Amirk Follow

Sep 1, 2020 · 4 min read · · Listen

Save The Date

How to generate missing dates in SQL queries

In this article we will address:

1. How to generate a table with a range of dates using Presto SQL
2. Rules of thumb for joining tables that are supposed to complete missing data from one another.

Data integrity is one of the most important things we need to address before performing calculations over our data. Even with all the right intentions, we can sometimes miss that there is an error in our data. This can become very difficult when the mistake is not in the data we have but **in the data we don't have**.

When performing a calculation that takes into account the number of samples in our data (calculating an average or median value) we need to address rows where the value is NULL or zero.

amount of purchases for each customer, we will get an overestimate.

```
   customer_id     | order_date | purchase_amount |
10000100005411274 | 2020-04-11 |        1        |
10000100005411274 | 2020-04-16 |        1        |
10000100005411274 | 2020-04-18 |        2        |
10000100005411274 | 2020-04-21 |        2        |
10000100005411274 | 2020-04-24 |        1        |
```

If we calculate the customer's daily average purchase amount without looking at our raw data, we would think his average purchase amount is 1.4 (what a customer!).

To overcome this issue, we must generate and match between all dates to all customers. Via Presto SQL we can do this in a simple query:

```
SELECT
     CAST(date_column AS DATE) date_column
 FROM
     (VALUES
         (SEQUENCE(date('2020-04-01'),
                   date('2020-04-30'),
                   INTERVAL '1' DAY)
         )
     ) AS t1(date_array)
 CROSS JOIN
     UNNEST(date_array) AS t2(date_column)
```

Using SEQUENCE, we will create an array with the dates in our range and preform a cross join between each element in the array to the array itself. The result is a column with a row for each of the different dates.

A quick alternative could be to extract all the different dates regardless of customer from our initial data and store it as a WITH AS statement as well.

Next, we will preform another cros [illegible] tween our customers and the different dates in order to fill in the missing ones:

```
with all_dates as (
SELECT
     CAST(date_column AS DATE) date_column
 FROM
     (VALUES
         (SEQUENCE(date('2020-04-01'),
                   date('2020-04-30'),
                   INTERVAL '1' DAY)
```

```
        )
    ) AS t1(date_array)
 CROSS JOIN
    UNNEST(date_array) AS t2(date_column)
)
select distinct customer_id
               ,date_column as order_date
from customer_purchases
cross join all_dates
```

Last, we will join the table with the new matches between customer and dates to the initial table with our data.

important notice

The **strong** table should be the new table we created with customer and dates while the *left* joined table should be our initial data. If we perform an inner join we will lose all of the dates that had no purchases. Also, we need to make sure not to put conditions in the where clause that address columns in the left joined table, this will turn the left join to an inner join.

```
with all_dates as (
SELECT
     CAST(date_column AS DATE) date_column
 FROM
     (VALUES
         (SEQUENCE(date('2020-04-01'),
                   date('2020-04-30'),
                   INTERVAL '1' DAY)
         )
     ) AS t1(date_array)
 CROSS JOIN
     UNNEST(date_array) AS t2(date_column)
)

,customers_dates as (
select distinct customer_id
               ,date_column as order_date
from customer_purchases
cross join all_dates

)

select u.customer_id
     , u.order_date
     , coalesce(p.purchase_amount,0) purchase_amount
from customers_dates u
left join customer_purchases p
on u.customer_id = p.customer_id
```

```
and u.order_date = p.order_date
order by customer_id asc

----------------------------------------------------------------------

   customer_id    | order_date | purchase_amount|
................. | .......... | .............  |
10000100005411274 | 2020-04-13 |       0        |
10000100005411274 | 2020-04-14 |       0        |
10000100005411274 | 2020-04-15 |       0        |
10000100005411274 | 2020-04-16 |       1        |
10000100005411274 | 2020-04-17 |       0        |
10000100005411274 | 2020-04-18 |       2        |
```

Now we can calculate the average daily purchase amount, the right way.

Summary (TL; DR):

1. Using SEQUENCE, we can create an array with a range of dates and turn them into a table. This method is effective for filling missing dates in our data to make sure that dates in which *nothing* happened will still appear.

2. When performing a left join between two tables having conditions in the where clause that address columns from the *left* table, will turn the left join to an **inner join**. It's easy to miss this and you should always think twice when applying conditions in the same query with a join between tables.

Presto Sql Data Science Data Engineering Data