | honey     | min      |
|     333 | 2023-11-01 | min       | dee      |
|     444 | 2023-05-08 | Bun       | will     |
|     555 | 2023-01-09 | Tonn      | jil      |
|     666 | 2023-10-10 | Tonn      | jil      |
|     777 | 2023-10-14 | John      | Doe      |
|     888 | 2023-07-31 | min       | dee      |
|     999 | 2023-04-04 | jack      | nin      |
|    1111 | 2023-09-01 | folom     | pae      |
+---------+------------+-----------+----------+
10 rows in set (0.00 sec)
```

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



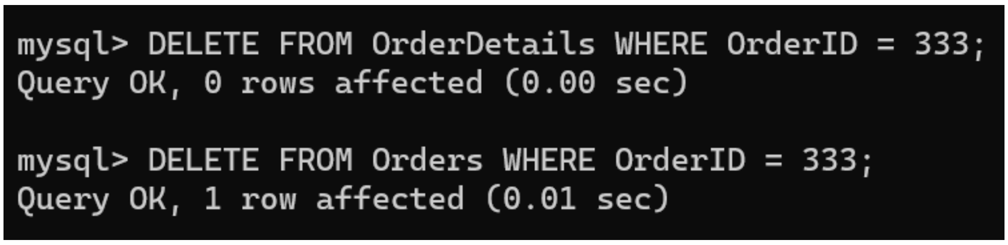
```
mysql> INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone, Address)
    -> VALUES (11,'joe', 'dan', 'jdan@email.com', '555-123-4567', '789 Pine St');
Query OK, 1 row affected (0.00 sec)
```

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



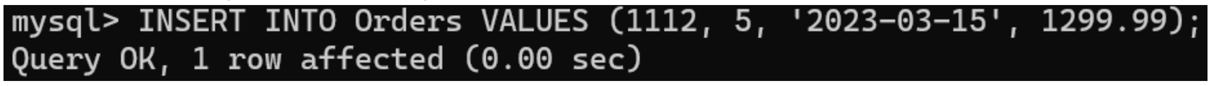
```
mysql> INSERT INTO Products VALUES
    -> (110, 'Electronic gadget', 'energy saving', 99999.99);
Query OK, 1 row affected (0.01 sec)

mysql> UPDATE Products SET Price = Price * 1.1 WHERE ProductName = 'Electronic gadget';
Query OK, 1 row affected, 1 warning (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 1
```

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.



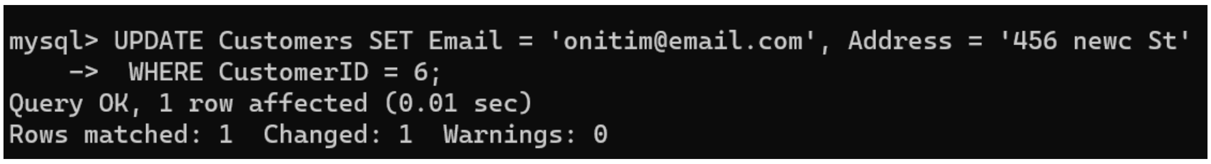
```
mysql> DELETE FROM OrderDetails WHERE OrderID = 333;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM Orders WHERE OrderID = 333;
Query OK, 1 row affected (0.01 sec)
```

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.



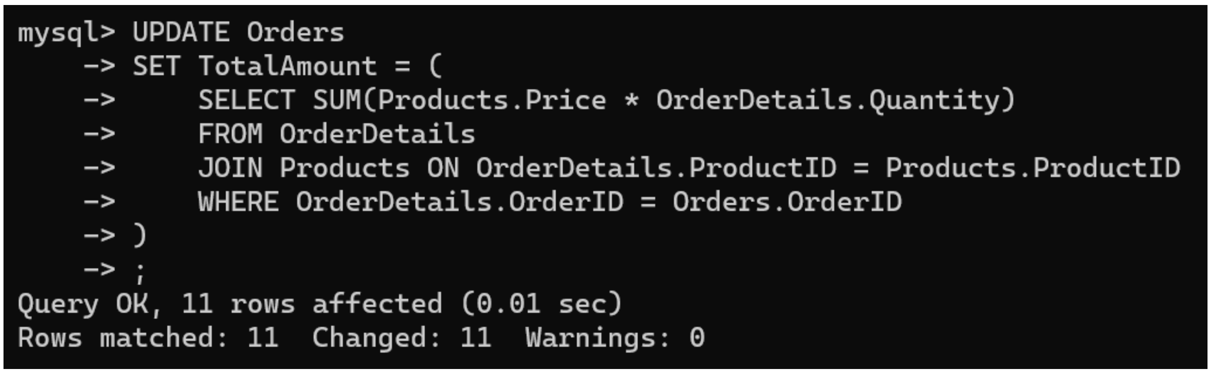
```
mysql> INSERT INTO Orders VALUES (1112, 5, '2023-03-15', 1299.99);
Query OK, 1 row affected (0.00 sec)
```

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.



```
mysql> UPDATE Customers SET Email = 'onitim@email.com', Address = '456 newc St'
    ->  WHERE CustomerID = 6;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

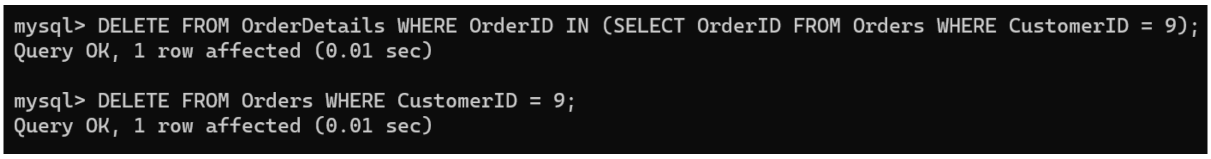
8. Write an SQL query to recalculate and update the total cost of each order in the "Orders" table based on the prices and quantities in the "OrderDetails" table.



```
mysql> UPDATE Orders
    -> SET TotalAmount = (
    ->     SELECT SUM(Products.Price * OrderDetails.Quantity)
    ->     FROM OrderDetails
    ->     JOIN Products ON OrderDetails.ProductID = Products.ProductID
    ->     WHERE OrderDetails.OrderID = Orders.OrderID
    -> )
    -> ;
Query OK, 11 rows affected (0.01 sec)
Rows matched: 11  Changed: 11  Warnings: 0
```

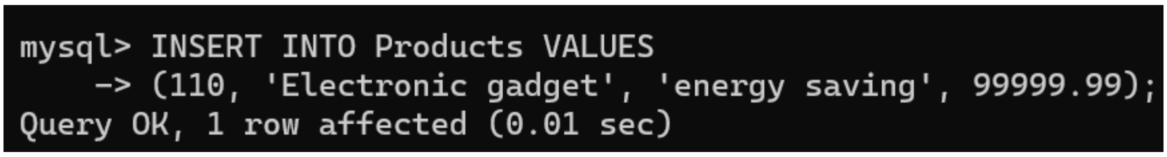
9. Write an SQL query to delete all orders and their associated order details for a specific customer from the "Orders" and "OrderDetails" tables. Allow users to input the customer ID

as a parameter.



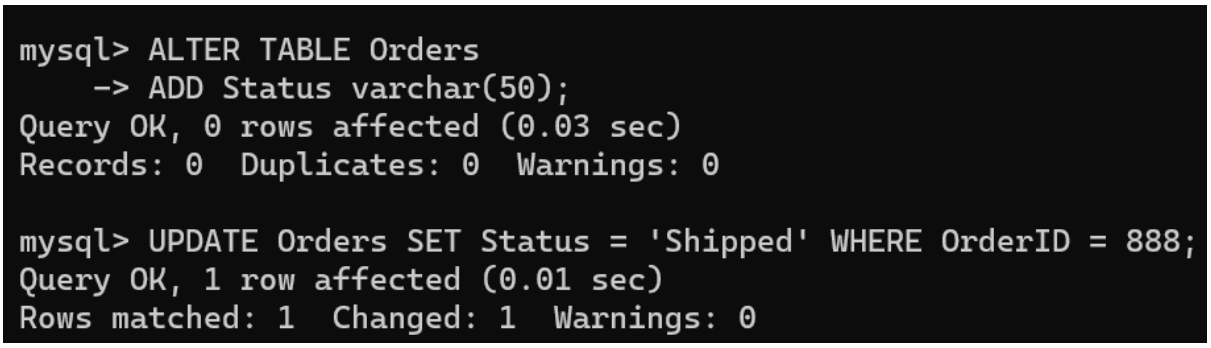
```
mysql> DELETE FROM OrderDetails WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = 9);
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM Orders WHERE CustomerID = 9;
Query OK, 1 row affected (0.01 sec)
```

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.



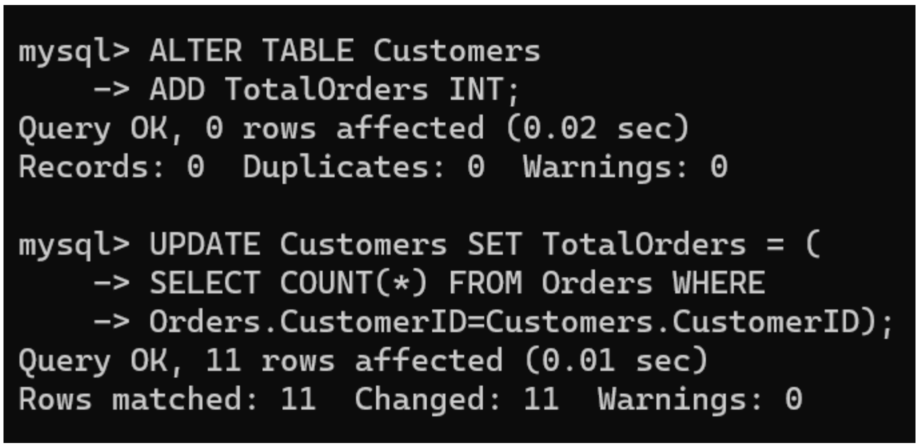
```
mysql> INSERT INTO Products VALUES
    -> (110, 'Electronic gadget', 'energy saving', 99999.99);
Query OK, 1 row affected (0.01 sec)
```

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.



```
mysql> ALTER TABLE Orders
    -> ADD Status varchar(50);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> UPDATE Orders SET Status = 'Shipped' WHERE OrderID = 888;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

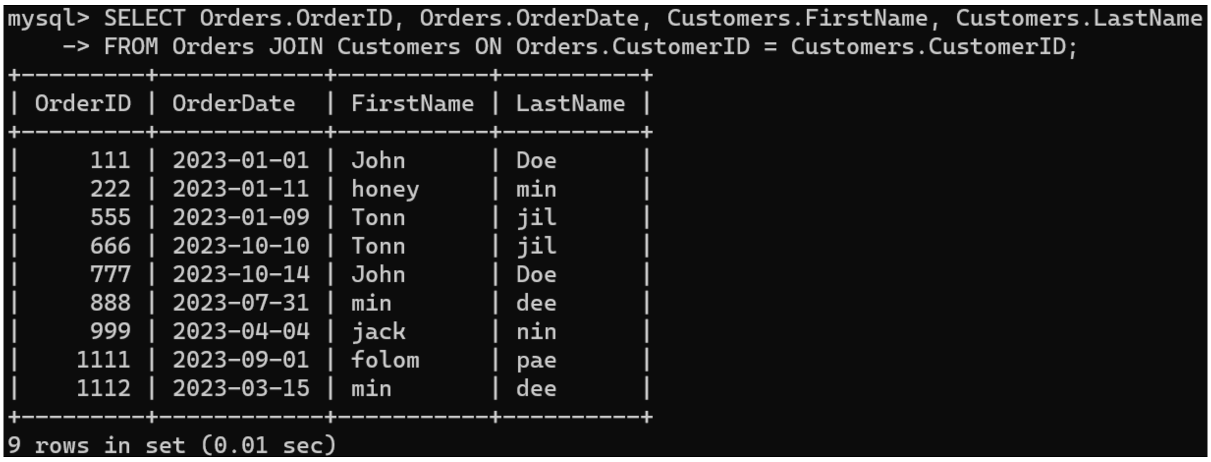
12. Write an SQL query to calculate and update the number of orders placed by each customer in the "Customers" table based on the data in the "Orders" table.



```
mysql> ALTER TABLE Customers
    -> ADD TotalOrders INT;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> UPDATE Customers SET TotalOrders = (
    -> SELECT COUNT(*) FROM Orders WHERE
    -> Orders.CustomerID=Customers.CustomerID);
Query OK, 11 rows affected (0.01 sec)
Rows matched: 11  Changed: 11  Warnings: 0
```

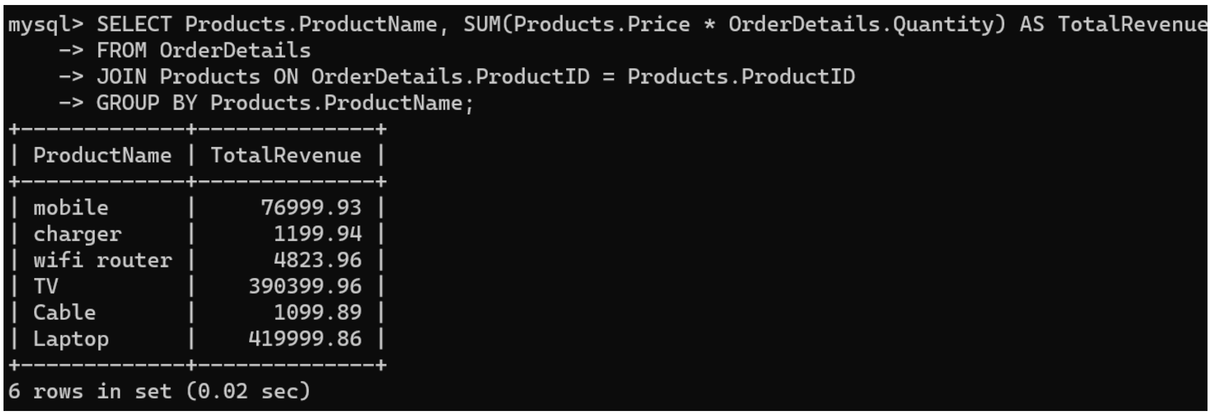
<u>Task-3 Aggregate functions, Having, Order By, Group By and Joins:</u>

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



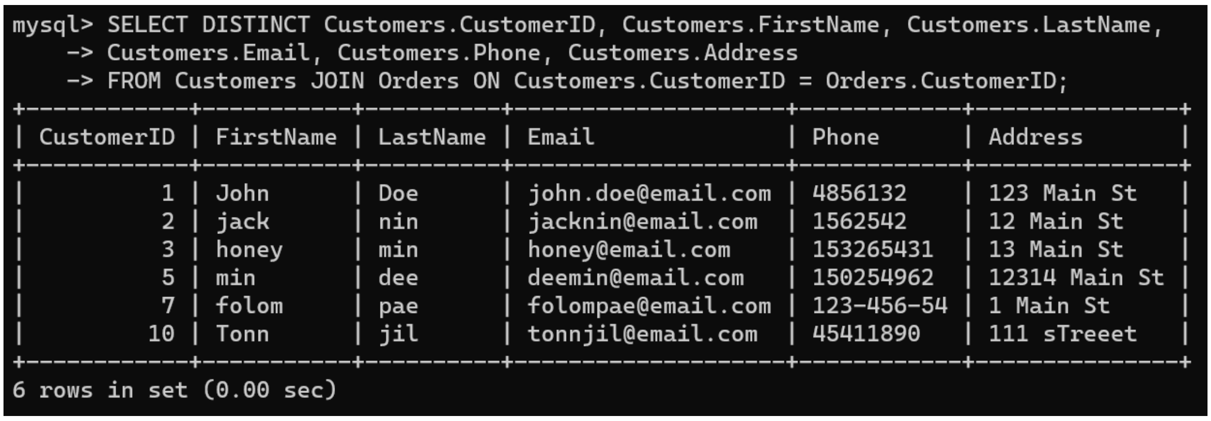
```
mysql> SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName
    -> FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
+---------+------------+-----------+----------+
| OrderID | OrderDate  | FirstName | LastName |
+---------+------------+-----------+----------+
|     111 | 2023-01-01 | John      | Doe      |
|     222 | 2023-01-11 | honey     | min      |
|     555 | 2023-01-09 | Tonn      | jil      |
|     666 | 2023-10-10 | Tonn      | jil      |
|     777 | 2023-10-14 | John      | Doe      |
|     888 | 2023-07-31 | min       | dee      |
|     999 | 2023-04-04 | jack      | nin      |
|    1111 | 2023-09-01 | folom     | pae      |
|    1112 | 2023-03-15 | min       | dee      |
+---------+------------+-----------+----------+
9 rows in set (0.01 sec)
```

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



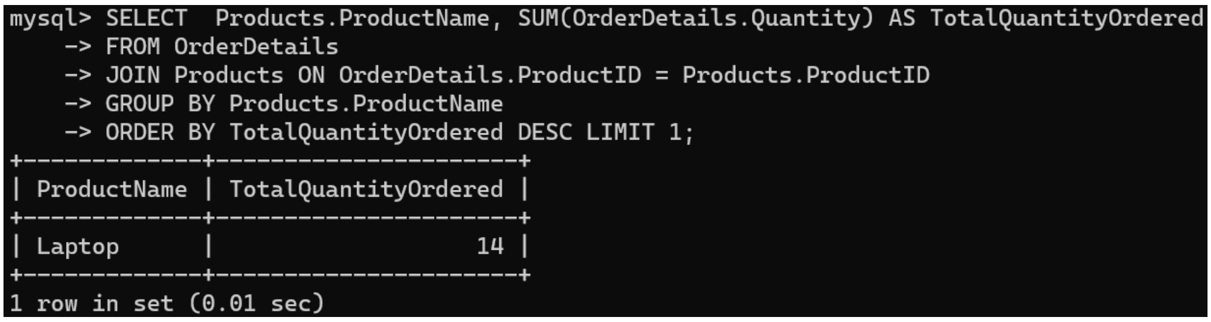
```
mysql> SELECT Products.ProductName, SUM(Products.Price * OrderDetails.Quantity) AS TotalRevenue
    -> FROM OrderDetails
    -> JOIN Products ON OrderDetails.ProductID = Products.ProductID
    -> GROUP BY Products.ProductName;
+-------------+--------------+
| ProductName | TotalRevenue |
+-------------+--------------+
| mobile      |     76999.93 |
| charger     |      1199.94 |
| wifi router |      4823.96 |
| TV          |    390399.96 |
| Cable       |      1099.89 |
| Laptop      |    419999.86 |
+-------------+--------------+
6 rows in set (0.02 sec)
```

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



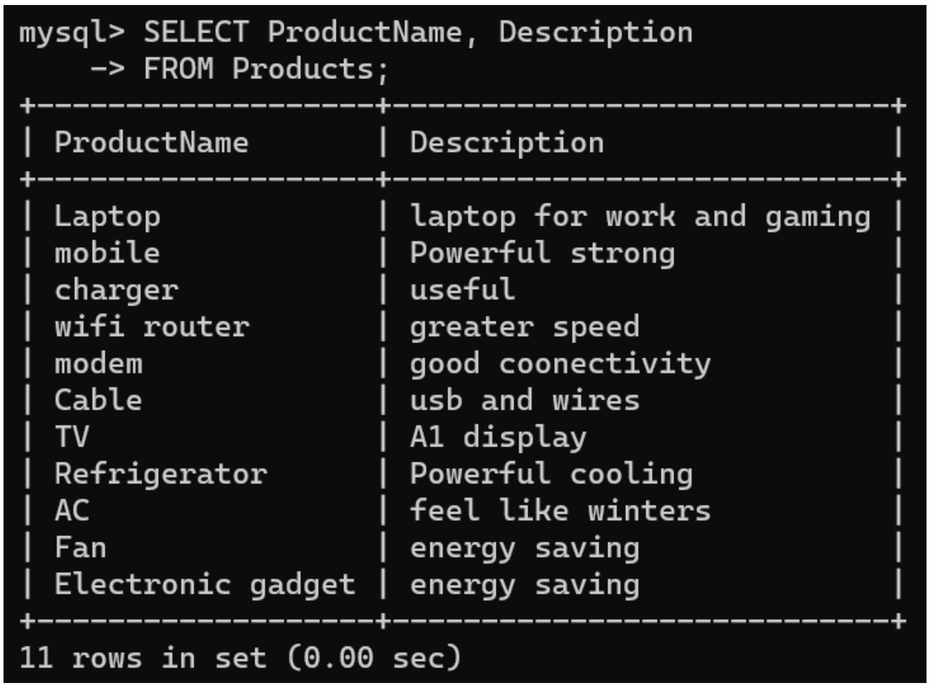
```
mysql> SELECT DISTINCT Customers.CustomerID, Customers.FirstName, Customers.LastName,
    -> Customers.Email, Customers.Phone, Customers.Address
    -> FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
+------------+-----------+----------+--------------------+------------+----------------+
| CustomerID | FirstName | LastName | Email              | Phone      | Address        |
+------------+-----------+----------+--------------------+------------+----------------+
|          1 | John      | Doe      | john.doe@email.com | 4856132    | 123 Main St    |
|          2 | jack      | nin      | jacknin@email.com  | 1562542    | 12 Main St     |
|          3 | honey     | min      | honey@email.com    | 153265431  | 13 Main St     |
|          5 | min       | dee      | deemin@email.com   | 150254962  | 12314 Main St  |
|          7 | folom     | pae      | folompae@email.com | 123-456-54 | 1 Main St      |
|         10 | Tonn      | jil      | tonnjil@email.com  | 45411890   | 111 sTreeet    |
+------------+-----------+----------+--------------------+------------+----------------+
6 rows in set (0.00 sec)
```

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.



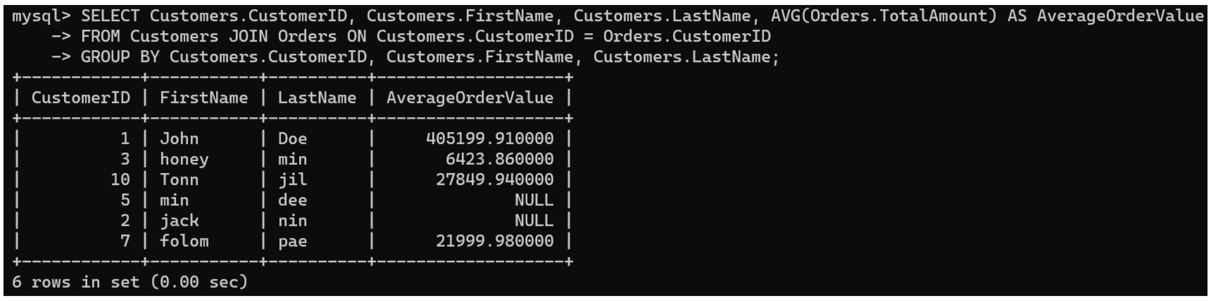
```
mysql> SELECT  Products.ProductName, SUM(OrderDetails.Quantity) AS TotalQuantityOrdered
    -> FROM OrderDetails
    -> JOIN Products ON OrderDetails.ProductID = Products.ProductID
    -> GROUP BY Products.ProductName
    -> ORDER BY TotalQuantityOrdered DESC LIMIT 1;
+-------------+----------------------+
| ProductName | TotalQuantityOrdered |
+-------------+----------------------+
| Laptop      |                   14 |
+-------------+----------------------+
1 row in set (0.01 sec)
```

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



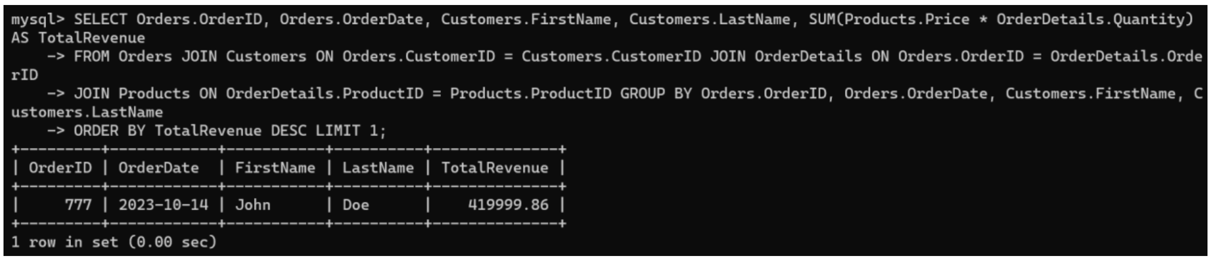
```
mysql> SELECT ProductName, Description
    -> FROM Products;
+-------------------+---------------------------+
| ProductName       | Description               |
+-------------------+---------------------------+
| Laptop            | laptop for work and gaming |
| mobile            | Powerful strong           |
| charger           | useful                    |
| wifi router       | greater speed             |
| modem             | good coonectivity         |
| Cable             | usb and wires             |
| TV                | A1 display                |
| Refrigerator      | Powerful cooling          |
| AC                | feel like winters         |
| Fan               | energy saving             |
| Electronic gadget | energy saving             |
+-------------------+---------------------------+
11 rows in set (0.00 sec)
```

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



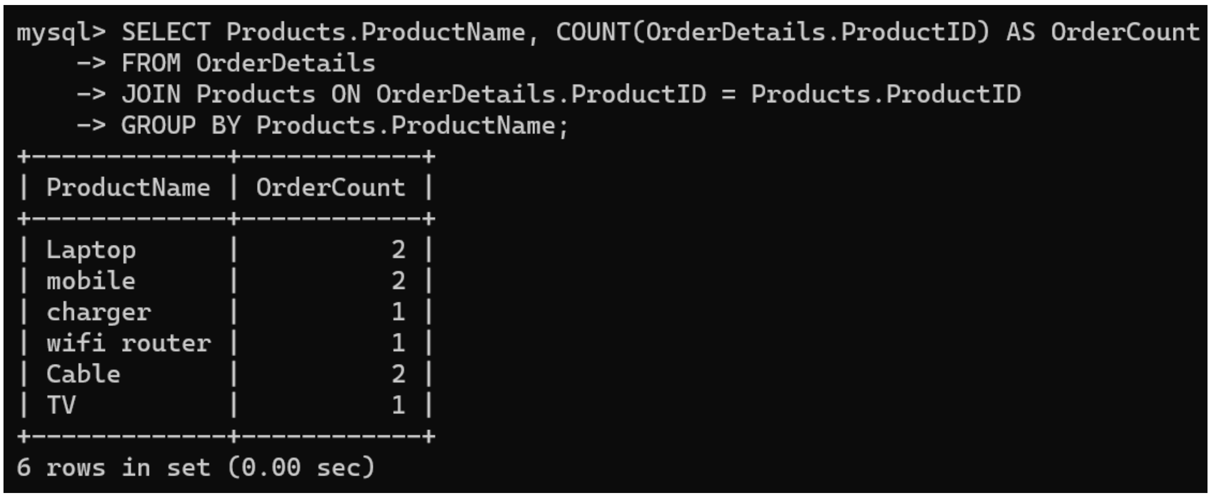
```
mysql> SELECT Customers.CustomerID, Customers.FirstName, Customers.LastName, AVG(Orders.TotalAmount) AS AverageOrderValue
    -> FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    -> GROUP BY Customers.CustomerID, Customers.FirstName, Customers.LastName;
+------------+-----------+----------+-------------------+
| CustomerID | FirstName | LastName | AverageOrderValue |
+------------+-----------+----------+-------------------+
|          1 | John      | Doe      |     405199.910000 |
|          3 | honey     | min      |       6423.860000 |
|         10 | Tonn      | jil      |      27849.940000 |
|          5 | min       | dee      |              NULL |
|          2 | jack      | nin      |              NULL |
|          7 | folom     | pae      |      21999.980000 |
+------------+-----------+----------+-------------------+
6 rows in set (0.00 sec)
```

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



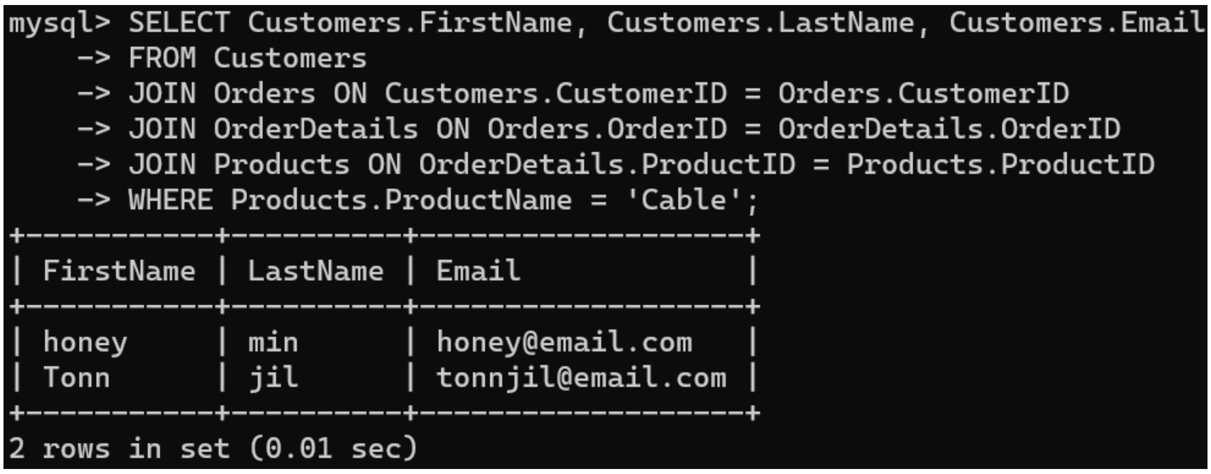
```
mysql> SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName, SUM(Products.Price * OrderDetails.Quantity)
AS TotalRevenue
    -> FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID JOIN OrderDetails ON Orders.OrderID = OrderDetails.Orde
rID
    -> JOIN Products ON OrderDetails.ProductID = Products.ProductID GROUP BY Orders.OrderID, Orders.OrderDate, Customers.FirstName, C
ustomers.LastName
    -> ORDER BY TotalRevenue DESC LIMIT 1;
+---------+------------+-----------+----------+--------------+
| OrderID | OrderDate  | FirstName | LastName | TotalRevenue |
+---------+------------+-----------+----------+--------------+
|     777 | 2023-10-14 | John      | Doe      |    419999.86 |
+---------+------------+-----------+----------+--------------+
1 row in set (0.00 sec)
```

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



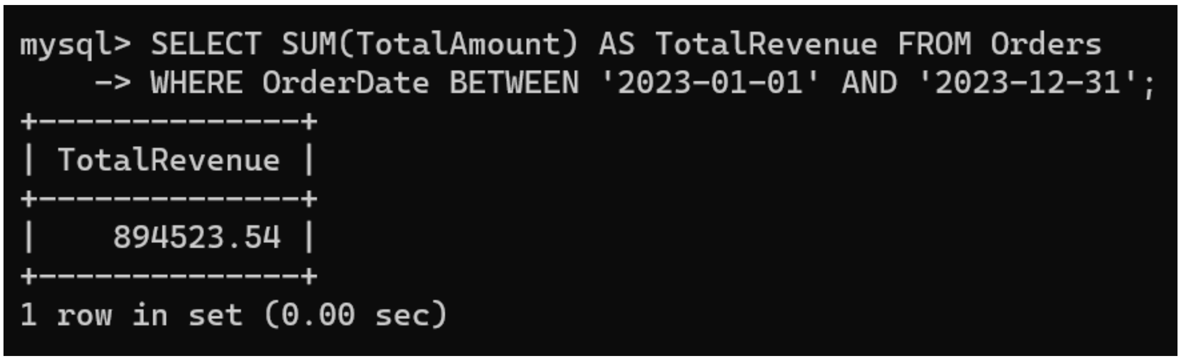
```
mysql> SELECT Products.ProductName, COUNT(OrderDetails.ProductID) AS OrderCount
    -> FROM OrderDetails
    -> JOIN Products ON OrderDetails.ProductID = Products.ProductID
    -> GROUP BY Products.ProductName;
+-------------+------------+
| ProductName | OrderCount |
+-------------+------------+
| Laptop      |          2 |
| mobile      |          2 |
| charger     |          1 |
| wifi router |          1 |
| Cable       |          2 |
| TV          |          1 |
+-------------+------------+
6 rows in set (0.00 sec)
```

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.



```
mysql> SELECT Customers.FirstName, Customers.LastName, Customers.Email
    -> FROM Customers
    -> JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    -> JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
    -> JOIN Products ON OrderDetails.ProductID = Products.ProductID
    -> WHERE Products.ProductName = 'Cable';
+-----------+----------+-------------------+
| FirstName | LastName | Email             |
+-----------+----------+-------------------+
| honey     | min      | honey@email.com   |
| Tonn      | jil      | tonnjil@email.com |
+-----------+----------+-------------------+
2 rows in set (0.01 sec)
```

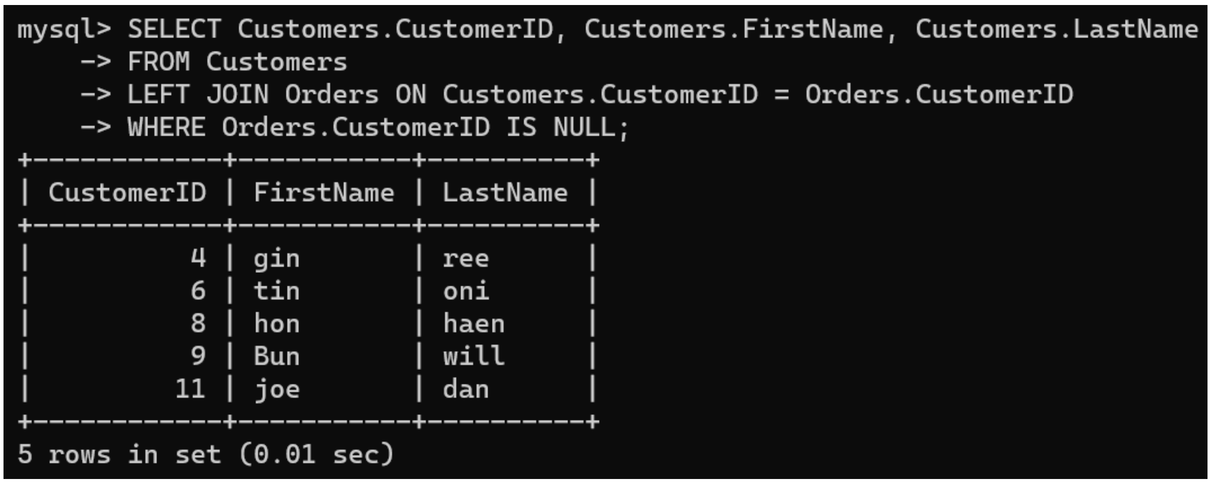
10. Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period. Allow users to input the start and end dates as parameters.



```
mysql> SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders
    -> WHERE OrderDate BETWEEN '2023-01-01' AND '2023-12-31';
+--------------+
| TotalRevenue |
+--------------+
|    894523.54 |
+--------------+
1 row in set (0.00 sec)
```

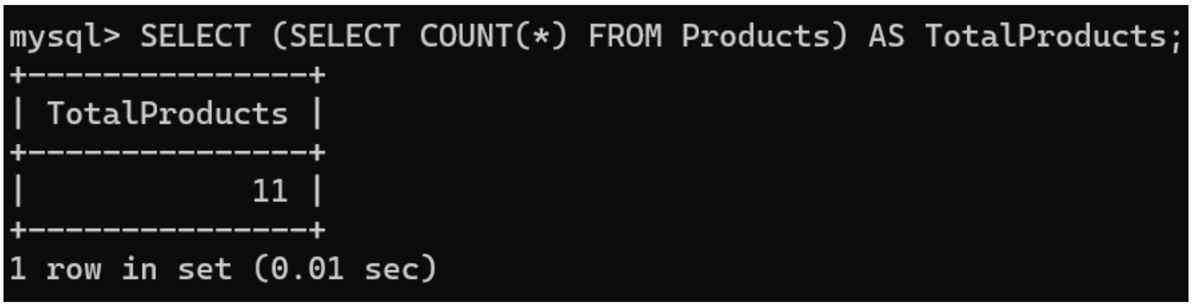
Task-4 Subquery and its type:

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



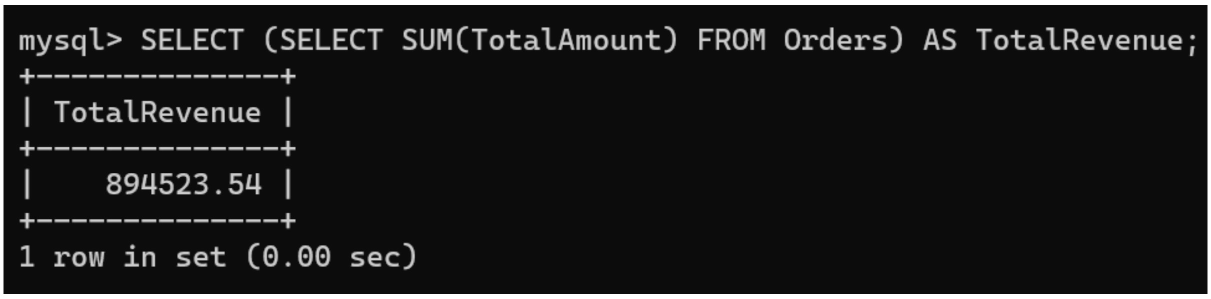
```
mysql> SELECT Customers.CustomerID, Customers.FirstName, Customers.LastName
    -> FROM Customers
    -> LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    -> WHERE Orders.CustomerID IS NULL;
+------------+-----------+----------+
| CustomerID | FirstName | LastName |
+------------+-----------+----------+
|          4 | gin       | ree      |
|          6 | tin       | oni      |
|          8 | hon       | haen     |
|          9 | Bun       | will     |
|         11 | joe       | dan      |
+------------+-----------+----------+
5 rows in set (0.01 sec)
```

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



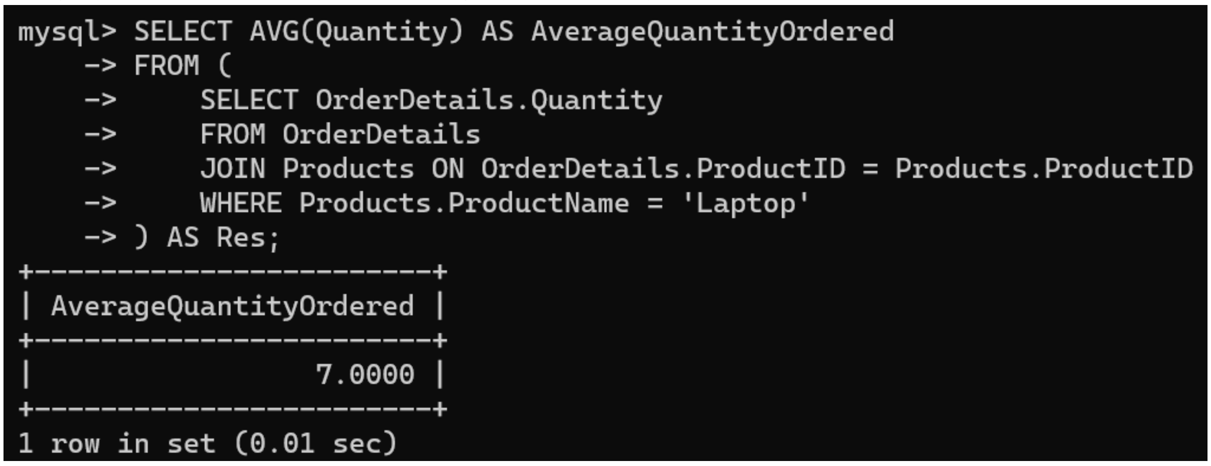
```
mysql> SELECT (SELECT COUNT(*) FROM Products) AS TotalProducts;
+---------------+
| TotalProducts |
+---------------+
|            11 |
+---------------+
1 row in set (0.01 sec)
```

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



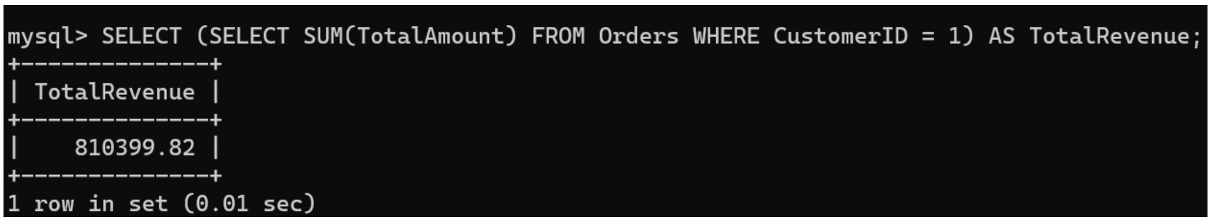
```
mysql> SELECT (SELECT SUM(TotalAmount) FROM Orders) AS TotalRevenue;
+--------------+
| TotalRevenue |
+--------------+
|    894523.54 |
+--------------+
1 row in set (0.00 sec)
```

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.



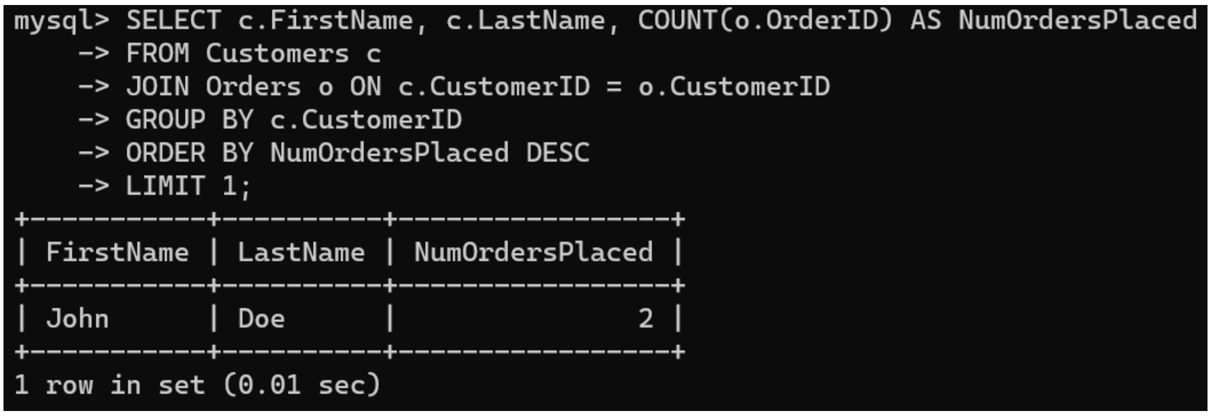
```
mysql> SELECT AVG(Quantity) AS AverageQuantityOrdered
    -> FROM (
    ->     SELECT OrderDetails.Quantity
    ->     FROM OrderDetails
    ->     JOIN Products ON OrderDetails.ProductID = Products.ProductID
    ->     WHERE Products.ProductName = 'Laptop'
    -> ) AS Res;
+------------------------+
| AverageQuantityOrdered |
+------------------------+
|                 7.0000 |
+------------------------+
1 row in set (0.01 sec)
```

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.



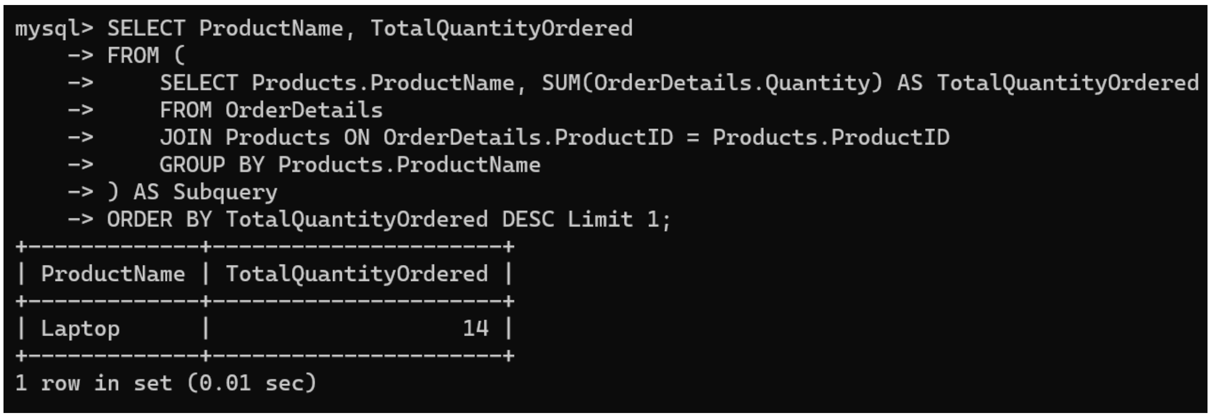
```
mysql> SELECT (SELECT SUM(TotalAmount) FROM Orders WHERE CustomerID = 1) AS TotalRevenue;
+--------------+
| TotalRevenue |
+--------------+
|    810399.82 |
+--------------+
1 row in set (0.01 sec)
```

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.



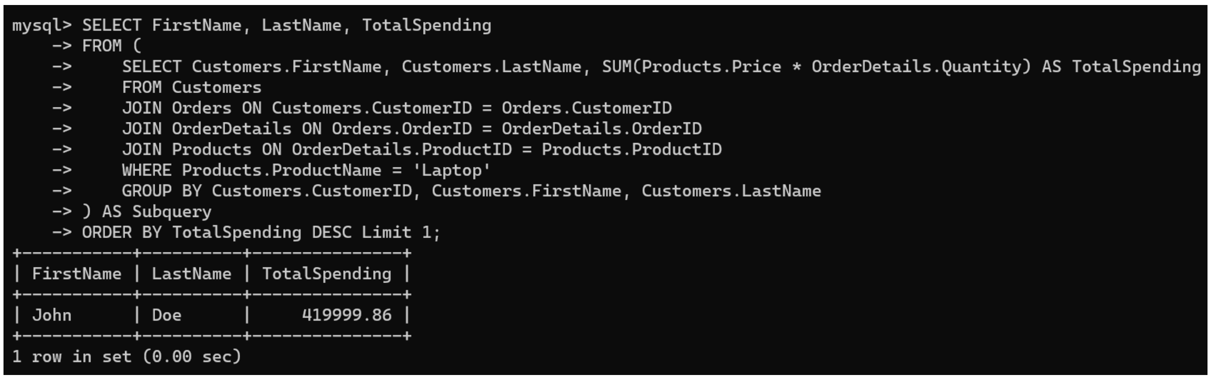
```
mysql> SELECT c.FirstName, c.LastName, COUNT(o.OrderID) AS NumOrdersPlaced
    -> FROM Customers c
    -> JOIN Orders o ON c.CustomerID = o.CustomerID
    -> GROUP BY c.CustomerID
    -> ORDER BY NumOrdersPlaced DESC
    -> LIMIT 1;
+-----------+----------+-----------------+
| FirstName | LastName | NumOrdersPlaced |
+-----------+----------+-----------------+
| John      | Doe      |               2 |
+-----------+----------+-----------------+
1 row in set (0.01 sec)
```

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.



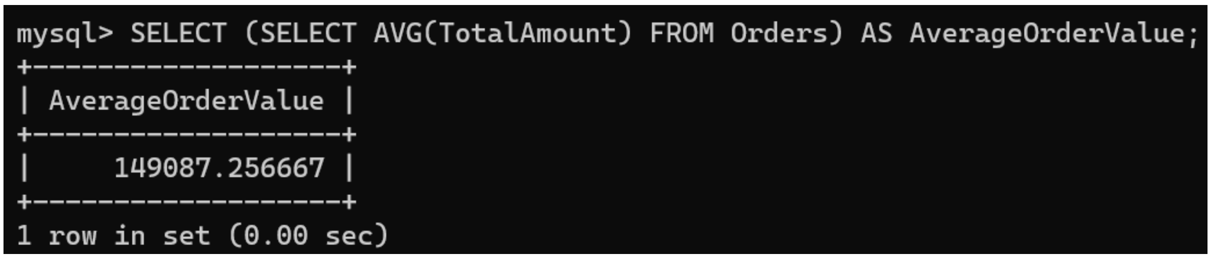
```
mysql> SELECT ProductName, TotalQuantityOrdered
    -> FROM (
    ->     SELECT Products.ProductName, SUM(OrderDetails.Quantity) AS TotalQuantityOrdered
    ->     FROM OrderDetails
    ->     JOIN Products ON OrderDetails.ProductID = Products.ProductID
    ->     GROUP BY Products.ProductName
    -> ) AS Subquery
    -> ORDER BY TotalQuantityOrdered DESC Limit 1;
+-------------+----------------------+
| ProductName | TotalQuantityOrdered |
+-------------+----------------------+
| Laptop      |                   14 |
+-------------+----------------------+
1 row in set (0.01 sec)
```

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.



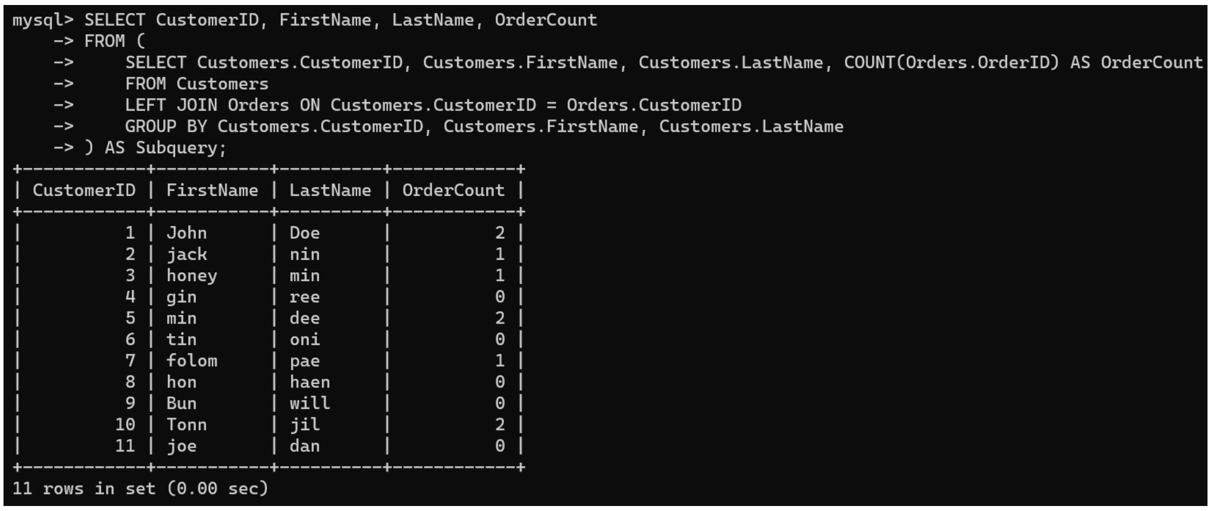
```
mysql> SELECT FirstName, LastName, TotalSpending
    -> FROM (
    ->     SELECT Customers.FirstName, Customers.LastName, SUM(Products.Price * OrderDetails.Quantity) AS TotalSpending
    ->     FROM Customers
    ->     JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    ->     JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
    ->     JOIN Products ON OrderDetails.ProductID = Products.ProductID
    ->     WHERE Products.ProductName = 'Laptop'
    ->     GROUP BY Customers.CustomerID, Customers.FirstName, Customers.LastName
    -> ) AS Subquery
    -> ORDER BY TotalSpending DESC Limit 1;
+-----------+----------+---------------+
| FirstName | LastName | TotalSpending |
+-----------+----------+---------------+
| John      | Doe      |     419999.86 |
+-----------+----------+---------------+
1 row in set (0.00 sec)
```

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



```
mysql> SELECT (SELECT AVG(TotalAmount) FROM Orders) AS AverageOrderValue;
+-------------------+
| AverageOrderValue |
+-------------------+
|     149087.256667 |
+-------------------+
1 row in set (0.00 sec)
```

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.



```
mysql> SELECT CustomerID, FirstName, LastName, OrderCount
    -> FROM (
    ->     SELECT Customers.CustomerID, Customers.FirstName, Customers.LastName, COUNT(Orders.OrderID) AS OrderCount
    ->     FROM Customers
    ->     LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    ->     GROUP BY Customers.CustomerID, Customers.FirstName, Customers.LastName
    -> ) AS Subquery;
+------------+-----------+----------+------------+
| CustomerID | FirstName | LastName | OrderCount |
+------------+-----------+----------+------------+
|          1 | John      | Doe      |          2 |
|          2 | jack      | nin      |          1 |
|          3 | honey     | min      |          1 |
|          4 | gin       | ree      |          0 |
|          5 | min       | dee      |          2 |
|          6 | tin       | oni      |          0 |
|          7 | folom     | pae      |          1 |
|          8 | hon       | haen     |          0 |
|          9 | Bun       | will     |          0 |
|         10 | Tonn      | jil      |          2 |
|         11 | joe       | dan      |          0 |
+------------+-----------+----------+------------+
11 rows in set (0.00 sec)
```