School of Code

PostgreSQL Dates Cheatsheet

The Complete Python/PostgreSQL Developer Course

Dates have been a problem in databases for a long time. There are a few reasons for this. First, that dates can be formatted in many ways, some of which are ambiguous. Second, that dates can have a timezone attached to them, and depending on where your users live, you may have to change the timezone on the fly in your application. Third, daylight saving—when the clocks move forward or backward one hour—is a huge hassle because we essentially lose or gain one hour which is unaccounted for.

> *I recommend storing dates in the database without timezone information (this is called naive dates).*

Date data types

PostgreSQL has a few data types used to store dates and times:

```
timestamp
date
time
interval
```

The timestamp and time types can optionally have timezone information attached to them.

Date formatting

Formatting dates in PostgreSQL is simple if you use default formats. It quickly becomes more complicated as you deviate from the recommended format for dates.

The recommended format, and the default, is ISO 8601. This is shown below:

```
1999-01-08 17:35:56
```

The long date there is the **8th of January 1999 at 5pm, 35 minutes, and 56 seconds**.

Use this format unless you have very important reasons not to. You can change the format in your Python applications much more easily than the overhead of changing it in PostgreSQL.

If you always adhere to the ISO standard, you'll always know whether the format you are using to enter data into the database is correct. If you deviate from the standard and another person tries to put data in, they need to know that you have deviated from the standard.

It's easier for everyone to use the ISO 8601 standard, even if it is not the norm in your country of residence.

However, when retrieving data from SQL, you can optionally format it a different way. For example, if you wanted to show the full day of week name:

```
to_char(current_timestamp, 'FMDay, FMDD  HH:MI:SS')
```

Which would show 'Saturday, 21 12:38:55'

There's many ways to format timestamps. Here's a few of the format patterns you can use:

```
Day - the name of the week day
Month - the name of the month
YYYY - the year as 4 digits
YY - the year as 2 digits
MM - the month as 2 digits
DD - the day of the month as 2 digits
HH - the hour as 2 digits
MI - the minutes as 2 digits
SS - the seconds as 2 digits
```

Again, there's many more. Most you will never use, but the documentation once again has everything you can use.

Date functions and operations

Operations

In PostgreSQL, there's a few functions and operations we can perform on date/time fields.

```
1) SELECT DATE ‘2001-09-28’ + INTEGER ‘7’

2) SELECT DATE ‘2001-09-28’ + INTERVAL ‘1 hour’

3) SELECT TIMESTAMP ‘2001-09-28 23:00:00’ - INTERVAL ‘23 hours’

4) SELECT TIMESTAMP ‘2001-09-28 23:00:00’ - TIMESTAMP ‘2001-09-27
12:00’
```

1) Would add 7 days to the date, ending up with `DATE ‘2001-10-05’`
2) Would add 1 hour to the date, ending up with `TIMESTAMP ‘2001-09-08 01:00:00’`
3) Would subtract 23 hours from the timestamp, ending up with `TIMESTAMP ‘2001-09-28 00:00:00’`
4) Would return the time interval between the two timestamps, ending up with `INTERVAL ‘1 day 11:00:00’`

There’s a bunch of other operations, but just know that in general, the operations you can do make sense. It’s worth trying some out!

Functions

If you use a date format that PostgreSQL understands (like ISO 8601), there are many functions you can use on the dates and times.

For example, you can extract the day of the month:

```
SELECT EXTRACT(DAY FROM TIMESTAMP '2001-02-16 20:38:40');
Result: 16
```

The hour in the time:

```
SELECT EXTRACT(HOUR FROM TIMESTAMP '2001-02-16 20:38:40');
Result: 20
```

The month, as a number:

```
SELECT EXTRACT(MONTH FROM TIMESTAMP '2001-02-16 20:38:40');
Result: 2

SELECT EXTRACT(MONTH FROM INTERVAL '2 years 3 months');
Result: 3

SELECT EXTRACT(MONTH FROM INTERVAL '2 years 13 months');
Result: 1
```

In general, this makes sense too. Here's a list of what you can extract from a complete timestamp:

```
CENTURY
DAY
DECADE
DOW (day of week 0-Monday to 6-Sunday)
DOY (day of year)
EPOCH (number of seconds passed since 1970-01-01 00:00:00)
HOUR
ISODOW (day of week 1-Monday to 7-Sunday)
ISOYEAR (don't use this)
MICROSECONDS
MILLENNIUM
MILLISECONDS
MINUTE
MONTH
QUARTER
SECOND
TIMEZONE
TIMEZONE_HOUR
TIMEZONE_MINUTE
WEEK
YEAR
```