Task 1:.Database Design:

1. Create the database named "TechShop"•

```
CREATE DATABASE TechShop;
USE TechShop;
```

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

```
CREATE TABLE Customers (
    ->      CustomerID INT PRIMARY KEY NOT NULL,
    ->      FirstName VARCHAR(50),
    ->      LastName VARCHAR(50),
    ->      Email TEXT,
    ->      Phone BIGINT,
    ->      ADDRESS LONGTEXT
    ->
    -> );
CREATE TABLE Products (
    ->      ProductID INT PRIMARY KEY NOT NULL,
    ->      ProductName VARCHAR(100),
    ->      Price DECIMAL(10, 2),
    ->      Description LONGTEXT
        -> );
CREATE TABLE Orders (
    ->      OrderID INT PRIMARY KEY NOT NULL,
    ->      CustomerID INT,
    ->      FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    ->      OrderDate DATE,
        ->      TOTALAMOUNT DOUBLE);

CREATE TABLE OrderDetails (
-       OrderDetailID INT PRIMARY KEY NOT NULL,
        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 NOT NULL,
    ->      ProductID INT,
    ->      QuantityinStock INT,
    ->      FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
        -> );
```

3. Create an ERO (Entity Relat onship Diagram) for the database.

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

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

- Customers

```
INSERT INTO Customers VALUES (1, 'John', 'Doe', 'johndoe@example.com', '123-456-7890', '123 Main St, Cityville');
INSERT INTO Customers VALUES (2, 'JANE', 'Foe', 'jANY@example.com', '234-567-4318', '163 Main St, US');
INSERT INTO Customers VALUES (3, 'John', 'Dep', 'johnydep@example.com', '567-980-2759', '03 Main St, AUSTRALIA');
INSERT INTO Customers VALUES (4, 'mowgli', 'dine', 'mowgli@example.com', '835-249-8742', '05 Main St,NEWZEALAND');
INSERT INTO Customers VALUES (5, 'thor', 'god', 'thor@example.com', '984-234-6756', '11 Main St, ASSGARD');
INSERT INTO Customers VALUES (6, 'ODIN', 'god', 'ODIN@example.com', '157-756-3248', '13 Main St, ASSGARD');
INSERT INTO Customers VALUES (7, 'IRON', 'god', 'IRON@example.com', '784-658-1453', '79 Main St, EARTH');
INSERT INTO Customers VALUES (8, 'HULK', 'god', 'HULK@example.com', '245-652-3791', '13 Main St, EARTH');
INSERT INTO Customers VALUES (9, 'CAPTAIN', 'god', 'CAPTAIN@example.com', '754-325-5641', '45 Main St, EARTH');
INSERT INTO Customers VALUES (10, 'ROCK', 'HADE', 'ROCK@example.com', '154-023-9007', '91 Main St, EARTH');
```

- Products

```
INSERT INTO Products VALUES (1, 'Laptop', 999.99, 'gaming laptop');
INSERT INTO Products VALUES (2, 'Smartphone', 499.99, 'Latest smartphone model');
INSERT INTO Products VALUES (3, 'Tablet', 299.99, 'Portable tablet device');
INSERT INTO Products VALUES (4, 'Smartwatch', 149.99, 'Fitness and smart features');
INSERT INTO Products VALUES (5, 'Desktop Computer', 1299.99, 'Powerful desktop computer');
INSERT INTO Products VALUES (6, 'Headphones', 79.99, 'Wireless over-ear headphones');
INSERT INTO Products VALUES (7, 'Digital Camera', 499.99, 'High-resolution digital camera');
INSERT INTO Products VALUES (8, 'External Hard Drive', 89.99, '1TB USB 3.0 external hard drive');
INSERT INTO Products VALUES (9, 'Wireless Mouse', 19.99, 'Ergonomic wireless mouse');
INSERT INTO Products VALUES (10, 'Gaming Laptop', 1499.99, 'High-performance gaming laptop');
```

- Orders

```
INSERT INTO Orders VALUES (1, 1, '2024-01-12','1999.9');
INSERT INTO Orders VALUES (2, 2, '2024-01-12','1899.9');
INSERT INTO Orders VALUES (3, 3, '2024-01-12','1799.9');
INSERT INTO Orders VALUES (4, 4, '2024-01-12','1699.9');
INSERT INTO Orders VALUES (5, 5, '2024-01-12','1599.9');
INSERT INTO Orders VALUES (6, 6, '2024-01-12','1499.9'),
INSERT INTO Orders VALUES (7, 7, '2024-01-12','1399.9'),
INSERT INTO Orders VALUES (8, 8, '2024-01-12','1299.9'),
INSERT INTO Orders VALUES (9, 9, '2024-01-12','1199.9'),
INSERT INTO Orders VALUES (10, 10, '2024-01-12','1099.9');
```

- OrderDetalls

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

- Inventory

```
  INSERT INTO Inventory (InventoryID, ProductID, QuantityInStock, LastStockUpdate)
-> VALUES
->       (1, 1, 15,   56, 50.00),
->       (2, 2, 25,   15),
->       (3, 3, 10,   123),
->       (4, 4, 30,   78),
->       (5, 5, 20,   48),
->       (6, 6, 15, 74 ),
->       (7, 7, 8,    75),
->       (8, 8, 12,   15),
->       (9, 9, 18,   90),
->       (10, 10, 10,10);
```

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

1. Write an SQLquery to retrieve the names and emails or all customers.

   SELECT FirstName, LastName, Email FROM Customers;

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

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

3. Write an SQL query Lo insert a new customer record nto the "Customers• table. hclude customer hformation such as name, email,and address.

INSERT INTO Customers VALUES (11, 'DEAD', 'POOL', 'DEAD@example.com', '456-173-1647', '093 Main St, NY');

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

   UPDATE Products SET Price = Price * 1.1;

5. Write an SQL query to delete a specific order and ls associated order details from the "Orders and orderDetans• tables Allow users toh put the orderD as a parameter

SET @OrderIdToDelete = <provide_order_id>;

START TRANSACTION;

DELETE FROM OrderDetails

WHERE OrderID = @OrderIdToDelete;

DELETE FROM Orders

WHERE OrderID = @OrderIdToDelete;

COMMIT;

6. Write an SQL query tohsert a new ordnto the "Orders· table hclude the customerD, order date,and any other necessary hformation.

   INSERT INTO Orders VALUES (11, 11, '2023-12-31','999.9');

7. Write an SQL query to update the contach nformation (eg, email and address) of a specific customer in the customers• table.Allow users to lnput the customerD and new contact lnformation.

   CREATE PROCEDURE UpdateCustomerContactInfo(IN p_CustomerID INT, IN p_NewEmail VARCHAR(255), IN p_NewAddress VARCHAR(255))

   -> BEGIN

   -> UPDATE Customers

```
    ->      SET Email = p_NewEmail, Address = p_NewAddress
    ->      WHERE CustomerID = p_CustomerID;
    -> END //
     DELIMITER ;
    CALL UpdateCustomerContactInfo(1, 'doe@example.com', '258 Main St AUCKLAND');
```

8. Write an SQl query to recalculate and update the total cost or each order Inthe orders• table based on the prices and quantities In the orderOetalls" table

```
UPDATE Orders
SET TotalAmount = (
    SELECT SUM(TotalAmount * Quantity)
    FROM OrderDetails
    WHERE OrderDetails.OrderID = Orders.OrderID
)
WHERE EXISTS (
    SELECT 1
    FROM OrderDetails
    WHERE OrderDetails.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 •orderOetalls" tables .Allow users tohput the customerD as a parameter.

```
-- Create a stored procedure
DELIMITER //

CREATE PROCEDURE DeleteOrdersAndDetailsForCustomer(IN p_CustomerID INT)
BEGIN
    DECLARE customerID INT;
    SET customerID = p_CustomerID;
    START TRANSACTION;
    DELETE FROM OrderDetails
    WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = customerID);
    DELETE FROM Orders
    WHERE CustomerID = customerID;
   COMMIT;
END //

DELIMITER ;

CALL DeleteOrdersAndDetailsForCustomer(<provide_customer_id>);
```

10. Write an SQ query tohsert a new electronic gadget product hto the "Products" table, hcludig product name, category, price, and any other relevant details.

```
INSERT INTO Products VALUES (11, 'JpyStick', 5899.99, 'gaming joystick');
```

11 Write an SQL query to update the status of a specific orderhn the •orders• table (e.g.,from "Pending• to 'Shipped").Allow users tohput the order ID and he new status

```
DECLARE @OrderID INT;
DECLARE @NewStatus VARCHAR(50);
SET @OrderID = 1
SET @NewStatus = 'Shipped ';
UPDATE Orders
SET Status = @NewStatus
WHERE OrderID = @OrderID;
```

12. Write an SQL query to calculate and updalhe number or orders pla by each customer in the "Customers table based on the datahn the orders· able

```
ALTER TABLE Customers ADD COLUMN NumberOfOrders INT;

UPDATE Customers
-> SET NumberOfOrders = (
->     SELECT COUNT(OrderID)
->     FROM Orders
->     WHERE Orders.CustomerID = Customers.CustomerID
-> )
-> WHERE EXISTS (
->     SELECT 1
->     FROM Orders
->     WHERE Orders.CustomerID = Customers.CustomerID
-> );
```

Task 3. Agregate functions,Halling. Order By,GroupBy and Joins:

1 Write an SQ query to retrieve a list of all 0<ders along with customer hformation (ea.. customer name) for each order

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

2 Wnte an SQl query to find the total revenue generated by each electronic gadcet product. hclude the product name and the totalrewnue

```
SELECT P.ProductName, SUM(OD.TotalAmount) AS TotalRevenue
    -> FROM Products P
```

```
-> JOIN OrderDetails OD ON P.ProductID = OD.ProductID
    -> GROUP BY P.ProductID, 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 FROM Customers 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, COALESCE(SUM(OD.Quantity), 0) AS TotalQuantityOrdered
FROM Products P
LEFT JOIN OrderDetails OD ON P.ProductID = OD.ProductID
GROUP BY P.ProductID, 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, Category
-> FROM Products
-> WHERE Category LIKE 'CHARGING%';
```

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(OD.TotalAmount) AS AverageOrderValue

FROM

    Customers C

JOIN

    Orders O ON C.CustomerID = O.CustomerID

JOIN

OrderDetails OD ON O.OrderID = OD.OrderID

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.

```
-- Find the order with the highest total revenue
SELECT
    O.OrderID,
    C.FirstName,
    C.LastName,
```

```
    C.Email,
    SUM(OD.TotalAmount) AS TotalRevenue
FROM
    Orders O
JOIN
    Customers C ON O.CustomerID = C.CustomerID
JOIN
    OrderDetails OD ON O.OrderID = OD.OrderID
GROUP BY
    O.OrderID, C.FirstName, C.LastName, C.Email
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.

```
-- List product names and their order counts
SELECT
    P.ProductID,
    P.ProductName,
    COUNT(OD.OrderID) AS OrderCount
FROM
    Products P
LEFT JOIN
    OrderDetails OD ON P.ProductID = OD.ProductID
GROUP BY
    P.ProductID, P.ProductName
ORDER BY
    OrderCount DESC;
```

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
    ->      C.CustomerID,
    ->      C.FirstName,
    ->      C.LastName,
    ->      C.Email
    -> FROM
    ->      Customers C
    -> JOIN
    ->      Orders O ON C.CustomerID = O.CustomerID
    -> JOIN
    ->      OrderDetails OD ON O.OrderID = OD.OrderID
    -> JOIN
    ->      Products P ON OD.ProductID = P.ProductID
    -> WHERE
```

```
    ->      P.ProductName = ' Gaming Laptop';
```

10. Wr te an SQL query to calculate the total revenue generated by all orders placed within a specific lme period. ARow users to input the start and end dates as parameters.

```
DECLARE @StartDate DATE = '2024-01-01'; -- Replace with the actual start date
DECLARE @EndDate DATE = '2024-12-31';     -- Replace with the actual end date
SELECT
SUM(OD.TotalAmount) AS TotalRevenue
FROM
Orders O
JOIN
OrderDetails OD ON O.OrderID = OD.OrderID
WHERE
O.OrderDate BETWEEN @StartDate AND @EndDate;
```

Task 4. Subquery and its type:

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

```
SELECT
    ->      C.CustomerID,
    ->      C.FirstName,
    ->      C.LastName,
    ->      C.Email
    -> FROM
    ->      Customers C
    -> LEFT JOIN
    ->      Orders O ON C.CustomerID = O.CustomerID
    -> WHERE
        ->      O.OrderID IS NULL;
```

2. Wme 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 average quantity ordered for products in a specific category. Allow users to input the category name as a parameter.

```
SELECT
    ->      P.Category,
    ->      AVG(OD.Quantity) AS AverageQuantityOrdered
    -> FROM
    ->      Products P
    -> JOIN
    ->      OrderDetails OD ON P.ProductID = OD.ProductID
    -> WHERE
    ->      P.Category = @CategoryName
    -> GROUP BY
    ->      P.Category;
```

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

```
SELECT
    ->      C.CustomerID,
    ->      C.FirstName,
    ->      C.LastName,
    ->      SUM(OD.TotalAmount) AS TotalRevenue
    -> FROM
    ->      Customers C
    -> JOIN
    ->      Orders O ON C.CustomerID = O.CustomerID
    -> JOIN
    ->      OrderDetails OD ON O.OrderID = OD.OrderID
    -> WHERE
    ->      C.CustomerID = @CustomerID
    -> GROUP BY
    ->      C.CustomerID, C.FirstName, C.LastName;
```

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 NumberOfOrders
    -> FROM
    ->     Customers C
    -> JOIN
    ->     Orders O ON C.CustomerID = O.CustomerID
    -> GROUP BY
    ->     C.CustomerID, C.FirstName, C.LastName
    -> ORDER BY
    ->     NumberOfOrders 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.

```
mysql> SELECT
    ->     P.Category,
    ->     SUM(OD.Quantity) AS TotalQuantityOrdered
    -> FROM
    ->     Products P
    -> JOIN
    ->     OrderDetails OD ON P.ProductID = OD.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(OD.TotalAmount) AS TotalSpending
    -> FROM
    ->     Customers C
    -> JOIN
    ->     Orders O ON C.CustomerID = O.CustomerID
    -> JOIN
    ->     OrderDetails OD ON O.OrderID = OD.OrderID
    -> JOIN
    ->     Products P ON OD.ProductID = P.ProductID
    -> WHERE
    ->     P.Category = 'CHARGING'
    -> GROUP BY
    ->     C.CustomerID, C.FirstName, C.LastName
    -> 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.

```
-- Calculate the average order value for all customers
mysql> SELECT
    ->     C.CustomerID,
    ->     C.FirstName,
    ->     C.LastName,
    ->     COUNT(O.OrderID) AS NumberOfOrders,
    ->     SUM(OD.TotalAmount) AS TotalRevenue,
    ->     AVG(OD.TotalAmount) AS AverageOrderValue
    -> FROM
    ->     Customers C
    -> JOIN
    ->     Orders O ON C.CustomerID = O.CustomerID
    -> JOIN
    ->     OrderDetails OD ON O.OrderID = OD.OrderID
    -> GROUP BY
    ->     C.CustomerID, C.FirstName, C.LastName
    -> ORDER BY
    ->     AverageOrderValue DESC;
```

10. Write an SQL query to find the total number of anders placed by each customer and list their names alongwith 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, C.FirstName, C.LastName
    -> ORDER BY
    ->     OrderCount DESC;
```