# 函数和operator

Expressions can be used at several points in [SQL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_sql) statements, such as in the **ORDER BY** or **HAVING** clauses of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, in the **WHERE** clause of a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete), or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement, or in [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statements. Expressions can be written using literal values, column values, **NULL**, built-in functions, stored functions, user-defined functions, and operators. This chapter describes the SQL functions and operators that are permitted for writing expressions in MySQL. Instructions for writing stored functions and user-defined functions are given in [Section 25.2, “Using Stored Routines”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-routines), and [Adding Functions to MySQL](https://dev.mysql.com/doc/extending-mysql/8.0/en/adding-functions.html). See [Section 9.2.5, “Function Name Parsing and Resolution”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#function-resolution), for the rules describing how the server interprets references to different kinds of functions.

An expression that contains **NULL** always produces a **NULL** value unless otherwise indicated in the documentation for a particular function or operator.

**Note**

By default, there must be no whitespace between a function name and the parenthesis following it. This helps the MySQL parser distinguish between function calls and references to tables or columns that happen to have the same name as a function. However, spaces around function arguments are permitted.

You can tell the MySQL server to accept spaces after function names by starting it with the [--sql-mode=IGNORE\_SPACE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_sql-mode) option. (See [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode).) Individual client programs can request this behavior by using the **CLIENT\_IGNORE\_SPACE** option for [**mysql\_real\_connect()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-connect.html). In either case, all function names become reserved words.

For the sake of brevity, most examples in this chapter display the output from the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) program in abbreviated form. Rather than showing examples in this format:

mysql> **SELECT MOD(29,9);**

+-----------+

| mod(29,9) |

+-----------+

| 2 |

+-----------+

1 rows in set (0.00 sec)

This format is used instead:

mysql> **SELECT MOD(29,9);**

-> 2

## 12.1 SQL Function and Operator Reference

The following table lists each SQL function and operator and provides a short description of each one. For a table listing user-defined functions, see [Section 12.2, “User-Defined Function Reference”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#udf-reference).

**Table 12.1 SQL Functions and Operators**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Introduced | Deprecated |
| [&](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-and) | Bitwise AND |  |  |
| [>](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_greater-than) | Greater than operator |  |  |
| [>>](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_right-shift) | Right shift |  |  |
| [>=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_greater-than-or-equal) | Greater than or equal operator |  |  |
| [<](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_less-than) | Less than operator |  |  |
| [<>, !=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-equal) | Not equal operator |  |  |
| [<<](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_left-shift) | Left shift |  |  |
| [<=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_less-than-or-equal) | Less than or equal operator |  |  |
| [<=>](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal-to) | NULL-safe equal to operator |  |  |
| [%, MOD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_mod) | Modulo operator |  |  |
| [\*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_times) | Multiplication operator |  |  |
| [+](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_plus) | Addition operator |  |  |
| [-](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_minus) | Minus operator |  |  |
| [-](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_unary-minus) | Change the sign of the argument |  |  |
| [->](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-column-path) | Return value from JSON column after evaluating path; equivalent to JSON\_EXTRACT(). |  |  |
| [->>](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-inline-path) | Return value from JSON column after evaluating path and unquoting the result; equivalent to JSON\_UNQUOTE(JSON\_EXTRACT()). |  |  |
| [/](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_divide) | Division operator |  |  |
| [:=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-value) | Assign a value |  |  |
| [=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-equal) | Assign a value (as part of a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement, or as part of the **SET** clause in an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement) |  |  |
| [=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal) | Equal operator |  |  |
| [^](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-xor) | Bitwise XOR |  |  |
| [ABS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_abs) | Return the absolute value |  |  |
| [ACOS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_acos) | Return the arc cosine |  |  |
| [ADDDATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_adddate) | Add time values (intervals) to a date value |  |  |
| [ADDTIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_addtime) | Add time |  |  |
| [AES\_DECRYPT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-decrypt) | Decrypt using AES |  |  |
| [AES\_ENCRYPT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-encrypt) | Encrypt using AES |  |  |
| [AND, &&](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_and) | Logical AND |  |  |
| [ANY\_VALUE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) | Suppress ONLY\_FULL\_GROUP\_BY value rejection |  |  |
| [ASCII()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ascii) | Return numeric value of left-most character |  |  |
| [ASIN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_asin) | Return the arc sine |  |  |
| [ATAN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_atan) | Return the arc tangent |  |  |
| [ATAN2(), ATAN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_atan2) | Return the arc tangent of the two arguments |  |  |
| [AVG()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_avg) | Return the average value of the argument |  |  |
| [BENCHMARK()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_benchmark) | Repeatedly execute an expression |  |  |
| [BETWEEN ... AND ...](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_between) | Whether a value is within a range of values |  |  |
| [BIN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bin) | Return a string containing binary representation of a number |  |  |
| [BIN\_TO\_UUID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bin-to-uuid) | Convert binary UUID to string |  |  |
| [BINARY](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_binary) | Cast a string to a binary string |  |  |
| [BIT\_AND()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-and) | Return bitwise AND |  |  |
| [BIT\_COUNT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-count) | Return the number of bits that are set |  |  |
| [BIT\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-length) | Return length of argument in bits |  |  |
| [BIT\_OR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-or) | Return bitwise OR |  |  |
| [BIT\_XOR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-xor) | Return bitwise XOR |  |  |
| [CAN\_ACCESS\_COLUMN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_can-access-column) | Internal use only |  |  |
| [CAN\_ACCESS\_DATABASE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_can-access-database) | Internal use only |  |  |
| [CAN\_ACCESS\_TABLE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_can-access-table) | Internal use only |  |  |
| [CAN\_ACCESS\_USER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_can-access-user) | Internal use only | 8.0.22 |  |
| [CAN\_ACCESS\_VIEW()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_can-access-view) | Internal use only |  |  |
| [CASE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case) | Case operator |  |  |
| [CAST()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) | Cast a value as a certain type |  |  |
| [CEIL()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ceil) | Return the smallest integer value not less than the argument |  |  |
| [CEILING()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ceiling) | Return the smallest integer value not less than the argument |  |  |
| [CHAR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char) | Return the character for each integer passed |  |  |
| [CHAR\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char-length) | Return number of characters in argument |  |  |
| [CHARACTER\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_character-length) | Synonym for CHAR\_LENGTH() |  |  |
| [CHARSET()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_charset) | Return the character set of the argument |  |  |
| [COALESCE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_coalesce) | Return the first non-NULL argument |  |  |
| [COERCIBILITY()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_coercibility) | Return the collation coercibility value of the string argument |  |  |
| [COLLATION()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_collation) | Return the collation of the string argument |  |  |
| [COMPRESS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_compress) | Return result as a binary string |  |  |
| [CONCAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat) | Return concatenated string |  |  |
| [CONCAT\_WS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat-ws) | Return concatenate with separator |  |  |
| [CONNECTION\_ID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_connection-id) | Return the connection ID (thread ID) for the connection |  |  |
| [CONV()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_conv) | Convert numbers between different number bases |  |  |
| [CONVERT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) | Cast a value as a certain type |  |  |
| [CONVERT\_TZ()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert-tz) | Convert from one time zone to another |  |  |
| [COS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cos) | Return the cosine |  |  |
| [COT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cot) | Return the cotangent |  |  |
| [COUNT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count) | Return a count of the number of rows returned |  |  |
| [COUNT(DISTINCT)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count-distinct) | Return the count of a number of different values |  |  |
| [CRC32()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_crc32) | Compute a cyclic redundancy check value |  |  |
| [CUME\_DIST()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cume-dist) | Cumulative distribution value |  |  |
| [CURDATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_curdate) | Return the current date |  |  |
| [CURRENT\_DATE(), CURRENT\_DATE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-date) | Synonyms for CURDATE() |  |  |
| [CURRENT\_ROLE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-role) | Return the current active roles |  |  |
| [CURRENT\_TIME(), CURRENT\_TIME](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-time) | Synonyms for CURTIME() |  |  |
| [CURRENT\_TIMESTAMP(), CURRENT\_TIMESTAMP](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-timestamp) | Synonyms for NOW() |  |  |
| [CURRENT\_USER(), CURRENT\_USER](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) | The authenticated user name and host name |  |  |
| [CURTIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_curtime) | Return the current time |  |  |
| [DATABASE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_database) | Return the default (current) database name |  |  |
| [DATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date) | Extract the date part of a date or datetime expression |  |  |
| [DATE\_ADD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-add) | Add time values (intervals) to a date value |  |  |
| [DATE\_FORMAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-format) | Format date as specified |  |  |
| [DATE\_SUB()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-sub) | Subtract a time value (interval) from a date |  |  |
| [DATEDIFF()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_datediff) | Subtract two dates |  |  |
| [DAY()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_day) | Synonym for DAYOFMONTH() |  |  |
| [DAYNAME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dayname) | Return the name of the weekday |  |  |
| [DAYOFMONTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dayofmonth) | Return the day of the month (0-31) |  |  |
| [DAYOFWEEK()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dayofweek) | Return the weekday index of the argument |  |  |
| [DAYOFYEAR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dayofyear) | Return the day of the year (1-366) |  |  |
| [DEFAULT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_default) | Return the default value for a table column |  |  |
| [DEGREES()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_degrees) | Convert radians to degrees |  |  |
| [DENSE\_RANK()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dense-rank) | Rank of current row within its partition, without gaps |  |  |
| [DIV](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_div) | Integer division |  |  |
| [ELT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_elt) | Return string at index number |  |  |
| [EXP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_exp) | Raise to the power of |  |  |
| [EXPORT\_SET()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_export-set) | Return a string such that for every bit set in the value bits, you get an on string and for every unset bit, you get an off string |  |  |
| [EXTRACT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extract) | Extract part of a date |  |  |
| [ExtractValue()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extractvalue) | Extract a value from an XML string using XPath notation |  |  |
| [FIELD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_field) | Index (position) of first argument in subsequent arguments |  |  |
| [FIND\_IN\_SET()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_find-in-set) | Index (position) of first argument within second argument |  |  |
| [FIRST\_VALUE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_first-value) | Value of argument from first row of window frame |  |  |
| [FLOOR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_floor) | Return the largest integer value not greater than the argument |  |  |
| [FORMAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format) | Return a number formatted to specified number of decimal places |  |  |
| [FORMAT\_BYTES()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format-bytes) | Convert byte count to value with units | 8.0.16 |  |
| [FORMAT\_PICO\_TIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format-pico-time) | Convert time in picoseconds to value with units | 8.0.16 |  |
| [FOUND\_ROWS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) | For a SELECT with a LIMIT clause, the number of rows that would be returned were there no LIMIT clause |  |  |
| [FROM\_BASE64()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-base64) | Decode base64 encoded string and return result |  |  |
| [FROM\_DAYS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-days) | Convert a day number to a date |  |  |
| [FROM\_UNIXTIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-unixtime) | Format Unix timestamp as a date |  |  |
| [GeomCollection()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geomcollection) | Construct geometry collection from geometries |  |  |
| [GeometryCollection()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geometrycollection) | Construct geometry collection from geometries |  |  |
| [GET\_DD\_COLUMN\_PRIVILEGES()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-dd-column-privileges) | Internal use only |  |  |
| [GET\_DD\_CREATE\_OPTIONS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-dd-create-options) | Internal use only |  |  |
| [GET\_DD\_INDEX\_SUB\_PART\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-dd-index-sub-part-length) | Internal use only |  |  |
| [GET\_FORMAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | Return a date format string |  |  |
| [GET\_LOCK()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) | Get a named lock |  |  |
| [GREATEST()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_greatest) | Return the largest argument |  |  |
| [GROUP\_CONCAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_group-concat) | Return a concatenated string |  |  |
| [GROUPING()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) | Distinguish super-aggregate ROLLUP rows from regular rows |  |  |
| [GTID\_SUBSET()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_gtid-subset) | Return true if all GTIDs in subset are also in set; otherwise false. |  |  |
| [GTID\_SUBTRACT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_gtid-subtract) | Return all GTIDs in set that are not in subset. |  |  |
| [HEX()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex) | Hexadecimal representation of decimal or string value |  |  |
| [HOUR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hour) | Extract the hour |  |  |
| [ICU\_VERSION()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_icu-version) | ICU library version |  |  |
| [IF()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if) | If/else construct |  |  |
| [IFNULL()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ifnull) | Null if/else construct |  |  |
| [IN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_in) | Whether a value is within a set of values |  |  |
| [INET\_ATON()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-aton) | Return the numeric value of an IP address |  |  |
| [INET\_NTOA()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-ntoa) | Return the IP address from a numeric value |  |  |
| [INET6\_ATON()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-aton) | Return the numeric value of an IPv6 address |  |  |
| [INET6\_NTOA()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-ntoa) | Return the IPv6 address from a numeric value |  |  |
| [INSERT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_insert) | Insert substring at specified position up to specified number of characters |  |  |
| [INSTR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_instr) | Return the index of the first occurrence of substring |  |  |
| [INTERNAL\_AUTO\_INCREMENT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-auto-increment) | Internal use only |  |  |
| [INTERNAL\_AVG\_ROW\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-avg-row-length) | Internal use only |  |  |
| [INTERNAL\_CHECK\_TIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-check-time) | Internal use only |  |  |
| [INTERNAL\_CHECKSUM()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-checksum) | Internal use only |  |  |
| [INTERNAL\_DATA\_FREE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-data-free) | Internal use only |  |  |
| [INTERNAL\_DATA\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-data-length) | Internal use only |  |  |
| [INTERNAL\_DD\_CHAR\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-dd-char-length) | Internal use only |  |  |
| [INTERNAL\_GET\_COMMENT\_OR\_ERROR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-get-comment-or-error) | Internal use only |  |  |
| [INTERNAL\_GET\_ENABLED\_ROLE\_JSON()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-get-enabled-role-json) | Internal use only | 8.0.19 |  |
| [INTERNAL\_GET\_HOSTNAME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-get-hostname) | Internal use only | 8.0.19 |  |
| [INTERNAL\_GET\_USERNAME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-get-username) | Internal use only | 8.0.19 |  |
| [INTERNAL\_GET\_VIEW\_WARNING\_OR\_ERROR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-get-view-warning-or-error) | Internal use only |  |  |
| [INTERNAL\_INDEX\_COLUMN\_CARDINALITY()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-index-column-cardinality) | Internal use only |  |  |
| [INTERNAL\_INDEX\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-index-length) | Internal use only |  |  |
| [INTERNAL\_IS\_ENABLED\_ROLE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-is-enabled-role) | Internal use only | 8.0.19 |  |
| [INTERNAL\_IS\_MANDATORY\_ROLE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-is-mandatory-role) | Internal use only | 8.0.19 |  |
| [INTERNAL\_KEYS\_DISABLED()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-keys-disabled) | Internal use only |  |  |
| [INTERNAL\_MAX\_DATA\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-max-data-length) | Internal use only |  |  |
| [INTERNAL\_TABLE\_ROWS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-table-rows) | Internal use only |  |  |
| [INTERNAL\_UPDATE\_TIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-update-time) | Internal use only |  |  |
| [INTERVAL()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_interval) | Return the index of the argument that is less than the first argument |  |  |
| [IS](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is) | Test a value against a boolean |  |  |
| [IS\_FREE\_LOCK()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-free-lock) | Whether the named lock is free |  |  |
| [IS\_IPV4()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv4) | Whether argument is an IPv4 address |  |  |
| [IS\_IPV4\_COMPAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv4-compat) | Whether argument is an IPv4-compatible address |  |  |
| [IS\_IPV4\_MAPPED()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv4-mapped) | Whether argument is an IPv4-mapped address |  |  |
| [IS\_IPV6()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv6) | Whether argument is an IPv6 address |  |  |
| [IS NOT](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-not) | Test a value against a boolean |  |  |
| [IS NOT NULL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-not-null) | NOT NULL value test |  |  |
| [IS NULL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-null) | NULL value test |  |  |
| [IS\_USED\_LOCK()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-used-lock) | Whether the named lock is in use; return connection identifier if true |  |  |
| [IS\_UUID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-uuid) | Whether argument is a valid UUID |  |  |
| [ISNULL()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_isnull) | Test whether the argument is NULL |  |  |
| [JSON\_ARRAY()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-array) | Create JSON array |  |  |
| [JSON\_ARRAY\_APPEND()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-array-append) | Append data to JSON document |  |  |
| [JSON\_ARRAY\_INSERT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-array-insert) | Insert into JSON array |  |  |
| [JSON\_ARRAYAGG()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-arrayagg) | Return result set as a single JSON array |  |  |
| [JSON\_CONTAINS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-contains) | Whether JSON document contains specific object at path |  |  |
| [JSON\_CONTAINS\_PATH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-contains-path) | Whether JSON document contains any data at path |  |  |
| [JSON\_DEPTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-depth) | Maximum depth of JSON document |  |  |
| [JSON\_EXTRACT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-extract) | Return data from JSON document |  |  |
| [JSON\_INSERT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-insert) | Insert data into JSON document |  |  |
| [JSON\_KEYS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-keys) | Array of keys from JSON document |  |  |
| [JSON\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-length) | Number of elements in JSON document |  |  |
| [JSON\_MERGE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-merge) | Merge JSON documents, preserving duplicate keys. Deprecated synonym for JSON\_MERGE\_PRESERVE() |  | Yes |
| [JSON\_MERGE\_PATCH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-merge-patch) | Merge JSON documents, replacing values of duplicate keys |  |  |
| [JSON\_MERGE\_PRESERVE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-merge-preserve) | Merge JSON documents, preserving duplicate keys |  |  |
| [JSON\_OBJECT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-object) | Create JSON object |  |  |
| [JSON\_OBJECTAGG()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-objectagg) | Return result set as a single JSON object |  |  |
| [JSON\_OVERLAPS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-overlaps) | Compares two JSON documents, returns TRUE (1) if these have any key-value pairs or array elements in common, otherwise FALSE (0) | 8.0.17 |  |
| [JSON\_PRETTY()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-pretty) | Print a JSON document in human-readable format |  |  |
| [JSON\_QUOTE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-quote) | Quote JSON document |  |  |
| [JSON\_REMOVE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-remove) | Remove data from JSON document |  |  |
| [JSON\_REPLACE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-replace) | Replace values in JSON document |  |  |
| [JSON\_SCHEMA\_VALID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-schema-valid) | Validate JSON document against JSON schema; returns TRUE/1 if document validates against schema, or FALSE/0 if it does not | 8.0.17 |  |
| [JSON\_SCHEMA\_VALIDATION\_REPORT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-schema-validation-report) | Validate JSON document against JSON schema; returns report in JSON format on outcome on validation including success or failure and reasons for failure | 8.0.17 |  |
| [JSON\_SEARCH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-search) | Path to value within JSON document |  |  |
| [JSON\_SET()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set) | Insert data into JSON document |  |  |
| [JSON\_STORAGE\_FREE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-storage-free) | Freed space within binary representation of JSON column value following partial update |  |  |
| [JSON\_STORAGE\_SIZE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-storage-size) | Space used for storage of binary representation of a JSON document |  |  |
| [JSON\_TABLE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-table) | Return data from a JSON expression as a relational table |  |  |
| [JSON\_TYPE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-type) | Type of JSON value |  |  |
| [JSON\_UNQUOTE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-unquote) | Unquote JSON value |  |  |
| [JSON\_VALID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-valid) | Whether JSON value is valid |  |  |
| [JSON\_VALUE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-value) | Extract value from JSON document at location pointed to by path provided; return this value as VARCHAR(512) or specified type | 8.0.21 |  |
| [LAG()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lag) | Value of argument from row lagging current row within partition |  |  |
| [LAST\_DAY](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-day) | Return the last day of the month for the argument |  |  |
| [LAST\_INSERT\_ID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) | Value of the AUTOINCREMENT column for the last INSERT |  |  |
| [LAST\_VALUE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-value) | Value of argument from last row of window frame |  |  |
| [LCASE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lcase) | Synonym for LOWER() |  |  |
| [LEAD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lead) | Value of argument from row leading current row within partition |  |  |
| [LEAST()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_least) | Return the smallest argument |  |  |
| [LEFT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_left) | Return the leftmost number of characters as specified |  |  |
| [LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_length) | Return the length of a string in bytes |  |  |
| [LIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) | Simple pattern matching |  |  |
| [LineString()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_linestring) | Construct LineString from Point values |  |  |
| [LN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ln) | Return the natural logarithm of the argument |  |  |
| [LOAD\_FILE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_load-file) | Load the named file |  |  |
| [LOCALTIME(), LOCALTIME](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_localtime) | Synonym for NOW() |  |  |
| [LOCALTIMESTAMP, LOCALTIMESTAMP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_localtimestamp) | Synonym for NOW() |  |  |
| [LOCATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_locate) | Return the position of the first occurrence of substring |  |  |
| [LOG()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log) | Return the natural logarithm of the first argument |  |  |
| [LOG10()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log10) | Return the base-10 logarithm of the argument |  |  |
| [LOG2()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log2) | Return the base-2 logarithm of the argument |  |  |
| [LOWER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lower) | Return the argument in lowercase |  |  |
| [LPAD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lpad) | Return the string argument, left-padded with the specified string |  |  |
| [LTRIM()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ltrim) | Remove leading spaces |  |  |
| [MAKE\_SET()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_make-set) | Return a set of comma-separated strings that have the corresponding bit in bits set |  |  |
| [MAKEDATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_makedate) | Create a date from the year and day of year |  |  |
| [MAKETIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_maketime) | Create time from hour, minute, second |  |  |
| [MASTER\_POS\_WAIT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_master-pos-wait) | Block until the replica has read and applied all updates up to the specified position |  |  |
| [MATCH](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) | Perform full-text search |  |  |
| [MAX()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_max) | Return the maximum value |  |  |
| [MBRContains()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcontains) | Whether MBR of one geometry contains MBR of another |  |  |
| [MBRCoveredBy()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcoveredby) | Whether one MBR is covered by another |  |  |
| [MBRCovers()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcovers) | Whether one MBR covers another |  |  |
| [MBRDisjoint()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrdisjoint) | Whether MBRs of two geometries are disjoint |  |  |
| [MBREquals()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrequals) | Whether MBRs of two geometries are equal |  |  |
| [MBRIntersects()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrintersects) | Whether MBRs of two geometries intersect |  |  |
| [MBROverlaps()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbroverlaps) | Whether MBRs of two geometries overlap |  |  |
| [MBRTouches()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrtouches) | Whether MBRs of two geometries touch |  |  |
| [MBRWithin()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrwithin) | Whether MBR of one geometry is within MBR of another |  |  |
| [MD5()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_md5) | Calculate MD5 checksum |  |  |
| [MEMBER OF()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_member-of) | Returns true (1) if first operand matches any element of JSON array passed as second operand, otherwise returns false (0) | 8.0.17 |  |
| [MICROSECOND()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_microsecond) | Return the microseconds from argument |  |  |
| [MID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mid) | Return a substring starting from the specified position |  |  |
| [MIN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_min) | Return the minimum value |  |  |
| [MINUTE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_minute) | Return the minute from the argument |  |  |
| [MOD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mod) | Return the remainder |  |  |
| [MONTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_month) | Return the month from the date passed |  |  |
| [MONTHNAME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_monthname) | Return the name of the month |  |  |
| [MultiLineString()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_multilinestring) | Contruct MultiLineString from LineString values |  |  |
| [MultiPoint()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_multipoint) | Construct MultiPoint from Point values |  |  |
| [MultiPolygon()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_multipolygon) | Construct MultiPolygon from Polygon values |  |  |
| [NAME\_CONST()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_name-const) | Cause the column to have the given name |  |  |
| [NOT, !](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not) | Negates value |  |  |
| [NOT BETWEEN ... AND ...](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-between) | Whether a value is not within a range of values |  |  |
| [NOT IN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-in) | Whether a value is not within a set of values |  |  |
| [NOT LIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-like) | Negation of simple pattern matching |  |  |
| [NOT REGEXP](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-regexp) | Negation of REGEXP |  |  |
| [NOW()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) | Return the current date and time |  |  |
| [NTH\_VALUE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_nth-value) | Value of argument from N-th row of window frame |  |  |
| [NTILE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ntile) | Bucket number of current row within its partition. |  |  |
| [NULLIF()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_nullif) | Return NULL if expr1 = expr2 |  |  |
| [OCT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_oct) | Return a string containing octal representation of a number |  |  |
| [OCTET\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_octet-length) | Synonym for LENGTH() |  |  |
| [OR, ||](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or) | Logical OR |  |  |
| [ORD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ord) | Return character code for leftmost character of the argument |  |  |
| [PERCENT\_RANK()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_percent-rank) | Percentage rank value |  |  |
| [PERIOD\_ADD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_period-add) | Add a period to a year-month |  |  |
| [PERIOD\_DIFF()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_period-diff) | Return the number of months between periods |  |  |
| [PI()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_pi) | Return the value of pi |  |  |
| [Point()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_point) | Construct Point from coordinates |  |  |
| [Polygon()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_polygon) | Construct Polygon from LineString arguments |  |  |
| [POSITION()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_position) | Synonym for LOCATE() |  |  |
| [POW()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_pow) | Return the argument raised to the specified power |  |  |
| [POWER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_power) | Return the argument raised to the specified power |  |  |
| [PS\_CURRENT\_THREAD\_ID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-current-thread-id) | Performance Schema thread ID for current thread | 8.0.16 |  |
| [PS\_THREAD\_ID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-thread-id) | Performance Schema thread ID for given thread | 8.0.16 |  |
| [QUARTER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_quarter) | Return the quarter from a date argument |  |  |
| [QUOTE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_quote) | Escape the argument for use in an SQL statement |  |  |
| [RADIANS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_radians) | Return argument converted to radians |  |  |
| [RAND()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) | Return a random floating-point value |  |  |
| [RANDOM\_BYTES()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_random-bytes) | Return a random byte vector |  |  |
| [RANK()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rank) | Rank of current row within its partition, with gaps |  |  |
| [REGEXP](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) | Whether string matches regular expression |  |  |
| [REGEXP\_INSTR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-instr) | Starting index of substring matching regular expression |  |  |
| [REGEXP\_LIKE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like) | Whether string matches regular expression |  |  |
| [REGEXP\_REPLACE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-replace) | Replace substrings matching regular expression |  |  |
| [REGEXP\_SUBSTR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-substr) | Return substring matching regular expression |  |  |
| [RELEASE\_ALL\_LOCKS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-all-locks) | Release all current named locks |  |  |
| [RELEASE\_LOCK()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-lock) | Release the named lock |  |  |
| [REPEAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_repeat) | Repeat a string the specified number of times |  |  |
| [REPLACE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_replace) | Replace occurrences of a specified string |  |  |
| [REVERSE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_reverse) | Reverse the characters in a string |  |  |
| [RIGHT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_right) | Return the specified rightmost number of characters |  |  |
| [RLIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) | Whether string matches regular expression |  |  |
| [ROLES\_GRAPHML()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_roles-graphml) | Return a GraphML document representing memory role subgraphs |  |  |
| [ROUND()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) | Round the argument |  |  |
| [ROW\_COUNT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count) | The number of rows updated |  |  |
| [ROW\_NUMBER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-number) | Number of current row within its partition |  |  |
| [RPAD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rpad) | Append string the specified number of times |  |  |
| [RTRIM()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rtrim) | Remove trailing spaces |  |  |
| [SCHEMA()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_schema) | Synonym for DATABASE() |  |  |
| [SEC\_TO\_TIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sec-to-time) | Converts seconds to 'hh:mm:ss' format |  |  |
| [SECOND()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_second) | Return the second (0-59) |  |  |
| [SESSION\_USER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_session-user) | Synonym for USER() |  |  |
| [SHA1(), SHA()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha1) | Calculate an SHA-1 160-bit checksum |  |  |
| [SHA2()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha2) | Calculate an SHA-2 checksum |  |  |
| [SIGN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sign) | Return the sign of the argument |  |  |
| [SIN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sin) | Return the sine of the argument |  |  |
| [SLEEP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sleep) | Sleep for a number of seconds |  |  |
| [SOUNDEX()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_soundex) | Return a soundex string |  |  |
| [SOUNDS LIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_sounds-like) | Compare sounds |  |  |
| [SPACE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_space) | Return a string of the specified number of spaces |  |  |
| [SQRT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sqrt) | Return the square root of the argument |  |  |
| [ST\_Area()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-area) | Return Polygon or MultiPolygon area |  |  |
| [ST\_AsBinary(), ST\_AsWKB()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-asbinary) | Convert from internal geometry format to WKB |  |  |
| [ST\_AsGeoJSON()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-asgeojson) | Generate GeoJSON object from geometry |  |  |
| [ST\_AsText(), ST\_AsWKT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-astext) | Convert from internal geometry format to WKT |  |  |
| [ST\_Buffer()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer) | Return geometry of points within given distance from geometry |  |  |
| [ST\_Buffer\_Strategy()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer-strategy) | Produce strategy option for ST\_Buffer() |  |  |
| [ST\_Centroid()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-centroid) | Return centroid as a point |  |  |
| [ST\_Collect()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-collect) | Aggregate spatial values into collection | 8.0.24 |  |
| [ST\_Contains()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-contains) | Whether one geometry contains another |  |  |
| [ST\_ConvexHull()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-convexhull) | Return convex hull of geometry |  |  |
| [ST\_Crosses()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-crosses) | Whether one geometry crosses another |  |  |
| [ST\_Difference()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-difference) | Return point set difference of two geometries |  |  |
| [ST\_Dimension()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-dimension) | Dimension of geometry |  |  |
| [ST\_Disjoint()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-disjoint) | Whether one geometry is disjoint from another |  |  |
| [ST\_Distance()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance) | The distance of one geometry from another |  |  |
| [ST\_Distance\_Sphere()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance-sphere) | Minimum distance on earth between two geometries |  |  |
| [ST\_EndPoint()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-endpoint) | End Point of LineString |  |  |
| [ST\_Envelope()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-envelope) | Return MBR of geometry |  |  |
| [ST\_Equals()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-equals) | Whether one geometry is equal to another |  |  |
| [ST\_ExteriorRing()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-exteriorring) | Return exterior ring of Polygon |  |  |
| [ST\_FrechetDistance()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-frechetdistance) | The discrete Fréchet distance of one geometry from another | 8.0.23 |  |
| [ST\_GeoHash()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geohash) | Produce a geohash value |  |  |
| [ST\_GeomCollFromText(), ST\_GeometryCollectionFromText(), ST\_GeomCollFromTxt()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomcollfromtext) | Return geometry collection from WKT |  |  |
| [ST\_GeomCollFromWKB(), ST\_GeometryCollectionFromWKB()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomcollfromwkb) | Return geometry collection from WKB |  |  |
| [ST\_GeometryN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geometryn) | Return N-th geometry from geometry collection |  |  |
| [ST\_GeometryType()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geometrytype) | Return name of geometry type |  |  |
| [ST\_GeomFromGeoJSON()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromgeojson) | Generate geometry from GeoJSON object |  |  |
| [ST\_GeomFromText(), ST\_GeometryFromText()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromtext) | Return geometry from WKT |  |  |
| [ST\_GeomFromWKB(), ST\_GeometryFromWKB()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromwkb) | Return geometry from WKB |  |  |
| [ST\_HausdorffDistance()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-hausdorffdistance) | The discrete Hausdorff distance of one geometry from another | 8.0.23 |  |
| [ST\_InteriorRingN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-interiorringn) | Return N-th interior ring of Polygon |  |  |
| [ST\_Intersection()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-intersection) | Return point set intersection of two geometries |  |  |
| [ST\_Intersects()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-intersects) | Whether one geometry intersects another |  |  |
| [ST\_IsClosed()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isclosed) | Whether a geometry is closed and simple |  |  |
| [ST\_IsEmpty()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isempty) | Whether a geometry is empty |  |  |
| [ST\_IsSimple()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-issimple) | Whether a geometry is simple |  |  |
| [ST\_IsValid()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isvalid) | Whether a geometry is valid |  |  |
| [ST\_LatFromGeoHash()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latfromgeohash) | Return latitude from geohash value |  |  |
| [ST\_Latitude()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latitude) | Return latitude of Point | 8.0.12 |  |
| [ST\_Length()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-length) | Return length of LineString |  |  |
| [ST\_LineFromText(), ST\_LineStringFromText()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-linefromtext) | Construct LineString from WKT |  |  |
| [ST\_LineFromWKB(), ST\_LineStringFromWKB()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-linefromwkb) | Construct LineString from WKB |  |  |
| [ST\_LineInterpolatePoint()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-lineinterpolatepoint) | The point a given percentage along a LineString | 8.0.24 |  |
| [ST\_LineInterpolatePoints()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-lineinterpolatepoints) | The points a given percentage along a LineString | 8.0.24 |  |
| [ST\_LongFromGeoHash()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-longfromgeohash) | Return longitude from geohash value |  |  |
| [ST\_Longitude()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-longitude) | Return longitude of Point | 8.0.12 |  |
| [ST\_MakeEnvelope()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-makeenvelope) | Rectangle around two points |  |  |
| [ST\_MLineFromText(), ST\_MultiLineStringFromText()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mlinefromtext) | Construct MultiLineString from WKT |  |  |
| [ST\_MLineFromWKB(), ST\_MultiLineStringFromWKB()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mlinefromwkb) | Construct MultiLineString from WKB |  |  |
| [ST\_MPointFromText(), ST\_MultiPointFromText()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpointfromtext) | Construct MultiPoint from WKT |  |  |
| [ST\_MPointFromWKB(), ST\_MultiPointFromWKB()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpointfromwkb) | Construct MultiPoint from WKB |  |  |
| [ST\_MPolyFromText(), ST\_MultiPolygonFromText()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpolyfromtext) | Construct MultiPolygon from WKT |  |  |
| [ST\_MPolyFromWKB(), ST\_MultiPolygonFromWKB()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpolyfromwkb) | Construct MultiPolygon from WKB |  |  |
| [ST\_NumGeometries()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-numgeometries) | Return number of geometries in geometry collection |  |  |
| [ST\_NumInteriorRing(), ST\_NumInteriorRings()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-numinteriorrings) | Return number of interior rings in Polygon |  |  |
| [ST\_NumPoints()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-numpoints) | Return number of points in LineString |  |  |
| [ST\_Overlaps()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-overlaps) | Whether one geometry overlaps another |  |  |
| [ST\_PointAtDistance()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointatdistance) | The point a given distance along a LineString | 8.0.24 |  |
| [ST\_PointFromGeoHash()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointfromgeohash) | Convert geohash value to POINT value |  |  |
| [ST\_PointFromText()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointfromtext) | Construct Point from WKT |  |  |
| [ST\_PointFromWKB()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointfromwkb) | Construct Point from WKB |  |  |
| [ST\_PointN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointn) | Return N-th point from LineString |  |  |
| [ST\_PolyFromText(), ST\_PolygonFromText()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-polyfromtext) | Construct Polygon from WKT |  |  |
| [ST\_PolyFromWKB(), ST\_PolygonFromWKB()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-polyfromwkb) | Construct Polygon from WKB |  |  |
| [ST\_Simplify()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-simplify) | Return simplified geometry |  |  |
| [ST\_SRID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) | Return spatial reference system ID for geometry |  |  |
| [ST\_StartPoint()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-startpoint) | Start Point of LineString |  |  |
| [ST\_SwapXY()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-swapxy) | Return argument with X/Y coordinates swapped |  |  |
| [ST\_SymDifference()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-symdifference) | Return point set symmetric difference of two geometries |  |  |
| [ST\_Touches()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-touches) | Whether one geometry touches another |  |  |
| [ST\_Transform()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-transform) | Transform coordinates of geometry | 8.0.13 |  |
| [ST\_Union()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-union) | Return point set union of two geometries |  |  |
| [ST\_Validate()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-validate) | Return validated geometry |  |  |
| [ST\_Within()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-within) | Whether one geometry is within another |  |  |
| [ST\_X()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-x) | Return X coordinate of Point |  |  |
| [ST\_Y()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-y) | Return Y coordinate of Point |  |  |
| [STATEMENT\_DIGEST()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_statement-digest) | Compute statement digest hash value |  |  |
| [STATEMENT\_DIGEST\_TEXT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_statement-digest-text) | Compute normalized statement digest |  |  |
| [STD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_std) | Return the population standard deviation |  |  |
| [STDDEV()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev) | Return the population standard deviation |  |  |
| [STDDEV\_POP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev-pop) | Return the population standard deviation |  |  |
| [STDDEV\_SAMP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev-samp) | Return the sample standard deviation |  |  |
| [STR\_TO\_DATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_str-to-date) | Convert a string to a date |  |  |
| [STRCMP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_strcmp) | Compare two strings |  |  |
| [SUBDATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_subdate) | Synonym for DATE\_SUB() when invoked with three arguments |  |  |
| [SUBSTR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substr) | Return the substring as specified |  |  |
| [SUBSTRING()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring) | Return the substring as specified |  |  |
| [SUBSTRING\_INDEX()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring-index) | Return a substring from a string before the specified number of occurrences of the delimiter |  |  |
| [SUBTIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_subtime) | Subtract times |  |  |
| [SUM()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sum) | Return the sum |  |  |
| [SYSDATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate) | Return the time at which the function executes |  |  |
| [SYSTEM\_USER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_system-user) | Synonym for USER() |  |  |
| [TAN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_tan) | Return the tangent of the argument |  |  |
| [TIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_time) | Extract the time portion of the expression passed |  |  |
| [TIME\_FORMAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_time-format) | Format as time |  |  |
| [TIME\_TO\_SEC()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_time-to-sec) | Return the argument converted to seconds |  |  |
| [TIMEDIFF()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timediff) | Subtract time |  |  |
| [TIMESTAMP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timestamp) | With a single argument, this function returns the date or datetime expression; with two arguments, the sum of the arguments |  |  |
| [TIMESTAMPADD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timestampadd) | Add an interval to a datetime expression |  |  |
| [TIMESTAMPDIFF()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timestampdiff) | Subtract an interval from a datetime expression |  |  |
| [TO\_BASE64()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-base64) | Return the argument converted to a base-64 string |  |  |
| [TO\_DAYS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-days) | Return the date argument converted to days |  |  |
| [TO\_SECONDS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-seconds) | Return the date or datetime argument converted to seconds since Year 0 |  |  |
| [TRIM()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_trim) | Remove leading and trailing spaces |  |  |
| [TRUNCATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_truncate) | Truncate to specified number of decimal places |  |  |
| [UCASE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ucase) | Synonym for UPPER() |  |  |
| [UNCOMPRESS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uncompress) | Uncompress a string compressed |  |  |
| [UNCOMPRESSED\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uncompressed-length) | Return the length of a string before compression |  |  |
| [UNHEX()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unhex) | Return a string containing hex representation of a number |  |  |
| [UNIX\_TIMESTAMP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) | Return a Unix timestamp |  |  |
| [UpdateXML()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_updatexml) | Return replaced XML fragment |  |  |
| [UPPER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper) | Convert to uppercase |  |  |
| [USER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user) | The user name and host name provided by the client |  |  |
| [UTC\_DATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-date) | Return the current UTC date |  |  |
| [UTC\_TIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-time) | Return the current UTC time |  |  |
| [UTC\_TIMESTAMP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-timestamp) | Return the current UTC date and time |  |  |
| [UUID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) | Return a Universal Unique Identifier (UUID) |  |  |
| [UUID\_SHORT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-short) | Return an integer-valued universal identifier |  |  |
| [UUID\_TO\_BIN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin) | Convert string UUID to binary |  |  |
| [VALIDATE\_PASSWORD\_STRENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_validate-password-strength) | Determine strength of password |  |  |
| [VALUES()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_values) | Define the values to be used during an INSERT |  |  |
| [VAR\_POP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_var-pop) | Return the population standard variance |  |  |
| [VAR\_SAMP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_var-samp) | Return the sample variance |  |  |
| [VARIANCE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_variance) | Return the population standard variance |  |  |
| [VERSION()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_version) | Return a string that indicates the MySQL server version |  |  |
| [WAIT\_FOR\_EXECUTED\_GTID\_SET()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_wait-for-executed-gtid-set) | Wait until the given GTIDs have executed on the replica. |  |  |
| [WAIT\_UNTIL\_SQL\_THREAD\_AFTER\_GTIDS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_wait-until-sql-thread-after-gtids) | Use **WAIT\_FOR\_EXECUTED\_GTID\_SET()**. |  | 8.0.18 |
| [WEEK()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) | Return the week number |  |  |
| [WEEKDAY()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weekday) | Return the weekday index |  |  |
| [WEEKOFYEAR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weekofyear) | Return the calendar week of the date (1-53) |  |  |
| [WEIGHT\_STRING()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string) | Return the weight string for a string |  |  |
| [XOR](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_xor) | Logical XOR |  |  |
| [YEAR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_year) | Return the year |  |  |
| [YEARWEEK()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_yearweek) | Return the year and week |  |  |
| [|](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-or) | Bitwise OR |  |  |
| [~](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-invert) | Bitwise inversion |  |  |

## 12.2 User-Defined Function Reference

The following table lists each user-defined function and provides a short description of each one. For a table listing SQL functions and operators, see [Section 12.1, “SQL Function and Operator Reference”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#sql-function-reference)

For general information about user-defined functions, see [Section 5.7, “MySQL Server User-Defined Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-udfs).

**Table 12.2 User-Defined Functions**

| **Name** | **Description** | **Introduced** | **Deprecated** |
| --- | --- | --- | --- |
| [**asymmetric\_decrypt()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_asymmetric-decrypt) | Decrypt ciphertext using private or public key |  |  |
| [**asymmetric\_derive()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_asymmetric-derive) | Derive symmetric key from asymmetric keys |  |  |
| [**asymmetric\_encrypt()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_asymmetric-encrypt) | Encrypt cleartext using private or public key |  |  |
| [**asymmetric\_sign()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_asymmetric-sign) | Generate signature from digest |  |  |
| [**asymmetric\_verify()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_asymmetric-verify) | Verify that signature matches digest |  |  |
| [**asynchronous\_connection\_failover\_add\_managed()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-add-managed) | Add a replication source server in a managed group to the source list | 8.0.23 |  |
| [**asynchronous\_connection\_failover\_add\_source()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-add-source) | Add a replication source server to the source list | 8.0.22 |  |
| [**asynchronous\_connection\_failover\_delete\_managed()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-delete-managed) | Remove managed group of replication source servers from the source list | 8.0.23 |  |
| [**asynchronous\_connection\_failover\_delete\_source()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-delete-source) | Remove a replication source server from the source list | 8.0.22 |  |
| [**audit\_api\_message\_emit\_udf()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_audit-api-message-emit-udf) | Add message event to audit log |  |  |
| [**audit\_log\_encryption\_password\_get()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_audit-log-encryption-password-get) | Fetch audit log encryption password |  |  |
| [**audit\_log\_encryption\_password\_set()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_audit-log-encryption-password-set) | Set audit log encryption password |  |  |
| [**audit\_log\_filter\_flush()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_audit-log-filter-flush) | Flush audit log filter tables |  |  |
| [**audit\_log\_filter\_remove\_filter()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_audit-log-filter-remove-filter) | Remove audit log filter |  |  |
| [**audit\_log\_filter\_remove\_user()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_audit-log-filter-remove-user) | Unassign audit log filter from user |  |  |
| [**audit\_log\_filter\_set\_filter()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_audit-log-filter-set-filter) | Define audit log filter |  |  |
| [**audit\_log\_filter\_set\_user()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_audit-log-filter-set-user) | Assign audit log filter to user |  |  |
| [**audit\_log\_read()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_audit-log-read) | Return audit log records |  |  |
| [**audit\_log\_read\_bookmark()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_audit-log-read-bookmark) | Bookmark for most recent audit log event |  |  |
| [**create\_asymmetric\_priv\_key()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_create-asymmetric-priv-key) | Create private key |  |  |
| [**create\_asymmetric\_pub\_key()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_create-asymmetric-pub-key) | Create public key |  |  |
| [**create\_dh\_parameters()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_create-dh-parameters) | Generate shared DH secret |  |  |
| [**create\_digest()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_create-digest) | Generate digest from string |  |  |
| [**firewall\_group\_delist()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_firewall-group-delist) | Remove account from firewall group profile | 8.0.23 |  |
| [**firewall\_group\_enlist()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_firewall-group-enlist) | Add account to firewall group profile | 8.0.23 |  |
| [**gen\_blacklist()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_gen-blacklist) | Perform dictionary term replacement |  | 8.0.23 |
| [**gen\_blocklist()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_gen-blocklist) | Perform dictionary term replacement | 8.0.23 |  |
| [**gen\_dictionary()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_gen-dictionary) | Return random term from dictionary |  |  |
| [**gen\_dictionary\_drop()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_gen-dictionary-drop) | Remove dictionary from registry |  |  |
| [**gen\_dictionary\_load()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_gen-dictionary-load) | Load dictionary into registry |  |  |
| [**gen\_range()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_gen-range) | Generate random number within range |  |  |
| [**gen\_rnd\_email()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_gen-rnd-email) | Generate random email address |  |  |
| [**gen\_rnd\_pan()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_gen-rnd-pan) | Generate random payment card Primary Account Number |  |  |
| [**gen\_rnd\_ssn()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_gen-rnd-ssn) | Generate random US Social Security number |  |  |
| [**gen\_rnd\_us\_phone()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_gen-rnd-us-phone) | Generate random US phone number |  |  |
| [**group\_replication\_get\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-get-communication-protocol) | Return Group Replication protocol version |  |  |
| [**group\_replication\_get\_write\_concurrency()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-get-write-concurrency) | Return maximum number of consensus instances executable in parallel |  |  |
| [**group\_replication\_set\_as\_primary()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-as-primary) | Assign group member as new primary |  |  |
| [**group\_replication\_set\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-communication-protocol) | Set Group Replication protocol version |  |  |
| [**group\_replication\_set\_write\_concurrency()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-write-concurrency) | Set maximum number of consensus instances executable in parallel |  |  |
| [**group\_replication\_switch\_to\_multi\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-multi-primary-mode) | Change group from single-primary to multi-primary mode |  |  |
| [**group\_replication\_switch\_to\_single\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-single-primary-mode) | Change group from multi-primary to single-primary mode |  |  |
| [**keyring\_aws\_rotate\_cmk()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_keyring-aws-rotate-cmk) | Rotate AWS customer master key |  |  |
| [**keyring\_aws\_rotate\_keys()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_keyring-aws-rotate-keys) | Rotate keys in keyring\_aws storage file |  |  |
| [**keyring\_hashicorp\_update\_config()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_keyring-hashicorp-update-config) | Cause runtime keyring\_hashicorp reconfiguration |  |  |
| [**keyring\_key\_fetch()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_keyring-key-fetch) | Fetch keyring key value |  |  |
| [**keyring\_key\_generate()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_keyring-key-generate) | Generate random keyring key |  |  |
| [**keyring\_key\_length\_fetch()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_keyring-key-length-fetch) | Return keyring key length |  |  |
| [**keyring\_key\_remove()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_keyring-key-remove) | Remove keyring key |  |  |
| [**keyring\_key\_store()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_keyring-key-store) | Store key in keyring |  |  |
| [**keyring\_key\_type\_fetch()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_keyring-key-type-fetch) | Return keyring key type |  |  |
| [**load\_rewrite\_rules()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_load-rewrite-rules) | Rewriter plugin helper routine |  |  |
| [**mask\_inner()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_mask-inner) | Mask interior part of string |  |  |
| [**mask\_outer()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_mask-outer) | Mask left and right parts of string |  |  |
| [**mask\_pan()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_mask-pan) | Mask payment card Primary Account Number part of string |  |  |
| [**mask\_pan\_relaxed()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_mask-pan-relaxed) | Mask payment card Primary Account Number part of string |  |  |
| [**mask\_ssn()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_mask-ssn) | Mask US Social Security number |  |  |
| [**mysql\_firewall\_flush\_status()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_mysql-firewall-flush-status) | Reset firewall status variables |  |  |
| [**mysql\_query\_attribute\_string()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_mysql-query-attribute-string) | Fetch query attribute value | 8.0.23 |  |
| [**normalize\_statement()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_normalize-statement) | Normalize SQL statement to digest form |  |  |
| [**read\_firewall\_group\_allowlist()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_read-firewall-group-allowlist) | Update firewall group profile recorded-statement cache | 8.0.23 |  |
| [**read\_firewall\_groups()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_read-firewall-groups) | Update firewall group profile cache | 8.0.23 |  |
| [**read\_firewall\_users()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_read-firewall-users) | Update firewall account profile cache |  | 8.0.25 |
| [**read\_firewall\_whitelist()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_read-firewall-whitelist) | Update firewall account profile recorded-statement cache |  | 8.0.25 |
| [**service\_get\_read\_locks()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_service-get-read-locks) | Acquire locking service shared locks |  |  |
| [**service\_get\_write\_locks()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_service-get-write-locks) | Acquire locking service exclusive locks |  |  |
| [**service\_release\_locks()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_service-release-locks) | Release locking service locks |  |  |
| [**set\_firewall\_group\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_set-firewall-group-mode) | Establish firewall group profile operational mode | 8.0.23 |  |
| [**set\_firewall\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#udf_set-firewall-mode) | Establish firewall account profile operational mode |  | 8.0.25 |
| [**version\_tokens\_delete()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_version-tokens-delete) | Delete tokens from version tokens list |  |  |
| [**version\_tokens\_edit()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_version-tokens-edit) | Modify version tokens list |  |  |
| [**version\_tokens\_lock\_exclusive()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_version-tokens-lock-exclusive) | Acquire exclusive locks on version tokens |  |  |
| [**version\_tokens\_lock\_shared()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_version-tokens-lock-shared) | Acquire shared locks on version tokens |  |  |
| [**version\_tokens\_set()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_version-tokens-set) | Set version tokens list |  |  |
| [**version\_tokens\_show()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_version-tokens-show) | Return version tokens list |  |  |
| [**version\_tokens\_unlock()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf_version-tokens-unlock) | Release version tokens locks |  |  |

## 12.3 Type Conversion in Expression Evaluation

When an operator is used with operands of different types, type conversion occurs to make the operands compatible. Some conversions occur implicitly. For example, MySQL automatically converts strings to numbers as necessary, and vice versa.

mysql> **SELECT 1+'1';**

-> 2

mysql> **SELECT CONCAT(2,' test');**

-> '2 test'

It is also possible to convert a number to a string explicitly using the [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) function. Conversion occurs implicitly with the [**CONCAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat) function because it expects string arguments.

mysql> **SELECT 38.8, CAST(38.8 AS CHAR);**

-> 38.8, '38.8'

mysql> **SELECT 38.8, CONCAT(38.8);**

-> 38.8, '38.8'

See later in this section for information about the character set of implicit number-to-string conversions, and for modified rules that apply to **CREATE TABLE ... SELECT** statements.

The following rules describe how conversion occurs for comparison operations:

If one or both arguments are **NULL**, the result of the comparison is **NULL**, except for the **NULL**-safe [**<=>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal-to) equality comparison operator. For **NULL <=> NULL**, the result is true. No conversion is needed.

If both arguments in a comparison operation are strings, they are compared as strings.

If both arguments are integers, they are compared as integers.

Hexadecimal values are treated as binary strings if not compared to a number.

If one of the arguments is a [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) or [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) column and the other argument is a constant, the constant is converted to a timestamp before the comparison is performed. This is done to be more ODBC-friendly. This is not done for the arguments to [**IN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_in). To be safe, always use complete datetime, date, or time strings when doing comparisons. For example, to achieve best results when using [**BETWEEN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_between) with date or time values, use [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) to explicitly convert the values to the desired data type.

A single-row subquery from a table or tables is not considered a constant. For example, if a subquery returns an integer to be compared to a [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value, the comparison is done as two integers. The integer is not converted to a temporal value. To compare the operands as [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) values, use [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) to explicitly convert the subquery value to [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime).

If one of the arguments is a decimal value, comparison depends on the other argument. The arguments are compared as decimal values if the other argument is a decimal or integer value, or as floating-point values if the other argument is a floating-point value.

In all other cases, the arguments are compared as floating-point (real) numbers. For example, a comparison of string and numeric operands takes place as a comparison of floating-point numbers.

For information about conversion of values from one temporal type to another, see [Section 11.2.7, “Conversion Between Date and Time Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#date-and-time-type-conversion).

Comparison of JSON values takes place at two levels. The first level of comparison is based on the JSON types of the compared values. If the types differ, the comparison result is determined solely by which type has higher precedence. If the two values have the same JSON type, a second level of comparison occurs using type-specific rules. For comparison of JSON and non-JSON values, the non-JSON value is converted to JSON and the values compared as JSON values. For details, see [Comparison and Ordering of JSON Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json-comparison).

The following examples illustrate conversion of strings to numbers for comparison operations:

mysql> **SELECT 1 > '6x';**

-> 0

mysql> **SELECT 7 > '6x';**

-> 1

mysql> **SELECT 0 > 'x6';**

-> 0

mysql> **SELECT 0 = 'x6';**

-> 1

For comparisons of a string column with a number, MySQL cannot use an index on the column to look up the value quickly. If ***str\_col*** is an indexed string column, the index cannot be used when performing the lookup in the following statement:

SELECT \* FROM ***tbl\_name*** WHERE ***str\_col***=1;

The reason for this is that there are many different strings that may convert to the value **1**, such as **'1'**, **' 1'**, or **'1a'**.

Comparisons between floating-point numbers and large values of **INTEGER** type are approximate because the integer is converted to double-precision floating point before comparison, which is not capable of representing all 64-bit integers exactly. For example, the integer value 253 + 1 is not representable as a float, and is rounded to 253 or 253 + 2 before a float comparison, depending on the platform.

To illustrate, only the first of the following comparisons compares equal values, but both comparisons return true (1):

mysql> **SELECT '9223372036854775807' = 9223372036854775807;**

-> 1

mysql> **SELECT '9223372036854775807' = 9223372036854775806;**

-> 1

When conversions from string to floating-point and from integer to floating-point occur, they do not necessarily occur the same way. The integer may be converted to floating-point by the CPU, whereas the string is converted digit by digit in an operation that involves floating-point multiplications. Also, results can be affected by factors such as computer architecture or the compiler version or optimization level. One way to avoid such problems is to use [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) so that a value is not converted implicitly to a float-point number:

mysql> **SELECT CAST('9223372036854775807' AS UNSIGNED) = 9223372036854775806;**

-> 0

For more information about floating-point comparisons, see [Section B.3.4.8, “Problems with Floating-Point Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#problems-with-float).

The server includes **dtoa**, a conversion library that provides the basis for improved conversion between string or [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) values and approximate-value ([**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types)/[**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types)) numbers:

Consistent conversion results across platforms, which eliminates, for example, Unix versus Windows conversion differences.

Accurate representation of values in cases where results previously did not provide sufficient precision, such as for values close to IEEE limits.

Conversion of numbers to string format with the best possible precision. The precision of **dtoa** is always the same or better than that of the standard C library functions.

Because the conversions produced by this library differ in some cases from non-**dtoa** results, the potential exists for incompatibilities in applications that rely on previous results. For example, applications that depend on a specific exact result from previous conversions might need adjustment to accommodate additional precision.

The **dtoa** library provides conversions with the following properties. ***D*** represents a value with a [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) or string representation, and ***F*** represents a floating-point number in native binary (IEEE) format.

***F*** -> ***D*** conversion is done with the best possible precision, returning ***D*** as the shortest string that yields ***F*** when read back in and rounded to the nearest value in native binary format as specified by IEEE.

***D*** -> ***F*** conversion is done such that ***F*** is the nearest native binary number to the input decimal string ***D***.

These properties imply that ***F*** -> ***D*** -> ***F*** conversions are lossless unless ***F*** is **-inf**, **+inf**, or **NaN**. The latter values are not supported because the SQL standard defines them as invalid values for [**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) or [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types).

For ***D*** -> ***F*** -> ***D*** conversions, a sufficient condition for losslessness is that ***D*** uses 15 or fewer digits of precision, is not a denormal value, **-inf**, **+inf**, or **NaN**. In some cases, the conversion is lossless even if ***D*** has more than 15 digits of precision, but this is not always the case.

Implicit conversion of a numeric or temporal value to string produces a value that has a character set and collation determined by the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variables. (These variables commonly are set with [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names). For information about connection character sets, see [Section 10.4, “Connection Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection).)

This means that such a conversion results in a character (nonbinary) string (a [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), or [**LONGTEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) value), except in the case that the connection character set is set to **binary**. In that case, the conversion result is a binary string (a [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), or [**LONGBLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) value).

For integer expressions, the preceding remarks about expression evaluation apply somewhat differently for expression assignment; for example, in a statement such as this:

CREATE TABLE t SELECT ***integer\_expr***;

In this case, the table in the column resulting from the expression has type [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) or [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) depending on the length of the integer expression. If the maximum length of the expression does not fit in an [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) is used instead. The length is taken from the **max\_length** value of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) result set metadata (see [C API Basic Data Structures](https://dev.mysql.com/doc/c-api/8.0/en/c-api-data-structures.html)). This means that you can force a [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) rather than [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) by use of a sufficiently long expression:

CREATE TABLE t SELECT 000000000000000000000;

## 12.4 Operators

[12.4.1 Operator Precedence](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator-precedence)

[12.4.2 Comparison Functions and Operators](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#comparison-operators)

[12.4.3 Logical Operators](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#logical-operators)

[12.4.4 Assignment Operators](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#assignment-operators)

**Table 12.3 Operators**

|  |  |  |
| --- | --- | --- |
| Name | Description | Introduced |
| [&](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-and) | Bitwise AND |  |
| [>](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_greater-than) | Greater than operator |  |
| [>>](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_right-shift) | Right shift |  |
| [>=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_greater-than-or-equal) | Greater than or equal operator |  |
| [<](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_less-than) | Less than operator |  |
| [<>, !=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-equal) | Not equal operator |  |
| [<<](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_left-shift) | Left shift |  |
| [<=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_less-than-or-equal) | Less than or equal operator |  |
| [<=>](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal-to) | NULL-safe equal to operator |  |
| [%, MOD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_mod) | Modulo operator |  |
| [\*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_times) | Multiplication operator |  |
| [+](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_plus) | Addition operator |  |
| [-](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_minus) | Minus operator |  |
| [-](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_unary-minus) | Change the sign of the argument |  |
| [->](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-column-path) | Return value from JSON column after evaluating path; equivalent to JSON\_EXTRACT(). |  |
| [->>](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-inline-path) | Return value from JSON column after evaluating path and unquoting the result; equivalent to JSON\_UNQUOTE(JSON\_EXTRACT()). |  |
| [/](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_divide) | Division operator |  |
| [:=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-value) | Assign a value |  |
| [=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-equal) | Assign a value (as part of a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement, or as part of the **SET** clause in an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement) |  |
| [=](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal) | Equal operator |  |
| [^](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-xor) | Bitwise XOR |  |
| [AND, &&](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_and) | Logical AND |  |
| [BETWEEN ... AND ...](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_between) | Whether a value is within a range of values |  |
| [BINARY](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_binary) | Cast a string to a binary string |  |
| [CASE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case) | Case operator |  |
| [DIV](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_div) | Integer division |  |
| [IN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_in) | Whether a value is within a set of values |  |
| [IS](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is) | Test a value against a boolean |  |
| [IS NOT](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-not) | Test a value against a boolean |  |
| [IS NOT NULL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-not-null) | NOT NULL value test |  |
| [IS NULL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-null) | NULL value test |  |
| [LIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) | Simple pattern matching |  |
| [MEMBER OF()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_member-of) | Returns true (1) if first operand matches any element of JSON array passed as second operand, otherwise returns false (0) | 8.0.17 |
| [NOT, !](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not) | Negates value |  |
| [NOT BETWEEN ... AND ...](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-between) | Whether a value is not within a range of values |  |
| [NOT IN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-in) | Whether a value is not within a set of values |  |
| [NOT LIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-like) | Negation of simple pattern matching |  |
| [NOT REGEXP](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-regexp) | Negation of REGEXP |  |
| [OR, ||](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or) | Logical OR |  |
| [REGEXP](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) | Whether string matches regular expression |  |
| [RLIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) | Whether string matches regular expression |  |
| [SOUNDS LIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_sounds-like) | Compare sounds |  |
| [XOR](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_xor) | Logical XOR |  |
| [|](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-or) | Bitwise OR |  |
| [~](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-invert) | Bitwise inversion |  |

### 12.4.1 Operator Precedence

Operator precedences are shown in the following list, from highest precedence to the lowest. Operators that are shown together on a line have the same precedence.

INTERVAL

BINARY, COLLATE

!

- (unary minus), ~ (unary bit inversion)

^

\*, /, DIV, %, MOD

-, +

<<, >>

&

|

= (comparison), <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN, MEMBER OF

BETWEEN, CASE, WHEN, THEN, ELSE

NOT

AND, &&

XOR

OR, ||

= (assignment), :=

The precedence of **=** depends on whether it is used as a comparison operator ([**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal)) or as an assignment operator ([**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-equal)). When used as a comparison operator, it has the same precedence as [**<=>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal-to), [**>=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_greater-than-or-equal), [**>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_greater-than), [**<=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_less-than-or-equal), [**<**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_less-than), [**<>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-equal), [**!=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-equal), [**IS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is), [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like), [**REGEXP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp), and [**IN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_in). When used as an assignment operator, it has the same precedence as [**:=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-value). [Section 13.7.6.1, “SET Syntax for Variable Assignment”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable), and [Section 9.4, “User-Defined Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#user-variables), explain how MySQL determines which interpretation of **=** should apply.

For operators that occur at the same precedence level within an expression, evaluation proceeds left to right, with the exception that assignments evaluate right to left.

The precedence and meaning of some operators depends on the SQL mode:

By default, [**||**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or) is a logical [**OR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or) operator. With [**PIPES\_AS\_CONCAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_pipes_as_concat) enabled, [**||**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or) is string concatenation, with a precedence between [**^**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-xor) and the unary operators.

By default, [**!**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not) has a higher precedence than **NOT**. With [**HIGH\_NOT\_PRECEDENCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_high_not_precedence) enabled, [**!**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not) and **NOT** have the same precedence.

See [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode).

The precedence of operators determines the order of evaluation of terms in an expression. To override this order and group terms explicitly, use parentheses. For example:

mysql> **SELECT 1+2\*3;**

-> 7

mysql> **SELECT (1+2)\*3;**

-> 9

### 12.4.2 Comparison Functions and Operators

**Table 12.4 Comparison Operators**

| **Name** | **Description** |
| --- | --- |
| [**>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_greater-than) | Greater than operator |
| [**>=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_greater-than-or-equal) | Greater than or equal operator |
| [**<**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_less-than) | Less than operator |
| [**<>**, **!=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-equal) | Not equal operator |
| [**<=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_less-than-or-equal) | Less than or equal operator |
| [**<=>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal-to) | NULL-safe equal to operator |
| [**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal) | Equal operator |
| [**BETWEEN ... AND ...**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_between) | Whether a value is within a range of values |
| [**COALESCE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_coalesce) | Return the first non-NULL argument |
| [**GREATEST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_greatest) | Return the largest argument |
| [**IN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_in) | Whether a value is within a set of values |
| [**INTERVAL()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_interval) | Return the index of the argument that is less than the first argument |
| [**IS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is) | Test a value against a boolean |
| [**IS NOT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-not) | Test a value against a boolean |
| [**IS NOT NULL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-not-null) | NOT NULL value test |
| [**IS NULL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-null) | NULL value test |
| [**ISNULL()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_isnull) | Test whether the argument is NULL |
| [**LEAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_least) | Return the smallest argument |
| [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) | Simple pattern matching |
| [**NOT BETWEEN ... AND ...**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-between) | Whether a value is not within a range of values |
| [**NOT IN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-in) | Whether a value is not within a set of values |
| [**NOT LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-like) | Negation of simple pattern matching |
| [**STRCMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_strcmp) | Compare two strings |

Comparison operations result in a value of **1** (**TRUE**), **0** (**FALSE**), or **NULL**. These operations work for both numbers and strings. Strings are automatically converted to numbers and numbers to strings as necessary.

The following relational comparison operators can be used to compare not only scalar operands, but row operands:

= > < >= <= <> !=

The descriptions for those operators later in this section detail how they work with row operands. For additional examples of row comparisons in the context of row subqueries, see [Section 13.2.11.5, “Row Subqueries”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#row-subqueries).

Some of the functions in this section return values other than **1** (**TRUE**), **0** (**FALSE**), or **NULL**. [**LEAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_least) and [**GREATEST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_greatest) are examples of such functions; [Section 12.3, “Type Conversion in Expression Evaluation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#type-conversion), describes the rules for comparison operations performed by these and similar functions for determining their return values.

**Note**

In previous versions of MySQL, when evaluating an expression containing **LEAST()** or **GREATEST()**, the server attempted to guess the context in which the function was used, and to coerce the function's arguments to the data type of the expression as a whole. For example, the arguments to **LEAST("11", "45", "2")** are evaluated and sorted as strings, so that this expression returns **"11"**. In MySQL 8.0.3 and earlier, when evaluating the expression **LEAST("11", "45", "2") + 0**, the server converted the arguments to integers (anticipating the addition of integer 0 to the result) before sorting them, thus returning 2.

Beginning with MySQL 8.0.4, the server no longer attempts to infer context in this fashion. Instead, the function is executed using the arguments as provided, performing data type conversions to one or more of the arguments if and only if they are not all of the same type. Any type coercion mandated by an expression that makes use of the return value is now performed following function execution. This means that, in MySQl 8.0.4 and later, **LEAST("11", "45", "2") + 0** evaluates to **"11" + 0** and thus to integer 11. (Bug #83895, Bug #25123839)

To convert a value to a specific type for comparison purposes, you can use the [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) function. String values can be converted to a different character set using [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert). See [Section 12.11, “Cast Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#cast-functions).

By default, string comparisons are not case-sensitive and use the current character set. The default is **utf8mb4**.

**[=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_equal)**

Equal:

mysql> **SELECT 1 = 0;**

-> 0

mysql> **SELECT '0' = 0;**

-> 1

mysql> **SELECT '0.0' = 0;**

-> 1

mysql> **SELECT '0.01' = 0;**

-> 0

mysql> **SELECT '.01' = 0.01;**

-> 1

For row comparisons, **(a, b) = (x, y)** is equivalent to:

(a = x) AND (b = y)

**[<=>](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_equal-to)**

**NULL**-safe equal. This operator performs an equality comparison like the [**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal) operator, but returns **1** rather than **NULL** if both operands are **NULL**, and **0** rather than **NULL** if one operand is **NULL**.

The [**<=>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal-to) operator is equivalent to the standard SQL **IS NOT DISTINCT FROM** operator.

mysql> **SELECT 1 <=> 1, NULL <=> NULL, 1 <=> NULL;**

-> 1, 1, 0

mysql> **SELECT 1 = 1, NULL = NULL, 1 = NULL;**

-> 1, NULL, NULL

For row comparisons, **(a, b) <=> (x, y)** is equivalent to:

(a <=> x) AND (b <=> y)

**[<>](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-equal)**, [**!=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-equal)

Not equal:

mysql> **SELECT '.01' <> '0.01';**

-> 1

mysql> **SELECT .01 <> '0.01';**

-> 0

mysql> **SELECT 'zapp' <> 'zappp';**

-> 1

For row comparisons, **(a, b) <> (x, y)** and **(a, b) != (x, y)** are equivalent to:

(a <> x) OR (b <> y)

**[<=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_less-than-or-equal)**

Less than or equal:

mysql> **SELECT 0.1 <= 2;**

-> 1

For row comparisons, **(a, b) <= (x, y)** is equivalent to:

(a < x) OR ((a = x) AND (b <= y))

**[<](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_less-than)**

Less than:

mysql> **SELECT 2 < 2;**

-> 0

For row comparisons, **(a, b) < (x, y)** is equivalent to:

(a < x) OR ((a = x) AND (b < y))

**[>=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_greater-than-or-equal)**

Greater than or equal:

mysql> **SELECT 2 >= 2;**

-> 1

For row comparisons, **(a, b) >= (x, y)** is equivalent to:

(a > x) OR ((a = x) AND (b >= y))

**[>](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_greater-than)**

Greater than:

mysql> **SELECT 2 > 2;**

-> 0

For row comparisons, **(a, b) > (x, y)** is equivalent to:

(a > x) OR ((a = x) AND (b > y))

***[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_between)*[BETWEEN](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_between)*[min](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_between)*[AND](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_between)*[max](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_between)***

If ***expr*** is greater than or equal to ***min*** and ***expr*** is less than or equal to ***max***, [**BETWEEN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_between) returns **1**, otherwise it returns **0**. This is equivalent to the expression **(*min* <= *expr* AND *expr* <= *max*)** if all the arguments are of the same type. Otherwise type conversion takes place according to the rules described in [Section 12.3, “Type Conversion in Expression Evaluation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#type-conversion), but applied to all the three arguments.

mysql> **SELECT 2 BETWEEN 1 AND 3, 2 BETWEEN 3 and 1;**

-> 1, 0

mysql> **SELECT 1 BETWEEN 2 AND 3;**

-> 0

mysql> **SELECT 'b' BETWEEN 'a' AND 'c';**

-> 1

mysql> **SELECT 2 BETWEEN 2 AND '3';**

-> 1

mysql> **SELECT 2 BETWEEN 2 AND 'x-3';**

-> 0

For best results when using [**BETWEEN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_between) with date or time values, use [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) to explicitly convert the values to the desired data type. Examples: If you compare a [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) to two [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) values, convert the [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) values to [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) values. If you use a string constant such as **'2001-1-1'** in a comparison to a [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), cast the string to a [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime).

***[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-between)*[NOT BETWEEN](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-between)*[min](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-between)*[AND](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-between)*[max](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-between)***

This is the same as **NOT (*expr* BETWEEN *min* AND *max*)**.

**[COALESCE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_coalesce)*[value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_coalesce)*[,...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_coalesce)**

Returns the first non-**NULL** value in the list, or **NULL** if there are no non-**NULL** values.

The return type of [**COALESCE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_coalesce) is the aggregated type of the argument types.

mysql> **SELECT COALESCE(NULL,1);**

-> 1

mysql> **SELECT COALESCE(NULL,NULL,NULL);**

-> NULL

**[GREATEST(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_greatest)*[value1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_greatest)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_greatest)*[value2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_greatest)*[,...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_greatest)**

With two or more arguments, returns the largest (maximum-valued) argument. The arguments are compared using the same rules as for [**LEAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_least).

mysql> **SELECT GREATEST(2,0);**

-> 2

mysql> **SELECT GREATEST(34.0,3.0,5.0,767.0);**

-> 767.0

mysql> **SELECT GREATEST('B','A','C');**

-> 'C'

[**GREATEST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_greatest) returns **NULL** if any argument is **NULL**.

***[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_in)*[IN (](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_in)*[value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_in)*[,...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_in)**

Returns **1** (true) if ***expr*** is equal to any of the values in the **IN()** list, else returns **0** (false).

Type conversion takes place according to the rules described in [Section 12.3, “Type Conversion in Expression Evaluation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#type-conversion), applied to all the arguments. If no type conversion is needed for the values in the **IN()** list, they are all non-**JSON** constants of the same type, and ***expr*** can be compared to each of them as a value of the same type (possibly after type conversion), an optimization takes place. The values the list are sorted and the search for ***expr*** is done using a binary search, which makes the **IN()** operation very quick.

mysql> **SELECT 2 IN (0,3,5,7);**

-> 0

mysql> **SELECT 'wefwf' IN ('wee','wefwf','weg');**

-> 1

**IN()** can be used to compare row constructors:

mysql> **SELECT (3,4) IN ((1,2), (3,4));**

-> 1

mysql> **SELECT (3,4) IN ((1,2), (3,5));**

-> 0

You should never mix quoted and unquoted values in an **IN()** list because the comparison rules for quoted values (such as strings) and unquoted values (such as numbers) differ. Mixing types may therefore lead to inconsistent results. For example, do not write an **IN()** expression like this:

SELECT val1 FROM tbl1 WHERE val1 IN (1,2,'a');

Instead, write it like this:

SELECT val1 FROM tbl1 WHERE val1 IN ('1','2','a');

Implicit type conversion may produce nonintuitive results:

mysql> **SELECT 'a' IN (0), 0 IN ('b');**

-> 1, 1

In both cases, the comparison values are converted to floating-point values, yielding 0.0 in each case, and a comparison result of 1 (true).

The number of values in the **IN()** list is only limited by the [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) value.

To comply with the SQL standard, **IN()** returns **NULL** not only if the expression on the left hand side is **NULL**, but also if no match is found in the list and one of the expressions in the list is **NULL**.

**IN()** syntax can also be used to write certain types of subqueries. See [Section 13.2.11.3, “Subqueries with ANY, IN, or SOME”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#any-in-some-subqueries).

***[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-in)*[NOT IN (](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-in)*[value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-in)*[,...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-in)**

This is the same as **NOT (*expr* IN (*value*,...))**.

**[INTERVAL(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_interval)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_interval)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_interval)*[N1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_interval)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_interval)*[N2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_interval)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_interval)*[N3](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_interval)*[,...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_interval)**

Returns **0** if ***N*** < ***N1***, **1** if ***N*** < ***N2*** and so on or **-1** if ***N*** is **NULL**. All arguments are treated as integers. It is required that ***N1*** < ***N2*** < ***N3*** < **...** < ***Nn*** for this function to work correctly. This is because a binary search is used (very fast).

mysql> **SELECT INTERVAL(23, 1, 15, 17, 30, 44, 200);**

-> 3

mysql> **SELECT INTERVAL(10, 1, 10, 100, 1000);**

-> 2

mysql> **SELECT INTERVAL(22, 23, 30, 44, 200);**

-> 0

**[IS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_is)*[boolean\_value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_is)***

Tests a value against a boolean value, where ***boolean\_value*** can be **TRUE**, **FALSE**, or **UNKNOWN**.

mysql> **SELECT 1 IS TRUE, 0 IS FALSE, NULL IS UNKNOWN;**

-> 1, 1, 1

**[IS NOT](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_is-not)*[boolean\_value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_is-not)***

Tests a value against a boolean value, where ***boolean\_value*** can be **TRUE**, **FALSE**, or **UNKNOWN**.

mysql> **SELECT 1 IS NOT UNKNOWN, 0 IS NOT UNKNOWN, NULL IS NOT UNKNOWN;**

-> 1, 1, 0

**[IS NULL](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_is-null)**

Tests whether a value is **NULL**.

mysql> **SELECT 1 IS NULL, 0 IS NULL, NULL IS NULL;**

-> 0, 0, 1

To work well with ODBC programs, MySQL supports the following extra features when using [**IS NULL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-null):

If [**sql\_auto\_is\_null**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_auto_is_null) variable is set to 1, then after a statement that successfully inserts an automatically generated **AUTO\_INCREMENT** value, you can find that value by issuing a statement of the following form:

SELECT \* FROM ***tbl\_name*** WHERE ***auto\_col*** IS NULL

If the statement returns a row, the value returned is the same as if you invoked the [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) function. For details, including the return value after a multiple-row insert, see [Section 12.16, “Information Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#information-functions). If no **AUTO\_INCREMENT** value was successfully inserted, the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement returns no row.

The behavior of retrieving an **AUTO\_INCREMENT** value by using an [**IS NULL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-null) comparison can be disabled by setting [**sql\_auto\_is\_null = 0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_auto_is_null). See [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

The default value of [**sql\_auto\_is\_null**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_auto_is_null) is 0.

For [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) and [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) columns that are declared as **NOT NULL**, you can find the special date **'0000-00-00'** by using a statement like this:

SELECT \* FROM ***tbl\_name*** WHERE ***date\_column*** IS NULL

This is needed to get some ODBC applications to work because ODBC does not support a **'0000-00-00'** date value.

See [Obtaining Auto-Increment Values](https://dev.mysql.com/doc/connector-odbc/en/connector-odbc-usagenotes-functionality-last-insert-id.html), and the description for the **FLAG\_AUTO\_IS\_NULL** option at [Connector/ODBC Connection Parameters](https://dev.mysql.com/doc/connector-odbc/en/connector-odbc-configuration-connection-parameters.html).

**[IS NOT NULL](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_is-null)**

Tests whether a value is not **NULL**.

mysql> **SELECT 1 IS NOT NULL, 0 IS NOT NULL, NULL IS NOT NULL;**

-> 1, 1, 0

**[ISNULL(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_isnull)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_isnull)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_isnull)**

If ***expr*** is **NULL**, [**ISNULL()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_isnull) returns **1**, otherwise it returns **0**.

mysql> **SELECT ISNULL(1+1);**

-> 0

mysql> **SELECT ISNULL(1/0);**

-> 1

[**ISNULL()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_isnull) can be used instead of [**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal) to test whether a value is **NULL**. (Comparing a value to **NULL** using [**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal) always yields **NULL**.)

The [**ISNULL()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_isnull) function shares some special behaviors with the [**IS NULL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-null) comparison operator. See the description of [**IS NULL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_is-null).

**[LEAST(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_least)*[value1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_least)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_least)*[value2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_least)*[,...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_least)**

With two or more arguments, returns the smallest (minimum-valued) argument. The arguments are compared using the following rules:

If any argument is **NULL**, the result is **NULL**. No comparison is needed.

If all arguments are integer-valued, they are compared as integers.

If at least one argument is double precision, they are compared as double-precision values. Otherwise, if at least one argument is a [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) value, they are compared as [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) values.

If the arguments comprise a mix of numbers and strings, they are compared as strings.

If any argument is a nonbinary (character) string, the arguments are compared as nonbinary strings.

In all other cases, the arguments are compared as binary strings.

The return type of [**LEAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_least) is the aggregated type of the comparison argument types.

mysql> **SELECT LEAST(2,0);**

-> 0

mysql> **SELECT LEAST(34.0,3.0,5.0,767.0);**

-> 3.0

mysql> **SELECT LEAST('B','A','C');**

-> 'A'

### 12.4.3 Logical Operators

**Table 12.5 Logical Operators**

| **Name** | **Description** |
| --- | --- |
| [**AND**, **&&**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_and) | Logical AND |
| [**NOT**, **!**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not) | Negates value |
| [**OR**, **||**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or) | Logical OR |
| [**XOR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_xor) | Logical XOR |

In SQL, all logical operators evaluate to **TRUE**, **FALSE**, or **NULL** (**UNKNOWN**). In MySQL, these are implemented as 1 (**TRUE**), 0 (**FALSE**), and **NULL**. Most of this is common to different SQL database servers, although some servers may return any nonzero value for **TRUE**.

MySQL evaluates any nonzero, non-**NULL** value to **TRUE**. For example, the following statements all assess to **TRUE**:

mysql> **SELECT 10 IS TRUE;**

-> 1

mysql> **SELECT -10 IS TRUE;**

-> 1

mysql> **SELECT 'string' IS NOT NULL;**

-> 1

**[NOT](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not)**, [**!**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not)

Logical NOT. Evaluates to **1** if the operand is **0**, to **0** if the operand is nonzero, and **NOT NULL** returns **NULL**.

mysql> **SELECT NOT 10;**

-> 0

mysql> **SELECT NOT 0;**

-> 1

mysql> **SELECT NOT NULL;**

-> NULL

mysql> **SELECT ! (1+1);**

-> 0

mysql> **SELECT ! 1+1;**

-> 1

The last example produces **1** because the expression evaluates the same way as **(!1)+1**.

The [**!**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not), operator is a nonstandard MySQL extension. As of MySQL 8.0.17, this operator is deprecated; expect it to be removed in a future version of MySQL. Applications should be adjusted to use the standard SQL [**NOT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not) operator.

**[AND](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_and)**, [**&&**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_and)

Logical AND. Evaluates to **1** if all operands are nonzero and not **NULL**, to **0** if one or more operands are **0**, otherwise **NULL** is returned.

mysql> **SELECT 1 AND 1;**

-> 1

mysql> **SELECT 1 AND 0;**

-> 0

mysql> **SELECT 1 AND NULL;**

-> NULL

mysql> **SELECT 0 AND NULL;**

-> 0

mysql> **SELECT NULL AND 0;**

-> 0

The [**&&**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_and), operator is a nonstandard MySQL extension. As of MySQL 8.0.17, this operator is deprecated; expect support for it to be removed in a future version of MySQL. Applications should be adjusted to use the standard SQL [**AND**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_and) operator.

**[OR](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_or)**, [**||**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or)

Logical OR. When both operands are non-**NULL**, the result is **1** if any operand is nonzero, and **0** otherwise. With a **NULL** operand, the result is **1** if the other operand is nonzero, and **NULL** otherwise. If both operands are **NULL**, the result is **NULL**.

mysql> **SELECT 1 OR 1;**

-> 1

mysql> **SELECT 1 OR 0;**

-> 1

mysql> **SELECT 0 OR 0;**

-> 0

mysql> **SELECT 0 OR NULL;**

-> NULL

mysql> **SELECT 1 OR NULL;**

-> 1

**Note**

If the [**PIPES\_AS\_CONCAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_pipes_as_concat) SQL mode is enabled, [**||**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or) signifies the SQL-standard string concatenation operator (like [**CONCAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat)).

The [**||**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or), operator is a nonstandard MySQL extension. As of MySQL 8.0.17, this operator is deprecated; expect support for it to be removed in a future version of MySQL. Applications should be adjusted to use the standard SQL [**OR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or) operator. Exception: Deprecation does not apply if [**PIPES\_AS\_CONCAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_pipes_as_concat) is enabled because, in that case, [**||**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or) signifies string concatentation.

**[XOR](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_xor)**

Logical XOR. Returns **NULL** if either operand is **NULL**. For non-**NULL** operands, evaluates to **1** if an odd number of operands is nonzero, otherwise **0** is returned.

mysql> **SELECT 1 XOR 1;**

-> 0

mysql> **SELECT 1 XOR 0;**

-> 1

mysql> **SELECT 1 XOR NULL;**

-> NULL

mysql> **SELECT 1 XOR 1 XOR 1;**

-> 1

**a XOR b** is mathematically equal to **(a AND (NOT b)) OR ((NOT a) and b)**.

### 12.4.4 Assignment Operators

**Table 12.6 Assignment Operators**

| **Name** | **Description** |
| --- | --- |
| [**:=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-value) | Assign a value |
| [**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-equal) | Assign a value (as part of a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement, or as part of the **SET** clause in an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement) |

**[:=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_assign-value)**

Assignment operator. Causes the user variable on the left hand side of the operator to take on the value to its right. The value on the right hand side may be a literal value, another variable storing a value, or any legal expression that yields a scalar value, including the result of a query (provided that this value is a scalar value). You can perform multiple assignments in the same [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement. You can perform multiple assignments in the same statement.

Unlike [**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-equal), the [**:=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-value) operator is never interpreted as a comparison operator. This means you can use [**:=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-value) in any valid SQL statement (not just in [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statements) to assign a value to a variable.

mysql> **SELECT @var1, @var2;**

-> NULL, NULL

mysql> **SELECT @var1 := 1, @var2;**

-> 1, NULL

mysql> **SELECT @var1, @var2;**

-> 1, NULL

mysql> **SELECT @var1, @var2 := @var1;**

-> 1, 1

mysql> **SELECT @var1, @var2;**

-> 1, 1

mysql> **SELECT @var1:=COUNT(\*) FROM t1;**

-> 4

mysql> **SELECT @var1;**

-> 4

You can make value assignments using [**:=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-value) in other statements besides [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), such as [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), as shown here:

mysql> **SELECT @var1;**

-> 4

mysql> **SELECT \* FROM t1;**

-> 1, 3, 5, 7

mysql> **UPDATE t1 SET c1 = 2 WHERE c1 = @var1:= 1;**

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> **SELECT @var1;**

-> 1

mysql> **SELECT \* FROM t1;**

-> 2, 3, 5, 7

While it is also possible both to set and to read the value of the same variable in a single SQL statement using the [**:=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-value) operator, this is not recommended. [Section 9.4, “User-Defined Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#user-variables), explains why you should avoid doing this.

**[=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_assign-equal)**

This operator is used to perform value assignments in two cases, described in the next two paragraphs.

Within a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement, **=** is treated as an assignment operator that causes the user variable on the left hand side of the operator to take on the value to its right. (In other words, when used in a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement, **=** is treated identically to [**:=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-value).) The value on the right hand side may be a literal value, another variable storing a value, or any legal expression that yields a scalar value, including the result of a query (provided that this value is a scalar value). You can perform multiple assignments in the same [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement.

In the **SET** clause of an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement, **=** also acts as an assignment operator; in this case, however, it causes the column named on the left hand side of the operator to assume the value given to the right, provided any **WHERE** conditions that are part of the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) are met. You can make multiple assignments in the same **SET** clause of an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement.

In any other context, **=** is treated as a [comparison operator](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal).

mysql> **SELECT @var1, @var2;**

-> NULL, NULL

mysql> **SELECT @var1 := 1, @var2;**

-> 1, NULL

mysql> **SELECT @var1, @var2;**

-> 1, NULL

mysql> **SELECT @var1, @var2 := @var1;**

-> 1, 1

mysql> **SELECT @var1, @var2;**

-> 1, 1

For more information, see [Section 13.7.6.1, “SET Syntax for Variable Assignment”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable), [Section 13.2.13, “UPDATE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [Section 13.2.11, “Subqueries”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#subqueries).

## 12.5 Flow Control Functions

**Table 12.7 Flow Control Operators**

|  |  |
| --- | --- |
| Name | Description |
| [CASE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case) | Case operator |
| [IF()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if) | If/else construct |
| [IFNULL()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ifnull) | Null if/else construct |
| [NULLIF()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_nullif) | Return NULL if expr1 = expr2 |

**[CASE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[WHEN](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[compare\_value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[THEN](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[result](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[[WHEN](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[compare\_value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[THEN](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[result](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[...] [ELSE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[result](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)*[] END](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_case)**

[**CASE WHEN *condition* THEN *result* [WHEN *condition* THEN *result* ...] [ELSE *result*] END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case)

The first [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case) syntax returns the ***result*** for the first ***value*=*compare\_value*** comparison that is true. The second syntax returns the result for the first condition that is true. If no comparison or condition is true, the result after **ELSE** is returned, or **NULL** if there is no **ELSE** part.

**Note**

The syntax of the [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case) operator described here differs slightly from that of the SQL [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#case) statement described in [Section 13.6.5.1, “CASE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#case), for use inside stored programs. The [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#case) statement cannot have an **ELSE NULL** clause, and it is terminated with **END CASE** instead of **END**.

The return type of a [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case) expression result is the aggregated type of all result values:

If all types are numeric, the aggregated type is also numeric:

If at least one argument is double precision, the result is double precision.

Otherwise, if at least one argument is [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types), the result is [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types).

Otherwise, the result is an integer type (with one exception):

If all integer types are all signed or all unsigned, the result is the same sign and the precision is the highest of all specified integer types (that is, [**TINYINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**SMALLINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**MEDIUMINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), or [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types)).

If there is a combination of signed and unsigned integer types, the result is signed and the precision may be higher. For example, if the types are signed [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) and unsigned [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), the result is signed [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types).

The exception is unsigned [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) combined with any signed integer type. The result is [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) with sufficient precision and scale 0.

If all types are [**BIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#bit-type), the result is [**BIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#bit-type). Otherwise, [**BIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#bit-type) arguments are treated similar to [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types).

If all types are [**YEAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#year), the result is [**YEAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#year). Otherwise, **YEAR** arguments are treated similar to [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types).

If all types are character string ([**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) or [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char)), the result is [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) with maximum length determined by the longest character length of the operands.

If all types are character or binary string, the result is [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary).

[**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) and [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) are treated similar to [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char); the result is [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char).

If all types are [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json), the result is [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json).

If all types are temporal, the result is temporal:

If all temporal types are [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time), or [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), the result is [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time), or [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), respectively.

Otherwise, for a mix of temporal types, the result is [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime).

If all types are **GEOMETRY**, the result is **GEOMETRY**.

If any type is [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), the result is [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob).

For all other type combinations, the result is [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char).

Literal **NULL** operands are ignored for type aggregation.

mysql> **SELECT CASE 1 WHEN 1 THEN 'one'**

-> **WHEN 2 THEN 'two' ELSE 'more' END;**

-> 'one'

mysql> **SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END;**

-> 'true'

mysql> **SELECT CASE BINARY 'B'**

-> **WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;**

-> NULL

**[IF(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_if)*[expr1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_if)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_if)*[expr2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_if)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_if)*[expr3](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_if)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_if)**

If ***expr1*** is **TRUE** (***expr1* <> 0** and ***expr1* <> NULL**), [**IF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if) returns ***expr2***. Otherwise, it returns ***expr3***.

**Note**

There is also an [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if) statement, which differs from the [**IF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if) function described here. See [Section 13.6.5.2, “IF Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if).

If only one of ***expr2*** or ***expr3*** is explicitly **NULL**, the result type of the [**IF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if) function is the type of the non-**NULL** expression.

The default return type of [**IF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if) (which may matter when it is stored into a temporary table) is calculated as follows:

If ***expr2*** or ***expr3*** produce a string, the result is a string.

If ***expr2*** and ***expr3*** are both strings, the result is case-sensitive if either string is case-sensitive.

If ***expr2*** or ***expr3*** produce a floating-point value, the result is a floating-point value.

If ***expr2*** or ***expr3*** produce an integer, the result is an integer.

mysql> **SELECT IF(1>2,2,3);**

-> 3

mysql> **SELECT IF(1<2,'yes','no');**

-> 'yes'

mysql> **SELECT IF(STRCMP('test','test1'),'no','yes');**

-> 'no'

**[IFNULL(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ifnull)*[expr1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ifnull)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ifnull)*[expr2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ifnull)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ifnull)**

If ***expr1*** is not **NULL**, [**IFNULL()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ifnull) returns ***expr1***; otherwise it returns ***expr2***.

mysql> **SELECT IFNULL(1,0);**

-> 1

mysql> **SELECT IFNULL(NULL,10);**

-> 10

mysql> **SELECT IFNULL(1/0,10);**

-> 10

mysql> **SELECT IFNULL(1/0,'yes');**

-> 'yes'

The default return type of [**IFNULL(*expr1*,*expr2*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ifnull) is the more “general” of the two expressions, in the order **STRING**, **REAL**, or **INTEGER**. Consider the case of a table based on expressions or where MySQL must internally store a value returned by [**IFNULL()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ifnull) in a temporary table:

mysql> **CREATE TABLE tmp SELECT IFNULL(1,'test') AS test;**

mysql> **DESCRIBE tmp;**

+-------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+--------------+------+-----+---------+-------+

| test | varbinary(4) | NO | | | |

+-------+--------------+------+-----+---------+-------+

In this example, the type of the **test** column is [**VARBINARY(4)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) (a string type).

**[NULLIF(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_nullif)*[expr1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_nullif)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_nullif)*[expr2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_nullif)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_nullif)**

Returns **NULL** if ***expr1* = *expr2*** is true, otherwise returns ***expr1***. This is the same as [**CASE WHEN *expr1* = *expr2* THEN NULL ELSE *expr1* END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case).

The return value has the same type as the first argument.

mysql> **SELECT NULLIF(1,1);**

-> NULL

mysql> **SELECT NULLIF(1,2);**

-> 1

**Note**

MySQL evaluates ***expr1*** twice if the arguments are not equal.

## 12.6 Numeric Functions and Operators

[12.6.1 Arithmetic Operators](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#arithmetic-functions)

[12.6.2 Mathematical Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#mathematical-functions)

**Table 12.8 Numeric Functions and Operators**

|  |  |
| --- | --- |
| Name | Description |
| [%, MOD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_mod) | Modulo operator |
| [\*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_times) | Multiplication operator |
| [+](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_plus) | Addition operator |
| [-](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_minus) | Minus operator |
| [-](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_unary-minus) | Change the sign of the argument |
| [/](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_divide) | Division operator |
| [ABS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_abs) | Return the absolute value |
| [ACOS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_acos) | Return the arc cosine |
| [ASIN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_asin) | Return the arc sine |
| [ATAN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_atan) | Return the arc tangent |
| [ATAN2(), ATAN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_atan2) | Return the arc tangent of the two arguments |
| [CEIL()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ceil) | Return the smallest integer value not less than the argument |
| [CEILING()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ceiling) | Return the smallest integer value not less than the argument |
| [CONV()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_conv) | Convert numbers between different number bases |
| [COS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cos) | Return the cosine |
| [COT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cot) | Return the cotangent |
| [CRC32()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_crc32) | Compute a cyclic redundancy check value |
| [DEGREES()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_degrees) | Convert radians to degrees |
| [DIV](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_div) | Integer division |
| [EXP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_exp) | Raise to the power of |
| [FLOOR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_floor) | Return the largest integer value not greater than the argument |
| [LN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ln) | Return the natural logarithm of the argument |
| [LOG()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log) | Return the natural logarithm of the first argument |
| [LOG10()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log10) | Return the base-10 logarithm of the argument |
| [LOG2()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log2) | Return the base-2 logarithm of the argument |
| [MOD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mod) | Return the remainder |
| [PI()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_pi) | Return the value of pi |
| [POW()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_pow) | Return the argument raised to the specified power |
| [POWER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_power) | Return the argument raised to the specified power |
| [RADIANS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_radians) | Return argument converted to radians |
| [RAND()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) | Return a random floating-point value |
| [ROUND()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) | Round the argument |
| [SIGN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sign) | Return the sign of the argument |
| [SIN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sin) | Return the sine of the argument |
| [SQRT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sqrt) | Return the square root of the argument |
| [TAN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_tan) | Return the tangent of the argument |
| [TRUNCATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_truncate) | Truncate to specified number of decimal places |

### 12.6.1 Arithmetic Operators

**Table 12.9 Arithmetic Operators**

|  |  |
| --- | --- |
| Name | Description |
| [%, MOD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_mod) | Modulo operator |
| [\*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_times) | Multiplication operator |
| [+](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_plus) | Addition operator |
| [-](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_minus) | Minus operator |
| [-](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_unary-minus) | Change the sign of the argument |
| [/](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_divide) | Division operator |
| [DIV](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_div) | Integer division |

The usual arithmetic operators are available. The result is determined according to the following rules:

In the case of [**-**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_minus), [**+**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_plus), and [**\***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_times), the result is calculated with [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) (64-bit) precision if both operands are integers.

If both operands are integers and any of them are unsigned, the result is an unsigned integer. For subtraction, if the [**NO\_UNSIGNED\_SUBTRACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_unsigned_subtraction) SQL mode is enabled, the result is signed even if any operand is unsigned.

If any of the operands of a [**+**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_plus), [**-**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_minus), [**/**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_divide), [**\***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_times), [**%**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_mod) is a real or string value, the precision of the result is the precision of the operand with the maximum precision.

In division performed with [**/**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_divide), the scale of the result when using two exact-value operands is the scale of the first operand plus the value of the [**div\_precision\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_div_precision_increment) system variable (which is 4 by default). For example, the result of the expression **5.05 / 0.014** has a scale of six decimal places (**360.714286**).

These rules are applied for each operation, such that nested calculations imply the precision of each component. Hence, **(14620 / 9432456) / (24250 / 9432456)**, resolves first to **(0.0014) / (0.0026)**, with the final result having 8 decimal places (**0.60288653**).

Because of these rules and the way they are applied, care should be taken to ensure that components and subcomponents of a calculation use the appropriate level of precision. See [Section 12.11, “Cast Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#cast-functions).

For information about handling of overflow in numeric expression evaluation, see [Section 11.1.7, “Out-of-Range and Overflow Handling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#out-of-range-and-overflow).

Arithmetic operators apply to numbers. For other types of values, alternative operations may be available. For example, to add date values, use [**DATE\_ADD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-add); see [Section 12.7, “Date and Time Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#date-and-time-functions).

**[+](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_plus)**

Addition:

mysql> **SELECT 3+5;**

-> 8

**[-](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_minus)**

Subtraction:

mysql> **SELECT 3-5;**

-> -2

**[-](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_unary-minus)**

Unary minus. This operator changes the sign of the operand.

mysql> **SELECT - 2;**

-> -2

**Note**

If this operator is used with a [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), the return value is also a [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types). This means that you should avoid using **-** on integers that may have the value of −263.

**[\*](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_times)**

Multiplication:

mysql> **SELECT 3\*5;**

-> 15

mysql> **SELECT 18014398509481984\*18014398509481984.0;**

-> 324518553658426726783156020576256.0

mysql> **SELECT 18014398509481984\*18014398509481984;**

-> out-of-range error

The last expression produces an error because the result of the integer multiplication exceeds the 64-bit range of [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) calculations. (See [Section 11.1, “Numeric Data Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#numeric-types).)

**[/](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_divide)**

Division:

mysql> **SELECT 3/5;**

-> 0.60

Division by zero produces a **NULL** result:

mysql> **SELECT 102/(1-1);**

-> NULL

A division is calculated with [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) arithmetic only if performed in a context where its result is converted to an integer.

**[DIV](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_div)**

Integer division. Discards from the division result any fractional part to the right of the decimal point.

If either operand has a noninteger type, the operands are converted to [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) and divided using [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) arithmetic before converting the result to [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types). If the result exceeds **BIGINT** range, an error occurs.

mysql> **SELECT 5 DIV 2, -5 DIV 2, 5 DIV -2, -5 DIV -2;**

-> 2, -2, -2, 2

***[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_mod)*[%](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_mod)*[M](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_mod)***, [***N* MOD *M***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_mod)

Modulo operation. Returns the remainder of ***N*** divided by ***M***. For more information, see the description for the [**MOD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mod) function in [Section 12.6.2, “Mathematical Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#mathematical-functions).

### 12.6.2 数学函数

**Table 12.10 Mathematical Functions**

|  |  |
| --- | --- |
| Name | Description |
| [ABS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_abs) | Return the absolute value |
| [ACOS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_acos) | Return the arc cosine |
| [ASIN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_asin) | Return the arc sine |
| [ATAN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_atan) | Return the arc tangent |
| [ATAN2(), ATAN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_atan2) | Return the arc tangent of the two arguments |
| [CEIL()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ceil) | Return the smallest integer value not less than the argument |
| [CEILING()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ceiling) | Return the smallest integer value not less than the argument |
| [CONV()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_conv) | Convert numbers between different number bases |
| [COS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cos) | Return the cosine |
| [COT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cot) | Return the cotangent |
| [CRC32()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_crc32) | Compute a cyclic redundancy check value |
| [DEGREES()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_degrees) | Convert radians to degrees |
| [EXP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_exp) | Raise to the power of |
| [FLOOR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_floor) | Return the largest integer value not greater than the argument |
| [LN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ln) | Return the natural logarithm of the argument |
| [LOG()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log) | Return the natural logarithm of the first argument |
| [LOG10()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log10) | Return the base-10 logarithm of the argument |
| [LOG2()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log2) | Return the base-2 logarithm of the argument |
| [MOD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mod) | Return the remainder |
| [PI()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_pi) | Return the value of pi |
| [POW()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_pow) | Return the argument raised to the specified power |
| [POWER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_power) | Return the argument raised to the specified power |
| [RADIANS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_radians) | Return argument converted to radians |
| [RAND()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) | Return a random floating-point value |
| [ROUND()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) | Round the argument |
| [SIGN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sign) | Return the sign of the argument |
| [SIN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sin) | Return the sine of the argument |
| [SQRT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sqrt) | Return the square root of the argument |
| [TAN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_tan) | Return the tangent of the argument |
| [TRUNCATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_truncate) | Truncate to specified number of decimal places |

All mathematical functions return **NULL** in the event of an error.

**[ABS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_abs)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_abs)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_abs)**

Returns the absolute value of ***X***, or **NULL** if ***X*** is **NULL**.

The result type is derived from the argument type. An implication of this is that [**ABS(-9223372036854775808)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_abs) produces an error because the result cannot be stored in a signed **BIGINT** value.

mysql> **SELECT ABS(2);**

-> 2

mysql> **SELECT ABS(-32);**

-> 32

This function is safe to use with [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) values.

**[ACOS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_acos)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_acos)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_acos)**

Returns the arc cosine of ***X***, that is, the value whose cosine is ***X***. Returns **NULL** if ***X*** is not in the range **-1** to **1**.

mysql> **SELECT ACOS(1);**

-> 0

mysql> **SELECT ACOS(1.0001);**

-> NULL

mysql> **SELECT ACOS(0);**

-> 1.5707963267949

**[ASIN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_asin)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_asin)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_asin)**

Returns the arc sine of ***X***, that is, the value whose sine is ***X***. Returns **NULL** if ***X*** is not in the range **-1** to **1**.

mysql> **SELECT ASIN(0.2);**

-> 0.20135792079033

mysql> **SELECT ASIN('foo');**

+-------------+

| ASIN('foo') |

+-------------+

| 0 |

+-------------+

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS;**

+---------+------+-----------------------------------------+

| Level | Code | Message |

+---------+------+-----------------------------------------+

| Warning | 1292 | Truncated incorrect DOUBLE value: 'foo' |

+---------+------+-----------------------------------------+

**[ATAN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_atan)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_atan)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_atan)**

Returns the arc tangent of ***X***, that is, the value whose tangent is ***X***.

mysql> **SELECT ATAN(2);**

-> 1.1071487177941

mysql> **SELECT ATAN(-2);**

-> -1.1071487177941

**[ATAN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_atan2)*[Y](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_atan2)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_atan2)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_atan2)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_atan2)**, [**ATAN2(*Y*,*X*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_atan2)

Returns the arc tangent of the two variables ***X*** and ***Y***. It is similar to calculating the arc tangent of ***Y* / *X***, except that the signs of both arguments are used to determine the quadrant of the result.

mysql> **SELECT ATAN(-2,2);**

-> -0.78539816339745

mysql> **SELECT ATAN2(PI(),0);**

-> 1.5707963267949

**[CEIL(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ceil)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ceil)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ceil)**

**[CEIL()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ceil)** is a synonym for [**CEILING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ceiling).

**[CEILING(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ceiling)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ceiling)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ceiling)**

Returns the smallest integer value not less than ***X***.

mysql> **SELECT CEILING(1.23);**

-> 2

mysql> **SELECT CEILING(-1.23);**

-> -1

For exact-value numeric arguments, the return value has an exact-value numeric type. For string or floating-point arguments, the return value has a floating-point type.

**[CONV(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_conv)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_conv)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_conv)*[from\_base](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_conv)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_conv)*[to\_base](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_conv)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_conv)**

Converts numbers between different number bases. Returns a string representation of the number ***N***, converted from base ***from\_base*** to base ***to\_base***. Returns **NULL** if any argument is **NULL**. The argument ***N*** is interpreted as an integer, but may be specified as an integer or a string. The minimum base is **2** and the maximum base is **36**. If ***from\_base*** is a negative number, ***N*** is regarded as a signed number. Otherwise, ***N*** is treated as unsigned. [**CONV()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_conv) works with 64-bit precision.

mysql> **SELECT CONV('a',16,2);**

-> '1010'

mysql> **SELECT CONV('6E',18,8);**

-> '172'

mysql> **SELECT CONV(-17,10,-18);**

-> '-H'

mysql> **SELECT CONV(10+'10'+'10'+X'0a',10,10);**

-> '40'

**[COS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cos)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cos)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cos)**

Returns the cosine of ***X***, where ***X*** is given in radians.

mysql> **SELECT COS(PI());**

-> -1

**[COT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cot)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cot)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cot)**

Returns the cotangent of ***X***.

mysql> **SELECT COT(12);**

-> -1.5726734063977

mysql> **SELECT COT(0);**

-> out-of-range error

**[CRC32(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_crc32)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_crc32)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_crc32)**

Computes a cyclic redundancy check value and returns a 32-bit unsigned value. The result is **NULL** if the argument is **NULL**. The argument is expected to be a string and (if possible) is treated as one if it is not.

mysql> **SELECT CRC32('MySQL');**

-> 3259397556

mysql> **SELECT CRC32('mysql');**

-> 2501908538

**[DEGREES(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_degrees)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_degrees)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_degrees)**

Returns the argument ***X***, converted from radians to degrees.

mysql> **SELECT DEGREES(PI());**

-> 180

mysql> **SELECT DEGREES(PI() / 2);**

-> 90

**[EXP(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_exp)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_exp)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_exp)**

Returns the value of e (the base of natural logarithms) raised to the power of ***X***. The inverse of this function is [**LOG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log) (using a single argument only) or [**LN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ln).

mysql> **SELECT EXP(2);**

-> 7.3890560989307

mysql> **SELECT EXP(-2);**

-> 0.13533528323661

mysql> **SELECT EXP(0);**

-> 1

**[FLOOR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_floor)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_floor)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_floor)**

Returns the largest integer value not greater than ***X***.

mysql> **SELECT FLOOR(1.23), FLOOR(-1.23);**

-> 1, -2

For exact-value numeric arguments, the return value has an exact-value numeric type. For string or floating-point arguments, the return value has a floating-point type.

[**FORMAT(*X*,*D*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format)

Formats the number ***X*** to a format like **'#,###,###.##'**, rounded to ***D*** decimal places, and returns the result as a string. For details, see [Section 12.8, “String Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#string-functions).

**[HEX(N\_or\_S)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_hex)**

This function can be used to obtain a hexadecimal representation of a decimal number or a string; the manner in which it does so varies according to the argument's type. See this function's description in [Section 12.8, “String Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#string-functions), for details.

**[LN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ln)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ln)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ln)**

Returns the natural logarithm of ***X***; that is, the base-e logarithm of ***X***. If ***X*** is less than or equal to 0.0E0, the function returns **NULL** and a warning “Invalid argument for logarithm” is reported.

mysql> **SELECT LN(2);**

-> 0.69314718055995

mysql> **SELECT LN(-2);**

-> NULL

This function is synonymous with [**LOG(*X*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log). The inverse of this function is the [**EXP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_exp) function.

**[LOG(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_log)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_log)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_log)**, [**LOG(*B*,*X*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log)

If called with one parameter, this function returns the natural logarithm of ***X***. If ***X*** is less than or equal to 0.0E0, the function returns **NULL** and a warning “Invalid argument for logarithm” is reported.

The inverse of this function (when called with a single argument) is the [**EXP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_exp) function.

mysql> **SELECT LOG(2);**

-> 0.69314718055995

mysql> **SELECT LOG(-2);**

-> NULL

If called with two parameters, this function returns the logarithm of ***X*** to the base ***B***. If ***X*** is less than or equal to 0, or if ***B*** is less than or equal to 1, then **NULL** is returned.

mysql> **SELECT LOG(2,65536);**

-> 16

mysql> **SELECT LOG(10,100);**

-> 2

mysql> **SELECT LOG(1,100);**

-> NULL

[**LOG(*B*,*X*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log) is equivalent to [**LOG(*X*) / LOG(*B*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log).

**[LOG2(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_log2)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_log2)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_log2)**

Returns the base-2 logarithm of ***X***. If ***X*** is less than or equal to 0.0E0, the function returns **NULL** and a warning “Invalid argument for logarithm” is reported.

mysql> **SELECT LOG2(65536);**

-> 16

mysql> **SELECT LOG2(-100);**

-> NULL

[**LOG2()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log2) is useful for finding out how many bits a number requires for storage. This function is equivalent to the expression [**LOG(*X*) / LOG(2)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log).

**[LOG10(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_log10)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_log10)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_log10)**

Returns the base-10 logarithm of ***X***. If ***X*** is less than or equal to 0.0E0, the function returns **NULL** and a warning “Invalid argument for logarithm” is reported.

mysql> **SELECT LOG10(2);**

-> 0.30102999566398

mysql> **SELECT LOG10(100);**

-> 2

mysql> **SELECT LOG10(-100);**

-> NULL

[**LOG10(*X*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log10) is equivalent to [**LOG(10,*X*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_log).

**[MOD(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mod)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mod)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mod)*[M](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mod)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mod)**, [***N* % *M***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_mod), [***N* MOD *M***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_mod)

Modulo operation. Returns the remainder of ***N*** divided by ***M***.

mysql> **SELECT MOD(234, 10);**

-> 4

mysql> **SELECT 253 % 7;**

-> 1

mysql> **SELECT MOD(29,9);**

-> 2

mysql> **SELECT 29 MOD 9;**

-> 2

This function is safe to use with [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) values.

[**MOD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mod) also works on values that have a fractional part and returns the exact remainder after division:

mysql> **SELECT MOD(34.5,3);**

-> 1.5

[**MOD(*N*,0)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mod) returns **NULL**.

**[PI()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_pi)**

Returns the value of π (pi). The default number of decimal places displayed is seven, but MySQL uses the full double-precision value internally.

mysql> **SELECT PI();**

-> 3.141593

mysql> **SELECT PI()+0.000000000000000000;**

-> 3.141592653589793116

**[POW(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_pow)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_pow)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_pow)*[Y](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_pow)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_pow)**

Returns the value of ***X*** raised to the power of ***Y***.

mysql> **SELECT POW(2,2);**

-> 4

mysql> **SELECT POW(2,-2);**

-> 0.25

**[POWER(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_power)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_power)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_power)*[Y](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_power)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_power)**

This is a synonym for [**POW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_pow).

**[RADIANS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_radians)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_radians)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_radians)**

Returns the argument ***X***, converted from degrees to radians. (Note that π radians equals 180 degrees.)

mysql> **SELECT RADIANS(90);**

-> 1.5707963267949

**[RAND([](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rand)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rand)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rand)**

Returns a random floating-point value ***v*** in the range **0** <= ***v*** < **1.0**. To obtain a random integer ***R*** in the range ***i*** <= ***R*** < ***j***, use the expression [**FLOOR(*i* + RAND() \* (*j***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_floor) − ***i*))**. For example, to obtain a random integer in the range the range **7** <= ***R*** < **12**, use the following statement:

SELECT FLOOR(7 + (RAND() \* 5));

If an integer argument ***N*** is specified, it is used as the seed value:

With a constant initializer argument, the seed is initialized once when the statement is prepared, prior to execution.

With a nonconstant initializer argument (such as a column name), the seed is initialized with the value for each invocation of [**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand).

One implication of this behavior is that for equal argument values, [**RAND(*N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) returns the same value each time, and thus produces a repeatable sequence of column values. In the following example, the sequence of values produced by **RAND(3)** is the same both places it occurs.

mysql> **CREATE TABLE t (i INT);**

Query OK, 0 rows affected (0.42 sec)

mysql> **INSERT INTO t VALUES(1),(2),(3);**

Query OK, 3 rows affected (0.00 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> **SELECT i, RAND() FROM t;**

+------+------------------+

| i | RAND() |

+------+------------------+

| 1 | 0.61914388706828 |

| 2 | 0.93845168309142 |

| 3 | 0.83482678498591 |

+------+------------------+

3 rows in set (0.00 sec)

mysql> **SELECT i, RAND(3) FROM t;**

+------+------------------+

| i | RAND(3) |

+------+------------------+

| 1 | 0.90576975597606 |

| 2 | 0.37307905813035 |

| 3 | 0.14808605345719 |

+------+------------------+

3 rows in set (0.00 sec)

mysql> **SELECT i, RAND() FROM t;**

+------+------------------+

| i | RAND() |

+------+------------------+

| 1 | 0.35877890638893 |

| 2 | 0.28941420772058 |

| 3 | 0.37073435016976 |

+------+------------------+

3 rows in set (0.00 sec)

mysql> **SELECT i, RAND(3) FROM t;**

+------+------------------+

| i | RAND(3) |

+------+------------------+

| 1 | 0.90576975597606 |

| 2 | 0.37307905813035 |

| 3 | 0.14808605345719 |

+------+------------------+

3 rows in set (0.01 sec)

[**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) in a **WHERE** clause is evaluated for every row (when selecting from one table) or combination of rows (when selecting from a multiple-table join). Thus, for optimizer purposes, [**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) is not a constant value and cannot be used for index optimizations. For more information, see [Section 8.2.1.20, “Function Call Optimization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#function-optimization).

Use of a column with [**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) values in an **ORDER BY** or **GROUP BY** clause may yield unexpected results because for either clause a [**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) expression can be evaluated multiple times for the same row, each time returning a different result. If the goal is to retrieve rows in random order, you can use a statement like this:

SELECT \* FROM ***tbl\_name*** ORDER BY RAND();

To select a random sample from a set of rows, combine **ORDER BY RAND()** with **LIMIT**:

SELECT \* FROM table1, table2 WHERE a=b AND c<d ORDER BY RAND() LIMIT 1000;

[**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) is not meant to be a perfect random generator. It is a fast way to generate random numbers on demand that is portable between platforms for the same MySQL version.

This function is unsafe for statement-based replication. A warning is logged if you use this function when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**.

**[ROUND(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_round)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_round)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_round)**, [**ROUND(*X*,*D*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round)

Rounds the argument ***X*** to ***D*** decimal places. The rounding algorithm depends on the data type of ***X***. ***D*** defaults to 0 if not specified. ***D*** can be negative to cause ***D*** digits left of the decimal point of the value ***X*** to become zero. The maximum absolute value for ***D*** is 30; any digits in excess of 30 (or -30) are truncated.

mysql> **SELECT ROUND(-1.23);**

-> -1

mysql> **SELECT ROUND(-1.58);**

-> -2

mysql> **SELECT ROUND(1.58);**

-> 2

mysql> **SELECT ROUND(1.298, 1);**

-> 1.3

mysql> **SELECT ROUND(1.298, 0);**

-> 1

mysql> **SELECT ROUND(23.298, -1);**

-> 20

mysql> **SELECT ROUND(.12345678901234567890123456789012345, 35);**

-> 0.123456789012345678901234567890

The return value has the same type as the first argument (assuming that it is integer, double, or decimal). This means that for an integer argument, the result is an integer (no decimal places):

mysql> **SELECT ROUND(150.000,2), ROUND(150,2);**

+------------------+--------------+

| ROUND(150.000,2) | ROUND(150,2) |

+------------------+--------------+

| 150.00 | 150 |

+------------------+--------------+

[**ROUND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) uses the following rules depending on the type of the first argument:

For exact-value numbers, [**ROUND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) uses the “round half away from zero” or “round toward nearest” rule: A value with a fractional part of .5 or greater is rounded up to the next integer if positive or down to the next integer if negative. (In other words, it is rounded away from zero.) A value with a fractional part less than .5 is rounded down to the next integer if positive or up to the next integer if negative.

For approximate-value numbers, the result depends on the C library. On many systems, this means that [**ROUND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) uses the “round to nearest even” rule: A value with a fractional part exactly halfway between two integers is rounded to the nearest even integer.

The following example shows how rounding differs for exact and approximate values:

mysql> **SELECT ROUND(2.5), ROUND(25E-1);**

+------------+--------------+

| ROUND(2.5) | ROUND(25E-1) |

+------------+--------------+

| 3 | 2 |

+------------+--------------+

For more information, see [Section 12.25, “Precision Math”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math).

In MySQL 8.0.21 and later, the data type returned by **ROUND()** (and [**TRUNCATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_truncate)) is determined according to the rules listed here:

When the first argument is of any integer type, the return type is always [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types).

When the first argument is of any floating-point type or of any non-numeric type, the return type is always [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types).

When the first argument is a [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) value, the return type is also **DECIMAL**.

The type attributes for the return value are also copied from the first argument, except in the case of **DECIMAL**, when the second argument is a constant value.

When the desired number of decimal places is less than the scale of the argument, the scale and the precision of the result are adjusted accordingly.

In addition, for **ROUND()** (but not for the [**TRUNCATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_truncate) function), the precision is extended by one place to accomodate rounding that increases the number of significant digits. If the second argument is negative, the return type is adjusted such that its scale is 0, with a corresponding precision. For example, **ROUND(99.999, 2)** returns **100.00**—the first argument is **DECIMAL(5, 3)**, and the return type is **DECIMAL(5, 2)**.

If the second argument is negative, the return type has scale 0 and a corresponding precision; **ROUND(99.999, -1)** returns **100**, which is **DECIMAL(3, 0)**.

**[SIGN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sign)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sign)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sign)**

Returns the sign of the argument as **-1**, **0**, or **1**, depending on whether ***X*** is negative, zero, or positive.

mysql> **SELECT SIGN(-32);**

-> -1

mysql> **SELECT SIGN(0);**

-> 0

mysql> **SELECT SIGN(234);**

-> 1

**[SIN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sin)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sin)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sin)**

Returns the sine of ***X***, where ***X*** is given in radians.

mysql> **SELECT SIN(PI());**

-> 1.2246063538224e-16

mysql> **SELECT ROUND(SIN(PI()));**

-> 0

**[SQRT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sqrt)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sqrt)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sqrt)**

Returns the square root of a nonnegative number ***X***.

mysql> **SELECT SQRT(4);**

-> 2

mysql> **SELECT SQRT(20);**

-> 4.4721359549996

mysql> **SELECT SQRT(-16);**

-> NULL

**[TAN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_tan)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_tan)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_tan)**

Returns the tangent of ***X***, where ***X*** is given in radians.

mysql> **SELECT TAN(PI());**

-> -1.2246063538224e-16

mysql> **SELECT TAN(PI()+1);**

-> 1.5574077246549

**[TRUNCATE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_truncate)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_truncate)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_truncate)*[D](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_truncate)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_truncate)**

Returns the number ***X***, truncated to ***D*** decimal places. If ***D*** is **0**, the result has no decimal point or fractional part. ***D*** can be negative to cause ***D*** digits left of the decimal point of the value ***X*** to become zero.

mysql> **SELECT TRUNCATE(1.223,1);**

-> 1.2

mysql> **SELECT TRUNCATE(1.999,1);**

-> 1.9

mysql> **SELECT TRUNCATE(1.999,0);**

-> 1

mysql> **SELECT TRUNCATE(-1.999,1);**

-> -1.9

mysql> **SELECT TRUNCATE(122,-2);**

-> 100

mysql> **SELECT TRUNCATE(10.28\*100,0);**

-> 1028

All numbers are rounded toward zero.

In MySQL 8.0.21 and later, the data type returned by **TRUNCATE()** follows the same rules that determine the return type of the **ROUND()** function; for details, see the description for [**ROUND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round).

## 12.7 Date and Time Functions

This section describes the functions that can be used to manipulate temporal values. See [Section 11.2, “Date and Time Data Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#date-and-time-types), for a description of the range of values each date and time type has and the valid formats in which values may be specified.

**Table 12.11 Date and Time Functions**

| **Name** | **Description** |
| --- | --- |
| [**ADDDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_adddate) | Add time values (intervals) to a date value |
| [**ADDTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_addtime) | Add time |
| [**CONVERT\_TZ()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert-tz) | Convert from one time zone to another |
| [**CURDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_curdate) | Return the current date |
| [**CURRENT\_DATE()**, **CURRENT\_DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-date) | Synonyms for CURDATE() |
| [**CURRENT\_TIME()**, **CURRENT\_TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-time) | Synonyms for CURTIME() |
| [**CURRENT\_TIMESTAMP()**, **CURRENT\_TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-timestamp) | Synonyms for NOW() |
| [**CURTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_curtime) | Return the current time |
| [**DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date) | Extract the date part of a date or datetime expression |
| [**DATE\_ADD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-add) | Add time values (intervals) to a date value |
| [**DATE\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-format) | Format date as specified |
| [**DATE\_SUB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-sub) | Subtract a time value (interval) from a date |
| [**DATEDIFF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_datediff) | Subtract two dates |
| [**DAY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_day) | Synonym for DAYOFMONTH() |
| [**DAYNAME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dayname) | Return the name of the weekday |
| [**DAYOFMONTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dayofmonth) | Return the day of the month (0-31) |
| [**DAYOFWEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dayofweek) | Return the weekday index of the argument |
| [**DAYOFYEAR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dayofyear) | Return the day of the year (1-366) |
| [**EXTRACT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extract) | Extract part of a date |
| [**FROM\_DAYS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-days) | Convert a day number to a date |
| [**FROM\_UNIXTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-unixtime) | Format Unix timestamp as a date |
| [**GET\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | Return a date format string |
| [**HOUR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hour) | Extract the hour |
| [**LAST\_DAY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-day) | Return the last day of the month for the argument |
| [**LOCALTIME()**, **LOCALTIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_localtime) | Synonym for NOW() |
| [**LOCALTIMESTAMP**, **LOCALTIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_localtimestamp) | Synonym for NOW() |
| [**MAKEDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_makedate) | Create a date from the year and day of year |
| [**MAKETIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_maketime) | Create time from hour, minute, second |
| [**MICROSECOND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_microsecond) | Return the microseconds from argument |
| [**MINUTE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_minute) | Return the minute from the argument |
| [**MONTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_month) | Return the month from the date passed |
| [**MONTHNAME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_monthname) | Return the name of the month |
| [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) | Return the current date and time |
| [**PERIOD\_ADD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_period-add) | Add a period to a year-month |
| [**PERIOD\_DIFF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_period-diff) | Return the number of months between periods |
| [**QUARTER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_quarter) | Return the quarter from a date argument |
| [**SEC\_TO\_TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sec-to-time) | Converts seconds to 'hh:mm:ss' format |
| [**SECOND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_second) | Return the second (0-59) |
| [**STR\_TO\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_str-to-date) | Convert a string to a date |
| [**SUBDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_subdate) | Synonym for DATE\_SUB() when invoked with three arguments |
| [**SUBTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_subtime) | Subtract times |
| [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate) | Return the time at which the function executes |
| [**TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_time) | Extract the time portion of the expression passed |
| [**TIME\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_time-format) | Format as time |
| [**TIME\_TO\_SEC()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_time-to-sec) | Return the argument converted to seconds |
| [**TIMEDIFF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timediff) | Subtract time |
| [**TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timestamp) | With a single argument, this function returns the date or datetime expression; with two arguments, the sum of the arguments |
| [**TIMESTAMPADD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timestampadd) | Add an interval to a datetime expression |
| [**TIMESTAMPDIFF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timestampdiff) | Subtract an interval from a datetime expression |
| [**TO\_DAYS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-days) | Return the date argument converted to days |
| [**TO\_SECONDS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-seconds) | Return the date or datetime argument converted to seconds since Year 0 |
| [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) | Return a Unix timestamp |
| [**UTC\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-date) | Return the current UTC date |
| [**UTC\_TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-time) | Return the current UTC time |
| [**UTC\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-timestamp) | Return the current UTC date and time |
| [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) | Return the week number |
| [**WEEKDAY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weekday) | Return the weekday index |
| [**WEEKOFYEAR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weekofyear) | Return the calendar week of the date (1-53) |
| [**YEAR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_year) | Return the year |
| [**YEARWEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_yearweek) | Return the year and week |

Here is an example that uses date functions. The following query selects all rows with a ***date\_col*** value from within the last 30 days:

mysql> **SELECT *something* FROM *tbl\_name***

-> **WHERE DATE\_SUB(CURDATE(),INTERVAL 30 DAY) <= *date\_col*;**

The query also selects rows with dates that lie in the future.

Functions that expect date values usually accept datetime values and ignore the time part. Functions that expect time values usually accept datetime values and ignore the date part.

Functions that return the current date or time each are evaluated only once per query at the start of query execution. This means that multiple references to a function such as [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) within a single query always produce the same result. (For our purposes, a single query also includes a call to a stored program (stored routine, trigger, or event) and all subprograms called by that program.) This principle also applies to [**CURDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_curdate), [**CURTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_curtime), [**UTC\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-date), [**UTC\_TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-time), [**UTC\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-timestamp), and to any of their synonyms.

The [**CURRENT\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-timestamp), [**CURRENT\_TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-time), [**CURRENT\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-date), and [**FROM\_UNIXTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-unixtime) functions return values in the current session time zone, which is available as the session value of the [**time\_zone**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_time_zone) system variable. In addition, [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) assumes that its argument is a datetime value in the session time zone. See [Section 5.1.15, “MySQL Server Time Zone Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#time-zone-support).

Some date functions can be used with “zero” dates or incomplete dates such as **'2001-11-00'**, whereas others cannot. Functions that extract parts of dates typically work with incomplete dates and thus can return 0 when you might otherwise expect a nonzero value. For example:

mysql> **SELECT DAYOFMONTH('2001-11-00'), MONTH('2005-00-00');**

-> 0, 0

Other functions expect complete dates and return **NULL** for incomplete dates. These include functions that perform date arithmetic or that map parts of dates to names. For example:

mysql> **SELECT DATE\_ADD('2006-05-00',INTERVAL 1 DAY);**

-> NULL

mysql> **SELECT DAYNAME('2006-05-00');**

-> NULL

Several functions are strict when passed a [**DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date) function value as their argument and reject incomplete dates with a day part of zero: [**CONVERT\_TZ()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert-tz), [**DATE\_ADD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-add), [**DATE\_SUB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-sub), [**DAYOFYEAR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dayofyear), [**TIMESTAMPDIFF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timestampdiff), [**TO\_DAYS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-days), [**TO\_SECONDS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-seconds), [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week), [**WEEKDAY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weekday), [**WEEKOFYEAR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weekofyear), [**YEARWEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_yearweek).

Fractional seconds for **TIME**, **DATETIME**, and **TIMESTAMP** values are supported, with up to microsecond precision. Functions that take temporal arguments accept values with fractional seconds. Return values from temporal functions include fractional seconds as appropriate.

**[ADDDATE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_adddate)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_adddate)*[,INTERVAL](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_adddate)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_adddate)**[unit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_adddate)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_adddate)**, [**ADDDATE(*expr*,*days*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_adddate)

When invoked with the **INTERVAL** form of the second argument, [**ADDDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_adddate) is a synonym for [**DATE\_ADD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-add). The related function [**SUBDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_subdate) is a synonym for [**DATE\_SUB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-sub). For information on the **INTERVAL** ***unit*** argument, see [Temporal Intervals](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#temporal-intervals).

mysql> **SELECT DATE\_ADD('2008-01-02', INTERVAL 31 DAY);**

-> '2008-02-02'

mysql> **SELECT ADDDATE('2008-01-02', INTERVAL 31 DAY);**

-> '2008-02-02'

When invoked with the ***days*** form of the second argument, MySQL treats it as an integer number of days to be added to ***expr***.

mysql> **SELECT ADDDATE('2008-01-02', 31);**

-> '2008-02-02'

**[ADDTIME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_addtime)*[expr1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_addtime)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_addtime)*[expr2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_addtime)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_addtime)**

**[ADDTIME()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_addtime)** adds ***expr2*** to ***expr1*** and returns the result. ***expr1*** is a time or datetime expression, and ***expr2*** is a time expression.

mysql> **SELECT ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002');**

-> '2008-01-02 01:01:01.000001'

mysql> **SELECT ADDTIME('01:00:00.999999', '02:00:00.999998');**

-> '03:00:01.999997'

**[CONVERT\_TZ(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert-tz)*[dt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert-tz)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert-tz)*[from\_tz](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert-tz)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert-tz)*[to\_tz](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert-tz)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert-tz)**

**[CONVERT\_TZ()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert-tz)** converts a datetime value ***dt*** from the time zone given by ***from\_tz*** to the time zone given by ***to\_tz*** and returns the resulting value. Time zones are specified as described in [Section 5.1.15, “MySQL Server Time Zone Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#time-zone-support). This function returns **NULL** if the arguments are invalid.

If the value falls out of the supported range of the [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) type when converted from ***from\_tz*** to UTC, no conversion occurs. The [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) range is described in [Section 11.2.1, “Date and Time Data Type Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#date-and-time-type-syntax).

mysql> **SELECT CONVERT\_TZ('2004-01-01 12:00:00','GMT','MET');**

-> '2004-01-01 13:00:00'

mysql> **SELECT CONVERT\_TZ('2004-01-01 12:00:00','+00:00','+10:00');**

-> '2004-01-01 22:00:00'

**Note**

To use named time zones such as **'MET'** or **'Europe/Amsterdam'**, the time zone tables must be properly set up. For instructions, see [Section 5.1.15, “MySQL Server Time Zone Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#time-zone-support).

**[CURDATE()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_curdate)**

Returns the current date as a value in **'*YYYY-MM-DD*'** or ***YYYYMMDD*** format, depending on whether the function is used in string or numeric context.

mysql> **SELECT CURDATE();**

-> '2008-06-13'

mysql> **SELECT CURDATE() + 0;**

-> 20080613

**[CURRENT\_DATE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_current-date)**, [**CURRENT\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-date)

**[CURRENT\_DATE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_current-date)** and [**CURRENT\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-date) are synonyms for [**CURDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_curdate).

**[CURRENT\_TIME](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_current-time)**, [**CURRENT\_TIME([*fsp*])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-time)

**[CURRENT\_TIME](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_current-time)** and [**CURRENT\_TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-time) are synonyms for [**CURTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_curtime).

**[CURRENT\_TIMESTAMP](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_current-timestamp)**, [**CURRENT\_TIMESTAMP([*fsp*])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-timestamp)

**[CURRENT\_TIMESTAMP](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_current-timestamp)** and [**CURRENT\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-timestamp) are synonyms for [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now).

**[CURTIME([](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_curtime)*[fsp](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_curtime)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_curtime)**

Returns the current time as a value in ***'hh:mm:ss'*** or ***hhmmss*** format, depending on whether the function is used in string or numeric context. The value is expressed in the session time zone.

If the ***fsp*** argument is given to specify a fractional seconds precision from 0 to 6, the return value includes a fractional seconds part of that many digits.

mysql> **SELECT CURTIME();**

-> '23:50:26'

mysql> **SELECT CURTIME() + 0;**

-> 235026.000000

**[DATE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date)**

Extracts the date part of the date or datetime expression ***expr***.

mysql> **SELECT DATE('2003-12-31 01:02:03');**

-> '2003-12-31'

**[DATEDIFF(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_datediff)*[expr1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_datediff)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_datediff)*[expr2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_datediff)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_datediff)**

**[DATEDIFF()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_datediff)** returns ***expr1*** − ***expr2*** expressed as a value in days from one date to the other. ***expr1*** and ***expr2*** are date or date-and-time expressions. Only the date parts of the values are used in the calculation.

mysql> **SELECT DATEDIFF('2007-12-31 23:59:59','2007-12-30');**

-> 1

mysql> **SELECT DATEDIFF('2010-11-30 23:59:59','2010-12-31');**

-> -31

**[DATE\_ADD(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-add)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-add)*[,INTERVAL](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-add)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-add)**[unit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-add)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-add)**, [**DATE\_SUB(*date*,INTERVAL *expr* *unit*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-sub)

These functions perform date arithmetic. The ***date*** argument specifies the starting date or datetime value. ***expr*** is an expression specifying the interval value to be added or subtracted from the starting date. ***expr*** is evaluated as a string; it may start with a **-** for negative intervals. ***unit*** is a keyword indicating the units in which the expression should be interpreted.

For more information about temporal interval syntax, including a full list of ***unit*** specifiers, the expected form of the ***expr*** argument for each ***unit*** value, and rules for operand interpretation in temporal arithmetic, see [Temporal Intervals](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#temporal-intervals).

The return value depends on the arguments:

[**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) if the ***date*** argument is a [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value and your calculations involve only **YEAR**, **MONTH**, and **DAY** parts (that is, no time parts).

[**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) if the first argument is a [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) (or [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime)) value, or if the first argument is a [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) and the ***unit*** value uses **HOURS**, **MINUTES**, or **SECONDS**.

String otherwise.

To ensure that the result is [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), you can use [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) to convert the first argument to [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime).

mysql> **SELECT DATE\_ADD('2018-05-01',INTERVAL 1 DAY);**

-> '2018-05-02'

mysql> **SELECT DATE\_SUB('2018-05-01',INTERVAL 1 YEAR);**

-> '2017-05-01'

mysql> **SELECT DATE\_ADD('2020-12-31 23:59:59',**

-> **INTERVAL 1 SECOND);**

-> '2021-01-01 00:00:00'

mysql> **SELECT DATE\_ADD('2018-12-31 23:59:59',**

-> **INTERVAL 1 DAY);**

-> '2019-01-01 23:59:59'

mysql> **SELECT DATE\_ADD('2100-12-31 23:59:59',**

-> **INTERVAL '1:1' MINUTE\_SECOND);**

-> '2101-01-01 00:01:00'

mysql> **SELECT DATE\_SUB('2025-01-01 00:00:00',**

-> **INTERVAL '1 1:1:1' DAY\_SECOND);**

-> '2024-12-30 22:58:59'

mysql> **SELECT DATE\_ADD('1900-01-01 00:00:00',**

-> **INTERVAL '-1 10' DAY\_HOUR);**

-> '1899-12-30 14:00:00'

mysql> **SELECT DATE\_SUB('1998-01-02', INTERVAL 31 DAY);**

-> '1997-12-02'

mysql> **SELECT DATE\_ADD('1992-12-31 23:59:59.000002',**

-> **INTERVAL '1.999999' SECOND\_MICROSECOND);**

-> '1993-01-01 00:00:01.000001'

**[DATE\_FORMAT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-format)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-format)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-format)*[format](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-format)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-format)**

Formats the ***date*** value according to the ***format*** string.

The specifiers shown in the following table may be used in the ***format*** string. The **%** character is required before format specifier characters. The specifiers apply to other functions as well: [**STR\_TO\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_str-to-date), [**TIME\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_time-format), [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp).

| **Specifier** | **Description** |
| --- | --- |
| **%a** | Abbreviated weekday name (**Sun**..**Sat**) |
| **%b** | Abbreviated month name (**Jan**..**Dec**) |
| **%c** | Month, numeric (**0**..**12**) |
| **%D** | Day of the month with English suffix (**0th**, **1st**, **2nd**, **3rd**, …) |
| **%d** | Day of the month, numeric (**00**..**31**) |
| **%e** | Day of the month, numeric (**0**..**31**) |
| **%f** | Microseconds (**000000**..**999999**) |
| **%H** | Hour (**00**..**23**) |
| **%h** | Hour (**01**..**12**) |
| **%I** | Hour (**01**..**12**) |
| **%i** | Minutes, numeric (**00**..**59**) |
| **%j** | Day of year (**001**..**366**) |
| **%k** | Hour (**0**..**23**) |
| **%l** | Hour (**1**..**12**) |
| **%M** | Month name (**January**..**December**) |
| **%m** | Month, numeric (**00**..**12**) |
| **%p** | **AM** or **PM** |
| **%r** | Time, 12-hour (***hh:mm:ss*** followed by **AM** or **PM**) |
| **%S** | Seconds (**00**..**59**) |
| **%s** | Seconds (**00**..**59**) |
| **%T** | Time, 24-hour (***hh:mm:ss***) |
| **%U** | Week (**00**..**53**), where Sunday is the first day of the week; [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) mode 0 |
| **%u** | Week (**00**..**53**), where Monday is the first day of the week; [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) mode 1 |
| **%V** | Week (**01**..**53**), where Sunday is the first day of the week; [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) mode 2; used with **%X** |
| **%v** | Week (**01**..**53**), where Monday is the first day of the week; [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) mode 3; used with **%x** |
| **%W** | Weekday name (**Sunday**..**Saturday**) |
| **%w** | Day of the week (**0**=Sunday..**6**=Saturday) |
| **%X** | Year for the week where Sunday is the first day of the week, numeric, four digits; used with **%V** |
| **%x** | Year for the week, where Monday is the first day of the week, numeric, four digits; used with **%v** |
| **%Y** | Year, numeric, four digits |
| **%y** | Year, numeric (two digits) |
| **%%** | A literal **%** character |
| **%*x*** | ***x***, for any “***x***” not listed above |

Ranges for the month and day specifiers begin with zero due to the fact that MySQL permits the storing of incomplete dates such as **'2014-00-00'**.

The language used for day and month names and abbreviations is controlled by the value of the [**lc\_time\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_time_names) system variable ([Section 10.16, “MySQL Server Locale Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#locale-support)).

For the **%U**, **%u**, **%V**, and **%v** specifiers, see the description of the [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) function for information about the mode values. The mode affects how week numbering occurs.

[**DATE\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-format) returns a string with a character set and collation given by [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) so that it can return month and weekday names containing non-ASCII characters.

mysql> **SELECT DATE\_FORMAT('2009-10-04 22:23:00', '%W %M %Y');**

-> 'Sunday October 2009'

mysql> **SELECT DATE\_FORMAT('2007-10-04 22:23:00', '%H:%i:%s');**

-> '22:23:00'

mysql> **SELECT DATE\_FORMAT('1900-10-04 22:23:00',**

->  **'%D %y %a %d %m %b %j');**

-> '4th 00 Thu 04 10 Oct 277'

mysql> **SELECT DATE\_FORMAT('1997-10-04 22:23:00',**

->  **'%H %k %I %r %T %S %w');**

-> '22 22 10 10:23:00 PM 22:23:00 00 6'

mysql> **SELECT DATE\_FORMAT('1999-01-01', '%X %V');**

-> '1998 52'

mysql> **SELECT DATE\_FORMAT('2006-06-00', '%d');**

-> '00'

**[DATE\_SUB(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-sub)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-sub)*[,INTERVAL](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-sub)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-sub)**[unit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-sub)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_date-sub)**

See the description for [**DATE\_ADD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-add).

**[DAY(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_day)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_day)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_day)**

**[DAY()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_day)** is a synonym for [**DAYOFMONTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dayofmonth).

**[DAYNAME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayname)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayname)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayname)**

Returns the name of the weekday for ***date***. The language used for the name is controlled by the value of the [**lc\_time\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_time_names) system variable ([Section 10.16, “MySQL Server Locale Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#locale-support)).

mysql> **SELECT DAYNAME('2007-02-03');**

-> 'Saturday'

**[DAYOFMONTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayofmonth)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayofmonth)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayofmonth)**

Returns the day of the month for ***date***, in the range **1** to **31**, or **0** for dates such as **'0000-00-00'** or **'2008-00-00'** that have a zero day part.

mysql> **SELECT DAYOFMONTH('2007-02-03');**

-> 3

**[DAYOFWEEK(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayofweek)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayofweek)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayofweek)**

Returns the weekday index for ***date*** (**1** = Sunday, **2** = Monday, …, **7** = Saturday). These index values correspond to the ODBC standard.

mysql> **SELECT DAYOFWEEK('2007-02-03');**

-> 7

**[DAYOFYEAR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayofyear)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayofyear)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dayofyear)**

Returns the day of the year for ***date***, in the range **1** to **366**.

mysql> **SELECT DAYOFYEAR('2007-02-03');**

-> 34

**[EXTRACT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_extract)*[unit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_extract)*[FROM](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_extract)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_extract)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_extract)**

The [**EXTRACT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extract) function uses the same kinds of ***unit*** specifiers as [**DATE\_ADD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-add) or [**DATE\_SUB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-sub), but extracts parts from the date rather than performing date arithmetic. For information on the ***unit*** argument, see [Temporal Intervals](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#temporal-intervals).

mysql> **SELECT EXTRACT(YEAR FROM '2019-07-02');**

-> 2019

mysql> **SELECT EXTRACT(YEAR\_MONTH FROM '2019-07-02 01:02:03');**

-> 201907

mysql> **SELECT EXTRACT(DAY\_MINUTE FROM '2019-07-02 01:02:03');**

-> 20102

mysql> **SELECT EXTRACT(MICROSECOND**

-> **FROM '2003-01-02 10:30:00.000123');**

-> 123

**[FROM\_DAYS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_from-days)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_from-days)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_from-days)**

Given a day number ***N***, returns a [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value.

mysql> **SELECT FROM\_DAYS(730669);**

-> '2000-07-03'

Use [**FROM\_DAYS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-days) with caution on old dates. It is not intended for use with values that precede the advent of the Gregorian calendar (1582). See [Section 12.9, “What Calendar Is Used By MySQL?”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#mysql-calendar).

**[FROM\_UNIXTIME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_from-unixtime)*[unix\_timestamp](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_from-unixtime)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_from-unixtime)*[format](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_from-unixtime)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_from-unixtime)**

Returns a representation of the ***unix\_timestamp*** argument as a value in **'*YYYY-MM-DD hh:mm:ss*'** or ***YYYYMMDDhhmmss*** format, depending on whether the function is used in a string or numeric context. ***unix\_timestamp*** is an internal timestamp value representing seconds since **'1970-01-01 00:00:00'** UTC, such as produced by the [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) function.

The return value is expressed in the session time zone. (Clients can set the session time zone as described in [Section 5.1.15, “MySQL Server Time Zone Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#time-zone-support).) The ***format*** string, if given, is used to format the result the same way as described in the entry for the [**DATE\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-format) function.

mysql> **SELECT FROM\_UNIXTIME(1447430881);**

-> '2015-11-13 10:08:01'

mysql> **SELECT FROM\_UNIXTIME(1447430881) + 0;**

-> 20151113100801

mysql> **SELECT FROM\_UNIXTIME(1447430881,**

-> **'%Y %D %M %h:%i:%s %x');**

-> '2015 13th November 10:08:01 2015'

**Note**

If you use [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) and [**FROM\_UNIXTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-unixtime) to convert between values in a non-UTC time zone and Unix timestamp values, the conversion is lossy because the mapping is not one-to-one in both directions. For details, see the description of the [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) function.

**[GET\_FORMAT({DATE|TIME|DATETIME}, {'EUR'|'USA'|'JIS'|'ISO'|'INTERNAL'})](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-format)**

Returns a format string. This function is useful in combination with the [**DATE\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-format) and the [**STR\_TO\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_str-to-date) functions.

The possible values for the first and second arguments result in several possible format strings (for the specifiers used, see the table in the [**DATE\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-format) function description). ISO format refers to ISO 9075, not ISO 8601.

| **Function Call** | **Result** |
| --- | --- |
| [**GET\_FORMAT(DATE,'USA')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%m.%d.%Y'** |
| [**GET\_FORMAT(DATE,'JIS')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%Y-%m-%d'** |
| [**GET\_FORMAT(DATE,'ISO')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%Y-%m-%d'** |
| [**GET\_FORMAT(DATE,'EUR')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%d.%m.%Y'** |
| [**GET\_FORMAT(DATE,'INTERNAL')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%Y%m%d'** |
| [**GET\_FORMAT(DATETIME,'USA')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%Y-%m-%d %H.%i.%s'** |
| [**GET\_FORMAT(DATETIME,'JIS')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%Y-%m-%d %H:%i:%s'** |
| [**GET\_FORMAT(DATETIME,'ISO')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%Y-%m-%d %H:%i:%s'** |
| [**GET\_FORMAT(DATETIME,'EUR')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%Y-%m-%d %H.%i.%s'** |
| [**GET\_FORMAT(DATETIME,'INTERNAL')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%Y%m%d%H%i%s'** |
| [**GET\_FORMAT(TIME,'USA')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%h:%i:%s %p'** |
| [**GET\_FORMAT(TIME,'JIS')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%H:%i:%s'** |
| [**GET\_FORMAT(TIME,'ISO')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%H:%i:%s'** |
| [**GET\_FORMAT(TIME,'EUR')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%H.%i.%s'** |
| [**GET\_FORMAT(TIME,'INTERNAL')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) | **'%H%i%s'** |

[**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) can also be used as the first argument to [**GET\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format), in which case the function returns the same values as for [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime).

mysql> **SELECT DATE\_FORMAT('2003-10-03',GET\_FORMAT(DATE,'EUR'));**

-> '03.10.2003'

mysql> **SELECT STR\_TO\_DATE('10.31.2003',GET\_FORMAT(DATE,'USA'));**

-> '2003-10-31'

**[HOUR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_hour)*[time](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_hour)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_hour)**

Returns the hour for ***time***. The range of the return value is **0** to **23** for time-of-day values. However, the range of [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time) values actually is much larger, so **HOUR** can return values greater than **23**.

mysql> **SELECT HOUR('10:05:03');**

-> 10

mysql> **SELECT HOUR('272:59:59');**

-> 272

**[LAST\_DAY(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_last-day)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_last-day)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_last-day)**

Takes a date or datetime value and returns the corresponding value for the last day of the month. Returns **NULL** if the argument is invalid.

mysql> **SELECT LAST\_DAY('2003-02-05');**

-> '2003-02-28'

mysql> **SELECT LAST\_DAY('2004-02-05');**

-> '2004-02-29'

mysql> **SELECT LAST\_DAY('2004-01-01 01:01:01');**

-> '2004-01-31'

mysql> **SELECT LAST\_DAY('2003-03-32');**

-> NULL

**[LOCALTIME](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_localtime)**, [**LOCALTIME([*fsp*])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_localtime)

**[LOCALTIME](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_localtime)** and [**LOCALTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_localtime) are synonyms for [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now).

**[LOCALTIMESTAMP](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_localtimestamp)**, [**LOCALTIMESTAMP([*fsp*])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_localtimestamp)

**[LOCALTIMESTAMP](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_localtimestamp)** and [**LOCALTIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_localtimestamp) are synonyms for [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now).

**[MAKEDATE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_makedate)*[year](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_makedate)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_makedate)*[dayofyear](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_makedate)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_makedate)**

Returns a date, given year and day-of-year values. ***dayofyear*** must be greater than 0 or the result is **NULL**.

mysql> **SELECT MAKEDATE(2011,31), MAKEDATE(2011,32);**

-> '2011-01-31', '2011-02-01'

mysql> **SELECT MAKEDATE(2011,365), MAKEDATE(2014,365);**

-> '2011-12-31', '2014-12-31'

mysql> **SELECT MAKEDATE(2011,0);**

-> NULL

**[MAKETIME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_maketime)*[hour](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_maketime)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_maketime)*[minute](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_maketime)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_maketime)*[second](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_maketime)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_maketime)**

Returns a time value calculated from the ***hour***, ***minute***, and ***second*** arguments.

The ***second*** argument can have a fractional part.

mysql> **SELECT MAKETIME(12,15,30);**

-> '12:15:30'

**[MICROSECOND(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_microsecond)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_microsecond)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_microsecond)**

Returns the microseconds from the time or datetime expression ***expr*** as a number in the range from **0** to **999999**.

mysql> **SELECT MICROSECOND('12:00:00.123456');**

-> 123456

mysql> **SELECT MICROSECOND('2019-12-31 23:59:59.000010');**

-> 10

**[MINUTE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_minute)*[time](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_minute)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_minute)**

Returns the minute for ***time***, in the range **0** to **59**.

mysql> **SELECT MINUTE('2008-02-03 10:05:03');**

-> 5

**[MONTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_month)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_month)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_month)**

Returns the month for ***date***, in the range **1** to **12** for January to December, or **0** for dates such as **'0000-00-00'** or **'2008-00-00'** that have a zero month part.

mysql> **SELECT MONTH('2008-02-03');**

-> 2

**[MONTHNAME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_monthname)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_monthname)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_monthname)**

Returns the full name of the month for ***date***. The language used for the name is controlled by the value of the [**lc\_time\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_time_names) system variable ([Section 10.16, “MySQL Server Locale Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#locale-support)).

mysql> **SELECT MONTHNAME('2008-02-03');**

-> 'February'

**[NOW([](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_now)*[fsp](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_now)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_now)**

Returns the current date and time as a value in **'*YYYY-MM-DD hh:mm:ss*'** or ***YYYYMMDDhhmmss*** format, depending on whether the function is used in string or numeric context. The value is expressed in the session time zone.

If the ***fsp*** argument is given to specify a fractional seconds precision from 0 to 6, the return value includes a fractional seconds part of that many digits.

mysql> **SELECT NOW();**

-> '2007-12-15 23:50:26'

mysql> **SELECT NOW() + 0;**

-> 20071215235026.000000

[**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) returns a constant time that indicates the time at which the statement began to execute. (Within a stored function or trigger, [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) returns the time at which the function or triggering statement began to execute.) This differs from the behavior for [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate), which returns the exact time at which it executes.

mysql> **SELECT NOW(), SLEEP(2), NOW();**

+---------------------+----------+---------------------+

| NOW() | SLEEP(2) | NOW() |

+---------------------+----------+---------------------+

| 2006-04-12 13:47:36 | 0 | 2006-04-12 13:47:36 |

+---------------------+----------+---------------------+

mysql> **SELECT SYSDATE(), SLEEP(2), SYSDATE();**

+---------------------+----------+---------------------+

| SYSDATE() | SLEEP(2) | SYSDATE() |

+---------------------+----------+---------------------+

| 2006-04-12 13:47:44 | 0 | 2006-04-12 13:47:46 |

+---------------------+----------+---------------------+

In addition, the **SET TIMESTAMP** statement affects the value returned by [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) but not by [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate). This means that timestamp settings in the binary log have no effect on invocations of [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate). Setting the timestamp to a nonzero value causes each subsequent invocation of [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) to return that value. Setting the timestamp to zero cancels this effect so that [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) once again returns the current date and time.

See the description for [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate) for additional information about the differences between the two functions.

**[PERIOD\_ADD(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_period-add)*[P](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_period-add)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_period-add)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_period-add)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_period-add)**

Adds ***N*** months to period ***P*** (in the format ***YYMM*** or ***YYYYMM***). Returns a value in the format ***YYYYMM***.

**Note**

The period argument ***P*** is not a date value.

mysql> **SELECT PERIOD\_ADD(200801,2);**

-> 200803

**[PERIOD\_DIFF(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_period-diff)*[P1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_period-diff)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_period-diff)*[P2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_period-diff)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_period-diff)**

Returns the number of months between periods ***P1*** and ***P2***. ***P1*** and ***P2*** should be in the format ***YYMM*** or ***YYYYMM***. Note that the period arguments ***P1*** and ***P2*** are not date values.

mysql> **SELECT PERIOD\_DIFF(200802,200703);**

-> 11

**[QUARTER(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_quarter)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_quarter)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_quarter)**

Returns the quarter of the year for ***date***, in the range **1** to **4**.

mysql> **SELECT QUARTER('2008-04-01');**

-> 2

**[SECOND(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_second)*[time](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_second)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_second)**

Returns the second for ***time***, in the range **0** to **59**.

mysql> **SELECT SECOND('10:05:03');**

-> 3

**[SEC\_TO\_TIME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sec-to-time)*[seconds](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sec-to-time)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sec-to-time)**

Returns the ***seconds*** argument, converted to hours, minutes, and seconds, as a [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time) value. The range of the result is constrained to that of the [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time) data type. A warning occurs if the argument corresponds to a value outside that range.

mysql> **SELECT SEC\_TO\_TIME(2378);**

-> '00:39:38'

mysql> **SELECT SEC\_TO\_TIME(2378) + 0;**

-> 3938

**[STR\_TO\_DATE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_str-to-date)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_str-to-date)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_str-to-date)*[format](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_str-to-date)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_str-to-date)**

This is the inverse of the [**DATE\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-format) function. It takes a string ***str*** and a format string ***format***. [**STR\_TO\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_str-to-date) returns a [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value if the format string contains both date and time parts, or a [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) or [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time) value if the string contains only date or time parts. If the date, time, or datetime value extracted from ***str*** is illegal, [**STR\_TO\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_str-to-date) returns **NULL** and produces a warning.

The server scans ***str*** attempting to match ***format*** to it. The format string can contain literal characters and format specifiers beginning with **%**. Literal characters in ***format*** must match literally in ***str***. Format specifiers in ***format*** must match a date or time part in ***str***. For the specifiers that can be used in ***format***, see the [**DATE\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-format) function description.

mysql> **SELECT STR\_TO\_DATE('01,5,2013','%d,%m,%Y');**

-> '2013-05-01'

mysql> **SELECT STR\_TO\_DATE('May 1, 2013','%M %d,%Y');**

-> '2013-05-01'

Scanning starts at the beginning of ***str*** and fails if ***format*** is found not to match. Extra characters at the end of ***str*** are ignored.

mysql> **SELECT STR\_TO\_DATE('a09:30:17','a%h:%i:%s');**

-> '09:30:17'

mysql> **SELECT STR\_TO\_DATE('a09:30:17','%h:%i:%s');**

-> NULL

mysql> **SELECT STR\_TO\_DATE('09:30:17a','%h:%i:%s');**

-> '09:30:17'

Unspecified date or time parts have a value of 0, so incompletely specified values in ***str*** produce a result with some or all parts set to 0:

mysql> **SELECT STR\_TO\_DATE('abc','abc');**

-> '0000-00-00'

mysql> **SELECT STR\_TO\_DATE('9','%m');**

-> '0000-09-00'

mysql> **SELECT STR\_TO\_DATE('9','%s');**

-> '00:00:09'

Range checking on the parts of date values is as described in [Section 11.2.2, “The DATE, DATETIME, and TIMESTAMP Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime). This means, for example, that “zero” dates or dates with part values of 0 are permitted unless the SQL mode is set to disallow such values.

mysql> **SELECT STR\_TO\_DATE('00/00/0000', '%m/%d/%Y');**

-> '0000-00-00'

mysql> **SELECT STR\_TO\_DATE('04/31/2004', '%m/%d/%Y');**

-> '2004-04-31'

If the [**NO\_ZERO\_DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_zero_date) SQL mode is enabled, zero dates are disallowed. In that case, [**STR\_TO\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_str-to-date) returns **NULL** and generates a warning:

mysql> **SET sql\_mode = '';**

mysql> **SELECT STR\_TO\_DATE('00/00/0000', '%m/%d/%Y');**

+---------------------------------------+

| STR\_TO\_DATE('00/00/0000', '%m/%d/%Y') |

+---------------------------------------+

| 0000-00-00 |

+---------------------------------------+

mysql> **SET sql\_mode = 'NO\_ZERO\_DATE';**

mysql> **SELECT STR\_TO\_DATE('00/00/0000', '%m/%d/%Y');**

+---------------------------------------+

| STR\_TO\_DATE('00/00/0000', '%m/%d/%Y') |

+---------------------------------------+

| NULL |

+---------------------------------------+

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Warning

Code: 1411

Message: Incorrect datetime value: '00/00/0000' for function str\_to\_date

**Note**

You cannot use format **"%X%V"** to convert a year-week string to a date because the combination of a year and week does not uniquely identify a year and month if the week crosses a month boundary. To convert a year-week to a date, you should also specify the weekday:

mysql> **SELECT STR\_TO\_DATE('200442 Monday', '%X%V %W');**

-> '2004-10-18'

**[SUBDATE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subdate)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subdate)*[,INTERVAL](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subdate)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subdate)**[unit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subdate)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subdate)**, [**SUBDATE(*expr*,*days*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_subdate)

When invoked with the **INTERVAL** form of the second argument, [**SUBDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_subdate) is a synonym for [**DATE\_SUB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-sub). For information on the **INTERVAL** ***unit*** argument, see the discussion for [**DATE\_ADD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-add).

mysql> **SELECT DATE\_SUB('2008-01-02', INTERVAL 31 DAY);**

-> '2007-12-02'

mysql> **SELECT SUBDATE('2008-01-02', INTERVAL 31 DAY);**

-> '2007-12-02'

The second form enables the use of an integer value for ***days***. In such cases, it is interpreted as the number of days to be subtracted from the date or datetime expression ***expr***.

mysql> **SELECT SUBDATE('2008-01-02 12:00:00', 31);**

-> '2007-12-02 12:00:00'

**[SUBTIME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subtime)*[expr1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subtime)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subtime)*[expr2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subtime)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subtime)**

**[SUBTIME()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_subtime)** returns ***expr1*** − ***expr2*** expressed as a value in the same format as ***expr1***. ***expr1*** is a time or datetime expression, and ***expr2*** is a time expression.

mysql> **SELECT SUBTIME('2007-12-31 23:59:59.999999','1 1:1:1.000002');**

-> '2007-12-30 22:58:58.999997'

mysql> **SELECT SUBTIME('01:00:00.999999', '02:00:00.999998');**

-> '-00:59:59.999999'

**[SYSDATE([](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sysdate)*[fsp](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sysdate)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sysdate)**

Returns the current date and time as a value in **'*YYYY-MM-DD hh:mm:ss*'** or ***YYYYMMDDhhmmss*** format, depending on whether the function is used in string or numeric context.

If the ***fsp*** argument is given to specify a fractional seconds precision from 0 to 6, the return value includes a fractional seconds part of that many digits.

[**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate) returns the time at which it executes. This differs from the behavior for [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now), which returns a constant time that indicates the time at which the statement began to execute. (Within a stored function or trigger, [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) returns the time at which the function or triggering statement began to execute.)

mysql> **SELECT NOW(), SLEEP(2), NOW();**

+---------------------+----------+---------------------+

| NOW() | SLEEP(2) | NOW() |

+---------------------+----------+---------------------+

| 2006-04-12 13:47:36 | 0 | 2006-04-12 13:47:36 |

+---------------------+----------+---------------------+

mysql> **SELECT SYSDATE(), SLEEP(2), SYSDATE();**

+---------------------+----------+---------------------+

| SYSDATE() | SLEEP(2) | SYSDATE() |

+---------------------+----------+---------------------+

| 2006-04-12 13:47:44 | 0 | 2006-04-12 13:47:46 |

+---------------------+----------+---------------------+

In addition, the **SET TIMESTAMP** statement affects the value returned by [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) but not by [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate). This means that timestamp settings in the binary log have no effect on invocations of [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate).

Because [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate) can return different values even within the same statement, and is not affected by **SET TIMESTAMP**, it is nondeterministic and therefore unsafe for replication if statement-based binary logging is used. If that is a problem, you can use row-based logging.

Alternatively, you can use the [--sysdate-is-now](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_sysdate-is-now) option to cause [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate) to be an alias for [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now). This works if the option is used on both the replication source server and the replica.

The nondeterministic nature of [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate) also means that indexes cannot be used for evaluating expressions that refer to it.

**[TIME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_time)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_time)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_time)**

Extracts the time part of the time or datetime expression ***expr*** and returns it as a string.

This function is unsafe for statement-based replication. A warning is logged if you use this function when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**.

mysql> **SELECT TIME('2003-12-31 01:02:03');**

-> '01:02:03'

mysql> **SELECT TIME('2003-12-31 01:02:03.000123');**

-> '01:02:03.000123'

**[TIMEDIFF(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timediff)*[expr1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timediff)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timediff)*[expr2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timediff)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timediff)**

**[TIMEDIFF()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timediff)** returns ***expr1*** − ***expr2*** expressed as a time value. ***expr1*** and ***expr2*** are time or date-and-time expressions, but both must be of the same type.

The result returned by **TIMEDIFF()** is limited to the range allowed for [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time) values. Alternatively, you can use either of the functions [**TIMESTAMPDIFF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timestampdiff) and [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp), both of which return integers.

mysql> **SELECT TIMEDIFF('2000:01:01 00:00:00',**

-> **'2000:01:01 00:00:00.000001');**

-> '-00:00:00.000001'

mysql> **SELECT TIMEDIFF('2008-12-31 23:59:59.000001',**

->  **'2008-12-30 01:01:01.000002');**

-> '46:58:57.999999'

**[TIMESTAMP(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestamp)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestamp)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestamp)**, [**TIMESTAMP(*expr1*,*expr2*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timestamp)

With a single argument, this function returns the date or datetime expression ***expr*** as a datetime value. With two arguments, it adds the time expression ***expr2*** to the date or datetime expression ***expr1*** and returns the result as a datetime value.

mysql> **SELECT TIMESTAMP('2003-12-31');**

-> '2003-12-31 00:00:00'

mysql> **SELECT TIMESTAMP('2003-12-31 12:00:00','12:00:00');**

-> '2004-01-01 00:00:00'

**[TIMESTAMPADD(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampadd)*[unit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampadd)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampadd)*[interval](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampadd)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampadd)*[datetime\_expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampadd)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampadd)**

Adds the integer expression ***interval*** to the date or datetime expression ***datetime\_expr***. The unit for ***interval*** is given by the ***unit*** argument, which should be one of the following values: **MICROSECOND** (microseconds), **SECOND**, **MINUTE**, **HOUR**, **DAY**, **WEEK**, **MONTH**, **QUARTER**, or **YEAR**.

The ***unit*** value may be specified using one of keywords as shown, or with a prefix of **SQL\_TSI\_**. For example, **DAY** and **SQL\_TSI\_DAY** both are legal.

mysql> **SELECT TIMESTAMPADD(MINUTE,1,'2003-01-02');**

-> '2003-01-02 00:01:00'

mysql> **SELECT TIMESTAMPADD(WEEK,1,'2003-01-02');**

-> '2003-01-09'

**[TIMESTAMPDIFF(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampdiff)*[unit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampdiff)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampdiff)*[datetime\_expr1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampdiff)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampdiff)*[datetime\_expr2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampdiff)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_timestampdiff)**

Returns ***datetime\_expr2*** − ***datetime\_expr1***, where ***datetime\_expr1*** and ***datetime\_expr2*** are date or datetime expressions. One expression may be a date and the other a datetime; a date value is treated as a datetime having the time part **'00:00:00'** where necessary. The unit for the result (an integer) is given by the ***unit*** argument. The legal values for ***unit*** are the same as those listed in the description of the [**TIMESTAMPADD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timestampadd) function.

mysql> **SELECT TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01');**

-> 3

mysql> **SELECT TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01');**

-> -1

mysql> **SELECT TIMESTAMPDIFF(MINUTE,'2003-02-01','2003-05-01 12:05:55');**

-> 128885

**Note**

The order of the date or datetime arguments for this function is the opposite of that used with the [**TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_timestamp) function when invoked with 2 arguments.

**[TIME\_FORMAT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_time-format)*[time](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_time-format)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_time-format)*[format](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_time-format)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_time-format)**

This is used like the [**DATE\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-format) function, but the ***format*** string may contain format specifiers only for hours, minutes, seconds, and microseconds. Other specifiers produce a **NULL** value or **0**.

If the ***time*** value contains an hour part that is greater than **23**, the **%H** and **%k** hour format specifiers produce a value larger than the usual range of **0..23**. The other hour format specifiers produce the hour value modulo 12.

mysql> **SELECT TIME\_FORMAT('100:00:00', '%H %k %h %I %l');**

-> '100 100 04 04 4'

**[TIME\_TO\_SEC(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_time-to-sec)*[time](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_time-to-sec)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_time-to-sec)**

Returns the ***time*** argument, converted to seconds.

mysql> **SELECT TIME\_TO\_SEC('22:23:00');**

-> 80580

mysql> **SELECT TIME\_TO\_SEC('00:39:38');**

-> 2378

**[TO\_DAYS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_to-days)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_to-days)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_to-days)**

Given a date ***date***, returns a day number (the number of days since year 0).

mysql> **SELECT TO\_DAYS(950501);**

-> 728779

mysql> **SELECT TO\_DAYS('2007-10-07');**

-> 733321

[**TO\_DAYS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-days) is not intended for use with values that precede the advent of the Gregorian calendar (1582), because it does not take into account the days that were lost when the calendar was changed. For dates before 1582 (and possibly a later year in other locales), results from this function are not reliable. See [Section 12.9, “What Calendar Is Used By MySQL?”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#mysql-calendar), for details.

Remember that MySQL converts two-digit year values in dates to four-digit form using the rules in [Section 11.2, “Date and Time Data Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#date-and-time-types). For example, **'2008-10-07'** and **'08-10-07'** are seen as identical dates:

mysql> **SELECT TO\_DAYS('2008-10-07'), TO\_DAYS('08-10-07');**

-> 733687, 733687

In MySQL, the zero date is defined as **'0000-00-00'**, even though this date is itself considered invalid. This means that, for **'0000-00-00'** and **'0000-01-01'**, [**TO\_DAYS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-days) returns the values shown here:

mysql> **SELECT TO\_DAYS('0000-00-00');**

+-----------------------+

| to\_days('0000-00-00') |

+-----------------------+

| NULL |

+-----------------------+

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS;**

+---------+------+----------------------------------------+

| Level | Code | Message |

+---------+------+----------------------------------------+

| Warning | 1292 | Incorrect datetime value: '0000-00-00' |

+---------+------+----------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT TO\_DAYS('0000-01-01');**

+-----------------------+

| to\_days('0000-01-01') |

+-----------------------+

| 1 |

+-----------------------+

1 row in set (0.00 sec)

This is true whether or not the [**ALLOW\_INVALID\_DATES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_allow_invalid_dates) SQL server mode is enabled.

**[TO\_SECONDS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_to-seconds)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_to-seconds)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_to-seconds)**

Given a date or datetime ***expr***, returns the number of seconds since the year 0. If ***expr*** is not a valid date or datetime value, returns **NULL**.

mysql> **SELECT TO\_SECONDS(950501);**

-> 62966505600

mysql> **SELECT TO\_SECONDS('2009-11-29');**

-> 63426672000

mysql> **SELECT TO\_SECONDS('2009-11-29 13:43:32');**

-> 63426721412

mysql> **SELECT TO\_SECONDS( NOW() );**

-> 63426721458

Like [**TO\_DAYS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-days), **TO\_SECONDS()** is not intended for use with values that precede the advent of the Gregorian calendar (1582), because it does not take into account the days that were lost when the calendar was changed. For dates before 1582 (and possibly a later year in other locales), results from this function are not reliable. See [Section 12.9, “What Calendar Is Used By MySQL?”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#mysql-calendar), for details.

Like [**TO\_DAYS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-days), **TO\_SECONDS()**, converts two-digit year values in dates to four-digit form using the rules in [Section 11.2, “Date and Time Data Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#date-and-time-types).

In MySQL, the zero date is defined as **'0000-00-00'**, even though this date is itself considered invalid. This means that, for **'0000-00-00'** and **'0000-01-01'**, [**TO\_SECONDS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-seconds) returns the values shown here:

mysql> **SELECT TO\_SECONDS('0000-00-00');**

+--------------------------+

| TO\_SECONDS('0000-00-00') |

+--------------------------+

| NULL |

+--------------------------+

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS;**

+---------+------+----------------------------------------+

| Level | Code | Message |

+---------+------+----------------------------------------+

| Warning | 1292 | Incorrect datetime value: '0000-00-00' |

+---------+------+----------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT TO\_SECONDS('0000-01-01');**

+--------------------------+

| TO\_SECONDS('0000-01-01') |

+--------------------------+

| 86400 |

+--------------------------+

1 row in set (0.00 sec)

This is true whether or not the [**ALLOW\_INVALID\_DATES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_allow_invalid_dates) SQL server mode is enabled.

**[UNIX\_TIMESTAMP([](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_unix-timestamp)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_unix-timestamp)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_unix-timestamp)**

If [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) is called with no ***date*** argument, it returns a Unix timestamp representing seconds since **'1970-01-01 00:00:00'** UTC.

If [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) is called with a ***date*** argument, it returns the value of the argument as seconds since **'1970-01-01 00:00:00'** UTC. The server interprets ***date*** as a value in the session time zone and converts it to an internal Unix timestamp value in UTC. (Clients can set the session time zone as described in [Section 5.1.15, “MySQL Server Time Zone Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#time-zone-support).) The ***date*** argument may be a [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), or [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) string, or a number in ***YYMMDD***, ***YYMMDDhhmmss***, ***YYYYMMDD***, or ***YYYYMMDDhhmmss*** format. If the argument includes a time part, it may optionally include a fractional seconds part.

The return value is an integer if no argument is given or the argument does not include a fractional seconds part, or [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) if an argument is given that includes a fractional seconds part.

When the ***date*** argument is a [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) column, [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) returns the internal timestamp value directly, with no implicit “string-to-Unix-timestamp” conversion.

The valid range of argument values is the same as for the [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) data type: **'1970-01-01 00:00:01.000000'** UTC to **'2038-01-19 03:14:07.999999'** UTC. If you pass an out-of-range date to [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp), it returns **0**.

mysql> **SELECT UNIX\_TIMESTAMP();**

-> 1447431666

mysql> **SELECT UNIX\_TIMESTAMP('2015-11-13 10:20:19');**

-> 1447431619

mysql> **SELECT UNIX\_TIMESTAMP('2015-11-13 10:20:19.012');**

-> 1447431619.012

If you use [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) and [**FROM\_UNIXTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-unixtime) to convert between values in a non-UTC time zone and Unix timestamp values, the conversion is lossy because the mapping is not one-to-one in both directions. For example, due to conventions for local time zone changes such as Daylight Saving Time (DST), it is possible for [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) to map two values that are distinct in a non-UTC time zone to the same Unix timestamp value. [**FROM\_UNIXTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-unixtime) maps that value back to only one of the original values. Here is an example, using values that are distinct in the **MET** time zone:

mysql> **SET time\_zone = 'MET';**

mysql> **SELECT UNIX\_TIMESTAMP('2005-03-27 03:00:00');**

+---------------------------------------+

| UNIX\_TIMESTAMP('2005-03-27 03:00:00') |

+---------------------------------------+

| 1111885200 |

+---------------------------------------+

mysql> **SELECT UNIX\_TIMESTAMP('2005-03-27 02:00:00');**

+---------------------------------------+

| UNIX\_TIMESTAMP('2005-03-27 02:00:00') |

+---------------------------------------+

| 1111885200 |

+---------------------------------------+

mysql> **SELECT FROM\_UNIXTIME(1111885200);**

+---------------------------+

| FROM\_UNIXTIME(1111885200) |

+---------------------------+

| 2005-03-27 03:00:00 |

+---------------------------+

**Note**

To use named time zones such as **'MET'** or **'Europe/Amsterdam'**, the time zone tables must be properly set up. For instructions, see [Section 5.1.15, “MySQL Server Time Zone Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#time-zone-support).

If you want to subtract [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp) columns, you might want to cast them to signed integers. See [Section 12.11, “Cast Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#cast-functions).

**[UTC\_DATE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_utc-date)**, [**UTC\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-date)

Returns the current UTC date as a value in **'*YYYY-MM-DD*'** or ***YYYYMMDD*** format, depending on whether the function is used in string or numeric context.

mysql> **SELECT UTC\_DATE(), UTC\_DATE() + 0;**

-> '2003-08-14', 20030814

**[UTC\_TIME](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_utc-time)**, [**UTC\_TIME([*fsp*])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-time)

Returns the current UTC time as a value in ***'hh:mm:ss'*** or ***hhmmss*** format, depending on whether the function is used in string or numeric context.

If the ***fsp*** argument is given to specify a fractional seconds precision from 0 to 6, the return value includes a fractional seconds part of that many digits.

mysql> **SELECT UTC\_TIME(), UTC\_TIME() + 0;**

-> '18:07:53', 180753.000000

**[UTC\_TIMESTAMP](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_utc-timestamp)**, [**UTC\_TIMESTAMP([*fsp*])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-timestamp)

Returns the current UTC date and time as a value in **'*YYYY-MM-DD hh:mm:ss*'** or ***YYYYMMDDhhmmss*** format, depending on whether the function is used in string or numeric context.

If the ***fsp*** argument is given to specify a fractional seconds precision from 0 to 6, the return value includes a fractional seconds part of that many digits.

mysql> **SELECT UTC\_TIMESTAMP(), UTC\_TIMESTAMP() + 0;**

-> '2003-08-14 18:08:04', 20030814180804.000000

**[WEEK(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_week)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_week)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_week)*[mode](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_week)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_week)**

This function returns the week number for ***date***. The two-argument form of [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) enables you to specify whether the week starts on Sunday or Monday and whether the return value should be in the range from **0** to **53** or from **1** to **53**. If the ***mode*** argument is omitted, the value of the [**default\_week\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_week_format) system variable is used. See [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

The following table describes how the ***mode*** argument works.

| **Mode** | **First day of week** | **Range** | **Week 1 is the first week …** |
| --- | --- | --- | --- |
| **0** | Sunday | 0-53 | with a Sunday in this year |
| **1** | Monday | 0-53 | with 4 or more days this year |
| **2** | Sunday | 1-53 | with a Sunday in this year |
| **3** | Monday | 1-53 | with 4 or more days this year |
| **4** | Sunday | 0-53 | with 4 or more days this year |
| **5** | Monday | 0-53 | with a Monday in this year |
| **6** | Sunday | 1-53 | with 4 or more days this year |
| **7** | Monday | 1-53 | with a Monday in this year |

For ***mode*** values with a meaning of “with 4 or more days this year,” weeks are numbered according to ISO 8601:1988:

If the week containing January 1 has 4 or more days in the new year, it is week 1.

Otherwise, it is the last week of the previous year, and the next week is week 1.

mysql> **SELECT WEEK('2008-02-20');**

-> 7

mysql> **SELECT WEEK('2008-02-20',0);**

-> 7

mysql> **SELECT WEEK('2008-02-20',1);**

-> 8

mysql> **SELECT WEEK('2008-12-31',1);**

-> 53

If a date falls in the last week of the previous year, MySQL returns **0** if you do not use **2**, **3**, **6**, or **7** as the optional ***mode*** argument:

mysql> **SELECT YEAR('2000-01-01'), WEEK('2000-01-01',0);**

-> 2000, 0

One might argue that [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) should return **52** because the given date actually occurs in the 52nd week of 1999. [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) returns **0** instead so that the return value is “the week number in the given year.” This makes use of the [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) function reliable when combined with other functions that extract a date part from a date.

If you prefer a result evaluated with respect to the year that contains the first day of the week for the given date, use **0**, **2**, **5**, or **7** as the optional ***mode*** argument.

mysql> **SELECT WEEK('2000-01-01',2);**

-> 52

Alternatively, use the [**YEARWEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_yearweek) function:

mysql> **SELECT YEARWEEK('2000-01-01');**

-> 199952

mysql> **SELECT MID(YEARWEEK('2000-01-01'),5,2);**

-> '52'

**[WEEKDAY(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weekday)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weekday)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weekday)**

Returns the weekday index for ***date*** (**0** = Monday, **1** = Tuesday, … **6** = Sunday).

mysql> **SELECT WEEKDAY('2008-02-03 22:23:00');**

-> 6

mysql> **SELECT WEEKDAY('2007-11-06');**

-> 1

**[WEEKOFYEAR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weekofyear)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weekofyear)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weekofyear)**

Returns the calendar week of the date as a number in the range from **1** to **53**. [**WEEKOFYEAR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weekofyear) is a compatibility function that is equivalent to [**WEEK(*date*,3)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week).

mysql> **SELECT WEEKOFYEAR('2008-02-20');**

-> 8

**[YEAR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_year)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_year)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_year)**

Returns the year for ***date***, in the range **1000** to **9999**, or **0** for the “zero” date.

mysql> **SELECT YEAR('1987-01-01');**

-> 1987

**[YEARWEEK(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_yearweek)*[date](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_yearweek)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_yearweek)**, [**YEARWEEK(*date*,*mode*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_yearweek)

Returns year and week for a date. The year in the result may be different from the year in the date argument for the first and the last week of the year.

The ***mode*** argument works exactly like the ***mode*** argument to [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week). For the single-argument syntax, a ***mode*** value of 0 is used. Unlike [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week), the value of [**default\_week\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_week_format) does not influence [**YEARWEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_yearweek).

mysql> **SELECT YEARWEEK('1987-01-01');**

-> 198652

The week number is different from what the [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) function would return (**0**) for optional arguments **0** or **1**, as [**WEEK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_week) then returns the week in the context of the given year.

## 12.8 String Functions and Operators

[12.8.1 String Comparison Functions and Operators](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#string-comparison-functions)

[12.8.2 Regular Expressions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#regexp)

[12.8.3 Character Set and Collation of Function Results](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#string-functions-charset)

**Table 12.12 String Functions and Operators**

|  |  |
| --- | --- |
| Name | Description |
| [ASCII()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ascii) | Return numeric value of left-most character |
| [BIN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bin) | Return a string containing binary representation of a number |
| [BIT\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-length) | Return length of argument in bits |
| [CHAR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char) | Return the character for each integer passed |
| [CHAR\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char-length) | Return number of characters in argument |
| [CHARACTER\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_character-length) | Synonym for CHAR\_LENGTH() |
| [CONCAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat) | Return concatenated string |
| [CONCAT\_WS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat-ws) | Return concatenate with separator |
| [ELT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_elt) | Return string at index number |
| [EXPORT\_SET()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_export-set) | Return a string such that for every bit set in the value bits, you get an on string and for every unset bit, you get an off string |
| [FIELD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_field) | Index (position) of first argument in subsequent arguments |
| [FIND\_IN\_SET()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_find-in-set) | Index (position) of first argument within second argument |
| [FORMAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format) | Return a number formatted to specified number of decimal places |
| [FROM\_BASE64()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-base64) | Decode base64 encoded string and return result |
| [HEX()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex) | Hexadecimal representation of decimal or string value |
| [INSERT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_insert) | Insert substring at specified position up to specified number of characters |
| [INSTR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_instr) | Return the index of the first occurrence of substring |
| [LCASE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lcase) | Synonym for LOWER() |
| [LEFT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_left) | Return the leftmost number of characters as specified |
| [LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_length) | Return the length of a string in bytes |
| [LIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) | Simple pattern matching |
| [LOAD\_FILE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_load-file) | Load the named file |
| [LOCATE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_locate) | Return the position of the first occurrence of substring |
| [LOWER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lower) | Return the argument in lowercase |
| [LPAD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lpad) | Return the string argument, left-padded with the specified string |
| [LTRIM()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ltrim) | Remove leading spaces |
| [MAKE\_SET()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_make-set) | Return a set of comma-separated strings that have the corresponding bit in bits set |
| [MATCH](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) | Perform full-text search |
| [MID()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mid) | Return a substring starting from the specified position |
| [NOT LIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-like) | Negation of simple pattern matching |
| [NOT REGEXP](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-regexp) | Negation of REGEXP |
| [OCT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_oct) | Return a string containing octal representation of a number |
| [OCTET\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_octet-length) | Synonym for LENGTH() |
| [ORD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ord) | Return character code for leftmost character of the argument |
| [POSITION()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_position) | Synonym for LOCATE() |
| [QUOTE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_quote) | Escape the argument for use in an SQL statement |
| [REGEXP](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) | Whether string matches regular expression |
| [REGEXP\_INSTR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-instr) | Starting index of substring matching regular expression |
| [REGEXP\_LIKE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like) | Whether string matches regular expression |
| [REGEXP\_REPLACE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-replace) | Replace substrings matching regular expression |
| [REGEXP\_SUBSTR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-substr) | Return substring matching regular expression |
| [REPEAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_repeat) | Repeat a string the specified number of times |
| [REPLACE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_replace) | Replace occurrences of a specified string |
| [REVERSE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_reverse) | Reverse the characters in a string |
| [RIGHT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_right) | Return the specified rightmost number of characters |
| [RLIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) | Whether string matches regular expression |
| [RPAD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rpad) | Append string the specified number of times |
| [RTRIM()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rtrim) | Remove trailing spaces |
| [SOUNDEX()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_soundex) | Return a soundex string |
| [SOUNDS LIKE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_sounds-like) | Compare sounds |
| [SPACE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_space) | Return a string of the specified number of spaces |
| [STRCMP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_strcmp) | Compare two strings |
| [SUBSTR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substr) | Return the substring as specified |
| [SUBSTRING()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring) | Return the substring as specified |
| [SUBSTRING\_INDEX()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring-index) | Return a substring from a string before the specified number of occurrences of the delimiter |
| [TO\_BASE64()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-base64) | Return the argument converted to a base-64 string |
| [TRIM()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_trim) | Remove leading and trailing spaces |
| [UCASE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ucase) | Synonym for UPPER() |
| [UNHEX()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unhex) | Return a string containing hex representation of a number |
| [UPPER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper) | Convert to uppercase |
| [WEIGHT\_STRING()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string) | Return the weight string for a string |

String-valued functions return **NULL** if the length of the result would be greater than the value of the [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) system variable. See [Section 5.1.1, “Configuring the Server”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-configuration).

For functions that operate on string positions, the first position is numbered 1.

For functions that take length arguments, noninteger arguments are rounded to the nearest integer.

**[ASCII(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ascii)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ascii)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ascii)**

Returns the numeric value of the leftmost character of the string ***str***. Returns **0** if ***str*** is the empty string. Returns **NULL** if ***str*** is **NULL**. [**ASCII()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ascii) works for 8-bit characters.

mysql> **SELECT ASCII('2');**

-> 50

mysql> **SELECT ASCII(2);**

-> 50

mysql> **SELECT ASCII('dx');**

-> 100

See also the [**ORD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ord) function.

**[BIN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bin)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bin)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bin)**

Returns a string representation of the binary value of ***N***, where ***N*** is a longlong ([**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types)) number. This is equivalent to [**CONV(*N*,10,2)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_conv). Returns **NULL** if ***N*** is **NULL**.

mysql> **SELECT BIN(12);**

-> '1100'

**[BIT\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-length)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-length)**

Returns the length of the string ***str*** in bits.

mysql> **SELECT BIT\_LENGTH('text');**

-> 32

**[CHAR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_char)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_char)*[,... [USING](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_char)*[charset\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_char)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_char)**

**[CHAR()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_char)** interprets each argument ***N*** as an integer and returns a string consisting of the characters given by the code values of those integers. **NULL** values are skipped.

mysql> **SELECT CHAR(77,121,83,81,'76');**

-> 'MySQL'

mysql> **SELECT CHAR(77,77.3,'77.3');**

-> 'MMM'

[**CHAR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char) arguments larger than 255 are converted into multiple result bytes. For example, [**CHAR(256)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char) is equivalent to [**CHAR(1,0)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char), and [**CHAR(256\*256)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char) is equivalent to [**CHAR(1,0,0)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char):

mysql> **SELECT HEX(CHAR(1,0)), HEX(CHAR(256));**

+----------------+----------------+

| HEX(CHAR(1,0)) | HEX(CHAR(256)) |

+----------------+----------------+

| 0100 | 0100 |

+----------------+----------------+

mysql> **SELECT HEX(CHAR(1,0,0)), HEX(CHAR(256\*256));**

+------------------+--------------------+

| HEX(CHAR(1,0,0)) | HEX(CHAR(256\*256)) |

+------------------+--------------------+

| 010000 | 010000 |

+------------------+--------------------+

By default, [**CHAR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char) returns a binary string. To produce a string in a given character set, use the optional **USING** clause:

mysql> **SELECT CHARSET(CHAR(X'65')), CHARSET(CHAR(X'65' USING utf8));**

+----------------------+---------------------------------+

| CHARSET(CHAR(X'65')) | CHARSET(CHAR(X'65' USING utf8)) |

+----------------------+---------------------------------+

| binary | utf8 |

+----------------------+---------------------------------+

If **USING** is given and the result string is illegal for the given character set, a warning is issued. Also, if strict SQL mode is enabled, the result from [**CHAR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char) becomes **NULL**.

**[CHAR\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_char-length)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_char-length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_char-length)**

Returns the length of the string ***str***, measured in characters. A multibyte character counts as a single character. This means that for a string containing five 2-byte characters, [**LENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_length) returns **10**, whereas [**CHAR\_LENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char-length) returns **5**.

**[CHARACTER\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_character-length)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_character-length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_character-length)**

**[CHARACTER\_LENGTH()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_character-length)** is a synonym for [**CHAR\_LENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char-length).

**[CONCAT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat)*[str1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat)*[str2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat)*[,...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat)**

Returns the string that results from concatenating the arguments. May have one or more arguments. If all arguments are nonbinary strings, the result is a nonbinary string. If the arguments include any binary strings, the result is a binary string. A numeric argument is converted to its equivalent nonbinary string form.

[**CONCAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat) returns **NULL** if any argument is **NULL**.

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

-> 'MySQL'

mysql> **SELECT CONCAT('My', NULL, 'QL');**

-> NULL

mysql> **SELECT CONCAT(14.3);**

-> '14.3'

For quoted strings, concatenation can be performed by placing the strings next to each other:

mysql> **SELECT 'My' 'S' 'QL';**

-> 'MySQL'

**[CONCAT\_WS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat-ws)*[separator](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat-ws)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat-ws)*[str1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat-ws)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat-ws)*[str2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat-ws)*[,...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat-ws)**

**[CONCAT\_WS()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_concat-ws)** stands for Concatenate With Separator and is a special form of [**CONCAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat). The first argument is the separator for the rest of the arguments. The separator is added between the strings to be concatenated. The separator can be a string, as can the rest of the arguments. If the separator is **NULL**, the result is **NULL**.

mysql> **SELECT CONCAT\_WS(',','First name','Second name','Last Name');**

-> 'First name,Second name,Last Name'

mysql> **SELECT CONCAT\_WS(',','First name',NULL,'Last Name');**

-> 'First name,Last Name'

[**CONCAT\_WS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat-ws) does not skip empty strings. However, it does skip any **NULL** values after the separator argument.

**[ELT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_elt)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_elt)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_elt)*[str1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_elt)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_elt)*[str2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_elt)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_elt)*[str3](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_elt)*[,...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_elt)**

**[ELT()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_elt)** returns the ***N***th element of the list of strings: ***str1*** if ***N*** = **1**, ***str2*** if ***N*** = **2**, and so on. Returns **NULL** if ***N*** is less than **1** or greater than the number of arguments. [**ELT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_elt) is the complement of [**FIELD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_field).

mysql> **SELECT ELT(1, 'Aa', 'Bb', 'Cc', 'Dd');**

-> 'Aa'

mysql> **SELECT ELT(4, 'Aa', 'Bb', 'Cc', 'Dd');**

-> 'Dd'

**[EXPORT\_SET(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_export-set)*[bits](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_export-set)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_export-set)*[on](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_export-set)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_export-set)*[off](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_export-set)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_export-set)*[separator](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_export-set)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_export-set)*[number\_of\_bits](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_export-set)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_export-set)**

Returns a string such that for every bit set in the value ***bits***, you get an ***on*** string and for every bit not set in the value, you get an ***off*** string. Bits in ***bits*** are examined from right to left (from low-order to high-order bits). Strings are added to the result from left to right, separated by the ***separator*** string (the default being the comma character **,**). The number of bits examined is given by ***number\_of\_bits***, which has a default of 64 if not specified. ***number\_of\_bits*** is silently clipped to 64 if larger than 64. It is treated as an unsigned integer, so a value of −1 is effectively the same as 64.

mysql> **SELECT EXPORT\_SET(5,'Y','N',',',4);**

-> 'Y,N,Y,N'

mysql> **SELECT EXPORT\_SET(6,'1','0',',',10);**

-> '0,1,1,0,0,0,0,0,0,0'

**[FIELD(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_field)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_field)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_field)*[str1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_field)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_field)*[str2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_field)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_field)*[str3](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_field)*[,...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_field)**

Returns the index (position) of ***str*** in the ***str1***, ***str2***, ***str3***, **...** list. Returns **0** if ***str*** is not found.

If all arguments to [**FIELD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_field) are strings, all arguments are compared as strings. If all arguments are numbers, they are compared as numbers. Otherwise, the arguments are compared as double.

If ***str*** is **NULL**, the return value is **0** because **NULL** fails equality comparison with any value. [**FIELD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_field) is the complement of [**ELT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_elt).

mysql> **SELECT FIELD('Bb', 'Aa', 'Bb', 'Cc', 'Dd', 'Ff');**

-> 2

mysql> **SELECT FIELD('Gg', 'Aa', 'Bb', 'Cc', 'Dd', 'Ff');**

-> 0

**[FIND\_IN\_SET(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_find-in-set)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_find-in-set)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_find-in-set)*[strlist](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_find-in-set)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_find-in-set)**

Returns a value in the range of 1 to ***N*** if the string ***str*** is in the string list ***strlist*** consisting of ***N*** substrings. A string list is a string composed of substrings separated by **,** characters. If the first argument is a constant string and the second is a column of type [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set), the [**FIND\_IN\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_find-in-set) function is optimized to use bit arithmetic. Returns **0** if ***str*** is not in ***strlist*** or if ***strlist*** is the empty string. Returns **NULL** if either argument is **NULL**. This function does not work properly if the first argument contains a comma (**,**) character.

mysql> **SELECT FIND\_IN\_SET('b','a,b,c,d');**

-> 2

**[FORMAT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format)*[X](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format)*[D](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format)*[locale](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format)**

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

The optional third parameter enables a locale to be specified to be used for the result number's decimal point, thousands separator, and grouping between separators. Permissible locale values are the same as the legal values for the [**lc\_time\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_time_names) system variable (see [Section 10.16, “MySQL Server Locale Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#locale-support)). If no locale is specified, the default is **'en\_US'**.

mysql> **SELECT FORMAT(12332.123456, 4);**

-> '12,332.1235'

mysql> **SELECT FORMAT(12332.1,4);**

-> '12,332.1000'

mysql> **SELECT FORMAT(12332.2,0);**

-> '12,332'

mysql> **SELECT FORMAT(12332.2,2,'de\_DE');**

-> '12.332,20'

**[FROM\_BASE64(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_from-base64)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_from-base64)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_from-base64)**

Takes a string encoded with the base-64 encoded rules used by [**TO\_BASE64()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-base64) and returns the decoded result as a binary string. The result is **NULL** if the argument is **NULL** or not a valid base-64 string. See the description of [**TO\_BASE64()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-base64) for details about the encoding and decoding rules.

mysql> **SELECT TO\_BASE64('abc'), FROM\_BASE64(TO\_BASE64('abc'));**

-> 'JWJj', 'abc'

**[HEX(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_hex)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_hex)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_hex)**, [**HEX(*N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex)

For a string argument ***str***, [**HEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex) returns a hexadecimal string representation of ***str*** where each byte of each character in ***str*** is converted to two hexadecimal digits. (Multibyte characters therefore become more than two digits.) The inverse of this operation is performed by the [**UNHEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unhex) function.

For a numeric argument ***N***, [**HEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex) returns a hexadecimal string representation of the value of ***N*** treated as a longlong ([**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types)) number. This is equivalent to [**CONV(*N*,10,16)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_conv). The inverse of this operation is performed by [**CONV(HEX(*N*),16,10)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_conv).

mysql> **SELECT X'616263', HEX('abc'), UNHEX(HEX('abc'));**

-> 'abc', 616263, 'abc'

mysql> **SELECT HEX(255), CONV(HEX(255),16,10);**

-> 'FF', 255

**[INSERT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_insert)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_insert)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_insert)*[pos](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_insert)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_insert)*[len](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_insert)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_insert)*[newstr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_insert)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_insert)**

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

mysql> **SELECT INSERT('Quadratic', 3, 4, 'What');**

-> 'QuWhattic'

mysql> **SELECT INSERT('Quadratic', -1, 4, 'What');**

-> 'Quadratic'

mysql> **SELECT INSERT('Quadratic', 3, 100, 'What');**

-> 'QuWhat'

This function is multibyte safe.

**[INSTR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_instr)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_instr)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_instr)*[substr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_instr)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_instr)**

Returns the position of the first occurrence of substring ***substr*** in string ***str***. This is the same as the two-argument form of [**LOCATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_locate), except that the order of the arguments is reversed.

mysql> **SELECT INSTR('foobarbar', 'bar');**

-> 4

mysql> **SELECT INSTR('xbar', 'foobar');**

-> 0

This function is multibyte safe, and is case-sensitive only if at least one argument is a binary string.

**[LCASE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lcase)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lcase)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lcase)**

**[LCASE()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lcase)** is a synonym for [**LOWER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lower).

**LCASE()** used in a view is rewritten as **LOWER()** when storing the view's definition. (Bug #12844279)

**[LEFT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_left)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_left)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_left)*[len](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_left)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_left)**

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

mysql> **SELECT LEFT('foobarbar', 5);**

-> 'fooba'

This function is multibyte safe.

**[LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_length)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_length)**

Returns the length of the string ***str***, measured in bytes. A multibyte character counts as multiple bytes. This means that for a string containing five 2-byte characters, [**LENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_length) returns **10**, whereas [**CHAR\_LENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char-length) returns **5**.

mysql> **SELECT LENGTH('text');**

-> 4

**Note**

The **Length()** OpenGIS spatial function is named [**ST\_Length()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-length) in MySQL.

**[LOAD\_FILE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_load-file)*[file\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_load-file)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_load-file)**

Reads the file and returns the file contents as a string. To use this function, the file must be located on the server host, you must specify the full path name to the file, and you must have the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege. The file must be readable by the server and its size less than [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) bytes. If the [**secure\_file\_priv**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_secure_file_priv) system variable is set to a nonempty directory name, the file to be loaded must be located in that directory. (Prior to MySQL 8.0.17, the file must be readable by all, not just readable by the server.)

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

The [**character\_set\_filesystem**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_filesystem) system variable controls interpretation of file names that are given as literal strings.

mysql> **UPDATE t**

**SET blob\_col=LOAD\_FILE('/tmp/picture')**

**WHERE id=1;**

**[LOCATE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_locate)*[substr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_locate)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_locate)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_locate)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_locate)**, [**LOCATE(*substr*,*str*,*pos*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_locate)

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

mysql> **SELECT LOCATE('bar', 'foobarbar');**

-> 4

mysql> **SELECT LOCATE('xbar', 'foobar');**

-> 0

mysql> **SELECT LOCATE('bar', 'foobarbar', 5);**

-> 7

This function is multibyte safe, and is case-sensitive only if at least one argument is a binary string.

**[LOWER(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lower)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lower)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lower)**

Returns the string ***str*** with all characters changed to lowercase according to the current character set mapping. The default is **utf8mb4**.

mysql> **SELECT LOWER('QUADRATICALLY');**

-> 'quadratically'

[**LOWER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lower) (and [**UPPER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper)) are ineffective when applied to binary strings ([**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)). To perform lettercase conversion of a binary string, first convert it to a nonbinary string using a character set appropriate for the data stored in the string:

mysql> **SET @str = BINARY 'New York';**

mysql> **SELECT LOWER(@str), LOWER(CONVERT(@str USING utf8mb4));**

+-------------+------------------------------------+

| LOWER(@str) | LOWER(CONVERT(@str USING utf8mb4)) |

+-------------+------------------------------------+

| New York | new york |

+-------------+------------------------------------+

For collations of Unicode character sets, [**LOWER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lower) and [**UPPER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper) work according to the Unicode Collation Algorithm (UCA) version in the collation name, if there is one, and UCA 4.0.0 if no version is specified. For example, **utf8mb4\_0900\_ai\_ci** and **utf8\_unicode\_520\_ci** work according to UCA 9.0.0 and 5.2.0, respectively, whereas **utf8\_unicode\_ci** works according to UCA 4.0.0. See [Section 10.10.1, “Unicode Character Sets”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets).

This function is multibyte safe.

**LCASE()** used within views is rewritten as **LOWER()**.

**[LPAD(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lpad)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lpad)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lpad)*[len](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lpad)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lpad)*[padstr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lpad)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lpad)**

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

mysql> **SELECT LPAD('hi',4,'??');**

-> '??hi'

mysql> **SELECT LPAD('hi',1,'??');**

-> 'h'

**[LTRIM(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ltrim)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ltrim)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ltrim)**

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

mysql> **SELECT LTRIM(' barbar');**

-> 'barbar'

This function is multibyte safe.

**[MAKE\_SET(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_make-set)*[bits](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_make-set)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_make-set)*[str1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_make-set)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_make-set)*[str2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_make-set)*[,...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_make-set)**

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

mysql> **SELECT MAKE\_SET(1,'a','b','c');**

-> 'a'

mysql> **SELECT MAKE\_SET(1 | 4,'hello','nice','world');**

-> 'hello,world'

mysql> **SELECT MAKE\_SET(1 | 4,'hello','nice',NULL,'world');**

-> 'hello'

mysql> **SELECT MAKE\_SET(0,'a','b','c');**

-> ''

**[MID(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[pos](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[len](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)**

**[MID(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[pos](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[len](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mid)** is a synonym for [**SUBSTRING(*str*,*pos*,*len*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring).

**[OCT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_oct)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_oct)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_oct)**

Returns a string representation of the octal value of ***N***, where ***N*** is a longlong ([**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types)) number. This is equivalent to [**CONV(*N*,10,8)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_conv). Returns **NULL** if ***N*** is **NULL**.

mysql> **SELECT OCT(12);**

-> '14'

**[OCTET\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_octet-length)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_octet-length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_octet-length)**

**[OCTET\_LENGTH()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_octet-length)** is a synonym for [**LENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_length).

**[ORD(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ord)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ord)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ord)**

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

(1st byte code)

+ (2nd byte code \* 256)

+ (3rd byte code \* 256^2) ...

If the leftmost character is not a multibyte character, [**ORD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ord) returns the same value as the [**ASCII()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ascii) function.

mysql> **SELECT ORD('2');**

-> 50

**[POSITION(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_position)*[substr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_position)*[IN](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_position)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_position)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_position)**

**[POSITION(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_position)*[substr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_position)*[IN](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_position)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_position)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_position)** is a synonym for [**LOCATE(*substr*,*str*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_locate).

**[QUOTE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_quote)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_quote)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_quote)**

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

mysql> **SELECT QUOTE('Don\'t!');**

-> 'Don\'t!'

mysql> **SELECT QUOTE(NULL);**

-> NULL

For comparison, see the quoting rules for literal strings and within the C API in [Section 9.1.1, “String Literals”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#string-literals), and [mysql\_real\_escape\_string\_quote()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-escape-string-quote.html).

**[REPEAT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_repeat)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_repeat)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_repeat)*[count](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_repeat)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_repeat)**

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

mysql> **SELECT REPEAT('MySQL', 3);**

-> 'MySQLMySQLMySQL'

**[REPLACE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_replace)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_replace)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_replace)*[from\_str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_replace)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_replace)*[to\_str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_replace)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_replace)**

Returns the string ***str*** with all occurrences of the string ***from\_str*** replaced by the string ***to\_str***. [**REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_replace) performs a case-sensitive match when searching for ***from\_str***.

mysql> **SELECT REPLACE('www.mysql.com', 'w', 'Ww');**

-> 'WwWwWw.mysql.com'

This function is multibyte safe.

**[REVERSE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_reverse)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_reverse)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_reverse)**

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

mysql> **SELECT REVERSE('abc');**

-> 'cba'

This function is multibyte safe.

**[RIGHT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_right)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_right)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_right)*[len](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_right)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_right)**

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

mysql> **SELECT RIGHT('foobarbar', 4);**

-> 'rbar'

This function is multibyte safe.

**[RPAD(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rpad)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rpad)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rpad)*[len](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rpad)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rpad)*[padstr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rpad)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rpad)**

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

mysql> **SELECT RPAD('hi',5,'?');**

-> 'hi???'

mysql> **SELECT RPAD('hi',1,'?');**

-> 'h'

This function is multibyte safe.

**[RTRIM(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rtrim)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rtrim)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rtrim)**

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

mysql> **SELECT RTRIM('barbar ');**

-> 'barbar'

This function is multibyte safe.

**[SOUNDEX(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_soundex)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_soundex)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_soundex)**

Returns a soundex string from ***str***. Two strings that sound almost the same should have identical soundex strings. A standard soundex string is four characters long, but the [**SOUNDEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_soundex) function returns an arbitrarily long string. You can use [**SUBSTRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring) on the result to get a standard soundex string. All nonalphabetic characters in ***str*** are ignored. All international alphabetic characters outside the A-Z range are treated as vowels.

**Important**

When using [**SOUNDEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_soundex), you should be aware of the following limitations:

This function, as currently implemented, is intended to work well with strings that are in the English language only. Strings in other languages may not produce reliable results.

This function is not guaranteed to provide consistent results with strings that use multibyte character sets, including **utf-8**. See Bug #22638 for more information.

mysql> **SELECT SOUNDEX('Hello');**

-> 'H400'

mysql> **SELECT SOUNDEX('Quadratically');**

-> 'Q36324'

**Note**

This function implements the original Soundex algorithm, not the more popular enhanced version (also described by D. Knuth). The difference is that original version discards vowels first and duplicates second, whereas the enhanced version discards duplicates first and vowels second.

***[expr1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_sounds-like)*[SOUNDS LIKE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_sounds-like)*[expr2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_sounds-like)***

This is the same as [**SOUNDEX(*expr1*) = SOUNDEX(*expr2*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_soundex).

**[SPACE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_space)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_space)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_space)**

Returns a string consisting of ***N*** space characters.

mysql> **SELECT SPACE(6);**

-> ' '

**[SUBSTR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substr)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substr)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substr)*[pos](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substr)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substr)**, [**SUBSTR(*str* FROM *pos*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substr), [**SUBSTR(*str*,*pos*,*len*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substr), [**SUBSTR(*str* FROM *pos* FOR *len*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substr)

**[SUBSTR()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substr)** is a synonym for [**SUBSTRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring).

**[SUBSTRING(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring)*[pos](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring)**, [**SUBSTRING(*str* FROM *pos*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring), [**SUBSTRING(*str*,*pos*,*len*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring), [**SUBSTRING(*str* FROM *pos* FOR *len*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring)

The forms without a ***len*** argument return a substring from string ***str*** starting at position ***pos***. The forms with a ***len*** argument return a substring ***len*** characters long from string ***str***, starting at position ***pos***. The forms that use **FROM** are standard SQL syntax. It is also possible to use a negative value for ***pos***. In this case, the beginning of the substring is ***pos*** characters from the end of the string, rather than the beginning. A negative value may be used for ***pos*** in any of the forms of this function. A value of 0 for ***pos*** returns an empty string.

For all forms of [**SUBSTRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring), the position of the first character in the string from which the substring is to be extracted is reckoned as **1**.

mysql> **SELECT SUBSTRING('Quadratically',5);**

-> 'ratically'

mysql> **SELECT SUBSTRING('foobarbar' FROM 4);**

-> 'barbar'

mysql> **SELECT SUBSTRING('Quadratically',5,6);**

-> 'ratica'

mysql> **SELECT SUBSTRING('Sakila', -3);**

-> 'ila'

mysql> **SELECT SUBSTRING('Sakila', -5, 3);**

-> 'aki'

mysql> **SELECT SUBSTRING('Sakila' FROM -4 FOR 2);**

-> 'ki'

This function is multibyte safe.

If ***len*** is less than 1, the result is the empty string.

**[SUBSTRING\_INDEX(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring-index)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring-index)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring-index)*[delim](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring-index)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring-index)*[count](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring-index)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_substring-index)**

Returns the substring from string ***str*** before ***count*** occurrences of the delimiter ***delim***. If ***count*** is positive, everything to the left of the final delimiter (counting from the left) is returned. If ***count*** is negative, everything to the right of the final delimiter (counting from the right) is returned. [**SUBSTRING\_INDEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring-index) performs a case-sensitive match when searching for ***delim***.

mysql> **SELECT SUBSTRING\_INDEX('www.mysql.com', '.', 2);**

-> 'www.mysql'

mysql> **SELECT SUBSTRING\_INDEX('www.mysql.com', '.', -2);**

-> 'mysql.com'

This function is multibyte safe.

**[TO\_BASE64(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_to-base64)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_to-base64)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_to-base64)**

Converts the string argument to base-64 encoded form and returns the result as a character string with the connection character set and collation. If the argument is not a string, it is converted to a string before conversion takes place. The result is **NULL** if the argument is **NULL**. Base-64 encoded strings can be decoded using the [**FROM\_BASE64()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-base64) function.

mysql> **SELECT TO\_BASE64('abc'), FROM\_BASE64(TO\_BASE64('abc'));**

-> 'JWJj', 'abc'

Different base-64 encoding schemes exist. These are the encoding and decoding rules used by [**TO\_BASE64()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_to-base64) and [**FROM\_BASE64()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-base64):

The encoding for alphabet value 62 is **'+'**.

The encoding for alphabet value 63 is **'/'**.

Encoded output consists of groups of 4 printable characters. Each 3 bytes of the input data are encoded using 4 characters. If the last group is incomplete, it is padded with **'='** characters to a length of 4.

A newline is added after each 76 characters of encoded output to divide long output into multiple lines.

Decoding recognizes and ignores newline, carriage return, tab, and space.

**[TRIM([{BOTH | LEADING | TRAILING} [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_trim)*[remstr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_trim)*[] FROM]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_trim)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_trim)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_trim)**, [**TRIM([*remstr* FROM] *str*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_trim)

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

mysql> **SELECT TRIM(' bar ');**

-> 'bar'

mysql> **SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');**

-> 'barxxx'

mysql> **SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');**

-> 'bar'

mysql> **SELECT TRIM(TRAILING 'xyz' FROM 'barxxyz');**

-> 'barx'

This function is multibyte safe.

**[UCASE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ucase)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ucase)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ucase)**

**[UCASE()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ucase)** is a synonym for [**UPPER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper).

**UCASE()** used within views is rewritten as **UPPER()**.

**[UNHEX(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_unhex)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_unhex)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_unhex)**

For a string argument ***str***, [**UNHEX(*str*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unhex) interprets each pair of characters in the argument as a hexadecimal number and converts it to the byte represented by the number. The return value is a binary string.

mysql> **SELECT UNHEX('4D7953514C');**

-> 'MySQL'

mysql> **SELECT X'4D7953514C';**

-> 'MySQL'

mysql> **SELECT UNHEX(HEX('string'));**

-> 'string'

mysql> **SELECT HEX(UNHEX('1267'));**

-> '1267'

The characters in the argument string must be legal hexadecimal digits: **'0'** .. **'9'**, **'A'** .. **'F'**, **'a'** .. **'f'**. If the argument contains any nonhexadecimal digits, the result is **NULL**:

mysql> **SELECT UNHEX('GG');**

+-------------+

| UNHEX('GG') |

+-------------+

| NULL |

+-------------+

A **NULL** result can occur if the argument to [**UNHEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unhex) is a [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) column, because values are padded with **0x00** bytes when stored but those bytes are not stripped on retrieval. For example, **'41'** is stored into a **CHAR(3)** column as **'41 '** and retrieved as **'41'** (with the trailing pad space stripped), so [**UNHEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unhex) for the column value returns **X'41'**. By contrast, **'41'** is stored into a **BINARY(3)** column as **'41\0'** and retrieved as **'41\0'** (with the trailing pad **0x00** byte not stripped). **'\0'** is not a legal hexadecimal digit, so [**UNHEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unhex) for the column value returns **NULL**.

For a numeric argument ***N***, the inverse of [**HEX(*N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex) is not performed by [**UNHEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unhex). Use [**CONV(HEX(*N*),16,10)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_conv) instead. See the description of [**HEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex).

**[UPPER(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_upper)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_upper)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_upper)**

Returns the string ***str*** with all characters changed to uppercase according to the current character set mapping. The default is **utf8mb4**.

mysql> **SELECT UPPER('Hej');**

-> 'HEJ'

See the description of [**LOWER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lower) for information that also applies to [**UPPER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper). This included information about how to perform lettercase conversion of binary strings ([**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)) for which these functions are ineffective, and information about case folding for Unicode character sets.

This function is multibyte safe.

**UCASE()** used within views is rewritten as **UPPER()**.

**[WEIGHT\_STRING(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weight-string)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weight-string)*[[AS {CHAR|BINARY}(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weight-string)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weight-string)*[)] [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weight-string)*[flags](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weight-string)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_weight-string)**

This function returns the weight string for the input string. The return value is a binary string that represents the comparison and sorting value of the string. It has these properties:

If [**WEIGHT\_STRING(*str1*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string) = [**WEIGHT\_STRING(*str2*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string), then ***str1* = *str2*** (***str1*** and ***str2*** are considered equal)

If [**WEIGHT\_STRING(*str1*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string) < [**WEIGHT\_STRING(*str2*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string), then ***str1* < *str2*** (***str1*** sorts before ***str2***)

[**WEIGHT\_STRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string) is a debugging function intended for internal use. Its behavior can change without notice between MySQL versions. It can be used for testing and debugging of collations, especially if you are adding a new collation. See [Section 10.14, “Adding a Collation to a Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation).

This list briefly summarizes the arguments. More details are given in the discussion following the list.

***str***: The input string expression.

**AS** clause: Optional; cast the input string to a given type and length.

***flags***: Optional; unused.

The input string, ***str***, is a string expression. If the input is a nonbinary (character) string such as a [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) value, the return value contains the collation weights for the string. If the input is a binary (byte) string such as a [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) value, the return value is the same as the input (the weight for each byte in a binary string is the byte value). If the input is **NULL**, [**WEIGHT\_STRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string) returns **NULL**.

Examples:

mysql> **SET @s = \_utf8mb4 'AB' COLLATE utf8mb4\_0900\_ai\_ci;**

mysql> **SELECT @s, HEX(@s), HEX(WEIGHT\_STRING(@s));**

+------+---------+------------------------+

| @s | HEX(@s) | HEX(WEIGHT\_STRING(@s)) |

+------+---------+------------------------+

| AB | 4142 | 1C471C60 |

+------+---------+------------------------+

mysql> **SET @s = \_utf8mb4 'ab' COLLATE utf8mb4\_0900\_ai\_ci;**

mysql> **SELECT @s, HEX(@s), HEX(WEIGHT\_STRING(@s));**

+------+---------+------------------------+

| @s | HEX(@s) | HEX(WEIGHT\_STRING(@s)) |

+------+---------+------------------------+

| ab | 6162 | 1C471C60 |

+------+---------+------------------------+

mysql> **SET @s = CAST('AB' AS BINARY);**

mysql> **SELECT @s, HEX(@s), HEX(WEIGHT\_STRING(@s));**

+------+---------+------------------------+

| @s | HEX(@s) | HEX(WEIGHT\_STRING(@s)) |

+------+---------+------------------------+

| AB | 4142 | 4142 |

+------+---------+------------------------+

mysql> **SET @s = CAST('ab' AS BINARY);**

mysql> **SELECT @s, HEX(@s), HEX(WEIGHT\_STRING(@s));**

+------+---------+------------------------+

| @s | HEX(@s) | HEX(WEIGHT\_STRING(@s)) |

+------+---------+------------------------+

| ab | 6162 | 6162 |

+------+---------+------------------------+

The preceding examples use [**HEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex) to display the [**WEIGHT\_STRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string) result. Because the result is a binary value, [**HEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex) can be especially useful when the result contains nonprinting values, to display it in printable form:

mysql> **SET @s = CONVERT(X'C39F' USING utf8) COLLATE utf8\_czech\_ci;**

mysql> **SELECT HEX(WEIGHT\_STRING(@s));**

+------------------------+

| HEX(WEIGHT\_STRING(@s)) |

+------------------------+

| 0FEA0FEA |

+------------------------+

For non-**NULL** return values, the data type of the value is [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) if its length is within the maximum length for [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), otherwise the data type is [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob).

The **AS** clause may be given to cast the input string to a nonbinary or binary string and to force it to a given length:

**AS CHAR(*N*)** casts the string to a nonbinary string and pads it on the right with spaces to a length of ***N*** characters. ***N*** must be at least 1. If ***N*** is less than the length of the input string, the string is truncated to ***N*** characters. No warning occurs for truncation.

**AS BINARY(*N*)** is similar but casts the string to a binary string, ***N*** is measured in bytes (not characters), and padding uses **0x00** bytes (not spaces).

mysql> **SET NAMES 'latin1';**

mysql> **SELECT HEX(WEIGHT\_STRING('ab' AS CHAR(4)));**

+-------------------------------------+

| HEX(WEIGHT\_STRING('ab' AS CHAR(4))) |

+-------------------------------------+

| 41422020 |

+-------------------------------------+

mysql> **SET NAMES 'utf8';**

mysql> **SELECT HEX(WEIGHT\_STRING('ab' AS CHAR(4)));**

+-------------------------------------+

| HEX(WEIGHT\_STRING('ab' AS CHAR(4))) |

+-------------------------------------+

| 0041004200200020 |

+-------------------------------------+

mysql> **SELECT HEX(WEIGHT\_STRING('ab' AS BINARY(4)));**

+---------------------------------------+

| HEX(WEIGHT\_STRING('ab' AS BINARY(4))) |

+---------------------------------------+

| 61620000 |

+---------------------------------------+

The ***flags*** clause currently is unused.

### 12.8.1 String Comparison Functions and Operators

**Table 12.13 String Comparison Functions and Operators**

| **Name** | **Description** |
| --- | --- |
| [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) | Simple pattern matching |
| [**NOT LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-like) | Negation of simple pattern matching |
| [**STRCMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_strcmp) | Compare two strings |

If a string function is given a binary string as an argument, the resulting string is also a binary string. A number converted to a string is treated as a binary string. This affects only comparisons.

Normally, if any expression in a string comparison is case-sensitive, the comparison is performed in case-sensitive fashion.

***[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_like)*[LIKE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_like)*[pat](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_like)*[[ESCAPE '](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_like)*[escape\_char](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_like)*[']](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_like)**

Pattern matching using an SQL pattern. Returns **1** (**TRUE**) or **0** (**FALSE**). If either ***expr*** or ***pat*** is **NULL**, the result is **NULL**.

The pattern need not be a literal string. For example, it can be specified as a string expression or table column. In the latter case, the column must be defined as one of the MySQL string types (see [Section 11.3, “String Data Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#string-types)).

Per the SQL standard, [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) performs matching on a per-character basis, thus it can produce results different from the [**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal) comparison operator:

mysql> **SELECT 'ä' LIKE 'ae' COLLATE latin1\_german2\_ci;**

+-----------------------------------------+

| 'ä' LIKE 'ae' COLLATE latin1\_german2\_ci |

+-----------------------------------------+

| 0 |

+-----------------------------------------+

mysql> **SELECT 'ä' = 'ae' COLLATE latin1\_german2\_ci;**

+--------------------------------------+

| 'ä' = 'ae' COLLATE latin1\_german2\_ci |

+--------------------------------------+

| 1 |

+--------------------------------------+

In particular, trailing spaces are always significant. This differs from comparisons performed with the [**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal) operator, for which the significance of trailing spaces in nonbinary strings (**CHAR**, **VARCHAR**, and **TEXT** values) depends on the pad attribute of the the collation used for the comparison. For more information, see [Trailing Space Handling in Comparisons](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations-trailing-space-comparisons).

With [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) you can use the following two wildcard characters in the pattern:

**%** matches any number of characters, even zero characters.

**\_** matches exactly one character.

mysql> **SELECT 'David!' LIKE 'David\_';**

-> 1

mysql> **SELECT 'David!' LIKE '%D%v%';**

-> 1

To test for literal instances of a wildcard character, precede it by the escape character. If you do not specify the **ESCAPE** character, **\** is assumed, unless the [**NO\_BACKSLASH\_ESCAPES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_backslash_escapes) SQL mode is enabled. In that case, no escape character is used.

**\%** matches one **%** character.

**\\_** matches one **\_** character.

mysql> **SELECT 'David!' LIKE 'David\\_';**

-> 0

mysql> **SELECT 'David\_' LIKE 'David\\_';**

-> 1

To specify a different escape character, use the **ESCAPE** clause:

mysql> **SELECT 'David\_' LIKE 'David|\_' ESCAPE '|';**

-> 1

The escape sequence should be one character long to specify the escape character, or empty to specify that no escape character is used. The expression must evaluate as a constant at execution time. If the [**NO\_BACKSLASH\_ESCAPES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_backslash_escapes) SQL mode is enabled, the sequence cannot be empty.

The following two statements illustrate that string comparisons are not case-sensitive unless one of the operands is case-sensitive (uses a case-sensitive collation or is a binary string):

mysql> **SELECT 'abc' LIKE 'ABC';**

-> 1

mysql> **SELECT 'abc' LIKE \_utf8mb4 'ABC' COLLATE utf8mb4\_0900\_as\_cs;**

-> 0

mysql> **SELECT 'abc' LIKE \_utf8mb4 'ABC' COLLATE utf8mb4\_bin;**

-> 0

mysql> **SELECT 'abc' LIKE BINARY 'ABC';**

-> 0

As an extension to standard SQL, MySQL permits [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) on numeric expressions.

mysql> **SELECT 10 LIKE '1%';**

-> 1

**Note**

MySQL uses C escape syntax in strings (for example, **\n** to represent the newline character). If you want a [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) string to contain a literal **\**, you must double it. (Unless the [**NO\_BACKSLASH\_ESCAPES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_backslash_escapes) SQL mode is enabled, in which case no escape character is used.) For example, to search for **\n**, specify it as **\\n**. To search for **\**, specify it as **\\\\**; this is because the backslashes are stripped once by the parser and again when the pattern match is made, leaving a single backslash to be matched against.

Exception: At the end of the pattern string, backslash can be specified as **\\**. At the end of the string, backslash stands for itself because there is nothing following to escape. Suppose that a table contains the following values:

mysql> **SELECT filename FROM t1;**

+--------------+

| filename |

+--------------+

| C: |

| C:\ |

| C:\Programs |

| C:\Programs\ |

+--------------+

To test for values that end with backslash, you can match the values using either of the following patterns:

mysql> **SELECT filename, filename LIKE '%\\' FROM t1;**

+--------------+---------------------+

| filename | filename LIKE '%\\' |

+--------------+---------------------+

| C: | 0 |

| C:\ | 1 |

| C:\Programs | 0 |

| C:\Programs\ | 1 |

+--------------+---------------------+

mysql> **SELECT filename, filename LIKE '%\\\\' FROM t1;**

+--------------+-----------------------+

| filename | filename LIKE '%\\\\' |

+--------------+-----------------------+

| C: | 0 |

| C:\ | 1 |

| C:\Programs | 0 |

| C:\Programs\ | 1 |

+--------------+-----------------------+

***[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-like)*[NOT LIKE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-like)*[pat](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-like)*[[ESCAPE '](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-like)*[escape\_char](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-like)*[']](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-like)**

This is the same as **NOT (*expr* LIKE *pat* [ESCAPE '*escape\_char*'])**.

**Note**

Aggregate queries involving [**NOT LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-like) comparisons with columns containing **NULL** may yield unexpected results. For example, consider the following table and data:

CREATE TABLE foo (bar VARCHAR(10));

INSERT INTO foo VALUES (NULL), (NULL);

The query **SELECT COUNT(\*) FROM foo WHERE bar LIKE '%baz%';** returns **0**. You might assume that **SELECT COUNT(\*) FROM foo WHERE bar NOT LIKE '%baz%';** would return **2**. However, this is not the case: The second query returns **0**. This is because **NULL NOT LIKE *expr*** always returns **NULL**, regardless of the value of ***expr***. The same is true for aggregate queries involving **NULL** and comparisons using [**NOT RLIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-regexp) or [**NOT REGEXP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-regexp). In such cases, you must test explicitly for **NOT NULL** using [**OR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or) (and not [**AND**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_and)), as shown here:

SELECT COUNT(\*) FROM foo WHERE bar NOT LIKE '%baz%' OR bar IS NULL;

**[STRCMP(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_strcmp)*[expr1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_strcmp)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_strcmp)*[expr2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_strcmp)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_strcmp)**

**[STRCMP()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_strcmp)** returns **0** if the strings are the same, **-1** if the first argument is smaller than the second according to the current sort order, and **1** otherwise.

mysql> **SELECT STRCMP('text', 'text2');**

-> -1

mysql> **SELECT STRCMP('text2', 'text');**

-> 1

mysql> **SELECT STRCMP('text', 'text');**

-> 0

[**STRCMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_strcmp) performs the comparison using the collation of the arguments.

mysql> **SET @s1 = \_utf8mb4 'x' COLLATE utf8mb4\_0900\_ai\_ci;**

mysql> **SET @s2 = \_utf8mb4 'X' COLLATE utf8mb4\_0900\_ai\_ci;**

mysql> **SET @s3 = \_utf8mb4 'x' COLLATE utf8mb4\_0900\_as\_cs;**

mysql> **SET @s4 = \_utf8mb4 'X' COLLATE utf8mb4\_0900\_as\_cs;**

mysql> **SELECT STRCMP(@s1, @s2), STRCMP(@s3, @s4);**

+------------------+------------------+

| STRCMP(@s1, @s2) | STRCMP(@s3, @s4) |

+------------------+------------------+

| 0 | -1 |

+------------------+------------------+

If the collations are incompatible, one of the arguments must be converted to be compatible with the other. See [Section 10.8.4, “Collation Coercibility in Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-coercibility).

mysql> SET @s1 = \_utf8mb4 'x' COLLATE utf8mb4\_0900\_ai\_ci;

mysql> SET @s2 = \_utf8mb4 'X' COLLATE utf8mb4\_0900\_ai\_ci;

mysql> SET @s3 = \_utf8mb4 'x' COLLATE utf8mb4\_0900\_as\_cs;

mysql> SET @s4 = \_utf8mb4 'X' COLLATE utf8mb4\_0900\_as\_cs;

-->

mysql> **SELECT STRCMP(@s1, @s3);**

ERROR 1267 (HY000): Illegal mix of collations (utf8mb4\_0900\_ai\_ci,IMPLICIT)

and (utf8mb4\_0900\_as\_cs,IMPLICIT) for operation 'strcmp'

mysql> **SELECT STRCMP(@s1, @s3 COLLATE utf8mb4\_0900\_ai\_ci);**

+---------------------------------------------+

| STRCMP(@s1, @s3 COLLATE utf8mb4\_0900\_ai\_ci) |

+---------------------------------------------+

| 0 |

+---------------------------------------------+

### 12.8.2 Regular Expressions

**Table 12.14 Regular Expression Functions and Operators**

| **Name** | **Description** |
| --- | --- |
| [**NOT REGEXP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-regexp) | Negation of REGEXP |
| [**REGEXP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) | Whether string matches regular expression |
| [**REGEXP\_INSTR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-instr) | Starting index of substring matching regular expression |
| [**REGEXP\_LIKE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like) | Whether string matches regular expression |
| [**REGEXP\_REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-replace) | Replace substrings matching regular expression |
| [**REGEXP\_SUBSTR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-substr) | Return substring matching regular expression |
| [**RLIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) | Whether string matches regular expression |

A regular expression is a powerful way of specifying a pattern for a complex search. This section discusses the functions and operators available for regular expression matching and illustrates, with examples, some of the special characters and constructs that can be used for regular expression operations. See also [Section 3.3.4.7, “Pattern Matching”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\tutorial.html#pattern-matching).

MySQL implements regular expression support using International Components for Unicode (ICU), which provides full Unicode support and is multibyte safe. (Prior to MySQL 8.0.4, MySQL used Henry Spencer's implementation of regular expressions, which operates in byte-wise fashion and is not multibyte safe. For information about ways in which applications that use regular expressions may be affected by the implementation change, see [Regular Expression Compatibility Considerations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#regexp-compatibility).)

[Regular Expression Functions and Operators](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#regexp-operators)

[Regular Expression Syntax](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#regexp-syntax)

[Regular Expression Resource Control](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#regexp-resource-control)

[Regular Expression Compatibility Considerations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#regexp-compatibility)

#### Regular Expression Functions and Operators

***[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-regexp)*[NOT REGEXP](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-regexp)*[pat](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_not-regexp)***, [***expr* NOT RLIKE *pat***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_not-regexp)

This is the same as **NOT (*expr* REGEXP *pat*)**.

***[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_regexp)*[REGEXP](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_regexp)*[pat](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_regexp)***, [***expr* RLIKE *pat***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp)

Returns 1 if the string ***expr*** matches the regular expression specified by the pattern ***pat***, 0 otherwise. If ***expr*** or ***pat*** is **NULL**, the return value is **NULL**.

[**REGEXP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) and [**RLIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) are synonyms for [**REGEXP\_LIKE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like).

For additional information about how matching occurs, see the description for [**REGEXP\_LIKE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like).

mysql> **SELECT 'Michael!' REGEXP '.\*';**

+------------------------+

| 'Michael!' REGEXP '.\*' |

+------------------------+

| 1 |

+------------------------+

mysql> **SELECT 'new\*\n\*line' REGEXP 'new\\\*.\\\*line';**

+---------------------------------------+

| 'new\*\n\*line' REGEXP 'new\\\*.\\\*line' |

+---------------------------------------+

| 0 |

+---------------------------------------+

mysql> **SELECT 'a' REGEXP '^[a-d]';**

+---------------------+

| 'a' REGEXP '^[a-d]' |

+---------------------+

| 1 |

+---------------------+

mysql> **SELECT 'a' REGEXP 'A', 'a' REGEXP BINARY 'A';**

+----------------+-----------------------+

| 'a' REGEXP 'A' | 'a' REGEXP BINARY 'A' |

+----------------+-----------------------+

| 1 | 0 |

+----------------+-----------------------+

**[REGEXP\_INSTR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[pat](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[pos](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[occurrence](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[return\_option](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[match\_type](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)*[]]]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-instr)**

Returns the starting index of the substring of the string ***expr*** that matches the regular expression specified by the pattern ***pat***, 0 if there is no match. If ***expr*** or ***pat*** is **NULL**, the return value is **NULL**. Character indexes begin at 1.

[**REGEXP\_INSTR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-instr) takes these optional arguments:

***pos***: The position in ***expr*** at which to start the search. If omitted, the default is 1.

***occurrence***: Which occurrence of a match to search for. If omitted, the default is 1.

***return\_option***: Which type of position to return. If this value is 0, [**REGEXP\_INSTR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-instr) returns the position of the matched substring's first character. If this value is 1, [**REGEXP\_INSTR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-instr) returns the position following the matched substring. If omitted, the default is 0.

***match\_type***: A string that specifies how to perform matching. The meaning is as described for [**REGEXP\_LIKE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like).

For additional information about how matching occurs, see the description for [**REGEXP\_LIKE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like).

mysql> **SELECT REGEXP\_INSTR('dog cat dog', 'dog');**

+------------------------------------+

| REGEXP\_INSTR('dog cat dog', 'dog') |

+------------------------------------+

| 1 |

+------------------------------------+

mysql> **SELECT REGEXP\_INSTR('dog cat dog', 'dog', 2);**

+---------------------------------------+

| REGEXP\_INSTR('dog cat dog', 'dog', 2) |

+---------------------------------------+

| 9 |

+---------------------------------------+

mysql> **SELECT REGEXP\_INSTR('aa aaa aaaa', 'a{2}');**

+-------------------------------------+

| REGEXP\_INSTR('aa aaa aaaa', 'a{2}') |

+-------------------------------------+

| 1 |

+-------------------------------------+

mysql> **SELECT REGEXP\_INSTR('aa aaa aaaa', 'a{4}');**

+-------------------------------------+

| REGEXP\_INSTR('aa aaa aaaa', 'a{4}') |

+-------------------------------------+

| 8 |

+-------------------------------------+

**[REGEXP\_LIKE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-like)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-like)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-like)*[pat](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-like)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-like)*[match\_type](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-like)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-like)**

Returns 1 if the string ***expr*** matches the regular expression specified by the pattern ***pat***, 0 otherwise. If ***expr*** or ***pat*** is **NULL**, the return value is **NULL**.

The pattern can be an extended regular expression, the syntax for which is discussed in [Regular Expression Syntax](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#regexp-syntax). The pattern need not be a literal string. For example, it can be specified as a string expression or table column.

The optional ***match\_type*** argument is a string that may contain any or all the following characters specifying how to perform matching:

**c**: Case-sensitive matching.

**i**: Case-insensitive matching.

**m**: Multiple-line mode. Recognize line terminators within the string. The default behavior is to match line terminators only at the start and end of the string expression.

**n**: The **.** character matches line terminators. The default is for **.** matching to stop at the end of a line.

**u**: Unix-only line endings. Only the newline character is recognized as a line ending by the **.**, **^**, and **$** match operators.

If characters specifying contradictory options are specified within ***match\_type***, the rightmost one takes precedence.

By default, regular expression operations use the character set and collation of the ***expr*** and ***pat*** arguments when deciding the type of a character and performing the comparison. If the arguments have different character sets or collations, coercibility rules apply as described in [Section 10.8.4, “Collation Coercibility in Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-coercibility). Arguments may be specified with explicit collation indicators to change comparison behavior.

mysql> **SELECT REGEXP\_LIKE('CamelCase', 'CAMELCASE');**

+---------------------------------------+

| REGEXP\_LIKE('CamelCase', 'CAMELCASE') |

+---------------------------------------+

| 1 |

+---------------------------------------+

mysql> **SELECT REGEXP\_LIKE('CamelCase', 'CAMELCASE' COLLATE utf8mb4\_0900\_as\_cs);**

+------------------------------------------------------------------+

| REGEXP\_LIKE('CamelCase', 'CAMELCASE' COLLATE utf8mb4\_0900\_as\_cs) |

+------------------------------------------------------------------+

| 0 |

+------------------------------------------------------------------+

***match\_type*** may be specified with the **c** or **i** characters to override the default case sensitivity. Exception: If either argument is a binary string, the arguments are handled in case-sensitive fashion as binary strings, even if ***match\_type*** contains the **i** character.

**Note**

MySQL uses C escape syntax in strings (for example, **\n** to represent the newline character). If you want your ***expr*** or ***pat*** argument to contain a literal **\**, you must double it. (Unless the [**NO\_BACKSLASH\_ESCAPES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_backslash_escapes) SQL mode is enabled, in which case no escape character is used.)

mysql> **SELECT REGEXP\_LIKE('Michael!', '.\*');**

+-------------------------------+

| REGEXP\_LIKE('Michael!', '.\*') |

+-------------------------------+

| 1 |

+-------------------------------+

mysql> **SELECT REGEXP\_LIKE('new\*\n\*line', 'new\\\*.\\\*line');**

+----------------------------------------------+

| REGEXP\_LIKE('new\*\n\*line', 'new\\\*.\\\*line') |

+----------------------------------------------+

| 0 |

+----------------------------------------------+

mysql> **SELECT REGEXP\_LIKE('a', '^[a-d]');**

+----------------------------+

| REGEXP\_LIKE('a', '^[a-d]') |

+----------------------------+

| 1 |

+----------------------------+

mysql> **SELECT REGEXP\_LIKE('a', 'A'), REGEXP\_LIKE('a', BINARY 'A');**

+-----------------------+------------------------------+

| REGEXP\_LIKE('a', 'A') | REGEXP\_LIKE('a', BINARY 'A') |

+-----------------------+------------------------------+

| 1 | 0 |

+-----------------------+------------------------------+

mysql> **SELECT REGEXP\_LIKE('abc', 'ABC');**

+---------------------------+

| REGEXP\_LIKE('abc', 'ABC') |

+---------------------------+

| 1 |

+---------------------------+

mysql> **SELECT REGEXP\_LIKE('abc', 'ABC', 'c');**

+--------------------------------+

| REGEXP\_LIKE('abc', 'ABC', 'c') |

+--------------------------------+

| 0 |

+--------------------------------+

**[REGEXP\_REPLACE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[pat](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[repl](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[pos](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[occurrence](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[match\_type](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)*[]]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-replace)**

Replaces occurrences in the string ***expr*** that match the regular expression specified by the pattern ***pat*** with the replacement string ***repl***, and returns the resulting string. If ***expr***, ***pat***, or ***repl*** is **NULL**, the return value is **NULL**.

[**REGEXP\_REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-replace) takes these optional arguments:

***pos***: The position in ***expr*** at which to start the search. If omitted, the default is 1.

***occurrence***: Which occurrence of a match to replace. If omitted, the default is 0 (which means “replace all occurrences”).

***match\_type***: A string that specifies how to perform matching. The meaning is as described for [**REGEXP\_LIKE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like).

Prior to MySQL 8.0.17, the result returned by this function used the **UTF-16** character set; in MySQL 8.0.17 and later, the character set and collation of the expression searched for matches is used. (Bug #94203, Bug #29308212)

For additional information about how matching occurs, see the description for [**REGEXP\_LIKE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like).

mysql> **SELECT REGEXP\_REPLACE('a b c', 'b', 'X');**

+-----------------------------------+

| REGEXP\_REPLACE('a b c', 'b', 'X') |

+-----------------------------------+

| a X c |

+-----------------------------------+

mysql> **SELECT REGEXP\_REPLACE('abc def ghi', '[a-z]+', 'X', 1, 3);**

+----------------------------------------------------+

| REGEXP\_REPLACE('abc def ghi', '[a-z]+', 'X', 1, 3) |

+----------------------------------------------------+

| abc def X |

+----------------------------------------------------+

**[REGEXP\_SUBSTR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-substr)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-substr)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-substr)*[pat](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-substr)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-substr)*[pos](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-substr)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-substr)*[occurrence](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-substr)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-substr)*[match\_type](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-substr)*[]]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_regexp-substr)**

Returns the substring of the string ***expr*** that matches the regular expression specified by the pattern ***pat***, **NULL** if there is no match. If ***expr*** or ***pat*** is **NULL**, the return value is **NULL**.

[**REGEXP\_SUBSTR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-substr) takes these optional arguments:

***pos***: The position in ***expr*** at which to start the search. If omitted, the default is 1.

***occurrence***: Which occurrence of a match to search for. If omitted, the default is 1.

***match\_type***: A string that specifies how to perform matching. The meaning is as described for [**REGEXP\_LIKE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like).

Prior to MySQL 8.0.17, the result returned by this function used the **UTF-16** character set; in MySQL 8.0.17 and later, the character set and collation of the expression searched for matches is used. (Bug #94203, Bug #29308212)

For additional information about how matching occurs, see the description for [**REGEXP\_LIKE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like).

mysql> **SELECT REGEXP\_SUBSTR('abc def ghi', '[a-z]+');**

+----------------------------------------+

| REGEXP\_SUBSTR('abc def ghi', '[a-z]+') |

+----------------------------------------+

| abc |

+----------------------------------------+

mysql> **SELECT REGEXP\_SUBSTR('abc def ghi', '[a-z]+', 1, 3);**

+----------------------------------------------+

| REGEXP\_SUBSTR('abc def ghi', '[a-z]+', 1, 3) |

+----------------------------------------------+

| ghi |

+----------------------------------------------+

#### Regular Expression Syntax

A regular expression describes a set of strings. The simplest regular expression is one that has no special characters in it. For example, the regular expression **hello** matches **hello** and nothing else.

Nontrivial regular expressions use certain special constructs so that they can match more than one string. For example, the regular expression **hello|world** contains the **|** alternation operator and matches either the **hello** or **world**.

As a more complex example, the regular expression **B[an]\*s** matches any of the strings **Bananas**, **Baaaaas**, **Bs**, and any other string starting with a **B**, ending with an **s**, and containing any number of **a** or **n** characters in between.

The following list covers some of the basic special characters and constructs that can be used in regular expressions. For information about the full regular expression syntax supported by the ICU library used to implement regular expression support, visit the [International Components for Unicode website](http://userguide.icu-project.org/strings/regexp).

**^**

Match the beginning of a string.

mysql> **SELECT REGEXP\_LIKE('fo\nfo', '^fo$');** -> 0

mysql> **SELECT REGEXP\_LIKE('fofo', '^fo');** -> 1

**$**

Match the end of a string.

mysql> **SELECT REGEXP\_LIKE('fo\no', '^fo\no$');** -> 1

mysql> **SELECT REGEXP\_LIKE('fo\no', '^fo$');** -> 0

**.**

Match any character (including carriage return and newline, although to match these in the middle of a string, the **m** (multiple line) match-control character or the **(?m)** within-pattern modifier must be given).

mysql> **SELECT REGEXP\_LIKE('fofo', '^f.\*$');** -> 1

mysql> **SELECT REGEXP\_LIKE('fo\r\nfo', '^f.\*$');** -> 0

mysql> **SELECT REGEXP\_LIKE('fo\r\nfo', '^f.\*$', 'm');** -> 1

mysql> **SELECT REGEXP\_LIKE('fo\r\nfo', '(?m)^f.\*$');** -> 1

**a\***

Match any sequence of zero or more **a** characters.

mysql> **SELECT REGEXP\_LIKE('Ban', '^Ba\*n');** -> 1

mysql> **SELECT REGEXP\_LIKE('Baaan', '^Ba\*n');** -> 1

mysql> **SELECT REGEXP\_LIKE('Bn', '^Ba\*n');** -> 1

**a+**

Match any sequence of one or more **a** characters.

mysql> **SELECT REGEXP\_LIKE('Ban', '^Ba+n');** -> 1

mysql> **SELECT REGEXP\_LIKE('Bn', '^Ba+n');** -> 0

**a?**

Match either zero or one **a** character.

mysql> **SELECT REGEXP\_LIKE('Bn', '^Ba?n');** -> 1

mysql> **SELECT REGEXP\_LIKE('Ban', '^Ba?n');** -> 1

mysql> **SELECT REGEXP\_LIKE('Baan', '^Ba?n');** -> 0

**de|abc**

Alternation; match either of the sequences **de** or **abc**.

mysql> **SELECT REGEXP\_LIKE('pi', 'pi|apa');** -> 1

mysql> **SELECT REGEXP\_LIKE('axe', 'pi|apa');** -> 0

mysql> **SELECT REGEXP\_LIKE('apa', 'pi|apa');** -> 1

mysql> **SELECT REGEXP\_LIKE('apa', '^(pi|apa)$');** -> 1

mysql> **SELECT REGEXP\_LIKE('pi', '^(pi|apa)$');** -> 1

mysql> **SELECT REGEXP\_LIKE('pix', '^(pi|apa)$');** -> 0

**(abc)\***

Match zero or more instances of the sequence **abc**.

mysql> **SELECT REGEXP\_LIKE('pi', '^(pi)\*$');** -> 1

mysql> **SELECT REGEXP\_LIKE('pip', '^(pi)\*$');** -> 0

mysql> **SELECT REGEXP\_LIKE('pipi', '^(pi)\*$');** -> 1

**{1}**, **{2,3}**

Repetition; **{*n*}** and **{*m*,*n*}** notation provide a more general way of writing regular expressions that match many occurrences of the previous atom (or “piece”) of the pattern. ***m*** and ***n*** are integers.

**a\***

Can be written as **a{0,}**.

**a+**

Can be written as **a{1,}**.

**a?**

Can be written as **a{0,1}**.

To be more precise, **a{*n*}** matches exactly ***n*** instances of **a**. **a{*n*,}** matches ***n*** or more instances of **a**. **a{*m*,*n*}** matches ***m*** through ***n*** instances of **a**, inclusive. If both ***m*** and ***n*** are given, ***m*** must be less than or equal to ***n***.

mysql> **SELECT REGEXP\_LIKE('abcde', 'a[bcd]{2}e');** -> 0

mysql> **SELECT REGEXP\_LIKE('abcde', 'a[bcd]{3}e');** -> 1

mysql> **SELECT REGEXP\_LIKE('abcde', 'a[bcd]{1,10}e');** -> 1

**[a-dX]**, **[^a-dX]**

Matches any character that is (or is not, if **^** is used) either **a**, **b**, **c**, **d** or **X**. A **-** character between two other characters forms a range that matches all characters from the first character to the second. For example, **[0-9]** matches any decimal digit. To include a literal **]** character, it must immediately follow the opening bracket **[**. To include a literal **-** character, it must be written first or last. Any character that does not have a defined special meaning inside a **[]** pair matches only itself.

mysql> **SELECT REGEXP\_LIKE('aXbc', '[a-dXYZ]');** -> 1

mysql> **SELECT REGEXP\_LIKE('aXbc', '^[a-dXYZ]$');** -> 0

mysql> **SELECT REGEXP\_LIKE('aXbc', '^[a-dXYZ]+$');** -> 1

mysql> **SELECT REGEXP\_LIKE('aXbc', '^[^a-dXYZ]+$');** -> 0

mysql> **SELECT REGEXP\_LIKE('gheis', '^[^a-dXYZ]+$');** -> 1

mysql> **SELECT REGEXP\_LIKE('gheisa', '^[^a-dXYZ]+$');** -> 0

**[=character\_class=]**

Within a bracket expression (written using **[** and **]**), **[=character\_class=]** represents an equivalence class. It matches all characters with the same collation value, including itself. For example, if **o** and **(+)** are the members of an equivalence class, **[[=o=]]**, **[[=(+)=]]**, and **[o(+)]** are all synonymous. An equivalence class may not be used as an endpoint of a range.

**[:character\_class:]**

Within a bracket expression (written using **[** and **]**), **[:character\_class:]** represents a character class that matches all characters belonging to that class. The following table lists the standard class names. These names stand for the character classes defined in the **ctype(3)** manual page. A particular locale may provide other class names. A character class may not be used as an endpoint of a range.

| **Character Class Name** | **Meaning** |
| --- | --- |
| **alnum** | Alphanumeric characters |
| **alpha** | Alphabetic characters |
| **blank** | Whitespace characters |
| **cntrl** | Control characters |
| **digit** | Digit characters |
| **graph** | Graphic characters |
| **lower** | Lowercase alphabetic characters |
| **print** | Graphic or space characters |
| **punct** | Punctuation characters |
| **space** | Space, tab, newline, and carriage return |
| **upper** | Uppercase alphabetic characters |
| **xdigit** | Hexadecimal digit characters |

mysql> **SELECT REGEXP\_LIKE('justalnums', '[[:alnum:]]+');** -> 1

mysql> **SELECT REGEXP\_LIKE('!!', '[[:alnum:]]+');** -> 0

To use a literal instance of a special character in a regular expression, precede it by two backslash (\) characters. The MySQL parser interprets one of the backslashes, and the regular expression library interprets the other. For example, to match the string **1+2** that contains the special **+** character, only the last of the following regular expressions is the correct one:

mysql> **SELECT REGEXP\_LIKE('1+2', '1+2');** -> 0

mysql> **SELECT REGEXP\_LIKE('1+2', '1\+2');** -> 0

mysql> **SELECT REGEXP\_LIKE('1+2', '1\\+2');** -> 1

#### Regular Expression Resource Control

[**REGEXP\_LIKE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-like) and similar functions use resources that can be controlled by setting system variables:

The match engine uses memory for its internal stack. To control the maximum available memory for the stack in bytes, set the [**regexp\_stack\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_regexp_stack_limit) system variable.

The match engine operates in steps. To control the maximum number of steps performed by the engine (and thus indirectly the execution time), set the [**regexp\_time\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_regexp_time_limit) system variable. Because this limit is expressed as number of steps, it affects execution time only indirectly. Typically, it is on the order of milliseconds.

#### Regular Expression Compatibility Considerations

Prior to MySQL 8.0.4, MySQL used the Henry Spencer regular expression library to support regular expression operations, rather than International Components for Unicode (ICU). The following discussion describes differences between the Spencer and ICU libraries that may affect applications:

With the Spencer library, the [**REGEXP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) and [**RLIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) operators work in byte-wise fashion, so they are not multibyte safe and may produce unexpected results with multibyte character sets. In addition, these operators compare characters by their byte values and accented characters may not compare as equal even if a given collation treats them as equal.

ICU has full Unicode support and is multibyte safe. Its regular expression functions treat all strings as **UTF-16**. You should keep in mind that positional indexes are based on 16-bit chunks and not on code points. This means that, when passed to such functions, characters using more than one chunk may produce unanticipated results, such as those shown here:

mysql> **SELECT REGEXP\_INSTR('🍣🍣b', 'b');**

+--------------------------+

| REGEXP\_INSTR('??b', 'b') |

+--------------------------+

| 5 |

+--------------------------+

1 row in set (0.00 sec)

mysql> **SELECT REGEXP\_INSTR('🍣🍣bxxx', 'b', 4);**

+--------------------------------+

| REGEXP\_INSTR('??bxxx', 'b', 4) |

+--------------------------------+

| 5 |

+--------------------------------+

1 row in set (0.00 sec)

Characters within the Unicode Basic Multilingual Plane, which includes characters used by most modern languages, are safe in this regard:

mysql> **SELECT REGEXP\_INSTR('бжb', 'b');**

+----------------------------+

| REGEXP\_INSTR('бжb', 'b') |

+----------------------------+

| 3 |

+----------------------------+

1 row in set (0.00 sec)

mysql> **SELECT REGEXP\_INSTR('עבb', 'b');**

+----------------------------+

| REGEXP\_INSTR('עבb', 'b') |

+----------------------------+

| 3 |

+----------------------------+

1 row in set (0.00 sec)

mysql> **SELECT REGEXP\_INSTR('µå周çб', '周');**

+------------------------------------+

| REGEXP\_INSTR('µå周çб', '周') |

+------------------------------------+

| 3 |

+------------------------------------+

1 row in set (0.00 sec)

Emoji, such as the “sushi” character **🍣** (U+1F363) used in the first two examples, are not included in the Basic Multilingual Plane, but rather in Unicode's Supplementary Multilingual Plane. Another issue can arise with emoji and other 4-byte characters when [**REGEXP\_SUBSTR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_regexp-substr) or a similar function begins searching in the middle of a character. Each of the two statements in the following example starts from the second 2-byte position in the first argument. The first statement works on a string consisting solely of 2-byte (BMP) characters. The second statement contains 4-byte characters which are incorrectly interpreted in the result because the first two bytes are stripped off and so the remainder of the character data is misaligned.

mysql> **SELECT REGEXP\_SUBSTR('周周周周', '.\*', 2);**

+----------------------------------------+

| REGEXP\_SUBSTR('周周周周', '.\*', 2) |

+----------------------------------------+

| 周周周 |

+----------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT REGEXP\_SUBSTR('🍣🍣🍣🍣', '.\*', 2);**

+--------------------------------+

| REGEXP\_SUBSTR('????', '.\*', 2) |

+--------------------------------+

| ?㳟揘㳟揘㳟揘 |

+--------------------------------+

1 row in set (0.00 sec)

For the **.** operator, the Spencer library matches line-terminator characters (carriage return, newline) anywhere in string expressions, including in the middle. To match line terminator characters in the middle of strings with ICU, specify the **m** match-control character.

The Spencer library supports word-beginning and word-end boundary markers (**[[:<:]]** and **[[:>:]]** notation). ICU does not. For ICU, you can use **\b** to match word boundaries; double the backslash because MySQL interprets it as the escape character within strings.

The Spencer library supports collating element bracket expressions (**[.characters.]** notation). ICU does not.

For repetition counts (**{n}** and **{m,n}** notation), the Spencer library has a maximum of 255. ICU has no such limit, although the maximum number of match engine steps can be limited by setting the [**regexp\_time\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_regexp_time_limit) system variable.

ICU interprets parentheses as metacharacters. To specify a literal open or close parenthesis **(** in a regular expression, it must be escaped:

mysql> **SELECT REGEXP\_LIKE('(', '(');**

ERROR 3692 (HY000): Mismatched parenthesis in regular expression.

mysql> **SELECT REGEXP\_LIKE('(', '\\(');**

+-------------------------+

| REGEXP\_LIKE('(', '\\(') |

+-------------------------+

| 1 |

+-------------------------+

mysql> **SELECT REGEXP\_LIKE(')', ')');**

ERROR 3692 (HY000): Mismatched parenthesis in regular expression.

mysql> **SELECT REGEXP\_LIKE(')', '\\)');**

+-------------------------+

| REGEXP\_LIKE(')', '\\)') |

+-------------------------+

| 1 |

+-------------------------+

ICU also interprets square brackets as metacharacters, but only the opening square bracket need be escaped to be used as a literal character:

mysql> **SELECT REGEXP\_LIKE('[', '[');**

ERROR 3696 (HY000): The regular expression contains an

unclosed bracket expression.

mysql> **SELECT REGEXP\_LIKE('[', '\\[');**

+-------------------------+

| REGEXP\_LIKE('[', '\\[') |

+-------------------------+

| 1 |

+-------------------------+

mysql> **SELECT REGEXP\_LIKE(']', ']');**

+-----------------------+

| REGEXP\_LIKE(']', ']') |

+-----------------------+

| 1 |

+-----------------------+

### 12.8.3 Character Set and Collation of Function Results

MySQL has many operators and functions that return a string. This section answers the question: What is the character set and collation of such a string?

For simple functions that take string input and return a string result as output, the output's character set and collation are the same as those of the principal input value. For example, [**UPPER(*X*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper) returns a string with the same character string and collation as ***X***. The same applies for [**INSTR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_instr), [**LCASE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lcase), [**LOWER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lower), [**LTRIM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ltrim), [**MID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mid), [**REPEAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_repeat), [**REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_replace), [**REVERSE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_reverse), [**RIGHT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_right), [**RPAD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rpad), [**RTRIM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rtrim), [**SOUNDEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_soundex), [**SUBSTRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring), [**TRIM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_trim), [**UCASE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ucase), and [**UPPER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper).

**Note**

The [**REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_replace) function, unlike all other functions, always ignores the collation of the string input and performs a case-sensitive comparison.

If a string input or function result is a binary string, the string has the **binary** character set and collation. This can be checked by using the [**CHARSET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_charset) and [**COLLATION()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_collation) functions, both of which return **binary** for a binary string argument:

mysql> **SELECT CHARSET(BINARY 'a'), COLLATION(BINARY 'a');**

+---------------------+-----------------------+

| CHARSET(BINARY 'a') | COLLATION(BINARY 'a') |

+---------------------+-----------------------+

| binary | binary |

+---------------------+-----------------------+

For operations that combine multiple string inputs and return a single string output, the “aggregation rules” of standard SQL apply for determining the collation of the result:

If an explicit **COLLATE *Y*** occurs, use ***Y***.

If explicit **COLLATE *Y*** and **COLLATE *Z*** occur, raise an error.

Otherwise, if all collations are ***Y***, use ***Y***.

Otherwise, the result has no collation.

For example, with **CASE ... WHEN a THEN b WHEN b THEN c COLLATE *X* END**, the resulting collation is ***X***. The same applies for [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union), [**||**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or), [**CONCAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat), [**ELT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_elt), [**GREATEST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_greatest), [**IF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if), and [**LEAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_least).

For operations that convert to character data, the character set and collation of the strings that result from the operations are defined by the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variables that determine the default connection character set and collation (see [Section 10.4, “Connection Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection)). This applies only to [**BIN\_TO\_UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bin-to-uuid), [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast), [**CONV()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_conv), [**FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format), [**HEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex), and [**SPACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_space).

An exception to the preceding principle occurs for expressions for virtual generated columns. In such expressions, the table character set is used for [**BIN\_TO\_UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bin-to-uuid), [**CONV()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_conv), or [**HEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex) results, regardless of connection character set.

If there is any question about the character set or collation of the result returned by a string function, use the [**CHARSET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_charset) or [**COLLATION()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_collation) function to find out:

mysql> **SELECT USER(), CHARSET(USER()), COLLATION(USER());**

+----------------+-----------------+-------------------+

| USER() | CHARSET(USER()) | COLLATION(USER()) |

+----------------+-----------------+-------------------+

| test@localhost | utf8 | utf8\_general\_ci |

+----------------+-----------------+-------------------+

mysql> **SELECT CHARSET(COMPRESS('abc')), COLLATION(COMPRESS('abc'));**

+--------------------------+----------------------------+

| CHARSET(COMPRESS('abc')) | COLLATION(COMPRESS('abc')) |

+--------------------------+----------------------------+

| binary | binary |

+--------------------------+----------------------------+

## 12.9 What Calendar Is Used By MySQL?

MySQL uses what is known as a proleptic Gregorian calendar.

Every country that has switched from the Julian to the Gregorian calendar has had to discard at least ten days during the switch. To see how this works, consider the month of October 1582, when the first Julian-to-Gregorian switch occurred.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
| 1 | 2 | 3 | 4 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |

There are no dates between October 4 and October 15. This discontinuity is called the cutover. Any dates before the cutover are Julian, and any dates following the cutover are Gregorian. Dates during a cutover are nonexistent.

A calendar applied to dates when it was not actually in use is called proleptic. Thus, if we assume there was never a cutover and Gregorian rules always rule, we have a proleptic Gregorian calendar. This is what is used by MySQL, as is required by standard SQL. For this reason, dates prior to the cutover stored as MySQL [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) or [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) values must be adjusted to compensate for the difference. It is important to realize that the cutover did not occur at the same time in all countries, and that the later it happened, the more days were lost. For example, in Great Britain, it took place in 1752, when Wednesday September 2 was followed by Thursday September 14. Russia remained on the Julian calendar until 1918, losing 13 days in the process, and what is popularly referred to as its “October Revolution” occurred in November according to the Gregorian calendar.

## 12.10 Full-Text Search Functions

[12.10.1 Natural Language Full-Text Searches](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-natural-language)

[12.10.2 Boolean Full-Text Searches](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-boolean)

[12.10.3 Full-Text Searches with Query Expansion](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-query-expansion)

[12.10.4 Full-Text Stopwords](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-stopwords)

[12.10.5 Full-Text Restrictions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-restrictions)

[12.10.6 Fine-Tuning MySQL Full-Text Search](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-fine-tuning)

[12.10.7 Adding a User-Defined Collation for Full-Text Indexing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#full-text-adding-collation)

[12.10.8 ngram Full-Text Parser](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search-ngram)

[12.10.9 MeCab Full-Text Parser Plugin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search-mecab)

**[MATCH (](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_match)*[col1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_match)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_match)*[col2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_match)*[,...) AGAINST (](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_match)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_match)*[[](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_match)*[search\_modifier](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_match)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_match)**

***search\_modifier:***

{

IN NATURAL LANGUAGE MODE

| IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION

| IN BOOLEAN MODE

| WITH QUERY EXPANSION

}

MySQL has support for full-text indexing and searching:

A full-text index in MySQL is an index of type **FULLTEXT**.

Full-text indexes can be used only with [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) or [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables, and can be created only for [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns.

MySQL provides a built-in full-text ngram parser that supports Chinese, Japanese, and Korean (CJK), and an installable MeCab full-text parser plugin for Japanese. Parsing differences are outlined in [Section 12.10.8, “ngram Full-Text Parser”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search-ngram), and [Section 12.10.9, “MeCab Full-Text Parser Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search-mecab).

A **FULLTEXT** index definition can be given in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement when a table is created, or added later using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) or [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index).

For large data sets, it is much faster to load your data into a table that has no **FULLTEXT** index and then create the index after that, than to load data into a table that has an existing **FULLTEXT** index.

Full-text searching is performed using [**MATCH() AGAINST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) syntax. [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) takes a comma-separated list that names the columns to be searched. **AGAINST** takes a string to search for, and an optional modifier that indicates what type of search to perform. The search string must be a string value that is constant during query evaluation. This rules out, for example, a table column because that can differ for each row.

There are three types of full-text searches:

A natural language search interprets the search string as a phrase in natural human language (a phrase in free text). There are no special operators, with the exception of double quote (") characters. The stopword list applies. For more information about stopword lists, see [Section 12.10.4, “Full-Text Stopwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-stopwords).

Full-text searches are natural language searches if the **IN NATURAL LANGUAGE MODE** modifier is given or if no modifier is given. For more information, see [Section 12.10.1, “Natural Language Full-Text Searches”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-natural-language).

A boolean search interprets the search string using the rules of a special query language. The string contains the words to search for. It can also contain operators that specify requirements such that a word must be present or absent in matching rows, or that it should be weighted higher or lower than usual. Certain common words (stopwords) are omitted from the search index and do not match if present in the search string. The **IN BOOLEAN MODE** modifier specifies a boolean search. For more information, see [Section 12.10.2, “Boolean Full-Text Searches”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-boolean).

A query expansion search is a modification of a natural language search. The search string is used to perform a natural language search. Then words from the most relevant rows returned by the search are added to the search string and the search is done again. The query returns the rows from the second search. The **IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION** or **WITH QUERY EXPANSION** modifier specifies a query expansion search. For more information, see [Section 12.10.3, “Full-Text Searches with Query Expansion”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-query-expansion).

For information about **FULLTEXT** query performance, see [Section 8.3.5, “Column Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#column-indexes).

For more information about **InnoDB** **FULLTEXT** indexes, see [Section 15.6.2.4, “InnoDB Full-Text Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index).

Constraints on full-text searching are listed in [Section 12.10.5, “Full-Text Restrictions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-restrictions).

The [**myisam\_ftdump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisam-ftdump) utility dumps the contents of a **MyISAM** full-text index. This may be helpful for debugging full-text queries. See [Section 4.6.3, “myisam\_ftdump — Display Full-Text Index information”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisam-ftdump).

### 12.10.1 Natural Language Full-Text Searches

By default or with the **IN NATURAL LANGUAGE MODE** modifier, the [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) function performs a natural language search for a string against a text collection. A collection is a set of one or more columns included in a **FULLTEXT** index. The search string is given as the argument to **AGAINST()**. For each row in the table, [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) returns a relevance value; that is, a similarity measure between the search string and the text in that row in the columns named in the [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) list.

mysql> **CREATE TABLE articles (**

**id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,**

**title VARCHAR(200),**

**body TEXT,**

**FULLTEXT (title,body)**

**) ENGINE=InnoDB;**

Query OK, 0 rows affected (0.08 sec)

mysql> **INSERT INTO articles (title,body) VALUES**

**('MySQL Tutorial','DBMS stands for DataBase ...'),**

**('How To Use MySQL Well','After you went through a ...'),**

**('Optimizing MySQL','In this tutorial, we show ...'),**

**('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),**

**('MySQL vs. YourSQL','In the following database comparison ...'),**

**('MySQL Security','When configured properly, MySQL ...');**

Query OK, 6 rows affected (0.01 sec)

Records: 6 Duplicates: 0 Warnings: 0

mysql> **SELECT \* FROM articles**

**WHERE MATCH (title,body)**

**AGAINST ('database' IN NATURAL LANGUAGE MODE);**

+----+-------------------+------------------------------------------+

| id | title | body |

+----+-------------------+------------------------------------------+

| 1 | MySQL Tutorial | DBMS stands for DataBase ... |

| 5 | MySQL vs. YourSQL | In the following database comparison ... |

+----+-------------------+------------------------------------------+

2 rows in set (0.00 sec)

By default, the search is performed in case-insensitive fashion. To perform a case-sensitive full-text search, use a case-sensitive or binary collation for the indexed columns. For example, a column that uses the **utf8mb4** character set of can be assigned a collation of **utf8mb4\_0900\_as\_cs** or **utf8mb4\_bin** to make it case-sensitive for full-text searches.

When [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) is used in a **WHERE** clause, as in the example shown earlier, the rows returned are automatically sorted with the highest relevance first. Relevance values are nonnegative floating-point numbers. Zero relevance means no similarity. Relevance is computed based on the number of words in the row (document), the number of unique words in the row, the total number of words in the collection, and the number of rows that contain a particular word.

**Note**

The term “document” may be used interchangeably with the term “row”, and both terms refer to the indexed part of the row. The term “collection” refers to the indexed columns and encompasses all rows.

To simply count matches, you could use a query like this:

mysql> **SELECT COUNT(\*) FROM articles**

**WHERE MATCH (title,body)**

**AGAINST ('database' IN NATURAL LANGUAGE MODE);**

+----------+

| COUNT(\*) |

+----------+

| 2 |

+----------+

1 row in set (0.00 sec)

You might find it quicker to rewrite the query as follows:

mysql> **SELECT**

**COUNT(IF(MATCH (title,body) AGAINST ('database' IN NATURAL LANGUAGE MODE), 1, NULL))**

**AS count**

**FROM articles;**

+-------+

| count |

+-------+

| 2 |

+-------+

1 row in set (0.03 sec)

The first query does some extra work (sorting the results by relevance) but also can use an index lookup based on the **WHERE** clause. The index lookup might make the first query faster if the search matches few rows. The second query performs a full table scan, which might be faster than the index lookup if the search term was present in most rows.

For natural-language full-text searches, the columns named in the [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) function must be the same columns included in some **FULLTEXT** index in your table. For the preceding query, note that the columns named in the [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) function (**title** and **body**) are the same as those named in the definition of the **article** table's **FULLTEXT** index. To search the **title** or **body** separately, you would create separate **FULLTEXT** indexes for each column.

You can also perform a boolean search or a search with query expansion. These search types are described in [Section 12.10.2, “Boolean Full-Text Searches”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-boolean), and [Section 12.10.3, “Full-Text Searches with Query Expansion”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-query-expansion).

A full-text search that uses an index can name columns only from a single table in the [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) clause because an index cannot span multiple tables. For **MyISAM** tables, a boolean search can be done in the absence of an index (albeit more slowly), in which case it is possible to name columns from multiple tables.

The preceding example is a basic illustration that shows how to use the [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) function where rows are returned in order of decreasing relevance. The next example shows how to retrieve the relevance values explicitly. Returned rows are not ordered because the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement includes neither **WHERE** nor **ORDER BY** clauses:

mysql> **SELECT id, MATCH (title,body)**

**AGAINST ('Tutorial' IN NATURAL LANGUAGE MODE) AS score**

**FROM articles;**

+----+---------------------+

| id | score |

+----+---------------------+

| 1 | 0.22764469683170319 |

| 2 | 0 |

| 3 | 0.22764469683170319 |

| 4 | 0 |

| 5 | 0 |

| 6 | 0 |

+----+---------------------+

6 rows in set (0.00 sec)

The following example is more complex. The query returns the relevance values and it also sorts the rows in order of decreasing relevance. To achieve this result, specify [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) twice: once in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) list and once in the **WHERE** clause. This causes no additional overhead, because the MySQL optimizer notices that the two [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) calls are identical and invokes the full-text search code only once.

mysql> **SELECT id, body, MATCH (title,body) AGAINST**

**('Security implications of running MySQL as root'**

**IN NATURAL LANGUAGE MODE) AS score**

**FROM articles WHERE MATCH (title,body) AGAINST**

**('Security implications of running MySQL as root'**

**IN NATURAL LANGUAGE MODE);**

+----+-------------------------------------+-----------------+

| id | body | score |

+----+-------------------------------------+-----------------+

| 4 | 1. Never run mysqld as root. 2. ... | 1.5219271183014 |

| 6 | When configured properly, MySQL ... | 1.3114095926285 |

+----+-------------------------------------+-----------------+

2 rows in set (0.00 sec)

A phrase that is enclosed within double quote (**"**) characters matches only rows that contain the phrase literally, as it was typed. The full-text engine splits the phrase into words and performs a search in the **FULLTEXT** index for the words. Nonword characters need not be matched exactly: Phrase searching requires only that matches contain exactly the same words as the phrase and in the same order. For example, **"test phrase"** matches **"test, phrase"**. If the phrase contains no words that are in the index, the result is empty. For example, if all words are either stopwords or shorter than the minimum length of indexed words, the result is empty.

The MySQL **FULLTEXT** implementation regards any sequence of true word characters (letters, digits, and underscores) as a word. That sequence may also contain apostrophes (**'**), but not more than one in a row. This means that **aaa'bbb** is regarded as one word, but **aaa''bbb** is regarded as two words. Apostrophes at the beginning or the end of a word are stripped by the **FULLTEXT** parser; **'aaa'bbb'** would be parsed as **aaa'bbb**.

The built-in **FULLTEXT** parser determines where words start and end by looking for certain delimiter characters; for example,  (space), **,** (comma), and **.** (period). If words are not separated by delimiters (as in, for example, Chinese), the built-in **FULLTEXT** parser cannot determine where a word begins or ends. To be able to add words or other indexed terms in such languages to a **FULLTEXT** index that uses the built-in **FULLTEXT** parser, you must preprocess them so that they are separated by some arbitrary delimiter. Alternatively, you can create **FULLTEXT** indexes using the ngram parser plugin (for Chinese, Japanese, or Korean) or the MeCab parser plugin (for Japanese).

It is possible to write a plugin that replaces the built-in full-text parser. For details, see [The MySQL Plugin API](https://dev.mysql.com/doc/extending-mysql/8.0/en/plugin-api.html). For example parser plugin source code, see the plugin/fulltext directory of a MySQL source distribution.

Some words are ignored in full-text searches:

Any word that is too short is ignored. The default minimum length of words that are found by full-text searches is three characters for **InnoDB** search indexes, or four characters for **MyISAM**. You can control the cutoff by setting a configuration option before creating the index: [**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size) configuration option for **InnoDB** search indexes, or [**ft\_min\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_min_word_len) for **MyISAM**.

**Note**

This behavior does not apply to **FULLTEXT** indexes that use the ngram parser. For the ngram parser, token length is defined by the [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) option.

Words in the stopword list are ignored. A stopword is a word such as “the” or “some” that is so common that it is considered to have zero semantic value. There is a built-in stopword list, but it can be overridden by a user-defined list. The stopword lists and related configuration options are different for **InnoDB** search indexes and **MyISAM** ones. Stopword processing is controlled by the configuration options [**innodb\_ft\_enable\_stopword**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_enable_stopword), [**innodb\_ft\_server\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_server_stopword_table), and [**innodb\_ft\_user\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_user_stopword_table) for **InnoDB** search indexes, and [**ft\_stopword\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_stopword_file) for **MyISAM** ones.

See [Section 12.10.4, “Full-Text Stopwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-stopwords) to view default stopword lists and how to change them. The default minimum word length can be changed as described in [Section 12.10.6, “Fine-Tuning MySQL Full-Text Search”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-fine-tuning).

Every correct word in the collection and in the query is weighted according to its significance in the collection or query. Thus, a word that is present in many documents has a lower weight, because it has lower semantic value in this particular collection. Conversely, if the word is rare, it receives a higher weight. The weights of the words are combined to compute the relevance of the row. This technique works best with large collections.

**MyISAM Limitation**

For very small tables, word distribution does not adequately reflect their semantic value, and this model may sometimes produce bizarre results for search indexes on **MyISAM** tables. For example, although the word “MySQL” is present in every row of the **articles** table shown earlier, a search for the word in a **MyISAM** search index produces no results:

mysql> **SELECT \* FROM articles**

**WHERE MATCH (title,body)**

**AGAINST ('MySQL' IN NATURAL LANGUAGE MODE);**

Empty set (0.00 sec)

The search result is empty because the word “MySQL” is present in at least 50% of the rows, and so is effectively treated as a stopword. This filtering technique is more suitable for large data sets, where you might not want the result set to return every second row from a 1GB table, than for small data sets where it might cause poor results for popular terms.

The 50% threshold can surprise you when you first try full-text searching to see how it works, and makes **InnoDB** tables more suited to experimentation with full-text searches. If you create a **MyISAM** table and insert only one or two rows of text into it, every word in the text occurs in at least 50% of the rows. As a result, no search returns any results until the table contains more rows. Users who need to bypass the 50% limitation can build search indexes on **InnoDB** tables, or use the boolean search mode explained in [Section 12.10.2, “Boolean Full-Text Searches”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-boolean).

### 12.10.2 Boolean Full-Text Searches

MySQL can perform boolean full-text searches using the **IN BOOLEAN MODE** modifier. With this modifier, certain characters have special meaning at the beginning or end of words in the search string. In the following query, the **+** and **-** operators indicate that a word must be present or absent, respectively, for a match to occur. Thus, the query retrieves all the rows that contain the word “MySQL” but that do not contain the word “YourSQL”:

mysql> **SELECT \* FROM articles WHERE MATCH (title,body)**

**AGAINST ('+MySQL -YourSQL' IN BOOLEAN MODE);**

+----+-----------------------+-------------------------------------+

| id | title | body |

+----+-----------------------+-------------------------------------+

| 1 | MySQL Tutorial | DBMS stands for DataBase ... |

| 2 | How To Use MySQL Well | After you went through a ... |

| 3 | Optimizing MySQL | In this tutorial, we show ... |

| 4 | 1001 MySQL Tricks | 1. Never run mysqld as root. 2. ... |

| 6 | MySQL Security | When configured properly, MySQL ... |

+----+-----------------------+-------------------------------------+

**Note**

In implementing this feature, MySQL uses what is sometimes referred to as implied Boolean logic, in which

**+** stands for **AND**

**-** stands for **NOT**

[no operator] implies **OR**

Boolean full-text searches have these characteristics:

They do not automatically sort rows in order of decreasing relevance.

**InnoDB** tables require a **FULLTEXT** index on all columns of the [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) expression to perform boolean queries. Boolean queries against a **MyISAM** search index can work even without a **FULLTEXT** index, although a search executed in this fashion would be quite slow.

The minimum and maximum word length full-text parameters apply to **FULLTEXT** indexes created using the built-in **FULLTEXT** parser and MeCab parser plugin. [**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size) and [**innodb\_ft\_max\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_max_token_size) are used for **InnoDB** search indexes. [**ft\_min\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_min_word_len) and [**ft\_max\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_max_word_len) are used for **MyISAM** search indexes.

Minimum and maximum word length full-text parameters do not apply to **FULLTEXT** indexes created using the ngram parser. ngram token size is defined by the [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) option.

The stopword list applies, controlled by [**innodb\_ft\_enable\_stopword**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_enable_stopword), [**innodb\_ft\_server\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_server_stopword_table), and [**innodb\_ft\_user\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_user_stopword_table) for **InnoDB** search indexes, and [**ft\_stopword\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_stopword_file) for **MyISAM** ones.

**InnoDB** full-text search does not support the use of multiple operators on a single search word, as in this example: **'++apple'**. Use of multiple operators on a single search word returns a syntax error to standard out. MyISAM full-text search successfully processes the same search, ignoring all operators except for the operator immediately adjacent to the search word.

**InnoDB** full-text search only supports leading plus or minus signs. For example, **InnoDB** supports **'+apple'** but does not support **'apple+'**. Specifying a trailing plus or minus sign causes **InnoDB** to report a syntax error.

**InnoDB** full-text search does not support the use of a leading plus sign with wildcard (**'+\*'**), a plus and minus sign combination (**'+-'**), or leading a plus and minus sign combination (**'+-apple'**). These invalid queries return a syntax error.

**InnoDB** full-text search does not support the use of the **@** symbol in boolean full-text searches. The **@** symbol is reserved for use by the **@distance** proximity search operator.

They do not use the 50% threshold that applies to **MyISAM** search indexes.

The boolean full-text search capability supports the following operators:

**+**

A leading or trailing plus sign indicates that this word must be present in each row that is returned. **InnoDB** only supports leading plus signs.

**-**

A leading or trailing minus sign indicates that this word must not be present in any of the rows that are returned. **InnoDB** only supports leading minus signs.

Note: The **-** operator acts only to exclude rows that are otherwise matched by other search terms. Thus, a boolean-mode search that contains only terms preceded by **-** returns an empty result. It does not return “all rows except those containing any of the excluded terms.”

(no operator)

By default (when neither **+** nor **-** is specified), the word is optional, but the rows that contain it are rated higher. This mimics the behavior of [**MATCH() AGAINST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) without the **IN BOOLEAN MODE** modifier.

**@*distance***

This operator works on **InnoDB** tables only. It tests whether two or more words all start within a specified distance from each other, measured in words. Specify the search words within a double-quoted string immediately before the **@*distance*** operator, for example, **MATCH(col1) AGAINST('"word1 word2 word3" @8' IN BOOLEAN MODE)**

**> <**

These two operators are used to change a word's contribution to the relevance value that is assigned to a row. The **>** operator increases the contribution and the **<** operator decreases it. See the example following this list.

**( )**

Parentheses group words into subexpressions. Parenthesized groups can be nested.

**~**

A leading tilde acts as a negation operator, causing the word's contribution to the row's relevance to be negative. This is useful for marking “noise” words. A row containing such a word is rated lower than others, but is not excluded altogether, as it would be with the **-** operator.

**\***

The asterisk serves as the truncation (or wildcard) operator. Unlike the other operators, it is appended to the word to be affected. Words match if they begin with the word preceding the **\*** operator.

If a word is specified with the truncation operator, it is not stripped from a boolean query, even if it is too short or a stopword. Whether a word is too short is determined from the [**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size) setting for **InnoDB** tables, or [**ft\_min\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_min_word_len) for **MyISAM** tables. These options are not applicable to **FULLTEXT** indexes that use the ngram parser.

The wildcarded word is considered as a prefix that must be present at the start of one or more words. If the minimum word length is 4, a search for **'+*word* +the\*'** could return fewer rows than a search for **'+*word* +the'**, because the second query ignores the too-short search term **the**.

**"**

A phrase that is enclosed within double quote (**"**) characters matches only rows that contain the phrase literally, as it was typed. The full-text engine splits the phrase into words and performs a search in the **FULLTEXT** index for the words. Nonword characters need not be matched exactly: Phrase searching requires only that matches contain exactly the same words as the phrase and in the same order. For example, **"test phrase"** matches **"test, phrase"**.

If the phrase contains no words that are in the index, the result is empty. The words might not be in the index because of a combination of factors: if they do not exist in the text, are stopwords, or are shorter than the minimum length of indexed words.

The following examples demonstrate some search strings that use boolean full-text operators:

**'apple banana'**

Find rows that contain at least one of the two words.

**'+apple +juice'**

Find rows that contain both words.

**'+apple macintosh'**

Find rows that contain the word “apple”, but rank rows higher if they also contain “macintosh”.

**'+apple -macintosh'**

Find rows that contain the word “apple” but not “macintosh”.

**'+apple ~macintosh'**

Find rows that contain the word “apple”, but if the row also contains the word “macintosh”, rate it lower than if row does not. This is “softer” than a search for **'+apple -macintosh'**, for which the presence of “macintosh” causes the row not to be returned at all.

**'+apple +(>turnover <strudel)'**

Find rows that contain the words “apple” and “turnover”, or “apple” and “strudel” (in any order), but rank “apple turnover” higher than “apple strudel”.

**'apple\*'**

Find rows that contain words such as “apple”, “apples”, “applesauce”, or “applet”.

**'"some words"'**

Find rows that contain the exact phrase “some words” (for example, rows that contain “some words of wisdom” but not “some noise words”). Note that the **"** characters that enclose the phrase are operator characters that delimit the phrase. They are not the quotation marks that enclose the search string itself.

#### Relevancy Rankings for InnoDB Boolean Mode Search

[**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) full-text search is modeled on the [Sphinx](http://sphinxsearch.com/) full-text search engine, and the algorithms used are based on [BM25](http://en.wikipedia.org/wiki/Okapi_BM25) and [TF-IDF](http://en.wikipedia.org/wiki/TF-IDF) ranking algorithms. For these reasons, relevancy rankings for **InnoDB** boolean full-text search may differ from [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) relevancy rankings.

**InnoDB** uses a variation of the “term frequency-inverse document frequency” (**TF-IDF**) weighting system to rank a document's relevance for a given full-text search query. The **TF-IDF** weighting is based on how frequently a word appears in a document, offset by how frequently the word appears in all documents in the collection. In other words, the more frequently a word appears in a document, and the less frequently the word appears in the document collection, the higher the document is ranked.

##### How Relevancy Ranking is Calculated

The term frequency (**TF**) value is the number of times that a word appears in a document. The inverse document frequency (**IDF**) value of a word is calculated using the following formula, where **total\_records** is the number of records in the collection, and **matching\_records** is the number of records that the search term appears in.

${IDF} = log10( ${total\_records} / ${matching\_records} )

When a document contains a word multiple times, the IDF value is multiplied by the TF value:

${TF} \* ${IDF}

Using the **TF** and **IDF** values, the relevancy ranking for a document is calculated using this formula:

${rank} = ${TF} \* ${IDF} \* ${IDF}

The formula is demonstrated in the following examples.

##### Relevancy Ranking for a Single Word Search

This example demonstrates the relevancy ranking calculation for a single-word search.

mysql> CREATE TABLE articles (

id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,

title VARCHAR(200),

body TEXT,

FULLTEXT (title,body)

) ENGINE=InnoDB;

Query OK, 0 rows affected (1.04 sec)

mysql> INSERT INTO articles (title,body) VALUES

('MySQL Tutorial','This database tutorial ...'),

("How To Use MySQL",'After you went through a ...'),

('Optimizing Your Database','In this database tutorial ...'),

('MySQL vs. YourSQL','When comparing databases ...'),

('MySQL Security','When configured properly, MySQL ...'),

('Database, Database, Database','database database database'),

('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),

('MySQL Full-Text Indexes', 'MySQL fulltext indexes use a ..');

Query OK, 8 rows affected (0.06 sec)

Records: 8 Duplicates: 0 Warnings: 0

mysql> SELECT id, title, body, MATCH (title,body) AGAINST ('database' IN BOOLEAN MODE)

AS score FROM articles ORDER BY score DESC;

+----+------------------------------+-------------------------------------+---------------------+

| id | title | body | score |

+----+------------------------------+-------------------------------------+---------------------+

| 6 | Database, Database, Database | database database database | 1.0886961221694946 |

| 3 | Optimizing Your Database | In this database tutorial ... | 0.36289870738983154 |

| 1 | MySQL Tutorial | This database tutorial ... | 0.18144935369491577 |

| 2 | How To Use MySQL | After you went through a ... | 0 |

| 4 | MySQL vs. YourSQL | When comparing databases ... | 0 |

| 5 | MySQL Security | When configured properly, MySQL ... | 0 |

| 7 | 1001 MySQL Tricks | 1. Never run mysqld as root. 2. ... | 0 |

| 8 | MySQL Full-Text Indexes | MySQL fulltext indexes use a .. | 0 |

+----+------------------------------+-------------------------------------+---------------------+

8 rows in set (0.00 sec)

There are 8 records in total, with 3 that match the “database” search term. The first record (**id 6**) contains the search term 6 times and has a relevancy ranking of **1.0886961221694946**. This ranking value is calculated using a **TF** value of 6 (the “database” search term appears 6 times in record **id 6**) and an **IDF** value of 0.42596873216370745, which is calculated as follows (where 8 is the total number of records and 3 is the number of records that the search term appears in):

${IDF} = log10( 8 / 3 ) = 0.42596873216370745

The **TF** and **IDF** values are then entered into the ranking formula:

${rank} = ${TF} \* ${IDF} \* ${IDF}

Performing the calculation in the MySQL command-line client returns a ranking value of 1.088696164686938.

mysql> SELECT 6\*log10(8/3)\*log10(8/3);

+-------------------------+

| 6\*log10(8/3)\*log10(8/3) |

+-------------------------+

| 1.088696164686938 |

+-------------------------+

1 row in set (0.00 sec)

**Note**

You may notice a slight difference in the ranking values returned by the **SELECT ... MATCH ... AGAINST** statement and the MySQL command-line client (**1.0886961221694946** versus **1.088696164686938**). The difference is due to how the casts between integers and floats/doubles are performed internally by **InnoDB** (along with related precision and rounding decisions), and how they are performed elsewhere, such as in the MySQL command-line client or other types of calculators.

##### Relevancy Ranking for a Multiple Word Search

This example demonstrates the relevancy ranking calculation for a multiple-word full-text search based on the **articles** table and data used in the previous example.

If you search on more than one word, the relevancy ranking value is a sum of the relevancy ranking value for each word, as shown in this formula:

${rank} = ${TF} \* ${IDF} \* ${IDF} + ${TF} \* ${IDF} \* ${IDF}

Performing a search on two terms ('mysql tutorial') returns the following results:

mysql> SELECT id, title, body, MATCH (title,body) AGAINST ('mysql tutorial' IN BOOLEAN MODE)

AS score FROM articles ORDER BY score DESC;

+----+------------------------------+-------------------------------------+----------------------+

| id | title | body | score |

+----+------------------------------+-------------------------------------+----------------------+

| 1 | MySQL Tutorial | This database tutorial ... | 0.7405621409416199 |

| 3 | Optimizing Your Database | In this database tutorial ... | 0.3624762296676636 |

| 5 | MySQL Security | When configured properly, MySQL ... | 0.031219376251101494 |

| 8 | MySQL Full-Text Indexes | MySQL fulltext indexes use a .. | 0.031219376251101494 |

| 2 | How To Use MySQL | After you went through a ... | 0.015609688125550747 |

| 4 | MySQL vs. YourSQL | When comparing databases ... | 0.015609688125550747 |

| 7 | 1001 MySQL Tricks | 1. Never run mysqld as root. 2. ... | 0.015609688125550747 |

| 6 | Database, Database, Database | database database database | 0 |

+----+------------------------------+-------------------------------------+----------------------+

8 rows in set (0.00 sec)

In the first record (**id 8**), 'mysql' appears once and 'tutorial' appears twice. There are six matching records for 'mysql' and two matching records for 'tutorial'. The MySQL command-line client returns the expected ranking value when inserting these values into the ranking formula for a multiple word search:

mysql> SELECT (1\*log10(8/6)\*log10(8/6)) + (2\*log10(8/2)\*log10(8/2));

+-------------------------------------------------------+

| (1\*log10(8/6)\*log10(8/6)) + (2\*log10(8/2)\*log10(8/2)) |

+-------------------------------------------------------+

| 0.7405621541938003 |

+-------------------------------------------------------+

1 row in set (0.00 sec)

**Note**

The slight difference in the ranking values returned by the **SELECT ... MATCH ... AGAINST** statement and the MySQL command-line client is explained in the preceding example.

### 12.10.3 Full-Text Searches with Query Expansion

Full-text search supports query expansion (and in particular, its variant “blind query expansion”). This is generally useful when a search phrase is too short, which often means that the user is relying on implied knowledge that the full-text search engine lacks. For example, a user searching for “database” may really mean that “MySQL”, “Oracle”, “DB2”, and “RDBMS” all are phrases that should match “databases” and should be returned, too. This is implied knowledge.

Blind query expansion (also known as automatic relevance feedback) is enabled by adding **WITH QUERY EXPANSION** or **IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION** following the search phrase. It works by performing the search twice, where the search phrase for the second search is the original search phrase concatenated with the few most highly relevant documents from the first search. Thus, if one of these documents contains the word “databases” and the word “MySQL”, the second search finds the documents that contain the word “MySQL” even if they do not contain the word “database”. The following example shows this difference:

mysql> **SELECT \* FROM articles**

**WHERE MATCH (title,body)**

**AGAINST ('database' IN NATURAL LANGUAGE MODE);**

+----+-------------------+------------------------------------------+

| id | title | body |

+----+-------------------+------------------------------------------+

| 1 | MySQL Tutorial | DBMS stands for DataBase ... |

| 5 | MySQL vs. YourSQL | In the following database comparison ... |

+----+-------------------+------------------------------------------+

2 rows in set (0.00 sec)

mysql> **SELECT \* FROM articles**

**WHERE MATCH (title,body)**

**AGAINST ('database' WITH QUERY EXPANSION);**

+----+-----------------------+------------------------------------------+

| id | title | body |

+----+-----------------------+------------------------------------------+

| 5 | MySQL vs. YourSQL | In the following database comparison ... |

| 1 | MySQL Tutorial | DBMS stands for DataBase ... |

| 3 | Optimizing MySQL | In this tutorial we show ... |

| 6 | MySQL Security | When configured properly, MySQL ... |

| 2 | How To Use MySQL Well | After you went through a ... |

| 4 | 1001 MySQL Tricks | 1. Never run mysqld as root. 2. ... |

+----+-----------------------+------------------------------------------+

6 rows in set (0.00 sec)

Another example could be searching for books by Georges Simenon about Maigret, when a user is not sure how to spell “Maigret”. A search for “Megre and the reluctant witnesses” finds only “Maigret and the Reluctant Witnesses” without query expansion. A search with query expansion finds all books with the word “Maigret” on the second pass.

**Note**

Because blind query expansion tends to increase noise significantly by returning nonrelevant documents, use it only when a search phrase is short.

### 12.10.4 Full-Text Stopwords

The stopword list is loaded and searched for full-text queries using the server character set and collation (the values of the [**character\_set\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) and [**collation\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server) system variables). False hits or misses might occur for stopword lookups if the stopword file or columns used for full-text indexing or searches have a character set or collation different from [**character\_set\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) or [**collation\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server).

Case sensitivity of stopword lookups depends on the server collation. For example, lookups are case-insensitive if the collation is **utf8mb4\_0900\_ai\_ci**, whereas lookups are case-sensitive if the collation is **utf8mb4\_0900\_as\_cs** or **utf8mb4\_bin**.

[Stopwords for InnoDB Search Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-stopwords-stopwords-for-innodb-search-indexes)

[Stopwords for MyISAM Search Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-stopwords-stopwords-for-myisam-search-indexes)

#### Stopwords for InnoDB Search Indexes

**InnoDB** has a relatively short list of default stopwords, because documents from technical, literary, and other sources often use short words as keywords or in significant phrases. For example, you might search for “to be or not to be” and expect to get a sensible result, rather than having all those words ignored.

To see the default **InnoDB** stopword list, query the [**INFORMATION\_SCHEMA.INNODB\_FT\_DEFAULT\_STOPWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-default-stopword-table) table.

mysql> SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FT\_DEFAULT\_STOPWORD;

+-------+

| value |

+-------+

| a |

| about |

| an |

| are |

| as |

| at |

| be |

| by |

| com |

| de |

| en |

| for |

| from |

| how |

| i |

| in |

| is |

| it |

| la |

| of |

| on |

| or |

| that |

| the |

| this |

| to |

| was |

| what |

| when |

| where |

| who |

| will |

| with |

| und |

| the |

| www |

+-------+

36 rows in set (0.00 sec)

To define your own stopword list for all **InnoDB** tables, define a table with the same structure as the [**INNODB\_FT\_DEFAULT\_STOPWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-default-stopword-table) table, populate it with stopwords, and set the value of the [**innodb\_ft\_server\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_server_stopword_table) option to a value in the form ***db\_name*/*table\_name*** before creating the full-text index. The stopword table must have a single [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column named **value**. The following example demonstrates creating and configuring a new global stopword table for **InnoDB**.

-- Create a new stopword table

mysql> CREATE TABLE my\_stopwords(value VARCHAR(30)) ENGINE = INNODB;

Query OK, 0 rows affected (0.01 sec)

-- Insert stopwords (for simplicity, a single stopword is used in this example)

mysql> INSERT INTO my\_stopwords(value) VALUES ('Ishmael');

Query OK, 1 row affected (0.00 sec)

-- Create the table

mysql> CREATE TABLE opening\_lines (

id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,

opening\_line TEXT(500),

author VARCHAR(200),

title VARCHAR(200)

) ENGINE=InnoDB;

Query OK, 0 rows affected (0.01 sec)

-- Insert data into the table

mysql> INSERT INTO opening\_lines(opening\_line,author,title) VALUES

('Call me Ishmael.','Herman Melville','Moby-Dick'),

('A screaming comes across the sky.','Thomas Pynchon','Gravity\'s Rainbow'),

('I am an invisible man.','Ralph Ellison','Invisible Man'),

('Where now? Who now? When now?','Samuel Beckett','The Unnamable'),

('It was love at first sight.','Joseph Heller','Catch-22'),

('All this happened, more or less.','Kurt Vonnegut','Slaughterhouse-Five'),

('Mrs. Dalloway said she would buy the flowers herself.','Virginia Woolf','Mrs. Dalloway'),

('It was a pleasure to burn.','Ray Bradbury','Fahrenheit 451');

Query OK, 8 rows affected (0.00 sec)

Records: 8 Duplicates: 0 Warnings: 0

-- Set the innodb\_ft\_server\_stopword\_table option to the new stopword table

mysql> SET GLOBAL innodb\_ft\_server\_stopword\_table = 'test/my\_stopwords';

Query OK, 0 rows affected (0.00 sec)

-- Create the full-text index (which rebuilds the table if no FTS\_DOC\_ID column is defined)

mysql> CREATE FULLTEXT INDEX idx ON opening\_lines(opening\_line);

Query OK, 0 rows affected, 1 warning (1.17 sec)

Records: 0 Duplicates: 0 Warnings: 1

Verify that the specified stopword ('Ishmael') does not appear by querying the words in [**INFORMATION\_SCHEMA.INNODB\_FT\_INDEX\_TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-table-table).

**Note**

By default, words less than 3 characters in length or greater than 84 characters in length do not appear in an **InnoDB** full-text search index. Maximum and minimum word length values are configurable using the [**innodb\_ft\_max\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_max_token_size) and [**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size) variables. This default behavior does not apply to the ngram parser plugin. ngram token size is defined by the [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) option.

mysql> SET GLOBAL innodb\_ft\_aux\_table='test/opening\_lines';

Query OK, 0 rows affected (0.00 sec)

mysql> SELECT word FROM INFORMATION\_SCHEMA.INNODB\_FT\_INDEX\_TABLE LIMIT 15;

+-----------+

| word |

+-----------+

| across |

| all |

| burn |

| buy |

| call |

| comes |

| dalloway |

| first |

| flowers |

| happened |

| herself |

| invisible |

| less |

| love |

| man |

+-----------+

15 rows in set (0.00 sec)

To create stopword lists on a table-by-table basis, create other stopword tables and use the [**innodb\_ft\_user\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_user_stopword_table) option to specify the stopword table that you want to use before you create the full-text index.

#### Stopwords for MyISAM Search Indexes

The stopword file is loaded and searched using **latin1** if **character\_set\_server** is **ucs2**, **utf16**, **utf16le**, or **utf32**.

To override the default stopword list for MyISAM tables, set the [**ft\_stopword\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_stopword_file) system variable. (See [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).) The variable value should be the path name of the file containing the stopword list, or the empty string to disable stopword filtering. The server looks for the file in the data directory unless an absolute path name is given to specify a different directory. After changing the value of this variable or the contents of the stopword file, restart the server and rebuild your **FULLTEXT** indexes.

The stopword list is free-form, separating stopwords with any nonalphanumeric character such as newline, space, or comma. Exceptions are the underscore character (**\_**) and a single apostrophe (**'**) which are treated as part of a word. The character set of the stopword list is the server's default character set; see [Section 10.3.2, “Server Character Set and Collation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-server).

The following list shows the default stopwords for **MyISAM** search indexes. In a MySQL source distribution, you can find this list in the storage/myisam/ft\_static.c file.

a's able about above according

accordingly across actually after afterwards

again against ain't all allow

allows almost alone along already

also although always am among

amongst an and another any

anybody anyhow anyone anything anyway

anyways anywhere apart appear appreciate

appropriate are aren't around as

aside ask asking associated at

available away awfully be became

because become becomes becoming been

before beforehand behind being believe

below beside besides best better

between beyond both brief but

by c'mon c's came can

can't cannot cant cause causes

certain certainly changes clearly co

com come comes concerning consequently

consider considering contain containing contains

corresponding could couldn't course currently

definitely described despite did didn't

different do does doesn't doing

don't done down downwards during

each edu eg eight either

else elsewhere enough entirely especially

et etc even ever every

everybody everyone everything everywhere ex

exactly example except far few

fifth first five followed following

follows for former formerly forth

four from further furthermore get

gets getting given gives go

goes going gone got gotten

greetings had hadn't happens hardly

has hasn't have haven't having

he he's hello help hence

her here here's hereafter hereby

herein hereupon hers herself hi

him himself his hither hopefully

how howbeit however i'd i'll

i'm i've ie if ignored

immediate in inasmuch inc indeed

indicate indicated indicates inner insofar

instead into inward is isn't

it it'd it'll it's its

itself just keep keeps kept

know known knows last lately

later latter latterly least less

lest let let's like liked

likely little look looking looks

ltd mainly many may maybe

me mean meanwhile merely might

more moreover most mostly much

must my myself name namely

nd near nearly necessary need

needs neither never nevertheless new

next nine no nobody non

none noone nor normally not

nothing novel now nowhere obviously

of off often oh ok

okay old on once one

ones only onto or other

others otherwise ought our ours

ourselves out outside over overall

own particular particularly per perhaps

placed please plus possible presumably

probably provides que quite qv

rather rd re really reasonably

regarding regardless regards relatively respectively

right said same saw say

saying says second secondly see

seeing seem seemed seeming seems

seen self selves sensible sent

serious seriously seven several shall

she should shouldn't since six

so some somebody somehow someone

something sometime sometimes somewhat somewhere

soon sorry specified specify specifying

still sub such sup sure

t's take taken tell tends

th than thank thanks thanx

that that's thats the their

theirs them themselves then thence

there there's thereafter thereby therefore

therein theres thereupon these they

they'd they'll they're they've think

third this thorough thoroughly those

though three through throughout thru

thus to together too took

toward towards tried tries truly

try trying twice two un

under unfortunately unless unlikely until

unto up upon us use

used useful uses using usually

value various very via viz

vs want wants was wasn't

way we we'd we'll we're

we've welcome well went were

weren't what what's whatever when

whence whenever where where's whereafter

whereas whereby wherein whereupon wherever

whether which while whither who

who's whoever whole whom whose

why will willing wish with

within without won't wonder would

wouldn't yes yet you you'd

you'll you're you've your yours

yourself yourselves zero

### 12.10.5 Full-Text Restrictions

Full-text searches are supported for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables only.

Full-text searches are not supported for partitioned tables. See [Section 24.6, “Restrictions and Limitations on Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-limitations).

Full-text searches can be used with most multibyte character sets. The exception is that for Unicode, the **utf8** character set can be used, but not the **ucs2** character set. Although **FULLTEXT** indexes on **ucs2** columns cannot be used, you can perform **IN BOOLEAN MODE** searches on a **ucs2** column that has no such index.

The remarks for **utf8** also apply to **utf8mb4**, and the remarks for **ucs2** also apply to **utf16**, **utf16le**, and **utf32**.

Ideographic languages such as Chinese and Japanese do not have word delimiters. Therefore, the built-in full-text parser cannot determine where words begin and end in these and other such languages.

A character-based ngram full-text parser that supports Chinese, Japanese, and Korean (CJK), and a word-based MeCab parser plugin that supports Japanese are provided for use with **InnoDB** and **MyISAM** tables.

Although the use of multiple character sets within a single table is supported, all columns in a **FULLTEXT** index must use the same character set and collation.

The [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) column list must match exactly the column list in some **FULLTEXT** index definition for the table, unless this [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) is **IN BOOLEAN MODE** on a **MyISAM** table. For **MyISAM** tables, boolean-mode searches can be done on nonindexed columns, although they are likely to be slow.

The argument to **AGAINST()** must be a string value that is constant during query evaluation. This rules out, for example, a table column because that can differ for each row.

Index hints are more limited for **FULLTEXT** searches than for non-**FULLTEXT** searches. See [Section 8.9.4, “Index Hints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#index-hints).

For **InnoDB**, all DML operations ([**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete)) involving columns with full-text indexes are processed at transaction commit time. For example, for an **INSERT** operation, an inserted string is tokenized and decomposed into individual words. The individual words are then added to full-text index tables when the transaction is committed. As a result, full-text searches only return committed data.

The '%' character is not a supported wildcard character for full-text searches.

### 12.10.6 Fine-Tuning MySQL Full-Text Search

MySQL's full-text search capability has few user-tunable parameters. You can exert more control over full-text searching behavior if you have a MySQL source distribution because some changes require source code modifications. See [Section 2.9, “Installing MySQL from Source”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#source-installation).

Full-text search is carefully tuned for effectiveness. Modifying the default behavior in most cases can actually decrease effectiveness. Do not alter the MySQL sources unless you know what you are doing.

Most full-text variables described in this section must be set at server startup time. A server restart is required to change them; they cannot be modified while the server is running.

Some variable changes require that you rebuild the **FULLTEXT** indexes in your tables. Instructions for doing so are given later in this section.

[Configuring Minimum and Maximum Word Length](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-word-length)

[Configuring the Natural Language Search Threshold](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-natural-language-threshold)

[Modifying Boolean Full-Text Search Operators](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-modify-boolean-operators)

[Character Set Modifications](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-modify-character-set)

[Rebuilding InnoDB Full-Text Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-rebuild-innodb-indexes)

[Optimizing InnoDB Full-Text Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-optimize)

[Rebuilding MyISAM Full-Text Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-rebuild-myisam-indexes)

#### Configuring Minimum and Maximum Word Length

The minimum and maximum lengths of words to be indexed are defined by the [**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size) and [**innodb\_ft\_max\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_max_token_size) for **InnoDB** search indexes, and [**ft\_min\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_min_word_len) and [**ft\_max\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_max_word_len) for **MyISAM** ones.

**Note**

Minimum and maximum word length full-text parameters do not apply to **FULLTEXT** indexes created using the ngram parser. ngram token size is defined by the [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) option.

After changing any of these options, rebuild your **FULLTEXT** indexes for the change to take effect. For example, to make two-character words searchable, you could put the following lines in an option file:

[mysqld]

innodb\_ft\_min\_token\_size=2

ft\_min\_word\_len=2

Then restart the server and rebuild your **FULLTEXT** indexes. For **MyISAM** tables, note the remarks regarding [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) in the instructions that follow for rebuilding **MyISAM** full-text indexes.

#### Configuring the Natural Language Search Threshold

For **MyISAM** search indexes, the 50% threshold for natural language searches is determined by the particular weighting scheme chosen. To disable it, look for the following line in storage/myisam/ftdefs.h:

#define GWS\_IN\_USE GWS\_PROB

Change that line to this:

#define GWS\_IN\_USE GWS\_FREQ

Then recompile MySQL. There is no need to rebuild the indexes in this case.

**Note**

By making this change, you severely decrease MySQL's ability to provide adequate relevance values for the [**MATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) function. If you really need to search for such common words, it would be better to search using **IN BOOLEAN MODE** instead, which does not observe the 50% threshold.

#### Modifying Boolean Full-Text Search Operators

To change the operators used for boolean full-text searches on **MyISAM** tables, set the [**ft\_boolean\_syntax**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_boolean_syntax) system variable. (**InnoDB** does not have an equivalent setting.) This variable can be changed while the server is running, but you must have privileges sufficient to set global system variables (see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges)). No rebuilding of indexes is necessary in this case.

#### Character Set Modifications

For the built-in full-text parser, you can change the set of characters that are considered word characters in several ways, as described in the following list. After making the modification, rebuild the indexes for each table that contains any **FULLTEXT** indexes. Suppose that you want to treat the hyphen character ('-') as a word character. Use one of these methods:

Modify the MySQL source: In storage/innobase/handler/ha\_innodb.cc (for **InnoDB**), or in storage/myisam/ftdefs.h (for **MyISAM**), see the **true\_word\_char()** and **misc\_word\_char()** macros. Add **'-'** to one of those macros and recompile MySQL.

Modify a character set file: This requires no recompilation. The **true\_word\_char()** macro uses a “character type” table to distinguish letters and numbers from other characters. . You can edit the contents of the **<ctype><map>** array in one of the character set XML files to specify that **'-'** is a “letter.” Then use the given character set for your **FULLTEXT** indexes. For information about the **<ctype><map>** array format, see [Section 10.13.1, “Character Definition Arrays”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#character-arrays).

Add a new collation for the character set used by the indexed columns, and alter the columns to use that collation. For general information about adding collations, see [Section 10.14, “Adding a Collation to a Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation). For an example specific to full-text indexing, see [Section 12.10.7, “Adding a User-Defined Collation for Full-Text Indexing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#full-text-adding-collation).

#### Rebuilding InnoDB Full-Text Indexes

For the changes to take effect, **FULLTEXT** indexes must be rebuilt after modifying any of the following full-text index variables: [**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size); [**innodb\_ft\_max\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_max_token_size); [**innodb\_ft\_server\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_server_stopword_table); [**innodb\_ft\_user\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_user_stopword_table); [**innodb\_ft\_enable\_stopword**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_enable_stopword); [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size). Modifying [**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size), [**innodb\_ft\_max\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_max_token_size), or [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) requires restarting the server.

To rebuild **FULLTEXT** indexes for an **InnoDB** table, use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) with the **DROP INDEX** and **ADD INDEX** options to drop and re-create each index.

#### Optimizing InnoDB Full-Text Indexes

Running [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) on a table with a full-text index rebuilds the full-text index, removing deleted Document IDs and consolidating multiple entries for the same word, where possible.

To optimize a full-text index, enable [**innodb\_optimize\_fulltext\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_optimize_fulltext_only) and run **OPTIMIZE TABLE**.

mysql> set GLOBAL innodb\_optimize\_fulltext\_only=ON;

Query OK, 0 rows affected (0.01 sec)

mysql> OPTIMIZE TABLE opening\_lines;

+--------------------+----------+----------+----------+

| Table | Op | Msg\_type | Msg\_text |

+--------------------+----------+----------+----------+

| test.opening\_lines | optimize | status | OK |

+--------------------+----------+----------+----------+

1 row in set (0.01 sec)

To avoid lengthy rebuild times for full-text indexes on large tables, you can use the [**innodb\_ft\_num\_word\_optimize**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_num_word_optimize) option to perform the optimization in stages. The **innodb\_ft\_num\_word\_optimize** option defines the number of words that are optimized each time [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is run. The default setting is 2000, which means that 2000 words are optimized each time [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is run. Subsequent [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) operations continue from where the preceding [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) operation ended.

#### Rebuilding MyISAM Full-Text Indexes

If you modify full-text variables that affect indexing ([**ft\_min\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_min_word_len), [**ft\_max\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_max_word_len), or [**ft\_stopword\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_stopword_file)), or if you change the stopword file itself, you must rebuild your **FULLTEXT** indexes after making the changes and restarting the server.

To rebuild the **FULLTEXT** indexes for a **MyISAM** table, it is sufficient to do a **QUICK** repair operation:

mysql> **REPAIR TABLE *tbl\_name* QUICK;**

Alternatively, use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) as just described. In some cases, this may be faster than a repair operation.

Each table that contains any **FULLTEXT** index must be repaired as just shown. Otherwise, queries for the table may yield incorrect results, and modifications to the table causes the server to see the table as corrupt and in need of repair.

If you use [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) to perform an operation that modifies **MyISAM**table indexes (such as repair or analyze), the **FULLTEXT** indexes are rebuilt using the default full-text parameter values for minimum word length, maximum word length, and stopword file unless you specify otherwise. This can result in queries failing.

The problem occurs because these parameters are known only by the server. They are not stored in **MyISAM** index files. To avoid the problem if you have modified the minimum or maximum word length or stopword file values used by the server, specify the same [**ft\_min\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_min_word_len), [**ft\_max\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_max_word_len), and [**ft\_stopword\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_stopword_file) values for [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) that you use for [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld). For example, if you have set the minimum word length to 3, you can repair a table with [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) like this:

myisamchk --recover --ft\_min\_word\_len=3 ***tbl\_name***.MYI

To ensure that [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) and the server use the same values for full-text parameters, place each one in both the **[mysqld]** and **[myisamchk]** sections of an option file:

[mysqld]

ft\_min\_word\_len=3

[myisamchk]

ft\_min\_word\_len=3

An alternative to using [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) for **MyISAM** table index modification is to use the [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table), [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table), [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table), or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements. These statements are performed by the server, which knows the proper full-text parameter values to use.

### 12.10.7 Adding a User-Defined Collation for Full-Text Indexing

This section describes how to add a user-defined collation for full-text searches using the built-in full-text parser. The sample collation is like **latin1\_swedish\_ci** but treats the **'-'** character as a letter rather than as a punctuation character so that it can be indexed as a word character. General information about adding collations is given in [Section 10.14, “Adding a Collation to a Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation); it is assumed that you have read it and are familiar with the files involved.

To add a collation for full-text indexing, use the following procedure. The instructions here add a collation for a simple character set, which as discussed in [Section 10.14, “Adding a Collation to a Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation), can be created using a configuration file that describes the character set properties. For a complex character set such as Unicode, create collations using C source files that describe the character set properties.

Add a collation to the Index.xml file. The permitted range of IDs for user-defined collations is given in [Section 10.14.2, “Choosing a Collation ID”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-choosing-id). The ID must be unused, so choose a value different from 1025 if that ID is already taken on your system.

<charset name="latin1">

...

<collation name="latin1\_fulltext\_ci" id="1025"/>

</charset>

Declare the sort order for the collation in the latin1.xml file. In this case, the order can be copied from **latin1\_swedish\_ci**:

<collation name="latin1\_fulltext\_ci">

<map>

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F

20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F

30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F

40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F

50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F

60 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F

50 51 52 53 54 55 56 57 58 59 5A 7B 7C 7D 7E 7F

80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F

90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F

A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF

B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF

41 41 41 41 5C 5B 5C 43 45 45 45 45 49 49 49 49

44 4E 4F 4F 4F 4F 5D D7 D8 55 55 55 59 59 DE DF

41 41 41 41 5C 5B 5C 43 45 45 45 45 49 49 49 49

44 4E 4F 4F 4F 4F 5D F7 D8 55 55 55 59 59 DE FF

</map>

</collation>

Modify the **ctype** array in latin1.xml. Change the value corresponding to 0x2D (which is the code for the **'-'** character) from 10 (punctuation) to 01 (uppercase letter). In the following array, this is the element in the fourth row down, third value from the end.

<ctype>

<map>

00

20 20 20 20 20 20 20 20 20 28 28 28 28 28 20 20

20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

48 10 10 10 10 10 10 10 10 10 10 10 10 ***01*** 10 10

84 84 84 84 84 84 84 84 84 84 10 10 10 10 10 10

10 81 81 81 81 81 81 01 01 01 01 01 01 01 01 01

01 01 01 01 01 01 01 01 01 01 01 10 10 10 10 10

10 82 82 82 82 82 82 02 02 02 02 02 02 02 02 02

02 02 02 02 02 02 02 02 02 02 02 10 10 10 10 20

10 00 10 02 10 10 10 10 10 10 01 10 01 00 01 00

00 10 10 10 10 10 10 10 10 10 02 10 02 00 02 01

48 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10

10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10

01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01

01 01 01 01 01 01 01 10 01 01 01 01 01 01 01 02

02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02

02 02 02 02 02 02 02 10 02 02 02 02 02 02 02 02

</map>

</ctype>

Restart the server.

To employ the new collation, include it in the definition of columns that are to use it:

mysql> **DROP TABLE IF EXISTS t1;**

Query OK, 0 rows affected (0.13 sec)

mysql> **CREATE TABLE t1 (**

**a TEXT CHARACTER SET latin1 COLLATE latin1\_fulltext\_ci,**

**FULLTEXT INDEX(a)**

**) ENGINE=InnoDB;**

Query OK, 0 rows affected (0.47 sec)

Test the collation to verify that hyphen is considered as a word character:

mysql> **INSERT INTO t1 VALUEs ('----'),('....'),('abcd');**

Query OK, 3 rows affected (0.22 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> **SELECT \* FROM t1 WHERE MATCH a AGAINST ('----' IN BOOLEAN MODE);**

+------+

| a |

+------+

| ---- |

+------+

1 row in set (0.00 sec)

### 12.10.8 ngram Full-Text Parser

The built-in MySQL full-text parser uses the white space between words as a delimiter to determine where words begin and end, which is a limitation when working with ideographic languages that do not use word delimiters. To address this limitation, MySQL provides an ngram full-text parser that supports Chinese, Japanese, and Korean (CJK). The ngram full-text parser is supported for use with [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine).

**Note**

MySQL also provides a MeCab full-text parser plugin for Japanese, which tokenizes documents into meaningful words. For more information, see [Section 12.10.9, “MeCab Full-Text Parser Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search-mecab).

An ngram is a contiguous sequence of ***n*** characters from a given sequence of text. The ngram parser tokenizes a sequence of text into a contiguous sequence of ***n*** characters. For example, you can tokenize “abcd” for different values of ***n*** using the ngram full-text parser.

n=1: 'a', 'b', 'c', 'd'

n=2: 'ab', 'bc', 'cd'

n=3: 'abc', 'bcd'

n=4: 'abcd'

The ngram full-text parser is a built-in server plugin. As with other built-in server plugins, it is automatically loaded when the server is started.

The full-text search syntax described in [Section 12.10, “Full-Text Search Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search) applies to the ngram parser plugin. Differences in parsing behavior are described in this section. Full-text-related configuration options, except for minimum and maximum word length options ([**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size), [**innodb\_ft\_max\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_max_token_size), [**ft\_min\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_min_word_len), [**ft\_max\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_max_word_len)) are also applicable.

#### Configuring ngram Token Size

The ngram parser has a default ngram token size of 2 (bigram). For example, with a token size of 2, the ngram parser parses the string “abc def” into four tokens: “ab”, “bc”, “de” and “ef”.

ngram token size is configurable using the [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) configuration option, which has a minimum value of 1 and maximum value of 10.

Typically, [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) is set to the size of the largest token that you want to search for. If you only intend to search for single characters, set [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) to 1. A smaller token size produces a smaller full-text search index, and faster searches. If you need to search for words comprised of more than one character, set [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) accordingly. For example, “Happy Birthday” is “生日快乐” in simplified Chinese, where “生日” is “birthday”, and “快乐” translates as “happy”. To search on two-character words such as these, set [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) to a value of 2 or higher.

As a read-only variable, [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) may only be set as part of a startup string or in a configuration file:

Startup string:

mysqld --ngram\_token\_size=2

Configuration file:

[mysqld]

ngram\_token\_size=2

**Note**

The following minimum and maximum word length configuration options are ignored for **FULLTEXT** indexes that use the ngram parser: [**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size), [**innodb\_ft\_max\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_max_token_size), [**ft\_min\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_min_word_len), and [**ft\_max\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_max_word_len).

#### Creating a FULLTEXT Index that Uses the ngram Parser

To create a **FULLTEXT** index that uses the ngram parser, specify **WITH PARSER ngram** with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), or [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index).

The following example demonstrates creating a table with an **ngram** **FULLTEXT** index, inserting sample data (Simplified Chinese text), and viewing tokenized data in the [**INFORMATION\_SCHEMA.INNODB\_FT\_INDEX\_CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-cache-table) table.

mysql> USE test;

mysql> CREATE TABLE articles (

id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,

title VARCHAR(200),

body TEXT,

FULLTEXT (title,body) WITH PARSER ngram

) ENGINE=InnoDB CHARACTER SET utf8mb4;

mysql> SET NAMES utf8mb4;

INSERT INTO articles (title,body) VALUES

('数据库管理','在本教程中我将向你展示如何管理数据库'),

('数据库应用开发','学习开发数据库应用程序');

mysql> SET GLOBAL innodb\_ft\_aux\_table="test/articles";

mysql> SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FT\_INDEX\_CACHE ORDER BY doc\_id, position;

To add a **FULLTEXT** index to an existing table, you can use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) or [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index). For example:

CREATE TABLE articles (

id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,

title VARCHAR(200),

body TEXT

) ENGINE=InnoDB CHARACTER SET utf8;

ALTER TABLE articles ADD FULLTEXT INDEX ft\_index (title,body) WITH PARSER ngram;

# Or:

CREATE FULLTEXT INDEX ft\_index ON articles (title,body) WITH PARSER ngram;

#### ngram Parser Space Handling

The ngram parser eliminates spaces when parsing. For example:

“ab cd” is parsed to “ab”, “cd”

“a bc” is parsed to “bc”

#### ngram Parser Stopword Handling

The built-in MySQL full-text parser compares words to entries in the stopword list. If a word is equal to an entry in the stopword list, the word is excluded from the index. For the ngram parser, stopword handling is performed differently. Instead of excluding tokens that are equal to entries in the stopword list, the ngram parser excludes tokens that contain stopwords. For example, assuming [**ngram\_token\_size=2**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size), a document that contains “a,b” is parsed to “a,” and “,b”. If a comma (“,”) is defined as a stopword, both “a,” and “,b” are excluded from the index because they contain a comma.

By default, the ngram parser uses the default stopword list, which contains a list of English stopwords. For a stopword list applicable to Chinese, Japanese, or Korean, you must create your own. For information about creating a stopword list, see [Section 12.10.4, “Full-Text Stopwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-stopwords).

Stopwords greater in length than [**ngram\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size) are ignored.

#### ngram Parser Term Search

For natural language mode search, the search term is converted to a union of ngram terms. For example, the string “abc” (assuming [**ngram\_token\_size=2**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size)) is converted to “ab bc”. Given two documents, one containing “ab” and the other containing “abc”, the search term “ab bc” matches both documents.

For boolean mode search, the search term is converted to an ngram phrase search. For example, the string 'abc' (assuming [**ngram\_token\_size=2**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size)) is converted to '“ab bc”'. Given two documents, one containing 'ab' and the other containing 'abc', the search phrase '“ab bc”' only matches the document containing 'abc'.

#### ngram Parser Wildcard Search

Because an ngram **FULLTEXT** index contains only ngrams, and does not contain information about the beginning of terms, wildcard searches may return unexpected results. The following behaviors apply to wildcard searches using ngram **FULLTEXT** search indexes:

If the prefix term of a wildcard search is shorter than ngram token size, the query returns all indexed rows that contain ngram tokens starting with the prefix term. For example, assuming [**ngram\_token\_size=2**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size), a search on “a\*” returns all rows starting with “a”.

If the prefix term of a wildcard search is longer than ngram token size, the prefix term is converted to an ngram phrase and the wildcard operator is ignored. For example, assuming [**ngram\_token\_size=2**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ngram_token_size), an “abc\*” wildcard search is converted to “ab bc”.

#### ngram Parser Phrase Search

Phrase searches are converted to ngram phrase searches. For example, The search phrase “abc” is converted to “ab bc”, which returns documents containing “abc” and “ab bc”.

The search phrase “abc def” is converted to “ab bc de ef”, which returns documents containing “abc def” and “ab bc de ef”. A document that contains “abcdef” is not returned.

### 12.10.9 MeCab Full-Text Parser Plugin

The built-in MySQL full-text parser uses the white space between words as a delimiter to determine where words begin and end, which is a limitation when working with ideographic languages that do not use word delimiters. To address this limitation for Japanese, MySQL provides a MeCab full-text parser plugin. The MeCab full-text parser plugin is supported for use with [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine).

**Note**

MySQL also provides an ngram full-text parser plugin that supports Japanese. For more information, see [Section 12.10.8, “ngram Full-Text Parser”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search-ngram).

The MeCab full-text parser plugin is a full-text parser plugin for Japanese that tokenizes a sequence of text into meaningful words. For example, MeCab tokenizes “データベース管理” (“Database Management”) into “データベース” (“Database”) and “管理” (“Management”). By comparison, the [ngram](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search-ngram) full-text parser tokenizes text into a contiguous sequence of ***n*** characters, where ***n*** represents a number between 1 and 10.

In addition to tokenizing text into meaningful words, MeCab indexes are typically smaller than ngram indexes, and MeCab full-text searches are generally faster. One drawback is that it may take longer for the MeCab full-text parser to tokenize documents, compared to the ngram full-text parser.

The full-text search syntax described in [Section 12.10, “Full-Text Search Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search) applies to the MeCab parser plugin. Differences in parsing behavior are described in this section. Full-text related configuration options are also applicable.

For additional information about the MeCab parser, refer to the [MeCab: Yet Another Part-of-Speech and Morphological Analyzer](http://taku910.github.io/mecab/) project on Github.

#### Installing the MeCab Parser Plugin

The MeCab parser plugin requires mecab and mecab-ipadic.

On supported Fedora, Debian and Ubuntu platforms (except Ubuntu 12.04 where the system mecab version is too old), MySQL dynamically links to the system mecab installation if it is installed to the default location. On other supported Unix-like platforms, libmecab.so is statically linked in libpluginmecab.so, which is located in the MySQL plugin directory. mecab-ipadic is included in MySQL binaries and is located in ***MYSQL\_HOME***\lib\mecab.

You can install mecab and mecab-ipadic using a native package management utility (on Fedora, Debian, and Ubuntu), or you can build mecab and mecab-ipadic from source. For information about installing mecab and mecab-ipadic using a native package management utility, see [Installing MeCab From a Binary Distribution (Optional)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#install-mecab-binary). If you want to build mecab and **mecab-ipadic** from source, see [Building MeCab From Source (Optional)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#build-mecab-from-source).

On Windows, libmecab.dll is found in the MySQL bin directory. mecab-ipadic is located in ***MYSQL\_HOME***/lib/mecab.

To install and configure the MeCab parser plugin, perform the following steps:

In the MySQL configuration file, set the [**mecab\_rc\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_mecab_rc_file) configuration option to the location of the **mecabrc** configuration file, which is the configuration file for MeCab. If you are using the MeCab package distributed with MySQL, the **mecabrc** file is located in MYSQL\_HOME/lib/mecab/etc/.

[mysqld]

loose-mecab-rc-file=MYSQL\_HOME/lib/mecab/etc/mecabrc

The **loose** prefix is an [option modifier](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option-modifiers). The [**mecab\_rc\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_mecab_rc_file) option is not recognized by MySQL until the MeCaB parser plugin is installed but it must be set before attempting to install the MeCaB parser plugin. The **loose** prefix allows you restart MySQL without encountering an error due to an unrecognized variable.

If you use your own MeCab installation, or build MeCab from source, the location of the mecabrc configuration file may differ.

For information about the MySQL configuration file and its location, see [Section 4.2.2.2, “Using Option Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option-files).

Also in the MySQL configuration file, set the minimum token size to 1 or 2, which are the values recommended for use with the MeCab parser. For **InnoDB** tables, minimum token size is defined by the [**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size) configuration option, which has a default value of 3. For **MyISAM** tables, minimum token size is defined by [**ft\_min\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_min_word_len), which has a default value of 4.

[mysqld]

innodb\_ft\_min\_token\_size=1

Modify the mecabrc configuration file to specify the dictionary you want to use. The mecab-ipadic package distributed with MySQL binaries includes three dictionaries (**ipadic\_euc-jp**, **ipadic\_sjis**, and **ipadic\_utf-8**). The mecabrc configuration file packaged with MySQL contains and entry similar to the following:

dicdir = /path/to/mysql/lib/mecab/lib/mecab/dic/ipadic\_euc-jp

To use the ipadic\_utf-8 dictionary, for example, modify the entry as follows:

dicdir=***MYSQL\_HOME***/lib/mecab/dic/ipadic\_utf-8

If you are using your own MeCab installation or have built MeCab from source, the default **dicdir** entry in the mecabrc file is likely to differ, as are the dictionaries and their location.

**Note**

After the MeCab parser plugin is installed, you can use the [**mecab\_charset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_mecab_charset) status variable to view the character set used with MeCab. The three MeCab dictionaries provided with the MySQL binary support the following character sets.

The **ipadic\_euc-jp** dictionary supports the **ujis** and **eucjpms** character sets.

The **ipadic\_sjis** dictionary supports the **sjis** and **cp932** character sets.

The **ipadic\_utf-8** dictionary supports the **utf8** and **utf8mb4** character sets.

[**mecab\_charset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_mecab_charset) only reports the first supported character set. For example, the **ipadic\_utf-8** dictionary supports both **utf8** and **utf8mb4**. [**mecab\_charset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_mecab_charset) always reports **utf8** when this dictionary is in use.

Restart MySQL.

Install the MeCab parser plugin:

The MeCab parser plugin is installed using [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) syntax. The plugin name is mecab, and the shared library name is libpluginmecab.so. For additional information about installing plugins, see [Section 5.6.1, “Installing and Uninstalling Plugins”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#plugin-loading).

INSTALL PLUGIN mecab SONAME 'libpluginmecab.so';

Once installed, the MeCab parser plugin loads at every normal MySQL restart.

Verify that the MeCab parser plugin is loaded using the [**SHOW PLUGINS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-plugins) statement.

mysql> SHOW PLUGINS;

A **mecab** plugin should appear in the list of plugins.

#### Creating a FULLTEXT Index that uses the MeCab Parser

To create a **FULLTEXT** index that uses the mecab parser, specify **WITH PARSER ngram** with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), or [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index).

This example demonstrates creating a table with a **mecab** **FULLTEXT** index, inserting sample data, and viewing tokenized data in the [**INFORMATION\_SCHEMA.INNODB\_FT\_INDEX\_CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-cache-table) table:

mysql> USE test;

mysql> CREATE TABLE articles (

id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,

title VARCHAR(200),

body TEXT,

FULLTEXT (title,body) WITH PARSER mecab

) ENGINE=InnoDB CHARACTER SET utf8;

mysql> SET NAMES utf8;

mysql> INSERT INTO articles (title,body) VALUES

('データベース管理','このチュートリアルでは、私はどのようにデータベースを管理する方法を紹介します'),

('データベースアプリケーション開発','データベースアプリケーションを開発することを学ぶ');

mysql> SET GLOBAL innodb\_ft\_aux\_table="test/articles";

mysql> SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FT\_INDEX\_CACHE ORDER BY doc\_id, position;

To add a **FULLTEXT** index to an existing table, you can use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) or [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index). For example:

CREATE TABLE articles (

id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,

title VARCHAR(200),

body TEXT

) ENGINE=InnoDB CHARACTER SET utf8;

ALTER TABLE articles ADD FULLTEXT INDEX ft\_index (title,body) WITH PARSER mecab;

# Or:

CREATE FULLTEXT INDEX ft\_index ON articles (title,body) WITH PARSER mecab;

#### MeCab Parser Space Handling

The MeCab parser uses spaces as separators in query strings. For example, the MeCab parser tokenizes データベース管理 as データベース and 管理.

#### MeCab Parser Stopword Handling

By default, the MeCab parser uses the default stopword list, which contains a short list of English stopwords. For a stopword list applicable to Japanese, you must create your own. For information about creating stopword lists, see [Section 12.10.4, “Full-Text Stopwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-stopwords).

#### MeCab Parser Term Search

For natural language mode search, the search term is converted to a union of tokens. For example, データベース管理 is converted to データベース 管理.

SELECT COUNT(\*) FROM articles WHERE MATCH(title,body) AGAINST('データベース管理' IN NATURAL LANGUAGE MODE);

For boolean mode search, the search term is converted to a search phrase. For example, データベース管理 is converted to データベース 管理.

SELECT COUNT(\*) FROM articles WHERE MATCH(title,body) AGAINST('データベース管理' IN BOOLEAN MODE);

#### MeCab Parser Wildcard Search

Wildcard search terms are not tokenized. A search on データベース管理\* is performed on the prefix, データベース管理.

SELECT COUNT(\*) FROM articles WHERE MATCH(title,body) AGAINST('データベース\*' IN BOOLEAN MODE);

#### MeCab Parser Phrase Search

Phrases are tokenized. For example, データベース管理 is tokenized as データベース 管理.

SELECT COUNT(\*) FROM articles WHERE MATCH(title,body) AGAINST('"データベース管理"' IN BOOLEAN MODE);

#### Installing MeCab From a Binary Distribution (Optional)

This section describes how to install mecab and mecab-ipadic from a binary distribution using a native package management utility. For example, on Fedora, you can use Yum to perform the installation:

yum mecab-devel

On Debian or Ubuntu, you can perform an APT installation:

apt-get install mecab

apt-get install mecab-ipadic

#### Installing MeCab From Source (Optional)

If you want to build mecab and mecab-ipadic from source, basic installation steps are provided below. For additional information, refer to the MeCab documentation.

Download the tar.gz packages for mecab and mecab-ipadic from <http://taku910.github.io/mecab/#download>. As of February, 2016, the latest available packages are mecab-0.996.tar.gz and mecab-ipadic-2.7.0-20070801.tar.gz.

Install mecab:

tar zxfv mecab-0.996.tar

cd mecab-0.996

./configure

make

make check

su

make install

Install mecab-ipadic:

tar zxfv mecab-ipadic-2.7.0-20070801.tar

cd mecab-ipadic-2.7.0-20070801

./configure

make

su

make install

Compile MySQL using the [WITH\_MECAB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_mecab) CMake option. Set the [WITH\_MECAB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_mecab) option to **system** if you have installed mecab and mecab-ipadic to the default location.

-DWITH\_MECAB=system

If you defined a custom installation directory, set [WITH\_MECAB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_mecab) to the custom directory. For example:

-DWITH\_MECAB=/path/to/mecab

## 12.11 Cast Functions and Operators

**Table 12.15 Cast Functions and Operators**

| **Name** | **Description** |
| --- | --- |
| [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_binary) | Cast a string to a binary string |
| [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) | Cast a value as a certain type |
| [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) | Cast a value as a certain type |

Cast functions and operators enable conversion of values from one data type to another.

[**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) with a **USING** clause converts data between character sets:

CONVERT(***expr*** USING ***transcoding\_name***)

In MySQL, transcoding names are the same as the corresponding character set names.

Examples:

SELECT CONVERT('test' USING utf8mb4);

SELECT CONVERT(\_latin1'Müller' USING utf8mb4);

INSERT INTO utf8mb4\_table (utf8mb4\_column)

SELECT CONVERT(latin1\_column USING utf8mb4) FROM latin1\_table;

To convert strings between character sets, you can also use [**CONVERT(*expr*, *type*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) syntax (without **USING**), or [**CAST(*expr* AS *type*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast), which is equivalent:

CONVERT(***string***, CHAR[(***N***)] CHARACTER SET ***charset\_name***)

CAST(***string*** AS CHAR[(***N***)] CHARACTER SET ***charset\_name***)

Examples:

SELECT CONVERT('test', CHAR CHARACTER SET utf8mb4);

SELECT CAST('test' AS CHAR CHARACTER SET utf8mb4);

If you specify **CHARACTER SET *charset\_name*** as just shown, the character set and collation of the result are ***charset\_name*** and the default collation of ***charset\_name***. If you omit **CHARACTER SET *charset\_name***, the character set and collation of the result are defined by the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variables that determine the default connection character set and collation (see [Section 10.4, “Connection Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection)).

A **COLLATE** clause is not permitted within a [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) or [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) call, but you can apply it to the function result. For example, these are legal:

SELECT CONVERT('test' USING utf8mb4) COLLATE utf8mb4\_bin;

SELECT CONVERT('test', CHAR CHARACTER SET utf8mb4) COLLATE utf8mb4\_bin;

SELECT CAST('test' AS CHAR CHARACTER SET utf8mb4) COLLATE utf8mb4\_bin;

But these are illegal:

SELECT CONVERT('test' USING utf8mb4 COLLATE utf8mb4\_bin);

SELECT CONVERT('test', CHAR CHARACTER SET utf8mb4 COLLATE utf8mb4\_bin);

SELECT CAST('test' AS CHAR CHARACTER SET utf8mb4 COLLATE utf8mb4\_bin);

Normally, you cannot compare a [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) value or other binary string in case-insensitive fashion because binary strings use the **binary** character set, which has no collation with the concept of lettercase. To perform a case-insensitive comparison, first use the [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) or [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) function to convert the value to a nonbinary string. Comparisons of the resulting string use its collation. For example, if the conversion result collation is not case-sensitive, a [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) operation is not case-sensitive. That is true for the following operation because the default **utf8mb4** collation (**utf8mb4\_0900\_ai\_ci**) is not case-sensitive:

SELECT 'A' LIKE CONVERT(***blob\_col*** USING utf8mb4)

FROM ***tbl\_name***;

To specify a particular collation for the converted string, use a **COLLATE** clause following the [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) call:

SELECT 'A' LIKE CONVERT(***blob\_col*** USING utf8mb4) COLLATE utf8mb4\_unicode\_ci

FROM ***tbl\_name***;

To use a different character set, substitute its name for **utf8mb4** in the preceding statements (and similarly to use a different collation).

[**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) and [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) can be used more generally for comparing strings represented in different character sets. For example, a comparison of these strings results in an error because they have different character sets:

mysql> **SET @s1 = \_latin1 'abc', @s2 = \_latin2 'abc';**

mysql> **SELECT @s1 = @s2;**

ERROR 1267 (HY000): Illegal mix of collations (latin1\_swedish\_ci,IMPLICIT)

and (latin2\_general\_ci,IMPLICIT) for operation '='

Converting one of the strings to a character set compatible with the other enables the comparison to occur without error:

mysql> **SELECT @s1 = CONVERT(@s2 USING latin1);**

+---------------------------------+

| @s1 = CONVERT(@s2 USING latin1) |

+---------------------------------+

| 1 |

+---------------------------------+

For string literals, another way to specify the character set is to use a character set introducer. **\_latin1** and **\_latin2** in the preceding example are instances of introducers. Unlike conversion functions such as [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast), or [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert), which convert a string from one character set to another, an introducer designates a string literal as having a particular character set, with no conversion involved. For more information, see [Section 10.3.8, “Character Set Introducers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-introducer).

Character set conversion is also useful preceding lettercase conversion of binary strings. [**LOWER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lower) and [**UPPER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper) are ineffective when applied directly to binary strings because the concept of lettercase does not apply. To perform lettercase conversion of a binary string, first convert it to a nonbinary string using a character set appropriate for the data stored in the string:

mysql> **SET @str = BINARY 'New York';**

mysql> **SELECT LOWER(@str), LOWER(CONVERT(@str USING utf8mb4));**

+-------------+------------------------------------+

| LOWER(@str) | LOWER(CONVERT(@str USING utf8mb4)) |

+-------------+------------------------------------+

| New York | new york |

+-------------+------------------------------------+

Be aware that if you convert an indexed column using [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_binary), [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast), or [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert), MySQL may not be able to use the index efficiently.

The cast functions are useful for creating a column with a specific type in a [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement:

mysql> **CREATE TABLE new\_table SELECT CAST('2000-01-01' AS DATE) AS c1;**

mysql> **SHOW CREATE TABLE new\_table\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: new\_table

Create Table: CREATE TABLE `new\_table` (

`c1` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4

The cast functions are useful for sorting [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) columns in lexical order. Normally, sorting of [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) columns occurs using the internal numeric values. Casting the values to [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) results in a lexical sort:

SELECT ***enum\_col*** FROM ***tbl\_name*** ORDER BY CAST(***enum\_col*** AS CHAR);

[**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) also changes the result if you use it as part of a more complex expression such as [**CONCAT('Date: ',CAST(NOW() AS DATE))**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat).

For temporal values, there is little need to use [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) to extract data in different formats. Instead, use a function such as [**EXTRACT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extract), [**DATE\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-format), or [**TIME\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_time-format). See [Section 12.7, “Date and Time Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#date-and-time-functions).

To cast a string to a number, it normally suffices to use the string value in numeric context:

mysql> **SELECT 1+'1';**

-> 2

That is also true for hexadecimal and bit literals, which are binary strings by default:

mysql> **SELECT X'41', X'41'+0;**

-> 'A', 65

mysql> **SELECT b'1100001', b'1100001'+0;**

-> 'a', 97

A string used in an arithmetic operation is converted to a floating-point number during expression evaluation.

A number used in string context is converted to a string:

mysql> **SELECT CONCAT('hello you ',2);**

-> 'hello you 2'

For information about implicit conversion of numbers to strings, see [Section 12.3, “Type Conversion in Expression Evaluation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#type-conversion).

MySQL supports arithmetic with both signed and unsigned 64-bit values. For numeric operators (such as [**+**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_plus) or [**-**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_minus)) where one of the operands is an unsigned integer, the result is unsigned by default (see [Section 12.6.1, “Arithmetic Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#arithmetic-functions)). To override this, use the **SIGNED** or **UNSIGNED** cast operator to cast a value to a signed or unsigned 64-bit integer, respectively.

mysql> **SELECT 1 - 2;**

-> -1

mysql> **SELECT CAST(1 - 2 AS UNSIGNED);**

-> 18446744073709551615

mysql> **SELECT CAST(CAST(1 - 2 AS UNSIGNED) AS SIGNED);**

-> -1

If either operand is a floating-point value, the result is a floating-point value and is not affected by the preceding rule. (In this context, [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) column values are regarded as floating-point values.)

mysql> **SELECT CAST(1 AS UNSIGNED) - 2.0;**

-> -1.0

The SQL mode affects the result of conversion operations (see [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode)). Examples:

For conversion of a “zero” date string to a date, [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) and [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) return **NULL** and produce a warning when the [**NO\_ZERO\_DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_zero_date) SQL mode is enabled.

For integer subtraction, if the [**NO\_UNSIGNED\_SUBTRACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_unsigned_subtraction) SQL mode is enabled, the subtraction result is signed even if any operand is unsigned.

The following list describes the available cast functions and operators:

**[BINARY](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_binary)** ***expr***

The [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_binary) operator converts the expression to a binary string (a string that has the **binary** character set and **binary** collation). A common use for [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_binary) is to force a character string comparison to be done byte by byte using numeric byte values rather than character by character. The [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_binary) operator also causes trailing spaces in comparisons to be significant. For information about the differences between the **binary** collation of the **binary** character set and the **\_bin** collations of nonbinary character sets, see [Section 10.8.5, “The binary Collation Compared to \_bin Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations).

mysql> **SELECT 'a' = 'A';**

-> 1

mysql> **SELECT BINARY 'a' = 'A';**

-> 0

mysql> **SELECT 'a' = 'a ';**

-> 1

mysql> **SELECT BINARY 'a' = 'a ';**

-> 0

In a comparison, [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_binary) affects the entire operation; it can be given before either operand with the same result.

To convert a string expression to a binary string, these constructs are equivalent:

CAST(***expr*** AS BINARY)

CONVERT(***expr*** USING BINARY)

BINARY ***expr***

If a value is a string literal, it can be designated as a binary string without converting it by using the **\_binary** character set introducer:

mysql> **SELECT 'a' = 'A';**

-> 1

mysql> **SELECT \_binary 'a' = 'A';**

-> 0

For information about introducers, see [Section 10.3.8, “Character Set Introducers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-introducer).

The [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_binary) operator in expressions differs in effect from the **BINARY** attribute in character column definitions. For a character column defined with the **BINARY** attribute, MySQL assigns the table default character set and the binary (**\_bin**) collation of that character set. Every nonbinary character set has a **\_bin** collation. For example, if the table default character set is **utf8mb4**, these two column definitions are equivalent:

CHAR(10) BINARY

CHAR(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_bin

The use of **CHARACTER SET binary** in the definition of a [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) column causes the column to be treated as the corresponding binary string data type. For example, the following pairs of definitions are equivalent:

CHAR(10) CHARACTER SET binary

BINARY(10)

VARCHAR(10) CHARACTER SET binary

VARBINARY(10)

TEXT CHARACTER SET binary

BLOB

**[CAST(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cast)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cast)*[AS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cast)*[type](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cast)*[[ARRAY])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cast)**

[**CAST(*timestamp\_value* AT TIME ZONE *timezone\_specifier* AS DATETIME[(*precision*)])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast)

***timezone\_specifier***: [INTERVAL] '+00:00' | 'UTC'

The [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) function takes an expression of any type and produces a result value of the specified type, similar to [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert). For more information, see the description of [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert).

In MySQL 8.0.17 and higher, [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) allows the use of an additional **ARRAY** keyword for creating a multi-valued index on a [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) array as part of [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index), [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements. **ARRAY** is not supported except when used to create a multi-valued index in one of these statements, in which case it is required. The column being indexed must be a column of type **JSON**. With **ARRAY**, the ***type*** following the **AS** keyword may specify any of the types supported by **CAST()**, with the exceptions of **BINARY**, **JSON**, and **YEAR**. For syntax information and examples, as well as other relevant information, see [Multi-Valued Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-multi-valued).

**Note**

Unlike [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast), [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) does not support multi-valued index creation or the **ARRAY** keyword.

Beginning with MySQL 8.0.22, **CAST()** supports retrieval of a [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value as being in UTC, using the **AT TIMEZONE** operator. The only supported time zone is UTC; this can be specified as either of **'+00:00'** or **'UTC'**. The only return type supported by this syntax is **DATETIME**, with an optional precision specifier in the range of 0 to 6, inclusive.

**TIMESTAMP** values that use timezone offsets are also supported.

mysql> **SELECT @@system\_time\_zone;**

+--------------------+

| @@system\_time\_zone |

+--------------------+

| EDT |

+--------------------+

1 row in set (0.00 sec)

mysql> **CREATE TABLE TZ (c TIMESTAMP);**

Query OK, 0 rows affected (0.41 sec)

mysql> **INSERT INTO tz VALUES**

> **ROW(CURRENT\_TIMESTAMP),**

> **ROW('2020-07-28 14:50:15+1:00');**

Query OK, 1 row affected (0.08 sec)

mysql> **TABLE tz;**

+---------------------+

| c |

+---------------------+

| 2020-07-28 09:22:41 |

| 2020-07-28 09:50:15 |

+---------------------+

2 rows in set (0.00 sec)

mysql> **SELECT CAST(c AT TIME ZONE '+00:00' AS DATETIME) AS u FROM tz;**

+---------------------+

| u |

+---------------------+

| 2020-07-28 13:22:41 |

| 2020-07-28 13:50:15 |

+---------------------+

2 rows in set (0.00 sec)

mysql> **SELECT CAST(c AT TIME ZONE 'UTC' AS DATETIME(2)) AS u FROM tz;**

+------------------------+

| u |

+------------------------+

| 2020-07-28 13:22:41.00 |

| 2020-07-28 13:50:15.00 |

+------------------------+

2 rows in set (0.00 sec)

If you use **'UTC'** as the time zone specifier with this form of **CAST()**, and the server raises an error such as Unknown or incorrect time zone: 'UTC', you may need to install the MySQL time zone tables (see [Populating the Time Zone Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#time-zone-installation)).

**AT TIME ZONE** does not support the **ARRAY** keyword, and is not supported by the [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) function.

**[CONVERT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert)*[USING](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert)*[transcoding\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_convert)**, [**CONVERT(*expr*,*type*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert)

The [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) function takes an expression of any type and produces a result value of the specified type.

[**CONVERT(... USING ...)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) is standard SQL syntax. The non-**USING** form of [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) is ODBC syntax.

[**CONVERT(*expr* USING *transcoding\_name*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) converts data between different character sets. In MySQL, transcoding names are the same as the corresponding character set names. For example, this statement converts the string **'abc'** in the default character set to the corresponding string in the **utf8mb4** character set:

SELECT CONVERT('abc' USING utf8mb4);

[**CONVERT(*expr*, *type*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) syntax (without **USING**) takes an expression and a ***type*** value specifying the result type. This operation may also be expressed as [**CAST(*expr* AS *type*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast), which is equivalent. These ***type*** values are permitted:

**BINARY[(*N*)]**

Produces a string with the [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) data type. For a description of how this affects comparisons, see [Section 11.3.3, “The BINARY and VARBINARY Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary). If the optional length ***N*** is given, **BINARY(*N*)** causes the cast to use no more than ***N*** bytes of the argument. Values shorter than ***N*** bytes are padded with **0x00** bytes to a length of ***N***.

**CHAR[(*N*)] [*charset\_info*]**

Produces a string with the [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) data type. If the optional length ***N*** is given, **CHAR(*N*)** causes the cast to use no more than ***N*** characters of the argument. No padding occurs for values shorter than ***N*** characters.

With no ***charset\_info*** clause, **CHAR** produces a string with the default character set. To specify the character set explicitly, these ***charset\_info*** values are permitted:

**CHARACTER SET *charset\_name***: Produces a string with the given character set.

**ASCII**: Shorthand for **CHARACTER SET latin1**.

**UNICODE**: Shorthand for **CHARACTER SET ucs2**.

In all cases, the string has the character set default collation.

**DATE**

Produces a [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value.

**DATETIME**

Produces a [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value.

**DECIMAL[(*M*[,*D*])]**

Produces a [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) value. If the optional ***M*** and ***D*** values are given, they specify the maximum number of digits (the precision) and the number of digits following the decimal point (the scale).

**DOUBLE**

Produces a [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) result. Added in MySQL 8.0.17.

**FLOAT[(*p*)]**

If the precision ***p*** is not specified, produces a result of type [**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types). If ***p*** is provided and 0 <= < ***p*** <= 24, the result is of type **FLOAT**. If 25 <= ***p*** <= 53, the result is of type [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types). If ***p*** < 0 or ***p*** > 53, an error is returned. Added in MySQL 8.0.17.

**JSON**

Produces a [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) value. For details on the rules for conversion of values between [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) and other types, see [Comparison and Ordering of JSON Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json-comparison).

**NCHAR[(*N*)]**

Like **CHAR**, but produces a string with the national character set. See [Section 10.3.7, “The National Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-national).

Unlike **CHAR**, **NCHAR** does not permit trailing character set information to be specified.

**REAL**

Produces a result of type [**REAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types). This is actually **FLOAT** if the **REAL\_AS\_FLOAT** SQL mode is enabled; otherwise the result is of type **DOUBLE**.

**SIGNED [INTEGER]**

Produces a signed integer value.

***spatial\_type***

As of MySQL 8.0.24, [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) and [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) support casting geometry values from one spatial type to another, for certain combinations of spatial types. The following list shows the permitted type combinations, where “MySQL extension” designates casts implemented in MySQL beyond those defined in the [SQL/MM standard](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#spatial-types):

From **Point** to:

**MultiPoint**

**GeometryCollection**

From **LineString** to:

**Polygon** (MySQL extension)

**MultiPoint** (MySQL extension)

**MultiLineString**

**GeometryCollection**

From **Polygon** to:

**LineString** (MySQL extension)

**MultiLineString** (MySQL extension)

**MultiPolygon**

**GeometryCollection**

From **MultiPoint** to:

**Point**

**LineString** (MySQL extension)

**GeometryCollection**

From **MultiLineString** to:

**LineString**

**Polygon** (MySQL extension)

**MultiPolygon** (MySQL extension)

**GeometryCollection**

From **MultiPolygon** to:

**Polygon**

**MultiLineString** (MySQL extension)

**GeometryCollection**

From **GeometryCollection** to:

**Point**

**LineString**

**Polygon**

**MultiPoint**

**MultiLineString**

**MultiPolygon**

In spatial casts, **GeometryCollection** and **GeomCollection** are synonyms for the same result type.

Some conditions apply to all spatial type casts, and some conditions apply only when the cast result is to have a particular spatial type. For information about terms such as “well-formed geometry,” see [Section 11.4.4, “Geometry Well-Formedness and Validity”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#geometry-well-formedness-validity).

[General Conditions for Spatial Casts](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-cast-general-conditions)

[Conditions for Casts to Point](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-cast-point-conditions)

[Conditions for Casts to LineString](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-cast-linestring-conditions)

[Conditions for Casts to Polygon](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-cast-polygon-conditions)

[Conditions for Casts to MultiPoint](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-cast-multipoint-conditions)

[Conditions for Casts to MultiLineString](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-cast-multilinestring-conditions)

[Conditions for Casts to MultiPolygon](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-cast-multipolygon-conditions)

[Conditions for Casts to GeometryCollection](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-cast-geometrycollection-conditions)

### General Conditions for Spatial Casts

These conditions apply to all spatial casts regardless of the result type:

The result of a cast is in the same SRS as that of the expression to cast.

Casting between spatial types does not change coordinate values or order.

If the expression to cast is **NULL**, the function result is **NULL**.

Casting to spatial types using the [**JSON\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-value) function with a **RETURNING** clause specifying a spatial type is not permitted.

Casting to an **ARRAY** of spatial types is not permitted.

If the spatial type combination is permitted but the expression to cast is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If the spatial type combination is permitted but the expression to cast is a syntactically well-formed geometry in an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

If the expression to cast has a geographic SRS but has a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_GEOMETRY\_PARAM\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_longitude_out_of_range) error occurs.

If a latitude value is not in the range [−90, 90], an [**ER\_GEOMETRY\_PARAM\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_latitude_out_of_range) error occurs.

Ranges shown are in degrees. If an SRS uses another unit, the range uses the corresponding values in its unit. The exact range limits deviate slightly due to floating-point arithmetic.

### Conditions for Casts to Point

These conditions apply when the cast result type is **Point**:

If the expression to cast is a well-formed geometry of type **Point**, the function result is that **Point**.

If the expression to cast is a well-formed geometry of type **MultiPoint** containing a single **Point**, the function result is that **Point**. If the expression contains more than one **Point**, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

If the expression to cast is a well-formed geometry of type **GeometryCollection** containing only a single **Point**, the function result is that **Point**. If the expression is empty, contains more than one **Point**, or contains other geometry types, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

If the expression to cast is a well-formed geometry of type other than **Point**, **MultiPoint**, **GeometryCollection**, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

### Conditions for Casts to LineString

These conditions apply when the cast result type is **LineString**:

If the expression to cast is a well-formed geometry of type **LineString**, the function result is that **LineString**.

If the expression to cast is a well-formed geometry of type **Polygon** that has no inner rings, the function result is a **LineString** containing the points of the outer ring in the same order. If the expression has inner rings, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

If the expression to cast is a well-formed geometry of type **MultiPoint** containing at least two points, the function result is a **LineString** containing the points of the **MultiPoint** in the order they appear in the expression. If the expression contains only one **Point**, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

If the expression to cast is a well-formed geometry of type **MultiLineString** containing a single **LineString**, the function result is that **LineString**. If the expression contains more than one **LineString**, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

If the expression to cast is a well-formed geometry of type **GeometryCollection**, containing only a single **LineString**, the function result is that **LineString**. If the expression is empty, contains more than one **LineString**, or contains other geometry types, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

If the expression to cast is a well-formed geometry of type other than **LineString**, **Polygon**, **MultiPoint**, **MultiLineString**, or **GeometryCollection**, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

### Conditions for Casts to Polygon

These conditions apply when the cast result type is **Polygon**:

If the expression to cast is a well-formed geometry of type **LineString** that is a ring (that is, the start and end points are the same), the function result is a **Polygon** with an outer ring consisting of the points of the **LineString** in the same order. If the expression is not a ring, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs. If the ring is not in the correct order (the exterior ring must be counter-clockwise), an [**ER\_INVALID\_CAST\_POLYGON\_RING\_DIRECTION**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_polygon_ring_direction) error occurs.

If the expression to cast is a well-formed geometry of type **Polygon**, the function result is that **Polygon**.

If the expression to cast is a well-formed geometry of type **MultiLineString** where all elements are rings, the function result is a **Polygon** with the first **LineString** as outer ring and any additional **LineString** values as inner rings. If any element of the expression is not a ring, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs. If any ring is not in the correct order (the exterior ring must be counter-clockwise, interior rings must be clockwise), an [**ER\_INVALID\_CAST\_POLYGON\_RING\_DIRECTION**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_polygon_ring_direction) error occurs.

If the expression to cast is a well-formed geometry of type **MultiPolygon** containing a single **Polygon**, the function result is that **Polygon**. If the expression contains more than one **Polygon**, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

If the expression to cast is a well-formed geometry of type **GeometryCollection** containing only a single **Polygon**, the function result is that **Polygon**. If the expression is empty, contains more than one **Polygon**, or contains other geometry types, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

If the expression to cast is a well-formed geometry of type other than **LineString**, **Polygon**, **MultiLineString**, **MultiPolygon**, or **GeometryCollection**, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

### Conditions for Casts to MultiPoint

These conditions apply when the cast result type is **MultiPoint**:

If the expression to cast is a well-formed geometry of type **Point**, the function result is a **MultiPoint** containing that **Point** as its sole element.

If the expression to cast is a well-formed geometry of type **LineString**, the function result is a **MultiPoint** containing the points of the **LineString** in the same order.

If the expression to cast is a well-formed geometry of type **MultiPoint**, the function result is that **MultiPoint**.

If the expression to cast is a well-formed geometry of type **GeometryCollection** containing only points, the function result is a **MultiPoint** containing those points. If the **GeometryCollection** is empty or contains other geometry types, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

If the expression to cast is a well-formed geometry of type other than **Point**, **LineString**, **MultiPoint**, or **GeometryCollection**, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

### Conditions for Casts to MultiLineString

These conditions apply when the cast result type is **MultiLineString**:

If the expression to cast is a well-formed geometry of type **LineString**, the function result is a **MultiLineString** containing that **LineString** as its sole element.

If the expression to cast is a well-formed geometry of type **Polygon**, the function result is a **MultiLineString** containing the outer ring of the **Polygon** as its first element and any inner rings as additional elements in the order they appear in the expression.

If the expression to cast is a well-formed geometry of type **MultiLineString**, the function result is that **MultiLineString**.

If the expression to cast is a well-formed geometry of type **MultiPolygon** containing only polygons without inner rings, the function result is a **MultiLineString** containing the polygon rings in the order they appear in the expression. If the expression contains any polygons with inner rings, an [**ER\_WRONG\_PARAMETERS\_TO\_STORED\_FCT**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_wrong_parameters_to_stored_fct) error occurs.

If the expression to cast is a well-formed geometry of type **GeometryCollection** containing only linestrings, the function result is a **MultiLineString** containing those linestrings. If the expression is empty or contains other geometry types, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

If the expression to cast is a well-formed geometry of type other than **LineString**, **Polygon**, **MultiLineString**, **MultiPolygon**, or **GeometryCollection**, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

### Conditions for Casts to MultiPolygon

These conditions apply when the cast result type is **MultiPolygon**:

If the expression to cast is a well-formed geometry of type **Polygon**, the function result is a **MultiPolygon** containing the **Polygon** as its sole element.

If the expression to cast is a well-formed geometry of type **MultiLineString** where all elements are rings, the function result is a **MultiPolygon** containing a **Polygon** with only an outer ring for each element of the expression. If any element is not a ring, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs. If any ring is not in the correct order (exterior ring must be counter-clockwise), an [**ER\_INVALID\_CAST\_POLYGON\_RING\_DIRECTION**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_polygon_ring_direction) error occurs.

If the expression to cast is a well-formed geometry of type **MultiPolygon**, the function result is that **MultiPolygon**.

If the expression to cast is a well-formed geometry of type **GeometryCollection** containing only polygons, the function result is a **MultiPolygon** containing those polygons. If the expression is empty or contains other geometry types, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

If the expression to cast is a well-formed geometry of type other than **Polygon**, **MultiLineString**, **MultiPolygon**, or **GeometryCollection**, an [**ER\_INVALID\_CAST\_TO\_GEOMETRY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_cast_to_geometry) error occurs.

### Conditions for Casts to GeometryCollection

These conditions apply when the cast result type is **GeometryCollection**:

**GeometryCollection** and **GeomCollection** are synonyms for the same result type.

If the expression to cast is a well-formed geometry of type **Point**, the function result is a **GeometryCollection** containing that **Point** as its sole element.

If the expression to cast is a well-formed geometry of type **LineString**, the function result is a **GeometryCollection** containing that **LineString** as its sole element.

If the expression to cast is a well-formed geometry of type **Polygon**, the function result is a **GeometryCollection** containing that **Polygon** as its sole element.

If the expression to cast is a well-formed geometry of type **MultiPoint**, the function result is a **GeometryCollection** containing the points in the order they appear in the expression.

If the expression to cast is a well-formed geometry of type **MultiLineString**, the function result is a **GeometryCollection** containing the linestrings in the order they appear in the expression.

If the expression to cast is a well-formed geometry of type **MultiPolygon**, the function result is a **GeometryCollection** containing the elements of the **MultiPolygon** in the order they appear in the expression.

If the expression to cast is a well-formed geometry of type **GeometryCollection**, the function result is that **GeometryCollection**.

**TIME**

Produces a [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time) value.

**UNSIGNED [INTEGER]**

Produces an unsigned integer value.

**YEAR**

Produces a [**YEAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#year) value. Added in MySQL 8.0.22. These rules govern conversion to **YEAR**:

For a four-digit number in the range 1901-2155 inclusive, or for a string which can be interpreted as a four-digit number in this range, return the corresponding **YEAR** value.

For a number consisting of one or two digits, or for a string which can be interpeted as such a number, return a **YEAR** value as follows:

If the number is in the range 1-69 inclusive, add 2000 and return the sum.

If the number is in the range 70-99 inclusive, add 1900 and return the sum.

For a string which evaluates to 0, return 2000.

For the number 0, return 0.

For a [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), or [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value, return the **YEAR** portion of the value. For a [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time) value, return the current year.

If you do not specify the type of a **TIME** argument, you may get a different result from what you expect, as shown here:

mysql> **SELECT CONVERT("11:35:00", YEAR), CONVERT(TIME "11:35:00", YEAR);**

+---------------------------+--------------------------------+

| CONVERT("11:35:00", YEAR) | CONVERT(TIME "11:35:00", YEAR) |

+---------------------------+--------------------------------+

| 2011 | 2020 |

+---------------------------+--------------------------------+

If the argument is of type [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types), [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types), [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types), or [**REAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types), round the value to the nearest integer, then attempt to cast the value to **YEAR** using the rules for integer values, as shown here:

mysql> **SELECT CONVERT(1944.35, YEAR), CONVERT(1944.50, YEAR);**

+------------------------+------------------------+

| CONVERT(1944.35, YEAR) | CONVERT(1944.50, YEAR) |

+------------------------+------------------------+

| 1944 | 1945 |

+------------------------+------------------------+

1 row in set (0.00 sec)

mysql> **SELECT CONVERT(66.35, YEAR), CONVERT(66.50, YEAR);**

+----------------------+----------------------+

| CONVERT(66.35, YEAR) | CONVERT(66.50, YEAR) |

+----------------------+----------------------+

| 2066 | 2067 |

+----------------------+----------------------+

1 row in set (0.00 sec)

An argument of type [**GEOMETRY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#spatial-type-overview) cannot be converted to [**YEAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#year).

For a value that cannot be successfully converted to **YEAR**, return **NULL**.

A string value containing non-numeric characters which must be truncated prior to conversion raises a warning, as shown here:

mysql> **SELECT CONVERT("1979aaa", YEAR);**

+--------------------------+

| CONVERT("1979aaa", YEAR) |

+--------------------------+

| 1979 |

+--------------------------+

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS;**

+---------+------+-------------------------------------------+

| Level | Code | Message |

+---------+------+-------------------------------------------+

| Warning | 1292 | Truncated incorrect YEAR value: '1979aaa' |

+---------+------+-------------------------------------------+

## 12.12 XML Functions

**Table 12.16 XML Functions**

| **Name** | **Description** |
| --- | --- |
| [**ExtractValue()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extractvalue) | Extract a value from an XML string using XPath notation |
| [**UpdateXML()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_updatexml) | Return replaced XML fragment |

This section discusses XML and related functionality in MySQL.

**Note**

It is possible to obtain XML-formatted output from MySQL in the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) and [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) clients by invoking them with the [--xml](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_xml) option. See [Section 4.5.1, “mysql — The MySQL Command-Line Client”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql), and [Section 4.5.4, “mysqldump — A Database Backup Program”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump).

Two functions providing basic XPath 1.0 (XML Path Language, version 1.0) capabilities are available. Some basic information about XPath syntax and usage is provided later in this section; however, an in-depth discussion of these topics is beyond the scope of this manual, and you should refer to the [XML Path Language (XPath) 1.0 standard](http://www.w3.org/TR/xpath) for definitive information. A useful resource for those new to XPath or who desire a refresher in the basics is the [Zvon.org XPath Tutorial](http://www.zvon.org/xxl/XPathTutorial/), which is available in several languages.

**Note**

These functions remain under development. We continue to improve these and other aspects of XML and XPath functionality in MySQL 8.0 and onwards. You may discuss these, ask questions about them, and obtain help from other users with them in the [MySQL XML User Forum](https://forums.mysql.com/list.php?44).

XPath expressions used with these functions support user variables and local stored program variables. User variables are weakly checked; variables local to stored programs are strongly checked (see also Bug #26518):

**User variables (weak checking).** Variables using the syntax **$@*variable\_name*** (that is, user variables) are not checked. No warnings or errors are issued by the server if a variable has the wrong type or has previously not been assigned a value. This also means the user is fully responsible for any typographical errors, since no warnings are given if (for example) **$@myvairable** is used where **$@myvariable** was intended.

Example:

mysql> **SET @xml = '<a><b>X</b><b>Y</b></a>';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SET @i =1, @j = 2;**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT @i, ExtractValue(@xml, '//b[$@i]');**

+------+--------------------------------+

| @i | ExtractValue(@xml, '//b[$@i]') |

+------+--------------------------------+

| 1 | X |

+------+--------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT @j, ExtractValue(@xml, '//b[$@j]');**

+------+--------------------------------+

| @j | ExtractValue(@xml, '//b[$@j]') |

+------+--------------------------------+

| 2 | Y |

+------+--------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT @k, ExtractValue(@xml, '//b[$@k]');**

+------+--------------------------------+

| @k | ExtractValue(@xml, '//b[$@k]') |

+------+--------------------------------+

| NULL | |

+------+--------------------------------+

1 row in set (0.00 sec)

**Variables in stored programs (strong checking).** Variables using the syntax **$*variable\_name*** can be declared and used with these functions when they are called inside stored programs. Such variables are local to the stored program in which they are defined, and are strongly checked for type and value.

Example:

mysql> **DELIMITER |**

mysql> **CREATE PROCEDURE myproc ()**

-> **BEGIN**

-> **DECLARE i INT DEFAULT 1;**

-> **DECLARE xml VARCHAR(25) DEFAULT '<a>X</a><a>Y</a><a>Z</a>';**

->

-> **WHILE i < 4 DO**

-> **SELECT xml, i, ExtractValue(xml, '//a[$i]');**

-> **SET i = i+1;**

-> **END WHILE;**

-> **END |**

Query OK, 0 rows affected (0.01 sec)

mysql> **DELIMITER ;**

mysql> **CALL myproc();**

+--------------------------+---+------------------------------+

| xml | i | ExtractValue(xml, '//a[$i]') |

+--------------------------+---+------------------------------+

| <a>X</a><a>Y</a><a>Z</a> | 1 | X |

+--------------------------+---+------------------------------+

1 row in set (0.00 sec)

+--------------------------+---+------------------------------+

| xml | i | ExtractValue(xml, '//a[$i]') |

+--------------------------+---+------------------------------+

| <a>X</a><a>Y</a><a>Z</a> | 2 | Y |

+--------------------------+---+------------------------------+

1 row in set (0.01 sec)

+--------------------------+---+------------------------------+

| xml | i | ExtractValue(xml, '//a[$i]') |

+--------------------------+---+------------------------------+

| <a>X</a><a>Y</a><a>Z</a> | 3 | Z |

+--------------------------+---+------------------------------+

1 row in set (0.01 sec)

**Parameters.** Variables used in XPath expressions inside stored routines that are passed in as parameters are also subject to strong checking.

Expressions containing user variables or variables local to stored programs must otherwise (except for notation) conform to the rules for XPath expressions containing variables as given in the XPath 1.0 specification.

**Note**

A user variable used to store an XPath expression is treated as an empty string. Because of this, it is not possible to store an XPath expression as a user variable. (Bug #32911)

**[ExtractValue(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_extractvalue)*[xml\_frag](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_extractvalue)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_extractvalue)*[xpath\_expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_extractvalue)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_extractvalue)**

**[ExtractValue()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_extractvalue)** takes two string arguments, a fragment of XML markup ***xml\_frag*** and an XPath expression ***xpath\_expr*** (also known as a locator); it returns the text (**CDATA**) of the first text node which is a child of the element or elements matched by the XPath expression.

Using this function is the equivalent of performing a match using the ***xpath\_expr*** after appending **/text()**. In other words, [**ExtractValue('<a><b>Sakila</b></a>', '/a/b')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extractvalue) and [**ExtractValue('<a><b>Sakila</b></a>', '/a/b/text()')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extractvalue) produce the same result.

If multiple matches are found, the content of the first child text node of each matching element is returned (in the order matched) as a single, space-delimited string.

If no matching text node is found for the expression (including the implicit **/text()**)—for whatever reason, as long as ***xpath\_expr*** is valid, and ***xml\_frag*** consists of elements which are properly nested and closed—an empty string is returned. No distinction is made between a match on an empty element and no match at all. This is by design.

If you need to determine whether no matching element was found in ***xml\_frag*** or such an element was found but contained no child text nodes, you should test the result of an expression that uses the XPath **count()** function. For example, both of these statements return an empty string, as shown here:

mysql> **SELECT ExtractValue('<a><b/></a>', '/a/b');**

+-------------------------------------+

| ExtractValue('<a><b/></a>', '/a/b') |

+-------------------------------------+

| |

+-------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT ExtractValue('<a><c/></a>', '/a/b');**

+-------------------------------------+

| ExtractValue('<a><c/></a>', '/a/b') |

+-------------------------------------+

| |

+-------------------------------------+

1 row in set (0.00 sec)

However, you can determine whether there was actually a matching element using the following:

mysql> **SELECT ExtractValue('<a><b/></a>', 'count(/a/b)');**

+-------------------------------------+

| ExtractValue('<a><b/></a>', 'count(/a/b)') |

+-------------------------------------+

| 1 |

+-------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT ExtractValue('<a><c/></a>', 'count(/a/b)');**

+-------------------------------------+

| ExtractValue('<a><c/></a>', 'count(/a/b)') |

+-------------------------------------+

| 0 |

+-------------------------------------+

1 row in set (0.01 sec)

**Important**

[**ExtractValue()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extractvalue) returns only **CDATA**, and does not return any tags that might be contained within a matching tag, nor any of their content (see the result returned as **val1** in the following example).

mysql> **SELECT**

->  **ExtractValue('<a>ccc<b>ddd</b></a>', '/a') AS val1,**

->  **ExtractValue('<a>ccc<b>ddd</b></a>', '/a/b') AS val2,**

->  **ExtractValue('<a>ccc<b>ddd</b></a>', '//b') AS val3,**

->  **ExtractValue('<a>ccc<b>ddd</b></a>', '/b') AS val4,**

->  **ExtractValue('<a>ccc<b>ddd</b><b>eee</b></a>', '//b') AS val5;**

+------+------+------+------+---------+

| val1 | val2 | val3 | val4 | val5 |

+------+------+------+------+---------+

| ccc | ddd | ddd | | ddd eee |

+------+------+------+------+---------+

This function uses the current SQL collation for making comparisons with **contains()**, performing the same collation aggregation as other string functions (such as [**CONCAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat)), in taking into account the collation coercibility of their arguments; see [Section 10.8.4, “Collation Coercibility in Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-coercibility), for an explanation of the rules governing this behavior.

(Previously, binary—that is, case-sensitive—comparison was always used.)

**NULL** is returned if ***xml\_frag*** contains elements which are not properly nested or closed, and a warning is generated, as shown in this example:

mysql> **SELECT ExtractValue('<a>c</a><b', '//a');**

+-----------------------------------+

| ExtractValue('<a>c</a><b', '//a') |

+-----------------------------------+

| NULL |

+-----------------------------------+

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Warning

Code: 1525

Message: Incorrect XML value: 'parse error at line 1 pos 11:

END-OF-INPUT unexpected ('>' wanted)'

1 row in set (0.00 sec)

mysql> **SELECT ExtractValue('<a>c</a><b/>', '//a');**

+-------------------------------------+

| ExtractValue('<a>c</a><b/>', '//a') |

+-------------------------------------+

| c |

+-------------------------------------+

1 row in set (0.00 sec)

**[UpdateXML(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_updatexml)*[xml\_target](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_updatexml)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_updatexml)*[xpath\_expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_updatexml)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_updatexml)*[new\_xml](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_updatexml)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_updatexml)**

This function replaces a single portion of a given fragment of XML markup ***xml\_target*** with a new XML fragment ***new\_xml***, and then returns the changed XML. The portion of ***xml\_target*** that is replaced matches an XPath expression ***xpath\_expr*** supplied by the user.

If no expression matching ***xpath\_expr*** is found, or if multiple matches are found, the function returns the original ***xml\_target*** XML fragment. All three arguments should be strings.

mysql> **SELECT**

->  **UpdateXML('<a><b>ccc</b><d></d></a>', '/a', '<e>fff</e>') AS val1,**

->  **UpdateXML('<a><b>ccc</b><d></d></a>', '/b', '<e>fff</e>') AS val2,**

->  **UpdateXML('<a><b>ccc</b><d></d></a>', '//b', '<e>fff</e>') AS val3,**

->  **UpdateXML('<a><b>ccc</b><d></d></a>', '/a/d', '<e>fff</e>') AS val4,**

->  **UpdateXML('<a><d></d><b>ccc</b><d></d></a>', '/a/d', '<e>fff</e>') AS val5**

-> **\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

val1: <e>fff</e>

val2: <a><b>ccc</b><d></d></a>

val3: <a><e>fff</e><d></d></a>

val4: <a><b>ccc</b><e>fff</e></a>

val5: <a><d></d><b>ccc</b><d></d></a>

**Note**

A discussion in depth of XPath syntax and usage are beyond the scope of this manual. Please see the [XML Path Language (XPath) 1.0 specification](http://www.w3.org/TR/xpath) for definitive information. A useful resource for those new to XPath or who are wishing a refresher in the basics is the [Zvon.org XPath Tutorial](http://www.zvon.org/xxl/XPathTutorial/), which is available in several languages.

Descriptions and examples of some basic XPath expressions follow:

**/*tag***

Matches **<*tag*/>** if and only if **<*tag*/>** is the root element.

Example: **/a** has a match in **<a><b/></a>** because it matches the outermost (root) tag. It does not match the inner ***a*** element in **<b><a/></b>** because in this instance it is the child of another element.

**/*tag1*/*tag2***

Matches **<*tag2*/>** if and only if it is a child of **<*tag1*/>**, and **<*tag1*/>** is the root element.

Example: **/a/b** matches the ***b*** element in the XML fragment **<a><b/></a>** because it is a child of the root element ***a***. It does not have a match in **<b><a/></b>** because in this case, ***b*** is the root element (and hence the child of no other element). Nor does the XPath expression have a match in **<a><c><b/></c></a>**; here, ***b*** is a descendant of ***a***, but not actually a child of ***a***.

This construct is extendable to three or more elements. For example, the XPath expression **/a/b/c** matches the ***c*** element in the fragment **<a><b><c/></b></a>**.

**//*tag***

Matches any instance of **<*tag*>**.

Example: **//a** matches the ***a*** element in any of the following: **<a><b><c/></b></a>**; **<c><a><b/></a></b>**; **<c><b><a/></b></c>**.

**//** can be combined with **/**. For example, **//a/b** matches the ***b*** element in either of the fragments **<a><b/></a>** or **<c><a><b/></a></c>**.

**Note**

**//*tag*** is the equivalent of **/descendant-or-self::\*/*tag***. A common error is to confuse this with **/descendant-or-self::*tag***, although the latter expression can actually lead to very different results, as can be seen here:

mysql> **SET @xml = '<a><b><c>w</c><b>x</b><d>y</d>z</b></a>';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT @xml;**

+-----------------------------------------+

| @xml |

+-----------------------------------------+

| <a><b><c>w</c><b>x</b><d>y</d>z</b></a> |

+-----------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT ExtractValue(@xml, '//b[1]');**

+------------------------------+

| ExtractValue(@xml, '//b[1]') |

+------------------------------+

| x z |

+------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT ExtractValue(@xml, '//b[2]');**

+------------------------------+

| ExtractValue(@xml, '//b[2]') |

+------------------------------+

| |

+------------------------------+

1 row in set (0.01 sec)

mysql> **SELECT ExtractValue(@xml, '/descendant-or-self::\*/b[1]');**

+---------------------------------------------------+

| ExtractValue(@xml, '/descendant-or-self::\*/b[1]') |

+---------------------------------------------------+

| x z |

+---------------------------------------------------+

1 row in set (0.06 sec)

mysql> **SELECT ExtractValue(@xml, '/descendant-or-self::\*/b[2]');**

+---------------------------------------------------+

| ExtractValue(@xml, '/descendant-or-self::\*/b[2]') |

+---------------------------------------------------+

| |

+---------------------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT ExtractValue(@xml, '/descendant-or-self::b[1]');**

+-------------------------------------------------+

| ExtractValue(@xml, '/descendant-or-self::b[1]') |

+-------------------------------------------------+

| z |

+-------------------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT ExtractValue(@xml, '/descendant-or-self::b[2]');**

+-------------------------------------------------+

| ExtractValue(@xml, '/descendant-or-self::b[2]') |

+-------------------------------------------------+

| x |

+-------------------------------------------------+

1 row in set (0.00 sec)

The **\*** operator acts as a “wildcard” that matches any element. For example, the expression **/\*/b** matches the ***b*** element in either of the XML fragments **<a><b/></a>** or **<c><b/></c>**. However, the expression does not produce a match in the fragment **<b><a/></b>** because ***b*** must be a child of some other element. The wildcard may be used in any position: The expression **/\*/b/\*** matches any child of a ***b*** element that is itself not the root element.

You can match any of several locators using the **|** ([**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union)) operator. For example, the expression **//b|//c** matches all ***b*** and ***c*** elements in the XML target.

It is also possible to match an element based on the value of one or more of its attributes. This done using the syntax ***tag*[@*attribute*="*value*"]**. For example, the expression **//b[@id="idB"]** matches the second ***b*** element in the fragment **<a><b id="idA"/><c/><b id="idB"/></a>**. To match against any element having ***attribute*="*value*"**, use the XPath expression **//\*[*attribute*="*value*"]**.

To filter multiple attribute values, simply use multiple attribute-comparison clauses in succession. For example, the expression **//b[@c="x"][@d="y"]** matches the element **<b c="x" d="y"/>** occurring anywhere in a given XML fragment.

To find elements for which the same attribute matches any of several values, you can use multiple locators joined by the **|** operator. For example, to match all ***b*** elements whose ***c*** attributes have either of the values 23 or 17, use the expression **//b[@c="23"]|//b[@c="17"]**. You can also use the logical **or** operator for this purpose: **//b[@c="23" or @c="17"]**.

**Note**

The difference between **or** and **|** is that **or** joins conditions, while **|** joins result sets.

**XPath Limitations.** The XPath syntax supported by these functions is currently subject to the following limitations:

Nodeset-to-nodeset comparison (such as **'/a/b[@c=@d]'**) is not supported.

All of the standard XPath comparison operators are supported. (Bug #22823)

Relative locator expressions are resolved in the context of the root node. For example, consider the following query and result:

mysql> **SELECT ExtractValue(**

-> **'<a><b c="1">X</b><b c="2">Y</b></a>',**

-> **'a/b'**

-> **) AS result;**

+--------+

| result |

+--------+

| X Y |

+--------+

1 row in set (0.03 sec)

In this case, the locator **a/b** resolves to **/a/b**.

Relative locators are also supported within predicates. In the following example, **d[../@c="1"]** is resolved as **/a/b[@c="1"]/d**:

mysql> **SELECT ExtractValue(**

-> **'<a>**

-> **<b c="1"><d>X</d></b>**

-> **<b c="2"><d>X</d></b>**

-> **</a>',**

-> **'a/b/d[../@c="1"]')**

-> **AS result;**

+--------+

| result |

+--------+

| X |

+--------+

1 row in set (0.00 sec)

Locators prefixed with expressions that evaluate as scalar values—including variable references, literals, numbers, and scalar function calls—are not permitted, and their use results in an error.

The **::** operator is not supported in combination with node types such as the following:

***axis*::comment()**

***axis*::text()**

***axis*::processing-instructions()**

***axis*::node()**

However, name tests (such as ***axis*::*name*** and ***axis*::\***) are supported, as shown in these examples:

mysql> **SELECT ExtractValue('<a><b>x</b><c>y</c></a>','/a/child::b');**

+-------------------------------------------------------+

| ExtractValue('<a><b>x</b><c>y</c></a>','/a/child::b') |

+-------------------------------------------------------+

| x |

+-------------------------------------------------------+

1 row in set (0.02 sec)

mysql> **SELECT ExtractValue('<a><b>x</b><c>y</c></a>','/a/child::\*');**

+-------------------------------------------------------+

| ExtractValue('<a><b>x</b><c>y</c></a>','/a/child::\*') |

+-------------------------------------------------------+

| x y |

+-------------------------------------------------------+

1 row in set (0.01 sec)

“Up-and-down” navigation is not supported in cases where the path would lead “above” the root element. That is, you cannot use expressions which match on descendants of ancestors of a given element, where one or more of the ancestors of the current element is also an ancestor of the root element (see Bug #16321).

The following XPath functions are not supported, or have known issues as indicated:

**id()**

**lang()**

**local-name()**

**name()**

**namespace-uri()**

**normalize-space()**

**starts-with()**

**string()**

**substring-after()**

**substring-before()**

**translate()**

The following axes are not supported:

**following-sibling**

**following**

**preceding-sibling**

**preceding**

XPath expressions passed as arguments to [**ExtractValue()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extractvalue) and [**UpdateXML()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_updatexml) may contain the colon character (**:**) in element selectors, which enables their use with markup employing XML namespaces notation. For example:

mysql> **SET @xml = '<a>111<b:c>222<d>333</d><e:f>444</e:f></b:c></a>';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT ExtractValue(@xml, '//e:f');**

+-----------------------------+

| ExtractValue(@xml, '//e:f') |

+-----------------------------+

| 444 |

+-----------------------------+

1 row in set (0.00 sec)

mysql> **SELECT UpdateXML(@xml, '//b:c', '<g:h>555</g:h>');**

+--------------------------------------------+

| UpdateXML(@xml, '//b:c', '<g:h>555</g:h>') |

+--------------------------------------------+

| <a>111<g:h>555</g:h></a> |

+--------------------------------------------+

1 row in set (0.00 sec)

This is similar in some respects to what is permitted by [Apache Xalan](http://xalan.apache.org/) and some other parsers, and is much simpler than requiring namespace declarations or the use of the **namespace-uri()** and **local-name()** functions.

**Error handling.** For both [**ExtractValue()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extractvalue) and [**UpdateXML()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_updatexml), the XPath locator used must be valid and the XML to be searched must consist of elements which are properly nested and closed. If the locator is invalid, an error is generated:

mysql> **SELECT ExtractValue('<a>c</a><b/>', '/&a');**

ERROR 1105 (HY000): XPATH syntax error: '&a'

If ***xml\_frag*** does not consist of elements which are properly nested and closed, **NULL** is returned and a warning is generated, as shown in this example:

mysql> **SELECT ExtractValue('<a>c</a><b', '//a');**

+-----------------------------------+

| ExtractValue('<a>c</a><b', '//a') |

+-----------------------------------+

| NULL |

+-----------------------------------+

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Warning

Code: 1525

Message: Incorrect XML value: 'parse error at line 1 pos 11:

END-OF-INPUT unexpected ('>' wanted)'

1 row in set (0.00 sec)

mysql> **SELECT ExtractValue('<a>c</a><b/>', '//a');**

+-------------------------------------+

| ExtractValue('<a>c</a><b/>', '//a') |

+-------------------------------------+

| c |

+-------------------------------------+

1 row in set (0.00 sec)

**Important**

The replacement XML used as the third argument to [**UpdateXML()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_updatexml) is not checked to determine whether it consists solely of elements which are properly nested and closed.

**XPath Injection.** code injection occurs when malicious code is introduced into the system to gain unauthorized access to privileges and data. It is based on exploiting assumptions made by developers about the type and content of data input from users. XPath is no exception in this regard.

A common scenario in which this can happen is the case of application which handles authorization by matching the combination of a login name and password with those found in an XML file, using an XPath expression like this one:

//user[login/text()='neapolitan' and password/text()='1c3cr34m']/attribute::id

This is the XPath equivalent of an SQL statement like this one:

SELECT id FROM users WHERE login='neapolitan' AND password='1c3cr34m';

A PHP application employing XPath might handle the login process like this:

<?php

$file = "users.xml";

$login = $POST["login"];

$password = $POST["password"];

$xpath = "//user[login/text()=$login and password/text()=$password]/attribute::id";

if( file\_exists($file) )

{

$xml = simplexml\_load\_file($file);

if($result = $xml->xpath($xpath))

echo "You are now logged in as user $result[0].";

else

echo "Invalid login name or password.";

}

else

exit("Failed to open $file.");

?>

No checks are performed on the input. This means that a malevolent user can “short-circuit” the test by entering **' or 1=1** for both the login name and password, resulting in $xpath being evaluated as shown here:

//user[login/text()='' or 1=1 and password/text()='' or 1=1]/attribute::id

Since the expression inside the square brackets always evaluates as **true**, it is effectively the same as this one, which matches the **id** attribute of every **user** element in the XML document:

//user/attribute::id

One way in which this particular attack can be circumvented is simply by quoting the variable names to be interpolated in the definition of **$xpath**, forcing the values passed from a Web form to be converted to strings:

$xpath = "//user[login/text()='$login' and password/text()='$password']/attribute::id";

This is the same strategy that is often recommended for preventing SQL injection attacks. In general, the practices you should follow for preventing XPath injection attacks are the same as for preventing SQL injection:

Never accepted untested data from users in your application.

Check all user-submitted data for type; reject or convert data that is of the wrong type

Test numeric data for out of range values; truncate, round, or reject values that are out of range. Test strings for illegal characters and either strip them out or reject input containing them.

Do not output explicit error messages that might provide an unauthorized user with clues that could be used to compromise the system; log these to a file or database table instead.

Just as SQL injection attacks can be used to obtain information about database schemas, so can XPath injection be used to traverse XML files to uncover their structure, as discussed in Amit Klein's paper [Blind XPath Injection](http://www.packetstormsecurity.org/papers/bypass/Blind_XPath_Injection_20040518.pdf) (PDF file, 46KB).

It is also important to check the output being sent back to the client. Consider what can happen when we use the MySQL [**ExtractValue()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extractvalue) function:

mysql> **SELECT ExtractValue(**

-> **LOAD\_FILE('users.xml'),**

-> **'//user[login/text()="" or 1=1 and password/text()="" or 1=1]/attribute::id'**

-> **) AS id;**

+-------------------------------+

| id |

+-------------------------------+

| 00327 13579 02403 42354 28570 |

+-------------------------------+

1 row in set (0.01 sec)

Because [**ExtractValue()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_extractvalue) returns multiple matches as a single space-delimited string, this injection attack provides every valid ID contained within users.xml to the user as a single row of output. As an extra safeguard, you should also test output before returning it to the user. Here is a simple example:

mysql> **SELECT @id = ExtractValue(**

-> **LOAD\_FILE('users.xml'),**

-> **'//user[login/text()="" or 1=1 and password/text()="" or 1=1]/attribute::id'**

-> **);**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT IF(**

-> **INSTR(@id, ' ') = 0,**

-> **@id,**

-> **'Unable to retrieve user ID')**

-> **AS singleID;**

+----------------------------+

| singleID |

+----------------------------+

| Unable to retrieve user ID |

+----------------------------+

1 row in set (0.00 sec)

In general, the guidelines for returning data to users securely are the same as for accepting user input. These can be summed up as:

Always test outgoing data for type and permissible values.

Never permit unauthorized users to view error messages that might provide information about the application that could be used to exploit it.

## 12.13 Bit Functions and Operators

**Table 12.17 Bit Functions and Operators**

| **Name** | **Description** |
| --- | --- |
| [**&**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-and) | Bitwise AND |
| [**>>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_right-shift) | Right shift |
| [**<<**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_left-shift) | Left shift |
| [**^**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-xor) | Bitwise XOR |
| [**BIT\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-count) | Return the number of bits that are set |
| [**|**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-or) | Bitwise OR |
| [**~**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-invert) | Bitwise inversion |

Bit functions and operators comprise [**BIT\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-count), [**BIT\_AND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-and), [**BIT\_OR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-or), [**BIT\_XOR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-xor), [**&**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-and), [**|**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-or), [**^**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-xor), [**~**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-invert), [**<<**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_left-shift), and [**>>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_right-shift). (The [**BIT\_AND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-and), [**BIT\_OR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-or), and [**BIT\_XOR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-xor) aggregate functions are described in [Section 12.20.1, “Aggregate Function Descriptions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#aggregate-functions).) Prior to MySQL 8.0, bit functions and operators required [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) (64-bit integer) arguments and returned [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) values, so they had a maximum range of 64 bits. Non-[**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) arguments were converted to [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) prior to performing the operation and truncation could occur.

In MySQL 8.0, bit functions and operators permit binary string type arguments ([**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), and the [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types) and return a value of like type, which enables them to take arguments and produce return values larger than 64 bits. Nonbinary string arguments are converted to [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) and processed as such, as before.

An implication of this change in behavior is that bit operations on binary string arguments might produce a different result in MySQL 8.0 than in 5.7. For information about how to prepare in MySQL 5.7 for potential incompatibilities between MySQL 5.7 and 8.0, see [Bit Functions and Operators](https://dev.mysql.com/doc/refman/5.7/en/bit-functions.html), in [MySQL 5.7 Reference Manual](https://dev.mysql.com/doc/refman/5.7/en/).

[Bit Operations Prior to MySQL 8.0](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-operations-5-7)

[Bit Operations in MySQL 8.0](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-operations-8-0)

[Binary String Bit-Operation Examples](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-operations-binary-string-examples)

[Bitwise AND, OR, and XOR Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-operations-and-or-xor)

[Bitwise Complement and Shift Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-operations-complement-shift)

[BIT\_COUNT() Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-operations-bit-count)

[BIT\_AND(), BIT\_OR(), and BIT\_XOR() Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-operations-bit-aggregate)

[Special Handling of Hexadecimal Literals, Bit Literals, and NULL Literals](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-operations-literal-handling)

[Bit-Operation Incompatibilities with MySQL 5.7](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-operations-incompatibilities)

### Bit Operations Prior to MySQL 8.0

Bit operations prior to MySQL 8.0 handle only unsigned 64-bit integer argument and result values (that is, unsigned [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) values). Conversion of arguments of other types to [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) occurs as necessary. Examples:

This statement operates on numeric literals, treated as unsigned 64-bit integers:

mysql> **SELECT 127 | 128, 128 << 2, BIT\_COUNT(15);**

+-----------+----------+---------------+

| 127 | 128 | 128 << 2 | BIT\_COUNT(15) |

+-----------+----------+---------------+

| 255 | 512 | 4 |

+-----------+----------+---------------+

This statement performs to-number conversions on the string arguments (**'127'** to **127**, and so forth) before performing the same operations as the first statement and producing the same results:

mysql> **SELECT '127' | '128', '128' << 2, BIT\_COUNT('15');**

+---------------+------------+-----------------+

| '127' | '128' | '128' << 2 | BIT\_COUNT('15') |

+---------------+------------+-----------------+

| 255 | 512 | 4 |

+---------------+------------+-----------------+

This statement uses hexadecimal literals for the bit-operation arguments. MySQL by default treats hexadecimal literals as binary strings, but in numeric context evaluates them as numbers (see [Section 9.1.4, “Hexadecimal Literals”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#hexadecimal-literals)). Prior to MySQL 8.0, numeric context includes bit operations. Examples:

mysql> **SELECT X'7F' | X'80', X'80' << 2, BIT\_COUNT(X'0F');**

+---------------+------------+------------------+

| X'7F' | X'80' | X'80' << 2 | BIT\_COUNT(X'0F') |

+---------------+------------+------------------+

| 255 | 512 | 4 |

+---------------+------------+------------------+

Handling of bit-value literals in bit operations is similar to hexadecimal literals (that is, as numbers).

### Bit Operations in MySQL 8.0

MySQL 8.0 extends bit operations to handle binary string arguments directly (without conversion) and produce binary string results. (Arguments that are not integers or binary strings are still converted to integers, as before.) This extension enhances bit operations in the following ways:

Bit operations become possible on values longer than 64 bits.

It is easier to perform bit operations on values that are more naturally represented as binary strings than as integers.

For example, consider UUID values and IPv6 addresses, which have human-readable text formats like this:

UUID: 6ccd780c-baba-1026-9564-5b8c656024db

IPv6: fe80::219:d1ff:fe91:1a72

It is cumbersome to operate on text strings in those formats. An alternative is convert them to fixed-length binary strings without delimiters. [**UUID\_TO\_BIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin) and [**INET6\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-aton) each produce a value of data type [**BINARY(16)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), a binary string 16 bytes (128 bits) long. The following statements illustrate this (**HEX()** is used to produce displayable values):

mysql> **SELECT HEX(UUID\_TO\_BIN('6ccd780c-baba-1026-9564-5b8c656024db'));**

+----------------------------------------------------------+

| HEX(UUID\_TO\_BIN('6ccd780c-baba-1026-9564-5b8c656024db')) |

+----------------------------------------------------------+

| 6CCD780CBABA102695645B8C656024DB |

+----------------------------------------------------------+

mysql> **SELECT HEX(INET6\_ATON('fe80::219:d1ff:fe91:1a72'));**

+---------------------------------------------+

| HEX(INET6\_ATON('fe80::219:d1ff:fe91:1a72')) |

+---------------------------------------------+

| FE800000000000000219D1FFFE911A72 |

+---------------------------------------------+

Those binary values are easily manipulable with bit operations to perform actions such as extracting the timestamp from UUID values, or extracting the network and host parts of IPv6 addresses. (For examples, see later in this discussion.)

Arguments that count as binary strings include column values, routine parameters, local variables, and user-defined variables that have a binary string type: [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), or one of the [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types.

What about hexadecimal literals and bit literals? Recall that those are binary strings by default in MySQL, but numbers in numeric context. How are they handled for bit operations in MySQL 8.0? Does MySQL continue to evaluate them in numeric context, as is done prior to MySQL 8.0? Or do bit operations evaluate them as binary strings, now that binary strings can be handled “natively” without conversion?

Answer: It has been common to specify arguments to bit operations using hexadecimal literals or bit literals with the intent that they represent numbers, so MySQL continues to evaluate bit operations in numeric context when all bit arguments are hexadecimal or bit literals, for backward compatility. If you require evaluation as binary strings instead, that is easily accomplished: Use the **\_binary** introducer for at least one literal.

These bit operations evaluate the hexadecimal literals and bit literals as integers:

mysql> **SELECT X'40' | X'01', b'11110001' & b'01001111';**

+---------------+---------------------------+

| X'40' | X'01' | b'11110001' & b'01001111' |

+---------------+---------------------------+

| 65 | 65 |

+---------------+---------------------------+

These bit operations evaluate the hexadecimal literals and bit literals as binary strings, due to the **\_binary** introducer:

mysql> **SELECT \_binary X'40' | X'01', b'11110001' & \_binary b'01001111';**

+-----------------------+-----------------------------------+

| \_binary X'40' | X'01' | b'11110001' & \_binary b'01001111' |

+-----------------------+-----------------------------------+

| A | A |

+-----------------------+-----------------------------------+

Although the bit operations in both statements produce a result with a numeric value of 65, the second statement operates in binary-string context, for which 65 is ASCII **A**.

In numeric evaluation context, permitted values of hexadecimal literal and bit literal arguments have a maximum of 64 bits, as do results. By contrast, in binary-string evaluation context, permitted arguments (and results) can exceed 64 bits:

mysql> **SELECT \_binary X'4040404040404040' | X'0102030405060708';**

+---------------------------------------------------+

| \_binary X'4040404040404040' | X'0102030405060708' |

+---------------------------------------------------+

| ABCDEFGH |

+---------------------------------------------------+

There are several ways to refer to a hexadecimal literal or bit literal in a bit operation to cause binary-string evaluation:

\_binary ***literal***

BINARY ***literal***

CAST(***literal*** AS BINARY)

Another way to produce binary-string evaluation of hexadecimal literals or bit literals is to assign them to user-defined variables, which results in variables that have a binary string type:

mysql> **SET @v1 = X'40', @v2 = X'01', @v3 = b'11110001', @v4 = b'01001111';**

mysql> **SELECT @v1 | @v2, @v3 & @v4;**

+-----------+-----------+

| @v1 | @v2 | @v3 & @v4 |

+-----------+-----------+

| A | A |

+-----------+-----------+

In binary-string context, bitwise operation arguments must have the same length or an [**ER\_INVALID\_BITWISE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_operands_size) error occurs:

mysql> **SELECT \_binary X'40' | X'0001';**

ERROR 3513 (HY000): Binary operands of bitwise

operators must be of equal length

To satisfy the equal-length requirement, pad the shorter value with leading zero digits or, if the longer value begins with leading zero digits and a shorter result value is acceptable, strip them:

mysql> **SELECT \_binary X'0040' | X'0001';**

+---------------------------+

| \_binary X'0040' | X'0001' |

+---------------------------+

| A |

+---------------------------+

mysql> **SELECT \_binary X'40' | X'01';**

+-----------------------+

| \_binary X'40' | X'01' |

+-----------------------+

| A |

+-----------------------+

Padding or stripping can also be accomplished using functions such as [**LPAD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lpad), [**RPAD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rpad), [**SUBSTR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substr), or [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast). In such cases, the expression arguments are no longer all literals and **\_binary** becomes unnecessary. Examples:

mysql> **SELECT LPAD(X'40', 2, X'00') | X'0001';**

+---------------------------------+

| LPAD(X'40', 2, X'00') | X'0001' |

+---------------------------------+

| A |

+---------------------------------+

mysql> **SELECT X'40' | SUBSTR(X'0001', 2, 1);**

+-------------------------------+

| X'40' | SUBSTR(X'0001', 2, 1) |

+-------------------------------+

| A |

+-------------------------------+

### Binary String Bit-Operation Examples

The following example illustrates use of bit operations to extract parts of a UUID value, in this case, the timestamp and IEEE 802 node number. This technique requires bitmasks for each extracted part.

Convert the text UUID to the corresponding 16-byte binary value so that it can be manipulated using bit operations in binary-string context:

mysql> **SET @uuid = UUID\_TO\_BIN('6ccd780c-baba-1026-9564-5b8c656024db');**

mysql> **SELECT HEX(@uuid);**

+----------------------------------+

| HEX(@uuid) |

+----------------------------------+

| 6CCD780CBABA102695645B8C656024DB |

+----------------------------------+

Construct bitmasks for the timestamp and node number parts of the value. The timestamp comprises the first three parts (64 bits, bits 0 to 63) and the node number is the last part (48 bits, bits 80 to 127):

mysql> **SET @ts\_mask = CAST(X'FFFFFFFFFFFFFFFF' AS BINARY(16));**

mysql> **SET @node\_mask = CAST(X'FFFFFFFFFFFF' AS BINARY(16)) >> 80;**

mysql> **SELECT HEX(@ts\_mask);**

+----------------------------------+

| HEX(@ts\_mask) |

+----------------------------------+

| FFFFFFFFFFFFFFFF0000000000000000 |

+----------------------------------+

mysql> **SELECT HEX(@node\_mask);**

+----------------------------------+

| HEX(@node\_mask) |

+----------------------------------+

| 00000000000000000000FFFFFFFFFFFF |

+----------------------------------+

The **CAST(... AS BINARY(16))** function is used here because the masks must be the same length as the UUID value against which they are applied. The same result can be produced using other functions to pad the masks to the required length:

SET @ts\_mask= RPAD(X'FFFFFFFFFFFFFFFF' , 16, X'00');

SET @node\_mask = LPAD(X'FFFFFFFFFFFF', 16, X'00') ;

Use the masks to extract the timestamp and node number parts:

mysql> **SELECT HEX(@uuid & @ts\_mask) AS 'timestamp part';**

+----------------------------------+

| timestamp part |

+----------------------------------+

| 6CCD780CBABA10260000000000000000 |

+----------------------------------+

mysql> **SELECT HEX(@uuid & @node\_mask) AS 'node part';**

+----------------------------------+

| node part |

+----------------------------------+

| 000000000000000000005B8C656024DB |

+----------------------------------+

The preceding example uses these bit operations: right shift ([**>>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_right-shift)) and bitwise AND ([**&**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-and)).

**Note**

[**UUID\_TO\_BIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin) takes a flag that causes some bit rearrangement in the resulting binary UUID value. If you use that flag, modify the extraction masks accordingly.

The next example uses bit operations to extract the network and host parts of an IPv6 address. Suppose that the network part has a length of 80 bits. Then the host part has a length of 128 − 80 = 48 bits. To extract the network and host parts of the address, convert it to a binary string, then use bit operations in binary-string context.

Convert the text IPv6 address to the corresponding binary string:

mysql> **SET @ip = INET6\_ATON('fe80::219:d1ff:fe91:1a72');**

Define the network length in bits:

mysql> **SET @net\_len = 80;**

Construct network and host masks by shifting the all-ones address left or right. To do this, begin with the address **::**, which is shorthand for all zeros, as you can see by converting it to a binary string like this:

mysql> **SELECT HEX(INET6\_ATON('::')) AS 'all zeros';**

+----------------------------------+

| all zeros |

+----------------------------------+

| 00000000000000000000000000000000 |

+----------------------------------+

To produce the complementary value (all ones), use the [**~**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-invert) operator to invert the bits:

mysql> **SELECT HEX(~INET6\_ATON('::')) AS 'all ones';**

+----------------------------------+

| all ones |

+----------------------------------+

| FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF |

+----------------------------------+

Shift the all-ones value left or right to produce the network and host masks:

mysql> **SET @net\_mask = ~INET6\_ATON('::') << (128 - @net\_len);**

mysql> **SET @host\_mask = ~INET6\_ATON('::') >> @net\_len;**

Display the masks to verify that they cover the correct parts of the address:

mysql> **SELECT INET6\_NTOA(@net\_mask) AS 'network mask';**

+----------------------------+

| network mask |

+----------------------------+

| ffff:ffff:ffff:ffff:ffff:: |

+----------------------------+

mysql> **SELECT INET6\_NTOA(@host\_mask) AS 'host mask';**

+------------------------+

| host mask |

+------------------------+

| ::ffff:255.255.255.255 |

+------------------------+

Extract and display the network and host parts of the address:

mysql> **SET @net\_part = @ip & @net\_mask;**

mysql> **SET @host\_part = @ip & @host\_mask;**

mysql> **SELECT INET6\_NTOA(@net\_part) AS 'network part';**

+-----------------+

| network part |

+-----------------+

| fe80::219:0:0:0 |

+-----------------+

mysql> **SELECT INET6\_NTOA(@host\_part) AS 'host part';**

+------------------+

| host part |

+------------------+

| ::d1ff:fe91:1a72 |

+------------------+

The preceding example uses these bit operations: Complement ([**~**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-invert)), left shift ([**<<**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_left-shift)), and bitwise AND ([**&**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-and)).

The remaining discussion provides details on argument handling for each group of bit operations, more information about literal-value handling in bit operations, and potential incompatibilities between MySQL 8.0 and older MySQL versions.

### Bitwise AND, OR, and XOR Operations

For [**&**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-and), [**|**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-or), and [**^**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-xor) bit operations, the result type depends on whether the arguments are evaluated as binary strings or numbers:

Binary-string evaluation occurs when the arguments have a binary string type, and at least one of them is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument conversion to unsigned 64-bit integers as necessary.

Binary-string evaluation produces a binary string of the same length as the arguments. If the arguments have unequal lengths, an [**ER\_INVALID\_BITWISE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_operands_size) error occurs. Numeric evaluation produces an unsigned 64-bit integer.

Examples of numeric evaluation:

mysql> **SELECT 64 | 1, X'40' | X'01';**

+--------+---------------+

| 64 | 1 | X'40' | X'01' |

+--------+---------------+

| 65 | 65 |

+--------+---------------+

Examples of binary-string evaluation:

mysql> **SELECT \_binary X'40' | X'01';**

+-----------------------+

| \_binary X'40' | X'01' |

+-----------------------+

| A |

+-----------------------+

mysql> **SET @var1 = X'40', @var2 = X'01';**

mysql> **SELECT @var1 | @var2;**

+---------------+

| @var1 | @var2 |

+---------------+

| A |

+---------------+

### Bitwise Complement and Shift Operations

For [**~**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-invert), [**<<**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_left-shift), and [**>>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_right-shift) bit operations, the result type depends on whether the bit argument is evaluated as a binary string or number:

Binary-string evaluation occurs when the bit argument has a binary string type, and is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument conversion to an unsigned 64-bit integer as necessary.

Binary-string evaluation produces a binary string of the same length as the bit argument. Numeric evaluation produces an unsigned 64-bit integer.

For shift operations, bits shifted off the end of the value are lost without warning, regardless of the argument type. In particular, if the shift count is greater or equal to the number of bits in the bit argument, all bits in the result are 0.

Examples of numeric evaluation:

mysql> **SELECT ~0, 64 << 2, X'40' << 2;**

+----------------------+---------+------------+

| ~0 | 64 << 2 | X'40' << 2 |

+----------------------+---------+------------+

| 18446744073709551615 | 256 | 256 |

+----------------------+---------+------------+

Examples of binary-string evaluation:

mysql> **SELECT HEX(\_binary X'1111000022220000' >> 16);**

+----------------------------------------+

| HEX(\_binary X'1111000022220000' >> 16) |

+----------------------------------------+

| 0000111100002222 |

+----------------------------------------+

mysql> **SELECT HEX(\_binary X'1111000022220000' << 16);**

+----------------------------------------+

| HEX(\_binary X'1111000022220000' << 16) |

+----------------------------------------+

| 0000222200000000 |

+----------------------------------------+

mysql> **SET @var1 = X'F0F0F0F0';**

mysql> **SELECT HEX(~@var1);**

+-------------+

| HEX(~@var1) |

+-------------+

| 0F0F0F0F |

+-------------+

### BIT\_COUNT() Operations

The [**BIT\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-count) function always returns an unsigned 64-bit integer, or **NULL** if the argument is **NULL**.

mysql> **SELECT BIT\_COUNT(127);**

+----------------+

| BIT\_COUNT(127) |

+----------------+

| 7 |

+----------------+

mysql> **SELECT BIT\_COUNT(b'010101'), BIT\_COUNT(\_binary b'010101');**

+----------------------+------------------------------+

| BIT\_COUNT(b'010101') | BIT\_COUNT(\_binary b'010101') |

+----------------------+------------------------------+

| 3 | 3 |

+----------------------+------------------------------+

### BIT\_AND(), BIT\_OR(), and BIT\_XOR() Operations

For the [**BIT\_AND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-and), [**BIT\_OR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-or), and [**BIT\_XOR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-xor) bit functions, the result type depends on whether the function argument values are evaluated as binary strings or numbers:

Binary-string evaluation occurs when the argument values have a binary string type, and the argument is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument value conversion to unsigned 64-bit integers as necessary.

Binary-string evaluation produces a binary string of the same length as the argument values. If argument values have unequal lengths, an [**ER\_INVALID\_BITWISE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_operands_size) error occurs. If the argument size exceeds 511 bytes, an [**ER\_INVALID\_BITWISE\_AGGREGATE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_aggregate_operands_size) error occurs. Numeric evaluation produces an unsigned 64-bit integer.

**NULL** values do not affect the result unless all values are **NULL**. In that case, the result is a neutral value having the same length as the length of the argument values (all bits 1 for [**BIT\_AND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-and), all bits 0 for [**BIT\_OR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-or), and [**BIT\_XOR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-xor)).

Example:

mysql> **CREATE TABLE t (group\_id INT, a VARBINARY(6));**

mysql> **INSERT INTO t VALUES (1, NULL);**

mysql> **INSERT INTO t VALUES (1, NULL);**

mysql> **INSERT INTO t VALUES (2, NULL);**

mysql> **INSERT INTO t VALUES (2, X'1234');**

mysql> **INSERT INTO t VALUES (2, X'FF34');**

mysql> **SELECT HEX(BIT\_AND(a)), HEX(BIT\_OR(a)), HEX(BIT\_XOR(a))**

**FROM t GROUP BY group\_id;**

+-----------------+----------------+-----------------+

| HEX(BIT\_AND(a)) | HEX(BIT\_OR(a)) | HEX(BIT\_XOR(a)) |

+-----------------+----------------+-----------------+

| FFFFFFFFFFFF | 000000000000 | 000000000000 |

| 1234 | FF34 | ED00 |

+-----------------+----------------+-----------------+

### Special Handling of Hexadecimal Literals, Bit Literals, and NULL Literals

For backward compatibility, MySQL 8.0 evaluates bit operations in numeric context when all bit arguments are hexadecimal literals, bit literals, or **NULL** literals. That is, bit operations on binary-string bit arguments do not use binary-string evaluation if all bit arguments are unadorned hexadecimal literals, bit literals, or **NULL** literals. (This does not apply to such literals if they are written with a **\_binary** introducer, [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_binary) operator, or other way of specifying them explicitly as binary strings.)

The literal handling just described is the same as prior to MySQL 8.0. Examples:

These bit operations evaluate the literals in numeric context and produce a **BIGINT** result:

b'0001' | b'0010'

X'0008' << 8

These bit operations evaluate **NULL** in numeric context and produce a **BIGINT** result that has a **NULL** value:

NULL & NULL

NULL >> 4

In MySQL 8.0, you can cause those operations to evaluate the arguments in binary-string context by indicating explicitly that at least one argument is a binary string:

\_binary b'0001' | b'0010'

\_binary X'0008' << 8

BINARY NULL & NULL

BINARY NULL >> 4

The result of the last two expressions is **NULL**, just as without the **BINARY** operator, but the data type of the result is a binary string type rather than an integer type.

### Bit-Operation Incompatibilities with MySQL 5.7

Because bit operations can handle binary string arguments natively in MySQL 8.0, some expressions produce a different result in MySQL 8.0 than in 5.7. The five problematic expression types to watch out for are:

***nonliteral\_binary*** { & | ^ } ***binary***

***binary*** { & | ^ } ***nonliteral\_binary***

***nonliteral\_binary*** { << >> } ***anything***

~ ***nonliteral\_binary***

***AGGR\_BIT\_FUNC***(***nonliteral\_binary***)

Those expressions return [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) in MySQL 5.7, binary string in 8.0.

Explanation of notation:

**{ *op1* *op2* ... }**: List of operators that apply to the given expression type.

***binary***: Any kind of binary string argument, including a hexadecimal literal, bit literal, or **NULL** literal.

***nonliteral\_binary***: An argument that is a binary string value other than a hexadecimal literal, bit literal, or **NULL** literal.

***AGGR\_BIT\_FUNC***: An aggregate function that takes bit-value arguments: [**BIT\_AND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-and), [**BIT\_OR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-or), [**BIT\_XOR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-xor).

For information about how to prepare in MySQL 5.7 for potential incompatibilities between MySQL 5.7 and 8.0, see [Bit Functions and Operators](https://dev.mysql.com/doc/refman/5.7/en/bit-functions.html), in [MySQL 5.7 Reference Manual](https://dev.mysql.com/doc/refman/5.7/en/).

The following list describes available bit functions and operators:

**[|](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_bitwise-or)**

Bitwise OR.

The result type depends on whether the arguments are evaluated as binary strings or numbers:

Binary-string evaluation occurs when the arguments have a binary string type, and at least one of them is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument conversion to unsigned 64-bit integers as necessary.

Binary-string evaluation produces a binary string of the same length as the arguments. If the arguments have unequal lengths, an [**ER\_INVALID\_BITWISE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_operands_size) error occurs. Numeric evaluation produces an unsigned 64-bit integer.

For more information, see the introductory discussion in this section.

mysql> **SELECT 29 | 15;**

-> 31

mysql> **SELECT \_binary X'40404040' | X'01020304';**

-> 'ABCD'

**[&](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_bitwise-and)**

Bitwise AND.

The result type depends on whether the arguments are evaluated as binary strings or numbers:

Binary-string evaluation occurs when the arguments have a binary string type, and at least one of them is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument conversion to unsigned 64-bit integers as necessary.

Binary-string evaluation produces a binary string of the same length as the arguments. If the arguments have unequal lengths, an [**ER\_INVALID\_BITWISE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_operands_size) error occurs. Numeric evaluation produces an unsigned 64-bit integer.

For more information, see the introductory discussion in this section.

mysql> **SELECT 29 & 15;**

-> 13

mysql> **SELECT HEX(\_binary X'FF' & b'11110000');**

-> 'F0'

**[^](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_bitwise-xor)**

Bitwise XOR.

The result type depends on whether the arguments are evaluated as binary strings or numbers:

Binary-string evaluation occurs when the arguments have a binary string type, and at least one of them is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument conversion to unsigned 64-bit integers as necessary.

Binary-string evaluation produces a binary string of the same length as the arguments. If the arguments have unequal lengths, an [**ER\_INVALID\_BITWISE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_operands_size) error occurs. Numeric evaluation produces an unsigned 64-bit integer.

For more information, see the introductory discussion in this section.

mysql> **SELECT 1 ^ 1;**

-> 0

mysql> **SELECT 1 ^ 0;**

-> 1

mysql> **SELECT 11 ^ 3;**

-> 8

mysql> **SELECT HEX(\_binary X'FEDC' ^ X'1111');**

-> 'EFCD'

**[<<](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_left-shift)**

Shifts a longlong ([**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types)) number or binary string to the left.

The result type depends on whether the bit argument is evaluated as a binary string or number:

Binary-string evaluation occurs when the bit argument has a binary string type, and is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument conversion to an unsigned 64-bit integer as necessary.

Binary-string evaluation produces a binary string of the same length as the bit argument. Numeric evaluation produces an unsigned 64-bit integer.

Bits shifted off the end of the value are lost without warning, regardless of the argument type. In particular, if the shift count is greater or equal to the number of bits in the bit argument, all bits in the result are 0.

For more information, see the introductory discussion in this section.

mysql> **SELECT 1 << 2;**

-> 4

mysql> **SELECT HEX(\_binary X'00FF00FF00FF' << 8);**

-> 'FF00FF00FF00'

**[>>](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_right-shift)**

Shifts a longlong ([**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types)) number or binary string to the right.

The result type depends on whether the bit argument is evaluated as a binary string or number:

Binary-string evaluation occurs when the bit argument has a binary string type, and is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument conversion to an unsigned 64-bit integer as necessary.

Binary-string evaluation produces a binary string of the same length as the bit argument. Numeric evaluation produces an unsigned 64-bit integer.

Bits shifted off the end of the value are lost without warning, regardless of the argument type. In particular, if the shift count is greater or equal to the number of bits in the bit argument, all bits in the result are 0.

For more information, see the introductory discussion in this section.

mysql> **SELECT 4 >> 2;**

-> 1

mysql> **SELECT HEX(\_binary X'00FF00FF00FF' >> 8);**

-> '0000FF00FF00'

**[~](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_bitwise-invert)**

Invert all bits.

The result type depends on whether the bit argument is evaluated as a binary string or number:

Binary-string evaluation occurs when the bit argument has a binary string type, and is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument conversion to an unsigned 64-bit integer as necessary.

Binary-string evaluation produces a binary string of the same length as the bit argument. Numeric evaluation produces an unsigned 64-bit integer.

For more information, see the introductory discussion in this section.

mysql> **SELECT 5 & ~1;**

-> 4

mysql> **SELECT HEX(~X'0000FFFF1111EEEE');**

-> 'FFFF0000EEEE1111'

**[BIT\_COUNT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-count)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-count)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-count)**

Returns the number of bits that are set in the argument ***N*** as an unsigned 64-bit integer, or **NULL** if the argument is **NULL**.

mysql> **SELECT BIT\_COUNT(64), BIT\_COUNT(BINARY 64);**

-> 1, 7

mysql> **SELECT BIT\_COUNT('64'), BIT\_COUNT(\_binary '64');**

-> 1, 7

mysql> **SELECT BIT\_COUNT(X'40'), BIT\_COUNT(\_binary X'40');**

-> 1, 1

## 12.14 Encryption and Compression Functions

**Table 12.18 Encryption Functions**

| **Name** | **Description** |
| --- | --- |
| [**AES\_DECRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-decrypt) | Decrypt using AES |
| [**AES\_ENCRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-encrypt) | Encrypt using AES |
| [**COMPRESS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_compress) | Return result as a binary string |
| [**MD5()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_md5) | Calculate MD5 checksum |
| [**RANDOM\_BYTES()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_random-bytes) | Return a random byte vector |
| [**SHA1()**, **SHA()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha1) | Calculate an SHA-1 160-bit checksum |
| [**SHA2()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha2) | Calculate an SHA-2 checksum |
| [**STATEMENT\_DIGEST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_statement-digest) | Compute statement digest hash value |
| [**STATEMENT\_DIGEST\_TEXT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_statement-digest-text) | Compute normalized statement digest |
| [**UNCOMPRESS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uncompress) | Uncompress a string compressed |
| [**UNCOMPRESSED\_LENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uncompressed-length) | Return the length of a string before compression |
| [**VALIDATE\_PASSWORD\_STRENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_validate-password-strength) | Determine strength of password |

Many encryption and compression functions return strings for which the result might contain arbitrary byte values. If you want to store these results, use a column with a [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) binary string data type. This avoids potential problems with trailing space removal or character set conversion that would change data values, such as may occur if you use a nonbinary string data type ([**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)).

Some encryption functions return strings of ASCII characters: [**MD5()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_md5), [**SHA()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha1), [**SHA1()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha1), [**SHA2()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha2), [**STATEMENT\_DIGEST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_statement-digest), [**STATEMENT\_DIGEST\_TEXT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_statement-digest-text). Their return value is a string that has a character set and collation determined by the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variables. This is a nonbinary string unless the character set is **binary**.

If an application stores values from a function such as [**MD5()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_md5) or [**SHA1()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha1) that returns a string of hex digits, more efficient storage and comparisons can be obtained by converting the hex representation to binary using [**UNHEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unhex) and storing the result in a [**BINARY(*N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) column. Each pair of hexadecimal digits requires one byte in binary form, so the value of ***N*** depends on the length of the hex string. ***N*** is 16 for an [**MD5()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_md5) value and 20 for a [**SHA1()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha1) value. For [**SHA2()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha2), ***N*** ranges from 28 to 32 depending on the argument specifying the desired bit length of the result.

The size penalty for storing the hex string in a [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column is at least two times, up to eight times if the value is stored in a column that uses the **utf8** character set (where each character uses 4 bytes). Storing the string also results in slower comparisons because of the larger values and the need to take character set collation rules into account.

Suppose that an application stores [**MD5()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_md5) string values in a [**CHAR(32)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column:

CREATE TABLE md5\_tbl (md5\_val CHAR(32), ...);

INSERT INTO md5\_tbl (md5\_val, ...) VALUES(MD5('abcdef'), ...);

To convert hex strings to more compact form, modify the application to use [**UNHEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unhex) and [**BINARY(16)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) instead as follows:

CREATE TABLE md5\_tbl (md5\_val BINARY(16), ...);

INSERT INTO md5\_tbl (md5\_val, ...) VALUES(UNHEX(MD5('abcdef')), ...);

Applications should be prepared to handle the very rare case that a hashing function produces the same value for two different input values. One way to make collisions detectable is to make the hash column a primary key.

**Note**

Exploits for the MD5 and SHA-1 algorithms have become known. You may wish to consider using another one-way encryption function described in this section instead, such as [**SHA2()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha2).

**Caution**

Passwords or other sensitive values supplied as arguments to encryption functions are sent as cleartext to the MySQL server unless an SSL connection is used. Also, such values appear in any MySQL logs to which they are written. To avoid these types of exposure, applications can encrypt sensitive values on the client side before sending them to the server. The same considerations apply to encryption keys. To avoid exposing these, applications can use stored procedures to encrypt and decrypt values on the server side.

**[AES\_DECRYPT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-decrypt)*[crypt\_str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-decrypt)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-decrypt)*[key\_str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-decrypt)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-decrypt)*[init\_vector](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-decrypt)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-decrypt)**

This function decrypts data using the official AES (Advanced Encryption Standard) algorithm. For more information, see the description of [**AES\_ENCRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-encrypt).

Statements that use [**AES\_DECRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-decrypt) are unsafe for statement-based replication.

**[AES\_ENCRYPT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-encrypt)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-encrypt)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-encrypt)*[key\_str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-encrypt)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-encrypt)*[init\_vector](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-encrypt)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-encrypt)**

**[AES\_ENCRYPT()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_aes-encrypt)** and [**AES\_DECRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-decrypt) implement encryption and decryption of data using the official AES (Advanced Encryption Standard) algorithm, previously known as “Rijndael.” The AES standard permits various key lengths. By default these functions implement AES with a 128-bit key length. Key lengths of 196 or 256 bits can be used, as described later. The key length is a trade off between performance and security.

[**AES\_ENCRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-encrypt) encrypts the string ***str*** using the key string ***key\_str*** and returns a binary string containing the encrypted output. [**AES\_DECRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-decrypt) decrypts the encrypted string ***crypt\_str*** using the key string ***key\_str*** and returns the original plaintext string. If either function argument is **NULL**, the function returns **NULL**.

The ***str*** and ***crypt\_str*** arguments can be any length, and padding is automatically added to ***str*** so it is a multiple of a block as required by block-based algorithms such as AES. This padding is automatically removed by the [**AES\_DECRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-decrypt) function. The length of ***crypt\_str*** can be calculated using this formula:

16 \* (trunc(***string\_length*** / 16) + 1)

For a key length of 128 bits, the most secure way to pass a key to the ***key\_str*** argument is to create a truly random 128-bit value and pass it as a binary value. For example:

INSERT INTO t

VALUES (1,AES\_ENCRYPT('text',UNHEX('F3229A0B371ED2D9441B830D21A390C3')));

A passphrase can be used to generate an AES key by hashing the passphrase. For example:

INSERT INTO t

VALUES (1,AES\_ENCRYPT('text', UNHEX(SHA2('My secret passphrase',512))));

Do not pass a password or passphrase directly to ***crypt\_str***, hash it first. Previous versions of this documentation suggested the former approach, but it is no longer recommended as the examples shown here are more secure.

If [**AES\_DECRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-decrypt) detects invalid data or incorrect padding, it returns **NULL**. However, it is possible for [**AES\_DECRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-decrypt) to return a non-**NULL** value (possibly garbage) if the input data or the key is invalid.

[**AES\_ENCRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-encrypt) and [**AES\_DECRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-decrypt) permit control of the block encryption mode and take an optional ***init\_vector*** initialization vector argument:

The [**block\_encryption\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_block_encryption_mode) system variable controls the mode for block-based encryption algorithms. Its default value is **aes-128-ecb**, which signifies encryption using a key length of 128 bits and ECB mode. For a description of the permitted values of this variable, see [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

The optional ***init\_vector*** argument provides an initialization vector for block encryption modes that require it.

For modes that require the optional ***init\_vector*** argument, it must be 16 bytes or longer (bytes in excess of 16 are ignored). An error occurs if ***init\_vector*** is missing.

For modes that do not require ***init\_vector***, it is ignored and a warning is generated if it is specified.

A random string of bytes to use for the initialization vector can be produced by calling [**RANDOM\_BYTES(16)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_random-bytes). For encryption modes that require an initialization vector, the same vector must be used for encryption and decryption.

mysql> **SET block\_encryption\_mode = 'aes-256-cbc';**

mysql> **SET @key\_str = SHA2('My secret passphrase',512);**

mysql> **SET @init\_vector = RANDOM\_BYTES(16);**

mysql> **SET @crypt\_str = AES\_ENCRYPT('text',@key\_str,@init\_vector);**

mysql> **SELECT AES\_DECRYPT(@crypt\_str,@key\_str,@init\_vector);**

+-----------------------------------------------+

| AES\_DECRYPT(@crypt\_str,@key\_str,@init\_vector) |

+-----------------------------------------------+

| text |

+-----------------------------------------------+

The following table lists each permitted block encryption mode and whether the initialization vector argument is required.

| **Block Encryption Mode** | **Initialization Vector Required** |
| --- | --- |
| ECB | No |
| CBC | Yes |
| CFB1 | Yes |
| CFB8 | Yes |
| CFB128 | Yes |
| OFB | Yes |

Statements that use [**AES\_ENCRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-encrypt) or [**AES\_DECRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-decrypt) are unsafe for statement-based replication.

**[COMPRESS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_compress)*[string\_to\_compress](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_compress)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_compress)**

Compresses a string and returns the result as a binary string. This function requires MySQL to have been compiled with a compression library such as **zlib**. Otherwise, the return value is always **NULL**. The compressed string can be uncompressed with [**UNCOMPRESS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uncompress).

mysql> **SELECT LENGTH(COMPRESS(REPEAT('a',1000)));**

-> 21

mysql> **SELECT LENGTH(COMPRESS(''));**

-> 0

mysql> **SELECT LENGTH(COMPRESS('a'));**

-> 13

mysql> **SELECT LENGTH(COMPRESS(REPEAT('a',16)));**

-> 15

The compressed string contents are stored the following way:

Empty strings are stored as empty strings.

Nonempty strings are stored as a 4-byte length of the uncompressed string (low byte first), followed by the compressed string. If the string ends with space, an extra **.** character is added to avoid problems with endspace trimming should the result be stored in a [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) or [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column. (However, use of nonbinary string data types such as [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) or [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) to store compressed strings is not recommended anyway because character set conversion may occur. Use a [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) binary string column instead.)

**[MD5(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_md5)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_md5)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_md5)**

Calculates an MD5 128-bit checksum for the string. The value is returned as a string of 32 hexadecimal digits, or **NULL** if the argument was **NULL**. The return value can, for example, be used as a hash key. See the notes at the beginning of this section about storing hash values efficiently.

The return value is a string in the connection character set.

If FIPS mode is enabled, MD5() returns **NULL**. See [Section 6.8, “FIPS Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#fips-mode).

mysql> **SELECT MD5('testing');**

-> 'ae2b1fca515949e5d54fb22b8ed95575'

This is the “RSA Data Security, Inc. MD5 Message-Digest Algorithm.”

See the note regarding the MD5 algorithm at the beginning this section.

**[RANDOM\_BYTES(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_random-bytes)*[len](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_random-bytes)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_random-bytes)**

This function returns a binary string of ***len*** random bytes generated using the random number generator of the SSL library. Permitted values of ***len*** range from 1 to 1024. For values outside that range, an error occurs.

[**RANDOM\_BYTES()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_random-bytes) can be used to provide the initialization vector for the [**AES\_DECRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-decrypt) and [**AES\_ENCRYPT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_aes-encrypt) functions. For use in that context, ***len*** must be at least 16. Larger values are permitted, but bytes in excess of 16 are ignored.

[**RANDOM\_BYTES()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_random-bytes) generates a random value, which makes its result nondeterministic. Consequently, statements that use this function are unsafe for statement-based replication.

**[SHA1(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sha1)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sha1)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sha1)**, [**SHA(*str*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha1)

Calculates an SHA-1 160-bit checksum for the string, as described in RFC 3174 (Secure Hash Algorithm). The value is returned as a string of 40 hexadecimal digits, or **NULL** if the argument was **NULL**. One of the possible uses for this function is as a hash key. See the notes at the beginning of this section about storing hash values efficiently. [**SHA()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha1) is synonymous with [**SHA1()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha1).

The return value is a string in the connection character set.

mysql> **SELECT SHA1('abc');**

-> 'a9993e364706816aba3e25717850c26c9cd0d89d'

[**SHA1()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha1) can be considered a cryptographically more secure equivalent of [**MD5()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_md5). However, see the note regarding the MD5 and SHA-1 algorithms at the beginning this section.

**[SHA2(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sha2)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sha2)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sha2)*[hash\_length](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sha2)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sha2)**

Calculates the SHA-2 family of hash functions (SHA-224, SHA-256, SHA-384, and SHA-512). The first argument is the plaintext string to be hashed. The second argument indicates the desired bit length of the result, which must have a value of 224, 256, 384, 512, or 0 (which is equivalent to 256). If either argument is **NULL** or the hash length is not one of the permitted values, the return value is **NULL**. Otherwise, the function result is a hash value containing the desired number of bits. See the notes at the beginning of this section about storing hash values efficiently.

The return value is a string in the connection character set.

mysql> **SELECT SHA2('abc', 224);**

-> '23097d223405d8228642a477bda255b32aadbce4bda0b3f7e36c9da7'

This function works only if MySQL has been configured with SSL support. See [Section 6.3, “Using Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connections).

[**SHA2()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha2) can be considered cryptographically more secure than [**MD5()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_md5) or [**SHA1()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sha1).

**[STATEMENT\_DIGEST(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_statement-digest)*[statement](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_statement-digest)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_statement-digest)**

Given an SQL statement as a string, returns the statement digest hash value as a string in the connection character set, or **NULL** if the argument is **NULL**. The related [**STATEMENT\_DIGEST\_TEXT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_statement-digest-text) function returns the normalized statement digest. For information about statement digesting, see [Section 27.10, “Performance Schema Statement Digests and Sampling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-statement-digests).

Both functions use the MySQL parser to parse the statement. If parsing fails, an error occurs. The error message includes the parse error only if the statement is provided as a literal string.

The [**max\_digest\_length**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_digest_length) system variable determines the maximum number of bytes available to these functions for computing normalized statement digests.

mysql> **SET @stmt = 'SELECT \* FROM mytable WHERE cola = 10 AND colb = 20';**

mysql> **SELECT STATEMENT\_DIGEST(@stmt);**

+------------------------------------------------------------------+

| STATEMENT\_DIGEST(@stmt) |

+------------------------------------------------------------------+

| 3bb95eeade896657c4526e74ff2a2862039d0a0fe8a9e7155b5fe492cbd78387 |

+------------------------------------------------------------------+

mysql> **SELECT STATEMENT\_DIGEST\_TEXT(@stmt);**

+----------------------------------------------------------+

| STATEMENT\_DIGEST\_TEXT(@stmt) |

+----------------------------------------------------------+

| SELECT \* FROM `mytable` WHERE `cola` = ? AND `colb` = ? |

+----------------------------------------------------------+

**[STATEMENT\_DIGEST\_TEXT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_statement-digest-text)*[statement](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_statement-digest-text)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_statement-digest-text)**

Given an SQL statement as a string, returns the normalized statement digest as a string in the connection character set, or **NULL** if the argument is **NULL**. For additional discussion and examples, see the description of the related [**STATEMENT\_DIGEST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_statement-digest) function.

**[UNCOMPRESS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_uncompress)*[string\_to\_uncompress](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_uncompress)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_uncompress)**

Uncompresses a string compressed by the [**COMPRESS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_compress) function. If the argument is not a compressed value, the result is **NULL**. This function requires MySQL to have been compiled with a compression library such as **zlib**. Otherwise, the return value is always **NULL**.

mysql> **SELECT UNCOMPRESS(COMPRESS('any string'));**

-> 'any string'

mysql> **SELECT UNCOMPRESS('any string');**

-> NULL

**[UNCOMPRESSED\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_uncompressed-length)*[compressed\_string](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_uncompressed-length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_uncompressed-length)**

Returns the length that the compressed string had before being compressed.

mysql> **SELECT UNCOMPRESSED\_LENGTH(COMPRESS(REPEAT('a',30)));**

-> 30

**[VALIDATE\_PASSWORD\_STRENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_validate-password-strength)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_validate-password-strength)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_validate-password-strength)**

Given an argument representing a plaintext password, this function returns an integer to indicate how strong the password is. The return value ranges from 0 (weak) to 100 (strong).

Password assessment by [**VALIDATE\_PASSWORD\_STRENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_validate-password-strength) is done by the **validate\_password** component. If that component is not installed, the function always returns 0. For information about installing **validate\_password**, see [Section 6.4.3, “The Password Validation Component”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#validate-password). To examine or configure the parameters that affect password testing, check or set the system variables implemented by **validate\_password**. See [Section 6.4.3.2, “Password Validation Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#validate-password-options-variables).

The password is subjected to increasingly strict tests and the return value reflects which tests were satisfied, as shown in the following table. In addition, if the [**validate\_password.check\_user\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#sysvar_validate_password.check_user_name) system variable is enabled and the password matches the user name, [**VALIDATE\_PASSWORD\_STRENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_validate-password-strength) returns 0 regardless of how other **validate\_password** system variables are set.

| **Password Test** | **Return Value** |
| --- | --- |
| Length < 4 | 0 |
| Length ≥ 4 and < [**validate\_password.length**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#sysvar_validate_password.length) | 25 |
| Satisfies policy 1 (**LOW**) | 50 |
| Satisfies policy 2 (**MEDIUM**) | 75 |
| Satisfies policy 3 (**STRONG**) | 100 |

## 12.15 Locking Functions

This section describes functions used to manipulate user-level locks.

**Table 12.19 Locking Functions**

| **Name** | **Description** |
| --- | --- |
| [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) | Get a named lock |
| [**IS\_FREE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-free-lock) | Whether the named lock is free |
| [**IS\_USED\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-used-lock) | Whether the named lock is in use; return connection identifier if true |
| [**RELEASE\_ALL\_LOCKS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-all-locks) | Release all current named locks |
| [**RELEASE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-lock) | Release the named lock |

**[GET\_LOCK(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-lock)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-lock)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-lock)*[timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-lock)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-lock)**

Tries to obtain a lock with a name given by the string ***str***, using a timeout of ***timeout*** seconds. A negative ***timeout*** value means infinite timeout. The lock is exclusive. While held by one session, other sessions cannot obtain a lock of the same name.

Returns **1** if the lock was obtained successfully, **0** if the attempt timed out (for example, because another client has previously locked the name), or **NULL** if an error occurred (such as running out of memory or the thread was killed with [**mysqladmin kill**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin)).

A lock obtained with [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) is released explicitly by executing [**RELEASE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-lock) or implicitly when your session terminates (either normally or abnormally). Locks obtained with [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) are not released when transactions commit or roll back.

[**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) is implemented using the metadata locking (MDL) subsystem. Multiple simultaneous locks can be acquired and [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) does not release any existing locks. For example, suppose that you execute these statements:

SELECT GET\_LOCK('lock1',10);

SELECT GET\_LOCK('lock2',10);

SELECT RELEASE\_LOCK('lock2');

SELECT RELEASE\_LOCK('lock1');

The second [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) acquires a second lock and both [**RELEASE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-lock) calls return 1 (success).

It is even possible for a given session to acquire multiple locks for the same name. Other sessions cannot acquire a lock with that name until the acquiring session releases all its locks for the name.

Uniquely named locks acquired with [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) appear in the Performance Schema [**metadata\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-metadata-locks-table) table. The **OBJECT\_TYPE** column says **USER LEVEL LOCK** and the **OBJECT\_NAME** column indicates the lock name. In the case that multiple locks are acquired for the same name, only the first lock for the name registers a row in the [**metadata\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-metadata-locks-table) table. Subsequent locks for the name increment a counter in the lock but do not acquire additional metadata locks. The [**metadata\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-metadata-locks-table) row for the lock is deleted when the last lock instance on the name is released.

The capability of acquiring multiple locks means there is the possibility of deadlock among clients. When this happens, the server chooses a caller and terminates its lock-acquisition request with an [**ER\_USER\_LOCK\_DEADLOCK**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_user_lock_deadlock) error. This error does not cause transactions to roll back.

MySQL enforces a maximum length on lock names of 64 characters.

[**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) can be used to implement application locks or to simulate record locks. Names are locked on a server-wide basis. If a name has been locked within one session, [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) blocks any request by another session for a lock with the same name. This enables clients that agree on a given lock name to use the name to perform cooperative advisory locking. But be aware that it also enables a client that is not among the set of cooperating clients to lock a name, either inadvertently or deliberately, and thus prevent any of the cooperating clients from locking that name. One way to reduce the likelihood of this is to use lock names that are database-specific or application-specific. For example, use lock names of the form ***db\_name.str*** or ***app\_name.str***.

If multiple clients are waiting for a lock, the order in which they acquire it is undefined. Applications should not assume that clients acquire the lock in the same order that they issued the lock requests.

[**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) is unsafe for statement-based replication. A warning is logged if you use this function when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**.

**Caution**

With the capability of acquiring multiple named locks, it is possible for a single statement to acquire a large number of locks. For example:

INSERT INTO ... SELECT GET\_LOCK(t1.col\_name) FROM t1;

These types of statements may have certain adverse effects. For example, if the statement fails part way through and rolls back, locks acquired up to the point of failure still exist. If the intent is for there to be a correspondence between rows inserted and locks acquired, that intent is not satisfied. Also, if it is important that locks are granted in a certain order, be aware that result set order may differ depending on which execution plan the optimizer chooses. For these reasons, it may be best to limit applications to a single lock-acquisition call per statement.

A different locking interface is available as either a plugin service or a set of user-defined functions. This interface provides lock namespaces and distinct read and write locks, unlike the interface provided by [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) and related functions. For details, see [Section 5.6.9.1, “The Locking Service”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#locking-service).

**[IS\_FREE\_LOCK(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-free-lock)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-free-lock)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-free-lock)**

Checks whether the lock named ***str*** is free to use (that is, not locked). Returns **1** if the lock is free (no one is using the lock), **0** if the lock is in use, and **NULL** if an error occurs (such as an incorrect argument).

This function is unsafe for statement-based replication. A warning is logged if you use this function when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**.

**[IS\_USED\_LOCK(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-used-lock)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-used-lock)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-used-lock)**

Checks whether the lock named ***str*** is in use (that is, locked). If so, it returns the connection identifier of the client session that holds the lock. Otherwise, it returns **NULL**.

This function is unsafe for statement-based replication. A warning is logged if you use this function when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**.

**[RELEASE\_ALL\_LOCKS()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_release-all-locks)**

Releases all named locks held by the current session and returns the number of locks released (0 if there were none)

This function is unsafe for statement-based replication. A warning is logged if you use this function when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**.

**[RELEASE\_LOCK(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_release-lock)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_release-lock)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_release-lock)**

Releases the lock named by the string ***str*** that was obtained with [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock). Returns **1** if the lock was released, **0** if the lock was not established by this thread (in which case the lock is not released), and **NULL** if the named lock did not exist. The lock does not exist if it was never obtained by a call to [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) or if it has previously been released.

The [**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) statement is convenient to use with [**RELEASE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-lock). See [Section 13.2.3, “DO Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do).

This function is unsafe for statement-based replication. A warning is logged if you use this function when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**.

## 12.16 Information Functions

**Table 12.20 Information Functions**

| **Name** | **Description** |
| --- | --- |
| [**BENCHMARK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_benchmark) | Repeatedly execute an expression |
| [**CHARSET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_charset) | Return the character set of the argument |
| [**COERCIBILITY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_coercibility) | Return the collation coercibility value of the string argument |
| [**COLLATION()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_collation) | Return the collation of the string argument |
| [**CONNECTION\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_connection-id) | Return the connection ID (thread ID) for the connection |
| [**CURRENT\_ROLE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-role) | Return the current active roles |
| [**CURRENT\_USER()**, **CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) | The authenticated user name and host name |
| [**DATABASE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_database) | Return the default (current) database name |
| [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) | For a SELECT with a LIMIT clause, the number of rows that would be returned were there no LIMIT clause |
| [**ICU\_VERSION()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_icu-version) | ICU library version |
| [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) | Value of the AUTOINCREMENT column for the last INSERT |
| [**ROLES\_GRAPHML()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_roles-graphml) | Return a GraphML document representing memory role subgraphs |
| [**ROW\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count) | The number of rows updated |
| [**SCHEMA()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_schema) | Synonym for DATABASE() |
| [**SESSION\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_session-user) | Synonym for USER() |
| [**SYSTEM\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_system-user) | Synonym for USER() |
| [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user) | The user name and host name provided by the client |
| [**VERSION()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_version) | Return a string that indicates the MySQL server version |

**[BENCHMARK(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_benchmark)*[count](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_benchmark)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_benchmark)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_benchmark)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_benchmark)**

The [**BENCHMARK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_benchmark) function executes the expression ***expr*** repeatedly ***count*** times. It may be used to time how quickly MySQL processes the expression. The result value is **0**, or **NULL** for inappropriate arguments such as a **NULL** or negative repeat count.

The intended use is from within the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client, which reports query execution times:

mysql> **SELECT BENCHMARK(1000000,AES\_ENCRYPT('hello','goodbye'));**

+---------------------------------------------------+

| BENCHMARK(1000000,AES\_ENCRYPT('hello','goodbye')) |

+---------------------------------------------------+

| 0 |

+---------------------------------------------------+

1 row in set (4.74 sec)

The time reported is elapsed time on the client end, not CPU time on the server end. It is advisable to execute [**BENCHMARK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_benchmark) several times, and to interpret the result with regard to how heavily loaded the server machine is.

[**BENCHMARK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_benchmark) is intended for measuring the runtime performance of scalar expressions, which has some significant implications for the way that you use it and interpret the results:

Only scalar expressions can be used. Although the expression can be a subquery, it must return a single column and at most a single row. For example, [**BENCHMARK(10, (SELECT \* FROM t))**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_benchmark) fails if the table **t** has more than one column or more than one row.

Executing a **SELECT *expr*** statement ***N*** times differs from executing **SELECT BENCHMARK(*N*, *expr*)** in terms of the amount of overhead involved. The two have very different execution profiles and you should not expect them to take the same amount of time. The former involves the parser, optimizer, table locking, and runtime evaluation ***N*** times each. The latter involves only runtime evaluation ***N*** times, and all the other components just once. Memory structures already allocated are reused, and runtime optimizations such as local caching of results already evaluated for aggregate functions can alter the results. Use of [**BENCHMARK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_benchmark) thus measures performance of the runtime component by giving more weight to that component and removing the “noise” introduced by the network, parser, optimizer, and so forth.

**[CHARSET(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_charset)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_charset)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_charset)**

Returns the character set of the string argument.

mysql> **SELECT CHARSET('abc');**

-> 'utf8'

mysql> **SELECT CHARSET(CONVERT('abc' USING latin1));**

-> 'latin1'

mysql> **SELECT CHARSET(USER());**

-> 'utf8'

**[COERCIBILITY(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_coercibility)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_coercibility)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_coercibility)**

Returns the collation coercibility value of the string argument.

mysql> **SELECT COERCIBILITY('abc' COLLATE utf8\_swedish\_ci);**

-> 0

mysql> **SELECT COERCIBILITY(USER());**

-> 3

mysql> **SELECT COERCIBILITY('abc');**

-> 4

mysql> **SELECT COERCIBILITY(1000);**

-> 5

The return values have the meanings shown in the following table. Lower values have higher precedence.

| **Coercibility** | **Meaning** | **Example** |
| --- | --- | --- |
| **0** | Explicit collation | Value with **COLLATE** clause |
| **1** | No collation | Concatenation of strings with different collations |
| **2** | Implicit collation | Column value, stored routine parameter or local variable |
| **3** | System constant | [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user) return value |
| **4** | Coercible | Literal string |
| **5** | Numeric | Numeric or temporal value |
| **5** | Ignorable | **NULL** or an expression derived from **NULL** |

For more information, see [Section 10.8.4, “Collation Coercibility in Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-coercibility).

**[COLLATION(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_collation)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_collation)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_collation)**

Returns the collation of the string argument.

mysql> **SELECT COLLATION('abc');**

-> 'utf8\_general\_ci'

mysql> **SELECT COLLATION(\_utf8mb4'abc');**

-> 'utf8mb4\_0900\_ai\_ci'

mysql> **SELECT COLLATION(\_latin1'abc');**

-> 'latin1\_swedish\_ci'

**[CONNECTION\_ID()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_connection-id)**

Returns the connection ID (thread ID) for the connection. Every connection has an ID that is unique among the set of currently connected clients.

The value returned by [**CONNECTION\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_connection-id) is the same type of value as displayed in the **ID** column of the [**INFORMATION\_SCHEMA.PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-processlist-table) table, the **Id** column of [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) output, and the **PROCESSLIST\_ID** column of the Performance Schema [**threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-threads-table) table.

mysql> **SELECT CONNECTION\_ID();**

-> 23786

**Warning**

Changing the session value of the [**pseudo\_thread\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_pseudo_thread_id) system variable changes the value returned by the [**CONNECTION\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_connection-id) function.

**[CURRENT\_ROLE()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_current-role)**

Returns a **utf8** string containing the current active roles for the current session, separated by commas, or **NONE** if there are none. The value reflects the setting of the [**sql\_quote\_show\_create**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_quote_show_create) system variable.

Suppose that an account is granted roles as follows:

GRANT 'r1', 'r2' TO 'u1'@'localhost';

SET DEFAULT ROLE ALL TO 'u1'@'localhost';

In sessions for **u1**, the initial [**CURRENT\_ROLE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-role) value names the default account roles. Using [**SET ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) changes that:

mysql> **SELECT CURRENT\_ROLE();**

+-------------------+

| CURRENT\_ROLE() |

+-------------------+

| `r1`@`%`,`r2`@`%` |

+-------------------+

mysql> **SET ROLE 'r1'; SELECT CURRENT\_ROLE();**

+----------------+

| CURRENT\_ROLE() |

+----------------+

| `r1`@`%` |

+----------------+

**[CURRENT\_USER](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_current-user)**, [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user)

Returns the user name and host name combination for the MySQL account that the server used to authenticate the current client. This account determines your access privileges. The return value is a string in the **utf8** character set.

The value of [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) can differ from the value of [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user).

mysql> **SELECT USER();**

-> 'davida@localhost'

mysql> **SELECT \* FROM mysql.user;**

ERROR 1044: Access denied for user ''@'localhost' to

database 'mysql'

mysql> **SELECT CURRENT\_USER();**

-> '@localhost'

The example illustrates that although the client specified a user name of **davida** (as indicated by the value of the [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user) function), the server authenticated the client using an anonymous user account (as seen by the empty user name part of the [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) value). One way this might occur is that there is no account listed in the grant tables for **davida**.

Within a stored program or view, [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) returns the account for the user who defined the object (as given by its **DEFINER** value) unless defined with the **SQL SECURITY INVOKER** characteristic. In the latter case, [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) returns the object's invoker.

Triggers and events have no option to define the **SQL SECURITY** characteristic, so for these objects, [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) returns the account for the user who defined the object. To return the invoker, use [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user) or [**SESSION\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_session-user).

The following statements support use of the [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) function to take the place of the name of (and, possibly, a host for) an affected user or a definer; in such cases, [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) is expanded where and as needed:

[**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user)

[**RENAME USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-user)

[**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant)

[**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke)

[**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function)

[**CREATE PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure)

[**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger)

[**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event)

[**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view)

[**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event)

[**ALTER VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-view)

[**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password)

For information about the implications that this expansion of [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) has for replication, see [Section 17.5.1.8, “Replication of CURRENT\_USER()”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-current-user).

**[DATABASE()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_database)**

Returns the default (current) database name as a string in the **utf8** character set. If there is no default database, [**DATABASE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_database) returns **NULL**. Within a stored routine, the default database is the database that the routine is associated with, which is not necessarily the same as the database that is the default in the calling context.

mysql> **SELECT DATABASE();**

-> 'test'

If there is no default database, [**DATABASE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_database) returns **NULL**.

**[FOUND\_ROWS()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_found-rows)**

**Note**

The **SQL\_CALC\_FOUND\_ROWS** query modifier and accompanying [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) function are deprecated as of MySQL 8.0.17; expect them to be removed in a future version of MySQL. As a replacement, considering executing your query with **LIMIT**, and then a second query with [**COUNT(\*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count) and without **LIMIT** to determine whether there are additional rows. For example, instead of these queries:

SELECT SQL\_CALC\_FOUND\_ROWS \* FROM ***tbl\_name*** WHERE id > 100 LIMIT 10;

SELECT FOUND\_ROWS();

Use these queries instead:

SELECT \* FROM ***tbl\_name*** WHERE id > 100 LIMIT 10;

SELECT COUNT(\*) FROM ***tbl\_name*** WHERE id > 100;

[**COUNT(\*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count) is subject to certain optimizations. **SQL\_CALC\_FOUND\_ROWS** causes some optimizations to be disabled.

A [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement may include a **LIMIT** clause to restrict the number of rows the server returns to the client. In some cases, it is desirable to know how many rows the statement would have returned without the **LIMIT**, but without running the statement again. To obtain this row count, include an **SQL\_CALC\_FOUND\_ROWS** option in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement, and then invoke [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) afterward:

mysql> **SELECT SQL\_CALC\_FOUND\_ROWS \* FROM *tbl\_name***

-> **WHERE id > 100 LIMIT 10;**

mysql> **SELECT FOUND\_ROWS();**

The second [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) returns a number indicating how many rows the first [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) would have returned had it been written without the **LIMIT** clause.

In the absence of the **SQL\_CALC\_FOUND\_ROWS** option in the most recent successful [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement, [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) returns the number of rows in the result set returned by that statement. If the statement includes a **LIMIT** clause, [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) returns the number of rows up to the limit. For example, [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) returns 10 or 60, respectively, if the statement includes **LIMIT 10** or **LIMIT 50, 10**.

The row count available through [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) is transient and not intended to be available past the statement following the **SELECT SQL\_CALC\_FOUND\_ROWS** statement. If you need to refer to the value later, save it:

mysql> **SELECT SQL\_CALC\_FOUND\_ROWS \* FROM ... ;**

mysql> **SET @rows = FOUND\_ROWS();**

If you are using **SELECT SQL\_CALC\_FOUND\_ROWS**, MySQL must calculate how many rows are in the full result set. However, this is faster than running the query again without **LIMIT**, because the result set need not be sent to the client.

**SQL\_CALC\_FOUND\_ROWS** and [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) can be useful in situations when you want to restrict the number of rows that a query returns, but also determine the number of rows in the full result set without running the query again. An example is a Web script that presents a paged display containing links to the pages that show other sections of a search result. Using [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) enables you to determine how many other pages are needed for the rest of the result.

The use of **SQL\_CALC\_FOUND\_ROWS** and [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) is more complex for [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) statements than for simple [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, because **LIMIT** may occur at multiple places in a [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union). It may be applied to individual [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements in the [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union), or global to the [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) result as a whole.

The intent of **SQL\_CALC\_FOUND\_ROWS** for [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) is that it should return the row count that would be returned without a global **LIMIT**. The conditions for use of **SQL\_CALC\_FOUND\_ROWS** with [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) are:

The **SQL\_CALC\_FOUND\_ROWS** keyword must appear in the first [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) of the [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union).

The value of [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) is exact only if [**UNION ALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) is used. If [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) without **ALL** is used, duplicate removal occurs and the value of [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) is only approximate.

If no **LIMIT** is present in the [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union), **SQL\_CALC\_FOUND\_ROWS** is ignored and returns the number of rows in the temporary table that is created to process the [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union).

Beyond the cases described here, the behavior of [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) is undefined (for example, its value following a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement that fails with an error).

**Important**

[**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) is not replicated reliably using statement-based replication. This function is automatically replicated using row-based replication.

**[ICU\_VERSION()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_icu-version)**

The version of the International Components for Unicode (ICU) library used to support regular expression operations (see [Section 12.8.2, “Regular Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#regexp)). This function is primarily intended for use in test cases.

**[LAST\_INSERT\_ID()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_last-insert-id)**, [**LAST\_INSERT\_ID(*expr*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id)

With no argument, [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) returns a **BIGINT UNSIGNED** (64-bit) value representing the first automatically generated value successfully inserted for an **AUTO\_INCREMENT** column as a result of the most recently executed [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement. The value of [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) remains unchanged if no rows are successfully inserted.

With an argument, [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) returns an unsigned integer.

For example, after inserting a row that generates an **AUTO\_INCREMENT** value, you can get the value like this:

mysql> **SELECT LAST\_INSERT\_ID();**

-> 195

The currently executing statement does not affect the value of [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id). Suppose that you generate an **AUTO\_INCREMENT** value with one statement, and then refer to [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) in a multiple-row [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement that inserts rows into a table with its own **AUTO\_INCREMENT** column. The value of [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) remains stable in the second statement; its value for the second and later rows is not affected by the earlier row insertions. (You should be aware that, if you mix references to [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) and [**LAST\_INSERT\_ID(*expr*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id), the effect is undefined.)

If the previous statement returned an error, the value of [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) is undefined. For transactional tables, if the statement is rolled back due to an error, the value of [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) is left undefined. For manual [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), the value of [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) is not restored to that before the transaction; it remains as it was at the point of the [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit).

Within the body of a stored routine (procedure or function) or a trigger, the value of [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) changes the same way as for statements executed outside the body of these kinds of objects. The effect of a stored routine or trigger upon the value of [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) that is seen by following statements depends on the kind of routine:

If a stored procedure executes statements that change the value of [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id), the changed value is seen by statements that follow the procedure call.

For stored functions and triggers that change the value, the value is restored when the function or trigger ends, so statements coming after it do not see a changed value.

The ID that was generated is maintained in the server on a per-connection basis. This means that the value returned by the function to a given client is the first **AUTO\_INCREMENT** value generated for most recent statement affecting an **AUTO\_INCREMENT** column by that client. This value cannot be affected by other clients, even if they generate **AUTO\_INCREMENT** values of their own. This behavior ensures that each client can retrieve its own ID without concern for the activity of other clients, and without the need for locks or transactions.

The value of [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) is not changed if you set the **AUTO\_INCREMENT** column of a row to a non-“magic” value (that is, a value that is not **NULL** and not **0**).

**Important**

If you insert multiple rows using a single [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement, [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) returns the value generated for the first inserted row only. The reason for this is to make it possible to reproduce easily the same [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement against some other server.

For example:

mysql> **USE test;**

mysql> **CREATE TABLE t (**

**id INT AUTO\_INCREMENT NOT NULL PRIMARY KEY,**

**name VARCHAR(10) NOT NULL**

**);**

mysql> **INSERT INTO t VALUES (NULL, 'Bob');**

mysql> **SELECT \* FROM t;**

+----+------+

| id | name |

+----+------+

| 1 | Bob |

+----+------+

mysql> **SELECT LAST\_INSERT\_ID();**

+------------------+

| LAST\_INSERT\_ID() |

+------------------+

| 1 |

+------------------+

mysql> **INSERT INTO t VALUES**

**(NULL, 'Mary'), (NULL, 'Jane'), (NULL, 'Lisa');**

mysql> **SELECT \* FROM t;**

+----+------+

| id | name |

+----+------+

| 1 | Bob |

| 2 | Mary |

| 3 | Jane |

| 4 | Lisa |

+----+------+

mysql> **SELECT LAST\_INSERT\_ID();**

+------------------+

| LAST\_INSERT\_ID() |

+------------------+

| 2 |

+------------------+

Although the second [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement inserted three new rows into **t**, the ID generated for the first of these rows was **2**, and it is this value that is returned by [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) for the following [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement.

If you use [**INSERT IGNORE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) and the row is ignored, the [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) remains unchanged from the current value (or 0 is returned if the connection has not yet performed a successful **INSERT**) and, for non-transactional tables, the **AUTO\_INCREMENT** counter is not incremented. For **InnoDB** tables, the **AUTO\_INCREMENT** counter is incremented if [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) is set to **1** or **2**, as demonstrated in the following example:

mysql> **USE test;**

mysql> **SELECT @@innodb\_autoinc\_lock\_mode;**

+----------------------------+

| @@innodb\_autoinc\_lock\_mode |

+----------------------------+

| 1 |

+----------------------------+

mysql> **CREATE TABLE `t` (**

**`id` INT(11) NOT NULL AUTO\_INCREMENT,**

**`val` INT(11) DEFAULT NULL,**

**PRIMARY KEY (`id`),**

**UNIQUE KEY `i1` (`val`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

# Insert two rows

mysql> **INSERT INTO t (val) VALUES (1),(2);**

# With auto\_increment\_offset=1, the inserted rows

# result in an AUTO\_INCREMENT value of 3

mysql> **SHOW CREATE TABLE t\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t

Create Table: CREATE TABLE `t` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`val` int(11) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `i1` (`val`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=latin1

# LAST\_INSERT\_ID() returns the first automatically generated

# value that is successfully inserted for the AUTO\_INCREMENT column

mysql> **SELECT LAST\_INSERT\_ID();**

+------------------+

| LAST\_INSERT\_ID() |

+------------------+

| 1 |

+------------------+

# The attempted insertion of duplicate rows fail but errors are ignored

mysql> **INSERT IGNORE INTO t (val) VALUES (1),(2);**

Query OK, 0 rows affected (0.00 sec)

Records: 2 Duplicates: 2 Warnings: 0

# With innodb\_autoinc\_lock\_mode=1, the AUTO\_INCREMENT counter

# is incremented for the ignored rows

mysql> **SHOW CREATE TABLE t\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t

Create Table: CREATE TABLE `t` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`val` int(11) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `i1` (`val`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=latin1

# The LAST\_INSERT\_ID is unchanged because the previous insert was unsuccessful

mysql> **SELECT LAST\_INSERT\_ID();**

+------------------+

| LAST\_INSERT\_ID() |

+------------------+

| 1 |

+------------------+

For more information, see [Section 15.6.1.6, “AUTO\_INCREMENT Handling in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-handling).

If ***expr*** is given as an argument to [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id), the value of the argument is returned by the function and is remembered as the next value to be returned by [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id). This can be used to simulate sequences:

Create a table to hold the sequence counter and initialize it:

mysql> **CREATE TABLE sequence (id INT NOT NULL);**

mysql> **INSERT INTO sequence VALUES (0);**

Use the table to generate sequence numbers like this:

mysql> **UPDATE sequence SET id=LAST\_INSERT\_ID(id+1);**

mysql> **SELECT LAST\_INSERT\_ID();**

The [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement increments the sequence counter and causes the next call to [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) to return the updated value. The [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement retrieves that value. The [**mysql\_insert\_id()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-insert-id.html) C API function can also be used to get the value. See [mysql\_insert\_id()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-insert-id.html).

You can generate sequences without calling [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id), but the utility of using the function this way is that the ID value is maintained in the server as the last automatically generated value. It is multi-user safe because multiple clients can issue the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement and get their own sequence value with the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement (or [**mysql\_insert\_id()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-insert-id.html)), without affecting or being affected by other clients that generate their own sequence values.

Note that [**mysql\_insert\_id()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-insert-id.html) is only updated after [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements, so you cannot use the C API function to retrieve the value for [**LAST\_INSERT\_ID(*expr*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) after executing other SQL statements like [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) or [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable).

**[ROLES\_GRAPHML()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_roles-graphml)**

Returns a **utf8** string containing a GraphML document representing memory role subgraphs. The [**ROLE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_role-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege) is required to see content in the **<graphml>** element. Otherwise, the result shows only an empty element:

mysql> **SELECT ROLES\_GRAPHML();**

+---------------------------------------------------+

| ROLES\_GRAPHML() |

+---------------------------------------------------+

| <?xml version="1.0" encoding="UTF-8"?><graphml /> |

+---------------------------------------------------+

**[ROW\_COUNT()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_row-count)**

**ROW\_COUNT()** returns a value as follows:

DDL statements: 0. This applies to statements such as [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table).

DML statements other than [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select): The number of affected rows. This applies to statements such as [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) (as before), but now also to statements such as [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data).

[**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select): -1 if the statement returns a result set, or the number of rows “affected” if it does not. For example, for **SELECT \* FROM t1**, [**ROW\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count) returns -1. For **SELECT \* FROM t1 INTO OUTFILE '*file\_name*'**, [**ROW\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count) returns the number of rows written to the file.

[**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statements: 0.

For [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements, the affected-rows value by default is the number of rows actually changed. If you specify the **CLIENT\_FOUND\_ROWS** flag to [**mysql\_real\_connect()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-connect.html) when connecting to [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld), the affected-rows value is the number of rows “found”; that is, matched by the **WHERE** clause.

For [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) statements, the affected-rows value is 2 if the new row replaced an old row, because in this case, one row was inserted after the duplicate was deleted.

For [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) statements, the affected-rows value per row is 1 if the row is inserted as a new row, 2 if an existing row is updated, and 0 if an existing row is set to its current values. If you specify the **CLIENT\_FOUND\_ROWS** flag, the affected-rows value is 1 (not 0) if an existing row is set to its current values.

The [**ROW\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count) value is similar to the value from the [**mysql\_affected\_rows()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-affected-rows.html) C API function and the row count that the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client displays following statement execution.

mysql> **INSERT INTO t VALUES(1),(2),(3);**

Query OK, 3 rows affected (0.00 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> **SELECT ROW\_COUNT();**

+-------------+

| ROW\_COUNT() |

+-------------+

| 3 |

+-------------+

1 row in set (0.00 sec)

mysql> **DELETE FROM t WHERE i IN(1,2);**

Query OK, 2 rows affected (0.00 sec)

mysql> **SELECT ROW\_COUNT();**

+-------------+

| ROW\_COUNT() |

+-------------+

| 2 |

+-------------+

1 row in set (0.00 sec)

**Important**

[**ROW\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count) is not replicated reliably using statement-based replication. This function is automatically replicated using row-based replication.

**[SCHEMA()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_schema)**

This function is a synonym for [**DATABASE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_database).

**[SESSION\_USER()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_session-user)**

**[SESSION\_USER()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_session-user)** is a synonym for [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user).

**[SYSTEM\_USER()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_system-user)**

**[SYSTEM\_USER()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_system-user)** is a synonym for [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user).

**Note**

The [**SYSTEM\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_system-user) function is distinct from the [**SYSTEM\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_system-user) privilege. The former returns the current MySQL account name. The latter distinguishes the system user and regular user account categories (see [Section 6.2.11, “Account Categories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-categories)).

**[USER()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_user)**

Returns the current MySQL user name and host name as a string in the **utf8** character set.

mysql> **SELECT USER();**

-> 'davida@localhost'

The value indicates the user name you specified when connecting to the server, and the client host from which you connected. The value can be different from that of [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user).

**[VERSION()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_version)**

Returns a string that indicates the MySQL server version. The string uses the **utf8** character set. The value might have a suffix in addition to the version number. See the description of the [**version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_version) system variable in [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

This function is unsafe for statement-based replication. A warning is logged if you use this function when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**.

mysql> **SELECT VERSION();**

-> '8.0.26-standard'

## 12.17 Spatial Analysis Functions

[12.17.1 Spatial Function Reference](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-function-reference)

[12.17.2 Argument Handling by Spatial Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-function-argument-handling)

[12.17.3 Functions That Create Geometry Values from WKT Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-wkt-functions)

[12.17.4 Functions That Create Geometry Values from WKB Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-wkb-functions)

[12.17.5 MySQL-Specific Functions That Create Geometry Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-mysql-specific-functions)

[12.17.6 Geometry Format Conversion Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-format-conversion-functions)

[12.17.7 Geometry Property Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-property-functions)

[12.17.8 Spatial Operator Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-operator-functions)

[12.17.9 Functions That Test Spatial Relations Between Geometry Objects](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-relation-functions)

[12.17.10 Spatial Geohash Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-geohash-functions)

[12.17.11 Spatial GeoJSON Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-geojson-functions)

[12.17.12 Spatial Aggregate Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-aggregate-functions)

[12.17.13 Spatial Convenience Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-convenience-functions)

MySQL provides functions to perform various operations on spatial data. These functions can be grouped into several major categories according to the type of operation they perform:

Functions that create geometries in various formats (WKT, WKB, internal)

Functions that convert geometries between formats

Functions that access qualitative or quantitative properties of a geometry

Functions that describe relations between two geometries

Functions that create new geometries from existing ones

For general background about MySQL support for using spatial data, see [Section 11.4, “Spatial Data Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#spatial-types).

### 12.17.1 Spatial Function Reference

The following table lists each spatial function and provides a short description of each one.

**Table 12.21 Spatial Functions**

| **Name** | **Description** | **Introduced** |
| --- | --- | --- |
| [**GeomCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geomcollection) | Construct geometry collection from geometries |  |
| [**GeometryCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geometrycollection) | Construct geometry collection from geometries |  |
| [**LineString()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_linestring) | Construct LineString from Point values |  |
| [**MBRContains()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcontains) | Whether MBR of one geometry contains MBR of another |  |
| [**MBRCoveredBy()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcoveredby) | Whether one MBR is covered by another |  |
| [**MBRCovers()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcovers) | Whether one MBR covers another |  |
| [**MBRDisjoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrdisjoint) | Whether MBRs of two geometries are disjoint |  |
| [**MBREquals()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrequals) | Whether MBRs of two geometries are equal |  |
| [**MBRIntersects()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrintersects) | Whether MBRs of two geometries intersect |  |
| [**MBROverlaps()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbroverlaps) | Whether MBRs of two geometries overlap |  |
| [**MBRTouches()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrtouches) | Whether MBRs of two geometries touch |  |
| [**MBRWithin()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrwithin) | Whether MBR of one geometry is within MBR of another |  |
| [**MultiLineString()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_multilinestring) | Contruct MultiLineString from LineString values |  |
| [**MultiPoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_multipoint) | Construct MultiPoint from Point values |  |
| [**MultiPolygon()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_multipolygon) | Construct MultiPolygon from Polygon values |  |
| [**Point()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_point) | Construct Point from coordinates |  |
| [**Polygon()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_polygon) | Construct Polygon from LineString arguments |  |
| [**ST\_Area()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-area) | Return Polygon or MultiPolygon area |  |
| [**ST\_AsBinary(), ST\_AsWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-asbinary) | Convert from internal geometry format to WKB |  |
| [**ST\_AsGeoJSON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-asgeojson) | Generate GeoJSON object from geometry |  |
| [**ST\_AsText(), ST\_AsWKT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-astext) | Convert from internal geometry format to WKT |  |
| [**ST\_Buffer()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer) | Return geometry of points within given distance from geometry |  |
| [**ST\_Buffer\_Strategy()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer-strategy) | Produce strategy option for ST\_Buffer() |  |
| [**ST\_Centroid()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-centroid) | Return centroid as a point |  |
| [**ST\_Collect()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-collect) | Aggregate spatial values into collection | 8.0.24 |
| [**ST\_Contains()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-contains) | Whether one geometry contains another |  |
| [**ST\_ConvexHull()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-convexhull) | Return convex hull of geometry |  |
| [**ST\_Crosses()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-crosses) | Whether one geometry crosses another |  |
| [**ST\_Difference()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-difference) | Return point set difference of two geometries |  |
| [**ST\_Dimension()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-dimension) | Dimension of geometry |  |
| [**ST\_Disjoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-disjoint) | Whether one geometry is disjoint from another |  |
| [**ST\_Distance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance) | The distance of one geometry from another |  |
| [**ST\_Distance\_Sphere()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance-sphere) | Minimum distance on earth between two geometries |  |
| [**ST\_EndPoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-endpoint) | End Point of LineString |  |
| [**ST\_Envelope()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-envelope) | Return MBR of geometry |  |
| [**ST\_Equals()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-equals) | Whether one geometry is equal to another |  |
| [**ST\_ExteriorRing()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-exteriorring) | Return exterior ring of Polygon |  |
| [**ST\_FrechetDistance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-frechetdistance) | The discrete Fréchet distance of one geometry from another | 8.0.23 |
| [**ST\_GeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geohash) | Produce a geohash value |  |
| [**ST\_GeomCollFromText(), ST\_GeometryCollectionFromText(), ST\_GeomCollFromTxt()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomcollfromtext) | Return geometry collection from WKT |  |
| [**ST\_GeomCollFromWKB(), ST\_GeometryCollectionFromWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomcollfromwkb) | Return geometry collection from WKB |  |
| [**ST\_GeometryN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geometryn) | Return N-th geometry from geometry collection |  |
| [**ST\_GeometryType()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geometrytype) | Return name of geometry type |  |
| [**ST\_GeomFromGeoJSON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromgeojson) | Generate geometry from GeoJSON object |  |
| [**ST\_GeomFromText(), ST\_GeometryFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromtext) | Return geometry from WKT |  |
| [**ST\_GeomFromWKB(), ST\_GeometryFromWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromwkb) | Return geometry from WKB |  |
| [**ST\_HausdorffDistance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-hausdorffdistance) | The discrete Hausdorff distance of one geometry from another | 8.0.23 |
| [**ST\_InteriorRingN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-interiorringn) | Return N-th interior ring of Polygon |  |
| [**ST\_Intersection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-intersection) | Return point set intersection of two geometries |  |
| [**ST\_Intersects()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-intersects) | Whether one geometry intersects another |  |
| [**ST\_IsClosed()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isclosed) | Whether a geometry is closed and simple |  |
| [**ST\_IsEmpty()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isempty) | Whether a geometry is empty |  |
| [**ST\_IsSimple()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-issimple) | Whether a geometry is simple |  |
| [**ST\_IsValid()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isvalid) | Whether a geometry is valid |  |
| [**ST\_LatFromGeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latfromgeohash) | Return latitude from geohash value |  |
| [**ST\_Latitude()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latitude) | Return latitude of Point | 8.0.12 |
| [**ST\_Length()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-length) | Return length of LineString |  |
| [**ST\_LineFromText(), ST\_LineStringFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-linefromtext) | Construct LineString from WKT |  |
| [**ST\_LineFromWKB(), ST\_LineStringFromWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-linefromwkb) | Construct LineString from WKB |  |
| [**ST\_LineInterpolatePoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-lineinterpolatepoint) | The point a given percentage along a LineString | 8.0.24 |
| [**ST\_LineInterpolatePoints()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-lineinterpolatepoints) | The points a given percentage along a LineString | 8.0.24 |
| [**ST\_LongFromGeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-longfromgeohash) | Return longitude from geohash value |  |
| [**ST\_Longitude()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-longitude) | Return longitude of Point | 8.0.12 |
| [**ST\_MakeEnvelope()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-makeenvelope) | Rectangle around two points |  |
| [**ST\_MLineFromText(), ST\_MultiLineStringFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mlinefromtext) | Construct MultiLineString from WKT |  |
| [**ST\_MLineFromWKB(), ST\_MultiLineStringFromWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mlinefromwkb) | Construct MultiLineString from WKB |  |
| [**ST\_MPointFromText(), ST\_MultiPointFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpointfromtext) | Construct MultiPoint from WKT |  |
| [**ST\_MPointFromWKB(), ST\_MultiPointFromWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpointfromwkb) | Construct MultiPoint from WKB |  |
| [**ST\_MPolyFromText(), ST\_MultiPolygonFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpolyfromtext) | Construct MultiPolygon from WKT |  |
| [**ST\_MPolyFromWKB(), ST\_MultiPolygonFromWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpolyfromwkb) | Construct MultiPolygon from WKB |  |
| [**ST\_NumGeometries()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-numgeometries) | Return number of geometries in geometry collection |  |
| [**ST\_NumInteriorRing(), ST\_NumInteriorRings()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-numinteriorrings) | Return number of interior rings in Polygon |  |
| [**ST\_NumPoints()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-numpoints) | Return number of points in LineString |  |
| [**ST\_Overlaps()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-overlaps) | Whether one geometry overlaps another |  |
| [**ST\_PointAtDistance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointatdistance) | The point a given distance along a LineString | 8.0.24 |
| [**ST\_PointFromGeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointfromgeohash) | Convert geohash value to POINT value |  |
| [**ST\_PointFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointfromtext) | Construct Point from WKT |  |
| [**ST\_PointFromWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointfromwkb) | Construct Point from WKB |  |
| [**ST\_PointN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointn) | Return N-th point from LineString |  |
| [**ST\_PolyFromText(), ST\_PolygonFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-polyfromtext) | Construct Polygon from WKT |  |
| [**ST\_PolyFromWKB(), ST\_PolygonFromWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-polyfromwkb) | Construct Polygon from WKB |  |
| [**ST\_Simplify()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-simplify) | Return simplified geometry |  |
| [**ST\_SRID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) | Return spatial reference system ID for geometry |  |
| [**ST\_StartPoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-startpoint) | Start Point of LineString |  |
| [**ST\_SwapXY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-swapxy) | Return argument with X/Y coordinates swapped |  |
| [**ST\_SymDifference()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-symdifference) | Return point set symmetric difference of two geometries |  |
| [**ST\_Touches()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-touches) | Whether one geometry touches another |  |
| [**ST\_Transform()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-transform) | Transform coordinates of geometry | 8.0.13 |
| [**ST\_Union()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-union) | Return point set union of two geometries |  |
| [**ST\_Validate()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-validate) | Return validated geometry |  |
| [**ST\_Within()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-within) | Whether one geometry is within another |  |
| [**ST\_X()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-x) | Return X coordinate of Point |  |
| [**ST\_Y()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-y) | Return Y coordinate of Point |  |

### 12.17.2 Argument Handling by Spatial Functions

Spatial values, or geometries, have the properties described in [Section 11.4.2.2, “Geometry Class”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#gis-class-geometry). The following discussion lists general spatial function argument-handling characteristics. Specific functions or groups of functions may have additional or different argument-handling characteristics, as discussed in the sections where those function descriptions occur. Where that is true, those descriptions take precedence over the general discussion here.

Spatial functions are defined only for valid geometry values. See [Section 11.4.4, “Geometry Well-Formedness and Validity”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#geometry-well-formedness-validity).

Each geometry value is associated with a spatial reference system (SRS), which is a coordinate-based system for geographic locations. See [Section 11.4.5, “Spatial Reference System Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#spatial-reference-systems).

The spatial reference identifier (SRID) of a geometry identifies the SRS in which the geometry is defined. In MySQL, the SRID value is an integer associated with the geometry value. The maximum usable SRID value is 232−1. If a larger value is given, only the lower 32 bits are used.

SRID 0 represents an infinite flat Cartesian plane with no units assigned to its axes. To ensure SRID 0 behavior, create geometry values using SRID 0. SRID 0 is the default for new geometry values if no SRID is specified.

For computations on multiple geometry values, all values must be in the same SRS or an error occurs. Thus, spatial functions that take multiple geometry arguments require those arguments to be in the same SRS. If a spatial function returns [**ER\_GIS\_DIFFERENT\_SRIDS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_different_srids), it means that the geometry arguments were not all in the same SRS. You must modify them to have the same SRS.

A geometry returned by a spatial function is in the SRS of the geometry arguments because geometry values produced by any spatial function inherit the SRID of the geometry arguments.

The [Open Geospatial Consortium](http://www.opengeospatial.org) guidelines require that input polygons already be closed, so unclosed polygons are rejected as invalid rather than being closed.

In MySQL, the only valid empty geometry is represented in the form of an empty geometry collection. Empty geometry collection handling is as follows: An empty WKT input geometry collection may be specified as **'GEOMETRYCOLLECTION()'**. This is also the output WKT resulting from a spatial operation that produces an empty geometry collection.

During parsing of a nested geometry collection, the collection is flattened and its basic components are used in various GIS operations to compute results. This provides additional flexibility to users because it is unnecessary to be concerned about the uniqueness of geometry data. Nested geometry collections may be produced from nested GIS function calls without having to be explicitly flattened first.

### 12.17.3 Functions That Create Geometry Values from WKT Values

These functions take as arguments a Well-Known Text (WKT) representation and, optionally, a spatial reference system identifier (SRID). They return the corresponding geometry. For a description of WKT format, see [Well-Known Text (WKT) Format](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#gis-wkt-format).

Functions in this section detect arguments in either Cartesian or geographic spatial reference systems (SRSs), and return results appropriate to the SRS.

[**ST\_GeomFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromtext) accepts a WKT value of any geometry type as its first argument. Other functions provide type-specific construction functions for construction of geometry values of each geometry type.

Functions such as [**ST\_MPointFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpointfromtext) and [**ST\_GeomFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromtext) that accept WKT-format representations of **MultiPoint** values permit individual points within values to be surrounded by parentheses. For example, both of the following function calls are valid:

ST\_MPointFromText('MULTIPOINT (1 1, 2 2, 3 3)')

ST\_MPointFromText('MULTIPOINT ((1 1), (2 2), (3 3))')

Functions such as [**ST\_GeomFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromtext) that accept WKT geometry collection arguments understand both OpenGIS **'GEOMETRYCOLLECTION EMPTY'** standard syntax and MySQL **'GEOMETRYCOLLECTION()'** nonstandard syntax. Functions such as [**ST\_AsWKT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-astext) that produce WKT values produce **'GEOMETRYCOLLECTION EMPTY'** standard syntax:

mysql> **SET @s1 = ST\_GeomFromText('GEOMETRYCOLLECTION()');**

mysql> **SET @s2 = ST\_GeomFromText('GEOMETRYCOLLECTION EMPTY');**

mysql> **SELECT ST\_AsWKT(@s1), ST\_AsWKT(@s2);**

+--------------------------+--------------------------+

| ST\_AsWKT(@s1) | ST\_AsWKT(@s2) |

+--------------------------+--------------------------+

| GEOMETRYCOLLECTION EMPTY | GEOMETRYCOLLECTION EMPTY |

+--------------------------+--------------------------+

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any geometry argument is **NULL** or is not a syntactically well-formed geometry, or if the SRID argument is **NULL**, the return value is **NULL**.

By default, geographic coordinates (latitude, longitude) are interpreted as in the order specified by the spatial reference system of geometry arguments. An optional ***options*** argument may be given to override the default axis order. options consists of a list of comma-separated ***key*=*value***. The only permitted ***key*** value is **axis-order**, with permitted values of **lat-long**, **long-lat** and **srid-defined** (the default).

If the ***options*** argument is **NULL**, the return value is **NULL**. If the ***options*** argument is invalid, an error occurs to indicate why.

If an SRID argument refers to an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

For geographic SRS geometry arguments, if any argument has a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) error occurs.

If a latitude value is not in the range [−90, 90], an [**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) error occurs.

Ranges shown are in degrees. If an SRS uses another unit, the range uses the corresponding values in its unit. The exact range limits deviate slightly due to floating-point arithmetic.

These functions are available for creating geometries from WKT values:

**[ST\_GeomCollFromText(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromtext)*[wkt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromtext)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromtext)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromtext)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromtext)**, [**ST\_GeometryCollectionFromText(*wkt* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomcollfromtext), [**ST\_GeomCollFromTxt(*wkt* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomcollfromtext)

Constructs a **GeometryCollection** value using its WKT representation and SRID.

These functions handle their arguments as described in the introduction to this section.

mysql> **SET @g = "MULTILINESTRING((10 10, 11 11), (9 9, 10 10))";**

mysql> **SELECT ST\_AsText(ST\_GeomCollFromText(@g));**

+--------------------------------------------+

| ST\_AsText(ST\_GeomCollFromText(@g)) |

+--------------------------------------------+

| MULTILINESTRING((10 10,11 11),(9 9,10 10)) |

+--------------------------------------------+

**[ST\_GeomFromText(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromtext)*[wkt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromtext)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromtext)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromtext)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromtext)**, [**ST\_GeometryFromText(*wkt* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromtext)

Constructs a geometry value of any type using its WKT representation and SRID.

These functions handle their arguments as described in the introduction to this section.

**[ST\_LineFromText(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromtext)*[wkt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromtext)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromtext)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromtext)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromtext)**, [**ST\_LineStringFromText(*wkt* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-linefromtext)

Constructs a **LineString** value using its WKT representation and SRID.

These functions handle their arguments as described in the introduction to this section.

**[ST\_MLineFromText(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromtext)*[wkt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromtext)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromtext)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromtext)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromtext)**, [**ST\_MultiLineStringFromText(*wkt* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mlinefromtext)

Constructs a **MultiLineString** value using its WKT representation and SRID.

These functions handle their arguments as described in the introduction to this section.

**[ST\_MPointFromText(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromtext)*[wkt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromtext)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromtext)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromtext)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromtext)**, [**ST\_MultiPointFromText(*wkt* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpointfromtext)

Constructs a **MultiPoint** value using its WKT representation and SRID.

These functions handle their arguments as described in the introduction to this section.

**[ST\_MPolyFromText(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromtext)*[wkt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromtext)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromtext)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromtext)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromtext)**, [**ST\_MultiPolygonFromText(*wkt* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpolyfromtext)

Constructs a **MultiPolygon** value using its WKT representation and SRID.

These functions handle their arguments as described in the introduction to this section.

**[ST\_PointFromText(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromtext)*[wkt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromtext)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromtext)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromtext)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromtext)**

Constructs a **Point** value using its WKT representation and SRID.

[**ST\_PointFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointfromtext) handles its arguments as described in the introduction to this section.

**[ST\_PolyFromText(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromtext)*[wkt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromtext)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromtext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromtext)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromtext)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromtext)**, [**ST\_PolygonFromText(*wkt* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-polyfromtext)

Constructs a **Polygon** value using its WKT representation and SRID.

These functions handle their arguments as described in the introduction to this section.

### 12.17.4 Functions That Create Geometry Values from WKB Values

These functions take as arguments a [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) containing a Well-Known Binary (WKB) representation and, optionally, a spatial reference system identifier (SRID). They return the corresponding geometry. For a description of WKB format, see [Well-Known Binary (WKB) Format](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#gis-wkb-format).

Functions in this section detect arguments in either Cartesian or geographic spatial reference systems (SRSs), and return results appropriate to the SRS.

[**ST\_GeomFromWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromwkb) accepts a WKB value of any geometry type as its first argument. Other functions provide type-specific construction functions for construction of geometry values of each geometry type.

Prior to MySQL 8.0, these functions also accepted geometry objects as returned by the functions in [Section 12.17.5, “MySQL-Specific Functions That Create Geometry Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-mysql-specific-functions). Geometry arguments are no longer permitted and produce an error. To migrate calls from using geometry arguments to using WKB arguments, follow these guidelines:

Rewrite constructs such as **ST\_GeomFromWKB(Point(0, 0))** as **Point(0, 0)**.

Rewrite constructs such as **ST\_GeomFromWKB(Point(0, 0), 4326)** as **ST\_SRID(Point(0, 0), 4326)** or **ST\_GeomFromWKB(ST\_AsWKB(Point(0, 0)), 4326)**.

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If the WKB or SRID argument is **NULL**, the return value is **NULL**.

By default, geographic coordinates (latitude, longitude) are interpreted as in the order specified by the spatial reference system of geometry arguments. An optional ***options*** argument may be given to override the default axis order. options consists of a list of comma-separated ***key*=*value***. The only permitted ***key*** value is **axis-order**, with permitted values of **lat-long**, **long-lat** and **srid-defined** (the default).

If the ***options*** argument is **NULL**, the return value is **NULL**. If the ***options*** argument is invalid, an error occurs to indicate why.

If an SRID argument refers to an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

For geographic SRS geometry arguments, if any argument has a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) error occurs.

If a latitude value is not in the range [−90, 90], an [**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) error occurs.

Ranges shown are in degrees. If an SRS uses another unit, the range uses the corresponding values in its unit. The exact range limits deviate slightly due to floating-point arithmetic.

These functions are available for creating geometries from WKB values:

**[ST\_GeomCollFromWKB(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromwkb)*[wkb](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromwkb)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromwkb)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromwkb)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomcollfromwkb)**, [**ST\_GeometryCollectionFromWKB(*wkb* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomcollfromwkb)

Constructs a **GeometryCollection** value using its WKB representation and SRID.

These functions handle their arguments as described in the introduction to this section.

**[ST\_GeomFromWKB(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromwkb)*[wkb](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromwkb)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromwkb)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromwkb)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromwkb)**, [**ST\_GeometryFromWKB(*wkb* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromwkb)

Constructs a geometry value of any type using its WKB representation and SRID.

These functions handle their arguments as described in the introduction to this section.

**[ST\_LineFromWKB(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromwkb)*[wkb](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromwkb)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromwkb)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromwkb)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-linefromwkb)**, [**ST\_LineStringFromWKB(*wkb* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-linefromwkb)

Constructs a **LineString** value using its WKB representation and SRID.

These functions handle their arguments as described in the introduction to this section.

**[ST\_MLineFromWKB(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromwkb)*[wkb](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromwkb)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromwkb)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromwkb)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mlinefromwkb)**, [**ST\_MultiLineStringFromWKB(*wkb* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mlinefromwkb)

Constructs a **MultiLineString** value using its WKB representation and SRID.

These functions handle their arguments as described in the introduction to this section.

**[ST\_MPointFromWKB(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromwkb)*[wkb](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromwkb)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromwkb)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromwkb)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpointfromwkb)**, [**ST\_MultiPointFromWKB(*wkb* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpointfromwkb)

Constructs a **MultiPoint** value using its WKB representation and SRID.

These functions handle their arguments as described in the introduction to this section.

**[ST\_MPolyFromWKB(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromwkb)*[wkb](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromwkb)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromwkb)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromwkb)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-mpolyfromwkb)**, [**ST\_MultiPolygonFromWKB(*wkb* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-mpolyfromwkb)

Constructs a **MultiPolygon** value using its WKB representation and SRID.

These functions handle their arguments as described in the introduction to this section.

**[ST\_PointFromWKB(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromwkb)*[wkb](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromwkb)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromwkb)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromwkb)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromwkb)**

Constructs a **Point** value using its WKB representation and SRID.

[**ST\_PointFromWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointfromwkb) handles its arguments as described in the introduction to this section.

**[ST\_PolyFromWKB(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromwkb)*[wkb](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromwkb)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromwkb)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromwkb)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromwkb)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-polyfromwkb)**, [**ST\_PolygonFromWKB(*wkb* [, *srid* [, *options*]])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-polyfromwkb)

Constructs a **Polygon** value using its WKB representation and SRID.

These functions handle their arguments as described in the introduction to this section.

### 12.17.5 MySQL-Specific Functions That Create Geometry Values

MySQL provides a set of useful nonstandard functions for creating geometry values. The functions described in this section are MySQL extensions to the OpenGIS specification.

These functions produce geometry objects from either WKB values or geometry objects as arguments. If any argument is not a proper WKB or geometry representation of the proper object type, the return value is **NULL**.

For example, you can insert the geometry return value from [**Point()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_point) directly into a **POINT** column:

INSERT INTO t1 (pt\_col) VALUES(Point(1,2));

**[GeomCollection(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_geomcollection)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_geomcollection)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_geomcollection)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_geomcollection)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_geomcollection)**

Constructs a **GeomCollection** value from the geometry arguments.

[**GeomCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geomcollection) returns all the proper geometries contained in the arguments even if a nonsupported geometry is present.

[**GeomCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geomcollection) with no arguments is permitted as a way to create an empty geometry. Also, functions such as [**ST\_GeomFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromtext) that accept WKT geometry collection arguments understand both OpenGIS **'GEOMETRYCOLLECTION EMPTY'** standard syntax and MySQL **'GEOMETRYCOLLECTION()'** nonstandard syntax.

[**GeomCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geomcollection) and [**GeometryCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geometrycollection) are synonymous, with [**GeomCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geomcollection) the preferred function.

**[GeometryCollection(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_geometrycollection)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_geometrycollection)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_geometrycollection)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_geometrycollection)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_geometrycollection)**

Constructs a **GeomCollection** value from the geometry arguments.

[**GeometryCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geometrycollection) returns all the proper geometries contained in the arguments even if a nonsupported geometry is present.

[**GeometryCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geometrycollection) with no arguments is permitted as a way to create an empty geometry. Also, functions such as [**ST\_GeomFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromtext) that accept WKT geometry collection arguments understand both OpenGIS **'GEOMETRYCOLLECTION EMPTY'** standard syntax and MySQL **'GEOMETRYCOLLECTION()'** nonstandard syntax.

[**GeomCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geomcollection) and [**GeometryCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geometrycollection) are synonymous, with [**GeomCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geomcollection) the preferred function.

**[LineString(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_linestring)*[pt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_linestring)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_linestring)*[pt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_linestring)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_linestring)**

Constructs a **LineString** value from a number of **Point** or WKB **Point** arguments. If the number of arguments is less than two, the return value is **NULL**.

**[MultiLineString(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multilinestring)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multilinestring)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multilinestring)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multilinestring)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multilinestring)**

Constructs a **MultiLineString** value using **LineString** or WKB **LineString** arguments.

**[MultiPoint(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multipoint)*[pt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multipoint)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multipoint)*[pt2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multipoint)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multipoint)**

Constructs a **MultiPoint** value using **Point** or WKB **Point** arguments.

**[MultiPolygon(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multipolygon)*[poly](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multipolygon)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multipolygon)*[poly](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multipolygon)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_multipolygon)**

Constructs a **MultiPolygon** value from a set of **Polygon** or WKB **Polygon** arguments.

**[Point(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_point)*[x](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_point)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_point)*[y](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_point)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_point)**

Constructs a **Point** using its coordinates.

**[Polygon(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_polygon)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_polygon)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_polygon)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_polygon)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_polygon)**

Constructs a **Polygon** value from a number of **LineString** or WKB **LineString** arguments. If any argument does not represent a **LinearRing** (that is, not a closed and simple **LineString**), the return value is **NULL**.

### 12.17.6 Geometry Format Conversion Functions

MySQL supports the functions listed in this section for converting geometry values from internal geometry format to WKT or WKB format, or for swapping the order of X and Y coordinates.

There are also functions to convert a string from WKT or WKB format to internal geometry format. See [Section 12.17.3, “Functions That Create Geometry Values from WKT Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-wkt-functions), and [Section 12.17.4, “Functions That Create Geometry Values from WKB Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-wkb-functions).

Functions such as [**ST\_GeomFromText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geomfromtext) that accept WKT geometry collection arguments understand both OpenGIS **'GEOMETRYCOLLECTION EMPTY'** standard syntax and MySQL **'GEOMETRYCOLLECTION()'** nonstandard syntax. Another way to produce an empty geometry collection is by calling [**GeometryCollection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_geometrycollection) with no arguments. Functions such as [**ST\_AsWKT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-astext) that produce WKT values produce **'GEOMETRYCOLLECTION EMPTY'** standard syntax:

mysql> **SET @s1 = ST\_GeomFromText('GEOMETRYCOLLECTION()');**

mysql> **SET @s2 = ST\_GeomFromText('GEOMETRYCOLLECTION EMPTY');**

mysql> **SELECT ST\_AsWKT(@s1), ST\_AsWKT(@s2);**

+--------------------------+--------------------------+

| ST\_AsWKT(@s1) | ST\_AsWKT(@s2) |

+--------------------------+--------------------------+

| GEOMETRYCOLLECTION EMPTY | GEOMETRYCOLLECTION EMPTY |

+--------------------------+--------------------------+

mysql> **SELECT ST\_AsWKT(GeomCollection());**

+----------------------------+

| ST\_AsWKT(GeomCollection()) |

+----------------------------+

| GEOMETRYCOLLECTION EMPTY |

+----------------------------+

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any argument is **NULL**, the return value is **NULL**.

If any geometry argument is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If any geometry argument is in an undefined spatial reference system, the axes are output in the order they appear in the geometry and an [**ER\_WARN\_SRS\_NOT\_FOUND\_AXIS\_ORDER**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_warn_srs_not_found_axis_order) warning occurs.

By default, geographic coordinates (latitude, longitude) are interpreted as in the order specified by the spatial reference system of geometry arguments. An optional ***options*** argument may be given to override the default axis order. options consists of a list of comma-separated ***key*=*value***. The only permitted ***key*** value is **axis-order**, with permitted values of **lat-long**, **long-lat** and **srid-defined** (the default).

If the ***options*** argument is **NULL**, the return value is **NULL**. If the ***options*** argument is invalid, an error occurs to indicate why.

Otherwise, the return value is non-**NULL**.

These functions are available for format conversions or coordinate swapping:

**[ST\_AsBinary(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asbinary)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asbinary)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asbinary)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asbinary)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asbinary)**, [**ST\_AsWKB(*g* [, *options*])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-asbinary)

Converts a value in internal geometry format to its WKB representation and returns the binary result.

The function return value has geographic coordinates (latitude, longitude) in the order specified by the spatial reference system that applies to the geometry argument. An optional ***options*** argument may be given to override the default axis order.

[**ST\_AsBinary()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-asbinary) and [**ST\_AsWKB()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-asbinary) handle their arguments as described in the introduction to this section.

mysql> **SET @g = ST\_LineFromText('LINESTRING(0 5,5 10,10 15)', 4326);**

mysql> **SELECT ST\_AsText(ST\_GeomFromWKB(ST\_AsWKB(@g)));**

+-----------------------------------------+

| ST\_AsText(ST\_GeomFromWKB(ST\_AsWKB(@g))) |

+-----------------------------------------+

| LINESTRING(5 0,10 5,15 10) |

+-----------------------------------------+

mysql> **SELECT ST\_AsText(ST\_GeomFromWKB(ST\_AsWKB(@g, 'axis-order=long-lat')));**

+----------------------------------------------------------------+

| ST\_AsText(ST\_GeomFromWKB(ST\_AsWKB(@g, 'axis-order=long-lat'))) |

+----------------------------------------------------------------+

| LINESTRING(0 5,5 10,10 15) |

+----------------------------------------------------------------+

mysql> **SELECT ST\_AsText(ST\_GeomFromWKB(ST\_AsWKB(@g, 'axis-order=lat-long')));**

+----------------------------------------------------------------+

| ST\_AsText(ST\_GeomFromWKB(ST\_AsWKB(@g, 'axis-order=lat-long'))) |

+----------------------------------------------------------------+

| LINESTRING(5 0,10 5,15 10) |

+----------------------------------------------------------------+

**[ST\_AsText(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-astext)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-astext)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-astext)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-astext)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-astext)**, [**ST\_AsWKT(*g* [, *options*])**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-astext)

Converts a value in internal geometry format to its WKT representation and returns the string result.

The function return value has geographic coordinates (latitude, longitude) in the order specified by the spatial reference system that applies to the geometry argument. An optional ***options*** argument may be given to override the default axis order.

[**ST\_AsText()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-astext) and [**ST\_AsWKT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-astext) handle their arguments as described in the introduction to this section.

mysql> **SET @g = 'LineString(1 1,2 2,3 3)';**

mysql> **SELECT ST\_AsText(ST\_GeomFromText(@g));**

+--------------------------------+

| ST\_AsText(ST\_GeomFromText(@g)) |

+--------------------------------+

| LINESTRING(1 1,2 2,3 3) |

+--------------------------------+

Output for **MultiPoint** values includes parentheses around each point. For example:

mysql> **SELECT ST\_AsText(ST\_GeomFromText(@mp));**

+---------------------------------+

| ST\_AsText(ST\_GeomFromText(@mp)) |

+---------------------------------+

| MULTIPOINT((1 1),(2 2),(3 3)) |

+---------------------------------+

**[ST\_SwapXY(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-swapxy)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-swapxy)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-swapxy)**

Accepts an argument in internal geometry format, swaps the X and Y values of each coordinate pair within the geometry, and returns the result.

[**ST\_SwapXY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-swapxy) handles its arguments as described in the introduction to this section.

mysql> **SET @g = ST\_LineFromText('LINESTRING(0 5,5 10,10 15)');**

mysql> **SELECT ST\_AsText(@g);**

+----------------------------+

| ST\_AsText(@g) |

+----------------------------+

| LINESTRING(0 5,5 10,10 15) |

+----------------------------+

mysql> **SELECT ST\_AsText(ST\_SwapXY(@g));**

+----------------------------+

| ST\_AsText(ST\_SwapXY(@g)) |

+----------------------------+

| LINESTRING(5 0,10 5,15 10) |

+----------------------------+

### 12.17.7 Geometry Property Functions

[12.17.7.1 General Geometry Property Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-general-property-functions)

[12.17.7.2 Point Property Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-point-property-functions)

[12.17.7.3 LineString and MultiLineString Property Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-linestring-property-functions)

[12.17.7.4 Polygon and MultiPolygon Property Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-polygon-property-functions)

[12.17.7.5 GeometryCollection Property Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-geometrycollection-property-functions)

Each function that belongs to this group takes a geometry value as its argument and returns some quantitative or qualitative property of the geometry. Some functions restrict their argument type. Such functions return **NULL** if the argument is of an incorrect geometry type. For example, the [**ST\_Area()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-area) polygon function returns **NULL** if the object type is neither **Polygon** nor **MultiPolygon**.

#### 12.17.7.1 General Geometry Property Functions

The functions listed in this section do not restrict their argument and accept a geometry value of any type.

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any argument is **NULL**, the return value is **NULL**.

If any geometry argument is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If any geometry argument is a syntactically well-formed geometry in an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

If any SRID argument is not within the range of a 32-bit unsigned integer, an [**ER\_DATA\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_data_out_of_range) error occurs.

If any SRID argument refers to an undefined SRS, an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

Otherwise, the return value is non-**NULL**.

These functions are available for obtaining geometry properties:

**[ST\_Dimension(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-dimension)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-dimension)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-dimension)**

Returns the inherent dimension of the geometry value ***g***. The dimension can be −1, 0, 1, or 2. The meaning of these values is given in [Section 11.4.2.2, “Geometry Class”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#gis-class-geometry).

[**ST\_Dimension()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-dimension) handles its arguments as described in the introduction to this section.

mysql> **SELECT ST\_Dimension(ST\_GeomFromText('LineString(1 1,2 2)'));**

+------------------------------------------------------+

| ST\_Dimension(ST\_GeomFromText('LineString(1 1,2 2)')) |

+------------------------------------------------------+

| 1 |

+------------------------------------------------------+

**[ST\_Envelope(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-envelope)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-envelope)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-envelope)**

Returns the minimum bounding rectangle (MBR) for the geometry value ***g***. The result is returned as a **Polygon** value that is defined by the corner points of the bounding box:

POLYGON((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))

mysql> **SELECT ST\_AsText(ST\_Envelope(ST\_GeomFromText('LineString(1 1,2 2)')));**

+----------------------------------------------------------------+

| ST\_AsText(ST\_Envelope(ST\_GeomFromText('LineString(1 1,2 2)'))) |

+----------------------------------------------------------------+

| POLYGON((1 1,2 1,2 2,1 2,1 1)) |

+----------------------------------------------------------------+

If the argument is a point or a vertical or horizontal line segment, [**ST\_Envelope()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-envelope) returns the point or the line segment as its MBR rather than returning an invalid polygon:

mysql> **SELECT ST\_AsText(ST\_Envelope(ST\_GeomFromText('LineString(1 1,1 2)')));**

+----------------------------------------------------------------+

| ST\_AsText(ST\_Envelope(ST\_GeomFromText('LineString(1 1,1 2)'))) |

+----------------------------------------------------------------+

| LINESTRING(1 1,1 2) |

+----------------------------------------------------------------+

[**ST\_Envelope()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-envelope) handles its arguments as described in the introduction to this section, with this exception:

If the geometry has an SRID value for a geographic spatial reference system (SRS), an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs.

**[ST\_GeometryType(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geometrytype)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geometrytype)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geometrytype)**

Returns a binary string indicating the name of the geometry type of which the geometry instance ***g*** is a member. The name corresponds to one of the instantiable **Geometry** subclasses.

[**ST\_GeometryType()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geometrytype) handles its arguments as described in the introduction to this section.

mysql> **SELECT ST\_GeometryType(ST\_GeomFromText('POINT(1 1)'));**

+------------------------------------------------+

| ST\_GeometryType(ST\_GeomFromText('POINT(1 1)')) |

+------------------------------------------------+

| POINT |

+------------------------------------------------+

**[ST\_IsEmpty(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-isempty)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-isempty)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-isempty)**

This function is a placeholder that returns 1 for an empty geometry collection value or 0 otherwise.

The only valid empty geometry is represented in the form of an empty geometry collection value. MySQL does not support GIS **EMPTY** values such as **POINT EMPTY**.

[**ST\_IsEmpty()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isempty) handles its arguments as described in the introduction to this section.

**[ST\_IsSimple(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-issimple)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-issimple)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-issimple)**

Returns 1 if the geometry value ***g*** is simple according to the ISO SQL/MM Part 3: Spatial standard. [**ST\_IsSimple()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-issimple) returns 0 if the argument is not simple.

The descriptions of the instantiable geometric classes given under [Section 11.4.2, “The OpenGIS Geometry Model”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#opengis-geometry-model) include the specific conditions that cause class instances to be classified as not simple.

[**ST\_IsSimple()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-issimple) handles its arguments as described in the introduction to this section, with this exception:

If the geometry has a geographic SRS with a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_GEOMETRY\_PARAM\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_longitude_out_of_range) error occurs ([**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) prior to MySQL 8.0.12).

If a latitude value is not in the range [−90, 90], an [**ER\_GEOMETRY\_PARAM\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_latitude_out_of_range) error occurs ([**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) prior to MySQL 8.0.12).

Ranges shown are in degrees. The exact range limits deviate slightly due to floating-point arithmetic.

**[ST\_SRID(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-srid)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-srid)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-srid)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-srid)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-srid)**

With a single argument representing a valid geometry object ***g***, [**ST\_SRID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) returns an integer indicating the ID of the spatial reference system (SRS) associated with ***g***.

With the optional second argument representing a valid SRID value, [**ST\_SRID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) returns an object with the same type as its first argument with an SRID value equal to the second argument. This only sets the SRID value of the object; it does not perform any transformation of coordinate values.

[**ST\_SRID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) handles its arguments as described in the introduction to this section, with this exception:

For the single-argument syntax, [**ST\_SRID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) returns the geometry SRID even if it refers to an undefined SRS. An [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error does not occur.

[**ST\_SRID(*g*, *target\_srid*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) and [**ST\_Transform(*g*, *target\_srid*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-transform) differ as follows:

[**ST\_SRID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) changes the geometry SRID value without transforming its coordinates.

[**ST\_Transform()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-transform) transforms the geometry coordinates in addition to changing its SRID value.

mysql> **SET @g = ST\_GeomFromText('LineString(1 1,2 2)', 0);**

mysql> **SELECT ST\_SRID(@g);**

+-------------+

| ST\_SRID(@g) |

+-------------+

| 0 |

+-------------+

mysql> **SET @g = ST\_SRID(@g, 4326);**

mysql> **SELECT ST\_SRID(@g);**

+-------------+

| ST\_SRID(@g) |

+-------------+

| 4326 |

+-------------+

It is possible to create a geometry in a particular SRID by passing to [**ST\_SRID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) the result of one of the MySQL-specific functions for creating spatial values, along with an SRID value. For example:

SET @g1 = ST\_SRID(Point(1, 1), 4326);

However, that method creates the geometry in SRID 0, then casts it to SRID 4326 (WGS 84). A preferable alternative is to create the geometry with the correct spatial reference system to begin with. For example:

SET @g1 = ST\_PointFromText('POINT(1 1)', 4326);

SET @g1 = ST\_GeomFromText('POINT(1 1)', 4326);

The two-argument form of [**ST\_SRID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) is useful for tasks such as correcting or changing the SRS of geometries that have an incorrect SRID.

#### 12.17.7.2 Point Property Functions

A **Point** consists of X and Y coordinates, which may be obtained using the [**ST\_X()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-x) and [**ST\_Y()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-y) functions, respectively. These functions also permit an optional second argument that specifies an X or Y coordinate value, in which case the function result is the **Point** object from the first argument with the appropriate coordinate modified to be equal to the second argument.

For **Point** objects that have a geographic spatial reference system (SRS), the longitude and latitude may be obtained using the [**ST\_Longitude()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-longitude) and [**ST\_Latitude()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latitude) functions, respectively. These functions also permit an optional second argument that specifies a longitude or latitude value, in which case the function result is the **Point** object from the first argument with the longitude or latitude modified to be equal to the second argument.

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any argument is **NULL**, the return value is **NULL**.

If any geometry argument is a valid geometry but not a **Point** object, an [**ER\_UNEXPECTED\_GEOMETRY\_TYPE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_unexpected_geometry_type) error occurs.

If any geometry argument is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If any geometry argument is a syntactically well-formed geometry in an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

If an X or Y coordinate argument is provided and the value is **-inf**, **+inf**, or **NaN**, an [**ER\_DATA\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_data_out_of_range) error occurs.

If a longitude or latitude value is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) error occurs.

If a latitude value is not in the range [−90, 90], an [**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) error occurs.

Ranges shown are in degrees. The exact range limits deviate slightly due to floating-point arithmetic.

Otherwise, the return value is non-**NULL**.

These functions are available for obtaining point properties:

**[ST\_Latitude(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-latitude)*[p](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-latitude)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-latitude)*[new\_latitude\_val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-latitude)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-latitude)**

With a single argument representing a valid **Point** object ***p*** that has a geographic spatial reference system (SRS), [**ST\_Latitude()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latitude) returns the latitude value of ***p*** as a double-precision number.

With the optional second argument representing a valid latitude value, [**ST\_Latitude()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latitude) returns a **Point** object like the first argument with its latitude equal to the second argument.

[**ST\_Latitude()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latitude) handles its arguments as described in the introduction to this section, with the addition that if the **Point** object is valid but does not have a geographic SRS, an [**ER\_SRS\_NOT\_GEOGRAPHIC**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_geographic) error occurs.

mysql> **SET @pt = ST\_GeomFromText('POINT(45 90)', 4326);**

mysql> **SELECT ST\_Latitude(@pt);**

+------------------+

| ST\_Latitude(@pt) |

+------------------+

| 45 |

+------------------+

mysql> **SELECT ST\_AsText(ST\_Latitude(@pt, 10));**

+---------------------------------+

| ST\_AsText(ST\_Latitude(@pt, 10)) |

+---------------------------------+

| POINT(10 90) |

+---------------------------------+

This function was added in MySQL 8.0.12.

**[ST\_Longitude(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-longitude)*[p](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-longitude)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-longitude)*[new\_longitude\_val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-longitude)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-longitude)**

With a single argument representing a valid **Point** object ***p*** that has a geographic spatial reference system (SRS), [**ST\_Longitude()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-longitude) returns the longitude value of ***p*** as a double-precision number.

With the optional second argument representing a valid longitude value, [**ST\_Longitude()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-longitude) returns a **Point** object like the first argument with its longitude equal to the second argument.

[**ST\_Longitude()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-longitude) handles its arguments as described in the introduction to this section, with the addition that if the **Point** object is valid but does not have a geographic SRS, an [**ER\_SRS\_NOT\_GEOGRAPHIC**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_geographic) error occurs.

mysql> **SET @pt = ST\_GeomFromText('POINT(45 90)', 4326);**

mysql> **SELECT ST\_Longitude(@pt);**

+-------------------+

| ST\_Longitude(@pt) |

+-------------------+

| 90 |

+-------------------+

mysql> **SELECT ST\_AsText(ST\_Longitude(@pt, 10));**

+----------------------------------+

| ST\_AsText(ST\_Longitude(@pt, 10)) |

+----------------------------------+

| POINT(45 10) |

+----------------------------------+

This function was added in MySQL 8.0.12.

**[ST\_X(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-x)*[p](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-x)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-x)*[new\_x\_val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-x)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-x)**

With a single argument representing a valid **Point** object ***p***, [**ST\_X()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-x) returns the X-coordinate value of ***p*** as a double-precision number. As of MySQL 8.0.12, the X coordinate is considered to refer to the axis that appears first in the **Point** spatial reference system (SRS) definition.

With the optional second argument, [**ST\_X()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-x) returns a **Point** object like the first argument with its X coordinate equal to the second argument. As of MySQL 8.0.12, if the **Point** object has a geographic SRS, the second argument must be in the proper range for longitude or latitude values.

[**ST\_X()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-x) handles its arguments as described in the introduction to this section.

mysql> **SELECT ST\_X(Point(56.7, 53.34));**

+--------------------------+

| ST\_X(Point(56.7, 53.34)) |

+--------------------------+

| 56.7 |

+--------------------------+

mysql> **SELECT ST\_AsText(ST\_X(Point(56.7, 53.34), 10.5));**

+-------------------------------------------+

| ST\_AsText(ST\_X(Point(56.7, 53.34), 10.5)) |

+-------------------------------------------+

| POINT(10.5 53.34) |

+-------------------------------------------+

**[ST\_Y(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-y)*[p](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-y)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-y)*[new\_y\_val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-y)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-y)**

With a single argument representing a valid **Point** object ***p***, [**ST\_Y()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-y) returns the Y-coordinate value of ***p*** as a double-precision number. As of MySQL 8.0.12, the Y coordinate is considered to refer to the axis that appears second in the **Point** spatial reference system (SRS) definition.

With the optional second argument, [**ST\_Y()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-y) returns a **Point** object like the first argument with its Y coordinate equal to the second argument. As of MySQL 8.0.12, if the **Point** object has a geographic SRS, the second argument must be in the proper range for longitude or latitude values.

[**ST\_Y()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-y) handles its arguments as described in the introduction to this section.

mysql> **SELECT ST\_Y(Point(56.7, 53.34));**

+--------------------------+

| ST\_Y(Point(56.7, 53.34)) |

+--------------------------+

| 53.34 |

+--------------------------+

mysql> **SELECT ST\_AsText(ST\_Y(Point(56.7, 53.34), 10.5));**

+-------------------------------------------+

| ST\_AsText(ST\_Y(Point(56.7, 53.34), 10.5)) |

+-------------------------------------------+

| POINT(56.7 10.5) |

+-------------------------------------------+

#### 12.17.7.3 LineString and MultiLineString Property Functions

A **LineString** consists of **Point** values. You can extract particular points of a **LineString**, count the number of points that it contains, or obtain its length.

Some functions in this section also work for **MultiLineString** values.

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any argument is **NULL** or any geometry argument is an empty geometry, the return value is **NULL**.

If any geometry argument is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If any geometry argument is a syntactically well-formed geometry in an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

Otherwise, the return value is non-**NULL**.

These functions are available for obtaining linestring properties:

**[ST\_EndPoint(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-endpoint)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-endpoint)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-endpoint)**

Returns the **Point** that is the endpoint of the **LineString** value ***ls***.

[**ST\_EndPoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-endpoint) handles its arguments as described in the introduction to this section.

mysql> **SET @ls = 'LineString(1 1,2 2,3 3)';**

mysql> **SELECT ST\_AsText(ST\_EndPoint(ST\_GeomFromText(@ls)));**

+----------------------------------------------+

| ST\_AsText(ST\_EndPoint(ST\_GeomFromText(@ls))) |

+----------------------------------------------+

| POINT(3 3) |

+----------------------------------------------+

**[ST\_IsClosed(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-isclosed)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-isclosed)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-isclosed)**

For a **LineString** value ***ls***, [**ST\_IsClosed()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isclosed) returns 1 if ***ls*** is closed (that is, its [**ST\_StartPoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-startpoint) and [**ST\_EndPoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-endpoint) values are the same).

For a **MultiLineString** value ***ls***, [**ST\_IsClosed()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isclosed) returns 1 if ***ls*** is closed (that is, the [**ST\_StartPoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-startpoint) and [**ST\_EndPoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-endpoint) values are the same for each **LineString** in ***ls***).

[**ST\_IsClosed()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isclosed) returns 0 if ***ls*** is not closed, and **NULL** if ***ls*** is **NULL**.

[**ST\_IsClosed()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isclosed) handles its arguments as described in the introduction to this section, with this exception:

If the geometry has an SRID value for a geographic spatial reference system (SRS), an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs.

mysql> **SET @ls1 = 'LineString(1 1,2 2,3 3,2 2)';**

mysql> **SET @ls2 = 'LineString(1 1,2 2,3 3,1 1)';**

mysql> **SELECT ST\_IsClosed(ST\_GeomFromText(@ls1));**

+------------------------------------+

| ST\_IsClosed(ST\_GeomFromText(@ls1)) |

+------------------------------------+

| 0 |

+------------------------------------+

mysql> **SELECT ST\_IsClosed(ST\_GeomFromText(@ls2));**

+------------------------------------+

| ST\_IsClosed(ST\_GeomFromText(@ls2)) |

+------------------------------------+

| 1 |

+------------------------------------+

mysql> **SET @ls3 = 'MultiLineString((1 1,2 2,3 3),(4 4,5 5))';**

mysql> **SELECT ST\_IsClosed(ST\_GeomFromText(@ls3));**

+------------------------------------+

| ST\_IsClosed(ST\_GeomFromText(@ls3)) |

+------------------------------------+

| 0 |

+------------------------------------+

**[ST\_Length(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-length)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-length)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-length)*[unit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-length)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-length)**

Returns a double-precision number indicating the length of the **LineString** or **MultiLineString** value ***ls*** in its associated spatial reference system. The length of a **MultiLineString** value is equal to the sum of the lengths of its elements.

[**ST\_Length()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-length) computes a result as follows:

If the geometry is a valid **LineString** in a Cartesian SRS, the return value is the Cartesian length of the geometry.

If the geometry is a valid **MultiLineString** in a Cartesian SRS, the return value is the sum of the Cartesian lengths of its elements.

If the geometry is a valid **LineString** in a geographic SRS, the return value is the geodetic length of the geometry in that SRS, in meters.

If the geometry is a valid **MultiLineString** in a geographic SRS, the return value is the sum of the geodetic lengths of its elements in that SRS, in meters.

[**ST\_Length()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-length) handles its arguments as described in the introduction to this section, with these exceptions:

If the geometry is not a **LineString** or **MultiLineString**, the return value is **NULL**.

If the geometry is geometrically invalid, either the result is an undefined length (that is, it can be any number), or an error occurs.

If the length computation result is **+inf**, an [**ER\_DATA\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_data_out_of_range) error occurs.

If the geometry has a geographic SRS with a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_GEOMETRY\_PARAM\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_longitude_out_of_range) error occurs ([**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) prior to MySQL 8.0.12).

If a latitude value is not in the range [−90, 90], an [**ER\_GEOMETRY\_PARAM\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_latitude_out_of_range) error occurs ([**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) prior to MySQL 8.0.12).

Ranges shown are in degrees. The exact range limits deviate slightly due to floating-point arithmetic.

As of MySQL 8.0.16, [**ST\_Length()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-length) permits an optional ***unit*** argument that specifies the linear unit for the returned length value. These rules apply:

If a unit is specified but not supported by MySQL, an [**ER\_UNIT\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_unit_not_found) error occurs.

If a supported linear unit is specified and the SRID is 0, an [**ER\_GEOMETRY\_IN\_UNKNOWN\_LENGTH\_UNIT**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_in_unknown_length_unit) error occurs.

If a supported linear unit is specified and the SRID is not 0, the result is in that unit.

If a unit is not specified, the result is in the unit of the SRS of the geometries, whether Cartesian or geographic. Currently, all MySQL SRSs are expressed in meters.

A unit is supported if it is found in the **INFORMATION\_SCHEMA** [**ST\_UNITS\_OF\_MEASURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-st-units-of-measure-table) table. See [Section 26.3.37, “The INFORMATION\_SCHEMA ST\_UNITS\_OF\_MEASURE Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-st-units-of-measure-table).

mysql> **SET @ls = ST\_GeomFromText('LineString(1 1,2 2,3 3)');**

mysql> **SELECT ST\_Length(@ls);**

+--------------------+

| ST\_Length(@ls) |

+--------------------+

| 2.8284271247461903 |

+--------------------+

mysql> **SET @mls = ST\_GeomFromText('MultiLineString((1 1,2 2,3 3),(4 4,5 5))');**

mysql> **SELECT ST\_Length(@mls);**

+-------------------+

| ST\_Length(@mls) |

+-------------------+

| 4.242640687119286 |

+-------------------+

mysql> **SET @ls = ST\_GeomFromText('LineString(1 1,2 2,3 3)', 4326);**

mysql> **SELECT ST\_Length(@ls);**

+-------------------+

| ST\_Length(@ls) |

+-------------------+

| 313701.9623204328 |

+-------------------+

mysql> **SELECT ST\_Length(@ls, 'metre');**

+-------------------------+

| ST\_Length(@ls, 'metre') |

+-------------------------+

| 313701.9623204328 |

+-------------------------+

mysql> **SELECT ST\_Length(@ls, 'foot');**

+------------------------+

| ST\_Length(@ls, 'foot') |

+------------------------+

| 1029205.9131247795 |

+------------------------+

**[ST\_NumPoints(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-numpoints)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-numpoints)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-numpoints)**

Returns the number of **Point** objects in the **LineString** value ***ls***.

[**ST\_NumPoints()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-numpoints) handles its arguments as described in the introduction to this section.

mysql> **SET @ls = 'LineString(1 1,2 2,3 3)';**

mysql> **SELECT ST\_NumPoints(ST\_GeomFromText(@ls));**

+------------------------------------+

| ST\_NumPoints(ST\_GeomFromText(@ls)) |

+------------------------------------+

| 3 |

+------------------------------------+

**[ST\_PointN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointn)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointn)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointn)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointn)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointn)**

Returns the ***N***-th **Point** in the **Linestring** value ***ls***. Points are numbered beginning with 1.

[**ST\_PointN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointn) handles its arguments as described in the introduction to this section.

mysql> **SET @ls = 'LineString(1 1,2 2,3 3)';**

mysql> **SELECT ST\_AsText(ST\_PointN(ST\_GeomFromText(@ls),2));**

+----------------------------------------------+

| ST\_AsText(ST\_PointN(ST\_GeomFromText(@ls),2)) |

+----------------------------------------------+

| POINT(2 2) |

+----------------------------------------------+

**[ST\_StartPoint(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-startpoint)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-startpoint)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-startpoint)**

Returns the **Point** that is the start point of the **LineString** value ***ls***.

[**ST\_StartPoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-startpoint) handles its arguments as described in the introduction to this section.

mysql> **SET @ls = 'LineString(1 1,2 2,3 3)';**

mysql> **SELECT ST\_AsText(ST\_StartPoint(ST\_GeomFromText(@ls)));**

+------------------------------------------------+

| ST\_AsText(ST\_StartPoint(ST\_GeomFromText(@ls))) |

+------------------------------------------------+

| POINT(1 1) |

+------------------------------------------------+

#### 12.17.7.4 Polygon and MultiPolygon Property Functions

Functions in this section return properties of **Polygon** or **MultiPolygon** values.

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any argument is **NULL** or any geometry argument is an empty geometry, the return value is **NULL**.

If any geometry argument is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If any geometry argument is a syntactically well-formed geometry in an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

For functions that take multiple geometry arguments, if those arguments are not in the same SRS, an [**ER\_GIS\_DIFFERENT\_SRIDS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_different_srids) error occurs.

Otherwise, the return value is non-**NULL**.

These functions are available for obtaining polygon properties:

**[ST\_Area({](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-area)*[poly](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-area)*[|](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-area)*[mpoly](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-area)*[})](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-area)**

Returns a double-precision number indicating the area of the **Polygon** or **MultiPolygon** argument, as measured in its spatial reference system.

As of MySQL 8.0.13, [**ST\_Area()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-area) handles its arguments as described in the introduction to this section, with these exceptions:

If the geometry is geometrically invalid, either the result is an undefined area (that is, it can be any number), or an error occurs.

If the geometry is valid but is not a **Polygon** or **MultiPolygon** object, an [**ER\_UNEXPECTED\_GEOMETRY\_TYPE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_unexpected_geometry_type) error occurs.

If the geometry is a valid **Polygon** in a Cartesian SRS, the result is the Cartesian area of the polygon.

If the geometry is a valid **MultiPolygon** in a Cartesian SRS, the result is the sum of the Cartesian area of the polygons.

If the geometry is a valid **Polygon** in a geographic SRS, the result is the geodetic area of the polygon in that SRS, in square meters.

If the geometry is a valid **MultiPolygon** in a geographic SRS, the result is the sum of geodetic area of the polygons in that SRS, in square meters.

If an area computation results in **+inf**, an [**ER\_DATA\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_data_out_of_range) error occurs.

If the geometry has a geographic SRS with a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_GEOMETRY\_PARAM\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_longitude_out_of_range) error occurs ([**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) prior to MySQL 8.0.12).

If a latitude value is not in the range [−90, 90], an [**ER\_GEOMETRY\_PARAM\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_latitude_out_of_range) error occurs ([**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) prior to MySQL 8.0.12).

Ranges shown are in degrees. The exact range limits deviate slightly due to floating-point arithmetic.

Prior to MySQL 8.0.13, [**ST\_Area()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-area) handles its arguments as described in the introduction to this section, with these exceptions:

For arguments of dimension 0 or 1, the result is 0.

If a geometry is empty, the return value is 0 rather than **NULL**.

For a geometry collection, the result is the sum of the area values of all components. If the geometry collection is empty, its area is returned as 0.

If the geometry has an SRID value for a geographic spatial reference system (SRS), an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs.

mysql> **SET @poly =**

**'Polygon((0 0,0 3,3 0,0 0),(1 1,1 2,2 1,1 1))';**

mysql> **SELECT ST\_Area(ST\_GeomFromText(@poly));**

+---------------------------------+

| ST\_Area(ST\_GeomFromText(@poly)) |

+---------------------------------+

| 4 |

+---------------------------------+

mysql> **SET @mpoly =**

**'MultiPolygon(((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1)))';**

mysql> **SELECT ST\_Area(ST\_GeomFromText(@mpoly));**

+----------------------------------+

| ST\_Area(ST\_GeomFromText(@mpoly)) |

+----------------------------------+

| 8 |

+----------------------------------+

**[ST\_Centroid({](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-centroid)*[poly](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-centroid)*[|](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-centroid)*[mpoly](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-centroid)*[})](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-centroid)**

Returns the mathematical centroid for the **Polygon** or **MultiPolygon** argument as a **Point**. The result is not guaranteed to be on the **MultiPolygon**.

This function processes geometry collections by computing the centroid point for components of highest dimension in the collection. Such components are extracted and made into a single **MultiPolygon**, **MultiLineString**, or **MultiPoint** for centroid computation.

[**ST\_Centroid()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-centroid) handles its arguments as described in the introduction to this section, with these exceptions:

The return value is **NULL** for the additional condition that the argument is an empty geometry collection.

If the geometry has an SRID value for a geographic spatial reference system (SRS), an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs.

mysql> **SET @poly =**

**ST\_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7,5 5))');**

mysql> **SELECT ST\_GeometryType(@poly),ST\_AsText(ST\_Centroid(@poly));**

+------------------------+--------------------------------------------+

| ST\_GeometryType(@poly) | ST\_AsText(ST\_Centroid(@poly)) |

+------------------------+--------------------------------------------+

| POLYGON | POINT(4.958333333333333 4.958333333333333) |

+------------------------+--------------------------------------------+

**[ST\_ExteriorRing(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-exteriorring)*[poly](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-exteriorring)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-exteriorring)**

Returns the exterior ring of the **Polygon** value ***poly*** as a **LineString**.

[**ST\_ExteriorRing()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-exteriorring) handles its arguments as described in the introduction to this section.

mysql> **SET @poly =**

**'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';**

mysql> **SELECT ST\_AsText(ST\_ExteriorRing(ST\_GeomFromText(@poly)));**

+----------------------------------------------------+

| ST\_AsText(ST\_ExteriorRing(ST\_GeomFromText(@poly))) |

+----------------------------------------------------+

| LINESTRING(0 0,0 3,3 3,3 0,0 0) |

+----------------------------------------------------+

**[ST\_InteriorRingN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-interiorringn)*[poly](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-interiorringn)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-interiorringn)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-interiorringn)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-interiorringn)**

Returns the ***N***-th interior ring for the **Polygon** value ***poly*** as a **LineString**. Rings are numbered beginning with 1.

[**ST\_InteriorRingN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-interiorringn) handles its arguments as described in the introduction to this section.

mysql> **SET @poly =**

**'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';**

mysql> **SELECT ST\_AsText(ST\_InteriorRingN(ST\_GeomFromText(@poly),1));**

+-------------------------------------------------------+

| ST\_AsText(ST\_InteriorRingN(ST\_GeomFromText(@poly),1)) |

+-------------------------------------------------------+

| LINESTRING(1 1,1 2,2 2,2 1,1 1) |

+-------------------------------------------------------+

**[ST\_NumInteriorRing(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-numinteriorrings)*[poly](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-numinteriorrings)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-numinteriorrings)**, [**ST\_NumInteriorRings(*poly*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-numinteriorrings)

Returns the number of interior rings in the **Polygon** value ***poly***.

[**ST\_NumInteriorRing()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-numinteriorrings) and [**ST\_NuminteriorRings()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-numinteriorrings) handle their arguments as described in the introduction to this section.

mysql> **SET @poly =**

**'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';**

mysql> **SELECT ST\_NumInteriorRings(ST\_GeomFromText(@poly));**

+---------------------------------------------+

| ST\_NumInteriorRings(ST\_GeomFromText(@poly)) |

+---------------------------------------------+

| 1 |

+---------------------------------------------+

#### 12.17.7.5 GeometryCollection Property Functions

These functions return properties of **GeometryCollection** values.

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any argument is **NULL** or any geometry argument is an empty geometry, the return value is **NULL**.

If any geometry argument is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If any geometry argument is a syntactically well-formed geometry in an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

Otherwise, the return value is non-**NULL**.

These functions are available for obtaining geometry collection properties:

**[ST\_GeometryN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geometryn)*[gc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geometryn)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geometryn)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geometryn)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geometryn)**

Returns the ***N***-th geometry in the **GeometryCollection** value ***gc***. Geometries are numbered beginning with 1.

[**ST\_GeometryN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geometryn) handles its arguments as described in the introduction to this section.

mysql> **SET @gc = 'GeometryCollection(Point(1 1),LineString(2 2, 3 3))';**

mysql> **SELECT ST\_AsText(ST\_GeometryN(ST\_GeomFromText(@gc),1));**

+-------------------------------------------------+

| ST\_AsText(ST\_GeometryN(ST\_GeomFromText(@gc),1)) |

+-------------------------------------------------+

| POINT(1 1) |

+-------------------------------------------------+

**[ST\_NumGeometries(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-numgeometries)*[gc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-numgeometries)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-numgeometries)**

Returns the number of geometries in the **GeometryCollection** value ***gc***.

[**ST\_NumGeometries()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-numgeometries) handles its arguments as described in the introduction to this section.

mysql> **SET @gc = 'GeometryCollection(Point(1 1),LineString(2 2, 3 3))';**

mysql> **SELECT ST\_NumGeometries(ST\_GeomFromText(@gc));**

+----------------------------------------+

| ST\_NumGeometries(ST\_GeomFromText(@gc)) |

+----------------------------------------+

| 2 |

+----------------------------------------+

### 12.17.8 Spatial Operator Functions

OpenGIS proposes a number of functions that can produce geometries. They are designed to implement spatial operators. These functions support all argument type combinations except those that are inapplicable according to the [Open Geospatial Consortium](http://www.opengeospatial.org) specification.

MySQL also implements certain functions that are extensions to OpenGIS, as noted in the function descriptions. In addition, [Section 12.17.7, “Geometry Property Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gis-property-functions), discusses several functions that construct new geometries from existing ones. See that section for descriptions of these functions:

[**ST\_Envelope(*g*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-envelope)

[**ST\_StartPoint(*ls*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-startpoint)

[**ST\_EndPoint(*ls*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-endpoint)

[**ST\_PointN(*ls*, *N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointn)

[**ST\_ExteriorRing(*poly*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-exteriorring)

[**ST\_InteriorRingN(*poly*, *N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-interiorringn)

[**ST\_GeometryN(*gc*, *N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geometryn)

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any argument is **NULL**, the return value is **NULL**.

If any geometry argument is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If any geometry argument is a syntactically well-formed geometry in an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

For functions that take multiple geometry arguments, if those arguments are not in the same SRS, an [**ER\_GIS\_DIFFERENT\_SRIDS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_different_srids) error occurs.

If any geometry argument has an SRID value for a geographic SRS and the function does not handle geographic geometries, an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs.

For geographic SRS geometry arguments, if any argument has a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_GEOMETRY\_PARAM\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_longitude_out_of_range) error occurs ([**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) prior to MySQL 8.0.12).

If a latitude value is not in the range [−90, 90], an [**ER\_GEOMETRY\_PARAM\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_latitude_out_of_range) error occurs ([**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) prior to MySQL 8.0.12).

Ranges shown are in degrees. If an SRS uses another unit, the range uses the corresponding values in its unit. The exact range limits deviate slightly due to floating-point arithmetic.

Otherwise, the return value is non-**NULL**.

These spatial operator functions are available:

**[ST\_Buffer(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer)*[d](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer)*[strategy1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer)*[strategy2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer)*[strategy3](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer)*[]]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer)**

Returns a geometry that represents all points whose distance from the geometry value ***g*** is less than or equal to a distance of ***d***. The result is in the same SRS as the geometry argument.

If the geometry argument is empty, [**ST\_Buffer()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer) returns an empty geometry.

If the distance is 0, [**ST\_Buffer()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer) returns the geometry argument unchanged:

mysql> **SET @pt = ST\_GeomFromText('POINT(0 0)');**

mysql> **SELECT ST\_AsText(ST\_Buffer(@pt, 0));**

+------------------------------+

| ST\_AsText(ST\_Buffer(@pt, 0)) |

+------------------------------+

| POINT(0 0) |

+------------------------------+

If the geometry argument is in a Cartesian SRS:

[**ST\_Buffer()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer) supports negative distances for **Polygon** and **MultiPolygon** values, and for geometry collections containing **Polygon** or **MultiPolygon** values.

If the result is reduced so much that that it disappears, the result is an empty geometry.

An [**ER\_WRONG\_ARGUMENTS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_wrong_arguments) error occurs for [**ST\_Buffer()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer) with a negative distance for **Point**, **MultiPoint**, **LineString**, and **MultiLineString** values, and for geometry collections not containing any **Polygon** or **MultiPolygon** values.

If the geometry argument is in a geographic SRS:

Prior to MySQL 8.0.25, an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs.

As of MySQL 8.0.25, **Point** geometries in a geographic SRS are permitted. For non-**Point** geometries, an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error still occurs.

For MySQL versions that permit geographic **Point** geometries:

If the distance is not negative and no strategies are specified, the function returns the geographic buffer of the **Point** in its SRS. The distance argument must be in the SRS distance unit (currently always meters).

If the distance is negative or any strategy (except **NULL**) is specified, an [**ER\_WRONG\_ARGUMENTS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_wrong_arguments) error occurs.

[**ST\_Buffer()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer) permits up to three optional strategy arguments following the distance argument. Strategies influence buffer computation. These arguments are byte string values produced by the [**ST\_Buffer\_Strategy()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer-strategy) function, to be used for point, join, and end strategies:

Point strategies apply to **Point** and **MultiPoint** geometries. If no point strategy is specified, the default is [**ST\_Buffer\_Strategy('point\_circle', 32)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer-strategy).

Join strategies apply to **LineString**, **MultiLineString**, **Polygon**, and **MultiPolygon** geometries. If no join strategy is specified, the default is [**ST\_Buffer\_Strategy('join\_round', 32)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer-strategy).

End strategies apply to **LineString** and **MultiLineString** geometries. If no end strategy is specified, the default is [**ST\_Buffer\_Strategy('end\_round', 32)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer-strategy).

Up to one strategy of each type may be specified, and they may be given in any order.

If the buffer strategies are invalid, an [**ER\_WRONG\_ARGUMENTS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_wrong_arguments) error occurs. Strategies are invalid under any of these circumstances:

Multiple strategies of a given type (point, join, or end) are specified.

A value that is not a strategy (such as an arbitrary binary string or a number) is passed as a strategy.

A **Point** strategy is passed and the geometry contains no **Point** or **MultiPoint** values.

An end or join strategy is passed and the geometry contains no **LineString**, **Polygon**, **MultiLinestring** or **MultiPolygon** values.

mysql> **SET @pt = ST\_GeomFromText('POINT(0 0)');**

mysql> **SET @pt\_strategy = ST\_Buffer\_Strategy('point\_square');**

mysql> **SELECT ST\_AsText(ST\_Buffer(@pt, 2, @pt\_strategy));**

+--------------------------------------------+

| ST\_AsText(ST\_Buffer(@pt, 2, @pt\_strategy)) |

+--------------------------------------------+

| POLYGON((-2 -2,2 -2,2 2,-2 2,-2 -2)) |

+--------------------------------------------+

mysql> **SET @ls = ST\_GeomFromText('LINESTRING(0 0,0 5,5 5)');**

mysql> **SET @end\_strategy = ST\_Buffer\_Strategy('end\_flat');**

mysql> **SET @join\_strategy = ST\_Buffer\_Strategy('join\_round', 10);**

mysql> **SELECT ST\_AsText(ST\_Buffer(@ls, 5, @end\_strategy, @join\_strategy))**

+---------------------------------------------------------------+

| ST\_AsText(ST\_Buffer(@ls, 5, @end\_strategy, @join\_strategy)) |

+---------------------------------------------------------------+

| POLYGON((5 5,5 10,0 10,-3.5355339059327373 8.535533905932738, |

| -5 5,-5 0,0 0,5 0,5 5)) |

+---------------------------------------------------------------+

**[ST\_Buffer\_Strategy(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer-strategy)*[strategy](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer-strategy)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer-strategy)*[points\_per\_circle](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer-strategy)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-buffer-strategy)**

This function returns a strategy byte string for use with [**ST\_Buffer()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer) to influence buffer computation.

Information about strategies is available at [Boost.org](http://www.boost.org).

The first argument must be a string indicating a strategy option:

For point strategies, permitted values are **'point\_circle'** and **'point\_square'**.

For join strategies, permitted values are **'join\_round'** and **'join\_miter'**.

For end strategies, permitted values are **'end\_round'** and **'end\_flat'**.

If the first argument is **'point\_circle'**, **'join\_round'**, **'join\_miter'**, or **'end\_round'**, the ***points\_per\_circle*** argument must be given as a positive numeric value. The maximum ***points\_per\_circle*** value is the value of the [**max\_points\_in\_geometry**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_points_in_geometry) system variable.

For examples, see the description of [**ST\_Buffer()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer).

[**ST\_Buffer\_Strategy()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-buffer-strategy) handles its arguments as described in the introduction to this section, with these exceptions:

If any argument is invalid, an [**ER\_WRONG\_ARGUMENTS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_wrong_arguments) error occurs.

If the first argument is **'point\_square'** or **'end\_flat'**, the ***points\_per\_circle*** argument must not be given or an [**ER\_WRONG\_ARGUMENTS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_wrong_arguments) error occurs.

**[ST\_ConvexHull(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-convexhull)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-convexhull)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-convexhull)**

Returns a geometry that represents the convex hull of the geometry value ***g***.

This function computes a geometry's convex hull by first checking whether its vertex points are colinear. The function returns a linear hull if so, a polygon hull otherwise. This function processes geometry collections by extracting all vertex points of all components of the collection, creating a **MultiPoint** value from them, and computing its convex hull.

[**ST\_ConvexHull()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-convexhull) handles its arguments as described in the introduction to this section, with this exception:

The return value is **NULL** for the additional condition that the argument is an empty geometry collection.

mysql> **SET @g = 'MULTIPOINT(5 0,25 0,15 10,15 25)';**

mysql> **SELECT ST\_AsText(ST\_ConvexHull(ST\_GeomFromText(@g)));**

+-----------------------------------------------+

| ST\_AsText(ST\_ConvexHull(ST\_GeomFromText(@g))) |

+-----------------------------------------------+

| POLYGON((5 0,25 0,15 25,5 0)) |

+-----------------------------------------------+

**[ST\_Difference(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-difference)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-difference)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-difference)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-difference)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-difference)**

Returns a geometry that represents the point set difference of the geometry values ***g1*** and ***g2***. The result is in the same SRS as the geometry arguments.

As of MySQL 8.0.25, [**ST\_Difference()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-difference) permits arguments in either a Cartesian or a geographic SRS. Prior to MySQL 8.0.25, [**ST\_Difference()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-difference) permits arguments in a Cartesian SRS only; for arguments in a geographic SRS, an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs.

[**ST\_Difference()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-difference) handles its arguments as described in the introduction to this section.

mysql> **SET @g1 = Point(1,1), @g2 = Point(2,2);**

mysql> **SELECT ST\_AsText(ST\_Difference(@g1, @g2));**

+------------------------------------+

| ST\_AsText(ST\_Difference(@g1, @g2)) |

+------------------------------------+

| POINT(1 1) |

+------------------------------------+

**[ST\_Intersection(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-intersection)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-intersection)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-intersection)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-intersection)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-intersection)**

Returns a geometry that represents the point set intersection of the geometry values ***g1*** and ***g2***.

[**ST\_Intersection()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-intersection) handles its arguments as described in the introduction to this section.

mysql> **SET @g1 = ST\_GeomFromText('LineString(1 1, 3 3)');**

mysql> **SET @g2 = ST\_GeomFromText('LineString(1 3, 3 1)');**

mysql> **SELECT ST\_AsText(ST\_Intersection(@g1, @g2));**

+--------------------------------------+

| ST\_AsText(ST\_Intersection(@g1, @g2)) |

+--------------------------------------+

| POINT(2 2) |

+--------------------------------------+

**[ST\_LineInterpolatePoint(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-lineinterpolatepoint)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-lineinterpolatepoint)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-lineinterpolatepoint)*[fractional\_distance](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-lineinterpolatepoint)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-lineinterpolatepoint)**

This function takes a **LineString** geometry and a fractional distance in the range [0.0, 1.0] and returns the **Point** along the **LineString** at the given fraction of the distance from its start point to its endpoint. It can be used to answer questions such as which **Point** lies halfway along the road described by the geometry argument.

The function is implemented for **LineString** geometries in all spatial reference systems, both Cartesian and geographic.

If the ***fractional\_distance*** argument is 1.0, the result may not be exactly the last point of the **LineString** argument but a point close to it due to numerical inaccuracies in approximate-value computations.

A related function, [**ST\_LineInterpolatePoints()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-lineinterpolatepoints), takes similar arguments but returns a **MultiPoint** consisting of **Point** values along the **LineString** at each fraction of the distance from its start point to its endpoint. For examples of both functions, see the [**ST\_LineInterpolatePoints()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-lineinterpolatepoints) description.

[**ST\_LineInterpolatePoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-lineinterpolatepoint) handles its arguments as described in the introduction to this section, with these exceptions:

If the geometry argument is not a **LineString**, an [**ER\_UNEXPECTED\_GEOMETRY\_TYPE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_unexpected_geometry_type) error occurs.

If the fractional distance argument is outside the range [0.0, 1.0], an [**ER\_DATA\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_data_out_of_range) error occurs.

[**ST\_LineInterpolatePoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-lineinterpolatepoint) is a MySQL extension to OpenGIS. This function was added in MySQL 8.0.24.

**[ST\_LineInterpolatePoints(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-lineinterpolatepoints)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-lineinterpolatepoints)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-lineinterpolatepoints)*[fractional\_distance](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-lineinterpolatepoints)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-lineinterpolatepoints)**

This function takes a **LineString** geometry and a fractional distance in the range (0.0, 1.0] and returns the **MultiPoint** consisting of the **LineString** start point, plus **Point** values along the **LineString** at each fraction of the distance from its start point to its endpoint. It can be used to answer questions such as which **Point** values lie every 10% of the way along the road described by the geometry argument.

The function is implemented for **LineString** geometries in all spatial reference systems, both Cartesian and geographic.

If the ***fractional\_distance*** argument divides 1.0 with zero remainder the result may not contain the last point of the **LineString** argument but a point close to it due to numerical inaccuracies in approximate-value computations.

A related function, [**ST\_LineInterpolatePoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-lineinterpolatepoint), takes similar arguments but returns the **Point** along the **LineString** at the given fraction of the distance from its start point to its endpoint.

[**ST\_LineInterpolatePoints()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-lineinterpolatepoints) handles its arguments as described in the introduction to this section, with these exceptions:

If the geometry argument is not a **LineString**, an [**ER\_UNEXPECTED\_GEOMETRY\_TYPE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_unexpected_geometry_type) error occurs.

If the fractional distance argument is outside the range [0.0, 1.0], an [**ER\_DATA\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_data_out_of_range) error occurs.

mysql> **SET @ls1 = ST\_GeomFromText('LINESTRING(0 0,0 5,5 5)');**

mysql> **SELECT ST\_AsText(ST\_LineInterpolatePoint(@ls1, .5));**

+----------------------------------------------+

| ST\_AsText(ST\_LineInterpolatePoint(@ls1, .5)) |

+----------------------------------------------+

| POINT(0 5) |

+----------------------------------------------+

mysql> **SELECT ST\_AsText(ST\_LineInterpolatePoint(@ls1, .75));**

+-----------------------------------------------+

| ST\_AsText(ST\_LineInterpolatePoint(@ls1, .75)) |

+-----------------------------------------------+

| POINT(2.5 5) |

+-----------------------------------------------+

mysql> **SELECT ST\_AsText(ST\_LineInterpolatePoint(@ls1, 1));**

+---------------------------------------------+

| ST\_AsText(ST\_LineInterpolatePoint(@ls1, 1)) |

+---------------------------------------------+

| POINT(5 5) |

+---------------------------------------------+

mysql> **SELECT ST\_AsText(ST\_LineInterpolatePoints(@ls1, .25));**

+------------------------------------------------+

| ST\_AsText(ST\_LineInterpolatePoints(@ls1, .25)) |

+------------------------------------------------+

| MULTIPOINT((0 2.5),(0 5),(2.5 5),(5 5)) |

+------------------------------------------------+

[**ST\_LineInterpolatePoints()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-lineinterpolatepoints) is a MySQL extension to OpenGIS. This function was added in MySQL 8.0.24.

**[ST\_PointAtDistance(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointatdistance)*[ls](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointatdistance)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointatdistance)*[distance](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointatdistance)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointatdistance)**

This function takes a **LineString** geometry and a distance in the range [0.0, [**ST\_Length(*ls*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-length)] measured in the unit of the spatial reference system (SRS) of the **LineString**, and returns the **Point** along the **LineString** at that distance from its start point. It can be used to answer questions such as which **Point** value is 400 meters from the start of the road described by the geometry argument.

The function is implemented for **LineString** geometries in all spatial reference systems, both Cartesian and geographic.

[**ST\_PointAtDistance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointatdistance) handles its arguments as described in the introduction to this section, with these exceptions:

If the geometry argument is not a **LineString**, an [**ER\_UNEXPECTED\_GEOMETRY\_TYPE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_unexpected_geometry_type) error occurs.

If the fractional distance argument is outside the range [0.0, [**ST\_Length(*ls*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-length)], an [**ER\_DATA\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_data_out_of_range) error occurs.

[**ST\_PointAtDistance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointatdistance) is a MySQL extension to OpenGIS. This function was added in MySQL 8.0.24.

**[ST\_SymDifference(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-symdifference)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-symdifference)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-symdifference)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-symdifference)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-symdifference)**

Returns a geometry that represents the point set symmetric difference of the geometry values ***g1*** and ***g2***, which is defined as:

***g1*** symdifference ***g2*** := (***g1*** union ***g2***) difference (***g1*** intersection ***g2***)

Or, in function call notation:

ST\_SymDifference(***g1***, ***g2***) = ST\_Difference(ST\_Union(***g1***, ***g2***), ST\_Intersection(***g1***, ***g2***))

[**ST\_SymDifference()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-symdifference) handles its arguments as described in the introduction to this section.

mysql> **SET @g1 = Point(1,1), @g2 = Point(2,2);**

mysql> **SELECT ST\_AsText(ST\_SymDifference(@g1, @g2));**

+---------------------------------------+

| ST\_AsText(ST\_SymDifference(@g1, @g2)) |

+---------------------------------------+

| MULTIPOINT((1 1),(2 2)) |

+---------------------------------------+

**[ST\_Transform(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-transform)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-transform)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-transform)*[target\_srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-transform)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-transform)**

Transforms a geometry from one spatial reference system (SRS) to another. The return value is a geometry of the same type as the input geometry with all coordinates transformed to the target SRID, ***target\_srid***. Transformation support is limited to geographic SRSs, unless the SRID of the geometry argument is the same as the target SRID value, in which case the return value is the input geometry for any valid SRS.

[**ST\_Transform()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-transform) handles its arguments as described in the introduction to this section, with these exceptions:

Geometry arguments that have an SRID value for a geographic SRS do not produce an error.

If the geometry or target SRID argument has an SRID value that refers to an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

If the geometry is in an SRS that [**ST\_Transform()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-transform) cannot transform from, an [**ER\_TRANSFORM\_SOURCE\_SRS\_NOT\_SUPPORTED**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_transform_source_srs_not_supported) error occurs.

If the target SRID is in an SRS that [**ST\_Transform()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-transform) cannot transform to, an [**ER\_TRANSFORM\_TARGET\_SRS\_NOT\_SUPPORTED**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_transform_target_srs_not_supported) error occurs.

If the geometry is in an SRS that is not WGS 84 and has no TOWGS84 clause, an [**ER\_TRANSFORM\_SOURCE\_SRS\_MISSING\_TOWGS84**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_transform_source_srs_missing_towgs84) error occurs.

If the target SRID is in an SRS that is not WGS 84 and has no TOWGS84 clause, an [**ER\_TRANSFORM\_TARGET\_SRS\_MISSING\_TOWGS84**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_transform_target_srs_missing_towgs84) error occurs.

[**ST\_SRID(*g*, *target\_srid*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) and [**ST\_Transform(*g*, *target\_srid*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-transform) differ as follows:

[**ST\_SRID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-srid) changes the geometry SRID value without transforming its coordinates.

[**ST\_Transform()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-transform) transforms the geometry coordinates in addition to changing its SRID value.

mysql> **SET @p = ST\_GeomFromText('POINT(52.381389 13.064444)', 4326);**

mysql> **SELECT ST\_AsText(@p);**

+----------------------------+

| ST\_AsText(@p) |

+----------------------------+

| POINT(52.381389 13.064444) |

+----------------------------+

mysql> **SET @p = ST\_Transform(@p, 4230);**

mysql> **SELECT ST\_AsText(@p);**

+---------------------------------------------+

| ST\_AsText(@p) |

+---------------------------------------------+

| POINT(52.38208611407426 13.065520672345304) |

+---------------------------------------------+

**[ST\_Union(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-union)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-union)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-union)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-union)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-union)**

Returns a geometry that represents the point set union of the geometry values ***g1*** and ***g2***. The result is in the same SRS as the geometry arguments.

As of MySQL 8.0.25, [**ST\_Union()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-union) permits arguments in either a Cartesian or a geographic SRS. Prior to MySQL 8.0.25, [**ST\_Union()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-union) permits arguments in a Cartesian SRS only; for arguments in a geographic SRS, an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs.

[**ST\_Union()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-union) handles its arguments as described in the introduction to this section.

mysql> **SET @g1 = ST\_GeomFromText('LineString(1 1, 3 3)');**

mysql> **SET @g2 = ST\_GeomFromText('LineString(1 3, 3 1)');**

mysql> **SELECT ST\_AsText(ST\_Union(@g1, @g2));**

+--------------------------------------+

| ST\_AsText(ST\_Union(@g1, @g2)) |

+--------------------------------------+

| MULTILINESTRING((1 1,3 3),(1 3,3 1)) |

+--------------------------------------+

### 12.17.9 Functions That Test Spatial Relations Between Geometry Objects

[12.17.9.1 Spatial Relation Functions That Use Object Shapes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-relation-functions-object-shapes)

[12.17.9.2 Spatial Relation Functions That Use Minimum Bounding Rectangles](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-relation-functions-mbr)

The functions described in this section take two geometries as arguments and return a qualitative or quantitative relation between them.

MySQL implements two sets of functions using function names defined by the OpenGIS specification. One set tests the relationship between two geometry values using precise object shapes, the other set uses object minimum bounding rectangles (MBRs).

#### 12.17.9.1 Spatial Relation Functions That Use Object Shapes

The OpenGIS specification defines the following functions to test the relationship between two geometry values ***g1*** and ***g2***, using precise object shapes. The return values 1 and 0 indicate true and false, respectively, except that distance functions return distance values.

Functions in this section detect arguments in either Cartesian or geographic spatial reference systems (SRSs), and return results appropriate to the SRS.

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any argument is **NULL** or any geometry argument is an empty geometry, the return value is **NULL**.

If any geometry argument is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If any geometry argument is a syntactically well-formed geometry in an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

For functions that take multiple geometry arguments, if those arguments are not in the same SRS, an [**ER\_GIS\_DIFFERENT\_SRIDS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_different_srids) error occurs.

If any geometry argument is geometrically invalid, either the result is true or false (it is undefined which), or an error occurs.

For geographic SRS geometry arguments, if any argument has a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_GEOMETRY\_PARAM\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_longitude_out_of_range) error occurs ([**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) prior to MySQL 8.0.12).

If a latitude value is not in the range [−90, 90], an [**ER\_GEOMETRY\_PARAM\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_latitude_out_of_range) error occurs ([**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) prior to MySQL 8.0.12).

Ranges shown are in degrees. If an SRS uses another unit, the range uses the corresponding values in its unit. The exact range limits deviate slightly due to floating-point arithmetic.

Otherwise, the return value is non-**NULL**.

Some functions in this section permit a unit argument that specifies the length unit for the return value. Unless otherwise specified, functions handle their unit argument as follows:

A unit is supported if it is found in the **INFORMATION\_SCHEMA** [**ST\_UNITS\_OF\_MEASURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-st-units-of-measure-table) table. See [Section 26.3.37, “The INFORMATION\_SCHEMA ST\_UNITS\_OF\_MEASURE Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-st-units-of-measure-table).

If a unit is specified but not supported by MySQL, an [**ER\_UNIT\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_unit_not_found) error occurs.

If a supported linear unit is specified and the SRID is 0, an [**ER\_GEOMETRY\_IN\_UNKNOWN\_LENGTH\_UNIT**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_in_unknown_length_unit) error occurs.

If a supported linear unit is specified and the SRID is not 0, the result is in that unit.

If a unit is not specified, the result is in the unit of the SRS of the geometries, whether Cartesian or geographic. Currently, all MySQL SRSs are expressed in meters.

These object-shape functions are available for testing geometry relationships:

**[ST\_Contains(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-contains)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-contains)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-contains)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-contains)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-contains)**

Returns 1 or 0 to indicate whether ***g1*** completely contains ***g2***. This tests the opposite relationship as [**ST\_Within()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-within).

[**ST\_Contains()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-contains) handles its arguments as described in the introduction to this section.

**[ST\_Crosses(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-crosses)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-crosses)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-crosses)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-crosses)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-crosses)**

Two geometries spatially cross if their spatial relation has the following properties:

Unless ***g1*** and ***g2*** are both of dimension 1: ***g1*** crosses ***g2*** if the interior of ***g2*** has points in common with the interior of ***g1***, but ***g2*** does not cover the entire interior of ***g1***.

If both ***g1*** and ***g2*** are of dimension 1: If the lines cross each other in a finite number of points (that is, no common line segments, only single points in common).

This function returns 1 or 0 to indicate whether ***g1*** spatially crosses ***g2***.

[**ST\_Crosses()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-crosses) handles its arguments as described in the introduction to this section except that the return value is **NULL** for these additional conditions:

***g1*** is of dimension 2 (**Polygon** or **MultiPolygon**).

***g2*** is of dimension 1 (**Point** or **MultiPoint**).

**[ST\_Disjoint(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-disjoint)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-disjoint)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-disjoint)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-disjoint)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-disjoint)**

Returns 1 or 0 to indicate whether ***g1*** is spatially disjoint from (does not intersect) ***g2***.

[**ST\_Disjoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-disjoint) handles its arguments as described in the introduction to this section.

**[ST\_Distance(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance)*[unit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance)**

Returns the distance between ***g1*** and ***g2***, measured in the length unit of the spatial reference system (SRS) of the geometry arguments, or in the unit of the optional ***unit*** argument if that is specified.

This function processes geometry collections by returning the shortest distance among all combinations of the components of the two geometry arguments.

[**ST\_Distance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance) handles its geometry arguments as described in the introduction to this section, with these exceptions:

[**ST\_Distance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance) detects arguments in a geographic (ellipsoidal) spatial reference system and returns the geodetic distance on the ellipsoid. As of MySQL 8.0.18, [**ST\_Distance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance) supports distance calculations for geographic SRS arguments of all geometry types. Prior to MySQL 8.0.18, the only permitted geographic argument types are **Point** and **Point**, or **Point** and **MultiPoint** (in any argument order). If called with other geometry type argument combinations in a geographic SRS, an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs.

If any argument is geometrically invalid, either the result is an undefined distance (that is, it can be any number), or an error occurs.

If an intermediate or final result produces **NaN** or a negative number, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

As of MySQL 8.0.14, [**ST\_Distance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance) permits an optional ***unit*** argument that specifies the linear unit for the returned distance value. [**ST\_Distance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance) handles its ***unit*** argument as described in the introduction to this section.

mysql> **SET @g1 = ST\_GeomFromText('POINT(1 1)');**

mysql> **SET @g2 = ST\_GeomFromText('POINT(2 2)');**

mysql> **SELECT ST\_Distance(@g1, @g2);**

+-----------------------+

| ST\_Distance(@g1, @g2) |

+-----------------------+

| 1.4142135623730951 |

+-----------------------+

mysql> **SET @g1 = ST\_GeomFromText('POINT(1 1)', 4326);**

mysql> **SET @g2 = ST\_GeomFromText('POINT(2 2)', 4326);**

mysql> **SELECT ST\_Distance(@g1, @g2);**

+-----------------------+

| ST\_Distance(@g1, @g2) |

+-----------------------+

| 156874.3859490455 |

+-----------------------+

mysql> **SELECT ST\_Distance(@g1, @g2, 'metre');**

+--------------------------------+

| ST\_Distance(@g1, @g2, 'metre') |

+--------------------------------+

| 156874.3859490455 |

+--------------------------------+

mysql> **SELECT ST\_Distance(@g1, @g2, 'foot');**

+-------------------------------+

| ST\_Distance(@g1, @g2, 'foot') |

+-------------------------------+

| 514679.7439273146 |

+-------------------------------+

For the special case of distance calculations on a sphere, see the [**ST\_Distance\_Sphere()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance-sphere) function.

**[ST\_Equals(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-equals)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-equals)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-equals)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-equals)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-equals)**

Returns 1 or 0 to indicate whether ***g1*** is spatially equal to ***g2***.

[**ST\_Equals()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-equals) handles its arguments as described in the introduction to this section, except that it does not return **NULL** for empty geometry arguments.

mysql> **SET @g1 = Point(1,1), @g2 = Point(2,2);**

mysql> **SELECT ST\_Equals(@g1, @g1), ST\_Equals(@g1, @g2);**

+---------------------+---------------------+

| ST\_Equals(@g1, @g1) | ST\_Equals(@g1, @g2) |

+---------------------+---------------------+

| 1 | 0 |

+---------------------+---------------------+

**[ST\_FrechetDistance(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-frechetdistance)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-frechetdistance)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-frechetdistance)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-frechetdistance)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-frechetdistance)*[unit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-frechetdistance)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-frechetdistance)**

Returns the discrete Fréchet distance between two geometries, reflecting how similar the geometries are. The result is a double-precision number measured in the length unit of the spatial reference system (SRS) of the geometry arguments, or in the length unit of the ***unit*** argument if that argument is given.

This function implements the discrete Fréchet distance, which means it is restricted to distances between the points of the geometries. For example, given two **LineString** arguments, only the points explicitly mentioned in the geometries are considered. Points on the line segments between these points are not considered.

[**ST\_FrechetDistance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-frechetdistance) handles its geometry arguments as described in the introduction to this section, with these exceptions:

The geometries may have a Cartesian or geographic SRS, but only **LineString** values are supported. If the arguments are in the same Cartesian or geographic SRS, but either is not a **LineString**, an [**ER\_NOT\_IMPLEMENTED\_FOR\_CARTESIAN\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_cartesian_srs) or [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs, depending on the SRS type.

[**ST\_FrechetDistance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-frechetdistance) handles its optional ***unit*** argument as described in the introduction to this section.

mysql> **SET @ls1 = ST\_GeomFromText('LINESTRING(0 0,0 5,5 5)');**

mysql> **SET @ls2 = ST\_GeomFromText('LINESTRING(0 1,0 6,3 3,5 6)');**

mysql> **SELECT ST\_FrechetDistance(@ls1, @ls2);**

+--------------------------------+

| ST\_FrechetDistance(@ls1, @ls2) |

+--------------------------------+

| 2.8284271247461903 |

+--------------------------------+

mysql> **SET @ls1 = ST\_GeomFromText('LINESTRING(0 0,0 5,5 5)', 4326);**

mysql> **SET @ls2 = ST\_GeomFromText('LINESTRING(0 1,0 6,3 3,5 6)', 4326);**

mysql> **SELECT ST\_FrechetDistance(@ls1, @ls2);**

+--------------------------------+

| ST\_FrechetDistance(@ls1, @ls2) |

+--------------------------------+

| 313421.1999416798 |

+--------------------------------+

mysql> **SELECT ST\_FrechetDistance(@ls1, @ls2, 'foot');**

+----------------------------------------+

| ST\_FrechetDistance(@ls1, @ls2, 'foot') |

+----------------------------------------+

| 1028284.7767115477 |

+----------------------------------------+

This function was added in MySQL 8.0.23.

**[ST\_HausdorffDistance(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-hausdorffdistance)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-hausdorffdistance)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-hausdorffdistance)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-hausdorffdistance)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-hausdorffdistance)*[unit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-hausdorffdistance)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-hausdorffdistance)**

Returns the discrete Hausdorff distance between two geometries, reflecting how similar the geometries are. The result is a double-precision number measured in the length unit of the spatial reference system (SRS) of the geometry arguments, or in the length unit of the ***unit*** argument if that argument is given.

This function implements the discrete Hausdorff distance, which means it is restricted to distances between the points of the geometries. For example, given two **LineString** arguments, only the points explicitly mentioned in the geometries are considered. Points on the line segments between these points are not considered.

[**ST\_HausdorffDistance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-hausdorffdistance) handles its geometry arguments as described in the introduction to this section, with these exceptions:

If the geometry arguments are in the same Cartesian or geographic SRS, but are not in a supported combination, an [**ER\_NOT\_IMPLEMENTED\_FOR\_CARTESIAN\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_cartesian_srs) or [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs, depending on the SRS type. These combinations are supported:

**LineString** and **LineString**

**Point** and **MultiPoint**

**LineString** and **MultiLineString**

**MultiPoint** and **MultiPoint**

**MultiLineString** and **MultiLineString**

[**ST\_HausdorffDistance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-hausdorffdistance) handles its optional ***unit*** argument as described in the introduction to this section.

mysql> **SET @ls1 = ST\_GeomFromText('LINESTRING(0 0,0 5,5 5)');**

mysql> **SET @ls2 = ST\_GeomFromText('LINESTRING(0 1,0 6,3 3,5 6)');**

mysql> **SELECT ST\_HausdorffDistance(@ls1, @ls2);**

+----------------------------------+

| ST\_HausdorffDistance(@ls1, @ls2) |

+----------------------------------+

| 1 |

+----------------------------------+

mysql> **SET @ls1 = ST\_GeomFromText('LINESTRING(0 0,0 5,5 5)', 4326);**

mysql> **SET @ls2 = ST\_GeomFromText('LINESTRING(0 1,0 6,3 3,5 6)', 4326);**

mysql> **SELECT ST\_HausdorffDistance(@ls1, @ls2);**

+----------------------------------+

| ST\_HausdorffDistance(@ls1, @ls2) |

+----------------------------------+

| 111319.49079326246 |

+----------------------------------+

mysql> **SELECT ST\_HausdorffDistance(@ls1, @ls2, 'foot');**

+------------------------------------------+

| ST\_HausdorffDistance(@ls1, @ls2, 'foot') |

+------------------------------------------+

| 365221.4264870815 |

+------------------------------------------+

This function was added in MySQL 8.0.23.

**[ST\_Intersects(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-intersects)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-intersects)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-intersects)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-intersects)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-intersects)**

Returns 1 or 0 to indicate whether ***g1*** spatially intersects ***g2***.

[**ST\_Intersects()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-intersects) handles its arguments as described in the introduction to this section.

**[ST\_Overlaps(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-overlaps)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-overlaps)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-overlaps)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-overlaps)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-overlaps)**

Two geometries spatially overlap if they intersect and their intersection results in a geometry of the same dimension but not equal to either of the given geometries.

This function returns 1 or 0 to indicate whether ***g1*** spatially overlaps ***g2***.

[**ST\_Overlaps()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-overlaps) handles its arguments as described in the introduction to this section except that the return value is **NULL** for the additional condition that the dimensions of the two geometries are not equal.

**[ST\_Touches(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-touches)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-touches)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-touches)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-touches)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-touches)**

Two geometries spatially touch if their interiors do not intersect, but the boundary of one of the geometries intersects either the boundary or the interior of the other.

This function returns 1 or 0 to indicate whether ***g1*** spatially touches ***g2***.

[**ST\_Touches()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-touches) handles its arguments as described in the introduction to this section except that the return value is **NULL** for the additional condition that both geometries are of dimension 0 (**Point** or **MultiPoint**).

**[ST\_Within(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-within)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-within)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-within)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-within)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-within)**

Returns 1 or 0 to indicate whether ***g1*** is spatially within ***g2***. This tests the opposite relationship as [**ST\_Contains()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-contains).

[**ST\_Within()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-within) handles its arguments as described in the introduction to this section.

#### 12.17.9.2 Spatial Relation Functions That Use Minimum Bounding Rectangles

MySQL provides several MySQL-specific functions that test the relationship between minimum bounding rectangles (MBRs) of two geometries ***g1*** and ***g2***. The return values 1 and 0 indicate true and false, respectively.

The bounding box of a point is interpreted as a point that is both boundary and interior.

The bounding box of a straight horizontal or vertical line is interpreted as a line where the interior of the line is also boundary. The endpoints are boundary points.

If any of the parameters are geometry collections, the interior, boundary, and exterior of those parameters are those of the union of all elements in the collection.

Functions in this section detect arguments in either Cartesian or geographic spatial reference systems (SRSs), and return results appropriate to the SRS.

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any argument is **NULL** or an empty geometry, the return value is **NULL**.

If any geometry argument is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If any geometry argument is a syntactically well-formed geometry in an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

For functions that take multiple geometry arguments, if those arguments are not in the same SRS, an [**ER\_GIS\_DIFFERENT\_SRIDS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_different_srids) error occurs.

If any argument is geometrically invalid, either the result is true or false (it is undefined which), or an error occurs.

For geographic SRS geometry arguments, if any argument has a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_GEOMETRY\_PARAM\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_longitude_out_of_range) error occurs ([**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) prior to MySQL 8.0.12).

If a latitude value is not in the range [−90, 90], an [**ER\_GEOMETRY\_PARAM\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_latitude_out_of_range) error occurs ([**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) prior to MySQL 8.0.12).

Ranges shown are in degrees. If an SRS uses another unit, the range uses the corresponding values in its unit. The exact range limits deviate slightly due to floating-point arithmetic.

Otherwise, the return value is non-**NULL**.

These MBR functions are available for testing geometry relationships:

**[MBRContains(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcontains)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcontains)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcontains)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcontains)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcontains)**

Returns 1 or 0 to indicate whether the minimum bounding rectangle of ***g1*** contains the minimum bounding rectangle of ***g2***. This tests the opposite relationship as [**MBRWithin()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrwithin).

[**MBRContains()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcontains) handles its arguments as described in the introduction to this section.

mysql> **SET @g1 = ST\_GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');**

mysql> **SET @g2 = ST\_GeomFromText('Point(1 1)');**

mysql> **SELECT MBRContains(@g1,@g2), MBRWithin(@g2,@g1);**

+----------------------+--------------------+

| MBRContains(@g1,@g2) | MBRWithin(@g2,@g1) |

+----------------------+--------------------+

| 1 | 1 |

+----------------------+--------------------+

**[MBRCoveredBy(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcoveredby)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcoveredby)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcoveredby)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcoveredby)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcoveredby)**

Returns 1 or 0 to indicate whether the minimum bounding rectangle of ***g1*** is covered by the minimum bounding rectangle of ***g2***. This tests the opposite relationship as [**MBRCovers()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcovers).

[**MBRCoveredBy()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcoveredby) handles its arguments as described in the introduction to this section.

mysql> **SET @g1 = ST\_GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');**

mysql> **SET @g2 = ST\_GeomFromText('Point(1 1)');**

mysql> **SELECT MBRCovers(@g1,@g2), MBRCoveredby(@g1,@g2);**

+--------------------+-----------------------+

| MBRCovers(@g1,@g2) | MBRCoveredby(@g1,@g2) |

+--------------------+-----------------------+

| 1 | 0 |

+--------------------+-----------------------+

mysql> **SELECT MBRCovers(@g2,@g1), MBRCoveredby(@g2,@g1);**

+--------------------+-----------------------+

| MBRCovers(@g2,@g1) | MBRCoveredby(@g2,@g1) |

+--------------------+-----------------------+

| 0 | 1 |

+--------------------+-----------------------+

**[MBRCovers(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcovers)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcovers)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcovers)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcovers)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrcovers)**

Returns 1 or 0 to indicate whether the minimum bounding rectangle of ***g1*** covers the minimum bounding rectangle of ***g2***. This tests the opposite relationship as [**MBRCoveredBy()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcoveredby). See the description of [**MBRCoveredBy()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcoveredby) for examples.

[**MBRCovers()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcovers) handles its arguments as described in the introduction to this section.

**[MBRDisjoint(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrdisjoint)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrdisjoint)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrdisjoint)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrdisjoint)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrdisjoint)**

Returns 1 or 0 to indicate whether the minimum bounding rectangles of the two geometries ***g1*** and ***g2*** are disjoint (do not intersect).

[**MBRDisjoint()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrdisjoint) handles its arguments as described in the introduction to this section.

**[MBREquals(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrequals)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrequals)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrequals)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrequals)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrequals)**

Returns 1 or 0 to indicate whether the minimum bounding rectangles of the two geometries ***g1*** and ***g2*** are the same.

[**MBREquals()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrequals) handles its arguments as described in the introduction to this section, except that it does not return **NULL** for empty geometry arguments.

**[MBRIntersects(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrintersects)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrintersects)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrintersects)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrintersects)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrintersects)**

Returns 1 or 0 to indicate whether the minimum bounding rectangles of the two geometries ***g1*** and ***g2*** intersect.

[**MBRIntersects()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrintersects) handles its arguments as described in the introduction to this section.

**[MBROverlaps(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbroverlaps)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbroverlaps)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbroverlaps)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbroverlaps)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbroverlaps)**

Two geometries spatially overlap if they intersect and their intersection results in a geometry of the same dimension but not equal to either of the given geometries.

This function returns 1 or 0 to indicate whether the minimum bounding rectangles of the two geometries ***g1*** and ***g2*** overlap.

[**MBROverlaps()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbroverlaps) handles its arguments as described in the introduction to this section.

**[MBRTouches(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrtouches)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrtouches)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrtouches)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrtouches)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrtouches)**

Two geometries spatially touch if their interiors do not intersect, but the boundary of one of the geometries intersects either the boundary or the interior of the other.

This function returns 1 or 0 to indicate whether the minimum bounding rectangles of the two geometries ***g1*** and ***g2*** touch.

[**MBRTouches()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrtouches) handles its arguments as described in the introduction to this section.

**[MBRWithin(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrwithin)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrwithin)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrwithin)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrwithin)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_mbrwithin)**

Returns 1 or 0 to indicate whether the minimum bounding rectangle of ***g1*** is within the minimum bounding rectangle of ***g2***. This tests the opposite relationship as [**MBRContains()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrcontains).

[**MBRWithin()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_mbrwithin) handles its arguments as described in the introduction to this section.

mysql> **SET @g1 = ST\_GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');**

mysql> **SET @g2 = ST\_GeomFromText('Polygon((0 0,0 5,5 5,5 0,0 0))');**

mysql> **SELECT MBRWithin(@g1,@g2), MBRWithin(@g2,@g1);**

+--------------------+--------------------+

| MBRWithin(@g1,@g2) | MBRWithin(@g2,@g1) |

+--------------------+--------------------+

| 1 | 0 |

+--------------------+--------------------+

### 12.17.10 Spatial Geohash Functions

Geohash is a system for encoding latitude and longitude coordinates of arbitrary precision into a text string. Geohash values are strings that contain only characters chosen from **"0123456789bcdefghjkmnpqrstuvwxyz"**.

The functions in this section enable manipulation of geohash values, which provides applications the capabilities of importing and exporting geohash data, and of indexing and searching geohash values.

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any argument is **NULL**, the return value is **NULL**.

If any argument is invalid, an error occurs.

If any argument has a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_GEOMETRY\_PARAM\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_longitude_out_of_range) error occurs ([**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) prior to MySQL 8.0.12).

If a latitude value is not in the range [−90, 90], an [**ER\_GEOMETRY\_PARAM\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_latitude_out_of_range) error occurs ([**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) prior to MySQL 8.0.12).

Ranges shown are in degrees. The exact range limits deviate slightly due to floating-point arithmetic.

If any point argument does not have SRID 0 or 4326, an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs. ***point*** argument SRID validity is not checked.

If any SRID argument refers to an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

If any SRID argument is not within the range of a 32-bit unsigned integer, an [**ER\_DATA\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_data_out_of_range) error occurs.

Otherwise, the return value is non-**NULL**.

These geohash functions are available:

**[ST\_GeoHash(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geohash)*[longitude](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geohash)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geohash)*[latitude](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geohash)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geohash)*[max\_length](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geohash)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geohash)**, [**ST\_GeoHash(*point*, *max\_length*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geohash)

Returns a geohash string in the connection character set and collation.

For the first syntax, the ***longitude*** must be a number in the range [−180, 180], and the ***latitude*** must be a number in the range [−90, 90]. For the second syntax, a **POINT** value is required, where the X and Y coordinates are in the valid ranges for longitude and latitude, respectively.

The resulting string is no longer than ***max\_length*** characters, which has an upper limit of 100. The string might be shorter than ***max\_length*** characters because the algorithm that creates the geohash value continues until it has created a string that is either an exact representation of the location or ***max\_length*** characters, whichever comes first.

[**ST\_GeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-geohash) handles its arguments as described in the introduction to this section.

mysql> **SELECT ST\_GeoHash(180,0,10), ST\_GeoHash(-180,-90,15);**

+----------------------+-------------------------+

| ST\_GeoHash(180,0,10) | ST\_GeoHash(-180,-90,15) |

+----------------------+-------------------------+

| xbpbpbpbpb | 000000000000000 |

+----------------------+-------------------------+

**[ST\_LatFromGeoHash(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-latfromgeohash)*[geohash\_str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-latfromgeohash)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-latfromgeohash)**

Returns the latitude from a geohash string value, as a double-precision number in the range [−90, 90].

The [**ST\_LatFromGeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latfromgeohash) decoding function reads no more than 433 characters from the ***geohash\_str*** argument. That represents the upper limit on information in the internal representation of coordinate values. Characters past the 433rd are ignored, even if they are otherwise illegal and produce an error.

[**ST\_LatFromGeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latfromgeohash) handles its arguments as described in the introduction to this section.

mysql> **SELECT ST\_LatFromGeoHash(ST\_GeoHash(45,-20,10));**

+------------------------------------------+

| ST\_LatFromGeoHash(ST\_GeoHash(45,-20,10)) |

+------------------------------------------+

| -20 |

+------------------------------------------+

**[ST\_LongFromGeoHash(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-longfromgeohash)*[geohash\_str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-longfromgeohash)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-longfromgeohash)**

Returns the longitude from a geohash string value, as a double-precision number in the range [−180, 180].

The remarks in the description of [**ST\_LatFromGeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latfromgeohash) regarding the maximum number of characters processed from the ***geohash\_str*** argument also apply to [**ST\_LongFromGeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-longfromgeohash).

[**ST\_LongFromGeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-longfromgeohash) handles its arguments as described in the introduction to this section.

mysql> **SELECT ST\_LongFromGeoHash(ST\_GeoHash(45,-20,10));**

+-------------------------------------------+

| ST\_LongFromGeoHash(ST\_GeoHash(45,-20,10)) |

+-------------------------------------------+

| 45 |

+-------------------------------------------+

**[ST\_PointFromGeoHash(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromgeohash)*[geohash\_str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromgeohash)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromgeohash)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromgeohash)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-pointfromgeohash)**

Returns a **POINT** value containing the decoded geohash value, given a geohash string value.

The X and Y coordinates of the point are the longitude in the range [−180, 180] and the latitude in the range [−90, 90], respectively.

The ***srid*** argument is an 32-bit unsigned integer.

The remarks in the description of [**ST\_LatFromGeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-latfromgeohash) regarding the maximum number of characters processed from the ***geohash\_str*** argument also apply to [**ST\_PointFromGeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointfromgeohash).

[**ST\_PointFromGeoHash()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-pointfromgeohash) handles its arguments as described in the introduction to this section.

mysql> **SET @gh = ST\_GeoHash(45,-20,10);**

mysql> **SELECT ST\_AsText(ST\_PointFromGeoHash(@gh,0));**

+---------------------------------------+

| ST\_AsText(ST\_PointFromGeoHash(@gh,0)) |

+---------------------------------------+

| POINT(45 -20) |

+---------------------------------------+

### 12.17.11 Spatial GeoJSON Functions

This section describes functions for converting between GeoJSON documents and spatial values. GeoJSON is an open standard for encoding geometric/geographical features. For more information, see <http://geojson.org>. The functions discussed here follow GeoJSON specification revision 1.0.

GeoJSON supports the same geometric/geographic data types that MySQL supports. Feature and FeatureCollection objects are not supported, except that geometry objects are extracted from them. CRS support is limited to values that identify an SRID.

MySQL also supports a native [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) data type and a set of SQL functions to enable operations on JSON values. For more information, see [Section 11.5, “The JSON Data Type”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json), and [Section 12.18, “JSON Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-functions).

**[ST\_AsGeoJSON(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asgeojson)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asgeojson)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asgeojson)*[max\_dec\_digits](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asgeojson)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asgeojson)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asgeojson)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-asgeojson)**

Generates a GeoJSON object from the geometry ***g***. The object string has the connection character set and collation.

If any argument is **NULL**, the return value is **NULL**. If any non-**NULL** argument is invalid, an error occurs.

***max\_dec\_digits***, if specified, limits the number of decimal digits for coordinates and causes rounding of output. If not specified, this argument defaults to its maximum value of 232 − 1. The minimum is 0.

***options***, if specified, is a bitmask. The following table shows the permitted flag values. If the geometry argument has an SRID of 0, no CRS object is produced even for those flag values that request one.

| **Flag Value** | **Meaning** |
| --- | --- |
| 0 | No options. This is the default if ***options*** is not specified. |
| 1 | Add a bounding box to the output. |
| 2 | Add a short-format CRS URN to the output. The default format is a short format (**EPSG:*srid***). |
| 4 | Add a long-format CRS URN (**urn:ogc:def:crs:EPSG::*srid***). This flag overrides flag 2. For example, option values of 5 and 7 mean the same (add a bounding box and a long-format CRS URN). |

mysql> **SELECT ST\_AsGeoJSON(ST\_GeomFromText('POINT(11.11111 12.22222)'),2);**

+-------------------------------------------------------------+

| ST\_AsGeoJSON(ST\_GeomFromText('POINT(11.11111 12.22222)'),2) |

+-------------------------------------------------------------+

| {"type": "Point", "coordinates": [11.11, 12.22]} |

+-------------------------------------------------------------+

**[ST\_GeomFromGeoJSON(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromgeojson)*[str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromgeojson)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromgeojson)*[options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromgeojson)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromgeojson)*[srid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromgeojson)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-geomfromgeojson)**

Parses a string ***str*** representing a GeoJSON object and returns a geometry.

If any argument is **NULL**, the return value is **NULL**. If any non-**NULL** argument is invalid, an error occurs.

***options***, if given, describes how to handle GeoJSON documents that contain geometries with coordinate dimensions higher than 2. The following table shows the permitted ***options*** values.

| **Option Value** | **Meaning** |
| --- | --- |
| 1 | Reject the document and produce an error. This is the default if ***options*** is not specified. |
| 2, 3, 4 | Accept the document and strip off the coordinates for higher coordinate dimensions. |

***options*** values of 2, 3, and 4 currently produce the same effect. If geometries with coordinate dimensions higher than 2 are supported in the future, you can expect these values to produce different effects.

The ***srid*** argument, if given, must be a 32-bit unsigned integer. If not given, the geometry return value has an SRID of 4326.

If ***srid*** refers to an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

For geographic SRS geometry arguments, if any argument has a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) error occurs.

If a latitude value is not in the range [−90, 90], an [**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) error occurs.

Ranges shown are in degrees. If an SRS uses another unit, the range uses the corresponding values in its unit. The exact range limits deviate slightly due to floating-point arithmetic.

GeoJSON geometry, feature, and feature collection objects may have a **crs** property. The parsing function parses named CRS URNs in the **urn:ogc:def:crs:EPSG::*srid*** and **EPSG:*srid*** namespaces, but not CRSs given as link objects. Also, **urn:ogc:def:crs:OGC:1.3:CRS84** is recognized as SRID 4326. If an object has a CRS that is not understood, an error occurs, with the exception that if the optional ***srid*** argument is given, any CRS is ignored even if it is invalid.

If a **crs** member that specifies an SRID different from the top-level object SRID is found at a lower level of the GeoJSON document, an [**ER\_INVALID\_GEOJSON\_CRS\_NOT\_TOP\_LEVEL**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_geojson_crs_not_top_level) error occurs.

As specified in the GeoJSON specification, parsing is case-sensitive for the **type** member of the GeoJSON input (**Point**, **LineString**, and so forth). The specification is silent regarding case sensitivity for other parsing, which in MySQL is not case-sensitive.

This example shows the parsing result for a simple GeoJSON object. Observe that the order of coordinates depends on the SRID used.

mysql> **SET @json = '{ "type": "Point", "coordinates": [102.0, 0.0]}';**

mysql> **SELECT ST\_AsText(ST\_GeomFromGeoJSON(@json));**

+--------------------------------------+

| ST\_AsText(ST\_GeomFromGeoJSON(@json)) |

+--------------------------------------+

| POINT(0 102) |

+--------------------------------------+

mysql> **SELECT ST\_SRID(ST\_GeomFromGeoJSON(@json));**

+------------------------------------+

| ST\_SRID(ST\_GeomFromGeoJSON(@json)) |

+------------------------------------+

| 4326 |

+------------------------------------+

mysql> **SELECT ST\_AsText(ST\_SRID(ST\_GeomFromGeoJSON(@json),0));**

+-------------------------------------------------+

| ST\_AsText(ST\_SRID(ST\_GeomFromGeoJSON(@json),0)) |

+-------------------------------------------------+

| POINT(102 0) |

+-------------------------------------------------+

### 12.17.12 Spatial Aggregate Functions

MySQL supports aggregate functions that perform a calculation on a set of values. For general information about these functions, see [Section 12.20.1, “Aggregate Function Descriptions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#aggregate-functions). This section describes the [**ST\_Collect()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-collect) spatial aggregate function.

[**ST\_Collect()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-collect) can be used as a window function, as signified in its syntax description by **[*over\_clause*]**, representing an optional **OVER** clause. ***over\_clause*** is described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage), which also includes other information about window function usage.

**[ST\_Collect([DISTINCT]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-collect)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-collect)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-collect)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-collect)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-collect)**

Aggregates geometry values and returns a single geometry collection value. With the **DISTINCT** option, returns the aggregation of the distinct geometry arguments.

As with other aggregate functions, **GROUP BY** may be used to group arguments into subsets. [**ST\_Collect()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-collect) returns an aggregate value for each subset.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage). In constrast to most aggregate functions that support windowing, [**ST\_Collect()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-collect) permits use of ***over\_clause*** together with **DISTINCT**.

[**ST\_Collect()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-collect) handles its arguments as follows:

**NULL** arguments are ignored.

If all arguments are **NULL** or the aggregate result is empty, the return value is **NULL**.

If any geometry argument is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If any geometry argument is a syntactically well-formed geometry in an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

If there are multiple geometry arguments and those arguments are in the same SRS, the return value is in that SRS. If those arguments are not in the same SRS, an [**ER\_GIS\_DIFFERENT\_SRIDS\_AGGREGATION**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_different_srids_aggregation) error occurs.

The result is the narrowest **Multi*Xxx*** or **GeometryCollection** value possible, with the result type determined from the non-**NULL** geometry arguments as follows:

If all arguments are **Point** values, the result is a **MultiPoint** value.

If all arguments are **LineString** values, the result is a **MultiLineString** value.

If all arguments are **Polygon** values, the result is a **MultiPolygon** value.

Otherwise, the arguments are a mix of geometry types and the result is a **GeometryCollection** value.

This example data set shows hypothetical products by year and location of manufacture:

CREATE TABLE product (

year INTEGER,

product VARCHAR(256),

location Geometry

);

INSERT INTO product

(year, product, location) VALUES

(2000, "Calculator", ST\_GeomFromText('point(60 -24)',4326)),

(2000, "Computer" , ST\_GeomFromText('point(28 -77)',4326)),

(2000, "Abacus" , ST\_GeomFromText('point(28 -77)',4326)),

(2000, "TV" , ST\_GeomFromText('point(38 60)',4326)),

(2001, "Calculator", ST\_GeomFromText('point(60 -24)',4326)),

(2001, "Computer" , ST\_GeomFromText('point(28 -77)',4326));

Some sample queries using [**ST\_Collect()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-collect) on the data set:

mysql> **SELECT ST\_AsText(ST\_Collect(location)) AS result**

**FROM product;**

+------------------------------------------------------------------+

| result |

+------------------------------------------------------------------+

| MULTIPOINT((60 -24),(28 -77),(28 -77),(38 60),(60 -24),(28 -77)) |

+------------------------------------------------------------------+

mysql> **SELECT ST\_AsText(ST\_Collect(DISTINCT location)) AS result**

**FROM product;**

+---------------------------------------+

| result |

+---------------------------------------+

| MULTIPOINT((60 -24),(28 -77),(38 60)) |

+---------------------------------------+

mysql> **SELECT year, ST\_AsText(ST\_Collect(location)) AS result**

**FROM product GROUP BY year;**

+------+------------------------------------------------+

| year | result |

+------+------------------------------------------------+

| 2000 | MULTIPOINT((60 -24),(28 -77),(28 -77),(38 60)) |

| 2001 | MULTIPOINT((60 -24),(28 -77)) |

+------+------------------------------------------------+

mysql> **SELECT year, ST\_AsText(ST\_Collect(DISTINCT location)) AS result**

**FROM product GROUP BY year;**

+------+---------------------------------------+

| year | result |

+------+---------------------------------------+

| 2000 | MULTIPOINT((60 -24),(28 -77),(38 60)) |

| 2001 | MULTIPOINT((60 -24),(28 -77)) |

+------+---------------------------------------+

# selects nothing

mysql> **SELECT ST\_Collect(location) AS result**

**FROM product WHERE year = 1999;**

+--------+

| result |

+--------+

| NULL |

+--------+

mysql> **SELECT ST\_AsText(ST\_Collect(location)**

**OVER (ORDER BY year, product ROWS BETWEEN 1 PRECEDING AND CURRENT ROW))**

**AS result**

**FROM product;**

+-------------------------------+

| result |

+-------------------------------+

| MULTIPOINT((28 -77)) |

| MULTIPOINT((28 -77),(60 -24)) |

| MULTIPOINT((60 -24),(28 -77)) |

| MULTIPOINT((28 -77),(38 60)) |

| MULTIPOINT((38 60),(60 -24)) |

| MULTIPOINT((60 -24),(28 -77)) |

+-------------------------------+

This function was added in MySQL 8.0.24.

### 12.17.13 Spatial Convenience Functions

The functions in this section provide convenience operations on geometry values.

Unless otherwise specified, functions in this section handle their geometry arguments as follows:

If any argument is **NULL**, the return value is **NULL**.

If any geometry argument is not a syntactically well-formed geometry, an [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs.

If any geometry argument is a syntactically well-formed geometry in an undefined spatial reference system (SRS), an [**ER\_SRS\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_srs_not_found) error occurs.

For functions that take multiple geometry arguments, if those arguments are not in the same SRS, an [**ER\_GIS\_DIFFERENT\_SRIDS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_different_srids) error occurs.

Otherwise, the return value is non-**NULL**.

These convenience functions are available:

**[ST\_Distance\_Sphere(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance-sphere)*[g1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance-sphere)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance-sphere)*[g2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance-sphere)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance-sphere)*[radius](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance-sphere)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-distance-sphere)**

Returns the mimimum spherical distance between **Point** or **MultiPoint** arguments on a sphere, in meters. (For general-purpose distance calculations, see the [**ST\_Distance()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance) function.) The optional ***radius*** argument should be given in meters.

If both geometry parameters are valid Cartesian **Point** or **MultiPoint** values in SRID 0, the return value is shortest distance between the two geometries on a sphere with the provided radius. If omitted, the default radius is 6,370,986 meters, Point X and Y coordinates are interpreted as longitude and latitude, respectively, in degrees.

If both geometry parameters are valid **Point** or **MultiPoint** values in a geographic spatial reference system (SRS), the return value is the shortest distance between the two geometries on a sphere with the provided radius. If omitted, the default radius is equal to the mean radius, defined as (2a+b)/3, where a is the semi-major axis and b is the semi-minor axis of the SRS.

[**ST\_Distance\_Sphere()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-distance-sphere) handles its arguments as described in the introduction to this section, with these exceptions:

Supported geometry argument combinations are **Point** and **Point**, or **Point** and **MultiPoint** (in any argument order). If at least one of the geometries is neither **Point** nor **MultiPoint**, and its SRID is 0, an [**ER\_NOT\_IMPLEMENTED\_FOR\_CARTESIAN\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_cartesian_srs) error occurs. If at least one of the geometries is neither **Point** nor **MultiPoint**, and its SRID refers to a geographic SRS, an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs. If any geometry refers to a projected SRS, an [**ER\_NOT\_IMPLEMENTED\_FOR\_PROJECTED\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_projected_srs) error occurs.

If any argument has a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_GEOMETRY\_PARAM\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_longitude_out_of_range) error occurs ([**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) prior to MySQL 8.0.12).

If a latitude value is not in the range [−90, 90], an [**ER\_GEOMETRY\_PARAM\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_latitude_out_of_range) error occurs ([**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) prior to MySQL 8.0.12).

Ranges shown are in degrees. If an SRS uses another unit, the range uses the corresponding values in its unit. The exact range limits deviate slightly due to floating-point arithmetic.

If the ***radius*** argument is present but not positive, an [**ER\_NONPOSITIVE\_RADIUS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_nonpositive_radius) error occurs.

If the distance exceeds the range of a double-precision number, an [**ER\_STD\_OVERFLOW\_ERROR**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_std_overflow_error) error occurs.

mysql> **SET @pt1 = ST\_GeomFromText('POINT(0 0)');**

mysql> **SET @pt2 = ST\_GeomFromText('POINT(180 0)');**

mysql> **SELECT ST\_Distance\_Sphere(@pt1, @pt2);**

+--------------------------------+

| ST\_Distance\_Sphere(@pt1, @pt2) |

+--------------------------------+

| 20015042.813723423 |

+--------------------------------+

**[ST\_IsValid(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-isvalid)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-isvalid)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-isvalid)**

Returns 1 if the argument is geometrically valid, 0 if the argument is not geometrically valid. Geometry validity is defined by the OGC specification.

The only valid empty geometry is represented in the form of an empty geometry collection value. [**ST\_IsValid()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isvalid) returns 1 in this case. MySQL does not support GIS **EMPTY** values such as **POINT EMPTY**.

[**ST\_IsValid()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isvalid) handles its arguments as described in the introduction to this section, with this exception:

If the geometry has a geographic SRS with a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_GEOMETRY\_PARAM\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_longitude_out_of_range) error occurs ([**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) prior to MySQL 8.0.12).

If a latitude value is not in the range [−90, 90], an [**ER\_GEOMETRY\_PARAM\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_latitude_out_of_range) error occurs ([**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) prior to MySQL 8.0.12).

Ranges shown are in degrees. If an SRS uses another unit, the range uses the corresponding values in its unit. The exact range limits deviate slightly due to floating-point arithmetic.

mysql> **SET @ls1 = ST\_GeomFromText('LINESTRING(0 0,-0.00 0,0.0 0)');**

mysql> **SET @ls2 = ST\_GeomFromText('LINESTRING(0 0, 1 1)');**

mysql> **SELECT ST\_IsValid(@ls1);**

+------------------+

| ST\_IsValid(@ls1) |

+------------------+

| 0 |

+------------------+

mysql> **SELECT ST\_IsValid(@ls2);**

+------------------+

| ST\_IsValid(@ls2) |

+------------------+

| 1 |

+------------------+

**[ST\_MakeEnvelope(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-makeenvelope)*[pt1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-makeenvelope)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-makeenvelope)*[pt2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-makeenvelope)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-makeenvelope)**

Returns the rectangle that forms the envelope around two points, as a **Point**, **LineString**, or **Polygon**.

Calculations are done using the Cartesian coordinate system rather than on a sphere, spheroid, or on earth.

Given two points ***pt1*** and ***pt2***, [**ST\_MakeEnvelope()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-makeenvelope) creates the result geometry on an abstract plane like this:

If ***pt1*** and ***pt2*** are equal, the result is the point ***pt1***.

Otherwise, if **(*pt1*, *pt2*)** is a vertical or horizontal line segment, the result is the line segment **(*pt1*, *pt2*)**.

Otherwise, the result is a polygon using ***pt1*** and ***pt2*** as diagonal points.

The result geometry has an SRID of 0.

[**ST\_MakeEnvelope()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-makeenvelope) handles its arguments as described in the introduction to this section, with these exceptions:

If the arguments are not **Point** values, an [**ER\_WRONG\_ARGUMENTS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_wrong_arguments) error occurs.

An [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error occurs for the additional condition that any coordinate value of the two points is infinite or **NaN**.

If any geometry has an SRID value for a geographic spatial reference system (SRS), an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs.

mysql> **SET @pt1 = ST\_GeomFromText('POINT(0 0)');**

mysql> **SET @pt2 = ST\_GeomFromText('POINT(1 1)');**

mysql> **SELECT ST\_AsText(ST\_MakeEnvelope(@pt1, @pt2));**

+----------------------------------------+

| ST\_AsText(ST\_MakeEnvelope(@pt1, @pt2)) |

+----------------------------------------+

| POLYGON((0 0,1 0,1 1,0 1,0 0)) |

+----------------------------------------+

**[ST\_Simplify(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-simplify)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-simplify)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-simplify)*[max\_distance](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-simplify)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-simplify)**

Simplifies a geometry using the Douglas-Peucker algorithm and returns a simplified value of the same type.

The geometry may be any geometry type, although the Douglas-Peucker algorithm may not actually process every type. A geometry collection is processed by giving its components one by one to the simplification algorithm, and the returned geometries are put into a geometry collection as result.

The ***max\_distance*** argument is the distance (in units of the input coordinates) of a vertex to other segments to be removed. Vertices within this distance of the simplified linestring are removed.

According to Boost.Geometry, geometries might become invalid as a result of the simplification process, and the process might create self-intersections. To check the validity of the result, pass it to [**ST\_IsValid()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-isvalid).

[**ST\_Simplify()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-simplify) handles its arguments as described in the introduction to this section, with this exception:

If the ***max\_distance*** argument is not positive, or is **NaN**, an [**ER\_WRONG\_ARGUMENTS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_wrong_arguments) error occurs.

mysql> **SET @g = ST\_GeomFromText('LINESTRING(0 0,0 1,1 1,1 2,2 2,2 3,3 3)');**

mysql> **SELECT ST\_AsText(ST\_Simplify(@g, 0.5));**

+---------------------------------+

| ST\_AsText(ST\_Simplify(@g, 0.5)) |

+---------------------------------+

| LINESTRING(0 0,0 1,1 1,2 3,3 3) |

+---------------------------------+

mysql> **SELECT ST\_AsText(ST\_Simplify(@g, 1.0));**

+---------------------------------+

| ST\_AsText(ST\_Simplify(@g, 1.0)) |

+---------------------------------+

| LINESTRING(0 0,3 3) |

+---------------------------------+

**[ST\_Validate(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-validate)*[g](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-validate)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_st-validate)**

Validates a geometry according to the OGC specification. A geometry can be syntactically well-formed (WKB value plus SRID) but geometrically invalid. For example, this polygon is geometrically invalid: **POLYGON((0 0, 0 0, 0 0, 0 0, 0 0))**

[**ST\_Validate()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-validate) returns the geometry if it is syntactically well-formed and is geometrically valid, **NULL** if the argument is not syntactically well-formed or is not geometrically valid or is **NULL**.

[**ST\_Validate()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-validate) can be used to filter out invalid geometry data, although at a cost. For applications that require more precise results not tainted by invalid data, this penalty may be worthwhile.

If the geometry argument is valid, it is returned as is, except that if an input **Polygon** or **MultiPolygon** has clockwise rings, those rings are reversed before checking for validity. If the geometry is valid, the value with the reversed rings is returned.

The only valid empty geometry is represented in the form of an empty geometry collection value. [**ST\_Validate()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-validate) returns it directly without further checks in this case.

As of MySQL 8.0.13, [**ST\_Validate()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-validate) handles its arguments as described in the introduction to this section, with these exceptions:

If the geometry has a geographic SRS with a longitude or latitude that is out of range, an error occurs:

If a longitude value is not in the range (−180, 180], an [**ER\_GEOMETRY\_PARAM\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_longitude_out_of_range) error occurs ([**ER\_LONGITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_longitude_out_of_range) prior to MySQL 8.0.12).

If a latitude value is not in the range [−90, 90], an [**ER\_GEOMETRY\_PARAM\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_geometry_param_latitude_out_of_range) error occurs ([**ER\_LATITUDE\_OUT\_OF\_RANGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_latitude_out_of_range) prior to MySQL 8.0.12).

Ranges shown are in degrees. The exact range limits deviate slightly due to floating-point arithmetic.

Prior to MySQL 8.0.13, [**ST\_Validate()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_st-validate) handles its arguments as described in the introduction to this section, with these exceptions:

If the geometry is not syntactically well-formed, the return value is **NULL**. An [**ER\_GIS\_INVALID\_DATA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_gis_invalid_data) error does not occur.

If the geometry has an SRID value for a geographic spatial reference system (SRS), an [**ER\_NOT\_IMPLEMENTED\_FOR\_GEOGRAPHIC\_SRS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_implemented_for_geographic_srs) error occurs.

mysql> **SET @ls1 = ST\_GeomFromText('LINESTRING(0 0)');**

mysql> **SET @ls2 = ST\_GeomFromText('LINESTRING(0 0, 1 1)');**

mysql> **SELECT ST\_AsText(ST\_Validate(@ls1));**

+------------------------------+

| ST\_AsText(ST\_Validate(@ls1)) |

+------------------------------+

| NULL |

+------------------------------+

mysql> **SELECT ST\_AsText(ST\_Validate(@ls2));**

+------------------------------+

| ST\_AsText(ST\_Validate(@ls2)) |

+------------------------------+

| LINESTRING(0 0,1 1) |

+------------------------------+

## 12.18 JSON Functions

[12.18.1 JSON Function Reference](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-function-reference)

[12.18.2 Functions That Create JSON Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-creation-functions)

[12.18.3 Functions That Search JSON Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-search-functions)

[12.18.4 Functions That Modify JSON Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-modification-functions)

[12.18.5 Functions That Return JSON Value Attributes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-attribute-functions)

[12.18.6 JSON Table Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-table-functions)

[12.18.7 JSON Schema Validation Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-validation-functions)

[12.18.8 JSON Utility Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-utility-functions)

The functions described in this section perform operations on JSON values. For discussion of the [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) data type and additional examples showing how to use these functions, see [Section 11.5, “The JSON Data Type”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json).

For functions that take a JSON argument, an error occurs if the argument is not a valid JSON value. Arguments parsed as JSON are indicated by ***json\_doc***; arguments indicated by ***val*** are not parsed.

A set of spatial functions for operating on GeoJSON values is also available. See [Section 12.17.11, “Spatial GeoJSON Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-geojson-functions).

### 12.18.1 JSON Function Reference

**Table 12.22 JSON Functions**

| **Name** | **Description** | **Introduced** | **Deprecated** |
| --- | --- | --- | --- |
| [**->**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-column-path) | Return value from JSON column after evaluating path; equivalent to JSON\_EXTRACT(). |  |  |
| [**->>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-inline-path) | Return value from JSON column after evaluating path and unquoting the result; equivalent to JSON\_UNQUOTE(JSON\_EXTRACT()). |  |  |
| [**JSON\_ARRAY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-array) | Create JSON array |  |  |
| [**JSON\_ARRAY\_APPEND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-array-append) | Append data to JSON document |  |  |
| [**JSON\_ARRAY\_INSERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-array-insert) | Insert into JSON array |  |  |
| [**JSON\_CONTAINS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-contains) | Whether JSON document contains specific object at path |  |  |
| [**JSON\_CONTAINS\_PATH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-contains-path) | Whether JSON document contains any data at path |  |  |
| [**JSON\_DEPTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-depth) | Maximum depth of JSON document |  |  |
| [**JSON\_EXTRACT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-extract) | Return data from JSON document |  |  |
| [**JSON\_INSERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-insert) | Insert data into JSON document |  |  |
| [**JSON\_KEYS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-keys) | Array of keys from JSON document |  |  |
| [**JSON\_LENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-length) | Number of elements in JSON document |  |  |
| [**JSON\_MERGE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-merge) | Merge JSON documents, preserving duplicate keys. Deprecated synonym for JSON\_MERGE\_PRESERVE() |  | Yes |
| [**JSON\_MERGE\_PATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-merge-patch) | Merge JSON documents, replacing values of duplicate keys |  |  |
| [**JSON\_MERGE\_PRESERVE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-merge-preserve) | Merge JSON documents, preserving duplicate keys |  |  |
| [**JSON\_OBJECT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-object) | Create JSON object |  |  |
| [**JSON\_OVERLAPS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-overlaps) | Compares two JSON documents, returns TRUE (1) if these have any key-value pairs or array elements in common, otherwise FALSE (0) | 8.0.17 |  |
| [**JSON\_PRETTY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-pretty) | Print a JSON document in human-readable format |  |  |
| [**JSON\_QUOTE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-quote) | Quote JSON document |  |  |
| [**JSON\_REMOVE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-remove) | Remove data from JSON document |  |  |
| [**JSON\_REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-replace) | Replace values in JSON document |  |  |
| [**JSON\_SCHEMA\_VALID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-schema-valid) | Validate JSON document against JSON schema; returns TRUE/1 if document validates against schema, or FALSE/0 if it does not | 8.0.17 |  |
| [**JSON\_SCHEMA\_VALIDATION\_REPORT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-schema-validation-report) | Validate JSON document against JSON schema; returns report in JSON format on outcome on validation including success or failure and reasons for failure | 8.0.17 |  |
| [**JSON\_SEARCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-search) | Path to value within JSON document |  |  |
| [**JSON\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set) | Insert data into JSON document |  |  |
| [**JSON\_STORAGE\_FREE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-storage-free) | Freed space within binary representation of JSON column value following partial update |  |  |
| [**JSON\_STORAGE\_SIZE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-storage-size) | Space used for storage of binary representation of a JSON document |  |  |
| [**JSON\_TABLE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-table) | Return data from a JSON expression as a relational table |  |  |
| [**JSON\_TYPE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-type) | Type of JSON value |  |  |
| [**JSON\_UNQUOTE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-unquote) | Unquote JSON value |  |  |
| [**JSON\_VALID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-valid) | Whether JSON value is valid |  |  |
| [**JSON\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-value) | Extract value from JSON document at location pointed to by path provided; return this value as VARCHAR(512) or specified type | 8.0.21 |  |
| [**MEMBER OF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_member-of) | Returns true (1) if first operand matches any element of JSON array passed as second operand, otherwise returns false (0) | 8.0.17 |  |

MySQL supports two aggregate JSON functions [**JSON\_ARRAYAGG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-arrayagg) and [**JSON\_OBJECTAGG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-objectagg). See [Section 12.20, “Aggregate Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#aggregate-functions-and-modifiers), for descriptions of these.

MySQL also supports “pretty-printing” of JSON values in an easy-to-read format, using the [**JSON\_PRETTY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-pretty) function. You can see how much storage space a given JSON value takes up, and how much space remains for additional storage, using [**JSON\_STORAGE\_SIZE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-storage-size) and [**JSON\_STORAGE\_FREE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-storage-free), respectively. For complete descriptions of these functions, see [Section 12.18.8, “JSON Utility Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-utility-functions).

### 12.18.2 Functions That Create JSON Values

The functions listed in this section compose JSON values from component elements.

**[JSON\_ARRAY([](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array)*[] ...])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array)**

Evaluates a (possibly empty) list of values and returns a JSON array containing those values.

mysql> **SELECT JSON\_ARRAY(1, "abc", NULL, TRUE, CURTIME());**

+---------------------------------------------+

| JSON\_ARRAY(1, "abc", NULL, TRUE, CURTIME()) |

+---------------------------------------------+

| [1, "abc", null, true, "11:30:24.000000"] |

+---------------------------------------------+

**[JSON\_OBJECT([](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-object)*[key](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-object)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-object)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-object)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-object)*[key](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-object)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-object)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-object)*[] ...])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-object)**

Evaluates a (possibly empty) list of key-value pairs and returns a JSON object containing those pairs. An error occurs if any key name is **NULL** or the number of arguments is odd.

mysql> **SELECT JSON\_OBJECT('id', 87, 'name', 'carrot');**

+-----------------------------------------+

| JSON\_OBJECT('id', 87, 'name', 'carrot') |

+-----------------------------------------+

| {"id": 87, "name": "carrot"} |

+-----------------------------------------+

**[JSON\_QUOTE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-quote)*[string](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-quote)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-quote)**

Quotes a string as a JSON value by wrapping it with double quote characters and escaping interior quote and other characters, then returning the result as a **utf8mb4** string. Returns **NULL** if the argument is **NULL**.

This function is typically used to produce a valid JSON string literal for inclusion within a JSON document.

Certain special characters are escaped with backslashes per the escape sequences shown in [Table 12.23, “JSON\_UNQUOTE() Special Character Escape Sequences”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-unquote-character-escape-sequences).

mysql> **SELECT JSON\_QUOTE('null'), JSON\_QUOTE('"null"');**

+--------------------+----------------------+

| JSON\_QUOTE('null') | JSON\_QUOTE('"null"') |

+--------------------+----------------------+

| "null" | "\"null\"" |

+--------------------+----------------------+

mysql> **SELECT JSON\_QUOTE('[1, 2, 3]');**

+-------------------------+

| JSON\_QUOTE('[1, 2, 3]') |

+-------------------------+

| "[1, 2, 3]" |

+-------------------------+

You can also obtain JSON values by casting values of other types to the **JSON** type using [**CAST(*value* AS JSON)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast); see [Converting between JSON and non-JSON values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json-converting-between-types), for more information.

Two aggregate functions generating JSON values are available. [**JSON\_ARRAYAGG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-arrayagg) returns a result set as a single JSON array, and [**JSON\_OBJECTAGG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-objectagg) returns a result set as a single JSON object. For more information, see [Section 12.20, “Aggregate Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#aggregate-functions-and-modifiers).

### 12.18.3 Functions That Search JSON Values

The functions in this section perform search or comparison operations on JSON values to extract data from them, report whether data exists at a location within them, or report the path to data within them. The [**MEMBER OF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_member-of) operator is also documented herein.

**[JSON\_CONTAINS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains)*[target](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains)*[candidate](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains)**

Indicates by returning 1 or 0 whether a given ***candidate*** JSON document is contained within a ***target*** JSON document, or—if a ***path*** argument was supplied—whether the candidate is found at a specific path within the target. Returns **NULL** if any argument is **NULL**, or if the path argument does not identify a section of the target document. An error occurs if ***target*** or ***candidate*** is not a valid JSON document, or if the ***path*** argument is not a valid path expression or contains a **\*** or **\*\*** wildcard.

To check only whether any data exists at the path, use [**JSON\_CONTAINS\_PATH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-contains-path) instead.

The following rules define containment:

A candidate scalar is contained in a target scalar if and only if they are comparable and are equal. Two scalar values are comparable if they have the same [**JSON\_TYPE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-type) types, with the exception that values of types **INTEGER** and **DECIMAL** are also comparable to each other.

A candidate array is contained in a target array if and only if every element in the candidate is contained in some element of the target.

A candidate nonarray is contained in a target array if and only if the candidate is contained in some element of the target.

A candidate object is contained in a target object if and only if for each key in the candidate there is a key with the same name in the target and the value associated with the candidate key is contained in the value associated with the target key.

Otherwise, the candidate value is not contained in the target document.

Starting with MySQL 8.0.17, queries using **JSON\_CONTAINS()** on [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables can be optimized using multi-valued indexes; see [Multi-Valued Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-multi-valued), for more information.

mysql> **SET @j = '{"a": 1, "b": 2, "c": {"d": 4}}';**

mysql> **SET @j2 = '1';**

mysql> **SELECT JSON\_CONTAINS(@j, @j2, '$.a');**

+-------------------------------+

| JSON\_CONTAINS(@j, @j2, '$.a') |

+-------------------------------+

| 1 |

+-------------------------------+

mysql> **SELECT JSON\_CONTAINS(@j, @j2, '$.b');**

+-------------------------------+

| JSON\_CONTAINS(@j, @j2, '$.b') |

+-------------------------------+

| 0 |

+-------------------------------+

mysql> **SET @j2 = '{"d": 4}';**

mysql> **SELECT JSON\_CONTAINS(@j, @j2, '$.a');**

+-------------------------------+

| JSON\_CONTAINS(@j, @j2, '$.a') |

+-------------------------------+

| 0 |

+-------------------------------+

mysql> **SELECT JSON\_CONTAINS(@j, @j2, '$.c');**

+-------------------------------+

| JSON\_CONTAINS(@j, @j2, '$.c') |

+-------------------------------+

| 1 |

+-------------------------------+

**[JSON\_CONTAINS\_PATH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains-path)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains-path)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains-path)*[one\_or\_all](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains-path)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains-path)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains-path)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains-path)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains-path)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-contains-path)**

Returns 0 or 1 to indicate whether a JSON document contains data at a given path or paths. Returns **NULL** if any argument is **NULL**. An error occurs if the ***json\_doc*** argument is not a valid JSON document, any ***path*** argument is not a valid path expression, or ***one\_or\_all*** is not **'one'** or **'all'**.

To check for a specific value at a path, use [**JSON\_CONTAINS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-contains) instead.

The return value is 0 if no specified path exists within the document. Otherwise, the return value depends on the ***one\_or\_all*** argument:

**'one'**: 1 if at least one path exists within the document, 0 otherwise.

**'all'**: 1 if all paths exist within the document, 0 otherwise.

mysql> **SET @j = '{"a": 1, "b": 2, "c": {"d": 4}}';**

mysql> **SELECT JSON\_CONTAINS\_PATH(@j, 'one', '$.a', '$.e');**

+---------------------------------------------+

| JSON\_CONTAINS\_PATH(@j, 'one', '$.a', '$.e') |

+---------------------------------------------+

| 1 |

+---------------------------------------------+

mysql> **SELECT JSON\_CONTAINS\_PATH(@j, 'all', '$.a', '$.e');**

+---------------------------------------------+

| JSON\_CONTAINS\_PATH(@j, 'all', '$.a', '$.e') |

+---------------------------------------------+

| 0 |

+---------------------------------------------+

mysql> **SELECT JSON\_CONTAINS\_PATH(@j, 'one', '$.c.d');**

+----------------------------------------+

| JSON\_CONTAINS\_PATH(@j, 'one', '$.c.d') |

+----------------------------------------+

| 1 |

+----------------------------------------+

mysql> **SELECT JSON\_CONTAINS\_PATH(@j, 'one', '$.a.d');**

+----------------------------------------+

| JSON\_CONTAINS\_PATH(@j, 'one', '$.a.d') |

+----------------------------------------+

| 0 |

+----------------------------------------+

**[JSON\_EXTRACT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-extract)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-extract)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-extract)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-extract)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-extract)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-extract)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-extract)**

Returns data from a JSON document, selected from the parts of the document matched by the ***path*** arguments. Returns **NULL** if any argument is **NULL** or no paths locate a value in the document. An error occurs if the ***json\_doc*** argument is not a valid JSON document or any ***path*** argument is not a valid path expression.

The return value consists of all values matched by the ***path*** arguments. If it is possible that those arguments could return multiple values, the matched values are autowrapped as an array, in the order corresponding to the paths that produced them. Otherwise, the return value is the single matched value.

mysql> **SELECT JSON\_EXTRACT('[10, 20, [30, 40]]', '$[1]');**

+--------------------------------------------+

| JSON\_EXTRACT('[10, 20, [30, 40]]', '$[1]') |

+--------------------------------------------+

| 20 |

+--------------------------------------------+

mysql> **SELECT JSON\_EXTRACT('[10, 20, [30, 40]]', '$[1]', '$[0]');**

+----------------------------------------------------+

| JSON\_EXTRACT('[10, 20, [30, 40]]', '$[1]', '$[0]') |

+----------------------------------------------------+

| [20, 10] |

+----------------------------------------------------+

mysql> **SELECT JSON\_EXTRACT('[10, 20, [30, 40]]', '$[2][\*]');**

+-----------------------------------------------+

| JSON\_EXTRACT('[10, 20, [30, 40]]', '$[2][\*]') |

+-----------------------------------------------+

| [30, 40] |

+-----------------------------------------------+

MySQL supports the [**->**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-column-path) operator as shorthand for this function as used with 2 arguments where the left hand side is a [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) column identifier (not an expression) and the right hand side is the JSON path to be matched within the column.

***[column](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_json-column-path)*[->](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_json-column-path)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_json-column-path)***

The [**->**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-column-path) operator serves as an alias for the [**JSON\_EXTRACT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-extract) function when used with two arguments, a column identifier on the left and a JSON path on the right that is evaluated against the JSON document (the column value). You can use such expressions in place of column identifiers wherever they occur in SQL statements.

The two [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements shown here produce the same output:

mysql> **SELECT c, JSON\_EXTRACT(c, "$.id"), g**

> **FROM jemp**

> **WHERE JSON\_EXTRACT(c, "$.id") > 1**

> **ORDER BY JSON\_EXTRACT(c, "$.name");**

+-------------------------------+-----------+------+

| c | c->"$.id" | g |

+-------------------------------+-----------+------+

| {"id": "3", "name": "Barney"} | "3" | 3 |

| {"id": "4", "name": "Betty"} | "4" | 4 |

| {"id": "2", "name": "Wilma"} | "2" | 2 |

+-------------------------------+-----------+------+

3 rows in set (0.00 sec)

mysql> **SELECT c, c->"$.id", g**

> **FROM jemp**

> **WHERE c->"$.id" > 1**

> **ORDER BY c->"$.name";**

+-------------------------------+-----------+------+

| c | c->"$.id" | g |

+-------------------------------+-----------+------+

| {"id": "3", "name": "Barney"} | "3" | 3 |

| {"id": "4", "name": "Betty"} | "4" | 4 |

| {"id": "2", "name": "Wilma"} | "2" | 2 |

+-------------------------------+-----------+------+

3 rows in set (0.00 sec)

This functionality is not limited to **SELECT**, as shown here:

mysql> **ALTER TABLE jemp ADD COLUMN n INT;**

Query OK, 0 rows affected (0.68 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> **UPDATE jemp SET n=1 WHERE c->"$.id" = "4";**

Query OK, 1 row affected (0.04 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> **SELECT c, c->"$.id", g, n**

> **FROM jemp**

> **WHERE JSON\_EXTRACT(c, "$.id") > 1**

> **ORDER BY c->"$.name";**

+-------------------------------+-----------+------+------+

| c | c->"$.id" | g | n |

+-------------------------------+-----------+------+------+

| {"id": "3", "name": "Barney"} | "3" | 3 | NULL |

| {"id": "4", "name": "Betty"} | "4" | 4 | 1 |

| {"id": "2", "name": "Wilma"} | "2" | 2 | NULL |

+-------------------------------+-----------+------+------+

3 rows in set (0.00 sec)

mysql> **DELETE FROM jemp WHERE c->"$.id" = "4";**

Query OK, 1 row affected (0.04 sec)

mysql> **SELECT c, c->"$.id", g, n**

> **FROM jemp**

> **WHERE JSON\_EXTRACT(c, "$.id") > 1**

> **ORDER BY c->"$.name";**

+-------------------------------+-----------+------+------+

| c | c->"$.id" | g | n |

+-------------------------------+-----------+------+------+

| {"id": "3", "name": "Barney"} | "3" | 3 | NULL |

| {"id": "2", "name": "Wilma"} | "2" | 2 | NULL |

+-------------------------------+-----------+------+------+

2 rows in set (0.00 sec)

(See [Indexing a Generated Column to Provide a JSON Column Index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#json-column-indirect-index), for the statements used to create and populate the table just shown.)

This also works with JSON array values, as shown here:

mysql> **CREATE TABLE tj10 (a JSON, b INT);**

Query OK, 0 rows affected (0.26 sec)

mysql> **INSERT INTO tj10**

> **VALUES ("[3,10,5,17,44]", 33), ("[3,10,5,17,[22,44,66]]", 0);**

Query OK, 1 row affected (0.04 sec)

mysql> **SELECT a->"$[4]" FROM tj10;**

+--------------+

| a->"$[4]" |

+--------------+

| 44 |

| [22, 44, 66] |

+--------------+

2 rows in set (0.00 sec)

mysql> **SELECT \* FROM tj10 WHERE a->"$[0]" = 3;**

+------------------------------+------+

| a | b |

+------------------------------+------+

| [3, 10, 5, 17, 44] | 33 |

| [3, 10, 5, 17, [22, 44, 66]] | 0 |

+------------------------------+------+

2 rows in set (0.00 sec)

Nested arrays are supported. An expression using **->** evaluates as **NULL** if no matching key is found in the target JSON document, as shown here:

mysql> **SELECT \* FROM tj10 WHERE a->"$[4][1]" IS NOT NULL;**

+------------------------------+------+

| a | b |

+------------------------------+------+

| [3, 10, 5, 17, [22, 44, 66]] | 0 |

+------------------------------+------+

mysql> **SELECT a->"$[4][1]" FROM tj10;**

+--------------+

| a->"$[4][1]" |

+--------------+

| NULL |

| 44 |

+--------------+

2 rows in set (0.00 sec)

This is the same behavior as seen in such cases when using **JSON\_EXTRACT()**:

mysql> **SELECT JSON\_EXTRACT(a, "$[4][1]") FROM tj10;**

+----------------------------+

| JSON\_EXTRACT(a, "$[4][1]") |

+----------------------------+

| NULL |

| 44 |

+----------------------------+

2 rows in set (0.00 sec)

***[column](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_json-inline-path)*[->>](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_json-inline-path)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_json-inline-path)***

This is an improved, unquoting extraction operator. Whereas the **->** operator simply extracts a value, the **->>** operator in addition unquotes the extracted result. In other words, given a [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) column value ***column*** and a path expression ***path***, the following three expressions return the same value:

[**JSON\_UNQUOTE(**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-unquote) [**JSON\_EXTRACT(*column*, *path*) )**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-extract)

**JSON\_UNQUOTE(*column*** [**->**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-column-path) ***path*)**

***column*->>*path***

The **->>** operator can be used wherever **JSON\_UNQUOTE(JSON\_EXTRACT())** would be allowed. This includes (but is not limited to) **SELECT** lists, **WHERE** and **HAVING** clauses, and **ORDER BY** and **GROUP BY** clauses.

The next few statements demonstrate some **->>** operator equivalences with other expressions in the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client:

mysql> **SELECT \* FROM jemp WHERE g > 2;**

+-------------------------------+------+

| c | g |

+-------------------------------+------+

| {"id": "3", "name": "Barney"} | 3 |

| {"id": "4", "name": "Betty"} | 4 |

+-------------------------------+------+

2 rows in set (0.01 sec)

mysql> **SELECT c->'$.name' AS name**

-> **FROM jemp WHERE g > 2;**

+----------+

| name |

+----------+

| "Barney" |

| "Betty" |

+----------+

2 rows in set (0.00 sec)

mysql> **SELECT JSON\_UNQUOTE(c->'$.name') AS name**

-> **FROM jemp WHERE g > 2;**

+--------+

| name |

+--------+

| Barney |

| Betty |

+--------+

2 rows in set (0.00 sec)

mysql> **SELECT c->>'$.name' AS name**

-> **FROM jemp WHERE g > 2;**

+--------+

| name |

+--------+

| Barney |

| Betty |

+--------+

2 rows in set (0.00 sec)

See [Indexing a Generated Column to Provide a JSON Column Index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#json-column-indirect-index), for the SQL statements used to create and populate the **jemp** table in the set of examples just shown.

This operator can also be used with JSON arrays, as shown here:

mysql> **CREATE TABLE tj10 (a JSON, b INT);**

Query OK, 0 rows affected (0.26 sec)

mysql> **INSERT INTO tj10 VALUES**

-> **('[3,10,5,"x",44]', 33),**

-> **('[3,10,5,17,[22,"y",66]]', 0);**

Query OK, 2 rows affected (0.04 sec)

Records: 2 Duplicates: 0 Warnings: 0

mysql> **SELECT a->"$[3]", a->"$[4][1]" FROM tj10;**

+-----------+--------------+

| a->"$[3]" | a->"$[4][1]" |

+-----------+--------------+

| "x" | NULL |

| 17 | "y" |

+-----------+--------------+

2 rows in set (0.00 sec)

mysql> **SELECT a->>"$[3]", a->>"$[4][1]" FROM tj10;**

+------------+---------------+

| a->>"$[3]" | a->>"$[4][1]" |

+------------+---------------+

| x | NULL |

| 17 | y |

+------------+---------------+

2 rows in set (0.00 sec)

As with [**->**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-column-path), the **->>** operator is always expanded in the output of [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain), as the following example demonstrates:

mysql> **EXPLAIN SELECT c->>'$.name' AS name**

-> **FROM jemp WHERE g > 2\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: SIMPLE

table: jemp

partitions: NULL

type: range

possible\_keys: i

key: i

key\_len: 5

ref: NULL

rows: 2

filtered: 100.00

Extra: Using where

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Note

Code: 1003

Message: /\* select#1 \*/ select

json\_unquote(json\_extract(`jtest`.`jemp`.`c`,'$.name')) AS `name` from

`jtest`.`jemp` where (`jtest`.`jemp`.`g` > 2)

1 row in set (0.00 sec)

This is similar to how MySQL expands the [**->**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-column-path) operator in the same circumstances.

**[JSON\_KEYS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-keys)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-keys)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-keys)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-keys)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-keys)**

Returns the keys from the top-level value of a JSON object as a JSON array, or, if a ***path*** argument is given, the top-level keys from the selected path. Returns **NULL** if any argument is **NULL**, the ***json\_doc*** argument is not an object, or ***path***, if given, does not locate an object. An error occurs if the ***json\_doc*** argument is not a valid JSON document or the ***path*** argument is not a valid path expression or contains a **\*** or **\*\*** wildcard.

The result array is empty if the selected object is empty. If the top-level value has nested subobjects, the return value does not include keys from those subobjects.

mysql> **SELECT JSON\_KEYS('{"a": 1, "b": {"c": 30}}');**

+---------------------------------------+

| JSON\_KEYS('{"a": 1, "b": {"c": 30}}') |

+---------------------------------------+

| ["a", "b"] |

+---------------------------------------+

mysql> **SELECT JSON\_KEYS('{"a": 1, "b": {"c": 30}}', '$.b');**

+----------------------------------------------+

| JSON\_KEYS('{"a": 1, "b": {"c": 30}}', '$.b') |

+----------------------------------------------+

| ["c"] |

+----------------------------------------------+

**[JSON\_OVERLAPS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-overlaps)*[json\_doc1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-overlaps)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-overlaps)*[json\_doc2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-overlaps)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-overlaps)**

Compares two JSON documents. Returns true (1) if the two document have any key-value pairs or array elements in common. If both arguments are scalars, the function performs a simple equality test.

This function serves as counterpart to [**JSON\_CONTAINS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-contains), which requires all elements of the array searched for to be present in the array searched in. Thus, **JSON\_CONTAINS()** performs an **AND** operation on search keys, while **JSON\_OVERLAPS()** performs an **OR** operation.

Queries on JSON columns of [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables using **JSON\_OVERLAPS()** in the **WHERE** clause can be optimized using multi-valued indexes. [Multi-Valued Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-multi-valued), provides detailed information and examples.

When comparing two arrays, **JSON\_OVERLAPS()** returns true if they share one or more array elements in common, and false if they do not:

mysql> **SELECT JSON\_OVERLAPS("[1,3,5,7]", "[2,5,7]");**

+---------------------------------------+

| JSON\_OVERLAPS("[1,3,5,7]", "[2,5,7]") |

+---------------------------------------+

| 1 |

+---------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT JSON\_OVERLAPS("[1,3,5,7]", "[2,6,7]");**

+---------------------------------------+

| JSON\_OVERLAPS("[1,3,5,7]", "[2,6,7]") |

+---------------------------------------+

| 1 |

+---------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT JSON\_OVERLAPS("[1,3,5,7]", "[2,6,8]");**

+---------------------------------------+

| JSON\_OVERLAPS("[1,3,5,7]", "[2,6,8]") |

+---------------------------------------+

| 0 |

+---------------------------------------+

1 row in set (0.00 sec)

Partial matches are treated as no match, as shown here:

mysql> **SELECT JSON\_OVERLAPS('[[1,2],[3,4],5]', '[1,[2,3],[4,5]]');**

+-----------------------------------------------------+

| JSON\_OVERLAPS('[[1,2],[3,4],5]', '[1,[2,3],[4,5]]') |

+-----------------------------------------------------+

| 0 |

+-----------------------------------------------------+

1 row in set (0.00 sec)

When comparing objects, the result is true if they have at least one key-value pair in common.

mysql> **SELECT JSON\_OVERLAPS('{"a":1,"b":10,"d":10}', '{"c":1,"e":10,"f":1,"d":10}');**

+-----------------------------------------------------------------------+

| JSON\_OVERLAPS('{"a":1,"b":10,"d":10}', '{"c":1,"e":10,"f":1,"d":10}') |

+-----------------------------------------------------------------------+

| 1 |

+-----------------------------------------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT JSON\_OVERLAPS('{"a":1,"b":10,"d":10}', '{"a":5,"e":10,"f":1,"d":20}');**

+-----------------------------------------------------------------------+

| JSON\_OVERLAPS('{"a":1,"b":10,"d":10}', '{"a":5,"e":10,"f":1,"d":20}') |

+-----------------------------------------------------------------------+

| 0 |

+-----------------------------------------------------------------------+

1 row in set (0.00 sec)

If two scalars are used as the arguments to the function, **JSON\_OVERLAPS()** performs a simple test for equality:

mysql> **SELECT JSON\_OVERLAPS('5', '5');**

+-------------------------+

| JSON\_OVERLAPS('5', '5') |

+-------------------------+

| 1 |

+-------------------------+

1 row in set (0.00 sec)

mysql> **SELECT JSON\_OVERLAPS('5', '6');**

+-------------------------+

| JSON\_OVERLAPS('5', '6') |

+-------------------------+

| 0 |

+-------------------------+

1 row in set (0.00 sec)

When comparing a scalar with an array, **JSON\_OVERLAPS()** attempts to treat the scalar as an array element. In this example, the second argument **6** is interpreted as **[6]**, as shown here:

mysql> **SELECT JSON\_OVERLAPS('[4,5,6,7]', '6');**

+---------------------------------+

| JSON\_OVERLAPS('[4,5,6,7]', '6') |

+---------------------------------+

| 1 |

+---------------------------------+

1 row in set (0.00 sec)

The function does not perform type conversions:

mysql> **SELECT JSON\_OVERLAPS('[4,5,"6",7]', '6');**

+-----------------------------------+

| JSON\_OVERLAPS('[4,5,"6",7]', '6') |

+-----------------------------------+

| 0 |

+-----------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT JSON\_OVERLAPS('[4,5,6,7]', '"6"');**

+-----------------------------------+

| JSON\_OVERLAPS('[4,5,6,7]', '"6"') |

+-----------------------------------+

| 0 |

+-----------------------------------+

1 row in set (0.00 sec)

**JSON\_OVERLAPS()** was added in MySQL 8.0.17.

**[JSON\_SEARCH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-search)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-search)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-search)*[one\_or\_all](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-search)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-search)*[search\_str](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-search)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-search)*[escape\_char](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-search)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-search)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-search)*[] ...])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-search)**

Returns the path to the given string within a JSON document. Returns **NULL** if any of the ***json\_doc***, ***search\_str***, or ***path*** arguments are **NULL**; no ***path*** exists within the document; or ***search\_str*** is not found. An error occurs if the ***json\_doc*** argument is not a valid JSON document, any ***path*** argument is not a valid path expression, ***one\_or\_all*** is not **'one'** or **'all'**, or ***escape\_char*** is not a constant expression.

The ***one\_or\_all*** argument affects the search as follows:

**'one'**: The search terminates after the first match and returns one path string. It is undefined which match is considered first.

**'all'**: The search returns all matching path strings such that no duplicate paths are included. If there are multiple strings, they are autowrapped as an array. The order of the array elements is undefined.

Within the ***search\_str*** search string argument, the **%** and **\_** characters work as for the [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) operator: **%** matches any number of characters (including zero characters), and **\_** matches exactly one character.

To specify a literal **%** or **\_** character in the search string, precede it by the escape character. The default is **\** if the ***escape\_char*** argument is missing or **NULL**. Otherwise, ***escape\_char*** must be a constant that is empty or one character.

For more information about matching and escape character behavior, see the description of [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) in [Section 12.8.1, “String Comparison Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#string-comparison-functions). For escape character handling, a difference from the [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) behavior is that the escape character for [**JSON\_SEARCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-search) must evaluate to a constant at compile time, not just at execution time. For example, if [**JSON\_SEARCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-search) is used in a prepared statement and the ***escape\_char*** argument is supplied using a **?** parameter, the parameter value might be constant at execution time, but is not at compile time.

***search\_str*** and ***path*** are always interpeted as utf8mb4 strings, regardless of their actual encoding. This is a known issue which is fixed in MySQL 8.0.24 ( Bug #32449181).

mysql> **SET @j = '["abc", [{"k": "10"}, "def"], {"x":"abc"}, {"y":"bcd"}]';**

mysql> **SELECT JSON\_SEARCH(@j, 'one', 'abc');**

+-------------------------------+

| JSON\_SEARCH(@j, 'one', 'abc') |

+-------------------------------+

| "$[0]" |

+-------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', 'abc');**

+-------------------------------+

| JSON\_SEARCH(@j, 'all', 'abc') |

+-------------------------------+

| ["$[0]", "$[2].x"] |

+-------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', 'ghi');**

+-------------------------------+

| JSON\_SEARCH(@j, 'all', 'ghi') |

+-------------------------------+

| NULL |

+-------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '10');**

+------------------------------+

| JSON\_SEARCH(@j, 'all', '10') |

+------------------------------+

| "$[1][0].k" |

+------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '10', NULL, '$');**

+-----------------------------------------+

| JSON\_SEARCH(@j, 'all', '10', NULL, '$') |

+-----------------------------------------+

| "$[1][0].k" |

+-----------------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '10', NULL, '$[\*]');**

+--------------------------------------------+

| JSON\_SEARCH(@j, 'all', '10', NULL, '$[\*]') |

+--------------------------------------------+

| "$[1][0].k" |

+--------------------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '10', NULL, '$\*\*.k');**

+---------------------------------------------+

| JSON\_SEARCH(@j, 'all', '10', NULL, '$\*\*.k') |

+---------------------------------------------+

| "$[1][0].k" |

+---------------------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '10', NULL, '$[\*][0].k');**

+-------------------------------------------------+

| JSON\_SEARCH(@j, 'all', '10', NULL, '$[\*][0].k') |

+-------------------------------------------------+

| "$[1][0].k" |

+-------------------------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '10', NULL, '$[1]');**

+--------------------------------------------+

| JSON\_SEARCH(@j, 'all', '10', NULL, '$[1]') |

+--------------------------------------------+

| "$[1][0].k" |

+--------------------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '10', NULL, '$[1][0]');**

+-----------------------------------------------+

| JSON\_SEARCH(@j, 'all', '10', NULL, '$[1][0]') |

+-----------------------------------------------+

| "$[1][0].k" |

+-----------------------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', 'abc', NULL, '$[2]');**

+---------------------------------------------+

| JSON\_SEARCH(@j, 'all', 'abc', NULL, '$[2]') |

+---------------------------------------------+

| "$[2].x" |

+---------------------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '%a%');**

+-------------------------------+

| JSON\_SEARCH(@j, 'all', '%a%') |

+-------------------------------+

| ["$[0]", "$[2].x"] |

+-------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '%b%');**

+-------------------------------+

| JSON\_SEARCH(@j, 'all', '%b%') |

+-------------------------------+

| ["$[0]", "$[2].x", "$[3].y"] |

+-------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '%b%', NULL, '$[0]');**

+---------------------------------------------+

| JSON\_SEARCH(@j, 'all', '%b%', NULL, '$[0]') |

+---------------------------------------------+

| "$[0]" |

+---------------------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '%b%', NULL, '$[2]');**

+---------------------------------------------+

| JSON\_SEARCH(@j, 'all', '%b%', NULL, '$[2]') |

+---------------------------------------------+

| "$[2].x" |

+---------------------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '%b%', NULL, '$[1]');**

+---------------------------------------------+

| JSON\_SEARCH(@j, 'all', '%b%', NULL, '$[1]') |

+---------------------------------------------+

| NULL |

+---------------------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '%b%', '', '$[1]');**

+-------------------------------------------+

| JSON\_SEARCH(@j, 'all', '%b%', '', '$[1]') |

+-------------------------------------------+

| NULL |

+-------------------------------------------+

mysql> **SELECT JSON\_SEARCH(@j, 'all', '%b%', '', '$[3]');**

+-------------------------------------------+

| JSON\_SEARCH(@j, 'all', '%b%', '', '$[3]') |

+-------------------------------------------+

| "$[3].y" |

+-------------------------------------------+

For more information about the JSON path syntax supported by MySQL, including rules governing the wildcard operators **\*** and **\*\***, see [JSON Path Syntax](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json-path-syntax).

**[JSON\_VALUE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-value)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-value)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-value)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-value)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-value)**

Extracts a value from a JSON document at the path given in the specified document, and returns the extracted value, optionally converting it to a desired type. The complete syntax is shown here:

JSON\_VALUE(***json\_doc***, ***path*** [RETURNING ***type***] [***on\_empty***] [***on\_error***])

***on\_empty***:

{NULL | ERROR | DEFAULT ***value***} ON EMPTY

***on\_error***:

{NULL | ERROR | DEFAULT ***value***} ON ERROR

***json\_doc*** is a valid JSON document.

***path*** is a JSON path pointing to a location in the document.

***type*** is one of the following data types:

[**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types)

[**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types)

[**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types)

**SIGNED**

**UNSIGNED**

[**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime)

[**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time)

[**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime)

[**YEAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#year) (MySQL 8.0.22 and later)

**YEAR** values of one or two digits are not supported.

[**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char)

[**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json)

The types just listed are the same as the (non-array) types supported by the [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) function.

If not specified by a **RETURNING** clause, the **JSON\_VALUE()** function's return type is [**VARCHAR(512)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char). When no character set is specified for the return type, **JSON\_VALUE()** uses **utf8mb4** with the binary collation, which is case-sensitive; if **utf8mb4** is specified as the character set for the result, the server uses the default collation for this character set, which is not case-sensitive.

When the data at the specified path consists of or resolves to a JSON null literal, the function returns SQL **NULL**.

***on\_empty***, if specified, determines how **JSON\_VALUE()** behaves when no data is found at the path given; this clause takes one of the following values:

**NULL ON EMPTY**: The function returns **NULL**; this is the default **ON EMPTY** behavior.

**DEFAULT *value* ON EMPTY**: the provided ***value*** is returned. The value's type must match that of the return type.

**ERROR ON EMPTY**: The function throws an error.

If used, ***on\_error*** takes one of the following values with the corresponding outcome when an error occurs, as listed here:

**NULL ON ERROR**: **JSON\_VALUE()** returns **NULL**; this is the default behavior if no **ON ERROR** clause is used.

**DEFAULT *value* ON ERROR**: This is the value returned; its value must match that of the return type.

**ERROR ON ERROR**: An error is thrown.

**ON EMPTY**, if used, must precede any **ON ERROR** clause. Specifying them in the wrong order results in a syntax error.

**Error handling.** In general, errors are handled by **JSON\_VALUE()** as follows:

All JSON input (document and path) is checked for validity. If any of it is not valid, an SQL error is thrown without triggering the **ON ERROR** clause.

**ON ERROR** is triggered whenever any of the following events occur:

Attempting to extract an object or an array, such as that resulting from a path that resolves to multiple locations within the JSON document

Conversion errors, such as attempting to convert **'asdf'** to an **UNSIGNED** value

Truncation of values

A conversion error always triggers a warning even if **NULL ON ERROR** or **DEFAULT ... ON ERROR** is specified.

The **ON EMPTY** clause is triggered when the source JSON document (***expr***) contains no data at the specified location (***path***).

**JSON\_VALUE()** was introduced in MySQL 8.0.21.

**Examples.** Two simple examples are shown here:

mysql> **SELECT JSON\_VALUE('{"fname": "Joe", "lname": "Palmer"}', '$.fname');**

+--------------------------------------------------------------+

| JSON\_VALUE('{"fname": "Joe", "lname": "Palmer"}', '$.fname') |

+--------------------------------------------------------------+

| Joe |

+--------------------------------------------------------------+

mysql> **SELECT JSON\_VALUE('{"item": "shoes", "price": "49.95"}', '$.price'**

-> **RETURNING DECIMAL(4,2)) AS price;**

+-------+

| price |

+-------+

| 49.95 |

+-------+

The statement **SELECT JSON\_VALUE(*json\_doc*, *path* RETURNING *type*)** is equivalent to the following statement:

SELECT CAST(

JSON\_UNQUOTE( JSON\_EXTRACT(***json\_doc***, ***path***) )

AS ***type***

);

**JSON\_VALUE()** simplifies creating indexes on JSON columns by making it unnecessary in many cases to create a generated column and then an index on the generated column. You can do this when creating a table **t1** that has a [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) column by creating an index on an expression that uses **JSON\_VALUE()** operating on that column (with a path that matches a value in that column), as shown here:

CREATE TABLE t1(

j JSON,

INDEX i1 ( (JSON\_VALUE(j, '$.id' RETURNING UNSIGNED)) )

);

The following [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) output shows that a query against **t1** employing the index expression in the **WHERE** clause uses the index thus created:

mysql> EXPLAIN SELECT \* FROM t1

-> WHERE JSON\_VALUE(j, '$.id' RETURNING UNSIGNED) = 123\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: SIMPLE

table: t1

partitions: NULL

type: ref

possible\_keys: i1

key: i1

key\_len: 9

ref: const

rows: 1

filtered: 100.00

Extra: NULL

This achieves much the same effect as creating a table **t2** with an index on a generated column (see [Indexing a Generated Column to Provide a JSON Column Index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#json-column-indirect-index)), like this one:

CREATE TABLE t2 (

j JSON,

g INT GENERATED ALWAYS AS (j->"$.id"),

INDEX i1 (j)

);

The [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) output for a query against this table, referencing the generated column, shows that the index is used in the same way as for the previous query against table **t1**:

mysql> EXPLAIN SELECT \* FROM t2 WHERE g = 123\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: SIMPLE

table: t2

partitions: NULL

type: ref

possible\_keys: i1

key: i1

key\_len: 5

ref: const

rows: 1

filtered: 100.00

Extra: NULL

For information about using indexes on generated columns for indirect indexing of [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) columns, see [Indexing a Generated Column to Provide a JSON Column Index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#json-column-indirect-index).

***[value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_member-of)*[MEMBER OF(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_member-of)*[json\_array](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_member-of)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "operator_member-of)**

Returns true (1) if ***value*** is an element of ***json\_array***, otherwise returns false (0). ***value*** must be a scalar or a JSON document; if it is a scalar, the operator attempts to treat it as an element of a JSON array.

Queries using **MEMBER OF()** on JSON columns of [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables in the **WHERE** clause can be optimized using multi-valued indexes. See [Multi-Valued Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-multi-valued), for detailed information and examples.

Simple scalars are treated as array values, as shown here:

mysql> **SELECT 17 MEMBER OF('[23, "abc", 17, "ab", 10]');**

+-------------------------------------------+

| 17 MEMBER OF('[23, "abc", 17, "ab", 10]') |

+-------------------------------------------+

| 1 |

+-------------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT 'ab' MEMBER OF('[23, "abc", 17, "ab", 10]');**

+---------------------------------------------+

| 'ab' MEMBER OF('[23, "abc", 17, "ab", 10]') |

+---------------------------------------------+

| 1 |

+---------------------------------------------+

1 row in set (0.00 sec)

Partial matches of array element values do not match:

mysql> **SELECT 7 MEMBER OF('[23, "abc", 17, "ab", 10]');**

+------------------------------------------+

| 7 MEMBER OF('[23, "abc", 17, "ab", 10]') |

+------------------------------------------+

| 0 |

+------------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT 'a' MEMBER OF('[23, "abc", 17, "ab", 10]');**

+--------------------------------------------+

| 'a' MEMBER OF('[23, "abc", 17, "ab", 10]') |

+--------------------------------------------+

| 0 |

+--------------------------------------------+

1 row in set (0.00 sec)

Conversions to and from string types are not performed:

mysql> **SELECT**

-> **17 MEMBER OF('[23, "abc", "17", "ab", 10]'),**

-> **"17" MEMBER OF('[23, "abc", 17, "ab", 10]')\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

17 MEMBER OF('[23, "abc", "17", "ab", 10]'): 0

"17" MEMBER OF('[23, "abc", 17, "ab", 10]'): 0

1 row in set (0.00 sec)

To use this operator with a value which itself an array, it is necessary to cast it explicitly as a JSON array. You can do this with [**CAST(... AS JSON)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast):

mysql> **SELECT CAST('[4,5]' AS JSON) MEMBER OF('[[3,4],[4,5]]');**

+--------------------------------------------------+

| CAST('[4,5]' AS JSON) MEMBER OF('[[3,4],[4,5]]') |

+--------------------------------------------------+

| 1 |

+--------------------------------------------------+

1 row in set (0.00 sec)

It is also possible to perform the necessary cast using the [**JSON\_ARRAY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-array) function, like this:

mysql> **SELECT JSON\_ARRAY(4,5) MEMBER OF('[[3,4],[4,5]]');**

+--------------------------------------------+

| JSON\_ARRAY(4,5) MEMBER OF('[[3,4],[4,5]]') |

+--------------------------------------------+

| 1 |

+--------------------------------------------+

1 row in set (0.00 sec)

Any JSON objects used as values to be tested or which appear in the target array must be coerced to the correct type using **CAST(... AS JSON)** or [**JSON\_OBJECT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-object). In addition, a target array containing JSON objects must itself be cast using **JSON\_ARRAY**. This is demonstrated in the following sequence of statements:

mysql> **SET @a = CAST('{"a":1}' AS JSON);**

Query OK, 0 rows affected (0.00 sec)

mysql> **SET @b = JSON\_OBJECT("b", 2);**

Query OK, 0 rows affected (0.00 sec)

mysql> **SET @c = JSON\_ARRAY(17, @b, "abc", @a, 23);**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT @a MEMBER OF(@c), @b MEMBER OF(@c);**

+------------------+------------------+

| @a MEMBER OF(@c) | @b MEMBER OF(@c) |

+------------------+------------------+

| 1 | 1 |

+------------------+------------------+

1 row in set (0.00 sec)

The **MEMBER OF()** operator was added in MySQL 8.0.17.

### 12.18.4 Functions That Modify JSON Values

The functions in this section modify JSON values and return the result.

**[JSON\_ARRAY\_APPEND(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-append)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-append)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-append)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-append)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-append)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-append)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-append)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-append)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-append)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-append)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-append)**

Appends values to the end of the indicated arrays within a JSON document and returns the result. Returns **NULL** if any argument is **NULL**. An error occurs if the ***json\_doc*** argument is not a valid JSON document or any ***path*** argument is not a valid path expression or contains a **\*** or **\*\*** wildcard.

The path-value pairs are evaluated left to right. The document produced by evaluating one pair becomes the new value against which the next pair is evaluated.

If a path selects a scalar or object value, that value is autowrapped within an array and the new value is added to that array. Pairs for which the path does not identify any value in the JSON document are ignored.

mysql> **SET @j = '["a", ["b", "c"], "d"]';**

mysql> **SELECT JSON\_ARRAY\_APPEND(@j, '$[1]', 1);**

+----------------------------------+

| JSON\_ARRAY\_APPEND(@j, '$[1]', 1) |

+----------------------------------+

| ["a", ["b", "c", 1], "d"] |

+----------------------------------+

mysql> **SELECT JSON\_ARRAY\_APPEND(@j, '$[0]', 2);**

+----------------------------------+

| JSON\_ARRAY\_APPEND(@j, '$[0]', 2) |

+----------------------------------+

| [["a", 2], ["b", "c"], "d"] |

+----------------------------------+

mysql> **SELECT JSON\_ARRAY\_APPEND(@j, '$[1][0]', 3);**

+-------------------------------------+

| JSON\_ARRAY\_APPEND(@j, '$[1][0]', 3) |

+-------------------------------------+

| ["a", [["b", 3], "c"], "d"] |

+-------------------------------------+

mysql> **SET @j = '{"a": 1, "b": [2, 3], "c": 4}';**

mysql> **SELECT JSON\_ARRAY\_APPEND(@j, '$.b', 'x');**

+------------------------------------+

| JSON\_ARRAY\_APPEND(@j, '$.b', 'x') |

+------------------------------------+

| {"a": 1, "b": [2, 3, "x"], "c": 4} |

+------------------------------------+

mysql> **SELECT JSON\_ARRAY\_APPEND(@j, '$.c', 'y');**

+--------------------------------------+

| JSON\_ARRAY\_APPEND(@j, '$.c', 'y') |

+--------------------------------------+

| {"a": 1, "b": [2, 3], "c": [4, "y"]} |

+--------------------------------------+

mysql> **SET @j = '{"a": 1}';**

mysql> **SELECT JSON\_ARRAY\_APPEND(@j, '$', 'z');**

+---------------------------------+

| JSON\_ARRAY\_APPEND(@j, '$', 'z') |

+---------------------------------+

| [{"a": 1}, "z"] |

+---------------------------------+

In MySQL 5.7, this function was named **JSON\_APPEND()**. That name is no longer supported in MySQL 8.0.

**[JSON\_ARRAY\_INSERT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-insert)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-insert)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-insert)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-insert)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-insert)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-insert)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-insert)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-insert)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-insert)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-insert)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-array-insert)**

Updates a JSON document, inserting into an array within the document and returning the modified document. Returns **NULL** if any argument is **NULL**. An error occurs if the ***json\_doc*** argument is not a valid JSON document or any ***path*** argument is not a valid path expression or contains a **\*** or **\*\*** wildcard or does not end with an array element identifier.

The path-value pairs are evaluated left to right. The document produced by evaluating one pair becomes the new value against which the next pair is evaluated.

Pairs for which the path does not identify any array in the JSON document are ignored. If a path identifies an array element, the corresponding value is inserted at that element position, shifting any following values to the right. If a path identifies an array position past the end of an array, the value is inserted at the end of the array.

mysql> **SET @j = '["a", {"b": [1, 2]}, [3, 4]]';**

mysql> **SELECT JSON\_ARRAY\_INSERT(@j, '$[1]', 'x');**

+------------------------------------+

| JSON\_ARRAY\_INSERT(@j, '$[1]', 'x') |

+------------------------------------+

| ["a", "x", {"b": [1, 2]}, [3, 4]] |

+------------------------------------+

mysql> **SELECT JSON\_ARRAY\_INSERT(@j, '$[100]', 'x');**

+--------------------------------------+

| JSON\_ARRAY\_INSERT(@j, '$[100]', 'x') |

+--------------------------------------+

| ["a", {"b": [1, 2]}, [3, 4], "x"] |

+--------------------------------------+

mysql> **SELECT JSON\_ARRAY\_INSERT(@j, '$[1].b[0]', 'x');**

+-----------------------------------------+

| JSON\_ARRAY\_INSERT(@j, '$[1].b[0]', 'x') |

+-----------------------------------------+

| ["a", {"b": ["x", 1, 2]}, [3, 4]] |

+-----------------------------------------+

mysql> **SELECT JSON\_ARRAY\_INSERT(@j, '$[2][1]', 'y');**

+---------------------------------------+

| JSON\_ARRAY\_INSERT(@j, '$[2][1]', 'y') |

+---------------------------------------+

| ["a", {"b": [1, 2]}, [3, "y", 4]] |

+---------------------------------------+

mysql> **SELECT JSON\_ARRAY\_INSERT(@j, '$[0]', 'x', '$[2][1]', 'y');**

+----------------------------------------------------+

| JSON\_ARRAY\_INSERT(@j, '$[0]', 'x', '$[2][1]', 'y') |

+----------------------------------------------------+

| ["x", "a", {"b": [1, 2]}, [3, 4]] |

+----------------------------------------------------+

Earlier modifications affect the positions of the following elements in the array, so subsequent paths in the same [**JSON\_ARRAY\_INSERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-array-insert) call should take this into account. In the final example, the second path inserts nothing because the path no longer matches anything after the first insert.

**[JSON\_INSERT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-insert)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-insert)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-insert)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-insert)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-insert)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-insert)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-insert)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-insert)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-insert)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-insert)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-insert)**

Inserts data into a JSON document and returns the result. Returns **NULL** if any argument is **NULL**. An error occurs if the ***json\_doc*** argument is not a valid JSON document or any ***path*** argument is not a valid path expression or contains a **\*** or **\*\*** wildcard.

The path-value pairs are evaluated left to right. The document produced by evaluating one pair becomes the new value against which the next pair is evaluated.

A path-value pair for an existing path in the document is ignored and does not overwrite the existing document value. A path-value pair for a nonexisting path in the document adds the value to the document if the path identifies one of these types of values:

A member not present in an existing object. The member is added to the object and associated with the new value.

A position past the end of an existing array. The array is extended with the new value. If the existing value is not an array, it is autowrapped as an array, then extended with the new value.

Otherwise, a path-value pair for a nonexisting path in the document is ignored and has no effect.

For a comparison of [**JSON\_INSERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-insert), [**JSON\_REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-replace), and [**JSON\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set), see the discussion of [**JSON\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set).

mysql> **SET @j = '{ "a": 1, "b": [2, 3]}';**

mysql> **SELECT JSON\_INSERT(@j, '$.a', 10, '$.c', '[true, false]');**

+----------------------------------------------------+

| JSON\_INSERT(@j, '$.a', 10, '$.c', '[true, false]') |

+----------------------------------------------------+

| {"a": 1, "b": [2, 3], "c": "[true, false]"} |

+----------------------------------------------------+

The third and final value listed in the result is a quoted string and not an array like the second one (which is not quoted in the output); no casting of values to the JSON type is performed. To insert the array as an array, you must perform such casts explicitly, as shown here:

mysql> **SELECT JSON\_INSERT(@j, '$.a', 10, '$.c', CAST('[true, false]' AS JSON));**

+------------------------------------------------------------------+

| JSON\_INSERT(@j, '$.a', 10, '$.c', CAST('[true, false]' AS JSON)) |

+------------------------------------------------------------------+

| {"a": 1, "b": [2, 3], "c": [true, false]} |

+------------------------------------------------------------------+

1 row in set (0.00 sec)

**[JSON\_MERGE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge)**

Merges two or more JSON documents. Synonym for **JSON\_MERGE\_PRESERVE()**; deprecated in MySQL 8.0.3 and subject to removal in a future release.

mysql> **SELECT JSON\_MERGE('[1, 2]', '[true, false]');**

+---------------------------------------+

| JSON\_MERGE('[1, 2]', '[true, false]') |

+---------------------------------------+

| [1, 2, true, false] |

+---------------------------------------+

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Warning

Code: 1287

Message: 'JSON\_MERGE' is deprecated and will be removed in a future release. \

Please use JSON\_MERGE\_PRESERVE/JSON\_MERGE\_PATCH instead

1 row in set (0.00 sec)

For additional examples, see the entry for [**JSON\_MERGE\_PRESERVE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-merge-preserve).

**[JSON\_MERGE\_PATCH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-patch)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-patch)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-patch)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-patch)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-patch)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-patch)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-patch)**

Performs an [RFC 7396](https://tools.ietf.org/html/rfc7396) compliant merge of two or more JSON documents and returns the merged result, without preserving members having duplicate keys. Raises an error if at least one of the documents passed as arguments to this function is not valid.

**Note**

For an explanation and example of the differences between this function and **JSON\_MERGE\_PRESERVE()**, see [JSON\_MERGE\_PATCH() compared with JSON\_MERGE\_PRESERVE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-merge-patch-json-merge-preserve-compared).

**JSON\_MERGE\_PATCH()** performs a merge as follows:

If the first argument is not an object, the result of the merge is the same as if an empty object had been merged with the second argument.

If the second argument is not an object, the result of the merge is the second argument.

If both arguments are objects, the result of the merge is an object with the following members:

All members of the first object which do not have a corresponding member with the same key in the second object.

All members of the second object which do not have a corresponding key in the first object, and whose value is not the JSON **null** literal.

All members with a key that exists in both the first and the second object, and whose value in the second object is not the JSON **null** literal. The values of these members are the results of recursively merging the value in the first object with the value in the second object.

For additional information, see [Normalization, Merging, and Autowrapping of JSON Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json-normalization).

mysql> **SELECT JSON\_MERGE\_PATCH('[1, 2]', '[true, false]');**

+---------------------------------------------+

| JSON\_MERGE\_PATCH('[1, 2]', '[true, false]') |

+---------------------------------------------+

| [true, false] |

+---------------------------------------------+

mysql> **SELECT JSON\_MERGE\_PATCH('{"name": "x"}', '{"id": 47}');**

+-------------------------------------------------+

| JSON\_MERGE\_PATCH('{"name": "x"}', '{"id": 47}') |

+-------------------------------------------------+

| {"id": 47, "name": "x"} |

+-------------------------------------------------+

mysql> **SELECT JSON\_MERGE\_PATCH('1', 'true');**

+-------------------------------+

| JSON\_MERGE\_PATCH('1', 'true') |

+-------------------------------+

| true |

+-------------------------------+

mysql> **SELECT JSON\_MERGE\_PATCH('[1, 2]', '{"id": 47}');**

+------------------------------------------+

| JSON\_MERGE\_PATCH('[1, 2]', '{"id": 47}') |

+------------------------------------------+

| {"id": 47} |

+------------------------------------------+

mysql> **SELECT JSON\_MERGE\_PATCH('{ "a": 1, "b":2 }',**

> **'{ "a": 3, "c":4 }');**

+-----------------------------------------------------------+

| JSON\_MERGE\_PATCH('{ "a": 1, "b":2 }','{ "a": 3, "c":4 }') |

+-----------------------------------------------------------+

| {"a": 3, "b": 2, "c": 4} |

+-----------------------------------------------------------+

mysql> **SELECT JSON\_MERGE\_PATCH('{ "a": 1, "b":2 }','{ "a": 3, "c":4 }',**

> **'{ "a": 5, "d":6 }');**

+-------------------------------------------------------------------------------+

| JSON\_MERGE\_PATCH('{ "a": 1, "b":2 }','{ "a": 3, "c":4 }','{ "a": 5, "d":6 }') |

+-------------------------------------------------------------------------------+

| {"a": 5, "b": 2, "c": 4, "d": 6} |

+-------------------------------------------------------------------------------+

You can use this function to remove a member by specifying **null** as the value of the same member in the seond argument, as shown here:

mysql> **SELECT JSON\_MERGE\_PATCH('{"a":1, "b":2}', '{"b":null}');**

+--------------------------------------------------+

| JSON\_MERGE\_PATCH('{"a":1, "b":2}', '{"b":null}') |

+--------------------------------------------------+

| {"a": 1} |

+--------------------------------------------------+

This example shows that the function operates in a recursive fashion; that is, values of members are not limited to scalars, but rather can themselves be JSON documents:

mysql> **SELECT JSON\_MERGE\_PATCH('{"a":{"x":1}}', '{"a":{"y":2}}');**

+----------------------------------------------------+

| JSON\_MERGE\_PATCH('{"a":{"x":1}}', '{"a":{"y":2}}') |

+----------------------------------------------------+

| {"a": {"x": 1, "y": 2}} |

+----------------------------------------------------+

**JSON\_MERGE\_PATCH()** is supported in MySQL 8.0.3 and later.

**JSON\_MERGE\_PATCH() compared with JSON\_MERGE\_PRESERVE().** The behavior of **JSON\_MERGE\_PATCH()** is the same as that of [**JSON\_MERGE\_PRESERVE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-merge-preserve), with the following two exceptions:

**JSON\_MERGE\_PATCH()** removes any member in the first object with a matching key in the second object, provided that the value associated with the key in the second object is not JSON **null**.

If the second object has a member with a key matching a member in the first object, **JSON\_MERGE\_PATCH()** replaces the value in the first object with the value in the second object, whereas **JSON\_MERGE\_PRESERVE()** appends the second value to the first value.

This example compares the results of merging the same 3 JSON objects, each having a matching key **"a"**, with each of these two functions:

mysql> **SET @x = '{ "a": 1, "b": 2 }',**

> **@y = '{ "a": 3, "c": 4 }',**

> **@z = '{ "a": 5, "d": 6 }';**

mysql> **SELECT JSON\_MERGE\_PATCH(@x, @y, @z) AS Patch,**

-> **JSON\_MERGE\_PRESERVE(@x, @y, @z) AS Preserve\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Patch: {"a": 5, "b": 2, "c": 4, "d": 6}

Preserve: {"a": [1, 3, 5], "b": 2, "c": 4, "d": 6}

**[JSON\_MERGE\_PRESERVE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-preserve)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-preserve)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-preserve)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-preserve)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-preserve)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-preserve)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-merge-preserve)**

Merges two or more JSON documents and returns the merged result. Returns **NULL** if any argument is **NULL**. An error occurs if any argument is not a valid JSON document.

Merging takes place according to the following rules. For additional information, see [Normalization, Merging, and Autowrapping of JSON Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json-normalization).

Adjacent arrays are merged to a single array.

Adjacent objects are merged to a single object.

A scalar value is autowrapped as an array and merged as an array.

An adjacent array and object are merged by autowrapping the object as an array and merging the two arrays.

mysql> **SELECT JSON\_MERGE\_PRESERVE('[1, 2]', '[true, false]');**

+------------------------------------------------+

| JSON\_MERGE\_PRESERVE('[1, 2]', '[true, false]') |

+------------------------------------------------+

| [1, 2, true, false] |

+------------------------------------------------+

mysql> **SELECT JSON\_MERGE\_PRESERVE('{"name": "x"}', '{"id": 47}');**

+----------------------------------------------------+

| JSON\_MERGE\_PRESERVE('{"name": "x"}', '{"id": 47}') |

+----------------------------------------------------+

| {"id": 47, "name": "x"} |

+----------------------------------------------------+

mysql> **SELECT JSON\_MERGE\_PRESERVE('1', 'true');**

+----------------------------------+

| JSON\_MERGE\_PRESERVE('1', 'true') |

+----------------------------------+

| [1, true] |

+----------------------------------+

mysql> **SELECT JSON\_MERGE\_PRESERVE('[1, 2]', '{"id": 47}');**

+---------------------------------------------+

| JSON\_MERGE\_PRESERVE('[1, 2]', '{"id": 47}') |

+---------------------------------------------+

| [1, 2, {"id": 47}] |

+---------------------------------------------+

mysql> **SELECT JSON\_MERGE\_PRESERVE('{ "a": 1, "b": 2 }',**

> **'{ "a": 3, "c": 4 }');**

+--------------------------------------------------------------+

| JSON\_MERGE\_PRESERVE('{ "a": 1, "b": 2 }','{ "a": 3, "c":4 }') |

+--------------------------------------------------------------+

| {"a": [1, 3], "b": 2, "c": 4} |

+--------------------------------------------------------------+

mysql> **SELECT JSON\_MERGE\_PRESERVE('{ "a": 1, "b": 2 }','{ "a": 3, "c": 4 }',**

> **'{ "a": 5, "d": 6 }');**

+----------------------------------------------------------------------------------+

| JSON\_MERGE\_PRESERVE('{ "a": 1, "b": 2 }','{ "a": 3, "c": 4 }','{ "a": 5, "d": 6 }') |

+----------------------------------------------------------------------------------+

| {"a": [1, 3, 5], "b": 2, "c": 4, "d": 6} |

+----------------------------------------------------------------------------------+

This function was added in MySQL 8.0.3 as a synonym for [**JSON\_MERGE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-merge). The **JSON\_MERGE()** function is now deprecated, and is subject to removal in a future release of MySQL.

This function is similar to but differs from [**JSON\_MERGE\_PATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-merge-patch) in significant respects; see [JSON\_MERGE\_PATCH() compared with JSON\_MERGE\_PRESERVE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-merge-patch-json-merge-preserve-compared), for more information.

**[JSON\_REMOVE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-remove)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-remove)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-remove)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-remove)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-remove)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-remove)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-remove)**

Removes data from a JSON document and returns the result. Returns **NULL** if any argument is **NULL**. An error occurs if the ***json\_doc*** argument is not a valid JSON document or any ***path*** argument is not a valid path expression or is **$** or contains a **\*** or **\*\*** wildcard.

The ***path*** arguments are evaluated left to right. The document produced by evaluating one path becomes the new value against which the next path is evaluated.

It is not an error if the element to be removed does not exist in the document; in that case, the path does not affect the document.

mysql> **SET @j = '["a", ["b", "c"], "d"]';**

mysql> **SELECT JSON\_REMOVE(@j, '$[1]');**

+-------------------------+

| JSON\_REMOVE(@j, '$[1]') |

+-------------------------+

| ["a", "d"] |

+-------------------------+

**[JSON\_REPLACE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-replace)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-replace)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-replace)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-replace)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-replace)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-replace)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-replace)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-replace)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-replace)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-replace)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-replace)**

Replaces existing values in a JSON document and returns the result. Returns **NULL** if any argument is **NULL**. An error occurs if the ***json\_doc*** argument is not a valid JSON document or any ***path*** argument is not a valid path expression or contains a **\*** or **\*\*** wildcard.

The path-value pairs are evaluated left to right. The document produced by evaluating one pair becomes the new value against which the next pair is evaluated.

A path-value pair for an existing path in the document overwrites the existing document value with the new value. A path-value pair for a nonexisting path in the document is ignored and has no effect.

In MySQL 8.0.4, the optimizer can perform a partial, in-place update of a **JSON** column instead of removing the old document and writing the new document in its entirety to the column. This optimization can be performed for an update statement that uses the [**JSON\_REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-replace) function and meets the conditions outlined in [Partial Updates of JSON Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json-partial-updates).

For a comparison of [**JSON\_INSERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-insert), [**JSON\_REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-replace), and [**JSON\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set), see the discussion of [**JSON\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set).

mysql> **SET @j = '{ "a": 1, "b": [2, 3]}';**

mysql> **SELECT JSON\_REPLACE(@j, '$.a', 10, '$.c', '[true, false]');**

+-----------------------------------------------------+

| JSON\_REPLACE(@j, '$.a', 10, '$.c', '[true, false]') |

+-----------------------------------------------------+

| {"a": 10, "b": [2, 3]} |

+-----------------------------------------------------+

**[JSON\_SET(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-set)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-set)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-set)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-set)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-set)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-set)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-set)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-set)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-set)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-set)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-set)**

Inserts or updates data in a JSON document and returns the result. Returns **NULL** if any argument is **NULL** or ***path***, if given, does not locate an object. An error occurs if the ***json\_doc*** argument is not a valid JSON document or any ***path*** argument is not a valid path expression or contains a **\*** or **\*\*** wildcard.

The path-value pairs are evaluated left to right. The document produced by evaluating one pair becomes the new value against which the next pair is evaluated.

A path-value pair for an existing path in the document overwrites the existing document value with the new value. A path-value pair for a nonexisting path in the document adds the value to the document if the path identifies one of these types of values:

A member not present in an existing object. The member is added to the object and associated with the new value.

A position past the end of an existing array. The array is extended with the new value. If the existing value is not an array, it is autowrapped as an array, then extended with the new value.

Otherwise, a path-value pair for a nonexisting path in the document is ignored and has no effect.

In MySQL 8.0.4, the optimizer can perform a partial, in-place update of a **JSON** column instead of removing the old document and writing the new document in its entirety to the column. This optimization can be performed for an update statement that uses the [**JSON\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set) function and meets the conditions outlined in [Partial Updates of JSON Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json-partial-updates).

The [**JSON\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set), [**JSON\_INSERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-insert), and [**JSON\_REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-replace) functions are related:

[**JSON\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set) replaces existing values and adds nonexisting values.

[**JSON\_INSERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-insert) inserts values without replacing existing values.

[**JSON\_REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-replace) replaces only existing values.

The following examples illustrate these differences, using one path that does exist in the document (**$.a**) and another that does not exist (**$.c**):

mysql> **SET @j = '{ "a": 1, "b": [2, 3]}';**

mysql> **SELECT JSON\_SET(@j, '$.a', 10, '$.c', '[true, false]');**

+-------------------------------------------------+

| JSON\_SET(@j, '$.a', 10, '$.c', '[true, false]') |

+-------------------------------------------------+

| {"a": 10, "b": [2, 3], "c": "[true, false]"} |

+-------------------------------------------------+

mysql> **SELECT JSON\_INSERT(@j, '$.a', 10, '$.c', '[true, false]');**

+----------------------------------------------------+

| JSON\_INSERT(@j, '$.a', 10, '$.c', '[true, false]') |

+----------------------------------------------------+

| {"a": 1, "b": [2, 3], "c": "[true, false]"} |

+----------------------------------------------------+

mysql> **SELECT JSON\_REPLACE(@j, '$.a', 10, '$.c', '[true, false]');**

+-----------------------------------------------------+

| JSON\_REPLACE(@j, '$.a', 10, '$.c', '[true, false]') |

+-----------------------------------------------------+

| {"a": 10, "b": [2, 3]} |

+-----------------------------------------------------+

**[JSON\_UNQUOTE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-unquote)*[json\_val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-unquote)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-unquote)**

Unquotes JSON value and returns the result as a **utf8mb4** string. Returns **NULL** if the argument is **NULL**. An error occurs if the value starts and ends with double quotes but is not a valid JSON string literal.

Within a string, certain sequences have special meaning unless the [**NO\_BACKSLASH\_ESCAPES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_backslash_escapes) SQL mode is enabled. Each of these sequences begins with a backslash (**\**), known as the escape character. MySQL recognizes the escape sequences shown in [Table 12.23, “JSON\_UNQUOTE() Special Character Escape Sequences”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#json-unquote-character-escape-sequences). For all other escape sequences, backslash is ignored. That is, the escaped character is interpreted as if it was not escaped. For example, **\x** is just **x**. These sequences are case-sensitive. For example, **\b** is interpreted as a backspace, but **\B** is interpreted as **B**.

**Table 12.23 JSON\_UNQUOTE() Special Character Escape Sequences**

| **Escape Sequence** | **Character Represented by Sequence** |
| --- | --- |
| **\"** | A double quote (**"**) character |
| **\b** | A backspace character |
| **\f** | A formfeed character |
| **\n** | A newline (linefeed) character |
| **\r** | A carriage return character |
| **\t** | A tab character |
| **\\** | A backslash (**\**) character |
| **\u*XXXX*** | UTF-8 bytes for Unicode value ***XXXX*** |

Two simple examples of the use of this function are shown here:

mysql> **SET @j = '"abc"';**

mysql> **SELECT @j, JSON\_UNQUOTE(@j);**

+-------+------------------+

| @j | JSON\_UNQUOTE(@j) |

+-------+------------------+

| "abc" | abc |

+-------+------------------+

mysql> **SET @j = '[1, 2, 3]';**

mysql> **SELECT @j, JSON\_UNQUOTE(@j);**

+-----------+------------------+

| @j | JSON\_UNQUOTE(@j) |

+-----------+------------------+

| [1, 2, 3] | [1, 2, 3] |

+-----------+------------------+

The following set of examples shows how **JSON\_UNQUOTE** handles escapes with [**NO\_BACKSLASH\_ESCAPES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_backslash_escapes) disabled and enabled:

mysql> **SELECT @@sql\_mode;**

+------------+

| @@sql\_mode |

+------------+

| |

+------------+

mysql> **SELECT JSON\_UNQUOTE('"\\t\\u0032"');**

+------------------------------+

| JSON\_UNQUOTE('"\\t\\u0032"') |

+------------------------------+

| 2 |

+------------------------------+

mysql> **SET @@sql\_mode = 'NO\_BACKSLASH\_ESCAPES';**

mysql> **SELECT JSON\_UNQUOTE('"\\t\\u0032"');**

+------------------------------+

| JSON\_UNQUOTE('"\\t\\u0032"') |

+------------------------------+

| \t\u0032 |

+------------------------------+

mysql> **SELECT JSON\_UNQUOTE('"\t\u0032"');**

+----------------------------+

| JSON\_UNQUOTE('"\t\u0032"') |

+----------------------------+

| 2 |

+----------------------------+

### 12.18.5 Functions That Return JSON Value Attributes

The functions in this section return attributes of JSON values.

**[JSON\_DEPTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-depth)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-depth)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-depth)**

Returns the maximum depth of a JSON document. Returns **NULL** if the argument is **NULL**. An error occurs if the argument is not a valid JSON document.

An empty array, empty object, or scalar value has depth 1. A nonempty array containing only elements of depth 1 or nonempty object containing only member values of depth 1 has depth 2. Otherwise, a JSON document has depth greater than 2.

mysql> **SELECT JSON\_DEPTH('{}'), JSON\_DEPTH('[]'), JSON\_DEPTH('true');**

+------------------+------------------+--------------------+

| JSON\_DEPTH('{}') | JSON\_DEPTH('[]') | JSON\_DEPTH('true') |

+------------------+------------------+--------------------+

| 1 | 1 | 1 |

+------------------+------------------+--------------------+

mysql> **SELECT JSON\_DEPTH('[10, 20]'), JSON\_DEPTH('[[], {}]');**

+------------------------+------------------------+

| JSON\_DEPTH('[10, 20]') | JSON\_DEPTH('[[], {}]') |

+------------------------+------------------------+

| 2 | 2 |

+------------------------+------------------------+

mysql> **SELECT JSON\_DEPTH('[10, {"a": 20}]');**

+-------------------------------+

| JSON\_DEPTH('[10, {"a": 20}]') |

+-------------------------------+

| 3 |

+-------------------------------+

**[JSON\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-length)*[json\_doc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-length)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-length)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-length)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-length)**

Returns the length of a JSON document, or, if a ***path*** argument is given, the length of the value within the document identified by the path. Returns **NULL** if any argument is **NULL** or the ***path*** argument does not identify a value in the document. An error occurs if the ***json\_doc*** argument is not a valid JSON document or the ***path*** argument is not a valid path expression or contains a **\*** or **\*\*** wildcard.

The length of a document is determined as follows:

The length of a scalar is 1.

The length of an array is the number of array elements.

The length of an object is the number of object members.

The length does not count the length of nested arrays or objects.

mysql> **SELECT JSON\_LENGTH('[1, 2, {"a": 3}]');**

+---------------------------------+

| JSON\_LENGTH('[1, 2, {"a": 3}]') |

+---------------------------------+

| 3 |

+---------------------------------+

mysql> **SELECT JSON\_LENGTH('{"a": 1, "b": {"c": 30}}');**

+-----------------------------------------+

| JSON\_LENGTH('{"a": 1, "b": {"c": 30}}') |

+-----------------------------------------+

| 2 |

+-----------------------------------------+

mysql> **SELECT JSON\_LENGTH('{"a": 1, "b": {"c": 30}}', '$.b');**

+------------------------------------------------+

| JSON\_LENGTH('{"a": 1, "b": {"c": 30}}', '$.b') |

+------------------------------------------------+

| 1 |

+------------------------------------------------+

**[JSON\_TYPE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-type)*[json\_val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-type)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-type)**

Returns a **utf8mb4** string indicating the type of a JSON value. This can be an object, an array, or a scalar type, as shown here:

mysql> **SET @j = '{"a": [10, true]}';**

mysql> **SELECT JSON\_TYPE(@j);**

+---------------+

| JSON\_TYPE(@j) |

+---------------+

| OBJECT |

+---------------+

mysql> **SELECT JSON\_TYPE(JSON\_EXTRACT(@j, '$.a'));**

+------------------------------------+

| JSON\_TYPE(JSON\_EXTRACT(@j, '$.a')) |

+------------------------------------+

| ARRAY |

+------------------------------------+

mysql> **SELECT JSON\_TYPE(JSON\_EXTRACT(@j, '$.a[0]'));**

+---------------------------------------+

| JSON\_TYPE(JSON\_EXTRACT(@j, '$.a[0]')) |

+---------------------------------------+

| INTEGER |

+---------------------------------------+

mysql> **SELECT JSON\_TYPE(JSON\_EXTRACT(@j, '$.a[1]'));**

+---------------------------------------+

| JSON\_TYPE(JSON\_EXTRACT(@j, '$.a[1]')) |

+---------------------------------------+

| BOOLEAN |

+---------------------------------------+

[**JSON\_TYPE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-type) returns **NULL** if the argument is **NULL**:

mysql> **SELECT JSON\_TYPE(NULL);**

+-----------------+

| JSON\_TYPE(NULL) |

+-----------------+

| NULL |

+-----------------+

An error occurs if the argument is not a valid JSON value:

mysql> **SELECT JSON\_TYPE(1);**

ERROR 3146 (22032): Invalid data type for JSON data in argument 1

to function json\_type; a JSON string or JSON type is required.

For a non-**NULL**, non-error result, the following list describes the possible [**JSON\_TYPE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-type) return values:

Purely JSON types:

**OBJECT**: JSON objects

**ARRAY**: JSON arrays

**BOOLEAN**: The JSON true and false literals

**NULL**: The JSON null literal

Numeric types:

**INTEGER**: MySQL [**TINYINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**SMALLINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**MEDIUMINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) and [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) and [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) scalars

**DOUBLE**: MySQL [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) [**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) scalars

**DECIMAL**: MySQL [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) and [**NUMERIC**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) scalars

Temporal types:

**DATETIME**: MySQL [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) and [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) scalars

**DATE**: MySQL [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) scalars

**TIME**: MySQL [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time) scalars

String types:

**STRING**: MySQL **utf8** character type scalars: [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum), and [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set)

Binary types:

**BLOB**: MySQL binary type scalars including [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), and [**BIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#bit-type)

All other types:

**OPAQUE** (raw bits)

**[JSON\_VALID(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-valid)*[val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-valid)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-valid)**

Returns 0 or 1 to indicate whether a value is valid JSON. Returns **NULL** if the argument is **NULL**.

mysql> **SELECT JSON\_VALID('{"a": 1}');**

+------------------------+

| JSON\_VALID('{"a": 1}') |

+------------------------+

| 1 |

+------------------------+

mysql> **SELECT JSON\_VALID('hello'), JSON\_VALID('"hello"');**

+---------------------+-----------------------+

| JSON\_VALID('hello') | JSON\_VALID('"hello"') |

+---------------------+-----------------------+

| 0 | 1 |

+---------------------+-----------------------+

### 12.18.6 JSON Table Functions

This section contains information about JSON functions that convert JSON data to tabular data. In MySQL 8.0.4 and later, one such function—**JSON\_TABLE()**—is supported.

**[JSON\_TABLE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-table)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-table)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-table)*[path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-table)*[COLUMNS (](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-table)*[column\_list](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-table)*[) [AS]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-table)*[alias](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-table)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-table)**

Extracts data from a JSON document and returns it as a relational table having the specified columns. The complete syntax for this function is shown here:

JSON\_TABLE(

***expr***,

***path*** COLUMNS (***column\_list***)

) [AS] ***alias***

***column\_list***:

***column***[, ***column***][, ...]

***column***:

***name*** FOR ORDINALITY

| ***name*** ***type*** PATH ***string path*** [***on\_empty***] [***on\_error***]

| ***name*** ***type*** EXISTS PATH ***string path***

| NESTED [PATH] ***path*** COLUMNS (***column\_list***)

***on\_empty***:

{NULL | DEFAULT ***json\_string*** | ERROR} ON EMPTY

***on\_error***:

{NULL | DEFAULT ***json\_string*** | ERROR} ON ERROR

***expr***: This is an expression that returns JSON data. This can be a constant (**'{"a":1}'**), a column (**t1.json\_data**, given table **t1** specified prior to **JSON\_TABLE()** in the **FROM** clause), or a function call (**JSON\_EXTRACT(t1.json\_data,'$.post.comments')**).

***path***: A JSON path expression, which is applied to the data source. We refer to the JSON value matching the path as the row source; this is used to generate a row of relational data. The **COLUMNS** clause evaluates the row source, finds specific JSON values within the row source, and returns those JSON values as SQL values in individual columns of a row of relational data.

The ***alias*** is required. The usual rules for table aliases apply (see [Section 9.2, “Schema Object Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifiers)).

**JSON\_TABLE()** supports four types of columns, described in the following list:

***name* FOR ORDINALITY**: This type enumerates rows in the **COLUMNS** clause; the column named ***name*** is a counter whose type is **UNSIGNED INT**, and whose initial value is 1. This is equivalent to specifying a column as **AUTO\_INCREMENT** in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement, and can be used to distinguish parent rows with the same value for multiple rows generated by a **NESTED [PATH]** clause.

***name* *type* PATH *string\_path* [*on\_empty*] [*on\_error*]**: Columns of this type are used to extract values specified by ***string\_path***. ***type*** is a MySQL scalar data type (that is, it cannot be an object or array). **JSON\_TABLE()** extracts data as JSON then coerces it to the column type, using the regular automatic type conversion applying to JSON data in MySQL. A missing value triggers the ***on\_empty*** clause. Saving an object or array triggers the optional ***on error*** clause; this also occurs when an error takes place during coercion from the value saved as JSON to the table column, such as trying to save the string **'asd'** to an integer column.

***name* *type* EXISTS PATH *path***: This column returns 1 if any data is present at the location specified by ***path***, and 0 otherwise. ***type*** can be any valid MySQL data type, but should normally be specified as some variety of [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types).

**NESTED [PATH] *path* COLUMNS (*column\_list*)**: This flattens nested objects or arrays in JSON data into a single row along with the JSON values from the parent object or array. Using multiple **PATH** options allows projection of JSON values from multiple levels of nesting into a single row.

The ***path*** is relative to the parent path row path of **JSON\_TABLE()**, or the path of the parent **NESTED [PATH]** clause in the event of nested paths.

***on empty***, if specified, determines what **JSON\_TABLE()** does in the event that data is missing (depending on type). This clause is also triggered on a column in a **NESTED PATH** clause when the latter has no match and a **NULL** complemented row is produced for it. ***on empty*** takes one of the following values:

**NULL ON EMPTY**: The column is set to **NULL**; this is the default behavior.

**DEFAULT *json\_string* ON EMPTY**: the provided ***json\_string*** is parsed as JSON, as long as it is valid, and stored instead of the missing value. Column type rules also apply to the default value.

**ERROR ON EMPTY**: An error is thrown.

If used, ***on\_error*** takes one of the following values with the corresponding result as shown here:

**NULL ON ERROR**: The column is set to **NULL**; this is the default behavior.

**DEFAULT *json string* ON ERROR**: The ***json\_string*** is parsed as JSON (provided that it is valid) and stored instead of the object or array.

**ERROR ON ERROR**: An error is thrown.

Prior to MySQL 8.0.20, a warning was thrown if a type conversion error occurred with **NULL ON ERROR** or **DEFAULT ... ON ERROR** was specified or implied. In MySQL 8.0.20 and later, this is no longer the case. (Bug #30628330)

Previously, it was possible to specify **ON EMPTY** and **ON ERROR** clauses in either order. This runs counter to the SQL standard, which stipulates that **ON EMPTY**, if specified, must precede any **ON ERROR** clause. For this reason, beginning with MySQL 8.0.20, specifying **ON ERROR** before **ON EMPTY** is deprecated; trying to do so causes the server to issue a warning. Expect support for the nonstandard syntax to be removed in a future version of MySQL.

When a value saved to a column is truncated, such as saving 3.14159 in a [**DECIMAL(10,1)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) column, a warning is issued independently of any **ON ERROR** option. When multiple values are truncated in a single statement, the warning is issued only once.

Prior to MySQL 8.0.21, when the expression and path passed to this function resolved to JSON null, **JSON\_TABLE()** raised an error. In MySQL 8.0.21 and later, it returns SQL **NULL** in such cases, in accordance with the SQL standard, as shown here (Bug #31345503, Bug #99557):

mysql> **SELECT \***

-> **FROM**

-> **JSON\_TABLE(**

-> **'[ {"c1": null} ]',**

-> **'$[\*]' COLUMNS( c1 INT PATH '$.c1' ERROR ON ERROR )**

-> **) as jt;**

+------+

| c1 |

+------+

| NULL |

+------+

1 row in set (0.00 sec)

The following query demonstrates the use of **ON EMPTY** and **ON ERROR**. The row corresponding to **{"b":1}** is empty for the path **"$.a"**, and attempting to save **[1,2]** as a scalar produces an error; these rows are highlighted in the output shown.

mysql> **SELECT \***

-> **FROM**

-> **JSON\_TABLE(**

-> **'[{"a":"3"},{"a":2},{"b":1},{"a":0},{"a":[1,2]}]',**

-> **"$[\*]"**

-> **COLUMNS(**

-> **rowid FOR ORDINALITY,**

-> **ac VARCHAR(100) PATH "$.a" DEFAULT '111' ON EMPTY DEFAULT '999' ON ERROR,**

-> **aj JSON PATH "$.a" DEFAULT '{"x": 333}' ON EMPTY,**

-> **bx INT EXISTS PATH "$.b"**

-> **)**

-> **) AS tt;**

+-------+------+------------+------+

| rowid | ac | aj | bx |

+-------+------+------------+------+

| 1 | 3 | "3" | 0 |

| 2 | 2 | 2 | 0 |

| 3 | 111 | {"x": 333} | 1 |

| 4 | 0 | 0 | 0 |

| 5 | 999 | [1, 2] | 0 |

+-------+------+------------+------+

5 rows in set (0.00 sec)

Column names are subject to the usual rules and limitations governing table column names. See [Section 9.2, “Schema Object Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifiers).

All JSON and JSON path expressions are checked for validity; an invalid expression of either type causes an error.

Each match for the ***path*** preceding the **COLUMNS** keyword maps to an individual row in the result table. For example, the following query gives the result shown here:

mysql> **SELECT \***

-> **FROM**

-> **JSON\_TABLE(**

-> **'[{"x":2,"y":"8"},{"x":"3","y":"7"},{"x":"4","y":6}]',**

-> **"$[\*]" COLUMNS(**

-> **xval VARCHAR(100) PATH "$.x",**

-> **yval VARCHAR(100) PATH "$.y"**

-> **)**

-> **) AS jt1;**

+------+------+

| xval | yval |

+------+------+

| 2 | 8 |

| 3 | 7 |

| 4 | 6 |

+------+------+

The expression **"$[\*]"** matches each element of the array. You can filter the rows in the result by modifying the path. For example, using **"$[1]"** limits extraction to the second element of the JSON array used as the source, as shown here:

mysql> **SELECT \***

-> **FROM**

-> **JSON\_TABLE(**

-> **'[{"x":2,"y":"8"},{"x":"3","y":"7"},{"x":"4","y":6}]',**

-> **"$[1]" COLUMNS(**

-> **xval VARCHAR(100) PATH "$.x",**

-> **yval VARCHAR(100) PATH "$.y"**

-> **)**

-> **) AS jt1;**

+------+------+

| xval | yval |

+------+------+

| 3 | 7 |

+------+------+

Within a column definition, **"$"** passes the entire match to the column; **"$.x"** and **"$.y"** pass only the values corresponding to the keys **x** and **y**, respectively, within that match. For more information, see [JSON Path Syntax](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json-path-syntax).

**NESTED PATH** (or simply **NESTED**; **PATH** is optional) produces a set of records for each match in the **COLUMNS** clause to which it belongs. If there is no match, all columns of the nested path are set to **NULL**. This implements an outer join between the topmost clause and **NESTED [PATH]**. An inner join can be emulated by applying a suitable condition in the **WHERE** clause, as shown here:

mysql> **SELECT \***

-> **FROM**

-> **JSON\_TABLE(**

-> **'[ {"a": 1, "b": [11,111]}, {"a": 2, "b": [22,222]}, {"a":3}]',**

-> **'$[\*]' COLUMNS(**

-> **a INT PATH '$.a',**

-> **NESTED PATH '$.b[\*]' COLUMNS (b INT PATH '$')**

-> **)**

-> **) AS jt**

-> **WHERE b IS NOT NULL;**

+------+------+

| a | b |

+------+------+

| 1 | 11 |

| 1 | 111 |

| 2 | 22 |

| 2 | 222 |

+------+------+

Sibling nested paths—that is, two or more instances of **NESTED [PATH]** in the same **COLUMNS** clause—are processed one after another, one at a time. While one nested path is producing records, columns of any sibling nested path expressions are set to **NULL**. This means that the total number of records for a single match within a single containing **COLUMNS** clause is the sum and not the product of all records produced by **NESTED [PATH]** modifiers, as shown here:

mysql> **SELECT \***

-> **FROM**

-> **JSON\_TABLE(**

-> **'[{"a": 1, "b": [11,111]}, {"a": 2, "b": [22,222]}]',**

-> **'$[\*]' COLUMNS(**

-> **a INT PATH '$.a',**

-> **NESTED PATH '$.b[\*]' COLUMNS (b1 INT PATH '$'),**

-> **NESTED PATH '$.b[\*]' COLUMNS (b2 INT PATH '$')**

-> **)**

-> **) AS jt;**

+------+------+------+

| a | b1 | b2 |

+------+------+------+

| 1 | 11 | NULL |

| 1 | 111 | NULL |

| 1 | NULL | 11 |

| 1 | NULL | 111 |

| 2 | 22 | NULL |

| 2 | 222 | NULL |

| 2 | NULL | 22 |

| 2 | NULL | 222 |

+------+------+------+

A **FOR ORDINALITY** column enumerates records produced by the **COLUMNS** clause, and can be used to distinguish parent records of a nested path, especially if values in parent records are the same, as can be seen here:

mysql> **SELECT \***

-> **FROM**

-> **JSON\_TABLE(**

-> **'[{"a": "a\_val",**

'> **"b": [{"c": "c\_val", "l": [1,2]}]},**

'> **{"a": "a\_val",**

'> **"b": [{"c": "c\_val","l": [11]}, {"c": "c\_val", "l": [22]}]}]',**

-> **'$[\*]' COLUMNS(**

-> **top\_ord FOR ORDINALITY,**

-> **apath VARCHAR(10) PATH '$.a',**

-> **NESTED PATH '$.b[\*]' COLUMNS (**

-> **bpath VARCHAR(10) PATH '$.c',**

-> **ord FOR ORDINALITY,**

-> **NESTED PATH '$.l[\*]' COLUMNS (lpath varchar(10) PATH '$')**

-> **)**

-> **)**

-> **) as jt;**

+---------+---------+---------+------+-------+

| top\_ord | apath | bpath | ord | lpath |

+---------+---------+---------+------+-------+

| 1 | a\_val | c\_val | 1 | 1 |

| 1 | a\_val | c\_val | 1 | 2 |

| 2 | a\_val | c\_val | 1 | 11 |

| 2 | a\_val | c\_val | 2 | 22 |

+---------+---------+---------+------+-------+

The source document contains an array of two elements; each of these elements produces two rows. The values of **apath** and **bpath** are the same over the entire result set; this means that they cannot be used to determine whether **lpath** values came from the same or different parents. The value of the **ord** column remains the same as the set of records having **top\_ord** equal to 1, so these two values are from a single object. The remaining two values are from different objects, since they have different values in the **ord** column.

### 12.18.7 JSON Schema Validation Functions

Beginning with MySQL 8.0.17, MySQL supports validation of JSON documents against JSON schemas conforming to [Draft 4 of the JSON Schema specification](https://json-schema.org/specification-links.html#draft-4). This can be done using either of the functions detailed in this section, both of which take two arguments, a JSON schema, and a JSON document which is validated against the schema. [**JSON\_SCHEMA\_VALID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-schema-valid) returns true if the document validates against the schema, and false if it does not; [**JSON\_SCHEMA\_VALIDATION\_REPORT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-schema-validation-report) provides a report in JSON format on the validation.

Both functions handle null or invalid input as follows:

If at least one of the arguments is **NULL**, the function returns **NULL**.

If at least one of the arguments is not valid JSON, the function raises an error ([**ER\_INVALID\_TYPE\_FOR\_JSON**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_type_for_json))

In addition, if the schema is not a valid JSON object, the function returns [**ER\_INVALID\_JSON\_TYPE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_json_type).

MySQL supports the **required** attribute in JSON schemas to enforce the inclusion of required properties (see the examples in the function descriptions).

MySQL supports the **id**, **$schema**, **description**, and **type** attributes in JSON schemas but does not require any of these.

MySQL does not support external resources in JSON schemas; using the **$ref** keyword causes **JSON\_SCHEMA\_VALID()** to fail with [**ER\_NOT\_SUPPORTED\_YET**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_not_supported_yet).

**Note**

MySQL supports regular expression patterns in JSON schema, which supports but silently ignores invalid patterns (see the description of **JSON\_SCHEMA\_VALID()** for an example).

These functions are described in detail in the following list:

**[JSON\_SCHEMA\_VALID(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-schema-valid)*[schema](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-schema-valid)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-schema-valid)*[document](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-schema-valid)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-schema-valid)**

Validates a JSON ***document*** against a JSON ***schema***. Both ***schema*** and ***document*** are required. The schema must be a valid JSON object; the document must be a valid JSON document. Provided that these conditions are met: If the document validates against the schema, the function returns true (1); otherwise, it returns false (0).

In this example, we set a user variable **@schema** to the value of a a JSON schema for geographical coordinates, and another one **@document** to the value of a JSON document containing one such coordinate. We then verify that **@document** validates according to **@schema** by using them as the arguments to **JSON\_SCHEMA\_VALID()**:

mysql> **SET @schema = '{**

'> **"id": "http://json-schema.org/geo",**

'> **"$schema": "http://json-schema.org/draft-04/schema#",**

'> **"description": "A geographical coordinate",**

'> **"type": "object",**

'> **"properties": {**

'> **"latitude": {**

'> **"type": "number",**

'> **"minimum": -90,**

'> **"maximum": 90**

'> **},**

'> **"longitude": {**

'> **"type": "number",**

'> **"minimum": -180,**

'> **"maximum": 180**

'> **}**

'> **},**

'> **"required": ["latitude", "longitude"]**

'>**}';**

Query OK, 0 rows affected (0.01 sec)

mysql> **SET @document = '{**

'> **"latitude": 63.444697,**

'> **"longitude": 10.445118**

'>**}';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT JSON\_SCHEMA\_VALID(@schema, @document);**

+---------------------------------------+

| JSON\_SCHEMA\_VALID(@schema, @document) |

+---------------------------------------+

| 1 |

+---------------------------------------+

1 row in set (0.00 sec)

Since **@schema** contains the **required** attribute, we can set **@document** to a value that is otherwise valid but does not contain the required properties, then test it against **@schema**, like this:

mysql> **SET @document = '{}';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT JSON\_SCHEMA\_VALID(@schema, @document);**

+---------------------------------------+

| JSON\_SCHEMA\_VALID(@schema, @document) |

+---------------------------------------+

| 0 |

+---------------------------------------+

1 row in set (0.00 sec)

If we now set the value of **@schema** to the same JSON schema but without the **required** attribute, **@document** validates because it is a valid JSON object, even though it contains no properties, as shown here:

mysql> **SET @schema = '{**

'> **"id": "http://json-schema.org/geo",**

'> **"$schema": "http://json-schema.org/draft-04/schema#",**

'> **"description": "A geographical coordinate",**

'> **"type": "object",**

'> **"properties": {**

'> **"latitude": {**

'> **"type": "number",**

'> **"minimum": -90,**

'> **"maximum": 90**

'> **},**

'> **"longitude": {**

'> **"type": "number",**

'> **"minimum": -180,**

'> **"maximum": 180**

'> **}**

'> **}**

'>**}';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT JSON\_SCHEMA\_VALID(@schema, @document);**

+---------------------------------------+

| JSON\_SCHEMA\_VALID(@schema, @document) |

+---------------------------------------+

| 1 |

+---------------------------------------+

1 row in set (0.00 sec)

**JSON\_SCHEMA\_VALID() and CHECK constraints.** **JSON\_SCHEMA\_VALID()** can also be used to enforce **CHECK** constraints.

Consider the table **geo** created as shown here, with a JSON column **coordinate** representing a point of latitude and longitude on a map, governed by the JSON schema used as an argument in a **JSON\_SCHEMA\_VALID()** call which is passed as the expression for a **CHECK** constraint on this table:

mysql> **CREATE TABLE geo (**

-> **coordinate JSON,**

-> **CHECK(**

-> **JSON\_SCHEMA\_VALID(**

-> **'{**

'> **"type":"object",**

'> **"properties":{**

'> **"latitude":{"type":"number", "minimum":-90, "maximum":90},**

'> **"longitude":{"type":"number", "minimum":-180, "maximum":180}**

'> **},**

'> **"required": ["latitude", "longitude"]**

'> **}',**

-> **coordinate**

-> **)**

-> **)**

-> **);**

Query OK, 0 rows affected (0.45 sec)

**Note**

Because a MySQL **CHECK** constraint cannot contain references to variables, you must pass the JSON schema to **JSON\_SCHEMA\_VALID()** inline when using it to specify such a constraint for a table.

We assign JSON values representing coordinates to three variables, as shown here:

mysql> **SET @point1 = '{"latitude":59, "longitude":18}';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SET @point2 = '{"latitude":91, "longitude":0}';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SET @point3 = '{"longitude":120}';**

Query OK, 0 rows affected (0.00 sec)

The first of these values is valid, as can be seen in the following [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement:

mysql> INSERT INTO geo VALUES(@point1);

Query OK, 1 row affected (0.05 sec)

The second JSON value is invalid and so fails the constraint, as shown here:

mysql> INSERT INTO geo VALUES(@point2);

ERROR 3819 (HY000): Check constraint 'geo\_chk\_1' is violated.

In MySQL 8.0.19 and later, you can obtain precise information about the nature of the failure—in this case, that the **latitude** value exceeds the maximum defined in the schema—by issuing a [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) statement:

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Error

Code: 3934

Message: The JSON document location '#/latitude' failed requirement 'maximum' at

JSON Schema location '#/properties/latitude'.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Error

Code: 3819

Message: Check constraint 'geo\_chk\_1' is violated.

2 rows in set (0.00 sec)

The third coordinate value defined above is also invalid, since it is missing the required **latitude** property. As before, you can see this by attempting to insert the value into the **geo** table, then issuing **SHOW WARNINGS** afterwards:

mysql> **INSERT INTO geo VALUES(@point3);**

ERROR 3819 (HY000): Check constraint 'geo\_chk\_1' is violated.

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Error

Code: 3934

Message: The JSON document location '#' failed requirement 'required' at JSON

Schema location '#'.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Error

Code: 3819

Message: Check constraint 'geo\_chk\_1' is violated.

2 rows in set (0.00 sec)

See [Section 13.1.20.6, “CHECK Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-check-constraints), for more information.

JSON Schema has support for specifying regular expression patterns for strings, but the implementation used by MySQL silently ignores invalid patterns. This means that **JSON\_SCHEMA\_VALID()** can return true even when a regular expression pattern is invalid, as shown here:

mysql> **SELECT JSON\_SCHEMA\_VALID('{"type":"string","pattern":"("}', '"abc"');**

+---------------------------------------------------------------+

| JSON\_SCHEMA\_VALID('{"type":"string","pattern":"("}', '"abc"') |

+---------------------------------------------------------------+

| 1 |

+---------------------------------------------------------------+

1 row in set (0.04 sec)

**[JSON\_SCHEMA\_VALIDATION\_REPORT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-schema-validation-report)*[schema](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-schema-validation-report)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-schema-validation-report)*[document](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-schema-validation-report)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-schema-validation-report)**

Validates a JSON ***document*** against a JSON ***schema***. Both ***schema*** and ***document*** are required. As with JSON\_VALID\_SCHEMA(), the schema must be a valid JSON object, and the document must be a valid JSON document. Provided that these conditions are met, the function returns a report, as a JSON document, on the outcome of the validation. If the JSON document is considered valid according to the JSON Schema, the function returns a JSON object with one property **valid** having the value "true". If the JSON document fails validation, the function returns a JSON object which includes the properties listed here:

**valid**: Always "false" for a failed schema validation

**reason**: A human-readable string containing the reason for the failure

**schema-location**: A JSON pointer URI fragment identifier indicating where in the JSON schema the validation failed (see Note following this list)

**document-location**: A JSON pointer URI fragment identifier indicating where in the JSON document the validation failed (see Note following this list)

**schema-failed-keyword**: A string containing the name of the keyword or property in the JSON schema that was violated

**Note**

JSON pointer URI fragment identifiers are defined in [RFC 6901 - JavaScript Object Notation (JSON) Pointer](https://tools.ietf.org/html/rfc6901#page-5). (These are not the same as the JSON path notation used by [**JSON\_EXTRACT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-extract) and other MySQL JSON functions.) In this notation, **#** represents the entire document, and **#/myprop** represents the portion of the document included in the top-level property named **myprop**. See the specification just cited and the examples shown later in this section for more information.

In this example, we set a user variable **@schema** to the value of a a JSON schema for geographical coordinates, and another one **@document** to the value of a JSON document containing one such coordinate. We then verify that **@document** validates according to **@schema** by using them as the arguments to **JSON\_SCHEMA\_VALIDATION\_REORT()**:

mysql> **SET @schema = '{**

'> **"id": "http://json-schema.org/geo",**

'> **"$schema": "http://json-schema.org/draft-04/schema#",**

'> **"description": "A geographical coordinate",**

'> **"type": "object",**

'> **"properties": {**

'> **"latitude": {**

'> **"type": "number",**

'> **"minimum": -90,**

'> **"maximum": 90**

'> **},**

'> **"longitude": {**

'> **"type": "number",**

'> **"minimum": -180,**

'> **"maximum": 180**

'> **}**

'> **},**

'> **"required": ["latitude", "longitude"]**

'>**}';**

Query OK, 0 rows affected (0.01 sec)

mysql> **SET @document = '{**

'> **"latitude": 63.444697,**

'> **"longitude": 10.445118**

'>**}';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT JSON\_SCHEMA\_VALIDATION\_REPORT(@schema, @document);**

+---------------------------------------------------+

| JSON\_SCHEMA\_VALIDATION\_REPORT(@schema, @document) |

+---------------------------------------------------+

| {"valid": true} |

+---------------------------------------------------+

1 row in set (0.00 sec)

Now we set **@document** such that it specifies an illegal value for one of its properties, like this:

mysql> **SET @document = '{**

'> **"latitude": 63.444697,**

'> **"longitude": 310.445118**

'> **}';**

Validation of **@document** now fails when tested with **JSON\_SCHEMA\_VALIDATION\_REPORT()**. The output from the function call contains detailed information about the failure (with the function wrapped by [**JSON\_PRETTY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-pretty) to provide better formatting), as shown here:

mysql> **SELECT JSON\_PRETTY(JSON\_SCHEMA\_VALIDATION\_REPORT(@schema, @document))\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

JSON\_PRETTY(JSON\_SCHEMA\_VALIDATION\_REPORT(@schema, @document)): {

"valid": false,

"reason": "The JSON document location '#/longitude' failed requirement 'maximum' at JSON Schema location '#/properties/longitude'",

"schema-location": "#/properties/longitude",

"document-location": "#/longitude",

"schema-failed-keyword": "maximum"

}

1 row in set (0.00 sec)

Since **@schema** contains the **required** attribute, we can set **@document** to a value that is otherwise valid but does not contain the required properties, then test it against **@schema**. The output of **JSON\_SCHEMA\_VALIDATION\_REPORT()** shows that validation fails due to lack of a required element, like this:

mysql> **SET @document = '{}';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT JSON\_PRETTY(JSON\_SCHEMA\_VALIDATION\_REPORT(@schema, @document))\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

JSON\_PRETTY(JSON\_SCHEMA\_VALIDATION\_REPORT(@schema, @document)): {

"valid": false,

"reason": "The JSON document location '#' failed requirement 'required' at JSON Schema location '#'",

"schema-location": "#",

"document-location": "#",

"schema-failed-keyword": "required"

}

1 row in set (0.00 sec)

If we now set the value of **@schema** to the same JSON schema but without the **required** attribute, **@document** validates because it is a valid JSON object, even though it contains no properties, as shown here:

mysql> **SET @schema = '{**

'> **"id": "http://json-schema.org/geo",**

'> **"$schema": "http://json-schema.org/draft-04/schema#",**

'> **"description": "A geographical coordinate",**

'> **"type": "object",**

'> **"properties": {**

'> **"latitude": {**

'> **"type": "number",**

'> **"minimum": -90,**

'> **"maximum": 90**

'> **},**

'> **"longitude": {**

'> **"type": "number",**

'> **"minimum": -180,**

'> **"maximum": 180**

'> **}**

'> **}**

'>**}';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT JSON\_SCHEMA\_VALIDATION\_REPORT(@schema, @document);**

+---------------------------------------------------+

| JSON\_SCHEMA\_VALIDATION\_REPORT(@schema, @document) |

+---------------------------------------------------+

| {"valid": true} |

+---------------------------------------------------+

1 row in set (0.00 sec)

### 12.18.8 JSON Utility Functions

This section documents utility functions that act on JSON values, or strings that can be parsed as JSON values. [**JSON\_PRETTY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-pretty) prints out a JSON value in a format that is easy to read. [**JSON\_STORAGE\_SIZE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-storage-size) and [**JSON\_STORAGE\_FREE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-storage-free) show, respectively, the amount of storage space used by a given JSON value and the amount of space remaining in a **JSON** column following a partial update.

**[JSON\_PRETTY(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-pretty)*[json\_val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-pretty)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-pretty)**

Provides pretty-printing of JSON values similar to that implemented in PHP and by other languages and database systems. The value supplied must be a JSON value or a valid string representation of a JSON value. Extraneous whitespaces and newlines present in this value have no effect on the output. For a **NULL** value, the function returns **NULL**. If the value is not a JSON document, or if it cannot be parsed as one, the function fails with an error.

Formatting of the output from this function adheres to the following rules:

Each array element or object member appears on a separate line, indented by one additional level as compared to its parent.

Each level of indentation adds two leading spaces.

A comma separating individual array elements or object members is printed before the newline that separates the two elements or members.

The key and the value of an object member are separated by a colon followed by a space ('**:**').

An empty object or array is printed on a single line. No space is printed between the opening and closing brace.

Special characters in string scalars and key names are escaped employing the same rules used by the [**JSON\_QUOTE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-quote) function.

mysql> **SELECT JSON\_PRETTY('123'); # scalar**

+--------------------+

| JSON\_PRETTY('123') |

+--------------------+

| 123 |

+--------------------+

mysql> **SELECT JSON\_PRETTY("[1,3,5]"); # array**

+------------------------+

| JSON\_PRETTY("[1,3,5]") |

+------------------------+

| [

1,

3,

5

] |

+------------------------+

mysql> **SELECT JSON\_PRETTY('{"a":"10","b":"15","x":"25"}'); # object**

+---------------------------------------------+

| JSON\_PRETTY('{"a":"10","b":"15","x":"25"}') |

+---------------------------------------------+

| {

"a": "10",

"b": "15",

"x": "25"

} |

+---------------------------------------------+

mysql> **SELECT JSON\_PRETTY('["a",1,{"key1":**

'> **"value1"},"5", "77" ,**

'> **{"key2":["value3","valueX",**

'> **"valueY"]},"j", "2" ]')\G # nested arrays and objects**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

JSON\_PRETTY('["a",1,{"key1":

"value1"},"5", "77" ,

{"key2":["value3","valuex",

"valuey"]},"j", "2" ]'): [

"a",

1,

{

"key1": "value1"

},

"5",

"77",

{

"key2": [

"value3",

"valuex",

"valuey"

]

},

"j",

"2"

]

**[JSON\_STORAGE\_FREE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-storage-free)*[json\_val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-storage-free)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-storage-free)**

For a [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) column value, this function shows how much storage space was freed in its binary representation after it was updated in place using [**JSON\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set), [**JSON\_REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-replace), or [**JSON\_REMOVE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-remove). The argument can also be a valid JSON document or a string which can be parsed as one—either as a literal value or as the value of a user variable—in which case the function returns 0. It returns a positive, nonzero value if the argument is a **JSON** column value which has been updated as described previously, such that its binary representation takes up less space than it did prior to the update. For a **JSON** column which has been updated such that its binary representation is the same as or larger than before, or if the update was not able to take advantage of a partial update, it returns 0; it returns **NULL** if the argument is **NULL**.

If ***json\_val*** is not **NULL**, and neither is a valid JSON document nor can be successfully parsed as one, an error results.

In this example, we create a table containing a **JSON** column, then insert a row containing a JSON object:

mysql> **CREATE TABLE jtable (jcol JSON);**

Query OK, 0 rows affected (0.38 sec)

mysql> **INSERT INTO jtable VALUES**

-> **('{"a": 10, "b": "wxyz", "c": "[true, false]"}');**

Query OK, 1 row affected (0.04 sec)

mysql> **SELECT \* FROM jtable;**

+----------------------------------------------+

| jcol |

+----------------------------------------------+

| {"a": 10, "b": "wxyz", "c": "[true, false]"} |

+----------------------------------------------+

1 row in set (0.00 sec)

Now we update the column value using **JSON\_SET()** such that a partial update can be performed; in this case, we replace the value pointed to by the **c** key (the array **[true, false]**) with one that takes up less space (the integer **1**):

mysql> **UPDATE jtable**

-> **SET jcol = JSON\_SET(jcol, "$.a", 10, "$.b", "wxyz", "$.c", 1);**

Query OK, 1 row affected (0.03 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> **SELECT \* FROM jtable;**

+--------------------------------+

| jcol |

+--------------------------------+

| {"a": 10, "b": "wxyz", "c": 1} |

+--------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT JSON\_STORAGE\_FREE(jcol) FROM jtable;**

+-------------------------+

| JSON\_STORAGE\_FREE(jcol) |

+-------------------------+

| 14 |

+-------------------------+

1 row in set (0.00 sec)

The effects of successive partial updates on this free space are cumulative, as shown in this example using **JSON\_SET()** to reduce the space taken up by the value having key **b** (and making no other changes):

mysql> **UPDATE jtable**

-> **SET jcol = JSON\_SET(jcol, "$.a", 10, "$.b", "wx", "$.c", 1);**

Query OK, 1 row affected (0.03 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> **SELECT JSON\_STORAGE\_FREE(jcol) FROM jtable;**

+-------------------------+

| JSON\_STORAGE\_FREE(jcol) |

+-------------------------+

| 16 |

+-------------------------+

1 row in set (0.00 sec)

Updating the column without using **JSON\_SET()**, **JSON\_REPLACE()**, or **JSON\_REMOVE()** means that the optimizer cannot perform the update in place; in this case, **JSON\_STORAGE\_FREE()** returns 0, as shown here:

mysql> **UPDATE jtable SET jcol = '{"a": 10, "b": 1}';**

Query OK, 1 row affected (0.05 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> **SELECT JSON\_STORAGE\_FREE(jcol) FROM jtable;**

+-------------------------+

| JSON\_STORAGE\_FREE(jcol) |

+-------------------------+

| 0 |

+-------------------------+

1 row in set (0.00 sec)

Partial updates of JSON documents can be performed only on column values. For a user variable that stores a JSON value, the value is always completely replaced, even when the update is performed using **JSON\_SET()**:

mysql> **SET @j = '{"a": 10, "b": "wxyz", "c": "[true, false]"}';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SET @j = JSON\_SET(@j, '$.a', 10, '$.b', 'wxyz', '$.c', '1');**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT @j, JSON\_STORAGE\_FREE(@j) AS Free;**

+----------------------------------+------+

| @j | Free |

+----------------------------------+------+

| {"a": 10, "b": "wxyz", "c": "1"} | 0 |

+----------------------------------+------+

1 row in set (0.00 sec)

For a JSON literal, this function always returns 0:

mysql> **SELECT JSON\_STORAGE\_FREE('{"a": 10, "b": "wxyz", "c": "1"}') AS Free;**

+------+

| Free |

+------+

| 0 |

+------+

1 row in set (0.00 sec)

**[JSON\_STORAGE\_SIZE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-storage-size)*[json\_val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-storage-size)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-storage-size)**

This function returns the number of bytes used to store the binary representation of a JSON document. When the argument is a **JSON** column, this is the space used to store the JSON document as it was inserted into the column, prior to any partial updates that may have been performed on it afterwards. ***json\_val*** must be a valid JSON document or a string which can be parsed as one. In the case where it is string, the function returns the amount of storage space in the JSON binary representation that is created by parsing the string as JSON and converting it to binary. It returns **NULL** if the argument is **NULL**.

An error results when ***json\_val*** is not **NULL**, and is not—or cannot be successfully parsed as—a JSON document.

To illustrate this function's behavior when used with a **JSON** column as its argument, we create a table named **jtable** containing a **JSON** column **jcol**, insert a JSON value into the table, then obtain the storage space used by this column with **JSON\_STORAGE\_SIZE()**, as shown here:

mysql> **CREATE TABLE jtable (jcol JSON);**

Query OK, 0 rows affected (0.42 sec)

mysql> **INSERT INTO jtable VALUES**

-> **('{"a": 1000, "b": "wxyz", "c": "[1, 3, 5, 7]"}');**

Query OK, 1 row affected (0.04 sec)

mysql> **SELECT**

-> **jcol,**

-> **JSON\_STORAGE\_SIZE(jcol) AS Size,**

-> **JSON\_STORAGE\_FREE(jcol) AS Free**

-> **FROM jtable;**

+-----------------------------------------------+------+------+

| jcol | Size | Free |

+-----------------------------------------------+------+------+

| {"a": 1000, "b": "wxyz", "c": "[1, 3, 5, 7]"} | 47 | 0 |

+-----------------------------------------------+------+------+

1 row in set (0.00 sec)

According to the output of **JSON\_STORAGE\_SIZE()**, the JSON document inserted into the column takes up 47 bytes. We also checked the amount of space freed by any previous partial updates of the column using [**JSON\_STORAGE\_FREE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-storage-free); since no updates have yet been performed, this is 0, as expected.

Next we perform an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) on the table that should result in a partial update of the document stored in **jcol**, and then test the result as shown here:

mysql> **UPDATE jtable SET jcol =**

-> **JSON\_SET(jcol, "$.b", "a");**

Query OK, 1 row affected (0.04 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> **SELECT**

-> **jcol,**

-> **JSON\_STORAGE\_SIZE(jcol) AS Size,**

-> **JSON\_STORAGE\_FREE(jcol) AS Free**

-> **FROM jtable;**

+--------------------------------------------+------+------+

| jcol | Size | Free |

+--------------------------------------------+------+------+

| {"a": 1000, "b": "a", "c": "[1, 3, 5, 7]"} | 47 | 3 |

+--------------------------------------------+------+------+

1 row in set (0.00 sec)

The value returned by **JSON\_STORAGE\_FREE()** in the previous query indicates that a partial update of the JSON document was performed, and that this freed 3 bytes of space used to store it. The result returned by **JSON\_STORAGE\_SIZE()** is unchanged by the partial update.

Partial updates are supported for updates using [**JSON\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set), [**JSON\_REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-replace), or [**JSON\_REMOVE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-remove). The direct assignment of a value to a **JSON** column cannot be partially updated; following such an update, **JSON\_STORAGE\_SIZE()** always shows the storage used for the newly-set value:

mysql> **UPDATE jtable**

mysql> **SET jcol = '{"a": 4.55, "b": "wxyz", "c": "[true, false]"}';**

Query OK, 1 row affected (0.04 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> **SELECT**

-> **jcol,**

-> **JSON\_STORAGE\_SIZE(jcol) AS Size,**

-> **JSON\_STORAGE\_FREE(jcol) AS Free**

-> **FROM jtable;**

+------------------------------------------------+------+------+

| jcol | Size | Free |

+------------------------------------------------+------+------+

| {"a": 4.55, "b": "wxyz", "c": "[true, false]"} | 56 | 0 |

+------------------------------------------------+------+------+

1 row in set (0.00 sec)

A JSON user variable cannot be partially updated. This means that this function always shows the space currently used to store a JSON document in a user variable:

mysql> **SET @j = '[100, "sakila", [1, 3, 5], 425.05]';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT @j, JSON\_STORAGE\_SIZE(@j) AS Size;**

+------------------------------------+------+

| @j | Size |

+------------------------------------+------+

| [100, "sakila", [1, 3, 5], 425.05] | 45 |

+------------------------------------+------+

1 row in set (0.00 sec)

mysql> **SET @j = JSON\_SET(@j, '$[1]', "json");**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT @j, JSON\_STORAGE\_SIZE(@j) AS Size;**

+----------------------------------+------+

| @j | Size |

+----------------------------------+------+

| [100, "json", [1, 3, 5], 425.05] | 43 |

+----------------------------------+------+

1 row in set (0.00 sec)

mysql> **SET @j = JSON\_SET(@j, '$[2][0]', JSON\_ARRAY(10, 20, 30));**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT @j, JSON\_STORAGE\_SIZE(@j) AS Size;**

+---------------------------------------------+------+

| @j | Size |

+---------------------------------------------+------+

| [100, "json", [[10, 20, 30], 3, 5], 425.05] | 56 |

+---------------------------------------------+------+

1 row in set (0.00 sec)

For a JSON literal, this function always returns the current storage space used:

mysql> **SELECT**

-> **JSON\_STORAGE\_SIZE('[100, "sakila", [1, 3, 5], 425.05]') AS A,**

-> **JSON\_STORAGE\_SIZE('{"a": 1000, "b": "a", "c": "[1, 3, 5, 7]"}') AS B,**

-> **JSON\_STORAGE\_SIZE('{"a": 1000, "b": "wxyz", "c": "[1, 3, 5, 7]"}') AS C,**

-> **JSON\_STORAGE\_SIZE('[100, "json", [[10, 20, 30], 3, 5], 425.05]') AS D;**

+----+----+----+----+

| A | B | C | D |

+----+----+----+----+

| 45 | 44 | 47 | 56 |

+----+----+----+----+

1 row in set (0.00 sec)

## 12.19 Functions Used with Global Transaction Identifiers (GTIDs)

The functions described in this section are used with GTID-based replication. It is important to keep in mind that all of these functions take string representations of GTID sets as arguments. As such, the GTID sets must always be quoted when used with them. See [GTID Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-concepts-gtid-sets) for more information.

The union of two GTID sets is simply their representations as strings, joined together with an interposed comma. In other words, you can define a very simple function for obtaining the union of two GTID sets, similar to that created here:

CREATE FUNCTION GTID\_UNION(g1 TEXT, g2 TEXT)

RETURNS TEXT DETERMINISTIC

RETURN CONCAT(g1,',',g2);

For more information about GTIDs and how these GTID functions are used in practice, see [Section 17.1.3, “Replication with Global Transaction Identifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids).

**Table 12.24 GTID Functions**

| **Name** | **Description** | **Deprecated** |
| --- | --- | --- |
| [**GTID\_SUBSET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_gtid-subset) | Return true if all GTIDs in subset are also in set; otherwise false. |  |
| [**GTID\_SUBTRACT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_gtid-subtract) | Return all GTIDs in set that are not in subset. |  |
| [**WAIT\_FOR\_EXECUTED\_GTID\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_wait-for-executed-gtid-set) | Wait until the given GTIDs have executed on the replica. |  |
| [**WAIT\_UNTIL\_SQL\_THREAD\_AFTER\_GTIDS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_wait-until-sql-thread-after-gtids) | Use **WAIT\_FOR\_EXECUTED\_GTID\_SET()**. | 8.0.18 |

**[GTID\_SUBSET(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_gtid-subset)*[set1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_gtid-subset)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_gtid-subset)*[set2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_gtid-subset)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_gtid-subset)**

Given two sets of global transaction identifiers ***set1*** and ***set2***, returns true if all GTIDs in ***set1*** are also in ***set2***. Returns false otherwise.

The GTID sets used with this function are represented as strings, as shown in the following examples:

mysql> **SELECT GTID\_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:23',**

-> **'3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57')\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GTID\_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:23',

'3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57'): 1

1 row in set (0.00 sec)

mysql> **SELECT GTID\_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:23-25',**

-> **'3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57')\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GTID\_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:23-25',

'3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57'): 1

1 row in set (0.00 sec)

mysql> **SELECT GTID\_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:20-25',**

-> **'3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57')\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GTID\_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:20-25',

'3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57'): 0

1 row in set (0.00 sec)

**[GTID\_SUBTRACT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_gtid-subtract)*[set1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_gtid-subtract)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_gtid-subtract)*[set2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_gtid-subtract)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_gtid-subtract)**

Given two sets of global transaction identifiers ***set1*** and ***set2***, returns only those GTIDs from ***set1*** that are not in ***set2***.

All GTID sets used with this function are represented as strings and must be quoted, as shown in these examples:

mysql> **SELECT GTID\_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',**

-> **'3E11FA47-71CA-11E1-9E33-C80AA9429562:21')\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GTID\_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',

'3E11FA47-71CA-11E1-9E33-C80AA9429562:21'): 3e11fa47-71ca-11e1-9e33-c80aa9429562:22-57

1 row in set (0.00 sec)

mysql> **SELECT GTID\_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',**

-> **'3E11FA47-71CA-11E1-9E33-C80AA9429562:20-25')\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GTID\_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',

'3E11FA47-71CA-11E1-9E33-C80AA9429562:20-25'): 3e11fa47-71ca-11e1-9e33-c80aa9429562:26-57

1 row in set (0.00 sec)

mysql> **SELECT GTID\_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',**

-> **'3E11FA47-71CA-11E1-9E33-C80AA9429562:23-24')\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GTID\_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',

'3E11FA47-71CA-11E1-9E33-C80AA9429562:23-24'): 3e11fa47-71ca-11e1-9e33-c80aa9429562:21-22:25-57

1 row in set (0.01 sec)

**[WAIT\_FOR\_EXECUTED\_GTID\_SET(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-for-executed-gtid-set)*[gtid\_set](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-for-executed-gtid-set)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-for-executed-gtid-set)*[timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-for-executed-gtid-set)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-for-executed-gtid-set)**

Wait until the server has applied all of the transactions whose global transaction identifiers are contained in ***gtid\_set***; that is, until the condition GTID\_SUBSET(***gtid\_subset***, **@@GLOBAL.gtid\_executed**) holds. See [Section 17.1.3.1, “GTID Format and Storage”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-concepts) for a definition of GTID sets.

If a timeout is specified, and ***timeout*** seconds elapse before all of the transactions in the GTID set have been applied, the function stops waiting. ***timeout*** is optional, and the default timeout is 0 seconds, in which case the function always waits until all of the transactions in the GTID set have been applied.

**WAIT\_FOR\_EXECUTED\_GTID\_SET()** monitors all the GTIDs that are applied on the server, including transactions that arrive from all replication channels and user clients. It does not take into account whether replication channels have been started or stopped.

For more information, see [Section 17.1.3, “Replication with Global Transaction Identifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids).

GTID sets used with this function are represented as strings and so must be quoted as shown in the following example:

mysql> **SELECT WAIT\_FOR\_EXECUTED\_GTID\_SET('3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5');**

-> 0

For a syntax description for GTID sets, see [Section 17.1.3.1, “GTID Format and Storage”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-concepts).

For **WAIT\_FOR\_EXECUTED\_GTID\_SET()**, the return value is the state of the query, where 0 represents success, and 1 represents timeout. Any other failures generate an error.

[**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) cannot be changed to OFF while any client is using this function to wait for GTIDs to be applied.

**[WAIT\_UNTIL\_SQL\_THREAD\_AFTER\_GTIDS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-until-sql-thread-after-gtids)*[gtid\_set](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-until-sql-thread-after-gtids)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-until-sql-thread-after-gtids)*[timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-until-sql-thread-after-gtids)*[][,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-until-sql-thread-after-gtids)*[channel](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-until-sql-thread-after-gtids)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_wait-until-sql-thread-after-gtids)**

**WAIT\_UNTIL\_SQL\_THREAD\_AFTER\_GTIDS()** is deprecated. Use **WAIT\_FOR\_EXECUTED\_GTID\_SET()** instead, which works regardless of the replication channel or user client through which the specified transactions arrive on the server.

## 12.20 Aggregate Functions

[12.20.1 Aggregate Function Descriptions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#aggregate-functions)

[12.20.2 GROUP BY Modifiers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#group-by-modifiers)

[12.20.3 MySQL Handling of GROUP BY](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#group-by-handling)

[12.20.4 Detection of Functional Dependence](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#group-by-functional-dependence)

Aggregate functions operate on sets of values. They are often used with a **GROUP BY** clause to group values into subsets. This section describes most aggregate functions. For information about aggregate functions that operate on geometry values, see [Section 12.17.12, “Spatial Aggregate Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#spatial-aggregate-functions).

### 12.20.1 Aggregate Function Descriptions

This section describes aggregate functions that operate on sets of values. They are often used with a **GROUP BY** clause to group values into subsets.

**Table 12.25 Aggregate Functions**

|  |  |
| --- | --- |
| Name | Description |
| [AVG()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_avg) | Return the average value of the argument |
| [BIT\_AND()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-and) | Return bitwise AND |
| [BIT\_OR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-or) | Return bitwise OR |
| [BIT\_XOR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-xor) | Return bitwise XOR |
| [COUNT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count) | Return a count of the number of rows returned |
| [COUNT(DISTINCT)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count-distinct) | Return the count of a number of different values |
| [GROUP\_CONCAT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_group-concat) | Return a concatenated string |
| [JSON\_ARRAYAGG()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-arrayagg) | Return result set as a single JSON array |
| [JSON\_OBJECTAGG()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-objectagg) | Return result set as a single JSON object |
| [MAX()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_max) | Return the maximum value |
| [MIN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_min) | Return the minimum value |
| [STD()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_std) | Return the population standard deviation |
| [STDDEV()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev) | Return the population standard deviation |
| [STDDEV\_POP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev-pop) | Return the population standard deviation |
| [STDDEV\_SAMP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev-samp) | Return the sample standard deviation |
| [SUM()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sum) | Return the sum |
| [VAR\_POP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_var-pop) | Return the population standard variance |
| [VAR\_SAMP()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_var-samp) | Return the sample variance |
| [VARIANCE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_variance) | Return the population standard variance |

Unless otherwise stated, aggregate functions ignore **NULL** values.

If you use an aggregate function in a statement containing no **GROUP BY** clause, it is equivalent to grouping on all rows. For more information, see [Section 12.20.3, “MySQL Handling of GROUP BY”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#group-by-handling).

Most aggregate functions can be used as window functions. Those that can be used this way are signified in their syntax description by **[*over\_clause*]**, representing an optional **OVER** clause. ***over\_clause*** is described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage), which also includes other information about window function usage.

For numeric arguments, the variance and standard deviation functions return a [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) value. The [**SUM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sum) and [**AVG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_avg) functions return a [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) value for exact-value arguments (integer or [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types)), and a [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) value for approximate-value arguments ([**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) or [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types)).

The [**SUM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sum) and [**AVG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_avg) aggregate functions do not work with temporal values. (They convert the values to numbers, losing everything after the first nonnumeric character.) To work around this problem, convert to numeric units, perform the aggregate operation, and convert back to a temporal value. Examples:

SELECT SEC\_TO\_TIME(SUM(TIME\_TO\_SEC(***time\_col***))) FROM ***tbl\_name***;

SELECT FROM\_DAYS(SUM(TO\_DAYS(***date\_col***))) FROM ***tbl\_name***;

Functions such as [**SUM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sum) or [**AVG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_avg) that expect a numeric argument cast the argument to a number if necessary. For [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) or [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) values, the cast operation causes the underlying numeric value to be used.

The [**BIT\_AND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-and), [**BIT\_OR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-or), and [**BIT\_XOR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-xor) aggregate functions perform bit operations. Prior to MySQL 8.0, bit functions and operators required [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) (64-bit integer) arguments and returned [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) values, so they had a maximum range of 64 bits. Non-[**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) arguments were converted to [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) prior to performing the operation and truncation could occur.

In MySQL 8.0, bit functions and operators permit binary string type arguments ([**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), and the [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types) and return a value of like type, which enables them to take arguments and produce return values larger than 64 bits. For discussion about argument evaluation and result types for bit operations, see the introductory discussion in [Section 12.13, “Bit Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-functions).

**[AVG([DISTINCT]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_avg)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_avg)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_avg)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_avg)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_avg)**

Returns the average value of ***expr***. The **DISTINCT** option can be used to return the average of the distinct values of ***expr***.

If there are no matching rows, [**AVG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_avg) returns **NULL**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage); it cannot be used with **DISTINCT**.

mysql> **SELECT student\_name, AVG(test\_score)**

**FROM student**

**GROUP BY student\_name;**

**[BIT\_AND(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-and)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-and)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-and)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-and)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-and)**

Returns the bitwise **AND** of all bits in ***expr***.

The result type depends on whether the function argument values are evaluated as binary strings or numbers:

Binary-string evaluation occurs when the argument values have a binary string type, and the argument is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument value conversion to unsigned 64-bit integers as necessary.

Binary-string evaluation produces a binary string of the same length as the argument values. If argument values have unequal lengths, an [**ER\_INVALID\_BITWISE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_operands_size) error occurs. If the argument size exceeds 511 bytes, an [**ER\_INVALID\_BITWISE\_AGGREGATE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_aggregate_operands_size) error occurs. Numeric evaluation produces an unsigned 64-bit integer.

If there are no matching rows, [**BIT\_AND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-and) returns a neutral value (all bits set to 1) having the same length as the argument values.

**NULL** values do not affect the result unless all values are **NULL**. In that case, the result is a neutral value having the same length as the argument values.

For more information discussion about argument evaluation and result types, see the introductory discussion in [Section 12.13, “Bit Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-functions).

As of MySQL 8.0.12, this function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

**[BIT\_OR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-or)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-or)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-or)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-or)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-or)**

Returns the bitwise **OR** of all bits in ***expr***.

The result type depends on whether the function argument values are evaluated as binary strings or numbers:

Binary-string evaluation occurs when the argument values have a binary string type, and the argument is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument value conversion to unsigned 64-bit integers as necessary.

Binary-string evaluation produces a binary string of the same length as the argument values. If argument values have unequal lengths, an [**ER\_INVALID\_BITWISE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_operands_size) error occurs. If the argument size exceeds 511 bytes, an [**ER\_INVALID\_BITWISE\_AGGREGATE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_aggregate_operands_size) error occurs. Numeric evaluation produces an unsigned 64-bit integer.

If there are no matching rows, [**BIT\_OR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-or) returns a neutral value (all bits set to 0) having the same length as the argument values.

**NULL** values do not affect the result unless all values are **NULL**. In that case, the result is a neutral value having the same length as the argument values.

For more information discussion about argument evaluation and result types, see the introductory discussion in [Section 12.13, “Bit Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-functions).

As of MySQL 8.0.12, this function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

**[BIT\_XOR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-xor)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-xor)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-xor)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-xor)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bit-xor)**

Returns the bitwise [**XOR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_xor) of all bits in ***expr***.

The result type depends on whether the function argument values are evaluated as binary strings or numbers:

Binary-string evaluation occurs when the argument values have a binary string type, and the argument is not a hexadecimal literal, bit literal, or **NULL** literal. Numeric evaluation occurs otherwise, with argument value conversion to unsigned 64-bit integers as necessary.

Binary-string evaluation produces a binary string of the same length as the argument values. If argument values have unequal lengths, an [**ER\_INVALID\_BITWISE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_operands_size) error occurs. If the argument size exceeds 511 bytes, an [**ER\_INVALID\_BITWISE\_AGGREGATE\_OPERANDS\_SIZE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_invalid_bitwise_aggregate_operands_size) error occurs. Numeric evaluation produces an unsigned 64-bit integer.

If there are no matching rows, [**BIT\_XOR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bit-xor) returns a neutral value (all bits set to 0) having the same length as the argument values.

**NULL** values do not affect the result unless all values are **NULL**. In that case, the result is a neutral value having the same length as the argument values.

For more information discussion about argument evaluation and result types, see the introductory discussion in [Section 12.13, “Bit Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#bit-functions).

As of MySQL 8.0.12, this function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

**[COUNT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_count)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_count)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_count)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_count)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_count)**

Returns a count of the number of non-**NULL** values of ***expr*** in the rows retrieved by a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. The result is a [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) value.

If there are no matching rows, [**COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count) returns **0**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

mysql> **SELECT student.student\_name,COUNT(\*)**

**FROM student,course**

**WHERE student.student\_id=course.student\_id**

**GROUP BY student\_name;**

[**COUNT(\*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count) is somewhat different in that it returns a count of the number of rows retrieved, whether or not they contain **NULL** values.

For transactional storage engines such as **InnoDB**, storing an exact row count is problematic. Multiple transactions may be occurring at the same time, each of which may affect the count.

**InnoDB** does not keep an internal count of rows in a table because concurrent transactions might “see” different numbers of rows at the same time. Consequently, **SELECT COUNT(\*)** statements only count rows visible to the current transaction.

As of MySQL 8.0.13, **SELECT COUNT(\*) FROM *tbl\_name*** query performance for **InnoDB** tables is optimized for single-threaded workloads if there are no extra clauses such as **WHERE** or **GROUP BY**.

**InnoDB** processes **SELECT COUNT(\*)** statements by traversing the smallest available secondary index unless an index or optimizer hint directs the optimizer to use a different index. If a secondary index is not present, **InnoDB** processes **SELECT COUNT(\*)** statements by scanning the clustered index.

Processing **SELECT COUNT(\*)** statements takes some time if index records are not entirely in the buffer pool. For a faster count, create a counter table and let your application update it according to the inserts and deletes it does. However, this method may not scale well in situations where thousands of concurrent transactions are initiating updates to the same counter table. If an approximate row count is sufficient, use [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status).

**InnoDB** handles **SELECT COUNT(\*)** and **SELECT COUNT(1)** operations in the same way. There is no performance difference.

For **MyISAM** tables, [**COUNT(\*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count) is optimized to return very quickly if the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) retrieves from one table, no other columns are retrieved, and there is no **WHERE** clause. For example:

mysql> **SELECT COUNT(\*) FROM student;**

This optimization only applies to **MyISAM** tables, because an exact row count is stored for this storage engine and can be accessed very quickly. **COUNT(1)** is only subject to the same optimization if the first column is defined as **NOT NULL**.

**[COUNT(DISTINCT](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_count)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_count)*[,[](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_count)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_count)*[...])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_count)**

Returns a count of the number of rows with different non-**NULL** ***expr*** values.

If there are no matching rows, [**COUNT(DISTINCT)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count) returns **0**.

mysql> **SELECT COUNT(DISTINCT results) FROM student;**

In MySQL, you can obtain the number of distinct expression combinations that do not contain **NULL** by giving a list of expressions. In standard SQL, you would have to do a concatenation of all expressions inside [**COUNT(DISTINCT ...)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count).

**[GROUP\_CONCAT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_group-concat)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_group-concat)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_group-concat)**

This function returns a string result with the concatenated non-**NULL** values from a group. It returns **NULL** if there are no non-**NULL** values. The full syntax is as follows:

GROUP\_CONCAT([DISTINCT] ***expr*** [,***expr*** ...]

[ORDER BY {***unsigned\_integer*** | ***col\_name*** | ***expr***}

[ASC | DESC] [,***col\_name*** ...]]

[SEPARATOR ***str\_val***])

mysql> **SELECT student\_name,**

**GROUP\_CONCAT(test\_score)**

**FROM student**

**GROUP BY student\_name;**

Or:

mysql> **SELECT student\_name,**

**GROUP\_CONCAT(DISTINCT test\_score**

**ORDER BY test\_score DESC SEPARATOR ' ')**

**FROM student**

**GROUP BY student\_name;**

In MySQL, you can get the concatenated values of expression combinations. To eliminate duplicate values, use the **DISTINCT** clause. To sort values in the result, use the **ORDER BY** clause. To sort in reverse order, add the **DESC** (descending) keyword to the name of the column you are sorting by in the **ORDER BY** clause. The default is ascending order; this may be specified explicitly using the **ASC** keyword. The default separator between values in a group is comma (**,**). To specify a separator explicitly, use **SEPARATOR** followed by the string literal value that should be inserted between group values. To eliminate the separator altogether, specify **SEPARATOR ''**.

The result is truncated to the maximum length that is given by the [**group\_concat\_max\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_group_concat_max_len) system variable, which has a default value of 1024. The value can be set higher, although the effective maximum length of the return value is constrained by the value of [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet). The syntax to change the value of [**group\_concat\_max\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_group_concat_max_len) at runtime is as follows, where ***val*** is an unsigned integer:

SET [GLOBAL | SESSION] group\_concat\_max\_len = ***val***;

The return value is a nonbinary or binary string, depending on whether the arguments are nonbinary or binary strings. The result type is [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) unless [**group\_concat\_max\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_group_concat_max_len) is less than or equal to 512, in which case the result type is [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) or [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary).

See also [**CONCAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat) and [**CONCAT\_WS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat-ws): [Section 12.8, “String Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#string-functions).

**[JSON\_ARRAYAGG(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-arrayagg)*[col\_or\_expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-arrayagg)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-arrayagg)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-arrayagg)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-arrayagg)**

Aggregates a result set as a single [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) array whose elements consist of the rows. The order of elements in this array is undefined. The function acts on a column or an expression that evaluates to a single value. Returns **NULL** if the result contains no rows, or in the event of an error.

As of MySQL 8.0.14, this function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

mysql> **SELECT o\_id, attribute, value FROM t3;**

+------+-----------+-------+

| o\_id | attribute | value |

+------+-----------+-------+

| 2 | color | red |

| 2 | fabric | silk |

| 3 | color | green |

| 3 | shape | square|

+------+-----------+-------+

4 rows in set (0.00 sec)

mysql> **SELECT o\_id, JSON\_ARRAYAGG(attribute) AS attributes**

> **FROM t3 GROUP BY o\_id;**

+------+---------------------+

| o\_id | attributes |

+------+---------------------+

| 2 | ["color", "fabric"] |

| 3 | ["color", "shape"] |

+------+---------------------+

2 rows in set (0.00 sec)

**[JSON\_OBJECTAGG(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-objectagg)*[key](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-objectagg)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-objectagg)*[value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-objectagg)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-objectagg)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-objectagg)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_json-objectagg)**

Takes two column names or expressions as arguments, the first of these being used as a key and the second as a value, and returns a JSON object containing key-value pairs. Returns **NULL** if the result contains no rows, or in the event of an error. An error occurs if any key name is **NULL** or the number of arguments is not equal to 2.

As of MySQL 8.0.14, this function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

mysql> **SELECT o\_id, attribute, value FROM t3;**

+------+-----------+-------+

| o\_id | attribute | value |

+------+-----------+-------+

| 2 | color | red |

| 2 | fabric | silk |

| 3 | color | green |

| 3 | shape | square|

+------+-----------+-------+

4 rows in set (0.00 sec)

mysql> **SELECT o\_id, JSON\_OBJECTAGG(attribute, value)**

> **FROM t3 GROUP BY o\_id;**

+------+---------------------------------------+

| o\_id | JSON\_OBJECTAGG(attribute, value) |

+------+---------------------------------------+

| 2 | {"color": "red", "fabric": "silk"} |

| 3 | {"color": "green", "shape": "square"} |

+------+---------------------------------------+

2 rows in set (0.00 sec)

**Duplicate key handling.** When the result of this function is normalized, values having duplicate keys are discarded. In keeping with the MySQL [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) data type specification that does not permit duplicate keys, only the last value encountered is used with that key in the returned object (“last duplicate key wins”). This means that the result of using this function on columns from a **SELECT** can depend on the order in which the rows are returned, which is not guaranteed.

When used as a window function, if there are duplicate keys within a frame, only the last value for the key is present in the result. The value for the key from the last row in the frame is deterministic if the **ORDER BY** specification guarantees that the values have a specific order. If not, the resulting value of the key is nondeterministic.

Consider the following:

mysql> **CREATE TABLE t(c VARCHAR(10), i INT);**

Query OK, 0 rows affected (0.33 sec)

mysql> **INSERT INTO t VALUES ('key', 3), ('key', 4), ('key', 5);**

Query OK, 3 rows affected (0.10 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> **SELECT c, i FROM t;**

+------+------+

| c | i |

+------+------+

| key | 3 |

| key | 4 |

| key | 5 |

+------+------+

3 rows in set (0.00 sec)

mysql> **SELECT JSON\_OBJECTAGG(c, i) FROM t;**

+----------------------+

| JSON\_OBJECTAGG(c, i) |

+----------------------+

| {"key": 5} |

+----------------------+

1 row in set (0.00 sec)

mysql> **DELETE FROM t;**

Query OK, 3 rows affected (0.08 sec)

mysql> **INSERT INTO t VALUES ('key', 3), ('key', 5), ('key', 4);**

Query OK, 3 rows affected (0.06 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> **SELECT c, i FROM t;**

+------+------+

| c | i |

+------+------+

| key | 3 |

| key | 5 |

| key | 4 |

+------+------+

3 rows in set (0.00 sec)

mysql> **SELECT JSON\_OBJECTAGG(c, i) FROM t;**

+----------------------+

| JSON\_OBJECTAGG(c, i) |

+----------------------+

| {"key": 4} |

+----------------------+

1 row in set (0.00 sec)

The key chosen from the last query is nondeterministic. If you prefer a particular key ordering, you can invoke **JSON\_OBJECTAGG()** as a window function by including an **OVER** clause with an **ORDER BY** specification to impose a particular order on frame rows. The following examples show what happens with and without **ORDER BY** for a few different frame specifications.

Without **ORDER BY**, the frame is the entire partition:

mysql> **SELECT JSON\_OBJECTAGG(c, i)**

**OVER () AS json\_object FROM t;**

+-------------+

| json\_object |

+-------------+

| {"key": 4} |

| {"key": 4} |

| {"key": 4} |

+-------------+

With **ORDER BY**, where the frame is the default of **RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW** (in both ascending and descending order):

mysql> **SELECT JSON\_OBJECTAGG(c, i)**

**OVER (ORDER BY i) AS json\_object FROM t;**

+-------------+

| json\_object |

+-------------+

| {"key": 3} |

| {"key": 4} |

| {"key": 5} |

+-------------+

mysql> **SELECT JSON\_OBJECTAGG(c, i)**

**OVER (ORDER BY i DESC) AS json\_object FROM t;**

+-------------+

| json\_object |

+-------------+

| {"key": 5} |

| {"key": 4} |

| {"key": 3} |

+-------------+

With **ORDER BY** and an explicit frame of the entire partition:

mysql> **SELECT JSON\_OBJECTAGG(c, i)**

**OVER (ORDER BY i**

**ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)**

**AS json\_object**

**FROM t;**

+-------------+

| json\_object |

+-------------+

| {"key": 5} |

| {"key": 5} |

| {"key": 5} |

+-------------+

To return a particular key value (such as the smallest or largest), include a **LIMIT** clause in the appropriate query. For example:

mysql> **SELECT JSON\_OBJECTAGG(c, i)**

**OVER (ORDER BY i) AS json\_object FROM t LIMIT 1;**

+-------------+

| json\_object |

+-------------+

| {"key": 3} |

+-------------+

mysql> **SELECT JSON\_OBJECTAGG(c, i)**

**OVER (ORDER BY i DESC) AS json\_object FROM t LIMIT 1;**

+-------------+

| json\_object |

+-------------+

| {"key": 5} |

+-------------+

See [Normalization, Merging, and Autowrapping of JSON Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json-normalization), for additional information and examples.

**[MAX([DISTINCT]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_max)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_max)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_max)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_max)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_max)**

Returns the maximum value of ***expr***. [**MAX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_max) may take a string argument; in such cases, it returns the maximum string value. See [Section 8.3.1, “How MySQL Uses Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#mysql-indexes). The **DISTINCT** keyword can be used to find the maximum of the distinct values of ***expr***, however, this produces the same result as omitting **DISTINCT**.

If there are no matching rows, [**MAX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_max) returns **NULL**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage); it cannot be used with **DISTINCT**.

mysql> **SELECT student\_name, MIN(test\_score), MAX(test\_score)**

**FROM student**

**GROUP BY student\_name;**

For [**MAX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_max), MySQL currently compares [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) and [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) columns by their string value rather than by the string's relative position in the set. This differs from how **ORDER BY** compares them.

**[MIN([DISTINCT]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_min)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_min)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_min)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_min)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_min)**

Returns the minimum value of ***expr***. [**MIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_min) may take a string argument; in such cases, it returns the minimum string value. See [Section 8.3.1, “How MySQL Uses Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#mysql-indexes). The **DISTINCT** keyword can be used to find the minimum of the distinct values of ***expr***, however, this produces the same result as omitting **DISTINCT**.

If there are no matching rows, [**MIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_min) returns **NULL**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage); it cannot be used with **DISTINCT**.

mysql> **SELECT student\_name, MIN(test\_score), MAX(test\_score)**

**FROM student**

**GROUP BY student\_name;**

For [**MIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_min), MySQL currently compares [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) and [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) columns by their string value rather than by the string's relative position in the set. This differs from how **ORDER BY** compares them.

**[STD(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_std)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_std)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_std)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_std)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_std)**

Returns the population standard deviation of ***expr***. [**STD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_std) is a synonym for the standard SQL function [**STDDEV\_POP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev-pop), provided as a MySQL extension.

If there are no matching rows, [**STD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_std) returns **NULL**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

**[STDDEV(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev)**

Returns the population standard deviation of ***expr***. [**STDDEV()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev) is a synonym for the standard SQL function [**STDDEV\_POP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev-pop), provided for compatibility with Oracle.

If there are no matching rows, [**STDDEV()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev) returns **NULL**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

**[STDDEV\_POP(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev-pop)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev-pop)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev-pop)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev-pop)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev-pop)**

Returns the population standard deviation of ***expr*** (the square root of [**VAR\_POP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_var-pop)). You can also use [**STD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_std) or [**STDDEV()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev), which are equivalent but not standard SQL.

If there are no matching rows, [**STDDEV\_POP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev-pop) returns **NULL**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

**[STDDEV\_SAMP(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev-samp)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev-samp)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev-samp)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev-samp)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_stddev-samp)**

Returns the sample standard deviation of ***expr*** (the square root of [**VAR\_SAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_var-samp).

If there are no matching rows, [**STDDEV\_SAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_stddev-samp) returns **NULL**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

**[SUM([DISTINCT]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sum)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sum)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sum)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sum)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sum)**

Returns the sum of ***expr***. If the return set has no rows, [**SUM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sum) returns **NULL**. The **DISTINCT** keyword can be used to sum only the distinct values of ***expr***.

If there are no matching rows, [**SUM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sum) returns **NULL**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage); it cannot be used with **DISTINCT**.

**[VAR\_POP(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_var-pop)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_var-pop)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_var-pop)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_var-pop)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_var-pop)**

Returns the population standard variance of ***expr***. It considers rows as the whole population, not as a sample, so it has the number of rows as the denominator. You can also use [**VARIANCE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_variance), which is equivalent but is not standard SQL.

If there are no matching rows, [**VAR\_POP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_var-pop) returns **NULL**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

**[VAR\_SAMP(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_var-samp)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_var-samp)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_var-samp)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_var-samp)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_var-samp)**

Returns the sample variance of ***expr***. That is, the denominator is the number of rows minus one.

If there are no matching rows, [**VAR\_SAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_var-samp) returns **NULL**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

**[VARIANCE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_variance)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_variance)*[) [](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_variance)*[over\_clause](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_variance)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_variance)**

Returns the population standard variance of ***expr***. [**VARIANCE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_variance) is a synonym for the standard SQL function [**VAR\_POP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_var-pop), provided as a MySQL extension.

If there are no matching rows, [**VARIANCE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_variance) returns **NULL**.

This function executes as a window function if ***over\_clause*** is present. ***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

### 12.20.2 GROUP BY Modifiers

The **GROUP BY** clause permits a **WITH ROLLUP** modifier that causes summary output to include extra rows that represent higher-level (that is, super-aggregate) summary operations. **ROLLUP** thus enables you to answer questions at multiple levels of analysis with a single query. For example, **ROLLUP** can be used to provide support for OLAP (Online Analytical Processing) operations.

Suppose that a **sales** table has **year**, **country**, **product**, and **profit** columns for recording sales profitability:

CREATE TABLE sales

(

year INT,

country VARCHAR(20),

product VARCHAR(32),

profit INT

);

To summarize table contents per year, use a simple **GROUP BY** like this:

mysql> **SELECT year, SUM(profit) AS profit**

**FROM sales**

**GROUP BY year;**

+------+--------+

| year | profit |

+------+--------+

| 2000 | 4525 |

| 2001 | 3010 |

+------+--------+

The output shows the total (aggregate) profit for each year. To also determine the total profit summed over all years, you must add up the individual values yourself or run an additional query. Or you can use **ROLLUP**, which provides both levels of analysis with a single query. Adding a **WITH ROLLUP** modifier to the **GROUP BY** clause causes the query to produce another (super-aggregate) row that shows the grand total over all year values:

mysql> **SELECT year, SUM(profit) AS profit**

**FROM sales**

**GROUP BY year WITH ROLLUP;**

+------+--------+

| year | profit |

+------+--------+

| 2000 | 4525 |

| 2001 | 3010 |

| NULL | 7535 |

+------+--------+

The **NULL** value in the **year** column identifies the grand total super-aggregate line.

**ROLLUP** has a more complex effect when there are multiple **GROUP BY** columns. In this case, each time there is a change in value in any but the last grouping column, the query produces an extra super-aggregate summary row.

For example, without **ROLLUP**, a summary of the **sales** table based on **year**, **country**, and **product** might look like this, where the output indicates summary values only at the year/country/product level of analysis:

mysql> **SELECT year, country, product, SUM(profit) AS profit**

**FROM sales**

**GROUP BY year, country, product;**

+------+---------+------------+--------+

| year | country | product | profit |

+------+---------+------------+--------+

| 2000 | Finland | Computer | 1500 |

| 2000 | Finland | Phone | 100 |

| 2000 | India | Calculator | 150 |

| 2000 | India | Computer | 1200 |

| 2000 | USA | Calculator | 75 |

| 2000 | USA | Computer | 1500 |

| 2001 | Finland | Phone | 10 |

| 2001 | USA | Calculator | 50 |

| 2001 | USA | Computer | 2700 |

| 2001 | USA | TV | 250 |

+------+---------+------------+--------+

With **ROLLUP** added, the query produces several extra rows:

mysql> **SELECT year, country, product, SUM(profit) AS profit**

**FROM sales**

**GROUP BY year, country, product WITH ROLLUP;**

+------+---------+------------+--------+

| year | country | product | profit |

+------+---------+------------+--------+

| 2000 | Finland | Computer | 1500 |

| 2000 | Finland | Phone | 100 |

| 2000 | Finland | NULL | 1600 |

| 2000 | India | Calculator | 150 |

| 2000 | India | Computer | 1200 |

| 2000 | India | NULL | 1350 |

| 2000 | USA | Calculator | 75 |

| 2000 | USA | Computer | 1500 |

| 2000 | USA | NULL | 1575 |

| 2000 | NULL | NULL | 4525 |

| 2001 | Finland | Phone | 10 |

| 2001 | Finland | NULL | 10 |

| 2001 | USA | Calculator | 50 |

| 2001 | USA | Computer | 2700 |

| 2001 | USA | TV | 250 |

| 2001 | USA | NULL | 3000 |

| 2001 | NULL | NULL | 3010 |

| NULL | NULL | NULL | 7535 |

+------+---------+------------+--------+

Now the output includes summary information at four levels of analysis, not just one:

Following each set of product rows for a given year and country, an extra super-aggregate summary row appears showing the total for all products. These rows have the **product** column set to **NULL**.

Following each set of rows for a given year, an extra super-aggregate summary row appears showing the total for all countries and products. These rows have the **country** and **products** columns set to **NULL**.

Finally, following all other rows, an extra super-aggregate summary row appears showing the grand total for all years, countries, and products. This row has the **year**, **country**, and **products** columns set to **NULL**.

The **NULL** indicators in each super-aggregate row are produced when the row is sent to the client. The server looks at the columns named in the **GROUP BY** clause following the leftmost one that has changed value. For any column in the result set with a name that matches any of those names, its value is set to **NULL**. (If you specify grouping columns by column position, the server identifies which columns to set to **NULL** by position.)

Because the **NULL** values in the super-aggregate rows are placed into the result set at such a late stage in query processing, you can test them as **NULL** values only in the select list or **HAVING** clause. You cannot test them as **NULL** values in join conditions or the **WHERE** clause to determine which rows to select. For example, you cannot add **WHERE product IS NULL** to the query to eliminate from the output all but the super-aggregate rows.

The **NULL** values do appear as **NULL** on the client side and can be tested as such using any MySQL client programming interface. However, at this point, you cannot distinguish whether a **NULL** represents a regular grouped value or a super-aggregate value. To test the distinction, use the [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) function, described later.

Previously, MySQL did not allow the use of **DISTINCT** or **ORDER BY** in a query having a **WITH ROLLUP** option. This restriction is lifted in MySQL 8.0.12 and later. (Bug #87450, Bug #86311, Bug #26640100, Bug #26073513)

For **GROUP BY ... WITH ROLLUP** queries, to test whether **NULL** values in the result represent super-aggregate values, the [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) function is available for use in the select list, **HAVING** clause, and (as of MySQL 8.0.12) **ORDER BY** clause. For example, [**GROUPING(year)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) returns 1 when **NULL** in the **year** column occurs in a super-aggregate row, and 0 otherwise. Similarly, [**GROUPING(country)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) and [**GROUPING(product)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) return 1 for super-aggregate **NULL** values in the **country** and **product** columns, respectively:

mysql> **SELECT**

**year, country, product, SUM(profit) AS profit,**

**GROUPING(year) AS grp\_year,**

**GROUPING(country) AS grp\_country,**

**GROUPING(product) AS grp\_product**

**FROM sales**

**GROUP BY year, country, product WITH ROLLUP;**

+------+---------+------------+--------+----------+-------------+-------------+

| year | country | product | profit | grp\_year | grp\_country | grp\_product |

+------+---------+------------+--------+----------+-------------+-------------+

| 2000 | Finland | Computer | 1500 | 0 | 0 | 0 |

| 2000 | Finland | Phone | 100 | 0 | 0 | 0 |

| 2000 | Finland | NULL | 1600 | 0 | 0 | 1 |

| 2000 | India | Calculator | 150 | 0 | 0 | 0 |

| 2000 | India | Computer | 1200 | 0 | 0 | 0 |

| 2000 | India | NULL | 1350 | 0 | 0 | 1 |

| 2000 | USA | Calculator | 75 | 0 | 0 | 0 |

| 2000 | USA | Computer | 1500 | 0 | 0 | 0 |

| 2000 | USA | NULL | 1575 | 0 | 0 | 1 |

| 2000 | NULL | NULL | 4525 | 0 | 1 | 1 |

| 2001 | Finland | Phone | 10 | 0 | 0 | 0 |

| 2001 | Finland | NULL | 10 | 0 | 0 | 1 |

| 2001 | USA | Calculator | 50 | 0 | 0 | 0 |

| 2001 | USA | Computer | 2700 | 0 | 0 | 0 |

| 2001 | USA | TV | 250 | 0 | 0 | 0 |

| 2001 | USA | NULL | 3000 | 0 | 0 | 1 |

| 2001 | NULL | NULL | 3010 | 0 | 1 | 1 |

| NULL | NULL | NULL | 7535 | 1 | 1 | 1 |

+------+---------+------------+--------+----------+-------------+-------------+

Instead of displaying the [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) results directly, you can use [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) to substitute labels for super-aggregate **NULL** values:

mysql> **SELECT**

**IF(GROUPING(year), 'All years', year) AS year,**

**IF(GROUPING(country), 'All countries', country) AS country,**

**IF(GROUPING(product), 'All products', product) AS product,**

**SUM(profit) AS profit**

**FROM sales**

**GROUP BY year, country, product WITH ROLLUP;**

+-----------+---------------+--------------+--------+

| year | country | product | profit |

+-----------+---------------+--------------+--------+

| 2000 | Finland | Computer | 1500 |

| 2000 | Finland | Phone | 100 |

| 2000 | Finland | All products | 1600 |

| 2000 | India | Calculator | 150 |

| 2000 | India | Computer | 1200 |

| 2000 | India | All products | 1350 |

| 2000 | USA | Calculator | 75 |

| 2000 | USA | Computer | 1500 |

| 2000 | USA | All products | 1575 |

| 2000 | All countries | All products | 4525 |

| 2001 | Finland | Phone | 10 |

| 2001 | Finland | All products | 10 |

| 2001 | USA | Calculator | 50 |

| 2001 | USA | Computer | 2700 |

| 2001 | USA | TV | 250 |

| 2001 | USA | All products | 3000 |

| 2001 | All countries | All products | 3010 |

| All years | All countries | All products | 7535 |

+-----------+---------------+--------------+--------+

With multiple expression arguments, [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) returns a result representing a bitmask the combines the results for each expression, with the lowest-order bit corresponding to the result for the rightmost expression. For example, [**GROUPING(year, country, product)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) is evaluated like this:

result for GROUPING(***product***)

+ result for GROUPING(***country***) << 1

+ result for GROUPING(***year***) << 2

The result of such a [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) is nonzero if any of the expressions represents a super-aggregate **NULL**, so you can return only the super-aggregate rows and filter out the regular grouped rows like this:

mysql> **SELECT year, country, product, SUM(profit) AS profit**

**FROM sales**

**GROUP BY year, country, product WITH ROLLUP**

**HAVING GROUPING(year, country, product) <> 0;**

+------+---------+---------+--------+

| year | country | product | profit |

+------+---------+---------+--------+

| 2000 | Finland | NULL | 1600 |

| 2000 | India | NULL | 1350 |

| 2000 | USA | NULL | 1575 |

| 2000 | NULL | NULL | 4525 |

| 2001 | Finland | NULL | 10 |

| 2001 | USA | NULL | 3000 |

| 2001 | NULL | NULL | 3010 |

| NULL | NULL | NULL | 7535 |

+------+---------+---------+--------+

The **sales** table contains no **NULL** values, so all **NULL** values in a **ROLLUP** result represent super-aggregate values. When the data set contains **NULL** values, **ROLLUP** summaries may contain **NULL** values not only in super-aggregate rows, but also in regular grouped rows. [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) enables these to be distinguished. Suppose that table **t1** contains a simple data set with two grouping factors for a set of quantity values, where **NULL** indicates something like “other” or “unknown”:

mysql> **SELECT \* FROM t1;**

+------+-------+----------+

| name | size | quantity |

+------+-------+----------+

| ball | small | 10 |

| ball | large | 20 |

| ball | NULL | 5 |

| hoop | small | 15 |

| hoop | large | 5 |

| hoop | NULL | 3 |

+------+-------+----------+

A simple **ROLLUP** operation produces these results, in which it is not so easy to distinguish **NULL** values in super-aggregate rows from **NULL** values in regular grouped rows:

mysql> **SELECT name, size, SUM(quantity) AS quantity**

**FROM t1**

**GROUP BY name, size WITH ROLLUP;**

+------+-------+----------+

| name | size | quantity |

+------+-------+----------+

| ball | NULL | 5 |

| ball | large | 20 |

| ball | small | 10 |

| ball | NULL | 35 |

| hoop | NULL | 3 |

| hoop | large | 5 |

| hoop | small | 15 |

| hoop | NULL | 23 |

| NULL | NULL | 58 |

+------+-------+----------+

Using [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) to substitute labels for the super-aggregate **NULL** values makes the result easier to interpret:

mysql> **SELECT**

**IF(GROUPING(name) = 1, 'All items', name) AS name,**

**IF(GROUPING(size) = 1, 'All sizes', size) AS size,**

**SUM(quantity) AS quantity**

**FROM t1**

**GROUP BY name, size WITH ROLLUP;**

+-----------+-----------+----------+

| name | size | quantity |

+-----------+-----------+----------+

| ball | NULL | 5 |

| ball | large | 20 |

| ball | small | 10 |

| ball | All sizes | 35 |

| hoop | NULL | 3 |

| hoop | large | 5 |

| hoop | small | 15 |

| hoop | All sizes | 23 |

| All items | All sizes | 58 |

+-----------+-----------+----------+

#### Other Considerations When using ROLLUP

The following discussion lists some behaviors specific to the MySQL implementation of **ROLLUP**.

Prior to MySQL 8.0.12, when you use **ROLLUP**, you cannot also use an **ORDER BY** clause to sort the results. In other words, **ROLLUP** and **ORDER BY** were mutually exclusive in MySQL. However, you still have some control over sort order. To work around the restriction that prevents using **ROLLUP** with **ORDER BY** and achieve a specific sort order of grouped results, generate the grouped result set as a derived table and apply **ORDER BY** to it. For example:

mysql> **SELECT \* FROM**

**(SELECT year, SUM(profit) AS profit**

**FROM sales GROUP BY year WITH ROLLUP) AS dt**

**ORDER BY year DESC;**

+------+--------+

| year | profit |

+------+--------+

| 2001 | 3010 |

| 2000 | 4525 |

| NULL | 7535 |

+------+--------+

As of MySQL 8.0.12, **ORDER BY** and **ROLLUP** can be used together, which enables the use of **ORDER BY** and [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) to achieve a specific sort order of grouped results. For example:

mysql> **SELECT year, SUM(profit) AS profit**

**FROM sales**

**GROUP BY year WITH ROLLUP**

**ORDER BY GROUPING(year) DESC;**

+------+--------+

| year | profit |

+------+--------+

| NULL | 7535 |

| 2000 | 4525 |

| 2001 | 3010 |

+------+--------+

In both cases, the super-aggregate summary rows sort with the rows from which they are calculated, and their placement depends on sort order (at the end for ascending sort, at the beginning for descending sort).

**LIMIT** can be used to restrict the number of rows returned to the client. **LIMIT** is applied after **ROLLUP**, so the limit applies against the extra rows added by **ROLLUP**. For example:

mysql> **SELECT year, country, product, SUM(profit) AS profit**

**FROM sales**

**GROUP BY year, country, product WITH ROLLUP**

**LIMIT 5;**

+------+---------+------------+--------+

| year | country | product | profit |

+------+---------+------------+--------+

| 2000 | Finland | Computer | 1500 |

| 2000 | Finland | Phone | 100 |

| 2000 | Finland | NULL | 1600 |

| 2000 | India | Calculator | 150 |

| 2000 | India | Computer | 1200 |

+------+---------+------------+--------+

Using **LIMIT** with **ROLLUP** may produce results that are more difficult to interpret, because there is less context for understanding the super-aggregate rows.

A MySQL extension permits a column that does not appear in the **GROUP BY** list to be named in the select list. (For information about nonaggregated columns and **GROUP BY**, see [Section 12.20.3, “MySQL Handling of GROUP BY”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#group-by-handling).) In this case, the server is free to choose any value from this nonaggregated column in summary rows, and this includes the extra rows added by **WITH ROLLUP**. For example, in the following query, **country** is a nonaggregated column that does not appear in the **GROUP BY** list and values chosen for this column are nondeterministic:

mysql> **SELECT year, country, SUM(profit) AS profit**

**FROM sales**

**GROUP BY year WITH ROLLUP;**

+------+---------+--------+

| year | country | profit |

+------+---------+--------+

| 2000 | India | 4525 |

| 2001 | USA | 3010 |

| NULL | USA | 7535 |

+------+---------+--------+

This behavior is permitted when the [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) SQL mode is not enabled. If that mode is enabled, the server rejects the query as illegal because **country** is not listed in the **GROUP BY** clause. With [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) enabled, you can still execute the query by using the [**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) function for nondeterministic-value columns:

mysql> **SELECT year, ANY\_VALUE(country) AS country, SUM(profit) AS profit**

**FROM sales**

**GROUP BY year WITH ROLLUP;**

+------+---------+--------+

| year | country | profit |

+------+---------+--------+

| 2000 | India | 4525 |

| 2001 | USA | 3010 |

| NULL | USA | 7535 |

+------+---------+--------+

### 12.20.3 MySQL Handling of GROUP BY

SQL-92 and earlier does not permit queries for which the select list, **HAVING** condition, or **ORDER BY** list refer to nonaggregated columns that are not named in the **GROUP BY** clause. For example, this query is illegal in standard SQL-92 because the nonaggregated **name** column in the select list does not appear in the **GROUP BY**:

SELECT o.custid, c.name, MAX(o.payment)

FROM orders AS o, customers AS c

WHERE o.custid = c.custid

GROUP BY o.custid;

For the query to be legal in SQL-92, the **name** column must be omitted from the select list or named in the **GROUP BY** clause.

SQL:1999 and later permits such nonaggregates per optional feature T301 if they are functionally dependent on **GROUP BY** columns: If such a relationship exists between **name** and **custid**, the query is legal. This would be the case, for example, were **custid** a primary key of **customers**.

MySQL implements detection of functional dependence. If the [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) SQL mode is enabled (which it is by default), MySQL rejects queries for which the select list, **HAVING** condition, or **ORDER BY** list refer to nonaggregated columns that are neither named in the **GROUP BY** clause nor are functionally dependent on them.

MySQL also permits a nonaggregate column not named in a **GROUP BY** clause when SQL [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) mode is enabled, provided that this column is limited to a single value, as shown in the following example:

mysql> **CREATE TABLE mytable (**

-> **id INT UNSIGNED NOT NULL PRIMARY KEY,**

-> **a VARCHAR(10),**

-> **b INT**

-> **);**

mysql> **INSERT INTO mytable**

-> **VALUES (1, 'abc', 1000),**

-> **(2, 'abc', 2000),**

-> **(3, 'def', 4000);**

mysql> **SET SESSION sql\_mode = sys.list\_add(@@session.sql\_mode, 'ONLY\_FULL\_GROUP\_BY');**

mysql> **SELECT a, SUM(b) FROM mytable WHERE a = 'abc';**

+------+--------+

| a | SUM(b) |

+------+--------+

| abc | 3000 |

+------+--------+

It is also possible to have more than one nonaggregate column in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) list when employing [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by). In this case, every such column must be limited to a single value in the **WHERE** clause, and all such limiting conditions must be joined by logical **AND**, as shown here:

mysql> **DROP TABLE IF EXISTS mytable;**

mysql> **CREATE TABLE mytable (**

-> **id INT UNSIGNED NOT NULL PRIMARY KEY,**

-> **a VARCHAR(10),**

-> **b VARCHAR(10),**

-> **c INT**

-> **);**

mysql> **INSERT INTO mytable**

-> **VALUES (1, 'abc', 'qrs', 1000),**

-> **(2, 'abc', 'tuv', 2000),**

-> **(3, 'def', 'qrs', 4000),**

-> **(4, 'def', 'tuv', 8000),**

-> **(5, 'abc', 'qrs', 16000),**

-> **(6, 'def', 'tuv', 32000);**

mysql> **SELECT @@session.sql\_mode;**

+---------------------------------------------------------------+

| @@session.sql\_mode |

+---------------------------------------------------------------+

| ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ENGINE\_SUBSTITUTION |

+---------------------------------------------------------------+

mysql> **SELECT a, b, SUM(c) FROM mytable**

-> **WHERE a = 'abc' AND b = 'qrs';**

+------+------+--------+

| a | b | SUM(c) |

+------+------+--------+

| abc | qrs | 17000 |

+------+------+--------+

If [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) is disabled, a MySQL extension to the standard SQL use of **GROUP BY** permits the select list, **HAVING** condition, or **ORDER BY** list to refer to nonaggregated columns even if the columns are not functionally dependent on **GROUP BY** columns. This causes MySQL to accept the preceding query. In this case, the server is free to choose any value from each group, so unless they are the same, the values chosen are nondeterministic, which is probably not what you want. Furthermore, the selection of values from each group cannot be influenced by adding an **ORDER BY** clause. Result set sorting occurs after values have been chosen, and **ORDER BY** does not affect which value within each group the server chooses. Disabling [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) is useful primarily when you know that, due to some property of the data, all values in each nonaggregated column not named in the **GROUP BY** are the same for each group.

You can achieve the same effect without disabling [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) by using [**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) to refer to the nonaggregated column.

The following discussion demonstrates functional dependence, the error message MySQL produces when functional dependence is absent, and ways of causing MySQL to accept a query in the absence of functional dependence.

This query might be invalid with [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) enabled because the nonaggregated **address** column in the select list is not named in the **GROUP BY** clause:

SELECT name, address, MAX(age) FROM t GROUP BY name;

The query is valid if **name** is a primary key of **t** or is a unique **NOT NULL** column. In such cases, MySQL recognizes that the selected column is functionally dependent on a grouping column. For example, if **name** is a primary key, its value determines the value of **address** because each group has only one value of the primary key and thus only one row. As a result, there is no randomness in the choice of **address** value in a group and no need to reject the query.

The query is invalid if **name** is not a primary key of **t** or a unique **NOT NULL** column. In this case, no functional dependency can be inferred and an error occurs:

mysql> **SELECT name, address, MAX(age) FROM t GROUP BY name;**

ERROR 1055 (42000): Expression #2 of SELECT list is not in GROUP

BY clause and contains nonaggregated column 'mydb.t.address' which

is not functionally dependent on columns in GROUP BY clause; this

is incompatible with sql\_mode=only\_full\_group\_by

If you know that, for a given data set, each **name** value in fact uniquely determines the **address** value, **address** is effectively functionally dependent on **name**. To tell MySQL to accept the query, you can use the [**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) function:

SELECT name, ANY\_VALUE(address), MAX(age) FROM t GROUP BY name;

Alternatively, disable [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by).

The preceding example is quite simple, however. In particular, it is unlikely you would group on a single primary key column because every group would contain only one row. For addtional examples demonstrating functional dependence in more complex queries, see [Section 12.20.4, “Detection of Functional Dependence”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#group-by-functional-dependence).

If a query has aggregate functions and no **GROUP BY** clause, it cannot have nonaggregated columns in the select list, **HAVING** condition, or **ORDER BY** list with [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) enabled:

mysql> **SELECT name, MAX(age) FROM t;**

ERROR 1140 (42000): In aggregated query without GROUP BY, expression

#1 of SELECT list contains nonaggregated column 'mydb.t.name'; this

is incompatible with sql\_mode=only\_full\_group\_by

Without **GROUP BY**, there is a single group and it is nondeterministic which **name** value to choose for the group. Here, too, [**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) can be used, if it is immaterial which **name** value MySQL chooses:

SELECT ANY\_VALUE(name), MAX(age) FROM t;

**ONLY\_FULL\_GROUP\_BY** also affects handling of queries that use **DISTINCT** and **ORDER BY**. Consider the case of a table **t** with three columns **c1**, **c2**, and **c3** that contains these rows:

c1 c2 c3

1 2 A

3 4 B

1 2 C

Suppose that we execute the following query, expecting the results to be ordered by **c3**:

SELECT DISTINCT c1, c2 FROM t ORDER BY c3;

To order the result, duplicates must be eliminated first. But to do so, should we keep the first row or the third? This arbitrary choice influences the retained value of **c3**, which in turn influences ordering and makes it arbitrary as well. To prevent this problem, a query that has **DISTINCT** and **ORDER BY** is rejected as invalid if any **ORDER BY** expression does not satisfy at least one of these conditions:

The expression is equal to one in the select list

All columns referenced by the expression and belonging to the query's selected tables are elements of the select list

Another MySQL extension to standard SQL permits references in the **HAVING** clause to aliased expressions in the select list. For example, the following query returns **name** values that occur only once in table **orders**:

SELECT name, COUNT(name) FROM orders

GROUP BY name

HAVING COUNT(name) = 1;

The MySQL extension permits the use of an alias in the **HAVING** clause for the aggregated column:

SELECT name, COUNT(name) AS c FROM orders

GROUP BY name

HAVING c = 1;

Standard SQL permits only column expressions in **GROUP BY** clauses, so a statement such as this is invalid because **FLOOR(value/100)** is a noncolumn expression:

SELECT id, FLOOR(value/100)

FROM ***tbl\_name***

GROUP BY id, FLOOR(value/100);

MySQL extends standard SQL to permit noncolumn expressions in **GROUP BY** clauses and considers the preceding statement valid.

Standard SQL also does not permit aliases in **GROUP BY** clauses. MySQL extends standard SQL to permit aliases, so another way to write the query is as follows:

SELECT id, FLOOR(value/100) AS val

FROM ***tbl\_name***

GROUP BY id, val;

The alias **val** is considered a column expression in the **GROUP BY** clause.

In the presence of a noncolumn expression in the **GROUP BY** clause, MySQL recognizes equality between that expression and expressions in the select list. This means that with [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) SQL mode enabled, the query containing **GROUP BY id, FLOOR(value/100)** is valid because that same [**FLOOR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_floor) expression occurs in the select list. However, MySQL does not try to recognize functional dependence on **GROUP BY** noncolumn expressions, so the following query is invalid with [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) enabled, even though the third selected expression is a simple formula of the **id** column and the [**FLOOR()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_floor) expression in the **GROUP BY** clause:

SELECT id, FLOOR(value/100), id+FLOOR(value/100)

FROM ***tbl\_name***

GROUP BY id, FLOOR(value/100);

A workaround is to use a derived table:

SELECT id, F, id+F

FROM

(SELECT id, FLOOR(value/100) AS F

FROM ***tbl\_name***

GROUP BY id, FLOOR(value/100)) AS dt;

### 12.20.4 Detection of Functional Dependence

The following discussion provides several examples of the ways in which MySQL detects functional dependencies. The examples use this notation:

{***X***} -> {***Y***}

Understand this as “***X*** uniquely determines ***Y***,” which also means that ***Y*** is functionally dependent on ***X***.

The examples use the **world** database, which can be downloaded from <https://dev.mysql.com/doc/index-other.html>. You can find details on how to install the database on the same page.

[Functional Dependencies Derived from Keys](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#functional-dependence-keys)

[Functional Dependencies Derived from Multiple-Column Keys and from Equalities](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#functional-dependence-multiple-column-keys)

[Functional Dependency Special Cases](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#functional-dependence-special-cases)

[Functional Dependencies and Views](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#functional-dependence-views)

[Combinations of Functional Dependencies](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#functional-dependence-combinations)

#### Functional Dependencies Derived from Keys

The following query selects, for each country, a count of spoken languages:

SELECT co.Name, COUNT(\*)

FROM countrylanguage cl, country co

WHERE cl.CountryCode = co.Code

GROUP BY co.Code;

**co.Code** is a primary key of **co**, so all columns of **co** are functionally dependent on it, as expressed using this notation:

{co.Code} -> {co.\*}

Thus, **co.name** is functionally dependent on **GROUP BY** columns and the query is valid.

A **UNIQUE** index over a **NOT NULL** column could be used instead of a primary key and the same functional dependence would apply. (This is not true for a **UNIQUE** index that permits **NULL** values because it permits multiple **NULL** values and in that case uniqueness is lost.)

#### Functional Dependencies Derived from Multiple-Column Keys and from Equalities

This query selects, for each country, a list of all spoken languages and how many people speak them:

SELECT co.Name, cl.Language,

cl.Percentage \* co.Population / 100.0 AS SpokenBy

FROM countrylanguage cl, country co

WHERE cl.CountryCode = co.Code

GROUP BY cl.CountryCode, cl.Language;

The pair (**cl.CountryCode**, **cl.Language**) is a two-column composite primary key of **cl**, so that column pair uniquely determines all columns of **cl**:

{cl.CountryCode, cl.Language} -> {cl.\*}

Moreover, because of the equality in the **WHERE** clause:

{cl.CountryCode} -> {co.Code}

And, because **co.Code** is primary key of **co**:

{co.Code} -> {co.\*}

“Uniquely determines” relationships are transitive, therefore:

{cl.CountryCode, cl.Language} -> {cl.\*,co.\*}

As a result, the query is valid.

As with the previous example, a **UNIQUE** key over **NOT NULL** columns could be used instead of a primary key.

An **INNER JOIN** condition can be used instead of **WHERE**. The same functional dependencies apply:

SELECT co.Name, cl.Language,

cl.Percentage \* co.Population/100.0 AS SpokenBy

FROM countrylanguage cl INNER JOIN country co

ON cl.CountryCode = co.Code

GROUP BY cl.CountryCode, cl.Language;

#### Functional Dependency Special Cases

Whereas an equality test in a **WHERE** condition or **INNER JOIN** condition is symmetric, an equality test in an outer join condition is not, because tables play different roles.

Assume that referential integrity has been accidentally broken and there exists a row of **countrylanguage** without a corresponding row in **country**. Consider the same query as in the previous example, but with a **LEFT JOIN**:

SELECT co.Name, cl.Language,

cl.Percentage \* co.Population/100.0 AS SpokenBy

FROM countrylanguage cl LEFT JOIN country co

ON cl.CountryCode = co.Code

GROUP BY cl.CountryCode, cl.Language;

For a given value of **cl.CountryCode**, the value of **co.Code** in the join result is either found in a matching row (determined by **cl.CountryCode**) or is **NULL**-complemented if there is no match (also determined by **cl.CountryCode**). In each case, this relationship applies:

{cl.CountryCode} -> {co.Code}

**cl.CountryCode** is itself functionally dependent on {**cl.CountryCode**, **cl.Language**} which is a primary key.

If in the join result **co.Code** is **NULL**-complemented, **co.Name** is as well. If **co.Code** is not **NULL**-complemented, then because **co.Code** is a primary key, it determines **co.Name**. Therefore, in all cases:

{co.Code} -> {co.Name}

Which yields:

{cl.CountryCode, cl.Language} -> {cl.\*,co.\*}

As a result, the query is valid.

However, suppose that the tables are swapped, as in this query:

SELECT co.Name, cl.Language,

cl.Percentage \* co.Population/100.0 AS SpokenBy

FROM country co LEFT JOIN countrylanguage cl

ON cl.CountryCode = co.Code

GROUP BY cl.CountryCode, cl.Language;

Now this relationship does not apply:

{cl.CountryCode, cl.Language} -> {cl.\*,co.\*}

Indeed, all **NULL**-complemented rows made for **cl** is put into a single group (they have both **GROUP BY** columns equal to **NULL**), and inside this group the value of **co.Name** can vary. The query is invalid and MySQL rejects it.

Functional dependence in outer joins is thus linked to whether determinant columns belong to the left or right side of the **LEFT JOIN**. Determination of functional dependence becomes more complex if there are nested outer joins or the join condition does not consist entirely of equality comparisons.

#### Functional Dependencies and Views

Suppose that a view on countries produces their code, their name in uppercase, and how many different official languages they have:

CREATE VIEW country2 AS

SELECT co.Code, UPPER(co.Name) AS UpperName,

COUNT(cl.Language) AS OfficialLanguages

FROM country AS co JOIN countrylanguage AS cl

ON cl.CountryCode = co.Code

WHERE cl.isOfficial = 'T'

GROUP BY co.Code;

This definition is valid because:

{co.Code} -> {co.\*}

In the view result, the first selected column is **co.Code**, which is also the group column and thus determines all other selected expressions:

{country2.Code} -> {country2.\*}

MySQL understands this and uses this information, as described following.

This query displays countries, how many different official languages they have, and how many cities they have, by joining the view with the **city** table:

SELECT co2.Code, co2.UpperName, co2.OfficialLanguages,

COUNT(\*) AS Cities

FROM country2 AS co2 JOIN city ci

ON ci.CountryCode = co2.Code

GROUP BY co2.Code;

This query is valid because, as seen previously:

{co2.Code} -> {co2.\*}

MySQL is able to discover a functional dependency in the result of a view and use that to validate a query which uses the view. The same would be true if **country2** were a derived table (or common table expression), as in:

SELECT co2.Code, co2.UpperName, co2.OfficialLanguages,

COUNT(\*) AS Cities

FROM

(

SELECT co.Code, UPPER(co.Name) AS UpperName,

COUNT(cl.Language) AS OfficialLanguages

FROM country AS co JOIN countrylanguage AS cl

ON cl.CountryCode=co.Code

WHERE cl.isOfficial='T'

GROUP BY co.Code

) AS co2

JOIN city ci ON ci.CountryCode = co2.Code

GROUP BY co2.Code;

#### Combinations of Functional Dependencies

MySQL is able to combine all of the preceding types of functional dependencies (key based, equality based, view based) to validate more complex queries.

## 12.21 Window Functions

[12.21.1 Window Function Descriptions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-function-descriptions)

[12.21.2 Window Function Concepts and Syntax](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage)

[12.21.3 Window Function Frame Specification](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-frames)

[12.21.4 Named Windows](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-named-windows)

[12.21.5 Window Function Restrictions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-function-restrictions)

MySQL supports window functions that, for each row from a query, perform a calculation using rows related to that row. The following sections discuss how to use window functions, including descriptions of the **OVER** and **WINDOW** clauses. The first section provides descriptions of the nonaggregate window functions. For descriptions of the aggregate window functions, see [Section 12.20.1, “Aggregate Function Descriptions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#aggregate-functions).

For information about optimization and window functions, see [Section 8.2.1.21, “Window Function Optimization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#window-function-optimization).

### 12.21.1 Window Function Descriptions

This section describes nonaggregate window functions that, for each row from a query, perform a calculation using rows related to that row. Most aggregate functions also can be used as window functions; see [Section 12.20.1, “Aggregate Function Descriptions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#aggregate-functions).

For window function usage information and examples, and definitions of terms such as the **OVER** clause, window, partition, frame, and peer, see [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

**Table 12.26 Window Functions**

| **Name** | **Description** |
| --- | --- |
| [**CUME\_DIST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cume-dist) | Cumulative distribution value |
| [**DENSE\_RANK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dense-rank) | Rank of current row within its partition, without gaps |
| [**FIRST\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_first-value) | Value of argument from first row of window frame |
| [**LAG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lag) | Value of argument from row lagging current row within partition |
| [**LAST\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-value) | Value of argument from last row of window frame |
| [**LEAD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lead) | Value of argument from row leading current row within partition |
| [**NTH\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_nth-value) | Value of argument from N-th row of window frame |
| [**NTILE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ntile) | Bucket number of current row within its partition. |
| [**PERCENT\_RANK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_percent-rank) | Percentage rank value |
| [**RANK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rank) | Rank of current row within its partition, with gaps |
| [**ROW\_NUMBER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-number) | Number of current row within its partition |

In the following function descriptions, ***over\_clause*** represents the **OVER** clause, described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage). Some window functions permit a ***null\_treatment*** clause that specifies how to handle **NULL** values when calculating results. This clause is optional. It is part of the SQL standard, but the MySQL implementation permits only **RESPECT NULLS** (which is also the default). This means that **NULL** values are considered when calculating results. **IGNORE NULLS** is parsed, but produces an error.

**[CUME\_DIST()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_cume-dist)** ***over\_clause***

Returns the cumulative distribution of a value within a group of values; that is, the percentage of partition values less than or equal to the value in the current row. This represents the number of rows preceding or peer with the current row in the window ordering of the window partition divided by the total number of rows in the window partition. Return values range from 0 to 1.

This function should be used with **ORDER BY** to sort partition rows into the desired order. Without **ORDER BY**, all rows are peers and have value ***N***/***N*** = 1, where ***N*** is the partition size.

***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

The following query shows, for the set of values in the **val** column, the **CUME\_DIST()** value for each row, as well as the percentage rank value returned by the similar **PERCENT\_RANK()** function. For reference, the query also displays row numbers using [**ROW\_NUMBER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-number):

mysql> **SELECT**

**val,**

**ROW\_NUMBER() OVER w AS 'row\_number',**

**CUME\_DIST() OVER w AS 'cume\_dist',**

**PERCENT\_RANK() OVER w AS 'percent\_rank'**

**FROM numbers**

**WINDOW w AS (ORDER BY val);**

+------+------------+--------------------+--------------+

| val | row\_number | cume\_dist | percent\_rank |

+------+------------+--------------------+--------------+

| 1 | 1 | 0.2222222222222222 | 0 |

| 1 | 2 | 0.2222222222222222 | 0 |

| 2 | 3 | 0.3333333333333333 | 0.25 |

| 3 | 4 | 0.6666666666666666 | 0.375 |

| 3 | 5 | 0.6666666666666666 | 0.375 |

| 3 | 6 | 0.6666666666666666 | 0.375 |

| 4 | 7 | 0.8888888888888888 | 0.75 |

| 4 | 8 | 0.8888888888888888 | 0.75 |

| 5 | 9 | 1 | 1 |

+------+------------+--------------------+--------------+

**[DENSE\_RANK()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_dense-rank)** ***over\_clause***

Returns the rank of the current row within its partition, without gaps. Peers are considered ties and receive the same rank. This function assigns consecutive ranks to peer groups; the result is that groups of size greater than one do not produce noncontiguous rank numbers. For an example, see the [**RANK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rank) function description.

This function should be used with **ORDER BY** to sort partition rows into the desired order. Without **ORDER BY**, all rows are peers.

***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

**[FIRST\_VALUE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_first-value)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_first-value)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_first-value)** [***null\_treatment***] ***over\_clause***

Returns the value of ***expr*** from the first row of the window frame.

***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage). ***null\_treatment*** is as described in the section introduction.

The following query demonstrates [**FIRST\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_first-value), [**LAST\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-value), and two instances of [**NTH\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_nth-value):

mysql> **SELECT**

**time, subject, val,**

**FIRST\_VALUE(val) OVER w AS 'first',**

**LAST\_VALUE(val) OVER w AS 'last',**

**NTH\_VALUE(val, 2) OVER w AS 'second',**

**NTH\_VALUE(val, 4) OVER w AS 'fourth'**

**FROM observations**

**WINDOW w AS (PARTITION BY subject ORDER BY time**

**ROWS UNBOUNDED PRECEDING);**

+----------+---------+------+-------+------+--------+--------+

| time | subject | val | first | last | second | fourth |

+----------+---------+------+-------+------+--------+--------+

| 07:00:00 | st113 | 10 | 10 | 10 | NULL | NULL |

| 07:15:00 | st113 | 9 | 10 | 9 | 9 | NULL |

| 07:30:00 | st113 | 25 | 10 | 25 | 9 | NULL |

| 07:45:00 | st113 | 20 | 10 | 20 | 9 | 20 |

| 07:00:00 | xh458 | 0 | 0 | 0 | NULL | NULL |

| 07:15:00 | xh458 | 10 | 0 | 10 | 10 | NULL |

| 07:30:00 | xh458 | 5 | 0 | 5 | 10 | NULL |

| 07:45:00 | xh458 | 30 | 0 | 30 | 10 | 30 |

| 08:00:00 | xh458 | 25 | 0 | 25 | 10 | 30 |

+----------+---------+------+-------+------+--------+--------+

Each function uses the rows in the current frame, which, per the window definition shown, extends from the first partition row to the current row. For the [**NTH\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_nth-value) calls, the current frame does not always include the requested row; in such cases, the return value is **NULL**.

**[LAG(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lag)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lag)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lag)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lag)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lag)*[default](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lag)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lag)** [***null\_treatment***] ***over\_clause***

Returns the value of ***expr*** from the row that lags (precedes) the current row by ***N*** rows within its partition. If there is no such row, the return value is ***default***. For example, if ***N*** is 3, the return value is ***default*** for the first two rows. If ***N*** or ***default*** are missing, the defaults are 1 and **NULL**, respectively.

***N*** must be a literal nonnegative integer. If ***N*** is 0, ***expr*** is evaluated for the current row.

Beginning with MySQL 8.0.22, ***N*** cannot be **NULL**. In addition, it must now be an integer in the range **1** to **263**, inclusive, in any of the following forms:

an unsigned integer constant literal

a positional parameter marker (**?**)

a user-defined variable

a local variable in a stored routine

***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage). ***null\_treatment*** is as described in the section introduction.

[**LAG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lag) (and the similar [**LEAD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lead) function) are often used to compute differences between rows. The following query shows a set of time-ordered observations and, for each one, the [**LAG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lag) and [**LEAD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lead) values from the adjoining rows, as well as the differences between the current and adjoining rows:

mysql> **SELECT**

**t, val,**

**LAG(val) OVER w AS 'lag',**

**LEAD(val) OVER w AS 'lead',**

**val - LAG(val) OVER w AS 'lag diff',**

**val - LEAD(val) OVER w AS 'lead diff'**

**FROM series**

**WINDOW w AS (ORDER BY t);**

+----------+------+------+------+----------+-----------+

| t | val | lag | lead | lag diff | lead diff |

+----------+------+------+------+----------+-----------+

| 12:00:00 | 100 | NULL | 125 | NULL | -25 |

| 13:00:00 | 125 | 100 | 132 | 25 | -7 |

| 14:00:00 | 132 | 125 | 145 | 7 | -13 |

| 15:00:00 | 145 | 132 | 140 | 13 | 5 |

| 16:00:00 | 140 | 145 | 150 | -5 | -10 |

| 17:00:00 | 150 | 140 | 200 | 10 | -50 |

| 18:00:00 | 200 | 150 | NULL | 50 | NULL |

+----------+------+------+------+----------+-----------+

In the example, the [**LAG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lag) and [**LEAD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lead) calls use the default ***N*** and ***default*** values of 1 and **NULL**, respectively.

The first row shows what happens when there is no previous row for [**LAG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lag): The function returns the ***default*** value (in this case, **NULL**). The last row shows the same thing when there is no next row for [**LEAD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lead).

[**LAG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lag) and [**LEAD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lead) also serve to compute sums rather than differences. Consider this data set, which contains the first few numbers of the Fibonacci series:

mysql> **SELECT n FROM fib ORDER BY n;**

+------+

| n |

+------+

| 1 |

| 1 |

| 2 |

| 3 |

| 5 |

| 8 |

+------+

The following query shows the [**LAG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lag) and [**LEAD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lead) values for the rows adjacent to the current row. It also uses those functions to add to the current row value the values from the preceding and following rows. The effect is to generate the next number in the Fibonacci series, and the next number after that:

mysql> **SELECT**

**n,**

**LAG(n, 1, 0) OVER w AS 'lag',**

**LEAD(n, 1, 0) OVER w AS 'lead',**

**n + LAG(n, 1, 0) OVER w AS 'next\_n',**

**n + LEAD(n, 1, 0) OVER w AS 'next\_next\_n'**

**FROM fib**

**WINDOW w AS (ORDER BY n);**

+------+------+------+--------+-------------+

| n | lag | lead | next\_n | next\_next\_n |

+------+------+------+--------+-------------+

| 1 | 0 | 1 | 1 | 2 |

| 1 | 1 | 2 | 2 | 3 |

| 2 | 1 | 3 | 3 | 5 |

| 3 | 2 | 5 | 5 | 8 |

| 5 | 3 | 8 | 8 | 13 |

| 8 | 5 | 0 | 13 | 8 |

+------+------+------+--------+-------------+

One way to generate the initial set of Fibonacci numbers is to use a recursive common table expression. For an example, see [Fibonacci Series Generation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#common-table-expressions-recursive-fibonacci-series).

Beginning with MySQL 8.0.22, you cannot use a negative value for the rows argument of this function.

**[LAST\_VALUE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_last-value)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_last-value)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_last-value)** [***null\_treatment***] ***over\_clause***

Returns the value of ***expr*** from the last row of the window frame.

***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage). ***null\_treatment*** is as described in the section introduction.

For an example, see the [**FIRST\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_first-value) function description.

**[LEAD(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lead)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lead)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lead)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lead)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lead)*[default](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lead)*[]])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_lead)** [***null\_treatment***] ***over\_clause***

Returns the value of ***expr*** from the row that leads (follows) the current row by ***N*** rows within its partition. If there is no such row, the return value is ***default***. For example, if ***N*** is 3, the return value is ***default*** for the last two rows. If ***N*** or ***default*** are missing, the defaults are 1 and **NULL**, respectively.

***N*** must be a literal nonnegative integer. If ***N*** is 0, ***expr*** is evaluated for the current row.

Beginning with MySQL 8.0.22, ***N*** cannot be **NULL**. In addition, it must now be an integer in the range **1** to **263**, inclusive, in any of the following forms:

an unsigned integer constant literal

a positional parameter marker (**?**)

a user-defined variable

a local variable in a stored routine

***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage). ***null\_treatment*** is as described in the section introduction.

For an example, see the [**LAG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lag) function description.

In MySQL 8.0.22 and later, use of a negative value for the rows argument of this function is not permitted.

**[NTH\_VALUE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_nth-value)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_nth-value)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_nth-value)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_nth-value)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_nth-value)** [***from\_first\_last***] [***null\_treatment***] ***over\_clause***

Returns the value of ***expr*** from the ***N***-th row of the window frame. If there is no such row, the return value is **NULL**.

***N*** must be a literal positive integer.

***from\_first\_last*** is part of the SQL standard, but the MySQL implementation permits only **FROM FIRST** (which is also the default). This means that calculations begin at the first row of the window. **FROM LAST** is parsed, but produces an error. To obtain the same effect as **FROM LAST** (begin calculations at the last row of the window), use **ORDER BY** to sort in reverse order.

***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage). ***null\_treatment*** is as described in the section introduction.

For an example, see the [**FIRST\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_first-value) function description.

In MySQL 8.0.22 and later, you cannot use **NULL** for the row argument of this function.

**[NTILE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ntile)*[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ntile)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ntile)** ***over\_clause***

Divides a partition into ***N*** groups (buckets), assigns each row in the partition its bucket number, and returns the bucket number of the current row within its partition. For example, if ***N*** is 4, **NTILE()** divides rows into four buckets. If ***N*** is 100, **NTILE()** divides rows into 100 buckets.

***N*** must be a literal positive integer. Bucket number return values range from 1 to ***N***.

Beginning with MySQL 8.0.22, ***N*** cannot be **NULL**. In addition, it must be an integer in the range **1** to **263**, inclusive, in any of the following forms:

an unsigned integer constant literal

a positional parameter marker (**?**)

a user-defined variable

a local variable in a stored routine

This function should be used with **ORDER BY** to sort partition rows into the desired order.

***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

The following query shows, for the set of values in the **val** column, the percentile values resulting from dividing the rows into two or four groups. For reference, the query also displays row numbers using [**ROW\_NUMBER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-number):

mysql> **SELECT**

**val,**

**ROW\_NUMBER() OVER w AS 'row\_number',**

**NTILE(2) OVER w AS 'ntile2',**

**NTILE(4) OVER w AS 'ntile4'**

**FROM numbers**

**WINDOW w AS (ORDER BY val);**

+------+------------+--------+--------+

| val | row\_number | ntile2 | ntile4 |

+------+------------+--------+--------+

| 1 | 1 | 1 | 1 |

| 1 | 2 | 1 | 1 |

| 2 | 3 | 1 | 1 |

| 3 | 4 | 1 | 2 |

| 3 | 5 | 1 | 2 |

| 3 | 6 | 2 | 3 |

| 4 | 7 | 2 | 3 |

| 4 | 8 | 2 | 4 |

| 5 | 9 | 2 | 4 |

+------+------------+--------+--------+

Beginning with MySQL 8.0.22, the construct **NTILE(NULL)** is no longer permitted.

**[PERCENT\_RANK()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_percent-rank)** ***over\_clause***

Returns the percentage of partition values less than the value in the current row, excluding the highest value. Return values range from 0 to 1 and represent the row relative rank, calculated as the result of this formula, where ***rank*** is the row rank and ***rows*** is the number of partition rows:

(***rank*** - 1) / (***rows*** - 1)

This function should be used with **ORDER BY** to sort partition rows into the desired order. Without **ORDER BY**, all rows are peers.

***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

For an example, see the [**CUME\_DIST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cume-dist) function description.

**[RANK()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_rank)** ***over\_clause***

Returns the rank of the current row within its partition, with gaps. Peers are considered ties and receive the same rank. This function does not assign consecutive ranks to peer groups if groups of size greater than one exist; the result is noncontiguous rank numbers.

This function should be used with **ORDER BY** to sort partition rows into the desired order. Without **ORDER BY**, all rows are peers.

***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

The following query shows the difference between [**RANK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rank), which produces ranks with gaps, and [**DENSE\_RANK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dense-rank), which produces ranks without gaps. The query shows rank values for each member of a set of values in the **val** column, which contains some duplicates. [**RANK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rank) assigns peers (the duplicates) the same rank value, and the next greater value has a rank higher by the number of peers minus one. [**DENSE\_RANK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dense-rank) also assigns peers the same rank value, but the next higher value has a rank one greater. For reference, the query also displays row numbers using [**ROW\_NUMBER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-number):

mysql> **SELECT**

**val,**

**ROW\_NUMBER() OVER w AS 'row\_number',**

**RANK() OVER w AS 'rank',**

**DENSE\_RANK() OVER w AS 'dense\_rank'**

**FROM numbers**

**WINDOW w AS (ORDER BY val);**

+------+------------+------+------------+

| val | row\_number | rank | dense\_rank |

+------+------------+------+------------+

| 1 | 1 | 1 | 1 |

| 1 | 2 | 1 | 1 |

| 2 | 3 | 3 | 2 |

| 3 | 4 | 4 | 3 |

| 3 | 5 | 4 | 3 |

| 3 | 6 | 4 | 3 |

| 4 | 7 | 7 | 4 |

| 4 | 8 | 7 | 4 |

| 5 | 9 | 9 | 5 |

+------+------------+------+------------+

**[ROW\_NUMBER()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_row-number)** ***over\_clause***

Returns the number of the current row within its partition. Rows numbers range from 1 to the number of partition rows.

**ORDER BY** affects the order in which rows are numbered. Without **ORDER BY**, row numbering is nondeterministic.

[**ROW\_NUMBER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-number) assigns peers different row numbers. To assign peers the same value, use [**RANK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rank) or [**DENSE\_RANK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dense-rank). For an example, see the [**RANK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rank) function description.

***over\_clause*** is as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage).

### 12.21.2 Window Function Concepts and Syntax

This section describes how to use window functions. Examples use the same sales information data set as found in the discussion of the [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) function in [Section 12.20.2, “GROUP BY Modifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#group-by-modifiers):

mysql> **SELECT \* FROM sales ORDER BY country, year, product;**

+------+---------+------------+--------+

| year | country | product | profit |

+------+---------+------------+--------+

| 2000 | Finland | Computer | 1500 |

| 2000 | Finland | Phone | 100 |

| 2001 | Finland | Phone | 10 |

| 2000 | India | Calculator | 75 |

| 2000 | India | Calculator | 75 |

| 2000 | India | Computer | 1200 |

| 2000 | USA | Calculator | 75 |

| 2000 | USA | Computer | 1500 |

| 2001 | USA | Calculator | 50 |

| 2001 | USA | Computer | 1500 |

| 2001 | USA | Computer | 1200 |

| 2001 | USA | TV | 150 |

| 2001 | USA | TV | 100 |

+------+---------+------------+--------+

A window function performs an aggregate-like operation on a set of query rows. However, whereas an aggregate operation groups query rows into a single result row, a window function produces a result for each query row:

The row for which function evaluation occurs is called the current row.

The query rows related to the current row over which function evaluation occurs comprise the window for the current row.

For example, using the sales information table, these two queries perform aggregate operations that produce a single global sum for all rows taken as a group, and sums grouped per country:

mysql> **SELECT SUM(profit) AS total\_profit**

**FROM sales;**

+--------------+

| total\_profit |

+--------------+

| 7535 |

+--------------+

mysql> **SELECT country, SUM(profit) AS country\_profit**

**FROM sales**

**GROUP BY country**

**ORDER BY country;**

+---------+----------------+

| country | country\_profit |

+---------+----------------+

| Finland | 1610 |

| India | 1350 |

| USA | 4575 |

+---------+----------------+

By contrast, window operations do not collapse groups of query rows to a single output row. Instead, they produce a result for each row. Like the preceding queries, the following query uses [**SUM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sum), but this time as a window function:

mysql> **SELECT**

**year, country, product, profit,**

**SUM(profit) OVER() AS total\_profit,**

**SUM(profit) OVER(PARTITION BY country) AS country\_profit**

**FROM sales**

**ORDER BY country, year, product, profit;**

+------+---------+------------+--------+--------------+----------------+

| year | country | product | profit | total\_profit | country\_profit |

+------+---------+------------+--------+--------------+----------------+

| 2000 | Finland | Computer | 1500 | 7535 | 1610 |

| 2000 | Finland | Phone | 100 | 7535 | 1610 |

| 2001 | Finland | Phone | 10 | 7535 | 1610 |

| 2000 | India | Calculator | 75 | 7535 | 1350 |

| 2000 | India | Calculator | 75 | 7535 | 1350 |

| 2000 | India | Computer | 1200 | 7535 | 1350 |

| 2000 | USA | Calculator | 75 | 7535 | 4575 |

| 2000 | USA | Computer | 1500 | 7535 | 4575 |

| 2001 | USA | Calculator | 50 | 7535 | 4575 |

| 2001 | USA | Computer | 1200 | 7535 | 4575 |

| 2001 | USA | Computer | 1500 | 7535 | 4575 |

| 2001 | USA | TV | 100 | 7535 | 4575 |

| 2001 | USA | TV | 150 | 7535 | 4575 |

+------+---------+------------+--------+--------------+----------------+

Each window operation in the query is signified by inclusion of an **OVER** clause that specifies how to partition query rows into groups for processing by the window function:

The first **OVER** clause is empty, which treats the entire set of query rows as a single partition. The window function thus produces a global sum, but does so for each row.

The second **OVER** clause partitions rows by country, producing a sum per partition (per country). The function produces this sum for each partition row.

Window functions are permitted only in the select list and **ORDER BY** clause. Query result rows are determined from the **FROM** clause, after **WHERE**, **GROUP BY**, and **HAVING** processing, and windowing execution occurs before **ORDER BY**, **LIMIT**, and **SELECT DISTINCT**.

The **OVER** clause is permitted for many aggregate functions, which therefore can be used as window or nonwindow functions, depending on whether the **OVER** clause is present or absent:

AVG()

BIT\_AND()

BIT\_OR()

BIT\_XOR()

COUNT()

JSON\_ARRAYAGG()

JSON\_OBJECTAGG()

MAX()

MIN()

STDDEV\_POP(), STDDEV(), STD()

STDDEV\_SAMP()

SUM()

VAR\_POP(), VARIANCE()

VAR\_SAMP()

For details about each aggregate function, see [Section 12.20.1, “Aggregate Function Descriptions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#aggregate-functions).

MySQL also supports nonaggregate functions that are used only as window functions. For these, the **OVER** clause is mandatory:

CUME\_DIST()

DENSE\_RANK()

FIRST\_VALUE()

LAG()

LAST\_VALUE()

LEAD()

NTH\_VALUE()

NTILE()

PERCENT\_RANK()

RANK()

ROW\_NUMBER()

For details about each nonaggregate function, see [Section 12.21.1, “Window Function Descriptions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-function-descriptions).

As an example of one of those nonaggregate window functions, this query uses [**ROW\_NUMBER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-number), which produces the row number of each row within its partition. In this case, rows are numbered per country. By default, partition rows are unordered and row numbering is nondeterministic. To sort partition rows, include an **ORDER BY** clause within the window definition. The query uses unordered and ordered partitions (the **row\_num1** and **row\_num2** columns) to illustrate the difference between omitting and including **ORDER BY**:

mysql> **SELECT**

**year, country, product, profit,**

**ROW\_NUMBER() OVER(PARTITION BY country) AS row\_num1,**

**ROW\_NUMBER() OVER(PARTITION BY country ORDER BY year, product) AS row\_num2**

**FROM sales;**

+------+---------+------------+--------+----------+----------+

| year | country | product | profit | row\_num1 | row\_num2 |

+------+---------+------------+--------+----------+----------+

| 2000 | Finland | Computer | 1500 | 2 | 1 |

| 2000 | Finland | Phone | 100 | 1 | 2 |

| 2001 | Finland | Phone | 10 | 3 | 3 |

| 2000 | India | Calculator | 75 | 2 | 1 |

| 2000 | India | Calculator | 75 | 3 | 2 |

| 2000 | India | Computer | 1200 | 1 | 3 |

| 2000 | USA | Calculator | 75 | 5 | 1 |

| 2000 | USA | Computer | 1500 | 4 | 2 |

| 2001 | USA | Calculator | 50 | 2 | 3 |

| 2001 | USA | Computer | 1500 | 3 | 4 |

| 2001 | USA | Computer | 1200 | 7 | 5 |

| 2001 | USA | TV | 150 | 1 | 6 |

| 2001 | USA | TV | 100 | 6 | 7 |

+------+---------+------------+--------+----------+----------+

As mentioned previously, to use a window function (or treat an aggregate function as a window function), include an **OVER** clause following the function call. The **OVER** clause has two forms:

***over\_clause***:

{OVER (***window\_spec***) | OVER ***window\_name***}

Both forms define how the window function should process query rows. They differ in whether the window is defined directly in the **OVER** clause, or supplied by a reference to a named window defined elsewhere in the query:

In the first case, the window specification appears directly in the **OVER** clause, between the parentheses.

In the second case, ***window\_name*** is the name for a window specification defined by a **WINDOW** clause elsewhere in the query. For details, see [Section 12.21.4, “Named Windows”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-named-windows).

For **OVER (*window\_spec*)** syntax, the window specification has several parts, all optional:

***window\_spec***:

[***window\_name***] [***partition\_clause***] [***order\_clause***] [***frame\_clause***]

If **OVER()** is empty, the window consists of all query rows and the window function computes a result using all rows. Otherwise, the clauses present within the parentheses determine which query rows are used to compute the function result and how they are partitioned and ordered:

***window\_name***: The name of a window defined by a **WINDOW** clause elsewhere in the query. If ***window\_name*** appears by itself within the **OVER** clause, it completely defines the window. If partitioning, ordering, or framing clauses are also given, they modify interpretation of the named window. For details, see [Section 12.21.4, “Named Windows”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-named-windows).

***partition\_clause***: A **PARTITION BY** clause indicates how to divide the query rows into groups. The window function result for a given row is based on the rows of the partition that contains the row. If **PARTITION BY** is omitted, there is a single partition consisting of all query rows.

**Note**

Partitioning for window functions differs from table partitioning. For information about table partitioning, see [Chapter 24, *Partitioning*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html).

***partition\_clause*** has this syntax:

***partition\_clause***:

PARTITION BY ***expr*** [, ***expr***] ...

Standard SQL requires **PARTITION BY** to be followed by column names only. A MySQL extension is to permit expressions, not just column names. For example, if a table contains a [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) column named **ts**, standard SQL permits **PARTITION BY ts** but not **PARTITION BY HOUR(ts)**, whereas MySQL permits both.

***order\_clause***: An **ORDER BY** clause indicates how to sort rows in each partition. Partition rows that are equal according to the **ORDER BY** clause are considered peers. If **ORDER BY** is omitted, partition rows are unordered, with no processing order implied, and all partition rows are peers.

***order\_clause*** has this syntax:

***order\_clause***:

ORDER BY ***expr*** [ASC|DESC] [, ***expr*** [ASC|DESC]] ...

Each **ORDER BY** expression optionally can be followed by **ASC** or **DESC** to indicate sort direction. The default is **ASC** if no direction is specified. **NULL** values sort first for ascending sorts, last for descending sorts.

An **ORDER BY** in a window definition applies within individual partitions. To sort the result set as a whole, include an **ORDER BY** at the query top level.

***frame\_clause***: A frame is a subset of the current partition and the frame clause specifies how to define the subset. The frame clause has many subclauses of its own. For details, see [Section 12.21.3, “Window Function Frame Specification”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-frames).

### 12.21.3 Window Function Frame Specification

The definition of a window used with a window function can include a frame clause. A frame is a subset of the current partition and the frame clause specifies how to define the subset.

Frames are determined with respect to the current row, which enables a frame to move within a partition depending on the location of the current row within its partition. Examples:

By defining a frame to be all rows from the partition start to the current row, you can compute running totals for each row.

By defining a frame as extending ***N*** rows on either side of the current row, you can compute rolling averages.

The following query demonstrates the use of moving frames to compute running totals within each group of time-ordered **level** values, as well as rolling averages computed from the current row and the rows that immediately precede and follow it:

mysql> **SELECT**

**time, subject, val,**

**SUM(val) OVER (PARTITION BY subject ORDER BY time**

**ROWS UNBOUNDED PRECEDING)**

**AS running\_total,**

**AVG(val) OVER (PARTITION BY subject ORDER BY time**

**ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING)**

**AS running\_average**

**FROM observations;**

+----------+---------+------+---------------+-----------------+

| time | subject | val | running\_total | running\_average |

+----------+---------+------+---------------+-----------------+

| 07:00:00 | st113 | 10 | 10 | 9.5000 |

| 07:15:00 | st113 | 9 | 19 | 14.6667 |

| 07:30:00 | st113 | 25 | 44 | 18.0000 |

| 07:45:00 | st113 | 20 | 64 | 22.5000 |

| 07:00:00 | xh458 | 0 | 0 | 5.0000 |

| 07:15:00 | xh458 | 10 | 10 | 5.0000 |

| 07:30:00 | xh458 | 5 | 15 | 15.0000 |

| 07:45:00 | xh458 | 30 | 45 | 20.0000 |

| 08:00:00 | xh458 | 25 | 70 | 27.5000 |

+----------+---------+------+---------------+-----------------+

For the **running\_average** column, there is no frame row preceding the first one or following the last. In these cases, [**AVG()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_avg) computes the average of the rows that are available.

Aggregate functions used as window functions operate on rows in the current row frame, as do these nonaggregate window functions:

FIRST\_VALUE()

LAST\_VALUE()

NTH\_VALUE()

Standard SQL specifies that window functions that operate on the entire partition should have no frame clause. MySQL permits a frame clause for such functions but ignores it. These functions use the entire partition even if a frame is specified:

CUME\_DIST()

DENSE\_RANK()

LAG()

LEAD()

NTILE()

PERCENT\_RANK()

RANK()

ROW\_NUMBER()

The frame clause, if given, has this syntax:

***frame\_clause***:

***frame\_units*** ***frame\_extent***

***frame\_units***:

{ROWS | RANGE}

In the absence of a frame clause, the default frame depends on whether an **ORDER BY** clause is present, as described later in this section.

The ***frame\_units*** value indicates the type of relationship between the current row and frame rows:

**ROWS**: The frame is defined by beginning and ending row positions. Offsets are differences in row numbers from the current row number.

**RANGE**: The frame is defined by rows within a value range. Offsets are differences in row values from the current row value.

The ***frame\_extent*** value indicates the start and end points of the frame. You can specify just the start of the frame (in which case the current row is implicitly the end) or use **BETWEEN** to specify both frame endpoints:

***frame\_extent***:

{***frame\_start*** | ***frame\_between***}

***frame\_between***:

BETWEEN ***frame\_start*** AND ***frame\_end***

***frame\_start***, ***frame\_end***: {

CURRENT ROW

| UNBOUNDED PRECEDING

| UNBOUNDED FOLLOWING

| ***expr*** PRECEDING

| ***expr*** FOLLOWING

}

With **BETWEEN** syntax, ***frame\_start*** must not occur later than ***frame\_end***.

The permitted ***frame\_start*** and ***frame\_end*** values have these meanings:

**CURRENT ROW**: For **ROWS**, the bound is the current row. For **RANGE**, the bound is the peers of the current row.

**UNBOUNDED PRECEDING**: The bound is the first partition row.

**UNBOUNDED FOLLOWING**: The bound is the last partition row.

***expr* PRECEDING**: For **ROWS**, the bound is ***expr*** rows before the current row. For **RANGE**, the bound is the rows with values equal to the current row value minus ***expr***; if the current row value is **NULL**, the bound is the peers of the row.

For ***expr* PRECEDING** (and ***expr* FOLLOWING**), ***expr*** can be a **?** parameter marker (for use in a prepared statement), a nonnegative numeric literal, or a temporal interval of the form **INTERVAL *val* *unit***. For **INTERVAL** expressions, ***val*** specifies nonnegative interval value, and ***unit*** is a keyword indicating the units in which the value should be interpreted. (For details about the permitted ***units*** specifiers, see the description of the [**DATE\_ADD()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-add) function in [Section 12.7, “Date and Time Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#date-and-time-functions).)

**RANGE** on a numeric or temporal ***expr*** requires **ORDER BY** on a numeric or temporal expression, respectively.

Examples of valid ***expr* PRECEDING** and ***expr* FOLLOWING** indicators:

10 PRECEDING

INTERVAL 5 DAY PRECEDING

5 FOLLOWING

INTERVAL '2:30' MINUTE\_SECOND FOLLOWING

***expr* FOLLOWING**: For **ROWS**, the bound is ***expr*** rows after the current row. For **RANGE**, the bound is the rows with values equal to the current row value plus ***expr***; if the current row value is **NULL**, the bound is the peers of the row.

For permitted values of ***expr***, see the description of ***expr* PRECEDING**.

The following query demonstrates [**FIRST\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_first-value), [**LAST\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-value), and two instances of [**NTH\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_nth-value):

mysql> **SELECT**

**time, subject, val,**

**FIRST\_VALUE(val) OVER w AS 'first',**

**LAST\_VALUE(val) OVER w AS 'last',**

**NTH\_VALUE(val, 2) OVER w AS 'second',**

**NTH\_VALUE(val, 4) OVER w AS 'fourth'**

**FROM observations**

**WINDOW w AS (PARTITION BY subject ORDER BY time**

**ROWS UNBOUNDED PRECEDING);**

+----------+---------+------+-------+------+--------+--------+

| time | subject | val | first | last | second | fourth |

+----------+---------+------+-------+------+--------+--------+

| 07:00:00 | st113 | 10 | 10 | 10 | NULL | NULL |

| 07:15:00 | st113 | 9 | 10 | 9 | 9 | NULL |

| 07:30:00 | st113 | 25 | 10 | 25 | 9 | NULL |

| 07:45:00 | st113 | 20 | 10 | 20 | 9 | 20 |

| 07:00:00 | xh458 | 0 | 0 | 0 | NULL | NULL |

| 07:15:00 | xh458 | 10 | 0 | 10 | 10 | NULL |

| 07:30:00 | xh458 | 5 | 0 | 5 | 10 | NULL |

| 07:45:00 | xh458 | 30 | 0 | 30 | 10 | 30 |

| 08:00:00 | xh458 | 25 | 0 | 25 | 10 | 30 |

+----------+---------+------+-------+------+--------+--------+

Each function uses the rows in the current frame, which, per the window definition shown, extends from the first partition row to the current row. For the [**NTH\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_nth-value) calls, the current frame does not always include the requested row; in such cases, the return value is **NULL**.

In the absence of a frame clause, the default frame depends on whether an **ORDER BY** clause is present:

With **ORDER BY**: The default frame includes rows from the partition start through the current row, including all peers of the current row (rows equal to the current row according to the **ORDER BY** clause). The default is equivalent to this frame specification:

RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

Without **ORDER BY**: The default frame includes all partition rows (because, without **ORDER BY**, all partition rows are peers). The default is equivalent to this frame specification:

RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

Because the default frame differs depending on presence or absence of **ORDER BY**, adding **ORDER BY** to a query to get deterministic results may change the results. (For example, the values produced by [**SUM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sum) might change.) To obtain the same results but ordered per **ORDER BY**, provide an explicit frame specification to be used regardless of whether **ORDER BY** is present.

The meaning of a frame specification can be nonobvious when the current row value is **NULL**. Assuming that to be the case, these examples illustrate how various frame specifications apply:

**ORDER BY X ASC RANGE BETWEEN 10 FOLLOWING AND 15 FOLLOWING**

The frame starts at **NULL** and stops at **NULL**, thus includes only rows with value **NULL**.

**ORDER BY X ASC RANGE BETWEEN 10 FOLLOWING AND UNBOUNDED FOLLOWING**

The frame starts at **NULL** and stops at the end of the partition. Because an **ASC** sort puts **NULL** values first, the frame is the entire partition.

**ORDER BY X DESC RANGE BETWEEN 10 FOLLOWING AND UNBOUNDED FOLLOWING**

The frame starts at **NULL** and stops at the end of the partition. Because a **DESC** sort puts **NULL** values last, the frame is only the **NULL** values.

**ORDER BY X ASC RANGE BETWEEN 10 PRECEDING AND UNBOUNDED FOLLOWING**

The frame starts at **NULL** and stops at the end of the partition. Because an **ASC** sort puts **NULL** values first, the frame is the entire partition.

**ORDER BY X ASC RANGE BETWEEN 10 PRECEDING AND 10 FOLLOWING**

The frame starts at **NULL** and stops at **NULL**, thus includes only rows with value **NULL**.

**ORDER BY X ASC RANGE BETWEEN 10 PRECEDING AND 1 PRECEDING**

The frame starts at **NULL** and stops at **NULL**, thus includes only rows with value **NULL**.

**ORDER BY X ASC RANGE BETWEEN UNBOUNDED PRECEDING AND 10 FOLLOWING**

The frame starts at the beginning of the partition and stops at rows with value **NULL**. Because an **ASC** sort puts **NULL** values first, the frame is only the **NULL** values.

### 12.21.4 Named Windows

Windows can be defined and given names by which to refer to them in **OVER** clauses. To do this, use a **WINDOW** clause. If present in a query, the **WINDOW** clause falls between the positions of the **HAVING** and **ORDER BY** clauses, and has this syntax:

WINDOW ***window\_name*** AS (***window\_spec***)

[, ***window\_name*** AS (***window\_spec***)] ...

For each window definition, ***window\_name*** is the window name, and ***window\_spec*** is the same type of window specification as given between the parentheses of an **OVER** clause, as described in [Section 12.21.2, “Window Function Concepts and Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-usage):

***window\_spec***:

[***window\_name***] [***partition\_clause***] [***order\_clause***] [***frame\_clause***]

A **WINDOW** clause is useful for queries in which multiple **OVER** clauses would otherwise define the same window. Instead, you can define the window once, give it a name, and refer to the name in the **OVER** clauses. Consider this query, which defines the same window multiple times:

SELECT

val,

ROW\_NUMBER() OVER (ORDER BY val) AS 'row\_number',

RANK() OVER (ORDER BY val) AS 'rank',

DENSE\_RANK() OVER (ORDER BY val) AS 'dense\_rank'

FROM numbers;

The query can be written more simply by using **WINDOW** to define the window once and referring to the window by name in the **OVER** clauses:

SELECT

val,

ROW\_NUMBER() OVER w AS 'row\_number',

RANK() OVER w AS 'rank',

DENSE\_RANK() OVER w AS 'dense\_rank'

FROM numbers

WINDOW w AS (ORDER BY val);

A named window also makes it easier to experiment with the window definition to see the effect on query results. You need only modify the window definition in the **WINDOW** clause, rather than multiple **OVER** clause definitions.

If an **OVER** clause uses **OVER (*window\_name* ...)** rather than **OVER *window\_name***, the named window can be modified by the addition of other clauses. For example, this query defines a window that includes partitioning, and uses **ORDER BY** in the **OVER** clauses to modify the window in different ways:

SELECT

DISTINCT year, country,

FIRST\_VALUE(year) OVER (w ORDER BY year ASC) AS first,

FIRST\_VALUE(year) OVER (w ORDER BY year DESC) AS last

FROM sales

WINDOW w AS (PARTITION BY country);

An **OVER** clause can only add properties to a named window, not modify them. If the named window definition includes a partitioning, ordering, or framing property, the **OVER** clause that refers to the window name cannot also include the same kind of property or an error occurs:

This construct is permitted because the window definition and the referring **OVER** clause do not contain the same kind of properties:

OVER (w ORDER BY country)

... WINDOW w AS (PARTITION BY country)

This construct is not permitted because the **OVER** clause specifies **PARTITION BY** for a named window that already has **PARTITION BY**:

OVER (w PARTITION BY year)

... WINDOW w AS (PARTITION BY country)

The definition of a named window can itself begin with a ***window\_name***. In such cases, forward and backward references are permitted, but not cycles:

This is permitted; it contains forward and backward references but no cycles:

WINDOW w1 AS (w2), w2 AS (), w3 AS (w1)

This is not permitted because it contains a cycle:

WINDOW w1 AS (w2), w2 AS (w3), w3 AS (w1)

### 12.21.5 Window Function Restrictions

The SQL standard imposes a constraint on window functions that they cannot be used in [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements to update rows. Using such functions in a subquery of these statements (to select rows) is permitted.

MySQL does not support these window function features:

**DISTINCT** syntax for aggregate window functions.

Nested window functions.

Dynamic frame endpoints that depend on the value of the current row.

The parser recognizes these window constructs which nevertheless are not supported:

The **GROUPS** frame units specifier is parsed, but produces an error. Only **ROWS** and **RANGE** are supported.

The **EXCLUDE** clause for frame specification is parsed, but produces an error.

**IGNORE NULLS** is parsed, but produces an error. Only **RESPECT NULLS** is supported.

**FROM LAST** is parsed, but produces an error. Only **FROM FIRST** is supported.

## 12.22 Performance Schema Functions

As of MySQL 8.0.16, MySQL includes built-in SQL functions that format or retrieve Performance Schema data, and that may be used as equivalents for the corresponding **sys** schema stored functions. The built-in functions can be invoked in any schema and require no qualifier, unlike the **sys** functions, which require either a **sys.** schema qualifier or that **sys** be the current schema.

**Table 12.27 Performance Schema Functions**

| **Name** | **Description** | **Introduced** |
| --- | --- | --- |
| [**FORMAT\_BYTES()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format-bytes) | Convert byte count to value with units | 8.0.16 |
| [**FORMAT\_PICO\_TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format-pico-time) | Convert time in picoseconds to value with units | 8.0.16 |
| [**PS\_CURRENT\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-current-thread-id) | Performance Schema thread ID for current thread | 8.0.16 |
| [**PS\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-thread-id) | Performance Schema thread ID for given thread | 8.0.16 |

The built-in functions supersede the corresponding **sys** functions, which are deprecated; expect them to be removed in a future version of MySQL. Applications that use the **sys** functions should be adjusted to use the built-in functions instead, keeping in mind some minor differences between the **sys** functions and the built-in functions. For details about these differences, see the function descriptions in this section.

**[FORMAT\_BYTES(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format-bytes)*[count](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format-bytes)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format-bytes)**

Given a numeric byte count, converts it to human-readable format and returns a string consisting of a value and a units indicator. The string contains the number of bytes rounded to 2 decimal places and a minimum of 3 significant digits. Numbers less than 1024 bytes are represented as whole numbers and are not rounded.

The units indicator depends on the size of the byte-count argument as shown in the following table.

| **Argument Value** | **Result Units** | **Result Units Indicator** |
| --- | --- | --- |
| **Up to 1023** | bytes | bytes |
| **Up to 10242 − 1** | kibibytes | KiB |
| **Up to 10243 − 1** | mebibytes | MiB |
| **Up to 10244 − 1** | gibibytes | GiB |
| **Up to 10245 − 1** | tebibytes | TiB |
| **Up to 10246 − 1** | pebibytes | PiB |
| **10246 and up** | exbibytes | EiB |

mysql> **SELECT FORMAT\_BYTES(512), FORMAT\_BYTES(18446644073709551615);**

+-------------------+------------------------------------+

| FORMAT\_BYTES(512) | FORMAT\_BYTES(18446644073709551615) |

+-------------------+------------------------------------+

| 512 bytes | 16.00 EiB |

+-------------------+------------------------------------+

[**FORMAT\_BYTES()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format-bytes) was added in MySQL 8.0.16. It may be used instead of the **sys** schema [**format\_bytes()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sys-schema.html#sys-format-bytes) function, keeping in mind this difference:

[**FORMAT\_BYTES()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format-bytes) uses the **EiB** units indicator. [**sys.format\_bytes()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sys-schema.html#sys-format-bytes) does not.

**[FORMAT\_PICO\_TIME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format-pico-time)*[time\_val](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format-pico-time)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_format-pico-time)**

Given a numeric Performance Schema latency or wait time in picoseconds, converts it to human-readable format and returns a string consisting of a value and a units indicator. The string contains the decimal time rounded to 2 decimal places and a minimum of 3 significant digits. Times under 1 nanosecond are represented as whole numbers and are not rounded.

The units indicator depends on the size of the time-value argument as shown in the following table.

| **Argument Value** | **Result Units** | **Result Units Indicator** |
| --- | --- | --- |
| **Up to 103 − 1** | picoseconds | ps |
| **Up to 106 − 1** | nanoseconds | ns |
| **Up to 109 − 1** | microseconds | us |
| **Up to 1012 − 1** | milliseconds | ms |
| **Up to 60×1012 − 1** | seconds | s |
| **Up to 3.6×1015 − 1** | minutes | min |
| **Up to 8.64×1016 − 1** | hours | h |
| **8.64×1016 and up** | days | d |

mysql> **SELECT FORMAT\_PICO\_TIME(3501), FORMAT\_PICO\_TIME(188732396662000);**

+------------------------+-----------------------------------+

| FORMAT\_PICO\_TIME(3501) | FORMAT\_PICO\_TIME(188732396662000) |

+------------------------+-----------------------------------+

| 3.50 ns | 3.15 min |

+------------------------+-----------------------------------+

[**FORMAT\_PICO\_TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format-pico-time) was added in MySQL 8.0.16. It may be used instead of the **sys** schema [**format\_time()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sys-schema.html#sys-format-time) function, keeping in mind these differences:

To indicate minutes, [**sys.format\_time()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sys-schema.html#sys-format-time) uses the **m** units indicator, whereas [**FORMAT\_PICO\_TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format-pico-time) uses **min**.

[**sys.format\_time()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sys-schema.html#sys-format-time) uses the **w** (weeks) units indicator. [**FORMAT\_PICO\_TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format-pico-time) does not.

**[PS\_CURRENT\_THREAD\_ID()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ps-current-thread-id)**

Returns a **BIGINT UNSIGNED** value representing the Performance Schema thread ID assigned to the current connection.

The thread ID return value is a value of the type given in the **THREAD\_ID** column of Performance Schema tables.

Performance Schema configuration affects [**PS\_CURRENT\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-current-thread-id) the same way as for [**PS\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-thread-id). For details, see the description of that function.

mysql> **SELECT PS\_CURRENT\_THREAD\_ID();**

+------------------------+

| PS\_CURRENT\_THREAD\_ID() |

+------------------------+

| 52 |

+------------------------+

mysql> **SELECT PS\_THREAD\_ID(CONNECTION\_ID());**

+-------------------------------+

| PS\_THREAD\_ID(CONNECTION\_ID()) |

+-------------------------------+

| 52 |

+-------------------------------+

[**PS\_CURRENT\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-current-thread-id) was added in MySQL 8.0.16. It may be used as a shortcut for invoking the **sys** schema [**ps\_thread\_id()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sys-schema.html#sys-ps-thread-id) function with an argument of **NULL** or [**CONNECTION\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_connection-id).

**[PS\_THREAD\_ID(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ps-thread-id)*[connection\_id](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ps-thread-id)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_ps-thread-id)**

Given a connection ID, returns a **BIGINT UNSIGNED** value representing the Performance Schema thread ID assigned to the connection ID, or **NULL** if no thread ID exists for the connection ID. The latter can occur for threads that are not instrumented.

The connection ID argument is a value of the type given in the **PROCESSLIST\_ID** column of the Performance Schema [**threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-threads-table) table or the **Id** column of [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) output.

The thread ID return value is a value of the type given in the **THREAD\_ID** column of Performance Schema tables.

Performance Schema configuration affects [**PS\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-thread-id) operation as follows. (These remarks also apply to [**PS\_CURRENT\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-current-thread-id).)

Disabling the **thread\_instrumentation** consumer disables statistics from being collected and aggregated at the thread level, but has no effect on [**PS\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-thread-id).

If [**performance\_schema\_max\_thread\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#sysvar_performance_schema_max_thread_instances) is not 0, the Performance Schema allocates memory for thread statistics and assigns an internal ID to each thread for which instance memory is available. If there are threads for which instance memory is not available, [**PS\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-thread-id) returns **NULL**; in this case, [**Performance\_schema\_thread\_instances\_lost**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#statvar_Performance_schema_thread_instances_lost) is nonzero.

If [**performance\_schema\_max\_thread\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#sysvar_performance_schema_max_thread_instances) is 0, the Performance Schema allocates no thread memory and [**PS\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-thread-id) returns **NULL**.

If the Performance Schema itself is disabled, [**PS\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-thread-id) produces an error.

mysql> **SELECT PS\_THREAD\_ID(6);**

+-----------------+

| PS\_THREAD\_ID(6) |

+-----------------+

| 45 |

+-----------------+

[**PS\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-thread-id) was added in MySQL 8.0.16. It may be used instead of the **sys** schema [**ps\_thread\_id()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sys-schema.html#sys-ps-thread-id) function, keeping in mind this difference:

With an argument of **NULL**, [**sys.ps\_thread\_id()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sys-schema.html#sys-ps-thread-id) returns the thread ID for the current connection, whereas [**PS\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-thread-id) returns **NULL**. To obtain the current connection thread ID, use [**PS\_CURRENT\_THREAD\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ps-current-thread-id) instead.

## 12.23 Internal Functions

**Table 12.28 Internal Functions**

|  |  |  |
| --- | --- | --- |
| Name | Description | Introduced |
| [CAN\_ACCESS\_COLUMN()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_can-access-column) | Internal use only |  |
| [CAN\_ACCESS\_DATABASE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_can-access-database) | Internal use only |  |
| [CAN\_ACCESS\_TABLE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_can-access-table) | Internal use only |  |
| [CAN\_ACCESS\_USER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_can-access-user) | Internal use only | 8.0.22 |
| [CAN\_ACCESS\_VIEW()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_can-access-view) | Internal use only |  |
| [GET\_DD\_COLUMN\_PRIVILEGES()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-dd-column-privileges) | Internal use only |  |
| [GET\_DD\_CREATE\_OPTIONS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-dd-create-options) | Internal use only |  |
| [GET\_DD\_INDEX\_SUB\_PART\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-dd-index-sub-part-length) | Internal use only |  |
| [INTERNAL\_AUTO\_INCREMENT()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-auto-increment) | Internal use only |  |
| [INTERNAL\_AVG\_ROW\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-avg-row-length) | Internal use only |  |
| [INTERNAL\_CHECK\_TIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-check-time) | Internal use only |  |
| [INTERNAL\_CHECKSUM()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-checksum) | Internal use only |  |
| [INTERNAL\_DATA\_FREE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-data-free) | Internal use only |  |
| [INTERNAL\_DATA\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-data-length) | Internal use only |  |
| [INTERNAL\_DD\_CHAR\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-dd-char-length) | Internal use only |  |
| [INTERNAL\_GET\_COMMENT\_OR\_ERROR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-get-comment-or-error) | Internal use only |  |
| [INTERNAL\_GET\_ENABLED\_ROLE\_JSON()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-get-enabled-role-json) | Internal use only | 8.0.19 |
| [INTERNAL\_GET\_HOSTNAME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-get-hostname) | Internal use only | 8.0.19 |
| [INTERNAL\_GET\_USERNAME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-get-username) | Internal use only | 8.0.19 |
| [INTERNAL\_GET\_VIEW\_WARNING\_OR\_ERROR()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-get-view-warning-or-error) | Internal use only |  |
| [INTERNAL\_INDEX\_COLUMN\_CARDINALITY()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-index-column-cardinality) | Internal use only |  |
| [INTERNAL\_INDEX\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-index-length) | Internal use only |  |
| [INTERNAL\_IS\_ENABLED\_ROLE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-is-enabled-role) | Internal use only | 8.0.19 |
| [INTERNAL\_IS\_MANDATORY\_ROLE()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-is-mandatory-role) | Internal use only | 8.0.19 |
| [INTERNAL\_KEYS\_DISABLED()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-keys-disabled) | Internal use only |  |
| [INTERNAL\_MAX\_DATA\_LENGTH()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-max-data-length) | Internal use only |  |
| [INTERNAL\_TABLE\_ROWS()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-table-rows) | Internal use only |  |
| [INTERNAL\_UPDATE\_TIME()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_internal-update-time) | Internal use only |  |

The functions listed in this section are intended only for internal use by the server. Attempts by users to invoke them result in an error.

**[CAN\_ACCESS\_COLUMN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-column)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-column)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-column)**

**[CAN\_ACCESS\_DATABASE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-database)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-database)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-database)**

**[CAN\_ACCESS\_TABLE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-table)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-table)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-table)**

**[CAN\_ACCESS\_USER(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-user)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-user)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-user)**

**[CAN\_ACCESS\_VIEW(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-view)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-view)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_can-access-view)**

**[GET\_DD\_COLUMN\_PRIVILEGES(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-dd-column-privileges)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-dd-column-privileges)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-dd-column-privileges)**

**[GET\_DD\_CREATE\_OPTIONS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-dd-create-options)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-dd-create-options)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-dd-create-options)**

**[GET\_DD\_INDEX\_SUB\_PART\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-dd-index-sub-part-length)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-dd-index-sub-part-length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_get-dd-index-sub-part-length)**

**[INTERNAL\_AUTO\_INCREMENT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-auto-increment)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-auto-increment)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-auto-increment)**

**[INTERNAL\_AVG\_ROW\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-avg-row-length)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-avg-row-length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-avg-row-length)**

**[INTERNAL\_CHECK\_TIME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-check-time)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-check-time)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-check-time)**

**[INTERNAL\_CHECKSUM(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-checksum)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-checksum)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-checksum)**

**[INTERNAL\_DATA\_FREE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-data-free)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-data-free)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-data-free)**

**[INTERNAL\_DATA\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-data-length)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-data-length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-data-length)**

**[INTERNAL\_DD\_CHAR\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-dd-char-length)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-dd-char-length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-dd-char-length)**

**[INTERNAL\_GET\_COMMENT\_OR\_ERROR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-comment-or-error)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-comment-or-error)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-comment-or-error)**

**[INTERNAL\_GET\_ENABLED\_ROLE\_JSON(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-enabled-role-json)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-enabled-role-json)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-enabled-role-json)**

**[INTERNAL\_GET\_HOSTNAME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-hostname)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-hostname)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-hostname)**

**[INTERNAL\_GET\_USERNAME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-username)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-username)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-username)**

**[INTERNAL\_GET\_VIEW\_WARNING\_OR\_ERROR(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-view-warning-or-error)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-view-warning-or-error)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-get-view-warning-or-error)**

**[INTERNAL\_INDEX\_COLUMN\_CARDINALITY(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-index-column-cardinality)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-index-column-cardinality)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-index-column-cardinality)**

**[INTERNAL\_INDEX\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-index-length)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-index-length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-index-length)**

**[INTERNAL\_IS\_ENABLED\_ROLE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-is-enabled-role)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-is-enabled-role)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-is-enabled-role)**

**[INTERNAL\_IS\_MANDATORY\_ROLE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-is-mandatory-role)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-is-mandatory-role)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-is-mandatory-role)**

**[INTERNAL\_KEYS\_DISABLED(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-keys-disabled)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-keys-disabled)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-keys-disabled)**

**[INTERNAL\_MAX\_DATA\_LENGTH(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-max-data-length)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-max-data-length)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-max-data-length)**

**[INTERNAL\_TABLE\_ROWS(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-table-rows)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-table-rows)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-table-rows)**

**[INTERNAL\_UPDATE\_TIME(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-update-time)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-update-time)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_internal-update-time)**

**[IS\_VISIBLE\_DD\_OBJECT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-visible-dd-object)*[ARGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-visible-dd-object)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-visible-dd-object)**

## 12.24 Miscellaneous Functions

**Table 12.29 Miscellaneous Functions**

| **Name** | **Description** |
| --- | --- |
| [**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) | Suppress ONLY\_FULL\_GROUP\_BY value rejection |
| [**BIN\_TO\_UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bin-to-uuid) | Convert binary UUID to string |
| [**DEFAULT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_default) | Return the default value for a table column |
| [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) | Distinguish super-aggregate ROLLUP rows from regular rows |
| [**INET\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-aton) | Return the numeric value of an IP address |
| [**INET\_NTOA()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-ntoa) | Return the IP address from a numeric value |
| [**INET6\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-aton) | Return the numeric value of an IPv6 address |
| [**INET6\_NTOA()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-ntoa) | Return the IPv6 address from a numeric value |
| [**IS\_IPV4()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv4) | Whether argument is an IPv4 address |
| [**IS\_IPV4\_COMPAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv4-compat) | Whether argument is an IPv4-compatible address |
| [**IS\_IPV4\_MAPPED()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv4-mapped) | Whether argument is an IPv4-mapped address |
| [**IS\_IPV6()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv6) | Whether argument is an IPv6 address |
| [**IS\_UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-uuid) | Whether argument is a valid UUID |
| [**MASTER\_POS\_WAIT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_master-pos-wait) | Block until the replica has read and applied all updates up to the specified position |
| [**NAME\_CONST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_name-const) | Cause the column to have the given name |
| [**SLEEP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sleep) | Sleep for a number of seconds |
| [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) | Return a Universal Unique Identifier (UUID) |
| [**UUID\_SHORT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-short) | Return an integer-valued universal identifier |
| [**UUID\_TO\_BIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin) | Convert string UUID to binary |
| [**VALUES()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_values) | Define the values to be used during an INSERT |

**[ANY\_VALUE(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_any-value)*[arg](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_any-value)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_any-value)**

This function is useful for **GROUP BY** queries when the [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) SQL mode is enabled, for cases when MySQL rejects a query that you know is valid for reasons that MySQL cannot determine. The function return value and type are the same as the return value and type of its argument, but the function result is not checked for the [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) SQL mode.

For example, if **name** is a nonindexed column, the following query fails with [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) enabled:

mysql> **SELECT name, address, MAX(age) FROM t GROUP BY name;**

ERROR 1055 (42000): Expression #2 of SELECT list is not in GROUP

BY clause and contains nonaggregated column 'mydb.t.address' which

is not functionally dependent on columns in GROUP BY clause; this

is incompatible with sql\_mode=only\_full\_group\_by

The failure occurs because **address** is a nonaggregated column that is neither named among **GROUP BY** columns nor functionally dependent on them. As a result, the **address** value for rows within each **name** group is nondeterministic. There are multiple ways to cause MySQL to accept the query:

Alter the table to make **name** a primary key or a unique **NOT NULL** column. This enables MySQL to determine that **address** is functionally dependent on **name**; that is, **address** is uniquely determined by **name**. (This technique is inapplicable if **NULL** must be permitted as a valid **name** value.)

Use [**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) to refer to **address**:

SELECT name, ANY\_VALUE(address), MAX(age) FROM t GROUP BY name;

In this case, MySQL ignores the nondeterminism of **address** values within each **name** group and accepts the query. This may be useful if you simply do not care which value of a nonaggregated column is chosen for each group. [**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) is not an aggregate function, unlike functions such as [**SUM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sum) or [**COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count). It simply acts to suppress the test for nondeterminism.

Disable [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by). This is equivalent to using [**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) with [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) enabled, as described in the previous item.

[**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) is also useful if functional dependence exists between columns but MySQL cannot determine it. The following query is valid because **age** is functionally dependent on the grouping column **age-1**, but MySQL cannot tell that and rejects the query with [**ONLY\_FULL\_GROUP\_BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_only_full_group_by) enabled:

SELECT age FROM t GROUP BY age-1;

To cause MySQL to accept the query, use [**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value):

SELECT ANY\_VALUE(age) FROM t GROUP BY age-1;

[**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) can be used for queries that refer to aggregate functions in the absence of a **GROUP BY** clause:

mysql> **SELECT name, MAX(age) FROM t;**

ERROR 1140 (42000): In aggregated query without GROUP BY, expression

#1 of SELECT list contains nonaggregated column 'mydb.t.name'; this

is incompatible with sql\_mode=only\_full\_group\_by

Without **GROUP BY**, there is a single group and it is nondeterministic which **name** value to choose for the group. [**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) tells MySQL to accept the query:

SELECT ANY\_VALUE(name), MAX(age) FROM t;

It may be that, due to some property of a given data set, you know that a selected nonaggregated column is effectively functionally dependent on a **GROUP BY** column. For example, an application may enforce uniqueness of one column with respect to another. In this case, using [**ANY\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_any-value) for the effectively functionally dependent column may make sense.

For additional discussion, see [Section 12.20.3, “MySQL Handling of GROUP BY”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#group-by-handling).

**[BIN\_TO\_UUID(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bin-to-uuid)*[binary\_uuid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bin-to-uuid)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bin-to-uuid)**, [**BIN\_TO\_UUID(*binary\_uuid*, *swap\_flag*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bin-to-uuid)

**[BIN\_TO\_UUID()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_bin-to-uuid)** is the inverse of [**UUID\_TO\_BIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin). It converts a binary UUID to a string UUID and returns the result. The binary value should be a UUID as a [**VARBINARY(16)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) value. The return value is a **utf8** string of five hexadecimal numbers separated by dashes. (For details about this format, see the [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) function description.) If the UUID argument is **NULL**, the return value is **NULL**. If any argument is invalid, an error occurs.

[**BIN\_TO\_UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bin-to-uuid) takes one or two arguments:

The one-argument form takes a binary UUID value. The UUID value is assumed not to have its time-low and time-high parts swapped. The string result is in the same order as the binary argument.

The two-argument form takes a binary UUID value and a swap-flag value:

If ***swap\_flag*** is 0, the two-argument form is equivalent to the one-argument form. The string result is in the same order as the binary argument.

If ***swap\_flag*** is 1, the UUID value is assumed to have its time-low and time-high parts swapped. These parts are swapped back to their original position in the result value.

For usage examples and information about time-part swapping, see the [**UUID\_TO\_BIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin) function description.

**[DEFAULT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_default)*[col\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_default)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_default)**

Returns the default value for a table column. An error results if the column has no default value.

The use of [**DEFAULT(*col\_name*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_default) to specify the default value for a named column is permitted only for columns that have a literal default value, not for columns that have an expression default value.

mysql> **UPDATE t SET i = DEFAULT(i)+1 WHERE id < 100;**

[**FORMAT(*X*,*D*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format)

Formats the number ***X*** to a format like **'#,###,###.##'**, rounded to ***D*** decimal places, and returns the result as a string. For details, see [Section 12.8, “String Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#string-functions).

**[GROUPING(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_grouping)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_grouping)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_grouping)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_grouping)*[] ...)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_grouping)**

For **GROUP BY** queries that include a **WITH ROLLUP** modifier, the **ROLLUP** operation produces super-aggregate output rows where **NULL** represents the set of all values. The [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) function enables you to distinguish **NULL** values for super-aggregate rows from **NULL** values in regular grouped rows.

[**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) is permitted only in the select list or **HAVING** clause.

Each argument to [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) must be an expression that exactly matches an expression in the **GROUP BY** clause. The expression cannot be a positional specifier. For each expression, [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) produces 1 if the expression value in the current row is a **NULL** representing a super-aggregate value. Otherwise, [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) produces 0, indicating that the expression value is a **NULL** for a regular result row or is not **NULL**.

Suppose that table **t1** contains these rows, where **NULL** indicates something like “other” or “unknown”:

mysql> **SELECT \* FROM t1;**

+------+-------+----------+

| name | size | quantity |

+------+-------+----------+

| ball | small | 10 |

| ball | large | 20 |

| ball | NULL | 5 |

| hoop | small | 15 |

| hoop | large | 5 |

| hoop | NULL | 3 |

+------+-------+----------+

A summary of the table without **WITH ROLLUP** looks like this:

mysql> **SELECT name, size, SUM(quantity) AS quantity**

**FROM t1**

**GROUP BY name, size;**

+------+-------+----------+

| name | size | quantity |

+------+-------+----------+

| ball | small | 10 |

| ball | large | 20 |

| ball | NULL | 5 |

| hoop | small | 15 |

| hoop | large | 5 |

| hoop | NULL | 3 |

+------+-------+----------+

The result contains **NULL** values, but those do not represent super-aggregate rows because the query does not include **WITH ROLLUP**.

Adding **WITH ROLLUP** produces super-aggregate summary rows containing additional **NULL** values. However, without comparing this result to the previous one, it is not easy to see which **NULL** values occur in super-aggregate rows and which occur in regular grouped rows:

mysql> **SELECT name, size, SUM(quantity) AS quantity**

**FROM t1**

**GROUP BY name, size WITH ROLLUP;**

+------+-------+----------+

| name | size | quantity |

+------+-------+----------+

| ball | NULL | 5 |

| ball | large | 20 |

| ball | small | 10 |

| ball | NULL | 35 |

| hoop | NULL | 3 |

| hoop | large | 5 |

| hoop | small | 15 |

| hoop | NULL | 23 |

| NULL | NULL | 58 |

+------+-------+----------+

To distinguish **NULL** values in super-aggregate rows from those in regular grouped rows, use [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping), which returns 1 only for super-aggregate **NULL** values:

mysql> **SELECT**

**name, size, SUM(quantity) AS quantity,**

**GROUPING(name) AS grp\_name,**

**GROUPING(size) AS grp\_size**

**FROM t1**

**GROUP BY name, size WITH ROLLUP;**

+------+-------+----------+----------+----------+

| name | size | quantity | grp\_name | grp\_size |

+------+-------+----------+----------+----------+

| ball | NULL | 5 | 0 | 0 |

| ball | large | 20 | 0 | 0 |

| ball | small | 10 | 0 | 0 |

| ball | NULL | 35 | 0 | 1 |

| hoop | NULL | 3 | 0 | 0 |

| hoop | large | 5 | 0 | 0 |

| hoop | small | 15 | 0 | 0 |

| hoop | NULL | 23 | 0 | 1 |

| NULL | NULL | 58 | 1 | 1 |

+------+-------+----------+----------+----------+

Common uses for [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping):

Substitute a label for super-aggregate **NULL** values:

mysql> **SELECT**

**IF(GROUPING(name) = 1, 'All items', name) AS name,**

**IF(GROUPING(size) = 1, 'All sizes', size) AS size,**

**SUM(quantity) AS quantity**

**FROM t1**

**GROUP BY name, size WITH ROLLUP;**

+-----------+-----------+----------+

| name | size | quantity |

+-----------+-----------+----------+

| ball | NULL | 5 |

| ball | large | 20 |

| ball | small | 10 |

| ball | All sizes | 35 |

| hoop | NULL | 3 |

| hoop | large | 5 |

| hoop | small | 15 |

| hoop | All sizes | 23 |

| All items | All sizes | 58 |

+-----------+-----------+----------+

Return only super-aggregate lines by filtering out the regular grouped lines:

mysql> **SELECT name, size, SUM(quantity) AS quantity**

**FROM t1**

**GROUP BY name, size WITH ROLLUP**

**HAVING GROUPING(name) = 1 OR GROUPING(size) = 1;**

+------+------+----------+

| name | size | quantity |

+------+------+----------+

| ball | NULL | 35 |

| hoop | NULL | 23 |

| NULL | NULL | 58 |

+------+------+----------+

[**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) permits multiple expression arguments. In this case, the [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) return value represents a bitmask combined from the results for each expression, where the lowest-order bit corresponds to the result for the rightmost expression. For example, with three expression arguments, [**GROUPING(*expr1*, *expr2*, *expr3*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) is evaluated like this:

result for GROUPING(***expr3***)

+ result for GROUPING(***expr2***) << 1

+ result for GROUPING(***expr1***) << 2

The following query shows how [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) results for single arguments combine for a multiple-argument call to produce a bitmask value:

mysql> **SELECT**

**name, size, SUM(quantity) AS quantity,**

**GROUPING(name) AS grp\_name,**

**GROUPING(size) AS grp\_size,**

**GROUPING(name, size) AS grp\_all**

**FROM t1**

**GROUP BY name, size WITH ROLLUP;**

+------+-------+----------+----------+----------+---------+

| name | size | quantity | grp\_name | grp\_size | grp\_all |

+------+-------+----------+----------+----------+---------+

| ball | NULL | 5 | 0 | 0 | 0 |

| ball | large | 20 | 0 | 0 | 0 |

| ball | small | 10 | 0 | 0 | 0 |

| ball | NULL | 35 | 0 | 1 | 1 |

| hoop | NULL | 3 | 0 | 0 | 0 |

| hoop | large | 5 | 0 | 0 | 0 |

| hoop | small | 15 | 0 | 0 | 0 |

| hoop | NULL | 23 | 0 | 1 | 1 |

| NULL | NULL | 58 | 1 | 1 | 3 |

+------+-------+----------+----------+----------+---------+

With multiple expression arguments, the [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) return value is nonzero if any expression represents a super-aggregate value. Multiple-argument [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) syntax thus provides a simpler way to write the earlier query that returned only super-aggregate rows, by using a single multiple-argument [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) call rather than multiple single-argument calls:

mysql> **SELECT name, size, SUM(quantity) AS quantity**

**FROM t1**

**GROUP BY name, size WITH ROLLUP**

**HAVING GROUPING(name, size) <> 0;**

+------+------+----------+

| name | size | quantity |

+------+------+----------+

| ball | NULL | 35 |

| hoop | NULL | 23 |

| NULL | NULL | 58 |

+------+------+----------+

Use of [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) is subject to these limitations:

Do not use subquery **GROUP BY** expressions as [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) arguments because matching might fail. For example, matching fails for this query:

mysql> **SELECT GROUPING((SELECT MAX(name) FROM t1))**

**FROM t1**

**GROUP BY (SELECT MAX(name) FROM t1) WITH ROLLUP;**

ERROR 3580 (HY000): Argument #1 of GROUPING function is not in GROUP BY

**GROUP BY** literal expressions should not be used within a **HAVING** clause as [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) arguments. Due to differences between when the optimizer evaluates **GROUP BY** and **HAVING**, matching may succeed but [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) evaluation does not produce the expected result. Consider this query:

SELECT a AS f1, 'w' AS f2

FROM t

GROUP BY f1, f2 WITH ROLLUP

HAVING GROUPING(f2) = 1;

[**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping) is evaluated earlier for the literal constant expression than for the **HAVING** clause as a whole and returns 0. To check whether a query such as this is affected, use [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) and look for **Impossible having** in the **Extra** column.

For more information about **WITH ROLLUP** and [**GROUPING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_grouping), see [Section 12.20.2, “GROUP BY Modifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#group-by-modifiers).

**[INET\_ATON(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet-aton)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet-aton)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet-aton)**

Given the dotted-quad representation of an IPv4 network address as a string, returns an integer that represents the numeric value of the address in network byte order (big endian). [**INET\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-aton) returns **NULL** if it does not understand its argument.

mysql> **SELECT INET\_ATON('10.0.5.9');**

-> 167773449

For this example, the return value is calculated as 10×2563 + 0×2562 + 5×256 + 9.

[**INET\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-aton) may or may not return a non-**NULL** result for short-form IP addresses (such as **'127.1'** as a representation of **'127.0.0.1'**). Because of this, [**INET\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-aton)a should not be used for such addresses.

**Note**

To store values generated by [**INET\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-aton), use an **INT UNSIGNED** column rather than [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), which is signed. If you use a signed column, values corresponding to IP addresses for which the first octet is greater than 127 cannot be stored correctly. See [Section 11.1.7, “Out-of-Range and Overflow Handling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#out-of-range-and-overflow).

**[INET\_NTOA(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet-ntoa)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet-ntoa)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet-ntoa)**

Given a numeric IPv4 network address in network byte order, returns the dotted-quad string representation of the address as a string in the connection character set. [**INET\_NTOA()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-ntoa) returns **NULL** if it does not understand its argument.

mysql> **SELECT INET\_NTOA(167773449);**

-> '10.0.5.9'

**[INET6\_ATON(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet6-aton)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet6-aton)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet6-aton)**

Given an IPv6 or IPv4 network address as a string, returns a binary string that represents the numeric value of the address in network byte order (big endian). Because numeric-format IPv6 addresses require more bytes than the largest integer type, the representation returned by this function has the [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) data type: [**VARBINARY(16)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) for IPv6 addresses and [**VARBINARY(4)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) for IPv4 addresses. If the argument is not a valid address, [**INET6\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-aton) returns **NULL**.

The following examples use [**HEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex) to display the [**INET6\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-aton) result in printable form:

mysql> **SELECT HEX(INET6\_ATON('fdfe::5a55:caff:fefa:9089'));**

-> 'FDFE0000000000005A55CAFFFEFA9089'

mysql> **SELECT HEX(INET6\_ATON('10.0.5.9'));**

-> '0A000509'

**INET6\_ATON()** observes several constraints on valid arguments. These are given in the following list along with examples.

A trailing zone ID is not permitted, as in **fe80::3%1** or **fe80::3%eth0**.

A trailing network mask is not permitted, as in **2001:45f:3:ba::/64** or **198.51.100.0/24**.

For values representing IPv4 addresses, only classless addresses are supported. Classful addresses such as **198.51.1** are rejected. A trailing port number is not permitted, as in **198.51.100.2:8080**. Hexadecimal numbers in address components are not permitted, as in **198.0xa0.1.2**. Octal numbers are not supported: **198.51.010.1** is treated as **198.51.10.1**, not **198.51.8.1**. These IPv4 constraints also apply to IPv6 addresses that have IPv4 address parts, such as IPv4-compatible or IPv4-mapped addresses.

To convert an IPv4 address ***expr*** represented in numeric form as an [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) value to an IPv6 address represented in numeric form as a [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) value, use this expression:

INET6\_ATON(INET\_NTOA(***expr***))

For example:

mysql> **SELECT HEX(INET6\_ATON(INET\_NTOA(167773449)));**

-> '0A000509'

**[INET6\_NTOA(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet6-ntoa)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet6-ntoa)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_inet6-ntoa)**

Given an IPv6 or IPv4 network address represented in numeric form as a binary string, returns the string representation of the address as a string in the connection character set. If the argument is not a valid address, [**INET6\_NTOA()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-ntoa) returns **NULL**.

[**INET6\_NTOA()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-ntoa) has these properties:

It does not use operating system functions to perform conversions, thus the output string is platform independent.

The return string has a maximum length of 39 (4 x 8 + 7). Given this statement:

CREATE TABLE t AS SELECT INET6\_NTOA(***expr***) AS c1;

The resulting table would have this definition:

CREATE TABLE t (c1 VARCHAR(39) CHARACTER SET utf8 DEFAULT NULL);

The return string uses lowercase letters for IPv6 addresses.

mysql> **SELECT INET6\_NTOA(INET6\_ATON('fdfe::5a55:caff:fefa:9089'));**

-> 'fdfe::5a55:caff:fefa:9089'

mysql> **SELECT INET6\_NTOA(INET6\_ATON('10.0.5.9'));**

-> '10.0.5.9'

mysql> **SELECT INET6\_NTOA(UNHEX('FDFE0000000000005A55CAFFFEFA9089'));**

-> 'fdfe::5a55:caff:fefa:9089'

mysql> **SELECT INET6\_NTOA(UNHEX('0A000509'));**

-> '10.0.5.9'

**[IS\_IPV4(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv4)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv4)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv4)**

Returns 1 if the argument is a valid IPv4 address specified as a string, 0 otherwise.

mysql> **SELECT IS\_IPV4('10.0.5.9'), IS\_IPV4('10.0.5.256');**

-> 1, 0

For a given argument, if [**IS\_IPV4()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv4) returns 1, [**INET\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-aton) (and [**INET6\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-aton)) returns non-**NULL**. The converse statement is not true: In some cases, [**INET\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-aton) returns non-**NULL** when [**IS\_IPV4()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv4) returns 0.

As implied by the preceding remarks, [**IS\_IPV4()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv4) is more strict than [**INET\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet-aton) about what constitutes a valid IPv4 address, so it may be useful for applications that need to perform strong checks against invalid values. Alternatively, use [**INET6\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-aton) to convert IPv4 addresses to internal form and check for a **NULL** result (which indicates an invalid address). [**INET6\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-aton) is equally strong as [**IS\_IPV4()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv4) about checking IPv4 addresses.

**[IS\_IPV4\_COMPAT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv4-compat)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv4-compat)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv4-compat)**

This function takes an IPv6 address represented in numeric form as a binary string, as returned by [**INET6\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-aton). It returns 1 if the argument is a valid IPv4-compatible IPv6 address, 0 otherwise. IPv4-compatible addresses have the form **::*ipv4\_address***.

mysql> **SELECT IS\_IPV4\_COMPAT(INET6\_ATON('::10.0.5.9'));**

-> 1

mysql> **SELECT IS\_IPV4\_COMPAT(INET6\_ATON('::ffff:10.0.5.9'));**

-> 0

The IPv4 part of an IPv4-compatible address can also be represented using hexadecimal notation. For example, **198.51.100.1** has this raw hexadecimal value:

mysql> **SELECT HEX(INET6\_ATON('198.51.100.1'));**

-> 'C6336401'

Expressed in IPv4-compatible form, **::198.51.100.1** is equivalent to **::c0a8:0001** or (without leading zeros) **::c0a8:1**

mysql> **SELECT**

->  **IS\_IPV4\_COMPAT(INET6\_ATON('::198.51.100.1')),**

->  **IS\_IPV4\_COMPAT(INET6\_ATON('::c0a8:0001')),**

->  **IS\_IPV4\_COMPAT(INET6\_ATON('::c0a8:1'));**

-> 1, 1, 1

**[IS\_IPV4\_MAPPED(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv4-mapped)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv4-mapped)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv4-mapped)**

This function takes an IPv6 address represented in numeric form as a binary string, as returned by [**INET6\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-aton). It returns 1 if the argument is a valid IPv4-mapped IPv6 address, 0 otherwise. IPv4-mapped addresses have the form **::ffff:*ipv4\_address***.

mysql> **SELECT IS\_IPV4\_MAPPED(INET6\_ATON('::10.0.5.9'));**

-> 0

mysql> **SELECT IS\_IPV4\_MAPPED(INET6\_ATON('::ffff:10.0.5.9'));**

-> 1

As with **IS\_IPV4\_COMPAT()** the IPv4 part of an IPv4-mapped address can also be represented using hexadecimal notation:

mysql> **SELECT**

->  **IS\_IPV4\_MAPPED(INET6\_ATON('::ffff:198.51.100.1')),**

->  **IS\_IPV4\_MAPPED(INET6\_ATON('::ffff:c0a8:0001')),**

->  **IS\_IPV4\_MAPPED(INET6\_ATON('::ffff:c0a8:1'));**

-> 1, 1, 1

**[IS\_IPV6(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv6)*[expr](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv6)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-ipv6)**

Returns 1 if the argument is a valid IPv6 address specified as a string, 0 otherwise. This function does not consider IPv4 addresses to be valid IPv6 addresses.

mysql> **SELECT IS\_IPV6('10.0.5.9'), IS\_IPV6('::1');**

-> 0, 1

For a given argument, if [**IS\_IPV6()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-ipv6) returns 1, [**INET6\_ATON()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_inet6-aton) returns non-**NULL**.

**[IS\_UUID(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-uuid)*[string\_uuid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-uuid)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_is-uuid)**

Returns 1 if the argument is a valid string-format UUID, 0 if the argument is not a valid UUID, and **NULL** if the argument is **NULL**.

“Valid” means that the value is in a format that can be parsed. That is, it has the correct length and contains only the permitted characters (hexadecimal digits in any lettercase and, optionally, dashes and curly braces). This format is most common:

aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee

These other formats are also permitted:

aaaaaaaabbbbccccddddeeeeeeeeeeee

{aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee}

For the meanings of fields within the value, see the [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) function description.

mysql> **SELECT IS\_UUID('6ccd780c-baba-1026-9564-5b8c656024db');**

+-------------------------------------------------+

| IS\_UUID('6ccd780c-baba-1026-9564-5b8c656024db') |

+-------------------------------------------------+

| 1 |

+-------------------------------------------------+

mysql> **SELECT IS\_UUID('6CCD780C-BABA-1026-9564-5B8C656024DB');**

+-------------------------------------------------+

| IS\_UUID('6CCD780C-BABA-1026-9564-5B8C656024DB') |

+-------------------------------------------------+

| 1 |

+-------------------------------------------------+

mysql> **SELECT IS\_UUID('6ccd780cbaba102695645b8c656024db');**

+---------------------------------------------+

| IS\_UUID('6ccd780cbaba102695645b8c656024db') |

+---------------------------------------------+

| 1 |

+---------------------------------------------+

mysql> **SELECT IS\_UUID('{6ccd780c-baba-1026-9564-5b8c656024db}');**

+---------------------------------------------------+

| IS\_UUID('{6ccd780c-baba-1026-9564-5b8c656024db}') |

+---------------------------------------------------+

| 1 |

+---------------------------------------------------+

mysql> **SELECT IS\_UUID('6ccd780c-baba-1026-9564-5b8c6560');**

+---------------------------------------------+

| IS\_UUID('6ccd780c-baba-1026-9564-5b8c6560') |

+---------------------------------------------+

| 0 |

+---------------------------------------------+

mysql> **SELECT IS\_UUID(RAND());**

+-----------------+

| IS\_UUID(RAND()) |

+-----------------+

| 0 |

+-----------------+

**[MASTER\_POS\_WAIT(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_master-pos-wait)*[log\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_master-pos-wait)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_master-pos-wait)*[log\_pos](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_master-pos-wait)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_master-pos-wait)*[timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_master-pos-wait)*[][,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_master-pos-wait)*[channel](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_master-pos-wait)*[])](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_master-pos-wait)**

This function is useful for control of source/replica synchronization. It blocks until the replica has read and applied all updates up to the specified position in the source's binary log. The return value is the number of log events the replica had to wait for to advance to the specified position. The function returns **NULL** if the replication SQL thread is not started, the replica's source information is not initialized, the arguments are incorrect, or an error occurs. It returns **-1** if the timeout has been exceeded. If the replication SQL thread stops while [**MASTER\_POS\_WAIT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_master-pos-wait) is waiting, the function returns **NULL**. If the replica is past the specified position, the function returns immediately.

On a multithreaded replica, the function waits until expiry of the limit set by the [**slave\_checkpoint\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_checkpoint_group) or [**slave\_checkpoint\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_checkpoint_period) system variable, when the checkpoint operation is called to update the status of the replica. Depending on the setting for the system variables, the function might therefore return some time after the specified position was reached.

If binary log transaction compression is in use and the transaction payload at the specified position is compressed (as a **Transaction\_payload\_event**), the function waits until the whole transaction has been read and applied, and the positions have updated.

If a ***timeout*** value is specified, [**MASTER\_POS\_WAIT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_master-pos-wait) stops waiting when ***timeout*** seconds have elapsed. ***timeout*** must be greater than 0; a zero or negative ***timeout*** means no timeout.

The optional ***channel*** value enables you to name which replication channel the function applies to. See [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels) for more information.

This function is unsafe for statement-based replication. A warning is logged if you use this function when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**.

**[NAME\_CONST(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_name-const)*[name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_name-const)*[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_name-const)*[value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_name-const)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_name-const)**

Returns the given value. When used to produce a result set column, [**NAME\_CONST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_name-const) causes the column to have the given name. The arguments should be constants.

mysql> **SELECT NAME\_CONST('myname', 14);**

+--------+

| myname |

+--------+

| 14 |

+--------+

This function is for internal use only. The server uses it when writing statements from stored programs that contain references to local program variables, as described in [Section 25.7, “Stored Program Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-logging). You might see this function in the output from [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog).

For your applications, you can obtain exactly the same result as in the example just shown by using simple aliasing, like this:

mysql> **SELECT 14 AS myname;**

+--------+

| myname |

+--------+

| 14 |

+--------+

1 row in set (0.00 sec)

See [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), for more information about column aliases.

**[SLEEP(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sleep)*[duration](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sleep)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_sleep)**

Sleeps (pauses) for the number of seconds given by the ***duration*** argument, then returns 0. The duration may have a fractional part. If the argument is **NULL** or negative, [**SLEEP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sleep) produces a warning, or an error in strict SQL mode.

When sleep returns normally (without interruption), it returns 0:

mysql> **SELECT SLEEP(1000);**

+-------------+

| SLEEP(1000) |

+-------------+

| 0 |

+-------------+

When [**SLEEP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sleep) is the only thing invoked by a query that is interrupted, it returns 1 and the query itself returns no error. This is true whether the query is killed or times out:

This statement is interrupted using [**KILL QUERY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) from another session:

mysql> **SELECT SLEEP(1000);**

+-------------+

| SLEEP(1000) |

+-------------+

| 1 |

+-------------+

This statement is interrupted by timing out:

mysql> **SELECT /\*+ MAX\_EXECUTION\_TIME(1) \*/ SLEEP(1000);**

+-------------+

| SLEEP(1000) |

+-------------+

| 1 |

+-------------+

When [**SLEEP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sleep) is only part of a query that is interrupted, the query returns an error:

This statement is interrupted using [**KILL QUERY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) from another session:

mysql> **SELECT 1 FROM t1 WHERE SLEEP(1000);**

ERROR 1317 (70100): Query execution was interrupted

This statement is interrupted by timing out:

mysql> **SELECT /\*+ MAX\_EXECUTION\_TIME(1000) \*/ 1 FROM t1 WHERE SLEEP(1000);**

ERROR 3024 (HY000): Query execution was interrupted, maximum statement

execution time exceeded

This function is unsafe for statement-based replication. A warning is logged if you use this function when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**.

**[UUID()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_uuid)**

Returns a Universal Unique Identifier (UUID) generated according to RFC 4122, “A Universally Unique IDentifier (UUID) URN Namespace” (<http://www.ietf.org/rfc/rfc4122.txt>).

A UUID is designed as a number that is globally unique in space and time. Two calls to [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) are expected to generate two different values, even if these calls are performed on two separate devices not connected to each other.

**Warning**

Although [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) values are intended to be unique, they are not necessarily unguessable or unpredictable. If unpredictability is required, UUID values should be generated some other way.

[**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) returns a value that conforms to UUID version 1 as described in RFC 4122. The value is a 128-bit number represented as a **utf8** string of five hexadecimal numbers in **aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee** format:

The first three numbers are generated from the low, middle, and high parts of a timestamp. The high part also includes the UUID version number.

The fourth number preserves temporal uniqueness in case the timestamp value loses monotonicity (for example, due to daylight saving time).

The fifth number is an IEEE 802 node number that provides spatial uniqueness. A random number is substituted if the latter is not available (for example, because the host device has no Ethernet card, or it is unknown how to find the hardware address of an interface on the host operating system). In this case, spatial uniqueness cannot be guaranteed. Nevertheless, a collision should have very low probability.

The MAC address of an interface is taken into account only on FreeBSD, Linux, and Windows. On other operating systems, MySQL uses a randomly generated 48-bit number.

mysql> **SELECT UUID();**

-> '6ccd780c-baba-1026-9564-5b8c656024db'

To convert between string and binary UUID values, use the [**UUID\_TO\_BIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin) and [**BIN\_TO\_UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bin-to-uuid) functions. To check whether a string is a valid UUID value, use the [**IS\_UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-uuid) function.

This function is unsafe for statement-based replication. A warning is logged if you use this function when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**.

**[UUID\_SHORT()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_uuid-short)**

Returns a “short” universal identifier as a 64-bit unsigned integer. Values returned by [**UUID\_SHORT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-short) differ from the string-format 128-bit identifiers returned by the [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) function and have different uniqueness properties. The value of [**UUID\_SHORT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-short) is guaranteed to be unique if the following conditions hold:

The [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) value of the current server is between 0 and 255 and is unique among your set of source and replica servers

You do not set back the system time for your server host between [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) restarts

You invoke [**UUID\_SHORT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-short) on average fewer than 16 million times per second between [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) restarts

The [**UUID\_SHORT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-short) return value is constructed this way:

(server\_id & 255) << 56

+ (server\_startup\_time\_in\_seconds << 24)

+ incremented\_variable++;

mysql> **SELECT UUID\_SHORT();**

-> 92395783831158784

**Note**

[**UUID\_SHORT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-short) does not work with statement-based replication.

**[UUID\_TO\_BIN(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_uuid-to-bin)*[string\_uuid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_uuid-to-bin)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_uuid-to-bin)**, [**UUID\_TO\_BIN(*string\_uuid*, *swap\_flag*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin)

Converts a string UUID to a binary UUID and returns the result. (The [**IS\_UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-uuid) function description lists the permitted string UUID formats.) The return binary UUID is a [**VARBINARY(16)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) value. If the UUID argument is **NULL**, the return value is **NULL**. If any argument is invalid, an error occurs.

[**UUID\_TO\_BIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin) takes one or two arguments:

The one-argument form takes a string UUID value. The binary result is in the same order as the string argument.

The two-argument form takes a string UUID value and a flag value:

If ***swap\_flag*** is 0, the two-argument form is equivalent to the one-argument form. The binary result is in the same order as the string argument.

If ***swap\_flag*** is 1, the format of the return value differs: The time-low and time-high parts (the first and third groups of hexadecimal digits, respectively) are swapped. This moves the more rapidly varying part to the right and can improve indexing efficiency if the result is stored in an indexed column.

Time-part swapping assumes the use of UUID version 1 values, such as are generated by the [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) function. For UUID values produced by other means that do not follow version 1 format, time-part swapping provides no benefit. For details about version 1 format, see the [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) function description.

Suppose that you have the following string UUID value:

mysql> **SET @uuid = '6ccd780c-baba-1026-9564-5b8c656024db';**

To convert the string UUID to binary with or without time-part swapping, use [**UUID\_TO\_BIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin):

mysql> **SELECT HEX(UUID\_TO\_BIN(@uuid));**

+----------------------------------+

| HEX(UUID\_TO\_BIN(@uuid)) |

+----------------------------------+

| 6CCD780CBABA102695645B8C656024DB |

+----------------------------------+

mysql> **SELECT HEX(UUID\_TO\_BIN(@uuid, 0));**

+----------------------------------+

| HEX(UUID\_TO\_BIN(@uuid, 0)) |

+----------------------------------+

| 6CCD780CBABA102695645B8C656024DB |

+----------------------------------+

mysql> **SELECT HEX(UUID\_TO\_BIN(@uuid, 1));**

+----------------------------------+

| HEX(UUID\_TO\_BIN(@uuid, 1)) |

+----------------------------------+

| 1026BABA6CCD780C95645B8C656024DB |

+----------------------------------+

To convert a binary UUID returned by [**UUID\_TO\_BIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin) to a string UUID, use [**BIN\_TO\_UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bin-to-uuid). If you produce a binary UUID by calling [**UUID\_TO\_BIN()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-to-bin) with a second argument of 1 to swap time parts, you should also pass a second argument of 1 to [**BIN\_TO\_UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_bin-to-uuid) to unswap the time parts when converting the binary UUID back to a string UUID:

mysql> **SELECT BIN\_TO\_UUID(UUID\_TO\_BIN(@uuid));**

+--------------------------------------+

| BIN\_TO\_UUID(UUID\_TO\_BIN(@uuid)) |

+--------------------------------------+

| 6ccd780c-baba-1026-9564-5b8c656024db |

+--------------------------------------+

mysql> **SELECT BIN\_TO\_UUID(UUID\_TO\_BIN(@uuid,0),0);**

+--------------------------------------+

| BIN\_TO\_UUID(UUID\_TO\_BIN(@uuid,0),0) |

+--------------------------------------+

| 6ccd780c-baba-1026-9564-5b8c656024db |

+--------------------------------------+

mysql> **SELECT BIN\_TO\_UUID(UUID\_TO\_BIN(@uuid,1),1);**

+--------------------------------------+

| BIN\_TO\_UUID(UUID\_TO\_BIN(@uuid,1),1) |

+--------------------------------------+

| 6ccd780c-baba-1026-9564-5b8c656024db |

+--------------------------------------+

If the use of time-part swapping is not the same for the conversion in both directions, the original UUID is not recovered properly:

mysql> **SELECT BIN\_TO\_UUID(UUID\_TO\_BIN(@uuid,0),1);**

+--------------------------------------+

| BIN\_TO\_UUID(UUID\_TO\_BIN(@uuid,0),1) |

+--------------------------------------+

| baba1026-780c-6ccd-9564-5b8c656024db |

+--------------------------------------+

mysql> **SELECT BIN\_TO\_UUID(UUID\_TO\_BIN(@uuid,1),0);**

+--------------------------------------+

| BIN\_TO\_UUID(UUID\_TO\_BIN(@uuid,1),0) |

+--------------------------------------+

| 1026baba-6ccd-780c-9564-5b8c656024db |

+--------------------------------------+

**[VALUES(](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_values)*[col\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_values)*[)](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\functions.html" \l "function_values)**

In an [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) statement, you can use the **VALUES(*col\_name*)** function in the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) clause to refer to column values from the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) portion of the statement. In other words, **VALUES(*col\_name*)** in the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) clause refers to the value of ***col\_name*** that would be inserted, had no duplicate-key conflict occurred. This function is especially useful in multiple-row inserts. The [**VALUES()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_values) function is meaningful only in the **ON DUPLICATE KEY UPDATE** clause of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements and returns **NULL** otherwise. See [Section 13.2.6.2, “INSERT ... ON DUPLICATE KEY UPDATE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate).

mysql> **INSERT INTO table (a,b,c) VALUES (1,2,3),(4,5,6)**

-> **ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);**

**Important**

This usage is deprecated in MySQL 8.0.20, and is subject to removal in a future release of MySQL. Use a row alias, or row and column aliases, instead. See [Section 13.2.6.2, “INSERT ... ON DUPLICATE KEY UPDATE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate), for more information and examples.

## 12.25 Precision Math

[12.25.1 Types of Numeric Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math-numbers)

[12.25.2 DECIMAL Data Type Characteristics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math-decimal-characteristics)

[12.25.3 Expression Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math-expressions)

[12.25.4 Rounding Behavior](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math-rounding)

[12.25.5 Precision Math Examples](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math-examples)

MySQL provides support for precision math: numeric value handling that results in extremely accurate results and a high degree control over invalid values. Precision math is based on these two features:

SQL modes that control how strict the server is about accepting or rejecting invalid data.

The MySQL library for fixed-point arithmetic.

These features have several implications for numeric operations and provide a high degree of compliance with standard SQL:

***Precise calculations***: For exact-value numbers, calculations do not introduce floating-point errors. Instead, exact precision is used. For example, MySQL treats a number such as **.0001** as an exact value rather than as an approximation, and summing it 10,000 times produces a result of exactly **1**, not a value that is merely “close” to 1.

***Well-defined rounding behavior***: For exact-value numbers, the result of [**ROUND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) depends on its argument, not on environmental factors such as how the underlying C library works.

***Platform independence***: Operations on exact numeric values are the same across different platforms such as Windows and Unix.

***Control over handling of invalid values***: Overflow and division by zero are detectable and can be treated as errors. For example, you can treat a value that is too large for a column as an error rather than having the value truncated to lie within the range of the column's data type. Similarly, you can treat division by zero as an error rather than as an operation that produces a result of **NULL**. The choice of which approach to take is determined by the setting of the server SQL mode.

The following discussion covers several aspects of how precision math works, including possible incompatibilities with older applications. At the end, some examples are given that demonstrate how MySQL handles numeric operations precisely. For information about controlling the SQL mode, see [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode).

### 12.25.1 Types of Numeric Values

The scope of precision math for exact-value operations includes the exact-value data types (integer and [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) types) and exact-value numeric literals. Approximate-value data types and numeric literals are handled as floating-point numbers.

Exact-value numeric literals have an integer part or fractional part, or both. They may be signed. Examples: **1**, **.2**, **3.4**, **-5**, **-6.78**, **+9.10**.

Approximate-value numeric literals are represented in scientific notation with a mantissa and exponent. Either or both parts may be signed. Examples: **1.2E3**, **1.2E-3**, **-1.2E3**, **-1.2E-3**.

Two numbers that look similar may be treated differently. For example, **2.34** is an exact-value (fixed-point) number, whereas **2.34E0** is an approximate-value (floating-point) number.

The [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) data type is a fixed-point type and calculations are exact. In MySQL, the [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) type has several synonyms: [**NUMERIC**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types), [**DEC**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types), [**FIXED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types). The integer types also are exact-value types.

The [**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) and [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) data types are floating-point types and calculations are approximate. In MySQL, types that are synonymous with [**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) or [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) are [**DOUBLE PRECISION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) and [**REAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types).

### 12.25.2 DECIMAL Data Type Characteristics

This section discusses the characteristics of the [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) data type (and its synonyms), with particular regard to the following topics:

Maximum number of digits

Storage format

Storage requirements

The nonstandard MySQL extension to the upper range of [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) columns

The declaration syntax for a [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) column is **DECIMAL(*M*,*D*)**. The ranges of values for the arguments are as follows:

***M*** is the maximum number of digits (the precision). It has a range of 1 to 65.

***D*** is the number of digits to the right of the decimal point (the scale). It has a range of 0 to 30 and must be no larger than ***M***.

If ***D*** is omitted, the default is 0. If ***M*** is omitted, the default is 10.

The maximum value of 65 for ***M*** means that calculations on [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) values are accurate up to 65 digits. This limit of 65 digits of precision also applies to exact-value numeric literals, so the maximum range of such literals differs from before. (There is also a limit on how long the text of [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) literals can be; see [Section 12.25.3, “Expression Handling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math-expressions).)

Values for [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) columns are stored using a binary format that packs nine decimal digits into 4 bytes. The storage requirements for the integer and fractional parts of each value are determined separately. Each multiple of nine digits requires 4 bytes, and any remaining digits left over require some fraction of 4 bytes. The storage required for remaining digits is given by the following table.

| **Leftover Digits** | **Number of Bytes** |
| --- | --- |
| 0 | 0 |
| 1–2 | 1 |
| 3–4 | 2 |
| 5–6 | 3 |
| 7–9 | 4 |

For example, a **DECIMAL(18,9)** column has nine digits on either side of the decimal point, so the integer part and the fractional part each require 4 bytes. A **DECIMAL(20,6)** column has fourteen integer digits and six fractional digits. The integer digits require four bytes for nine of the digits and 3 bytes for the remaining five digits. The six fractional digits require 3 bytes.

[**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) columns do not store a leading **+** character or **-** character or leading **0** digits. If you insert **+0003.1** into a **DECIMAL(5,1)** column, it is stored as **3.1**. For negative numbers, a literal **-** character is not stored.

[**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) columns do not permit values larger than the range implied by the column definition. For example, a **DECIMAL(3,0)** column supports a range of **-999** to **999**. A **DECIMAL(*M*,*D*)** column permits up to ***M*** - ***D*** digits to the left of the decimal point.

The SQL standard requires that the precision of **NUMERIC(*M*,*D*)** be exactly ***M*** digits. For **DECIMAL(*M*,*D*)**, the standard requires a precision of at least ***M*** digits but permits more. In MySQL, **DECIMAL(*M*,*D*)** and **NUMERIC(*M*,*D*)** are the same, and both have a precision of exactly ***M*** digits.

For a full explanation of the internal format of **DECIMAL** values, see the file strings/decimal.c in a MySQL source distribution. The format is explained (with an example) in the **decimal2bin()** function.

### 12.25.3 Expression Handling

With precision math, exact-value numbers are used as given whenever possible. For example, numbers in comparisons are used exactly as given without a change in value. In strict SQL mode, for [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) into a column with an exact data type ([**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) or integer), a number is inserted with its exact value if it is within the column range. When retrieved, the value should be the same as what was inserted. (If strict SQL mode is not enabled, truncation for [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) is permissible.)

Handling of a numeric expression depends on what kind of values the expression contains:

If any approximate values are present, the expression is approximate and is evaluated using floating-point arithmetic.

If no approximate values are present, the expression contains only exact values. If any exact value contains a fractional part (a value following the decimal point), the expression is evaluated using [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) exact arithmetic and has a precision of 65 digits. The term “exact” is subject to the limits of what can be represented in binary. For example, **1.0/3.0** can be approximated in decimal notation as **.333...**, but not written as an exact number, so **(1.0/3.0)\*3.0** does not evaluate to exactly **1.0**.

Otherwise, the expression contains only integer values. The expression is exact and is evaluated using integer arithmetic and has a precision the same as [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) (64 bits).

If a numeric expression contains any strings, they are converted to double-precision floating-point values and the expression is approximate.

Inserts into numeric columns are affected by the SQL mode, which is controlled by the [**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) system variable. (See [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode).) The following discussion mentions strict mode (selected by the [**STRICT\_ALL\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_strict_all_tables) or [**STRICT\_TRANS\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_strict_trans_tables) mode values) and [**ERROR\_FOR\_DIVISION\_BY\_ZERO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_error_for_division_by_zero). To turn on all restrictions, you can simply use [**TRADITIONAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_traditional) mode, which includes both strict mode values and [**ERROR\_FOR\_DIVISION\_BY\_ZERO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_error_for_division_by_zero):

SET sql\_mode='TRADITIONAL';

If a number is inserted into an exact type column ([**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) or integer), it is inserted with its exact value if it is within the column range and precision.

If the value has too many digits in the fractional part, rounding occurs and a note is generated. Rounding is done as described in [Section 12.25.4, “Rounding Behavior”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math-rounding). Truncation due to rounding of the fractional part is not an error, even in strict mode.

If the value has too many digits in the integer part, it is too large (out of range) and is handled as follows:

If strict mode is not enabled, the value is truncated to the nearest legal value and a warning is generated.

If strict mode is enabled, an overflow error occurs.

For [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) literals, in addition to the precision limit of 65 digits, there is a limit on how long the text of the literal can be. If the value exceeds approximately 80 characters, unexpected results can occur. For example:

mysql> **SELECT**

**CAST(0000000000000000000000000000000000000000000000000000000000000000000000000000000020.01 AS DECIMAL(15,2)) as val;**

+------------------+

| val |

+------------------+

| 9999999999999.99 |

+------------------+

1 row in set, 2 warnings (0.00 sec)

mysql> **SHOW WARNINGS;**

+---------+------+----------------------------------------------+

| Level | Code | Message |

+---------+------+----------------------------------------------+

| Warning | 1292 | Truncated incorrect DECIMAL value: '20' |

| Warning | 1264 | Out of range value for column 'val' at row 1 |

+---------+------+----------------------------------------------+

2 rows in set (0.00 sec)

Underflow is not detected, so underflow handling is undefined.

For inserts of strings into numeric columns, conversion from string to number is handled as follows if the string has nonnumeric contents:

A string that does not begin with a number cannot be used as a number and produces an error in strict mode, or a warning otherwise. This includes the empty string.

A string that begins with a number can be converted, but the trailing nonnumeric portion is truncated. If the truncated portion contains anything other than spaces, this produces an error in strict mode, or a warning otherwise.

By default, division by zero produces a result of **NULL** and no warning. By setting the SQL mode appropriately, division by zero can be restricted.

With the [**ERROR\_FOR\_DIVISION\_BY\_ZERO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_error_for_division_by_zero) SQL mode enabled, MySQL handles division by zero differently:

If strict mode is not enabled, a warning occurs.

If strict mode is enabled, inserts and updates involving division by zero are prohibited, and an error occurs.

In other words, inserts and updates involving expressions that perform division by zero can be treated as errors, but this requires [**ERROR\_FOR\_DIVISION\_BY\_ZERO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_error_for_division_by_zero) in addition to strict mode.

Suppose that we have this statement:

INSERT INTO t SET i = 1/0;

This is what happens for combinations of strict and [**ERROR\_FOR\_DIVISION\_BY\_ZERO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_error_for_division_by_zero) modes.

| [**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode)**Value** | **Result** |
| --- | --- |
| **''** (Default) | No warning, no error; **i** is set to **NULL**. |
| strict | No warning, no error; **i** is set to **NULL**. |
| [**ERROR\_FOR\_DIVISION\_BY\_ZERO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_error_for_division_by_zero) | Warning, no error; **i** is set to **NULL**. |
| strict,[**ERROR\_FOR\_DIVISION\_BY\_ZERO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_error_for_division_by_zero) | Error condition; no row is inserted. |

### 12.25.4 Rounding Behavior

This section discusses precision math rounding for the [**ROUND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) function and for inserts into columns with exact-value types ([**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) and integer).

The [**ROUND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) function rounds differently depending on whether its argument is exact or approximate:

For exact-value numbers, [**ROUND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) uses the “round half up” rule: A value with a fractional part of .5 or greater is rounded up to the next integer if positive or down to the next integer if negative. (In other words, it is rounded away from zero.) A value with a fractional part less than .5 is rounded down to the next integer if positive or up to the next integer if negative. (In other words, it is rounded toward zero.)

For approximate-value numbers, the result depends on the C library. On many systems, this means that [**ROUND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) uses the “round to nearest even” rule: A value with a fractional part exactly half way between two integers is rounded to the nearest even integer.

The following example shows how rounding differs for exact and approximate values:

mysql> **SELECT ROUND(2.5), ROUND(25E-1);**

+------------+--------------+

| ROUND(2.5) | ROUND(25E-1) |

+------------+--------------+

| 3 | 2 |

+------------+--------------+

For inserts into a [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) or integer column, the target is an exact data type, so rounding uses “round half away from zero,” regardless of whether the value to be inserted is exact or approximate:

mysql> **CREATE TABLE t (d DECIMAL(10,0));**

Query OK, 0 rows affected (0.00 sec)

mysql> **INSERT INTO t VALUES(2.5),(2.5E0);**

Query OK, 2 rows affected, 2 warnings (0.00 sec)

Records: 2 Duplicates: 0 Warnings: 2

mysql> **SHOW WARNINGS;**

+-------+------+----------------------------------------+

| Level | Code | Message |

+-------+------+----------------------------------------+

| Note | 1265 | Data truncated for column 'd' at row 1 |

| Note | 1265 | Data truncated for column 'd' at row 2 |

+-------+------+----------------------------------------+

2 rows in set (0.00 sec)

mysql> **SELECT d FROM t;**

+------+

| d |

+------+

| 3 |

| 3 |

+------+

2 rows in set (0.00 sec)

The [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) statement displays the notes that are generated by truncation due to rounding of the fractional part. Such truncation is not an error, even in strict SQL mode (see [Section 12.25.3, “Expression Handling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math-expressions)).

### 12.25.5 Precision Math Examples

This section provides some examples that show precision math query results in MySQL. These examples demonstrate the principles described in [Section 12.25.3, “Expression Handling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math-expressions), and [Section 12.25.4, “Rounding Behavior”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math-rounding).

***Example 1***. Numbers are used with their exact value as given when possible:

mysql> **SELECT (.1 + .2) = .3;**

+----------------+

| (.1 + .2) = .3 |

+----------------+

| 1 |

+----------------+

For floating-point values, results are inexact:

mysql> **SELECT (.1E0 + .2E0) = .3E0;**

+----------------------+

| (.1E0 + .2E0) = .3E0 |

+----------------------+

| 0 |

+----------------------+

Another way to see the difference in exact and approximate value handling is to add a small number to a sum many times. Consider the following stored procedure, which adds **.0001** to a variable 1,000 times.

CREATE PROCEDURE p ()

BEGIN

DECLARE i INT DEFAULT 0;

DECLARE d DECIMAL(10,4) DEFAULT 0;

DECLARE f FLOAT DEFAULT 0;

WHILE i < 10000 DO

SET d = d + .0001;

SET f = f + .0001E0;

SET i = i + 1;

END WHILE;

SELECT d, f;

END;

The sum for both **d** and **f** logically should be 1, but that is true only for the decimal calculation. The floating-point calculation introduces small errors:

+--------+------------------+

| d | f |

+--------+------------------+

| 1.0000 | 0.99999999999991 |

+--------+------------------+

***Example 2***. Multiplication is performed with the scale required by standard SQL. That is, for two numbers ***X1*** and ***X2*** that have scale ***S1*** and ***S2***, the scale of the result is ***S1* + *S2***:

mysql> **SELECT .01 \* .01;**

+-----------+

| .01 \* .01 |

+-----------+

| 0.0001 |

+-----------+

***Example 3***. Rounding behavior for exact-value numbers is well-defined:

Rounding behavior (for example, with the [**ROUND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) function) is independent of the implementation of the underlying C library, which means that results are consistent from platform to platform.

Rounding for exact-value columns ([**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) and integer) and exact-valued numbers uses the “round half away from zero” rule. A value with a fractional part of .5 or greater is rounded away from zero to the nearest integer, as shown here:

mysql> **SELECT ROUND(2.5), ROUND(-2.5);**

+------------+-------------+

| ROUND(2.5) | ROUND(-2.5) |

+------------+-------------+

| 3 | -3 |

+------------+-------------+

Rounding for floating-point values uses the C library, which on many systems uses the “round to nearest even” rule. A value with a fractional part exactly half way between two integers is rounded to the nearest even integer:

mysql> **SELECT ROUND(2.5E0), ROUND(-2.5E0);**

+--------------+---------------+

| ROUND(2.5E0) | ROUND(-2.5E0) |

+--------------+---------------+

| 2 | -2 |

+--------------+---------------+

***Example 4***. In strict mode, inserting a value that is out of range for a column causes an error, rather than truncation to a legal value.

When MySQL is not running in strict mode, truncation to a legal value occurs:

mysql> **SET sql\_mode='';**

Query OK, 0 rows affected (0.00 sec)

mysql> **CREATE TABLE t (i TINYINT);**

Query OK, 0 rows affected (0.01 sec)

mysql> **INSERT INTO t SET i = 128;**

Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> **SELECT i FROM t;**

+------+

| i |

+------+

| 127 |

+------+

1 row in set (0.00 sec)

However, an error occurs if strict mode is in effect:

mysql> **SET sql\_mode='STRICT\_ALL\_TABLES';**

Query OK, 0 rows affected (0.00 sec)

mysql> **CREATE TABLE t (i TINYINT);**

Query OK, 0 rows affected (0.00 sec)

mysql> **INSERT INTO t SET i = 128;**

ERROR 1264 (22003): Out of range value adjusted for column 'i' at row 1

mysql> **SELECT i FROM t;**

Empty set (0.00 sec)

***Example 5***: In strict mode and with [**ERROR\_FOR\_DIVISION\_BY\_ZERO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_error_for_division_by_zero) set, division by zero causes an error, not a result of **NULL**.

In nonstrict mode, division by zero has a result of **NULL**:

mysql> **SET sql\_mode='';**

Query OK, 0 rows affected (0.01 sec)

mysql> **CREATE TABLE t (i TINYINT);**

Query OK, 0 rows affected (0.00 sec)

mysql> **INSERT INTO t SET i = 1 / 0;**

Query OK, 1 row affected (0.00 sec)

mysql> **SELECT i FROM t;**

+------+

| i |

+------+

| NULL |

+------+

1 row in set (0.03 sec)

However, division by zero is an error if the proper SQL modes are in effect:

mysql> **SET sql\_mode='STRICT\_ALL\_TABLES,ERROR\_FOR\_DIVISION\_BY\_ZERO';**

Query OK, 0 rows affected (0.00 sec)

mysql> **CREATE TABLE t (i TINYINT);**

Query OK, 0 rows affected (0.00 sec)

mysql> **INSERT INTO t SET i = 1 / 0;**

ERROR 1365 (22012): Division by 0

mysql> **SELECT i FROM t;**

Empty set (0.01 sec)

***Example 6***. Exact-value literals are evaluated as exact values.

Approximate-value literals are evaluated using floating point, but exact-value literals are handled as [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types):

mysql> **CREATE TABLE t SELECT 2.5 AS a, 25E-1 AS b;**

Query OK, 1 row affected (0.01 sec)

Records: 1 Duplicates: 0 Warnings: 0

mysql> **DESCRIBE t;**

+-------+-----------------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-----------------------+------+-----+---------+-------+

| a | decimal(2,1) unsigned | NO | | 0.0 | |

| b | double | NO | | 0 | |

+-------+-----------------------+------+-----+---------+-------+

2 rows in set (0.01 sec)

***Example 7***. If the argument to an aggregate function is an exact numeric type, the result is also an exact numeric type, with a scale at least that of the argument.

Consider these statements:

mysql> **CREATE TABLE t (i INT, d DECIMAL, f FLOAT);**

mysql> **INSERT INTO t VALUES(1,1,1);**

mysql> **CREATE TABLE y SELECT AVG(i), AVG(d), AVG(f) FROM t;**

The result is a double only for the floating-point argument. For exact type arguments, the result is also an exact type:

mysql> **DESCRIBE y;**

+--------+---------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------+---------------+------+-----+---------+-------+

| AVG(i) | decimal(14,4) | YES | | NULL | |

| AVG(d) | decimal(14,4) | YES | | NULL | |

| AVG(f) | double | YES | | NULL | |

+--------+---------------+------+-----+---------+-------+

The result is a double only for the floating-point argument. For exact type arguments, the result is also an exact type.