Assignment – 1

TechShop, an electronic gadgets shop

Task:1. Database Design:

1. Create the database named "TechShop".

```
create database TechShop;
```

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

```
create table Customers(CustomerID int primary key , FirstName text,
LastName text,Email varchar(30),Phone Bigint, Address varchar(50));
```

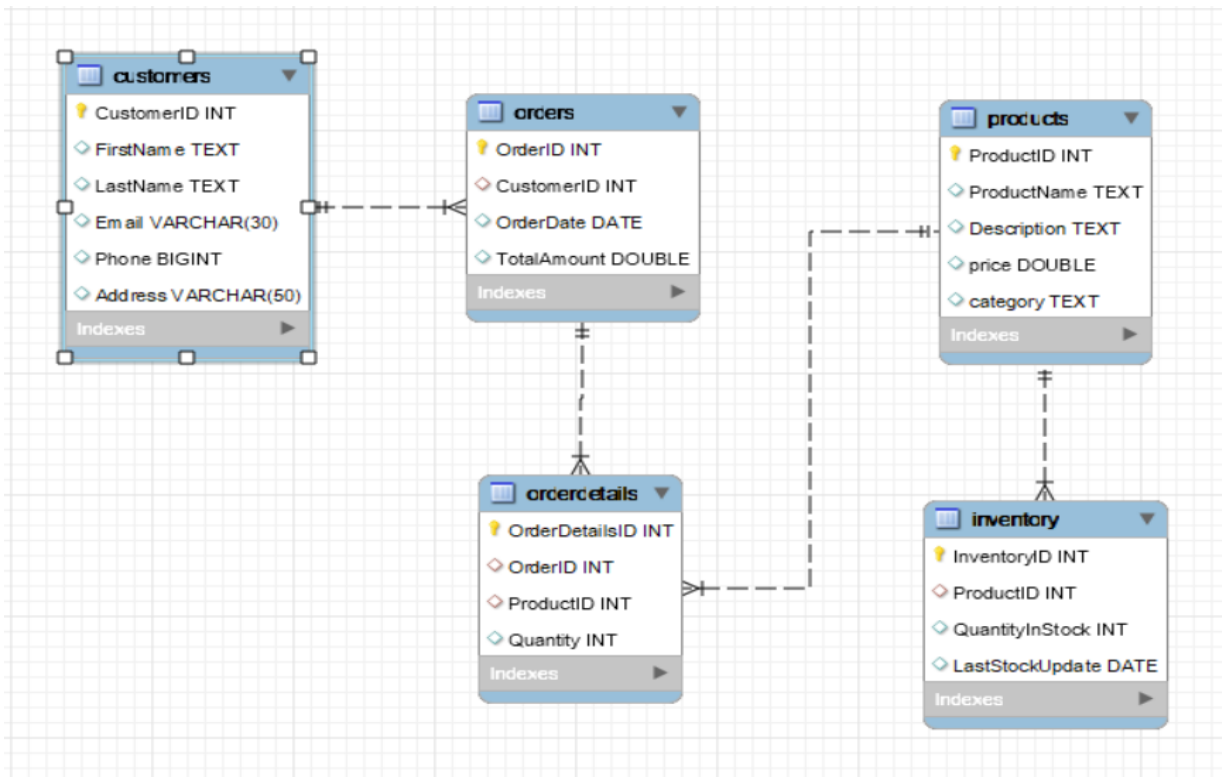
```
create table Products(ProductID int primary key, ProductName  text,
Description text, price double);
```

```
create table Orders(OrderID int primary key, CustomerID int ,
OrderDate date, TotalAmount double, foreign key(CustomerID)
 references Customers(CustomerID));
```

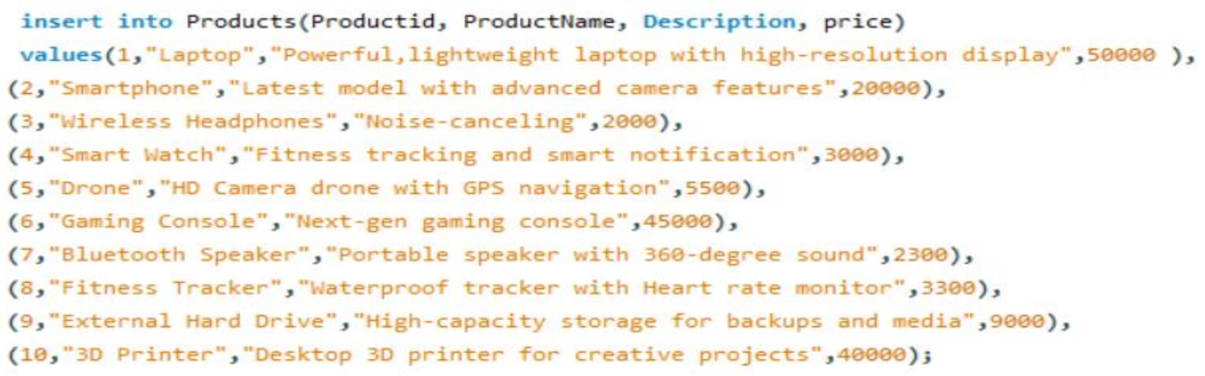
```
create table OrderDetails(OrderDetailsID int primary key,
OrderID int, ProductID int,
Quantity int, foreign key(OrderID) references
Orders(OrderID), foreign key(ProductID)
references Products(ProductID));
```

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

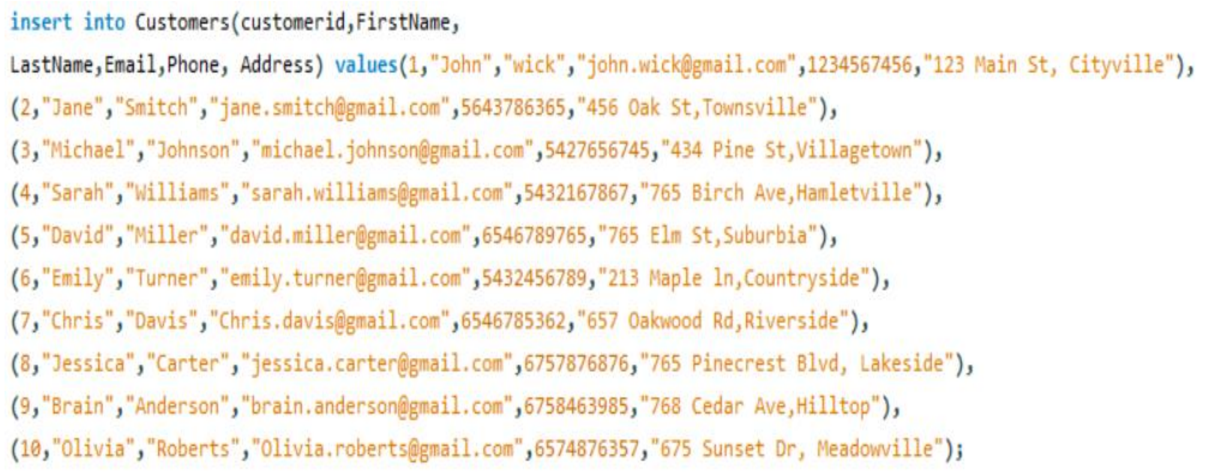
3. Create an ERD (Entity Relationship Diagram) for the database.



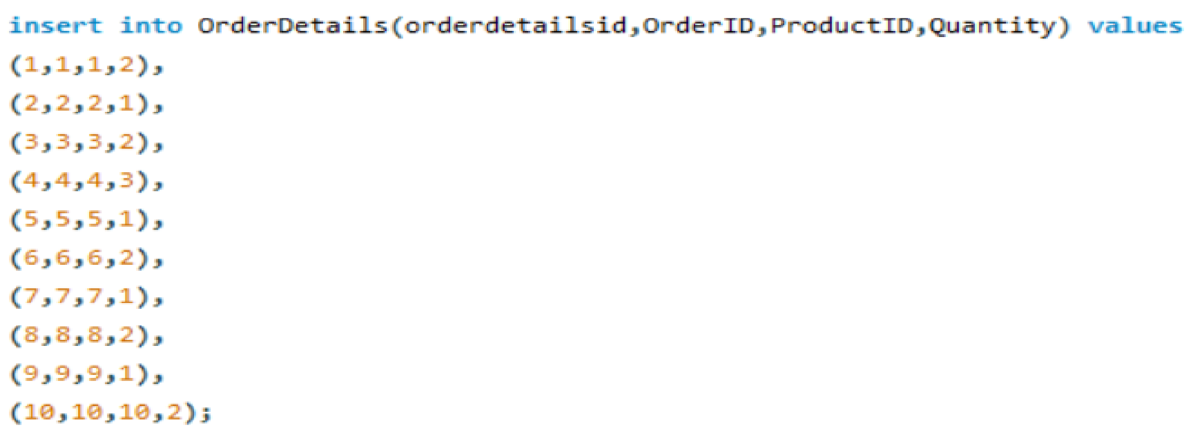
4. Insert at least 10 sample records into each of the following tables. a. Customers b. Products c. Orders d. OrderDetails e. Inventory



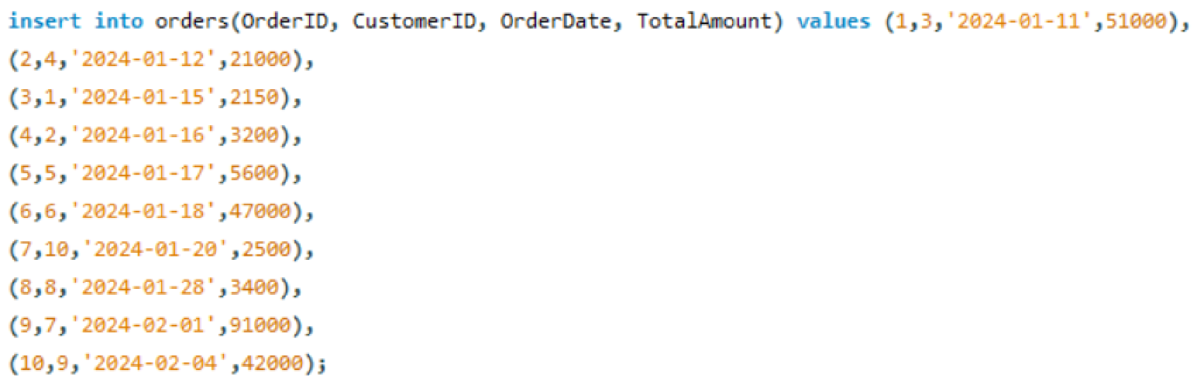
```
 insert into Products(Productid, ProductName, Description, price)
 values(1,"Laptop","Powerful,lightweight laptop with high-resolution display",50000 ),
(2,"Smartphone","Latest model with advanced camera features",20000),
(3,"Wireless Headphones","Noise-canceling",2000),
(4,"Smart Watch","Fitness tracking and smart notification",3000),
(5,"Drone","HD Camera drone with GPS navigation",5500),
(6,"Gaming Console","Next-gen gaming console",45000),
(7,"Bluetooth Speaker","Portable speaker with 360-degree sound",2300),
(8,"Fitness Tracker","Waterproof tracker with Heart rate monitor",3300),
(9,"External Hard Drive","High-capacity storage for backups and media",9000),
(10,"3D Printer","Desktop 3D printer for creative projects",40000);
```



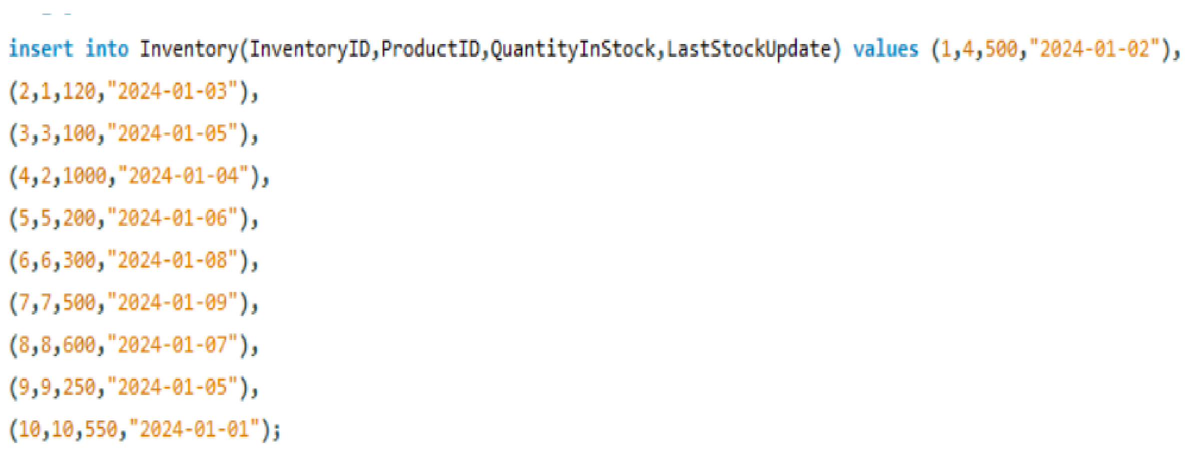
```
insert into Customers(customerid,FirstName,
LastName,Email,Phone, Address) values(1,"John","wick","john.wick@gmail.com",1234567456,"123 Main St, Cityville"),
(2,"Jane","Smitch","jane.smitch@gmail.com",5643786365,"456 Oak St,Townsville"),
(3,"Michael","Johnson","michael.johnson@gmail.com",5427656745,"434 Pine St,Villagetown"),
(4,"Sarah","Williams","sarah.williams@gmail.com",5432167867,"765 Birch Ave,Hamletville"),
(5,"David","Miller","david.miller@gmail.com",6546789765,"765 Elm St,Suburbia"),
(6,"Emily","Turner","emily.turner@gmail.com",5432456789,"213 Maple ln,Countryside"),
(7,"Chris","Davis","Chris.davis@gmail.com",6546785362,"657 Oakwood Rd,Riverside"),
(8,"Jessica","Carter","jessica.carter@gmail.com",6757876876,"765 Pinecrest Blvd, Lakeside"),
(9,"Brain","Anderson","brain.anderson@gmail.com",6758463985,"768 Cedar Ave,Hilltop"),
(10,"Olivia","Roberts","Olivia.roberts@gmail.com",6574876357,"675 Sunset Dr, Meadowville");
```



```sql
insert into OrderDetails(orderdetailsid,OrderID,ProductID,Quantity) values
(1,1,1,2),
(2,2,2,1),
(3,3,3,2),
(4,4,4,3),
(5,5,5,1),
(6,6,6,2),
(7,7,7,1),
(8,8,8,2),
(9,9,9,1),
(10,10,10,2);
```



```sql
insert into orders(OrderID, CustomerID, OrderDate, TotalAmount) values (1,3,'2024-01-11',51000),
(2,4,'2024-01-12',21000),
(3,1,'2024-01-15',2150),
(4,2,'2024-01-16',3200),
(5,5,'2024-01-17',5600),
(6,6,'2024-01-18',47000),
(7,10,'2024-01-20',2500),
(8,8,'2024-01-28',3400),
(9,7,'2024-02-01',91000),
(10,9,'2024-02-04',42000);
```



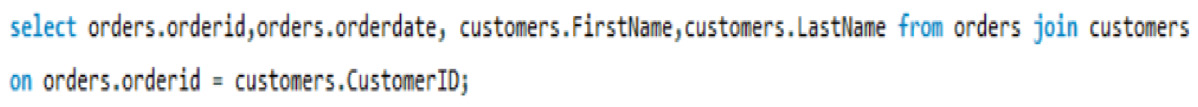
```sql
insert into Inventory(InventoryID,ProductID,QuantityInStock,LastStockUpdate) values (1,4,500,"2024-01-02"),
(2,1,120,"2024-01-03"),
(3,3,100,"2024-01-05"),
(4,2,1000,"2024-01-04"),
(5,5,200,"2024-01-06"),
(6,6,300,"2024-01-08"),
(7,7,500,"2024-01-09"),
(8,8,600,"2024-01-07"),
(9,9,250,"2024-01-05"),
(10,10,550,"2024-01-01");
```

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

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

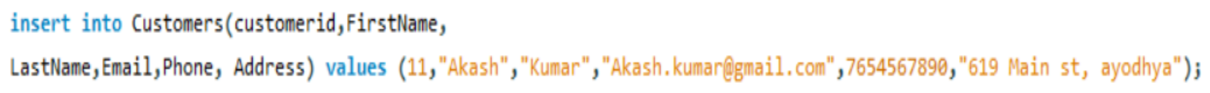
```sql
select FirstName,LastName,Email from Customers;
```

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



```sql
select orders.orderid,orders.orderdate, customers.FirstName,customers.LastName from orders join customers
on orders.orderid = customers.CustomerID;
```

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

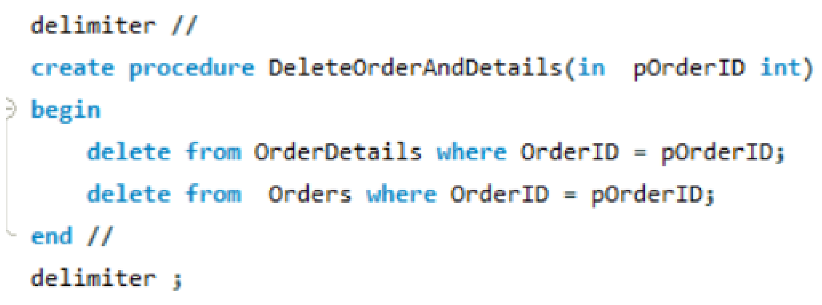


```sql
insert into Customers(customerid,FirstName,
LastName,Email,Phone, Address) values (11,"Akash","Kumar","Akash.kumar@gmail.com",7654567890,"619 Main st, ayodhya");
```

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

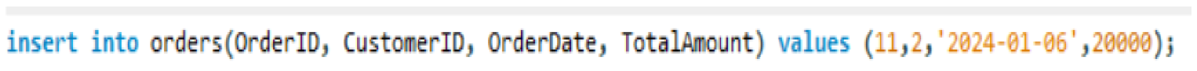
```sql
update products set price = price+price*0.1 where productid =1;
update products set price = price+price*0.1 where productid =2;
update products set price = price+price*0.1 where productid =3;
update products set price = price+price*0.1 where productid =4;
update products set price = price+price*0.1 where productid =5;
update products set price = price+price*0.1 where productid =6;
update products set price = price+price*0.1 where productid =7;
update products set price = price+price*0.1 where productid =8;
update products set price = price+price*0.1 where productid =9;
update products set price = price+price*0.1 where productid =10;
```

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.



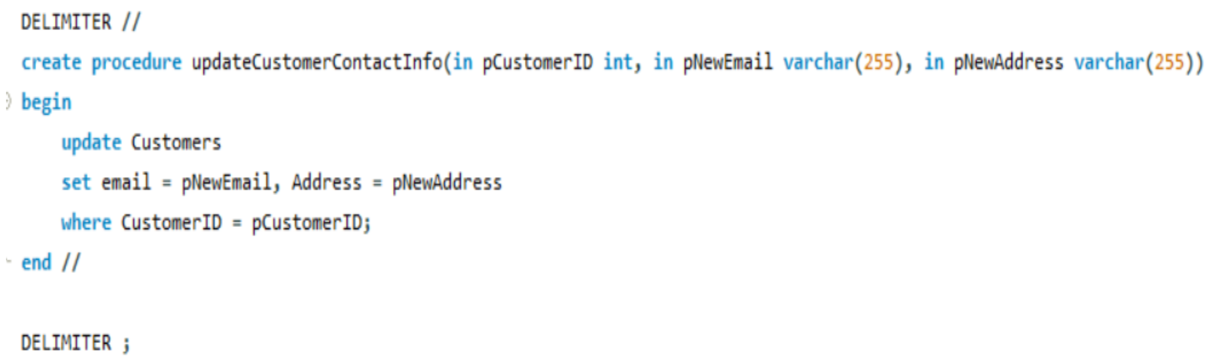
```sql
delimiter //
create procedure DeleteOrderAndDetails(in  pOrderID int)
begin
    delete from OrderDetails where OrderID = pOrderID;
    delete from  Orders where OrderID = pOrderID;
end //
delimiter ;
```

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.



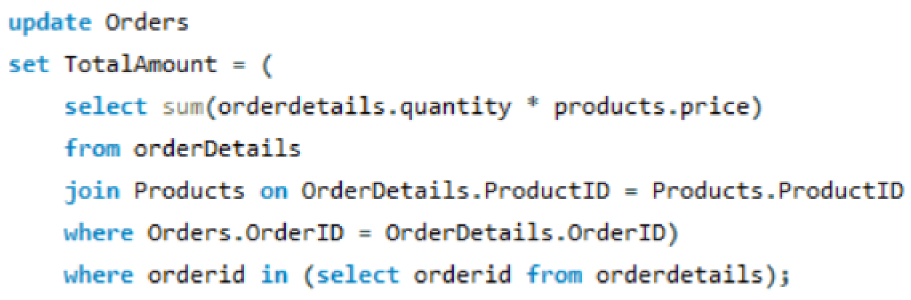
```sql
insert into orders(OrderID, CustomerID, OrderDate, TotalAmount) values (11,2,'2024-01-06',20000);
```

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.



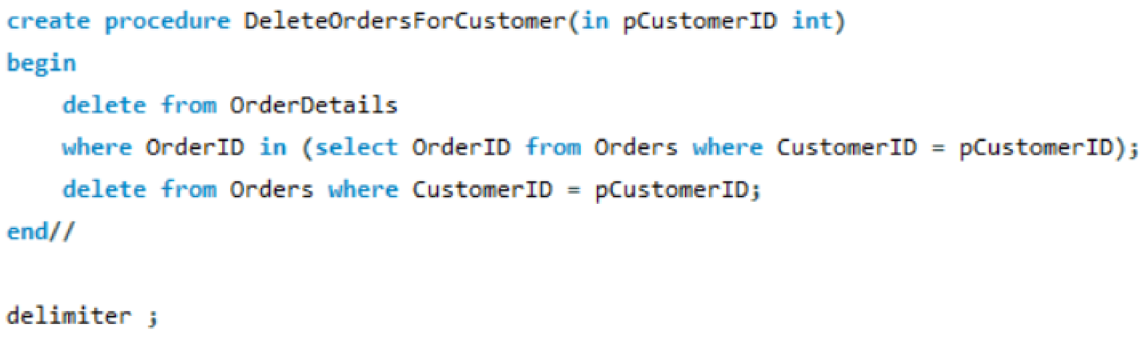
```sql
DELIMITER //
create procedure updateCustomerContactInfo(in pCustomerID int, in pNewEmail varchar(255), in pNewAddress varchar(255))
begin
    update Customers
    set email = pNewEmail, Address = pNewAddress
    where CustomerID = pCustomerID;
end //

DELIMITER ;
```

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.



```sql
update Orders
set TotalAmount = (
    select sum(orderdetails.quantity * products.price)
    from orderDetails
    join Products on OrderDetails.ProductID = Products.ProductID
    where Orders.OrderID = OrderDetails.OrderID)
    where orderid in (select orderid from orderdetails);
```

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.



```sql
create procedure DeleteOrdersForCustomer(in pCustomerID int)
begin
    delete from OrderDetails
    where OrderID in (select OrderID from Orders where CustomerID = pCustomerID);
    delete from Orders where CustomerID = pCustomerID;
end//

delimiter ;
```

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(productid,productname,description,price,category) value
(11,"earbuds","light-weight and noise-cancelling",750,"electronic gadget");
```

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.

```
delimiter //
create procedure updateOrderStatus(in pOrderID int, in pNewStatus varchar(50))
begin
    update Orders
    set status = pNewStatus
    where OrderID = pOrderID;
end //
delimiter ;
```

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.

```
select count(orders.orderid) as numberOforders from orders join customers
on orders.customerid = customers.customerid
group by customers.customerid;
```

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

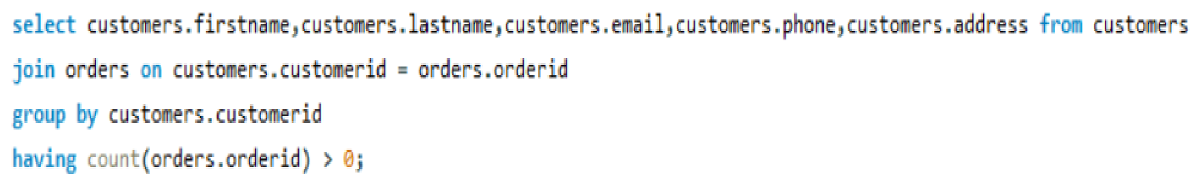
1. Write an SQL query to retrieve a list of all orders along with customer information (e.g., customer name) for each order.

```
select orders.orderid,orders.orderdate,orders.totalamount,customers.firstname,customers.lastname from orders
join customers on orders.orderid = customers.customerid;
```

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

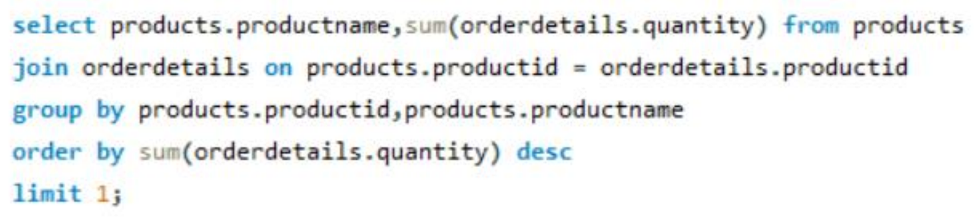
```
select products.productname,sum(price * orderdetails.quantity) as TotalRevenue from products
join orderdetails on products.productid = orderdetails.orderdetailsid
group by products.productid,products.productname;
```

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



```
select customers.firstname,customers.lastname,customers.email,customers.phone,customers.address from customers
join orders on customers.customerid = orders.orderid
group by customers.customerid
having count(orders.orderid) > 0;
```

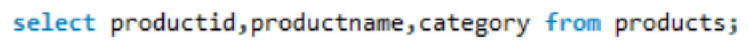
4. Write an SQL query to find the most popular electronic gadget, which is the one with the highest total quantity ordered. Include the product name and the total quantity ordered.



```
select products.productname,sum(orderdetails.quantity) from products
join orderdetails on products.productid = orderdetails.productid
group by products.productid,products.productname
order by sum(orderdetails.quantity) desc
limit 1;
```

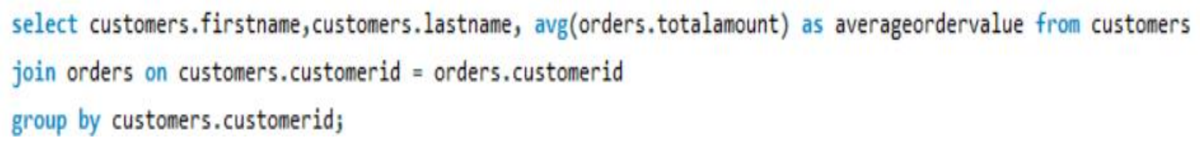
5. Write an SQL query to retrieve a list of electronic gadgets along with their corresponding categories.

Note – I added the category column in products tables that is why I didn't use join here



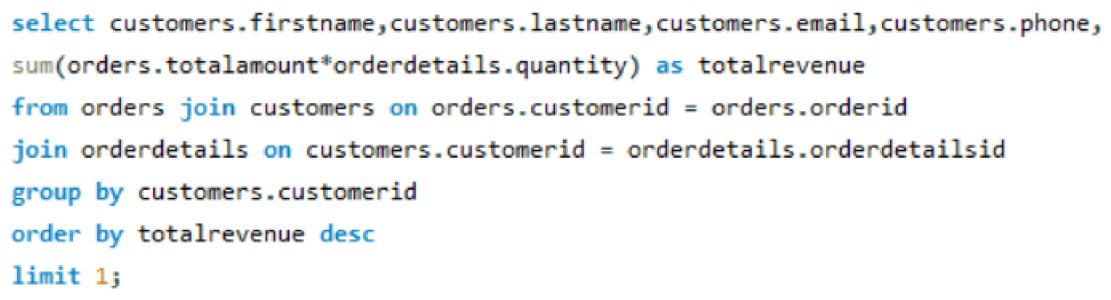
```
select productid,productname,category from products;
```

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



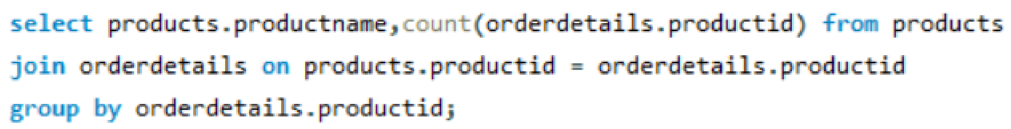
```
select customers.firstname,customers.lastname, avg(orders.totalamount) as averageordervalue from customers
join orders on customers.customerid = orders.customerid
group by customers.customerid;
```

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



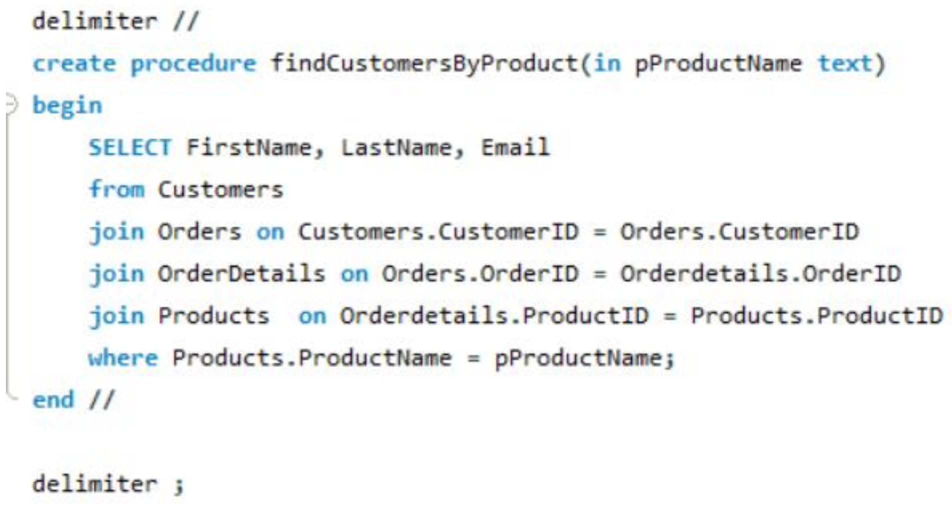
```
select customers.firstname,customers.lastname,customers.email,customers.phone,
sum(orders.totalamount*orderdetails.quantity) as totalrevenue
from orders join customers on orders.customerid = orders.orderid
join orderdetails on customers.customerid = orderdetails.orderdetailsid
group by customers.customerid
order by totalrevenue desc
limit 1;
```

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



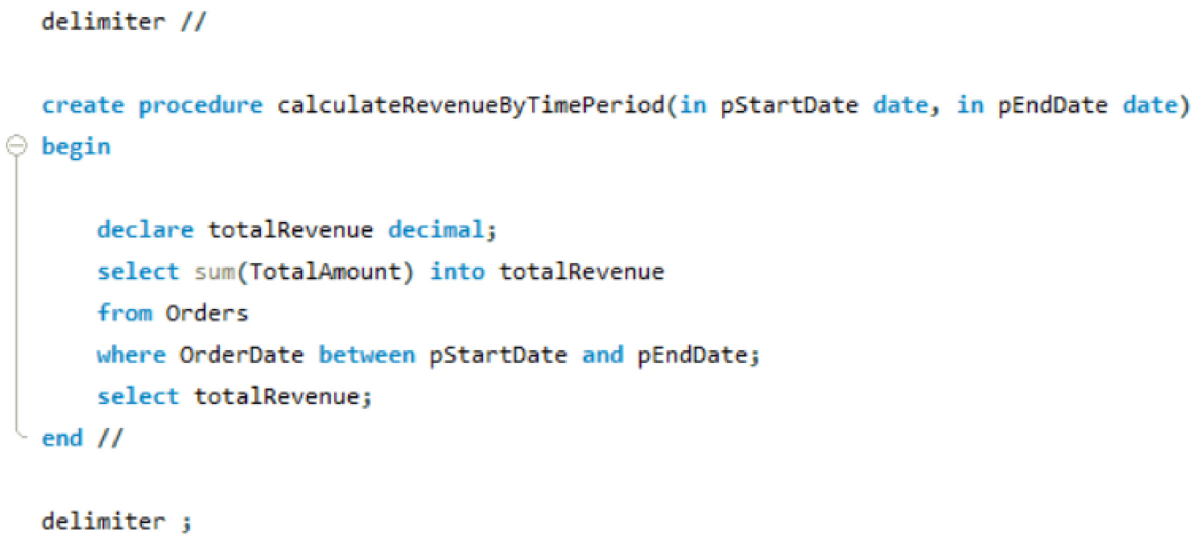
```sql
select products.productname,count(orderdetails.productid) from products
join orderdetails on products.productid = orderdetails.productid
group by orderdetails.productid;
```

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.



```sql
delimiter //
create procedure findCustomersByProduct(in pProductName text)
begin
    SELECT FirstName, LastName, 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 = pProductName;
end //

delimiter ;
```

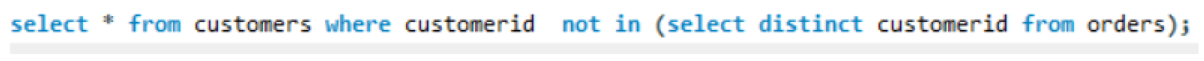
10. Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period. Allow users to input the start and end dates as parameters.



```sql
delimiter //

create procedure calculateRevenueByTimePeriod(in pStartDate date, in pEndDate date)
begin

    declare totalRevenue decimal;
    select sum(TotalAmount) into totalRevenue
    from Orders
    where OrderDate between pStartDate and pEndDate;
    select totalRevenue;
end //

delimiter ;
```

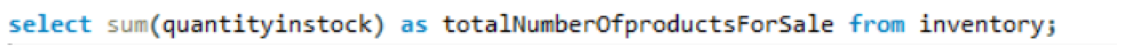
Task 4. Subquery and its type:

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



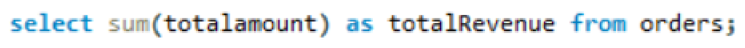
```
select * from customers where customerid  not in (select distinct customerid from orders);
```

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



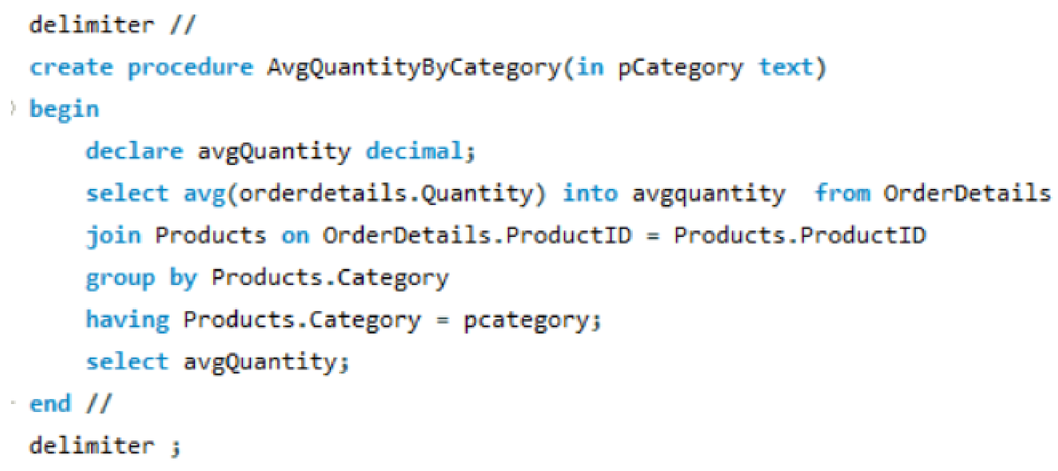
```
select sum(quantityinstock) as totalNumberOfproductsForSale from inventory;
```

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



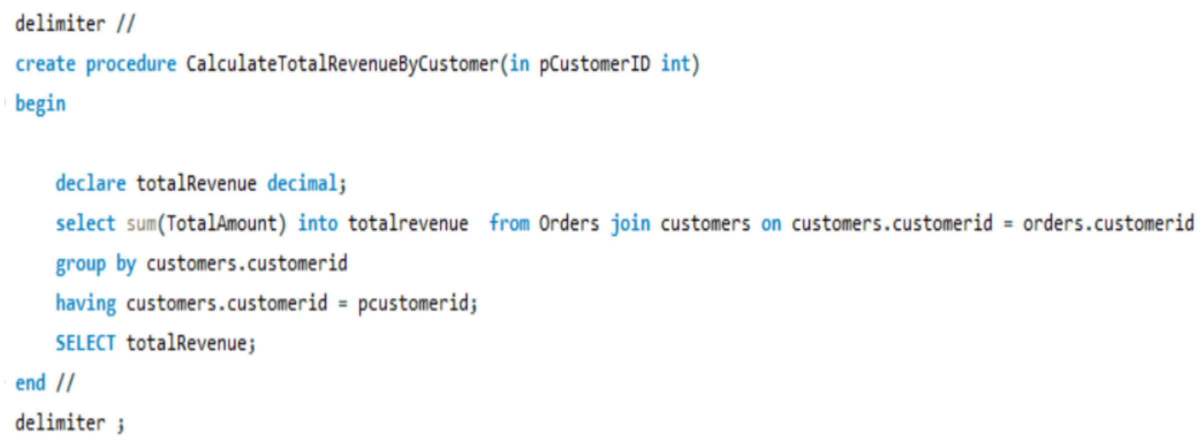
```
select sum(totalamount) as totalRevenue from orders;
```

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.



```
delimiter //
create procedure AvgQuantityByCategory(in pCategory text)
begin
    declare avgQuantity decimal;
    select avg(orderdetails.Quantity) into avgquantity  from OrderDetails
    join Products on OrderDetails.ProductID = Products.ProductID
    group by Products.Category
    having Products.Category = pcategory;
    select avgQuantity;
end //
delimiter ;
```

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.



```
delimiter //
create procedure CalculateTotalRevenueByCustomer(in pCustomerID int)
begin

    declare totalRevenue decimal;
    select sum(TotalAmount) into totalrevenue  from Orders join customers on customers.customerid = orders.customerid
    group by customers.customerid
    having customers.customerid = pcustomerid;
    SELECT totalRevenue;
end //
delimiter ;
```

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.firstname,customers.lastname,count(orders.customerid) from customers
join orders on customers.customerid = orders.customerid
group by customers.customerid
order by count(orders.customerid) desc
limit 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.category,sum(orderdetails.quantity) from products
join orderdetails on products.productid = orderdetails.productid
group by products.category
order by sum(orderdetails.quantity) desc
limit 1;
```

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.FirstName, Customers.lastname, SUM(orders.TotalAmount) AS TotalSpending
from Orders
join Customers ON Orders.CustomerID = Customers.CustomerID
group by Customers.CustomerID
order by TotalSpending desc
limit 1;
```

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

```sql
select customers.firstname,customers.lastname, avg(orders.totalamount) from customers
join orders on customers.customerid = orders.customerid
group by customers.customerid;
```

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.firstname,customers.lastname,count(orders.customerid) from customers
left join orders on customers.customerid = orders.customerid
group by customers.customerid;
```