# **Challenge 2: Stock Analysis in VBA**

# Overview of Project

# I: Purpose

This analysis’s purpose was to assist Steven, a recent finance graduate, in making an informed decision about investing his client’s money. In this challenge, VBA code was already created and our task was to edit, or refactor, the dataset with VBA solution code to loop through the entire set of data one time to collect information regarding the stocks which will assist in Steve evaluating the stocks rate of success. After refactoring the data, we can determine whether the code we edited assisted in making the VBA script run more efficiently by condensing our code into fewer lines and therefore improving the code’s logic to make it easier for future users to read and use.

# Analysis and Results

# I: Examples of Code Used

For this section, I have included the criteria for the challenge as well as screenshots of the code used with comments alongside them to provide simple explanations.

1. Created a tickerIndex variable and set it equal to zero before iterating over all the rows. Will use this tickerIndex to access the correct index across the four different arrays on VBA Code: the tickers array and the three output arrays created on next requirement.

A picture containing text

Description automatically generated

1. Created three output arrays, tickerVolumes, tickerStartingPrices, and tickerEndingPrices.

Created three output arrays: tickerVolumes, tickerStartingPrices, and tickerEndingPrices. The tickerVolumes array should be a long data type. But in our VBA code the tickerStartingPrices and tickerEndingPrices arrays should be a Single data type.

A screenshot of a computer

Description automatically generated with medium confidence

1. Created a loop to initialize tickerVolumes to 0. If the next row ticker does not match, then it will increase the volume for the current ticker.

Graphical user interface, text, application

Description automatically generated

1. Created a loop that will loop over all the rows in the spreadsheet. Inside the loop, we created a script that increases the current stock tickerVolumes variable and adds the ticker volume for the current stock ticker. Created and if-then statement to check if the current row is the first row with the tickerIndex selected. If it is, then the current closing price is assigned to the tickerStartingPrices and tickerEndingPrices Variable.

Graphical user interface, text, application

Description automatically generated

1. Created code that will format the spreadsheet making positive returns green and negative returns red, to be a lot easier to determine which stocks were successful and which were not.

Text

Description automatically generated

# II: Dataset Analysis

After dissecting the code, now have a better understanding of the data we are about to analyze and how it will be formatted. Our stock analysis outputs for 2017 and 2018 are correct when compared to the previous examples shown in the challenge. The code did run faster than before but not by a significant amount. However, our code is more legible than before and easier to understand as well as edit for future use.

Results and Time for VBA\_Challenge\_2017.PNG

Graphical user interface, application, table, Excel

Description automatically generated

Results and Time for VBA\_Challenge\_2018.PNG

Graphical user interface, application, table, Excel

Description automatically generated

# Summary

# I: Pros and Cons of Refactoring

In sum, refactoring data is a very useful tool to a data analyst by making small adjustments to code you already have access to. Each adjustment you make to your code makes your code slightly more efficient but more importantly more legible and does not bar access to someone who may not understanding coding. Another great feature one can accomplish while refactoring is using the comments feature to show your train of thought when working in VBA or any other coding languages.

Pros: -Addressing errors are easier as they appear in well written code that contains nested conditionals and loops.

-In our case, using Excel flow displays program logic in a more comprehensible manner, not tied to the order that the underlying code is written.

-VBA interpretation (Excel) of code can reveal patterns that are not easy to see in the source.

-Refactoring gives opportunity to use the comments feature to make code more legible

Cons: -A long procedure may contain the same line of code in several locations, you can change the logic to eliminate the duplicate lines.

-A logical structure may be duplicated in two or more procedures (possibly via copy & paste coding). When detected, this logic is best moved to a new function and called from the other functions.

-A complex unstructured code is usually best to split in several functions.

-Refactoring process can affect the testing outcomes.

# 2. Pros and Cons when refactoring the original VBA script?

The most important thing to keep in mind when refactoring code is that you are not seeking to change what your code is seeking to accomplish. Your code maintains the same functions that it did before but condensing and rewriting pieces of it can make things easier for the user in the long run. It is easy to see when refactoring is common practice because the reuse of code (I assume) is common. Going back to a code you have not worked with for some time may be confusing if it is not simple enough to understand. This is where refactoring and adding comments can be a major factor in helping an analyst complete their task(s) efficiently and without confusion. Keeping up with this kind of maintenance is however a large undertaking and this challenge was difficult for our first coding challenge. Practice is key to refactoring, and I believe it will get easier as we evaluate more code.