Top 30 LeetCode SQL Questions and Answers

by tarun reddy

1. Combine Two Tables

- Combine two tables: Person and Address using LEFT JOIN to get complete person info, even if address is missing.

```
SELECT FirstName, LastName, City, State
FROM Person p
LEFT JOIN Address a ON p.PersonId = a.PersonId;
```

2. Second Highest Salary

- Find the second highest unique salary from the Employee table.

```
SELECT MAX(Salary) AS SecondHighestSalary
FROM Employee
WHERE Salary < (SELECT MAX(Salary) FROM Employee);
```

3. Nth Highest Salary

- Create a function to find the Nth highest unique salary in the Employee table.

```
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
  RETURN (
      SELECT DISTINCT Salary
      FROM Employee
      ORDER BY Salary DESC
      LIMIT 1 OFFSET N-1
  );
```

4. Rank Scores

- Rank the scores using DENSE_RANK function so that same scores have the same rank.

```
SELECT Score, DENSE_RANK() OVER (ORDER BY Score DESC) AS Rank
FROM Scores;
```

5. Consecutive Numbers

- Find numbers that appear at least three times consecutively in the Logs table.

```
SELECT DISTINCT l1.Num AS ConsecutiveNums
FROM Logs l1, Logs l2, Logs l3
WHERE l1.Id = l2.Id - 1 AND l2.Id = l3.Id - 1 AND
      l1.Num = l2.Num AND l2.Num = l3.Num;
```

6. Employees Earning More Than Their Managers

- Return names of employees whose salary is greater than their manager's salary.

```
SELECT e.Name AS Employee
FROM Employee e
```

```
JOIN Employee m ON e.ManagerId = m.Id
WHERE e.Salary > m.Salary;
```

7. Duplicate Emails

- Identify all email addresses that appear more than once in the Person table.

```
SELECT Email
FROM Person
GROUP BY Email
HAVING COUNT(*) > 1;
```

8. Customers Who Never Order

- List all customers who never placed any orders.

```
SELECT Name AS Customers
FROM Customers
WHERE Id NOT IN (SELECT CustomerId FROM Orders);
```

9. Department Highest Salary

- Return the highest salary paid in each department along with the employee name.

```
SELECT d.Name AS Department, e.Name AS Employee, e.Salary
FROM Employee e
JOIN Department d ON e.DepartmentId = d.Id
WHERE (e.Salary, e.DepartmentId) IN (
    SELECT MAX(Salary), DepartmentId
    FROM Employee
    GROUP BY DepartmentId
);
```

10. Department Top Three Salaries

- Find top 3 highest salaries in each department.

```
SELECT Department, Employee, Salary
FROM (
  SELECT d.Name AS Department, e.Name AS Employee, e.Salary,
         DENSE_RANK() OVER (PARTITION BY DepartmentId ORDER BY Salary DESC) AS rank
  FROM Employee e
  JOIN Department d ON e.DepartmentId = d.Id
) ranked
WHERE rank <= 3;
```

11. Delete Duplicate Emails

- Remove duplicate emails, keeping only the record with the smallest Id.

```
DELETE FROM Person
```

```
WHERE Id NOT IN (
    SELECT MIN(Id)
    FROM Person
    GROUP BY Email
);
```

12. Rising Temperature

- Find dates where the temperature was higher than the previous day.

```
SELECT w1.Id
FROM Weather w1
JOIN Weather w2 ON DATEDIFF(w1.RecordDate, w2.RecordDate) = 1
WHERE w1.Temperature > w2.Temperature;
```

13. Trips and Users

- Calculate the cancellation rate for trips on specific dates.

```
SELECT Request_at AS Day,
         ROUND(SUM(CASE WHEN Status != 'completed' THEN 1 ELSE 0 END)/COUNT(*), 2) AS
'Cancellation Rate'
FROM Trips t
JOIN Users c ON t.Client_Id = c.Users_Id
JOIN Users d ON t.Driver_Id = d.Users_Id
WHERE Request_at BETWEEN '2013-10-01' AND '2013-10-03'
  AND c.Banned='No' AND d.Banned='No'
GROUP BY Request_at;
```

14. Game Play Analysis I

- Find the first login date for each player.

```
SELECT player_id, MIN(event_date) AS first_login
FROM Activity
GROUP BY player_id;
```

15. Game Play Analysis II

- Get player ID and device ID of first login for each player.

```
SELECT player_id, device_id
FROM Activity
WHERE (player_id, event_date) IN (
  SELECT player_id, MIN(event_date)
  FROM Activity
  GROUP BY player_id
);
```

16. Game Play Analysis III

- Compute average sessions per user by counting distinct event dates.

```
SELECT player_id,
               ROUND(COUNT(DISTINCT  event_date)  /  COUNT(DISTINCT  login_date),  2)  AS
avg_sessions_per_user
FROM (
  SELECT player_id, event_date, login_date
  FROM Activity
) a
GROUP BY player_id;
```

17. Game Play Analysis IV

- Calculate the fraction of players who logged in on consecutive days.

```
SELECT  ROUND(COUNT(DISTINCT  a1.player_id)  /  (SELECT  COUNT(DISTINCT  player_id)  FROM
Activity), 2) AS fraction
FROM Activity a1
JOIN Activity a2 ON a1.player_id = a2.player_id
WHERE DATEDIFF(a1.event_date, a2.event_date) = 1;
```

18. Median Employee Salary

- Get median salary for each company using window functions.

```
SELECT Id, Company, Salary
FROM (
  SELECT Id, Company, Salary,
         ROW_NUMBER() OVER (PARTITION BY Company ORDER BY Salary) AS rn,
         COUNT(*) OVER (PARTITION BY Company) AS cnt
  FROM Employee
) sub
WHERE rn = cnt / 2 + 1 OR rn = (cnt + 1) / 2;
```

19. Managers with at Least 5 Direct Reports

- Find managers who have at least five direct reports.

```
SELECT Name
FROM Employee
WHERE Id IN (
    SELECT ManagerId
    FROM Employee
    GROUP BY ManagerId
    HAVING COUNT(*) >= 5
);
```

20. Find Median Given Frequency

- Calculate median from a table where numbers have frequencies.

```
SELECT Number FROM Numbers
ORDER BY Number
LIMIT 1 OFFSET (
  SELECT (SUM(Frequency) - 1) / 2 FROM Numbers
);
```

21. Get Highest Answer Rate Question

- Identify the question with the highest answer-to-view ratio.

```
SELECT question_id
FROM SurveyLog
GROUP BY question_id
ORDER BY SUM(CASE WHEN action = 'answer' THEN 1 ELSE 0 END)/COUNT(*) DESC
LIMIT 1;
```

22. Find Cumulative Salary of an Employee

- Calculate running salary totals using window functions.

```
SELECT Id, Month, Salary,
       SUM(Salary) OVER (PARTITION BY Id ORDER BY Month) AS CumulativeSalary
FROM Employee;
```

23. Count Student Number in Departments

- Count number of students per department.

```
SELECT d.dept_name, COUNT(s.student_id) AS student_number
FROM department d
LEFT JOIN student s ON d.dept_id = s.dept_id
GROUP BY d.dept_name;
```

24. Find Customer Referee

- Return customers who were not referred by ID 2.

```
SELECT name
FROM Customer
WHERE referee_id != 2 OR referee_id IS NULL;
```

25. Customer Placing the Largest Number of Orders

- Find the customer who placed the most orders.

```
SELECT customer_number
FROM Orders
GROUP BY customer_number
ORDER BY COUNT(*) DESC
LIMIT 1;
```

26. Big Countries

- List countries with population >= 25M or area >= 3M km².

```
SELECT name, population, area
FROM World
WHERE area >= 3000000 OR population >= 25000000;
```

27. Classes More Than 5 Students

- Get classes with at least 5 unique students.

```
SELECT class
FROM Courses
GROUP BY class
HAVING COUNT(DISTINCT student) >= 5;
```

28. Human Traffic of Stadium

- Return records where people count was >= 100 for 3+ consecutive days.

```
SELECT Id
FROM (
  SELECT Id, Lead(People, 1) OVER (ORDER BY Id) AS next1,
              Lead(People, 2) OVER (ORDER BY Id) AS next2
  FROM Stadium
  WHERE People >= 100
) t
WHERE next1 >= 100 AND next2 >= 100;
```

29. Friend Requests I

- Find the user who sent the most friend requests.

```
SELECT requester_id, COUNT(*) AS request_count
FROM RequestAccepted
GROUP BY requester_id
ORDER BY request_count DESC
LIMIT 1;
```

30. Swap Gender Values

- Swap 'm' and 'f' values in a gender column using CASE.

```
UPDATE Students
SET Gender = CASE Gender
  WHEN 'm' THEN 'f'
  WHEN 'f' THEN 'm'
END;
```