DATE

The following page shows the most commonly used PostgreSQL date functions that allow you to manipulate date and time values effectively.

Timestamp is a digital record of the time of occurrence of a particular event.

Timestamptz is accepted as an abbreviation of timestamp with timezone

FUNCTION	RETURN TYPE	DESCRIPTION
AGE	INTERVAL	Calculate ages between two timestamps or current date (at midnight) and a timestamp and returns a “symbolic” result which uses years and months.
CLOCK_TIMESTAMP	TIMESTAMPTZ	Return the current date and time which changes during statement execution
CURRENT_DATE	DATE	Return the current date
CURRENT_TIME	TIMESTAMPTZ	Return the current time
CURRENT_TIMESTAMP	TIMESTAMPTZ	Return the current date and time with time zone at which the current transaction starts
DATE_PART	DOUBLE PRECISION	Get a field of a timestamp or an interval e.g., year, month, day, etc.
DATE_TRUNC	TIMESTAMP	Return a timestamp truncated to a specified precision
EXTRACT	DOUBLE PRECISION	Same as DATE_PART() function
ISFINITE	BOOLEAN	Check if a date, a timestamp, or an interval is finite or not (not +/-infinity)
JUSTIFY_DAYS	INTERVAL	Adjust interval so 30-day time periods are represented as months
JUSTIFY_HOURS	INTERVAL	Adjust interval so 24-hour time periods

		are represented as days
JUSTIFY_INTERVAL	INTERVAL	Adjust interval using justify_days and justify_hours, with additional sign adjustments
LOCALTIME	TIME	Return the time at which the current transaction start
LOCALTIMESTAMP	TIMESTAMP	Return the date and time at which the current transaction start
NOW	TIMESTAMPTZ	Return the date and time with time zone at which the current transaction start
STATEMENT_TIMESTAMP	TIMESTAMPTZ	Return the current date and time at which the current statement executes
TIMEOFDAY	TEXT	Return the current date and time, like clock_timestamp, as a text string)
TRANSACTION_TIMESTAMP	TIMESTAMPTZ	Same as NOW() function
TO_DATE	DATE	Convert a string to a date
TO_TIMESTAMP	TIMESTAMPTZ	Convert a string to a timestamp

AGE() FUNCTION

We typically have to calculate ages in business applications e.g., ages of people, years of services of employees, etc. In PostgreSQL, you can use the AGE() function to achieve these tasks.

Syntax

```
AGE(timestamp,timestamp);
```

The AGE() function accepts two TIMESTAMP values. It subtracts the second argument from the first one and returns an interval as a result.

```
SELECT AGE('2017-01-01','2011-06-24');
```

```
         AGE
----------------------
 5 years 6 mons 7 days
(1 row)
```

If you want to take the current date as the first argument, you can use the following form of the AGE() function:

```
SELECT current_date,
       AGE(timestamp '2000-01-01');
```

```
    date    |           AGE
------------+--------------------------
 2017-03-20 | 17 years 2 mons 19 days
(1 row)
```

CURRENT_DATE() FUNCTION

The PostgreSQL CURRENT_DATE function returns the current date.

Syntax

```
CURRENT_DATE
```

Example:

```
SELECT CURRENT_DATE;
```

Output

```
2017-08-15
```

CURRENT_TIME() FUNCTION

The PostgreSQL CURRENT_TIME function returns the current time with the time zone.

Syntax:

```
CURRENT_TIME(precision)
```

The CURRENT_TIME function accepts one optional argument which is the precision

Example without precision:

```
SELECT CURRENT_TIME;
```

Output:

```
       timetz
--------------------
 19:25:24.805985-07
(1 row)
```

Example with precision set to two:

```
SELECT CURRENT_TIME(2);
```

Output:

```
     timetz
----------------
 19:26:43.01-07
(1 row)
```

Using it in a create table

```
CREATE TABLE log (
    log_id SERIAL PRIMARY KEY,
    message VARCHAR(255) NOT NULL,
    created_at TIME DEFAULT CURRENT_TIME,
    created_on DATE DEFAULT CURRENT_DATE
);
```

DATE_PART() FUNCTION

The DATE_PART() function extracts a subfield from a date or time value. The following illustrates the DATE_PART() function:

```
DATE_PART(field,source)
```

The field is an identifier that determines what field to extract from the source. The values of the field must be in a list of permitted values mentioned below:

- century
- decade
- year

- month
- day
- hour
- minute
- second
- microseconds
- milliseconds
- dow
- doy
- Epoch
- isodow
- isoyear
- timezone
- timezone_hour
- timezone_minute

The source is a temporal expression that evaluates to TIMESTAMP, TIME, or INTERVAL. If the source evaluates to DATE, the function will be cast to TIMESTAMP.

Examples:

To extract century from timestamp

```
SELECT date_part('century',TIMESTAMP '2017-01-01');

 date_part
-----------
        21
(1 row)
```

To extract year from timestamp

```
SELECT date_part('year',TIMESTAMP '2017-01-01');
 date_part
-----------
      2017
(1 row)
```

To extract quarter from timestamp

```
SELECT date_part('quarter',TIMESTAMP '2017-01-01');
 date_part
-----------
         1
(1 row)
```

To extract month from timestamp

```
 SELECT date_part('month',TIMESTAMP '2017-09-30');
 date_part
-----------
         9
(1 row)
```

To extract dexade from timestamp

```
 SELECT date_part('decade',TIMESTAMP '2017-09-30');
 date_part
-----------
       201
(1 row)
```

To extract week number from timestamp

```
SELECT date_part('week',TIMESTAMP '2017-09-30');
 date_part
-----------
        39
(1 row)
```

To get the current millennium, you use the DATE_PART() function with the NOW() function as follows:

```
 SELECT date_part('millennium',now());
 date_part
-----------
         3
(1 row)
```

To extract day from timestamp

```
SELECT date_part('day',TIMESTAMP '2017-03-18 10:20:30');
 date_part
-----------
        18
(1 row)
```

To extract the hour, minute, second from a timestamp.

```
SELECT date_part('hour',TIMESTAMP '2017-03-18 10:20:30') h,
       date_part('minute',TIMESTAMP '2017-03-18 10:20:30') m,
       date_part('second',TIMESTAMP '2017-03-18 10:20:30') s;

 h  | m  | s
----+----+----
 10 | 20 | 30
(1 row)
```

To extract the day of week and or day of year from a timestamp, you use the following statement:

```
SELECT date_part('dow',TIMESTAMP '2017-03-18 10:20:30') dow,
       date_part('doy',TIMESTAMP '2017-03-18 10:20:30') doy;

 dow | doy
-----+-----
   6 |  77
(1 row)
```

NOW() FUNCTION

The NOW() function returns the current date and time based on the database server's time zone setting.. The return type of the NOW() function is the timestamp with time zone. See the following example:

```
SELECT NOW();
              now
-------------------------------
 2017-03-18 08:21:36.175627+07
(1 row)
```

If we change the timezone to 'America/Los_Angeles':

```
SET TIMEZONE='America/Los_angeles';
```

And get the current date and time:

```
SELECT NOW();
              now
-------------------------------
 2017-03-17 18:29:21.758315-07
(1 row)
```

As you can see, the value returned by the NOW() function is adjusted to the new timezone.

If you want get the current date and time without timezone, you can cast it explicitly as follows:

```
SELECT NOW()::timestamp;
            now
----------------------------
 2017-03-17 18:37:29.229991
(1 row)
```

You can use the common date and time operator to the NOW() function. For example, to get 1 hour from now:

```
SELECT (NOW() + interval '1 hour') AS an_hour_later;
         an_hour_later
-------------------------------
 2017-03-17 19:42:37.110567-07
(1 row)
```

To get this time tomorrow, you add 1 day to the current time:

```
SELECT (NOW() + interval '1 day') AS this_time_tomorrow;
      this_time_tomorrow
-------------------------------
 2017-03-17 19:43:35.178882-07
(1 row)
```

To get 2 hours 30 minutes ago, you use the minus (-) operator as follows:

```
SELECT now() - interval '2 hours 30 minutes' AS two_hour_30_min_go;

      two_hour_30_min_go
-------------------------------
 2017-03-17 16:17:07.742688-07
(1 row)
```

Besides the NOW() function, you can use the CURRENT_TIME or CURRENT_TIMESTAMP to get the current date and time with timezone:

```
SELECT CURRENT_TIME, CURRENT_TIMESTAMP;
       timetz       |              now
--------------------+-------------------------------
 18:50:51.191353-07 | 2017-03-17 18:50:51.191353-07
(1 row)
```

To get the current date and time without a time zone, you use the LOCALTIME and LOCALTIMESTAMP functions.

```
SELECT LOCALTIME, LOCALTIMESTAMP;
      time       |         timestamp
-----------------+----------------------------
 19:13:41.423371 | 2017-03-17 19:13:41.423371
(1 row)
```

If you want to get the current date and time that does advance during the transaction, you can use the TIMEOFDAY() function. Consider the following example:

```
SELECT
    TIMEOFDAY(),
    pg_sleep(5),
    TIMEOFDAY();
              timeofday              | pg_sleep |              timeofday
-------------------------------------+----------+-------------------------------
 Fri Mar 17 19:36:09.216064 2017 PDT |          | Fri Mar 17 19:36:14.217636 20
(1 row)
```

TO_DATE() FUNCTION

The TO_DATE() function converts a string literal to a date value. The following illustrates the syntax of the TO_DATE() function:

Syntax:

```
TO_DATE(text,format);
```

Example:

```
SELECT TO_DATE('20170103','YYYYMMDD');
```

The output shows:

```
  TO_DATE
------------
 2017-01-03
```

```
SELECT TO_DATE('2017 Feb 20','YYYY Mon DD');
```

Output:

```
  TO_DATE
------------
 2017-02-20
(1 row)
```