**Multiply ways display the highest and lowest salary from the table**



**Create table:**

create table employee

(

id int primary key GENERATED ALWAYS AS IDENTITY,

name varchar(100),

dept varchar(100),

salary int

);

insert into employee values(default, 'Alexander', 'Admin', 6500);

insert into employee values(default, 'Leo', 'Finance', 7000);

insert into employee values(default, 'Robin', 'IT', 2000);

insert into employee values(default, 'Ali', 'IT', 4000);

insert into employee values(default, 'Maria', 'IT', 6000);

insert into employee values(default, 'Alice', 'Admin', 5000);

insert into employee values(default, 'Sebastian', 'HR', 3000);

insert into employee values(default, 'Emma', 'Finance', 4000);

insert into employee values(default, 'John', 'HR', 4500);

insert into employee values(default, 'Kabir', 'IT', 8000);

**Display table**

select \* from employee;

A screenshot of a computer

Description automatically generated

**Method I**

SELECT \* FROM employee

WHERE salary IN (SELECT MAX(salary) AS max\_salary FROM employee)

UNION ALL

SELECT \* FROM employee

WHERE salary IN (SELECT MIN(salary) AS min\_salary FROM employee)

ORDER BY salary DESC;

**Result:**

A screenshot of a computer

Description automatically generated

1. **Subqueries for Maximum and Minimum Salaries:**
   * **(SELECT MAX(salary) AS max\_salary FROM employee)**: This subquery calculates the maximum salary in the "employee" table.
   * **(SELECT MIN(salary) AS min\_salary FROM employee)**: This subquery calculates the minimum salary in the "employee" table.
2. **Main Queries with IN Clause:**
   * **SELECT \* FROM employee WHERE salary IN (...)**: These two main queries use the **IN** clause to filter rows where the salary matches either the maximum or minimum salary obtained from the subqueries.
     + The first query retrieves all rows where the salary is equal to the maximum salary.
     + The second query retrieves all rows where the salary is equal to the minimum salary.
3. **Combining Results with UNION ALL:**
   * **UNION ALL** combines the results of the two main queries. **UNION ALL** includes all rows, even if there are duplicates.
4. **Ordering Results by Salary in Descending Order:**
   * **ORDER BY salary DESC** arranges the final result set in descending order based on the salary column.

In summary, the query retrieves all rows from the "employee" table where the salary matches either the maximum or minimum salary. The result is a union of these two sets of rows, and the final result set is ordered by salary in descending order. This way, you get all employees with the highest and lowest salaries combined in a single result set.

**Method II**

SELECT \*

FROM (

SELECT

employee.\*,

RANK() OVER (ORDER BY salary DESC) as rank\_desc,

RANK() OVER (ORDER BY salary ASC) as rank\_asc

FROM employee

) ranked

WHERE rank\_desc = 1 OR rank\_asc = 1;

**Result:**

A screenshot of a phone

Description automatically generated

1. **Inner Subquery (Derived Table):**
   * The inner subquery selects all columns from the "employee" table.
   * It uses the **RANK()** window function twice, once for descending order (**rank\_desc**) and once for ascending order (**rank\_asc**), ranking the rows based on the **salary** column.
2. **Outer Query:**
   * The outer query selects all columns from the derived table named **ranked**.
   * The **WHERE** clause filters the results to include only rows where either **rank\_desc** or **rank\_asc** is equal to 1. This ensures that only the rows with the highest or lowest salary are included.

In summary, the query uses window functions to rank rows based on salary in both descending and ascending order. The results are then filtered to include only the rows with the highest and lowest salaries. This query effectively retrieves the employees with the maximum and minimum salaries from the "employee" table.

**Method III**

(

SELECT \*

FROM employee

ORDER BY salary DESC

LIMIT 1

)

UNION ALL

(

SELECT \*

FROM employee

ORDER BY salary ASC

LIMIT 1

)

**Result:**

A screenshot of a computer

Description automatically generated

1. **First Subquery:**
   * **(SELECT \* FROM employee ORDER BY salary DESC LIMIT 1)**: This subquery selects all columns from the "employee" table, orders the result by salary in descending order (**ORDER BY salary DESC**), and limits the result to only one row (**LIMIT 1**). It effectively retrieves the row with the highest salary.
2. **Second Subquery:**
   * **(SELECT \* FROM employee ORDER BY salary ASC LIMIT 1)**: This subquery is similar to the first one but orders the result by salary in ascending order (**ORDER BY salary ASC**), effectively retrieving the row with the lowest salary.
3. **UNION ALL Operator:**
   * **UNION ALL** is used to combine the results of the two subqueries. Unlike **UNION**, **UNION ALL** does not eliminate duplicate rows. In this case, duplicates are not an issue because each subquery is using **LIMIT 1**, so there can be at most one row from each.

In summary, this query retrieves two rows—one with the highest salary and another with the lowest salary—from the "employee" table. The result is a union of these two rows, and the order of the rows is determined by the ordering in the individual subqueries.

---Display highest and lowest salary per department---

SELECT

dept,

MIN(salary) AS min\_salary,

MAX(salary) AS max\_salary

FROM employee

GROUP BY dept

ORDER BY dept;

-----------------------------

I used PostgreSQL for above queries.