10/03

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Different Methods to Find the Nth Highest/Lowest Salary in SQL (MSSQL Server)**

There are multiple ways to find the **Nth highest or lowest salary**, each suitable for different use cases. Below are the most efficient approaches:

**1️⃣ Using OFFSET FETCH (Best for Large Data)**

✅ **Best for performance when salary is indexed**  
✅ **Simple and efficient**

**Nth Highest Salary**

SELECT salary

FROM (SELECT DISTINCT salary FROM employees) AS distinct\_salaries

ORDER BY salary DESC

OFFSET N-1 ROWS FETCH NEXT 1 ROWS ONLY;

**Nth Lowest Salary**

SELECT salary

FROM (SELECT DISTINCT salary FROM employees) AS distinct\_salaries

ORDER BY salary ASC

OFFSET N-1 ROWS FETCH NEXT 1 ROWS ONLY;

💡 **Replace N with the required rank** (e.g., N=3 for 3rd highest).  
⚡ **Requires indexing on salary for best performance**.

**2️⃣ Using DENSE\_RANK() (Best for Full Employee Details)**

✅ **Returns the full employee details (name, department, etc.)**  
✅ **Handles duplicate salaries properly**

**Nth Highest Salary**

WITH SalaryRank AS (

SELECT \*, DENSE\_RANK() OVER (ORDER BY salary DESC) AS rnk

FROM employees

)

SELECT \* FROM SalaryRank WHERE rnk = N;

**Nth Lowest Salary**

WITH SalaryRank AS (

SELECT \*, DENSE\_RANK() OVER (ORDER BY salary ASC) AS rnk

FROM employees

)

SELECT \* FROM SalaryRank WHERE rnk = N;

💡 **Why use DENSE\_RANK()?**

* Keeps **duplicate salaries in ranking**.
* Useful when you need **employee details** with the salary.

**3️⃣ Using ROW\_NUMBER() (If You Need Unique Rows)**

✅ **Ensures unique ranking (no duplicate ranks)**

**Nth Highest Salary**

WITH SalaryRank AS (

SELECT \*, ROW\_NUMBER() OVER (ORDER BY salary DESC) AS rnk

FROM employees

)

SELECT \* FROM SalaryRank WHERE rnk = N;

**Nth Lowest Salary**

WITH SalaryRank AS (

SELECT \*, ROW\_NUMBER() OVER (ORDER BY salary ASC) AS rnk

FROM employees

)

SELECT \* FROM SalaryRank WHERE rnk = N;

💡 **Difference from DENSE\_RANK()?**

* ROW\_NUMBER() assigns a **unique rank** to each row, even if salaries are the same.

**4️⃣ Using TOP WITH TIES (Handles Duplicate Salaries)**

✅ **Keeps all employees with the Nth salary**

**Nth Highest Salary**

SELECT TOP 1 WITH TIES salary

FROM (SELECT DISTINCT salary FROM employees) AS distinct\_salaries

ORDER BY salary DESC

OFFSET N-1 ROWS;

**Nth Lowest Salary**

SELECT TOP 1 WITH TIES salary

FROM (SELECT DISTINCT salary FROM employees) AS distinct\_salaries

ORDER BY salary ASC

OFFSET N-1 ROWS;

💡 **Why use this?**

* Ensures **all employees with the same Nth salary are included**.

**5️⃣ Using SUBQUERY (Simpler, but Less Efficient)**

✅ **Good for small datasets**

**Nth Highest Salary**

SELECT MIN(salary)

FROM (SELECT DISTINCT TOP N salary FROM employees ORDER BY salary DESC) AS temp;

**Nth Lowest Salary**

SELECT MAX(salary)

FROM (SELECT DISTINCT TOP N salary FROM employees ORDER BY salary ASC) AS temp;

⚡ **This method is slower than OFFSET FETCH on large datasets.**

**🚀 Comparison of Methods**

| **Method** | **Best For** | **Handles Duplicates?** | **Performance** |
| --- | --- | --- | --- |
| OFFSET FETCH | Large data | ❌ No | ✅ Fastest with Index |
| DENSE\_RANK() | Employee details | ✅ Yes | ✅ Good |
| ROW\_NUMBER() | Unique ranks | ❌ No | ✅ Good |
| TOP WITH TIES | All employees with Nth salary | ✅ Yes | ✅ Fast |
| SUBQUERY | Small datasets | ✅ Yes | ❌ Slow for big data |

**⚡ Performance Boost: Add an Index**

For **large datasets**, indexing significantly improves performance:

CREATE INDEX idx\_salary ON employees(salary);

Would you like a query that fetches **employees with salaries in the top N% range**? 🚀