

Excel Functions and Formulas I

Numeric Operators

A symbol that instructs Excel to perform a specific arithmetic calculation on a set of values.

Spreadsheets typically contain lots of quantitative data—data that has a numeric quality. Spreadsheets are useful when you're working with quantitative data because you can perform calculations to get even more information. For example, if you have a large retail dataset with all sales transactions over the last two years, you can use addition to find out what the total sales were. Or, if you have a large ridesharing dataset with all Uber rides completed over the last two years, you can use subtraction to find out the difference in total rides completed between this year and last year.

The four numeric operators available in Excel formulas follow the four basic mathematical operations: addition, subtraction, multiplication, and division. You can use a numeric operator to carry out each of these operations; the Excel numeric operators are +, -, *, and /.

Locking cells

As you now know, formulas can include cell references to point to values in other cells. When you drag a formula down a column, you're telling Excel to apply the same formula to each of the next cells.

In the screenshot below, the formula =C2/B2 is first created in cell D2. Then it's dragged down to cells D3, D4, D5, and D6. But it's important to note that it isn't =C2/B2 that is being applied to each of the other cells. Instead, it's =C3/B3 for cell D3, =C4/B4 for cell D4, =C5/B5 for cell D5, and =C6/B6 for cell D6. Excel is smart; it knows that you want to apply the same formula structure to the rest of the cells, but not the exact formula =C2/B2. As you drag a formula down, the cell references are automatically updated based on the new row and column address.

But what if you wanted to drag down the exact formula =C2/B2 for the whole column? How do you stop Excel from automatically updating the cell references as it gets to each new cell address? This is where locking cell references comes in. To lock a cell reference means to keep it from automatically updating when the formula is dragged down or across. To lock a cell reference, add a dollar sign \$ in front of both the letter and the number of the cell reference. To lock =C2/B2, for example, use =\$C\$2/\$B\$2. If you drag this formula down, the formula will apply to each cell. Take a look:

You can also lock a column reference by inserting \$ in front of only the letter, or lock a row reference by inserting \$ in front of only the number. If you lock only a column reference or only a row reference, then the unlocked references will still update as you drag formulas.

Excel Numeric Functions

A function is a command that is already defined and built into Excel. And a numeric function is a function that can be applied to numeric (quantitative) values. Functions are made up of letters instead of symbols. These letters spell out the command, or an abbreviated version of the command.

To use any function, start with the equals sign =, followed by the function name, followed by parentheses (). Inside the parentheses, add the arguments, which are the values that the function will use to perform the calculation.

The five basic numeric functions: SUM, AVERAGE, PRODUCT, MIN, MAX.

String Functions

A string is a data type that is made up of text. Examples of strings include customer names, addresses, or names of items. Strings are typically made up of letters, but they sometimes include numbers or symbols. Some string functions include: CONCAT, "&", LEFT, RIGHT, and MID, UPPER, LOWER, and PROPER, and there is TRIM, and LEN.

CONCAT won't do any additional cleaning or manipulating; it only combines two or more strings together. So, if you are combining the two words United and States together using CONCAT, it will result in UnitedStates. Be mindful of this when you are combining separate words. You may need to manually add a space afterward.

The & (ampersand) operator is the symbol version of the CONCAT function. It is used to join strings together. You can use & in the same situations where you would use CONCAT. This operator has the structure =text1&text2. In the example below, the & operator was used to combine the name of the country Brazil.

You may encounter a data field that has characters or symbols that you want to extract while deleting the rest of the field. The LEFT function returns a specified number of characters from the beginning of the string. And the RIGHT function returns a specified number of characters from the end of the string. The structure of these functions is =LEFT(argument, number of characters) and =RIGHT(argument, number of characters). The MID function returns the characters in the middle of a string, using a specified starting position and length. The MID function structure is =MID(text, starting number, number of characters).

There are three functions that make it easy to fix letter casing without having to retype the strings. The PROPER function will change the string to have proper noun casing, meaning that the first letter will be capitalized while the rest will be lowercase. The UPPER function will change all letters to uppercase, while the LOWER function will change all letters to lowercase.

The TRIM function removes all the spaces in a string, except for single spaces between words. This is especially useful for removing the leading spaces that sometimes appear in data that has been converted from a different database.

The LEN function returns the length of characters of the string. This can be useful if you need to know the length of a string to create a formula that manipulates the string properly. Or perhaps you want to use the LEFT, RIGHT, or MID functions but need to first check how long the original string is.

Comparisons and Logical Functions

The > operator is the greater-than operator. This operator is used to determine if a value is greater than another value. The formula structure for using this operator is =value1>value2. This returns TRUE if value1 is greater than value2. Otherwise, it returns FALSE.

The < operator is the less-than operator. This operator is used to determine if a value is less than another value. The formula structure for using this operator is =value1<value2. This returns TRUE if value1 is less than value2. Otherwise, it returns FALSE.

The = operator is the equals operator. This is used to determine if two values are equal. The formula structure for using this operator is =value1=value2. This returns TRUE if value1 is equal to value2. Otherwise, it returns FALSE.

The >= operator is the greater-than-or-equal-to operator. This operator is used to determine if a value is greater than or equal to another value. The formula structure for using this operator is =value1>=value2. This returns TRUE if value1 is greater than or equal to value2. Otherwise, it returns FALSE.

The <= operator is the less-than-or-equal-to operator. This operator is used to determine if a value is less than or equal to another value. The formula structure for using this operator is =value1<=value2. This returns TRUE if value1 is less than or equal to value2. Otherwise, it returns FALSE.

The <> operator is the not-equal-to operator. This operator checks if two values are not equal. It's the opposite of the = operator. The formula structure for using this operator is =value1<>value2. This returns TRUE if value1 isn't equal to value2. Otherwise, it returns FALSE.

Transitioning from operators to functions, the IF function is used to create a logical test. It checks if a condition is met and then returns one value if the condition is met (or is TRUE) and another value if the condition isn't met (or is FALSE). When creating the formula, specify what the condition is and which values are returned if the condition is met or not. The structure of a formula using the IF function is =IF(logical test, value if TRUE, value if FALSE). As with most formulas, the arguments can use cell references.

The SUMIF function is another logical test like the IF function. But the SUMIF function doesn't return a specified value if a condition is met. Instead, it sums up a group of values if the condition is met. The formula structure using the SUMIF function is =SUMIF(range of cells to check criteria, criteria, range of cells to sum).

The COUNTIF function is similar to the SUMIF function, except that it counts the number of populated cells if a condition is met. The formula structure using the COUNTIF function is =COUNTIF(range of cells to check criteria, criteria).

To conclude the Excel Functions and Formula I section of the Certificate, we used the hotel bookings dataset. Taking the data, we first used an IF statement to determine if a hotel was considered "Cheap", or "Moderate". Then, we used a similar IF statement to classify the booking data as a "Short Stay" or a "Long Stay". And Finally, a COUNTIF statement to determine the number of hotel bookings that are considered "Cheap".

This project utilizes Comparison Operators, Numerical and String Functions, and Logical Functions.

VLOOKUP

To get a better sense of how you might use VLOOKUP, consider the following example: Imagine that you are a manager at a company. The spreadsheet below contains your employee names, along with the hours they worked in each of the first two weeks. You keep track of their hours worked to monitor that they are accurate and to check if anyone has worked overtime.

	A	B	C	D
1	Employee	Week 1 Hours Worked	Week 2 Hours Worked	
2	Sarah	40	42	
3	Tim	35	40	
4	Lucy	34	40	
5	Ellen	38	38	
6	Roberto	37	38	
7	Julie	40	39	
8	Jose	42	38	
9	Maya	45	41	
10	Lily	41	40	
11	Katya	39	40	
12	Alexa	40	38	
13	Sai	40	35	
14	Kelly	40	40	
15	Monica	42	42	
16	Arun	37	39	
17	Francesca	40	38	

The first argument is the lookup value, which is "Sai" in this case.

	A	B	C	D	E	F	G	H	I	J
1	Employee	Week 1 Hours Worked	Week 2 Hours Worked							
2	Sarah	40	42							
3	Tim	35	40							
4	Lucy	34	40		=VLOOKUP("Sai"					
5	Ellen	38	38		VLOOKUP(lookup_value, table_array, col_index_num, [range_lookup])					
6	Roberto	37	38							
7	Julie	40	39							
8	Jose	42	38							
9	Maya	45	41							
10	Lily	41	40							
11	Katya	39	40							
12	Alexa	40	38							
13	Sai	40	35							
14	Kelly	40	40							
15	Monica	42	42							
16	Arun	37	39							
17	Francesca	40	38							

The second argument is the table array, which is the table or range of cells that VLOOKUP searches through. In this case, it's table A2:C17. Ignore table headers when specifying the table array for VLOOKUP.

SUM				=VLOOKUP("Sai", A2:C17							
	A	B	C	D	E	F	G	H	I	J	K
1	Employee	Week 1 Hours Worked	Week 2 Hours Worked								
2	Sarah	40	42								
3	Tim	35	40								
4	Lucy	34	40								
5	Ellen	38	38								
6	Roberto	37	38								
7	Julie	40	39								
8	Jose	42	38								
9	Maya	45	41								
10	Lily	41	40								
11	Katya	39	40								
12	Alexa	40	38								
13	Sai	40	35								
14	Kelly	40	40								
15	Monica	42	42								
16	Arun	37	39								
17	Francesca	40	38								

The third argument is the column index number. Week 1 Hours Worked is what you're looking for, and it's in the second column of the table array. So, the column index number is 2. (If you wanted to look up Week 2 Hours Worked instead, then the column index number would be 3.)

SUM				=VLOOKUP("Sai", A2:C17,2							
	A	B	C	D	E	F	G	H	I	J	K
1	Employee	Week 1 Hours Worked	Week 2 Hours Worked								
2	Sarah	40	42								
3	Tim	35	40								
4	Lucy	34	40								
5	Ellen	38	38								
6	Roberto	37	38								
7	Julie	40	39								
8	Jose	42	38								
9	Maya	45	41								
10	Lily	41	40								
11	Katya	39	40								
12	Alexa	40	38								
13	Sai	40	35								
14	Kelly	40	40								
15	Monica	42	42								
16	Arun	37	39								
17	Francesca	40	38								

The fourth and final argument is the range lookup value. This is TRUE if you want an approximate match, or FALSE if you want an exact match. Most people want an exact match, so they use FALSE.

SUM				=VLOOKUP("Sai", A2:C17,2,FALSE							
	A	B	C	D	E	F	G	H	I	J	K
1	Employee	Week 1 Hours Worked	Week 2 Hours Worked								
2	Sarah	40	42								
3	Tim	35	40								
4	Lucy	34	40								
5	Ellen	38	38								
6	Roberto	37	38								
7	Julie	40	39								
8	Jose	42	38								
9	Maya	45	41								
10	Lily	41	40								
11	Katya	39	40								
12	Alexa	40	38								
13	Sai	40	35								
14	Kelly	40	40								
15	Monica	42	42								
16	Arun	37	39								
17	Francesca	40	38								

The result of this formula is 40. Sai worked 40 hours in Week 1.

E4					=VLOOKUP("Sai", A2:C17,2,FALSE)
	A	B	C	D	E
1	Employee	Week 1 Hours Worked	Week 2 Hours Worked		
2	Sarah	40	42		
3	Tim	35	40		
4	Lucy	34	40		40
5	Ellen	38	38		
6	Roberto	37	38		
7	Julie	40	39		
8	Jose	42	38		
9	Maya	45	41		
10	Lily	41	40		
11	Katya	39	40		
12	Alexa	40	38		
13	Sai	40	35		
14	Kelly	40	40		
15	Monica	42	42		
16	Arun	37	39		
17	Francesca	40	38		

VLOOKUP with cell references

If you want to check hours for multiple employees but don't want to rewrite the formula over and over, you can use cell references. Below, the same formula is used, but notice that the lookup value is E3. Also, notice that cell E3 has Katya entered. As you update cell E3 with other employee names, the VLOOKUP formula will automatically recalculate.

E4							=VLOOKUP(E3, A2:C17,2,FALSE)
	A	B	C	D	E	F	G
1	Employee	Week 1 Hours Worked	Week 2 Hours Worked				
2	Sarah	40	42				
3	Tim	35	40		Katya	<-- lookup value	
4	Lucy	34	40		39	<-- formula result	
5	Ellen	38	38				
6	Roberto	37	38				
7	Julie	40	39				
8	Jose	42	38				
9	Maya	45	41				
10	Lily	41	40				
11	Katya	39	40				
12	Alexa	40	38				
13	Sai	40	35				
14	Kelly	40	40				
15	Monica	42	42				
16	Arun	37	39				
17	Francesca	40	38				

VLOOKUP in other sheets

VLOOKUP becomes really powerful when you use it with multiple sheets. In fact, this function is typically used to pull in data from other sheets or workbooks. This is an important feature because data often comes in more than one file or sheet. Sometimes, data comes from two completely different sources, and you'll want to combine them to make the data easier to view.

The example below uses the same data as before. But now, the formula is in the current sheet, and the data is housed in another sheet called Sheet2. The formula structure is the same as before, but the table array now includes a sheet name and an exclamation point !. This is the notation to use when pointing to a different sheet.

B3		✕ ✓ <i>f_x</i>		=VLOOKUP(B2,Sheet2!A2:C17,2,FALSE)				
	A	B	C	D	E	F	G	H
1								
2		Katya	<-- lookup value					
3		39	<-- formula result					
4								

The same idea applies if you are pulling data that is housed in a different workbook or file. The table array below shows the workbook name in brackets [], followed by the sheet name and an exclamation point.

B3		✕ ✓ <i>f_x</i>		=VLOOKUP(B2,[Book1]Sheet2!A2:C17,2,FALSE)				
	A	B	C	D	E	F	G	H
1								
2		Katya	<-- lookup value					
3		39	<-- formula result					
4								

VLOOKUP to retrieve a column of data

When working with data, you may need to pull in an entire column of data from a different sheet into your current sheet. Imagine that the same employee data is split out over two sheets: Sheet1 contains employee names and hours worked in Week 1, and Sheet2 contains employee names and hours worked in Week 2.

If you want to compare the two weeks, you'd need to flip back and forth between the two sheets to look at the Week 1 and Week 2 values. Or, instead, you could use a VLOOKUP function to pull the Week 2 values into the same sheet as the Week 1 values. Then all your data would be together in one place.

To do this, create a VLOOKUP formula in cell C2 of Sheet1. This formula is for the corresponding employee in A2.

SUM						
	A	B	C	D	E	F
1	Employee	Week 1 Hours Worked				
2	Sarah	40	=VLOOKUP(A2, Sheet2!A2:B17,2,FALSE)			
3	Tim	35				
4	Lucy	34				
5	Ellen	38				
6	Roberto	37				
7	Julie	40				
8	Jose	42				
9	Maya	45				
10	Lily	41				
11	Katya	39				
12	Alexa	40				
13	Sai	40				
14	Kelly	40				
15	Monica	42				
16	Arun	37				
17	Francesca	40				

Now that your formula has been created, you just need to drag the formula down for all employees. But before dragging down any VLOOKUP formula, remember to lock the cells of the table array in the top formula! If you don't, the table array will keep shifting as you drag the formula down. That means that the data returned may be incorrect.

C2						
	A	B	C	D	E	F
1	Employee	Week 1 Hours Worked				
2	Sarah	40	42			
3	Tim	35	40			
4	Lucy	34	40			
5	Ellen	38	38			
6	Roberto	37	38			
7	Julie	40	39			
8	Jose	42	38			
9	Maya	45	41			
10	Lily	41	40			
11	Katya	39	40			
12	Alexa	40	38			
13	Sai	40	35			
14	Kelly	40	40			
15	Monica	42	42			
16	Arun	37	39			
17	Francesca	40	38			

The HLOOKUP function

This lesson covers the HLOOKUP function. The HLOOKUP function is used to merge or pull data quickly.

In the previous lesson, you learned about the VLOOKUP function, which stands for vertical lookup. This lesson covers the HLOOKUP function, which stands for horizontal lookup. Like VLOOKUP, HLOOKUP finds a specified value and returns another value close by. The difference is that while VLOOKUP searches for the lookup value in the leftmost column of the table array, HLOOKUP searches for the lookup value in the topmost row of the array. Once it locates the lookup value, it then returns a value from the same column. The difference between these two functions is depicted below.

SUM								
	A	B	C	D	E	F	G	H
1								
2						Lookup Value		
3						=VLOOKUP(F2,A2:D10,2,FALSE)		
4								
5								
6								
7								
8								
9								
10								

SUM								
	A	B	C	D	E	F	G	H
1								
2						Lookup Value		
3						=HLOOKUP(F2,A2:D10,2,FALSE)		
4								
5								
6								
7								
8								
9								
10								

Imagine that you have a side business selling handmade clothing. You have a dataset with 15 rows, each representing a different one of your items. The dataset also has 30 columns, representing each item's sales metrics—such as cost of supplies, price, revenue, profit, number sold, and date sold. If you wanted to quickly look up the number sold for item 9, you could locate item 9 and then manually look through all 30 columns until you land on Number sold. But you could also use HLOOKUP to do this for you! The beauty of HLOOKUP is that you can simply tell it which column name to look for. That way, you don't have to scroll through them all yourself. HLOOKUP searches for Number sold and then moves down until it gets to the row for item 9 and returns the value.

HLOOKUP is extremely useful when you need to find the value of a field for a particular row but there are a lot of fields present. You may encounter data with 40 or more fields, which is a lot to look through for a certain field name. HLOOKUP makes this faster and less manual. Thinking back to the example above about your handmade clothing business, it's much easier to tell Excel which column to look for so that you don't have to scroll through them all yourself.

How HLOOKUP is structured

The HLOOKUP function is called a horizontal lookup because it looks for the initial lookup value horizontally in the top row of the table array. Once it locates it, it then moves down the same column a specified number of times to grab the desired data.

The structure of HLOOKUP is as follows:

=HLOOKUP(lookup_value, table_array, row_index_number, range_lookup)

Like VLOOKUP, this formula has four arguments:

1. The lookup value is the initial value that the formula looks for in the topmost row.
2. The table array is the table of data that it searches through.
3. The row index number is the number of rows that the formula should move down from the lookup value.
4. The range lookup is either TRUE if you want an approximate match or FALSE for an exact match.

As with VLOOKUP, people usually use FALSE as the range lookup value, because they want an exact match. TRUE is rarely used, and only if the lookup value that you need doesn't exist in the data for some reason.

HLOOKUP example

In the spreadsheet below, the annual inches of rain and annual inches of snow are listed for 19 different regions.

G17			
	A	B	C
1	Region	Annual Inches of Rain	Annual Inches of Snow
2	1	4	10
3	2	5	12
4	3	13	12
5	4	15	14
6	5	11	20
7	6	10	22
8	7	5	15
9	8	6	9
10	9	2	11
11	10	9	5
12	11	11	4
13	12	13	6
14	13	25	13
15	14	30	12
16	15	33	29
17	16	45	33
18	17	50	30
19	18	66	35
20	19	67	27

Imagine that you want to use HLOOKUP to find the annual inches of snow for region 14. In cell E4, begin your formula by typing =HLOOKUP{.

The first argument after the parenthesis is the lookup_value. Here, your lookup value will be "Annual Inches of Snow", because that is what you're trying to find for region 14.

Remember to use quotation marks if any of your formula arguments are text strings!

See this units project using the Minimum Wage dataset showcasing VLOOKUP where we use data from multiple sheets to compare the minimum wages across 2018, 2019, 2020.

SUM				=HLOOKUP("Annual Inches of Snow"						
	A	B	C	D	E	F	G	H	I	J
1	Region	Annual Inches of Rain	Annual Inches of Snow							
2	1		4	10						
3	2		5	12						
4	3		13	12	=HLOOKUP("Annual Inches of Snow"					
5	4		15	14	HLOOKUP(lookup_value, table_array, row_index_num, [range_lookup])					
6	5		11	20						
7	6		10	22						
8	7		5	15						
9	8		6	9						
10	9		2	11						
11	10		9	5						
12	11		11	4						
13	12		13	6						
14	13		25	13						
15	14		30	12						
16	15		33	29						
17	16		45	33						
18	17		50	30						
19	18		66	35						
20	19		67	27						

The table_array is the table of values from A1:C20.

				=HLOOKUP("Annual Inches of Snow", A1:C20						
	A	B	C	D	E	F	G	H	I	J
1	Region	Annual Inches of Rain	Annual Inches of Snow							
2	1		4	10						
3	2		5	12						
4	3		13	12	=HLOOKUP("Annual Inches of Snow", A1:C20					
5	4		15	14	HLOOKUP(lookup_value, table_array, row_index_num, [range_lookup])					
6	5		11	20						
7	6		10	22						
8	7		5	15						
9	8		6	9						
10	9		2	11						
11	10		9	5						
12	11		11	4						
13	12		13	6						
14	13		25	13						
15	14		30	12						
16	15		33	29						
17	16		45	33						
18	17		50	30						
19	18		66	35						
20	19		67	27						

The next argument is row_index_number. You specifically want to know the annual inches of snow for region 14. Because region 14 is in the fifteenth row of the table array, the row index number is 15.

SUM				=HLOOKUP("Annual Inches of Snow", A1:C20,15						
	A	B	C	D	E	F	G	H	I	J
1	Region	Annual Inches of Rain	Annual Inches of Snow							
2	1		4	10						
3	2		5	12						
4	3		13	12	=HLOOKUP("Annual Inches of Snow", A1:C20,15					
5	4		15	14	HLOOKUP(lookup_value, table_array, row_index_num, [range_lookup])					
6	5		11	20						
7	6		10	22						
8	7		5	15						
9	8		6	9						
10	9		2	11						
11	10		9	5						
12	11		11	4						
13	12		13	6						
14	13		25	13						
15	14		30	12						
16	15		33	29						
17	16		45	33						
18	17		50	30						
19	18		66	35						
20	19		67	27						

The range_lookup value is FALSE; this will ensure that the formula will return only an exact match.

Press Enter when you finish typing the formula. The result is 12—so now you know that region 14 has about 12 inches of snow per year.

HLOOKUP with cell references

Just like VLOOKUP formulas, HLOOKUP formulas can also harness the power of cell references. To check annual inches of snow and then also check annual inches of rain, set up a cell reference for the lookup value. In the example below, the lookup value is the cell reference E3, and cell E3 is currently populated with Annual Inches of Snow. As you update the lookup value in E3 with Annual Inches of Rain, the formula will automatically update and recalculate.

SUM				=HLOOKUP(E3, A1:C20,15, FALSE)		
	A	B	C	D	E	F
1	Region	Annual Inches of Rain	Annual Inches of Snow			
2	1	4	10			
3	2	5	12			
4	3	13	12		Annual Inches of Snow	<--Lookup Value
5	4	15	14		=HLOOKUP(E3, A1:C20,15, FALSE)	
6	5	11	20			
7	6	10	22			
8	7	5	15			
9	8	6	9			
10	9	2	11			
11	10	9	5			
12	11	11	4			
13	12	13	6			
14	13	25	13			
15	14	30	12			
16	15	33	29			
17	16	45	33			
18	17	50	30			
19	18	66	35			
20	19	67	27			

And like VLOOKUP formulas, you can also use HLOOKUP formulas to pull in data from other sheets or workbooks.

INDEX and MATCH Functions

The INDEX function

This lesson explores how to use Excel's INDEX function to retrieve a value at the intersection of a row and a column.

Imagine that you're in the market to buy a new car, and you're comparing a few different cars and their features. The data that you're looking at has five rows: Mazda, Honda, Lexus, Volvo, and Kia. It also has five columns: Price, Gas Mileage, Latest Model, Touchscreen, and Automatic Headlights. If you want to know what the latest model is for Lexus, you'd have to find the Lexus row and then scan over to the Latest Model column. To check if the Volvo has automatic headlights, you'd need to locate the Volvo row and then scan over to the Automatic Headlights row.

Imagine doing this every time that you want to look up one of these cars and its features. Now imagine if this data table contained hundreds of rows of cars and hundreds of columns of car features. You wouldn't be able to look things up manually in such a big data table. But you could use the INDEX function to look this info up! The INDEX function makes it easy to identify a value if you know the row number and column number.

How INDEX is structured

The INDEX function returns the value of a cell if you provide the row number and column number of that cell. INDEX formulas have the following structure:

```
=INDEX(array, row_number, column_number)
```

This formula has three arguments:

1. The array, or table array, is the table that contains the data.
2. The row number is the cell's row number in the table array.
3. The column number is the cell's column number in the table array.

The MATCH function

This lesson covers the MATCH function, which returns the relative position of a specific value.

Imagine that you have a long guest list with 150 names. The names aren't ordered alphabetically; instead, they are ordered by each person's RSVP date. You want to locate a few names, but you really don't want to go down the names one by one to find them. Good thing Excel has the MATCH function!

The MATCH function looks for a value in a list and then returns the value's relative position. If you use the MATCH function to locate a guest name, the MATCH function will return a number, like 43. This tells you that the name that you're searching for is forty-third in the list. This is much easier than scanning the list yourself, right?

Lookup array

The row, column, or other array that the function searches through.

How MATCH is structured

The MATCH function looks for a value that you tell it, in a list of data that you select. Once it finds the value, it returns the relative position of the value in the list. MATCH is structured as follows:

=MATCH(lookup_value, lookup_array, match_type)

This formula has three arguments:

1. The *lookup value* is whatever value you're looking for. This can be entered manually or as a cell reference.
2. The *lookup array* is the range of cells that you want it to look through. The lookup array is different from the table array that you've seen in past lessons. The lookup array is a one-dimensional array, like a list, while a table array is a two-dimensional array, like a table. So, when using the MATCH function, you'll use a single column or row as the lookup array.
3. The *match type* can be -1, 0, or 1. A match type of -1 finds the smallest value that is greater than or equal to the lookup value. A match type of 0 finds the exact value of the lookup value. A match type of 1 finds the largest value that is less than or equal to the lookup value. Most of the time, a match type of 0 is used because the exact value is desired.

A match type of -1 or 1 may be used to get an approximate match. This could come in handy if the exact value doesn't exist in the data, and a close match will be found instead. Say a column contains the values 8, 9, 10, 15, 16, and 17. If your desired lookup value is 14 but that value doesn't exist in the data, a match type of -1 would return the position of the value 15. A match type of 1 would return the position of the value 10. This may work in certain situations when you're looking for something close to a particular value. But you'll usually want an exact match, so you'll use a match type of 0 in most cases.

MATCH with cell references

Like all functions, the MATCH function can take cell references as the lookup value. If you need to search for a few names, you can designate a cell to take lookup values. Then you can have the formula point to the cell reference. This is demonstrated below.

C3						
	A	B	C	D	E	F
1	Guest	RSVP				
2	John	Y	Lucy	<-- Input lookup value		
3	Michael	N	8	<-- Match result		
4	Kelly	Y				
5	Kevin	N				
6	Aretha	N				
7	Neo	Y				
8	Laura	Y				
9	Lucy	Y				
10	Bill	N				
11	David	Y				
12	Joe	Y				
13	Morgan	Y				
14	Sarah	N				
15	Anya	N				
16	Kai	Y				
17	Maya	N				
18	Lena	Y				
19	Alex	Y				
20	Mo	N				

This lesson walked us through how the MATCH function works. But the MATCH function usually isn't used alone; it's typically used with the INDEX function, as you'll explore later in this module.

Nesting with IF, AND, and OR Functions

Nested function

A function that is used as an argument of another function.

This lesson demonstrates how to create nested functions with IF, AND, and OR. By using these nested functions, you'll be able to create powerful formulas and to add multiple conditions to logical tests.

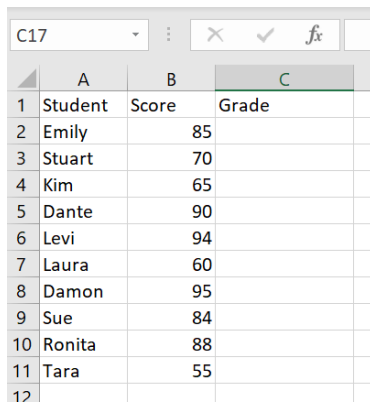
The beauty of Excel functions is that they can perform simple calculations and they can be combined with other functions to do more complex calculations. This lesson shows you how to use nested functions—which are functions that are used as arguments of other functions—with IF, AND, and OR.

Say that you're teaching a summer school class, and you're assigning grades to each student based on their final project scores. You don't want to use a simple IF statement that assigns an A to everyone above 70% and an F to everyone below 70%. You want an IF statement that assigns an A to everyone above 90%, a B to everyone between 80% and 89%, a C to everyone between 70% and 79%, a D to everyone between 60% and 69%, and an F to everyone else. If you use a nested function, you can do this with a single formula!

Nesting IF functions with IF Functions

A nested IF function is when one IF function is used as an argument of another IF function. This is useful whenever you need a logical statement with multiple conditions, each with its own resulting value if the condition is met.

To revisit the example at the top of this lesson, imagine that you're teaching a summer class for students. You have a spreadsheet with each student's name, along with their final project scores. You're trying to assign a letter grade to each student based on this score.



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C
1	Student	Score	Grade
2	Emily	85	
3	Stuart	70	
4	Kim	65	
5	Dante	90	
6	Levi	94	
7	Laura	60	
8	Damon	95	
9	Sue	84	
10	Ronita	88	
11	Tara	55	
12			

If a student gets a 90% or higher on their project, they receive an A. If the student gets 80%-89%, they receive a B. If they get 70%-79%, they receive a C. If they get 60%-69%, they receive a D. And if they get anything below 60%, they receive an F.

If you try doing this type of categorization with a single IF statement, you'll quickly see that a single IF statement allows you to make only two categories—one if the condition is met and one if the condition

isn't met. Nested IF functions let you create as many categories as you want, depending on how many functions you nest.

For this example, start with the first condition for the first student: if the student gets 90% or higher, they receive an A. You can write that in Excel as follows:

SUM				=IF(B2>90,"A")		
	A	B	C	D	E	F
1	Student	Score	Grade			
2	Emily	85	=IF(B2>90,"A")			
3	Stuart	70	IF(logical_test, [value_if_true], [value_if_false])			
4	Kim	65				
5	Dante	90				
6	Levi	94				
7	Laura	60				
8	Damon	95				
9	Sue	84				
10	Ronita	88				
11	Tara	55				
12						

Now, recall the formula structure for a regular IF function, which you learned about in the previous module:

=IF(logical_test, value_if_TRUE, value_if_FALSE)

Now, compare that with the formula structure for a nested IF function:

=IF(logical_test, value_if_TRUE, nested_IF_function)

The nesting happens at the value_if_FALSE argument. As you've learned, an IF function starts by evaluating the logical_test argument to see if it's TRUE or FALSE. If it's FALSE, it moves to the value_if_FALSE argument. But with a nested function, it sees that the value_if_FALSE argument is another IF function, and it starts evaluating this next IF function.

So, add the next function to define a score between 80% and 89% as a B. That should look like this:

SUM				=IF(B2>90,"A", IF(B2>80, "B"			
	A	B	C	D	E	F	G
1	Student	Score	Grade				
2	Emily	85	=IF(B2>90,"A", IF(B2>80, "B"				
3	Stuart	70	IF(logical_test, [value_if_true], [value_if_false])				
4	Kim	65					
5	Dante	90					
6	Levi	94					
7	Laura	60					
8	Damon	95					
9	Sue	84					
10	Ronita	88					
11	Tara	55					

You can see that the condition doesn't say anything about 89%. That is because the previous logical test already took care of the upper limit. The same process continues for grades C and D, as shown below.

	A	B	C	D	E	F	G	H	I	J
1	Student	Score	Grade							
2	Emily	85	=IF(B2>90,"A", IF(B2>80, "B", IF(B2>70, "C", IF(B2>60, "D"							
3	Stuart	70								
4	Kim	65								
5	Dante	90								
6	Levi	94								
7	Laura	60								
8	Damon	95								
9	Sue	84								
10	Ronita	88								
11	Tara	55								

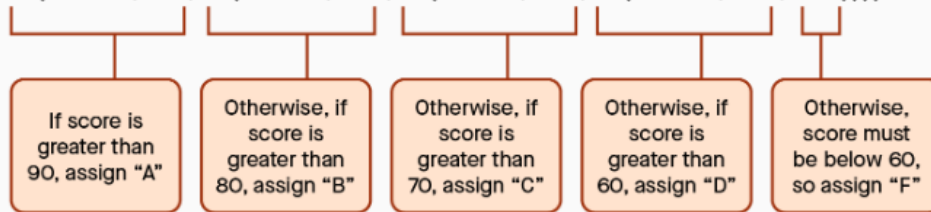
For the very last category in the nested function, you can simply write "F" as the `value_if_FALSE` argument. Then add as many closing parentheses `)` as there are opening parentheses `(`.

The formula evaluates Emily's score of 85% as a B.

	A	B	C	D	E	F	G	H	I
1	Student	Score	Grade						
2	Emily	85	B						
3	Stuart	70							
4	Kim	65							
5	Dante	90							
6	Levi	94							
7	Laura	60							
8	Damon	95							
9	Sue	84							
10	Ronita	88							
11	Tara	55							

The following diagram breaks down what the formula above is doing.

=IF(B2>90, "A", IF(B2>80, "B", IF(B2>70, "C", IF(B2>60, "D", "F"))))



Nesting IF functions with AND and OR Functions

You can also nest IF functions with AND and OR functions.

Say that you're making paper crafts to sell on your online store. You have one column of data containing the item names, one column with their sizes, and one column with the cost to make each item. You're trying to figure out how to price each craft based on this data. If you're limited to simple functions, you can use a conditional IF function that says, "if the cost of the item is less than \$10, price the item at \$15." Or you could use a function that says, "if the item size is large, price the item at \$20."

But what if you want to price each item based on its price and size? In situations like this, you can use the IF function with AND or OR. AND tells the formula that both conditions must be met, while OR tells the formula that at least one of the listed conditions must be met.

The spreadsheet below shows a list of the craft items along with their sizes and costs. The Price column needs to be filled in.

	A	B	C	D
1	Paper Craft	Size	Cost to make (\$)	Price
2	A	Large	9	
3	B	Medium	9	
4	C	Small	5	
5	D	Large	25	
6	E	Medium	15	
7	F	Medium	10	

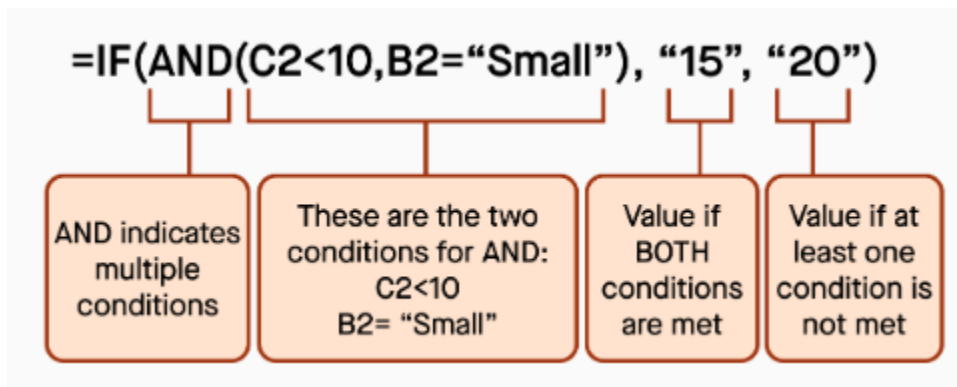
You want to try pricing the items using the following logic: If the cost to make is less than \$10 and it's a small item, price it at \$15. If both conditions aren't met, then price the item at \$20. The formula for this logic is an IF statement using AND. This is the structure:

=IF(AND(condition1, condition2), value_if_TRUE, value_if_FALSE)

The formula for cell D2 is shown below. The first condition is met; the cost is less than \$10. But the second condition isn't met; it isn't a small item. Because both conditions aren't met, the price is set at \$20.

D2							
	A	B	C	D	E	F	G
1	Paper Craft	Size	Cost to make (\$)	Price			
2	A	Large	9	20			
3	B	Medium	9				
4	C	Small	5				
5	D	Large	25				
6	E	Medium	15				
7	F	Medium	10				

The following diagram further describes the formula used in the example above:



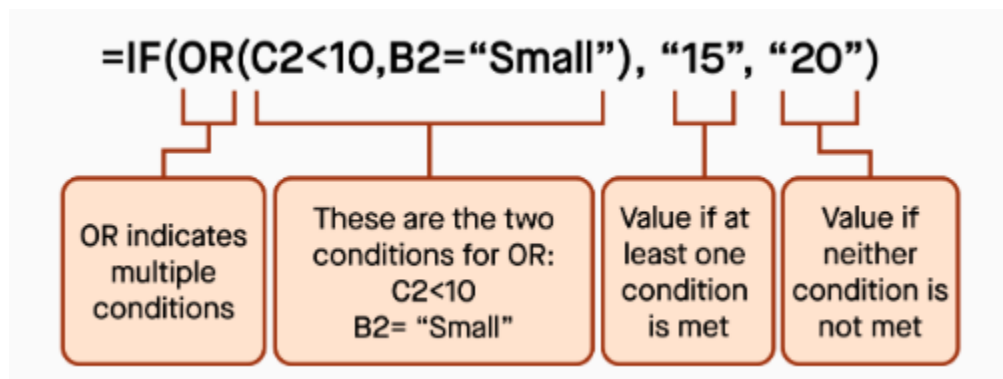
Now, imagine that you want the pricing to use slightly different logic: If the cost is less than \$10 or if it is a small item, price it at \$15. If either of these conditions isn't met, then price the item at \$20. Here, you would use OR instead of AND, like this:

=IF(OR(condition1, condition2), value_if_TRUE, value_if_FALSE)

In the example below, the first condition is met because the cost is under \$10. But the second condition isn't met because it isn't a small item. Because one of the conditions was met, the formula is evaluated as TRUE, and the price is set at \$15.

D2							
	A	B	C	D	E	F	G
1	Paper Craft	Size	Cost to make (\$)	Price			
2	A	Large	9	15			
3	B	Medium	9				
4	C	Small	5				
5	D	Large	25				
6	E	Medium	15				
7	F	Medium	10				

The following diagram further describes the formula used in the example above:



Nesting with INDEX and MATCH

This lesson demonstrates how to create nested functions with INDEX and MATCH. By using INDEX and MATCH together, you can create powerful formulas and add multiple conditions to logical tests.

Recall the INDEX and MATCH functions that you learned in the previous lessons:

=INDEX(array, row_number, column_number)

=MATCH(lookup_value, lookup_array, match_type)

Although INDEX and MATCH can be used alone to create simple formulas, they are usually used together. When MATCH is nested inside of INDEX, together, they act like a more powerful VLOOKUP.

When using a VLOOKUP function, the lookup value must be in the leftmost column of the table array. If it isn't, then the VLOOKUP will return an #N/A error.

Using MATCH and INDEX together allows you to perform a VLOOKUP without this limitation. As you might imagine, data isn't always conveniently organized for a VLOOKUP. Sometimes, you want to look up a value in a column on the right and retrieve a value to the left of it.

Nesting with INDEX and MATCH functions

The spreadsheet below shows the same students and project scores that you saw in the previous lesson's example. But there is an additional table on the right that shows each project name and its scores.

	A	B	C	D	E	F	G	H
1	Student	Score					Project	Score
2	Emily	85					Book Report	95
3	Stuart	70					Presentation	85
4	Kim	65					Poster	65
5	Dante	90					Skit	55
6	Levi	94					Demonstration	88
7	Laura	60					How-To Presentation	90
8	Damon	95					Volcano	84
9	Sue	84					Toothpick Tower	70
10	Ronita	88					Model Solar System	60
11	Tara	55					Scrapbook	94

You can pull each student's name into column F, next to their corresponding projects, using a MATCH function nested in an INDEX function. Compare the formula structure for a regular INDEX function above to the formula structures for nested INDEX and MATCH functions below:

=INDEX(array, nested_MATCH_function, column_number)

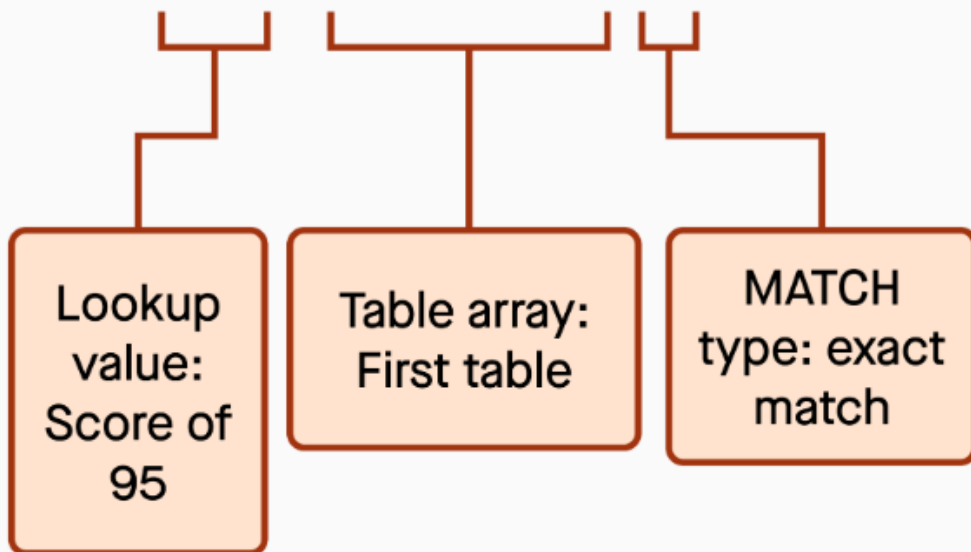
Or

=INDEX(array, row_number, nested_MATCH_function)

Think about it this way: MATCH is used as either the row_number argument or the column_number argument inside of the INDEX function. When creating this nested function, it's helpful to write out the functions separately—first the inner MATCH function, then the outer INDEX function—before combining them.

Recall that MATCH functions find positions of data. To start with the MATCH function, first figure out what the common field is between the two tables. Here, the common field is Score. Then you can write the inner MATCH function can be written to look up the common field. In this case, Score is in both tables, so focus your MATCH function on the lookup value for project score. The first project, called Book Report, has a score of 95. So create a MATCH function that looks for a score of 95 in the first table. It will look like this:

MATCH(H2, B2:B11,0)

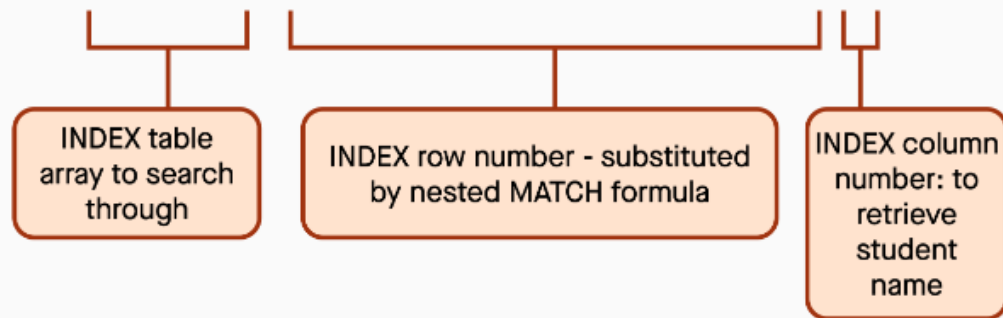


The formula `=MATCH(H2,B2:B11,0)` searches for a score of 95 in the first table, and returns its position. If the returned position is 7, that means that it's in the seventh row in the array B2:B11 (note that's it's in the seventh row of the array, not the seventh row of the spreadsheet). Because it's the seventh row of the array, plan on using this MATCH function as the row index number in the outer INDEX function. Now, set this aside for a moment.

Next, set up the outer INDEX function. Recall that INDEX functions find values of data at a row-column intersection. In this example, you want to retrieve student names from the first table. For the project Book Report, start a formula in cell F2 with `=INDEX(`. The first argument is the array that the formula searches through, which is A2:C11. The next argument is the row index number. Remember that you just created a MATCH function above to retrieve a row position; now, it's time to nest that MATCH function as the row index argument in this INDEX function. The last argument is the column index number. You want to retrieve the corresponding student name, which is in the first column of the lookup array A2:C11. So this argument is 1.

Putting it all together, here's the final INDEX formula with the nested MATCH function:

=INDEX(A2:B11, MATCH(H2, B2:B11,0),1)



The result is Damon, the student who received a score of 95%.

When dragging this formula down a column or row, always remember to lock the cell references for the table and lookup arrays! Leaving them unlocked allows the arrays to shift down as you drag the formula, which can lead to an incorrect result.

If you try this using a VLOOKUP, you'll get an error. VLOOKUP functions cannot retrieve values to the left of the lookup value. This is why INDEX and MATCH work well together. You can locate and retrieve specific data no matter what the layout of the table is. You can also use INDEX and MATCH together to pull data from a different sheet or workbook.

Check out the project using the School Rankings dataset that utilizes the concept of nesting functions like INDEX and MATCH to accomplish complex searches that extend beyond the capabilities of VLOOKUP and HLOOKUP.