**TechShop, an electronic gadgets shop**

You are working as a database administrator for a fictional company named "TechShop," which sells electronic gadgets. TechShop maintains data related to their products, customers, and orders. Your task is to design and implement a database for TechShop based on the following requirements:

**Database Tables:**

**1.Customers:**

• CustomerID (Primary Key)

• FirstName

• LastName

• Email

• Phone

• Address

**2.Products:**

• ProductID (Primary Key)

• ProductName

• Description

• Price

**3.Orders:**

• OrderID (Primary Key)

• CustomerID (Foreign Key referencing Customers)

• OrderDate

• TotalAmount

**4.OrderDetails:**

• OrderDetailID (Primary Key)

• OrderID (Foreign Key referencing Orders)

• ProductID (Foreign Key referencing Products)

• Quantity

**5. Inventory**

• InventoryID (Primary Key)

• ProductID (Foreign Key referencing Products)

• QuantityInStock

• LastStockUpdate

**Task:1. Database Design:**

1. **Create the database named "TechShop"**

CREATE DATABASE TechShop;

USE TechShop;

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

CREATE TABLE Customers (

CustomerID INT AUTO\_INCREMENT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

Phone VARCHAR(20),

Address VARCHAR(255)

);

CREATE TABLE Products (

ProductID INT AUTO\_INCREMENT PRIMARY KEY,

ProductName VARCHAR(100),

Description TEXT,

Price DECIMAL(10, 2)

);

CREATE TABLE Orders (

OrderID INT AUTO\_INCREMENT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

TotalAmount DECIMAL(10, 2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderDetails (

OrderDetailID INT AUTO\_INCREMENT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

CREATE TABLE Inventory (

InventoryID INT AUTO\_INCREMENT PRIMARY KEY,

ProductID INT,

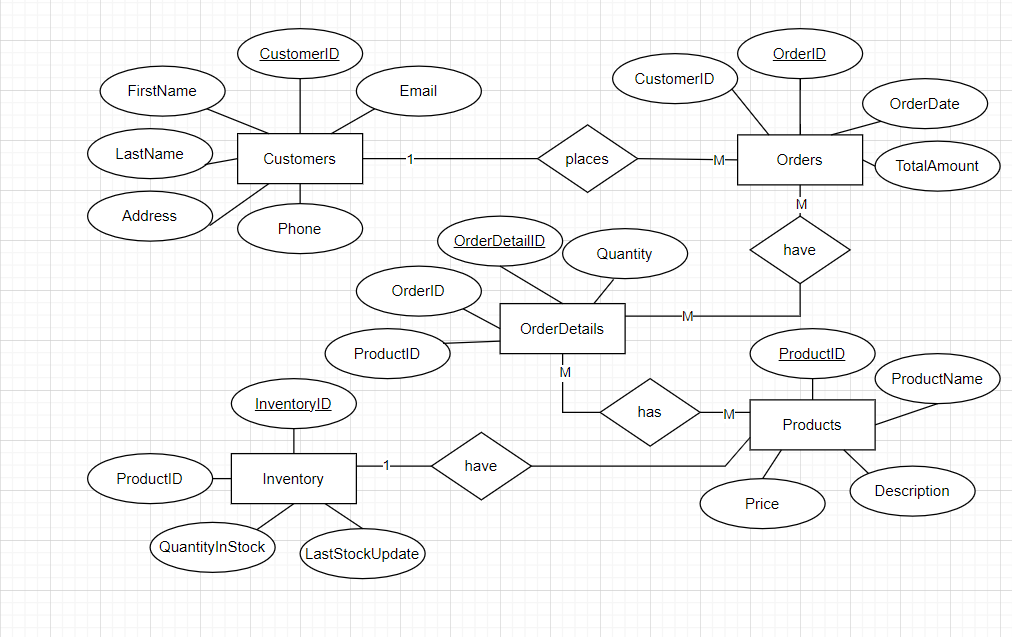
QuantityInStock INT,

LastStockUpdate TIMESTAMP,

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

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



1. **Create appropriate Primary Key and Foreign Key constraints for referential integrity.**

* Primary Key constraints are already defined in the table definition.
* Foreign Key constraints are also defined in the table definitions.

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

**a. Customers**

**b. Products**

**c. Orders**

**d. OrderDetails**

**e. Inventory**

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

VALUES

('Amy', 'Key', 'amykey@email.com', '123-456-7890', '1 Main St'),

('Ben', 'Rog', 'benrog@email.com', '888-654-3210', '4 Hill St'),

('Bob', 'Roy', 'bobroy@email.com', '555-123-4567', '7 Pine St'),

('Cal', 'Pan', 'calpan@email.com', '111-222-3333', '3 Elm St'),

('Dia', 'Deb', 'diadeb@email.com', '444-555-6666', '5 Lily St'),

('Eva', 'Fay', 'evafay@email.com', '777-888-9999', '9 Park St'),

('Jan', 'Ivy', 'janivy@email.com', '333-222-1111', '2 Moss St'),

('Kai', 'Kim', 'kaikim@email.com', '666-777-8888', '8 Rose St'),

('Liv', 'Mae', 'livmae@email.com', '999-888-7777', '17 Tree St'),

('Min', 'Lee', 'minlee@email.com', '222-333-4444', '41 Plan St');

INSERT INTO Products (ProductName, Description, Price)

VALUES

('Laptop', 'Powerful laptop with high performance', 1000),

('Smartphone', 'Latest model with advanced features', 499.99),

('Tablet', 'Lightweight and portable tablet', 200),

('Desktop', 'High-end desktop computer for gaming', 1500),

('Printer', 'Wireless printer with scanning capabilities', 300),

('Headphones', 'Noise-canceling headphones for immersive audio', 99.99),

('Camera', 'Digital camera with HD recording', 800),

('External Hard Drive', '1TB external hard drive for data storage', 700),

('Wireless Mouse', 'Ergonomic wireless mouse for improved productivity', 50),

('Monitor', 'Ultra-wide monitor for multitasking', 500);

INSERT INTO Orders (CustomerID, OrderDate, TotalAmount)

VALUES

(1, '2023-02-01', 1000),

(3, '2023-01-02', 499.99),

(5, '2023-01-03', 200),

(7, '2023-01-04', 1500),

(9, '2023-01-05', 300),

(2, '2023-01-06', 99.99),

(4, '2023-01-07', 800),

(6, '2023-01-08', 700),

(8, '2023-01-09', 50),

(10, '2023-01-10', 500);

INSERT INTO OrderDetails (OrderID, ProductID, Quantity)

VALUES

(1, 1, 2),

(3, 3, 1),

(5, 5, 3),

(7, 7, 1),

(9, 9, 2),

(2, 2, 1),

(4, 4, 1),

(6, 6, 1),

(8, 8, 2),

(10, 10, 1);

INSERT INTO Inventory (ProductID, QuantityInStock, LastStockUpdate)

VALUES

(1, 10, '2023-01-01 03:00:00'),

(2, 20, '2023-02-02 02:00:00'),

(3, 15, '2023-03-03 08:00:00'),

(4, 5, '2023-04-04 10:00:00'),

(5, 25, '2023-05-05 07:00:00'),

(6, 30, '2023-06-06 09:00:00'),

(7, 8, '2023-07-07 11:00:00'),

(8, 12, '2023-08-08 04:00:00'),

(9, 18, '2023-09-09 05:00:00'),

(10, 22, '2023-10-10 06:00:00');

**Tasks 2: 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;

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

SELECT o.OrderID, o.OrderDate, c.FirstName, c.LastName

FROM Orders o

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

1. **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 ('May', 'day', 'mayday@gmail.com', '987-654-3210', '45 coco St');

1. **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

WHERE Description LIKE '%electronic gadget%';

1. **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 = @OrderIDToDelete;

DELETE FROM Orders WHERE OrderID = @OrderIDToDelete;

1. **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, '2023-02-01', 1299.99);

1. **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.email@example.com', Address = '789 Maple St'

WHERE CustomerID = @CustomerIDToUpdate;

1. **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

SET TotalAmount = (

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

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

WHERE od.OrderID = Orders.OrderID

);

1. **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 = @CustomerIDToDelete);

DELETE FROM Orders WHERE CustomerID = @CustomerIDToDelete;

1. **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.**

ALTER TABLE Products

ADD COLUMN Category VARCHAR(255);

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

VALUES ('New Gadget', 'Description of the new gadget', 399.99, 'Electronic Gadgets');

1. **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 COLUMN Status VARCHAR(255);

UPDATE Orders SET Status = 'Shipped' WHERE OrderID = @OrderIDToUpdate;

1. **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.**

ALTER TABLE Customers

ADD COLUMN OrderCount INT;

UPDATE Customers

SET OrderCount = (

SELECT COUNT(\*) FROM Orders WHERE CustomerID = Customers.CustomerID

);

**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.**

SELECT o.OrderID, o.OrderDate, c.FirstName, c.LastName

FROM Orders o

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

1. **Write an SQL query to 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;

1. **Write an SQL query to list all customers who have made at least**

**one purchase. Include their names and contact information.**

SELECT c.FirstName, c.LastName, c.Email

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID

HAVING COUNT(o.OrderID) > 0;

1. **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, SUM(od.Quantity) AS TotalQuantityOrdered

FROM Products p

JOIN OrderDetails od ON p.ProductID = od.ProductID

WHERE p.Description LIKE '%electronic gadget%'

GROUP BY p.ProductName

ORDER BY TotalQuantityOrdered DESC

LIMIT 1;

1. **Write an SQL query to retrieve a list of electronic gadgets along**

**with their corresponding categories.**

SELECT p.ProductName, p.Category

FROM Products p;

1. **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(o.TotalAmount) AS AvgOrderValue

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

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

1. **Write an SQL query to 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

JOIN Customers c ON o.CustomerID = c.CustomerID

ORDER BY o.TotalAmount DESC

LIMIT 1;

1. **Write an SQL query to list electronic gadgets and the number of**

**times each product has been ordered.**

SELECT p.ProductName, COUNT(od.OrderDetailID) AS OrderCount

FROM Products p

LEFT JOIN OrderDetails od ON p.ProductID = od.ProductID

GROUP BY p.ProductName;

1. **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 DISTINCT c.FirstName, c.LastName

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 = @ProductNameParam;

1. **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.**

SELECT SUM(o.TotalAmount) AS TotalRevenue

FROM Orders o

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 \* FROM Customers

WHERE CustomerID NOT IN (SELECT CustomerID FROM Orders);

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

SELECT COUNT(\*) FROM Products;

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

SELECT SUM(TotalAmount) FROM Orders;

1. **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 AVG(od.Quantity) AS AvgQuantityOrdered

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

WHERE p.Category = @CategoryParam;

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

SELECT SUM(o.TotalAmount) AS TotalRevenue

FROM Orders o

WHERE o.CustomerID = @CustomerIDParam;

1. **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.FirstName, c.LastName, COUNT(\*) AS NumberOfOrders

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID

ORDER BY NumberOfOrders DESC;

1. **Write an SQL query to find the most popular product category, which is the one with the highest total quantity ordered across all orders.**

SELECT p.Description, SUM(od.Quantity) AS TotalQuantity

FROM Products p

JOIN OrderDetails od ON p.ProductID = od.ProductID

GROUP BY p.Description

ORDER BY TotalQuantity DESC

LIMIT 1;

1. **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.FirstName, c.LastName, SUM(o.TotalAmount) AS TotalSpending

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.FirstName, c.LastName

ORDER BY TotalSpending DESC

LIMIT 1;

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

SELECT AVG(TotalAmount) AS AvgOrderValue FROM Orders;

1. **Write an SQL query to 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.FirstName, c.LastName;