Table of Contents

This document is focusing on the functions that Oracle provides to deal with dates. You should use these functions rather than trying to build your own functions or expressions. For example, if you want to get the month of a date value as the month named spelled out, you use

```
select To_char(sysdate, 'FMMonth') from dual;
```

```
TO_CHAR(S
---------
February
```

You do not extract the month as number and write a 12 part case expression to get the month name.

If you want to find a date two months after a specified date, you use the Add_Months function.

1. Extracting part of a date value

Demo 01: Extract date part

```
select hire_date
, EXTRACT (YEAR   FROM hire_date) AS "YearHired"
, EXTRACT (MONTH  FROM hire_date) AS "MonthHired"
, EXTRACT (DAY    FROM hire_date) AS "DayHired"
from emp_employees
where rownum < 4;
```

```
HIRE_DATE  YearHired MonthHired   DayHired
--------- ---------- ---------- ----------
17-JUN-89       1989          6         17
30-MAR-01       2001          3         30
28-OCT-01       2001         10         28
```

Demo 02: We want the people hired in August of any year.

```
select emp_id, hire_date
from emp_employees
where EXTRACT (MONTH FROM hire_date) = 8;
```

```
    EMP_ID HIRE_DATE
---------- ----------
       201  25-AUG-04
       103  01-AUG-10
```

Demo 03: Who has been hired in the current year?

```
select emp_id
, hire_date
from emp_employees
where EXTRACT (YEAR FROM hire_date) = EXTRACT (YEAR FROM sysdate);
```

```
no rows selected
```

Demo 04: Who has been hired in the year three years ago?

```
select emp_id
, hire_date
from emp_employees
where EXTRACT (YEAR FROM hire_date) = EXTRACT (YEAR FROM sysdate) - 3;
```

```
    EMP_ID HIRE_DATE
---------- ---------
       206 15_JUN-13
       204 15_JUN-13
```

Sometimes when you talk about date arithmetic, you have to double check the meaning. This is not always what people would mean by "who was hired in the last three years".

2. Date arithmetic

You can do arithmetic with date values. You can add or subtract a date value and a number. You can subtract two date values. You cannot do multiplication or division with date values.

From the example below you can see that the basic unit of time in Oracle is the day- if we add 1 to a date value we are adding 1 day. Note that this approach to date arithmetic is not common in other systems.

Demo 05: Date arithmetic.

```
select sysdate
, sysdate + 2   as Plus2
, sysdate - 40  as minus40
, sysdate + 285 as Plus285
from dual
;
```

```
SYSDATE   PLUS2     MINUS40   PLUS285
--------- --------- --------- ---------
13-FEB-16 15-FEB-16 04-JAN-16 24-NOV-16
```

Demo 06: Date arithmetic.

```
select MyDate, MyDate - to_date('15-May-2003')   DaysBetween
from z_tst_cal
where    ID <=5
;
```

```
MYDATE       DAYSBETWEEN
---------    -----------
01-JAN-03           -134
28-FEB-03            -76
11-MAY-03             -4
05-JUN-03             21
08-SEP-03            116
```

Demo 07: date arithmetic. The results will vary with the date this is run.

```
select emp_id, hire_date
, trunc(SYSDATE - hire_date)as "How Many Days Since Hired"
from emp_employees;
```

```
    EMP_ID HIRE_DATE How Many Days Since Hired
---------- --------- ----------------------
       100 17-JUN-89                   9737
       145 30-MAR-01                   5433
       150 28-OCT-01                   5221
       155 05-MAR-04                   4362
       201 25-AUG-04                   4189
. . . rows omitted
```

3. Date manipulation functions

3.1. Add_Months

Demo 08: ADD_MONTHS(date, count) adds (or subtracts) the specified number of months.

```
select MyDate
, ADD_MONTHS ( MyDate, 2)    AS Col1
, ADD_MONTHS ( MyDate, -1)   AS Col2
from z_tst_cal
where ID <=4
;
```

```
MYDATE                    ADD_MONTHS(MYDATE,2)      ADD_MONTHS(MYDATE,-1)
------------------------- ------------------------- -------------------------
01-JAN-03                 01-MAR-03                 01-DEC-02
28-FEB-03                 30-APR-03                 31-JAN-03
11-MAY-03                 11-JUL-03                 11-APR-03
05-JUN-03                 05-AUG-03                 05-MAY-03
```

3.1. Months_Between

Demo 09: MONTHS_BETWEEN (date1, date2)—The time between two dates expressed as a decimal value. The fractional part of the return value assumes a 31 day month.

```
select MyDate
, MONTHS_BETWEEN ('15-May-2003', MyDate)           AS Col1
, FLOOR( MONTHS_BETWEEN ('15-May-2003', MyDate) ) AS Col2
from z_tst_cal
where ID <=5
;
```

```
MYDATE             COL1          COL2
---------    ----------    ----------
01-JAN-03     4.4516129             4
28-FEB-03    2.58064516             2
11-MAY-03    .129032258             0
05-JUN-03    -.67741935            -1
08-SEP-03    -3.7741935            -4
```

Demo 10: MONTHS_BETWEEN with order dates.

```
select order_id
, order_date
from oe_orderHeaders
where months_between(date '2015-10-15', order_date) between 0.75 and 2;
```

```
    ORD_ID ORD_DATE
---------- ---------
       414 20-AUG-15
       415 23-AUG-15
       605 05-SEP-15
       606 07-SEP-15
       607 15-SEP-15
```

3.2. Last_Day

Demo 11: LAST_DAY (date)— returns the last day of the month of the argument.

```
select MyDate
, LAST_DAY ( MyDate)      AS  Col1
, LAST_DAY ( MyDate) +1   AS  Col2
from z_tst_cal
where ID <=4
;
```

MYDATE	COL1	COL2
01-JAN-03	31-JAN-03	01-FEB-03
28-FEB-03	28-FEB-03	01-MAR-03
11-MAY-03	31-MAY-03	01-JUN-03
05-JUN-03	30-JUN-03	01-JUL-03

3.3. Next_Day

Demo 12: NEXT_DAY (date, 'DAYINDICTOR')— returns the next date of the indicated day of the week. The return date will always be greater than the argument date.

```
select MyDate
, NEXT_DAY ( MyDate, 'SUN')      Col1
, NEXT_DAY ( MyDate, 'Friday')   Col2
, NEXT_DAY ( MyDate - 1, 'SUN')    Col3
, NEXT_DAY ( MyDate - 1, 'Friday') Col4
from z_tst_cal
where ID <=5
;
```

MYDATE	COL1	COL2	COL3	COL4
01-JAN-03	05-JAN-03	03-JAN-03	05-JAN-03	03-JAN-03
28-FEB-03	02-MAR-03	07-MAR-03	02-MAR-03	28-FEB-03
11-MAY-03	18-MAY-03	16-MAY-03	11-MAY-03	16-MAY-03
05-JUN-03	08-JUN-03	06-JUN-03	08-JUN-03	06-JUN-03
08-SEP-03	14-SEP-03	12-SEP-03	14-SEP-03	12-SEP-03

3.4. Using To_Char to extract part of a date

Demo 13: Another way to do this- To_Char returns string; add To_Number to get actual numeric values returned. This has to do two conversions if I want to get numbers.

```
select hire_date
, TO_NUMBER(TO_CHAR (hire_date, 'YYYY')) AS "YearHired"
, TO_NUMBER(TO_CHAR (hire_date, 'MM'))   AS "MonthHired"
from emp_employees;
```

Demo 14: Alternate syntax. But note that the To_char returns a string value which you are now comparing to a numeric value requiring more data casting.

```
select emp_id, hire_date
from emp_employees
where TO_CHAR (hire_date, 'MM') = 8;
```

What do you suppose happens if you decide that you will change the Where clause to do a string to string test avoiding one of the data casting steps.

```
select emp_id, hire_date
from emp_employees
where TO_CHAR (hire_date, 'MM') = '8';
```

4. Using Round, Trunc, Greatest, Least with date values

The TRUNC function will take a date value and truncate the time component to 12:00 am (midnight) of that date. The ROUND function will set the time component to12:00 a.m. of that date if the time part is before noon and to 12:00 a.m. of the next day if the time component is at or after noon.

If you use GREATEST, LEAST with date literals you need to include to TO_DATE function. A date literal such as '12-JAN-02' looks just like a string literal.

Demo 15: The Round function will let you round date values to a specified precision. I had to wrap these dates in a TO_CHAR to show the full date values.

```
select ID
, TO_CHAR( MyDate,                 'YYYY-MM-DD BC') Col0
, TO_CHAR( ROUND (MyDate, 'DD'),   'YYYY-MM-DD BC') Col1
, TO_CHAR( ROUND (MyDate, 'YYYY'), 'YYYY-MM-DD BC') Col2
, TO_CHAR( ROUND (MyDate, 'MM'),   'YYYY-MM-DD BC') Col3
from z_tst_cal   ;
```

```
-- selected rows
        ID    COL0             COL1             COL2             COL3
----------    -------------    -------------    -------------    -------------
         1    2003-01-01 AD    2003-01-01 AD    2003-01-01 AD    2003-01-01 AD
         2    2003-02-28 AD    2003-02-28 AD    2003-01-01 AD    2003-03-01 AD
         3    2003-05-11 AD    2003-05-11 AD    2003-01-01 AD    2003-05-01 AD
         4    2003-06-05 AD    2003-06-05 AD    2003-01-01 AD    2003-06-01 AD
        10    4000-01-01 BC    4000-01-01 BC    4000-01-01 BC    4000-01-01 BC
        11    9999-12-31 AD    9999-12-31 AD    0000-00-00 00    0000-00-00 00
```

Demo 16: The TRUNC function will truncate to a specified precision.

```
select ID
, TO_CHAR( MyDate,                 'YYYY-MM-DD BC') Col0
, TO_CHAR( TRUNC (MyDate, 'DD'),   'YYYY-MM-DD BC') Col1
, TO_CHAR( TRUNC (MyDate, 'YYYY'), 'YYYY-MM-DD BC') Col2
, TO_CHAR( TRUNC (MyDate, 'MM'),   'YYYY-MM-DD BC') Col3
from z_tst_cal   ;
```

```
 -- selected rows
       ID    COL0             COL1             COL2             COL3
----------    -------------    -------------    -------------    -------------
         1    2003-01-01 AD    2003-01-01 AD    2003-01-01 AD    2003-01-01 AD
         2    2003-02-28 AD    2003-02-28 AD    2003-01-01 AD    2003-02-01 AD
         3    2003-05-11 AD    2003-05-11 AD    2003-01-01 AD    2003-05-01 AD
         4    2003-06-05 AD    2003-06-05 AD    2003-01-01 AD    2003-06-01 AD
        10    4000-01-01 BC    4000-01-01 BC    4000-01-01 BC    4000-01-01 BC
        11    9999-12-31 AD    9999-12-31 AD    9999-01-01 AD    9999-12-01 AD
```

Demo 17: Coerce the first argument to a date value.

```
select ID, MyDate
, GREATEST( TO_DATE ('15-MAY-2003', 'dd-MON-yy'), MyDate)
from z_tst_cal
where ID < =4;
```

```
        ID    MYDATE       GREATEST(
----------    ---------    ---------
         1    01-JAN-03    15-MAY-03
         2    28-FEB-03    15-MAY-03
         3    11-MAY-03    15-MAY-03
         4    05-JUN-03    05-JUN-03
```

Demo 18: Notice the values returned here- these are string comparisons

```
select ID, MyDate
, GREATEST ('15-MAY-2003', MyDate)
from z_tst_cal
where ID < =4
;
```

```
        ID    MYDATE       GREATEST('1
----------    ---------    -----------
         1    01-JAN-03    15-MAY-2003
         2    28-FEB-03    28-FEB-03
         3    11-MAY-03    15-MAY-2003
         4    05-JUN-03    15-MAY-2003
```

Demo 19: Why does this one work correctly?

```
select emp_id, hire_date
, GREATEST (hire_date, date '1990-01-01')
from emp_employees
where rownum < =6;
```

```
    EMP_ID HIRE_DATE GREATEST(
---------- --------- ---------
       100 17-JUN-89 01-JAN-90
       145 30-MAR-01 30-MAR-01
       150 28-OCT-01 28-OCT-01
       155 05-MAR-04 05-MAR-04
       201 25-AUG-04 25-AUG-04
       101 17-JUN-08 17-JUN-08
```

Demo 20: Be certain that you understand this one.

```
select GREATEST ('5', 90, 456, 23, -4)
, GREATEST (23, '5', 456, 90, -4)
from DUAL;
```

```
GR    GREATEST(23,'5',456,90-4)
--    -------------------------
90                          456
```

5. Things you should not do with dates

As I write this it is Feb 13, 2016. I can write a query that gets the current month.

```
select extract(month from sysdate) from dual;
```

```
EXTRACT(MONTHFROMSYSDATE)
-------------------------
                        2
```

Suppose I want to calculate which month it will be two months from now. I could try the following (which is WRONG even if the answer is correct. The query is wrong.)

```
select extract(month from sysdate) +2  from dual;
```

```
EXTRACT(MONTHFROMSYSDATE)+2
---------------------------
                          4
```

I know, two months from the date I write this is April- but if this is correct then I should be able to use this expression to find the month 12 months from now.

```
select extract(month from sysdate) +12  from dual;
```

```
EXTRACT(MONTHFROMSYSDATE)+12
----------------------------
                          14
```

Our calendar does not have a month 14 so this is NOT the way to do date arithmetic. Twelve months from now is Feb (month 2). Our calendar is cyclic; after month 12 comes month 1- not month 13. Some people will try to fix this by using a mod approach . But you can do this more clearly by using the date arithmetic functions.
Twelve months from now is Add_months (sysdate, 12)

```
select Add_months(sysdate, 12) from dual;
```

```
ADD_MONTH
---------
13-FEB-17
```

And the month, 12 months from now, is

```
select extract(month from Add_months(sysdate, 12) ) from dual;
```

The year 12 months from now is simple. (deriving that calculation yourself is not worth your time.)

```
select extract(year from Add_months(sysdate, 12) ) from dual;
```

The year and the month, 19 months **ago** is

```
select
  extract(year  from Add_months(sysdate, -19) ) as YR
, extract(month from Add_months(sysdate, -19) ) as MN
from dual;
```

```
        YR         MN
---------- ----------
      2014          7
```

Or

```
With CalcDate as (
    select Add_months(sysdate, -19) as dtm from dual
)
select
  extract(year  from dtm ) as YR
, extract(month from dtm ) as MN
from CalcDate;
```

Make the system do the date arithmetic.

After that, why is it ok to calculate the next year as extract(year from sysdate) + 1 ?

Our calendar does not use a cyclic year. We do not have a year where the next year goes to 1.