**MySQL Window Functions**

1. MySQL has supported window functions since version 8.0. The window functions allow you to solve query problems in new, easier ways and with better performance.
2. A window function performs an aggregate-like operation on a set of query rows. However, whereas an aggregate operation groups query rows into a single result row, a window function produces a result for each query row:

* The row for which function evaluation occurs is called the current row.
* The query rows related to the current row over which function evaluation occurs comprise the window for the current row.

1. It means window functions perform operations on a set of rows and **produces an aggregated value for each row**. Therefore each row maintains the unique identities.

### **Syntax**

The following are the basic syntax for using a window function:

window\_function\_name(expression)

OVER (

    [partition\_defintion]

    [order\_definition]

[frame definition]

)

Here, the OVER() clause is used to define which rows will be in the window frame.

### **Partition Clause**

### ***partition\_clause***: A PARTITION BY clause indicates how to divide the query rows into groups. The window function result for a given row is based on the rows of the partition that contains the row. If PARTITION BY is omitted, there is a single partition consisting of all query rows.

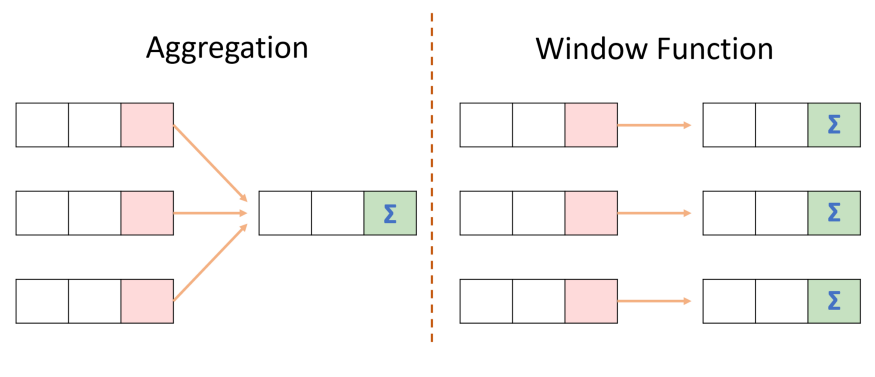
### **ORDER BY Clause**

### An ORDER BY clause indicates how to sort rows in each partition or specify the order of the rows within a partition.

ORDER BY defines the order of the rows in the window frame.

***frame\_clause***: A frame is a subset of the current partition and the frame clause specifies how to define the subset. The frame clause has many subclauses of its own.

### **Window Function Concept**



### let us first create a table named **"Sales"** using the following statement:

CREATE TABLE Sales(

Employee\_Name VARCHAR(45) NOT NULL,

Year INT NOT NULL,

Country VARCHAR(45) NOT NULL,

Product VARCHAR(45) NOT NULL,

Sale DECIMAL(12,2) NOT NULL,

PRIMARY KEY(Employee\_Name, Year) );

INSERT INTO Sales(Employee\_Name, Year, Country, Product, Sale)

VALUES('Joseph', 2017, 'India', 'Laptop', 10000),

('Joseph', 2018, 'India', 'Laptop', 15000),

('Joseph', 2019, 'India', 'TV', 20000),

('Bob', 2017, 'US', 'Computer', 15000),

('Bob', 2018, 'US', 'Computer', 10000),

('Bob', 2019, 'US', 'TV', 20000),

('Peter', 2017, 'Canada', 'Mobile', 20000),

('Peter', 2018, 'Canada', 'Calculator', 1500),

('Peter', 2019, 'Canada', 'Mobile', 25000);

Select \* from sales;

### **Types of Window Function**

We can categorize the window functions mainly in three types that are given below:

### **Aggregate Functions**

It is a function that operates on multiple rows and produces the result in a single row. Some of the important aggregate functions are:

COUNT, SUM, AVG, MIN, MAX, and many more.

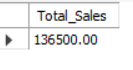
### **Analytical Functions**

It is a function, which is locally represented by a power series. Some of the important analytical functions are:

NTILE, LEAD, LAG, NTH, FIRST\_VALUE, LAST\_VALUE,rank,dense\_rank,row\_number etc.

For example, using the sales information table, these two queries perform aggregate operations that produce a single global sum for all rows taken as a group, and sums grouped per country:

SELECT SUM(sale) AS Total\_Sales FROM Sales;

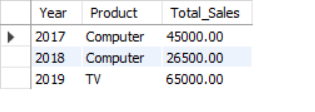


SELECT Year, Product, SUM(Sale) AS Total\_Sales

FROM Sales

GROUP BY Year

ORDER BY Product;



By contrast, window operations do not collapse groups of query rows to a single output row. Instead, they produce a result for each row. Like the preceding queries, the following query uses [SUM()](https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html#function_sum), but this time as a window function:

SELECT

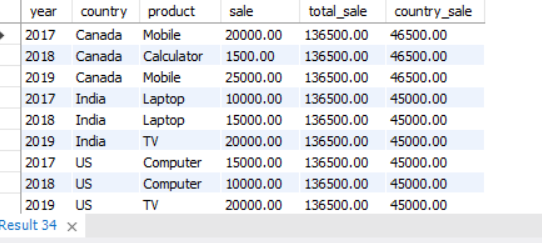
year, country, product, sale,

SUM(sale) OVER() AS total\_sale,

SUM(sale) OVER(PARTITION BY country) AS country\_sale

FROM sales

ORDER BY country, year, product, sale;



Each window operation in the query is signified by inclusion of an OVER clause that specifies how to partition query rows into groups for processing by the window function:

* The first OVER clause is empty, which treats the entire set of query rows as a single partition. The window function thus produces a global sum, but does so for each row.
* The second OVER clause partitions rows by country, producing a sum per partition (per country). The function produces this sum for each partition row.

Window functions are permitted only in the select list and ORDER BY clause. Query result rows are determined from the FROM clause, after WHERE, GROUP BY, and HAVING processing, and windowing execution occurs before ORDER BY, LIMIT, and SELECT DISTINCT.

**Example of Analytical Function**

Here, we are going to use the NTILE window function. This function takes an integer value as an argument that divides the group into a number of integer values. For example, if we use NTILE(4), then it divides the total records into four groups. When the total record is odd, it adds the odd records in the first row. The following query explains it more clearly.

SELECT Year, Product, Sale,

NTile(4) OVER() AS Total\_Sales

FROM Sales;

Let us see another example using the "LEAD" function. This function is used to query more than one row in a table without joining the table itself. It means we can access the data of the next row from the current row. It returns the output from the next row.

SELECT Year, Product, Sale,

LEAD(Sale,1) OVER(ORDER BY Year) AS Total\_Sales

FROM Sales;