SQL ASSIGNMENT

NAME: LAKSHMI PRIYA S

TITLE: ELECTRONIC GADGETS

Task:1. Database Design:

1.Create the database named "TechShop"

```
CREATE DATABASE TechShop;
```

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

```
 USE TechShop;
CREATE TABLE Customers (
CustomerID INT PRIMARY KEY AUTO_INCREMENT,
 FirstName VARCHAR(50) NOT NULL,
LastName VARCHAR(50) NOT NULL,
Email VARCHAR(100) NOT NULL,
Phone VARCHAR(15) NOT NULL,
Address VARCHAR(30) NOT NULL
 );

CREATE TABLE Products (
 ProductID INT PRIMARY KEY AUTO_INCREMENT,
 ProductName VARCHAR(100) NOT NULL,
 Descriptions TEXT,
 Price INT NOT NULL);

CREATE TABLE Orders (
 OrderID INT PRIMARY KEY AUTO_INCREMENT,
 CustomerID INT NOT NULL,
 OrderDate DATE NOT NULL,
 TotalAmount INT NOT NULL,
 FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
 );

CREATE TABLE OrderDetails (
 OrderDetailID 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)
 );
```

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 (FirstName, LastName, Email, Phone, Address) VALUES
 ('John', 'Doe', 'john.doe@example.com', '1234567890', '123 Main St'),
 ('Alice', 'Smith', 'alice.smith@example.com', '9876543210', '456 Oak St'),
 ('Bob', 'Johnson', 'bob.j@example.com', '5556667777', '789 Pine St'),
 ('Clara', 'Brown', 'clara.b@example.com', '4445556666', '147 Birch St'),
 ('David', 'White', 'david.w@example.com', '3332221111', '369 Cedar St'),
 ('Emma', 'Clark', 'emma.c@example.com', '1112223333', '258 Spruce St'),
 ('Frank', 'Adams', 'frank.a@example.com', '6667778888', '753 Maple St'),
 ('Grace', 'Baker', 'grace.b@example.com', '9998887777', '951 Elm St'),
 ('Henry', 'Miller', 'henry.m@example.com', '7778889999', '852 Walnut St'),
 ('Ivy', 'Williams', 'ivy.w@example.com', '1239874560', '654 Willow St');

INSERT INTO Products (ProductName, DescriptionS, Price) VALUES
 ('Laptop', 'High performance laptop', 12000),
 ('Smartphone', 'Latest model smartphone', 8000),
 ('Tablet', '10-inch screen tablet', 4500),
 ('Smartwatch', 'Fitness and health tracking', 2000),
 ('Gaming Console', 'Latest-gen gaming console', 5000),
 ('Wireless Headphones', 'Noise-canceling headphones', 1500),
('External Hard Drive', '1TB storage', 1000),
 ('Keyboard', 'Mechanical gaming keyboard', 7500),
 ('Mouse', 'Ergonomic wireless mouse', 500),
 ('Monitor', '27-inch 4K UHD display', 30000);

INSERT INTO Orders (CustomerID, OrderDate, TotalAmount) VALUES
 (1, '2024-03-01', 14000),
 (2, '2024-03-05', 95000),
 (3, '2024-03-10', 3000),
 (4, '2024-03-15', 6500),
 (5, '2024-03-18', 8500),
 (6, '2024-03-21', 12500),
 (7, '2024-03-25', 5000),
 (8, '2024-03-28', 17500),
 (9, '2024-03-30', 7000),
 (10, '2024-04-01', 20000);

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

INSERT INTO Inventory (ProductID, QuantityInStock, LastStockUpdate)
 VALUES
  (1, 10, '2024-03-01'),
  (2, 15, '2024-03-02'),
  (3, 20, '2024-03-03'),
  (4, 12, '2024-03-04'),
  (5, 8, '2024-03-05'),
  (6, 30, '2024-03-06'),
  (7, 18, '2024-03-07'),
  (8, 25, '2024-03-08'),
  (9, 22, '2024-03-09'),
  (10, 10, '2024-03-10');
```

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;
   ```

2. Write an SQL query to list all orders with their order dates and corresponding customer names.
   ```
   SELECT O.OrderID, O.OrderDate, C.FirstName, C.LastName
    FROM Orders O
    JOIN Customers C ON 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, Phone, Address)
   VALUES
   ('Michael', 'Jordan', 'michael.jordan@example.com', '9997776665',
    '23 Basketball    St');
   ```

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

```
update Products set Price = Price * 1.10 ;
```

5. Write an SQL query to delete a specific order and its associated order details from the "Orders" and "OrderDetails" tables.

```
DELETE FROM OrderDetails WHERE OrderID = 5;
DELETE FROM Orders WHERE OrderID = 5;
```

6. Write an SQL query to insert a new order into the "Orders" table.
Include the customer ID, order date, and any other necessary information.

```
 INSERT INTO Orders (CustomerID, OrderDate, TotalAmount)
VALUES (3, '2024-03-15', 750.00);
```

7. Write an SQL query to update the contact information (e.g., email and address) of a specific customer in the "Customers" table.

```
UPDATE Customers  SET Email = 'newemail@example.com', Address = '456 New
Address St'
WHERE CustomerID = 3;
```

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

```
DELETE FROM OrderDetails
   WHERE OrderID IN (
 SELECT OrderID FROM Orders WHERE CustomerID = 4
   );
DELETE FROM Orders WHERE CustomerID = 4;
```

9.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 ('Bluetooth Speaker', 'Portable waterproof Bluetooth speaker', 800);
```

10.Write an SQL query to update the status of a specific order in the "Orders" table (e.g., from "Pending" to "Shipped").

```
ALTER TABLE Orders ADD COLUMN OrderStatus VARCHAR(20) DEFAULT 'Pending';
UPDATE Orders SET OrderStatus = 'Shipped' WHERE OrderID = 2;
```

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

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

```
 SELECT
 O.OrderID,
 C.FirstName,
 C.LastName,
 O.OrderDate,
 O.TotalAmount
 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 * P.Price) AS TotalRevenue
FROM OrderDetails OD
JOIN Products P ON OD.ProductID = P.ProductID
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 DISTINCT
 C.CustomerID,
 C.FirstName,
 C.LastName,
C.Email,
 C.Phone
 FROMCustomers C
 JOIN Orders O ON C.CustomerID = O.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 OrderDetails OD
JOIN Products P ON OD.ProductID = P.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.

```
SELECT
ProductName,
Descriptions AS 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
C.CustomerID,
C.FirstName,
C.LastName,
AVG(O.TotalAmount) AS AvgOrderValue
FROM Orders O
JOIN Customers C ON O.CustomerID = C.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 O.TotalAmount 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.OrderID) AS OrderCount
FROM OrderDetails OD
JOIN Products P ON OD.ProductID = P.ProductID
GROUP BY P.ProductName
ORDER BY OrderCount DESC;
```

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 COUNT(*) AS TotalProducts FROM Products;
```

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 total revenue generated by a specific customer.

```
SELECT SUM(O.TotalAmount) AS CustomerTotalRevenue
FROM Orders O
WHERE O.CustomerID = 1;
```

6. Write an SQL query to find the customers who have placed the most orders. List their names and the number of orders they've placed.

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

```
SELECT P.Category, SUM(OD.Quantity) AS TotalQuantityOrdered
FROM OrderDetails OD
JOIN Products P ON OD.ProductID = P.ProductID
GROUP BY P.Category
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, C.FirstName, C.LastName, SUM(O.TotalAmount) AS
TotalSpent
FROM Customers C
JOIN Orders O ON C.CustomerID = O.CustomerID
GROUP BY C.CustomerID
ORDER BY TotalSpent 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 AVG(O.TotalAmount) AS AvgOrderValue FROM Orders O;
```

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, COUNT(O.OrderID) AS
OrderCount
FROM Customers C
LEFT JOIN Orders O ON C.CustomerID = O.CustomerID
GROUP BY C.CustomerID;
```