**SQL\_Window Functions**

Window functions apply aggregate and ranking functions over a particular window (set of rows). OVER clause is used with window functions to define that window. OVER clause does two things:

* Partitions rows into form set of rows. (PARTITION BY clause is used)
* Orders rows within those partitions into a particular order. (ORDER BY clause is used)



**Basic Syntax:**

SELECT coulmn\_name1,

window\_function(cloumn\_name2),

OVER([PARTITION BY column\_name1] [ORDER BY column\_name3]) AS new\_column

FROM table\_name;

**window\_function=** any aggregate or ranking function

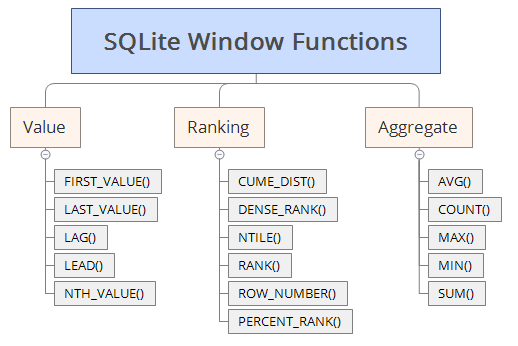
**column\_name1**= column to be selected

**coulmn\_name2=** column on which window function is to be applied

**column\_name3**= column on whose basis partition of rows is to be done

**new\_column=** Name of new column

**table\_name=** Name of table



**Aggregate Window Functions:**

Various aggregate functions such as SUM(), COUNT(), AVERAGE(), MAX(), MIN() applied over a particular window (set of rows) are called aggregate window functions.

Consider the following **employee** table:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Age | Department | Salary |
| Simran | 20 | Technical | 50000 |
| Harsh | 22 | Marketing | 40000 |
| Abhishek | 21 | Marketing | 30000 |
| Tanuja | 20 | Technical | 45000 |
| Vedant | 21 | Marketing | 35000 |

**Example –**

Find average salary of employees for each department and order employees within a department by age.

SELECT Name, Age, Department, Salary,

AVERAGE(Salary) OVER( PARTITION BY Department ORDER BY Age) AS Avg\_Salary

FROM employee;

The output of above query will be:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Age | Department | Salary | Avg\_Salary |
| Abhishek | 21 | Marketing | 30000 | 35000 |
| Vedant | 21 | Marketing | 35000 | 35000 |
| Harsh | 22 | Marketing | 40000 | 35000 |
| Simran | 20 | Technical | 50000 | 47500 |
| Tanuja | 20 | Technical | 45000 | 47500 |

**Q1. Find out the number of employees in each department in the order of decreasing salary.**

**SELECT Name, Age, Department, Salary,**

**count(Name) OVER( PARTITION BY Department ORDER BY Salary Desc) AS Employee\_Count**

**FROM employee;**

**Ranking Window Functions:**

* **RANK() –**   
  As the name suggests, the rank function assigns rank to all the rows within every partition.
* **DENSE\_RANK() –**   
  It assigns rank to each row within partition. The difference between RANK() and DENSE\_RANK() is that in DENSE\_RANK(), for the next rank after two same rank, consecutive integer is used, no rank is skipped.
* **ROW\_NUMBER() –**   
  It assigns consecutive integers to all the rows within partition. Within a partition, no two rows can have same row number.

**Q2. Calculate row no., rank, dense rank of employees is employee table according to salary within each department.**

**SELECT row\_number() OVER( PARTITION BY Department ORDER BY Salary Desc) AS Roll\_NO\_By\_Dept,Name , Age, Department, Salary,**

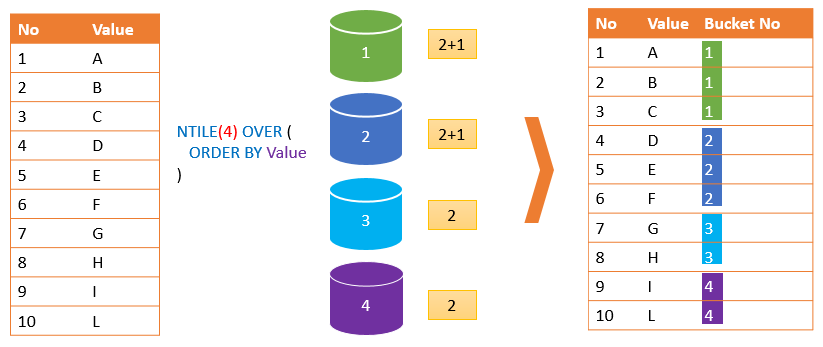
**Rank() OVER( PARTITION BY Department ORDER BY Salary Desc) AS Employee\_Rank,**

**Dense\_Rank() OVER( PARTITION BY Department ORDER BY Salary Desc) AS Employee\_Dense\_Rank**

**FROM employee;**

* **NTILE () –**

It distributes rows of an ordered partition into a pre-defined number of roughly equal groups.



**Syntax :**

NTILE(number\_expression) OVER (

[PARTITION BY partition\_expression ]

ORDER BY sort\_expression [ASC | DESC]

)

**Q3. Use NTILE() function to divide above rows into 2 groups.**

**SELECT Name, Age, Department, Salary,**

**ntile(2) OVER( ) AS Employee\_Group**

**FROM employee;**

* **CUME\_DIST () –**

The CUME\_DIST() is a [window function](https://www.mysqltutorial.org/mysql-window-functions/) that returns the cumulative distribution of a value within a set of values. It represents the number of rows with values less than or equal to that row’s value divided by the total number of rows.

**Syntax:**

CUME\_DIST() OVER (

PARTITION BY expr, ...

ORDER BY expr [ASC | DESC], ...

)

**Q4. Apply the cume\_dist function to the salary column.**

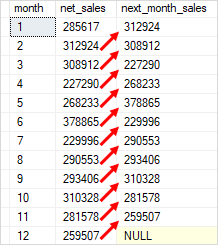
**SELECT Name, Age, Department, Salary,**

**cume\_dist() OVER( PARTITION BY Department ORDER BY Salary) AS Cumulative**

**FROM employee;**

**Value Window Functions:**

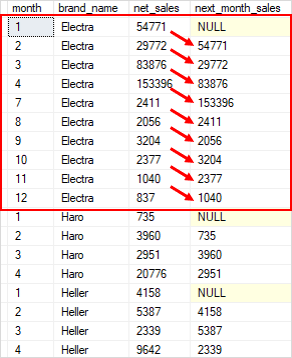
* **LEAD() –**   
  The LEAD() function in MySQL are used to get preceding value of any row within its partition.



**Syntax:** 

LEAD(expr, N, default) OVER (Window\_specification | Window\_name)

* **LAG() –**   
  The LAG() function in MySQL are used to get succeeding value of any row within its partition.



**Syntax:**

LAG(expr, N, default) OVER (Window\_specification | Window\_name)

**Q5. Assign the salary of the next employee to the current employee.**

**SELECT Name, Age, Department, Salary,**

**lead(salary,1) OVER( ORDER BY Salary) AS NExt\_salary**

**FROM employee;**

**Q6. Assign the salary of the previous employee to the current employee.**

SELECT Name, Age, Department, Salary,

Lag(salary,1) OVER( ORDER BY Salary) AS Previous\_salary

FROM employee;

* **FIRST\_VALUE() –**   
  It returns the first value within partition.
* **NTH\_VALUE()-**

It returns the nth value within partition.

**Q7. Find the employee having the lowest salary from each department.**

**SELECT Name, Age, Department, Salary,**

**first\_value(Salary) OVER( PARTITION BY Department ORDER BY Salary) AS Lowest\_salary**

**FROM employee;**

**Q8. Find the employee having the 2nd lowest salary from each department.**

**SELECT Name, Age, Department, Salary,**

**nth\_value(Salary,2) OVER( PARTITION BY Department ORDER BY Salary) AS 2nd\_Lowest\_salary**

**FROM employee;**

****