Time and date commands

This lesson covers some of the commonly used date and time commands in SQL.

While creating a SQL code, you may want to assign dates to records.

- You may also want to find out how much time has passed between two dates, or how many days there are between today's date and the date of a particular record.
- Or you may want to extract the years from a bunch of dates, disregarding the months and days, to create a year column.

The date and time commands in SQL help you perform these actions. SQL has a very long list of date and time commands, but in this lesson, you'll focus on a few of the most used ones.

CURRENT_DATE

The command CURRENT_DATE returns the current date.

If you include this in a SQL code, it automatically finds the current date when the code is run. This is useful when you want a code to always reference the current date, without you having to manually update the code each time that it's run. For example, if you want to track how many days have passed between a transaction date and the current date, this command makes it so that you don't have to manually calculate the difference yourself.

This is the structure of the CURRENT_DATE command:

```
1  CURRENT_DATE;
```

As always, if you want SQL to return this value in the output, you can use the SELECT command in front of CURRENT_DATE. That looks like this:

```
1  SELECT CURRENT_DATE;
```

You can also apply operators to this query to add or subtract days from the current date. For example, say you have a data table of product orders placed today. Adding five days to the current date might be a useful way to estimate processing or shipping time.

You can add five days to the current date by running the following statement:

```
1  SELECT CURRENT_DATE + 5;
```

CURRENT_TIME

The command CURRENT_TIME returns the current time. If you include this in a SQL code, it automatically finds the current time when the code is run. This is useful if you want to check the current time, or if you need your code to track when you executed a run or made a change to a table.

The structure of the CURRENT_TIME command is as follows:

```
CURRENT_TIME;
```

If you want SQL to return this value in the output, use the SELECT command in front of the CURRENT_TIME command, like this:

```
SELECT CURRENT_TIME;
```

In a blank Query Tool in pgAdmin, execute the above SELECT query. You'll see the current time in the output. Note that this current time is based on the default setting of your SQL server and may not match up with your local time zone. If you need to set your server's time zone to your local time zone instead, use the following statement:

```
SET timezone ='timezone';
```

There are multiple timezone string options for many of the major cities in the United States. Some of the most commonly used string options include the following:

- America/New_York for the Eastern Time Zone
- America/Chicago for the Central Time Zone
- America/Denver for the Mountain Time Zone
- America/Los_Angeles for the Pacific Time Zone

For example, if you're in the Central Time Zone, you can set your SQL server's time zone with this command:

```
SET timezone ='America/Chicago';
```

Then, if you execute the SELECT CURRENT_TIME query again, it will reflect your local time zone.

CURRENT_TIMESTAMP

The command CURRENT_TIMESTAMP returns the current time and date. If you include this in a SQL query, it automatically finds the current time and date when the code is run. This is useful when you want your code to track when you executed a run or made a change to a table, without you having to manually update the code each time that it's run.

This is the structure of the CURRENT_TIMESTAMP command:

```
CURRENT_TIMESTAMP;
```

If you want SQL to return this value in the output, use the SELECT command in front of the CURRENT_TIMESTAMP command, like this:

```
SELECT CURRENT_TIMESTAMP;
```

In a blank Query Tool in pgAdmin, execute this SELECT query. You will see the current time and date in the output.

EXTRACT YEAR

Often, datasets have columns with full dates, such as processed dates or transaction dates. But you may only be interested in the year; you may want to see year-over-year changes or identify which years the data includes. If this is the case, you can use the EXTRACT command with YEAR.

To use EXTRACT YEAR, you can use either of these date formats:

```
EXTRACT (YEAR from date'YYYY-MM-DD');
```

```
EXTRACT (YEAR from date'MM/DD/YYYY');
```

Try this in pgAdmin by creating and executing a query that extracts the year from the date June 12, 2020.

Your query and output should look something like this:

Query Editor Query History

```
1  SELECT EXTRACT (YEAR from date '6/12/2020');
```

Data Output Explain Messages Notifications

	date_part double precision
1	2020

To extract the year from an entire column of dates, use this query structure:

```
1  SELECT *, EXTRACT (YEAR from column_name)
2  FROM table_name;
```

Notice, the only changes made to the query are the inclusion of SELECT, the inclusion of the * symbol to retrieve all columns, and the inclusion of column_name instead of a date.

In pgAdmin, open a new Query Tool for the nycflights13 database. Use the following query to extract years from the time_hour field.

```
1  SELECT *, EXTRACT (YEAR from time_hour)
2  FROM flights;
```

Your output contains all columns, as well as a new column showing the extracted year.

Data Output Explain Messages Notifications

	tailnum text	origin text	dest text	air_time integer	distance integer	hour integer	minute integer	time_hour timestamp without time zone	date_part double precision
1545	N14228	EWR	IAH	227	1400	5	15	2013-01-01 05:00:00	2013
1714	N24211	LGA	IAH	227	1416	5	29	2013-01-01 05:00:00	2013
1141	N619AA	JFK	MIA	160	1089	5	40	2013-01-01 05:00:00	2013
725	N804JB	JFK	BQN	183	1576	5	45	2013-01-01 05:00:00	2013
461	N668DN	LGA	ATL	116	762	6	0	2013-01-01 06:00:00	2013
1696	N39463	EWR	ORD	150	719	5	58	2013-01-01 05:00:00	2013
507	N516JB	EWR	FLL	158	1065	6	0	2013-01-01 06:00:00	2013
5708	N829AS	LGA	IAD	53	229	6	0	2013-01-01 06:00:00	2013

EXTRACT MONTH and **EXTRACT DAY** work the same.

AGE

If you're working with data that contains dates, such as shipping or delivery dates, you may want to see how much time has passed since a product has shipped. Or you may want to know how long it took for products to go from shipping status to delivery status.

To find the difference between two dates, use the AGE command, as shown below:

```
1  AGE ('date2','date1');
```

When using AGE, date1 is the earlier date and date2 is the more recent date. Using SELECT with this statement returns the difference between the two dates, as measured in number of days.

Because date1 comes after date2 in this second query, the difference is shown as a negative. The number of days in between the dates is the same.