**Task 1: Database Design**

1. Create the database

CREATE DATABASE TechShop;

2. Create Tables

USE TechShop;

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100) UNIQUE,

Phone VARCHAR(15),

Address VARCHAR(255)

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(50),

Description VARCHAR(255),

Price DECIMAL(10,2)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

TotalAmount DECIMAL(10,2),

Status VARCHAR(20),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID) ON DELETE CASCADE,

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

CREATE TABLE Inventory (

InventoryID INT PRIMARY KEY,

ProductID INT,

QuantityInStock INT,

LastStockUpdate DATE,

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

5. Insert Sample Records

INSERT INTO Customers VALUES

(1, 'John', 'Doe', 'john.doe@example.com', '1234567890', '123 Main St'),

(2, 'Jane', 'Smith', 'jane.smith@example.com', '9876543210', '456 Oak St');

INSERT INTO Products VALUES

(1, 'Laptop', 'High Performance Laptop', 1200.00),

(2, 'Smartphone', 'Latest Model Smartphone', 800.00);

INSERT INTO Orders VALUES

(1, 1, '2025-03-01', 2000.00, 'Pending'),

(2, 2, '2025-03-02', 800.00, 'Shipped');

INSERT INTO OrderDetails VALUES

(1, 1, 1, 1),

(2, 2, 2, 1);

INSERT INTO Inventory VALUES

(1, 1, 50, '2025-02-28'),

(2, 2, 30, '2025-02-28');

**Task 2: Select, Where, Between, AND, LIKE**

-- 1. Retrieve names and emails of all customers

SELECT FirstName, LastName, Email FROM Customers;

-- 2. List all orders with their order dates and customer names

SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName

FROM Orders

JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

-- 3. Insert a new customer

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

VALUES (3, 'Emily', 'Johnson', 'emily.johnson@example.com', '1122334455', '789 Pine St');

-- 4. Update prices of electronic gadgets by increasing them by 10%

UPDATE Products

SET Price = Price \* 1.10

WHERE ProductName LIKE '%Laptop%' OR ProductName LIKE '%Smartphone%';

-- 5. Delete a specific order with associated details

DELETE FROM OrderDetails WHERE OrderID = 1;

DELETE FROM Orders WHERE OrderID = 1;

-- 6. Insert a new order

INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount, Status)

VALUES (3, 3, '2025-03-03', 1500.00, 'Pending');

-- 7. Update customer contact information

UPDATE Customers

SET Email = 'new.email@example.com', Address = '123 Updated St'

WHERE CustomerID = 1;

-- 8. Recalculate and update total cost of orders

UPDATE Orders

SET TotalAmount = (

SELECT SUM(Products.Price \* OrderDetails.Quantity)

FROM OrderDetails

JOIN Products ON OrderDetails.ProductID = Products.ProductID

WHERE OrderDetails.OrderID = Orders.OrderID

);

-- 9. Delete all orders for a specific customer

DELETE FROM OrderDetails WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = 2);

DELETE FROM Orders WHERE CustomerID = 2;

-- 10. Insert a new product

INSERT INTO Products (ProductID, ProductName, Description, Price)

VALUES (3, 'Tablet', 'Latest Model Tablet', 500.00);

-- 11. Update order status

UPDATE Orders

SET Status = 'Shipped'

WHERE OrderID = 3;

-- 12. Calculate the number of orders

SELECT COUNT(\*) AS TotalOrders FROM Orders;

**Task 3: Aggregate Functions, Having, Order By, Group By, and Joins**

-- 1. Retrieve all orders with customer information

SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName

FROM Orders

JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

-- 2. Calculate total revenue for each product

SELECT Products.ProductName, SUM(Products.Price \* OrderDetails.Quantity) AS TotalRevenue

FROM OrderDetails

JOIN Products ON OrderDetails.ProductID = Products.ProductID

GROUP BY Products.ProductName;

-- 3. List customers who made at least one purchase

SELECT DISTINCT Customers.FirstName, Customers.LastName, Customers.Email

FROM Orders

JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

-- 4. Find the most popular product

SELECT Products.ProductName, SUM(OrderDetails.Quantity) AS TotalQuantity

FROM OrderDetails

JOIN Products ON OrderDetails.ProductID = Products.ProductID

GROUP BY Products.ProductName

ORDER BY TotalQuantity DESC

LIMIT 1;

-- 5. Retrieve product categories and products

SELECT ProductName, Description FROM Products;

-- 6. Calculate the average order value

SELECT Customers.FirstName, Customers.LastName, AVG(Orders.TotalAmount) AS AverageOrderValue

FROM Orders

JOIN Customers ON Orders.CustomerID = Customers.CustomerID

GROUP BY Customers.CustomerID;

-- 7. Find the highest revenue order

SELECT Orders.OrderID, Customers.FirstName, Customers.LastName, Orders.TotalAmount

FROM Orders

JOIN Customers ON Orders.CustomerID = Customers.CustomerID

ORDER BY Orders.TotalAmount DESC

LIMIT 1;

-- 8. List products and their order count

SELECT Products.ProductName, COUNT(OrderDetails.OrderDetailID) AS OrderCount

FROM OrderDetails

JOIN Products ON OrderDetails.ProductID = Products.ProductID

GROUP BY Products.ProductName;

-- 9. Find customers who purchased a specific product

SELECT DISTINCT Customers.FirstName, Customers.LastName

FROM Orders

JOIN Customers ON Orders.CustomerID = Customers.CustomerID

JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID

WHERE OrderDetails.ProductID = 1;

**Task 4: Subquery and Its Type**

-- 1. Find customers without orders

SELECT FirstName, LastName FROM Customers

WHERE CustomerID NOT IN (SELECT DISTINCT CustomerID FROM Orders);

-- 2. Find the total number of products

SELECT COUNT(\*) AS TotalProducts FROM Products;

-- 3. Calculate total revenue

SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders;

-- 4. Calculate average quantity per category

SELECT AVG(OrderDetails.Quantity) AS AverageQuantity

FROM OrderDetails

JOIN Products ON OrderDetails.ProductID = Products.ProductID

WHERE Products.Description LIKE '%Tablet%';

-- 5. Calculate revenue for a specific customer

SELECT SUM(TotalAmount) AS CustomerRevenue FROM Orders

WHERE CustomerID = 1;

-- 6. Find customers with the most orders

SELECT CustomerID, COUNT(\*) AS OrderCount FROM Orders

GROUP BY CustomerID

ORDER BY OrderCount DESC

LIMIT 1;

-- 7. Find the most popular category

SELECT Products.Description, SUM(OrderDetails.Quantity) AS TotalQuantity

FROM OrderDetails

JOIN Products ON OrderDetails.ProductID = Products.ProductID

GROUP BY Products.Description

ORDER BY TotalQuantity DESC

LIMIT 1;

-- 8. Find the customer who spent the most money (highest total revenue)

SELECT Customers.FirstName, Customers.LastName, SUM(Orders.TotalAmount) AS TotalSpent

FROM Customers

JOIN Orders ON Customers.CustomerID = Orders.CustomerID

GROUP BY Customers.CustomerID, Customers.FirstName, Customers.LastName

ORDER BY TotalSpent DESC

LIMIT 1;

-- 9. Calculate the average order value for all customers

SELECT AVG(TotalAmount) AS AverageOrderValue

FROM Orders;

-- 10. Find the total number of orders placed by each customer

SELECT Customers.FirstName, Customers.LastName, COUNT(Orders.OrderID) AS OrderCount

FROM Customers

LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID

GROUP BY Customers.CustomerID, Customers.FirstName, Customers.LastName;