Cheatsheet – Operators and Functions

QuickSight offers multiple options to add calculations to your data. This can be confusing, especially if you just started working with QuickSight. So whenever you are not sure what options you have or maybe also don't have when it comes to adding calculations to your data, just take a look at this Cheatsheet.

Arithmetic Operators

These are the basic functions, so "+", "-", "*" and "/" which can simply be applied by adding it to a calculated field.

Comparison Operators

Name	Function	Description
equal/not equal	"=" vs. "!="	Analysis if a value is equal/not equal to another value
greater than/less than	">" vs. "<"	Analysis if a value is greater than/less than another value
greater/less than or equal to	">=" vs. "<="	Analysis if a value is greater/less than or equal to another value
and/or	"AND" vs. "OR"	Analysis if either multiple or a single condition is true; normally applied for example in combination with a conditional function

Conditional Functions

Name	Syntax	Description
coalesce	"coalesce(expression, expression [, expression, ...])"	Returns the value of the first argument that is not null
ifelse	"ifelse(if, then [, if, then ...], else)"	Evaluates a set of "if, then" expression pairings
isNotNull/ Null	"isNotNull(expression)" "isNull(expression)"	Evaluates an expression to see if it is not null/if it is null (result = "true" or "false")
nullif	"nullIf(expression, expression)"	Compares two expressions. If they are equal, the function returns null

String Functions

Name	Syntax	Description
concat	"concat(expression, expression [, expression ...]))"	Concatenates two or more strings
left/ right	"left(expression, limit)" "right(expression, limit)"	Returns the leftmost/rightmost characters from a string; limit = number of characters returned
strlen	"strlen(expression)"	Returns the number of characters in a string, including spaces
subString	"substring(expression, start, length)"	Returns the characters in a string, starting at the location specified by the start argument and proceeding for the number of characters specified by the length arguments

replace	“replace(expression, substring, replacement)”	Replaces part of a string with another string defined by you
trim	“trim(expression)”	Removes both preceding and following whitespace from a string
ltrim/ rtrim	“ltrim(expression)” “rtrim(expression)”	Removes preceding whitespace from a string
parseDecimal/ parseInt/ parseDate	“parseDecimal(expression)” “parseInt(expression)” “parseDate(expression, ['format'])”	Parses a string to determine if it contains an integer/decimal/date value; Only rows that contain an integer/decimal/date value will be kept, the remaining rows will be skipped parseDate is not supported for SPICE data sets
locate	“locate(expression, substring, start)	Locates a substring that you specify within another string, and returns the number of characters until the first character in the substring
toLower/ toUpper/ toString	“toLower(expression)” “toUpper(expression)” “toString(expression)”	Formats all characters of a string in lowercase/uppercase Formats the input expression as a string

Numeric Functions

Name	Syntax	Description
ceil/ floor	“ceil(decimal)” “floor(decimal)”	Rounds a decimal value to the next highest/lowest integer
round	“round(decimal, scale)”	Rounds a decimal value to the closest integer (if no scale is specified) or according to the scale specified
decimalToInt	“decimalToInt(decimal)”	Converts a decimal value to the integer data type
intToDecimal	“intToDecimal(int)”	Converts an integer value to the decimal data type

Date Functions

Name	Syntax	Description
dateDiff	“dateDiff(date, date)”	Returns the difference in days between two date fields
extract	“extract('period', date)	Returns a specified portion of a date value
truncDate	“truncDate('period', date)”	Returns a date value that represents a specified portion of a date
now	“now()”	File/sales force datasets: Returns UTC date and time Database (direct query): Returns the current date and time specified by the database server Not supported for SPICE datasets
formatDate	“formatDate(date, ['format'], ['time_zone'])”	Formats a date using a pattern you specify
epochDate	“epochDate(epochdate)”	Converts an epoch date into a standard date