Filter by Date

by Sophia

WHAT'S COVERED

In this lesson, you will explore filtering data based on dates and formatting date elements, in three parts. Specifically, this lesson will cover:

1. Dates
2. Dates and Times
3. Date Formatting

1. Dates

Working with dates in databases can be challenging because different databases have different formats for storing dates. Different countries and regions customarily use differing date formats, and that can lead to confusion. For example, the USA and the UK both use a MM/DD/YYYY format, but France uses DD/MM/YYYY, and Japan uses YYYY/MM/DD. In addition, when a year is expressed as a two-digit number rather than four-digit, there may be confusion over which century it refers to.

In PostgreSQL, the date is formatted as YYYY-MM-DD. For example, a September 15th, 2015, date would be formatted as 2015-09-15 in PostgreSQL. When you search on a date or insert data into the column, it is important to use the same format that the database expects in the column.

Dates must be enclosed in single quotes, like text strings. For example, if we wanted to find all of the invoices that were submitted on January 1st, 2009, we would use the following query:

```
SELECT *
FROM invoice
WHERE invoice_date = '2009-01-01';
```

When you search for the invoice just on the date alone, you will also get time data along with the date. For now, we will just work with dates and not worry about the time data. Later on, you will learn how to use time as well to make your SQL queries very specific to what you are looking for.

Query Results

Row count: 1

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
1	2	2009-01-01T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	2

You can use ranges for date parameters as well. For example, if we want to find all invoices before January 31st, 2009, we can use the < (less than) operator in the comparison instead of the = operator. Remember to use the single quotes on the dates; otherwise, it might be seen as a number and not a string of data.

```
SELECT *
FROM invoice
WHERE invoice_date < '2009-01-31';
```

This will return the result set:

Query Results

Row count: 6

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
1	2	2009-01-01T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	2
2	4	2009-01-02T00:00:00.000Z	Ullevålsveien 14	Oslo		Norway	0171	4
3	8	2009-01-03T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	6
4	14	2009-01-06T00:00:00.000Z	8210 111 ST NW	Edmonton	AB	Canada	T6G 2C7	9
5	23	2009-01-11T00:00:00.000Z	69 Salem Street	Boston	MA	USA	2113	14
6	37	2009-01-19T00:00:00.000Z	Berger Straße 10	Frankfurt		Germany	60316	1

This method gives us the flexibility to control the date ranges.

---

2. Dates and Times

Retrieving and manipulating time values is important in database management, especially one where program execution decisions depend on the date and time. For example, in an e-commerce site, you might need to keep track of the amount of time a shopping cart's stored items should remain active.

You can use the ‘datetime’ variable to store both the date and the time. Remember in PostgreSQL the date is stored as YYYY-MM-DD, time is stored as HH:MI:SS.MMMM for hour, minute, second, and millisecond. The precision for millisecond is very important, especially for items that use cryptography or digital signing.

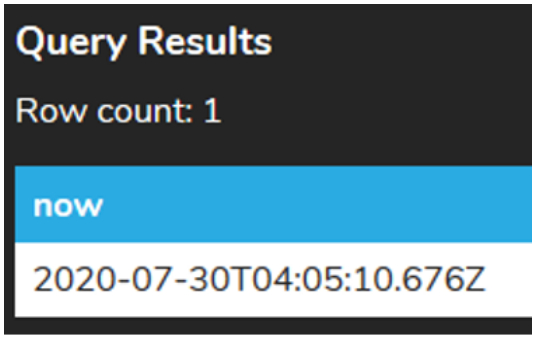
One way to customize a clause is to use functions. You may be familiar with functions from using spreadsheet applications; the concept is similar. A function is a named operation that performs some type of calculation or formatting.

Some date-related functions can be useful for getting the current date and time.

For example, suppose you want to get the server's current date and time. To do this, you would use the now() function, like this:

```
SELECT now();
```

When run, you should see a result similar to the following:

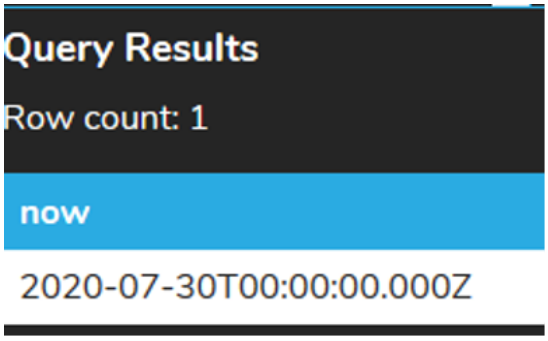


Notice that the results come in the format YYYY-MM-DD followed by T for time and HH:MI:SS.MMM and ending with a Z. The “Z” specifies GMT as the time zone. The date/time functions in SQL are based on the date and time on the server on which you run them. On the PostgreSQL server we use in this course, by default, the time is expressed in GMT (Greenwich Mean Time).

If you only want the date, not the time, you can add ::date to the end of the now() function like this:

```
SELECT now()::date;
```

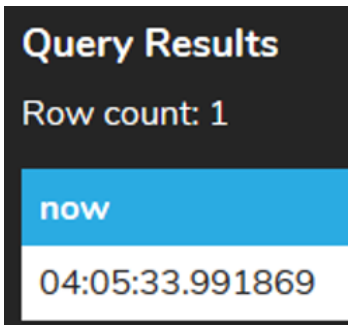
This would return the following:



Note that when you get only the date, the time is still there; it's just all zeros, as shown above.

If you only want the time, not the date, you can add ::time to the now() function, like this:

```
SELECT now()::time;
```



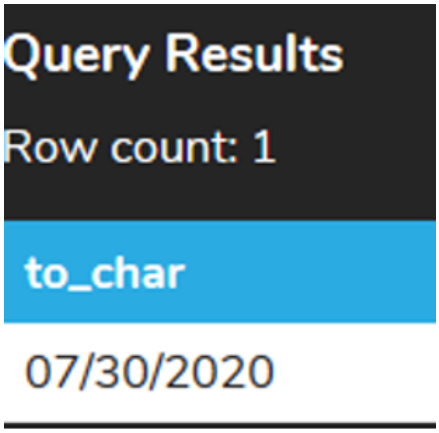
Note that when you exclude the date and just get the time, as shown above, the time matches the system time, not the GMT time.

3. Date Formatting

Sometimes you need to convert data formats to a standard that is used in the database. The default in PostgreSQL Database is YYYY-MM-DD, but if we need to store dates in a USA format, we need DD-MM-YYYY, so we need to change the date format to fit our table column expectation.

To do this, we can change the format of the date by using the TO_CHAR function. TO_CHAR can convert the date to the format you want. It takes two parameters. The first parameter is the date value that we want to format. The second is the template that defines the format. For example, if we wanted to output the current date in an MM/DD/YYYY format, we can do so by writing:

```
SELECT TO_CHAR(now()::date, 'mm/dd/yyyy');
```



Note in the above example that the now() function is nested inside the TO_CHAR() function.

If we wanted to do this as part of a program, we could use:

```
SELECT payment_date, TO_CHAR(payment_date, 'HH12:MI:SS' ) payment_time FROM payment ORDER BY payment_date;
```

This takes in the system time and converts it to the HH (Hour of the day) to 12 for the payment time.

There are many different template patterns for date formatting. Some of the most common include:

- hh – Hour of the day (01-12)
- hh24 – Hour of the day (00-23)
- mi – Minute
- ss – Second
- ms – Millisecond
- yyyy – A year in 4 digits
- yy – Last two digits of the year
- Month – The whole month name with the capital first letter
- month – The whole month name in lowercase
- Mon – The abbreviated month name
- MM – Month number (01-12)
- Day – Full capitalized day name
- dd – Day of the month
- tz – Uppercase time zone name

Considering what you see above, think about what this command would return and then run it in PostgreSQL to see if you were right.

```
SELECT TO_CHAR(now(), 'Day, Month dd, yyyy hh24:mi ss tz');
```

WATCH

TRY IT

Your turn! Open the SQL tool by clicking on the LAUNCH DATABASE button below. Then, enter in one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.

SUMMARY

In this lesson, you learned that different database systems store **dates** in different formats, and that PostgreSQL stores them in a YYYY-MM-dd format by default. Then you learned how to include date and time values in WHERE clauses to filter data by them. You can filter by specific dates or date ranges.

Next, you learned about inserting or retrieving both **dates and times**. The now() function is used to get the current date and time. PostgreSQL expresses the current time as Greenwich Mean Time (GMT). You saw how to query for only the date, only the time, or both together.

Finally, you learned about **date formatting** and how to use the TO_CHAR function to convert between formats. You learned about the many template patterns for date formatting and how to use one of these patterns to specify the date format you want your query results to use.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND Faithe Wempen (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR **TERMS OF USE**.