Window Functions in SQL

Overview

SQL window functions are a powerful tool that allows you to perform calculations across rows of a result set, providing insights that go beyond simple aggregations.

One of the most commonly used window functions is SUM.

In this article, we'll explore how to leverage SUM within window functions to gain valuable insights from your data.

Understanding Window Functions

Before diving into SUM, let's clarify what a window function is.

Essentially, it operates on a set of rows, called a window frame, and returns a value based on the calculation applied to that frame.

This window frame can be defined in various ways, such as:

- Rows: A specific number of rows before and/or after the current row.
- Range: A range of values, like dates or numbers, before and/or after the current row.

Case Using SUM as a Window Function

When used as a window function, SUM calculates a running total or a cumulative sum within a specified window frame.

Import libraries

```
import duckdb as db
import polars as pl
```

Create database

```
conn = db.connect('lm.db')
```

Create table

```
conn.sql('''
    create table sales (
        sales_month varchar,
        product varchar,
        sales float,
        )
''')
```

Populate table

```
conn.sql('''
    insert into sales (sales_month, product, sales)
    VALUES
        ('January', 'Product A', 1000),
        ('February', 'Product B', 1200),
        ('March', 'Product C', 1500),
        ('April', 'Product A', 1800),
        ('May', 'Product B', 2000),
        ('June', 'Product C', 2200),
        ('July', 'Product A', 2500),
        ('August', 'Product B', 2800),
        ('September', 'Product C', 3000),
        ('October', 'Product A', 3200),
        ('November', 'Product B', 3500),
        ('December', 'Product C', 3800),
        ('January', 'Product A', 1100),
        ('February', 'Product B', 1300),
        ('March', 'Product C', 1600),
        ('April', 'Product A', 1900),
        ('May', 'Product B', 2100),
        ('June', 'Product C', 2300),
        ('July', 'Product A', 2600),
        ('August', 'Product B', 2900),
        ('September', 'Product C', 3100),
        ('October', 'Product A', 3300),
        ('November', 'Product B', 3600),
        ('December', 'Product C', 3900),
        ('January', 'Product A', 1200),
        ('February', 'Product B', 1400),
        ('March', 'Product C', 1700),
        ('April', 'Product A', 2000),
        ('May', 'Product B', 2200),
        ('June', 'Product C', 2400),
        ('July', 'Product A', 2700),
        ('August', 'Product B', 3000),
''')
```

Preview data

```
conn.sql('select * from sales limit 5').pl()
```

sales_month	product	sales
str	str	f32
"January"	"Product A"	1000.0
"February"	"Product B"	1200.0
"March"	"Product C"	1500.0
"April"	"Product A"	1800.0
"May"	"Product B"	2000.0

Sum Window Function

```
conn.sql('''
    select *
        , sum(sales) over(
            partition by product
            order by sales_month
            rows between unbounded preceding and current row
            ) cumulative_sales
    from sales
    limit 5
''').pl()
```

sales_month	product	sales	cumulative_sales
str	str	f32	f64
"April"	"Product A"	1800.0	1800.0
"April"	"Product A"	1900.0	3700.0
"April"	"Product A"	2000.0	5700.0
"April"	"Product A"	2100.0	7800.0
"January"	"Product A"	1000.0	8800.0

Window Function Breakdown

- All window functions use the over() clause.
- Partition by() defines how to group the data.
- Order by defines how to sort the data within the group.
- Rows between defines the relative window to aggregate on.
    - Unbounded preceding means the start of the window is at the beginning of the group.
    - current row means the end of the window is the current row.

Conclusions

SQL window functions, particularly SUM, provide a flexible and powerful way to analyze data and uncover valuable insights.

By understanding the core concepts and applying them to practical scenarios, you can unlock the full potential of your data analysis.

References

This article is inspired on Dawn Choo (2024) "Breaking Down Window Functions in SQL" retrieved from Linkedin.

Contact

Jesus L. Monroy

Economist & Data Scientist

| Portfolio | Medium | Linkedin | Twitter |