**Step 1: Create Database**

CREATE DATABASE e\_commerce;

USE e\_commerce;

**Step 2: Create Tables**

Here’s a simplified structure covering Customers, Orders, Products, OrderItems, Suppliers, Categories, Employees, and Addresses.

-- Customers

CREATE TABLE Customers (

CustomerID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(100),

Email VARCHAR(100),

City VARCHAR(50)

);

-- Addresses (for detailed location)

CREATE TABLE Addresses (

AddressID INT AUTO\_INCREMENT PRIMARY KEY,

CustomerID INT,

AddressLine VARCHAR(255),

City VARCHAR(50),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

-- Suppliers

CREATE TABLE Suppliers (

SupplierID INT AUTO\_INCREMENT PRIMARY KEY,

SupplierName VARCHAR(100)

);

-- Categories

CREATE TABLE Categories (

CategoryID INT AUTO\_INCREMENT PRIMARY KEY,

CategoryName VARCHAR(100)

);

-- Products

CREATE TABLE Products (

ProductID INT AUTO\_INCREMENT PRIMARY KEY,

ProductName VARCHAR(100),

Price DECIMAL(10,2),

CategoryID INT,

SupplierID INT,

IsActive BOOLEAN DEFAULT TRUE,

Rating DECIMAL(2,1),

FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID),

FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)

);

-- Orders

CREATE TABLE Orders (

OrderID INT AUTO\_INCREMENT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

ProcessedByEmployeeID INT,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

-- OrderItems

CREATE TABLE OrderItems (

OrderItemID INT AUTO\_INCREMENT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

-- Employees

CREATE TABLE Employees (

EmployeeID INT AUTO\_INCREMENT PRIMARY KEY,

EmployeeName VARCHAR(100)

);

**Step 3: Insert Sample Data**

-- Customers

INSERT INTO Customers (Name, Email, City) VALUES

('Alice', 'alice@example.com', 'Chennai'),

('Bob', 'bob@example.com', 'Mumbai'),

('Charlie', 'charlie@example.com', 'Chennai');

-- Addresses

INSERT INTO Addresses (CustomerID, AddressLine, City) VALUES

(1, '123 Street, Chennai', 'Chennai'),

(2, '456 Avenue, Mumbai', 'Mumbai'),

(3, '789 Road, Chennai', 'Chennai');

-- Suppliers

INSERT INTO Suppliers (SupplierName) VALUES

('Supplier A'),

('Supplier B'),

('Supplier C');

-- Categories

INSERT INTO Categories (CategoryName) VALUES

('Electronics'),

('Clothing');

-- Products

INSERT INTO Products (ProductName, Price, CategoryID, SupplierID, IsActive, Rating) VALUES

('Laptop', 1000.00, 1, 1, TRUE, 4.5),

('Smartphone', 700.00, 1, 2, TRUE, 4.2),

('T-Shirt', 20.00, 2, 3, FALSE, 3.8),

('Jeans', 50.00, 2, 3, TRUE, 4.0);

-- Employees

INSERT INTO Employees (EmployeeName) VALUES

('John Doe'),

('Jane Smith');

-- Orders

INSERT INTO Orders (CustomerID, OrderDate, ProcessedByEmployeeID) VALUES

(1, '2025-01-15', 1),

(2, '2024-12-10', 2),

(1, '2025-05-01', 1);

-- OrderItems

INSERT INTO OrderItems (OrderID, ProductID, Quantity) VALUES

(1, 1, 1),

(1, 2, 2),

(2, 3, 5),

(3, 4, 1);

**Step 4: Now, run the queries for the exercises.**

**1. Customers whose total purchase amount is greater than average purchase amount**

SELECT c.CustomerID, c.Name,

SUM(p.Price \* oi.Quantity) AS TotalPurchase

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderItems oi ON o.OrderID = oi.OrderID

JOIN Products p ON oi.ProductID = p.ProductID

GROUP BY c.CustomerID

HAVING TotalPurchase > (

SELECT AVG(CustomerTotal) FROM (

SELECT SUM(p2.Price \* oi2.Quantity) AS CustomerTotal

FROM Customers c2

JOIN Orders o2 ON c2.CustomerID = o2.CustomerID

JOIN OrderItems oi2 ON o2.OrderID = oi2.OrderID

JOIN Products p2 ON oi2.ProductID = p2.ProductID

GROUP BY c2.CustomerID

) AS sub

);

**2. Products whose price is higher than average price in the same category**

SELECT p1.ProductName, p1.Price, c.CategoryName

FROM Products p1

JOIN Categories c ON p1.CategoryID = c.CategoryID

WHERE p1.Price > (

SELECT AVG(p2.Price)

FROM Products p2

WHERE p2.CategoryID = p1.CategoryID

);

**3. Suppliers who do not supply any active products**

SELECT s.SupplierID, s.SupplierName

FROM Suppliers s

LEFT JOIN Products p ON s.SupplierID = p.SupplierID AND p.IsActive = TRUE

WHERE p.ProductID IS NULL;

**4. Customers who haven’t placed any orders in the last 6 months**

SELECT c.CustomerID, c.Name

FROM Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID AND o.OrderDate >= DATE\_SUB(CURDATE(), INTERVAL 6 MONTH)

WHERE o.OrderID IS NULL;

**5. Category with the highest average product rating**

SELECT c.CategoryName, AVG(p.Rating) AS AvgRating

FROM Categories c

JOIN Products p ON c.CategoryID = p.CategoryID

GROUP BY c.CategoryID

ORDER BY AvgRating DESC

LIMIT 1;

**6. Join Orders and Customers to get order + customer details**

SELECT o.OrderID, o.OrderDate, c.Name, c.Email, c.City

FROM Orders o

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

**7. Products and OrderItems to show product names and quantity sold**

SELECT oi.OrderID, p.ProductName, oi.Quantity

FROM OrderItems oi

JOIN Products p ON oi.ProductID = p.ProductID;

**8. Products and Suppliers for supplier name per product**

SELECT p.ProductName, s.SupplierName

FROM Products p

JOIN Suppliers s ON p.SupplierID = s.SupplierID;

**9. Orders placed by customers from "Chennai"**

SELECT o.OrderID, c.Name, a.City, o.OrderDate

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN Addresses a ON c.CustomerID = a.CustomerID

WHERE a.City = 'Chennai';

**10. Employees and Orders to find staff processing each order**

SELECT o.OrderID, e.EmployeeName

FROM Orders o

JOIN Employees e ON o.ProcessedByEmployeeID = e.EmployeeID;

**11. Product categories with total number of products**

SELECT c.CategoryName, COUNT(p.ProductID) AS TotalProducts

FROM Categories c

LEFT JOIN Products p ON c.CategoryID = p.CategoryID

GROUP BY c.CategoryID;

**12. Customers whose total order value exceeds average order value in their city**

SELECT c.CustomerID, c.Name, c.City, SUM(p.Price \* oi.Quantity) AS TotalOrderValue

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderItems oi ON o.OrderID = oi.OrderID

JOIN Products p ON oi.ProductID = p.ProductID

GROUP BY c.CustomerID, c.City

HAVING TotalOrderValue > (

SELECT AVG(CustomerTotal)

FROM (

SELECT c2.City, SUM(p2.Price \* oi2.Quantity) AS CustomerTotal

FROM Customers c2

JOIN Orders o2 ON c2.CustomerID = o2.CustomerID

JOIN OrderItems oi2 ON o2.OrderID = oi2.OrderID

JOIN Products p2 ON oi2.ProductID = p2.ProductID

GROUP BY c2.CustomerID

HAVING c2.City = c.City

) AS sub

);

If you want, I can help you step by step with the insertions or queries or explain any part more deeply. How do you want to proceed?