**ELECTRONIC GADGETS SHOP**

**TASK1: Database Design**

1. **Create the Database**

CREATE DATABASE **TechShop**;

To use that use this query

USE **TechShop**;

1. **Define the Schema for Each Table**

**Customers table:**

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY AUTO\_INCREMENT,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL,

Phone VARCHAR(15),

Address VARCHAR(255)

);

Query OK, 0 rows affected (0.02 sec)

**Products table:**

CREATE TABLE Products (

ProductID INT PRIMARY KEY AUTO\_INCREMENT,

ProductName VARCHAR(100) NOT NULL,

Description TEXT,

Price DECIMAL(10, 2) NOT NULL

);

Query OK, 0 rows affected (0.01 sec)

**Orders table:**

CREATE TABLE Orders (

OrderID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT,

OrderDate DATE NOT NULL,

TotalAmount DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

Query OK, 0 rows affected (0.01 sec)

**OrderDetails table:**

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY AUTO\_INCREMENT,

OrderID INT,

ProductID INT,

Quantity INT NOT NULL,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

Query OK, 0 rows affected (0.01 sec)

**Inventory table:**

CREATE TABLE Inventory (

InventoryID INT PRIMARY KEY AUTO\_INCREMENT,

ProductID INT,

QuantityInStock INT NOT NULL,

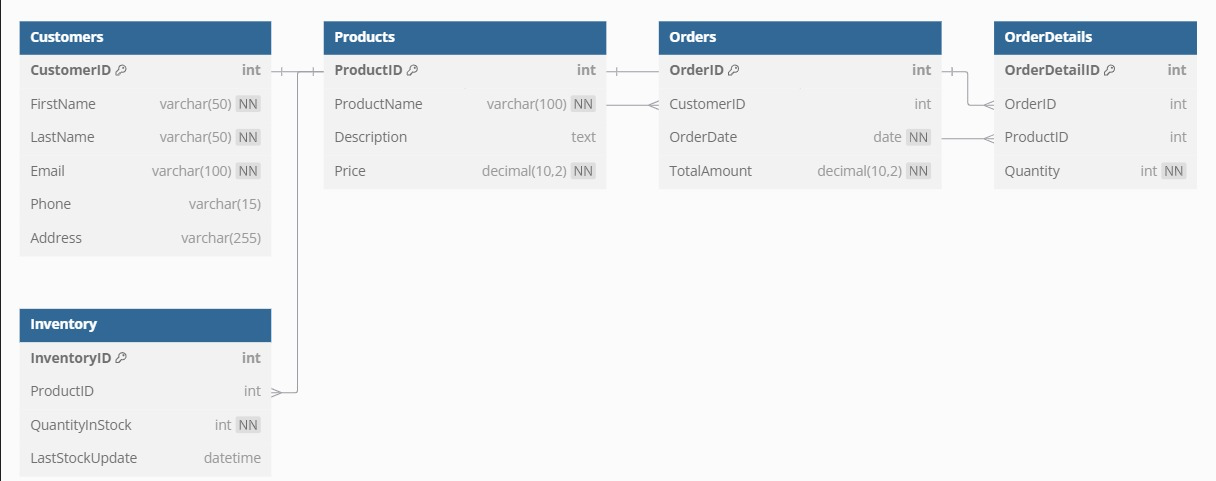
LastStockUpdate DATETIME,

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

Query OK, 0 rows affected (0.01 sec)

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



1. **Create Appropriate Primary Key and Foreign Key Constraints for referential integrity.**

• **Customers Table:**

**Primary Key:** CustomerID

There is no foreign key in customers table because it does not reference any

other table.

• **Products Table:**

**Primary Key:** ProductID

No foreign key in this table because it does not reference any other table.

• **Orders Table:**

**Primary Key:** OrderID

**Foreign Key:** CustomerID references Customers (CustomerID)

• **OrderDetails Table:**

**Primary Key:** OrderDetailID

**Foreign Key:** OrderID references Orders (OrderID)

**Foreign Key:** ProductID references Products (ProductID)

• **Inventory Table:**

**Primary Key:** InventoryID

**Foreign Key:** ProductID references Products (ProductID)

1. **Insert Sample Records into Each Table**
2. **Customers Table:**

INSERT INTO Customers (FirstName, LastName, Email, Phone, Address) VALUES

('Eli', 'Fox', 'ab@gmail.com', '13454545770', '123 street'),

('Ava', 'Lee', 'cd@gmail.com', '9876543210', '456 street'),

('Max', 'Roy', 'ef@gmail.com', '3456754323', '789 street'),

('Zoe', 'kim', 'gh@gmail.com', '5557676787', '321 street'),

('Ian', 'Wu', 'ij@gmail.com', '8787878787', '458 street'),

('Mia', 'Ray', 'kl@gmail.com', '6234565432', '751 street'),

('Ben', 'Kay', 'mn@gmail.com', '9988776655', '991 street'),

('Sam', 'Lin', 'op@gmail.com', '4544554455', '340 street'),

('Eve', 'Tan', 'qr@gmail.com', '2345675432', '565 street'),

('Leo', 'Park', 'st@gmail.com', '9878654383', '789street');

+------------+-----------+----------+--------------+-------------+------------+

| CustomerID | FirstName | LastName | Email | Phone | Address |

+------------+-----------+----------+--------------+-------------+------------+

| 1 | Eli | Fox | ab@gmail.com | 13454545770 | 123 street |

| 2 | Ava | Lee | cd@gmail.com | 9876543210 | 456 street |

| 3 | Max | Roy | ef@gmail.com | 3456754323 | 789 street |

| 4 | Zoe | kim | gh@gmail.com | 5557676787 | 321 street |

| 5 | Ian | Wu | ij@gmail.com | 8787878787 | 458 street |

| 6 | Mia | Ray | kl@gmail.com | 6234565432 | 751 street |

| 7 | Ben | Kay | mn@gmail.com | 9988776655 | 991 street |

| 8 | Sam | Lin | op@gmail.com | 4544554455 | 340 street |

| 9 | Eve | Tan | qr@gmail.com | 2345675432 | 565 street |

| 10 | Leo | Park | st@gmail.com | 9878654383 | 789street |

+------------+-----------+----------+--------------+-------------+------------+

**b. Products Table:**

INSERT INTO Products (ProductName, Description, Price) VALUES

('Smartphone ', '5G & high-resolution cam', 8999),

('Laptop ', '16GB RAM & 512GB SSD', 69999),

('Tablet ', '4GB RAM & 64GB storage', 35000),

('Smartwatch ', 'heart rate monitor &GPS', 6000),

('Headphones ', 'Active noise cancellation', 25000),

('Bluetooth Speaker ', 'deep bass &12-hour battery life', 15000),

('Gaming Console ', '4K HDR support', 39999),

('Camera ', '24.2MP with Wi-Fi & Bluetooth', 60000),

('Wireless Charger ', '10W compatible', 1500),

('Smart Home Hub ', 'Voice-controlled with Wi-Fi connectivity', 2000);

**c.Orders Table:**

INSERT INTO Orders (CustomerID, OrderDate, TotalAmount) VALUES

(1, '2024-09-1', 80000),

(2, '2024-09-6', 100000),

(3, '2024-09-7', 7000),

(4, '2024-09-8', 60000),

(5, '2024-09-9', 5000),

(6, '2024-09-2', 1500),

(7, '2024-09-10', 40000),

(8, '2024-09-11', 60000),

(9, '2024-09-12', 35000),

(10, '2024-09-14', 45000);

+---------+------------+------------+-------------+

| OrderID | CustomerID | OrderDate | TotalAmount |

+---------+------------+------------+-------------+

| 1 | 1 | 2024-09-01 | 80000.00 |

| 2 | 2 | 2024-09-06 | 100000.00 |

| 3 | 3 | 2024-09-07 | 7000.00 |

| 4 | 4 | 2024-09-08 | 60000.00 |

| 5 | 5 | 2024-09-09 | 5000.00 |

| 6 | 6 | 2024-09-02 | 1500.00 |

| 7 | 7 | 2024-09-10 | 40000.00 |

| 8 | 8 | 2024-09-11 | 60000.00 |

| 9 | 9 | 2024-09-12 | 35000.00 |

| 10 | 10 | 2024-09-14 | 45000.00 |

+---------+------------+------------+-------------+

**d. OrderDetails Table:**

INSERT INTO OrderDetails (OrderID, ProductID, Quantity) VALUES

(1, 1, 1),

(1, 5, 1),

(2, 2, 1),

(3, 4, 1),

(4, 7, 1),

(5, 2, 1),

(5, 8, 1),

(6, 5, 1),

(7, 10, 1),

(8, 2, 1),

(8, 6, 1),

(9, 3, 1),

(10, 4, 1);

+---------------+---------+-----------+----------+

| OrderDetailID | OrderID | ProductID | Quantity |

+---------------+---------+-----------+----------+

| 1 | 1 | 1 | 1 |

| 2 | 1 | 5 | 1 |

| 3 | 2 | 2 | 1 |

| 4 | 3 | 4 | 1 |

| 5 | 4 | 7 | 1 |

| 6 | 5 | 2 | 1 |

| 7 | 5 | 8 | 1 |

| 8 | 6 | 5 | 1 |

| 9 | 7 | 10 | 1 |

| 10 | 8 | 2 | 1 |

| 11 | 8 | 6 | 1 |

| 12 | 9 | 3 | 1 |

| 13 | 10 | 4 | 1 |

+---------------+---------+-----------+----------+

**e. Inventory Table:**

INSERT INTO Inventory (ProductID, QuantityInStock, LastStockUpdate) VALUES

(1, 50, '2024-09-10 10:00'),

(2, 30, '2024-09-11 11:00'),

(3, 40, '2024-09-12 12:00'),

(4, 25, '2024-09-13 13:00'),

(5, 60, '2024-09-14 14:00'),

(6, 75, '2024-09-15 15:00'),

(7, 15, '2024-09-16 16:00'),

(8, 20, '2024-09-17 17:00'),

(9, 80, '2024-09-18 18:00'),

(10, 10, '2024-09-19 19:00');

+-------------+-----------+-----------------+---------------------+

| InventoryID | ProductID | QuantityInStock | LastStockUpdate |

+-------------+-----------+-----------------+---------------------+

| 1 | 1 | 50 | 2024-09-10 10:00:00 |

| 2 | 2 | 30 | 2024-09-11 11:00:00 |

| 3 | 3 | 40 | 2024-09-12 12:00:00 |

| 4 | 4 | 25 | 2024-09-13 13:00:00 |

| 5 | 5 | 60 | 2024-09-14 14:00:00 |

| 6 | 6 | 75 | 2024-09-15 15:00:00 |

| 7 | 7 | 15 | 2024-09-16 16:00:00 |

| 8 | 8 | 20 | 2024-09-17 17:00:00 |

| 9 | 9 | 80 | 2024-09-18 18:00:00 |

| 10 | 10 | 10 | 2024-09-19 19:00:00 |

+-------------+-----------+-----------------+---------------------+

**TASK2: 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;

+-----------+----------+--------------+

| FirstName | LastName | Email |

+-----------+----------+--------------+

| Eli | Fox | ab@gmail.com |

| Ava | Lee | cd@gmail.com |

| Max | Roy | ef@gmail.com |

| Zoe | kim | gh@gmail.com |

| Ian | Wu | ij@gmail.com |

| Mia | Ray | kl@gmail.com |

| Ben | Kay | mn@gmail.com |

| Sam | Lin | op@gmail.com |

| Eve | Tan | qr@gmail.com |

| Leo | Park | st@gmail.com |

+-----------+----------+--------------+

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

**names.**

SELECT OrderID, OrderDate, FirstName, LastName

FROM Orders

JOIN Customers USING (CustomerID);

+---------+------------+-----------+----------+

| OrderID | OrderDate | FirstName | LastName |

+---------+------------+-----------+----------+

| 1 | 2024-09-01 | Eli | Fox |

| 2 | 2024-09-06 | Ava | Lee |

| 3 | 2024-09-07 | Max | Roy |

| 4 | 2024-09-08 | Zoe | kim |

| 5 | 2024-09-09 | Ian | Wu |

| 6 | 2024-09-02 | Mia | Ray |

| 7 | 2024-09-10 | Ben | Kay |

| 8 | 2024-09-11 | Sam | Lin |

| 9 | 2024-09-12 | Eve | Tan |

| 10 | 2024-09-14 | Leo | Park |

+---------+------------+-----------+----------+

**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 ('Liam', 'zen', 'xy@gmail.com', '5551234567', '123street');

+------------+-----------+----------+--------------+-------------+------------+

| CustomerID | FirstName | LastName | Email | Phone | Address |

+------------+-----------+----------+--------------+-------------+------------+

| 1 | Eli | Fox | ab@gmail.com | 13454545770 | 123 street |

| 2 | Ava | Lee | cd@gmail.com | 9876543210 | 456 street |

| 3 | Max | Roy | ef@gmail.com | 3456754323 | 789 street |

| 4 | Zoe | kim | gh@gmail.com | 5557676787 | 321 street |

| 5 | Ian | Wu | ij@gmail.com | 8787878787 | 458 street |

| 6 | Mia | Ray | kl@gmail.com | 6234565432 | 751 street |

| 7 | Ben | Kay | mn@gmail.com | 9988776655 | 991 street |

| 8 | Sam | Lin | op@gmail.com | 4544554455 | 340 street |

| 9 | Eve | Tan | qr@gmail.com | 2345675432 | 565 street |

| 10 | Leo | Park | st@gmail.com | 9878654383 | 789street |

| 11 | Liam | zen | xy@gmail.com | 5551234567 | 123street |

+------------+-----------+----------+--------------+-------------+------------+

**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. Allow users to input the order ID as a parameter.**

DELETE FROM OrderDetails WHERE OrderID = 1;

+---------------+---------+-----------+----------+

| OrderDetailID | OrderID | ProductID | Quantity |

+---------------+---------+-----------+----------+

| 3 | 2 | 2 | 1 |

| 4 | 3 | 4 | 1 |

| 5 | 4 | 7 | 1 |

| 6 | 5 | 2 | 1 |

| 7 | 5 | 8 | 1 |

| 8 | 6 | 5 | 1 |

| 9 | 7 | 10 | 1 |

| 10 | 8 | 2 | 1 |

| 11 | 8 | 6 | 1 |

| 12 | 9 | 3 | 1 |

| 13 | 10 | 4 | 1 |

+---------------+---------+-----------+----------+

**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 (1, '2024-09-25', 299.99);

+---------+------------+------------+-------------+

| OrderID | CustomerID | OrderDate | TotalAmount |

+---------+------------+------------+-------------+

| 2 | 2 | 2024-09-06 | 1329.99 |

| 3 | 3 | 2024-09-07 | 199.99 |

| 4 | 4 | 2024-09-08 | 999.99 |

| 5 | 5 | 2024-09-09 | 950.98 |

| 6 | 6 | 2024-09-02 | 150.99 |

| 7 | 7 | 2024-09-10 | 179.99 |

| 8 | 8 | 2024-09-11 | 1999.99 |

| 9 | 9 | 2024-09-12 | 399.99 |

| 10 | 10 | 2024-09-14 | 499.99 |

| 11 | 1 | 2024-09-25 | 299.99 |

+---------+------------+------------+-------------+

**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.**

UPDATE Customers

SET Email = 'new@gmail.com', Address = 'New, 33street'

WHERE CustomerID = 1;

+------------+-----------+----------+---------------+-------------+---------------+

| CustomerID | FirstName | LastName | Email | Phone | Address |

+------------+-----------+----------+---------------+-------------+---------------+

| 1 | Eli | Fox | new@gmail.com | 13454545770 | New, 33street |

| 2 | Ava | Lee | cd@gmail.com | 9876543210 | 456 street |

| 3 | Max | Roy | ef@gmail.com | 3456754323 | 789 street |

| 4 | Zoe | kim | gh@gmail.com | 5557676787 | 321 street |

| 5 | Ian | Wu | ij@gmail.com | 8787878787 | 458 street |

| 6 | Mia | Ray | kl@gmail.com | 6234565432 | 751 street |

| 7 | Ben | Kay | mn@gmail.com | 9988776655 | 991 street |

| 8 | Sam | Lin | op@gmail.com | 4544554455 | 340 street |

| 9 | Eve | Tan | qr@gmail.com | 2345675432 | 565 street |

| 10 | Leo | Park | st@gmail.com | 9878654383 | 789street |

| 11 | Liam | zen | xy@gmail.com | 5551234567 | 123street |

+------------+-----------+----------+---------------+-------------+---------------+

**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 o

SET TotalAmount = (

SELECT SUM(od.Quantity \* p.Price)

FROM OrderDetails od

JOIN Products p USING (ProductID)

WHERE od.OrderID = o.OrderID

);

+---------+------------+------------+-------------+

| OrderID | CustomerID | OrderDate | TotalAmount |

+---------+------------+------------+-------------+

| 2 | 2 | 2024-09-06 | 76998.90 |

| 3 | 3 | 2024-09-07 | 6600.00 |

| 4 | 4 | 2024-09-08 | 43998.90 |

| 5 | 5 | 2024-09-09 | 142998.90 |

| 6 | 6 | 2024-09-02 | 27500.00 |

| 7 | 7 | 2024-09-10 | 2200.00 |

| 8 | 8 | 2024-09-11 | 93498.90 |

| 9 | 9 | 2024-09-12 | 38500.00 |

| 10 | 10 | 2024-09-14 | 6600.00 |

| 11 | 1 | 2024-09-25 | 0.00 |

+---------+------------+------------+-------------+

**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.**

DELETE FROM OrderDetails

WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = 1);

+---------------+---------+-----------+----------+

| OrderDetailID | OrderID | ProductID | Quantity |

+---------------+---------+-----------+----------+

| 3 | 2 | 2 | 1 |

| 4 | 3 | 4 | 1 |

| 5 | 4 | 7 | 1 |

| 6 | 5 | 2 | 1 |

| 7 | 5 | 8 | 1 |

| 8 | 6 | 5 | 1 |

| 9 | 7 | 10 | 1 |

| 10 | 8 | 2 | 1 |

| 11 | 8 | 6 | 1 |

| 12 | 9 | 3 | 1 |

| 13 | 10 | 4 | 1 |

+---------------+---------+-----------+----------+

DELETE FROM Orders

WHERE CustomerID = 1;

+---------+------------+------------+-------------+

| OrderID | CustomerID | OrderDate | TotalAmount |

+---------+------------+------------+-------------+

| 2 | 2 | 2024-09-06 | 76998.90 |

| 3 | 3 | 2024-09-07 | 6600.00 |

| 4 | 4 | 2024-09-08 | 43998.90 |

| 5 | 5 | 2024-09-09 | 142998.90 |

| 6 | 6 | 2024-09-02 | 27500.00 |

| 7 | 7 | 2024-09-10 | 2200.00 |

| 8 | 8 | 2024-09-11 | 93498.90 |

| 9 | 9 | 2024-09-12 | 38500.00 |

| 10 | 10 | 2024-09-14 | 6600.00 |

+---------+------------+------------+-------------+

**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, Description, Price)

VALUES ('TV ', '55-inch with HDR', 20000);

+-----------+--------------------+------------------------------------------+----------+

| ProductID | ProductName | Description | Price |

+-----------+--------------------+------------------------------------------+----------+

| 1 | Smartphone | 5G & high-resolution cam | 9898.90 |

| 2 | Laptop | 16GB RAM & 512GB SSD | 76998.90 |

| 3 | Tablet | 4GB RAM & 64GB storage | 38500.00 |

| 4 | Smartwatch | heart rate monitor &GPS | 6600.00 |

| 5 | Headphones | Active noise cancellation | 27500.00 |

| 6 | Bluetooth Speaker | deep bass &12-hour battery life | 16500.00 |

| 7 | Gaming Console | 4K HDR support | 43998.90 |

| 8 | Camera | 24.2MP with Wi-Fi & Bluetooth | 66000.00 |

| 9 | Wireless Charger | 10W compatible | 1650.00 |

| 10 | Smart Home Hub | Voice-controlled with Wi-Fi connectivity | 2200.00

| 11 | TV | 55-inch with HDR | 20000.00 |

+-----------+--------------------+------------------------------------------+----------+

**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);

+---------+------------+------------+-------------+--------+

| OrderID | CustomerID | OrderDate | TotalAmount | Status |

+---------+------------+------------+-------------+--------+

| 2 | 2 | 2024-09-06 | 76998.90 | NULL |

| 3 | 3 | 2024-09-07 | 6600.00 | NULL |

| 4 | 4 | 2024-09-08 | 43998.90 | NULL |

| 5 | 5 | 2024-09-09 | 142998.90 | NULL |

| 6 | 6 | 2024-09-02 | 27500.00 | NULL |

| 7 | 7 | 2024-09-10 | 2200.00 | NULL |

| 8 | 8 | 2024-09-11 | 93498.90 | NULL |

| 9 | 9 | 2024-09-12 | 38500.00 | NULL |

| 10 | 10 | 2024-09-14 | 6600.00 | NULL |

+---------+------------+------------+-------------+--------+--

Once the `Status` column is added, you can use the update query:

The Query is

UPDATE Orders

SET Status = 'Shipped'

WHERE OrderID = 2;

+---------+------------+------------+-------------+---------+

| OrderID | CustomerID | OrderDate | TotalAmount | Status |

+---------+------------+------------+-------------+---------+

| 2 | 2 | 2024-09-06 | 76998.90 | Shipped |

| 3 | 3 | 2024-09-07 | 6600.00 | NULL |

| 4 | 4 | 2024-09-08 | 43998.90 | NULL |

| 5 | 5 | 2024-09-09 | 142998.90 | NULL |

| 6 | 6 | 2024-09-02 | 27500.00 | NULL |

| 7 | 7 | 2024-09-10 | 2200.00 | NULL |

| 8 | 8 | 2024-09-11 | 93498.90 | NULL |

| 9 | 9 | 2024-09-12 | 38500.00 | NULL |

| 10 | 10 | 2024-09-14 | 6600.00 | NULL |

+---------+------------+------------+-------------+---------+

9 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.**

Add the OrderCount column:

ALTER TABLE Customers

ADD OrderCount INT DEFAULT 0;

Query to calculate and update the number of orders:

UPDATE Customers C

JOIN (SELECT CustomerID, COUNT(\*) AS OrderCount

FROM Orders

GROUP BY CustomerID) AS T ON C.CustomerID = T.CustomerID

SET C.OrderCount = T.OrderCount;

+------------+-----------+----------+---------------+-------------+---------------+------------+

| CustomerID | FirstName | LastName | Email | Phone | Address | OrderCount |

+------------+-----------+----------+---------------+-------------+---------------+------------+

| 1 | Eli | Fox | new@gmail.com | 13454545770 | New, 33street | 0 |

| 2 | Ava | Lee | cd@gmail.com | 9876543210 | 456 street | 1 |

| 3 | Max | Roy | ef@gmail.com | 3456754323 | 789 street | 1 |

| 4 | Zoe | kim | gh@gmail.com | 5557676787 | 321 street | 1 |

| 5 | Ian | Wu | ij@gmail.com | 8787878787 | 458 street | 1 |

| 6 | Mia | Ray | kl@gmail.com | 6234565432 | 751 street | 1 |

| 7 | Ben | Kay | mn@gmail.com | 9988776655 | 991 street | 1 |

| 8 | Sam | Lin | op@gmail.com | 4544554455 | 340 street | 1 |

| 9 | Eve | Tan | qr@gmail.com | 2345675432 | 565 street | 1 |

| 10 | Leo | Park | st@gmail.com | 9878654383 | 789street | 1 |

| 11 | Liam | zen | xy@gmail.com | 5551234567 | 123street | 0 |

+------------+-----------+----------+---------------+-------------+---------------+------------+

11 rows in set (0.00 sec)

OR

We can just use left join as follows:

SELECT c.CustomerID, COUNT(o.OrderID) AS OrderCount

FROM Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID;

+------------+------------+

| CustomerID | OrderCount |

+------------+------------+

| 1 | 0 |

| 2 | 1 |

| 3 | 1 |

| 4 | 1 |

| 5 | 1 |

| 6 | 1 |

| 7 | 1 |

| 8 | 1 |

| 9 | 1 |

| 10 | 1 |

| 11 | 0 |

+------------+------------+

11 rows in set (0.01 sec)

**Task 3:**

**Aggregate Functions, HAVING, ORDER BY, GROUP BY, and Joins**

**1.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

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID;

+---------+-----------+----------+------------+

| OrderID | FirstName | LastName | OrderDate |

+---------+-----------+----------+------------+

| 2 | Ava | Lee | 2024-09-06 |

| 3 | Max | Roy | 2024-09-07 |

| 4 | Zoe | kim | 2024-09-08 |

| 5 | Ian | Wu | 2024-09-09 |

| 6 | Mia | Ray | 2024-09-02 |

| 7 | Ben | Kay | 2024-09-10 |

| 8 | Sam | Lin | 2024-09-11 |

| 9 | Eve | Tan | 2024-09-12 |

| 10 | Leo | Park | 2024-09-14 |

+---------+-----------+----------+------------+

2.**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;

+--------------------+--------------+

| ProductName | TotalRevenue |

+--------------------+--------------+

| Laptop | 230996.70 |

| Smartwatch | 13200.00 |

| Gaming Console | 43998.90 |

| Camera | 66000.00 |

| Headphones | 27500.00 |

| Smart Home Hub | 2200.00 |

| Bluetooth Speaker | 16500.00 |

| Tablet | 38500.00 |

+--------------------+--------------+

3.**List all customers who have made at least one purchase. Include their names and contact information.**

SELECT DISTINCT c.FirstName, c.LastName, c.Email, c.Phone, c.Address

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID;

+-----------+----------+--------------+------------+------------+

| FirstName | LastName | Email | Phone | Address |

+-----------+----------+--------------+------------+------------+

| Ava | Lee | cd@gmail.com | 9876543210 | 456 street |

| Max | Roy | ef@gmail.com | 3456754323 | 789 street |

| Zoe | kim | gh@gmail.com | 5557676787 | 321 street |

| Ian | Wu | ij@gmail.com | 8787878787 | 458 street |

| Mia | Ray | kl@gmail.com | 6234565432 | 751 street |

| Ben | Kay | mn@gmail.com | 9988776655 | 991 street |

| Sam | Lin | op@gmail.com | 4544554455 | 340 street |

| Eve | Tan | qr@gmail.com | 2345675432 | 565 street |

| Leo | Park | st@gmail.com | 9878654383 | 789street |

+-----------+----------+--------------+------------+------------+

4.**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;

+-------------+----------------------+

| ProductName | TotalQuantityOrdered |

+-------------+----------------------+

| Laptop | 3 |

+-------------+----------------------+

5.**Retrieve a list of electronic gadgets along with their corresponding categories.**

SELECT ProductName

FROM Products;

+--------------------+

| ProductName |

+--------------------+

| Smartphone |

| Laptop |

| Tablet |

| Smartwatch |

| Headphones |

| Bluetooth Speaker |

| Gaming Console |

| Camera |

| Wireless Charger |

| Smart Home Hub |

| TV |

+--------------------+

6.**Calculate the average order value for each customer. Include the customer’s name and their average order value.**

SELECT c.FirstName, c.LastName, AVG(TotalAmount) AS AverageOrderValue

FROM Customers c, Orders o

WHERE c.CustomerID = o.CustomerID

GROUP BY c.CustomerID;

+-----------+----------+-------------------+

| FirstName | LastName | AverageOrderValue |

+-----------+----------+-------------------+

| Ava | Lee | 76998.900000 |

| Max | Roy | 6600.000000 |

| Zoe | kim | 43998.900000 |

| Ian | Wu | 142998.900000 |

| Mia | Ray | 27500.000000 |

| Ben | Kay | 2200.000000 |

| Sam | Lin | 93498.900000 |

| Eve | Tan | 38500.000000 |

| Leo | Park | 6600.000000 |

+-----------+----------+-------------------+

9 rows in set (0.00 sec)

7.**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

FROM Orders o, Customers c

WHERE o.CustomerID = c.CustomerID

ORDER BY o.TotalAmount DESC

LIMIT 1;

+---------+-----------+----------+-------------+

| OrderID | FirstName | LastName | TotalAmount |

+---------+-----------+----------+-------------+

| 5 | Ian | Wu | 142998.90 |

+---------+-----------+----------+-------------+

8.**List electronic gadgets and the number of times each product has been ordered.**

SELECT p. ProductName, COUNT(\*) AS TimesOrdered

FROM OrderDetails od

JOIN Products p ON od. ProductID = p.ProductID

GROUP BY p. ProductName;

+--------------------+--------------+

| ProductName | TimesOrdered |

+--------------------+--------------+

| Laptop | 3 |

| Tablet | 1 |

| Smartwatch | 2 |

| Headphones | 1 |

| Bluetooth Speaker | 1 |

| Gaming Console | 1 |

| Camera | 1 |

| Smart Home Hub | 1 |

+--------------------+--------------+

9.**Find customers who have purchased a specific electronic gadget product. Allow users to input the product name as a parameter.**

SELECT AVG(od.Quantity) AS AverageQuantity

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

WHERE p.ProductName = 'Tablet';

+-----------------+

| AverageQuantity |

+-----------------+

| NULL |

+-----------------+

1 row in set (0.00 sec)

10.**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.**

SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders

WHERE OrderDate BETWEEN 2024-09-02 AND 2024-09-25 ;

+--------------+

| TotalRevenue |

+--------------+

| NULL |

+--------------+

**Task 4: Subqueries and Their Types**

1. **Find out which customers have not placed any orders.**

SELECT c.FirstName, c.LastName

FROM Customers c

WHERE c.CustomerID NOT IN (SELECT CustomerID FROM Orders);

+-----------+----------+

| FirstName | LastName |

+-----------+----------+

| Eli | Fox |

| Liam | zen |

+-----------+----------+

**2. Find the total number of products available for sale.**

SELECT COUNT(\*) AS TotalProducts

FROM Products;

**+---------------+**

**| TotalProducts |**

**+---------------+**

**| 11 |**

**+---------------+**

**3. Calculate the total revenue generated by TechShop.**

SELECT SUM(o.TotalAmount) AS TotalRevenue

FROM Orders o;

+--------------+

| TotalRevenue |

+--------------+

| 438895.60 |

+--------------+

**4****.** **Calculate the average quantity ordered for products in a specific category.**

SELECT

AVG(od.Quantity) AS AverageQuantity

FROM

OrderDetails od

INNER JOIN

Products p ON od.ProductID = p.ProductID

WHERE

LOWER(TRIM(p.ProductName)) = 'tv';

+-----------------+

| AverageQuantity |

+-----------------+

| 1.0000 |

+-----------------+

1 row in set (0.00 sec)

**5. Calculate the total revenue generated by a specific customer.**

SELECT SUM(o.TotalAmount) AS TotalRevenue

FROM Orders o

WHERE o.CustomerID = 2;

+--------------+

| TotalRevenue |

+--------------+

| 76998.90 |

+--------------+

1 row in set (0.00 sec)

**6. Find the customers who have placed the most orders.**

SELECT 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;

+-----------+----------+------------+

| FirstName | LastName | OrderCount |

+-----------+----------+------------+

| Ava | Lee | 1 |

+-----------+----------+------------+

1 row in set (0.01 sec)

1. **Find the most popular product category, which is the one with the highest total quantity ordered across all orders.**

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;

+-------------+----------------------+

| ProductName | TotalQuantityOrdered |

+-------------+----------------------+

| Laptop | 3 |

+-------------+----------------------+

1 row in set (0.00 sec)

8**. Find the customer who has spent the most money (highest total revenue) on electronic gadgets.**

SELECT 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.

+-----------+----------+------------+

| FirstName | LastName | TotalSpent |

+-----------+----------+------------+

| Ian | Wu | 142998.90 |

+-----------+----------+------------+

1 row in set (0.00 sec)

9**. Calculate the average order value (total revenue divided by the number of orders) for all customers.**

SELECT AVG(TotalAmount) AS AverageOrderValue

FROM Orders;

+-------------------+

| AverageOrderValue |

+-------------------+

| 48766.177778 |

+-------------------+

1 row in set (0.00 sec)

10**. Find the total number of orders placed by each customer and list their names along with the order count.**

SELECT 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;

+-----------+----------+------------+

| FirstName | LastName | OrderCount |

+-----------+----------+------------+

| Eli | Fox | 0 |

| Ava | Lee | 1 |

| Max | Roy | 1 |

| Zoe | kim | 1 |

| Ian | Wu | 1 |

| Mia | Ray | 1 |

| Ben | Kay | 1 |

| Sam | Lin | 1 |

| Eve | Tan | 1 |

| Leo | Park | 1 |

| Liam | zen | 0 |

+-----------+----------+------------+

11 rows in set (0.00 sec)