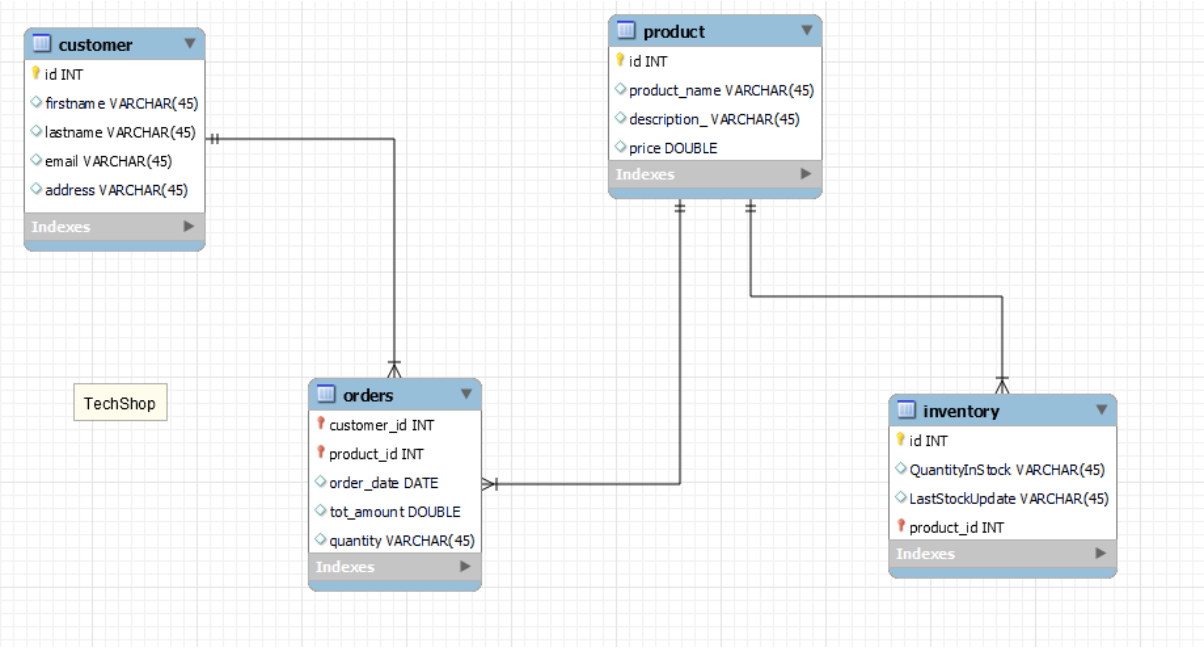
TECH SHOP




```
create database Techshop;
use Techshop;

-- Create the 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(20),
  Address VARCHAR(255)
);

-- Create the Products table
CREATE TABLE Products (
  ProductID INT PRIMARY KEY AUTO_INCREMENT,
  ProductName VARCHAR(100) NOT NULL,
```

```
  Description_ TEXT,
  Price DECIMAL(10,2) NOT NULL
);

-- Create the Orders table
CREATE TABLE Orders (
  OrderID INT PRIMARY KEY AUTO_INCREMENT,
  CustomerID INT NOT NULL,
  OrderDate DATE NOT NULL,
  TotalAmount DECIMAL(10,2) NOT NULL,
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

-- Create the OrderDetails table
CREATE TABLE OrderDetails (
  OrderDetailID INT PRIMARY KEY AUTO_INCREMENT,
  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 the Inventory table
CREATE TABLE Inventory (
  InventoryID INT PRIMARY KEY AUTO_INCREMENT,
  ProductID INT NOT NULL,
  QuantityInStock INT NOT NULL DEFAULT 0,
  LastStockUpdate VARCHAR(50) DEFAULT NULL,
```

```
  FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);

INSERT INTO Customers (FirstName, LastName, Email, Phone, Address)
VALUES ('John', 'Doe', 'john.doe@email.com', '123-456-7890', '123 Main Street'),
    ('Jane', 'Smith', 'jane.smith@email.com', '555-123-4567', '456 Elm Street'),
    ('Michael', 'Lee', 'michael.lee@email.com', '987-654-3210', '789 Oak Avenue'),
    ('Olivia', 'Jones', 'olivia.jones@email.com', '234-567-8901', '1011 Beach Road'),
    ('William', 'Johnson', 'william.johnson@email.com', '876-012-3456', '1213 Maple Lane');

INSERT INTO Products (ProductName, Description_, Price)
VALUES ('Wireless Keyboard', 'Compact and ergonomic wireless keyboard', 39.99),
    ('Gaming Mouse', 'High-precision gaming mouse with customizable RGB lighting', 69.99),
    ('Laptop Stand', 'Adjustable laptop stand for improved posture', 24.99),
    ('Portable Charger', 'High-capacity power bank for charging your devices on the go',
29.99),
    ('Noise-Cancelling Headphones', 'Wireless headphones with active noise cancellation for
immersive audio', 199.99);


INSERT INTO Orders (CustomerID, OrderDate, TotalAmount)
VALUES (1, '2024-02-15', 29.99),
    (2, '2024-01-28', 19.98),
    (3, '2023-12-10', 44.97),
    (4, '2023-11-03', 159.98),
    (5, '2023-10-20', 39.99);

INSERT INTO OrderDetails (OrderID, ProductID, Quantity)
VALUES (1, 1, 2),
    (2, 2, 1),
```

```
    (3, 1, 1),
    (3, 3, 2),
    (4, 4, 1);

INSERT INTO Inventory (ProductID, QuantityInStock, LastStockUpdate)
VALUES (1, 10, 'Manual stock update on 2024-03-08'),
    (2, 20, 'Automatic stock update from sales data'),
    (3, 50, NULL),
    (4, 15, 'Stock adjusted after damaged items removed'),
    (5, 3, 'Initial stock added');

-- 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;

-- 2. 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;

-- INSERT INTO Customers (FirstName, LastName, Email, Address)
 INSERT INTO Customers (FirstName, LastName, Email, Address) VALUES ('maa', 'maduraa', 'madurai@gmail.com', 'madurai thoppu');

 -- 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*1.1;
Where Productname like '%elec%';
```

```
-- 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 o.*

FROM Orders o

WHERE o.OrderID IN (

  SELECT OrderID

  FROM OrderDetails

  WHERE OrderID = 3

);


-- 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 (3, '2023-07-05', 100.00);


-- 7. Write an SQL query to update the contact information (e.g., email and address) of a specific customer in the "Customers" table.

UPDATE Customers

SET Email = 'newmail@gmail.com', Address = 'new found addresss'

WHERE CustomerID = 2;


-- 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 o.TotalAmount = (

  SELECT SUM(od.Price * od.Quantity)

  FROM OrderDetails od

  WHERE od.OrderID = o.OrderID

);
```

-- 9. Write an SQL query to delete all orders and their associated order details for a specific customer from the "Orders" and "OrderDetails" tables.

```
DELETE o

FROM Orders o

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

WHERE o.CustomerID = 2;
```

-- 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 ('Wireless mouse', 'Compact and ergonomic wireless mouse', 35.99);
```

-- 11. Write an SQL query to update the status of a specific order in the "Orders" table (e.g., from "Pending" to "Shipped").

```
UPDATE Orders

SET OrderStatus = 'shipped'

WHERE OrderID = 4;
```

-- 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.

-- TASK 3 --

-- 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, o.TotalAmount, c.firstName

FROM Orders o

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

-- 2. 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(o.totalamount * od.Quantity) AS TotalRevenue

FROM Products p

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

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

GROUP BY p.ProductID, p.ProductName;
```

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

```
SELECT DISTINCT c.CustomerID, c.FirstName, c.email

FROM Customers c

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

WHERE o.OrderID IS NOT NULL;
```

-- 4. 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

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

GROUP BY p.ProductID, p.ProductName

ORDER BY TotalQuantityOrdered DESC

LIMIT 1;
```

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

```
SELECT DISTINCT ProductName, Description_

FROM Products;
```

-- 6. 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.FirstName, AVG(o.TotalAmount / od.Quantity) AS AverageOrderValue

FROM Customers c

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

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

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

-- 7. 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.email, SUM(o.TotalAmount * od.Quantity) AS
TotalRevenue

FROM Orders o

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

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

GROUP BY o.OrderID, c.CustomerID, c.FirstName, c.email

ORDER BY TotalRevenue DESC

LIMIT 1;
```

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

```
SELECT p.ProductName, SUM(od.Quantity) AS NumTimesOrdered

FROM Products p

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

GROUP BY p.ProductID, p.ProductName;
```

-- 9. 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 c.CustomerID, c.FirstName, c.email

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 = 'Gaming mouse';


-- 10. Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period.

SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders

WHERE OrderDate BETWEEN '2024-01-01' AND '2024-12-12';


-- TASK 4 --

-- 1. Write an SQL query to find out which customers have not placed any orders.

SELECT c.CustomerID, c.FirstName

FROM Customers c

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

WHERE o.OrderID IS NULL;


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

SELECT Distinct COUNT(*) AS TotalProducts

FROM Products;


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

SELECT SUM(o.totalamount * od.Quantity) AS TotalRevenue

FROM Orders o

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


-- 4. Write an SQL query to calculate the average quantity ordered for products in a specific category.

SELECT p. productname,p.description_ AS Category,AVG(od.Quantity) AS AverageQuantity

FROM Products p
```

```
INNER JOIN OrderDetails od ON p.ProductID = od.ProductID
GROUP BY p.productname;

-- 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.
SELECT c.CustomerID, c.FirstName,
    (SELECT SUM(o.TotalAmount)
     FROM Orders o
     WHERE o.CustomerID = c.CustomerID) AS TotalRevenue
FROM Customers c;

-- 6. 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.CustomerID, c.FirstName,
(SELECT COUNT(*) FROM Orders o WHERE o.CustomerID = c.CustomerID) AS NumOrders
FROM Customers c
ORDER BY NumOrders DESC;

-- 7. 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.ProductName AS Popular_Product, SUM(od.Quantity) AS TotalQuantityOrdered
FROM Products p
INNER JOIN OrderDetails od ON p.ProductID = od.ProductID
GROUP BY p.ProductID, p.ProductName
ORDER BY TotalQuantityOrdered DESC
LIMIT 1;
```