Oracle Date Functions: The Complete Guide

5 Comments

There are many different date and time functions in Oracle, and they behave differently than other databases. Learn what they are and see some useful queries in this article.

Oracle date functions are any functions that work with date and/or time values. They often take dates as an input, and return many different types of output.

Table of Contents

In this guide, we'll cover the following topics. Click on each of them to be taken to that section of this page.

I'll explain what each of the Oracle date functions are and show an example.

Let's start by taking a look at the different date-related data types in Oracle.

```
SELECT SYSDATE,
ADD_MONTHS(SYSDATE, 5) AS new_date
FROM dual;

SELECT SYSDATE,
ADD_MONTHS(SYSDATE, -12) AS new_date
FROM dual;

SELECT SYSDATE,
LAST_DAY(SYSDATE) AS last_of_month
FROM dual;

SELECT LAST_DAY('03-OCT-2016') AS last_of_m
FROM dual;

SELECT NEXT_DAY(SYSDATE, 'THURSDAY') AS nex
FROM dual;

SELECT NEXT_DAY(SYSDATE, 'MONDAY') AS nextd
FROM dual;
```

Date and Time Data Types in Oracle

Back to Top

There are a few different data types in Oracle that store date values. They are:

- DATE: The "standard" date value in Oracle. It stores year, month, day, as well as hour, minute and second. Yes, even though it's called "date", it stores the time component. This is a good thing to remember.
- TIMESTAMP: This data type stores year, month, day, hour, minute, second, as well as fractional seconds.
- TIMESTAMP WITH TIME ZONE: This data type is the same as TIMESTAMP, but it stores the timezone along with it.
- TIMESTAMP WITH LOCAL TIME ZONE: This data type is similar to TIMESTAMP WITH TIME ZONE, but the timezone that's stored is the database timezone.

- INTERVAL YEAR TO MONTH: This data type stores a “period of time” (rather than a “point in time” like other data types), and represents a period of time in months up to years.
- INTERVAL DAY TO SECOND: This data type also stores a period of time, but stores a value that captures seconds all the way up to days.

So now we’ve looked at the different data types, let’s take a look at the different Oracle date functions.

SYSDATE Function

Back to Top

The Oracle SYSDATE function allows you to easily output the current date. It shows the date and time of the database server.

To use it, you simply type the word SYSDATE. It has no parameters, which means you don’t need any brackets after it.

An example of the SYSDATE function is:

```
SELECT SYSDATE
FROM dual;
```

SYSDATE
10/SEP/17

I’ve used the DUAL table because Oracle needs to select from a table in a select query, and the DUAL table is useful for this (read more about the dual table here).

As you can see, the current date is shown. It only shows the date, though. This is because the output format for SELECT queries for dates is currently set to the DD/MM/YYYY format.

The SYSDATE function does return the time component as well though. We can see this if we change the session’s date format, or surround the SYSDATE in a TO_CHAR function.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS';
```

Now, let’s rerun the SELECT statement with the SYSDATE function.

```
SELECT SYSDATE
FROM dual;
```

SYSDATE
2017-09-10 08:54:19

As you can see, the output now shows the date and time returned from SYSDATE. We changed the date format in the session to include the time.

We could use the TO_CHAR function to change the way this particular function is shown:

```
SELECT TO_CHAR(SYSDATE, 'DD/MM/YYYY HH:MI:SS') AS sysdate_time
FROM dual;
```

SYSDATE
10/09/2017 08:54:19

You can see that the output now includes the date and time.

CURRENT_DATE

Back to Top

The CURRENT_DATE function is similar to the SYSDATE function, but it shows you the **current date and time in the session timezone**.

This is the timezone that your session is in, or the timezone you're in when you log in to the system. This can sometimes be different to the database timezone.

To use this function, simply add the word CURRENT_DATE to your query. No brackets are needed as there are no parameters.

```
SELECT CURRENT_DATE
FROM dual;
```

CURRENT_DATE
10/SEP/17

This shows the current date of your user session.

Just like with the SYSDATE function, this function **returns a DATE data type**, which actually includes a date and a time. To show the time component of this value, either use the TO_CHAR function or alter your session to include the time format.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS';
```

The other way is to use the TO_CHAR function to change the format.

```
SELECT TO_CHAR(CURRENT_DATE, 'DD/MM/YYYY HH:MI:SS') AS currentdate_time
FROM dual;
```

CURRENTDATE_TIME
10/09/2017 08:54:19

You can see that the output now includes the date and time.

Oracle CURRENT_DATE Vs SYSDATE

Back to Top

The main difference between CURRENT_DATE and SYSDATE is:

- CURRENT_DATE returns the date from your **session** timezone (your timezone).
- SYSDATE returns the date from the **database** timezone.

If you're in the east coast of the United States and your database is on the west coast, these functions will show different values. If you're in Australia and the database is in London, they will show different values.

They might even *look* the same but actually be different.

This is because the default output is DD-MON-YY, which hides the time.

This is the default output, assuming I'm in Melbourne (UTC +10) and the database is in Perth (UTC +8).

```
SELECT CURRENT_DATE, SYSDATE
FROM dual;
```

Result:

CURRENT_DATE	SYSDATE
14/JUN/16	14/JUN/16

These values look the same, right?

Now, if I include the time component by using TO_CHAR, my result shows these two values as being different.

```
SELECT TO_CHAR(CURRENT_DATE, 'DD-MON-YY HH:MI:SS') AS CDATE,
TO_CHAR(SYSDATE, 'DD-MON-YY HH:MI:SS') AS SDATE
FROM dual;
```

Result:

CDATE	SDATE
14-JUN-16 05:53:33	14-JUN-16 03:53:33

How Can I Get the Oracle CURRENT_DATE – 1 Day?

If you want to find records that are less than, greater than, or equal to the day before today (or any comparison, really), then you can do this with the Oracle CURRENT_DATE function.

You'll just have to use TRUNC as well.

```
SELECT CURRENT_DATE,
TRUNC(CURRENT_DATE) AS trunc_today,
TRUNC(CURRENT_DATE)-1 AS trunc_yesterday
FROM dual;
```

Result:

CURRENT_DATE	TRUNC_TODAY	TRUNC_YESTERDAY
14/JUN/16	14/JUN/16	13/JUN/16

You can use this TRUNC(CURRENT_DATE)-1 logic in your other SQL queries:

```
--Find employees hired yesterday or later.
SELECT name
FROM employee
WHERE hire_date >= TRUNC(CURRENT_DATE)-1
```

SYSTIMESTAMP

Back to Top

The SYSTIMESTAMP function returns the **date and time of the database**.

The important part to remember is that it returns the time of the database, not your local time.

So, if you're accessing a database in a different time zone, it will return the time and timezone of the place where the database is stored, not your local time zone.

For example, if your database is in London but you are in New York, the SYSTIMESTAMP will return a time that is 5 hours ahead of your time.

It's similar to the SYSDATE function, however it **returns a TIMESTAMP WITH TIME ZONE** instead of a DATE data type.

This means it includes the date, time, and fractional seconds.

```
SELECT SYSTIMESTAMP
FROM dual;
```

SYSTIMESTAMP
10/SEP/17 08:57:17.067000000 AM +10:00

Now, because it's a TIMESTAMP WITH TIME ZONE, it shows the time by default, unlike the DATE data type.

You can see the output here shows the date, the time, the fractions of seconds, and the timezone. You don't need to format it to see the complete output.

How Can I Insert With SYSTIMESTAMP?

You can insert the value returned by SYSTIMESTAMP in a few ways.

First, you can ensure that the column you're inserting into is of type TIMESTAMP WITH TIME ZONE.

Alternatively, if it isn't, you can use the TO_CHAR function to convert the value into a VARCHAR2 data type, and insert that.

However, it's generally not a good idea to insert dates into character fields. I would recommend inserting the value as a TIMESTAMP WITH TIME ZONE.

Can I Find the SYSTIMESTAMP Without Timezone?

Yes, you can.

You can use SYSDATE to just get the date and time, but it won't have fractional seconds.

Can I Use TO_CHAR With SYSTIMESTAMP?

Yes, you can. See the Examples below for more information.

You can include the SYSTIMESTAMP function inside TO_CHAR, and specify the format of the values to output.

Example 1

This is the SYSDATE and SYSTIMESTAMP function side-by-side.

```
SELECT SYSDATE,
SYSTIMESTAMP
FROM dual;
```

Result:

SYSDATE	SYSTIMESTAMP
28/03/2016	28/MAR/16 09:35:48.801000000 AM +11:00

Example 2

This example uses the TO_CHAR function on SYSTIMESTAMP.

```
SELECT SYSTIMESTAMP,
TO_CHAR(SYSTIMESTAMP, 'dd Mon yyyy') AS text_output
FROM dual;
```

Result:

SYSTIMESTAMP	TEXT_OUTPUT
28/MAR/16 09:36:18.196000000 AM +11:00	28 Mar 2016

CURRENT_TIMESTAMP

Back to Top

The CURRENT_TIMESTAMP function returns the **date and time in the timezone of your session**, or where you have logged in from.

It's similar to the CURRENT_DATE function in that it returns the date and time of your session, instead of the database. It's also similar to the SYSTIMESTAMP function in that it returns the timestamp instead of a date.

The **return data type is a TIMESTAMP WITH TIME ZONE**, which includes date, time, fractional seconds, and a timezone.

The CURRENT_TIMESTAMP has one optional parameter: the precision or number of fractional seconds to return. If omitted, it uses 6 places.

```
CURRENT_TIMESTAMP([precision])
```

An example of this function is:

```
SELECT CURRENT_TIMESTAMP
FROM dual;
```

CURRENT_TIMESTAMP
10/SEP/17 08:57:49.020000000 AM AUSTRALIA/MELBOURNE

You can see that the current date and time is shown here, along with the timezone.

Can I Subtract Hours from the CURRENT_TIMESTAMP?

Yes, you can. The best way to do this is to use an interval data type, as you'll keep your original data type.

For example, if I wanted to subtract 4 hours from the current time, I would do this:

```
SELECT CURRENT_TIMESTAMP,
CURRENT_TIMESTAMP - INTERVAL '4' HOUR
FROM dual;
```

Result:

CURRENT_TIMESTAMP	CURRENT_TIMESTAMP-INTERVAL
26/MAY/16 06:17:25.690000000 AM AUSTRALIA/SYDNEY	26/MAY/16 02:17:25.690000000 AM AUSTRALIA/SYDNEY

LOCALTIMESTAMP

Back to Top

The final function that gets the current date is the LOCALTIMESTAMP function.

This function gets the **current date and time of your session**.

It's similar to the CURRENT_TIMESTAMP function, but **LOCALTIMESTAMP returns a TIMESTAMP value** and CURRENT_TIMESTAMP returns a TIMESTAMP WITH TIME ZONE VALUE.

This means that LOCALTIMESTAMP returns just the date and time, and no timezone.

The LOCALTIMESTAMP function looks like this:

```
SELECT LOCALTIMESTAMP
FROM dual;
```

LOCAL_TIMESTAMP
10/SEP/17 08:58:25.614000000 AM

You can see that the output shows date, time, and fractional seconds.

The parameters of the LOCALTIMESTAMP function are:

- timestamp_precision (mandatory): This specifies the number of digits of seconds of the return value, also known as the “fractional second precision”.

ADD_MONTHS

Back to Top

The ADD_MONTHS function allows you to input a date value, and a number of months, and return another date value. The value returned is the input date value plus the number of months you supply.

So, if you start with Jan 10th 2017, and add 3 months, the function will return Apr 10th, 2017.

The syntax is:

```
ADD_MONTHS (input_date, number_of_months)
```

An example of this function is:

```
SELECT SYSDATE,
ADD_MONTHS(SYSDATE, 5) AS new_date
FROM dual;
```

SYSDATE	NEW_DATE
10/SEP/17	10/FEB/18

This query shows the SYSDATE, as well as the SYSDATE with 5 months added. You can see that it has the same day, but it is 5 months in the future.

You can also provide negative values:

```
SELECT SYSDATE,
ADD_MONTHS(SYSDATE, -12) AS new_date
FROM dual;
```

SYSDATE	NEW_DATE
10/SEP/17	10/SEP/16

This will show a date 12 months in the past.

For more information on using the ADD_MONTHS function, including some “what if” questions, read this article: Oracle ADD_MONTHS Function with Examples.

LAST_DAY

Back to Top

The LAST_DAY function returns the last day of the month of the specified date value.

You supply a date, and another date is returned which is the last day of the month.

The syntax is:

```
LAST_DAY (input_date)
```

Let’s see an example:

```
SELECT SYSDATE,
LAST_DAY(SYSDATE) AS last_of_month
FROM dual;
```

SYSDATE	LAST_OF_MONTH
10/SEP/17	30/SEP17

You can see here that I’ve shown today’s date using SYSDATE, and also the last day of this month using the LAST_DAY function.

What if I wanted to specify a particular date?

You can do that as well, by entering the date in the database's default format (usually DD-MON-YY) or using the TO_DATE function.

```
SELECT LAST_DAY('03-OCT-2016') AS last_of_month
FROM dual;
```

LAST_OF_MONTH
31/OCT/2016

For more information on the LAST_DAY function, such as returning the first day of the next month (and other examples), read my article: Oracle LAST_DAY Function with Examples.

NEXT_DAY

Back to Top

The NEXT_DAY function returns the date of the next specified weekday that comes after the specified value.

It's handy for working with any business logic you have where you need to use weekdays.

It takes two parameters:

```
NEXT_DAY (input_date, weekday)
```

The input date is the date to start from, and the weekday is the name of the day you're looking for.

For example, running SYSDATE on today's date will show 4th Apr 2017, and it's a Tuesday.

To find the next Thursday that appears after today, use this function:

```
SELECT NEXT_DAY(SYSDATE, 'THURSDAY') AS nextday
FROM dual;
```

NEXTDAY
14/SEP/17

You can see it shows a date of 6th April, which is the next Thursday that comes after the date I mentioned.

If I want to find the next Monday, I change the function slightly:

```
SELECT NEXT_DAY(SYSDATE, 'MONDAY') AS nextday
FROM dual;
```

NEXTDAY
11/SEP/17

You can see it shows a date of 10th April, which is next Monday. It doesn't show the day before, even though it's closer. It always goes forwards.

For more information on the NEXT_DAY function, read my article here: Oracle NEXT_DAY Function with Examples.

MONTHS_BETWEEN

Back to Top

The MONTHS_BETWEEN function allows you to find the number of months between two specified dates.

You specify two dates as parameters, and a number is returned:

```
MONTHS_BETWEEN (date1, date2)
```

Date1 is usually the later date.

If date1 is greater than date2, the value is positive. Otherwise, it is negative.

The returned value can be a whole or a decimal number. So, it returns a partial month value.

Let's see an example:

```
SELECT MONTHS_BETWEEN('12-MAY-2017', '10-FEB-2017') AS months_bt
FROM dual;
```

MONTHS_BT
3.064516129

The value returned is 3.06 because there is approximately 3.06 months between these two dates.

ROUND

Back to Top

The ROUND function allows you to round a date value to a format you specify.

This function is often used with numbers, but can also be used with dates.

If you use it with a date value, you can specify a DATE or TIMESTAMP value. You can specify any format mask, but the default is the nearest day, and is returned as a DATE.

The syntax is:

```
ROUND (input_date, round_to)
```

If I wanted to round a date to the nearest month, I would use something like this:

```
SELECT ROUND(SYSDATE, 'MM') AS rounded_date
FROM dual;
```

ROUNDED_DATE
01/SEP/17

This shows a value of 1 Sep because the specified date (SYSDATE, or 10 Sep) has been rounded forwards to this date.

If we specify a different date and input:

```
SELECT ROUND(SYSDATE, 'YYYY') AS rounded_date
FROM dual;
```

ROUNDED_DATE
01/JAN/18

We can see that the date shown is 1 Jan 2018. This is because it has rounded forward to the nearest year, and in this case, it is the start of 2018.

For more information on the ROUND function, read my article: Oracle ROUND Function with Examples.

TRUNC

Back to Top

The TRUNC function, like the round function, works with numbers as well as dates.

It truncates or removes a part of the date to the format you specify.

The syntax is:

```
TRUNC (input_date, format_mask)
```

If you don't specify a format mask, then the function will truncate the value to the nearest day. This is helpful if you want to remove the time part of a date value.

For example, to show only the date part of today's date:

```
SELECT TRUNC(SYSDATE) AS today_date
FROM dual;
```

TODAY_DATE
10/SEP/17

This shows only the date part of today.

Or, you can show only the YEAR:

```
SELECT TRUNC(SYSDATE, 'YYYY') AS curr_year
FROM dual;
```

CURR_YEAR
01/JAN/17

You can see that a date is shown that matches the first day of the current year.

For more information on the TRUNC function, read my article here: Oracle TRUNC Function with Examples.

EXTRACT

Back to Top

The EXTRACT function in Oracle extracts a specific part of a date from a date or interval value.

This means that it can get the month, or year, for example, from a DATE value. I think it's easier than using a conversion function such as TO_CHAR.

The EXTRACT function looks like this:

```
EXTRACT (date_component FROM expression)
```

The date_component is a keyword that represents the part of the date to extract, such as MONTH, DAY, YEAR, HOUR.

The expression is a value or column that is a date or interval data type.

An example of this function is:

```
SELECT SYSDATE,
EXTRACT(MONTH FROM SYSDATE) AS extract_month
FROM dual;
```

SYSDATE	EXTRACT_MONTH
10/SEP/17	9

This shows the month part of the SYSDATE.

For more information on the EXTRACT function, you can read my article here: Oracle EXTRACT Function with Examples.

TO_DATE

Back to Top

The TO_DATE function allows you to convert a character value into a date value. It converts any character data type (CHAR, VARCHAR2, NCHAR, or NVARCHAR2) into a DATE type.

It's useful if you have a date in a particular format in a text format or text column, and you need it in a DATE format (for a function or to insert into a column, for example).

The syntax is:

```
TO_DATE (date_text [, format_mask] [, nls_date_language])
```

The date_text is the date you want to convert, which is in some kind of text or character format. You can optionally specify the format mask (which is the format that this date value was provided in), and the nls_date_langauge is used for dates in different languages or countries.

An example of this function is:

```
SELECT TO_DATE('21-JAN-2017', 'DD-MON-YYYY') AS converted_date
FROM dual;
```

CONVERTED_DATE
21/JAN/17

This shows the specified date (21 Jan 2017) converted to a date format. It might look the same in your IDE, but that's just how dates are displayed. The value started as a character value and is converted to a date value.

If your date is in a different format:

```
SELECT TO_DATE('20170115_142309', 'YYYYMMDD_HH24MISS') AS converted_date
FROM dual;
```

CONVERTED_DATE
15/JAN/17

This example shows a completely different format, but it can still be converted to a date format. We have specified the format mask here, which is the format that the first parameter is in.

For more information on the TO_DATE function, read my article here: Oracle TO_DATE Function with Examples.

NEW_TIME

Back to Top

The NEW_TIME function converts a date from one timezone to another.

It's not something I've used very often, but if you need to work with different timezones, it can be very helpful.

The syntax is:

```
NEW_TIME (input_date, input_timezone, output_timezone)
```

To use this function, you specify the input_date, and the timezone this date is in as the input_timezone. You then specify the timezone you want to convert to in the output_timezone.

Let's see an example:

```
SELECT SYSDATE,
NEW_TIME(SYSDATE, 'GMT', 'PST') AS converted_time
FROM dual;
```

SYSDATE	CONVERTED_TIME
10/SEP/17	10/SEP/17

This converts the current date and time from a GMT timezone to a PST timezone. Note that you can only see the dates here and not times, because I haven't changed the session settings or used a function such as TO_CHAR to format the output.

I'll explain all about the TO_CHAR function and using it to format dates below, but here's a brief example:

```
SELECT TO_CHAR(SYSDATE, 'dd-mm-yy hh:mi:ss AM') AS sysdate_time,
TO_CHAR(NEW_TIME(SYSDATE, 'GMT', 'PST'), 'dd-mm-yy hh:mi:ss AM') AS newtime_test
FROM dual;
```

SYSDATE_TIME	NEWTIME_TEST

10-09-17 09:16:37 AM	10-09-17 01:16:37 AM

This now shows the date, which was in GMT but then converted to PST.

FROM_TZ

Back to Top

This function converts a TIMESTAMP value and a specified TIME ZONE to a TIMESTAMP WITH TIME ZONE value.

If you need a value that's in a TIMESTAMP WITH TIME ZONE format, then this is the function to use.

The syntax is:

FROM_TZ (timestamp_value, timezone_value)

An example of this function is:

```
SELECT FROM_TZ(TIMESTAMP '2017-04-19 07:13:50', '-9:00') AS from_tz_output
FROM dual;
```

FROM_TZ_OUTPUT
19/APR/17 07:13:50.000000000 AM -09:00

This converts the timestamp specified (which is a date and time) into a TIMESTAMP WITH TIME ZONE, in the timezone of -9 hours from GMT.

Note the TIMESTAMP keyword in front of the date and time value, which denotes that the value is a TIMESTAMP value.

SYS_EXTRACT_UTC

Back to Top

The SYS_EXTRACT_UTC will extract or convert the specified date and time into a UTC (also known as GMT) date and time.

You specify a data type that has a timezone, and a TIMESTAMP is returned that shows the time in UTC time.

For example, we can convert this date to UTC:

```
SELECT SYS_EXTRACT_UTC(TIMESTAMP '2017-05-15 19:10:45 +10:00') AS utc_time
FROM dual;
```

UTC_TIME
15/MAY/17 09:10:45.000000000 AM

This converts the specified time, which is +10 UTC, into a UTC time. The UTC time is 10 hours before the specified time.

This function will return a different day if the conversion results in the output date being a different day to the input date:

```
SELECT SYS_EXTRACT_UTC(TIMESTAMP '2017-05-15 22:06:12 -7:00') AS utc_time
FROM dual;
```

UTC_TIME
16/MAY/17 05:06:12.000000000 AM

This moves the time forward to get to UTC time, which results in the date being on a different day.

SESSIONTIMEZONE

Back to Top

The SESSIONTIMEZONE function returns the timezone offset of your session, in the format of [+|-]TZH:TZM, or a time zone region name.

This can be helpful to know, especially if you're doing a lot of work with dates and work on databases in different time zones.

The syntax for the SESSIONTIMEZONE function is quite simple.

```
SESSIONTIMEZONE
```

There is really only one way to use the SESSIONTIMEZONE function.

```
SELECT SESSIONTIMEZONE
FROM dual;
```

Result:

SESSIONTIMEZONE
Australia/Sydney

This function shows that I am in the Sydney time zone.

How Can You Change the Session Time Zone?

While you can't easily change the database time zone, the session time zone is something that is easy to change.

To change the session time zone, you run the ALTER SESSION command:

```
ALTER SESSION SET TIME_ZONE = '+8:0';
```

What's The Difference Between SESSIONTIMEZONE and CURRENT_TIMESTAMP?

Both of these functions look at the time of the session. But there are some differences.

- CURRENT_TIMESTAMP returns the entire date and time, including the time zone.
- SESSIONTIMEZONE returns only the time zone.

Sometimes, SESSIONTIMEZONE may be easier to use than trying to perform string manipulation on the CURRENT_TIMESTAMP function.

DBTIMEZONE

Back to Top

The DBTIMEZONE function returns the timezone offset of the database, in the format of [+|-]TZH:TZM, or a time zone region name.

It's helpful when working with dates to know what timezone the database is in. This is easier than using one of the other date functions, or performing an extraction from a TIMESTAMP WITH TIME ZONE value.

This DBTIMEZONE function is how to check the timezone in Oracle database 11g – and all other database versions.

The syntax for the DBTIMEZONE function is quite simple:

```
DBTIMEZONE
```

There is really only one way to use the Oracle DBTIMEZONE function.

```
SELECT DBTIMEZONE
FROM dual;
```

Result:

DBTIMEZONE
+00:00

This shows that my database is in the UTC + 0 time zone.

How Can You Change the Database Time Zone?

The database time zone is actually the time zone of the operating system of the server it runs on.

So, you can't specifically change the time zone on the database.

You can view it by running this query:

```
SELECT SESSIONTIMEZONE FROM DUAL;
```

Or, this query will show the current time with timezone:

```
SELECT current_timestamp FROM DUAL;
```

What's the Difference Between Oracle DBTIMEZONE and SESSIONTIMEZONE in Oracle?

The difference between these two functions is:

- DBTIMEZONE returns the time zone of the database server.
- SESSIONTIMEZONE returns the time zone of your session.

These two values may be different, if you're in a different location to your database.

What's The Difference Between Oracle DBTIMEZONE vs SYSTIMESTAMP?

The difference between DBTIMEZONE and SYSTIMESTAMP is:

- DBTIMEZONE returns just the timezone offset of the database (e.g. +7:00).
- SYSTIMESTAMP returns the timestamp with time zone of the database (e.g. 13-JUL-2016 08:49:02 AM +7:00)

So, the SYSTIMESTAMP includes more information than the DBTIMEZONE function.

Formatting Dates

Back to Top

So far we're worked with date values. We've changed their data type, worked with timezones, and even extracted and truncated parts of them.

However, you might have noticed that the data types are displayed in a certain format. And, that the DATE data type doesn't show the time, even though the time is included.

Fortunately, Oracle lets you format a DATE value for your output. This can be helpful to check if your function is returning the right value in a query, or for displaying an output in your application.

You can do this with the TO_CHAR function.

For example, we show the SYSDATE like this:

```
SELECT SYSDATE
FROM dual;
```

SYSDATE

10/SEP/17

If we want to see it in a different format, such as 2017-04-06, we can write this:

```
SELECT SYSDATE,
TO_CHAR(SYSDATE, 'YYYY-MM-DD') AS formatted_date
FROM dual;
```

SYSDATE	FORMATTED_DATE
10/SEP/17	2017-09-10

You can see that the output is in a different format. It has 4 digits for the year, then a dash, then two digits for the month, then a dash, then two digits for the day.

You can also use the TO_CHAR function to show the time. This is a common way to use this function.

```
SELECT SYSDATE,
TO_CHAR(SYSDATE, 'YYYY-MM-DD HH:MI:SS AM') AS formatted_date
FROM dual;
```

SYSDATE	FORMATTED_DATE
10/SEP/17	2017-09-10 09:21:29 AM

This shows the same date, but in a different format. It shows the hours, minutes, seconds, and an AM/PM marker.

Now, you're probably wondering, what do all of those formatting codes mean? Some of them may seem obvious (such as YYYY means a 4-digit year). But what about the others?

Let's look at them now.

Date Format Parameters

Back to Top

In many Oracle functions that deal with dates, such as TO_CHAR, you can specify a "format mask". This is a parameter that lets you specify a certain combination of characters, which allow Oracle to translate into a specific format for a date.

You can use these characters to change the way that dates are formatted.

A full list of the date format parameters are shown here:

Year

Parameter	Explanation
YEAR	Year, spelled out in full words
YYYY	4-digit year
YYY	Last 3 digits of year
YY	Last 2 digits of year
Y	Last digit of year
IYY	Last 3 digits of ISO year
IY	Last 2 digits of ISO year
I	Last digit of ISO year
IYYY	4-digit year, which is based on the ISO standard
RRRR	This format accepts a 2-digit year, and returns a 4-digit year. If the provided value is between 0 and 49, it will return a year greater than or equal to 2000. If the provided value is between 50 and 99, it will return a year less than 2000

Month

Parameter	Explanation
Q	Quarter of year, from 1 to 4. JAN to MAR = 1
MM	Month, from 01 to 12. JAN = 01
MON	Abbreviated name of month.
MONTH	Name of month, padded with blanks to length of 9 characters.
RM	Roman numeral month, from I to XII. JAN = I.

Week

Parameter	Explanation
WW	Week of year, from 1 to 53. Week 1 starts on the first day of the year, and continues to the seventh day of the year.
W	Week of month, from 1 to 5. Week 1 starts on the first day of the month and ends on the seventh.
IW	Week of year, from 1 to 52 or 1 to 53, based on the ISO standard.

Day

Parameter	Explanation
D	Day of week, from 1 to 7.
DAY	Name of day.
DD	Day of month, from 1 to 31.
DDD	Day of year, from 1 to 366.
DY	Abbreviated name of day.
J	Julian day, which is the number of days since January 1, 4712 BC.

Time

Parameter	Explanation
HH	Hour of day, from 1 to 12.
HH12	Hour of day, from 1 to 12.
HH24	Hour of day, from 0 to 23.
MI	Minute, from 0 to 59
SS	Second, from 0 to 59
SSSSS	Seconds past midnight, from 0 to 86399.
FF	Fractional seconds. This uses a value from 1 to 9 after FF, to indicate the number of digits in the fractional seconds (e.g. FF7)

Indicators

Parameter	Explanation
AM, A.M., PM, or P.M.	Meridian indicator
AD or A.D	AD indicator
BC or B.C.	BC indicator
TZD	Daylight savings information
TZH	Time zone hour.

TZM	Time zone minute.
TZR	Time zone region.

Common Uses for Date Functions

Back to Top

So, we've seen explanations of all of the date functions in Oracle, and some examples for how they work.

Now, let's take a look at some common examples of date functions.

These examples will show you how to do common tasks in Oracle using dates. So, if you're looking up how to find the date one year from now, or format a date in a certain way, this is where you'd look.

Let's see the examples.

Find the Number of Days Between Two Dates

To find the number of days between two dates, you can simply subtract one date from another. You don't need to use a function for this.

For example, to find the number of days between 31 Dec 2017 and today, run this query:

```
SELECT TO_DATE('31-DEC-2017') - SYSDATE AS days_to_go
FROM dual;
```

DAYS_TO_GO
111.6092245

Add Days To A Date

Just like subtracting two dates, you can add a number to a date to get a number of days in the future. Adding numbers to a date is treated as adding days.

```
SELECT SYSDATE + 7 AS next_week_date
FROM dual;
```

NEXT_WEEK_DATE
17/SEP/17

This shows the date 7 days from now.

Subtracting Days From A Date

To subtract days from a date, just subtract a number. A number represents the number of days.

```
SELECT SYSDATE - 21 AS old_date
FROM dual;
```

OLD_DATE
20/AUG/17

This shows the date 21 days in the past.

Show Today's Date and Time

To show today's date and time, you can use the SYSDATE function, along with the TO_CHAR function to format it. Alternatively, you can use the CURRENT_DATE function.

SYSDATE shows the server's date and time, and CURRENT_DATE shows your session's date and time.

You can write a query like this:

```
SELECT TO_CHAR(SYSDATE, 'DD-MM-YYYY HH:MI:SS AM') AS curdate
FROM dual;
```

CURDATE
10-09-2017 09:24:11 AM

Or, you can write the same query using CURRENT_DATE, if your session time is more important (if, for example, your database is in another timezone and you want to find the time where you are now).

```
SELECT TO_CHAR(CURRENT_DATE, 'DD-MM-YYYY HH:MI:SS AM') AS curdate
FROM dual;
```

CUR_DATE
10-09-2017 09:24:11 AM

Add One Year To A Date

Earlier, I showed you how to add a number of days to a date. This is helpful for adding days, but when it comes to adding years, using days is not reliable.

This is because the number of days in each year is not consistent. Rather than using a complicated lookup table and formula, you can just use the ADD_MONTHS function.

To add one year to today's date:

```
SELECT ADD_MONTHS(SYSDATE, 12) AS one_year
FROM dual;
```

ONE_YEAR
10/SEP/18

Subtract Years From a Date

Similar to adding a year to a date, you can also subtract years from a date. This is best done using the ADD_MONTHS function and providing a negative number value.

To find the date 5 years before a specific date:

```
SELECT ADD_MONTHS('10-JUL-2017', -60) AS old_date
FROM dual;
```

ONE_YEAR
10/JUL/12

Notice that I added the number 60 in there, because 5 years * 12 months is 60.

It could be easier to read and configure if you use the number of years * 12:

```
SELECT ADD_MONTHS('10-JUL-2017', 5 * -12) AS old_date
FROM dual;
```

ONE_YEAR
10/JUL/12

That way, you can just change the number 5 if you ever need to change this, rather than work out how many months are in a year.

Find the Number of Months Between Two Dates

To find the number of months between two dates in Oracle, use the MONTHS_BETWEEN function.

This is helpful to find the number of months an employee has worked at a company, or the number of months since a sale was made, for example.

```
SELECT employee_id,
hire_date,
ROUND(MONTHS_BETWEEN(SYSDATE, HIRE_DATE), 1) AS months_worked
FROM employee;
```

EMPLOYEE_ID	HIRE_DATE	MONTHS_WORKED
1	27/AUG/11	72.5
2	2/JAN/12	68.3
3	4/DEC/16	9.2
4	12/OCT/11	70.9
5	15/NOV11	69.9
6	2/JUL/10	86.3
7	3/OCT/10	83.2
8	20/MAY10	87.7
9	15/MAR/15	29.9
10	3/JUL14	38.2

This shows the ID, the hire date, and the number of months the employee has worked at the company.

Note that I used the ROUND function, to clean up the output, as the MONTHS_BETWEEN returns a decimal number that could be quite long.

Get the First Day of the Month

To find the first day of the month in Oracle SQL, you can use the TRUNC function.

```
SELECT TRUNC(SYSDATE, 'MONTH') AS first_day
FROM dual;
```

FIRST_DAY
01/SEP/17

Get the Last Day of the Month

To find the last day of the month, use a combination of the TRUNC and LAST_DAY functions.

```
SELECT TRUNC(LAST_DAY(SYSDATE)) AS last_day
FROM dual;
```

LAST_DAY
30/SEP/17

Get the First Day of the Year

To get the first day of the year, which is always Jan 1st, use the TRUNC function to truncate it to a year.

```
SELECT TRUNC(SYSDATE, 'YEAR') AS first_of_year
FROM dual;
```

FIRST_OF_YEAR
01/JAN/17

Get the Last Day Of The Year

The last day of the year is always 31 Dec. To find the last day of the year in SQL, use this query:

```
SELECT ADD_MONTHS(TRUNC(SYSDATE, 'YEAR'), 12) - 1 AS last_day_of_year
FROM dual;
```

LAST_DAY
31/DEC/17

This query first truncates the current date to the first day of the year (1 Jan), then adds 12 months to it (to make it 1 Jan of the next year). Then, it subtracts 1 day from that, to find the last day of the current year.

Get the Number of Days in a Month

To find the number of days in a particular month, whether it's the current month or a month from another date, use this query.

It uses the LAST_DAY, TO_CHAR, and CAST functions.

```
SELECT CAST(TO_CHAR(LAST_DAY(SYSDATE), 'dd') AS INT) AS number_of_days
FROM dual;
```

NUMBER_OF_DAYS
30

This finds the last day of the month using LAST_DAY, then converts it to an INT value. You can use any date here instead of SYSDATE.

Conclusion

Back to Top

So, this article has highlighted all of the DATE functions in Oracle. I've explained what they do, provided a small example, and covered a few common and useful ways to use these date functions.

If you have any questions on these functions, please let me know!

Lastly, if you enjoy the information and career advice I've been providing, sign up to my newsletter below to stay up-to-date on my articles. You'll also receive a fantastic bonus. Thanks!

← Previous Post

Next Post →

5 thoughts on "Oracle Date Functions: The Complete Guide"

CKRISHNAN
APRIL 2, 2018 AT 1:32 AM

Wow simply superb.

Reply

BIKRAM
MAY 1, 2018 AT 5:55 AM

Very well compiled.
Thanks.

Reply

CKRISHNAN
AUGUST 5, 2018 AT 2:20 AM

Excellent

Reply

RICK
FEBRUARY 25, 2019 AT 10:37 AM

Impressive. Great job!

Reply

CHRIS VEERAGALOO
MAY 10, 2019 AT 10:45 PM

Thanks, Ben. Will prove very useful. I will use it often. Cheers.

Reply

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

Name*

Email*

Website

Notify me of follow-up comments by email.

Notify me of new posts by email.

Post Comment »

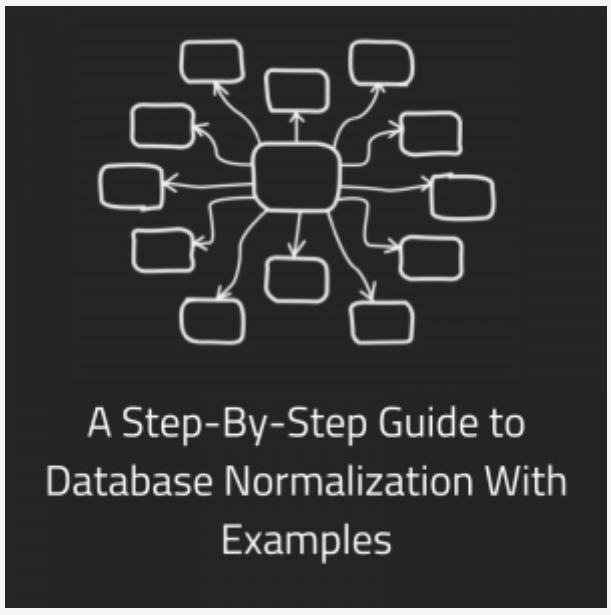
Ben Brumm

DatabaseStar

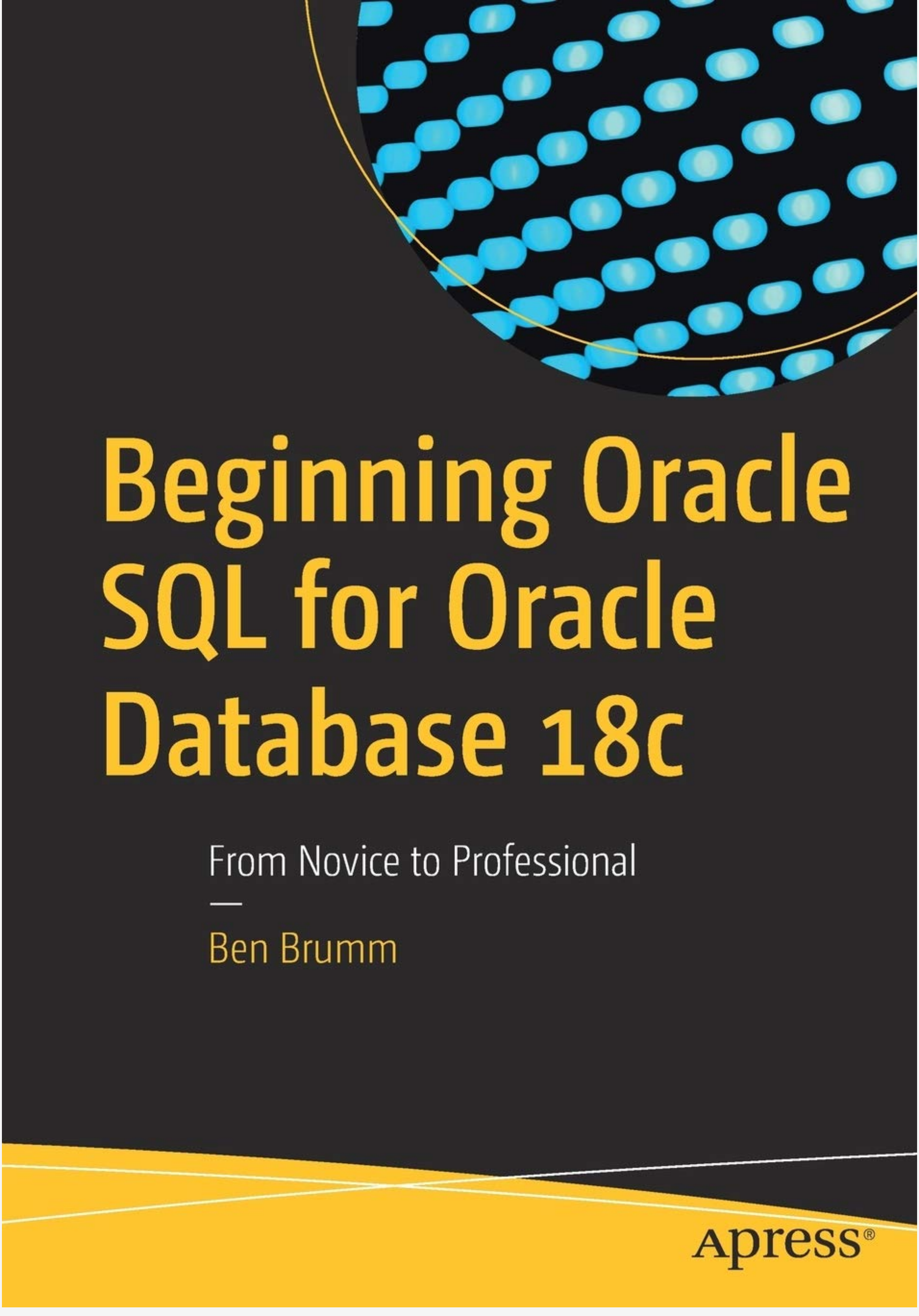
Database Star Academy:

Login

Popular Posts




The Definitive Guide
to Becoming a
Database Developer



Get my book: Beginning Oracle SQL for Oracle Database 18c