Overview of SQL Server LAG() function

SQL Server LAG() is a [window function](https://www.sqlservertutorial.net/sql-server-window-functions/) that provides access to a row at a specified physical offset which comes before the current row.

In other words, by using the LAG() function, from the current row, you can access data of the previous row, or the row before the previous row, and so on.

The LAG() function can be very useful for comparing the value of the current row with the value of the previous row.

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

LAG(return\_value ,offset [,default])

OVER (

[PARTITION BY partition\_expression, ... ]

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

)

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

In this syntax:

 return\_value

The return value of the previous row based on a specified offset. The return value must evaluate to a single value and cannot be another window function.

 offset

The number of rows back from the current row from which to access data.  offset can be an expression, [subquery](https://www.sqlservertutorial.net/sql-server-basics/sql-server-subquery/), or column that evaluates to a positive integer.

The default value of offset is 1 if you don’t specify it explicitly.

 default

default is the value to be returned if offset goes beyond the scope of the partition. It defaults to NULL if it is not specified.

 PARTITION BY clause

The PARTITION BY clause distributes rows of the result set into partitions to which the LAG() function is applied.

If you omit the PARTITION BY clause, the 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 LAG() function is applied.

SQL Server LAG() function examples

We will reuse the view sales.vw\_netsales\_brands created in the LEAD() function tutorial for the demonstration.

The following query shows the data from the sales.vw\_netsales\_brands view:

SELECT

\*

FROM

sales.vw\_netsales\_brands

ORDER BY

year,

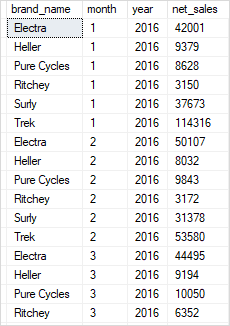
month,

brand\_name,

net\_sales;

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

Here is the output:



A) Using SQL Server LAG() function over a result set example

This example uses the LAG() function to return the net sales of the current month and the previous month in the year 2018:

WITH cte\_netsales\_2018 AS(

SELECT

month,

SUM(net\_sales) net\_sales

FROM

sales.vw\_netsales\_brands

WHERE

year = 2018

GROUP BY

month

)

SELECT

month,

net\_sales,

LAG(net\_sales,1) OVER (

ORDER BY month

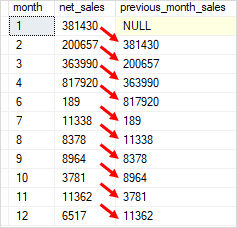
) previous\_month\_sales

FROM

cte\_netsales\_2018;

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

Here is the output:



In this example:

* First, the CTE returns net sales aggregated by month.
* Then, the outer query uses the LAG() function to return sales of the previous month.

B) Using SQL Server LAG() function over partitions example

The following statement uses the LAG() function to compare the sales of the current month with the previous month of each brand in the year 2018:

SELECT

month,

brand\_name,

net\_sales,

LAG(net\_sales,1) OVER (

PARTITION BY brand\_name

ORDER BY month

) next\_month\_sales

FROM

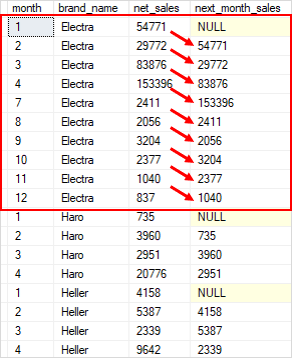
sales.vw\_netsales\_brands

WHERE

year = 2018;

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

This picture shows the output:



In this example:

* The PARTITION BY clause divided rows into partitions by brand name.
* For each partition (or brand name), the ORDER BY clause sorts the rows by month.
* For each row in each partition, the LAG() function returns the net sales of the previous row.

To compare the sales of the current month with the previous month of net sales by brand in 2018, you use the following query:

WITH cte\_sales AS (

SELECT

month,

brand\_name,

net\_sales,

LAG(net\_sales,1) OVER (

PARTITION BY brand\_name

ORDER BY month

) previous\_sales

FROM

sales.vw\_netsales\_brands

WHERE

year = 2018

)

SELECT

month,

brand\_name,

net\_sales,

previous\_sales,

FORMAT(

(net\_sales - previous\_sales) / previous\_sales,

'P'

) vs\_previous\_month

FROM

cte\_sales;

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

The output is as follows:

