**Overview of Project**

Steve works in finance industry and helping his parents to analyze the stock market particularly in green industry (DQ) but he wants to give his parents a complete picture of the stock market so he can give good recommendation.

**Results**

* In 2017 only “TERP” has negative return which is -7.2%
* In 2018 only 2 tickers have positive return. ENPH has 81.9% and RUN has 84.0% return
* There is no co-relation between total daily volume with the return
* Before refactoring 2017, ran in 1.345703 seconds and after refactoring ran in 0.1015625

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

* Before refactoring 2018, ran in 0.6679688 seconds and after refactoring ran in 0.1015625 seconds

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Summary**

Refactoring Advantages

* Good tool to maintain the code
* Make the code clean and organized

Refactoring Disadvantages

It’s risky when:

* The application is big
* The existing code doesn’t have proper test cases
* The developers do not understand what is all about

Refactoring Pros \*)

* *Chances of Enhancement are high*If modules have chances to add new features or functionalities then make sure design and current code is good and following Open Close Principle
* *Code Smell is Detected*Sometimes bad patterns like tight coupling, duplicate code, long methods, large classes, etc. are detected in the code;  the code should be refactored in this case.
* *Bug Fixing*Codes are written badly in some cases, and so many bugs are raised. In this case, fixing of bugs take too much effort. So, the root cause of bugs can be code smell. So, before fixing bugs code should be refactored.
* *Peer Review*Peer review is an important part of code refactoring. If the peer-reviewer finds some code smell then code should be refactored during peer review.

Refactoring Cons \*)

* Delivery Deadline is near and new development is planned.
* The cost of refactoring is higher than rewriting the code from scratch.
* Don't refactor the code if you don't have the time to test the refactored code before release. It can introduce bugs.
* Don't do delayed refactoring because it contains a big mess and makes it very difficult to refactor. Do early refactoring.
* Stable code should not be refactored.

\*) https://www.c-sharpcorner.com/article/pros-and-cons-of-code-refactoring/