Written Report – NGO Nicolas

**Overview**

The script wants to analyze real-world stock data by combining Python programming expertise with financial knowledge.

**Python Implementation**

Libraries Used: The code extracts financial data using « yfinance », manipulates data using `pandas` and `numpy`, visualizes data using `matplotlib.pyplot} and `seaborn}, and performs statistical analysis using `scipy.stats}.

Environment Setup: The script assumes that the libraries listed above are available, but it is not required to create a specific environment.

**Data**

Data Source: The ‘yfinance’ library is used by the script to retrieve data from Yahoo Finance. It complies with the need to use data that is publicly available.

Stocks Analyzed: AAPL, GOOGL, AMZN, MSFT, TSLA, JPM, GS, IBM, NFLX, and CSCO are among the stocks that are included in the analysis.

Time Frame: Data from January 1, 2020, to January 1, 2023.

**Analysis Done**

1. Stock Prices Analysis: The script plots the chosen stocks' adjusted close prices over the given period of time.
2. Rolling Volatility: Each stock's 20-day rolling volatility is computed and plotted.
3. Returns Analysis: To analyze returns, the percentage change in stock prices is computed.
4. Statistical Measures:
   * a. The computation of mean returns.
   * b. Plotting stock return covariance and correlation matrices sheds light on the connections between various stocks.
5. CAPM Analysis:
   * c. To determine each stock's market risk, beta values are computed in relation to the S&P 500 index.
   * d. The Capital Asset Pricing Model (CAPM) is used to calculate and display expected returns.
6. Security Market Line (SML): The script plots the Security Market Line, providing a graphical representation of the CAPM, showing the relationship between expected return and beta.

**Model, Results and Structure**

1. - Capital Asset Pricing Model (CAPM): The model and its outcomes

* Model: The Capital Asset Pricing Model (CAPM) is used to calculate an asset's expected return based on its risk relative to the market. Utilizing the equation Expected Return=Risk-Free Rate+β×(Market Return−Risk-Free Rate) results in the expected return.
* Outcomes: The code determines the chosen stocks' beta values, which express their volatility in relation to the market. Next, it calculates these stocks' expected returns using the CAPM.
* Implications: Investors need to know these calculations in order to comprehend the risk-reward characteristics of stocks. Greater volatility and possibly higher returns, but at a higher risk, are indicated by higher beta values.

2. Rolling Volatility:

* Model: The short-term volatility of stock prices is evaluated using the 20-day rolling standard deviation.
* Outcomes: This rolling volatility is plotted by the script for every stock, giving investors a dynamic perspective of how prices change over time.
* Implications: In order for traders and investors to comprehend the short-term risk attached to the stock, this analysis is essential.

3. Examination of Statistics:

* Model: The script computes and displays the stock return covariance and correlation matrices.
* Findings: It illustrates how stock returns fluctuate in relation to one another, revealing the degree of diversification that can be attained in a portfolio made up of these stocks.
* Implications: Diversification is a key strategy in portfolio management, where minimizing risk is achieved through it.

1 - Libraries for the Python Project Structure:

* Yahoo Finance is the source from which historical stock data is retrieved. It's an easy and efficient method of getting thorough stock market data.
* pandas: Uses DataFrame structures to handle and manipulate data effectively. It's essential to using Python to handle big datasets.
* Numpy: Offers assistance with intricate mathematical operations on arrays. It is applied to financial statistics computation in an efficient manner.
* Seaborn and matplotlib.pyplot: These libraries are for visualizing data. For creating charts, Matplotlib is a flexible tool, whereas Seaborn is utilized for complex statistical plots.
* **scipy.stats**: Used for more complex statistical calculations.

2. Code Structure:

* The script is set up to import the required libraries first, then use pandas and yfinance to fetch and preprocess data.
* After that, it moves into different analysis sections, each of which focuses on a different kind of financial analysis, such as statistical analysis, CAPM, volatility analysis, and stock price visualization.
* For financial analysis projects to be clear and maintainable, a logical flow from data preparation and acquisition to in-depth analysis and visualization is ensured by this structure.

**Conclusion**

The script effectively demonstrates the application of Python in financial analysis, covering a range of analyses from basic stock price visualization to advanced concepts like CAPM and SML. The report should focus on interpreting these analyses in the context of financial theory and real-world application.