“How can I use the REDUCE function in Google Sheets to combine data from multiple columns into a single cell?”

Authored by

stats writer

July 1, 2024

The REDUCE function in Google Sheets allows users to easily combine data from multiple columns into a single cell. It simplifies the process of consolidating and organizing data, making it more efficient and convenient. By using this function, users can quickly and accurately create a comprehensive overview of their data, saving time and effort. This feature is particularly useful for managing large datasets or creating reports, as it eliminates the need for manual data entry and reduces the risk of errors. Overall, the REDUCE function in Google Sheets is a powerful tool for streamlining data management tasks and improving productivity.

REDUCE function

This function reduces an array to an accumulated result by application of a `LAMBDA` function to each value.

Sample Usage

```
REDUCE(5, A1:A3, LAMBDA(accumulator, current_value, accumulator+current_value))
```

```
REDUCE(2, A1:A3, LAMBDA(accumulator, current_value, accumulator*current_value))
```

Syntax

```
REDUCE(initial_value, array_or_range, LAMBDA)
```

`initial_value`: The initial `accumulator` value.`array_or_range`: An array or range to be reduced.`LAMBDA`: A `LAMBDA` that's applied to each value in `array_or_range` to reduce it.**Syntax:**`LAMBDA(name1, name2, formula_expression)`**Requirements:**The `LAMBDA` must have exactly 2 `name` arguments along with a `formula_expression` which uses those `names`. The `name1` resolves to the current value in the `accumulator` and `name2` resolves to the `current_value` in `array_or_range`, when applying the `LAMBDA`. The `accumulator` is updated in each step to the intermediate value obtained in the previous step. Go to simple multiplication operation example.

Notes

The passed `LAMBDA` should accept exactly 2 `name` arguments, otherwise an `#N/A` error is returned. These arguments correspond to `accumulator` and `current_value`, in order. These are explained as:

`name1`: Resolves to the current value in the `accumulator`.`name2`: Resolves to the `current_value` in the input array.The `accumulator` is initialized by `initial_value` and updated in each step to the intermediate value obtained in the previous step.The `current_value` in the input array are

found row by row, while the `LAMBDA` is being applied.A `named function` can be passed for the `LAMBDA` parameter and behaves like a `LAMBDA` in this case. Learn more about named functions.

The `named function` must follow the `LAMBDA` syntax for `REDUCE` with exactly 2 argument placeholders defined for it.

Parenthesis shouldn't follow the `named function.`

Examples

Simple multiplication operation

Return the product of all elements in A1:A3 and `initial_value`.

Example data:

	A
1	3
2	2
3	4

Example:`=REDUCE(5, A1:A3, LAMBDA(accumulator, current_value, accumulator*current_value))`

How it works:

Initially, `accumulator` = 5

1. Processing cell A1:

```
accumulator= 5
current_value= 3
```

After processing cell A1:

`accumulator`= (return value of `LAMBDA`)= 5*3= 15

2. Processing cell A2:

```
accumulator= 15
current_value= 2
```

After processing cell A2:

`accumulator`= (return value of `LAMBDA`)= 15*2= 30

3. Processing cell A3:

`accumulator`= 30
`current_value`= 4

After processing cell A3:

`accumulator`= (return value of `LAMBDA`)= 30*4= 120

Result:

120

Sum if price is greater than or equal to $20

Add all the prices that are greater than or equal to $20.

Example Data:

	A
1	$50
2	$10
3	$30
4	$20

Example:`=REDUCE(0, A1:A4, LAMBDA(accumulator, price, if(price>=20, accumulator + price, accumulator)))`

Result:

$100

Use a named function as LAMBDA function

Return the end price after increasing it by a certain percentage every period.

Make a Copy

Example Data:

	A	B	C
1	2022	10%	Starting Price:
2	2023	5%	$100
3	2024	5%	
4	2025	10%	

Example: `=REDUCE(C2,B1:B4,PRICE_INCREASE)`

Named function:`PRICE_INCREASE` is a `named function` which outputs the result after increasing by the percentage value in column B.

Formula definition:`=accumulator+accumulator*cell` where `accumulator` and `cell` are argument placeholders defined for `PRICE_INCREASE`.

Result:

133.4

Use a named function as LAMBDA with 2-dimensional dataset

Find the list of unique employees of the quarter, preserving row-wise order.

Make a Copy

Example data:

	A	B	C		
1		**Q1**	**Q2**	**Q3**	**Q4**
2	**2020**	John	Adam	Stacy	Adam
3	**2021**	Peter	Maurice	John	Kimberly
4	**2022**	Stacy	Michael	Peter	Adam

Named function:`ADD_IF_NOT_PRESENT` is a `named function` which adds a given string value to an array of values.

Function definition: `=IF(CONTAINS(new_value, existing_values), existing_values, {existing_values, new_value})`, where `existing_values` and `new_value` are argument placeholders defined for `ADD_IF_NOT_PRESENT` in that order, and `CONTAINS` is another `named function`.

Example: `=REDUCE({B2}, B2:E4, ADD_IF_NOT_PRESENT)`

Result:

John	Adam	Stacy	Peter	Maurice	Kimberly	Michael

Common Errors

The passed LAMBDA doesn't have exactly 2 name arguments

If the `LAMBDA` function doesn't have 2 `name` arguments, this error occurs:

"Wrong number of arguments to LAMBDA. Expected 3 arguments, but got 2 arguments."

Example: `=REDUCE(5, C1:C4, LAMBDA(current_value, current_value+1))`

In this example, `LAMBDA` was given only 1 `name` argument when it needed 2.

The last parameter of REDUCE wasn't a LAMBDA

If the last parameter of `REDUCE` function wasn't a `LAMBDA` function, this error occurs:

"Argument must be a LAMBDA."

Example: `=REDUCE(5, C1:C4, 3)`

In this example the last function is `3`, instead of a `LAMBDA` function.

The LAMBDA passed to REDUCE was incorrect

If 1 or more `name` arguments aren't valid, this error occurs:

"Argument 1 of function LAMBDA is not a valid name."

Example: `=REDUCE(5, C1:C4, LAMBDA(C1, v, C1+v))`

In this example, `C1` is an invalid `name` since it clashes with a range.

Related functions

LAMBDA function: This function lets create and return a custom function with a set of `names` and a `formula_expression` that uses them.MAP function: This function maps each value in the given arrays to a new value.BYROW function: This function groups an array by rows.BYCOL function: This function groups an array by columns.SCAN function: This function scans an array and produces intermediate values.MAKEARRAY function: This function creates a calculated array of specified dimensions.Create & use named functions: This function lets you create and store custom functions, similar to `LAMBDA`.