RANK Window Function

RANK is a simple analytical function. It does not take any column list arguments, like all other rank analytical functions except NTILE. It returns a **rank** or a number based on the ordering of the rows by some condition.

How to use the RANK function in SQL? Let’s say we want to rank our salespeople based on the number of items they’ve sold. We do this using the RANK function over the “sales\_num” column, which records the number of sales for each rep.  Here’s the code:

|  |
| --- |
| SELECT id,    salesman\_id,    sales\_item,    sales\_num,    sales\_price,    rank() over (order by sales\_num)  FROM sales  ORDER BY sales\_num; |

And the result set is displayed as :

| **ID** | **SALESMAN \_ID** | **SALES \_ITEM** | **SALES \_NUM** | **SALES \_PRICE** | **RANK()OVER (ORDERBYSALES\_NUM)** |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 100 | 4 | 50.5 | 1 |
| 2 | 30 | 200 | 7 | 50 | 2 |
| 5 | 60 | 200 | 8 | 30.5 | 3 |
| 4 | 50 | 200 | 15 | 110 | 4 |
| 6 | 70 | 200 | 22 | 10.5 | 5 |
| 3 | 40 | 200 | 40 | 10.5 | 6 |

ORDER BY is mandatory when using the RANK, unction in SQL, as it displays  records in an ascending or descending order based on our requirements. In the above example, we ranked “sales\_num” in ascending order so the largest sales number has a correspondingly higher rank. (Note: Ascending is the default order in an ORDER BY clause; you do not need to specify it with the ASC keyword.)

If we were to reverse the order by specifying the DESC keyword – which you do have to specify – the sales reps’ ranks would change:

|  |
| --- |
| SELECT id,    salesman\_id,    sales\_item,    sales\_num,    sales\_price,    rank() over (order by sales\_num desc)  FROM sales  ORDER BY sales\_num; |

The results:

| **ID** | **SALESMAN \_ID** | **SALES \_ITEM** | **SALES \_NUM** | **SALES \_PRICE** | **RANK()OVER (ORDERBYSALES\_NUMDESC)** |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 100 | 4 | 50.5 | 6 |
| 2 | 30 | 200 | 7 | 50 | 5 |
| 5 | 60 | 200 | 8 | 30.5 | 4 |
| 4 | 50 | 200 | 15 | 110 | 3 |
| 6 | 70 | 200 | 22 | 10.5 | 2 |
| 3 | 40 | 200 | 40 | 10.5 | 1 |

OFFSET

SELECT product\_name, list\_price FROM production.products ORDER BY list\_price, product\_name;

To skip the first 10 products and return the rest, you use the OFFSET clause

SELECT

    product\_name,

    list\_price

FROM

    production.products

ORDER BY

    list\_price,

    product\_name

OFFSET 10 ROWS;

SELECT product\_name,list\_price FROM production.products ORDER BY

    list\_price,  product\_name

OFFSET 10 ROWS

FETCH NEXT 10 ROWS ONLY;