## MySQL текстові функції

|  |  |
| --- | --- |
| **Функція** | **Опис** |
| [ASCII](https://www.w3schools.com/sql/func_mysql_ascii.asp) | Повертає ASCII значення символу |
| [CHAR\_LENGTH](https://www.w3schools.com/sql/func_mysql_char_length.asp) | Повертає довжину рядку у символах |
| [CHARACTER\_LENGTH](https://www.w3schools.com/sql/func_mysql_character_length.asp) | Повертає довжину рядка у символах |
| [CONCAT](https://www.w3schools.com/sql/func_mysql_concat.asp) | Об’єднання кількох виразів разом |
| [CONCAT\_WS](https://www.w3schools.com/sql/func_mysql_concat_ws.asp) | Об’єднання кількох виразів за допомогою сепаратора |
| [FIELD](https://www.w3schools.com/sql/func_mysql_field.asp) | Повертає позицію змінної у списку змінних |
| [FIND\_IN\_SET](https://www.w3schools.com/sql/func_mysql_find_in_set.asp) | Повертає позицію рядка у списку рядків |
| [FORMAT](https://www.w3schools.com/sql/func_mysql_format.asp) | Форматує число за форматом "#,###,###.##", а також округлює його з певною точністю |
| [INSERT](https://www.w3schools.com/sql/func_mysql_insert.asp) | Вставляє певну кількість символів рядок у певну позицію іншого рядка |
| [INSTR](https://www.w3schools.com/sql/func_mysql_instr.asp) | Повертає перше входження рядка у інший рядок |
| [LCASE](https://www.w3schools.com/sql/func_mysql_lcase.asp) | Конвертує рядок у нижній регістр |
| [LEFT](https://www.w3schools.com/sql/func_mysql_left.asp) | Повертає перші зліва символи рядка |
| [LENGTH](https://www.w3schools.com/sql/func_mysql_length.asp) | Повертає довжина рядка у байтах |
| [LOCATE](https://www.w3schools.com/sql/func_mysql_locate.asp) | Повертає перше входження рядку у рядок |
| [LOWER](https://www.w3schools.com/sql/func_mysql_lower.asp) | Повертає рядок, конвертований до нижнього регістру  |
| [LPAD](https://www.w3schools.com/sql/func_mysql_lpad.asp) | Заповнює рядок зліва іншим рядком певної довжини |
| [LTRIM](https://www.w3schools.com/sql/func_mysql_ltrim.asp) | Вилучає пропуски у лівій частині рядка |
| [MID](https://www.w3schools.com/sql/func_mysql_mid.asp) | Виділяє рядок з певної позиції поточного рядка |
| [POSITION](https://www.w3schools.com/sql/func_mysql_position.asp) | Повертає перше входження рядка у поточний рядок |
| [REPEAT](https://www.w3schools.com/sql/func_mysql_repeat.asp) | Повертає рядок, утворений певним повтором певного рядка |
| [REPLACE](https://www.w3schools.com/sql/func_mysql_replace.asp) | Повертає всі входження рядка у поточний рядок |
| [REVERSE](https://www.w3schools.com/sql/func_mysql_reverse.asp) | Повертає реверсований рядок |
| [RIGHT](https://www.w3schools.com/sql/func_mysql_right.asp) | Виділяє певну кількість символів з правого боку поточного рядка |
| [RPAD](https://www.w3schools.com/sql/func_mysql_rpad.asp) | Заповнює рядок з правого боку Іншим рядком |
| [RTRIM](https://www.w3schools.com/sql/func_mysql_rtrim.asp) | Вилучає всі пропуски з поточного рядка |
| [SPACE](https://www.w3schools.com/sql/func_mysql_space.asp) | Повертає рядок певної довжини, утворений пропусками |
| [STRCMP](https://www.w3schools.com/sql/func_mysql_strcmp.asp) | Порівняння двох рядків |
| [SUBSTR](https://www.w3schools.com/sql/func_mysql_substr.asp) | Повертає з поточного рядка рядок повної довжини |
| [SUBSTRING](https://www.w3schools.com/sql/func_mysql_substring.asp) | Повертає текст з поточного рядка, що починається з певної позиції  |
| [SUBSTRING\_INDEX](https://www.w3schools.com/sql/func_mysql_substring_index.asp) | Повертає рядок перед певною позицією або символом-роздільником  |
| [TRIM](https://www.w3schools.com/sql/func_mysql_trim.asp) | Вилучає пропуски з правого та лівого боку рядка |
| [UCASE](https://www.w3schools.com/sql/func_mysql_ucase.asp) | Конвертує рядок у верхній регістр |
| [UPPER](https://www.w3schools.com/sql/func_mysql_upper.asp) | Конвертує рядок у верхній регістр |

## Чисельні функції

|  |  |
| --- | --- |
| **Функція** | **Опис** |
| [ABS](https://www.w3schools.com/sql/func_mysql_abs.asp) | |x| |
| [ACOS](https://www.w3schools.com/sql/func_mysql_acos.asp) |  arccos(x) |
| [ASIN](https://www.w3schools.com/sql/func_mysql_asin.asp) | arcsin(x) |
| [ATAN](https://www.w3schools.com/sql/func_mysql_atan.asp)(x) | arctg(x) |
| [ATAN2](https://www.w3schools.com/sql/func_mysql_atan2.asp)(x,y) | arctg(x/y) |
| [AVG](https://www.w3schools.com/sql/func_mysql_avg.asp) | Середнє значення виразу |
| [CEIL](https://www.w3schools.com/sql/func_mysql_ceil.asp)(x) | Найменше ціле число >= х |
| [CEILING](https://www.w3schools.com/sql/func_mysql_ceiling.asp)(x) | Найменше ціле число >= х |
| [COS](https://www.w3schools.com/sql/func_mysql_cos.asp)(x) | cos(x) |
| [COT](https://www.w3schools.com/sql/func_mysql_cot.asp)(x) | ctg(x) |
| [COUNT](https://www.w3schools.com/sql/func_mysql_count.asp)(x) | Кількцість записів певного запиту |
| [DEGREES](https://www.w3schools.com/sql/func_mysql_degrees.asp)(х) | Перетворює х у градуси |
| [DIV](https://www.w3schools.com/sql/func_mysql_div.asp)(х, у) | Ціле ділення div(2,5)=0 div(5,2)=2 |
| [EXP](https://www.w3schools.com/sql/func_mysql_exp.asp)(х) |  EXP(X) |
| [FLOOR](https://www.w3schools.com/sql/func_mysql_floor.asp) | Найбільше ціле число <= х |
| [GREATEST](https://www.w3schools.com/sql/func_mysql_greatest.asp) | Найбільша величина зі списку |
| [LEAST](https://www.w3schools.com/sql/func_mysql_least.asp) | Нійменша величина зі списку |
| [LN](https://www.w3schools.com/sql/func_mysql_ln.asp)(x) | ln(x) |
| [LOG](https://www.w3schools.com/sql/func_mysql_log.asp)(x,b) | Натуральний логарифм числа, або логарифм з певним Базисом |
| [LOG10](https://www.w3schools.com/sql/func_mysql_log10.asp)(x) | lg(x) |
| [LOG2](https://www.w3schools.com/sql/func_mysql_log2.asp)(x) | lb(x) двійковий логарифм числа lb(16)=4 lb(2)=1 |
| [MAX](https://www.w3schools.com/sql/func_mysql_max.asp) |  Максимальне значення елементів множини |
| [MIN](https://www.w3schools.com/sql/func_mysql_min.asp) | Мінімальне значення елементів множини |
| [MOD](https://www.w3schools.com/sql/func_mysql_mod.asp) | Залишок від ділення двох чисел |
| [PI](https://www.w3schools.com/sql/func_mysql_pi.asp) |  Число 3.14159265358… |
| [POW](https://www.w3schools.com/sql/func_mysql_pow.asp)(x,y) | xy |
| [POWER](https://www.w3schools.com/sql/func_mysql_power.asp)(x,y) | xy  |
| [RADIANS](https://www.w3schools.com/sql/func_mysql_radians.asp)(degree) | Конвертує градуси у радіани |
| [RAND](https://www.w3schools.com/sql/func_mysql_rand.asp) | Повертає випадкове число |
| [ROUND](https://www.w3schools.com/sql/func_mysql_round.asp)(х,п) | Округлення числа до певної точності |
| [SIGN](https://www.w3schools.com/sql/func_mysql_sign.asp)(х) | Знак числа  |
| [SIN](https://www.w3schools.com/sql/func_mysql_sin.asp)(х) | sin (x) |
| [SQRT](https://www.w3schools.com/sql/func_mysql_sqrt.asp)(х) | Квадратний корінь числа |
| [SUM](https://www.w3schools.com/sql/func_mysql_sum.asp) | Сума елементів множини |
| [TAN](https://www.w3schools.com/sql/func_mysql_tan.asp) (х) | tg(x)  |
| [TRUNCATE](https://www.w3schools.com/sql/func_mysql_truncate.asp) | Залишення специфікованої кількості десятичних цифр |

## Функції дати

|  |  |
| --- | --- |
| **Функція** | **Опис** |
| [ADDDATE](https://www.w3schools.com/sql/func_mysql_adddate.asp) | Додає час / інтервал до дати та формує нову дату |
| [ADDTIME](https://www.w3schools.com/sql/func_mysql_addtime.asp) | Додає інтервал до часу/дати та формує нову дату |
| [CURDATE](https://www.w3schools.com/sql/func_mysql_curdate.asp) | Повертає поточну дату |
| [CURRENT\_DATE](https://www.w3schools.com/sql/func_mysql_current_date.asp) | Повертає поточну дату |
| [CURRENT\_TIME](https://www.w3schools.com/sql/func_mysql_current_time.asp) | Повертає поточний час |
| [CURRENT\_TIMESTAMP](https://www.w3schools.com/sql/func_mysql_current_timestamp.asp) | Повертає поточний час та дату |
| [CURTIME](https://www.w3schools.com/sql/func_mysql_curtime.asp) | Повертає поточний час |
| [DATE](https://www.w3schools.com/sql/func_mysql_date.asp) | Повертає дату виразу |
| [DATEDIFF](https://www.w3schools.com/sql/func_mysql_datediff.asp) | Кількість днів між двома датами |
| [DATE\_ADD](https://www.w3schools.com/sql/func_mysql_date_add.asp) | Додає час /інтервал до дати та формує нову дату |
| [DATE\_FORMAT](https://www.w3schools.com/sql/func_mysql_date_format.asp) | Форматування дати |
| [DATE\_SUB](https://www.w3schools.com/sql/func_mysql_date_sub.asp) | Різниця інтервалів та формування нової дати |
| [DAY](https://www.w3schools.com/sql/func_mysql_day.asp) | Номер дня місяця для дати |
| [DAYNAME](https://www.w3schools.com/sql/func_mysql_dayname.asp) | Назва дня тижня для дати |
| [DAYOFMONTH](https://www.w3schools.com/sql/func_mysql_dayofmonth.asp) | Номер дня місяця для дати |
| [DAYOFWEEK](https://www.w3schools.com/sql/func_mysql_dayofweek.asp) | Назва дня тижня для дати |
| [DAYOFYEAR](https://www.w3schools.com/sql/func_mysql_dayofyear.asp) | Номер дня у році для певної дати |
| [EXTRACT](https://www.w3schools.com/sql/func_mysql_extract.asp) | Повертає певну частину дати |
| [FROM\_DAYS](https://www.w3schools.com/sql/func_mysql_from_days.asp) | Повертає дату з числа  |
| [HOUR](https://www.w3schools.com/sql/func_mysql_hour.asp) | Повертає години з дати |
| [LAST\_DAY](https://www.w3schools.com/sql/func_mysql_last_day.asp) | Останній день місяця для дати  |
| [LOCALTIME](https://www.w3schools.com/sql/func_mysql_localtime.asp) | Локальний час та дата |
| [LOCALTIMESTAMP](https://www.w3schools.com/sql/func_mysql_localtimestamp.asp) | Локальний час та дата |
| [MAKEDATE](https://www.w3schools.com/sql/func_mysql_makedate.asp) | Створює дату з року та номеру дня |
| [MAKETIME](https://www.w3schools.com/sql/func_mysql_maketime.asp) | Створює час з годин, хвилин, секунд |
| [MICROSECOND](https://www.w3schools.com/sql/func_mysql_microsecond.asp) | Повертає мікросекунди з дати |
| [MINUTE](https://www.w3schools.com/sql/func_mysql_minute.asp) | Повертає хвилини з часу/ дати |
| [MONTH](https://www.w3schools.com/sql/func_mysql_month.asp) | Повертає місяць з дати |
| [MONTHNAME](https://www.w3schools.com/sql/func_mysql_monthname.asp) | Повертає назву місяця для дати |
| [NOW](https://www.w3schools.com/sql/func_mysql_now.asp) | Повертає поточну дату та час |
| [PERIOD\_ADD](https://www.w3schools.com/sql/func_mysql_period_add.asp) | Додає кількість місяців до періоду |
| [PERIOD\_DIFF](https://www.w3schools.com/sql/func_mysql_period_diff.asp) | Повертає різницю двох періодів |
| [QUARTER](https://www.w3schools.com/sql/func_mysql_quarter.asp) | Квартал року для дати |
| [SECOND](https://www.w3schools.com/sql/func_mysql_second.asp) | Секунди з часу або дати |
| [SEC\_TO\_TIME](https://www.w3schools.com/sql/func_mysql_sec_to_time.asp) | Формування дати з кількості секунд |
| [STR\_TO\_DATE](https://www.w3schools.com/sql/func_mysql_str_to_date.asp) | Перетворює рядок на дату |
| [SUBDATE](https://www.w3schools.com/sql/func_mysql_subdate.asp) | Різниця двох інтервалів часу/дати та формування нової величини |
| [SUBTIME](https://www.w3schools.com/sql/func_mysql_subtime.asp) | Різниця двох інтервалів та формування нового часу |
| [SYSDATE](https://www.w3schools.com/sql/func_mysql_sysdate.asp) | Поточна дата та час |
| [TIME](https://www.w3schools.com/sql/func_mysql_time.asp) | Повертає час з а виразу  |
| [TIME\_FORMAT](https://www.w3schools.com/sql/func_mysql_time_format.asp) | Форматування часу за форматом |
| [TIME\_TO\_SEC](https://www.w3schools.com/sql/func_mysql_time_to_sec.asp) | Конвертування часу в секунди |
| [TIMEDIFF](https://www.w3schools.com/sql/func_mysql_timediff.asp) | Різниця двох виразів часу та дати |
| [TIMESTAMP](https://www.w3schools.com/sql/func_mysql_timestamp.asp) | Час та дата, що утворюються з дати або дати та часу |
| [TO\_DAYS](https://www.w3schools.com/sql/func_mysql_to_days.asp) | Кількість днів між поточною датою та "0000-00-00" |
| [WEEK](https://www.w3schools.com/sql/func_mysql_week.asp) | Номер тижня з дати |
| [WEEKDAY](https://www.w3schools.com/sql/func_mysql_weekday.asp) | День тижня для певної дати |
| [WEEKOFYEAR](https://www.w3schools.com/sql/func_mysql_weekofyear.asp) | Номер тижня з певної дати  |
| [YEAR](https://www.w3schools.com/sql/func_mysql_year.asp) | Рік з дати |
| [YEARWEEK](https://www.w3schools.com/sql/func_mysql_yearweek.asp) | Рік та номер тижня для певної дати |

## Спеціальні функції

|  |  |
| --- | --- |
| **Функція** | **Опис** |
| [BIN](https://www.w3schools.com/sql/func_mysql_bin.asp) | Повертає бінарне подання числа |
| [BINARY](https://www.w3schools.com/sql/func_mysql_binary.asp) | Конвертує величину у вінарний рядок |
| [CASE](https://www.w3schools.com/sql/func_mysql_case.asp) | Перевіряє ряд умов та повертає значення, коли певна умова виконується |
| [CAST](https://www.w3schools.com/sql/func_mysql_cast.asp)  | Конвернтує величину одного типа у інший тип |
| [COALESCE](https://www.w3schools.com/sql/func_mysql_coalesce.asp) | Перше не null значення у списку |
| [CONNECTION\_ID](https://www.w3schools.com/sql/func_mysql_connection_id.asp) | Повертає ID унікального з’єднання |
| [CONV](https://www.w3schools.com/sql/func_mysql_conv.asp) | Конвертує число з однієї системи у іншу |
| [CONVERT](https://www.w3schools.com/sql/func_mysql_convert.asp) | Конвертує величину у певний тип даних або множину симовлів |
| [CURRENT\_USER](https://www.w3schools.com/sql/func_mysql_current_user.asp) | Повертає користувача та хостинг |
| [DATABASE](https://www.w3schools.com/sql/func_mysql_database.asp) | Повертає назву поточної БД |
| [IF](https://www.w3schools.com/sql/func_mysql_if.asp) | Повертає значення, якщо умова if TRUE, інакше FALSE |
| [IFNULL](https://www.w3schools.com/sql/func_mysql_ifnull.asp) | Повертає специфіковане значення якщо вираз NULL, Інакше повертається вираз |
| [ISNULL](https://www.w3schools.com/sql/func_mysql_isnull.asp) | Повертає 1 або 0 якщо вираз NULL |
| [LAST\_INSERT\_ID](https://www.w3schools.com/sql/func_mysql_last_insert_id.asp) | Повертає AUTO\_INCREMENT id для останнього рядка, що буде додано у таблицю  |
| [NULLIF](https://www.w3schools.com/sql/func_mysql_nullif.asp) | Порівнює два вирази та повертає NULL якщо вони еквівалентні. Інакше повертається перший вираз |
| [SESSION\_USER](https://www.w3schools.com/sql/func_mysql_session_user.asp) | Поточний користувач та хостинг |
| [SYSTEM\_USER](https://www.w3schools.com/sql/func_mysql_system_user.asp) | Поточний користувач та хостинг |
| [USER](https://www.w3schools.com/sql/func_mysql_user.asp) | Поточний користувач та хостинг |
| [VERSION](https://www.w3schools.com/sql/func_mysql_version.asp) | Поточна версія MySQL  |

# ASCII()

The ASCII() function returns the ASCII value for the specific character.

## Syntax

ASCII(character)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| character | Required. The character to return the ASCII value for. If more than one character is entered, it will only return the value for the first character |

# CHAR\_LENGTH()

The CHAR\_LENGTH() function return the length of a string (in characters).

**Note:** This function is equal to the [CHARACTER\_LENGTH()](https://www.w3schools.com/sql/func_mysql_character_length.asp) function.

CHAR\_LENGTH(string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to count the length for |

## More Examples

Return the length of the text in the "CustomerName" column:

SELECT CHAR\_LENGTH(CustomerName) AS LengthOfName
FROM Customers;

# CONCAT()

Add several strings together:

SELECT CONCAT("SQL ", "Tutorial ", "is ", "fun!") AS ConcatenatedString;

The CONCAT() function adds two or more expressions together.

**Note:** Also look at the [CONCAT\_WS()](https://www.w3schools.com/sql/func_mysql_concat_ws.asp) function.

CONCAT(expression1, expression2, expression3,...)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression1,expression2,expression3,etc. | Required. The expressions to add together.**Note:** If any of the expressions is a NULL value, it returns NULL |

# CONCAT\_WS()

### **Example**

Add several expressions together, and add a "-" separator between them:

SELECT CONCAT\_WS("-", "SQL", "Tutorial", "is", "fun!") AS ConcatenatedString;

The CONCAT\_WS() function adds two or more expressions together with a separator.

**Note:** Also look at the [CONCAT()](https://www.w3schools.com/sql/func_mysql_concat.asp) function.

CONCAT\_WS(separator, expression1, expression2, expression3,...)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| separator | Required. The separator to add between each of the expressions. If separator is NULL, this function returns NULL |
| expression1,expression2,expression3,etc. | Required. The expressions to add together. An expression with a NULL value will be skipped |

# FIELD()

Return the index position of  "q" in the string list:

SELECT FIELD("q", "s", "q", "l");

The FIELD() function returns the index position of a value in a list of values.

This function performs a case-insensitive search.

**Note:** If the specified value is not found in the list of values, this function will return 0. If value is NULL, this function will return 0.

FIELD(value, val1, val2, val3, ...)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| value | Required. The value to search for in the list |
| val1, val2, val3, .... | Required. The list of values to search |

## More Examples

Return the index position of "c" in the string list:

SELECT FIELD("c", "a", "b");

Return the index position of "Q" in the string list:

SELECT FIELD("Q", "s", "q", "l");

Return the index position of 5 in the numeric list:

SELECT FIELD(5, 0, 1, 2, 3, 4, 5);

# FIND\_IN\_SET()

Search for "q" within the list of strings:

SELECT FIND\_IN\_SET("q", "s,q,l");

The FIND\_IN\_SET() function returns the position of a string within a list of strings.

FIND\_IN\_SET(string, string\_list)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to search for |
| string\_list | Required. The list of string values to be searched (separated by commas) |

* If string is not found in string\_list, this function returns 0
* If string or string\_list is NULL, this function returns NULL
* If string\_list is an empty string (""), this function returns 0

Search for "a" within the list of strings:

SELECT FIND\_IN\_SET("a", "s,q,l");

Search for "q" within the list of strings (string list is NULL):

SELECT FIND\_IN\_SET("q", null);

# FORMAT()

Format the number as "#,###,###.##" (and round with two decimal places):

SELECT FORMAT(250500.5634, 2);

The FORMAT() function formats a number to a format like "#,###,###.##", rounded to a specified number of decimal places, then it returns the result as a string.

FORMAT(number, decimal\_places)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. The number to be formatted |
| decimal\_places | Required. The number of decimal places for number. If this parameter is 0, this function returns a string with no decimal places |

Format the number as a format of "#,###,###.##" (and round with 0 decimal places):

SELECT FORMAT(250500.5634, 0);

# INSERT()

Insert the string "Example" into the string "W3Schools.com". Replace the first nine characters:

SELECT INSERT("W3Schools.com", 1, 9, "Example");

The INSERT() function inserts a string within a string at the specified position and for a certain number of characters.

INSERT(string, position, number, string2)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string that will be modified |
| position | Required. The position where to insert string2 |
| number | Required. The number of characters to replace |
| string2 | Required. The string to insert into string |

* If position is outside the length of string, this function returns string
* If number is higher than the length of the rest of the string, this function replaces string from position until the end of string

Insert the string "no" into the string "W3Schools.com". Replace three characters, starting from position 11:

SELECT INSERT("W3Schools.com", 11, 3, "no");

# INSTR()

Search for "3" in string "W3Schools.com", and return position:

SELECT INSTR("W3Schools.com", "3") AS MatchPosition;

The INSTR() function returns the position of the first occurrence of a string in another string.

This function performs a case-insensitive search.

INSTR(string1, string2)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string1 | Required. The string that will be searched |
| string2 | Required. The string to search for in string1. If string2 is not found, this function returns 0 |

Search for "COM" in string "W3Schools.com", and return position:

SELECT INSTR("W3Schools.com", "COM") AS MatchPosition;

# LCASE()

Convert the text to lower-case:

SELECT LCASE("SQL Tutorial is FUN!");

The LCASE() function converts a string to lower-case.

**Note:** The [LOWER()](https://www.w3schools.com/sql/func_mysql_lower.asp) function is a synonym for the LCASE() function.

LCASE(text)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| text | Required. The string to convert |

Convert the text in "CustomerName" to lower-case:

SELECT LCASE(CustomerName) AS LowercaseCustomerName
FROM Customers;

# LEFT()

Extract 3 characters from a string (starting from left):

SELECT LEFT("SQL Tutorial", 3) AS ExtractString;

The LEFT() function extracts a number of characters from a string (starting from left).

**Tip:** Also look at the [RIGHT()](https://www.w3schools.com/sql/func_mysql_right.asp) function.

LEFT(string, number\_of\_chars)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to extract from |
| number\_of\_chars | Required. The number of characters to extract. If this parameter is larger than the number of characters in string, this function will return string |

Extract 5 characters from the text in the "CustomerName" column (starting from left):

SELECT LEFT(CustomerName, 5) AS ExtractString
FROM Customers;

# LENGTH()

Return the length of the string, in bytes:

SELECT LENGTH("SQL Tutorial") AS LengthOfString;

The LENGTH() function returns the length of a string (in bytes).

LENGTH(string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to count the length for |

Return the length of the text in the "CustomerName" column, in bytes:

SELECT LENGTH(CustomerName) AS LengthOfName
FROM Customers;

# LOCATE()

Search for "3" in string "W3Schools.com", and return position:

SELECT LOCATE("3", "W3Schools.com") AS MatchPosition;

The LOCATE() function returns the position of the first occurrence of a substring in a string.

If the substring is not found within the original string, this function returns 0.

This function performs a case-insensitive search.

**Note:** This function is equal to the [POSITION()](https://www.w3schools.com/sql/func_mysql_position.asp) function.

LOCATE(substring, string, start)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| substring | Required. The substring to search for in string |
| string | Required. The string that will be searched |
| start | Optional. The starting position for the search. Position 1 is default |

Search for "com" in string "W3Schools.com" (start at position 3), and return position:

SELECT LOCATE("com", "W3Schools.com", 3) AS MatchPosition;

Search for "a" in CustomerName column, and return position:

SELECT LOCATE("a", CustomerName)
FROM Customers;

# LOWER()

Convert the text to lower-case:

SELECT LOWER("SQL Tutorial is FUN!");

The LOWER() function converts a string to lower-case.

**Note:** The [LCASE()](https://www.w3schools.com/sql/func_mysql_lcase.asp) function is equal to the LOWER() function.

LOWER(text)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| text | Required. The string to convert |

Convert the text in "CustomerName" to lower-case:

SELECT LOWER(CustomerName) AS LowercaseCustomerName
FROM Customers;

# LPAD()

Left-pad the string with "ABC", to a total length of 20:

SELECT LPAD("SQL Tutorial", 20, "ABC");

The LPAD() function left-pads a string with another string, to a certain length.

**Note:** Also look at the [RPAD()](https://www.w3schools.com/sql/func_mysql_rpad.asp) function.

## Syntax

LPAD(string, length, lpad\_string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The original string. If the length of the original string is larger than the length parameter, this function removes the overfloating characters from string |
| length | Required. The length of the string after it has been left-padded |
| lpad\_string | Required. The string to left-pad to string |

Left-pad the text in "CustomerName" with "ABC", to a total length of 30:

SELECT LPAD(CustomerName, 30, "ABC") AS LeftPadCustomerName
FROM Customers;

# LTRIM()

Remove leading spaces from a string:

SELECT LTRIM("     SQL Tutorial") AS LeftTrimmedString;

The LTRIM() function removes leading spaces from a string.

LTRIM(string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to remove leading spaces from |

# MID()

Extract a substring from a string (start at position 5, extract 3 characters):

SELECT MID("SQL Tutorial", 5, 3) AS ExtractString;

The MID() function extracts a substring from a string (starting at any position).

**Note:** The MID() and [SUBSTR()](https://www.w3schools.com/sql/func_mysql_substr.asp) functions equals the [SUBSTRING()](https://www.w3schools.com/sql/func_mysql_substring.asp) function.

MID(string, start, length)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to extract from |
| start | Required. The start position. Can be both a positive or negative number. If it is a positive number, this function extracts from the beginning of the string. If it is a negative number, this function extracts from the end of the string |
| length | Required. The number of characters to extract |

Extract a substring from the text in a column (start at position 2, extract 5 characters):

SELECT MID(CustomerName, 2, 5) AS ExtractString
FROM Customers;

Extract a substring from a string (start from the end, at position -5, extract 5 characters):

SELECT MID("SQL Tutorial", -5, 5) AS ExtractString;

# POSITION()

Search for "3" in string "W3Schools.com", and return position:

SELECT POSITION("3" IN "W3Schools.com") AS MatchPosition;

The POSITION() function returns the position of the first occurrence of a substring in a string.

If the substring is not found within the original string, this function returns 0.

This function performs a case-insensitive search.

**Note:** The [LOCATE()](https://www.w3schools.com/sql/func_mysql_locate.asp) function is equal to the POSITION() function.

POSITION(substring IN string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| substring | Required. The substring to search for in string |
| string | Required. The original string that will be searched |

Search for "COM" in string "W3Schools.com", and return position:

SELECT POSITION("COM" IN "W3Schools.com") AS MatchPosition;

# REPEAT()

Repeat a string 3 times:

SELECT REPEAT("SQL Tutorial", 3);

The REPEAT() function repeats a string as many times as specified.

REPEAT(string, number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to repeat |
| number | Required. The number of times to repeat the string |

Repeat the text in CustomerName 2 times:

SELECT REPEAT(CustomerName, 2)
FROM Customers;

Repeat the string 0 times:

SELECT REPEAT("SQL Tutorial", 0);

# REPLACE()

Replace "SQL" with "HTML":

SELECT REPLACE("SQL Tutorial", "SQL", "HTML");

The REPLACE() function replaces all occurrences of a substring within a string, with a new substring.

**Note:** This function performs a case-sensitive replacement.

REPLACE(string, from\_string, new\_string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The original string |
| from\_string | Required. The substring to be replaced |
| new\_string | Required. The new replacement substring |

Replace "X" with "M":

SELECT REPLACE("XYZ FGH XYZ", "X", "M");

Replace "X" with "m":

SELECT REPLACE("XYZ FGH XYZ", "X", "m");

Replace "x" with "m":

SELECT REPLACE("XYZ FGH XYZ", "x", "m");

# REVERSE()

Reverse a string:

SELECT REVERSE("SQL Tutorial");

The REVERSE() function reverses a string and returns the result.

REVERSE(string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to reverse |

Reverse the text in CustomerName:

SELECT REVERSE(CustomerName)
FROM Customers;

# RIGHT()

Extract 4 characters from a string (starting from right):

SELECT RIGHT("SQL Tutorial is cool", 4) AS ExtractString;

The RIGHT() function extracts a number of characters from a string (starting from right).

**Tip:** Also look at the [LEFT()](https://www.w3schools.com/sql/func_mysql_left.asp) function.

RIGHT(string, number\_of\_chars)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to extract from |
| number\_of\_chars | Required. The number of characters to extract. If this parameter is larger than the number of characters in string, this function will return string |

Extract 5 characters from the text in the "CustomerName" column (starting from right):

SELECT RIGHT(CustomerName, 5) AS ExtractString
FROM Customers;

# RPAD()

Right-pad the string with "ABC", to a total length of 20:

SELECT RPAD("SQL Tutorial", 20, "ABC");

The RPAD() function right-pads a string with another string, to a certain length.

**Note:** Also look at the [LPAD()](https://www.w3schools.com/sql/func_mysql_lpad.asp) function.

RPAD(string, length, rpad\_string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The original string. If the length of the original string is larger than the length parameter, this function removes the overfloating characters from string |
| length | Required. The length of the string after it has been right-padded |
| rpad\_string | Required. The string to right-pad to string |

Right-pad the text in "CustomerName" with "ABC", to a total length of 30:

SELECT RPAD(CustomerName, 30, "ABC") AS RightPadCustomerName
FROM Customers;

# RTRIM()

Remove trailing spaces from a string:

SELECT RTRIM("SQL Tutorial     ") AS RightTrimmedString;

The RTRIM() function removes trailing spaces from a string.

RTRIM(string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to remove trailing spaces from |

# SPACE()

Return a string with 10 space characters:

SELECT SPACE(10);

The SPACE() function returns a string of the specified number of space characters.

SPACE(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. The number of space characters to return |

# STRCMP()

Compare two strings:

SELECT STRCMP("SQL Tutorial", "SQL Tutorial");

The STRCMP() function compares two strings.

STRCMP(string1, string2)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string1, string2 | Required. The two strings to be compared |

* If string1 = string2, this function returns 0
* If string1 < string2, this function returns -1
* If string1 > string2, this function returns 1

Compare two strings:

SELECT STRCMP("SQL Tutorial", "HTML Tutorial");

# SUBSTR()

Extract a substring from a string (start at position 5, extract 3 characters):

SELECT SUBSTR("SQL Tutorial", 5, 3) AS ExtractString;

The SUBSTR() function extracts a substring from a string (starting at any position).

**Note:** The SUBSTR() and [MID()](https://www.w3schools.com/sql/func_mysql_mid.asp) functions equals to the [SUBSTRING()](https://www.w3schools.com/sql/func_mysql_substring.asp) function.

SUBSTR(string, start, length)

**OR:**

SUBSTR(string FROM start FOR length)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to extract from |
| start | Required. The start position. Can be both a positive or negative number. If it is a positive number, this function extracts from the beginning of the string. If it is a negative number, this function extracts from the end of the string |
| length | Optional. The number of characters to extract. If omitted, the whole string will be returned (from the start position) |

Extract a substring from the text in a column (start at position 2, extract 5 characters):

SELECT SUBSTR(CustomerName, 2, 5) AS ExtractString
FROM Customers;

Extract a substring from a string (start from the end, at position -5, extract 5 characters):

SELECT SUBSTR("SQL Tutorial", -5, 5) AS ExtractString;

# SUBSTRING()

Extract a substring from a string (start at position 5, extract 3 characters):

SELECT SUBSTRING("SQL Tutorial", 5, 3) AS ExtractString;

The SUBSTRING() function extracts a substring from a string (starting at any position).

**Note:** The [SUBSTR()](https://www.w3schools.com/sql/func_mysql_substr.asp) and [MID()](https://www.w3schools.com/sql/func_mysql_mid.asp) functions equals to the SUBSTRING() function.

SUBSTRING(string, start, length)

**OR:**

SUBSTRING(string FROM start FOR length)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to extract from |
| start | Required. The start position. Can be both a positive or negative number. If it is a positive number, this function extracts from the beginning of the string. If it is a negative number, this function extracts from the end of the string |
| length | Optional. The number of characters to extract. If omitted, the whole string will be returned (from the start position) |

Extract a substring from the text in a column (start at position 2, extract 5 characters):

SELECT SUBSTRING(CustomerName, 2, 5) AS ExtractString
FROM Customers;

Extract a substring from a string (start from the end, at position -5, extract 5 characters):

SELECT SUBSTRING("SQL Tutorial", -5, 5) AS ExtractString;

# SUBSTRING\_INDEX()

Return a substring of a string before a specified number of delimiter occurs:

SELECT SUBSTRING\_INDEX("www.w3schools.com", ".", 1);

The SUBSTRING\_INDEX() function returns a substring of a string before a specified number of delimiter occurs.

SUBSTRING\_INDEX(string, delimiter, number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The original string |
| delimiter | Required. The delimiter to search for |
| number | Required. The number of times to search for the delimiter. Can be both a positive or negative number. If it is a positive number, this function returns all to the left of the delimiter. If it is a negative number, this function returns all to the right of the delimiter. |

Return a substring of a string before a specified number of delimiter occurs:

SELECT SUBSTRING\_INDEX("www.w3schools.com", ".", 2);

# TRIM()

Remove leading and trailing spaces from a string:

SELECT TRIM('    SQL Tutorial    ') AS TrimmedString;

The TRIM() function removes leading and trailing spaces from a string.

TRIM(string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to remove leading and trailing spaces from |

# UCASE()

Convert the text to upper-case:

SELECT UCASE("SQL Tutorial is FUN!");

The UCASE() function converts a string to upper-case.

**Note:** This function is equal to the [UPPER()](https://www.w3schools.com/sql/func_mysql_upper.asp) function.

UCASE(text)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| text | Required. The string to convert |

Convert the text in "CustomerName" to upper-case:

SELECT UCASE(CustomerName) AS UppercaseCustomerName
FROM Customers;

# UPPER()

Convert the text to upper-case:

SELECT UPPER("SQL Tutorial is FUN!");

The UPPER() function converts a string to upper-case.

**Note:** This function is equal to the [UCASE()](https://www.w3schools.com/sql/func_mysql_ucase.asp) function.

UPPER(text)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| text | Required. The string to convert |

Convert the text in "CustomerName" to upper-case:

SELECT UPPER(CustomerName) AS UppercaseCustomerName
FROM Customers;

NUMERIC FUNCTION

# ABS()

Return the absolute value of a number:

SELECT ABS(-243.5);

The ABS() function returns the absolute (positive) value of a number.

ABS(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A numeric value |

# ACOS()ASIN()ATAN()COS()

Return the arc cosine of a number:

SELECT ACOS(0.25);

[Try it Yourself »](https://www.w3schools.com/sql/trymysql.asp?filename=trysql_func_mysql_acos)

The ACOS() function returns the arc cosine of a number.

The specified number must be between -1 to 1, otherwise this function returns NULL.

ACOS(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A numeric value |

Return the arc cosine of a number:

SELECT ACOS(-0.8);

# ATAN2()

Return the arc tangent of two values:

SELECT ATAN2(0.50, 1);

# COT()

Return the cotangent of a number:

SELECT COT(6);

The COT() function returns the cotangent of a number.

COT(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A numeric value. If number is 0, an error or NULL is returned |

# AVG()

Return the average value for the "Price" column in the "Products" table:

SELECT AVG(Price) AS AveragePrice FROM Products;

The AVG() function returns the average value of an expression.

**Note:** NULL values are ignored.

AVG(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Required. A numeric value (can be a field or a formula) |

# COUNT()

Return the number of products in the "Products" table:

SELECT COUNT(ProductID) AS NumberOfProducts FROM Products;

The COUNT() function returns the number of records returned by a select query.

**Note:** NULL values are not counted.

COUNT(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Required. A field or a string value |

# CEIL()

Return the smallest integer value that is greater than or equal to 25.75:

SELECT CEIL(25.75);

The CEIL() function returns the smallest integer value that is bigger than or equal to a number.

**Note:** This function is equal to the [CEILING()](https://www.w3schools.com/sql/func_mysql_ceiling.asp) function.

CEIL(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A numeric value |

Return the smallest integer value that is greater than or equal to 25:

SELECT CEIL(25);

# CEILING()

Return the smallest integer value that is greater than or equal to 25.75:

SELECT CEILING(25.75);

The CEILING() function returns the smallest integer value that is bigger than or equal to a number.

**Note:** This function is equal to the [CEIL()](https://www.w3schools.com/sql/func_mysql_ceil.asp) function.

CEILING(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A numeric value |

Return the smallest integer value that is greater than or equal to 25:

SELECT CEILING(25);

# DEGREES()

Convert the radian value into degrees:

SELECT DEGREES(1.5);

The DEGREES() function converts a value in radians to degrees.

**Note:** See also the [RADIANS()](https://www.w3schools.com/sql/func_mysql_radians.asp) and [PI()](https://www.w3schools.com/sql/func_mysql_pi.asp) functions.

DEGREES(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A numeric value |

# DIV

Integer division (10/5):

SELECT 10 DIV 5;

The DIV function is used for integer division (x is divided by y). An integer value is returned.

x DIV y

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| x | Required. A value that will be divided by y |
| y | Required. The divisor |

Integer division (8/3):

SELECT 8 DIV 3;

# EXP()

Return e raised to the power of 1:

SELECT EXP(1);

The EXP() function returns e raised to the power of the specified number.

The constant e (2.718281...), is the base of natural logarithms.

**Tip:** Also look at the [LOG()](https://www.w3schools.com/sql/func_mysql_log.asp) and [LN()](https://www.w3schools.com/sql/func_mysql_ln.asp) functions.

EXP(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. The power number |

# FLOOR()

Return the largest integer value that is less than or equal to 25.75:

SELECT FLOOR(25.75);

The FLOOR() function returns the largest integer value that is smaller than or equal to a number.

**Note:** Also look at the [ROUND()](https://www.w3schools.com/sql/func_mysql_round.asp), [CEIL()](https://www.w3schools.com/sql/func_mysql_ceil.asp), [CEILING()](https://www.w3schools.com/sql/func_mysql_ceiling.asp), [TRUNCATE()](https://www.w3schools.com/sql/func_mysql_truncate.asp), and [DIV](https://www.w3schools.com/sql/func_mysql_div.asp) functions.

FLOOR(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A numeric value |

Return the largest integer value that is greater than or equal to 25:

SELECT FLOOR(25);

# GREATEST()

[❮ MySQL Functions](https://www.w3schools.com/sql/sql_ref_mysql.asp)

### **Example**

Return the greatest value of the list of arguments:

SELECT GREATEST(3, 12, 34, 8, 25);

The GREATEST() function returns the greatest value of the list of arguments.

**Note:** See also the [LEAST()](https://www.w3schools.com/sql/func_mysql_least.asp) function.

GREATEST(arg1, arg2, arg3, ...)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| arg1, arg2, arg3, ... | Required. The list of arguments to be evaluated |

Return the greatest value of the list of arguments:

SELECT GREATEST("w3Schools.com", "microsoft.com", "apple.com");

# LEAST()

Return the smallest value of the list of arguments:

SELECT LEAST(3, 12, 34, 8, 25);

The LEAST() function returns the smallest value of the list of arguments.

**Note:** See also the [GREATEST()](https://www.w3schools.com/sql/func_mysql_greatest.asp) function.

LEAST(arg1, arg2, arg3, ...)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| arg1, arg2, arg3, ... | Required. The list of arguments to be evaluated |

Return the smallest value of the list of arguments:

SELECT LEAST("w3Schools.com", "microsoft.com", "apple.com");

# LN()

Return the natural logarithm of 2:

SELECT LN(2);

The LN() function returns the natural logarithm of a number.

**Note:** See also the [LOG()](https://www.w3schools.com/sql/func_mysql_log.asp) and [EXP()](https://www.w3schools.com/sql/func_mysql_exp.asp) functions.

LN(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A number. Must be greater than 0 |

# LOG()

Return the natural logarithm of 2:

SELECT LOG(2);

The LOG() function returns the natural logarithm of a specified number, or the logarithm of the number to the specified base.

**Note:** See also the [LN()](https://www.w3schools.com/sql/func_mysql_ln.asp) and [EXP()](https://www.w3schools.com/sql/func_mysql_exp.asp) functions.

LOG(number)

OR:

LOG(base, number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A number. Must be greater than 0 |
| base | The base of number. Must be greater than 1 |

Return the natural logarithm of 4 to a specified base (2):

SELECT LOG(2, 4);

# LOG10()

Return the base-10 logarithm of 2:

SELECT LOG10(2);

The LOG10() function returns the natural logarithm of a number to base-10.

**Note:** See also the [LOG()](https://www.w3schools.com/sql/func_mysql_log.asp) function.

LOG10(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | A number greater than 0 |

Return the base-10 logarithm of 4.5:

SELECT LOG10(4.5);

# LOG2()

Return the base-2 logarithm of 6:

SELECT LOG2(6);

The LOG2() function returns the natural logarithm of a number to base-2.

**Note:** See also the [LOG()](https://www.w3schools.com/sql/func_mysql_log.asp) function.

LOG2(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | A number. Must be greater than 0 |

Return the base-2 logarithm of 64:

SELECT LOG2(64);

# MAX()

Find the price of the most expensive product in the "Products" table:

SELECT MAX(Price) AS LargestPrice FROM Products;

The MAX() function returns the maximum value in a set of values.

**Note:** See also the [MIN()](https://www.w3schools.com/sql/func_mysql_min.asp) function.

MAX(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Required. A numeric value (can be a field or a formula) |

# MIN()

Find the price of the cheapest product in the "Products" table:

SELECT MIN(Price) AS SmallestPrice FROM Products;

The MIN() function returns the minimum value in a set of values.

**Note:** See also the [MAX()](https://www.w3schools.com/sql/func_mysql_max.asp) function.

## Syntax

MIN(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Required. A numeric value (can be a field or a formula) |

# MOD()

Return the remainder of 18/4:

SELECT MOD(18, 4);

The MOD() function returns the remainder of a number divided by another number.

MOD(x, y)

**OR:**

x MOD y

**OR:**

x % y

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| x | Required. A value that will be divided by y |
| y | Required. The divisor |

# PI()

Return the value of PI:

SELECT PI();

# POW()

Return 4 raised to the second power:

SELECT POW(4, 2);

The POW() function returns the value of a number raised to the power of another number.

# **Note:** This function is equal to the [POWER()](https://www.w3schools.com/sql/func_mysql_power.asp) function.

# POWER()

Return 4 raised to the second power:

SELECT POWER(4, 2);

POW(x, y)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| x | Required. A number (the base) |
| y | Required. A number (the exponent) |

# RADIANS()

Convert a degree value into radians:

SELECT RADIANS(180);

The RADIANS() function converts a degree value into radians.

**Note:** See also the [DEGREES()](https://www.w3schools.com/sql/func_mysql_degrees.asp) and [PI()](https://www.w3schools.com/sql/func_mysql_pi.asp) functions.

RADIANS(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A number in degrees |

Convert a degree value into radians:

SELECT RADIANS(-45);

# RAND()

Return a random decimal number (no seed value - so it returns a completely random number >= 0 and <1):

SELECT RAND();

The RAND() function returns a random number between 0 (inclusive) and 1 (exclusive).

RAND(seed)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| seed | Optional. If seed is specified, it returns a repeatable sequence of random numbers. If no seed is specified, it returns a completely random number |

Return a random decimal number (with seed value of 6):

SELECT RAND(6);

[Try it Yourself »](https://www.w3schools.com/sql/trymysql.asp?filename=trysql_func_mysql_rand2)

Return a random decimal number >= 5 and <10:

SELECT RAND()\*(10-5)+5;

[Try it Yourself »](https://www.w3schools.com/sql/trymysql.asp?filename=trysql_func_mysql_rand3)

Return a random number >= 5 and <=10:

SELECT FLOOR(RAND()\*(10-5+1)+5);

# ROUND()

Round the number to 2 decimal places:

SELECT ROUND(135.375, 2);

The ROUND() function rounds a number to a specified number of decimal places.

**Note:** See also the [FLOOR()](https://www.w3schools.com/sql/func_mysql_floor.asp), [CEIL()](https://www.w3schools.com/sql/func_mysql_ceil.asp), [CEILING()](https://www.w3schools.com/sql/func_mysql_ceiling.asp), and [TRUNCATE()](https://www.w3schools.com/sql/func_mysql_truncate.asp) functions.

ROUND(number, decimals)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. The number to be rounded |
| decimals | Optional. The number of decimal places to round number to. If omitted, it returns the integer (no decimals) |

Round the number to 0 decimal places:

SELECT ROUND(345.156, 0);

Round the Price column (to 1 decimal) in the "Products" table:

SELECT ProductName, Price, ROUND(Price, 1) AS RoundedPrice
FROM Products;

# SIGN()

Return the sign of a number:

SELECT SIGN(255.5);

The SIGN() function returns the sign of a number.

This function will return one of the following:

* If number > 0, it returns 1
* If number = 0, it returns 0
* If number < 0, it returns -1

SIGN(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. The number to return the sign for |

Return the sign of a number:

SELECT SIGN(-12);

# SIN()

Return the sine of a number:

SELECT SIN(2);

The SIN() function returns the sine of a number.

SIN(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A numeric value |

Return the sine of a number:

SELECT SIN(-1);

# SQRT()

### **Example**

Return the square root of a number:

SELECT SQRT(64);

The SQRT() function returns the square root of a number.

SQRT(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A number to calculate the square root of. Must be greater than 0 |

Return the square root of a number:

SELECT SQRT(13);

# SUM()

Return the sum of the "Quantity" field in the "OrderDetails" table:

SELECT SUM(Quantity) AS TotalItemsOrdered FROM OrderDetails;

The SUM() function calculates the sum of a set of values.

**Note:** NULL values are ignored.

SUM(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Required. A field or a formula |

# TAN()

Return the tangent of a number:

SELECT TAN(1.75);

The TAN() function returns the tangent of a number.

TAN(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A numeric value |

Return the tangent of a number:

SELECT TAN(-3);

# TRUNCATE()

Return a number truncated to 2 decimal places:

SELECT TRUNCATE(135.375, 2);

he TRUNCATE() function truncates a number to the specified number of decimal places.

**Note:** See also the [FLOOR()](https://www.w3schools.com/sql/func_mysql_floor.asp), [CEIL()](https://www.w3schools.com/sql/func_mysql_ceil.asp), [CEILING()](https://www.w3schools.com/sql/func_mysql_ceiling.asp), and [ROUND()](https://www.w3schools.com/sql/func_mysql_round.asp) functions.

TRUNCATE(number, decimals)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. The number to be truncated |
| decimals | Required. The number of decimal places to truncate to |

Return a number truncated to 0 decimal places:

SELECT TRUNCATE(345.156, 0);

|  |
| --- |
| **TRUNCATE(345.156, 0)** |
| 345 |

DATE FUNCTION

#   ADDDATE()

Add 10 days to a date and return the date:

SELECT ADDDATE("2017-06-15", INTERVAL 10 DAY);

The ADDDATE() function adds a time/date interval to a date and then returns the date.

ADDDATE(date, INTERVAL value addunit)

**OR:**

ADDDATE(date, days)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required.  The date to be modified |
| days | Required. The number of days to add to date |
| value | Required. The value of the time/date interval to add. Both positive and negative values are allowed |
| addunit | Required. The type of interval to add. Can be one of the following values:* MICROSECOND SECOND MINUTE HOUR
* DAY WEEK MONTH QUARTER YEAR
* SECOND\_MICROSECOND MINUTE\_MICROSECOND
* MINUTE\_SECOND HOUR\_MICROSECOND
* HOUR\_SECOND HOUR\_MINUTE
* DAY\_MICROSECOND DAY\_SECOND
* DAY\_MINUTE DAY\_HOUR YEAR\_MONTH
 |

Add 15 minutes to a date and return the date:

SELECT ADDDATE("2017-06-15 09:34:21", INTERVAL 15 MINUTE);

#   ADDTIME()

Add 2 seconds to a time and return the datetime:

SELECT ADDTIME("2017-06-15 09:34:21", "2");

The ADDTIME() function adds a time interval to a time/datetime and then returns the time/datetime.

ADDTIME(datetime, addtime)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| datetime | Required.  The time/datetime to be modified |
| addtime | Required. The time interval to add to datetime. Both positive and negative values are allowed |

Add 5 seconds and 3 microseconds to a time and return the datetime:

SELECT ADDTIME("2017-06-15 09:34:21.000001", "5.000003");

Add 2 hours, 10 minutes, 5 seconds, and 3 microseconds to a time and return the datetime:

SELECT ADDTIME("2017-06-15 09:34:21.000001", "2:10:5.000003");

Add 5 days, 2 hours, 10 minutes, 5 seconds, and 3 microseconds to a time and return the datetime:

SELECT ADDTIME("2017-06-15 09:34:21.000001", "5 2:10:5.000003");

Add 2 hours, 10 minutes, 5 seconds, and 3 microseconds to a time and return the time:

SELECT ADDTIME("09:34:21.000001", "2:10:5.000003");

#  CURDATE()

Return the current date:

SELECT CURDATE();

The CURDATE() function returns the current date.

**Note:** The date is returned as "YYYY-MM-DD" (string) or as YYYYMMDD (numeric).

**Note:** This function equals the [CURRENT\_DATE()](https://www.w3schools.com/sql/func_mysql_current_date.asp) function.

CURDATE()

Return the current date + 1:

SELECT CURDATE() + 1;

#   CURRENT\_DATE()

Return the current date:

SELECT CURRENT\_DATE();

The CURRENT\_DATE() function returns the current date.

**Note:** The date is returned as "YYYY-MM-DD" (string) or as YYYYMMDD (numeric).

**Note:**This function equals the [CURDATE()](https://www.w3schools.com/sql/func_mysql_curdate.asp) function.

CURRENT\_DATE()

Return the current date + 1:

SELECT CURRENT\_DATE() + 1;

#  CURRENT\_TIME()

Return current time:

SELECT CURRENT\_TIME();

The CURRENT\_TIME() function returns the current time.

**Note:** The time is returned as "HH-MM-SS" (string) or as HHMMSS.uuuuuu (numeric).

**Note:**This function equals the [CURTIME()](https://www.w3schools.com/sql/func_mysql_curtime.asp) function.

CURRENT\_TIME()

Return current time + 1:

SELECT CURRENT\_TIME() + 1;

#   CURRENT\_TIMESTAMP()

Return the current date and time:

SELECT CURRENT\_TIMESTAMP();

The CURRENT\_TIMESTAMP() function returns the current date and time.

**Note:** The date and time is returned as "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS.uuuuuu (numeric).

CURRENT\_TIMESTAMP()

Return the current date and time + 1:

SELECT CURRENT\_TIMESTAMP() + 1;

Number of Records: 1

|  |
| --- |
| **CURRENT\_TIMESTAMP() + 1** |
| 20201118160136 |

#   CURTIME()

Return current time:

SELECT CURTIME();

The CURTIME() function returns the current time.

**Note:** The time is returned as "HH-MM-SS" (string) or as HHMMSS.uuuuuu (numeric).

**Note:** This function equals the [CURRENT\_TIME()](https://www.w3schools.com/sql/func_mysql_current_time.asp) function.

CURTIME()

Return current time + 1:

SELECT CURTIME() + 1;

|  |
| --- |
| **CURTIME() + 1** |
| 160214 |

#   DATE()

Extract the date part:

SELECT DATE("2017-06-15");

The DATE() function extracts the date part from a datetime expression.

DATE(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Required.  A valid date/datetime value. Returns NULL if expression is not a date or a datetime |

Extract the date part:

SELECT DATE("2017-06-15 09:34:21");

Extract the date part (will return NULL):

SELECT DATE("The date is 2017-06-15");

Extract the date part:

SELECT DATE(OrderDate) FROM Orders;

#  DATEDIFF()

Return the number of days between two date values:

SELECT DATEDIFF("2017-06-25", "2017-06-15");

The DATEDIFF() function returns the number of days between two date values.

DATEDIFF(date1, date2)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date1, date2 | Required. Two dates to calculate the number of days between. (date1 - date2) |

Return the number of days between two date values:

SELECT DATEDIFF("2017-06-25 09:34:21", "2017-06-15 15:25:35");

Return the number of days between two date values:

SELECT DATEDIFF("2017-01-01", "2016-12-24");

#   DATE\_ADD()

Add 10 days to a date and return the date:

SELECT DATE\_ADD("2017-06-15", INTERVAL 10 DAY);

The DATE\_ADD() function adds a time/date interval to a date and then returns the date.

DATE\_ADD(date, INTERVAL value addunit)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The date to be modified |
| value | Required. The value of the time/date interval to add. Both positive and negative values are allowed |
| addunit | Required. The type of interval to add. Can be one of the following values:* MICROSECOND SECOND MINUTE HOUR
* DAY WEEK MONTH QUARTER YEAR
* SECOND\_MICROSECOND MINUTE\_MICROSECOND
* MINUTE\_SECOND HOUR\_MICROSECOND
* HOUR\_SECOND HOUR\_MINUTE
* DAY\_MICROSECOND DAY\_SECOND
* DAY\_MINUTE DAY\_HOUR YEAR\_MONTH
 |

Add 15 minutes to a date and return the date:

SELECT DATE\_ADD("2017-06-15 09:34:21", INTERVAL 15 MINUTE);

Subtract 3 hours to a date and return the date:

SELECT DATE\_ADD("2017-06-15 09:34:21", INTERVAL -3 HOUR);

Subtract 2 months to a date and return the date:

SELECT DATE\_ADD("2017-06-15", INTERVAL -2 MONTH);

Number of Records: 1

|  |
| --- |
| **DATE\_ADD("2017-06-15", INTERVAL -2 MONTH)** |
| 2017-04-15 |

#   DATE\_FORMAT()

Format a date:

SELECT DATE\_FORMAT("2017-06-15", "%Y");

The DATE\_FORMAT() function formats a date as specified.

DATE\_FORMAT(date, format)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The date to be formatted |
| format | Required. The format to use. Can be one or a combination of the following values:

|  |  |
| --- | --- |
| **Format** | **Description** |
| %a | Abbreviated weekday name (Sun to Sat) |
| %b | Abbreviated month name (Jan to Dec) |
| %c | Numeric month name (0 to 12) |
| %D | Day of the month as a numeric value, followed by suffix (1st, 2nd, 3rd, ...) |
| %d | Day of the month as a numeric value (01 to 31) |
| %e | Day of the month as a numeric value (0 to 31) |
| %f | Microseconds (000000 to 999999) |
| %H | Hour (00 to 23) |
| %h | Hour (00 to 12) |
| %I | Hour (00 to 12) |
| %i | Minutes (00 to 59) |
| %j | Day of the year (001 to 366) |
| %k | Hour (0 to 23) |
| %l | Hour (1 to 12) |
| %M | Month name in full (January to December) |
| %m | Month name as a numeric value (00 to 12) |
| %p | AM or PM |
| %r | Time in 12 hour AM or PM format (hh:mm:ss AM/PM) |
| %S | Seconds (00 to 59) |
| %s | Seconds (00 to 59) |
| %T | Time in 24 hour format (hh:mm:ss) |
| %U | Week where Sunday is the first day of the week (00 to 53) |
| %u | Week where Monday is the first day of the week (00 to 53) |
| %V | Week where Sunday is the first day of the week (01 to 53). Used with %X |
| %v | Week where Monday is the first day of the week (01 to 53). Used with %x |
| %W | Weekday name in full (Sunday to Saturday) |
| %w | Day of the week where Sunday=0 and Saturday=6 |
| %X | Year for the week where Sunday is the first day of the week. Used with %V |
| %x | Year for the week where Monday is the first day of the week. Used with %v |
| %Y | Year as a numeric, 4-digit value |
| %y | Year as a numeric, 2-digit value |

 |

Format a date:

SELECT DATE\_FORMAT("2017-06-15", "%M %d %Y");

Format a date:

SELECT DATE\_FORMAT("2017-06-15", "%W %M %e %Y");

Format a date:

SELECT DATE\_FORMAT(BirthDate, "%W %M %e %Y") FROM Employees;

Number of Records: 9

|  |
| --- |
| **DATE\_FORMAT(BirthDate, "%W %M %e %Y")** |
| Sunday December 8 1968 |
| Tuesday February 19 1952 |
| Friday August 30 1963 |
| Friday September 19 1958 |
| Friday March 4 1955 |
| Tuesday July 2 1963 |
| Sunday May 29 1960 |
| Thursday January 9 1958 |

#   DATE\_SUB()

Subtract 10 days from a date and return the date:

SELECT DATE\_SUB("2017-06-15", INTERVAL 10 DAY);

The DATE\_SUB() function subtracts a time/date interval from a date and then returns the date.

DATE\_SUB(date, INTERVAL value interval)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The date to be modified |
| value | Required. The value of the time/date interval to subtract. Both positive and negative values are allowed |
| interval | Required. The type of interval to subtract. Can be one of the following values:* MICROSECOND SECOND MINUTE HOUR
* DAY WEEK MONTH QUARTER YEAR
* SECOND\_MICROSECOND MINUTE\_MICROSECOND
* MINUTE\_SECOND HOUR\_MICROSECOND
* HOUR\_SECOND HOUR\_MINUTE DAY\_MICROSECOND
* DAY\_SECOND DAY\_MINUTE DAY\_HOUR YEAR\_MONTH
 |

Subtract 15 minutes from a date and return the date:

SELECT DATE\_SUB("2017-06-15 09:34:21", INTERVAL 15 MINUTE);

Subtract 3 hours from a date and return the date:

SELECT DATE\_SUB("2017-06-15 09:34:21", INTERVAL 3 HOUR);

Add 2 months to a date and return the date:

SELECT DATE\_SUB("2017-06-15", INTERVAL -2 MONTH);

#  DAY()

Return the day of the month for a date:

SELECT DAY("2017-06-15");

The DAY() function returns the day of the month for a given date (a number from 1 to 31).

**Note:** This function equals the [DAYOFMONTH()](https://www.w3schools.com/sql/func_mysql_dayofmonth.asp) function.

DAY(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The date to extract the day from |

Return the day of the month for a date:

SELECT DAY("2017-06-15 09:34:21");

Return the day of the month for the current system date:

SELECT DAY(CURDATE());

#  DAYNAME()

Return the weekday name for a date:

SELECT DAYNAME("2017-06-15");

The DAYNAME() function returns the weekday name for a given date.

DAYNAME(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required.  The date to extract the weekday name from |

Return the weekday name for a date:

SELECT DAYNAME("2017-06-15 09:34:21");

Return the weekday name for the current system date:

SELECT DAYNAME(CURDATE());

Number of Records: 1

|  |
| --- |
| **DAYNAME(CURDATE())** |
| Wednesday |

#   DAYOFMONTH()

Return the day of the month for a date:

SELECT DAYOFMONTH("2017-06-15");

The DAYOFMONTH() function returns the day of the month for a given date (a number from 1 to 31).

**Note:** This function equals the [DAY()](https://www.w3schools.com/sql/func_mysql_day.asp) function.

DAYOFMONTH(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The date to extract the day from |

Return the day of the month for a date:

SELECT DAYOFMONTH("2017-06-15 09:34:21");

Return the day of the month for the current system date:

SELECT DAYOFMONTH(CURDATE());

#  DAYOFWEEK()

Return the weekday index for a date:

SELECT DAYOFWEEK("2017-06-15");

The DAYOFWEEK() function returns the weekday index for a given date (a number from 1 to 7).

**Note:** 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, 7=Saturday.

DAYOFWEEK(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The date to return the weekday index from |

Return the weekday index for a date:

SELECT DAYOFWEEK("2017-06-15 09:34:21");

[Try it Yourself »](https://www.w3schools.com/sql/trymysql.asp?filename=trysql_func_mysql_dayofweek2)

Return the weekday index for the current system date:

SELECT DAYOFWEEK(CURDATE());

Number of Records: 1

|  |
| --- |
| **DAYOFWEEK(CURDATE())** |
| 4 |

#   DAYOFYEAR()

Return the day of the year for a date:

SELECT DAYOFYEAR("2017-06-15");

The DAYOFYEAR() function returns the day of the year for a given date (a number from 1 to 366).

DAYOFYEAR(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The date to return the day of the year from |

Return the day of the year for a date:

SELECT DAYOFYEAR("2017-01-01");

[Try it Yourself »](https://www.w3schools.com/sql/trymysql.asp?filename=trysql_func_mysql_dayofyear2)

Return the day of the year for the current system date:

SELECT DAYOFYEAR(CURDATE());

#   EXTRACT()

Extract the month from a date:

SELECT EXTRACT(MONTH FROM "2017-06-15");

The EXTRACT() function extracts a part from a given date.

EXTRACT(part FROM date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| part | Required. The part to extract. Can be one of the following:* MICROSECOND SECOND MINUTE HOUR DAY WEEK
* MONTH QUARTER YEAR
* SECOND\_MICROSECOND MINUTE\_MICROSECOND
* MINUTE\_SECOND HOUR\_MICROSECOND
* HOUR\_SECOND HOUR\_MINUTE
* DAY\_MICROSECOND DAY\_SECOND
* DAY\_MINUTE DAY\_HOUR YEAR\_MONTH
 |
| date | Required. The date to extract a part from |

Extract the week from a date:

SELECT EXTRACT(WEEK FROM "2017-06-15");

Extract the minute from a datetime:

SELECT EXTRACT(MINUTE FROM "2017-06-15 09:34:21");

Extract the year and month from a datetime:

SELECT EXTRACT(YEAR\_MONTH FROM "2017-06-15 09:34:21");

Number of Records: 1

|  |
| --- |
| **EXTRACT(YEAR\_MONTH FROM "2017-06-15 09:34:21")** |
| 201706 |

#   FROM\_DAYS()

Return a date from a numeric representation of the day:

SELECT FROM\_DAYS(685467);

The FROM\_DAYS() function returns a date from a numeric datevalue.

The FROM\_DAYS() function is to be used only with dates within the Gregorian calendar.

**Note:** This function is the opposite of the [TO\_DAYS()](https://www.w3schools.com/sql/func_mysql_to_days.asp) function.

FROM\_DAYS(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required.  The numeric day to convert to a date |

Return a date from a numeric representation of the day:

SELECT FROM\_DAYS(780500);

Number of Records: 1

|  |
| --- |
| **FROM\_DAYS(780500)** |
| 2136-12-08 |

#   HOUR()

Return the hour part of a datetime:

SELECT HOUR("2017-06-20 09:34:00");

The HOUR() function returns the hour part for a given date (from 0 to 838).

HOUR(datetime)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| datetime | Required. The datetime value to extract the hour from |

Return the hour part of a datetime:

SELECT HOUR("838:59:59");

Number of Records: 1

|  |
| --- |
| **HOUR("838:59:59")** |
| 838 |

#   LAST\_DAY()

Extract the last day of the month for the given date:

SELECT LAST\_DAY("2017-06-20");

The LAST\_DAY() function extracts the last day of the month for a given date.

LAST\_DAY(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The date to extract the last day of the month from |

Extract the last day of the month for the given date:

SELECT LAST\_DAY("2017-02-10 09:34:00");

Number of Records: 1

|  |
| --- |
| **LAST\_DAY("2017-02-10 09:34:00")** |
| 2017-02-28 |

#   LOCALTIME()

Return current date and time:

SELECT LOCALTIME();

The LOCALTIME() function returns the current date and time.

**Note:** The date and time is returned as "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS.uuuuuu (numeric).

LOCALTIME()

Return current date and time

SELECT LOCALTIME(), LOCALTIME()+10, LOCALTIME()+100 ;

Number of Records: 1

|  |  |  |
| --- | --- | --- |
| **LOCALTIME()** | **LOCALTIME()+10** | **LOCALTIME()+100** |
| 2020-11-18 16:15:35 | 20201118161545 | 20201118161635 |

#   LOCALTIMESTAMP()

Return current date and time:

SELECT LOCALTIMESTAMP();

The LOCALTIMESTAMP() function returns the current date and time.

**Note:** The date and time is returned as "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS.uuuuuu (numeric).

LOCALTIMESTAMP()

Return current date and time +0, 10,100:

SELECT LOCALTIMESTAMP(), LOCALTIMESTAMP() + 10, LOCALTIMESTAMP() + 100;

Number of Records: 1

|  |  |  |
| --- | --- | --- |
| LOCALTIMESTAMP() | LOCALTIMESTAMP() + 10 | LOCALTIMESTAMP() + 100 |
| 2020-11-18 16:20:29 | 20201118162039 | 20201118162129 |

#   MAKEDATE()

Create and return a date based on  a year and a number of days value:

SELECT MAKEDATE(2017, 3);

The MAKEDATE() function creates and returns a date based on a year and a number of days value.

MAKEDATE(year, day)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| year | Required. A year (4-digits) |
| day | Required. A number that indicates the day of the year |

Create and return a date based on  a year and a number of days value:

SELECT MAKEDATE(2017, 175);

Create and return a date based on  a year and a number of days value:

SELECT MAKEDATE(2017, 366);

Number of Records: 1

|  |
| --- |
| **MAKEDATE(2017, 366)** |
| 2018-01-01 |

#   MAKETIME()

Create and return a time value based on an hour, minute, and second value:

SELECT MAKETIME(11, 35, 4);

The MAKETIME() function creates and returns a time based on an hour, minute, and second value.

MAKETIME(hour, minute, second)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| hour | Required. The hour value |
| minute | Required. The minute value |
| second | Required. The seconds value |

Create and return a time value based on an hour, minute, and second value:

SELECT MAKETIME(16, 1, 0);

[Try it Yourself »](https://www.w3schools.com/sql/trymysql.asp?filename=trysql_func_mysql_maketime2)

Create and return a time value based on an hour, minute, and second value:

SELECT MAKETIME(21, 59, 59);

Number of Records: 1

|  |
| --- |
| **MAKETIME(21, 59, 59)** |
| 21:59:59 |

#   MICROSECOND()

Return the microsecond part of a datetime:

SELECT MICROSECOND("2017-06-20 09:34:00.000023");

The MICROSECOND() function returns the microsecond part of a time/datetime (from 0 to 999999).

MICROSECOND(datetime)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| datetime | Required. The time or datetime to extract the microsecond from |

Return the microsecond part of a time value:

SELECT MICROSECOND("23:59:59.000045");

Number of Records: 1

|  |
| --- |
| **MICROSECOND("23:59:59.000045")** |
| 45 |

#   MINUTE()

Return the minute part of a datetime value:

SELECT MINUTE("2017-06-20 09:34:00");

The MINUTE() function returns the minute part of a time/datetime (from 0 to 59).

MINUTE(datetime)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| datetime | Required. The time or datetime extract the minute from |

Return the minute part of a time value:

SELECT MINUTE("23:59:59");

Number of Records: 1

|  |
| --- |
| **MINUTE("23:59:59")** |
| 59 |

#   MONTH()

Return the month part of a date:

SELECT MONTH("2017-06-15");

The MONTH() function returns the month part for a given date (a number from 1 to 12).

MONTH(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The date or datetime to extract the month from |

Return the month part of a date:

SELECT MONTH("2017-06-15 09:34:21");

Return the month part of the current system date:

SELECT MONTH(CURDATE());

Number of Records: 1

|  |
| --- |
| **MONTH(CURDATE())** |
| 11 |

#   MONTHNAME()

Return the name of the month for a date:

SELECT MONTHNAME("2017-06-15");

The MONTHNAME() function returns the name of the month for a given date.

MONTHNAME(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The date or datetime value to extract the month name from |

Return the name of the month for a date:

SELECT MONTHNAME("2017-06-15 09:34:21");

Return the name of the month for the current system date:

SELECT MONTHNAME(CURDATE());

Number of Records: 1

|  |
| --- |
| **MONTHNAME(CURDATE())** |
| November |

#   NOW()

Return current date and time:

SELECT NOW();

The NOW() function returns the current date and time.

**Note:** The date and time is returned as "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS.uuuuuu (numeric).

NOW()

Return current date and time + 0,10,100:

SELECT NOW(), NOW() + 10, NOW() + 100;

Number of Records: 1

|  |  |  |
| --- | --- | --- |
| **NOW()** | **NOW() + 10** | **NOW() + 100** |
| 2020-11-18 17:45:40 | 20201118174550 | 20201118174640 |

#   PERIOD\_ADD()

Add a specified number of months to a period:

SELECT PERIOD\_ADD(201703, 5);

The PERIOD\_ADD() function adds a specified number of months to a period.

The PERIOD\_ADD() function will return the result formatted as YYYYMM.

PERIOD\_ADD(period, number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| period | Required. A period. Format: YYMM or YYYYMM |
| number | Required. The number of months to add to period. Both positive and negative values are allowed |

Add a specified number of months to a period:

SELECT PERIOD\_ADD(201703, 15);

Add a specified number of months to a period:

SELECT PERIOD\_ADD(201703, -2);

Number of Records: 1

|  |
| --- |
| **PERIOD\_ADD(201703, -2)** |
| 201701 |

#   PERIOD\_DIFF()

Return the difference between two periods:

SELECT PERIOD\_DIFF(201710, 201703);

The PERIOD\_DIFF() function returns the difference between two periods. The result will be in months.

**Note:** period1 and period2 should be in the same format.

PERIOD\_DIFF(period1, period2)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| period1 | Required. A period. Format: YYMM or YYYYMM |
| period2 | Required. Another period. Format: YYMM or YYYYMM |

Return the difference between two periods:

SELECT PERIOD\_DIFF(201703, 201803);

Return the difference between two periods:

SELECT PERIOD\_DIFF(1703, 1612);

Number of Records: 1

|  |
| --- |
| **PERIOD\_DIFF(1703, 1612)** |
| 3 |

#   QUARTER()

Return the quarter of the year for the date:

SELECT QUARTER("2017-06-15");

The QUARTER() function returns the quarter of the year for a given date value (a number from 1 to 4).

* January-March returns 1
* April-June returns 2
* July-Sep returns 3
* Oct-Dec returns 4

QUARTER(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The date or datetime to extract the quarter from |

Return the quarter of the year for the date:

SELECT QUARTER("2017-01-01 09:34:21");

Return the quarter of the year for the date:

SELECT QUARTER(CURDATE());

Number of Records: 1

|  |
| --- |
| **QUARTER(CURDATE())** |
| 4 |

#   SECOND()

Return the seconds part of a datetime value:

SELECT SECOND("2017-06-20 09:34:00.000023");

The SECOND() function returns the seconds part of a time/datetime (from 0 to 59).

SECOND(datetime)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| datetime | Required. The time or datetime to extract the second from |

Return the seconds part of a time value:

SELECT SECOND("23:59:59");

Number of Records: 1

|  |
| --- |
| **SECOND("23:59:59")** |
| 59 |

#   SEC\_TO\_TIME()

Return a time value based on a specified seconds value:

SELECT SEC\_TO\_TIME(1);

Number of Records: 1

|  |
| --- |
| **SEC\_TO\_TIME(1)** |
| 00:00:01 |

The SEC\_TO\_TIME() function returns a time value (in format HH:MM:SS) based on the specified seconds.

SEC\_TO\_TIME(seconds)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| seconds | Required. The number of seconds. Both positive or negative values are allowed |

Return a time value based on a specified seconds value:

SELECT SEC\_TO\_TIME(-6897);

Number of Records: 1

|  |
| --- |
| **SEC\_TO\_TIME(-6897)** |
| -01:54:57 |

#   STR\_TO\_DATE()

Return a date based on a string and a format:

SELECT STR\_TO\_DATE("August 10 2017", "%M %d %Y");

The STR\_TO\_DATE() function returns a date based on a string and a format.

STR\_TO\_DATE(string, format)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string | Required. The string to be formatted to a date |
| format | Required. The format to use. Can be one or a combination of the following values:

|  |  |
| --- | --- |
| **Format** | **Description** |
| %a | Abbreviated weekday name (Sun to Sat) |
| %b | Abbreviated month name (Jan to Dec) |
| %c | Numeric month name (0 to 12) |
| %D | Day of the month as a numeric value, followed by suffix (1st, 2nd, 3rd, ...) |
| %d | Day of the month as a numeric value (01 to 31) |
| %e | Day of the month as a numeric value (0 to 31) |
| %f | Microseconds (000000 to 999999) |
| %H | Hour (00 to 23) |
| %h | Hour (00 to 12) |
| %I | Hour (00 to 12) |
| %i | Minutes (00 to 59) |
| %j | Day of the year (001 to 366) |
| %k | Hour (0 to 23) |
| %l | Hour (1 to 12) |
| %M | Month name in full (January to December) |
| %m | Month name as a numeric value (01 to 12) |
| %p | AM or PM |
| %r | Time in 12 hour AM or PM format (hh:mm:ss AM/PM) |
| %S | Seconds (00 to 59) |
| %s | Seconds (00 to 59) |
| %T | Time in 24 hour format (hh:mm:ss) |
| %U | Week where Sunday is the first day of the week (00 to 53) |
| %u | Week where Monday is the first day of the week (00 to 53) |
| %V | Week where Sunday is the first day of the week (01 to 53). Used with %X |
| %v | Week where Monday is the first day of the week (01 to 53). Used with %X |
| %W | Weekday name in full (Sunday to Saturday) |
| %w | Day of the week where Sunday=0 and Saturday=6 |
| %X | Year for the week where Sunday is the first day of the week. Used with %V |
| %x | Year for the week where Monday is the first day of the week. Used with %V |
| %Y | Year as a numeric, 4-digit value |
| %y | Year as a numeric, 2-digit value |

 |

Return a date based on a string and a format:

SELECT STR\_TO\_DATE("August,5,2017", "%M %e %Y");

Return a date based on a string and a format:

SELECT STR\_TO\_DATE("Monday, August 14, 2017", "%W %M %e %Y");

Return a date based on a string and a format:

SELECT STR\_TO\_DATE("2017,8,14 10,40,10", "%Y,%m,%d %h,%i,%s");

|  |
| --- |
| **STR\_TO\_DATE("2017,8,14 10,40,10", "%Y,%m,%d %h,%i,%s")** |
| 2017-08-14 10:40:10 |

#   SUBDATE()

Subtract 10 days from a date and return the date:

SELECT SUBDATE("2017-06-15", INTERVAL 10 DAY);

The SUBDATE() function subtracts a time/date interval from a date and then returns the date.

SUBDATE(date, INTERVAL value unit)

OR:

SUBDATE(date, days)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The original date |
| days | Required. The number of days to subtract from date |
| value | Required. The value of the time/date interval to subtract. Both positive and negative values are allowed |
| unit | Required. The type of interval. Can be one of the following values:* MICROSECOND SECOND MINUTE HOUR
* DAY WEEK MONTH QUARTER YEAR
* SECOND\_MICROSECOND MINUTE\_MICROSECOND
* MINUTE\_SECOND HOUR\_MICROSECOND
* HOUR\_SECOND HOUR\_MINUTE
* DAY\_MICROSECOND DAY\_SECOND
* DAY\_MINUTE DAY\_HOUR YEAR\_MONTH
 |

Subtract 15 minutes from a date and return the date:

SELECT SUBDATE("2017-06-15 09:34:21", INTERVAL 15 MINUTE);

|  |
| --- |
| **SUBDATE("2017-06-15 09:34:21", INTERVAL 15 MINUTE)** |
| 2017-06-15 09:19:21 |

Subtract 3 hours from a date and return the date:

SELECT SUBDATE("2017-06-15 09:34:21", INTERVAL 3 HOUR);

|  |
| --- |
| **SUBDATE("2017-06-15 09:34:21", INTERVAL 3 HOUR)** |
| 2017-06-15 06:34:21 |

Add 2 months to a date and return the date:

SELECT SUBDATE("2017-06-15", INTERVAL -2 MONTH);

Number of Records: 1

|  |
| --- |
| **SUBDATE("2017-06-15", INTERVAL -2 MONTH)** |
| 2017-08-15 |

#   SUBTIME()

Subtract 5.000001 seconds and return the datetime:

SELECT SUBTIME("2017-06-15 10:24:21.000004", "5.000001");

|  |
| --- |
| **SUBTIME("2017-06-15 10:24:21.000004", "5.000001")** |
| 2017-06-15 10:24:16.000003 |

The SUBTIME() function subtracts time from a time/datetime expression and then returns the new time/datetime.

SUBTIME(datetime, time\_interval)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| datetime | Required. The time or datetime to be modified |
| time\_interval | Required. The time interval to subtract from datetime. Both positive and negative values are allowed |

Subtract 3 hours, 2 minutes, 5.000001 seconds and return the datetime:

SELECT SUBTIME("2017-06-15 10:24:21.000004", "3:2:5.000001");

|  |
| --- |
| **SUBTIME("2017-06-15 10:24:21.000004", "3:2:5.000001")** |
| 2017-06-15 07:22:16.000003 |

Subtract 5 seconds and return the time:

SELECT SUBTIME("10:24:21", "5");

|  |
| --- |
| **SUBTIME("10:24:21", "5")** |
| 10:24:16 |

Subtract 3 minutes and return the time:

SELECT SUBTIME("10:24:21", "100"), SUBTIME("10:24:21", "200"), SUBTIME("10:24:21", "300");

|  |  |  |
| --- | --- | --- |
| SUBTIME("10:24:21", "100") | SUBTIME("10:24:21", "200") | SUBTIME("10:24:21", "300") |
| 10:23:21 | 10:22:21 | 10:21:21 |

Add 3 hours, 2 minutes, and 5 seconds, and return the time:

SELECT SUBTIME("10:24:21", "-3:2:5");

|  |
| --- |
| **SUBTIME("10:24:21", "-3:2:5")** |
| 13:26:26 |

#   SYSDATE()

Return the current date and time:

SELECT SYSDATE();

|  |
| --- |
| **SYSDATE()** |
| 2020-11-18 18:22:11 |

The SYSDATE() function returns the current date and time.

**Note:** The date and time is returned as "YYYY-MM-DD HH:MM:SS" (string) or as YYYYMMDDHHMMSS (numeric).

SYSDATE()

Return the current date and time + 1, 10, 100:

SELECT SYSDATE() + 1, SYSDATE() + 10,SYSDATE() + 100;

|  |  |  |
| --- | --- | --- |
| **SYSDATE() + 1** | **SYSDATE() + 10** | **SYSDATE() + 100** |
| 20201118182338 | 20201118182347 | 20201118182437 |

#   TIME()

Extract the time part from a time expression:

SELECT TIME("19:30:10");

|  |
| --- |
| **TIME("19:30:10")** |
| 19:30:10 |

The TIME() function extracts the time part from a given time/datetime.

**Note:** This function returns "00:00:00" if expression is not a datetime/time, or NULL if expression is NULL.

TIME(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Required. The time/datetime to extract the time from |

Extract the time part from the datetime expression:

SELECT TIME("2017-08-15 19:30:10");

|  |
| --- |
| **TIME("2017-08-15 19:30:10")** |
| 19:30:10 |

Extract the time part from the datetime expression:

SELECT TIME("2017-08-15 19:30:10.000001");

|  |
| --- |
| **TIME("2017-08-15 19:30:10.000001")** |
| 19:30:10.000001 |

Extract the time part from NULL:

SELECT TIME(NULL), TIME(NULL)+100;

|  |  |
| --- | --- |
| **TIME(NULL)** | **TIME(NULL)+100** |

#   TIME\_FORMAT()

Format a time:

SELECT TIME\_FORMAT("19:30:10", "%H %i %s");

|  |
| --- |
| **TIME\_FORMAT("19:30:10", "%H %i %s")** |
| 19 30 10 |

The TIME\_FORMAT() function formats a time by a specified format.

TIME\_FORMAT(time, format)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| time | Required. The time to be formatted |
| format | Required. The format to use. Can be one or a combination of the following:

|  |  |
| --- | --- |
| **Format** | **Description** |
| %f | Microseconds (000000 to 999999) |
| %H | Hour (00 to 23) |
| %h | Hour (00 to 12) |
| %I | Hour (00 to 12) |
| %i | Minutes (00 to 59) |
| %p | AM or PM |
| %r | Time in 12 hour AM or PM format (hh:mm:ss AM/PM) |
| %S | Seconds (00 to 59) |
| %s | Seconds (00 to 59) |
| %T | Time in 24 hour format (hh:mm:ss) |

 |

Format a time:

SELECT TIME\_FORMAT("19:30:10", "%h %i %s %p");

|  |
| --- |
| **TIME\_FORMAT("19:30:10", "%h %i %s %p")** |
| 07 30 10 PM |

Format a time:

SELECT TIME\_FORMAT("19:30:10", "%r");

|  |
| --- |
| **TIME\_FORMAT("19:30:10", "%r")** |
| 07:30:10 PM |

Format a time:

SELECT TIME\_FORMAT("19:30:10", "%T");

|  |
| --- |
| **TIME\_FORMAT("19:30:10", "%T")** |
| 19:30:10 |

#   TIME\_TO\_SEC()

[❮ MySQL Functions](https://www.w3schools.com/sql/sql_ref_mysql.asp)

Convert a time value into seconds:

SELECT TIME\_TO\_SEC("19:30:10");

[Try it Yourself »](https://www.w3schools.com/sql/trymysql.asp?filename=trysql_func_mysql_time_to_sec)

The TIME\_TO\_SEC() function converts a time value into seconds.

TIME\_TO\_SEC(time)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| time | Required. The time value |

Convert a time value into seconds:

SELECT TIME\_TO\_SEC("00:00:05");

|  |
| --- |
| **TIME\_TO\_SEC("00:00:05")** |
| 5 |

Convert a time value into seconds:

SELECT TIME\_TO\_SEC("03:30:00.999999");

|  |
| --- |
| **TIME\_TO\_SEC("03:30:00.999999")** |
| 12600 |

Convert a time value into seconds:

SELECT TIME\_TO\_SEC("-03:30:00");

|  |
| --- |
| **TIME\_TO\_SEC("-03:30:00")** |
| -12600 |

#   TIMEDIFF()

Return the difference between two time expressions:

SELECT TIMEDIFF("13:10:11", "13:10:10");

|  |
| --- |
| **TIMEDIFF("13:10:11", "13:10:10")** |
| 00:00:01 |

The TIMEDIFF() function returns the difference between two time/datetime expressions.

**Note:** time1 and time2 should be in the same format, and the calculation is time1 - time2.

TIMEDIFF(time1, time2)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| time1 | Required. A time value |
| time2 | Required. Another time value |

Return the difference between two datetime expressions:

SELECT TIMEDIFF("2017-06-25 13:10:11", "2017-06-15 13:10:10");

|  |
| --- |
| **TIMEDIFF("2017-06-25 13:10:11", "2017-06-15 13:10:10")** |
| 240:00:01 |

#   TIMESTAMP()

Return a datetime value based on the arguments:

SELECT TIMESTAMP("2017-07-23",  "13:10:11");

|  |
| --- |
| **TIMESTAMP("2017-07-23", "13:10:11")** |
| 2017-07-23 13:10:11 |

The TIMESTAMP() function returns a datetime value based on a date or datetime value.

**Note:** If there are specified two arguments with this function, it first adds the second argument to the first, and then returns a datetime value.

## Syntax

TIMESTAMP(expression, time)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Required. A date or datetime value |
| time | Optional. A time value to add to expression |

Return a datetime value based on the arguments:

SELECT TIMESTAMP("2017-07-23");

|  |
| --- |
| **TIMESTAMP("2017-07-23")** |
| 2017-07-23 00:00:00 |

#   TO\_DAYS()

Return the number of days between the date and year 0:

SELECT TO\_DAYS("2017-06-20");

|  |
| --- |
| **TO\_DAYS("2017-06-20")** |
| 736865 |

The TO\_DAYS() function returns the number of days between a date and year 0 (date "0000-00-00").

The TO\_DAYS() function can be used only with dates within the Gregorian calendar.

**Note:** This function is the opposite of the [FROM\_DAYS()](https://www.w3schools.com/sql/func_mysql_from_days.asp) function.

TO\_DAYS(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required. The given date |

Return the number of days between the date and year 0:

SELECT TO\_DAYS("2017-06-20 09:34:00");

|  |
| --- |
| **TO\_DAYS("2017-06-20 09:34:00")** |
| 736865 |

SELECT TO\_DAYS(now());

|  |
| --- |
| **TO\_DAYS(now())** |
| 738112 |

#   WEEK()

Return the week number for a date:

SELECT WEEK("2017-06-15");

The WEEK() function returns the week number for a given date (a number from 0 to 53).

WEEK(date, firstdayofweek)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required.  The date or datetime to extract the week number form |
| firstdayofweek | Optional. Specifies what day the week starts on. Can be one of the following:* 0 - First day of week is Sunday
* 1 - First day of week is Monday and the first week
* of the year has more than 3 days
* 2 - First day of week is Sunday
* 3 - First day of week is Monday and the first week
* of the year has more than 3 days
* 4 - First day of week is Sunday and the first week
* of the year has more than 3 days
* 5 - First day of week is Monday
* 6 - First day of week is Sunday and the first week
* of the year has more than 3 days
* 7 - First day of week is Monday
 |

Return the week number for a date:

SELECT WEEK("2017-10-25");

|  |
| --- |
| **WEEK("2017-10-25")** |
| 43 |

#  WEEKDAY()

Return the weekday number for a date:

SELECT WEEKDAY("2017-06-15");

|  |
| --- |
| **WEEKDAY("2017-06-15")** |
| 3 |

The WEEKDAY() function returns the weekday number for a given date.

**Note:** 0 = Monday, 1 = Tuesday, 2 = Wednesday, 3 = Thursday, 4 = Friday, 5 = Saturday, 6 = Sunday.

WEEKDAY(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required.  The date or datetime to extract the weekday number from |

Return the weekday number for a date:

SELECT WEEKDAY("2017-01-01");

Return the weekday number for the current system date:

SELECT WEEKDAY(CURDATE());

|  |
| --- |
| **WEEKDAY(CURDATE())** |
| 2 |

#  WEEKOFYEAR()

Return the week number for a date:

SELECT WEEKOFYEAR("2017-06-15");

|  |
| --- |
| **WEEKOFYEAR("2017-06-15")** |
| 24 |

The WEEKOFYEAR() function returns the week number for a given date (a number from 1 to 53).

**Note:** This function assumes that the first day of the week is Monday and the first week of the year has more than 3 days.

**Tip:** Also look at the [WEEK()](https://www.w3schools.com/sql/func_mysql_week.asp) function.

WEEKOFYEAR(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required.  The date or datetime to extract the week number from |

Return the week number for a date:

SELECT WEEKOFYEAR("2017-01-01");

|  |
| --- |
| **WEEKOFYEAR("2017-01-01")** |
| 52 |

Return the week number for the current system date:

SELECT WEEKOFYEAR(CURDATE());

|  |
| --- |
| **WEEKOFYEAR(CURDATE())** |
| 47 |

#   YEAR()

Return the year part of a date:

SELECT YEAR("2017-06-15");

|  |
| --- |
| **YEAR("2017-06-15")** |
| 2017 |

The YEAR() function returns the year part for a given date (a number from 1000 to 9999).

YEAR(date)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required.  The date/datetime to extract the year from |

Return the year part of a date:

SELECT YEAR("2017-06-15 09:34:21");

|  |
| --- |
| **YEAR("2017-06-15 09:34:21")** |
| 2017 |

Return the year part of the current system date:

SELECT YEAR(CURDATE());

|  |
| --- |
| **YEAR(CURDATE())** |
| 2020 |

#   YEARWEEK()

Return the year and week number for a date:

SELECT YEARWEEK("2017-06-15");

|  |
| --- |
| **YEARWEEK("2017-06-15")** |
| 201724 |

The YEARWEEK() function returns the year and week number (a number from 0 to 53) for a given date.

YEARWEEK(date, firstdayofweek)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| date | Required.  The date or datetime value to extract the year and week number from |
| firstdayofweek | Optional. Specifies what day the week starts on. Can be one of the following:* 0 - First day of week is Sunday
* 1 - First day of week is Monday and the first week
* has more than 3 days
* 2 - First day of week is Sunday
* 3 - First day of week is Monday and the first week
* has more than 3 days
* 4 - First day of week is Sunday and the first
* week has more than 3 days
* 5 - First day of week is Monday
* 6 - First day of week is Sunday and the first
* week has more than 3 days
* 7 - First day of week is Monday
 |

Return the year and week number for a date:

SELECT YEARWEEK("2017-10-25");

|  |
| --- |
| **YEARWEEK("2017-10-25")** |
| 201743 |

Return the year and week number for the current system date:

SELECT YEARWEEK(CURDATE());

|  |
| --- |
| **YEARWEEK(CURDATE())** |
| 202046 |

*Advanced Functions*

#   BIN()

Return a binary representation of 15:

SELECT BIN(15);

The BIN() function returns a binary representation of a number, as a string value.

BIN(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A number |

Return a binary representation of 111:

SELECT BIN(111);

Return a binary representation of 8:

SELECT BIN(8);

|  |
| --- |
| **BIN(8)** |
| 1000 |

#   BINARY

Convert a value to a binary string:

SELECT BINARY "W3Schools.com";

|  |
| --- |
| **BINARY "W3Schools.com"** |
| W3Schools.com |

The BINARY function converts a value to a binary string.

This function is equivalent to using [CAST(value AS BINARY)](https://www.w3schools.com/sql/func_mysql_cast.asp).

BINARY value

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| value | Required. The value to convert |

Here MySQL performs a character-by-character comparison of  "HELLO" and "hello" and return 1 (because on a character-by-character basis, they are equivalent):

SELECT "HELLO" = "hello";

Here MySQL performs a byte-by-byte comparison of  "HELLO" and "hello" and return 0 (because on a byte-by-byte basis, they are NOT equivalent):

SELECT BINARY "HELLO" = "hello";

#  CASE

Go through conditions and return a value when the first condition is met:

SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN "The quantity is greater than 30"
    WHEN Quantity = 30 THEN "The quantity is 30"
    ELSE "The quantity is under 30"
END
FROM OrderDetails;

#   CAST()

Convert a value to a DATE datatype:

SELECT CAST("2017-08-29" AS DATE);

The CAST() function converts a value (of any type) into the specified datatype.

**Tip:** See also the [CONVERT()](https://www.w3schools.com/sql/func_mysql_convert.asp) function.

CAST(value AS datatype)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| value | Required. The value to convert |
| datatype | Required. The datatype to convert to. Can be one of the following:

|  |  |
| --- | --- |
| **Value** | **Description** |
| DATE | Converts value to DATE. Format: "YYYY-MM-DD" |
| DATETIME | Converts value to DATETIME. Format: "YYYY-MM-DD HH:MM:SS" |
| TIME | Converts value to TIME. Format: "HH:MM:SS" |
| CHAR | Converts value to CHAR (a fixed length string) |
| SIGNED | Converts value to SIGNED (a signed 64-bit integer) |
| UNSIGNED | Converts value to UNSIGNED (an unsigned 64-bit integer) |
| BINARY | Converts value to BINARY (a binary string) |

 |

Convert a value to a CHAR datatype:

SELECT CAST(150 AS CHAR);

|  |
| --- |
| **CAST(150 AS CHAR)** |
| 150 |

#   COALESCE()

Return the first non-null value in a list:

SELECT COALESCE(NULL, NULL, NULL, 'W3Schools.com', NULL, 'Example.com');

|  |
| --- |
| **COALESCE(NULL, NULL, NULL, 'W3Schools.com', NULL, 'Example.com')** |
| W3Schools.com |

The COALESCE() function returns the first non-null value in a list.

COALESCE(val1, val2, ...., val\_n)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| val1, val2, val\_n | Required. The values to test |

Return the first non-null value in a list:

SELECT COALESCE(NULL, 1, 2, 'W3Schools.com');

|  |
| --- |
| **COALESCE(NULL, 1, 2, 'W3Schools.com')** |
| 1 |

#   CONV()

Convert a number from numeric base system 10 to numeric base system 2:

SELECT CONV(15, 10, 2);

The CONV() function converts a number from one numeric base system to another, and returns the result as a string value.

**Note:** This function returns NULL if any of the parameters are NULL.

**Tip:** Also look at the [BIN()](https://www.w3schools.com/sql/func_mysql_bin.asp) function.

CONV(number, from\_base, to\_base)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. A number |
| from\_base | The numeric base system of number (a number between 2 and 36) |
| to\_base | The numeric base system to convert to (a number between 2 and 36 or -2 and -36) |

Convert a number from numeric base system 2 to numeric base system 10:

SELECT CONV(1111, 2, 10);

Convert a number from numeric base system 10 to numeric base system 16:

SELECT CONV(88, 10, 16);

|  |
| --- |
| **CONV(88, 10, 16)** |
| 58 |

#   CONVERT()

Convert a value to a DATE datatype:

SELECT CONVERT("2017-08-29", DATE);

|  |
| --- |
| **CONVERT("2017-08-29", DATE)** |
| 2017-08-29 |

The CONVERT() function converts a value into the specified datatype or character set.

**Tip:** Also look at the [CAST()](https://www.w3schools.com/sql/func_mysql_cast.asp) function.

## Syntax

CONVERT(value, type)

OR:

CONVERT(value USING charset)

## Parameter Values

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| value | Required. The value to convert |
| type | Required. The datatype to convert to. Can be one of the following:

|  |  |
| --- | --- |
| **Value** | **Description** |
| DATE | Converts value to DATE. Format: "YYYY-MM-DD" |
| DATETIME | Converts value to DATETIME. Format: "YYYY-MM-DD HH:MM:SS" |
| TIME | Converts value to TIME. Format: "HH:MM:SS" |
| CHAR | Converts value to CHAR (a fixed length string) |
| SIGNED | Converts value to SIGNED (a signed 64-bit integer) |
| UNSIGNED | Converts value to UNSIGNED (an unsigned 64-bit integer) |
| BINARY | Converts value to BINARY (a binary string) |

 |
| charset | Required. The character set to convert to |

Convert a value to a CHAR datatype:

SELECT CONVERT(150, CHAR);

|  |
| --- |
| **CONVERT(150, CHAR)** |
| 150 |

Convert a value to a TIME datatype:

SELECT CONVERT("14:06:10", TIME);

|  |
| --- |
| **CONVERT("14:06:10", TIME)** |
| 14:06:10 |

Convert a value to LATIN1 character set:

SELECT CONVERT("W3Schools.com" USING latin1);

[Try it Yourself »](https://www.w3schools.com/sql/trymysql.asp?filename=trysql_func_mysql_convert4)

|  |
| --- |
| **CONVERT("W3Schools.com" USING latin1)** |
| W3Schools.com |

#   CURRENT\_USER()

Return the user name and host name for the MySQL account:

SELECT CURRENT\_USER();

|  |
| --- |
| **CURRENT\_USER()** |
| guest@% |

The CURRENT\_USER() function returns the user name and host name for the MySQL account that the server used to authenticate the current client.

The result is returned as a string in the UTF8 character set.

**Tip:** See also the [USER()](https://www.w3schools.com/sql/func_mysql_user.asp) function.

CURRENT\_USER()

#   DATABASE()

Return the name of the current (default) database:

SELECT DATABASE();

|  |
| --- |
| **DATABASE()** |
| mydb |

The DATABASE() function returns the name of the current database.

If there is no current database, this function returns NULL or "".

DATABASE()

#   IF()

Return "YES" if the condition is TRUE, or "NO" if the condition is FALSE:

SELECT IF(500<1000, "YES", "NO");

|  |
| --- |
| **IF(500<1000, "YES", "NO")** |
| YES |

The IF() function returns a value if a condition is TRUE, or another value if a condition is FALSE.

IF(condition, value\_if\_true, value\_if\_false)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| condition | Required. The value to test |
| value\_if\_true | Optional. The value to return if condition is TRUE |
| value\_if\_false | Optional. The value to return if condition is FALSE |

Return 5 if the condition is TRUE, or 10 if the condition is FALSE:

SELECT IF(500<1000, 5, 10);

|  |
| --- |
| **IF(500<1000, 5, 10)** |
| 5 |

Test whether two strings are the same and return "YES" if they are, or "NO" if not:

SELECT IF(STRCMP("hello","bye") = 0, "YES", "NO");

|  |
| --- |
| **IF(STRCMP("hello","bye") = 0, "YES", "NO")** |
| NO |

Return "MORE" if the condition is TRUE, or "LESS" if the condition is FALSE:

SELECT OrderID, Quantity, IF(Quantity>10, "MORE", "LESS")
FROM OrderDetails;

#   IFNULL()

Return the specified value IF the expression is NULL, otherwise return the expression:

SELECT IFNULL(NULL, "W3Schools.com");

The IFNULL() function returns a specified value if the expression is NULL.

If the expression is NOT NULL, this function returns the expression.

IFNULL(expression, alt\_value)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Required. The expression to test whether is NULL |
| alt\_value | Required. The value to return if expression is NULL |

Return the specified value IF the expression is NULL, otherwise return the expression:

SELECT IFNULL("Hello", "W3Schools.com");

|  |
| --- |
| **IFNULL(NULL, "W3Schools.com")** |
| W3Schools.com |

Return the specified value IF the expression is NULL, otherwise return the expression:

SELECT IFNULL(NULL, 500);

|  |
| --- |
| **IFNULL(NULL, 500)** |
| 500 |

#   ISNULL()

Test whether an expression is NULL:

SELECT ISNULL(NULL);

|  |
| --- |
| **ISNULL(NULL)** |
| 1 |

The ISNULL() function returns 1 or 0 depending on whether an expression is NULL.

If expression is NULL, this function returns 1. Otherwise, it returns 0.

ISNULL(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Required. The value to test |

Test whether an expression is NULL:

SELECT ISNULL("");

|  |
| --- |
| **ISNULL("")** |
| 0 |

Test whether an expression is NULL:

SELECT ISNULL(350);

|  |
| --- |
| **ISNULL(350)** |
| 0 |

Test whether an expression is NULL:

SELECT ISNULL("Hello world!");

|  |
| --- |
| **ISNULL("Hello world!")** |
| 0 |

#   LAST\_INSERT\_ID()

Return the AUTO\_INCREMENT id of the last row that has been inserted or updated in a table:

SELECT LAST\_INSERT\_ID();

|  |
| --- |
| **LAST\_INSERT\_ID()** |
| 0 |

The LAST\_INSERT\_ID() function returns the AUTO\_INCREMENT id of the last row that has been inserted or updated in a table.

LAST\_INSERT\_ID(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Optional. An expression |

#   NULLIF()

Compare two expressions:

SELECT NULLIF(25, 25);

**NULLIF(25, 25)**

The NULLIF() function compares two expressions and returns NULL if they are equal. Otherwise, the first expression is returned.

NULLIF(expr1, expr2)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expr1, expr2 | Required. The two expressions to be compared |

Compare two expressions:

SELECT NULLIF(25, "Hello");

|  |
| --- |
| **NULLIF(25, "Hello")** |
| 25 |

Compare two expressions:

SELECT NULLIF("Hello", "world");

|  |
| --- |
| **NULLIF("Hello", "world")** |
| Hello |

Compare two expressions:

SELECT NULLIF("2017-08-25", "2017-08-25");

|  |
| --- |
| **NULLIF("2017-08-25", "2017-08-25")** |
|  |

#   SESSION\_USER()

Return the current user name and host name for the MySQL connection:

SELECT SESSION\_USER();

|  |
| --- |
| **SESSION\_USER()** |
| guest@35.192.20.199 |

The SESSION\_USER() function returns the current user name and host name for the MySQL connection.

**Note:** This function is equal to the [SYSTEM\_USER()](https://www.w3schools.com/sql/func_mysql_system_user.asp) and the [USER()](https://www.w3schools.com/sql/func_mysql_user.asp) function.

SESSION\_USER()

#   SYSTEM\_USER()

Return the current user name and host name for the MySQL connection:

SELECT SYSTEM\_USER();

|  |
| --- |
| **[SYSTEM\_USER()](https://www.w3schools.com/sql/trymysql.asp?filename=trysql_func_mysql_system_user" \t "_blank)** |
| [guest@35.192.20.199](https://www.w3schools.com/sql/trymysql.asp?filename=trysql_func_mysql_system_user" \t "_blank) |

The SYSTEM\_USER() function returns the current user name and host name for the MySQL connection.

**Note:** This function is equal to the [SESSION\_USER()](https://www.w3schools.com/sql/func_mysql_session_user.asp) function and the [USER()](https://www.w3schools.com/sql/func_mysql_user.asp) function.

## Syntax

SYSTEM\_USER()

#   USER()

Return the current user name and host name for the MySQL connection:

SELECT USER();

|  |
| --- |
| **USER()** |
| guest@35.192.20.199 |

The USER() function returns the current user name and host name for the MySQL connection.

**Note:** This function is equal to the [SESSION\_USER()](https://www.w3schools.com/sql/func_mysql_session_user.asp) function and the [SYSTEM\_USER()](https://www.w3schools.com/sql/func_mysql_system_user.asp) function.

**Tip:** Also look at the [CURRENT\_USER()](https://www.w3schools.com/sql/func_mysql_current_user.asp) function.

## Syntax

USER()

#   VERSION()

Return the current version of the MySQL database:

SELECT VERSION();

|  |
| --- |
| **VERSION()** |
| 5.6.44 |

The VERSION() function returns the current version of the MySQL database, as a string.