# **# Overview of Project:**

**## Background:**

Steve wants us wants to expand the dataset to include the entire stock market over the last few years. Although your code works well for a dozen stocks, he fears it might not work as well for thousands of stocks. And if it does, it may take a long time to execute.

**## Purpose:**

Refactor the code so that it can accommodate thousands of stocks. The new code should not only be capable of analysing thousands of stocks but also give the best investment recommendations to his parents in a few seconds or less.

# **# Results:**

**## Performance of the stocks**

When we compare the stocks for the years 2017 and 2018, we can see a stark difference in returns. While 2017 is green all the way except for one red; 2018 is a total contrast with red all the way except for two tickers.

**### The Year 2017**

All stocks except one gave positive returns *indicated by green* for the year 2017. In fact, *four of the 12 stocks gave over two times* the returns. ***DQ was the best performing ticker*** giving 199% returns, closely followed by SEDG giving a return of 184.5%

Table

Description automatically generated

**### The Year 2018**

This wasn’t a great year for stocks, as highlighted in red the chart above. All except RUN and ENPH had negative returns. DQ was the worst performing stock in 2018 with negative returns of 62.6%

**### Both years**

Looking at both the years we can conclude that TERP was the most stable in both years. ENPH performed well and gave positive returns of 129% and 82% in the years 2017 and 2018. Looking at the positive returns given by ENPH Steve’s parents may definitely want to place their bets on that one!

**## Execution times**

Comparing the execution times of the original script and the refactored script

**### For the year 2017.** The original code took 0.32 seconds to run while the. Refactored code ran in 0.08 seconds.

Graphical user interface, text, application, chat or text message

Description automatically generated Graphical user interface, application

Description automatically generated

**### For the year 2018.** Similar to the previous year 2017, the original code took 0.32 seconds to run while the Refactored code ran in 0.08 seconds.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, application

Description automatically generated

# **# Summary:**

## **Some of the advantages of refactoring code**

1. ##### Refactoring Improves the Design of Software by making it more organized and cleaner
2. ##### Refactoring Makes Software Easier to Understand since it becomes more logical
3. ##### Refactoring Helps Finding Bugs
4. ##### Refactoring Helps make the program run faster as can be seen in the above pop-up examples between the original code and the refactored code.

**## Some disadvantages of refactoring code**

* 1. ### It is more time-consuming and therefore costs more
  2. ### It could sometimes introduce new bugs to the code which might require more time for debugging

**## How do these pros and cons apply to refactoring the original VBA script?**

When we look at the final output of the refactored it is undoubtedly more efficient. What took 0.3 seconds can now be run in 0.08 seconds. However, one cannot overlook the high risk that refactoring bears. What if in the bargain some new bugs were introduced? What if some important code got deleted in the process of refactoring? It’s better to save a copy of the original code which can be used as a reference just in case of these uncertainties.

Refactoring also needed additional time investment. If we have a time-bound commitment then the additional time to refactor a code definitely must be accounted for. When I was refactoring, I ran into a couple of errors. Debugging these errors also took me some additional time.

In conclusion, the benefits of refactoring including making a more logical code, that is clear, more organized and runs efficiently should be weighed against the additional time needed to refactor the code.