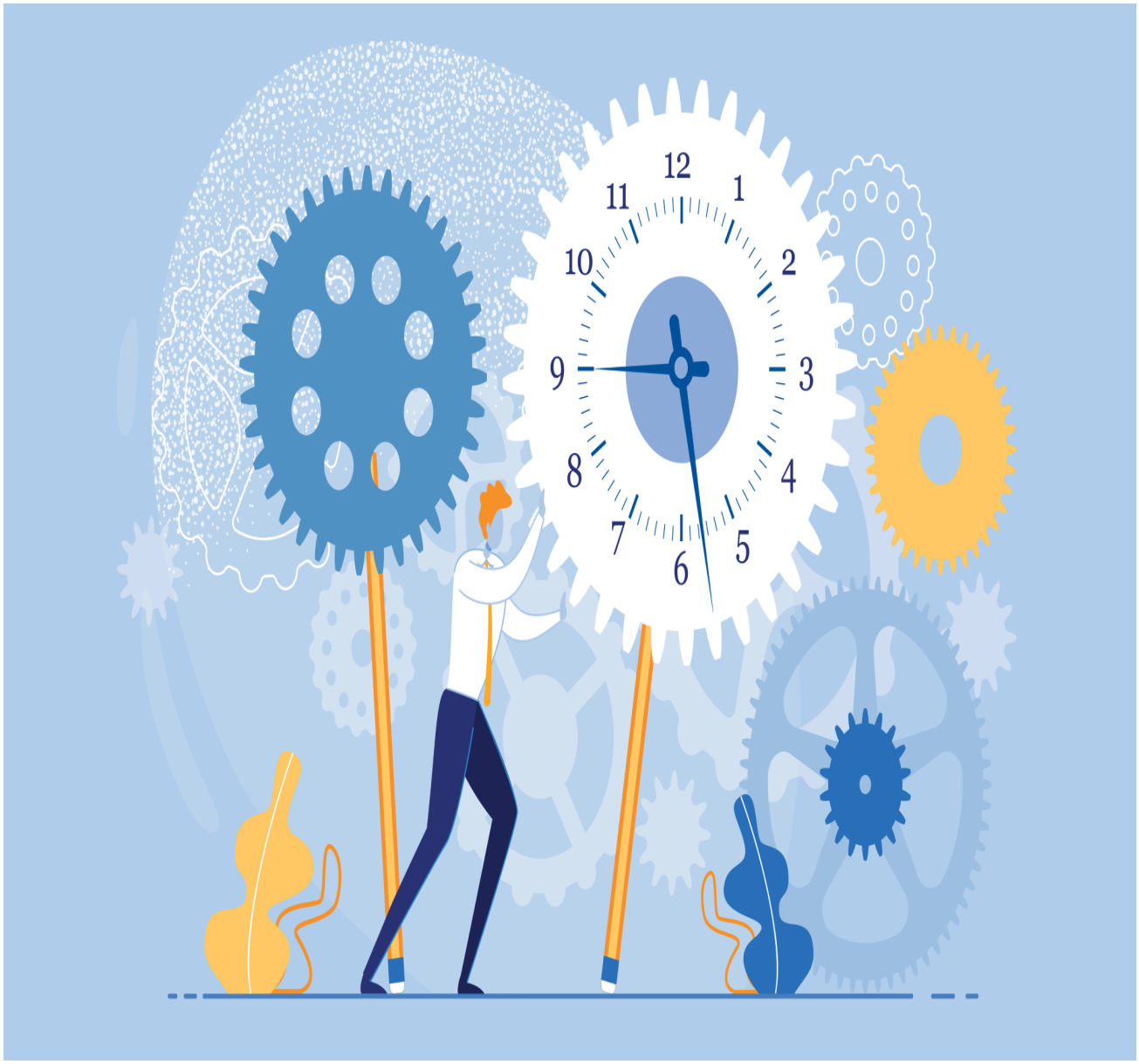
Exploring Time Series Analysis in SQL

Calculating Series Length:

Urmila Rajawat



What Is a Series?

“Time series” is basically a sequence of data listed in time order.
In a database, this is usually represented by events separated by an equal time. For example, user website logins on consecutive days is considered a series. The table below shows such login dates:

id	date	consecutive logins
1	2020-06-01	3 days
2	2020-06-02	
3	2020-06-03	
4	2020-06-06	6 days
5	2020-06-07	
6	2020-06-08	
7	2020-06-09	
8	2020-06-10	
9	2020-06-11	

10	2020-06-13	4 days
11	2020-06-14	
12	2020-06-15	
13	2020-06-16	

Here, this series length will be the count of consecutive logins. The first series length is three days, since the user logged in on 2020-06-01, 2020-06-02, and 2020-06-03. The second series length is six days (the user logged in every day between 2020-06-06 and 2020-06-11). Following the same logic, the next series length is four days.

Why Calculate a Series Length?

Time series are extensively used and there are many situations where you'd need to calculate series length. Some examples of calculating series lengths include:

- Measuring a login streak on Stack Overflow.
- Seeing your activity streak on Duolingo.
- Tracking how many days you've used a fitness app.
- Analyzing a sales streak in an e-commerce site.
- Finding the consecutive increase or decrease of a currency value.

Basically, anywhere we have a time series, we'll probably need to calculate its length.

How to Calculate the Length of a Series in SQL:

For this example, let's imagine you're practicing SQL on InterviewQuery. The InterviewQuery platform has a thing called a streak. As the site explains, a streak is "the number of days in a row you have solved a question. Once you solve a question on the website, your streak will increase by one day. Monitoring your streak allows you to receive specific platform incentives. So how does InterviewQuery know how long your streak is?
Our learning streak for July 2020 can be presented by the table `question_solved`. It contains the following columns:

- `id`: The ID of the question.
- `date_solved`: The date you solve the question.

Now let's write a query to find our streak. We'll use a Common Table Expression (CTE) to help organize this query. Running this query will calculate the length of the series:

```
WITH grouped AS (
    SELECT
        RANK() OVER (ORDER BY date_solved) AS row_num,
        date_completed,
        DATEADD (day, -RANK() OVER (ORDER BY date_solved),
            date_solved) AS date_group
FROM
    question_solved)

SELECT
    COUNT(*) AS days_streak,
    MIN (date_solved) AS min_date,
    MAX (date_solved) AS max_date
FROM
    grouped
GROUP BY date_group;
```

The query can be divided into two parts:

- Creating the CTE.
- Selecting data from the CTE.

Creating the CTE:

The part of the query that creates the CTE is given again below:

```
WITH grouped AS (
    SELECT
        RANK() OVER (ORDER BY date_solved) AS row_num,
        date_solved,
        DATEADD (day, -RANK() OVER (ORDER BY date_solved),
            date_solved) AS date_group
FROM
    question_solved)
```

The CTE is defined by the `WITH` clause. We've decided the name of this CTE is “grouped”. Everything that is written in the parentheses after the AS keyword is just a regular `SELECT` statement.

Now, what does this do? First, we’ve added the number of rows to the table. To do that, we’ve used the `RANK()` function. This is a window function, which is why it’s defined by the `OVER()` clause. We want the ranks to be added sequentially according to the dates, so this function’s result is ordered by the column `date_solved`.

The CTE then selects the column `date_solved`. We've used the `DATEADD` function (SQL Server) to deduct the row number from the `date_solved`.

In this statement …

```
DATEADD (day, -RANK() OVER (ORDER BY date_solved), date_solved) AS
date_group
```

we had to define the interval that will be added (or deducted), which is the day. How many days do we want to deduct? The days that are equal to the number of rows – that’s why we’ve only copied the `RANK()` window function we already had defined and added the negative sign in front of it. From what do we want this to be deducted? From the `date_solved`, of course!

This part of the query will give the following result:

Row_num	date_solved	date_group
1	2020-07-01	2020-06-30
2	2020-07-02	2020-06-30
3	2020-07-03	2020-06-30
4	2020-07-04	2020-06-30
5	2020-07-05	2020-06-30
6	2020-07-08	2020-07-02
7	2020-07-09	2020-07-02
8	2020-07-10	2020-07-02
9	2020-07-18	2020-07-09
10	2020-07-19	2020-07-09
11	2020-07-20	2020-07-09

Row_num	date_solved	date_group
12	2020-07-21	2020-07-09
13	2020-07-22	2020-07-09
14	2020-07-23	2020-07-09
15	2020-07-24	2020-07-09
16	2020-07-25	2020-07-09
17	2020-07-26	2020-07-09
18	2020-07-28	2020-07-10
19	2020-07-29	2020-07-10
20	2020-07-30	2020-07-10
21	2020-07-31	2020-07-10

Why do we need this? It will help us calculate the series length. In the table we notice that the consecutive days belong to the same date group? If the dates are consecutive and we deduct the row number from it, we'll always get the same date. Look at this:

row_number	date_completed	date_group
1	2020-07-01	2020-06-30

Deduct the row number (1) from the date (2020-07-01) and we'll get 2020-06-30. That's precisely what we have on the table.

row_num	date_solved	date_group
2	2020-07-02	2020-06-30

And then the next row. If we deduct the row number (2) from the date (2020-07-02), the result is 2020-06-30 again! Now we will check on when the dates are not consecutive.

row_num	date_solved	date_group
5	2020-07-05	2020-06-30
6	2020-07-08	2020-07-02

For row 5, the result is still the same, 2020-06-30. but for the next row, when we deduct row number 6 from the date (2020-07-08), we get 2020-07-02. This is now a new date group, since 2020-07-08 isn't consecutive to 2020-07-05.

Regarding dates in the `date_group` column, it really doesn't matter what dates we get. They'll only serve as values, which will be counted in the second part of the query. Count how many times every group occurs, and we'll have our series length!

Selecting Data from the CTE:

The second part of the query selects the data from the CTE we've defined above:

```
SELECT
    COUNT(*) AS days_streak,
    MIN(date_solved) AS min_date,
    MAX(date_solved) AS max_date
FROM
    grouped
GROUP BY date_group;
```

This simple `SELECT` statement counts the number of rows and shows the results in the new column `days_streak`. Then it selects the minimum and maximum date in the column `date_solved` with the results shown in the columns `min_date` and `max_date`, respectively. All this data will be selected from the CTE named grouped.

Finally, the data has to be grouped by the `date_group`. Because we don't need the total number of rows; we want the number of rows for each date group

days_streak	min_date	max_date
5	2020-07-01	2020-07-05
3	2020-07-08	2020-07-10

9	2020-07-18	2020-07-26
4	2020-07-28	2020-07-31

That’s One Way to Calculate a Series Length ...

In conclusion, this example is an idea of how to calculate the length of a series with SQL. However, there’s no single way to approach it and there’s no simple SQL function to achieve results. our data and what we need to calculate will determine our SQL code. It usually requires a little trick; What we certainly need to know when calculating the length of the series are **SQL window functions**.This article is a glimpse of what time series are.Time series analysis is a crucial aspect of data analysis in various domains such as finance, marketing, healthcare, and more.