▯ SQL Challenge: Top 3 Salaries in Each Department

▯ Problem Statement

You are given two tables:

`Employee` Table:

Column Name	Type
id	int
name	varchar
salary	int
departmentId	int

`Department` Table:

Column Name	Type
id	int
name	varchar

- Each employee belongs to one department.
- Find the **top 3 employees with the highest salaries** in each department.
- If there are **less than 3 employees** in a department, return as many as exist.

▯ Output Format

Department	Employee	Salary
Sales	John	90000
Sales	Alice	80000
Sales	Bob	75000
HR	Claire	60000

▯ SQL Query

```sql
SELECT d.name AS Department, e.name AS Employee, e.salary AS Salary
FROM Employee e
JOIN Department d ON e.departmentId = d.id
WHERE (
    SELECT COUNT(DISTINCT e2.salary)
    FROM Employee e2
    WHERE e2.departmentId = e.departmentId AND e2.salary > e.salary
) < 3;
```

Understanding the Problem

We need to:

1. Find the employees with the top 3 highest salaries in each department
2. Include all employees if a department has fewer than 3 employees
3. Display the department name, employee name, and salary in the results

Breaking Down the SQL Query

```sql
SELECT d.name AS Department, e.name AS Employee, e.salary AS Salary
FROM Employee e
JOIN Department d ON e.departmentId = d.id
WHERE (
    SELECT COUNT(DISTINCT e2.salary)
    FROM Employee e2
    WHERE e2.departmentId = e.departmentId AND e2.salary > e.salary
) < 3;
```

This query may look complex at first, but it uses a clever approach. Let's break it down:

1. The Main Query Structure

```sql
SELECT d.name AS Department, e.name AS Employee, e.salary AS Salary
FROM Employee e
JOIN Department d ON e.departmentId = d.id
WHERE (subquery condition);
```

- We're selecting the department name, employee name, and salary
- We're joining the Employee and Department tables to get the department names
- The WHERE clause will filter employees based on a subquery

2. The JOIN Operation

```
FROM Employee e
JOIN Department d ON e.departmentId = d.id
```

This simply connects each employee to their department using the departmentId foreign key.

3. The Critical Subquery

```sql
WHERE (
    SELECT COUNT(DISTINCT e2.salary)
    FROM Employee e2
    WHERE e2.departmentId = e.departmentId AND e2.salary > e.salary
) < 3;
```

This is the most important part:

- For each employee (e) in the outer query, we're counting how many DISTINCT salaries are HIGHER than their salary within the SAME department
- If that count is less than 3, we include the employee in our results

4. How the Subquery Works

For each employee in the main query:

1. The subquery finds all other employees in the same department
2. It counts only those with HIGHER salaries
3. It counts DISTINCT salaries (to handle ties correctly)
4. If fewer than 3 distinct higher salaries exist, the employee is in the top 3

Walking Through an Example

Let's use a simple example:

Department Table: | id | name | |----|-------| | 1 | Sales | | 2 | HR |

Employee Table: | id | name | salary | departmentId | |----|-------|--------|-------------| | 1 | John | 90000 | 1 | | 2 | Alice | 80000 | 1 | | 3 | Bob | 75000 | 1 | | 4 | David | 75000 | 1 | | 5 | Claire| 60000 | 2 |

Let's trace through how the query processes this data:

1. For John (salary 90000) in Sales:
    - Count of higher salaries in Sales: 0
    - 0 < 3, so include John
2. For Alice (salary 80000) in Sales:
    - Count of DISTINCT higher salaries in Sales: 1 (John's 90000)
    - 1 < 3, so include Alice
3. For Bob (salary 75000) in Sales:
    - Count of DISTINCT higher salaries in Sales: 2 (John's 90000, Alice's 80000)
    - 2 < 3, so include Bob
4. For David (salary 75000) in Sales:
    - Count of DISTINCT higher salaries in Sales: 2 (John's 90000, Alice's 80000)
    - 2 < 3, so include David too (notice both Bob and David are included because they're tied)
5. For Claire (salary 60000) in HR:
    - Count of higher salaries in HR: 0
    - 0 < 3, so include Claire

The final result would be:

Department	Employee	Salary
Sales	John	90000

Sales	Alice	80000
Sales	Bob	75000
Sales	David	75000
HR	Claire	60000

Notice that both Bob and David are included because they have the same salary and are both in the top 3 salary ranges.

Key Concepts for Beginners

1. **Correlated Subquery**: The inner query references the outer query (e2.departmentId = e.departmentId)
2. **DISTINCT**: Used to count unique salary values, which handles ties correctly
3. **Comparison Logic**: Instead of directly finding the top 3, we're checking how many salaries are higher
4. **JOIN**: Combining the Employee and Department tables to get department names

Alternative Approach: Using Window Functions

If your SQL system supports window functions, there's an alternative approach using `DENSE_RANK()`:

```sql
SELECT Department, Employee, Salary
FROM (
    SELECT
        d.name AS Department,
        e.name AS Employee,
        e.salary AS Salary,
        DENSE_RANK() OVER (PARTITION BY e.departmentId ORDER BY e.salary DESC) AS salary_rank
    FROM Employee e
    JOIN Department d ON e.departmentId = d.id
) ranked
WHERE salary_rank <= 3;
```

This approach:

1. Assigns a rank to each employee within their department based on salary
2. Uses DENSE_RANK() to handle ties (employees with the same salary get the same rank)
3. Filters to only include employees with rank 1, 2, or 3

The subquery approach in the original solution works on all SQL systems, even those without window function support.