SQL in SeaTable¶

You can use SQL to query data in SeaTable. If some tables in a base are archived, archived rows are also queried, as well as rows that are not archived yet.

Supported SQL Syntax¶

Currently only `select`, `insert`, `update`, and `delete` statements are supported. (`insert`, `update`, and `delete` statements require version 2.7 or later)

The synatx of `select` statement is:

```
SELECT [DISTINCT] fields FROM table_name [WhereClause] [GroupByClause] [HavingClause] [OrderByClause] [Limit Option]
```

Since version 4.3, basic JOIN query is supported, for example:

```
SELECT ... FROM table1, table2 WHERE table1.column1 = table2.column2 AND ...
```

The JOIN query has the following restrictions:

- Only `INNER JOIN` is supported, `LEFT JOIN`, `RIGHT JOIN`, and `FULL JOIN` are not supported.
- Tables in the `FROM` clause should be unique (no duplicate tables).
- Each table in the `FROM` clause should be associated with at least one join condition.
- Join conditions should be placed in the `WHERE` clause, and connected with `AND`s.
- Join conditions can only use equality operator on columns, e.g. `table1.column1 = table2.column2`.
- Columns in join conditions must be indexed, unless the table is not archived.

Notes:

- Most SQL syntax can be used in where clause, including arithmetic expressions, comparison operators, `[NOT] LIKE`, `IN`, `BETWEEN ... AND ...`, `AND`, `OR`, `NOT`, `IS [NOT] TRUE`, `IS [NOT] NULL`.
- Arithmetic expressions only support numbers.
- `LIKE` only supports strings. The key word `ILIKE` can be used instead of `LIKE` to make the match case-insensitive.
- `BETWEEN ... AND ...` only supports numbers and time.
- Time constants should be strings in ISO format (e.g. "2020-09-08 00:11:23"). Since 2.8 version, strings in RFC 3339 format are supported (such as "2020-12-31T23:59:60Z").

- GROUP BY uses strict syntax. The selected fields must appear in group by list, except for aggregation functions (COUNT, SUM, MAX, MIN, AVG) and formulas (see extended syntax section below).
- HAVING filters rows resulting from the group by clause. Only fields referred in the "GROUP BY" clause or aggregation functions (such as "SUM") can be used in having clause. Other syntax is the same as specified for the where clause.
- Fields in "order by" list must be a column or an expression in the selected fields. For example, select a from table order by b is invalid; while select a from table order by b and select abs(a), b from table order by abs(a) are valid.
- Limit options are in MySQL format. The general syntax is LIMIT ... OFFSET .... You may obmit LIMIT or OFFSET.
- Field alias with AS syntax is supported. For example, select table.a as a from tablereturns rows whose first column is keyed by "a". There are two notes:
- Field alias can be referred in group by, having and order by clauses. E.g., select t.registration as r, count(*) as c from t group by r having c > 100 is valid.
- Field alias cannot be referred in where clause. E.g., select t.registration as r, count(*) from t group by r where r > "2020-01-01" will report syntax error.

Each returned row is a JSON map. The keys of the maps are the column keys, NOT column names. To use column names as keys, the convert_keys parameter (available since version 2.4) in query request should be TRUE. For JOIN query, the keys of row maps match the "id" fields (not the "key" or the "name"). Those column fields (e.g. id, key, name) are returned under metadata array in query response.

The synatx of insert, update, and delete statements are:

```
INSERT INTO table_name [column_list] VALUES value_list [, ...]

UPDATE table_name SET column_name = value [, ...] [WhereClause]

DELETE FROM table_name [WhereClause]
```

- column_list is a list of column names surrounded by parentheses. If omitted, it defaults to all updatable columns.
- value_list is a list of values surrounded by parentheses. Values must be in the same order as the column list, for example: (1, "2", 3.0).
- Multivalued columns, such as multiple-select column type, requires values to be surrounded by parentheses, for example: (1, "2", 3.0, ("foo", "bar")).
- Values of single-select and multiple-select column types must be option names, not option keys.
- WhereClause is optional. If omitted, all rows in the table are included.
- INSERT statement only supports bases that have been archived. The rows will be inserted into big-data storage. It'll return error if the base is not archived yet. If you want to insert rows into such bases, please use API for adding rows (e.g. Python API).

- UPDATE and DELETE statements allows updating/deleting rows in both normal and big-data storage.

Note: these column types are *not* allowed to insert or update:

- built-in columns, such as _id, _ctime.
- image, file, formula, link, link-formula, geolocation, auto-number, button

Data Types¶

Below is mapping from SeaTable column types to SQL column types.

SeaTable Column Type	SQL Column Type	Query result format	Use in WH clause
text	String		Supported
long-text	String	Raw text in Markdown formst	Supported
number	Float		Supported
single-select	String	Returned rows contain the option key by default. To return the option name, the convert_keysparameter (available since version 2.4) in query request should be TRUE.	Refer an o by its nam E.g. where single_selec "New York"
multiple-select	List of strings	Returned rows contain the option key by default. To return the option name, the convert_keysparameter (available since version 2.4) in query request should be TRUE.	Refer an o by its nam E.g. where multi_select "New York" More detai "List Type section bel
checkbox	Bool		Supported

SeaTable Column Type	SQL Column Type	Query result format	Use in WH clause
date	Datetime	Time strings in RFC 3339 format	Constants expressed strings in I format. e.g "2006-1-2' "2006-1-2 15:04:05". 2.8 version strings in R 3339 forma supported as "2020-1 31T23:59:
image	List of URL for images	A JSON array with image URLs as elements	Supported. More detai "List Type section bel
file	Will be returned as JSON format string when queried.	Not supported	Not Suppo
collaborator	List of user IDs	Format is like 5758ecdce3e741ad81293a304b6d3388@auth.local. If you need user names, you have to convert with seatable APIs.	Supported. More detai "List Type section bel

SeaTable Column Type	SQL Column Type	Query result format	Use in WH clause
link to other records	List of linked rows	Supported. More details in "List Types" section below.	Supported. More detai "List Type section bel
formula	The type depends on the return value of the formula.	Depends on the type of the return value	Depends o type of the return valu
_creator	User ID as stirng	Format is like 5758ecdce3e741ad81293a304b6d3388@auth.local. If you need user names, you have to convert with seatable APIs.	Supported
_ctime	Datetime	Time strings in RFC 3339 format	Constants expressed strings in I format. e.g "2006-1-2" "2006-1-2 15:04:05". 2.8 version strings in R 3339 forma supported as "2020-1 31T23:59:
_last_modifier	User ID as string	Format is like 5758ecdce3e741ad81293a304b6d3388@auth.local. If you need user names, you have to convert with seatable APIs.	Supported

SeaTable Column Type	SQL Column Type	Query result format	Use in WH clause
_mtime	Datetime	Time strings in RFC 3339 format	Constants expressed strings in I format. e.g "2006-1-2" "2006-1-2 15:04:05". 2.8 version strings in R 3339 forma supported as "2020-1 31T23:59:
auto number	String		Supported
url	String		Supported
email	String		Supported
duration	Float	Returned in the unit of seconds	Supported

List Types¶

In SeaTable, two categories of column types are list types:

- Built-in list types: including multiple selection, image, collaborator, and link to other records.
- Return values for the following link formulas: formula columns whose formula is {link.column} or lookup; link formula columns whose formula is lookup, findmin or findmax.

When referring a column with list type in where conditions, the following rules apply, depending on the type for the list elements. (If an operator is not listed below, it's unsupported.)

Element Type	Operator	Rule
string	IN, extended list operators (e.g. has any of)	Follow the rules of the operator.
string	LIKE, ILIKE	Always take the first element for comparison; if there is no element, use "".
string	IS NULL	Return true when the list is empty or no data in the cell.
string	=, !=	Always take the first element for comparison; if there is no element, use "".
float	IN, extended list operators (e.g. has any of)	Follow the rules of the operator.
float	=, !=, \<, \<=, >, >=, between	If there is only 1 element, use that element; otherwise only return true for != operator.
float	IS NULL	Return true when the list is empty or no data in the cell.
float	Arithmetics such as +/-/*//	Use the first element for calculation.
Datetime	IN, extended list operators (e.g. has any of)	Follow the rules of the operator.
Datetime	=, !=, \<, \<=, >, >=, between	If there is only 1 element, use that element; otherwise only return true for != operator.
Datetime	IS NULL	Return true when the list is empty or no data in the cell.

Element Type	Operator	Rule
bool	IS TRUE	Always take the first element for comparison; return false if there are no elements.
linked record		Follow the rules for the type of the display column.

When a list column is returned in a selected field, only the first 10 elements are returned.

When used in group by or order by clauses, the elements for each list will first be sorted in ascending order, then the lists will be sorted by the rules below:

- Compare the elements one by one, list with smaller element is sorted before list with larger element.
- If all elements compared in step 1 are equal, shorter list is sorted before longer list.
- Otherwise the tow lists are equal.

If a list column is passed as parameter to a formula, and the parameter expects a scalar value, the first element will be used. And if the element is a linked record, the value of its display column will be used.

When applying aggregate functions (min, max, sum, avg) to a list column, if there is only 1 element in the list, use that element; otherwise this row will not be aggregated.

NULL Values¶

NULL value is distinct from 0. It represents a missing value. The following values are treated as NULL:

- Empty cells in a table is treated as NULL.
- Values which cannot be converted to the column type will be treated as NULL.
- Empty strings ("") will be treated as NULL too. This is different from standard SQL.
- Lists are treated as NULL based on the rules described in the "List Types" section.
- Functions or formula columns that return error will be treated as NULL.

In the Where clause:

- Arithmetic operations (+, -, * etc.) on NULL values will return NULL.
- `!=`, `NOT LIKE`, `NOT IN`, `NOT BETWEEN`, `HAS NONE OF`, `IS NOT TRUE`, and `IS NULL`operations will return true when the value is NULL.
- `AND`, `OR`, `NOT` treat NULL values as false.
- Aggregate functions (min, max, sum, avg) will ignore NULL values.

In formulas, NULL values will be converted to 0 or an empty strings.

Extended Syntax¶

Use Formulas in SQL Query¶

You may use a formula syntax that's almost the same as SeaTable's formulas in SQL queries. There are a few special notes:

- Link formulas are not supported. e.g. {link.age} is invalid.
- Reference to columns should not be enclosed by curly brackets ("{}"). Don't write `select abs({column}) from talbe;`. Wirte `select abs(column) from table;`. This is consistent with standard SQL syntax.
- You can use back quote ("``") to enclose column references, when column name contains space or "-". E.g. select abs(`column-a`) from table;
- You may not use column alias in formulas. E.g. `select abs(t.column) from table as t;`is invalid.
- formulas can be use in group by and order by clauses.

A few extended formulas are supported:

- `STARTOFWEEK(date, weekStart)`: returns the first day of the week where "date" is in. "weekstart" can be used to choose "sunday" or "monday" as the first day of a week.
- `Quarter(date)`: Returns the quater of the date. Return value is 1, 2, 3 or 4.
- `ISODate(date)`: Returns ISO format string for the date. E.g. "2020-09-08".
- `ISOMonth(date)`: Returns ISO format string for the month where "date" is in. E.g. "07".

The above formulas can be used for group by week, quater, date and month. E.g. `select sum(sale) from SalesRecord group by ISODate(SalesTime);` will return the total sales amount for each day.

For more details, please refer to [./function.md].

Extended List Operators¶

Some column types in SeaTable have list values. The SeaTable UI supports a few special filters for such types. They are `HAS ANY OF`, `HAS ALL OF`, `HAS NONE OF`, `IS EXACTLY`. You may also use the same syntax to filter such columns with SQL.

For example, if column "city" is of type multi-select, and we want to find out all rows that contains "New York" or "Paris" in the "city" column, you can query: `select * from table where city has any of ("New York", "Paris");`. The list of string constant are enclosed with brackets, just like the syntax for `IN`.

Indexes¶

To improve query performance, SeaTable will automatically create indexes for the rows stored in big data storage engine. Currently, text, number, date, single-select, multiple-select, collaborators, creator, create date, modifier, modification date columns are indexed.

When you add or delete a column in a table, the index for this column is not added/removed immediately. Indexes creation and deletion are triggered in two cases:

1. When you archive the table for the next time, indexes are created for new columns and indexes for remvoed columns are removed.
2. Users may manage indexes from "index management" UI. You can open it from the "Big data management" menu in the base.

SQL function reference¶

You can use supported functions in SQL query statements.

Functions for SQL¶

With functions you can transform, calculate, combine or merge the values of other columns from the current table. On top of that, functions can refer to each other.

The functions supported in SQL are roughly the same as the set of functions supported by formulas in SeaTable.

The basic syntax of functions is as follows:

```
FunctionName(parameters...)
```

The parameters can be number,string,constants,column name or other functions. Column name cannot be an alias. If the column name contains "-", you can use "`" to enclose it.

Currently SQL query offers the following functions:

- Operands
- Mathematical functions
- Text functions
- Date functions
- Geo functions
- Logical functions
- Statistical functions

In this article, we will show you a complete overview of all functions with examples. If you are looking for a specific function, you can use the Ctrl+F to quickly find an entry on this page.

Functions with examples¶

You can use the following constants in the function:

OPERATOR	DESCRIPTION	INPUT	RESULT
e	Returns the Euler number e=2.71828...	e+1	3.71828183

OPERATOR	DESCRIPTION	INPUT	RESULT
pi	Returns the circle number Pi.	pi	3.14159265
true()	Returns the logical value 'true'.	true()	true
false()	Returns the logical value 'false'.	false()	false

Operands¶

Parameters must be strings or numbers. If a number is passed to a parameter that expects a string, it'll be converted to string, and vice versa.

OPERATOR	DESCRIPTION	INPUT	RESULT
add(num1,num2)	Adds two numeric values (num1 and num2) and returns the result.	add(1,2)	3
subtract(num1,num2)	Subtracts one numeric value (num2) from another (num1).	subtract(5,4)	1
multiply(num1,num2)	Multiplies two numeric values.	multiply(3,4)	12
divide(num1,num2)	Divides one numeric value (num1) by another (num2).	divide(3,2)	1.5
mod(num1,num2)	Calculates the remainder of a division.	mod(15,7)	1

OPERATOR	DESCRIPTION	INPUT	RESULT
power(num1,num2)	Calculates the power (num2) of a number (num1).	power(3,2)	9
greater(num1,num2)	Checks if a numeric value (num1) is greater than another (num2) and returns the logical value 'true' or 'false'.	greater(2,3)	false
lessthan(num1,num2)	Checks if a numeric value (num1) is less than another (num2) and returns the logical value 'true' or 'false'.	lessthan(2,3)	true
greatereq(num1,num2)	Checks whether a numeric value (num1) is greater than or equal to another (num2) and returns the logical value 'true' or 'false'.	greatereq(2,3)	false
lessthaneq(num1,num2)	Checks whether a numeric value (num1) is less than or equal to another (num2) and returns the logical value 'true' or 'false'.	lessthaneq(2,3)	false
equal(num1,num2)	Checks if two values (number1, number 2) are equal and returns the logical value 'true' or 'false'.	equal(`Old price`, `New price`)	false
unequal	Checks whether two values (number1,	unequal(`Old price`, `New price`)	true

OPERATOR	DESCRIPTION	INPUT	RESULT
	number2) are not equal and returns the logical value 'true' or 'false'.		
concatenate(string1, string2, ...)	Combines several character strings (string1, string 2, ...) into one character string.	concatenate(`Supplier`, " has the product ", `Product`)	Microsoft has the product Windows

Mathematical functions¶

Parameters must be numbers. If string is passed to a parameter, it'll be converted to number.

OPERATOR	DESCRIPTION	INPUT	RESULT
abs(number)	Returns the absolute value of a number.	abs(-2)	2
ceiling(number, significance)	Rounds a number to the nearest integer or to the nearest multiple of the specified significance. If either argument is non-numeric, the formula returns an empty value.	ceiling(2.14)	3
	If the number is an exact multiple of the significance, then no rounding occurs. If the number and the significance are negative, then the rounding is away from 0. If the number is negative and the significance is positive, then the rounding is towards 0.	ceiling(-2.14, 4)	0
even(number)	Assigns a real number to the nearest larger even number.	even(2.14)	4

OPERATOR	DESCRIPTION	INPUT	RESULT
exp(number)	Exponential function for Euler's number e. Returns the value of e given high (number).	expr(1)	2.71828...
floor(number, significance)	Rounds a number to the nearest integer or to the nearest multiple of the specified significance. If either argument is non-numeric, the formula returns an empty value.	floor(2.86)	2
	If the number is an exact multiple of the significance, then no rounding takes place. If the sign of the number is positive, then the rounding is towards 0. If the sign of the number is negative, then the rounding is away from 0.	floor(-3.14, 5)	-5
int(number)	Assigns the nearest smaller integer to a real number.	int(-3.14)	-4
lg(number)	Logarithm function (number) with 10 as base.	lg(100)	2
log(number, base)	Logarithm function (number) with definable base.	log(81, 3)	4
	But if no base is given, this function works exactly like lg(), with 10 as base.	log(1000)	3
odd(number)	Assigns a real number to the nearest larger odd number.	odd(-2.14)	-1
round(number, digits)	Rounds a number to the nearest integer. If no decimal place (digits) is specified, the number is rounded to the 1st digit to the left of the decimal point.	round(3.14)	3

OPERATOR	DESCRIPTION	INPUT	RESULT
	If a positive decimal place (digits) is given, the digit to the right of the decimal point is rounded.	round(3.14, 1)	3.1
	If a negative decimal place (digits) is given, is rounded to the left of the decimal point.	round(3.14, -3)	0
rounddown(number, digits)	Rounds a number towards zero. If no decimal place (digits) is given, the number is rounded to the 1st digit left of the decimal point.	rounddown(3.12, 1)	3.1
roundup(number, digits)	Rounds a number from zero to the nearest whole number. If no decimal place (digits) is given, the number is rounded to the 1st digit left of the decimal point.	roundup(-3.15)	-4
sign(number)	Checks whether a number is greater, equal or less than 0. Returns the values 1, 0 and -1 respectively. In other words: it returns the sign of a number, for '+', 'zero' and '-' with 1, 0, and -1 respectively.	sign(-2)	-1
sqrt(number)	Returns the square root of a number.	sqrt(81)	9

Text functions¶

OPERATOR	DESCRIPTION	INPUT	RESULT
exact(string1, string2)	Checks whether two character strings (string1, string2) are	exact('SeaTable', 'Seatable')	false

OPERATOR	DESCRIPTION	INPUT	RESULT
	exactly identical. Returns the values 'true' or 'false' respectively. Case sensitive.		
find(findString, sourceString, startPosition)	Returns the start position of a string (findString) within another string (sourceString). It is case sensitive. Without find, 0 is returned. If the start position (startPosition) is given as decimal, it is rounded down. If the cell in the column for the keyword (findString) is still empty, 1 is returned. If the cell in the column for the target string (sourceString) is still empty, an empty value ('') is returned.	find('Sea', 'seaTable', 1)	0
	The search will start from the given 'startPosition'. This 'startPosition' has no influence on the result: it always returns	find('table', 'big table', 4)	5

OPERATOR	DESCRIPTION	INPUT	RESULT
	the absolute start position. If the 'startPosition' of the character string to be searched for (findString) is given after the actual start position of the character string (sourceString), 0 is returned, since nothing was found from this position.		
left(string, count)	Returns the specified number (count) of characters at the beginning of a string.	left('SeaTable', 3)	Sea
len(string)	Returns the number of characters in a string.	len('SeaTable')	8
lower(string)	Converts a character string to lower case letters.	lower('German)	german
mid(string, startPosition, count)	Returns the specified number (count) of characters from the specified start position (startPosition) of a string.	mid('SeaTable is the best', 1, 8)	SeaTable

OPERATOR	DESCRIPTION	INPUT	RESULT
	Start position (startPosition) and count must not be empty, negative or zero. However, if start position (startPosition) and number (count) are given as decimal, they are rounded down. Too much count is ignored.	mid('SeaTable is the best.', 10.9, 27.3)	is the best.
replace(sourceString, startPosition, count, newString)	Replaces a part (count) of a character string (sourceString) from a certain start position (startPosition) with another character string (newString). The number (count) of characters is only taken into account for the old string (sourceString), but not for the new string (newString).	replace('SeaTable is the best.', 1, 8, 'Seafile')	Seafile is the best.
	If number (count) is given as zero, the new string (newString) is simply added to the old string (sourceString) from the start	replace('SeaTable is the best.', 1, 0, 'Seafile')	SeafileSeaTable is the best.

OPERATOR	DESCRIPTION	INPUT	RESULT
	position (startPosition).		
rept(string, number)	Repeats a string as often (number) as specified.	rept('Sea ', 3)	SeaSeaSea
right(string, count)	Returns the specified number (count) of characters at the end of a string.	right('SeaTable', 5)	Table
search(findString, sourceString, startPosition)	Returns the start position of a string (findString) within another string (sourceString). It is not case-sensitive. Without find, 0 is returned. If the start position (startPosition) is given as decimal, it is rounded down. If the cell in the column for the keyword (findString) is still empty, 1 is returned. If the cell in the column for the target string (sourceString) is still empty, an empty value (") is returned.	search('Sea', 'seaTable', 1)	1

OPERATOR	DESCRIPTION	INPUT	RESULT
	The search will start from the given 'startPosition'. This 'startPosition' has no influence on the result: it always returns the absolute start position. If the 'startPosition' of the character string to be searched for (findString) is given after the actual start position of the character string (sourceString), 0 is returned, since nothing was found from this position.	search('table', 'big table', 6)	0
substitute(sourceString, oldString, newString, index)	Replaces existing text (oldString) with new text (newString) in a string (sourceString). If there is more than one text (oldString) in the string (sourceString), only the 'index'-th text is replaced. The text is case-sensitive.	substitute('SeaTableTable', 'Table', 'file', 1)	SeafileTable

OPERATOR	DESCRIPTION	INPUT	RESULT
	If the index is given as 0 or not, all found text (oldString) will be replaced by the new text (newString).	substitute('SeaTableTable', 'Table', 'file')	Seafilefile
T(value)	Checks whether a value is text. If so, the text is returned. If no, the return value is empty.	T('123')	123
text(number, format)	Converts a number into text and formats it in the specified format. The format can be percent and number as well as dollar, euro and yuan.	text(150, 'euro')	€150
	When a number is converted directly to percent, its absolute value is retained. In other words, 50 is converted into 5000%. But if you want 50%, you have to divide the number by 100 before the conversion.	text(50, 'percent')	5000%
trim(string)	Removes spaces at the beginning	trim(' SeaTable ')	SeaTable

OPERATOR	DESCRIPTION	INPUT	RESULT
	and end of a string.		
upper(string)	Converts a string to uppercase letters.	upper('German)	GERMAN
value(string)	Converts a text (string) representing a number into a number.	value('123')	123

Date functions¶

When passing a parameter with time or date type, you can specify a contant in "2021-09-01 12:00:01" or "2021-09-01" format. When you query the result of a date function in SQL, the result will be converted to a string in RFC3339 format, e.g. "2021-09-03T00:00:00+02:00". Please note that if a date function returns a date type, it cannot be used as parameter for text or maths functions.

OPERATOR	DESCRIPTION	INPUT	RESULT
date(year, month, day)	Returns a date in international format (ISO) from entered year, month and day. If the year is entered with two digits, it is automatically understood as a year in the 1900s. If the number of the month or day is too large (greater than 12 or 31 respectively), these months or days are	date(2021, 1, 3)	2021-01-03T00:00:00+02:00

OPERATOR	DESCRIPTION	INPUT	RESULT
	automatically converted to the next year or month.		
dateAdd(date, count, unit)	Adds the specified number (count) of years ('years'), months ('months'), weeks ('weeks'), days ('days'), hours ('hours'), minutes ('minutes') or seconds ('seconds') to a date/time ('date').	dateAdd('2020-02-03', 2, 'days')	2020-02-05T00:00:00+02:00
	Tip: if you want to add a complex duration (count) such as 1 day 12 hours, you can convert it to e.g. 24+12=36 hours ('hours') and enter it into the formula as a uniform duration (count). The duration is converted to the smallest unit: in this case, hours.	dateAdd('2020-09-04 13:05:18', 36, 'hours') ODER dateAdd(`form submission`, 36, 'hours')	2020-09-06T01:05:18+02:00
datedif(startDate, endDate, unit)	Calculates the seconds, days, months, or years between two date values. The optional unit argument can be one of the following: S (seconds), D (full	dateDif('2018-01-01', '2020-01-01')	2

OPERATOR	DESCRIPTION	INPUT	RESULT
	days), M (full months), Y (full years), YD (full days, ignoring years), YM (full months, ignoring days and years), MD (full days, ignoring months and years). If the startDate is empty, a default value of "1900-01-01" will be set. If both date values are empty, it will return 0.		
	The optional unit argument can be one of the following: S (seconds), D (full days), M (full months), Y (full years), YD (full days, ignoring years), YM (full months, ignoring days and years), MD (full days, ignoring months and years).	dateDif('2019-10-11', '2020-12-12', 'M')	14
day(date)	Returns the day of a date as a number. The returned number is between 1 and 31.	day('2020-01-03)	3
eomonth(startDate, months)	Determines the date of the last day of the month that is the	eomonth('2020-01-01', 1)	2020-02-29T00:00:00+02:00

OPERATOR	DESCRIPTION	INPUT	RESULT
	specified number (months) of months after the specified date (startDate). If the number (months) is given as 0, the last day of the month is simply determined.		
	If the number (months) is given as negative, the date of the last day of the month that contains the absolute number (months) of months before the specified date (startDate) is determined.	eomonth('2020-01-01', -1)	2019-12-31T00:00:00+02:00
hour(date)	Returns the hour of a date as a number. The number returned is between 0 and 23.	hour('2020-02-14 13:14:52)	13
	If no hour is contained in the time specification (date), 0 is returned.	hour('2020-02-14)	0
hours(startDate, endDate)	Returns the number of hours between two date values (startDate and endDate). The minutes in	hours('2020-02-14 13:14', '2020-02-14 15:14')	2

OPERATOR	DESCRIPTION	INPUT	RESULT
	the date values are not taken into account.		
	If no hours are included in the time specification (startDate or endDate), 0 o'clock on this day is automatically assumed.	hours('2020-02-14', '2020-02-14 15:14')	15
minute(date)	Returns the minutes of a time specification (date) as a number. The number returned is between 0 and 59.	minute('2020-02-14 13:14:52	14
	If no minutes are included in the time (date), 0 is returned.	minute('2020-02-14)	0
month(date)	Returns the month of a date as a number. The returned number is between 1 (January) and 12 (December).	month('2020-02-14 13:14:52)	2
months(startDate, endDate)	Returns the number of months between two date values (startDate and endDate). The	months('2020-02-01 13:14', '2020-03-31 15:54')	1

OPERATOR	DESCRIPTION	INPUT	RESULT
	days and time in the date values are not taken into account.		
	If no month is given in the date values (startDate, endDate), January is automatically assumed to be the month.	months('2020', '2021')	12
networkdays(startDate, endDate, holiday1, holiday2, ...)	Returns the number of full working days between two dates (startDate and endDate). You can also define holidays other than Saturday and Sunday (holiday1, holiday2, etc.), which are also deducted. If you do not want to include public holidays, you can simply omit these parameters.	networkdays('2020-01-01', '2020-01-07','2020-01-01')	4
	Please note that the specified last day (endDate) is also included in the formula. That means, as in this formula, three working days are counted: the 7th,	networkdays('2020-09-07', '2020-09-09')	3

OPERATOR	DESCRIPTION	INPUT	RESULT
	8th and 9th of September, 2020.		
now()	Returns the current date and time. This column is only updated automatically when the Base is reloaded.	now()	2020-09-07T12:59+02:00
second(date)	Returns the seconds of a time (date) as a number. The number returned is between 0 and 59.	second('2020-02-14 13:14:52')	52
today()	Returns the current date. This column is only updated automatically if the Base has been reloaded.	today()	2020-09-07T00:00:00+02:00
	This function is handy for calculating time between a certain date & time and now. On each reload or recalculation of the Base, the calculation is updated.	networkdays('2020-09-01', today())	4
weekday(date, weekStart)	Returns the weekday of a date as a number.	weekday('2020-01-01', 'Monday')	3

OPERATOR	DESCRIPTION	INPUT	RESULT
	The returned number between 1 and 7, where you can define the first day of the week (weekStart): Monday ('Monday') or Sunday ('Sunday' or omitted, since the start as Sunday is the default). A third option is not possible. Upper/lower case is not considered.		
	If no 'weekStart' is given or if a 'weekStart' other than 'Monday' or 'Sunday' is given, it is always assumed to be 'Sunday'. So if it should be 'Monday', enter 'Monday'; if it should be 'Sunday', you can omit this parameter.	weekday('2020-01-01', 'Thursday') OR weekday('2020-01-01')	4
weeknum(date, return_type)	Returns the absolute week number of a date as a number. The returned number is between 1 and 53, where you can define the first day of the week	weeknum('2020-01-12', 11)	2

OPERATOR	DESCRIPTION	INPUT	RESULT
	(return_type). Enter the number 1 or 2, or 11 to 17, and 21 as "return_type" to define the start of a week: 1/Sunday、 2/Monday、 11/Monday、 12/Tuesday、 13/Wednesday、 14/Thursday、 15/Friday、 16/Saturday、 17/Sunday. If you want the week number to be returned according to ISO standard, specify the number of 21 as "return_type", or use the function isoweeknum.		
	If no 'return_type' is given, it is always assumed to be 'Sunday'.	weeknum('2020-01-12')	3
year(date)	Returns the year of a date as a number.	year('2020-01-01')	2020
startofweek(date, weekStart)	Returns the first day of the week in which the date	startofweek('2021-04-28')	2021-4-25T00:00:00+02:00

OPERATOR	DESCRIPTION	INPUT	RESULT
	is located. WeekStart defaults to sunday, or it can be set to monday.		
quarter(date)	Returns the quarter of the date, the return value is 1, 2, 3, 4.	quarter('2021-01-01')	1
isodate(date)	Returns the ISO string representation of the date.	isodate('2021-01-01 11:00:00')	2021-01-01
isomonth(date)	Returns the ISO string representation of the year and month	isomonth('2021-01-01 11:00:00')	2021-01

Geo functions¶

OPERATOR	DESCRIPTION	INPUT	RESULT
country(geolocation)	Returns the country or region of a Geolocation column. (Since version 5.1.0)	country(column_name)	Germany

Logical functions¶

OPERATOR	DESCRIPTION	INPUT	RESULT
and(logical1, logical2, ...)	Checks if all arguments (logical1, logical2, ...) are true (valid, not empty and not	and(1, '', 2)	false

OPERATOR	DESCRIPTION	INPUT	RESULT
	equal to zero). If yes, 'true' is returned, otherwise 'false'.		
if(logical, value1, value2)	Checks if an argument (logical) is true and if yes, returns the first value (value1) and if no, returns the second value (value2).	if(1>2, 3, 4)	4
	For the condition (logical) only a comparison with is allowed. If you enter only condition (logical) and the first value (value1): it will return the first value (value1) if the condition (logical) is true; and it will return an empty value (") if the condition (logical) is false.	if(`Budget`>`Price`, 'Yes')	Yes
ifs(logical1, value1, logical2, value2, ...)	Checks if one or more conditions (logical1, logical2, ...) are true and returns a value (value1, value2, ...) that matches the first TRUE condition.	ifs(1>2, 3, 5>4, 9)	9
not(boolean)	Inverts the logical value (boolean). In other words: converts 'true' to 'false' and 'false' to 'true'.	not(and(1, ", 2))	true
or(logical1, logical2, ...)	Checks if at least 1 of the arguments (value1, value2, ...) is true (valid, not empty and not equal to zero), and returns 'true' in this case. If all arguments are false, then returns 'false'.	or(1,",2)	true
switch(logical, matcher1,	Evaluates an expression (logical) against a list of	switch(`grades`, 1, 'very good', 2, 'good',	very good

OPERATOR	DESCRIPTION	INPUT	RESULT
value1, matcher2, value2, ..., default)	values (matcher) and returns the result (value) corresponding to the first matching value. If there is no match, an optional default value is returned. At least 3 parameters (logical, matcher, value) must be specified.	3, 'satisfactory', 4, 'passed', 'failed')	
	If there are several identical values in the value list (matcher), only the first hit is taken into account.	switch(int(68/10), 6, 'OK', 6, 'KO')	OK
xor(logical1, logical2, ...)	Returns the contravalence of all arguments. In other words, checks if the number of true arguments is (logical) odd and returns 'true'.	xor(1, 0, 2\<1)	false

Statistical functions¶

OPERATOR	DESCRIPTION	INPUT	RESULT
average(number1, number2, ...)	Returns the average of the numbers (number1, number2, ...)	average(1, 2, 3, 4, 5)	3
counta(textORnumber1, textORnumber2, ...)	Counts the number of non-empty cells (textORnumber1, textORnumber2, ...). These cells can be text or numbers. In this example, 1 and 2 are numbers, '3' is text, and '' is an empty value.	counta(1, '', 2, '3')	3

OPERATOR	DESCRIPTION	INPUT	RESULT
countall(textORnumber1, textORnumber2, ...)	Counts the number of elements (textORnumber1, textORnumber2, ...) including numbers (1, 2), text ('3') and empty cells ('').	countall(1, '', 2, '3')	4
countblank(textORnumber1, textORnumber2, ...)	Counts the number of empty cells.	countall(1, '', 2, '3')	1
countItems(column)	Counts the number of items in a column. The supported column types are multiple select, collaborator, file, image. (available since version 2.7.0)	countItems(column_name)	2