K2's Excel Update - 2025

Excel remains a staple technology for almost all business professionals, particularly those involved with accounting and financial functions. Furthermore, Microsoft continues to add significant functionality to its market-leading spreadsheet, but many remain unaware of Excel's newer features. In this session, you will learn about some of Excel's newer features and also about some "legacy" features that may have slipped by you.

Among the many topics covered in this session are Dynamic Arrays, XLOOKUP, PivotTable secrets, Excel's GROUPBY and PIVOTBY functions, date arithmetic, and Power Query. If you're going to take only one Excel course this year, this should be the one!

K2 Enterprises

Table of Contents

Introduction

Excel remains a staple technology for almost all business professionals, particularly those involved with accounting and financial functions. Further, Microsoft continues to add enormous amounts of functionality to its market-leading spreadsheet, but many remain unaware of Excel's newer features. In this session, you will learn about some of Excel's newer features and also about some "legacy" features that may have slipped by you.

Among the many topics covered in this session are Dynamic Arrays, XLOOKUP, PivotTable secrets, Excel's GROUPBY and PIVOTBY functions, date arithmetic, and Power Query. If you're going to take only one Excel course this year, this should be the one!

Learning Objectives

Upon completing this session, you should be able to:

- List examples of how Dynamic Arrays improve efficiency and reduce errors;
- Differentiate between XLOOKUP and VLOOKUP;
- Identify how Excel's PIVOTBY function differs from traditional PivotTables;
- Name an approach you can use to find new features in Excel; and
- List at least three benefits associated with using Power Query.

Your Version Of Excel And Why It Matters

Before detailing specific new Excel features and functions, let us quickly describe what "new" means in the context of this session. This conversation is necessary because "new" is a relative term, depending on what version of Excel you have and how often you receive updates.

Perpetual Licenses

If you use a perpetual license of Excel, such as Excel 2019, the concept of "new" is straightforward. You receive new features whenever you upgrade to the next version of Excel. Thus, if you recently upgraded to a newer version of Excel, then all the new features included in that release are indeed "new" to you. You will not receive additional new features until you upgrade to the next version of Excel.

Subscription Licenses

On the other hand, if you run a version of Excel provided through a Microsoft 365/Office 365 subscription plan, you receive periodic updates to your instance of Excel (and other Office applications). These updates occur at different intervals, depending on what ***channel*** you or your IT staff elect. In this environment, three primary channels are available: 1) **Current Channel**, 2) **Monthly Enterprise Channel**, and 3) **Semi-Annual Enterprise Channel**. As detailed in **Table 1**, feature updates occur on an unscheduled basis, monthly, or semi-annually.[1]

Comparing The Primary Update Channels For Microsoft 365 Apps			
	Current Channel	**Monthly Enterprise Channel**	**Semi-Annual Enterprise Channel**
Recommended use	Provide your users with new Office features as soon as they are ready, but on no set schedule.	Provide your users with new Office features only once a month and on a predictable schedule.	For select devices in your organization, where extensive testing is needed before rolling out new Office features. For example, to comply with regulatory, governmental, or other organizational requirements.
Release frequency	At least once a month (likely more often), but on no set schedule	Once a month, on the second Tuesday of the month	Once a month, on the second Tuesday of the month
Feature updates	As soon as they're ready (usually once a month), but on no set schedule	Once a month, on the second Tuesday of the month	Twice a year (in January and July), on the second Tuesday of the month
Security updates (if needed)	Once a month, on the second Tuesday of the month	Once a month, on the second Tuesday of the month	Once a month, on the second Tuesday of the month
Non-security updates (if needed)	Usually at least once a month (possibly more often), but no set schedule	Once a month, on the second Tuesday of the month	Once a month, on the second Tuesday of the month
Support duration for a given version	Until the next version is released with new features, which is usually about one month	Two months	Fourteen months

Table 1 - Comparing Microsoft 365 Apps Update Channels

Notably, there is an option to enroll in the Current Channel (Preview). This channel provides users with an "early look" at new and updated features that will soon debut in the Current Channel.

So, effectively, there are four primary update channels available for Microsoft 365 Apps. If team members within an organization are enrolled in different channels, you may have four distinct sets of features and functions in use simultaneously within the organization. Thus, what a user in the Semi-Annual Enterprise channel considers a “new” feature might be old-hat for someone in the Current Channel.

You can see which channel you are in by clicking **File**, **Account**, **Options** to display the window pictured in **Figure 1**.

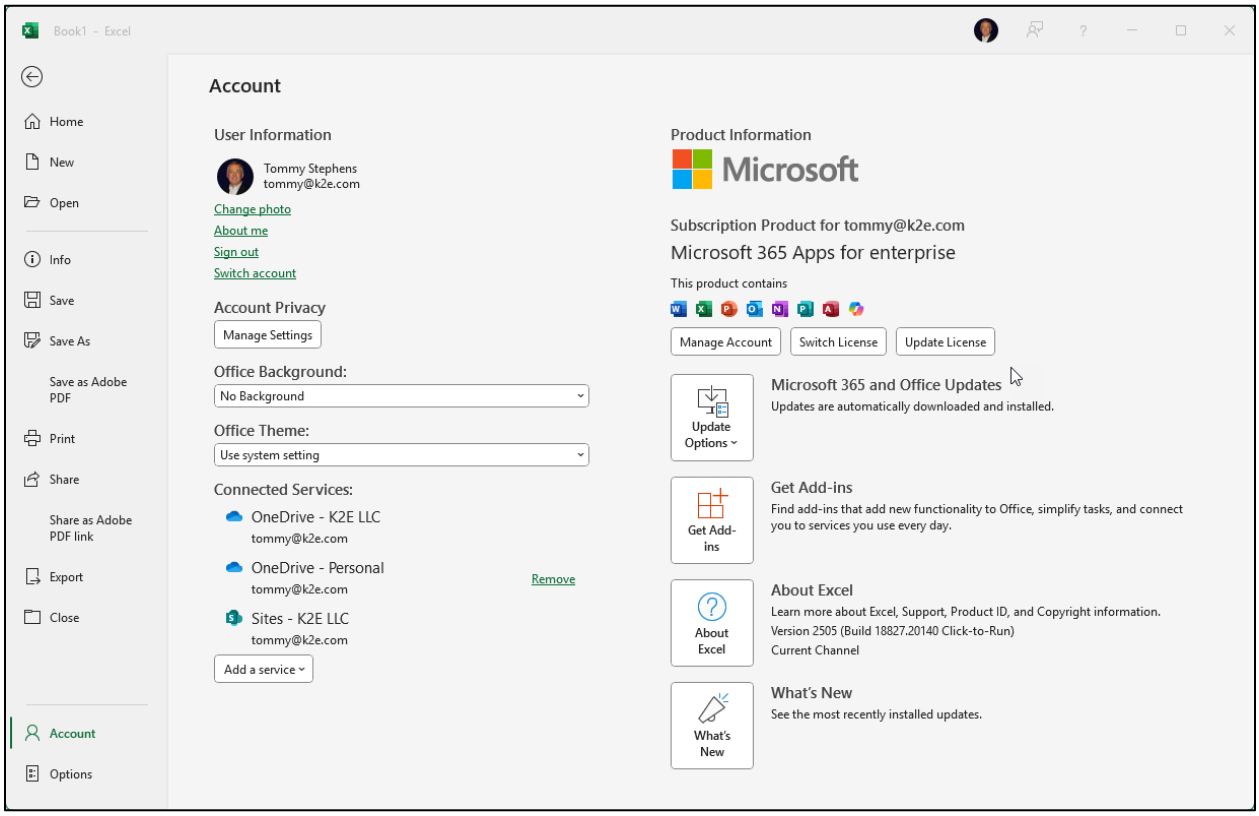


Figure 1 - Identifying Channel and Version

Also identified in Figure 1 is the *version number*, which is also necessary to understand new features in the subscription environment. The first two digits of the version number represent the last two digits of the year Microsoft published it. Likewise, the second two digits of the version number represent the month when Microsoft issued the version. Thus, “2505” translates into the May 2025 version.

Knowing the channel and the version number allows you to identify which features became available in that release. To do so, visit the following website:

https://docs.microsoft.com/en-us/officeupdates/update-history-microsoft365-apps-by-date

Next, select your channel from the window's left side, followed by **Release Notes**. Then scroll through the releases until you locate your version number. Upon doing so, as shown in **Figure 2**, you can see what new features became available in that version of Excel and other apps.

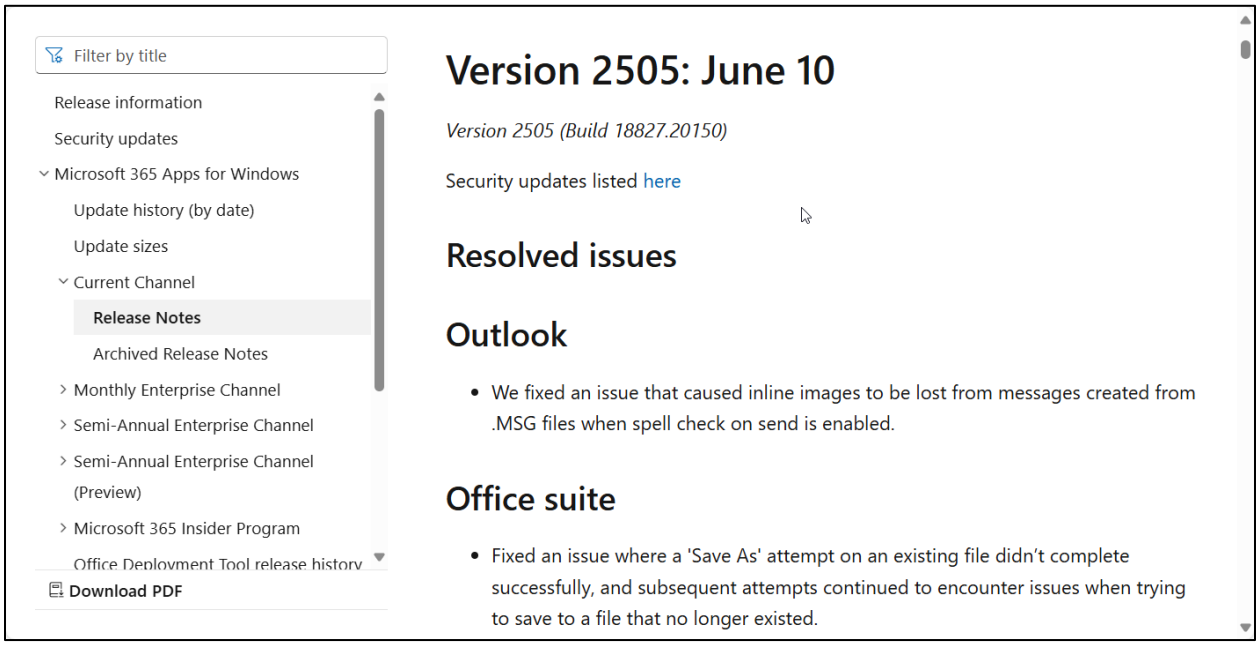


Figure 2 - Office Versions And New Features

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. However, do not confuse a dynamic array with a table, as these two features are distinctly 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 altering the original data, which is not possible 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 enable us to break free from the "one-cell, one formula" mentality that has traditionally prevailed 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 3**. 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.

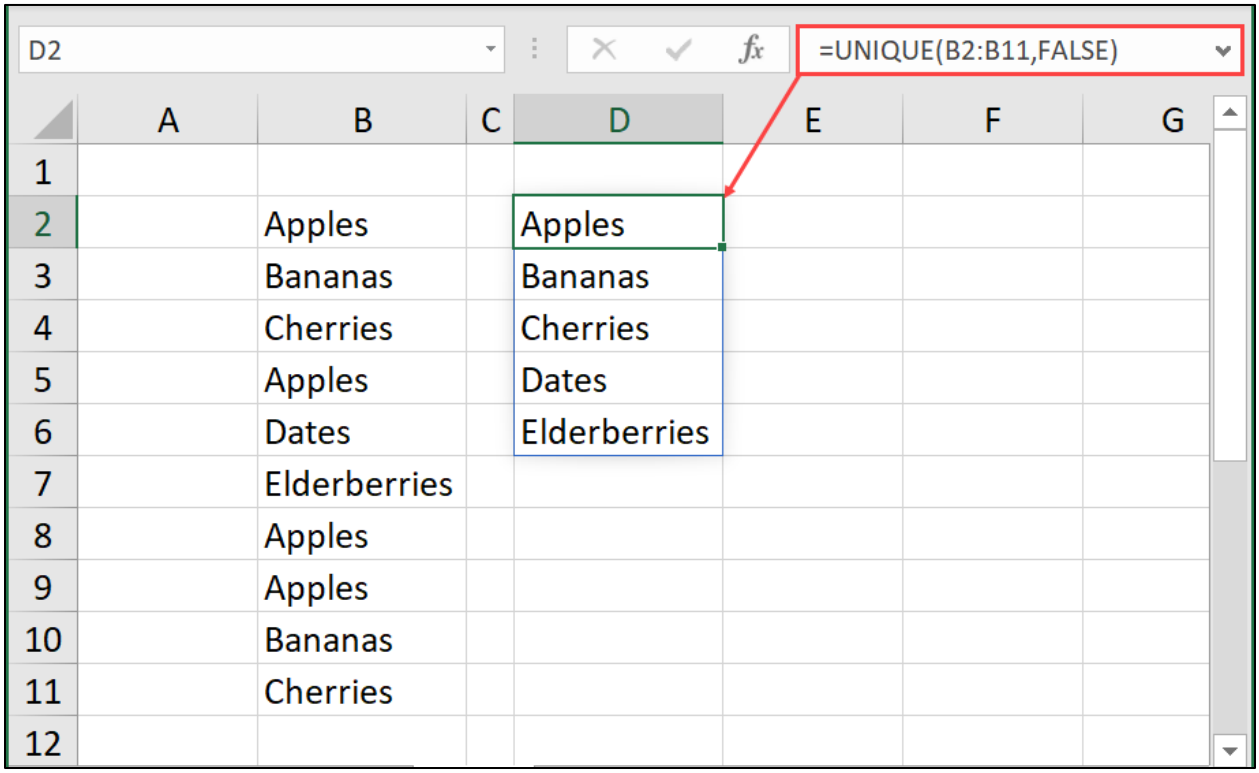


Figure 3 – First Example of a Formula Based on a Dynamic Array

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 3 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. The following are 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.

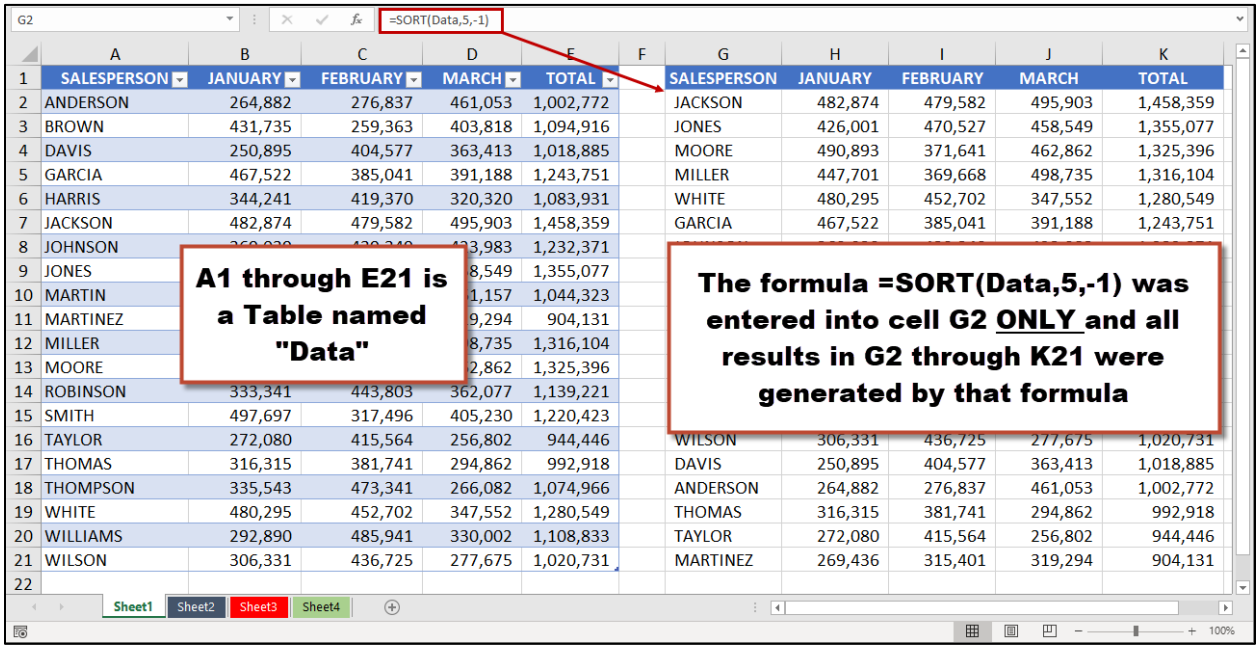
5. You can use the **SEQUENCE** function to generate a listing of sequential numbers displayed in an array.

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 4**. 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.



SALESPERSON	JANUARY	FEBRUARY	MARCH	TOTAL
ANDERSON	264,882	276,837	461,053	1,002,772
BROWN	431,735	259,363	403,818	1,094,916
DAVIS	250,895	404,577	363,413	1,018,885
GARCIA	467,522	385,041	391,188	1,243,751
HARRIS	344,241	419,370	320,320	1,083,931
JACKSON	482,874	479,582	495,903	1,458,359
JOHNSON			3,983	1,232,371
JONES			8,549	1,355,077
MARTIN			1,157	1,044,323
MARTINEZ			9,294	904,131
MILLER			8,735	1,316,104
MOORE			2,862	1,325,396
ROBINSON	333,341	443,803	362,077	1,139,221
SMITH	497,697	317,496	405,230	1,220,423
TAYLOR	272,080	415,564	256,802	944,446
THOMAS	316,315	381,741	294,862	992,918
THOMPSON	335,543	473,341	266,082	1,074,966
WHITE	480,295	452,702	347,552	1,280,549
WILLIAMS	292,890	485,941	330,002	1,108,833
WILSON	306,331	436,725	277,675	1,020,731

SALESPERSON	JANUARY	FEBRUARY	MARCH	TOTAL
JACKSON	482,874	479,582	495,903	1,458,359
JONES	426,001	470,527	458,549	1,355,077
MOORE	490,893	371,641	462,862	1,325,396
MILLER	447,701	369,668	498,735	1,316,104
WHITE	480,295	452,702	347,552	1,280,549
GARCIA	467,522	385,041	391,188	1,243,751
WILSON	306,331	436,725	277,675	1,020,731
DAVIS	250,895	404,577	363,413	1,018,885
ANDERSON	264,882	276,837	461,053	1,002,772
THOMAS	316,315	381,741	294,862	992,918
TAYLOR	272,080	415,564	256,802	944,446
MARTINEZ	269,436	315,401	319,294	904,131

Figure 4 – 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 is expected to 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.

Date And Time Arithmetic In Excel

A date is a number formatted to look like a date in Excel. To be precise, a date is a serial number representing the number of days since January 1, 1900, with "1" representing January 1, 1900, and "2" representing January 2, 1900, and so on. Similarly, 41,274 represents December 31, 2012.

Because dates in Excel are simply numbers formatted to look like dates, it is easy to deal with dates in formulas. For example, you can easily subtract the earlier date from the later date to compute the number of days between two dates. Accountants and other financial professionals can utilize this feature to calculate items such as accrued or deferred revenues and expenses. For example, **Figure 5** displays a worksheet that uses date arithmetic to compute accrued interest on notes.

	A	B	C	D	E
1	**ABC Holding Company**				
2	**Accrued Interest on Notes**				
3	**For the Date Ended:**				
4	**December 31, 2022**				
5					
6	**Days in Year:**	**365**			
7	**Date Issued**	**Due Date**	**Amount**	**Interest Rate**	**Accrued Interest**
8	6/19/2021	6/19/2023	$ 360,000.00	4.20%	$ 23,197.81
9	7/21/2021	7/21/2023	210,000.00	3.90%	11,847.45
10	8/15/2021	8/15/2023	150,000.00	4.30%	8,888.63
11	10/1/2021	10/1/2023	450,000.00	4.40%	24,736.44
12	10/15/2021	10/15/2023	850,000.00	4.60%	47,348.49
13	10/21/2021	10/21/2023	320,000.00	4.80%	18,347.84
14			$ 2,340,000.00		$ 134,366.66
15					

Date Arithmetic

Figure 5 - Date And Time Arithmetic In Excel

Table 2 below provides a listing of some of the commonly used date functions in Excel.

Function	Description
TODAY()	Returns the current date (no time).
DATE(year, month, day)	Creates a date from individual year, month, and day values.
DATEVALUE(date_text)	Converts a date in text format to a serial number.
YEAR(date)	Returns the year from a date.
MONTH(date)	Returns the month number from a date (1 to 12).
DAY(date)	Returns the day of the month from a date.
WEEKDAY(date, [return_type])	Returns the day of the week as a number (1 = Sunday by default).
EOMONTH(start_date, months)	Returns the last day of the month a number of months before or after a date.
DAYS(end_date, start_date)	Returns the number of days between two dates.

Table 2 – Summary Of Common Date Functions In Excel

Move Over VLOOKUP – XLOOKUP Is Here!

Many Excel users now have access to **XLOOKUP**, the successor to VLOOKUP, HLOOKUP, and potentially INDEX and MATCH. XLOOKUP offers enhanced functionality and addresses many of the limitations of previous functions. For example, XLOOKUP defaults to an exact match and can perform lookups from top to bottom or bottom to top. Further, you no longer need to sort the lookup column or row for approximate matches. It also allows users to specify a range of cells on which to perform the lookup instead of a column number in a lookup table. Hence, the order of the table columns in a lookup table does not matter. In addition, this feature allows XLOOKUP to look to the left of the lookup column – something VLOOKUP cannot do. You can see the full syntax of XLOOKUP below.

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

where:

Lookup value represents the value to be looked up.

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 related value.

If not found (optional) represents what to return if the lookup value is not found; defaults to #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

We will use XLOOKUP to perform a reverse lookup in this first example. Since XLOOKUP uses a lookup array (column or row) instead of a lookup table, you can execute the lookup on any column (or row) and return the result from any column (or row), even columns to the left of the lookup column (or rows above.)

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. Because the SKUs are in a column to the left of the column containing vendor part numbers, VLOOKUP cannot perform this task. Although you could use the **MATCH** and **INDEX** functions to perform this task, Debbie 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 that uses a vendor's part number to retrieve a Stock-Keeping Unit (SKU) from the Inventory list.

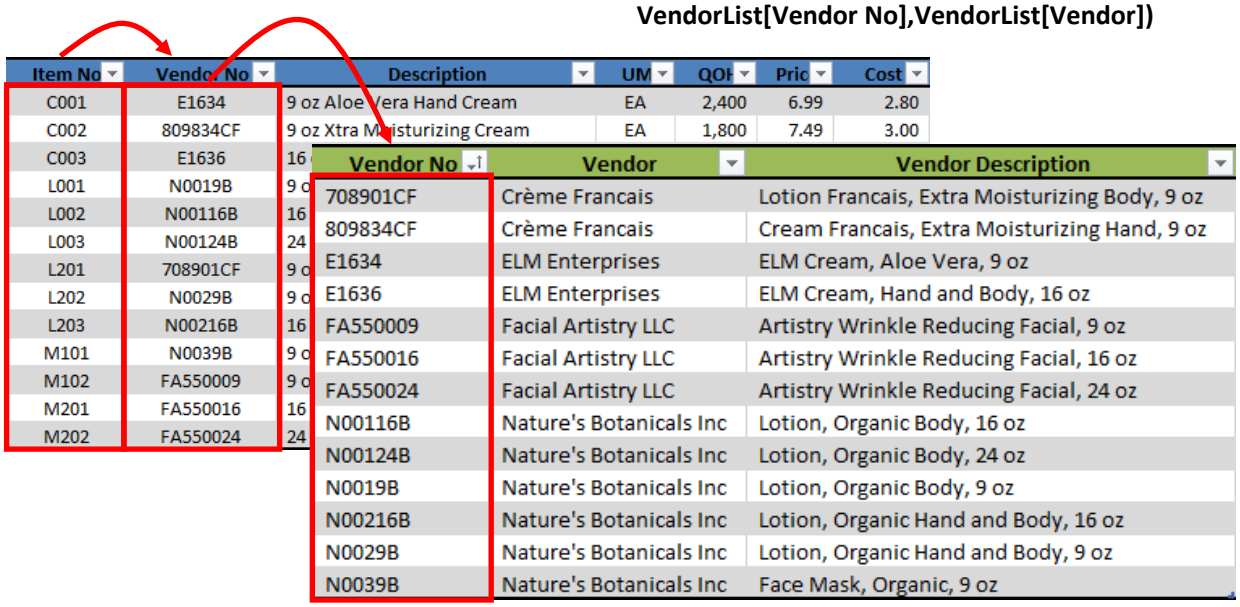
=XLOOKUP(C5,InventoryList[Vendor No],InventoryList[SKU])

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

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

Debbie needs to retrieve the vendor's name when reordering a product. Unfortunately, the vendor list is located in a separate 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])



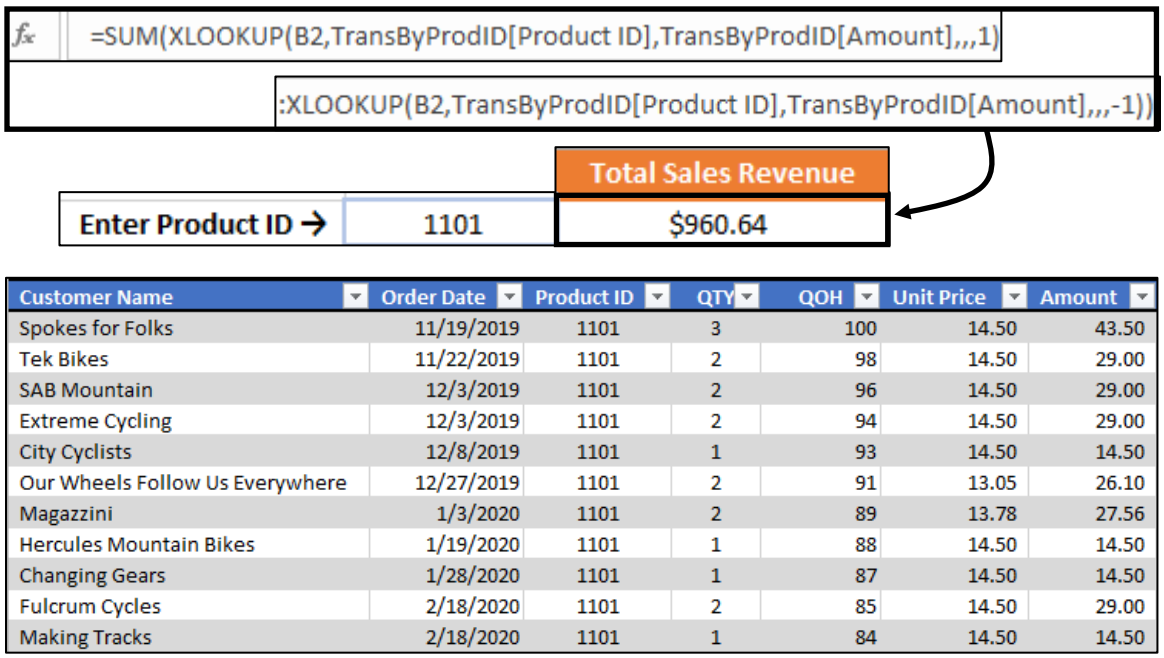
Item No	Vendor No	Description	UM	QOH	Price	Cost
C001	E1634	9 oz Aloe Vera Hand Cream	EA	2,400	6.99	2.80
C002	809834CF	9 oz Xtra Moisturizing Cream	EA	1,800	7.49	3.00
C003	E1636	16				
L001	N0019B	9 o				
L002	N00116B	16				
L003	N00124B	24				
L201	708901CF	9 o				
L202	N0029B	9 o				
L203	N00216B	16				
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 6 – XLOOKUP Can Look Across Worksheets or Workbooks

The formula shown in **Figure 6** 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 used in another XLOOKUP formula to return the vendor name from the Vendor List on a separate 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 these values as inputs to other functions, such as SUM, COUNT, AVERAGE, MIN, or MAX. For example, **Figure 7** displays a formula to sum 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 7 – Using XLOOKUP to Sum Sales of a Specified Product ID

Move over, VLOOKUP; your day in the sun is over! XLOOKUP is here, and it can do everything VLOOKUP can and more. With similar functionality advances, XLOOKUP and its companion XMATCH will revolutionize how you use lookup functions. They are easy to use, intuitive, and offer productivity-enhancing power.

Excel's STOCKHISTORY Function

Excel's long-awaited **STOCKHISTORY** function allows you to 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. Furthermore, if you select a range of dates, you can specify *daily*, *weekly*, or *monthly* intervals. STOCKHISTORY displays the date and closing price by default. However, you can optionally choose to show *opening price*, *high price*, *low price*, and *volume*, if desired.

Using STOCKHISTORY

The syntax for using the STOCKHISTORY function can be relatively simple, as indicated below. However, note that of the arguments available, only the *stock* and *start_date* are required. Thus, a formula using STOCKHISTORY could be as simple as **=STOCKHISTORY("MSFT","6/10/2025")**. Of course, this formula returns the closing price for a share of Microsoft stock on June 10, 2025.

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).

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

=STOCKHISTORY("MSFT","1/1/2025","5/31/2025")

To illustrate, **Figure** 8**8** below provides an abbreviated set of results from the formula shown above, with numerous rows hidden for presentation purposes.

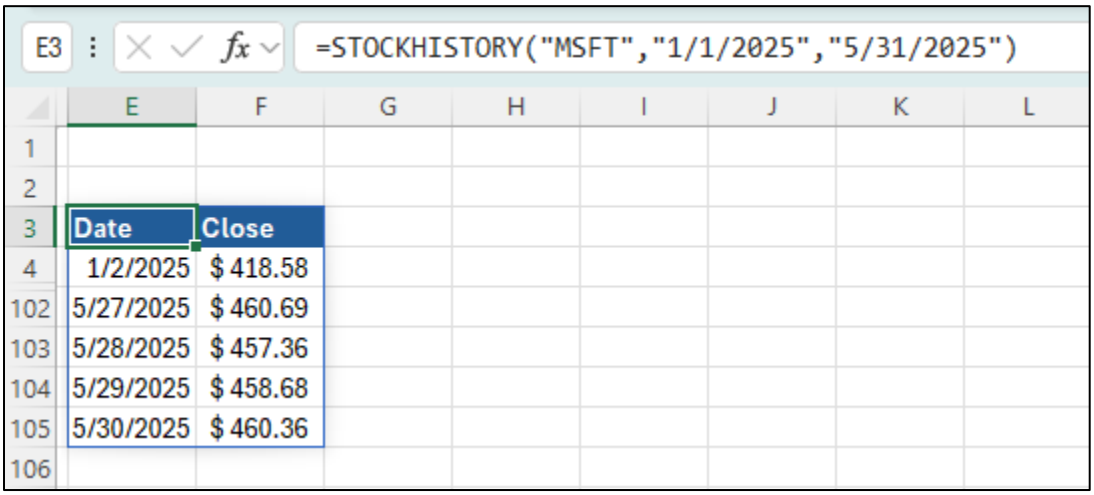


Figure 8 - Using STOCKHISTORY

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

STOCKHISTORY was 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. Importantly, you can use STOCKHISTORY to retrieve stocks' historical prices and incorporate them into other calculations in your spreadsheets. Therefore, the next time you need to perform research to obtain historical data about a stock, consider using STOCKHISTORY. Most importantly, if you do, you will reduce the time spent retrieving data.

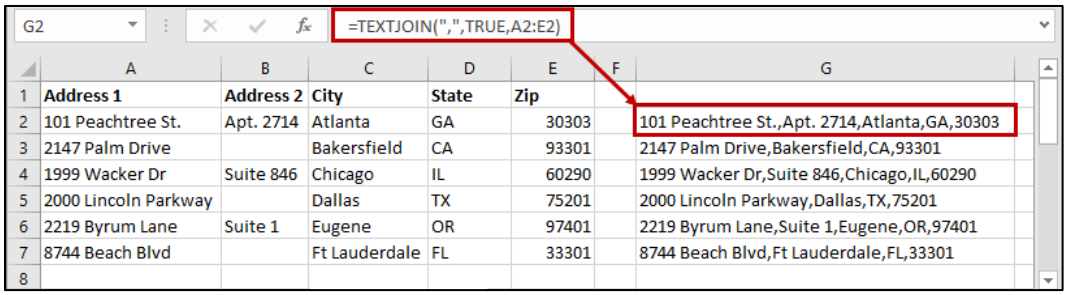
Six Functions for Easier Calculations

In 2016, Microsoft added six functions to Excel that many users will find helpful when creating formulas to manipulate and analyze data.

1. **TEXTJOIN**
2. **CONCAT**
3. **IFS**
4. **SWITCH**
5. **MAXIFS**
6. **MINIFS**

TEXTJOIN

You can use Excel's **TEXTJOIN** function to concatenate a list or range of text, which places a delimiting character, such as a space or comma, between each field it joins. **Figure 9** provides a simple illustration of where TEXTJOIN is used to concatenate five data columns into one. Notably, because the formula uses relative referencing, you can copy it to join data from other rows.



	A	B	C	D	E	F	G
1	**Address 1**	**Address 2**	**City**	**State**	**Zip**		
2	101 Peachtree St.	Apt. 2714	Atlanta	GA	30303		101 Peachtree St.,Apt. 2714,Atlanta,GA,30303
3	2147 Palm Drive		Bakersfield	CA	93301		2147 Palm Drive,Bakersfield,CA,93301
4	1999 Wacker Dr	Suite 846	Chicago	IL	60290		1999 Wacker Dr,Suite 846,Chicago,IL,60290
5	2000 Lincoln Parkway		Dallas	TX	75201		2000 Lincoln Parkway,Dallas,TX,75201
6	2219 Byrum Lane	Suite 1	Eugene	OR	97401		2219 Byrum Lane,Suite 1,Eugene,OR,97401
7	8744 Beach Blvd		Ft Lauderdale	FL	33301		8744 Beach Blvd,Ft Lauderdale,FL,33301
8							

Figure 9 - Using TEXTJOIN to Concatenate Multiple Columns of Data

The following is the syntax of a formula that uses the TEXTJOIN function.

=TEXTJOIN(delimiter, ignore_empty, text1, [text2], …)

Of note, you must specify the delimiting character(s); this can be a cell reference to a valid text string. You can also instruct TEXTJOIN on how to handle empty cells. If the formula's ignore_empty argument is blank or **TRUE** (as in Figure 9), TEXTJOIN ignores empty cells. For example, this trait can be beneficial when some addresses contain apartment or suite numbers and others do not. On the other hand, if the ignore_empty argument is **FALSE**, TEXTJOIN includes the blank cells in the formula's results.

CONCAT

The **CONCAT** function is an alternative to the legacy CONCATENATE function, joining multiple text strings. **Figure 10** presents an example of using CONCAT to combine numerous text strings. Compare the formula in Figure 9 to the one in Figure 10 and notice the relative brevity and simplicity of the TEXTJOIN function, particularly for how it inserts commas into the concatenated text string.

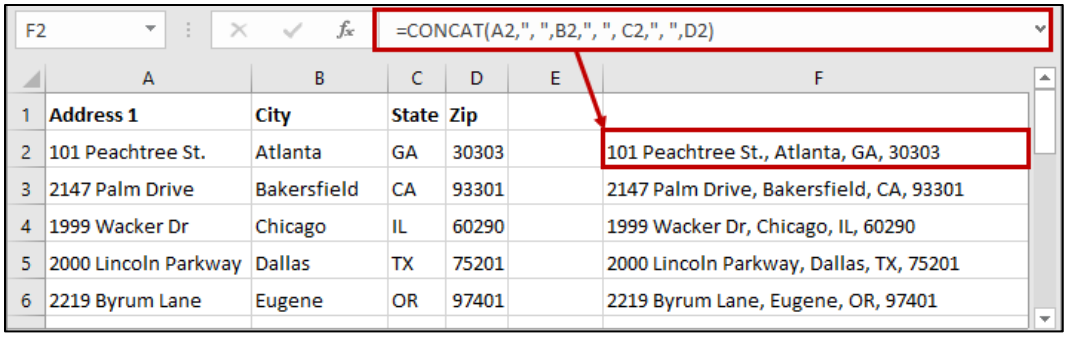


Figure 10 - Using Excel's CONCAT Function to Join Text Strings

IFS

Similar to Excel's **IF** function, the **IFS** function allows you to check whether one or more conditions are met; if so, the IFS function returns a value corresponding to the first true condition. Likewise, the IFS function returns **#N/A!** if none of the tested conditions are satisfied. **Figure 11** provides an example of how you can use the IFS function to simplify a process where you might have used nested IF functions in the same formula.

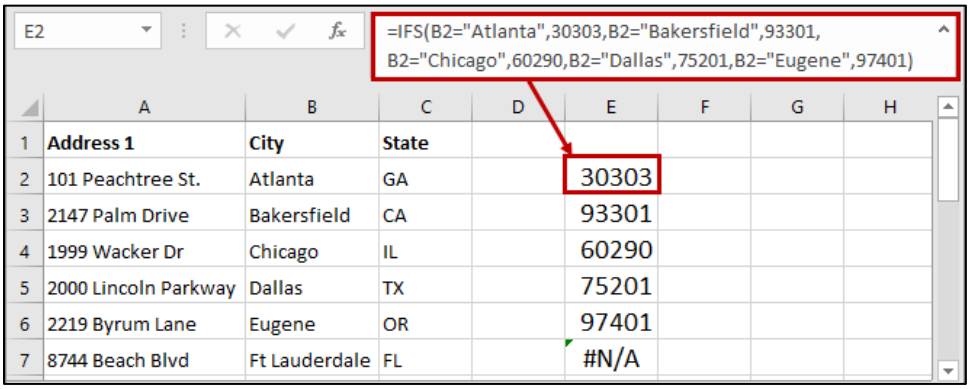
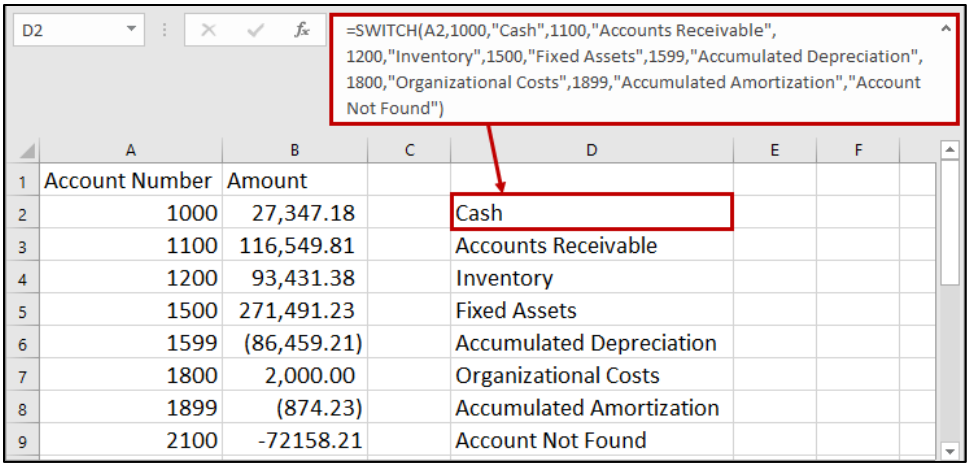


Figure 11 - Testing Multiple Conditions Using Excel's IFS Function

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 related to the first matching value. In addition, you can direct SWITCH to produce an optional default value if it does not find a match. To illustrate, consider the worksheet and formula based on the SWITCH function pictured in **Figure 12**. 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, such as *Cash* or *Accounts Receivable*. If SWITCH does not find an exact match, it returns *Account Not Found*. As with IFS, SWITCH can reduce your need for nested IF functions in a formula.



	A	B	C	D	E	F
1	Account Number	Amount				
2	1000	27,347.18		Cash		
3	1100	116,549.81		Accounts Receivable		
4	1200	93,431.38		Inventory		
5	1500	271,491.23		Fixed Assets		
6	1599	(86,459.21)		Accumulated Depreciation		
7	1800	2,000.00		Organizational Costs		
8	1899	(874.23)		Accumulated Amortization		
9	2100	-72158.21		Account Not Found		

Figure 12 - Using Excel's SWITCH Function Instead of Nested IF Functions

MAXIFS

The **MAXIFS** function allows you to identify and return the maximum value in a range of cells specified by a set of conditions or criteria you specify. Similar to **SUMIFS**, **COUNTIFS**, and **AVERAGEIFS**, the syntax for this function is

=MAXIFS(max_range, criteria_range1, criteria1, [criteria_range2, criteria2], ...)

where

max_range is the actual range of cells in which the maximum will be determined,

critera_range 1, **2, etc.** is the set of cells to evaluate with the criteria, and

criteria 1, 2, etc., are the criteria to evaluate.

To illustrate, consider the example pictured in **Figure 13** (rows hidden for presentation purposes). In this illustration, MAXIFS identifies the maximum value in the range of **B2:B25** for *Munoz*, the specified salesperson.

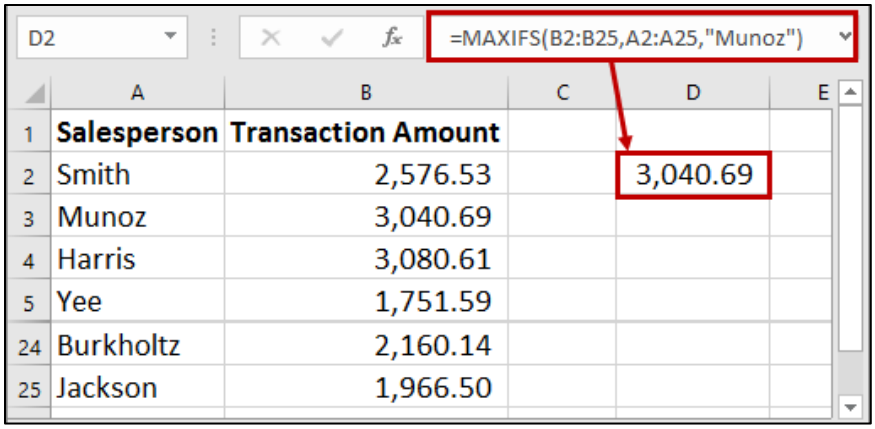


Figure 13 - Using MAXIFS to Identify the Largest Transaction for a Salesperson

Continuing with the example in Figure 13, a MAXIFS function can evaluate up to 127 criteria. For example, in **Figure 1**, MAXIFS calculates the result by examining two criteria: *Salesperson* and *Month*.

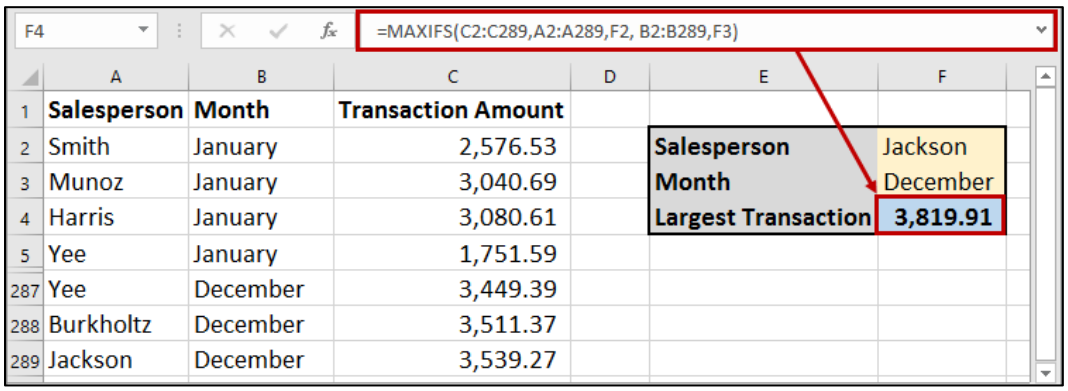


Figure 1 - MAXIFS Function: Identifying the Largest Value Based on Multiple Conditions

MINIFS

The **MINIFS** function is the opposite of the MAXIFS function – you can use MINIFS to identify the smallest value in a range based on up to 127 criteria. The syntax for MINIFS is identical to that of MAXIFS.

GROUPBY And PIVOTBY – Two Of Excel's Newest And Best Functions

Formula-Based Alternatives To PivotTables

In 1994, Microsoft added PivotTables to Excel. Since then, countless business professionals have adopted this feature as their *de facto* way of summarizing and analyzing information – for good reasons! A PivotTable offers powerful data summarization options without using time-consuming and error-prone formulas.

However, PivotTables have limitations. One such restriction is that they are rigid, with few options for manipulating and presenting their summarized data. To address this issue, Microsoft

recently added GROUPBY and PIVOTBY – two functions in Excel that offer the computational advantages of traditional PivotTables while also providing formula-based flexibility.

Getting Started With GROUPBY

Like Excel's **SUM**, **SUBTOTAL**, and other functions, GROUPBY is an Excel function you can use to summarize information quickly, easily, and accurately. However, GROUPBY differs from other Excel features as it allows you to quickly and easily aggregate data by categories, such as product line, region, and customer. In other words, you can use GROUPBY to generate a data summary filtered to specific criteria.

Although up to eight arguments can be present in the formula, only three are necessary when creating a formula based on the GROUPBY function.

1. First, indicate the row(s) in a table or range by which you will summarize your data; in the formula below, the **(FinancialData[Segment])** argument serves that purpose. Note that you can specify multiple rows; if you do, you will have numerous groupings in your formula results.

2. Second, specify the column(s) that serves as the numeric values you wish to summarize; in the example below, the **(FinancialData[Net Sales])** argument serves that purpose. Like rows, you can specify multiple columns; if you do, your report will have multiple aggregations.

3. Finally, specify the aggregation function; in the example below, that is the **SUM** argument.

To illustrate, consider the formula shown below.

=GROUPBY(FinancialData[Segment],FinancialData[Net Sales],SUM)

You could use this formula to summarize the amounts in FinancialData's "Net Sales" column by segment. Upon creating the formula, your output might resemble that shown in **Figure 15**.

Channel Partners	$	1,800,593.64
Enterprise	$	19,611,694.38
Government	$	52,504,260.67
Midmarket	$	2,381,883.08
Small Business	$	42,427,918.50
Total	$	118,726,350.26

Figure 15 - Output Generated By Simple GROUPBY Formula

In this simple example, the GROUPBY function creates summaries of the sales data for each of the specified business segments. You do not need to sort, filter, or otherwise manipulate or rearrange the data to use GROUPBY.

Multiple Grouping Levels

Next, assume you need to summarize your data not only by **Segment** but also by **Country**. In that case, you can modify your formula to include **Country** as a second summarizing criterion. The modified formula appears below.

=GROUPBY(FinancialData[[Country]:[Segment]],FinancialData[Net Sales],SUM)

Upon entering the formula outlined above, the output changes to that pictured in **Figure 15**.

Channel Partners	Canada	$	491,164.14
Channel Partners	France	$	372,090.36
Channel Partners	Germany	$	336,425.88
Channel Partners	Mexico	$	234,379.08
Channel Partners	United States of America	$	366,534.18
Enterprise	Canada	$	3,967,491.25
Enterprise	France	$	3,890,890.63
Enterprise	Germany	$	4,086,826.25
Enterprise	Mexico	$	3,315,881.25
Enterprise	United States of America	$	4,350,605.00
Government	Canada	$	10,741,236.52
Government	France	$	12,127,782.72
Government	Germany	$	11,452,895.94
Government	Mexico	$	9,791,599.38
Government	United States of America	$	8,390,746.11
Midmarket	Canada	$	510,213.98
Midmarket	France	$	593,802.08
Midmarket	Germany	$	301,344.75
Midmarket	Mexico	$	511,136.40
Midmarket	United States of America	$	465,385.88
Small Business	Canada	$	9,177,549.00
Small Business	France	$	7,369,606.50
Small Business	Germany	$	7,327,848.00
Small Business	Mexico	$	7,096,356.00
Small Business	United States of America	$	11,456,559.00
Total		$	118,726,350.26

Figure 15 - GROUPBY Formula With Two Summarizing Criteria

Summarizing Multiple Columns Of Data By Multiple Criteria Using GROUPBY

Continuing with the same data, let us use GROUPBY to summarize multiple columns of data using various criteria. This illustration will summarize **Net Sales** and **Gross Profit** by **Segment** and **Country**. Sometimes, it may be necessary to rearrange your data when summarizing multiple columns using GROUPBY. Specifically, the summarized columns must be contiguous when using

this function. Therefore, if the summarizing columns are not contiguous, rearrange them in that order. Alternatively, you could insert a new "helper" column contiguous to the other columns of interest and use a formula to link the value from the original column into the helper column. We can use the following formula to summarize multiple columns by multiple criteria – precisely, Segment followed by Country, and Net Sales followed by Gross Profit. **Figure 16** illustrates the formula's results.

=GROUPBY(FinancialData[[Country]:[Segment]],
FinancialData[[Net Sales]:[Gross Profit]],SUM)

Channel Partners	Canada	$ 491,164.14	$ 358,978.14
Channel Partners	France	$ 372,090.36	$ 271,581.36
Channel Partners	Germany	$ 336,425.88	$ 247,358.88
Channel Partners	Mexico	$ 234,379.08	$ 170,890.08
Channel Partners	United States of America	$ 366,534.18	$ 267,994.68
Enterprise	Canada	$ 3,967,491.25	$ (121,508.75)
Enterprise	France	$ 3,890,890.63	$ (95,749.38)
Enterprise	Germany	$ 4,086,826.25	$ (101,473.75)
Enterprise	Mexico	$ 3,315,881.25	$ (120,678.75)
Enterprise	United States of America	$ 4,350,605.00	$ (175,135.00)
Government	Canada	$ 10,741,236.52	$ 2,258,471.52
Government	France	$ 12,127,782.72	$ 2,709,915.22
Government	Germany	$ 11,452,895.94	$ 2,677,175.94
Government	Mexico	$ 9,791,599.38	$ 2,039,159.38
Government	United States of America	$ 8,390,746.11	$ 1,703,451.11
Midmarket	Canada	$ 510,213.98	$ 132,488.98
Midmarket	France	$ 593,802.08	$ 164,542.08
Midmarket	Germany	$ 301,344.75	$ 85,354.75
Midmarket	Mexico	$ 511,136.40	$ 150,546.40
Midmarket	United States of America	$ 465,385.88	$ 127,170.88
Small Business	Canada	$ 9,177,549.00	$ 900,799.00
Small Business	France	$ 7,369,606.50	$ 730,731.50
Small Business	Germany	$ 7,327,848.00	$ 771,973.00
Small Business	Mexico	$ 7,096,356.00	$ 667,606.00
Small Business	United States of America	$ 11,456,559.00	$ 1,072,059.00
Total		**$ 118,726,350.26**	**$ 16,893,702.26**

Figure 16 - Summarizing Multiple Columns Of Data Using GROUPBY

Optional Arguments For GROUPBY Summarizations

As indicated earlier, there are only three mandated arguments for using the GROUPBY function: 1) row fields, 2) values, and 3) the summarizing function, such as SUM, COUNT, and AVERAGE. However, you can use up to five optional arguments to enhance your formula's results. We discuss each of these discussed below.

1. The *field_headers* argument allows you to specify whether your *row_fields* and *values* have headers and whether those headers should appear in your formula's results.
2. The *total_depth* argument determines whether row headers should contain totals.
3. You can use the *sort_order* argument to determine how Excel should sort rows in your formula's results.
4. A *filter_array* indicates whether to consider a corresponding row of data when filtering.
5. The *field_relationship* argument specifies the relationship fields when multiple columns are provided to row fields.

Although a complete discussion of each of the five options is beyond the scope of this session, a simple example can help you to understand the power of these options. For instance, suppose you want the output shown in Figure 1 to display sorted in *descending* order based on the summarized values. To achieve that result, you could modify the formula as shown below to indicate that you want the data sorted on the second column (thus, the entry of "2" in the formula) and, because you prefaced your entry of "2" with a minus sign, Excel knows to sort the data in *descending* order.

=GROUPBY(FinancialData[Segment],FinancialData[Net Sales],SUM,,,-2)

You can learn more about GROUPBY's optional arguments at https://k2e.fyi/groupby.

PIVOTBY – An Even More Flexible And Powerful Tool

Excel's **PIVOTBY** function provides even more flexibility and utility than the GROUPBY function. Specifically, PIVOTBY allows you to create data summaries and reports using *any* Excel function. Fortunately, the process for using PIVOTBY is like that of using GROUPBY, so if you know how to work with GROUPBY, the learning curve to adapt to PIVOTBY is short.

When using PIVOTBY, Excel requires you to specify four arguments:

1. **Row fields**, to describe which column of data to use as row labels,
2. **Column fields**, to identify which column of data to use as column headers,
3. **Values**, to indicate which column of data contains the data to summarize, and
4. **Function**, the summarizing function you want to use to analyze the data.

Using the data from the previous example, we could build the following formula to summarize our sales data by channel and country. The formula to accommodate that need appears below, along with the report generated by the formula.

=PIVOTBY(FinancialData[Segment],FinancialData[Country],FinancialData[Net Sales],SUM)

	Canada	France	Germany	Mexico	United States of America	Total
Channel Partners	491,164.14	372,090.36	336,425.88	234,379.08	366,534.18	1,800,593.64
Enterprise	3,967,491.25	3,890,890.63	4,086,826.25	3,315,881.25	4,350,605.00	19,611,694.38
Government	10,741,236.52	12,127,782.72	11,452,895.94	9,791,599.38	8,390,746.11	52,504,260.67
Midmarket	510,213.98	593,802.08	301,344.75	511,136.40	465,385.88	2,381,883.08
Small Business	9,177,549.00	7,369,606.50	7,327,848.00	7,096,356.00	11,456,559.00	42,427,918.50
Total	**24,887,654.89**	**24,354,172.28**	**23,505,340.82**	**20,949,352.11**	**25,029,830.17**	**118,726,350.26**

Figure 17 - Sample Report Generated With Excel's PIVOTBY Function

If necessary, you could use any of the seven available optional arguments to modify your PIVOTBY formula so that the results meet your needs better. Several of these are similar to the optional arguments available with GROUPBY, and you can learn more about PIVOTBY's optional arguments at https://k2e.fyi/pivotby.

For example, suppose we modify the original formula to that shown below.

=PIVOTBY(FinancialData[Segment],FinancialData[Country],FinancialData[Net Sales],SUM,,-1)

In this case, we have added two commas and a "-1" to the end of the PIVOTBY formula. That modification causes Excel to display the PivotTable with the Grand Total as the first numerical row of the report instead of the last.

	Canada	France	Germany	Mexico	United States of America	Total
Total	24,887,654.89	24,354,172.28	23,505,340.82	20,949,352.11	25,029,830.17	118,726,350.26
Channel Partners	491,164.14	372,090.36	336,425.88	234,379.08	366,534.18	1,800,593.64
Enterprise	3,967,491.25	3,890,890.63	4,086,826.25	3,315,881.25	4,350,605.00	19,611,694.38
Government	10,741,236.52	12,127,782.72	11,452,895.94	9,791,599.38	8,390,746.11	52,504,260.67
Midmarket	510,213.98	593,802.08	301,344.75	511,136.40	465,385.88	2,381,883.08
Small Business	**9,177,549.00**	**7,369,606.50**	**7,327,848.00**	**7,096,356.00**	**11,456,559.00**	**42,427,918.50**

Figure 18 - Sample PIVOTBY-Based Formula With Optional Argument

Although PivotTables remain firmly entrenched in many business professionals' Excel repertoire, GROUPBY and PIVOTBY are welcome additions to the lineup. Notably, because they are both formula-based, these functions update dynamically as your data changes – there is no need to initiate a refresh, such as with a PivotTable.

Building Charts Using GROUPBY And PIVOTBY Outputs

One ancillary advantage of using the GROUPBY and PIVOTBY functions is the ability to create dynamic charts linked to the aggregations produced using these functions. Specifically, you can take the values produced by GROUPBY and PIVOTBY and link them directly into an Excel chart. This may not seem significant at first glance, given the presence of PivotCharts and the ability to generate them from PivotTables. However, there is a considerable limitation associated with PivotCharts. Specifically, Excel does not provide access to all chart types if you attempt to create a PivotChart. For example, suppose you created a PivotTable and wanted to create a corresponding PivotChart in the form of a Treemap. If you attempt to do so, Excel displays the message in **Figure 19**.

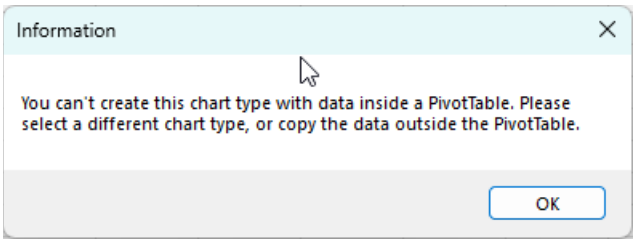


Figure 19 - Error Message Associated With Attempting To Create Certain Types Of Charts As PivotCharts

This message arises because Excel excludes the following chart types from the PivotChart library.

- Box and Whisker
- Funnel
- Histograms
- Map
- Stock
- Sunburst
- Treemaps

Fortunately, there is a simple solution to this issue. Specifically, you can use GROUPBY or PIVOTBY to create your aggregations and then build your chart from those aggregations. When you use this method, you avoid the limitations imposed by the absence of the seven chart types listed above.

To illustrate, let's return to the FinancialData table used earlier in this session; the PivotTable pictured in **Figure 20** summarizes the data from that table.

Sum of Net Sales	Column Labels					
Row Labels	Canada	France	Germany	Mexico	United States of America	Grand Total
Government	$ 10,741,236.52	$ 12,127,782.72	$ 11,452,895.94	$ 9,791,599.38	$ 8,390,746.11	$ 52,504,260.67
Small Business	$ 9,177,549.00	$ 7,369,606.50	$ 7,327,848.00	$ 7,096,356.00	$ 11,456,559.00	$ 42,427,918.50
Enterprise	$ 3,967,491.25	$ 3,890,890.63	$ 4,086,826.25	$ 3,315,881.25	$ 4,350,605.00	$ 19,611,694.38
Midmarket	$ 510,213.98	$ 593,802.08	$ 301,344.75	$ 511,136.40	$ 465,385.88	$ 2,381,883.08
Channel Partners	$ 491,164.14	$ 372,090.36	$ 336,425.88	$ 234,379.08	$ 366,534.18	$ 1,800,593.64
Grand Total	**$ 24,887,654.89**	**$ 24,354,172.28**	**$ 23,505,340.82**	**$ 20,949,352.11**	**$ 25,029,830.17**	**$ 118,726,350.26**

Figure 20 - PivotTable Created From "FinancialData" Table

If we wanted the data displayed as a Treemap chart, we could not complete that task because, as shown below, Treemaps are unavailable as PivotCharts.

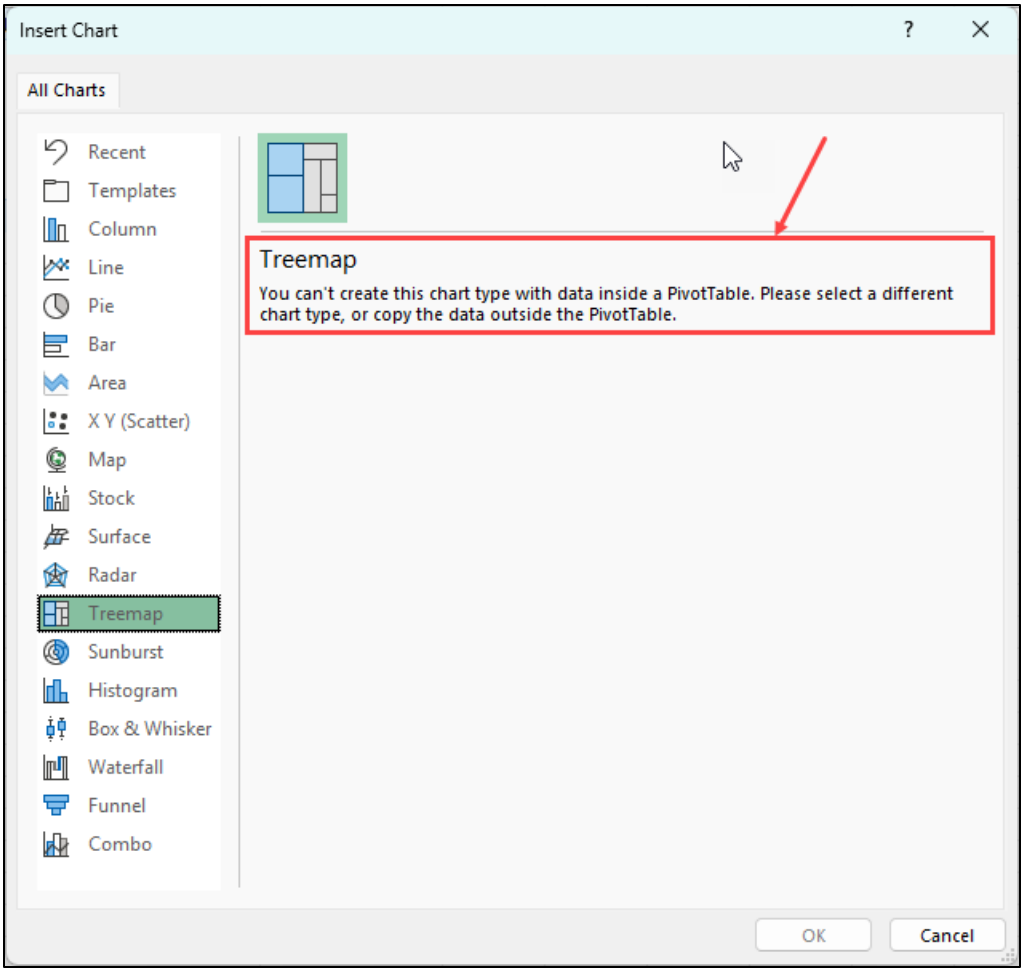


Figure 21 - Sample "Blocked" Treemap Chart

However, we could change our approach and use the results of a GROUPBY-based formula to achieve our desired results. Specifically, we could create the following formula:

GROUPBY(FinancialData[Segment],FinancialData[Net Sales],SUM,,0,-2)

The formula above creates the data summary pictured in **Figure 22**.

Government	52,504,260.67
Small Business	42,427,918.50
Enterprise	19,611,694.38
Midmarket	2,381,883.08
Channel Partners	1,800,593.64

Figure 22 - Data Summary Produced By GROUPBY-based Formula

With the tabulation complete, we could quickly generate the chart pictured in **Figure 23**, providing a graphical representation of the data. Again, note that this would not be possible with a PivotChart.

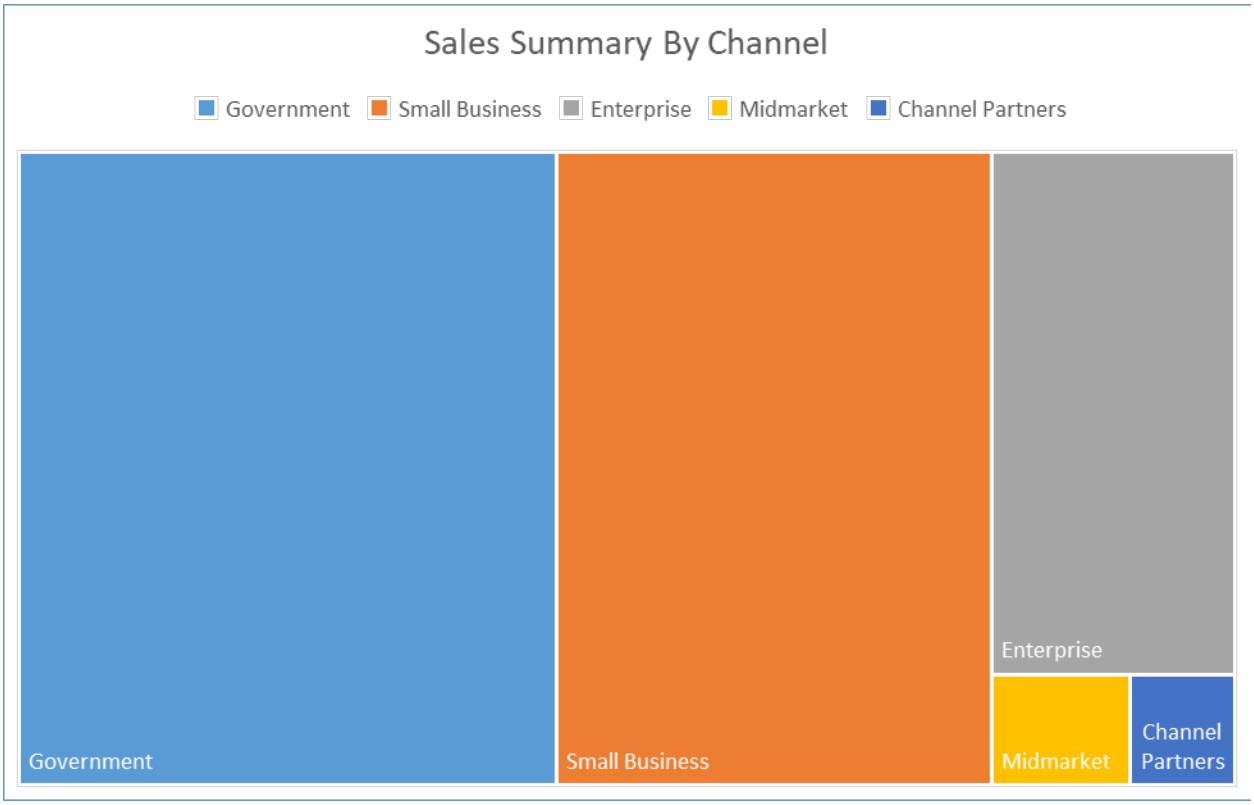


Figure 23 - Using Excel's GROUPBY Function To Bypass PivotChart Limitations

Power Query

Beginning with Excel 2010, Microsoft provided a new tool – **Power Query** – that makes connecting to external data sources relatively easy, even for novices. Power Query enables end-users to connect to virtually all data sources and link the data from those sources into Excel. Once you link the data into Excel, you can act using any available feature, including summarizing the data with formulas, building PivotTables from the data, and creating charts and other visualizations.

One of the most crucial advantages of Power Query is that it facilitates queries from many data sources, including the forty data sources indicated in **Table 1**. In particular, notice the "non-standard" data sources such as PDF documents, Exchange, and Salesforce. These capabilities enable users with limited technical skills to quickly and easily extract data from their accounting, ERP, CRM, practice management, or other line-of-business databases into Excel for reporting and data analysis. Moreover, Power Query's capabilities allow you to automate transforming – cleaning up – your data as part of the query process.

Files	Databases	Azure	Online Services	Other Sources
Excel	SQL Server	Azure SQL Database	SharePoint Online List	Table/ Range
Text/CSV	Access	Azure Synapse Data Analytics	Microsoft Exchange Online	Web
XML	Analysis Services	Azure HDInsight	Dynamics 365	Microsoft Query
JSON	SQL Server Analysis Services Database (Import)	Azure Blob Storage	Salesforce Objects	SharePoint List
Folder	Oracle	Azure Table Storage	Salesforce Reports	OData Feed
SharePoint Folder	IBM DB2	Azure Data Lake Store		Hadoop File (HDFS)
PDFs	MySQL	Azure Data Explorer		Active Directory
Pictures	PostgreSQL			Microsoft Exchange
	Sybase			ODBC
	Teradata			OLEDB

Files	Databases	Azure	Online Services	Other Sources
	SAP Hana			Blank Query

Table 1 - Potential Data Sources for Power Query

The following section explores several examples of using Power Query to load data into Excel. While an example for each compatible data type is not practical, the examples chosen will provide you with the background you need to query data from virtually any data source into Excel.

Querying Data from SQL Server

As shown in **Figure 24,** to launch a new query with Power Query, click **Get Data** and choose your data source; in this example, the data source will be a SQL Server database that serves an organization's ERP system.

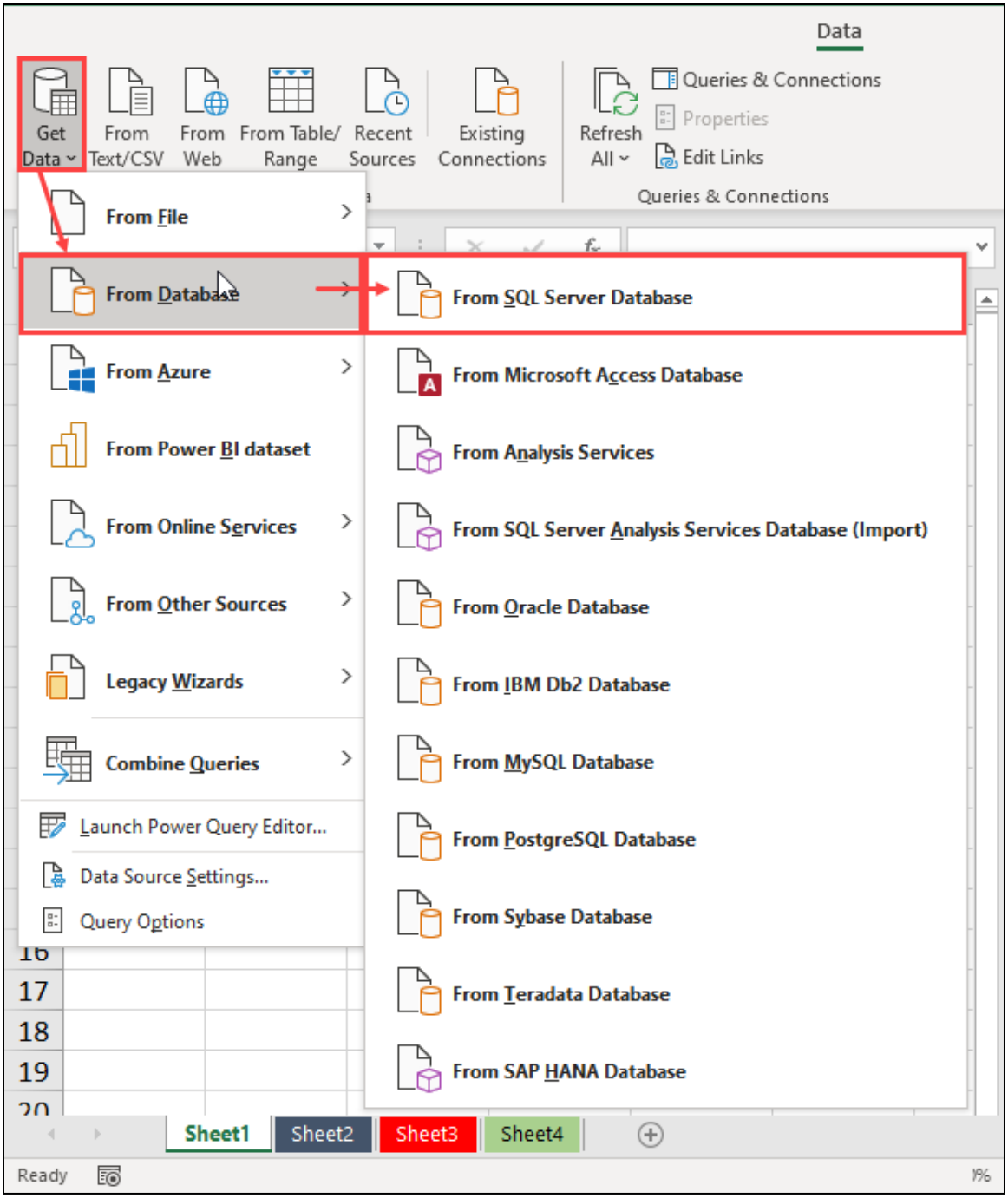


Figure 24 - Initiating a New Query Using Power Query in Excel

In the dialog box pictured in **Figure 25**, enter the necessary details to identify the correct data source for the query. Click **OK** to continue.

Figure 25 - Specifying the Data Source for Power Query

Next, expand the name of the database you wish to query so that you can see all of the data available for the query. Then, select the items to include in your query and choose the appropriate **Load** option in the Navigator window's bottom right corner, as shown in **Figure 26.** When choosing Load options, you can add the data to a table(s) in the workbook or create only the data connection. You can specify whether the data should populate the workbook's Data Model regardless of your choice.

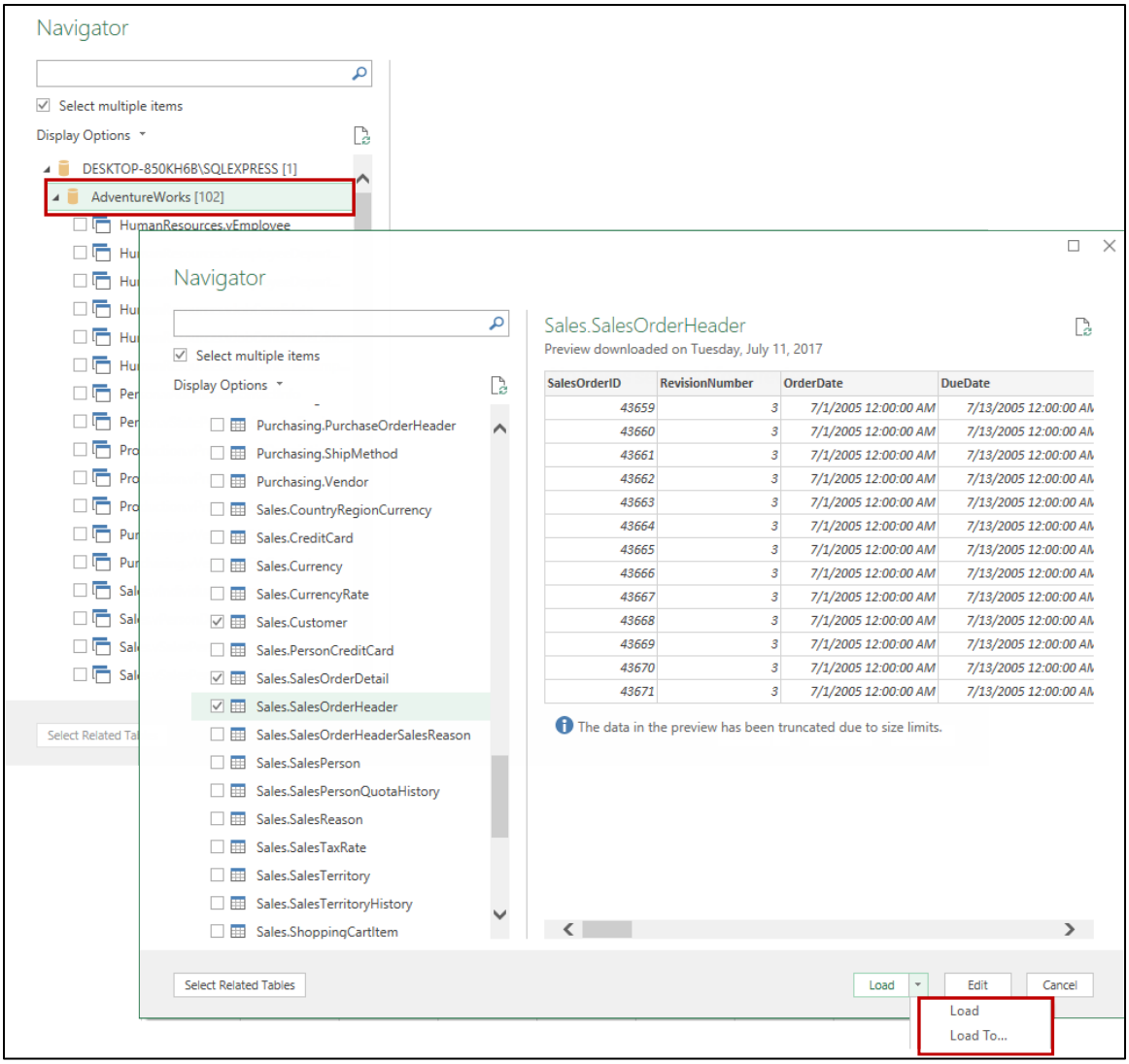
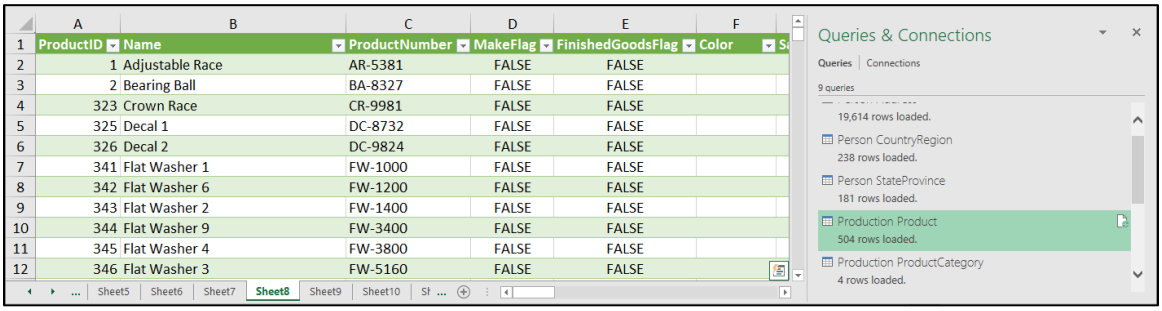


Figure 26 - Select Data for Query in Power Query

We selected nine tables for query in the example shown above. Additionally, we made the election to load the data to worksheets in the workbook and the workbook's Data Model. **Figure 27** presents a snapshot of the query's results.



	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color
2	1	Adjustable Race	AR-5381	FALSE	FALSE	
3	2	Bearing Ball	BA-8327	FALSE	FALSE	
4	323	Crown Race	CR-9981	FALSE	FALSE	
5	325	Decal 1	DC-8732	FALSE	FALSE	
6	326	Decal 2	DC-9824	FALSE	FALSE	
7	341	Flat Washer 1	FW-1000	FALSE	FALSE	
8	342	Flat Washer 6	FW-1200	FALSE	FALSE	
9	343	Flat Washer 2	FW-1400	FALSE	FALSE	
10	344	Flat Washer 9	FW-3400	FALSE	FALSE	
11	345	Flat Washer 4	FW-3800	FALSE	FALSE	
12	346	Flat Washer 3	FW-5160	FALSE	FALSE	

Figure 27 - Results of Power Query of Nine Tables

Power Query created nine independent queries – one for each queried table – in this example. If necessary, you can edit a query once you create it by right-clicking on the query in the **Workbook Queries** pane and choosing the **Edit** option, as shown in **Figure** .

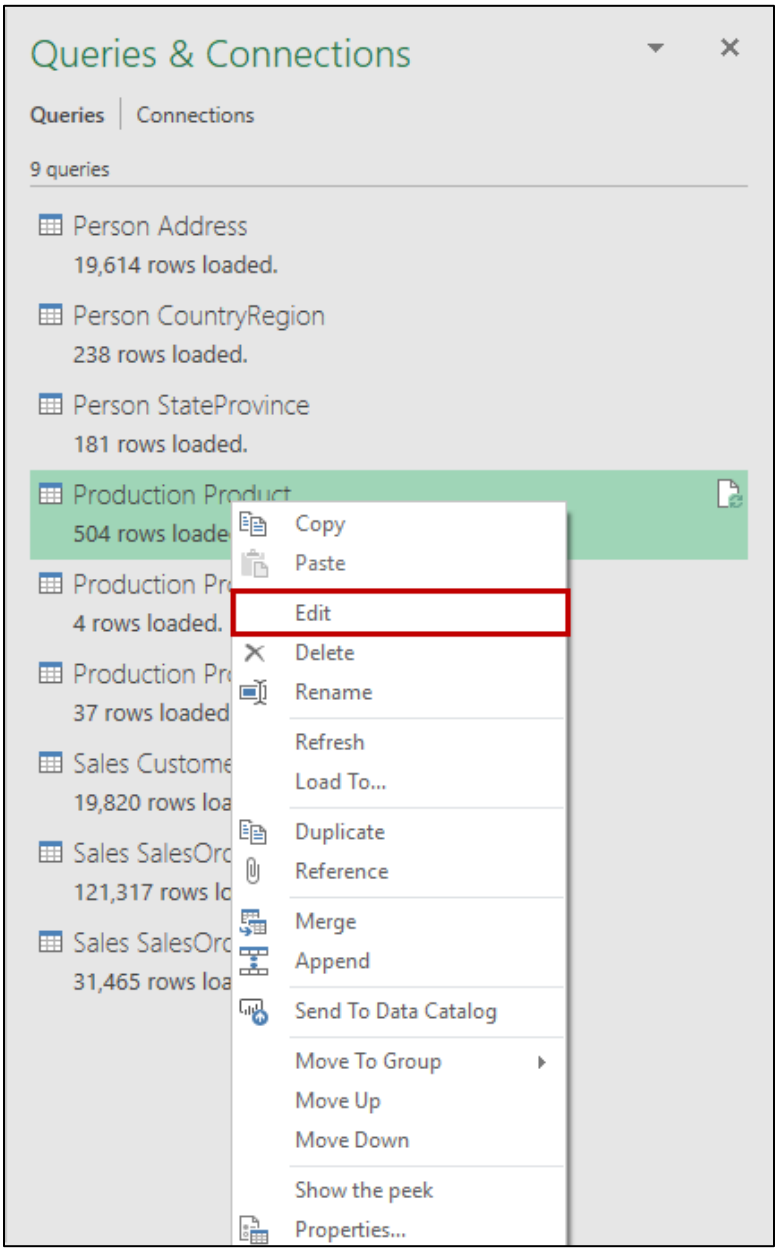


Figure 28 - Choosing to Edit in Power Query

Suppose you choose to load the data into the workbook's Data Model. In that case, you can access and manage the Data Model by clicking **Manage Data Model** on the **Data** tab of the Ribbon, as shown in **Figure 29**. Alternatively, you could use Excel's **Power Pivot** add-in to access and manage the Data Model.

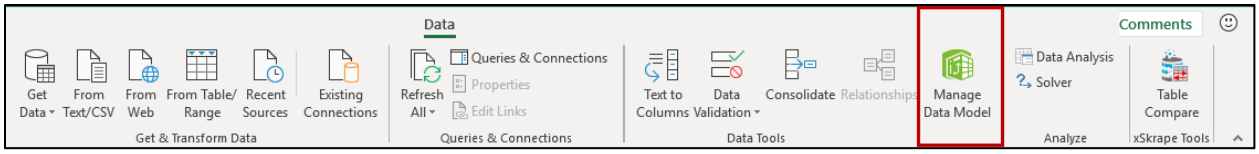


Figure 29 - Accessing the Data Model from the Data Tab of Excel's Ribbon

Additionally, you can summarize the data in the Data Model using Power Pivot or create visualizations of the data with traditional Excel techniques. You can also refresh your queries by choosing the **Refresh** option in the task pane pictured in Figure . Doing so ensures that you always have the most up-to-date data available.

Summary And Wrap Up

Excel's fundamental tools and features are no doubt a staple of many modern business environments. But, a large number of Excel's features go unnoticed by many users, particularly those who use subscription-based versions of Excel. In this session, you have learned about many of these features and how you can put them to work to help you and your team work more efficiently and effectively.