Knack Training

Modern Excel Functions

16 Powerful & Productive Tools for looking up, reshaping, and combining data

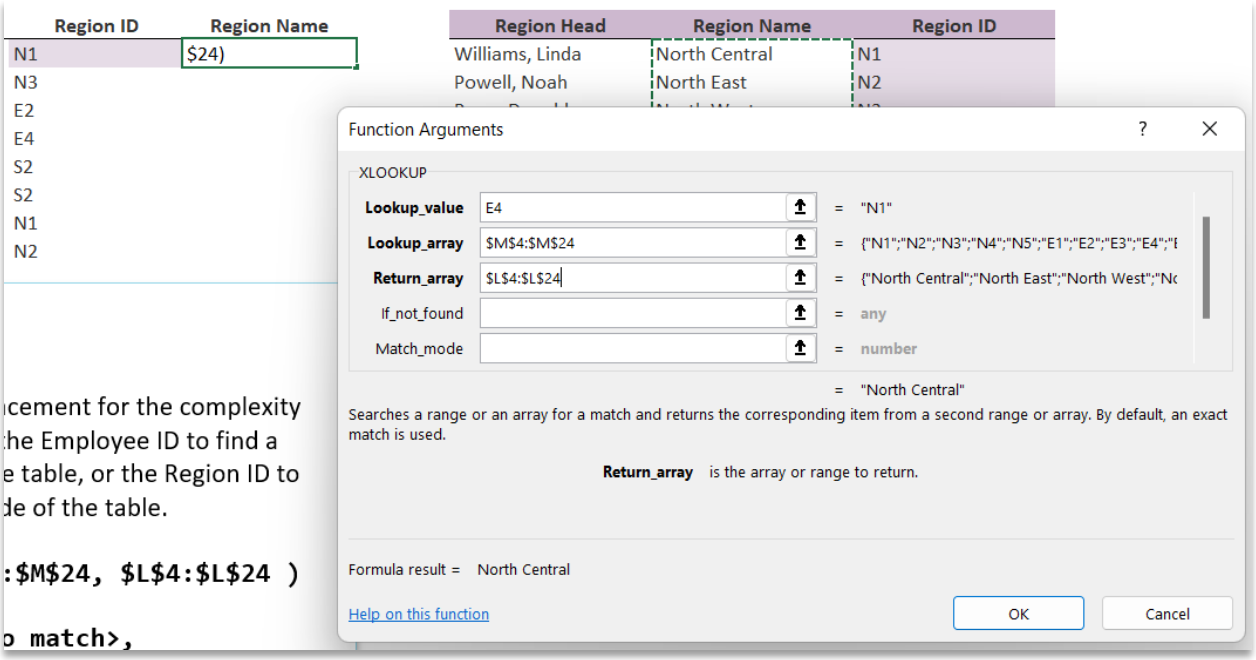
Contents

XLOOKUP

XLOOKUP is the modern function that was created to replace VLOOKUP and HLOOKUP (and many of the uses of other lookup tools like INDEX and MATCH). XLOOKUP is preferrable to these functions because:

1. XLOOKUP can perform a vertical lookup from the top down or bottom up;
2. XLOOKUP can perform a vertical lookup to find related data to the left or right of the lookup column;
3. XLOOKUP can also perform the same lookups horizontally as it can vertically;
4. XLOOKUP has a built-in tool for handling records that can't be matched; and
5. XLOOKUP can return an array of values, rather than a single value

All of these put it above VLOOKUP in functionality.



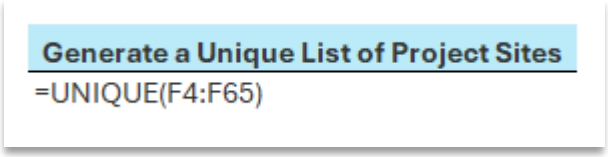
In this example, we see that we have a **lookup value**, that will be matched with a **lookup array**. Then, you can select anything you like for the **return array** for the values to return once the lookup is complete. Note the **if not found** value that can be filled in when there are no matches.

Dynamic Array Functions

UNIQUE Function

The UNIQUE function is easy to define; when you use it against an array of cells, it returns only the *unique* results.

ID	PROJECT SITE	V
	Luanda	
	Cascadia	
	Grey Coast	
	San Clemente	
	Santiago	
	Hyderabad	
	Santiago	
	Cascadia	
	San Clemente	
	San Clemente	
	Luanda	
	Cascadia	



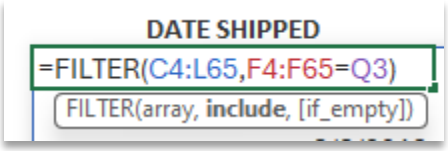
Generate a Unique List of Project Sites
=UNIQUE(F4:F65)

You can see here that we've simply asked the function for the unique values in the Project Site column, and we return:

Generate a Unique List of Project Sites
Luanda
Cascadia
Grey Coast
San Clemente
Santiago
Hyderabad
Kano
Snake River
Birmingham
Surat

FILTER Function

The FILTER function allows you to apply multiple filters against a dataset, and return the array of all matching values. This can then be spilled onto a worksheet, or used in another function.



In this function, we are expecting to return values from the range C4:L65, but only when the F column contains a specific value. Here, we see:

DATE SHIPPED	DATE ARRIVED	SHIPMENT ID	PROJECT SITE	VOLUME (m^3)	WEIGHT (kg)
4/23/2019	4/28/2019	SANN-2038	Santiago	29	316.1
8/31/2019	9/3/2019	SANN-1659	Santiago	35.4	1001.8
3/3/2019	3/8/2019	SANN-2069	Santiago	37.8	536.8
3/30/2019	4/5/2019	SANN-1642	Santiago	56.7	1190.7
8/25/2019	8/27/2019	SANN-2105	Santiago	8.9	162
11/10/2019	11/13/2019	SANN-1483	Santiago	16.8	247

All the shipments to **Santiago** show up in the resulting table, connected with our filter criteria.

Data Shaping Functions

VSTACK and HSTACK

The VSTACK and HSTACK functions are for creating *vertical* or *horizontal* stacks of related data lists. Vertical stacks are more obvious when dealing with vertical data, but HSTACK can be useful if you'd like to create side-by-side datasets.

In this example, we have three datasets on different tabs – but they all have the same columns:

	A	B	C	D
1	**Employee ID**	**First Name**	**Last Name**	**Event**
2	EM-1199	Mizuki	Tamura	Online L2 Oct 2023
3	EM-7211	Isaac	Lewis	Dallas L2 Sept 2023
4	EM-7690	Maria Teresa	Arias	Dallas L2 Sept 2023
5	EM-1525	Danny	Ford	Online L2 Oct 2023
6	EM-2217	Idris	Oni	Online L2 Oct 2023
7	EM-7765	June	Ochs	Atlanta L2 Oct 2023
8	EM-2406	Jason	Cooper	Online L2 Oct 2023
9	EM-1406	Nerea	Campos	Dallas L2 Sept 2023
10	EM-6003	Leilani	Park	Dallas L2 Sept 2023
11	EM-6686	Alana	Lopez	Atlanta L2 Oct 2023

They don't all have to be in columns A:D, but the order of these columns is important. The VSTACK function simply references all the datasets we want to compile:

```
=VSTACK('DT1'!A2:D47,'DT2'!A2:D46,'DT3'!A2:D35)
```

Our resulting dataset is everything on the DT1, DT2, and DT3 sheets, in columns A:D.

TOCOL and TOROW

The TOCOL function takes any set of cells and creates a simple, vertical column from the data, while TOROW takes any array of cells and creates a single, horizontal row from it.

Employee 1	Employee 2	Employee 3	Employee 4
Mejia, Mario	Fathy, Jomana	Nichols, Sue Ellen	Hill, Daniel
Chahine, Yasir	James, Ava	Vega, Fabricio	Cocco, Dorothy
Marchal, Marine	Silva, Everly	Levine, Ron	Mcintyre, David
Crow, Lawrence	Bowman, Christine	Graham, Martin	Yang, Tao
White, Chloe	Gurmani, Kheezran	Xie, Lan	Woods, Rebecca
Campbell, Everett	Seidl, Alexandra	Bello, Margaret	
Kennedy, Bertha	O'Dea, Irma	Campos, Manuel	
Khushk, Ayan	Sabry, Jomana		
Popovic, Elizabeta	Park, Seo-yeon		
Jerkovic, Liia	Robbins, Enrique		

In this example, we see that there are four columns containing the same type of data. However, columns 3 and 4 aren't completely filled, like columns 1 and 2 are.

```
=TOCOL(C6:F15,1)
```

In this example, we use **TOCOL** on columns C:F, and the **1** we see at the end of the function represents that we should *ignore blanks*. The result:

Employees
Mejia, Mario
Fathy, Jomana
Nichols, Sue Ellen
Hill, Daniel
Chahine, Yasir
James, Ava
Vega, Fabricio
Cocco, Dorothy
Marchal, Marine
Silva, Everly
Levine, Ron
Mcintyre, David
Crow, Lawrence
Bowman, Christine

WRAPCOLS and WRAPROWS

These functions take a set of values (either horizontal or vertical), and create multiple rows or columns out of them. Often, you're looking to create rows of data in a data table, so the WRAPROWS function is more popular and widely used.

Employee ID
First Name
Last Name
Date of Hire
Salary
EM-9868
Dora
Bennett
1/30/2011
$ 131,900.00
EM-7187
John
Lawal
8/6/2019
$ 160,100.00
EM-5282
Emily
Rodgers

Employee ID	First Name	Last Name	Date of Hire	Salary
EM-9868	Dora	Bennett	1/30/2011	$ 131,900.00
EM-7187	John	Lawal	8/6/2019	$ 160,100.00
EM-5282	Emily	Rodgers	10/4/2015	$ 175,500.00
EM-4769	Janice	Park	10/20/2007	$ 137,300.00
EM-8622	Ye-jun	Kan	5/30/2005	$ 169,400.00
EM-4846	Abir	Hosen	2/15/1997	$ 44,500.00
EM-8085	Daniel	Morgan	4/6/2002	$ 167,400.00
EM-7822	Jeremy	Delaney	6/7/2013	$ 77,000.00
EM-6861	Wataru	Kimura	9/3/2016	$ 75,500.00
EM-6073	Guiren	Liu	6/1/2005	$ 58,900.00
EM-4213	Ji	Yang	11/26/1997	$ 120,300.00
EM-7777	Jacqueline	Diaz	11/10/2006	$ 188,000.00
EM-6227	Luke	Marusic	7/23/2020	$ 60,100.00
EM-7057	Sienna	Price	1/12/2003	$ 35,500.00
EM-8093	Qiang	Teoh	2/4/2002	$ 166,200.00
EM-9445	Talora	Hahn	5/7/2020	$ 173,000.00
EM-6636	Jamie	Shaw	5/1/1998	$ 132,800.00

In this example, you can see that we are trying to take *first name, last name, date of hire, and salary*, and push those into their own columns, then **wraprows** down to the next row for the next record:

```
=WRAPROWS(C5:C94,5)
```

In this function, we see that we want to create a new row every time we reach **5 column entries** (in other words, Employee ID, First Name, Last Name, Date of Hire, and Salary).

TAKE and DROP

The TAKE function allows you to select either the top or bottom rows of a data set to *take*, or include in your data. The DROP function allows you to select top or bottom rows to *drop*, or exclude from your data. Since these functions work on the top or bottom rows of a data set, they are often used in conjunction with SORT functions.

If we FILTER and/or SORT the data to make it more likely that the top 10 or 20 are important, we can use TAKE to create that list in seconds.

Customer ID	Customer Name	Account Value	Date Initiated
CL-1006	Carolinas Holdings	$ 8,800.00	10/4/2014
CL-1007	Wall Team	$ 840,000.00	1/11/2011
CL-1019	Matthews & Arold Services	$ 34,000.00	12/3/2013
CL-1022	Ariadne Equity	$ 2,800.00	8/9/2018
CL-1032	Grimes & Jones Electrical	$ 330,000.00	11/2/2009
CL-1043	KR & C Guitars	$ 1,400.00	12/3/2012
CL-1043	Wahine Acoustics	$ 87,000.00	7/29/2014
CL-1044	Winter & Tucker General Partners	$ 920,000.00	3/24/2010
CL-1047	West Coast Investments	$ 460,000.00	3/26/2017

In this example, we will want to SORT by the Account Value column to make the largest values go to the top, then **TAKE** the top 10 rows.

```
=TAKE(SORT(C6:F1054,3,-1),10)
```

In this example, we see that the data is sorted by the **third column** (the Account Value) because of the 3 column index, and that it is sorted in **descending order** (the negative 1). Then, we simply **TAKE** the top 10 records with the 10 entry at the end of the function.

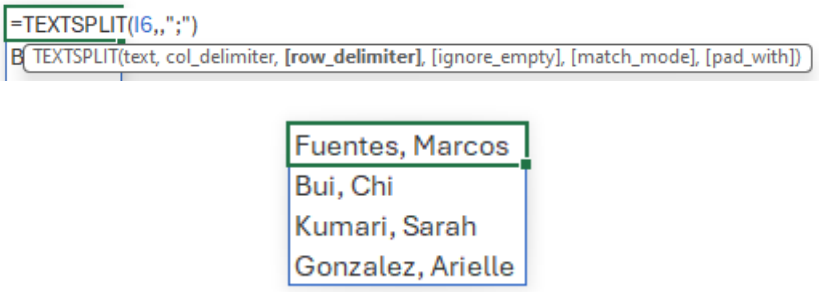
Text Manipulation Functions

TEXTSPLIT Function

The TEXTSPLIT function is used in scenarios where there is a single type of delimiter used in an entry, and you'd like to create new entries for each of them.

Roster
Fuentes, Marcos;Bui, Chi;Kumari, Sarah;Gonzalez, Arielle
Munoz, Marcos;Bukhari, Ayan;Masson, Céline;Munoz, Marcos
Fuentes, Marcos;Guerin, Kevin;Guy, Jason;Ngo, Yen
Jilani, Claudio;Ku, Martin;Robinson, Glenda;Clarke, Benjamin
Munoz, Marcos;Schroeder, Aria;Sosa, Terrance;Gill, Avi
Wilson, Thomas;Jilani, Claudio;Pastor, Nuria;Wynn, Kathleen
Muller, Andreas;Perez, Nicole;Maier, Okan;Daniels, Everett
Maes, Emeline;Ku, Martin;Millet, Margaux;Muller, Andreas

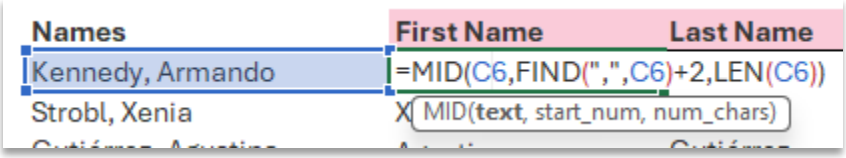
In the example here, we can see that there are multiple people's names separated by semicolons. Using =**TEXTSPLIT(I6,,";")** splits the values within cell I6 on the location of the semicolon. Specifically, this is the position for creating new **rows** from the data. The result looks like this:



We've broken that single cell into multiple *rows*. However, because TEXTSPLIT can accept an *array* of cells rather than a single cell, we can also split all the text in the entire range:

TEXTBEFORE and TEXTAFTER

These functions identify a single delimiter, then allow you to return either the text *before* that delimiter or *after* it. In a conventional (pre-2021) Excel workbook, we could use a complex function including LEFT, FIND, and other functions to determine the location of the punctuation marks or spaces in text, and determine how much of the text to remove:



In this example, we used the **MID** function to start somewhere in the middle of the text; the **FIND** function to identify the location of the comma, and the **LEN** function to haphazardly select all the text to the end of the entry. This successfully identifies the name ***Armando***.

However, the function below also does the same thing:

Custom Function Builders

LET for Custom Variables in Functions

In everyday operations, we often become mired down with overly complex functions. This can be especially bad when there are multiple functions in a single cell, or even multiple uses of the same function for some purpose. The LET function was designed to simplify the building of complex functions.

The structure of the LET function is the following:

=LET(variable_name, variable_value, …, calculation)

So, for example, let's imagine a function where we recalculate a multiplication multiple times:

```
=IF(E6*F6<500000,E6*F6,IF(E6*F6<1000000,E6*F6*0.9,E6*F6*0.8))
```

In this calculation, we are calculating E6*F6 to:

1. Create the basic value for a sale
2. Compare the value against 500,000
3. Compare the value against 1,000,000
4. Multiply the value times .9 if the value is between 500k and 1M
5. Multiply the value times .8 if the value is above 1M

This is clearly something that is used repetitively across the function. This introduces the following problems:

1. It's easy to make a mistake in one of those calculations
2. It's nearly impossible to read this function
3. Excel is actually *recalculating* the value as many times as we've typed it

So, instead we replace it with:

```
=LET(
StandardCost, E6*F6,
IF(
StandardCost < 500000, StandardCost,
IF(
StandardCost < 1000000, StandardCost * 0.9,
StandardCost * 0.8
)))
```

In this example, we start by saying ***let E6*F6 equal StandardCost***. This is the only time Excel calculates that multiplication, and it reduces our opportunities to make a mistake to one. Then, we see that we are judging ***is StandardCost < 500,000***? And ***is StandardCost < 1,000,000***?

(also notice that we've used the shortcut **[ALT] + [ENTER]** in building our calculation to make it a bit easier to read.

LAMBDA to Define Custom Functions in the Workbook

The LAMBDA function performs a similar task to VBA custom functions that have been available in previous versions of Excel. To see whether you need the LAMBDA function, ask yourself the question: ***are there repeated uses of the same complex function in my workbook?***

If you've got a scenario that sounds like that, you might actually define a custom function name in Excel, and use that to simplify the work in that file. Start with the calculation that works, and test it. In my example, that is the following function:

=DATEDIF(<Date of Hire>, <Today>, "Y")

That function is a little-known calculation that takes two dates and determines how much time is between them. Specifically, we've inserted the value "Y" to request the number of full years between the dates, but there are many other possible entries. This function has the following problems:

1. Not many people know it exists
2. Fewer people know how to use it
3. When people look at it in a workbook, they become confused
4. It's easy to make a mistake

So, instead, we create the following entry:

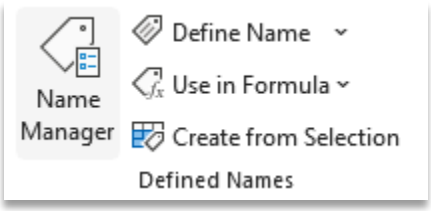
=LAMBDA(StartDate, DATEDIF(StartDate, TODAY(), "Y"))

The structure of what you see above is:

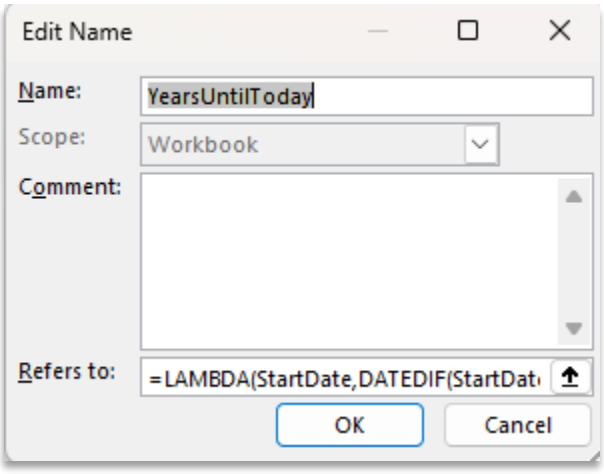
=LAMBDA(argument, ..., calculation)

... where we're defining the argument of StartDate as the value we want to accept from the user. The DATEDIF function, with its nested TODAY function and preset "Y" value, are all hard-coded into the function.

Finally, we need to create a Named Range that uses this structure. First, **copy** the code you've written. Then, click **Formulas tab > Name Manager**:



Create a name, and refer it, not to a cell, but the function you now **Paste** into the bottom field. Named Ranges are not allowed to have spaces, so I've named this **YearsUntilToday**:



Now, as you work in your file, you'll see that you can create a function from this definition: