Report on

IMPACT OF COVID-19 ON STOCK MARKET 6 COUNTRIES

Submitted By :

ASHUTOSH KUMAR SINGH

2K18/MBA/905

DEEPAK SHARMA

2K18/MBA/906

Under the Guidance of :

SOMNATH MITRA

UNIVERSITY SCHOOL OF MANAGEMENT & ENTREPRENEURSHIP

Delhi Technological University

MAY 2020

DECLARATION

I , ASHUTOSH KUMAR SINGH bearing Reg. No 2K18/MBA/905 hereby declare that this project report entitled **IMPACT OF COVID-19 ON STOCK MARKET 6 COUNTRIES** has been prepared by me towards the partial fulfillment of the requirement for the award of the Master of Business Administration (MBA) Degree under the guidance o**f SOMNATH MITRA AND DR. DEEPTI AGGARWAL**

I also declare that this project report is my original work and has not been previously submitted for the award of any Degree, Diploma, Fellowship, or other similar titles.

Place:
ASHUTOSH KUMAR SINGH

Date:
2K18/MBA/905

DECLARATION

I , DEEPAK SHARMA bearing Reg. No 2K18/MBA/906 hereby declare that this project report entitled **IMPACT OF COVID-19 ON STOCK MARKET 6 COUNTRIES** has been prepared by me towards the partial fulfillment of the requirement for the award of the Master of Business Administration (MBA) Degree under the guidance o**f SOMNATH MITRA AND DR. DEEPTI AGGARWAL**

I also declare that this project report is my original work and has not been previously submitted for the award of any Degree, Diploma, Fellowship, or other similar titles.

Place:
DEEPAK SHARMA

Date:
2K18/MBA/906

ACKNOWLEDGEMENT

Every project, big or small, is successful largely due to the collective efforts of wonderful people who have never shied from lending a helping hand or giving valuable advice. I extend my sincere gratitude to these wonderful people, for their constant support, guidance, and motivation that I could undertake this project and making it a success. I, student of USME DTU, extremely grateful to Mr. Somnath Mitra for the confidence shown in me and providing me with the opportunity to pursue my Project semester training. At this juncture, I feel deeply honored in expressing my sincere thanks to my mentor for his constant guidance and supervision and making resources available at the right time for the completion of the projects. My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

ABSTRACT

No previous infectious disease outbreak, including the Spanish Flu, has impacted the stock market as powerfully as the COVID-19 pandemic. We use Machine learning algorithms to forecast the impact of this pandemic on the stock market of top 6 performing countries. We also argue that policy responses to the COVID-19 pandemic provide the most compelling explanation for its unprecedented stock market impact. As the Novel Coronavirus (COVID-19) spread from a regional crisis in China's Hubei Province to a global pandemic, equities plummeted and market volatility rocketed upwards around the world. In the India, half of the Nifty 50 stocks, excluding financials, are trading at single-digit valuations as Indian equity markets are nearly one-third off record-high levels they scaled in January. Motivated by these observations, we consider the role of COVID-19 developments in recent stock market behavior and draw comparisons between different stock indexes across the world. To quantify the data comparison we use Python language as a tool to get the more clear forecast. Looking back since independence, we find no other infectious disease outbreak that had more than a tiny effect on International stock market volatility. Looking back to 1900, we find not a single instance in which contemporary newspaper accounts attributed a large daily market move to pandemic-related developments. That includes the Spanish Flu of 1918-20, which killed an estimated 2.0 percent of the world's population (Barro, Ursua and Weng, 2020). In striking contrast, news related to COVID-19 developments is overwhelmingly the dominant driver of large daily international stock market moves since 31st December,2019

EXECUTIVE SUMMARY

In this world of globalization each and every economy is interconnected and they have an effect when one economy going down and other up .For the macroeconomic stability of the country it is better to understand the stock market of that particular country .The index of the stock market gave the total real time economic picture of the country. Similarly when one is want to have the economic picture of the whole world it is better to go for performance of the large economies of the world respectively. It could augment ones understanding to deduce an understanding base on the current situations and probably it could also be used to make an strategy to rectify the current slowdown based on the comparative analysis among the other economies. Based on these thoughts we are trying to analyse the situation during the pandemic.

Various governments had announced the stimulus package their effectiveness on the arket sentiments is to be analysed in the real time by looking at the stock market indices. The government financial packages is itself an burden of future days and when the whole of the big economy announceing these measure the certainly had an adverse effect which needs to be studied . so we are trying to coreelate these economies on the basis of stock market readings that will give an clear picture of the risk that these countries are facing in financial term.

TABLE OF CONTENT

INTRODUCTION

Stock market prediction using machine learning

One of the most prominent use cases of machine learning is “Fintech” (Financial Technology for those who aren't buzz-word aficionados); a large subset of which is in the stock market. Financial theorists, and data scientists for the better part of the last 50 years, have been employed to make sense of the marketplace in order to increase return on investment. However, due to the multidimensional nature of the problem, the scale of the system, and inherent variation with time, it has been an overwhelmingly tough challenge for humans to solve, even with the assistance of conventional data analytics tools.

The general research associated with the stock or share market is highly focusing on neither buy nor sell but it fails to address the dimensionality and expectancy of a new investor. The common trend towards the stock market among the society is that it is highly risky for investment or not suitable for trade so most of the people are not even interested.

"The world has rarely seen a crisis of this magnitude, and no one can stand on the sidelines; we cannot leave any country behind in our response. We have provided emergency support to 30 countries across Africa so far, with more to come, and will continue to advocate for debt relief and increased resources, especially for those countries hardest hit by COVID-19."

World Bank Group President David Malpas

“IMF projecting 3% contraction of global GDP due to COVID-19 pandemic”

Gita Gopinath

OBJECTIVE OF THE STUDY

To predict the effect of the pandemic on the stock market of top 6 countries and to forecast the behavior of stock market. thereby applying the machine learning algorithm on the stock market data and build the system to predict the, market behavior during the COVID -19

To apply the data analytics tool and statistics on the stock market data to reach a conclusion. The mixture of data analytics in the stock market prediction can help in decision making for individual as well as the companies

To test the accuracy of the data driven project on the real-life scenarios and test their outcomes which further able to leverage our knowledge in the field of the data analytics and machine learning.

To study the kind of uncertainties in the market due to the lockdown and shutdowns in many large parts of the world and understand their relations with respect to each other.

To get the clear picture of the market sentiments prevailing in the market now a days to analyze their effects their pros and cons to other economies

LITERATURE REVIEW

A good number of articles, research papers ,and studies has been conducted on the topic of stock market prediction using machine language .Yet there is always an uncertainty or no model that could accurately predict the market sentiments

For ours study we use the research papers from the International Research Journal of Engineering and Technology (IRJET) the author proposes support vector machine (SVM) for prediction

There are 3 types of machines in data analytics on which the studies are focused **ARIMA** (Autoregressive integrated moving average) has an embedded weakness---that is it the predict value is regression against the prior value. This is a good way to see the past, but not a good enough means to see the future. ARIMA is used when there is a shock and the effect of the shock is "integrated" into the system, thus, destroyed the old equilibrium, i.e. the subsequent series establish the new mean. In risk analysis or stock exchange studies, this is often a weak modeling. it is a honest tool to ascertain where has the stock (time series data) had been. It is less robust to answer the question: where will the time series go?

NEURAL NETWORK: This is an improvement over ARMA (Autoregressive moving average [without integration]: Box-Jenkins) and ARIMA because NN takes empirical risk into account. This allows local optimum. For short-term stock investment, it might work well.

SUPPORT VECTOR MACHINE (SVM): It is also a step up from ARIMA. Similar to NN, SVM consider risk as a relevant things about building the model. SVM uses "structural risk" in modeling; this enables SVM to realise "global optimum." in practical term for stock investment, it may mean that a global optimum may be used as a tool for long-term stock selection or investment, i.e. how sensitive the saftey is to the effect of structural shock?

With these things in our mind we will go for ARIMA model for prediction.

RESEARCH METHODOLOGY

Ours research methodology includes the following aspects :

STAGE 1 **(DATA COLLECTION)**	**Looking for the top performing countries** **Analyze their best performing stock indices** **Take the suitable one indices** **Collect the historical data of the index for past 3 months**
STAGE 2 (DATA ANALYSIS)	Using the PYTHON LANGUAGE Draw comparison between the death rate and market

Research hypothesis:

- The effect of the COVID -19 is continue to shaken the economies for next 3 months
- Countries of observations are in the same state as they are during our prediction Government measures and other external stimulus to revive the economy is not taken into the consideration
- The trend of stock market continue to repeat itself .
- The inconsistencies of the data is already taken care off but uncertainties of future are ignored

In [1]:

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

In [2]:

```
dataset=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//nse nifty 50 data.csv')
dataset_2=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//nse april data.csv')
```

In [3]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64 entries, 0 to 63
Data columns (total 7 columns):
Date                 64 non-null object
Open                 64 non-null float64
High                 64 non-null float64
Low                  64 non-null float64
Close                64 non-null float64
Shares Traded        64 non-null int64
Turnover (Rs. Cr)    64 non-null float64
dtypes: float64(5), int64(1), object(1)
memory usage: 3.6+ KB
```

In [4]:

```
dataset.head()
```

Out[4]:

	Date	Open	High	Low	Close	Shares Traded	Turnover (Rs. Cr)
0	01-Jan-2020	12202.15	12222.20	12165.30	12182.50	304078039	10445.68
1	02-Jan-2020	12198.55	12289.90	12195.25	12282.20	407697594	15256.55
2	03-Jan-2020	12261.10	12265.60	12191.35	12226.65	428770054	16827.27
3	06-Jan-2020	12170.60	12179.10	11974.20	11993.05	396501419	16869.22
4	07-Jan-2020	12079.10	12152.15	12005.35	12052.95	447818617	17797.68

TABLE 1

In [5]:

```
type(dataset)
```

Out[5]:

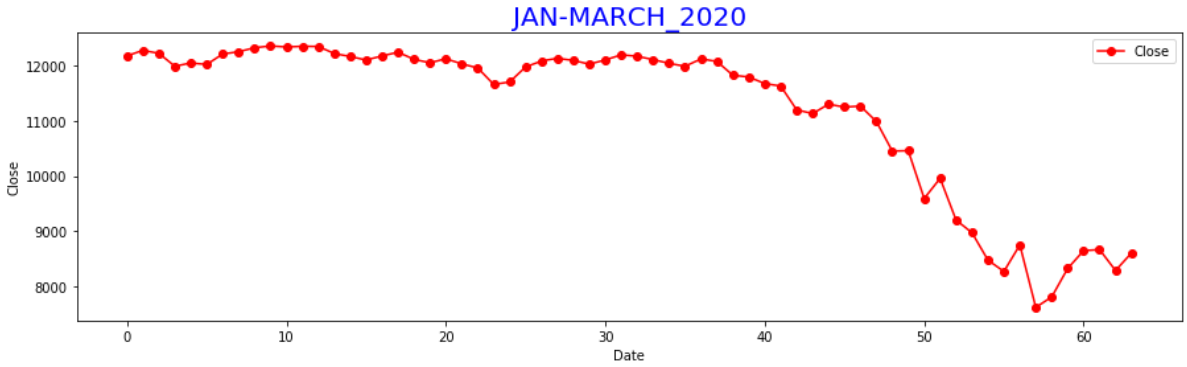
```
pandas.core.frame.DataFrame
```

In [6]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Close')
plt.plot(dataset['Close'],color='red',marker='o',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[6]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



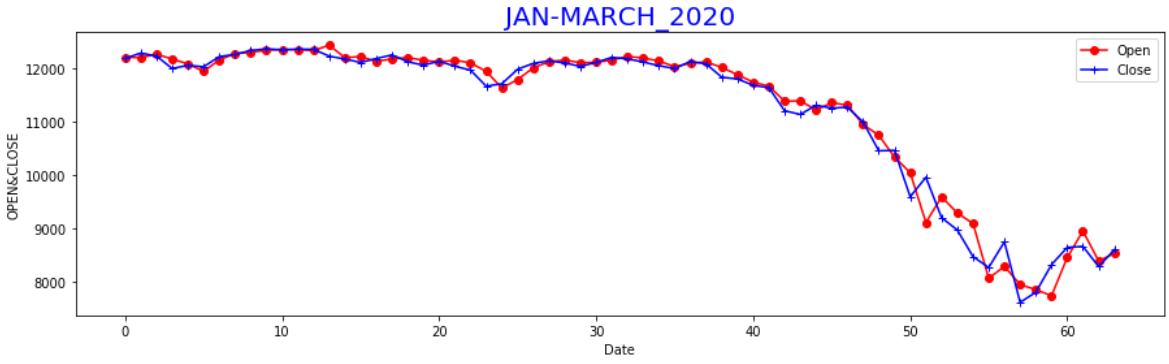
GRAPH 1

In [7]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('OPEN&CLOSE')
plt.plot(dataset['Open'],color='red',marker='o',label='Open')
plt.plot(dataset['Close'],color='blue',marker='+',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[7]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



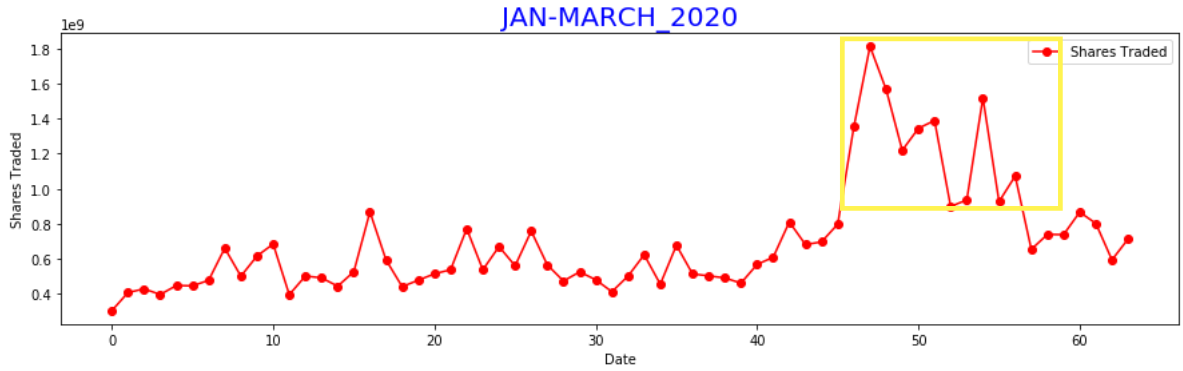
GRAPH 2

In [8]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Shares Traded')
plt.plot(dataset['Shares Traded'],color='red',marker='o',label='Shares Traded')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[8]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```

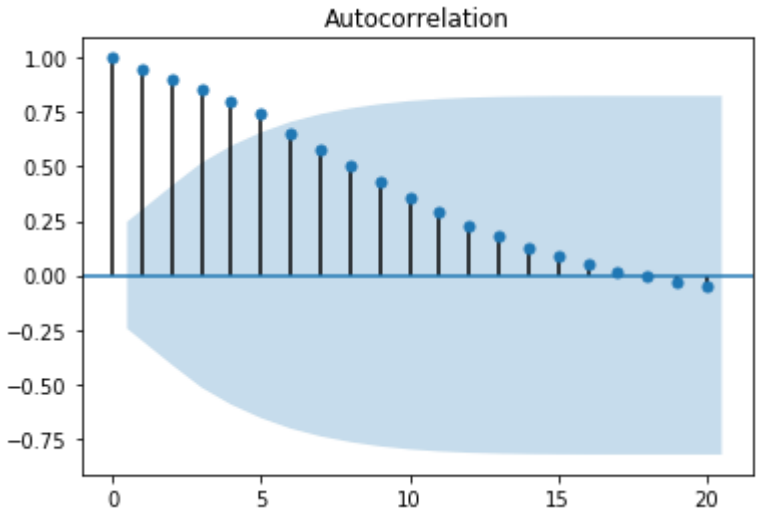


GRAPH 3

In [9]:

```
plt.figure(figsize=(15,4))
#CHECKING STATIONARY OF THE DATA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
# Show autocorrelation upto lag 10
acf_plot = plot_acf( dataset.Close,
lags=20)
```

```
<Figure size 1080x288 with 0 Axes>
```



GRAPH 4

In [10]:

```
dataset['Close_diff'] = dataset.Close - dataset.Close.shift(1)
dataset_diff=dataset.dropna()
dataset_diff.head()
```

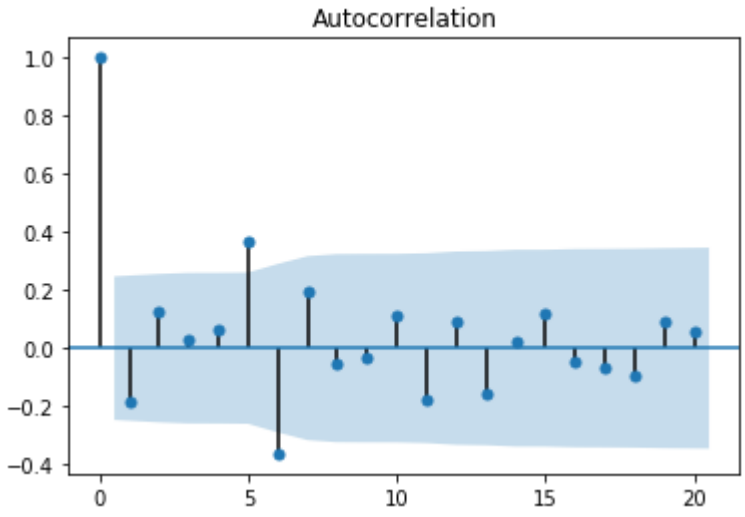
Out[10]:

	Date	Open	High	Low	Close	Shares Traded	Turnover (Rs. Cr)	Close_diff
1	02-Jan-2020	12198.55	12289.90	12195.25	12282.20	407697594	15256.55	99.70
2	03-Jan-2020	12261.10	12265.60	12191.35	12226.65	428770054	16827.27	-55.55
3	06-Jan-2020	12170.60	12179.10	11974.20	11993.05	396501419	16869.22	-233.60
4	07-Jan-2020	12079.10	12152.15	12005.35	12052.95	447818617	17797.68	59.90
5	08-Jan-2020	11939.10	12044.95	11929.60	12025.35	445991640	18281.15	-27.60

TABLE 2

In [11]:

```
acf_plot = plot_acf( dataset_diff.Close_diff,
lags=20)
```



GRAPH 5

In [12]:

```
from statsmodels.tsa.arima_model import ARIMA
arima = ARIMA( dataset.Close.astype(np.float64).as_matrix(),
order = (0,1,1))
ar_model = arima.fit()
ar_model.summary2()
```

```
C:\Users\deepa\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWa
rning: Method .as_matrix will be removed in a future version. Use .values in
stead.
```

Out[12]:

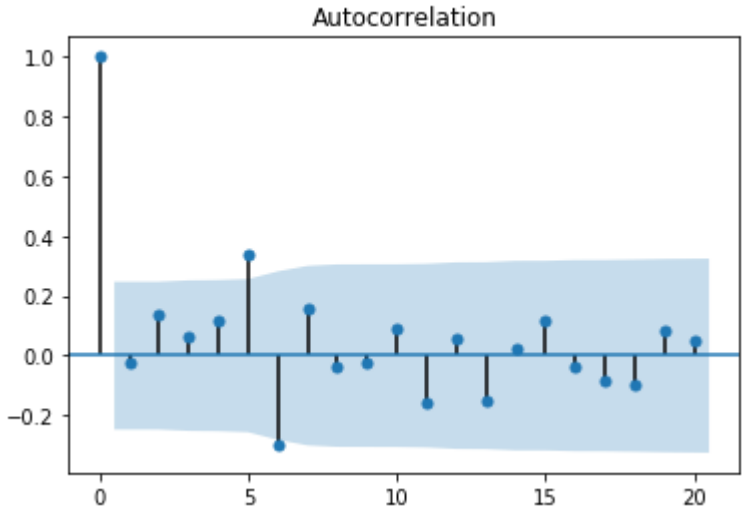
Model:	ARIMA	BIC:	898.5087
Dependent Variable:	D.y	Log-Likelihood:	-443.04
Date:	2020-04-19 21:53	Scale:	1.0000
No. Observations:	63	Method:	css-mle
Df Model:	2	Sample:	1
Df Residuals:	61		4
Converged:	1.0000	S.D. of innovations:	274.036
No. Iterations:	9.0000	HQIC:	894.608
AIC:	892.0793		

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	-58.0516	29.4547	-1.9709	0.0533	-115.7818	-0.3214
ma.L1.D.y	-0.1495	0.1084	-1.3799	0.1727	-0.3619	0.0629

	Real	Imaginary	Modulus	Frequency
MA.1	6.6876	0.0000	6.6876	0.0000

In [13]:

```
acf_plot = plot_acf( ar_model.resid,
lags=20)
```



GRAPH 6

In [14]:

```
predict,stderr,ci=ar_model.forecast(steps=15)
predict
```

Out[14]:

```
array([8490.35678718, 8432.30516718, 8374.25354719, 8316.2019272 ,
       8258.15030721, 8200.09868721, 8142.04706722, 8083.99544723,
       8025.94382724, 7967.89220724, 7909.84058725, 7851.78896726,
       7793.73734727, 7735.68572727, 7677.63410728])
```

In [15]:

```
#PREDICTION OF 3 MONTHS
predict_2,stderr,ci=ar_model.forecast(steps=7)
predict_2
```

Out[15]:

```
array([8490.35678718, 8432.30516718, 8374.25354719, 8316.2019272 ,
       8258.15030721, 8200.09868721, 8142.04706722])
```

In [16]:

```
#PREDICTION OF 3 MONTHS
predict_2,stderr,ci=ar_model.forecast(steps=57)
predict_2
```

Out[16]:

```
array([8490.35678718, 8432.30516718, 8374.25354719, 8316.2019272 ,
       8258.15030721, 8200.09868721, 8142.04706722, 8083.99544723,
       8025.94382724, 7967.89220724, 7909.84058725, 7851.78896726,
       7793.73734727, 7735.68572727, 7677.63410728, 7619.58248729,
       7561.5308673 , 7503.4792473 , 7445.42762731, 7387.37600732,
       7329.32438733, 7271.27276734, 7213.22114734, 7155.16952735,
       7097.11790736, 7039.06628737, 6981.01466737, 6922.96304738,
       6864.91142739, 6806.8598074 , 6748.8081874 , 6690.75656741,
       6632.70494742, 6574.65332743, 6516.60170743, 6458.55008744,
       6400.49846745, 6342.44684746, 6284.39522746, 6226.34360747,
       6168.29198748, 6110.24036749, 6052.18874749, 5994.1371275 ,
       5936.08550751, 5878.03388752, 5819.98226753, 5761.93064753,
       5703.87902754, 5645.82740755, 5587.77578756, 5529.72416756,
       5471.67254757, 5413.62092758, 5355.56930759, 5297.51768759,
       5239.4660676 ])
```

In [17]:

```
predict_2=pd.DataFrame(predict_2)
predict_2.columns
```

Out[17]:

```
RangeIndex(start=0, stop=1, step=1)
```

In [18]:

```
predict_2.columns=['FORECAST']
```

In [19]:

```
predict_2.head()
```

Out[19]:

	FORECAST
0	8490.356787
1	8432.305167
2	8374.253547
3	8316.201927
4	8258.150307

In [20]:

```
dataset_2.head()
```

Out[20]:

	Date	Open	High	Low	Close	Shares Traded	Turnover (Rs. Cr)	PREDICTION
0	01-Apr-20	8584.10	8588.10	8198.35	8253.80	507509685	21307.30	8490.356787
1	03-Apr-20	8356.55	8356.55	8055.80	8083.80	697752438	26561.18	8432.305167
2	07-Apr-20	8446.30	8819.40	8360.95	8792.20	815245226	35206.69	8374.253547
3	08-Apr-20	8688.90	9131.70	8653.90	8748.75	897562690	37641.68	8316.201927
4	09-Apr-20	8973.05	9128.35	8904.55	9111.90	743036134	32907.82	8258.150307

TABLE 3

In [21]:

```
dataset_2.shape
```

Out[21]:

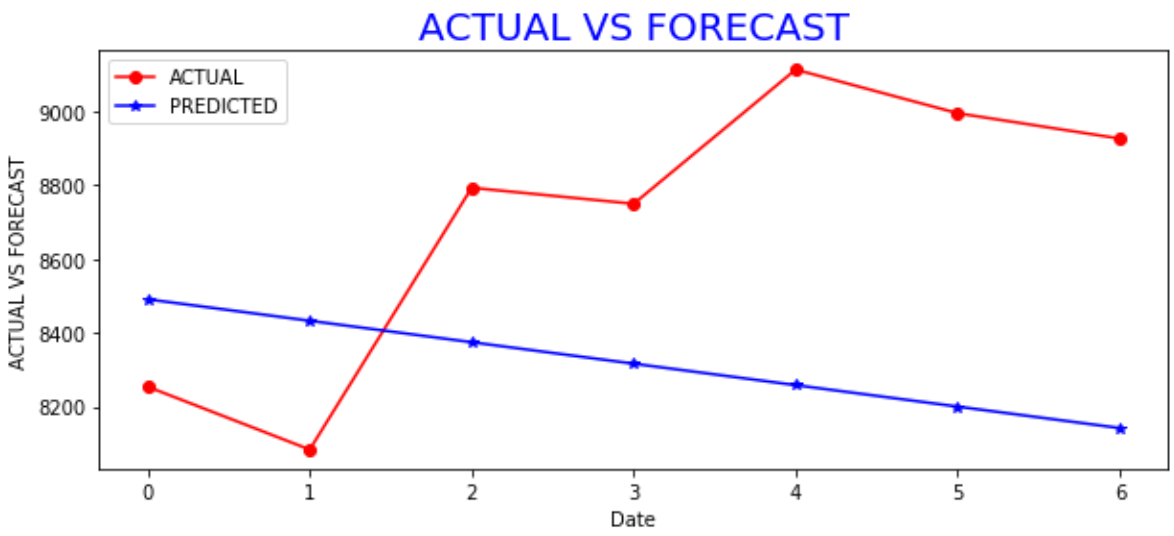
```
(7, 8)
```

In [22]:

```
plt.figure(figsize=(10,4))
plt.title('ACTUAL VS FORECAST',color='blue',size=20)
plt.xlabel('Date')
plt.ylabel('ACTUAL VS FORECAST')
plt.plot(dataset_2['Close'],marker='o',color='red',label='ACTUAL')
plt.plot(dataset_2['PREDICTION'],marker='*',color='BLUE',label='PREDICTED')
plt.legend()
```

Out[22]:

```
<matplotlib.legend.Legend at 0x19799e7cf08>
```



GRAPH 7

In [23]

```
def get_mape(actual, predicted):
 y_true, y_pred = np.array(actual), np.array(predicted)
 return np.round( np.mean(np.abs((actual - predicted) / actual)) * 100, 2 )
print('MEAN_ABSOLUTE_PERCENTAGE_ERROR:',get_mape( dataset_2['PREDICTION'],dataset_2['Close'
```

```
MEAN_ABSOLUTE_PERCENTAGE_ERROR: 6.68
```

In [24]:

```
from sklearn.metrics import mean_squared_error

print('ROOT_MEAN_SQUARE_ERROR:',np.sqrt(mean_squared_error( dataset_2['PREDICTION'],dataset
```

```
ROOT_MEAN_SQUARE_ERROR: 598.9999857110273
```

In [32]:

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

In [33]:

```
dataset=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//nya.csv')
dataset_2=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//nya april data.csv')
```

In [34]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 7 columns):
Date         62 non-null object
Open         62 non-null float64
High         62 non-null float64
Low          62 non-null float64
Close       62  non-null  float64
Adj  Close  62  non-null  float64
Volume 62 non-null int64
dtypes: float64(5), int64(1), object(1)
memory usage: 3.5+ KB
```

In [35]:

```
dataset.head()
```

Out[35]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	02-01-2020	13913.03027	14003.38965	13913.03027	14002.49023	14002.49023	3458250000
1	03-01-2020	13877.48047	13950.74023	13870.74023	13917.04981	13917.04981	3461290000
2	06-01-2020	13856.71973	13943.29981	13852.73047	13941.79981	13941.79981	3674070000
3	07-01-2020	13911.19043	13923.50977	13880.53027	13898.45020	13898.45020	3420380000
4	08-01-2020	13897.00977	13986.69043	13896.58984	13934.44043	13934.44043	3720890000

TABLE 4

In [36]:

```
type(dataset)
```

Out[36]:

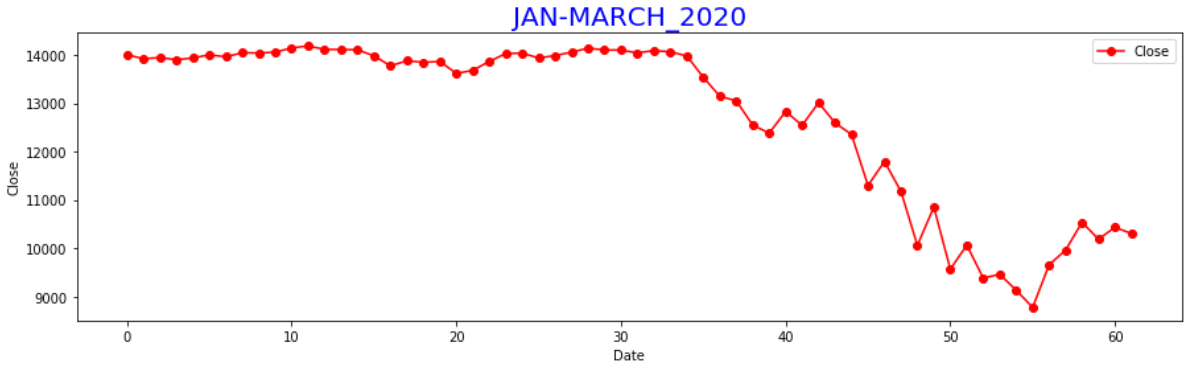
```
pandas.core.frame.DataFrame
```

In [37]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Close')
plt.plot(dataset['Close'],color='red',marker='o',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[37]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



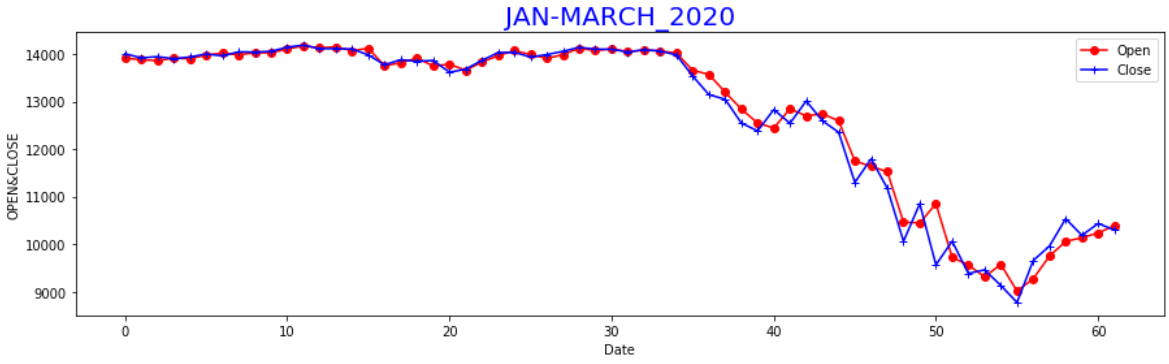
GRAPH 8

In [38]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('OPEN&CLOSE')
plt.plot(dataset['Open'],color='red',marker='o',label='Open')
plt.plot(dataset['Close'],color='blue',marker='+',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[38]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



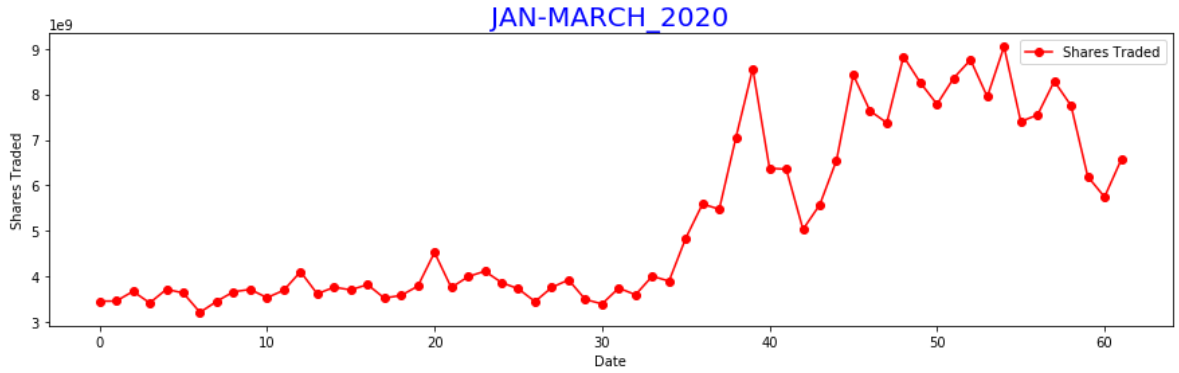
GRAPH 10

In [39]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Shares Traded')
plt.plot(dataset['Volume'],color='red',marker='o',label='Shares Traded')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[39]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```

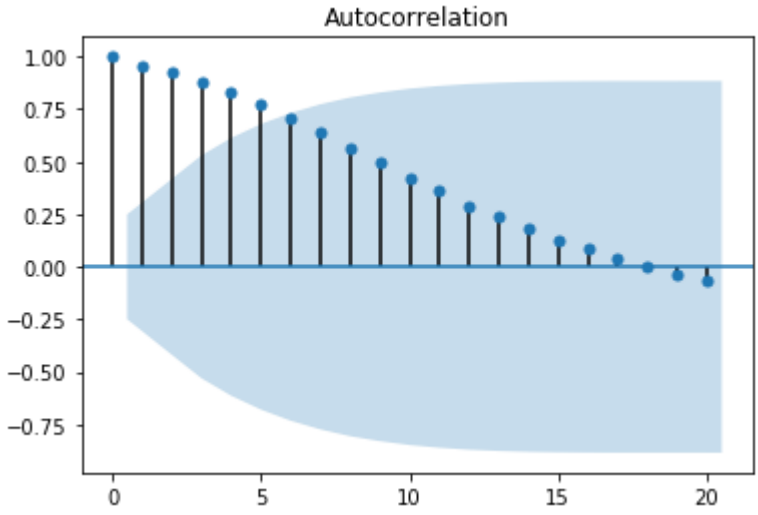


GRAPH 9

In [40]:

```
plt.figure(figsize=(15,4))
#CHECKING STATIONARY OF THE DATA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
# Show autocorrelation upto lag 10
acf_plot = plot_acf( dataset.Close,
lags=20)
```

```
<Figure size 1080x288 with 0 Axes>
```



GRAPH 11

In [41]:

```
dataset['Close_diff'] = dataset.Close - dataset.Close.shift(1)
dataset_diff=dataset.dropna()
dataset_diff.head()
```

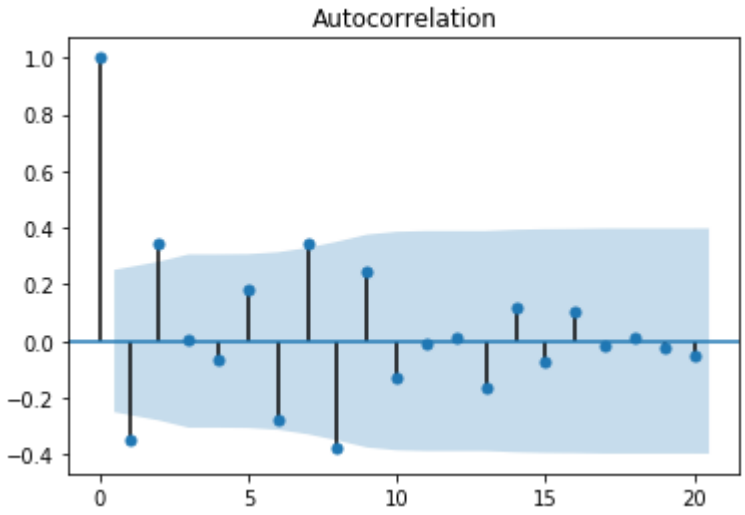
Out[41]:

	Date	Open	High	Low	Close	Adj Close	Volume	Close_
1	03-01-2020	13877.48047	13950.74023	13870.74023	13917.04981	13917.04981	3461290000	-85.44
2	06-01-2020	13856.71973	13943.29981	13852.73047	13941.79981	13941.79981	3674070000	24.75
3	07-01-2020	13911.19043	13923.50977	13880.53027	13898.45020	13898.45020	3420380000	-43.34
4	08-01-2020	13897.00977	13986.69043	13896.58984	13934.44043	13934.44043	3720890000	35.99
5	09-01-2020	13974.07031	14004.41992	13960.16992	13997.65039	13997.65039	3638390000	63.20

TABLE 5

In [42]:

```
acf_plot = plot_acf( dataset_diff.Close_diff,
lags=20)
```



GRAPH 12

In [43]:

```
from statsmodels.tsa.arima_model import ARIMA
arima = ARIMA( dataset.Close.astype(np.float64).as_matrix(),
order = (0,1,1))
ar_model = arima.fit()
ar_model.summary2()
```

```
C:\Users\deepa\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWa
rning: Method .as_matrix will be removed in a future version. Use .values in
stead.
```

Out[43]:

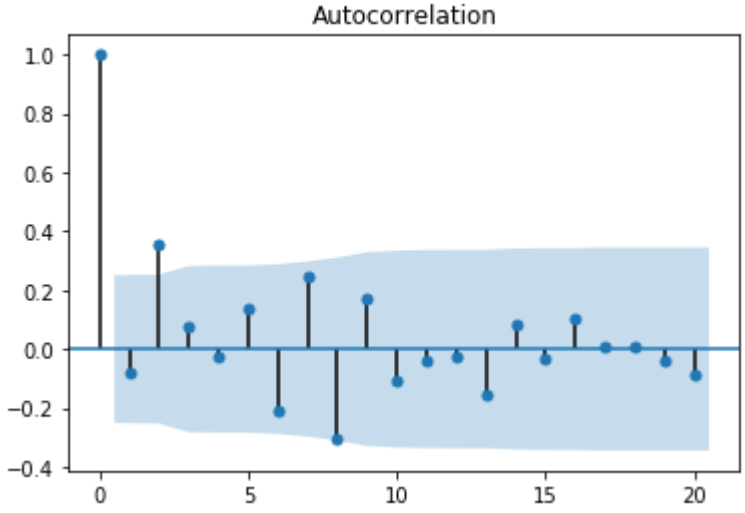
Model:	ARIMA	BIC:	906.8112
Dependent Variable:	D.y	Log-Likelihood:	-447.24
Date:	2020-04-19 22:33	Scale:	1.0000
No. Observations:	61	Method:	css-mle
Df Model:	2	Sample:	1
Df Residuals:	59		2
Converged:	1.0000	S.D. of innovations:	369.616
No. Iterations:	7.0000	HQIC:	902.960
AIC:	900.4785		

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	-60.6103	37.2997	-1.6250	0.1095	-133.7164	12.4959
ma.L1.D.y	-0.2154	0.0944	-2.2817	0.0261	-0.4004	-0.0304

	Real	Imaginary	Modulus	Frequency
MA.1	4.6428	0.0000	4.6428	0.0000

In [44]:

```
acf_plot = plot_acf( ar_model.resid,
lags=20)
```



GRAPH 13

In [45]:

```
predict,stderr,ci=ar_model.forecast(steps=12)
predict
```

Out[45]:

```
array([10243.7881212 , 10183.17786896, 10122.56761672, 10061.95736448,
       10001.34711223,  9940.73685999,  9880.12660775,  9819.51635551,
        9758.90610327,  9698.29585102,  9637.68559878,  9577.07534654])
```

In [46]:

```
#PREDICTION OF 3 MONTHS
predict_2,stderr,ci=ar_model.forecast(steps=7)
predict_2
```

Out[46]:

```
array([10243.7881212 , 10183.17786896, 10122.56761672, 10061.95736448,
       10001.34711223,  9940.73685999,  9880.12660775])
```

In [47]:

```
#PREDICTION OF 3 MONTHS
predict_2,stderr,ci=ar_model.forecast(steps=57)
predict_2
```

Out[47]:

```
array([10243.7881212 , 10183.17786896, 10122.56761672, 10061.95736448,
       10001.34711223,  9940.73685999,  9880.12660775,  9819.51635551,
        9758.90610327,  9698.29585102,  9637.68559878,  9577.07534654,
        9516.4650943 ,  9455.85484206,  9395.24458981,  9334.63433757,
        9274.02408533,  9213.41383309,  9152.80358085,  9092.1933286 ,
        9031.58307636,  8970.97282412,  8910.36257188,  8849.75231964,
        8789.14206739,  8728.53181515,  8667.92156291,  8607.31131067,
        8546.70105843,  8486.09080618,  8425.48055394,  8364.8703017 ,
        8304.26004946,  8243.64979722,  8183.03954497,  8122.42929273,
        8061.81904049,  8001.20878825,  7940.59853601,  7879.98828376,
        7819.37803152,  7758.76777928,  7698.15752704,  7637.5472748 ,
        7576.93702255,  7516.32677031,  7455.71651807,  7395.10626583,
        7334.49601359,  7273.88576134,  7213.2755091 ,  7152.66525686,
        7092.05500462,  7031.44475238,  6970.83450013,  6910.22424789,
        6849.61399565])
```

In [48]:

```
predict_2=pd.DataFrame(predict_2)
predict_2.columns
```

Out[48]:

```
RangeIndex(start=0, stop=1, step=1)
```

In [49]:

```
predict_2.columns=['FORECAST']
```

In [50]:

```
predict_2.head()
```

Out[50]:

	FORECAST
0	10243.788121
1	10183.177869
2	10122.567617
3	10061.957364
4	10001.347112

In [51]:

```
dataset_2.head()
```

Out[51]:

	Date	Open	High	Low	Close	Adj Close	Volume	PR
0	01-04-2020	9917.309570	10029.00000	9766.799805	9844.849609	9844.849609	5947900000	10
1	02-04-2020	9815.269531	10142.00977	9813.509766	10062.370120	10062.370120	6454990000	10
2	03-04-2020	10012.469730	10075.91016	9774.240234	9880.629883	9880.629883	6087190000	10
3	06-04-2020	10264.129880	10560.73047	10233.040040	10515.240230	10515.240230	6391860000	10
4	07-04-2020	10894.200200	10912.55957	10534.740230	10537.040040	10537.040040	7040720000	10

TABLE 6

In [52]:

```
dataset_2.shape
```

Out[52]:

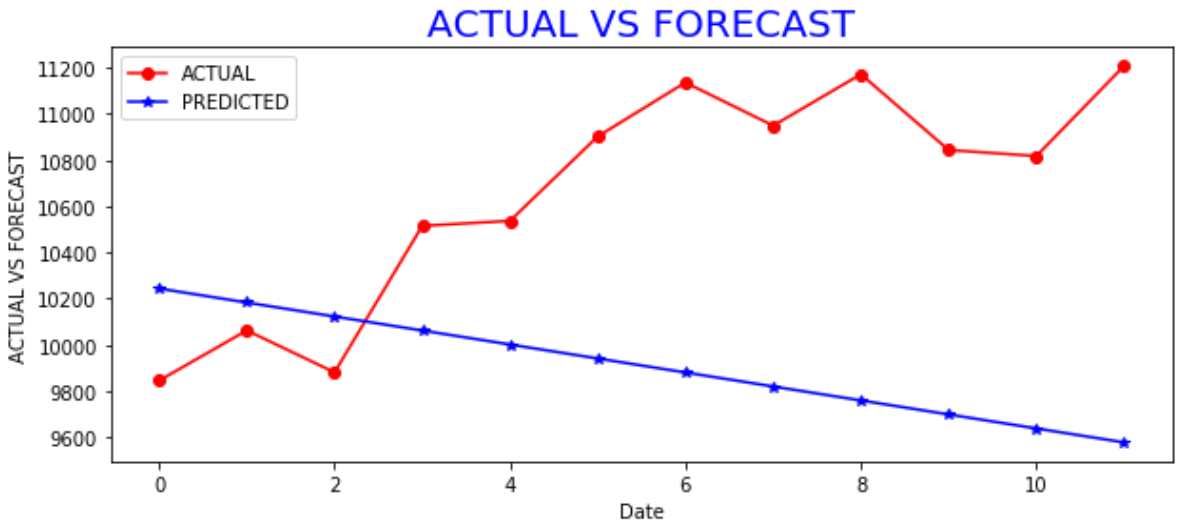
```
(12, 8)
```

In [54]:

```
plt.figure(figsize=(10,4))
plt.title('ACTUAL VS FORECAST',color='blue',size=20)
plt.xlabel('Date')
plt.ylabel('ACTUAL VS FORECAST')
plt.plot(dataset_2['Close'],marker='o',color='red',label='ACTUAL')
plt.plot(dataset_2['PREDICTED'],marker='*',color='BLUE',label='PREDICTED')
plt.legend()
```

Out[54]:

```
<matplotlib.legend.Legend at 0x276cbec3bc8>
```



GRAPH 14

In [56]:

```
def get_mape(actual, predicted):
 y_true, y_pred = np.array(actual), np.array(predicted)
 return np.round( np.mean(np.abs((actual - predicted) / actual)) * 100, 2 )
print('MEAN_ABSOLUTE_PERCENTAGE_ERROR:',get_mape( dataset_2['PREDICTED'],dataset_2['Close'])
```

```
MEAN_ABSOLUTE_PERCENTAGE_ERROR: 8.9
```

In [57]:

```
from sklearn.metrics import mean_squared_error

print('ROOT_MEAN_SQUARE_ERROR:',np.sqrt(mean_squared_error( dataset_2['PREDICTED'],dataset_
```

```
ROOT_MEAN_SQUARE_ERROR: 994.5470583836233
```

In [16]:

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

In [17]:

```
dataset=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//korea.csv')
dataset_2=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//korea_april_data.csv')
```

In [18]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 7 columns):
Date         62 non-null object
Open         62 non-null float64
High         62 non-null float64
Low          62 non-null float64
Close       62  non-null  float64
Adj  Close  62  non-null  float64
Volume 62 non-null int64
dtypes: float64(5), int64(1), object(1)
memory usage: 3.5+ KB
```

In [19]:

```
dataset.head()
```

Out[19]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	02-01-2020	2201.209961	2202.320068	2171.840088	2175.169922	2175.169922	494700
1	03-01-2020	2192.580078	2203.379883	2165.389893	2176.459961	2176.459961	631600
2	06-01-2020	2154.969971	2164.419922	2149.949951	2155.070068	2155.070068	592700
3	07-01-2020	2166.600098	2181.620117	2164.270020	2175.540039	2175.540039	568200
4	08-01-2020	2156.270020	2162.320068	2137.719971	2151.310059	2151.310059	913800

TABLE 7

In [20]:

```
type(dataset)
```

Out[20]:

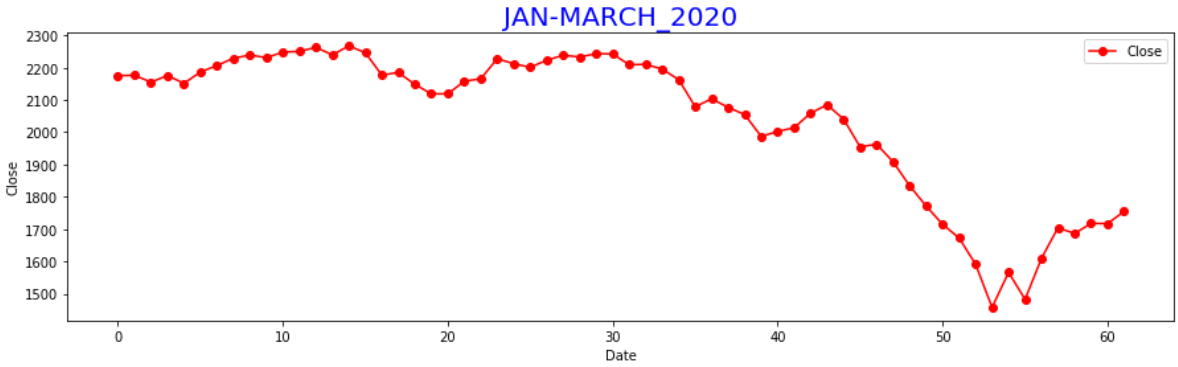
```
pandas.core.frame.DataFrame
```

In [21]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Close')
plt.plot(dataset['Close'],color='red',marker='o',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[21]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



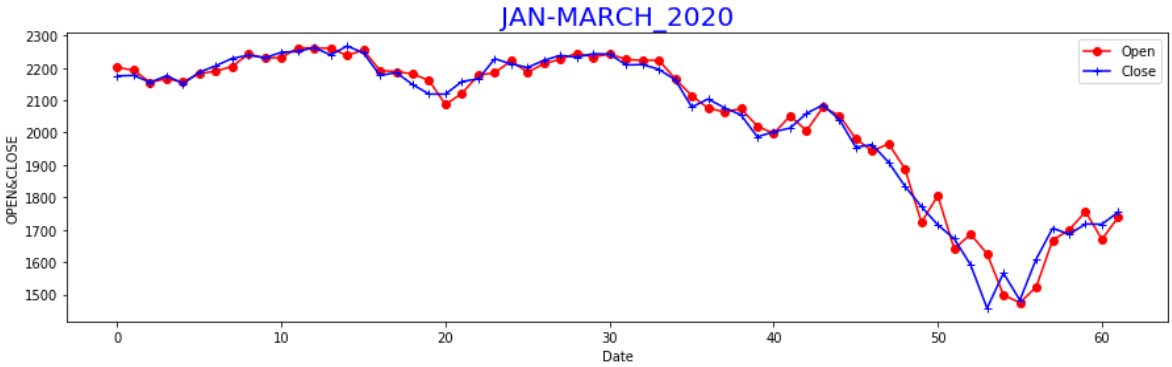
GRAPH 15

In [22]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('OPEN&CLOSE')
plt.plot(dataset['Open'],color='red',marker='o',label='Open')
plt.plot(dataset['Close'],color='blue',marker='+',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[22]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



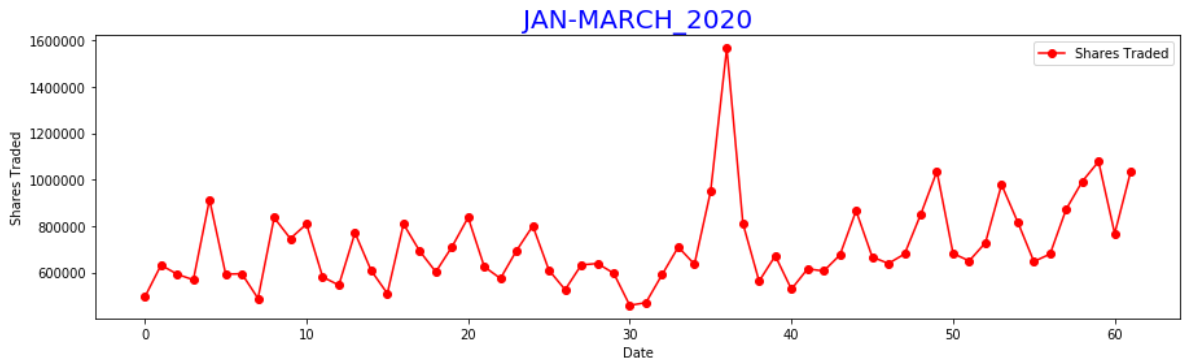
GRAPH 16

In [23]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Shares Traded')
plt.plot(dataset['Volume'],color='red',marker='o',label='Shares Traded')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[23]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```

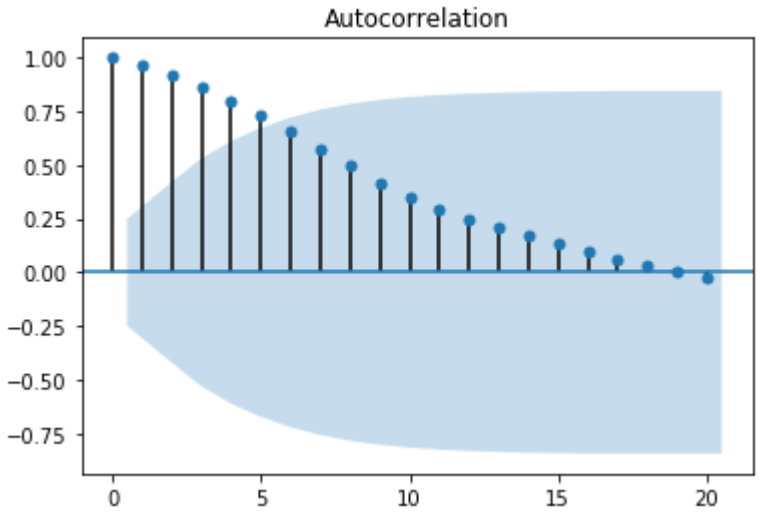


GRAPH 17

In [24]:

```
plt.figure(figsize=(15,4))
#CHECKING STATIONARY OF THE DATA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
# Show autocorrelation upto lag 10
acf_plot = plot_acf( dataset.Close,
lags=20)
```

```
<Figure size 1080x288 with 0 Axes>
```



GRAPH 18

In [25]:

```
dataset['Close_diff'] = dataset.Close - dataset.Close.shift(1)
dataset_diff=dataset.dropna()
dataset_diff.head()
```

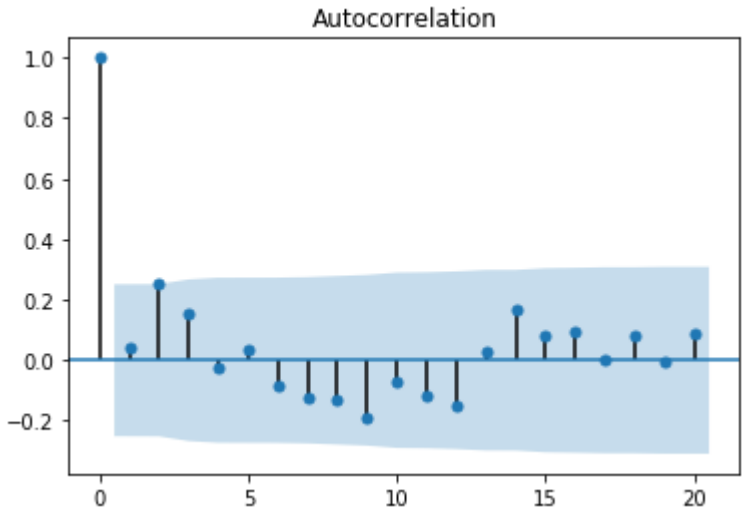
Out[25]:

	Date	Open	High	Low	Close	Adj Close	Volume	Close_diff
1	03-01-2020	2192.580078	2203.379883	2165.389893	2176.459961	2176.459961	631600	1.290039
2	06-01-2020	2154.969971	2164.419922	2149.949951	2155.070068	2155.070068	592700	-21.389893
3	07-01-2020	2166.600098	2181.620117	2164.270020	2175.540039	2175.540039	568200	20.469971
4	08-01-2020	2156.270020	2162.320068	2137.719971	2151.310059	2151.310059	913800	-24.229980
5	09-01-2020	2182.199951	2186.449951	2172.159912	2186.449951	2186.449951	592600	35.139892

TABLE 8

In [26]:

```
acf_plot = plot_acf( dataset_diff.Close_diff,
lags=20)
```



GRAPH 19

In [27]:

```
from statsmodels.tsa.arima_model import ARIMA
arima = ARIMA( dataset.Close.astype(np.float64).as_matrix(),
order = (0,1,1))
ar_model = arima.fit()
ar_model.summary2()
```

```
C:\Users\deepa\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWa
rning: Method .as_matrix will be removed in a future version. Use .values in
stead.
```

Out[27]:

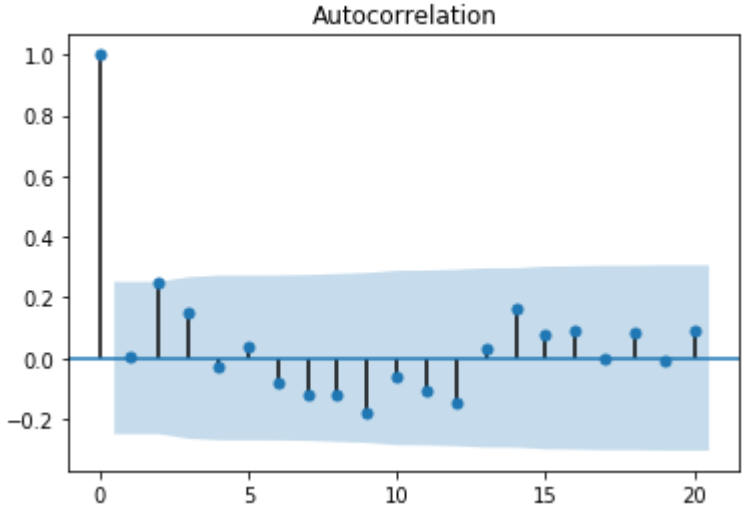
Model:	ARIMA	BIC:	654.6621
Dependent Variable:	D.y	Log-Likelihood:	-321.16
Date:	2020-04-19 23:01	Scale:	1.0000
No. Observations:	61	Method:	css-mle
Df Model:	2	Sample:	1
Df Residuals:	59		2
Converged:	1.0000	S.D. of innovations:	46.808
No. Iterations:	5.0000	HQIC:	650.811
AIC:	648.3295		

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	-6.8695	6.1584	-1.1155	0.2692	-18.9397	5.2008
ma.L1.D.y	0.0279	0.1051	0.2655	0.7915	-0.1782	0.2340

	Real	Imaginary	Modulus	Frequency
MA.1	-35.8153	0.0000	35.8153	0.5000

In [28]:

```
acf_plot = plot_acf( ar_model.resid,
lags=20)
```



GRAPH 19

In [29]:

```
predict,stderr,ci=ar_model.forecast(steps=12)
predict
```

Out[29]:

```
array([1749.00592304, 1742.13646663, 1735.26701022, 1728.3975538 ,
       1721.52809739, 1714.65864098, 1707.78918456, 1700.91972815,
       1694.05027174, 1687.18081532, 1680.31135891, 1673.44190249])
```

In [30]:

```
#PREDICTION OF 3 MONTHS
predict_2,stderr,ci=ar_model.forecast(steps=57)
predict_2
```

Out[30]:

```
array([1749.00592304, 1742.13646663, 1735.26701022, 1728.3975538 ,
       1721.52809739, 1714.65864098, 1707.78918456, 1700.91972815,
       1694.05027174, 1687.18081532, 1680.31135891, 1673.44190249,
       1666.57244608, 1659.70298967, 1652.83353325, 1645.96407684,
       1639.09462043, 1632.22516401, 1625.3557076 , 1618.48625119,
       1611.61679477, 1604.74733836, 1597.87788195, 1591.00842553,
       1584.13896912, 1577.26951271, 1570.40005629, 1563.53059988,
       1556.66114347, 1549.79168705, 1542.92223064, 1536.05277423,
       1529.18331781, 1522.3138614 , 1515.44440499, 1508.57494857,
       1501.70549216, 1494.83603574, 1487.96657933, 1481.09712292,
       1474.2276665 , 1467.35821009, 1460.48875368, 1453.61929726,
       1446.74984085, 1439.88038444, 1433.01092802, 1426.14147161,
       1419.2720152 , 1412.40255878, 1405.53310237, 1398.66364596,
       1391.79418954, 1384.92473313, 1378.05527672, 1371.1858203 ,
       1364.31636389])
```

In [31]:

```
predict_2=pd.DataFrame(predict_2)
predict_2.columns
```

Out[31]:

```
RangeIndex(start=0, stop=1, step=1)
```

In [32]:

```
predict_2.columns=['FORECAST']
```

In [33]:

```
predict_2.head()
```

Out[33]:

	FORECAST
0	1749.005923
1	1742.136467
2	1735.267010
3	1728.397554
4	1721.528097

In [34]:

```
dataset_2.head()
```

Out[34]:

	Date	Open	High	Low	Close	Adj Close	Volume	PREDICTE
0	01-04-2020	1737.280029	1762.439941	1685.369995	1685.459961	1685.459961	1243600	1749.0059
1	02-04-2020	1693.530029	1726.760010	1664.130005	1724.859985	1724.859985	766300	1742.1364
2	03-04-2020	1731.170044	1743.910034	1706.680054	1725.439941	1725.439941	1000100	1735.2670
3	06-04-2020	1745.250000	1794.189941	1742.670044	1791.880005	1791.880005	1219400	1728.3975
4	07-04-2020	1826.719971	1838.969971	1791.920044	1823.599976	1823.599976	951400	1721.5280

TABLE 9

In [35]:

```
dataset_2.shape
```

Out[35]:

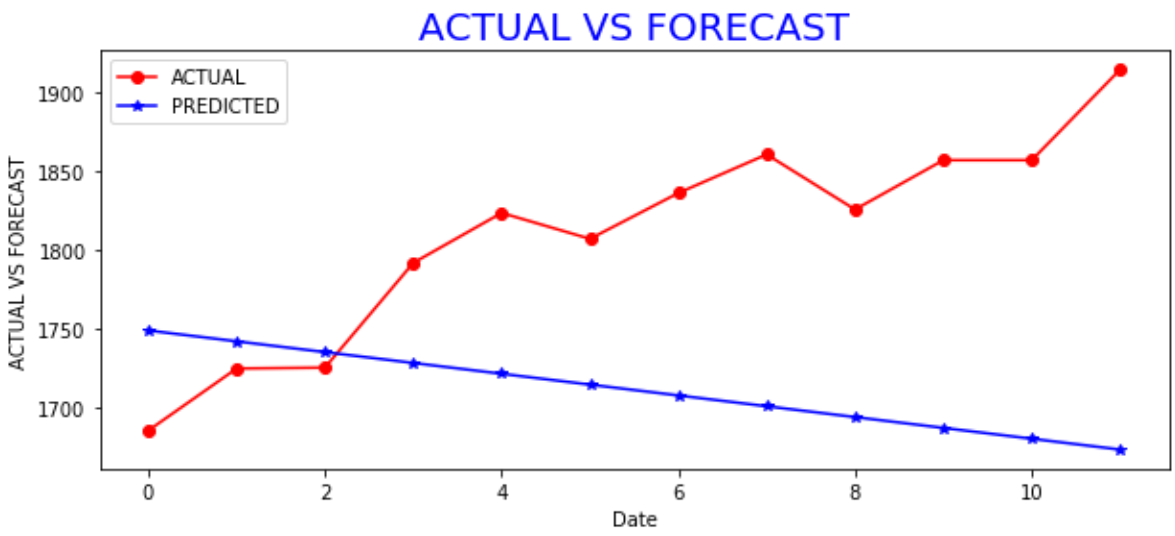
```
(12, 8)
```

In [36]:

```
plt.figure(figsize=(10,4))
plt.title('ACTUAL VS FORECAST',color='blue',size=20)
plt.xlabel('Date')
plt.ylabel('ACTUAL VS FORECAST')
plt.plot(dataset_2['Close'],marker='o',color='red',label='ACTUAL')
plt.plot(dataset_2['PREDICTED'],marker='*',color='BLUE',label='PREDICTED')
plt.legend()
```

Out[36]:

```
<matplotlib.legend.Legend at 0x215052d8548>
```



GRAPH 20

In [37]:

```
def get_mape(actual, predicted):
 y_true, y_pred = np.array(actual), np.array(predicted)
 return np.round( np.mean(np.abs((actual - predicted) / actual)) * 100, 2 )
print('MEAN_ABSOLUTE_PERCENTAGE_ERROR:',get_mape( dataset_2['PREDICTED'],dataset_2['Close']
```

```
MEAN_ABSOLUTE_PERCENTAGE_ERROR: 6.66
```

In [38]:

```
from sklearn.metrics import mean_squared_error

print('ROOT_MEAN_SQUARE_ERROR:',np.sqrt(mean_squared_error( dataset_2['PREDICTED'],dataset_
```

```
ROOT_MEAN_SQUARE_ERROR: 130.71486259283398
```

In [15]:

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

In [16]:

```
dataset=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//lse.csv')
dataset_2=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//lse_april_data.csv')
```

In [17]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64 entries, 0 to 63
Data columns (total 7 columns):
Date         64 non-null object
Open         64 non-null int64
High         64 non-null float64
Low          64 non-null float64
Close       64  non-null  int64
Adj  Close  64  non-null  int64
Volume 64 non-null int64
dtypes: float64(2), int64(4), object(1)
memory usage: 3.6+ KB
```

In [18]:

```
dataset.head()
```

Out[18]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	02-01-2020	7746	7816.000000	7674.000000	7700	7700	448320
1	03-01-2020	7656	7658.000000	7580.000000	7638	7638	395569
2	06-01-2020	7576	7660.000000	7494.000000	7500	7500	434347
3	07-01-2020	7550	7559.799805	7424.000000	7440	7440	676227
4	08-01-2020	7410	7542.000000	7355.680176	7542	7542	524342

TABLE 10

In [19]:

```
type(dataset)
```

Out[19]:

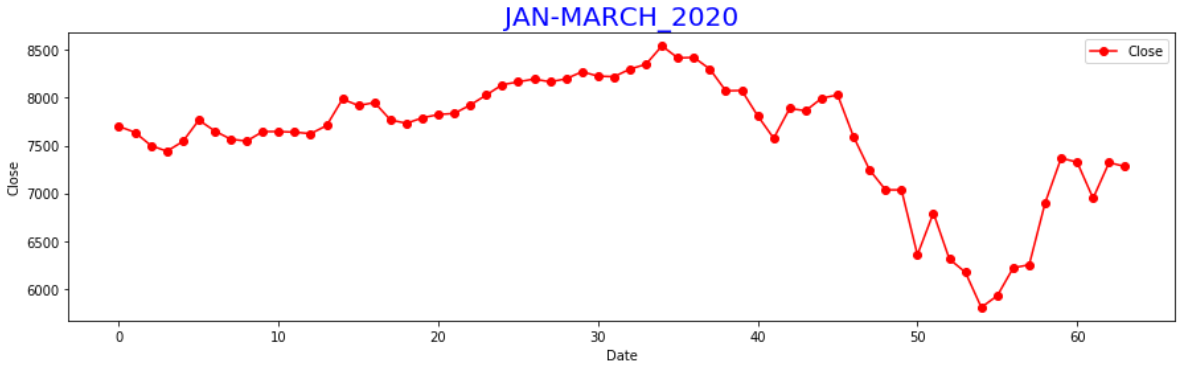
```
pandas.core.frame.DataFrame
```

In [20]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Close')
plt.plot(dataset['Close'],color='red',marker='o',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[20]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



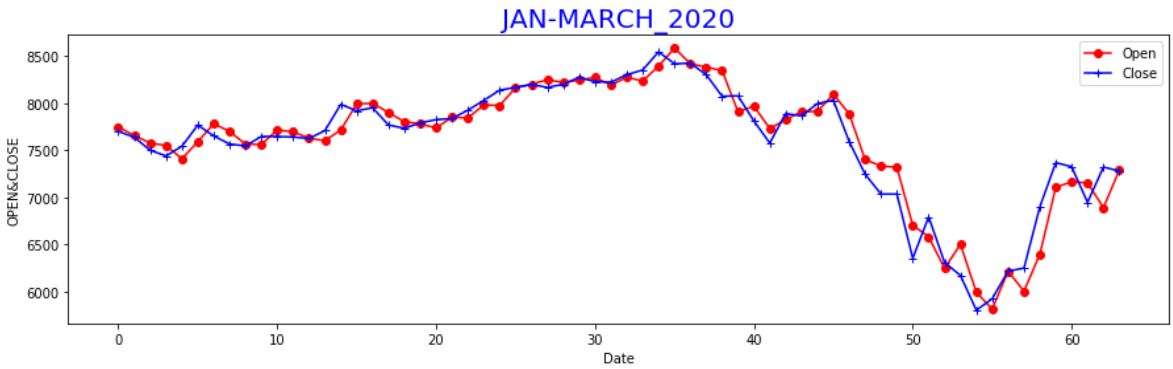
GRAPH 21

In [21]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('OPEN&CLOSE')
plt.plot(dataset['Open'],color='red',marker='o',label='Open')
plt.plot(dataset['Close'],color='blue',marker='+',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[21]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



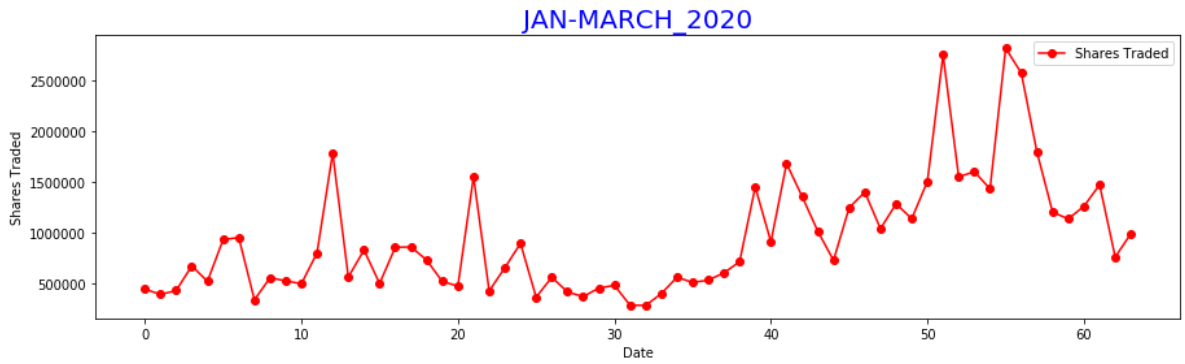
GRAPH 22

In [22]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Shares Traded')
plt.plot(dataset['Volume'],color='red',marker='o',label='Shares Traded')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[22]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```

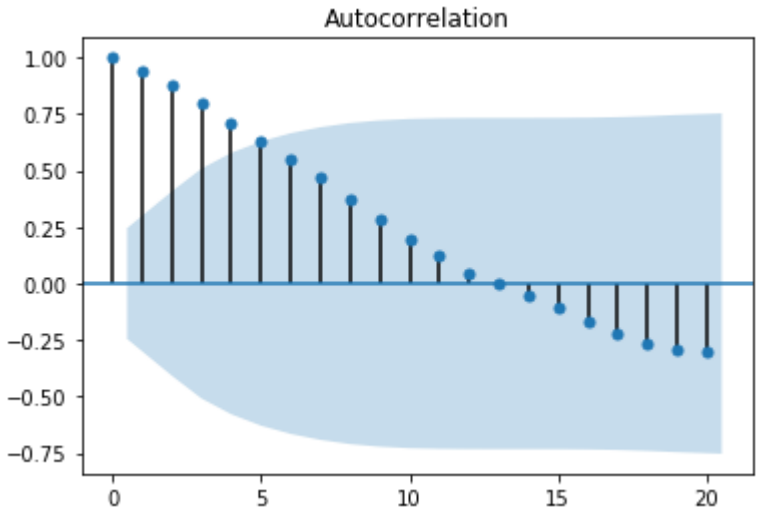


GRAPH 23

In [23]:

```
plt.figure(figsize=(15,4))
#CHECKING STATIONARY OF THE DATA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
# Show autocorrelation upto lag 10
acf_plot = plot_acf( dataset.Close,
lags=20)
```

```
<Figure size 1080x288 with 0 Axes>
```



GRAPH 24

In [24]:

```
dataset['Close_diff'] = dataset.Close - dataset.Close.shift(1)
dataset_diff=dataset.dropna()
dataset_diff.head()
```

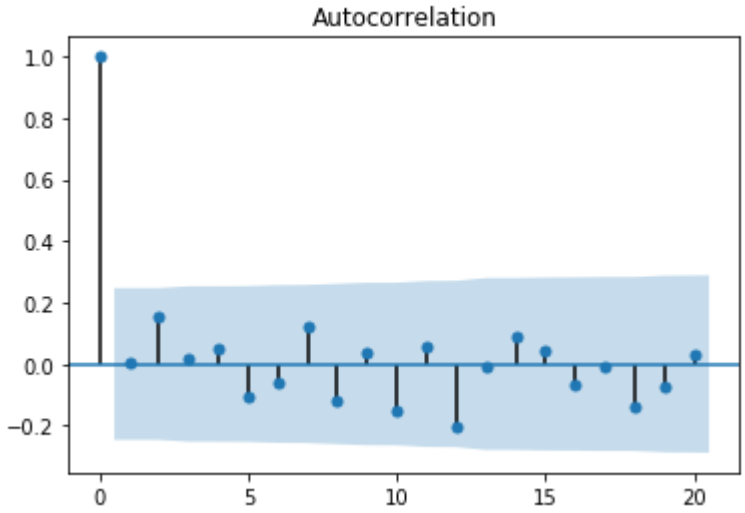
Out[24]:

	Date	Open	High	Low	Close	Adj Close	Volume	Close_diff
1	03-01-2020	7656	7658.000000	7580.000000	7638	7638	395569	-62.0
2	06-01-2020	7576	7660.000000	7494.000000	7500	7500	434347	-138.0
3	07-01-2020	7550	7559.799805	7424.000000	7440	7440	676227	-60.0
4	08-01-2020	7410	7542.000000	7355.680176	7542	7542	524342	102.0
5	09-01-2020	7590	7778.000000	7578.520020	7766	7766	933950	224.0

TABLE 11

In [25]:

```
acf_plot = plot_acf( dataset_diff.Close_diff,
lags=20)
```



GRAPH 25

In [26]:

```
from statsmodels.tsa.arima_model import ARIMA
arima = ARIMA( dataset.Close.astype(np.float64).as_matrix(),
order = (0,1,1))
ar_model = arima.fit()
ar_model.summary2()
```

```
C:\Users\deepa\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWa
rning: Method .as_matrix will be removed in a future version. Use .values in
stead.
```

Out[26]:

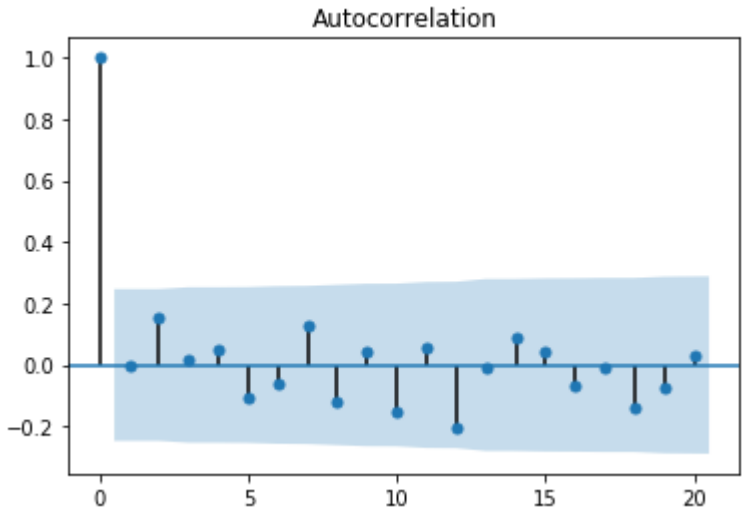
Model:	ARIMA	BIC:	871.4445
Dependent Variable:	D.y	Log-Likelihood:	-429.51
Date:	2020-04-19 23:18	Scale:	1.0000
No. Observations:	63	Method:	css-mle
Df Model:	2	Sample:	1
Df Residuals:	61		4
Converged:	1.0000	S.D. of innovations:	221.106
No. Iterations:	3.0000	HQIC:	867.544
AIC:	865.0151		

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	-6.6667	27.9697	-0.2384	0.8124	-61.4862	48.1529
ma.L1.D.y	0.0041	0.1094	0.0375	0.9702	-0.2103	0.2185

	Real	Imaginary	Modulus	Frequency
MA.1	-243.7004	0.0000	243.7004	0.5000

In [27]:

```
acf_plot = plot_acf( ar_model.resid,
lags=20)
```



GRAPH 26

In [28]:

```
predict,stderr,ci=ar_model.forecast(steps=12)
predict
```

Out[28]:

```
array([7273.1819086 , 7266.51523922, 7259.84856984, 7253.18190045,
       7246.51523107, 7239.84856168, 7233.1818923 , 7226.51522292,
       7219.84855353, 7213.18188415, 7206.51521476, 7199.84854538])
```

In [29]:

```
#PREDICTION OF 3 MONTHS
predict_2,stderr,ci=ar_model.forecast(steps=57)
predict_2
```

Out[29]:

```
array([7273.1819086 , 7266.51523922, 7259.84856984, 7253.18190045,
       7246.51523107, 7239.84856168, 7233.1818923 , 7226.51522292,
       7219.84855353, 7213.18188415, 7206.51521476, 7199.84854538,
       7193.181876  , 7186.51520661, 7179.84853723, 7173.18186785,
       7166.51519846, 7159.84852908, 7153.18185969, 7146.51519031,
       7139.84852093, 7133.18185154, 7126.51518216, 7119.84851277,
       7113.18184339, 7106.51517401, 7099.84850462, 7093.18183524,
       7086.51516585, 7079.84849647, 7073.18182709, 7066.5151577 ,
       7059.84848832, 7053.18181893, 7046.51514955, 7039.84848017,
       7033.18181078, 7026.5151414 , 7019.84847201, 7013.18180263,
       7006.51513325, 6999.84846386, 6993.18179448, 6986.51512509,
       6979.84845571, 6973.18178633, 6966.51511694, 6959.84844756,
       6953.18177818, 6946.51510879, 6939.84843941, 6933.18177002,
       6926.51510064, 6919.84843126, 6913.18176187, 6906.51509249,
       6899.8484231 ])
```

In [30]:

```
predict_2=pd.DataFrame(predict_2)
predict_2.columns
```

Out[30]:

```
RangeIndex(start=0, stop=1, step=1)
```

In [31]:

```
predict_2.columns=['FORECAST']
```

In [32]:

```
predict_2.head()
```

Out[32]:

	FORECAST
0	7273.181909
1	7266.515239
2	7259.848570
3	7253.181900
4	7246.515231

In [33]:

```
dataset_2.head()
```

Out[33]:

	Date	Open	High	Low	Close	Adj Close	Volume	PREDICTED
0	01-04-2020	7044	7186.000000	6864.0	7070	7070	812461	7273.181909
1	02-04-2020	7118	7142.000000	6496.0	6736	6736	1281215	7266.515239
2	03-04-2020	6802	7222.000000	6702.0	7200	7200	705400	7259.848570
3	06-04-2020	7332	7533.299805	6804.0	7064	7064	1247490	7253.181900
4	07-04-2020	7274	7426.000000	7006.0	7180	7180	1015501	7246.515231

TABLE 12

In [34]:

```
dataset_2.shape
```

Out[34]:

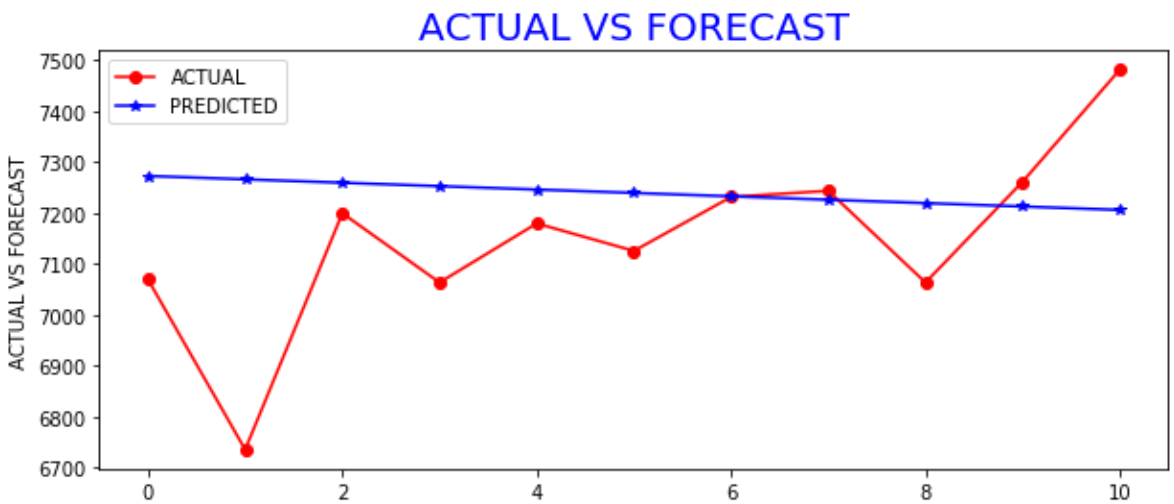
```
(11, 8)
```

In [35]:

```
plt.figure(figsize=(10,4))
plt.title('ACTUAL VS FORECAST',color='blue',size=20)
plt.xlabel('Date')
plt.ylabel('ACTUAL VS FORECAST')
plt.plot(dataset_2['Close'],marker='o',color='red',label='ACTUAL')
plt.plot(dataset_2['PREDICTED'],marker='*',color='BLUE',label='PREDICTED')
plt.legend()
```

Out[35]:

```
<matplotlib.legend.Legend at 0x22e10af35c8>
```



GRAPH 27

In [36]:

```
def get_mape(actual, predicted):
 y_true, y_pred = np.array(actual), np.array(predicted)
 return np.round( np.mean(np.abs((actual - predicted) / actual)) * 100, 2 )
print('MEAN_ABSOLUTE_PERCENTAGE_ERROR:',get_mape( dataset_2['PREDICTED'],dataset_2['Close']
```

```
MEAN_ABSOLUTE_PERCENTAGE_ERROR: 2.08
```

In [37]:

```
from sklearn.metrics import mean_squared_error

print('ROOT_MEAN_SQUARE_ERROR:',np.sqrt(mean_squared_error( dataset_2['PREDICTED'],dataset_
```

```
ROOT_MEAN_SQUARE_ERROR: 209.40558338801713
```

In [16]:

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

In [17]:

```
dataset=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//china.csv')
dataset_2=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//china_april_data.csv')
```

In [18]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58 entries, 0 to 57
Data columns (total 7 columns):
Date         58 non-null object
Open         58 non-null float64
High         58 non-null float64
Low          58 non-null float64
Close       58  non-null  float64
Adj  Close  58  non-null  float64
Volume 58 non-null int64
dtypes: float64(5), int64(1), object(1)
memory usage: 3.3+ KB
```

In [19]:

```
dataset.head()
```

Out[19]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	02-01-2020	3066.335938	3098.100098	3066.335938	3085.197998	3085.197998	292500
1	03-01-2020	3089.021973	3093.819092	3074.518066	3083.785889	3083.785889	261500
2	06-01-2020	3070.908936	3107.202881	3065.309082	3083.407959	3083.407959	312600
3	07-01-2020	3085.488037	3105.450928	3084.329102	3104.802002	3104.802002	276600
4	08-01-2020	3094.239014	3094.239014	3059.131104	3066.893066	3066.893066	297900

TABLE 13

In [20]:

```
type(dataset)
```

Out[20]:

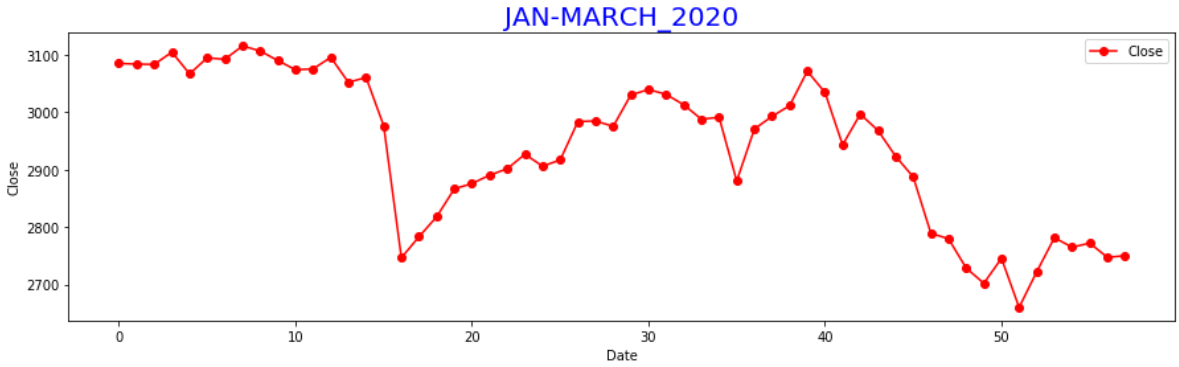
```
pandas.core.frame.DataFrame
```

In [21]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Close')
plt.plot(dataset['Close'],color='red',marker='o',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[21]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



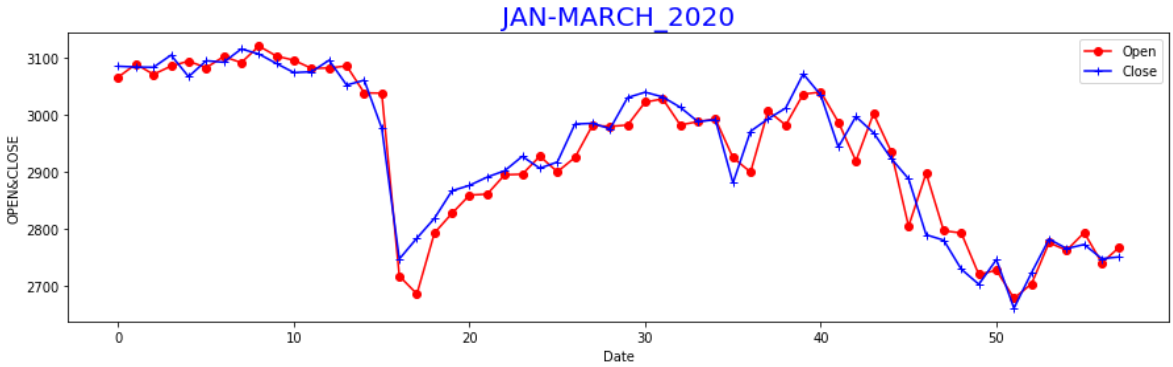
GRAPH 28

In [22]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('OPEN&CLOSE')
plt.plot(dataset['Open'],color='red',marker='o',label='Open')
plt.plot(dataset['Close'],color='blue',marker='+',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[22]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



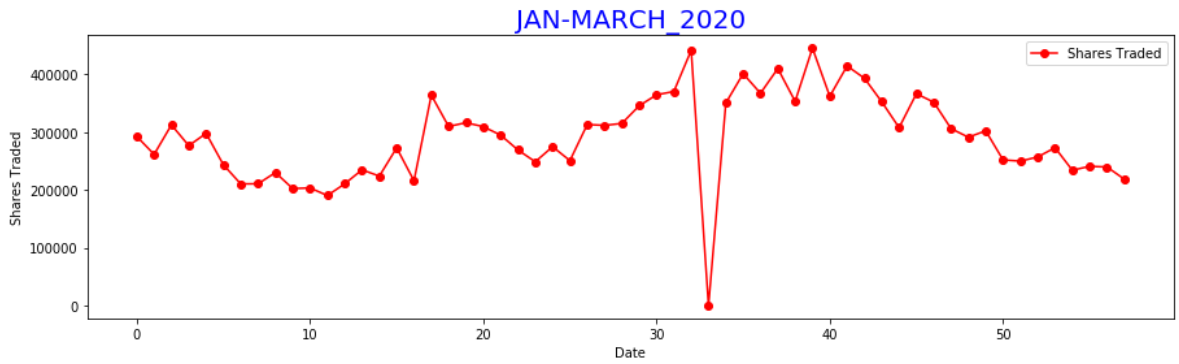
GRAPH 29

In [23]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Shares Traded')
plt.plot(dataset['Volume'],color='red',marker='o',label='Shares Traded')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[23]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```

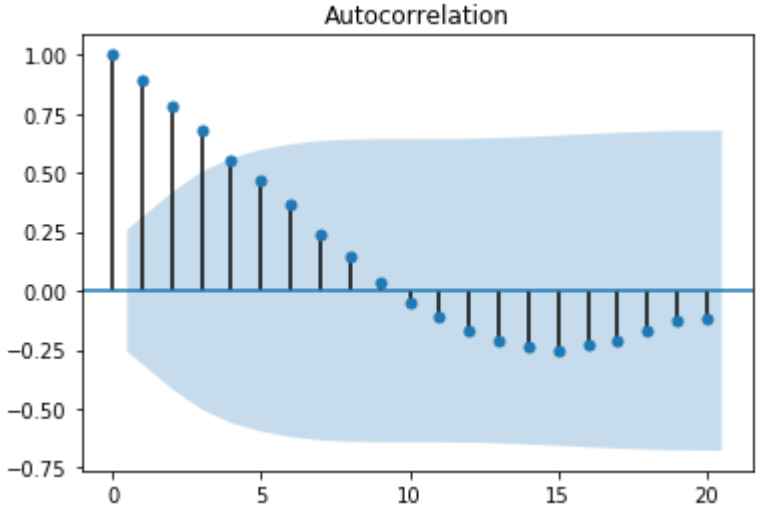


GRAPH 30

In [24]:

```
plt.figure(figsize=(15,4))
#CHECKING STATIONARY OF THE DATA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
# Show autocorrelation upto lag 10
acf_plot = plot_acf( dataset.Close,
lags=20)
```

```
<Figure size 1080x288 with 0 Axes>
```



GRAPH 31

In [25]:

```
dataset['Close_diff'] = dataset.Close - dataset.Close.shift(1)
dataset_diff=dataset.dropna()
dataset_diff.head()
```

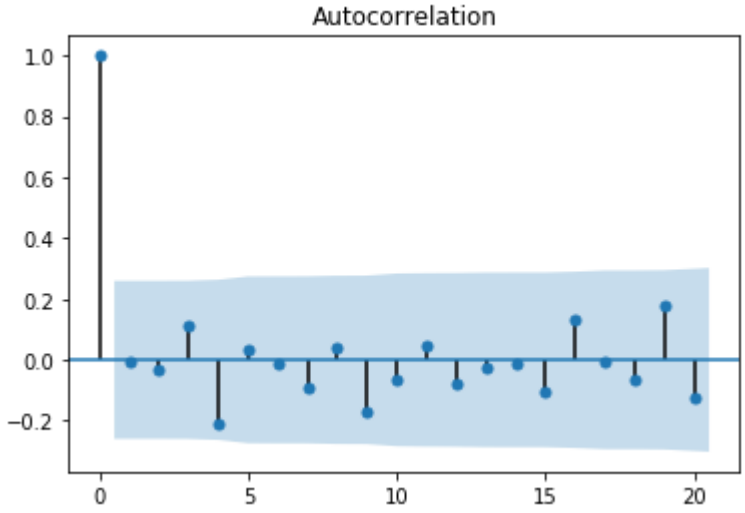
Out[25]:

	Date	Open	High	Low	Close	Adj Close	Volume	Close_diff
1	03-01-2020	3089.021973	3093.819092	3074.518066	3083.785889	3083.785889	261500	-1.412109
2	06-01-2020	3070.908936	3107.202881	3065.309082	3083.407959	3083.407959	312600	-0.377930
3	07-01-2020	3085.488037	3105.450928	3084.329102	3104.802002	3104.802002	276600	21.394043
4	08-01-2020	3094.239014	3094.239014	3059.131104	3066.893066	3066.893066	297900	-37.908936
5	09-01-2020	3082.639893	3097.329102	3080.131104	3094.882080	3094.882080	243400	27.989014

TABLE 14

In [26]:

```
acf_plot = plot_acf( dataset_diff.Close_diff,
lags=20)
```



GRAPH 32

In [27]:

```
from statsmodels.tsa.arima_model import ARIMA
arima = ARIMA( dataset.Close.astype(np.float64).as_matrix(),
order = (0,1,1))
ar_model = arima.fit()
ar_model.summary2()
```

```
C:\Users\deepa\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWa
rning: Method .as_matrix will be removed in a future version. Use .values in
stead.
```

Out[27]:

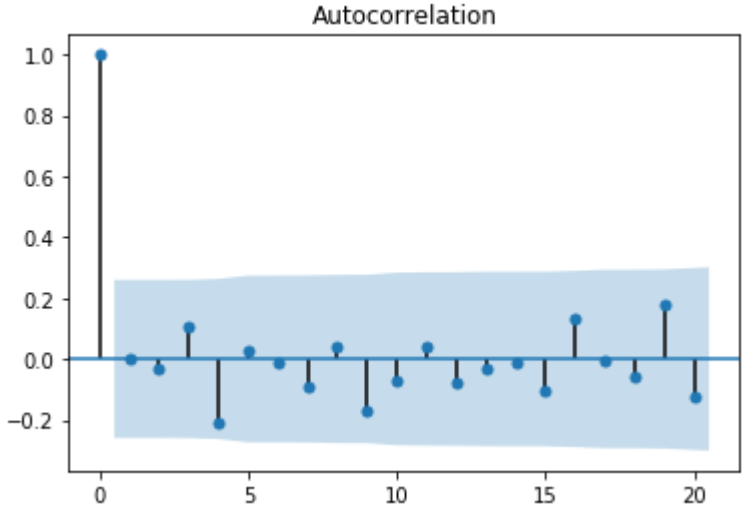
Model:	ARIMA	BIC:	623.3957
Dependent Variable:	D.y	Log-Likelihood:	-305.63
Date:	2020-04-19 23:30	Scale:	1.0000
No. Observations:	57	Method:	css-mle
Df Model:	2	Sample:	1
Df Residuals:	55		8
Converged:	1.0000	S.D. of innovations:	51.576
No. Iterations:	2.0000	HQIC:	619.649
AIC:	617.2666		

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	-5.8774	6.7772	-0.8672	0.3896	-19.1605	7.4058
ma.L1.D.y	-0.0081	0.1358	-0.0595	0.9528	-0.2742	0.2580

	Real	Imaginary	Modulus	Frequency
MA.1	123.8490	0.0000	123.8490	0.0000

In [28]:

```
acf_plot = plot_acf( ar_model.resid,
lags=20)
```



GRAPH 33

In [29]:

```
predict,stderr,ci=ar_model.forecast(steps=12)
predict
```

Out[29]:

```
array([2744.34744762, 2738.47009733, 2732.59274704, 2726.71539675,
       2720.83804646, 2714.96069617, 2709.08334588, 2703.20599559,
       2697.3286453 , 2691.45129501, 2685.57394472, 2679.69659443])
```

In [30]:

```
#PREDICTION OF 3 MONTHS
predict_2,stderr,ci=ar_model.forecast(steps=57)
predict_2
```

Out[30]:

```
array([2744.34744762, 2738.47009733, 2732.59274704, 2726.71539675,
       2720.83804646, 2714.96069617, 2709.08334588, 2703.20599559,
       2697.3286453 , 2691.45129501, 2685.57394472, 2679.69659443,
       2673.81924414, 2667.94189386, 2662.06454357, 2656.18719328,
       2650.30984299, 2644.4324927 , 2638.55514241, 2632.67779212,
       2626.80044183, 2620.92309154, 2615.04574125, 2609.16839096,
       2603.29104067, 2597.41369038, 2591.53634009, 2585.6589898 ,
       2579.78163951, 2573.90428922, 2568.02693893, 2562.14958864,
       2556.27223835, 2550.39488806, 2544.51753777, 2538.64018748,
       2532.76283719, 2526.8854869 , 2521.00813661, 2515.13078632,
       2509.25343603, 2503.37608574, 2497.49873545, 2491.62138516,
       2485.74403487, 2479.86668458, 2473.98933429, 2468.111984  ,
       2462.23463371, 2456.35728342, 2450.47993313, 2444.60258284,
       2438.72523255, 2432.84788226, 2426.97053197, 2421.09318168,
       2415.21583139])
```

In [31]:

```
predict_2=pd.DataFrame(predict_2)
predict_2.columns
```

Out[31]:

```
RangeIndex(start=0, stop=1, step=1)
```

In [32]:

```
predict_2.columns=['FORECAST']
```

In [33]:

```
predict_2.head()
```

Out[33]:

	FORECAST
0	2744.347448
1	2738.470097
2	2732.592747
3	2726.715397
4	2720.838046

In [34]:

```
dataset_2.head()
```

Out[34]:

	Date	Open	High	Low	Close	Adj Close	Volume	PREDICTE
0	01-04-2020	2743.541016	2773.364014	2731.079102	2734.521973	2734.521973	217300	2744.34744
1	02-04-2020	2720.228027	2780.637939	2719.904053	2780.637939	2780.637939	217900	2738.47009
2	03-04-2020	2773.575928	2780.586914	2754.072998	2763.987061	2763.987061	200800	2732.59274
3	07-04-2020	2806.968018	2823.277100	2801.839111	2820.762939	2820.762939	270200	2726.71539
4	08-04-2020	2805.916992	2823.214111	2800.295898	2815.368896	2815.368896	243500	2720.83804

TABLE 15

In [35]:

```
dataset_2.shape
```

Out[35]:

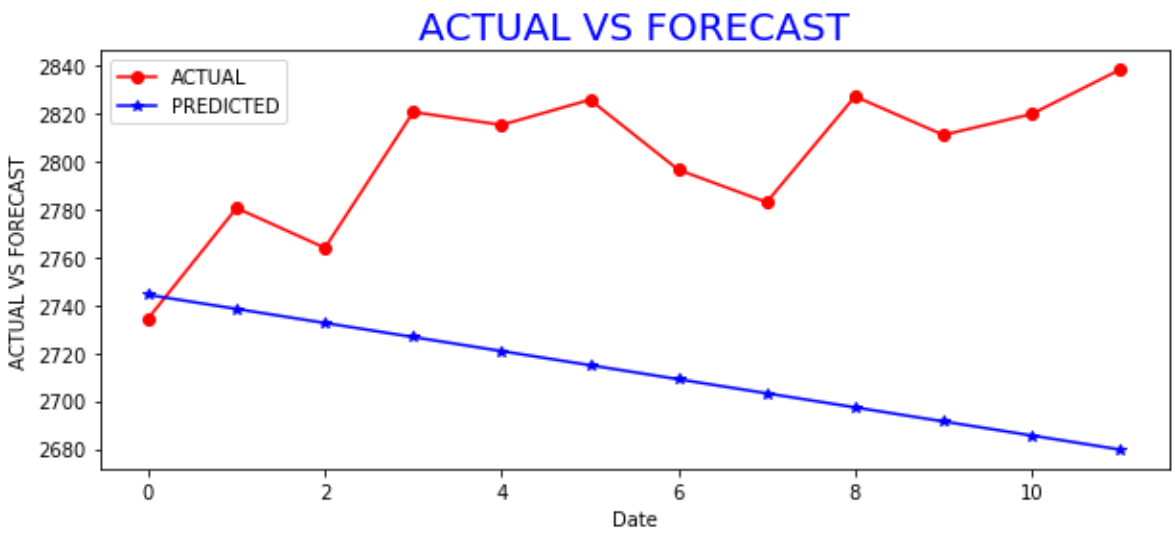
```
(12, 8)
```

In [36]:

```
plt.figure(figsize=(10,4))
plt.title('ACTUAL VS FORECAST',color='blue',size=20)
plt.xlabel('Date')
plt.ylabel('ACTUAL VS FORECAST')
plt.plot(dataset_2['Close'],marker='o',color='red',label='ACTUAL')
plt.plot(dataset_2['PREDICTED'],marker='*',color='BLUE',label='PREDICTED')
plt.legend()
```

Out[36]:

```
<matplotlib.legend.Legend at 0x206ba87f788>
```



GRAPH 34

In [37]:

```
def get_mape(actual, predicted):
 y_true, y_pred = np.array(actual), np.array(predicted)
 return np.round( np.mean(np.abs((actual - predicted) / actual)) * 100, 2 )
print('MEAN_ABSOLUTE_PERCENTAGE_ERROR:',get_mape( dataset_2['PREDICTED'],dataset_2['Close']
```

```
MEAN_ABSOLUTE_PERCENTAGE_ERROR: 3.37
```

In [38]:

```
from sklearn.metrics import mean_squared_error

print('ROOT_MEAN_SQUARE_ERROR:',np.sqrt(mean_squared_error( dataset_2['PREDICTED'],dataset_
```

```
ROOT_MEAN_SQUARE_ERROR: 100.61504104555245
```

In [17]:

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

In [18]:

```
dataset=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//japan.csv')
dataset_2=pd.read_csv('C://Users//deepa//Desktop//PROJECT_REPORT//japan_april_data.csv')
```

In [19]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58 entries, 0 to 57
Data columns (total 7 columns):
Date         58 non-null object
Open         58 non-null float64
High         58 non-null float64
Low          58 non-null float64
Close       58  non-null  float64
Adj  Close  58  non-null  float64
Volume 58 non-null int64
dtypes: float64(5), int64(1), object(1)
memory usage: 3.3+ KB
```

In [20]:

```
dataset.head()
```

Out[20]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	06-01-2020	23319.75977	23365.35938	23148.52930	23204.85938	23204.85938	72800
1	07-01-2020	23320.11914	23577.43945	23299.91992	23575.72070	23575.72070	64300
2	08-01-2020	23217.49023	23303.21094	22951.17969	23204.75977	23204.75977	79400
3	09-01-2020	23530.28906	23767.08984	23506.15039	23739.86914	23739.86914	62200
4	10-01-2020	23813.27930	23903.28906	23761.08008	23850.57031	23850.57031	55900

TABLE 16

In [21]:

```
type(dataset)
```

Out[21]:

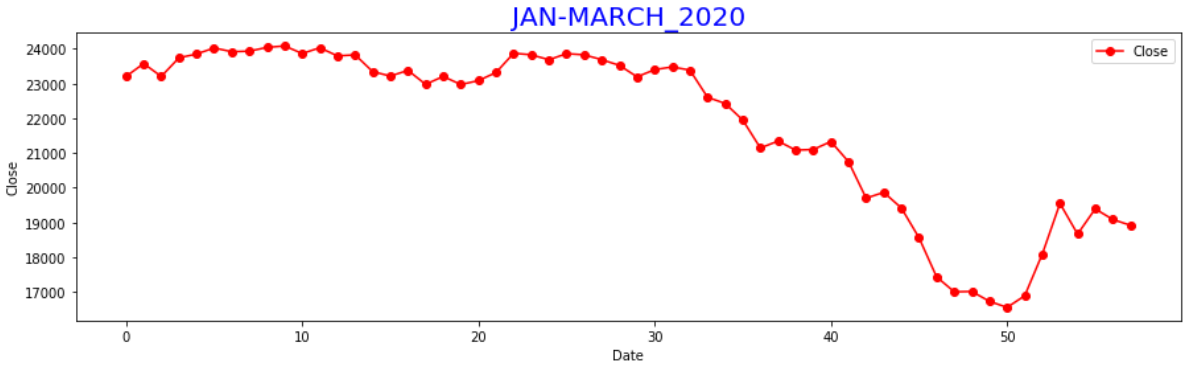
```
pandas.core.frame.DataFrame
```

In [22]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Close')
plt.plot(dataset['Close'],color='red',marker='o',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[22]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



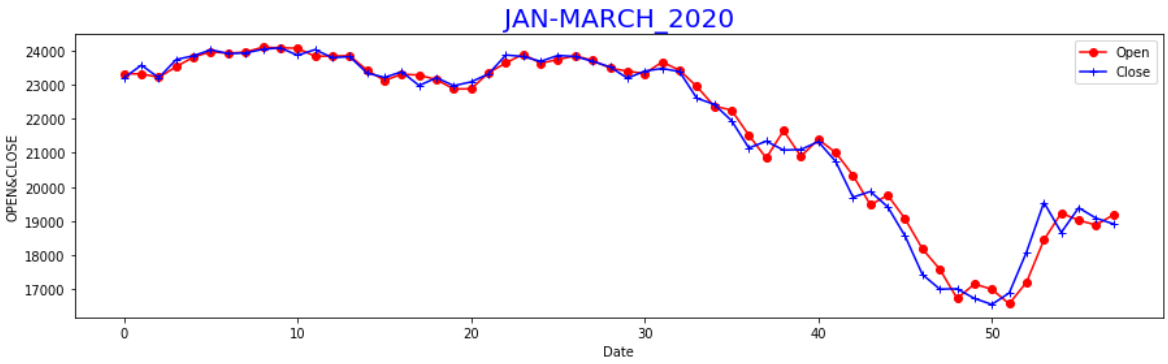
GRAPH 35

In [23]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('OPEN&CLOSE')
plt.plot(dataset['Open'],color='red',marker='o',label='Open')
plt.plot(dataset['Close'],color='blue',marker='+',label='Close')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[23]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```



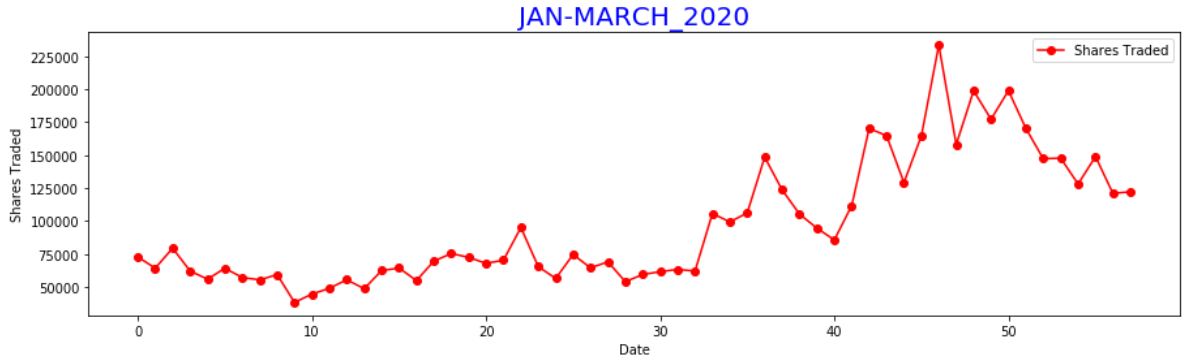
GRAPH 36

In [24]:

```
plt.figure(figsize=(15,4))
plt.xlabel('Date')
plt.ylabel('Shares Traded')
plt.plot(dataset['Volume'],color='red',marker='o',label='Shares Traded')
plt.legend()
plt.title('JAN-MARCH_2020',color='blue',size=20)
```

Out[24]:

```
Text(0.5, 1.0, 'JAN-MARCH_2020')
```

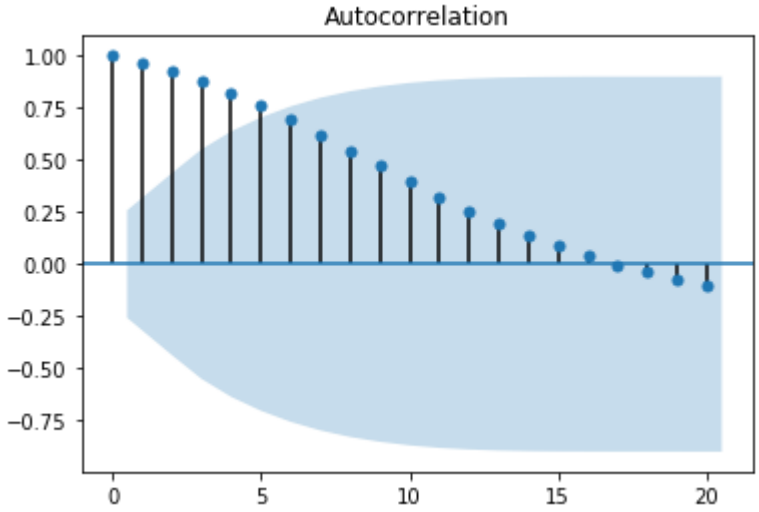


GRAPH 37

In [25]:

```
plt.figure(figsize=(15,4))
#CHECKING STATIONARY OF THE DATA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
# Show autocorrelation upto lag 10
acf_plot = plot_acf( dataset.Close,
lags=20)
```

```
<Figure size 1080x288 with 0 Axes>
```



GRAPH 38

In [26]:

```
dataset['Close_diff'] = dataset.Close - dataset.Close.shift(1)
dataset_diff=dataset.dropna()
dataset_diff.head()
```

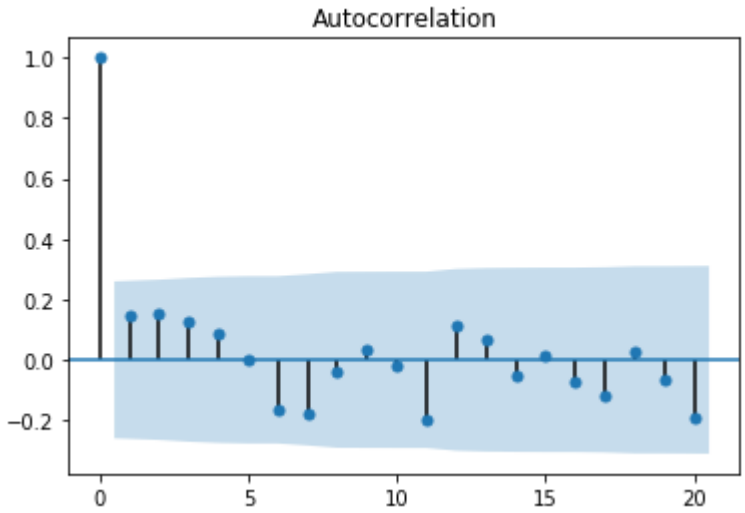
Out[26]:

	Date	Open	High	Low	Close	Adj Close	Volume	Close_diff
1	07-01-2020	23320.11914	23577.43945	23299.91992	23575.72070	23575.72070	64300	370.86132
2	08-01-2020	23217.49023	23303.21094	22951.17969	23204.75977	23204.75977	79400	-370.96093
3	09-01-2020	23530.28906	23767.08984	23506.15039	23739.86914	23739.86914	62200	535.10937
4	10-01-2020	23813.27930	23903.28906	23761.08008	23850.57031	23850.57031	55900	110.70117
5	14-01-2020	23969.03906	24059.85938	23951.66016	24025.16992	24025.16992	64200	174.59961

TABLE 17

In [27]:

```
acf_plot = plot_acf( dataset_diff.Close_diff,
lags=20)
```



GRAPH 39

In [28]:

```
from statsmodels.tsa.arima_model import ARIMA
arima = ARIMA( dataset.Close.astype(np.float64).as_matrix(),
order = (0,1,1))
ar_model = arima.fit()
ar_model.summary2()
```

```
C:\Users\deepa\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWa
rning: Method .as_matrix will be removed in a future version. Use .values in
stead.
```

Out[28]:

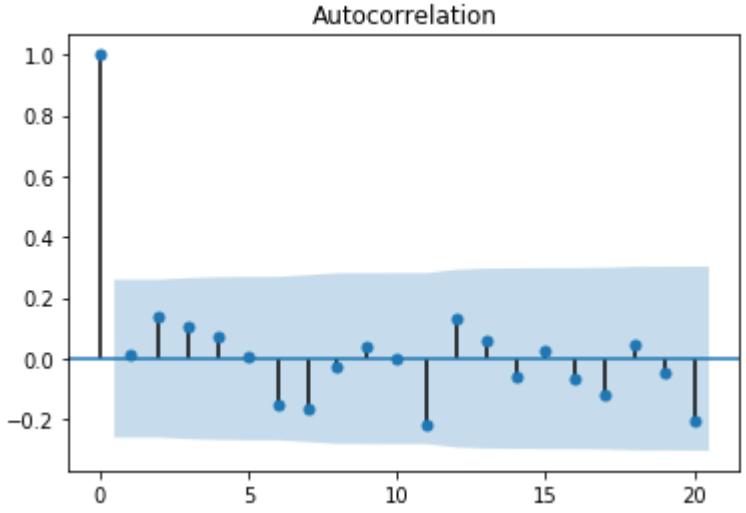
Model:	ARIMA	BIC:	873.4027
Dependent Variable:	D.y	Log-Likelihood:	-430.64
Date:	2020-04-19 23:42	Scale:	1.0000
No. Observations:	57	Method:	css-mle
Df Model:	2	Sample:	1
Df Residuals:	55		8
Converged:	1.0000	S.D. of innovations:	462.189
No. Iterations:	13.0000	HQIC:	869.656
AIC:	867.2736		

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	-74.3472	68.1989	-1.0902	0.2804	-208.0146	59.3202
ma.L1.D.y	0.1159	0.1189	0.9744	0.3341	-0.1172	0.3490

	Real	Imaginary	Modulus	Frequency
MA.1	-8.6278	0.0000	8.6278	0.5000

In [29]:

```
acf_plot = plot_acf( ar_model.resid,
lags=20)
```



GRAPH 40

In [30]:

```
predict,stderr,ci=ar_model.forecast(steps=12)
predict
```

Out[30]:

```
array([18836.32271565, 18761.97551983, 18687.62832401, 18613.28112818,
       18538.93393236, 18464.58673654, 18390.23954072, 18315.89234489,
       18241.54514907, 18167.19795325, 18092.85075743, 18018.50356161])
```

In [31]:

```
#PREDICTION OF 3 MONTHS
predict_2,stderr,ci=ar_model.forecast(steps=57)
predict_2
```

Out[31]:

```
array([18836.32271565, 18761.97551983, 18687.62832401, 18613.28112818,
       18538.93393236, 18464.58673654, 18390.23954072, 18315.89234489,
       18241.54514907, 18167.19795325, 18092.85075743, 18018.50356161,
       17944.15636578, 17869.80916996, 17795.46197414, 17721.11477832,
       17646.76758249, 17572.42038667, 17498.07319085, 17423.72599503,
       17349.3787992 , 17275.03160338, 17200.68440756, 17126.33721174,
       17051.99001591, 16977.64282009, 16903.29562427, 16828.94842845,
       16754.60123263, 16680.2540368 , 16605.90684098, 16531.55964516,
       16457.21244934, 16382.86525351, 16308.51805769, 16234.17086187,
       16159.82366605, 16085.47647022, 16011.1292744 , 15936.78207858,
       15862.43488276, 15788.08768694, 15713.74049111, 15639.39329529,
       15565.04609947, 15490.69890365, 15416.35170782, 15342.004512  ,
       15267.65731618, 15193.31012036, 15118.96292453, 15044.61572871,
       14970.26853289, 14895.92133707, 14821.57414124, 14747.22694542,
       14672.8797496 ])
```

In [32]:

```
predict_2=pd.DataFrame(predict_2)
predict_2.columns
```

Out[32]:

```
RangeIndex(start=0, stop=1, step=1)
```

In [33]:

```
predict_2.columns=['FORECAST']
```

In [34]:

```
predict_2.head()
```

Out[34]:

	FORECAST
0	18836.322716
1	18761.975520
2	18687.628324
3	18613.281128
4	18538.933932

In [35]:

```
dataset_2.head()
```

Out[35]:

	Date	Open	High	Low	Close	Adj Close	Volume	PREDICTE
0	01-04-2020	18686.11914	18784.25000	17871.61914	18065.41016	18065.41016	105800	18836.3227
1	02-04-2020	17934.41992	18132.03906	17707.66016	17818.72070	17818.72070	107300	18761.9755
2	03-04-2020	17951.43945	18059.15039	17646.50000	17820.18945	17820.18945	96000	18687.6283
3	06-04-2020	17857.99023	18672.25977	17802.61914	18576.30078	18576.30078	105700	18613.2811
4	07-04-2020	18878.85938	19162.51953	18553.14063	18950.17969	18950.17969	108200	18538.9339

TABLE 18

In [36]:

```
dataset_2.shape
```

Out[36]:

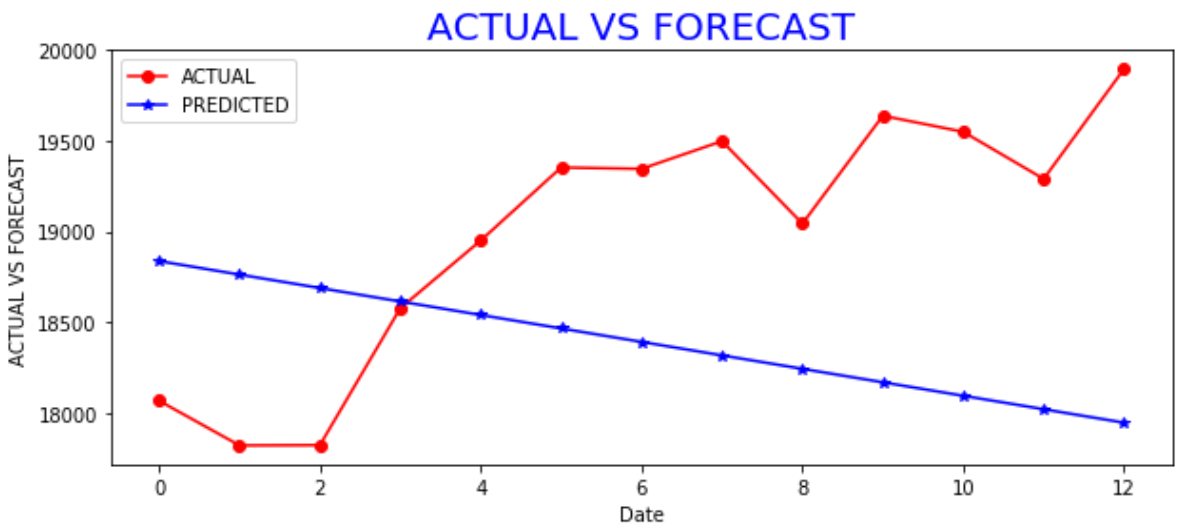
```
(13, 8)
```

In [37]:

```
plt.figure(figsize=(10,4))
plt.title('ACTUAL VS FORECAST',color='blue',size=20)
plt.xlabel('Date')
plt.ylabel('ACTUAL VS FORECAST')
plt.plot(dataset_2['Close'],marker='o',color='red',label='ACTUAL')
plt.plot(dataset_2['PREDICTED'],marker='*',color='BLUE',label='PREDICTED')
plt.legend()
```

Out[37]:

```
<matplotlib.legend.Legend at 0x237c7333d48>
```



GRAPH 41

In [38]:

```
def get_mape(actual, predicted):
 y_true, y_pred = np.array(actual), np.array(predicted)
 return np.round( np.mean(np.abs((actual - predicted) / actual)) * 100, 2 )
print('MEAN_ABSOLUTE_PERCENTAGE_ERROR:',get_mape( dataset_2['PREDICTED'],dataset_2['Close']
```

```
MEAN_ABSOLUTE_PERCENTAGE_ERROR: 5.47
```

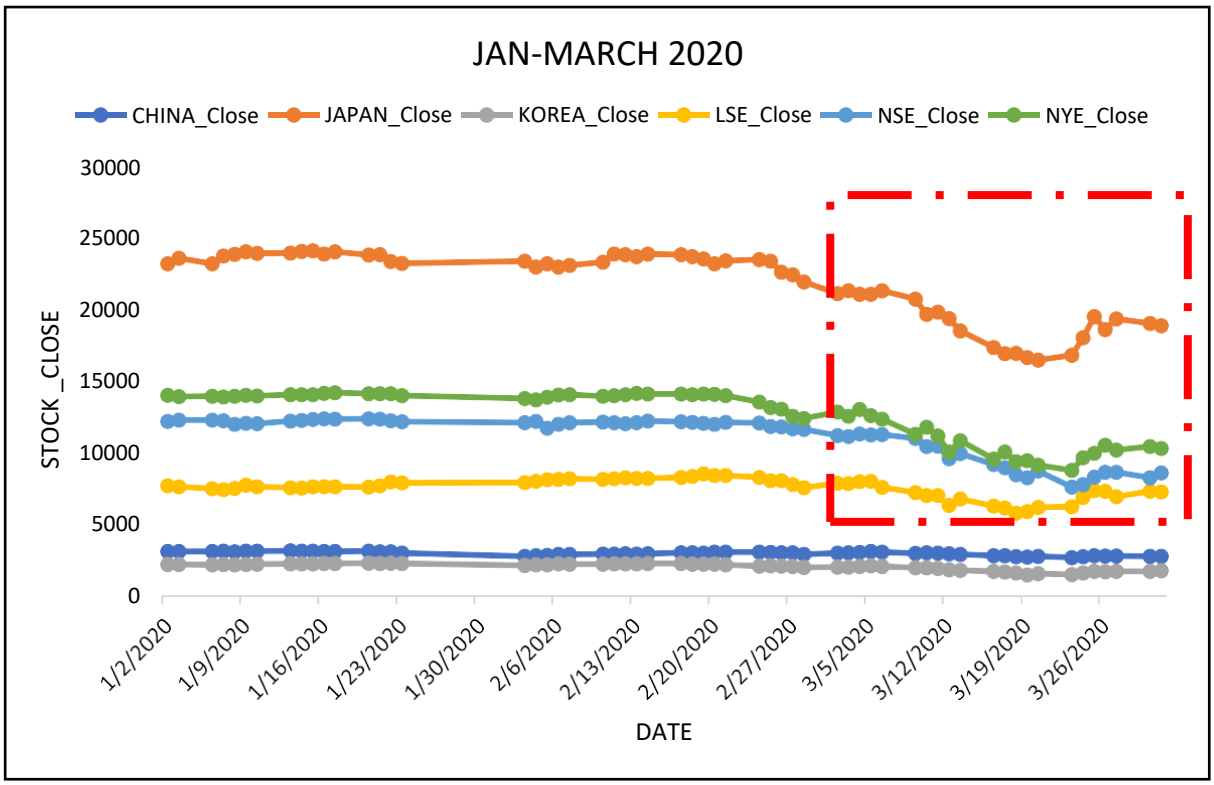
In [39]:

```
from sklearn.metrics import mean_squared_error

print('ROOT_MEAN_SQUARE_ERROR:',np.sqrt(mean_squared_error( dataset_2['PREDICTED'],dataset_
```

```
ROOT_MEAN_SQUARE_ERROR: 1104.963071405453
```

STOCK_MARKET_OF 6_MAJOR_ECONOMIES(JAN-MARCH_2020)

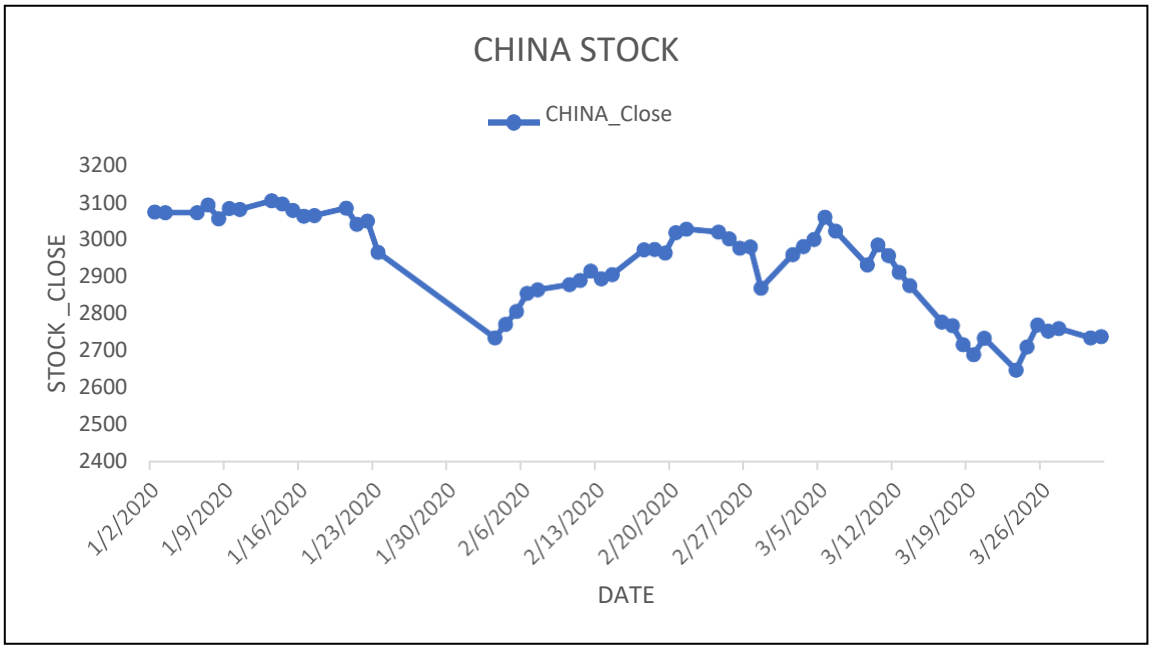


GRAPH 42

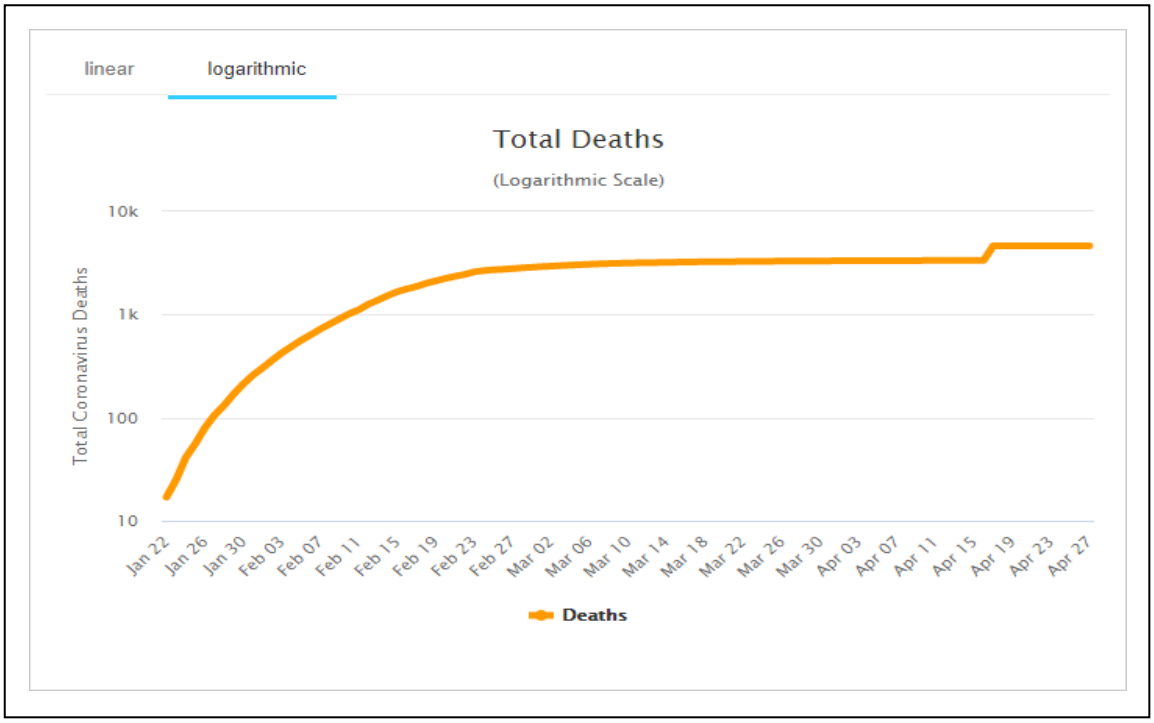
- Panic was maximum in the month of March-2020
- China banned its stock exchange from trading shares
- South_Korea was able to retain the impact of Covid-19 which is reflected in their stock market

STOCK_MARKETS_COMPARISON

WITH COVID DEATH



GRAPH 43



GRAPH 44

ECONOMIC STIMULUS MEASURES:

The People's Bank Of China, 3rd of February, 2020

- Launched 1.2 trillion Yuan of the public market reverse repurchase operation on February 3rd

The People's Bank Of China 10th of February, 2020

- Issuing the first batch of the special re-loans

The People's Bank Of China, 17th of February, 2020

- Carry out medium-term lending facility (MLF) of RMB 200 billion and 7-days reverse repos of RMB 100 billion, and the interest rate of this MLF is 10 BP lower than the previous

The state council executive meeting, 18th of February, 2020

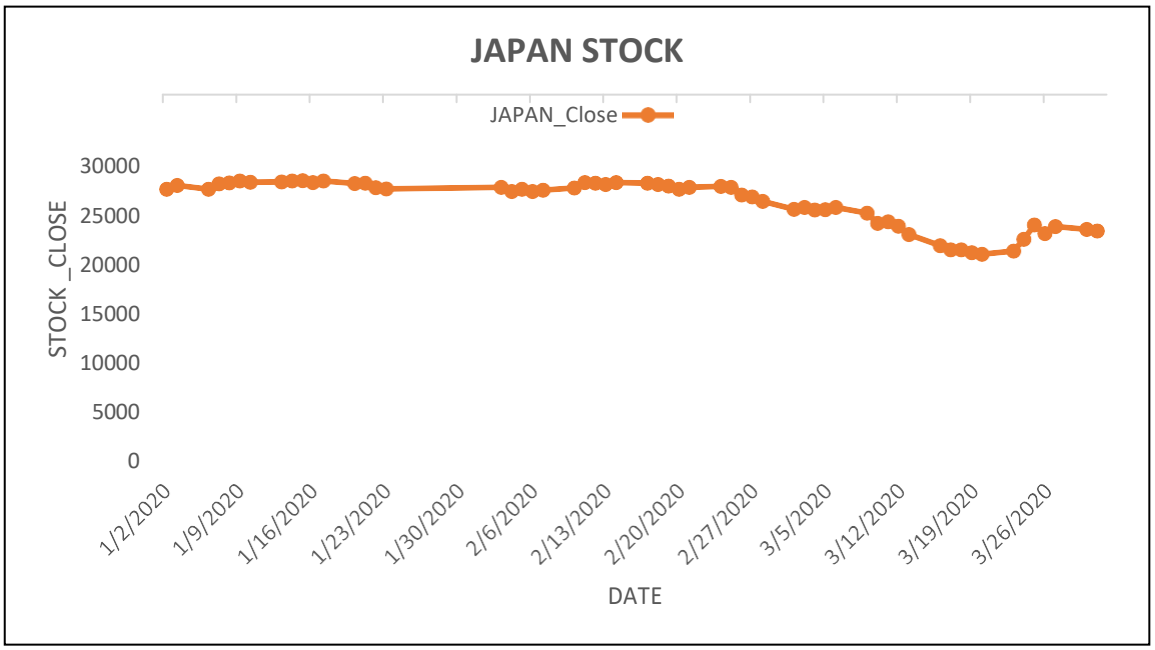
- Phased reduction and exemption of corporate social insurance fees and implementing the policy of payment delaying of housing fund by enterprises

The People's Bank Of China(PBOC) 20th of February, 2020

- LPR interest rate reduction operation

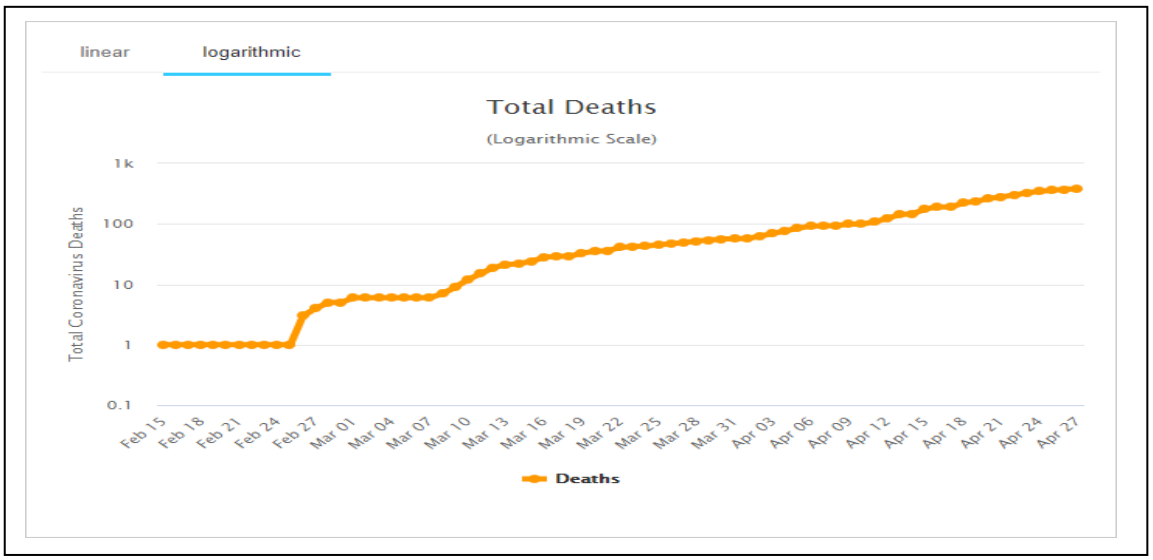
Cuts on required reserve ratio

- Cuts on required reserve ratio (one in March and one in April), each injecting RMB550bn and RMB400bn liquidity respectively into the market.



GRAPH 45

- Fluctuation was less in the 2 month data.
- Much Fluctuation was seen in March data.
- Lockdown measures and rapid testing help to keep the market strong
- Govt keep infusing the liquidity in the market which help to sustain the market.



GRAPH 46

ECONOMIC MEASURES:

- Doubling the target for net purchases of exchange-traded funds to JPY 12 trillion ($112 billion)
- Agreeing to coordinated foreign swap lines, to lower the cost of borrowing dollars internationally, with the US Federal Reserve
- Increasing the upper limit for its purchases of commercial paper and corporate bonds by ¥2tn.

Loan for large companies

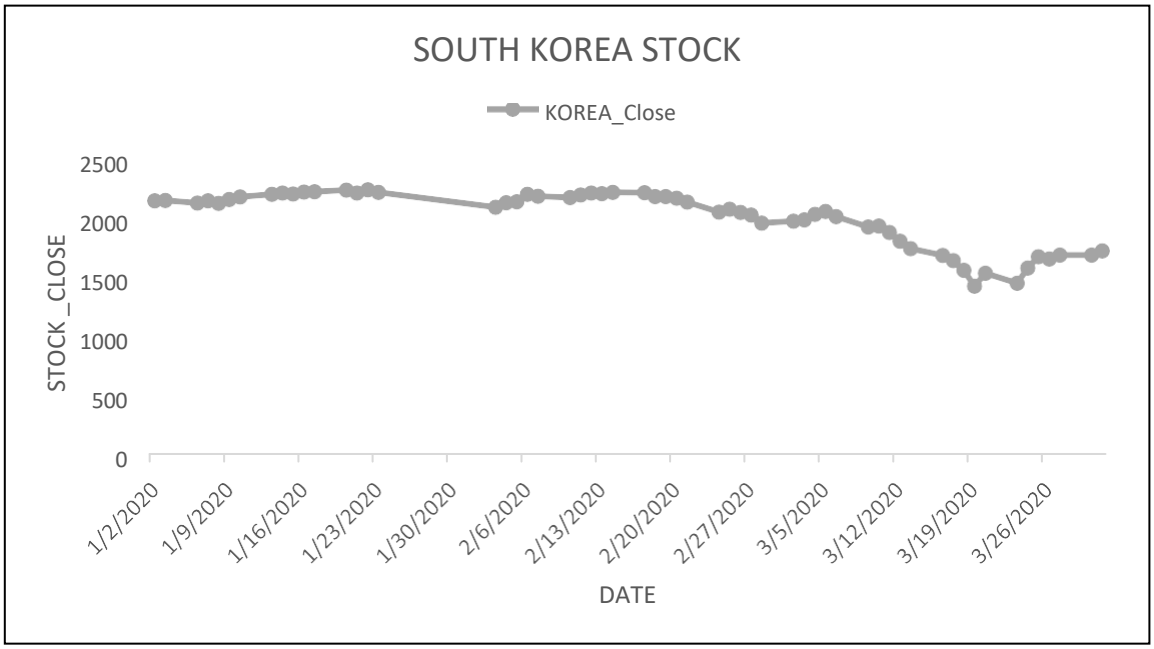
- Ministry of Finance(MOF) decided to use loan for crisis response. Low interest rate loan provided by Development Bank of Japan(DBJ) .

Loan, loan guarantee and cash benefit for SMEs

- Ministry of Finance(MOF) decided to use safety net loan(loan limit:JPY720million) and loan for crisis response(loan limit:JPY300million) to support SMEs. Low interest rate loan provided by government financial institutions (JFC, Shoko Chukin Bank etc.).
- Ministry of Finance(MOF) established special loan program for novel coronavirus. Low interest rate loan provided by Japan Finance Corporation(JFC) and Okinawa development finance corporation. Loan limit is JPY300milion.
- Ministry of Economy, Trade and Industry(METI) decided to establishment on subsidy program for sustaining businesses.

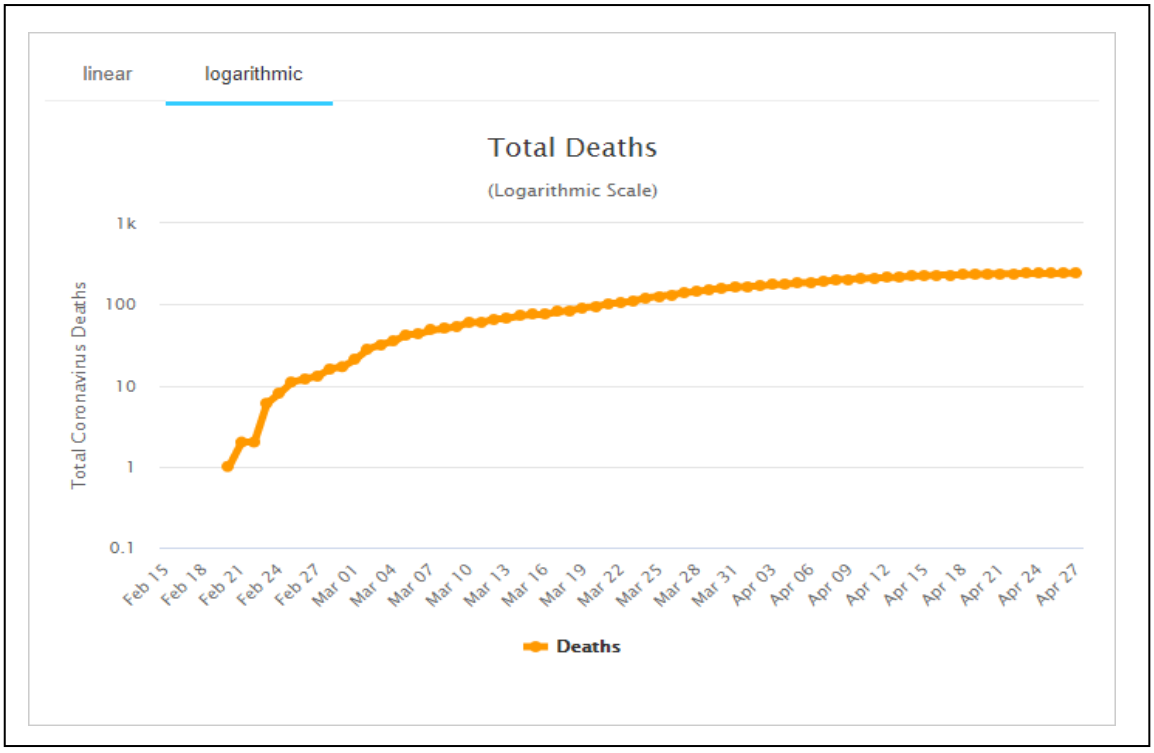
Customs Measures:

- Priority is given to the import and export customs clearance of goods that need to be cleared urgently, such as relief goods related to countermeasures against coronavirus and goods for securing lifelines.



GRAPH 47

- Fluctuation was less in the 3 month data.



GRAPH 48

Economic stimulus measures

Monetary Policy

- On March 23, The Bank of Korea ledged to begin purchasing an unspecified amount of local bonds to help prevent a possible liquidity crunch as well as expand the scope of its purchase program to include bonds issued by public enterprises.
- The Bank of Korea slashed its benchmark interest rate to 0.75% in an emergency move following actions by the Federal Reserve.
- Korea will lower interest rates applied to its loan facility for smaller companies, and add bonds issued by banks to its open market operations to enhance liquidity

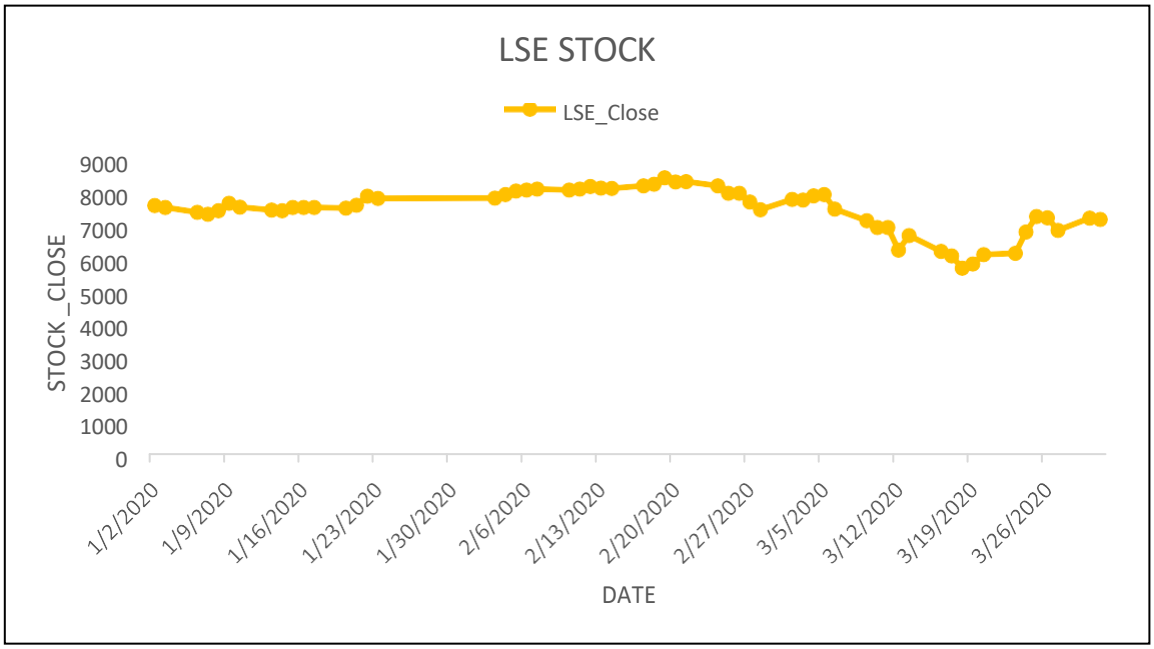
Customs Measures

- Delayed payment of duties/instalment payment
- Suspension of Customs audits
- Filing paperless duty drawback

Other measures and sources

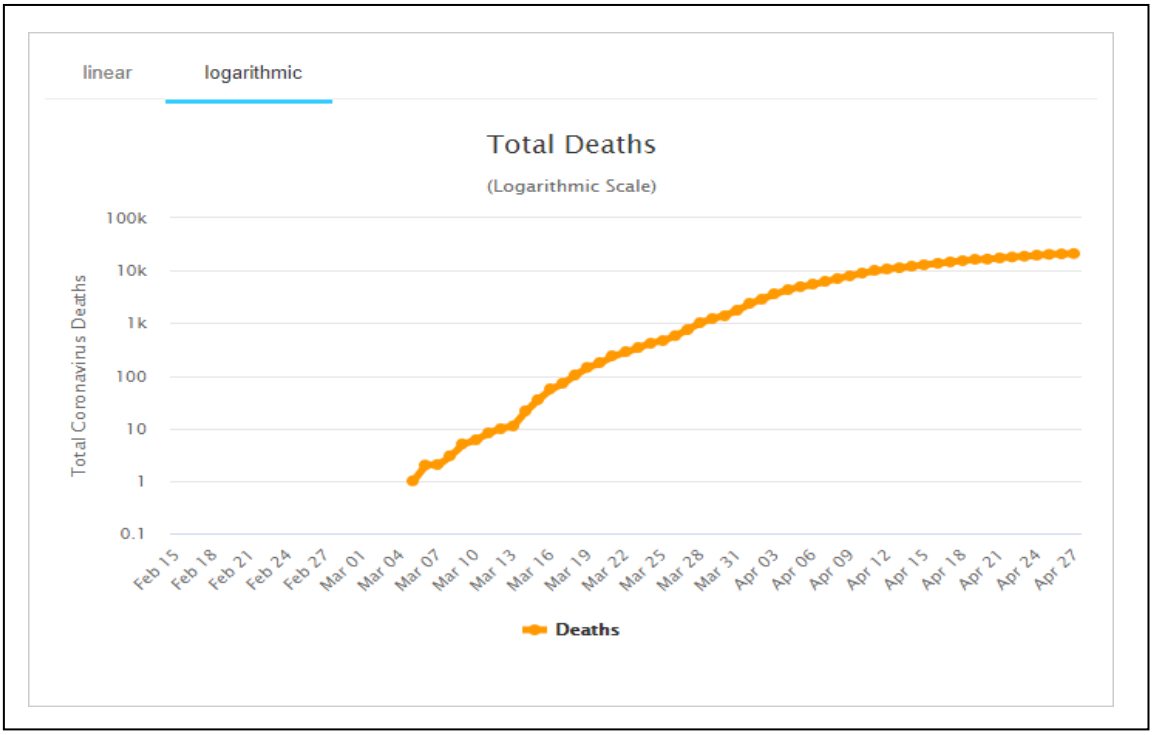
Trade restrictions:

- On March 19, Tokyo imposed additional restrictions on South Koreans seeking entry into Japan. Subsequently, Seoul imposed similar restrictions.
- These restrictions have not produced any new obstacles to bilateral trade, but it could indicate rising tension and further complicate efforts to resolve an existing trade dispute between the two countries that began last summer.



GRAPH 49

- Not much fluctuation seen



GRAPH 50

ECONOMIC MEASURES:

- Under the Coronavirus Business Interruption Loan Scheme (CBILS) UK businesses with annual turnover of no more than £45m can borrow up to £5m interest-free for 12 months under a British Business Bank (BBB) scheme where the Government provides the lender with a guarantee for 80% of each loan (subject to a per-lender cap on claims) and covers the cost of the first 12 months of interest.
- Financing can be provided under CBILS for up to 6 years through term loans, overdrafts, invoice finance and asset finance.
- The £45m turnover threshold applies to group turnover, rather than at individual company turnover level.

Coronavirus Large Business Interruption Loan Scheme (for Businesses with turnover of more than £45m)

- The Chancellor announced that a new scheme, Coronavirus Large Business Interruption Loan Scheme (CLBILS) is to be introduced. Similar to the SME CBILS scheme this involves a government guarantee of 80% to enable banks to make loans of up to £25 million (CBILS was capped at £5 million) to businesses with an annual turnover of between £45 million and £250 million. Firms with a turnover of more than £250 million can borrow up to £50 million from lenders.
- This is intended to give banks the confidence to lend to more businesses which are impacted by coronavirus but which they would not lend to without CLBILS..

COVID-19 Corporate Finance Facility (CCFF)

- The CCFF has been created to provide funding to large businesses through the purchase of short-term corporate debt in the form of commercial paper..
- Funding is open to companies (1) making “a material contribution to the UK economy”; (2) able to demonstrate they were in sound financial health prior to the pandemic; and (3) with a short term or long term investment grade credit rating or otherwise able to demonstrate financial

- The CCFF launched on 23 March 2020 and Bank of England data released on 2 April 2020 showed that £1.9 billion of commercial paper has been purchased under this facility already and according to a HM Treasury release on 3 April 2020 a further £1.6 billion has been committed.

£750m coronavirus fund for frontline charities

- The UK government has announced £750m of funding for frontline charities across the UK – including hospices and those supporting domestic abuse victims. £60m of this will be provided to Scotland, Wales and Northern Ireland.
- £360 million will be directly allocated by government departments to charities providing key services and supporting vulnerable people during the crisis – including hospices and domestic abuse victims. £370m will go to small and medium-sized charities, including through a grant to the National Lottery Community Fund for those in England.

Future Fund for high-growth companies

- The government announced that it will be establishing a £500m loan scheme aimed at ensuring that high-growth companies in the UK receive the investment they need to continue during the crisis.
- The scheme will be delivered in partnership with the British Business Bank and is due to launch in May.
- It will provide UK-based companies with between £125,000 and £5m from government, with private investors at least matching the government's commitment. (The £500m fund is comprised of £250 million from government combined with equal match funding from private investors.)
- These loans will automatically convert into equity on the company's next qualifying funding round, or at the end of the loan if they are not repaid.
- The scheme is open to unlisted UK registered companies that have previously raised at least £250,000 in equity investment from third party investors in the last five years.

Support for SMEs focused on research and development

- as part of its wider package of support for innovative firms hit by the COVID-19 outbreak, the government announced £750m of targeted support for small and medium sized businesses focusing on research and development.
- Innovate UK, the national innovation agency, will accelerate up to £200 million of grant and loan payments for its 2,500 existing Innovate UK customers on an opt-in basis.

Customs Measures

Duty relief

- HMRC have introduced a relief on the import of medical supplies and equipment, protective garments and similar goods imported from non-EU countries to combat the impact of COVID 19.

Payment facilities

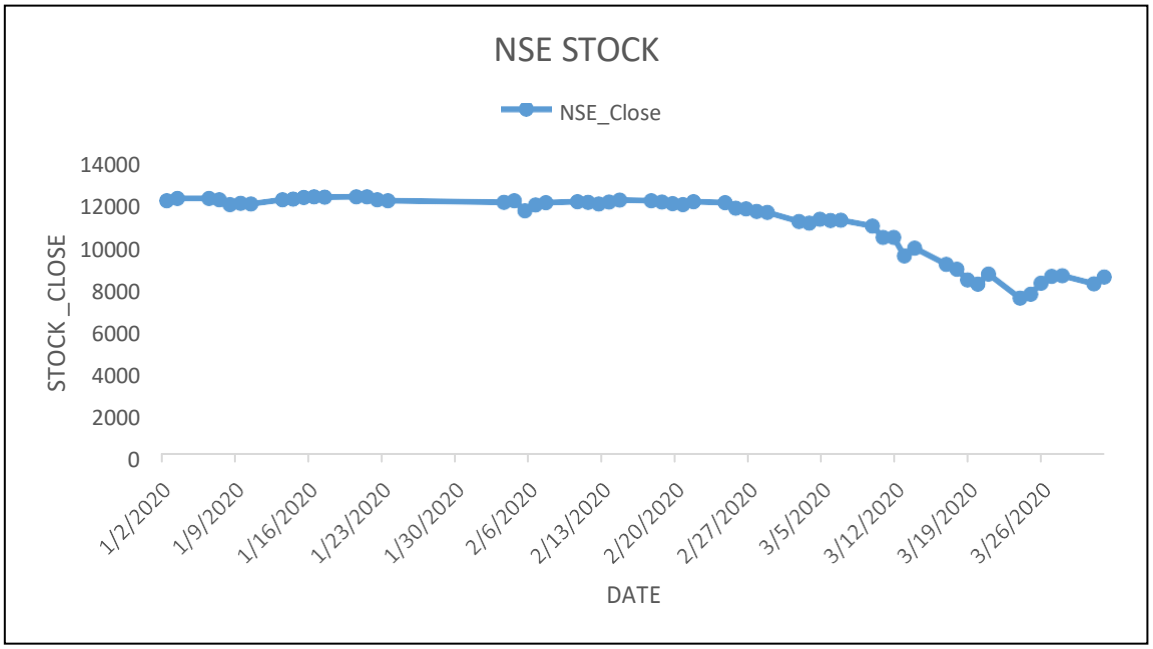
- Deferral of UK VAT due in between 20 March 2020 and 30th June 2020 up till the 31st March 2021. All VAT deferred payments are required to be paid by the 31st March 2021..

Customs procedures

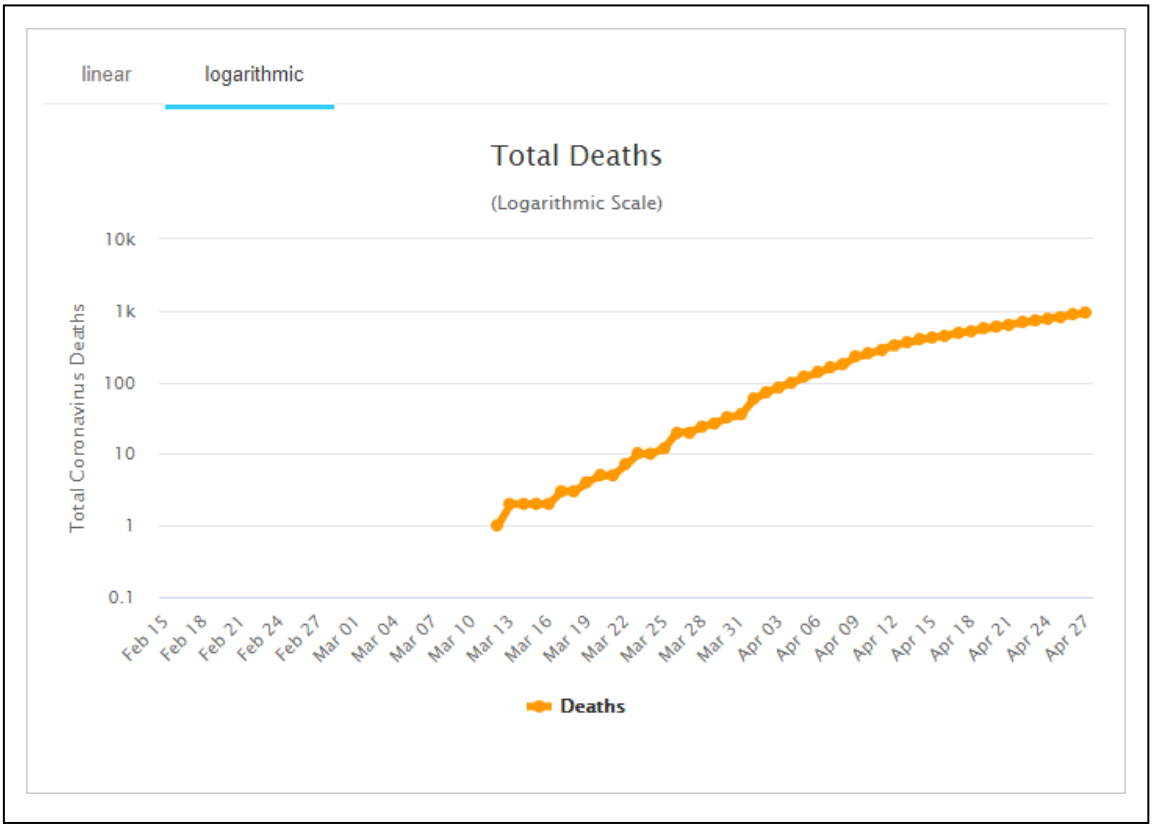
- HMRC have implemented an automatic extension scheme on all UK authorised customs special procedures

Other measures

- Our understanding is that there is a priority for movement of goods - medical or scientific supplies etc.



GRAPH 51



GRAPH 52

Economic stimulus measures

Financial services (Relaxation for 3 Months):

- Complete waiver of minimum balance charges for Savings Bank account

- Debit card holders can withdraw cash from any bank ATM for free of charge
- Bank charges for digital trade transactions will be reduced for all trade finance customers

Insolvency and Bankruptcy Code (IBC):

- Threshold of default under section 4 of the IBC has been increased from Rs 100,00 to Rs 10 million with the intention to prevent triggering of insolvency proceedings against MSMEs.
- If the current situation continues beyond 30 April 2020, Section 7, 9 and 10 of IBC to be suspended for 6 months in an effort to stop companies at large from being forced into insolvency proceedings in such force majeure causes of default.
- Relief measures announced by Reserve Bank of India on 27 March 2020:

Liquidity measures

- Reduction of policy repo rate by 75 basis points (from current 5.15% to 4.40%)
- RBI will conduct auctions of TLTRO (Targeted Long Term Repo Operations) of up to three-year tenor of appropriate sizes for a total amount up to INR 1 lakh crore (~USD 13 billion) at a floating rate, linked to policy repo rate
- CRR of all banks to be reduced by 100 basis points to 3% beginning March 28, for 1 year. This will release liquidity of INR 1,37,000 crore across the banking system
- MSF raised from 2% of SLR to 3% with immediate effect. Applicable up to June 30, 2020.
  These three liquidity measures will inject liquidity of INR 3.74 lakh crore (~USD 50 billion) to the system.

Regulatory measures

- All lending institutions are being permitted to allow a moratorium of three months on repayment of installments for term loans outstanding as on March 1, 2020

- Lending institutions permitted to allow deferment of 3 months on payment of interest w.r.t all such working capital facilities o/s as of March 1, 2020
- Deferring payments will not result in asset classification downgrade.
- Further deferring implementation of last tranche of 0.625 % of capital conservation buffer to Sept. 30, 2020
- Banks in India that operate IFSC banking units allowed to participate in offshore INR NDF market w.e.f. June 1, 2020

Custom Measures

Exports of medical equipment

- Prohibition on exports of following:
- Surgical masks/disposable masks (2/3 ply),
- Textile material for masks and coveralls
- All ventilators including any artificial respiratory apparatus or oxygen therapy apparatus or any breathing appliance/device
- Sanitizers
- Hydroxychloroquine (allowed on certain cases on a government to government basis only)
- Formulations made from hydroxychloroquine

Imports of medical equipment

- Exemption from customs duty and health cess, (w.e.f .9 April 2020 to 30 September 2020) on import of following:
- Artificial respiration or other therapeutic respiration apparatus (ventilators)
- Face masks and surgical masks
- Personal protection equipment (PPE)
- COVID-19 testing kits

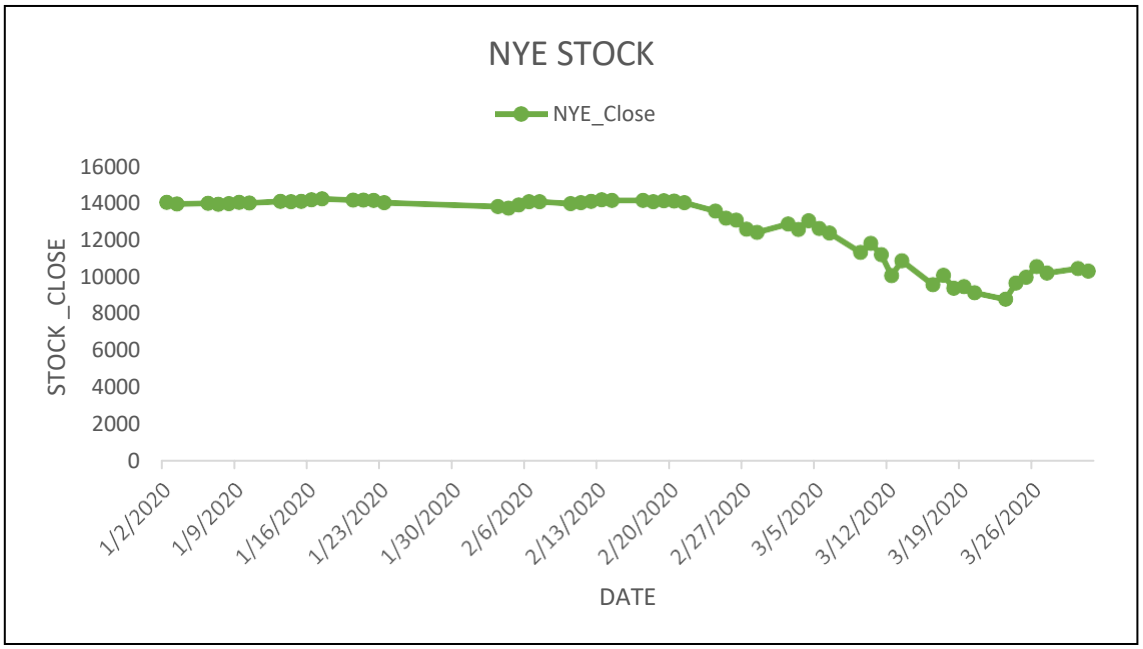
- Inputs for manufacturing of aforesaid four items (subject to certain conditions)

Other measures and sources

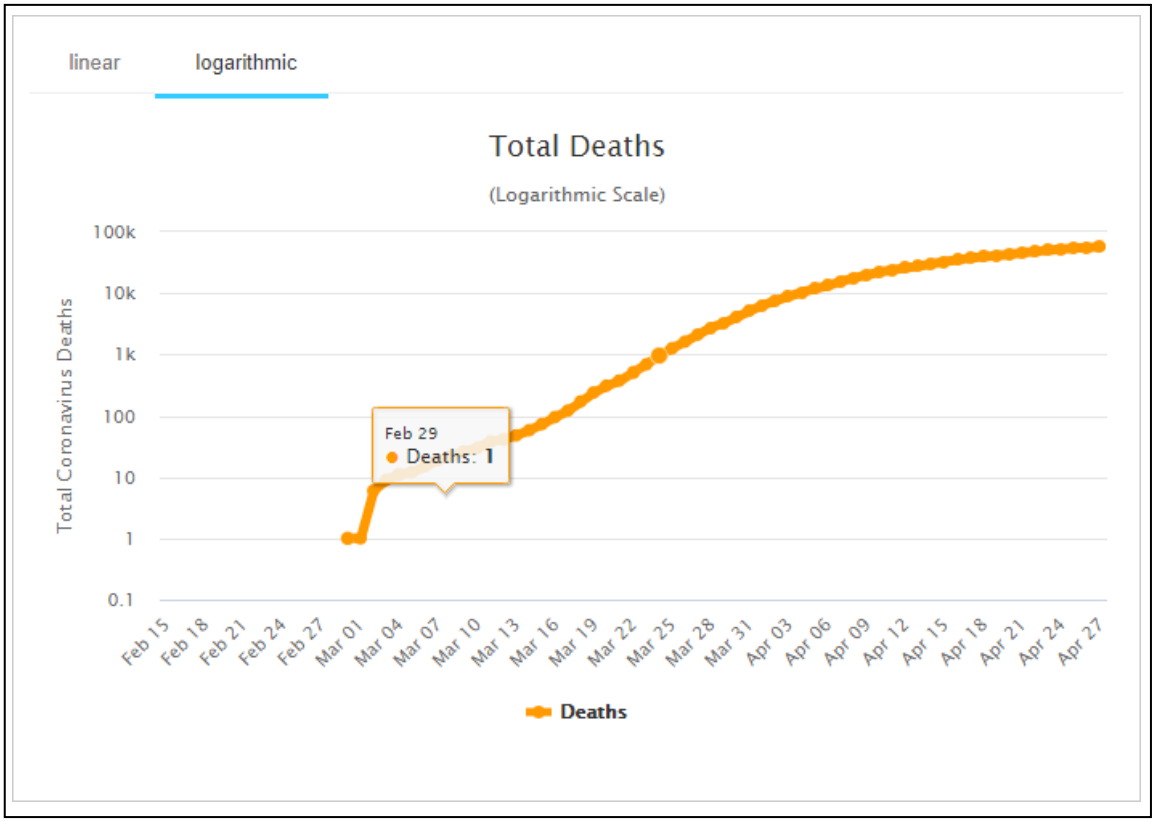
Corporate affairs:

- No fees to be charged for late filing during moratorium period (01 April 2020 to 30 September 2020, 6 months) in respect of any document, return, statements, etc. required to be filed with MCA (Ministry of Corporate Affairs)
- The mandatory requirement of holding board meeting within 120 days of last meeting shall be extended by period of 60 days. Relaxation is till 30 September 2020.
- Applicability of CARO, 2020 has been shifted to FY21 instead of FY20. (CARO is Companies Auditors' Report Order)
- Companies Act requirement of creating deposit reserve of 20% of deposits maturing in FY21 and investing 15% of debentures maturing in FY21 before 30 April 2020 may be done before 30 June 2020

GRAPH 53



- Not much Fluctuation was less in the 3 month data.



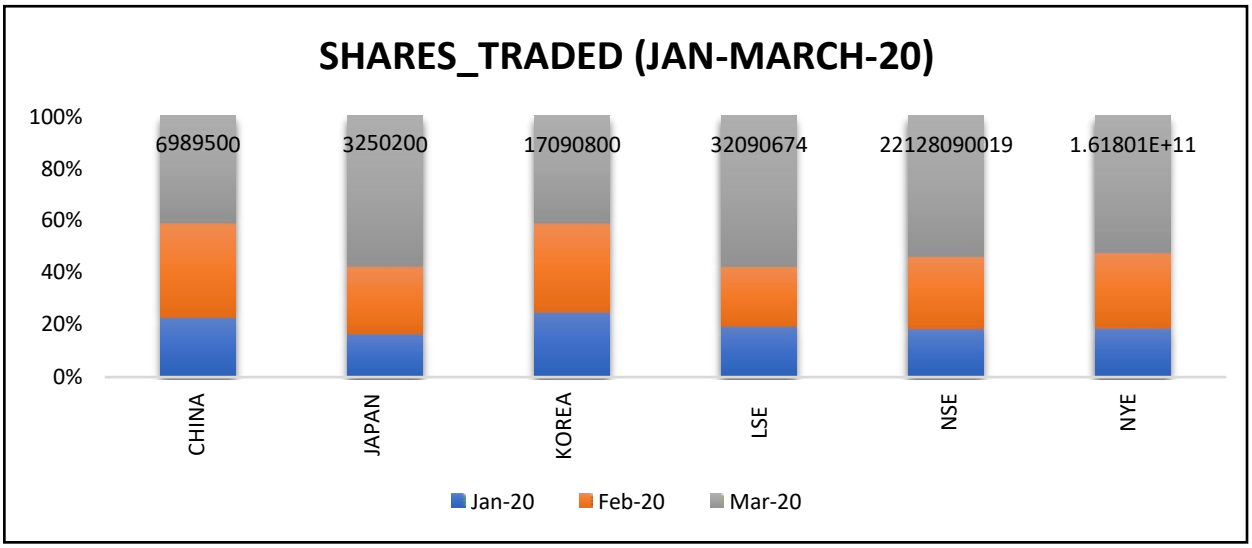
GRAPH 54

Economic stimulus measures

- A Primary Dealer Credit Facility (PDCF) to offer overnight and term funding with maturities up to 90 days to Primary Dealers of the New York Federal Reserve Bank.
- Money Market Mutual Fund Liquidity Facility (MMLF) that will make loans available to eligible financial institutions secured by high-quality assets purchased by the financial institution from money market mutual funds. "Eligible financial institutions" are defined as U.S. depository institutions, U.S. bank holding companies, and U.S. branches and agencies of a foreign bank.
- The Primary Market Corporate Credit Facility (PMCCF) for new bond and loan issuances. This facility is open to investment grade companies and will provide bridge financing of four years.
- The Secondary Market Corporate Credit Facility (SMCCF) to provide liquidity for outstanding corporate bonds.

- Provides $349 billion in direct appropriations for Small Business Administration (SBA) loan guarantees and additional funding for SBA programs and relief to small business borrowers and lenders.
- Issued an order providing temporary conditional exemptive relief for business development companies (BDCs) to enable them to make additional investments in small and medium-sized businesses, including those with operations affected by COVID-19.
- Issued disclosure guidance providing the staff's current views regarding disclosure and other securities law obligations that companies should consider with respect to COVID-19 and related business and market disruptions.

SHARES_TRADED COMPARISON (JAN-MARCH 2020)



GRAPH 55

- March is the peak as the corona cases were high , Shares trading was high as many investors leave the market to minimize the loss.
- Panic was the max in the month of March

LIMITATIONS OF THE STUDY

- Although the care has been taken but there is always an factor of uncertainty in the market .

- The existence system reported highly predictive values, by selecting an appropriate time period for their experiment to obtain highly predictive scores.

- The existing system does not perform well when there is a change in the operating environment.

- It doesn‟t focus on external events in the environment, like news events or social media.

- It exploits only one data source, thus highly biased

FUTURE ENHANCEMENT

- Future scope of this project will involve adding more parameters and factors like the financial ratios, multiple instances, etc.
- The more the parameters are taken into account more will be the accuracy.
- The algorithms can also be applied for analyzing the contents of public comments and thus determine patterns/relationships between the customer and the corporate employee.
- The use of traditional algorithms and data mining techniques can also help predict the corporation’s performance structure as a whole.

REFERENCES

- Ashish Sharma, Dinesh Bhuriya, Upendra Singh. "Survey of Stock Market Prediction Using Machine Learning Approach", ICECA 2017.

- VivekKanade, BhausahebDevikar, SayaliPhadatare, PranaliMunde, ShubhangiSonone. “Stock Market Prediction: Using Historical Data Analysis”, IJARCSSE 2017.

- SachinSampatPatil, Prof. Kailash Patidar, Asst. Prof. Megha Jain, “A Survey on Stock Market Prediction Using SVM”, IJCTET 2016.

- Hakob GRIGORYAN, “A Stock Market Prediction Method Based on Support Vector Machines (SVM) and Independent Component Analysis (ICA)”, DSJ 2016.

- RautSushrut Deepak, ShindeIshaUday, Dr. D. Malathi, “Machine Learning Approach In Stock Market

ANNEXURE 1

INTRODUCTION OF VARIOUS STOCK EXCHANGES

INDIA

The Nifty is an flagship benchmark of the National Stock Exchange (NSE), which is an diversified index, comprising top 50 companies in terms of free-float market capitalisation that are traded on the stock market. It is used to reflect the health of the listed universe of Indian companies, and hence the broader economy, in all market conditions**.**

Officially called the Nifty50, the index is computed using the free float market capitalisation method, which is actually count of shares in active circulation in the market at any given point of time. The Nifty brand and indices are managed by the Mumbai-based India Index Services and Products Limited (IISL) which itself is a subsidiary of NSE. IISL has a 3-tier governance structure comprising the board of directors, the index policy committee and the index maintenance sub-committee. IISL managed 67 indices under the Nifty brand as of September 30, 2016. IISL rebalances the Nifty index semi-annually. The cut-off dates for the semi-annual review of the index are January 31 and July 31 annually. Average data for the six months ending the deadline is taken into account. The exchange notifies any change within the index four weeks before such changes become. There are defined eligibility criteria for selection of Nifty constituent stocks. The liquidity of a stock is measured by the market impact cost, which is actually the value involved in transacting a stock. For a stock to qualify for inclusion within the Nifty50, it must have traded at an mean impact cost of 0.50 per cent or less for 6 months and for 90 per cent of observation cases. Only those stocks which are eligible for

trade in the F&O segment of NSE are considered for inclusion as Nifty constituents.

USA

The NY stock market (NYSE) is stock market located in NY City that's the most important equities-based exchange within the world, supported the entire market capitalisation of its listed securities. Formerly run as a personal organization, the NYSE became a public entity in 2005 following the acquisition of electronic trading exchange Archipelago. In 2007 a merger with Euronext, the largest stock exchange in Europe, led to the creation of NYSE Euronext, which was later acquired by Intercontinental Exchange, the current parent of the New York Stock Exchange.

The Dow Jones Industrial Average (DJIA) is an index that tracks 30 large, publicly-owned blue chip companies trading on the New York Stock Exchange (NYSE) and the NASDAQ. The Dow Jones derived from Charles Dow, who created the index back in 1896, along side with his business partner Edward Jones**.**

The key point about the DJIA is that it's not a weighted arithmetic average, nor does it represent its component companies market capitalisation as does the S&P 500. Rather, it reflects the sum of the price of 1 share of stock for all the components, divided by the divisor. Thus, a one-point move in any of the component stocks will move the index by an unifrom number of points.

$$\text{DJIA Price} = \frac{\text{SUM (Component stock prices)}}{\text{Dow Divisor}}$$

CHINA

The SSE Composite, short for the Shanghai Stock Exchange Composite Index, is a market composite made up of all the A-shares and B-shares that are traded on the Shanghai Stock Exchange. The index is calculated by using a base period of 100. The first day of reporting was July 15, 1991.

The composite figure can be calculated by using the formula:

$$\text{Current Index} = \frac{\text{Market Cap of Composite Members}}{\text{Base Period}} \times \text{BaseValue}$$

The SSE Composite is a good way to get a wide overview of the performance of companies listed on the Shanghai market. More selective indexes, such as the SSE 50 Index and SSE 180 Index, show market leaders by market capitalisation.

JAPAN

The Nikkei short for Japan's Nikkei 225 Stock Average, the leading and most-respected index of Japanese stocks. It is a price-weighted index composed of Japan's top 225 blue-chip companies traded on the Tokyo Stock Exchange. The Nikkei is equivalent to the Dow Jones Industrial Average Index in the United States.

The Nikkei was established as part of the rebuilding and industrialization of Japan in the aftermath of the 2nd World War. Stocks are ranked by share price, rather than by market capitalization as is common in most indexes. Valuations are denominated in Japanese yen. The composition of the index is reviewed every September, and any needed changes take place in October.

It is not possible to directly purchase an stock, but there are several exchange traded funds (ETFs) whose components correlate to the Nikkei. ETFs that track the Nikkei and trade on the Tokyo Stock Exchange include Blackrock Japan's iShares Nikkei 225 and Nomura Asset Management's Nikkei 225 Exchange Traded Fund. The MAXIS Nikkei 225 Index ETF is a dollar-denominated fund that trades on the NY Stock Exchange.

SOUTH KOREA

The Korea Stock Exchange is a division of the much larger Korea Exchange (KRX, or the Exchange). Previously, Korea's stock market was a standalone entity. In 2005, the Korea Stock Exchange added with the Korea Futures Exchange and the electronic market, KOSDAQ, to form the Korea Exchange.

The Exchange is the only securities exchange operator in South Korea, making markets in equities, bonds, stock index futures, stock index options, and equity options. KRX's headquarters are in Busan, and it has an office for cash markets and oversight is in Seoul.

In 2019, the KRX listed 2194 companies with a combined market capitalization (market cap) of $1.9 trillion. Normal trading sessions look the same as those of other major stock markets around the world. As with the S&P 500 in the United States, an index, the Korea Composite Stock Price Index (KOSPI) tracks the health of the Exchange.

LONDON

The Financial Times Stock Exchange Group (FTSE), also known by "Footsie," is an independent organization. It is similar to the S&P which specializes in creating index offerings for the global financial markets. An index will represent a market segment and is an hypothetical portfolio of stock holdings. The most well-known index, among many at FTSE, is FTSE 100, which is composed of blue-chip stocks listed on the LSE.

The London Stock Exchange Group (LSEG) owns the FTSE Group. LSEG is an London-based parent organization which also owns Russell Indexes, Borsa Italiana, MilenniumIT and other financial organizations. In May 2015, LSEG announced the merging of FTSE Group and Russell forming the new name FTSE Russell.

Market analysts, traders, and investors will follow the FTSE indices. The two most popular of the many indexes FTSE Russell oversees is the FTSE 100 and the Russell 2000. The FTSE 100 is arguably the popular and widely used stock exchange index in Europe. At its creation in January 1984, the index had a base level of 1,000.

ANNEXURE 2

Time Series Forecasting

A time series is a sequence where a metrix is recorded over regular time intervals.
Depending upon the frequency, a time series can be of yearly/annually (ex: annual budget), quarterly (ex: expenses), monthly (ex: air traffic), weekly (ex: sales qty), daily (ex: weather), hourly (ex: stocks price), minutes (ex: inbound calls in a call canter) and even seconds wise (ex: web traffic).Forecasting a time series can be broadly divided into two types

If we use only the previous values of the time series to predict its future values, it is known as **Univariate Time Series Forecasting**.

And if we use predictors other than the series to forecast it is known as **Multi Variate Time Series Forecasting**.

ARIMA Models 'AutoRegressive Integrated Moving Average', is a forecasting algorithm based on the idea that the knowledge in the past values of the time series can alone be used to predict the future outputs **.**Actually a class of models that explains a given time series based on its own past values, its own lags and the lagged forecast errors, so that equation can be used to forecast future values**.** Any 'non-seasonal' time series that exhibits patterns and is not a random white noise can be modeled with ARIMA models **.**

An ARIMA model is characterized by 3 terms: p, d, q

where,

p = order of the AR term is

q =order of the MA term

d = number of differencing required to make the time series stationary

If a time series, has seasonal patterns, then need to take seasonal terms and it becomes SARIMA, short for 'Seasonal ARIMA'.

First step to build an ARIMA model is to make the time series stationary or static.

Because, term 'Auto Regressive' in ARIMA means it is an linear regression model that uses its own lags as predictors. Linear regression models, works best when the predictors are not correlated and are independent of one another.

The value of d, is the minimum number of differencing(subtract the previous value from the current value) needed to make the series stationary. And if the series is already stationary, then d = 0.

p is the order of the 'Auto Regressive' (AR) term. It points the number of lags of Y to be used as predictors. And 'q' is order of the 'Moving Average' (MA) term. It shows the number of lagged forecast errors that should go into the ARIMA Model.

A pure **Auto Regressive (AR only) model** is one where Yt depends only on its own lags means, Yt is a function of the 'lags of Yt'.

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \epsilon_1$$

where, $Y*{t-1}$ *is the lag1 of the series,* $\beta1$ is the coefficient of lag1 that the model estimates and α is the intercept term, also estimated by the model.

Pure **Moving Average (MA only) model** is one where Yt depends only on the lagged forecast errors.

$$Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

where the error terms are the errors of the autoregressive models of the respective lags. The errors Et and E(t-1) are the errors from the following equations:

$$Y_t = \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_0 Y_0 + \epsilon_t$$

$$Y_{t-1} = \beta_1 Y_{t-2} + \beta_2 Y_{t-3} + .. + \beta_0 Y_0 + \epsilon_{t-1}$$

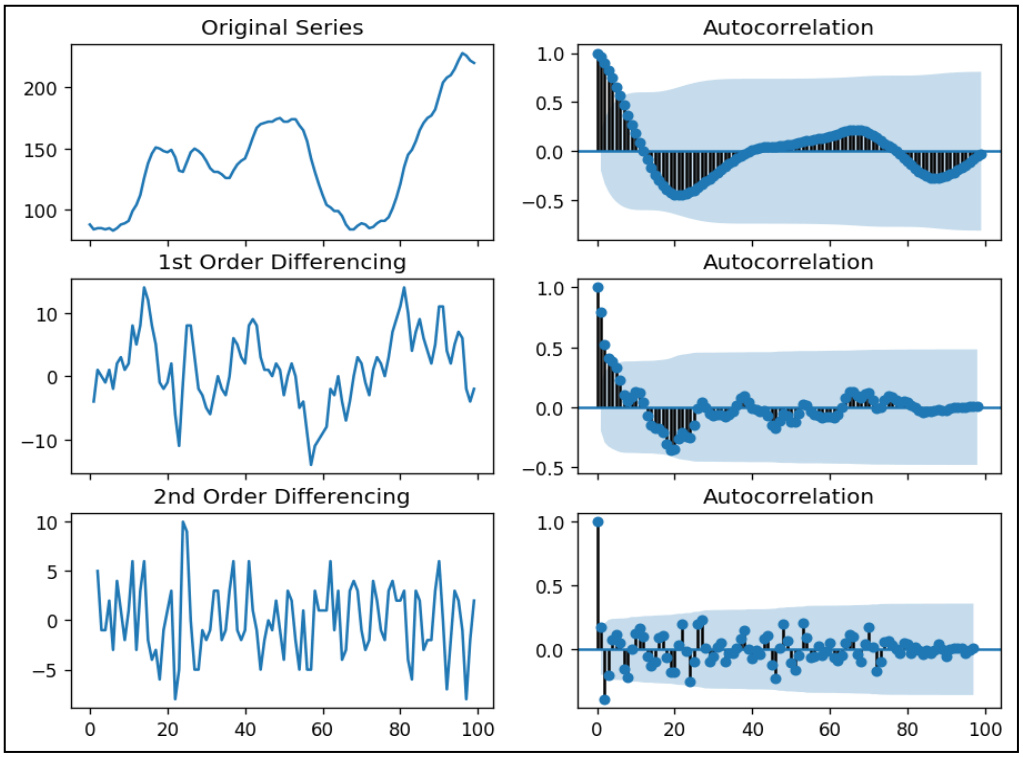
An ARIMA model is one where the time series was differenced at least one time to make it stationary and AR and MA terms combined gives the following equation:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

ARIMA model in words:

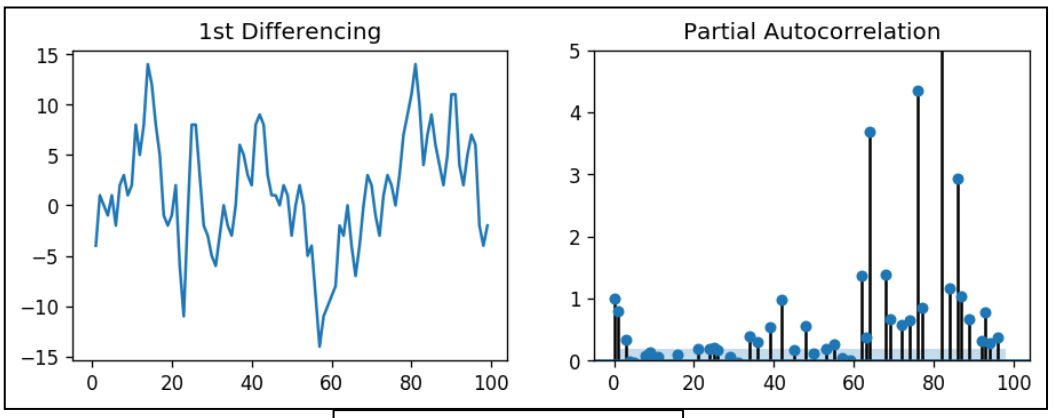
Predicted Yt = Constant + Linear combination Lags of Y (upto p lags) + Linear Combination of Lagged forecast errors (upto q lags)

The right order of differencing is the minimum differencing required to get a near-stationary series which moves around a defined mean and the ACF plot reaches to zero quickly.
If the autocorrelations are positive for many number of lags (10 or more), then the series needs further differencing. On the other hand, if the lag 1 autocorrelation itself is too negative, then the series is probably over - differenced.



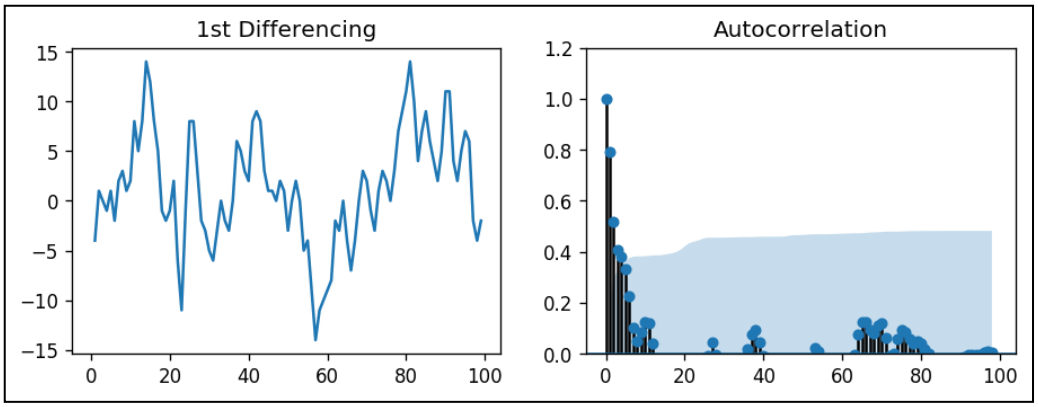
GRAPH 56

Partial autocorrelation (PACF) is the correlation between the series and its lag, after removing the contributions from the intermediate lags. So, PACF sort of conveys the pure correlation between a lag and series. In this way, we can found that if lag is needed in the AR term or not.
Partial autocorrelation of lag (k) of a series is the coefficient of that lag in the autoregression equation of Y.



GRAPH 57

PACF plot for the number of AR terms, can be looked at the ACF plot for the number of MA terms. An MA term is technically, the error of the lagged forecast.
The ACF tells how many MA terms are required to remove any autocorrelation in the stationarized series.



GRAPH 58

And if the series is slightly under differenced, adding one or more additional AR terms usually makes it well . Likewise, if it is slightly over-differenced, adding an additional MA term can help us.