Introduction to SQL Server RANK() function

The RANK() function is a [window function](https://www.sqlservertutorial.net/sql-server-window-functions/) that assigns a rank to each row within a partition of a result set.

The rows within a partition that have the same values will receive the same rank. The rank of the first row within a partition is one. The RANK() function adds the number of tied rows to the tied rank to calculate the rank of the next row, therefore, the ranks may not be consecutive.

The following shows the syntax of the RANK() function:

RANK() OVER (

[PARTITION BY partition\_expression, ... ]

ORDER BY sort\_expression [ASC | DESC], ...

)

Code language: SQL (Structured Query Language) (sql)

In this syntax:

* First, the PARTITION BY clause divides the rows of the result set partitions to which the function is applied.
* Second, the ORDER BY clause specifies the logical sort order of the rows in each a partition to which the function is applied.

The RANK() function is useful for top-N and bottom-N reports.

SQL Server RANK() illustration

First, [create a new table](https://www.sqlservertutorial.net/sql-server-basics/sql-server-create-table/) named sales.rank\_demo that has one column:

CREATE TABLE sales.rank\_demo (

v VARCHAR(10)

);

Code language: SQL (Structured Query Language) (sql)

Second, [insert](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/) some rows into the sales.rank\_demo table:

INSERT INTO sales.rank\_demo(v)

VALUES('A'),('B'),('B'),('C'),('C'),('D'),('E');

Code language: SQL (Structured Query Language) (sql)

Third, query data from the sales.rank\_demo table:

SELECT v FROM sales.rank\_demo;

Code language: SQL (Structured Query Language) (sql)

Fourth, use the ROW\_NUMBER() to assign ranks to the rows in the result set of sales.rank\_demo table:

SELECT

v,

RANK () OVER (

ORDER BY v

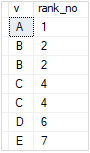
) rank\_no

FROM

sales.rank\_demo;

Code language: SQL (Structured Query Language) (sql)

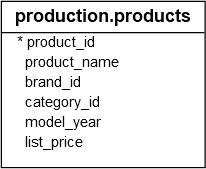
Here is the output:



As shown clearly from the output, the second and third rows receive the same rank because they have the same value B. The fourth and fifth rows get the rank 4 because the RANK() function skips the rank 3 and both of them also have the same values.

SQL Server RANK() function examples

We’ll use the production.products table to demonstrate the RANK() function:



Using SQL Server RANK() function over a result set example

The following example uses the RANK() function to assign ranks to the products by their list prices:

SELECT

product\_id,

product\_name,

list\_price,

RANK () OVER (

ORDER BY list\_price DESC

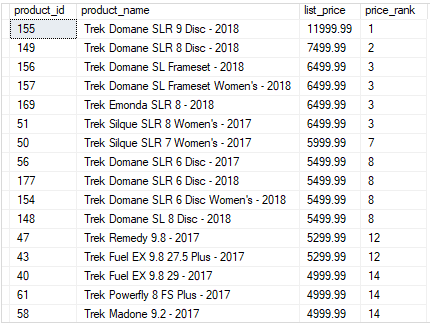
) price\_rank

FROM

production.products;

Code language: SQL (Structured Query Language) (sql)

Here is the result set:



In this example, because we skipped the PARTITION BY clause, the RANK() function treated the whole result set as a single partition.

The RANK() function assigns a rank to each row within the result set sorted by list price from high to low.

Using SQL Server RANK() function over partitions example

This example uses the RANK() function to assign a rank to each product by list price in each brand and returns products with rank less than or equal to three:

SELECT \* FROM (

SELECT

product\_id,

product\_name,

brand\_id,

list\_price,

RANK () OVER (

PARTITION BY brand\_id

ORDER BY list\_price DESC

) price\_rank

FROM

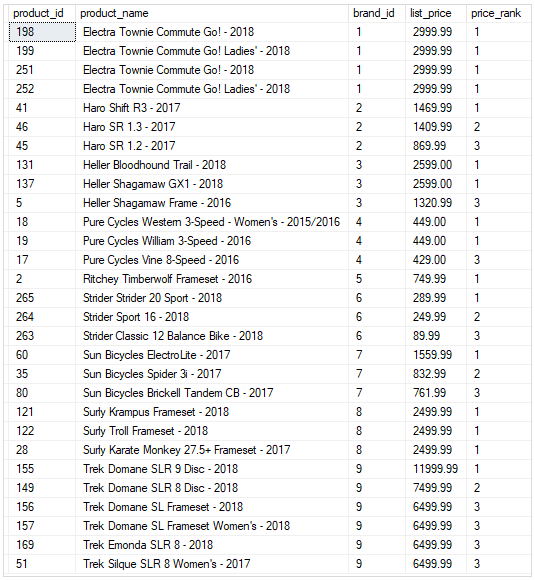
production.products

) t

WHERE price\_rank <= 3;

Code language: SQL (Structured Query Language) (sql)

The following picture shows the output:



In this example:

* First, the PARTITION BY clause divides the products into partitions by brand Id.
* Second, the ORDER BY clause sorts products in each partition by list prices.
* Third, the outer query returns the products whose rank values are less than or equal to three.

The RANK() function is applied to each row in each partition and reinitialized when crossing the partition’s boundary.