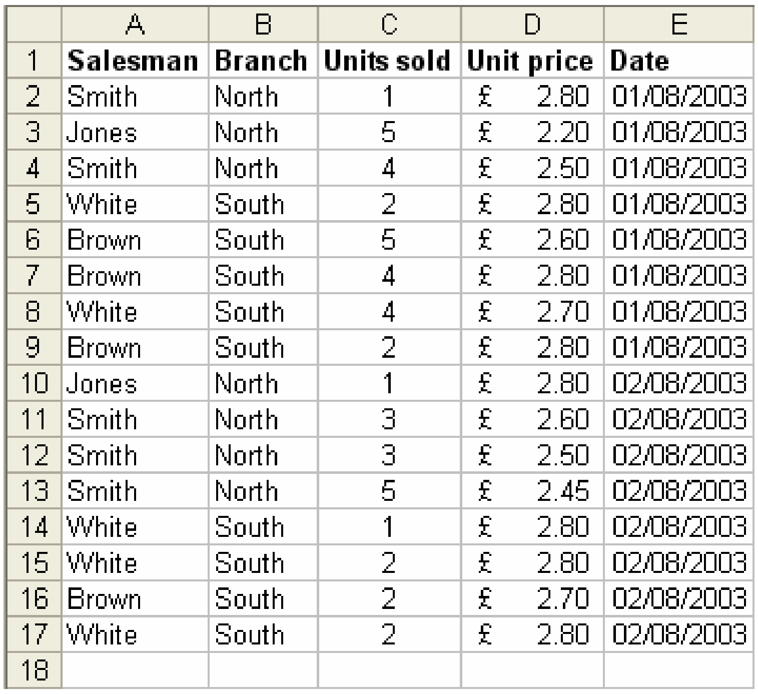
Beyond Excel's Holy Grail

Moving beyond a dependence on array formulae in Excel.

It's something of a rite of passage, really, the day when the intermediate Excel user discovers array formulae. If you haven't got to that stage just yet, stop reading now, but keep this article handy. When you get to the point (and you will) when you can't imagine life again without array formulae, take out this article and read it.

Array formulae are without doubt one of Excel's crown jewels. Few features in any programme can offer the kind of magic which is possible with arrays. Look at this data for instance:



	A	B	C	D	E
1	**Salesman**	**Branch**	**Units sold**	**Unit price**	**Date**
2	Smith	North	1	£ 2.80	01/08/2003
3	Jones	North	5	£ 2.20	01/08/2003
4	Smith	North	4	£ 2.50	01/08/2003
5	White	South	2	£ 2.80	01/08/2003
6	Brown	South	5	£ 2.60	01/08/2003
7	Brown	South	4	£ 2.80	01/08/2003
8	White	South	4	£ 2.70	01/08/2003
9	Brown	South	2	£ 2.80	01/08/2003
10	Jones	North	1	£ 2.80	02/08/2003
11	Smith	North	3	£ 2.60	02/08/2003
12	Smith	North	3	£ 2.50	02/08/2003
13	Smith	North	5	£ 2.45	02/08/2003
14	White	South	1	£ 2.80	02/08/2003
15	White	South	2	£ 2.80	02/08/2003
16	Brown	South	2	£ 2.70	02/08/2003
17	White	South	2	£ 2.80	02/08/2003
18					

We can, of course, total the columns. We can even use functions like **SUMIF** and **COUNTIF** to add up the data based on conditions, like the number of occasions four or more units were sold, or the total income on or after a particular date.

But as soon as we want to combine 2 or more conditions, these basic functions let us down, since they are limited to only a single condition.

So, if we want to know the total number of units sold by South Branch on 2 August, we can use:

```
{=SUM(IF(B2:B17="South",IF(E2:E17=DATEVALUE("02/08/03"),C2:C17,0)))}
```

At its heart, of course, this is a sum command. If we were to render the command into English, it might say something like this:

> *Add up where it says 'South' in column B and where the date in column E is 02/08/03 whatever is in the corresponding row of column C.*

Note the interesting framing of this function by the curly brackets. This is required of every array formula, but you can't just type them. After typing in the rest of the formula, you simply type [Ctrl]+[Shift]+Enter, rather than just Enter, as you usually would. Excel will add the curly brackets automatically. Also remember that whenever you edit the cell with an array formula in it, you should exit with the same 3-key combination.

The power of the array formula soon becomes apparent once you start playing with more fields and bigger, more complex lists. Some time ago, infatuated with arrays, I built a large spreadsheet to track results of 3,000 salespeople going through a number of assessments. I had array formulae to show how many people had passed how many assessments mapped against dates, assessment centres and case studies. In all, there were several hundred array formulae across a few worksheets in the one file. I was delighted with my achievement, and installed the file on the client's network.

As more and more data came in, though, it started taking longer and longer for the file to open, close or save. Before long, users were having to wait 4 minutes or more to get their statistics, and another 4 to close the file again. Clearly, this wasn't acceptable and I determined to find out what was happening and what I had done wrong.

The answer turned out to be nothing; I hadn't made any mistakes, and the sheet was working exactly as I'd designed it to. But I'd had my first taste of "array crawl". It seems that array formulae are highly volatile. In Excel terms this means that they will be forced to recalculate at the drop of a hat. Virtually anything you do in Excel will cause a chain reaction forcing all your arrays to recalculate, whether or not whatever you've done has any impact on them – and remember that each individual array formula could represent thousands of calculations. I was using far too many of them and they had to go.

But my sheet depended on them utterly. How could I do without them? The answer actually turned out to be quite easy. I knew about the Database functions – **DSUM**, **DCOUNT** and the like – but they'd never seemed terribly useful to me. I was about to be enlightened in a big way.

The basic syntax of a D formula (taking DSUM as an example) is:

```
DSUM(database,field,criteria)
```

‘**Database**’ is the range of cells which contains the data you want to query. It should contain a title row and look pretty much like the example we’re using, although it can be any number of columns wide, and any number of rows long. Often, you will want to define a dynamic range name for it.

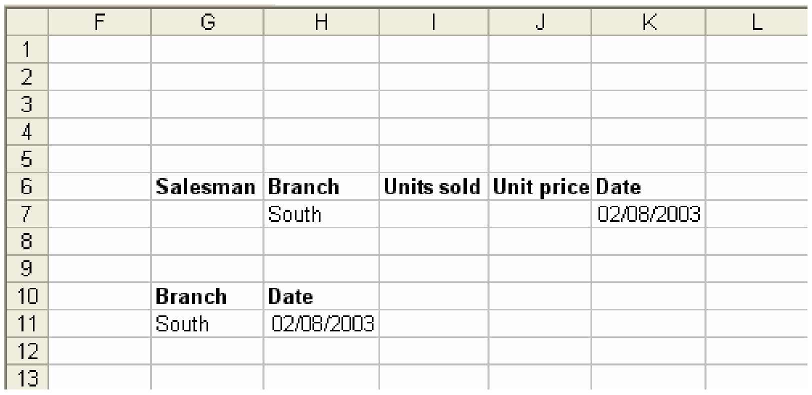
‘**Field**’ indicates which column is used in the function. For example, are you totalling units sold, or the unit price?

‘**Criteria**’ sets out the conditions you want to match, and here a little knowledge of advanced filters will pay dividends, because this works in exactly the same way. In order to prepare for the D formula to work, you effectively set up as you would for an advanced filter.

We’ll need an example to see this working, and let’s look back at the earlier example we used in the array formula:

> *Add up where it says ‘South’ in column B and where the date in column E is 02/08/03 whatever is in the corresponding row of column C.*

So let’s set up the filter. Look at the range in G6:K7 below.



	F	G	H	I	J	K	L
1							
2							
3							
4							
5							
6		**Salesman**	**Branch**	**Units sold**	**Unit price**	**Date**	
7			South			02/08/2003	
8							
9							
10		**Branch**	**Date**				
11		South	02/08/2003				
12							
13							

I started by copying and pasting the entire title row. This is a good idea, because any misspellings at all will make the D-functions fail. Then we can type under the headings any criteria we want for each field.

In our example, we just want to identify those sold by South on 02/08/03, so I’ve entered those details in the appropriate columns. This would do the trick quite nicely, but if you’re only using criteria in those two columns, you could simplify as in the range G10:H11.

We can now enter the D-function in the cell of our choice:

```
=DSUM(A1:E17,3,G6:K7)
```

Look at the three arguments in detail. The ‘database’ is A1:E17, which is the table containing all the data to query, including the column titles. At the end, the ‘criteria’ refers to G6:K7, where I’ve set up the advanced filter. Note that I could have shown G10:H11 using the simplified table for the same effect here.

The middle argument, 'field', is shown here as 3. We want to total the units sold, and this is the 3rd column in the database, so we show this as '3'. Personally, I find this difficult to debug, and counting columns every time is no fun. Fortunately, Excel allows you an alternative syntax in which you can refer to the field by its database column title, thus:

```
=DSUM(A1:E17,"Units sold",G6:K7)
```

In fact, if you name both the database range and the filter range, you can create D-functions which are virtually self-documenting, like this:

```
=DSUM(AllData, “Units sold”, Filter_South2August)
```

So I recoded my 4 minute sheet using D-functions. The sheet now has even more data than before and is open and usable within 5 seconds. In all respects it is just as good as the old one, but is now an entirely array-formula-free zone. I have a personal rule these days; if I ever find myself going beyond 10 array formulae in a spreadsheet workbook, I convert to D-functions before moving on.

This has been an introduction only to the D-functions, and how they can replace array formulae in your spreadsheets. I hope it will be enough to send you off to the Excel help file to check out the other D-functions. I am sure you will find as I did that these tools can exceed array formulae in terms of power and speed and that, like me, you won't look back.