569. Median Employee Salary

May 13, 2017 | 20.6K views

Average Rating: 2.96 (23 votes)

The `Employee` table holds all employees. The employee table has three columns: Employee Id, Company Name, and Salary.

```
+-----+------------+--------+
|Id   | Company    | Salary |
+-----+------------+--------+
|1    | A          | 2341   |
|2    | A          | 341    |
|3    | A          | 15     |
|4    | A          | 15314  |
|5    | A          | 451    |
|6    | A          | 513    |
|7    | B          | 15     |
|8    | B          | 13     |
|9    | B          | 1154   |
|10   | B          | 1345   |
|11   | B          | 1221   |
|12   | B          | 234    |
|13   | C          | 2345   |
|14   | C          | 2645   |
|15   | C          | 2645   |
|16   | C          | 2652   |
|17   | C          | 65     |
+-----+------------+--------+
```

Write a SQL query to find the median salary of each company. Bonus points if you can solve it without using any built-in SQL functions.

```
+-----+------------+--------+
|Id   | Company    | Salary |
+-----+------------+--------+
|5    | A          | 451    |
|6    | A          | 513    |
|12   | B          | 234    |
|9    | B          | 1154   |
|14   | C          | 2645   |
+-----+------------+--------+
```

Solution

Approach #1: Using the definition of *median* [Accepted]

Intuition

By the definition of *median*, the count of the bigger numbers than itself should be equal to the count of the smaller ones in an *odd* array.

Algorithm

Take array [1,3,2] for example, is the first number 1 the median? No, because this array only have 3 elements but there are 2 of them (3, 2) are greater than 1. To continue, we know 3 is not the median as well since there are 2 elements smaller. But for the last element 2, there are equal amount of bigger and smaller numbers. So it is the median in this array!

What if an array has *even* amount of distinct values, the median is the average of the middle *two elements* next to each other after sorting this array. It is not hard to understand that for either of these two elements, the difference (absolute value) of its bigger and smaller number than itself in this array is 1, which is the exactly frequency of a element in the distinct array.

So in general, the median's frequency should be equal or grater than the absolute difference of its bigger elements and small ones in an array no matter whether it has odd or even amount of numbers and whether they are distinct. This rule is the key, and it is represented as the following code.

```
SUM(CASE
    WHEN Employee.Salary = alias.Salary THEN 1
    ELSE 0
END) >= ABS(SUM(SIGN(Employee.Salary - alias.Salary)))
```

Thus, this approach is as following in MySQL code.

MySQL

```
SELECT
    Employee.Id, Employee.Company, Employee.Salary
FROM
    Employee,
    Employee alias
WHERE
    Employee.Company = alias.Company
GROUP BY Employee.Company , Employee.Salary
HAVING SUM(CASE
    WHEN Employee.Salary = alias.Salary THEN 1
    ELSE 0
END) >= ABS(SUM(SIGN(Employee.Salary - alias.Salary)))
ORDER BY Employee.Id
;
```

> Note: In MySQL 5.6, this code runs fine, but if you are using MySQL 5.7+, please use `ANY_VALUE(Employee.Id)` instead of `Employee.Id` in the SELECT statement. Otherwise, you may encouter the following error message: Error Code: 1055. Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'Employee.Id' which is not functionally dependent on columns in GROUP BY clause; this is incompatible with sql_mode=only_full_group_by. For more details on how to user `ANY_VALUE(arg)`, please refer to this link.

Approach #2: Sort and then select the *median* [Accepted]

Intuition

In general, we can just pick the middle one(s) to get the *median* if the records are ranked by **salary**. But how can we get them sorted particularly MySQL does not have the build-in rank function, and these are several companies in this case.

Algorithm

By adding a virtual column to simulate the ranking, we can sort these records by **salary** and pick up the middle one(s). Here we need to use the session variable to achieve this goal.

This approach is more efficient than the first one since it does not calculate all the **salary** one by one in the table.

```
SELECT
    Id, Company, Salary
FROM
    (SELECT
        e.Id,
            e.Salary,
            e.Company,
            IF(@prev = e.Company, @Rank:=@Rank + 1, @Rank:=1) AS rank,
            @prev:=e.Company
    FROM
        Employee e, (SELECT @Rank:=0, @prev:=0) AS temp
    ORDER BY e.Company , e.Salary , e.Id) Ranking
        INNER JOIN
    (SELECT
        COUNT(*) AS totalcount, Company AS name
    FROM
        Employee e2
    GROUP BY e2.Company) companycount ON companycount.name = Ranking.Company
WHERE
    Rank = FLOOR((totalcount + 1) / 2)
        OR Rank = FLOOR((totalcount + 2) / 2)
;
```

Rate this article:

Previous Next

Comments: 35 Sort By

Type comment here... (Markdown is supported)

Preview Post

henrykira ★ 190 January 23, 2019 8:49 PM
I can't understand this statement 'It is not hard to understand that for either of these two elements, the difference (absolute value) of its bigger and smaller number than itself in this array is 1, which is the exactly frequency of a element in the distinct array.'

Can someone translate it into better English?
Read More
43 Share Reply
SHOW 1 REPLY

pachiappan ★ 12 May 23, 2018 5:31 AM
with c as
(select id,
company,
salary,
row_number() over(partition by company order by salary) rno,
Read More
8 Share Reply
SHOW 2 REPLIES

vkhang55 ★ 5 May 11, 2018 4:21 PM
This is my solution without using any variable.

```
SELECT
  ranked_employees.id      AS Id,
  ranked_employees.company AS Company
```

Read More
4 Share Reply

shubhamg823 ★ 2 January 2, 2018 4:48 AM
Please explain this:
the difference (absolute value) of its bigger and smaller number than itself in this array is 1, which is the exactly frequency of a element in the distinct array.
2 Share Reply
SHOW 2 REPLIES

vineeshbnair ★ 2 July 29, 2017 8:43 AM
WITH TMP
AS (SELECT COMPANY, COUNT (1) CNT
FROM EMPLOYEE
GROUP BY COMPANY)
SELECT E.*
Read More
2 Share Reply
SHOW 1 REPLY

wyue5 ★ 3 September 28, 2018 3:26 AM
Why we should use group by Employee.Salary in the first solution?
3 Share Reply

frankjintwo ★ 1 September 6, 2019 4:08 PM
My solution:

1. rank and have row number.
2. company row number with avg row number if the abs value is smaller than 1 then it ok.

Read More
1 Share Reply

Elaine66 ★ 10 August 2, 2019 10:40 AM
use row_number, no join at all.

select id, company, salary
from
(select Employee.,
Read More
1 Share Reply

errandel ★ 1 November 17, 2017 4:09 PM
@Mr.Bin:
regarding approach 2's session variables (@rank, @prev), doesn't the select clause occur before the order by clause? How is it that the ranking changes when you call order by? I thought the rank is evaluated first, then order by occurs.
1 Share Reply

shquq23 ★ 1 July 12, 2019 11:06 AM
T-SQL:

```
-- Sort all records for each company
with rn as (
select Id, Company,
```

Read More
0 Share Reply
SHOW 1 REPLY