Unit 1 - Chapter 2

Oracle Built in Functions

There are two types of functions in Oracle.

1) Single Row Functions: Single row or Scalar functions return a value for every row that is processed in a query.

2) Group Functions: These functions group the rows of data based on the values returned by the query. This is discussed in SQL GROUP Functions. The group functions are used to calculate aggregate values like total or average, which return just one total or one average value after processing a group of rows.

There are four types of single row functions. They are:

1) Numeric Functions: These are functions that accept numeric input and return numeric values.

2) Character or Text Functions: These are functions that accept character input and can return both character and number values.

3) Date Functions: These are functions that take values that are of datatype DATE as input and return values of datatype DATE, except for the MONTHS_BETWEEN function, which returns a number.

4) Conversion Functions: These are functions that help us to convert a value in one form to another form. For Example: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE etc.

You can combine more than one function together in an expression. This is known as nesting of functions.

What is a DUAL Table in Oracle?

This is a single row and single column dummy table provided by oracle. This is used to perform mathematical calculations without using a table.

```
Select * from DUAL
```

Output:

```
DUMMY
-------
X
```

```
Select 777 * 888 from Dual
```

Output:

```
777 * 888
---------
689976
```

1) Numeric Functions:

Numeric functions are used to perform operations on numbers. They accept numeric values as input and return numeric values as output. Few of the Numeric functions are:

Function Name	Return Value
ABS (x)	Absolute value of the number '*x*'
CEIL (x)	Integer value that is Greater than or equal to the number '*x*'
FLOOR (x)	Integer value that is Less than or equal to the number '*x*'
TRUNC (x, y)	Truncates value of number '*x*' up to '*y*' decimal places
ROUND (x, y)	Rounded off value of the number '*x*' up to the number '*y*' decimal places

The following examples explains the usage of the above numeric functions

Function Name	**Examples**	**Return Value**
ABS (x)	ABS (1) ABS (-1)	1 -1
CEIL (x)	CEIL (2.83) CEIL (2.49) CEIL (-1.6)	3 3 -1
FLOOR (x)	FLOOR (2.83) FLOOR (2.49) FLOOR (-1.6)	2 2 -2
TRUNC (x, y)	ROUND (125.456, 1) ROUND (125.456, 0) ROUND (124.456, -1)	125.4 125 120
ROUND (x, y)	TRUNC (140.234, 2) TRUNC (-54, 1)	140.23 54

	TRUNC (5.7) TRUNC (142, -1)	5 140

These functions can be used on database columns.

For Example: Let's consider the product table used in sql joins. We can use ROUND to round off the unit_price to the nearest integer, if any product has prices in fraction.

```
SELECT ROUND (unit_price) FROM product;
```

2) Character or Text Functions:

Character or text functions are used to manipulate text strings. They accept strings or characters as input and can return both character and number values as output.

Few of the character or text functions are as given below:

Function Name	**Return Value**
LOWER (string_value)	All the letters in *'string_value'*is converted to lowercase.
UPPER (string_value)	All the letters in *'string_value'*is converted to uppercase.
INITCAP (string_value)	All the letters in *'string_value'*is converted to mixed case.
LTRIM (string_value, trim_text)	All occurrences of *'trim_text'* is removed from the left of*'string_value'*.
RTRIM (string_value, trim_text)	All occurrences of *'trim_text'* is removed from the right of*'string_value'* .
TRIM (trim_text FROM string_value)	All occurrences of *'trim_text'*from the left and right of*'string_value'* , *'trim_text'* can also be only one character long .
SUBSTR (string_value, m, n)	Returns *'n'* number of characters from *'string_value'*starting from the '*m'* position.
LENGTH (string_value)	Number of characters in*'string_value'* in returned.
LPAD (string_value, n, pad_value)	Returns '*string_value'* left-padded with *'pad_value'* . The length of the whole string will be of *'n'* characters.
RPAD (string_value, n, pad_value)	Returns '*string_value'* right-padded with *'pad_value'* . The length of the whole string will be of *'n'* characters.

For Example, we can use the above UPPER() text function with the column value as follows.

```
SELECT UPPER (product_name) FROM product;
```

The following examples explains the usage of the above character or text functions

Function Name	**Examples**	**Return Value**
LOWER(string_value)	LOWER('Good Morning')	good morning
UPPER(string_value)	UPPER('Good Morning')	GOOD MORNING

INITCAP(string_value)	INITCAP('GOOD MORNING')	Good Morning
LTRIM(string_value, trim_text)	LTRIM ('Good Morning', 'Good)	Morning
RTRIM (string_value, trim_text)	RTRIM ('Good Morning', ' Morning')	Good
TRIM (trim_text FROM string_value)	TRIM ('o' FROM 'Good Morning')	Gd Mrning
SUBSTR (string_value, m, n)	SUBSTR ('Good Morning', 6, 7)	Morning
LENGTH (string_value)	LENGTH ('Good Morning')	12
LPAD (string_value, n, pad_value)	LPAD ('Good', 6, '*')	**Good
RPAD (string_value, n, pad_value)	RPAD ('Good', 6, '*')	Good**

3) Date Functions:

These are functions that take values that are of datatype DATE as input and return values of datatypes DATE, except for the MONTHS_BETWEEN function, which returns a number as output.

Few date functions are as given below.

Function Name	**Return Value**
ADD_MONTHS (date, n)	Returns a date value after adding *'n'* months to the date*'x'*.
MONTHS_BETWEEN (x1, x2)	Returns the number of months between dates x1 and x2.
ROUND (x, date_format)	Returns the date *'x'* rounded off to the nearest century, year, month, date, hour, minute, or second as specified by the*'date_format'*.
TRUNC (x, date_format)	Returns the date *'x'* lesser than or equal to the nearest century, year, month, date, hour, minute, or second as specified by the 'date_format'.
NEXT_DAY (x, week_day)	Returns the next date of the*'week_day'* on or after the date*'x'* occurs.
LAST_DAY (x)	It is used to determine the number of days remaining in a month from the date *'x'*specified.
SYSDATE	Returns the systems current date and time.
NEW_TIME (x, zone1, zone2)	Returns the date and time in zone2 if date 'x' represents the time in zone1.

The below table provides the examples for the above functions

Function Name	**Examples**	**Return Value**
ADD_MONTHS ( )	ADD_MONTHS ('16-Sep-81', 3)	16-Dec-81
MONTHS_BETWEEN( )	MONTHS_BETWEEN ('16-Sep-81', '16-Dec-81')	3

NEXT_DAY( )	NEXT_DAY ('01-Jun-08', 'Wednesday')	04-JUN-08
LAST_DAY( )	LAST_DAY ('01-Jun-08')	30-Jun-08
NEW_TIME( )	NEW_TIME ('01-Jun-08', 'IST', 'EST')	31-May-08

4) Conversion Functions:

These are functions that help us to convert a value in one form to another form. For Ex: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE.

Few of the conversion functions available in oracle are:

Function Name	**Return Value**
TO_CHAR (x [,y])	Converts Numeric and Date values to a character string value. It cannot be used for calculations since it is a string value.
TO_DATE (x [, date_format])	Converts a valid Numeric and Character values to a Date value. Date is formatted to the format specified by *'date_format'*.
NVL (x, y)	If *'x'* is NULL, replace it with *'y'*. *'x'*and *'y'* must be of the same datatype.
DECODE (a, b, c, d, e, default_value)	Checks the value of *'a'*, if *a = b*, then returns *'c'*. If *a = d*, then returns *'e'*. Else, returns *default_value*.

The below table provides the examples for the above functions

Function Name	**Examples**	**Return Value**
TO_CHAR ()	TO_CHAR (3000, '$9999') TO_CHAR (SYSDATE, 'Day, Month YYYY')	$3000 Monday, June 2008
TO_DATE ()	TO_DATE ('01-Jun-08')	01-Jun-08
NVL ()	NVL (null, 1)	1

Numeric Functions:

These are functions that accept numeric input and return numeric values. Below are few of the examples

ABS: Absolute value of the number

```
SELECT ABS(12) FROM DUAL;
  ABS(12)
---------
       12
```

CEIL: Integer value that is Greater than or equal to the number

```
SQL> SELECT CEIL(48.99) FROM DUAL;
CEIL(48.99)

-----------
        49

SQL> SELECT CEIL(48.11) FROM DUAL;
CEIL(48.11)
-----------
        49
```

FLOOR: Integer value that is Less than or equal to the number

```
SQL> SELECT FLOOR(49.99) FROM DUAL;
FLOOR(49.99)
------------
         49

SQL> SELECT FLOOR(49.11) FROM DUAL;
FLOOR(49.11)
------------
         49
```

ROUND: Rounded off value of the number '*x*' up to the number '*y*' decimal places

```
SQL> SELECT ROUND(49.11321,2) FROM DUAL;
ROUND(49.11321,2)
-----------------
           49.11

SQL> SELECT ROUND(49.11321,3) FROM DUAL;
ROUND(49.11321,3)
-----------------
          49.113

SQL> SELECT ROUND(49.11321,4) FROM DUAL;
ROUND(49.11321,4)
-----------------
         49.1132
```

Few other functions,

POWER

```
SQL> SELECT POWER(4,2) FROM DUAL;
POWER(4,2)
----------
       16
```

MOD

```
SQL> SELECT MOD(4,2) FROM DUAL;
 MOD(4,2)
---------
       0

SQL> SELECT SIGN(-98) FROM DUAL;
SIGN(-98)
---------
      -1

SQL> SELECT SIGN(98) FROM DUAL;
 SIGN(98)
---------
       1
```

Character String:

Function 1: **UPPER**
Purpose : Returns the string in uppercase

Syntax : UPPER('str')

Example : SELECT UPPER('karuvachi') from Dual;

Output:KARUVACHI

————————————————————————————————————-

Function 2: **lower**
Purpose : Returns the string in lowercase

Syntax : lower('str')

Example : SELECT LOWER('KaRuVaChi') FROM DUAL;

Output:karuvachi

————————————————————————————————————-

Function 3: **Initcap**
Purpose : Returns the string with first letter in uppercase and rest of the letters in lowercase

Syntax : Initcap('str')

Example : SELECT Initcap('KaRuVaChi') FROM DUAL;

Output:Karuvachi

————————————————————————————————————-

Function 4: **Concat**
Purpose : Concatenate two strings

Syntax : concat(‘str1’,’str2’)

Example : SELECT CONCAT(‘Karu’,’Nand’) FROM DUAL;

Output:KaruNand

————————————————————————————————-

Function 5: **Lpad**
Purpose : Pad in the left side of the string for given times – length of the string

Syntax : Lpad(‘str1’,n,’str2’)

Example : SELECT Lpad(‘Karu’,6,’?’) FROM DUAL;

Output:??Karu

————————————————————————————————-

Function 6: **Rpad**
Purpose : Pad in the right side of the string for given times – length of the string

Syntax : Rpad(‘str1’,n,’str2’)

Example : SELECT Rpad(‘Karu’,6,’?’) FROM DUAL;

Output:Karu??

————————————————————————————————-

Function 7: **trim**
Purpose : Trim the whitespaces in both the sides of the string

Syntax : trim(‘str’)

Example : SELECT TRIM(‘   karu   ‘) FROM DUAL;

Output:karu

————————————————————————————————-

Function 8: **Ltrim**
Purpose : Trim the whitespaces in left the side of the string

Syntax : Ltrim(‘str’)

Example : SELECT LTRIM(‘   karu   ‘) FROM DUAL;

Output:karu….(. dot are spaces)

————————————————————————————————-

Function 9: **Rtrim**

Purpose : Trim the whitespaces in right the side of the string

Syntax : Rtrim('str')

Example : SELECT RTRIM(' karu ') FROM DUAL;

Output:....karu(. dot are spaces)

——————————————————————————————————-

Function 10: **Length**
Purpose : length of the string

Syntax : length('str')

Example : SELECT LENGTH('karuvachi') FROM DUAL;

Output:9

——————————————————————————————————-

Function 11: **Instr**
Purpose : Find the position of the string in another string

Syntax : Instr('str1','str2')

Example : SELECT INSTR('karuvachi','ka') FROM DUAL;

Output:1

——————————————————————————————————-

Function 12: **substr**
Purpose : get a sub string from string

Syntax : substr('str',start_pos,number_of_chars)

Example : SELECT substr('karuvachi',2,4) FROM DUAL;

Output: aruv

<u>Date Functions and Operators.</u>

To see the system date and time use the following functions :

CURRENT_DATE :returns the current date in the session time zone, in a value in the Gregorian calendar of datatype
DATE
SYSDATE :Returns the current date and time.
SYSTIMESTAMP :The SYSTIMESTAMP function returns the system date, including fractional

seconds and time zone

of the database. The return type is TIMESTAMP WITH TIME ZONE.

FORMAT	MEANING
D	Day of the week
DD	Day of the month
DDD	Day of the year
DAY	Full day for ex. ‘Monday’, ’Tuesday’, ’Wednesday’
DY	Day in three letters for ex. ‘MON’, ‘TUE’,’FRI’
W	Week of the month
WW	Week of the year
MM	Month in two digits (1-Jan, 2-Feb,…12-Dec)
MON	Month in three characters like “Jan”, ”Feb”, ”Apr”
MONTH	Full Month like “January”, ”February”, ”April”
RM	Month in Roman Characters (I-XII, I-Jan, II-Feb,…XII-Dec)
Q	Quarter of the Month
YY	Last two digits of the year.
YYYY	Full year
YEAR	Year in words like “Nineteen Ninety Nine”
HH	Hours in 12 hour format
HH12	Hours in 12 hour format
HH24	Hours in 24 hour format
MI	Minutes
SS	Seconds
FF	Fractional Seconds
SSSSS	Milliseconds
J	Julian Day i.e Days since 1st-Jan-4712BC to till-date
RR	If the year is less than 50 Assumes the year as 21ST Century. If the year is greater than 50 then assumes the year in 20th Century.

Date and time functions and formats are quite different in various databases. In this article, let’s review the most common functions that manipulates dates in an Oracle database.

The function SYSDATE() returns a 7 byte binary data element whose bytes represents:

- century,
- year,
- month,
- day,
- hour,
- minute,
- second

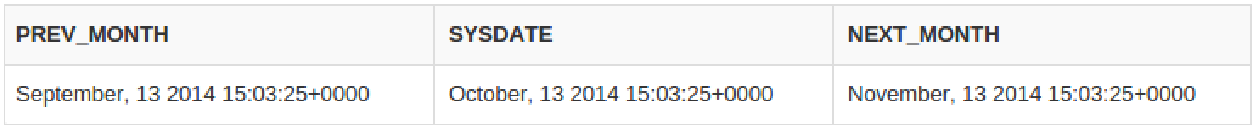
Select sysdate from dual;

Oracle enables you to extract the **day**, **month**, and **year** from a date using an **extract** function:

```
select extract(day from sysdate) as only_day from dual
select extract(month from sysdate) as only_month from dual
select extract(year from sysdate) as only_year from dual
```

ADD_MONTHS(date, n) – Adds the specific number of months (n) to a date. The ‘n’ can be both negative and positive:

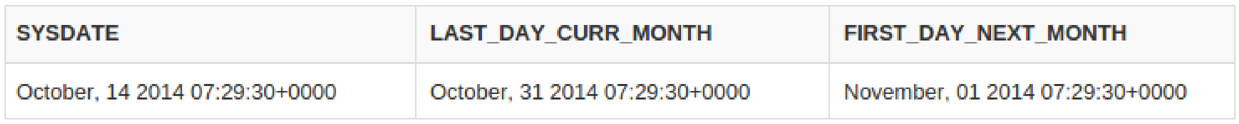
```
Select add_months(sysdate, -1) as prev_month , sysdate, add_months (sysdate, 1) as next_month
from dual
```



PREV_MONTH	SYSDATE	NEXT_MONTH
September, 13 2014 15:03:25+0000	October, 13 2014 15:03:25+0000	November, 13 2014 15:03:25+0000

LAST_DAY(date) – Returns the last day in the month of the specified date d.

```
select sysdate, last_day(sysdate) as last_day_curr_month,
last_day(sysdate) + 1 as first_day_next_month from dual
```



SYSDATE	LAST_DAY_CURR_MONTH	FIRST_DAY_NEXT_MONTH
October, 14 2014 07:29:30+0000	October, 31 2014 07:29:30+0000	November, 01 2014 07:29:30+0000

The number of days until the end of the month.

```
select last_day(sysdate) - sysdate as days_left
from dual
```

MONTHS_BETWEEN(date, date) – Calculates the number of months between two dates.

Example:

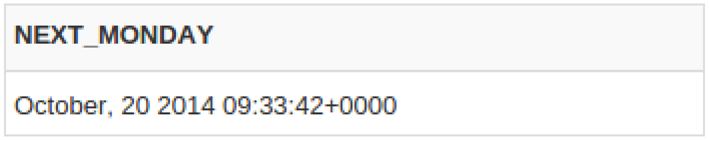
```
select MONTHS_BETWEEN ('31-JAN-2014', '28-FEB-2014')
from dual

select MONTHS_BETWEEN ('31-MAR-2013', '28-FEB-2013')
from dual
```

Let's select the number of months an employee has worked for the company.

```
Select months_between (sysdate, date_of_hire)
from employees
```

NEXT_DAY(date, day_of_week) – Returns the date of the first weekday specified that is later than the date.

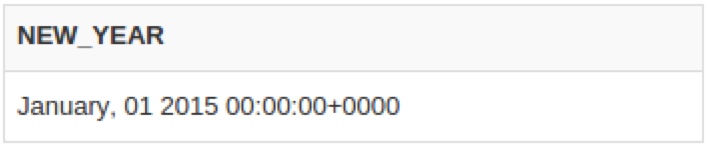
```
select next_day(sysdate, 'monday') as next_Monday from dual
```



NEXT_MONDAY
October, 20 2014 09:33:42+0000

ROUND(date [, format_mask VARCHAR2]) – Returns the date with time rounded to midnight (12 A.M.) in the default. The format mask is optional. The following example rounds a date to the first day of the following year:

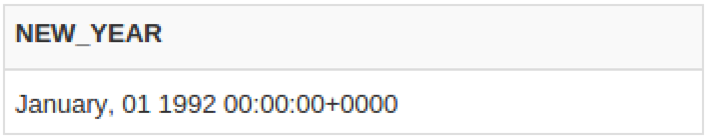
```
SELECT ROUND (TO_DATE ('10-SEP-14'),'YEAR') as new_year
FROM DUAL;
```



NEW_YEAR
January, 01 2015 00:00:00+0000

TRUNC(date, [format]) – Truncates the specified date of its time portion according to the format provided. If the 'format' is omitted, the hours, minutes or seconds will be truncated.

```
SELECT TRUNC(TO_DATE('27-OCT-92'), 'year')
as new_year FROM DUAL;
```



NEW_YEAR
January, 01 1992 00:00:00+0000

Arithmetic Operations With Dates

- **Date + number**

```
select sysdate + 1 as tomorrow
from dual

select sysdate + (5/1440) as five_mintues_from_now
from dual
```

- **Date – number**

```
select sysdate - 1 as yesterday
from dual
```

- **Date – date**

  You can subtract a date from a date in Oracle. The result will be in days. You can also multiply by 24 to get hours and so on.

select 24 * (to_date('2014-10-10 22:00', 'YYYY-MM-DD hh24:mi') - to_date('2014-10- 9 21:00', 'YYYY-MM-DD hh24:mi'))

difference_in_hours from dual;

Besides the SQL utility functions, Oracle inbuilt function library contains type conversion functions. There may be scenarios where the query expects input in a specific data type, but it receives it in a different data type. In such cases, Oracle implicitly tries to convert the unexpected value to a compatible data type which can be substituted in place and application continuity is not compromised. Type conversion can be either implicitly done by Oracle or explicitly done by the programmer.

Implicit data type conversion works based on a matrix which showcases the Oracle's support for internal type casting. Besides these rules, Oracle offers type conversion functions which can be used in the queries for explicit conversion and formatting. As a matter of fact, it is recommended to perform explicit conversion instead of relying on software intelligence. Though implicit conversion works well, but to eliminate the skew chances where bad inputs could be difficult to typecast internally.

Implicit Data Type Conversion

A VARCHAR2 or CHAR value can be implicitly converted to NUMBER or DATE type value by Oracle. Similarly, a NUMBER or DATA type value can be automatically converted to character data by Oracle server. Note that the impicit interconversion happens only when the character represents the a valid number or date type value respectively.

For example, examine the below SELECT queries. Both the queries will give the same result because Oracle internally treats 15000 and '15000' as same.

Query-1

```
SELECT employee_id,first_name,salary

FROM employees

WHERE salary > 15000;
```

Query-2

```
SELECT employee_id,first_name,salary

FROM employees

WHERE salary > '15000';
```

Explicit Data Type Conversion

SQL Conversion functions are single row functions which are capable of typecasting column value, literal or an expression . TO_CHAR, TO_NUMBER and TO_DATE are the three functions which perform cross modification of data types.

TO_CHAR function

TO_CHAR function is used to typecast a numeric or date input to character type with a format model (optional).

Syntax

```
TO_CHAR(number1, [format], [nls_parameter])
```

For number to character conversion, nls parameters can be used to specify decimal characters, group separator, local currency model, or international currency model. It is an optional specification - if not available, session level nls settings will be used. For date to character conversion, the nls parameter can be used to specify the day and month names, as applicable.

Dates can be formatted in multiple formats after converting to character types using TO_CHAR function. The TO_CHAR function is used to have Oracle 11g display dates in a particular format. Format models are case sensitive and must be enclosed within single quotes.

Consider the below SELECT query. The query format the HIRE_DATE and SALARY columns of EMPLOYEES table using TO_CHAR function.

```
SELECT first_name,
    TO_CHAR (hire_date, 'MONTH DD, YYYY') HIRE_DATE,
        TO_CHAR (salary, '$99999.99') Salary
FROM employees
WHERE rownum < 5;

FIRST_NAME          HIRE_DATE          SALARY
-------------------- ------------------ ----------
Steven               JUNE      17, 2003  $24000.00
Neena                SEPTEMBER 21, 2005  $17000.00
Lex                  JANUARY   13, 2001  $17000.00
Alexander            JANUARY   03, 2006   $9000.00
```

The first TO_CHAR is used to convert the hire date to the date format MONTH DD, YYYY i.e. month spelled out and padded with spaces, followed by the two-digit day of the month, and then the four-digit year. If you prefer displaying the month name in mixed case (that is, "December"), simply use this case in the format argument: ('Month DD, YYYY').

The second TO_CHAR function in Figure 10-39 is used to format the SALARY to display the currency sign and two decimal positions.

TO_NUMBER function

The TO_NUMBER function converts a character value to a numeric datatype. If the string being converted contains nonnumeric characters, the function returns an error.

Syntax

```
TO_NUMBER (string1, [format], [nls_parameter])
```

TO_DATE function

The function takes character values as input and returns formatted date equivalent of the same. The TO_DATE function allows users to enter a date in any format, and then it converts the entry into the default format used by Oracle 11g.

Syntax:

```
TO_DATE( string1, [ format_mask ], [ nls_language ] )
```