

MYSQL STRING FUNCTIONS

<http://www.tutorialspoint.com/mysql/mysql-string-functions.htm>

Copyright © tutorialspoint.com

Name	Description
<u>ASCII</u>	Returns numeric value of left-most character
<u>BIN</u>	Returns a string representation of the argument
<u>BIT_LENGTH</u>	Returns length of argument in bits
<u>CHAR_LENGTH</u>	Returns number of characters in argument
<u>CHAR</u>	Returns the character for each integer passed
<u>CHARACTER_LENGTH</u>	A synonym for CHAR_LENGTH
<u>CONCAT_WS</u>	Returns concatenate with separator
<u>CONCAT</u>	Returns concatenated string
<u>CONV</u>	Converts numbers between different number bases
<u>ELT</u>	Returns string at index number
<u>EXPORT_SET</u>	Returns a string such that for every bit set in the value bits, you get an on string and for every unset bit, you get an off string
<u>FIELD</u>	Returns the index <i>position</i> of the first argument in the subsequent arguments
<u>FIND_IN_SET</u>	Returns the index position of the first argument within the second argument
<u>FORMAT</u>	Returns a number formatted to specified number of decimal places
<u>HEX</u>	Returns a string representation of a hex value
<u>INSERT</u>	Inserts a substring at the specified position up to the specified number of characters
<u>INSTR</u>	Returns the index of the first occurrence of substring
<u>LCASE</u>	Synonym for LOWER
<u>LEFT</u>	Returns the leftmost number of characters as specified
<u>LENGTH</u>	Returns the length of a string in bytes
<u>LOAD_FILE</u>	Loads the named file
<u>LOCATE</u>	Returns the position of the first occurrence of substring
<u>LOWER</u>	Returns the argument in lowercase
<u>LPAD</u>	Returns the string argument, left-padded with the specified string
<u>LTRIM</u>	Removes leading spaces
<u>MAKE_SET</u>	Returns a set of comma-separated strings that have the corresponding bit in bits set
<u>MID</u>	Returns a substring starting from the specified position
<u>OCT</u>	Returns a string representation of the octal argument

<u>OCTET_LENGTH</u>	A synonym for LENGTH
<u>ORD</u>	If the leftmost character of the argument is a multi-byte character, returns the code for that character
<u>POSITION</u>	A synonym for LOCATE
<u>QUOTE</u>	Escapes the argument for use in an SQL statement
<u>REGEXP</u>	Pattern matching using regular expressions
<u>REPEAT</u>	Repeats a string the specified number of times
<u>REPLACE</u>	Replaces occurrences of a specified string
<u>REVERSE</u>	Reverses the characters in a string
<u>RIGHT</u>	Returns the specified rightmost number of characters
<u>RPAD</u>	Appends string the specified number of times
<u>RTRIM</u>	Removes trailing spaces
<u>SOUNDEX</u>	Returns a soundex string
<u>SOUNDS LIKE</u>	Compares sounds
<u>SPACE</u>	Returns a string of the specified number of spaces
<u>STRCMP</u>	Compares two strings
<u>SUBSTRING_INDEX</u>	Returns a substring from a string before the specified number of occurrences of the delimiter
<u>SUBSTRING, SUBSTR</u>	Returns the substring as specified
<u>TRIM</u>	Removes leading and trailing spaces
<u>UCASE</u>	Synonym for UPPER
<u>UNHEX</u>	Converts each pair of hexadecimal digits to a character
<u>UPPER</u>	Converts to uppercase

ASCIIstr

Returns the numeric value of the leftmost character of the string str. Returns 0 if str is the empty string. Returns NULL if str is NULL. ASCII works for characters with numeric values from 0 to 255.

```
mysql> SELECT ASCII('2');
+-----+
| ASCII('2') |
+-----+
| 50         |
+-----+
1 row in set (0.00 sec)

mysql> SELECT ASCII('dx');
+-----+
| ASCII('dx') |
+-----+
| 100         |
+-----+
1 row in set (0.00 sec)
```

BIN

Returns a string representation of the binary value of N, where N is a longlong *BIGINT* number. This is equivalent to CONV(N, 10, 2). Returns NULL if N is NULL.

```
mysql> SELECT BIN(12);
+-----+
| BIN(12) |
+-----+
| 1100    |
+-----+
1 row in set (0.00 sec)
```

BIT_LENGTHstr

Returns the length of the string str in bits.

```
mysql> SELECT BIT_LENGTH('text');
+-----+
| BIT_LENGTH('text') |
+-----+
| 32                 |
+-----+
1 row in set (0.00 sec)
```

CHARN,...[USINGcharset_name]

CHAR interprets each argument N as an integer and returns a string consisting of the characters given by the code values of those integers. NULL values are skipped.

```
mysql> SELECT CHAR(77,121,83,81,'76');
+-----+
| CHAR(77,121,83,81,'76') |
+-----+
| MySQL                   |
+-----+
1 row in set (0.00 sec)
```

CHAR_LENGTHstr

Returns the length of the string str, measured in characters. A multi-byte character counts as a single character. This means that for a string containing five two-byte characters, LENGTH returns 10, whereas CHAR_LENGTH returns 5.

```
mysql> SELECT CHAR_LENGTH("text");
+-----+
| CHAR_LENGTH("text") |
+-----+
| 4                   |
+-----+
1 row in set (0.00 sec)
```

CHARACTER_LENGTHstr

CHARACTER_LENGTH is a synonym for CHAR_LENGTH.

CONCATstr1,str2,...

Returns the string that results from concatenating the arguments. May have one or more arguments. If all arguments are non-binary strings, the result is a non-binary string. If the arguments include any binary strings, the result is a binary string. A numeric argument is converted to its equivalent binary string form; if you want to avoid that, you can use an explicit type cast, as in this example:

```
mysql> SELECT CONCAT('My', 'S', 'QL');
```

```

+-----+
| CONCAT('My', 'S', 'QL') |
+-----+
| MySQL |
+-----+
1 row in set (0.00 sec)

```

CONCAT_WS(separator, str1, str2, ...)

CONCAT_WS stands for Concatenate With Separator and is a special form of CONCAT. The first argument is the separator for the rest of the arguments. The separator is added between the strings to be concatenated. The separator can be a string, as can the rest of the arguments. If the separator is NULL, the result is NULL.

```

mysql> SELECT CONCAT_WS(',', 'First name', 'Last Name' );
+-----+
| CONCAT_WS(',', 'First name', 'Last Name' ) |
+-----+
| First name, Last Name |
+-----+
1 row in set (0.00 sec)

```

CONV(N, from_base, to_base)

Converts numbers between different number bases. Returns a string representation of the number N, converted from base from_base to base to_base. Returns NULL if any argument is NULL. The argument N is interpreted as an integer, but may be specified as an integer or a string. The minimum base is 2 and the maximum base is 36. If to_base is a negative number, N is regarded as a signed number. Otherwise, N is treated as unsigned. CONV works with 64-bit precision.

```

mysql> SELECT CONV('a', 16, 2);
+-----+
| CONV('a', 16, 2) |
+-----+
| 1010 |
+-----+
1 row in set (0.00 sec)

```

ELT(N, str1, str2, str3, ...)

Returns str1 if N = 1, str2 if N = 2, and so on. Returns NULL if N is less than 1 or greater than the number of arguments. ELT is the complement of FIELD.

```

mysql> SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo');
+-----+
| ELT(1, 'ej', 'Heja', 'hej', 'foo') |
+-----+
| ej |
+-----+
1 row in set (0.00 sec)

```

EXPORT_SET(bits, on, off[, separator[, number_of_bits]])

Returns a string such that for every bit set in the value bits, you get an on string and for every bit not set in the value, you get an off string. Bits in bits are examined from right to left from low – order to high – order bits. Strings are added to the result from left to right, separated by the separator string the default being the comma character. . . . The number of bits examined is given by number_of_bits default to 64.

```

mysql> SELECT EXPORT_SET(5, 'Y', 'N', ',', 4);
+-----+
| EXPORT_SET(5, 'Y', 'N', ',', 4) |
+-----+
| Y,N,Y,N |
+-----+

```

```
+-----+
1 row in set (0.00 sec)
```

FIELDstr, str1, str2, str3, ...

Returns the index *position starting with 1* of str in the str1, str2, str3, ... list. Returns 0 if str is not found.

```
mysql> SELECT FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo');
+-----+
| FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo') |
+-----+
| 2 |
+-----+
1 row in set (0.00 sec)
```

FIND_IN_SETstr, strlist

Returns a value in the range of 1 to N if the string str is in the string list strlist consisting of N substrings.

```
mysql> SELECT FIND_IN_SET('b', 'a,b,c,d');
+-----+
| SELECT FIND_IN_SET('b', 'a,b,c,d') |
+-----+
| 2 |
+-----+
1 row in set (0.00 sec)
```

FORMATX,D

Formats the number X to a format like '#,###,###.##', rounded to D decimal places, and returns the result as a string. If D is 0, the result has no decimal point or fractional part.

```
mysql> SELECT FORMAT(12332.123456, 4);
+-----+
| FORMAT(12332.123456, 4) |
+-----+
| 12,332.1235 |
+-----+
1 row in set (0.00 sec)
```

HEXN_or_S

If N_or_S is a number, returns a string representation of the hexadecimal value of N, where N is a longlong *BIGINT* number. This is equivalent to CONV(N, 10, 16).

If N_or_S is a string, returns a hexadecimal string representation of N_or_S where each character in N_or_S is converted to two hexadecimal digits.

```
mysql> SELECT HEX(255);
+-----+
| HEX(255) |
+-----+
| FF |
+-----+
1 row in set (0.00 sec)

mysql> SELECT 0x616263;
+-----+
| 0x616263 |
+-----+
| abc |
+-----+
1 row in set (0.00 sec)
```

INSERTstr, pos, len, newstr

Returns the string str, with the substring beginning at position pos and len characters long replaced by the string newstr. Returns the original string if pos is not within the length of the string. Replaces the rest of the string from position pos if len is not within the length of the rest of the string. Returns NULL if any argument is NULL.

```
mysql> SELECT INSERT('Quadratic', 3, 4, 'What');
+-----+
| INSERT('Quadratic', 3, 4, 'What') |
+-----+
| QuWhattic                        |
+-----+
1 row in set (0.00 sec)
```

INSTRstr, substr

Returns the position of the first occurrence of substring substr in string str. This is the same as the two-argument form of LOCATE, except that the order of the arguments is reversed.

```
mysql> SELECT INSTR('foobarbar', 'bar');
+-----+
| INSTR('foobarbar', 'bar') |
+-----+
| 4                          |
+-----+
1 row in set (0.00 sec)
```

LCASEstr

LCASE is a synonym for LOWER.

LEFTstr, len

Returns the leftmost len characters from the string str, or NULL if any argument is NULL.

```
mysql> SELECT LEFT('foobarbar', 5);
+-----+
| LEFT('foobarbar', 5) |
+-----+
| fooba                |
+-----+
1 row in set (0.00 sec)
```

LENGTHstr

Returns the length of the string str, measured in bytes. A multi-byte character counts as multiple bytes. This means that for a string containing five two-byte characters, LENGTH returns 10, whereas CHAR_LENGTH returns 5.

```
mysql> SELECT LENGTH('text');
+-----+
| LENGTH('text') |
+-----+
| 4              |
+-----+
1 row in set (0.00 sec)
```

LOAD_FILEfile_name

Reads the file and returns the file contents as a string. To use this function, the file must be located on the server host, you must specify the full pathname to the file, and you must have the FILE privilege. The file must be readable by all and its size less than max_allowed_packet bytes.

If the file does not exist or cannot be read because one of the preceding conditions is not satisfied, the function returns NULL.

As of MySQL 5.0.19, the `character_set_filesystem` system variable controls interpretation of filenames that are given as literal strings.

```
mysql> UPDATE table_test
-> SET blob_col=LOAD_FILE('/tmp/picture')
-> WHERE id=1;
.....
```

LOCATE*substr, str*, **LOCATE***substr, str, pos*

The first syntax returns the position of the first occurrence of substring *substr* in string *str*. The second syntax returns the position of the first occurrence of substring *substr* in string *str*, starting at position *pos*. Returns 0 if *substr* is not in *str*.

```
mysql> SELECT LOCATE('bar', 'foobarbar');
+-----+
| LOCATE('bar', 'foobarbar') |
+-----+
| 4                           |
+-----+
1 row in set (0.00 sec)
```

LOWER*str*

Returns the string *str* with all characters changed to lowercase according to the current character set mapping.

```
mysql> SELECT LOWER('QUADRATICALLY');
+-----+
| LOWER('QUADRATICALLY') |
+-----+
| quadratically          |
+-----+
1 row in set (0.00 sec)
```

LPAD*str, len, padstr*

Returns the string *str*, left-padded with the string *padstr* to a length of *len* characters. If *str* is longer than *len*, the return value is shortened to *len* characters.

```
mysql> SELECT LPAD('hi', 4, '??');
+-----+
| LPAD('hi', 4, '??') |
+-----+
| ??hi                |
+-----+
1 row in set (0.00 sec)
```

LTRIM*str*

Returns the string *str* with leading space characters removed.

```
mysql> SELECT LTRIM('  barbar');
+-----+
| LTRIM('  barbar') |
+-----+
| barbar            |
+-----+
1 row in set (0.00 sec)
```

MAKE_SET*bits, str1, str2, ...*

Returns a set value *astringcontainingstringsseparatedby. , . characters* consisting of the strings that have the corresponding bit in bits set. str1 corresponds to bit 0, str2 to bit 1, and so on. NULL values in str1, str2, ... are not appended to the result.

```
mysql> SELECT MAKE_SET(1, 'a', 'b', 'c');
+-----+
| MAKE_SET(1, 'a', 'b', 'c') |
+-----+
| a                            |
+-----+
1 row in set (0.00 sec)
```

MIDstr, pos, len

MIDstr, pos, len is a synonym for SUBSTRINGstr, pos, len.

OCTN

Returns a string representation of the octal value of N, where N is a longlong *BIGINT* number. This is equivalent to CONVN, 10, 8. Returns NULL if N is NULL.

```
mysql> SELECT OCT(12);
+-----+
| OCT(12) |
+-----+
| 14      |
+-----+
1 row in set (0.00 sec)
```

OCTET_LENGTHstr

OCTET_LENGTH is a synonym for LENGTH.

ORDstr

If the leftmost character of the string str is a multi-byte character, returns the code for that character, calculated from the numeric values of its constituent bytes using this formula:

```
(1st byte code)
+ (2nd byte code * 256)
+ (3rd byte code * 2562) ...
```

If the leftmost character is not a multi-byte character, ORD returns the same value as the ASCII function.

```
mysql> SELECT ORD('2');
+-----+
| ORD('2') |
+-----+
| 50       |
+-----+
1 row in set (0.00 sec)
```

POSITIONsubstrINstr

POSITIONsubstrINstr is a synonym for LOCATEsubstr, str.

QUOTEstr

Quotes a string to produce a result that can be used as a properly escaped data value in an SQL statement. The string is returned enclosed by single quotes and with each instance of single quote ' , backslash '\ , ASCII NUL, and Control-Z preceded by a backslash. If the argument is NULL, the return value is the word 'NULL' without enclosing single quotes.

```
mysql> SELECT QUOTE('Don\'t!');
+-----+
| QUOTE('Don\'t!') |
+-----+
| 'Don\'t!' |
+-----+
1 row in set (0.00 sec)
```

NOTE: Please check if your installation has any bug with this function, then don't use this function.

expr REGEXP pattern

This function performs a pattern match of expr against pattern. Returns 1 if expr matches pat; otherwise it returns 0. If either expr or pat is NULL, the result is NULL. REGEXP is not case sensitive, except when used with binary strings.

```
mysql> SELECT 'ABCDEF' REGEXP 'A%C%';
+-----+
| 'ABCDEF' REGEXP 'A%C%' |
+-----+
| 0 |
+-----+
1 row in set (0.00 sec)
```

Another example is:

```
mysql> SELECT 'ABCDE' REGEXP '.*';
+-----+
| 'ABCDE' REGEXP '.*' |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)
```

Let's see one more example:

```
mysql> SELECT 'new*\n*line' REGEXP 'new\\*\\.\\*line';
+-----+
| 'new*\n*line' REGEXP 'new\\*\\.\\*line' |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)
```

REPEATstr, count

Returns a string consisting of the string str repeated count times. If count is less than 1, returns an empty string. Returns NULL if str or count are NULL.

```
mysql> SELECT REPEAT('MySQL', 3);
+-----+
| REPEAT('MySQL', 3) |
+-----+
| MySQLMySQLMySQL |
+-----+
1 row in set (0.00 sec)
```

REPLACEstr, from_str, to_str

Returns the string str with all occurrences of the string from_str replaced by the string to_str. REPLACE performs a case-sensitive match when searching for from_str.

```
mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');
+-----+
| REPLACE('www.mysql.com', 'w', 'Ww') |
+-----+
```

```
+-----+
| wwwwww.mysql.com |
+-----+
1 row in set (0.00 sec)
```

REVERSEstr

Returns the string str with the order of the characters reversed.

```
mysql> SELECT REVERSE('abcd');
+-----+
| REVERSE('abcd') |
+-----+
| dcba |
+-----+
1 row in set (0.00 sec)
```

RIGHTstr, len

Returns the rightmost len characters from the string str, or NULL if any argument is NULL.

```
mysql> SELECT RIGHT('foobarbar', 4);
+-----+
| RIGHT('foobarbar', 4) |
+-----+
| rbar |
+-----+
1 row in set (0.00 sec)
```

RPADstr, len, padstr

Returns the string str, right-padded with the string padstr to a length of len characters. If str is longer than len, the return value is shortened to len characters.

```
mysql> SELECT RPAD('hi',5,'?');
+-----+
| RPAD('hi',5,'?') |
+-----+
| hi??? |
+-----+
1 row in set (0.00 sec)
```

RTRIMstr

Returns the string str with trailing space characters removed.

```
mysql> SELECT RTRIM('barbar ');
+-----+
| RTRIM('barbar ') |
+-----+
| barbar |
+-----+
1 row in set (0.00 sec)
```

SOUNDEXstr

Returns a soundex string from str. Two strings that sound almost the same should have identical soundex strings. A standard soundex string is four characters long, but the SOUNDEX function returns an arbitrarily long string. You can use SUBSTRING on the result to get a standard soundex string. All non-alphabetic characters in str are ignored. All international alphabetic characters outside the A-Z range are treated as vowels.

```
mysql> SELECT SOUNDEX('Hello');
+-----+
```

```
| SOUNDEX('Hello') |
+-----+
| H400 |
+-----+
1 row in set (0.00 sec)
```

expr1 SOUNDS LIKE expr2

This is the same as $\text{SOUNDEX}_{\text{expr1}} = \text{SOUNDEX}_{\text{expr2}}$.

SPACE*N*

```
mysql> SELECT SPACE(6);
+-----+
| SELECT SPACE(6) |
+-----+
| '      ' |
+-----+
1 row in set (0.00 sec)
```

STRCMP*str1, str2*

Compares two strings and returns 0 if both strings are equal, it returns -1 if the first argument is smaller than the second according to the current sort order otherwise it returns 1.

```
mysql> SELECT STRCMP('MOHD', 'MOHD');
+-----+
| STRCMP('MOHD', 'MOHD') |
+-----+
| 0 |
+-----+
1 row in set (0.00 sec)
```

Another example is:

```
mysql> SELECT STRCMP('AMOHD', 'MOHD');
+-----+
| STRCMP('AMOHD', 'MOHD') |
+-----+
| -1 |
+-----+
1 row in set (0.00 sec)
```

Let's see one more example:

```
mysql> SELECT STRCMP('MOHD', 'AMOHD');
+-----+
| STRCMP('MOHD', 'AMOHD') |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)
```

SUBSTRING*str, pos*

SUBSTRING*strFROMpos*

SUBSTRING*str, pos, len*

SUBSTRING*strFROMposFORlen*

The forms without a len argument return a substring from string str starting at position pos. The forms with a len argument return a substring len characters long from string str, starting at position pos. The forms that use FROM are standard SQL syntax. It is also possible to use a negative

value for pos. In this case, the beginning of the substring is pos characters from the end of the string, rather than the beginning. A negative value may be used for pos in any of the forms of this function.

```
mysql> SELECT SUBSTRING('Quadratically',5);
+-----+
| SSUBSTRING('Quadratically',5) |
+-----+
| ratically                     |
+-----+
1 row in set (0.00 sec)

mysql> SELECT SUBSTRING('foobarbar' FROM 4);
+-----+
| SUBSTRING('foobarbar' FROM 4) |
+-----+
| barbar                        |
+-----+
1 row in set (0.00 sec)

mysql> SELECT SUBSTRING('Quadratically',5,6);
+-----+
| SUBSTRING('Quadratically',5,6) |
+-----+
| ratica                          |
+-----+
1 row in set (0.00 sec)
```

SUBSTRING_INDEX(*str*,*delim*,*count*)

Returns the substring from string *str* before *count* occurrences of the delimiter *delim*. If *count* is positive, everything to the left of the final delimiter *countingfromtheleft* is returned. If *count* is negative, everything to the right of the final delimiter *countingfromtheright* is returned. SUBSTRING_INDEX performs a case-sensitive match when searching for *delim*.

```
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);
```

SUBSTRING_INDEX('www.mysql.com', '.', 2)
www.mysql

```
1 row in set (0.00 sec)
```

TRIM[*BOTH* | *LEADING* | *TRAILING*[*remstr*]*FROM*]*str*

TRIM*[remstrFROM]str*

Returns the string `str` with all `remstr` prefixes or suffixes removed. If none of the specifiers `BOTH`, `LEADING`, or `TRAILING` is given, `BOTH` is assumed. `remstr` is optional and, if not specified, spaces are removed.

```
mysql> SELECT TRIM(' bar ');  
+-----+  
| TRIM(' bar ') |  
+-----+  
| bar           |  
+-----+  
  
1 row in set (0.00 sec)  
  
mysql> SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');  
+-----+  
| TRIM(LEADING 'x' FROM 'xxxbarxxx') |  
+-----+  
| barxxx                               |  
+-----+  
  
1 row in set (0.00 sec)
```

```
mysql> SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');
+-----+
| TRIM(BOTH 'x' FROM 'xxxbarxxx') |
+-----+
| bar                               |
+-----+
1 row in set (0.00 sec)

mysql> SELECT TRIM(TRAILING 'xyz' FROM 'barxyz');
+-----+
| TRIM(TRAILING 'xyz' FROM 'barxyz') |
+-----+
| barx                               |
+-----+
1 row in set (0.00 sec)
```

UCASE_{str}

UCASE is a synonym for UPPER.

UNHEX_{str}

Performs the inverse operation of HEX_{str}. That is, it interprets each pair of hexadecimal digits in the argument as a number and converts it to the character represented by the number. The resulting characters are returned as a binary string.

```
mysql> SELECT UNHEX('4D7953514C');
+-----+
| UNHEX('4D7953514C') |
+-----+
| MySQL                |
+-----+
1 row in set (0.00 sec)
```

The characters in the argument string must be legal hexadecimal digits: '0' .. '9', 'A' .. 'F', 'a' .. 'f'. If UNHEX encounters any non-hexadecimal digits in the argument, it returns NULL.

UPPER_{str}

Returns the string str with all characters changed to uppercase according to the current character set mapping.

```
mysql> SELECT UPPER('Allah-hus-samad');
+-----+
| UPPER('Allah-hus-samad') |
+-----+
| ALLAH-HUS-SAMAD         |
+-----+
1 row in set (0.00 sec)
```

For more information, check [MySQL Official Website - String Functions](#)

Processing math: 100%