SINGLE-ROW NUMERIC FUNCTIONS

Single-row numeric functions operate on numeric data and perform some kind of mathematical or arithmetic manipulation. All have numeric arguments and returns numeric values.

ABS(<*n*>)

Where *n* is a number. This function returns the absolute of n.

SQL>*SELECT ABS(-52) negative, ABS(52) positive FROM dual;*

OUTPUT:

NEGATIVE	POSITIVE
52	52

CEIL(<*n*>)

Where *n* is a number. This function returns the smallest integer that is greater than or equal to *n*. CEIL rounds up to a whole number. See also FLOOR.

SQL>*SELECT CEIL (9.8), CEIL(-32.85), CEIL(0) FROM dual;*

OUTPUT:

CEIL(9.8)	CEIL(-32.85)	CEIL (0)
10	-32	0

COS(<n>**)**

It returns trigonometric cosine of the number n.

SQL> *SELECT COS(45) FROM DUAL;*

OUTPUT:

COS(45)
.52532199

EXP(<*n*>)

Where *n* is a number. This function returns *e* (the base of natural logarithms) raised to the n^{th} power.

SQL>*SELECT EXP(1) "e" FROM dual*;

OUTPUT:

e
2.71828183

FLOOR(<*n*>)

Where *n* is a number. This function returns the largest integer that is less than or equal to *n*. FLOOR round down to a whole number. See also CEIL.

SQL>*SELECT FLOOR(9.8), FLOOR(-32.85), FLOOR(137) FROM dual;*

OUTPUT:

FLOOR(9.8)	FLOOR(-32.85)	FLOOR(137)
9	-33`	137

LOG(<*n1*>, <*n2*>)

Where *n1* and *n2* are numbers. This function returns the logarithm base *n1* of *n2*.

SQL>*SELECT LOG(8,64), LOG(3,27), LOG(2,1024) FROM dual;*

OUTPUT:

LOG(8,64)	LOG(3,27)	LOG(2,1024)
2	3	10

MOD(<*n1*>, <*n2*>)

Where *n1* and *n2* are numbers. This function returns *n1* modulo *n2* or the remainder of *n1* divided by *n2*. If *n1* is negative, the result is negative. The sign of *n2* has no effect on the result. This behavior differs from the mathematical definition of the modulus operation.

SQL>*SELECT MOD(14,5), MOD(8,2.5), MOD(-64,7) FROM dual;*

OUTPUT:

MOD(14,5)	MOD(8,2.5)	MOD(-64,7)
4	.5	-1

POWER(<*n1*>, <*n2*>)

Where *n1* and *n2* are numbers. This function returns *n1* to the $n2^{th}$ power.

SQL>*SELECT POWER(2, 10), POWER(3,3), POWER(5,3) FROM dual;*

OUTPUT:

POWER(2,10)	POWER(3,3)	POWER(5,3)
1024	27	125

ROUND(<*n1*>, <*n2*>)

Where *n1* and *n2* are numbers. This function returns *n1* rounded to *n2* digits of precision to the right of the decimal. If *n2* is negative, *n1* is rounded to left of the decimal. This function is similar to TRUNC( ).

SQL>*SELECT ROUND(12345,-2), ROUND(12345.54321,2) from dual;*

OUTPUT:

ROUND(12345,-2)	ROUND(12345.54321,2)
12300	12345.54

SIGN(<*n*>)

Where *n* is a number. This function returns -1 if *n* is negative, 1 if *n* is positive, and 0 if *n* is 0.

SQL>*SELECT SIGN(-2.3), SIGN(0), SIGN(47) FROM dual;*

OUTPUT:

SIGN(-2.3)	SIGN(0)	SIGN(47)
-1	0	1

SQRT (<*n*>)

Where *n* is a number. This function returns the square root of *n*.

SQL>*SELECT SQRT(64), SQRT(49), SQRT(5) FROM dual;*

OUTPUT:

SQRT(64)	SQRT(49)	SQRT(5)
8	7	2.23606798

TRUNC (<*n*>)

Where *n1* is a number and *n2* is an integer. This function returns *n1* truncated to *n2* digits of precision to the right of the decimal. If *n2* is negative, *n1* is truncated to left of the decimal. See also ROUND.

SQL>*SELECT TRUNC(123.456,2) pos, TRUNC(123.456,-1) neg FROM dual;*

OUTPUT:

POS	NEG
123.45	120

Comparison of ROUND and TRUC Functions

The comparison of Round and Trunc functions has been further illustrated below:

Example	OUTPUT OF ROUND	TRUNC	Output of TRUNC
ROUND(6876.678, -1)	6880	TRUNC(6876.678, -1)	6870
ROUND(6876.678, -2)	6900	TRUNC(6876.678, -2)	6800
ROUND(6876.678, -3)	7000	TRUNC(6876.678, -3)	6000
ROUND(6876.678, -4)	10000	TRUNC(6876.678, -4)	0
ROUND(6876.678, -5)	1	TRUNC(6876.678, -5)	0
ROUND(6876.678, -6)	0	TRUNC(6876.678, -6)	0

Numeric Function Summary

Function	Description
ABS	Returns the absolute value
CEIL	Returns the next higher integer
COS	Returns the cosine
EXP	Returns the base of natural logarithms raised to a power
FLOOR	Returns the next smaller integer
LN	Returns the natural logarithm
LOG	Returns the logarithm
MOD	Returns modulo (remainder) of a division operation
POWER	Returns a number raised to an arbitrary power
ROUND	Rounds a number
SIGN	Returns an indicator of sign: negative, positive, or zero
SIN	Returns the sine
SQRT	Returns the square root of a number
TRUNC	Truncates a number

Single-Row Date Functions

Single-row date functions operate on date data type.

ADD_MONTHS(<*d*>, <*i*>)

Where *d* is a date and *i* is an integer. This function returns the data *d* plus *i* months. If *i* is a decimal number, the database will implicitly convert it to an integer by truncating the decimal portion (for example, 3.9 becomes 3).

SQL> *SELECT SYSDATE, ADD_MONTHS(SYSDATE,3)plus_3,* ADD_MONTHS(SYSDATE,-2) minus_2 FROM DUAL;

OUTPUT:

SYSDATE	PLUS_3	MINUS_2
01-JAN-98	01-APR-98	01-NOV-97

LAST_DAY(<*d*>)

Where *d* is a date. This function returns the last day of the month for the date *d*.

SQL>*SELECT SYSDATE, LAST_DAY(SYSDATE)+1FROM dual;*

OUTPUT:

SYSDATE	LAST_DAY(SY
23-NOV-1999	01-DEC-1999

MONTHS_BETWEEN(<*d1*>, <*d2*>)

Where *d1* and *d2* are both dates. This function returns the number of months that *d2* is later then *d1*. A whole number is returned if *d1* and *d2* are the same day of the month or if both dates are the last day of a month.

SQL>*SELECT MONTHS_BETWEEN ('19-Dec-1999', '19-Mar-2000') FROM dual;*

OUTPUT:

MONTHS_BETWEEN('19-DEC-1999', '19-MAR-2000')
3

NEXT_DAY(<*d*>, <*dow*>)

Where *d* is a date and *dow* is a text string containing the full or abbreviated day of the week in the session's language. This function returns the next *dow* following *d.* The time portion of the return date is the same as the time portion of *d.*

SQL> *SELECT NEXT_DAY('01-Jan-2004','Monday') "1st Monday" FROM dual;*

OUTPUT:

1st Monda
05-JAN-04

ROUND(<*d*>[, <*fmt*>])

Where *d* is a date and *fmt* is a character string containing a date-format string.

SQL> *SELECT SYSDATE, ROUND(SYSDATE, 'MM') FROM dual;*

OUTPUT:

SYSDATE	ROUND(SYS
16-JAN-98	01-FEB-98

SYSDATE

This function takes no arguments and returns the current date and time to the second level.

SQL>*SELECT SYSDATE FROM dual;*

OUTPUT:

SYSDATE
24-Nov-1999 09:26:01

Arithmetic with Dates

Important points regarding arithmetic operations on dates are given below:

- Add or subtract a number to or from a date for a resultant date value.
- Subtract two dates to find the number of days between those dates.
- Add hours to a date by dividing the number of hours by 24.

Examples:

SQL> *SELECT SYSDATE+2 FROM DUAL;*

OUTPUT:

SYSDATE+2
09-JUN-16

It shows the date after two days.

SQL> *SELECT ROLLNUMBER,SYSDATE-DATEOFBIRTH FROM STUDENT;*

OUTPUT:

ROLLNUMBER	SYSDATE - DATEOFBIRTH
1	9674.917997685185185185185185185185185 19
2	9520.917997685185185185185185185185185 19
3	9288.917997685185185185185185185185185 19
4	8761.917997685185185185185185185185185 19
5	9454.917997685185185185185185185185185 19

It provides the age in number of days, it is important to note that it provides number of days up to points depending upon the time of day.

SQL> *SELECT ROLLNUMBER,DATEOFBIRTH, SYSDATE,(SYSDATE-DATEOFBIRTH)/365 FROM STUDENT;*

OUTPUT:

ROLLNUMBER	DATEOFBIRTH	SYSDATE	(SYSDATE - DATEOF BIRTH)/365
1	12-DEC-89	07-JUN-16	26.506630580923389142567224759005580 9234
2	15-MAY-90	07-JUN-16	26.084712772704211060375443937087772 7042
3	02-JAN-91	07-JUN-16	25.449096334348046676813800101471334 3481
4	12-JUN-92	07-JUN-16	24.005260717909690512430238457635717 9097
5	20-JUL-90	07-JUN-16	25.903890854895991882293252156265854 896

It provides the age in years.

Date Function Summary

Function	Description
ADD_MONTHS	Adds a number of months to a date
LAST_DAY	Returns the last day of a month
MONTHS_BETWEEN	Returns the number of months between two dates
NEXT_DAY	Returns the next day of a week following a given date
ROUND	Rounds a date/time
SYSDATE	Returns the current date/time

Single-Row Conversion Functions

Single-row conversion functions operate on multiple data types. In Oracle, we have To_NUMBER, TO_DATE and TO_CHAR as inbuilt functions for explicit data types conversion as shown in figure 5.4.

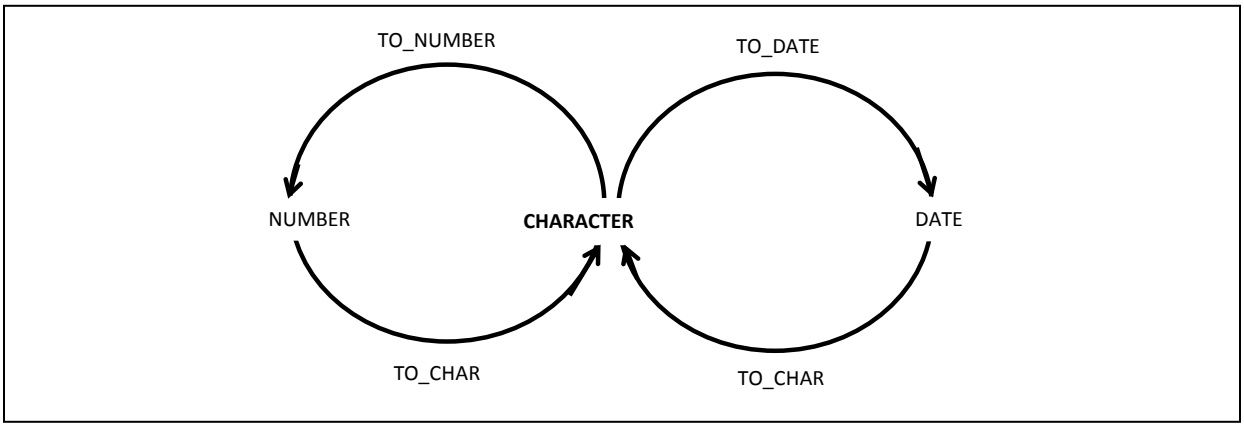

Figure 24.4: Explicit Data Type Conversion

To_char(number|date,fmt)

This function converts a number or date value to a varchar2 character string with format model fmt . It facilitates the retrieval of data in a format different from the default format (DD-MON-YY) when using for dates. With the help of this function part of the date i.e the date, month or year can also be extracted. While using this function to convert dates following guidelines must be followed:

- the format model must be enclosed in single quotation marks and is case sensitive
- the format model can include any valid date format element be sure to separate the date value from the format model by the comma
- the names of days and months in the output are automatically padded with blanks
- you can resize the display width of the resulting character field with the SQL * Plus column command.

SQL>*Select sysdate, to_char(sysdate ,'DAY') from dual;*

OUTPUT:

SYSDATE	TO_CHAR(S
07-JULY-03	MONDAY

Example: To display current time in three different columns in form of hour, minutes and second.

SQL> *SELECT TO_CHAR(SYSDATE,'HH') HOUR, TO_CHAR(SYSDATE,'MI') MIN,TO_CHAR(SYSDATE,'SS') SEC FROM DUAL;*

OUTPUT:

HO	MI	SE
--	--	--
03	01	16

Example: To display current date and time.-

SQL>*SELECT TO_CHAR(SYSDATE, 'DD-MON-YYYY HH24:MI:SS') CURRENT_DATE_TIME FROM DUAL;*

OUTPUT:

CURRENT_DATE_TIME
07-JUN-16 22:08:33

Example: To display current day.

SQL> *SELECT TO_CHAR(SYSDATE, 'DAY') CURRENT_DAY FROM DUAL;*

OUTPUT:

CURRENT_DAY
TUESDAY

Following table shows all the available options for date format.

Elements of the Date Format Model

YYYY	Full year in numbers
YEAR	Year spelled out
MM	Two-digit value for month
MONTH	Full name for month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month

SQL>*Select to_char(sal ,'$99,999') salary from emp where ename ='SCOTT' ;*

OUTPUT:

SALARY
$3,000

To_date(char[,'fmt'])

This function converts a character value into a date value where *char* stand for the value to be inserted in the date column and *fmt* is the date format in which *char* is specified.

SQL>*Select to_date('07-july-03','RM') from dual ;*

OUTPUT:

to_date('07-july-03'),'RM'
7

To_number(text|date)

This function which converts text or date information into a number.

SQL>*SELECT TO_NUMBER('49583') FROM DUAL;*

OUTPUT:

TO_NUMBER('49583')
49583

YY and RR Date Format

There are two dates format, i.e, RR and YY format. These formats play vital roles when we specify only digits of year. In case of YY format when we specify two digits of year, the other digits (20) are automatically assigned to the current century, i.e, 99 will be considered ir stored as 2099.

RR converts two-digit years into four-digit years by rounding. It means, 50-99 are stored as 1950-1999, and dates ending in 00-49 are stored as 2000-2049. RRRR accepts a four-digit

input (although not required), and converts two-digit dates as RR does. YYYY accepts 4-digit inputs but doesn't do any date converting

Examples:

```
SQL> SELECT ENAME FROM EMP WHERE HIREDATE = TODATE('01/05/81','DD/MM/YY');
```

```
SQL>SELECT ENAME FROM EMP WHERE HIREDATE = TODATE('01/05/81','DD/MM/RR');
```

Here, in first query it will search records for hiredate 2081 while in second query it will search for hiredate 1981. Obviously, first query will not return any records while second query will work.

The further illustration of YY and RR date format has been given below:

RR Date Format

Current Year	Specified Date	RR Format	YY Format
1976	25-FEB-85	1985	1995
1976	27-OCT-16	2016	1916
2016	27-OCT-16	2016	2016
2016	27-OCT-95	1995	2095

		If the specified two-digit year is:	
		0 - 49	50 - 99
If two digits of the current year are:	0 - 49	The return date is in the current century	The return date is in the century before the current one
	50 - 99	The return date is in the century after the current one	The return date is in the current century