**✅ Basic SQL Coding Questions**

1. **Select all data from a table**
2. SELECT \* FROM employees;
3. **Select specific columns**
4. SELECT name, salary FROM employees;
5. **Use WHERE clause**
6. SELECT \* FROM employees WHERE department = 'HR';
7. **Filter using multiple conditions (AND/OR)**
8. SELECT \* FROM employees WHERE age > 30 AND department = 'Sales';
9. **Use IN, NOT IN**
10. SELECT \* FROM employees WHERE department IN ('IT', 'Finance');
11. **Use LIKE operator**
12. SELECT \* FROM employees WHERE name LIKE 'A%';
13. **ORDER BY ASC/DESC**
14. SELECT \* FROM employees ORDER BY salary DESC;
15. **LIMIT and OFFSET**
16. SELECT \* FROM employees LIMIT 10 OFFSET 5;

**✅ Aggregate Functions + GROUP BY**

1. **Count total rows**
2. SELECT COUNT(\*) FROM employees;
3. **Sum, Avg, Max, Min**

SELECT AVG(salary) FROM employees;

1. **Group by a column**

SELECT department, COUNT(\*) FROM employees GROUP BY department;

1. **Group by with HAVING**

SELECT department, COUNT(\*)

FROM employees

GROUP BY department

HAVING COUNT(\*) > 5;

**✅ Joins (Super Important)**

1. **INNER JOIN**

SELECT e.name, d.name

FROM employees e

INNER JOIN departments d ON e.department\_id = d.id;

1. **LEFT JOIN**

SELECT e.name, d.name

FROM employees e

LEFT JOIN departments d ON e.department\_id = d.id;

1. **RIGHT JOIN**

SELECT e.name, d.name

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.id;

1. **FULL OUTER JOIN (workaround using UNION)**

SELECT e.name, d.name

FROM employees e

LEFT JOIN departments d ON e.department\_id = d.id

UNION

SELECT e.name, d.name

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.id;

**✅ Subqueries**

1. **Scalar subquery**

SELECT name FROM employees

WHERE salary > (SELECT AVG(salary) FROM employees);

1. **IN subquery**

SELECT name FROM employees

WHERE department\_id IN (SELECT id FROM departments WHERE location = 'Chennai');

1. **Correlated subquery**

SELECT name, salary

FROM employees e

WHERE salary > (

SELECT AVG(salary) FROM employees WHERE department\_id = e.department\_id

);

**✅ Advanced Queries**

1. **Find duplicate rows**

SELECT name, COUNT(\*)

FROM students

GROUP BY name

HAVING COUNT(\*) > 1;

1. **Find second highest salary**

SELECT MAX(salary)

FROM employees

WHERE salary < (SELECT MAX(salary) FROM employees);

1. **Top 3 salaries**

SELECT DISTINCT salary

FROM employees

ORDER BY salary DESC

LIMIT 3;

1. **Update using JOIN**

UPDATE employees e

JOIN departments d ON e.department\_id = d.id

SET e.salary = e.salary + 1000

WHERE d.name = 'IT';

1. **Delete duplicates (keep one)**

DELETE FROM students

WHERE id NOT IN (

SELECT MIN(id) FROM students GROUP BY name, age

);

**✅ DDL & Table Design**

1. **Create table**

CREATE TABLE employees (

id INT PRIMARY KEY,

name VARCHAR(100),

salary INT,

department\_id INT

);

1. **Alter table – Add column**

ALTER TABLE employees ADD gender VARCHAR(10);

1. **Alter table – Drop column**

ALTER TABLE employees DROP COLUMN gender;

1. **Create foreign key**

ALTER TABLE employees

ADD CONSTRAINT fk\_dept

FOREIGN KEY (department\_id) REFERENCES departments(id);

**✅ Real-Life Scenarios**

1. **Get students who didn’t enroll in any course**

SELECT \*

FROM students s

WHERE NOT EXISTS (

SELECT 1 FROM enrollments e WHERE e.student\_id = s.id

);

1. **Find departments with highest total salary**

SELECT department\_id, SUM(salary) AS total

FROM employees

GROUP BY department\_id

ORDER BY total DESC

LIMIT 1;