Task 1

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.

use Techshop;

Create Table Customers(

CustomerID int Primary Key auto\_increment,

FirstName varchar(255) Not Null,

LastName varchar(15) Not Null,

Email varchar(30) Not Null,

Phone varchar(10) Default Null,

Address varchar(100) Default Null

);

Create Table Products(

ProductID int Primary Key auto\_increment,

ProductName varchar(50) Not Null,

Descriptions varchar(200) Not Null,

Price float default 0

);

Create Table Orders(

OrderId int Primary Key auto\_increment,

CustomerID int Not Null,

OrderDate Date Not Null,

TotalAmount int Default 0,

foreign key(CustomerID) references Customers(CustomerID)

);

Create Table OrderDeatails(

OrderDeatailId int Primary key auto\_increment,

orderId int Not Null,

ProductId int Not Null,

Quantity int Not Null,

foreign key(OrderId) references Orders(OrderId),

foreign key(ProductId) references Products(ProductId)

);

Create Table Inventory(

InventoryId int Primary Key auto\_increment,

ProductId int Not Null,

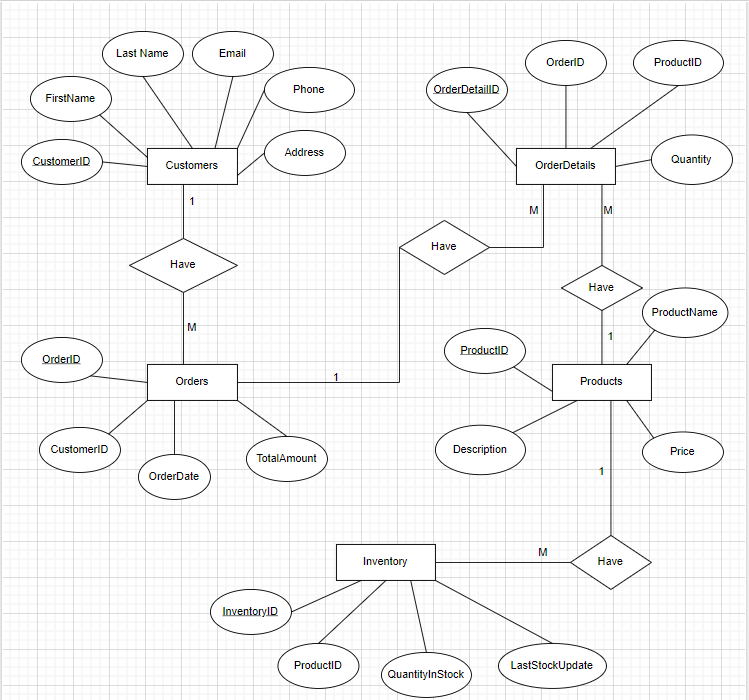
QuantityInStock int Not Null,

LastStockUpdate date Not Null,

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 Key and Foreign Key constraints**

**Customers**

* Primary Keys: CustomerID
* Foreign Keys: None

**Products**

* Primary Keys: ProductID
* Foreign Keys:None

**Orders**

* Primary Keys: OrderID
* Foreign Keys: CustomerID (Referencing Customers)

**OrderDeatails**

* Primary Keys: OrderDeatailID
* Foreign Keys: OrderID (Referencing Orders), ProductID (Referencing Products)

**Inventory**

* Primary Keys: InventoryID
* Foreign Keys ProductID (Referencing Products)

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

a. Customers

b. Products

c. Orders

d. OrderDetails

e. Inventory

Insert Into Customers Values

(1, 'Danish' , 'Mishra' , 'danishmishra12@gmail.com' , '1234567890' , 'Flat no10,abc,state'),

(2, 'Sarika' , 'Mishra' , 'sarikamishra@gmail.com' , '2345678901' , 'Flat no10,abc,state'),

(3, 'Harshit' , 'Sachan' , 'harshitsachan@gmail.com' , '3456789012' , 'Flat no10,abc,state'),

(4, 'Niharika' , 'Mishra' , 'niharikamishra@gmail.com' , '4567890123' , 'Flat no10,abc,state'),

(5, 'Madhuri' , 'Mishra' , 'madhurimishra@gmail.com' , '5678901234' , 'Flat no10,abc,state'),

(6, 'Ayush' , 'Aggarwal' , 'ayushaggarwal@gmail.com' , '6789012345' , 'Flat no10,abc,state'),

(7, 'Shikhar' , 'Bhaduria' , 'shikharbhaduria@gmail.com' , '7890123456' , 'Flat no10,abc,state'),

(8, 'Anushka' , 'Kapoor' , 'anushkakapoor@gmail.com' , '8901234567' , 'Flat no10,abc,state'),

(9, 'Vikas' , 'Mishra' , 'vikasmishra@gmail.com' , '9012345678' , 'Flat no10,abc,state'),

(10, 'Pragati' , 'Gaur' , 'pragatigaur@gmail.com', '0123456789' , 'Flat no10,abc,state');

Insert Into Products Values

(101,'Watch','Analog Watch',1000),

(102,'Smart watch','Pick Call',2000),

(103,'Earphone','Listening',1200),

(104,'TV','See Movies',3000),

(105,'Smart Phone','Play Games',4000),

(106,'Laptop','Portable',5000),

(107,'Dekstop','Use Internet',5600),

(108,'Speaker','songs',6000),

(109,'Wifi Router','Wifi',1700),

(200,'Blower','Hot Air',8000);

Insert Into Orders Values

(301,1,'2023-10-02',20000),

(302,2,'2023-10-03',30000),

(303,3,'2023-10-04',50000),

(304,4,'2023-10-05',40000),

(305,5,'2023-10-06',70000),

(306,6,'2023-10-07',80000),

(307,7,'2023-10-08',90000),

(308,8,'2023-10-09',24000),

(309,9,'2023-10-10',27000),

(400,10,'2023-10-11',85000);

Insert Into OrderDeatails Values

(501,301,101,1),

(502,302,102,4),

(503,303,103,1),

(504,304,104,1),

(505,305,105,6),

(506,306,106,7),

(507,307,107,1),

(508,308,108,5),

(509,309,109,1),

(600,400,200,2);

Insert Into Inventory Values

(701,101,12,'2023-09-08'),

(702,102,10,'2023-09-09'),

(703,103,17,'2023-09-10'),

(704,104,18,'2023-09-11'),

(705,105,20,'2023-09-12'),

(706,106,19,'2023-09-13'),

(707,107,20,'2023-09-14'),

(708,108,29,'2023-09-15'),

(709,109,40,'2023-09-16'),

(800,200,30,'2023-09-17');

Task 2

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

SELECT

concat(FirstName,' ',LastName) AS Name, Email FROM Customers;

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

SELECT

o.OrderID, concat(c.FirstName,' ',c.LastName) AS Name, o.OrderDate

FROM

Orders o, Customers c

WHERE

o.customerId=c.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.

INSERT INTO Customers(FirstName,LastName,Email,Address) VALUES('dev','gupta','devgupta@gmail.com','Flat no20,abkc,state');

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.

SET @OrderIdtoDel='301'

DELETE FROM OrderDeatails

WHERE

OrderId =@OrderIdtoDel;

DELETE FROM Orders

WHERE

orderId=@orderIdtodel;

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(11,'2023-10-12',89000);

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.

SET @CustomerID := 10

SET @Email := 'pgaur@gmail.com'

SET @Address := 'flat 47, Dehradun'

UPDATE Customers

SET Email = @Email,

Address = @Address

WHERE CustomerID = @CustomerID;

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(Quantity \* Price)

FROM OrderDeatails

JOIN Products ON OrderDeatails.ProductID = Products.ProductID

WHERE OrderDeatails.OrderID = Orders.OrderID

)

WHERE EXISTS (

SELECT 1

FROM OrderDeatails

WHERE OrderDeatails.OrderID = Orders.OrderID

);

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.

SET @CustomerIDToDelete :='10';

DELETE FROM OrderDeatails

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

DELETE FROM Orders

WHERE CustomerID = @CustomerIDToDelete;

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, Descriptions, Price)

VALUES ('Electronic Gadget', 'Description of the gadget', 99.99);

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) DEFAULT NULL AFTER TotalAmount;

SET @orderidtochange=208;

SET @newstatus='Shipped';

UPDATE orders SET status = @newstatus WHERE orderid=@orderidtochange;

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

);

Task 3

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,concat(FirstName,' ',LastName) AS name , c.email, c.phone

FROM orders o

JOIN customers c ON o.customerid = c.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 P.ProductName, SUM(OD.Quantity \* O.TotalAmount) AS TotalRevenue

FROM Products P

JOIN orderdeatails OD ON P.ProductID = OD.ProductID

JOIN Orders O ON OD.OrderID = O.OrderID

WHERE P.ProductName = 'Portable'

GROUP BY P.ProductName;

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

SELECT concat(c.FirstName,' ',c.LastName) as Name, c.email, c.phone

FROM customers c

JOIN orders o ON c.customerid = o.customerid

GROUP BY c.customerid;

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 Products P

JOIN orderdeatails OD ON P.ProductID = OD.ProductID

GROUP BY P.ProductName

ORDER BY TotalQuantityOrdered DESC

LIMIT 1;

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

Alter Table Products ADD Category varchar(45) Default Null

Update Products SET Category= 'ElectronicGadgets';

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

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

GROUP BY C.CustomerID, C.FirstName, C.LastName;

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,O.TotalAmount AS TotalRevenue

FROM Orders O

JOIN Customers C ON O.CustomerID = C.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.

SELECT P.ProductName, COUNT(OD.OrderDeatailId) AS OrderCount

FROM Products P

JOIN orderdeatails OD ON P.ProductID = OD.ProductID

GROUP BY P.ProductName;

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.

SET @ProductNameParam = 'Electronic Gadget';

SELECT C.CustomerID,C.FirstName,C.LastName

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

JOIN orderdeatails OD ON O.OrderID = OD.OrderID

JOIN Products P ON OD.ProductID = P.ProductID

WHERE P.ProductName = @ProductNameParam;

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.

SET @StartDateParam = '2023-01-01';

SET @EndDateParam = '2023-12-31';

SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders

WHERE OrderDate BETWEEN @StartDateParam AND @EndDateParam;

Task 4

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

SELECT CONCAT(firstname,' ',lastname) as Name, c.email, c.phone

FROM customers c

WHERE customerid NOT IN (SELECT customerid FROM orders);

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

SELECT COUNT(\*) AS totalproducts

FROM products;

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

SELECT SUM(o.totalamount) AS totalrevenue

FROM orders o;

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.

SET @categoryName = 'Electronic Gadget';

SELECT AVG(quantity) AS AverageQuantityOrdered

FROM OrderDeatails

WHERE productid IN (SELECT productid FROM products WHERE category = @categoryName);

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.

SET @CustomerIDParam = 9;

SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders

WHERE CustomerID = @CustomerIDParam;

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 customerid, CONCAT(firstname, ' ', lastname) AS Name, (

SELECT COUNT(\*) FROM orders o WHERE o.customerid = c.customerid

) AS NumberOfOrders FROM customers c

ORDER BY NumberOfOrders DESC;

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.category,(SELECT SUM(quantity) FROM orderdeatails d

WHERE p.productid = d.productid) AS TotalQuantityOrdered

FROM products p

ORDER BY TotalQuantityOrdered 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.

SELECT c.customerid, CONCAT(c.firstname, ' ', c.lastname) AS Name,

( SELECT SUM(d.quantity \* p.price) FROM products p, Orderdeatails d, orders o

WHERE o.customerid = c.customerid

AND o.orderid = d.orderid

AND d.productid IN (

SELECT productid FROM products WHERE category = 'Electronic Gadget' )

) AS TotalSpending FROM customers c

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.

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

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

GROUP BY C.CustomerID, C.FirstName, C.LastName;

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.CustomerID,C.FirstName,C.LastName,

(

SELECT COUNT(\*)

FROM Orders O

WHERE O.CustomerID = C.CustomerID

) AS OrderCount

FROM Customers C;