√× Special Functions

List of functions that can be used in Branching Logic, Calculations, Report filtering, Data Quality Module, Automated Survey Invitations, etc.

REDCap logic can be used in a variety of places, such as Branching Logic, Calculations, Report filtering, Data Quality Module, Automated Survey Invitations, and more. Special functions can be used in the logic, if desired. A complete list of ALL available functions is listed below. Listed below are some examples of common use cases where these functions might be used.

NOTICE: Please be advised that it is not possible to pipe the choice label of a multiple choice field into any of the special functions listed below. In other words, you cannot use the ':label' option, such as [my_field:label], to output the text label into a function. **The special functions only utilize the value of a field, never its label.**

Practical examples for common use cases

1. Calculate the number of days separating today's date and a date/datetime field's value in the past or future.
   `datediff([date1], 'today', 'd')`
2. Calculate a person's age based on date of birth.
   `rounddown(datediff([date_of_birth], 'today', 'y'))`
3. Calculate a person's BMI (in metric units of 'cm' and 'kg') and rounding to the first decimal place.
   `round(([weight]*10000)/(([height])^(2)), 1)`
4. Calculate a person's BMI (in English units of 'lb' and 'in') and rounding to the first decimal place.
   `round(([weight]/(([height])^(2))*703), 1)`
5. Remove a prefix and dash from the beginning of a record name (e.g., convert '4890-2318' to '2318').
   `mid([record-name], find('-', [record-name])+1, length([record-name])-find('-', [record-name])+1)`
6. Convert a person's first and last name into a username-looking format (e.g., convert 'John' and 'Doe' to 'john_doe'). We may want to trim the values just in case there were spaces accidentally entered.
   `lower( concat( trim([first_name]), '_', trim([last_name]) ) )`
7. Add leading zeros to an integer, in which the number near the end of the equation represents the maximum length of the result after adding the leading zeros (i.e., converts '7' to '007'). Note: The amount of 0s listed in '00' should be at least one less than the value of the number near the end (e.g., if '4', then we need '000'; if '3', then '00').
   `right(concat('00', [my_integer]), 3)`

Function	Name/Type of Function	Notes / Examples
if (CONDITION, VALUE if condition is TRUE, VALUE if condition is FALSE)	**If/Then/Else conditional logic**	Return a value based upon a condition. If CONDITION evaluates as a true statement, then it returns the first VALUE, and if false, it returns the second VALUE. E.g. if([weight] > 100, 44, 11) will return 44 if "weight" is greater than 100, otherwise it will return 11.
datediff ([date1], [date2], "units", returnSignedValue)	**Datediff**	Calculate the difference between two dates or datetimes. Options for 'units': 'y' (years, 1 year = 365.2425 days), 'M' (months, 1 month = 30.44 days), 'd' (days), 'h' (hours), 'm' (minutes), 's' (seconds). The parameter 'returnSignedValue' must be either *true* or *false* and denotes whether you want the returned result to be either signed (have a minus in front if negative) or unsigned (absolute value), in which the default value is FALSE, which returns the absolute value of the difference. For example, if [date1] is larger than [date2], then the result will be negative if returnSignedValue is set to TRUE. If returnSignedValue is not set or is set to FALSE, then the result will ALWAYS be a positive number. If returnSignedValue is set to FALSE or not set, then the order of the dates in the equation does not matter because the resulting value will always be positive (although the + sign is not displayed but implied).
isblankormissingcode (value)	**Is a field's value blank/null or is it a Missing Data Code?**	Returns a boolean (true or false) if the field value is blank/null/'' or if the value is a Missing Data Code, in which Missing Data Codes have been explicitly defined in the project on the Project Setup page under Additional Customizations. E.g. isblankormissingcode([age]), in which if 'age' has a value of 'UNK' (which might be a Missing Data Code in a project), then it will return TRUE. And if the field has any non-blank/non-null value that is also not a Missing Data Code, it will return FALSE.
Functions to use with Numbers		
round (number, decimal places)	**Round**	If the "decimal places" parameter is not provided, it defaults to 0. E.g. To round 14.384 to one decimal place: round(14.384,1) will yield 14.4
roundup (number,decimal places)	**Round Up**	If the "decimal places" parameter is not provided, it defaults to 0. E.g. To round up 14.384 to one decimal place: roundup(14.384,1) will yield 14.4
rounddown (number,decimal places)	**Round Down**	If the "decimal places" parameter is not provided, it defaults to 0. E.g. To round down 14.384 to one decimal place: rounddown(14.384,1) will yield 14.3
sqrt (number)	**Square Root**	E.g. sqrt([height]) or sqrt(([value1]*34)/98.3)
(number)^(exponent)	**Exponents**	Use caret ^ character and place both the number and its exponent inside parentheses. NOTE: The surrounding parentheses are VERY important, as it wil not function correctly without them. For example, (4)^(3) or ([weight]+43)^(2)
abs (number)	**Absolute Value**	Returns the absolute value (i.e. the magnitude of a real number without regard to its sign). E.g. abs(-7.1) will return 7.1 and abs(45) will return 45.

min (number,number,...)	**Minimum**	Returns the minimum value of a set of values in the format min([num1],[num2],[num3],...). NOTE: All blank values will be ignored and thus will only return the lowest numerical value. There is no limit to the amount of numbers used in this function.
max (number,number,...)	**Maximum**	Returns the maximum value of a set of values in the format max([num1],[num2],[num3],...). NOTE: All blank values will be ignored and thus will only return the highest numerical value. There is no limit to the amount of numbers used in this function.
mean (number,number,...)	**Mean**	Returns the mean (i.e. average) value of a set of values in the format mean([num1],[num2],[num3],...). NOTE: All blank values will be ignored and thus will only return the mean value computed from all numerical, non-blank values. There is no limit to the amount of numbers used in this function.
median (number,number,...)	**Median**	Returns the median value of a set of values in the format median([num1],[num2],[num3],...). NOTE: All blank values will be ignored and thus will only return the median value computed from all numerical, non-blank values. There is no limit to the amount of numbers used in this function.
sum (number,number,...)	**Sum**	Returns the sum total of a set of values in the format sum([num1],[num2],[num3],...). NOTE: All blank values will be ignored and thus will only return the sum total computed from all numerical, non-blank values. There is no limit to the amount of numbers used in this function.
stdev (number,number,...)	**Standard Deviation**	Returns the standard deviation of a set of values in the format stdev([num1],[num2],[num3],...). NOTE: All blank values will be ignored and thus will only return the standard deviation computed from all numerical, non-blank values. There is no limit to the amount of numbers used in this function.
log (number, base)	**Logarithm**	Returns the logarithm of the number provided for a specified base (e.g. base 10, base "e"). If base is not provided or is not numeric, it defaults to base "e" (natural log).
isnumber (value)	**Is value a number?**	Returns a boolean (true or false) for if the value is an integer OR floating point decimal number.
isinteger (value)	**Is value an integer?**	Returns a boolean (true or false) for if the value is an integer (whole number without decimals).
Functions to use with Text values		
contains (haystack, needle)	**Does text CONTAIN another text string?**	Returns a boolean (true or false) for if "needle" exists inside (is a substring of) the text string "haystack". Is case insensitive. E.g. contains("Rob Taylor", "TAYLOR") will return as TRUE and contains("Rob Taylor", "paul") returns FALSE. NOTE: This function will *not* work for calculated fields but *will* work in all other places (Data Quality, report filters, Survey Queue, etc.).
not_contain (haystack, needle)	**Does text NOT CONTAIN another text string?**	The opposite of contains(). Returns a boolean (true or false) for if "needle" DOES NOT exist inside (is a substring of) the text string "haystack". Is case insensitive. E.g. not_contain("Rob Taylor", "TAYLOR") will return as FALSE and not_contain("Rob Taylor", "paul") returns TRUE. NOTE: This function will *not* work for calculated fields but *will* work in all other places (Data Quality, report filters, Survey Queue, etc.).
starts_with (haystack, needle)	**Does text START WITH another text string?**	Returns a boolean (true or false) if the text string "haystack" begins with the text string "needle". Is case insensitive. E.g. starts_with("Rob Taylor", "rob") will return as TRUE and starts_with("Rob Taylor", "Tay") returns FALSE. NOTE: This function will *not* work for calculated fields but *will* work in all other places (Data Quality, report filters, Survey Queue, etc.).
ends_with (haystack, needle)	**Does text END WITH another text string?**	Returns a boolean (true or false) if the text string "haystack" ends with the text string "needle". Is case insensitive. E.g. ends_with("Rob Taylor", "Lor") will return as TRUE and ends_with("Rob Taylor", "Tay") returns FALSE. NOTE: This function will *not* work for calculated fields but *will* work in all other places (Data Quality, report filters, Survey Queue, etc.).
left (text, number of characters)	**Returns the leftmost characters**	Returns the leftmost characters from a text value. For example, left([last_name], 3) would return 'Tay' if the value of [last_name] is 'Taylor'.
right (text, number of characters)	**Returns the rightmost characters**	Returns the rightmost characters from a text value. For example, right([last_name], 4) would return 'ylor' if the value of [last_name] is 'Taylor'.
length (text)	**Returns the number of characters**	Returns the number of characters in a text string. For example, length([last_name]) would return '6' if the value of [last_name] is 'Taylor'.
find (needle, haystack)	**Finds one text value within another**	Finds one text value within another. Is case insensitive. The "needle" may be one or more characters long. For example, find('y', [last_name']) would return '3' if the value of [last_name] is 'Taylor'. The value '0' will be returned if "needle" is not found within "haystack".
mid (text, start position, number of characters)	**Returns characters from a text string starting at a position**	Returns a specific number of characters from a text string starting at the position you specify. The second parameter denotes the starting position, in which the beginning of the text value would be '1'. The third parameter represents how many characters to return. For example, mid([last_name], 2, 3) would return 'AYL' if the value of [last_name] is 'TAYLOR'.
concat (text,text,...)	**Combines the text from multiple text strings**	Combines/concatenates the text from multiple text strings into a single text value. For example, concat([first_name], ' ', [last_name]) would return something like 'Rob Taylor'. Each item inside the function must be separated by commas. Each item might be static text (wrapped in single quotes or double quotes), a field variable, or a Smart Variable.
upper (text)	**Converts text to uppercase**	Converts text to uppercase. For example, upper('John Doe') will return 'JOHN DOE'.
lower (text)	**Converts text to lowercase**	Converts text to lowercase. For example, lower('John Doe') will return 'john doe'.
trim (text)	**Removes spaces from the beginning and end of text**	Removes any spaces from both the beginning and end of a text value. For example, trim(' Sentence with spaces on end. ') will return 'Sentence with spaces on end.'.