

Window Functions on SQL - Analyzing Tech Stocks on BigQuery

Antonio Rueda-Toicen
October 2023

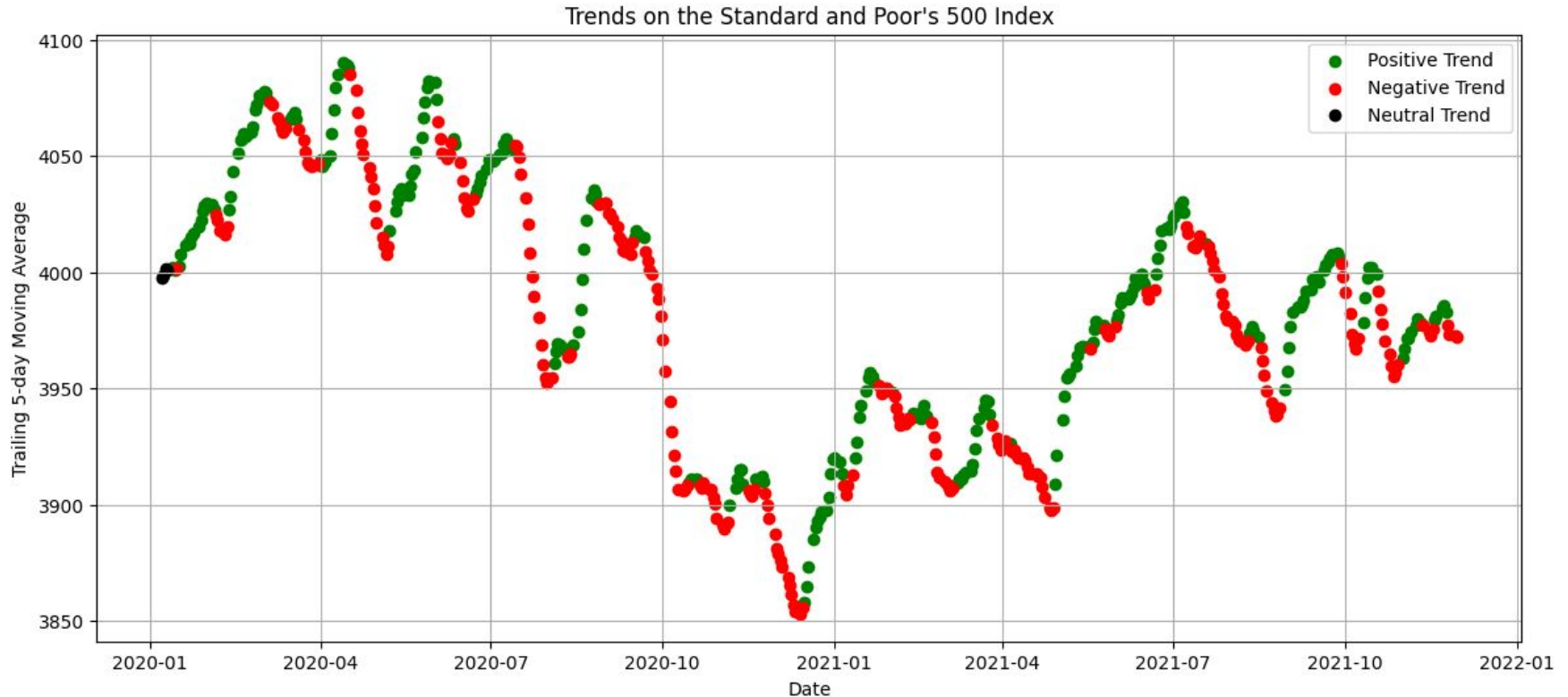
Agenda

- What are window functions?
 - Use cases
 - Window functions on SQL
- Using window functions on Google Bigquery and Google Colab
- Analyzing stock data from tech companies and the Standard and Poor's 500 Index
- Further exercises
 - LeetCode challenges

What we assume that you know today

- What a table and a schema are
- SELECT statements in SQL
- The following operators
 - WHERE, GROUP BY, ORDER, LIMIT, GROUP BY

What we will be building



**We will use moving averages to tell whether tech stocks
are on a “bull” or a “bear” market**





Use cases for window functions

Anywhere where we would like to:

- Aggregate (SUM, AVG, MAX, MIN)
- Rank (define order)
- Access lagging or leading values without a self-join

Example: time series analysis is a prime use case for window functions

What are window functions?

We map an operation (*the function*) to a set of rows (*the window*) using the **OVER()** keyword in a **SELECT** statement

```
1 SELECT
2     function() OVER (
3 FROM TABLE
```



What are window functions?

Window functions allow us to aggregate, rank, and lag/lead the values in a set of rows

```
1 SELECT
2     function() OVER (
3 FROM TABLE
```



What are window functions?



```
1 SELECT
2     function() OVER (
3 FROM TABLE
```

Select a function

- Aggregate
- Ranking
- Lead/Lag/First/Last/Percentile (“Analytic”)

Define a window

- PARTITION BY
- ORDER BY
- ROWS

Defining Windows

We define windows inside the OVER clause:

- PARTITION BY - divides according to a common value
- ORDER BY - defines an order for rows within a group
- ROWS or RANGE - sets start and ending points for a group


Rows

function() OVER (ORDER by Date

ROWS BETWEEN 4 PRECEDING AND CURRENT ROW)

In this case ORDER BY is required

What are window functions?



```
1  
2 SELECT  
3     Symbol,  
4     AVG(Close) OVER (PARTITION BY Symbol) AS avg_close  
5 FROM  
6     tech_stocks AS stocks;
```

Window functions do not reduce rows

Row	Symbol	avg_close
1	NVDA	309.2771317683...
2	NVDA	309.2771317683...
3	NVDA	309.2771317683...
4	NVDA	309.2771317683...
5	NVDA	309.2771317683...
6	NVDA	309.2771317683...
7	NVDA	309.2771317683...
8	NVDA	309.2771317683...
9	NVDA	309.2771317683...

The avg_close column is added to the data, compare this with a GROUP BY



```
1  
2 SELECT  
3     Symbol,  
4     AVG(Close) AS avg_close  
5 FROM  
6     tech_stocks AS stocks;  
7 GROUP BY Symbol
```


GROUP BY performs row reduction

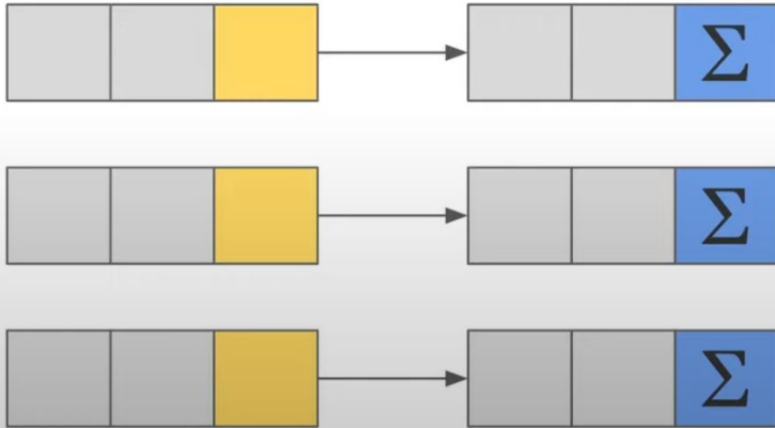
Row	Symbol ▼	average_close_price
1	AAPL	164.3439443215...
2	MSFT	291.3612353564...
3	NVDA	309.2771317683...
4	META	222.2317132227...
5	GOOGL	111.5903588405...
6	NFLX	360.4188846420...

GROUP BY can only do aggregation

Window functions provide us aggregation PLUS ranking and lead/lag

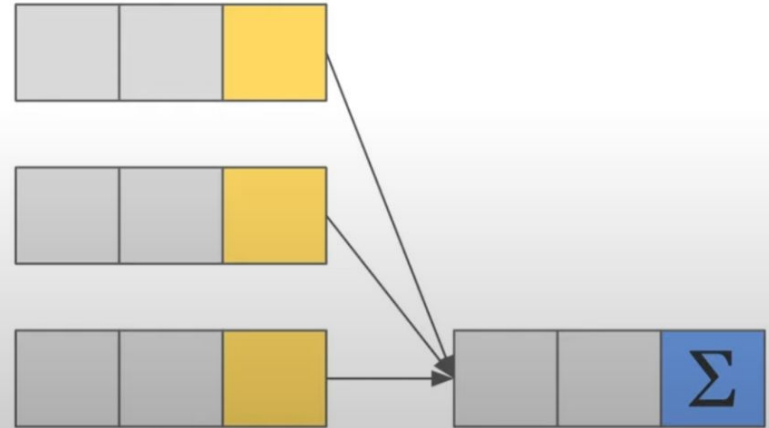
Window Functions:

The rows maintain their separate identities

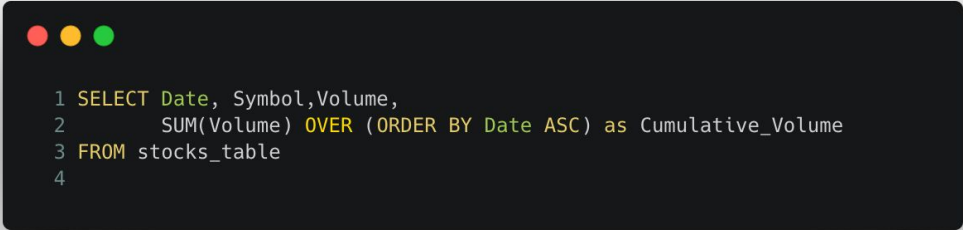


GROUP BY Statement:

Populates only one row per aggregation



Window functions produce an output with the same size as the initial table



```
1 SELECT Date, Symbol, Volume,  
2      SUM(Volume) OVER (ORDER BY Date ASC) as Cumulative_Volume  
3 FROM stocks_table  
4
```

Common questions



Common questions

- Window functions are not only for aggregation?
 - OVER without PARTITION BY() is equal to GROUP BY
 - We will see this in code
- Understanding how to properly nest functions within each other and in what order they are executed can be challenging
 - Knowing which functions correspond to which windows

Practice notebook

- [Window Functions in SQL with Bigquery](#)

References

- [Moving Averages - Investopedia](#)
- [A Simple Time Series Analysis Of The S&P 500 Index](#)
- [SQL Window Functions Demonstrated with Real World Interview Questions from LeetCode](#)