Filter by Date

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores filtering data based on dates and formatting date elements in three parts:

1. Getting Started
2. Date and Time
3. Date Formatting

1. Getting Started

Working with dates in databases can be challenging, because different databases have different formats for storing dates. 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. This is the same format that is used when we insert dates into a date column.

Note that we must quote the dates with single quotes in the same way that we do with a string. 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';
```

Notice that with the invoice_date in the invoice table, the data type is of type timestamp. We'll get into that in the next part but be aware that this does exist.

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 wanted to find all invoices prior to January 31st, 2009, we can use the < operator in the comparison instead of the = operator.

```
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 flexibility to control the date ranges.

---

2. Date and Time

There is also a datetime variable that stores both the date and time. The time is set as hh:mi:ss.mmm, where mmm are in milliseconds.

There are some date-related functions that can be useful for getting the current date and time. Note that these functions are based on the server date and time, so when you are running them, they may not be in your timezone. By default, on the PostgreSQL server that we use, the date and time are in GMT.

For example, you can use the following function in this query to get the current date and time:

SELECT now();
When run, you should see a result similar to the following:

Query Results

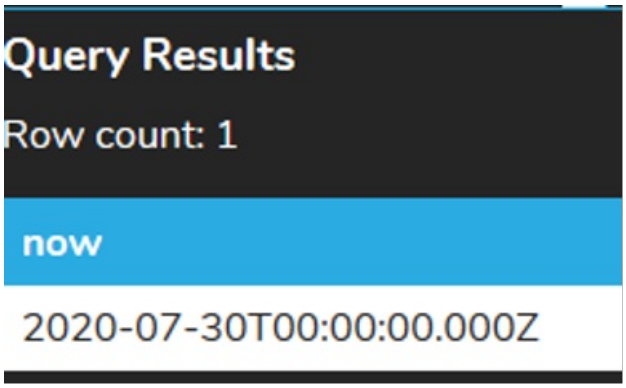
Row count: 1

now
2020-07-30T04:05:10.676Z

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.

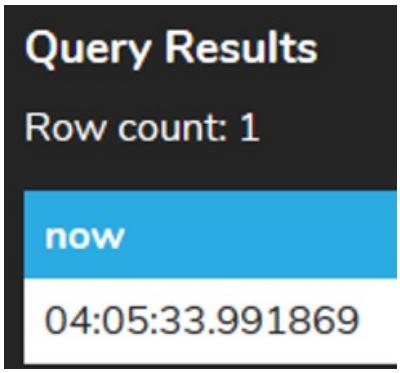
If you only wanted to get the date and not the time, you can do a conversion using:

SELECT now()::date;
This would return the following:



We can also just get the time by using:

SELECT now()::time;

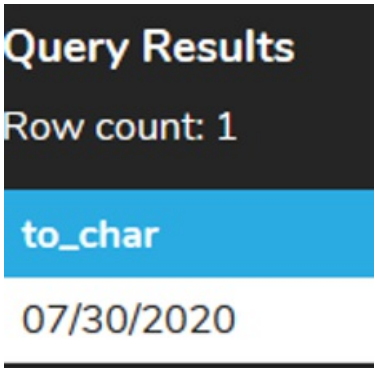


---

3. Date Formatting

We can change the format of the date using the TO_CHAR function. The TO_CHAR to convert the date takes in two parameters. The first parameter is the 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 a mm/dd/yyyy format, we can do so by writing:

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



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 – year in 4 digits
- yy – last 2 digits of the year
- Month – the full month name with the capital first letter
- month – the full 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 – upper case time zone name

Considering what you see above, think about what this command would return and run it in PostgreSQL:

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

Video Transcription

[MUSIC PLAYING] We can also be able to filter the data based on dates. So if we take a look at this invoice table here, we'll have 412 records. If we're looking for a specific date, you'll notice that under the invoice date it has a certain date format. This is set up as a timestamp in this case here. It has year, year, year, year, and then a dash, and then month, month, dash, day, day for the characters, and then comes the time. So T for time, and then it has hour, hour, minute, minute, second, second, and then four characters for the milliseconds.

However, if we're only looking at dates, we have the ability to be able to utilize the WHERE clause invoice_date equal to the date. So the date again is yyyy for year, year, year, year, then dash, then month in two character format, and then the day in two character format. Let's see if we have any invoices that were sent out on this date. We'll see it comes back with one invoice. We'll see that the invoice date is shown here.

There are other options to utilize as well if we're looking for certain ranges. So if we're looking for all the invoices that were prior to this date, we can utilize the listen operator. Run that. We'll see that it returns with eight rows.

[MUSIC PLAYING]

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

Working with date and time can be database-specific, but the core format is consistent between databases. There exist many different methods to extract specific date information from a timestamp data type.

Source: Authored by Vincent Tran