About the Data: The stocks we have chosen are from various industries and market caps namely,

• Apple • Google • Microsoft • Amazon

The following tasks are to be performed:

• 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 trend frequency through a Pie Chart.

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

• Read the Data from Yahoo finance website directly.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import yfinance as yf

tick1='AMZN'
tick2='AAPL'
tick3='GOOG'
tick4='MSFT'

amz=yf.download(tick1,start='2020-01-01',end='2023-01-01')
apl=yf.download(tick2,start='2020-01-01',end='2023-01-01')
goog=yf.download(tick3,start='2020-01-01',end='2023-01-01')
msft=yf.download(tick4,start='2020-01-01',end='2023-01-01')

amz.head()
```

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

Out[1]:

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	93.750000	94.900497	93.207497	94.900497	94.900497	80580000
2020-01-03	93.224998	94.309998	93.224998	93.748497	93.748497	75288000
2020-01-06	93.000000	95.184502	93.000000	95.143997	95.143997	81236000
2020-01-07	95.224998	95.694504	94.601997	95.343002	95.343002	80898000
2020-01-08	94.902000	95.550003	94.321999	94.598503	94.598503	70160000

• Perform cleaning.

In [2]:
```
lst=["Amazon", "Apple", "Google", "Microsoft"]
ticks=[amz,apl,goog,msft]
```

In [3]:
```
for i in range(4):
    print(lst[i])

    print(ticks[i].head())
    print("************************")
```

```
Amazon
                 Open       High        Low      Close  Adj Close    Volume
Date
2020-01-02  93.750000  94.900497  93.207497  94.900497  94.900497  80580000
2020-01-03  93.224998  94.309998  93.224998  93.748497  93.748497  75288000
2020-01-06  93.000000  95.184502  93.000000  95.143997  95.143997  81236000
2020-01-07  95.224998  95.694504  94.601997  95.343002  95.343002  80898000
2020-01-08  94.902000  95.550003  94.321999  94.598503  94.598503  70160000
************************
Apple
```

```
                 Open       High        Low      Close  Adj Close     Volume
Date
2020-01-02  74.059998  75.150002  73.797501  75.087502  73.561539  135480400
2020-01-03  74.287498  75.144997  74.125000  74.357498  72.846375  146322800
2020-01-06  73.447502  74.989998  73.187500  74.949997  73.426834  118387200
2020-01-07  74.959999  75.224998  74.370003  74.597504  73.081490  108872000
2020-01-08  74.290001  76.110001  74.290001  75.797501  74.257111  132079200
**************************
Google
                 Open       High        Low      Close  Adj Close    Volume
Date
2020-01-02  67.077499  68.406998  67.077499  68.368500  68.368500  28132000
2020-01-03  67.392998  68.625000  67.277199  68.032997  68.032997  23728000
2020-01-06  67.500000  69.824997  67.500000  69.710503  69.710503  34646000
2020-01-07  69.897003  70.149498  69.518997  69.667000  69.667000  30054000
2020-01-08  69.603996  70.579002  69.542000  70.216003  70.216003  30560000
**************************
Microsoft
                  Open        High         Low       Close   Adj Close  \
Date
2020-01-02  158.779999  160.729996  158.330002  160.619995  156.151932
2020-01-03  158.320007  159.949997  158.059998  158.619995  154.207565
2020-01-06  157.080002  159.100006  156.509995  159.029999  154.606171
2020-01-07  159.320007  159.669998  157.320007  157.580002  153.196503
2020-01-08  158.929993  160.800003  157.949997  160.089996  155.636703

              Volume
Date
2020-01-02  22622100
2020-01-03  21116200
2020-01-06  20813700
2020-01-07  21634100
2020-01-08  27746500
**************************
```

In [4]:

```
for i in range(4):
    print(lst[i])
    print(ticks[i].info())
    print("********************")
```

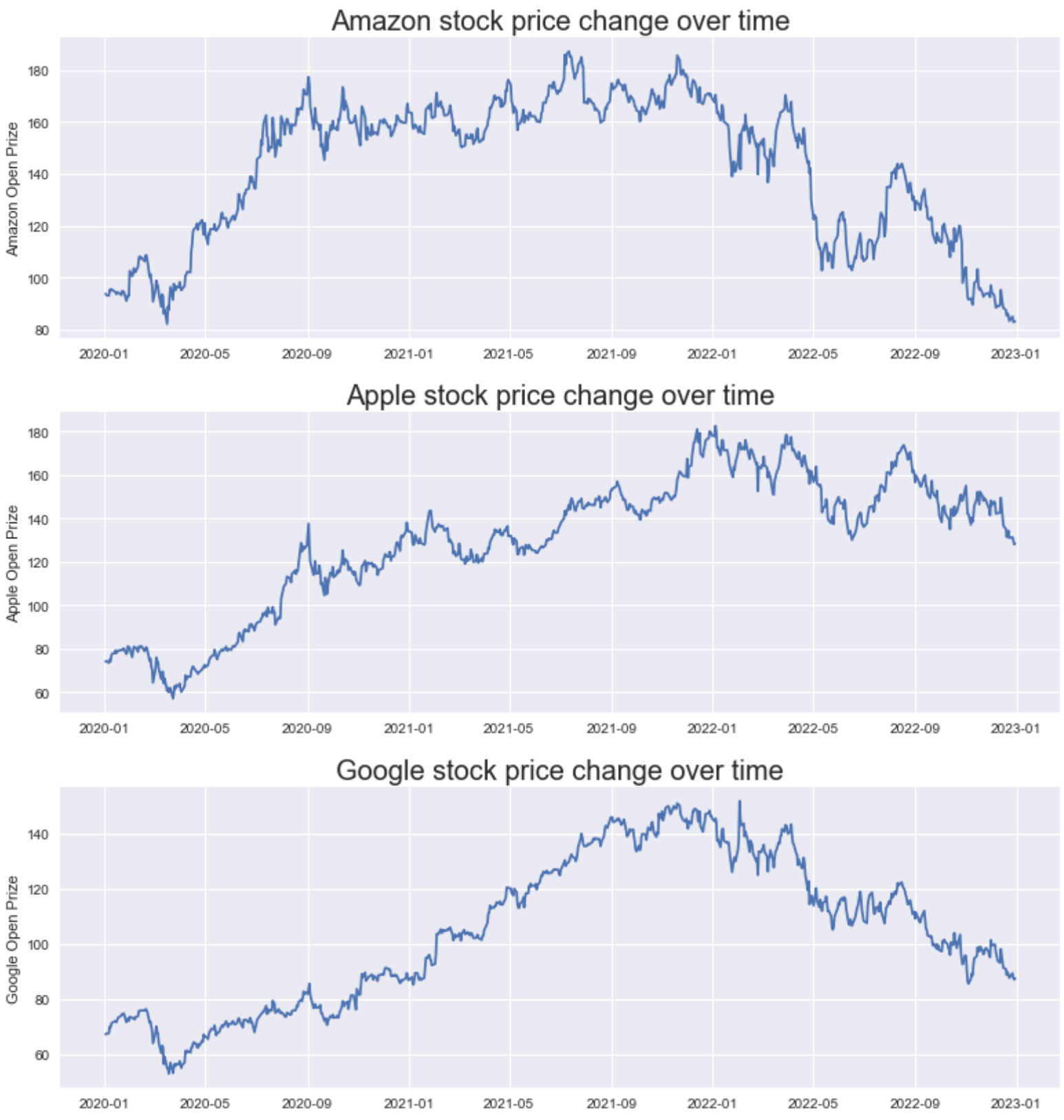
```
Amazon
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 756 entries, 2020-01-02 to 2022-12-30
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       756 non-null    float64
 1   High       756 non-null    float64
 2   Low        756 non-null    float64
 3   Close      756 non-null    float64
 4   Adj Close  756 non-null    float64
 5   Volume     756 non-null    int64
dtypes: float64(5), int64(1)
memory usage: 41.3 KB
None
********************
Apple
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 756 entries, 2020-01-02 to 2022-12-30
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       756 non-null    float64
 1   High       756 non-null    float64
 2   Low        756 non-null    float64
 3   Close      756 non-null    float64
 4   Adj Close  756 non-null    float64
 5   Volume     756 non-null    int64
dtypes: float64(5), int64(1)
memory usage: 41.3 KB
None
********************
Google
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 756 entries, 2020-01-02 to 2022-12-30
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       756 non-null    float64
 1   High       756 non-null    float64
 2   Low        756 non-null    float64
 3   Close      756 non-null    float64
 4   Adj Close  756 non-null    float64
 5   Volume     756 non-null    int64
dtypes: float64(5), int64(1)
memory usage: 41.3 KB
None
********************
Microsoft
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 756 entries, 2020-01-02 to 2022-12-30
Data columns (total 6 columns):
```

```
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       756 non-null    float64
 1   High       756 non-null    float64
 2   Low        756 non-null    float64
 3   Close      756 non-null    float64
 4   Adj Close  756 non-null    float64
 5   Volume     756 non-null    int64
dtypes: float64(5), int64(1)
memory usage: 41.3 KB
None
********************
```

- **What was the change in stock price over time?**

In [5]:

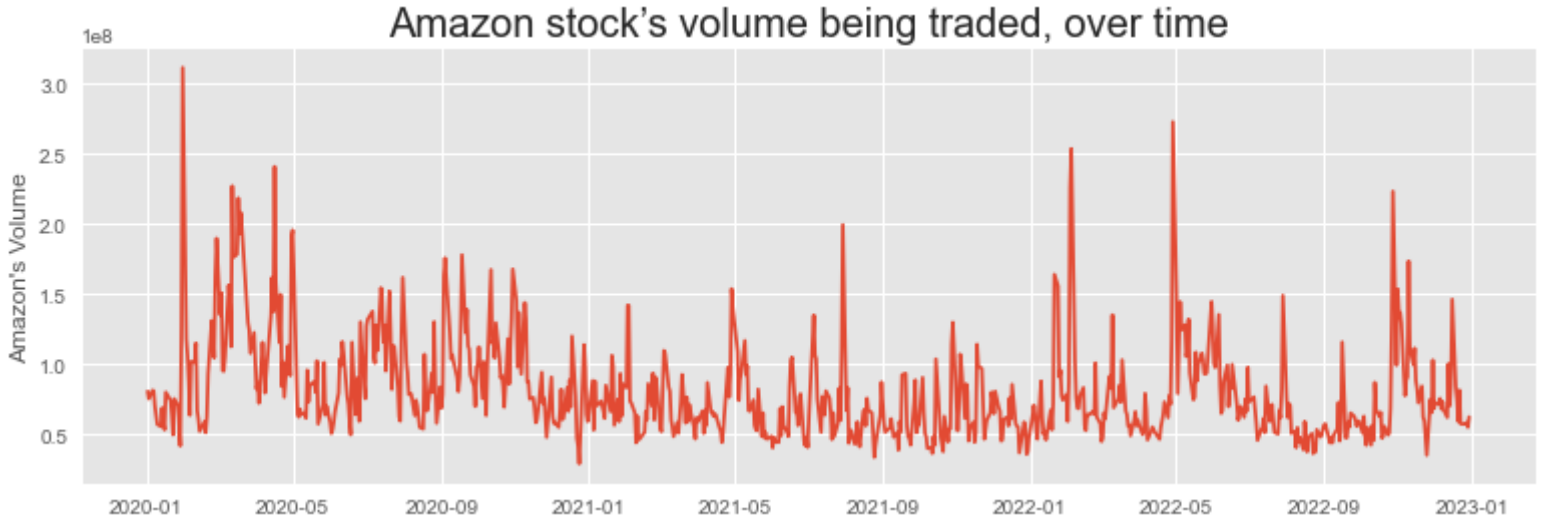
```
for i in range(4):

    plt.figure(figsize=(13,4))
    plt.style.use('seaborn')
    plt.plot(ticks[i]["Open"])
    plt.ylabel(lst[i]+" Open Prize")
    plt.title(lst[i]+" stock price change over time",fontdict={'fontsize': 20})
```

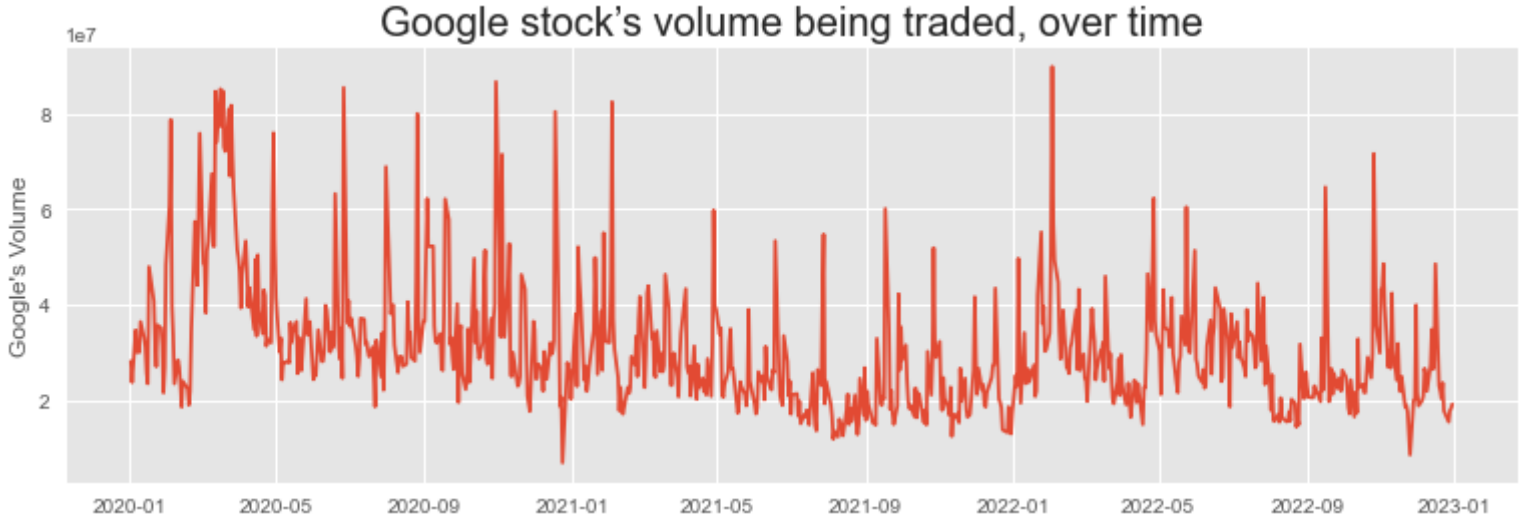


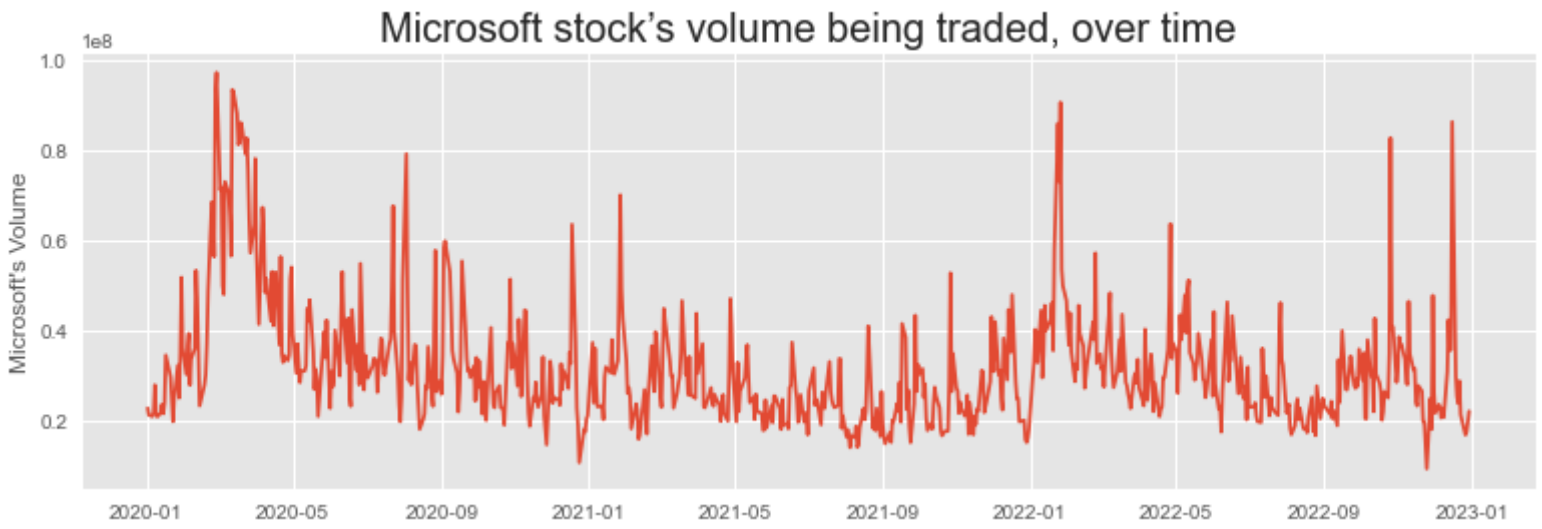
- **Visualize the change in a stock's volume being traded, over time?**

In [6]:

```
for i in range(4):

    plt.figure(figsize=(13,4))
    plt.style.use('ggplot')
    plt.plot(ticks[i]["Volume"])
    plt.ylabel(lst[i]+"'s Volume")
    plt.title(lst[i]+" stock's volume being traded, over time",fontdict={'fontsize': 20})
```



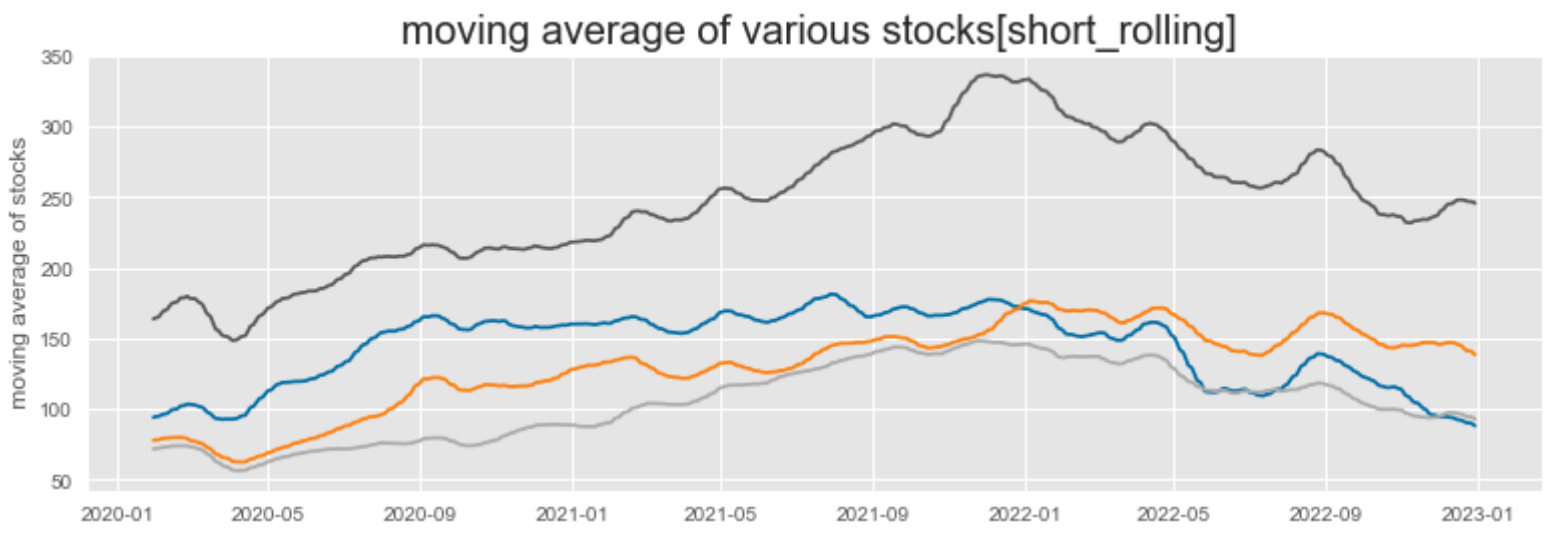




- **What was the moving average of various stocks?**

In [7]:

```
plt.figure(figsize=(13,4))
plt.style.use('tableau-colorblind10')
for i in range(4):
    ticks[i]["short_rolling"] = ticks[i]["Open"].rolling(window=20).mean()


    plt.plot(ticks[i]["short_rolling"],label=lst[i])
    plt.ylabel(" moving average of stocks")
    plt.title(" moving average of various stocks[short_rolling]",fontdict={'fontsize': 20})
```

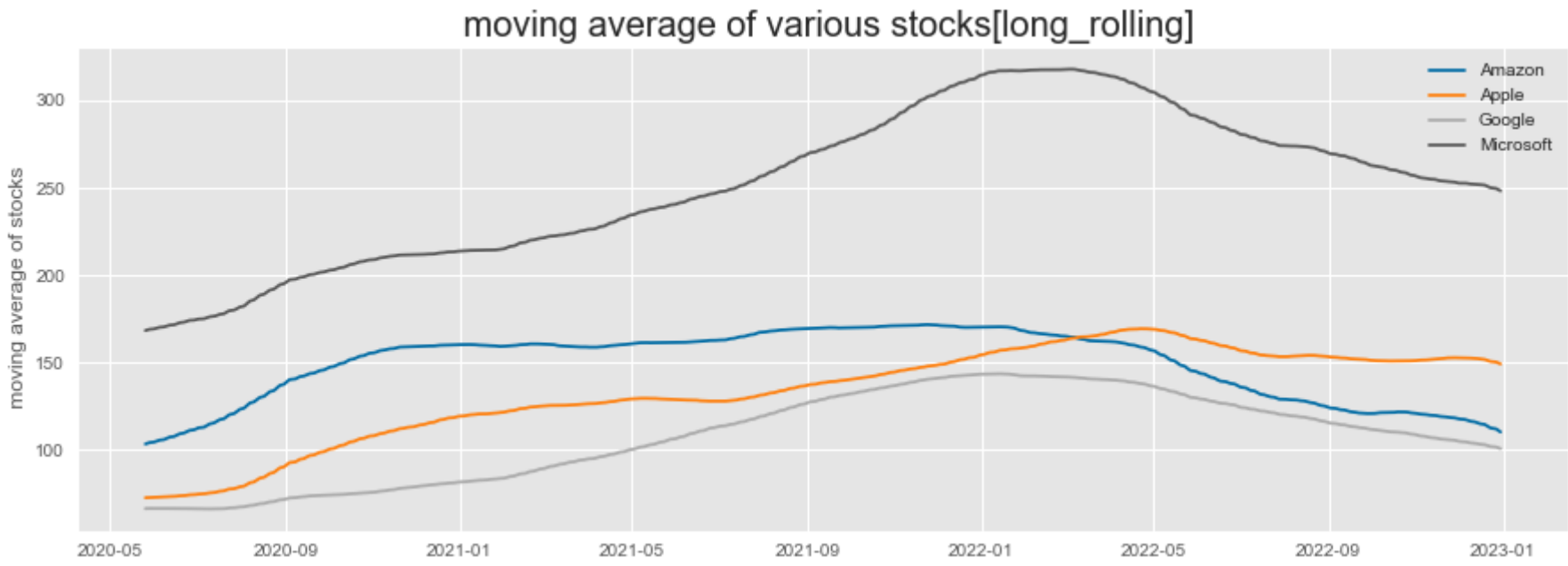


In [8]:

```
plt.figure(figsize=(15,5))
plt.style.use('tableau-colorblind10')

for i in range(4):
    ticks[i]["long_rolling"] = ticks[i]["Open"].rolling(window=100).mean()


    plt.plot(ticks[i]["long_rolling"], label=lst[i])
    plt.ylabel(" moving average of stocks")
    plt.title(" moving average of various stocks[long_rolling]",fontdict={'fontsize': 20})
    plt.legend()

ticks[1].head()
```

Out[8]:

Date	Open	High	Low	Close	Adj Close	Volume	short_rolling	long_rolling
2020-01-02	74.059998	75.150002	73.797501	75.087502	73.561539	135480400	NaN	NaN
2020-01-03	74.287498	75.144997	74.125000	74.357498	72.846375	146322800	NaN	NaN
2020-01-06	73.447502	74.989998	73.187500	74.949997	73.426834	118387200	NaN	NaN
2020-01-07	74.959999	75.224998	74.370003	74.597504	73.081490	108872000	NaN	NaN
2020-01-08	74.290001	76.110001	74.290001	75.797501	74.257111	132079200	NaN	NaN



- **What was the daily return average of a stock?**

In [9]:
```
daily_return={}
for i in range(4):
    ret = ticks[i]['Adj Close'].pct_change()
    daily_return[lst[i]]=list(ret)
daily_return=pd.DataFrame(daily_return)
daily_return.head()
```

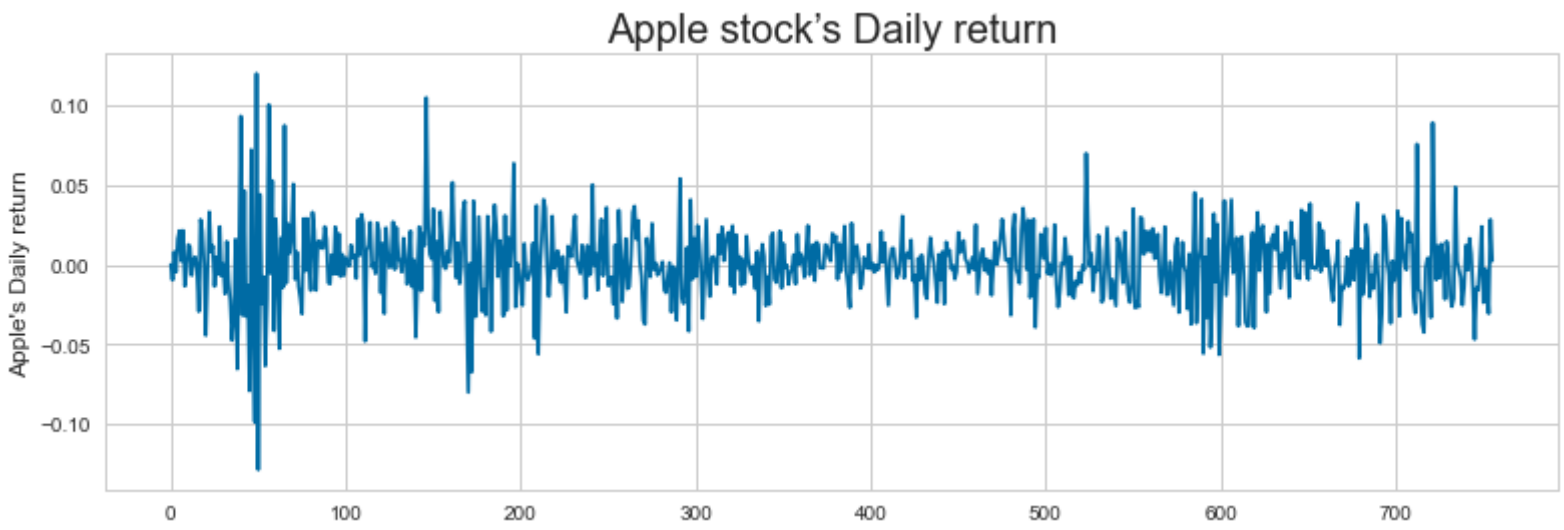
Out[9]:

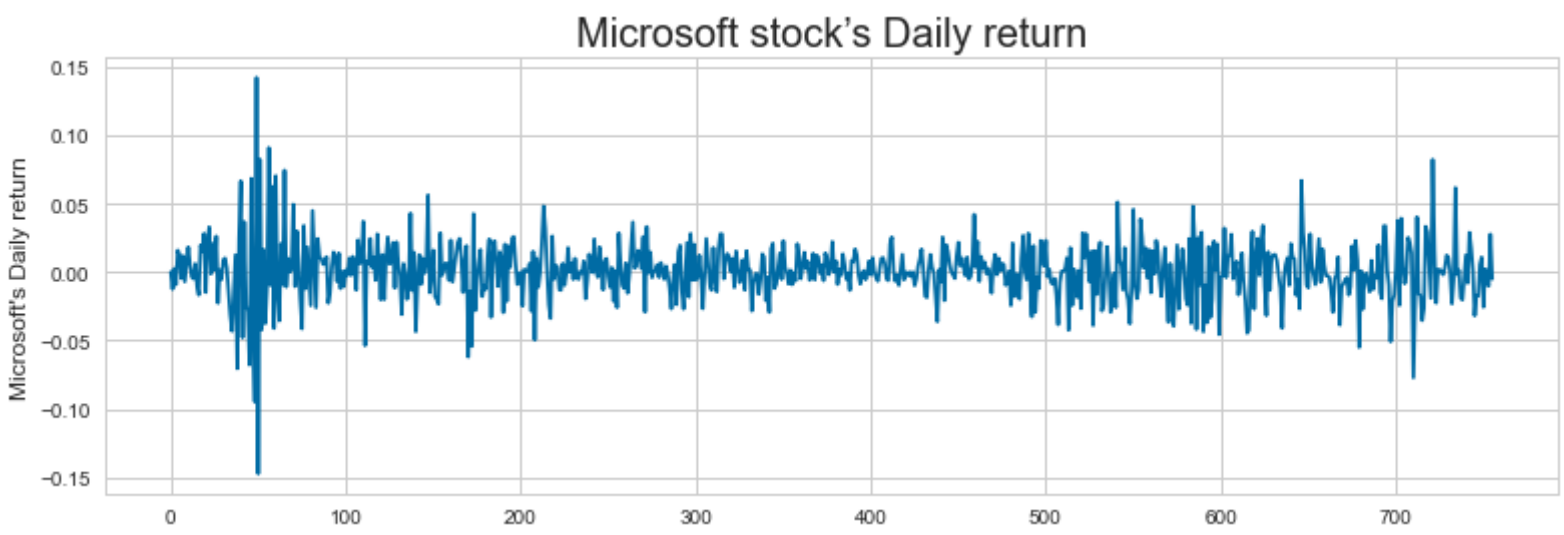
	Amazon	Apple	Google	Microsoft
0	NaN	NaN	NaN	NaN
1	-0.012139	-0.009722	-0.004907	-0.012452
2	0.014886	0.007968	0.024657	0.002585
3	0.002092	-0.004703	-0.000624	-0.009118
4	-0.007809	0.016086	0.007880	0.015929

In [11]:
```
# Filling the Null values with 0

daily_return=daily_return.fillna(0)
daily_return.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 756 entries, 0 to 755
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Amazon     756 non-null    float64
 1   Apple      756 non-null    float64
 2   Google     756 non-null    float64
 3   Microsoft  756 non-null    float64
dtypes: float64(4)
memory usage: 23.8 KB
```

In [12]:
```
for i in lst:

    plt.figure(figsize=(13,4))
    plt.style.use('seaborn-whitegrid')
    plt.plot(daily_return[i])

    plt.ylabel(i+"'s Daily return")
    plt.title(i+" stock's Daily return",fontdict={'fontsize': 20})
```





• Add a new column 'Trend' whose values are based on the 'Daily Return'.

In [13]:

```
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'

for i in range(4):
    ticks[i]["Trend"] = list(daily_return[lst[i]].apply(lambda x:trend(x)))

amz.tail(5)
```

Out[13]:

Date	Open	High	Low	Close	Adj Close	Volume	short_rolling	long_rolling	Trend
2022-12-23	83.250000	85.779999	82.930000	85.250000	85.250000	57433700	90.241500	112.6503	Slight Positive
2022-12-27	84.970001	85.349998	83.000000	83.040001	83.040001	57284000	89.793500	112.0942	Slight Negative
2022-12-28	82.800003	83.480003	81.690002	81.820000	81.820000	58228600	89.231500	111.5212	Slight or No change
2022-12-29	82.870003	84.550003	82.550003	84.180000	84.180000	54995900	88.751500	110.9294	Slight Positive

Date	Open	High	Low	Close	Adj Close	Volume	short_rolling	long_rolling	Trend
2022-12-30	83.120003	84.050003	82.470001	84.000000	84.000000	62330000	88.058001	110.3801	Slight or No change

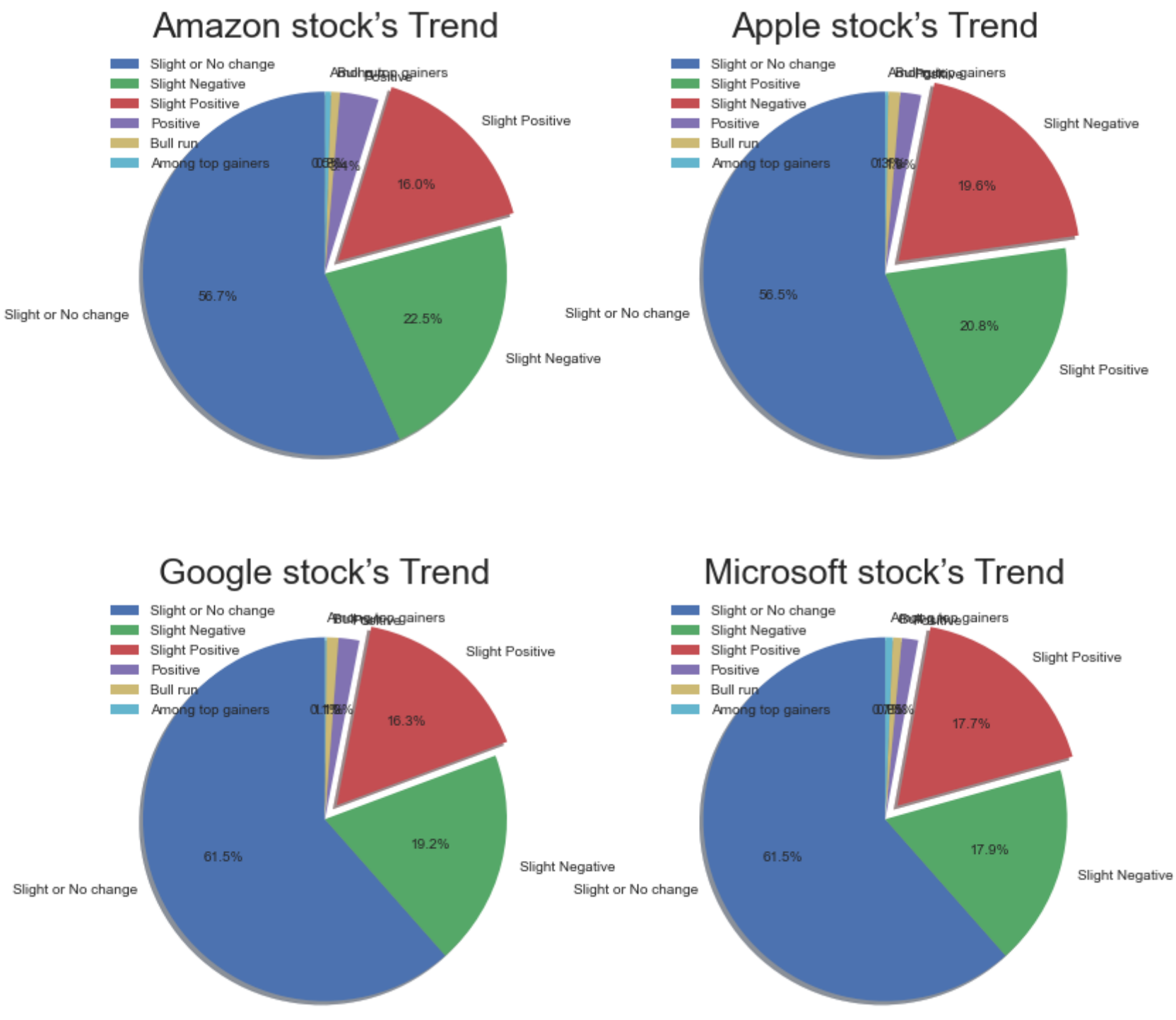
In [14]:
```
ticks[2].tail()
```

Out[14]:

Date	Open	High	Low	Close	Adj Close	Volume	short_rolling	long_rolling	Trend
2022-12-23	87.620003	90.099998	87.620003	89.809998	89.809998	17815000	94.61325	102.03560	Slight Positive
2022-12-27	89.309998	89.500000	87.535004	87.930000	87.930000	15470900	94.21875	101.74570	Slight Negative
2022-12-28	87.500000	88.519997	86.370003	86.459999	86.459999	17879600	93.79375	101.45140	Slight Negative
2022-12-29	87.029999	89.364998	86.989998	88.949997	88.949997	18280700	93.38925	101.13050	Slight Positive
2022-12-30	87.364998	88.830002	87.029999	88.730003	88.730003	19179300	92.68750	100.82425	Slight or No change

• Visualize trend frequency through a Pie Chart.

In [15]:
```
j=1
plt.figure(figsize=(13,13))
for i in range(4):

    size=ticks[i]["Trend"].value_counts()
    labels=list(size.index)
    explode = (0,0, 0.1,0, 0, 0)
    plt.style.use('seaborn')

    plt.subplot(2,2,j)

    plt.pie(size,explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)

    plt.title(lst[i]+" stock's Trend",fontdict={'fontsize': 25})
    plt.legend()
    j=j+1
```



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

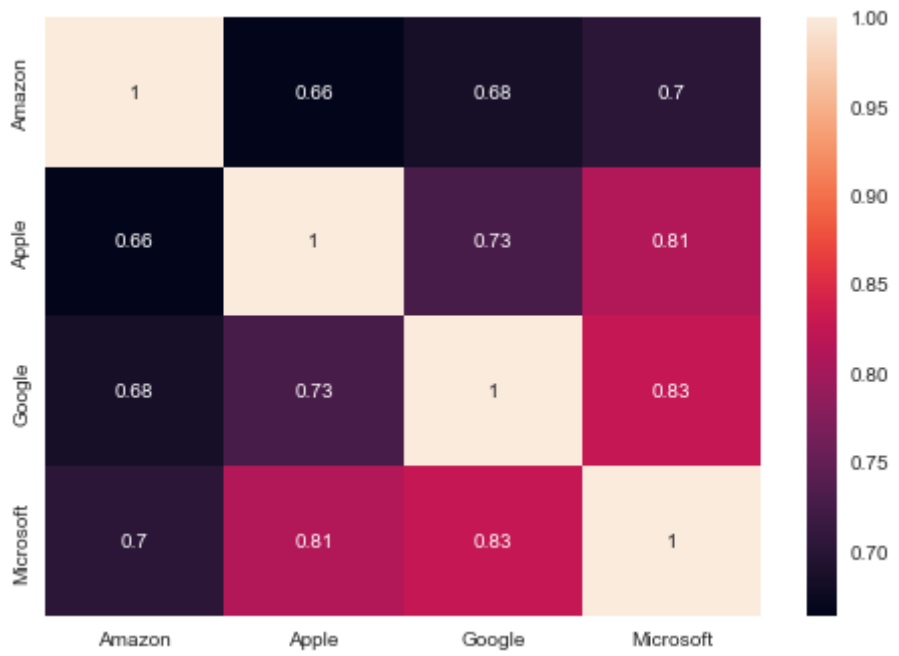
In [16]:
```
daily_return.corr()
```

Out[16]:

	Amazon	Apple	Google	Microsoft
Amazon	1.000000	0.663192	0.683296	0.701369
Apple	0.663192	1.000000	0.727659	0.811363
Google	0.683296	0.727659	1.000000	0.825172
Microsoft	0.701369	0.811363	0.825172	1.000000

In [17]:
```
sns.heatmap(daily_return.corr(), annot=True, cbar= True)
```

Out[17]:
<AxesSubplot:>



In [ ]:

In [ ]:

In [ ]: