STOCK MARKET

ANALYSIS USING PYTHON

GROUP MEMBERS:

Rajarshi Bhowmik, G.C.E.C.T, REG NO: 161130110058 of 2016-17
Somnath Chakraborty , G.C.E.C.T, REG NO: 161130110068 of 2016-17
Swarnendu Biswas, G.C.E.C.T, REG NO: 161130110073 of 2016-17

TABLE OF CONTENTS

- Acknowledgement
- Project Objective
- Project Scope
- Data Description
- Data Loading
- Interpreting the Data
- Data Cleaning & Munging
- Analyzing the Data Based on Various Parameters
- Project Certificate

Document sign date :19-Sep-2019

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my faculty Professor TITAS ROY CHOWDHURY for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Swarnendu Biswas
Rajarshi Bhowmik
Somnath Chakraborty

PROJECT OBJECTIVE

The main objectives of stock market are:

1. **RAISING MONEY FOR BUSINESS:** Stock exchanges around the world enable companies around the world to raise money. Nowadays, they're mostly electronic markets where licensed stock brokerages, and the traders representing them, buy and sell shares. Through exchanges, private companies sell stock in the form of publicly traded shares. Those wishing to invest in stock place buy or sell orders through regulated brokerage firms.
2. **CAPITAL FORMATION:** The primary function of a stock exchange is to help companies raise money. To accomplish this task, ownership in a private corporation is sold to the public in the form of shares of stock. Funds received from the sale of stock contribute to the firm's capital formation. Companies plan to use the newly-raised funds to invest in productive business assets and grow revenues and profits. This positive business expansion then may be reflected in a higher stock trading price.
3. **SECURITY AND TRANSPARENCY:** The legitimate sale of stock on any exchange requires reliable and accurate information. By requiring a high level of transparency from all trading companies, the stock exchange creates a more secure environment for investors, which helps them to determine the risks of investing.
4. **TRADING OF STOCKS:** An organized and regulated stock exchange facilitates the efficient trading of stock and other investment vehicles. Without this highly controlled and coordinated stock exchange, the global trading of stock would not be possible.

PROJECT SCOPE

Analysis of stocks using data mining will be useful for new investors to invest in stock market based on the various factors considered by the software. Stock market includes daily activities like sensex calculation, exchange of shares. The exchange provides an efficient and transparent market for trading in equity, debt instruments and derivatives.

Our software will be analyzing sensex based on company's stock value. The stock values of company depend on many factors, some of them are:

1> *Demand and Supply:*

Demand and Supply of shares of a company is a major reason price change in stocks. When Demand Increase and Supply is less, price rises. and vice versa.

2> *Corporate results:* This will be regarding to the profits or progress of the company over a span of time say 3 months.

Popularity of a company can effect on buyers. Like if any good news of a company, may result in rise of stock price. And a bad news may break dreams.

The stock value depends on other factors as well, but we are taking into consideration only

3> *Popularity:* Main Strength in hands of share buyer. these main factors.

PROMINENT FEATURES OF THE PROJECT:

Analyzing stock data.

We need to provide data of a particular company, and its Monthly Sales / Profit report with Months High and Low points of its Stock.

Analyzing the factors.

We have to obtain the data in the same period for the following factors.
1. Demand and Supply: We will obtain by the previous data entered.
2. Corporate results: Companies declare their performance results and profit at the end of each quarter.
3. Popularity: If any news about a company is about to come and is it bad or good.

We have to analyze the variations in the stock value of the companies with respect to these factors using some data mining algorithms.

DATA DESCRIPTION

For our project “Stock Market Analysis Using Python”, we have a dataset viz :

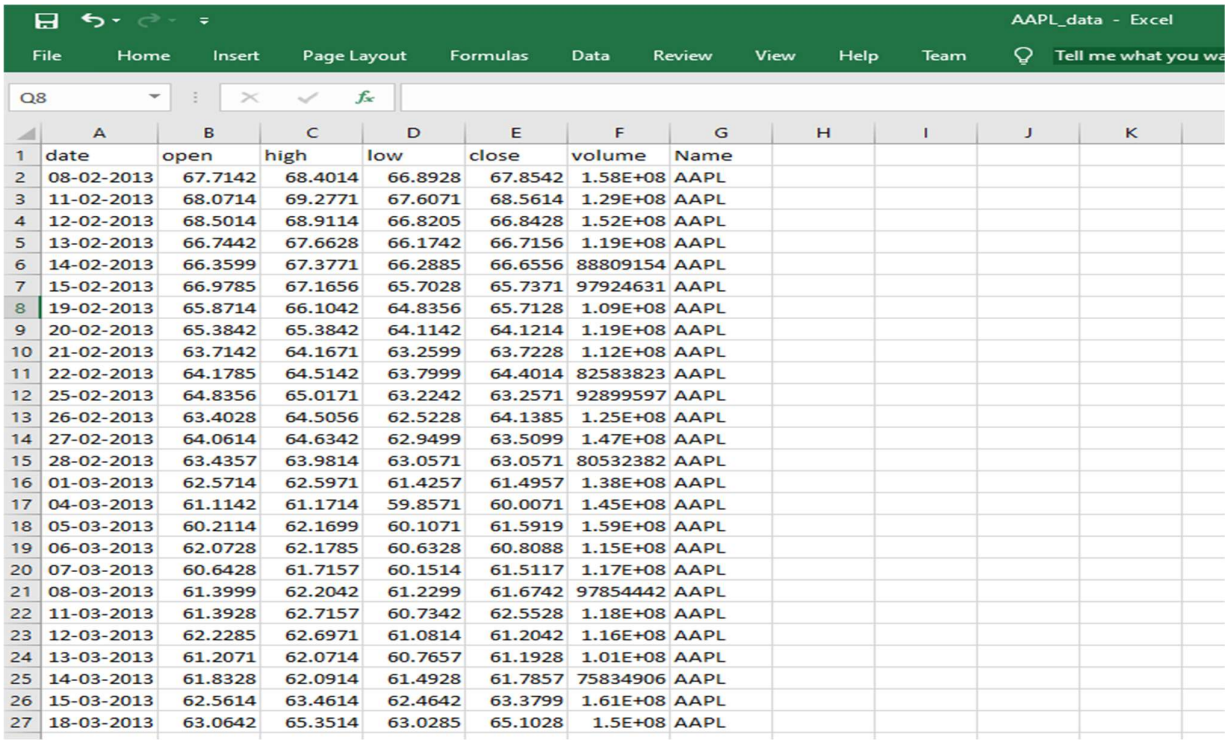
“AAPL.csv”,”AMZN.csv”,”GOOGL.csv”,”MSFT.csv”. This dataset contains 1259 records each containing 6 columns. Each column has its significance & meaning which will help us in our analysis. Among this, “Volume” column is the most important variable because trading **volume**

itself doesn't **affect stock price** directly, but it does have a huge impact on the way that **shares** move.

Document sign date :19-Sep-2019

DATA LOADING

The datasets “AAPL.csv”,“AMZN.csv”,“GOOGL.csv”, “MSFT.csv” on which we are working & will be doing analysis, is provided to us by our project guide, Titas sir.



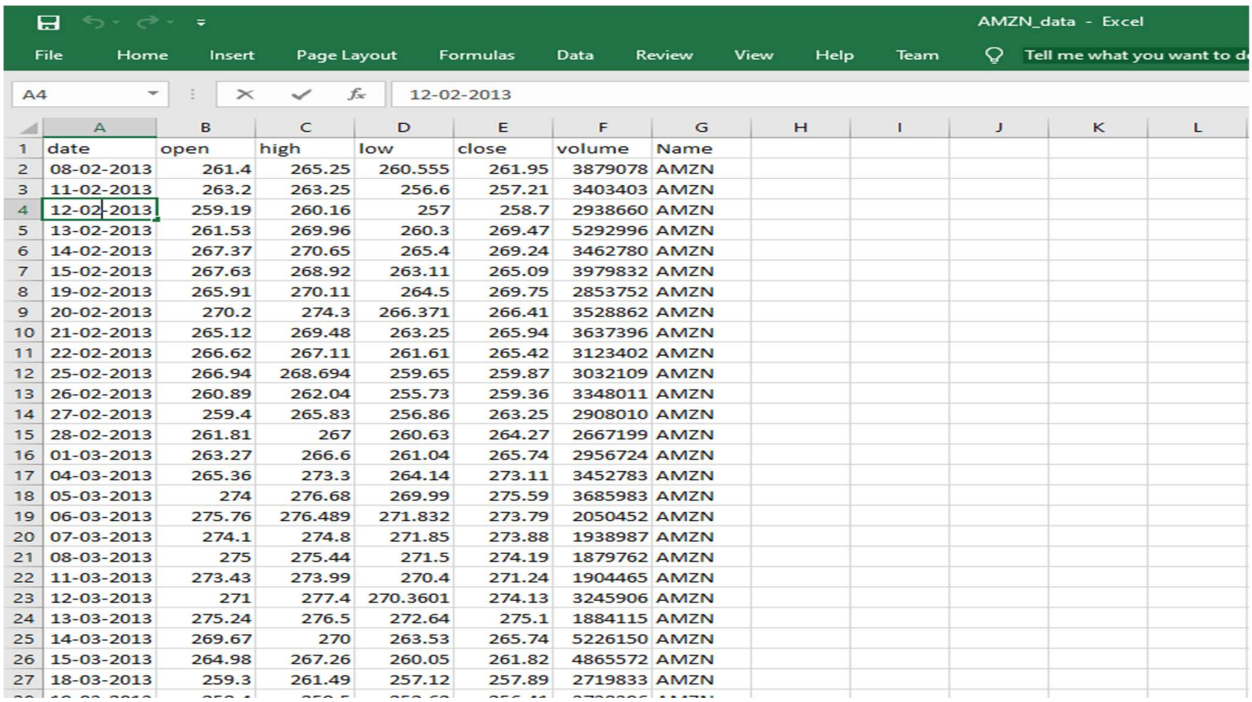
AAPL_data - Excel

File Home Insert Page Layout Formulas Data Review View Help Team Tell me what you wa

Q8

	A	B	C	D	E	F	G	H	I	J	K
1	date	open	high	low	close	volume	Name				
2	08-02-2013	67.7142	68.4014	66.8928	67.8542	1.58E+08	AAPL				
3	11-02-2013	68.0714	69.2771	67.6071	68.5614	1.29E+08	AAPL				
4	12-02-2013	68.5014	68.9114	66.8205	66.8428	1.52E+08	AAPL				
5	13-02-2013	66.7442	67.6628	66.1742	66.7156	1.19E+08	AAPL				
6	14-02-2013	66.3599	67.3771	66.2885	66.6556	88809154	AAPL				
7	15-02-2013	66.9785	67.1656	65.7028	65.7371	97924631	AAPL				
8	19-02-2013	65.8714	66.1042	64.8356	65.7128	1.09E+08	AAPL				
9	20-02-2013	65.3842	65.3842	64.1142	64.1214	1.19E+08	AAPL				
10	21-02-2013	63.7142	64.1671	63.2599	63.7228	1.12E+08	AAPL				
11	22-02-2013	64.1785	64.5142	63.7999	64.4014	82583823	AAPL				
12	25-02-2013	64.8356	65.0171	63.2242	63.2571	92899597	AAPL				
13	26-02-2013	63.4028	64.5056	62.5228	64.1385	1.25E+08	AAPL				
14	27-02-2013	64.0614	64.6342	62.9499	63.5099	1.47E+08	AAPL				
15	28-02-2013	63.4357	63.9814	63.0571	63.0571	80532382	AAPL				
16	01-03-2013	62.5714	62.5971	61.4257	61.4957	1.38E+08	AAPL				
17	04-03-2013	61.1142	61.1714	59.8571	60.0071	1.45E+08	AAPL				
18	05-03-2013	60.2114	62.1699	60.1071	61.5919	1.59E+08	AAPL				
19	06-03-2013	62.0728	62.1785	60.6328	60.8088	1.15E+08	AAPL				
20	07-03-2013	60.6428	61.7157	60.1514	61.5117	1.17E+08	AAPL				
21	08-03-2013	61.3999	62.2042	61.2299	61.6742	97854442	AAPL				
22	11-03-2013	61.3928	62.7157	60.7342	62.5528	1.18E+08	AAPL				
23	12-03-2013	62.2285	62.6971	61.0814	61.2042	1.16E+08	AAPL				
24	13-03-2013	61.2071	62.0714	60.7657	61.1928	1.01E+08	AAPL				
25	14-03-2013	61.8328	62.0914	61.4928	61.7857	75834906	AAPL				
26	15-03-2013	62.5614	63.4614	62.4642	63.3799	1.61E+08	AAPL				
27	18-03-2013	63.0642	65.3514	63.0285	65.1028	1.5E+08	AAPL				

Document sign date :19-Sep-2019

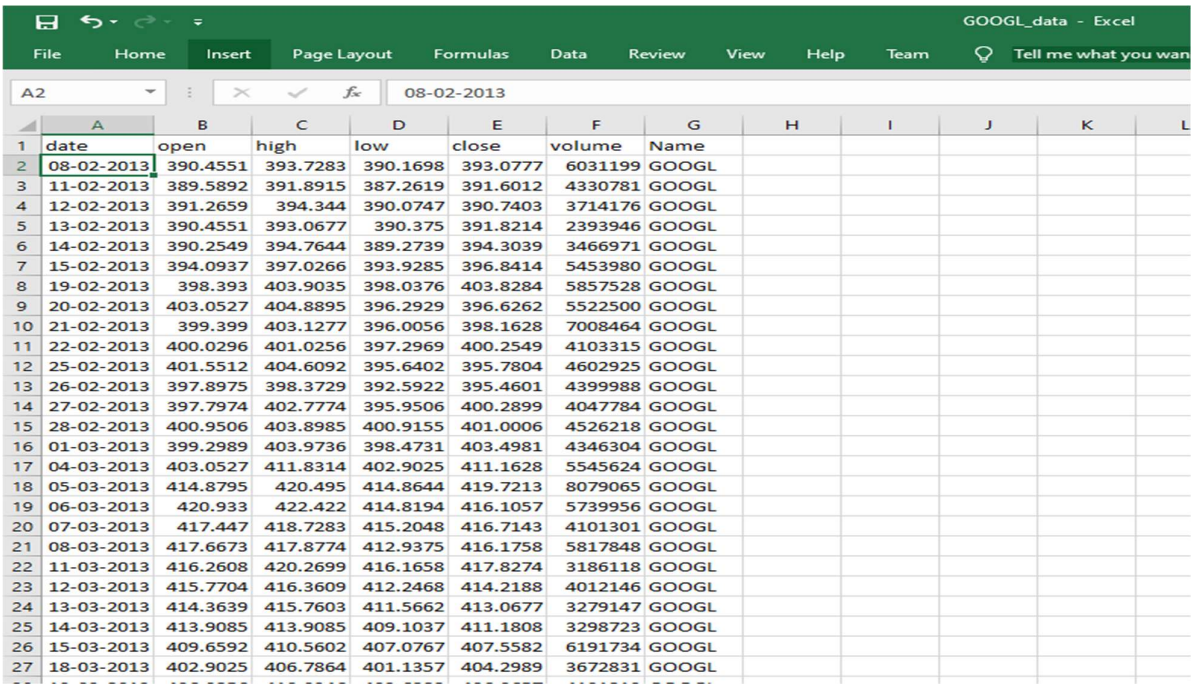


AMZN_data - Excel

File Home Insert Page Layout Formulas Data Review View Help Team Tell me what you want to d

A4 12-02-2013

	A	B	C	D	E	F	G
1	date	open	high	low	close	volume	Name
2	08-02-2013	261.4	265.25	260.555	261.95	3879078	AMZN
3	11-02-2013	263.2	263.25	256.6	257.21	3403403	AMZN
4	12-02-2013	259.19	260.16	257	258.7	2938660	AMZN
5	13-02-2013	261.53	269.96	260.3	269.47	5292996	AMZN
6	14-02-2013	267.37	270.65	265.4	269.24	3462780	AMZN
7	15-02-2013	267.63	268.92	263.11	265.09	3979832	AMZN
8	19-02-2013	265.91	270.11	264.5	269.75	2853752	AMZN
9	20-02-2013	270.2	274.3	266.371	266.41	3528862	AMZN
10	21-02-2013	265.12	269.48	263.25	265.94	3637396	AMZN
11	22-02-2013	266.62	267.11	261.61	265.42	3123402	AMZN
12	25-02-2013	266.94	268.694	259.65	259.87	3032109	AMZN
13	26-02-2013	260.89	262.04	255.73	259.36	3348011	AMZN
14	27-02-2013	259.4	265.83	256.86	263.25	2908010	AMZN
15	28-02-2013	261.81	267	260.63	264.27	2667199	AMZN
16	01-03-2013	263.27	266.6	261.04	265.74	2956724	AMZN
17	04-03-2013	265.36	273.3	264.14	273.11	3452783	AMZN
18	05-03-2013	274	276.68	269.99	275.59	3685983	AMZN
19	06-03-2013	275.76	276.489	271.832	273.79	2050452	AMZN
20	07-03-2013	274.1	274.8	271.85	273.88	1938987	AMZN
21	08-03-2013	275	275.44	271.5	274.19	1879762	AMZN
22	11-03-2013	273.43	273.99	270.4	271.24	1904465	AMZN
23	12-03-2013	271	277.4	270.3601	274.13	3245906	AMZN
24	13-03-2013	275.24	276.5	272.64	275.1	1884115	AMZN
25	14-03-2013	269.67	270	263.53	265.74	5226150	AMZN
26	15-03-2013	264.98	267.26	260.05	261.82	4865572	AMZN
27	18-03-2013	259.3	261.49	257.12	257.89	2719833	AMZN



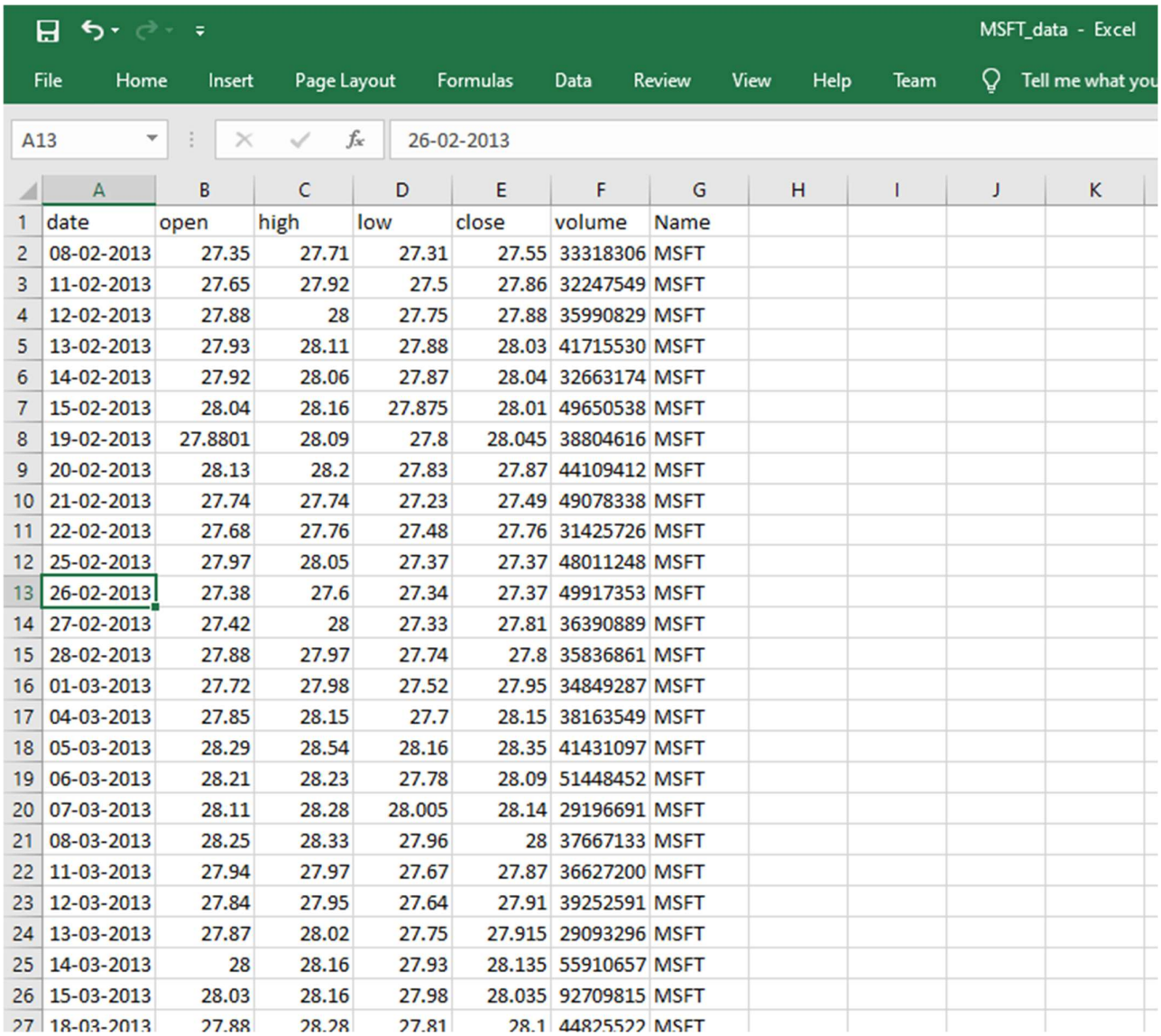
GOOGL_data - Excel

File Home Insert Page Layout Formulas Data Review View Help Team Tell me what you wan

A2 08-02-2013

	A	B	C	D	E	F	G
1	date	open	high	low	close	volume	Name
2	08-02-2013	390.4551	393.7283	390.1698	393.0777	6031199	GOOGL
3	11-02-2013	389.5892	391.8915	387.2619	391.6012	4330781	GOOGL
4	12-02-2013	391.2659	394.344	390.0747	390.7403	3714176	GOOGL
5	13-02-2013	390.4551	393.0677	390.375	391.8214	2393946	GOOGL
6	14-02-2013	390.2549	394.7644	389.2739	394.3039	3466971	GOOGL
7	15-02-2013	394.0937	397.0266	393.9285	396.8414	5453980	GOOGL
8	19-02-2013	398.393	403.9035	398.0376	403.8284	5857528	GOOGL
9	20-02-2013	403.0527	404.8895	396.2929	396.6262	5522500	GOOGL
10	21-02-2013	399.399	403.1277	396.0056	398.1628	7008464	GOOGL
11	22-02-2013	400.0296	401.0256	397.2969	400.2549	4103315	GOOGL
12	25-02-2013	401.5512	404.6092	395.6402	395.7804	4602925	GOOGL
13	26-02-2013	397.8975	398.3729	392.5922	395.4601	4399988	GOOGL
14	27-02-2013	397.7974	402.7774	395.9506	400.2899	4047784	GOOGL
15	28-02-2013	400.9506	403.8985	400.9155	401.0006	4526218	GOOGL
16	01-03-2013	399.2989	403.9736	398.4731	403.4981	4346304	GOOGL
17	04-03-2013	403.0527	411.8314	402.9025	411.1628	5545624	GOOGL
18	05-03-2013	414.8795	420.495	414.8644	419.7213	8079065	GOOGL
19	06-03-2013	420.933	422.422	414.8194	416.1057	5739956	GOOGL
20	07-03-2013	417.447	418.7283	415.2048	416.7143	4101301	GOOGL
21	08-03-2013	417.6673	417.8774	412.9375	416.1758	5817848	GOOGL
22	11-03-2013	416.2608	420.2699	416.1658	417.8274	3186118	GOOGL
23	12-03-2013	415.7704	416.3609	412.2468	414.2188	4012146	GOOGL
24	13-03-2013	414.3639	415.7603	411.5662	413.0677	3279147	GOOGL
25	14-03-2013	413.9085	413.9085	409.1037	411.1808	3298723	GOOGL
26	15-03-2013	409.6592	410.5602	407.0767	407.5582	6191734	GOOGL
27	18-03-2013	402.9025	406.7864	401.1357	404.2989	3672831	GOOGL

Document sign date :19-Sep-2019



	A	B	C	D	E	F	G	H	I	J	K
1	date	open	high	low	close	volume	Name				
2	08-02-2013	27.35	27.71	27.31	27.55	33318306	MSFT				
3	11-02-2013	27.65	27.92	27.5	27.86	32247549	MSFT				
4	12-02-2013	27.88	28	27.75	27.88	35990829	MSFT				
5	13-02-2013	27.93	28.11	27.88	28.03	41715530	MSFT				
6	14-02-2013	27.92	28.06	27.87	28.04	32663174	MSFT				
7	15-02-2013	28.04	28.16	27.875	28.01	49650538	MSFT				
8	19-02-2013	27.8801	28.09	27.8	28.045	38804616	MSFT				
9	20-02-2013	28.13	28.2	27.83	27.87	44109412	MSFT				
10	21-02-2013	27.74	27.74	27.23	27.49	49078338	MSFT				
11	22-02-2013	27.68	27.76	27.48	27.76	31425726	MSFT				
12	25-02-2013	27.97	28.05	27.37	27.37	48011248	MSFT				
13	26-02-2013	27.38	27.6	27.34	27.37	49917353	MSFT				
14	27-02-2013	27.42	28	27.33	27.81	36390889	MSFT				
15	28-02-2013	27.88	27.97	27.74	27.8	35836861	MSFT				
16	01-03-2013	27.72	27.98	27.52	27.95	34849287	MSFT				
17	04-03-2013	27.85	28.15	27.7	28.15	38163549	MSFT				
18	05-03-2013	28.29	28.54	28.16	28.35	41431097	MSFT				
19	06-03-2013	28.21	28.23	27.78	28.09	51448452	MSFT				
20	07-03-2013	28.11	28.28	28.005	28.14	29196691	MSFT				
21	08-03-2013	28.25	28.33	27.96	28	37667133	MSFT				
22	11-03-2013	27.94	27.97	27.67	27.87	36627200	MSFT				
23	12-03-2013	27.84	27.95	27.64	27.91	39252591	MSFT				
24	13-03-2013	27.87	28.02	27.75	27.915	29093296	MSFT				
25	14-03-2013	28	28.16	27.93	28.135	55910657	MSFT				
26	15-03-2013	28.03	28.16	27.98	28.035	92709815	MSFT				
27	18-03-2013	27.88	28.28	27.81	28.1	44825522	MSFT				

Document sign date :19-Sep-2019

INTERPRETING THE DATA

After downloading the data set, we have to understand each & every column's meaning & significance in the data set. The structural information of the columns is also our concern. We also understand the data type of the values of each column, unique values present in each column using python programming. We describe all the columns one by one here under -

1.Name- Under "Name" column,we come to know the names of the stocks which are analysed by us.

```
In [37]: Stock_AAPL["Name"].describe()

Out[37]: count     1259
         unique       1
         top       AAPL
         freq      1259
         Name: Name, dtype: object
```

Document sign date :19-Sep-2019

2.Closing price- The **closing price** is the final **price** at which a security is traded on a given trading day. The **closing price** represents the most up-to-date valuation of a security until trading commences again on the next trading day.

```
In [35]: Stock_AAPL["close"].describe()

Out[35]: count    1259.000000
         mean      109.066698
         std        30.556812
         min        55.789900
         25%        84.830650
         50%       109.010000
         75%       127.120000
         max       179.260000
         Name: close, dtype: float64
```

3.Opening price-The **opening price** is the **price** at which a security first trades upon the **opening** of an exchange on a trading day; for example, the New York **Stock** Exchange (NYSE) opens at precisely 9:30 a.m. Eastern time. The **price** of the first trade for any listed**stock** is its daily **opening price**.

```
In [25]: Stock_AAPL["open"].describe()

Out[25]: count    1259.000000
         mean      109.055429
         std        30.549220
         min        55.424200
         25%        84.647800
         50%       108.970000
         75%       127.335000
         max       179.370000
         Name: open, dtype: float64
```

4.Day High price of a stock- Day high price of a stock is the highest price Which the stock touches on a particular day.

```
In [33]: Stock_AAPL["high"].describe()

Out[33]: count    1259.000000
         mean      109.951118
         std        30.686186
         min        57.085700
         25%        85.334950
         50%       110.030000
         75%       128.100000
         max       180.100000
         Name: high, dtype: float64
```

5.Day Low price of a stock- Day low price of a stock is the lowest price of a stock which it touches on a particular day.

```
In [34]: Stock_AAPL["low"].describe()

Out[34]: count    1259.000000
         mean      108.141589
         std        30.376224
         min        55.014200
         25%        84.250650
         50%       108.050000
         75%       126.290000
         max       178.250000
         Name: low, dtype: float64
```

.

6.Volume- **Volume** is the number of **shares** or contracts traded in a security or an entire market during a given period of time. For every buyer, there is a seller, and each transaction contributes to the count of total **volume.**

```
In [36]: Stock_AAPL["volume"].describe()

Out[36]: count    1.259000e+03
         mean     5.404790e+07
         std      3.346835e+07
         min      1.147592e+07
         25%      2.969438e+07
         50%      4.566893e+07
         75%      6.870872e+07
         max      2.668336e+08
         Name: volume, dtype: float64
```

DATA CLEANING & MUNGING

Data cleaning removes major errors and inconsistencies that are inevitable when multiple sources of data are getting pulled into one dataset.

We have to check firstly if there is any null value present in the above mentioned data sets.

```
In [11]: Stock_AAPL.isnull().sum() #pd.isnull(Stock_AAPL)

Out[11]: open      0
         high      0
         low       0
         close     0
         volume    0
         Name      0
         dtype: int64
```

After performing the above operations, we have not encountered any null value in the given data set

Document sign date :19-Sep-2019

Analyzing the Data Based on Various Parameters

We have prepared some questions & based on them we will perform our analysis.

QUESTION 1:

WHAT WAS THE CHANGE IN PRICE OF STOCK OVER TIME ?

Stock prices change everyday by market forces. By this we mean that share prices change because of supply and demand. If more people want to buy a stock (demand) than sell it (supply), then the price moves up. Conversely, if more people wanted to sell a stock than buy it, there would be greater supply than demand, and the price would fall.

Understanding supply and demand is easy. What is difficult to comprehend is what makes people like a particular stock and dislike another stock. This comes down to figuring out what news is positive for a company and what news is negative. There are many answers to this problem and just about any investor you ask has their own ideas and strategies.

1>. CHANGE IN PRICE OF STOCK

A>. AAPL STOCK

In [26]:
```
price_change_AAPL=(Stock_AAPL["close"]-Stock_AAPL["open"])/(Stock_AAPL["open"])
```

In [27]:
```
Stock_AAPL["price_change"]=price_change_AAPL
```

In [28]:
```
Stock_AAPL
```

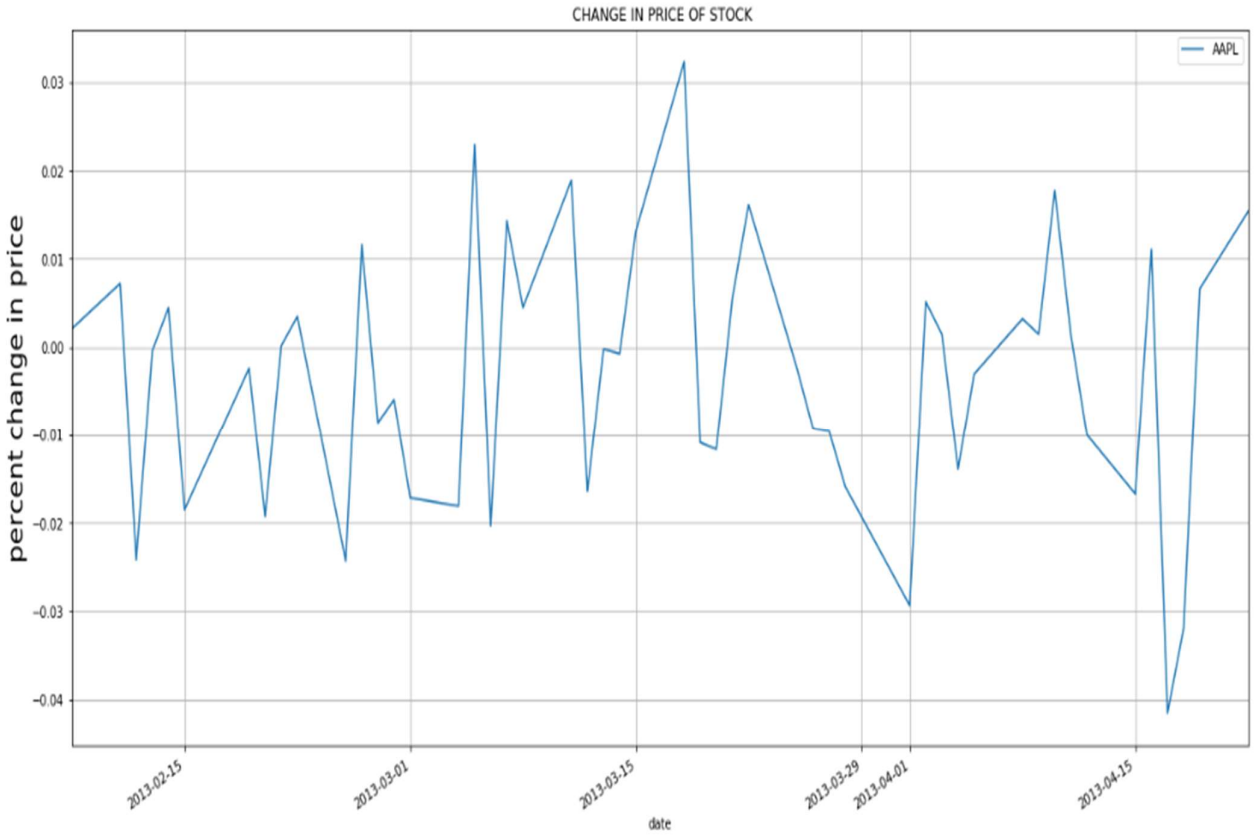
Out[28]:

date	open	high	low	close	volume	Name	price_change
2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL	0.002068
2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL	0.007198
2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL	-0.024213
2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL	-0.000429
2013-02-14	66.3599	67.3771	66.2885	66.6556	88809154	AAPL	0.004456
2013-02-15	66.9785	67.1656	65.7028	65.7371	97924631	AAPL	-0.018534
2013-02-19	65.8714	66.1042	64.8356	65.7128	108854046	AAPL	-0.002408
2013-02-20	65.3842	65.3842	64.1142	64.1214	118891367	AAPL	-0.019314
2013-02-21	63.7142	64.1671	63.2599	63.7228	111596821	AAPL	0.000135
2013-02-22	64.1785	64.5142	63.7999	64.4014	82583823	AAPL	0.003473
2013-02-25	64.8356	65.0171	63.2242	63.2571	92899597	AAPL	-0.024346
2013-02-26	63.4028	64.5056	62.5228	64.1385	125096657	AAPL	0.011604
2013-02-27	64.0614	64.6342	62.9499	63.5099	146674682	AAPL	-0.008609
2013-02-28	63.4357	63.9814	63.0571	63.0571	80532382	AAPL	-0.005968

AAPL STOCK PRICE CHANGE VISUALIZATION

In [55]:
```
Stock_AAPL["price_change"][:50].plot(label='AAPL', figsize=(20,10), title=" CHANGE IN PRICE OF STOCK")
ax = plt.gca()
ax.set_ylabel('percent change in price',fontsize=20)
plt.grid()
plt.legend()
```

Out[55]: <matplotlib.legend.Legend at 0x26967b03ef0>



Document sign date :19-Sep-2019

B> AMZN STOCK

In [31]:
```
price_change_AMZN=(Stock_AMZN["close"]-Stock_AMZN["open"])/(Stock_AMZN["open"])
```

In [32]:
```
Stock_AMZN["price_change"]=price_change_AMZN
```

In [33]:
```
Stock_AMZN
```

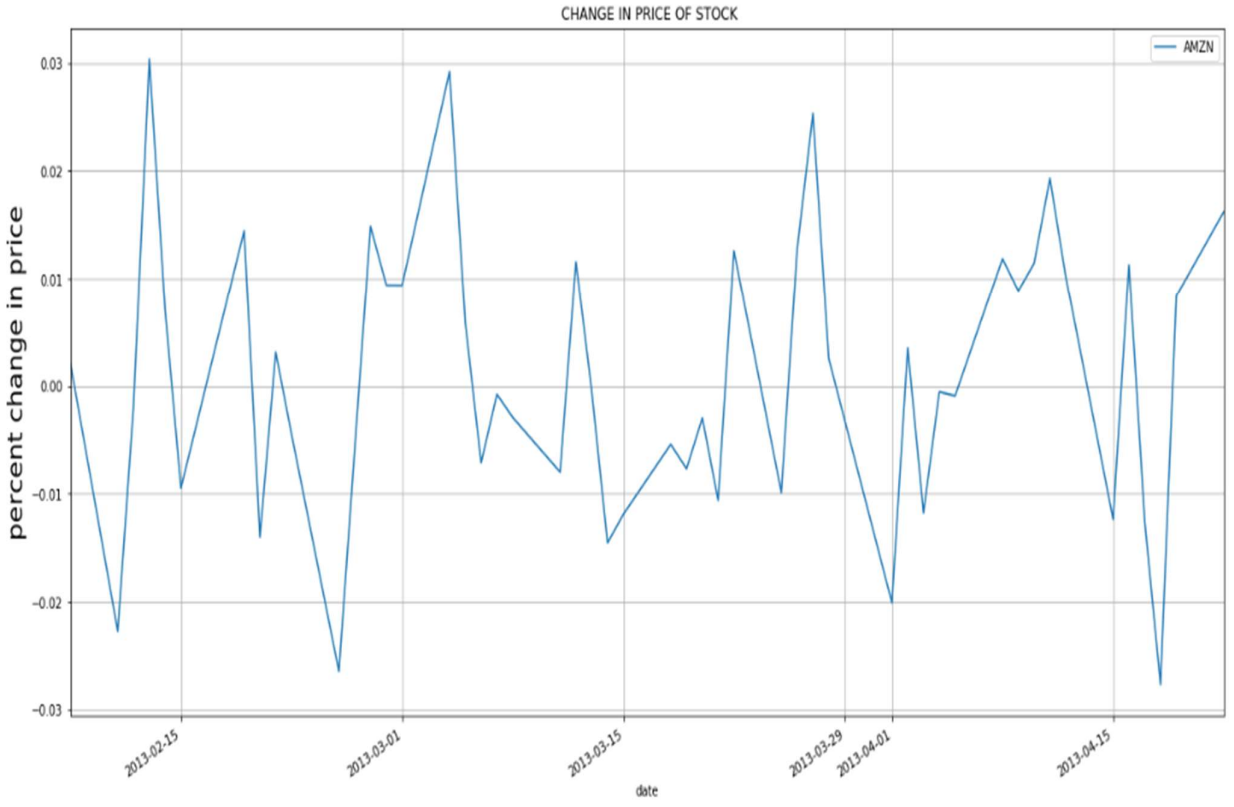
Out[33]:

date	open	high	low	close	volume	Name	price_change
2013-02-08	261.40	265.250	260.5550	261.950	3879078	AMZN	0.002104
2013-02-11	263.20	263.250	256.6000	257.210	3403403	AMZN	-0.022758
2013-02-12	259.19	260.160	257.0000	258.700	2938660	AMZN	-0.001891
2013-02-13	261.53	269.960	260.3000	269.470	5292996	AMZN	0.030360
2013-02-14	267.37	270.650	265.4000	269.240	3462780	AMZN	0.006994
2013-02-15	267.63	268.920	263.1100	265.090	3979832	AMZN	-0.009491
2013-02-19	265.91	270.110	264.5000	269.750	2853752	AMZN	0.014441
2013-02-20	270.20	274.300	266.3710	266.410	3528862	AMZN	-0.014027
2013-02-21	265.12	269.480	263.2500	265.940	3637396	AMZN	0.003093
2013-02-22	266.62	267.110	261.6100	265.420	3123402	AMZN	-0.004501
2013-02-25	266.94	268.694	259.6500	259.870	3032109	AMZN	-0.026485
2013-02-26	260.89	262.040	255.7300	259.360	3348011	AMZN	-0.005865
2013-02-27	259.40	265.830	256.8600	263.250	2908010	AMZN	0.014842
2013-02-28	261.81	267.000	260.6300	264.270	2667199	AMZN	0.009396
2013-03-01	263.27	266.600	261.0400	265.740	2956724	AMZN	0.009382
2013-03-04	265.36	273.300	264.1400	273.110	3452783	AMZN	0.029206

AMZN STOCK PRICE CHANGE VISUALIZATION

In [56]:
```
Stock_AMZN["price_change"][:50].plot(label='AMZN', figsize=(20,10), title=" CHANGE IN PRICE OF STOCK")
ax = plt.gca()
ax.set_ylabel('percent change in price',fontsize=20)
plt.grid()
plt.legend()
```

Out[56]: <matplotlib.legend.Legend at 0x26967b68c18>



Document sign date :19-Sep-2019

C>.FOR GOOGL STOCK

In [35]:
```
price_change_GOOGL=(Stock_GOOGL["close"]-Stock_GOOGL["open"])/(Stock_GOOGL["open"])
```

In [48]:
```
Stock_GOOGL["price_change"]=price_change_GOOGL
```

In [49]:
```
Stock_GOOGL
```

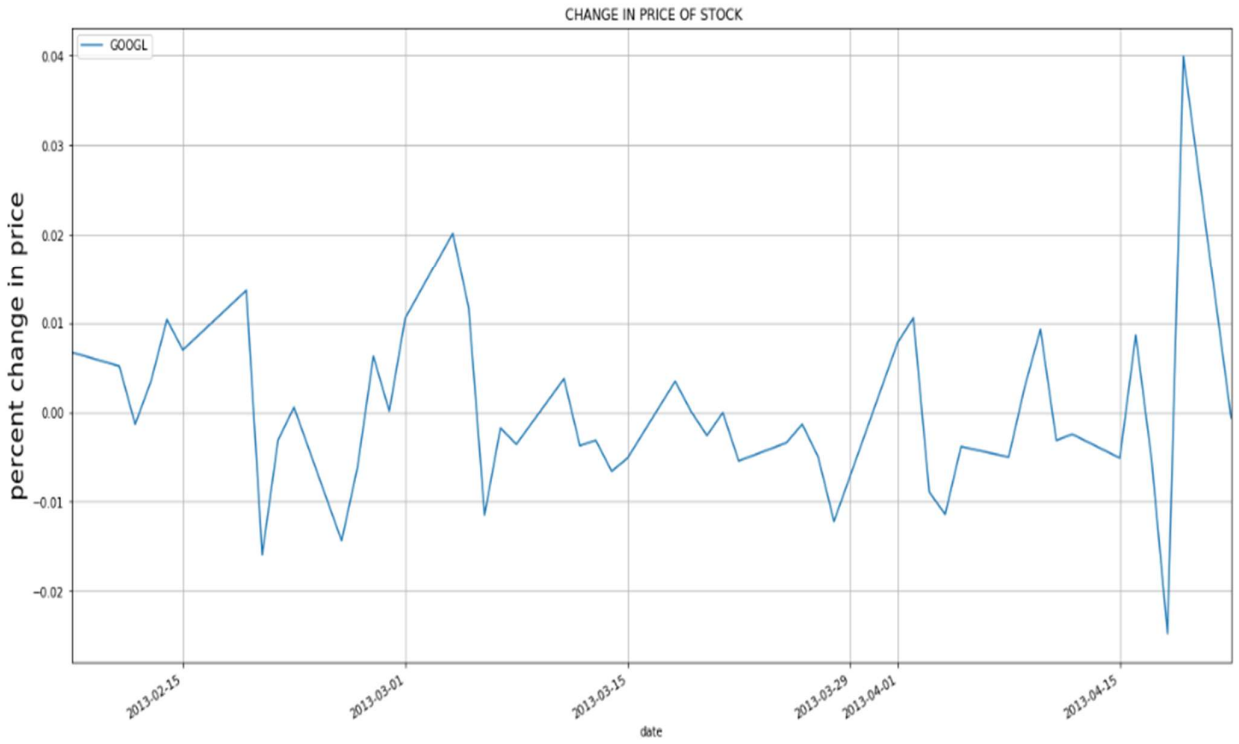
Out[49]:

date	open	high	low	close	volume	Name	price_change
2013-02-08	390.4551	393.7283	390.1698	393.0777	6031199	GOOGL	0.006717
2013-02-11	389.5892	391.8915	387.2619	391.6012	4330781	GOOGL	0.005164
2013-02-12	391.2659	394.3440	390.0747	390.7403	3714176	GOOGL	-0.001343
2013-02-13	390.4551	393.0677	390.3750	391.8214	2393946	GOOGL	0.003499
2013-02-14	390.2549	394.7644	389.2739	394.3039	3466971	GOOGL	0.010375
2013-02-15	394.0937	397.0266	393.9285	396.8414	5453980	GOOGL	0.006972
2013-02-19	398.3930	403.9035	398.0376	403.8284	5857528	GOOGL	0.013643
2013-02-20	403.0527	404.8895	396.2929	396.6262	5522500	GOOGL	-0.015945
2013-02-21	399.3990	403.1277	396.0056	398.1628	7008464	GOOGL	-0.003095
2013-02-22	400.0296	401.0256	397.2969	400.2549	4103315	GOOGL	0.000563
2013-02-25	401.5512	404.6092	395.6402	395.7804	4602925	GOOGL	-0.014371
2013-02-26	397.8975	398.3729	392.5922	395.4601	4399988	GOOGL	-0.006126
2013-02-27	397.7974	402.7774	395.9506	400.2899	4047784	GOOGL	0.006266
2013-02-28	400.9506	403.8985	400.9155	401.0006	4526218	GOOGL	0.000125
2013-03-01	399.2989	403.9736	398.4731	403.4981	4346304	GOOGL	0.010516
2013-03-04	403.0527	411.8314	402.9025	411.1628	5545624	GOOGL	0.020122
2013-03-05	414.8795	420.4950	414.8644	419.7213	8079065	GOOGL	0.011670

GOOGL STOCK PRICE CHANGE VISUALIZATION

In [57]:
```
Stock_GOOGL["price_change"][:50].plot(label='GOOGL', figsize=(20,10), title=" CHANGE IN PRICE OF STOCK")
ax = plt.gca()
ax.set_ylabel('percent change in price',fontsize=20)
plt.grid()
plt.legend()
```

Out[57]: <matplotlib.legend.Legend at 0x269683cfe10>



Document sign date :19-Sep-2019

D>. FOR MSFT STOCK

In [41]:
```
price_change_MSFT=(Stock_MSFT["close"]-Stock_MSFT["open"])/(Stock_MSFT["open"])
```

In [45]:
```
Stock_MSFT["price_change"]=price_change_MSFT
```

In [46]:
```
Stock_GOOGL
```

Out[46]:

date	open	high	low	close	volume	Name	price_change
2013-02-08	390.4551	393.7283	390.1698	393.0777	6031199	GOOGL	0.006717
2013-02-11	389.5892	391.8915	387.2619	391.6012	4330781	GOOGL	0.005164
2013-02-12	391.2659	394.3440	390.0747	390.7403	3714176	GOOGL	-0.001343
2013-02-13	390.4551	393.0677	390.3750	391.8214	2393946	GOOGL	0.003499
2013-02-14	390.2549	394.7644	389.2739	394.3039	3466971	GOOGL	0.010375
2013-02-15	394.0937	397.0266	393.9285	396.8414	5453980	GOOGL	0.006972
2013-02-19	398.3930	403.9035	398.0376	403.8284	5857528	GOOGL	0.013643
2013-02-20	403.0527	404.8895	396.2929	396.6262	5522500	GOOGL	-0.015945
2013-02-21	399.3990	403.1277	396.0056	398.1628	7008464	GOOGL	-0.003095
2013-02-22	400.0296	401.0256	397.2969	400.2549	4103315	GOOGL	0.000563
2013-02-25	401.5512	404.6092	395.6402	395.7804	4602925	GOOGL	-0.014371
2013-02-26	397.8975	398.3729	392.5922	395.4601	4399988	GOOGL	-0.006126
2013-02-27	397.7974	402.7774	395.9506	400.2899	4047784	GOOGL	0.006266
2013-02-28	400.9506	403.8985	400.9155	401.0006	4526218	GOOGL	0.000125
2013-03-01	399.2989	403.9736	398.4731	403.4981	4346304	GOOGL	0.010516

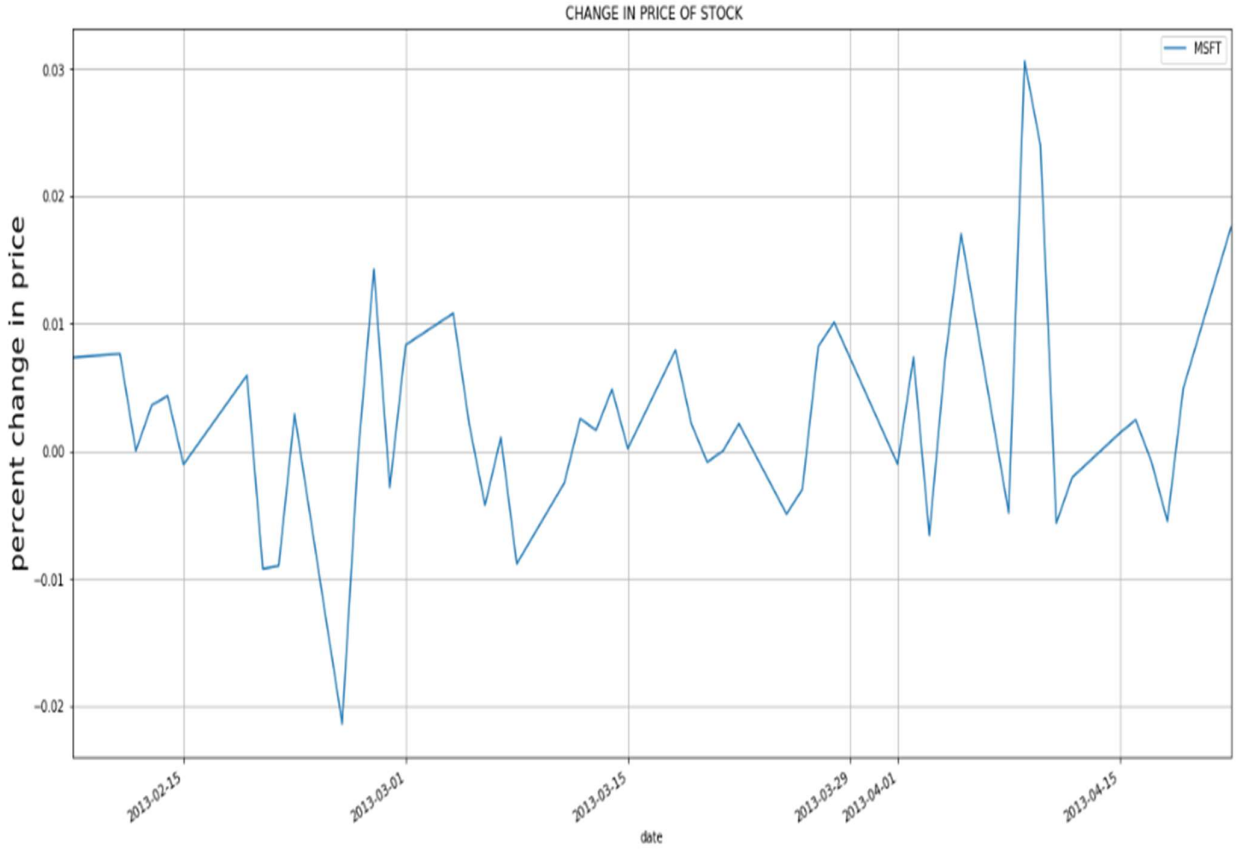
Globsyn Knowledge Academy

Document sign date :19-Sep-2019

MSFT PRICE CHANGE VISUALIZATION

In [58]:
```
Stock_MSFT["price_change"][:50].plot(label='MSFT', figsize=(20,10), title=" CHANGE IN PRICE OF STOCK")
ax = plt.gca()
ax.set_ylabel('percent change in price',fontsize=20)
plt.grid()
plt.legend()
```

Out[58]: <matplotlib.legend.Legend at 0x269686f5588>

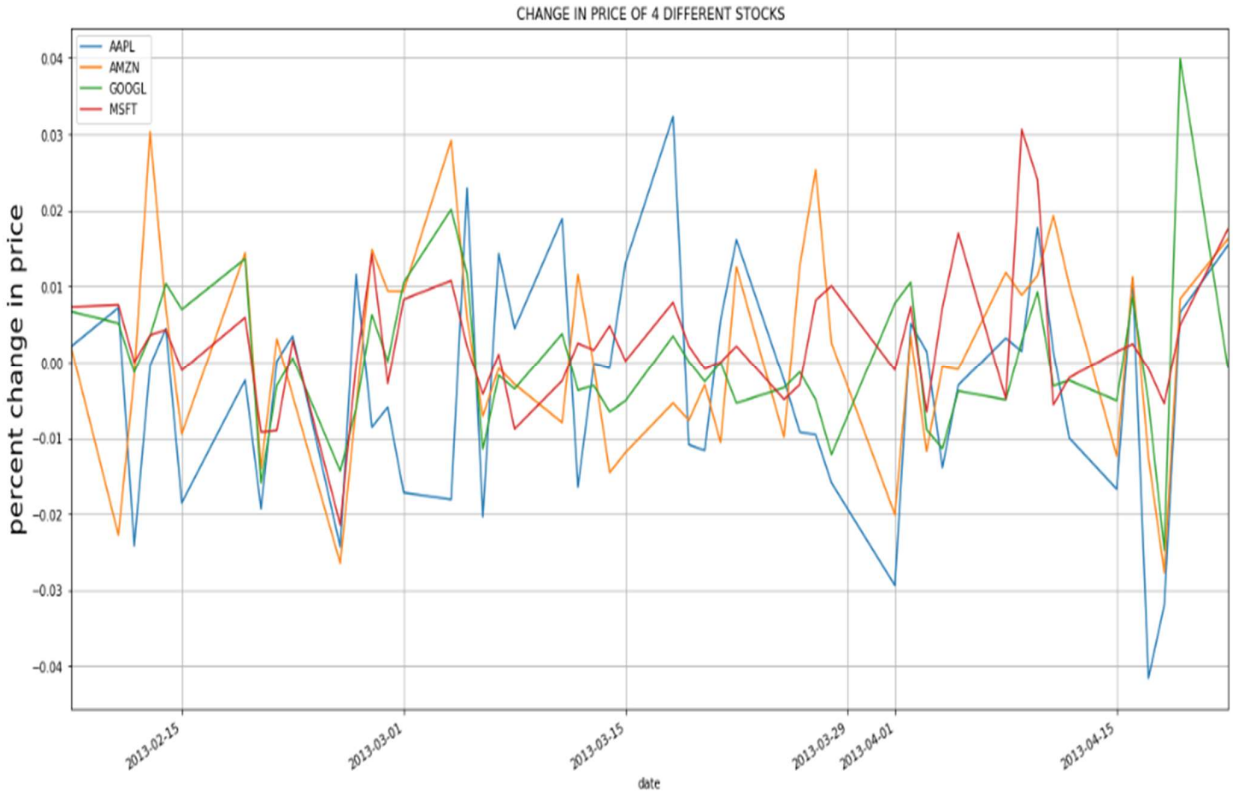


Document sign date :19-Sep-2019

ALL FOUR STOCKS PRICE CHANGE VISUALIZATION

```
In [60]: Stock_AAPL["price_change"][:50].plot(label='AAPL', figsize=(20,10), title=" CHANGE IN PRICE OF 4 DIFFERENT STOCKS")
         Stock_AMZN["price_change"][:50].plot(label='AMZN')
         Stock_GOOGL["price_change"][:50].plot(label='GOOGL')
         Stock_MSFT["price_change"][:50].plot(label='MSFT')
         ax = plt.gca()
         ax.set_ylabel('percent change in price',fontsize=20)
         plt.grid()
         plt.legend()
```

```
Out[60]: <matplotlib.legend.Legend at 0x26968d71f60>
```



Document sign date :19-Sep-2019

QUESTION 2:

WHAT WAS THE DAILY RETURN OF THE STOCK ON AVERAGE?

To calculate how much you gained or lost per day for a stock, subtract the opening price from the closing price. Then, multiply the result by the number of shares you own in the company. For example, say you own 100 shares of a stock that opened the day at $20 and ended the day at $21

A> FOR AAPL STOCK

In [37]:
```
daily_return_AAPL=(Stock_AAPL["close"]-Stock_AAPL["open"])*(Stock_AAPL["volume"])
```

In [38]:
```
Stock_AAPL["daily_return"]=daily_return_AAPL
```

In [39]:
```
Stock_AAPL
```

Out[39]:

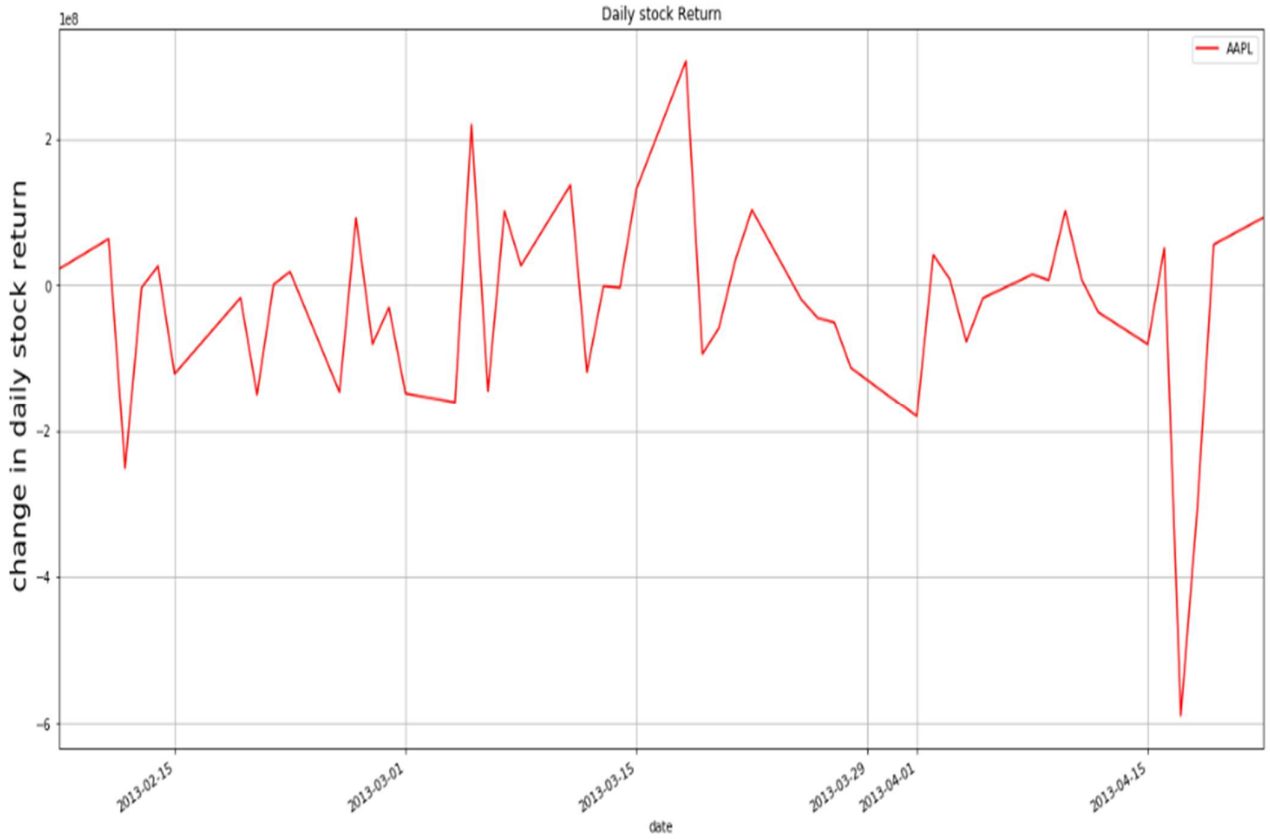
date	open	high	low	close	volume	Name	price_change	daily_return
2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL	0.002068	2.214358e+07
2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL	0.007198	6.322442e+07
2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL	-0.024213	-2.518242e+08
2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL	-0.000429	-3.395449e+06
2013-02-14	66.3599	67.3771	66.2885	66.6556	88809154	AAPL	0.004456	2.626087e+07
2013-02-15	66.9785	67.1656	65.7028	65.7371	97924631	AAPL	-0.018534	-1.215636e+08
2013-02-19	65.8714	66.1042	64.8356	65.7128	108854046	AAPL	-0.002408	-1.726425e+07
2013-02-20	65.3842	65.3842	64.1142	64.1214	118891367	AAPL	-0.019314	-1.501360e+08
2013-02-21	63.7142	64.1671	63.2599	63.7228	111596821	AAPL	0.000135	9.597327e+05
2013-02-22	64.1785	64.5142	63.7999	64.4014	82583823	AAPL	0.003473	1.840793e+07
2013-02-25	64.8356	65.0171	63.2242	63.2571	92899597	AAPL	-0.024346	-1.466420e+08
2013-02-26	63.4028	64.5056	62.5228	64.1385	125096657	AAPL	0.011604	9.203361e+07
2013-02-27	64.0614	64.6342	62.9499	63.5099	146674682	AAPL	-0.008609	-8.089109e+07
2013-02-28	63.4357	63.9814	63.0571	63.0571	80532382	AAPL	-0.005968	-3.048956e+07

Document sign date :19-Sep-2019

AAPL DAILY RETURN ON AVERAGE VISUALIZATION

```
In [40]: Stock_AAPL["daily_return"][:50].plot(label='AAPL', figsize=(20,10), title="Daily stock Return",color="red")
         ax = plt.gca()
         ax.set_ylabel(' change in daily stock return',fontsize=20)
         plt.grid()
         plt.legend()
```

```
Out[40]: <matplotlib.legend.Legend at 0x1a7de29f828>
```



Document sign date :19-Sep-2019

B> AMZN STOCK

In [41]:
```
daily_return_AMZN=(Stock_AMZN["close"]-Stock_AMZN["open"])*(Stock_AMZN["volume"])
```

In [42]:
```
Stock_AMZN["daily_return"]=daily_return_AMZN
```

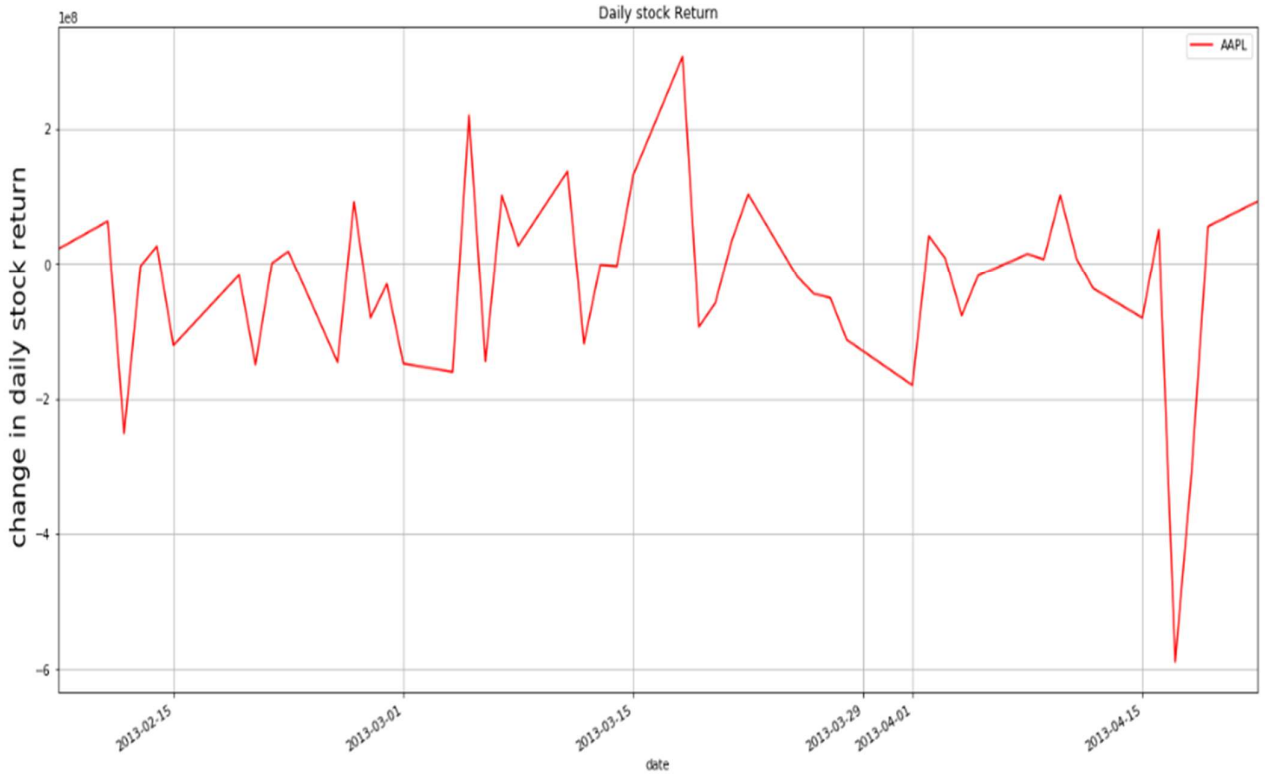
In [43]:
```
Stock_AMZN
```

Out[43]:

date	open	high	low	close	volume	Name	price_change	daily_return
2013-02-08	261.40	265.250	260.5550	261.950	3879078	AMZN	0.002104	2.133493e+06
2013-02-11	263.20	263.250	256.6000	257.210	3403403	AMZN	-0.022758	-2.038638e+07
2013-02-12	259.19	260.160	257.0000	258.700	2938660	AMZN	-0.001891	-1.439943e+06
2013-02-13	261.53	269.960	260.3000	269.470	5292996	AMZN	0.030360	4.202639e+07
2013-02-14	267.37	270.650	265.4000	269.240	3462780	AMZN	0.006994	6.475399e+06
2013-02-15	267.63	268.920	263.1100	265.090	3979832	AMZN	-0.009491	-1.010877e+07
2013-02-19	265.91	270.110	264.5000	269.750	2853752	AMZN	0.014441	1.095841e+07
2013-02-20	270.20	274.300	266.3710	266.410	3528862	AMZN	-0.014027	-1.337439e+07
2013-02-21	265.12	269.480	263.2500	265.940	3637396	AMZN	0.003093	2.982665e+06
2013-02-22	266.62	267.110	261.6100	265.420	3123402	AMZN	-0.004501	-3.748082e+06
2013-02-25	266.94	268.694	259.6500	259.870	3032109	AMZN	-0.026485	-2.143701e+07
2013-02-26	260.89	262.040	255.7300	259.360	3348011	AMZN	-0.005865	-5.122457e+06
2013-02-27	259.40	265.830	256.8600	263.250	2908010	AMZN	0.014842	1.119584e+07
2013-02-28	261.81	267.000	260.6300	264.270	2667199	AMZN	0.009396	6.561310e+06
2013-03-01	263.27	266.600	261.0400	265.740	2956724	AMZN	0.009382	7.303108e+06
2013-03-04	265.36	273.300	264.1400	273.110	3452783	AMZN	0.029206	2.675907e+07

AMZN DAILY RETURN ON AVERAGE VISUALIZATION

```
In [40]: Stock_AAPL["daily_return"][:50].plot(label='AAPL', figsize=(20,10), title="Daily stock Return",color="red")
         ax = plt.gca()
         ax.set_ylabel(' change in daily stock return',fontsize=20)
         plt.grid()
         plt.legend()

Out[40]: <matplotlib.legend.Legend at 0x1a7de29f828>
```



Document sign date :19-Sep-2019

C>GOOGL STOCK

In [45]:
```
daily_return_GOOGL=(Stock_GOOGL["close"]-Stock_GOOGL["open"])*(Stock_GOOGL["volume"])
```

In [46]:
```
Stock_GOOGL["daily_return"]=daily_return_GOOGL
```

In [47]:
```
Stock_GOOGL
```

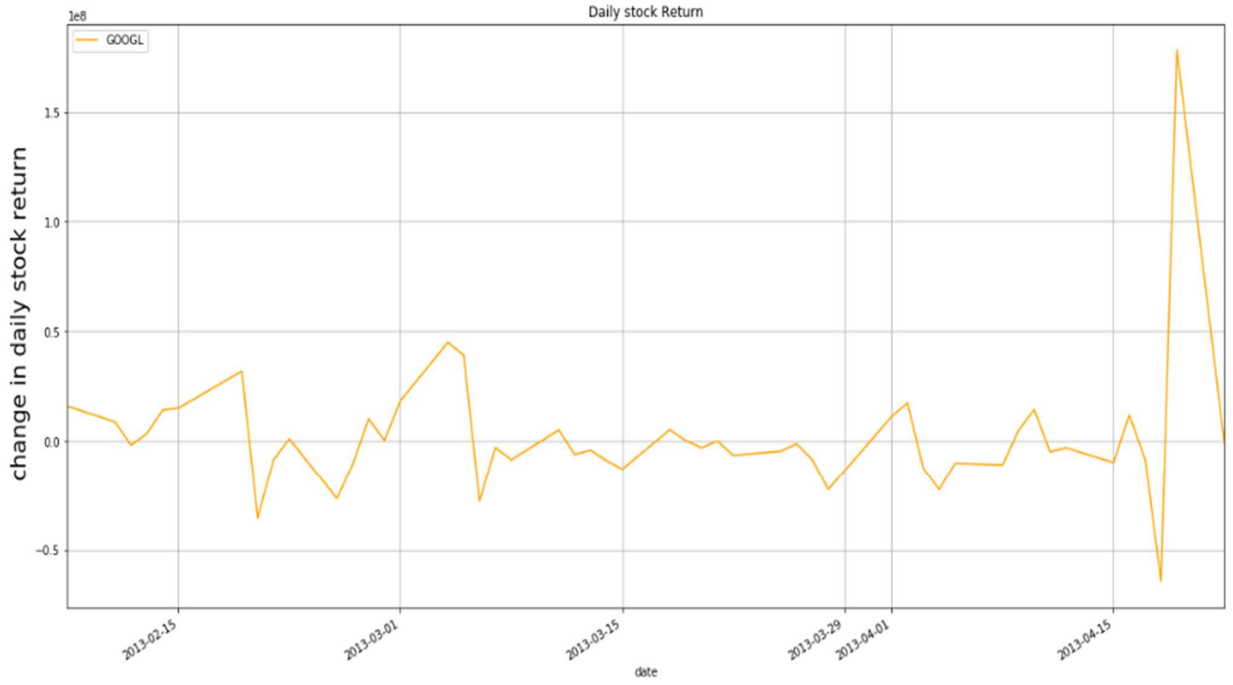
Out[47]:

date	open	high	low	close	volume	Name	price_change	daily_return
2013-02-08	390.4551	393.7283	390.1698	393.0777	6031199	GOOGL	0.006717	1.581742e+07
2013-02-11	389.5892	391.8915	387.2619	391.6012	4330781	GOOGL	0.005164	8.713531e+06
2013-02-12	391.2659	394.3440	390.0747	390.7403	3714176	GOOGL	-0.001343	-1.952171e+06
2013-02-13	390.4551	393.0677	390.3750	391.8214	2393946	GOOGL	0.003499	3.270848e+06
2013-02-14	390.2549	394.7644	389.2739	394.3039	3466971	GOOGL	0.010375	1.403777e+07
2013-02-15	394.0937	397.0266	393.9285	396.8414	5453980	GOOGL	0.006972	1.498590e+07
2013-02-19	398.3930	403.9035	398.0376	403.8284	5857528	GOOGL	0.013643	3.183801e+07
2013-02-20	403.0527	404.8895	396.2929	396.6262	5522500	GOOGL	-0.015945	-3.549035e+07
2013-02-21	399.3990	403.1277	396.0056	398.1628	7008464	GOOGL	-0.003095	-8.663863e+06
2013-02-22	400.0296	401.0256	397.2969	400.2549	4103315	GOOGL	0.000563	9.244769e+05
2013-02-25	401.5512	404.6092	395.6402	395.7804	4602925	GOOGL	-0.014371	-2.656256e+07
2013-02-26	397.8975	398.3729	392.5922	395.4601	4399988	GOOGL	-0.006126	-1.072453e+07
2013-02-27	397.7974	402.7774	395.9506	400.2899	4047784	GOOGL	0.006266	1.008910e+07
2013-02-28	400.9506	403.8985	400.9155	401.0006	4526218	GOOGL	0.000125	2.263109e+05
2013-03-01	399.2989	403.9736	398.4731	403.4981	4346304	GOOGL	0.010516	1.825100e+07
2013-03-04	403.0527	411.8314	402.9025	411.1628	5545624	GOOGL	0.020122	4.497557e+07
2013-03-05	414.9705	420.4950	414.9644	419.7212	9070965	GOOGL	0.011670	2.011722e+07

Document sign date :19-Sep-2019

GOOGL DAILY REURN ON AVERAGE VISUALIZATION

```
In [48]: Stock_GOOGL["daily_return"][:50].plot(label='GOOGL', figsize=(20,10), title="Daily stock Return", color="orange")
         ax = plt.gca()
         ax.set_ylabel(' change in daily stock return',fontsize=20)
         plt.grid()
         plt.legend()

Out[48]: <matplotlib.legend.Legend at 0x1a7de36bdd8>
```



Document sign date :19-Sep-2019

D> MSFT STOCK

In [49]:
```
daily_return_MSFT=(Stock_MSFT["close"]-Stock_MSFT["open"])*(Stock_MSFT["volume"])
```

In [50]:
```
Stock_MSFT["daily_return"]=daily_return_MSFT
```

In [51]:
```
Stock_MSFT
```

Out[51]:

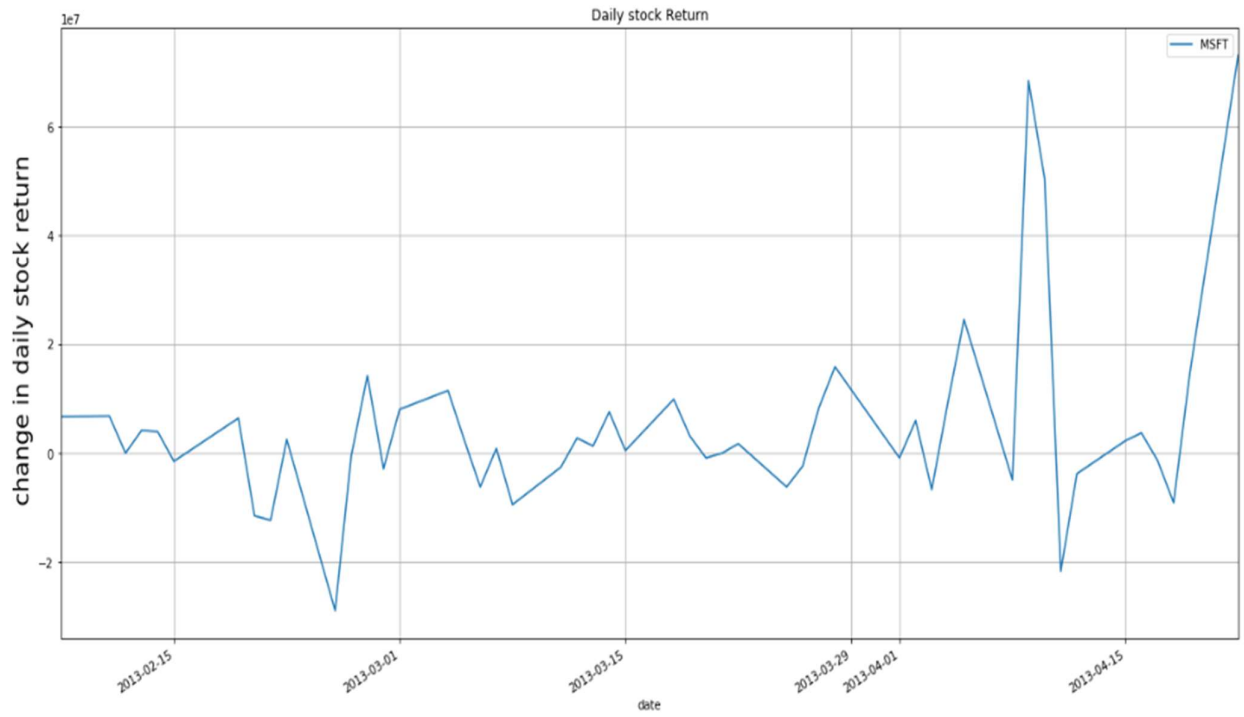
date	open	high	low	close	volume	Name	price_change	daily_return
2013-02-08	27.3500	27.7100	27.3100	27.550	33318306	MSFT	0.007313	6.663661e+06
2013-02-11	27.6500	27.9200	27.5000	27.860	32247549	MSFT	0.007595	6.771985e+06
2013-02-12	27.8800	28.0000	27.7500	27.880	35990829	MSFT	0.000000	0.000000e+00
2013-02-13	27.9300	28.1100	27.8800	28.030	41715530	MSFT	0.003580	4.171553e+06
2013-02-14	27.9200	28.0600	27.8700	28.040	32663174	MSFT	0.004298	3.919581e+06
2013-02-15	28.0400	28.1600	27.8750	28.010	49650538	MSFT	-0.001070	-1.489516e+06
2013-02-19	27.8801	28.0900	27.8000	28.045	38804616	MSFT	0.005915	6.398881e+06
2013-02-20	28.1300	28.2000	27.8300	27.870	44109412	MSFT	-0.009243	-1.146845e+07
2013-02-21	27.7400	27.7400	27.2300	27.490	49078338	MSFT	-0.009012	-1.226958e+07
2013-02-22	27.6800	27.7600	27.4800	27.760	31425726	MSFT	0.002890	2.514058e+06
2013-02-25	27.9700	28.0500	27.3700	27.370	48011248	MSFT	-0.021452	-2.880675e+07
2013-02-26	27.3800	27.6000	27.3400	27.370	49917353	MSFT	-0.000365	-4.991735e+05
2013-02-27	27.4200	28.0000	27.3300	27.810	36390889	MSFT	0.014223	1.419245e+07
2013-02-28	27.8800	27.9700	27.7400	27.800	35836861	MSFT	-0.002869	-2.866949e+06
2013-03-01	27.7200	27.9800	27.5200	27.950	34849287	MSFT	0.008297	8.015336e+06
2013-03-04	27.8500	28.1500	27.7000	28.150	38163549	MSFT	0.010772	1.144906e+07

Document sign date :19-Sep-2019

MSFT DAILY REURN ON AVERAGE VISUALIZATION

```
In [52]: Stock_MSFT["daily_return"][:50].plot(label='MSFT', figsize=(20,10), title="Daily stock Return")
         ax = plt.gca()
         ax.set_ylabel(' change in daily stock return',fontsize=20)
         plt.grid()
         plt.legend()
```

```
Out[52]: <matplotlib.legend.Legend at 0x1a7de93eac8>
```

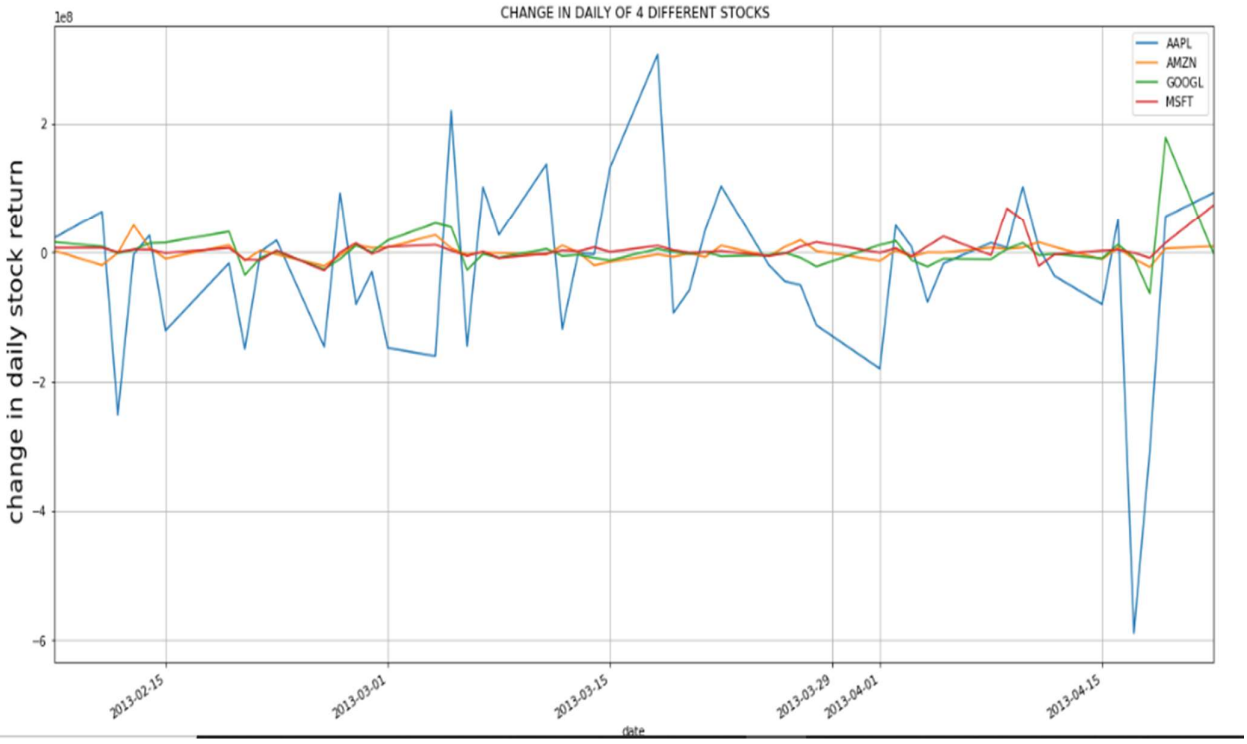


Document sign date :19-Sep-2019

FOUR STOCKS DAILY REURN ON AVERAGE VISUALIZATION

In [53]:
```
Stock_AAPL["daily_return"][:50].plot(label='AAPL', figsize=(20,10), title=" CHANGE IN DAILY OF 4 DIFFERENT STOCKS")
Stock_AMZN["daily_return"][:50].plot(label='AMZN')
Stock_GOOGL["daily_return"][:50].plot(label='GOOGL')
Stock_MSFT["daily_return"][:50].plot(label='MSFT')
ax = plt.gca()
ax.set_ylabel(' change in daily stock return',fontsize=20)
plt.grid()
plt.legend()
```

Out[53]: <matplotlib.legend.Legend at 0x1a7de98e518>



Document sign date :19-Sep-2019

QUESTION 3:

WHAT WAS THE MOVING AVERAGE OF ALL FOUR STOCKS?

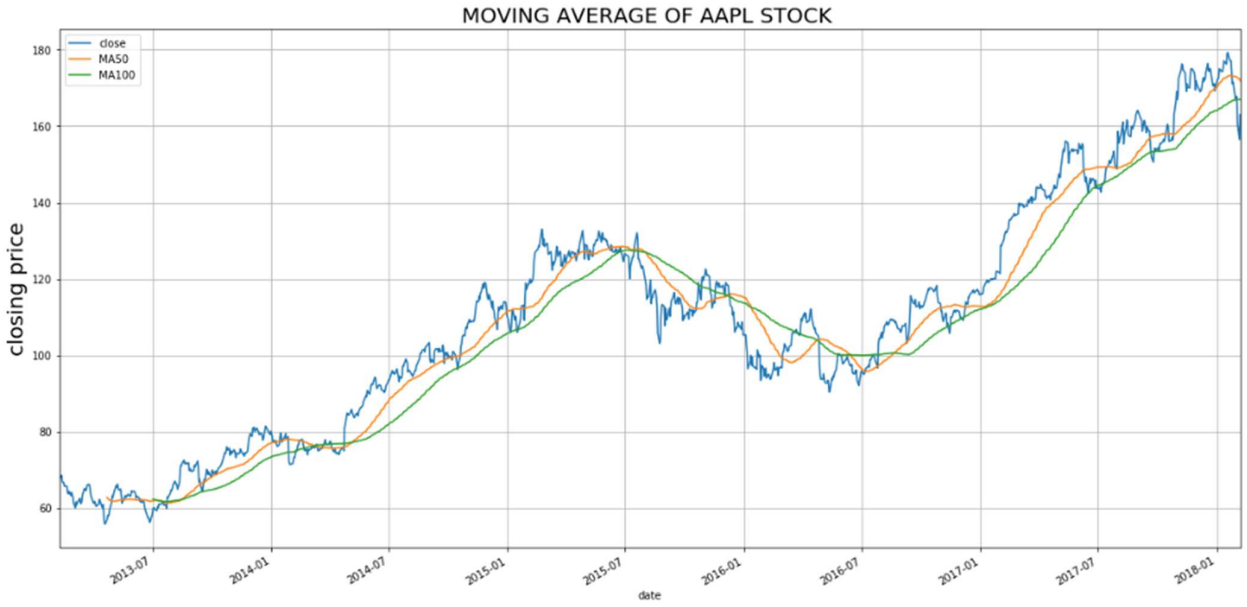
The **moving average** (MA) is a simple technical analysis tool that smooths out price data by creating a constantly updated **average** price. The **average** is taken over a specific period of time, like 10 days, 20 minutes, 30 weeks or any time period the trader chooses.

A> AAPL STOCK

In [217]:
```
Stock_AAPL["MA50"]=Stock_AAPL["close"].rolling(50).mean()
Stock_AAPL["MA100"]=Stock_AAPL["close"].rolling(100).mean()
Stock_AAPL[["close","MA50","MA100"]].plot(label="MSFT",figsize=(20,10))
plt.title("MOVING AVERAGE OF AAPL STOCK ",fontsize=20)
ax = plt.gca()
ax.set_ylabel('closing price',fontsize=20)
plt.grid()
plt.legend()
```

Out[217]: <matplotlib.legend.Legend at 0x1a515f88b70>

MOVING AVERAGE OF AAPL VISUALIZATION

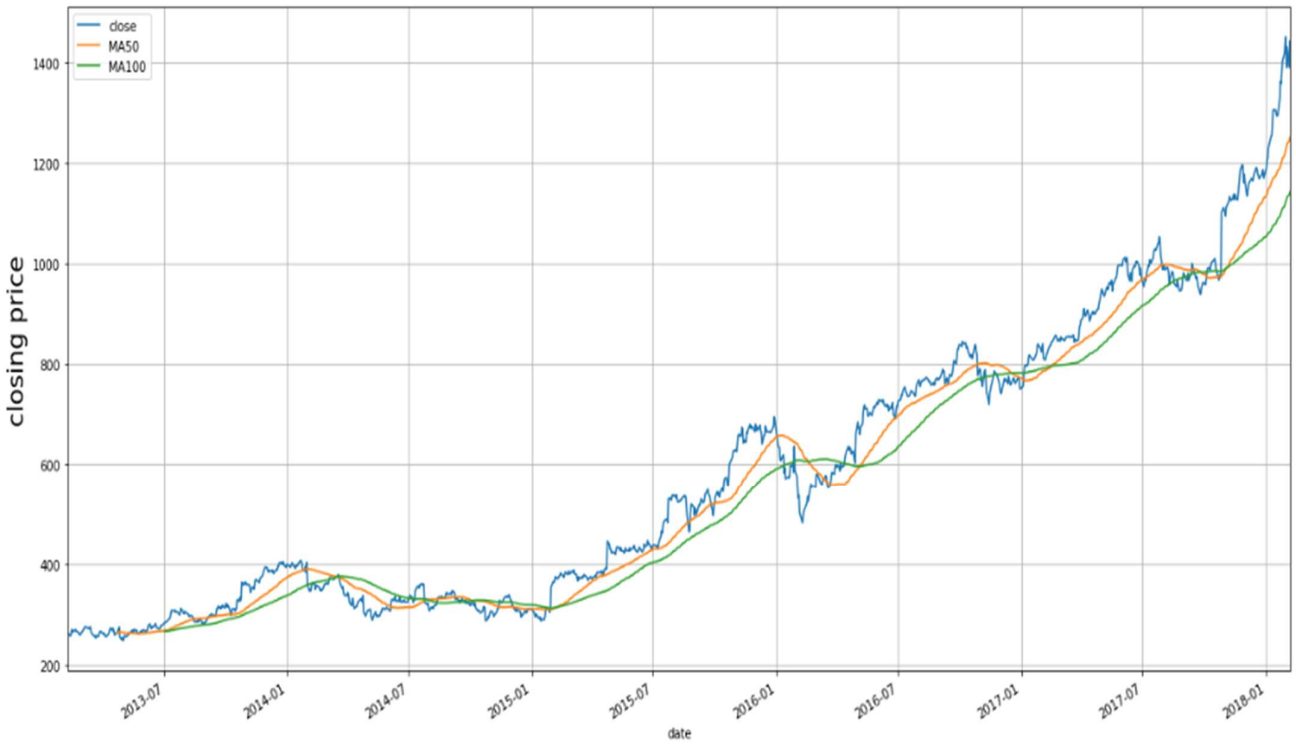


Document sign date :19-Sep-2019

B> AMZN STOCK

In [67]:
```
Stock_AMZN["MA50"]=Stock_AMZN["close"].rolling(50).mean()
Stock_AMZN["MA100"]=Stock_AMZN["close"].rolling(100).mean()
Stock_AMZN[["close","MA50","MA100"]].plot(label="MSFT",figsize=(20,10))
ax = plt.gca()
ax.set_ylabel('closing price',fontsize=20)
plt.grid()
plt.legend()
```

MOVING AVERAGE OF AMZN VISUALIZATION



Document sign date :19-Sep-2019

date

C> GOOGL STOCK

In [64]:
```
Stock_GOOGL["MA50"]=Stock_GOOGL["close"].rolling(50).mean()
Stock_GOOGL["MA100"]=Stock_GOOGL["close"].rolling(100).mean()
Stock_GOOGL[["close","MA50","MA100"]].plot(label="MSFT",figsize=(20,10))
ax = plt.gca()
ax.set_ylabel('closing price',fontsize=20)
plt.grid()
plt.legend()
```

MOVING AVERAGE OF GOOGL VISUALIZATION

Document sign date :19-Sep-2019

D>MSFT STOCK ¶

In [69]:
```
Stock_MSFT["MA50"]=Stock_MSFT["close"].rolling(50).mean()
Stock_MSFT["MA100"]=Stock_MSFT["close"].rolling(100).mean()
Stock_MSFT[["close","MA50","MA100"]].plot(label="MSFT",figsize=(20,10))
ax = plt.gca()
ax.set_ylabel('closing price',fontsize=20)
plt.grid()
plt.legend()
```

MOVING AVERAGE OF MSFT VISUALIZATION

Document sign date :19-Sep-2019

QUESTION 4:

WHAT WAS CORRELATION BETWEEN ALL FOUR STOCKS CLOSING PRICES?

Correlation, by itself, cannot affect the stock market because it is simply the degree to which two things behave in the same way. However, the correlation between the activity of two stocks, or between a stock and the performance of a given index, sector or industry, can be a very important factor in developing a prudent investing strategy. Stock analysts use a measure called the correlation coefficient to make predictions about how a stock will behave based on past performance and the activity of another security with which the stock in question has been shown to correlate.

4> CORRELATION BETWEEN DIFFERENT STOCKS CLOSING PRICE

In [71]:
```
close=pd.DataFrame({"AAPL":Stock_AAPL["close"],
                    "AMZN":Stock_AMZN["close"],
                   "GOOGL":Stock_GOOGL["close"],
                   "MSFT": Stock_MSFT["close"]})
```

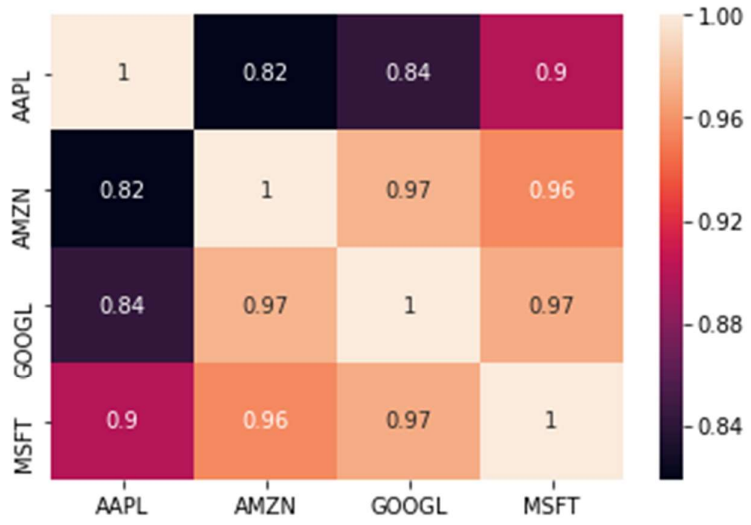
In [72]:
```
close
```

Out[72]:

date	AAPL	AMZN	GOOGL	MSFT
2013-02-08	67.8542	261.950	393.0777	27.550
2013-02-11	68.5614	257.210	391.6012	27.860
2013-02-12	66.8428	258.700	390.7403	27.880
2013-02-13	66.7156	269.470	391.8214	28.030
2013-02-14	66.6556	269.240	394.3039	28.040
2013-02-15	65.7371	265.090	396.8414	28.010
2013-02-19	65.7128	269.750	403.8284	28.045
2013-02-20	64.1214	266.410	396.6262	27.870
2013-02-21	63.7228	265.940	398.1628	27.490
2013-02-22	64.4014	265.420	400.2549	27.760
2013-02-25	63.2571	259.870	395.7804	27.370
2013-02-26	64.1385	259.360	395.4601	27.370
2013-02-27	63.5099	263.250	400.2899	27.810
2013-02-28	63.0571	264.270	401.0006	27.800
2013-03-01	61.4957	265.740	403.4981	27.950

CLOSING PRICE CORRELATION BETWEEN ALL FOUR STOCKS VISUALIZATION

```
In [73]: a=close.corr()

In [74]: sns.heatmap(a,annot=True)

Out[74]: <matplotlib.axes._subplots.AxesSubplot at 0x1b2bb977f98>
```



Document sign date :19-Sep-2019

QUESTION 5:

WHAT WAS CORRELATION BETWEEN ALL FOUR STOCKS DAILY RETURNS?

5>CORRELATION BETWEEN DIFFERENT STOCKS DAILY RETURN

In [75]:
```
daily_return=pd.DataFrame({"AAPL":Stock_AAPL["daily_return"],
                           "AMZN":Stock_AMZN["daily_return"],
                          "GOOGL":Stock_GOOGL["daily_return"],
                          "MSFT": Stock_MSFT["daily_return"]})
```

In [76]:
```
daily_return
```

Out[76]:

date	AAPL	AMZN	GOOGL	MSFT
2013-02-08	2.214358e+07	2.133493e+06	1.581742e+07	6.663661e+06
2013-02-11	6.322442e+07	-2.038638e+07	8.713531e+06	6.771985e+06
2013-02-12	-2.518242e+08	-1.439943e+06	-1.952171e+06	0.000000e+00
2013-02-13	-3.395449e+06	4.202639e+07	3.270848e+06	4.171553e+06
2013-02-14	2.626087e+07	6.475399e+06	1.403777e+07	3.919581e+06
2013-02-15	-1.215636e+08	-1.010877e+07	1.498590e+07	-1.489516e+06
2013-02-19	-1.726425e+07	1.095841e+07	3.183801e+07	6.398881e+06
2013-02-20	-1.501360e+08	-1.337439e+07	-3.549035e+07	-1.146845e+07
2013-02-21	9.597327e+05	2.982665e+06	-8.663863e+06	-1.226958e+07
2013-02-22	1.840793e+07	-3.748082e+06	9.244769e+05	2.514058e+06
2013-02-25	-1.466420e+08	-2.143701e+07	-2.656256e+07	-2.880675e+07
2013-02-26	9.203361e+07	-5.122457e+06	-1.072453e+07	-4.991735e+05
2013-02-27	-8.089109e+07	1.119584e+07	1.008910e+07	1.419245e+07
2013-02-28	-3.048956e+07	6.561310e+06	2.263109e+05	-2.866949e+06
2013-03-01	-1.483380e+08	7.303108e+06	1.825100e+07	8.015336e+06
2013-03-04	1.609794e+08	2.675997e+07	4.497557e+07	1.144996e+07

Document sign date :19-Sep-2019

DAILY RETURN CORRELATION BETWEEN ALL FOUR STOCKS VISUALIZATION

```
In [77]: b=daily_return.corr()

In [78]: sns.heatmap(b,annot=True)

Out[78]: <matplotlib.axes._subplots.AxesSubplot at 0x1b2bba34ef0>
```

Document sign date :19-Sep-2019

QUESTION 6:

HOW MUCH VALUE DO WE PUT AT RISK BY INVESTING IN A PARTICULAR STOCK?

Monte Carlo simulations are done to predict how the stock can perform in the future

A> AAPL STOCK

In [215]:
```
rets = Stock_AAPL.price_change
rets = rets[1:]
daily_vol = rets.std()
daily_ret = rets.mean()
simulation_df = pd.DataFrame()
num_simulations = 10000
predicted_days = 252
```

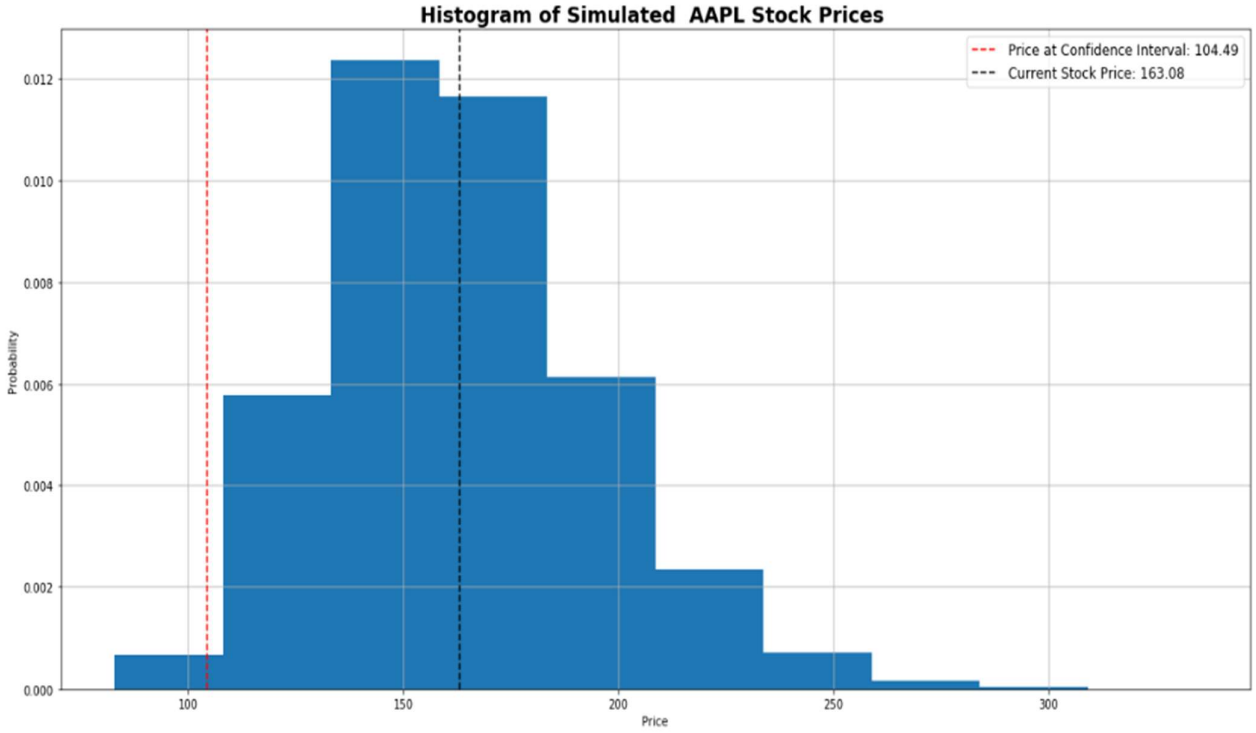
Document sign date :19-Sep-2019

In [233]:

```
last_price = Stock_AAPL.iloc[-1:,0]
last_price = last_price[0]
for x in range(num_simulations):
    count = 0
    price_series = []
    price_series.append(last_price)
    for i in range(predicted_days):
        if count == 251:
            break
        price = price_series[count] * (1 + np.random.normal(0, daily_vol))
        price_series.append(price)
        count += 1
    simulation_df[x] = price_series
price_array = simulation_df.iloc[-1, :]
price_array = sorted(price_array, key=int)
var =  np.percentile(price_array, .95)
var1 =  np.percentile(price_array, .99)
var2 =  np.percentile(price_array, .9999)
print ("VaR at 95% Confidence: " + '${:,.2f}'.format(last_price - var))
print ("VaR at 99% Confidence: " + '${:,.2f}'.format(last_price - var1))
print ("VaR at 99.99% Confidence:" + '${:,.2f}'.format(last_price - var2))
plt.figure(figsize=(20,10))
plt.hist(price_array,normed=True)
plt.xlabel('Price')
plt.ylabel('Probability')
plt.title(r'Histogram of Simulated Stock Prices', fontsize=18, fontweight='bold')
plt.axvline(x=var, color='r', linestyle='--', label='Price at Confidence Interval: '  + str(round(var, 2)))
plt.axvline(x=last_price, color='k', linestyle='--', label = 'Current Stock Price: ' + str(round(last_price, 2)))
plt.legend(loc='upper right', fontsize = 'large')
plt.grid()
plt.show()
```

Document sign date :19-Sep-2019

```
VaR at 95% Confidence: $58.60
VaR at 99% Confidence: $58.17
VaR at 99.99% Confidence:$58.16
```



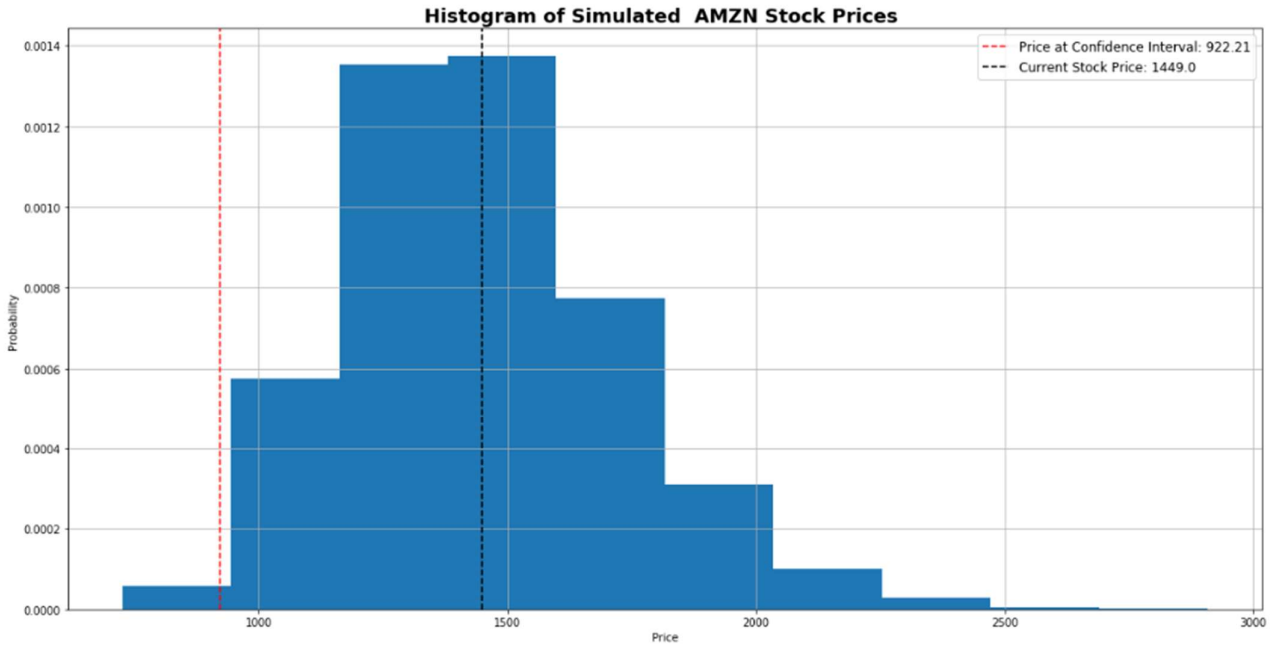
Document sign date :19-Sep-2019

B>AMZN STOCK

In [229]:
```
rets = Stock_AMZN.price_change
rets = rets[1:]
daily_vol = rets.std()
daily_ret = rets.mean()
simulation_df = pd.DataFrame()
num_simulations = 10000
predicted_days = 252
```

In [235]:
```
last_price = Stock_AMZN.iloc[-1:,0]
last_price = last_price[0]
for x in range(num_simulations):
    count = 0
    price_series = []
    price_series.append(last_price)
    for i in range(predicted_days):
        if count == 251:
            break
        price = price_series[count] * (1 + np.random.normal(0, daily_vol))
        price_series.append(price)
        count += 1
    simulation_df[x] = price_series
price_array = simulation_df.iloc[-1, :]
price_array = sorted(price_array, key=int)
var =  np.percentile(price_array, .95)
var1 =  np.percentile(price_array, .99)
var2 =  np.percentile(price_array, .9999)
print ("VaR at 95% Confidence: " + '${:,.2f}'.format(last_price - var))
print ("VaR at 99% Confidence: " + '${:,.2f}'.format(last_price - var1))
print ("VaR at 99.99% Confidence:" + '${:,.2f}'.format(last_price - var2))
plt.figure(figsize=(20,10))
plt.hist(price_array,normed=True)
plt.xlabel('Price')
plt.ylabel('Probability')
plt.title(r'Histogram of Simulated  AMZN Stock Prices ', fontsize=18, fontweight='bold')
plt.axvline(x=var, color='r', linestyle='--', label='Price at Confidence Interval: '  + str(round(var, 2)))
plt.axvline(x=last_price, color='k', linestyle='--', label = 'Current Stock Price: ' + str(round(last_price, 2)))
plt.legend(loc='upper right', fontsize = 'large')
plt.grid()
plt.show()
```

```
VaR at 95% Confidence: $526.79
VaR at 99% Confidence: $525.43
VaR at 99.99% Confidence:$525.21
```



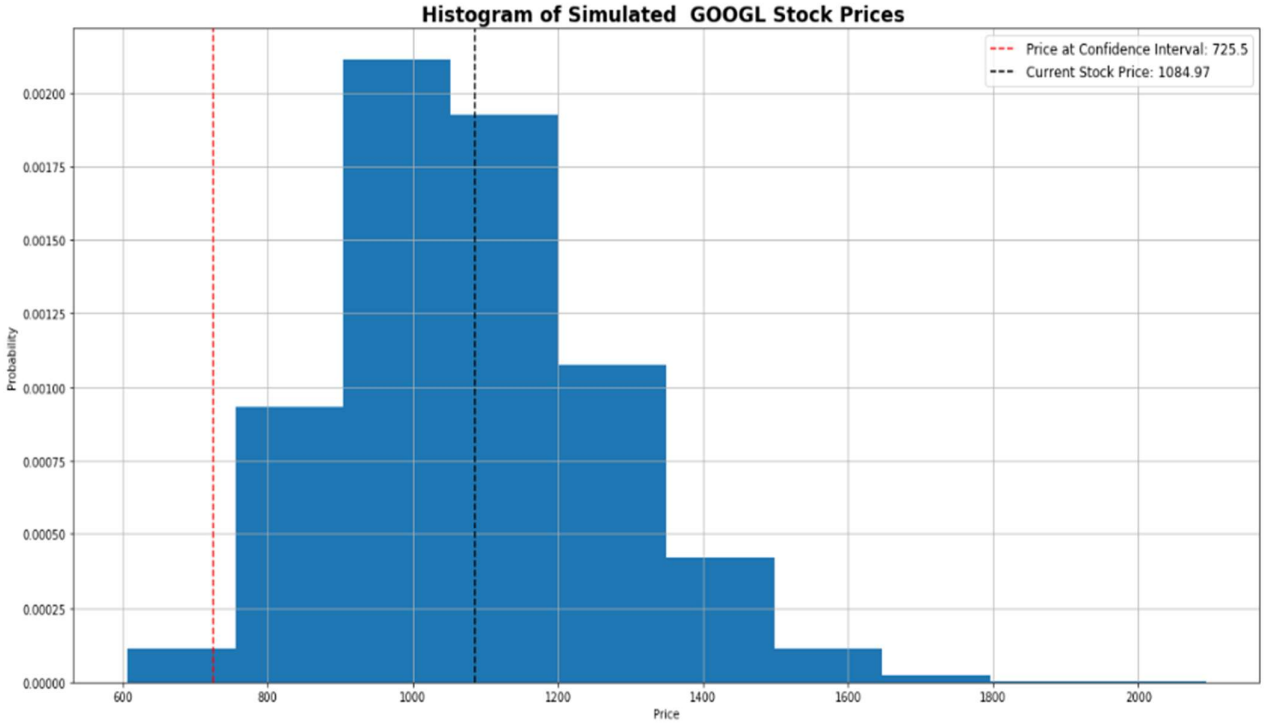
Document sign date :19-Sep-2019

C> GOOGL STOCK

In [233]:
```
rets = Stock_GOOGL.price_change
rets = rets[1:]
daily_vol = rets.std()
daily_ret = rets.mean()
simulation_df = pd.DataFrame()
num_simulations = 10000
predicted_days = 252
```

In [234]:
```
last_price = Stock_GOOGL.iloc[-1:,0]
last_price = last_price[0]
for x in range(num_simulations):
    count = 0
    price_series = []
    price_series.append(last_price)
    for i in range(predicted_days):
        if count == 251:
            break
        price = price_series[count] * (1 + np.random.normal(0, daily_vol))
        price_series.append(price)
        count += 1
    simulation_df[x] = price_series
price_array = simulation_df.iloc[-1, :]
price_array = sorted(price_array, key=int)
var =  np.percentile(price_array, .95)
var1 =  np.percentile(price_array, .99)
var2 =  np.percentile(price_array, .9999)
print ("VaR at 95% Confidence: " + '${:,.2f}'.format(last_price - var))
print ("VaR at 99% Confidence: " + '${:,.2f}'.format(last_price - var1))
print ("VaR at 99.99% Confidence:" + '${:,.2f}'.format(last_price - var2))
plt.figure(figsize=(20,10))
plt.hist(price_array,normed=True)
plt.xlabel('Price')
plt.ylabel('Probability')
plt.title(r'Histogram of Simulated  GOOGL Stock Prices ', fontsize=18, fontweight='bold')
plt.axvline(x=var, color='r', linestyle='--', label='Price at Confidence Interval: '  + str(round(var, 2)))
plt.axvline(x=last_price, color='k', linestyle='--', label = 'Current Stock Price: ' + str(round(last_price, 2)))
plt.legend(loc='upper right', fontsize = 'large')
plt.grid()
plt.show()
```

```
VaR at 95% Confidence: $359.47
VaR at 99% Confidence: $357.82
VaR at 99.99% Confidence:$357.68
```



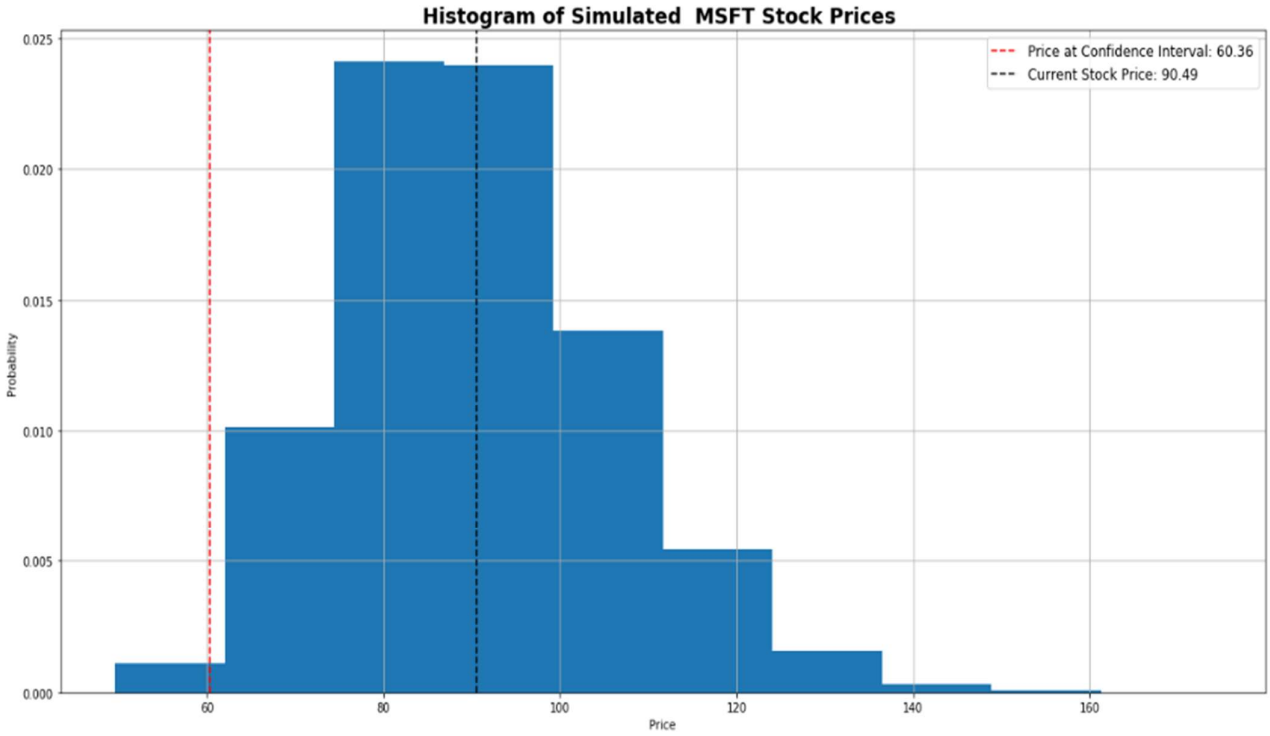
Document sign date :19-Sep-2019

D> MSFT STOCK

In [237]:
```
rets = Stock_MSFT.price_change
rets = rets[1:]
daily_vol = rets.std()
daily_ret = rets.mean()
simulation_df = pd.DataFrame()
num_simulations = 10000
predicted_days = 252
```

In [238]:
```
last_price = Stock_MSFT.iloc[-1:,0]
last_price = last_price[0]
for x in range(num_simulations):
    count = 0
    price_series = []
    price_series.append(last_price)
    for i in range(predicted_days):
        if count == 251:
            break
        price = price_series[count] * (1 + np.random.normal(0, daily_vol))
        price_series.append(price)
        count += 1
    simulation_df[x] = price_series
price_array = simulation_df.iloc[-1, :]
price_array = sorted(price_array, key=int)
var =  np.percentile(price_array, .95)
var1 =  np.percentile(price_array, .99)
var2 =  np.percentile(price_array, .9999)
print ("VaR at 95% Confidence: " + '${:,.2f}'.format(last_price - var))
print ("VaR at 99% Confidence: " + '${:,.2f}'.format(last_price - var1))
print ("VaR at 99.99% Confidence:" + '${:,.2f}'.format(last_price - var2))
plt.figure(figsize=(20,10))
plt.hist(price_array,normed=True)
plt.xlabel('Price')
plt.ylabel('Probability')
plt.title(r'Histogram of Simulated  AAPL Stock Prices ', fontsize=18, fontweight='bold')
plt.axvline(x=var, color='r', linestyle='--', label='Price at Confidence Interval: '  + str(round(var, 2)))
plt.axvline(x=last_price, color='k', linestyle='--', label = 'Current Stock Price: ' + str(round(last_price, 2)))
plt.legend(loc='upper right', fontsize = 'large')
plt.grid()
plt.show()
```

```
VaR at 95% Confidence: $30.13
VaR at 99% Confidence: $29.93
VaR at 99.99% Confidence:$29.86
```



Document sign date :19-Sep-2019

QUESTION 7: HOW DO WE CAN ATTEMPT TO PREDICT FUTURE STOCK BEHAVIOUR?

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable. Others disagree and those with this viewpoint possess myriad methods and technologies which purportedly allow them to gain future price information

For predicting opening price and closing price ,we have to import **norm** module from **scipy.stats** package library

7>.PREDICTING FUTURE STOCK BEHAVIOUR

In [84]:
```
from scipy.stats import norm
```

A>AAPL STOCK

In [86]:
```
log_returns=np.log(1+Stock_AAPL.price_change)
```

In [87]:
```
log_returns.tail()
```

Out[87]:
```
date
2018-02-01    0.003672
2018-02-02   -0.033694
2018-02-05   -0.016541
2018-02-06    0.051606
2018-02-07   -0.021977
Name: price_change, dtype: float64
```

In [88]:
```
u=pd.Series(log_returns.mean())
u
```

Out[88]:
```
0    0.000068
dtype: float64
```

In [89]:
```
var=pd.Series(log_returns.var())
var
```

Out[89]:
```
0    0.000141
dtype: float64
```

In [90]:
```
drift=u-(0.5*var)
drift
```

```
In [90]: drift=u-(0.5*var)
         drift

Out[90]: 0   -0.000002
         dtype: float64

In [91]: stdev=pd.Series(log_returns.std())

In [92]: stdev

Out[92]: 0    0.011864
         dtype: float64

In [93]: np.array(drift)

Out[93]: array([-2.1851437e-06])

In [94]: drift.values

Out[94]: array([-2.1851437e-06])

In [95]: stdev.values

Out[95]: array([0.01186432])

In [96]: norm.ppf(0.95)

Out[96]: 1.6448536269514722

In [97]: x=np.random.rand(10,2)

In [99]: x

Out[99]: array([[0.91267873, 0.11991812],
                [0.33001073, 0.03626091],
                [0.32249468, 0.33710437],
```

Document sign date :19-Sep-2019

```
In [97]: x=np.random.rand(10,2)

In [99]: x

Out[99]: array([[0.91267873, 0.11991812],
                [0.33001073, 0.03626091],
                [0.32249468, 0.33710437],
                [0.95285604, 0.67575476],
                [0.90954597, 0.49783213],
                [0.95375079, 0.34714103],
                [0.357152  , 0.4456322 ],
                [0.97780134, 0.80956799],
                [0.73428026, 0.05595194],
                [0.47615052, 0.47002418]])

In [100]: norm.ppf(x)

Out[100]: array([[ 1.35743658, -1.1753962 ],
                 [-0.43988353, -1.79582836],
                 [-0.46073414, -0.42037882],
                 [ 1.67320008,  0.45586024],
                 [ 1.33796437, -0.00543407],
                 [ 1.68236331, -0.39305067],
                 [-0.36608185, -0.13670447],
                 [ 2.01032014,  0.87630543],
                 [ 0.62581013, -1.58969363],
                 [-0.05981743, -0.07520908]])

In [101]: z=norm.ppf(np.random.rand(10,2))
          z

Out[101]: array([[-0.64724774,  1.61793272],
                 [ 0.60254757,  2.24607736],
                 [-0.34494304,  0.97910202],
                 [-0.904072  ,  2.25398645],
                 [ 0.40794404,  0.87105152],
```

In [102]:
```
t_intervals=1000
iteration=10
```

In [103]:
```
daily_returns=np.exp(drift.values+stdev.values*norm.ppf(np.random.rand(t_intervals,iteration)))
```

In [104]:
```
daily_returns
```

Out[104]:
```
array([[1.00164556, 1.01580954, 1.00993516, ..., 0.99247344, 1.00060551,
        1.00623722],
       [1.01139802, 0.98056742, 0.99550078, ..., 0.98424603, 1.00176914,
        0.99717516],
       [1.02090762, 0.99442363, 1.01466335, ..., 0.99513308, 1.0078675 ,
        1.01697633],
       ...,
       [1.01633394, 1.00122852, 0.99562353, ..., 0.98839404, 0.99467393,
        0.99607664],
       [0.98719943, 0.96965032, 0.99766022, ..., 0.98773702, 1.01256866,
        1.01077394],
       [0.99409633, 1.00975756, 1.01600989, ..., 1.04263073, 1.00969586,
        1.00416222]])
```

In [106]:
```
S0=Stock_AAPL[["open"]].iloc[-1]
```

In [110]:
```
S0
```

Out[110]:
```
open    163.085
Name: 2018-02-07 00:00:00, dtype: float64
```

In [111]:
```
price_list=np.zeros_like(daily_returns)
```

In [112]:
```
price_list
```

Out[112]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ....
```

```
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [113]:
```
price_list[0]
```

Out[113]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [114]:
```
price_list[0]=S0
```

In [115]:
```
price_list
```

Out[115]:
```
array([[163.085, 163.085, 163.085, ..., 163.085, 163.085, 163.085],
       [  0.   ,   0.   ,   0.   , ...,   0.   ,   0.   ,   0.   ],
       [  0.   ,   0.   ,   0.   , ...,   0.   ,   0.   ,   0.   ],
       ...,
       [  0.   ,   0.   ,   0.   , ...,   0.   ,   0.   ,   0.   ],
       [  0.   ,   0.   ,   0.   , ...,   0.   ,   0.   ,   0.   ],
       [  0.   ,   0.   ,   0.   , ...,   0.   ,   0.   ,   0.   ]])
```

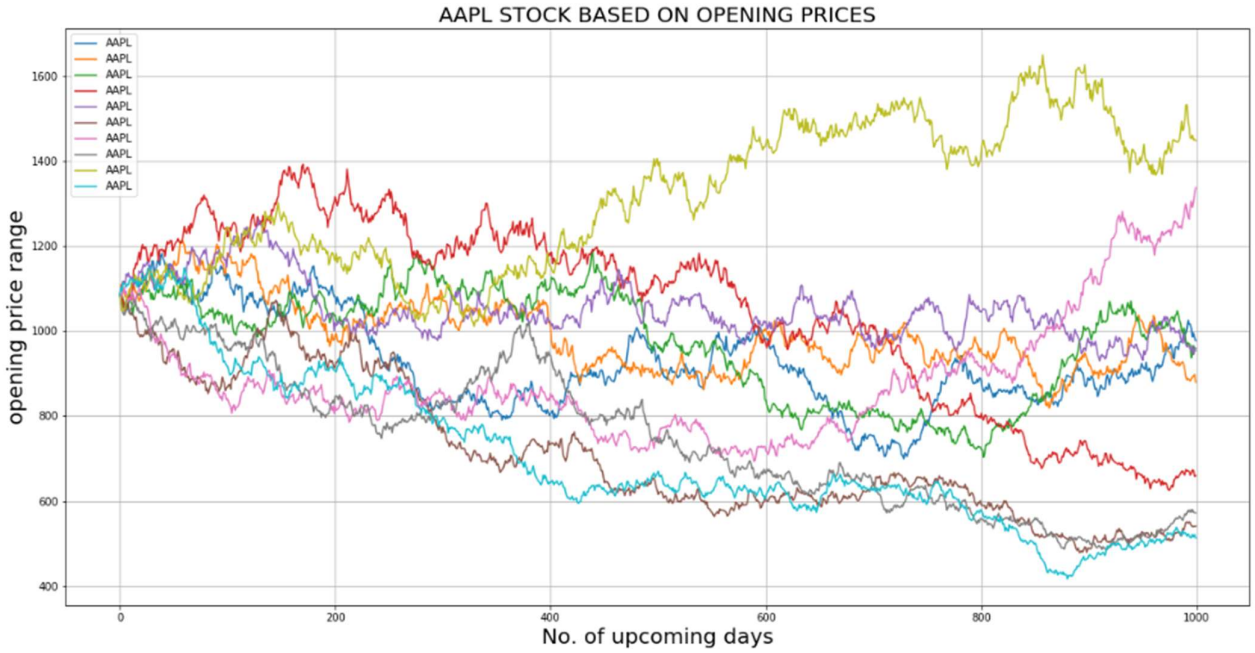
In [116]:
```
for t in range(1,t_intervals):
    price_list[t]=price_list[t-1]*daily_returns[t]
```

In [117]:
```
price_list
```

Out[117]:
```
array([[163.085     , 163.085     , 163.085     , ..., 163.085     ,
        163.085     , 163.085     ],
       [164.94384655, 159.91583761, 162.35124428, ..., 160.51576342,
        163.37352076, 162.62431172],
       [168.39242968, 159.02408695, 164.73185776, ..., 159.73454534,
        164.65886273, 165.38507623],
       ...,
       [108.21943328, 161.88019571, 193.66146405, ..., 185.75268979,
        128.60823265, 123.89953276],
       [106.83416298, 156.96718307, 193.20833853, ..., 183.47480846,
        130.22466563, 125.23441887],
       [106.20344888, 158.49880052, 196.30158315, ..., 191.29647401,
        131.48730625, 125.75567189]])
```

```
In [222]: plt.figure(figsize=(20,10))
          plt.plot(price_list ,label="AAPL")
          plt.title("AAPL STOCK BASED ON OPENING PRICES", fontsize=20)
          ax = plt.gca()
          ax.set_ylabel('opening price range',fontsize=20)
          ax.set_xlabel("No. of upcoming days", fontsize=20)
          plt.grid()
          plt.legend()
```

AAPL STOCK OPENING PRICE PREDICTION VISUALIZATION



Globsyn Knowledge Academy

Document sign date :19-Sep-2019

```
In [129]: S1=Stock_AAPL[["close"]].iloc[-1]

In [130]: S1

Out[130]: close    159.54
          Name: 2018-02-07 00:00:00, dtype: float64

In [133]: price_list=np.zeros_like(daily_returns)

In [134]: price_list

Out[134]: array([[0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 ...,
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.]])

In [135]: price_list[0]

Out[135]: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

In [136]: price_list[0]=S1

In [137]: price_list

Out[137]: array([[159.54, 159.54, 159.54, ..., 159.54, 159.54, 159.54],
                 [  0.  ,   0.  ,   0.  , ...,   0.  ,   0.  ,   0.  ],
                 [  0.  ,   0.  ,   0.  , ...,   0.  ,   0.  ,   0.  ],
                 ...,
                 [  0.  ,   0.  ,   0.  , ...,   0.  ,   0.  ,   0.  ],
                 [  0.  ,   0.  ,   0.  , ...,   0.  ,   0.  ,   0.  ],
                 [  0.  ,   0.  ,   0.  , ...,   0.  ,   0.  ,   0.  ]])
```

Document sign date :19-Sep-2019

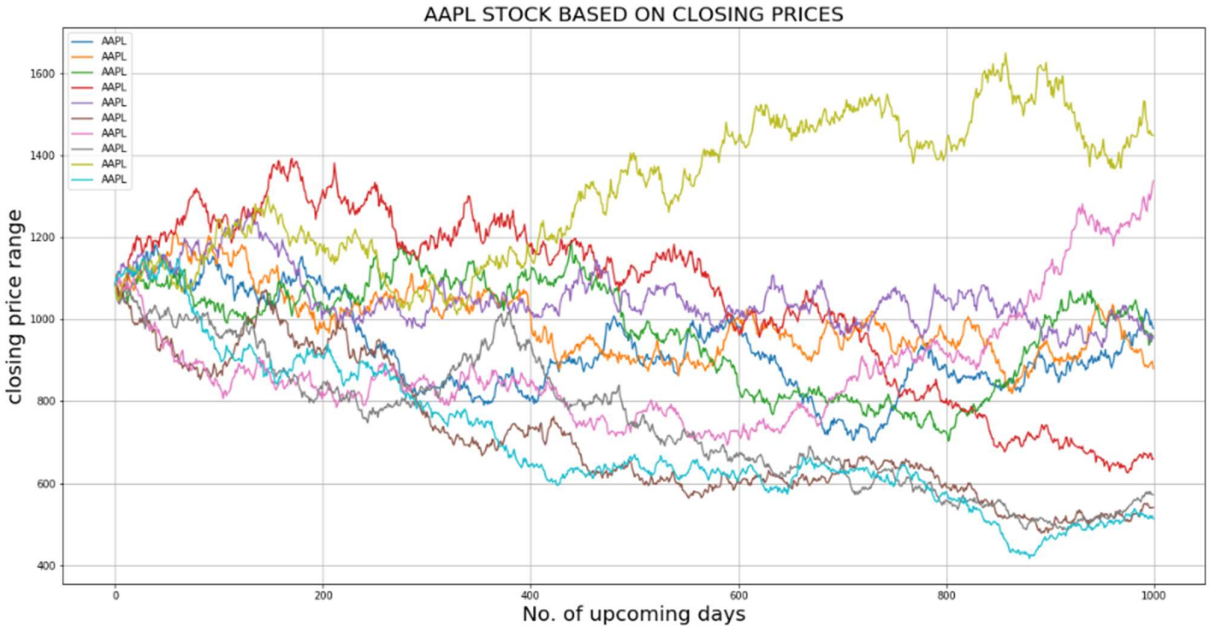
In [138]:
```
for t in range(1,t_intervals):
    price_list[t]=price_list[t-1]*daily_returns[t]
```

In [139]:
```
price_list
```

Out[139]:
```
array([[159.54      , 159.54      , 159.54      , ..., 159.54      ,
        159.54      , 159.54      ],
       [161.35844056, 156.43972611, 158.82219403, ..., 157.02661126,
        159.82224915, 159.08932576],
       [164.73206139, 155.56735955, 161.1510598 , ..., 156.26237461,
        161.07965147, 161.79007917],
       ...,
       [105.86705329, 158.3613847 , 189.45181944, ..., 181.71495925,
        125.81265866, 121.20631239],
       [104.51189479, 153.55516686, 189.00854358, ..., 179.48659252,
        127.39395502, 122.51218191],
       [103.89489061, 155.05349134, 192.03454993, ..., 187.13823751,
        128.62914946, 123.02210438]])
```

In [221]:
```
plt.figure(figsize=(20,10))
plt.plot(price_list ,label="AAPL")
plt.title("AAPL STOCK BASED ON CLOSING PRICES", fontsize=20)
ax = plt.gca()
ax.set_ylabel('closing price range',fontsize=20)
ax.set_xlabel("No. of upcoming days", fontsize=20)
plt.grid()
plt.legend()
```

AAPL STOCK CLOSING PRICE PREDICTION VISUALIZATION



Document sign date :19-Sep-2019

B>AMZN STOCK

In [141]:
```
log_returns=np.log(1+Stock_AMZN.price_change)
```

In [142]:
```
log_returns.tail()
```

Out[142]:
```
date
2018-02-01   -0.038806
2018-02-02   -0.032638
2018-02-05   -0.009038
2018-02-06    0.058056
2018-02-07   -0.022487
Name: price_change, dtype: float64
```

In [143]:
```
u=pd.Series(log_returns.mean())
u
```

Out[143]:
```
0    0.000088
dtype: float64
```

In [144]:
```
var=pd.Series(log_returns.var())
var
```

Out[144]:
```
0    0.000183
dtype: float64
```

In [145]:
```
drift=u-(0.5*var)
drift
```

Out[145]:
```
0   -0.000004
dtype: float64
```

In [146]:
```
stdev=pd.Series(log_returns.std())
```

```
In [146]: stdev=pd.Series(log_returns.std())

In [147]: stdev

Out[147]: 0    0.013541
          dtype: float64

In [148]: np.array(drift)

Out[148]: array([-3.8562456e-06])

In [149]: drift.values

Out[149]: array([-3.8562456e-06])

In [150]: stdev.values

Out[150]: array([0.01354133])

In [151]: norm.ppf(0.95)

Out[151]: 1.6448536269514722

In [152]: x=np.random.rand(10,2)

In [154]: x

Out[154]: array([[5.10291841e-01, 3.06888444e-01],
                 [3.00396622e-04, 9.45290086e-01],
                 [3.34515109e-01, 2.07261744e-01],
                 [1.43956637e-01, 2.38746947e-01],
                 [9.80171935e-01, 4.06528907e-02],
                 [8.02124870e-01, 8.51956036e-01],
                 [1.55627043e-01, 1.43996372e-01],
                 [9.95046368e-01, 7.84119134e-01],
                 [9.87752094e-01, 5.52898855e-01],
```

In [159]:
```
norm.ppf(x)
```

Out[159]:
```
array([[ 0.02580068, -0.50468957],
       [-3.43125604,  1.6008063 ],
       [-0.42747936, -0.81595917],
       [-1.06271046, -0.71033907],
       [ 2.05731297, -1.74315936],
       [ 0.84923551,  1.04485946],
       [-1.01259408, -1.06253529],
       [ 2.57904934,  0.78618053],
       [ 2.24926233,  0.13298873],
       [-1.77312734,  1.21321215]])
```

In [162]:
```
z=norm.ppf(np.random.rand(10,2))
z
```

Out[162]:
```
array([[-0.53185661,  0.04130204],
       [ 0.24072897,  0.02596939],
       [-0.43652324, -0.54507281],
       [ 2.91276688, -1.2292554 ],
       [-1.17373099, -0.39932568],
       [ 0.58359732,  0.65150131],
       [-0.40980224,  1.63881516],
       [-2.0332023 , -0.01026179],
       [-0.60735278,  0.4026912 ],
       [ 0.22157448,  0.43272877]])
```

In [157]:
```
t_intervals=1000
iteration=10
```

In [158]:
```
daily_returns=np.exp(drift.values+stdev.values*norm.ppf(np.random.rand(t_intervals,iteration)))
```

In [160]:
```
daily_returns
```

Out[160]:
```
array([[0.98317227, 1.03971905, 0.98029671, ..., 0.99159043, 1.01882894,
        0.9949099 ],
       [1.0119606 , 1.00472322, 0.96812884, ..., 1.0214852 , 1.00270738,
```

Document sign date :19-Sep-2019

```
       ...,
       [0.97749093, 1.0147195 , 0.98574803, ..., 1.01399695, 1.00502007,
        0.99289091],
       [0.99512181, 1.00546325, 1.0113699 , ..., 0.99707837, 0.98949802,
        1.00147813],
       [0.99763413, 0.99411476, 0.99866009, ..., 0.99416147, 0.99984189,
        0.97814051]])
```

In [163]:
```
S0=Stock_AMZN[["open"]].iloc[-1]
```

In [164]:
```
S0
```

Out[164]:
```
open    1449.0
Name: 2018-02-07 00:00:00, dtype: float64
```

In [165]:
```
price_list=np.zeros_like(daily_returns)
```

In [171]:
```
price_list
```

Out[171]:
```
array([[1449.        , 1449.        , 1449.        , ..., 1449.        ,
        1449.        , 1449.        ],
       [1466.33090905, 1455.84393964, 1402.81868687, ..., 1480.13205874,
        1454.37199386, 1480.55135824],
       [1455.91002889, 1428.64475359, 1397.23538868, ..., 1507.97873905,
        1432.65674752, 1501.01858182],
       ...,
       [1417.65556819, 2602.5944249 , 1033.7925512 , ..., 1637.97148131,
        1266.02072151, 1510.89112795],
       [1410.73997242, 2616.81306042, 1045.54666498, ..., 1633.18593881,
        1252.72500242, 1513.12441472],
       [1407.40234065, 2601.41249841, 1044.14572704, ..., 1623.65052749,
        1252.52692937, 1480.04828496]])
```

In [167]:
```
price_list[0]
```

Out[167]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [168]:
```
price_list[0]=S0
```

In [169]:
```
price_list
```

Out[169]:
```
array([[1449., 1449., 1449., ..., 1449., 1449., 1449.],
       [   0.,    0.,    0., ...,    0.,    0.,    0.],
       [   0.,    0.,    0., ...,    0.,    0.,    0.],
       ...,
       [   0.,    0.,    0., ...,    0.,    0.,    0.],
       [   0.,    0.,    0., ...,    0.,    0.,    0.],
       [   0.,    0.,    0., ...,    0.,    0.,    0.]])
```

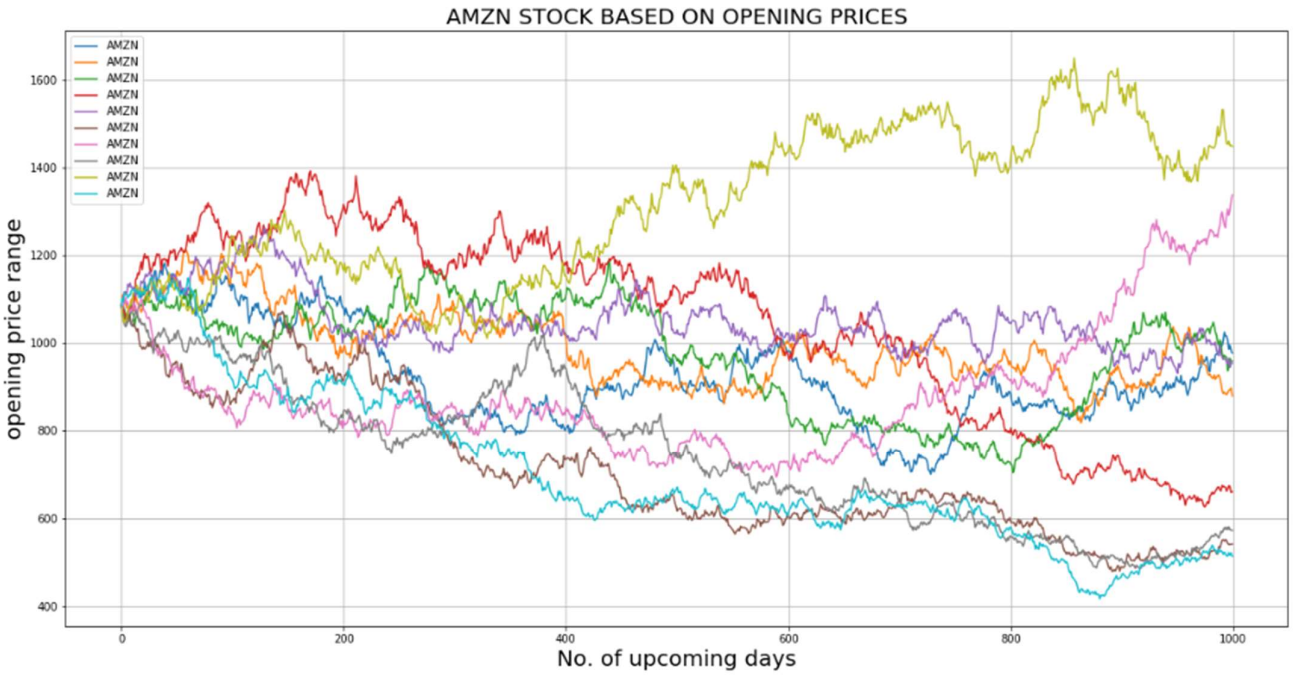
In [170]:
```
for t in range(1,t_intervals):
    price_list[t]=price_list[t-1]*daily_returns[t]
```

In [172]:
```
price_list
```

Out[172]:
```
array([[1449.        , 1449.        , 1449.        , ..., 1449.        ,
        1449.        , 1449.        ],
       [1466.33090905, 1455.84393964, 1402.81868687, ..., 1480.13205874,
        1454.37199386, 1480.55135824],
       [1455.91002889, 1428.64475359, 1397.23538868, ..., 1507.97873905,
        1432.65674752, 1501.01858182],
       ...,
       [1417.65556819, 2602.5944249 , 1033.7925512 , ..., 1637.97148131,
        1266.02072151, 1510.89112795],
       [1410.73997242, 2616.81306042, 1045.54666498, ..., 1633.18593881,
        1252.72500242, 1513.12441472],
       [1407.40234065, 2601.41249841, 1044.14572704, ..., 1623.65052749,
        1252.52692937, 1480.04828496]])
```

In [220]:
```
plt.figure(figsize=(20,10))
plt.plot(price_list ,label="AMZN")
plt.title("AMZN STOCK BASED ON OPENING PRICES", fontsize=20)
ax = plt.gca()
ax.set_ylabel('opening price range',fontsize=20)
ax.set_xlabel("No. of upcoming days", fontsize=20)
plt.grid()
plt.legend()
```

AMZN STOCK OPENING PRICE PREDICTION VISUALIZATION



Document sign date :19-Sep-2019

```
In [176]: S1=Stock_AMZN[["close"]].iloc[-1]

In [177]: S1

Out[177]: close    1416.78
          Name: 2018-02-07 00:00:00, dtype: float64

In [178]: price_list=np.zeros_like(daily_returns)

In [179]: price_list

Out[179]: array([[0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 ...,
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.]])

In [180]: price_list[0]

Out[180]: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

In [181]: price_list[0]=S1

In [182]: price_list

Out[182]: array([[1416.78, 1416.78, 1416.78, ..., 1416.78, 1416.78, 1416.78],
                 [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
                 [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
                 ...,
                 [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
                 [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
                 [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ]])
```

In [183]:
```
for t in range(1,t_intervals):
    price_list[t]=price_list[t-1]*daily_returns[t]
```

In [184]:
```
price_list
```

Out[184]:
```
array([[1416.78      , 1416.78      , 1416.78      , ..., 1416.78      ,
        1416.78      , 1416.78      ],
       [1433.72553853, 1423.47175763, 1371.6255757 , ..., 1447.2198055 ,
        1422.03254207, 1447.62978145],
       [1423.53637732, 1396.87737335, 1366.16642786, ..., 1474.44728634,
        1400.80015648, 1467.64189535],
       ...,
       [1386.13254376, 2544.7230706 , 1010.80511434, ..., 1601.54950676,
        1237.86945329, 1477.29491529],
       [1379.37072334, 2558.62554019, 1022.29786336, ..., 1596.8703757 ,
        1224.86937814, 1479.47854264],
       [1376.10730724, 2543.56742547, 1020.92807672, ..., 1587.54699402,
        1224.67570945, 1447.13789453]])
```

In [218]:
```
plt.figure(figsize=(20,10))
plt.plot(price_list ,label="AMZN")
plt.title("AMZN STOCK BASED ON CLOSING PRICES", fontsize=20)
ax = plt.gca()
ax.set_ylabel('closing price range',fontsize=20)
ax.set_xlabel("No. of upcoming days", fontsize=20)
plt.grid()
plt.legend()
```

AMZN STOCK CLOSING PRICE PREDICTION VISUALIZATION

Document sign date :19-Sep-2019

C>GOOGL STOCK

In [191]:
```
log_returns=np.log(1+Stock_GOOGL.price_change)
```

In [192]:
```
log_returns.tail()
```

Out[192]:
```
date
2018-02-01    0.004751
2018-02-02   -0.007318
2018-02-05   -0.035343
2018-02-06    0.047639
2018-02-07   -0.027623
Name: price_change, dtype: float64
```

In [193]:
```
u=pd.Series(log_returns.mean())
u
```

Out[193]:
```
0   -0.000228
dtype: float64
```

In [194]:
```
var=pd.Series(log_returns.var())
var
```

Out[194]:
```
0    0.00011
dtype: float64
```

In [195]:
```
drift=u-(0.5*var)
drift
```

Out[195]:
```
0   -0.000283
dtype: float64
```

In [196]:
```
stdev=pd.Series(log_returns.std())
stdev
```

Out[196]:
```
0    0.010489
```

```
In [196]: stdev=pd.Series(log_returns.std())
          stdev

Out[196]: 0    0.010489
          dtype: float64

In [197]: np.array(drift)

Out[197]: array([-0.00028339])

In [198]: drift.values

Out[198]: array([-0.00028339])

In [199]: stdev.values

Out[199]: array([0.01048944])

In [200]: norm.ppf(0.95)

Out[200]: 1.6448536269514722

In [201]: x=np.random.rand(10,2)
          x

Out[201]: array([[0.55055306, 0.93539301],
                 [0.97332434, 0.90247388],
                 [0.08024434, 0.697043  ],
                 [0.8863888 , 0.24304178],
                 [0.40096236, 0.7966413 ],
                 [0.35808031, 0.03114584],
                 [0.86647097, 0.42860547],
                 [0.88372355, 0.58834515],
                 [0.36647309, 0.05719163],
                 [0.3341289 , 0.10040188]])
```

Document sign date :19-Sep-2019

In [202]:
```
norm.ppf(x)
```

Out[202]:
```
array([[ 0.12705878,  1.51720877],
       [ 1.93206634,  1.29577727],
       [-1.40342992,  0.51591469],
       [ 1.20754482, -0.69655143],
       [-0.25085694,  0.82968411],
       [-0.36359479, -1.86421387],
       [ 1.10986302, -0.17992565],
       [ 1.19380844,  0.22329013],
       [-0.3412091 , -1.57879426],
       [-0.42854029, -1.27926499]])
```

In [203]:
```
z=norm.ppf(np.random.rand(10,2))
z
```

Out[203]:
```
array([[ 1.50339722,  0.37050181],
       [-0.47473042, -0.93255088],
       [-0.38071707, -1.36952056],
       [ 0.28422144,  3.27109494],
       [ 0.39829186, -0.36084289],
       [-1.40519835,  1.10870802],
       [-2.27947057, -2.37050019],
       [ 1.74293935, -0.64454483],
       [ 0.06617514,  1.31030128],
       [ 1.20835254,  1.7555962 ]])
```

In [204]:
```
t_intervals=1000
iteration=10
```

In [205]:
```
daily_returns=np.exp(drift.values+stdev.values*norm.ppf(np.random.rand(t_intervals,iteration)))
```

In [206]:
```
daily_returns
```

Out[206]:
```
array([[0.99250175, 1.00439046, 0.99470883, ..., 1.00007622, 0.99151931,
        0.99742815],
       [0.99442607, 1.02343082, 0.99361784, ..., 0.98034858, 0.9892386 ,
        1.0161845 ],
       [0.98915218, 0.98513374, 1.00774494, ..., 0.99288237, 0.97645743,
        0.99341901],
       ...,
       [1.00916172, 1.00105871, 1.01595888, ..., 0.98940697, 0.99267975,
        0.98886562],
       [0.98844857, 1.00551144, 0.99306332, ..., 0.99781296, 1.00099372,
        1.01087513],
       [0.9929229 , 0.97937894, 0.99526929, ..., 0.99903446, 0.99777952,
        0.98411272]])
```

In [207]:
```
S0=Stock_GOOGL[["open"]].iloc[-1]
```

In [208]:
```
S0
```

Out[208]:
```
open    1084.97
Name: 2018-02-07 00:00:00, dtype: float64
```

In [209]:
```
price_list=np.zeros_like(daily_returns)
```

In [210]:
```
price_list
```

Out[210]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

Globsyn Knowledge Kolkata

Document sign date :19-Sep-2019

In [211]:
```
price_list[0]
```

Out[211]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [212]:
```
price_list[0]=S0
```

In [213]:
```
price_list
```

Out[213]:
```
array([[1084.97, 1084.97, 1084.97, ..., 1084.97, 1084.97, 1084.97],
       [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
       [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
       ...,
       [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
       [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
       [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ]])
```

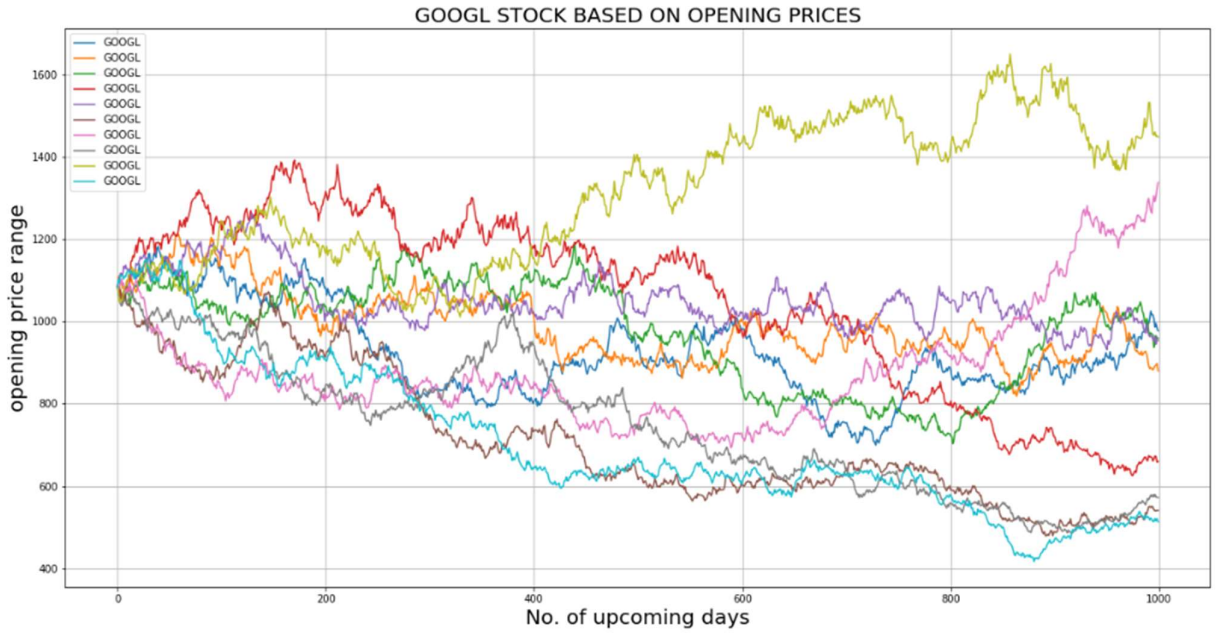
In [214]:
```
for t in range(1,t_intervals):
    price_list[t]=price_list[t-1]*daily_returns[t]
```

In [215]:
```
price_list
```

Out[215]:
```
array([[1084.97      , 1084.97      , 1084.97      , ..., 1084.97      ,
        1084.97      , 1084.97      ],
       [1078.92245027, 1110.39174049, 1078.04554607, ..., 1063.64880034,
        1073.29420098, 1102.52969338],
       [1067.21849458, 1093.88436755, 1086.39494764, ..., 1056.07814367,
        1048.0260947 , 1095.27395132],
       ...,
       [ 995.89690223,  893.33512398,  965.52958661, ...,  573.48797525,
        1450.83784945,  515.15729696],
       [ 984.39286395,  898.25868824,  958.83201353, ...,  572.23373197,
        1452.27957881,  520.7596979 ],
       [ 977.42622192,  879.73564394,  954.29605882, ...,  571.68121546,
        1449.05482498,  512.48624168]])
```

```
In [216]: plt.figure(figsize=(20,10))
          plt.plot(price_list ,label="GOOGL")
          plt.title("GOOGL STOCK BASED ON OPENING PRICES", fontsize=20)
          ax = plt.gca()
          ax.set_ylabel('opening price range',fontsize=20)
          ax.set_xlabel("No. of upcoming days", fontsize=20)
          plt.grid()
          plt.legend()

Out[216]: <matplotlib.legend.Legend at 0x1b2c045f860>
```

GOOGL STOCK OPENING PRICE PREDICTION VISUALIZATION



Document sign date :19-Sep-2019

```
In [223]: S1=Stock_GOOGL[["close"]].iloc[-1]

In [224]: S1

Out[224]: close    1055.41
          Name: 2018-02-07 00:00:00, dtype: float64

In [225]: price_list=np.zeros_like(daily_returns)

In [226]: price_list

Out[226]: array([[0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 ...,
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.]])

In [227]: price_list[0]

Out[227]: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

In [228]: price_list[0]=S1

In [229]: price_list

Out[229]: array([[1055.41, 1055.41, 1055.41, ..., 1055.41, 1055.41, 1055.41],
                 [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
                 [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
                 ...,
                 [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
                 [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ],
                 [   0.  ,    0.  ,    0.  , ...,    0.  ,    0.  ,    0.  ]])
```

In [230]:
```
for t in range(1,t_intervals):
    price_list[t]=price_list[t-1]*daily_returns[t]
```

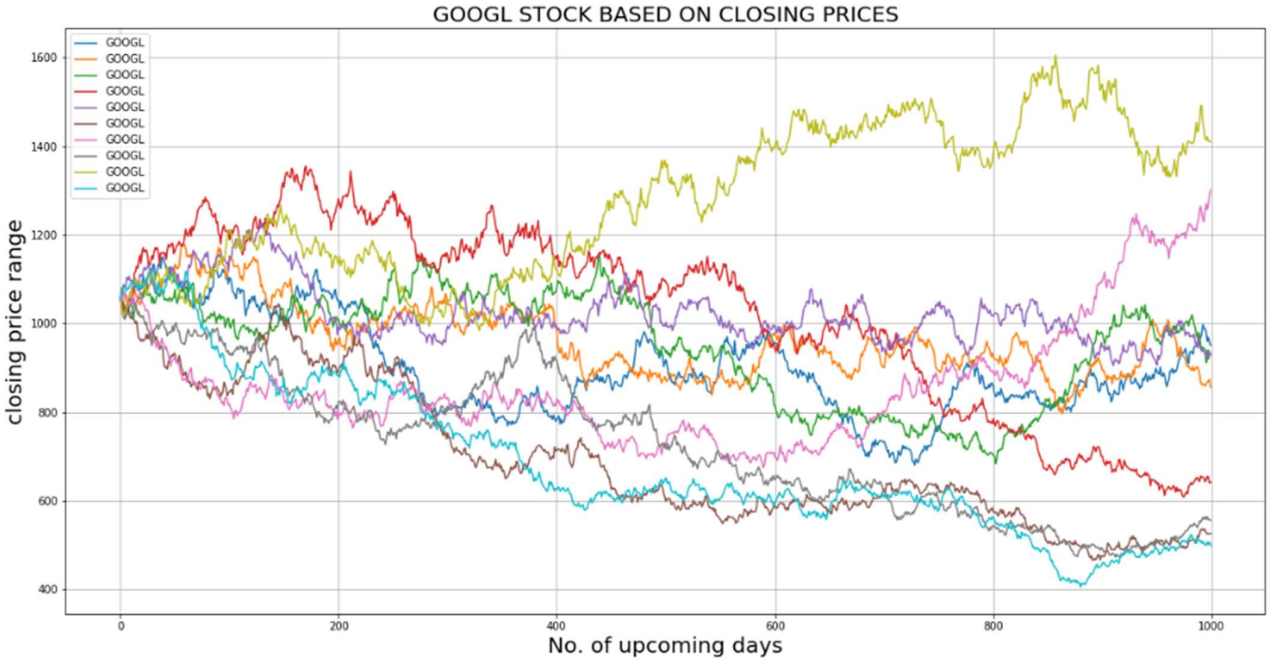
In [231]:
```
price_list
```

Out[231]:
```
array([[1055.41      , 1055.41      , 1055.41      , ..., 1055.41      ,
        1055.41      , 1055.41      ],
       [1049.52721572, 1080.13912535, 1048.67420277, ..., 1034.66969627,
        1044.05230804, 1072.49127965],
       [1038.14213422, 1064.08149567, 1056.79612495, ..., 1027.30530209,
        1019.47263114, 1065.43322024],
       ...,
       [ 968.76369815,  868.99621482,  939.22373983, ...,  557.86329941,
        1411.30978247,  501.12184004],
       [ 957.57308731,  873.78563661,  932.70864208, ...,  556.64322798,
        1412.71223192,  506.57160361],
       [ 950.7962514 ,  855.76725252,  928.29626943, ...,  556.10576478,
        1409.57533649,  498.52355764]])
```

In [232]:
```
plt.figure(figsize=(20,10))
plt.plot(price_list ,label="GOOGL")
plt.title("GOOGL STOCK BASED ON CLOSING PRICES", fontsize=20)
ax = plt.gca()
ax.set_ylabel('closing price range',fontsize=20)
ax.set_xlabel("No. of upcoming days", fontsize=20)
plt.grid()
plt.legend()
```

Document sign date :19-Sep-2019

GOOGL STOCK CLOSING PRICE PREDICTION VISUALIZATION



Document sign date :19-Sep-2019

D> MSFT STOCK

In [233]:
```
log_returns=np.log(1+Stock_MSFT.price_change)
```

In [234]:
```
log_returns.tail()
```

Out[234]:
```
date
2018-02-01   -0.005607
2018-02-02   -0.020063
2018-02-05   -0.028676
2018-02-06    0.049836
2018-02-07   -0.009772
Name: price_change, dtype: float64
```

In [235]:
```
u=pd.Series(log_returns.mean())
u
```

Out[235]:
```
0    0.00082
dtype: float64
```

In [236]:
```
var=pd.Series(log_returns.var())
var
```

Out[236]:
```
0    0.000112
dtype: float64
```

In [237]:
```
drift=u-(0.5*var)
drift
```

Out[237]:
```
0    0.000764
dtype: float64
```

```
Out[238]: 0    0.010591
          dtype: float64

In [239]: np.array(drift)

Out[239]: array([0.00076432])

In [240]: drift.values

Out[240]: array([0.00076432])

In [241]: stdev.values

Out[241]: array([0.01059126])

In [242]: norm.ppf(0.95)

Out[242]: 1.6448536269514722

In [243]: x=np.random.rand(10,2)
          x

Out[243]: array([[0.34486491, 0.52355163],
                 [0.70473442, 0.26372507],
                 [0.08145522, 0.13973049],
                 [0.71453633, 0.42887335],
                 [0.55700755, 0.44899103],
                 [0.57155863, 0.45829732],
                 [0.9471079 , 0.35590126],
                 [0.36375446, 0.78690492],
                 [0.19704163, 0.08515417],
                 [0.46073043, 0.822803  ]])
```

In [244]:
```
norm.ppf(x)
```

Out[244]:
```
array([[-0.39922175,  0.0590695 ],
       [ 0.53806646, -0.63190318],
       [-1.3953496 , -1.081531  ],
       [ 0.56668628, -0.17924325],
       [ 0.14338657, -0.12821092],
       [ 0.18034372, -0.10472423],
       [ 1.61743598, -0.36943634],
       [-0.34844112,  0.79572796],
       [-0.85223576, -1.37121371],
       [-0.09859373,  0.92610004]])
```

In [245]:
```
z=norm.ppf(np.random.rand(10,2))
z
```

Out[245]:
```
array([[-1.8719294 , -1.55564156],
       [-2.86287977,  0.84725816],
       [-0.45655789, -0.98539484],
       [-0.71965623,  0.41884251],
       [-0.16986149, -0.87695981],
       [ 0.07434706,  1.43678699],
       [-0.499658  ,  0.23521738],
       [-1.4860444 , -0.76990828],
       [-0.87156783,  0.01625264],
       [ 0.13119784, -1.98351606]])
```

In [246]:
```
t_intervals=1000
iteration=10
```

In [247]:
```
daily_returns=np.exp(drift.values+stdev.values*norm.ppf(np.random.rand(t_intervals,iteration)))
```

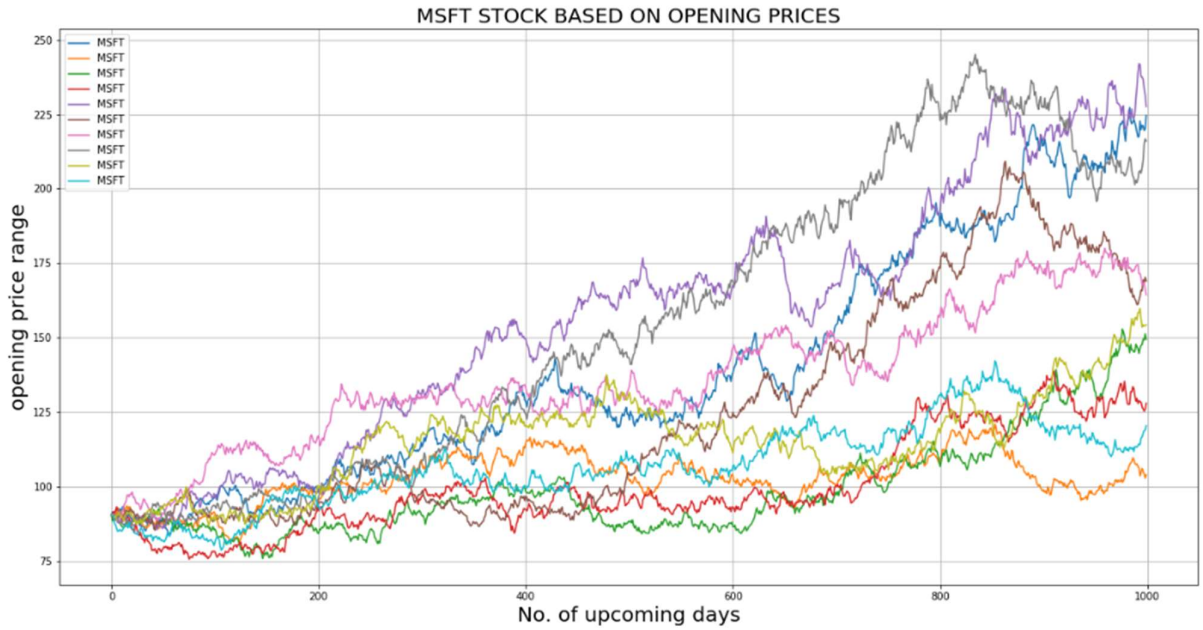
In [248]:
```
daily_returns
```

```
Out[248]: array([[1.00621849, 1.01223609, 0.97652013, ..., 0.99463052, 1.02655693,
                  1.00446019],
                 [1.01066822, 0.99077816, 0.97767272, ..., 0.99593549, 0.99754262,
                  0.9999499 ],
                 [0.98730029, 1.01431981, 1.00890903, ..., 0.99908723, 0.9948987 ,
                  0.98817334],
                 ...,
                 [0.99334456, 0.99119738, 0.99810691, ..., 1.02661651, 0.99764582,
                  1.00787297],
                 [1.00253214, 0.98751532, 1.01385994, ..., 1.00724505, 1.00014267,
                  1.00092832],
                 [1.02026297, 1.01104464, 0.98801108, ..., 0.99687205, 1.00223826,
                  1.01252053]])

In [249]: S0=Stock_MSFT[["open"]].iloc[-1]

In [250]: S0

Out[250]: open    90.49
          Name: 2018-02-07 00:00:00, dtype: float64

In [251]: price_list=np.zeros_like(daily_returns)

In [252]: price_list

Out[252]: array([[0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 ...,
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.]])

In [253]: price_list[0]
```

```
In [253]: price_list[0]

Out[253]: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

In [254]: price_list[0]=S0

In [255]: price_list

Out[255]: array([[90.49, 90.49, 90.49, ..., 90.49, 90.49, 90.49],
                 [ 0.  ,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ],
                 [ 0.  ,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ],
                 ...,
                 [ 0.  ,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ],
                 [ 0.  ,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ],
                 [ 0.  ,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ]])

In [256]: for t in range(1,t_intervals):
              price_list[t]=price_list[t-1]*daily_returns[t]

In [257]: price_list

Out[257]: array([[ 90.49      ,  90.49      ,  90.49      , ...,  90.49      ,
                    90.49      ,  90.49      ],
                 [ 91.45536684,  89.65551582,  88.46960472, ...,  90.12220213,
                    90.26763143,  90.48546681],
                 [ 90.29390982,  90.93936615,  89.25778268, ...,  90.0399411 ,
                    89.80714942,  89.41532607],
                 ...,
                 [219.61229554, 104.10825477, 149.18507851, ..., 215.18076065,
                  153.95229245, 118.79619458],
                 [220.16838428, 102.80849698, 151.25277545, ..., 216.73975704,
                  153.97425627, 118.90647519],
                 [224.62965006, 103.94397986, 149.4394187 , ..., 216.0618061 ,
                  154.31889056, 120.39524715]])
```

```
In [258]: plt.figure(figsize=(20,10))
          plt.plot(price_list ,label="MSFT")
          plt.title("MSFT STOCK BASED ON OPENING PRICES", fontsize=20)
          ax = plt.gca()
          ax.set_ylabel('opening price range',fontsize=20)
          ax.set_xlabel("No. of upcoming days", fontsize=20)
          plt.grid()
          plt.legend()

Out[258]: <matplotlib.legend.Legend at 0x1b2c0e721d0>
```

MSFT STOCK OPENING PRICE PREDICTION VISUALIZATION



Document sign date :19-Sep-2019

```
In [259]: S1=Stock_MSFT[["close"]].iloc[-1]

In [260]: S1

Out[260]: close    89.61
          Name: 2018-02-07 00:00:00, dtype: float64

In [262]: price_list=np.zeros_like(daily_returns)

In [263]: price_list

Out[263]: array([[0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 ...,
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.]])

In [264]: price_list[0]

Out[264]: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

In [265]: price_list[0]=S1

In [266]: price_list

Out[266]: array([[89.61, 89.61, 89.61, ..., 89.61, 89.61, 89.61],
                 [ 0.  ,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ],
                 [ 0.  ,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ],
                 ...,
                 [ 0.  ,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ],
                 [ 0.  ,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ],
                 [ 0.  ,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ]])
```

In [267]:
```
for t in range(1,t_intervals):
    price_list[t]=price_list[t-1]*daily_returns[t]
```

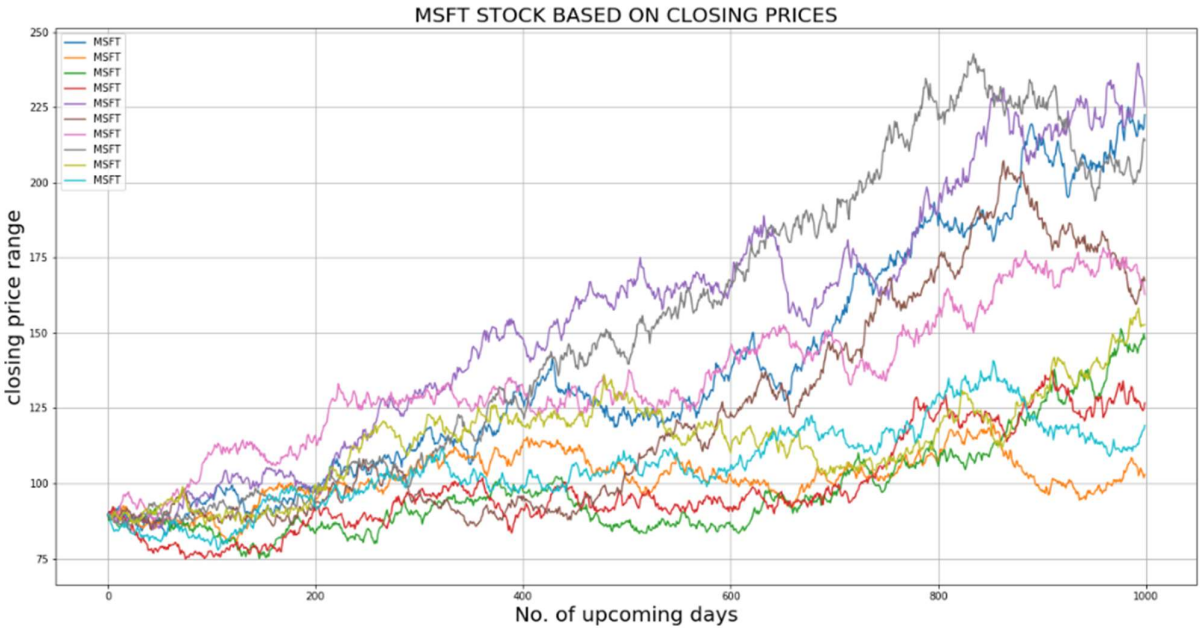
In [268]:
```
price_list
```

Out[268]:
```
array([[ 89.61      ,  89.61      ,  89.61      , ...,  89.61      ,
         89.61      ,  89.61      ],
       [ 90.56597881,  88.78363104,  87.60925273, ...,  89.2457789 ,
         89.38979393,  89.6055109 ],
       [ 89.41581676,  90.05499614,  88.38976578, ...,  89.16431785,
         88.93379003,  88.54577709],
       ...,
       [217.47660298, 103.09581954, 147.73427876, ..., 213.08816402,
        152.45513235, 117.6409216 ],
       [218.02728385, 101.80870167, 149.7818677 , ..., 214.63199943,
        152.47688257, 117.75012976],
       [222.44516457, 102.93314218, 147.98614554, ..., 213.96064144,
        152.81816536, 119.22442366]])
```

In [269]:
```
plt.figure(figsize=(20,10))
plt.plot(price_list ,label="MSFT")
plt.title("MSFT STOCK BASED ON CLOSING PRICES", fontsize=20)
ax = plt.gca()
ax.set_ylabel('closing price range',fontsize=20)
ax.set_xlabel("No. of upcoming days", fontsize=20)
plt.grid()
plt.legend()
```

Out[269]:
```
<matplotlib.legend.Legend at 0x1b2c0f046a0>
```

MSFT STOCK CLOSING PRICE PREDICTION VISUALIZATION



Document sign date :19-Sep-2019

Certificate

This is to certify that SOMNATH CHAKRABORTY of GOVERNMENT COLLEGE OF ENGINEERING & CERAMIC TECHNOLOGY,

registration number: 161130110068 of 2016-2017, has successfully completed a project on “STOCK MARKET ANALYSIS” using Python under the guidance of Prof. TITAS ROYCHOUDHURY

TITAS ROYCHOUDHURI

Globsyn Finishing School

Document sign date :19-Sep-2019

Certificate

This is to certify that SWARNENDU BISWAS of GOVERNMENT COLLEGE OF ENGINEERING & CERAMIC TECHNOLOGY,

registration number: 161130110073 of 2016-2017, has successfully completed a project on “STOCK MARKET ANALYSIS” using Python under the guidance of Prof. TITAS ROYCHOUDHURY

TITAS ROYCHOUDHURI

Globsyn Finishing School

Document sign date :19-Sep-2019

Certificate

This is to certify that RAJARSHI BHOWMIK of GOVERNMENT COLLEGE OF ENGINEERING & CERAMIC TECHNOLOGY,

registration number: 161130110058 of 2016-2017, has successfully completed a project on “STOCK MARKET ANALYSIS” using Python under the guidance of Prof. TITAS ROYCHOUDHURY

TITAS ROYCHOUDHURI

Globsyn Finishing School

Document sign date :19-Sep-2019