

```
In [2]: import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns
```

```
In [3]: import glob
```

```
In [4]: glob.glob(r'C:\\Users\\Rahul\\OneDrive\\Desktop\\Data analyst project\\S&P 500 stoc
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
'C:\\\\Users\\\\Rahul\\\\OneDrive\\\\Desktop\\\\Data analyst project\\\\S&P 500 s
tock return\\\\individual_stocks_5yr\\\\ZTS_data.csv']
```

```
In [5]: len(glob.glob(r'C:\\Users\\Rahul\\OneDrive\\Desktop\\Data analyst project\\S&P 500
'))
```

```
Out[5]: 505
```

```
In [6]: # Lets store files of those stock that we have to consider for analysis ..
```

```
In [7]: company_list = [
    r'C:\\Users\\Rahul\\OneDrive\\Desktop\\Data analyst project\\S&P 500 stock retu
    r'C:\\Users\\Rahul\\OneDrive\\Desktop\\Data analyst project\\S&P 500 stock retu
    r'C:\\Users\\Rahul\\OneDrive\\Desktop\\Data analyst project\\S&P 500 stock retu
    r'C:\\Users\\Rahul\\OneDrive\\Desktop\\Data analyst project\\S&P 500 stock retu

]
```

```
In [ ]:
```

```
In [8]: #collect data from various files ..
```

```
In [9]: all_data = pd.DataFrame()

for file in company_list:

    current_df = pd.read_csv(file)

    all_data = current_df.append(all_data , ignore_index=True)
    ##full_df = pd.concat([full_df , current_df] , ignore_index=True)
```

```
C:\Users\Rahul\AppData\Local\Temp\ipykernel_15292\593578919.py:7: FutureWarning: The
frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
```

```
all_data = current_df.append(all_data , ignore_index=True)
```

```
C:\Users\Rahul\AppData\Local\Temp\ipykernel_15292\593578919.py:7: FutureWarning: The
frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
```

```
all_data = current_df.append(all_data , ignore_index=True)
```

```
C:\Users\Rahul\AppData\Local\Temp\ipykernel_15292\593578919.py:7: FutureWarning: The
frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
```

```
all_data = current_df.append(all_data , ignore_index=True)
```

```
C:\Users\Rahul\AppData\Local\Temp\ipykernel_15292\593578919.py:7: FutureWarning: The
frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
```

```
all_data = current_df.append(all_data , ignore_index=True)
```

```
In [10]: all_data.shape ## dimensions of all_data dataframe ..
```

```
Out[10]: (4752, 7)
```

```
In [11]: all_data.head(6)
```

```
Out[11]:
```

	date	open	high	low	close	volume	Name
0	2013-02-08	27.35	27.71	27.310	27.55	33318306	MSFT
1	2013-02-11	27.65	27.92	27.500	27.86	32247549	MSFT
2	2013-02-12	27.88	28.00	27.750	27.88	35990829	MSFT
3	2013-02-13	27.93	28.11	27.880	28.03	41715530	MSFT
4	2013-02-14	27.92	28.06	27.870	28.04	32663174	MSFT
5	2013-02-15	28.04	28.16	27.875	28.01	49650538	MSFT

```
In [12]: all_data['Name'].unique()
```

```
Out[12]: array(['MSFT', 'GOOG', 'AMZN', 'AAPL'], dtype=object)
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [13]: #2.. Analysing change in price of the stock overtime
```

```
In [14]: all_data.isnull().sum() ## checking missing values
```

```
Out[14]: date      0
open      0
high      0
low       0
close     0
volume    0
Name      0
dtype: int64
```

```
In [15]: all_data.dtypes ## checking data-types
```

```
Out[15]: date      object
open      float64
high      float64
low       float64
close     float64
volume    int64
Name      object
dtype: object
```

```
In [16]: all_data['date'] = pd.to_datetime(all_data['date']) ## converting data-type of "date"
```

```
In [17]: all_data['date']
```

```

Out[17]: 0      2013-02-08
         1      2013-02-11
         2      2013-02-12
         3      2013-02-13
         4      2013-02-14
         ...
        4747    2018-02-01
        4748    2018-02-02
        4749    2018-02-05
        4750    2018-02-06
        4751    2018-02-07
        Name: date, Length: 4752, dtype: datetime64[ns]

```

```
In [18]: tech_list = all_data['Name'].unique()
```

```
In [19]: tech_list
```

```
Out[19]: array(['MSFT', 'GOOG', 'AMZN', 'AAPL'], dtype=object)
```

```
In [ ]:
```

```

In [20]: plt.figure(figsize=(20,12))

for index , company in enumerate(tech_list , 1):
    plt.subplot(2 , 2 , index) ## creating subplot for each stock
    filter1 = all_data['Name']==company
    df = all_data[filter1]
    plt.plot(df['date'] , df['close']) ## plotting "date" vs "close"
    plt.title(company)

```



```
In [ ]:
```

In []:

In []:

In [21]: *#3.. moving average of the various stocks*In [22]: `all_data.head(15)`

Out[22]:

	date	open	high	low	close	volume	Name
0	2013-02-08	27.3500	27.71	27.310	27.550	33318306	MSFT
1	2013-02-11	27.6500	27.92	27.500	27.860	32247549	MSFT
2	2013-02-12	27.8800	28.00	27.750	27.880	35990829	MSFT
3	2013-02-13	27.9300	28.11	27.880	28.030	41715530	MSFT
4	2013-02-14	27.9200	28.06	27.870	28.040	32663174	MSFT
5	2013-02-15	28.0400	28.16	27.875	28.010	49650538	MSFT
6	2013-02-19	27.8801	28.09	27.800	28.045	38804616	MSFT
7	2013-02-20	28.1300	28.20	27.830	27.870	44109412	MSFT
8	2013-02-21	27.7400	27.74	27.230	27.490	49078338	MSFT
9	2013-02-22	27.6800	27.76	27.480	27.760	31425726	MSFT
10	2013-02-25	27.9700	28.05	27.370	27.370	48011248	MSFT
11	2013-02-26	27.3800	27.60	27.340	27.370	49917353	MSFT
12	2013-02-27	27.4200	28.00	27.330	27.810	36390889	MSFT
13	2013-02-28	27.8800	27.97	27.740	27.800	35836861	MSFT
14	2013-03-01	27.7200	27.98	27.520	27.950	34849287	MSFT

In []:

In [23]: `all_data['close'].rolling(window=10).mean().head(14)`

```
Out[23]: 0      NaN
         1      NaN
         2      NaN
         3      NaN
         4      NaN
         5      NaN
         6      NaN
         7      NaN
         8      NaN
         9    27.8535
        10    27.8355
        11    27.7865
        12    27.7795
        13    27.7565
        Name: close, dtype: float64
```

```
In [24]: new_data = all_data.copy()
```

```
In [25]: ##### now Lets consider different windows of rolling ,ie 10 days ,20 days ,30 days
        ma_day = [10 ,20 , 50]

        for ma in ma_day:
            new_data['close_'+str(ma)] = new_data['close'].rolling(ma).mean()
```

```
In [26]: new_data.tail(7)
```

```
Out[26]:
```

	date	open	high	low	close	volume	Name	close_10	close_20	clo
4745	2018-01-30	165.525	167.3700	164.7000	166.97	46048185	AAPL	174.263	174.3340	174
4746	2018-01-31	166.870	168.4417	166.5000	167.43	32478930	AAPL	173.096	174.0925	174
4747	2018-02-01	167.165	168.6200	166.7600	167.78	47230787	AAPL	171.948	173.8700	174
4748	2018-02-02	166.000	166.8000	160.1000	160.50	86593825	AAPL	170.152	173.2435	174
4749	2018-02-05	159.100	163.8800	156.0000	156.49	72738522	AAPL	168.101	172.3180	174
4750	2018-02-06	154.830	163.7200	154.0000	163.03	68243838	AAPL	166.700	171.7520	174
4751	2018-02-07	163.085	163.4000	159.0685	159.54	51608580	AAPL	165.232	171.0125	174

```
In [ ]:
```

```
In [27]: new_data.set_index('date' , inplace=True)
```


In [28]: new_data

Out[28]:

	open	high	low	close	volume	Name	close_10	close_20	close_50
date									
2013-02-08	27.350	27.71	27.3100	27.55	33318306	MSFT	NaN	NaN	NaN
2013-02-11	27.650	27.92	27.5000	27.86	32247549	MSFT	NaN	NaN	NaN
2013-02-12	27.880	28.00	27.7500	27.88	35990829	MSFT	NaN	NaN	NaN
2013-02-13	27.930	28.11	27.8800	28.03	41715530	MSFT	NaN	NaN	NaN
2013-02-14	27.920	28.06	27.8700	28.04	32663174	MSFT	NaN	NaN	NaN
...
2018-02-01	167.165	168.62	166.7600	167.78	47230787	AAPL	171.948	173.8700	172.8252
2018-02-02	166.000	166.80	160.1000	160.50	86593825	AAPL	170.152	173.2435	172.6356
2018-02-05	159.100	163.88	156.0000	156.49	72738522	AAPL	168.101	172.3180	172.3026
2018-02-06	154.830	163.72	154.0000	163.03	68243838	AAPL	166.700	171.7520	172.0640
2018-02-07	163.085	163.40	159.0685	159.54	51608580	AAPL	165.232	171.0125	171.7554

4752 rows × 9 columns

In [29]: new_data.columns

Out[29]: Index(['open', 'high', 'low', 'close', 'volume', 'Name', 'close_10', 'close_20', 'close_50'], dtype='object')

```
In [30]: plt.figure(figsize=(20,12))

for index , company in enumerate(tech_list , 1):
    plt.subplot(2 , 2 , index)
    filter1 = new_data['Name']==company
    df = new_data[filter1]
    df[['close_10','close_20', 'close_50']].plot(ax=plt.gca())
    plt.title(company)
```



In []:

In []:

In []:

In [31]: *#4.. analyse Closing price change in apple stock !*

In [32]: *#Daily Stock Return Formula*
#To calculate how much you gained or lost per day for a stock, subtract the opening

In [33]: company_list

Out[33]: ['C:\\\\Users\\\\Rahul\\\\OneDrive\\\\Desktop\\\\Data analyst project\\\\S&P 500 s
 tock return\\\\individual_stocks_5yr\\\\AAPL_data.csv',
 'C:\\\\Users\\\\Rahul\\\\OneDrive\\\\Desktop\\\\Data analyst project\\\\S&P 500 s
 tock return\\\\individual_stocks_5yr\\\\AMZN_data.csv',
 'C:\\\\Users\\\\Rahul\\\\OneDrive\\\\Desktop\\\\Data analyst project\\\\S&P 500 s
 tock return\\\\individual_stocks_5yr\\\\GOOG_data.csv',
 'C:\\\\Users\\\\Rahul\\\\OneDrive\\\\Desktop\\\\Data analyst project\\\\S&P 500 s
 tock return\\\\individual_stocks_5yr\\\\MSFT_data.csv']

In [34]: apple = pd.read_csv(r'C:\\\\Users\\\\Rahul\\\\OneDrive\\\\Desktop\\\\Data analyst p

In [35]: apple.head(4)

Out[35]:

	date	open	high	low	close	volume	Name
0	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL
1	2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL
2	2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL
3	2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL

In [36]: `apple['close']`

Out[36]:

0	67.8542
1	68.5614
2	66.8428
3	66.7156
4	66.6556
...	
1254	167.7800
1255	160.5000
1256	156.4900
1257	163.0300
1258	159.5400

Name: close, Length: 1259, dtype: float64

In [37]: `apple.head(4)`

Out[37]:

	date	open	high	low	close	volume	Name
0	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL
1	2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL
2	2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL
3	2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL

In [38]: `apple['Daily return(in %)'] = apple['close'].pct_change() * 100`

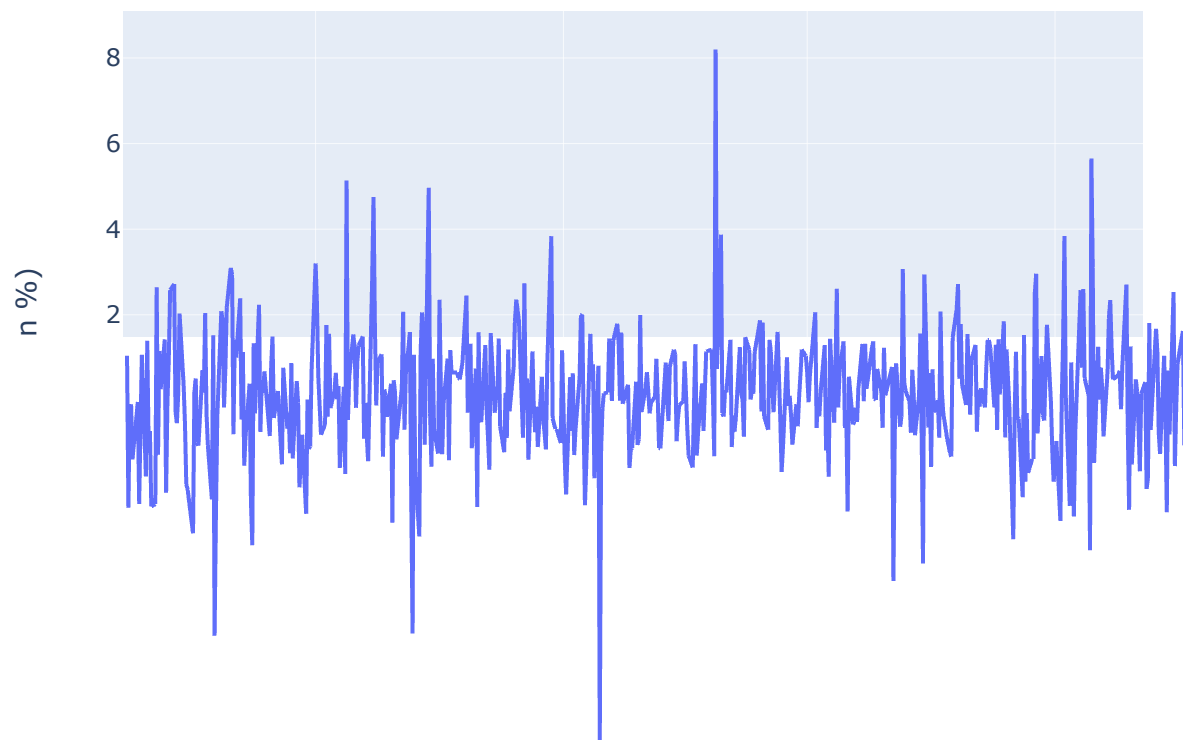
pct_change() returns : Percentage change between the current and a prior elemen

In [39]: `apple.head(4)`

Out[39]:

	date	open	high	low	close	volume	Name	Daily return(in %)
0	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL	NaN
1	2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL	1.042235
2	2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL	-2.506658
3	2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL	-0.190297

In []:

In [40]: `import plotly.express as px`In [41]: `px.line(apple , x="date" , y="Daily return(in %)") ## Plotting Line-plot of "date"`

```
In [42]: #5.. Performing resampling analysis of closing price
```

```
In [43]: #Before doing resampling,first u have to make your date feature 'row-index' so that

#a..yearly('Y') ,
#b..quarterly('Q') ,
#c..monthly('M') ,
#d..weekly basis ('W'),
#e..Daily_basis('D')
#f..minutes ('3T') ,
#g..30 second bins('30S') ,
#h..resample('17min')
```

```
In [44]: apple.dtypes
```

```
Out[44]: date                object
open                float64
high                float64
low                 float64
close               float64
volume              int64
Name                object
Daily return(in %)  float64
dtype: object
```

```
In [45]: apple['date'] =pd.to_datetime(apple['date'])
```

```
In [46]: apple.dtypes
```

```
Out[46]: date                datetime64[ns]
open                float64
high                float64
low                 float64
close               float64
volume              int64
Name                object
Daily return(in %)  float64
dtype: object
```

```
In [47]: apple.head(4)
```

Out[47]:

	date	open	high	low	close	volume	Name	Daily return(in %)
0	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL	NaN
1	2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL	1.042235
2	2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL	-2.506658
3	2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL	-0.190297

In [48]: `apple.set_index('date', inplace=True)`In [49]: `apple.head(4)`

Out[49]:

	open	high	low	close	volume	Name	Daily return(in %)
date							
2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL	NaN
2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL	1.042235
2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL	-2.506658
2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL	-0.190297

In [50]: `apple['close'].resample('M').mean() ## resample data on monthly basis ..`

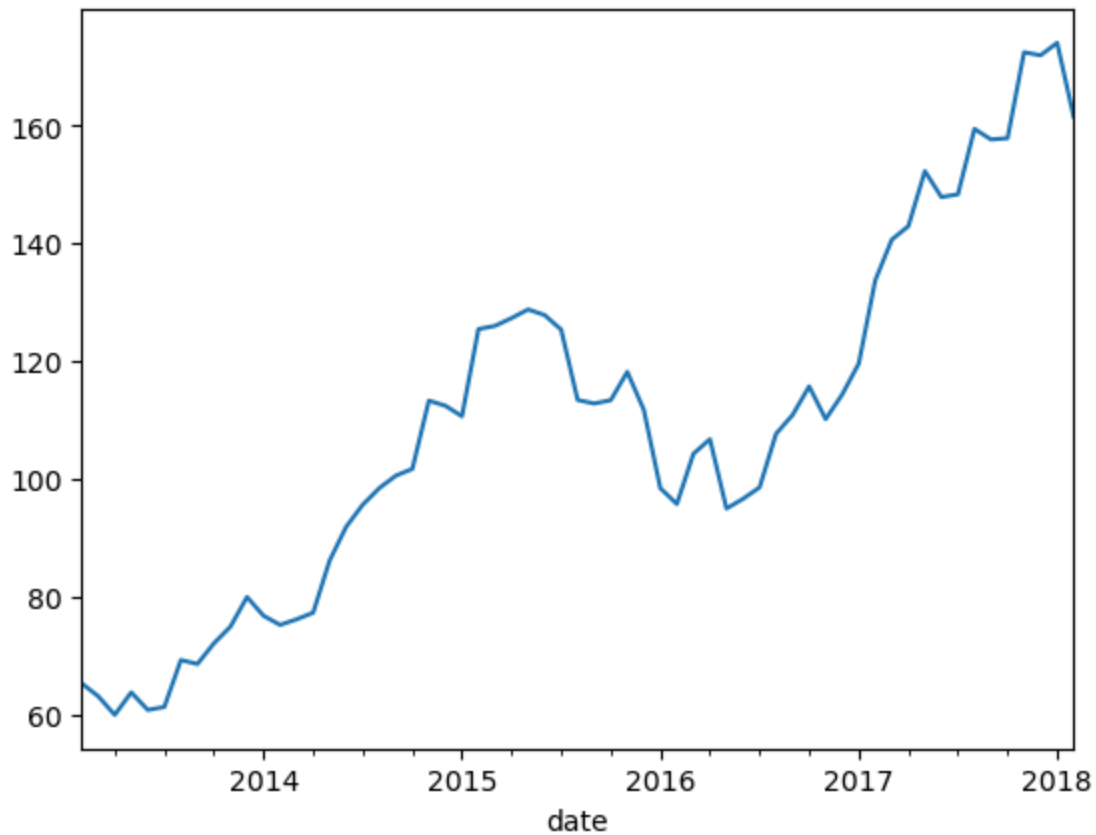
Out[50]:

```

date
2013-02-28    65.306264
2013-03-31    63.120110
2013-04-30    59.966432
2013-05-31    63.778927
2013-06-30    60.791120
...
2017-10-31   157.817273
2017-11-30   172.406190
2017-12-31   171.891500
2018-01-31   174.005238
2018-02-28   161.468000
Freq: M, Name: close, Length: 61, dtype: float64

```

In [51]: `apple['close'].resample('M').mean().plot()`Out[51]: `<AxesSubplot:xlabel='date'>`



In []:

In [52]: `apple['close'].resample('Y').mean()` *## resample data on Yearly basis ..*

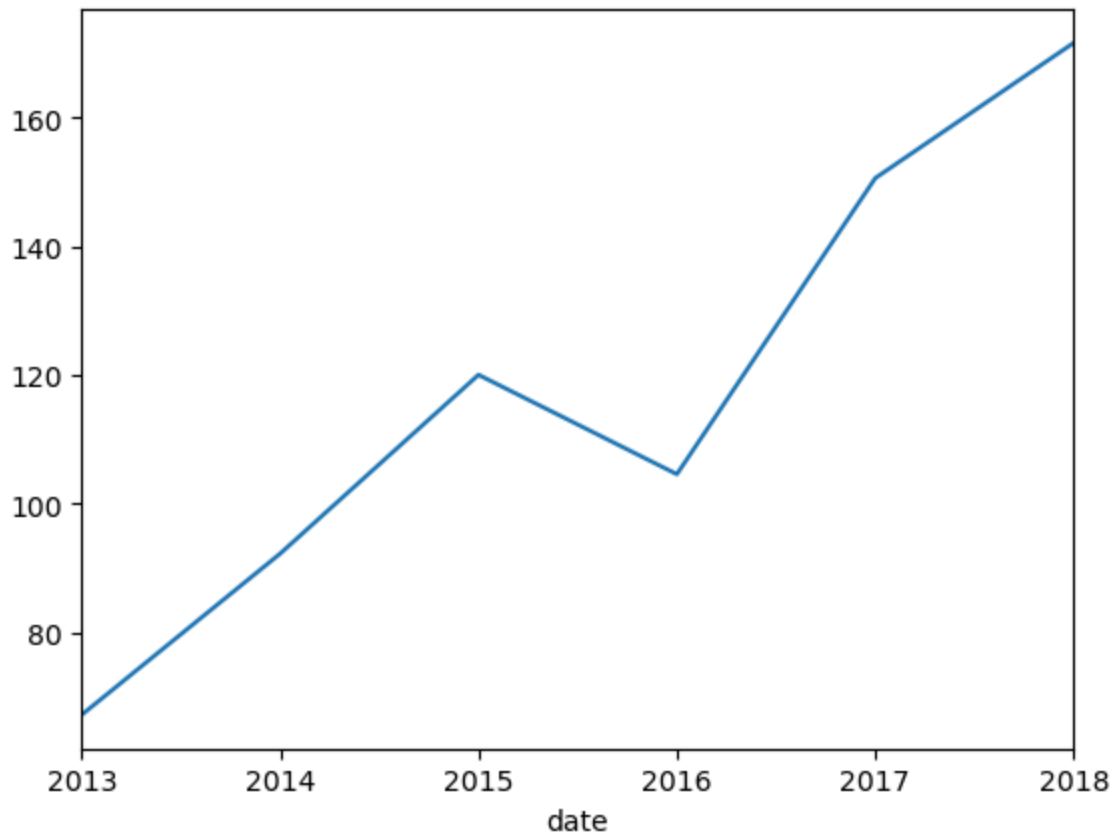
Out[52]:

date	
2013-12-31	67.237839
2014-12-31	92.264531
2015-12-31	120.039861
2016-12-31	104.604008
2017-12-31	150.585080
2018-12-31	171.594231

Freq: A-DEC, Name: close, dtype: float64

In [53]: `apple['close'].resample('Y').mean().plot()`

Out[53]: `<AxesSubplot:xlabel='date'>`

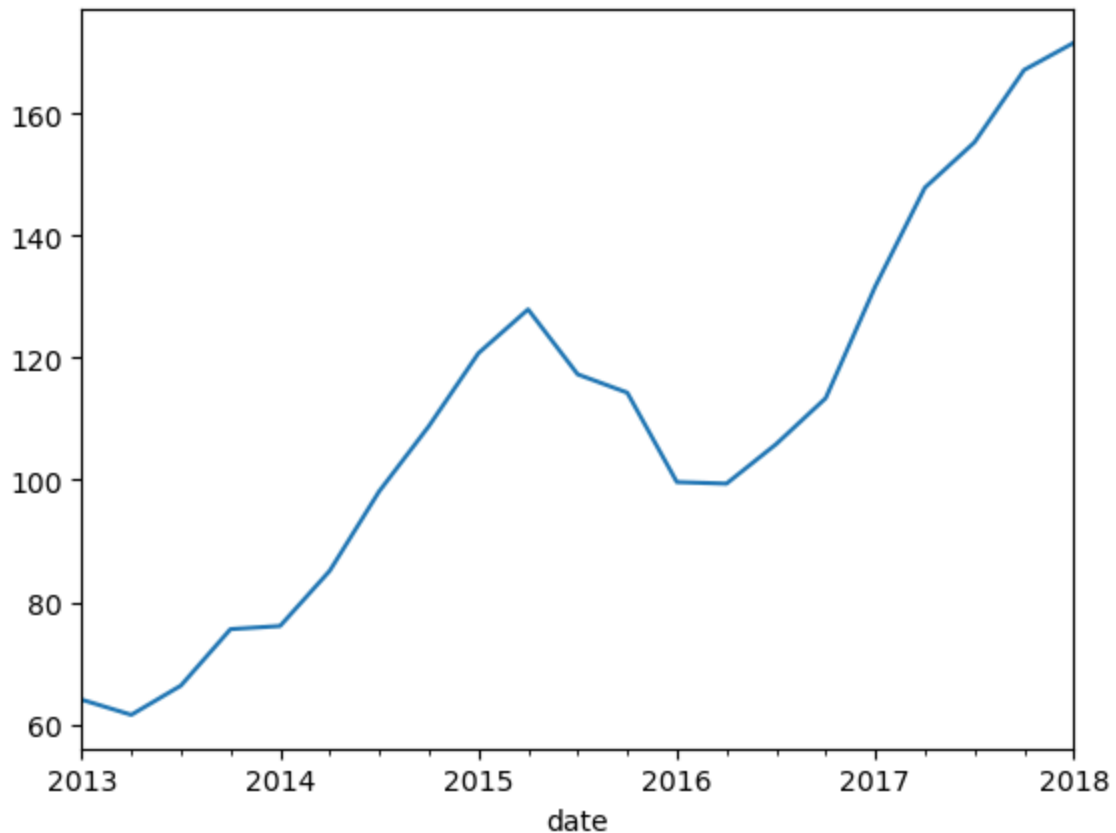


```
In [54]: apple['close'].resample('Q').mean() ## resample data on Quarterly basis ..
```

```
Out[54]: date
2013-03-31    64.020291
2013-06-30    61.534692
2013-09-30    66.320670
2013-12-31    75.567478
2014-03-31    76.086293
2014-06-30    85.117475
2014-09-30    98.163311
2014-12-31   108.821016
2015-03-31   120.776721
2015-06-30   127.937937
2015-09-30   117.303438
2015-12-31   114.299297
2016-03-31    99.655082
2016-06-30    99.401250
2016-09-30   105.866094
2016-12-31   113.399048
2017-03-31   131.712500
2017-06-30   147.875397
2017-09-30   155.304603
2017-12-31   167.148254
2018-03-31   171.594231
Freq: Q-DEC, Name: close, dtype: float64
```

```
In [55]: apple['close'].resample('Q').mean().plot()
```

```
Out[55]: <AxesSubplot:xlabel='date'>
```

In []:

In []:

In []:

In [56]: *#6.. Whether closing prices of these tech companies (Amazon,Apple,Google,Microsoft)*

In [57]: company_list

```
Out[57]: ['C:\\\\Users\\\\Rahul\\\\OneDrive\\\\Desktop\\\\Data analyst project\\\\S&P 500 s
tock return\\\\individual_stocks_5yr\\\\AAPL_data.csv',
'C:\\\\Users\\\\Rahul\\\\OneDrive\\\\Desktop\\\\Data analyst project\\\\S&P 500 s
tock return\\\\individual_stocks_5yr\\\\AMZN_data.csv',
'C:\\\\Users\\\\Rahul\\\\OneDrive\\\\Desktop\\\\Data analyst project\\\\S&P 500 s
tock return\\\\individual_stocks_5yr\\\\GOOG_data.csv',
'C:\\\\Users\\\\Rahul\\\\OneDrive\\\\Desktop\\\\Data analyst project\\\\S&P 500 s
tock return\\\\individual_stocks_5yr\\\\MSFT_data.csv']
```

In [58]: company_list[0]

```
Out[58]: 'C:\\\\Users\\\\Rahul\\\\OneDrive\\\\Desktop\\\\Data analyst project\\\\S&P 500 st
ock return\\\\individual_stocks_5yr\\\\AAPL_data.csv'
```

```
In [59]: app = pd.read_csv(company_list[0])
amzn = pd.read_csv(company_list[1])
google = pd.read_csv(company_list[2])
msft = pd.read_csv(company_list[3])
```

```
In [60]: closing_price = pd.DataFrame()
```

```
In [62]: closing_price['apple_close'] = app['close']
closing_price['amzn_close'] = amzn['close']
closing_price['goog_close'] = google['close']
closing_price['msft_close'] = msft['close']
```

```
In [63]: closing_price
```

```
Out[63]:
```

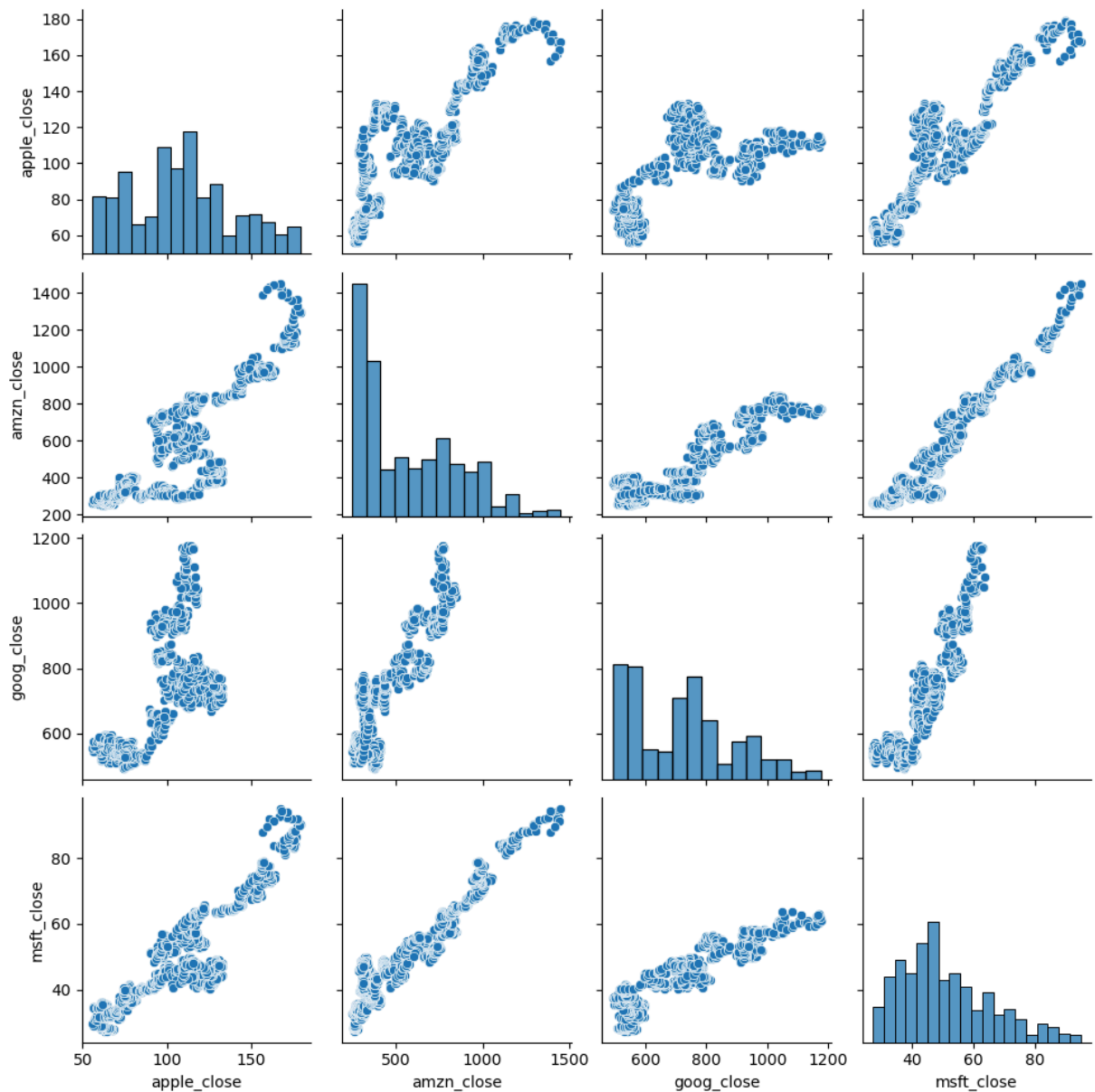
	apple_close	amzn_close	goog_close	msft_close
0	67.8542	261.95	558.46	27.55
1	68.5614	257.21	559.99	27.86
2	66.8428	258.70	556.97	27.88
3	66.7156	269.47	567.16	28.03
4	66.6556	269.24	567.00	28.04
...
1254	167.7800	1390.00	NaN	94.26
1255	160.5000	1429.95	NaN	91.78
1256	156.4900	1390.00	NaN	88.00
1257	163.0300	1442.84	NaN	91.33
1258	159.5400	1416.78	NaN	89.61

1259 rows × 4 columns

```
In [64]: #Pair-plot is all about , we can considering some pairs &
#we are trying to plot scatterplot of it..
```

```
In [65]: sns.pairplot(closing_price)
```

```
Out[65]: <seaborn.axisgrid.PairGrid at 0x22705514520>
```



In []:

In []:

In [66]: `closing_price.corr()`

Out[66]:

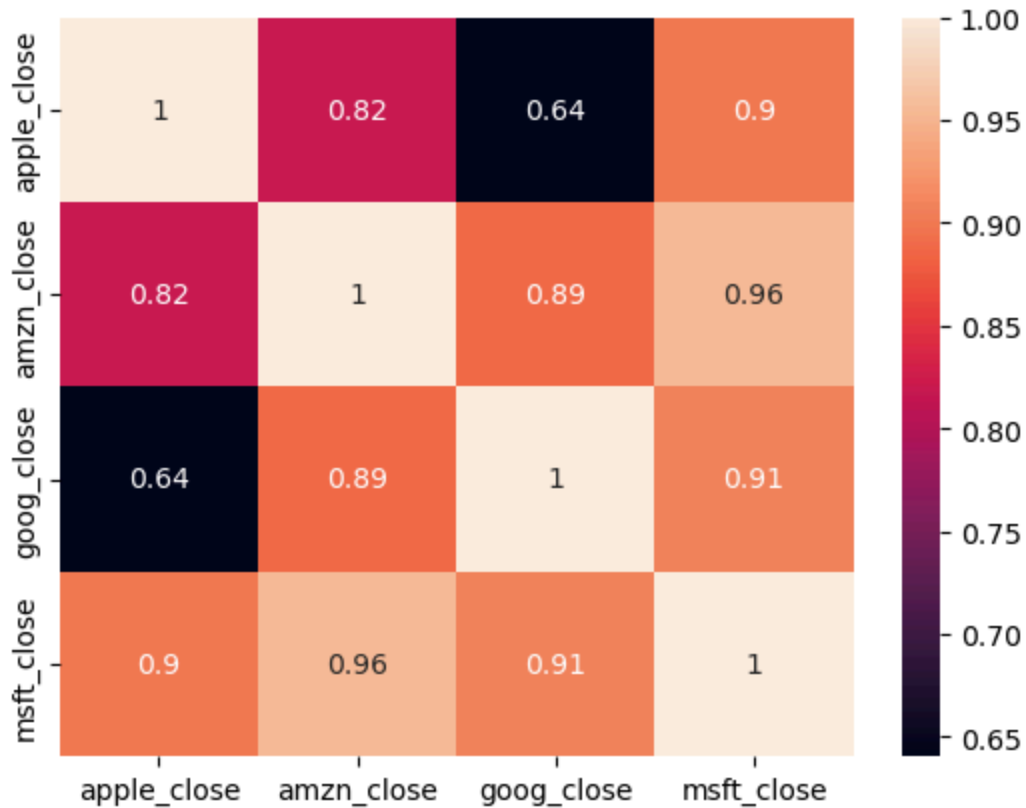
	apple_close	amzn_close	goog_close	msft_close
apple_close	1.000000	0.819078	0.640522	0.899689
amzn_close	0.819078	1.000000	0.888456	0.955977
goog_close	0.640522	0.888456	1.000000	0.907011
msft_close	0.899689	0.955977	0.907011	1.000000

In []:

In [67]: `#co-relation plot for stock prices`

In [68]: `sns.heatmap(closing_price.corr() , annot=True)`

Out[68]: `<AxesSubplot:>`



In [69]: `#Conclusions :`
`#Closing price of Google and Microsoft are well correlated`
`#& Closing price of Amazon and Microsoft have a co-relation of 0.96`

In []:

In [70]: `#7.. analyse Whether Daily change in Closing price of stocks or Daily Returns in St`

In [71]: `closing_price`

Out[71]:

	apple_close	amzn_close	goog_close	msft_close
0	67.8542	261.95	558.46	27.55
1	68.5614	257.21	559.99	27.86
2	66.8428	258.70	556.97	27.88
3	66.7156	269.47	567.16	28.03
4	66.6556	269.24	567.00	28.04
...
1254	167.7800	1390.00	NaN	94.26
1255	160.5000	1429.95	NaN	91.78
1256	156.4900	1390.00	NaN	88.00
1257	163.0300	1442.84	NaN	91.33
1258	159.5400	1416.78	NaN	89.61

1259 rows × 4 columns

In [72]: closing_price['apple_close']

Out[72]:

0	67.8542
1	68.5614
2	66.8428
3	66.7156
4	66.6556
...	...
1254	167.7800
1255	160.5000
1256	156.4900
1257	163.0300
1258	159.5400

Name: apple_close, Length: 1259, dtype: float64

In [73]: closing_price['apple_close'].shift(1)

Out[73]:

0	NaN
1	67.8542
2	68.5614
3	66.8428
4	66.7156
...	...
1254	167.4300
1255	167.7800
1256	160.5000
1257	156.4900
1258	163.0300

Name: apple_close, Length: 1259, dtype: float64

In [74]: (closing_price['apple_close'] - closing_price['apple_close'].shift(1))/closing_pric

```
Out[74]: 0      NaN
         1      1.042235
         2     -2.506658
         3     -0.190297
         4     -0.089934
         ...
        1254    0.209043
        1255   -4.339015
        1256   -2.498442
        1257    4.179181
        1258   -2.140710
        Name: apple_close, Length: 1259, dtype: float64
```

```
In [75]: for col in closing_price.columns:
         closing_price[col + '_pct_change'] = (closing_price[col] - closing_price[col].s
```

```
In [76]: closing_price
```

```
Out[76]:
```

	apple_close	amzn_close	goog_close	msft_close	apple_close_pct_change	amzn_close
0	67.8542	261.95	558.46	27.55	NaN	
1	68.5614	257.21	559.99	27.86	1.042235	
2	66.8428	258.70	556.97	27.88	-2.506658	
3	66.7156	269.47	567.16	28.03	-0.190297	
4	66.6556	269.24	567.00	28.04	-0.089934	
...
1254	167.7800	1390.00	NaN	94.26	0.209043	
1255	160.5000	1429.95	NaN	91.78	-4.339015	
1256	156.4900	1390.00	NaN	88.00	-2.498442	
1257	163.0300	1442.84	NaN	91.33	4.179181	
1258	159.5400	1416.78	NaN	89.61	-2.140710	

1259 rows × 8 columns



```
In [77]: closing_price.columns
```

```
Out[77]: Index(['apple_close', 'amzn_close', 'goog_close', 'msft_close',
               'apple_close_pct_change', 'amzn_close_pct_change',
               'goog_close_pct_change', 'msft_close_pct_change'],
              dtype='object')
```

```
In [78]: clsing_p = closing_price[['apple_close_pct_change', 'amzn_close_pct_change',
                                   'goog_close_pct_change', 'msft_close_pct_change']]
```

```
In [79]: clsing_p
```

Out[79]:

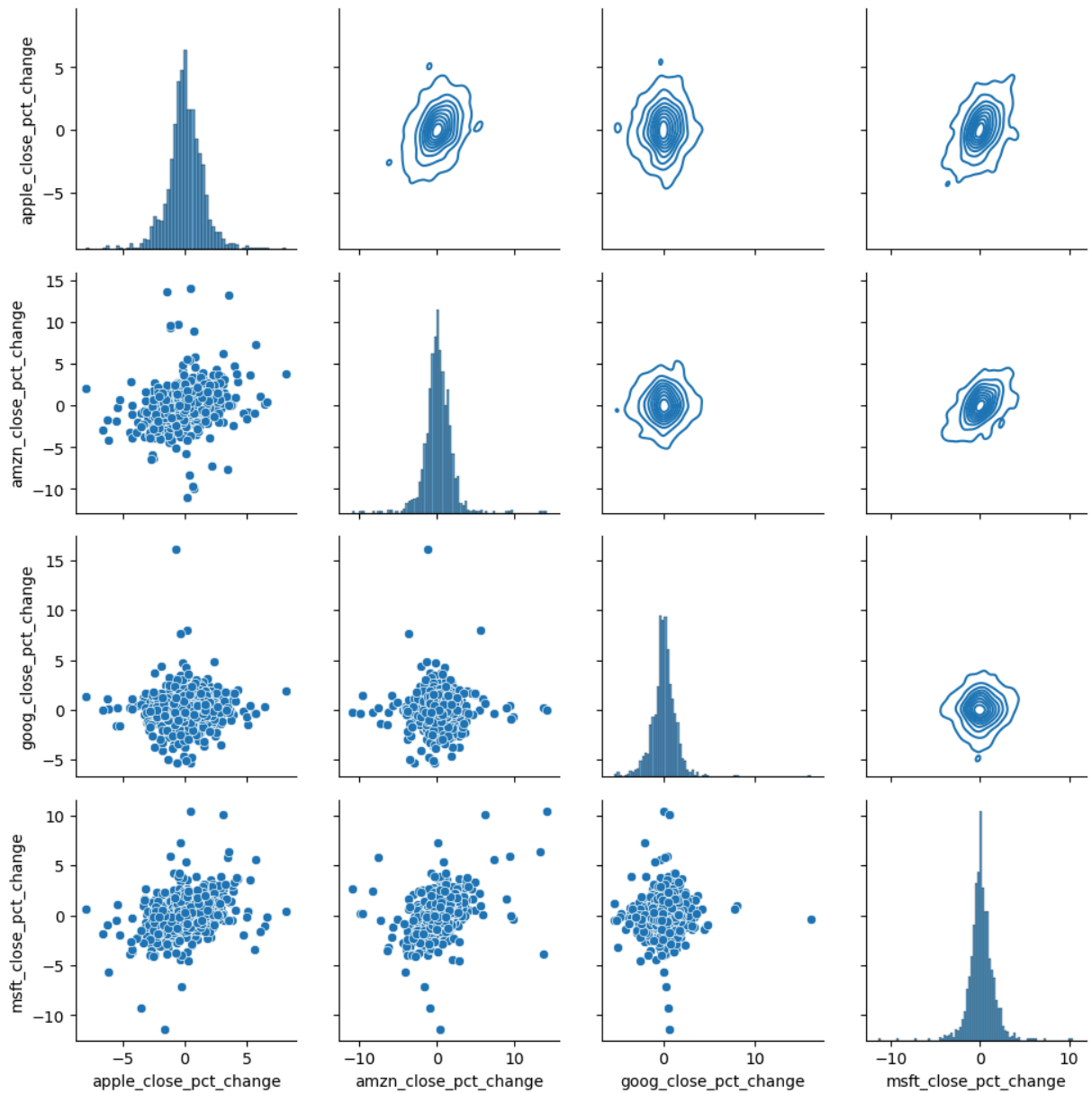
	apple_close_pct_change	amzn_close_pct_change	goog_close_pct_change	msft_close_pct_change
0	NaN	NaN	NaN	NaN
1	1.042235	-1.809506	0.273968	NaN
2	-2.506658	0.579293	-0.539295	NaN
3	-0.190297	4.163123	1.829542	NaN
4	-0.089934	-0.085353	-0.028211	NaN
...
1254	0.209043	-4.196734	NaN	NaN
1255	-4.339015	2.874101	NaN	NaN
1256	-2.498442	-2.793804	NaN	NaN
1257	4.179181	3.801439	NaN	NaN
1258	-2.140710	-1.806160	NaN	NaN

1259 rows × 4 columns

In [80]: *### since we have used Pairplot already , Lets use extension of Pairplot , ie Pairgrid*In [81]: *#Pairplot : we have histogram on diagonals & scatterplot/kde/any_other_plot which t*In [82]: *#Pairgrid : Once we create grid , we can set plot as per our need :*In [83]: *#ie , if we have 4 features , it creates total 16 graphs/plots or matrices of 4*4*

```
In [84]: g = sns.PairGrid(data= clsing_p)
g.map_diag(sns.histplot)
g.map_lower(sns.scatterplot)
g.map_upper(sns.kdeplot)
```

Out[84]: <seaborn.axisgrid.PairGrid at 0x22706510ca0>



In [85]: *#Conclusion :
#While Comparing 'AAPL_close_pct_change' to 'AMZN_close_pct_change' , it shows a L*

In [86]: `clsing_p.corr()`

Out[86]:

	apple_close_pct_change	amzn_close_pct_change	goog_close_pct_change	msft_close_pct_change
apple_close_pct_change	1.000000	0.287659	0.036202	0.366598
amzn_close_pct_change	0.287659	1.000000	0.027698	0.402678
goog_close_pct_change	0.036202	0.027698	1.000000	0.038
msft_close_pct_change	0.366598	0.402678	0.038	1.000000

In []:

In []:

In []: