Table of Contents

Oracle has many string functions. The ones we will discuss are:

Length	InitCap	Trim	Substr	Translate.
Upper	RPad	LTrim	Instr	Chr
Lower	LPad	RTrim	Replace	Ascii

Many of these functions have optional arguments to let you return a more precise value. The examples will start with the simpler versions and then add the additional arguments. Particularly with string functions we need to consider what happens if we "go off the end" of a string. The examples will show some of these situations.

Many of the expressions below are concatenated with an 'X' character to let you see any trailing blanks returned.

1. Capitalization

UPPER, LOWER, INITCAP return the string in a specific case pattern. This is often used in a Where clause:
WHERE UPPER(last_name) = 'KING' when you want to do a case-independent match
The InitCap pattern displays the string in a mixed case format that does not always refer to the way that a human would case a name. If the case pattern of people's name is critical to your business, then you should store the name in the pattern that a person uses for their name.

Demo 01:

```
select Upper( 'This is MY sTrInG'), Upper( '50 Phelan Ave SF 94112')
from dual;
```

```
UPPER('THISISMYST    UPPER('50PHELANAVESF94
-----------------    ----------------------
THIS IS MY STRING    50 PHELAN AVE SF 94112
```

```
select Lower( 'This is MY sTrInG')   , Initcap ( 'This is MY sTrInG')
from dual;
```

```
LOWER('THISISMYST    INITCAP('THISISMY
-----------------    -----------------
this is my string    This Is My String
```

```
select Initcap( 'MACDONALD  McDonald   o''reilly') , Initcap('e. e. Cummings')
from dual;
```

```
INITCAP('MACDONALDMCDONALDO''R    INITCAP('E.E.C
------------------------------    --------------
Macdonald  Mcdonald   O'Reilly    E. E. Cummings
```

Demo 02: This is a query with a wildcard test and the Upper function.

```
select prod_id, prod_name
from prd_products
where upper(prod_desc) like '%BIRD%;
```

```
PROD_ID                PROD_NAME
---------------------- -------------------------
1140                   Bird cage- simple
1141                   Bird cage- deluxe
1142                   Bird seed
```

2. Padding and Trimming strings

RPAD, LPAD returns a string of a specified length
RPAD (strExp1, len, strExp2) returns the value of the strExp1 padded with the characters in strExp2 to length len. This could result in a truncated string. StrExp2 defaults to the space character. StrExpr2 could be a column name or other string expression.

Demo 03:

```
select
  RPad ( 'MyString', 20)  || 'X'
, RPad ( 'MyString', 4)   || 'X'
, RPad ( 'MyString', 20, '-*')  || 'X'
, LPad ( 'MyString', 20, ':')
from dual;
```

```
RPAD('MYSTRING',20)||    RPAD(    RPAD('MYSTRING',20,'-    LPAD('MYSTRING',20,'
---------------------    -----    ---------------------    --------------------
MyString            X    MyStX    MyString-*-*-*-*-*-*X    ::::::::::::MyString
```

```
select Lpad(25.95, 10, 0)    from dual;
```

```
LPAD(25.95
----------
0000025.95
```

In this case Oracle silently casts the number to a string to do the LPad since LPad expects a string.

Demo 04:

```
select rpad(  lpad ( name_last, 20 + length (name_last)/2, '.'), 40,'.')
from emp_employees where rownum < 5;
```

```
RPAD(LPAD(NAME_LAST,20+LENGTH(NAME_LAST)
----------------------------------------
..................King..................
.................Harts..................
..................Koch..................
.................Green..................
```

RTRIM, LTRIM removes leading/trailing characters
RTRIM (strExp1, strExp2) returns the value of strExp1 with the specified characters removed from the right-hand end of the string. strExp2 defaults to the blank character.
The second argument can be a list of characters to trim from the end of the string

Demo 05:

```
select 'X' ||   RTRIM  ( '   This is my string    ')   || 'X'  from dual;
```

```
'X'||RTRIM('THISISMYST
----------------------
X   This is my stringX
```

This removes $ and * from the Right. Left or both sides of the string.

```
   select
     RTRIM ( '**$$*$This ** is $50.00***', '*$')
   , LTRIM ( '**$$*$This ** is $50.00***', '*$')
   , RTRIM(LTRIM ( '**$$*$This ** is $50.00***', '*$'), '*$')
   from dual;
```

```
RTRIM('**$$*$THIS**IS$5    LTRIM('**$$*$THIS**I    RTRIM(LTRIM('**$$
-----------------------    --------------------    -----------------
**$$*$This ** is $50.00    This ** is $50.00***    This ** is $50.00
```

TRIM lets you trim from both ends with one function call. Trim takes a single character only. The plain Trim removes leading and trailing blanks.

Demo 06:

```
   select TRIM( '   This is my string    ')   || 'X'  from dual;
```

```
TRIM('THISISMYSTRI
------------------
This is my stringX
```

```
   Select
     TRIM( LEADING   ' ' FROM '   This is my string    ')  || 'X'
   , TRIM( TRAILING  ' ' FROM '   This is my string    ')  || 'X'
   , TRIM( BOTH      ' ' FROM '   This is my string    ')  || 'X'
   From dual;
```

```
TRIM(LEADING''FROM'THI    TRIM(TRAILING''FROM'T    TRIM(BOTH''FROM'TH
----------------------    ---------------------    ------------------
This is my string    X       This is my stringX    This is my stringX
```

```
   select TRIM( BOTH '*' FROM '**$$*$This ** is $50.00***')  from dual;
```

```
TRIM(BOTH'*'FROM'**$$
---------------------
$$*$This ** is $50.00
```

You will often want to use Upper and Trim when testing data values. Data values often have case issues and may contain trailing or leading blanks. There also are issues in dealing with testing Char and Varchar data with trailing blanks.

Demo 07:

```
   Create table ZStrings (col_id number, col_varchar varchar2(10), col_char
   Char(10));
   Insert into ZStrings  values (1, 'CAT', 'CAT');
   Insert into ZStrings  values (2, '  CAT', '  CAT');
   Insert into ZStrings  values (3, 'CAT   ', 'CAT   ');
   Insert into ZStrings  values (4, '  CAT  ', '  CAT  ');
```

Demo 08:

```
   select col_id,
   'x' || col_varchar || 'x' AS Varchardata,
   'x' || col_char || 'x' AS chardata
   from ZStrings
   ;
```

```
COL_ID                 VARCHARDATA  CHARDATA
---------------------- ------------ ------------
1                      xCATx        xCAT       x
2                      x  CATx      x  CAT     x   ← has leading blank
3                      xCAT   x     xCAT       x   ← has trailing blank
4                      x  CAT  x    x  CAT     x   ← has leading & trailing blank
```

If we run this with a filter of `WHERE colChar = 'CAT'`, then rows 1 and 3 are returned. The literal is end padded with blanks to 10 characters.

Demo 09:

```
select col_id, col_char
from zStrings
where col_char = 'CAT';
```

```
COL_ID                 COL_CHAR
---------------------- ----------
1                      CAT
3                      CAT
```

If we run this with a filter of `WHERE colVarChar = 'CAT'`, then only the first row is returned. The literal is not padded with the varchar test.

Demo 10:

```
select col_id, col_varchar
from zStrings
where col_varchar = 'CAT';
```

```
COL_ID                 COL_VARCHAR
---------------------- -----------
1                      CAT
```

Use `WHERE Trim(colVarChar) = 'CAT'` to avoid this issue if the leading and trailing blanks are not to be considered important.

Demo 11:

```
select col_id, col_varchar
from zstrings
where Trim(col_VarChar) = 'CAT';
```

```
COL_ID                 COL_VARCHAR
---------------------- -----------
1                      CAT
2                        CAT
3                      CAT
4                        CAT
```

3. Parts of strings and matches within a string

SUBSTR –returns part of a string. SUBSTR (strExp1, pos_start, len) returns part of strExp1, starting from position pos_start and continuing for len characters. If len is omitted, the trailing end of the string is returned. If pos_start is 0, it is treated as 1. If len is negative, a null is returned.

Demo 12:

```
SUBSTR( 'ABCDEFGHIJK', 5)   returns  EFGHIJK
From position 5 to the end of the string

SUBSTR( 'ABCDEFGHIJK', 5, 3)  returns  EFG
From position 5 for a length of 3 characters

SUBSTR( 'ABCDEFGHIJK', 5, 60)  returns EFGHIJK
It is not an error to ask for too many characters

SUBSTR( 'ABCDEFGHIJK', 25)  returns  no value is returned since we are past the
end of the string

SUBSTR( 'ABCDEFGHIJK', -5)  returns GHIJK
-- a negative value for start, means you are counting from the end of the
string.

SUBSTR( 'ABCDEFGHIJK', -5, 2) returns GH
```

INSTR–locates one string pattern inside another string. INSTR (strExp1, strExp2, pos_start, NumOccurences) returns the position in strExp1 where strExp2 starts. The third argument allows the search to start at position pos_start. The fourth argument searches for the NumOccurences occurrence. Strings are numbered from position 1

Demo 13:

```
INSTR( 'ABCDEABCDE', 'CD')  returns 3
```

The first match of CD is in position 3

```
INSTR( 'ABCDEABCDE', 'cd')  returns 0
```

This is a case specific test and there is no match

```
INSTR( 'ABCDEABCDE', 'CD', 5)  returns 8
```

The search starts at position 5; the return value is the count from the start of the string.

```
INSTR( 'ABCDEABCDE', 'CD', 25)  returns 0
```

It is not an error to start the search after the end of the string.

```
INSTR( 'ABCDEABCDE', 'CD', 1, 2)  returns 8
```

This is looking for the second occurrence of CD

```
INSTR( 'ABCDEABCDE', 'CD', 1, 3)  returns  0
```

This is looking for the third occurrence of CD

Demo 14: Returns all products that have the string "mixer" in their description.

```
select prod_id
, prod_name
, prod_desc
from prd_products
where INSTR( UPPER(prod_desc ), 'MIXER') > 0;
```

Demo 15: Returns all products that do not have the string "mixer" in their description.

```
select prod_id
, prod_name
, prod_desc
from prd_products
where INSTR( UPPER(prod_desc ), 'MIXER') =0;
```

4. Changing the string contents

REPLACE (`strExp1, StrExp2, strExp3`) replaces, or deletes, occurrences of one string within another.

Demo 16:

```
select  REPLACE( 'ABCDABCDABCD',   'B',  'cat')
,       REPLACE( 'ABCDABCDABCD',  'BCD',  '-')
from dual;
```

```
REPLACE('ABCDABCDA    REPLAC
------------------    ------
AcatCDAcatCDAcatCD    A-A-A-
```

```
select  REPLACE( 'ABCDABCDABCD', 'DCB', 'xy')
,       REPLACE( 'ABCDABCDABCD', 'C')
from dual;
```

```
REPLACE('ABC    REPLACE('
------------    ---------
ABCDABCDABCD    ABDABDABD
```

TRANSLATE replaces one set of characters with another set of characters; if any of the arguments are null, the return value is null. We have three arguments, the expression to be translated, the from_string and the to_string. This is done on a character basis. (Replace works on a string basis)

Demo 17: In this function call 1 is translated to A, 2 to B and 3 to C. 4 and 5 have no translation characters in the from_string so they are not translated.

```
select TRANSLATE( '123452313',    '123',   'ABC') from dual;
```

```
TRANSLATE('123452313','123','ABC')
--------------------------------
ABC45BCAC
```

Now 3 is translated to C; 4 and 5 have no characters in the to_string argument and they are removed from the final result.

```
select TRANSLATE( '4152393768', '345',   'C') from dual;
```

```
TRANSLATE('4152393768','345','C')
---------------------------------
12C9C768
```

Demo 18: Removing characters with translate

Suppose I have phone numbers that may have been entered with punctuation marks and I want to strip out the punctuation- specifically any space, parentheses, hyphens, dots, underscores. The result in column 1 returns a null because the ZLS in Oracle is considered a null. The solution is to tack on a character and translate it to itself (col2- all of the W were translated to W and the other characters in the from_string had no match in the to_string and were removed.).

```
variable ph varchar2(20);
```

```
exec :ph :='(415) 239-3768';

select
translate(:ph, ' ()-.#_', '') as col1,
translate(:ph, 'W ()-.#_', 'W') as col2
from dual;
```

```
COL1          COL2
----------    ----------
              4152393768
```

Demo 19: Switching delimiters in a string- assumes that all of the commas are delimiters

```
select translate ('cat,dog,bird,horse,house fly', ',', '|') as col1
from dual;
```

```
COL1
----------------------------
cat|dog|bird|horse|house fly
```

Switching delimiters and removing all blanks ; the blanks within a value in the list are also removed.

```
select translate ('cat ,dog  , bird  ,  horse, house fly ', ', ', '|') as col1
from dual;
```

```

COL1
---------------------------
cat|dog|bird|horse|housefly
```

5. Misc string functions

Length: Length(strExp) returns the number of characters in the expression. If the argument is a varchar2 column, then Length returns the actual number of characters in the data value, not the defined maximum length of the column. If the argument is a null string, Length returns null (not 0).

Demo 20: **ASCII** returns the ASCII number corresponding to the first character in the argument string. The last column aliases defaulted to uppercase but the function ran with the lower case argument.

```
select ASCII('BUSH'), ASCII('B'), ASCII('baby') from dual;
```

```
ASCII('BUSH')    ASCII('B')     ASCII('BABY')
-------------    -----------    -------------
           66             66               98
```

Demo 21: **CHR** returns the character associated with an ASCII number

```
select CHR( 73), CHR( 74) ,CHR( 888)   from dual;
```

```
C    C    C
-    -    -
I    J    x
```