VBA Stock Analysis

Introduction

In this Module 2 challenge we are going to refactor the code to make the code more efficient. We will then check whether the refactoring has helped to decrease the run time of the VBA script.

Challenge Background

Steve just finished his finance degree and an excel savvy. His parents would like to invest in stocks and asked his expertise to take a decision. Here we have used Excel with VBA scripting to analyze the entire stock. We helped Steve to create a workbook “Analyze 2017 and 2018 Stock” and he is pretty much happy with that. Now, to do a little more research for his parents, he wants to expand the dataset to include the entire stock market over the last few years. Although the code works well for a dozen stocks, it might not work as well for thousands of stocks. And if it does, it may take a long time to execute.

Analysis

First download the challenge_starter_code.vbs and rename it to VBA_Challenge.vbs. Then create a Resource folder to hold the screenshots. Rename the green_stocks.xlsm to VBA_Challenge.xlsm then add the VBA_Challenge.vbs script to the Microsoft Visual Basic editor. Then start refactoring the code.

These are the changes.

- First add a ticker index
- Then create three arrays named tickerVolumes with Long data type, tickerEndingprices with Single data type, and tickerStartingprices with Single data type as well

```
tickerIndex = 0

'1b) Create three output arrays
Dim tickerVolumes(12) As Long
Dim tickerStartingPrices(12) As Single
Dim tickerEndingPrices(12) As Single
```

- Create a for loop to initialize the tickerVolumes to zero and then create a for loop that will loop over all the rows in the spreadsheet

```
For i = 0 To 11
    tickerVolumes(i) = 0
    tickerStartingPrices(i) = 0
    tickerEndingPrices(i) = 0
Next i
```

- Loop over all the rows in the spreadsheet and finding totalvolume, ticker starting price and ticker ending price for each stock

```
"2b) Loop over all the rows in the spreadsheet.
For i = 2 To RowCount

    '3a) Increase volume for current ticker
    If Cells(i, 1).Value = tickers(tickerIndex) Then

        tickerVolumes(tickerIndex) = tickerVolumes(tickerIndex) + Cells(i, 8).Value

    End If


    '3b) Check if the current row is the first row with the selected tickerIndex.
    'If  Then
    If Cells(i, 1).Value = tickers(tickerIndex) And Cells(i - 1, 1).Value <> tickers(tickerIndex) Then

            tickerStartingPrices(tickerIndex) = Cells(i, 6).Value

    End If
```

```
'3c) check if the current row is the last row with the selected ticker
 'If the next row,Äôs ticker doesn,Äôt match, increase the tickerIndex.
'If  Then
If Cells(i, 1).Value = tickers(tickerIndex) And Cells(i + 1, 1).Value <> tickers(tickerIndex) Then

    tickerEndingPrices(tickerIndex) = Cells(i, 6).Value

End If
```

- Then increase the Tickerindex by 1

```
        '3d Increase the tickerIndex.
        If Cells(i, 1).Value = tickers(tickerIndex) And Cells(i + 1, 1).Value <> tickers(tickerIndex) Then

        tickerIndex = tickerIndex + 1
```

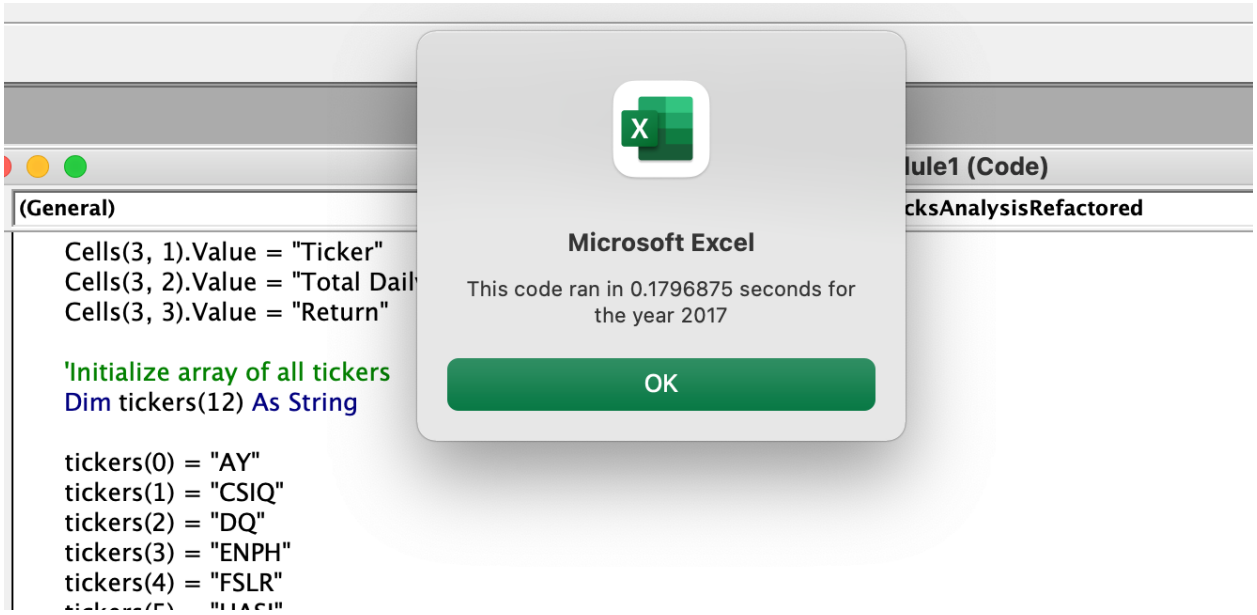
- Use a for loop to loop through the arrays (tickers, tickerVolumes, tickerStartingPrices, and tickerEndingPrices) to output the “Ticker,” “Total Daily Volume,” and “Return” columns in the spreadsheet

```
'4) Loop through your arrays to output the Ticker, Total Daily Volume, and Return.
For i = 0 To 11

    Worksheets("All Stocks Analysis").Activate

    Cells(4 + i, 1).Value = tickers(i)
    Cells(4 + i, 2).Value = tickerVolumes(i)
    Cells(4 + i, 3).Value = (tickerEndingPrices(i) / tickerStartingPrices(i)) - 1
Next i
```

- Finally, run the stock analysis, then confirm that the stock analysis outputs for 2017 and 2018 are the same as they were in the module
- Time taken to run the code when we input 2017

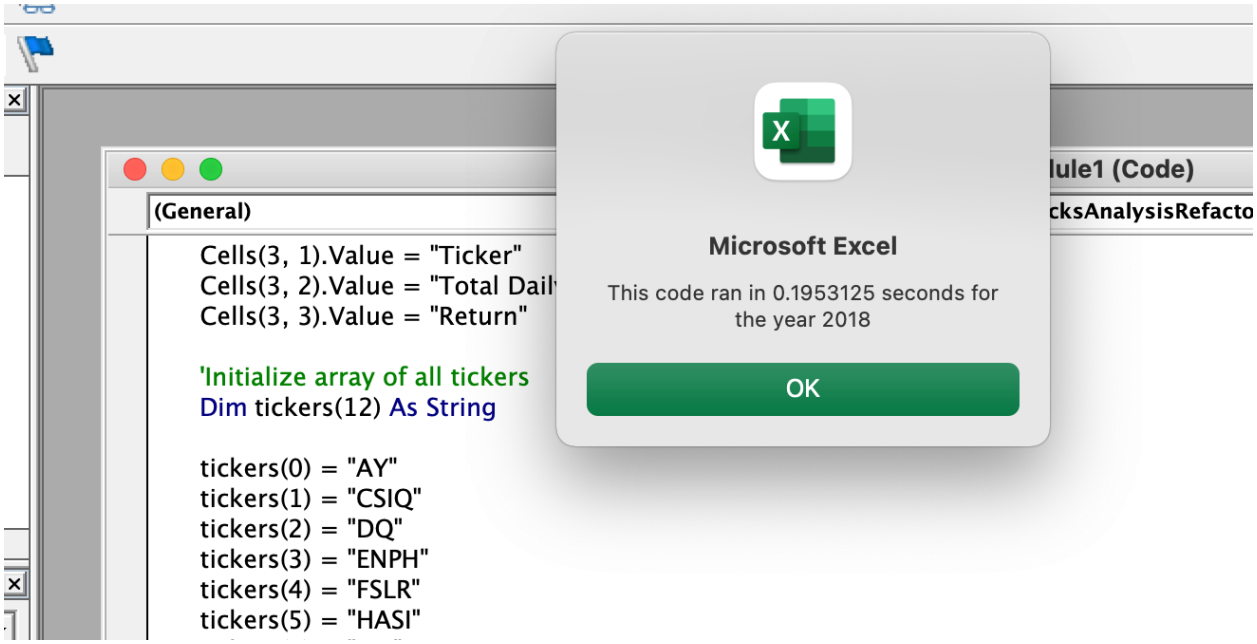


Result for 2017

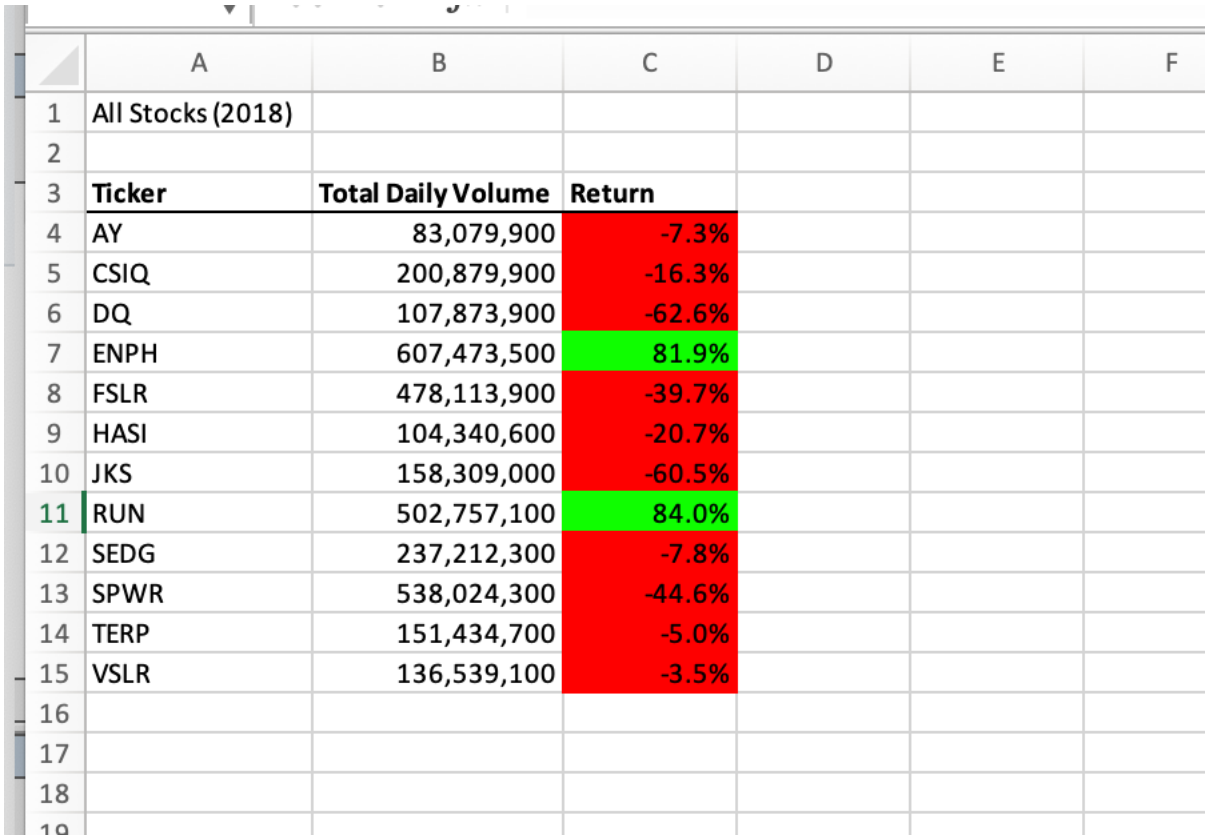
K11

	A	B	C	D	E	F	G
1	All Stocks ()						All Stocks
2							
3	**Ticker**	**Total Daily Volume**	**Return**				
4	AY	136,070,900	8.9%				
5	CSIQ	310,592,800	33.1%				
6	DQ	35,796,200	199.4%				
7	ENPH	221,772,100	129.5%				
8	FSLR	684,181,400	101.3%				
9	HASI	80,949,300	25.8%				
10	JKS	191,632,200	53.9%				
11	RUN	267,681,300	5.5%				
12	SEDG	206,885,200	184.5%				
13	SPWR	782,187,000	23.1%				
14	TERP	139,402,800	-7.2%				
15	VSLR	109,487,900	50.0%				
16							
17							
18							
19							
20							
21							

- Time taken to run the code when we input 2018



Result for 2018



	A	B	C	D	E	F
1	All Stocks (2018)					
2						
3	**Ticker**	**Total Daily Volume**	**Return**			
4	AY	83,079,900	-7.3%			
5	CSIQ	200,879,900	-16.3%			
6	DQ	107,873,900	-62.6%			
7	ENPH	607,473,500	81.9%			
8	FSLR	478,113,900	-39.7%			
9	HASI	104,340,600	-20.7%			
10	JKS	158,309,000	-60.5%			
11	RUN	502,757,100	84.0%			
12	SEDG	237,212,300	-7.8%			
13	SPWR	538,024,300	-44.6%			
14	TERP	151,434,700	-5.0%			
15	VSLR	136,539,100	-3.5%			
16						
17						
18						

From this we can conclude that after refactoring the code it took less time to run the code and the results are same as the “All stock analysis”.

Advantages

Refactoring is a key part of the coding process. When refactoring code, you aren’t adding new functionality, you just want to make the code more efficient by taking fewer steps, using less memory, or improving the logic of the code to make it easier for future users to read. Refactoring is common on the job because first attempts at code won’t always be the best way to accomplish a task. Sometimes, refactoring someone else’s code will be your entry point to working with the existing code at a job.
By refactoring our “All stock analysis” code, we were able to decrease the time for running the code.

Cons

Even if we are running the same code on the same spreadsheet for same year we are getting different running time.