Harness the Power of Array Formulas

K2 Enterprises®
Professional Education and Consulting

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

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

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

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.

Rounding Off in a Total

This example uses an array formula to round off in a total. Many accounting professionals have been faced with the need to sum a large number of amounts that result from floating point calculations. The amounts may be the result of formulas or may be imbedded in data imported from other systems. A problem arises when the amounts are not rounded off because Excel uses the underlying values rather than the displayed values in calculating formulas, including summary totals. Conventional practice requires the use of the ROUND function to round off all of the calculated amounts prior to summing their values, but that solution is cumbersome and time-consuming when a large number of values are present. A better alternative would be to build a formula that can round and sum the amounts in a single calculation. A simple array formula like the one displayed below can perform this task quickly and easily.

{=SUM(ROUND(E5:E8,2))}

Note the braces that surround the array formula. The braces are not typed in but are added by Excel when an array formula is entered. To enter an ordinary array formula, type in the formula and then press **CTRL + SHIFT + ENTER**.

		No Rounding	Conventional Rounding	Array Formula
Total	Units	Average Unit Cost	Average Unit Cost	Average Unit Cost
247.11	4.00	61.78	61.78	61.78
494.22	5.00	98.84	98.84	98.84
741.33	6.00	123.56	123.56	123.56
988.44	7.00	141.21	141.21	141.21
		425.38	425.39	425.39

{=SUM(ROUND(E5:E8,2))}

Figure 4 - Using an Array Formula to Round and Sum Data in a Total

Again, let's use the technique discussed earlier to examine the array formula. Position the cursor in cell **E10** and press **F2**. Then, click inside the **ROUND** function, click ***number*** in the **Screen Tip**, and press **F9** to see the *unrounded* values. Press **CTRL + Z** to undo the changes, and then click inside the **SUM** function but before the **ROUND** function, click ***number 1*** in the **Screen Tip,** and press **F9** to see the *rounded* array values. This examination helps us understand just how the array formula is making the calculation. It's not rounding the sum, it's summing the rounds!

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.

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

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**.

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**.

Extracting Data from a Range or Table

Array formulas can be used to extract data from an Excel table or an ordinary table array. While VLOOKUP can return a single value based on specified criteria, the function only returns the data from the first record that matches the condition when multiple records match the criteria. Array formulas can be used to extract *all* records that match one or more specified criteria.

In the first demonstration, an array formula will be used to extract multiple records that match a single criterion. The task is to extract a list of employees assigned to specified operating groups. Further, this example looks up data in column 2 of a table array and returns the data from column 1 in the same array. Have you ever tried that with VLOOKUP?

1. First, create a simple formula using COUNTIFS in cell **F2** to count the number of instances in the data where the selected operating group appears. This will serve as the **k** argument in the AGGREGATE function that identifies what rows to extract.

2. Build the following array formula in cell **E5** and then copy it down to cell **E46**:

=IF(ROWS(D$5:D5)>$F$2,"",INDEX(EMP[Employee],AGGREGATE(15,6,

(ROW(EMP[Employee])-ROW($A$2)+1)/(EMP[Group]=$E$2),ROWS(D$5:D5))))

Press **ENTER** to enter the formula. The formula identifies the rows in the employee table containing the specified operating group and extracts the names of all employees assigned to that group. The analysis and the necessary formulas are displayed in **Figure 7.**

Figure 7 – Using an Array Formula to Extract Data

In the next example, array formulas will be used to extract transactional summaries from an Excel table. The data extracted must meet user-defined criteria. Similar to the previous demonstration, first create a simple formula using COUNTIFS to count the number of instances in the data where the criteria are met. Then build the following array formula in cell **J14** and copy it down to cell **J115**:

=IF(ROWS(J$14:J14)>$K$6,"",INDEX(Sales[EOMonth],AGGREGATE(15,6,

(ROW(Sales[EOMonth])-ROW($A$2)+1)/((Sales[Quarter]=$K$3)*

(Sales[Product Line]=$L$3)*(Sales[Region]=$M$3)*(Sales[Year]=$J$3)),ROWS(J$14:J14))))

Create similar array formulas for each of the desired columns to be extracted by changing the *array* argument in the INDEX function to match the desired column. Copy the formulas down to row **115**. As the selection criteria are modified, the array formulas will extract the matching records. A portion of the worksheet is shown in **Figure 8**.

J	K	L	M	N
		Criteria		
Year	Quarter	Product Line	Region	
2014	Q3	Creams	Southeast	
Row Count	9			
EOMonth	Year	Region	Product Name	Amount
7/31/2014	2014	Southeast	Cream, Aloe Vera Hand, 9 oz	19,402.00
7/31/2014	2014	Southeast	Cream, Extra Moisturizing Hand, 9 oz	20,082.00
7/31/2014	2014	Southeast	Cream, Hand and Body, 16 oz	36,549.00
8/31/2014	2014	Southeast	Cream, Aloe Vera Hand, 9 oz	17,643.00
8/31/2014	2014	Southeast	Cream, Extra Moisturizing Hand, 9 oz	18,536.00
8/31/2014	2014	Southeast	Cream, Hand and Body, 16 oz	31,029.00
9/30/2014	2014	Southeast	Cream, Aloe Vera Hand, 9 oz	16,266.00
9/30/2014	2014	Southeast	Cream, Extra Moisturizing Hand, 9 oz	17,278.00
9/30/2014	2014	Southeast	Cream, Hand and Body, 16 oz	31,400.00

Figure 8 – Extracting Data Matching Multiple Criteria

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

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	C	D	E	F	G	H	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)

	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

The last reporting example automates the preparation of a financial ratio analysis from an exported QuickBooks trial balance. The ratio components (quick assets, current assets, current liabilities, etc.) are calculated using array formulas and then the financial ratios are calculated from the ratio components, as shown in **Figure 11**. The formulas are built to function and calculate correctly even if additional accounts have been added, although the newly created accounts must fall within identified boundaries for particular account types. Note that all accounts *must* have an account number and the account numbers must be exported to Excel.

Figure 11 – Using Array Formulas to Calculate Ratio Components

Functions that Return Result Arrays

Excel includes a set of functions that produce *result arrays*. In other words, users can enter a single formula and have the formula return results to multiple cells. Among them are TRANSPOSE, TREND, FREQUENCY, LINEST, AND LOGEST. Two examples of result arrays will be demonstrated, the first of which is the calculation of linear regression components, useful in the process of forecasting revenues and expenses.

Regression analysis is a statistical method used to estimate and understand relationships between a dependent variable and one or more independent variables. Linear regression models the relationship between two or more variables by fitting a linear equation to observed data. In regression analysis, one or more independent variables are used to predict a single dependent variable. For example, a sales executive may want to investigate the effect of advertising expenditures on sales revenue. The relationship may be explained by the linear regression equation:

$$\mathbf{Y = \alpha + \beta x}$$

where **Y** is the dependent variable (sales revenue)

x is the independent variable (advertising expenses)

α is the intercept (the value of **Y** when ***x*** = 0), and

β is the slope of the line.

The strength of the association between the independent and dependent variables is quantified by their correlation coefficient. The correlation coefficient always takes a value between minus one and one, with one or minus one indicating perfect correlation (with all points along a straight line). A positive correlation indicates a positive association between the variables (where increasing values in one variable correspond to increasing values in the other variable), while a negative correlation indicates a negative association between the variables (where increasing values in one variable correspond to decreasing values in the other variable). A correlation value of zero indicates that there is no association between the variables.

The square of the correlation coefficient, $\mathbf{R^2}$, represents the percentage of variation in the dependent variable that is explained by variation in the independent variable. If a correlation of 0.80 is observed between the variables (for example, sales revenue and advertising expense), then a linear regression model will account for 64% (0.80^2) of the variability in the data.

The major conceptual limitation of regression analysis is that one can ascertain relationships but can never be sure about the underlying causal mechanism. For example, a strong positive relationship may exist between the number of overtime hours worked by production personnel and the quantity of goods produced, but a regression analysis cannot support the conclusion that the quantity of goods produced actually caused the overtime. Other factors, such as machinery breakdowns or production line inefficiencies, may have caused it. While regression analysis can quantify the relationship between overtime hours worked and quantity of goods produced, it cannot explain the cause of the relationship.

Excel contains a number of tools that are useful in conducting regression analysis. The regression analysis tool in the Analysis ToolPak add-in allows estimation of a single dependent variable based on up to sixteen independent variables. Regression analysis is not the only tool available in the ToolPak. Tools are also available to compute descriptive statistics, calculate moving averages and ANOVAs, generate random numbers, and determine ranks and percentiles, etc. To conduct regression analysis using the ToolPak requires multiple discrete steps. A better way for most business professionals to do regression analysis is with formulas because the results can be referenced easily with other formulas and because a change in the data causes the formulas to update immediately.

LINEST can be used to produce an array formula to calculate the slope and intercept of a linear regression without using the Analysis ToolPak. It also can be used to generate descriptive statistics regarding the linear regression, including R-Square. As shown in **Figure 14**, LINEST calculates the slope and intercept based on the input of the dependent and independent variables in an array formula. Prior to entering the formula, select the range **E2:G6** to produce the output displayed. To modify the LINEST formula so that it returns descriptive statistics, and then enter the array formula: **=LINEST(C2:C31,A2:B31,TRUE,TRUE)**. Press **CTRL + SHIFT + ENTER** to enter the formula. The results with a legend to interpret the results are displayed in **Figure 12**.

β2	β1	α
17.00881257	73.03751936	-91683.19386
6.170402417	6.298189624	102594.5494
0.931225503	19683.5027	
182.7936937	27	
1.41643E+11	10460887519	
Legend		
β2	β1	α
Coefficient	*Coefficient*	*Coefficient*
Std Error	*Std Error*	*Std Error*
R2	*Std Error Reg*	
F Stat Reg	*df*	
SS Reg	*SS Residuals*	

Figure 12
Using LINEST to Calculate Linear Regression Components

In our last example, the TREND function will be used to forecast sales revenue. TREND generates values along a linear trendline and can be used to predict multiple future dependent values based on the relationship between existing dependent and independent variables. Note that TREND formulas can produce result arrays, but must be entered using **CTRL + SHIFT + ENTER**. **Figure 13**, shows a TREND formula being used to forecast sales for two additional months based on the actual advertising expenses and monthly sales of thirty prior months.

fx {=TREND(C2:C31,A2:B31,A32:B33)}

Ad Expense 1 Month Prior	Ad Expense 2 Months Prior	Monthly Sales
$ 20,000	$ 20,000	$ 1,700,000
19,000	20,000	1,650,000
20,000	19,000	1,725,000
21,000	20,000	1,750,000
20,000	21,000	1,725,000
21,000	20,000	1,750,000
20,500	21,000	1,750,000
21,000	20,500	1,775,000
21,000	21,000	1,775,000
21,000	21,000	1,800,000
21,500	21,000	1,800,000
20,500	21,500	1,775,000
21,750	22,500	1,850,000
22,000	21,750	1,900,000
24,000	22,000	2,035,411
25,000	23,000	2,125,457

Figure 13
Using a Trend Array to Forecast Sales Revenues

Using Array Constants

An *array constant* is an array of hard-coded values. It can be entered directly into a formula or can be saved and referenced as a defined name. In this example, the US corporate income tax rate table will be stored as a defined name. The defined name will refer to the array constant that makes up the table.

To create the defined name, enter a simple formula that references the tax table. Then press **F9** to convert the cell references into an array. Copy the array to the clipboard with **CTRL + C**. Open the **Name Manager** on the **Formulas** tab and click **New**. Name the defined name **TaxTable** and paste the array into the **Refers to**: box and press **OK**. Note that column items are separated by columns and rows are separated by semi-colons.

Once the defined name is created, the array constant can be used just as any other *table array* in a VLOOKUP function to calculate estimated corporate taxes. Save the constant in your tax worksheet templates or your default Excel template so that it will always be available. The worksheet used to create the array id shown in **Figure 14**.

Figure 14 – Storing an Array Constant as a Defined Name