# **SQL | String functions**

**String functions**  
are used to perform an operation on input string and return an output string.  
Following are the string functions defined in SQL:

1. **ASCII():** This function is used to find the ASCII value of a character.
2. **Syntax:** SELECT ascii('t');

**Output:** 116

1. **CHAR\_LENGTH():** This function is used to find the length of a word.
2. **Syntax:** SELECT char\_length('Hello!');

**Output:** 6

1. **CHARACTER\_LENGTH():** This function is used to find the length of a line.
2. **Syntax:** SELECT CHARACTER\_LENGTH('geeks for geeks');

**Output:** 15

1. **CONCAT():** This function is used to add two words or strings.
2. **Syntax:** SELECT 'Geeks' || ' ' || 'forGeeks' FROM dual;

**Output:** ‘GeeksforGeeks’

1. **CONCAT\_WS():** This function is used to add two words or strings with a symbol as concatenating symbol.
2. **Syntax:** SELECT CONCAT\_WS('\_', 'geeks', 'for', 'geeks');

**Output:** geeks\_for\_geeks

1. **FIND\_IN\_SET():** This function is used to find a symbol from a set of symbols.
2. **Syntax:** SELECT FIND\_IN\_SET('b', 'a, b, c, d, e, f');

**Output:** 2

1. **FORMAT():** This function is used to display a number in the given format.
2. **Syntax:** Format("0.981", "Percent");

**Output:** ‘98.10%’

1. **INSERT():** This function is used to insert the data into a database.
2. **Syntax:** INSERT INTO database (geek\_id, geek\_name) VALUES (5000, 'abc');

**Output:** successfully updated

1. **INSTR():** This function is used to find the occurrence of an alphabet.
2. **Syntax:** INSTR('geeks for geeks', 'e');

**Output:** 2 (the first occurrence of ‘e’)

**Syntax:** INSTR('geeks for geeks', 'e', 1, 2 );

**Output:** 3 (the second occurrence of ‘e’)

1. **LCASE():** This function is used to convert the given string into lower case.
2. **Syntax:** LCASE ("GeeksFor Geeks To Learn");

**Output:** geeksforgeeks to learn

1. **LEFT():** This function is used to SELECT a sub string from the left of given size or characters.
2. **Syntax:** SELECT LEFT('geeksforgeeks.org', 5);

**Output:** geeks

1. **LENGTH():** This function is used to find the length of a word.
2. **Syntax:** LENGTH('GeeksForGeeks');

**Output:** 13

1. **LOCATE():** This function is used to find the nth position of the given word in a string.
2. **Syntax:** SELECT LOCATE('for', 'geeksforgeeks', 1);

**Output:** 6

1. **LOWER():** This function is used to convert the upper case string into lower case.
2. **Syntax:** SELECT LOWER('GEEKSFORGEEKS.ORG');

**Output:** geeksforgeeks.org

1. **LPAD():** This function is used to make the given string of the given size by adding the given symbol.
2. **Syntax:** LPAD('geeks', 8, '0');
3. **Output:**

000geeks

1. **LTRIM():** This function is used to cut the given sub string from the original string.
2. **Syntax:** LTRIM('123123geeks', '123');

**Output:** geeks

1. **MID():** This function is to find a word from the given position and of the given size.
2. **Syntax:** Mid ("geeksforgeeks", 6, 2);

**Output:** for

1. **POSITION():** This function is used to find position of the first occurrence of the given alphabet.
2. **Syntax:** SELECT POSITION('e' IN 'geeksforgeeks');

**Output:** 2

1. **REPEAT():** This function is used to write the given string again and again till the number of times mentioned.
2. **Syntax:** SELECT REPEAT('geeks', 2);

**Output:** geeksgeeks

1. **REPLACE():** This function is used to cut the given string by removing the given sub string.
2. **Syntax:** REPLACE('123geeks123', '123');

**Output:** geeks

1. **REVERSE():** This function is used to reverse a string.
2. **Syntax:** SELECT REVERSE('geeksforgeeks.org');

**Output:** ‘gro.skeegrofskeeg’

1. **RIGHT():** This function is used to SELECT a sub string from the right end of the given size.
2. **Syntax:** SELECT RIGHT('geeksforgeeks.org', 4);

**Output:** ‘.org’

1. **RPAD():** This function is used to make the given string as long as the given size by adding the given symbol on the right.
2. **Syntax:** RPAD('geeks', 8, '0');

**Output:** ‘geeks000’

1. **RTRIM():** This function is used to cut the given sub string from the original string.
2. **Syntax:** RTRIM('geeksxyxzyyy', 'xyz');

**Output:** ‘geeks’

1. **SPACE():** This function is used to write the given number of spaces.
2. **Syntax:** SELECT SPACE(7);

**Output:** ‘ ‘

1. **STRCMP():** This function is used to compare 2 strings.
   * If string1 and string2 are the same, the STRCMP function will return 0.
   * If string1 is smaller than string2, the STRCMP function will return -1.
   * If string1 is larger than string2, the STRCMP function will return 1.
2. **Syntax:** SELECT STRCMP('google.com', 'geeksforgeeks.com');

**Output:** -1

1. **SUBSTR():** This function is used to find a sub string from the a string from the given position.
2. **Syntax:**SUBSTR('geeksforgeeks', 1, 5);

**Output:** ‘geeks’

1. **SUBSTRING():** This function is used to find an alphabet from the mentioned size and the given string.
2. **Syntax:** SELECT SUBSTRING('GeeksForGeeks.org', 9, 1);

**Output:** ‘G’

1. **SUBSTRING\_INDEX():** This function is used to find a sub string before the given symbol.
2. **Syntax:** SELECT SUBSTRING\_INDEX('www.geeksforgeeks.org', '.', 1);

**Output:** ‘www’

1. **TRIM():** This function is used to cut the given symbol from the string.
2. **Syntax:** TRIM(LEADING '0' FROM '000123');

**Output:** 123

1. **UCASE():** This function is used to make the string in upper case.
2. **Syntax:** UCASE ("GeeksForGeeks");
3. **Output:**

GEEKSFORGEEKS

**Built-in Functions in SQL**

**Numeric Functions**

|  |  |  |
| --- | --- | --- |
| **Function** | **Input Argument** | **Value Returned** |
| ABS ( m ) | m = value | Absolute value of m |
| MOD ( m, n ) | m = value, n = divisor | Remainder of m divided by n |
| POWER ( m, n ) | m = value, n = exponent | m raised to the nth power |
| ROUND ( m [, n ] ) | m = value, n = number of decimal places, default 0 | m rounded to the nth decimal place |
| TRUNC ( m [, n ] ) | m = value, n = number of decimal places, default 0 | m truncated to the nth decimal place |
| SIN ( n ) | n = angle expressed in radians | sine (n) |
| COS ( n ) | n = angle expressed in radians | cosine (n) |
| TAN ( n ) | n = angle expressed in radians | tan (n) |
| ASIN ( n ) | n is in the range -1 to +1 | arc sine of n in the range -π/2 to +π/2 |
| ACOS ( n ) | n is in the range -1 to +1 | arc cosine of n in the range 0 to π |
| ATAN ( n ) | n is unbounded | arc tangent of n in the range -π/2 to + π/2 |
| SINH ( n ) | n = value | hyperbolic sine of n |
| COSH ( n ) | n = value | hyperbolic cosine of n |
| TANH ( n ) | n = value | hyperbolic tangent of n |
| SQRT ( n ) | n = value | positive square root of n |
| EXP ( n ) | n = value | e raised to the power n |
| LN ( n ) | n > 0 | natural logarithm of n |
| LOG ( n2, n1 ) | base n2 any positive value other than 0 or 1, n1 any positive value | logarithm of n1, base n2 |
| CEIL ( n ) | n = value | smallest integer greater than or equal to n |
| FLOOR ( n ) | n = value | greatest integer smaller than or equal to n |
| SIGN ( n ) | n = value | -1 if n < 0, 0 if n = 0, and 1 if n > 0 |

Here are some examples of the use of some of these numeric functions:

select round (83.28749, 2) from dual;

select sqrt (3.67) from dual;

select power (2.512, 5) from dual;

**String Functions**

|  |  |  |
| --- | --- | --- |
| **Function** | **Input Argument** | **Value Returned** |
| INITCAP ( s ) | s = character string | First letter of each word is changed to uppercase and all other letters are in lower case. |
| LOWER ( s ) | s = character string | All letters are changed to lowercase. |
| UPPER ( s ) | s = character string | All letters are changed to uppercase. |
| CONCAT ( s1, s2 ) | s1 and s2 are character strings | Concatenation of s1 and s2. Equivalent to *s1 || s2* |
| LPAD ( s1, n [, s2] ) | s1 and s2 are character strings and n is an integer value | Returns s1 right justified and padded left with n characters from s2; s2 defaults to space. |
| RPAD ( s1, n [, s2] ) | s1 and s2 are character strings and n is an integer value | Returns s1 left justified and padded right with n characters from s2; s2 defaults to space. |
| LTRIM ( s [, set ] ) | s is a character string and *set*is a set of characters | Returns s with characters removed up to the first character not in set; defaults to space |
| RTRIM ( s [, set ] ) | s is a character string and *set*is a set of characters | Returns s with final characters removed after the last character not in set; defaults to space |
| REPLACE ( s, search\_s [, replace\_s ] ) | s = character string, search\_s = target string, replace\_s = replacement string | Returns s with every occurrence of search\_s in s replaced by replace\_s; default removes search\_s |
| SUBSTR ( s, m [, n ] ) | s = character string, m = beginning position, n = number of characters | Returns a substring from s, beginning in position m and n characters long; default returns to end of s. |
| LENGTH ( s ) | s = character string | Returns the number of characters in s. |
| INSTR ( s1, s2 [, m [, n ] ] ) | s1 and s2 are character strings, m = beginning position, n = occurrence of s2 in s1 | Returns the position of the nth occurrence of s2 in s1, beginning at position m, both m and n default to 1. |

Here are some examples of the use of String functions:

select concat ('Alan', 'Turing') as "NAME" from dual;

select 'Alan' || 'Turing' as "NAME" from dual;

select initcap ("now is the time for all good men to come to the aid of the

party") as "SLOGAN" from dual;

select substr ('Alan Turing', 1, 4) as "FIRST" from dual;

**String / Number Conversion Functions**

|  |  |  |
| --- | --- | --- |
| **Function** | **Input Argument** | **Value Returned** |
| NANVL ( n2, n1 ) | n1, n2 = value | if (n2 = NaN) returns n1 else returns n2 |
| TO\_CHAR ( m [, fmt ] ) | m = numeric value, fmt = format | Number m converted to character string as specified by the format |
| TO\_NUMBER ( s [, fmt ] ) | s = character string, fmt = format | Character string s converted to a number as specified by the format |

**Formats for TO\_CHAR Function**

|  |  |
| --- | --- |
| **Symbol** | **Explanation** |
| 9 | Each 9 represents one digit in the result |
| 0 | Represents a leading zero to be displayed |
| $ | Floating dollar sign printed to the left of number |
| L | Any local floating currency symbol |
| **.** | Prints the decimal point |
| **,** | Prints the comma to represent thousands |
|  |  |

**Group Functions**

|  |  |  |
| --- | --- | --- |
| **Function** | **Input Argument** | **Value Returned** |
| AVG ( [ DISTINCT | ALL ] col ) | col = column name | The average value of that column |
| COUNT ( \* ) | none | Number of rows returned including duplicates and NULLs |
| COUNT ( [ DISTINCT | ALL ] col ) | col = column name | Number of rows where the value of the column is not NULL |
| MAX ( [ DISTINCT | ALL ] col ) | col = column name | Maximum value in the column |
| MIN ( [ DISTINCT | ALL ] col ) | col = column name | Minimum value in the column |
| SUM ( [ DISTINCT | ALL ] col ) | col = column name | Sum of the values in the column |
| CORR ( e1, e2 ) | e1 and e2 are column names | Correlation coefficient between the two columns after eliminating nulls |
| MEDIAN ( col ) | col = column name | Middle value in the sorted column, interpolating if necessary |
| STDDEV ( [ DISTINCT | ALL ] col ) | col = column name | Standard deviation of the column ignoring NULL values |
| VARIANCE ( [ DISTINCT | ALL ] col ) | col = column name | Variance of the column ignoring NULL values |

**Date and Time Functions**

|  |  |  |
| --- | --- | --- |
| **Function** | **Input Argument** | **Value Returned** |
| ADD\_MONTHS ( d, n ) | d = date, n = number of months | Date d plus n months |
| LAST\_DAY ( d ) | d = date | Date of the last day of the month containing d |
| MONTHS\_BETWEEN ( d, e ) | d and e are dates | Number of months by which e precedes d |
| NEW\_TIME ( d, a, b ) | d = date, a = time zone (char), b = time zone (char) | The date and time in time zone b when date d is for time zone a |
| NEXT\_DAY ( d, day ) | d = date, day = day of the week | Date of the first day of the week after d |
| SYSDATE | none | Current date and time |
| GREATEST ( d1, d2, ..., dn ) | d1 ... dn = list of dates | Latest of the given dates |
| LEAST ( d1, d2, ..., dn ) | d1 ... dn = list of dates | Earliest of the given dates |

**Date Conversion Functions**

|  |  |  |
| --- | --- | --- |
| **Function** | **Input Argument** | **Value Returned** |
| TO\_CHAR ( d [, fmt ] ) | d = date value, fmt = format for string | The date d converted to a string in the given format |
| TO\_DATE ( s [, fmt ] ) | s = character string, fmt = format for date | String s converted to a date value |
| ROUND ( d [, fmt ] ) | d = date value, fmt = format for string | Date d rounded as specified by the format |
| TRUNC ( d [, fmt ] ) | d = date value, fmt = format for string | Date d truncated as specified by the format |

**Date Formats**

|  |  |  |
| --- | --- | --- |
| **Format Code** | **Description** | **Range of Values** |
| DD | Day of the month | 1 - 31 |
| DY | Name of the day in 3 uppercase letters | SUN, ..., SAT |
| DAY | Complete name of the day in uppercase, padded to 9 characters | SUNDAY, ..., SATURDAY |
| MM | Number of the month | 1 - 12 |
| MON | Name of the month in 3 uppercase letters | JAN, ..., DEC |
| MONTH | Name of the month in uppercase padded to a length of 9 characters | JANUARY, ..., DECEMBER |
| RM | Roman numeral for the month | I, ..., XII |
| YY or YYYY | Two or four digit year | 71 or 1971 |
| HH:MI:SS | Hours : Minutes : Seconds | 10:28:53 |
| HH 12 or HH 24 | Hour displayed in 12 or 24 hour format | 1 - 12 or 1 - 24 |
| MI | Minutes of the hour | 0 - 59 |
| SS | Seconds of the minute | 0 - 59 |
| AM or PM | Meridian indicator | AM or PM |
| SP | A suffix that forces the number to be spelled out. | e.g. TWO THOUSAND NINE |
| TH | A suffix meaning that the ordinal number is to be added | e.g. 1st, 2nd, 3rd, ... |
| FM | Prefix to DAY or MONTH or YEAR to suppress padding | e.g. MONDAY with no extra spaces at the end |

Here are some examples of the use of the Date functions:

select to\_char ( sysdate, 'MON DD, YYYY' ) from dual;

select to\_char ( sysdate, 'HH12:MI:SS AM' ) from dual;

select to\_char ( new\_time ( sysdate, 'CDT', 'GMT'), 'HH24:MI' ) from dual;

select greatest ( to\_date ( 'JAN 19, 2000', 'MON DD, YYYY' ),

to\_date ( 'SEP 27, 1999', 'MON DD, YYYY' ),

to\_date ( '13-Mar-2009', 'DD-Mon-YYYY' ) )

from dual;

select next\_day ( sysdate, 'FRIDAY' ) from dual;

select last\_day ( add\_months ( sysdate, 1 ) ) from dual;