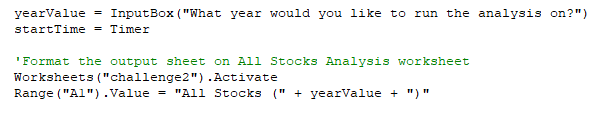
**Stock Analysis Refactor and Applicable Benefits**

In order to improve upon the performance for our client, Steve, code refactoring was done to improve the initialization and performance output for Steve to review his desired stocks. Enhancements, not only to code were performed as we wanted to improve the customer experience Steve had by also including additional buttons to run and clear his macros for quick utilization.

In the initial code base, the client was limited to data on his excel worksheet, but improvements were made to allow the yearly input selector to isolate and run desired output by designated stocks through various different years, such as 2017 and 2018. This allows for Steve to add more data into his worksheet for iterative years such as 2019, 2020, 2021, and beyond and still be able to repurpose without contracting additional work to re-write the internal VBA code to activate particular worksheets (Figure 1). In Figure 1, the line of code adds the input to reference the specific yearly sheet, as well as changes the header dynamically so that a hard code input of a specific year is shown as the Range(“A1”) value.



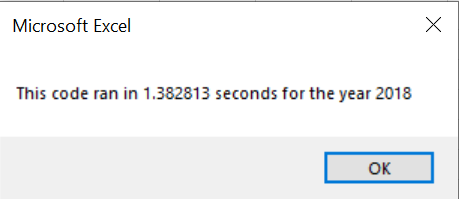
**Figure 1**

Aside from being more dynamic and versatile, the code refactor allows Steve to run the lines of code quickly as designated stocks are set through arrays, with ability to quickly iterate with minimal variables using ticker index a working variable within itself. This removes the need for additional nested functions with multiple I and j variables for code to be read in a more organized fashion. For example, in Figure 2 and 3, using the original and refactored code side by side for 2018; the original code takes approximately 1.38 seconds to complete, whereas the refactored code completes in 0.57 seconds, which is 2.42x quicker.

Graphical user interface, application, table

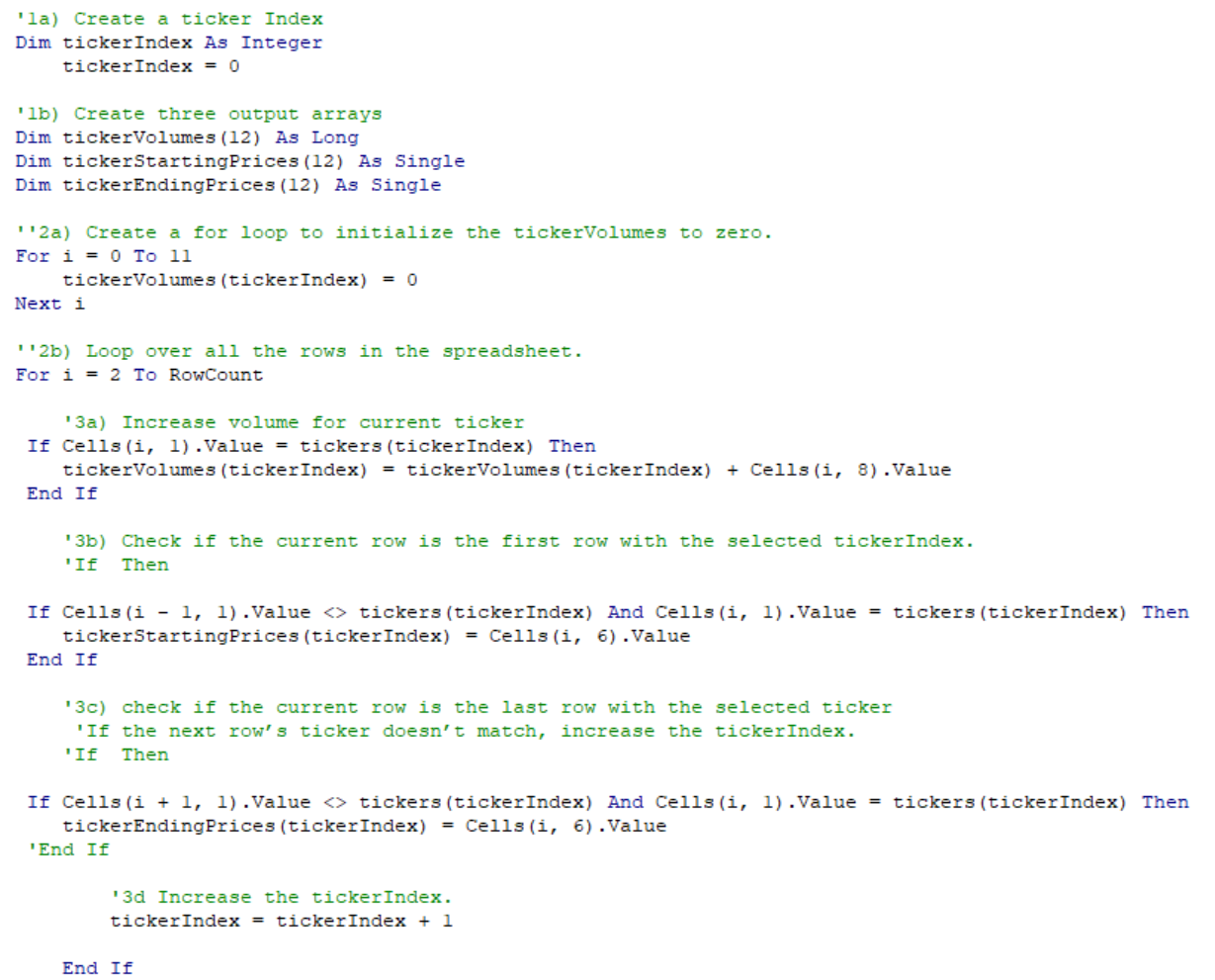
Description automatically generated

**Figure 2**



**Figure 3**

The key components of the efficiency from a backend view, organizes the code base through iterations lies with using ‘tickerIndex’ as the iterator value which removed the nested functions and references the index through iteration ‘I’ across the array rows and from row 2 to row end (Figure 4). This removed additional layers and pre-sets and allows the data to go through the data sets more quickly.



**Figure 4**

Regarding the advantages and disadvantages of refactoring code, through this exercise for client, Steve, an assessment was able to be made. The advantages were to future proof this macro for the client in order for the client to be able to intuitively use for years to come. Although speed of 1.38 seconds may be minimal, once the client parses more rows of code in the hundreds of thousands or more, this will take more time for the client to receive the output as well as more strain on his operating system to complete. By improving, this minimizes the speed in accordance with scaling factor. In my opinion, there were no disadvantages of refactoring the existing code. Although this worked for the client, this would eventually be outdated, and as technology changes, so does the need of the client, and makes sense to improve upon the macro product that was provided.