Common String Functions

There are many string functions available for use in SQL statements. These are useful for working with text or character string data types. The following list is not exhaustive, but it does present some of the more common ones you might want to use.

Unless otherwise noted, the function takes a string argument and returns a string.

Note the bottom square box character (␣) is sometimes used to represent whitespace, which could be, for example, a space or a tab character. For some of these, it would be difficult to see the effect of the function if regular spaces are used here.

length(str)

This returns an integer value equal to the number of characters in the string argument str.

Notes:

- The name of this function is different, depending on the SQL engine you're using. For example, some engines use len(str) or char_len(str). (The other functions described below have the same name across all the major SQL engines.)
- For Apache Hive and Apache Impala, use the length function; it works as described here.
- Some SQL engines have functions that are similar to length as described above, but that return the number of bytes or other units of information that are required to store a character string. If you're using some other SQL engine besides Hive or Impala, check the documentation to be sure you understand what the length function returns and to see what other similar functions are available.

Examples:

length('Common String Functions') = 23

length(' Common String Functions ') = 25

length('') = 0

reverse(str)

This returns the characters within the string argument str, but in the reverse order. Try it with your favorite palindrome!

Examples:

reverse('Common String Functions') = 'snoitcnuF gnirtS nommoC'

reverse('never odd or even') = 'neve ro ddo reven'

upper(str), lower(str)

These return the string str but with all characters converted either to uppercase or lowercase. These can be useful for doing case-insensitive string comparisons (by converting the string to be compared to one case, for example, WHERE upper(fname) = 'BOB' or WHERE lower(fname) = 'bob').

Examples:

upper('Common String Functions') = 'COMMON STRING FUNCTIONS'

lower('Common String Functions') = 'common string functions'

trim(str), ltrim(str), rtrim(str)

These remove whitespace at the ends of the argument str. You can choose to remove only leading whitespace (ltrim for left trim), trailing whitespace (rtrim for right trim), or both (trim). If there is no whitespace on the specified end, the string is unchanged.

Examples:

trim('␣Common String Functions␣ ␣ ␣') = 'Common String Functions'

ltrim('␣Common String Functions␣ ␣ ␣') = 'Common String Functions␣ ␣ ␣'

rtrim('␣Common String Functions␣ ␣ ␣') = '␣Common String Functions'

ltrim('Common String Functions␣ ␣ ␣') = 'Common String Functions␣ ␣ ␣'

rtrim('␣Common String Functions') = '␣Common String Functions'

lpad(str, n, padstr), rpad(str, n, padstr)

These functions take a string str and an integer n and return a string of length n. If the original string str is shorter than n characters, the returned string will be str with characters from padstr added at the left (lpad) or the right (rpad) to make it length n. (This is called *padding* the string, and is the opposite of trimming.) These functions are often used to add zeros to the left or right of numbers that are represented in strings (this is called *zero-padding*). If necessary, the pad string will be repeated. If the length of str is longer, however, the function will return a truncated version of the string. Truncated characters will be taken from the right, regardless of which function you specify.

Examples:

lpad('.50', 4, '0') = '0.50'

rpad('0.5', 4, '0') = '0.50'

rpad('Common', 13, ' String') = 'Common String'

rpad('Common', 17, ' String') = 'Common String Str'

lpad('Common', 17, ' String') = ' String StrCommon'

rpad('Common String', 6, ' Function') = 'Common'

lpad('Common String', 6, ' Function') = 'Common'

substring(str, index, max_length)

This function takes a string and two integers, and returns a portion of the original string. The argument index indicates where to start the substring (indexing the original string str starting at 1) and max_length is how many characters to include (though it might be fewer, if the end of the original string is reached). With many SQL engines, you can also use substr which is an alias for substring.

Examples:

substring('Common String',1,6) = 'Common'

substring('Common String',8,3) = 'Str'

substring('Common String',8,6) = 'String'

substring('Common String',8,10) = 'String'

concat(str1, str2[, str3, …]), concat_ws(sep, str1, str2[, str3, …])

These functions *concatenate* strings—that is, they put them together into a single string. The *ws* in concat_ws stands for “with separator,” the first argument in that case is placed between each pair of strings. In both cases, the arguments are concatenated in the order given.

Notes:

- Both concat and concat_ws must include at least two strings to concatenate. They can take more than two, as well.
- Some SQL engines have an *operator* for string concatenation, usually + or ||. However, Hive and Impala do *not* have concatenation operators; one of these functions must be used.

Examples:

concat('Common','String') = 'CommonString'

concat_ws(' ','Common','String') = 'Common String'

concat('Common','String','Functions') = 'CommonStringFunctions'

concat_ws(' ','Common','String','Functions') = 'Common String Functions'

concat_ws(', ','Common','String','Functions') = 'Common, String, Functions'

Non-ASCII characters

Note that the string functions in different SQL engines can differ in their handling of non-ASCII characters. For example: In most SQL engines, upper('é') returns É, but in others it might return é or throw an error. You should test or consult the documentation to see how this works.

Other String Functions

Many more string functions are available in most SQL engines. For example, there are functions for splitting strings, extracting parts of strings, and finding and replacing specific characters or substrings within strings. If you are interested in them, check the documentation of the SQL engine you are using (probably under “String Functions”).