SQL Challenge: Find Median (or Near Median) Salaries Per Company

Problem Statement

You are given a table `Employee` with employee salary details. Your task is to find **one representative employee** per company whose salary is **at or near the median** salary for that company.

- If there are multiple employees at or near the median, return just one per `salary, company` pair.
- "Near median" is defined as being within 1 rank difference from the true median salary rank for that company.

Input Table: `Employee`

id	company	salary
...	...	...

(Assume there are multiple entries for each company with varying salaries.)

Output

id	company	salary
...	...	...

Each row corresponds to a single employee per company whose salary is closest to the company's median.

SQL Solution

```
SELECT
    id,
    company,
    salary
FROM (
    SELECT
        id,
        company,
        salary,
        ROW_NUMBER() OVER (
            PARTITION BY salary, company
            ORDER BY salary, company
        ) AS rn
    FROM (
        SELECT
            id,
```

```
            company,
            salary,
            ROW_NUMBER() OVER (
                PARTITION BY company
                ORDER BY salary ASC
            ) AS ROWASC,
            ROW_NUMBER() OVER (
                PARTITION BY company
                ORDER BY salary DESC
            ) AS ROWDESC
        FROM Employee
    ) X
    WHERE ROWASC IN (ROWDESC, ROWDESC - 1, ROWDESC + 1)
) tmp
WHERE rn = 1
ORDER BY company, salary;
```

Understanding the Problem

We need to:

1. Find the median salary for each company
2. If there's an exact median, return an employee with that salary
3. If no exact median, return an employee whose salary is "near" the median (within 1 rank)
4. If multiple employees have the same median/near-median salary, return just one

Breaking Down the Solution

This SQL solution uses nested subqueries and window functions to solve the problem. Let's break it down step by step:

Step 1: Assign Row Numbers in Both Directions

The innermost subquery assigns row numbers to each employee within their company, both in ascending and descending order by salary:

```
SELECT
    id,
    company,
    salary,
    ROW_NUMBER() OVER (
        PARTITION BY company
        ORDER BY salary ASC
    ) AS ROWASC,
    ROW_NUMBER() OVER (
        PARTITION BY company
        ORDER BY salary DESC
    ) AS ROWDESC
FROM Employee
```

Let's see what this might produce with some sample data:

id	company	salary	ROWASC	ROWDESC
1	A	1000	1	5
2	A	2000	2	4
3	A	3000	3	3
4	A	4000	4	2
5	A	5000	5	1
6	B	1500	1	4
7	B	2500	2	3
8	B	3500	3	2
9	B	4500	4	1

Step 2: Identify Median and Near-Median Salaries

The second subquery filters for rows where the ascending rank matches the descending rank (exact median) or is within 1 of the descending rank (near median):

```
WHERE ROWASC IN (ROWDESC, ROWDESC - 1, ROWDESC + 1)
```

For Company A with 5 employees:

- Employee with id=3 has ROWASC=3 and ROWDESC=3, so it's the exact median
- The condition becomes: 3 IN (3, 2, 4) which is TRUE

For Company B with 4 employees:

- Employee with id=7 has ROWASC=2 and ROWDESC=3
- The condition becomes: 2 IN (3, 2, 4) which is TRUE
- Employee with id=8 has ROWASC=3 and ROWDESC=2
- The condition becomes: 3 IN (2, 1, 3) which is TRUE

So after this filter, we might have:

id	company	salary	ROWASC	ROWDESC
3	A	3000	3	3
7	B	2500	2	3
8	B	3500	3	2

Notice that for company B, we have two potential near-median employees. We need to pick just one.

Step 3: Handle Duplicate Median Salaries

The next layer assigns another row number to handle cases where multiple employees have the same salary that qualifies as median or near-median:

```
SELECT
    id,
```

```
    company,
    salary,
    ROW_NUMBER() OVER (
        PARTITION BY salary, company
        ORDER BY salary, company
    ) AS rn
```

This ensures we pick just one employee per salary and company combination. If multiple employees have the exact same median-qualifying salary, we'll only pick one of them.

For our sample data (assuming no duplicate salaries):

id	company	salary	rn
3	A	3000	1
7	B	2500	1
8	B	3500	1

Step 4: Select Only One Representative Per Salary-Company Pair

The outermost query filters to only include rows where rn=1, ensuring we get just one employee per salary-company combination:

```
WHERE rn = 1
```

Since company B had two near-median salaries, we need a way to pick just one. The query doesn't specify which one to choose in case of ties (both are equally close to the median), so it would return one based on the sort order in the ROW_NUMBER function.

Step 5: Final Result

The final query returns the id, company, and salary for the selected employees, ordered by company and salary:

```
SELECT
    id,
    company,
    salary
FROM (...) tmp
WHERE rn = 1
ORDER BY company, salary;
```

For our sample data, the result would be:

id	company	salary
3	A	3000
7	B	2500

Company B had two near-median salaries (2500 and 3500), but based on the ordering, we only kept the one with the lower salary.

Why This Works (Detailed Explanation)

Finding the True Median

The key insight is that for the true median in a set, the position from the bottom equals the position from the top. For example:

- In a set of 5 items, the middle item (3rd) is 3 positions from the top and 3 positions from the bottom
- In a set of 6 items, there's no true middle, but items 3 and 4 are closest

By comparing ROWASC and ROWDESC, we can identify these cases:

- When ROWASC = ROWDESC: This is the exact median (odd number of employees)
- When ROWASC = ROWDESC±1: This is near the median (even number of employees)

Handling Multiple Employees with the Same Salary

The additional ROW_NUMBER() function partitioned by salary and company ensures we don't return multiple employees with the same salary from the same company.

Real-World Applications

This type of query is useful for:

- Finding representative salaries for market analysis
- Identifying median compensation for HR reporting
- Finding a "typical" example from a dataset for demonstration purposes

Alternative Approaches

A simpler approach using SQL's percentile functions (available in some SQL dialects):

```
WITH MedianSalaries AS (
    SELECT
        company,
        PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY salary) AS median_salary
    FROM Employee
    GROUP BY company
),
RankedEmployees AS (
    SELECT
        e.id,
        e.company,
        e.salary,
        ABS(e.salary - ms.median_salary) AS distance_from_median,
        ROW_NUMBER() OVER (
            PARTITION BY e.company
            ORDER BY ABS(e.salary - ms.median_salary), e.salary
        ) AS rn
    FROM Employee e
    JOIN MedianSalaries ms ON e.company = ms.company
)
SELECT
    id,
```

```
    company,
    salary
FROM RankedEmployees
WHERE rn = 1
ORDER BY company, salary;
```

This approach:

1. Calculates the true median salary for each company
2. Finds the employee whose salary is closest to the median
3. If multiple employees are equally close, picks the one with the lower salary

The original solution is more complex but works in more SQL dialects, including those that don't support percentile functions.