MATH FUNCTIONS

Round x to y decimal places	`round(x, [y])`
Return modulo of x % y	`mod(x, y)`
Computes value of x^y	`pow(x, y)` `power(X, y)`
Returns square root of x	`sqrt(x)`
Truncates x to integer value	`trunc(x)`
Returns first int >= x	`floor(x)`
Returns first int <= x	`ceil(x)` `ceiling(x)`
Returns absolute value of x	`abs(x)`

DATE FUNCTIONS

Returns the date as text in the format 'YYYY-MM-DD'	`date(time-value, modifier, ...)`
Returns time as text in the format 'HH:MM:SS'	`time(time-value, modifier, ...)`
Returns the date and time as text in the format: 'YYYY-MM-DD HH:MM:SS	`datetime(time-value, modifier, ...)`
Returns the Julian day	`julianday(time-value, modifier, ...)`
Returns a unix timestamp	`unixepoch(time-value, modifier, ...)`
Returns the date formatted according to the format string specified as the first argument.	`strftime(format, time-value, modifier, ...)`
Returns a string that describes the amount of time that must be added to time-value2 in order to reach time-value1.	`timediff(time-value1, time-value2)`

FORMATTING STRINGS	
Day of month: 00	`%d`
Fractional seconds: SS.SSS	`$f`
Hour: 00-24	`%H`
Day of year: 001-366	`%j`
Julian day number	`%J`
Month: 01-12	`%m`
Minute: 00-59	`%M`
Unix time	`%s`
Seconds: 00-59	`%S`
Day of week: 0-6 (Sunday=0)	`%w`
Week of year: 00-53	`%W`
Year: 0000-9999	`%Y`
TIME VALUES	
T is a character separating the time and date (ex: 2013-10-07**T**08:23)	`YYYY-MM-DD [HH:MM][:SS][.SSS]` `YYYY-MM-DDTHH:MM[:SS][.SSS]`
The date of 2000-01-01 is used	`HH:MM[:SS][.SSS]`
Current UTC date	`now`
Julian day number (int or float)	`DDDDDDDDDD`
TIME VALUES	
Add or subtract x corresponding time from the time value	`x days / hours / minutes / seconds / months / years`
Add or subtract HH:MM from time value	`HH:MM[:SS]`
Add or subtract from time value	`± YYYY-MM-DD [HH][:MM][:SS]`
Shift time value back to start of date value	`start of month / year / day`
Advances time value to next weekday x	`weekday x`

OTHER FUNCTIONS

Returns number of rows changed by last DML statement	`changes()`
Returns total number of row changes caused by DML since connection opened	`total_changes()`
Returns copy of first non-NULL argument or NULL if both are NULL	`ifnull(x, y)`
Returns number of characters in x	`length(x)`
Returns a pseudo-random integer	`random()`
Returns datatype of expression x	`typeof(x)`

CASE/IIF

CASE equality syntax	`CASE expression`
	`  WHEN value_1 THEN result_1`
	`  WHEN value_2 THEN result_2 ...`
	`  [ ELSE result_else ]`
	`END`

Boolean comparison CASE syntax	`CASE`
	`  WHEN bool_expression_1 THEN result_1`
	`  WHEN bool_expression_2 THEN result_2 ...`
	`  [ ELSE result_else ]`
	`END`

Multiple conditions	`CASE`
	`  WHEN expr_1 AND expr_2 THEN ...`
	`  WHEN expr_3 OR expr_4 THEN ...`

If check is true, returns x, else returns y	`iif(check, x, y)`
Example:	`IIF(milliseconds <= 300000, 'Short', 'Long')`

Nested iif statements	`IFF(condition_1, evaluates_to_true,`
	`  IIF(condition_2, evaluates_to_true,`
	`  IIF(condition_3, evaluates_to_true, all_false)`
	`  )`
	`)`