Set the environment

CREATE DATABASE salesdb;

USE salesdb;

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

PhoneNumber VARCHAR(20),

Address VARCHAR(100),

City VARCHAR(50),

State VARCHAR(50),

ZipCode VARCHAR(10),

Country VARCHAR(50),

SignUpDate DATE

);

INSERT INTO Customers VALUES

(1, 'John', 'Doe', 'john.doe@example.com', '123-456-7890', '123 Elm St', 'Springfield', 'IL', '62701', 'USA', '2023-01-15'),

(2, 'Jane', 'Smith', 'jane.smith@example.com', '234-567-8901', '456 Oak St', 'Chicago', 'IL', '60601', 'USA', '2023-02-20'),

(3, 'Alice', 'Johnson', 'alice.johnson@example.com', '345-678-9012', '789 Pine St', 'Houston', 'TX', '77001', 'USA', '2023-03-10');

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10, 2),

StockQuantity INT,

SupplierID INT

);

INSERT INTO Products VALUES

(1, 'Laptop', 'Electronics', 999.99, 50, 1),

(2, 'Smartphone', 'Electronics', 599.99, 200, 2),

(3, 'Tablet', 'Electronics', 399.99, 150, 1);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

TotalAmount DECIMAL(10, 2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Orders VALUES

(1, 1, '2023-03-15', 1599.98),

(2, 2, '2023-04-10', 599.99),

(3, 3, '2023-05-05', 999.99);

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT,

UnitPrice DECIMAL(10, 2),

TotalPrice DECIMAL(10, 2),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

INSERT INTO OrderDetails VALUES

(1, 1, 1, 1, 999.99, 999.99),

(2, 1, 2, 1, 599.99, 599.99),

(3, 2, 2, 1, 599.99, 599.99),

(4, 3, 1, 1, 999.99, 999.99);

CREATE TABLE Reviews (

ReviewID INT PRIMARY KEY,

CustomerID INT,

ProductID INT,

Rating INT,

ReviewDate DATE,

Comments TEXT,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

INSERT INTO Reviews VALUES

(1, 1, 1, 5, '2023-03-20', 'Great laptop!'),

(2, 2, 2, 4, '2023-04-15', 'Good smartphone, but battery life could be better.'),

(3, 3, 1, 5, '2023-05-10', 'Excellent performance!');

Easy Questions

1. **List all customers' names and email addresses.**

SELECT FirstName, LastName, Email  
FROM Customers;

1. **Retrieve all products with their names and prices.**

SELECT ProductName, Price  
FROM Products;

1. **Find all orders placed on '2023-04-10'.**

SELECT \*  
FROM Orders  
WHERE OrderDate = '2023-04-10';

1. **Get the total amount of all orders placed by the customer with ID 1.**

SELECT SUM(TotalAmount) AS TotalSpent  
FROM Orders  
WHERE CustomerID = 1;

1. **List all reviews with a rating of 5.**

SELECT \*  
FROM Reviews  
WHERE Rating = 5;

Intermediate Questions

1. **Find the total quantity of each product sold.**

SELECT ProductID, SUM(Quantity) AS TotalQuantitySold  
FROM OrderDetails  
GROUP BY ProductID;

1. **List the names of customers who have placed an order with a total amount greater than $1000.**

SELECT DISTINCT c.FirstName, c.LastName  
FROM Customers c  
JOIN Orders o ON c.CustomerID = o.CustomerID  
WHERE o.TotalAmount > 1000;

1. **Get the average rating for each product.**

SELECT ProductID, AVG(Rating) AS AverageRating  
FROM Reviews  
GROUP BY ProductID;

1. **Retrieve the order details (including product name and quantity) for order ID 1.**

SELECT p.ProductName, od.Quantity  
FROM OrderDetails od  
JOIN Products p ON od.ProductID = p.ProductID  
WHERE od.OrderID = 1;

1. **Find the most recent order date for each customer.**

SELECT CustomerID, MAX(OrderDate) AS MostRecentOrderDate  
FROM Orders  
GROUP BY CustomerID;

### Medium Questions

1. **Find the top 3 products with the highest total sales amount.**

SELECT p.ProductName, SUM(od.TotalPrice) AS TotalSales  
FROM OrderDetails od  
JOIN Products p ON od.ProductID = p.ProductID  
GROUP BY p.ProductName  
ORDER BY TotalSales DESC  
LIMIT 3;

1. **List customers who have never placed an order.**

SELECT c.CustomerID, c.FirstName, c.LastName  
FROM Customers c  
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID  
WHERE o.OrderID IS NULL;

1. **Retrieve the total number of orders and the total sales amount for each month in 2023.**

SELECT DATE\_FORMAT(OrderDate, '%Y-%m') AS Month, COUNT(\*) AS TotalOrders, SUM(TotalAmount) AS TotalSales  
FROM Orders  
WHERE YEAR(OrderDate) = 2023  
GROUP BY DATE\_FORMAT(OrderDate, '%Y-%m');

1. **Find all customers who have written more than one review.**

SELECT CustomerID, COUNT(\*) AS ReviewCount  
FROM Reviews  
GROUP BY CustomerID  
HAVING ReviewCount > 1;

1. **Get the average rating for products in each category.**

SELECT p.Category, AVG(r.Rating) AS AverageRating  
FROM Reviews r  
JOIN Products p ON r.ProductID = p.ProductID  
GROUP BY p.Category;

### Hard Questions

1. **Find the customer who spent the most money on orders.**

SELECT c.CustomerID, c.FirstName, c.LastName, SUM(o.TotalAmount) AS TotalSpent  
FROM Customers c  
JOIN Orders o ON c.CustomerID = o.CustomerID  
GROUP BY c.CustomerID  
ORDER BY TotalSpent DESC  
LIMIT 1;

1. **List the customers and their total order amount if they have ordered products from more than one category.**

SELECT c.CustomerID, c.FirstName, c.LastName, SUM(o.TotalAmount) AS TotalOrderAmount  
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  
GROUP BY c.CustomerID  
HAVING COUNT(DISTINCT p.Category) > 1;

1. **Retrieve the product that has the highest average rating.**

SELECT p.ProductID, p.ProductName, AVG(r.Rating) AS AverageRating  
FROM Reviews r  
JOIN Products p ON r.ProductID = p.ProductID  
GROUP BY p.ProductID  
ORDER BY AverageRating DESC  
LIMIT 1;

1. **Get the list of products that were never ordered.**

SELECT p.ProductID, p.ProductName  
FROM Products p  
LEFT JOIN OrderDetails od ON p.ProductID = od.ProductID  
WHERE od.OrderDetailID IS NULL;

1. **Find the total sales amount and the number of orders for each product category.**

SELECT p.Category, COUNT(DISTINCT o.OrderID) AS TotalOrders, SUM(od.TotalPrice) AS TotalSales  
FROM Products p  
JOIN OrderDetails od ON p.ProductID = od.ProductID  
JOIN Orders o ON od.OrderID = o.OrderID  
GROUP BY p.Category;