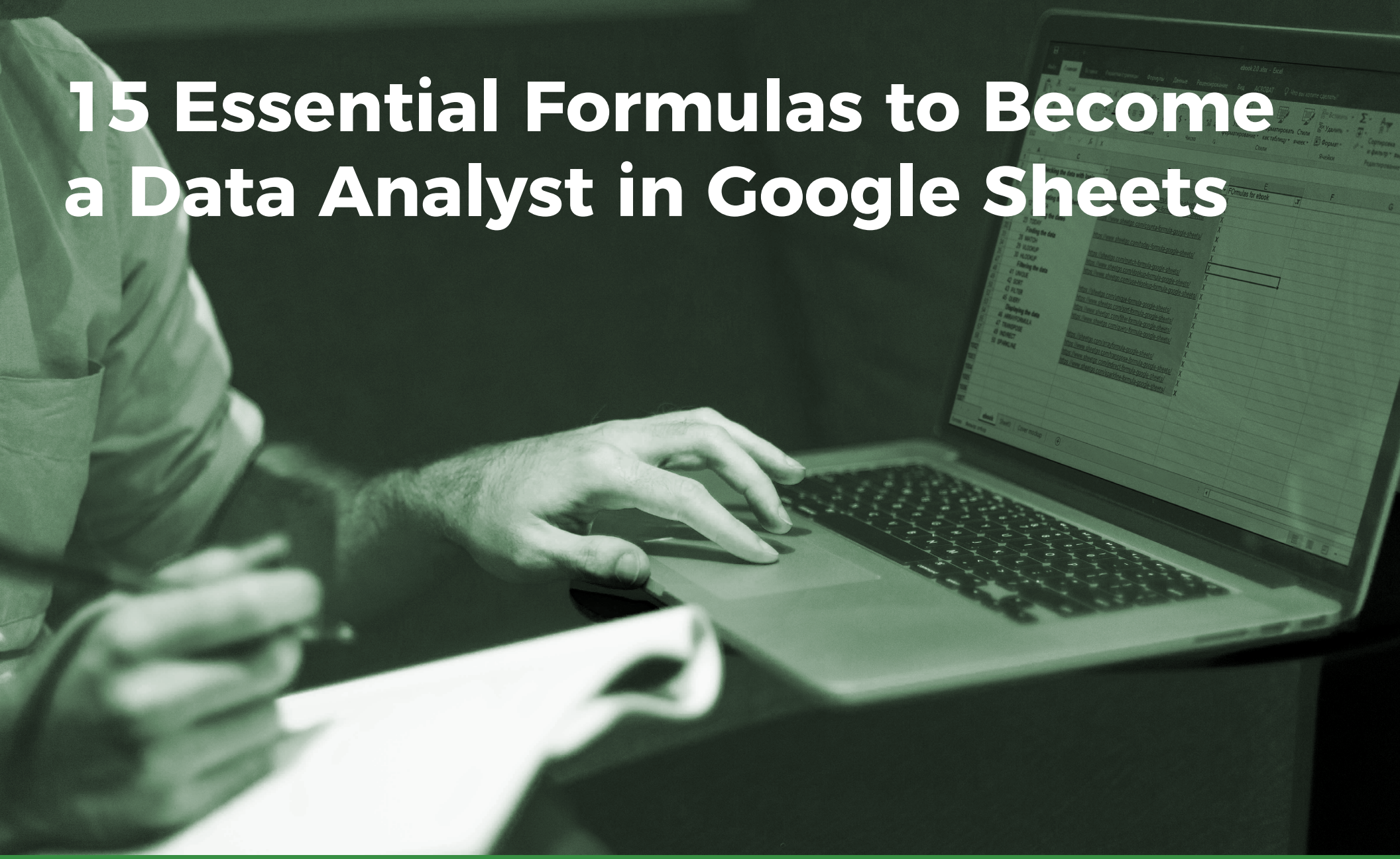
15 Essential Formulas to Become a Data Analyst in Google Sheets



SHEETGO

15 Essential Formulas to Become a Data Analyst in Google Sheets

Spreadsheets are by far the best software tools to analyze data. With spreadsheets you can dispose data on several two-dimensional tables that create a relationship between them, and finally, obtain useful information from the data. Today, Google Sheets is the most widely used online collaborative spreadsheet software. Thus this short ebook focuses on 15 formulas that help you become a data analyst in Google Sheets.

Sheetgo has prepared this ebook, with 15 articles on how to use some of the most commonly used formulas which are essential to becoming a 'data analyst' using Google Sheets. With these formulas, analyzing data should become easier, quicker and more fun to do.

After reading this ebook you should be able to **check your data based on logical statements**, **determine and separate a dataset** from one another, **count** to find how many occurrences you have of a certain subset of your data and **find specific data** within those subsets, **check dates** to split data based on a relative point in time, **filter data** to retrieve only the necessary data from a database and work with a smaller data sample for an improved overview, and **display & visualize** data in various ways based on your different needs.

We hope that you find this ebook useful. If not, please let us know. With your feedback, we can make this ebook even better!

Regards,

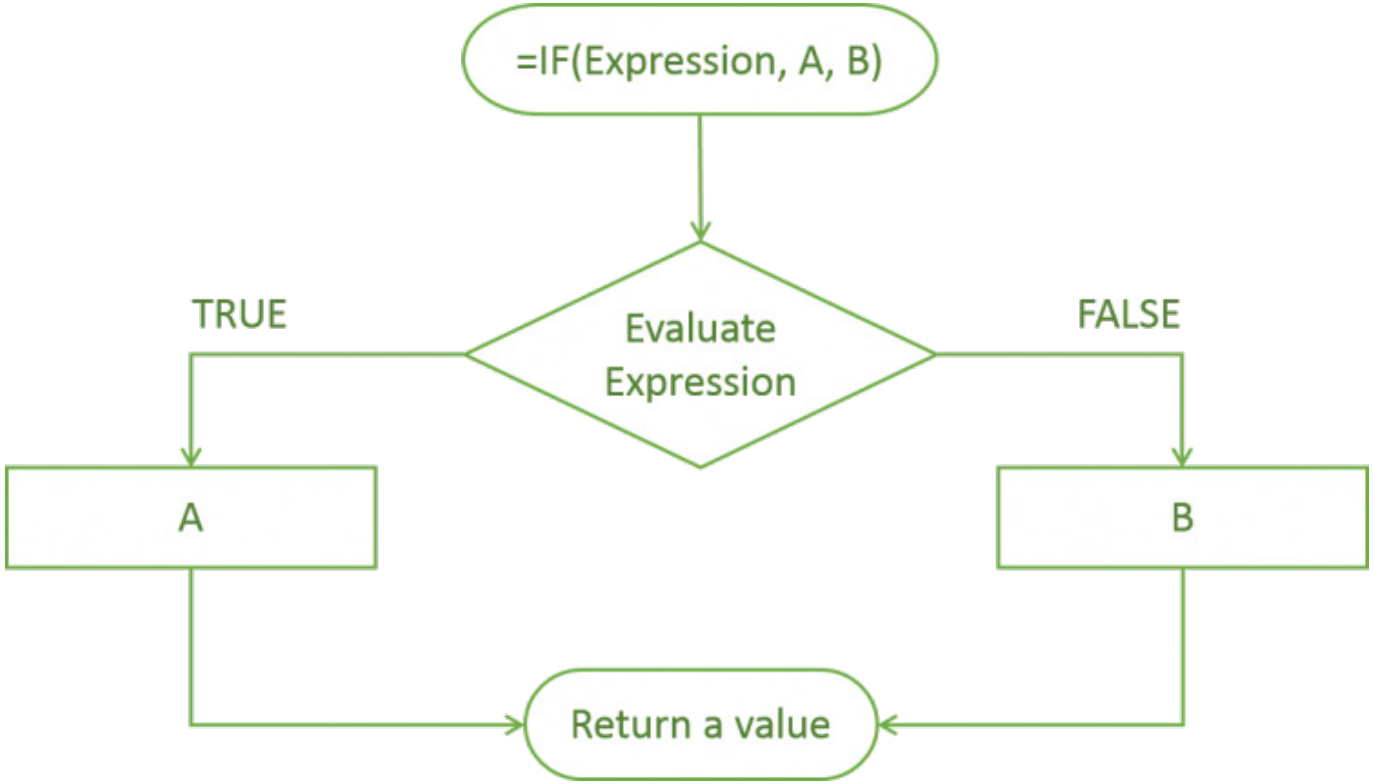
The Sheetgo Team

Table of Contents

Checking the data with logic

IF

Among the myriad formulas that Google Sheets provides, probably the IF formula is one of the most widely used. It returns a value through an if-then-else logical construct. First, it evaluates the given logical expression. Then Based on whether the evaluation is TRUE or FALSE, the formula returns the first value, or the second respectively. The process is illustrated in the flow chart below.



Syntax

IF(logical _ expression, value _ if _ true, value _ if _ false)

- `logical _ expression` – an expression or reference to a cell containing an expression that evaluates to either `TRUE` or `FALSE`.
- `value _ if _ true` – the value that the formula returns if the `logical _ expression` evaluates to `TRUE`. This can be a number, text, or even another formula that returns a value. We can also embed another IF formula in here, as part of the nested IF constructs that we will see in a while.
- `value _ if _ false` – this is the value that the IF formula returns if the `logical _ expression` evaluates to `FALSE`. Similar to the `value _ if _ true`, this can be a number, text or another formula that returns a value. It can also be another embedded IF formula. Please note that it is an optional input and if we ignore this, we get a blank value in return.

Usage: IF Formula

Use case # 1: Regular IF formula constructs

Examples always help us understand the concepts better. So, I used the following sample data (columns A through E). And, I tried a few common variations of the formula (column F). You will notice that we have experimented with the Boolean values, dates, numbers, and also text.

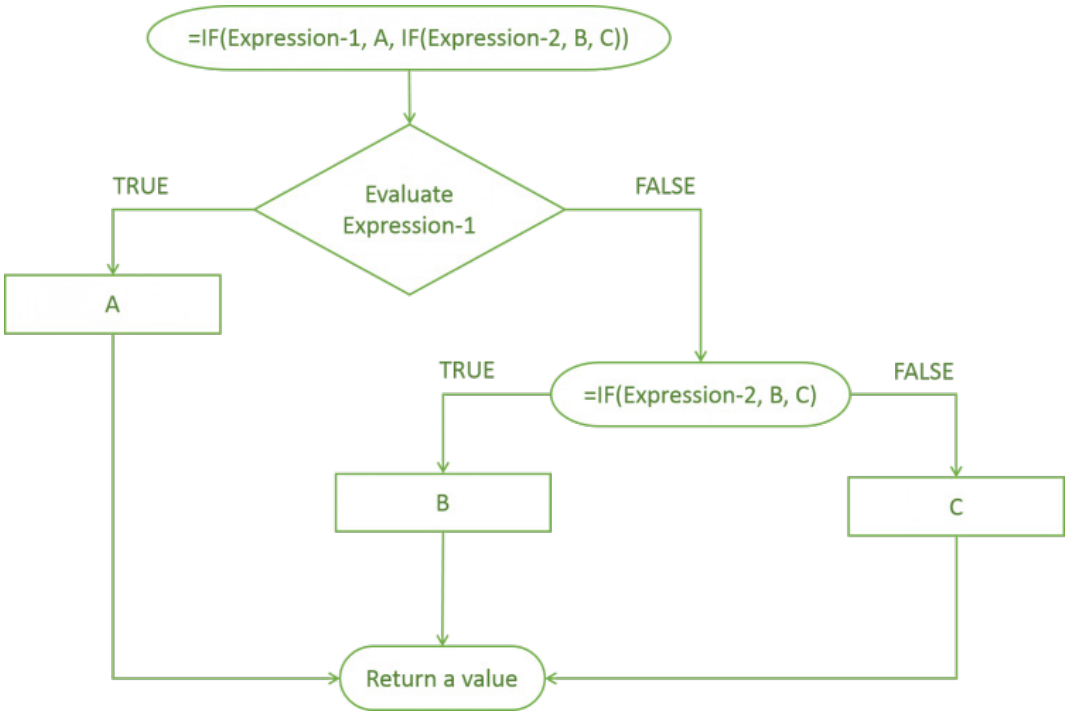
```
=IF(E2="Yes","On Campus","Not On Campus")
```

	A	B	C	D	E	F	G
1	**Name**	**Age**	**Department**	**Join Date**	**On Campus**	**Value Returned**	**Formula Used**
2	Alfred	28	Arts	23-Dec-2016	Yes	On Campus	=IF(E2="Yes","On Campus","Not On Campus")
3	Angela	24	Science	16-Jan-2017	No	Not On Campus	=IF(E3="Yes","On Campus","Not On Campus")
4	Bob	28	Finance	23-Jan-2017	No	Aged 28	=If(B4=28,"Aged 28","Not aged 28")
5	Brett	27	Law	23-Dec-2016	Yes	Not aged 28	=If(B5=28,"Aged 28","Not aged 28")
6	Caroline	23	Engineering	29-Dec-2016	No	After 27th	=IF(D6<DATE(2016,12,27),"Before 27th","After 27th")
7	Charlie	25	Arts	25-Dec-2016	No	Before 27th	=IF(D7<DATE(2016,12,27),"Before 27th","After 27th")
8	David	26	Law	21-Dec-2016	No	<= 5 characters	=IF(LEN(A8)>5,"> 5 characters","<= 5 characters")
9	Dennis	26	Engineering	15-Dec-2016	Yes	> 5 characters	=IF(LEN(A9)>5,"> 5 characters","<= 5 characters")
10	Eric	22	Medicine	03-Jan-2017	Yes	Same Department	=IF(C10=C11,"Same Department","Different Department")
11	Evan	25	Medicine	21-Dec-2016	No	Different Department	=IF(C11=C12,"Same Department","Different Department")
12	Filip	22	Law	21-Jan-2017	Yes	Filip is younger to Francis	=IF(B12<B13,A12&" is younger to "&A13,A12&" is not younger to "&A13)
13	Francis	27	Finance	09-Jan-2017	Yes	Francis is not younger to George	=IF(B13<B14,A13&" is younger to "&A14,A13&" is not younger to "&A14)
14	George	26	Engineering	20-Jan-2017	No	Value evaluated to TRUE	=IF(TRUE,"Value evaluated to TRUE","Value evaluated to FALSE")
15	Grace	21	Medicine	20-Dec-2016	Yes	Value evaluated to FALSE	=IF(FALSE,"Value evaluated to TRUE","Value evaluated to FALSE")

Usage: IF Formula

Use case # 2: Nested IF formula constructs

This is a scenario where we embed IF formulas within IF formulas. And we come across this more often than not. We demonstrate it diagrammatically below.



As explained in the above flow chart, we nested an IF within the value_if_false. Similarly, if need be, we can also nest an IF within the value_if_true. So, if the Expression-1 evaluates to FALSE, the control moves to evaluate the nested IF function. Accordingly, it returns either B or C based on whether the Expression-2 evaluates to either TRUE or FALSE respectively.

Also the diagram is the case for single level nesting. But, we can also try multi-level nesting, i.e. multiple IF formulas placed in a hierarchical fashion. Consider the examples below to understand this concept better.

```
=IF(AND(B4>20,B4<=24),"Younger",IF(AND(B4>24,B4<=26),"Middle Aged",IF(AND(B4>26,B4<=28),"Elder","Age Not applicable")))
```

	A	B	C	D	E
1	**Name**	**Age**	**Work Exp.**	**Value Returned**	**Formula Used**
2	Alfred	28	13	Less than 2 years	=IF(D2="","No experience",IF(D2<12,"Less than 1 year","Less than 2 years"))
3	Angela	24	11	Less than 1 year	=IF(D3="","No experience",IF(D3<12,"Less than 1 year","Less than 2 years"))
4	Bob	28		Elder	=IF(AND(B4>20,B4<=24),"Younger",IF(AND(B4>24,B4<=26),"Middle Aged",IF(AND(B4>26,B4<=28),"Elder","Age Not applicable")))
5	Brett	21	6	Younger	=IF(AND(B5>20,B5<=24),"Younger",IF(AND(B5>24,B5<=26),"Middle Aged",IF(AND(B5>26,B5<=28),"Elder","Age Not applicable")))
6	Caroline	25		Middle Aged	=IF(AND(B6>20,B6<=24),"Younger",IF(AND(B6>24,B6<=26),"Middle Aged",IF(AND(B6>26,B6<=28),"Elder","Age Not applicable")))

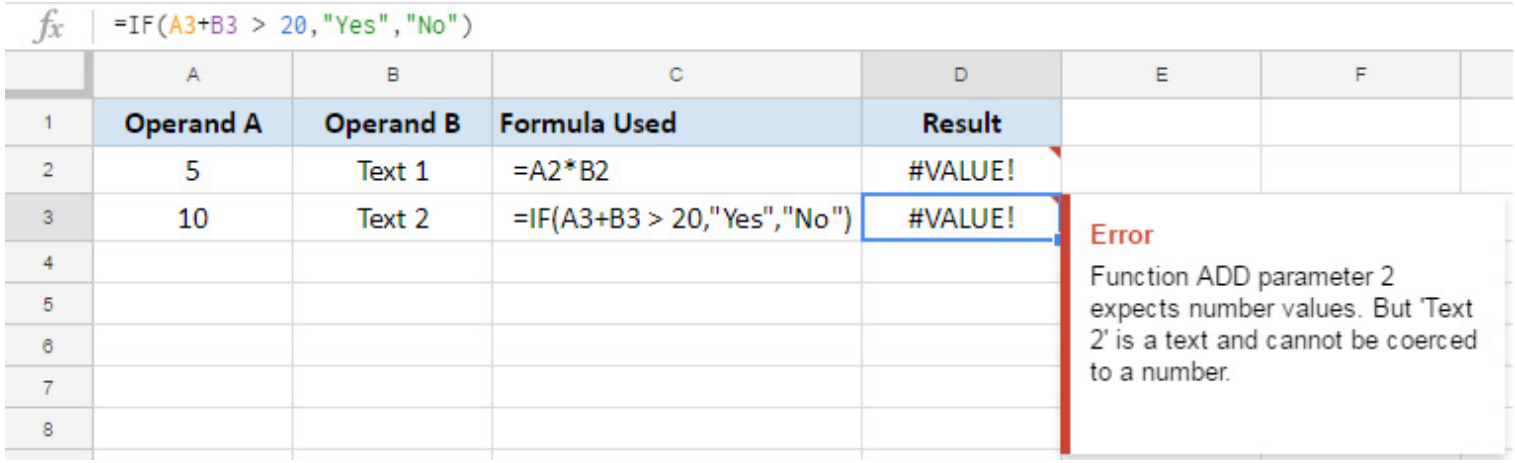
IFERROR

Google Sheets is built in such a way that the expressions return appropriate values in accordance with the given inputs. However, more often than not, we may encounter certain unpleasant errors. This happens due to one or more reasons as explained in this article. To handle such errors, we have the IFERROR formula.

Errors? How do they look like?

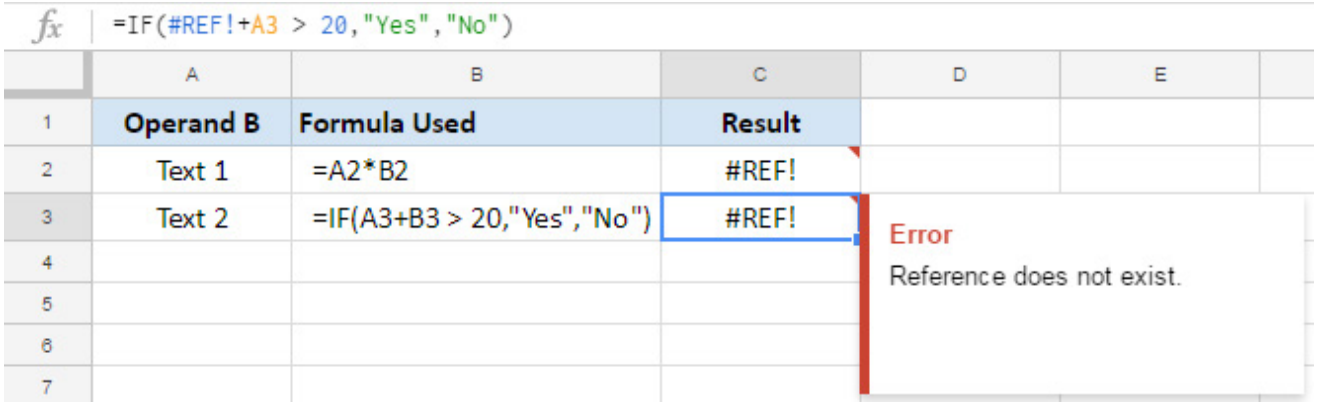
The #VALUE! error

Perhaps, this is the most frequent error type. It occurs when we use an incorrect data type for the expected input arguments as illustrated below.



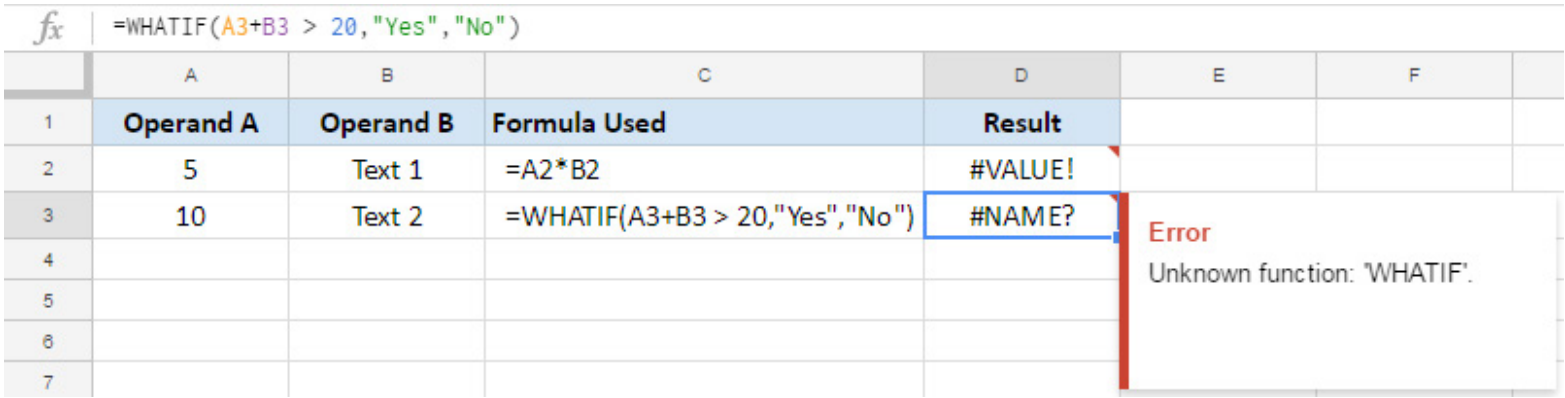
The #REF! error

Usually, this springs up when we intentionally or accidentally delete rows, columns or worksheets, that are referenced in other cells. From the above examples, let's try removing the "Operand A" column, and this is what happens. Notice that even within the formula, the #REF! shows, indicating it lost a reference in that placeholder.



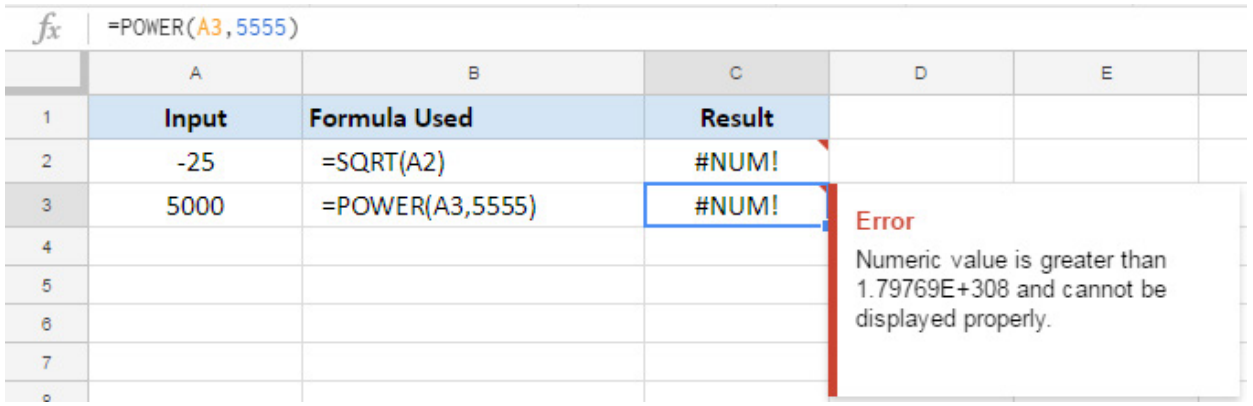
The #NAME! error

Google Sheets throws this up when it sees any function that it doesn't recognize. To understand that further, let's try by using WHATIF, instead of IF in the third row.



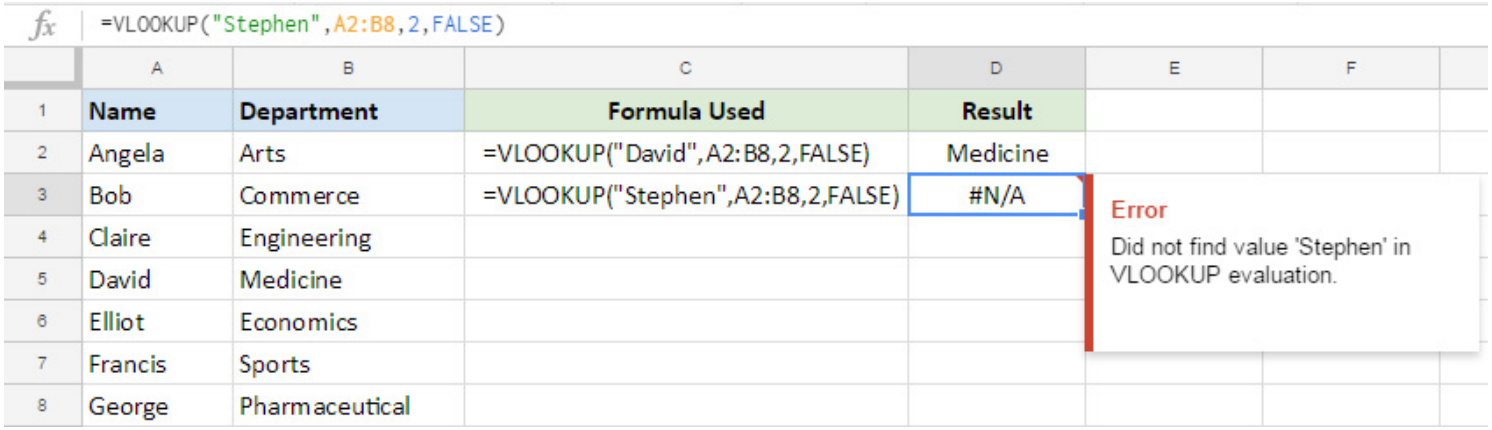
The #NUM! error

When the expected input arguments do not match with what we provide, this might show up. For instance, consider the first example in the snapshot below. A negative number is not expected in the SQRT function, hence the error. This can also surface when the number to display is out of range, as shown in the snapshot below.



The #N/A! error

This is another common error usually associated with the lookup functions when they cannot find a match for the value they are asked to find.



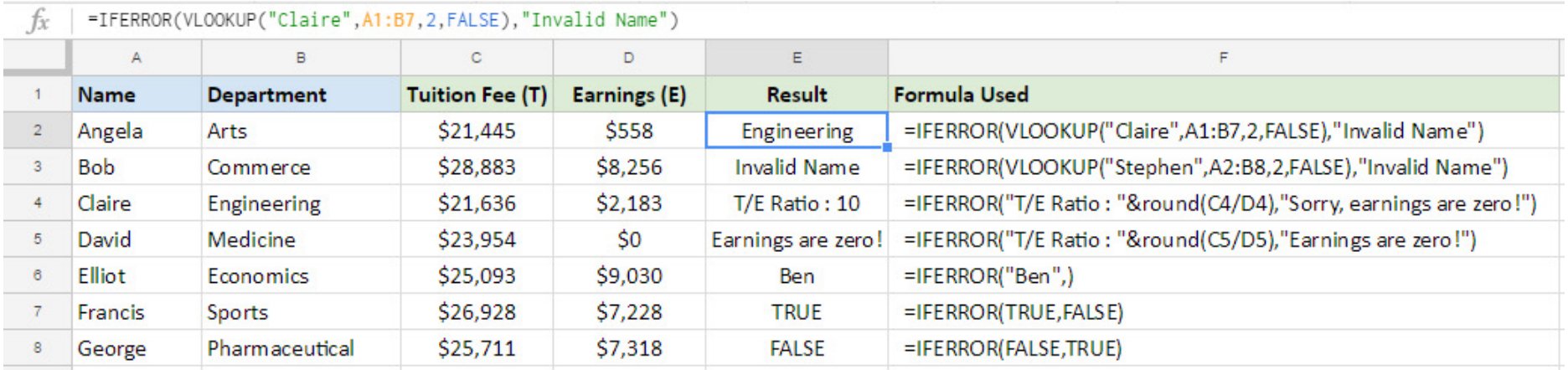
Syntax

`IFERROR(value, [value _ if _ error])`

- `value` – this is the value that the formula returns if `value` itself is not an error.
- `value _ if _ error` – this is the value the function returns if `value` is an error. Since it is an optional parameter, the IFERROR formula returns blank when this argument is not specified

Examples: IFERROR formula

Given below are a few use cases where we put this formula to use.



`fx` `=IFERROR(VLOOKUP("Claire",A1:B7,2,FALSE),"Invalid Name")`

	A	B	C	D	E	F
1	**Name**	**Department**	**Tuition Fee (T)**	**Earnings (E)**	**Result**	**Formula Used**
2	Angela	Arts	$21,445	$558	Engineering	=IFERROR(VLOOKUP("Claire",A1:B7,2,FALSE),"Invalid Name")
3	Bob	Commerce	$28,883	$8,256	Invalid Name	=IFERROR(VLOOKUP("Stephen",A2:B8,2,FALSE),"Invalid Name")
4	Claire	Engineering	$21,636	$2,183	T/E Ratio : 10	=IFERROR("T/E Ratio : "&round(C4/D4),"Sorry, earnings are zero!")
5	David	Medicine	$23,954	$0	Earnings are zero!	=IFERROR("T/E Ratio : "&round(C5/D5),"Earnings are zero!")
6	Elliot	Economics	$25,093	$9,030	Ben	=IFERROR("Ben",)
7	Francis	Sports	$26,928	$7,228	TRUE	=IFERROR(TRUE,FALSE)
8	George	Pharmaceutical	$25,711	$7,318	FALSE	=IFERROR(FALSE,TRUE)

Calculating the numbers

COUNTA

In Google Sheets, like the COUNT formula, the COUNTA formula is yet another simpler and one of the most widely used formulas for day to day spreadsheet needs. It gives us the total number of values within the specified data set.

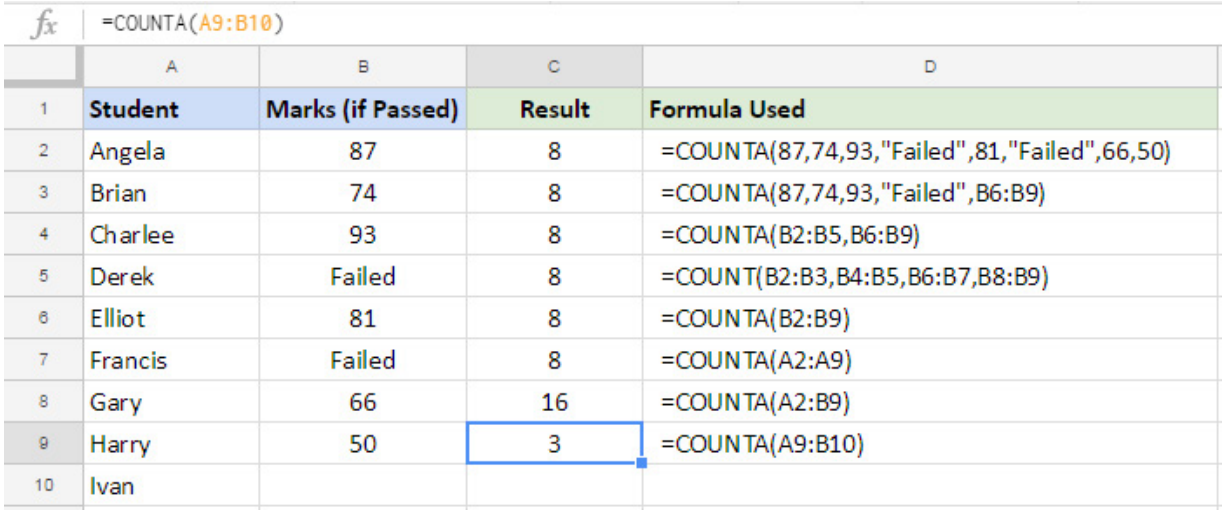
Syntax

`COUNTA(value1, [value2, ...])`

- `value1` – is the value or reference to a range of cell(s) to consider while counting.
- `value2` – and above are optional and additional values or ranges to consider for counting.

Usage: COUNTA Formula

We will try and explain the syntax with a few examples below – as it helps strengthen our understanding.



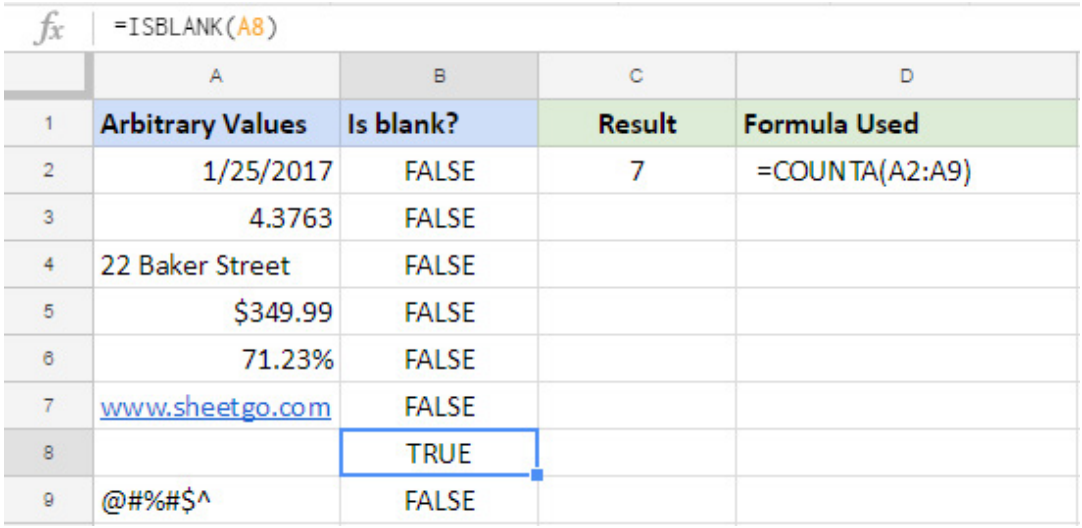
	A	B	C	D
1	**Student**	**Marks (if Passed)**	**Result**	**Formula Used**
2	Angela	87	8	=COUNTA(87,74,93,"Failed",81,"Failed",66,50)
3	Brian	74	8	=COUNTA(87,74,93,"Failed",B6:B9)
4	Charlee	93	8	=COUNTA(B2:B5,B6:B9)
5	Derek	Failed	8	=COUNT(B2:B3,B4:B5,B6:B7,B8:B9)
6	Elliot	81	8	=COUNTA(B2:B9)
7	Francis	Failed	8	=COUNTA(A2:A9)
8	Gary	66	16	=COUNTA(A2:B9)
9	Harry	50	3	=COUNTA(A9:B10)
10	Ivan			

From the list of cases illustrated above, we can deduce that the input parameters can take any of the following forms. It can be a number, text within double quotes, and reference to a range of cells.

No matter what kind of combination we use for the input parameters, the COUNTA formula does its job of counting the values. In all of these cases from row 2 through to 7, the formula returned the result 8, as there are 8 different values in the third column (including the "Failed" ones). Therefore, we can establish that the formula is counting both the numbers and text values.

We included two column ranges in the case on row 8, hence the result showed up as 16. But in the final example, we see the output returned is 3, while the data set consists of values from cells A9, A10, B9 and B10. This difference is because, the COUNTA formula doesn't count blank values!

Let us see what other things the COUNTA formula counts.



	A	B	C	D
1	**Arbitrary Values**	**Is blank?**	**Result**	**Formula Used**
2	1/25/2017	FALSE	7	=COUNTA(A2:A9)
3	4.3763	FALSE		
4	22 Baker Street	FALSE		
5	$349.99	FALSE		
6	71.23%	FALSE		
7	www.sheetgo.com	FALSE		
8		TRUE		
9	@#%#$^	FALSE		

In the above image, we have put in various arbitrary values like a date, currency, percentage, decimal, and general text and even special characters. Accordingly, we have validated for blank values on the second column, using the ISBLANK formula. So, all the values on the rows 2 through to 9 barring row 8 are non blank values. And, not surprisingly, the COUNTA formula on the column C counted only the non blank values.

Checking the dates

TODAY

In Google Sheets, the TODAY formula returns the current date, as per the local date and time settings on the user's computer. For example, if a user from Australia used the TODAY formula at 6:00 AM AEST, and the output it returned is 25-Apr-2017. Then, a different user located in the United States of America using it at the exact same time will see the output as 24-Apr-2017.

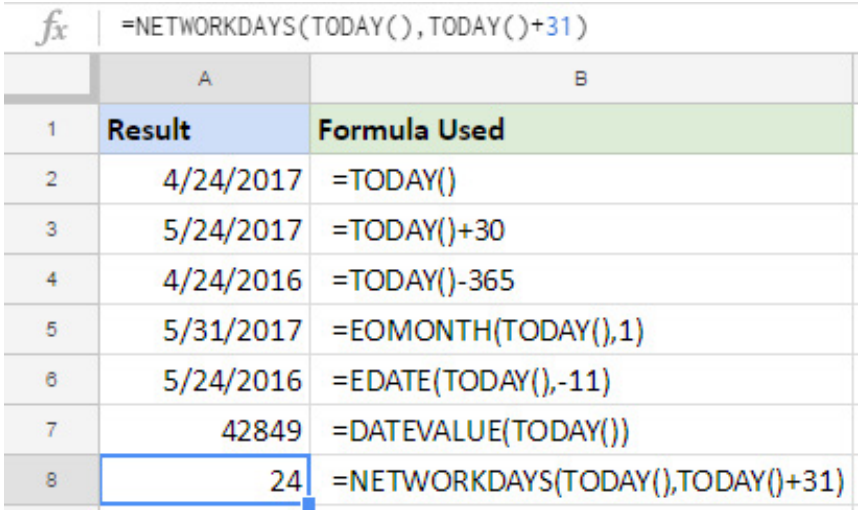
Syntax

`TODAY()`

Note: Unlike many other formulas that Google Sheets comes with, this one doesn't take in any input parameters. Trying to do so will certainly result in an error.

Usage: COUNTA Formula

This is one of the simplest and straightforward formulas available within Google Sheets. We can use this as a standalone formula, or we can me this an input parameter to other existing formulas as shown in the snapshot below.



	A	B
1	**Result**	**Formula Used**
2	4/24/2017	=TODAY()
3	5/24/2017	=TODAY()+30
4	4/24/2016	=TODAY()-365
5	5/31/2017	=EOMONTH(TODAY(),1)
6	5/24/2016	=EDATE(TODAY(),-11)
7	42849	=DATEVALUE(TODAY())
8	24	=NETWORKDAYS(TODAY(),TODAY()+31)

The TODAY formula can get very handy when we use Google Sheets to hold our time critical information. However, it is very important to understand that the formula will always represent the current date (from the last time the spreadsheet calculated formulas). And, it does not remain at the date it displayed when we first used it. So, for example, if the TODAY formula shows 24-Apr-2017 now. And if we open the file one day from now, the formula will show 25-Apr-2017. While that is how we expect it to work, there are two problems with this behavior.

1. This 'running' date may not be a desirable functionality for many users.
    - So, instead of a dynamic date, if we need a static current date to populate within a cell, all we need to do is press the key combination Ctrl + ; (semicolon) [Command + ; (semicolon) for Mac]. Doing so will generate the current date that doesn't change every time the spreadsheet recalculates formulas.
2. Having many TODAY formulas within the spreadsheet may call for performance issues.
    - Google sheets don't calculate most of the other formulas as often as it does the TODAY formula. Since this one is time sensitive Google Sheets continuously ensures we always have current date information.

Finding the data

SHEETGO

MATCH

In Google Sheets, the MATCH formula gives us the relative position of an item in a range of cells. To understand this, please see the snapshot below. The position of **'Evan'** is 5 within the range of cells A1 through A6.

	A
1	Alfred
2	Bob
3	Charlie
4	David
5	Evan
6	Filip

What if we place the cells as shown in the below snapshot (B4 through B9)? The relative position of "Evan" would still remain 5.

	A	B
1		
2		
3		
4		Alfred
5		Bob
6		Charlie
7		David
8		Evan
9		Filip

That is exactly what the MATCH formula is set out to do – return the relative position of an **item** ('**Evan**') in a **range** of cells (**A1:A6** or **B4:B9**).

Syntax

MATCH(search_key, range, search_type)

- `search_key` – is the **item** that the MATCH formula searches within the `range` of cells. It can be a pure text ('`Evan`'), or a cell reference (like `A7`), or even a function that returns a string or a number (like `LEFT`("`Mike Johnson`",8) or `DATE`(2017,1,1))
- `range` – is the group of cells where the MATCH formula searches for the **item** (`search_key`). This must be a one-dimensional array, i.e. either a range with a single column or a single row.
- `search_type` – is an optional input that directs how the MATCH formula should search for the `search_key` in the `range`. This takes in three different values:
  - **1**, is the default value (i.e. when no input is provided against `search_type`). Going with this option, Google Sheets assumes that the range of cells are sorted in ascending order, and accordingly returns the largest value less than or equal to `search_key`.
  - **0**, specifies Google Sheets that it must go for an exact match. This is the ideal option to go with if the range of cells are not sorted in any order.
  - **-1**, as one would guess, is the exact opposite of **1**. This option assumes that the range of cells are sorted in descending order, and returns the smallest value greater than or equal to `search_key`.

Examples

Within column A, I have a test data in ascending order, on which I've tried a few variations of the MATCH formula.

	A	B	C	D
1	**ID (Asc Order)**	**Formula Used**	**Position Returned**	**Reference Cells**
2	1022	=MATCH(1400,A2:A15,1)	7	1727
3	1087	=MATCH(1400,A2:A15,0)	7	
4	1119	=MATCH(1300,A2:A15,0)	#N/A	
5	1125	=MATCH(D2,A2:A15,1)	11	
6	1313	=MATCH(D2,A2:A15,0)	10	
7	1323			
8	1400			
9	1669			
10	1718			
11	1727			
12	1727			
13	1753			
14	1922			
15	1996			

	A	B	C	D
1	ID (Desc Order)	Formula Used	Position Returned	Reference Cells
2	1996	=MATCH(1400,A2:A15,-1)	8	1727
3	1922	=MATCH(1400,A2:A15,0)	8	
4	1753	=MATCH(1300,A2:A15,0)	#N/A	
5	1727	=MATCH(D2,A2:A15,-1)	4	
6	1727	=MATCH(D2,A2:A15,0)	4	
7	1718			
8	1669			
9	1400			
10	1323			
11	1313			
12	1125			
13	1119			
14	1087			
15	1022			

Within column A, I have a test data in ascending order, on which I've tried a few variations of the MATCH formula.

	A	B	C	D
1	Names	Formula Used	Position Returned	Reference Cells
2	Filip	=MATCH("Eric",A2:A14,0)	#N/A	Caroline
3	Eric Sr.	=MATCH("Eric Sr.",A2:A14,0)	2	
4	Brett	=MATCH(D2,A2:A15,0)	6	
5	Francis			
6	Angela			
7	Caroline			
8	Charlie			
9	Alfred			
10	David			
11	Dennis			
12	Bob			
13	Caroline			
14	Evan			
15	Eric Jr.			

That was with the numeric values. How about using the MATCH formula with text values? Since we can't exactly define the 'less than' and 'greater than' values for text forms, we usually go with the `search_type` option **0**, that tells Google Sheets to go for an exact match. Following are a few examples.

Use case: INDEX & MATCH formula combination

Perhaps the most powerful use of MATCH formula in Google Sheets is when we use it along with the INDEX formula, in order to lookup values. But we already have VLOOKUP formula in Google Sheets for this purpose, don't we? We have an example below:

	A	B	C	D
1	**ID**	**Name**	**Formula Used**	**Value Returned**
2	1101	Filip	=VLOOKUP(1432,A2:B15,2,FALSE)	Charlie
3	1129	Eric Sr.		
4	1304	Brett		
5	1304	Francis		
6	1349	Angela		
7	1407	Caroline		
8	1432	Charlie		
9	1515	Alfred		
10	1661	David		
11	1683	Dennis		
12	1773	Bob		
13	1801	Caroline		
14	1913	Evan		
15	1927	Eric Jr.		

The VLOOKUP formula in D2 looks up 1432 in the ID column (as it the left most in the range A2:B15), and from the row where it finds 1432, it fetches the value located in the second column (i.e. column B), while assuming the data is not sorted. So far, so good. But we have two critical problems with using VLOOKUP in Google Sheets.

Problem # 1: Static cell referencing

What happens when we insert a new column between the first and second columns? Let's try that.

	A	B	C	D	E
1	**ID**	**Location**	**Name**	**Formula Used**	**Value Returned**
2	1101	CO	Filip	=VLOOKUP(1432,A2:C15,2,FALSE)	DC
3	1129	AR	Eric Sr.		
4	1304	AK	Brett		
5	1304	IL	Francis		
6	1349	GU	Angela		
7	1407	DE	Caroline		
8	1432	DC	Charlie		
9	1515	CT	Alfred		
10	1661	GA	David		
11	1683	AS	Dennis		
12	1773	FL	Bob		
13	1801	AZ	Caroline		
14	1913	HI	Evan		
15	1927	ID	Eric Jr.		

You'll notice the returned value is not '**Charlie**' anymore. Because, it was a semi-static formula. Google Sheets updated the second parameter to reflect the new range, but it did not accordingly change the column index (third parameter) when we added a new column before the '**Name**' column.

Problem # 2: Lookup column is always the left most

Situations may arise where the we might have to lookup values from a column (ID) that is not the left most, as shown below.

	A	B	C
1	**Name**	**Location**	**ID**
2	Filip	CO	1101
3	Eric Sr.	AR	1129
4	Brett	AK	1304
5	Francis	IL	1304

What if we move the ID column to make it the left most? Good idea. But, that isn't an ideal approach. Because, there can be data layout or presentation specifications that do not allow us to re-arrange columns. What, in such a situation, is the solution? The INDEX and MATCH formula combination comes to our rescue. Here's the combination syntax, followed by a few examples.

INDEX(reference, MATCH(search_key, range, search_type))

	A	B	C	D	E
1	**Name**	**Location**	**ID**	**Formula Used**	**Value Returned**
2	Filip	CO	1101	=INDEX(A2:A15,MATCH(1773,C2:C15,0))	Bob
3	Eric Sr.	AR	1129	=INDEX(B2:B15,MATCH(1683,C2:C15,0))	AS
4	Brett	AK	1304	=INDEX(A2:A15,MATCH("IL",B2:B15,0))	Francis
5	Francis	IL	1304	=INDEX(C2:C15,MATCH("Brett",A2:A15,0))	1304
6	Angela	GU	1349		
7	Caroline	DE	1407		
8	Charlie	DC	1432		
9	Alfred	CT	1515		
10	David	GA	1661		
11	Dennis	AS	1683		
12	Bob	FL	1773		
13	Caroline	AZ	1801		
14	Evan	HI	1913		
15	Eric Jr.	ID	1927		

As we can see, unlike VLOOKUP, the combination works even if the lookup column is not the left most (first three examples). Not surprisingly, it also works like VLOOKUP, when the lookup column is the left most.

Also, let's see whether the formulas still hold good, if we introduce a new column in between.

	A	B	C	D	E	F
1	**Name**	**Location**	**State**	**ID**	**Formula Used**	**Value Returned**
2	Filip	CO	Colorado	1101	=INDEX(A2:A15,MATCH(1773,D2:D15,0))	Bob
3	Eric Sr.	AR	Arkansas	1129	=INDEX(B2:B15,MATCH(1683,D2:D15,0))	AS
4	Brett	AK	Alaska	1304	=INDEX(A2:A15,MATCH("IL",B2:B15,0))	Francis
5	Francis	IL	Illinois	1304	=INDEX(D2:D15,MATCH("Brett",A2:A15,0))	1304
6	Angela	GU	Guam	1349		
7	Caroline	DE	Delaware	1407		
8	Charlie	DC	Dist. of Col	1432		
9	Alfred	CT	Connecticut	1515		
10	David	GA	Georgia	1661		
11	Dennis	AS	American S	1683		
12	Bob	FL	Florida	1773		
13	Caroline	AZ	Arizona	1801		
14	Evan	HI	Hawaii	1913		
15	Eric Jr.	ID	Idaho	1927		

Thankfully, yes, they hold good. As soon as we introduced a new column (State), Google Sheets updated the references automatically to accommodate this change. Therefore, the INDEX & MATCH formula combination is much more flexible and versatile than the already popular and powerful VLOOKUP formula.

VLOOKUP

Given its utility, VLOOKUP formula is perhaps one of the most widely used formulas in Google Sheets. It stands for 'Vertical Lookup'. This searches for a key value in the first column of the input range. And, it returns the value of a specified cell from the row where it finds the key. We can expect an error if the key doesn't exist.

Syntax

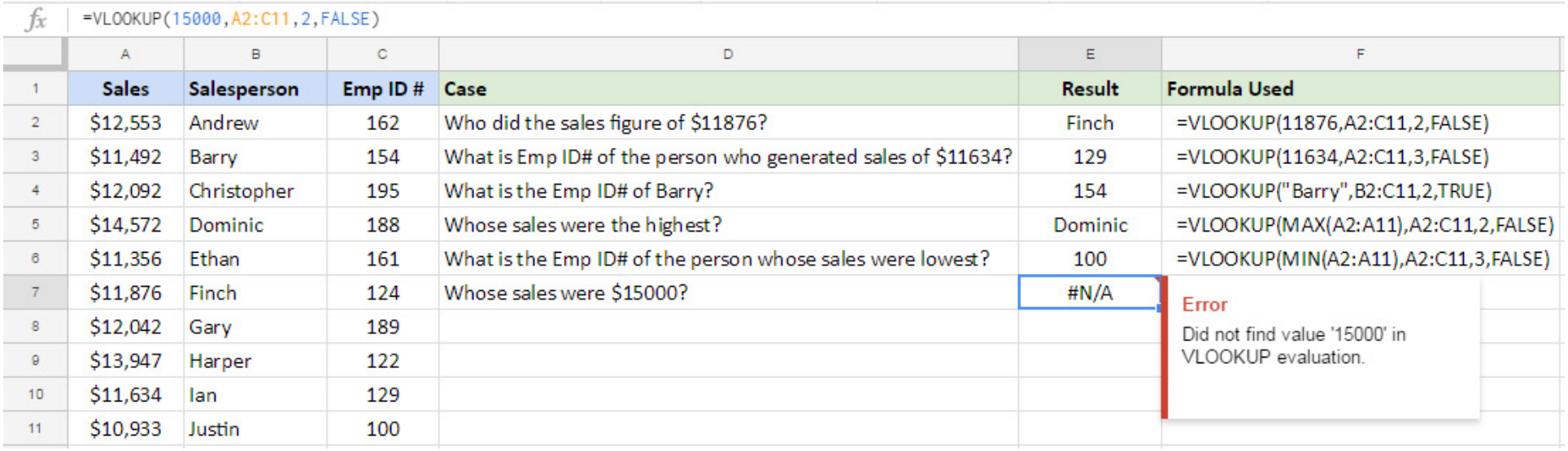
`VLOOKUP(search_key, range, index, [is_sorted])`

- search_key – is the value that the VLOOKUP formula uses to search.
- range – is the reference to the range of cells that we are performing a search on. Google Sheets application looks for the key in the first column of the range.
- index – this is the column index of the cell within the range, whose value the formula returns. The first column's index within the range is 1, the second in the range is 2 and so on. For example, if we input 3 against this parameter, the VLOOKUP formula returns the cell value from the third column and the row in which it finds the search_key.
- [is_sorted] – is an optional parameter which is TRUE by default. This indicates whether the first column in the range is sorted in ascending order or not. If not, we should specify the value as FALSE.

Usage: VLOOKUP Formula

Use case: Numbers in the first column

Let us try our hands on a few examples. Here is a test data with sales figures generated by a set of salespersons. We tried to answer various business case scenarios (column D), using this formula. You'll observe how the `index` parameter value affects the outcome.

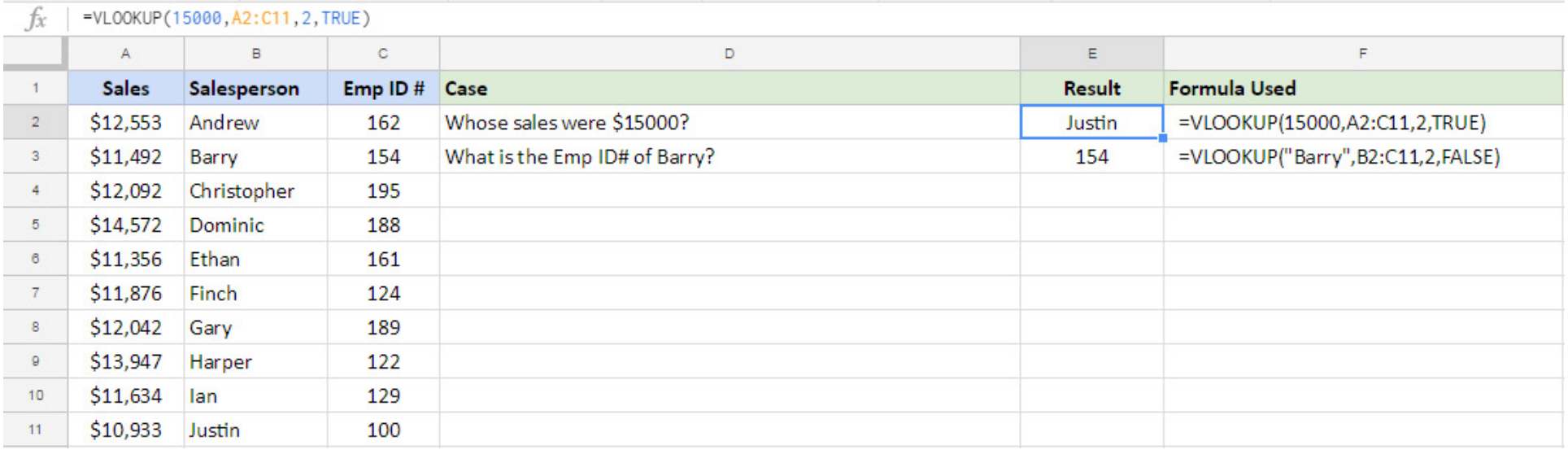


	A	B	C	D	E	F
1	**Sales**	**Salesperson**	**Emp ID #**	**Case**	**Result**	**Formula Used**
2	$12,553	Andrew	162	Who did the sales figure of $11876?	Finch	=VLOOKUP(11876,A2:C11,2,FALSE)
3	$11,492	Barry	154	What is Emp ID# of the person who generated sales of $11634?	129	=VLOOKUP(11634,A2:C11,3,FALSE)
4	$12,092	Christopher	195	What is the Emp ID# of Barry?	154	=VLOOKUP("Barry",B2:C11,2,TRUE)
5	$14,572	Dominic	188	Whose sales were the highest?	Dominic	=VLOOKUP(MAX(A2:A11),A2:C11,2,FALSE)
6	$11,356	Ethan	161	What is the Emp ID# of the person whose sales were lowest?	100	=VLOOKUP(MIN(A2:A11),A2:C11,3,FALSE)
7	$11,876	Finch	124	Whose sales were $15000?	#N/A	
8	$12,042	Gary	189			
9	$13,947	Harper	122			
10	$11,634	Ian	129			
11	$10,933	Justin	100			

Here's an interesting thing to note in the last case (row # 7 above). The formula returned an #N/A! error. Why? That is, as explained in the error description because the $15000 sales figure does not exist in the first column. Therefore, it cannot return an appropriate value.

The curious case of `[is _ sorted]`

In the examples, we have chosen FALSE for the last parameter if the first column within the input range is not sorted in ascending order. We chose TRUE otherwise. Now, what if we tried to do the opposite?



fx =VLOOKUP(15000,A2:C11,2,TRUE)

	A	B	C	D	E	F
1	**Sales**	**Salesperson**	**Emp ID #**	**Case**	**Result**	**Formula Used**
2	$12,553	Andrew	162	Whose sales were $15000?	Justin	=VLOOKUP(15000,A2:C11,2,TRUE)
3	$11,492	Barry	154	What is the Emp ID# of Barry?	154	=VLOOKUP("Barry",B2:C11,2,FALSE)
4	$12,092	Christopher	195			
5	$14,572	Dominic	188			
6	$11,356	Ethan	161			
7	$11,876	Finch	124			
8	$12,042	Gary	189			
9	$13,947	Harper	122			
10	$11,634	Ian	129			
11	$10,933	Justin	100			

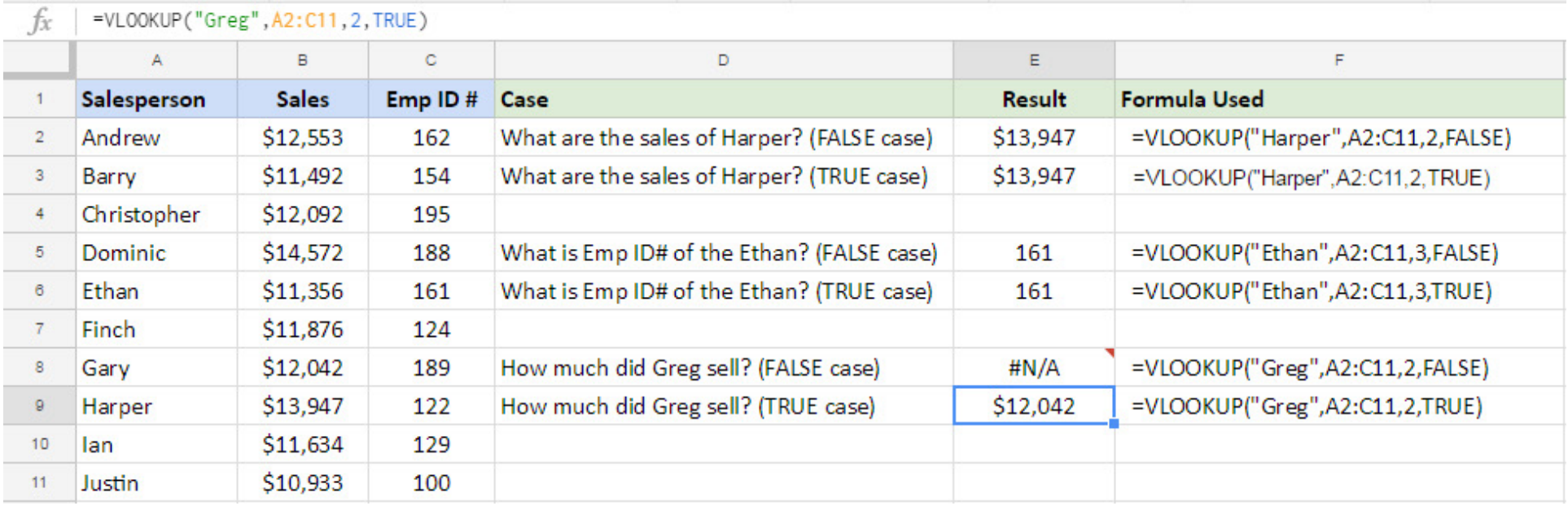
The second case worked out well. But the first one, not as much! Why is that? Here is a note from the official documentation of VLOOKUP formula:

> ***If the is_sorted argument is set to TRUE or omitted, and the first column of the range is not in sorted order, an incorrect value might be returned.***

So, there, they said it themselves! It will not give us expected results when we go with the TRUE option when the first column is not sorted. However, if we have sorted values in the first column, using TRUE for the last parameter will result in a much better performance.

Use case: Sorted strings in the first column

Now that we got our hands greased with numbers, let us try with an example data set, in which the first column has sorted string values, as shown below. There are two examples each for all the three cases available, but with one distinction of `is _ sorted` parameter. Please observe how the VLOOKUP formula is behaving.



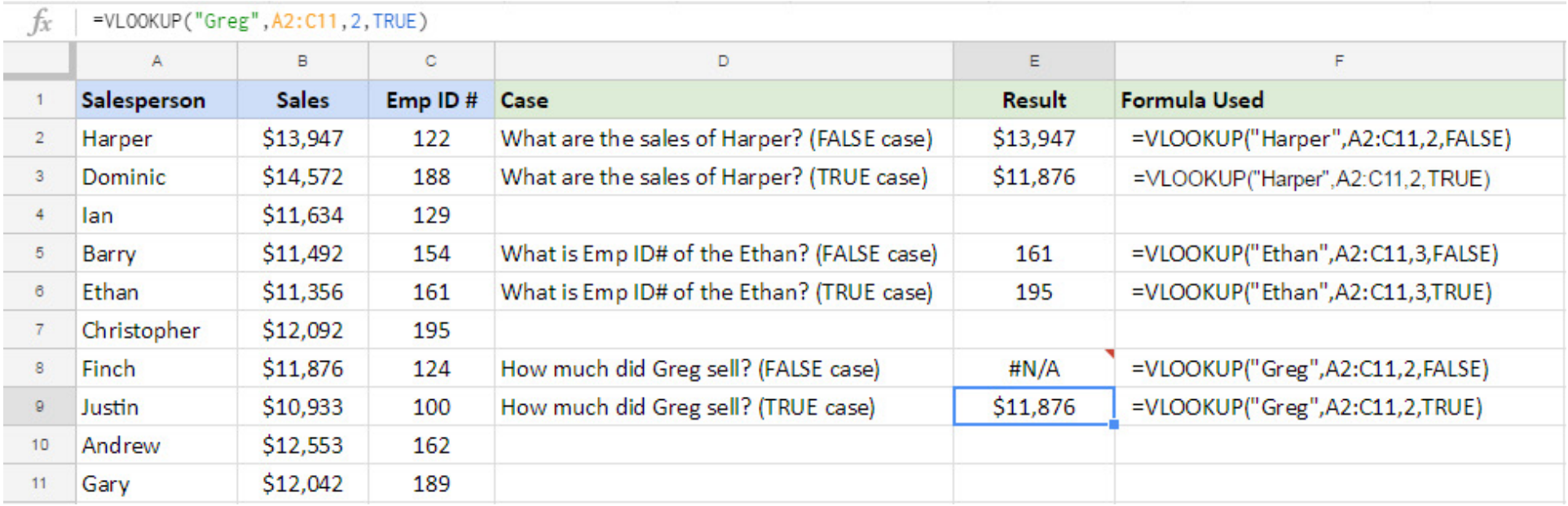
=VLOOKUP("Greg",A2:C11,2,TRUE)

	A	B	C	D	E	F
1	**Salesperson**	**Sales**	**Emp ID #**	**Case**	**Result**	**Formula Used**
2	Andrew	$12,553	162	What are the sales of Harper? (FALSE case)	$13,947	=VLOOKUP("Harper",A2:C11,2,FALSE)
3	Barry	$11,492	154	What are the sales of Harper? (TRUE case)	$13,947	=VLOOKUP("Harper",A2:C11,2,TRUE)
4	Christopher	$12,092	195			
5	Dominic	$14,572	188	What is Emp ID# of the Ethan? (FALSE case)	161	=VLOOKUP("Ethan",A2:C11,3,FALSE)
6	Ethan	$11,356	161	What is Emp ID# of the Ethan? (TRUE case)	161	=VLOOKUP("Ethan",A2:C11,3,TRUE)
7	Finch	$11,876	124			
8	Gary	$12,042	189	How much did Greg sell? (FALSE case)	#N/A	=VLOOKUP("Greg",A2:C11,2,FALSE)
9	Harper	$13,947	122	How much did Greg sell? (TRUE case)	$12,042	=VLOOKUP("Greg",A2:C11,2,TRUE)
10	Ian	$11,634	129			
11	Justin	$10,933	100			

Apparently, the TRUE or FALSE value for `is _ sorted` parameter doesn't really affect the behavior when it finds the `search _ key` within the first column of the range. But an interesting development happens when it doesn't find a match for the `search _ key`. In the case of FALSE (row # 8), it was looking for an exact match. But in the case of TRUE (row # 9), it apparently is looking for a near match. This, obviously, may not be desirable in many cases.

Use case: Unsorted strings in the first column

The data we are considering for this use case is essentially the same as that used above. Except, we have non-sorted strings in the first row.



=VLOOKUP("Greg",A2:C11,2,TRUE)

	A	B	C	D	E	F
1	**Salesperson**	**Sales**	**Emp ID #**	**Case**	**Result**	**Formula Used**
2	Harper	$13,947	122	What are the sales of Harper? (FALSE case)	$13,947	=VLOOKUP("Harper",A2:C11,2,FALSE)
3	Dominic	$14,572	188	What are the sales of Harper? (TRUE case)	$11,876	=VLOOKUP("Harper",A2:C11,2,TRUE)
4	Ian	$11,634	129			
5	Barry	$11,492	154	What is Emp ID# of the Ethan? (FALSE case)	161	=VLOOKUP("Ethan",A2:C11,3,FALSE)
6	Ethan	$11,356	161	What is Emp ID# of the Ethan? (TRUE case)	195	=VLOOKUP("Ethan",A2:C11,3,TRUE)
7	Christopher	$12,092	195			
8	Finch	$11,876	124	How much did Greg sell? (FALSE case)	#N/A	=VLOOKUP("Greg",A2:C11,2,FALSE)
9	Justin	$10,933	100	How much did Greg sell? (TRUE case)	$11,876	=VLOOKUP("Greg",A2:C11,2,TRUE)
10	Andrew	$12,553	162			
11	Gary	$12,042	189			

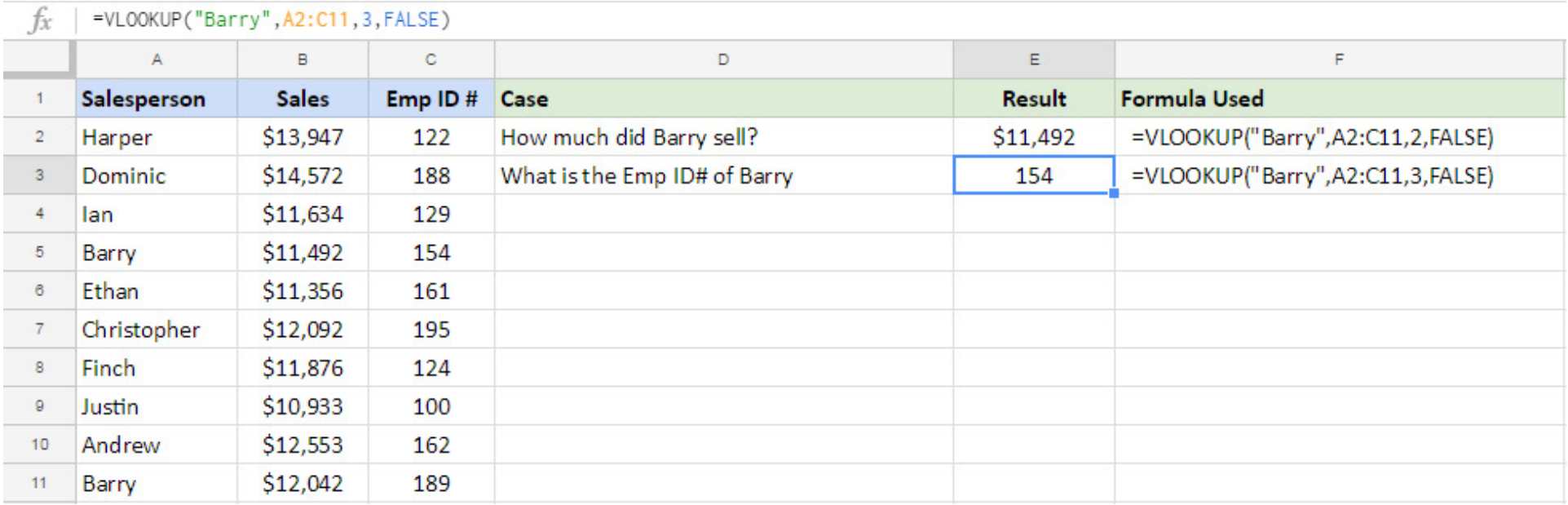
Unlike the previous example, the use of TRUE and FALSE values for the fourth parameter is generating different results. And we can deduce that the results are correct only when we used FALSE for `is_sorted`.

So, what is the bottom line?

> ***Never lie to Google Sheets in the*** `is_sorted` ***parameter! Seemingly, it returns the favor!***

Use case: Multiple matches in the first column

There will be instances where we may encounter multiple instances of the same value in the first column. In the example below, Barry occurred twice. What happens if we use VLOOKUP formula in such scenario? Let's find out.



fx =VLOOKUP("Barry",A2:C11,3,FALSE)

	A	B	C	D	E	F
1	**Salesperson**	**Sales**	**Emp ID #**	**Case**	**Result**	**Formula Used**
2	Harper	$13,947	122	How much did Barry sell?	$11,492	=VLOOKUP("Barry",A2:C11,2,FALSE)
3	Dominic	$14,572	188	What is the Emp ID# of Barry	154	=VLOOKUP("Barry",A2:C11,3,FALSE)
4	Ian	$11,634	129			
5	Barry	$11,492	154			
6	Ethan	$11,356	161			
7	Christopher	$12,092	195			
8	Finch	$11,876	124			
9	Justin	$10,933	100			
10	Andrew	$12,553	162			
11	Barry	$12,042	189			

You may have noticed, it picks up the first Barry it encounters in the list. And the second Barry is not considered.

Use case: Drawbacks with the VLOOKUP formula

There are two problems with this formula:

1. For looking up search_key, it always uses the first column within the input range. So it is not possible with VLOOKUP formula to fetch a cell value that is to the left of the lookup column.
2. The formula is not dynamic enough, in that, the column index values are not updated if we inserted a column between the input range.

There's an alternative that solves the above two problems. Please go through the explanation for the INDEX and MATCH formula combination here.

HLOOKUP

The HLOOKUP formula is perhaps not as famous and widely used as its counterpart, VLOOKUP formula. But, this too serves similar purpose, albeit in a transposed form. It stands for 'Horizontal Lookup'. This searches for a key value in the first row of the input range. And, it returns the value of a specified cell from the column where it finds the key. We can expect an error if the key doesn't exist.

Syntax

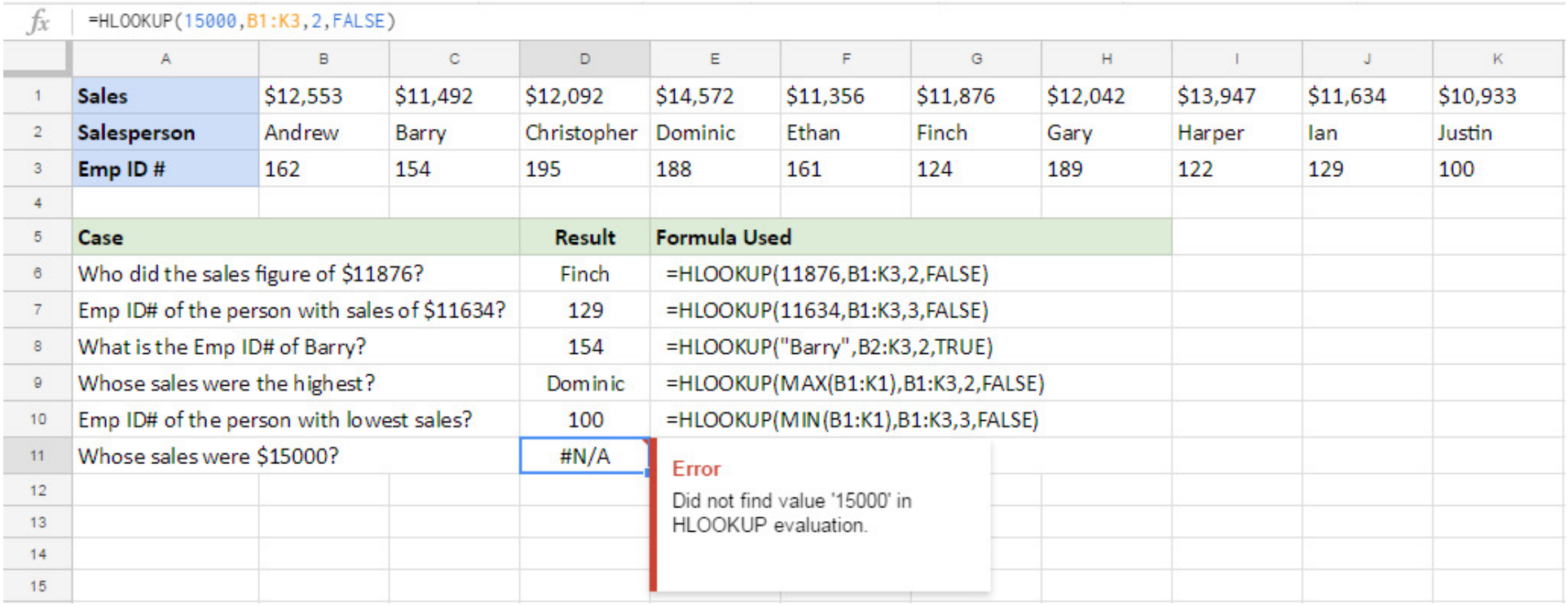
`HLOOKUP(search_key, range, index, [is_sorted])`

- `search_key` – is the value that the HLOOKUP formula uses to search.
- `range` – is the reference to the range of cells that we are performing a search on. Google Sheets application looks for the key in the first row of the range.
- `index` – this is the row index of the cell within the range, whose value the formula returns. The first row's index within the `range` is 1, the second in the `range` is 2 and so on. For example, if we input 3 against this parameter, the HLOOKUP formula returns the cell value from the third row and the column in which it finds the `search_key`.
- `[is_sorted]` – is an optional parameter which is TRUE by default. This indicates whether the first row in the range is sorted in ascending order or not. If not, we should specify the value as FALSE.

Usage: HLOOKUP Formula

Use case: Numbers in the first row

Let us try our hands on a few examples. Here is a test data with sales figures that a set of salespersons generated. We tried to answer various business case scenarios, using this formula. You'll observe the how `index` parameter value affects the outcome.

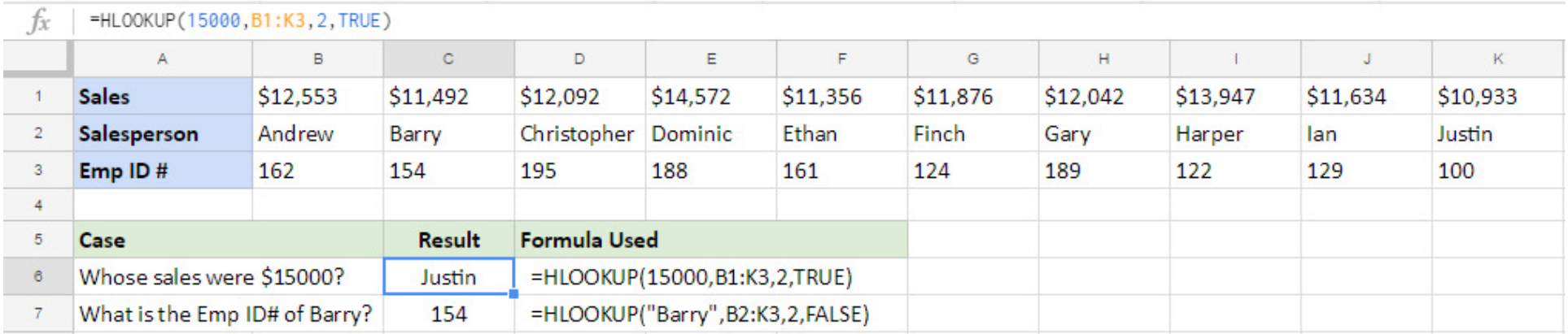


	A	B	C	D	E	F	G	H	I	J	K
1	**Sales**	$12,553	$11,492	$12,092	$14,572	$11,356	$11,876	$12,042	$13,947	$11,634	$10,933
2	**Salesperson**	Andrew	Barry	Christopher	Dominic	Ethan	Finch	Gary	Harper	Ian	Justin
3	**Emp ID #**	162	154	195	188	161	124	189	122	129	100
4											
5	**Case**			**Result**	**Formula Used**						
6	Who did the sales figure of $11876?			Finch	=HLOOKUP(11876,B1:K3,2,FALSE)						
7	Emp ID# of the person with sales of $11634?			129	=HLOOKUP(11634,B1:K3,3,FALSE)						
8	What is the Emp ID# of Barry?			154	=HLOOKUP("Barry",B2:K3,2,TRUE)						
9	Whose sales were the highest?			Dominic	=HLOOKUP(MAX(B1:K1),B1:K3,2,FALSE)						
10	Emp ID# of the person with lowest sales?			100	=HLOOKUP(MIN(B1:K1),B1:K3,3,FALSE)						
11	Whose sales were $15000?			#N/A							
12											
13											
14											
15											

Here's an interesting thing to note in the last case (row # 11 above). The formula returned an #N/A error. Why? That is, as explained in the error description, because the $15000 sales figure does not exist in the first row. Therefore, it cannot return an appropriate value.

The curious case of `[is _ sorted]`

In the examples, we have chosen FALSE for the last parameter if the first row within the input range is not sorted in ascending order. We chose TRUE otherwise. Now, what if we tried to do the opposite?



fx =HLOOKUP(15000,B1:K3,2,TRUE)

	A	B	C	D	E	F	G	H	I	J	K
1	**Sales**	$12,553	$11,492	$12,092	$14,572	$11,356	$11,876	$12,042	$13,947	$11,634	$10,933
2	**Salesperson**	Andrew	Barry	Christopher	Dominic	Ethan	Finch	Gary	Harper	Ian	Justin
3	**Emp ID #**	162	154	195	188	161	124	189	122	129	100
4											
5	**Case**		**Result**	**Formula Used**							
6	Whose sales were $15000?		Justin	=HLOOKUP(15000,B1:K3,2,TRUE)							
7	What is the Emp ID# of Barry?		154	=HLOOKUP("Barry",B2:K3,2,FALSE)							

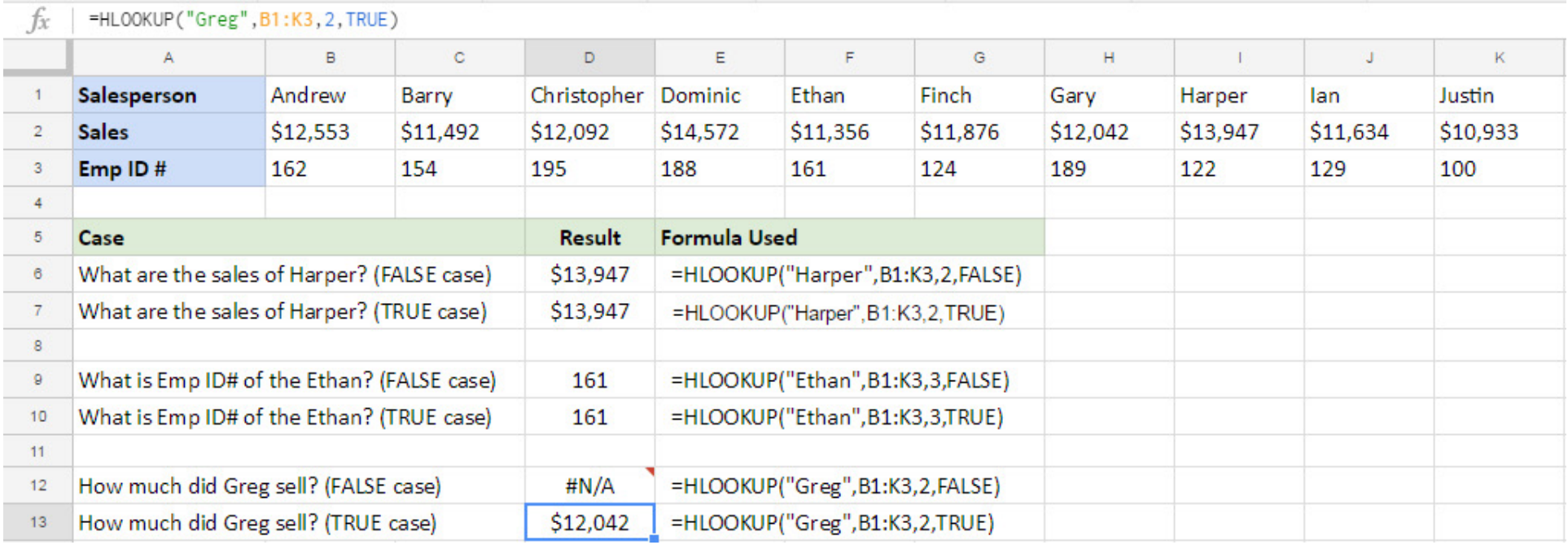
The second case worked out well. But the first one, not as much! Why is that? Here is a note from the official documentation of HLOOKUP formula:

> ***Using an incorrect sort type may cause incorrect values to be returned.***

So, there, they said it themselves! It will not give us expected results when we go with the TRUE option when the first row is not sorted. However, if we have sorted values in the first row, using TRUE for the last parameter will result in a much better performance.

Use case: Sorted strings in the first row

Now that we got our hands greased with numbers, let us try with an example data set, in which the first row has sorted string values, as shown below. There are two examples each for all the three cases available, but with one distinction of `is_sorted` parameter. Please observe how the HLOOKUP formula is behaving.

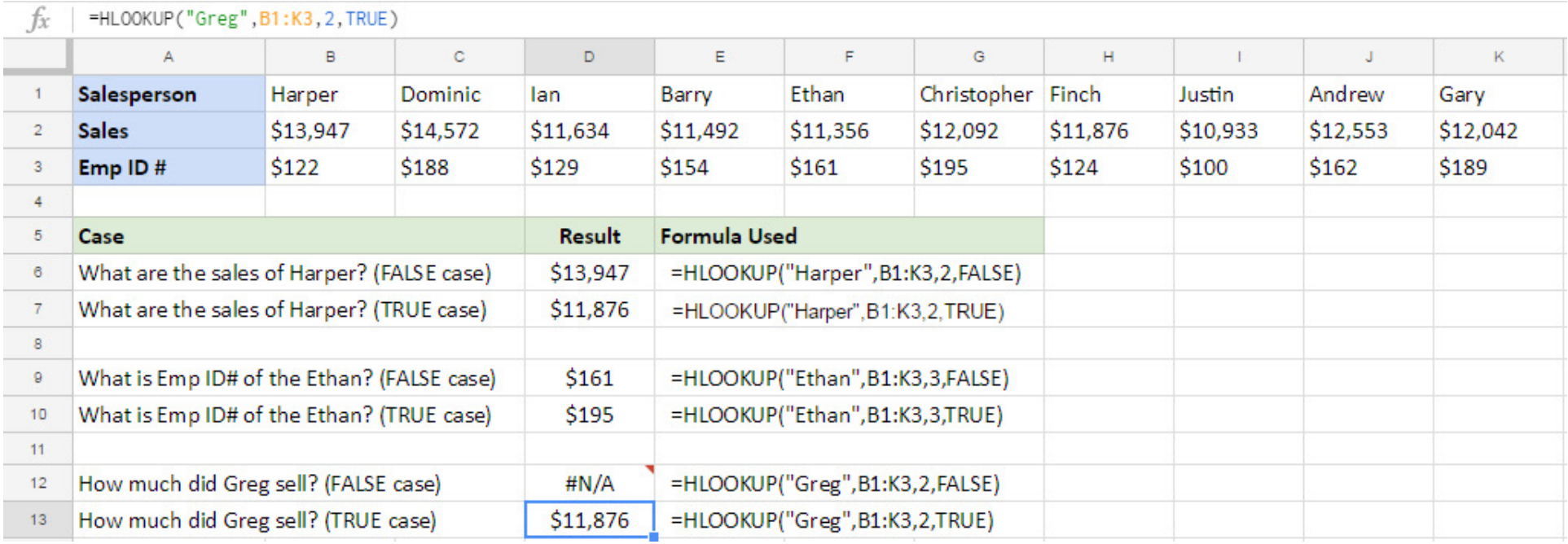


	A	B	C	D	E	F	G	H	I	J	K
1	**Salesperson**	Andrew	Barry	Christopher	Dominic	Ethan	Finch	Gary	Harper	Ian	Justin
2	**Sales**	$12,553	$11,492	$12,092	$14,572	$11,356	$11,876	$12,042	$13,947	$11,634	$10,933
3	**Emp ID #**	162	154	195	188	161	124	189	122	129	100
4											
5	**Case**			**Result**	**Formula Used**						
6	What are the sales of Harper? (FALSE case)			$13,947	=HLOOKUP("Harper",B1:K3,2,FALSE)						
7	What are the sales of Harper? (TRUE case)			$13,947	=HLOOKUP("Harper",B1:K3,2,TRUE)						
8											
9	What is Emp ID# of the Ethan? (FALSE case)			161	=HLOOKUP("Ethan",B1:K3,3,FALSE)						
10	What is Emp ID# of the Ethan? (TRUE case)			161	=HLOOKUP("Ethan",B1:K3,3,TRUE)						
11											
12	How much did Greg sell? (FALSE case)			#N/A	=HLOOKUP("Greg",B1:K3,2,FALSE)						
13	How much did Greg sell? (TRUE case)			$12,042	=HLOOKUP("Greg",B1:K3,2,TRUE)						

Apparently, the TRUE or FALSE value for `is_sorted` parameter doesn't really affect the behavior when it finds the `search_key` within the first row of the range. But an interesting development happens when it doesn't find a match for the `search_key`. In the case of FALSE (row # 12), it was looking for an exact match. But in the case of TRUE (row # 13), it apparently is looking for a near match. This, obviously, may not be desirable in many cases.

Use case: Unsorted strings in the first row

The data we are considering for this use case is essentially the same as that used above. Except, we have non-sorted strings in the first row.



=HLOOKUP("Greg",B1:K3,2,TRUE)

	A	B	C	D	E	F	G	H	I	J	K
1	**Salesperson**	Harper	Dominic	Ian	Barry	Ethan	Christopher	Finch	Justin	Andrew	Gary
2	**Sales**	$13,947	$14,572	$11,634	$11,492	$11,356	$12,092	$11,876	$10,933	$12,553	$12,042
3	**Emp ID #**	$122	$188	$129	$154	$161	$195	$124	$100	$162	$189
4											
5	**Case**			**Result**	**Formula Used**						
6	What are the sales of Harper? (FALSE case)			$13,947	=HLOOKUP("Harper",B1:K3,2,FALSE)						
7	What are the sales of Harper? (TRUE case)			$11,876	=HLOOKUP("Harper",B1:K3,2,TRUE)						
8											
9	What is Emp ID# of the Ethan? (FALSE case)			$161	=HLOOKUP("Ethan",B1:K3,3,FALSE)						
10	What is Emp ID# of the Ethan? (TRUE case)			$195	=HLOOKUP("Ethan",B1:K3,3,TRUE)						
11											
12	How much did Greg sell? (FALSE case)			#N/A	=HLOOKUP("Greg",B1:K3,2,FALSE)						
13	How much did Greg sell? (TRUE case)			$11,876	=HLOOKUP("Greg",B1:K3,2,TRUE)						

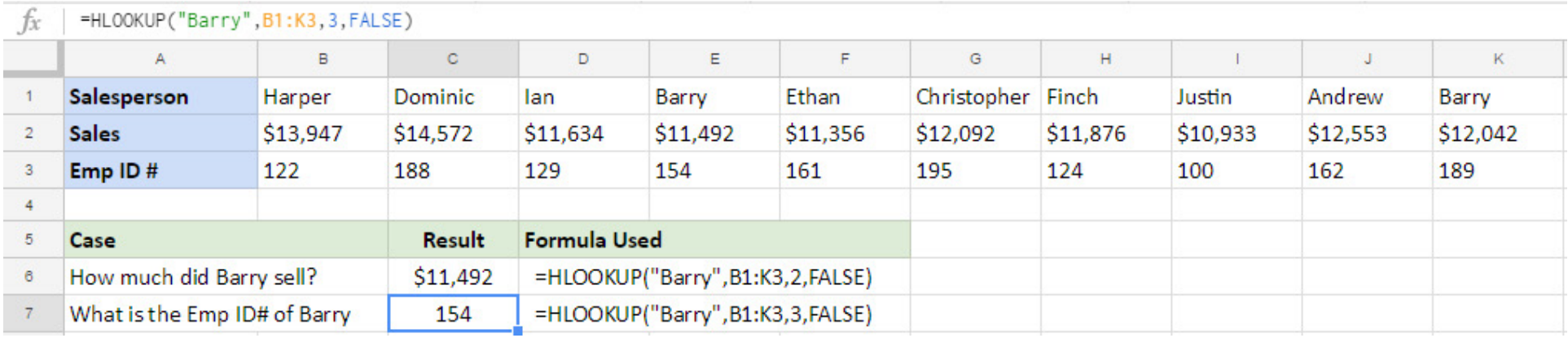
Unlike the previous example, the use of TRUE and FALSE values for the fourth parameter is generating different results. And we can deduce that the results are correct only when we used FALSE for `is_sorted`.

So, what is the bottom line?

> ***Never lie to Google Sheets in the*** `is_sorted` ***parameter! Seemingly, it returns the favor!***

Use case: Multiple matches in the first row

There will be instances where we may encounter multiple instances of the same value in the first row. In the example below, Barry occurred twice. What happens if we use HLOOKUP formula in such scenario? Let's find out.



	A	B	C	D	E	F	G	H	I	J	K
1	**Salesperson**	Harper	Dominic	Ian	Barry	Ethan	Christopher	Finch	Justin	Andrew	Barry
2	**Sales**	$13,947	$14,572	$11,634	$11,492	$11,356	$12,092	$11,876	$10,933	$12,553	$12,042
3	**Emp ID #**	122	188	129	154	161	195	124	100	162	189
4											
5	**Case**		**Result**	**Formula Used**							
6	How much did Barry sell?		$11,492	=HLOOKUP("Barry",B1:K3,2,FALSE)							
7	What is the Emp ID# of Barry		154	=HLOOKUP("Barry",B1:K3,3,FALSE)							

You may have noticed, it picks up the first Barry it encounters in the list. And it doesn't consider the the second Barry within the row.

Use case: Drawbacks with the HLOOKUP formula

There are two problems with this formula:

1. For looking up `search_key`, it always uses the first row within the input range. So it is not possible with HLOOKUP formula to fetch a cell value that is above the lookup row.
2. The formula is not dynamic enough, in that, inserting a row between the input `range` doesn't automatically update the row index.

There's an alternative that solves the above two problems. Please go through the explanation for the INDEX and MATCH formula combination here. It is explained in the context of VLOOKUP formula, but is equally applicable as an HLOOKUP alternative.

Filtering the data

SHEETGO

UNIQUE

In Google Sheets, the UNIQUE formula gives us unique rows from a range, while discarding the duplicates in the process. To understand this better, suppose that we have the following list of names. You'll notice we have two sets of duplicates – Eric and Caroline.

	A
1	**Name**
2	Filip
3	Eric
4	Brett
5	Francis
6	Angela
7	Caroline
8	Charlie
9	Alfred
10	David
11	Dennis
12	Bob
13	Caroline
14	Evan
15	Eric

If you are familiar with Microsoft Excel, we have 'Remove Duplicates' functionality to get rid of any repetitive values in a range. Unfortunately, we do not have such a feature in Google Sheets. So, how do we remove duplicates in Google Sheets? With UNIQUE formula, of course!

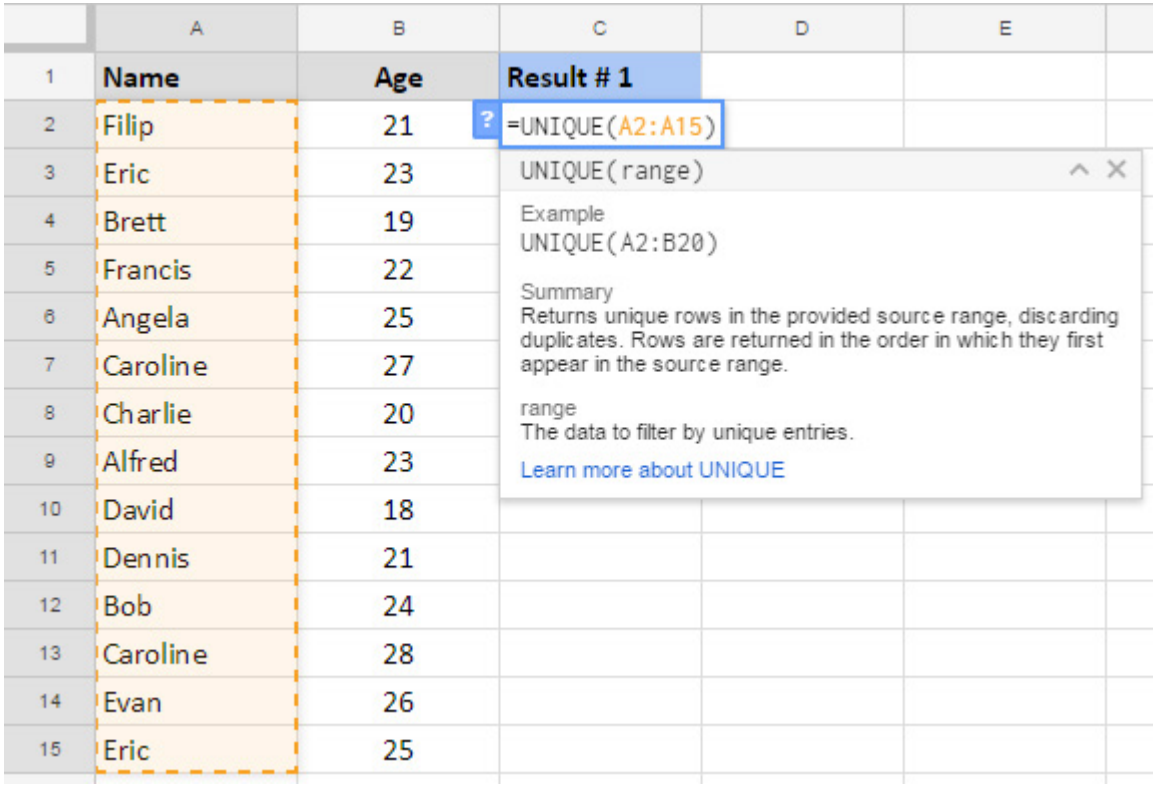
Syntax

`UNIQUE(range)`

- `range` – is the address of the group of cells that the UNIQUE formula references and returns unique or distinct set of values available in the `range`.

Usage: UNIQUE formula

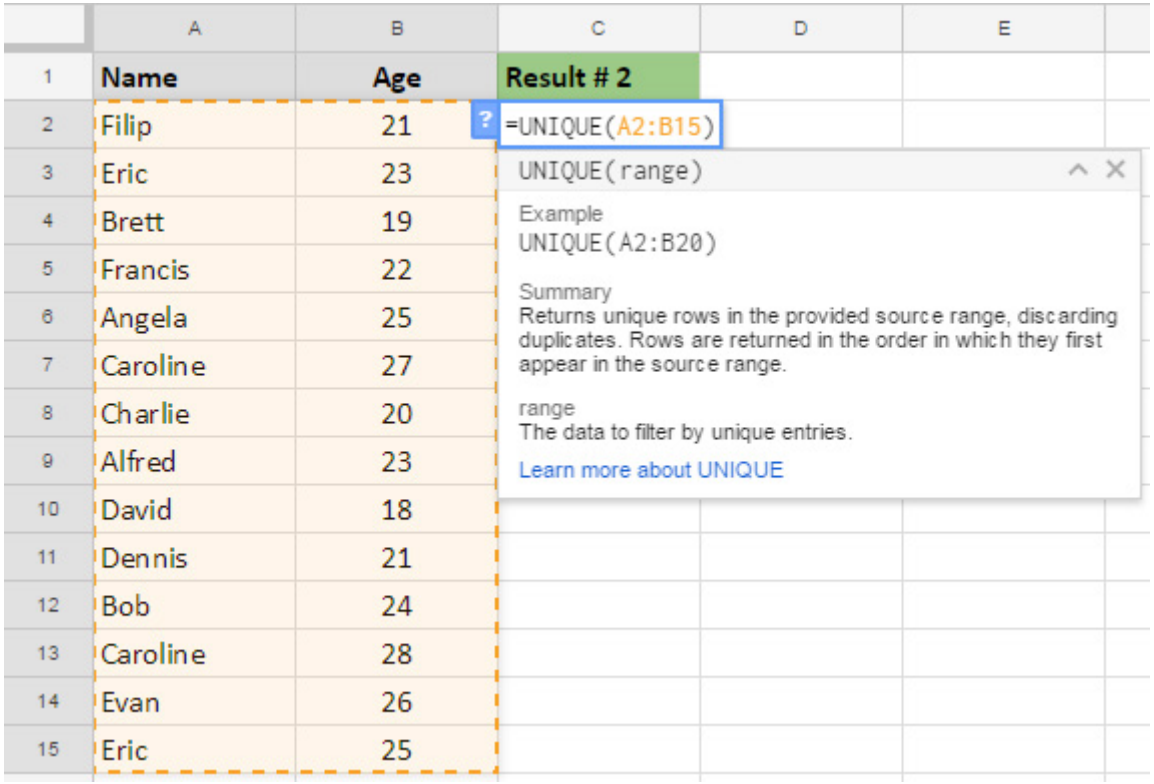
We will use a similar data set as shown in the first image. Please note, the UNIQUE formula is not one dimensional, meaning that it can accept ranges with two or more rows/columns. But let's start with a single dimensional column, in order to be able to grasp the concept better. Here's an example.



As soon as we type in the formula in cell C2, and hit the Enter key, the UNIQUE formula makes Google Sheets return all the unique values from range A2:A15. They start appearing from C2 (which is where the formula is) and flow further downwards as shown in the next image. You'll notice that the duplicate instances, Eric and Caroline are removed in the result column.

	A	B	C	D
1	**Name**	**Age**	**Result # 1**	**Formula Used**
2	Filip	21	Filip	=UNIQUE(A2:A15)
3	Eric	23	Eric	
4	Brett	19	Brett	
5	Francis	22	Francis	
6	Angela	25	Angela	
7	Caroline	27	Caroline	
8	Charlie	20	Charlie	
9	Alfred	23	Alfred	
10	David	18	David	
11	Dennis	21	Dennis	
12	Bob	24	Bob	
13	Caroline	28	Evan	
14	Evan	26		
15	Eric	25		

What happens if we extend the input range to multiple columns? Let's find out. We will use both the Name and Age columns for this purpose as shown in the image below.



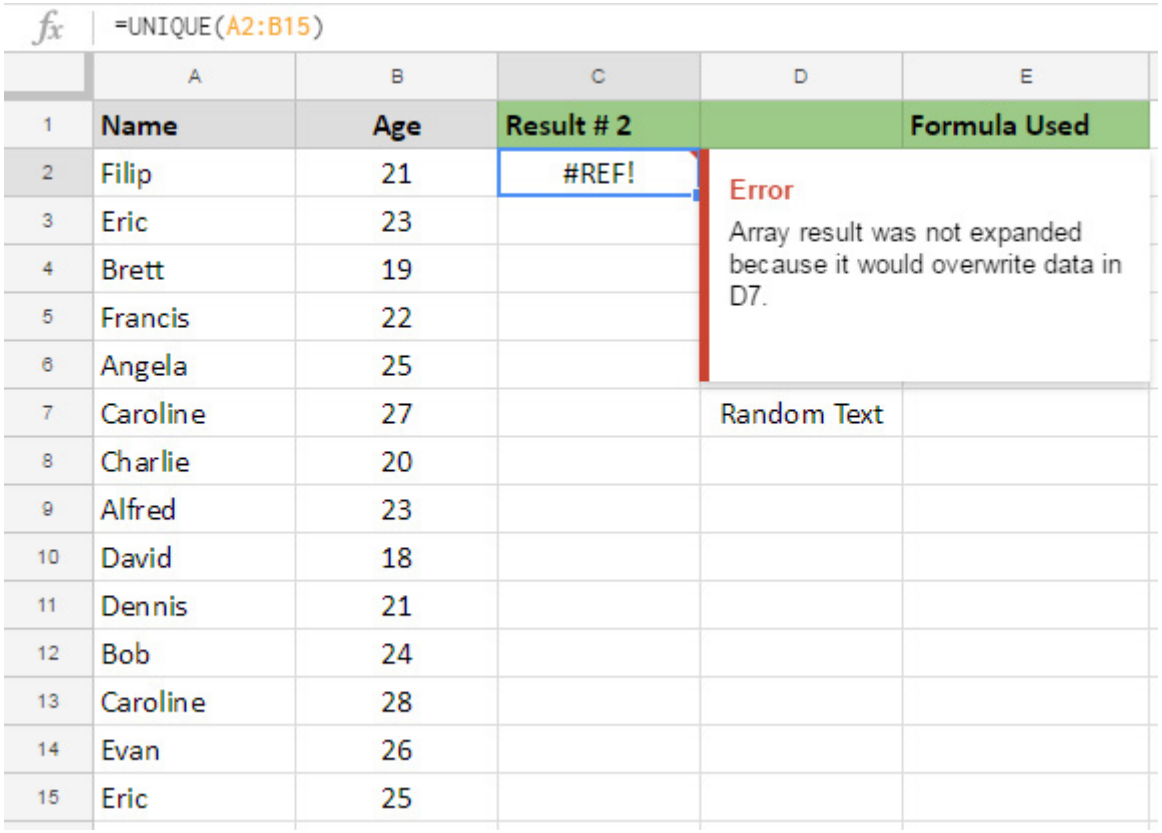
Just like before, the UNIQUE formula returns the distinct values starting from cell C2. We'll notice the output values not only have flown downwards, but also extended one column to the right. That is because, Google Sheets returns as many columns as that of the input range. But that may not be true in case of rows, because it eliminates the duplicates.

	A	B	C	D	E
1	**Name**	**Age**	**Result # 2**		**Formula Used**
2	Filip	21	Filip	21	=UNIQUE(A2:B15)
3	Eric	23	Eric	23	
4	Brett	19	Brett	19	
5	Francis	22	Francis	22	
6	Angela	25	Angela	25	
7	Caroline	27	Caroline	27	
8	Charlie	20	Charlie	20	
9	Alfred	23	Alfred	23	
10	David	18	David	18	
11	Dennis	21	Dennis	21	
12	Bob	24	Bob	24	
13	Caroline	28	Caroline	28	
14	Evan	26	Evan	26	
15	Eric	25	Eric	25	

In the above scenario, though there are duplicate instances of Eric and Caroline, when combined with age, they turn out to be unique. For example, Eric in row # 3 is 23 years old, whereas Eric in row # 15 is 25 years old. Clearly, they are different, and cannot be duplicates. However, had there been another occurrence of Eric with 23 years of age, that definitely qualifies for a duplicate and accordingly won't appear in the output.

Interesting notes

- For the UNIQUE formula to work, we need to give it sufficient real estate to display its output. If, in the above example, we have certain values in column D, where as the output from UNIQUE formula is expected to flow into that column, Google Sheets accordingly gives a #REF! error.



	A	B	C	D	E
1	**Name**	**Age**	**Result # 2**		**Formula Used**
2	Filip	21	#REF!		
3	Eric	23			
4	Brett	19			
5	Francis	22			
6	Angela	25			
7	Caroline	27		Random Text	
8	Charlie	20			
9	Alfred	23			
10	David	18			
11	Dennis	21			
12	Bob	24			
13	Caroline	28			
14	Evan	26			
15	Eric	25			

- Google Sheets doesn't allow us to delete the values that returned by the UNIQUE formula, unless we are doing so from the cell where we have entered the formula (in the above cases, C2).
- If we need to copy and paste the distinct values that the UNIQUE formula returned, we need to do this. Copy the distinct values, and select a destination cell. Now, navigate to Edit – Paste – Paste Special – Paste values only. That way, the formula is removed and the values are retained.

SORT

The SORT formula in Google Sheets, helps us sort and return the rows in a range either in ascending or descending order. It also allows us to add multiple criteria across columns, similar to that of 'Sort Range' functionality within Google Sheets. Except, with the SORT formula, we will be able to generate a new set of data while the original data remains intact. The test data we'll consider in order to explore the SORT formula is as shown in the image below.

	A	B	C
1	**ID**	**Name**	**Location**
2	1304	Francis	IL
3	1101	Charlie	CO
4	1913	Evan	HI
5	1129	Eric Sr.	AR
6	1927	Alfred	ID
7	1683	Bob	AS
8	1661	David	GA
9	1432	Filip	DC
10	1801	Caroline A.	AZ
11	1407	Brett	DE
12	1304	Dennis	AK
13	1773	Eric Jr.	FL
14	1349	Angela	GU
15	1515	Caroline S.	CT

Syntax

SORT(range, sort_column, is_ascending, [sort_column2, is_ascending2, ...])

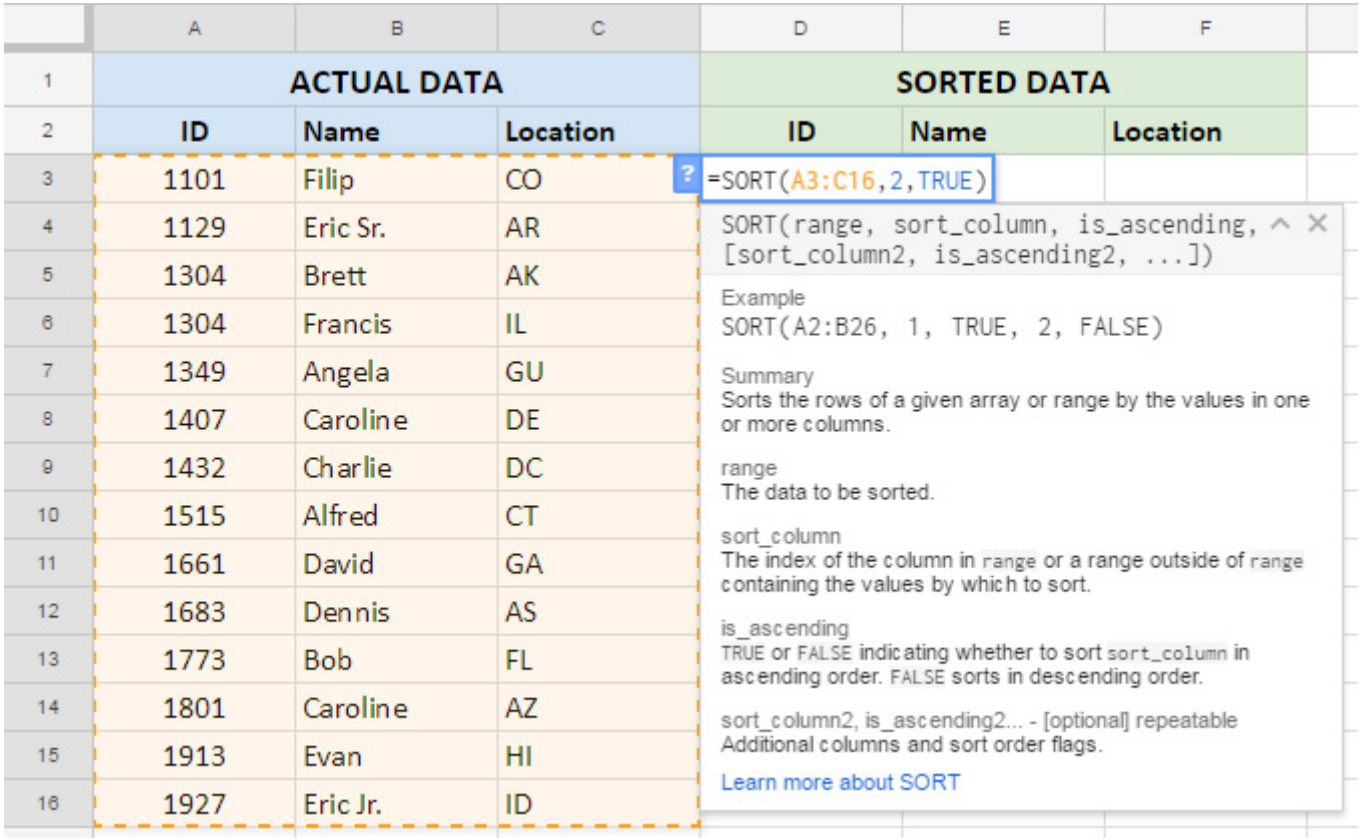
- `range` – is the reference to the group of cells that the SORT formula uses that to arrange the rows as specified in the next two parameters.
- `sort_column` – this can be two things, which essentially are the same.
  - An index number to specify a column number in the `range` on which Google Sheets needs to sort.
  - A reference (or address) of a column within the range of cells that we are sorting. Please note the column reference should include one single column that covers all the existing rows within the `range`. Otherwise, the formula returns an #N/A! error.
- `is_ascending` – this parameter takes in the values TRUE or FALSE to specify the order of data arrangement. It will be sorted in ascending order if TRUE, and in descending order if the option is FALSE.
- `sort_column2, is_ascending2, ...`
  - While the syntax for these parameters remains the same as that of the above two, these are optional and will be effective in case we are going for multiple column sort.

Use Cases: SORT Formula

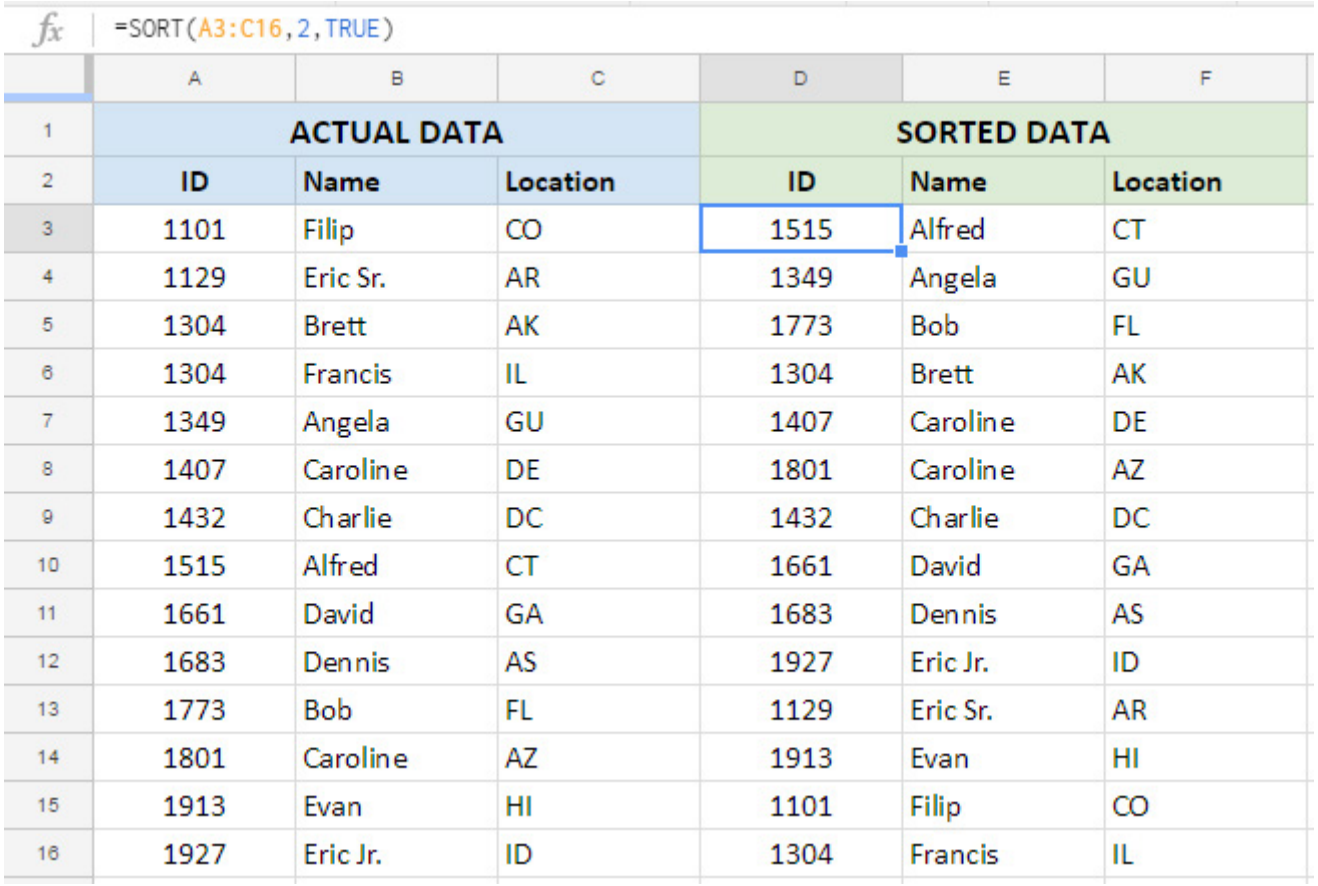
For the ease of reference and understanding, I will be showing both the reference data as well as the resulting data that Google Sheets returned. Following are some of the use cases corresponding to this formula.

Use Case # 1: SORT(range, column_index, ascending_order)

I'm keying in the formula in cell D3, and will hit Enter key. We'll see the returned data flows from D3 towards right and further down. The number of columns and rows returned by this formula is same as that of the input range. So it is very important that we keep the cells where the data is expected to flow, clear of any values. Otherwise, the formula returns #REF! error.

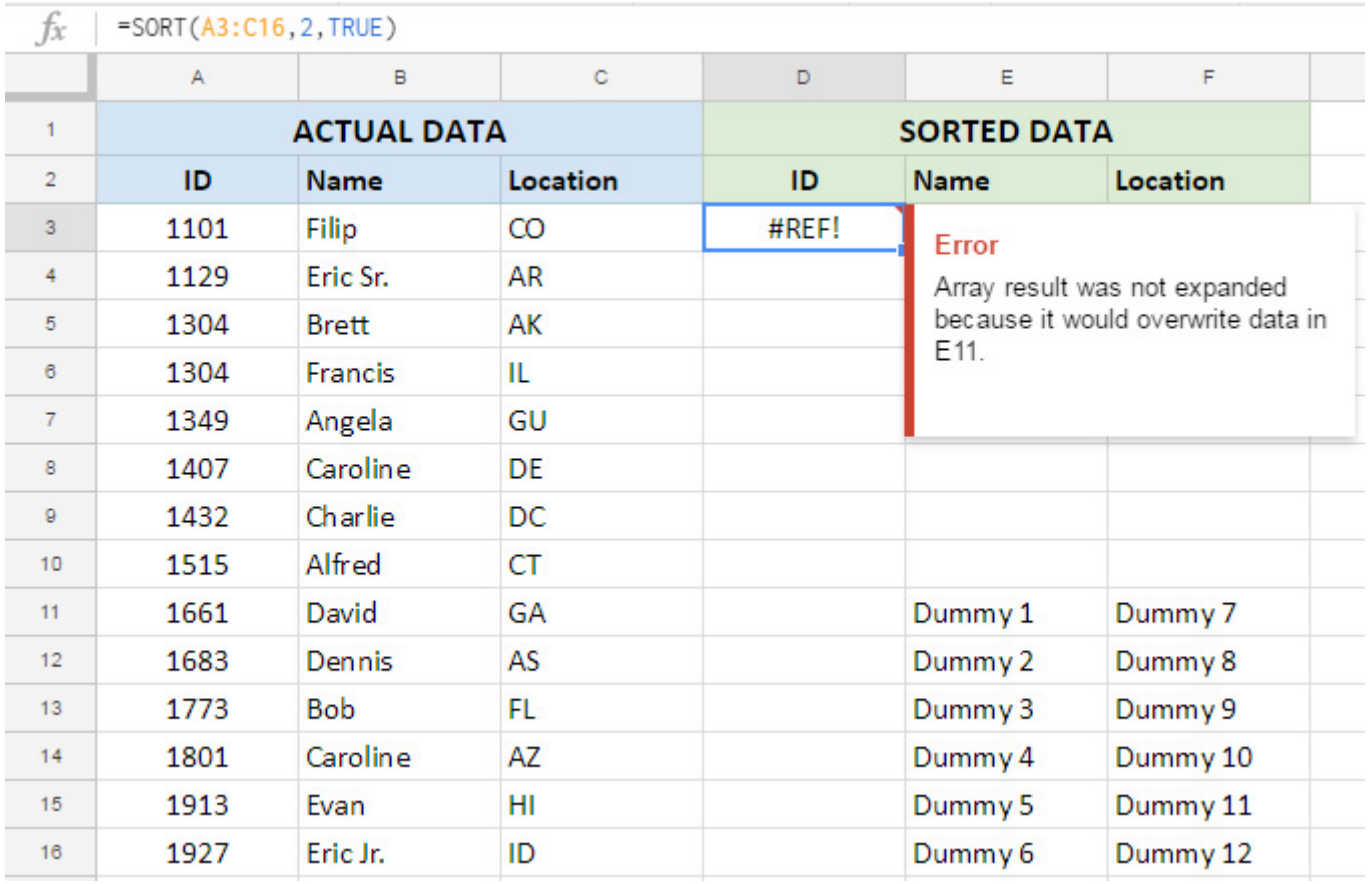


As you may have inferred from the formula, we are sorting the range A3;C16 on the second (Name) column in ascending order.



	A	B	C	D	E	F
1	ACTUAL DATA			SORTED DATA		
2	ID	Name	Location	ID	Name	Location
3	1101	Filip	CO	1515	Alfred	CT
4	1129	Eric Sr.	AR	1349	Angela	GU
5	1304	Brett	AK	1773	Bob	FL
6	1304	Francis	IL	1304	Brett	AK
7	1349	Angela	GU	1407	Caroline	DE
8	1407	Caroline	DE	1801	Caroline	AZ
9	1432	Charlie	DC	1432	Charlie	DC
10	1515	Alfred	CT	1661	David	GA
11	1661	David	GA	1683	Dennis	AS
12	1683	Dennis	AS	1927	Eric Jr.	ID
13	1773	Bob	FL	1129	Eric Sr.	AR
14	1801	Caroline	AZ	1913	Evan	HI
15	1913	Evan	HI	1101	Filip	CO
16	1927	Eric Jr.	ID	1304	Francis	IL

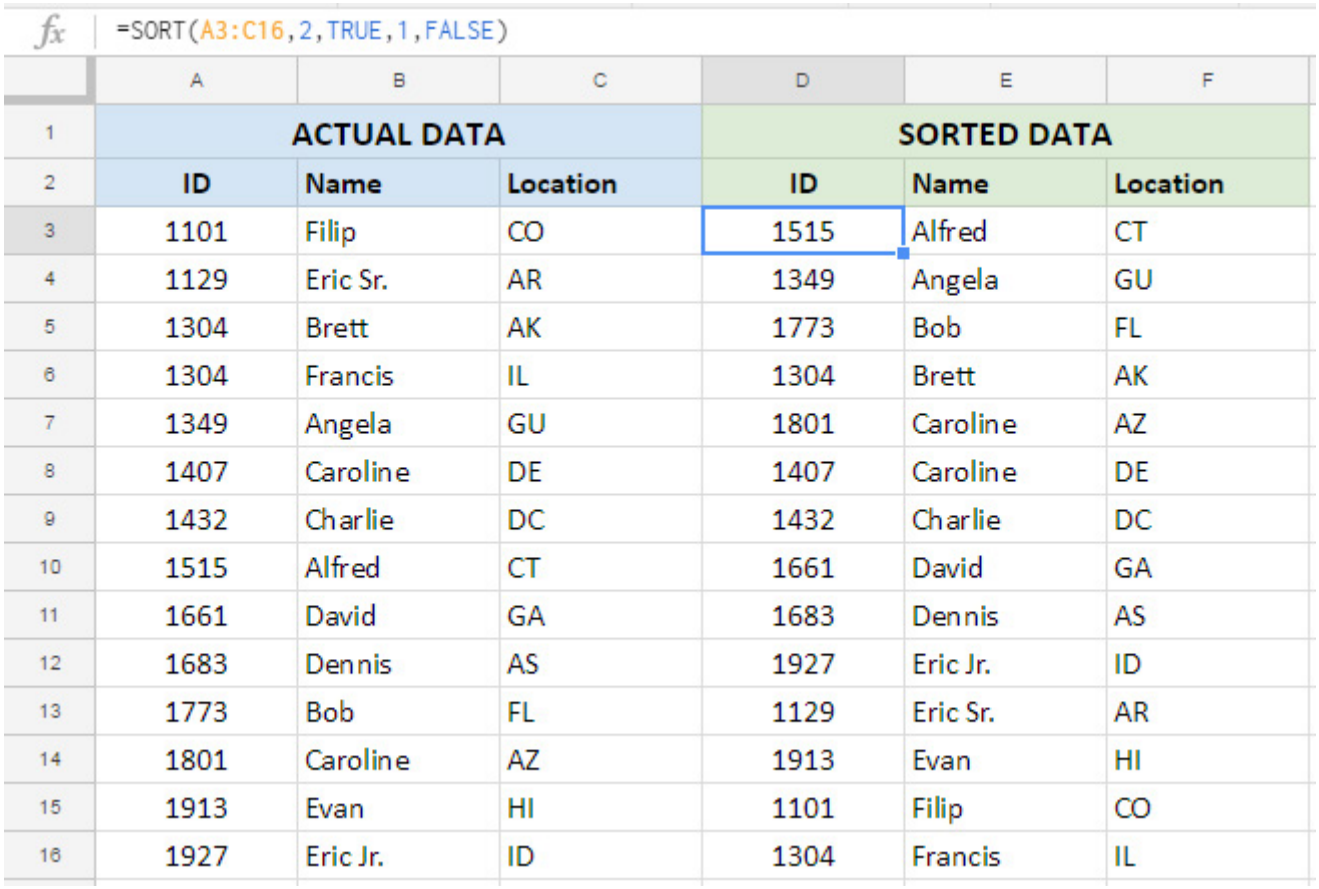
If by any chance, there are values present in columns C or D or E, the formula returns an error as shown in the image below.



	A	B	C	D	E	F
1	ACTUAL DATA			SORTED DATA		
2	ID	Name	Location	ID	Name	Location
3	1101	Filip	CO	#REF!		
4	1129	Eric Sr.	AR			
5	1304	Brett	AK			
6	1304	Francis	IL			
7	1349	Angela	GU			
8	1407	Caroline	DE			
9	1432	Charlie	DC			
10	1515	Alfred	CT			
11	1661	David	GA		Dummy 1	Dummy 7
12	1683	Dennis	AS		Dummy 2	Dummy 8
13	1773	Bob	FL		Dummy 3	Dummy 9
14	1801	Caroline	AZ		Dummy 4	Dummy 10
15	1913	Evan	HI		Dummy 5	Dummy 11
16	1927	Eric Jr.	ID		Dummy 6	Dummy 12

Use Case # 2: SORT(range, column_index1, ascending_order1, column_index2, descending_order2)

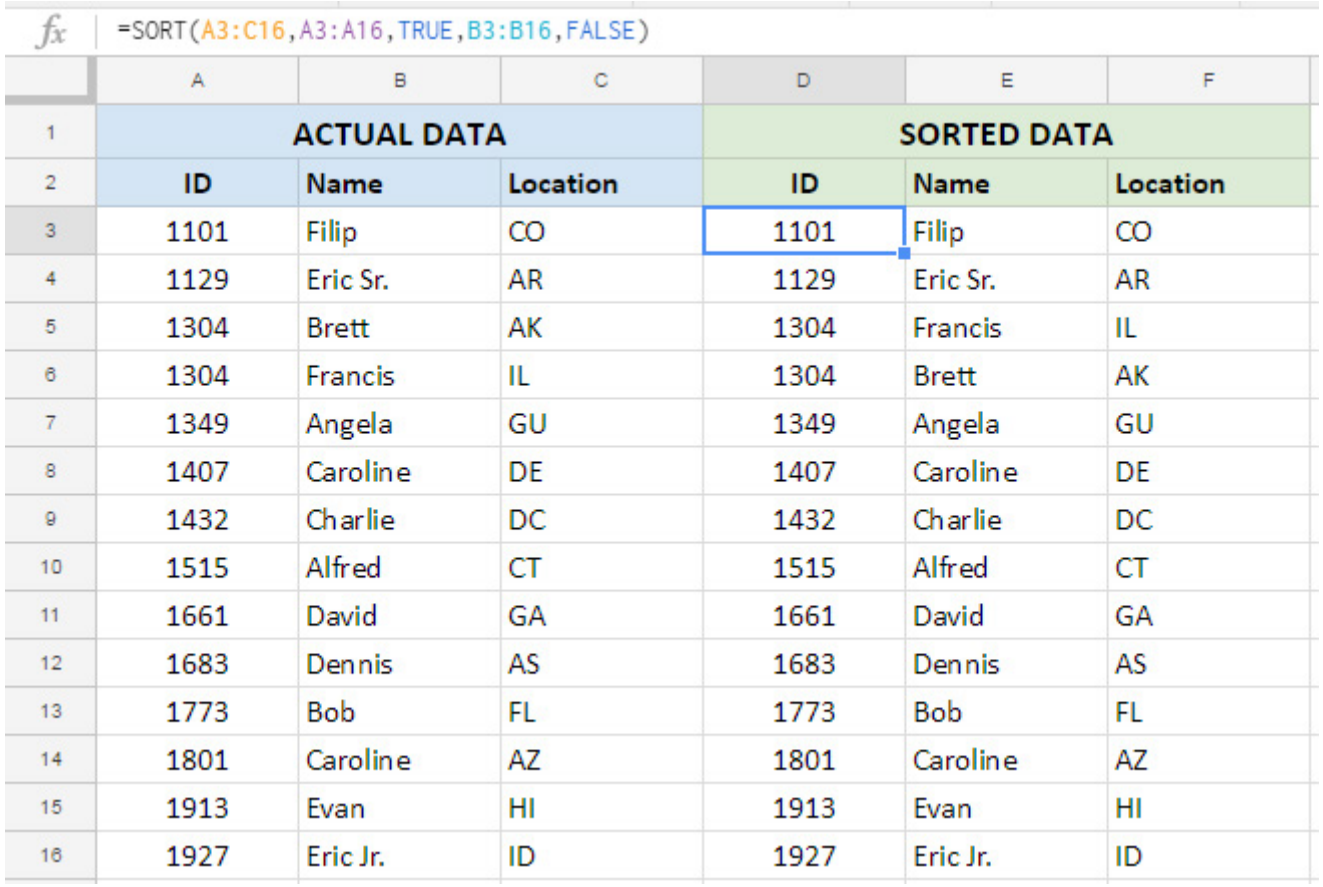
This is a simple case of sorting on multiple columns. With the formula below, we are first sorting the range in ascending order on the second (Name) column, and then in descending order on the first (ID) column.



	A	B	C	D	E	F
1	ACTUAL DATA			SORTED DATA		
2	ID	Name	Location	ID	Name	Location
3	1101	Filip	CO	1515	Alfred	CT
4	1129	Eric Sr.	AR	1349	Angela	GU
5	1304	Brett	AK	1773	Bob	FL
6	1304	Francis	IL	1304	Brett	AK
7	1349	Angela	GU	1801	Caroline	AZ
8	1407	Caroline	DE	1407	Caroline	DE
9	1432	Charlie	DC	1432	Charlie	DC
10	1515	Alfred	CT	1661	David	GA
11	1661	David	GA	1683	Dennis	AS
12	1683	Dennis	AS	1927	Eric Jr.	ID
13	1773	Bob	FL	1129	Eric Sr.	AR
14	1801	Caroline	AZ	1913	Evan	HI
15	1913	Evan	HI	1101	Filip	CO
16	1927	Eric Jr.	ID	1304	Francis	IL

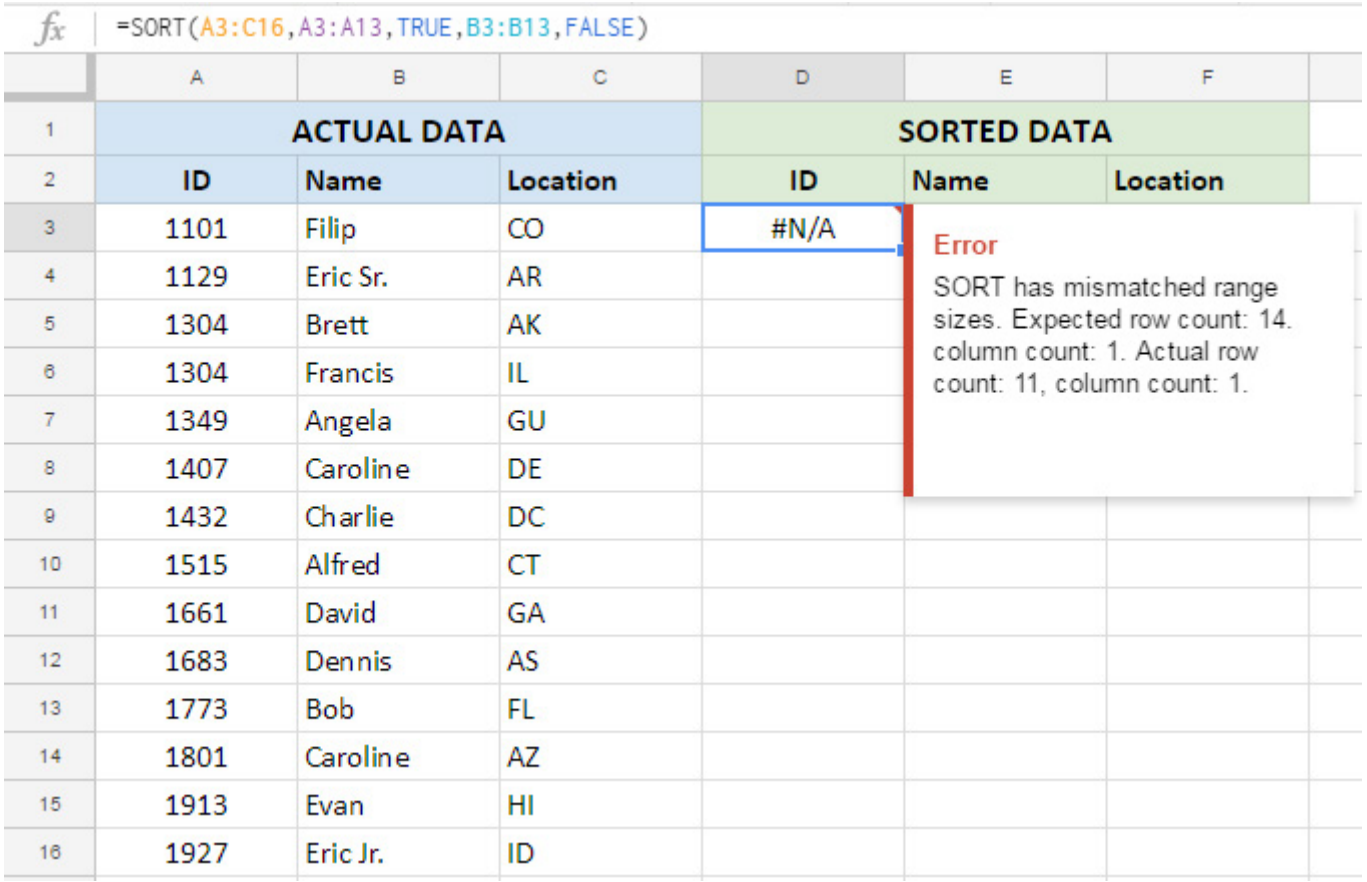
Use Case # 3: SORT(range, column_reference1, ascending_order1, column_reference2, descending_order2)

Now, for the sort_column parameter, we'll try inputting the column references instead of column indexes.



	A	B	C	D	E	F
1	ACTUAL DATA			SORTED DATA		
2	ID	Name	Location	ID	Name	Location
3	1101	Filip	CO	1101	Filip	CO
4	1129	Eric Sr.	AR	1129	Eric Sr.	AR
5	1304	Brett	AK	1304	Francis	IL
6	1304	Francis	IL	1304	Brett	AK
7	1349	Angela	GU	1349	Angela	GU
8	1407	Caroline	DE	1407	Caroline	DE
9	1432	Charlie	DC	1432	Charlie	DC
10	1515	Alfred	CT	1515	Alfred	CT
11	1661	David	GA	1661	David	GA
12	1683	Dennis	AS	1683	Dennis	AS
13	1773	Bob	FL	1773	Bob	FL
14	1801	Caroline	AZ	1801	Caroline	AZ
15	1913	Evan	HI	1913	Evan	HI
16	1927	Eric Jr.	ID	1927	Eric Jr.	ID

We may be curious, as to what happens when the column reference doesn't cover all the rows within the range. Let's try that (hint: SORT formula returns an #N/A! error!).



FILTER

The FILTER formula in Google Sheets helps us filter and return the rows in a range that meet specified criteria. And, we can add multiple criteria across columns. Accordingly, we will be able to generate a new set of data while the original data remains intact.

Syntax

`FILTER(range, condition1, [condition2, ...])`

- `range` – is the address reference to the range of cells that the formula filters.
- `condition1` – is an array, row or column equal in length or width as that of the corresponding first row or column of range respectively. It contains evaluated TRUE or FALSE values.
- `condition2 ...` – these are optional and additional arrays, rows or columns containing evaluated TRUE or FALSE values to specify if the corresponding row or column within the `range` needs consideration for the filtering process.

Please note that we cannot use both row and column conditions in the same formula. Either all the conditions should be of column type or row type, and the FILTER formula doesn't allow mixing them.

Usage: FILTER Formula

Here is the sample data on which we will try various combinations of the FILTER formula, and understand its behavior.

	A	B	C	D
1	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**
2	Nuts	Almonds	579	49.93
3	Fruits	Apple	52	0.17
4	Vegetables	Asparagus	20	0.12
5	Fruits	Avocado	160	14.66
6	Vegetables	Broccoli	34	0.37
7	Vegetables	Cabbage	25	0.1
8	Vegetables	Cauliflower	25	0.28
9	Fruits	Kiwi Fruit	61	0.52
10	Nuts	Walnuts	654	65.21

Case 1: FILTER(range, condition1)

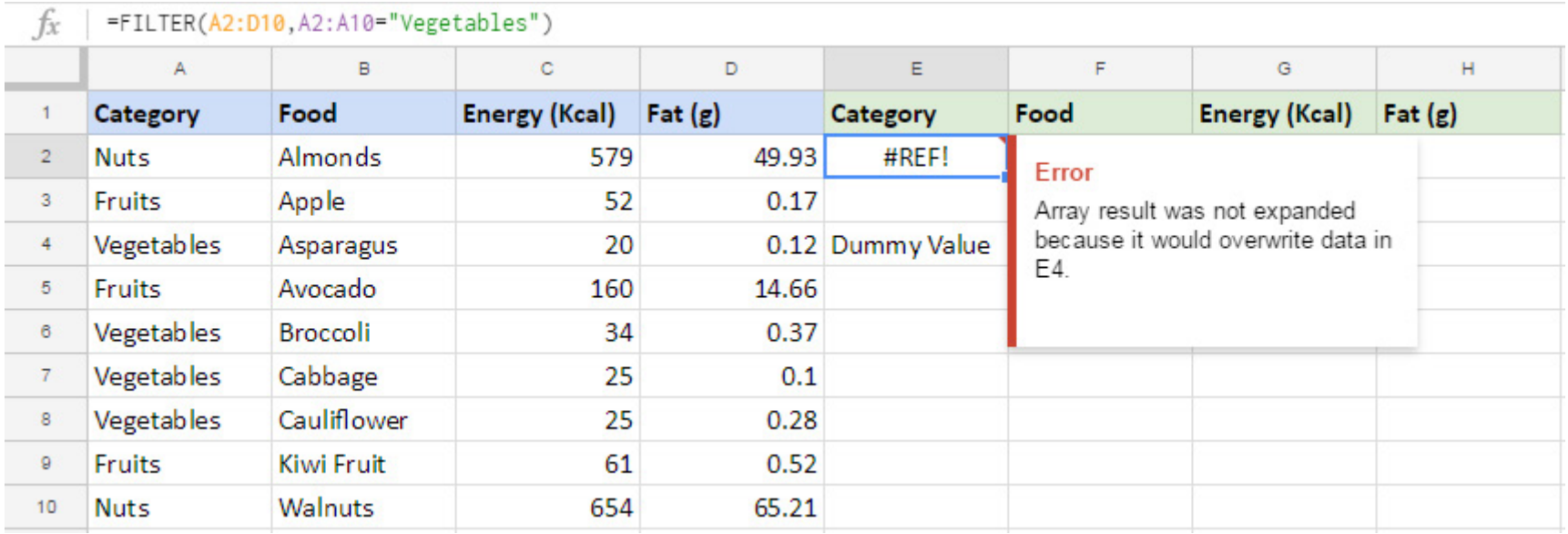
In this example, we will use just one condition and see how it works.

```
=FILTER(A2:D10,A2:A10="Vegetables")
```

	A	B	C	D	E	F	G	H
1	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**
2	Nuts	Almonds	579	49.93	Vegetables	Asparagus	20	0.12
3	Fruits	Apple	52	0.17	Vegetables	Broccoli	34	0.37
4	Vegetables	Asparagus	20	0.12	Vegetables	Cabbage	25	0.1
5	Fruits	Avocado	160	14.66	Vegetables	Cauliflower	25	0.28
6	Vegetables	Broccoli	34	0.37				
7	Vegetables	Cabbage	25	0.1				
8	Vegetables	Cauliflower	25	0.28				
9	Fruits	Kiwi Fruit	61	0.52				
10	Nuts	Walnuts	654	65.21				

In the above example, our condition to filter on the first column is “Vegetables”. So the FILTER formula fetches all those rows where the first column has the value “Vegetables”.

Please note that we have keyed in the formula in the cell E1, and accordingly the returned data flows from E1 towards the right and further down. The number of columns the formula returned is same as that of the input range. However, the number of rows might vary based on the filter conditions. So, it is very important that we keep the cells, where we expect the data to flow, clear of any values. Otherwise, the formula returns #REF! error. This is demonstrated in the screenshot below.



	A	B	C	D	E	F	G	H
1	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**
2	Nuts	Almonds	579	49.93	#REF!			
3	Fruits	Apple	52	0.17				
4	Vegetables	Asparagus	20	0.12	Dummy Value			
5	Fruits	Avocado	160	14.66				
6	Vegetables	Broccoli	34	0.37				
7	Vegetables	Cabbage	25	0.1				
8	Vegetables	Cauliflower	25	0.28				
9	Fruits	Kiwi Fruit	61	0.52				
10	Nuts	Walnuts	654	65.21				

Case 2: FILTER(range, condition1, condition2)

Let us add one more filter condition and see what happens.

```
=FILTER(A2:D10,A2:A10="Vegetables",C2:C10>20)
```

	A	B	C	D	E	F	G	H
1	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**
2	Nuts	Almonds	579	49.93	Vegetables	Broccoli	34	0.37
3	Fruits	Apple	52	0.17	Vegetables	Cabbage	25	0.1
4	Vegetables	Asparagus	20	0.12	Vegetables	Cauliflower	25	0.28
5	Fruits	Avocado	160	14.66				
6	Vegetables	Broccoli	34	0.37				
7	Vegetables	Cabbage	25	0.1				
8	Vegetables	Cauliflower	25	0.28				
9	Fruits	Kiwi Fruit	61	0.52				
10	Nuts	Walnuts	654	65.21				

We essentially asked Google Sheets to show those rows that belong to Vegetables food category and whose energy is greater than 20 Kcal, and the FILTER formula obliged!

Case 3: FILTER(range, condition1, condition2, condition3)

We will add one more condition and see the outcome.

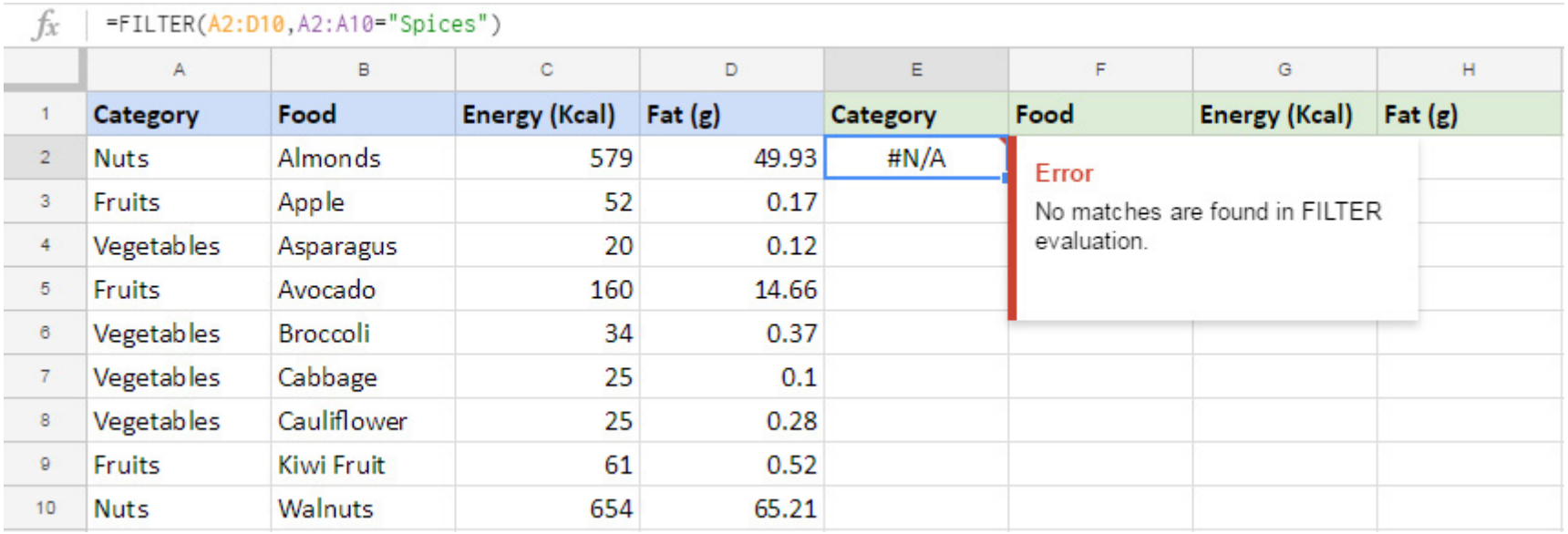
```
=FILTER(A2:D10,A2:A10="Vegetables",C2:C10>20,D2:D10<0.3)
```

	A	B	C	D	E	F	G	H
1	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**
2	Nuts	Almonds	579	49.93	Vegetables	Cabbage	25	0.1
3	Fruits	Apple	52	0.17	Vegetables	Cauliflower	25	0.28
4	Vegetables	Asparagus	20	0.12				
5	Fruits	Avocado	160	14.66				
6	Vegetables	Broccoli	34	0.37				
7	Vegetables	Cabbage	25	0.1				
8	Vegetables	Cauliflower	25	0.28				
9	Fruits	Kiwi Fruit	61	0.52				
10	Nuts	Walnuts	654	65.21				

And it works like a charm!

Case 4: When there is no match!

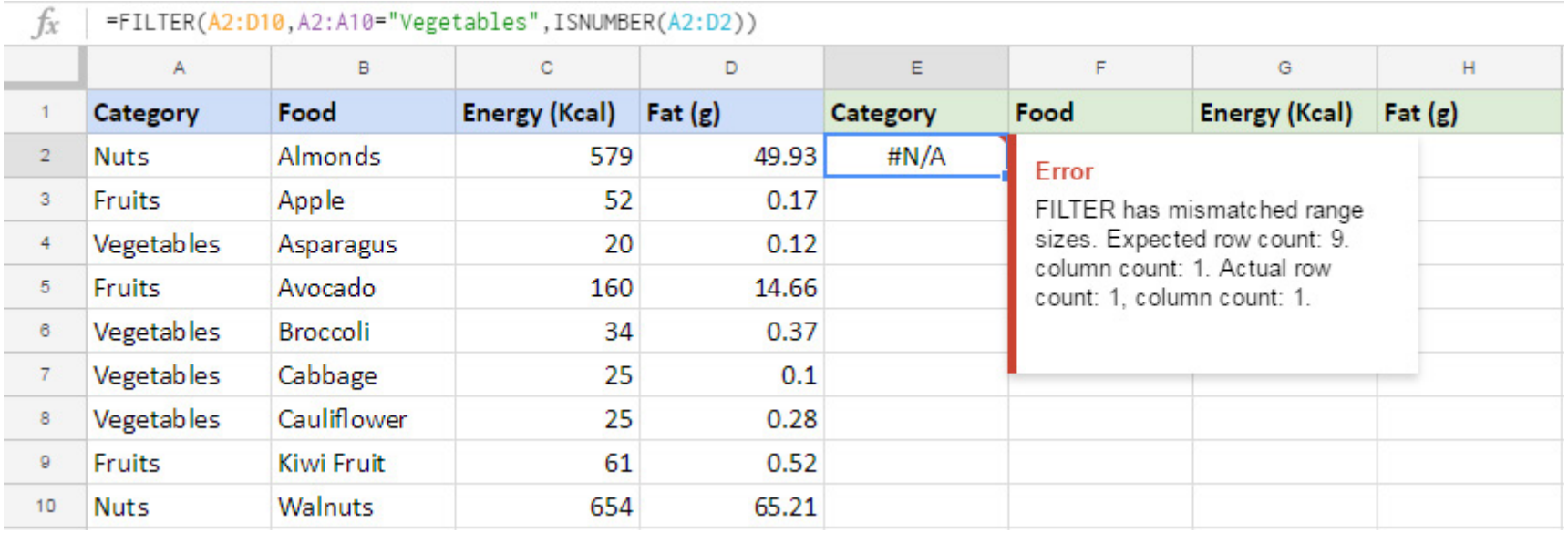
What would happen if it doesn't find any rows that match the filter conditions? Of course, it throws up an error!



	A	B	C	D	E	F	G	H
1	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**
2	Nuts	Almonds	579	49.93	#N/A			
3	Fruits	Apple	52	0.17				
4	Vegetables	Asparagus	20	0.12				
5	Fruits	Avocado	160	14.66				
6	Vegetables	Broccoli	34	0.37				
7	Vegetables	Cabbage	25	0.1				
8	Vegetables	Cauliflower	25	0.28				
9	Fruits	Kiwi Fruit	61	0.52				
10	Nuts	Walnuts	654	65.21				

Case 5: When we mix things up!

We already know we can't input row and column type conditions within a single FILTER formula. Let us try that anyway and see what Google Sheets has to say.



	A	B	C	D	E	F	G	H
1	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**	**Category**	**Food**	**Energy (Kcal)**	**Fat (g)**
2	Nuts	Almonds	579	49.93	#N/A			
3	Fruits	Apple	52	0.17				
4	Vegetables	Asparagus	20	0.12				
5	Fruits	Avocado	160	14.66				
6	Vegetables	Broccoli	34	0.37				
7	Vegetables	Cabbage	25	0.1				
8	Vegetables	Cauliflower	25	0.28				
9	Fruits	Kiwi Fruit	61	0.52				
10	Nuts	Walnuts	654	65.21				

QUERY

The QUERY formula in Google Sheets is quite a powerful and versatile function. So much so that, we can call it a one-stop-shop for all the logical, lookup, summation, counting, averaging, filtering and sorting requirements. It helps us fetch specific information from a data set, through a query statement. Much like fetching result sets from a database using queries. The syntax of the query language used in this formula is similar to SQL.

Syntax

`QUERY(data, query, [headers])`

- `data` – is the reference to the range of cells on which we want to query upon.
- `query` – is the text using which the QUERY formula churns out the information we are looking for from the data set. Since it is expected to be a string, it has to be enclosed within a set of quotes. Or, it can also be a reference to a cell, where the query text is stored.
- `headers` – is an optional parameter that indicates the number of header rows at the top of the data. If left out, Google Sheets guesses the value based on the content within the data.

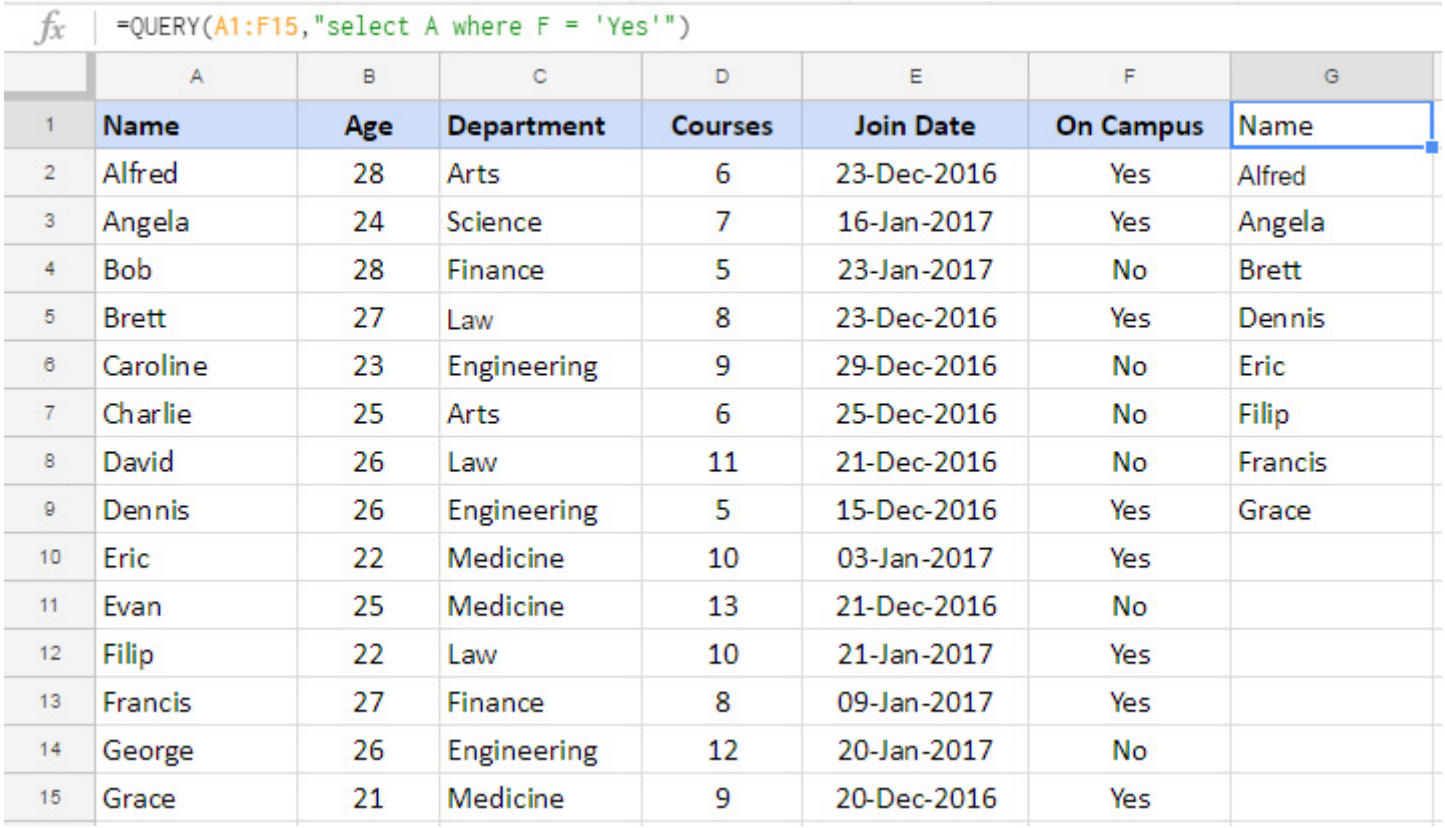
Usage: Query Formula

First of all, to understand how the formula is put to use, let us consider the following sample data. It consists of information corresponding to a list of students who have enrolled into various courses at a university. For all the demonstration purposes, we'll enter the formula in the cell G1. And, it will be displayed in the formula bar in the snapshots.

	A	B	C	D	E	F
1	**Name**	**Age**	**Department**	**Courses**	**Join Date**	**On Campus**
2	Alfred	28	Arts	6	23-Dec-2016	Yes
3	Angela	24	Science	7	16-Jan-2017	Yes
4	Bob	28	Finance	5	23-Jan-2017	No
5	Brett	27	Law	8	23-Dec-2016	Yes
6	Caroline	23	Engineering	9	29-Dec-2016	No
7	Charlie	25	Arts	6	25-Dec-2016	No
8	David	26	Law	11	21-Dec-2016	No
9	Dennis	26	Engineering	5	15-Dec-2016	Yes
10	Eric	22	Medicine	10	03-Jan-2017	Yes
11	Evan	25	Medicine	13	21-Dec-2016	No
12	Filip	22	Law	10	21-Jan-2017	Yes
13	Francis	27	Finance	8	09-Jan-2017	Yes
14	George	26	Engineering	12	20-Jan-2017	No
15	Grace	21	Medicine	9	20-Dec-2016	Yes

Example # 1:

We will start off with a very fundamental demonstration. So, we use the QUERY formula to fetch the names of the students who are residing on campus.



```
=QUERY(A1:F15,"select A where F = 'Yes'")
```

	A	B	C	D	E	F	G
1	**Name**	**Age**	**Department**	**Courses**	**Join Date**	**On Campus**	Name
2	Alfred	28	Arts	6	23-Dec-2016	Yes	Alfred
3	Angela	24	Science	7	16-Jan-2017	Yes	Angela
4	Bob	28	Finance	5	23-Jan-2017	No	Brett
5	Brett	27	Law	8	23-Dec-2016	Yes	Dennis
6	Caroline	23	Engineering	9	29-Dec-2016	No	Eric
7	Charlie	25	Arts	6	25-Dec-2016	No	Filip
8	David	26	Law	11	21-Dec-2016	No	Francis
9	Dennis	26	Engineering	5	15-Dec-2016	Yes	Grace
10	Eric	22	Medicine	10	03-Jan-2017	Yes	
11	Evan	25	Medicine	13	21-Dec-2016	No	
12	Filip	22	Law	10	21-Jan-2017	Yes	
13	Francis	27	Finance	8	09-Jan-2017	Yes	
14	George	26	Engineering	12	20-Jan-2017	No	
15	Grace	21	Medicine	9	20-Dec-2016	Yes	

Example # 2:

Having dealt with the basic example, let us now try fetching the names of the students who are NOT residing on campus.

```
=QUERY(A1:F15,"select A where F <> 'Yes'")
```

	A	B	C	D	E	F	G
1	**Name**	**Age**	**Department**	**Courses**	**Join Date**	**On Campus**	Name
2	Alfred	28	Arts	6	23-Dec-2016	Yes	Bob
3	Angela	24	Science	7	16-Jan-2017	Yes	Caroline
4	Bob	28	Finance	5	23-Jan-2017	No	Charlie
5	Brett	27	Law	8	23-Dec-2016	Yes	David
6	Caroline	23	Engineering	9	29-Dec-2016	No	Evan
7	Charlie	25	Arts	6	25-Dec-2016	No	George
8	David	26	Law	11	21-Dec-2016	No	
9	Dennis	26	Engineering	5	15-Dec-2016	Yes	
10	Eric	22	Medicine	10	03-Jan-2017	Yes	
11	Evan	25	Medicine	13	21-Dec-2016	No	
12	Filip	22	Law	10	21-Jan-2017	Yes	
13	Francis	27	Finance	8	09-Jan-2017	Yes	
14	George	26	Engineering	12	20-Jan-2017	No	
15	Grace	21	Medicine	9	20-Dec-2016	Yes	

Example # 3:

Now, we will fetch the names, ages, departments of the students whose have taken more than 7 courses.

```
=QUERY(A1:F15,"select A, B, C where D > 7")
```

	A	B	C	D	E	F	G	H	I
1	**Name**	**Age**	**Department**	**Courses**	**Join Date**	**On Campus**	Name	Age	Department
2	Alfred	28	Arts	6	23-Dec-2016	Yes	Brett	27	Law
3	Angela	24	Science	7	16-Jan-2017	Yes	Caroline	23	Engineering
4	Bob	28	Finance	5	23-Jan-2017	No	David	26	Law
5	Brett	27	Law	8	23-Dec-2016	Yes	Eric	22	Medicine
6	Caroline	23	Engineering	9	29-Dec-2016	No	Evan	25	Medicine
7	Charlie	25	Arts	6	25-Dec-2016	No	Filip	22	Law
8	David	26	Law	11	21-Dec-2016	No	Francis	27	Finance
9	Dennis	26	Engineering	5	15-Dec-2016	Yes	George	26	Engineering
10	Eric	22	Medicine	10	03-Jan-2017	Yes	Grace	21	Medicine
11	Evan	25	Medicine	13	21-Dec-2016	No			
12	Filip	22	Law	10	21-Jan-2017	Yes			
13	Francis	27	Finance	8	09-Jan-2017	Yes			
14	George	26	Engineering	12	20-Jan-2017	No			
15	Grace	21	Medicine	9	20-Dec-2016	Yes			

Example # 4:

We will now attempt taking this a step further. We bring up the names, departments, join dates of the students aged 25 or below, and have joined the university between 25-Dec-2016 and 20-Jan-2017. Please note, in the query text, the dates always have to go with yyyy-mm-dd format, enclosed within single quotes.

```
=QUERY(A1:F15,"select A, C, E where B <= 25 and E >= date '2016-12-25' and E <= date '2017-01-20'")
```

	A	B	C	D	E	F	G	H	I
1	**Name**	**Age**	**Department**	**Courses**	**Join Date**	**On Campus**	Name	Department	Join Date
2	Alfred	28	Arts	6	23-Dec-2016	Yes	Angela	Science	1/16/2017
3	Angela	24	Science	7	16-Jan-2017	Yes	Caroline	Engineering	12/29/2016
4	Bob	28	Finance	5	23-Jan-2017	No	Charlie	Arts	12/25/2016
5	Brett	27	Law	8	23-Dec-2016	Yes	Eric	Medicine	1/3/2017
6	Caroline	23	Engineering	9	29-Dec-2016	No			
7	Charlie	25	Arts	6	25-Dec-2016	No			
8	David	26	Law	11	21-Dec-2016	No			
9	Dennis	26	Engineering	5	15-Dec-2016	Yes			
10	Eric	22	Medicine	10	03-Jan-2017	Yes			
11	Evan	25	Medicine	13	21-Dec-2016	No			
12	Filip	22	Law	10	21-Jan-2017	Yes			
13	Francis	27	Finance	8	09-Jan-2017	Yes			
14	George	26	Engineering	12	20-Jan-2017	No			
15	Grace	21	Medicine	9	20-Dec-2016	Yes			

Example # 5:

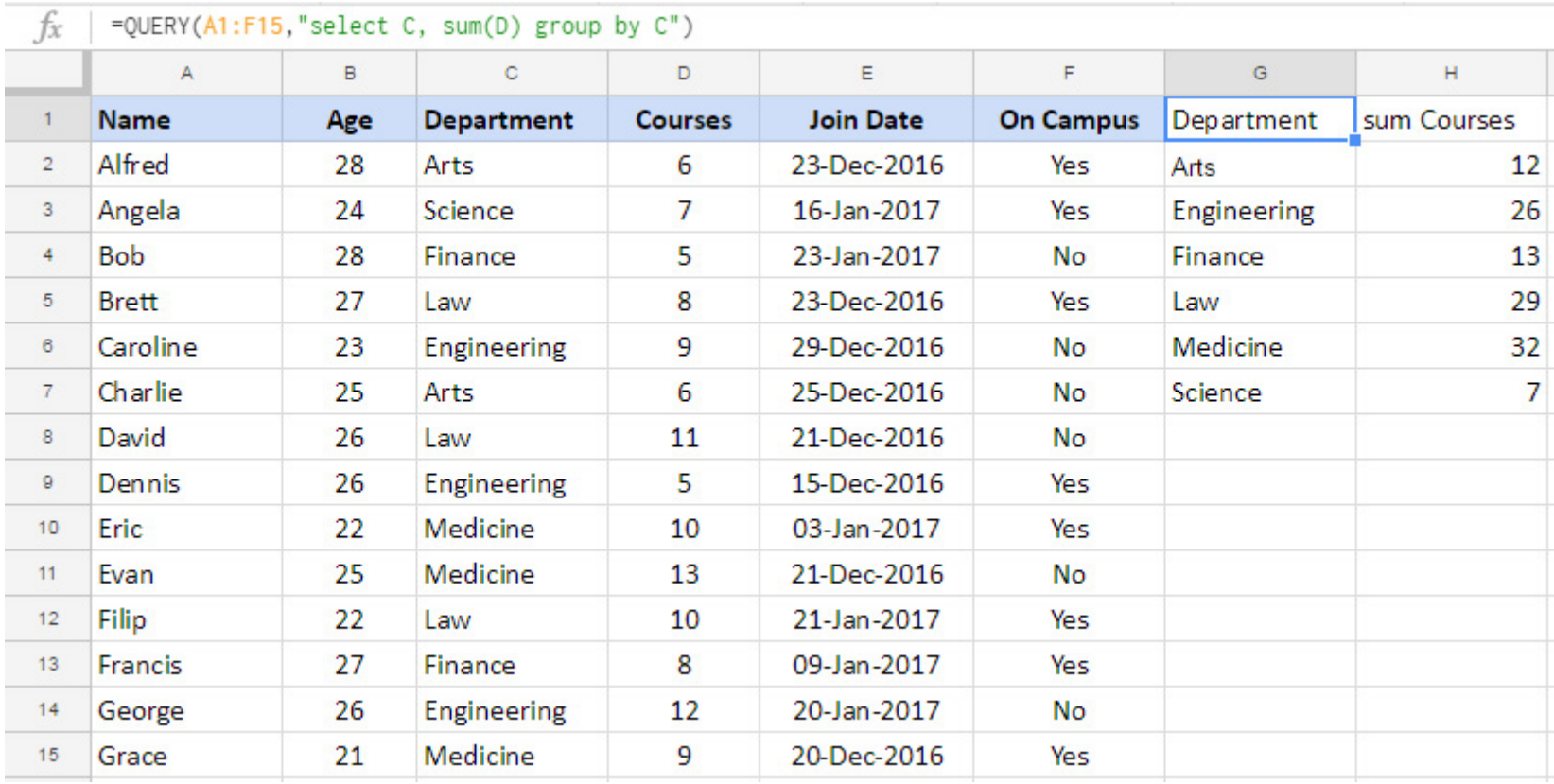
What if we need to reference the date from a cell? No problem there! We will get around with the help of concatenating operators and a text function. Therefore, in the example below, we will get the names and join dates of the students whose joined after 1-Jan-2017.

```
=QUERY(A1:F15,"select A, E where E > date '"& TEXT(I1,"yyyy-mm-dd")&"'")
```

	A	B	C	D	E	F	G	H	I
1	**Name**	**Age**	**Department**	**Courses**	**Join Date**	**On Campus**	Name	Join Date	**1-Jan-2017**
2	Alfred	28	Arts	6	23-Dec-2016	Yes	Angela	1/16/2017	
3	Angela	24	Science	7	16-Jan-2017	Yes	Bob	1/23/2017	
4	Bob	28	Finance	5	23-Jan-2017	No	Eric	1/3/2017	
5	Brett	27	Law	8	23-Dec-2016	Yes	Filip	1/21/2017	
6	Caroline	23	Engineering	9	29-Dec-2016	No	Francis	1/9/2017	
7	Charlie	25	Arts	6	25-Dec-2016	No	George	1/20/2017	
8	David	26	Law	11	21-Dec-2016	No			
9	Dennis	26	Engineering	5	15-Dec-2016	Yes			
10	Eric	22	Medicine	10	03-Jan-2017	Yes			
11	Evan	25	Medicine	13	21-Dec-2016	No			
12	Filip	22	Law	10	21-Jan-2017	Yes			
13	Francis	27	Finance	8	09-Jan-2017	Yes			
14	George	26	Engineering	12	20-Jan-2017	No			
15	Grace	21	Medicine	9	20-Dec-2016	Yes			

Example # 6:

Is there a way to list out all the departments and the display number of courses taken from the respective department? Yes, there is! And, we might as well understand the power and versatility that the QUERY formula offers.



`=QUERY(A1:F15,"select C, sum(D) group by C")`

	A	B	C	D	E	F	G	H
1	**Name**	**Age**	**Department**	**Courses**	**Join Date**	**On Campus**	Department	sum Courses
2	Alfred	28	Arts	6	23-Dec-2016	Yes	Arts	12
3	Angela	24	Science	7	16-Jan-2017	Yes	Engineering	26
4	Bob	28	Finance	5	23-Jan-2017	No	Finance	13
5	Brett	27	Law	8	23-Dec-2016	Yes	Law	29
6	Caroline	23	Engineering	9	29-Dec-2016	No	Medicine	32
7	Charlie	25	Arts	6	25-Dec-2016	No	Science	7
8	David	26	Law	11	21-Dec-2016	No		
9	Dennis	26	Engineering	5	15-Dec-2016	Yes		
10	Eric	22	Medicine	10	03-Jan-2017	Yes		
11	Evan	25	Medicine	13	21-Dec-2016	No		
12	Filip	22	Law	10	21-Jan-2017	Yes		
13	Francis	27	Finance	8	09-Jan-2017	Yes		
14	George	26	Engineering	12	20-Jan-2017	No		
15	Grace	21	Medicine	9	20-Dec-2016	Yes		

You will notice that the QUERY formula returned the second column with the header “sum Courses”. Honestly, it is a bit awkward to have that for a header. But, we can fix that and rename it. Not only that, we will also use the second column (now renamed to ‘Courses Taken’) to sort in ascending order. Here is how we do it.

```
=QUERY(A1:F15,"select C, sum(D) group by C order by sum(D) label sum(D) 'Courses Taken'")
```

	A	B	C	D	E	F	G	H
1	**Name**	**Age**	**Department**	**Courses**	**Join Date**	**On Campus**	Department	Courses Taken
2	Alfred	28	Arts	6	23-Dec-2016	Yes	Science	7
3	Angela	24	Science	7	16-Jan-2017	Yes	Arts	12
4	Bob	28	Finance	5	23-Jan-2017	No	Finance	13
5	Brett	27	Law	8	23-Dec-2016	Yes	Engineering	26
6	Caroline	23	Engineering	9	29-Dec-2016	No	Law	29
7	Charlie	25	Arts	6	25-Dec-2016	No	Medicine	32
8	David	26	Law	11	21-Dec-2016	No		
9	Dennis	26	Engineering	5	15-Dec-2016	Yes		
10	Eric	22	Medicine	10	03-Jan-2017	Yes		
11	Evan	25	Medicine	13	21-Dec-2016	No		
12	Filip	22	Law	10	21-Jan-2017	Yes		
13	Francis	27	Finance	8	09-Jan-2017	Yes		
14	George	26	Engineering	12	20-Jan-2017	No		
15	Grace	21	Medicine	9	20-Dec-2016	Yes		

Example # 7:

Can we display the number of instances of each of the departments? Of course, we can! The QUERY formula got us covered here as well.

```
=QUERY(A1:F15,"select C, count(D) group by C label count(D) '# Instances'")
```

	A	B	C	D	E	F	G	H
1	**Name**	**Age**	**Department**	**Courses**	**Join Date**	**On Campus**	Department	# Instances
2	Alfred	28	Arts	6	23-Dec-2016	Yes	Arts	2
3	Angela	24	Science	7	16-Jan-2017	Yes	Engineering	3
4	Bob	28	Finance	5	23-Jan-2017	No	Finance	2
5	Brett	27	Law	8	23-Dec-2016	Yes	Law	3
6	Caroline	23	Engineering	9	29-Dec-2016	No	Medicine	3
7	Charlie	25	Arts	6	25-Dec-2016	No	Science	1
8	David	26	Law	11	21-Dec-2016	No		
9	Dennis	26	Engineering	5	15-Dec-2016	Yes		
10	Eric	22	Medicine	10	03-Jan-2017	Yes		
11	Evan	25	Medicine	13	21-Dec-2016	No		
12	Filip	22	Law	10	21-Jan-2017	Yes		
13	Francis	27	Finance	8	09-Jan-2017	Yes		
14	George	26	Engineering	12	20-Jan-2017	No		
15	Grace	21	Medicine	9	20-Dec-2016	Yes		

Example # 8:

Consequently, we will now experiment with the third parameter. While this is an optional input, it might come handy when we come across headers that span across multiple rows. In such cases, this parameter helps us combine the headers in one single row, as shown below.

```
=QUERY(A1:F16,"select A, B, C, D where F <> 'No'",2)
```

	A	B	C	D	E	F	G	H	I	J
1	**First**	**Given**	**University**	**#**	**Join**	**Residence**	First Name	Given Age	University Department	# Courses
2	**Name**	**Age**	**Department**	**Courses**	**Date**	**On Campus**	Alfred	28	Arts	6
3	Alfred	28	Arts	6	23-Dec-2016	Yes	Angela	24	Science	7
4	Angela	24	Science	7	16-Jan-2017	Yes	Brett	27	Law	8
5	Bob	28	Finance	5	23-Jan-2017	No	Dennis	26	Engineering	5
6	Brett	27	Law	8	23-Dec-2016	Yes	Eric	22	Medicine	10
7	Caroline	23	Engineering	9	29-Dec-2016	No	Filip	22	Law	10
8	Charlie	25	Arts	6	25-Dec-2016	No	Francis	27	Finance	8
9	David	26	Law	11	21-Dec-2016	No	Grace	21	Medicine	9
10	Dennis	26	Engineering	5	15-Dec-2016	Yes				
11	Eric	22	Medicine	10	03-Jan-2017	Yes				
12	Evan	25	Medicine	13	21-Dec-2016	No				
13	Filip	22	Law	10	21-Jan-2017	Yes				
14	Francis	27	Finance	8	09-Jan-2017	Yes				
15	George	26	Engineering	12	20-Jan-2017	No				
16	Grace	21	Medicine	9	20-Dec-2016	Yes				

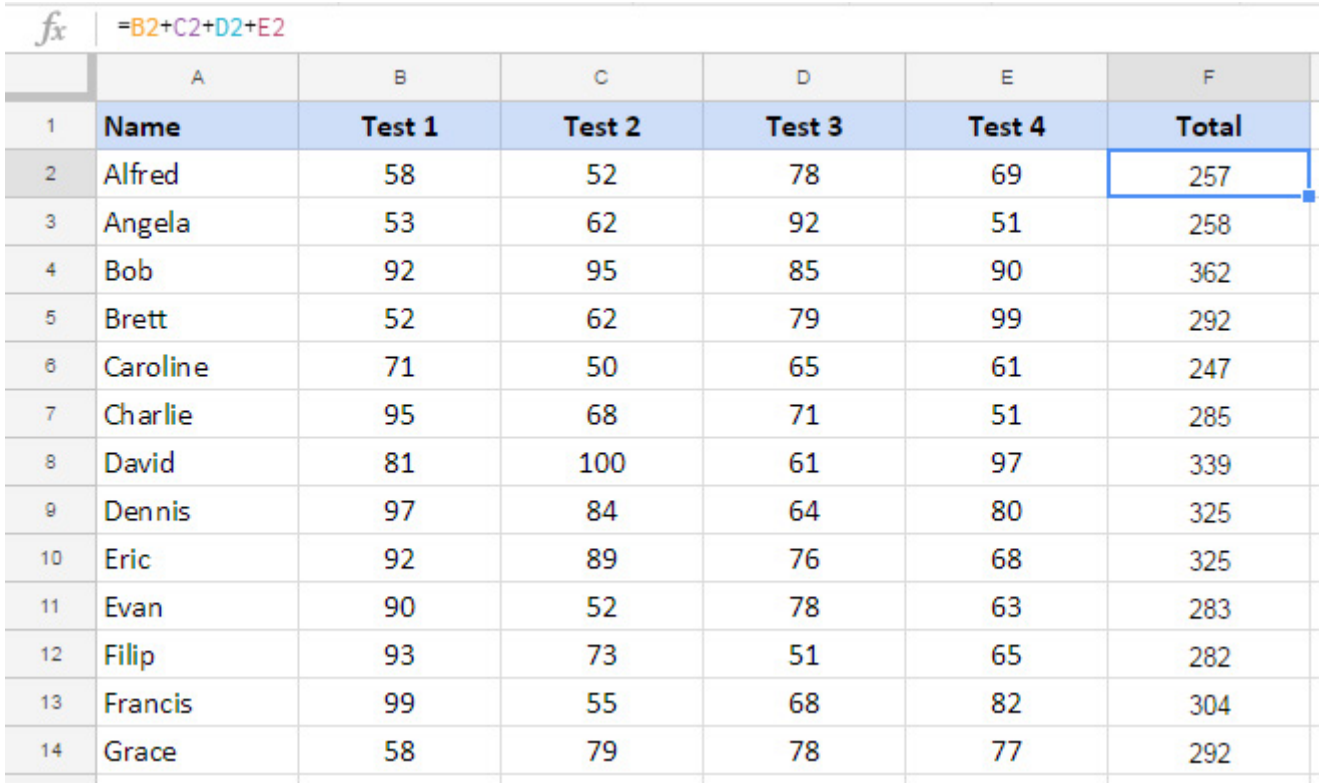
Without a doubt, this is one of the complex formulas to master in Google Sheets. Get a good hang on this one and it could serve you as one of the most potent tools. We encourage you to further explore the QUERY formula here.

Displaying the data

ARRAYFORMULA

To understand the importance and utility of the ARRAYFORMULA in Google Sheets, let us first go through a fundamental concept we already know.

Many a time, We tend to use structurally similar formulas across the length of a column in a data range. In doing so, we take the advantage of 'relative referencing'. Meaning, the Google Sheets automatically adjusts the formula if we copy and paste it in the subsequent rows. Let us consider a basic example. The data set is a list of students along with their test scores. We should now calculate the totals in column F.



	A	B	C	D	E	F
1	**Name**	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Total**
2	Alfred	58	52	78	69	257
3	Angela	53	62	92	51	258
4	Bob	92	95	85	90	362
5	Brett	52	62	79	99	292
6	Caroline	71	50	65	61	247
7	Charlie	95	68	71	51	285
8	David	81	100	61	97	339
9	Dennis	97	84	64	80	325
10	Eric	92	89	76	68	325
11	Evan	90	52	78	63	283
12	Filip	93	73	51	65	282
13	Francis	99	55	68	82	304
14	Grace	58	79	78	77	292

The total score, for Alfred, is calculated using a simple formula "=B2+C2+D2+F2". We copy and paste this formula in the cells below to repeat such calculations for all the students. So, for Angela, the formula automatically becomes "=B3+C3+D3+F3". For Bob it is "=B4+C4+D4+F4", and so on and so forth. Though this approach is seemingly convenient, there are inherent problems to doing this.

- If in the above case the data set is huge, we end up with a lot of formulas. This could bloat the spreadsheet and make it a tad slower.
- If we need to make any changes to the formula, this has to be repeated across all the formulas.
- What if a new student by name Charlotte joins the class? When we include a new row for her below row # 7, the formula isn't copied automatically. In a sense, this approach is not dynamic enough.
-

The ARRAYFORMULA solves all the above mentioned problems.

- As opposed to a bunch of similar formulas that individually calculate values, we can have one single ARRAYFORMULA that processes the data in a batch.
- Since this is a single formula, we can make changes in just one place and the effect takes place across the data range.
- We can induce the dynamism that was missing with a bunch of individual formulas, in that the ARRAYFORMULA got us covered even if a new row is introduced that needs similar formula calculations.

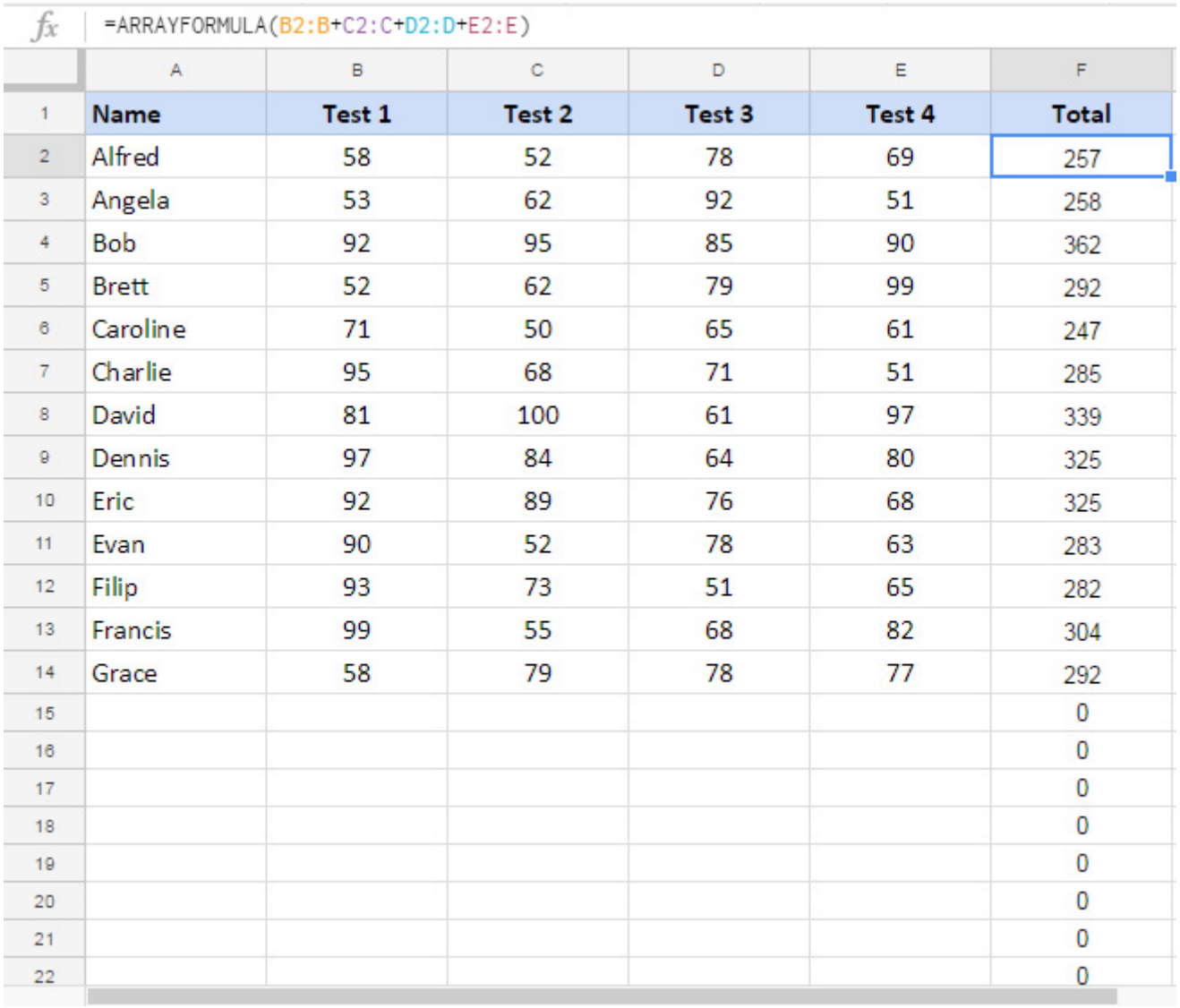
Syntax

`ARRAYFORMULA(array_formula)`

- `array_formula` – this parameter can either be
  - a range,
  - a mathematical expression using one cell range or multiple ranges of the same size, or
  - a function that returns a result greater than one cell.

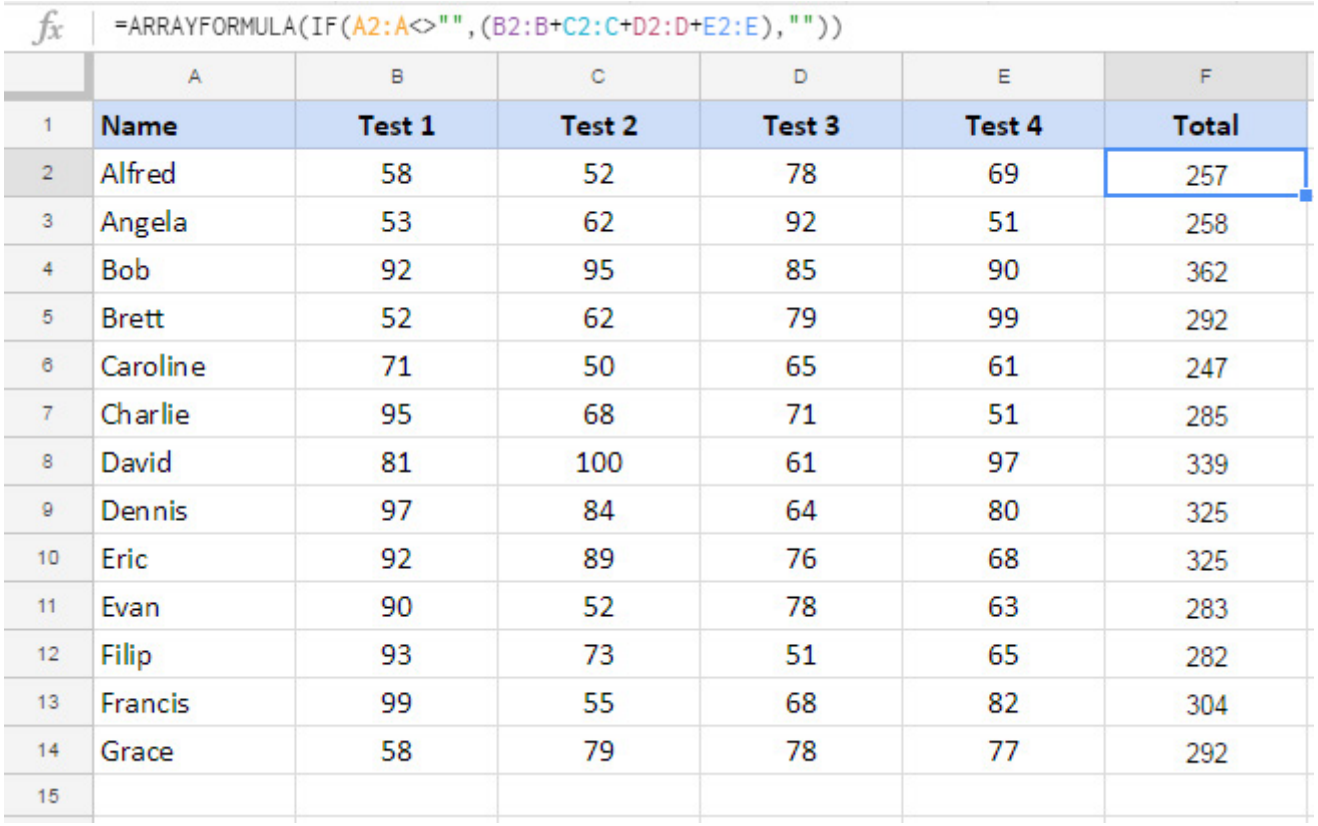
Usage: ARRAYFORMULA

Building on the previous example, in order to calculate the totals, we can use the ARRAYFORMULA as shown below. Notice that, unlike before where we've added individual cells, we are now adding the ranges. And we key this in the very first total cell, F2. All the following Total cells should be clear of any values or formulas, so that ARRAYFORMULA can show us the results without any errors.



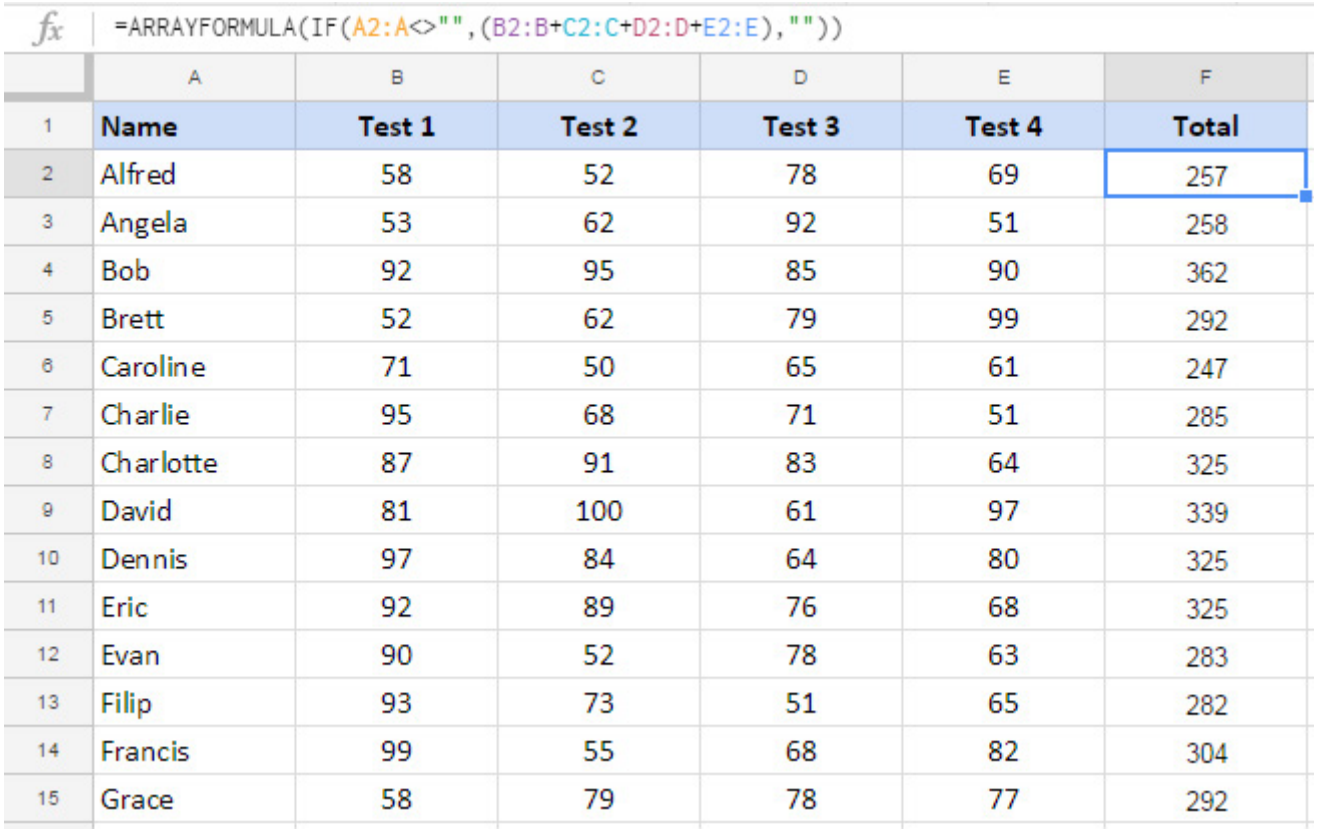
	A	B	C	D	E	F
1	**Name**	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Total**
2	Alfred	58	52	78	69	257
3	Angela	53	62	92	51	258
4	Bob	92	95	85	90	362
5	Brett	52	62	79	99	292
6	Caroline	71	50	65	61	247
7	Charlie	95	68	71	51	285
8	David	81	100	61	97	339
9	Dennis	97	84	64	80	325
10	Eric	92	89	76	68	325
11	Evan	90	52	78	63	283
12	Filip	93	73	51	65	282
13	Francis	99	55	68	82	304
14	Grace	58	79	78	77	292
15						0
16						0
17						0
18						0
19						0
20						0
21						0
22						0

To quickly fix the zeroes at the end, we change the formula in just one place (i.e. cell F2). Yet, it affects the calculations across the range, as shown below. Notice that even the expression used in the condition within the IF formula is a range A2:A.



	A	B	C	D	E	F
1	**Name**	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Total**
2	Alfred	58	52	78	69	257
3	Angela	53	62	92	51	258
4	Bob	92	95	85	90	362
5	Brett	52	62	79	99	292
6	Caroline	71	50	65	61	247
7	Charlie	95	68	71	51	285
8	David	81	100	61	97	339
9	Dennis	97	84	64	80	325
10	Eric	92	89	76	68	325
11	Evan	90	52	78	63	283
12	Filip	93	73	51	65	282
13	Francis	99	55	68	82	304
14	Grace	58	79	78	77	292
15						

Now, We'll include a row for Charlotte below row # 7. Let's find out whether the ARRAYFORMULA automatically calculates the Total for her.



=ARRAYFORMULA(IF(A2:A<>"",(B2:B+C2:C+D2:D+E2:E),""))

	A	B	C	D	E	F
1	**Name**	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Total**
2	Alfred	58	52	78	69	257
3	Angela	53	62	92	51	258
4	Bob	92	95	85	90	362
5	Brett	52	62	79	99	292
6	Caroline	71	50	65	61	247
7	Charlie	95	68	71	51	285
8	Charlotte	87	91	83	64	325
9	David	81	100	61	97	339
10	Dennis	97	84	64	80	325
11	Eric	92	89	76	68	325
12	Evan	90	52	78	63	283
13	Filip	93	73	51	65	282
14	Francis	99	55	68	82	304
15	Grace	58	79	78	77	292

Well, not surprisingly, it does. For more information on ARRAYFORMULA, please check out this link.

TRANSPOSE

In Google Sheets, if we ever need to flip (or transpose) the columns and rows of an array or a data range, TRANSPOSE formula is the one to go with.

Syntax

`TRANSPOSE(array _ or _ range)`

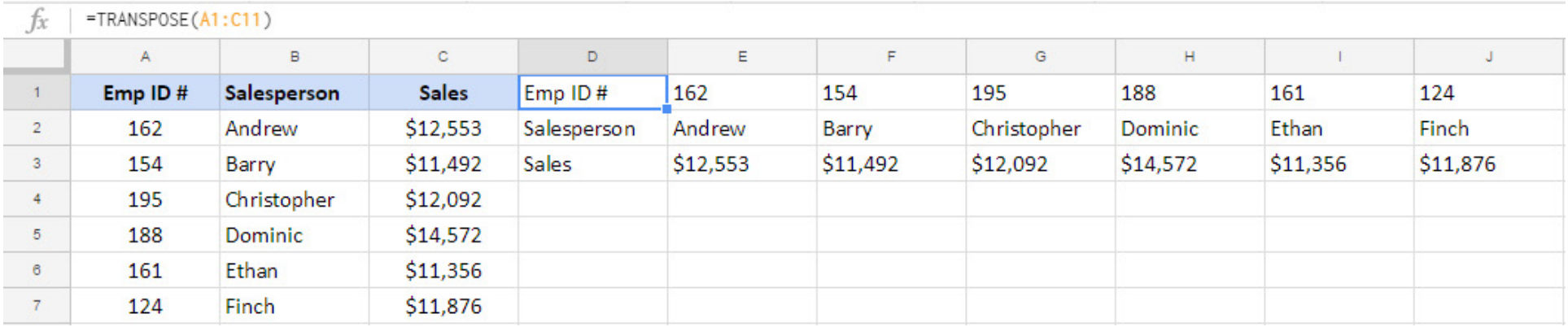
- `array _ or _ range` – the array or the address reference to the range whose columns and rows we need to be swapped or transposed.

Usage: TRANSPOSE Formula

This is one of the simplest formulas we encounter while using the Google Sheets spreadsheet application. Let us explore its working with help of a few examples, and we will use the following sample data.

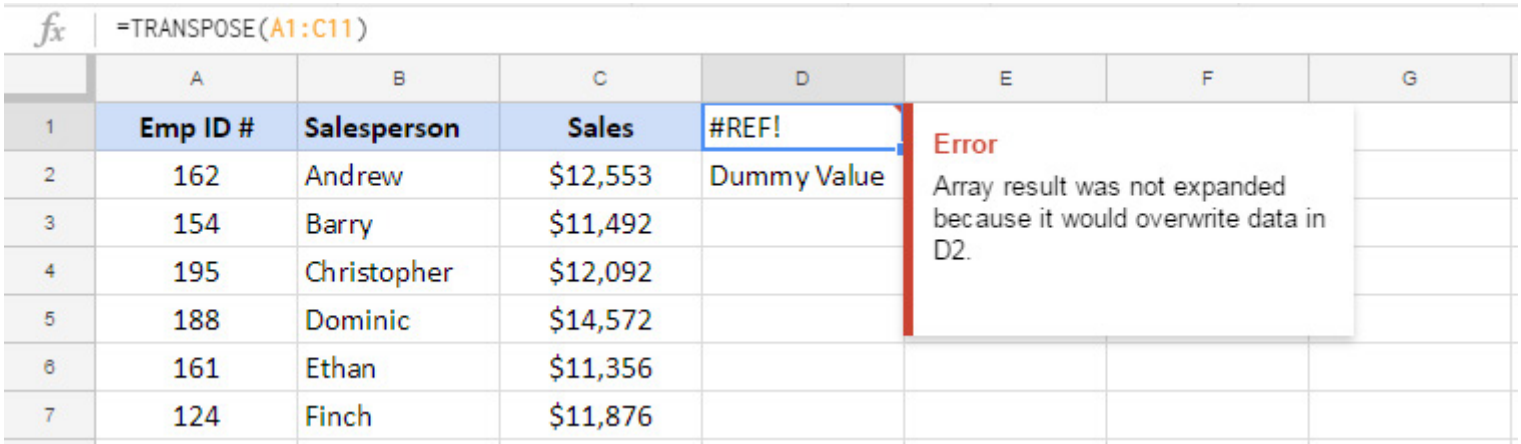
	A	B	C
1	**Emp ID #**	**Salesperson**	**Sales**
2	162	Andrew	$12,553
3	154	Barry	$11,492
4	195	Christopher	$12,092
5	188	Dominic	$14,572
6	161	Ethan	$11,356
7	124	Finch	$11,876

The test data spans from cells A1 through to C7. So, we will key in the formula in the cell D1 as illustrated in the snapshot below, and try to understand its behavior.



	A	B	C	D	E	F	G	H	I	J
1	**Emp ID #**	**Salesperson**	**Sales**	Emp ID #	162	154	195	188	161	124
2	162	Andrew	$12,553	Salesperson	Andrew	Barry	Christopher	Dominic	Ethan	Finch
3	154	Barry	$11,492	Sales	$12,553	$11,492	$12,092	$14,572	$11,356	$11,876
4	195	Christopher	$12,092							
5	188	Dominic	$14,572							
6	161	Ethan	$11,356							
7	124	Finch	$11,876							

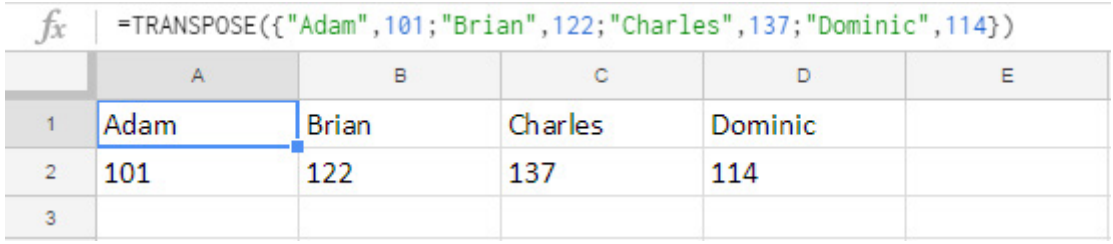
We'll see the returned data flows from D1 towards right and further down. The number of columns and rows returned by this formula is same as the rows and columns of the input `array _ or _ range` respectively. So, it is very important that we keep the cells, where we expect the data to flow, clear of any values. Otherwise, the formula returns #REF! error. This is demonstrated in the screenshot below.



	A	B	C	D	E	F	G
1	**Emp ID #**	**Salesperson**	**Sales**	#REF!			
2	162	Andrew	$12,553	Dummy Value			
3	154	Barry	$11,492				
4	195	Christopher	$12,092				
5	188	Dominic	$14,572				
6	161	Ethan	$11,356				
7	124	Finch	$11,876				

Going back to the second image, we will notice that the rows in the source range changes to columns and vice versa.

We have witnessed the working of the TRANSPOSE formula when we keyed in the address reference as its input parameter. Let us now try to input an array instead, as illustrated in the image below.



The semi-colons separate each individual row values in the input range. Whereas commas separate each individual column values. Such an input is now transposed as shown in the image above. Just like in the previous case, the expected real estate for this formula should be free from any other values. Otherwise, it returns an #REF! error.

INDIRECT

The INDIRECT formula in Google Sheets takes in the cell address in the form of text and returns a cell reference. It works the opposite as that of the ADDRESS formula, that returns an address in the text form.

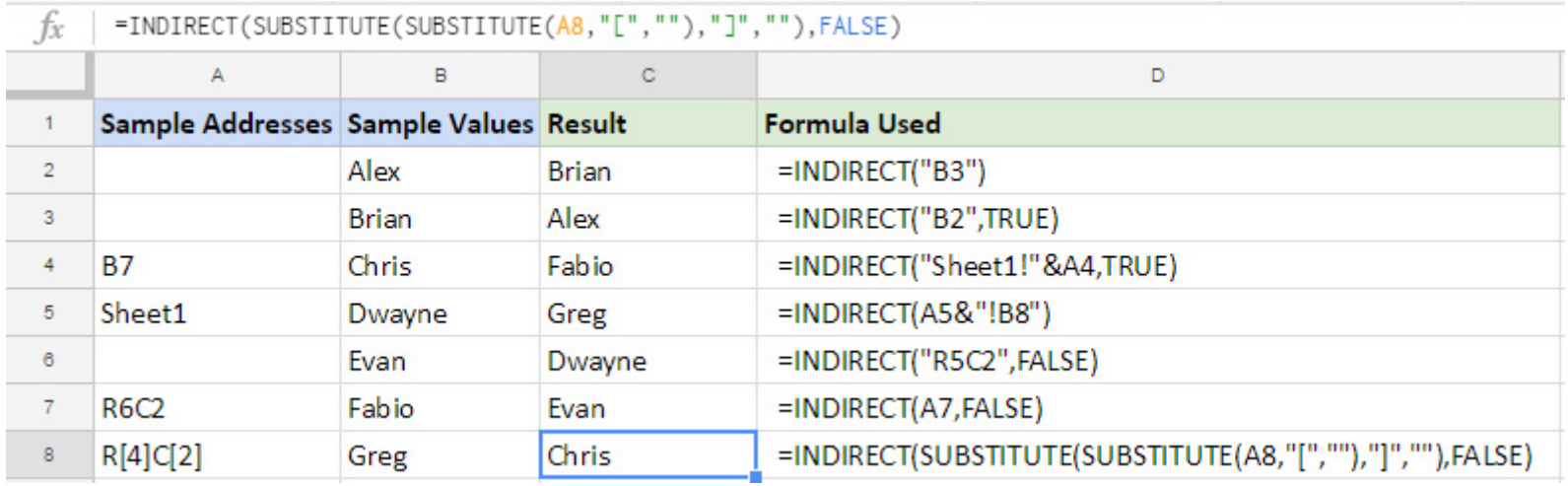
Syntax

`INDIRECT(cell_reference_as_string, [is_A1_notation])`

- `cell_reference_as_string` – is the text form of a cell address. Notice that it is not the cell reference itself.
- `is_a1_notation` – there are two kinds of representations for a cell address. One in the form of A1. And the other way of representing the same cell is R1C1 (just a short representation for row # 1 and column # 1). TRUE is the value by default, and we get A1 notation in return. Otherwise, if we need R1C1 notation, we need to specify FALSE.

Usage: INDIRECT formula

We understand the concept better with help of examples. So, here are a few combinations of the formula.



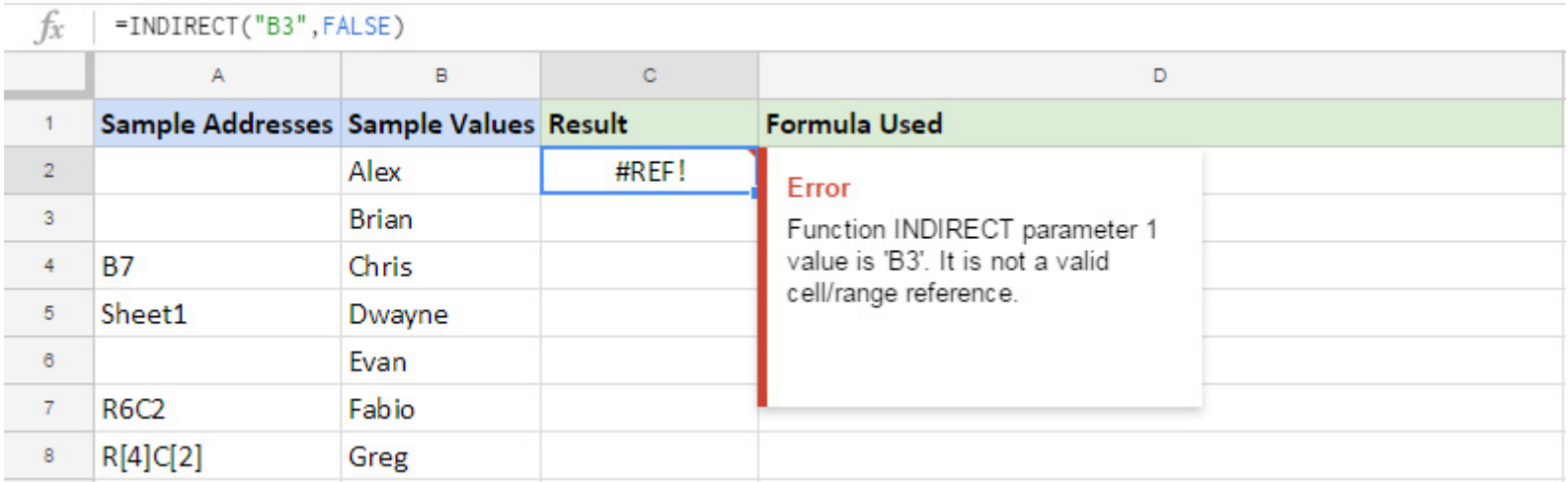
	A	B	C	D
1	**Sample Addresses**	**Sample Values**	**Result**	**Formula Used**
2		Alex	Brian	=INDIRECT("B3")
3		Brian	Alex	=INDIRECT("B2",TRUE)
4	B7	Chris	Fabio	=INDIRECT("Sheet1!"&A4,TRUE)
5	Sheet1	Dwayne	Greg	=INDIRECT(A5&"!B8")
6		Evan	Dwayne	=INDIRECT("R5C2",FALSE)
7	R6C2	Fabio	Evan	=INDIRECT(A7,FALSE)
8	R[4]C[2]	Greg	Chris	=INDIRECT(SUBSTITUTE(SUBSTITUTE(A8,"[",""),"]",""),FALSE)

You'll notice that the first parameter can be a direct string enclosed in double quotes (rows 2 and 3), a cell reference that holds the address string (row # 7), or even a concatenated string (rows 4 and 5).

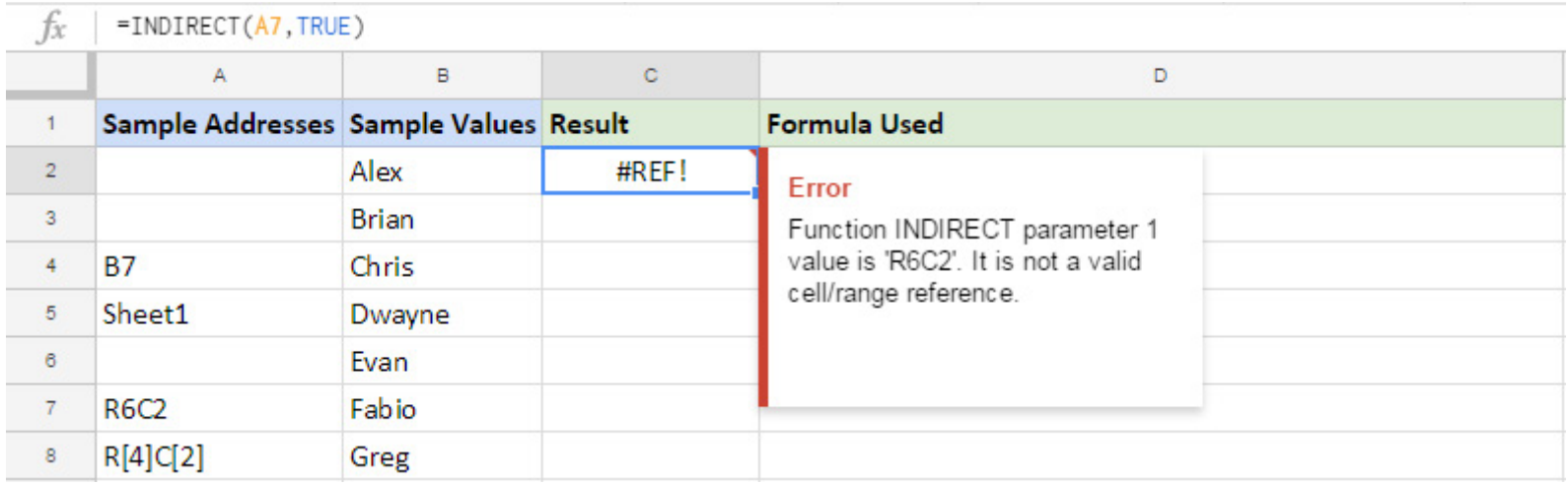
Please note that the INDIRECT formula cannot handle R[1]C[1] style of address notation. Instead it can process R1C1 style. So, in case we have to process the former representation, we can work around it like we did in the row # 8. We substituted the square brackets with empty strings using the SUBSTITUTE formula.

Will the INDIRECT formula result in errors? Yes, of course!

The following screenshot illustrates the outcome when we try to input A1 style address notation for `cell_reference_as_string` and indicate FALSE for the `is_A1_notation` parameter. Doing this will obviously leave the INDIRECT formula looking for R1C1 style notation, and it doesn't find one. Hence the error!

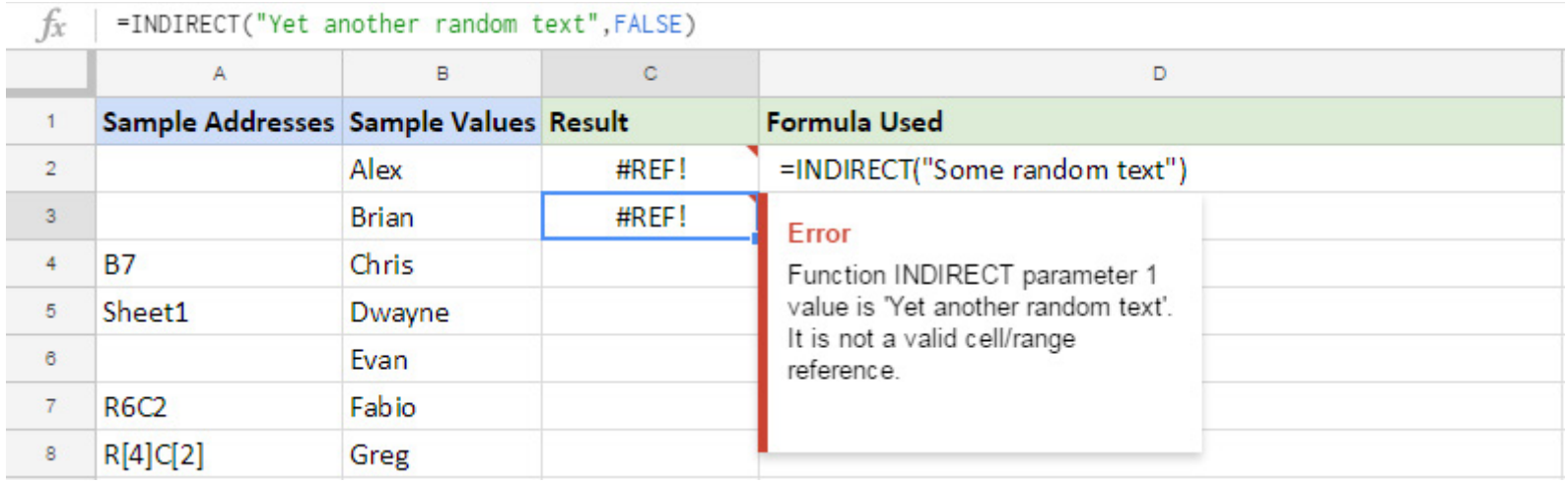


The vice versa is also true, as indicated in the snapshot below.



	A	B	C	D
1	**Sample Addresses**	**Sample Values**	**Result**	**Formula Used**
2		Alex	#REF!	
3		Brian		
4	B7	Chris		
5	Sheet1	Dwayne		
6		Evan		
7	R6C2	Fabio		
8	R[4]C[2]	Greg		

Now the worst case scenario. We try and input a text form that is neither A1 representation nor the R1C1 representation, but just some dummy text. And, of course it should throw up an error!



	A	B	C	D
1	**Sample Addresses**	**Sample Values**	**Result**	**Formula Used**
2		Alex	#REF!	=INDIRECT("Some random text")
3		Brian	#REF!	
4	B7	Chris		
5	Sheet1	Dwayne		
6		Evan		
7	R6C2	Fabio		
8	R[4]C[2]	Greg		

SPARKLINE

In Google Sheets, if you need to visually analyze data, but not by building bulky charts every time, SPARKLINE formula is the one to go with. It helps us come up with mini-sized in-cell charts, that quickly helps us visualize the trends.

Let's assume we have the following set of of data, on which we need to gauge the performance based on the scores of various students of a class.

	A	B	C	D	E
1	**Name**	**Test 1**	**Test 2**	**Test 3**	**Test 4**
2	Alfred	58	52	78	69
3	Angela	53	62	92	51
4	Bob	92	95	85	90
5	Brett	52	62	79	99
6	Caroline	71	50	65	61
7	Charlie	95	68	71	51
8	David	81	100	61	97
9	Dennis	97	84	64	80
10	Eric	92	89	76	68
11	Evan	90	52	78	63
12	Filip	93	73	51	65
13	Francis	99	55	68	82
14	Grace	58	79	78	77

At the least, it is not so easy on the eyes. And, when the data extends further, it can get even more difficult to grasp how the students are doing over time. To put this in perspective, let's just say we have over a hundred students and they have taken about 15 tests. Of course, we can't imagine establishing trends with bare eyes! And it isn't feasible to build a chart for each student.

What, then, would be the solution? It is the SPARKLINE formula! It certainly eases up a lot of things we just discussed.

Syntax

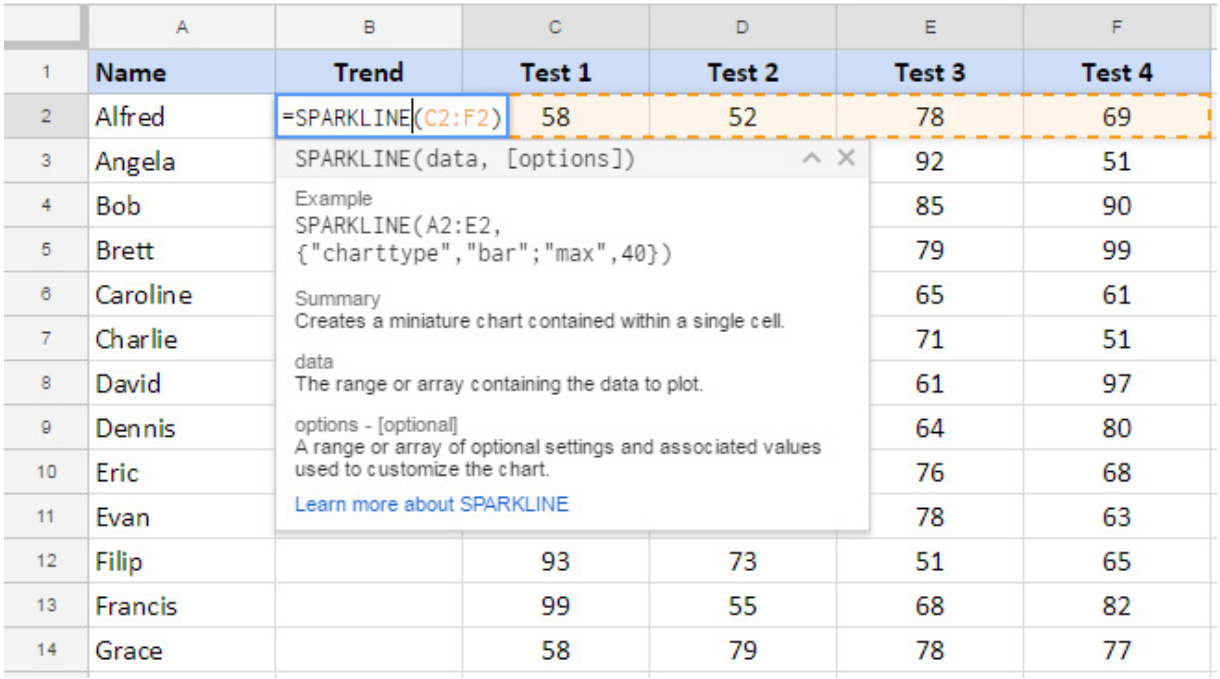
`SPARKLINE(data, [options])`

- `data` – this is the address reference to the range of data cells we want to plot using the SPARKLINE formula
- `options` – these are optional attributes that are used to customize the chart. There are many such options – you may want to take a look at them here. These can be input in two ways.
  - As an array of option key and option value pairs, following the data parameter.
  - As an address reference to a two column range, where the first column cells hold the option keys, and the second column cells hold the corresponding option values.

Usage: SPARKLINE Formula

Use Case # 1: SPARKLINE(data_range)

To accommodate the Sparklines, let us insert a blank column after the Name column, and We'll give it a header name 'Trend'. Now, in the cell B2, we'll type in the formula as shown in the snapshot below.



	A	B	C	D	E	F
1	Name	Trend	Test 1	Test 2	Test 3	Test 4
2	Alfred	=SPARKLINE(C2:F2)	58	52	78	69
3	Angela				92	51
4	Bob				85	90
5	Brett				79	99
6	Caroline				65	61
7	Charlie				71	51
8	David				61	97
9	Dennis				64	80
10	Eric				76	68
11	Evan				78	63
12	Filip		93	73	51	65
13	Francis		99	55	68	82
14	Grace		58	79	78	77

Let's hit the Enter key, and what we see in the cell B2 is the Sparkline. The miniature version of a chart, all of it, nicely embedded within just the area of a cell.

	A	B	C	D	E	F
1	**Name**	**Trend**	**Test 1**	**Test 2**	**Test 3**	**Test 4**
2	Alfred		58	52	78	69
3	Angela		53	62	92	51
4	Bob		92	95	85	90
5	Brett		52	62	79	99
6	Caroline		71	50	65	61
7	Charlie		95	68	71	51
8	David		81	100	61	97
9	Dennis		97	84	64	80
10	Eric		92	89	76	68
11	Evan		90	52	78	63
12	Filip		93	73	51	65
13	Francis		99	55	68	82
14	Grace		58	79	78	77

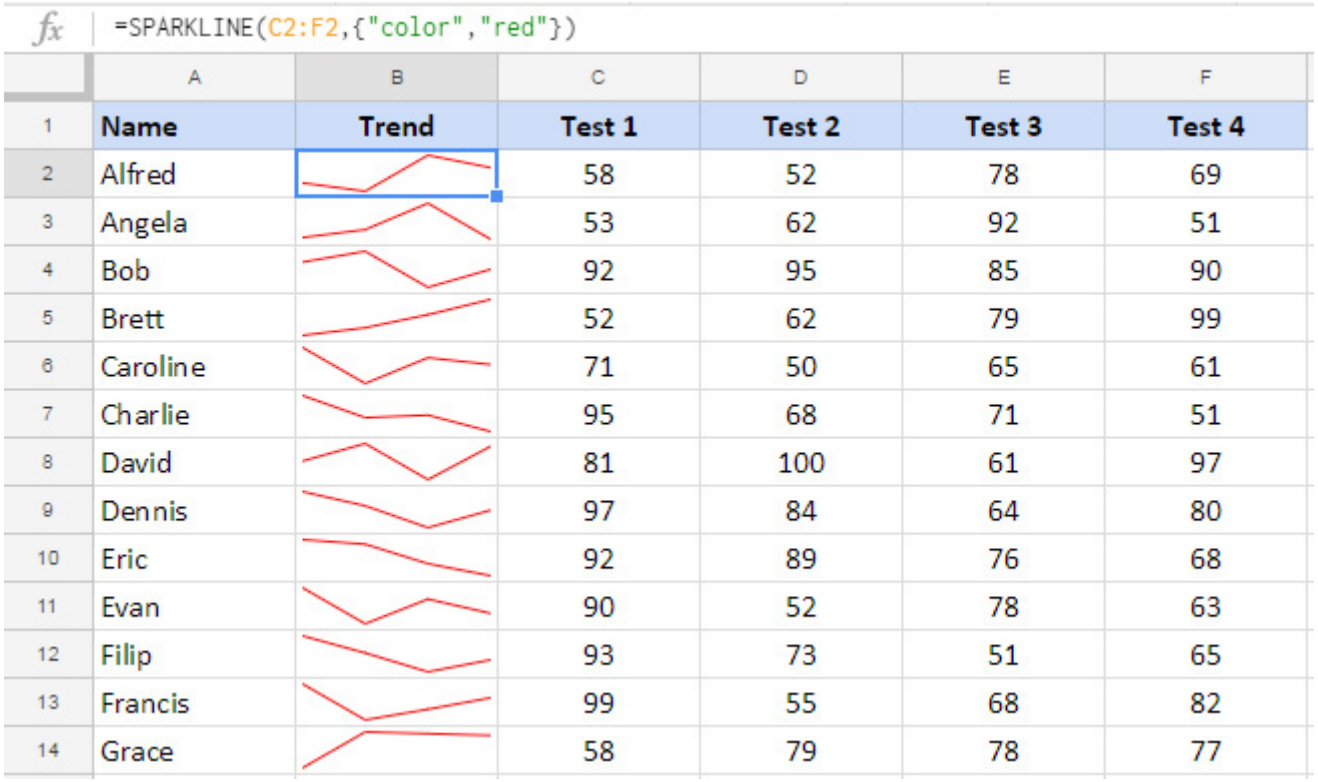
What we'll need to do now, is that we need such Sparklines for each and every student. It is as simple as dragging the formula all the way down (or just a simple copy paste will do). Here's how it looks like when we do that.

	A	B	C	D	E	F
1	**Name**	**Trend**	**Test 1**	**Test 2**	**Test 3**	**Test 4**
2	Alfred		58	52	78	69
3	Angela		53	62	92	51
4	Bob		92	95	85	90
5	Brett		52	62	79	99
6	Caroline		71	50	65	61
7	Charlie		95	68	71	51
8	David		81	100	61	97
9	Dennis		97	84	64	80
10	Eric		92	89	76	68
11	Evan		90	52	78	63
12	Filip		93	73	51	65
13	Francis		99	55	68	82
14	Grace		58	79	78	77

This, certainly, is easier to interpret the data, than manually scour the scores against each student. For instance, Brett has done great in consistently improving his scores, which is rather evident from the line that's going upwards. Whereas, Eric has done the opposite. His Sparkline shows that it's a downward trend.

Use Case # 2: SPARKLINE(data_range, color_option as key value pair)

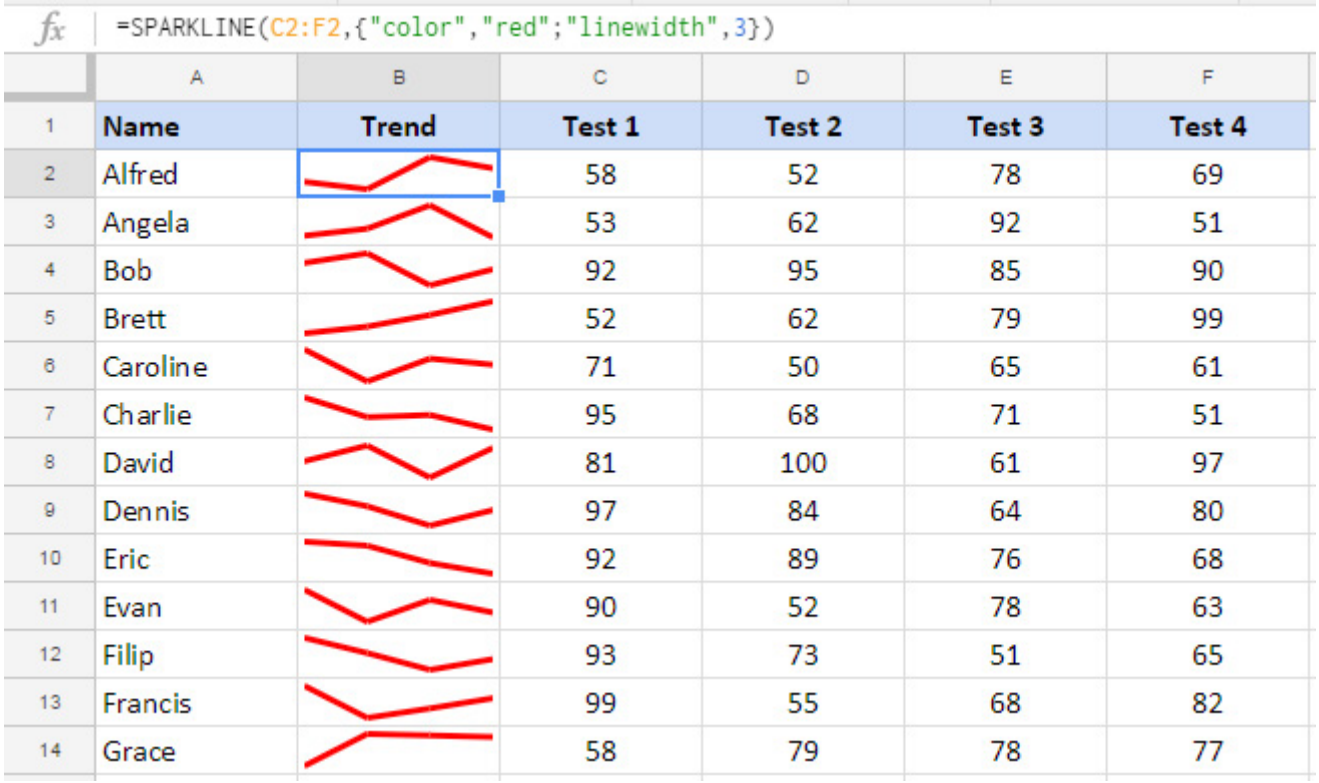
So far, so good. Now, let's try and change the way the lines are colored. Here's how we can do it. Notice that both the key ("color") and value ("red") are within double quotes, and that they are within curly braces that indicate an array of key value pairs.



	A	B	C	D	E	F
1	**Name**	**Trend**	**Test 1**	**Test 2**	**Test 3**	**Test 4**
2	Alfred		58	52	78	69
3	Angela		53	62	92	51
4	Bob		92	95	85	90
5	Brett		52	62	79	99
6	Caroline		71	50	65	61
7	Charlie		95	68	71	51
8	David		81	100	61	97
9	Dennis		97	84	64	80
10	Eric		92	89	76	68
11	Evan		90	52	78	63
12	Filip		93	73	51	65
13	Francis		99	55	68	82
14	Grace		58	79	78	77

Use Case # 3: SPARKLINE(data_range, [color_option and line_thickness as key value pairs])

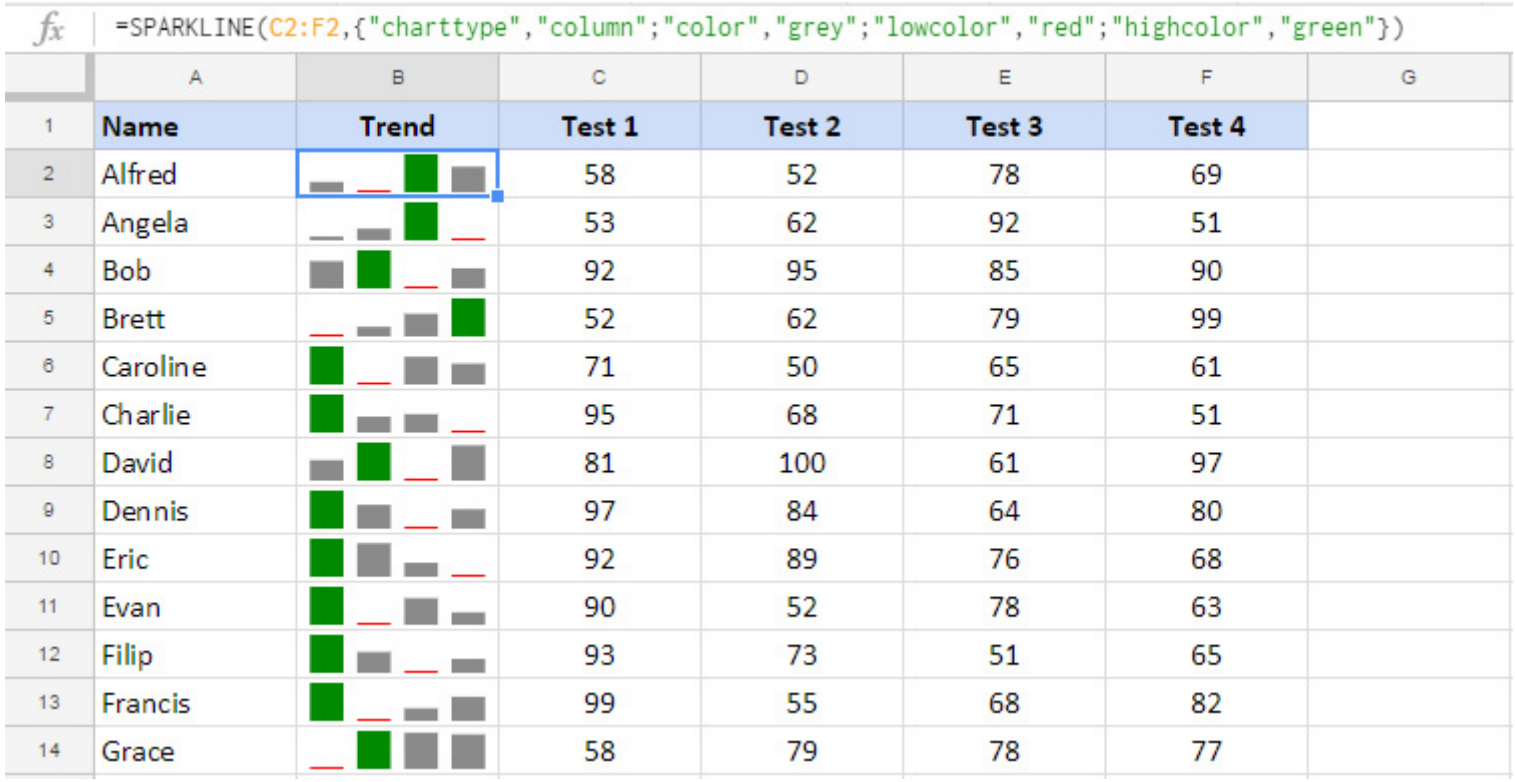
Probably the lines are too thin. Let's try and increase the width. To do that we need to include an additional option key-value pair, as shown below. Notice that the two key-value pairs are separated by a semi-colon.



	A	B	C	D	E	F
1	**Name**	**Trend**	**Test 1**	**Test 2**	**Test 3**	**Test 4**
2	Alfred		58	52	78	69
3	Angela		53	62	92	51
4	Bob		92	95	85	90
5	Brett		52	62	79	99
6	Caroline		71	50	65	61
7	Charlie		95	68	71	51
8	David		81	100	61	97
9	Dennis		97	84	64	80
10	Eric		92	89	76	68
11	Evan		90	52	78	63
12	Filip		93	73	51	65
13	Francis		99	55	68	82
14	Grace		58	79	78	77

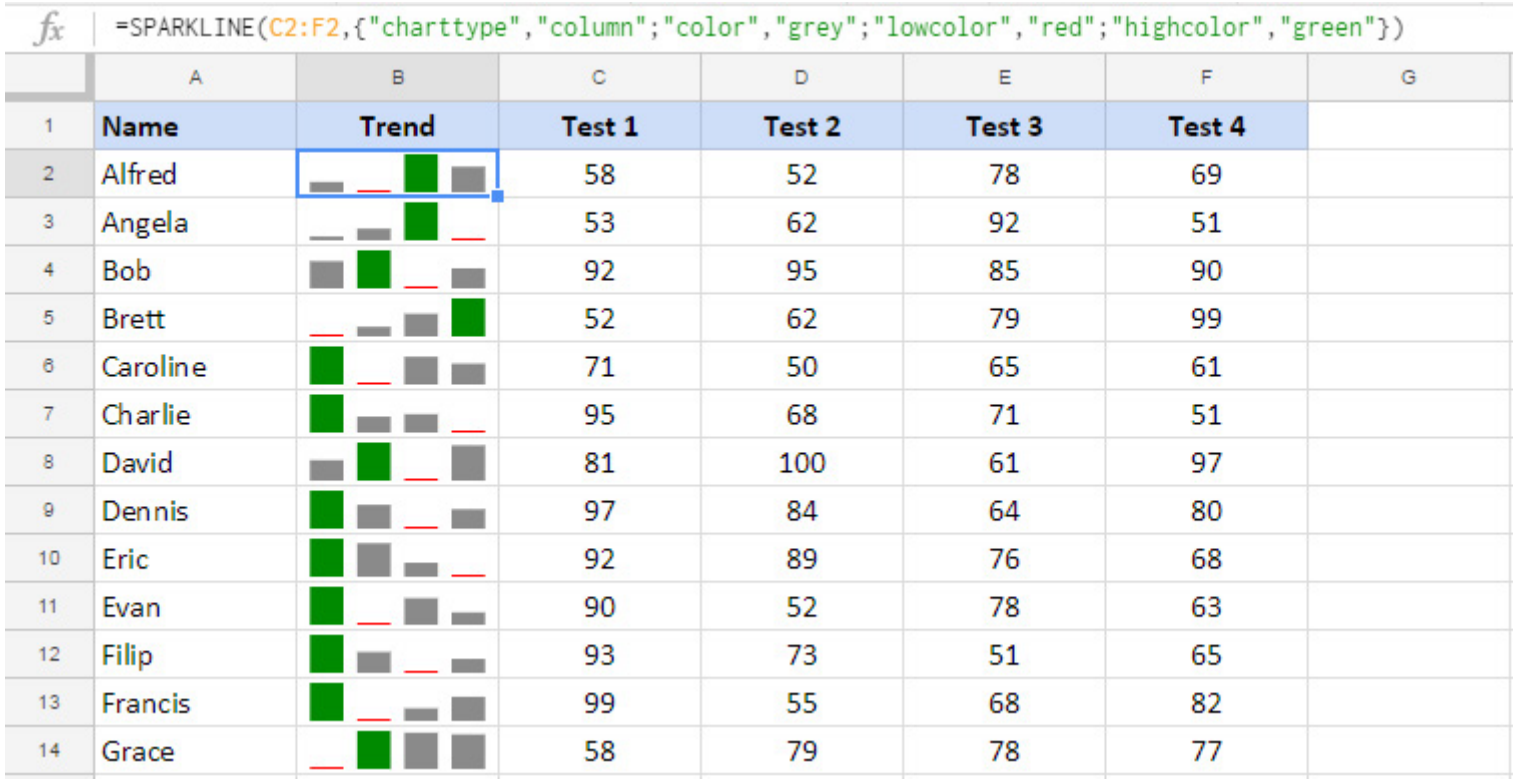
Use Case # 4: SPARKLINE(data_range,)

What if we do not want line graphs, after all? We can change it to other chart representations wherever appropriate. Let's try the column chart, shall we? Also, let's explore the other color options as shown in the snapshot below. Not the ideal looking Sparklines, I must admit. But I used this example to drive home the point that chart types can be changed too.



	A	B	C	D	E	F	G
1	**Name**	**Trend**	**Test 1**	**Test 2**	**Test 3**	**Test 4**	
2	Alfred		58	52	78	69	
3	Angela		53	62	92	51	
4	Bob		92	95	85	90	
5	Brett		52	62	79	99	
6	Caroline		71	50	65	61	
7	Charlie		95	68	71	51	
8	David		81	100	61	97	
9	Dennis		97	84	64	80	
10	Eric		92	89	76	68	
11	Evan		90	52	78	63	
12	Filip		93	73	51	65	
13	Francis		99	55	68	82	
14	Grace		58	79	78	77	

You can also use Hex codes to indicate the colors, like this.



	A	B	C	D	E	F	G
1	**Name**	**Trend**	**Test 1**	**Test 2**	**Test 3**	**Test 4**	
2	Alfred		58	52	78	69	
3	Angela		53	62	92	51	
4	Bob		92	95	85	90	
5	Brett		52	62	79	99	
6	Caroline		71	50	65	61	
7	Charlie		95	68	71	51	
8	David		81	100	61	97	
9	Dennis		97	84	64	80	
10	Eric		92	89	76	68	
11	Evan		90	52	78	63	
12	Filip		93	73	51	65	
13	Francis		99	55	68	82	
14	Grace		58	79	78	77	

Given the myriad of customization options provided by Google Sheets to SPARKLINE formula, I'd rather encourage you to try other variations of the SPARKLINE formula and find out your best suitable combination.


SHEETGO