K2's Mastering Advanced Excel Functions

Continuing Professional Education Series

K2 Enterprises www.k2e.com

K2E Canada Inc. www.k2e.ca

K2 Enterprises
1905 W. Thomas St., Suite D #363
Hammond, LA 70401-2901
985.542.9390

K2E Canada Inc.
174 Brewer Court
Burlington, Ontario, L7L4G4
905-746-1581

CONTRIBUTORS: The shareholders, associates, and staff of K2 Enterprises and K2 Canada Inc. contributed to producing these materials.

Table of Contents

Introduction

Three Rules of this Seminar

1. **There Are No Dumb Questions.** Please ask questions if you need further explanation. We welcome your questions and will do our best to provide accurate and meaningful answers.

2. **Obtain the Information for Which You Came.** If you have a specific interest in a topic, please let us know. If your topic is not part of the course materials, we will do our best to accommodate your request. Your instructor is anxious to meet your needs.

3. **Share and Share-Alike.** Your instructor is here to share relevant information about the course. We hope that you will be willing to do the same. If you have helpful tips or experiences, please share them with the other participants and us.

Course Development

The shareholders, associates, and staff of K2 Enterprises and K2E Canada Inc. developed this course and the related course materials. We aim to deliver high-quality educational practices that help you use and understand the tools and concepts discussed in this session more effectively.

We designed our course presentation style for busy professionals. Through numerous short case studies and feature reviews, we deliver concise, easy-to-understand, easy-to-absorb information.

To maximize your time and the value of attending this course, we have carefully filtered out trivial and obscure features. Our goal is not to cover every keystroke and menu option, but to provide a fast-paced course that focuses on state-of-the-art information technology and how it can benefit you and your staff.

About K2 Enterprises

For over three decades, our team members have delivered more than 45,000 technology-focused seminars, webinars, and conferences worldwide. To stay on top of technology, our team members read numerous technology trade magazines, attend conferences, and speak with technology companies each year to stay current with the latest information for our courses. Also, we experience everything we teach and recommend to our audiences. Because our shareholders and associates are based in different states and provinces, using technology efficiently and appropriately is essential to our business's success and survival.

Our Mission Statement

K2 aims to develop and deliver the highest-quality technology seminars and conferences for business professionals. We work cooperatively with professional organizations such as state CPA societies, associations of Chartered Professional Accountants, and technology vendors. We make every effort to maintain high integrity, family values, and friendship among all involved.

Our Instructional Team

For a complete listing of instructors and bios, please visit:

K2 Enterprises

Instructors - K2 Enterprises

K2E Canada Inc.

Instructors - K2E Canada Inc

The K2 Seminar Curriculum

Full-day Seminars

- K2's Advanced Excel
- K2's Business Intelligence, Featuring Microsoft's Power BI Tools
- K2's Case Studies In Fraud And Technology Controls
- K2's Excel Best Practices
- K2's Excel Essentials For Staff Accountants
- K2's Excel PivotTables For Accountants
- K2's Excel Tips, Tricks, And Techniques For Accountants
- K2's Next Generation Excel Reporting
- K2's QuickBooks For Accountants
- K2's Small Business Internal Controls, Security, And Fraud Prevention And Detection
- K2's Technology For CPAs – Don't Get Left Behind

Half-day Seminars

- K2's Artificial Intelligence For Accounting And Financial Professionals
- K2's Best Word, Outlook, And PowerPoint Features
- K2's Better Productivity Through Artificial Intelligence and Automation Tools
- K2's Case Studies In Fraud And Technology Controls
- K2's Data Analytics For Accountants And Auditors
- K2's Ethics And Technology
- K2's Excel Charting And Visualizations
- K2's Implementing Internal Controls In QuickBooks Online Environments
- K2's Improving Productivity With Microsoft 365Cloud Applications And Services
- K2's Mastering Advanced Excel Functions
- K2's Microsoft Teams
- K2's Securing Your Data: Practical Tools For Protecting Information
- K2's Technology Update
- K2's Top PDF Features You Should Know

Besides the seminars listed above, K2 also produces numerous one and two-day technology conferences annually. You can find a complete listing of our course offerings, including dates, locations, and course descriptions, on our website at:

K2 Enterprises
Technology Conference - K2 Enterprises

K2E Canada Inc.
Technology Conference - K2E Canada Inc.

On-Site Training

Working cooperatively with state CPA organizations and associations of Chartered Professional Accountants, we are pleased to offer all our seminars and conference sessions as on-site training opportunities. On-site training provides companies of all sizes with numerous advantages, including the following.

- ***Customization***. K2 can customize the content of all on-site training events to meet your organization's specific needs. For example, we can combine selected parts of existing courses into a focused learning experience to meet your team's needs. Likewise, we can develop custom content from the ground up and deliver it to your team members, ensuring we meet your training objectives.

- ***Cost savings***. For organizations of all sizes, on-site training can provide significant cost savings compared to public training venues. In addition to potentially reduced registration fees, these cost savings materialize in the form of reduced travel costs and reducing the amount of time team members spend away from the office.

- ***Convenience***. Because you select your on-site training event's date, time, and location, on-site training is more convenient for you and your team than traditional training options. Many organizations schedule on-site training with staff meetings, retreats, and customer/client events to leverage the value of all participants' time.

Customization, ***cost savings***, and ***convenience*** are the three C's of "training to go." For more information on how you and your organization can experience these benefits or schedule an on-site training event, please visit:

K2 Enterprises
On-site Training - K2 Enterprises

caroline@k2e.com

K2E Canada Inc.
Seminars - K2E Canada Inc

caroline@k2e.ca

Web-Based Training

K2 offers web-based CPE and CPD to accounting and business professionals throughout the United States and Canada. K2's award-winning instructors deliver two-, four-, and eight-hour webinars on numerous technology-focused topics.

Additionally, K2 provides on-demand training. On-demand courses offer you the advantage of receiving top-quality training at a time and location that fits your busy schedule. You can start a class today and complete it in the future. For more information, please visit:

For more information on either platform, please visit:

K2 Enterprises
Webinars - K2 Enterprises

K2E Canada Inc.
Webinars - K2E Canada Inc

K2 Internet Sites

K2 maintains multiple websites containing information and links relevant to CPAs and other business professionals as a service to the profession. Further, the content of these sites can enrich your experience during and after our seminars. In addition, our pages contain links to some of the top accounting and content management software solutions and even to a few sites that are just fun places to visit. If you have any comments or questions regarding our web pages, please email *webmaster@k2e.com*.

K2 has two primary websites. You can view information regarding CPE and CPD opportunities, access a library of technology tips and articles, and link to K2-related websites.

www.k2e.com

www.k2e.ca

www.cpafirmtech.com

CPA Firm Technology contains technology-related information explicitly targeted at public accounting firms. We include product reviews, hardware and operating system discussions, affinity programs for CPA firms, and Cloud computing resources.

www.accountingsoftwareworld.com

Accounting Software World provides information related to accounting and financial management solutions. Here, you can find software and service reviews, product comparisons, white papers, and other material of interest to those seeking information about accounting software and service solutions.

www.totallypaperless.com

For information about document management and paperless processes, please visit **Totally Paperless**. In addition, you will find reviews on hardware and software used by all types of businesses in document imaging and management processes.

Course Summary

Excel offers over 500 functions, many of which are unused by most accounting and finance professionals. You can use these functions to calculate mortgage interest and principal amortization, trap errors, and build reports directly from data summarized in a PivotTable or the Excel Data Model, among other items. This session will put you on the path to improved Excel productivity by introducing you to powerful and time-saving Excel functions and array formulas, including IPMT and PPMT. You will also learn about Excel's GETPIVOTDATA, OFFSET, and INDIRECT functions. Further, we will introduce you to Data Analysis Expressions (DAX), cube formulas, and new tools available only in subscription-based versions of Excel, such as XLOOKUP.

Learning Objectives

Upon completing this chapter, you should be able to:

- List examples of best practices for constructing formulas in Excel spreadsheets;
- Identify situations in which each of the following functions might be useful: SUMIFS, SWITCH, and STOCKHISTORY;
- Distinguish between the XLOOKUP function and legacy Excel functions such as VLOOKUP, HLOOKUP, INDEX, and MATCH;
- Cite examples of when using Dynamic Arrays would be useful;
- Differentiate between Excel's AGGREGATE and SUBTOTAL functions; and
- Identify conditions in which the FORECAST.ETS function is preferable to the FORECAST function.

Five Fundamentals of Spreadsheet Design

Document Your Workbooks

While it may seem like this is stating the obvious, you should add at least some minimal documentation to each Excel workbook you create. Documenting the workbook will help you and others understand the workbook's nature, purpose, scope, and status. The level of documentation will vary based on many factors, including:

- The complexity of the workbook,
- The sensitivity of the data in the workbook, and
- Whether others use the workbook.

However, at a minimum, you should incorporate each of the following items into every Excel workbook you create:

- Your name and contact information,
- The general purpose of the workbook, and
- Any special considerations associated with the workbook.

Manage Spreadsheet Assumptions

A frequent spreadsheet design flaw is embedding assumptions into formulas. To illustrate, compare the two formulas presented in **Figure 1**. The left formula contains embedded assumptions, whereas the one on the right includes cell references to these assumptions. If you or another user of this spreadsheet wanted to change the interest rate to 4.5% and see the impact on the monthly payment, which scenario would be easier to edit? Clearly, "what-if" analysis is easier with the design on the right because changing assumptions requires only editing assumption cells in the spreadsheet. On the other hand, changing assumptions mandates editing the entire formula with the scenario on the left.

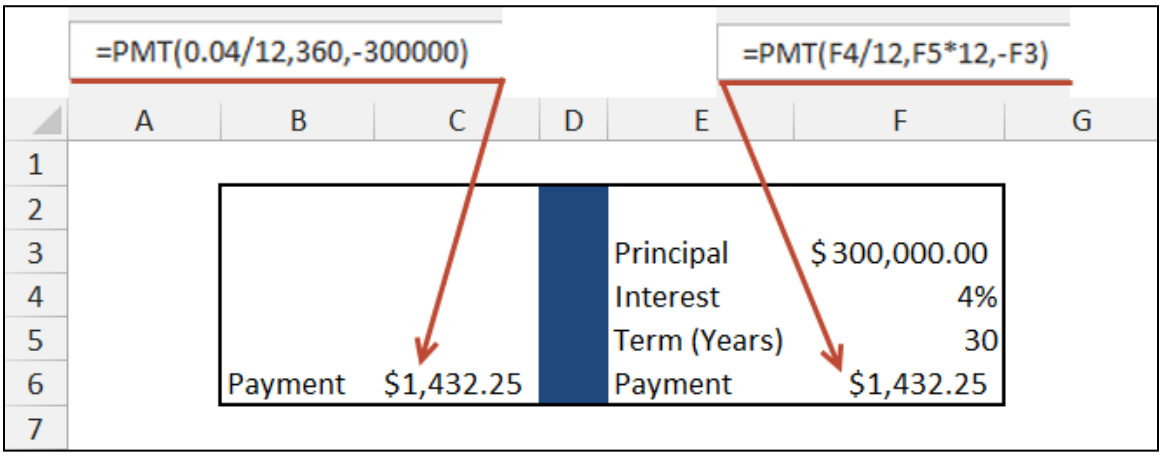


Figure 1 - Comparing Formulas with and without Embedded Assumptions

In addition to the ease of editing, changing assumptions in the scenario on the left presents another issue – it is more prone to error. Whenever users edit formulas, they risk inadvertently altering the formula or some component, generating an incorrect result. Consequently, to help ensure that the integrity of formulas remains intact, you should never force other users to edit formulas to change assumptions.

Ease Workbook Navigation

If your workbook contains many worksheets or some include large volumes of data, you should ensure it is easy to navigate. Multiple options exist to facilitate navigation, including inserting a table of contents and hyperlinks and standardizing workbook and worksheet layouts.

Build Error Checking Routines into Your Workbooks

Despite best efforts, errors may still occasionally creep into your workbooks. Therefore, as a best practice, incorporate routines into your workbooks to alert you to erroneous conditions. Multiple options exist for this purpose. One such option is to use an **IF** statement to alert you to an incorrect condition. For example, when creating a balance sheet, you could use the formula shown in **Figure 2** to compare the value of Total Assets (cell B35) to Total Liabilities and Equity (cell B61) and generate a warning message when these amounts differ.

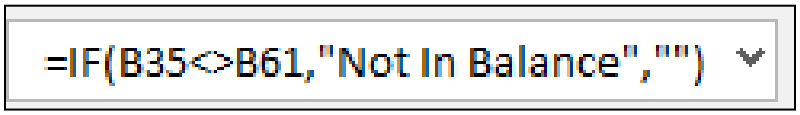


Figure 2 - Formula to Test for Out-of-Balance Condition

Planning for Additional Data

When creating a spreadsheet, you should assume that the model will grow over time and that you, or others, will add additional data in the future. Given this assumption, you should incorporate expansion principles into the model. Examples of best practices in this area include:

- Using Tables for storing data;
- Using Defined Names and Dynamic Defined Names; and
- Modifying Spreadsheet Layout and Formulas to Accommodate Future Data Expansion.

XLOOKUP

Microsoft 365 Apps subscribers gain access to powerful new functionality with continuous updates. Among the most useful new functions is **XLOOKUP**, a potential successor to VLOOKUP, HLOOKUP, and INDEX. Notably, XLOOKUP works with traditional data ranges or tables, and you can use it across multiple worksheets or workbooks. Additionally, XLOOKUP provides greater functionality and overcomes many of the limitations of its predecessors. For example, XLOOKUP defaults to an exact match and can perform lookups from the bottom up and the top down. Further, with XLOOKUP, the lookup column or row no longer needs to be sorted. It also allows users to specify a range of cells to perform the lookup instead of a column number in a lookup table. Hence, the order of the columns in a lookup range or table does not matter. This characteristic allows XLOOKUP to look to the lookup column's left or above the lookup row, which are things that VLOOKUP and HLOOKUP cannot do. The syntax of XLOOKUP is as follows:

XLOOKUP(lookup value, lookup array, return array,
[if not found], [match mode], [search mode])

where:

Lookup value represents the value to find.

Lookup array is the range or table column in which to search.

Return array is the range or table column from which to return a corresponding value.

If not found (optional) represents what to return if XLOOKUP does not find the lookup value; the default setting is "#NA."

Match mode (optional) determines how to perform the search; defaults to (0) exact match.

Option Number	Option Behavior
0 or Omitted	Exact match (default)
-1	Exact match or next smaller item (default for VLOOKUP)
1	Exact match or next larger item
2	Wildcard character match

Search mode (optional) determines the search order; defaults to first-to-last.

Option Number	Option Behavior
1	Search first-to-last (default)
-1	Search last-to-first
2	Binary search sorted in ascending order
-2	Binary search sorted in descending order

In this first example, XLOOKUP will perform a reverse lookup. Since XLOOKUP uses a lookup array (column or row) instead of a lookup table, you can lookup on any column (or row). Further, XLOOKUP can return the result from any column (or row).

Nature's Softness carries an extensive inventory of quality facial products. Owner Debbie Nelson often looks up a vendor's part number when restocking a product. She uses a simple VLOOKUP formula to accomplish this task. However, she often needs to look up an internal SKU from a vendor's part number. Since the SKUs are in a column to the left of the one containing vendor part numbers, VLOOKUP cannot perform this task. While Debbie could use the MATCH and INDEX functions to perform this task, she has limited knowledge of their use. Her assistant pointed out that the XLOOKUP function is a Swiss Army knife for data lookups and built a simple formula to retrieve the desired information. The following formula performs a reverse lookup using a vendor's part number to retrieve a SKU from the Inventory List.

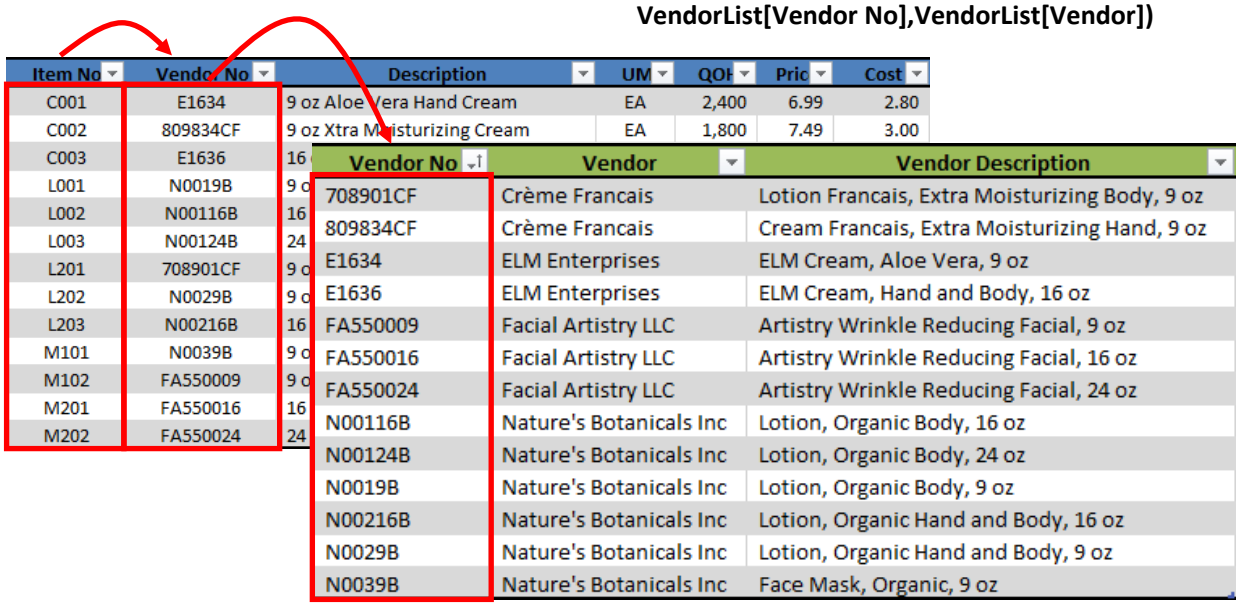
=XLOOKUP(C5,InventoryList[Vendor No],InventoryList[SKU])

However, when XLOOKUP does not find a vendor's part number in the lookup array, it returns **#N/A,** just like VLOOKUP and HLOOKUP. A minor change to the formula cures that shortcoming without using IFERROR. The following formula adds the fourth optional argument, which XLOOKUP inserts when it doesn't find the lookup value. The argument can be a value, formula, cell reference, defined name, or text (enclosed in quotation marks). In this case, XLOOKUP will display "Item Not Found" when it does not find a vendor part number in the lookup array.

=XLOOKUP(C6,InventoryList[Vendor No],InventoryList[SKU],"Item Not Found")

Debbie needs to retrieve the name of the vendor when reordering a product. Unfortunately, the vendor list is in another table. XLOOKUP overcomes this problem because it works across multiple worksheets and workbooks.

=XLOOKUP(XLOOKUP(B2,InventoryList[Item No],InventoryList[Vendor No]),
VendorList[Vendor No],VendorList[Vendor])



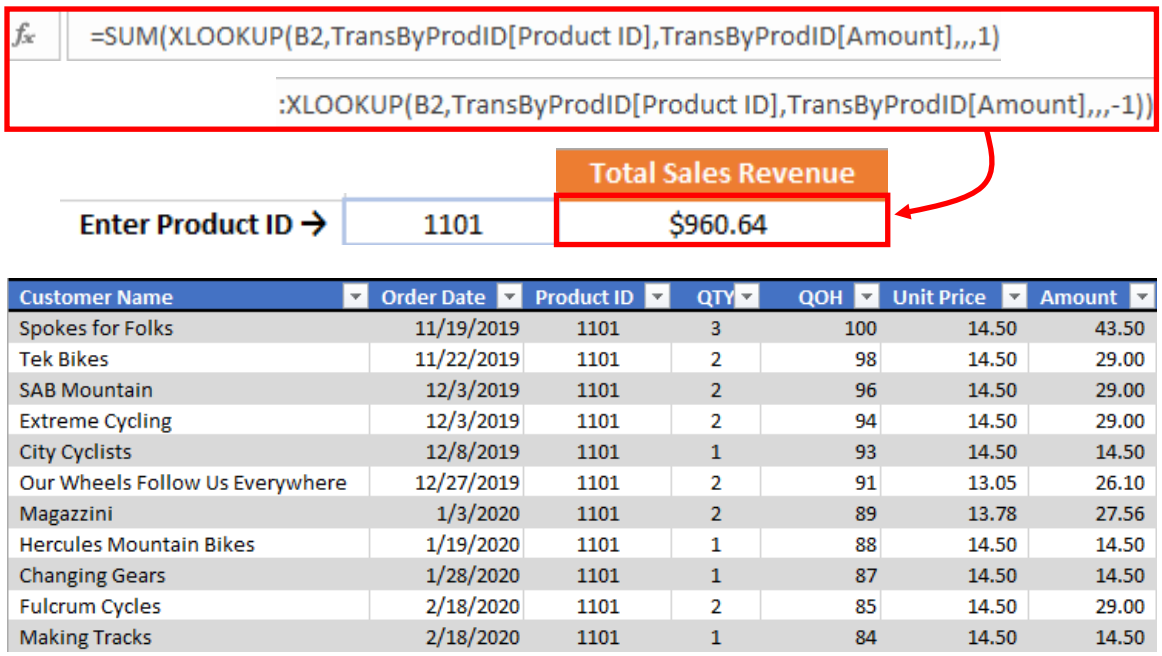
Item No	Vendor No	Description	UM	QOH	Price	Cost
C001	E1634	9 oz Aloe Vera Hand Cream	EA	2,400	6.99	2.80
C002	809834CF	9 oz Xtra Moisturizing Cream	EA	1,800	7.49	3.00
C003	E1636	16				
L001	N0019B	9 o				
L002	N00116B	16				
L003	N00124B	24				
L201	708901CF	9 o				
L202	N0029B	9 o				
L203	N00216B	16				
M101	N0039B	9 o				
M102	FA550009	9 o				
M201	FA550016	16				
M202	FA550024	24				

Vendor No	Vendor	Vendor Description
708901CF	Crème Francais	Lotion Francais, Extra Moisturizing Body, 9 oz
809834CF	Crème Francais	Cream Francais, Extra Moisturizing Hand, 9 oz
E1634	ELM Enterprises	ELM Cream, Aloe Vera, 9 oz
E1636	ELM Enterprises	ELM Cream, Hand and Body, 16 oz
FA550009	Facial Artistry LLC	Artistry Wrinkle Reducing Facial, 9 oz
FA550016	Facial Artistry LLC	Artistry Wrinkle Reducing Facial, 16 oz
FA550024	Facial Artistry LLC	Artistry Wrinkle Reducing Facial, 24 oz
N00116B	Nature's Botanicals Inc	Lotion, Organic Body, 16 oz
N00124B	Nature's Botanicals Inc	Lotion, Organic Body, 24 oz
N0019B	Nature's Botanicals Inc	Lotion, Organic Body, 9 oz
N00216B	Nature's Botanicals Inc	Lotion, Organic Hand and Body, 16 oz
N0029B	Nature's Botanicals Inc	Lotion, Organic Hand and Body, 9 oz
N0039B	Nature's Botanicals Inc	Face Mask, Organic, 9 oz

Figure 3 – XLOOKUP Can Look Across Worksheets or Workbooks

The formula shown in **Figure 3** performs a double lookup to return the value from the vendor list. It first uses XLOOKUP to find the vendor product number in the Inventory List, which is then input to XLOOKUP to return the vendor name from the Vendor List on another worksheet.

XLOOKUP can return values (as in the previous formula), arrays (ranges of values), or references (a reference to a cell or range of cells, such as B3 or A1:A10). Because XLOOKUP can search first-to-last or last-to-first, you can use it to identify the cell references of a specific product's first and last sales transactions. You can then use this result as input to other formulas, including functions such as SUM, COUNT, AVERAGE, MIN, or MAX. For example, **Figure 4** displays a formula that sums sales revenue by product ID from a transaction table.



Customer Name	Order Date	Product ID	QTY	QOH	Unit Price	Amount
Spokes for Folks	11/19/2019	1101	3	100	14.50	43.50
Tek Bikes	11/22/2019	1101	2	98	14.50	29.00
SAB Mountain	12/3/2019	1101	2	96	14.50	29.00
Extreme Cycling	12/3/2019	1101	2	94	14.50	29.00
City Cyclists	12/8/2019	1101	1	93	14.50	14.50
Our Wheels Follow Us Everywhere	12/27/2019	1101	2	91	13.05	26.10
Magazzini	1/3/2020	1101	2	89	13.78	27.56
Hercules Mountain Bikes	1/19/2020	1101	1	88	14.50	14.50
Changing Gears	1/28/2020	1101	1	87	14.50	14.50
Fulcrum Cycles	2/18/2020	1101	2	85	14.50	29.00
Making Tracks	2/18/2020	1101	1	84	14.50	14.50

Figure 4 – Using XLOOKUP to Sum Sales of a Specified Product ID

Move over VLOOKUP; your day in the sun is over! XLOOKUP is here and can do everything VLOOKUP can and more, revolutionizing how you use lookup functions. Both functions are easy to use, intuitive, and offer productivity-enhancing analytical power. Unfortunately, these features are available only in Excel Online and Microsoft 365 Excel, version 2001 and newer.

MATCH and INDEX

Of course, VLOOKUP and HLOOKUP can continue to meet most day-to-day lookup needs if you cannot access XLOOKUP. However, there are several situations where VLOOKUP and HLOOKUP are not sufficient.

- **The task requires a reverse lookup.** VLOOKUP requires the lookup column in any lookup array table to be the first or left-most column. What if you need to identify an item in a lookup column by looking up a value in an inside data column? This sort of reverse lookup is not possible with VLOOKUP.

- **The task requires a concatenated lookup.** VLOOKUP can only look up a value in a single column, the lookup column. However, what if the unique value to look up resides in two or more columns? For example, a nonprofit organization maintains a donor list containing names, addresses, total donations, and last donation date. The donor's last name is in the first column, and the donor's first name is in the second. Because some donors may have the same last name, you must perform lookups on both name columns. This sort of concatenated lookup is not possible with VLOOKUP.

You can use MATCH and **INDEX** together to overcome these problems with VLOOKUP. The **MATCH** function searches for a specified item in a range of cells and then returns the item's relative position. The **INDEX** function returns a value or reference to a value specified by its relative position in a range. In other words, MATCH is used to find an item, and INDEX is used to return the item's value.

The MATCH function uses the following syntax:

MATCH(lookup value, lookup array, [1, -1, 0])

where:

Lookup value represents the value to find.

Lookup array is the range or table column in which to search

Match type (optional) determines how to perform the search; defaults to 1

Option Number	Option Behavior
0	Exact match
1 or Omitted	Exact match or next smaller item; the lookup array must be sorted in ascending order (default)
-1	Exact match or next larger item; the lookup array must be sorted in descending order

The INDEX function uses the following syntax:

INDEX(array, row number, column number)

where:

Array is a range of cells or an array constant.

Row number selects the row in the array from which to return a value; if the row number is omitted, the column number is required.

Column number selects the column in the array from which to return a value; if the column number is omitted, the row number is required.

If you use row and column numbers, INDEX returns the cell's value at their intersection. If the row or column number is zero (0), INDEX returns the array of values for the entire column or row. Enter the INDEX function as an array formula to use values returned as an array.

Now that we understand the basic syntax of the MATCH and INDEX functions let us put them to work. In the first example, a merchandising company needs to look up a vendor part number using an item number *and* an item number using a vendor part number. If the lookup table contains the item number in the first column, VLOOKUP can retrieve a vendor part number based on an item number. However, it cannot do a reverse lookup, such as retrieving an item number based on a vendor part number. Therefore, we create a formula using MATCH and INDEX to execute the reverse lookup. Recall that MATCH finds an item's location in a lookup table, and INDEX returns the item's value. **Figure 5** illustrates the lookup application.

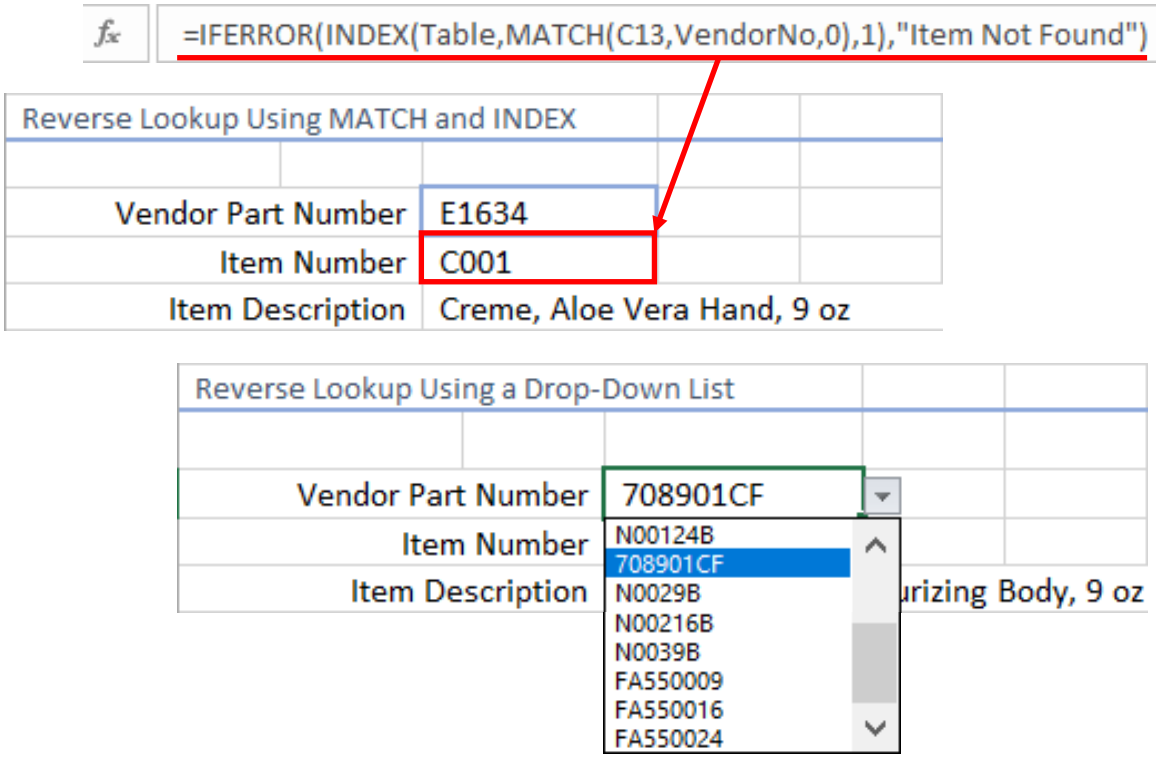
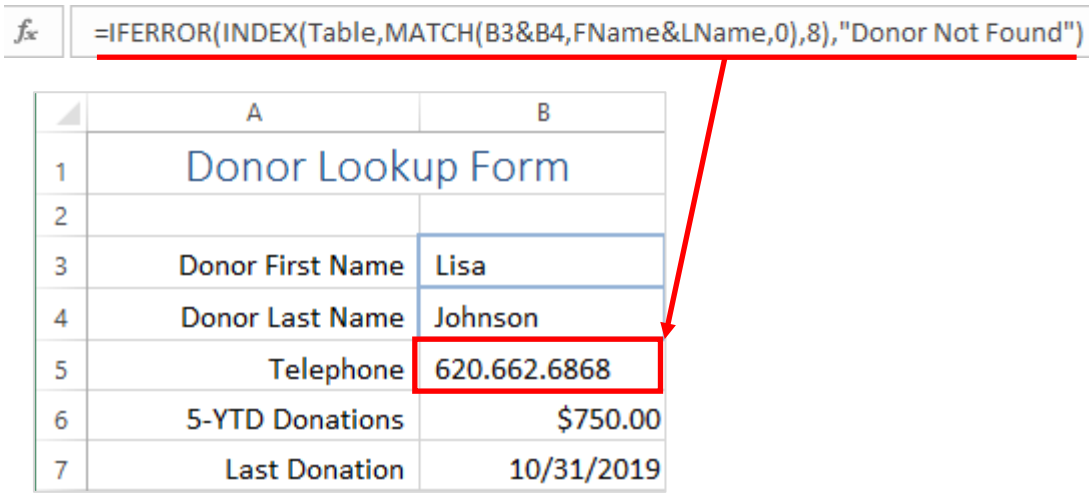


Figure 5 - Using MATCH and INDEX to Perform a Reverse Lookup

In the top example shown in Figure 5, users type in a vendor part number, and the formula returns the item number. In the bottom illustration, the same formula returns the item number. **Data Validation** creates a drop-down list from which the user selects the vendor part number. As a result, the menu is easier to use and reduces errors caused by inexact matches resulting from data entry errors. Additionally, note the use of **IFERROR** to trap errors.

In the following example, a nonprofit organization must look up donor information for a telephone fundraising campaign. The table that contains the donor information has separate columns for donor first names and last names. Therefore, the formula must concatenate the lookup's first and last name columns to ensure a unique record is selected. This action is done using the ampersand operator (**&**), the input cells, and concatenating the lookup columns, as shown in the formula in **Figure 6**.

In this case, the formula must be entered as an *array formula* because it tests the entire array of first and last name combinations. Press **ENTER** to enter the formula in dynamic array aware Excel (Microsoft 365 Versions 1912 or later). Press **CTRL + SHIFT + ENTER** in older Excel versions to complete the entry.



Title	FirstName	LastName	Address	City	State	ZipCode	Telephone
Mr	Lawrence	McClelland	1413 Independence Dr	Slidell	LA	70458	985.646.6855
Ms	Mary	McClelland	3 Harbor Blvd	Slidell	LA	70460	985.644.1298
Dr	Rodney	Hesson	618 Dale DR	Slidell	LA	70461	985.455.6363
Mr	Thomas	Stephens	55 Meadowlark Ln	Norcross	GA	32456	770.777.3434
Ms	Joan	Wiley	763 Maine Ave	Red Bank	NJ	10256	212.655.3233
Ms	Lisa	Johnson	21 Charter Oak Ave	Wichita	KS	45231	620.662.6868
Ms	Harriet	Johnson	13562 Hiway 16 East	Pierre	SD	67675	512.523.3099
Mr	Bob	Spencer	33 Gulf Shore Road	Pensacola	FL	21555	801.782.6320

FNAME **LNAME**

Figure 6 - Concatenating Input Cells to Perform a MATCH Lookup

IPMT, PPMT, and STOCKHISTORY

Excel offers over 50 financial functions to assist users in creating financial calculations. These functions span three broad categories – 1) investments, 2) depreciation, and 3) securities. For example, many accounting and financial professionals use the **PMT** function to calculate the periodic principal and interest payment given a loan's principal, interest rate, and term. However, Excel also has functions for computing the amount of interest and principal reduction for a selected payment. Instead of referring to an amortization schedule for these amounts, you can calculate them directly with the **IPMT** and **PPMT** functions.

IPMT and PPMT

IPMT uses the following syntax:

IPMT(rate per period, selected period, number of periods in the loan, loan amount, [future value], [type: ordinary annuity = 0, annuity due = 1])

PPMT uses the following syntax:

PPMT(rate per period, selected period, number of periods in the loan, loan amount, [future value], [type: ordinary annuity = 0, annuity due = 1])

Figure 7 presents calculations of the interest amount and principal reduction associated with a selected payment on a loan.

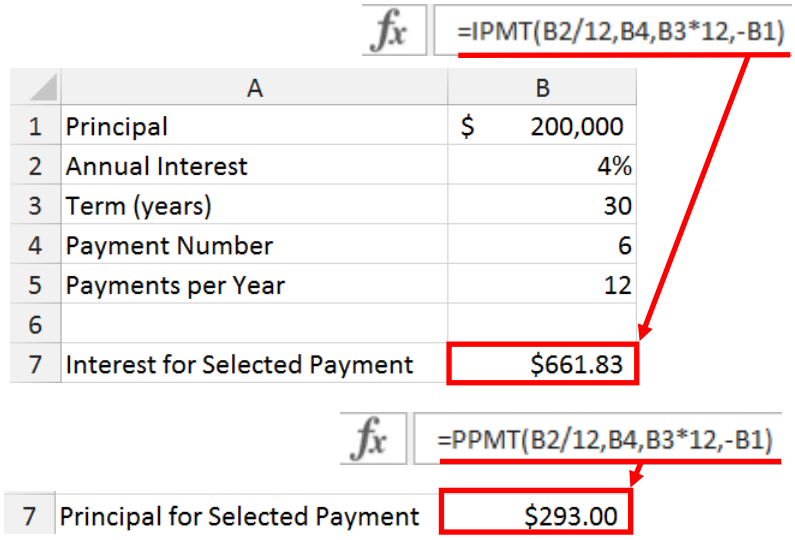


Figure 7 - Using Excel's IPMT and PPMT Functions

With dynamic arrays, Microsoft 365 subscribers can expand PMT, IPMT, and PPMT functionality. While the **IPMT** function allows you to compute the interest component of any single payment, dynamic arrays facilitate calculating the total interest paid over an extended period, such as a specific reporting year, with a single formula. For example, **Figure 8** shows a simple template that uses **SEQUENCE** to calculate a loan's annual interest.

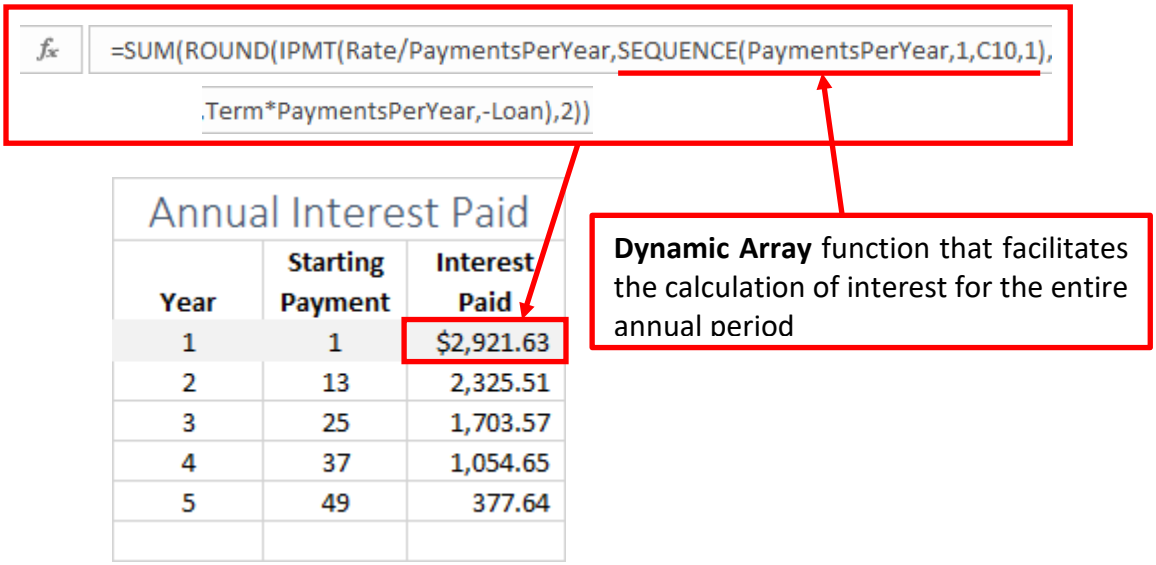


Figure 8 – Using a Dynamic Array Function to Calculate Annual Interest Expense

Other dynamic array functions available to Microsoft 365 subscribers include **UNIQUE**, **SORT**, **SORTBY**, **FILTER**, and **RANDARRAY**. In addition, with the release of dynamic arrays, users may see an **@** symbol in a formula that identifies a specific value in a returned array, referred to as an *implicit intersection*. Do not be concerned if an **@** symbol appears in a formula; it calculates the same way it always did.

STOCKHISTORY

Excel's long-awaited **STOCKHISTORY** function is now available in subscription-based versions of Excel. Unfortunately, this function is not available in Excel 2016 and Excel 2019.

This feature can retrieve historical stock prices by entering a few variables into a formula. Moreover, you can retrieve values for a single date or a range of dates. Further, if you choose a range of dates, you can designate *daily*, *weekly*, or *monthly* intervals. STOCKHISTORY displays the date and closing price by default. However, if desired, you can show the opening price, high price, low price, and volume.

Using STOCKHISTORY

The syntax for using the STOCKHISTORY function can be relatively simple, as indicated below. However, note that only the *stock* and *start_date* are required. Thus, a formula using STOCKHISTORY could be as simple as **=STOCKHISTORY("MSFT","1/29/2021")**. Of course, this formula returns the closing price for a share of Microsoft stock on January 29, 2021.

Additionally, you can create more sophisticated formulas using STOCKHISTORY if your needs require additional information. Specifically, the full syntax of a formula can include all the following items.

STOCKHISTORY(stock, start_date, [end_date],[interval],[headers], [property0], [property1] [property2], [property3], [property4], [property5])

- **stock**: The identifier for the financial instrument targeted. This reference can be a ticker symbol or a stock's data type.
- **start_date**: The earliest date for which you want information.
- **end_date** (optional): The latest date for which you want information.
- **interval** (optional): Daily (0), Weekly (1), or Monthly (2) interval options for data
- **headers** (optional): Specifies if the formula returns additional header rows with the array.
- **property0 – property5** (optional): Specifies which information to include in the result: Date (0), Close (1), Open (2), High (3), Low (4), Volume (5).

Extending the previous example, we can create more powerful formulas that use the function. For instance, we can use the following formula to generate a listing of closing prices for a range of dates:

=STOCKHISTORY("MSFT","1/1/2020","12/31/2020")

To illustrate, **Figure 9** below provides an abbreviated set of results from the above formula.

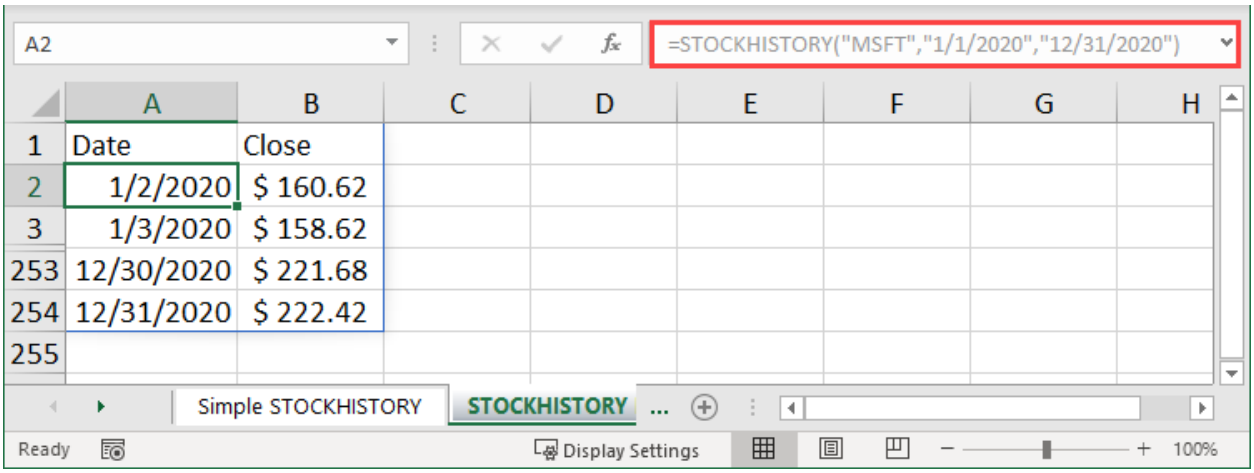


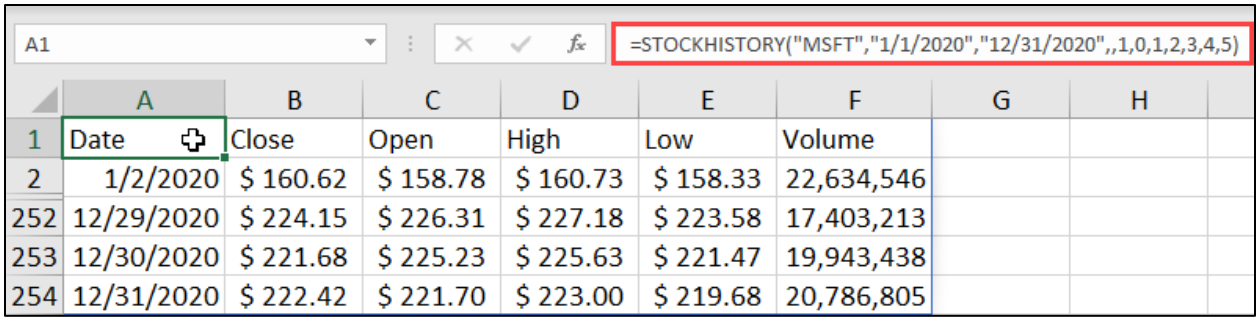
Figure 9 - Sample Results Using STOCKHISTORY

A Deeper Dive into STOCKHISTORY

The following example of the new function incorporates optional arguments to add columns for each trading interval for the Open price, High price, Low price, and Volume.

=STOCKHISTORY("MSFT", "1/1/2020", "12/31/2020",,1,0,1,2,3,4,5)

Figure 10 illustrates an abbreviated set of results using this formula.



	A	B	C	D	E	F	G	H
1	Date	Close	Open	High	Low	Volume		
2	1/2/2020	$ 160.62	$ 158.78	$ 160.73	$ 158.33	22,634,546		
252	12/29/2020	$ 224.15	$ 226.31	$ 227.18	$ 223.58	17,403,213		
253	12/30/2020	$ 221.68	$ 225.23	$ 225.63	$ 221.47	19,943,438		
254	12/31/2020	$ 222.42	$ 221.70	$ 223.00	$ 219.68	20,786,805		

Figure 10 - STOCKHISTORY Example with Optional Arguments

Of course, you can use STOCKHISTORY results in the same fashion as if you entered the data manually.

Summarizing STOCKHISTORY

In short, STOCKHISTORY is one of the most widely-anticipated functions added to Excel in recent years. If you have a subscription-based version of Excel, you should already have access to this feature or receive access soon. Importantly, you can use STOCKHISTORY to retrieve stocks' historical prices and incorporate them into other calculations in your spreadsheets. Therefore, the next time you research a stock's historical data, consider using STOCKHISTORY. If you do, you will reduce the time you spend retrieving data.

Array Formulas

An ***array*** is a collection or range of two or more cells. 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 calculations in a single cell and return the results to one or multiple cells. Note that the results of an array formula can be embedded within other formulas, as was demonstrated when calculating interest for an entire year with IPMT. The bottom line is that array formulas can perform calculations that are impossible with traditional formulas.

You can create array formulas using ordinary functions that take individual cells as their values. To complete the entry, you must construct this array formula using a **CTRL + SHIFT + ENTER** sequence in non-dynamic array-aware versions of Excel. Excel also includes functions that operate on arrays natively, such as **SUMPRODUCT**, **AGGREGATE**, **LOOKUP**, **INDEX** and **SUMIFS**, **COUNTIFS**, **AVERAGEIFS**, **MINIFS**, and **MAXIFS**. These functions produce array formulas *without* entering CTRL + SHIFT + ENTER. Another set of functions generates resultant arrays or multi-cell results. Among them are **TRANSPOSE**, **TREND**, **FREQUENCY**, **LINEST**, and **LOGEST**.

Simple Array Formulas

In this first example, three methods – 1) conventional analysis using a helper column, 2) an array formula using the SUM function, and 3) an array formula using SUMPRODUCT – will be used to calculate the total billings of a project team for the week as shown in **Figure 11**.

	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 11 – Team Timesheet

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

2. Build the following array formula: **=SUM(B2:B5*C2:C5) in cell D9** and press **Enter**. In *dynamic array aware versions of Excel*, the formula will calculate correctly. However, in *non-dynamic array versions of Excel*, the formula evaluates to **#VALUE!**. Therefore, in non-dynamic aware versions of Excel, you must press **CTRL + SHIFT + ENTER** to enter the formula. Note the braces **{ }** surrounding the formula in the formula bar. Braces **{ }** no longer appear in the formula bar of dynamic array aware versions of Excel.

3. Use the SUMPRODUCT function to build the formula. For example, 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 the complexities of an ordinary array formula. It can handle up to 255 arrays of data.

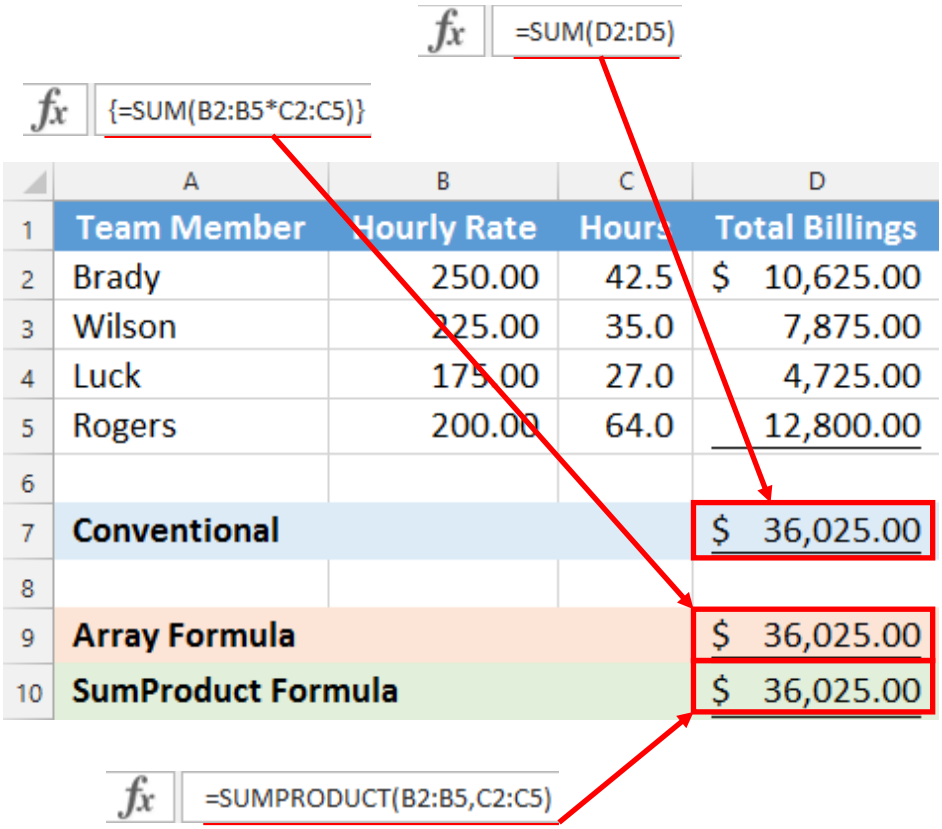


Figure 12 – Conventional vs. Array Formulas

Note that these three methods returned the same result, but the array formulas used a single cell to calculate. In the simple context presented, the most commonly used technique with a helper column works, as well as the array formulas. However, calculating the extended price or cost of goods to calculate the total revenue or cost of goods sold would be cumbersome and time-consuming in situations involving thousands of transactions. A single-cell array formula would be a better alternative in those situations. 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 array formulas entered using CTRL + SHIFT + ENTER.

Before leaving this example, let us use an advanced technique to examine or troubleshoot array formulas. Position the cursor in cell **D9**, the cell containing the array formula. Press **F2** to edit the formula, and then click ***number 1*** in the **Screen Tip** that appears to select the array reference. Next, press **F9** to see the underlying array, as shown in **Figure 13**. Do *not* press **Enter** without first pressing **CTRL + Z** to undo the change. Otherwise, Excel replaces the cell references with the *array constant*. As a result, the formula no longer calculates by reference to the cells' values. Instead, it will calculate by referencing the SUM function's values, thereby breaking the formula as created.

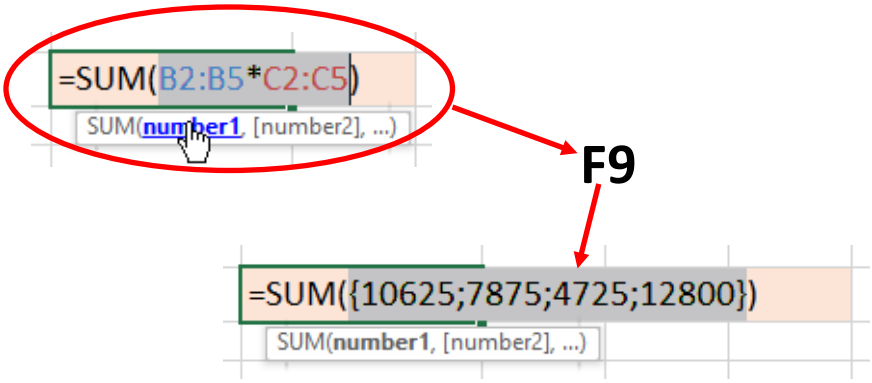


Figure 13 – Troubleshooting an Array Formula

You can also examine the formula using the SUMPRODUCT function. Note that you can view the individual items in the arrays rather than the results of the array calculations displayed in the ordinary array formula.

Rounding in a Total

This example uses an array formula to round the results of a formula. Many accounting professionals have faced the need to sum many amounts resulting from floating-point calculations. The values may result from formulas or flow from data imported from other systems. A problem arises when the amounts do not appear to correctly sum because Excel uses the *underlying* values rather than the *displayed* values in calculating formulas, including summary totals. Conventional practice requires using Excel's **ROUND** function to round all the computed amounts before summing their values. However, that solution is cumbersome and time-consuming when many values are present. A better alternative would be to build a formula to round and sum the amounts in a single calculation. A simple array formula like the one below can quickly and easily perform this task.

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

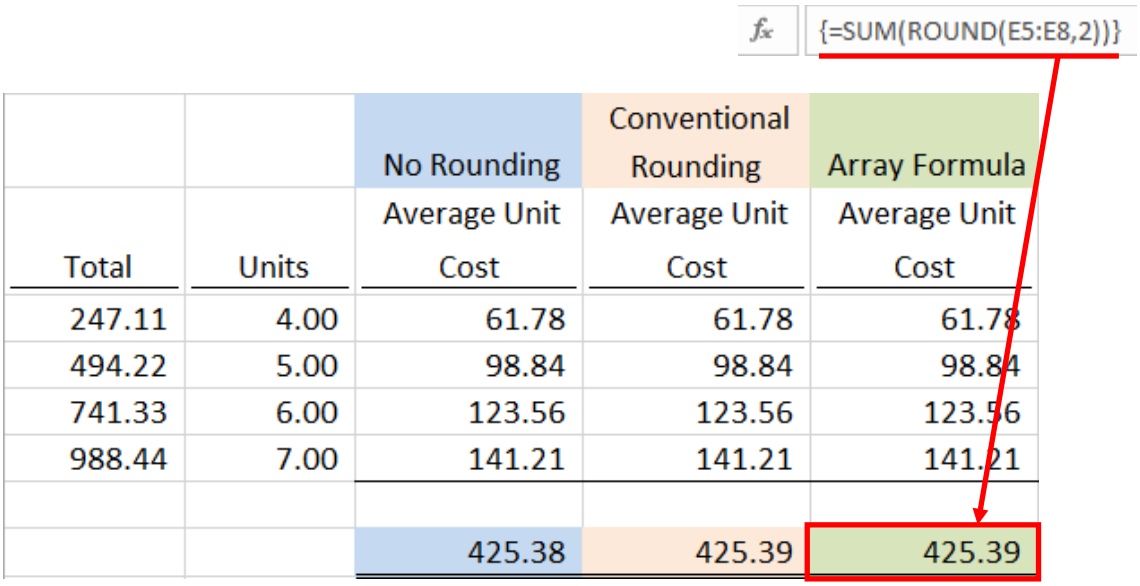


Figure 14 - Using an Array Formula to Round and Sum Data in a Total

Again, let us use the technique discussed earlier to examine the array formula. First, 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. Next, press **CTRL + Z** to undo the changes and 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.

Dynamic Arrays – New Forms of Array Power

You can access even more powerful array-related features using Excel version 1907 or newer through a Microsoft 365 subscription. Specifically, you can use *dynamic arrays* and six new functions to work more efficiently. Further, if you are in a dynamic array aware Excel version, you no longer need to ensure braces surround your array formulas. Instead, you can press the **Enter** key when entering an array formula and not worry about the **CTRL** + **SHIFT** + **ENTER** keyboard sequence.

A dynamic array is a range of data that automatically resizes as users add or delete data. But do not confuse a dynamic array with a table, for these two features are decidedly different. Whereas a table can serve as a dynamically resizing range of data, it cannot do everything a dynamic array can. For example, you can use dynamic arrays to sort or filter data without disturbing the original data, which is impossible with tables. On the other hand, dynamic arrays cannot do everything tables can. To illustrate, you cannot create a data model from multiple dynamic arrays.

As shown below, dynamic arrays allow us to break free from the "one-cell, one formula" mentality that has always existed in spreadsheets. Before dynamic arrays, we had to enter formulas into every cell where we wanted calculation results to display. Dynamic arrays change

that by allowing us to create a single formula, and its results will appear in as many cells as necessary, given the volume of data under consideration. To illustrate, consider the example provided in **Figure 15**. In this example, the user entered the formula in the formula bar *into cell D2 only*. However, the formula *dynamically copied* itself so that its results list all the unique values in the range.

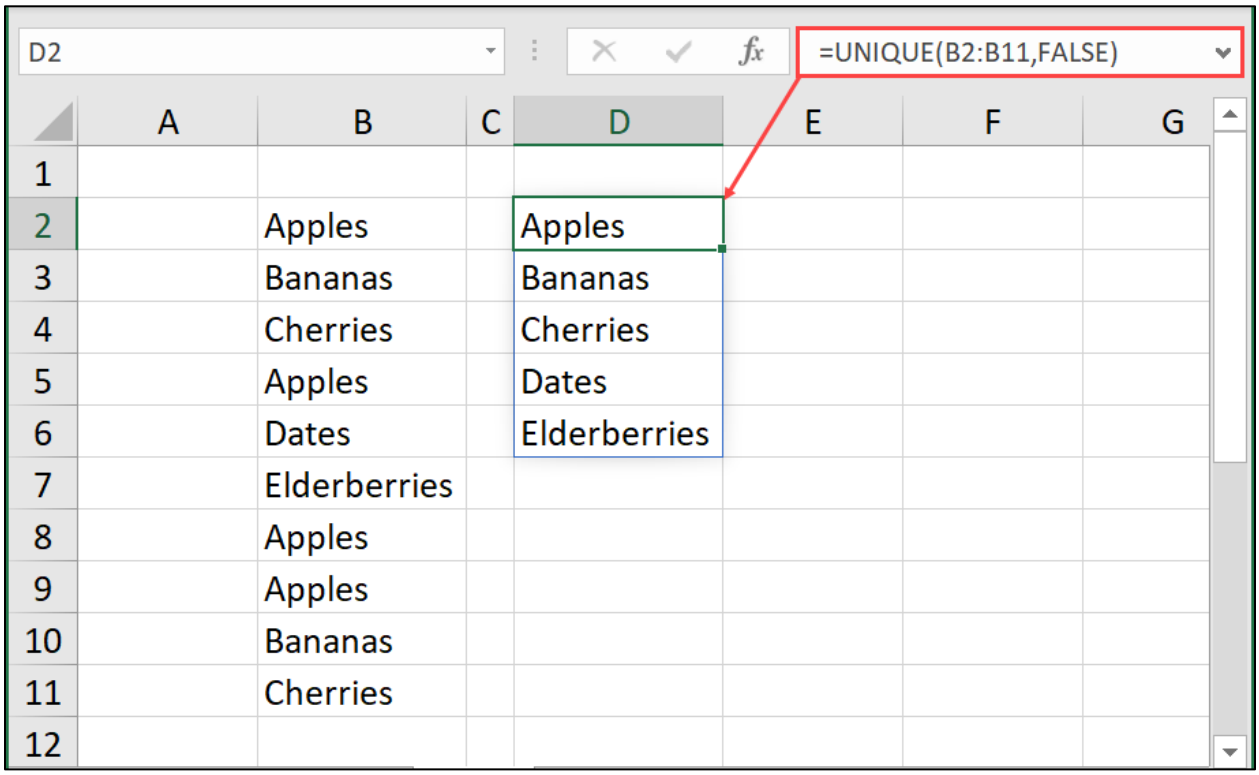


Figure 15 – First Example of a Formula Based on a Dynamic Array

In the example presented, note that cells B2 through B11 are a traditional range of data. However, we could have used the dynamic array formula with a table supplying the data. The **UNIQUE** function illustrated in Figure 15 is one of six new functions you can use in a dynamic array-aware Excel version. As its name implies, UNIQUE extracts all the unique entries from an array. Following is a listing of the other five functions.

1. **FILTER** filters a dynamic array based on criteria specified in the formula. Neither FILTER nor any other dynamic array formulas alter the source data. Instead, each formula displays its results as a separate output range.

2. You can use **SORT** to arrange data in a dynamic array in ascending or descending order without disturbing the source data's arrangement.

3. **SORTBY** sorts based on values in a corresponding range or array.

4. **RANDARRAY** returns an array of random numbers.

5. You can use the **SEQUENCE** function to generate a listing of sequential numbers displayed in an array.

Note that each of the six functions outlined above works only in the context of dynamic arrays.

To provide an example of how you can use one of the new functions on a dynamic array, consider the example in **Figure 16**. The left side of the image shows a table named "Data." A user must sort the data in that table in descending order based on the "Total" column values. However, she does not want to change the data's sort order in the source table. Instead, she enters the following simple formula into the worksheet to achieve the objective.

=SORT(Data,5,-1)

The formula sorts the dynamic array, known as *Data*, on the *fifth* column, in descending order (*-1*). Equally important, it left the original data set unchanged.

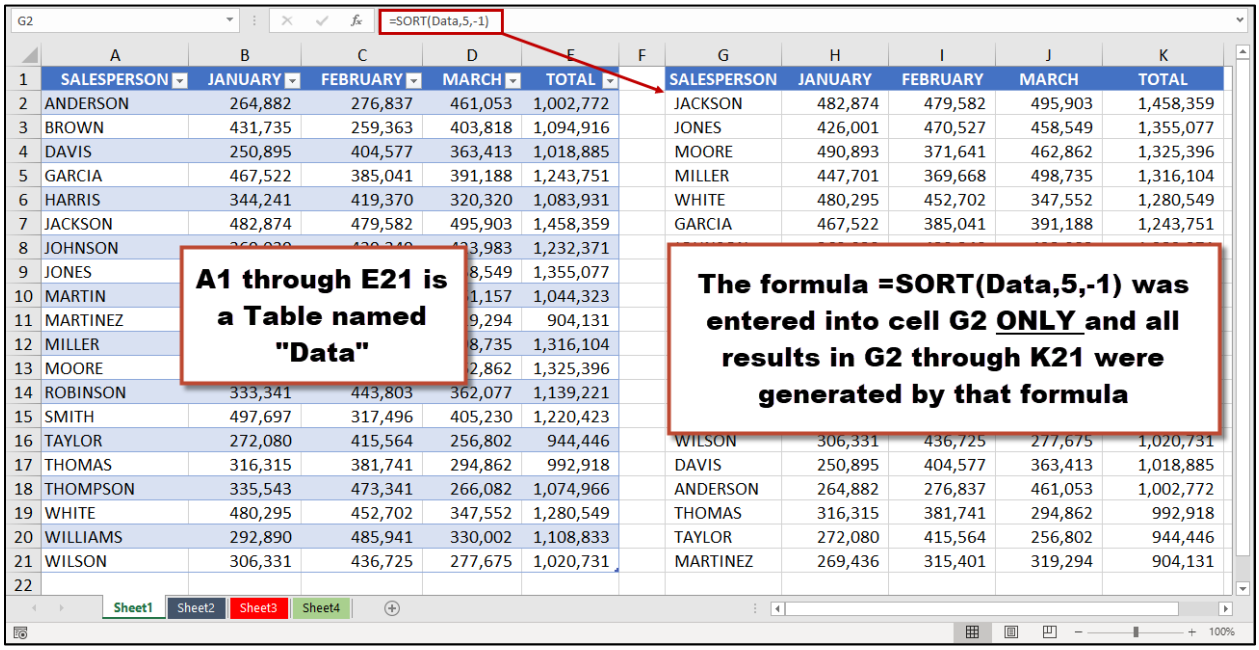


Figure 16 – Sorting a Dynamic Array with the SORT Function

Dynamic array formulas remain a relatively new and obscure feature in Excel. However, as more users become aware of their power and ease of use, their popularity will skyrocket. Moreover, suppose you are using a dynamic array aware version of Excel. In that case, you can use legacy array formulas without worrying about the **CTRL** + **SHIFT** + **ENTER** keystroke sequence to enter your array formulas.

SUMIFS, SUBTOTAL, and AGGREGATE

Many clients and co-workers think accountants are mathematicians, but most of our reports and analyses require little more than simple arithmetic. We add and subtract and sometimes use multiplication or division. We often use Excel's SUM function, but other functions may be better choices in some circumstances. For example, **SUMIFS** is useful when adding amounts that meet some specified criteria. **SUBTOTAL** helps add values in a range while ignoring those on hidden rows, and **AGGREGATE** can ignore errors and hidden data in the sum range. AGGREGATE also provides more sub-functions than SUBTOTAL, making AGGREGATE a very flexible tool.

SUMIFS

Accountants and business professionals can use SUMIFS in many contexts. For example, SUMIFS can summarize sales data by product and region, as shown in **Figure 17**.

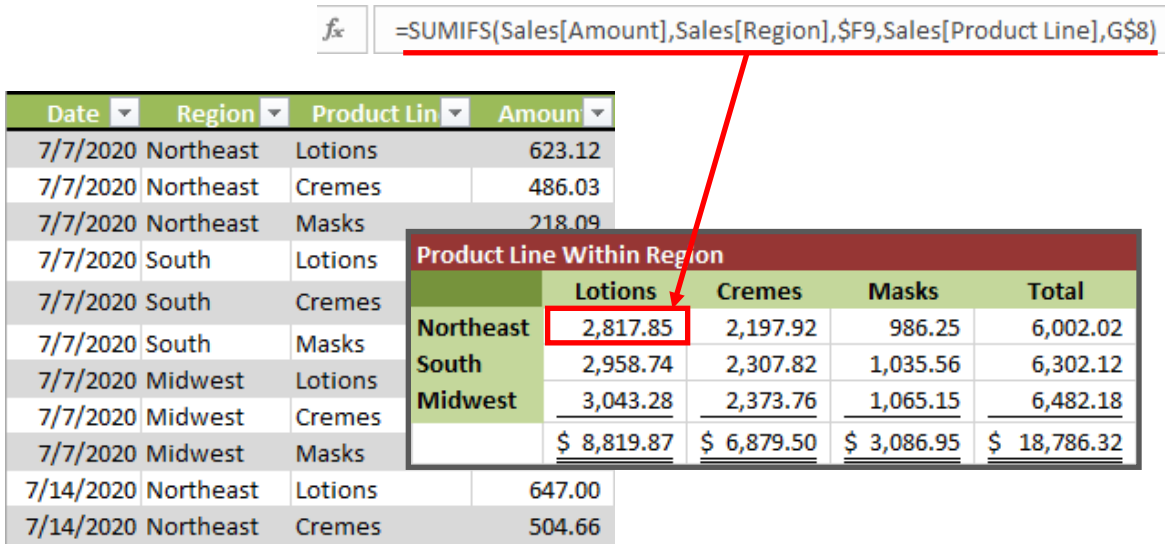


Figure 17 – Using SUMIFS to Sum Sales by Product within Region

You can also use SUMIFS to sum data within a user-defined date range. To illustrate, consider the data and analysis in **Figure 18.** SUMIFS adds data within the beginning and end dates specified in cells **F3** and **F4**. Note how the operators append to the cells' contents containing the dates, creating a *composed formula*. In other words, the formula is composed, in part, by adding the operators to the criteria cell references. Note that formula composition is often necessary when using comparison operators (**=**, **>**, **<**, **>=**, **<=**, **<>**) in Excel.

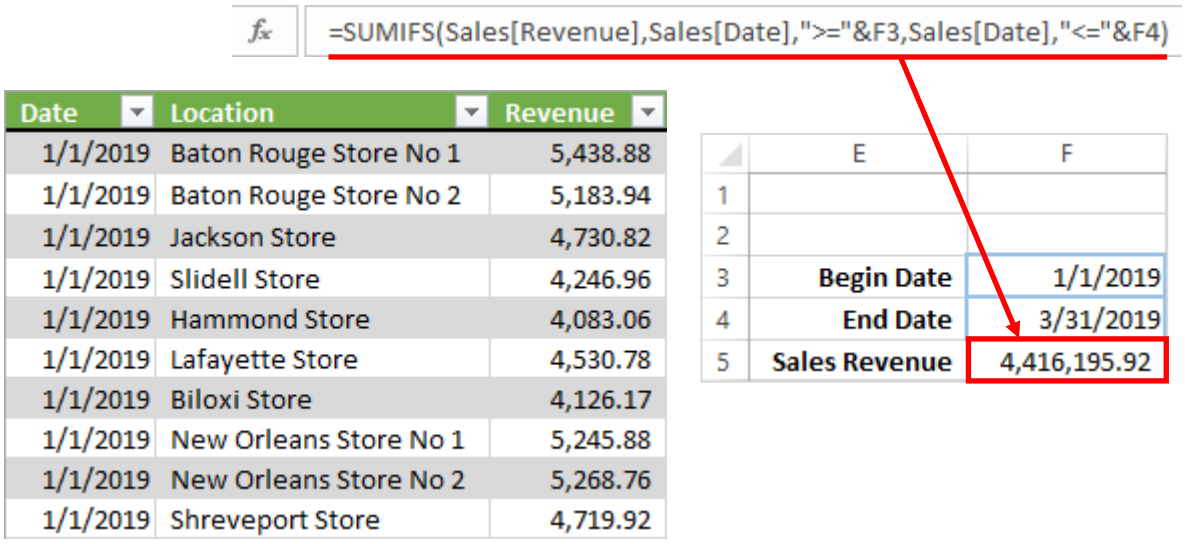


Figure 18 - Sample Dataset Used in SUMIFS Formula

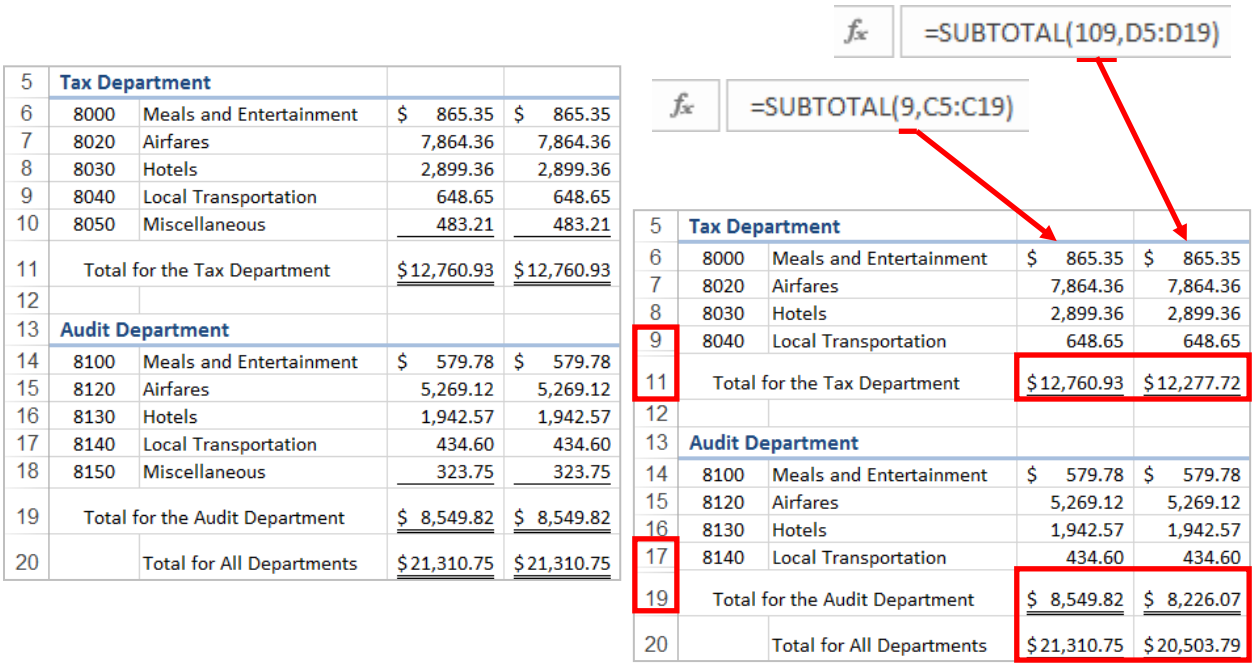
SUBTOTAL

Many accounting workbooks have multiple levels of subtotals that ultimately summarize data into a grand total. The **SUBTOTAL** function makes this process easier and less prone to error, especially when using big datasets with many subtotals. The SUBTOTAL function ignores nested subtotals (subtotals within subtotals) to avoid double-counting values. As a result, grand totals use detailed data points instead of intermediate (nested) subtotals. The SUBTOTAL function can summarize columns of data using any of the twenty-two sub-functions identified in the following table. The sub-functions in the first column include values on hidden rows within the subtotal range in the calculation. The sub-functions in the second column ignore values on hidden rows.

	Sub-Function Number	
Sub-Function	**Includes Hidden Values**	**Ignores Hidden Values**
AVERAGE	1	101
COUNT	2	102
COUNTA	3	103
MAX	4	104
MIN	5	105
PRODUCT	6	106
STDEV.S	7	107
STDEV.P	8	108
SUM	9	109
VAR.S	10	110
VAR.P	11	111

Filters alter the rules of the SUBTOTAL function. For example, summarizations calculated on filtered data *always exclude values on hidden rows* regardless of the sub-function used. In other words, functions 1 and 101 do not include values on hidden rows when calculating subtotals. Further, the **AutoSum** button's operation – accessible on the Ribbon's Home tab – changes when a filter is active. AutoSum inserts a formula based on the SUBTOTAL function when a filter is in place instead of using simple functions, such as SUM, AVERAGE, or COUNT.

Figure 19 uses the SUBTOTAL function to produce subtotals on rows 11 and 19 and a grand total on row 20. The formulas in the left column use sub-function 9 in the SUBTOTAL and include hidden values in the subtotals. The formulas in the right column use sub-function 109 in the SUBTOTAL function. These formulas do not contain hidden values in the subtotals. If you use sub-function 109, creating a report excluding Miscellaneous Expenses is simple. Just hide the two miscellaneous expense rows; the subtotals and grand total calculations immediately reflect their absence.



5	**Tax Department**			
6	8000	Meals and Entertainment	$ 865.35	$ 865.35
7	8020	Airfares	7,864.36	7,864.36
8	8030	Hotels	2,899.36	2,899.36
9	8040	Local Transportation	648.65	648.65
10	8050	Miscellaneous	483.21	483.21
11	Total for the Tax Department		$12,760.93	$12,760.93
12				
13	**Audit Department**			
14	8100	Meals and Entertainment	$ 579.78	$ 579.78
15	8120	Airfares	5,269.12	5,269.12
16	8130	Hotels	1,942.57	1,942.57
17	8140	Local Transportation	434.60	434.60
18	8150	Miscellaneous	323.75	323.75
19	Total for the Audit Department		$ 8,549.82	$ 8,549.82
20		Total for All Departments	$21,310.75	$21,310.75

5	**Tax Department**			
6	8000	Meals and Entertainment	$ 865.35	$ 865.35
7	8020	Airfares	7,864.36	7,864.36
8	8030	Hotels	2,899.36	2,899.36
9	8040	Local Transportation	648.65	648.65
11	Total for the Tax Department		$12,760.93	$12,277.72
12				
13	**Audit Department**			
14	8100	Meals and Entertainment	$ 579.78	$ 579.78
15	8120	Airfares	5,269.12	5,269.12
16	8130	Hotels	1,942.57	1,942.57
17	8140	Local Transportation	434.60	434.60
19	Total for the Audit Department		$ 8,549.82	$ 8,226.07
20		Total for All Departments	$21,310.75	$20,503.79

Figure 19 - Using SUBTOTAL to Automate the Totaling Process

AGGREGATE

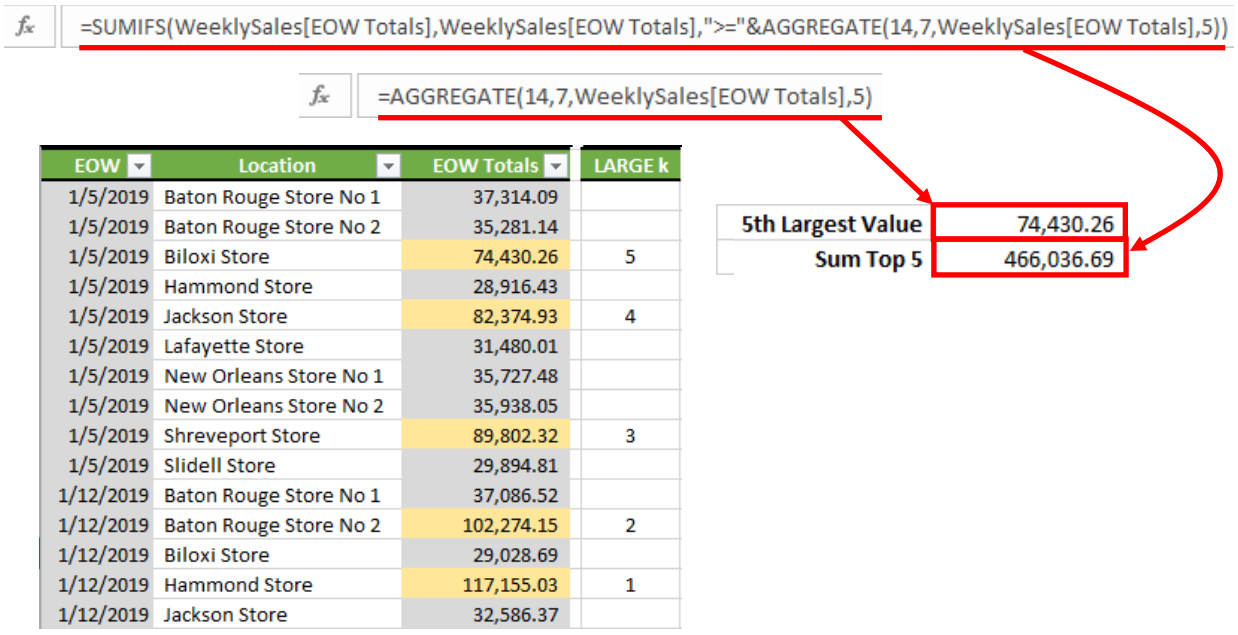
You can describe the **AGGREGATE** function as "SUBTOTAL on steroids." The move to AGGREGATE is easy and intuitive for those already using SUBTOTAL. AGGREGATE includes more sub-functions, such as *Median*, *Mode*, *Large*, *Small*, *Percentile*, and *Quartile*. Further, AGGREGATE can ignore nested summaries produced with AGGREGATE or SUBTOTAL, hidden rows, or *errors*. Users must specify the types of values to ignore. Some sub-functions operate on cell references, while others work on arrays, as identified in the following table.

Reference Sub-Functions	Sub-Function Number	Array Sub-Functions	Sub-Function Number
AVERAGE	1	MEDIAN	12
COUNT	2	MODE.SNGL	13
COUNTA	3	LARGE	14
MAX	4	SMALL	15
MIN	5	PERCENTILE.INC	16
PRODUCT	6	QUARTILE.INC	17
STDEV.S	7	PERCENTILE.EXC	18
STDEV.P	8	QUARTILE.EXC	19
SUM	9		
VAR.S	10		
VARP.P	11		

The *array sub-functions* require the specification of the "**k**" element. For example, if you wanted the third-largest value returned from a referenced range, the sub-function would be **14** (LARGE), and **k** would be **3**. The options to ignore nested SUBTOTAL and AGGREGATE functions, hidden rows, error values, or some combination of these options appear in the table below. In default, the AGGREGATE function ignores nested SUBTOTAL and AGGREGATE functions in the referenced range.

Option Number	Option Behavior
0 or Omitted	Ignore nested SUBTOTAL and AGGREGATE functions
1	Ignore hidden rows, nested SUBTOTAL and AGGREGATE functions
2	Ignore error values, nested SUBTOTAL and AGGREGATE functions
3	Ignore hidden rows, error values, nested SUBTOTAL and AGGREGATE functions
4	Ignore nothing
5	Ignore hidden rows
6	Ignore error values
7	Ignore hidden rows and error values

The worksheet in **Figure 20** uses AGGREGATE and SUMIFS to sum the top five weekly sales results. AGGREGATE identifies the fifth-largest sales result, which is then used as the criteria in SUMIFS to add all sales values greater than or equal to the fifth-largest result. Similarly, users could sum the top or bottom ten or twenty results or all weekly sales in an identified quartile or percentile. AGGREGATE is also very useful in analyses that generate errors because of its ability to ignore them when summarizing data.



EOW	Location	EOW Totals	LARGE k
1/5/2019	Baton Rouge Store No 1	37,314.09	
1/5/2019	Baton Rouge Store No 2	35,281.14	
1/5/2019	Biloxi Store	74,430.26	5
1/5/2019	Hammond Store	28,916.43	
1/5/2019	Jackson Store	82,374.93	4
1/5/2019	Lafayette Store	31,480.01	
1/5/2019	New Orleans Store No 1	35,727.48	
1/5/2019	New Orleans Store No 2	35,938.05	
1/5/2019	Shreveport Store	89,802.32	3
1/5/2019	Slidell Store	29,894.81	
1/12/2019	Baton Rouge Store No 1	37,086.52	
1/12/2019	Baton Rouge Store No 2	102,274.15	2
1/12/2019	Biloxi Store	29,028.69	
1/12/2019	Hammond Store	117,155.03	1
1/12/2019	Jackson Store	32,586.37	

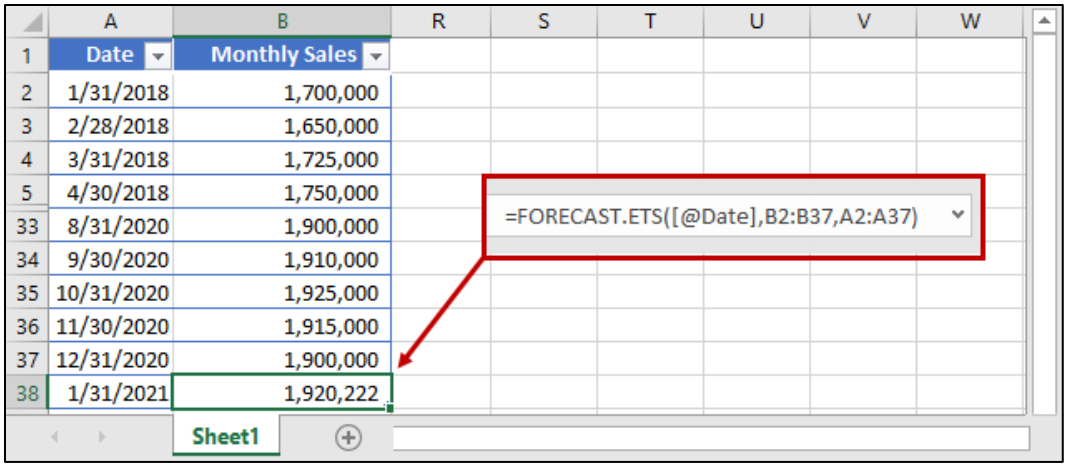
Figure 20 – Using AGGREGATE with SUMIFS to SUM the TOP 5 Sales Results

FORECAST.ETS and Forecast Sheet

Looking back at features added with the 2016 release of Excel, two enhancements stand out for their usefulness and effectiveness, yet these tools remain underutilized today. Specifically, the **FORECAST.ETS** function and **Forecast Sheet** feature provide improvements when creating forecasts based on historical data that includes seasonality.

FORECAST.ETS

For those who engage in budgeting and forecasting activities, the **FORECAST.ETS** function added to Excel 2016 is noteworthy. Before this function, the **FORECAST** function provided only linear forecasting capabilities. With FORECAST.ETS, you can now generate forecasts with exponential smoothing. The primary benefit of using FORECAST.ETS is that its exponential smoothing capacity allows you to create predictions that account for seasonality in your data. To illustrate, consider the data set pictured in **Figure 21** (with multiple rows hidden for presentation purposes). In that data set, a formula that uses the FORECAST.ETS function is computing forecasted sales for January 2021. Because this formula uses the FORECAST.ETS function, instead of the FORECAST function, it accounts for seasonality in the data.

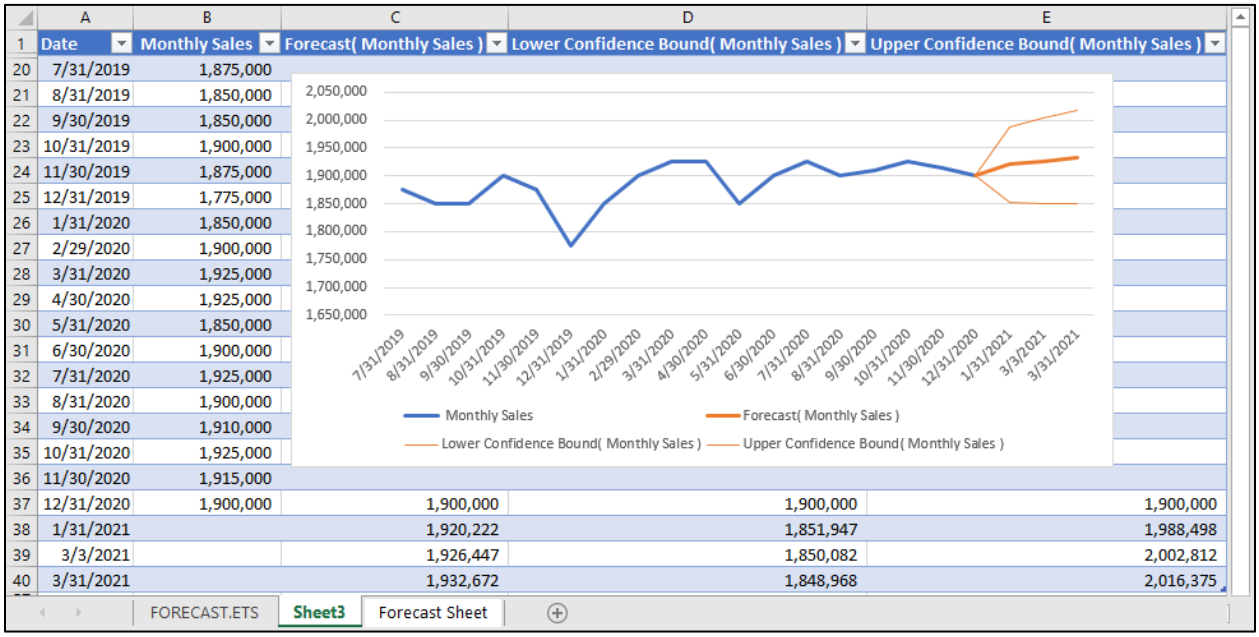


	A	B
1	Date	Monthly Sales
2	1/31/2018	1,700,000
3	2/28/2018	1,650,000
4	3/31/2018	1,725,000
5	4/30/2018	1,750,000
33	8/31/2020	1,900,000
34	9/30/2020	1,910,000
35	10/31/2020	1,925,000
36	11/30/2020	1,915,000
37	12/31/2020	1,900,000
38	1/31/2021	1,920,222

Figure 21 - Using FORECAST.ETS to Account for Seasonality in a Forecast

Forecast Sheet

Similar in some respects to FORECAST.ETS, Excel's **Forecast Sheets** tool creates forecasted values that account for seasonality, but it also can provide confidence boundaries around the predicted values. Additionally, you can use Forecast Sheets to create forecasted values for multiple future periods. Using the same data as the previous example, to create a Forecast Sheet, select the data you wish to use in your forecast. Next, click **Forecast Sheet** on the Ribbon's **Data** tab. In the resulting dialog box, make any adjustments necessary for items such as periods and confidence level, and click **Create** to build the Forecast Sheet. Upon doing so, your Forecast Sheet will appear, similar to that shown in **Figure 22.**



	A	B	C	D	E
1	Date	Monthly Sales	Forecast(Monthly Sales)	Lower Confidence Bound(Monthly Sales)	Upper Confidence Bound(Monthly Sales)
20	7/31/2019	1,875,000			
21	8/31/2019	1,850,000			
22	9/30/2019	1,850,000			
23	10/31/2019	1,900,000			
24	11/30/2019	1,875,000			
25	12/31/2019	1,775,000			
26	1/31/2020	1,850,000			
27	2/29/2020	1,900,000			
28	3/31/2020	1,925,000			
29	4/30/2020	1,925,000			
30	5/31/2020	1,850,000			
31	6/30/2020	1,900,000			
32	7/31/2020	1,925,000			
33	8/31/2020	1,900,000			
34	9/30/2020	1,910,000			
35	10/31/2020	1,925,000			
36	11/30/2020	1,915,000			
37	12/31/2020	1,900,000	1,900,000	1,900,000	1,900,000
38	1/31/2021		1,920,222	1,851,947	1,988,498
39	3/3/2021		1,926,447	1,850,082	2,002,812
40	3/31/2021		1,932,672	1,848,968	2,016,375

Figure 22 - Completed Forecast Sheet

The above example shows the three forecasted periods' upper and lower confidence boundaries. Further, see that the output provided is both graphical and tabular, helping all consumers of this data understand the meaning behind the numbers.

Conditional and Boolean Functions

IF

The **IF** function makes calculations when specified conditions exist. Excel supports up to sixty-four (64) nested IFs in a formula. The syntax of the IF function follows. Note that the true or false values returned are optional, but the formula must contain at least one of these options. In other words, if a user needs conditional branching in calculation logic, a single action is all that is necessary.

IF(logical test, [value if true], [value if false])

The value returned by the function depends on the logical test results. For example, in **Figure 23**, a formula based on the IF function calculates a budget variance percentage, but only if the budget amount is not equal to zero, thereby avoiding the **#DIV/0** error. If the budget amount equals zero, the formula returns a null string and displays a blank cell. To return a null string, use two quotation marks entered side-by-side, as shown in the following formula.

=IF(Budget<>0,(Actual-Budget)/Budget,"")

AND, OR, NOT

To produce compound conditional formulas, you can use Boolean functions such as AND, OR, and NOT in combination with IF. For example, when using the AND function with multiple conditions, all conditions must test TRUE for the IF logical test to return TRUE. The following formula uses AND to satisfy various tests. When using OR with multiple conditions, at least one must test TRUE. The NOT function reverses the test result so that TRUE becomes FALSE and FALSE becomes TRUE.

=IF(AND(F10>500,A10= "Check"), "Yes", "No")

IFERROR

The **IFERROR** function combines the conditional logic of IF with error trapping. As a result, it simplifies creating complex formulas by trapping all errors with a single function. Previously, users employed unintuitive procedures to trap simple errors or used multiple error-trapping functions, which trapped only specific errors. The syntax for IFERROR is as follows.

IFERROR(original value or formula, value if error)

In this example, the variance column formula uses the IFERROR function to trap #DIV/0 errors. Before IFERROR, users had to use an IF function to test whether the divisor was zero. Based on the test, the formula completes the division calculation or displays some desired number, character, or text. The worksheet shown in **Figure 23** illustrates the process of using IF and IFERROR to trap errors.

	A	B	C	D	E	F	G	H
1	Larry's Landscaping Services							
2	Actual to Budget Performance							
3	For the Month Ended November 30, 2020							
4						Conventional Formula	Using IF	Using IFERROR
5			Budget	Actual	Variance	% Variance	% Variance	% Variance
6	6230	Auto Fuel	$ 1,275	$ 1,267	$ (8)	-0.63%	-0.63%	-0.63%
7	6240	Auto Maintenance	425	442	17	4.00%	4.00%	4.00%
8	6400	Bank Service Charges	213	213	-	0.00%	0.00%	0.00%
9	6550	Payroll Expenses	79,900	82,450	2,550	3.19%	3.19%	3.19%
10	6600	Delivery Fee	8,500	8,330	(170)	-2.00%	-2.00%	-2.00%
11	6900	Insurance	2,125	2,125	-	0.00%	0.00%	0.00%
12	7100	Equipment rental	-	4,887	4,887	#DIV/0!		
13	7200	Miscellaneous	850	2,235	1,385	162.94%	162.94%	162.94%
14	7300	Office Supplies	425	349	(77)	-18.00%	-18.00%	-18.00%
15	7500	Rent	17,000	17,000	-	0.00%	0.00%	0.00%
16	7553	Equipment Repairs	-	1,456	1,456	#DIV/0!		
17	7700	Tools and Misc. Equipment	1,275	1,131	(145)	-11.33%	-11.33%	-11.33%
18	7751	Gas and Electric	2,125	1,938	(187)	-8.80%	-8.80%	-8.80%
19	7753	Telephone	425	433	8	1.88%	1.88%	1.88%
20	7752	Water	638	629	(9)	-1.41%	-1.41%	-1.41%
21	9000	Interest Expense	2,975	2,933	(42)	-1.41%	-1.41%	-1.41%
22								
23	**Total Forecasted Operating Expenses**		$ 118,151	$ 127,857	$ 9,706	8.21%	8.21%	8.21%

Figure 23 - Using IFERROR to Simplify Error Trapping

Similarly, you could use IFERROR with VLOOKUP to create the exact match option. However, if no match exists, VLOOKUP returns **#N/A**. A better solution would be to use IFERROR to return "*Item Not Found*" when an exact match fails, as shown in the following formula.

=IFERROR(VLOOKUP(C5,LookupTable,3,FALSE), "Item Not Found")

IFS or Nested IFs

Microsoft 365 subscribers can use the relatively new **IFS** function instead of nested IFs. The feature is computationally more efficient and uses a simple and intuitive syntax, making it easier and less prone to error. In addition, IFS can test up to 127 conditions. The syntax for IFS is as follows.

IFS(logical test 1, value if test 1 is true, logical test 2, value if test 2 is true,… logical test n, value if test n is true)

Note that there is no specified result when any logical test is FALSE. That requires a user to identify what to do when the final logical test does not evaluate TRUE. For example, the following formula calculates the fiscal quarter and fiscal year for a specified date. In this case, the company uses a fiscal year that starts on November 1 and ends on October 31.

="Q"&@IFS(MONTH(DATE)<2,1,MONTH(DATE)<5,2,MONTH(DATE)<8,3, MONTH(DATE)<11,4,MONTH(DATE)>=11,1)

The IFS function evaluates from left to right, just like the IF function. As Excel progresses through the logical tests, the testing process stops once it finds a test that evaluates to TRUE. That explains why a January date evaluates to "Q1" even though it also meets the next test for "Q2".

Immediate IFs

An *Immediate IF* is an implied conditional test, often used in array formulas. You may use any of the comparison operators (**=**, **>**, **<**, **>=**, **<=**, **<>**) in the test. This example demonstrates how an array formula can summarize detailed expense amounts by Job and Account.

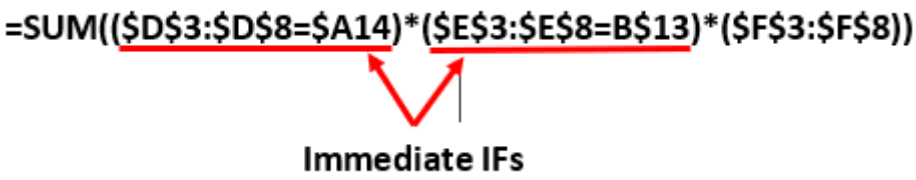


Figure 24 – Simple Array Formula to Calculate Expenses by Account and by Job Using Immediate IFs

The formula in cell B14 first tests to see if each Account in the range D3:D8 is equal to the Account number in cell A14 (8000) and then tests to see if each Job in the range E3:E8 matches the Job number in cell B13 (701). If the result of a test is **False**, the formula returns **0**. However, if the result is **True**, the formula returns **1**. Next, Excel multiplies the results of each test together. They are then multiplied by the appropriate amount from the range F3:F8. Then, Excel sums the data, as shown in **Figure 25**. Enter this formula using **CTRL** + **SHIFT** + **ENTER** in non-dynamic array aware Excel versions.

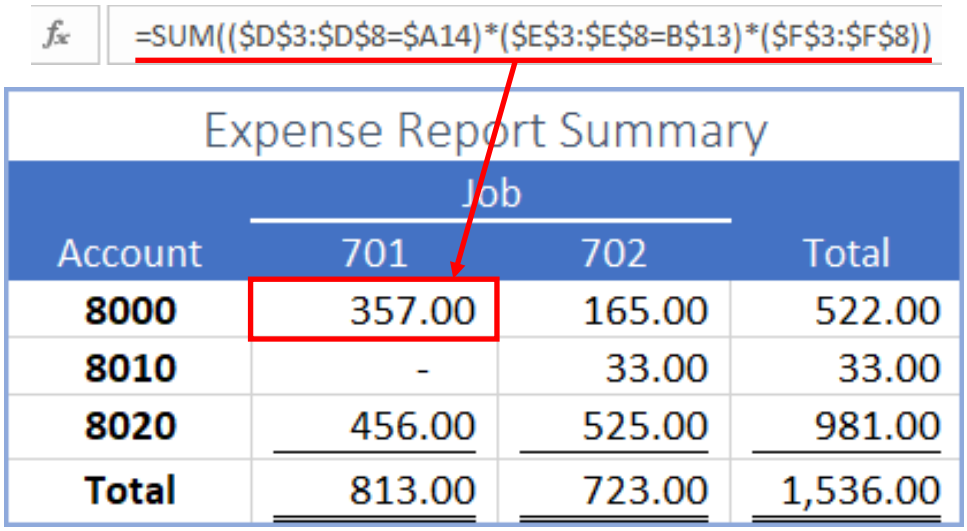


Figure 25 – Expense Report Summary Using an Array Formula with Immediate IFs

Multiple detailed expense entries exist for Account 8000 and Job 701, so the formula in cell B14 evaluates to $357.00. When testing numerous criteria, the multiplication operator (*) represents an **AND** condition, and the addition operator (+) translates to an **OR** condition.

SWITCH

You can use Excel's **SWITCH** function to evaluate a single expression against a list of up to 126 values and return the result corresponding to the first matching value. If SWITCH does not find a match, you can choose to return an optional default value. To illustrate, consider the worksheet and formula based on the SWITCH function pictured in **Figure 26**. In this example, the SWITCH function checks to see whether the value in cell A2 matches any of the following account numbers: 1000, 1100, 1200, 1500, 1599, 1800, or 1899. If SWITCH finds an exact match, it returns the corresponding account name – *Cash*, *Accounts Receivable*, etc. If SWITCH does not find an exact match, it returns *Account Not Found*. Like IFS, SWITCH can reduce your need for nested IF functions in a formula.

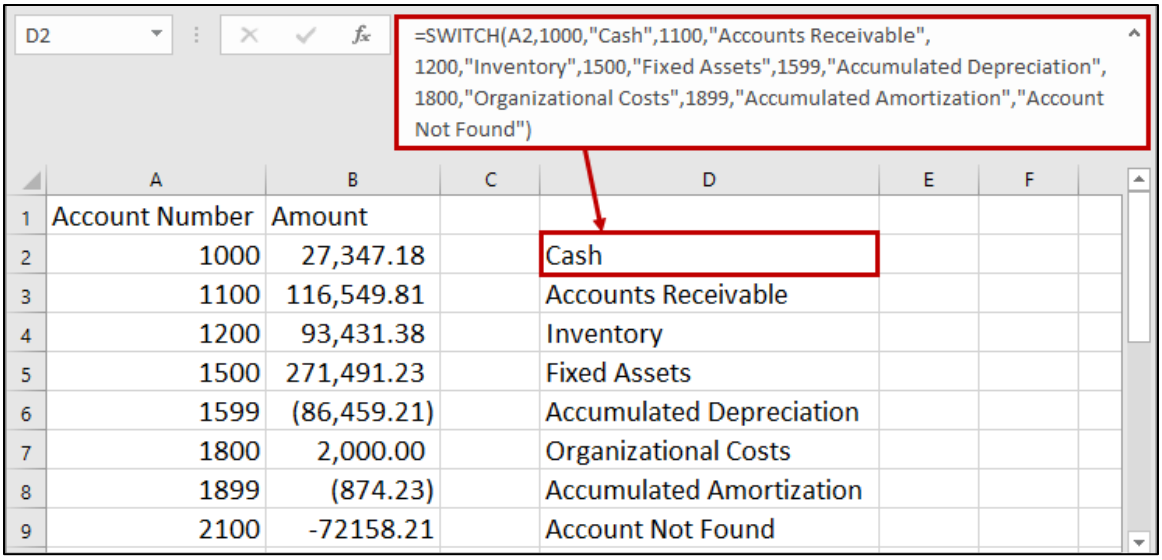


Figure 26 - Using Excel's SWITCH Function Instead of Nested IF Functions

Understanding GETPIVOTDATA and Cube Formulas

GETPIVOTDATA

The **GETPIVOTDATA** function extracts summarized information from a PivotTable. This function is handy when users want to prepare formal reports from PivotTable summaries or when users want to look up data summarized in a PivotTable.

The GETPIVOTDATA function has the following syntax:

GETPIVOTDATA(data field, pivottable, field 1, item 1, field 2, item 2,… field n, item n)

where:

> **Data field** represents the name of the data field, enclosed in quotation marks, that contains the data to extract.
>
> **PivotTable** is any cell, range of cells, or a defined name in the PivotTable from which to extract the data; it is generally specified as an absolute reference to the cell in the upper left corner of the PivotTable report.
>
> **Field *x*, item *x*** are pairs of field and item names that describe the data to retrieve. Up to 126 pairs of field and item names are permitted. If the field or item is not a date or a number, it must be enclosed in quotation marks.

The function extracts data from a PivotTable report based on the specified field and item pairs. **Figure 27** illustrates a formal report created from information extracted from a PivotTable. Note

how the formula identifies fields and items by referring to the report's row and column headings. This practice allows users to redefine the summary by changing the row labels or column headers.

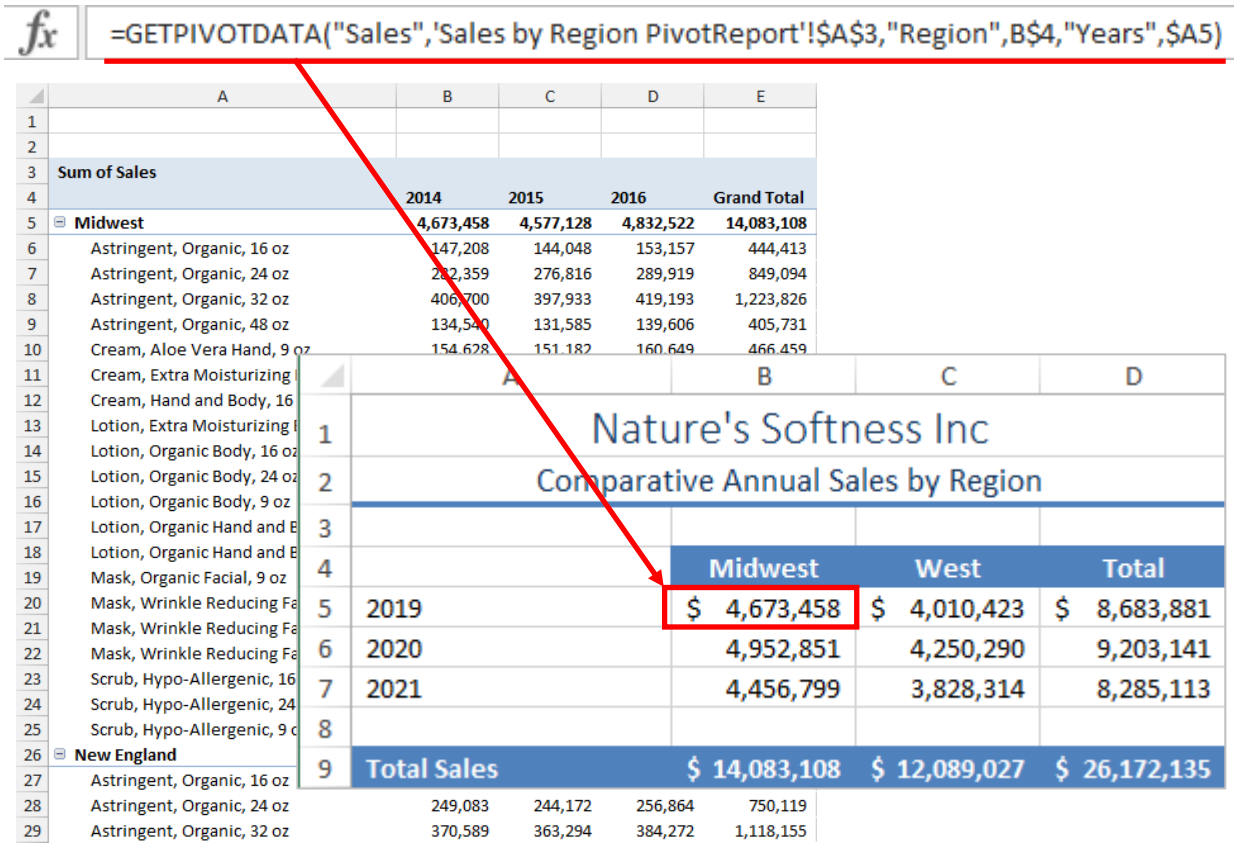


Figure 27 – Formal Report Created from PivotTable Using GETPIVOTDATA

GETPIVOTDATA extracts the specified data, even if users subsequently add additional data points or rearrange the PivotTable, as long as the desired data is ***visible*** on the face of the PivotTable. If the arguments do not describe a visible field and item pair, or if they include data on a filtered report so that the filtered data does not describe a visible field and item pair, GETPIVOTDATA returns **#REF!**

By default, Excel enables GETPIVOTDATA functionality. However, suppose you are creating a formula outside a PivotTable. As part of that process, you click inside the PivotTable. In that case, Excel automatically inserts the GETPIVOTDATA function into the formula. The arguments entered by the process point to a specific field and item intersection. This behavior frustrates many users because copying the formula down or across – a common method for building a report – does not index the formulas like other relative address formulas. Instead, the copied formulas display the same result as the original formula because they point to the same field and item intersection.

If you want to refer to a cell in a PivotTable using a relative reference that you can copy down or across, type the cell reference as part of the formula-building or editing process. In other words,

if you want to refer to cell **C16** in a PivotTable, enter the formula **=C16**. Alternatively, a user could turn off the automatic GETPIVOTDATA functionality. To disable GETPIVOTDATA, perform one of the following steps:

- Click **File, Options.** Click on **Formulas** in the **Navigation Pane** on the left and then uncheck **Use GetPivotData functions for PivotTable references**.

- Alternatively, position the cursor inside a PivotTable. On the **PivotTable Analyze** contextual tab, click Options. Uncheck **Generate GetPivotData**.

Note that this does not prevent a user from manually entering the GETPIVOTDATA function. Also, note that existing GETPIVOTDATA functions remain intact and continue to function correctly.

Cube Formulas

The Excel Data Model contains a self-dimensioning OLAP cube called an *intelligent cube*. Think of a cube as a multidimensional PivotTable, summarizing the intersections of rows, columns, and other dimensions already created. These intersections can be extracted to build reports using cube functions, like building reports with GETPIVOTDATA. Hence, a PivotTable is not required to summarize the data stored in the Excel Data Model.

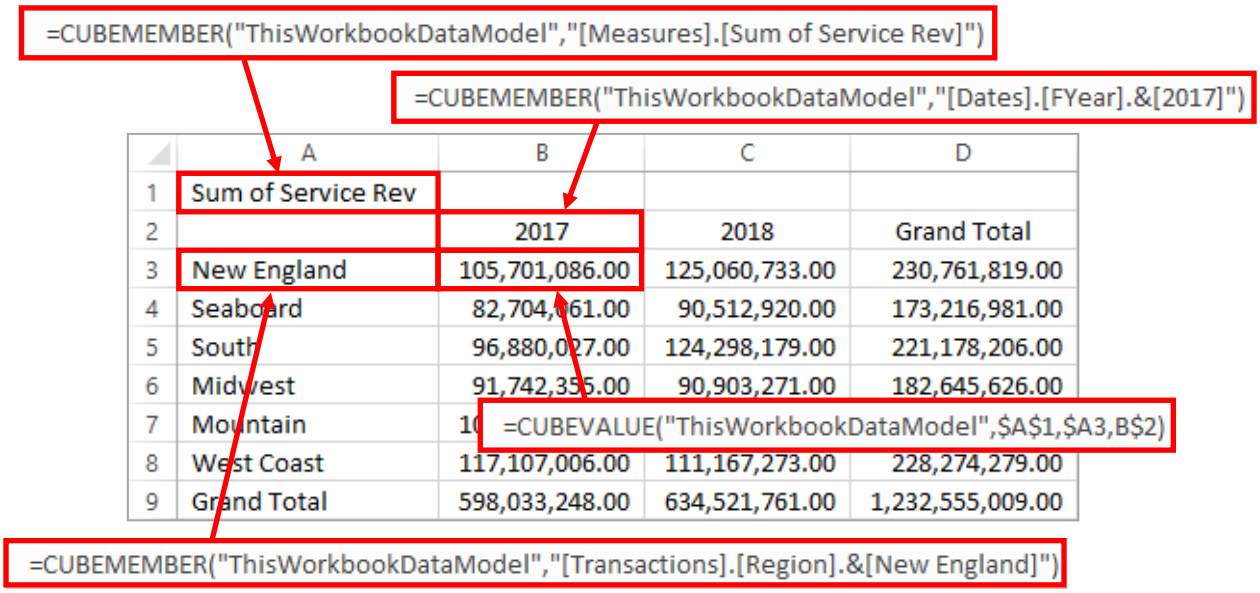
Excel has seven built-in cube functions for creating reports without using a PivotTable. These functions preserve the ability to refresh your reports when the underlying data changes.

Function	Description
CUBEKPIMEMBER	Returns a key performance indicator (KPI) property
CUBEMEMBER	Returns a member from the cube
CUBEMEMBERPROPERTY	Returns the value of a member property from the cube
CUBERANKEDMEMBER	Returns the nth, or ranked, member in a set
CUBESET	Defines a calculated set of members in a cube
CUBESETCOUNT	Returns the number of items in a set
CUBEVALUE	Returns an aggregated value from the cube

To explore cube functions, create a simple PivotTable based on a Data Model. In this example, the Data Model consists of four related tables – *Transactions*, *Industry*, *Service*, and *Dates*. A simple PivotTable is used to reduce the complexity of the example. Convert the PivotTable to formulas by selecting **OLAP Tools, Convert to Formulas** on the **PivotTable Tools, Analyze** tab.

This process converts a PivotTable built on the Excel Data Model into a report that uses cube functions to display specified intersections in the Data Model. Notably, these cube functions do not impair users' ability to refresh the report. Refreshing a report updates the values for any changes made in the underlying data. For example, you could update a monthly sales revenue report to display updated sales revenue amounts given the transactions occurring in the

intervening period. **Figure 28** shows the converted PivotTable, along with selected cube formulas.



	A	B	C	D
1	Sum of Service Rev			
2		2017	2018	Grand Total
3	New England	105,701,086.00	125,060,733.00	230,761,819.00
4	Seaboard	82,704,061.00	90,512,920.00	173,216,981.00
5	South	96,880,027.00	124,298,179.00	221,178,206.00
6	Midwest	91,742,355.00	90,903,271.00	182,645,626.00
7	Mountain	10		
8	West Coast	117,107,006.00	111,167,273.00	228,274,279.00
9	Grand Total	598,033,248.00	634,521,761.00	1,232,555,009.00

Figure 28 – PivotTable Converted Into Cube Formulas

By default, Excel uses cell references in the formulas to return the desired values from the Data Model. Examine more closely the cube formulas created by the conversion of the PivotTable. The column headers and row labels use the **CUBEMEMBER** function to identify their respective *dimensions*. The formulas in the report's body use the **CUBEVALUE** function to reference the dimensions defined in the column headers and row labels. These formulas identify specific intersections in the Excel Data Model containing a multidimensional cube. The intersection identified in each report cell determines the *measure's* value returned from the cube. The column headings and row labels act as filters applied to the measure – in this case, an *implicit* measure produced by the PivotTable creation process named **Sum of Service Rev**.

Make sure to arrange the PivotTable to display the level of detail required before converting it to formulas. After conversion, the limitations imposed on PivotTables disappear. For example, you can insert rows or columns across the report for summary calculations. This action is not possible in a PivotTable. **Figure 29** displays the report, including a summarizing calculation for **Total Eastern US** sales. As the underlying data changes over time, the report based on cube formulas can be refreshed, just like a PivotTable.

	2017	2018	Grand Total
New England	105,701,086.00	125,060,733.00	230,761,819.00
Seaboard	82,704,061.00	90,	
South	96,880,027.00	124,	
Midwest	91,742,355.00	90,	
Total Eastern US	**377,027,529.00**	**430,775,103.00**	**807,802,632.00**
Mountain	103,898,713.00	92,579,385.00	196,478,098.00
West Coast	117,107,006.00	111,167,273.00	228,274,279.00
Grand Total	**598,033,248.00**	**634,521,761.00**	**1,232,555,009.00**

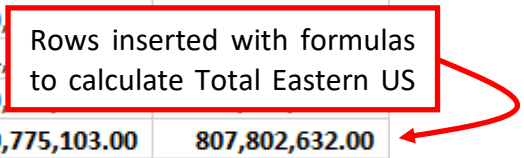


Figure 29 – Reports Based on Cube Formulas Remove PivotTable Limitations

Report authors with access to Power Pivot can create reports directly from the Data Model without creating a PivotTable because they can create *explicit* measures in PowerPivot. If you don't have access to Power Pivot, you must use a PivotTable to create *implicit* measures before creating cube formula reports. Alternatively, depending on your Excel version, you can create explicit measures in selected tables in the PivotTable Field List. To do so, right-click on a table in the Field List and click **Add Measure**.

Calculating Explicit Measures with DAX Functions

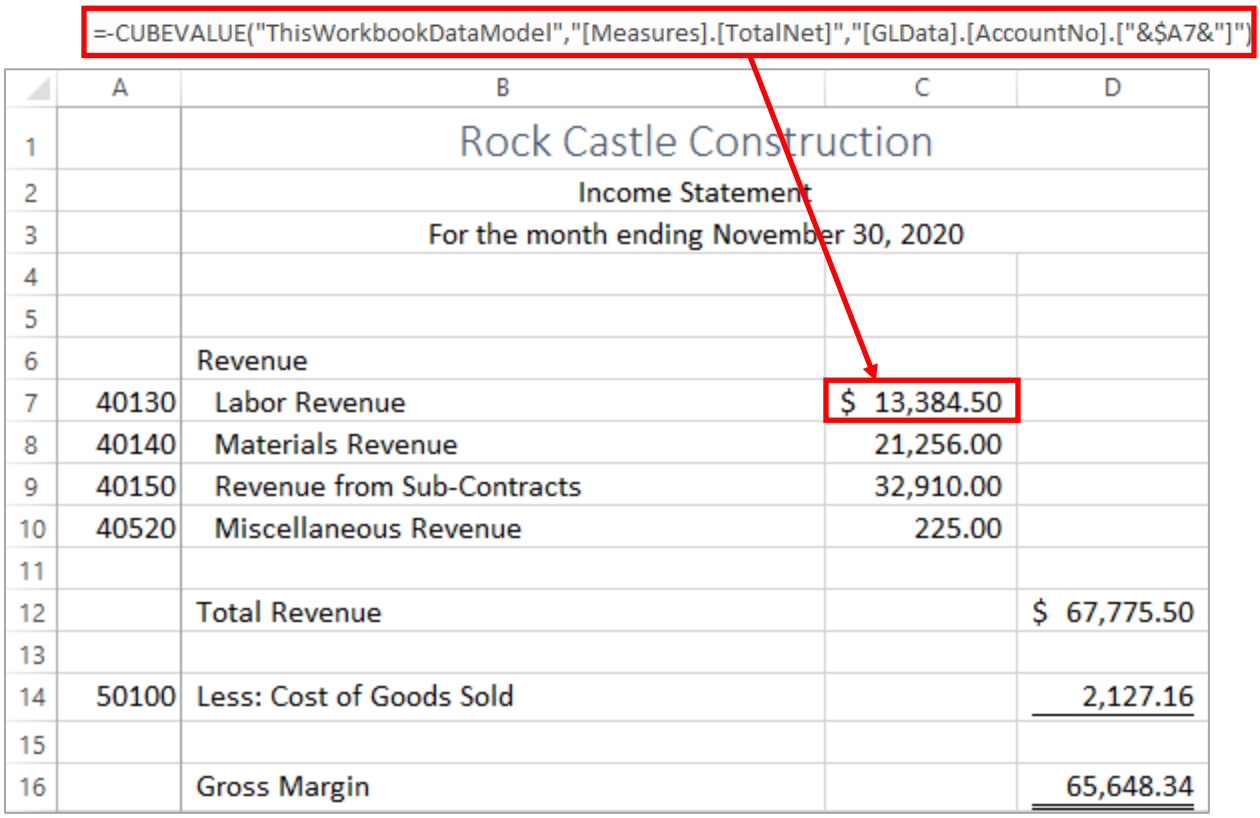
Explicit measures are created in the Data Model using **Data Access Expression** (DAX) functions. PowerPivot and the Excel Data Model provide over 300 DAX functions for creating measures and calculated columns to facilitate sophisticated data analysis and reporting. From simple functions, such as **FORMAT** (for reformatting data), to advanced functions, such as **RELATED** (for de-normalizing data), DAX offers functions for nearly every data task. Super functions, like **SUMX** (to aggregate calculations across all records) and **CALCULATE** (to modify the filter context), provide incredible power and flexibility for data analysis. You can combine those with thirty-five functions supplying date and time intelligence for added computational power, such as **TOTALYTD**, **PREVIOUSMONTH**, and **SAMEPERIODLASTYEAR**.

The DAX function library is based partly on the Excel function library. Many functions, such as **SUM**, **COUNT**, and **AVERAGE,** are in both libraries, but the syntax of these two sets of functions is very different. The following list summarizes the differences and similarities between Excel and DAX functions.

- DAX functions use tables and columns as references, while Excel functions use cell, range, and worksheet references.
- You *cannot* use DAX functions in Excel formulas; likewise, you cannot use Excel functions in DAX formulas.
- DAX date and time functions return a date/time data type, while Excel date and time functions return a numeric value representing a specific date and time.

- Some DAX functions return a table of values or calculations based on a table of values. No Excel functions return a table, but some work with arrays.
- DAX has lookup functions similar to the lookup functions in Excel, but the DAX functions *require* that relationships exist between the tables.
- All data in a column must be the same data type. If data is not the same type, DAX changes the data type of the entire column to one that best accommodates all values.

To illustrate, we will use an explicit measure to prepare a custom-formatted income statement from QuickBooks data using cube formulas, as shown in **Figure 30**. The General Ledger report is first exported to a CSV file and then transformed and imported into the Data Model using Power Query.



	A	B	C	D
1		Rock Castle Construction		
2		Income Statement		
3		For the month ending November 30, 2020		
4				
5				
6		Revenue		
7	40130	Labor Revenue	$ 13,384.50	
8	40140	Materials Revenue	21,256.00	
9	40150	Revenue from Sub-Contracts	32,910.00	
10	40520	Miscellaneous Revenue	225.00	
11				
12		Total Revenue		$ 67,775.50
13				
14	50100	Less: Cost of Goods Sold		2,127.16
15				
16		Gross Margin		65,648.34

Figure 30 – Building an Income Statement with Cube Formulas

Note that this report uses an *explicit measure* named **[TotalNet]**. Also, note that column **A**, which we will exclude from the Print Area, contains account numbers to identify a specific member in a dimension named **[GLData].[AccountNo]** representing a table and column name in the Data Model. The formula uses mixed referencing (partially relative and partially absolute cell references) to facilitate copying it down and across columns. The minus sign (**-**) is used to reverse the sign of the revenue values. You can check the report's accuracy by comparing it to a printed financial statement for the same period.

INDIRECT and OFFSET

INDIRECT

Excel's **INDIRECT** function returns the contents of a cell or cells specified by a text string. INDIRECT can be very useful when converting a text reference to a valid cell reference in a formula. The reference created by INDIRECT will not change even when cells, rows, or columns are inserted or deleted. For example, **=INDIRECT("A1:A100")** always refers to the first 100 rows of column A, even if rows are deleted or inserted.

The syntax for INDIRECT is:

INDIRECT(reference text, [A1])

where:

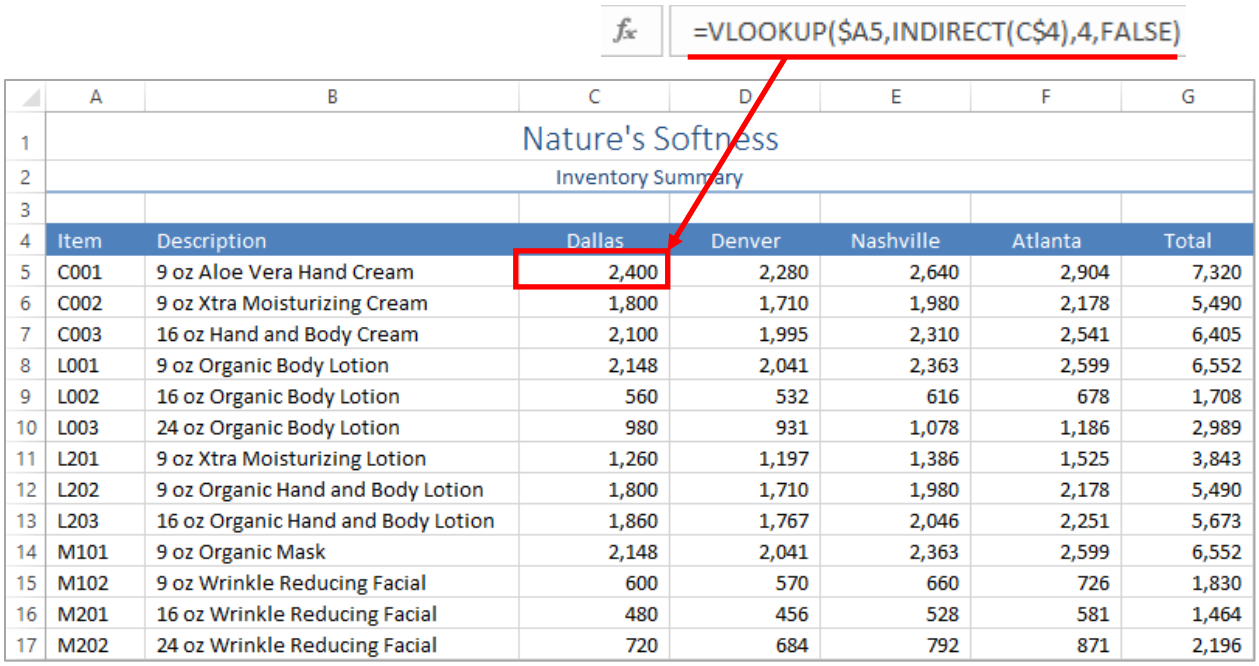
Reference text refers to a cell in the current worksheet, another worksheet in the same workbook, or a worksheet in another workbook. If the referenced cell is in another workbook, then the other workbook must be open.

A1 (optional) specifies the reference type contained in the text.

Option	Option Behavior
TRUE or Omitted	Reference text is interpreted as A1-style reference
FALSE	Reference text is interpreted as R1C1-style reference

Figure 31 presents a simple example of using INDIRECT. In this illustration, INDIRECT identifies the table array from the column name (referring to a table of the same name). INDIRECT allows users to build a single formula and *copy it down and across* to produce the report. Conventional practice would have required six formulas, one for each column.

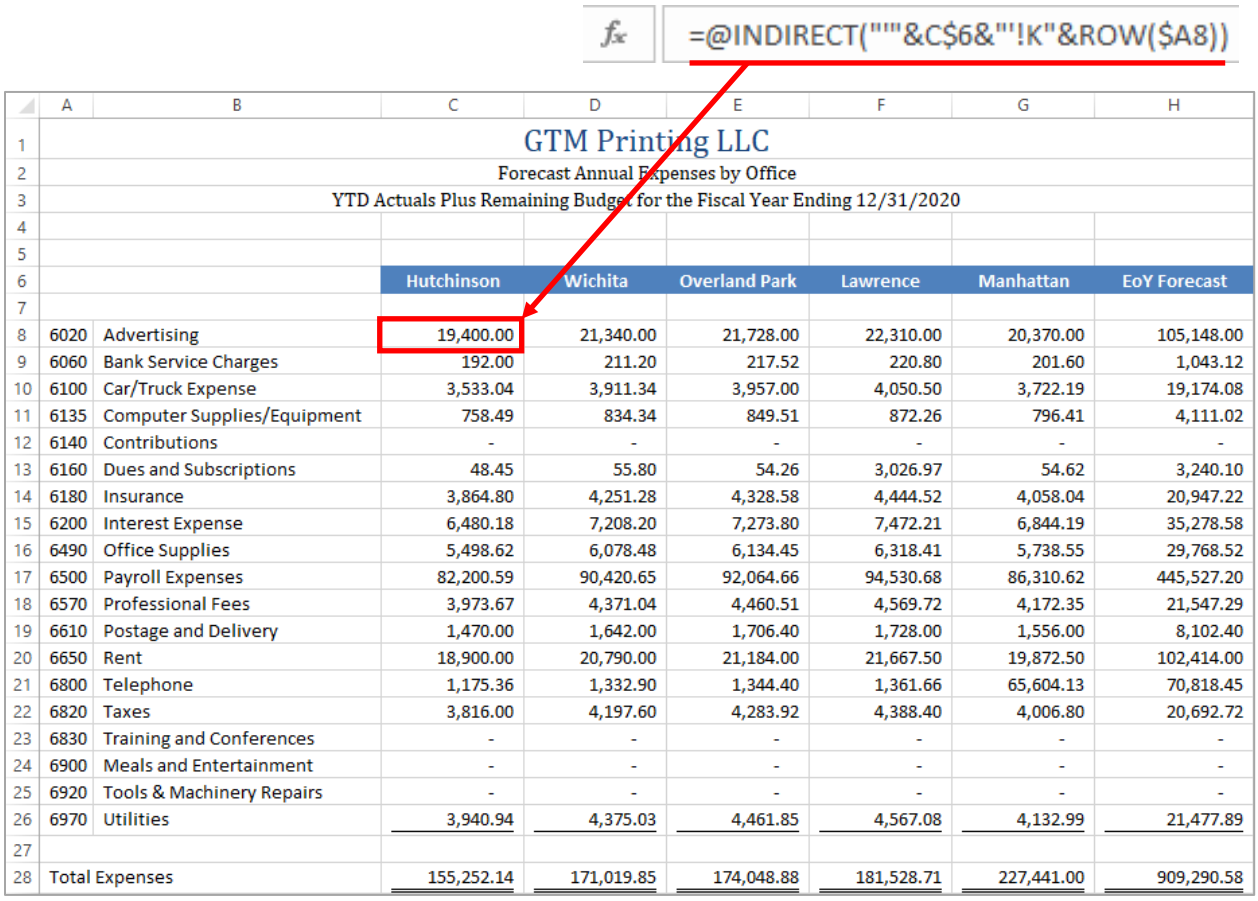
As shown in this example, using INDIRECT is an Excel best practice because it 1) saves time, 2) reduces the potential for error, and 3) is easier to audit or verify because it is only one formula.



	A	B	C	D	E	F	G
1	Nature's Softness						
2	Inventory Summary						
3							
4	Item	Description	Dallas	Denver	Nashville	Atlanta	Total
5	C001	9 oz Aloe Vera Hand Cream	2,400	2,280	2,640	2,904	7,320
6	C002	9 oz Xtra Moisturizing Cream	1,800	1,710	1,980	2,178	5,490
7	C003	16 oz Hand and Body Cream	2,100	1,995	2,310	2,541	6,405
8	L001	9 oz Organic Body Lotion	2,148	2,041	2,363	2,599	6,552
9	L002	16 oz Organic Body Lotion	560	532	616	678	1,708
10	L003	24 oz Organic Body Lotion	980	931	1,078	1,186	2,989
11	L201	9 oz Xtra Moisturizing Lotion	1,260	1,197	1,386	1,525	3,843
12	L202	9 oz Organic Hand and Body Lotion	1,800	1,710	1,980	2,178	5,490
13	L203	16 oz Organic Hand and Body Lotion	1,860	1,767	2,046	2,251	5,673
14	M101	9 oz Organic Mask	2,148	2,041	2,363	2,599	6,552
15	M102	9 oz Wrinkle Reducing Facial	600	570	660	726	1,830
16	M201	16 oz Wrinkle Reducing Facial	480	456	528	581	1,464
17	M202	24 oz Wrinkle Reducing Facial	720	684	792	871	2,196

Figure 31 – Using INDIRECT to Identify the Table Array from the Column Name in VLOOKUP

Figure 32 shows INDIRECT in a more sophisticated composed formula, one produced from individual text strings (enclosed in quotation marks) concatenated using the ampersand symbol (**&**). Remember that the INDIRECT function returns the contents of a cell or cells specified by a text string. Hence, we can compose text strings within INDIRECT to produce the desired calculation or data.



		Hutchinson	Wichita	Overland Park	Lawrence	Manhattan	EoY Forecast
6020	Advertising	19,400.00	21,340.00	21,728.00	22,310.00	20,370.00	105,148.00
6060	Bank Service Charges	192.00	211.20	217.52	220.80	201.60	1,043.12
6100	Car/Truck Expense	3,533.04	3,911.34	3,957.00	4,050.50	3,722.19	19,174.08
6135	Computer Supplies/Equipment	758.49	834.34	849.51	872.26	796.41	4,111.02
6140	Contributions	-	-	-	-	-	-
6160	Dues and Subscriptions	48.45	55.80	54.26	3,026.97	54.62	3,240.10
6180	Insurance	3,864.80	4,251.28	4,328.58	4,444.52	4,058.04	20,947.22
6200	Interest Expense	6,480.18	7,208.20	7,273.80	7,472.21	6,844.19	35,278.58
6490	Office Supplies	5,498.62	6,078.48	6,134.45	6,318.41	5,738.55	29,768.52
6500	Payroll Expenses	82,200.59	90,420.65	92,064.66	94,530.68	86,310.62	445,527.20
6570	Professional Fees	3,973.67	4,371.04	4,460.51	4,569.72	4,172.35	21,547.29
6610	Postage and Delivery	1,470.00	1,642.00	1,706.40	1,728.00	1,556.00	8,102.40
6650	Rent	18,900.00	20,790.00	21,184.00	21,667.50	19,872.50	102,414.00
6800	Telephone	1,175.36	1,332.90	1,344.40	1,361.66	65,604.13	70,818.45
6820	Taxes	3,816.00	4,197.60	4,283.92	4,388.40	4,006.80	20,692.72
6830	Training and Conferences	-	-	-	-	-	-
6900	Meals and Entertainment	-	-	-	-	-	-
6920	Tools & Machinery Repairs	-	-	-	-	-	-
6970	Utilities	3,940.94	4,375.03	4,461.85	4,567.08	4,132.99	21,477.89
Total Expenses		155,252.14	171,019.85	174,048.88	181,528.71	227,441.00	909,290.58

Figure 32 – INDIRECT in a Composed Formula

OFFSET

OFFSET returns a reference to a range that is a specified number of rows and columns from an identified cell or range of cells. The reference OFFSET returns can be a single cell or a range of cells, and users can specify the number of rows and columns to return.

The syntax for OFFSET is as follows:

OFFSET(reference, rows, cols, [height], [width])

where:

Reference is the cell or range of cells that will serve as the base for the offset.

Rows is the number of rows above (specified as a negative number) or below (specified as a positive number) from the base reference.

Columns is the number of columns to the left (specified as a negative number) or to the right (specified as a positive number) from the base reference.

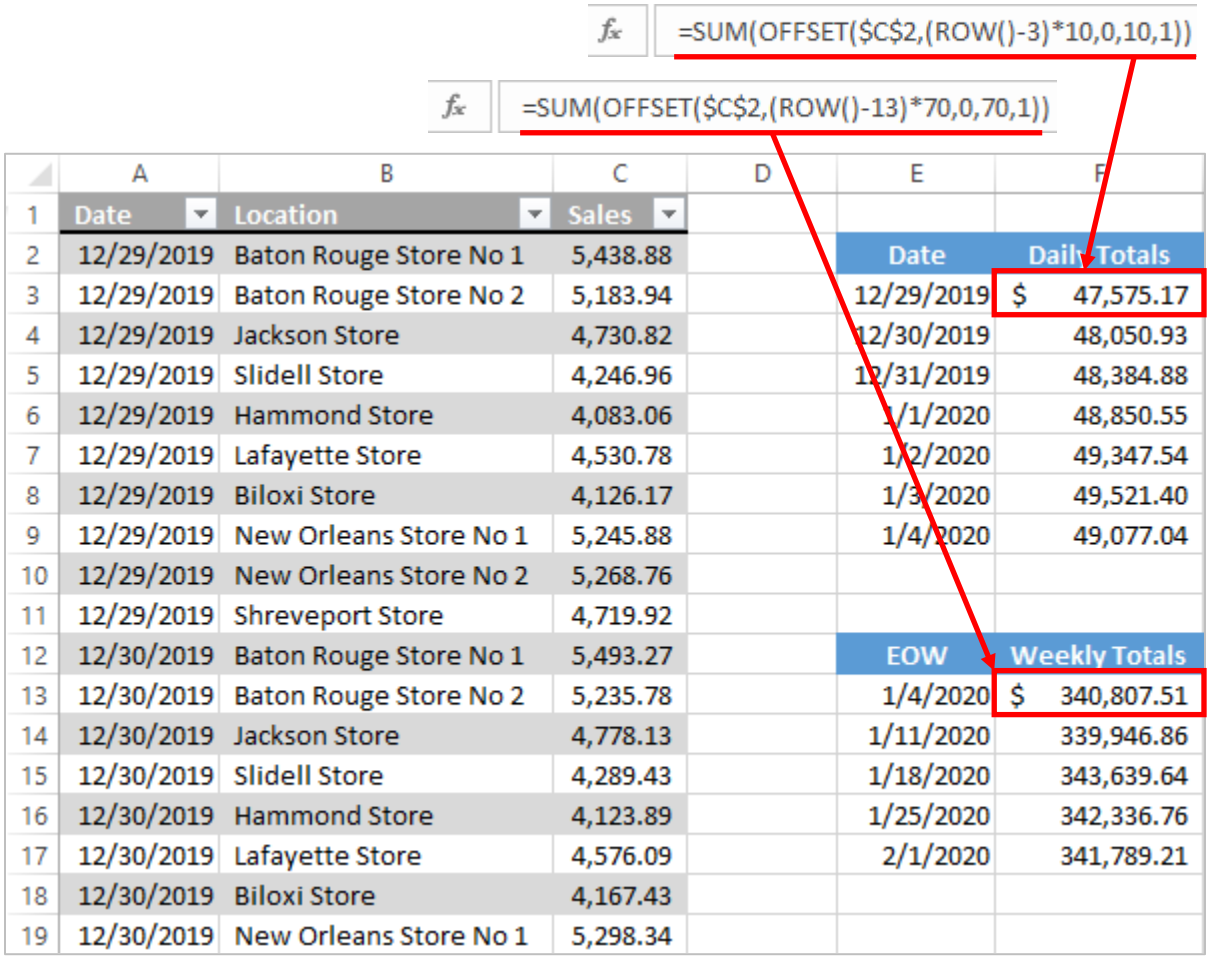
Height (optional) is the number of rows in the returned reference.

Width (optional) is the number of columns in the returned reference.

If **height** or **width** is omitted, the returned reference is assumed to be the same height or width as the **reference**. OFFSET returns a reference; you can use it with any function expecting a reference argument.

You can use OFFSET to build dynamic named ranges for charts or PivotTables so that the data range expands or contracts as the data set changes. However, since the advent of Excel tables, using OFFSET in these environments is less valuable. Still, OFFSET has many uses, including summarizing data for every *n* rows or *n* columns.

In this example, Joan owns several fast-food outlets. Her controller has prepared a report identifying daily sales for each of the ten stores sorted by date. In addition, she wants a report that calculates daily and weekly revenue for all stores to analyze trends in sales. OFFSET can be used with SUM to produce the summary, as shown in **Figure 33**.



	A	B	C	D	E	F
1	Date	Location	Sales			
2	12/29/2019	Baton Rouge Store No 1	5,438.88		Date	Daily Totals
3	12/29/2019	Baton Rouge Store No 2	5,183.94		12/29/2019	$ 47,575.17
4	12/29/2019	Jackson Store	4,730.82		12/30/2019	48,050.93
5	12/29/2019	Slidell Store	4,246.96		12/31/2019	48,384.88
6	12/29/2019	Hammond Store	4,083.06		1/1/2020	48,850.55
7	12/29/2019	Lafayette Store	4,530.78		1/2/2020	49,347.54
8	12/29/2019	Biloxi Store	4,126.17		1/3/2020	49,521.40
9	12/29/2019	New Orleans Store No 1	5,245.88		1/4/2020	49,077.04
10	12/29/2019	New Orleans Store No 2	5,268.76			
11	12/29/2019	Shreveport Store	4,719.92			
12	12/30/2019	Baton Rouge Store No 1	5,493.27		EOW	Weekly Totals
13	12/30/2019	Baton Rouge Store No 2	5,235.78		1/4/2020	$ 340,807.51
14	12/30/2019	Jackson Store	4,778.13		1/11/2020	339,946.86
15	12/30/2019	Slidell Store	4,289.43		1/18/2020	343,639.64
16	12/30/2019	Hammond Store	4,123.89		1/25/2020	342,336.76
17	12/30/2019	Lafayette Store	4,576.09		2/1/2020	341,789.21
18	12/30/2019	Biloxi Store	4,167.43			
19	12/30/2019	New Orleans Store No 1	5,298.34			

Figure 33 – Using OFFSET to SUM Daily and Weekly Sales

There are ten stores, so the formula to create weekly sales totals must sum each ten-row set beginning with cell C2. Here's the OFFSET-based formula for the calculation.

=SUM(OFFSET(C2,(ROW()-2)*10,0,10,1))

Cell **C2** serves as the anchor **reference** for OFFSET. The **ROW** function provides a dynamic calculation for the **row** offset (the first row) of each ten-store range. ROW() returns the row number of the cell (**F3**) in which the function is used, in this case, "3". In summing the sales, we initially want to start with the anchor reference, represented by "0". Hence, we subtract 3 from the ROW() so that the resulting value is zero (3 – 3 = 0). Then, the ROW()-3 calculation is multiplied by 10 (ten stores) to calculate the **row** offset for each ten-store sum. In cell **F3**, the offset is 0 ((3-3)*10); in cell **F4**, it is 10 ((4-3)*10); and in cell **F5**, it is 20 (5-3)*10); and so forth.

To complete the OFFSET formula, set the **column** offset to 0, the **height** of the returned reference to 10 rows (the number of stores to sum), and the **width** to 1 column.

Dates and Time

Dates and times in Excel are simply numbers formatted to look like dates and times. Dates are whole numbers ranging from 1 to 2958465, with "1" representing January 1, 1900, and "2958465" representing December 31, 9999. When you enter data into Excel that resembles a date or time, such as "3/29/2023", Excel converts the entry into the appropriate date value and formats it to look like a date.

While many Excel users understand that whole numbers represent dates, some struggle with times. Decimal values ranging from 0 to 0.99999 represent time in Excel, with "0" representing "12:00:00 AM" and "0.99999" representing "11:59:59 PM". The decimal identifies what portion of a 24-hour day has elapsed. For example, "0.50" represents "12:00 PM (noon)", and "0.75" represents "6:00 PM". You can combine both elements to express a specific date and time when working with dates and times. For example, value "44104.75" represents "9/30/2020 6:00 PM".

You can add and subtract these date and time values to calculate based on the passage of time, such as for accruals, deferrals, depreciation, and amortization. For example, to calculate the amount of interest accrued on a $200,000 loan that originates on February 19, 2020, carries an annual interest rate of 3%, and matures on December 31, 2020, use the formula displayed in **Figure 34**.

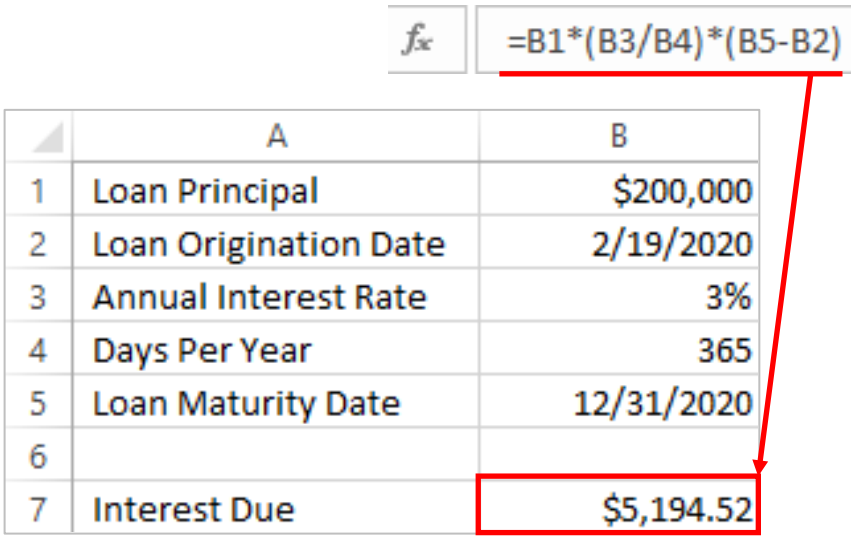


Figure 34 - Simple Interest Calculation Using Date Arithmetic

Date and Time Functions

Excel provides twenty-two date and time-specific functions to assist users in manipulating dates and times. Though a complete discussion of all twenty-two is beyond the scope of this session, the following table summarizes several of the more relevant ones.

Function and Syntax	Calculates or Returns
DATE(year, month, day)	Combines year, month, and day entries into a number that represents a date in Excel
DATEVALUE(date text)	Converts a text string into a number that represents a date in Excel
WEEKDAY(serial number, return type)	Returns a value of 1 to 7, representing the named days of the week
YEAR(serial number)	Returns the year of the date in the referenced cell or the number in the formula
MONTH(serial number)	Returns the month of the date in the referenced cell or the number in the formula
DAY(serial number)	Returns the day of the date in the referenced cell or the number in the formula
TODAY( )	Returns the current date based on the computer clock
NOW( )	Returns the current date and time based on the computer clock
EOMONTH(serial number, month)	Returns the serial number of the last day of the month before or after a specified number of months

Adding Date and Time Stamps

Instead of manually typing dates and times into cells, you can quickly add date or time stamps to any worksheet using the keyboard shortcuts shown in **Figure 35**.

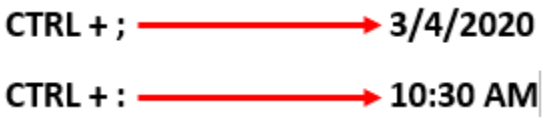


Figure 35 – Add Date and Time Stamps Using Keyboard Shortcuts

Note that date and time stamps added using these techniques are hard-coded into the worksheet and do not update with the system clock. For a dynamic datestamp, enter **=TODAY( )** in a cell. For a dynamically updating timestamp, enter **=NOW( )** and change the format so that the cell only displays the time.

Measuring Total Elapsed Hours

A simple formula that subtracts the departure time from the arrival time will calculate the elapsed time. However, if the *General* or *Number* format applies to the cell containing the calculation, Excel presents the result in terms of days and fractions of a day, as shown in **Figure 36**. To display the formula's calculation in hours and minutes, change the cell format to **[h]:mm.** Make sure to enclose the hour component in brackets. Otherwise, Excel ignores the fact that the trip took more than one day. In that case, the cell would display 20:45 instead of the correct elapsed time of 44:45.

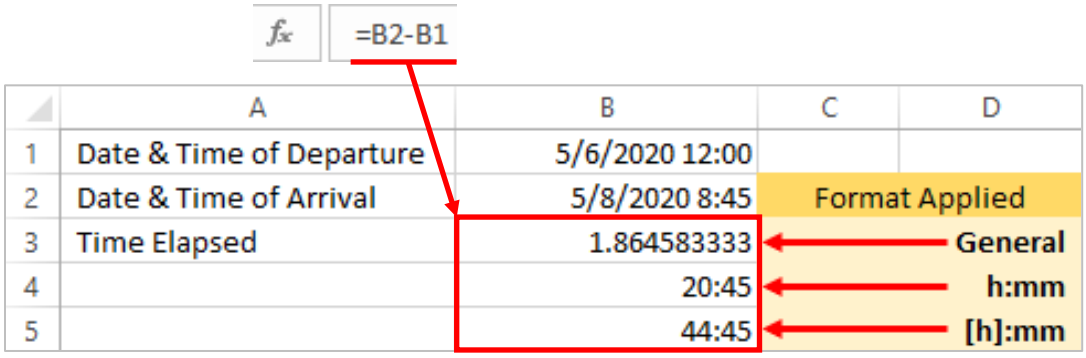
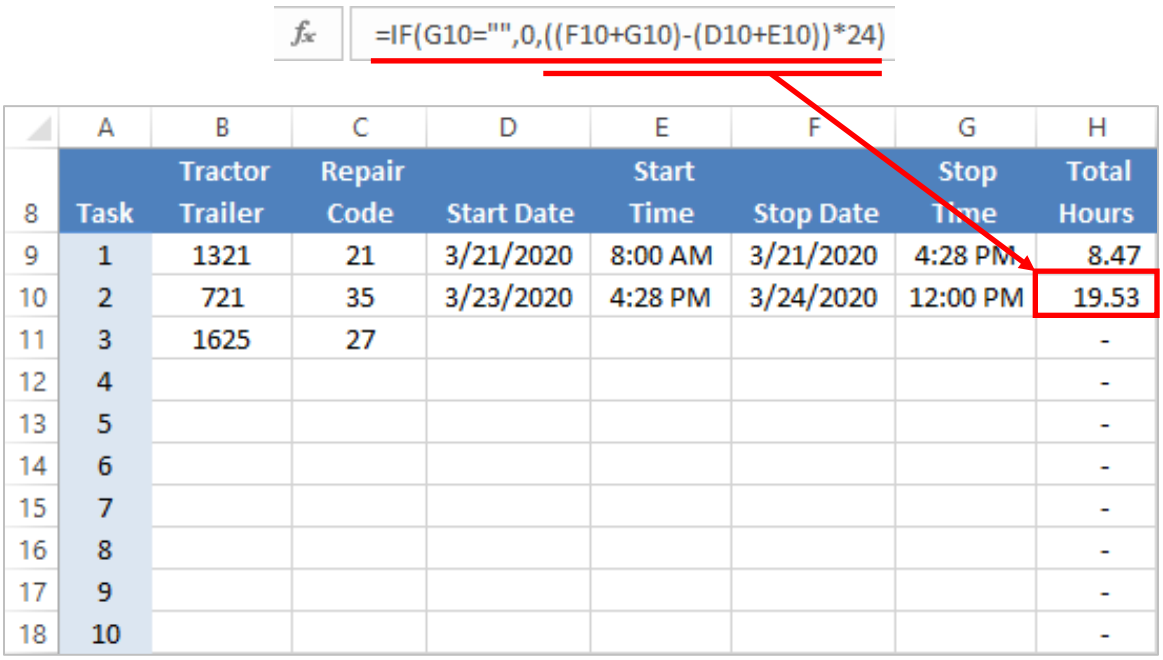


Figure 36 - Measuring Total Elapsed Time

The timesheet in **Figure 37** contains separate columns for starting and stopping dates and starting and stopping times. To compute elapsed time in this circumstance, use the following general formulation.

(Stop Date + Stop Time) – (Start Date + Start Time)

In the example, Excel presents the calculated elapsed time in days and fractions of a day. To convert the elapsed time to hours and fractions of an hour, multiply the elapsed time by 24, as shown in Figure 37.

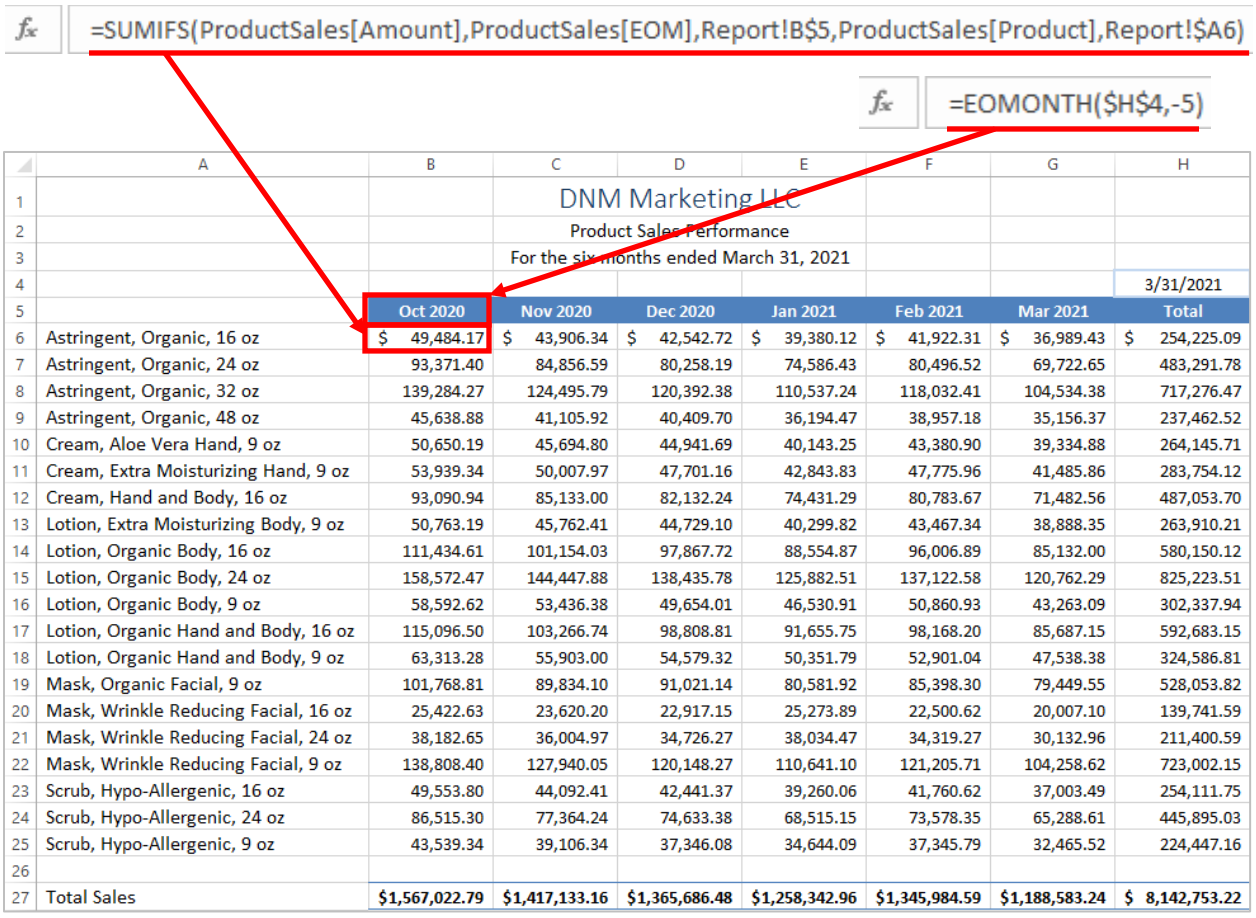


	A	B	C	D	E	F	G	H
8	Task	Tractor Trailer	Repair Code	Start Date	Start Time	Stop Date	Stop Time	Total Hours
9	1	1321	21	3/21/2020	8:00 AM	3/21/2020	4:28 PM	8.47
10	2	721	35	3/23/2020	4:28 PM	3/24/2020	12:00 PM	19.53
11	3	1625	27					-
12	4							-
13	5							-
14	6							-
15	7							-
16	8							-
17	9							-
18	10							-

Figure 37 – Calculating Elapsed Time in Hours

EOMONTH

In the example shown in **Figure 38**, a multicolumn report using **SUMIFS** on data stored in a table to retrieve and summarize transactions that meet specific criteria. Cell H4 uses **List validation** with **EOMDates** (a defined name) to add end-user interaction as the list values source. The formulas in cells **B6**:**G6** use the **EOMONTH** function to head the columns with the appropriate month-ending dates, formatted with a custom date format, "**mmm yyyy**". Adding conditional formatting would turn this into a visual exception report. Considering that the report is built with formulas connected to a table, the conditional formats will reflect changes in the data as the data is updated.



	A	B	C	D	E	F	G	H
1			DNM Marketing LLC					
2			Product Sales Performance					
3			For the six months ended March 31, 2021					
4								3/31/2021
5		Oct 2020	Nov 2020	Dec 2020	Jan 2021	Feb 2021	Mar 2021	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 38 – Summary Report Constructed Using SUMIFS and EOMONTH

Summary

The subset of advanced functions and formulas presented in this session should interest accounting and finance professionals. Mastering and using these functions can provide the power to yield substantial productivity gains and improve accuracy when working with sophisticated worksheet models. Additionally, although some of these functions are available only to those using a subscription-based version of Excel, you can access most of these tools in all Excel 2016 and newer versions. Accordingly, we encourage you to seek opportunities to take advantage of these features to improve your efficiency and effectiveness with Excel.