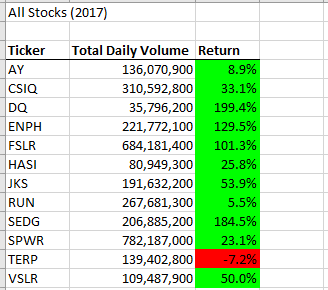
**All Stock Analysis**

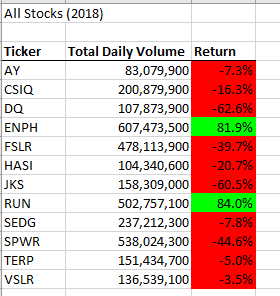
**Purpose:**

Analyze stock performances of a given set of stocks using most efficient coding process. Develop a refactored code that can run through thousands of stocks. The analysis will show how active a stock is by showing how often a stock is traded by analyzing the Total Daily Volume. The analysis will also show the yearly return of each stock by looking at its starting and ending prices for the year.

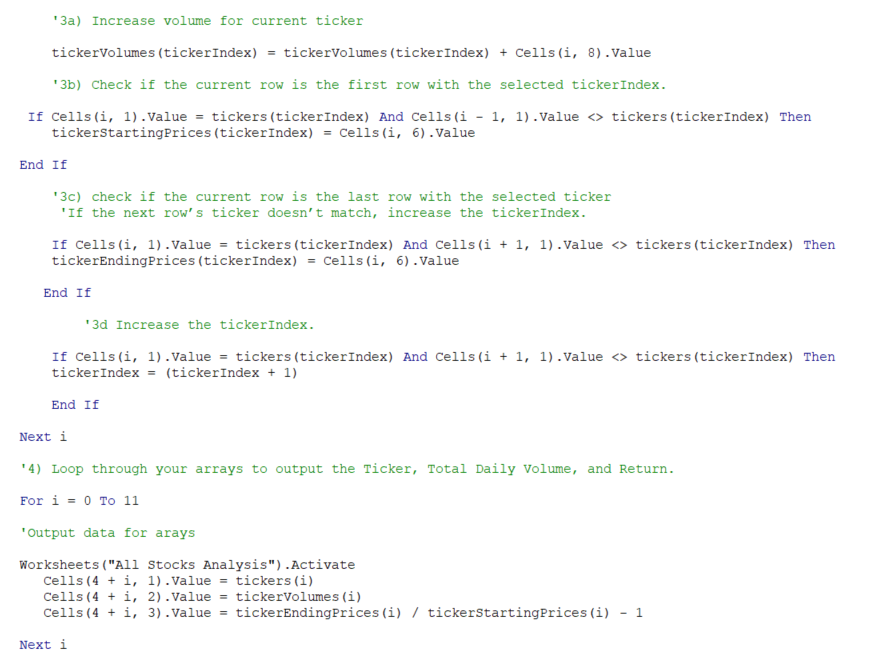
**Results Refactored Code:**

Overall the stocks had a better return in 2017 than in 2018. Except for ‘TERP’, all of the 2017 stocks had a positive return, which you can see below. In 2018 the only stocks to have a positive return was ‘ENPH’ and ‘RUN”. This analysis shows then that the two best stocks to invest in were ‘ENPH’ and ‘RUN’ because they had growth both years.



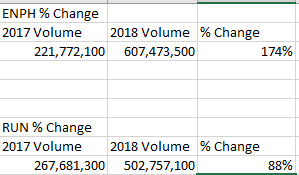


The above tables were created in part with the following code. The code was designed to loop through the 2017 and 2018 worksheets, pull and add the daily volumes for each ticker. The second to last line of code shows how the ‘Return’ percentage was calculated.

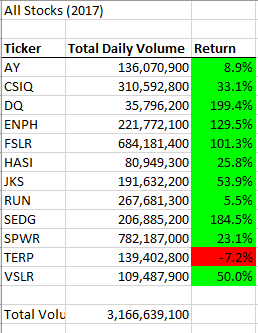


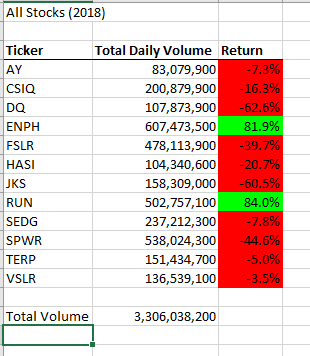
‘ENPH’ also has a 174% year over year percent increase in daily volume of trades.

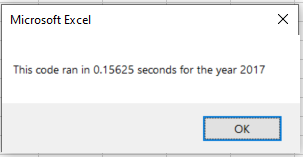
‘RUN’ had a 88% year over year percent increase in daily volume of trades.

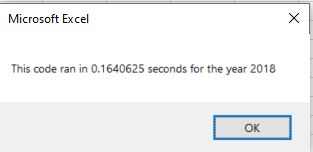


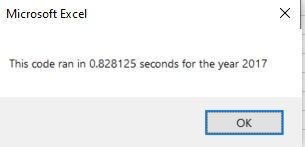
You can see below 2018 had more trade volume overall for all the analyzed stocks, it still had a worse return of stock price than 2017.

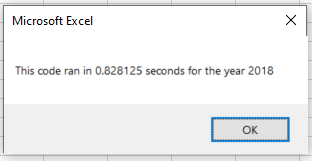












**Execution Times**

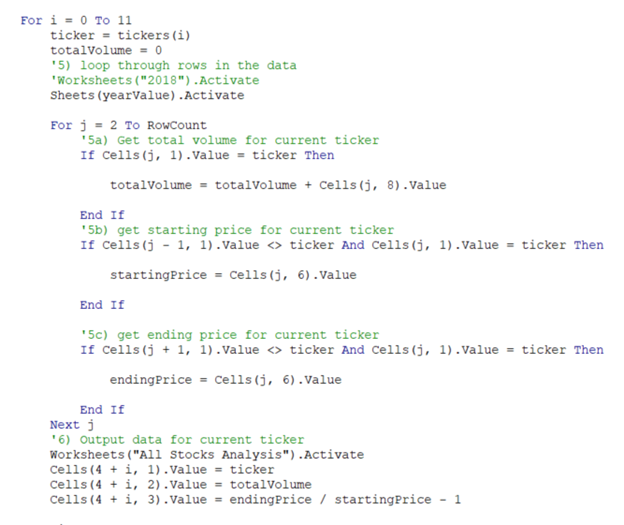
The refactored times for 2017 and 2018 were faster than the times for the original code.

**Summary**

The benefit of refactoring the code is not only will the code run faster, but it is easier to read and therefore easier to update in the future. The refactored code allows for a better understanding of the function of each line of code. However code refactoring can be time consuming, as you are trying to figure out a new way to perform the same function as the original code. Also, refactoring code can take multiple trials in order to test each change to the code.

Figuring out the refactored loop did take a significant amount of time longer than writing the double loop. Refactoring the original code decreased the run time and thus increased the efficiency. This efficiency was attained but removing the double loop and refactoring the code for a single loop. Please see below.

**Original Loop**



**Refactored Loop**

