SQL ASSIGNMENT

**By**: **CHOUTAKURI LEENA**

**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

1. Products:

• ProductID (Primary Key)

• ProductName

• Description

• Price

1. Orders:

• OrderID (Primary Key)

• CustomerID (Foreign Key referencing Customers)

• OrderDate

• TotalAmount

1. OrderDetails:

• OrderDetailID (Primary Key)

• OrderID (Foreign Key referencing Orders)

• ProductID (Foreign Key referencing Products)

• Quantity

1. 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;

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

CREATE TABLE Customers (

CustomerID int PRIMARY KEY,

Firstname varchar(30) NOT NULL,

Lastname varchar(30) NOT NULL,

Email varchar(40) NOT NULL,

Phone int NOT NULL,

Address varchar(100) DEFAULT NULL

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(25) NOT NULL,

DescriptionOfP VARCHAR(100) NOT NULL,

Price FLOAT NOT NULL

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

OrderDate DATE NOT NULL,

TotalAmount FLOAT NOT NULL,

CustomerID int NOT NULL,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderDetails(

OrderDetailID INT PRIMARY KEY,

OrderID INT NOT NULL,

ProductID INT NOT NULL,

Quantity INT NOT NULL,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

CREATE TABLE Inventory(

InventoryID INT PRIMARY KEY,

ProductID INT,

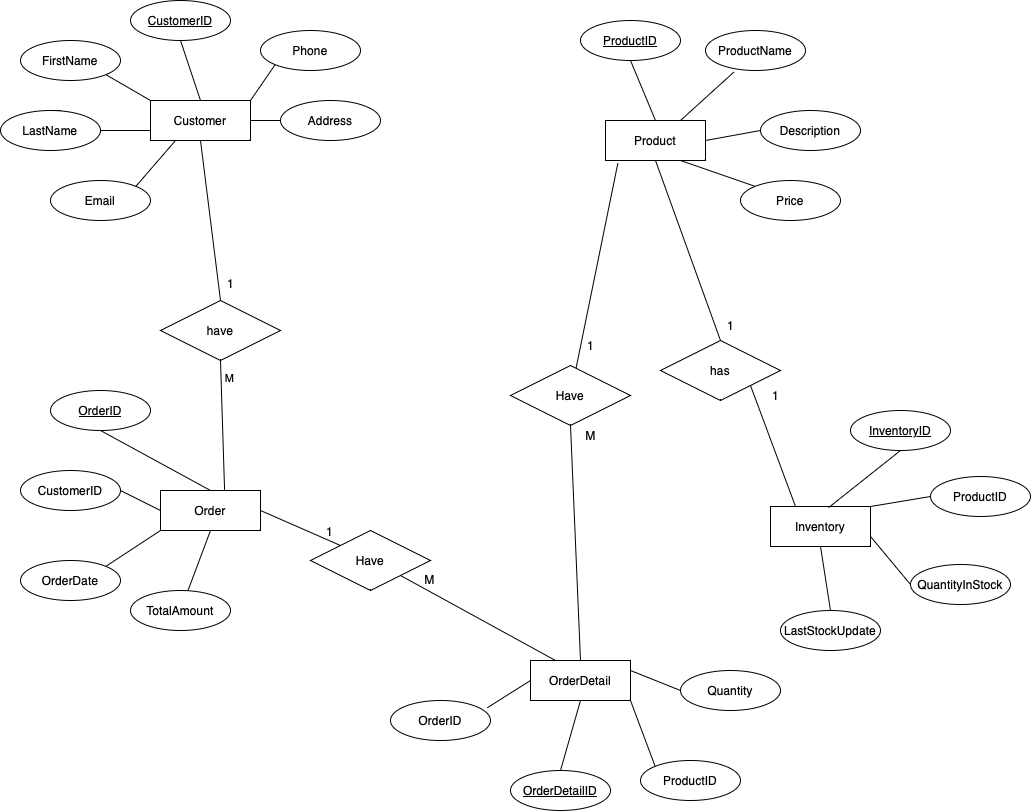
QuantityInStock INT,

LastStockUpdate DATE,

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

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



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

CREATE TABLE Customers (

CustomerID int PRIMARY KEY,

Firstname varchar(30) NOT NULL,

Lastname varchar(30) NOT NULL,

Email varchar(40) NOT NULL,

Phone int NOT NULL,

Address varchar(100) DEFAULT NULL

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(25) NOT NULL,

DescriptionOfP VARCHAR(100) NOT NULL,

Price FLOAT NOT NULL

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

OrderDate DATE NOT NULL,

TotalAmount FLOAT NOT NULL,

CustomerID int NOT NULL,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderDetails(

OrderDetailID INT PRIMARY KEY,

OrderID INT NOT NULL,

ProductID INT NOT NULL,

Quantity INT NOT NULL,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

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 at least 6 sample records into each of the following tables.

a. Customers b. Products c. Orders d. OrderDetails

INSERT INTO Customers VALUES

(1, 'Yuji', 'Itadori', 'yuji@gmail.com', 7669, '634 JP'),

(2, 'Gojo', 'Satoru', 'gojo@gmail.com', 5469, '345 Tokyo'),

(3, 'Anya', 'Forger', 'anya@gmail.com', 7509, '128 Park Avenue'),

(4, 'Eren', 'Jaeger', 'eren@gmail.com', 8945, 'Shiganshina'),

(5, 'Kugisaki', 'Nobara', 'nobara@gmail.com', 23578, '692 Tk'),

(6, 'Mikasa', 'Ackerman', 'mikasa@gmail.com', 14526, '423 Paradis');

INSERT INTO Products VALUES

(1, 'Laptop', 'High-performance laptop', 80000.00),

(2, 'Phone', 'Latest smartphone model', 28000.00),

(3, 'Tablet', 'Portable tablet device', 3000.00),

(4, 'Smartwatch', 'Fitness and health monitoring smartwatch', 2500.00),

(5, 'Wireless Earbuds', 'Bluetooth earbuds with noise cancellation', 1200.00),

(6, 'Gaming Console', 'High-performance gaming console', 5000.00);

INSERT INTO Orders VALUES

(1, '2023-12-01', 2000.00, 1),

(2, '2023-12-02', 1500.00, 2),

(3, '2023-12-03', 1000.00, 3),

(4, '2023-12-04', 1800.00, 4),

(5, '2023-12-05', 1200.00, 5),

(6, '2023-12-06', 900.00, 6);

INSERT INTO OrderDetails VALUES

-- OrderDetailID, OrderID, ProductID, Quantity.

(1, 1, 1, 2), -- Order 1: Laptop (2 units)

(2, 2, 2, 1), -- Order 2: Phone (1 unit)

(3, 3, 3, 3), -- Order 3: Tablet (3 units)

(4, 4, 4, 1), -- Order 4: Smartwatch (1 unit)

(5, 5, 5, 2), -- Order 5: Wireless Earbuds (2 units)

(6, 6, 6, 1); -- Order 6: Gaming Console (1 unit)

INSERT INTO OrderDetails VALUES

(7, 5, 4, 2);

INSERT INTO Inventory VALUES

-- InventoryID ProductID, QuantityInStock, LastStockUpdate

(346, 1, 20, '2023-12-01'),-- Inventory record for Laptop

(456, 2, 15, '2023-12-02'),-- Inventory record for Phone

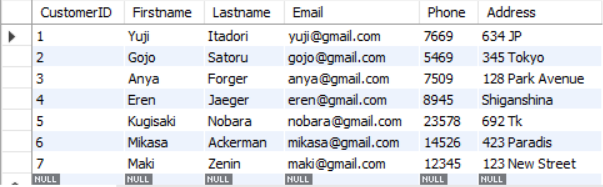
(556, 3, 25, '2023-12-03'),-- Inventory record for Tablet

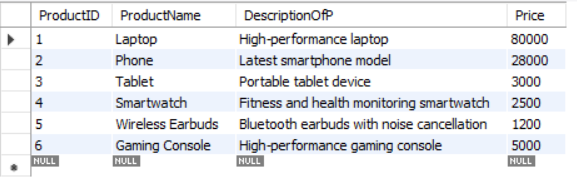
(656, 4, 30, '2023-12-04'), -- Inventory record for Smartwatch

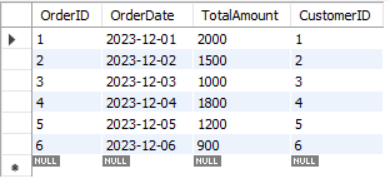
(756, 5, 25, '2023-12-05'), -- Inventory record for Wireless Earbuds

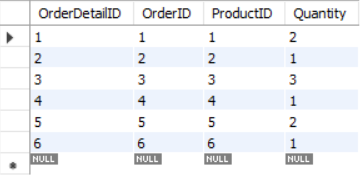
(856, 6, 20, '2023-12-06'); -- Inventory record for Gaming Console

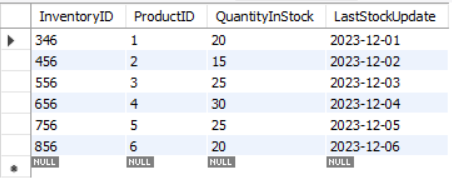
**TABLES:**











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

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

SELECT concat(FirstName,' ',LastName) as Name, Email From 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

INNER 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 (CustomerID, Firstname, Lastname, Email, Phone, Address)

VALUES (7, 'Maki', 'Zenin', 'maki@gmail.com', 12345, '123 New Street');

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 \* 1.1

WHERE ProductID IN (1, 2, 3, 4, 5, 6);

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 = 3;

DELETE FROM Orders WHERE OrderID = 3;

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 VALUES

(3, '2023-12-03', 1000.00, 3);

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 = 'itadori@gmail.com', Address = '225 Japan'

WHERE CustomerID = 1;

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 O, Orderdetails D

SET TotalAmount = (

SELECT SUM(price\*quantity) FROM OrderDetails D

JOIN Products P

ON P.productID = D.productID

WHERE D.orderID = O.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 = 1);

DELETE FROM Orders WHERE CustomerID = 1;

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.

INSERT INTO Products VALUES (7, 'Smart Speaker', 'Voice-controlled smart speaker', 3500.00);

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 Order\_Status VARCHAR(45);

UPDATE Orders SET Order\_Status='shipped' WHERE OrderID=2;

UPDATE Orders SET Order\_Status='pending' WHERE OrderID=3;

UPDATE Orders SET Order\_Status='pending' WHERE OrderID=4;

UPDATE Orders SET Order\_Status='shipped' WHERE OrderID=5;

UPDATE Orders SET Order\_Status='shipped' WHERE OrderID=6;

SET @orderid = 3;

SET @oder\_status ='shipped';

UPDATE Orders SET order\_status = @status WHERE orderID = @orderid;

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 ordercount INT;

UPDATE Customers

SET ordercount = (

SELECT COUNT(orderID) FROM Orders

WHERE Customers.customerID= Orders.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 C.first\_name, C.last\_name, O.\* 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.ProductID, ProductName, price\*SUM(quantity) AS TotalRevenue

FROM Products p

JOIN Orderdetails D

ON p.ProductID = D.ProductID

GROUP BY ProductID;

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

SELECT C.customerID, C.FirstName, C.LastName, C.Email

FROM CUSTOMERS C

JOIN ORDERS O

ON C.customerID = O.customerID

GROUP BY C.customerID;

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(D.quantity) AS Total\_Quantity

FROM PRODUCTS P

JOIN ORDERDETAILS D

ON P.productID = D.productID

GROUP BY P.productID

ORDER BY Total\_Quantity DESC LIMIT 1;

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

SELECT P.ProductName, C.CategoryName

FROM Products P

JOIN Categories C

ON P.categoryID = C.categoryID;

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 CONCAT(C.FirstName,' ',C.LastName) CustomName,

AVG(O.totalamount) AS AvgOrderValue

FROM Customers C JOIN Orders O

ON C.customerID = O.customerID

GROUP BY O.customerID;

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, FirstName, LastName, address, O.totalamount AS TotalRevenue

FROM ORDERS O JOIN CUSTOMERS C

ON O.customerID = C.customerID

ORDER BY TotalRevenue DESC LIMIT 1;

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

SELECT P.productID, P.ProductName, COUNT(D.orderDetailID) AS TotalOrders FROM Products P

JOIN OrderDetails D

ON P.productID = D.productID

GROUP BY productID;

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

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

INNER JOIN OrderDetails od ON o.OrderID = od.OrderID

INNER JOIN Products p ON od.ProductID = p.ProductID

WHERE p.ProductName = 'Phone';

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.

SET @startdate = '2023-12-01';

SET @enddate = '2023-12-06';

SELECT sum(totalamount)

FROM orders

WHERE orders.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 CONCAT(FirstName,' ',LastName) 'NameOfTheCustomer' FROM customers

WHERE customerid NOT IN (SELECT customerid FROM orders);

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

SELECT COUNT(\*) as "Total Products"

FROM Inventory

WHERE QuantityInStock>0;

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

SELECT SUM(totalamount) 'Total\_Revenue' FROM orders;

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.

SET @cname = 'Phones';

SELECT AVG(quantity) FROM OrderDetails

WHERE productid IN (

SELECT productid FROM Products WHERE categoryid = (

SELECT categoryid FROM Categories WHERE categoryname = @cname

)

);

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.

SET @customerid = 5;

SELECT customerID, SUM(totalamount) AS Total\_Revenue

FROM orders

WHERE customerID = @customerid

GROUP BY customerID;

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 firstname, cnt

FROM customers, (

SELECT customerid, COUNT(orderid) AS 'cnt'

FROM orders

GROUP BY orders.customerid

) AS t

WHERE t.customerid = customers.customerid

AND cnt = (

SELECT MAX(cnt)

FROM (

SELECT COUNT(orderid) AS 'cnt'

FROM orders

GROUP BY orders.customerid

) AS c

)

GROUP BY t.customerid;

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 C.categoryname, SUM(D.quantity) AS TotalQuantity

FROM OrderDetails D, Products P, Categories C

WHERE P.productID = D.productID

AND C.categoryID = P.categoryID

GROUP BY C.categoryID, C.categoryname

ORDER BY TotalQuantity DESC

LIMIT 1;

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.

SELECT O.CustomerID,

(

SELECT CONCAT(C.FirstName, ' ', C.Lastname) FROM Customers C

WHERE C.CustomerID = O.CustomerID

) AS 'Customer Name',

(

SELECT SUM(totalamount) FROM Orders

WHERE CustomerID = O.CustomerID

) AS TotalSpending

FROM Orders O

WHERE O.CustomerID IN (

SELECT CustomerID FROM OrderDetails

)

GROUP BY O.CustomerID

ORDER BY TotalSpending DESC

LIMIT 1;

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

SELECT customerID,

(

SELECT CONCAT(C.firstname, ' ', C.lastname) FROM customers C

WHERE C.customerid = O.customerid

) AS "Customer Name",

SUM(O.totalamount) / COUNT(O.orderid) AS "Avg Order Value"

FROM Orders O

GROUP BY O.customerid;

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.

SELECT O.customerID,

CONCAT(

(SELECT C.firstname FROM customers C WHERE C.customerid = O.customerid),

' ',

(SELECT C.lastname FROM customers C WHERE C.customerid = O.customerid)

) AS "Customer Name",

COUNT(O.orderid) AS "Order Count"

FROM orders O

GROUP BY O.customerID;