Window functions in SQL (Structured Query Language) are a powerful feature used for performing calculations across a set of table rows that are somehow related to the current row. They are often used in data analysis and reporting.

Definition

A window function performs a calculation across a set of table rows that are related to the current row. This set of rows is called the "window frame." Unlike aggregate functions, window functions do not cause rows to become grouped into a single output row; rows retain their separate identities.

Key Components of Window Functions

1. **PARTITION BY**: Divides the result set into partitions to which the window function is applied.
2. **ORDER BY**: Defines the logical order of the rows within each partition.
3. **Frame Definition**: Specifies the subset of rows in the window frame.

Common Window Functions

* **ROW\_NUMBER()**: Assigns a unique sequential integer to rows within a partition of a result set.
* **RANK()**: Assigns a rank to each row within a partition of a result set.
* **DENSE\_RANK()**: Similar to RANK() but without gaps in ranking values.
* **NTILE(n)**: Divides rows in an ordered partition into a specified number of groups.
* **LAG()**: Provides access to a row at a given physical offset before the current row.
* **LEAD()**: Provides access to a row at a given physical offset after the current row.
* **SUM() OVER**: Calculates the sum over a specified window frame.
* **AVG() OVER**: Calculates the average over a specified window frame.

Real-World Example

Let's consider a real-world dataset: a sales database with the following columns:

* sales\_id
* salesperson\_id
* sale\_date
* sale\_amount

Step 1: Create a Database

First, create a database named sales\_db.

CREATE DATABASE sales\_db;

Step 2: Create a Table

Next, switch to the sales\_db database and create a table named sales with the relevant columns.

USE sales\_db;  
  
CREATE TABLE sales (  
    sales\_id INT PRIMARY KEY,  
    salesperson\_id INT,  
    sale\_date DATE,  
    sale\_amount DECIMAL(10, 2)  
);

Step 3: Insert Sample Data

Insert some sample data into the sales table.

INSERT INTO sales (sales\_id, salesperson\_id, sale\_date, sale\_amount) VALUES  
(1, 101, '2023-01-01', 500.00),  
(2, 102, '2023-01-02', 700.00),  
(3, 101, '2023-01-03', 200.00),  
(4, 103, '2023-01-04', 300.00),  
(5, 101, '2023-01-05', 400.00),  
(6, 102, '2023-01-06', 1000.00),  
(7, 103, '2023-01-07', 600.00),  
(8, 101, '2023-01-08', 700.00),  
(9, 102, '2023-01-09', 300.00),  
(10, 103, '2023-01-10', 800.00);

Step 4: Practice Window Functions

Now that you have a table with sample data, you can start practicing window functions.

Example Query 1: Running Total of Sales

SELECT   
    salesperson\_id,  
    sale\_date,  
    sale\_amount,  
    SUM(sale\_amount) OVER (PARTITION BY salesperson\_id ORDER BY sale\_date) AS running\_total  
FROM   
    sales;

Example Query 2: Rank Sales by Salesperson

SELECT   
    salesperson\_id,  
    sale\_date,  
    sale\_amount,  
    RANK() OVER (PARTITION BY salesperson\_id ORDER BY sale\_amount DESC) AS sales\_rank  
FROM   
    sales;

Advantages and Disadvantages Revisited

Advantages

1. **Enhanced Analytical Capabilities**: The running total and rank examples show how you can easily compute cumulative metrics and rankings.
2. **Performance**: These queries can be more efficient than using subqueries and joins.
3. **Readability**: The window functions make the intent of the query clear and straightforward.

Disadvantages

1. **Complexity**: Understanding the window frame and partitioning can be complex initially.
2. **Performance on Large Datasets**: As the size of the dataset grows, performance might degrade.
3. **Support and Portability**: Some older SQL systems may not support window functions, or they might have variations in syntax.

Summary

Window functions are invaluable for performing advanced data analysis directly within SQL queries. They simplify the process of computing running totals, rankings, and other complex calculations while often improving performance and readability. However, they can be complex to use and might have performance implications on very large datasets. Understanding their advantages and limitations is crucial for leveraging their full potential in real-world scenarios.

**Another datasets**

create database sales\_performance\_db;

USE sales\_performance\_db;

CREATE TABLE new\_sales (

transaction\_id INT AUTO\_INCREMENT PRIMARY KEY,

transaction\_date DATE,

product\_id INT,

customer\_id INT,

region VARCHAR(50),

sales\_amount DECIMAL(10, 2)

);

**Datapoints**

INSERT INTO sales (transaction\_date, product\_id, customer\_id, region, sales\_amount) VALUES

('2024-01-01', 101, 1001, 'North', 250.00),

('2024-01-02', 102, 1002, 'South', 300.00),

('2024-01-03', 103, 1003, 'East', 150.00),

('2024-01-04', 104, 1004, 'West', 220.00),

('2024-01-05', 105, 1005, 'North', 180.00),

('2024-01-06', 106, 1006, 'South', 275.00),

('2024-01-07', 107, 1007, 'East', 350.00),

('2024-01-08', 108, 1008, 'West', 400.00),

('2024-01-09', 109, 1009, 'North', 190.00),

('2024-01-10', 110, 1010, 'South', 310.00),

('2024-01-11', 101, 1011, 'East', 260.00),

('2024-01-12', 102, 1012, 'West', 320.00),

('2024-01-13', 103, 1013, 'North', 230.00),

('2024-01-14', 104, 1014, 'South', 290.00),

('2024-01-15', 105, 1015, 'East', 170.00),

('2024-01-16', 106, 1016, 'West', 210.00),

('2024-01-17', 107, 1017, 'North', 200.00),

('2024-01-18', 108, 1018, 'South', 340.00),

('2024-01-19', 109, 1019, 'East', 370.00),

('2024-01-20', 110, 1020, 'West', 410.00),

('2024-01-21', 101, 1021, 'North', 290.00),

('2024-01-22', 102, 1022, 'South', 330.00),

('2024-01-23', 103, 1023, 'East', 280.00),

('2024-01-24', 104, 1024, 'West', 250.00),

('2024-01-25', 105, 1025, 'North', 240.00),

('2024-01-26', 106, 1026, 'South', 260.00),

('2024-01-27', 107, 1027, 'East', 310.00),

('2024-01-28', 108, 1028, 'West', 360.00),

('2024-01-29', 109, 1029, 'North', 270.00),

('2024-01-30', 110, 1030, 'South', 320.00),

INSERT INTO sales (transaction\_date, product\_id, customer\_id, region, sales\_amount) VALUES

('2024-02-01', 101, 1031, 'East', 280.00),

('2024-02-02', 102, 1032, 'West', 290.00),

('2024-02-03', 103, 1033, 'North', 260.00),

('2024-02-04', 104, 1034, 'South', 270.00),

('2024-02-05', 105, 1035, 'East', 300.00),

('2024-02-06', 106, 1036, 'West', 320.00),

('2024-02-07', 107, 1037, 'North', 310.00),

('2024-02-08', 108, 1038, 'South', 330.00),

('2024-02-09', 109, 1039, 'East', 340.00),

('2024-02-10', 110, 1040, 'West', 350.00),

('2024-02-11', 101, 1041, 'North', 360.00),

('2024-02-12', 102, 1042, 'South', 370.00),

('2024-02-13', 103, 1043, 'East', 380.00),

('2024-02-14', 104, 1044, 'West', 390.00),

('2024-02-15', 105, 1045, 'North', 400.00),

('2024-02-16', 106, 1046, 'South', 410.00),

('2024-02-17', 107, 1047, 'East', 420.00),

('2024-02-18', 108, 1048, 'West', 430.00),

('2024-02-19', 109, 1049, 'North', 440.00),

('2024-02-20', 110, 1050, 'South', 450.00),

('2024-02-21', 101, 1051, 'East', 460.00),

('2024-02-22', 102, 1052, 'West', 470.00),

('2024-02-23', 103, 1053, 'North', 480.00),

('2024-02-24', 104, 1054, 'South', 490.00),

('2024-02-25', 105, 1055, 'East', 500.00),

('2024-02-26', 106, 1056, 'West', 510.00),

('2024-02-27', 107, 1057, 'North', 520.00),

('2024-02-28', 108, 1058, 'South', 530.00),

('2024-02-29', 109, 1059, 'East', 540.00),

('2024-03-01', 110, 1060, 'West', 550.00),

('2024-03-02', 101, 1061, 'North', 560.00),

('2024-03-03', 102, 1062, 'South', 570.00),

('2024-03-04', 103, 1063, 'East', 580.00),

('2024-03-05', 104, 1064, 'West', 590.00),

('2024-03-06', 105, 1065, 'North', 600.00),

('2024-03-07', 106, 1066, 'South', 610.00),

('2024-03-08', 107, 1067, 'East', 620.00),

('2024-03-09', 108, 1068, 'West', 630.00),

('2024-03-10', 109, 1069, 'North', 640.00),

('2024-03-11', 110, 1070, 'South', 650.00),

('2024-03-12', 101, 1071, 'East', 660.00),

('2024-03-13', 102, 1072, 'West', 670.00),

('2024-03-14', 103, 1073, 'North', 680.00),

('2024-03-15', 104, 1074, 'South', 690.00),

('2024-03-16', 105, 1075, 'East', 700.00),

('2024-03-17', 106, 1076, 'West', 710.00),

('2024-03-18', 107, 1077, 'North', 720.00),

('2024-03-19', 108, 1078, 'South', 730.00),

('2024-03-20', 109, 1079, 'East', 740.00),

('2024-03-21', 110, 1080, 'West', 750.00);

### 1. ****Calculate the running total of sales amount for each region****

**Question:** What is the running total of sales amount for each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    SUM(sales\_amount) OVER (PARTITION BY region ORDER BY transaction\_date) AS running\_total  
FROM sales;

**Explanation:** This query calculates the cumulative sum of sales\_amount within each region, ordered by transaction\_date.

### 2. ****Find the rank of each sale within its region based on sales amount****

**Question:** What is the rank of each sale within its region based on sales amount?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    RANK() OVER (PARTITION BY region ORDER BY sales\_amount DESC) AS sales\_rank  
FROM sales;

**Explanation:** This query assigns a rank to each sale within its region based on the sales\_amount, with the highest sales amount getting the highest rank.

### 3. ****Identify the previous sale amount for each transaction****

**Question:** What was the previous sale amount for each transaction in the same region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    LAG(sales\_amount, 1) OVER (PARTITION BY region ORDER BY transaction\_date) AS previous\_sales\_amount  
FROM sales;

**Explanation:** This query retrieves the sales amount of the previous transaction within the same region, ordered by transaction\_date.

### 4. ****Find the highest sale amount within each region****

**Question:** What is the highest sale amount within each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    MAX(sales\_amount) OVER (PARTITION BY region) AS max\_sales\_amount  
FROM sales;

**Explanation:** This query finds the maximum sales amount within each region.

### 5. ****Calculate the average sales amount within each region****

**Question:** What is the average sales amount within each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    AVG(sales\_amount) OVER (PARTITION BY region) AS average\_sales\_amount  
FROM sales;

**Explanation:** This query calculates the average sales amount for each region.

### 6. ****Determine the difference between the current and the next sale amount****

**Question:** What is the difference between the current and the next sale amount within each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    LEAD(sales\_amount, 1) OVER (PARTITION BY region ORDER BY transaction\_date) - sales\_amount AS next\_sales\_difference  
FROM sales;

**Explanation:** This query calculates the difference between the current sale amount and the next sale amount within the same region.

### 7. ****Identify the first sale amount for each region****

**Question:** What was the first sale amount in each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    FIRST\_VALUE(sales\_amount) OVER (PARTITION BY region ORDER BY transaction\_date) AS first\_sales\_amount  
FROM sales;

**Explanation:** This query retrieves the first sale amount for each region, ordered by transaction\_date.

### 8. ****Calculate the percentage of total sales amount within each region****

**Question:** What is the percentage of each sale amount relative to the total sales amount within each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    sales\_amount / SUM(sales\_amount) OVER (PARTITION BY region) \* 100 AS sales\_percentage  
FROM sales;

**Explanation:** This query calculates the percentage of each sale amount relative to the total sales amount within each region.

### 9. ****Find the row number for each sale within its region****

**Question:** What is the row number for each sale within its region based on the transaction date?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    ROW\_NUMBER() OVER (PARTITION BY region ORDER BY transaction\_date) AS row\_num  
FROM sales;

**Explanation:** This query assigns a unique row number to each sale within its region based on the transaction\_date.

### 10. ****Group transactions into 4 equal parts based on sales amount within each region****

**Question:** How can we group transactions into 4 equal parts based on sales amount within each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    NTILE(4) OVER (PARTITION BY region ORDER BY sales\_amount) AS sales\_quartile  
FROM sales;

**Explanation:** This query divides the transactions into

four equal groups (quartiles) based on the sales amount within each region.

### 11. ****Calculate the moving average of sales amount over the last 3 transactions for each region****

**Question:** What is the moving average of sales amount over the last 3 transactions for each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    AVG(sales\_amount) OVER (PARTITION BY region ORDER BY transaction\_date ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS moving\_avg  
FROM sales;

**Explanation:** This query calculates the moving average of sales amount over the last 3 transactions within each region.

### 12. ****Identify the cumulative sales amount percentage within each region****

**Question:** What is the cumulative sales amount percentage within each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    SUM(sales\_amount) OVER (PARTITION BY region ORDER BY transaction\_date) / SUM(sales\_amount) OVER (PARTITION BY region) \* 100 AS cumulative\_sales\_percentage  
FROM sales;

**Explanation:** This query calculates the cumulative sales amount percentage within each region, showing the proportion of sales up to each transaction.

### 13. ****Find the difference between the highest and the lowest sales amount within each region****

**Question:** What is the difference between the highest and the lowest sales amount within each region?

**Query:**

SELECT   
    region,  
    MAX(sales\_amount) - MIN(sales\_amount) AS sales\_range  
FROM sales  
GROUP BY region;

**Explanation:** This query calculates the difference between the highest and the lowest sales amount within each region.

### 14. ****Determine the total number of transactions per region****

**Question:** How many transactions have occurred in each region?

**Query:**

SELECT   
    region,  
    COUNT(transaction\_id) AS total\_transactions  
FROM sales  
GROUP BY region;

**Explanation:** This query counts the total number of transactions in each region.

### 15. ****Identify the sale with the maximum sales amount for each region****

**Question:** Which sale had the maximum sales amount in each region?

**Query:**

WITH max\_sales AS (  
    SELECT   
        region,  
        MAX(sales\_amount) AS max\_sales\_amount  
    FROM sales  
    GROUP BY region  
)  
SELECT   
    s.transaction\_id,  
    s.transaction\_date,  
    s.region,  
    s.sales\_amount  
FROM sales s  
JOIN max\_sales m  
ON s.region = m.region AND s.sales\_amount = m.max\_sales\_amount;

**Explanation:** This query identifies the transaction with the maximum sales amount within each region by first finding the maximum sales amount per region and then joining it back to the original table.

### 16. ****Calculate the difference between each sale and the average sale amount for the respective region****

**Question:** What is the difference between each sale amount and the average sale amount for its region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    sales\_amount - AVG(sales\_amount) OVER (PARTITION BY region) AS sales\_difference  
FROM sales;

**Explanation:** This query calculates the difference between each sale amount and the average sale amount for the respective region.

### 17. ****Find the date of the next transaction for each sale****

**Question:** What is the date of the next transaction for each sale within the same region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    LEAD(transaction\_date, 1) OVER (PARTITION BY region ORDER BY transaction\_date) AS next\_transaction\_date  
FROM sales;

**Explanation:** This query retrieves the date of the next transaction within the same region.

### 18. ****Identify the sales amount of the last transaction within each region****

**Question:** What was the sales amount of the last transaction within each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    LAST\_VALUE(sales\_amount) OVER (PARTITION BY region ORDER BY transaction\_date ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS last\_sales\_amount  
FROM sales;

**Explanation:** This query retrieves the sales amount of the last transaction within each region.

### 19. ****Calculate the difference between the highest and the current sales amount for each region****

**Question:** What is the difference between the highest and the current sales amount for each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    MAX(sales\_amount) OVER (PARTITION BY region) - sales\_amount AS diff\_max\_current  
FROM sales;

**Explanation:** This query calculates the difference between the highest sales amount and the current sales amount for each region.

### 20. ****Determine the number of transactions within the last 7 days for each sale****

**Question:** How many transactions occurred within the last 7 days for each sale within the same region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    COUNT(transaction\_id) OVER  
  
(PARTITION BY region ORDER BY transaction\_date RANGE BETWEEN INTERVAL 7 DAY PRECEDING AND CURRENT ROW) AS transactions\_last\_7\_days  
FROM sales;

**Explanation:** This query calculates the number of transactions that occurred within the last 7 days (including the current transaction date) for each sale within the same region.

### 21. ****Identify the sales amount of the second highest transaction within each region****

**Question:** What is the sales amount of the second highest transaction within each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount  
FROM (  
    SELECT   
        transaction\_id,  
        transaction\_date,  
        region,  
        sales\_amount,  
        DENSE\_RANK() OVER (PARTITION BY region ORDER BY sales\_amount DESC) AS sales\_rank  
    FROM sales  
) ranked\_sales  
WHERE sales\_rank = 2;

**Explanation:** This query identifies the transaction with the second highest sales amount within each region by using the DENSE\_RANK() window function.

### 22. ****Calculate the percentage difference between each sale and the maximum sale amount in its region****

**Question:** What is the percentage difference between each sale amount and the maximum sale amount in its region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    (sales\_amount / MAX(sales\_amount) OVER (PARTITION BY region) \* 100) AS percent\_of\_max\_sales  
FROM sales;

**Explanation:** This query calculates the percentage of each sale amount relative to the maximum sale amount within its region.

### 23. ****Identify the median sale amount within each region****

**Question:** What is the median sale amount within each region?

**Query:**

SELECT   
    region,  
    PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY sales\_amount) OVER (PARTITION BY region) AS median\_sales\_amount  
FROM sales;

**Explanation:** This query calculates the median sale amount within each region using the PERCENTILE\_CONT() function.

### 24. ****Find the cumulative sum of sales amount for each product****

**Question:** What is the cumulative sum of sales amount for each product?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    product\_id,  
    sales\_amount,  
    SUM(sales\_amount) OVER (PARTITION BY product\_id ORDER BY transaction\_date) AS cumulative\_sales  
FROM sales;

**Explanation:** This query calculates the cumulative sum of sales amount for each product, ordered by transaction\_date.

### 25. ****Calculate the sales amount difference from the average sales amount for each product****

**Question:** What is the difference between each sale amount and the average sales amount for its product?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    product\_id,  
    sales\_amount,  
    sales\_amount - AVG(sales\_amount) OVER (PARTITION BY product\_id) AS sales\_diff\_from\_avg  
FROM sales;

**Explanation:** This query calculates the difference between each sale amount and the average sale amount for the respective product.

### 26. ****Identify the region with the highest total sales amount****

**Question:** Which region has the highest total sales amount?

**Query:**

SELECT   
    region,  
    SUM(sales\_amount) AS total\_sales  
FROM sales  
GROUP BY region  
ORDER BY total\_sales DESC  
LIMIT 1;

**Explanation:** This query calculates the total sales amount for each region and returns the region with the highest total sales amount.

### 27. ****Calculate the number of unique customers in each region****

**Question:** How many unique customers are there in each region?

**Query:**

SELECT   
    region,  
    COUNT(DISTINCT customer\_id) AS unique\_customers  
FROM sales  
GROUP BY region;

**Explanation:** This query counts the number of unique customers in each region.

### 28. ****Determine the transaction date difference between consecutive transactions for each product****

**Question:** What is the transaction date difference between consecutive transactions for each product?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    product\_id,  
    sales\_amount,  
    DATEDIFF(transaction\_date, LAG(transaction\_date, 1) OVER (PARTITION BY product\_id ORDER BY transaction\_date)) AS days\_between\_transactions  
FROM sales;

**Explanation:** This query calculates the difference in days between consecutive transactions for each product.

### 29. ****Find the cumulative percentage of total sales amount for each region****

**Question:** What is the cumulative percentage of total sales amount for each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount,  
    SUM(sales\_amount) OVER (PARTITION BY region ORDER BY transaction\_date) / SUM(sales\_amount) OVER (PARTITION BY region) \* 100 AS cumulative\_sales\_percentage  
FROM sales;

**Explanation:** This query calculates the cumulative percentage of total sales amount for each region.

### 30. ****Identify the top 3 sales amounts within each region****

**Question:** What are the top 3 sales amounts within each region?

**Query:**

SELECT   
    transaction\_id,  
    transaction\_date,  
    region,  
    sales\_amount  
FROM (  
    SELECT   
        transaction\_id,  
        transaction\_date,  
        region,  
        sales\_amount,  
        ROW\_NUMBER() OVER (PARTITION BY region ORDER BY sales\_amount DESC) AS sales\_rank  
    FROM sales  
) ranked\_sales  
WHERE sales\_rank <= 3;

**Explanation:** This query identifies the top 3 sales amounts within each region by using the ROW\_NUMBER() window function.

These scenarios cover a wide range of window function applications in MySQL, providing useful insights into the sales data for various business analysis purposes.