**Module 2 Challenge: VBA of WallStreet**

***Overview***

The purpose of this report is to utilize excel and VBA to assist Steve with his stock market endeavor with DQ and all stocks analysis. The refactoring method was introduced in order to see if the yearly data loop was faster with refactoring the coded data. Conditional formatting was implemented to clearly display the stock performance for both 2017 and 2018 and run buttons were placed, so that Steve could check and clear the stocks information easily on a regular basis. The loops were timed for both the original 2018 data and the refactored time and a comparison was made to see which one was faster1,2.

***Results***

Using code to obtain the figures 1 through four, a comparison of the 2017 and 2018 stock analysis could be viewed, as could the times acquired to run the code. The initial steps required were to implement the VBA in excel to enable the ability to create macros. A DQ analysis was completed for 2018. Tickers were created for 2017 (figure 1) and 2018 (figure 2) in order to allow Steve to visualize the stock performance easily. Static and conditional formatting were implemented to visualize the positive and negative stocks and run buttons were placed so that Steve could rerun the looped code and clear sheet for easy, repeated analysis.

***Graphical user interface, application, table, Excel

Description automatically generated***

Figure 1: 2017 stock results

When comparing the 2 figures, it can easily be ascertained that Steve’s stocks of interest were not performing as well in 2018 as they were in 2017, as there were more stocks ‘in the red’.

***Graphical user interface, application, table, Excel

Description automatically generated***

Figure 2: Stock results for 2018

After this analysis was completed, a refactored code was created to compare the times that the data would run. In the original code, the run time was 0.3242188 seconds (Figure 3) and the refactored code ran in 0.1138125 seconds (Figure 4). This demonstrates a more efficient code run time after refactoring.

***Graphical user interface, application

Description automatically generated***

Figure 3: original time for loop 2018

***Graphical user interface, application

Description automatically generated***

Figure 4: Refactored time for loop 2018

***Summary***

Some of the advantages of refactoring code are that it provides the opportunity to clean up code in order to update and improve the data output. It can also perform a function faster and more efficiently, particularly in reference to Steve and his stock endeavors. This allows for easy readability and comprehension.

Some of the disadvantages of refactoring code generally, especially if one is a beginner at coding, is that there is a lot of room to change code and introduce errors, as in this case. The original challenge had to be reconstructed and pasted together as VBA\_Challenge2, due to a possible ‘never ending for loop’ that kept freezing, requiring multiple computer force-quits. This became extremely time consuming and frustrating.

The advantages of the original VBA script is that it provided Steve the data in a concise and easy to read format that, due to looping, could be checked and compared on a regular basis. Refactoring could improve the run times of the code for more efficiency.

The disadvantages are that refactoring can send the creator into their own loop, where it becomes unclear where the errors occurred, making it very time consuming and confusing. Therefore, unfortunately in relation to this challenge, by the time this assignment was completed, with errors, poor Steve and his parents lost all their money in the stockmarket and I was out of a job.

***References***

1. DataBootcamp. <https://courses.bootcampspot.com/courses/967/assignments/19046?module_item_id=343851> . Accessed oct 21, 2021
2. StackOverflow. <https://stackoverflow.com/questions/18088729/row-count-where-> data- exists .Accessed Nov 7, 2021