ASSIGNMENT

Task:1. Database Design:

1. Create the database named "TechShop"

QUERY:

```
mysql> create database techsop;
Query OK, 1 row affected (0.00 sec)
```

ANSWER:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| techshop           |
| techsop            |
+--------------------+
6 rows in set (0.02 sec)
```

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

QUERY:

```
mysql> CREATE TABLE Customers (
    ->     CustomerID INT,
    ->     FirstName VARCHAR(50),
    ->     LastName VARCHAR(50),
    ->     Email VARCHAR(100),
    ->     Phone VARCHAR(15),
    ->     Address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> CREATE TABLE Orders (
    ->     OrderID INT,
    ->     CustomerID INT,
    ->     OrderDate DATETIME,
    ->     TotalAmount DECIMAL(10,2)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> CREATE TABLE Products (
    ->     ProductID INT,
    ->     ProductName VARCHAR(100),
    ->     Description TEXT,
    ->     Price DECIMAL(10, 2)
    -> );
```

```
mysql> CREATE TABLE OrderDetails (
    ->     OrderDetailID INT,
    ->     OrderID INT,
    ->     ProductID INT,
    ->     Quantity INT
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Inventory (
    ->     InventoryID INT,
    ->     ProductID INT,
    ->     QuantityInStock INT,
    ->     LastStockUpdate DATETIME
    -> );
Query OK, 0 rows affected (0.02 sec)
```

ANSWERS:

```
mysql> show full tables;
+--------------------+------------+
| Tables_in_techshop | Table_type |
+--------------------+------------+
| customers          | BASE TABLE |
| inventory          | BASE TABLE |
| orderdetails       | BASE TABLE |
| orders             | BASE TABLE |
| products           | BASE TABLE |
+--------------------+------------+
5 rows in set (0.01 sec)
```

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

QUERY: PRIMARY KEY

```
mysql> ALTER TABLE Customers ADD PRIMARY KEY (CustomerID);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Products ADD PRIMARY KEY (ProductID);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Orders ADD PRIMARY KEY (OrderID);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE OrderDetails ADD PRIMARY KEY (OrderDetailID);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Inventory ADD PRIMARY KEY (InventoryID);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

FOREIGN KEY CONSTRAINTS:

```
mysql> ALTER TABLE Orders
    -> ADD CONSTRAINT FK_Orders_Customers
    -> FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE OrderDetails
    -> ADD CONSTRAINT FK_OrderDetails_Orders
    -> FOREIGN KEY (OrderID) REFERENCES Orders(OrderID);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql>
mysql> ALTER TABLE OrderDetails
    -> ADD CONSTRAINT FK_OrderDetails_Products
    -> FOREIGN KEY (ProductID) REFERENCES Products(ProductID);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Inventory
    -> ADD CONSTRAINT FK_Inventory_Products
    -> FOREIGN KEY (ProductID) REFERENCES Products(ProductID);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

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

a. Customers

b. Products

c. Orders

d. OrderDetails

e. Inventory

QUERY:

A)

```
mysql> INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone, Address) VALUES
    -> (1, 'John', 'Doe', 'john@example.com', '9876543210', '123 Main St'),
    -> (2, 'Jane', 'Smith', 'jane@example.com', '9876543211', '456 Oak Ave'),
    -> (3, 'Alice', 'Brown', 'alice@example.com', '9876543212', '789 Pine Rd'),
    -> (4, 'Bob', 'Jones', 'bob@example.com', '9876543213', '321 Cedar Blvd'),
    -> (5, 'Charlie', 'Miller', 'charlie@example.com', '9876543214', '654 Maple Dr'),
    -> (6, 'Diana', 'Clark', 'diana@example.com', '9876543215', '987 Birch Ln'),
    -> (7, 'Ethan', 'Wilson', 'ethan@example.com', '9876543216', '741 Elm St'),
    -> (8, 'Fiona', 'Davis', 'fiona@example.com', '9876543217', '159 Spruce Ct'),
    -> (9, 'George', 'Anderson', 'george@example.com', '9876543218', '258 Ash Pkwy'),
    -> (10, 'Hannah', 'Thomas', 'hannah@example.com', '9876543219', '369 Willow Way');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

ANS:

```
mysql> select * from customers;
+------------+-----------+----------+---------------------+------------+----------------+
| CustomerID | FirstName | LastName | Email               | Phone      | Address        |
+------------+-----------+----------+---------------------+------------+----------------+
|          1 | John      | Doe      | john@example.com    | 9876543210 | 123 Main St    |
|          2 | Jane      | Smith    | jane@example.com    | 9876543211 | 456 Oak Ave    |
|          3 | Alice     | Brown    | alice@example.com   | 9876543212 | 789 Pine Rd    |
|          4 | Bob       | Jones    | bob@example.com     | 9876543213 | 321 Cedar Blvd |
|          5 | Charlie   | Miller   | charlie@example.com | 9876543214 | 654 Maple Dr   |
|          6 | Diana     | Clark    | diana@example.com   | 9876543215 | 987 Birch Ln   |
|          7 | Ethan     | Wilson   | ethan@example.com   | 9876543216 | 741 Elm St     |
|          8 | Fiona     | Davis    | fiona@example.com   | 9876543217 | 159 Spruce Ct  |
|          9 | George    | Anderson | george@example.com  | 9876543218 | 258 Ash Pkwy   |
|         10 | Hannah    | Thomas   | hannah@example.com  | 9876543219 | 369 Willow Way |
+------------+-----------+----------+---------------------+------------+----------------+
10 rows in set (0.00 sec)
```

B)

```
mysql> INSERT INTO Products (ProductID, ProductName, Description, Price) VALUES
    -> (1, 'Laptop', '14-inch laptop with 8GB RAM', 55000.00),
    -> (2, 'Smartphone', '6.5-inch display, 128GB storage', 22000.00),
    -> (3, 'Keyboard', 'Mechanical keyboard', 2500.00),
    -> (4, 'Mouse', 'Wireless optical mouse', 1200.00),
    -> (5, 'Monitor', '24-inch Full HD monitor', 10500.00),
    -> (6, 'Headphones', 'Noise-cancelling headphones', 3500.00),
    -> (7, 'Webcam', 'HD webcam with mic', 2000.00),
    -> (8, 'Charger', '65W fast charger', 1800.00),
    -> (9, 'USB Cable', '1m Type-C cable', 500.00),
    -> (10, 'External HDD', '1TB external hard disk', 4500.00);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

ANS:

```
mysql> select * from products;
+-----------+-------------+---------------------------------+----------+
| ProductID | ProductName | Description                     | Price    |
+-----------+-------------+---------------------------------+----------+
|         1 | Laptop      | 14-inch laptop with 8GB RAM     | 55000.00 |
|         2 | Smartphone  | 6.5-inch display, 128GB storage | 22000.00 |
|         3 | Keyboard    | Mechanical keyboard             |  2500.00 |
|         4 | Mouse       | Wireless optical mouse          |  1200.00 |
|         5 | Monitor     | 24-inch Full HD monitor         | 10500.00 |
|         6 | Headphones  | Noise-cancelling headphones     |  3500.00 |
|         7 | Webcam      | HD webcam with mic              |  2000.00 |
|         8 | Charger     | 65W fast charger                |  1800.00 |
|         9 | USB Cable   | 1m Type-C cable                 |   500.00 |
|        10 | External HDD | 1TB external hard disk         |  4500.00 |
+-----------+-------------+---------------------------------+----------+
10 rows in set (0.00 sec)
```

C)

```
mysql> INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount) VALUES
    -> (1, 1, '2025-06-01', 57500.00),
    -> (2, 2, '2025-06-02', 23500.00),
    -> (3, 3, '2025-06-03', 13500.00),
    -> (4, 4, '2025-06-04', 1200.00),
    -> (5, 5, '2025-06-05', 9500.00),
    -> (6, 6, '2025-06-06', 3500.00),
    -> (7, 7, '2025-06-07', 5000.00),
    -> (8, 8, '2025-06-08', 22000.00),
    -> (9, 9, '2025-06-09', 6800.00),
    -> (10, 10, '2025-06-10', 1800.00);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

ANS:

```
mysql> SELECT * FROM ORDERS;
+---------+------------+---------------------+-------------+
| OrderID | CustomerID | OrderDate           | TotalAmount |
+---------+------------+---------------------+-------------+
|       1 |          1 | 2025-06-01 00:00:00 |    57500.00 |
|       2 |          2 | 2025-06-02 00:00:00 |    23500.00 |
|       3 |          3 | 2025-06-03 00:00:00 |    13500.00 |
|       4 |          4 | 2025-06-04 00:00:00 |     1200.00 |
|       5 |          5 | 2025-06-05 00:00:00 |     9500.00 |
|       6 |          6 | 2025-06-06 00:00:00 |     3500.00 |
|       7 |          7 | 2025-06-07 00:00:00 |     5000.00 |
|       8 |          8 | 2025-06-08 00:00:00 |    22000.00 |
|       9 |          9 | 2025-06-09 00:00:00 |     6800.00 |
|      10 |         10 | 2025-06-10 00:00:00 |     1800.00 |
+---------+------------+---------------------+-------------+
10 rows in set (0.00 sec)
```

D)

```
mysql> INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity) VALUES
    -> (1, 1, 1, 1),
    -> (2, 1, 3, 1),
    -> (3, 2, 2, 1),
    -> (4, 3, 5, 1),
    -> (5, 3, 4, 1),
    -> (6, 4, 4, 1),
    -> (7, 5, 5, 1),
    -> (8, 6, 6, 1),
    -> (9, 7, 7, 2),
    -> (10, 8, 2, 1);
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

ANS:

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

10 rows in set (0.00 sec)

E)

```
mysql> INSERT INTO Inventory (InventoryID, ProductID, QuantityInStock, LastStockUpdate) VALUES
    -> (1, 1, 50, '2025-06-01'),
    -> (2, 2, 100, '2025-06-01'),
    -> (3, 3, 75, '2025-06-02'),
    -> (4, 4, 200, '2025-06-02'),
    -> (5, 5, 60, '2025-06-03'),
    -> (6, 6, 120, '2025-06-04'),
    -> (7, 7, 80, '2025-06-05'),
    -> (8, 8, 90, '2025-06-06'),
    -> (9, 9, 150, '2025-06-07'),
    -> (10, 10, 40, '2025-06-08');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

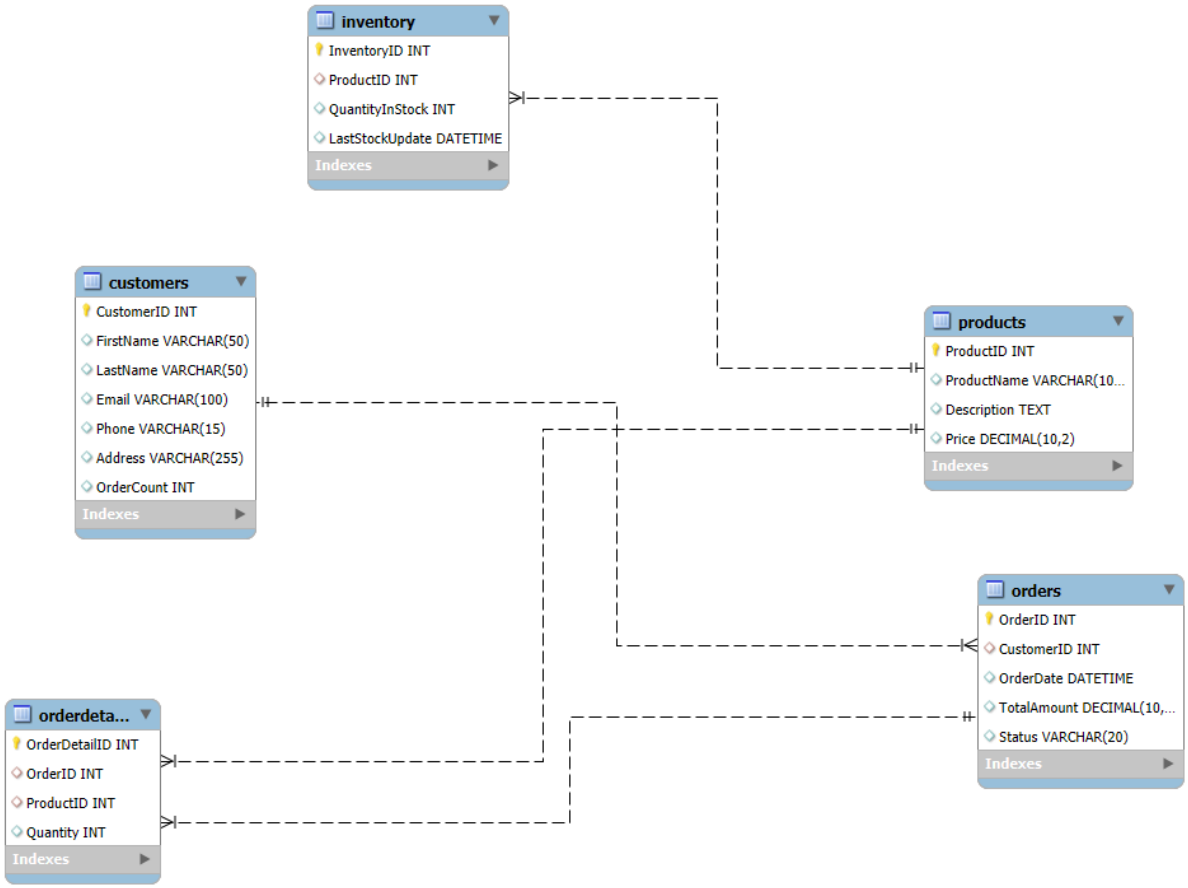
ANS:

```
mysql> SELECT * FROM INVENTORY;
```

InventoryID	ProductID	QuantityInStock	LastStockUpdate
1	1	50	2025-06-01 00:00:00
2	2	100	2025-06-01 00:00:00
3	3	75	2025-06-02 00:00:00
4	4	200	2025-06-02 00:00:00
5	5	60	2025-06-03 00:00:00
6	6	120	2025-06-04 00:00:00
7	7	80	2025-06-05 00:00:00
8	8	90	2025-06-06 00:00:00
9	9	150	2025-06-07 00:00:00
10	10	40	2025-06-08 00:00:00

10 rows in set (0.00 sec)

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



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

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

QUERY:

```
mysql> SELECT FIRSTNAME,LASTNAME,EMAIL FROM CUSTOMERS;
+-----------+----------+--------------------+
| FIRSTNAME | LASTNAME | EMAIL              |
+-----------+----------+--------------------+
| John      | Doe      | john@example.com   |
| Jane      | Smith    | jane@example.com   |
| Alice     | Brown    | alice@example.com  |
| Bob       | Jones    | bob@example.com    |
| Charlie   | Miller   | charlie@example.com |
| Diana     | Clark    | diana@example.com  |
| Ethan     | Wilson   | ethan@example.com  |
| Fiona     | Davis    | fiona@example.com  |
| George    | Anderson | george@example.com |
| Hannah    | Thomas   | hannah@example.com |
+-----------+----------+--------------------+
10 rows in set (0.00 sec)
```

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

QUERY:

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

ANS:

OrderID	OrderDate	FirstName	LastName
1	2025-06-01 00:00:00	John	Doe
2	2025-06-02 00:00:00	Jane	Smith
3	2025-06-03 00:00:00	Alice	Brown
4	2025-06-04 00:00:00	Bob	Jones
5	2025-06-05 00:00:00	Charlie	Miller
6	2025-06-06 00:00:00	Diana	Clark
7	2025-06-07 00:00:00	Ethan	Wilson
8	2025-06-08 00:00:00	Fiona	Davis
9	2025-06-09 00:00:00	George	Anderson
10	2025-06-10 00:00:00	Hannah	Thomas

10 rows in set (0.00 sec)

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

QUERY:

```
mysql> INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone, Address)
    -> values(11,'Bruce','Wayne','bruce@example.com','0000000000','000 Gotham st');
Query OK, 1 row affected (0.01 sec)
```

ANS:

```
mysql> select*from customers;
```

CustomerID	FirstName	LastName	Email	Phone	Address
1	John	Doe	john@example.com	9876543210	123 Main St
2	Jane	Smith	jane@example.com	9876543211	456 Oak Ave
3	Alice	Brown	alice@example.com	9876543212	789 Pine Rd
4	Bob	Jones	bob@example.com	9876543213	321 Cedar Blvd
5	Charlie	Miller	charlie@example.com	9876543214	654 Maple Dr
6	Diana	Clark	diana@example.com	9876543215	987 Birch Ln
7	Ethan	Wilson	ethan@example.com	9876543216	741 Elm St
8	Fiona	Davis	fiona@example.com	9876543217	159 Spruce Ct
9	George	Anderson	george@example.com	9876543218	258 Ash Pkwy
10	Hannah	Thomas	hannah@example.com	9876543219	369 Willow Way
11	Bruce	Wayne	bruce@example.com	0000000000	000 Gotham st

11 rows in set (0.00 sec)

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

QUERY:

```
mysql> UPDATE Products
    -> SET Price = Price * 1.10
    -> WHERE ProductName LIKE '%Laptop%'
    ->    OR ProductName LIKE '%Smartphone%'
    ->    OR ProductName LIKE '%Monitor%'
    ->    OR ProductName LIKE '%Headphones%'
    ->    OR ProductName LIKE '%Webcam%'
    ->    OR ProductName LIKE '%Charger%'
    ->    OR ProductName LIKE '%External%'
    ->    OR ProductName LIKE '%Keyboard%'
    ->    OR ProductName LIKE '%Mouse%';
Query OK, 9 rows affected (0.01 sec)
Rows matched: 9  Changed: 9  Warnings: 0
```

ANS:

```
mysql> select*from products;
+-----------+--------------+---------------------------------+----------+
| ProductID | ProductName  | Description                     | Price    |
+-----------+--------------+---------------------------------+----------+
|         1 | Laptop       | 14-inch laptop with 8GB RAM     | 60500.00 |
|         2 | Smartphone   | 6.5-inch display, 128GB storage | 24200.00 |
|         3 | Keyboard     | Mechanical keyboard             |  2750.00 |
|         4 | Mouse        | Wireless optical mouse          |  1320.00 |
|         5 | Monitor      | 24-inch Full HD monitor         | 11550.00 |
|         6 | Headphones   | Noise-cancelling headphones     |  3850.00 |
|         7 | Webcam       | HD webcam with mic              |  2200.00 |
|         8 | Charger      | 65W fast charger                |  1980.00 |
|         9 | USB Cable    | 1m Type-C cable                 |   500.00 |
|        10 | External HDD | 1TB external hard disk          |  4950.00 |
+-----------+--------------+---------------------------------+----------+
10 rows in set (0.00 sec)
```

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.

QUERY:

```
mysql> SET @ORDERID = 3;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM OrderDetails
    -> WHERE OrderID = @OrderID;
Query OK, 2 rows affected (0.02 sec)

mysql> DELETE FROM Orders
    -> WHERE OrderID = @OrderID;
Query OK, 1 row affected (0.01 sec)
```

ANS:

```
mysql> SELECT * FROM ORDERS;
+---------+------------+---------------------+-------------+
| OrderID | CustomerID | OrderDate           | TotalAmount |
+---------+------------+---------------------+-------------+
|       1 |          1 | 2025-06-01 00:00:00 |    57500.00 |
|       2 |          2 | 2025-06-02 00:00:00 |    23500.00 |
|       4 |          4 | 2025-06-04 00:00:00 |     1200.00 |
|       5 |          5 | 2025-06-05 00:00:00 |     9500.00 |
|       6 |          6 | 2025-06-06 00:00:00 |     3500.00 |
|       7 |          7 | 2025-06-07 00:00:00 |     5000.00 |
|       8 |          8 | 2025-06-08 00:00:00 |    22000.00 |
|       9 |          9 | 2025-06-09 00:00:00 |     6800.00 |
|      10 |         10 | 2025-06-10 00:00:00 |     1800.00 |
+---------+------------+---------------------+-------------+
9 rows in set (0.00 sec)

mysql> SELECT*FROM ORDERDETAILS;
+---------------+---------+-----------+----------+
| OrderDetailID | OrderID | ProductID | Quantity |
+---------------+---------+-----------+----------+
|             1 |       1 |         1 |        1 |
|             2 |       1 |         3 |        1 |
|             3 |       2 |         2 |        1 |
|             6 |       4 |         4 |        1 |
|             7 |       5 |         5 |        1 |
|             8 |       6 |         6 |        1 |
|             9 |       7 |         7 |        2 |
|            10 |       8 |         2 |        1 |
+---------------+---------+-----------+----------+
8 rows in set (0.00 sec)
```

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.

QUERY:

```
mysql> INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
    -> VALUES (11, 5, '2025-06-11', 15400.00);
Query OK, 1 row affected (0.01 sec)
```

ANS:

```
mysql> select * from orders;
+---------+------------+---------------------+-------------+
| OrderID | CustomerID | OrderDate           | TotalAmount |
+---------+------------+---------------------+-------------+
|       1 |          1 | 2025-06-01 00:00:00 |    57500.00 |
|       2 |          2 | 2025-06-02 00:00:00 |    23500.00 |
|       4 |          4 | 2025-06-04 00:00:00 |     1200.00 |
|       5 |          5 | 2025-06-05 00:00:00 |     9500.00 |
|       6 |          6 | 2025-06-06 00:00:00 |     3500.00 |
|       7 |          7 | 2025-06-07 00:00:00 |     5000.00 |
|       8 |          8 | 2025-06-08 00:00:00 |    22000.00 |
|       9 |          9 | 2025-06-09 00:00:00 |     6800.00 |
|      10 |         10 | 2025-06-10 00:00:00 |     1800.00 |
|      11 |          5 | 2025-06-11 00:00:00 |    15400.00 |
+---------+------------+---------------------+-------------+
10 rows in set (0.00 sec)
```

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.

QUERY:

```
mysql> UPDATE Customers
    -> SET Email = 'new.email@example.com',
    ->     Address = '123 New Address Street, City'
    -> WHERE CustomerID = 5;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

ANS:

```
mysql> SELECT * FROM CUSTOMERS;
+------------+-----------+----------+-----------------------+------------+------------------------------+
| CustomerID | FirstName | LastName | Email                 | Phone      | Address                      |
+------------+-----------+----------+-----------------------+------------+------------------------------+
|          1 | John      | Doe      | john@example.com      | 9876543210 | 123 Main St                  |
|          2 | Jane      | Smith    | jane@example.com      | 9876543211 | 456 Oak Ave                  |
|          3 | Alice     | Brown    | alice@example.com     | 9876543212 | 789 Pine Rd                  |
|          4 | Bob       | Jones    | bob@example.com       | 9876543213 | 321 Cedar Blvd               |
|          5 | Charlie   | Miller   | new.email@example.com | 9876543214 | 123 New Address Street, City |
|          6 | Diana     | Clark    | diana@example.com     | 9876543215 | 987 Birch Ln                 |
|          7 | Ethan     | Wilson   | ethan@example.com     | 9876543216 | 741 Elm St                   |
|          8 | Fiona     | Davis    | fiona@example.com     | 9876543217 | 159 Spruce Ct                |
|          9 | George    | Anderson | george@example.com    | 9876543218 | 258 Ash Pkwy                 |
|         10 | Hannah    | Thomas   | hannah@example.com    | 9876543219 | 369 Willow Way               |
|         11 | Bruce     | Wayne    | bruce@example.com     | 0000000000 | 000 Gotham st                |
+------------+-----------+----------+-----------------------+------------+------------------------------+
11 rows in set (0.00 sec)
```

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.

QUERY:

```
mysql> UPDATE Orders
    -> SET TotalAmount = (
    ->     SELECT SUM(od.Quantity * p.Price)
    ->     FROM OrderDetails od
    ->     JOIN Products p ON od.ProductID = p.ProductID
    ->     WHERE od.OrderID = Orders.OrderID
    -> );
Query OK, 10 rows affected (0.02 sec)
Rows matched: 10  Changed: 10  Warnings: 0
```

ANS:

```
mysql> SELECT * FROM ORDERS;
+---------+------------+---------------------+-------------+
| OrderID | CustomerID | OrderDate           | TotalAmount |
+---------+------------+---------------------+-------------+
|       1 |          1 | 2025-06-01 00:00:00 |    63250.00 |
|       2 |          2 | 2025-06-02 00:00:00 |    24200.00 |
|       4 |          4 | 2025-06-04 00:00:00 |     1320.00 |
|       5 |          5 | 2025-06-05 00:00:00 |    11550.00 |
|       6 |          6 | 2025-06-06 00:00:00 |     3850.00 |
|       7 |          7 | 2025-06-07 00:00:00 |     4400.00 |
|       8 |          8 | 2025-06-08 00:00:00 |    24200.00 |
|       9 |          9 | 2025-06-09 00:00:00 |        NULL |
|      10 |         10 | 2025-06-10 00:00:00 |        NULL |
|      11 |          5 | 2025-06-11 00:00:00 |        NULL |
+---------+------------+---------------------+-------------+
10 rows in set (0.00 sec)
```

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.

QUERY:

```
mysql> DELETE FROM OrderDetails
    -> WHERE OrderID IN (
    ->     SELECT OrderID FROM Orders WHERE CustomerID = 5);
Query OK, 1 row affected (0.02 sec)

mysql> DELETE FROM Orders
    -> WHERE CustomerID = 5;
Query OK, 2 rows affected (0.01 sec)
```

ANS:

```
mysql> SELECT*FROM ORDERS;
+---------+------------+---------------------+-------------+
| OrderID | CustomerID | OrderDate           | TotalAmount |
+---------+------------+---------------------+-------------+
|       1 |          1 | 2025-06-01 00:00:00 |    63250.00 |
|       2 |          2 | 2025-06-02 00:00:00 |    24200.00 |
|       4 |          4 | 2025-06-04 00:00:00 |     1320.00 |
|       6 |          6 | 2025-06-06 00:00:00 |     3850.00 |
|       7 |          7 | 2025-06-07 00:00:00 |     4400.00 |
|       8 |          8 | 2025-06-08 00:00:00 |    24200.00 |
|       9 |          9 | 2025-06-09 00:00:00 |        NULL |
|      10 |         10 | 2025-06-10 00:00:00 |        NULL |
+---------+------------+---------------------+-------------+
8 rows in set (0.00 sec)

mysql> SELECT*FROM ORDERDETAILS;
+--------------+---------+-----------+----------+
| OrderDetailID | OrderID | ProductID | Quantity |
+--------------+---------+-----------+----------+
|            1 |       1 |         1 |        1 |
|            2 |       1 |         3 |        1 |
|            3 |       2 |         2 |        1 |
|            6 |       4 |         4 |        1 |
|            8 |       6 |         6 |        1 |
|            9 |       7 |         7 |        2 |
|           10 |       8 |         2 |        1 |
+--------------+---------+-----------+----------+
7 rows in set (0.00 sec)
```

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.

QUERY:

```
mysql> INSERT into products(productid,productname,description,price)
    -> values(11,'bluetooth speakers','wireless',249.00);
Query OK, 1 row affected (0.01 sec)
```

ANS:

```
mysql> select*from products;
+-----------+--------------------+---------------------------------+----------+
| ProductID | ProductName        | Description                     | Price    |
+-----------+--------------------+---------------------------------+----------+
|         1 | Laptop             | 14-inch laptop with 8GB RAM     | 60500.00 |
|         2 | Smartphone         | 6.5-inch display, 128GB storage | 24200.00 |
|         3 | Keyboard           | Mechanical keyboard             |  2750.00 |
|         4 | Mouse              | Wireless optical mouse          |  1320.00 |
|         5 | Monitor            | 24-inch Full HD monitor         | 11550.00 |
|         6 | Headphones         | Noise-cancelling headphones     |  3850.00 |
|         7 | Webcam             | HD webcam with mic              |  2200.00 |
|         8 | Charger            | 65W fast charger                |  1980.00 |
|         9 | USB Cable          | 1m Type-C cable                 |   500.00 |
|        10 | External HDD       | 1TB external hard disk          |  4950.00 |
|        11 | bluetooth speakers | wireless                        |   249.00 |
+-----------+--------------------+---------------------------------+----------+
11 rows in set (0.00 sec)
```

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.

QUERY:

```
mysql> UPDATE Orders
    -> SET Status = 'Shipped'
    -> WHERE OrderID = 6;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

ANS:

```
mysql> SELECT * FROM ORDERS;
+---------+------------+---------------------+-------------+---------+
| OrderID | CustomerID | OrderDate           | TotalAmount | Status  |
+---------+------------+---------------------+-------------+---------+
|       1 |          1 | 2025-06-01 00:00:00 |    63250.00 | Pending |
|       2 |          2 | 2025-06-02 00:00:00 |    24200.00 | Pending |
|       4 |          4 | 2025-06-04 00:00:00 |     1320.00 | Pending |
|       6 |          6 | 2025-06-06 00:00:00 |     3850.00 | Shipped |
|       7 |          7 | 2025-06-07 00:00:00 |     4400.00 | Pending |
|       8 |          8 | 2025-06-08 00:00:00 |    24200.00 | Pending |
|       9 |          9 | 2025-06-09 00:00:00 |        NULL | Pending |
|      10 |         10 | 2025-06-10 00:00:00 |        NULL | Pending |
+---------+------------+---------------------+-------------+---------+
8 rows in set (0.00 sec)
```

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.

QUERY:

```
mysql> ALTER TABLE Customers ADD OrderCount INT DEFAULT 0;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> UPDATE Customers c
    -> SET c.OrderCount = (
    ->     SELECT COUNT(*)
    ->     FROM Orders o
    ->     WHERE o.CustomerID = c.CustomerID
    -> );
Query OK, 8 rows affected (0.01 sec)
Rows matched: 11  Changed: 8  Warnings: 0
```

ANS:

```
mysql> SELECT*FROM CUSTOMERS;
+------------+-----------+----------+---------------------+------------+-----------------------------+------------+
| CustomerID | FirstName | LastName | Email               | Phone      | Address                     | OrderCount |
+------------+-----------+----------+---------------------+------------+-----------------------------+------------+
|          1 | John      | Doe      | john@example.com    | 9876543210 | 123 Main St                 |          1 |
|          2 | Jane      | Smith    | jane@example.com    | 9876543211 | 456 Oak Ave                 |          1 |
|          3 | Alice     | Brown    | alice@example.com   | 9876543212 | 789 Pine Rd                 |          0 |
|          4 | Bob       | Jones    | bob@example.com     | 9876543213 | 321 Cedar Blvd              |          1 |
|          5 | Charlie   | Miller   | new.email@example.com | 9876543214 | 123 New Address Street, City |          0 |
|          6 | Diana     | Clark    | diana@example.com   | 9876543215 | 987 Birch Ln                |          1 |
|          7 | Ethan     | Wilson   | ethan@example.com   | 9876543216 | 741 Elm St                  |          1 |
|          8 | Fiona     | Davis    | fiona@example.com   | 9876543217 | 159 Spruce Ct               |          1 |
|          9 | George    | Anderson | george@example.com  | 9876543218 | 258 Ash Pkwy                |          1 |
|         10 | Hannah    | Thomas   | hannah@example.com  | 9876543219 | 369 Willow Way              |          1 |
|         11 | Bruce     | Wayne    | bruce@example.com   | 0000000000 | 000 Gotham st               |          0 |
+------------+-----------+----------+---------------------+------------+-----------------------------+------------+
11 rows in set (0.00 sec)
```

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**.**

QUERY:

```
mysql> SELECT
    ->     Orders.OrderID,
    ->     Orders.OrderDate,
    ->     Orders.TotalAmount,
    ->     Customers.FirstName,
    ->     Customers.LastName,
    ->     Customers.Phone
    -> FROM Orders
    -> JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

ANS:

OrderID	OrderDate	TotalAmount	FirstName	LastName	Phone
1	2025-06-01 00:00:00	63250.00	John	Doe	9876543210
2	2025-06-02 00:00:00	24200.00	Jane	Smith	9876543211
4	2025-06-04 00:00:00	1320.00	Bob	Jones	9876543213
6	2025-06-06 00:00:00	3850.00	Diana	Clark	9876543215
7	2025-06-07 00:00:00	4400.00	Ethan	Wilson	9876543216
8	2025-06-08 00:00:00	24200.00	Fiona	Davis	9876543217
9	2025-06-09 00:00:00	NULL	George	Anderson	9876543218
10	2025-06-10 00:00:00	NULL	Hannah	Thomas	9876543219

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

QUERY:

```
mysql> SELECT
    ->     p.ProductName,
    ->     SUM(od.Quantity * p.Price) AS TotalRevenue
    -> FROM OrderDetails od
    -> JOIN Products p ON od.ProductID = p.ProductID
    -> group by p.productid,p.productname;
```

ANS:

```
+-------------+--------------+
| ProductName | TotalRevenue |
+-------------+--------------+
| Laptop      |     60500.00 |
| Keyboard    |      2750.00 |
| Smartphone  |     48400.00 |
| Mouse       |      1320.00 |
| Headphones  |      3850.00 |
| Webcam      |      4400.00 |
+-------------+--------------+
6 rows in set (0.01 sec)
```

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

QUERY:

```
mysql> SELECT DISTINCT
    ->     c.FirstName,
    ->     c.LastName,
    ->     c.Email,
    ->     c.Phone,
    ->     c.Address
    -> FROM Customers c
    -> JOIN Orders o ON c.CustomerID = o.CustomerID;
```

ANS:

FirstName	LastName	Email	Phone	Address
John	Doe	john@example.com	9876543210	123 Main St
Jane	Smith	jane@example.com	9876543211	456 Oak Ave
Bob	Jones	bob@example.com	9876543213	321 Cedar Blvd
Diana	Clark	diana@example.com	9876543215	987 Birch Ln
Ethan	Wilson	ethan@example.com	9876543216	741 Elm St
Fiona	Davis	fiona@example.com	9876543217	159 Spruce Ct
George	Anderson	george@example.com	9876543218	258 Ash Pkwy
Hannah	Thomas	hannah@example.com	9876543219	369 Willow Way

8 rows in set (0.01 sec)

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

QUERY:

```
mysql> SELECT DISTINCT
    ->     c.FirstName,
    ->     c.LastName,
    ->     c.Email,
    ->     c.Phone,
    ->     c.Address
    -> FROM Customers c
    -> JOIN Orders o ON c.CustomerID = o.CustomerID;
```

ANS:

FirstName	LastName	Email	Phone	Address
John	Doe	john@example.com	9876543210	123 Main St
Jane	Smith	jane@example.com	9876543211	456 Oak Ave
Bob	Jones	bob@example.com	9876543213	321 Cedar Blvd
Diana	Clark	diana@example.com	9876543215	987 Birch Ln
Ethan	Wilson	ethan@example.com	9876543216	741 Elm St
Fiona	Davis	fiona@example.com	9876543217	159 Spruce Ct
George	Anderson	george@example.com	9876543218	258 Ash Pkwy
Hannah	Thomas	hannah@example.com	9876543219	369 Willow Way

8 rows in set (0.01 sec)

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.

QUERY:

```
mysql> SELECT
    ->     p.ProductName,
    ->     SUM(od.Quantity) AS TotalQuantityOrdered
    -> FROM OrderDetails od
    -> JOIN Products p ON od.ProductID = p.ProductID
    -> GROUP BY p.ProductID, p.ProductName
    -> ORDER BY TotalQuantityOrdered DESC
    -> LIMIT 1;
```

ANS:

```
+-------------+-----------------------+
| ProductName | TotalQuantityOrdered |
+-------------+-----------------------+
| Webcam      |                     2 |
+-------------+-----------------------+
1 row in set (0.01 sec)
```

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

QUERY:

```
mysql> SELECT
    ->     ProductName,
    ->     Description AS Category
    -> FROM Products;
```

ANS:

```
+--------------------+---------------------------------+
| ProductName        | Category                        |
+--------------------+---------------------------------+
| Laptop             | 14-inch laptop with 8GB RAM     |
| Smartphone         | 6.5-inch display, 128GB storage |
| Keyboard           | Mechanical keyboard             |
| Mouse              | Wireless optical mouse          |
| Monitor            | 24-inch Full HD monitor         |
| Headphones         | Noise-cancelling headphones     |
| Webcam             | HD webcam with mic              |
| Charger            | 65W fast charger                |
| USB Cable          | 1m Type-C cable                 |
| External HDD       | 1TB external hard disk          |
| bluetooth speakers | wireless                        |
+--------------------+---------------------------------+
11 rows in set (0.00 sec)
```

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

QUERY:

```
mysql> SELECT
    ->     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;
```

ANS:

```
+-----------+----------+--------------------+
| FirstName | LastName | AverageOrderValue  |
+-----------+----------+--------------------+
| John      | Doe      |       63250.000000 |
| Jane      | Smith    |       24200.000000 |
| Bob       | Jones    |        1320.000000 |
| Diana     | Clark    |        3850.000000 |
| Ethan     | Wilson   |        4400.000000 |
| Fiona     | Davis    |       24200.000000 |
| George    | Anderson |                NULL |
| Hannah    | Thomas   |                NULL |
+-----------+----------+--------------------+
8 rows in set (0.01 sec)
```

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

QUERY:

```
mysql> SELECT
    ->     o.OrderID,
    ->     c.FirstName,
    ->     c.LastName,
    ->     c.Email,
    ->     o.TotalAmount AS TotalRevenue
    -> FROM Orders o
    -> JOIN Customers c ON o.CustomerID = c.CustomerID
    -> ORDER BY o.TotalAmount DESC
    -> LIMIT 1;
```

ANS:

```
+---------+-----------+----------+------------------+--------------+
| OrderID | FirstName | LastName | Email            | TotalRevenue |
+---------+-----------+----------+------------------+--------------+
|       1 | John      | Doe      | john@example.com |     63250.00 |
+---------+-----------+----------+------------------+--------------+
1 row in set (0.00 sec)
```

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

QUERY:

```
mysql> SELECT
    ->     p.ProductName,
    ->     COUNT(od.OrderDetailID) AS TimesOrdered
    -> FROM Products p
    -> LEFT JOIN OrderDetails od ON p.ProductID = od.ProductID
    -> GROUP BY p.ProductID, p.ProductName;
```

ANS:

```
+--------------------+--------------+
| ProductName        | TimesOrdered |
+--------------------+--------------+
| Laptop             |            1 |
| Smartphone         |            2 |
| Keyboard           |            1 |
| Mouse              |            1 |
| Monitor            |            0 |
| Headphones         |            1 |
| Webcam             |            1 |
| Charger            |            0 |
| USB Cable          |            0 |
| External HDD       |            0 |
| bluetooth speakers |            0 |
+--------------------+--------------+
11 rows in set (0.01 sec)
```

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.

QUERY:

```
mysql> SELECT DISTINCT
    ->     c.FirstName,
    ->     c.LastName,
    ->     c.Email,
    ->     c.Phone
    -> 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 = 'HEADPHONES';
```

ANS:

```
+-----------+----------+-------------------+------------+
| FirstName | LastName | Email             | Phone      |
+-----------+----------+-------------------+------------+
| Diana     | Clark    | diana@example.com | 9876543215 |
+-----------+----------+-------------------+------------+
1 row in set (0.00 sec)
```

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.

QUERY:

```
mysql> SELECT
    ->     SUM(TotalAmount) AS TotalRevenue
    -> FROM Orders
    -> WHERE OrderDate BETWEEN '2025-06-01' AND '2025-06-10';
```

ANS:

```
+--------------+
| TotalRevenue |
+--------------+
|    121220.00 |
+--------------+
1 row in set (0.00 sec)
```

Task 4. Subquery and its type:

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

QUERY:

```
mysql> SELECT
    ->     CustomerID,
    ->     FirstName,
    ->     LastName,
    ->     Email
    -> FROM Customers
    -> WHERE CustomerID NOT IN (
    ->     SELECT DISTINCT CustomerID FROM Orders
    -> );
```

ANS:

```
+------------+-----------+----------+----------------------+
| CustomerID | FirstName | LastName | Email                |
+------------+-----------+----------+----------------------+
|          3 | Alice     | Brown    | alice@example.com    |
|          5 | Charlie   | Miller   | new.email@example.com |
|         11 | Bruce     | Wayne    | bruce@example.com    |
+------------+-----------+----------+----------------------+
3 rows in set (0.02 sec)
```

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

QUERY:

```
mysql> SELECT
    ->     o.OrderID,
    ->     o.OrderDate,
    ->     c.FirstName,
    ->     c.LastName
    -> FROM Orders o
    -> JOIN Customers c ON o.CustomerID = c.CustomerID;
```

ANS:

```
+---------+---------------------+-----------+----------+
| OrderID | OrderDate           | FirstName | LastName |
+---------+---------------------+-----------+----------+
|       1 | 2025-06-01 00:00:00 | John      | Doe      |
|       2 | 2025-06-02 00:00:00 | Jane      | Smith    |
|       4 | 2025-06-04 00:00:00 | Bob       | Jones    |
|       6 | 2025-06-06 00:00:00 | Diana     | Clark    |
|       7 | 2025-06-07 00:00:00 | Ethan     | Wilson   |
|       8 | 2025-06-08 00:00:00 | Fiona     | Davis    |
|       9 | 2025-06-09 00:00:00 | George    | Anderson |
|      10 | 2025-06-10 00:00:00 | Hannah    | Thomas   |
+---------+---------------------+-----------+----------+
8 rows in set (0.00 sec)
```

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

QUERY:

```
mysql> SELECT
    ->     SUM(od.Quantity * p.Price) AS TotalRevenue
    -> FROM
    ->     OrderDetails od
    -> JOIN
    ->     Products p ON od.ProductID = p.ProductID;
+--------------+
| TotalRevenue |
+--------------+
|    121220.00 |
+--------------+
1 row in set (0.01 sec)
```

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

QUERY:

```
mysql> SELECT
    ->     p.Description AS Category,
    ->     AVG(od.Quantity) AS AverageQuantityOrdered
    -> FROM
    ->     OrderDetails od
    -> JOIN
    ->     Products p ON od.ProductID = p.ProductID
    -> WHERE
    ->     p.Description = 'Mechanical keyboard'  -- ← Replace this with desired category
    -> GROUP BY
    ->     p.Description;
```

ANS:

```
+--------------------+--------------------------+
| Category           | AverageQuantityOrdered   |
+--------------------+--------------------------+
| Mechanical keyboard |                   1.0000 |
+--------------------+--------------------------+
1 row in set (0.00 sec)
```

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.

QUERY:

```
mysql> SELECT
    ->     c.CustomerID,
    ->     CONCAT(c.FirstName, ' ', c.LastName) AS CustomerName,
    ->     SUM(o.TotalAmount) AS TotalRevenue
    -> FROM
    ->     Customers c
    -> JOIN
    ->     Orders o ON c.CustomerID = o.CustomerID
    -> WHERE
    ->     c.CustomerID = 1   -- Replace 1 with the desired customer ID
    -> GROUP BY
    ->     c.CustomerID, c.FirstName, c.LastName;
```

ANS:

```
+------------+--------------+--------------+
| CustomerID | CustomerName | TotalRevenue |
+------------+--------------+--------------+
|          1 | John Doe     |     63250.00 |
+------------+--------------+--------------+
1 row in set (0.01 sec)
```

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.

QUERY:

```
mysql> SELECT
    ->     c.CustomerID,
    ->     CONCAT(c.FirstName, ' ', c.LastName) AS CustomerName,
    ->     COUNT(o.OrderID) AS NumberOfOrders
    -> FROM
    ->     Customers c
    -> JOIN
    ->     Orders o ON c.CustomerID = o.CustomerID
    -> GROUP BY
    ->     c.CustomerID, c.FirstName, c.LastName
    -> HAVING
    ->     COUNT(o.OrderID) = (
    ->         SELECT MAX(OrderCount)
    ->         FROM (
    ->             SELECT COUNT(OrderID) AS OrderCount
    ->             FROM Orders
    ->             GROUP BY CustomerID
    ->         ) AS OrderCounts
    ->     );
```

ANS:

```
+------------+-----------------+-----------------+
| CustomerID | CustomerName    | NumberOfOrders  |
+------------+-----------------+-----------------+
|          1 | John Doe        |               1 |
|          2 | Jane Smith      |               1 |
|          4 | Bob Jones       |               1 |
|          6 | Diana Clark     |               1 |
|          7 | Ethan Wilson    |               1 |
|          8 | Fiona Davis     |               1 |
|          9 | George Anderson |               1 |
|         10 | Hannah Thomas   |               1 |
+------------+-----------------+-----------------+
8 rows in set (0.01 sec)
```

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.

QUERY:

```
mysql> SELECT
    ->     p.Description AS Category,
    ->     SUM(od.Quantity) AS TotalQuantityOrdered
    -> FROM
    ->     OrderDetails od
    -> JOIN
    ->     Products p ON od.ProductID = p.ProductID
    -> GROUP BY
    ->     p.Description
    -> ORDER BY
    ->     TotalQuantityOrdered DESC
    -> LIMIT 1;
```

ANS:

```
+--------------------+-----------------------+
| Category           | TotalQuantityOrdered  |
+--------------------+-----------------------+
| HD webcam with mic |                     2 |
+--------------------+-----------------------+
1 row in set (0.00 sec)
```

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.

QUERY:

```
mysql> SELECT
    ->     CONCAT(c.FirstName, ' ', c.LastName) AS CustomerName,
    ->     SUM(od.Quantity * p.Price) AS TotalSpent
    -> 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 IN ('Laptop', 'Smartphone', 'Headphones', 'Webcam', 'Charger', 'External HDD', 'bluetooth speakers')
    -> GROUP BY
    ->     c.CustomerID
    -> ORDER BY
    ->     TotalSpent DESC
    -> LIMIT 1;
```

ANS:

```
+--------------+------------+
| CustomerName | TotalSpent |
+--------------+------------+
| John Doe     |   60500.00 |
+--------------+------------+
1 row in set (0.00 sec)
```

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

QUERY:

```
mysql> SELECT
    ->     AVG(TotalAmount) AS AverageOrderValue
    -> FROM
    ->     Orders
    -> WHERE
    ->     TotalAmount IS NOT NULL;
+-------------------+
| AverageOrderValue |
+-------------------+
|      20203.333333 |
+-------------------+
1 row in set (0.00 sec)
```

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.

QUERY:

```
mysql> 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;
```

ANS:

CustomerID	FirstName	LastName	OrderCount
1	John	Doe	1
2	Jane	Smith	1
3	Alice	Brown	0
4	Bob	Jones	1
5	Charlie	Miller	0
6	Diana	Clark	1
7	Ethan	Wilson	1
8	Fiona	Davis	1
9	George	Anderson	1
10	Hannah	Thomas	1
11	Bruce	Wayne	0

11 rows in set (0.00 sec)