1. **Overview of Project**

Steve has requested an analysis of stock information to help inform his stock decisions for his clients. By analyzing the daily volume and yearly return percentage for stocks in the energy sector, he can make informed decisions on what the average company’s performance looks like and how his clients preferred stocks (in this case, DQ) perform against others in the sector. Because this is a large quantity of data and Steve wants to be able to analyze different years at a time, the easiest solution is to build a script that can create a summary of this data for any given year.

1. **Results: 2017 vs 2018**
   * + 1. ***Stock Performance***

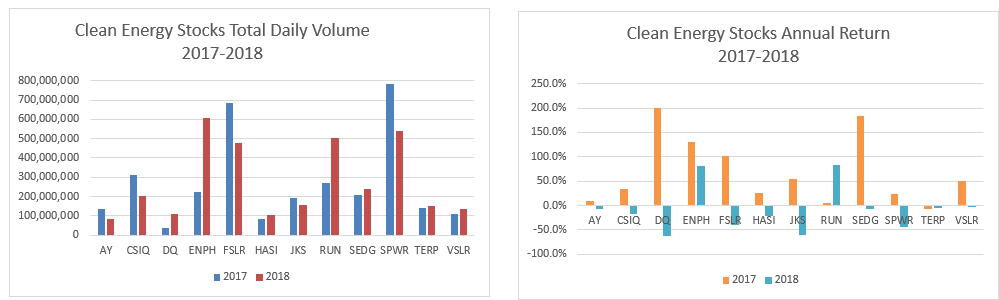


Figure 1: Source: https://github.com/Roland791/VBA\_Challenge/blob/main/Resources/Performance%20Charts%202017-2018.PNG

When looking at the overall data from year to year, we can graph the performance of all stocks. As demonstrated in Figure 1, DQ, which is the client’s preferred stock, had a strong year in 2017 but their return rate went from the highest of the group (at almost 200%) to the lowest of the group (over -50%) in 2018. And while a general drop of performance holds true for most of the companies analyzed, the degree of change is significantly higher for DQ. ENPH, however, managed to keep a consistently high Return and was one of the few companies that showed a significant jump in Daily volume between 2017 and 2018 as well.

* + - 1. Scripting Performance

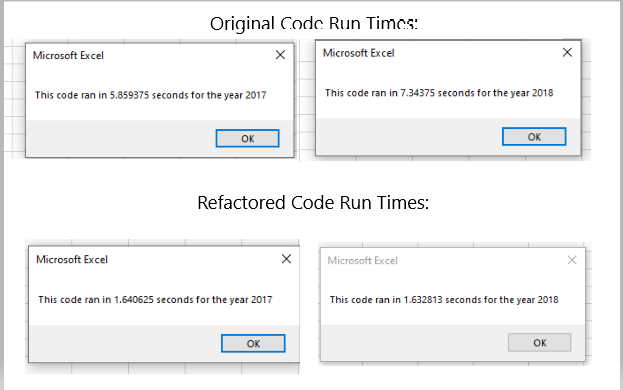


Figure : Source:https://github.com/Roland791/VBA\_Challenge/blob/main/Resources/Summary%20Comparison%20-%20Code%20Run%20times.png

When upgrading the script to analyze the full set of data for each year, rather than just the DQ stock for a specific year, a certain degree of refactoring took place that served to optimize the overall process. As noted in Figure 2, the final runtimes for the script were reduced significantly, (from 6-7 seconds to less than 2). This reduction would be even more noticeable as the size of the datasets increased.

1. **Summary:**

By refactoring existing code, we can save time by not having to write the code and figure out the logic from scratch. It also gives us the opportunity to optimize the code and find ways to improve the overall performance, possibly even making it more robust and flexible for use in additional scenarios in the future. The drawback to this is that if you are using code that you did not your self create, or code that is not well organized, you can spend more time troubleshooting and trying to work through the logic of the code, leaving less time for higher level analysis and improvements.

In the case of this particular script, when writing the original DQ code, I did not spend sufficient time writing notes on what specific sections did, so when refactoring the first time, I and to go back and rework through the logic. I learned from this during the second iteration though, and made sure to include the notes regarding what each section was doing and it made the final iteration much easier to work with.