

POSTGRESQL - DATE/TIME FUNCTIONS AND OPERATORS

http://www.tutorialspoint.com/postgresql/postgresql_date_time.htm

Copyright © tutorialspoint.com

We had discussed about Date/Time data types in the chapter [Data Types](#). Now, let us see the Date/Time operators and Functions.

The following table lists the behaviors of the basic arithmetic operators:

Operator	Example	Result
+	date '2001-09-28' + integer '7'	date '2001-10-05'
+	date '2001-09-28' + interval '1 hour'	timestamp '2001-09-28 01:00:00'
+	date '2001-09-28' + time '03:00'	timestamp '2001-09-28 03:00:00'
+	interval '1 day' + interval '1 hour'	interval '1 day 01:00:00'
+	timestamp '2001-09-28 01:00' + interval '23 hours'	timestamp '2001-09-29 00:00:00'
+	time '01:00' + interval '3 hours'	time '04:00:00'
-	- interval '23 hours'	interval '-23:00:00'
-	date '2001-10-01' - date '2001-09-28'	integer '3' (days)
-	date '2001-10-01' - integer '7'	date '2001-09-24'
-	date '2001-09-28' - interval '1 hour'	timestamp '2001-09-27 23:00:00'
-	time '05:00' - time '03:00'	interval '02:00:00'
-	time '05:00' - interval '2 hours'	time '03:00:00'
-	timestamp '2001-09-28 23:00' - interval '23 hours'	timestamp '2001-09-28 00:00:00'
-	interval '1 day' - interval '1 hour'	interval '1 day -01:00:00'
-	timestamp '2001-09-29 03:00' - timestamp '2001-09-27 12:00'	interval '1 day 15:00:00'
*	900 * interval '1 second'	interval '00:15:00'
*	21 * interval '1 day'	interval '21 days'
*	double precision '3.5' * interval '1 hour'	interval '03:30:00'
/	interval '1 hour' / double precision '1.5'	interval '00:40:00'

Following is the list of all important Date and Time related functions available.

Function	Description
AGE()	Subtract arguments
CURRENT DATE/TIME()	Current date and time

<u>DATE_PART()</u>	Get subfield (equivalent to extract)
<u>EXTRACT()</u>	Get subfield
<u>ISFINITE()</u>	Test for finite date,time and interval (not +/-infinity)
<u>JUSTIFY</u>	Adjust interval

AGE(timestamp, timestamp), AGE(timestamp)

Function	Description
AGE(timestamp, timestamp)	When invoked with the TIMESTAMP form of the second argument, AGE() subtract arguments, producing a "symbolic" result that uses years and months and is of type INTERVAL.
AGE(timestamp)	When invoked with only the TIMESTAMP as argument, AGE() subtracts from the current_date (at midnight).

Example for function AGE(timestamp, timestamp) is:

```
testdb=# SELECT AGE(timestamp '2001-04-10', timestamp '1957-06-13');
```

Above PostgreSQL statement will produce the following result:

age

43 years 9 mons 27 days

Example for function AGE(timestamp) is:

```
testdb=# select age(timestamp '1957-06-13');
```

Above PostgreSQL statement will produce the following result:

age

55 years 10 mons 22 days

CURRENT DATE/TIME()

PostgreSQL provides a number of functions that return values related to the current date and time. Following are some functions:

Function	Description
CURRENT_DATE	Delivers current date.
CURRENT_TIME	Delivers values with time zone.
CURRENT_TIMESTAMP	Delivers values with time zone.
CURRENT_TIME(precision)	Optionally takes a precision parameter, which causes the result to be rounded to that many fractional digits in the seconds field.

CURRENT_TIMESTAMP(precision)	Optionally takes a precision parameter, which causes the result to be rounded to that many fractional digits in the seconds field.
LOCALTIME	Delivers values without time zone.
LOCALTIMESTAMP	Delivers values without time zone.
LOCALTIME(precision)	Optionally takes a precision parameter, which causes the result to be rounded to that many fractional digits in the seconds field.
LOCALTIMESTAMP(precision)	Optionally takes a precision parameter, which causes the result to be rounded to that many fractional digits in the seconds field.

Examples using the functions in the table above:

```
testdb=# SELECT CURRENT_TIME;
         timetz
-----
 08:01:34.656+05:30
(1 row)

testdb=# SELECT CURRENT_DATE;
         date
-----
 2013-05-05
(1 row)

testdb=# SELECT CURRENT_TIMESTAMP;
         now
-----
 2013-05-05 08:01:45.375+05:30
(1 row)

testdb=# SELECT CURRENT_TIMESTAMP(2);
         timestampz
-----
 2013-05-05 08:01:50.89+05:30
(1 row)

testdb=# SELECT LOCALTIMESTAMP;
         timestamp
-----
 2013-05-05 08:01:55.75
(1 row)
```

PostgreSQL also provides functions that return the start time of the current statement, as well as the actual current time at the instant the function is called. These functions are:

Function	Description
transaction_timestamp()	It is equivalent to CURRENT_TIMESTAMP, but is named to clearly reflect what it returns.
statement_timestamp()	It returns the start time of the current statement.

clock_timestamp()	It returns the actual current time, and therefore its value changes even within a single SQL command.
timeofday()	It returns the actual current time, but as a formatted text string rather than a timestamp with time zone value.
now()	It is a traditional PostgreSQL equivalent to transaction_timestamp().

**DATE_PART(text, timestamp),
DATE_PART(text, interval),
DATE_TRUNC(text, timestamp)**

Function	Description
DATE_PART('field', source)	<p>These functions get the subfields. The <i>field</i> parameter needs to be a string value, not a name.</p> <p>.</p> <p>The valid field names are: <i>century, day, decade, dow, doy, epoch, hour, isodow, isoyear, microseconds, millennium, milliseconds, minute, month, quarter, second, timezone, timezone_hour, timezone_minute, week, year</i>.</p>
DATE_TRUNC('field', source)	<p>This function is conceptually similar to the <i>trunc</i> function for numbers. <i>source</i> is a value expression of type timestamp or interval. <i>field</i> selects to which precision to truncate the input value. The return value is of type <i>timestamp</i> or <i>interval</i>.</p> <p>The valid values for <i>field</i> are : <i>microseconds, milliseconds, second, minute, hour, day, week, month, quarter, year, decade, century, millennium</i></p>

Following are examples for DATE_PART(*field*, source) functions:

```
testdb=# SELECT date_part('day', TIMESTAMP '2001-02-16 20:38:40');
date_part
-----
         16
(1 row)

testdb=# SELECT date_part('hour', INTERVAL '4 hours 3 minutes');
date_part
-----
         4
(1 row)
```

Following are examples for DATE_TRUNC(*field*, source) functions:

```
testdb=# SELECT date_trunc('hour', TIMESTAMP '2001-02-16 20:38:40');
date_trunc
-----
2001-02-16 20:00:00
(1 row)

testdb=# SELECT date_trunc('year', TIMESTAMP '2001-02-16 20:38:40');
date_trunc
-----
2001-01-01 00:00:00
(1 row)
```

EXTRACT(field from timestamp), EXTRACT(field from interval)

The **EXTRACT(field FROM source)** function retrieves subfields such as year or hour from date/time values. *source* must be a value expression of type *timestamp*, *time*, or *interval*. *field* is an identifier or string that selects what field to extract from the source value. The EXTRACT function returns values of type *double precision*.

The following are valid field names (similar to DATE_PART function field names): *century*, *day*, *decade*, *dow*, *doy*, *epoch*, *hour*, *isodow*, *isoyear*, *microseconds*, *millennium*, *milliseconds*, *minute*, *month*, *quarter*, *second*, *timezone*, *timezone_hour*, *timezone_minute*, *week*, *year*.

Following are examples for EXTRACT('field', source) functions:

```
testdb=# SELECT EXTRACT(CENTURY FROM TIMESTAMP '2000-12-16 12:21:13');
date_part
-----
          20
(1 row)

testdb=# SELECT EXTRACT(DAY FROM TIMESTAMP '2001-02-16 20:38:40');
date_part
-----
          16
(1 row)
```

ISFINITE(date), ISFINITE(timestamp), ISFINITE(interval)

Function	Description
ISFINITE(date)	Tests for finite date.
ISFINITE(timestamp)	Tests for finite time stamp.
ISFINITE(interval)	Tests for finite interval.

Following are the examples for the ISFINITE() functions:

```
testdb=# SELECT isfinite(date '2001-02-16');
isfinite
-----
t
(1 row)

testdb=# SELECT isfinite(timestamp '2001-02-16 21:28:30');
isfinite
-----
t
(1 row)

testdb=# SELECT isfinite(interval '4 hours');
isfinite
-----
t
(1 row)
```

JUSTIFY_DAYS(interval),

**JUSTIFY_HOURS(interval),
JUSTIFY_INTERVAL(interval)**

Function	Description
JUSTIFY_DAYS(interval)	Adjusts interval so 30-day time periods are represented as months. Return the interval type
JUSTIFY_HOURS(interval)	Adjusts interval so 24-hour time periods are represented as days. Return the interval type
JUSTIFY_INTERVAL(interval)	Adjusts interval using JUSTIFY_DAYS and JUSTIFY_HOURS, with additional sign adjustments. Return the interval type

Following are the examples for the ISFINITE() functions:

```
testdb=# SELECT justify_days(interval '35 days');
justify_days
-----
1 mon 5 days
(1 row)

testdb=# SELECT justify_hours(interval '27 hours');
justify_hours
-----
1 day 03:00:00
(1 row)

testdb=# SELECT justify_interval(interval '1 mon -1 hour');
justify_interval
-----
29 days 23:00:00
(1 row)
```