Stock Market Analysis

Business Scenario:

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. However, we will analyse 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 analyse the information, visualise it, and look at different ways to analyse the risk of a stock, based on its performance history

About the Data:

The stocks I 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/

Here is some of the Questions we shall try to answer

- Read the Data from Yahoo finance website directly
- Perform cleaning
- What was the change in stock price over time?
- Visualize the change in a stock's volume being traded, over time?
- What was the moving average of various stocks?
- What was the daily return average of a stock?
- Add a new column 'Trend' whose values are based on the 'Daily Return'
- Visualize the trend frequency through a Pie Chart
- What was the correlation between daily returns of different stocks?

Import the required Libraries

```
#Python Data Analysis imports
import pandas as pd
from pandas import Series,DataFrame
import numpy as np

!pip install --upgrade pandas
!pip install --upgrade pandas-datareader

#Visualisation imports
import matplotlib.pyplot as plt
from matplotlib.pyplot import subplot

import seaborn as sns
sns.set_style('whitegrid')
%matplotlib inline

#To grab stock data
#!pip install pandas-datareader
import pandas_datareader.data as web
from datetime import datetime
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/pythc
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-pack
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
Requirement already satisfied: pandas-datareader in /usr/local/lib/python3.7/
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pandas>=0.23 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/pythc
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /us
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/
```

To know more about Pandas DataReader click here: https://pandas-datareader.readthedocs.io/en/latest/

Creating a list with the stock names, for future looping

```
#We're going to analyse stock info for Apple, Google, Microsoft and Amazon.
tech_list = ['AAPL','GOOG','MSFT','AMZN']


#Setting the end date to today
end = datetime.now()


#Start date set to 1 year back
start = datetime(end.year-1,end.month,end.day)
```

Reading the Data from Yahoo finance website directly

```
ɔck data

ɛr(stock,'yahoo',start,end) #The globals method sets the stock name to a global variable
```

Thanks to the globals method, **Amazon**'s stock data will be stored in the AMZN global variable dataframe. Let's see if it works

```
AMZN
```

Date	High	Low	Open	Close	Volume	Adj Close
2021-06-01	3250.979980	3209.060059	3243.500000	3218.649902	2430000	3218.649902
2021-06-02	3235.000000	3208.000000	3223.100098	3233.989990	2014500	3233.989990
2021-06-03	3214.439941	3184.030029	3204.229980	3187.010010	2398300	3187.010010
2021-06-04	3221.000000	3198.810059	3212.000000	3206.219971	2249700	3206.219971
2021-06-07	3208.000000	3172.199951	3197.330078	3198.010010	2215800	3198.010010
...	...	...	...	...	...	...
2022-05-23	2176.379883	2079.000000	2169.219971	2151.139893	5389900	2151.139893
2022-05-24	2108.000000	2025.199951	2080.500000	2082.000000	5146700	2082.000000
2022-05-25	2163.500000	2073.000000	2073.110107	2135.500000	4656000	2135.500000
2022-05-26	2253.310059	2149.070068	2159.399902	2221.550049	4650100	2221.550049
2022-05-27	2303.739990	2252.560059	2271.000000	2302.929932	4675800	2302.929932

252 rows × 6 columns

Inference:

Now, we have successfully grabbed the data from Yahoo Finance!!

Amazon's Stock’s minimum, maximum, and average price was

```
AMZN.describe()
```

	High	Low	Open	Close	Volume	Adj Close
count	252.000000	252.000000	252.000000	252.000000	2.520000e+02	252.000000
mean	3259.430356	3177.744645	3220.760284	3217.881395	3.575421e+06	3217.881395
std	354.104909	368.543901	359.754837	362.428431	1.594278e+06	362.428431
min	2108.000000	2025.199951	2055.000000	2082.000000	1.680300e+06	2082.000000
25%	3158.012512	3032.502502	3111.594910	3092.090027	2.578850e+06	3092.090027
50%	3345.165039	3287.669922	3315.505005	3309.994995	3.195350e+06	3309.994995
75%	3475.182434	3404.604980	3441.425049	3444.822510	4.029450e+06	3444.822510
max	3773.080078	3696.790039	3744.000000	3731.409912	1.361650e+07	3731.409912

Cleaning the Data

Adding a new column named **Company** in each Dataframe

```
AAPL['Company'] = 'Apple'
GOOG['Company'] = 'Google'
AMZN['Company'] = 'Amazon'
MSFT['Company'] = 'Microsoft'
```

Merging all the Dataframe together to create a single Dataframe

```
stocks=pd.concat([AAPL,AMZN,GOOG,MSFT])
```

```
stocks.head()
```

Date	High	Low	Open	Close	Volume	Adj Close	Company
2021-06-01	125.349998	123.940002	125.080002	124.279999	67637100.0	123.573997	Appl
2021-06-02	125.239998	124.050003	124.279999	125.059998	59278900.0	124.349556	Appl
2021-06-03	124.849998	123.129997	124.680000	123.540001	76229200.0	122.838196	Appl
2021-06-04	126.160004	123.849998	124.070000	125.889999	75169300.0	125.174843	Appl
2021-06-07	126.320000	124.830002	126.169998	125.900002	71057600.0	125.184792	Appl

Store the data in a CSV file (if required)

```
stocks.to_csv('Stocks.csv',index=False)
```

Checking the basic information of the data

```
stocks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1008 entries, 2021-06-01 to 2022-05-27
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   High       1008 non-null   float64
 1   Low        1008 non-null   float64
 2   Open       1008 non-null   float64
 3   Close      1008 non-null   float64
 4   Volume     1008 non-null   float64
 5   Adj Close  1008 non-null   float64
 6   Company    1008 non-null   object 
dtypes: float64(6), object(1)
memory usage: 63.0+ KB
```

Checking for null values

```
stocks.isna().sum()
```

```
High         0
Low          0
Open         0
Close        0
Volume       0
Adj Close    0
```

```
Company      0
dtype: int64
```

Inference:

No missing info in the dataframe

Checking for duplicates

```
stocks.duplicated().sum()
```

```
0
```

Inference:

No duplicate record in the dataframe

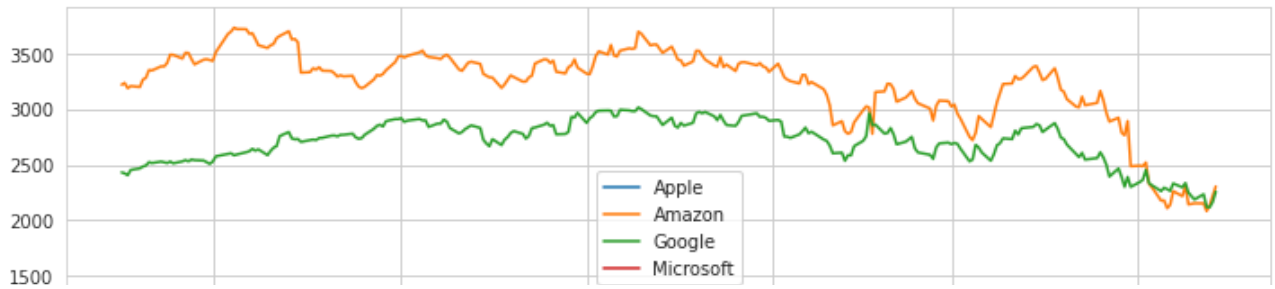
What was the change in a stock's price over time?

```
Companies = stocks['Company'].drop_duplicates().values
print(Companies)
```

```
['Apple' 'Amazon' 'Google' 'Microsoft']
```

```
grouped = stocks.groupby('Company')
fig, ax = plt.subplots()

for comp in Companies:
    stock = grouped.get_group(comp)
    stock['Adj Close'].plot(legend=True,figsize=(12,5), ax=ax)

ax.legend(Companies)
```

```
<matplotlib.legend.Legend at 0x7fae78990ed0>
```



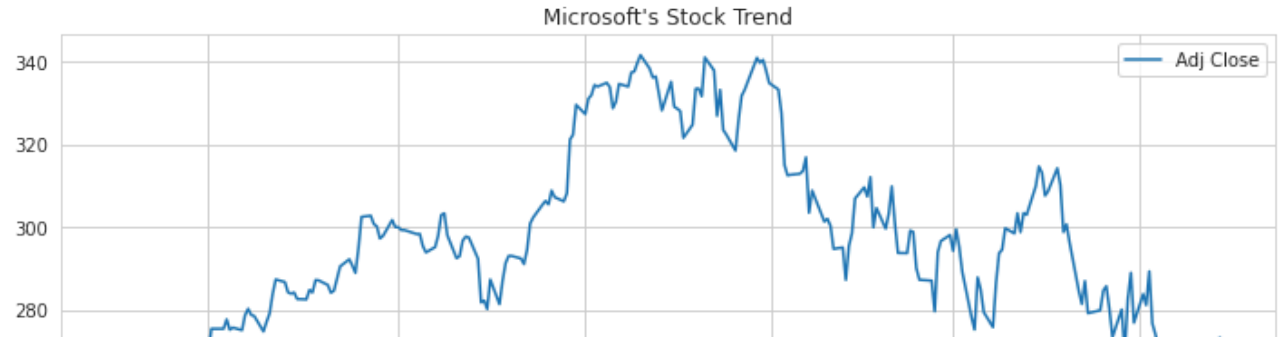
Let us look at the price trend of individual stocks

```
AMZN['Adj Close'].plot(legend=True,figsize=(12,5),title = "Amazon's Stock Trend")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fae755bcf10>
```


```
#Plotting the stock's adjusted closing price using pandas
MSFT['Adj Close'].plot(legend=True,figsize=(12,5),title = "Microsoft's Stock Trend")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fae7562fa10>
```



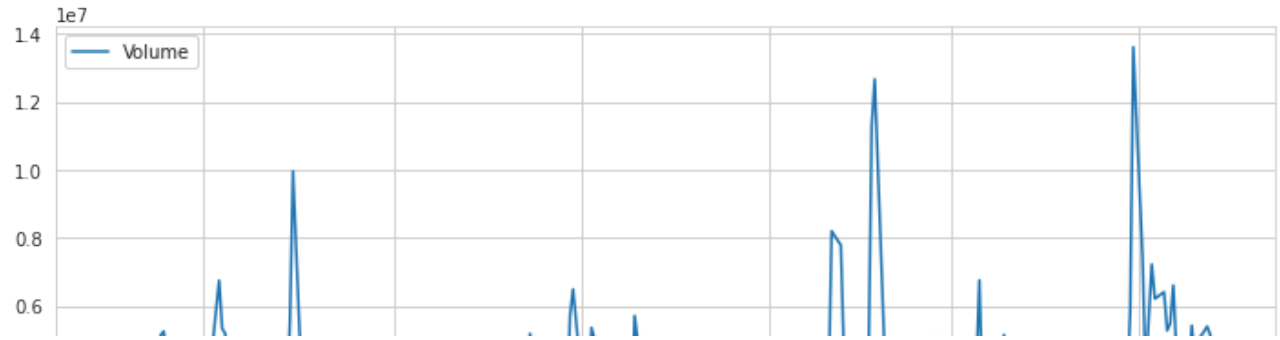
Inference:

MSFT stock notched a record high of 349.67 on Nov. 22. But it tumbled during the recent stock market correction. It ended the regular session May 6 at 274.73.

Visualize the change in a stock's volume being traded, over time

```
#Plotting the total volume being traded over time
AMZN['Volume'].plot(legend=True,figsize=(12,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fae7563e1d0>
```



Inference:

As we can observe, there was a spike in the volume in the Feb 2022 and then again in th month of May.

What was the moving average of various stocks?

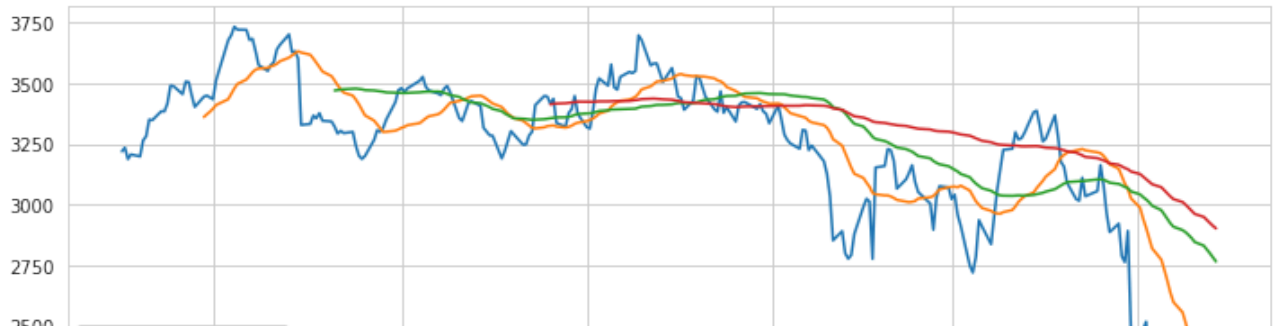
```
ma_per_day = [20,50,100]

for ma in ma_per_day:
    column_name = "MA for %s days" %(str(ma))

    AMZN[column_name] = AMZN['Adj Close'].rolling(window=ma,center=False).mean()
```

```
AMZN.tail()
```

Date	High	Low	Open	Close	Volume	Adj Close	Comp
2022-05-23	2176.379883	2079.000000	2169.219971	2151.139893	5389900	2151.139893	Ama
2022-05-24	2108.000000	2025.199951	2080.500000	2082.000000	5146700	2082.000000	Ama
2022-05-25	2163.500000	2073.000000	2073.110107	2135.500000	4656000	2135.500000	Ama
2022-05-26	2253.310059	2149.070068	2159.399902	2221.550049	4650100	2221.550049	Ama
2022-05-27	2303.739990	2252.560059	2271.000000	2302.929932	4675800	2302.929932	Ama

```
AMZN[['Adj Close','MA for 20 days','MA for 50 days','MA for 100 days']].plot(subplots=Fa
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fae75652d50>
```



Inference:

Moving averages for more days have a smoother plot, as they're less reliable on daily fluctuations. Still we can see that Amazon's stock wahas been seeing a slight dip this year.

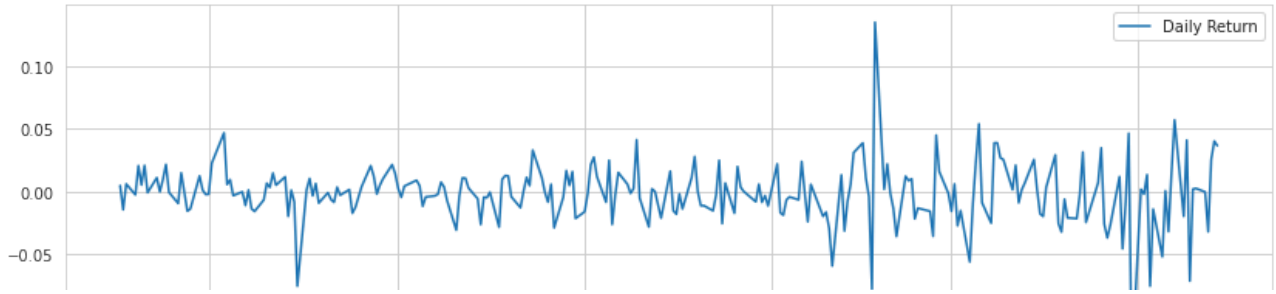
What was the daily return average of a stock?

```
#The daily return column can be created by using the percentage change over the adjuste
AMZN['Daily Return'] = AMZN['Adj Close'].pct_change()


AMZN['Daily Return'].tail()
```

```
Date
2022-05-23   -0.000316
2022-05-24   -0.032141
2022-05-25    0.025696
2022-05-26    0.040295
2022-05-27    0.036632
Name: Daily Return, dtype: float64
```

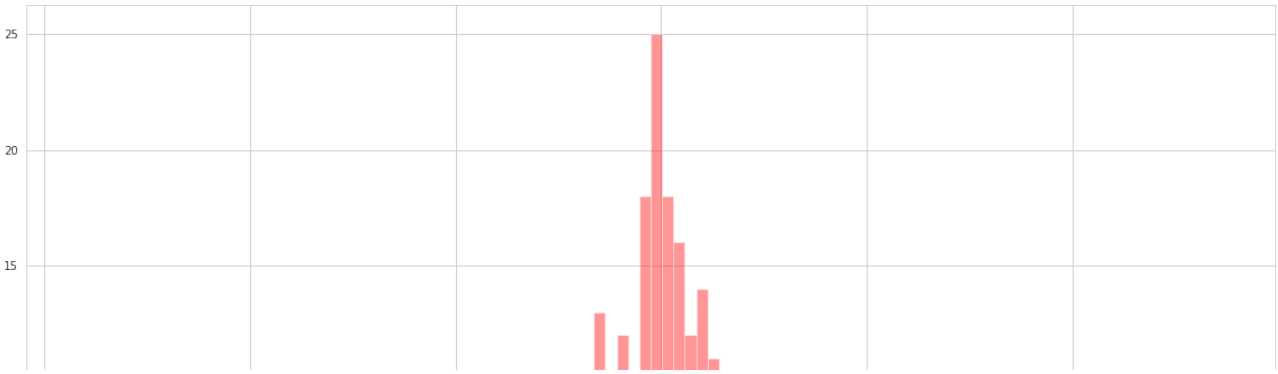
```
#Plotting the daily return
AMZN['Daily Return'].plot(figsize=(14,5),legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fae753f9150>
```




```
f,ax=plt.subplots(figsize=(20,10))
sns.distplot(AMZN['Daily Return'].dropna(),bins=100,color='red',kde=False,ax=ax)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureW
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fae753eab10>
```



Inference:

Positive daily returns are almost as frequent as negative returns for Amazon currently.

Add a new column ‘Trend’ whose values are based on the 'Daily Return'

```
def trend(x):
  if x > -0.015 and x <= 0.015:
    return 'Slight or No change'
  elif x > 0.015 and x <= 0.04:
```

```
    return 'Slight Positive'
  elif x < -0.015 and x >= -0.4:
    return 'Slight Negative'
  elif x > 0.04 and x <= 0.06:
    return 'Positive'
  elif x < -0.04 and x >= -0.06:
    return 'Negative'
  elif x > 0.06 and x <= 0.07:
    return 'Among top gainers'
  elif x < -0.06 and x >= -0.07:
    return 'Among top losers'
  elif x > 0.07:
    return 'Bull run'
  elif x <= -0.07:
    return 'Bear drop'
AMZN['Trend']= np.zeros(AMZN['Daily Return'].count()+1)
AMZN['Trend']= AMZN['Daily Return'].apply(lambda x:trend(x))
AMZN.tail(5)
```

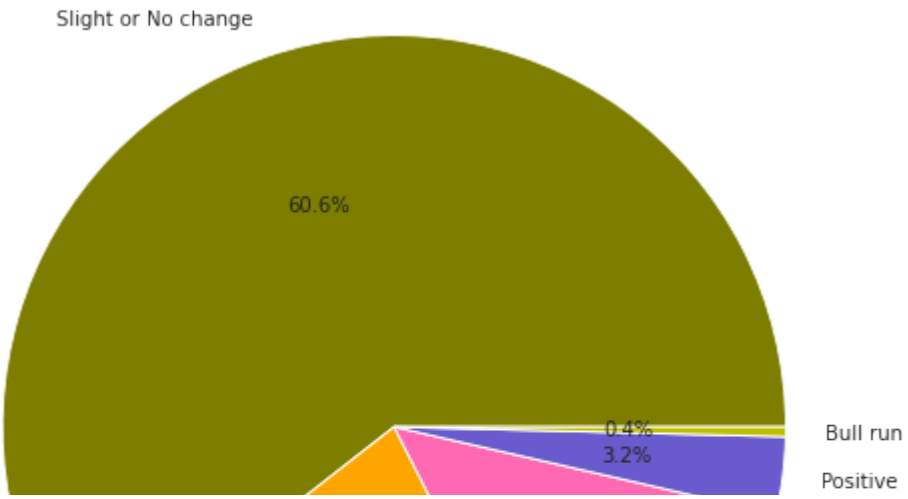
Date	High	Low	Open	Close	Volume	Adj Close	Comp
2022-05-23	2176.379883	2079.000000	2169.219971	2151.139893	5389900	2151.139893	Ama
2022-05-24	2108.000000	2025.199951	2080.500000	2082.000000	5146700	2082.000000	Ama
2022-05-25	2163.500000	2073.000000	2073.110107	2135.500000	4656000	2135.500000	Ama
2022-05-26	2253.310059	2149.070068	2159.399902	2221.550049	4650100	2221.550049	Ama
2022-05-27	2303.739990	2252.560059	2271.000000	2302.929932	4675800	2302.929932	Ama

We can visualize to see how the stock was performing for past and this year using a Pie chart

Visualize the trend frequency through a Pie Chart

```
labels=AMZN.Trend.value_counts().index
colors=["olive","orange","hotpink","slateblue","y","lime"]
sizes=AMZN.Trend.value_counts().values
plt.figure(figsize=(9,9))
plt.pie(sizes,labels=labels,colors=colors,autopct="%1.1f%%")
plt.title("Trend Frequency using a Pie Chart",color="saddlebrown",fontsize=15)
```

```
Text(0.5, 1.0, 'Trend Frequency using a Pie Chart')
```



Inference:

The Amazon stock was in the bull run for just about 0.4% of the time,for about 3.2% of the time period, the stock has performed positively on a given day. Likewise, for most period of time (about 60.6%) the stock showed a very slight or no change in the price.

What was the correlation between daily returns of different stocks?

Merging the columns of all companies Adjacent close

```
all_adj = pd.concat([AAPL['Adj Close'] ,GOOG['Adj Close'],AMZN['Adj Close'],MSFT['Adj C
```

```
all_adj.columns = ['Apple','Google','Amazon','Microsoft']
```

```
all_adj
```

Date	Apple	Google	Amazon	Microsoft
2021-06-01	123.573997	2429.810059	3218.649902	245.396210
2021-06-02	124.349556	2421.280029	3233.989990	245.297012
2021-06-03	122.838196	2404.610107	3187.010010	243.719894
2021-06-04	125.174843	2451.760010	3206.219971	248.758728
2021-06-07	125.184792	2466.090088	3198.010010	251.754303
...	...	...	...	...
2022-05-23	143.110001	2233.330078	2151.139893	260.649994
2022-05-24	140.360001	2118.520020	2082.000000	259.619995
2022-05-25	140.520004	2116.790039	2135.500000	262.519989

```
pct_df = all_adj.pct_change()
pct_df.tail()
```

Date	Apple	Google	Amazon	Microsoft
2022-05-23	0.040119	0.021530	-0.000316	0.032032
2022-05-24	-0.019216	-0.051408	-0.032141	-0.003952
2022-05-25	0.001140	-0.000817	0.025696	0.011170
2022-05-26	0.023200	0.023210	0.040295	0.012875
2022-05-27	0.040757	0.041581	0.036632	0.027604