Introduction:

In the fast-paced world of finance, it is crucial for both investors and financial professionals to comprehend the performance and trends of certain equities. Comparing and assessing the performance of several stocks over a certain time frame can provide crucial details about their respective benefits, drawbacks, and investment possibilities.

This comparative visualisation research aims to analyse and contrast the stock performance of three selected companies over a lengthy period, from November 2019 to May 2023. By thoroughly examining and visually displaying the swings and patterns in their stock values, we may get a clear understanding of how the firms behave in the market and identify themes that may be the cause of their triumphs or setbacks.

To ensure a more thorough understanding of the performance of the overall market, the three stocks under review have been carefully chosen to represent different market capitalizations and industrial sectors. In order to give a full picture of the evolution of each firm, this research will investigate historical stock data for AMAZON, MICROSOFT, and APPLE and employ a range of visualisation techniques.

Variable Description

- **Date -** identifies the trading day
- **Open -** the price at which trading begins
- **High -** the day's highest price
- **Low -** the day's lowest price
- **Close -** the close price adjusted for splits
- **Adj Close** - adjusted close price that has been split and dividend adjusted.
- **Volume -** the total number of shares that were traded on a certain day.

AIM

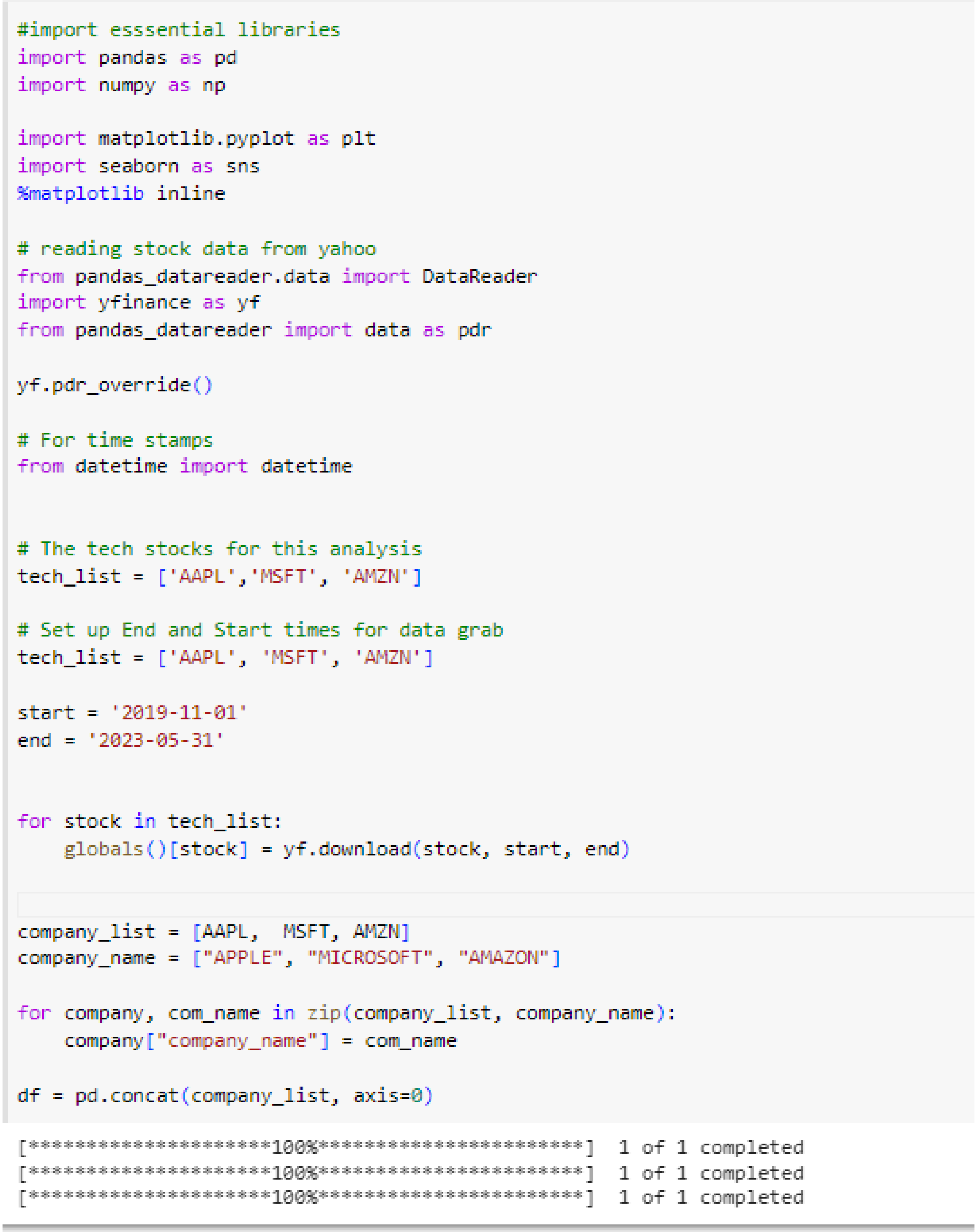
This comparative visualisation study's objective is to evaluate and contrast the stock performance of three chosen businesses from November 2019 to May 2023. The study tries to find trends, patterns, and relative performance amongst the equities by looking at historical data and using different visualisation techniques. The goal is to learn more about how the stocks behave in the market, assess how they respond to key events, and look for any connections or divergences. In order to help investors, financial experts, and market aficionados make wise judgements and comprehend the dynamics of these stocks during the given timeframe, this study aims to offer useful information.

DATASET:

Data Source:
https://finance.yahoo.com/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAALevU7VV1IPpd-2FeyTkry-2dFyTsqHFXHZg7-pF0jYHsJHmmCH-yZdi3JPve7pWPPceU6iv2sAOZpX3HTSJr8-h2scu3t-ZZvg5U-K3TvGTUdN8s9xQywSW6cVx-gzMCX5PW3ZXdS6VyqL2v2mThtrHgT6TZwP8t3gHF4wh7Zkn

DATA PREPARATION:

Before starting with the data pre-processing let us first understand the data by reading it, importing necessary python libraries and the dataset.



```
#import esssential libraries
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# reading stock data from yahoo
from pandas_datareader.data import DataReader
import yfinance as yf
from pandas_datareader import data as pdr

yf.pdr_override()

# For time stamps
from datetime import datetime


# The tech stocks for this analysis
tech_list = ['AAPL','MSFT', 'AMZN']

# Set up End and Start times for data grab
tech_list = ['AAPL', 'MSFT', 'AMZN']

start = '2019-11-01'
end = '2023-05-31'


for stock in tech_list:
    globals()[stock] = yf.download(stock, start, end)


company_list = [AAPL,  MSFT, AMZN]
company_name = ["APPLE", "MICROSOFT", "AMAZON"]

for company, com_name in zip(company_list, company_name):
    company["company_name"] = com_name

df = pd.concat(company_list, axis=0)
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

The code segment performs the following steps:

Library Imports: The necessary libraries are imported, including yfinance, pandas, numpy, seaborn, and matplotlib.pyplot.The charts are rendered inline thanks to matplotlib inline.

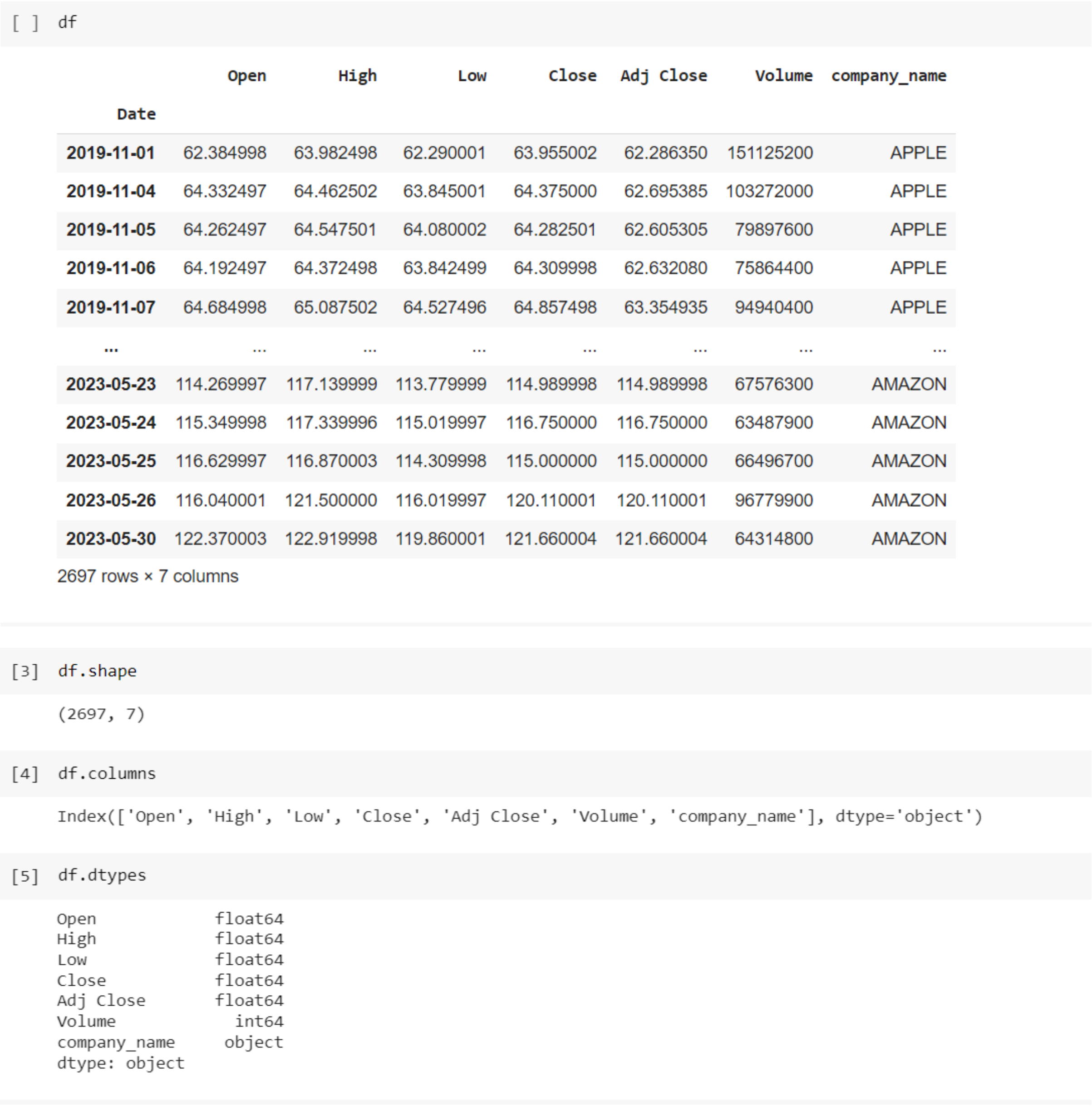
The important libraries used in for the data visualization are:

- Pandas: Pandas is a strong library for data analysis and manipulation. It offers data structures that make processing structured data simple, like Data Frames and Series. Pandas is a well-liked option for data pre-processing and exploratory data analysis activities since it has features for data cleaning, filtering, merging, and aggregating.
- Matplotlib: A popular Python charting library is Matplotlib. Line plots, scatter plots, bar plots, histograms, and other types of plots can be made using its versatile and extensive collection of functions. Matplotlib is appropriate for creating publication-quality visualisations since it enables thorough customisation of plot aesthetics, including colours, labels, titles, and annotations.
- NumPy: The foundational Python library for scientific computing is called NumPy. In order to carry out effective numerical computations, it offers a multidimensional array object in addition to a variety of mathematical operations. Due to its ability to perform quick and vectorized operations on huge datasets, NumPy is a crucial tool for tasks like numerical simulations, statistical analysis, and linear algebra.
- Seaborn: A statistical data visualisation library based on Matplotlib is called Seaborn. It offers a sophisticated user interface for producing attractive and educational statistical visuals. Complex visualisations like heatmaps, violin plots, box plots, and regression plots can be easily created with Seaborn. Additionally, it supports statistical features like automatic estimation and confidence interval graphing.
- yfinance: A handy interface for downloading financial data from Yahoo Finance is offered by the library known as yfinance. In order to do analysis, customers can access historical stock prices, financial statements, and other market information. Yfinance is a well-liked option for financial analysis and quantitative research jobs since it makes the process of gathering and organising financial data easier.

Stock Data Retrieval: The programme downloads historical stock data from Yahoo Finance for the tech giants Apple (AAPL), Microsoft (MSFT), and Amazon (AMZN) using the yfinance library. '2019-11-01' and '2023-05-31' are the start and end dates for the data retrieval, respectively.

Data Preparation: Each company's stock data that was obtained is kept in its own data frame (AAPL, MSFT, AMZN). Each data frame now includes a company name column that displays the name of the corresponding organisation (for example, "APPLE" for AAPL). After using the pd.concat() function to combine the data frames, a single data frame called 'df' with the combined data for all three firms is produced.

In conclusion, the programme gathers historical stock data for Amazon, Apple, and Microsoft and merges it into a single data frame for further analysis and visualisation.



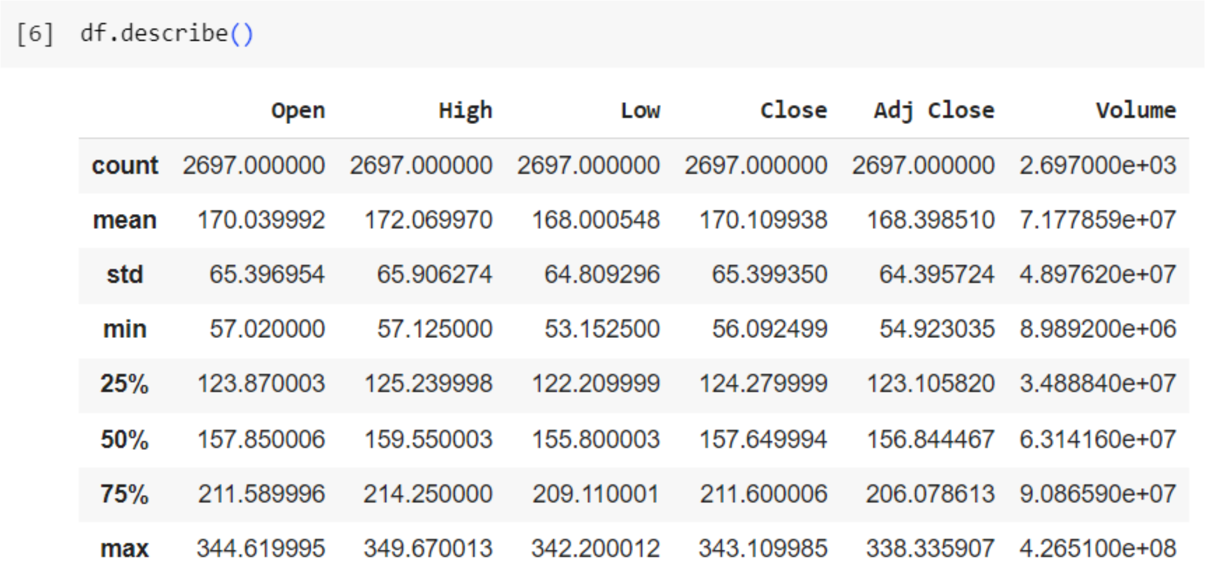
```
[ ]  df
```

	Open	High	Low	Close	Adj Close	Volume	company_name
Date							
2019-11-01	62.384998	63.982498	62.290001	63.955002	62.286350	151125200	APPLE
2019-11-04	64.332497	64.462502	63.845001	64.375000	62.695385	103272000	APPLE
2019-11-05	64.262497	64.547501	64.080002	64.282501	62.605305	79897600	APPLE
2019-11-06	64.192497	64.372498	63.842499	64.309998	62.632080	75864400	APPLE
2019-11-07	64.684998	65.087502	64.527496	64.857498	63.354935	94940400	APPLE
...	...	...	...	...	...	...	...
2023-05-23	114.269997	117.139999	113.779999	114.989998	114.989998	67576300	AMAZON
2023-05-24	115.349998	117.339996	115.019997	116.750000	116.750000	63487900	AMAZON
2023-05-25	116.629997	116.870003	114.309998	115.000000	115.000000	66496700	AMAZON
2023-05-26	116.040001	121.500000	116.019997	120.110001	120.110001	96779900	AMAZON
2023-05-30	122.370003	122.919998	119.860001	121.660004	121.660004	64314800	AMAZON

2697 rows × 7 columns

```
[3]  df.shape

     (2697, 7)

[4]  df.columns

     Index(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume', 'company_name'], dtype='object')

[5]  df.dtypes

     Open            float64
     High            float64
     Low             float64
     Close           float64
     Adj Close       float64
     Volume            int64
     company_name     object
     dtype: object
```

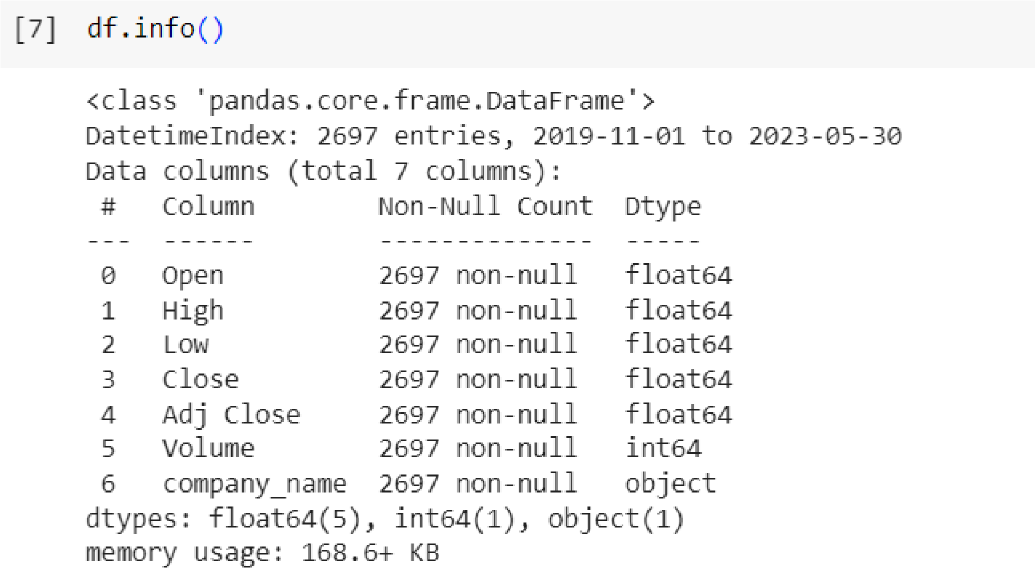
- The pandas DataFrame with the combined stock data for the three tech giants Apple (AAPL), Microsoft (MSFT), and Amazon (AMZN) is represented by the variable 'df'. The individual DataFrames of each organisation are combined to produce it.
- A tuple indicating the dimensions of the DataFrame df is returned by the code segment df.shape. The first value is the number of rows, and the second value is the number of columns.
- With the first value being the number of rows and the second being the number of columns, the code segment df.shape produces a tuple that represents the dimensions of the DataFrame df.
- The DataFrame df's column data types are represented by Series object by the code segment df.dtypes. It describes the kind of data that is kept in each column, including whether it is a float, integer, date, or object.



```
[6] df.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	2697.000000	2697.000000	2697.000000	2697.000000	2697.000000	2.697000e+03
mean	170.039992	172.069970	168.000548	170.109938	168.398510	7.177859e+07
std	65.396954	65.906274	64.809296	65.399350	64.395724	4.897620e+07
min	57.020000	57.125000	53.152500	56.092499	54.923035	8.989200e+06
25%	123.870003	125.239998	122.209999	124.279999	123.105820	3.488840e+07
50%	157.850006	159.550003	155.800003	157.649994	156.844467	6.314160e+07
75%	211.589996	214.250000	209.110001	211.600006	206.078613	9.086590e+07
max	344.619995	349.670013	342.200012	343.109985	338.335907	4.265100e+08

- The code segment df.describe() generates descriptive statistics of the numerical columns in the DataFrame df. It provides statistical measures such as count, mean, standard deviation, minimum, quartiles, and maximum values, giving an overview of the distribution and summary statistics of the data.



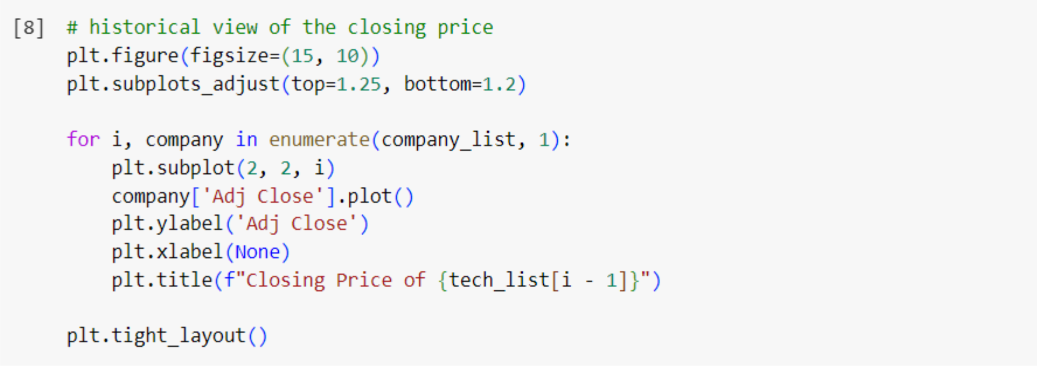
```
[7] df.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2697 entries, 2019-11-01 to 2023-05-30
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Open          2697 non-null   float64
 1   High          2697 non-null   float64
 2   Low           2697 non-null   float64
 3   Close         2697 non-null   float64
 4   Adj Close     2697 non-null   float64
 5   Volume        2697 non-null   int64  
 6   company_name  2697 non-null   object 
dtypes: float64(5), int64(1), object(1)
memory usage: 168.6+ KB
```

The code segment df.info() provides summary of the DataFrame df by displaying concise information about the DataFrame's columns. It includes the column names, the number of non-null values in each column, and the data type of each column. It also provides information about the total number of entries (rows) in the DataFrame. This method is useful for quickly understanding the structure and completeness of the data.

DATA ANALYSIS:

Closing Price:

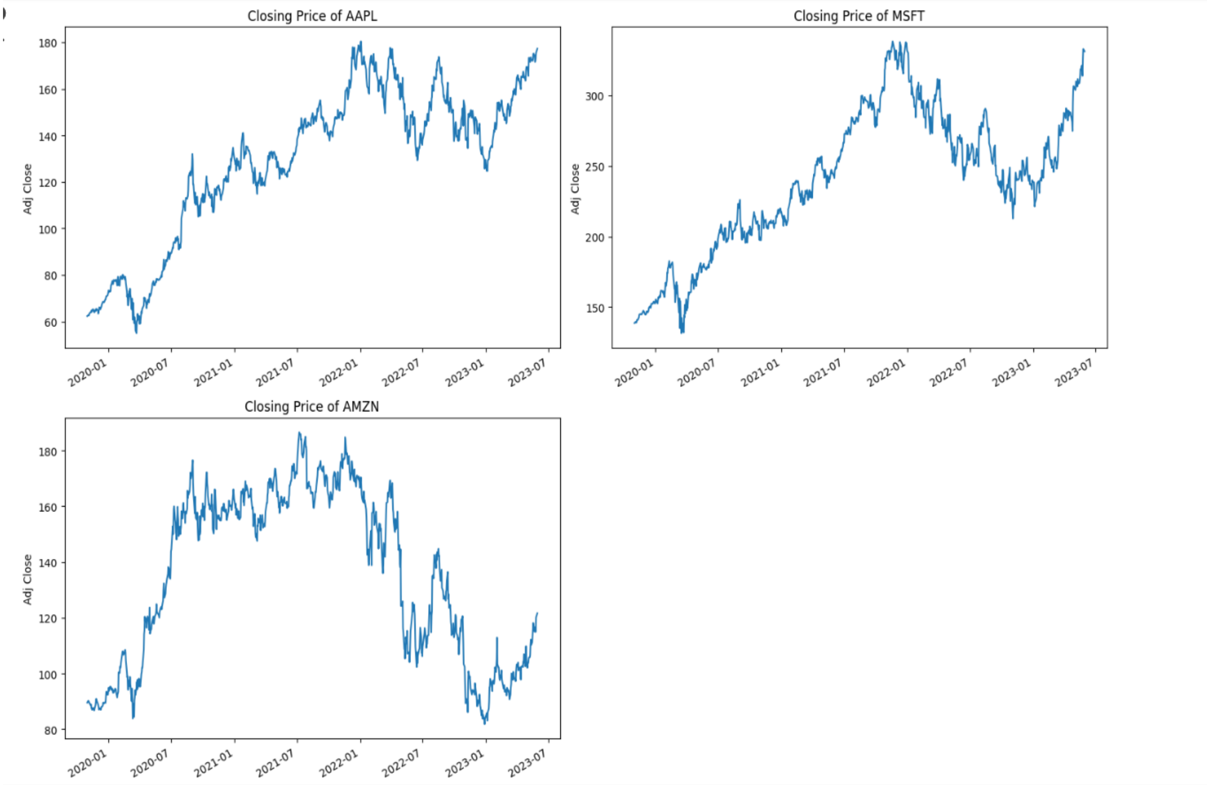
The last price at which the stock is exchanged during a standard trading day is known as the closing price. The common benchmark used by investors to monitor a stock's performance over time is its closing price.



```
[8]  # historical view of the closing price
     plt.figure(figsize=(15, 10))
     plt.subplots_adjust(top=1.25, bottom=1.2)

     for i, company in enumerate(company_list, 1):
         plt.subplot(2, 2, i)
         company['Adj Close'].plot()
         plt.ylabel('Adj Close')
         plt.xlabel(None)
         plt.title(f"Closing Price of {tech_list[i - 1]}")

     plt.tight_layout()
```

The last price at which the stock is exchanged during a standard trading day is known as the closing price. The common benchmark used by investors to monitor a stock's performance over time is its closing price.

For each tech business in the company_list (AAPL, MSFT, AMZN), the code section builds a figure with subplots to show the closing price over time.



Closing Price of AAPL

Closing Price of MSFT

Closing Price of AMZN

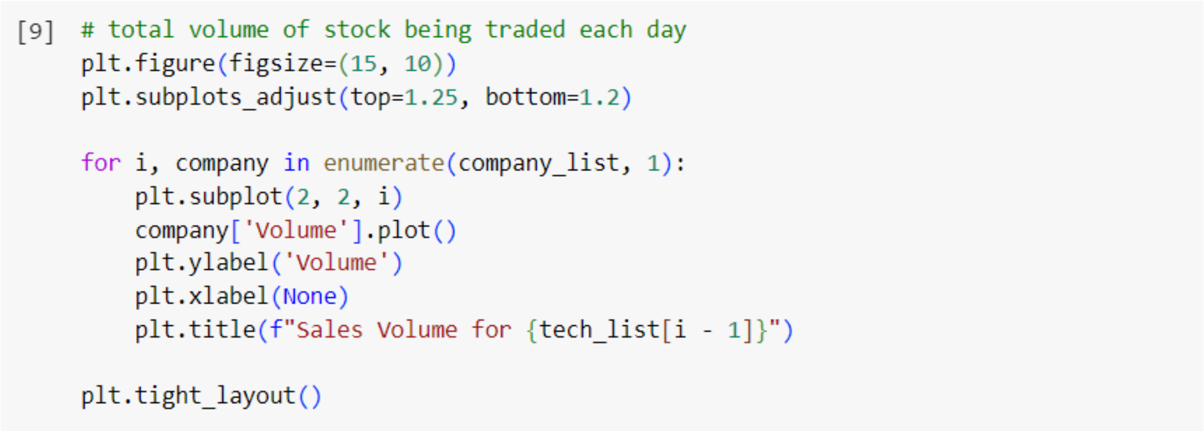
The code is broken down as follows:

- Figure Setup:
  The plt.figure(figsize=(15, 10)) line sets size of the overall figure to 15 inches in width and 10 inches in height.
  plt.subplots_adjust(top=1.25, bottom=1.2) adjusts spacing between the subplots to avoid overlap.
- Subplot Loop:
  A loop is used to iterate over each company in the company_list.
  plt.subplot(2, 2, i) creates a subplot grid with 2 rows and 2 columns and selects the i-th subplot for the current iteration.
  company['Adj Close'].plot() plots the adjusted closing prices for the current company on the selected subplot.
  plt.ylabel('Adj Close') sets the y-axis label as "Adj Close" to represent the adjusted closing price.
  plt.xlabel(None) removes the x-axis label.
  plt.title(f"Closing Price of {tech_list[i - 1]}") sets the title for the current subplot, indicating the closing price and the corresponding tech company.
- Layout Adjustment:
  plt.tight_layout() adjusts the spacing between subplots to improve readability and prevent overlap.

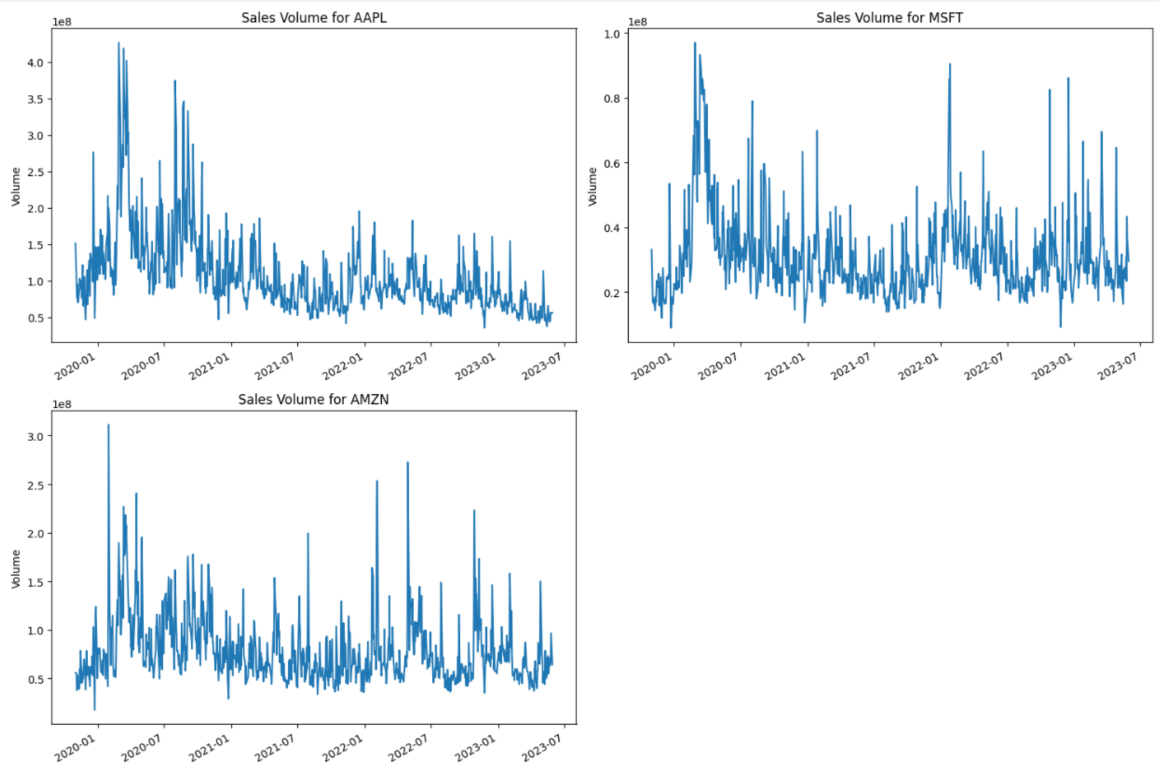
Overall, the code segment generates graph of the historical closing prices for each tech company, allowing for easy comparison and analysis of their price trends over time.

Volume of Sales:

Volume is the total quantity of a security or asset that is traded over time, frequently in a single day. For instance, the number of shares of a securities traded between its daily open and close would be referred to as the stock trading volume. Technical traders need key information like trading volume and variations in volume over time.



```
[9]  # total volume of stock being traded each day
     plt.figure(figsize=(15, 10))
     plt.subplots_adjust(top=1.25, bottom=1.2)

     for i, company in enumerate(company_list, 1):
         plt.subplot(2, 2, i)
         company['Volume'].plot()
         plt.ylabel('Volume')
         plt.xlabel(None)
         plt.title(f"Sales Volume for {tech_list[i - 1]}")

     plt.tight_layout()
```



To visualise the total daily volume of stock traded for each tech business in the company_list (AAPL, MSFT, AMZN), the code segment generates a figure with subplots.
Here is a breakdown of the code:

- Figure Setup:
  Similar to the previous code segment, this sets up the figure size and adjusts the spacing between subplots.

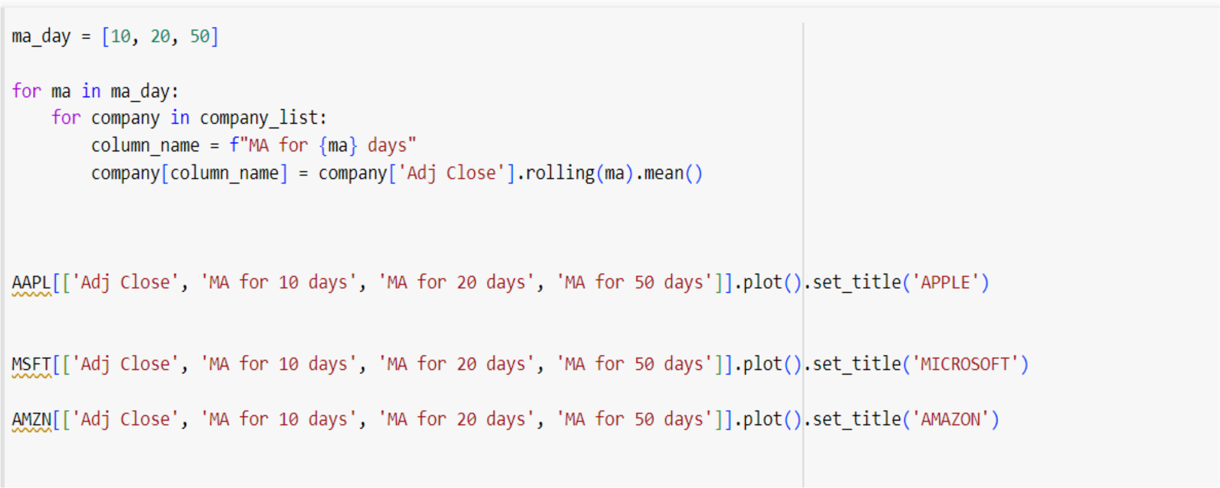
- Subplot Loop:
  A loop is used to iterate over each company in the company_list.
  plt.subplot(2, 2, i) creates a subplot grid with 2 rows and 2 columns and selects the i-th subplot for the current iteration.
  company['Volume'].plot() plots the volume of stock traded each day for the current company on the selected subplot.
  plt.ylabel('Volume') sets the y-axis label as "Volume" to represent the trading volume.
  plt.xlabel(None) removes the x-axis label.
  plt.title(f"Sales Volume for {tech_list[i - 1]}") sets the title for the current subplot, indicating the sales volume and the corresponding tech company.

- Layout Adjustment:
  plt.tight_layout() adjusts the spacing between subplots.

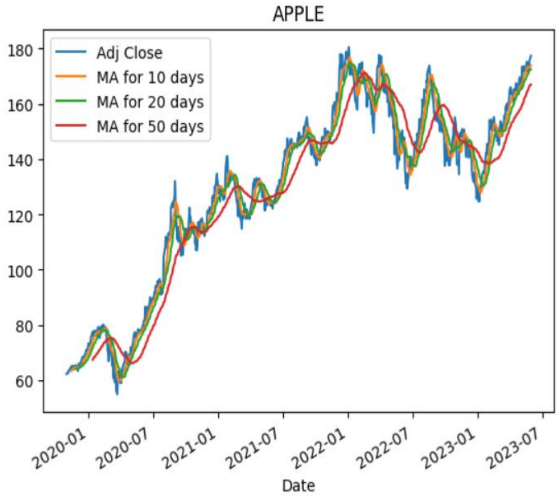
Each tech company's daily trading volume is represented visually by this code section, enabling comparison and study of volume patterns over time.

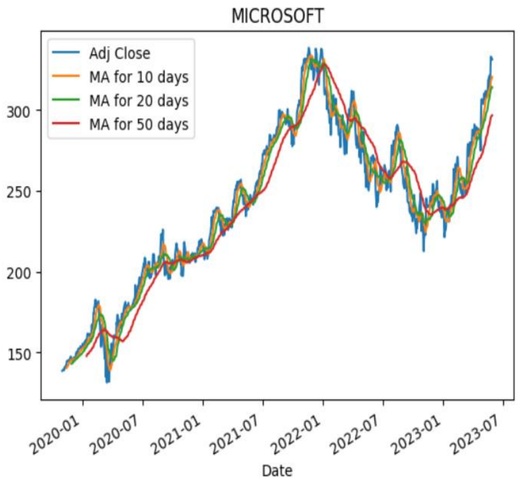
Moving average of the various stocks:

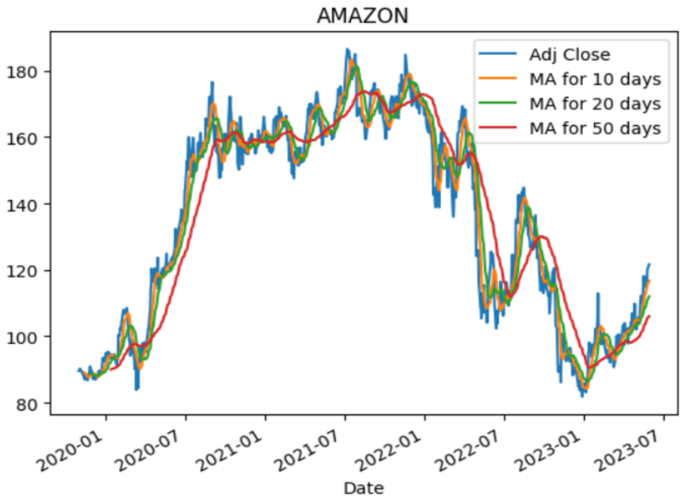
The moving average (MA) is a straightforward technical analysis technique that creates a continuously updated average price to smooth out price data. The average is calculated over a predetermined time frame, such as 10 days, 20 minutes, 30 weeks, or any other time frame the trader specifies.



```
ma_day = [10, 20, 50]

for ma in ma_day:
    for company in company_list:
        column_name = f"MA for {ma} days"
        company[column_name] = company['Adj Close'].rolling(ma).mean()



AAPL[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot().set_title('APPLE')


MSFT[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot().set_title('MICROSOFT')

AMZN[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot().set_title('AMAZON')
```







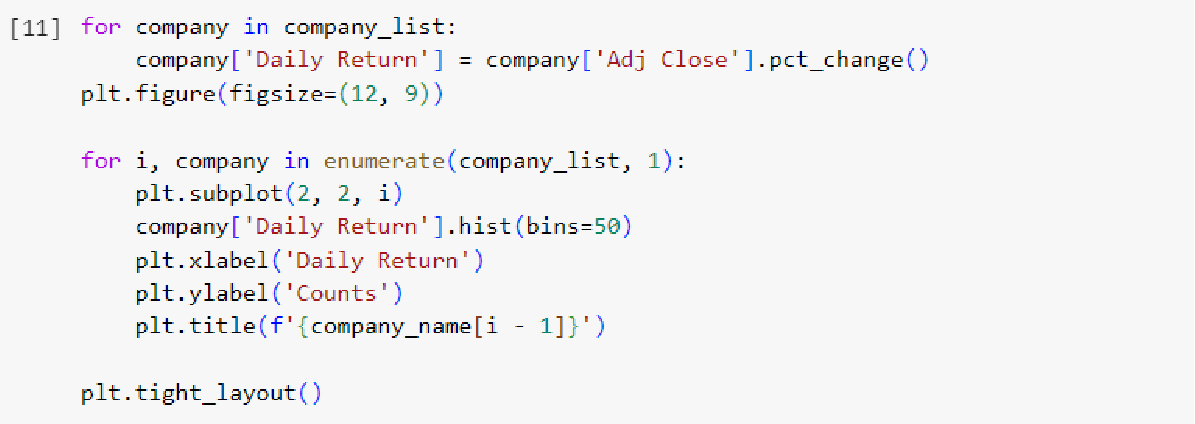
The code segment calculates and plots the moving averages (MA) for different time periods (10, 20, and 50 days) for the adjusted close prices of three tech companies: Apple (AAPL), Microsoft (MSFT), and Amazon (AMZN).
Here is a breakdown of the code:

- Moving Average Calculation:
  A list ma_day is defined with the desired moving average time periods (10, 20, and 50 days).
  Each time period (ma) and each business in the company_list are iterated through by nested loops.
  For each moving average, a new column called column_name is generated with the word "days" in the name.
  The moving average values are calculated for the 'Adj Close' column of each firm using the rolling() function and the given time period (ma).
- Plotting:
  For each company, a plot is created using plot() function on the DataFrame containing the adjusted close prices and the corresponding moving averages.
  The desired columns ('Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days') are selected for plotting.
  set_title() is used to set the title of each plot, indicating the company name ('APPLE', 'MICROSOFT', 'AMAZON').

This code section creates unique plots for each firm that display the adjusted close prices and moving averages for various time periods, enabling visual examination of moving average patterns and crossovers.

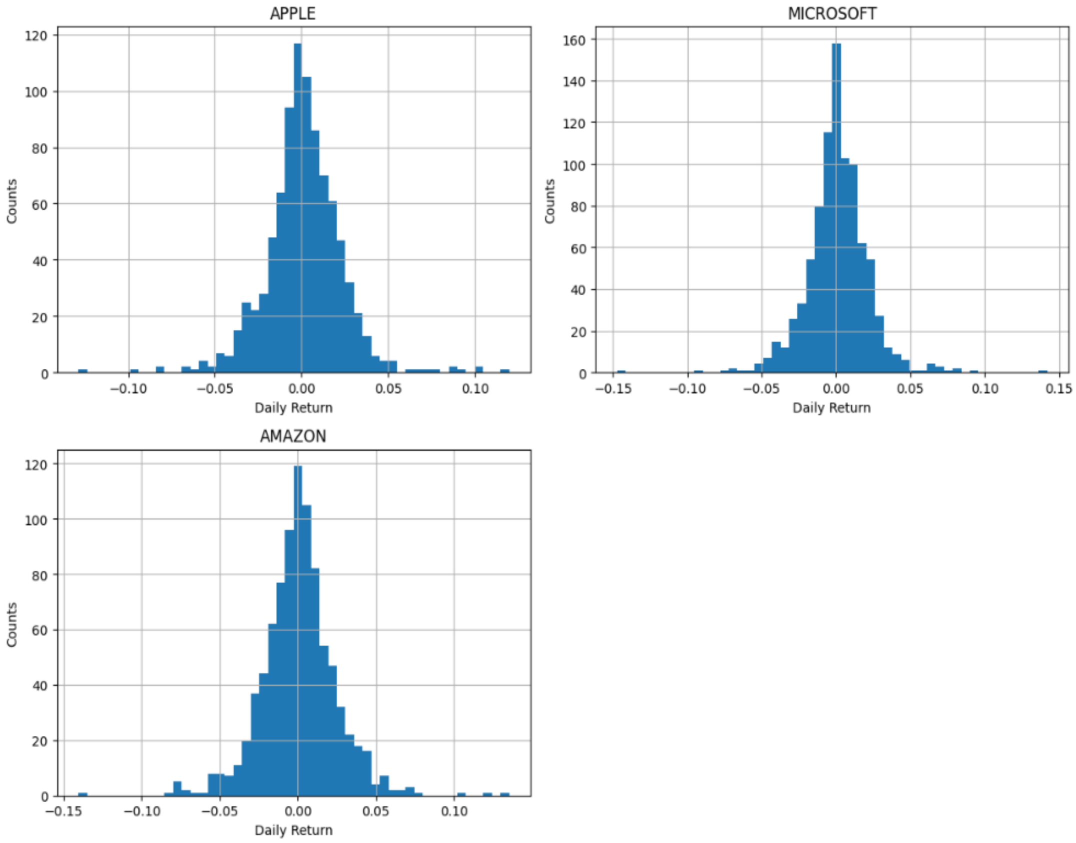
Daily return of the stock on average:

The daily return of a stock on average refers to the average change in the stock's price from one day to the next, expressed as a percentage. It provides an indication of the stock's performance on a daily basis, allowing investors to assess the average daily profit or loss potential of the stock.



```
[11] for company in company_list:
         company['Daily Return'] = company['Adj Close'].pct_change()
     plt.figure(figsize=(12, 9))

     for i, company in enumerate(company_list, 1):
         plt.subplot(2, 2, i)
         company['Daily Return'].hist(bins=50)
         plt.xlabel('Daily Return')
         plt.ylabel('Counts')
         plt.title(f'{company_name[i - 1]}')

     plt.tight_layout()
```

By calculating the percentage change in the adjusted close prices from one day to the next, the code segment determines the daily return of each stock in the company_list. After that, a histogram is produced to show how each company's daily returns are distributed.

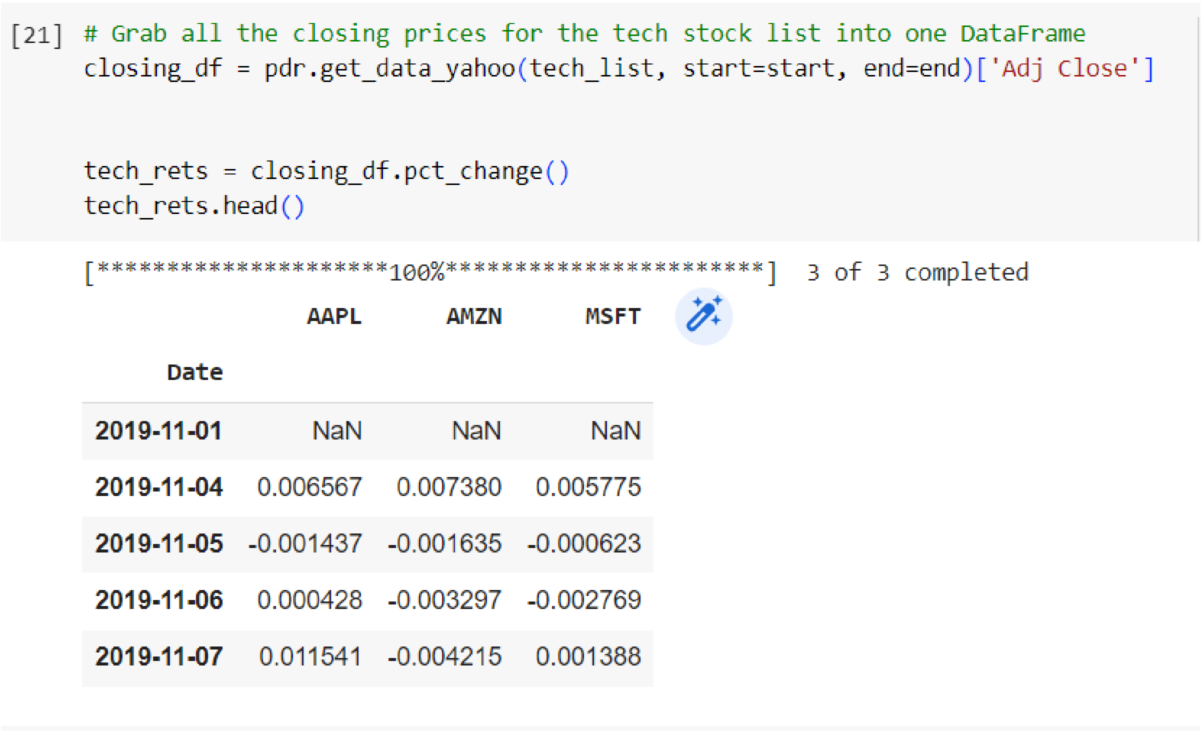


The code is explained as follows:

- Daily Return Calculation:
  Every firm in the company_list is iterated through in the for loop.
  The pct_change() method is used on the 'Adj Close' column to construct the 'Daily Return' column for each firm.
- Histogram Plotting:
  The plt.figure(figsize=(12, 9)) line sets the size the overall figure.
  Another for loop is utilized to iterate over each company.
  plt.subplot(2, 2, i) creates a subplot grid with 2 rows and 2 columns and selects the i-th subplot for the current iteration.
  company['Daily Return'].hist(bins=50) plots a histogram of the daily returns for the current company with 50 bins.
  plt.xlabel('Daily Return') sets the x-axis label as "Daily Return".
  plt.ylabel('Counts') sets the y-axis label as "Counts".
  plt.title(f'{company_name[i - 1]}') sets the title for the current subplot, indicating the name of the company.
- Layout Adjustment:
  plt.tight_layout() adjusts the spacing between subplots.
  This code segment provides a visual representation of the distribution of daily returns for each company, allowing for analysis of the volatility and risk associated with their stock prices.

Correlation between different stocks closing prices?

A correlation statistic, whose value must fall between -1.0 and +1.0, quantifies how much two variables change in relation to one another. Correlation quantifies correlation but cannot determine whether x causes y or vice versa, or whether a third component is responsible for the association.



```
[21] # Grab all the closing prices for the tech stock list into one DataFrame
     closing_df = pdr.get_data_yahoo(tech_list, start=start, end=end)['Adj Close']


     tech_rets = closing_df.pct_change()
     tech_rets.head()
```

```
[*********************100%***********************]  3 of 3 completed
```

	AAPL	AMZN	MSFT
Date			
2019-11-01	NaN	NaN	NaN
2019-11-04	0.006567	0.007380	0.005775
2019-11-05	-0.001437	-0.001635	-0.000623
2019-11-06	0.000428	-0.003297	-0.002769
2019-11-07	0.011541	-0.004215	0.001388

The code segment acquires the adjusted closing prices of the tech stocks mentioned in the tech_list using the pdr.get_data_yahoo() function. The prices are determined for the chosen start and finish dates. The resulting DataFrame, closing_df, contains the closing prices for each stock.

The percentage change in the changed closing prices is then calculated using the closing_df's pct_change() function. A new DataFrame called tech_rets was constructed throughout this procedure to reflect the daily returns of the tech stocks.

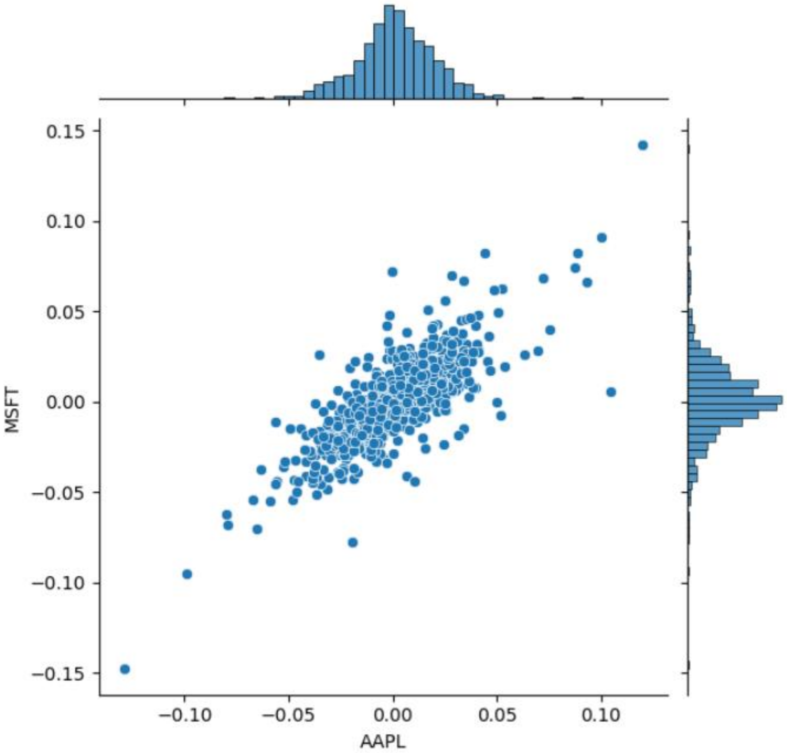
The head() function is then used to display the first few rows of the tech_rets DataFrame to provide a summary of the anticipated daily returns for the tech stocks.

Joint Plots:

Using scatter plots, histograms, or kernel density estimations, the jointplot() function in Seaborn may combine univariate and bivariate plots to show the connection between two variables.

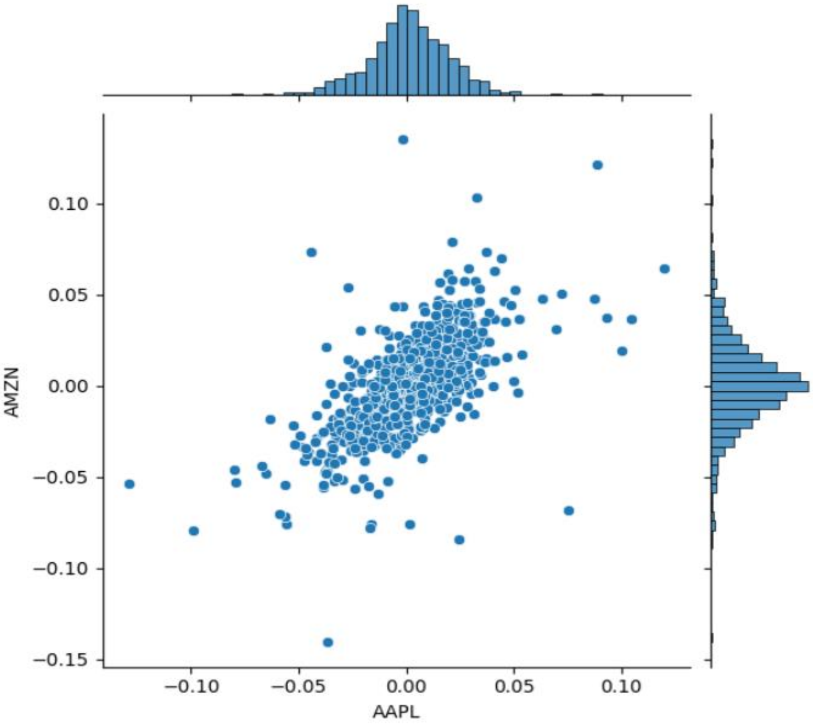
```
[13] sns.jointplot(x='AAPL', y='MSFT', data=tech_rets, kind='scatter')
```

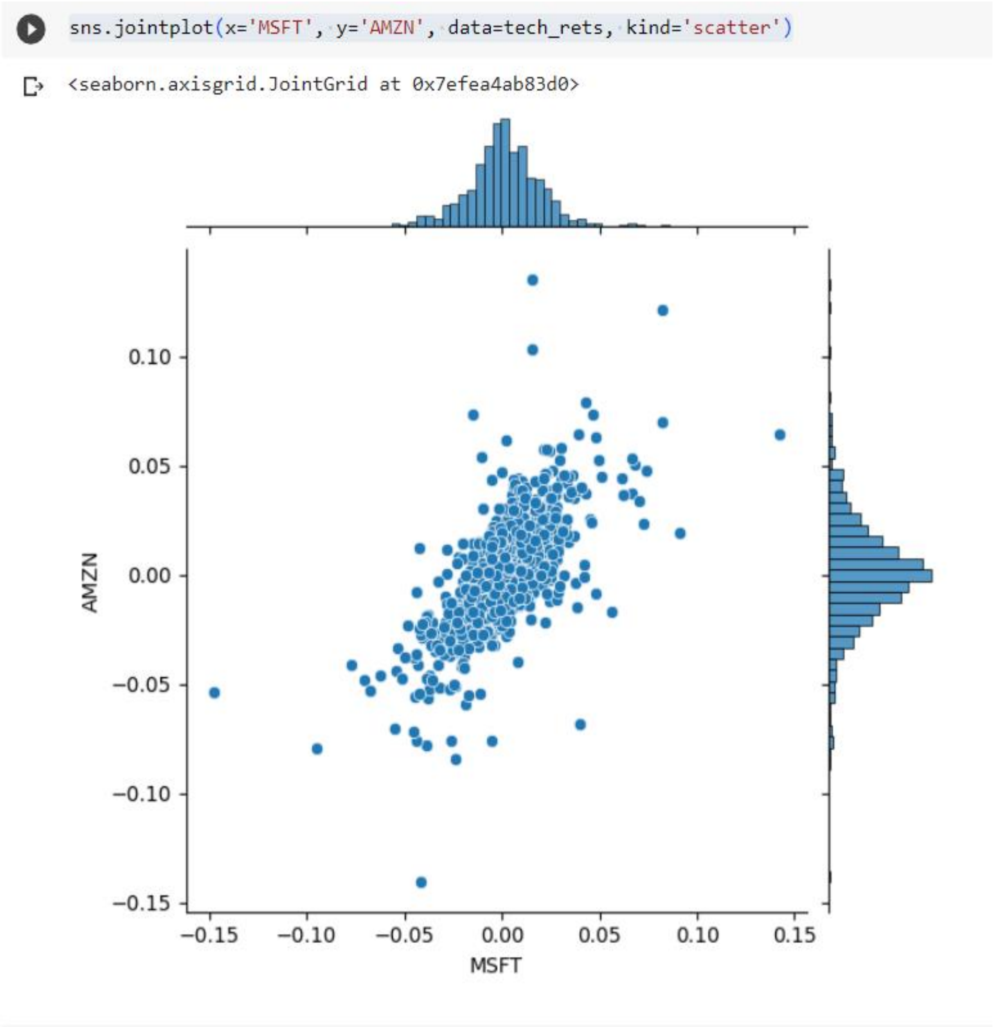
```
<seaborn.axisgrid.JointGrid at 0x7efeec49d150>
```



```
[14] sns.jointplot(x='AAPL', y='AMZN', data=tech_rets, kind='scatter')
```

```
<seaborn.axisgrid.JointGrid at 0x7efeec49db40>
```





```
sns.jointplot(x='MSFT', y='AMZN', data=tech_rets, kind='scatter')
```

```
<seaborn.axisgrid.JointGrid at 0x7efea4ab83d0>
```

The code segment uses Seaborn's jointplot() function to create scatter plots between the daily returns of different pairs of tech stocks.

Here is an explanation of the jointplot() between APPLE-MICROSOFT , APPLE-AMAZON and MICROSOFT-AMAZON :

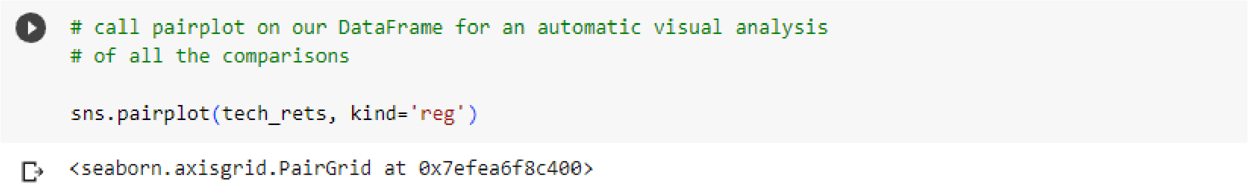
- x='AAPL' and y='MSFT' create a scatter plot between the daily returns of Apple and Microsoft stocks.
- x='AAPL' and y='AMZN' create a scatter plot between the daily returns of Apple and Amazon stocks.
- x='MSFT' and y='AMZN' create a scatter plot between the daily returns of Microsoft and Amazon stocks.

kind='scatter' specifies the type of plot as a scatter plot.

These code segments generate scatter plots that visualize the relationships and correlations between the daily returns of different pairs of tech stocks. They help analyse the co-movements and interactions between the stock returns of these companies.

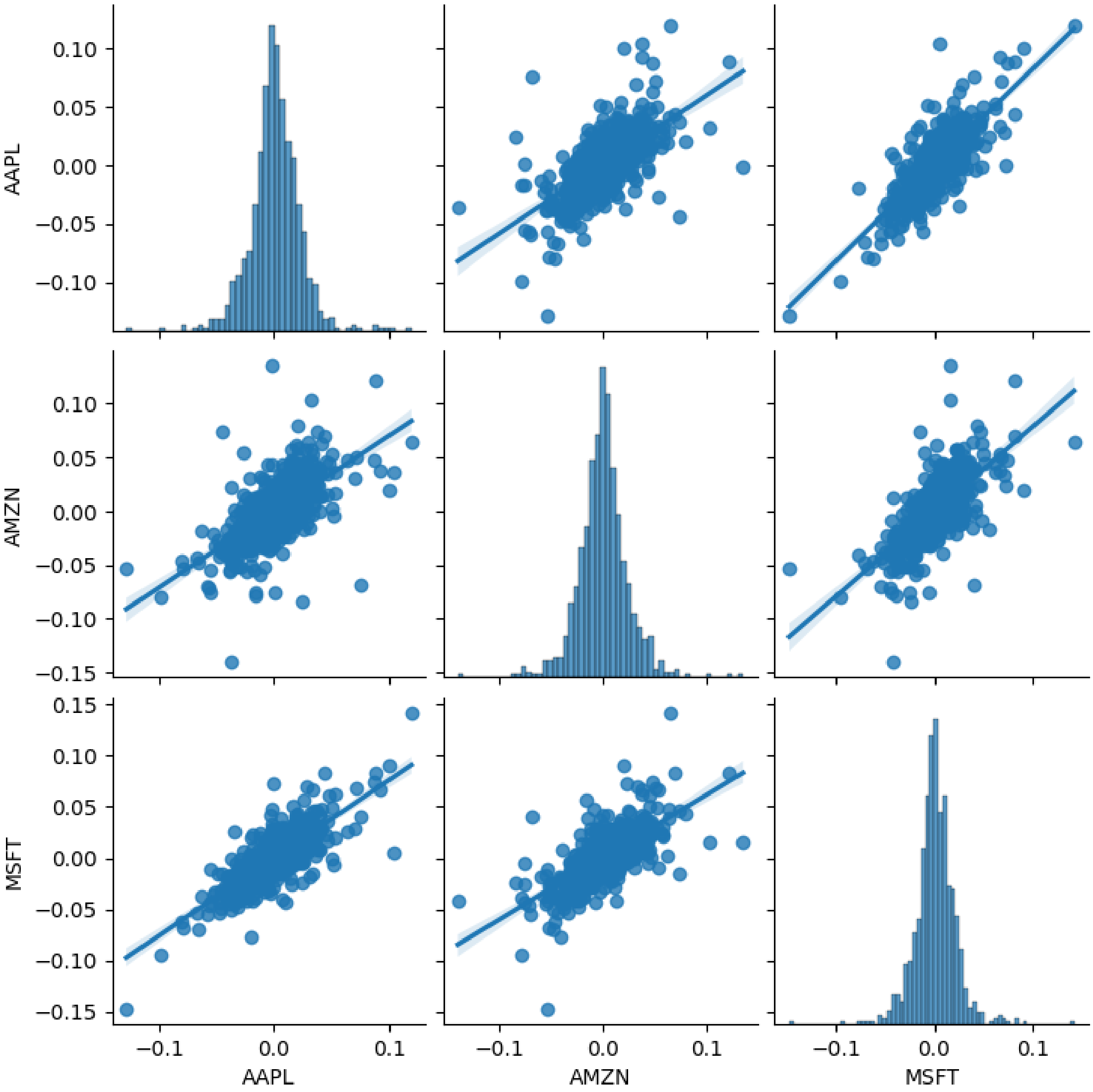
Pair Plots:

pairplot() in Seaborn is a function used to create a grid of scatter plots and histograms to visualize the pairwise relationships between multiple variables in a dataset, allowing for quick analysis of correlations and distributions.



```
# call pairplot on our DataFrame for an automatic visual analysis
# of all the comparisons

sns.pairplot(tech_rets, kind='reg')
```

```
<seaborn.axisgrid.PairGrid at 0x7efea6f8c400>
```



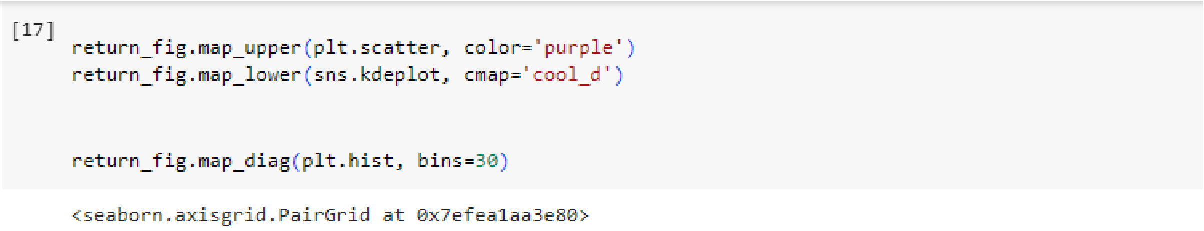
The code segment uses Seaborn's pairplot() function to create a grid of scatter plots and histograms for visual analysis of the pairwise relationships within the tech_rets DataFrame. Here is an explanation of the code:

- tech_rets is the DataFrame containing the daily returns of the tech stocks.
- kind='reg' instructs the grid's plots to display regression lines as a representation of the linear connections between the variables.

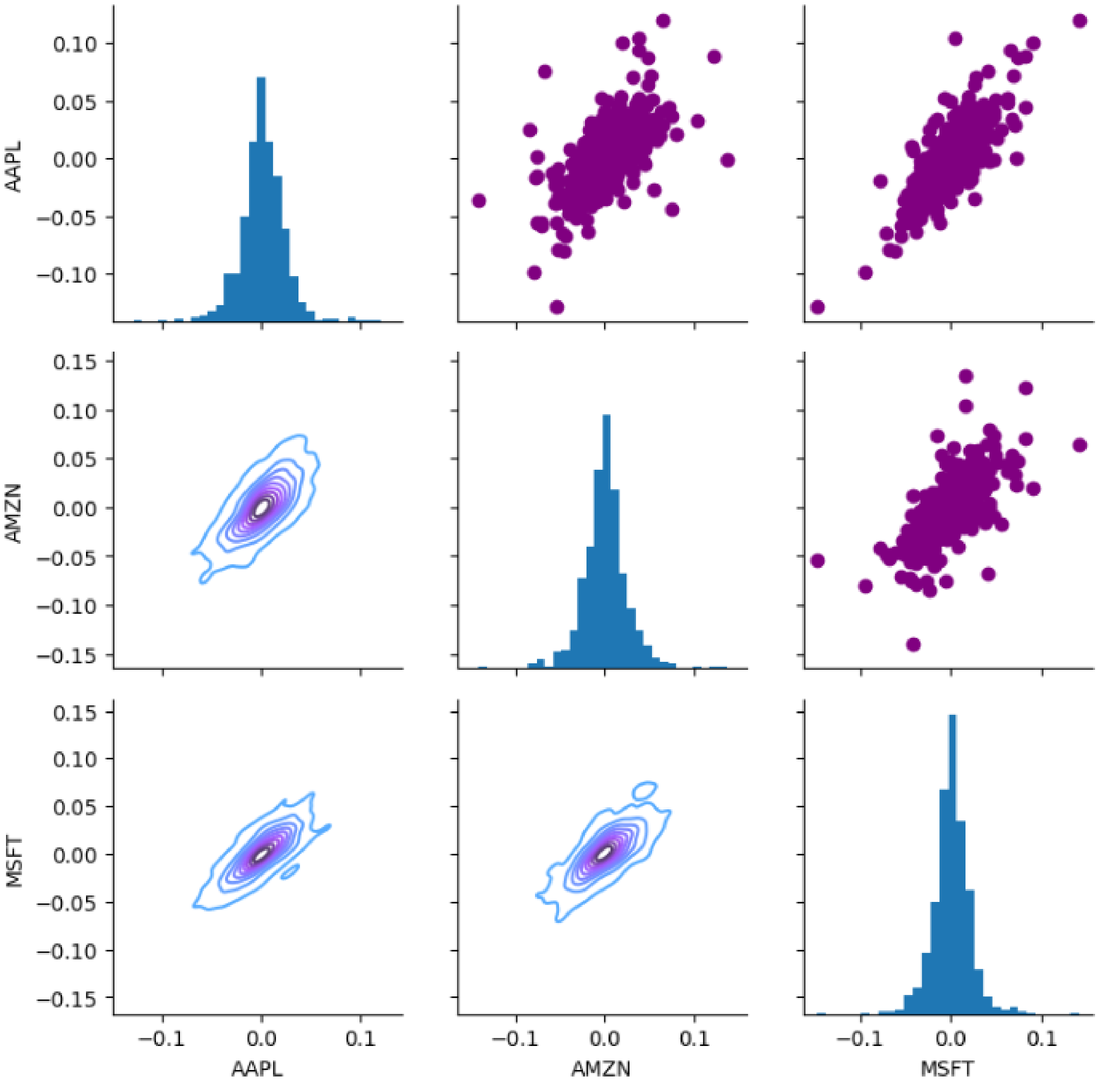
Using a grid of scatter plots produced by the pairplot() function, each variable in the DataFrame is compared to every other variable. The graphs on the diagonal show the histograms for each variable. The off-diagonal plots display the scatter plots between two variables.

In order to better understand the general trend and strength of the linear relationship between the variables, regression lines are also added to the scatter plots using the kind='reg' option.

This code section offers a complete visual study of the correlations and distributions between the daily returns of the tech stocks, aiding in the research of their connections and likely patterns.



```
[17]
     return_fig.map_upper(plt.scatter, color='purple')
     return_fig.map_lower(sns.kdeplot, cmap='cool_d')


     return_fig.map_diag(plt.hist, bins=30)

     <seaborn.axisgrid.PairGrid at 0x7efea1aa3e80>
```

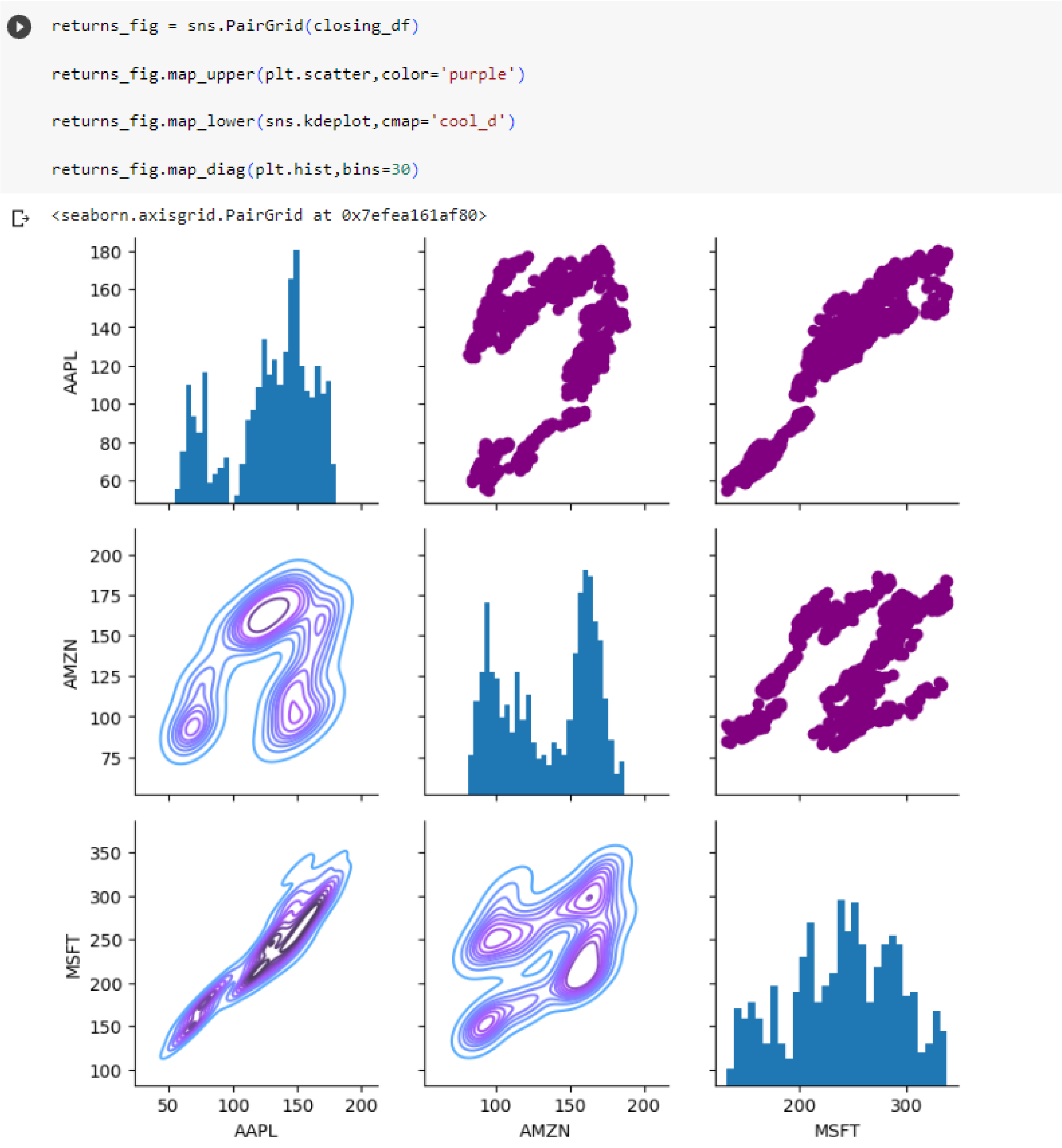


The code segment sets up a figure named returns_fig and uses Seaborn's PairGrid function on the tech_rets DataFrame with missing values dropped.

The code is explained as follows:

- return_fig = sns.PairGrid(tech_rets.dropna()) creates a PairGrid object, which is a grid of subplots for pairwise relationships.
- return_fig.map_upper(plt.scatter, color='purple') maps the upper triangle of the grid to scatter plots with purple dots.
- return_fig.map_lower(sns.kdeplot, cmap='cool_d') maps the lower triangle of the grid to kernel density estimation (KDE) plots using a cool color map.
- return_fig.map_diag(plt.hist, bins=30) maps the diagonal of the grid to histogram plots of the daily returns with 30 bins.

A graphic with a grid of subplots illustrating the pairwise relationships between the daily returns of the tech stocks is produced by the code section. Scatter plots are displayed in the upper triangle, KDE plots in the lower triangle, and histograms are displayed in the diagonal. It offers a thorough visual study of the daily returns' distribution and correlations.



```
returns_fig = sns.PairGrid(closing_df)

returns_fig.map_upper(plt.scatter,color='purple')

returns_fig.map_lower(sns.kdeplot,cmap='cool_d')

returns_fig.map_diag(plt.hist,bins=30)
```

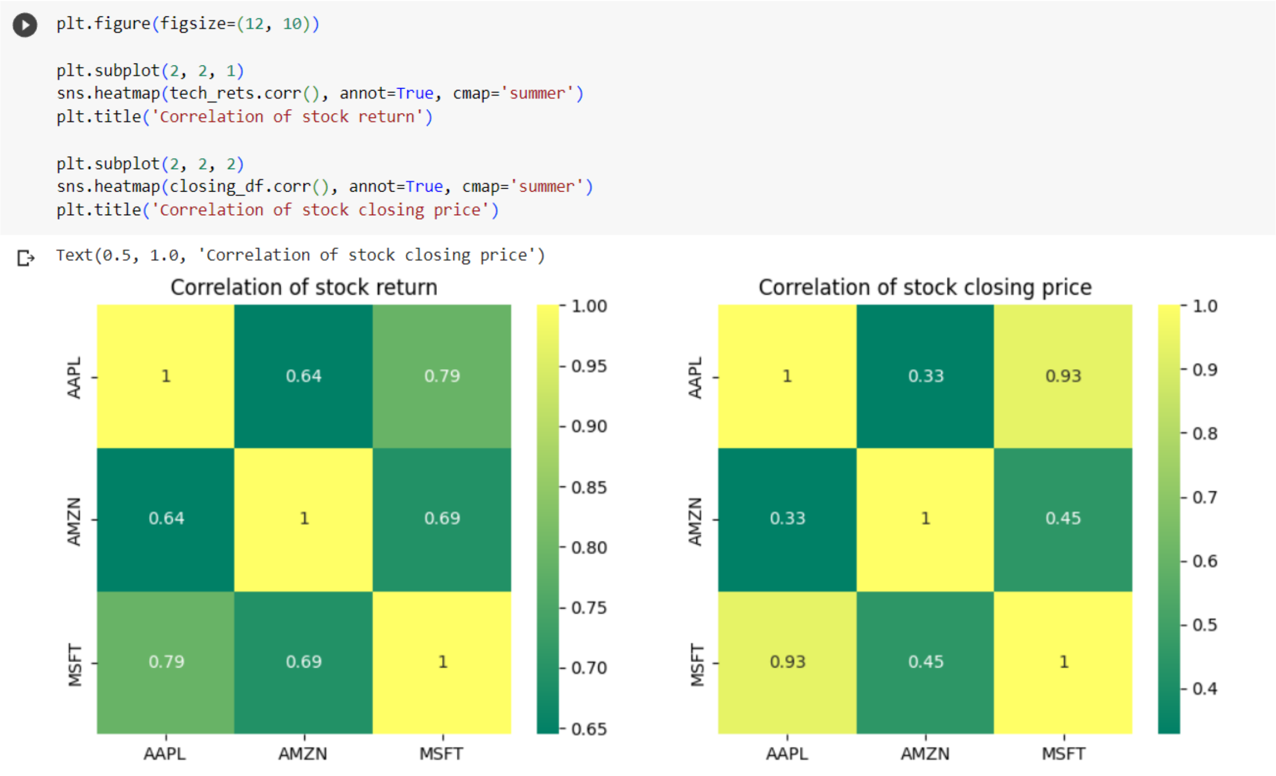
```
<seaborn.axisgrid.PairGrid at 0x7efea161af80>
```

The code segment sets up a figure named returns_fig and uses Seaborn's PairGrid function on the closing_df DataFrame.

The code is explained as follows:

- returns_fig = sns.PairGrid(closing_df) creates a PairGrid object, which is a grid of subplots for pairwise relationships.
- returns_fig.map_upper(plt.scatter, color='purple') maps to upper triangle of the grid to scatter plots with purple dots.
- returns_fig.map_lower(sns.kdeplot, cmap='cool_d') maps to lower triangle of the grid to kernel density estimation (KDE) plots using a cool color map.
- returns_fig.map_diag(plt.hist, bins=30) maps the diagonal of the grid to histogram plots of the daily returns with 30 bins.

The code part generates a graph with a grid of subplots showing the pairwise associations between the closing prices of the tech stocks. The upper triangle shows scatter plots, the lower triangle shows KDE plots, and the diagonal shows histograms. Visual analysis is done on the distribution and correlations between the closing prices of the tech stocks.



```
plt.figure(figsize=(12, 10))

plt.subplot(2, 2, 1)
sns.heatmap(tech_rets.corr(), annot=True, cmap='summer')
plt.title('Correlation of stock return')

plt.subplot(2, 2, 2)
sns.heatmap(closing_df.corr(), annot=True, cmap='summer')
plt.title('Correlation of stock closing price')
```

```
Text(0.5, 1.0, 'Correlation of stock closing price')
```

The above code segment creates a figure with two subplots and uses Seaborn's heatmap() function to visualize the correlation matrices.

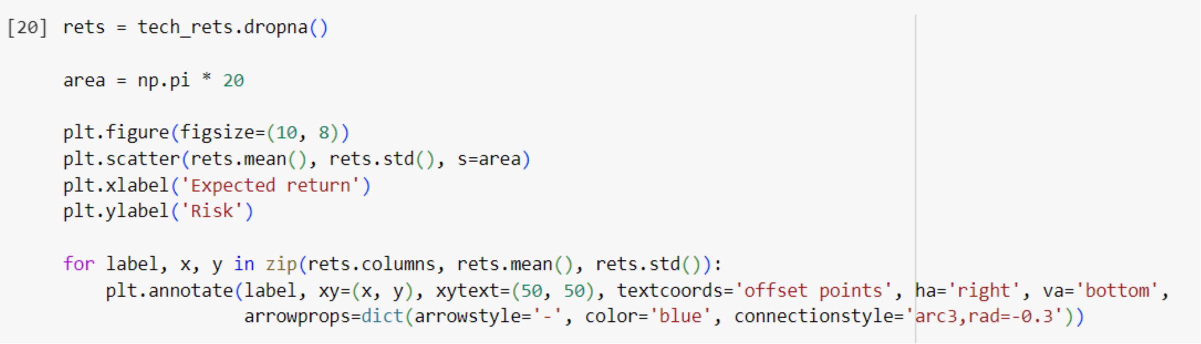
- Here's an explanation of the code:
- plt.figure(figsize=(12, 10)) creates a figure with a specific size of 12x10 inches.
- plt.subplot(2, 2, 1) creates the first subplot in a 2x2 grid.
- sns.heatmap(tech_rets.corr(), annot=True, cmap='summer') generates a heatmap of the correlation matrix for the daily returns of the tech stocks. The correlation values are annotated on the heatmap, and the colormap used is 'summer'.

- plt.title('Correlation of stock return') sets the title for the first subplot.

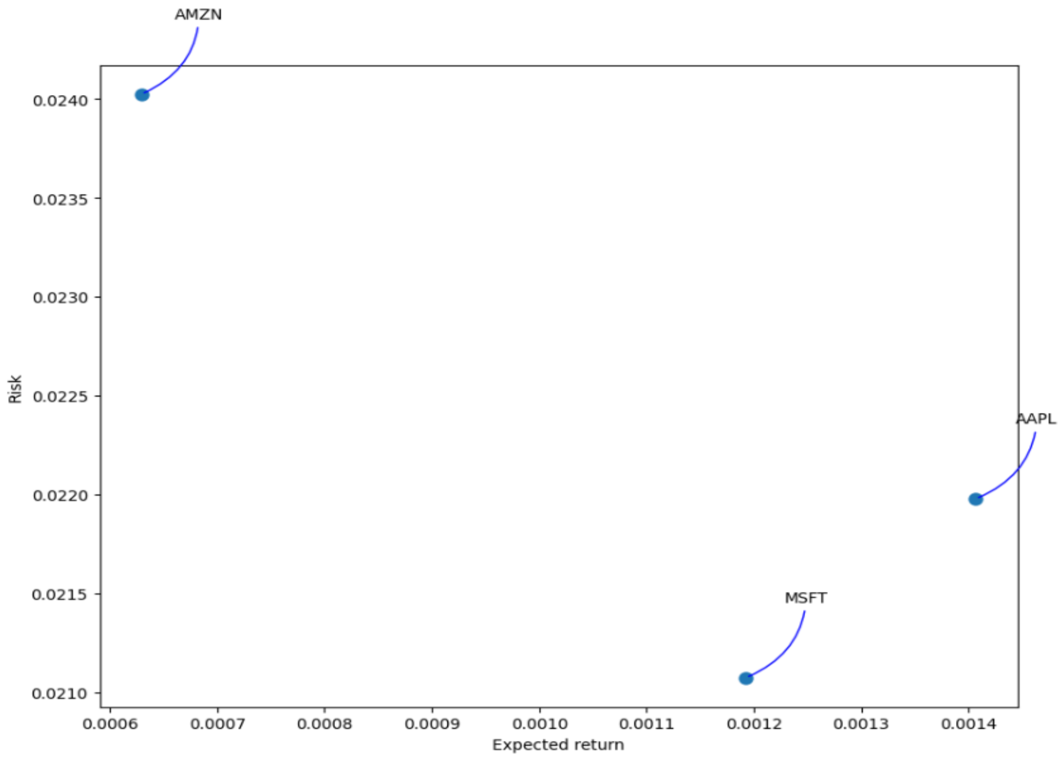
For the second subplot, the same procedure is done, but this time the correlation matrix is based on the closing prices of the tech stocks.

The visual examination of the correlations between stock returns and the correlations between stock closing prices is made possible by this code section. The heatmaps' use of colour gradients to depict the relationships' intensity and direction is obvious.

How much value do we put at risk by investing in a particular stock?



```
[20] rets = tech_rets.dropna()

     area = np.pi * 20

     plt.figure(figsize=(10, 8))
     plt.scatter(rets.mean(), rets.std(), s=area)
     plt.xlabel('Expected return')
     plt.ylabel('Risk')

     for label, x, y in zip(rets.columns, rets.mean(), rets.std()):
         plt.annotate(label, xy=(x, y), xytext=(50, 50), textcoords='offset points', ha='right', va='bottom',
                      arrowprops=dict(arrowstyle='-', color='blue', connectionstyle='arc3,rad=-0.3'))
```



The code segment calculates the expected return and risk (standard deviation) for each tech stock and creates a scatter plot to visualize the relationship between them.

Here is an explanation of the code:

- rets = tech_rets.dropna() drops any rows having missing values from the tech_rets DataFrame and assigns the result to rets.

- area = np.pi * 20 calculates the area for the markers in the scatter plot.
- plt.figure(figsize=(10, 8)) creates a figure with a size of 10x8 inches.
- plt.scatter(rets.mean(), rets.std(), s=area) plots the scatter plot with the expected return on the x-axis and the risk (standard deviation) on the y-axis. The size of the markers is determined by the area variable.
- plt.xlabel('Expected return') and plt.ylabel('Risk') set the x-axis and y-axis labels, respectively.

The stock label is added to each data point by the for loop as it iterates across the columns of rets. The label text, position, and style are all annotated.

The risk-return tradeoff for tech equities is illustrated visually in this code snippet. It assists in identifying equities that, for a given amount of risk, have higher predicted returns and vice versa. About each stock's label, the annotations offer more details.

REFERENCES:

Data Preparation:

- https://www.simplilearn.com/tutorials/machine-learning-tutorial/stock-price-prediction-using-machine-learning
- https://www.analyticsvidhya.com/blog/2021/10/machine-learning-for-stock-market-prediction-with-step-by-step-implementation/

Kaggle:

- https://www.kaggle.com/code/faressayah/stock-market-analysis-prediction-using-lstm/notebook

Thank You