SQL Server FIRST\_VALUE() function overview

The FIRST\_VALUE() function is a [window function](https://www.sqlservertutorial.net/sql-server-window-functions/) that returns the first value in an ordered partition of a result set.

The following shows the syntax of the FIRST\_VALUE() function:

FIRST\_VALUE ( scalar\_expression )

OVER (

[PARTITION BY partition\_expression, ... ]

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

)

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

In this syntax:

scalar\_expression

scalar\_expression is an expression evaluated against the value of the first row of the ordered partition of a result set. The scalar\_expression can be a column, [subquery](https://www.sqlservertutorial.net/sql-server-basics/sql-server-subquery/), or expression that evaluates to a single value. It cannot be a window function.

PARTITION BY clause

The PARTITION BY clause distributes rows of the result set into partitions to which the FIRST\_VALUE() function is applied. If you skip the PARTITION BY clause, the FIRST\_VALUE() function will treat the whole result set as a single partition.

ORDER BY clause

The ORDER BY clause specifies the logical order of the rows in each partition to which the FIRST\_VALUE()function is applied.

rows\_range\_clause

The rows\_range\_clause further limits the rows within the partition by defining start and end points.

SQL Server FIRST\_VALUE() function examples

The following statement creates a new view named sales.vw\_category\_sales\_volume that returns the number of products sold by product category and year.

CREATE VIEW

sales.vw\_category\_sales\_volume

AS

SELECT

category\_name,

YEAR(order\_date) year,

SUM(quantity) qty

FROM

sales.orders o

INNER JOIN sales.order\_items i

ON i.order\_id = o.order\_id

INNER JOIN production.products p

ON p.product\_id = i.product\_id

INNER JOIN production.categories c

ON c.category\_id = p.product\_id

GROUP BY

category\_name,

YEAR(order\_date);

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

Here is the data from the view:

SELECT

\*

FROM

sales.vw\_category\_sales\_volume

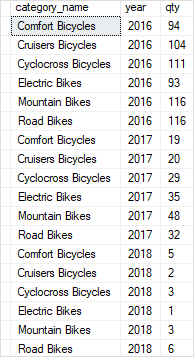
ORDER BY

year,

category\_name,

qty;

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



A) Using FIRST\_VALUE() over a result set example

This example uses FIRST\_VALUE() function to return category name with the lowest sales volume in 2017:

SELECT

category\_name,

year,

qty,

FIRST\_VALUE(category\_name) OVER(

ORDER BY qty

) lowest\_sales\_volume

FROM

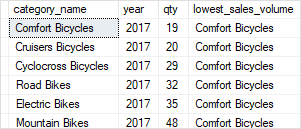
sales.vw\_category\_sales\_volume

WHERE

year = 2017;

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

Here is the output:



In this example:

* The PARTITION BY clause was not specified therefore the whole result set was treated as a single partition.
* The ORDER BY clause sorted rows in each partition by quantity (qty) from low to high.

B) Using FIRST\_VALUE() over partitions example

The following example uses the FIRST\_VALUE() function to return product categories with the lowest sales volumes in 2016 and 2017.

SELECT

category\_name,

year,

qty,

FIRST\_VALUE(category\_name) OVER(

`PARTITION BY` year

ORDER BY qty

) lowest\_sales\_volume

FROM

sales.vw\_category\_sales\_volume

WHERE

year BETWEEN 2016 AND 2017;

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

The following picture shows the output:



In this example:

* The PARTITION BY clause distributed rows by year into two partitions, one for 2016 and the other for 2017.
* The ORDER BY clause sorted rows in each partition by quantity (qty) from low to high.
* The FIRST\_VALUE() function is applied to each partition separately. For the first partition, it returned Electric Bikes and for the second partition it returned Comfort Bicycles because these categories were the first rows in each partition.