# Analysis of Financial Stock Data via VBA programming and a Retro Factor of the Code.

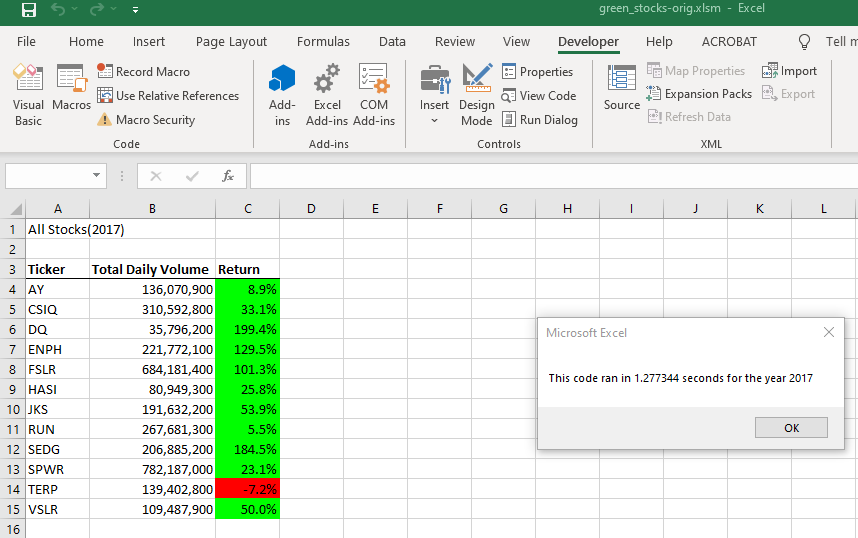
## Overview of Project:

Statistical data for 12 stocks has been collected over a period of two years (2017, 2018) on days that the stock market is open. The data consists of daily start price, end price and total volume of the respective stocks. The analysis was performed using VBA code and provides a one-line summary of each stock with its first stock market day of the year and last stock market day of the year and total volume. From there the Return can be derived. This analysis provides valuable input to an investor who has these stocks or is looking to include these stocks. The analysis is then further refined through a refactoring of the original VBA code to make it more efficient and more user friendly.

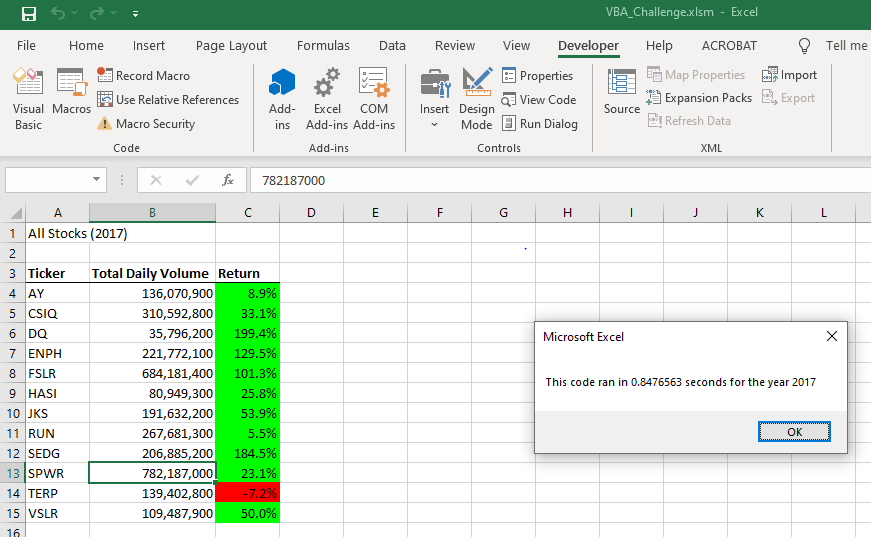
*Disclaimer*: This analysis is only provided for a statistical and analytical viewpoint. We request all users to perform their own research with due diligence prior to purchasing any of these stocks.

## Results

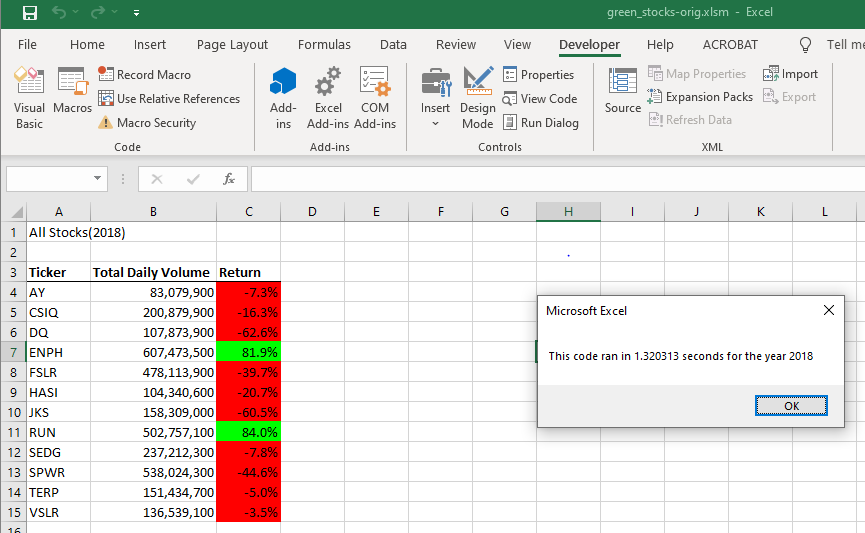
### Image for 2017 - below shows the original code



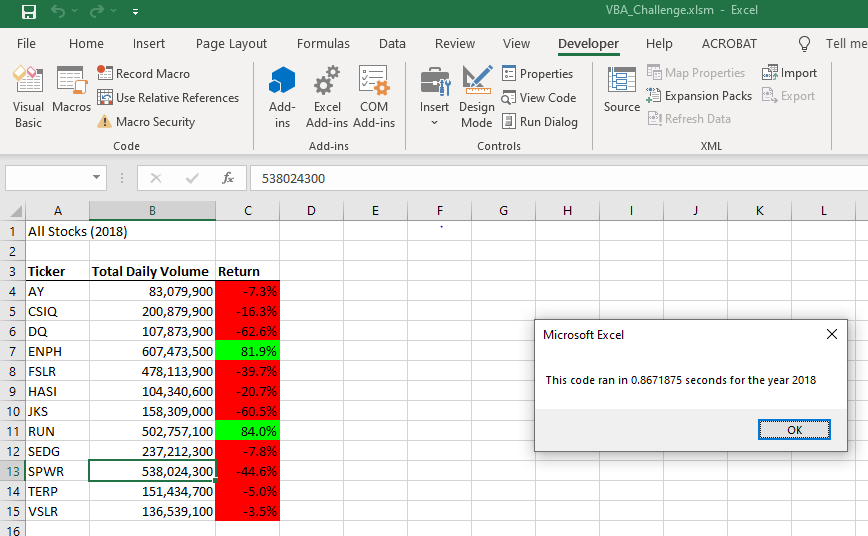
### Image below is from refactored code



### Image below for 2018 – below shows the original code



### Image below is for 2018 – below shows refactored code



### Comparing stock performance over the two-year period

## From an investor perspective investing in the stocks ENPH and RUN should be validated more closely as they show positive results over the two years of analysis.

## Code snippets for comparison – User input

The refactored code validates the user’s input for the years 2017 and 2018 only. If the user hits the Cancel button, the program exits gracefully.

User input is not validated in original code

**yearValue = InputBox("What year would you like to run the analysis on?")**

User input is validated in refactored code and workflow is managed using labels

**initialProcessing:**

**yearvalue = InputBox("What year would you like to run the analysis on?")**

**If (yearvalue = "2017" Or yearvalue = "2018") Then**

**GoTo startProcessing**

**ElseIf yearvalue = "" Then**

**MsgBox ("Exiting the application")**

**Exit Sub**

**Else**

**MsgBox ("you must type 2017 or 2018 to obtain the analysis. Please try again")**

**GoTo initialProcessing**

**End If**

**startProcessing:**

**startTime = Timer**

## Code snippets - Use of arrays for processing to coordinate a write operation rather than reading and writing line by line to the output Analysis sheet.

Writing into a file is an expensive operation. In the original code, one analyzes the data from thee 2017/2018 sheet and then write the result to the Analysis tab. So, the sheet is opened for writing 12 times (based on number of tickers)

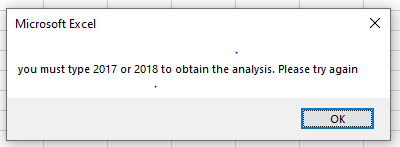
|  |
| --- |
| '6) Output data for current ticker  Worksheets("All Stocks Analysis").Activate  Cells(4 + i, 1).Value = ticker  Cells(4 + i, 2).Value = totalVolume  Cells(4 + i, 3).Value = endingPrice / startingPrice - 1 |

In the refactored code the data is collected in arrays and then written to the analysis sheet in one operation.

|  |
| --- |
| 4) Loop through your arrays to output the Ticker, Total Daily Volume, and Return.    Worksheets("All Stocks analysis").Activate  For i = 0 To 11  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 |

## Advantages or Disadvantages of Refactoring Code

Code is refactored primarily to obtain a positive result or outcome. In this case the code was refactored to provide a better user experience. This was a benefit and probably the outcome of feedback from usability testing. The UI should be friendly so as to not frustrate the user. In this case the user is guided by labels and warning if a date or value other than 2017 or 2018 is added in the refactored code.



From a coding and systems perspective, the number of writes to the file was decreased from 12 to 1. While this data set might not provide a discernible difference in execution times it will add up when there are large number of data sets. However, one can still see a difference in execution times.

1.27 v/s .84 (2017) and 1.32 v/s .86 (2018)

The disadvantage in using arrays is that they are stored in-memory so there is always an out of memory situation that can arise. One must consider available memory before refactoring code to use memory type code designs. Questions for analysis include - Can we use an array to chunk up all the values in memory and then write to the file? or should we chunk up a batch and then do it? Maybe process stocks that have an alphabetic pattern can be processed. For e.g. stocks A – D in one batch, etc.

Refactoring of code should at least attempt to

* Make the code more efficient – use of arrays and one time writes
* Make it more user friendly – guide the user to inputting correct values