In [1]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
from pandas.plotting import autocorrelation_plot
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_digits
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
file = pd.read_csv('China_record_update.csv')
file1 = pd.read_csv('Italy_record_update.csv')
file2 = pd.read_csv('India_record_update.csv')
```

In [3]:
```
file['date_parsed'] = pd.to_datetime(file['Date'], format = "%d/%m/%Y")
file['date_parsed'].head()
```

Out[3]:
```
0   2020-01-22
1   2020-01-29
2   2020-02-04
3   2020-02-11
4   2020-02-18
Name: date_parsed, dtype: datetime64[ns]
```

In [4]:
```
file1['date_parsed'] = pd.to_datetime(file1['Date'], format = "%d/%m/%Y")
file1['date_parsed'].head()
```

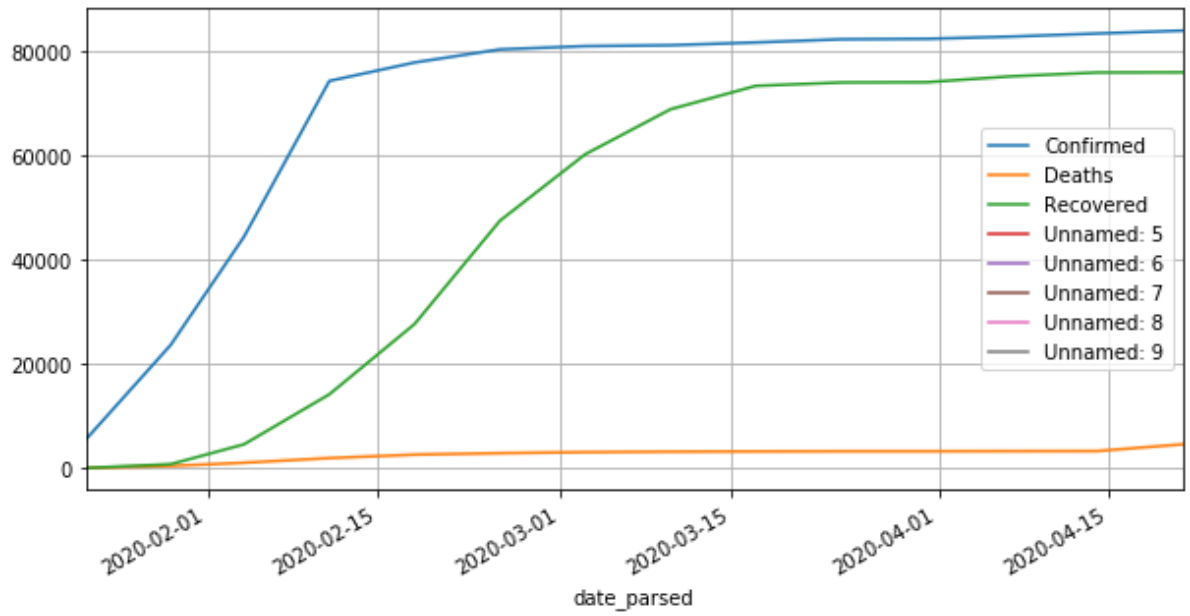
Out[4]:
```
0   2020-01-22
1   2020-01-29
2   2020-02-04
3   2020-02-11
4   2020-02-18
Name: date_parsed, dtype: datetime64[ns]
```

In [5]:
```
file2['date_parsed'] = pd.to_datetime(file2['Date'], format = "%d/%m/%Y")
file2['date_parsed'].head()
```

Out[5]:
```
0   2020-01-22
1   2020-01-29
2   2020-02-04
3   2020-02-11
4   2020-02-18
Name: date_parsed, dtype: datetime64[ns]
```

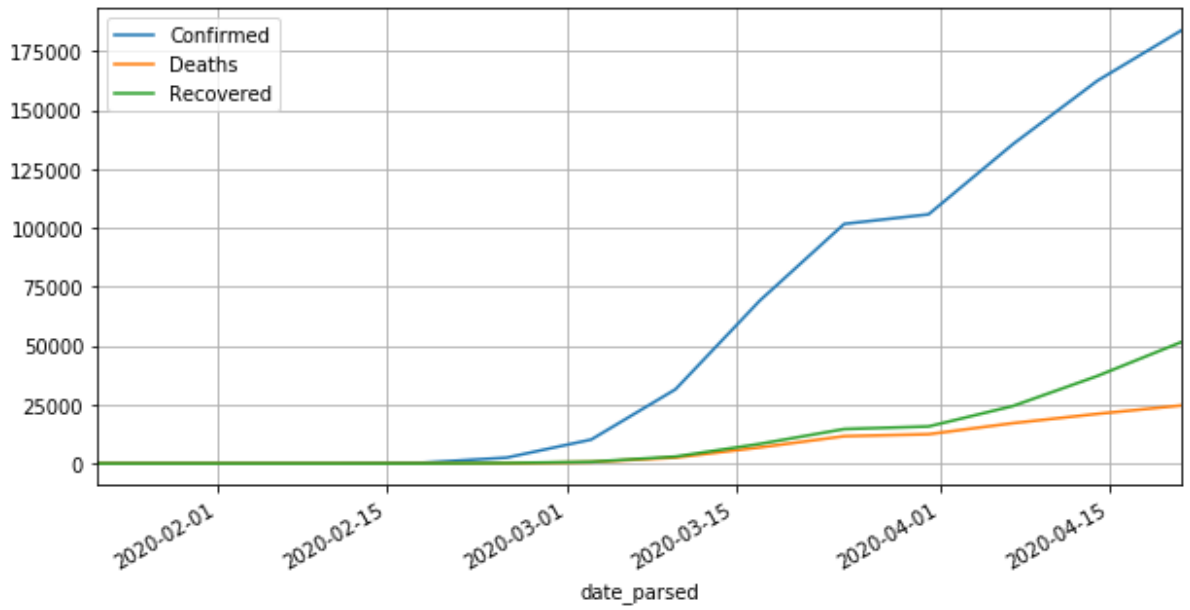
In [6]:
```
file.set_index('date_parsed').plot(figsize=(10,5), grid=True)
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x1bd7d2fae48>



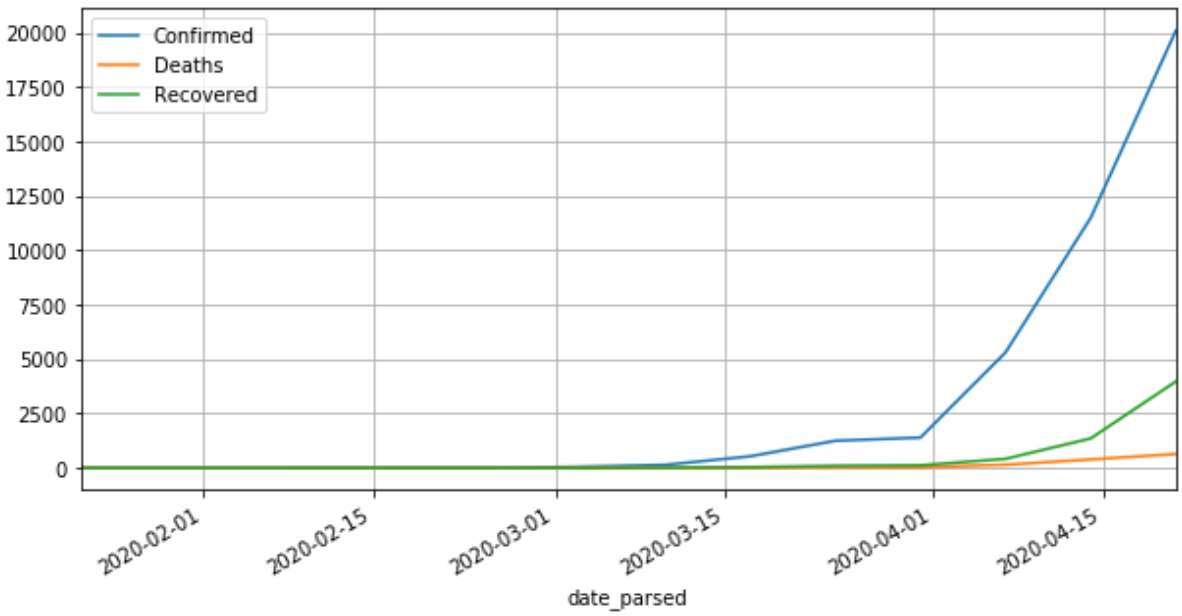
In [7]:
```
file1.set_index('date_parsed').plot(figsize=(10,5), grid=True)
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1bd02d56108>



In [8]:
```
file2.set_index('date_parsed').plot(figsize=(10,5), grid=True)
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd02dec2c8>
```



In [9]:
```
X = file[['Confirmed']]
Y = file2[['Confirmed']]
```

In [10]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, random_state=100)
```

In [11]:
```
lr = LinearRegression()
lr.fit(X_train, Y_train)
```

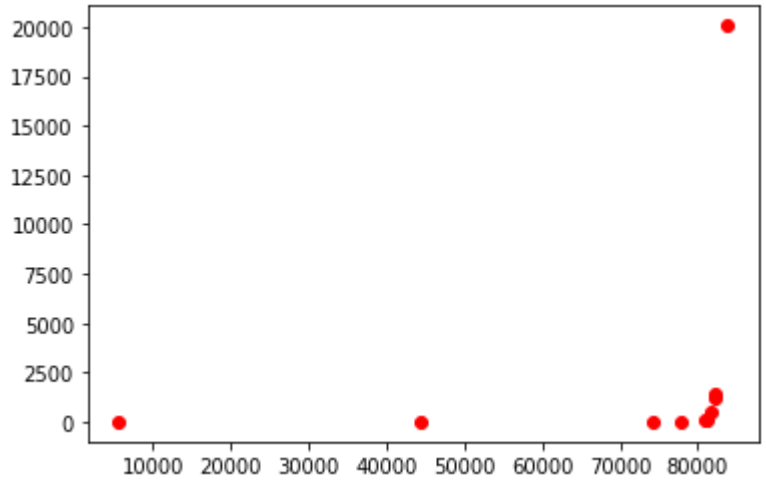
Out[11]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

In [12]:
```
Y_pred = lr.predict(X_test)
```

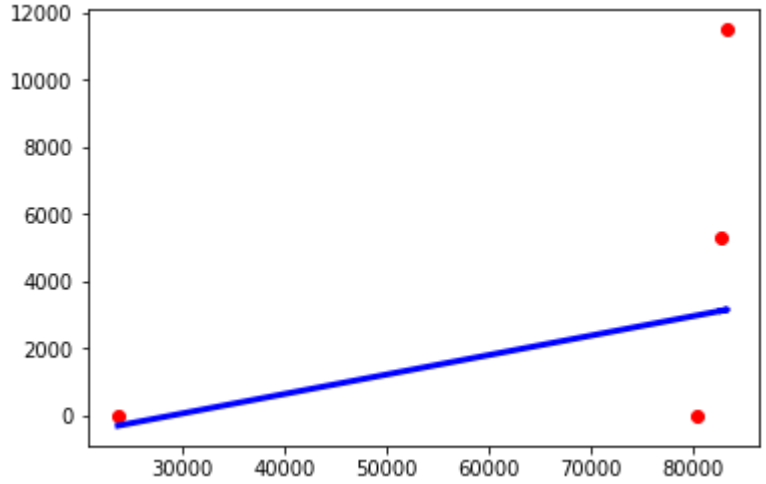
In [13]:
```
r2_score(Y_test, Y_pred)
```

Out[13]:
```
0.0713879451986319
```

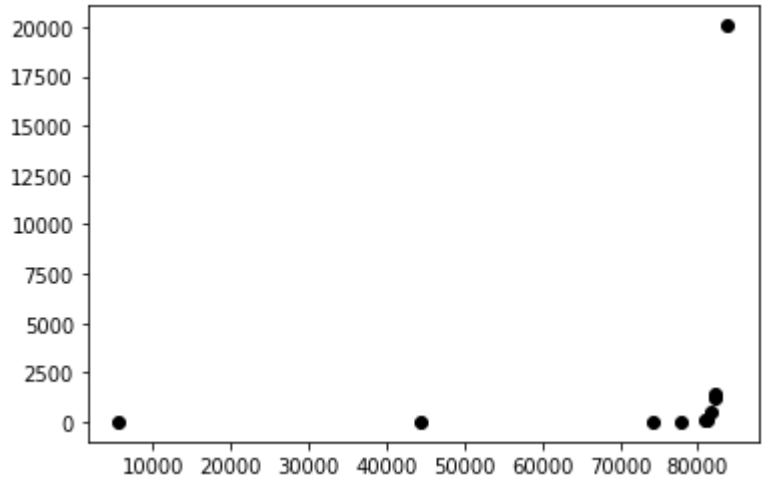
In [14]:
```
plt.scatter(X_train, Y_train, color='red', linewidth=1)
plt.show()
```



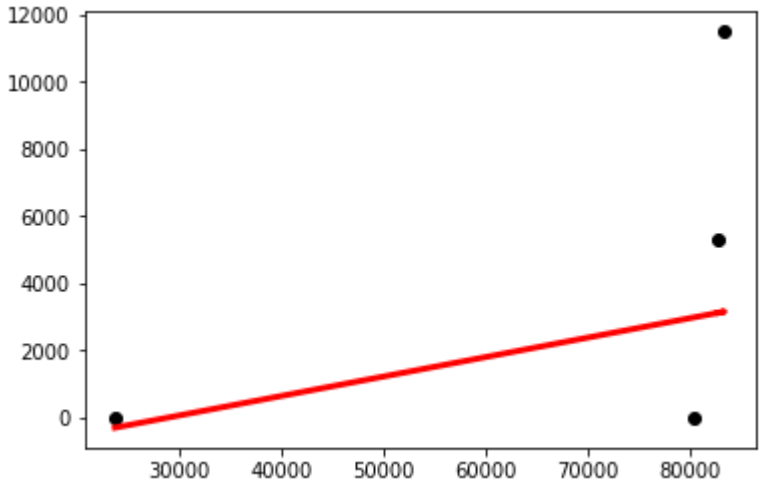
In [15]:
```
plt.scatter(X_test, Y_test,  color='red')
plt.plot(X_test, Y_pred, color='blue', linewidth=3)
plt.show()
```



In [16]:
```
plt.scatter(X_train, Y_train, color='black', linewidth=1)
plt.show()
```



In [17]:
```
plt.scatter(X_test, Y_test,  color='black')
plt.plot(X_test, Y_pred, color='red', linewidth=3)
plt.show()
```



In [18]:
```
X1 = file1[['Confirmed']]
Y1 = file2[['Confirmed']]
```

In [19]:
```
X1_train, X1_test, Y1_train, y1_test  = train_test_split(X1,Y1, random_state=1
00)
```

In [20]:
```
lr = LinearRegression()
lr.fit(X1_train, Y1_train)
```

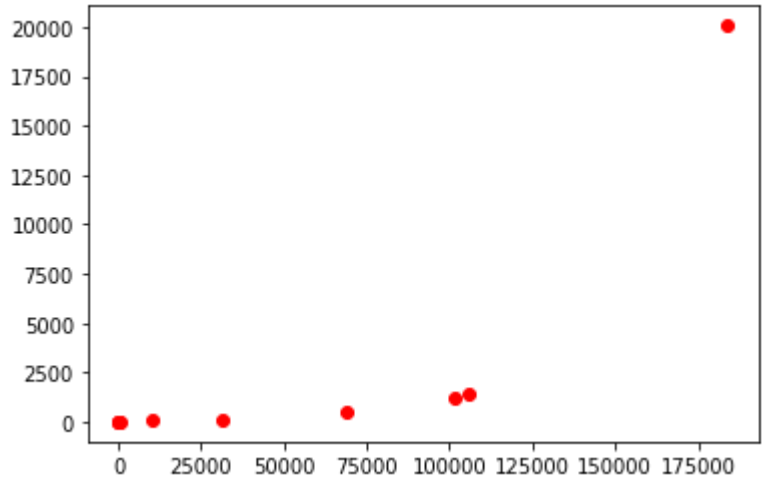
Out[20]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

In [21]:
```
Y1_pred = lr.predict(X1_test)
```

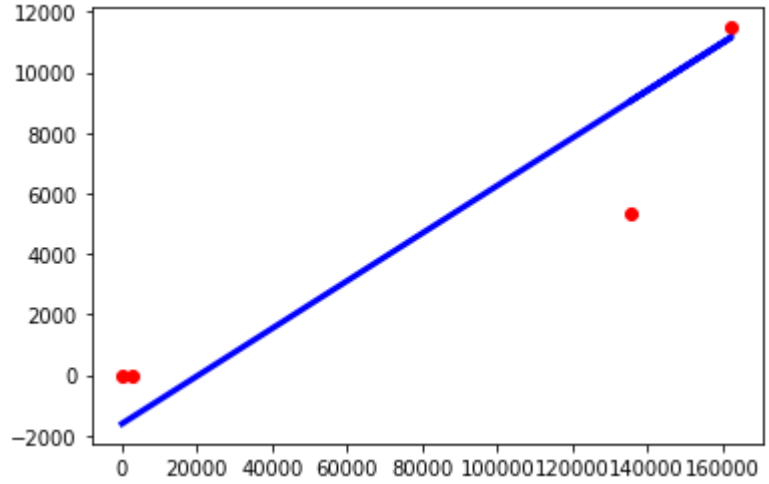
In [22]:
```
r2_score(y1_test, Y1_pred)
```

Out[22]:
```
0.7908875922483625
```

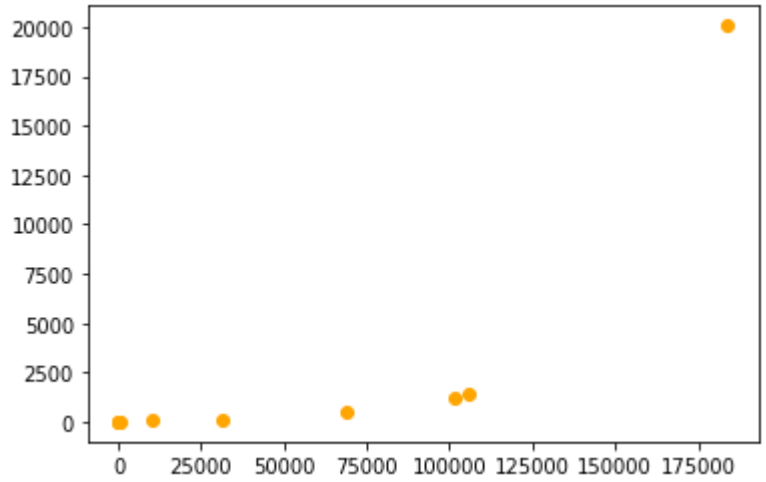
In [23]:
```
plt.scatter(X1_train, Y1_train, color='red', linewidth=1)
plt.show()
```



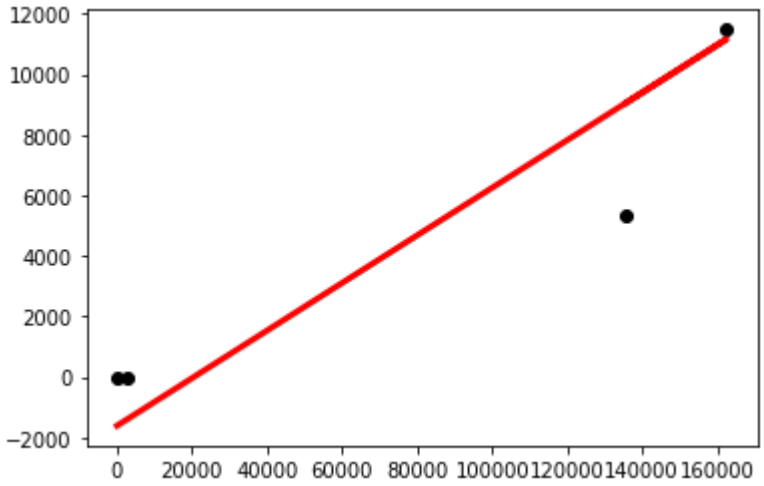
In [24]:
```
plt.scatter(X1_test, y1_test,  color='red')
plt.plot(X1_test, Y1_pred, color='blue', linewidth=3)
plt.show()
```



In [25]:
```
plt.scatter(X1_train, Y1_train, color='orange', linewidth=1)
plt.show()
```



In [26]:
```
plt.scatter(X1_test, y1_test,  color='black')
plt.plot(X1_test, Y1_pred, color='red', linewidth=3)
plt.show()
```



In [27]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, random_state=90)
```

In [28]:
```
LogModel = LogisticRegression()
LogModel.fit(X_train, Y_train)
```

Out[28]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [29]:
```
predictions = LogModel.predict(Y_test)
print(classification_report(X_test, predictions))
print(confusion_matrix(X_test, predictions))
print(accuracy_score(X_test, predictions))
```

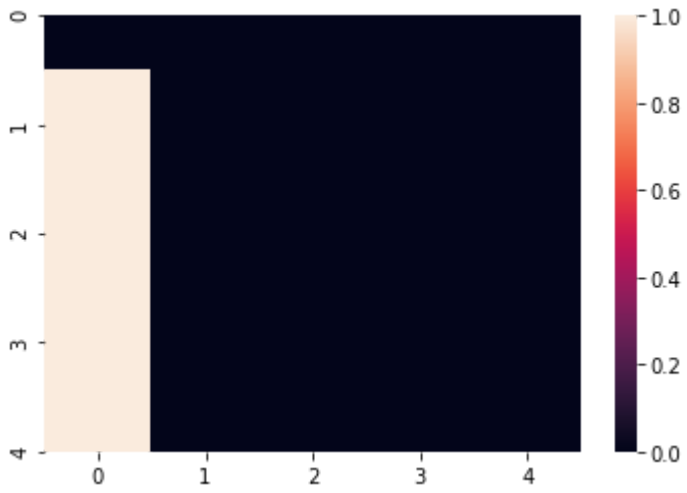
```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00       0.0
       23707       0.00      0.00      0.00       1.0
       80887       0.00      0.00      0.00       1.0
       81591       0.00      0.00      0.00       1.0
       82198       0.00      0.00      0.00       1.0

    accuracy                           0.00       4.0
   macro avg       0.00      0.00      0.00       4.0
weighted avg       0.00      0.00      0.00       4.0

[[0 0 0 0 0]
 [1 0 0 0 0]
 [1 0 0 0 0]
 [1 0 0 0 0]
 [1 0 0 0 0]]
0.0
```

In [30]:
```
sns.heatmap(pd.DataFrame(confusion_matrix(X_test,predictions)))
plt.show()
```



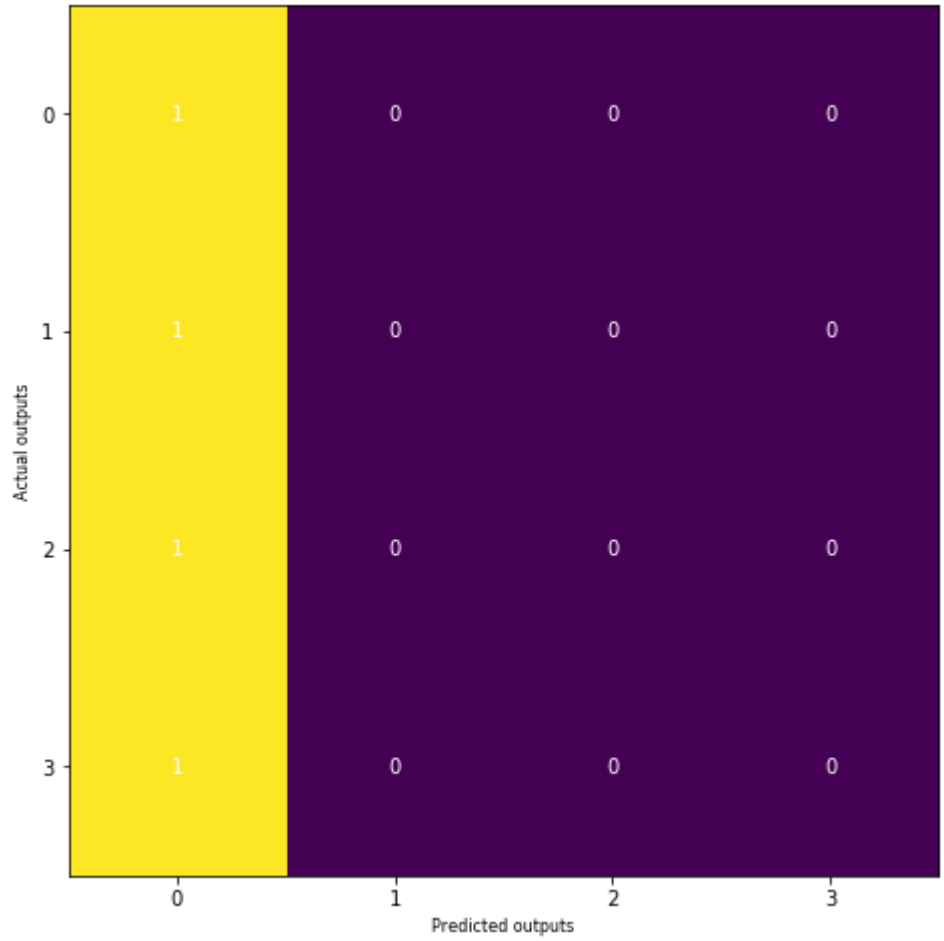
In [31]:
```
y_pred=LogModel.predict(X_test)
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(Y_test, y_pred)
cnf_matrix
```

Out[31]:
```
array([[1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0]], dtype=int64)
```

In [32]:
```
print("Accuracy:",metrics.accuracy_score(Y_test, y_pred))
```

```
Accuracy: 0.25
```

In [33]:
```
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cnf_matrix)
ax.set_xlabel('Predicted outputs', fontsize=8, color='black')
ax.set_ylabel('Actual outputs', fontsize=8, color='black')
ax.xaxis.set(ticks=range(4))
ax.yaxis.set(ticks=range(4))
ax.set_ylim(3.5, -0.5)
for i in range(4):
    for j in range(4):
        ax.text(j, i, cnf_matrix[i, j], ha='center', va='center', color='whit
e')
plt.show()
```



In [34]:
```
X1_train, X1_test, Y1_train, y1_test  = train_test_split(X1,Y1, random_state=9
0)
```

In [35]:
```
LogModel1 = LogisticRegression()
LogModel1.fit(X1_train, Y1_train)
```

Out[35]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [36]:
```
predictions = LogModel.predict(y1_test)
print(classification_report(X1_test, predictions))
print(confusion_matrix(X1_test, predictions))
print(accuracy_score(X1_test, predictions))
```

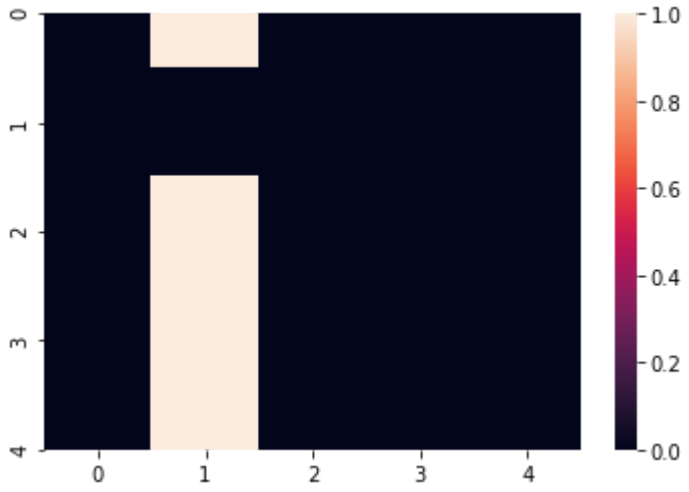
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       1.0
           3       0.00      0.00      0.00       0.0
       10149       0.00      0.00      0.00       1.0
       69176       0.00      0.00      0.00       1.0
      101739       0.00      0.00      0.00       1.0

    accuracy                           0.00       4.0
   macro avg       0.00      0.00      0.00       4.0
weighted avg       0.00      0.00      0.00       4.0

[[0 1 0 0 0]
 [0 0 0 0 0]
 [0 1 0 0 0]
 [0 1 0 0 0]
 [0 1 0 0 0]]
0.0
```

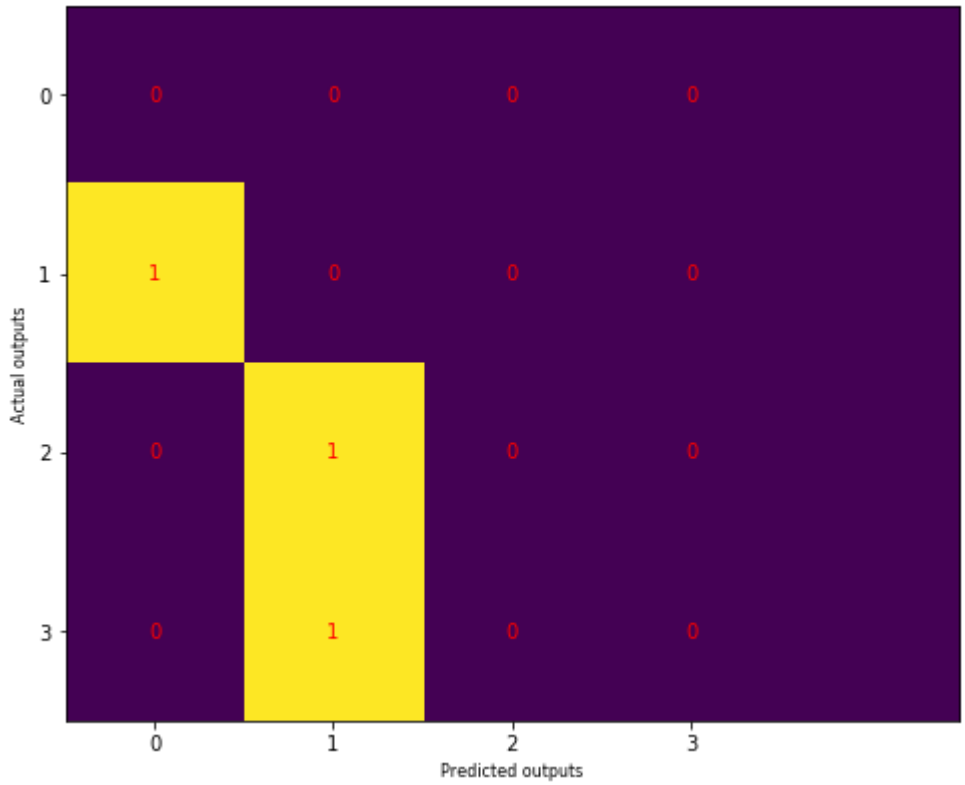
In [37]:
```
sns.heatmap(pd.DataFrame(confusion_matrix(X1_test,predictions)))
plt.show()
```



In [38]:
```
y1_pred=LogModel.predict(X1_test)
from sklearn import metrics
cnf_matrix1 = metrics.confusion_matrix(y1_test, y1_pred)
cnf_matrix1
```

Out[38]:
```
array([[0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0],
       [0, 1, 0, 0, 0],
       [0, 1, 0, 0, 0]], dtype=int64)
```

In [39]:
```
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cnf_matrix1)
ax.set_xlabel('Predicted outputs', fontsize=8, color='black')
ax.set_ylabel('Actual outputs', fontsize=8, color='black')
ax.xaxis.set(ticks=range(4))
ax.yaxis.set(ticks=range(4))
ax.set_ylim(3.5, -0.5)
for i in range(4):
    for j in range(4):
        ax.text(j, i, cnf_matrix1[i, j], ha='center', va='center', color='red'
)
plt.show()
```



In [40]:
```
print("Accuracy:",metrics.accuracy_score(y1_test, y1_pred))
```

```
Accuracy: 0.0
```

In [41]:
```
file2['Date'] = file2['Date'].astype('datetime64[ns]')
```

In [42]:
```
file2.head(6)
```

Out[42]:

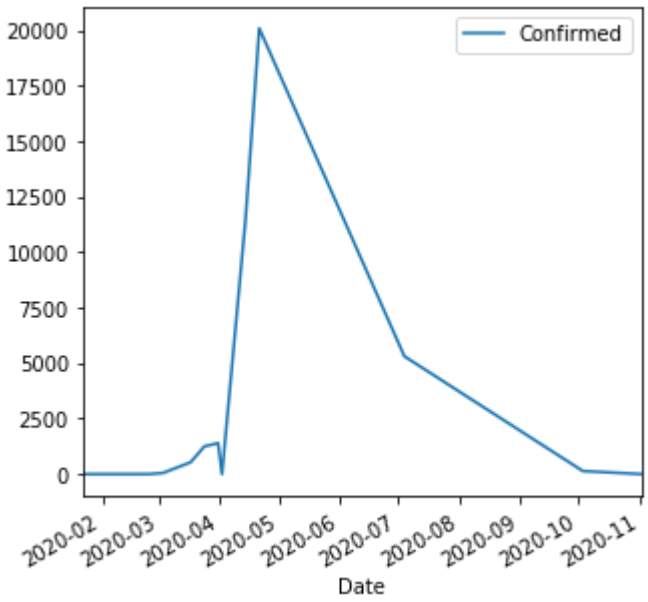
	Country	Date	Confirmed	Deaths	Recovered	date_parsed
0	India	2020-01-22	0	0	0	2020-01-22
1	India	2020-01-29	3	0	0	2020-01-29
2	India	2020-04-02	3	0	0	2020-02-04
3	India	2020-11-02	3	0	3	2020-02-11
4	India	2020-02-18	3	0	3	2020-02-18
5	India	2020-02-25	5	0	3	2020-02-25

In [43]:
```
file2.Date[1]
```

Out[43]:
```
Timestamp('2020-01-29 00:00:00')
```

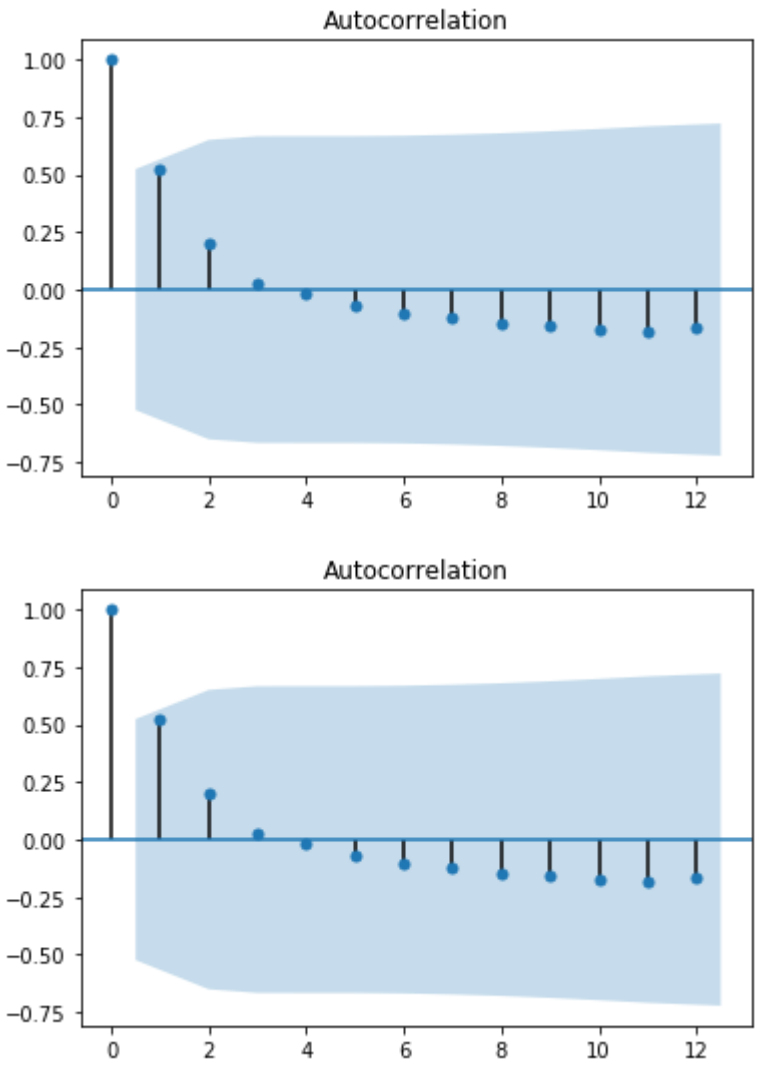
In [44]:
```
file2[['Date', 'Confirmed']].plot('Date', figsize=(5,5))
```

Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd034566c8>
```



In [45]:
```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(file2['Confirmed'])
```

Out[45]:



In [46]:
```
file2['Confirmed'].shift(1)
```

Out[46]:
```
0         NaN
1         0.0
2         3.0
3         3.0
4         3.0
5         3.0
6         5.0
7        56.0
8       142.0
9       536.0
10     1251.0
11     1397.0
12     5311.0
13    11487.0
Name: Confirmed, dtype: float64
```

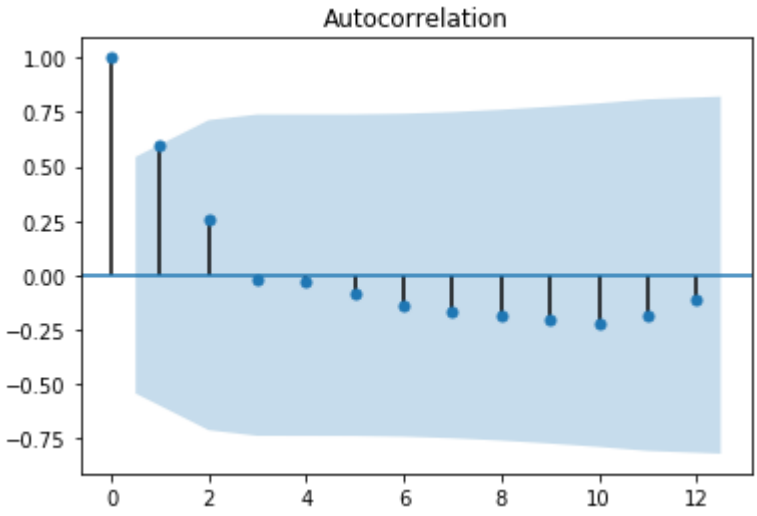
In [47]:
```
covid_diff = (file2['Confirmed'].diff(periods=1))
```

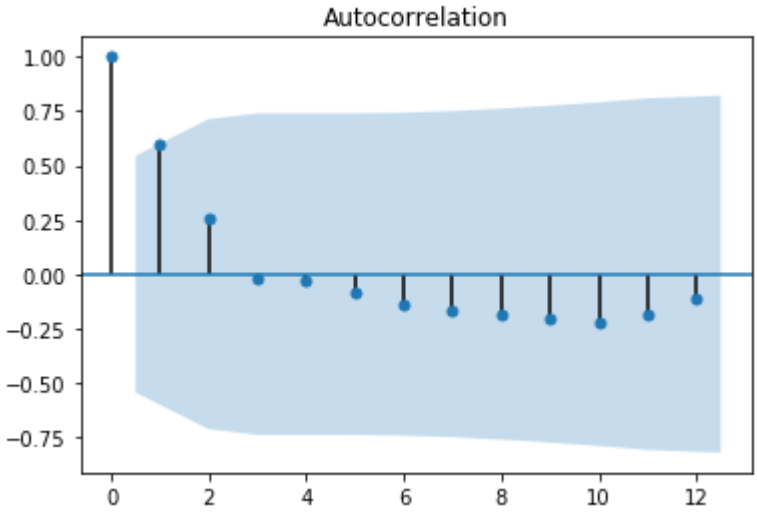
In [48]:
```
covid_diff = covid_diff[1:]
covid_diff.head(13)
```

Out[48]:
```
1        3.0
2        0.0
3        0.0
4        0.0
5        2.0
6       51.0
7       86.0
8      394.0
9      715.0
10     146.0
11    3914.0
12    6176.0
13    8593.0
Name: Confirmed, dtype: float64
```

In [49]:
```
plot_acf(covid_diff)
```

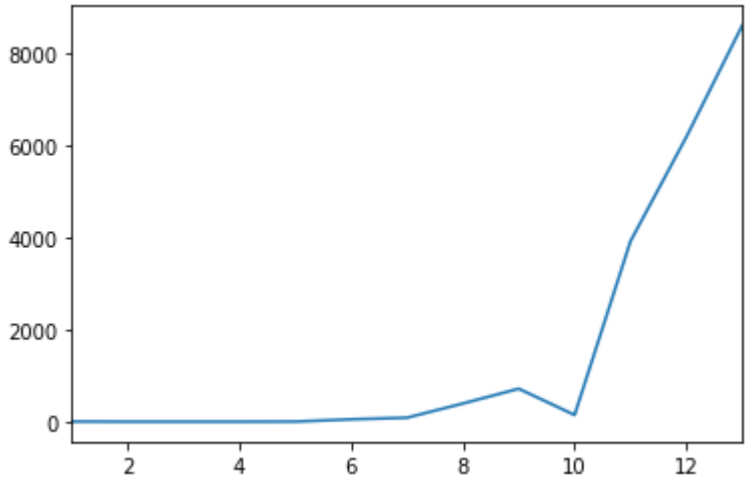
Out[49]:





In [50]:
```
covid_diff.plot()
```

Out[50]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd032be9c8>
```



In [51]:
```
Z = file2['Confirmed'].values
train = Z[0:11]
test = Z[11:]
predictions = []
```

In [52]:
```
test
```

Out[52]:
```
array([ 5311, 11487, 20080], dtype=int64)
```

In [53]:
```
from statsmodels.tsa.arima_model import ARIMA
```

In [54]:
```
model_arima = ARIMA(train,order=(1,2,0))
model_arima_fit = model_arima.fit()
print(model_arima_fit.aic)
```

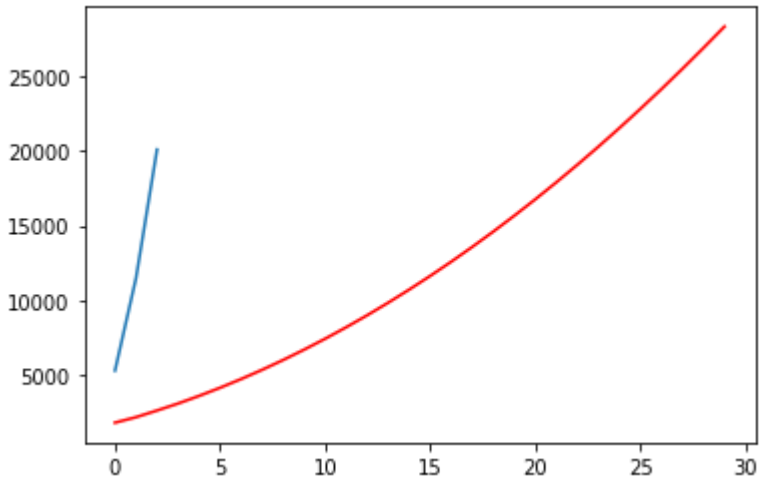
```
129.6501584055624
```

In [55]:
```
predictions = model_arima_fit.forecast(steps=30)[0]
predictions
```

Out[55]:
```
array([ 1830.38488644,  2197.07015703,  2643.4570815 ,  3109.01397423,
        3618.76852091,  4162.37266921,  4744.10496842,  5362.19639153,
        6017.3783679 ,  6709.34847768,  7438.23176063,  8203.9765173 ,
        9006.60412357,  9846.10574126, 10722.48502465, 11635.74046282,
       12585.87268049, 13572.88141936, 14596.76678623, 15657.52873693,
       16755.16728973, 17889.68243708, 19061.07418209, 20269.34252348,
       21514.48746179, 22796.50899678, 24115.40712856, 25471.18185709,
       26863.83318237, 28293.3611044 ])
```

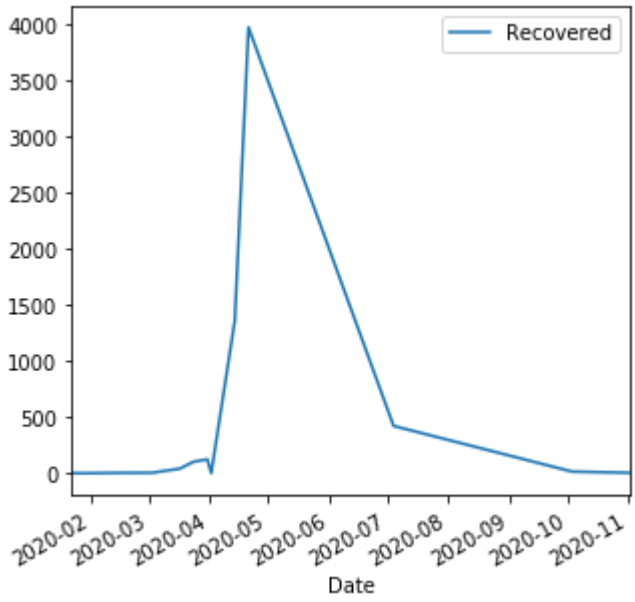
In [56]:
```
plt.plot(test)
plt.plot(predictions,color='red')
```

Out[56]:
```
[<matplotlib.lines.Line2D at 0x1bd04510988>]
```



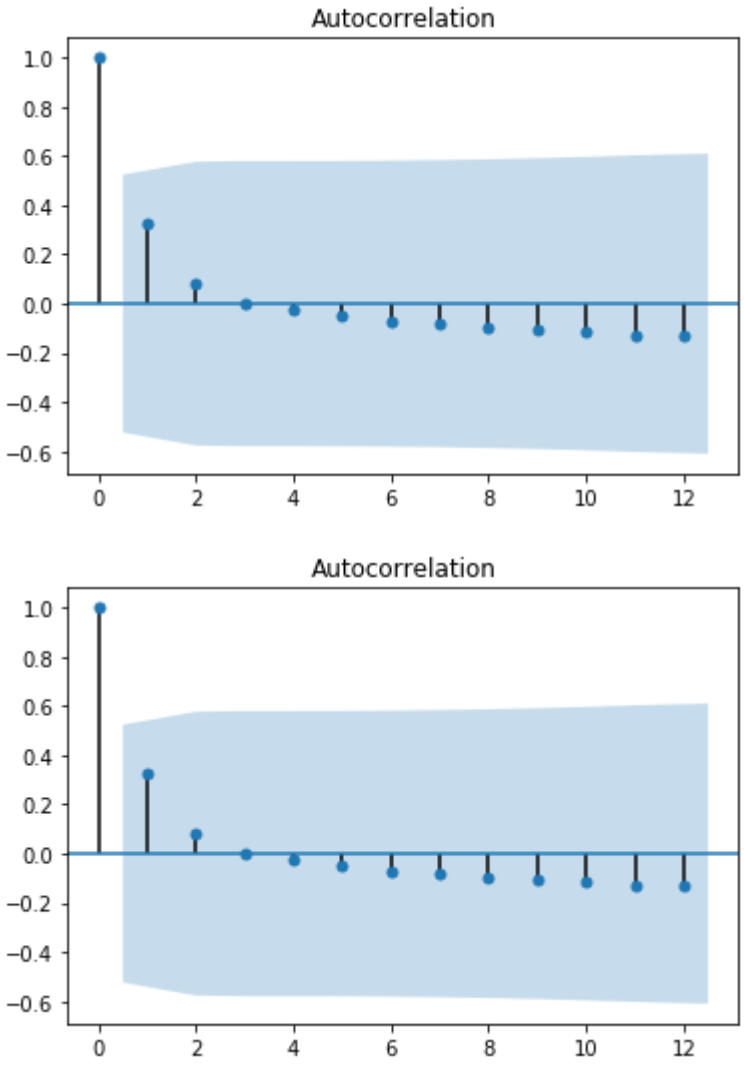
In [57]:
```
file2[['Date', 'Recovered']].plot('Date', figsize=(5,5))
```

Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd04554e08>
```



In [58]:
```
plot_acf(file2['Recovered'])
```

Out[58]:



In [59]:
```
file2['Recovered'].shift(1)
```

Out[59]:
```
0        NaN
1        0.0
2        0.0
3        0.0
4        3.0
5        3.0
6        3.0
7        4.0
8       14.0
9       40.0
10     102.0
11     123.0
12     421.0
13    1359.0
Name: Recovered, dtype: float64
```

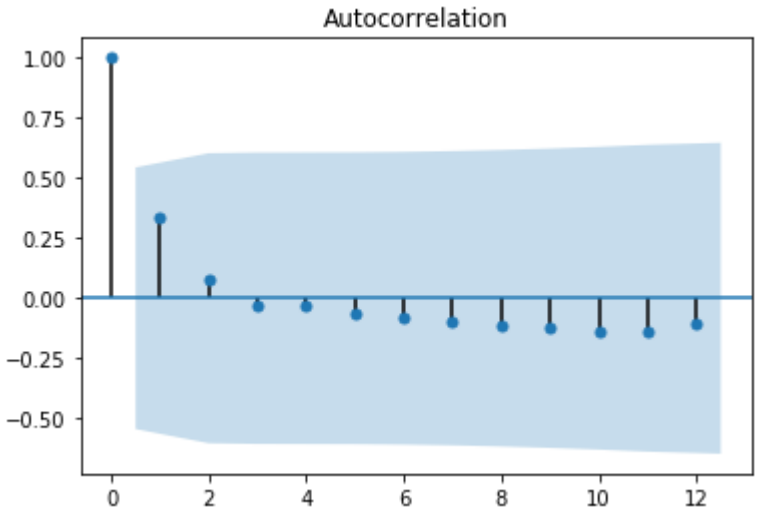
In [60]:
```
covid_diff1 = (file2['Recovered'].diff(periods=1))
```

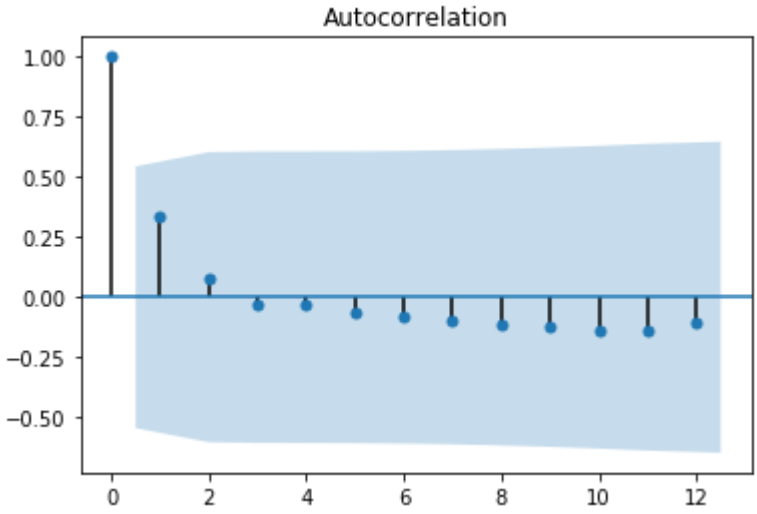
In [61]:
```
covid_diff1 = covid_diff1[1:]
covid_diff1.head(13)
```

Out[61]:
```
1        0.0
2        0.0
3        3.0
4        0.0
5        0.0
6        1.0
7       10.0
8       26.0
9       62.0
10      21.0
11     298.0
12     938.0
13    2616.0
Name: Recovered, dtype: float64
```

In [62]:
```
plot_acf(covid_diff1)
```

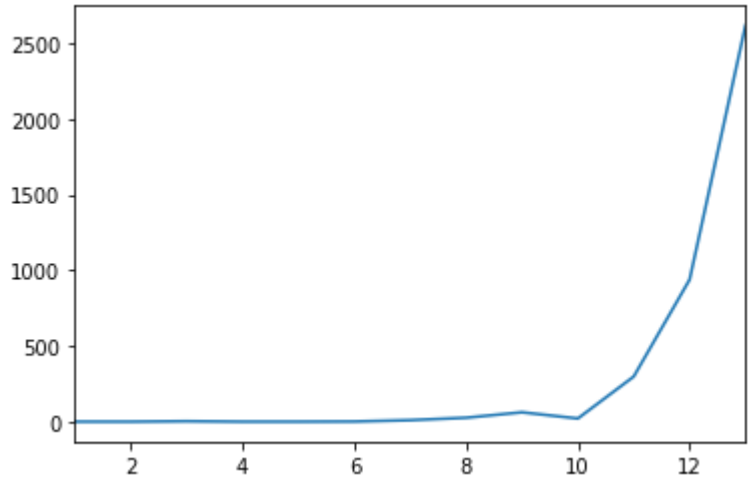
Out[62]:





In [63]:
```
covid_diff1.plot()
```

Out[63]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd046f72c8>
```



In [64]:
```
Z1 = file2['Recovered'].values
train = Z1[0:11]
test = Z1[11:]
predictions1 = []
```

In [65]:
```
from statsmodels.tsa.arima_model import ARIMA
```

In [66]:
```
model_arima1 = ARIMA(train,order=(1,2,0))
model_arima_fit1 = model_arima1.fit()
print(model_arima_fit1.aic)
```

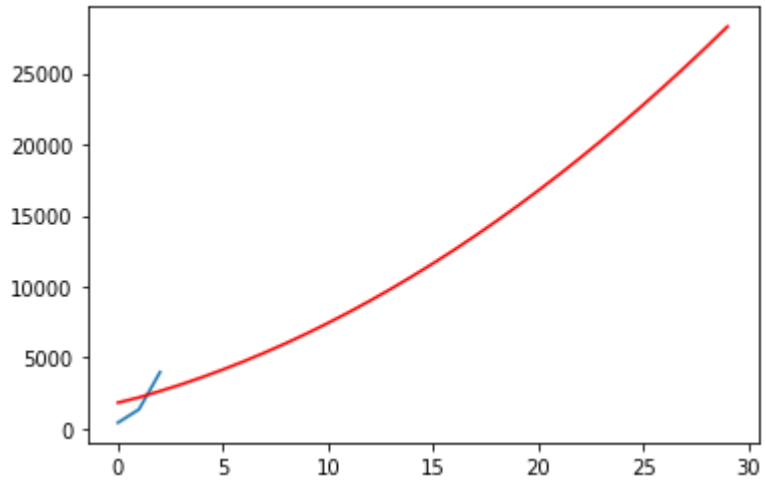
```
83.01158695029116
```

In [67]:
```
predictions1 = model_arima_fit1.forecast(steps=30)[0]
predictions1
```

Out[67]:
```
array([ 174.41243925,  215.19369039,  268.93319197,  322.07313618,
        382.40574547,  445.45251333,  513.78741921,  585.93109266,
        662.73378792,  743.70682936,  829.1310787 ,  918.84511332,
       1012.94170935, 1111.36754458, 1214.15326545, 1321.2812582 ,
       1432.76164617, 1548.58861108, 1668.76549692, 1793.29038177,
       1922.16437023, 2055.38682745, 2192.95811829, 2334.87803306,
       2481.14669228, 2631.76402668, 2786.73007607, 2946.04481757,
       3109.70826433, 3277.72040879])
```

In [68]:
```
plt.plot(test)
plt.plot(predictions,color='red')
```

Out[68]:
```
[<matplotlib.lines.Line2D at 0x1bd04756c48>]
```



In [69]:
```
X2 = file2[['Confirmed']]
Y2 = file2[['Deaths']]
```

In [70]:
```
X2_train, X2_test, Y2_train, y2_test  = train_test_split(X2,Y2, random_state=1
0000)
```

In [71]:
```
LogModel2 = LogisticRegression()
LogModel2.fit(X2_train, Y2_train)
```

Out[71]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [72]:
```
file2.describe()
```

Out[72]:

	Confirmed	Deaths	Recovered
count	14.000000	14.000000	14.000000
mean	2876.928571	90.571429	431.928571
std	5892.090558	192.251324	1082.740360
min	0.000000	0.000000	0.000000
25%	3.000000	0.000000	3.000000
50%	99.000000	1.500000	9.000000
75%	1360.500000	34.250000	117.750000
max	20080.000000	645.000000	3975.000000

In [73]:
```
predictions2 = LogModel2.predict(y2_test)
print(classification_report(X2_test, predictions2))
print(confusion_matrix(X2_test, predictions2))
print(accuracy_score(X2_test, predictions2))
```

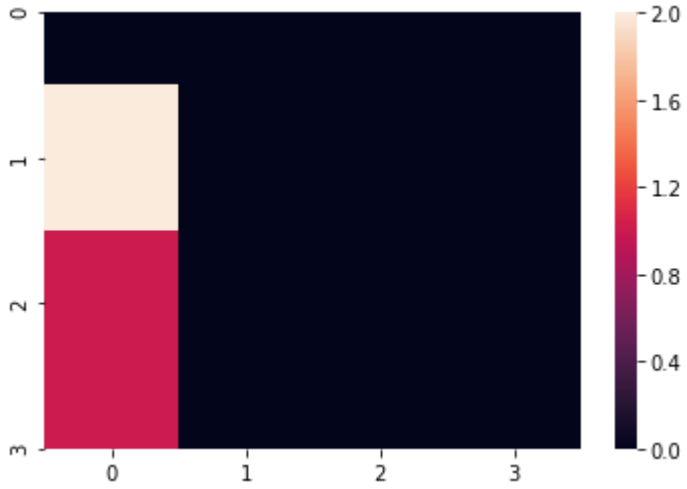
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       0.0
           3       0.00      0.00      0.00       2.0
          56       0.00      0.00      0.00       1.0
        1397       0.00      0.00      0.00       1.0

    accuracy                           0.00       4.0
   macro avg       0.00      0.00      0.00       4.0
weighted avg       0.00      0.00      0.00       4.0

[[0 0 0 0]
 [2 0 0 0]
 [1 0 0 0]
 [1 0 0 0]]
0.0
```

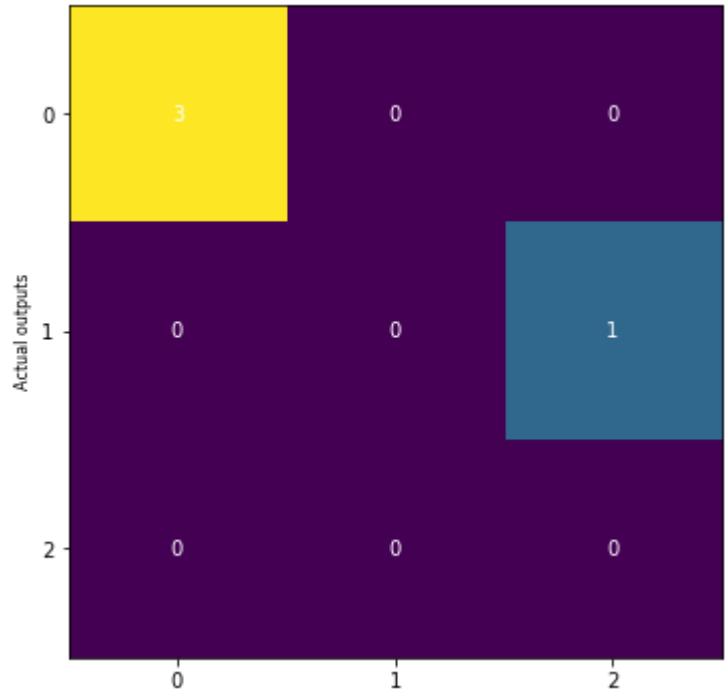
In [74]:
```
sns.heatmap(pd.DataFrame(confusion_matrix(X2_test,predictions2)))
plt.show()
```



In [75]:
```
y2_pred=LogModel2.predict(X2_test)
from sklearn import metrics
cnf_matrix2 = metrics.confusion_matrix(y2_test, y2_pred)
cnf_matrix2
```

Out[75]:
```
array([[3, 0, 0],
       [0, 0, 1],
       [0, 0, 0]], dtype=int64)
```

In [76]:
```
fig, ax = plt.subplots(figsize=(6, 6))
ax.imshow(cnf_matrix2)
ax.set_ylabel('Actual outputs', fontsize=8, color='black')
ax.xaxis.set(ticks=range(3))
ax.yaxis.set(ticks=range(3))
ax.set_ylim(2.5, -0.5)
for i in range(3):
    for j in range(3):
        ax.text(j, i, cnf_matrix2[i, j], ha='center', va='center', color='whit
e')
plt.show()
```



In [77]:
```
print("Accuracy:",metrics.accuracy_score(y2_test, y2_pred))
```

```
Accuracy: 0.75
```

In [78]:
```
X3 = file2[['Confirmed']]
Y3 = file2[['Recovered']]
```

In [79]:
```
X3_train, X3_test, Y3_train, y3_test  = train_test_split(X3,Y3, random_state=1
000)
```

In [80]:
```
LogModel3 = LogisticRegression()
LogModel3.fit(X3_train, Y3_train)
```

Out[80]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [81]:
```
predictions3 = LogModel.predict(y3_test)
print(classification_report(X3_test, predictions3))
print(confusion_matrix(X3_test, predictions3))
print(accuracy_score(X3_test, predictions3))
```

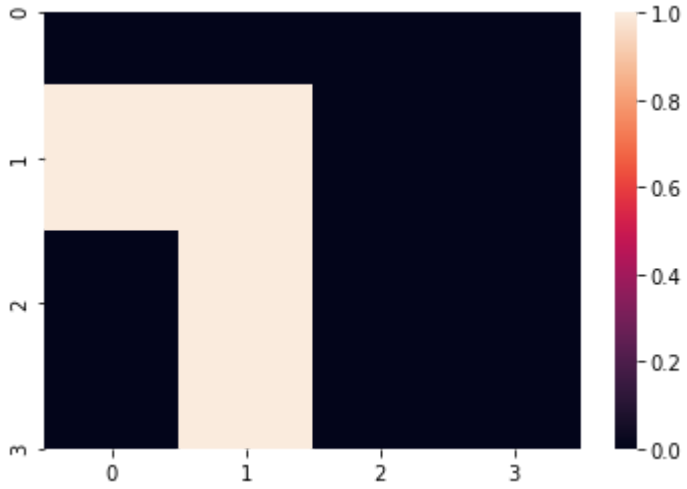
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           3       0.33      0.50      0.40         2
          56       0.00      0.00      0.00         1
       20080       0.00      0.00      0.00         1

    accuracy                           0.25         4
   macro avg       0.08      0.12      0.10         4
weighted avg       0.17      0.25      0.20         4

[[0 0 0 0]
 [1 1 0 0]
 [0 1 0 0]
 [0 1 0 0]]
0.25
```

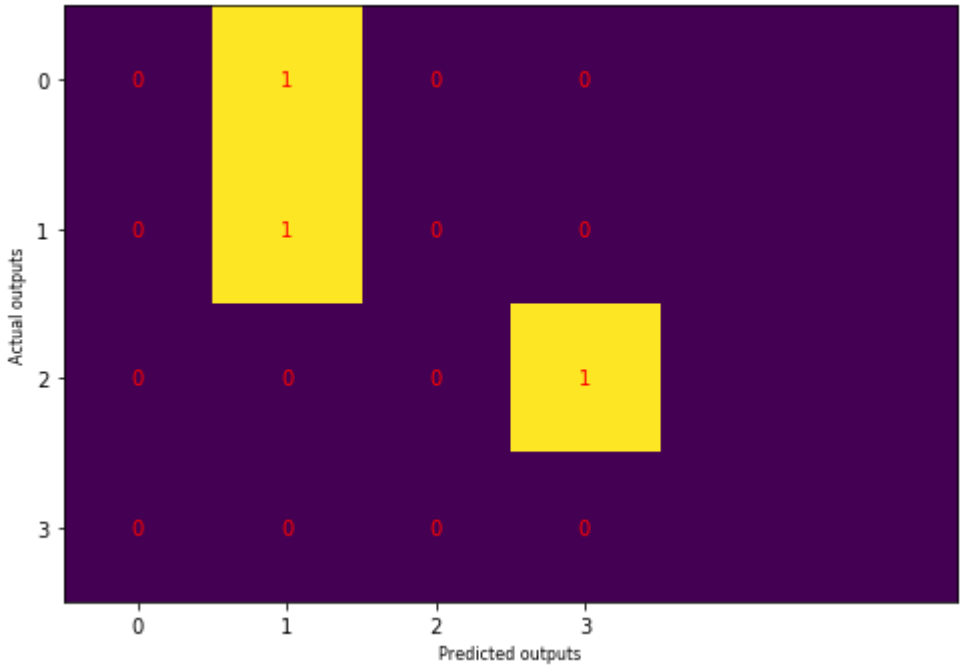
In [82]:
```
sns.heatmap(pd.DataFrame(confusion_matrix(X3_test,predictions3)))
plt.show()
```



In [83]:
```
y3_pred=LogModel3.predict(X3_test)
from sklearn import metrics
cnf_matrix3 = metrics.confusion_matrix(y3_test, y3_pred)
cnf_matrix3
```

Out[83]:
```
array([[0, 1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0]], dtype=int64)
```

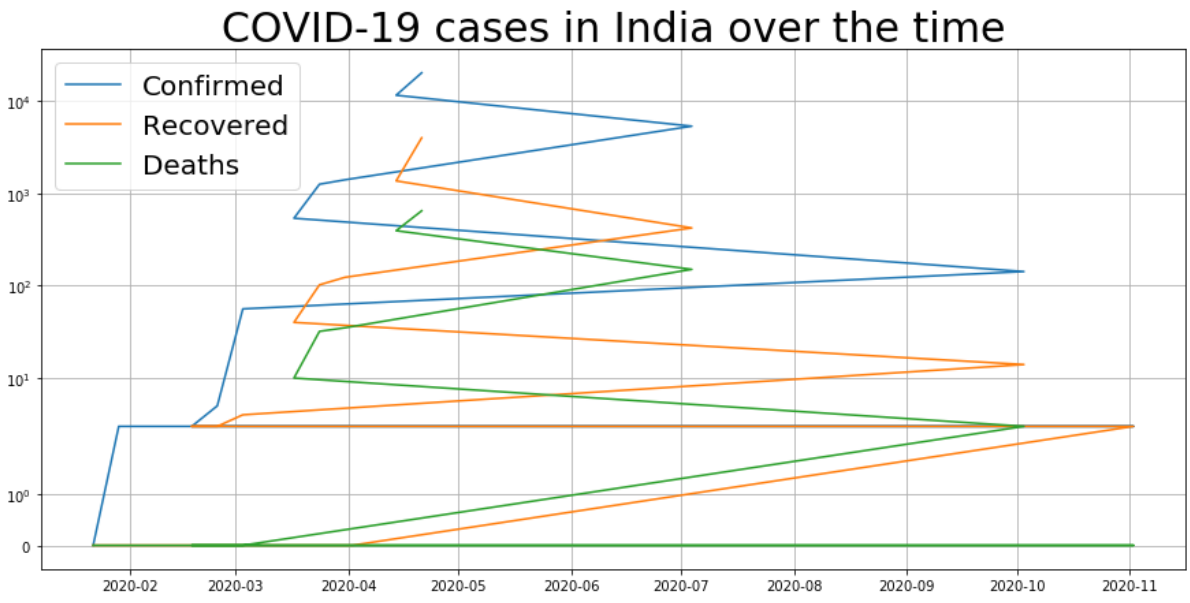
In [84]:
```
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cnf_matrix3)
ax.set_xlabel('Predicted outputs', fontsize=8, color='black')
ax.set_ylabel('Actual outputs', fontsize=8, color='black')
ax.xaxis.set(ticks=range(4))
ax.yaxis.set(ticks=range(4))
ax.set_ylim(3.5, -0.5)
for i in range(4):
    for j in range(4):
        ax.text(j, i, cnf_matrix3[i, j], ha='center', va='center', color='red'
)
plt.show()
```



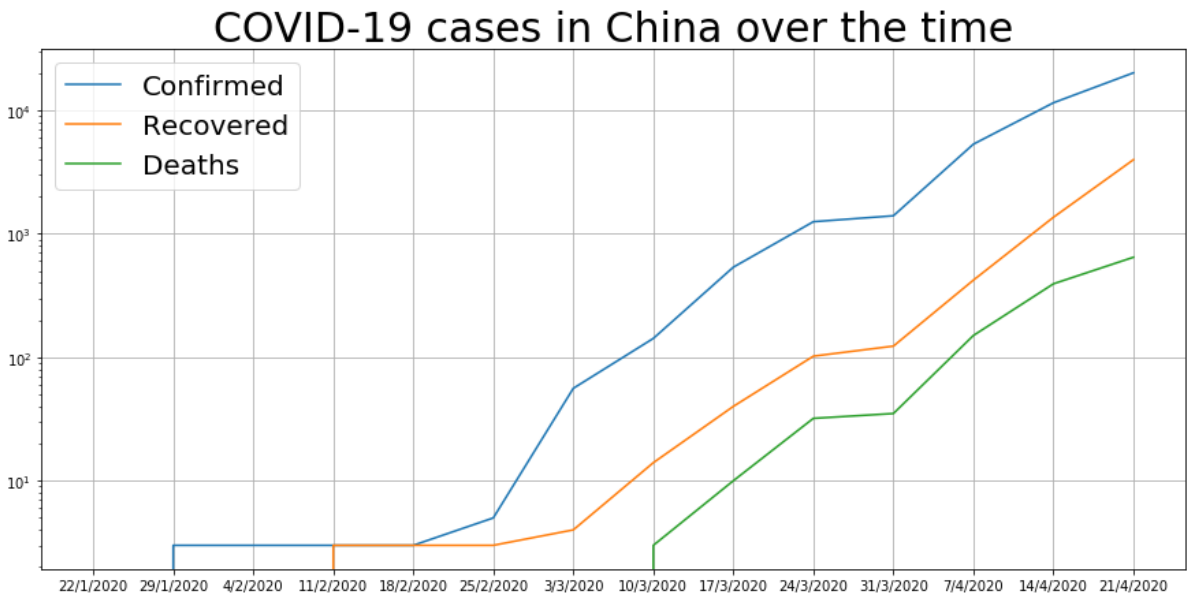
In [85]:
```
print("Accuracy:",metrics.accuracy_score(y3_test, y3_pred))
```

```
Accuracy: 0.25
```

In [86]:
```
plt.figure(figsize=(15,7))
plt.plot(file2['Date'], file2['Confirmed'])
plt.plot(file2['Date'], file2['Recovered'])
plt.plot(file2['Date'], file2['Deaths'])
plt.grid()
plt.yscale("symlog")
plt.legend(['Confirmed', 'Recovered', 'Deaths'], loc='upper left', fontsize=20
)
plt.title('COVID-19 cases in India over the time', fontsize=30)
plt.show()
```

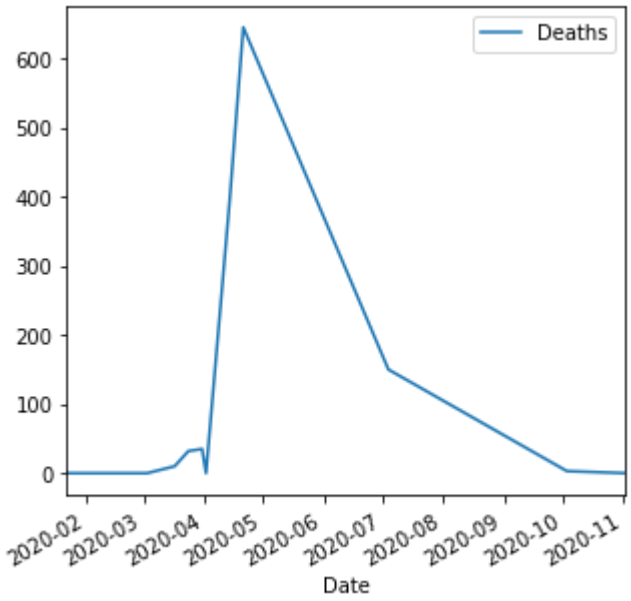


In [87]:
```
plt.figure(figsize=(15,7))
plt.plot(file['Date'], file2['Confirmed'])
plt.plot(file['Date'], file2['Recovered'])
plt.plot(file['Date'], file2['Deaths'])
plt.grid()
plt.yscale("log")
plt.legend(['Confirmed', 'Recovered', 'Deaths'], loc='upper left', fontsize=20
)
plt.title('COVID-19 cases in China over the time', fontsize=30)
plt.show()
```



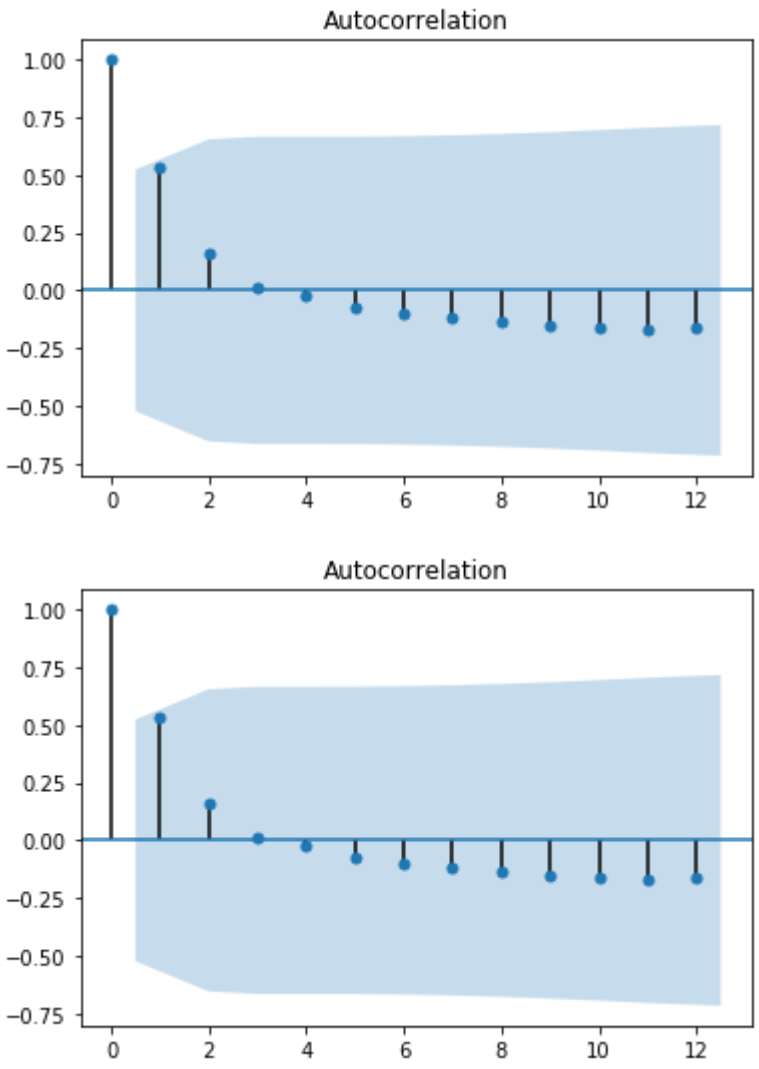
In [88]:
```
file2[['Date', 'Deaths']].plot('Date', figsize=(5,5))
```

Out[88]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd051e1bc8>
```



In [89]:
```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(file2['Deaths'])
```

Out[89]:



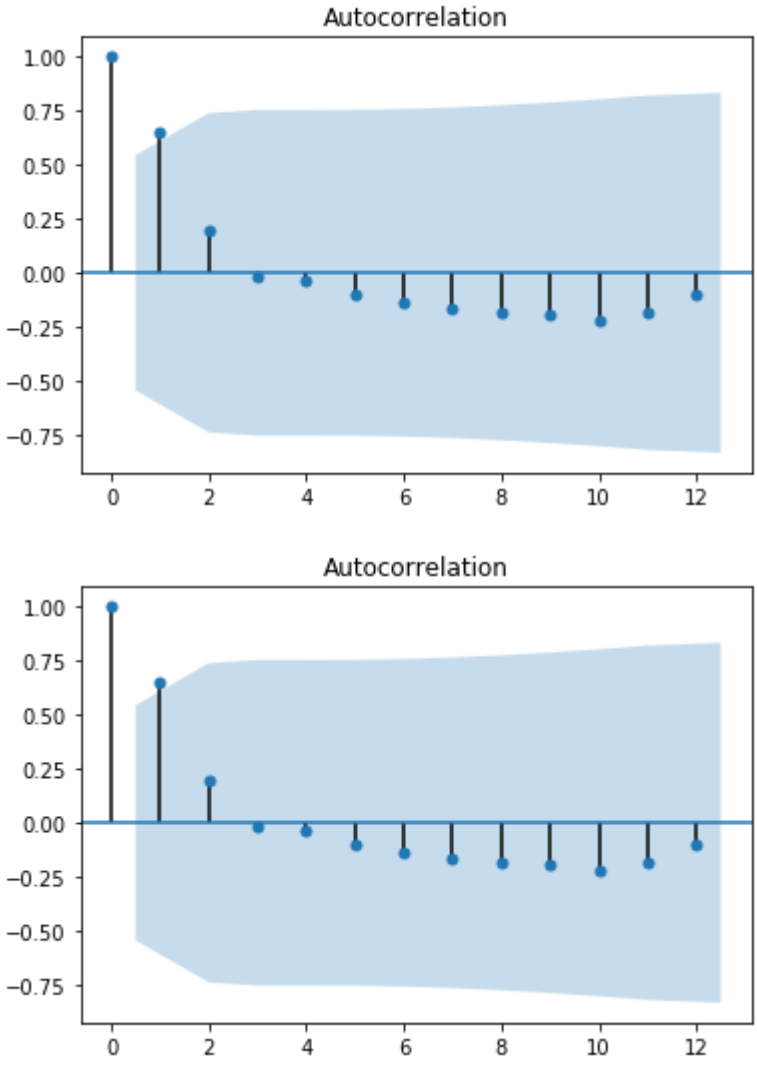
In [91]:
```
covid_diff2 = (file2['Deaths'].diff(periods=1))
```

In [92]:
```
covid_diff2 = covid_diff2[1:]
covid_diff2.head(13)
```

Out[92]:
```
1       0.0
2       0.0
3       0.0
4       0.0
5       0.0
6       0.0
7       3.0
8       7.0
9      22.0
10      3.0
11    115.0
12    243.0
13    252.0
Name: Deaths, dtype: float64
```

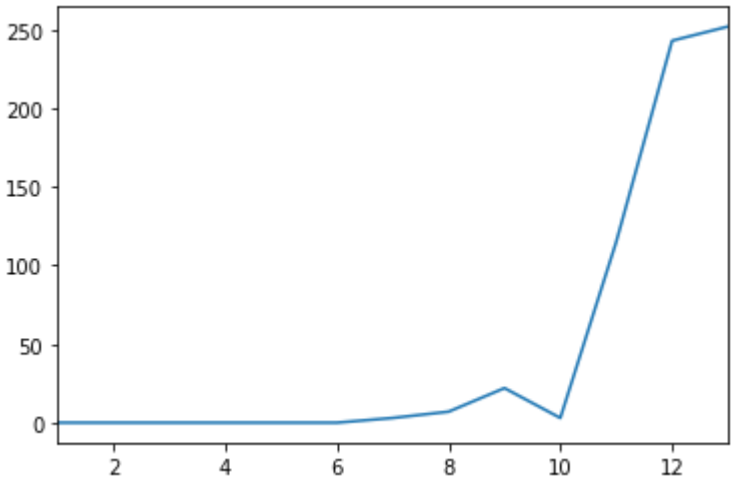
In [93]:
```
plot_acf(covid_diff2)
```

Out[93]:



In [94]:
```
covid_diff2.plot()
```

Out[94]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd0514a808>
```



In [95]:
```
Z2 = file2['Deaths'].values
train3 = Z2[0:11]
test3 = Z2[11:]
predictions2 = []
```

In [96]:
```
from statsmodels.tsa.arima_model import ARIMA
```

In [97]:
```
model_arima2 = ARIMA(train3,order=(1,2,0))
model_arima_fit2 = model_arima2.fit()
print(model_arima_fit2.aic)
```

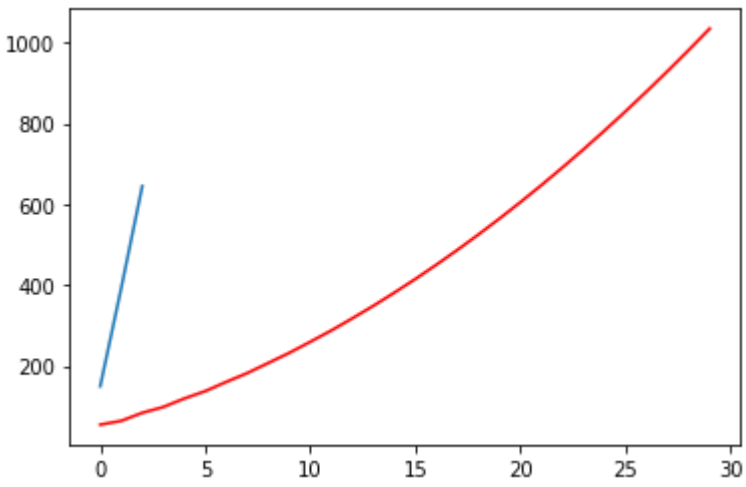
```
66.17972516856993
```

In [98]:
```
predictions2 = model_arima_fit2.forecast(steps=30)[0]
predictions2
```

Out[98]:
```
array([  54.92444034,   64.37107246,   84.24163398,   98.59281155,
        119.58575872,  137.94425714,  160.74397002,  182.5877616 ,
        207.59242964,  232.61778561,  260.05908538,  288.08928875,
        318.10201547,  349.03424787,  381.69682877,  415.47126792,
        450.82933393,  487.4111737 ,  525.49127403,  564.86026337,
        605.67784501,  647.82220135,  691.38625175,  736.29911969,
        782.61486797,  830.29225887,  879.36274747,  929.80234063,
        981.62933971, 1034.8297849 ])
```

In [99]:
```
plt.plot(test3)
plt.plot(predictions2,color='red')
```

Out[99]:
```
[<matplotlib.lines.Line2D at 0x1bd04cb7388>]
```



In [ ]: