BestExcel Example. Ever.

Maybe an overstatement, but you know me!

Here is a common problem that has been a pain point for modellers for many years. Imagine you had a dataset of customer names (*say*):

Fictitious Data

Names	
Claudio Lam	
Josefina Dickersor	1 I
Britt Vazquez	
Hans Patel	
Francisco Valenzue	ela
Marylou Mccarthy	
Imelda Lloyd	
Carol Stevenson	
Stuart Riddle	
Ned Lowe	
Kim Mccall	
Mamie Wong	
Eli Galvan	
Josie Pineda	
Mohamed Allison	
Lincoln Bond	
Waldo Spencer	
Gabriela Moore	
Renato Simmons	
Malcolm Newman	
Max Smith	
Edwin Sparks	
Chalcas Cilhart	

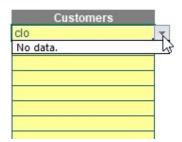
In the example in the attached workbook, I have put 200 random names in (but this method will work for up to 10,000). What I want to do is create an output sheet where I can select a name from a dropdown list, but with it filtering for any letters I have already typed into the cell, *viz*.

Outputs

Customers	
cl	
Clare Whitney	ſ
Clarissa Archer	
Claudio Lam	
John Cline	
Vito Clark	

Do you see how I have typed in "cl" and it has returned any names with "cl" in the name – including names not necessarily starting with these two letters (*e.g.* John <u>Cl</u>ine). Furthermore, if I type a further letter, say "o", this then becomes:

Outputs



How good would that be when you have long lists, unordered, with potential duplicates? No more endless scrolling – simply type in several key letters and choose from a filtered selection instead. Accountants, analysts, managers and modellers have been striving for this sort of functionality in Excel *for years*.

Previous Solutions

People have developed VBA solutions or used Excel's ability to AutoComplete. For example, if you ensure Formula AutoComplete is enabled (**File -> Options**, or **ALT + T + O**) by checking the appropriate box in the 'Working with formulas' section of 'Formulas',

cel Options		?
General Formulas	\blacksquare_{fx} Change options related to formula calculation, perfor	mance, and error handling.
Data	Calculation options	
Proofing Save Language Ease of Access Advanced	Workbook Calculation	Enable iterative calculation Maximum Iterations: 100 Maximum <u>C</u> hange: 0.001
Customize Ribbon Quick Access Toolbar Add-ins	Working with formulas B1C1 reference ctyle ① Formula AutoComplete ① Use table names in formulas V Use GetPivotData functions for PivotTable references	
Trust Center	Suggest formula variations that are supported by older version	ns of Excel ①
Trust Center		is of Excel ①
Trust Center	Suggest formula variations that are supported by older version	
Trust Center	Suggest formula variations that are supported by older version Error Checking Enable background error checking Reset langred E	

You may then hide a complete list above your input cell (ensuring there are no gaps and start typing):



Here, I started typing the name "Hunter Short" into cell **F213**. Once I typed the third letter in, Excel was ready to AutoComplete, as there was only one option remaining. This is one option, but:

- It will not work with lists containing duplicates
- It assumes you are typing the first letters of the name in
- You have to replicate this list immediately above the input cell(s)
- It only displays once Excel has ruled out all other alternatives.

This is not ideal – so I shall stick with my plan instead.

New Solution

This solution will only work in Office 365 as it relies on the new feature, dynamic arrays. You might think I am being a little niche in this instance. However, if you do not have Office 365, keep reading. This might convince you it's time to make the switch.

To start, I want to turn my data table, which I will assume is on a worksheet called **Data**, into an Excel Table. I highlight the table and choose Table from the Tables group on the Insert tab of the Ribbon (**CTRL + T**). Since the first row is the heading, I ensure the 'My table has headers' check box is ticked in the 'Create Table' dialog:

Create Table	?	×
Where is the data for y	our table?	
=\$F\$12:\$F\$213		Ť
☑ <u>M</u> y table has h	eaders	
ОК	Can	cel

As you will see later, creating a Table is like a double-edged sword: it is useful as it will allow us to add more names to the list and the range will automatically extend, but it will cause us headaches elsewhere.

Having named the table 'Customers' in the 'Table Design' tab, I add an input column to my Table:

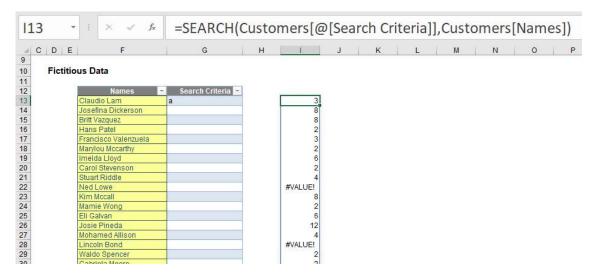
Fictitious Data

Names	-	Search Criteria 📼
Claudio Lam		
Josefina Dickerson		
Britt Vazquez		
Hans Patel		
Francisco Valenzuela		
Marylou Mccarthy		
Imelda Lloyd		
Carol Stevenson		
Stuart Riddle		
Ned Lowe		
Kim Mccall		
Mamie Wong		
Eli Galvan		
Josie Pineda		
Mahamad Alliaan		

I can now start to build up my helper formulae. The first function I am going to call upon is SEARCH.

SEARCH(find_text, within_text, [start_number]) is a search function which is <u>not</u> case sensitive but does allow for wildcard characters. It seeks out the first instance of a character or characters (typed in inverted commas) in the **within_text** text string. The **start_number** argument is optional (hence the square brackets in the syntax) so that the first few characters in a text string may be ignored. If the **find_text** cannot be located within **within_text**, the error *#VALUE!* is returned.

As in the image (*below*), I add a formula, but I exclude it from the Table by leaving an empty column between them, *viz*.



I will explain why this formula is not incorporated into the Table in a moment. However, let's first take a look at the formula which was typed into cell **I13** <u>only</u>:

=SEARCH(Customers[@[Search Criteria]],Customers[Names])

Customers[@[Search Criteria]] is the structured referencing syntax (*i.e.* how formulae work when references are from within an Excel Table) stating the **Search Criteria** item (*sic*) for that row. This is what the **@** symbol denotes – it's not the Table's Twitter handle. In this case, this is cell **G13**, where I have typed in "a".

Customers[Names] denotes the entire **Names** field. Presently, this is represented by cells **F13:F212** (*i.e.* 200 records) in my example. However, if I were to add names to the bottom of the list, the range would extend automatically. This is why I put this data in a Table: my formula is flexible.

If I had simply referred to the first row rather than the entire range, the **SEARCH** formula would have sought the character "a" in cell **F13** – "Claudio Lam" – and returned the position of the first "a", which would be 3 (third character). However, I did it for the entire range, so this formula *spilled*: it added formulae down the entire column to match the length of the source argument and found the "a" in each row's entry of **Customers[Names]**. This is what Office 365 will do (if you have Office 365 and it doesn't, be patient, the update is *very* close). You have created a **dynamic array**: it's dynamic as the formula will extend / contract automatically as the source data length changes.

The reason I have not included this formula in the Table (which would seem to make more sense) is as follows. Let's imagine I had:

	13		fx =SEARCH	H([@[Sear	un un	itenaj	1)[110	mesj
-A	CDE	F	G	Н	1	J	к	L
9								
10	Fictitio	ous Data						
11								
12		Names	🗾 🗧 Search Criteria	Column1 🔫				
13		Claudio Lam	а	#SPILL!				
14		Josefina Dickerson		#SPILL!				
15		Britt Vazquez		#SPILL!				
16		Hans Patel		#SPILL!				
17		Francisco Valenzuela	а	#SPILL!				
18		Marylou Mccarthy		#SPILL!				
19		Imelda Lloyd		#SPILL!				
20		Carol Stevenson		#SPILL!				
21		Stuart Riddle		#SPILL!				
22		Ned Lowe		#SPILL!				
23		Kim Mccall		#SPILL!				
24		Mamie Wong		#SPILL!				
25		Eli Galvan		#SPILL!				
26		Josie Pineda		#SPILL!				
27		Mohamed Allison		#SPILL!				
28		Lincoln Bond		#SPILL!				
29		Waldo Spencer		#SPILL!				
30		Cabriela Moore		#SPILL				

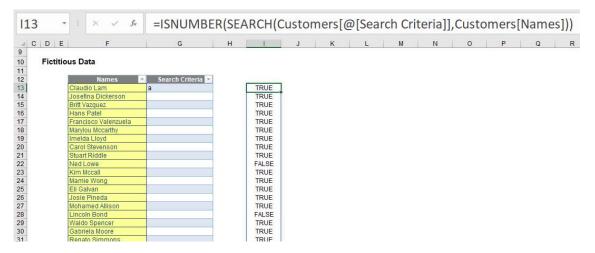
In this instance, the formula would have simplified because there is no reason to specify the Table name (**Customers**). However, it doesn't work. The spill feature is not supported in Excel Tables – hence the double-edged sword I was referring to earlier – so this formula <u>must</u> be excluded from the source Table. That will be a source of frustration later (oh yes, I do like to keep you on tenterhooks).

Returning to our current situation,

<mark> 1</mark> 3	Ŧ	Ŧ	$\times \checkmark f_x$	=	SEARCH	(Cu	usto	om	ers[@[Sea	arc	h C	rite	eria	a]],	Cus	to	me	rs[l	Var	ne	s])
	DE		F		G		н		1		J		к		L		М		N		0		P
	Fictitio	us Da	ata																				
11 12 13 14 15 16 17 18 19 20 21 22 22 22 22 22 22 22 22 22			Names	-	Search Criteria	+		_		-													
13			dio Lam	а						3													
14			ina Dickerson							3													
15			/azquez							3													
6			Patel			_				2													
7			cisco Valenzuela							3													
8		Maryl	ou Mccarthy						3	2													
9		Imeld	la Lloyd						(3													
20		Carol	Stevenson						3	2													
1		Stuar	t Riddle						4	4													
2		Ned L	Lowe					#	VALUE!														
13		Kim I	/iccall					200	1	3													
24		Mami	e Wong							2													
15		Eli Ga	alvan						(5													
26		Josie	Pineda			_			1:	2													
27		Moha	med Allison							4													
28		Linco	In Bond					#	VALUE!														
29			o Spencer					23		7. E													
20			ala Maara			-				5													

do you see two of the entries in my extract (rows 22 and 28) return *#VALUE!* errors? This is what happens when there is no "a" (in this example) in these text strings ("Ned Lowe" and "Lincoln Bond" respectively). I don't care *where* my "a" occurs, only that it occurs; further, I wish to remove my errors too.

Therefore, I use the **ISNUMBER** function. One of 12 **IS** functions, this one is best suited here as it filters between numbers (TRUE) and everything else (FALSE) – including errors:



I could now filter on column I to produce a list in column F of those names which contain the letter "a" only. However, I don't want to do it this way. I want to formulaically filter it, so I use the function new to Office 365 (which supports and creates dynamic arrays), **FILTER**.

FILTER Function

The **FILTER** function will accept an array, allow you to filter a range of data based upon criteria you define and return the results to a spill range.

The syntax of **FILTER** is as follows:

=FILTER(array, include, [if_empty]).

It has three arguments:

- **array:** this is required and represents the range that is to be filtered
- include: this is also required. This specifies the condition(s) that must be met
- **if_empty:** this argument is optional. This is what will be returned if no data meets the criterion / criteria specified in the **include** argument. It's generally a good idea to at least use "" here.

For example, consider the following source data:

	CDE	F	G	Н	1
9					
10	Origina	Data			
11					
12		Item	Shape	Colour	Sides
13		1	Triangle	Red	3
14		2	Rectangle	Amber	4
15		3	Circle	Green	1
16		4	Triangle	Red	3
17		5	Square	Blue	4
18		6	Rectangle	Blue	4
19		7	Rectangle	Amber	4
20		8	Circle	Amber	1
21		9	Triangle	Red	3
22		10	Square	Green	4
23		11	Circle	Blue	1
24		12	Square	Amber	4
25		13	Triangle	Blue	3
26		14	Circle	Green	1
27		15	Rectangle	Blue	4
28					

It is simple to **FILTER**:

F3	6		٣	: ×		f _x =FILTE	R(F12:127,G	12:G27=G	33,"Not L	ocated.
4	с	D	E	F	G	н	1	J	K	L
30										
31		Filt	er R	esults						
32										
33				Shape	Triangle					
34										
35				Item	Shape	Colour	Sides			
36				1	Triangle	Red	3			
37				4	Triangle	Red	3			
1000				g	Triangle	Red	3			
38				47	Triangle	Blue	3			
31 32 33 34 35 36 37 38 39 40				13	mangle	Dide	5			

Here, in cell **F36**, I have created the formula

=FILTER(F12:I27,G12:G27=G33,"Not Located.")

F12:I27 is my source **array** and I wish only to **include** shapes (column **G12:G27**) that are 'Triangles' (specified by cell **G33**). If there are no such shapes, then **"Not Located."** is returned instead. To show this, I will change the shape as follows:

F3	6		٣	: ×	√ f _x	=FILTER(F	F12:127,G12:0	7,G12:G27=G33,"Not Locat				
4	с	D	E	F	G	н	1	J	K	L.		
30												
31		Filt	er Re	sults								
33			S	Shape	Pentagon							
34												
35				Item	Shape	Colour	Sides					
			I.	lot Located.								
32 33 34 35 36 37				voi Localeu.								

Returning to Our Solution

Therefore, let me use the following formula:

=FILTER(Customers[Names],ISNUMBER(SEARCH(Customers[@[Search Criteria]],Customers[Names])),"No data.")

This merely builds on the last **ISNUMBER** calculation (which is the criterion), applying it to the field **Customers[Names]**, with "No data." returned should there be no matching results. Having modified my input requirements from "a" to "au" (feeling Australian today), my calculation would appear as follows:

4 C	DE	F	G	н	L
9					- 64
10	Fictitio	ous Data			
11					
12		Names	 Search Criteria 		
13		Claudio Lam	au		Claudio Lam
14		Josefina Dickerson			Elinor Kaufman
15		Britt Vazquez			Paul Norman
16		Hans Patel			Aubrey Giles
17		Francisco Valenzuela			Maurice Jacobson
18		Marylou Mccarthy			Ruthie Daugherty
19		Imelda Lloyd			Mauricio Bautista
20		Carol Stevenson			Delores Krause
21		Stuart Riddle			
22		Ned Lowe			
23		Kim Mccall			
24		Mamie Wong			
25		Eli Galvan			
26		Josie Pineda			
27		Mohamed Allison			
28		Lincoln Bond			
29		Waldo Spencer			
30		Gabriela Moore			
31		Renato Simmons			

My list of TRUE and FALSE entries is no more, and has been replaced by the names that match the criterion (must contain "au"). It's starting to come together now. Some of you might think you now have your data, but you can do better by ensuring there are no duplicate entries and sorting the data. This requires two functions: **UNIQUE** and **SORT**, both new to Office 365 and both supporting the brave new world of dynamic arrays.

UNIQUE Function

Bizarrely, **UNIQUE** details distinct items (*i.e.* provides each value that occurs with no repetition) and also it can return values which occur once and only once in a referred range. I will be focusing on the former use here.

The **UNIQUE** function has the following syntax:

=UNIQUE(array, [by_column], [occurs_once]).

It has three arguments:

- **array:** this is required and represents the range or array from which to return unique values
- **by_column:** this argument is optional. This is a logical value (TRUE / FALSE) indicating how to compare. If you wish to compare by row, the argument should be FALSE or omitted (since this is the default). To compare by column, you will need to select TRUE
- occurs_once: this argument is also optional. This requires a logical value too:
 - **TRUE:** only return unique values that occur once
 - FALSE: include all distinct values (default if omitted).

It's probably clearer with an example. Consider the following source data:

	С	D	Е	F	G	Н	1
9	0						
10		Ori	gina	al Data			
11							
12				Store	Salesperson	Section	Manager
13				North	Alice	White Goods	Zack
14				North	Barbara	Groceries	Zack
15				North	Charlie	White Goods	Zack
16				North	Dion	Computers	Yvonne
17				North	Echo	Insurance	Xander
18				North	Fred	Bedding	Winnie
19				North	George	Audio Video	Yvonne
20				North	Helen	Furniture	Winnie
21				North	Iris	White Goods	Zack
22				North	Jack	Furniture	Winnie
23				North	Karla	Groceries	Zack
24				East	Lindsay	Insurance	Xander
25				East	Barbara	Groceries	Zack
26				East	Iris	White Goods	Zack
27				East	Michael	Computers	Yvonne
28				East	Fred	Bedding	Winnie
29				East	Dion	Computers	Yvonne
30				South	Nancy	Audio Video	Yvonne
31				South	Oprah	Furniture	Winnie
32				South	Helen	Furniture	Winnie
33				South	Alice	White Goods	Zack
34				South	Pete	Groceries	Zack
35				West	Karla	Groceries	Zack
36				West	Pete	Groceries	Zack
37				West	Charlie	White Goods	Zack
38				West	Dion	Computers	Yvonne
39				West	George	Audio Video	Yvonne
40				West	Nancy	Audio Video	Yvonne
41				West	Michael	Computers	Yvonne
12							

I can derive the unique items in each list:

L13				X V	fx =UNIQUE	(F13:F41)	
4	J	L SI	<	L	M	N	0
9				- W - 1103			
10		Res	ults	5			
11							
12				Store	Salesperson	Section	Manage
13				North	Alice	White Goods	Zack
14				East	Barbara	Groceries	Yvonne
15				South	Charlie	Computers	Xander
16				West	Dion	Insurance	Winnie
17					Echo	Bedding	
18					Fred	Audio Video	
19					George	Furniture	
20					Helen		
21					Iris		
22					Jack		
23					Karla		
24					Lindsay		
25					Michael		
26					Nancy		
27					Oprah		
28					Pete		
29							

In cell L13, I have simply typed

=UNIQUE(F13:F41).

No optional arguments; everything in default. This has simply listed each store that appears; if "North" and "North" (extra space) were there, then both would appear. **UNIQUE** is not case sensitive though and each entry would appear as it first occurs reading down the range **F13:F41**.

SORT Function

The **SORT** function sorts the contents of a range or array:

=SORT(array, [sort_index], [sort_order], [by_column]).

It has four arguments:

- array: this is required and represents the range that is required to be sorted
- **sort_index:** this is optional and refers to the position of the row or the column in the selected **array** (*e.g.* second row, third column). 99 times out of 98 you will be defining the column, but to select a row you will need to use this argument in conjunction with the fourth argument, **by_column**. And be careful, it's a little counter-intuitive! The default value is 1
- **sort_order:** this is also optional. The choices for **sort_order** are 1 for ascending (default) or 1 for descending. It should be noted that you might not want to hold your breath waiting for 'Sort by Color' (*sic*), 'Sort by Formula' or 'Sort by Custom List' using this function
- **by_column:** this final argument is also optional. Most people want to sort rows of data, so they will want the value to be FALSE (which is the default value if not specified). Should you be booking your mental health check, you may wish to use TRUE to sort by column in certain instances.

Again, it's simple. Consider the following data:

	C D E	F	G	Н
9			12 A	
10	Origina	al Data		
11				
12		First Name	Last Name	Points
13		lvan	Idea	717
14		Amanda	Hugankiss	885
15		Artie	Detoo	976
16		Blake	Seven	508
17		Piper	Pied	978
18		Ivana	Tinkle	508
19		Artie	Chokes	300
20		Mike	Stand	778
21		Shelley	Ack	954
22		Blade	Runner	203
23		Sheikh	Spear	711
24		Mike	Robe	305
25		Daley	News	839
26		Hugo	There	611
27		Mimi	Selfish	197
28				

Sorting the 'Points' column in order is easy as this:

F3:	2		٣	1	\times	1	$f_{\rm x}$	=SO	RT(H13	3:H27
A	С	D	E	1	F	G	1	н		T
29				_						
30		So	rted	Poin	ts					
31										
32			[197					
33			1	-	203					
34					300					
35					305					
36					508		4	5		
37					508					
38					611					
39					711					
40					717					
41					778					
42					839					
43					885					
44					954					
45					976					
46					978					
47										

All you have to do is type **=SORT(H13:H27)** into cell **F32**. That's it. However, do note that the duplicates are repeated; there is no cull. That's why it is needed here alongside **UNIQUE**.

Returning to Our Solution Again

Here, I need to remove duplicates and sort my data. It does matter the order I perform these calculations: {2, 1, 3} is easier to sort than {1, 2, 1, 2, 2, 1, 3, 3, 1, 1, 2, 3, 2, 1}. I should remove duplicates first, then sort.

This is an important mindset to get into: working with dynamic arrays can mean you start taking for granted some rather voluminous but unnecessary tasks otherwise. Putting functions in the wrong order can make the difference between a one and a 10 second calculation. Therefore, my formula extends to

=SORT(UNIQUE(FILTER(Customers[Names],ISNUMBER(SEARCH(Customers[@[Search Criteria]],Customers[Names])),"No data.")))

4 C	DE	F	G	н	L
9					
10	Fictitious	s Data			
1					
2		Names	 Search Criteria 		
3	C	laudio Lam	au		Aubrey Giles
4	J	osefina Dickerson			Claudio Lam
5	B	Iritt Vazquez			Delores Krause
6	H	lans Patel			Elinor Kaufman
7	F	rancisco Valenzuela			Maurice Jacobsor
8	IV	larylou Mccarthy			Mauricio Bautista
9	Ir	melda Lloyd			Paul Norman
0	C	arol Stevenson			Ruthie Daugherty
1	S	tuart Riddle			
2	N	led Lowe			
3	K	im Mccall			
4	IV.	lamie Wong			

Now comes a trick: instead of having this list propagate vertically, I wish it to fill horizontally. I can achieve this with the **TRANSPOSE** function:

=TRANSPOSE(SORT(UNIQUE(FILTER(Customers[Names],ISNUMBER(SEARCH(Customers[@[Search Criteria]],Customers[Names])),"No data."))))



Yes, the graphic is tiny – but it's more about the concept than the detail here. My list now extends across the row, which means I can now copy my formula down column I, *viz*.

	С	D	E	F	G	н	1	J	K	L
9		-								
10		Fic	titio	us Data						
11										
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26				Names	 Search Criteria 					
13				Claudio Lam	au		Aubrey Giles	Claudio Lam	Delores Krause	Elinor Kaufi
14				Josefina Dickerson	bc		No data.			
15				Britt Vazquez	de		Arden Ayala	Brianna Alexa	Broderick Gilles	Christa Fre
16				Hans Patel	fo		April Fox	Ethel Forbes	Rodolfo Scott	Vincenzo St
17				Francisco Valenzuela	ab		Gabriela Moore	Gabrielle Patr	Jayson Abbott	
18				Marylou Mccarthy			Broderick Gillespie	Carmela Mull	i Carolina Russel	Cheryl Willi
19				Imelda Lloyd	gr		Grady Ellis	Grover Escob	Judith Griffin	Lupe Ingrar
20				Carol Stevenson	ea		Chelsea Gilbert	Earlene Hest	Earline Shepher	Gerry Onea
21				Stuart Riddle	rc		Clarissa Archer	Marcella Ever	Marco Bright	Marcos Sch
22				Ned Lowe	nab		No data.			
23				Kim Mccall	mc		Darrin Mcneil	llene Mcguire	Kim Mccall	Leroy Mcint
24				Mamie Wong	dr		Adriana Todd	Sandra Potts	Solomon Andrac	le
25				Eli Galvan	gra		Grady Ellis	Lupe Ingram		
26				Josie Pineda	nc		Francisco Valenzuela	Lincoln Bond	Robert French	Vincent Mcc

Hopefully, my idea is becoming clearer. I can use the criteria typed in column **G** to generate the spilled horizontal lists in column **I** onwards. Therefore, if I change the contents of column **G** to

reference the data in other cells where I want my output dropdown boxes to be, I can then create lists for these cells based upon column I onwards.

Hence, on a separate worksheet, I now undertake some workings:

	С	D	E	F	G	Н	I	J	K	L	M
9											
10		Wo	orkings								
11											
12			We don't recom	nmend you put these on an output	page in general						
13 14								_			
				Cell in row above first list			Lists	=Data	a!/12		
15 16				Formula			=Data!I12	=IFEF	RROR(F	ORMULATE.	XT(I14), "")
16				Sheet name and column			Data!I	=MID	(I15,2,LE	EN(115)-LEN((117)-1)
17 18				Row number			12	=ROV	N(Data!I	12)	
18											

It might not seem obvious what I am doing, so allow me to explain. I want to set a reference cell immediately above where the formula for my first dynamic list is (*i.e.* cell **I13** on the Data worksheet). This is so that I can set a base cell for my source data.

If you cannot follow the formula in cell **114**, then I humbly suggest Excel may not be for you; cell **115** then displays the formula in cell **114** using **FORMULATEXT** (with an error trap in case of unforeseen issues). This has then catered for any change in cell for the base cell, or a change of sheet name.

I will skip a formula momentarily: the calculation in cell **I17** (=**ROW(Data!I12)**) merely generates the row number, so that the formula in cell **I16** is

=MID(I15,2,LEN(I15)-LEN(I17)-1)

which is not easily understandable as it could be!

The function LEN determines the length of a text string. Therefore:

- LEN(115) determines the length of =Data!!12, which is nine (9) characters
- LEN(117) determines the length of the row number (12), which is two (2) characters
- LEN(I15)–LEN(I17) is therefore the length of =Data!I (no row number reference), which is seven (7) characters
- LEN(I15)-LEN(I17)-1 is one character less. This is not to get rid of the column reference (I) but actually the equals sign (=) at the beginning. This will become clearer shortly.

MID(text, start_number, n) extracts **n** characters from the referenced **text** string starting with the character in position **start_number**. Thus,

=MID(I15,2,LEN(I15)-LEN(I17)-1)

will extract six (6) characters from the text string in cell **I15** (**=Data!I12**), starting at the second character (*i.e.* ignoring the equals sign). This gives the result **Data!I**.

This all does beg the question, so what?

I want to create a set of customisable drop-down lists starting in cell **F23**. To assist, I add a Helper column in column **E**:

E23	•	1 × ×	<i>f</i> _x =\$	\$1\$16&(\$1\$2	17+ROW	S(\$E\$22	:\$E22))	&"#"	
123		C D	E	F		G	Н	111	J
-	9	147							
	10 11	Workin	gs						
	11	Wed	on't recomi	mend you put these	e on an output	nade in dener	ral		
	13		ontrecom	mena you put these	c on an output	page in gener	ci.		
	14			Cell in row above	e first list			Lists	1
	15			Formula				=Data!I1	12
	16			Sheet name and	column			Data!I	
	17			Row number				12	
	18 19								
		Output							
1 甲.	20 21	Outputs	5						
	22	Ĥ	elper	Custom	ers	Ĺ			
	23	-	! 13#			1			
	24	Data	! 14#			1			
	25	Data	115#						
	26		!!16#						
	27		117#						
	28	Data	!!18# !!10#			-			

Cells F23 down are then referred to back on the Data sheet:

G1	L3 • : × ✓ fx	=IF('Resulti	ng List'!F23="","",'Resulting List'!F23)
	D E F	G	н
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	Fictitious Data		
12	Names	Search Criteria	
13	Claudio Lam		
14	Josefina Dickerson		
15	Britt Vazquez		
16	Hans Patel		
17	Francisco Valenzuela		
18	Marylou Mccarthy		
19	Imelda Lloyd		
20	Carol Stevenson		
21	Stuart Riddle		
22	Ned Lowe		
23	Kim Mccall		
24	Mamie Wong		

This will now drive my dynamic lists in column I onwards.

Returning to the output sheet, the formula in cell **E23** needs explanation:

=\$I\$16&(\$I\$17+ROWS(\$E\$22:\$E22))&"#"

Through the concatenation operator (&), this formula joins up the text in cell **I16** (**Data!I**) with a number that starts with 12 (cell **I17**) and adds on the number of rows from row 22 (**\$I\$17+ROWS(\$E\$22:\$E22)**). This is then joined to "#" to form

Data!I13#

The **#** symbol is known as the spilled range operator and denotes the full range (given that it is dynamic and may vary in size). Next, I create a dropdown list in cell **F23**. To do this, I select this cell, then go to 'Data Validation' in the 'Data Tools' group of the Data tab of the Ribbon (ALT + D + L),

Settings	Input Message	Error Alert		
Validation	criteria			
Allow:				
List		✓ Ignore blank		
Data:		In-cell dropdow	/n	
betwee	n	\sim		
Source:				
=Data!	13#	1	<u>E</u>	
	these changes to a	all other cells with the sar	me settings	

On the Settings tab of the resulting dialog box, I choose to List from the 'Allow:' selection and initially, I could type in **=Data!!13#** as the source (this has to be typed as **#** will not appear automatically by selecting a range). If I selected the entire possible range instead this would make the data validation lists unnecessarily large and show many blank rows needlessly.

This would give an error if I tried to start typing something:

Outputs

Helper	Customers	
Data!I13#	ab	
Data! 14#		
Data!!15#		Microsoft Excel
Data!116#		
Data!I17#	1	This value doesn't match the data validation restrictions defined for this cell
Data!I18#		
Data! 19#		Retry Cancel Help
Data!/20#		
Datall21#		

Therefore, I need to go back to the Data Validation dialog and click on the third tab, 'Error Alert':

Settings	Input Message	Error Alert	
<u>Show</u>	error alert after inv	alid data is entered	0
When use	r enters invalid da	ta, show this error aler	t
Style:		Title:	
Stop		~	
		Error message:	
			~
			~
		h	

I must uncheck 'Show error alert after invalid data is entered'. After clicking OK, it would then work as envisaged.

1	С	D		E	F	
19			1,			
20		Ou	tput	s		
21						
22			H	elper	Customers	
23			Data	! 13#	ab	-
24			Data		ela Moore	
25			Data	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	elle Patrick	45
26			Data	Jaysor	Abbott	
27			Data	17#		
28			Data	! 18#		
20			Data	1110#		

However, I would have to create each dropdown individually (due to the #), which would be a pain if I were to add 200 dropdown boxes (*say*). Ain't nobody got time for that. Therefore, I turn to one of the most divisive functions in Excel...

INDIRECT Function

Excel's **INDIRECT** function allows the creation of a formula by referring to the contents of a cell, rather than the cell reference itself. The **INDIRECT(ref_text, [a1])** function syntax has two arguments:

- ref_text: this is a required reference to a cell that contains an A1-style reference, an R1C1style reference, a name defined as a reference or a reference to a cell as a text string. If ref_text is not a valid cell reference, INDIRECT returns the #REF! error value. If ref_text refers to another workbook (an external reference), the other workbook must be open. If the source workbook is not open, INDIRECT again returns the #REF! error value
- 2. **[a1]:** this is optional (hence the square brackets) and represents a logical value that specifies what type of reference is contained in the cell **ref_text**. If **a1** is TRUE or omitted, **ref_text** is interpreted as an A1-style reference (*e.g.* **A1**, **C5**, **J199**). If **a1** is **FALSE**, **ref_text** is interpreted as an R1C1-style reference.

Essentially, **INDIRECT** works as follows:

	А	B C D E F	G	Н	1
1		Basic Concept			
2		INDIRECT Examples			
3		Go to Table of Contents			
4	+	()			
5					
6					
7		Basic Concept			
8					
9		Simple Illustration			
10					
11		Cell Reference	e	H13	
12					
13		Value		187	
14					
15					
16		INDIRECT Example			
17					
18		Simple		187	
19					
20					

In the above example, the formula in cell H18 (the yellow cell) is

=INDIRECT(H11).

With only one argument in this function, **INDIRECT** assumes the A1 cell notation. Note that the value in cell **H11** is **H13**, so this formula returns the value / contents of cell **H13**, *i.e.* 187.

Completing Our Solution

Now do you see why I have that Helper column in column E? I modify my data validation list (ALT + D + L) one final time:

23	٣	1.2	< - fx											
2 3	1	C	D E	F	G		Н	1	J	к	1	L		
ŀ	19													
-	20	0	utputs											
•	21													
*	22		Helper	Customers		Data Valio	lation				?	×		
	23		Data!I13#		Ŧ	-								
	24		Data!I14#		T	Settings	Input M	essage Er	ror Alert					
	25		Data!I15#			Validation criteria								
	26		Data!I16#			Allow: List V Ignore blank Data: In-cell dropdown between V			blank					
- 2	27		Data!I17#						10000					
	28		Data!I18#						1					
· •	29		Data!I19#		-									
	30		Data!120#			Source: =INDIRECT(\$E23)				1				
	31		Data!!21#											
	32		Data! 22#											
· · .	33		Data!123#			Appl	y these char	iges to all o	ther cells wit	th the same	settings			
	34		Data!!24#											
- 2	35		Data!!25#			<u>C</u> lear A	dt			OK	Ca	incel		
	36		Data!!26#		1				7.5			-		

I have replaced **=Data!!13#** with **=INDIRECT(\$E23)** – that is the equivalent of **=Data!!13#**. This step allows the data validation to be copied down the range, *viz*.

1	С	D		E	F	
19						
20		Ou	tputs			
21						
22			He	lper	Customers	1.0
23			Data!	113#	Amanda Beck	
24			Data!	14#	Chelsea Gilbert	
25			Data!	115#	rc	-
26			Data!		a Archer	
27			Data!		la Everett	
28			Data!	Marco	Bright s Schaefer	6
29			Hays			
30			Data!	Zelma	SERVICE NOT A CONTRACT OF A	
31			Data!	21#		
32			Data!	122#		
33			Datal	123#		

Success! The <u>attached Excel file</u> provides the full example for review.

Word to the Wise

This article has only required access to dynamic ranges, data validation, creating a Table, structured referencing, three operators (**#**, **&** and **=**), three text functions (**SEARCH**, **LEN** and **MID**), an array function (**TRANSPOSE**), three dynamic array functions (**FILTER**, **UNIQUE** and **SORT**) and one non-auditable function (**INDIRECT**). I think it's my most comprehensive example yet – hence Best Excel Example. Ever. (Warning: may not live up to billing...)