Chapter 7

STRING, CONVERSION, AND MISCELLANEOUS FUNCTIONS

- String functions
- Conversion functions
- Miscellaneous functions

In the last chapter we have seen how to use arithmetic and date functions. In this chapter let us see how to use string, conversion and miscellaneous functions.

String Functions

String functions are functions that manipulate a set of characters. A set of characters is a string. For example, the name of the company, the address of a person all these are examples of a string. CHAR and VARCHAR data types contain strings. Let us first see how to concatenate strings in Oracle.

Concatenating Strings

Two strings can be concatenated (added one after another) to form a single string using the string concatenation operator, which is **||** (two pipe symbols).

The following example concatenates name of the faculty with qualification. We also put two spaces between these two values to provide required space.

```
select name || '  ' ||  qual from faculty

NAME||''||QUAL
--------------------------------------------------------
George Koch  MS Computer Science
Dan Appleman  CS and EE graduate
Herbert Schildt  MS Computer Science
David Hunter  MS Electronics
Stephen Walther  Ph.D. in Philosophy
Kevin Loney  MS Eletronics
Jamie Jaworski  Bachlors of Electrical
Jason Couchman  OCP DBA
```

Table 1 shows the list of string functions. These functions generally take a string as parameter and also return a string as return value.

Function	Description
LENGTH (string)	Returns the number of characters in the *string*.
LOWER (string)	Returns the string after converting the string to lowercase.
UPPER (string)	Returns the string after converting the string to uppercase.
INITCAP (string)	Converts first character of every word to uppercase and remaining to lower case.
LPAD (string, length [, fillstring])	Makes the *string* of the given *length* by padding the string on the left either with space or with *fillstring*.
RPAD (string, length [, fillstring])	Same as LPAD but pads on the right.
LTRIM (string [, charset])	Removes all left most characters of string up to the first character that is not in the *charset*. if *charset* is not given then it defaults to blank.
RTRIM (string [, charset])	Same as LTRIM, but trims on the right.
TRIM (string)	Trims space on both sides.
SUBSTR (string, pos , length)	Extracts *length* number of characters from position *pos* in the string. If *length* is not given then extracts everything from *pos*.
INSTR (s1,s2 [,pos [,occurrence]])	Finds the starting position of *s2* in *s1*. If *occurrence* is not given then it finds first occurrence. Search starts at *pos*, if given, otherwise at the first character in *s1*.
ASCII (string)	Returns ASCII code of the first character in the given string
CHR (number)	Returns ASCII character for the given ASCII code.
TRANSLATE (string, from, to)	Replaces characters in *from* with *to* in string.
REPLACE (string, source, replace)	Replaces *source* in *string* with *replace*.

Table 1: String functions.

Converting Case

Functions LOWER and UPPER are straightforward. And they play a very important role in string comparison. As string comparison is case sensitive, LOWER or UPPER can be used to convert strings to uniform case before they are compared.

The following query tries to retrieve details of courses related to programming.

```
select name,duration from courses
where  name like '%programming%'

NAME                           DURATION
------------------------------ ---------
C programming                         20
```

The above query retrieves only one two whereas there are two rows that contain the word programming. It is because of the difference in the case. So the following query is converting the name to lowercase before comparison.

```
select name,duration from courses
where  LOWER(name) like '%programming%'

NAME                           DURATION
------------------------------ ---------
C programming                         20
XML Programming                       15
```

As NAME is converted to lowercase during comparison and compared with programming, which is in lowercase, the difference in case is ignored. The same result can be achieved even by using UPPER function. But in that case the string must be given in uppercase – PROGRAMMING.

INITCAP converts first letter of each word to capital and remaining letters to lowercase.

```
select initcap('this IS to Test INITCAP') Result
from  dual;

RESULT
-----------------------
This Is To Test Initcap
```

INSTR function

INSTR returns the position in the first string where the second string starts in the first string. If second string is not found in first string, it returns **0**.

The default is to return the position of first occurrence by starting the search at the very first character in first string. However, INSTR has options using which we can specify from where the search should start and which occurrence is to be considered.

The following examples illustrate the usage of two optional parameters; *start* and *occurrence***.**

```
select instr('How do you do','do') Postion
from dual;

  POSTION
---------
        5
```

Though string "do" occurs for twice, the position of first occurrence is be returned. It is possible to specify to INSTR that it should start looking for *do* starting from the given position as follows.

```
select instr('How do you do','do',8) Postion
from dual

  POSTION
---------
       12
```

It is possible to specify that the position of the specified occurrence is to be returned as follows:

select instr('How do you do','do',1,2) Postion
from dual

```
  POSTION
---------
       12
```

Note: *When occurrence is specified then starting position must also be specified, as third parameter cannot be omitted while fourth parameter is given.*

The following example displays the details of courses where the letter p exists in the name of the course after 6th position.

```
select ccode,name
from courses
where  instr(name,'n') > 6
```

```
CCODE NAME
----- ----------------------------
c     C programming
java  Java Language
xml   XML Programming
```

The same query can also be written as follows using LIKE operator and six underscores to indicate that first six letters may be anything but ***n*** must exists after that. But INSTR version will be more flexible.

```
select ccode,name
from courses
where  name like '______%n%'

CCODE NAME
----- ------------------------------
c     C programming
java  Java Language
xml   XML Programming
```

We will se some more applications of INSTR at a later stage.

SUBSTR function

SUBSTR is used to extract a sub string from the given string. It takes the position from where extraction starts and the number of characters to be extracted.

The following example displays the first 2 character from the code of the course.

```
select  ccode, substr(ccode,1,2) sn from courses

CCODE SN
----- --
ora   or
vbnet vb
c     c
asp   as
java  ja
xml   xm
```

It is possible to omit third parameter – length of the sub string. The following example illustrates it.

```
select  substr('Srikanth Technologies',10) Result from dual

RESULT
------------
Technologies
```

The result of one function can be passed to another function as input. We have already seen nesting functions in the previous chapter. Now let us see how we can combine SUBSTR and INSTR to get the first name of each faculty. First name is the name before space.

```
select  name, substr( name, 1, instr(name,' ') - 1) Firstname from faculty;

NAME                           FIRSTNAME
------------------------------ ------------------------------
George Koch                    George
Dan Appleman                   Dan
Herbert Schildt                Herbert
David Hunter                   David
Stephen Walther                Stephen
Kevin Loney                    Kevin
Jamie Jaworski                 Jamie
Jason Couchman                 Jason
```

INSTR function is used to find out the position of first space. Then that position is used to specify the number of character to be taken from name. We subtracted one from position because position indicates the position of space but we have to take up to the character before the space.

The following is another example of these two functions. It take last name of the faculty.

```
select  name, substr(name,instr(name,' ') + 1) Lastname from faculty

NAME                           LASTNAME
------------------------------ ------------------------------
George Koch                    Koch
Dan Appleman                   Appleman
Herbert Schildt                Schildt
David Hunter                   Hunter
Stephen Walther                Walther
Kevin Loney                    Loney
Jamie Jaworski                 Jaworski
Jason Couchman                 Couchman
```

The following is the sequence of steps in the above query.

- INSTR returns the position of first space in NAME
- The return value of the INSTR, after 1 is added, is passed to SUBSTR as the starting position.
- Everything on the right of the given position is taken by SUBSTR

Since Oracle converts the given values to required data type automatically, INSTR and SUBSTR can also be used with numbers and dates. The following query displays payments made in the month of April.

```
select * from payments
where  instr(dp,'APR') <> 0

   ROLLNO DP            AMOUNT
--------- --------- ---------
        9 07-APR-01      3000
       10 10-APR-01      4500
       11 07-APR-01      1000
       11 10-APR-01      3500
```

Trimming Strings

LTRIM and RTRIM are used to trim off unwanted characters from the left and right ends of the string respectively. Leftmost spaces are called as ***leading spaces*** and rightmost spaces are called as ***trailing spaces.***

They trim spaces by default. Optionally, you can specify which set of characters you want to trim.

The following example is used to trim spaces on the left using LEFT. The length of the string will show the result.

```
select length('  abc  xyz   ') Before,
       length(  ltrim('  abc  xyz   ')) After
from dual

   BEFORE     AFTER
--------- ---------
       13        11
```

You can also trim a specified set of characters as shown below.

```
select ltrim( 'aabcbadxyabc','abc') Result from dual

RESULT
------
dxyabc
```

In the above example, trimming stopped at 'd' because 'd ' is the first character that doesn't fall in the character set of 'abc'.

While trimming, each character from left or right is taken and checked against the characters in the set. If character is same as any character in the character set, then character is trimmed otherwise trimming ends at that character. The same is true with RTRIM function, but it trims on the right.

TRIM function, which was introduced in Oracle8i, is used to trim both leading and trailing spaces.

```
select length('  abc  xyz  ') Before,
       length(trim('  abc  xyz  ')) After
from dual

   BEFORE     AFTER
--------- ---------
       12         8
```

Padding Strings

A string can be made of a given length by padding either on the left using LPAD or on the right using RPAD. By default Oracle uses space to pad strings. However, it is possible to specify which character(s) should be used for padding.

The following example course to 12 characters it specifies that dot is to be used for padding.

```
select rpad(name,12,'.') Name from courses
```

```
NAME
------------
Oracle datab
VB.NET......
C programmin
ASP.NET.....
Java Languag
XML Programm
```

The above example is padding strings that are shorter than 12 characters and truncating strings that are larger than 12 characters. Names like *VB.NET* and *ASP.NET* are padded on the right using dots. Whereas names like *Oracle database* and *C programming* are truncated to 12 characters.

***Note**: RPAD and LAPD truncate the given string if string has more number of characters than the given length.*

TRANSLATE and REPLACE functions

These two functions return the string after modifying the given string. TRANSLATE works on individual characters, whereas REPLACE replaces a string with another string.

The following two examples will make the difference clear.

```
select replace('ABC ABAC XYZ DABC','ABC','PQR') Result from dual

RESULT
-----------------
PQR ABAC XYZ DPQR
```

REPLACE replaces every occurrence of string 'ABC' with string 'PRQ'.

```
select translate('ABC ABAC XYZ DABC','ABC','PQR') Result
from dual

RESULT
-----------------
PQR PQPR XYZ DPQR
```

TRANSLATE changes every occurrence of letter A with P, B with Q and C with R.

Conversion Functions

Conversion functions are used to convert a value from one data type into another. These functions are not required if Oracle can automatically convert the value. But there are cases where these conversion functions are required to convert the value to the required data type. The following table lists conversion functions.

FUNCTION	DESCRIPTION
TO_CHAR (value [, format])	Converts *value,* which is of DATE or NUMBER type, to CHAR type.
TO_DATE (char [, format])	Converts the given CHAR type value to DATE type.
TO_NUMBER (char)	Converts given CHAR type value to NUMBER type.

Table 2: Conversion Functions.

Before we understand how and where we use conversion functions, let us see how Oracle tries to convert the given data to the required data type.

Automatic Type Conversion

Oracle automatically converts the value to the required data type if it is possible. For example, if a number is used with string function, number is converted to string and then the function is executed.

In the same way, if a DATE type value is required but if a CHAR type value is given in the format DD-MON-YY or DD-MON-YYYY then Oracle converts it to DATE type.

But this automatic data type conversion is not always possible. To convert the value to the required data type, the given value must already look like the data type it is being converted to. The following are guidelines that describe automatic type conversion.

GUIDELINES FOR AUTOMATIC CONVERSION OF DATA TYPE

- ❑ Any NUMBER or DATE will be converted to a CHAR.
- ❑ If DATE is a literal enclose it in quotes.
- ❑ CHAR type will be converted to NUMBER if it contains only digits, decimal point, or minus sign on the left.
- ❑ CHAR will be converted to DATE type if it is in DD-MON-YY or DD-MON-YYYY format.
- ❑ A DATE will NOT be converted to NUMBER.
- ❑ A NUMBER will NOT be converted to DATE.

The following few examples will give you better idea about automatic conversion of data type:

In the following example NUMBER is automatically converted to CHAR before LENGTH function is used.

```
select length(1133) from dual;

LENGTH(1133)
------------
           4
```

In the example below, a DATE given in CHAR format is converted to DATE before LAST_DAY function is applied.

```
select  last_day('20-aug-2001') from dual;

LAST_DAY(
---------
31-AUG-01
```

Similarly it is possible to use a CHAR value where a NUMBER is required, as shown below.

```
select 5 * '20' from dual;

   5*'20'
---------
      100
```

Here are a few examples where Oracle cannot automatically convert the value.

```
SQL> select  next_day('12-1-2001', 'Fri') from dual;
select  next_day('12-1-2001', 'Fri') from dual
                 *
ERROR at line 1:
ORA-01843: not a valid month
```

Oracle returns an error saying the date is not having valid month because Oracle expects months to be of first three letters of the month name. As we have given only month number, it is not acceptable to Oracle. In this case we need to explicitly convert the value to DATE type using TO_DATE function.

The following sections will show how to use conversion functions.

TO_CHAR Function

This function is used to convert the given DATE or NUMBER to CHAR type. TO_CHAR function may also be used to format the given date or number while converting the value to CHAR type. For example, to display date in DD-MM-YYYY format instead of standard format - DD-MON-YY, enter the following:

```
select to_char(sysdate,'dd-mm-yyyy') Result from dual

RESULT
----------
15-08-2000
```

In fact, TO_CHAR is one of the most frequently used functions. Here in the example, below it is used to display both date and time of SYSDATE. Remember this operation needs explicit usage of TO_CHAR as by default Oracle displays only date.

```
select to_char(sysdate,'dd Month yyyy hh24:mi:ss')
from  dual

TO_CHAR(SYSDATE,'DDMONTHYY
--------------------------
15 August    2000 02:18:56
```

In the above example **Month** is standing for complete month name, **yyyy** stands for four digits year, **hh24** for 24 hours based hour, **mi** minutes and **ss** for seconds.

Format in TO_CHAR function is a collection of more than 40 formatting options. Please see Table 3 for more options. For complete list, please see on-line help for *Date Format Elements*.

All options in the format are replaced with the corresponding values and remaining characters are returned as they are. In the above example, ':' between HH24 and MI is returned as it is but HH24 and MI are replaced with the corresponding values.

Format Option	**Description**
MM	Number of the month: 10
MON	First three letters of month name: OCT
MONTH	Complete month name: OCTOBER
DDD	Day of the year since January 1st: 340
DD	Day of the month: 16
D	Day of the week: 5
Day	Day fully spelled: Wednesday
YYYY	Four digits year: 1996
YY	Two digits year: 96
YEAR	Year spelled out: NINTEEN-NINTY-SIX
HH or HH12	Hour of the day: 5
HH24	Hour of the day: 20
MI	Minute of hour: 30
SS	Second of minute: 30
A.M. or P.M.	Displays A.M. or P.M. depending on the time.
Fm	Removes trailing spaces. 'May ' becomes 'May'
TH	Suffix to number: DDTH will produce 16th
SP	Number Spelled out: DDSP will produce THIRD for day 3.

Table 3: TO_CHAR and TO_DATE formats.

The following query retrieves details of the students who have joined in the month of April in year 2001.

```
select bcode, name from students
where  to_char(dj,'mmyyyy') = '042001';

BCODE NAME
----- ------------------------------
b5    Richard Marx
b5    Tina Turner
b5    Jody Foster
```

In the following example TO_CHAR is used to display month name of the year. However, as you can see in the output, there are trailing spaces after month name. This is because Oracle pads the name to 9 characters. Months that have smaller name than that will have trailing spaces.

```
select bcode, name, to_char(dj,'dd-Month-yyyy') dj from students

BCODE NAME                           DJ
----- ------------------------------ -----------------
b1    George Micheal                 10-January  -2001
b1    Micheal Douglas                11-January  -2001
b2    Andy Roberts                   11-January  -2001
b2    Malcom Marshall                16-January  -2001
b2    Vivan Richards                 16-January  -2001
b3    Chirs Evert                    14-January  -2001
b3    Ivan Lendal                    15-January  -2001
b4    George Micheal                 01-March    -2001
b5    Richard Marx                   06-April    -2001
b5    Tina Turner                    06-April    -2001
b5    Jody Foster                    07-April    -2001
```

Format fm can be used to remove these trailing spaces in months name. Here is revised version of the above query.

```
select bcode, name, to_char(dj,'dd-fmMonth-yyyy') dj from students

BCODE NAME                           DJ
----- ------------------------------ -----------------
b1    George Micheal                 10-January-2001
b1    Micheal Douglas                11-January-2001
b2    Andy Roberts                   11-January-2001
b2    Malcom Marshall                16-January-2001
b2    Vivan Richards                 16-January-2001
b3    Chirs Evert                    14-January-2001
b3    Ivan Lendal                    15-January-2001
b4    George Micheal                 01-March-2001
b5    Richard Marx                   06-April-2001
b5    Tina Turner                    06-April-2001
b5    Jody Foster                    07-April-2001
```

Note: *The output of TO_CHAR will be in the same case as the format. For example, if* Month *is given then output will be* April*; if* MONTH *is given then output will be* APRIL*.*

TO_DATE function

TO_DATE is used to convert a CHAR type value to DATE type. If the value is in DD-MON-YY or DD-MM-YYYY format then TO_DATE is not needed because Oracle implicitly converts the value to DATE type.

When you insert a record with only date in DD-MON-YY format, time portion of the date is set to 00:00:00. The following INSERT inserts a new row into PAYMETS table with date as well as time.

```
insert into payments
     values ( 10,to_date('14-04-2001 10:20:00',
                         'dd-mm-yyyy hh24:mi:ss'), 2000);
```

It is important to make sure the values given are matching with the format. That means, in the above example, as we gave *dd-mm-yyyy hh24:mi:ss* as the formation even the data is to given in the same format. The format informs to Oracle how to interpret the given values. If there is any mismatch, the values may be misinterpreted.

The *format options* are same as TO_CHAR function format options. See **Table 3** for available format options.

TO_NUMBER function

This function is required in only two occasions. The following are the two cases.

- To convert formatted number to number.
- To sort CHAR data in numeric order.

The first application of TO_NUMBER is to convert formatted number to number. The following example is trying to multiply $333 by 20. But as the number with currency symbol is not taken as a number by Oracle, it results in error.

```
SQL> select  $333 * 20 from dual;
select  $333 * 20 from dual
        *
ERROR at line 1:
ORA-00911: invalid character
```

TO_NUMBER function can be used to convert $333 to a number so that it is treated as a number by Oracle. The format in TO_NUMBER specified that the first character is to be taken as currency symbol and remaining as digits.

```
SQL> select to_number('$333','$999') * 20 from dual

TO_NUMBER('$333','$999')*20
---------------------------
                       6660
```

Sorting strings in numeric order

Another usage of TO_NUMBER is to sort a column that contains numbers but stored in the form of CHAR type.

Assume the following data is existing in VNO column of VEHICLES table. Column VNO is defined as VARCHAR2(10).

```
SQL> select vno from vehicles;

VNO
----------
1133
1583
2502
5657
9
234
45
```

The following SELECT sorts the data but sorts the column VNO as a collection of strings. That means the numeric values are not taken into account and numbers are taken as a collection of characters (each digits is a character).

```
select vno from vehicles order by vno;

VNO
----------
1133
1583
234
2502
45
5657
9
```

The output show that number 9 is at the bottom. It is because of the way strings are compared in sorting – first character first and then second character and so on.

To sort the data using numeric value, issue the following command where VNO column is converted to a number before it is sorted using TO_NUMBER function.

```
select vno from vehicles order by to_number(vno);
```

```
VNO
----------
9
45
234
1133
1583
2502
5657
```

We will see a lot of usage of conversion function TO_CHAR throughout the rest of the book.

Miscellaneous Functions

Miscellaneous functions are the functions that can be used with any data type. See table 4 for the list of miscellaneous functions.

Function	**Description**
DECODE(expression,cond,value, cond,value,...,elsevalue)	If *expression* is equivalent to first *cond* then first *value* is returned otherwise Oracle checks whether the *expression* is equivalent to second *cond* then second value is returned. If *expression* doesn't match with any of the values then *elsevalue* is returned.
GREATEST(value1,value2,...)	Returns the greatest of the given values.
LEAST(value1, value2, ...)	Returns the least value of the given values.
NVL(value1,value2)	Return *value2* if *value1* is null otherwise returns *value1*.

Table 4: Miscellaneous Functions.

The following examples will show you how to use miscellaneous functions.

DECODE function

This function works like a multiple IF statement or a CASE/SWITCH statement in a typical programming language.

It takes a value and compares it with the given values one by one. Wherever the value is equivalent to the given value it returns the corresponding value.

The following example shows how to decode the GRADE of COURSE_FACULTY table.

```
select fcode, ccode, decode(grade,'A','Very Good',
                                  'B','Good',
                                  'C', 'Average',
                                  'Unknown') Grade
from course_faculty

FCODE CCODE GRADE
----- ----- -------------
gk    ora   Very Good
kl    ora   Very Good
jc    ora   Very Good
da    vbnet Very Good
sw    asp   Very Good
da    asp   Good
hs    c     Very Good
dh    xml   Very Good
jj    java  Very Good
hs    java  Good
jj    c     Very Good
jj    vbnet Good
```

The function is used to display meaningful text for column GRADE, which contains only A,B or C.

The following example shows another usage of DECODE where we display the total remuneration paid to faculty. Assuming the payment is based on the time of the batch and no. of days of the batch.

```
select bcode,ccode,fcode, stdate, enddate,
decode(timing,1,200,2,150,175) * (enddate-stdate)  Amount from batches
where enddate is not null

BCODE CCODE FCODE STDATE    ENDDATE      AMOUNT
----- ----- ----- --------- --------- ---------
b1    ora   gk    12-JAN-01 20-FEB-01      7800
b2    asp   da    15-JAN-01 05-MAR-01      7350
b3    c     hs    20-JAN-01 27-FEB-01      6650
b4    xml   dh    02-MAR-01 30-MAR-01      4900
b5    java  hs    05-APR-01 10-MAY-01      7000
```

GREATEST and LEAST functions

These functions take a collection of values and return a single value which is either the least or greatest of the given values as the case may be.

GREATEST is used to return the largest of the given values and LEAST the smallest of the given values.

The following example shows the discount to be given to each course. The scheme is to given discount of 10% on the course fee or 500 whichever is higher. The following query with GREATEST function will achieve the result.

```
select  ccode, name, greatest( fee * 0.10,500) Discount from courses;

CCODE NAME                              DISCOUNT
----- ------------------------------ ---------
ora   Oracle database                      500
vbnet VB.NET                               550
c     C programming                        500
asp   ASP.NET                              500
java  Java Language                        500
xml   XML Programming                      500
```

LEAST Function can be used in the same manner but it sets the upper limit. In the following query the discount is either 10% of the course fee or 500 whichever is lower.

```
select  ccode,name, least( fee * 0.10,500) Discount from courses

CCODE NAME                              DISCOUNT
----- ------------------------------ ---------
ora   Oracle database                      450
vbnet VB.NET                               500
c     C programming                        350
asp   ASP.NET                              500
java  Java Language                        450
xml   XML Programming                      400
```

***Note**: GREATEST and LEAST will not treat string literal that is in date format as date. Instead these dates are taken as strings.*

The following example shows how these two functions treat dates that are given in date format but as strings.

```
select  greatest ( '12-jun-2001','17-mar-2001')
from  dual

GREATEST('1
-----------
17-mar-2001
```

The above command returns '17-mar-2001' instead of '12-jun-2001' because when these two are treated as strings, value in second position in first string (7) is greater than it corresponding value in first string (2), so 17-mar-2001 is returned as the greatest value.

NVL function

It is used to return the second value if first value is null. This function has a lot of significance since Oracle returns a null value from any expression containing a null value.

Note: Any expression involving a null value will result in a null value.

The following query is to display the details of all batches. But we get nothing – actually null value - for NODAYS of batches b6 and b7 as they are have null value in ENDDATE. Since Oracle results in null value for any expression having a null value the result of ENDDATE-STDATE is a null value.

```
select bcode, stdate, enddate - stdate nodays from batches;

BCODE STDATE       NODAYS
----- --------- ---------
b1    12-JAN-01        39
b2    15-JAN-01        49
b3    20-JAN-01        38
b4    02-MAR-01        28
b5    05-APR-01        35
b6    12-JUL-01
b7    15-AUG-01
```

However, now we want to take ending date if batch is completed otherwise we want to take system date as ending date.

```
select bcode, stdate, nvl(enddate,sysdate) - stdate nodays from batches;

BCODE STDATE       NODAYS
----- --------- ---------
b1    12-JAN-01        39
b2    15-JAN-01        49
b3    20-JAN-01        38
```

```
b4    02-MAR-01        28
b5    05-APR-01        35
b6    12-JUL-01  50.17985
b7    15-AUG-01  16.17985
```

```
Now we want to include even the status of the batch, which will be COMPLETED if
ENDDATE is not null otherwise RUNNING.
```

```
select bcode, stdate, nvl(enddate,sysdate) - stdate nodays,
       decode(enddate,null,'Running','Completed') Status from batches;

BCODE STDATE       NODAYS STATUS
----- --------- --------- ---------
b1    12-JAN-01        39 Completed
b2    15-JAN-01        49 Completed
b3    20-JAN-01        38 Completed
b4    02-MAR-01        28 Completed
b5    05-APR-01        35 Completed
b6    12-JUL-01   50.1811 Running
b7    15-AUG-01   16.1811 Running
```

Summary

String functions manipulate strings. Conversion functions are used to convert the data type of a value from one to another. In fact, Oracle always tries to convert the given value to the required data type. But in some cases as Oracle cannot convert implicitly, conversion functions are to be used to convert the value to the required data type.

Miscellaneous functions like DECODE and NVL can be used with any data type. DECODE is an if-elseif-else structure. NVL returns either the first value if it is not null or second value if first value is null.

Exercises

1. ______ function performs one to one character substitution.
2. ______ format option is used to get complete year spelled out in TO_CHAR function.
3. _____ symbol is used to concatenate strings.
4. What happens if ‘replace string’ is not given for REPLACE functions.

5. Can a NUMBER be converted to DATE? [Yes/No] _____.
6. How do you change the name of each student to uppercase in STUDENTS table.
7. Display the names of the students who have more than 15 characters in the name.
8. Display students 'first name' second and 'second name' first.

   For example, Louis Figo should be displayed as Figo Louis.
9. Display the details of the students who have more than 10 characters in the first name.
10. What is the result of AMOUNT – DISCOUNT if column DISCOUNT is null.
11. How do you get the position of 5th occurrence of letter 'o' in student's name.
12. What will be the result of select '10' * '20' from dual;