Informatics Institute of Technology

6BUIS017C

Student ID – 20220274

Name – Kaveendra Walalawela

Module Leader – Fouzul Hassan

Date – 11/25/2025

GitHub Link-https://github.com/kaveendrawalalawela/CRM-CW-.git

Table of Contents

01.Introduction

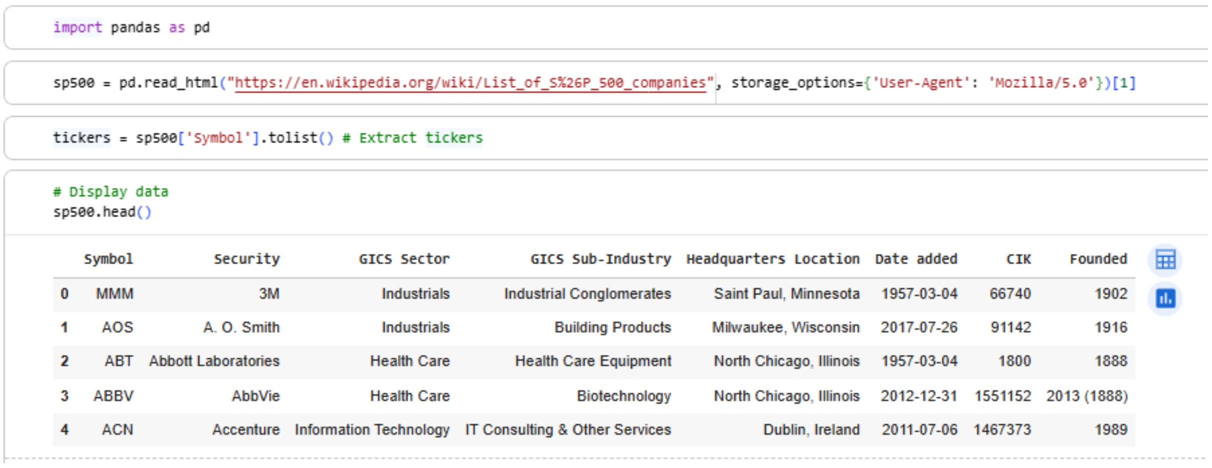
The stock market is inherently volatile and influenced by multiple factors, making it essential for investors to understand how different stocks behave under varying market conditions. To support informed investment decisions, this study analyses the behaviour of S&P 500 stocks using a data-driven approach. Historical price data from 2022 to 2025 was collected through the yfinance library, cleaned, and transformed to compute three key financial metrics: Daily Returns, Beta, and Annual Volatility. These metrics capture short-term fluctuations, market sensitivity, and long-term risk, providing a comprehensive view of stock performance.

Using these metrics, the study applies two clustering techniques Agglomerative Clustering and K-Means to segment stocks into meaningful groups based on their risk and behavioural characteristics. Agglomerative clustering helps identify natural groupings based on Beta alone, while K-Means clustering considers both Beta and Annual Volatility to form multi-dimensional investor-oriented clusters. These clusters reveal patterns such as low-risk defensive stocks, moderate-risk core holdings, and high-risk aggressive stocks.

The ultimate goal of this analysis is to extract practical business insights for investors, enabling them to build a well-diversified portfolio aligned with their risk appetite. By combining statistical calculations with machine learning techniques, this report provides a structured and data-supported method for understanding stock behaviour and supporting better investment decisions**.**

Q01-Extracting S&P 500 Tickers from Wikipedia

I used pandas.read_html() to take the S&P 500 companies' table directly from the Wikipedia page. The reason I chose this route is that it is the easiest and most reliable way to get all the tickers at once without having to type anything manually. Since the table is already organized on the site, read_html() automatically converts it into a DataFrame; from there, I needed only the Symbol column to have my list of tickers. I also wanted to check whether correct data was loaded using sp500.head().



```python
import pandas as pd
```

```python
sp500 = pd.read_html("https://en.wikipedia.org/wiki/List_of_S%26P_500_companies", storage_options={'User-Agent': 'Mozilla/5.0'})[1]
```

```python
tickers = sp500['Symbol'].tolist() # Extract tickers
```

```python
# Display data
sp500.head()
```

```
  Symbol            Security              GICS Sector              GICS Sub-Industry  Headquarters Location  Date added      CIK     Founded
0    MMM                  3M              Industrials        Industrial Conglomerates  Saint Paul, Minnesota  1957-03-04    66740        1902
1    AOS         A. O. Smith              Industrials               Building Products   Milwaukee, Wisconsin  2017-07-26    91142        1916
2    ABT Abbott Laboratories              Health Care           Health Care Equipment  North Chicago, Illinois 1957-03-04     1800        1888
3   ABBV              AbbVie              Health Care                   Biotechnology  North Chicago, Illinois 2012-12-31  1551152 2013 (1888)
4    ACN           Accenture   Information Technology IT Consulting & Other Services          Dublin, Ireland  2011-07-06  1467373        1989
```

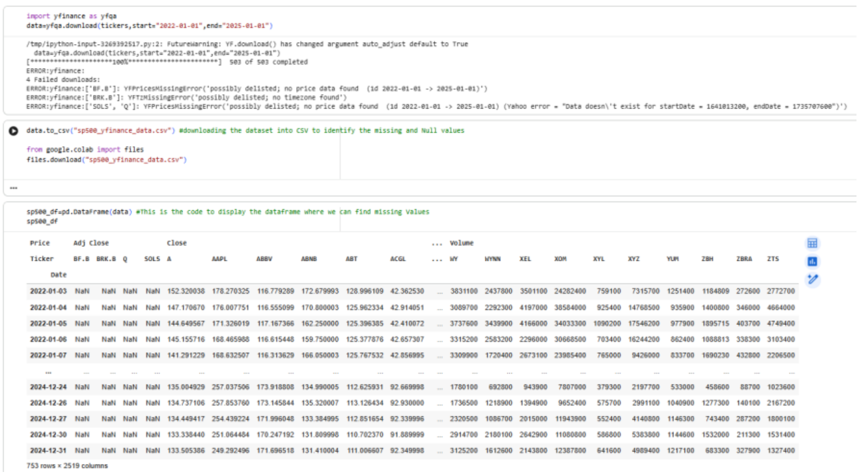
Part B- Loading the DataFrame

To import the daily price data for the S&P 500 companies, I first explored different Python libraries. Initially, I considered using YahooFinancials, but after reviewing recent documentation and user reports, I found that the package has become outdated. It often fails to retrieve complete historical data, and many tickers return empty or incomplete results. Because reliability and data completeness are essential for financial analysis, I decided not to use YahooFinancials.

Instead, I selected the yfinance library, which is currently one of the most widely used and well-maintained Python packages for downloading stock market data. It provides consistent access to Yahoo Finance's historical datasets and supports bulk downloading of multiple tickers at once, making it more suitable for this type of large-scale analysis.

Using yfinance, I downloaded daily price data for all S&P 500 tickers for the period 01-01-2022 to 01-01-2025. The API successfully returned data for 503 tickers, although 4 of them had no price data at all. To verify this, I exported the full raw dataset into a CSV file and inspected it manually. This step helped ensure that missing tickers were correctly identified rather than overlooked by the script.

The final DataFrame contained 753 rows, representing the number of trading days in the selected three-year period. It also had 2,519 columns, because each ticker includes six separate fields:Open, High, Low, Close, Adjusted Close, and Volume.
This structure allowed me to clearly observe which tickers had complete series and which contained entire columns of NaN values.



```
import yfinance as yfqa
data=yfqa.download(tickers,start="2022-01-01",end="2025-01-01")
```

```
/tmp/ipython-input-3269392517.py:2: FutureWarning: YF.download() has changed argument auto_adjust default to True
  data=yfqa.download(tickers,start="2022-01-01",end="2025-01-01")
[*********************100%***********************]  503 of 503 completed
ERROR:yfinance:
4 Failed downloads:
ERROR:yfinance:['BF.B']: YFPricesMissingError('possibly delisted; no price data found  (1d 2022-01-01 -> 2025-01-01)')
ERROR:yfinance:['BRK.B']: YFTzMissingError('possibly delisted; no timezone found')
ERROR:yfinance:['SOLS', 'Q']: YFPricesMissingError('possibly delisted; no price data found  (1d 2022-01-01 -> 2025-01-01) (Yahoo error = "Data doesn\'t exist for startDate = 1641013200, endDate = 1735707600")')
```

```
data.to_csv("sp500_yfinance_data.csv") #downloading the dataset into CSV to identify the missing and Null values

from google.colab import files
files.download("sp500_yfinance_data.csv")
```

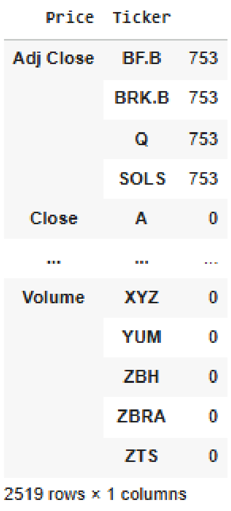
```
sp500_df=pd.DataFrame(data) #This is the code to display the dataframe where we can find missing Values
sp500_df
```

Price	Adj Close				Close						...	Volume									
Ticker	BF.B	BRK.B	Q	SOLS	A	AAPL	ABBV	ABNB	ABT	ACGL	...	WY	WYNN	XEL	XOM	XYL	XYZ	YUM	ZBH	ZBRA	ZTS
Date																					
2022-01-03	NaN	NaN	NaN	NaN	152.320038	178.270325	116.779289	172.679993	128.996109	42.362530	...	3831100	2437800	3501100	24282400	759100	7315700	1251400	1184809	272600	2772700
2022-01-04	NaN	NaN	NaN	NaN	147.170670	176.007751	116.555099	170.800003	125.962334	42.914051	...	3089700	2292300	4197000	38584000	925400	14768500	935900	1400800	346000	4664000
2022-01-05	NaN	NaN	NaN	NaN	144.649567	171.326019	117.167366	162.250000	125.396385	42.410072	...	3737600	3439900	4166000	34033300	1090200	17546200	977900	1895715	403700	4749400
2022-01-06	NaN	NaN	NaN	NaN	145.155716	168.465988	116.615448	159.750000	125.377876	42.657307	...	3315200	2583200	2296000	30668500	703400	16244200	862400	1088813	338300	3103400
2022-01-07	NaN	NaN	NaN	NaN	141.291229	168.632507	116.313629	166.050003	125.767532	42.856995	...	3309900	1720400	2673100	23985400	765000	9426000	833700	1690230	432800	2206500
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2024-12-24	NaN	NaN	NaN	NaN	135.004929	257.037506	173.918808	134.990005	112.625931	92.669998	...	1780100	692800	943900	7807000	379300	2197700	533000	458600	88700	1023600
2024-12-26	NaN	NaN	NaN	NaN	134.737106	257.853760	173.145844	135.320007	113.126434	92.930000	...	1736500	1218900	1394900	9652400	575700	2991100	1040900	1277300	140100	2167200
2024-12-27	NaN	NaN	NaN	NaN	134.449417	254.439224	171.996048	133.384995	112.851654	92.339996	...	2320500	1086700	2015000	11943900	552400	4140800	1146300	743400	287200	1800100
2024-12-30	NaN	NaN	NaN	NaN	133.338440	251.064484	170.247192	131.809998	110.702370	91.889999	...	2914700	2180100	2642900	11080800	586800	5383800	1144600	1532000	211300	1531400
2024-12-31	NaN	NaN	NaN	NaN	133.505386	249.292496	171.696518	131.410004	111.006607	92.349998	...	3125200	1612600	2143800	12387800	641600	4989400	1217100	683300	327900	1327400

753 rows × 2519 columns

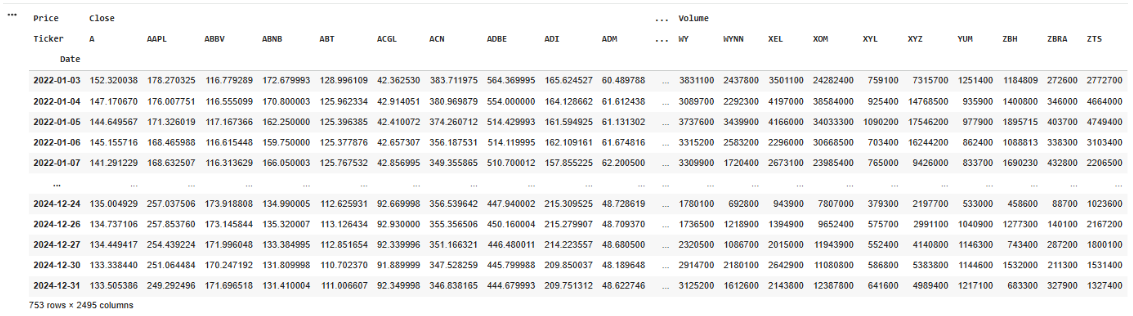
Part C- Cleaning and removing Invalid Data

For this step, I first checked the loaded dataset to identify which tickers were fully missing by using data.isna().sum() and the missing-value loop, which helped me clearly see the tickers that had no data at all. After that, I removed those useless columns using dropna(axis=1, how='all') because fully empty tickers do not contribute anything. For the analysis, I selected only the Close price since closing prices are commonly used for calculating returns and beta. Then, instead of removing partially missing values, I filled the gaps using **forward-fill (ffill) and backward-fill (bfill)** so the time series stays complete without losing important financial information. After cleaning, I ended up with a dataset of 753 rows and 499 tickers, which is the final clean input used for my calculations.



Price	Ticker	
Adj Close	BF.B	753
	BRK.B	753
	Q	753
	SOLS	753
Close	A	0
...	...	...
Volume	XYZ	0
	YUM	0
	ZBH	0
	ZBRA	0
	ZTS	0

2519 rows × 1 columns



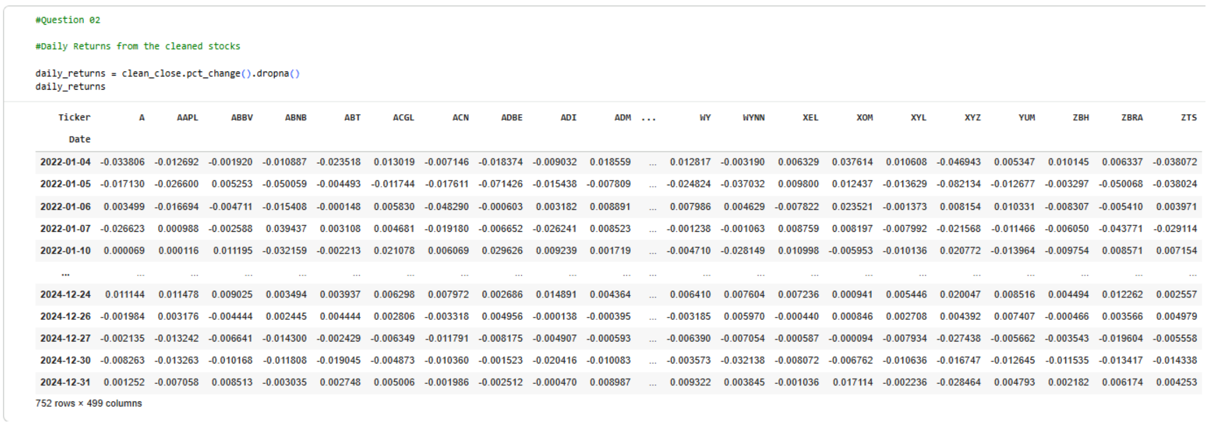
Price	Close										...	Volume									
Ticker	A	AAPL	ABBV	ABNB	ABT	ACGL	ACN	ADBE	ADI	ADM	...	WY	WYNN	XEL	XOM	XYL	XYZ	YUM	ZBH	ZBRA	ZTS
Date																					
2022-01-03	152.320038	178.270325	116.779289	172.679993	128.996109	42.362530	383.711975	564.369995	165.624527	60.489788	...	3831100	2437800	3501100	24282400	759100	7315700	1251400	1184809	272600	2772700
2022-01-04	147.170670	176.007751	116.555099	170.800003	125.962334	42.914051	380.969879	554.000000	164.128662	61.612438	...	3089700	2292300	4197000	38584000	925400	14768500	935900	1400800	346000	4664000
2022-01-05	144.649567	171.326019	117.167366	162.250000	125.396385	42.410072	374.260712	514.429993	161.594925	61.131302	...	3737600	3439900	4166000	34033300	1090200	17546200	977900	1895715	403700	4749400
2022-01-06	145.155716	168.465988	116.615448	159.750000	125.377876	42.657307	356.187531	514.119995	162.109161	61.674816	...	3315200	2583200	2296000	30668500	703400	16244200	862400	1088813	338300	3103400
2022-01-07	141.291229	168.632507	116.313629	166.050003	125.767532	42.856995	349.355865	510.700012	157.855225	62.200500	...	3309900	1720400	2673100	23985400	765000	9426000	833700	1690230	432800	2206500
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2024-12-24	135.004929	257.037506	173.918808	134.990005	112.625931	92.669998	356.539642	447.940002	215.309525	48.728619	...	1780100	692800	943900	7807000	379300	2197700	533000	458600	88700	1023600
2024-12-26	134.737106	257.853760	173.145844	135.320007	113.126434	92.930000	355.356506	450.160004	215.279907	48.709370	...	1736500	1218900	1394900	9652400	575700	2991100	1040900	1277300	140100	2167200
2024-12-27	134.449417	254.439224	171.996048	133.384995	112.851654	92.339996	351.166321	446.480011	214.223557	48.680500	...	2320500	1086700	2015000	11943900	552400	4140800	1146300	743400	287200	1800100
2024-12-30	133.338440	251.064484	170.247192	131.809998	110.702370	91.889999	347.528259	445.799988	209.850037	48.189648	...	2914700	2180100	2642900	11080800	586800	5383800	1144600	1532000	211300	1531400
2024-12-31	133.505386	249.292496	171.696518	131.410004	111.006607	92.349998	346.838165	444.679993	209.751312	48.622746	...	3125200	1612600	2143800	12387800	641600	4989400	1217100	683300	327900	1327400

753 rows × 2495 columns

Q2 Calculate in Google Colab the Daily Return, Beta and Annual Volatility Metrics:

Part A Calculating Daily Return

I used the **clean_close.pct_change().dropna()** code to calculate the daily returns because it automatically shows how much each stock went up or down compared to the previous day. For example, in my output on 2022-01-05, the stock AAPL shows a return of –0.026600, which means AAPL fell by about –2.66% on that day. On the same day, ABBV has a return of 0.005253, meaning it increased by around +0.52%. This clearly shows how different stocks behave daily, some drop while others rise.For investors, negative daily returns indicate short-term declines, which may worry low-risk investors but can offer opportunities for risk-takers. Positive daily returns show stability or upward movement, which is generally preferred by long-term, low-risk investors. Overall, daily returns help investors understand whether a stock moves calmly or fluctuates sharply, allowing them to pick stocks that match their risk level.



```python
#Question 02

#Daily Returns from the cleaned stocks

daily_returns = clean_close.pct_change().dropna()
daily_returns
```

Ticker	A	AAPL	ABBV	ABNB	ABT	ACGL	ACN	ADBE	ADI	ADM	...	WY	WYNN	XEL	XOM	XYL	XYZ	YUM	ZBH	ZBRA	ZTS
Date																					
2022-01-04	-0.033806	-0.012692	-0.001920	-0.010887	-0.023518	0.013019	-0.007146	-0.018374	-0.009032	0.018559	...	0.012817	-0.003190	0.006329	0.037614	0.010608	-0.046943	0.005347	0.010145	0.006337	-0.038072
2022-01-05	-0.017130	-0.026600	0.005253	-0.050059	-0.004493	-0.011744	-0.017611	-0.071426	-0.015438	-0.007809	...	-0.024824	-0.037032	0.009800	0.012437	-0.013629	-0.082134	-0.012677	-0.003297	-0.050068	-0.038024
2022-01-06	0.003499	-0.016694	-0.004711	-0.015408	-0.000148	0.005830	-0.048290	-0.000603	0.003182	0.008891	...	0.007986	0.004629	-0.007822	0.023521	-0.001373	0.008154	0.010331	-0.008307	-0.005410	0.003971
2022-01-07	-0.026623	0.000988	-0.002588	0.039437	0.003108	0.004681	-0.019180	-0.006652	-0.026241	0.008523	...	-0.001238	-0.001063	0.008759	0.008197	-0.007992	-0.021568	-0.011466	-0.006050	-0.043771	-0.029114
2022-01-10	0.000069	0.000116	0.011195	-0.032159	-0.002213	0.021078	0.006069	0.029626	0.009239	0.001719	...	-0.004710	-0.028149	0.010998	-0.005953	-0.010136	0.020772	-0.013964	-0.009754	0.008571	0.007154
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2024-12-24	0.011144	0.011478	0.009025	0.003494	0.003937	0.006298	0.007972	0.002686	0.014891	0.004364	...	0.006410	0.007604	0.007236	0.000941	0.005446	0.020047	0.008516	0.004494	0.012262	0.002557
2024-12-26	-0.001984	0.003176	-0.004444	0.002445	0.004444	0.002806	-0.003318	0.004956	-0.000138	-0.000395	...	-0.003185	0.005970	-0.000440	0.000846	0.002708	0.004392	0.007407	-0.000466	0.003566	0.004979
2024-12-27	-0.002135	-0.013242	-0.006641	-0.014300	-0.002429	-0.006349	-0.011791	-0.008175	-0.004907	-0.000593	...	-0.006390	-0.007054	-0.000587	-0.000094	-0.007934	-0.027438	-0.005662	-0.003543	-0.019604	-0.005558
2024-12-30	-0.008263	-0.013263	-0.010168	-0.011808	-0.019045	-0.004873	-0.010360	-0.001523	-0.020416	-0.010083	...	-0.003573	-0.032138	-0.008072	-0.006762	-0.010636	-0.016747	-0.012645	-0.011535	-0.013417	-0.014338
2024-12-31	0.001252	-0.007058	0.008513	-0.003035	0.002748	0.005006	-0.001986	-0.002512	-0.000470	0.008987	...	0.009322	0.003845	-0.001036	0.017114	-0.002236	-0.028464	0.004793	0.002182	0.006174	0.004253

752 rows × 499 columns

Part B – Calculating Beta

To calculate the beta values, I used the S&P 500 index (GSPC) as the market benchmark because it is the standard measure of overall market movement and most S&P-500 companies naturally compare well against it. After calculating daily returns for each stock and aligning them with the S&P 500 returns, the code applied the beta formula using correlation and standard deviation. This is why the output shows different beta values across the stocks. For example, AAPL has a beta of around 1.21, which means it tends to move slightly more than the market when the market rises or falls, AAPL usually reacts stronger. On the other hand, ABBV has a beta around 0.29, showing that it moves much less than the market and is more stable. These differences come from each stock's volatility and how closely it follows market trends. Based on these results, I can advise investors that low-beta stocks are more suitable if they prefer stability, beta values close to 1 suit investors who want performance similar to the market, and high-beta stocks are better for those willing to accept more risk for potentially higher movement in returns.

Part C – Calculating Annual volatility

To calculate the Annual Volatility, I used the daily returns of the cleaned Close-price dataset. The code first calculated daily returns using pct_change(), then found the daily standard deviation, and finally converted it to annual volatility using the formula:

$$\text{Annual Volatility} = \text{Daily Std Dev} \times \sqrt{252}$$

The number 252 represents the average number of trading days in a year. This method is standard in finance because daily volatility alone does not show the true risk level over a full year, so multiplying by √252 annualises it.

The code steps clearly reflect this approach:

Step 1: calculate daily returns

Step 2: calculate standard deviation of daily returns

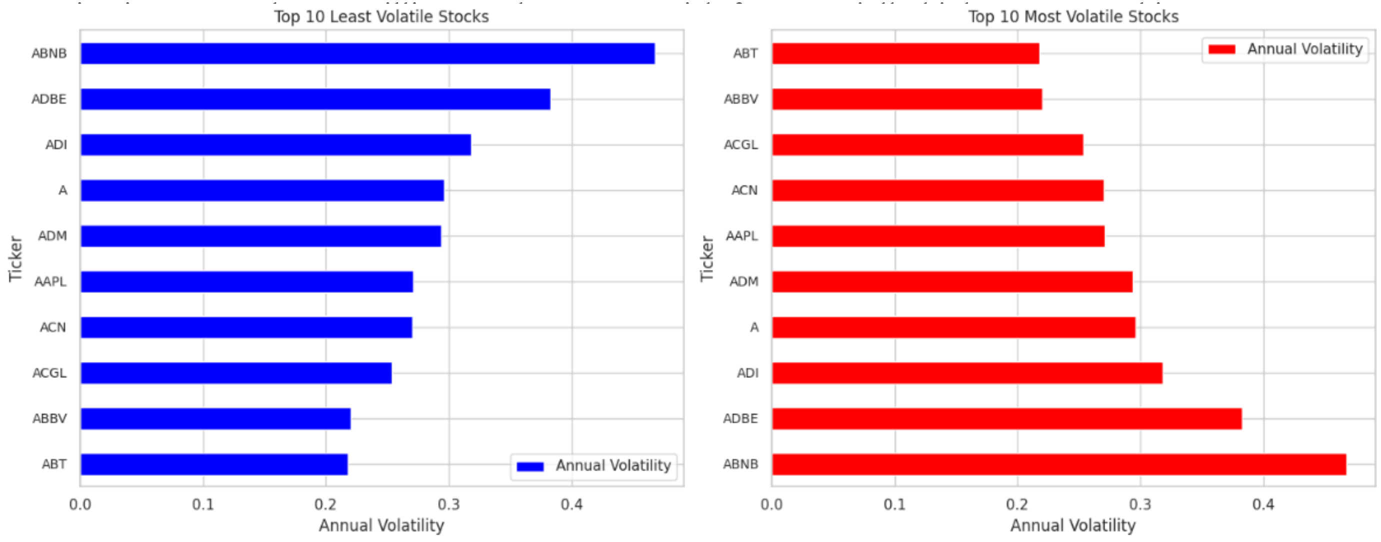
Step 3: annualise using √252

Step 4: convert results into a clean DataFrame

This is why each stock in the output shows a single volatility number between 0 and 1, representing how much the stock typically fluctuates in a year.

From my results, AAPL has an annual volatility of around 0.27, meaning it has moderate price fluctuations over the year. In contrast, ABNB has a volatility around 0.46, meaning it experiences larger ups and downs and is more unpredictable. These differences appear because each stock has different daily return behaviour some move steadily while others react more sharply to market events.

Based on these results, I can advise investors that low-volatility stocks (around 0.20–0.30) are more suitable for conservative investors looking for stability, while high-volatility stocks (above 0.40) fit a
h
a



Q03.Stock segmentation with agglomerative clustering

Part A- Appropriateness of Agglomerative Clustering for Stock Segmentation

Agglomerative clustering is appropriate for segmenting the stocks based on their Beta values because it groups stocks according to how similar or different their market sensitivity is. Since Beta is a single continuous metric, this method can easily measure the distance between stocks and merge the most similar ones step by step. One advantage of agglomerative clustering is that it does not require selecting the number of clusters at the start, which allows me to observe how stocks naturally form groups using the dendrogram. It also provides a clear visual structure that shows how low-beta, medium-beta, and high-beta stocks join together, making it easier to understand risk patterns in the portfolio. Overall, agglomerative clustering is suitable for this task because it helps identify meaningful stock groups based purely on Beta, which supports better analysis and decision-making for investors.

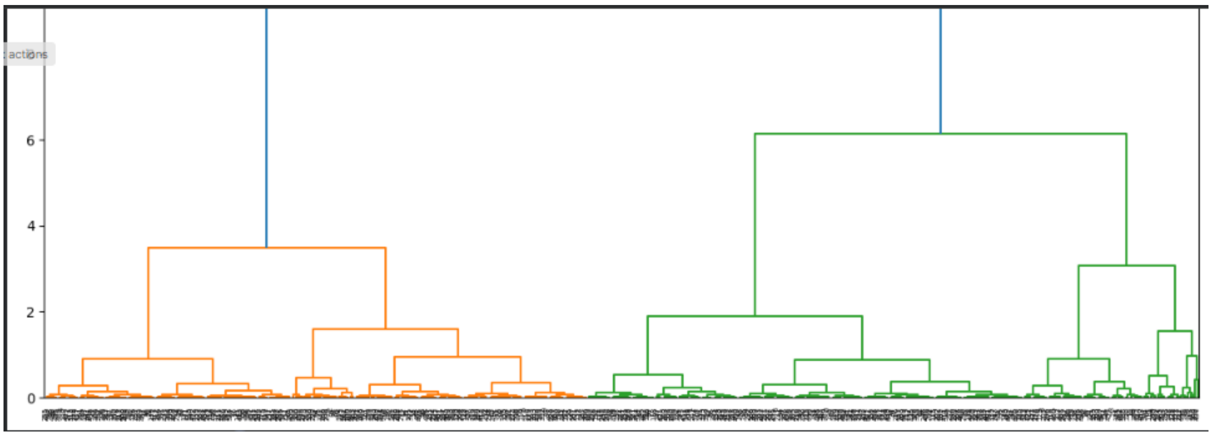
Part B-Techniques Used to Determine the Optimal Number of Clusters

To determine the most suitable number of clusters for segmenting the stocks based on Beta, I used two techniques: the dendrogram and the silhouette score.

1. Dendrogram Method

The dendrogram allowed me to visually observe how stocks merge together step-by-step based on their Beta values. By following the hierarchical structure, I focused on the largest vertical jump in the diagram, which shows the point where the data naturally separates into distinct groups. From this jump, I identified that the natural split occurs at K = 2, meaning the stocks form two broad clusters.

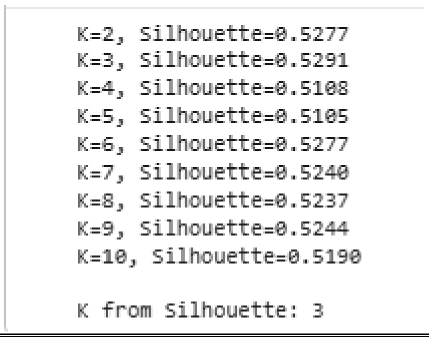
Using K = 2 would place all stocks into just two categories: low-beta and high-beta. While this split is technically correct, it is too broad to be practically useful for investors. Most investors including balanced and moderate-risk investors need a clearer distinction between low risk, medium risk, and high-risk stocks. With only two clusters, the entire middle-risk group becomes mixed with the high beta stocks, making it difficult for investors to identify suitable options that match their portfolio preferences. Therefore, even though the dendrogram suggested K = 2, this segmentation does not provide the level of detail required for effective investment decision-making.



2. Silhouette Score

I then calculated the silhouette scores for K = 2 to K = 10. The silhouette score measures how well each stock fits within its assigned cluster compared to other clusters. A higher score means the clusters are more distinct and better separated. In my results, the highest silhouette value appeared at K = 3, which indicates that using three clusters provides a strong balance between compactness and separation.

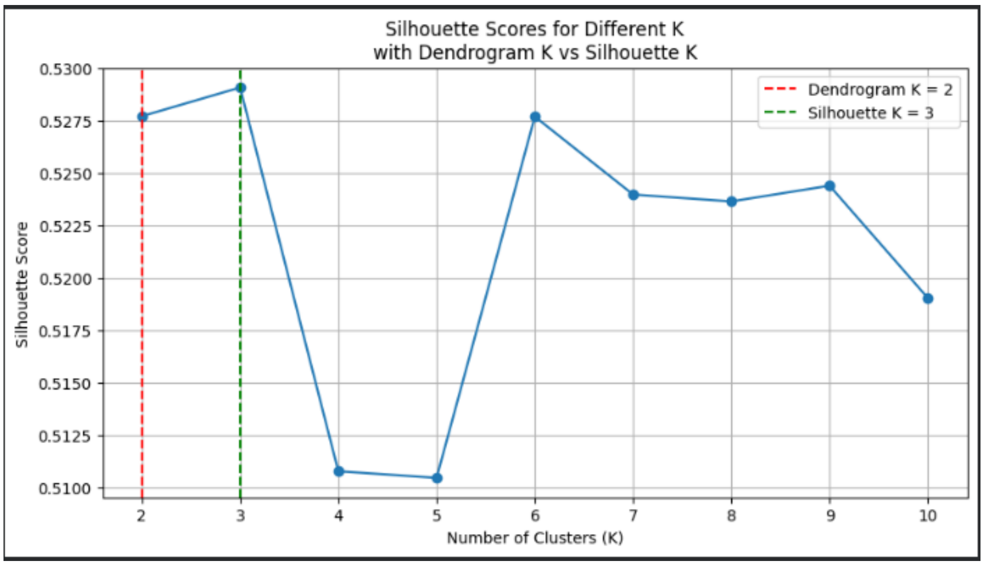
Choosing K = 3 allows me to create three clearly defined groups of stocks based on their Beta values. This is useful for investors because it introduces a middle risk category that was missing in the dendrogram's K = 2 suggestion. With three clusters, investors can now differentiate between low-risk, medium-risk, and high-risk stocks, giving them more flexibility when matching stock selections to their personal risk appetite. This leads to better diversification and more informed portfolio construction.

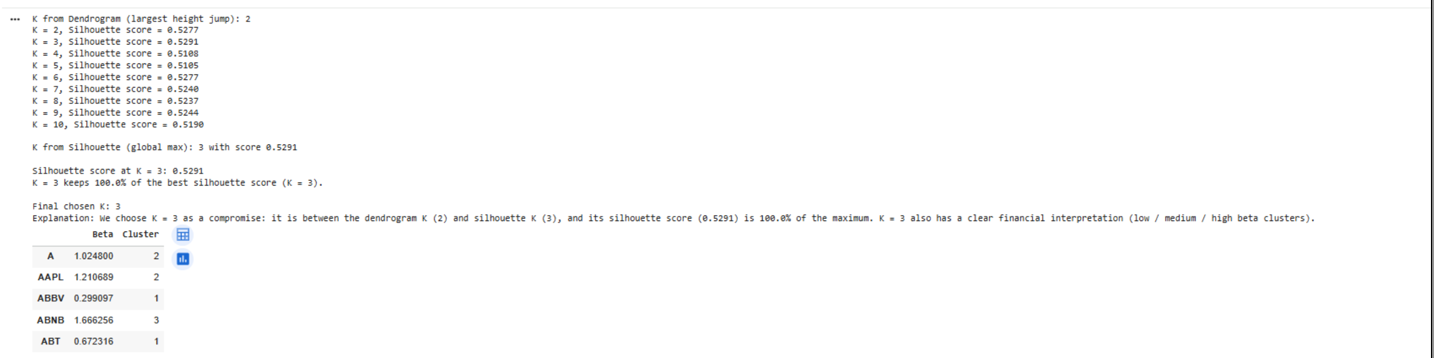

```
K=2, Silhouette=0.5277
K=3, Silhouette=0.5291
K=4, Silhouette=0.5108
K=5, Silhouette=0.5105
K=6, Silhouette=0.5277
K=7, Silhouette=0.5240
K=8, Silhouette=0.5237
K=9, Silhouette=0.5244
K=10, Silhouette=0.5190

K from Silhouette: 3
```

Even though the dendrogram suggested K = 2, the silhouette score and investor usefulness strongly supported K = 3. Choosing K = 3 gives clearer risk segmentation, which helps investors pick stocks based on whether they prefer stability, balanced exposure, or higher risk for higher returns.

Part C-Implementation of Agglomerative Clustering Using Python in Google Colab

After selecting K = 3 as the most meaningful number of clusters, I implemented the agglomerative clustering model in Python. I used the Ward linkage method to build the linkage matrix, plotted the dendrogram to understand how the stocks merge, and then assigned the final cluster labels using fcluster(). To support the decision made in Part B, I also created a comparison graph that shows the K value from the dendrogram (K = 2) and the K value from the silhouette method (K = 3). The silhouette curve clearly peaks at K = 3, confirming that this option provides stronger separation and more reliable grouping. From an investor point of view, choosing three clusters creates a practical risk-based structure: Cluster 1 represents low-beta stocks with more stable performance, Cluster 2 includes medium-beta stocks that move in line with the market and suit balanced investors, and Cluster 3 contains high-beta stocks with more aggressive movements, making them suitable for return-seeking investors who are comfortable with higher risk. This makes the clustering results directly useful when recommending stock categories to different types of investors.





```
K from Dendrogram (largest height jump): 2
K = 2, Silhouette score = 0.5277
K = 3, Silhouette score = 0.5291
K = 4, Silhouette score = 0.5108
K = 5, Silhouette score = 0.5105
K = 6, Silhouette score = 0.5277
K = 7, Silhouette score = 0.5240
K = 8, Silhouette score = 0.5237
K = 9, Silhouette score = 0.5244
K = 10, Silhouette score = 0.5190

K from Silhouette (global max): 3 with score 0.5291

Silhouette score at K = 3: 0.5291
K = 3 keeps 100.0% of the best silhouette score (K = 3).

Final chosen K: 3
Explanation: We choose K = 3 as a compromise: it is between the dendrogram K (2) and silhouette K (3), and its silhouette score (0.5291) is 100.0% of the maximum. K = 3 also has a clear financial interpretation (low / medium / high beta clusters).
```

	Beta	Cluster
A	1.024800	2
AAPL	1.210689	2
ABBV	0.299097	1
ABNB	1.666256	3
ABT	0.672316	1

Q04- Stock segmentation with K-Means

Part A - **Appropriateness of K-Means Clustering for Stock Segmentation**

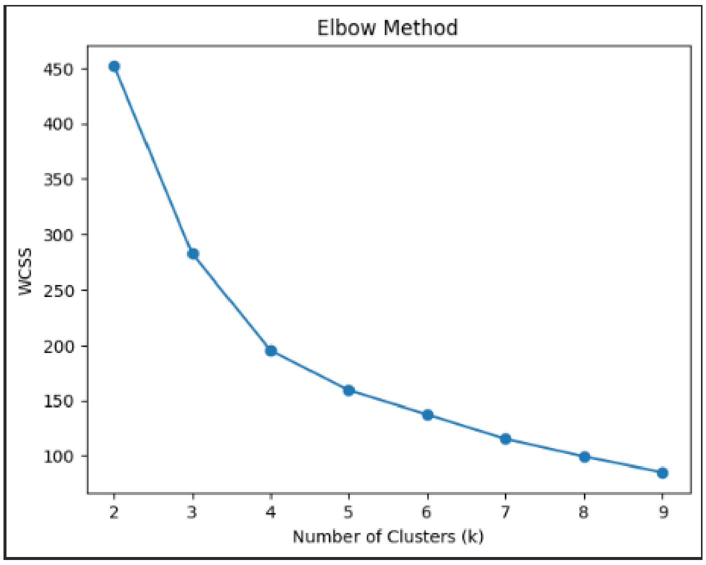
K-Means is appropriate for this task because both **Beta** and **Annual Volatility** are continuous numerical variables, and K-Means works best with this type of data because it groups items based on their distance in feature space. Since these two metrics represent different dimensions of stock risk, K-Means helps to naturally form **tiered risk groups** such as low-risk, medium-risk, and high-risk stocks. The method is also computationally efficient and widely used in finance for portfolio segmentation, especially when the goal is to create clusters with similar risk characteristics. After scaling the data, K-Means produces clear, well-separated clusters, allowing us to interpret the behaviour of each stock group easily and communicate meaningful insights to investors

Part B- Techniques Used to Determine the Optimal Number of Clusters

To decide the best number of clusters for the K-Means model, I used two recognised validation techniques: the Elbow Method and the Silhouette Score, because both are commonly recommended when clustering numerical data like Beta and Annual Volatility.

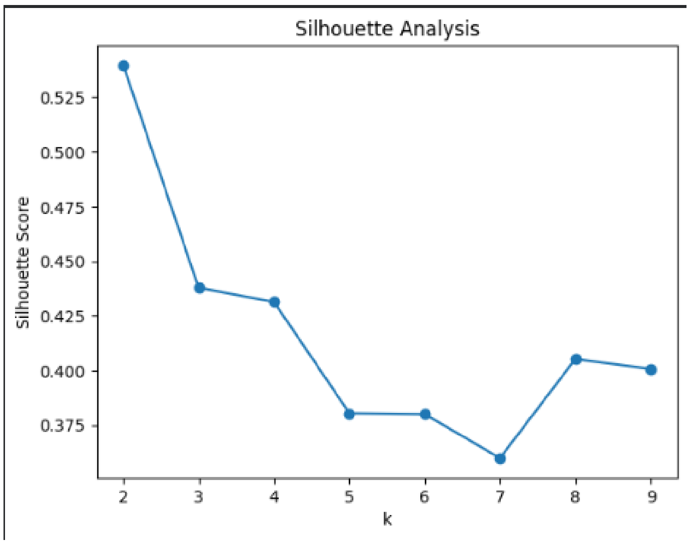
Elbow Method

First, I calculated the Within-Cluster Sum of Squares (WCSS) for K values from 2 to 9 and plotted the Elbow graph. This shows how much the clustering improves when increasing K. From my output, the WCSS dropped sharply from K=2 to K=3 and then started to flatten gradually. This "bend" in the curve indicates that K = 3 is a reasonable point where adding more clusters does not significantly improve compactness. Therefore, the Elbow Method suggested that K=3 is a suitable choice.



Silhouette Analysis

Next, I computed the Silhouette Score for the same K range. The Silhouette Score measures how well-separated and cohesive the clusters are. In my results, the highest score appeared at K = 2, meaning that with two clusters, the separation between groups was strongest. However, the score for K=3 was still acceptable and provided more meaningful segmentation for investor interpretation compared to K=2, which was too broad.



Final Choice of K

Since the Elbow Method supported K = 3 and the Silhouette Score showed that K = 2 and K = 3 were the strongest options, I selected K = 3 as the final value. Choosing K=3 gave me well-balanced clusters and allowed me to form three clear risk tiers (low, medium, high), which makes more sense from an investor insight perspective.

Part C- Implementation of K-Means Clustering Using Python in Google Colab

To implement K-Means clustering, I first combined the Beta and Annual Volatility metrics and standardised them using StandardScaler, since K-Means relies on distance. I then used the Elbow Method by calculating WCSS for K values from 2–9 and plotting the curve. The bend appeared around K= 3, suggesting that three clusters capture the main structure in the data. I next used Silhouette Analysis, which showed the highest score at K = 2, but K = 3 still had a strong silhouette value and provided more useful segmentation for investment decisions. Based on both techniques, I selected K= 3 as the final number of clusters.

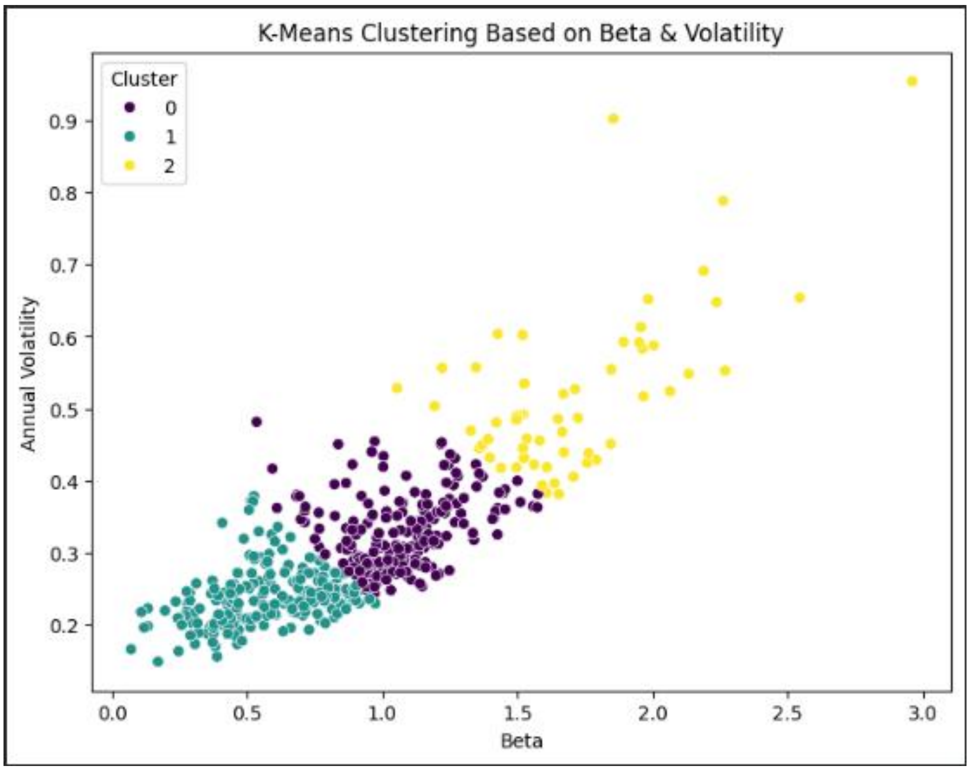
After choosing K, I ran KMeans(n_clusters=3) and assigned each stock to a cluster. I then plotted the results using a scatterplot of Beta vs Annual Volatility, coloured by cluster. The plot clearly shows three separate groups:

Cluster 0: low Beta & low volatility

Cluster 1: medium Beta & medium volatility

Cluster 2: high Beta & high volatility

These three clusters help investors identify which stocks best match their risk preferences. Cluster 0 is suited for low-risk, stable investment strategies, Cluster 1 represents balanced risk and moderate growth, and Cluster 2 highlights high-risk stocks that may offer higher returns but also higher volatility.



Q05- Business Value of Daily Return, Beta & Annual Volatility for Portfolio Diversification

The three financial metrics used in this analysis Daily Return, Beta, and Annual Volatility work together to help investors understand stock behaviour from different perspectives. When these metrics are applied to cluster stocks, they create groups with distinct risk return characteristics, allowing investors to diversify their portfolios more effectively.

Daily Return – Short-Term Performance Indicator
Daily returns show how much a stock's price changes from one day to another. Stocks with stable daily returns typically behave more predictably, while stocks with large fluctuations indicate short-term risk. Within the clusters, stocks with smoother daily returns form lower risk groups, making them suitable for conservative investors. On the other hand, clusters containing stocks with highly volatile daily returns highlight opportunities for investors seeking short-term movements or momentum trading.

2. Beta Market Sensitivity
Beta measures how strongly a stock moves relative to the overall market. When clusters are built using Beta, they naturally separate into low-beta, medium-beta, and high-beta groups.

- Low-Beta clusters contain defensive stocks that move less than the market, helping reduce overall portfolio risk.

- Medium-Beta clusters offer balanced exposure, suitable for diversified portfolios.

- High-Beta clusters include aggressive stocks that move more sharply with the market, appealing to investors looking for higher potential returns.

This dependency between clustering and Beta enables investors to understand how each cluster reacts to market movements and select stocks based on their risk tolerance.

3. Annual Volatility – Long-Term Risk Profile
While daily return volatility gives a short-term view, annual volatility tells investors how much a stock typically fluctuates over a year. Clusters formed using this metric highlight the long-term stability or instability of stocks.

- Low-volatility clusters represent stable, predictable stocks ideal for long-term, risk-averse investors.

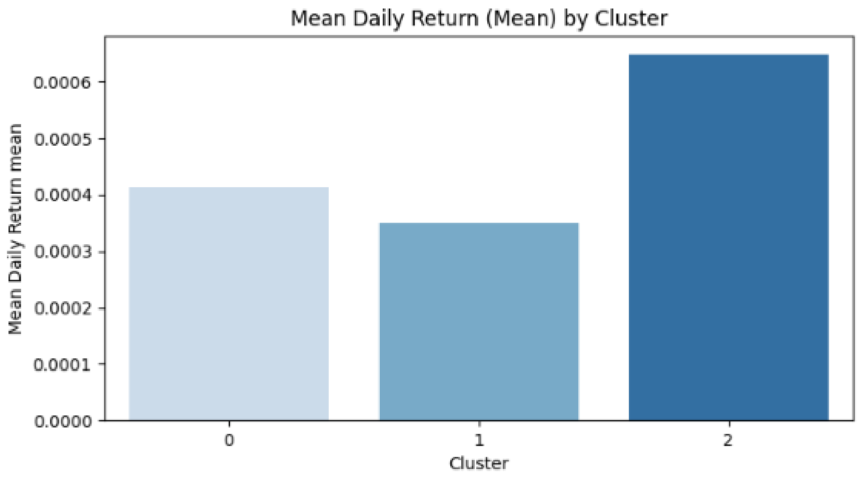
- High-volatility clusters signal stocks that experience larger price swings, suitable for aggressive or growth-oriented investors.

Because volatility directly reflects uncertainty, it becomes one of the strongest indicators for assigning stocks to appropriate risk tiers within a diversified portfolio.

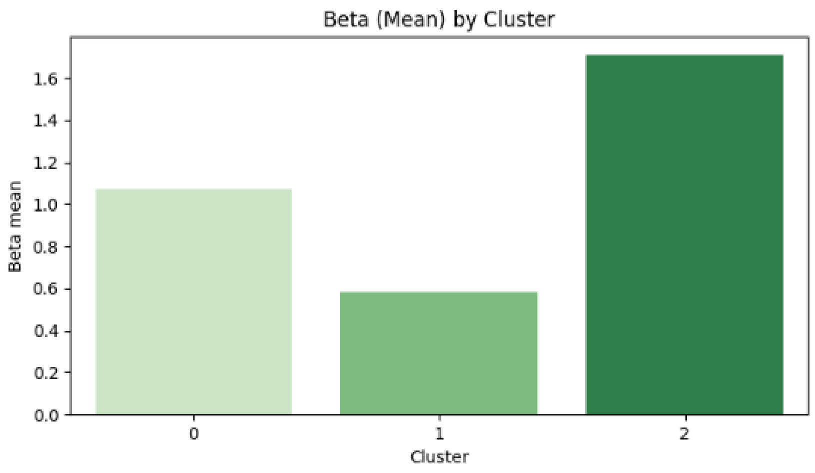
Overall Value for Investors By combining these three metrics, the final clusters clearly separate stocks into distinct behavioural groups, allowing investors to Spread their investments across low-, medium-, and high-risk categories Balance short-term price movement with long-term stability Manage exposure to market sensitivity (Beta)Tailor portfolios to conservative, balanced, or aggressive strategies In summary, clustering based on Daily Return, Beta, and Annual Volatility adds significant business value because it transforms raw market data into actionable stock categories. portfolios that match their risk appetite while improving diversification and reducing exposure to unnecessary risk.

Part B-Implementation using Python via Google Colab

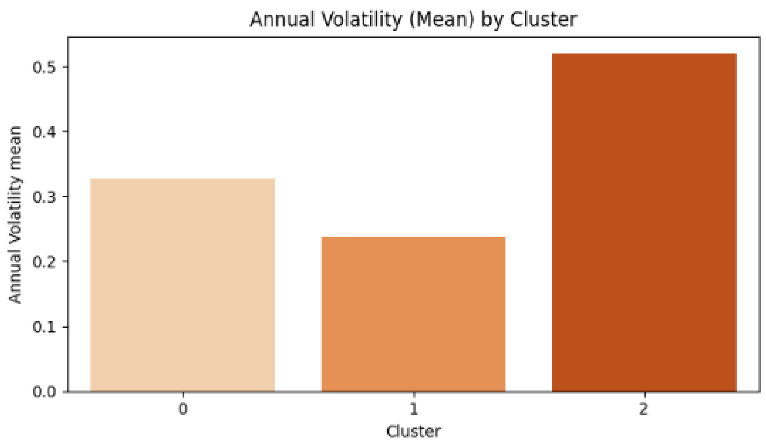
The Mean Daily Return chart shows that Cluster 2 achieves the highest average returns, while Cluster 1 produces the lowest. This means stocks in Cluster 2 generally deliver stronger day to day performance, whereas Cluster 1 reflects more modest growth. Understanding these patterns helps identify which clusters contribute more to overall portfolio returns, making it easier to balance growth with stability when constructing an investment strategy



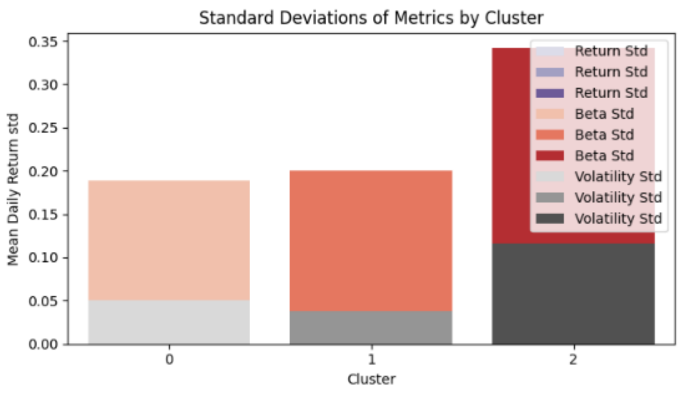
The Beta comparison highlights clear risk differences between the clusters. Cluster 2 has the highest Beta values, meaning these stocks move more aggressively with the market, whereas Cluster 1 has the lowest Beta and behaves more defensively. This allows a portfolio to be structured based on market sensitivity more stable clusters can support downside protection, while higher-beta clusters can be used to capture stronger market upswings.



Annual Volatility shows how much each cluster fluctuates over a full year. Cluster 2 exhibits the highest volatility, indicating greater unpredictability, while Cluster 1 remains the most stable. These differences make it easier to match portfolio holdings with risk preferences; more stable clusters support long-term consistency, while high-volatility clusters add growth potential despite their higher uncertainty.



The standard deviation plot shows how consistently stocks behave within each cluster. Cluster 2 has the highest variation across all metrics, meaning individual stocks in this group behave differently from one another, while Cluster 1 is the most uniform. This consistency is useful when selecting dependable performers, while clusters with higher variability may require more careful stock selection due to their mixed behaviour.



Conclusion

This coursework began with building a reliable and clean dataset of S&P 500 stocks, which formed the foundation for all later analysis. In Question 1, I first loaded the list of S&P 500 tickers from Wikipedia and then attempted to load their historical price data. Through testing, I found that the YahooFinancials package was outdated and did not return complete data, so I switched to yfinance, which provided accurate daily prices from 01-01-2022 to 01-01-2025. After loading 503 tickers, I identified four invalid ones that contained fully missing values, saved the dataset to CSV for manual verification, and removed unusable tickers. By analysing missing values using Python and applying forward/backward filling where appropriate, I prepared a clean dataset with 753 trading days and a complete set of price fields. This ensured that the dataset was reliable enough for financial calculations and clustering.

Using this clean dataset, the following questions explored key financial metrics Daily Returns, Beta, and Annual Volatility which are crucial for understanding a stock's behaviour. Daily Returns measured short-term price changes, Beta showed how sensitive each stock was compared to the overall market, and Annual Volatility captured long-term risk. These metrics helped build a deeper understanding of stock characteristics, which later supported meaningful segmentation.

Clustering techniques were applied to group stocks into logical categories. Agglomerative clustering (Question 3) helped visualise relationships between stocks using the Ward linkage dendrogram, and both the dendrogram and silhouette score were used to identify the ideal number of clusters. K-Means clustering (Question 4) further evaluated cluster structures using the elbow method and silhouette analysis. Across both methods, three clusters consistently emerged as the most practical and meaningful option, providing a balanced split between low-risk, medium-risk, and high-risk stocks.

Finally, in Question 5, I reviewed how these clusters differ in terms of Daily Returns, Beta, and Volatility. The results showed clear behaviour patterns: one cluster had lower risk and stable performance, another had moderate metrics suitable for balanced portfolios, and the final cluster contained high-beta, high-volatility stocks that offered higher growth potential with higher uncertainty. These insights can help investors choose stocks more confidently based on risk appetite, return expectations, and portfolio diversification goals.

Overall, this project successfully combined data cleaning, financial metric analysis, and clustering techniques to produce investor-ready insights. From loading and preparing the raw S&P 500 data to identifying clear risk-based stock groups, each step contributed to building a strong analytical foundation that can support better portfolio decisions.

References

Wikipedia. (2025). *List of S&P 500 companies*. Available at: https://en.wikipedia.org/wiki/List_of_S%26P_500_companies (Accessed: 01 February 2025).

Yahoo Finance. (2025). *Historical price data*. Available at: https://finance.yahoo.com/ (Accessed: 01 February 2025).

Pandas Documentation. (2025). *pandas.pydata.org*. Available at: https://pandas.pydata.org/

NumPy Documentation. (2025). *numpy.org*. Available at: https://numpy.org/

scikit-learn Documentation. (2025). *scikit-learn.org*.
Clustering algorithms: https://scikit-learn.org/stable/modules/clustering.html
Silhouette score: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

yfinance Documentation. (2025).
Available at: https://pypi.org/project/yfinance/
(This is the library you used to load S&P 500 historical data.)

YahooFinancials (Outdated Package).
Pypi (2021). *yahoofinancials*. Available at: https://pypi.org/project/yahoofinancials/
(Shows last update several years ago → explains outdated behaviour.)

Brealey, R., Myers, S., & Allen, F. (2020). *Principles of Corporate Finance*. 13th ed. McGraw-Hill.
(Explains Beta, market sensitivity, CAPM.)

Investopedia. (2024). *Beta Definition*. Available at: https://www.investopedia.com/terms/b/beta.asp

Hull, J. (2018). *Options, Futures, and Other Derivatives*. 10th ed. Pearson. (Standard derivation of annual volatility.)

Investopedia. (2024). *Volatility Definition*. https://www.investopedia.com/terms/v/volatility.asp

Pandas Documentation — pct_change(). https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.pct_change.html

Murtagh, F. & Legendre, P. (2014). *Ward's Hierarchical Clustering Method: Which Algorithms Implement Ward's Criterion?* Journal of Classification, 31(3), pp. 274–295.

https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering

MacQueen, J. (1967). *Some methods for classification and analysis of multivariate observations*. Berkeley: University of California Press.

scikit-learn — KMeans
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Ketchen, D. J., & Shook, C. L. (1996). *The application of cluster analysis in strategic management research: An analysis and critique*. Strategic Management Journal, 17(6), pp. 441–458.
(One of the foundational academic references.)

Silhouette Method

Rousseeuw, P. J. (1987). *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis*. Journal of Computational and Applied Mathematics, 20, pp. 53–65.

I Have used open AI to structure my report and