Myself,

BANKA SUSHANTH KUMAR

E-mail ID :- bankasushanthkumar@gmail.com (mailto:bankasushanthkumar@gmail.com)

LinkedIn :- www.linkedin.com/in/banka-sushanth-kumar-05043004010614092904 (http://www.linkedin.com/in/banka-sushanth-kumar-05043004010614092904)

Business Scenario: Data Science extracts meaningful insight from chunks of raw data, which is useful to different business segments for planning their future course of action. Finance is probably one of the first to catch on to this trend with a rise in the penetration of analytics into many aspects of our lives. Here, we will analyze data from the stock market for some technology stocks such as Apple, Google, Amazon, and Microsoft.

Objective: Use Python libraries such as Pandas, Seaborn, and Matplotlib to extract and analyze the information, visualize it, and look at different ways to analyze the risk of a stock, based on its performance history.

About the Data: The stocks we have chosen are from various industries and market caps namely, • Apple • Google • Microsoft • Amazon

For the start, we shall investigate the Amazon stock individually and then move on to the combined analysis.

To know more about stocks and their data please visit the below link:
https://in.finance.yahoo.com/ (https://in.finance.yahoo.com/)

DATASET:- https://drive.google.com/file/d/1P2K6PGLmr2uiN45ZZ9ohu2FsRDPSsIs2/view?usp=share_link (https://drive.google.com/file/d/1P2K6PGLmr2uiN45ZZ9ohu2FsRDPSsIs2/view?usp=share_link)

The following tasks are to be performed:

1.Read the Data from Yahoo finance website directly.

2.Perform cleaning.

3.What was the change in stock price over time?

4.Visualize the change in a stock’s volume being traded, over time?

5.What was the moving average of various stocks?

6.What was the daily return average of a stock?

7.Adda new column ‘Trend’ whose values are based on the 'Daily Return'.

8.Visualize trend frequency through a Pie Chart.

9.What was the correlation between the daily returns of different stocks?

In [1]:
```
# Importing Necessary Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('AMZN.csv') #Importing the Dataset
```

In [3]:
```
df.head() #Getting Top 5 observations of Data
```

Out[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2022-10-24	119.980003	120.389999	116.570000	119.820000	119.820000	49531500
1	2022-10-25	119.650002	121.320000	118.949997	120.599998	120.599998	50934600
2	2022-10-26	116.000000	119.349998	114.760002	115.660004	115.660004	68802300
3	2022-10-27	113.919998	114.120003	109.769997	110.959999	110.959999	129605400
4	2022-10-28	97.910004	103.959999	97.660004	103.410004	103.410004	223133400

What are Open, High, Low, Prev. Close Prices?

Open - Price at which trade of the scrip starts at the beginning of the trade session.

High - Highest price at which the scrip has traded during a trade session.

Low - Lowest price at which the scrip has traded during a trade session.

Close - Price at which the scrip has traded at the end of the last trade session.

In [4]:
```
df.isnull().sum() #there are no null values/ missing values
```

Out[4]:
```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [5]:
```
# Task-3
#To calculate the daily price change, you can use the .diff() function, which compu
df['Price Change'] = df['Close'].diff()
df
```

Out[5]:

	Date	Open	High	Low	Close	Adj Close	Volume	Price Change
0	2022-10-24	119.980003	120.389999	116.570000	119.820000	119.820000	49531500	NaN
1	2022-10-25	119.650002	121.320000	118.949997	120.599998	120.599998	50934600	0.779998
2	2022-10-26	116.000000	119.349998	114.760002	115.660004	115.660004	68802300	-4.939994
3	2022-10-27	113.919998	114.120003	109.769997	110.959999	110.959999	129605400	-4.700005
4	2022-10-28	97.910004	103.959999	97.660004	103.410004	103.410004	223133400	-7.549995
...	...	...	...	...	...	...	...	...
245	2023-10-16	130.690002	133.070007	130.429993	132.550003	132.550003	42832900	2.760010

The diff() function computes the difference between consecutive values, and the result will be NaN for the first row, as there's no previous day's data to compare with. You can choose to drop or handle these NaN values as needed. I'm filling the NaN value with Average value of price change column.

In [6]:
```
np.mean(df['Price Change'])
```

Out[6]:
```
0.021485935742971942
```

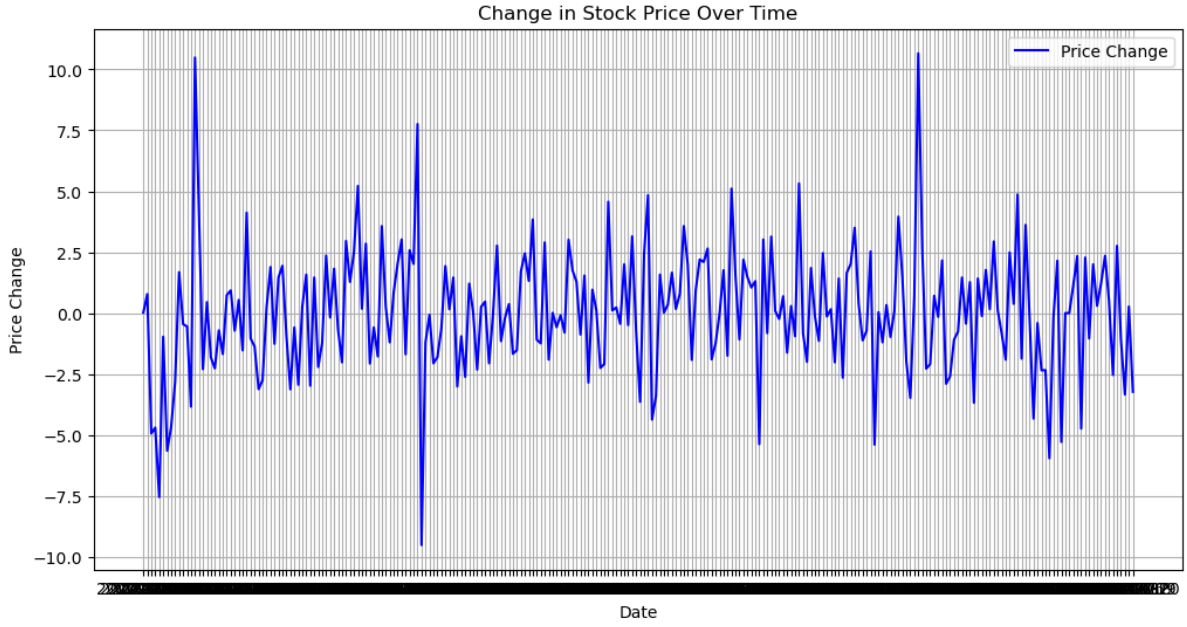
In [7]:
```
df['Price Change'].fillna(0.021485935742971942, inplace= True)
```

In [8]:
```
df.isnull().sum() # So Now there is No null or missing values in dataset
```

Out[8]:
```
Date            0
Open            0
High            0
Low             0
Close           0
Adj Close       0
Volume          0
Price Change    0
dtype: int64
```

In [9]:
```
# Task-4
#What was the change in stock price over time?
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(df['Date'], df['Price Change'], label='Price Change', color='b')
plt.title('Change in Stock Price Over Time')
plt.xlabel('Date')
plt.ylabel('Price Change')
plt.legend()
plt.grid(True)
plt.show()
```



Calculate Moving Average: You can use the rolling function in Pandas to calculate the moving average. Here's an example of calculating a simple moving average (SMA) with a window of 20 days:

Traders use simple moving averages (SMAs) to chart the long-term trajectory of a stock or other security, while ignoring the noise of day-to-day price movements. This allows traders to compare medium- and long-term trends over a larger time horizon.

Which moving average is best profitable?

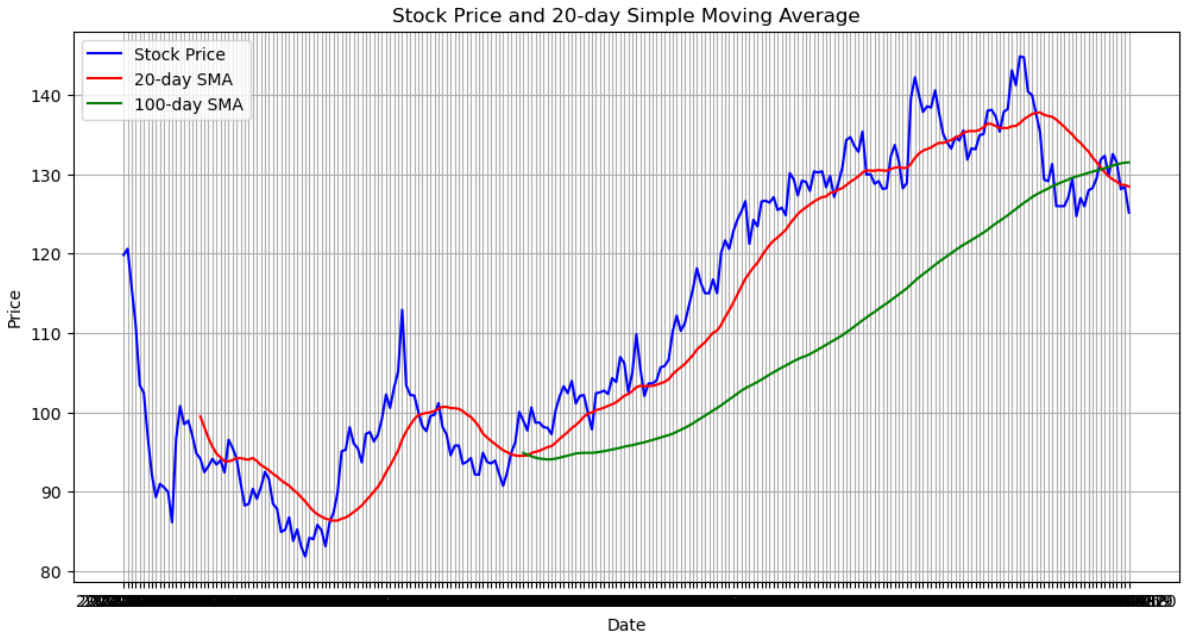
That depends on whether you have a short-term horizon or a long-term horizon. For short-term trades the 5, 10, and 20 period moving averages are best, while longer-term trading makes best use of the 50, 100, and 200 period moving averages.

In [10]:
```
# Task-5
window_size = 20 #This code calculates the 20-day simple moving average for each d
df['SMA_20'] = df['Close'].rolling(window=window_size).mean()
window_size = 100
df['SMA_100'] = df['Close'].rolling(window= window_size).mean()
df
```

	Date	Open	High	Low	Close	Adj Close	Volume	Price Change
0	2022-10-24	119.980003	120.389999	116.570000	119.820000	119.820000	49531500	0.021486
1	2022-10-25	119.650002	121.320000	118.949997	120.599998	120.599998	50934600	0.779998
2	2022-10-26	116.000000	119.349998	114.760002	115.660004	115.660004	68802300	-4.939994
3	2022-10-27	113.919998	114.120003	109.769997	110.959999	110.959999	129605400	-4.700005
4	2022-10-28	97.910004	103.959999	97.660004	103.410004	103.410004	223133400	-7.549995
...	...	...	...	...	...	...	...	...
245	2023-10-16	130.690002	133.070007	130.429993	132.550003	132.550003	42832900	2.760010
	2023-							

In [11]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(df['Date'], df['Close'], label='Stock Price', color='b')
plt.plot(df['Date'], df['SMA_20'], label='20-day SMA', color='r')
plt.plot(df['Date'], df['SMA_100'], label='100-day SMA', color='g')
plt.title('Stock Price and 20-day Simple Moving Average')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()
```



You can calculate different moving averages (e.g., 50-day, 200-day) by changing the window_size and creating additional columns for each moving average you want to calculate.

The moving average helps you identify trends and potential buy/sell signals in the stock's price movements.

Task-6 To calculate the daily return average of a stock, you need to compute the daily returns for the stock's closing prices. Daily returns are typically calculated as the percentage change in price from one day to the next. Here's how you can calculate the daily return average for a stock using Python and Pandas:

Calculate the daily returns by taking the percentage change in the closing price column. You can use the pct_change() function to do this.

In [12]:
```
df['Daily Return'] = df['Close'].pct_change() * 100  # Multiply by 100 to get the
df
```

	Date	Open	High	Low	Close	Adj Close	Volume	Price Change
0	2022-10-24	119.980003	120.389999	116.570000	119.820000	119.820000	49531500	0.021486
1	2022-10-25	119.650002	121.320000	118.949997	120.599998	120.599998	50934600	0.779998
2	2022-10-26	116.000000	119.349998	114.760002	115.660004	115.660004	68802300	-4.939994
3	2022-10-27	113.919998	114.120003	109.769997	110.959999	110.959999	129605400	-4.700005
4	2022-10-28	97.910004	103.959999	97.660004	103.410004	103.410004	223133400	-7.549995
...	...	...	...	...	...	...	...	...
245	2023-10-16	130.690002	133.070007	130.429993	132.550003	132.550003	42832900	2.760010
	2023-							

In [13]:
```
#To calculate the average of the daily returns, you can use the .mean() function o
average_return = df['Daily Return'].mean()
print(f"Average Daily Return: {average_return:.2f}%")
```

```
Average Daily Return: 0.05%
```

The average daily return gives you an idea of the stock's typical daily price movement. Positive values indicate an average daily gain, while negative values indicate an average daily loss.

Task 7 To add a new column called 'Trend' based on the 'Daily Return' of a stock, you can define criteria to categorize the daily returns into different trends. For example, you can classify them as

'Up' for positive returns,

'Down' for negative returns, and

'Stable' for zero returns. Here's how you can do it in Python using Pandas:

Define your criteria for categorizing the daily returns. For this example, we'll use the following criteria:

'Up' for daily returns greater than 0.05

'Down' for daily returns less than 0.05

'Stable' for daily returns equal to 0.05

In [14]:
```
#You can use the np.select() function from NumPy to create the 'Trend' column base
# Define the criteria and corresponding labels.
conditions = [
    df['Daily Return'] > 0.05,    # 'Up' condition
    df['Daily Return'] < 0.05,    # 'Down' condition
    df['Daily Return'] == 0.05    # 'Stable' condition
]

choices = ['Up', 'Down', 'Stable']

# Create the 'Trend' column based on the criteria.
df['Trend'] = np.select(conditions, choices, default='Stable')
```

In [15]:
```
df.head(10)
```

Out[15]:

	Date	Open	High	Low	Close	Adj Close	Volume	Price Change	SMA_20
0	2022-10-24	119.980003	120.389999	116.570000	119.820000	119.820000	49531500	0.021486	NaN
1	2022-10-25	119.650002	121.320000	118.949997	120.599998	120.599998	50934600	0.779998	NaN
2	2022-10-26	116.000000	119.349998	114.760002	115.660004	115.660004	68802300	-4.939994	NaN
3	2022-10-27	113.919998	114.120003	109.769997	110.959999	110.959999	129605400	-4.700005	NaN
4	2022-10-28	97.910004	103.959999	97.660004	103.410004	103.410004	223133400	-7.549995	NaN
5	2022-10-31	103.559998	104.870003	100.739998	102.440002	102.440002	99251400	-0.970002	NaN
6	2022-11-01	103.989998	104.580002	96.059998	96.790001	96.790001	153370000	-5.650001	NaN
7	2022-11-02	97.320000	97.739998	92.010002	92.120003	92.120003	135761800	-4.669998	NaN
8	2022-11-03	92.470001	93.500000	89.019997	89.300003	89.300003	136683300	-2.820000	NaN
9	2022-11-04	91.489998	92.440002	88.040001	90.980003	90.980003	129101300	1.680000	NaN

Task 8

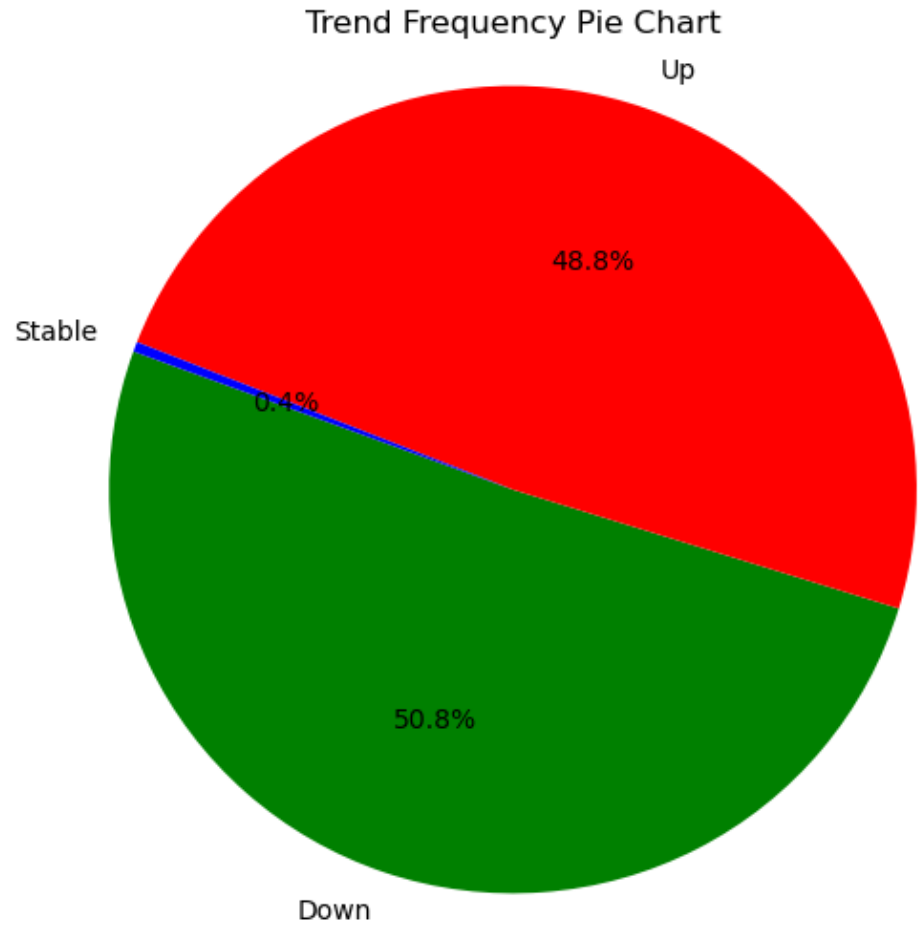
Visualize trend frequency through a Pie Chart.

In [16]:
```
# Count the frequency of each trend category in the 'Trend' column.
trend_counts = df['Trend'].value_counts()

# Define the labels and corresponding colors for the pie chart.
labels = trend_counts.index
colors = ['green', 'red', 'blue']  # You can adjust the colors as needed.

# Create the pie chart.
plt.figure(figsize=(6, 6))
plt.pie(trend_counts, labels=labels, colors=colors, autopct='%1.1f%%',startangle=1
plt.title('Trend Frequency Pie Chart')
plt.axis('equal')  # Equal aspect ratio ensures that the pie chart is circular.

# Show the pie chart.
plt.show()
```

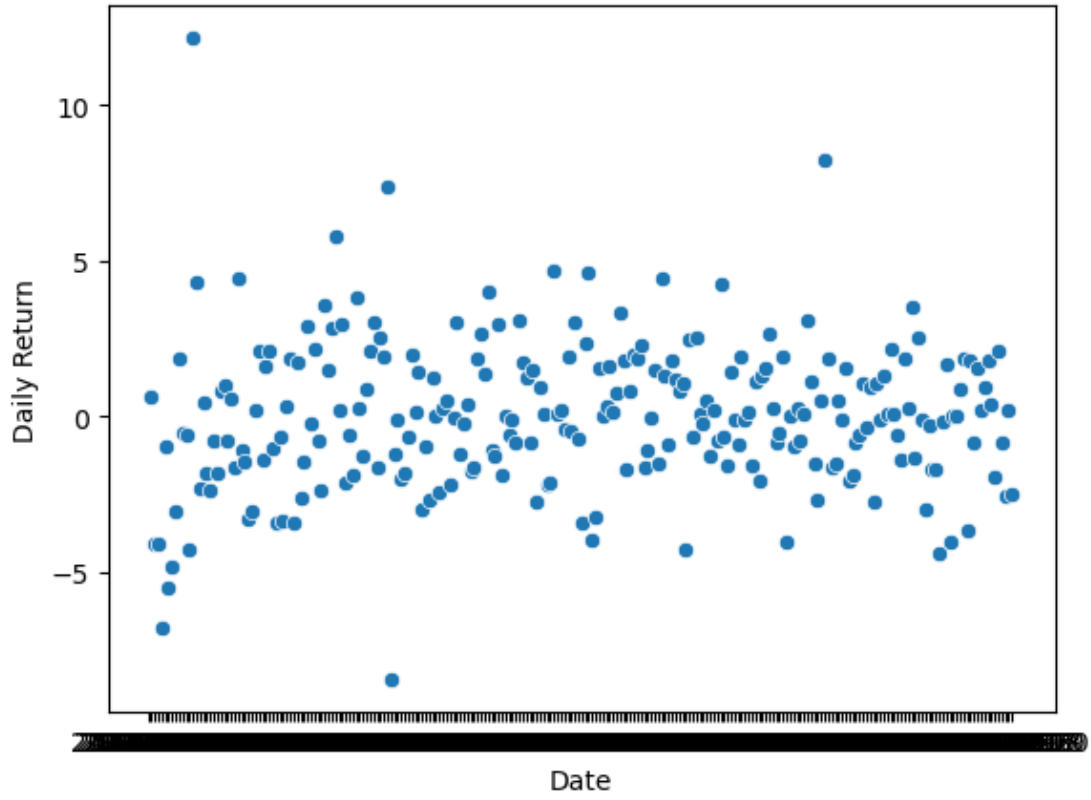


TASK 9

What was the correlation between the daily returns of different stocks?

In [19]:
```
sns.scatterplot(x='Date',y='Daily Return',data=df)
```

Out[19]:
```
<Axes: xlabel='Date', ylabel='Daily Return'>
```



In [ ]:
```

```