Updated as of
March 18, 2021

KURVE CORE

Custom Column
User Guide

tjene

Table of Contents

Now that you know how to create a custom column as explained in our Kurve Core user guide and how-to video, this document will provide examples of expressions to help you customize your report and make the most out of custom columns.

Column Type	Description
Text Column	Text columns should be used for alphabetic characters. For example, if you are creating a column to display months, it would be a text column displaying months January through December based on the source column. A text column will filter for unique values.
Number Column	A number column would be used for columns that will result in numeric characters. For example, if you are calculating a percentage you would use a number column. Using a number column will allow you to filter by greater than, less than, equal to, not equal to, etc.
Date Column	A date column will be used for all custom columns involving dates. For example, if you are adding a column to show Today's Date, the column type would be Date. A date column filter will provide a calendar for you to choose what you would like to filter for.
Date/Time Column	A date/time column will be used for all custom columns involving dates in which the source column has the time of day as well. For example, 10/09/2018 12:01:47. A date/time column filter will provide a calendar and time selection for you to choose what you would like to filter for.
Colour Column	Colour columns are used for colour coding specific columns. Any new column can be colored.

Basic Expressions

Number

Make a percentage

If I have a column that is already a decimal, I could add that column into the expression tab and then multiply by 100.

```
numCol('decimal')*100
```

If you want to find the percentage change between two numbers:

```
(numCol('Score')/numCol('Total'))*100
```

Add two columns together

To add two columns together, I would select the first column from my source column, add it as a number, click or type the plus sign, and then select the second column I would like to add.

For example, to get a total number of parking spaces in the example below:

```
numCol('Parking Spaces (Open)')+numCol('Parking Spaces (Covered)')
```

Text

Check One Condition – true/false output

This would be used to determine if something is true or false according to your parameter.

The output of the following expression will be true if the value for that row in the State/Province column is New York. If it isn't New York, the output will be false. Don't forget the double equal sign!

```
col('State/Province')=="New York"
```

This also works for checking if something is greater than or less than! The example below will return true if the number is greater than 10,000. When adding the source column into the expression tab, add it as a number to ensure it is the right format.

```
numCol('Site Gross Area')>10000
```

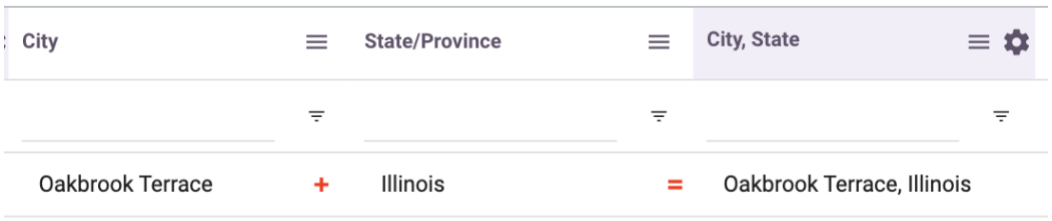
This works for checking dates as well.

```
dateCol('Must Exercise Option By')>Today()
```

Adding Two Columns Together

If you would like to combine two text columns you can easily do so in the expression tab. Let's say we want City and State in the same column, separated by a comma.

```
col("City") +", " +col("State/Province")
```



Date

We can also output a date. The best ways to do this are outlined in the functions section.

Colour

A color column can be created with either just one color or multiple colors according to certain conditions.

To output just one column, in the expression tab, put the color or hex code in quotations:

"Red"

To output multiple colors you would use an if statement, this will be covered in the section regarding IF statements.

Use the color picker within the operators to easily add a color into the expression box.

Functions

TRIRIGA Functions

Get Path Level
Data Type: Text
This function is used to retrieve a specific level from a TRIRIGA hierarchy path field. For example, if you have a hierarchy path for Space and you want to drill down to the fourth level I would add my Source column to replace path and then indicate the level with a numerical value:

GetPathLevel(path, level) → GetPathLevel(col('Hierarchy Path'), 4)

Is Blank
Data Type: Text
Sometimes the formatting of blanks gets confusing in TRIRIGA. This function returns true if the value is either “ ” or null, otherwise, it will return false.

IsBlank(value) → IsBlank(col('City'))

Date Functions

Important Note:
When outputting a date column, we recommend doing all of the operations within one custom column (including re-formatting and any use of functions). We don't recommend users chain reference date columns in multiple custom columns.
If you do need to chain them, please avoid changing the output date format within the process, and only use FormatDate function at very end of the chaining operations.
We can combine multiple functions within one column.

For example, if you would like to add time as well as reformat a date we can nest the functions together like the following expression:

FormatDate(AddTime(dateCol("Project Plan End"), 2, "seconds"),"yyyy-MM-dd hh:mm:ss")

Add Days
Data Type: Date or Date/Time
Adds a specified number of days to the given date. Can also be used to subtract days. For example, if you want to add a buffer period to an expiration date you can use this

function. The example below will return the date in 14 days from the Contract Expiration Date.

AddDays(date, numberToAdd) → AddDays(dateCol('Contract Expiration Date'), 14)

Add Time

Data Type: Date or Date/Time
Adds a specified number of a given unit of time to the given date. For example, you can add hours, days, weeks, months, etc. The following adds two weeks to the Contract Expiration Date.

AddTime(date, numberToAdd, unit) →

AddTime(dateCol('Contract Expiration Date'), 2, "weeks")

Add TRIRIGA Duration

Data Type: Number
Adds a TRIRIGA duration to a given date and returns the resulting date. For example, if you have a field Due Within that is a duration field and a start date of works tasks, you can use this function to determine when the task is supposed to be due.

AddTRIDuration(date, duration)→ AddTRIDuration(dateCol('Start Date'), col('Due Within'))

Convert Duration

Data Type: Number
Converts a given duration from one unit of time to another. Supports milliseconds, seconds, minutes, hours, days, weeks, months, and years. The following expressions will convert the column Days Between from the number of days to the number of weeks.

ConvertDuration(duration, input, output) →

ConvertDuration(numCol('Days Between'), "days", "weeks")

Convert Milliseconds

Data Type: Number
TRIRIGA uses Epoch time to manage their dates so sometimes your output in a custom column may provide milliseconds. This function will format a given duration into another

specified unit of time. Supports years, months, week, days, hours, minutes, and seconds.

ConvertMilliseconds(duration, output) → ConvertMilliseconds(4.32e+8, "days")

Convert TRIRIGA Duration

Data Type: Number
Converts a duration from TRIRIGA into a duration in milliseconds. TRIRIGA Duration fields return the date in a text format, you can use this function to put it into a numerical value.

ConvertTRIDuration(duration) → ConvertTRIDuration(col('Due Date'))

Days Between

Data Type: Number
Returns the number of calendar days between two given dates. The showNegative option will control whether or not negative days will be displayed or shown as a 0. The default is true, which will show the negative numbers.

DaysBetween(date1, date2) →

DaysBetween(dateCol('Planned Start Date'), dateCol('Actual Start'))

You can also use functions in combination with another to find the number of days between Today and a given date:

DaysBetween(Today(), dateCol('Planned Start Date'))

FilterDay(date, range)

Data Type: Text
Returns true if the date entered is within the range of days in relation to the current date. If not specified, default range is 30 days. In the example below, it will return true if the In-Service date was in the last 15 days.

FilterDay(date, range) → FilterDay(dateCol('In Service Date'), -15)

FilterMonth(date, range)

Data Type: Text

Returns true if the date entered is within the range of months in relation to the current date. If not specified, default range is 3 months. In the example below, it will return true if the In-Service date is in the next 4 months.

FilterMonth(date, range) → FilterMonth(dateCol('In Service Date'), 4)

FilterYear(date, range)
Data Type: Text
Returns true if the date entered is within the range of year(s) in relation to the current date. If not specified, default range is 1 year. In the example below, it will return true if the In-Service date is in the next 2 years.

FilterYear(date, range) → FilterYear(dateCol('In Service Date'), 2)

Format Date
Data Type: Date or Date/Time
You can reformat a date in a specific format, for example, "MM/dd/yyyy", "yyyy-MM-dd hh:mm:ss". This function is case sensitive so ensure you check the results. In particular, it is important to note that “mm” would output milliseconds, whereas “MM” will output month.

FormatDate(date,format) → FormatDate(dateCol('In Service'),"yyyy/MM/dd")

For a list of supported date formats see here.

Within TRIRIGA, the following Date/Time formats are not supported:

- Any format that contains ‘h’ and ‘min’ (the single quote matters). For example, “dd-MM-yyyy HH’h’mm’min’ “
- Any format in which the month uses 5 numbers. For example, “MMMM dd, yyyy hh:mm”
    - IBM uses 5 M for full month name, the actual Unicode standard is 4

Get Day
Data Type: Number
Determines the day number of the given date. Can optionally return days as a 2 digit number (e.g. 01, 02) if addZeroes is true.

GetDay(date, addZeroes) → GetDay(dateCol('In Service'), true)

Get Day of Week

Data Type: Text Column
Use this function to return the day name of a given date (e.g. Monday)

GetDayOfWeek(date) → GetDayOfWeek(dateCol('In Service'))

Get Day of Year

Data Type: Text Column
Use this function to return the day number in a year of a given date (e.g. February 1 = 32)

GetDayOfYear(date) → GetDayOfYear(dateCol('In Service'))

Get End of Month

Data Type: Date or Date/Time
Use this function to the return the date at the last day of the given date's month. The time is set to 1 millisecond before midnight. This can be used to group dates on a monthly basis.

GetEndOfMonth(date) → GetEndOfMonth(dateCol('In Service'))

Get End of Year

Data Type: Date or Date/Time
Returns the date at the last day of the given date's year. The time is set to 1 millisecond before midnight. This can be used to group dates on a yearly basis.

GetEndOfYear(date) → GetEndOfYear(dateCol('In Service'))

Get First Of Month

Data Type: Date or Date/Time
Returns the date at the first day of the given date's month. This can be used to group dates on a monthly basis.

GetFirstOfMonth(date) → GetFirstOfMonth(dateCol("Active Start Date"))

Get First Of Year

Data Type: Date or Date/Time

Returns the date at the first day of the given date's year. This can be used to group dates on a yearly basis.

GetFirstOfYear(date) → GetFirstOfYear(dateCol("Active Start Date"))

Get Month

Data Type: Number

Determines the month number of the given date. Can optionally return months as a 2 digit number (e.g. 01, 02) if addZeroes is true. This will ensure column is sorted appropriately and can be graphed in order. If you input false, it will add the numbers as is.

GetMonth(date, addZeroes) → GetMonth(dateCol('In Service'), true)

Get Month Name

Data Type: Text Column

Use this function to return the full month name of a given date (e.g. January).

GetMonthName(date) → GetMonthName(dateCol('In Service'))

Get Quarter

Data Type: Number

Use this function to return the year quarter of the given date. Based on the calendar year where: January – March = Q1, April – June = Q2, July – September = Q3, October – December = Q4.

GetQuarter(date) → GetQuarter(dateCol('In Service'))

Get Week

Data Type: Number

Determines the week number of a given date. Can optionally return weeks as a 2 digit number (e.g. 01, 02) if addZeroes is true. This will ensure the column is sorted accordingly and can be graphed in order. If you input false, it will add the number as is.

GetWeek(date, addZeroes) → GetWeek(dateCol('In Service'),true)

Get Weeks in a Month

Data Type: Number
Returns the number of calendar weeks in the given date's month. The first day of the week can be optionally specified with a number from 0-6, the default is 0, for Sunday.

GetWeeksInMonth(date, weekStartDate) → GetWeeksInMonth(dateCol('In Service'), 1)

The expression above will return the number of weeks in the given month of In Service Date starting Monday.

Get Year

Data Type: Number
Determines the year of the given date

GetYear(date) → GetYear(dateCol('In Service'))

Is Date After

Data Type: Text
Returns true if the given date occurs after the date to compare. Returns N/A if dates are missing or invalid. For example, in the example below, if Today's date is after the expiration date, the result will be True.

IsDateAfter(date, dateToCompare) → IsDateAfter(Today(), dateCol('Expiration Date'))

Is Date Before

Data Type: Text
Returns true if the given date occurs before the date to compare. Returns N/A if dates are missing or invalid. For example, in the example below, if Today's date is before the expiration date, the result will be True.

IsDateBefore(date, dateToCompare) → IsDateBefore((Today(), dateCol('Expiration Date'))

Is Date Between

Data Type: Text
Returns true if the given date occurs between the start and end dates. Returns N/A if dates are missing or invalid.

IsDateBetween(date, start, end) → IsDateBetween(Today(), dateCol('Lease Start Date'), dateCol('Expiration Date'))

Now

Data Type: Date or Date/Time
If you want to return the current date and time, use this function.

Now()

Subtract TRIRIGA Duration

Data Type: Number
Subtracts a TRIRIGA duration from a given date and returns the resulting date.

SubtractTRIDuration(date, duration) → SubtractTRIDuration(dateCol('Start Date'), col('Due Within'))

Time Between

Data Type: Number
Returns the amount of time between two given dates in the specified unit of measurement. For example, days, months, weeks, etc. The example below will return the number of weeks between the planned start and actual start date.

TimeBetween(date1, date2, units) →
TimeBetween(dateCol('Planned Start'), dateCol('Actual Start'), “weeks”)

Today

Data Type: Date or Date/Time

This function will return the current date with the time at 00:00.

```
Today()
```

Number Functions

Format Currency
Data Type: Number
This function will format numbers as money values rounded to the specified level of precision. If not specified, the default precision is 2. This is another way to format numbers instead of using the format functionality.

FormatCurrency(number, symbol, thousandSeparator, decimalSeparator, precision)

→ FormatCurrency(numCol('Usable Area'), "$", ",", ".", 2)

Example: 20000.023 → $20,000.02

Maximum
Data Type: Number
Find the maximum number between two columns with the Max function. Replace the numx with as many numbers or columns you wish and it will output the number that is the greatest.

Math.max(num1, num2, ...) → Math.max(numCol('Usable Area'), numCol('Rentable Area'))

Minimum
Data Type: Number
Find the minimum number between two columns with the Min function. Replace the numx with as many numbers or columns you wish to determine the min.

Math.min(num1, num2, ...) → Math.min(numCol('Usable Area'), numCol('Rentable Area'))

Round a Number
Data Type: Number
Returns a number rounded to the specified level of precision. If not specified, default precision is 0. The following will round the Rentable Area column to two decimals in the new column.

RoundNumber(number, precision) → RoundNumber(numCol('Rentable Area'), 2)

Text Functions

Delimiter/Splitter
Data Type: Any
Converts a string delimited by a specified input character and replaces the input delimiter with the chosen output character. The example below will remove the special characters between the path and replace it with spaces.

Delimiter(input, inputSplit, outputSplit) → Delimiter(col('Request Class Hierarchy Path'), "\\", " ")

Group Nulls
Data Type: Text
Converts any null values to empty strings. Can be used to force blank values to be grouped together.

GroupNulls(column) → GroupNulls(col('City'))

PadEnd(input, length, fill)
Pads an input string up to the specified length with the given fill character. Fill characters are inserted at the end of the input string. The below example would add a zero to the end of the Active Start Month column if the length of the column is 1. If the length of the column is 2 (i.e. 10), it will do nothing.

PadEnd(input, length, fill) → PadEnd(input, length, fill) → PadEnd(Col("Active Start Month"),2,0)

PadStart(input, length, fill)
Pads an input (source column) string up to the specified length with the given fill character. Fill characters are inserted at the start of the input string. Note, the length is how long you want the output to be, so if the output is already that long, nothing will happen in the function. The below example would add a zero to the start of the Active Start Month column if the length of the column is 1. If the length of the column is 2 (i.e. 10), it will do nothing.

PadStart(input, length, fill) → PadStart(Col("Active Start Month"),2,0)

Truncate

Data Type: Text

Extracts a given number of characters from a text input, starting at the specified character. The first character is counted as 0. If length is not provided, will extract from start to the end. In the example below, if my column showed "CALIFORNIA" and then I used the truncate function per below, my output would show "CA"

Truncate(input, start, length) → Truncate(col('State/Prov'), "0", "2")

IF statements

Both Excel style IF and JavaScript if are supported. See examples below to better understand the syntax:

Text

We can use IF statements to help group data into buckets.

Text output – 2 conditions

The example below will check whether the Forecast Final is less than the Budget Original. If so it will output Under Budget, otherwise it will output Over Budget. This can be useful in coloring a graph, you can determine what's under versus over budget.

Excel:

```
IF(col('Forecast Final') < col('Budget Original') ,"Under Budget", "Over Budget")
```

JavaScript:

```
if(col('Forecast Final') < col('Budget Original')) {
'Under Budget'
} else {
'Over Budget'
}
```

Text output – 3+ conditions

The example below groups the data in buckets which can be useful for coloring a graph. We check the months until expiry column and if its less than 12 the output will be 'a. <1 year', if it's between 12 and 24 then the output will be 'b. 1-2 years', if it's between 24 and 36 then it will be 'c. 2-3 years', and anything greater than 36 will show 'd. 3+ years'. The reason we put a, b, c, d in front of each of the outputs is to easily organize the data.

Excel:

```
IF(col('Months Until Expiry') < 12, "a. <1 year", IF(col('Months Until Expiry') < 24,"b. 1-2 years",
IF(col('Months Until Expiry') < 36,"c. 2-3 years", "d. 3+ years")))
```

JavaScript:

```
if(col('Months Until Expiry') < 12){
        'a. <1 year'
} else if (col('Months Until Expiry') < 24){
        'b. 1-2 years'
} else if(col('Months Until Expiry') < 36){
        'c. 2-3 years'
} else {
        'd. 3+ years'
}
```

Number

Numerical output – 2 conditions

The following example will check whether the field is blank, if so, it will return a 0, if it is not blank it will return a 1.

Excel:

```
IF(IsBlank(numCol('Original Budget')), 0, 1)
```

JavaScript:

```
if(IsBlank(numCol('Original Budget'))) {
        0
} else {
        1
}
```

Date

Date output – 2 conditions

This checks whether there is an option, it will output the option expire date, otherwise, it will output the lease expiration date.

Excel:

```
IF(IsBlank(col('Option Name')), dateCol('Contract Expiration Date'), dateCol('Option Expiration Date'))
```

JavaScript:

```
if(IsBlank(col('Option Name'))) {
    dateCol('Contract Expiration Date')
} else {
    dateCol('Option Expiration Date')
}
```

Color

We can use IF statements to add multiple colors into one column.

Color – 2 conditions

The following formula will check whether the budget original is less than the forecast final, if it is, it will color Green, otherwise it will color it red.

Excel:

```
IF(numCol('Budget Original') < numCol('Forecast Final')),"Green", "Red")
```

JavaScript:

```
if(numCol('Budget Original') < numCol('Forecast Final')) {
    "Green"
} else {
    "Red"
}
```

Color – 3+ conditions

You can add as many colors as you wish. The following checks the percentage completion of a task and assigns Green if it's greater than 80, Yellow if it's greater than 50, and Red for everything else.

Excel:

```
IF(numCol('Percentage Complete') > 80, "Green",IF(numCol('Percentage Complete')>50,
"Yellow","Red"))
```

JavaScript:

```
if(numCol('Percentage Complete') > 80) {
        "Green"
} else if(numCol('Percentage Complete') > 50){
        "Yellow"
} else {
        "Red"
}
```

Color & Text

Function:

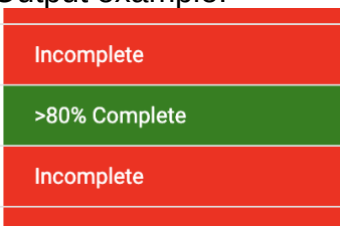
Color Column Snippet

An example of a conditional color column with text. Replace content in [] with your own conditions/text/colors

Function	Filled Out

if([CONDITION_1]){
'[CONDITION_1_TEXT] &&
[CONDITION_1_COLOR]'
 	
} else if([CONDITION_2]){
'[CONDITION_2_TEXT] &&
[CONDITION_2_COLOR]'
 	
} else {
'[DEFAULT_TEXT] &&
[DEFAULT_COLOR]'
}
	if(numCol('Percentage Complete') > 80) {
	'>80% Complete &&
	Green'
	
	} else if(numCol('Percentage Complete') > 50){
	'> 50% Complete &&
	Yellow '
	
	} else {
	'Incomplete &&
	Red '
	}

Output example:



Color Column Snippet: Referencing Other Column as Output

To embed another column as the output text, the syntax looks different. For example, let's say we want to output a column entitled "Original Budget" with specific colors and parameters:

```
if(numCol("Original Budget") < 100000) {numCol("Original Budget")  +'&& Green'
}
else {numCol("Original Budget") + '&& Red'}
```

The above statement displays how you would output a column as the output text and a color.

OR Statements

We sometimes want to combine multiple comparisons together in a conditional expression. These operators let us say things like "if both X and Y are true" or "if either X or Y are true" in our IF statement.
If we want to require that two conditions are both true, we can use the || operator.

For example, let's say we want to determine based on statuses if we want to include a line item in our report. Let's take four statuses: Draft, Active, Revision in Progress, and Closed. If it is one of the first three, I want to include it in my report. If Status is anything else, I want to be able to filter it out. We can do this by using conditional if statement and outputting true or false.

Excel:

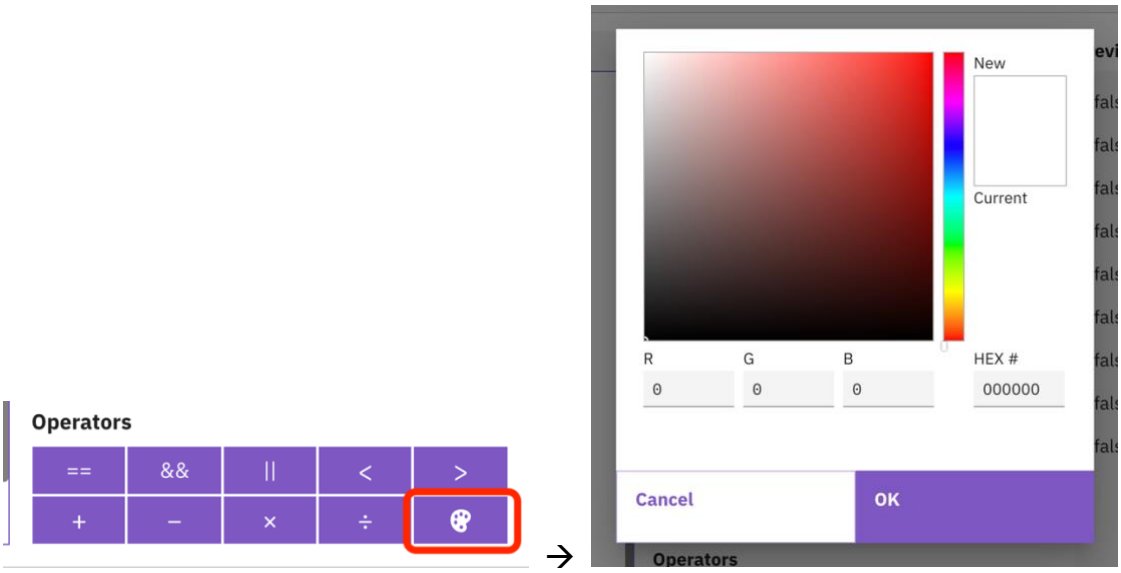
```
IF(col("Status") == "Active" || col("Status") == "Revision in Progress" || col("Status") == "Draft"
,“True”,“False”)
```

Javascript

```
if(col("Status") == "Active"
|| col("Status") == "Revision in Progress"
|| col("Status") == "Draft"
{“True” } else {“False”}
```

Color Picker (Introduced in Version 1.12)

There is an operator that pops up a color picker to easily add any color to your expression. By selecting a color and then clicking OK, the hex code of the color selected will be added to the expression.


Operators
==
&&
||
<
>
+
−
×
÷
New
Current
R
G
B
HEX #
0
0
0
000000
Cancel
OK
Operators