### 1–10: Complex Query and JOIN Questions

**1.** Retrieve the *second highest salary* from an employees table.

SELECT MAX(salary)

FROM employees

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

**2.** Find employees who earn more than the average salary in their department.

SELECT e.emp\_id, e.name, e.salary, e.dept\_id

FROM employees e

WHERE salary > (

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

);

**3.** Get departments with *more than 5 employees*.

SELECT dept\_id, COUNT(\*) AS total\_employees

FROM employees

GROUP BY dept\_id

HAVING COUNT(\*) > 5;

**4.** Find employees who *do not belong* to any department.

**SELECT name FROM employees** 

WHERE dept\_id IS NULL;

**5.** Retrieve employees who have *the same salary* as another employee.

SELECT DISTINCT e1.name, e1.salary

FROM employees e1

JOIN employees e2 ON e1.salary = e2.salary AND e1.emp\_id <> e2.emp\_id;

**6.** Find departments without any employees.

SELECT d.dept name

FROM departments d

LEFT JOIN employees e ON d.dept\_id = e.dept\_id

WHERE e.emp\_id IS NULL;

**7.** Retrieve top 3 salaries per department.

#### **SELECT**

department,

```
name,
  salary
FROM (
  SELECT
    department,
    name,
    salary,
    ROW_NUMBER() OVER (PARTITION BY department ORDER BY salary DESC) AS rank
  FROM employees
) ranked
WHERE rank <= 3;
2 nd method
SELECT e1.*
FROM employees e1
WHERE 3 > (
 SELECT COUNT(DISTINCT e2.salary)
 FROM employees e2
 WHERE e2.dept_id = e1.dept_id AND e2.salary > e1.salary
);
8. List employee names who have manager in their job title.
SELECT name FROM employees WHERE job_title LIKE '%Manager%';
9. Find departments with average salary greater than overall average salary.
SELECT dept_id, AVG(salary) AS avg_salary
FROM employees
GROUP BY dept_id
HAVING AVG(salary) > (SELECT AVG(salary) FROM employees);
10. Get all employees hired in the last 30 days.
SELECT * FROM employees WHERE hire_date >= CURDATE() - INTERVAL 30 DAY;
```

# 11–20: Subqueries, EXISTS, and CTEs

**11.** Find employees who have subordinates.

SELECT name FROM employees e

```
SELECT 1 FROM employees s WHERE s.manager_id = e.emp_id
);

12. Use a CTE to find the total salary expense per department.

WITH dept_salary AS (

SELECT dept_id, SUM(salary) AS total_salary

FROM employees

GROUP BY dept_id
)

SELECT * FROM dept_salary;

13. Get employee(s) with the maximum salary in each department.
```

Excellent, Gaurav 👋

WHERE EXISTS (

This is one of the most frequently asked SQL interview questions — to find employee(s) with the maximum salary in each department.

Let's go through this step by step with multiple approaches so you understand it completely 👇

## 🧱 Sample employees table

emp_id	l name	dept_id	l salary
1	Aditi	10	50000
2	Rahul	10	70000
3	Priya	10	70000
4	Aman	20	55000
5	Neha	20	75000
6	Ravi	20	75000
7	Simran	30	60000

### **©** Goal:

Return the employee(s) who have the maximum salary in their department.

That means:

- In dept 10 → Rahul & Priya (both 70000)
- In dept 20 → Neha & Ravi (both 75000)
- In dept 30 → Simran (60000)

```
✓ Approach 1 — Using Subquery
SELECT name, dept_id, salary
FROM employees e
WHERE salary = (
  SELECT MAX(salary)
  FROM employees
  WHERE dept_id = e.dept_id
);
Explanation:
    1. For each employee e,
       the subquery finds the maximum salary in that employee's department.
    2. If e.salary equals that maximum \rightarrow include in output.
@ Result:
name dept_id salary
               70000
Rahul 10
               70000
Priya
      10
Neha
       20
               75000
Ravi
       20
               75000
Simran 30
               60000
Approach 2 — Using Window Function (MySQL 8+, PostgreSQL, SQL Server)
SELECT name, dept_id, salary
FROM (
  SELECT
```

```
SELECT

name,

dept_id,

salary,

RANK() OVER (PARTITION BY dept_id ORDER BY salary DESC) AS rnk

FROM employees
) ranked
```

# **Q** Explanation:

- RANK() assigns 1 to employees with the highest salary in each department.
- WHERE rnk = 1 filters those top earners.

#### **©** Result:

Same as above <

### name dept\_id salary

Rahul 10 70000

Priya 10 70000

Neha 20 75000

Ravi 20 75000

Simran 30 60000

# **Summary Table**

Method	SQL Feature Used	Works in	Description

1 Subquery Scalar subquery All databases Easiest to write

2 CTE + JOIN CTE (Common Table Expression) All modern databases Most readable

3 Window Function RANK() / DENSE\_RANK() MySQL 8+, PostgreSQL, SQL Server Best for ranking problems

Excellent follow-up, Gaurav 👋

You want to find employees with the maximum salary in each department using GROUP BY.

Let's go step by step 👇

# Sample employees Table

# emp\_id name dept\_id salary

1	Aditi	10	50000

2 Rahul 10 70000

3 Priya 10 70000

4 Aman 20 55000

# emp\_id name dept\_id salary 5 Neha 20 75000 Ravi 20 75000 7 Simran 30 60000 @ Goal: Get employees who earn the maximum salary in their department, but use a **GROUP BY** approach. Step 1: Find the maximum salary per department SELECT dept\_id, MAX(salary) AS max\_salary FROM employees GROUP BY dept\_id; **Result:** dept\_id max\_salary 10 70000 20 75000 30 60000 Step 2: Join this result with the employees table SELECT e.name, e.dept\_id, e.salary FROM employees e JOIN ( SELECT dept\_id, MAX(salary) AS max\_salary FROM employees GROUP BY dept\_id ) m ON e.dept\_id = m.dept\_id AND e.salary = m.max\_salary;

# **Explanation:**

- 1. The **subquery (m)** groups employees by department and finds the **maximum salary** per department.
- 2. The **outer query** joins this result with the main employees table.

3. The join condition e.salary = m.max\_salary ensures we only pick employees with that top salary. **©** Final Output: name dept\_id salary 70000 Rahul 10 Priya 10 70000 Neha 20 75000 Ravi 20 75000 Simran 30 60000 Summary: **Step Operation Purpose** GROUP BY dept\_id Find max salary per department JOIN Match employees having that salary **Final Output** Employees with max salary in each department Would you like me to extend this query to also show the department name (assuming a departments table exists)? Would you like me to modify this query to also show the department name (if it's stored in a separate departments table)? **14.** Retrieve employees who have the *same hire date* as someone else. SELECT name, hire\_date FROM employees WHERE hire date IN ( SELECT hire\_date FROM employees GROUP BY hire\_date HAVING COUNT(\*) > 1 **15.** Find employees earning above company median salary. SELECT e.\*

);

FROM employees e

SELECT AVG(s1.salary)

WHERE salary > (

```
FROM (SELECT salary FROM employees ORDER BY salary LIMIT 50 PERCENT) s1
);
(Note: MySQL doesn't have direct median — use percentile workaround.)
16. Retrieve employees who joined before their manager.
SELECT e.name
FROM employees e
JOIN employees m ON e.manager_id = m.emp_id
WHERE e.hire_date < m.hire_date;
17. Display employees and their rank based on salary.
SELECT name, salary, RANK() OVER (ORDER BY salary DESC) AS rank_no
FROM employees;
18. Use a CTE to calculate cumulative salary by department.
WITH cte AS (
 SELECT dept_id, name, salary,
    SUM(salary) OVER (PARTITION BY dept_id ORDER BY salary DESC) AS cumulative_salary
 FROM employees
SELECT * FROM cte;
19. Find duplicate email IDs in a user table.
SELECT email, COUNT(*) FROM users GROUP BY email HAVING COUNT(*) > 1;
20. Show employees who work in the same city as their manager.
SELECT e.name
FROM employees e
JOIN employees m ON e.manager_id = m.emp_id
WHERE e.city = m.city;
21–30: Triggers, Procedures, and Transactions
21. Create a trigger to log deleted employees.
CREATE TRIGGER after_employee_delete
AFTER DELETE ON employees
```

FOR EACH ROW

VALUES (OLD.emp\_id, NOW());

INSERT INTO employee\_log(emp\_id, deleted\_at)

```
22. Create a stored procedure to increase salary by 10% for a department.
DELIMITER //
CREATE PROCEDURE raise_salary(IN dept INT)
BEGIN
 UPDATE employees SET salary = salary * 1.1 WHERE dept_id = dept;
END //
DELIMITER;
23. How to handle atomic operations?
t Use Transactions:
START TRANSACTION;
UPDATE accounts SET balance = balance - 500 WHERE acc_id = 1;
UPDATE accounts SET balance = balance + 500 WHERE acc_id = 2;
COMMIT;
24. Rollback example:
START TRANSACTION;
DELETE FROM employees WHERE dept_id = 10;
ROLLBACK;
25. Create a trigger to prevent negative salary.
CREATE TRIGGER check_salary
BEFORE INSERT ON employees
FOR EACH ROW
IF NEW.salary < 0 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Salary cannot be negative';
END IF;
26. Create a stored function to calculate annual salary.
DELIMITER //
CREATE FUNCTION annual_salary(s DECIMAL(10,2))
RETURNS DECIMAL(10,2)
RETURN s * 12;
//
DELIMITER;
27. Use a trigger to auto-update last_modified.
CREATE TRIGGER update_timestamp
BEFORE UPDATE ON employees
```

```
SET NEW.last_modified = NOW();
28. Use transaction isolation levels:
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
START TRANSACTION;
-- operations
COMMIT;
29. Example of using SAVEPOINT.
START TRANSACTION;
INSERT INTO users VALUES(1, 'A');
SAVEPOINT s1;
INSERT INTO users VALUES(2, 'B');
ROLLBACK TO s1;
COMMIT;
30. Disable foreign key checks temporarily.
SET FOREIGN_KEY_CHECKS = 0;
-- bulk insert/delete
SET FOREIGN_KEY_CHECKS = 1;
♦ 31–40: Optimization & Indexing
31. View query execution plan.
EXPLAIN SELECT * FROM employees WHERE dept_id = 5;
32. Create an index on email.
CREATE INDEX idx_email ON users(email);
33. Composite index example.
CREATE INDEX idx_dept_salary ON employees(dept_id, salary);
34. Drop an index.
DROP INDEX idx_email ON users;
35. Check index usage.
SHOW INDEX FROM employees;
36. Use ANALYZE TABLE.
ANALYZE TABLE employees;
37. Optimize table storage.
```

FOR EACH ROW

OPTIMIZE TABLE employees;

38. Difference between PRIMARY KEY and UNIQUE KEY?

- PRIMARY KEY: One per table, can't be NULL.
- UNIQUE KEY: Multiple allowed, can contain NULL.

**39.** How does MySQL use indexes internally?

Indexes create B-tree or hash structures to speed up lookups, joins, and sorting by minimizing full table scans.

**40.** How to detect slow queries?

Enable slow query log:

SET GLOBAL slow query log = 1;

SET GLOBAL long\_query\_time = 2;

### **41–50:** Views, Joins, Advanced Concepts

**41.** Create a view of high-salary employees.

CREATE VIEW high salary AS

SELECT name, salary FROM employees WHERE salary > 100000;

42. Update data through a view.

UPDATE high\_salary SET salary = 120000 WHERE name = 'John';

43. Drop a view.

DROP VIEW high salary;

**44.** Self join example.

SELECT e1.name AS Employee, e2.name AS Manager

FROM employees e1

JOIN employees e2 ON e1.manager id = e2.emp id;

45. What's the difference between INNER JOIN and LEFT JOIN?

- INNER JOIN: Returns matching records only.
- **LEFT JOIN:** Returns all from left table, even if no match.

46. Difference between UNION and UNION ALL?

- UNION removes duplicates.
- UNION ALL keeps duplicates.

47. Difference between CHAR and VARCHAR?

- CHAR → Fixed length.
- VARCHAR → Variable length.

**48.** How to find the *nth highest salary*?

**SELECT DISTINCT salary** 

```
WHERE n-1 = (
 SELECT COUNT(DISTINCT salary)
 FROM employees e2
 WHERE e2.salary > e1.salary
);
49. Use CASE for conditional aggregation.
SELECT
 dept_id,
 SUM(CASE WHEN gender = 'M' THEN 1 ELSE 0 END) AS male,
 SUM(CASE WHEN gender = 'F' THEN 1 ELSE 0 END) AS female
FROM employees
GROUP BY dept_id;
50. Detect and delete duplicate rows keeping only one.
DELETE e1 FROM employees e1
JOIN employees e2
ON e1.email = e2.email AND e1.emp_id > e2.emp_id;
```

FROM employees e1

Would you like me to give you these **50 questions in a structured table (with difficulty, concept tags, and short explanations)** — perfect for revision before interviews?