Here are **technical scenario-based SQL interview questions** for query writing:

Basic Queries

1. Retrieve all employees who earn more than their department's average salary.
2. Find employees whose names contain both 'A' and 'B', regardless of order.
3. Get the third highest salary from the employees table.
4. Select the second most frequently occurring salary in a table.
5. Retrieve all employees who were hired on a Monday.
6. Get the top `N` salaries in a department.
7. Retrieve employees who have the same salary as someone else in the company.
8. Find departments with fewer than `5` employees.
9. Fetch employee details along with their manager's name.
10. Get the total salary expense for each department.

Intermediate Queries

1. List employees who joined within the last `6` months.
2. Retrieve employees who have never received a bonus.
3. Find the department with the highest total salary expense.
4. Get employees whose salaries fall in the **top 10%** of the company.
5. Identify gaps in an invoice number sequence.
6. Get employees who report directly or indirectly to a specific manager.
7. Retrieve duplicate records from a table without using `DISTINCT`.
8. Find consecutive employees who have the same salary.
9. Swap values of two columns in a table.
10. Select `N` random employees from a table.

Advanced Queries

1. Retrieve employees whose salaries are greater than the **average salary** of their department.
2. Get the cumulative salary for each employee based on joining date.
3. Fetch all employees who have **more than one** manager.
4. List employees who have changed their department more than twice.
5. Select employees who received a salary increment of more than **10%** compared to their previous salary.
6. Find the **maximum consecutive** days an employee was present in an attendance table.
7. Retrieve employees who have never taken leave.
8. Find employees whose joining date is the same as someone else's.
9. Fetch departments where no employees have been hired in the last `1` year.
10. Identify and correct `NULL` values in a dataset dynamically.

Complex Scenarios

1. Find the top 3 earning employees in each department.
2. Retrieve products that have been sold **every** month in the last year.
3. Get employees who have had at least **two** promotions.
4. Identify and remove duplicate rows while keeping only the first occurrence.
5. Fetch employees whose salary has increased every year consistently.
6. Find customers who have placed orders in **every** month of a given year.
7. Get employees who work in multiple departments.
8. Identify users who have logged in at least 5 times in a single day.
9. List products that have been purchased by **at least 2 different customers** on the same day.
10. Retrieve employees who joined within 7 days of each other.

Performance Optimization & Query Tuning

1. Retrieve the **last 5 records** inserted into a table without using `ORDER BY`.
2. Optimize a query that retrieves the 10 highest salaries from a large dataset.
3. Find slow queries running on a database and suggest an index.
4. Identify and remove duplicate indexes on a table.
5. Implement pagination efficiently in an SQL query.
6. Fetch records efficiently from a table with **millions of rows**.
7. Retrieve employee records with an **index-friendly** query approach.
8. Detect unused indexes in a database.
9. Find all foreign key constraints referencing a particular table.
10. Analyze a query execution plan and suggest improvements.

🔥 **Want more tricky SQL scenarios?** Let me know! 🚀
Here's a sample database schema with some **realistic data** that you can use to practice the SQL questions.

Database Schema (HR Management System)

1️⃣ Employees Table

```
CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    department_id INT,
    salary DECIMAL(10,2),
    hire_date DATE,
    manager_id INT,
    email VARCHAR(100) UNIQUE
);
```

2️⃣ Departments Table

```
CREATE TABLE departments (
    id INT PRIMARY KEY AUTO_INCREMENT,
    department_name VARCHAR(100),
    location VARCHAR(100)
);
```

3️⃣ Salaries Table (Salary History)

```
CREATE TABLE salaries (
    employee_id INT,
```

```
    salary DECIMAL(10,2),
    change_date DATE,
    PRIMARY KEY (employee_id, change_date),
    FOREIGN KEY (employee_id) REFERENCES employees(id) ON DELETE
CASCADE
);
```

4️⃣ Attendance Table

```
CREATE TABLE attendance (
    employee_id INT,
    attendance_date DATE,
    status ENUM('Present', 'Absent', 'Leave'),
    PRIMARY KEY (employee_id, attendance_date),
    FOREIGN KEY (employee_id) REFERENCES employees(id) ON DELETE
CASCADE
);
```

5️⃣ Employee Promotions Table

```
CREATE TABLE promotions (
    employee_id INT,
    promotion_date DATE,
    new_salary DECIMAL(10,2),
    PRIMARY KEY (employee_id, promotion_date),
    FOREIGN KEY (employee_id) REFERENCES employees(id) ON DELETE
CASCADE
);
```

Insert Sample Data

```
INSERT INTO departments (id, department_name, location) VALUES
(1, 'HR', 'Mumbai'),
(2, 'Engineering', 'Bangalore'),
(3, 'Sales', 'Delhi'),
(4, 'Finance', 'Pune');

INSERT INTO employees (id, name, department_id, salary, hire_date,
manager_id, email) VALUES
(101, 'Amit Sharma', 2, 75000, '2022-03-15', NULL,
'amit.sharma@example.com'),
(102, 'Priya Mehta', 1, 60000, '2021-07-10', 101,
'priya.mehta@example.com'),
(103, 'Rahul Verma', 3, 85000, '2020-05-22', 101,
'rahul.verma@example.com'),
(104, 'Sanya Gupta', 2, 95000, '2019-11-30', NULL,
'sanya.gupta@example.com'),
(105, 'Neha Joshi', 4, 55000, '2023-01-10', 102,
'neha.joshi@example.com');

INSERT INTO salaries (employee_id, salary, change_date) VALUES
(101, 70000, '2021-01-01'),
(101, 75000, '2022-03-15'),
(102, 55000, '2020-06-01'),
(102, 60000, '2021-07-10'),
(103, 80000, '2019-04-01'),
(103, 85000, '2020-05-22');

INSERT INTO attendance (employee_id, attendance_date, status) VALUES
(101, '2024-02-01', 'Present'),
(101, '2024-02-02', 'Absent'),
(102, '2024-02-01', 'Present'),
(103, '2024-02-01', 'Leave');
```

```
INSERT INTO promotions (employee_id, promotion_date, new_salary)
VALUES
(101, '2022-03-15', 75000),
(102, '2021-07-10', 60000),
(103, '2020-05-22', 85000);
```

How to Use This?

1️⃣ **Copy-paste these queries** into your SQL database (MySQL, PostgreSQL, or any RDBMS).
2️⃣ **Use this schema** to solve SQL practice questions.
3️⃣ **Modify the data** or add more test cases as needed.

🔥 Now, go ahead and **practice SQL queries** on this dataset! 🚀 If you need **more complex scenarios**, let me know! 😎