SQL Practice Questions:

Here is a sample schema for the tables referenced in the SQL queries below. This schema includes the employees, departments, and salaries tables, along with their attributes.

1. employees Table

This table stores information about employees.

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    job_title VARCHAR(100),
    salary DECIMAL(10, 2),
    hire_date DATE,
    department_id INT,
    manager_id INT,
    FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

2. departments Table

This table stores information about the departments in the company.

```
sql
Copy code
CREATE TABLE departments (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(100)
);
```

3. salaries Table

This table stores salary information for employees. You may store historical salary data or a current salary record.

```
CREATE TABLE salaries (
    salary_id INT PRIMARY KEY,
```

AlgoBoost

```
    employee_id INT,
    salary DECIMAL(10, 2),
    from_date DATE,
    to_date DATE,
    FOREIGN KEY (employee_id) REFERENCES employees(employee_id)
);
```

1. employees Table Data

```
INSERT INTO employees (employee_id, first_name, last_name, job_title, salary, hire_date, department_id, manager_id)
VALUES
(1, 'John', 'Doe', 'Software Engineer', 60000, '2015-06-01', 1, 2),
(2, 'Jane', 'Smith', 'Software Engineer', 65000, '2016-07-15', 1, 2),
(3, 'Mike', 'Johnson', 'HR Manager', 55000, '2014-03-22', 2, NULL),
(4, 'Emily', 'Davis', 'Marketing Specialist', 48000, '2018-11-05', 3, 5),
(5, 'James', 'Brown', 'CEO', 120000, '2010-01-01', 1, NULL),
(6, 'Sophia', 'Williams', 'CTO', 110000, '2012-04-20', 1, 5);
```

2. departments Table Data

```
INSERT INTO departments (department_id, department_name)
VALUES
(1, 'Engineering'),
(2, 'Human Resources'),
(3, 'Marketing');
```

3. salaries Table Data

```
INSERT INTO salaries (salary_id, employee_id, salary, from_date, to_date)
VALUES
(1, 1, 60000, '2015-06-01', '2024-12-31'),
(2, 2, 65000, '2016-07-15', '2024-12-31'),
(3, 3, 55000, '2014-03-22', '2024-12-31'),
(4, 4, 48000, '2018-11-05', '2024-12-31'),
(5, 5, 120000, '2010-01-01', '2024-12-31'),
(6, 6, 110000, '2012-04-20', '2024-12-31');
```

AlgoBoost

Relationships:

- **Employees are linked to Departments via department_id.**
- **Employees may also have a manager_id, which refers to another employee_id (indicating the manager).**
- **Salaries are linked to Employees via employee_id.**

With this schema, you can now execute the SQL queries from the below list on this database structure.

AlgoBoost

Basic SQL Query Questions

1. **Write a query to select all columns from a table called employees.**

```
SELECT * FROM employees;
```

2. **Write a query to select the first name and last name of all employees from the employees table.**

```
SELECT first_name, last_name FROM employees;
```

3. **Write a query to find the total number of employees in the employees table.**

```
SELECT COUNT(*) FROM employees;
```

4. **Write a query to select all employees who have a salary greater than 50,000.**

```
SELECT * FROM employees WHERE salary > 50000;
```

5. **Write a query to find employees whose first name starts with the letter "A".**

```
SELECT * FROM employees WHERE first_name LIKE 'A%';
```

6. **Write a query to find employees whose salary is between 40,000 and 60,000.**

```
SELECT * FROM employees WHERE salary BETWEEN 40000 AND 60000;
```

7. **Write a query to sort employees by their salary in descending order.**

```
SELECT * FROM employees ORDER BY salary DESC;
```

8. **Write a query to find the maximum salary from the employees table.**

```
SELECT MAX(salary) FROM employees;
```

9. **Write a query to find the average salary of employees.**

```
SELECT AVG(salary) FROM employees;
```

10. **Write a query to find employees who do not have a manager (i.e., the manager_id is NULL).**

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

Intermediate SQL Query Questions

11. **Write a query to count the number of employees in each department.**

```
SELECT department_id, COUNT(*) FROM employees GROUP BY department_id;
```

12. **Write a query to select the employees who have a salary greater than the average salary.**

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

13. **Write a query to select the second highest salary from the employees table.**

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

14. **Write a query to find the employees with the highest salary in each department.**

```
SELECT department_id, first_name, last_name, salary
FROM employees
WHERE salary IN (SELECT MAX(salary) FROM employees GROUP BY department_id);
```

15. **Write a query to select the employees whose first name is 'John' or 'Jane'.**

```
SELECT * FROM employees WHERE first_name IN ('John', 'Jane');
```

AlgoBoost

16. **Write a query to select employees whose salary is less than the average salary.**

```
SELECT * FROM employees WHERE salary < (SELECT AVG(salary) FROM employees);
```

17. **Write a query to find the department with the highest number of employees.**

```
SELECT department_id, COUNT(*) AS employee_count
FROM employees
GROUP BY department_id
ORDER BY employee_count DESC
LIMIT 1;
```

18. **Write a query to find the employee who has been with the company the longest.**

```
SELECT * FROM employees ORDER BY hire_date ASC LIMIT 1;
```

19. **Write a query to update the salary of an employee with employee_id 101 to 55,000.**

```
UPDATE employees SET salary = 55000 WHERE employee_id = 101;
```

20. **Write a query to delete all employees from the employees table who have a salary below 30,000.**

```
DELETE FROM employees WHERE salary < 30000;
```

AlgoBoost

JOIN Queries

21. **Write a query to join the employees and departments tables to display employee names and their department names.**

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id;
```

22. **Write a query to perform an INNER JOIN between employees and departments tables and display employee names and their department names.**

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
```

```
INNER JOIN departments d ON e.department_id = d.department_id;
```

23. **Write a query to perform a LEFT JOIN between employees and departments and display all employees and their department names (if available).**

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id;
```

24. **Write a query to perform a RIGHT JOIN between employees and departments and display all departments and their employees (if available).**

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
RIGHT JOIN departments d ON e.department_id = d.department_id;
```

25. **Write a query to perform a FULL OUTER JOIN between employees and departments and display all employees and departments.**

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
FULL OUTER JOIN departments d ON e.department_id = d.department_id;
```

26. **Write a query to join three tables: employees, departments, and salaries, and display the employee name, department, and salary.**

```
SELECT e.first_name, e.last_name, d.department_name, s.salary
FROM employees e
JOIN departments d ON e.department_id = d.department_id
JOIN salaries s ON e.employee_id = s.employee_id;
```

27. **Write a query to find employees who do not belong to any department (use LEFT JOIN).**

```
SELECT e.first_name, e.last_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id
WHERE d.department_id IS NULL;
```

AlgoBoost

Group By and Aggregate Functions

28. **Write a query to find the total salary paid to employees in each department.**

```
SELECT department_id, SUM(salary) AS total_salary
FROM employees
GROUP BY department_id;
```

29. **Write a query to find the number of employees in each department.**

```
SELECT department_id, COUNT(*) AS num_employees
FROM employees
GROUP BY department_id;
```

30. **Write a query to find the minimum, maximum, and average salary in the employees table.**

```
SELECT MIN(salary) AS min_salary, MAX(salary) AS max_salary, AVG(salary) AS avg_salary
FROM employees;
```

31. **Write a query to group employees by their department and count how many employees are in each department.**

```
SELECT department_id, COUNT(*) AS num_employees
FROM employees
GROUP BY department_id;
```

32. **Write a query to find the employee with the highest salary in each department.**

```
SELECT department_id, first_name, last_name, salary
FROM employees e
WHERE salary IN (SELECT MAX(salary) FROM employees WHERE department_id = e.department_id
GROUP BY department_id);
```

33. **Write a query to find the departments that have more than 10 employees.**

```
SELECT department_id, COUNT(*) AS num_employees
FROM employees
GROUP BY department_id
```

AlgoBoost

HAVING COUNT(*) > 10;

Subqueries

34. **Write a query to find employees who earn more than the average salary of all employees.**

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

35. **Write a query to find the department with the highest average salary.**

SELECT department_id

FROM employees

GROUP BY department_id

ORDER BY AVG(salary) DESC

LIMIT 1;

36. **Write a query to find employees whose salary is higher than the salary of the employee with employee_id 100.**

SELECT * FROM employees WHERE salary > (SELECT salary FROM employees WHERE employee_id = 100);

37. **Write a query to find employees who do not have any subordinates (i.e., employees who are not managers).**

SELECT * FROM employees WHERE employee_id NOT IN (SELECT DISTINCT manager_id FROM employees WHERE manager_id IS NOT NULL);

38. **Write a query to find the employee with the lowest salary.**

SELECT * FROM employees WHERE salary = (SELECT MIN(salary) FROM employees);

39. **Write a query to find employees who work in the same department as employee with employee_id 101.**

SELECT * FROM employees WHERE department_id = (SELECT department_id FROM employees WHERE employee_id = 101);

40. **Write a query to find employees who have the same salary as the highest-paid employee.**

SELECT * FROM employees WHERE salary = (SELECT MAX(salary) FROM employees);

AlgoBoost

Advanced SQL Query Questions

41. **Write a query to find employees who have the same job title as the employee with employee_id 101.**

```
SELECT * FROM employees WHERE job_title = (SELECT job_title FROM employees WHERE employee_id = 101);
```

42. **Write a query to find employees who were hired in the last 6 months.**

```
SELECT * FROM employees WHERE hire_date > CURRENT_DATE - INTERVAL 6 MONTH;
```

43. **Write a query to find the top 3 highest-paid employees.**

```
SELECT * FROM employees ORDER BY salary DESC LIMIT 3;
```

44. **Write a query to find employees who have been with the company for more than 5 years.**

```
SELECT * FROM employees WHERE hire_date < CURRENT_DATE - INTERVAL 5 YEAR;
```

45. **Write a query to update the salary of employees who have been with the company for more than 5 years by 10%.**

```
UPDATE employees
SET salary = salary * 1.1
WHERE hire_date < CURRENT_DATE - INTERVAL 5 YEAR;
```

46. **Write a query to find the average salary of employees in each department, but only for departments with more than 5 employees.**

```
SELECT department_id, AVG(salary) AS avg_salary
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 5;
```

47. **Write a query to calculate the total salary paid to employees for each year (group by hire year).**

```
SELECT YEAR(hire_date) AS hire_year, SUM(salary) AS total_salary
FROM employees
```

AlgoBoost

```
GROUP BY YEAR(hire_date);
```

48. **Write a query to find the employees who were hired in the same month as employee with employee_id 101.**

```
SELECT * FROM employees WHERE MONTH(hire_date) = MONTH((SELECT hire_date FROM employees WHERE employee_id = 101));
```

49. **Write a query to find the employee who earns the most in each department (use window functions).**

```
SELECT employee_id, department_id, first_name, last_name, salary,
    RANK() OVER (PARTITION BY department_id ORDER BY salary DESC) AS rank
FROM employees
WHERE rank = 1;
```

50. **Write a query to find employees who have the same job title and salary as another employee in the company.**

```
SELECT e1.first_name, e1.last_name, e1.salary, e1.job_title
FROM employees e1
JOIN employees e2 ON e1.job_title = e2.job_title AND e1.salary = e2.salary
WHERE e1.employee_id != e2.employee_id;
```

AlgoBoost