First steps with Office Scripts: Solving a data challenge!

Owen Price
flexyourdata.com, @flexyourdata

Here’s a data challenge from Crispo Mwangi!

Students	Math	Eng	Phy	Geo	Chem
Daniel	77	53	87	46	91
Dennis	93	99	82	78	53
Joan	60	48	45	72	53
Ellen	46	62	80	82	47
Dominic	48	65	95	79	97
Jackie	91	53	47	66	70
Ray	63	72	50	66	56
Jade	48	53	45	59	90

At Least 2 Students

Marks above or equal	Subjects
90	Math ; Chem

Easy Sunday Excel Challenge

- **Lookup Subjects where at least 2 students scored 90 and above**
- **Dynamic array function allowed but Extra marks for Legacy Array Functions or PowerQuery Solution**

CrispExcel Training & Consulting

UNLOCKING YOUR DATA POWER

Owen Price
flexyourdata.com, @flexyourdata

And here’s one way to solve it with Office Scripts!

```
function main(workbook: ExcelScript.Workbook) {

  // Get an array of TableColumn objects for the subject columns
  const columns = workbook.getActiveWorksheet().getTable("Marks").getColumns().slice(1)

  // Get the data for the columns
  const data = columns.map(c => c.getRange().getValues())

  // Skip the header, the filter the numbers for > 89 and return true if more than one
  const mask = data.map(c => c.slice(1).map(s => parseInt(s.toString())).filter(v => v > 89).length > 1 )

  // Filter the data for where the mask is true, then slice for the header
  const headers = data.filter((_, i) => mask[i]).map(c => c.slice(0, 1))

// join and print the output
  console.log(headers.join(" ; "))
}
```

Owen Price
flexyourdata.com, @flexyourdata

Line by line – 1/2

When using an A1 address to pull some data into the script, you get an array of rows, so use a Table to get an array of columns

```
function main(workbook: ExcelScript.Workbook) {

  // Get an array of TableColumn objects for the subject columns
  const columns = workbook.getActiveWorksheet().getTable("Marks").getColumns().slice(1)

  // Get the data for the columns
  const data = columns.map(c => c.getRange().getValues())

...
```

'columns' is an array of TableColumn.

To get at the data, we need to refer to the range and get the values from that range.

Data is an array of arrays of (number | string | boolean). This is a 'mixed type', which makes the rest of the code more difficult ☹

Line by line – 2/2

The filter returns an array of those rows whose value is greater than 89. We check for length > 1 and get a true/false

c.slice(1) returns the values in the column from the 2nd row onward

map(s => parseInt(s.toString()))

Converts each mixed-type value to an integer

```
// Skip the header, the filter the numbers for > 89 and return true if more than one
const mask = data.map(c => c.slice(1).map(s => parseInt(s.toString())).filter(v => v > 89).length > 1 )

// Filter the data for where the mask is true, then slice for the header
const headers = data.filter((_, i) => mask[i]).map(c => c.slice(0, 1))

// join and print the output
console.log(headers.join(" ; "))
}
```

We then map a function to the list of subjects to return only the first value – the header

The second argument of the functino in filter is the index of the element being checked. By passing that index into the Boolean 'mask', we get only those columns with more than one high score.

Takeaways:

1. The worksheet.getRange() method returns an 2D array of rows of (string | number | boolean)
2. The Table.getColumns() method returns an array of columns
3. The Array.map and Array.filter methods work similarly to the MAP and FILTER functions
4. The Array.filter allows direct access to the array index – (element, index) => etc

Owen Price
flexyourdata.com, @flexyourdata