Harness the Power of Array Formulas

Array formulas, which work on arrays of data rather than individual cells or ranges, allow users to make calculations that would otherwise be impossible with ordinary formulas. They can perform multiple discrete calculations in a single cell or return results to multiple cells. For example, an array formula can be used to round off totals or subtotals, a feat that is impossible to accomplish with an ordinary formula. Don't miss this opportunity to harness the power of array formulas to improve your analytical abilities in Excel.

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.

Introduction

Array formulas, which work on arrays of data rather than individual cells or ranges, allow users to make calculations that would otherwise be impossible with ordinary formulas. They can perform multiple discrete calculations in a single cell or return results to multiple cells. For example, an array formula can be used to round off totals or subtotals, a feat that is impossible to accomplish with an ordinary formula. Don't miss this opportunity to harness the power of array formulas to improve your analytical abilities in Excel.

Learning Objectives

Upon completing this course, participants should be able to:

- Explain array formulas and how they differ from ordinary formulas
- Identify an array formula from the Formula Bar
- Compose array formulas from the keyboard
- Describe common accounting situations where array formulas would be useful

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.

Array Formula Basics

An *array* is a collection or range of two or more items. An *array formula* is a formula that acts on an array, a range of cells, instead of individual cells, or a formula that delivers results to more than one cell. In a nutshell, array formulas can perform multiple discrete calculations in single cell and return the results to a single cell or return the results to multiple cells. Note that the results of an array formula can be imbedded within other formulas. Since array formulas work on a group of cells in a single operation, array formulas can perform calculations that are not possible with conventional formulas.

Excel has three types of arrays:

- A *reference array* is an area of a worksheet that contains more than one cell. It can be an ordinary cell range, a worksheet reference, or a defined name.
- A *result array* is an array of items created by an array operation. For example, the LINEST function can return an array of statistics than defines the results of a linear regression model.
- An *array constant* is an array of hard-coded values (constants). An array constant can be entered directly into a formula or can be saved and referenced as a defined name.

Array formulas can be created using ordinary functions that take individual cells as their values. This type of array formula must be entered using CTRL + SHIFT + ENTER. Excel also includes functions that operate on arrays natively, such as SUMPRODUCT, AGGREGATE, LOOKUP, INDEX and SUMIF(S), COUNTIF(S), and AVERAGEIF(S). These functions produce array formulas without entering CTRL + SHIFT + ENTER. Another set of functions produce result arrays. Among them are TRANSPOSE, TREND, FREQUENCY, LINEST, AND LOGEST.

Creating Simple Array Formulas

In this first example, three methods – conventional analysis using a helper column, an array formula using the SUM function, and an array formula using SUMPRODUCT – will be used to calculate the total wages paid to a project team for the week, as shown in **Figure 1**.

	A	B	C
1	Team Member	Hourly Rate	Hours
2	Brady	250.00	42.5
3	Wilson	225.00	35.0
4	Luck	175.00	27.0
5	Rogers	200.00	64.0

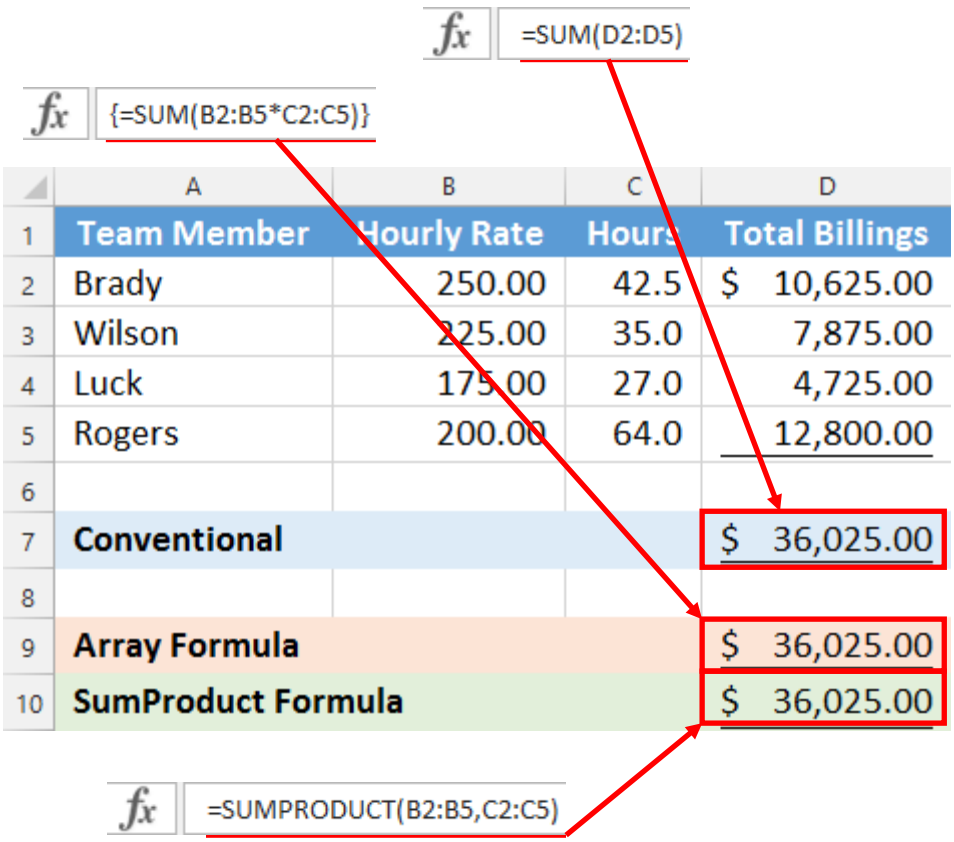
Figure 1 – Team Timesheet

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.

1. In column D, create individual formulas to calculate the wages for each team member. Copy the formula down and then sum the results in cell **D7**.

2. Build an ordinary array formula. Enter **=SUM(B2:B5*C2:C5)** in cell **D9**. Make sure to press **CTRL + SHIFT + ENTER** to enter the formula. Otherwise, the formula will evaluate to #VALUE!. Note the braces **{ }** surrounding the formula in the formula bar.

3. Use the SUMPRODUCT function to build the formula. Enter **=SUMPRODUCT(B2:B5,C2:C5)** into cell **D10** and press **ENTER**. CTRL + SHIFT + ENTER is *not* needed to enter the formula, nor do braces **{ }** surround the formula in the formula bar. The SUMPRODUCT function produces an array formula without all of the complexities of an ordinary array formula. It can handle up to 255 arrays of data.



	A	B	C	D
1	Team Member	Hourly Rate	Hours	Total Billings
2	Brady	250.00	42.5	$ 10,625.00
3	Wilson	225.00	35.0	7,875.00
4	Luck	175.00	27.0	4,725.00
5	Rogers	200.00	64.0	12,800.00
6				
7	**Conventional**			$ 36,025.00
8				
9	**Array Formula**			$ 36,025.00
10	**SumProduct Formula**			$ 36,025.00

Figure 2 – Conventional vs Array Formulas

Note that each of the three methods returned the same result, but the array formulas used a single cell to make the calculation. In the simple context presented, the most commonly used method with a helper column works as well as the array formulas, but in situations involving thousands of transactions, calculating the extended price or cost of goods just to calculate total revenue or total cost of goods sold would be cumbersome and time consuming. In those situations, a single cell array formula would be a better alternative. Further, using SUMPRODUCT in situations like these is a better solution because it

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.

calculates faster and is easier for average Excel users to understand than ordinary array formulas entered using CTRL + SHIFT + ENTER.

Before leaving this example, let's use an advanced technique to examine or troubleshoot array formulas. Position the cursor in cell **D9**, the cell containing the ordinary array formula. Press **F2** to edit the formula and then click ***number 1*** in the **Screen Tip** that appears to select the array reference. Press **F9** to see the underlying array, as shown in **Figure 3**. Do *not* press Enter without first pressing **CTRL + Z** to undo the change. Otherwise, the cell references will be replaced with the *array constant* so that the formula will no longer calculate by reference to the values in the cells, but by reference to the values inside the SUM function, thereby breaking the formula envisaged.

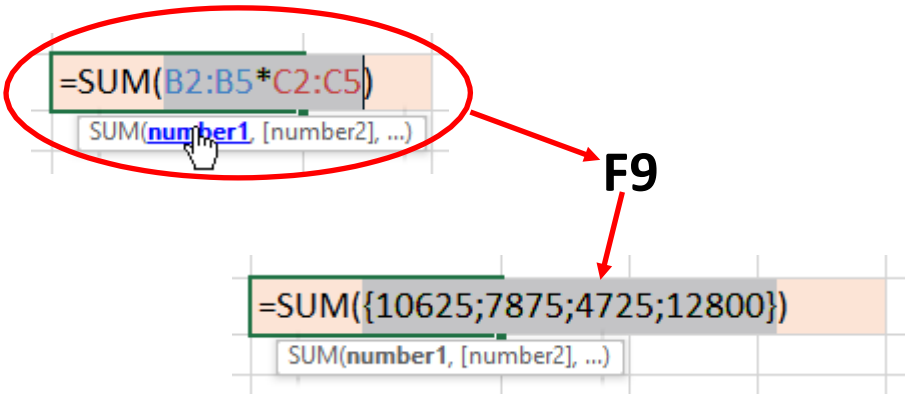


Figure 3 – Troubleshooting an Array Formula

Similarly, the formula built using SUMPRODUCT can be examined. Note that the individual items of the arrays can be examined rather than the results of the array calculations displayed in the ordinary array formula.

Making Calculations Based on Conditions

A more sophisticated example demonstrates how array formulas can be used to summarize data based on conditions, in this case expenses by job and by account. Three methods for making the calculation will be demonstrated – an array formula using the SUM function, and an array formula using SUMPRODUCT, and an array formula using SUMIFS.

1. Build an array formula using SUM. Enter **=SUM(($D$4:$D$7=$B14)*($E$4:$E$7=C$13)*($F$4:$F$7))** in cell **C14**. Note the mixed reference addressing used for cells containing the conditions so that the formula can be copied down and across. Make sure to press **CTRL + SHIFT + ENTER** to enter the formula. Otherwise, the formula will evaluate to #VALUE!. Note the braces **{ }** surrounding the formula in the formula bar.

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.

Figure 5 displays the expense list and the array formula that summarizes the expenses by job and by account. Examine the formula's operation by using the **F9** trick presented earlier.

The formula first tests to see if each Account in the range D4:D7 is equal to 8000 and then tests to see if each Job in the range E4:E7 is equal to 701. If the result of a test is **False**, the formula returns **0**. If the result is **True**, the formula returns **1**. The results of each test are multiplied together and then multiplied by the requisite amount from the range F4:F7 before being summed by the formula.

There is a single detail expense entry recorded for Account 8000 and Job 701, so the formula in cell C14 evaluates to $234.00. Note that when testing multiple criteria, the multiplication operator (*) represents an **AND** condition, and the addition operator (+) represents an **OR** condition.

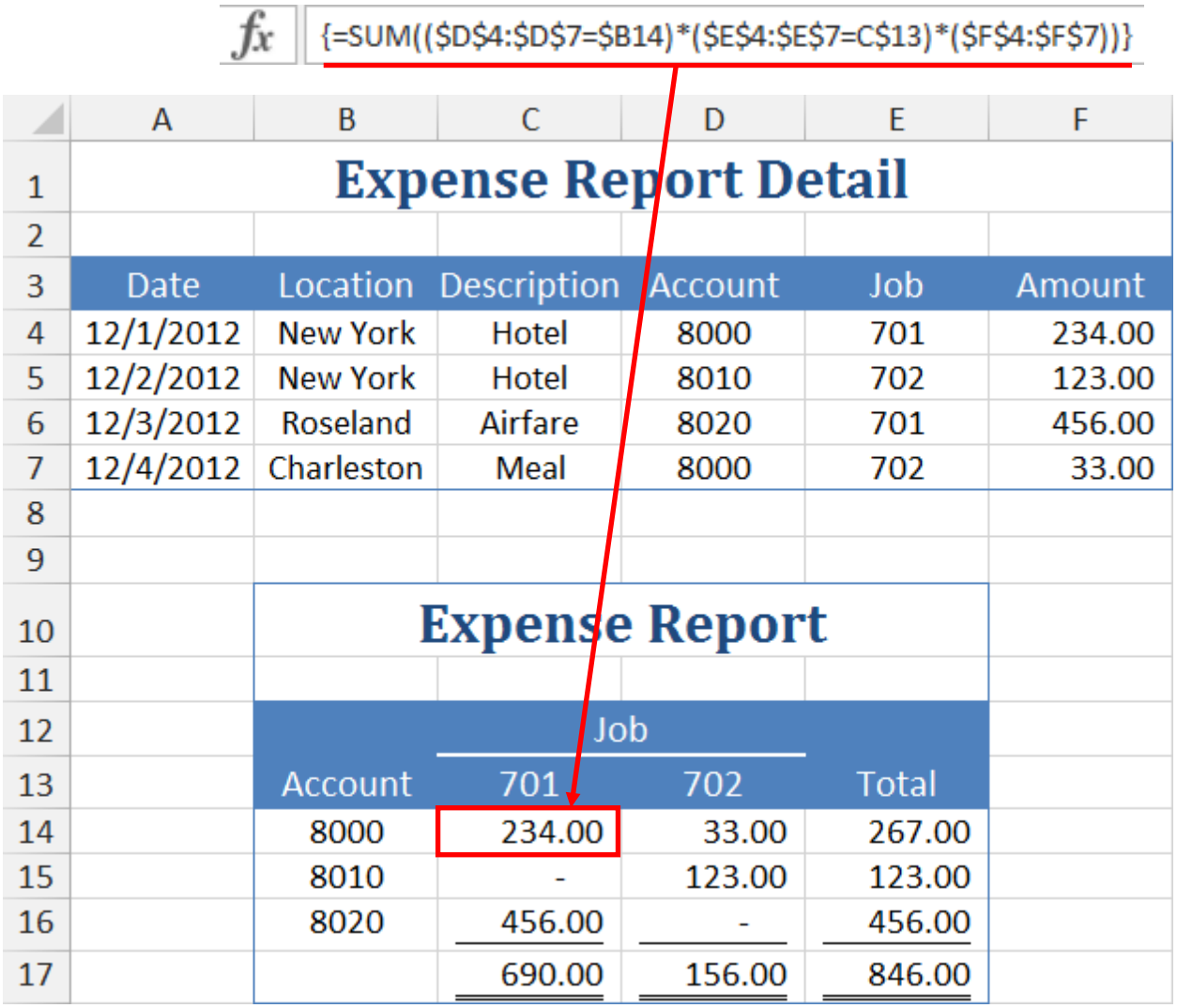


Figure 5 - Simplified Expense Report Using Array Formula

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.

2. Use the SUMPRODUCT function to build the "array" formula. Position the cursor in cell **C14,** type in **=SUMPRODUCT(--($D$4:$D$7=$B14),--($E$4:$E$7=C$13),$F$4:$F$7)** and press **ENTER.** CTRL + SHIFT + ENTER is *not* needed to enter the formula, nor do braces **{ }** surround the formula in the formula bar.

Use the **F9** trick to examine the formula. The conditional tests add an additional layer of complexity to the formula requiring double negatives (--) before each conditional array. Note that the first array evaluates to TRUE;FALSE;FALSE;TRUE. In computing the product of the three arrays, SUMPRODUCT uses the *text labels* resulting from the conditional tests. In formulas, labels evaluate as zero, so a formula without the double negatives (--) in front of the conditional arrays will not calculate correctly. To force the conditional arrays to evaluate as 1;0;0;1, simply perform an arithmetic operation on the result array. Putting a minus sign in front of an array is the same as multiplying the array by -1, but that would make the tests evaluating to TRUE equal to -1. To overcome this limitation, place two minus signs (double negatives) in front of the array to convert them to positive numbers (1), which will then add and multiply correctly.

3. Use the SUMIFS function to build the "array" formula. Position the cursor in cell **C14,** type in **=SUMIFS($F$4:$F$7,$D$4:$D$7,$B14,$E$4:$E$7,C$13)** and press **ENTER.** CTRL + SHIFT + ENTER is *not* needed to enter the formula, nor do braces **{ }** surround the formula in the formula bar.

Use the **F9** trick to examine the formula. The conditional tests do not add any additional complexity to the formula because the function is designed to accommodate the conditional tests. Given the three formulas demonstrated, SUMIFS is clearly the best choice for making these types of conditional array calculations. Similarly, COUNTIFS and AVERAGEIFS can be used to perform conditional array calculations.

Conditional Calculations other than SUM, COUNT, and AVERAGE

Excel's functions for conditional sums, counts, and averages fit the bill in many, but not all circumstances. If a user wants to identify the largest, smallest, or top three observations based on specified conditions, Excel's built in conditional array functions can't provide a solution. Under those circumstances, users must create their own array formulas to make the calculations.

In these examples, two formulas will be constructed – one to calculate the minimum sales by product by quarter using the MIN and IF functions, and one to calculate the maximum sales by product by quarter using the MAX and IF functions.

1. Build an array formula using MIN and IF. In cell **B7** enter the following formula:

   =MIN(IF(Sales!$F$2:$F$881=A7,IF(Sales!$C$2:$C$881=$E$5,Sales!$G$2:$G$881)))

   Make sure to press **CTRL + SHIFT + ENTER** to enter the formula. Otherwise, the formula will evaluate to 0. Note the braces **{ }** surrounding the formula in the formula bar. The formula finds the minimum sales value that meets two conditions – whether the product name in the data record is equal to the product name on the report row and whether the quarter in the data record is equal to the reporting quarter using the IF function. Note that the *value if false* argument of the IF function is not specified for either test. A portion of the report and the formula is displayed in **Figure 6**.

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.

fx {=MIN(IF(Sales!$F$2:$F$881=A7,IF(Sales!$C$2:$C$881=$E$5,Sales!$G$2:$G$881)))}

	A	B	C	D	E
1	Nature's Softness Inc				
2	Quarterly Sales Analysis by Product				
3	For the Year Ended 12/31/2015				
4					
5					Q1
6		Min	Max	Avg	Total
7	Astringent, Organic, 16 oz	8,798.00	11,537.00	10,076.75	80,614.00
8	Astringent, Organic, 24 oz	16,809.00	22,390.00	19,416.50	155,332.00
9	Astringent, Organic, 32 oz	24,765.00	34,650.00	28,454.38	227,635.00
10	Astringent, Organic, 48 oz	8,318.00	11,148.00	9,321.25	74,570.00
11	Cream, Aloe Vera Hand, 9 oz	8,511.00	12,819.00	10,426.13	83,409.00
12	Cream, Extra Moisturizing Hand, 9 oz	9,099.00	13,608.00	11,233.75	89,870.00

fx {=MAX(IF(Sales!$F$2:$F$881=A7,IF(Sales!$C$2:$C$881=$E$5,Sales!$G$2:$G$881)))}

Figure 6 – Array Formulas to Calculate the Minimum and Maximum Values

2. Similar to the formula above, build an array formula using MAX and IF. In cell **C7** enter the following formula:

   =MAX(IF(Sales!$F$2:$F$881=A7,IF(Sales!$C$2:$C$881=$E$5,Sales!$G$2:$G$881)))

   Make sure to press **CTRL + SHIFT + ENTER** to enter the formula. Otherwise, the formula will evaluate to 0. Note the braces **{ }** surrounding the formula in the formula bar. The formula finds the minimum sales value that meets two conditions – whether the product name in the data record is equal to the product name on the report row and whether the quarter in the data record is equal to the reporting quarter using the IF function. Note that the *value if false* argument of the IF function is not specified for either test. The formula is also displayed in **Figure 6**.

Array Formulas in Reporting

Array formulas can be used effectively in the financial reporting and analysis context. In the following examples, three array formulas will be examined. One will use a conventional array formula to automate a six-month rolling report. Another will use SUMIFs to produce a six-month summary report with user interaction. The third will produce a ratio analysis on an exported trial balance.

Accountants often create rolling reports where a new data column is added and a prior data column is removed. Often, the formulas need to be recreated and copied down to complete the analysis. In this example, we will create a rolling report that allows a professional to copy in new data, hide the unwanted data, and have the analysis calculate properly without modifying any of the formulas.

In cell **J9** build the array formula: **=SUM(($B$8:$I$8>EOMONTH(MAX($B$8:$I$8),-6))*(B9:I9))**. Press **CTRL + SHIFT + ENTER** to enter the formula. For the formula to operate correctly, the column headings in the

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.

report must be dates rather than labels. To add data to the report, simply insert a new column **I**, copy in the new data, and hide the oldest month. Nothing else is required. The formula automatically calculates the report from the most recent six months without modification. Even the date in the report heading updates to reflect the latest data column. The report and formula are shown in **Figure 9**.

fx {=SUM(($B$8:$I$8>EOMONTH(MAX($B$8:$I$8),-6))*(B9:I9))}

	A	E	C	D	E	F	G	H	I	J
1	GTM Manufacturing Inc									
2	Retail Sales Division									
3	Summarized Income Statement									
4	Stated in Millions of US Dollars									
5	For the Month Ended August 31, 2015									
6										
7										
8			Mar-15	Apr-15	May-15	Jun-15	Jul-15	Aug-15		Six Months to Date
9	Revenue		$ 250.00	$ 287.50	$ 330.00	$ 380.00	$ 1,247.50	$ 275.00		$ 2,770.00
10										
11	Cost of Sales		125.00	132.50	140.00	147.50	545.00	137.50		1,227.50
12										
13	Gross Margin		125.00	155.00	190.00	232.50	702.50	137.50		1,542.50
14										
15	SG&A		25.00	32.50	40.00	47.50	145.00	27.50		317.50
16										
17	Net Income		$ 100.00	$ 122.50	$ 150.00	$ 185.00	$ 557.50	$ 110.00		$ 1,225.00

Figure 9 – Using an Array Formula to Auto-Calculate a Rolling Report

The next reporting example produces a rolling six-month exception report from data stored in an Excel table. The table could consist of imported data or could be connected to an external data source, such as an accounting solution. An ODBC connection to an external data source would facilitate updating the table and associated reports with a single click. The formulas use the SUMIFS function to make the calculations. Conditional formatting is applied to the report to highlight exceptional values and end-user interaction is accomplished with data validation. The completed report is shown in **Figure 10**.

Similar to the previous example, the column headings in the report must be dates rather than text labels. The column headings are tied by formula to the date selected by the end user in cell **H4**. The formula in the field of the report is shown below. It need *not* be entered using CTRL + SHIFT + ENTER because SUMIFS supports array calculations natively.

=SUMIFS(ProductSales[Amount],ProductSales[EOM],B$5,ProductSales[Product], $A6)

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.

	A	B	C	D	E	F	G	H
1	DNM Marketing LLC							
2	Product Sales Performance							
3	For the six months ended March 31, 2012							
4								3/31/2012
5		Oct 2011	Nov 2011	Dec 2011	Jan 2012	Feb 2012	Mar 2012	Total
6	Astringent, Organic, 16 oz	$ 49,484.17	$ 43,906.34	$ 42,542.72	$ 39,380.12	$ 41,922.31	$ 36,989.43	$ 254,225.09
7	Astringent, Organic, 24 oz	93,371.40	84,856.59	80,258.19	74,586.43	80,496.52	69,722.65	483,291.78
8	Astringent, Organic, 32 oz	139,284.27	124,495.79	120,392.38	110,537.24	118,032.41	104,534.38	717,276.47
9	Astringent, Organic, 48 oz	45,638.88	41,105.92	40,409.70	36,194.47	38,957.18	35,156.37	237,462.52
10	Cream, Aloe Vera Hand, 9 oz	50,650.19	45,694.80	44,941.69	40,143.25	43,380.90	39,334.88	264,145.71
11	Cream, Extra Moisturizing Hand, 9 oz	53,939.34	50,007.97	47,701.16	42,843.83	47,775.96	41,485.86	283,754.12
12	Cream, Hand and Body, 16 oz	93,090.94	85,133.00	82,132.24	74,431.29	80,783.67	71,482.56	487,053.70
13	Lotion, Extra Moisturizing Body, 9 oz	50,763.19	45,762.41	44,729.10	40,299.82	43,467.34	38,888.35	263,910.21
14	Lotion, Organic Body, 16 oz	111,434.61	101,154.03	97,867.72	88,554.87	96,006.89	85,132.00	580,150.12
15	Lotion, Organic Body, 24 oz	158,572.47	144,447.88	138,435.78	125,882.51	137,122.58	120,762.29	825,223.51
16	Lotion, Organic Body, 9 oz	58,592.62	53,436.38	49,654.01	46,530.91	50,860.93	43,263.09	302,337.94
17	Lotion, Organic Hand and Body, 16 oz	115,096.50	103,266.74	98,808.81	91,655.75	98,168.20	85,687.15	592,683.15
18	Lotion, Organic Hand and Body, 9 oz	63,313.28	55,903.00	54,579.32	50,351.79	52,901.04	47,538.38	324,586.81
19	Mask, Organic Facial, 9 oz	101,768.81	89,834.10	91,021.14	80,581.92	85,398.30	79,449.55	528,053.82
20	Mask, Wrinkle Reducing Facial, 16 oz	25,422.63	23,620.20	22,917.15	25,273.89	22,500.62	20,007.10	139,741.59
21	Mask, Wrinkle Reducing Facial, 24 oz	38,182.65	36,004.97	34,726.27	38,034.47	34,319.27	30,132.96	211,400.59
22	Mask, Wrinkle Reducing Facial, 9 oz	138,808.40	127,940.05	120,148.27	110,641.10	121,205.71	104,258.62	723,002.15
23	Scrub, Hypo-Allergenic, 16 oz	49,553.80	44,092.41	42,441.37	39,260.06	41,760.62	37,003.49	254,111.75
24	Scrub, Hypo-Allergenic, 24 oz	86,515.30	77,364.24	74,633.38	68,515.15	73,578.35	65,288.61	445,895.03
25	Scrub, Hypo-Allergenic, 9 oz	43,539.34	39,106.34	37,346.08	34,644.09	37,345.79	32,465.52	224,447.16
26								
27	Total Sales	**$1,567,022.79**	**$1,417,133.16**	**$1,365,686.48**	**$1,258,342.96**	**$1,345,984.59**	**$1,188,583.24**	**$ 8,142,753.22**

Figure 10 – Exception Report Created Using SUMIFS

Array Constants

Array constants are a list of values that can be used as arguments in your array formulas.
Arrays can be either 1-dimensional or 2-dimensional depending on the number of rows and columns.
A 1-dimensional array can exist in a single row or a single column.
You can create an array constant and you can give it a name that can then be used in your formulas.

Array constants are a list of values that can be used as arguments in your array formulas.
Arrays can be either 1-dimensional or 2-dimensional depending on the number of rows and columns.
A 1-dimensional array can exist in a single row or a single column.
One dimensional array {1, 2, 3, 4}

	A	B	C	D	E	F
1						
2		1	2	3	4	
3						

©BetterSolutions.com

A 2-dimensional array can exist in a block of cells, made up of multiple rows and columns.
Two dimensional array {1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12}
Notice that vertical elements are separated by a semi-colon (**;**) as opposed to a comma.

	A	B	C	D	E	F
1						
2		1	2	3	4	
3		5	6	7	8	
4		9	10	11	12	
5						

©BetterSolutions.com

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.

An array constant can contain numbers, text, logical values as well as error values. Any text must be enclosed in double quotation marks (i.e. "some text").
An array constant cannot contain any numbers with commas, dollars, parentheses, percent signs, worksheet functions or any other arrays.
Arrays do not actually need to be stored in cells and can alternatively be stored in memory during a calculation.
When an array is stored in memory during a calculation the formula is executed slightly faster.

One Dimensional arrays

{"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"}

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2		Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	
3														

©BetterSolutions.com

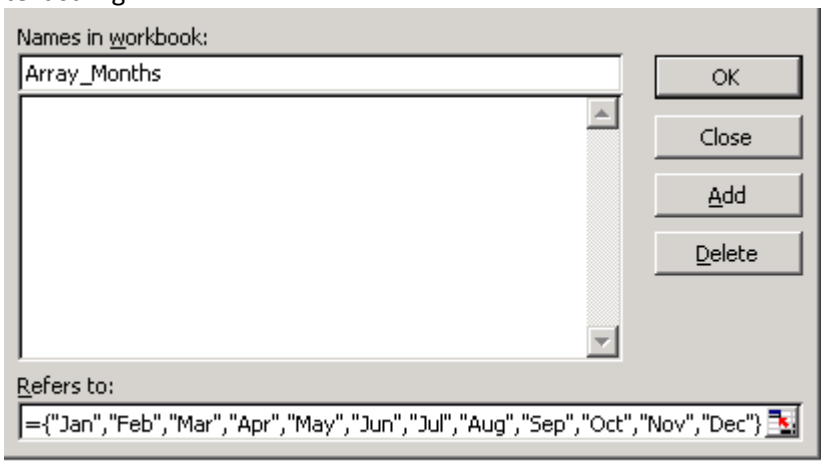
This array has commas separating the elements which means that the array is a horizontal array.
You could replace the commas with semi-colon's to define a vertical array, {"Jan"; "Feb"; "Mar"; "Apr"; "May"; "Jun"; "Jul"; "Aug"; "Sep"; "Oct"; "Nov"; "Dec"}

Two Dimensional arrays

{"Sales","Sales","Sales","Profit","Profit","Profit";2000,2001,2002,2000,2001,2002}

Naming an Array Constant.

If you find yourself entering the same list again and again you can save a list as an array constant.
Saving your list as an array constant will allow to quickly enter the list into cells.
Select **(Insert > Name > Define)** to display the Define Name dialog box.
Enter the name of your array constant and enter the array in the Refers to box.
Notice that the array must have an equal sign before it. Without this equal sign the name will just refer to a text string.



©BetterSolutions.com

You must remember to enter the curly brackets as well. They will not be entered automatically in this case.

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.

Using the Array Constant

Once the array constant has been defined you can enter it quickly into cells and also use it directly in your formulas.

You can then enter the array constant directly in your cells {=Array_Months}

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2		Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	
3														

©BetterSolutions.com

Copyright © 2018, K2 Enterprises, LLC. Reproduction or reuse for purposes other than a K2 Enterprises' training event is prohibited.