📝TOP 20 SQL QUESTIONS FOR INTERVIEW

Basic SQL Questions

1. Retrieve all records from a table.

```sql
SELECT * FROM employees;
```

Explanation: Fetches all columns and rows from the employees table.

2. Fetch unique departments from the employees table.

```sql
SELECT DISTINCT department_id FROM employees;
```

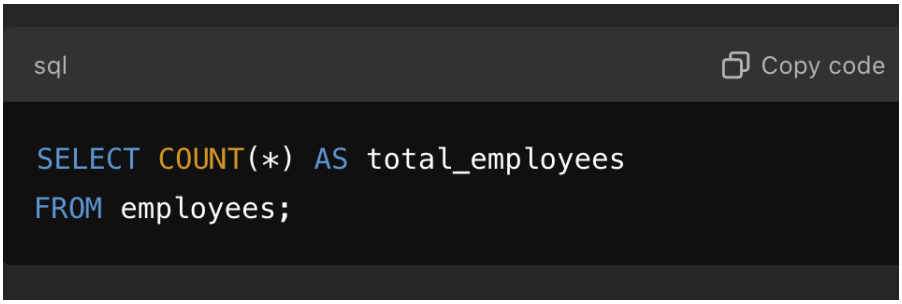
Explanation: Eliminates duplicates and lists unique department IDs.

3. Sort employees by their salary in descending order.

```sql
SELECT employee_id, name, salary
FROM employees
ORDER BY salary DESC;
```

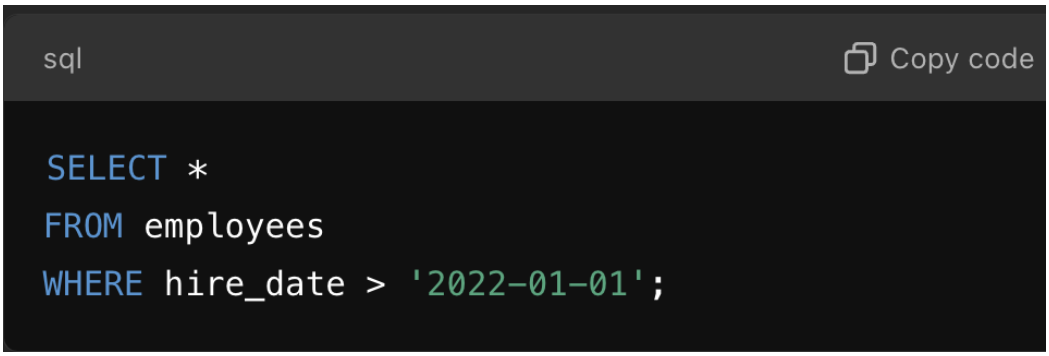
- **Explanation: Orders rows by salary in descending order.**

4. Count the total number of employees in the company.



- **Explanation: Counts all rows in the employees table.**

5. Retrieve employees who joined after a specific date (e.g., '2022-01-01').



- **Explanation: Filters employees whose hire_date is after the given date.**

6. Find employees whose name starts with 'A'.

```sql
SELECT *
FROM employees
WHERE name LIKE 'A%';
```

- **Explanation: Uses LIKE to match names that begin with 'A'.**

7. Fetch employees who belong to specific departments (e.g., HR and Finance).

```sql
SELECT *
FROM employees
WHERE department_id IN (1, 2); -- Assume 1 = HR, 2
```

- **Explanation: Filters rows for specified department IDs using IN.**

8. Retrieve employees with salaries between 50,000 and 100,000.

```sql
SELECT *
FROM employees
WHERE salary BETWEEN 50000 AND 100000;
```

- **Explanation: Fetches employees with salaries within the given range.**

9. Find the total salary paid to all employees.

```sql
SELECT SUM(salary) AS total_salary
FROM employees;
```

Explanation: Computes the sum of all salaries in the employees table.

10. Fetch employees who do not have a manager assigned.

```sql
SELECT *
FROM employees
WHERE manager_id IS NULL;
```

- **Explanation: Finds rows where manager_id is NULL.**

Intermediate SQL Questions

11. Find the department-wise average salary of employees.

```sql
SELECT department_id, AVG(salary) AS avg_salary
FROM employees
GROUP BY department_id;
```

- **Explanation: Groups rows by department_id and calculates average salary for each department.**
- **Example Output:**

```lua
| Department_ID | Avg_Salary |
|---------------|------------|
|      1        |   75000    |
|      2        |   85000    |
```

12. Fetch the top 3 highest salaries.

```sql
SELECT DISTINCT salary
FROM employees
ORDER BY salary DESC
LIMIT 3;
```

- **Explanation: Sorts salaries in descending order and fetches the top 3.**

---

13. Identify duplicate records based on employee name and department.

```sql
SELECT name, department_id, COUNT(*)
FROM employees
GROUP BY name, department_id
HAVING COUNT(*) > 1;
```

- **Explanation: Groups by name and department_id, filtering duplicates using HAVING COUNT(*) > 1.**

---

14. Retrieve employees with no matching records in another table (e.g., no projects assigned).

```sql
SELECT e.*
FROM employees e
LEFT JOIN projects p ON e.employee_id = p.employee
WHERE p.employee_id IS NULL;
```

- **Explanation: Uses a LEFT JOIN to find employees without matching records in projects.**
- **Diagram:**

```sql
Employees Table            Projects Table
| Employee_ID |           | Employee_ID |
|-------------|           |-------------|
|      101    |           |      102    |
|      102    |           |      103    |
|      103    |           |      NULL   |

Result:
| Employee_ID |
|-------------|
|      101    |
```

15. Write a query to find the employee with the maximum salary.

```sql
SELECT employee_id, name, salary
FROM employees
WHERE salary = (SELECT MAX(salary) FROM employees);
```

- **Explanation: Retrieves the employee with the highest salary using a subquery.**

---

16. Calculate the difference between the highest and lowest salaries.

```sql
SELECT MAX(salary) - MIN(salary) AS salary_differen
FROM employees;
```

- **Explanation: Computes the difference between the maximum and minimum salaries.**

---

17. Retrieve employees who share the same manager.

```sql
SELECT manager_id, COUNT(*) AS num_employees
FROM employees
GROUP BY manager_id
HAVING COUNT(*) > 1;
```

- **Explanation: Groups employees by their manager_id and filters managers with more than one reportee.**

---

18. Fetch employees along with their department names.

```sql
SELECT e.employee_id, e.name, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.departmen
```

- **Explanation: Joins employees and departments to fetch department names.**

---

19. Rank employees based on their salary.

```sql
SELECT employee_id, name, salary,
       RANK() OVER (ORDER BY salary DESC) AS rank
FROM employees;
```

- **Explanation: Uses a window function to rank employees by their salary.**

20. Identify employees who earn above the department's average salary.

```sql
SELECT e.employee_id, e.name, e.salary, e.department_id
FROM employees e
JOIN (
    SELECT department_id, AVG(salary) AS avg_salary
    FROM employees
    GROUP BY department_id
) dept_avg ON e.department_id = dept_avg.department_id
WHERE e.salary > dept_avg.avg_salary;
```

- **Explanation: Joins employees with a subquery calculating department averages to filter high earners.**