ASSIGNMENT -1

Submitted by : SANKAR ROY

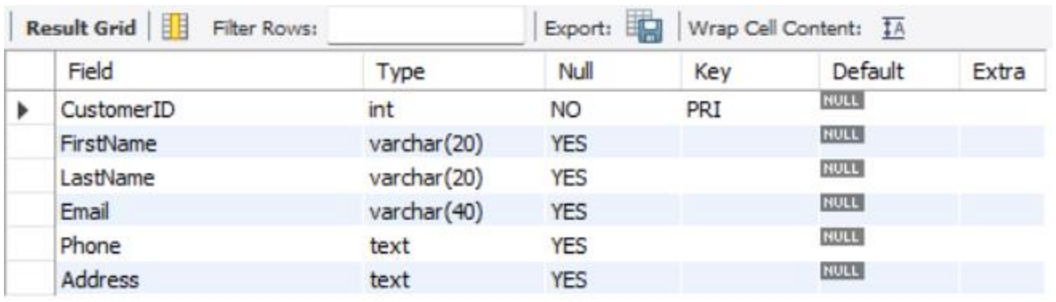
Task:1. Database Design:

1. Create the database named "TechShop"

```
create database TechShop;
use TechShop;
```

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

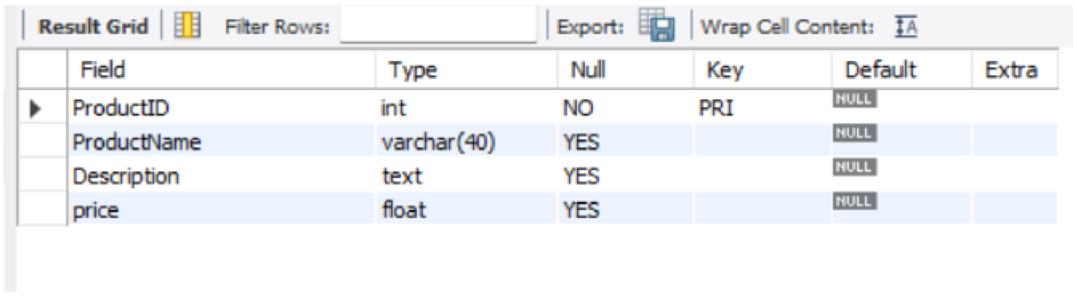
```
create table Customers(CustomerID int primary key ,
 FirstName varchar(20) ,
 LastName varchar(20) ,
 Email varchar(40) , Phone text
 , Address text);
desc customers;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

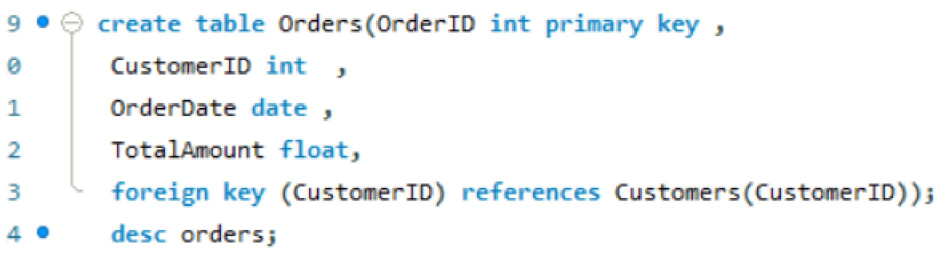
Field	Type	Null	Key	Default	Extra
CustomerID	int	NO	PRI	NULL	
FirstName	varchar(20)	YES		NULL	
LastName	varchar(20)	YES		NULL	
Email	varchar(40)	YES		NULL	
Phone	text	YES		NULL	
Address	text	YES		NULL	

```
create table Products(ProductID int primary key,
ProductName varchar(40) ,
Description text ,
price float);
desc products;
```

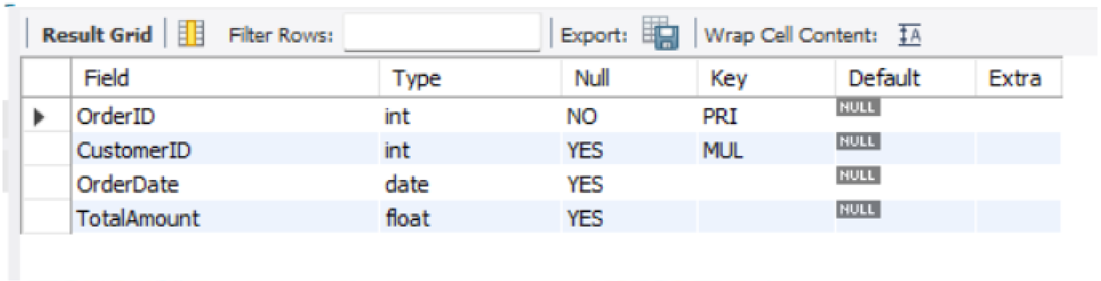


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

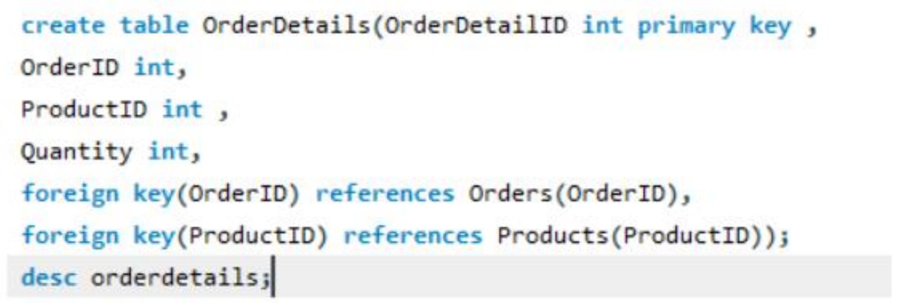
Field	Type	Null	Key	Default	Extra
ProductID	int	NO	PRI	NULL	
ProductName	varchar(40)	YES		NULL	
Description	text	YES		NULL	
price	float	YES		NULL	



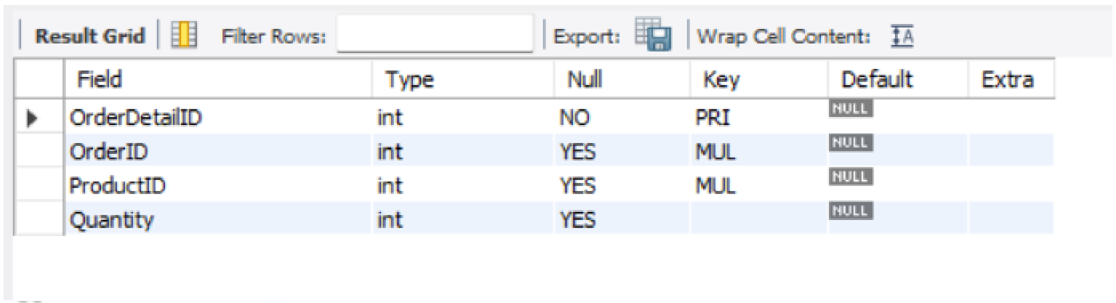
```sql
create table Orders(OrderID int primary key ,
 CustomerID int  ,
 OrderDate date ,
 TotalAmount float,
 foreign key (CustomerID) references Customers(CustomerID));
 desc orders;
```



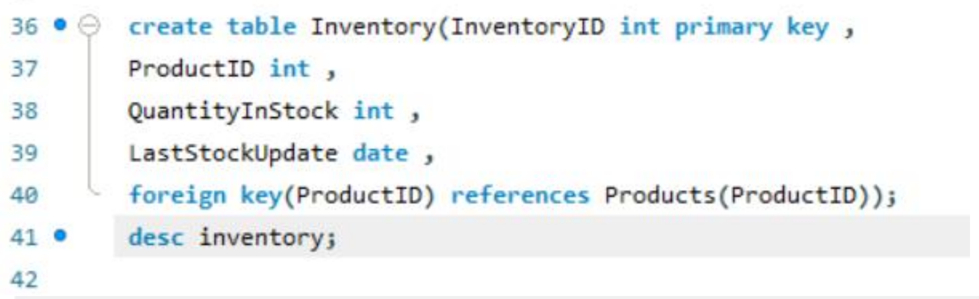
Field	Type	Null	Key	Default	Extra
OrderID	int	NO	PRI	NULL	
CustomerID	int	YES	MUL	NULL	
OrderDate	date	YES		NULL	
TotalAmount	float	YES		NULL	



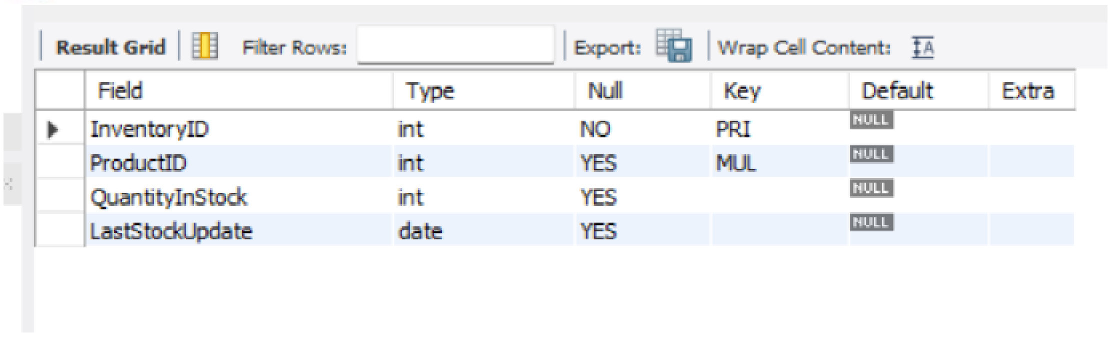
```sql
create table OrderDetails(OrderDetailID int primary key ,
OrderID int,
ProductID int ,
Quantity int,
foreign key(OrderID) references Orders(OrderID),
foreign key(ProductID) references Products(ProductID));
desc orderdetails;
```



Field	Type	Null	Key	Default	Extra
OrderDetailID	int	NO	PRI	NULL	
OrderID	int	YES	MUL	NULL	
ProductID	int	YES	MUL	NULL	
Quantity	int	YES		NULL	

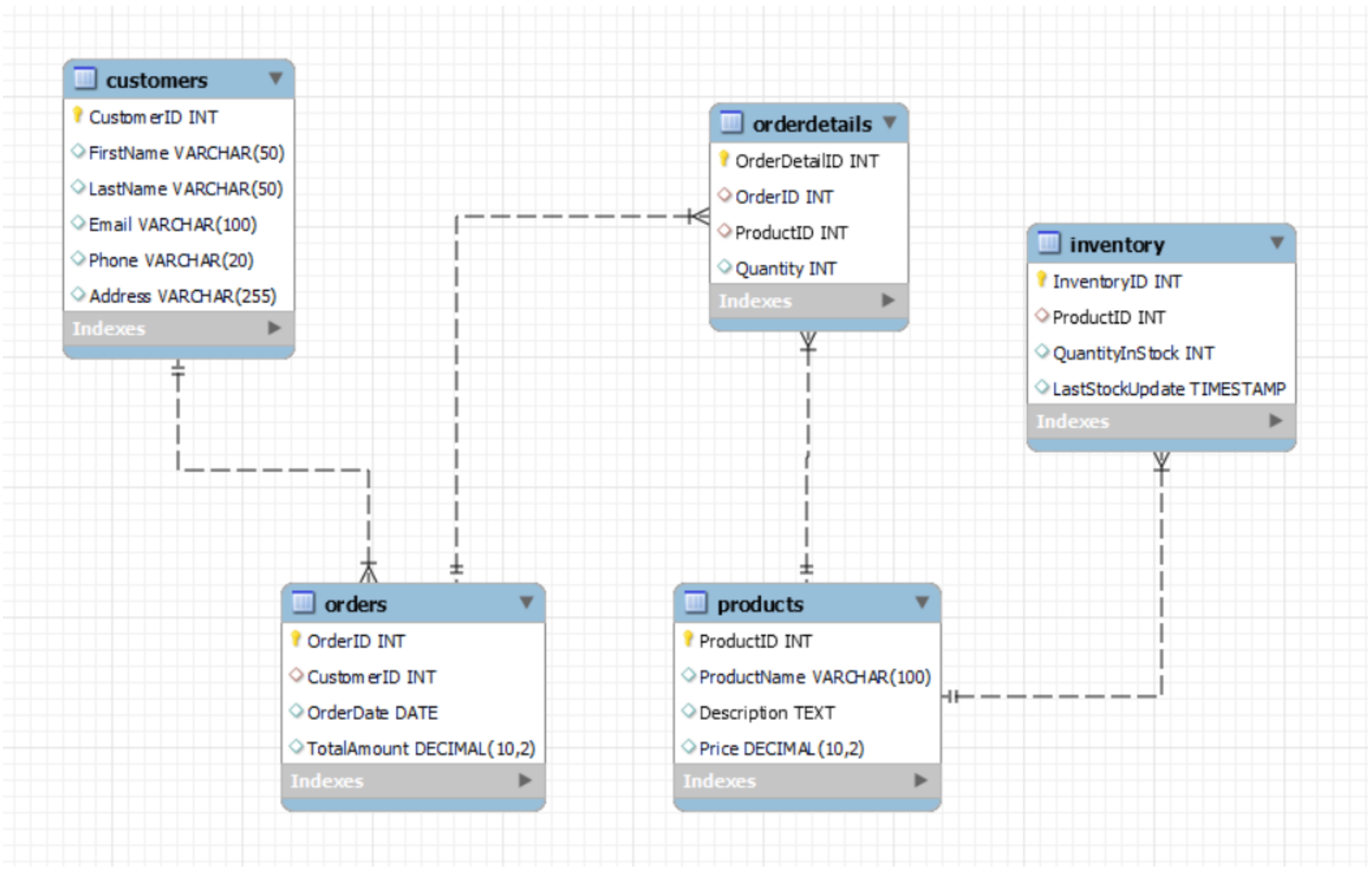


```sql
create table Inventory(InventoryID int primary key ,
ProductID int ,
QuantityInStock int ,
LastStockUpdate date ,
foreign key(ProductID) references Products(ProductID));
desc inventory;
```

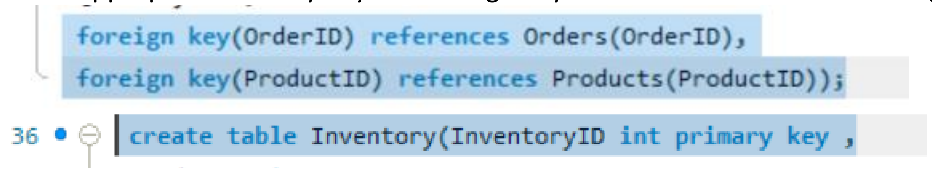


Field	Type	Null	Key	Default	Extra
InventoryID	int	NO	PRI	NULL	
ProductID	int	YES	MUL	NULL	
QuantityInStock	int	YES		NULL	
LastStockUpdate	date	YES		NULL	

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



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

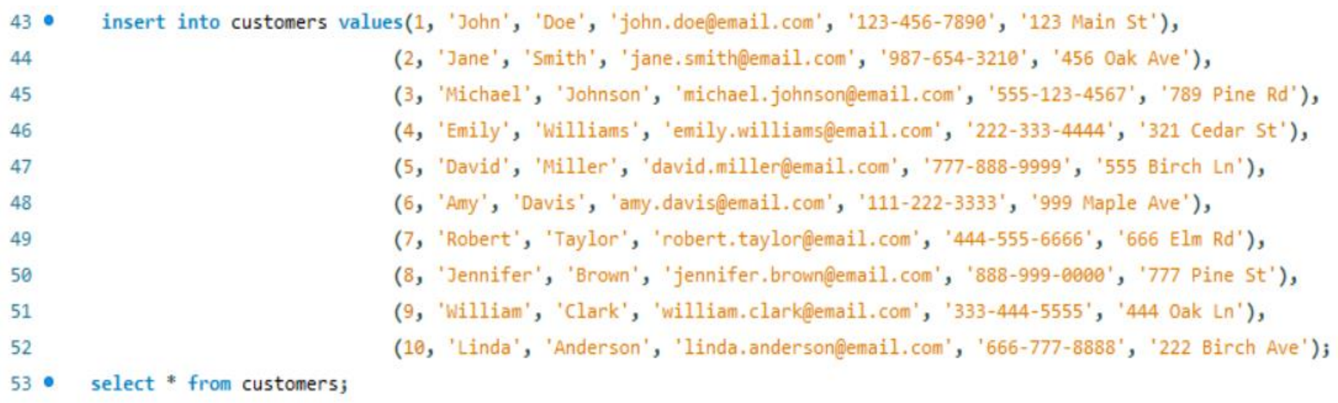


```
foreign key(OrderID) references Orders(OrderID),
foreign key(ProductID) references Products(ProductID));
```

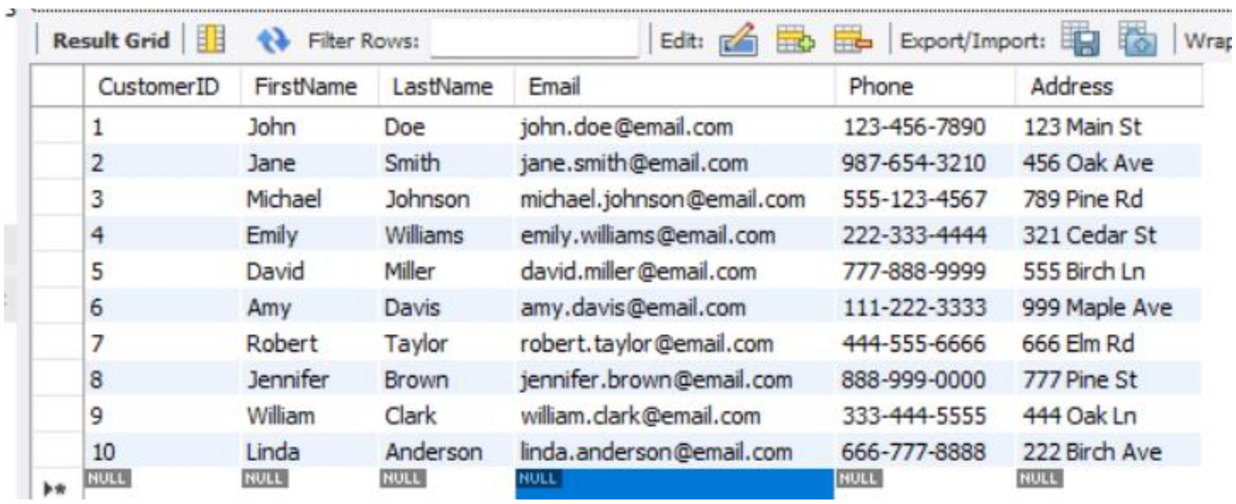
```
36 create table Inventory(InventoryID int primary key ,
```

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

   a. Customers

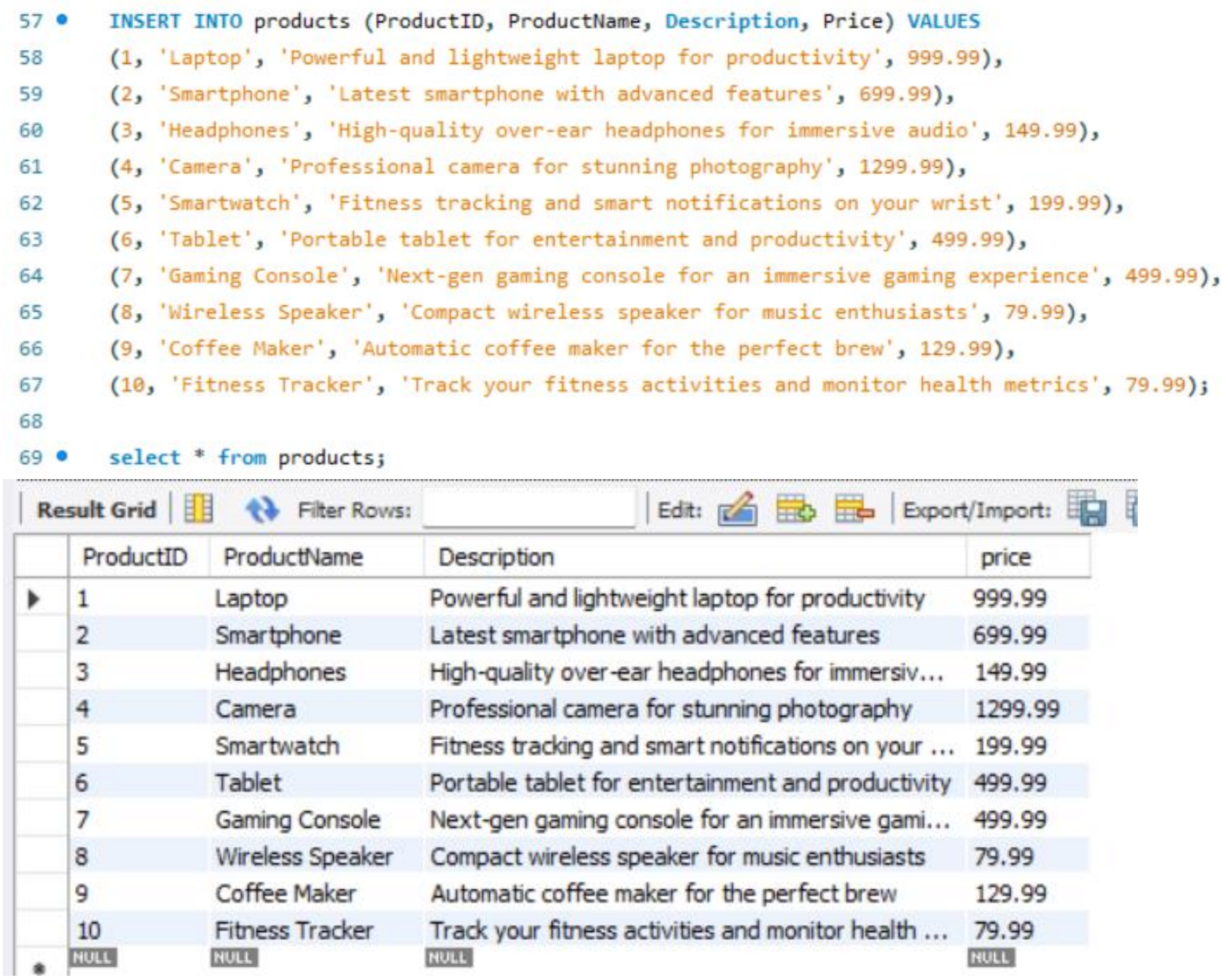


```
insert into customers values(1, 'John', 'Doe', 'john.doe@email.com', '123-456-7890', '123 Main St'),
                             (2, 'Jane', 'Smith', 'jane.smith@email.com', '987-654-3210', '456 Oak Ave'),
                             (3, 'Michael', 'Johnson', 'michael.johnson@email.com', '555-123-4567', '789 Pine Rd'),
                             (4, 'Emily', 'Williams', 'emily.williams@email.com', '222-333-4444', '321 Cedar St'),
                             (5, 'David', 'Miller', 'david.miller@email.com', '777-888-9999', '555 Birch Ln'),
                             (6, 'Amy', 'Davis', 'amy.davis@email.com', '111-222-3333', '999 Maple Ave'),
                             (7, 'Robert', 'Taylor', 'robert.taylor@email.com', '444-555-6666', '666 Elm Rd'),
                             (8, 'Jennifer', 'Brown', 'jennifer.brown@email.com', '888-999-0000', '777 Pine St'),
                             (9, 'William', 'Clark', 'william.clark@email.com', '333-444-5555', '444 Oak Ln'),
                             (10, 'Linda', 'Anderson', 'linda.anderson@email.com', '666-777-8888', '222 Birch Ave');
select * from customers;
```



	CustomerID	FirstName	LastName	Email	Phone	Address
	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 Ave
	3	Michael	Johnson	michael.johnson@email.com	555-123-4567	789 Pine Rd
	4	Emily	Williams	emily.williams@email.com	222-333-4444	321 Cedar St
	5	David	Miller	david.miller@email.com	777-888-9999	555 Birch Ln
	6	Amy	Davis	amy.davis@email.com	111-222-3333	999 Maple Ave
	7	Robert	Taylor	robert.taylor@email.com	444-555-6666	666 Elm Rd
	8	Jennifer	Brown	jennifer.brown@email.com	888-999-0000	777 Pine St
	9	William	Clark	william.clark@email.com	333-444-5555	444 Oak Ln
	10	Linda	Anderson	linda.anderson@email.com	666-777-8888	222 Birch Ave
	NULL	NULL	NULL	NULL	NULL	NULL

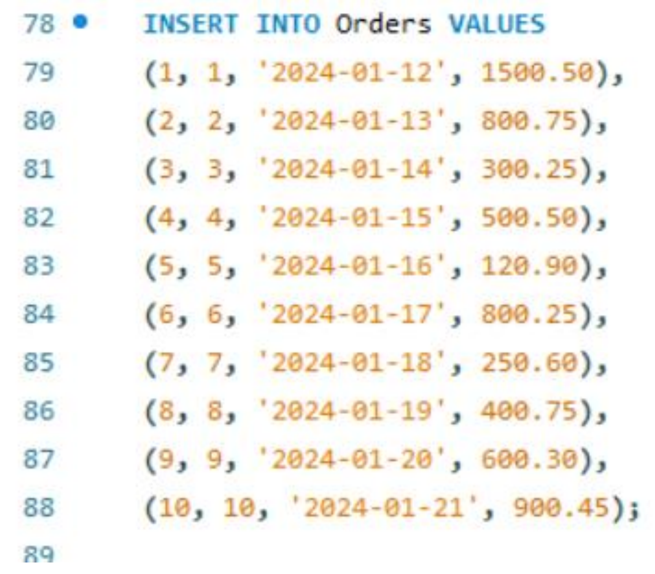
b. Products



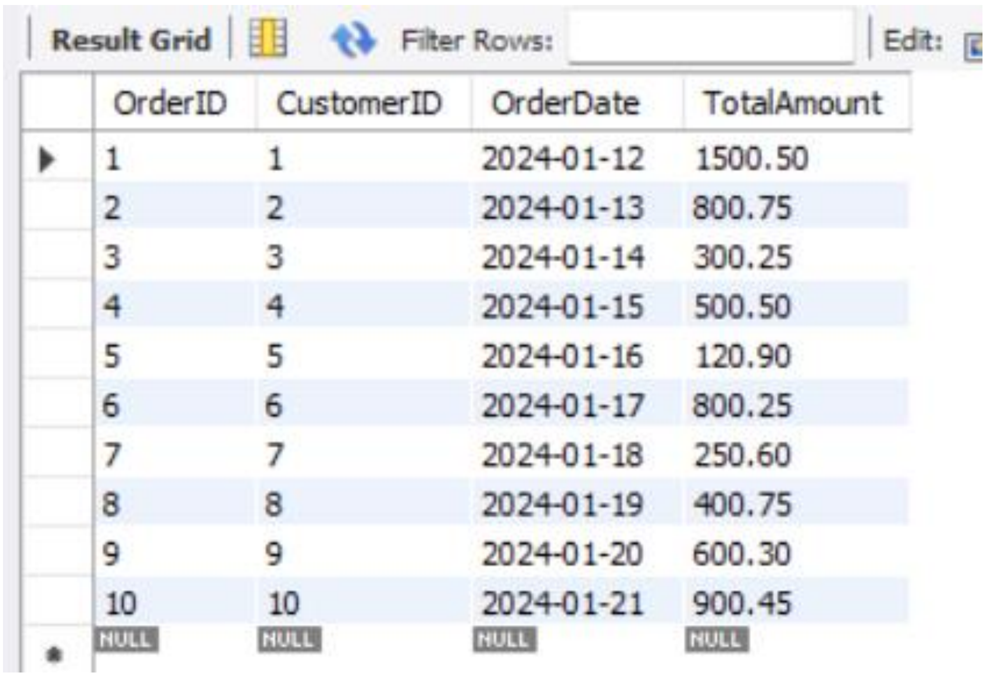
```sql
INSERT INTO products (ProductID, ProductName, Description, Price) VALUES
(1, 'Laptop', 'Powerful and lightweight laptop for productivity', 999.99),
(2, 'Smartphone', 'Latest smartphone with advanced features', 699.99),
(3, 'Headphones', 'High-quality over-ear headphones for immersive audio', 149.99),
(4, 'Camera', 'Professional camera for stunning photography', 1299.99),
(5, 'Smartwatch', 'Fitness tracking and smart notifications on your wrist', 199.99),
(6, 'Tablet', 'Portable tablet for entertainment and productivity', 499.99),
(7, 'Gaming Console', 'Next-gen gaming console for an immersive gaming experience', 499.99),
(8, 'Wireless Speaker', 'Compact wireless speaker for music enthusiasts', 79.99),
(9, 'Coffee Maker', 'Automatic coffee maker for the perfect brew', 129.99),
(10, 'Fitness Tracker', 'Track your fitness activities and monitor health metrics', 79.99);

select * from products;
```

	ProductID	ProductName	Description	price
▶	1	Laptop	Powerful and lightweight laptop for productivity	999.99
	2	Smartphone	Latest smartphone with advanced features	699.99
	3	Headphones	High-quality over-ear headphones for immersiv...	149.99
	4	Camera	Professional camera for stunning photography	1299.99
	5	Smartwatch	Fitness tracking and smart notifications on your ...	199.99
	6	Tablet	Portable tablet for entertainment and productivity	499.99
	7	Gaming Console	Next-gen gaming console for an immersive gami...	499.99
	8	Wireless Speaker	Compact wireless speaker for music enthusiasts	79.99
	9	Coffee Maker	Automatic coffee maker for the perfect brew	129.99
	10	Fitness Tracker	Track your fitness activities and monitor health ...	79.99
	NULL	NULL	NULL	NULL

c. Orders

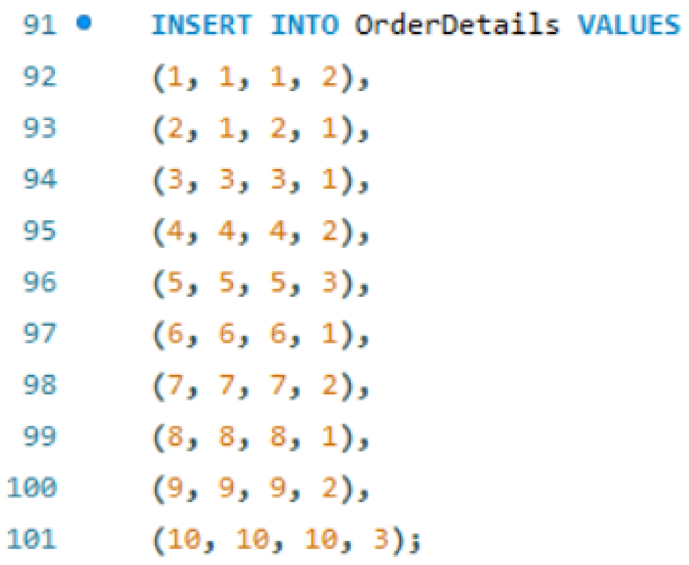


```sql
INSERT INTO Orders VALUES
(1, 1, '2024-01-12', 1500.50),
(2, 2, '2024-01-13', 800.75),
(3, 3, '2024-01-14', 300.25),
(4, 4, '2024-01-15', 500.50),
(5, 5, '2024-01-16', 120.90),
(6, 6, '2024-01-17', 800.25),
(7, 7, '2024-01-18', 250.60),
(8, 8, '2024-01-19', 400.75),
(9, 9, '2024-01-20', 600.30),
(10, 10, '2024-01-21', 900.45);
```

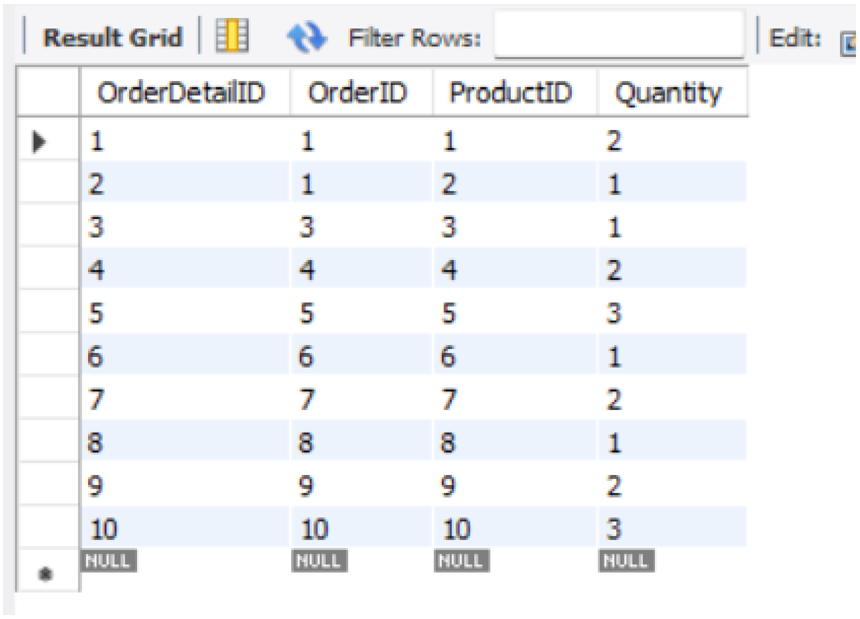


OrderID	CustomerID	OrderDate	TotalAmount
1	1	2024-01-12	1500.50
2	2	2024-01-13	800.75
3	3	2024-01-14	300.25
4	4	2024-01-15	500.50
5	5	2024-01-16	120.90
6	6	2024-01-17	800.25
7	7	2024-01-18	250.60
8	8	2024-01-19	400.75
9	9	2024-01-20	600.30
10	10	2024-01-21	900.45
NULL	NULL	NULL	NULL

d. OrderDetails

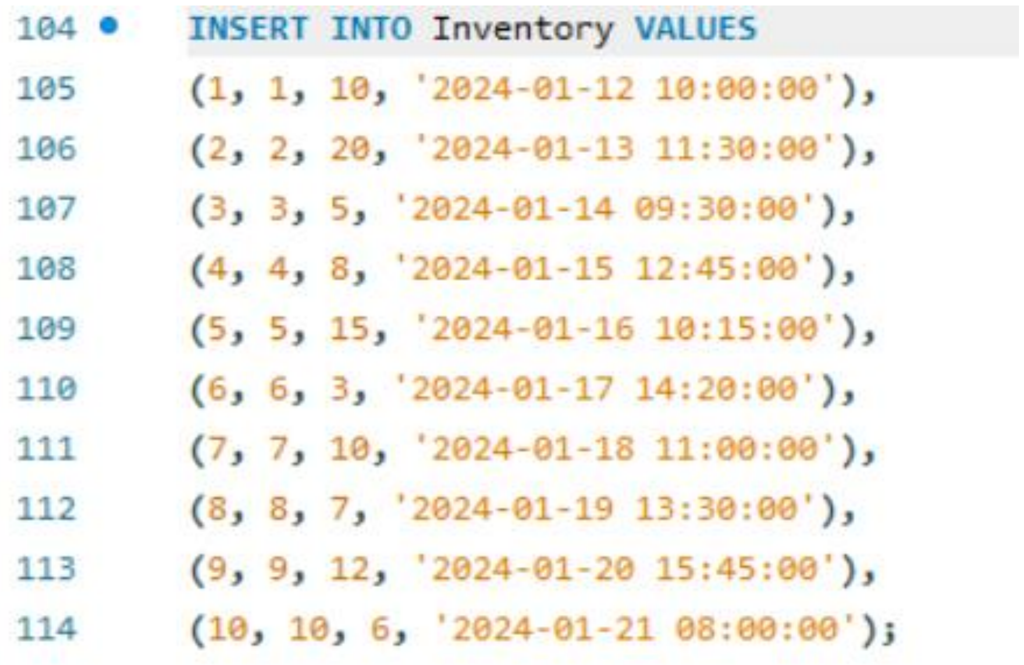


```
INSERT INTO OrderDetails VALUES
(1, 1, 1, 2),
(2, 1, 2, 1),
(3, 3, 3, 1),
(4, 4, 4, 2),
(5, 5, 5, 3),
(6, 6, 6, 1),
(7, 7, 7, 2),
(8, 8, 8, 1),
(9, 9, 9, 2),
(10, 10, 10, 3);
```

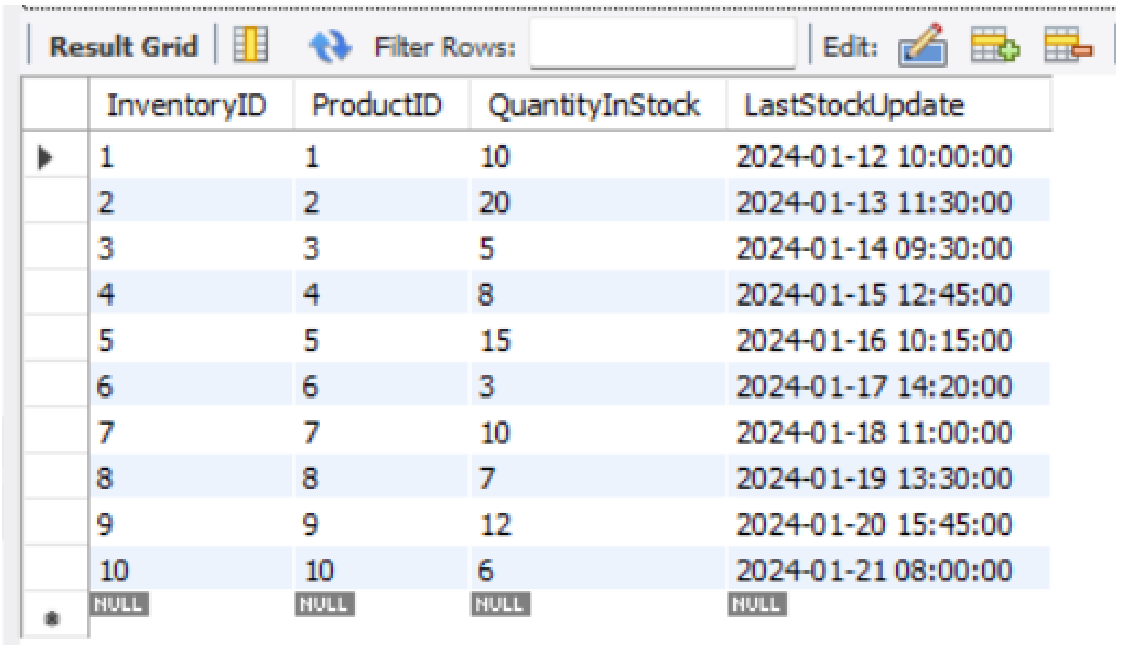


OrderDetailID	OrderID	ProductID	Quantity
1	1	1	2
2	1	2	1
3	3	3	1
4	4	4	2
5	5	5	3
6	6	6	1
7	7	7	2
8	8	8	1
9	9	9	2
10	10	10	3
NULL	NULL	NULL	NULL

e. Inventory



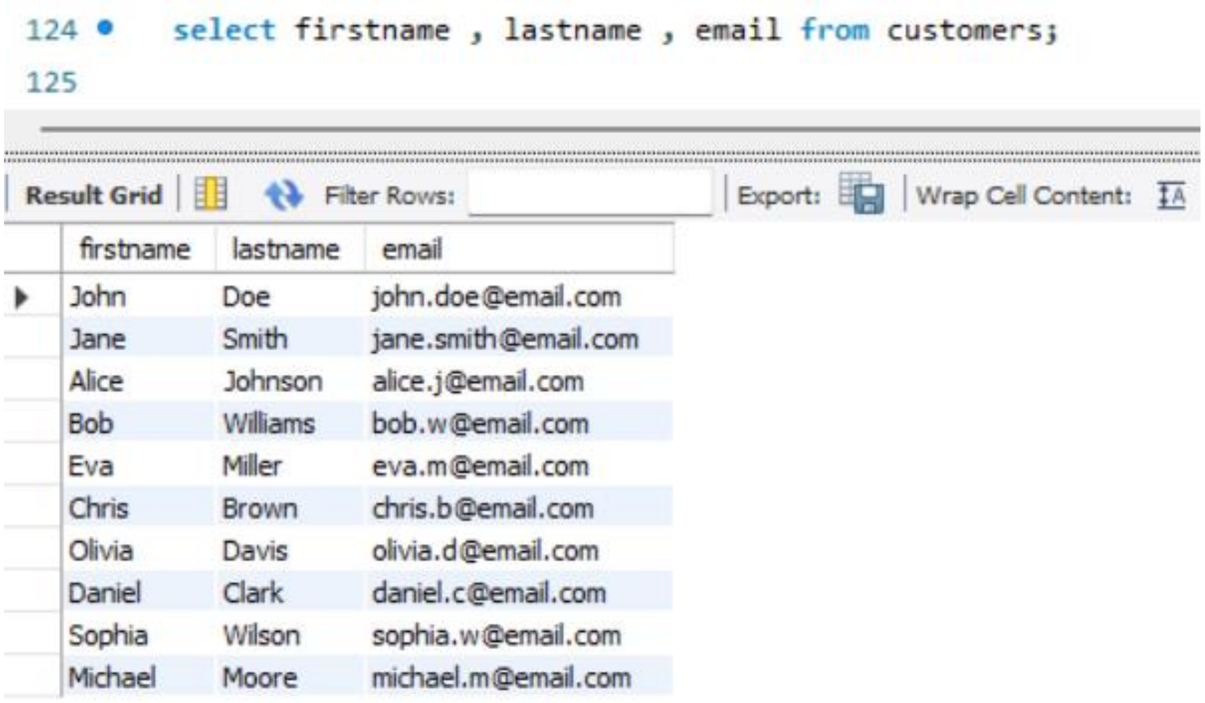
```sql
INSERT INTO Inventory VALUES
(1, 1, 10, '2024-01-12 10:00:00'),
(2, 2, 20, '2024-01-13 11:30:00'),
(3, 3, 5, '2024-01-14 09:30:00'),
(4, 4, 8, '2024-01-15 12:45:00'),
(5, 5, 15, '2024-01-16 10:15:00'),
(6, 6, 3, '2024-01-17 14:20:00'),
(7, 7, 10, '2024-01-18 11:00:00'),
(8, 8, 7, '2024-01-19 13:30:00'),
(9, 9, 12, '2024-01-20 15:45:00'),
(10, 10, 6, '2024-01-21 08:00:00');
```



InventoryID	ProductID	QuantityInStock	LastStockUpdate
1	1	10	2024-01-12 10:00:00
2	2	20	2024-01-13 11:30:00
3	3	5	2024-01-14 09:30:00
4	4	8	2024-01-15 12:45:00
5	5	15	2024-01-16 10:15:00
6	6	3	2024-01-17 14:20:00
7	7	10	2024-01-18 11:00:00
8	8	7	2024-01-19 13:30:00
9	9	12	2024-01-20 15:45:00
10	10	6	2024-01-21 08:00:00
NULL	NULL	NULL	NULL

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

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

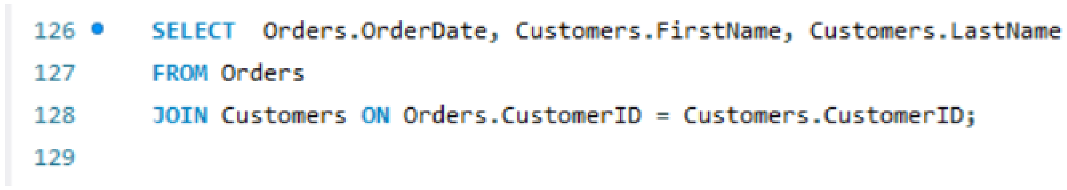


```
select firstname , lastname , email from customers;
```

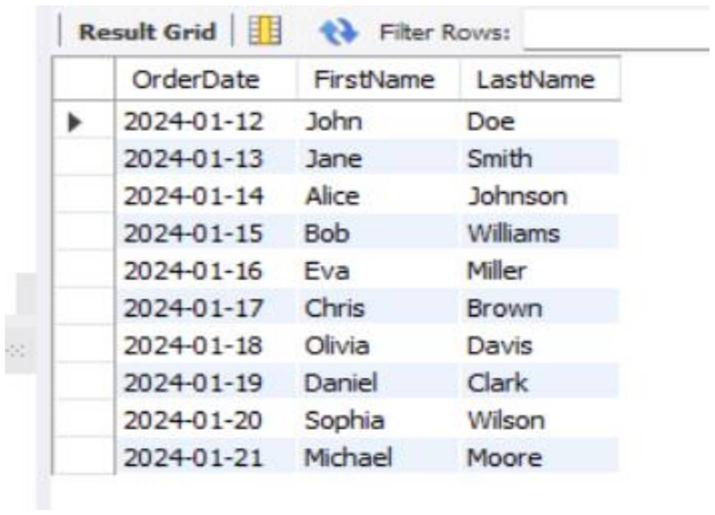
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

firstname	lastname	email
John	Doe	john.doe@email.com
Jane	Smith	jane.smith@email.com
Alice	Johnson	alice.j@email.com
Bob	Williams	bob.w@email.com
Eva	Miller	eva.m@email.com
Chris	Brown	chris.b@email.com
Olivia	Davis	olivia.d@email.com
Daniel	Clark	daniel.c@email.com
Sophia	Wilson	sophia.w@email.com
Michael	Moore	michael.m@email.com

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



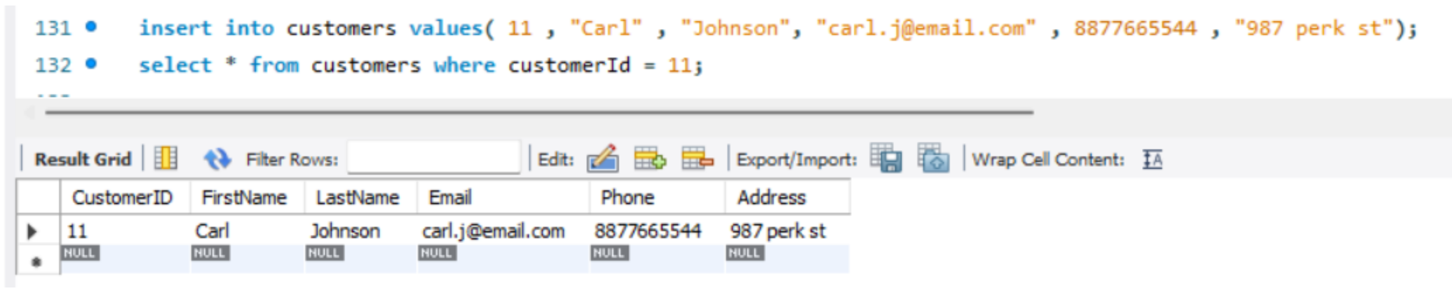
```
SELECT  Orders.OrderDate, Customers.FirstName, Customers.LastName
FROM Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```



Result Grid | Filter Rows:

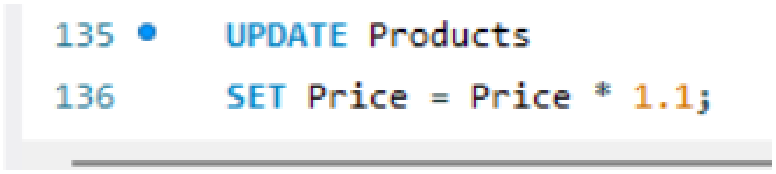
OrderDate	FirstName	LastName
2024-01-12	John	Doe
2024-01-13	Jane	Smith
2024-01-14	Alice	Johnson
2024-01-15	Bob	Williams
2024-01-16	Eva	Miller
2024-01-17	Chris	Brown
2024-01-18	Olivia	Davis
2024-01-19	Daniel	Clark
2024-01-20	Sophia	Wilson
2024-01-21	Michael	Moore

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

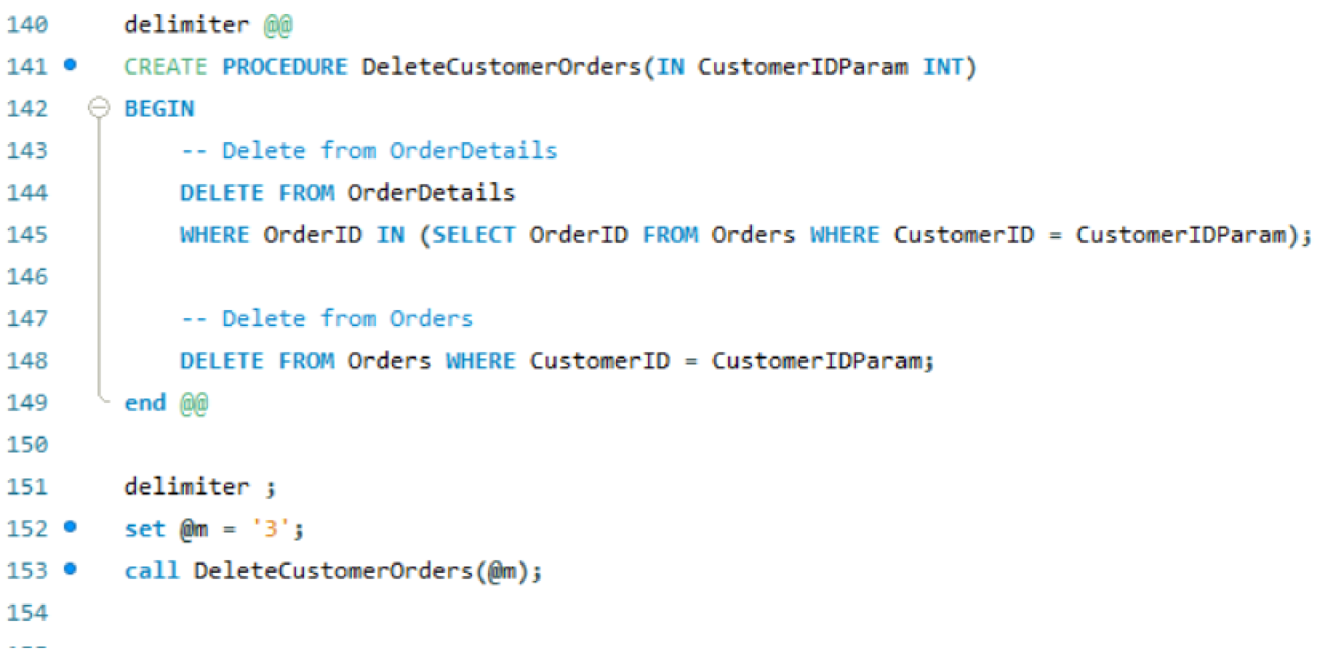

```
insert into customers values( 11 , "Carl" , "Johnson", "carl.j@email.com" , 8877665544 , "987 perk st");
select * from customers where customerId = 11;
```

CustomerID	FirstName	LastName	Email	Phone	Address
11	Carl	Johnson	carl.j@email.com	8877665544	987 perk st
NULL	NULL	NULL	NULL	NULL	NULL

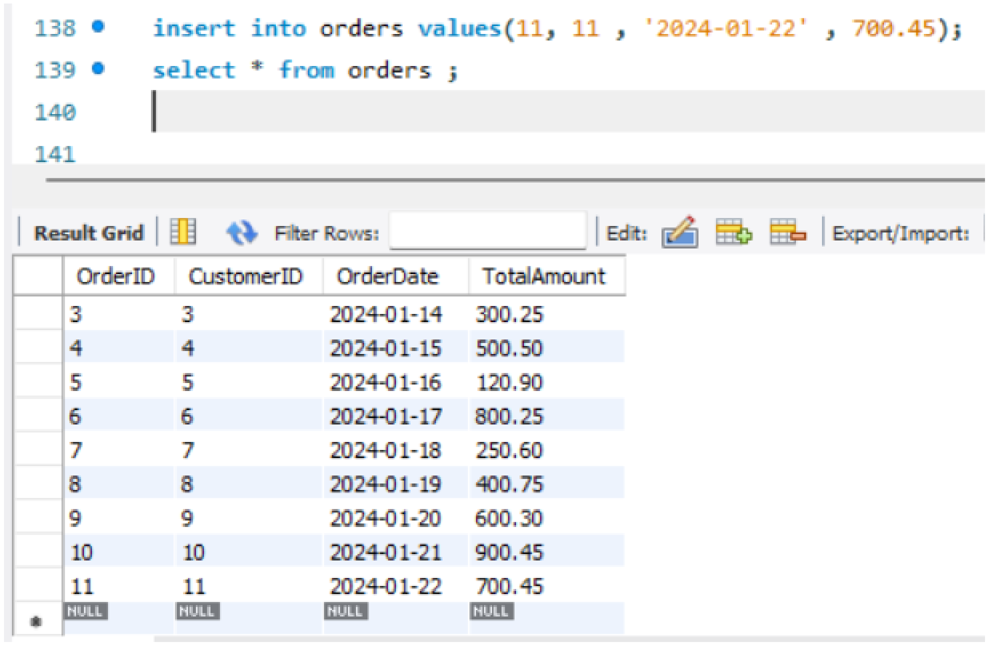
4. Write an SQL query to update the prices of all electronic gadgets in the "Products" table by increasing them by 10%.


```
UPDATE Products
SET Price = Price * 1.1;
```

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.


```
delimiter @@
CREATE PROCEDURE DeleteCustomerOrders(IN CustomerIDParam INT)
BEGIN
    -- Delete from OrderDetails
    DELETE FROM OrderDetails
    WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = CustomerIDParam);

    -- Delete from Orders
    DELETE FROM Orders WHERE CustomerID = CustomerIDParam;
end @@

delimiter ;
set @m = '3';
call DeleteCustomerOrders(@m);
```

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.

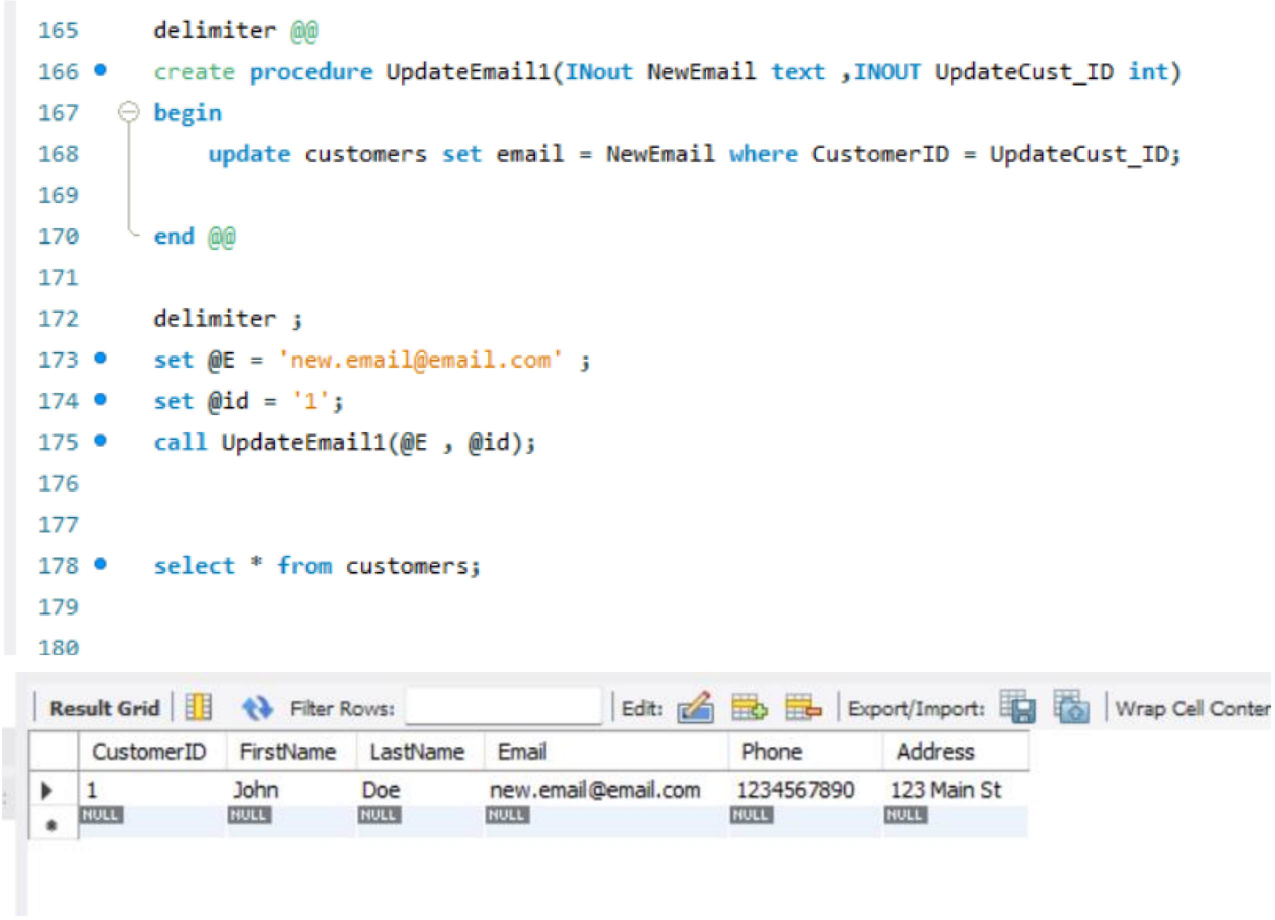


```
insert into orders values(11, 11 , '2024-01-22' , 700.45);
select * from orders ;
```

```
Result Grid | Filter Rows: | Edit: | Export/Import:
```

OrderID	CustomerID	OrderDate	TotalAmount
3	3	2024-01-14	300.25
4	4	2024-01-15	500.50
5	5	2024-01-16	120.90
6	6	2024-01-17	800.25
7	7	2024-01-18	250.60
8	8	2024-01-19	400.75
9	9	2024-01-20	600.30
10	10	2024-01-21	900.45
11	11	2024-01-22	700.45
NULL	NULL	NULL	NULL

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.

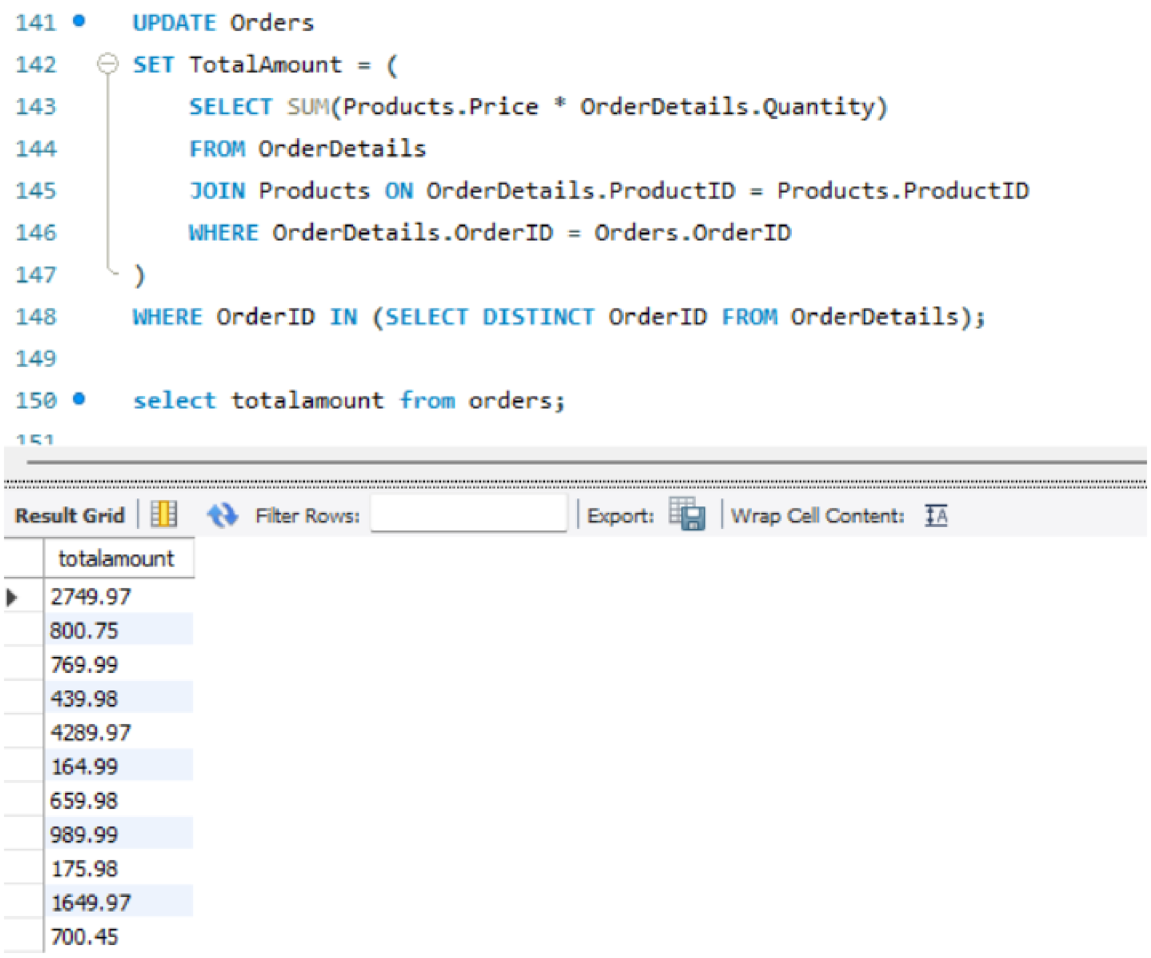


```
delimiter @@
create procedure UpdateEmail1(INout NewEmail text ,INOUT UpdateCust_ID int)
begin
    update customers set email = NewEmail where CustomerID = UpdateCust_ID;

end @@

delimiter ;
set @E = 'new.email@email.com' ;
set @id = '1';
call UpdateEmail1(@E , @id);


select * from customers;
```

```
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Conter
```

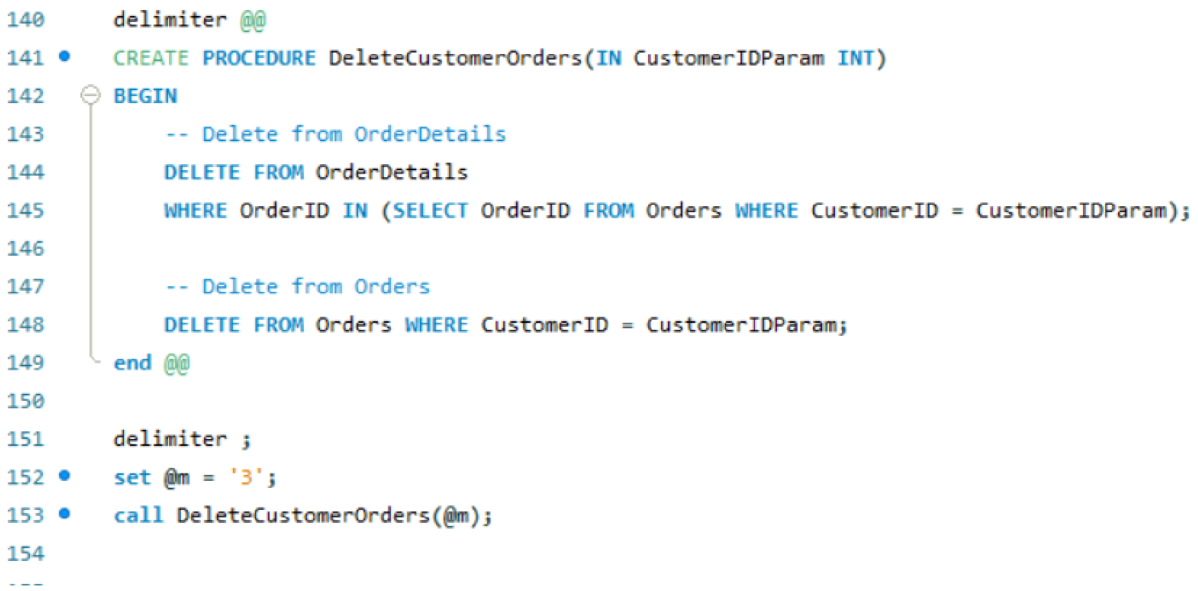
CustomerID	FirstName	LastName	Email	Phone	Address
1	John	Doe	new.email@email.com	1234567890	123 Main St
NULL	NULL	NULL	NULL	NULL	NULL

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(Products.Price * OrderDetails.Quantity)
    FROM OrderDetails
    JOIN Products ON OrderDetails.ProductID = Products.ProductID
    WHERE OrderDetails.OrderID = Orders.OrderID
)
WHERE OrderID IN (SELECT DISTINCT OrderID FROM OrderDetails);

select totalamount from orders;
```

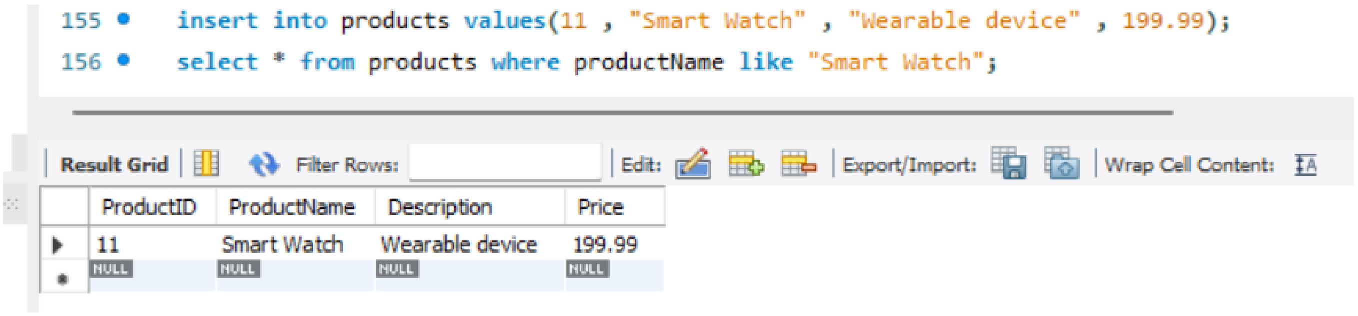
```
totalamount
2749.97
800.75
769.99
439.98
4289.97
164.99
659.98
989.99
175.98
1649.97
700.45
```

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.


```
delimiter @@
CREATE PROCEDURE DeleteCustomerOrders(IN CustomerIDParam INT)
BEGIN
    -- Delete from OrderDetails
    DELETE FROM OrderDetails
    WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = CustomerIDParam);

    -- Delete from Orders
    DELETE FROM Orders WHERE CustomerID = CustomerIDParam;
end @@

delimiter ;
set @m = '3';
call DeleteCustomerOrders(@m);
```

10. Write an SQL query to insert a new electronic gadget product into the "Products" table,

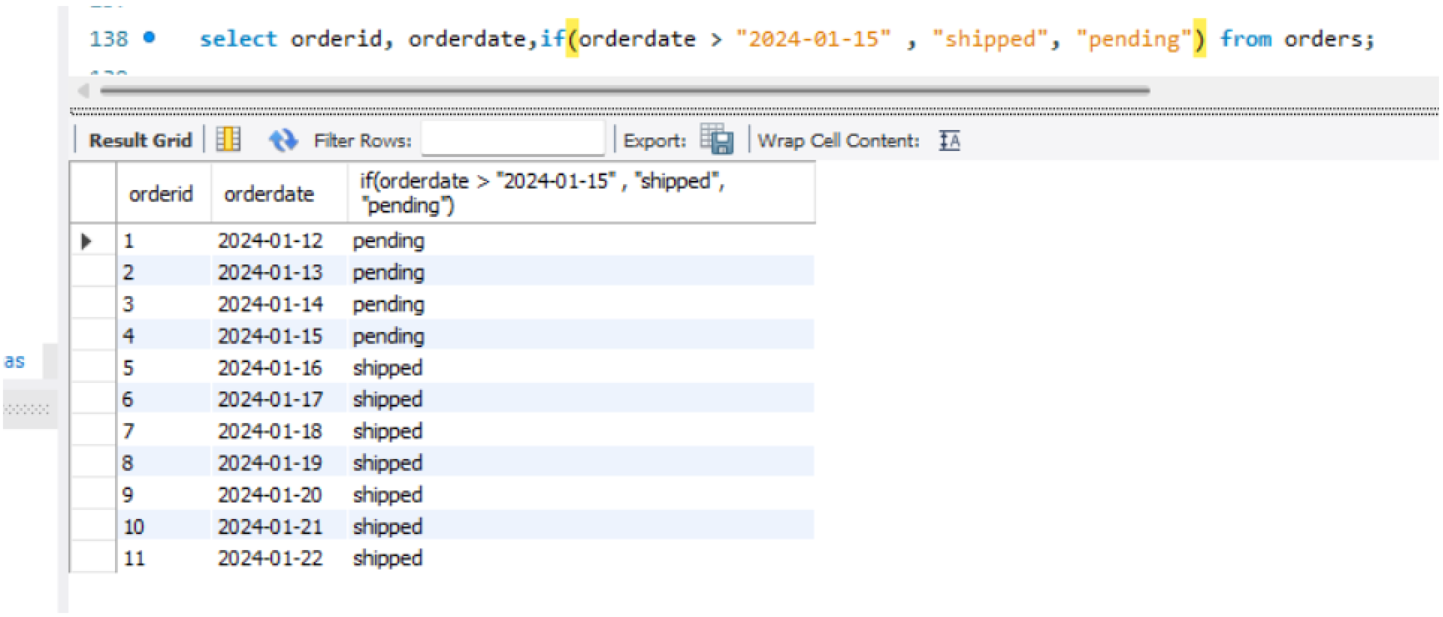
including product name, category, price, and any other relevant details.



```
insert into products values(11 , "Smart Watch" , "Wearable device" , 199.99);
select * from products where productName like "Smart Watch";
```

ProductID	ProductName	Description	Price
11	Smart Watch	Wearable device	199.99
NULL	NULL	NULL	NULL

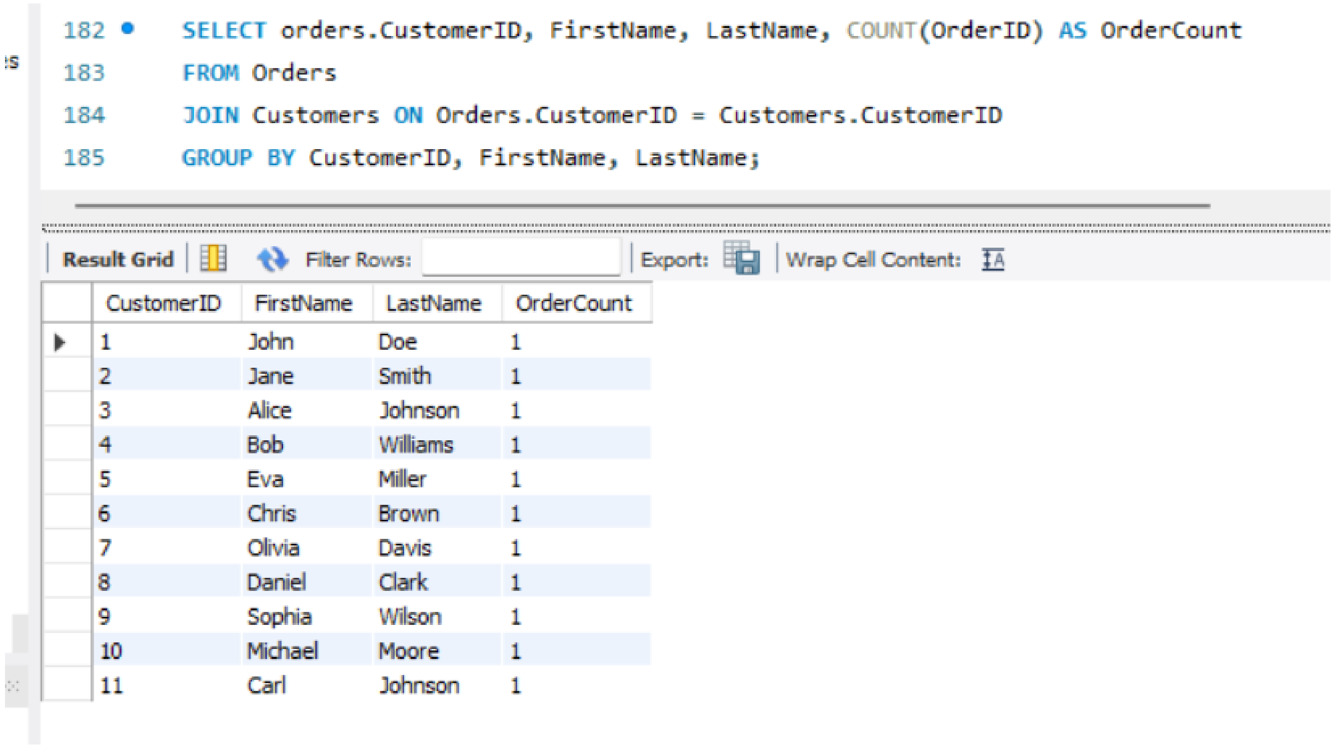
11. Write an SQL query to update the status of a specific order in the "Orders" table (e.g., from "Pending" to "Shipped"). Allow users to input the order ID and the new status.



```
select orderid, orderdate,if(orderdate > "2024-01-15" , "shipped", "pending") from orders;
```

orderid	orderdate	if(orderdate > "2024-01-15" , "shipped", "pending")
1	2024-01-12	pending
2	2024-01-13	pending
3	2024-01-14	pending
4	2024-01-15	pending
5	2024-01-16	shipped
6	2024-01-17	shipped
7	2024-01-18	shipped
8	2024-01-19	shipped
9	2024-01-20	shipped
10	2024-01-21	shipped
11	2024-01-22	shipped

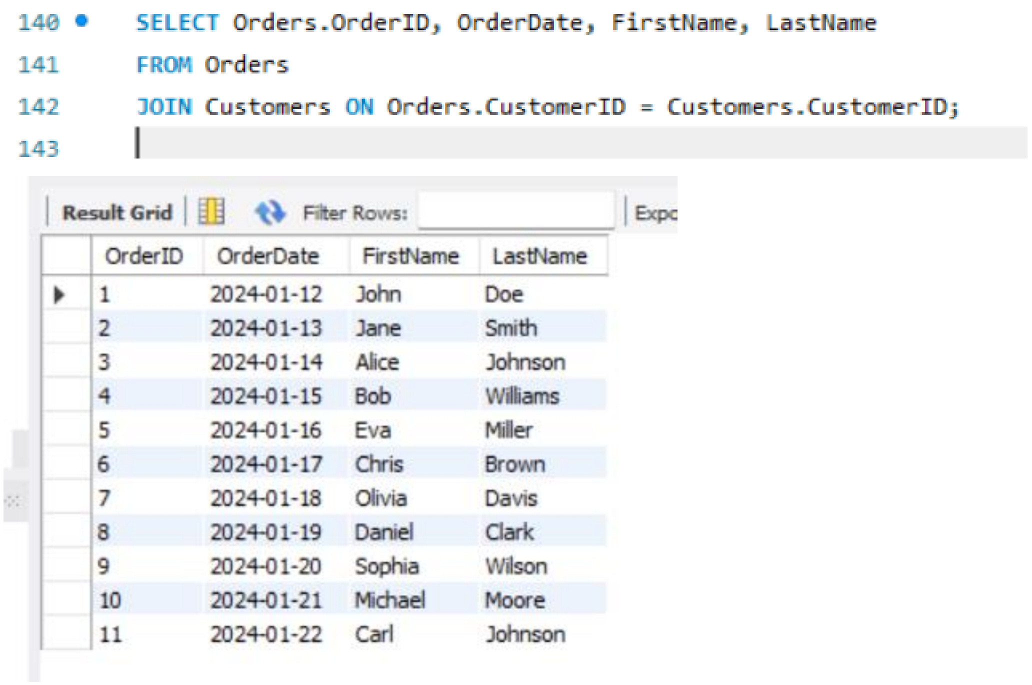
12. Write an SQL query to calculate and update the number of orders placed by each customer in the "Customers" table based on the data in the "Orders" table.


```sql
SELECT orders.CustomerID, FirstName, LastName, COUNT(OrderID) AS OrderCount
FROM Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID
GROUP BY CustomerID, FirstName, LastName;
```

CustomerID	FirstName	LastName	OrderCount
1	John	Doe	1
2	Jane	Smith	1
3	Alice	Johnson	1
4	Bob	Williams	1
5	Eva	Miller	1
6	Chris	Brown	1
7	Olivia	Davis	1
8	Daniel	Clark	1
9	Sophia	Wilson	1
10	Michael	Moore	1
11	Carl	Johnson	1

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

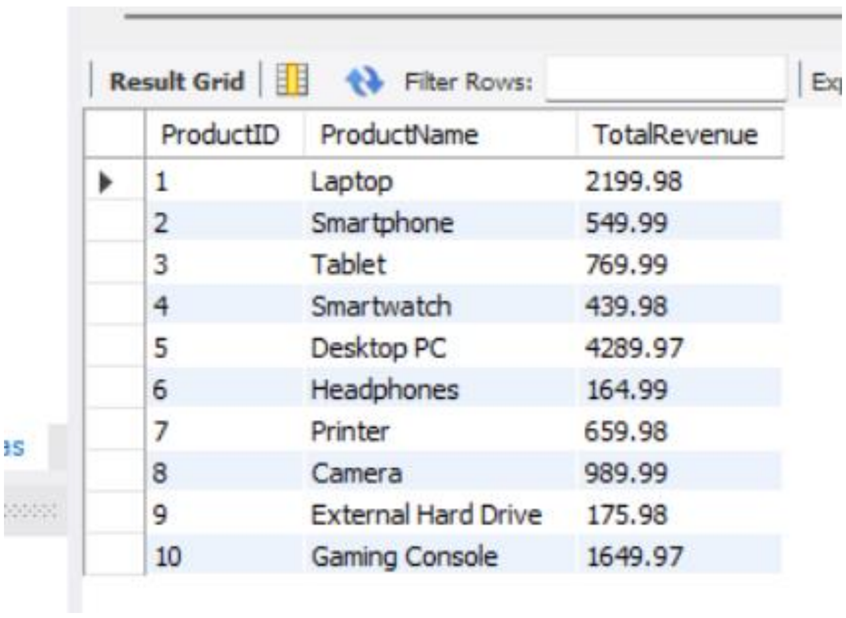
1. Write an SQL query to retrieve a list of all orders along with customer information (e.g., customer name) for each order.


```sql
SELECT Orders.OrderID, OrderDate, FirstName, LastName
FROM Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

OrderID	OrderDate	FirstName	LastName
1	2024-01-12	John	Doe
2	2024-01-13	Jane	Smith
3	2024-01-14	Alice	Johnson
4	2024-01-15	Bob	Williams
5	2024-01-16	Eva	Miller
6	2024-01-17	Chris	Brown
7	2024-01-18	Olivia	Davis
8	2024-01-19	Daniel	Clark
9	2024-01-20	Sophia	Wilson
10	2024-01-21	Michael	Moore
11	2024-01-22	Carl	Johnson

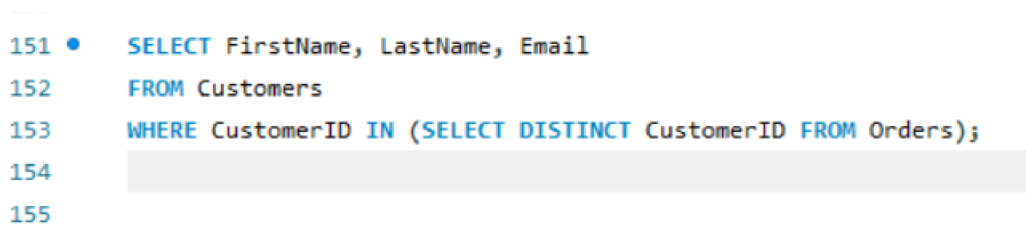
2. Write an SQL query to find the total revenue generated by each electronic gadget product. Include the product name and the total revenue.

```sql
SELECT Products.ProductID, ProductName, SUM(Quantity * Price) AS TotalRevenue
FROM OrderDetails
JOIN Products ON OrderDetails.ProductID = Products.ProductID
GROUP BY Products.ProductID, ProductName;
```

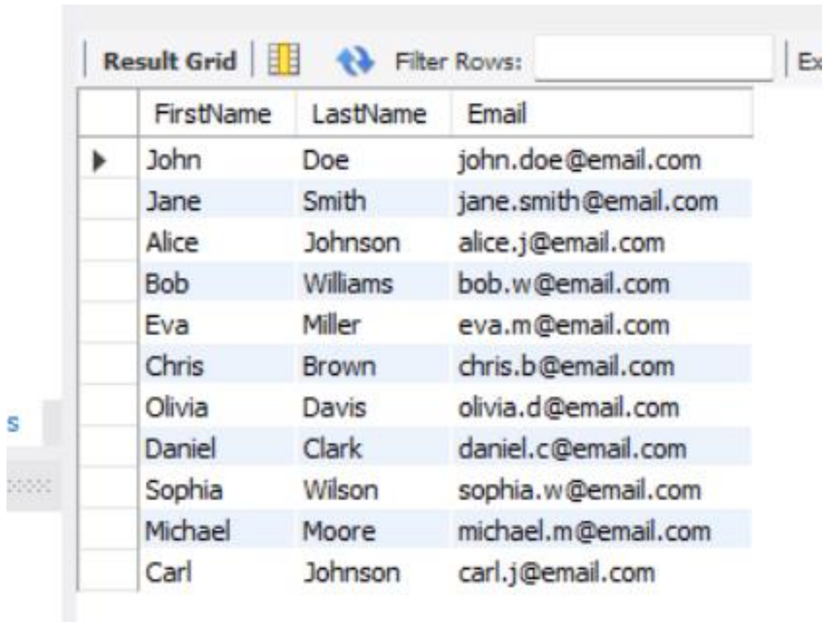


ProductID	ProductName	TotalRevenue
1	Laptop	2199.98
2	Smartphone	549.99
3	Tablet	769.99
4	Smartwatch	439.98
5	Desktop PC	4289.97
6	Headphones	164.99
7	Printer	659.98
8	Camera	989.99
9	External Hard Drive	175.98
10	Gaming Console	1649.97

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

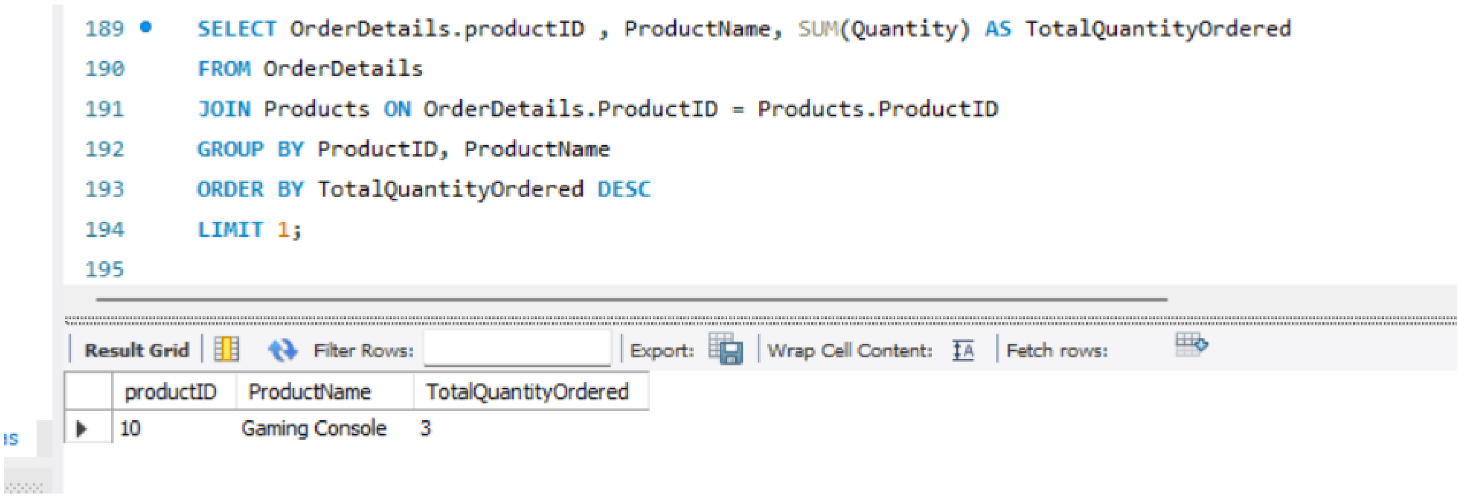


```sql
SELECT FirstName, LastName, Email
FROM Customers
WHERE CustomerID IN (SELECT DISTINCT CustomerID FROM Orders);
```



FirstName	LastName	Email
John	Doe	john.doe@email.com
Jane	Smith	jane.smith@email.com
Alice	Johnson	alice.j@email.com
Bob	Williams	bob.w@email.com
Eva	Miller	eva.m@email.com
Chris	Brown	chris.b@email.com
Olivia	Davis	olivia.d@email.com
Daniel	Clark	daniel.c@email.com
Sophia	Wilson	sophia.w@email.com
Michael	Moore	michael.m@email.com
Carl	Johnson	carl.j@email.com

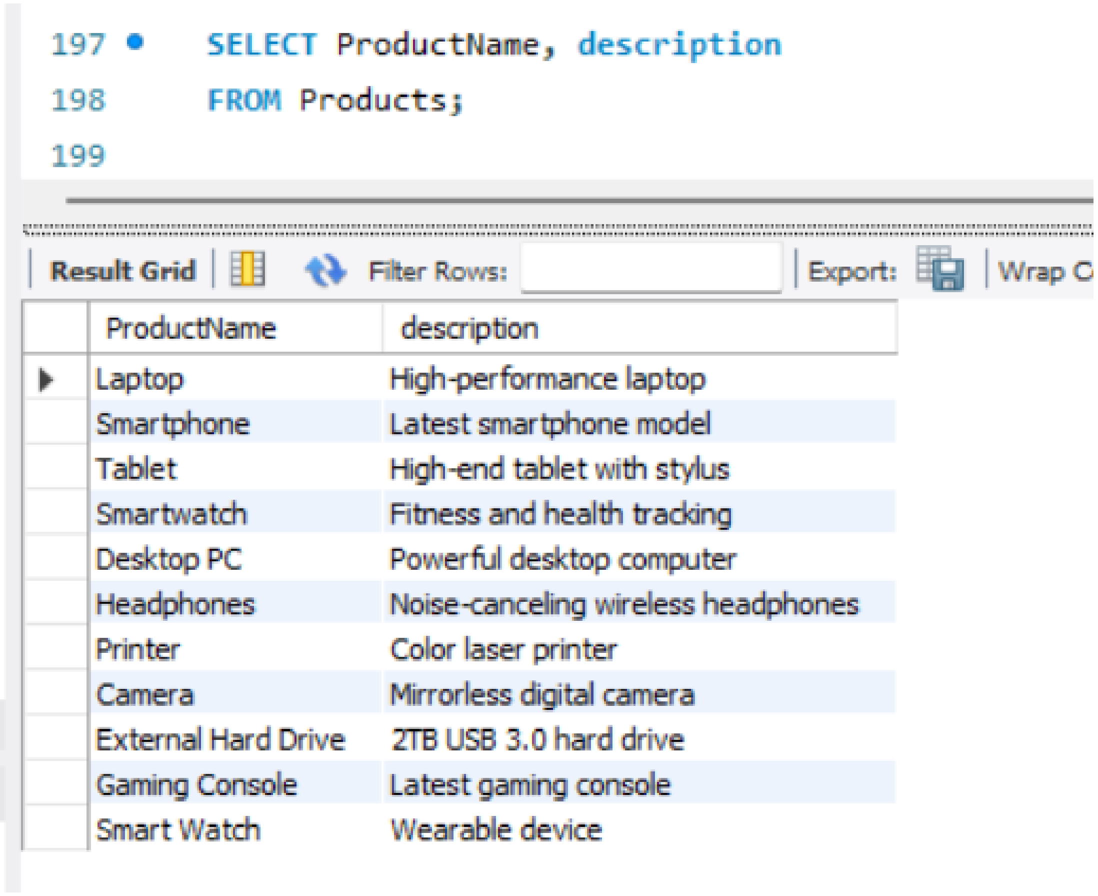
4. Write an SQL query to find the most popular electronic gadget, which is the one with the highest total quantity ordered. Include the product name and the total quantity ordered.



```sql
SELECT OrderDetails.productID , ProductName, SUM(Quantity) AS TotalQuantityOrdered
FROM OrderDetails
JOIN Products ON OrderDetails.ProductID = Products.ProductID
GROUP BY ProductID, ProductName
ORDER BY TotalQuantityOrdered DESC
LIMIT 1;
```

productID	ProductName	TotalQuantityOrdered
10	Gaming Console	3

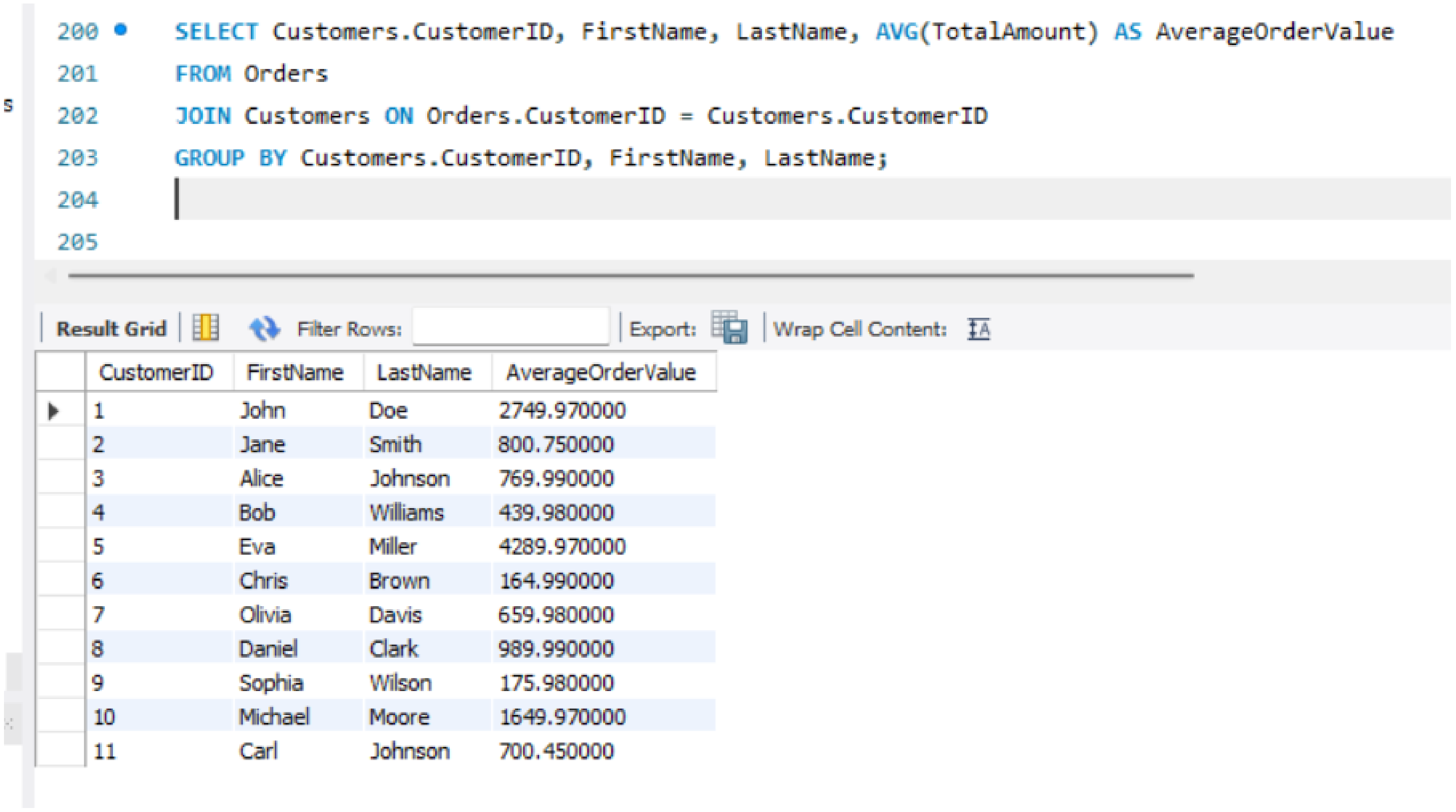
5. Write an SQL query to retrieve a list of electronic gadgets along with their corresponding categories.



```sql
SELECT ProductName, description
FROM Products;
```

ProductName	description
Laptop	High-performance laptop
Smartphone	Latest smartphone model
Tablet	High-end tablet with stylus
Smartwatch	Fitness and health tracking
Desktop PC	Powerful desktop computer
Headphones	Noise-canceling wireless headphones
Printer	Color laser printer
Camera	Mirrorless digital camera
External Hard Drive	2TB USB 3.0 hard drive
Gaming Console	Latest gaming console
Smart Watch	Wearable device

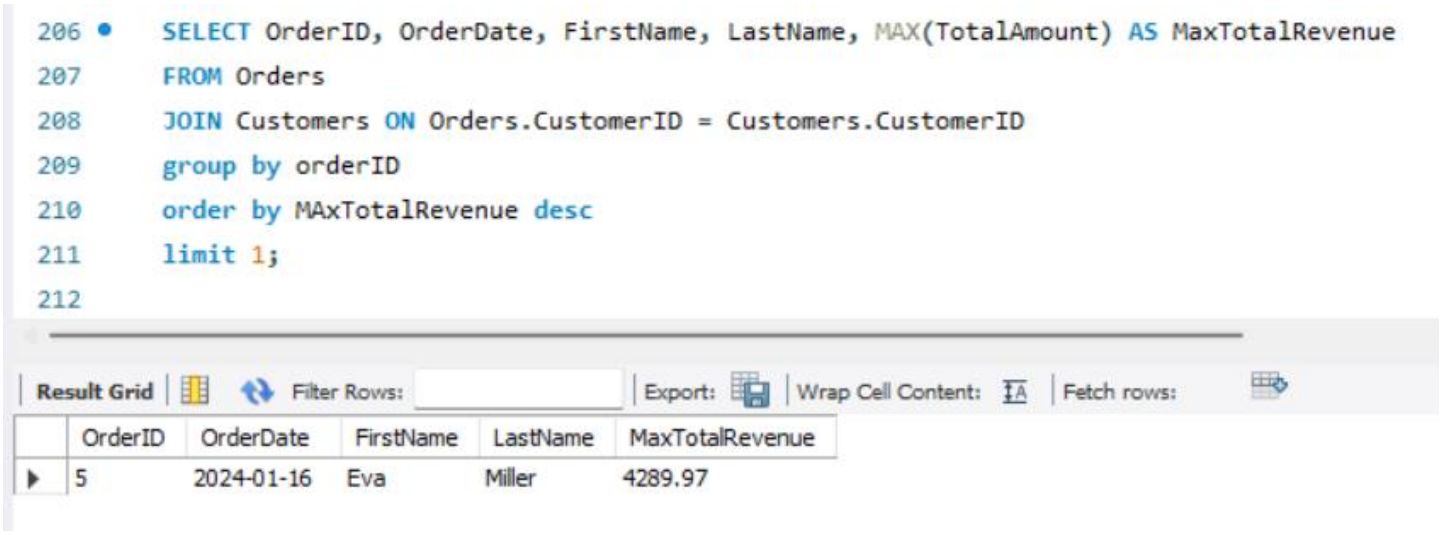
6. Write an SQL query to calculate the average order value for each customer. Include the customer's name and their average order value.



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

CustomerID	FirstName	LastName	AverageOrderValue
1	John	Doe	2749.970000
2	Jane	Smith	800.750000
3	Alice	Johnson	769.990000
4	Bob	Williams	439.980000
5	Eva	Miller	4289.970000
6	Chris	Brown	164.990000
7	Olivia	Davis	659.980000
8	Daniel	Clark	989.990000
9	Sophia	Wilson	175.980000
10	Michael	Moore	1649.970000
11	Carl	Johnson	700.450000

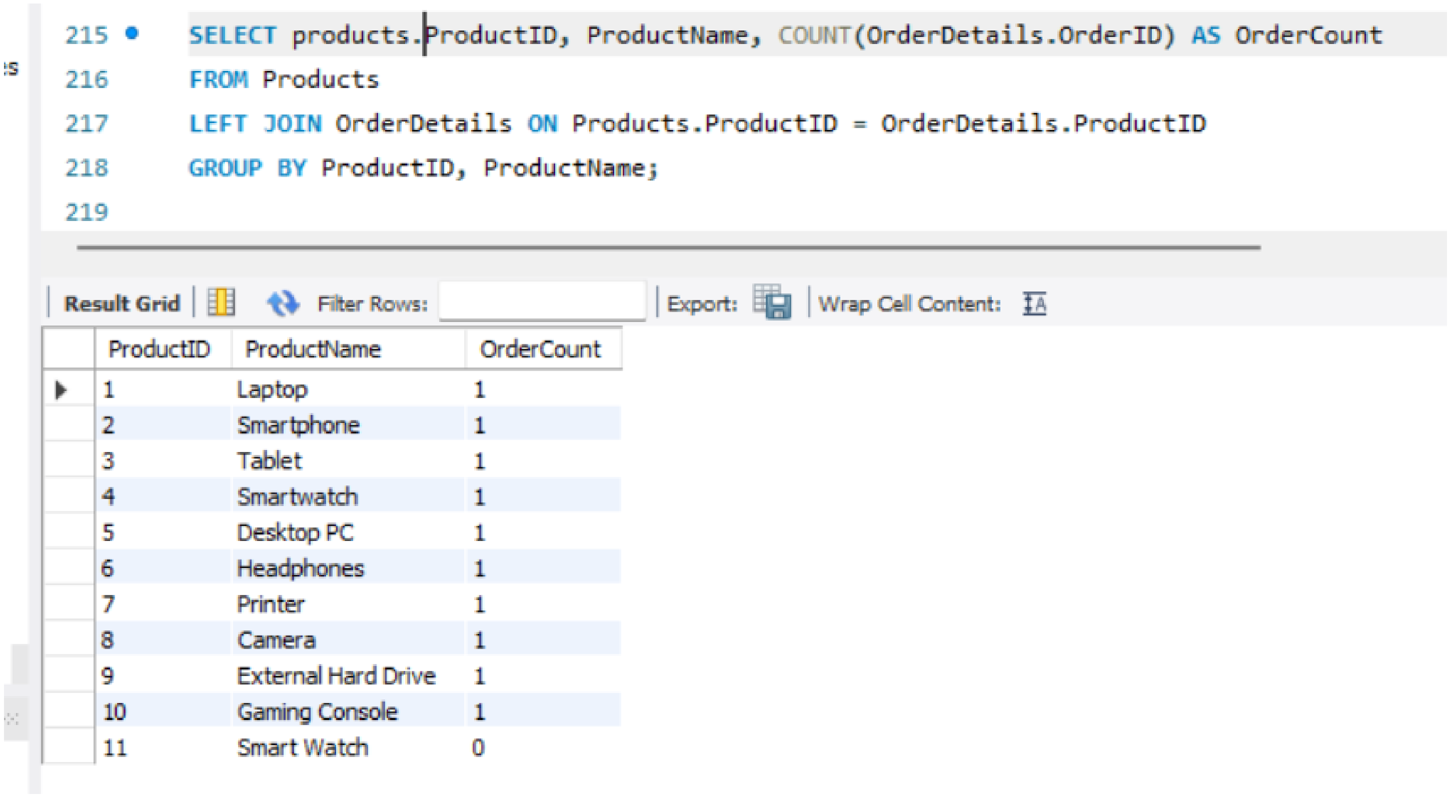
7. Write an SQL query to find the order with the highest total revenue. Include the order ID, customer information, and the total revenue.



```sql
SELECT OrderID, OrderDate, FirstName, LastName, MAX(TotalAmount) AS MaxTotalRevenue
FROM Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID
group by orderID
order by MAxTotalRevenue desc
limit 1;
```

OrderID	OrderDate	FirstName	LastName	MaxTotalRevenue
5	2024-01-16	Eva	Miller	4289.97

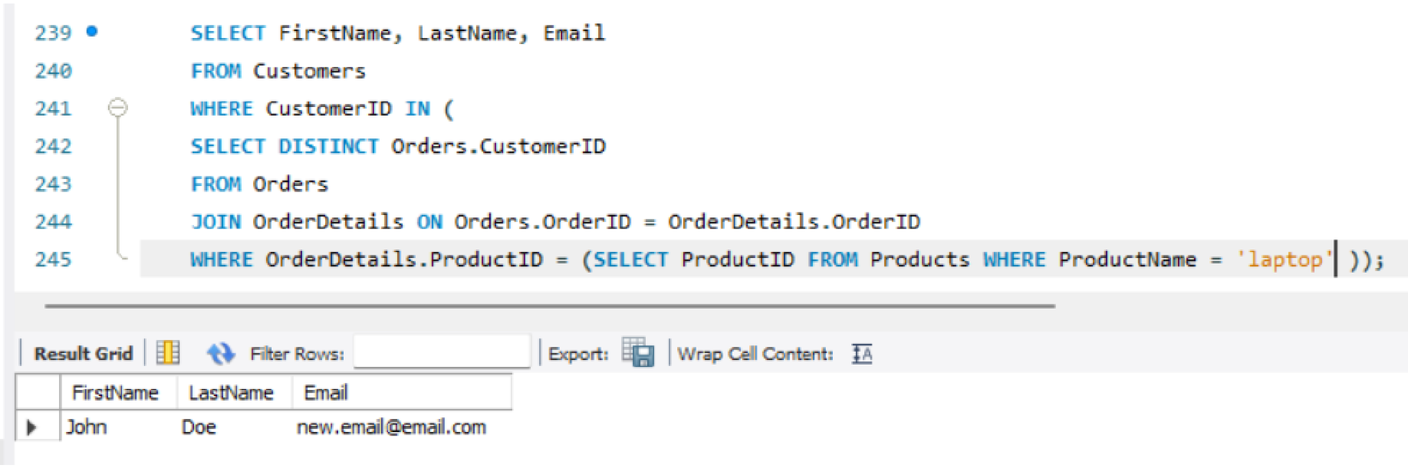
8. Write an SQL query to list electronic gadgets and the number of times each product has been ordered.



```sql
SELECT products.ProductID, ProductName, COUNT(OrderDetails.OrderID) AS OrderCount
FROM Products
LEFT JOIN OrderDetails ON Products.ProductID = OrderDetails.ProductID
GROUP BY ProductID, ProductName;
```

ProductID	ProductName	OrderCount
1	Laptop	1
2	Smartphone	1
3	Tablet	1
4	Smartwatch	1
5	Desktop PC	1
6	Headphones	1
7	Printer	1
8	Camera	1
9	External Hard Drive	1
10	Gaming Console	1
11	Smart Watch	0

9. Write an SQL query to find customers who have purchased a specific electronic gadget product. Allow users to input the product name as a parameter.



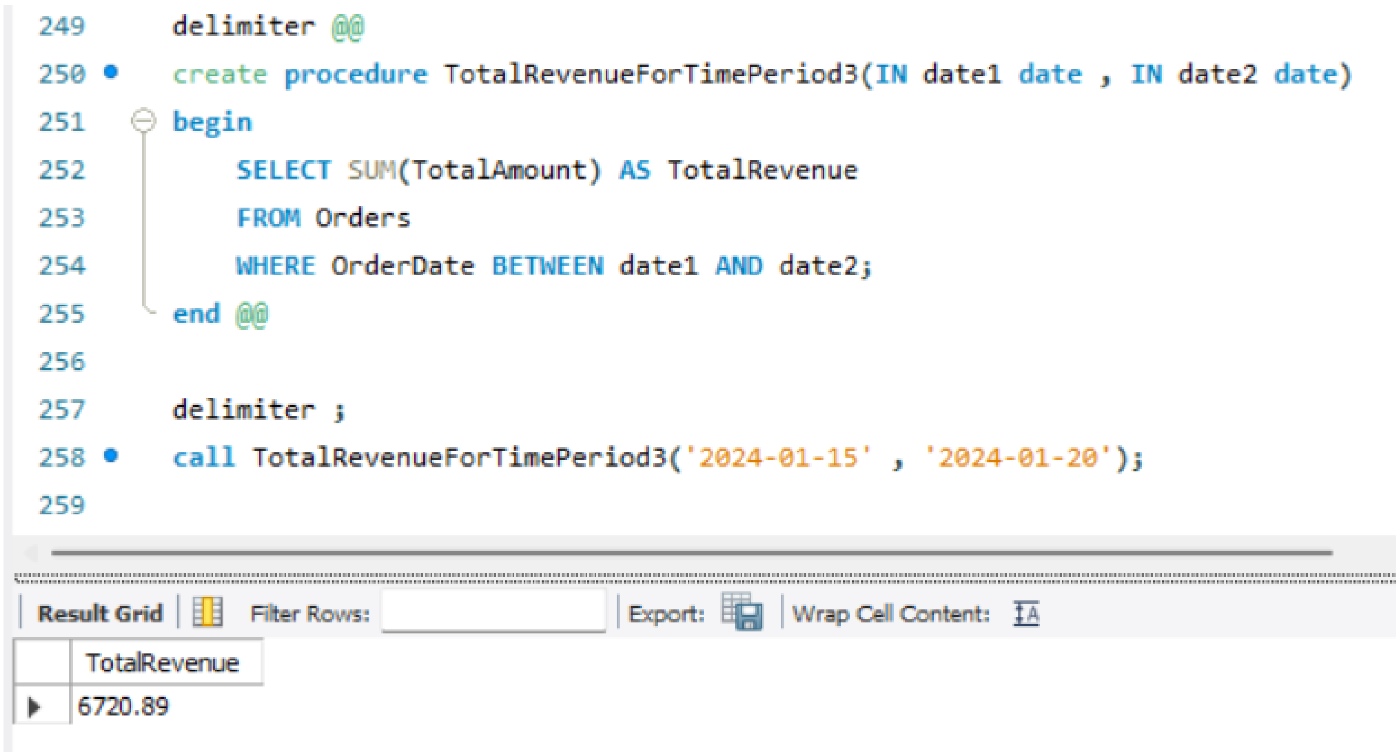
```
SELECT FirstName, LastName, Email
FROM Customers
WHERE CustomerID IN (
SELECT DISTINCT Orders.CustomerID
FROM Orders
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
WHERE OrderDetails.ProductID = (SELECT ProductID FROM Products WHERE ProductName = 'laptop' ));
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	FirstName	LastName	Email
▶	John	Doe	new.email@email.com

as

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.



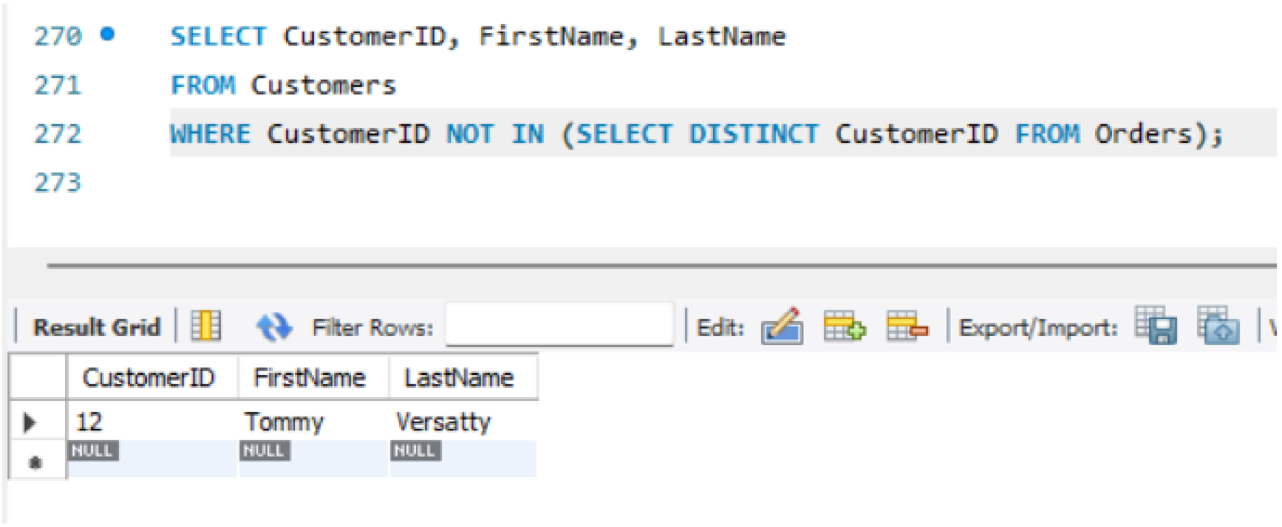
```
delimiter @@
create procedure TotalRevenueForTimePeriod3(IN date1 date , IN date2 date)
begin
    SELECT SUM(TotalAmount) AS TotalRevenue
    FROM Orders
    WHERE OrderDate BETWEEN date1 AND date2;
end @@

delimiter ;
call TotalRevenueForTimePeriod3('2024-01-15' , '2024-01-20');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	TotalRevenue
▶	6720.89

Task 4. Subquery and its type:

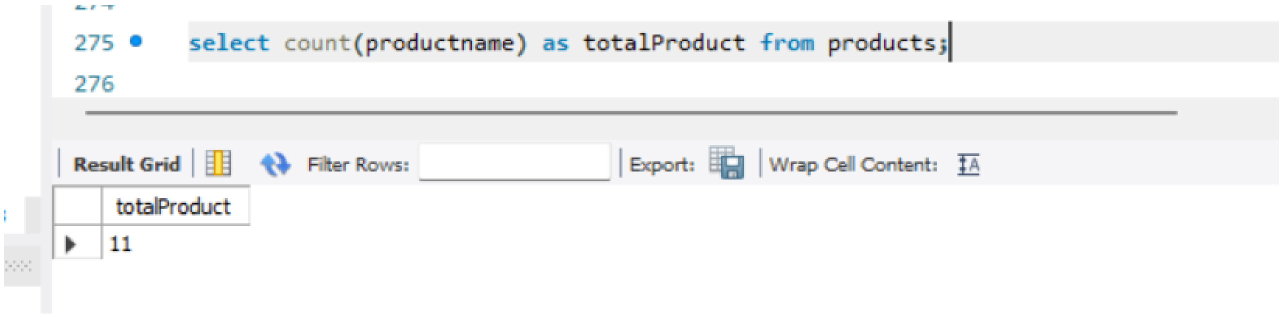
1. Write an SQL query to find out which customers have not placed any orders.



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

	CustomerID	FirstName	LastName
▶	12	Tommy	Versatty
	NULL	NULL	NULL

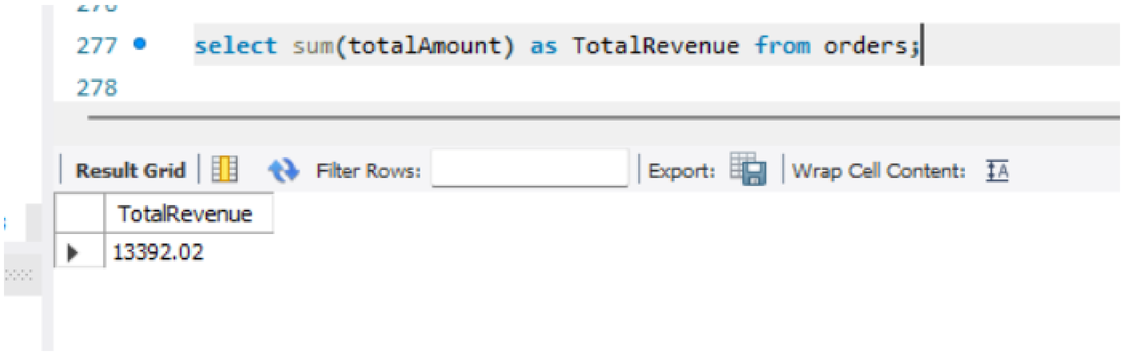
2. Write an SQL query to find the total number of products available for sale.



```sql
select count(productname) as totalProduct from products;
```

	totalProduct
▶	11

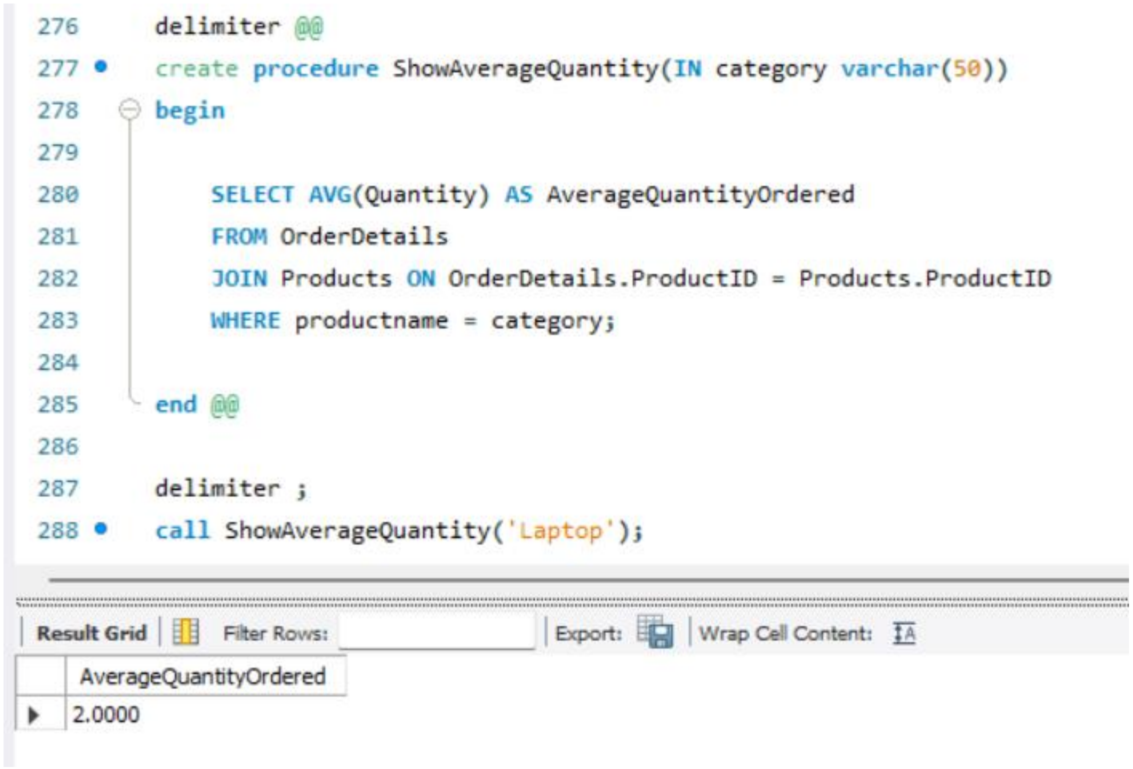
3. Write an SQL query to calculate the total revenue generated by TechShop.



```sql
select sum(totalAmount) as TotalRevenue from orders;
```

	TotalRevenue
▶	13392.02

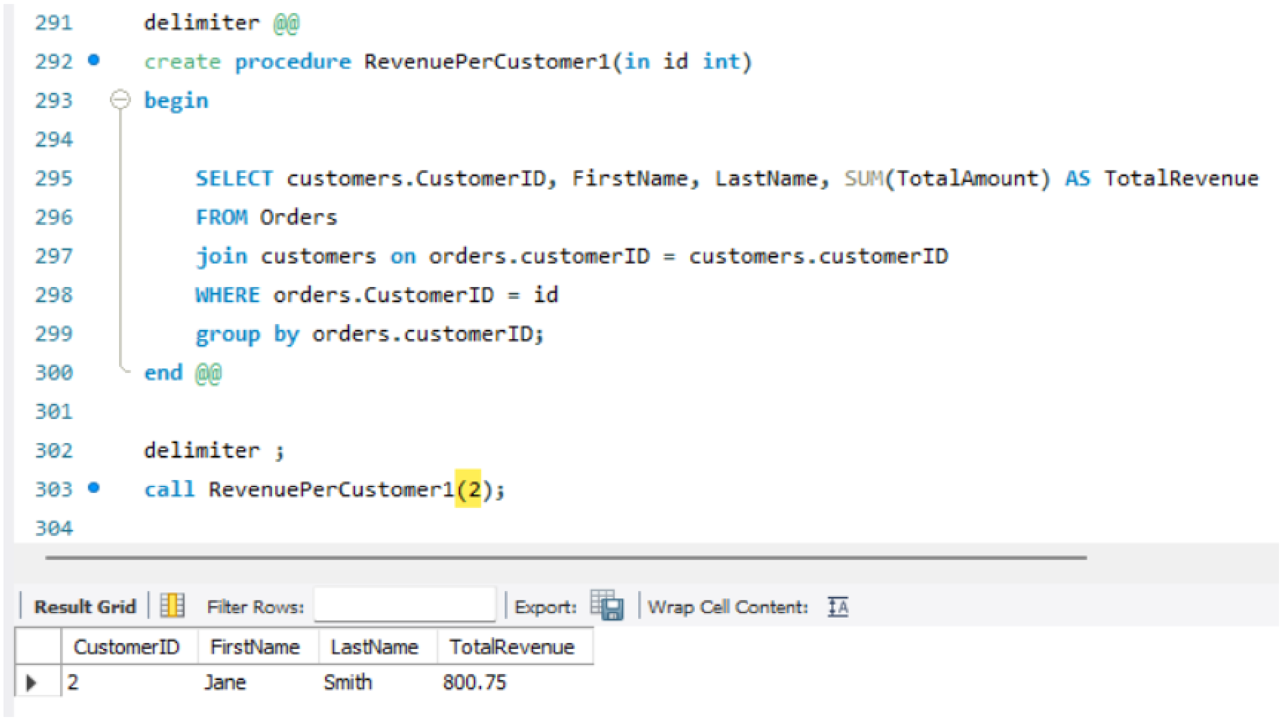
4. Write an SQL query to calculate the average quantity ordered for products in a specific category. Allow users to input the category name as a parameter.



```sql
delimiter @@
create procedure ShowAverageQuantity(IN category varchar(50))
begin

    SELECT AVG(Quantity) AS AverageQuantityOrdered
    FROM OrderDetails
    JOIN Products ON OrderDetails.ProductID = Products.ProductID
    WHERE productname = category;

end @@

delimiter ;
call ShowAverageQuantity('Laptop');
```

```
AverageQuantityOrdered
2.0000
```

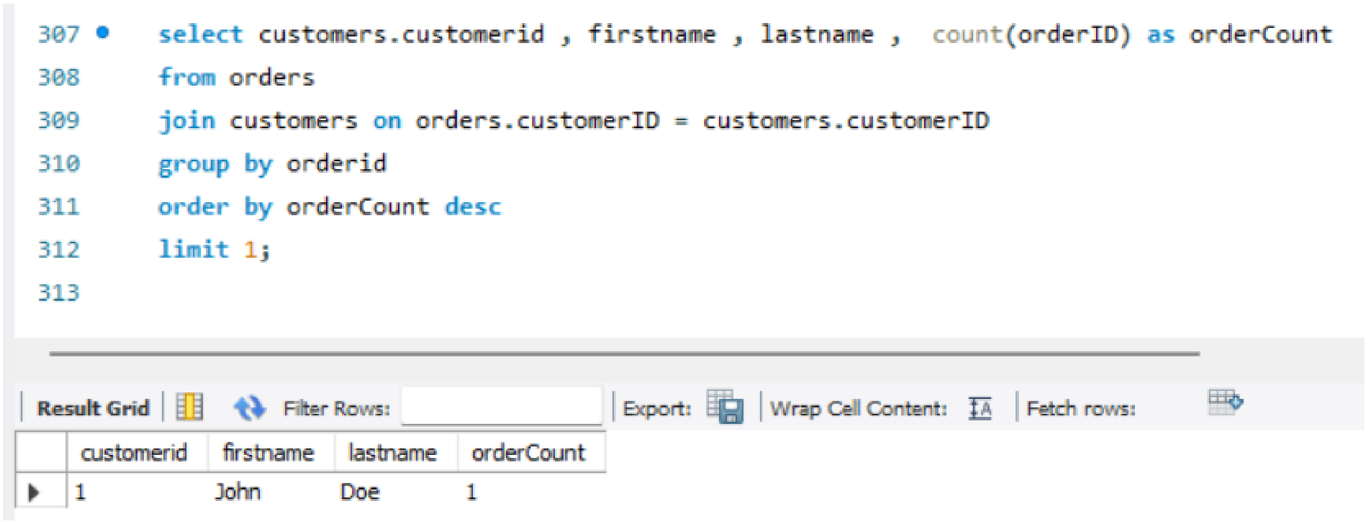
5. Write an SQL query to calculate the total revenue generated by a specific customer. Allow users to input the customer ID as a parameter.



```sql
delimiter @@
create procedure RevenuePerCustomer1(in id int)
begin

    SELECT customers.CustomerID, FirstName, LastName, SUM(TotalAmount) AS TotalRevenue
    FROM Orders
    join customers on orders.customerID = customers.customerID
    WHERE orders.CustomerID = id
    group by orders.customerID;
end @@

delimiter ;
call RevenuePerCustomer1(2);

```

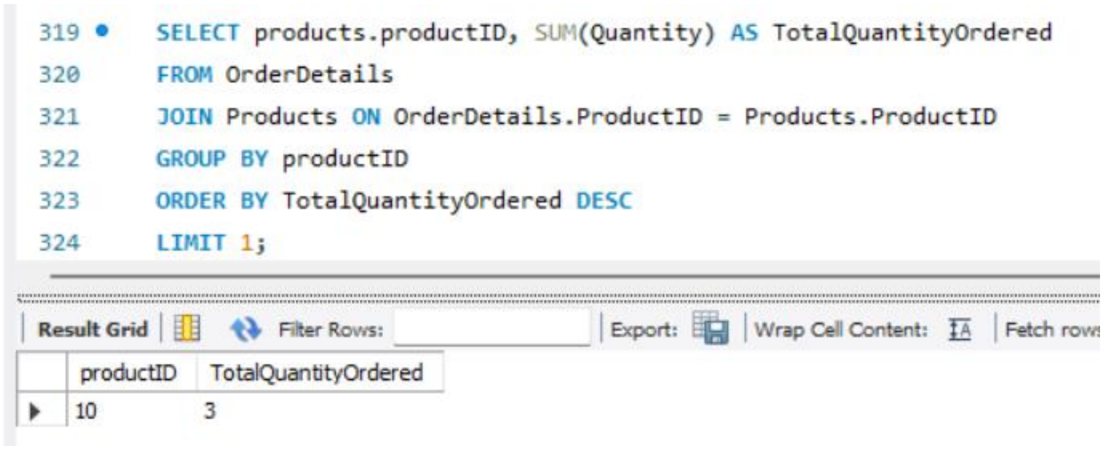
```
CustomerID  FirstName  LastName  TotalRevenue
2           Jane       Smith     800.75
```

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.


```sql
select customers.customerid , firstname , lastname ,  count(orderID) as orderCount
from orders
join customers on orders.customerID = customers.customerID
group by orderid
order by orderCount desc
limit 1;
```

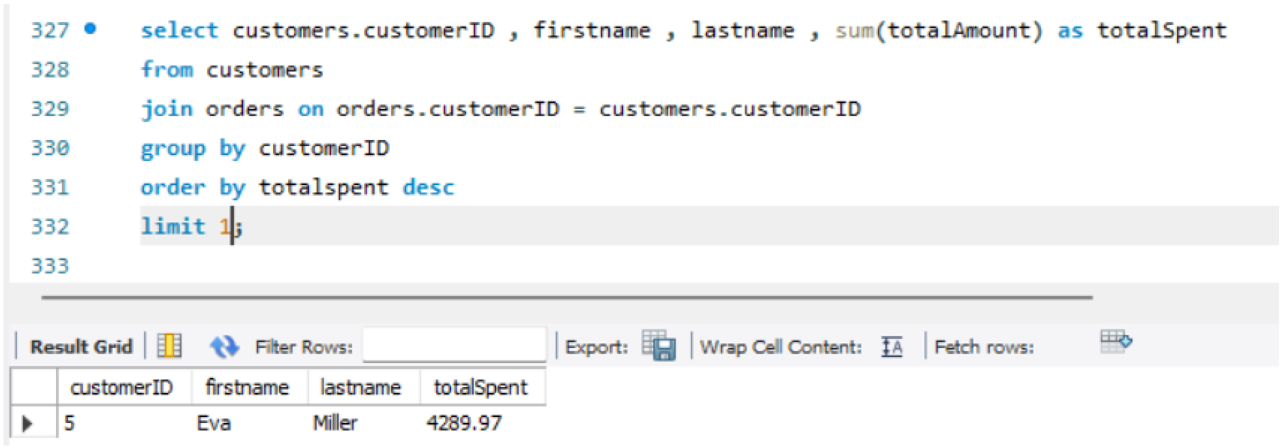
	customerid	firstname	lastname	orderCount
▶	1	John	Doe	1

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.


```sql
SELECT products.productID, SUM(Quantity) AS TotalQuantityOrdered
FROM OrderDetails
JOIN Products ON OrderDetails.ProductID = Products.ProductID
GROUP BY productID
ORDER BY TotalQuantityOrdered DESC
LIMIT 1;
```

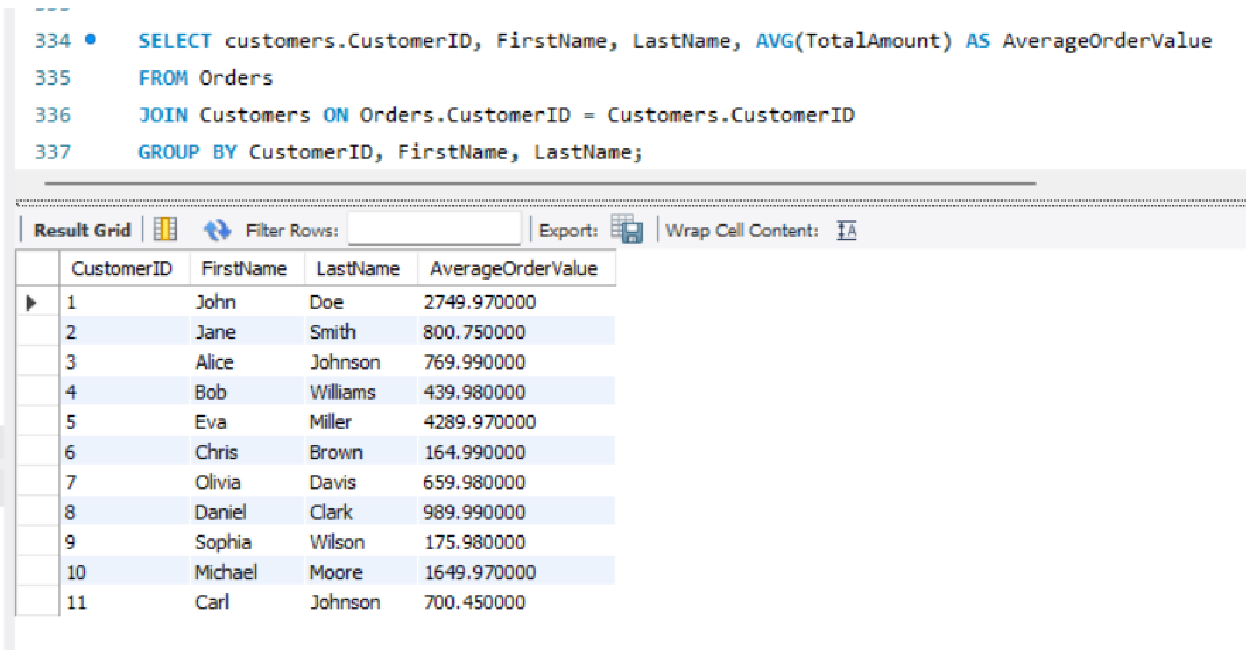
	productID	TotalQuantityOrdered
▶	10	3

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.


```sql
select customers.customerID , firstname , lastname , sum(totalAmount) as totalSpent
from customers
join orders on orders.customerID = customers.customerID
group by customerID
order by totalspent desc
limit 1;
```

	customerID	firstname	lastname	totalSpent
▶	5	Eva	Miller	4289.97

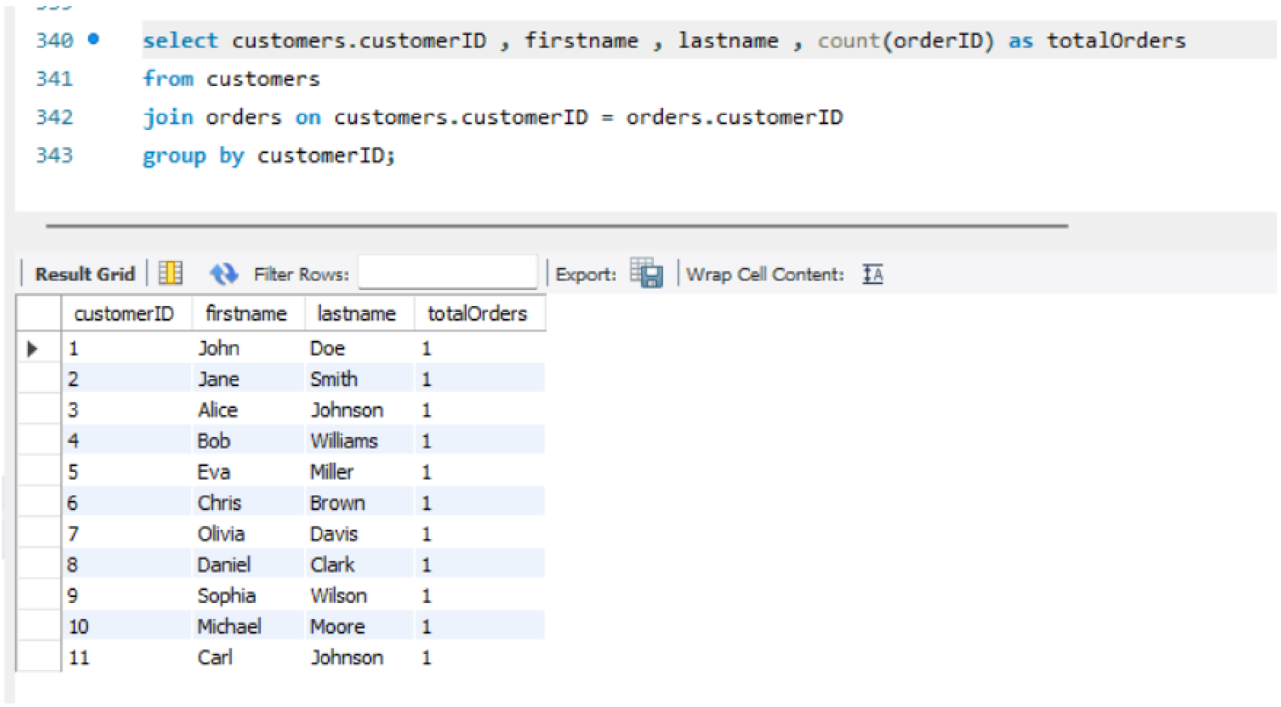
9. Write an SQL query to calculate the average order value (total revenue divided by the number of orders) for all customers.



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

CustomerID	FirstName	LastName	AverageOrderValue
1	John	Doe	2749.970000
2	Jane	Smith	800.750000
3	Alice	Johnson	769.990000
4	Bob	Williams	439.980000
5	Eva	Miller	4289.970000
6	Chris	Brown	164.990000
7	Olivia	Davis	659.980000
8	Daniel	Clark	989.990000
9	Sophia	Wilson	175.980000
10	Michael	Moore	1649.970000
11	Carl	Johnson	700.450000

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.



```sql
select customers.customerID , firstname , lastname , count(orderID) as totalOrders
from customers
join orders on customers.customerID = orders.customerID
group by customerID;
```

customerID	firstname	lastname	totalOrders
1	John	Doe	1
2	Jane	Smith	1
3	Alice	Johnson	1
4	Bob	Williams	1
5	Eva	Miller	1
6	Chris	Brown	1
7	Olivia	Davis	1
8	Daniel	Clark	1
9	Sophia	Wilson	1
10	Michael	Moore	1
11	Carl	Johnson	1