

QUESTION 1:- Display the name of all emp who are working in department no 10

```
CREATE TABLE EMP (
```

```
    EID INT PRIMARY KEY,
```

```
    EMP_NAME VARCHAR(50),
```

```
    SALARY DECIMAL(10, 2),
```

```
    DESIGNATION VARCHAR(30),
```

```
    DEPTNO INT
```

```
);
```

```
INSERT INTO EMP (EID, EMP_NAME, SALARY, DESIGNATION, DEPTNO) VALUES
```

```
(1, 'Alice', 3200.00, 'CLERK', 10),    -- Clerk, salary > 3000, 5-letter name, dept 10
```

```
(2, 'Bruce', 2800.00, 'CLERK', 20),    -- Clerk, but salary < 3000
```

```
(3, 'Cathy', 4000.00, 'MANAGER', 10),  -- Manager, dept 10
```

```
(4, 'David', 4500.00, 'CLERK', 30),    -- Clerk, salary > 3000, 5-letter name
```

```
(5, 'Ellen', 3800.00, 'ANALYST', 40),  -- 5-letter name
```

```
(6, 'Frank', 2200.00, 'CLERK', 10),    -- Clerk, dept 10, but salary < 3000
```

```
(7, 'Grace', 5100.00, 'MANAGER', 20),  -- High salary
```

```
(8, 'Hitesh', 3000.00, 'CLERK', 10),   -- Clerk, salary = 3000
```

```
(9, 'John', 3500.00, 'CLERK', 10),    -- Clerk, salary > 3000, 4-letter name
```

```
(10, 'Ritesh', 4100.00, 'ANALYST', 20); -- Salary > average
```

```
SELECT EMP_NAME
```

```
FROM EMP
```

```
WHERE DEPTNO = 10;
```

Display the name of all emp working as clerk and drawing a salary more than 3000

```
SELECT EMP_NAME
```

```
FROM EMP
```

```
WHERE DESIGNATION = 'CLERK'
```

AND SALARY > 3000;

Display the name of emp whose name is exactly 5 character in length

SELECT EMP_NAME

FROM EMP

WHERE LENGTH(EMP_NAME) = 5;

Write query will return the top n record using top command

SELECT * FROM EMP LIMIT 4;

Emp who earn more than avg salary of all emp of their company

SELECT EMP_NAME

FROM EMP

WHERE SALARY > (SELECT AVG(SALARY) FROM EMP);

QUESTION 2: -

-- EMPLOYEE TABLE

CREATE TABLE EMP(

emp_id INT PRIMARY KEY,

emp_name VARCHAR(50),

emp_city VARCHAR(50),

salary DECIMAL(10,2),

company_name VARCHAR(50),

street VARCHAR(50) -- assuming for Q3

);

INSERT INTO EMP (emp_id, emp_name, emp_city, salary, company_name, street)

VALUES

(1, 'John', 'New York', 70000.00, 'IBM', 'Main Street'), -- same city/street as manager
(Alice)

(2, 'Emily', 'San Francisco', 75000.00, 'Google', 'Ocean Ave'), -- same city as company

(3, 'Raj', 'Pune', 50000.00, 'Infosys', 'Church Road'), -- same city/street as manager
(Bob)

(4, 'Priya', 'Pune', 55000.00, 'Infosys', 'MG Road'),

(5, 'Karan', 'Mumbai', 80000.00, 'TCS', 'Hill Road'), -- same city/street as manager
(Charlie)

(6, 'Neha', 'Mumbai', 45000.00, 'TCS', 'Marine Drive'),

(7, 'Sara', 'Delhi', 90000.00, 'IBM', 'Ring Road'), -- different city

(8, 'Tom', 'New York', 30000.00, 'IBM', 'Main Street'); -- lower salary (IBM)

-- COMPANY TABLE

CREATE TABLE COMPANY(

company_name VARCHAR(50) PRIMARY KEY,

company_city VARCHAR(50)

);

INSERT INTO COMPANY (company_name, company_city) VALUES

('IBM', 'New York'),

('Google', 'San Francisco'),

('Infosys', 'Pune'),

('TCS', 'Mumbai');

-- MANAGER TABLE

CREATE TABLE MANAGER(

manager_id INT PRIMARY KEY,

manager_name VARCHAR(50),

street VARCHAR(50),

city VARCHAR(50) -- assuming to match with employee for Q3

);

INSERT INTO MANAGER (manager_id, manager_name, street, city) VALUES

(1, 'Alice', 'Main Street', 'New York'),

(2, 'Bob', 'Church Road', 'Pune'),

(3, 'Charlie', 'Hill Road', 'Mumbai');

--1. Find the names of all employees who work for IBM.

```
SELECT emp_name
FROM EMP
WHERE company_name = 'IBM';
```

--2. Find all employees who live in the same cities as the companies they work for.

```
SELECT E.emp_name
FROM EMP E
JOIN COMPANY C ON E.company_name = C.company_name
WHERE E.emp_city = C.company_city;
```

--3. Find all employees who live in the same cities and on the same streets as their managers.

```
SELECT E.emp_name
FROM EMP E
JOIN MANAGER M ON E.emp_city = M.city AND E.street = M.street;
```

--4. Find all employees who earn more than the average salary of all employees of their company.

```
SELECT E.emp_name
FROM EMP E
WHERE E.salary > (
    SELECT AVG(salary)
    FROM EMP
    WHERE company_name = E.company_name
);
```

--5. Find the company that has the smallest payroll (sum of salaries).

```
SELECT company_name
FROM EMP
GROUP BY company_name
ORDER BY SUM(salary) ASC
LIMIT 1;
```

QUESTION 3:-

-- Student Table

CREATE TABLE Student (

 stud_id INT PRIMARY KEY,

 stud_name VARCHAR(50),

 course VARCHAR(100)

);

INSERT INTO Student (stud_id, stud_name, course) VALUES

(1, 'Sana', 'Comp.Sci.'),

(2, 'Ravi', 'Physics'),

(3, 'Simran', 'Comp.Sci.'),

(4, 'John', 'Maths'),

(5, 'Neha', 'Comp.Sci.'),

(6, 'Pratu', 'Biology');

-- Instructor Table

CREATE TABLE Instructor (

 instructor_id INT PRIMARY KEY,

 name VARCHAR(50),

 department VARCHAR(50),

 salary DECIMAL(10,2)

);

INSERT INTO Instructor (instructor_id, name, department, salary) VALUES

(101, 'Anil', 'Comp.Sci.', 80000),

(102, 'Suresh', 'Physics', 75000),

(103, 'Salma', 'Maths', 60000),

(104, 'Sameer', 'Comp.Sci.', 90000),

```
(105, 'David', 'Biology', 70000);
```

-- Optional: Course_Offering Table (for query 2 - not originally listed but implied)

```
CREATE TABLE Course_Offering (
```

```
    course_id VARCHAR(20),
```

```
    stud_id INT,
```

```
    semester VARCHAR(20),
```

```
    FOREIGN KEY (stud_id) REFERENCES Student(stud_id)
```

```
);
```

```
INSERT INTO Course_Offering (course_id, stud_id, semester) VALUES
```

```
('CS101', 1, 'Fall 2008'),
```

```
('PHY202', 2, 'Spring 2009'),
```

```
('CS102', 3, 'Spring 2009'),
```

```
('MTH111', 4, 'Summer 2010'),
```

```
('BIO101', 6, 'Spring 2010');
```

--1. Find the names of all students who have taken at least one Comp.Sci. course. No duplicates.

```
SELECT DISTINCT stud_name
```

```
FROM Student
```

```
WHERE course = 'Comp.Sci.';
```

--2. Find the IDs and names of all students who have not taken any course offering before Spring 2009.

```
SELECT S.stud_id, S.stud_name
```

```
FROM Student S
```

```
WHERE S.stud_id NOT IN (
```

```
    SELECT stud_id
```

```
    FROM Course_Offering
```

```
    WHERE semester < 'Spring 2009'
```

```
);
```

--3. For each department, find the maximum salary of instructors in that department.

```
SELECT department, MAX(salary) AS max_salary
```

```
FROM Instructor
```

```
GROUP BY department;
```

--4. Find the lowest, across all departments, of the per-department maximum salary.

```
SELECT MIN(max_salary) AS lowest_max_salary
```

```
FROM (
```

```
    SELECT MAX(salary) AS max_salary
```

```
    FROM Instructor
```

```
    GROUP BY department
```

```
) AS dept_max_salaries;
```

--5. Find the names of employees (instructors) whose names start with 'S'.

```
SELECT name
```

```
FROM Instructor
```

```
WHERE name LIKE 'S%';
```

QUESTION 4:-

-- EMPLOYEE TABLE

```
CREATE TABLE EMP (
```

```
    emp_id INT PRIMARY KEY,
```

```
    emp_name VARCHAR(50),
```

```
    emp_city VARCHAR(50),
```

```
    salary DECIMAL(10,2),
```

```
    company_name VARCHAR(50),
```

```
    street VARCHAR(50)
```

```
);
```

```
INSERT INTO EMP (emp_id, emp_name, emp_city, salary, company_name, street)
```

```
VALUES
```

```
(1, 'John', 'New York', 70000.00, 'First Bank Corporation', 'Main Street'),
```

```
(2, 'Ravi', 'San Francisco', 65000.00, 'TechSoft', 'Ocean Ave'),  
(3, 'Neha', 'Pune', 50000.00, 'Infosys', 'Church Road'),  
(4, 'Amit', 'Mumbai', 40000.00, 'TCS', 'Hill Road'),  
(5, 'Sara', 'New York', 30000.00, 'First Bank Corporation', 'Ring Road'),  
(6, 'Raj', 'Pune', 45000.00, 'Infosys', 'MG Road');
```

-- COMPANY TABLE

```
CREATE TABLE COMPANY (  
    company_name VARCHAR(50) PRIMARY KEY,  
    company_city VARCHAR(50)  
);  
  
INSERT INTO COMPANY (company_name, company_city) VALUES  
( 'First Bank Corporation', 'New York'),  
( 'TechSoft', 'San Francisco'),  
( 'Infosys', 'Pune'),  
( 'TCS', 'Mumbai');
```

-- MANAGER TABLE

```
CREATE TABLE MANAGER (  
    manager_id INT PRIMARY KEY,  
    manager_name VARCHAR(50),  
    street VARCHAR(50),  
    city VARCHAR(50)  
);  
  
INSERT INTO MANAGER (manager_id, manager_name, street, city) VALUES  
(1, 'Alice', 'Main Street', 'New York'),  
(2, 'Bob', 'Church Road', 'Pune'),  
(3, 'Charlie', 'Hill Road', 'Mumbai');
```

--1. Find the names of all employees who work for First Bank Corporation.


```
SELECT emp_name  
FROM EMP  
WHERE company_name = 'First Bank Corporation';
```

--2. Find all employees who live in the same cities as the companies for which they work.

```
SELECT E.emp_name  
FROM EMP E  
JOIN COMPANY C ON E.company_name = C.company_name  
WHERE E.emp_city = C.company_city;
```

--3. Find all employees who live in the same cities and on the same streets as their managers.

```
SELECT E.emp_name  
FROM EMP E  
JOIN MANAGER M ON E.emp_city = M.city AND E.street = M.street;
```

--4. Find all employees who earn more than the average salary of all employees of their company.

```
SELECT E.emp_name  
FROM EMP E  
WHERE E.salary > (  
    SELECT AVG(salary)  
    FROM EMP  
    WHERE company_name = E.company_name  
);
```

--5. Find the company that has the highest payroll.

```
SELECT company_name  
FROM EMP  
GROUP BY company_name  
ORDER BY SUM(salary) DESC  
LIMIT 1;
```

QUESTION 5:-

```
CREATE TABLE Emp (  
    E_id INT PRIMARY KEY,  
    E_name VARCHAR(50),  
    Emp_dept VARCHAR(50)  
);
```

```
INSERT INTO Emp (E_id, E_name, Emp_dept) VALUES  
(101, 'Sneha', 'HR'),  
(102, 'Amit', 'IT'),  
(103, 'Priya', 'Finance'),  
(104, 'Neha', 'IT'),  
(105, 'Vikram', 'Marketing'),  
(106, 'Lata', 'HR');
```

```
CREATE TABLE Project (  
    Project_id INT PRIMARY KEY,  
    Project_name VARCHAR(50),  
    Project_start_date DATE,  
    Project_End_Date DATE  
);
```

```
INSERT INTO Project (Project_id, Project_name, Project_start_date, Project_End_Date)  
VALUES  
(1, 'Website Redesign', '2023-01-15', '2023-12-15'),  
(2, 'Mobile App', '2024-03-10', '2025-01-20'),  
(3, 'Data Migration', '2024-06-01', '2024-11-30');
```

```
CREATE TABLE Works_On (  
    E_id INT,
```

```
Project_id INT,  
FOREIGN KEY (E_id) REFERENCES Emp(E_id),  
FOREIGN KEY (Project_id) REFERENCES Project(Project_id)  
);
```

```
INSERT INTO Works_On (E_id, Project_id) VALUES
```

```
(101, 1),
```

```
(102, 1),
```

```
(103, 2),
```

```
(104, 2),
```

```
(105, 3),
```

```
(106, 1),
```

```
(104, 3); -- Neha works on both Project 2 and 3
```

1. List the number of departments of employees in each project.

sql

Copy code

```
SELECT
```

```
    w.Project_id,
```

```
    COUNT(DISTINCT e.Emp_dept) AS Department_Count
```

```
FROM
```

```
    Works_On w
```

```
JOIN
```

```
    Emp e ON w.E_id = e.E_id
```

```
GROUP BY
```

```
    w.Project_id;
```

2. (Duplicate of 1) List the number of departments of employees in each project.

(Same query as above.)

3. How many projects started and finished in the same year?

sql

Copy code

```
SELECT
    COUNT(*) AS Same_Year_Projects
FROM
    Project
WHERE
    YEAR(Project_start_date) = YEAR(Project_End_Date);
```

4. Select the department name of the company which is assigned to the employee whose employee ID is greater than 103.

sql

Copy code

```
SELECT
    DISTINCT Emp_dept
FROM
    Emp
WHERE
    E_id > 103;
```

5. List the names of all employees whose name ends with 'a'.

sql

Copy code

```
SELECT
    E_name
FROM
```

Emp

WHERE

E_name LIKE '%a';

QUESTION 6:-

CREATE TABLE Emp (

Empid INT PRIMARY KEY,

Emp_name VARCHAR(50),

Emp_salary DECIMAL(10, 2),

city VARCHAR(50),

Project VARCHAR(10)

);

INSERT INTO Emp (Empid, Emp_name, Emp_salary, city, Project) VALUES

(101, 'John', 10000, 'Toronto', 'P1'),

(102, 'Rita', 9500, 'Toronto', 'P1'),

(103, 'Amit', 12000, 'Delhi', 'P2'),

(104, 'Neha', 8000, 'Mumbai', 'P3'),

(105, 'Suresh', 15000, 'Toronto', 'P1');

-- MANAGER TABLE

CREATE TABLE Manager (

Manager_id INT,

Emp_name VARCHAR(50),

Project VARCHAR(10)

);

INSERT INTO Manager (Manager_id, Emp_name, Project) VALUES

(986, 'Arun', 'P1'),

(321, 'Meera', 'P3'),

(450, 'Karan', 'P2');

--1. Fetch the EmpId and FullName of employees working under Manager with ID '986'.

```
SELECT E.Empid, E.Emp_name  
FROM Emp E  
JOIN Manager M ON E.Project = M.Project  
WHERE M.Manager_id = 986;
```

--2. Fetch the count of employees working in project 'P1'.

```
SELECT COUNT(*) AS employee_count  
FROM Emp  
WHERE Project = 'P1';
```

--3. Find employee IDs whose salary lies between 9000 and 15000.

```
SELECT Empid  
FROM Emp  
WHERE Emp_salary BETWEEN 9000 AND 15000;
```

--4. Fetch employees who live in Toronto and work under manager with ID 321.

```
SELECT E.Empid, E.Emp_name  
FROM Emp E  
JOIN Manager M ON E.Project = M.Project  
WHERE E.city = 'Toronto' AND M.Manager_id = 321;
```

--5. Find the maximum, minimum, and average salary of employees.

```
SELECT  
    MAX(Emp_salary) AS MaxSalary,  
    MIN(Emp_salary) AS MinSalary,  
    AVG(Emp_salary) AS AvgSalary  
FROM Emp;
```

QUESTION 7:-

-- DROP the table if it already exists


```
DROP TABLE IF EXISTS Customer;
```

-- CREATE the table


```
CREATE TABLE Customer (  
    C_id INT PRIMARY KEY,  
    C_name VARCHAR(100),  
    C_city VARCHAR(100)  
);
```

-- INSERT sample data


```
INSERT INTO Customer (C_id, C_name, C_city) VALUES  
(1, 'Rahul', 'Mumbai'),  
(2, 'Anita', 'Chennai'),  
(3, 'Bairistow', 'London'),  
(4, 'Meena', 'Chennai'),  
(5, 'David', 'New York');
```

--  1. Delete records where city is 'Chennai'

```
DELETE FROM Customer  
WHERE C_city = 'Chennai';
```

--  2. UPDATE: change city to 'Oslo' before deleting the column

```
UPDATE Customer  
SET C_city = 'Oslo'  
WHERE C_name = 'Bairistow';
```


--  3. Delete the C_city column

```
ALTER TABLE Customer  
DROP COLUMN C_city; -- Only works in MySQL/PostgreSQL (NOT SQLite)
```

--  4. Add an Address column


```
ALTER TABLE Customer
```

```
ADD COLUMN Address VARCHAR(255);
```

--  5. Select where C_id = 12

```
SELECT * FROM Customer
```

```
WHERE C_id = 12;
```

--  6. Drop the Customer table

```
DROP TABLE Customer;
```

QUESTION 8:-

-- Create Employees table

```
CREATE TABLE Employees (
```

```
    EmpID INT PRIMARY KEY,
```

```
    EmpName VARCHAR(100),
```

```
    DeptID INT
```

```
);
```

```
INSERT INTO Employees (EmpID, EmpName, DeptID) VALUES
```

```
(1, 'Alice', 101),
```

```
(2, 'Bob', 102),
```

```
(3, 'Charlie', NULL),
```

```
(4, 'David', 103),
```

```
(5, 'Eve', 105);
```

```
CREATE TABLE Departments (
```

```
    DeptID INT PRIMARY KEY,
```




```
    DeptName VARCHAR(100)
);
INSERT INTO Departments (DeptID, DeptName) VALUES
(101, 'HR'),
(102, 'Finance'),
(103, 'IT'),
(104, 'Marketing');
```

--1. INNER JOIN

--Returns only matching rows from both tables.

```
SELECT E.EmpID, E.EmpName, D.DeptName
FROM Employees E
INNER JOIN Departments D ON E.DeptID = D.DeptID;
```

--  2. LEFT JOIN (a.k.a. LEFT OUTER JOIN)

```
SELECT E.EmpID, E.EmpName, D.DeptName
FROM Employees E
LEFT JOIN Departments D ON E.DeptID = D.DeptID;
```

--  5. CROSS JOIN

--Returns Cartesian product of both tables (every combination).

```
SELECT E.EmpName, D.DeptName
FROM Employees E
CROSS JOIN Departments D;
```

QUESTION 9:-

```
DROP TABLE Department;
CREATE TABLE Department (
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(100) UNIQUE NOT NULL
);
```

```
INSERT INTO Department (DeptID, DeptName) VALUES
```

```
(1, 'HR'),
```

```
(2, 'Finance'),
```

```
(3, 'IT');
```

```
CREATE TABLE Employee (
```

```
    EmpID INT PRIMARY KEY,          -- Primary Key
```

```
    EmpName VARCHAR(100) NOT NULL,  -- NOT NULL
```

```
    Email VARCHAR(100) UNIQUE,      -- UNIQUE
```

```
    Salary DECIMAL(10,2) CHECK (Salary > 5000), -- CHECK Constraint
```

```
    City VARCHAR(100) DEFAULT 'Mumbai', -- DEFAULT Value
```

```
    DeptID INT,
```

```
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID) -- FOREIGN KEY
```

```
);
```

```
INSERT INTO Employee (EmpID, EmpName, Email, Salary, DeptID)
```

```
VALUES
```

```
(101, 'Alice', 'alice@example.com', 7500.00, 1),
```

```
(102, 'Bob', 'bob@example.com', 12000.00, 2),
```

```
(103, 'Charlie', 'charlie@example.com', 9000.00, 3);
```

QUESTION 10:-

```
CREATE TABLE EMP (
```

```
    EID INT PRIMARY KEY,
```

```
    EMP_NAME VARCHAR(100),
```

```
    SALARY DECIMAL(10,2),
```

```
    DESIGNATION VARCHAR(50),
```

```
    DEPTNO INT
```

```
);
```

```
INSERT INTO EMP (EID, EMP_NAME, SALARY, DESIGNATION, DEPTNO) VALUES
```

```
(101, 'Sagar', 3500.00, 'Clerk', 10),
```

```
(102, 'Aarti', 5000.00, 'Manager', 20),
```

```
(103, 'Rohit', 2800.00, 'Clerk', 10),
```

```
(104, 'Neeta', 6000.00, 'Analyst', 30),
```

```
(105, 'Ramesh', 3200.00, 'Clerk', 10),
```

```
(106, 'Swara', 4200.00, 'Salesman', 40),
```

```
(107, 'Aisha', 7200.00, 'Manager', 20),
```

```
(108, 'Vikas', 2500.00, 'Clerk', 10),
```

```
(109, 'Radha', 6500.00, 'Analyst', 30),
```


```
(110, 'Sonya', 3800.00, 'Clerk', 10);
```

--1. Display the names of all employees who are working in department number 10:

```
SELECT EMP_NAME
```

```
FROM EMP
```

```
WHERE DEPTNO = 10;
```


--  2. Display the names of all employees working as clerks and drawing a salary more than 3000:

```
SELECT EMP_NAME
```

```
FROM EMP
```

```
WHERE DESIGNATION = 'Clerk'
```

```
AND SALARY > 3000;
```

--  3. Display the names of employees whose name is exactly five characters in length:

```
SELECT EMP_NAME
```

```
FROM EMP
```

```
WHERE LENGTH(EMP_NAME) = 5;
```


--For SQL Server, use LEN(EMP_NAME) instead of LENGTH()

--  4. Write a query to return the top N records using the TOP command:

```
SELECT * FROM EMP
```

```
LIMIT 5;
```

--Replace 5 with your desired N.

--  5. Find all employees who earn more than the average salary of all employees of their company:

```
SELECT EMP_NAME, SALARY
```

```
FROM EMP
```

```
WHERE SALARY > (
```

```
    SELECT AVG(SALARY) FROM EMP
```

```
);
```

QUESTION 11:-

1. Fetch the first 5 records from the table

sql

Copy code

```
SELECT * FROM Employee
```

```
LIMIT 5;
```

2. Fetch the three minimum salaries from the table

sql

Copy code

```
SELECT DISTINCT Salary
```

```
FROM Employee
```

ORDER BY Salary ASC

LIMIT 3;

3. Display the maximum salary being paid to a clerk

(Assuming clerks are under the "**HR**" department or you can modify WHERE clause if you have a Designation column.)

sql

Copy code

```
SELECT MAX(Salary) AS MaxClerkSalary
```

```
FROM Employee
```

```
WHERE Department = 'HR';
```

4. Display department names and total number of employees in each group

sql

Copy code

```
SELECT Department, COUNT(*) AS TotalEmployees
```

```
FROM Employee
```

```
GROUP BY Department;
```

5. Display departments with more than 3 employees

sql

Copy code

```
SELECT Department, COUNT(*) AS TotalEmployees
```

```
FROM Employee
```

```
GROUP BY Department
```

```
HAVING COUNT(*) > 3;
```

QUESTION 12:-

```
CREATE TABLE Employee (  
    E_id INT PRIMARY KEY,  
    E_name VARCHAR(100),  
    Salary DECIMAL(10, 2),  
    Department VARCHAR(50),  
    Designation VARCHAR(50)  
);  
  
INSERT INTO Employee (E_id, E_name, Salary, Department, Designation) VALUES  
(101, 'Sagar', 3000.00, 'HR', 'Clerk'),  
(102, 'Aarti', 4000.00, 'Finance', 'Manager'),  
(103, 'Rohit', 2800.00, 'HR', 'Clerk'),  
(104, 'Neeta', 3500.00, 'IT', 'Developer'),  
(105, 'Ramesh', 3600.00, 'HR', 'Clerk'),  
(106, 'Swara', 4500.00, 'IT', 'Analyst'),  
(107, 'Aisha', 2500.00, 'Finance', 'Clerk'),  
(108, 'Vikas', 3100.00, 'HR', 'Clerk'),  
(109, 'Radha', 3700.00, 'Sales', 'Clerk'),  
(110, 'Sonya', 2900.00, 'IT', 'Tester'),  
(111, 'Sonal', 4600.00, 'IT', 'Manager');
```

--1. Fetch the first 5 records from a table

```
SELECT * FROM Employee  
LIMIT 5;
```

--2. Fetch the three minimum salaries from the table

```
SELECT DISTINCT Salary  
FROM Employee
```

ORDER BY Salary ASC

LIMIT 3;

--3. Display the maximum salary being paid to a Clerk

SELECT MAX(Salary) AS MaxClerkSalary

FROM Employee

WHERE Designation = 'Clerk';

--4. Display department names and total number of employees in each group

SELECT Department, COUNT(*) AS TotalEmployees

FROM Employee

GROUP BY Department;

--5. Display the departments with more than three employees

SELECT Department, COUNT(*) AS TotalEmployees

FROM Employee

GROUP BY Department

HAVING COUNT(*) > 3;

QUESTION 12:- M

QUERY

QUESTION 13:-

DROP TABLE IF EXISTS Products;

CREATE TABLE Products (

product_id INT PRIMARY KEY,

product_name VARCHAR(100),

```
category VARCHAR(50),
unit_price DECIMAL(10, 2)
);
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES
(101, 'Laptop', 'Electronics', 500.00),
(102, 'Smartphone', 'Electronics', 300.00),
(103, 'Headphones', 'Electronics', 30.00),
(104, 'Keyboard', 'Electronics', 20.00),
(105, 'Mouse', 'Electronics', 15.00);
DROP TABLE IF EXISTS Sales;
```

```
CREATE TABLE Sales (
    sale_id INT PRIMARY KEY,
    product_id INT,
    quantity INT,
    sale_date DATE,
    total_price DECIMAL(10,2),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
INSERT INTO Sales (sale_id, product_id, quantity, sale_date, total_price) VALUES
(1, 101, 2, '2024-01-03', 1000.00),
(2, 102, 1, '2024-01-03', 300.00),
(3, 103, 3, '2024-01-04', 90.00),
(4, 104, 5, '2024-01-05', 100.00),
(5, 105, 4, '2024-01-05', 60.00);
```

--1. Retrieve the product_name and unit_price from the Products table

```
SELECT product_name, unit_price
```


FROM Products;

--2. Filter the Products table to show only products in the 'Electronics' category

SELECT *

FROM Products

WHERE category = 'Electronics';

--3. Retrieve the sale_id and total_price from the Sales table for sales made on January 3, 2024

SELECT sale_id, total_price

FROM Sales

WHERE sale_date = '2024-01-03';

--4. Calculate the total revenue generated from all sales

SELECT SUM(total_price) AS total_revenue

FROM Sales;

--5. Count Sales Per Day

SELECT sale_date, COUNT(*) AS sales_count

FROM Sales

GROUP BY sale_date;

--6. Retrieve product_name and unit_price from Products with the Highest Unit Price

SELECT product_name, unit_price

FROM Products

WHERE unit_price = (SELECT MAX(unit_price) FROM Products);

--7. Retrieve product_name and unit_price from Products, ordered by unit_price in descending order

```
SELECT product_name, unit_price  
FROM Products  
ORDER BY unit_price DESC;
```