In [26]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly as pl
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [2]:

```
df = pd.read_csv("covid_19_india.csv")
```

In [3]:

```
df = df.drop(['ConfirmedIndianNational','ConfirmedForeignNational'], ax
is=1)
```

In [4]:

```
df.rename(columns={'State/UnionTerritory':'State'},inplace=True)
df
```

Out[4]:

	Sno	Date	Time	State	Cured	Deaths	Confirmed
0	1	30/01/20	6:00 PM	Kerala	0	0	1
1	2	31/01/20	6:00 PM	Kerala	0	0	1
2	3	01/02/20	6:00 PM	Kerala	0	0	2
3	4	02/02/20	6:00 PM	Kerala	0	0	3
4	5	03/02/20	6:00 PM	Kerala	0	0	3
...	...	...	...	...	...	...	...
9286	9287	09/12/20	8:00 AM	Telengana	266120	1480	275261
9287	9288	09/12/20	8:00 AM	Tripura	32169	373	32945
9288	9289	09/12/20	8:00 AM	Uttarakhand	72435	1307	79141
9289	9290	09/12/20	8:00 AM	Uttar Pradesh	528832	7967	558173
9290	9291	09/12/20	8:00 AM	West Bengal	475425	8820	507995

9291 rows × 7 columns

In [5]:
```
df = df.replace('Telengana','Telangana')
df = df.replace('Telengana***','Telangana')
df = df.replace('Telangana***','Telangana')
df = df.replace('Punjab***','Punjab')
df = df.replace('Chandigarh***','Chandigarh')
df = df.replace('Maharashtra***', 'Maharashtra')
```

In [6]:
```
df_row =df[(df['State'] == 'Cases being reassigned to states')].index
df.drop(df_row,inplace=True)
df_row1 = df[(df['State'] == 'Unassigned')].index
df.drop(df_row1,inplace=True)
df_row2 = df[(df['State'] == 'Dadra and Nagar Haveli and Daman and Diu'
)].index
df.drop(df_row2,inplace=True)
```

In [7]:
```
df['Date'] = pd.to_datetime(df['Date'],format='%d/%m/%y',)
df
```

Out[7]:

	Sno	Date	Time	State	Cured	Deaths	Confirmed
0	1	2020-01-30	6:00 PM	Kerala	0	0	1
1	2	2020-01-31	6:00 PM	Kerala	0	0	1
2	3	2020-02-01	6:00 PM	Kerala	0	0	2
3	4	2020-02-02	6:00 PM	Kerala	0	0	3
4	5	2020-02-03	6:00 PM	Kerala	0	0	3
...	...	...	...	...	...	...	...
9286	9287	2020-12-09	8:00 AM	Telangana	266120	1480	275261
9287	9288	2020-12-09	8:00 AM	Tripura	32169	373	32945
9288	9289	2020-12-09	8:00 AM	Uttarakhand	72435	1307	79141
9289	9290	2020-12-09	8:00 AM	Uttar Pradesh	528832	7967	558173

	Sno	Date	Time	State	Cured	Deaths	Confirmed
9290	9291	2020-12-09	8:00 AM	West Bengal	475425	8820	507995

9047 rows × 7 columns

In [8]:

```
df_row =df[(df['State'] == 'Cases being reassigned to states')].index
df.drop(df_row,inplace=True)
df_row1 = df[(df['State'] == 'Unassigned')].index
df.drop(df_row1,inplace=True)
df_row2 = df[(df['State'] == 'Dadra and Nagar Haveli and Daman and Diu'
)].index
df.drop(df_row2,inplace=True)
```

In [9]:

```
#changing date from object datatype to readable datatype
df['Date'] = pd.to_datetime(df['Date'],format='%d/%m/%y',)
df
```

Out[9]:

	Sno	Date	Time	State	Cured	Deaths	Confirmed
0	1	2020-01-30	6:00 PM	Kerala	0	0	1
1	2	2020-01-31	6:00 PM	Kerala	0	0	1
2	3	2020-02-01	6:00 PM	Kerala	0	0	2
3	4	2020-02-02	6:00 PM	Kerala	0	0	3
4	5	2020-02-03	6:00 PM	Kerala	0	0	3
...	...	...	...	...	...	...	...
9286	9287	2020-12-09	8:00 AM	Telangana	266120	1480	275261
9287	9288	2020-12-09	8:00 AM	Tripura	32169	373	32945
9288	9289	2020-12-09	8:00 AM	Uttarakhand	72435	1307	79141
9289	9290	2020-12-09	8:00 AM	Uttar Pradesh	528832	7967	558173
9290	9291	2020-12-09	8:00 AM	West Bengal	475425	8820	507995

9047 rows × 7 columns

In [10]:

```
df.dtypes
```

Out[10]:

```
Sno                   int64
Date         datetime64[ns]
Time                 object
State                object
Cured                 int64
Deaths                int64
Confirmed             int64
dtype: object
```

In [11]:

```
#check for duplicated values
df.duplicated().sum()
```

Out[11]:

```
0
```

In [12]:

```
df.corr()
```

Out[12]:

	Sno	Cured	Deaths	Confirmed
Sno	1.000000	0.440493	0.296131	0.433529
Cured	0.440493	1.000000	0.892949	0.994994
Deaths	0.296131	0.892949	1.000000	0.913588
Confirmed	0.433529	0.994994	0.913588	1.000000

In [15]:

```
#Total number of cured patients, deaths and confirmed cases
cases = df[df['Date'] == df['Date'].max()].copy().fillna(0)
cases.index = cases["State"]
cases = cases.drop(['State', 'Date'], axis=1)
cases.head()
cases = cases.drop(['Time','Sno'],axis=1)
df1 = pd.DataFrame(pd.to_numeric(cases.sum())).transpose()
df1.style.background_gradient(cmap='Greens',axis=1)
```

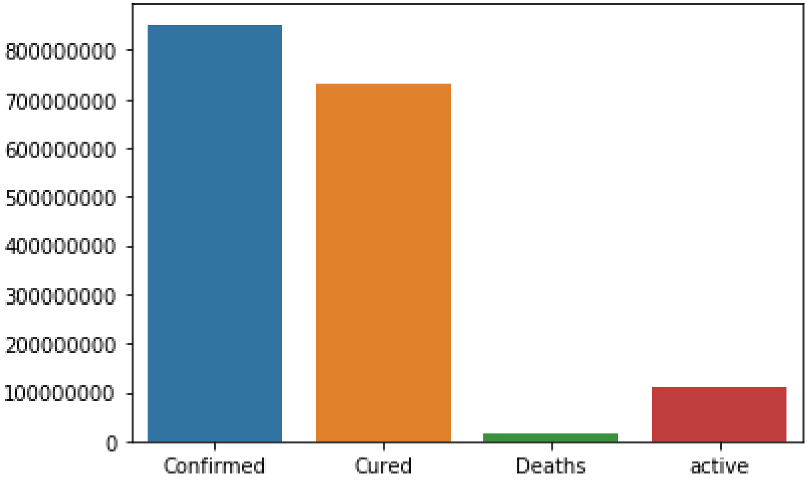
Out[15]:

	Cured	Deaths	Confirmed
0	9212251	141358	9732499

In [16]:

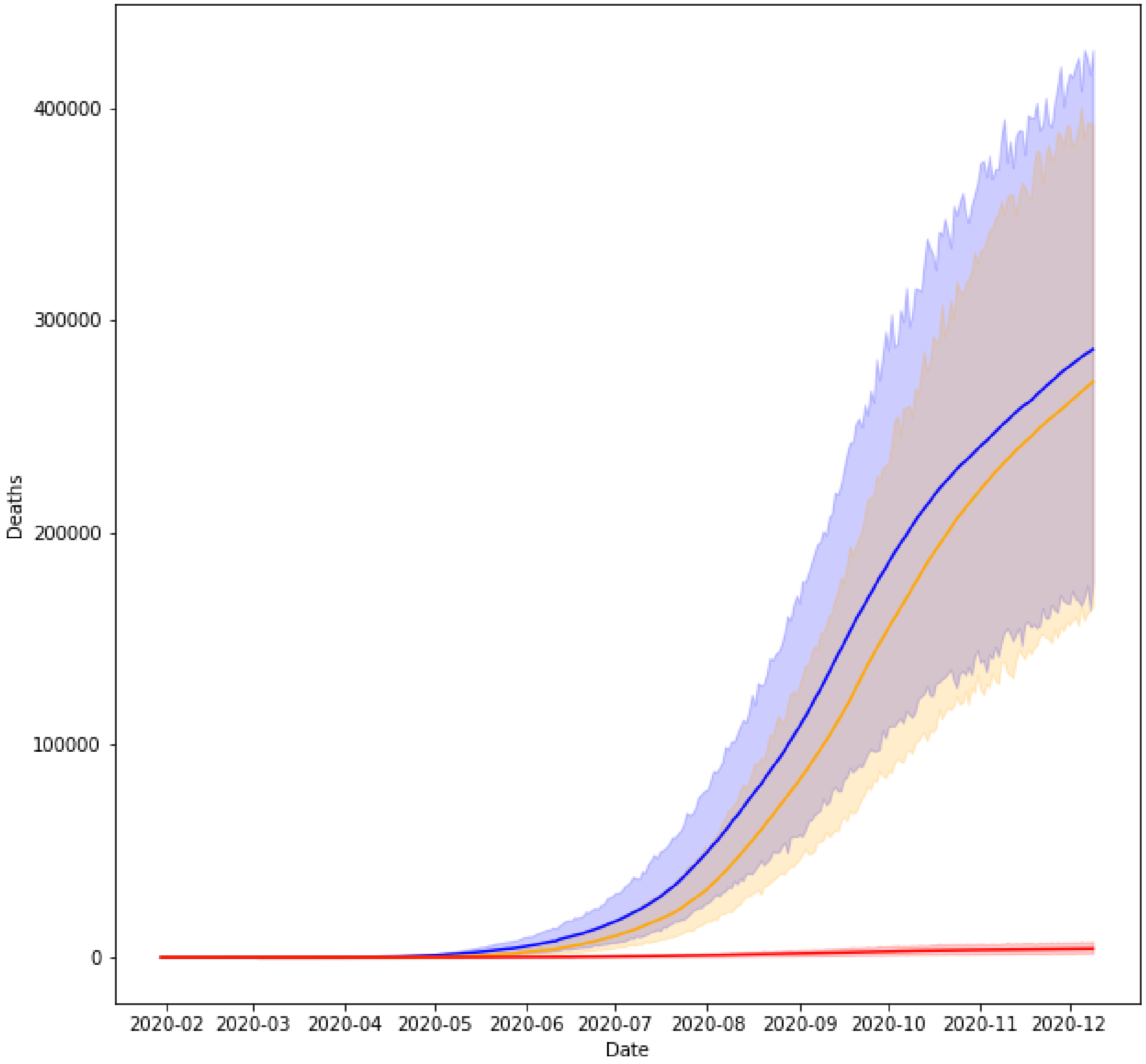
```
#bar chart representation of confirmed, cured, deaths and active cases
x = df['Confirmed'].sum()
y = df['Cured'].sum()
z= df['Deaths'].sum()
active= x-(y+z)
print('Total Confirmed cases =',x)
print('Total Cured cases =',y)
print('Total Active cases =',active)
print('Total Number of Deaths =',z)
barp = sns.barplot(x=['Confirmed','Cured','Deaths','active'],y=[x,y,z,a
ctive])
barp.set_yticklabels(labels=(barp.get_yticks()*1).astype(int))
plt.show()
```

```
Total Confirmed cases = 852553085
Total Cured cases = 730223531
Total Active cases = 108508369
Total Number of Deaths = 13821185
```



In [17]:

```
#variation of confirmed, cured and death rates with date
plt.figure(figsize=(10,10))
sns.lineplot(data=df,x='Date',y='Confirmed',color='Blue')
sns.lineplot(data=df,x='Date',y='Cured',color='Orange')
sns.lineplot(data=df,x='Date',y='Deaths',color='Red')
plt.show()
```

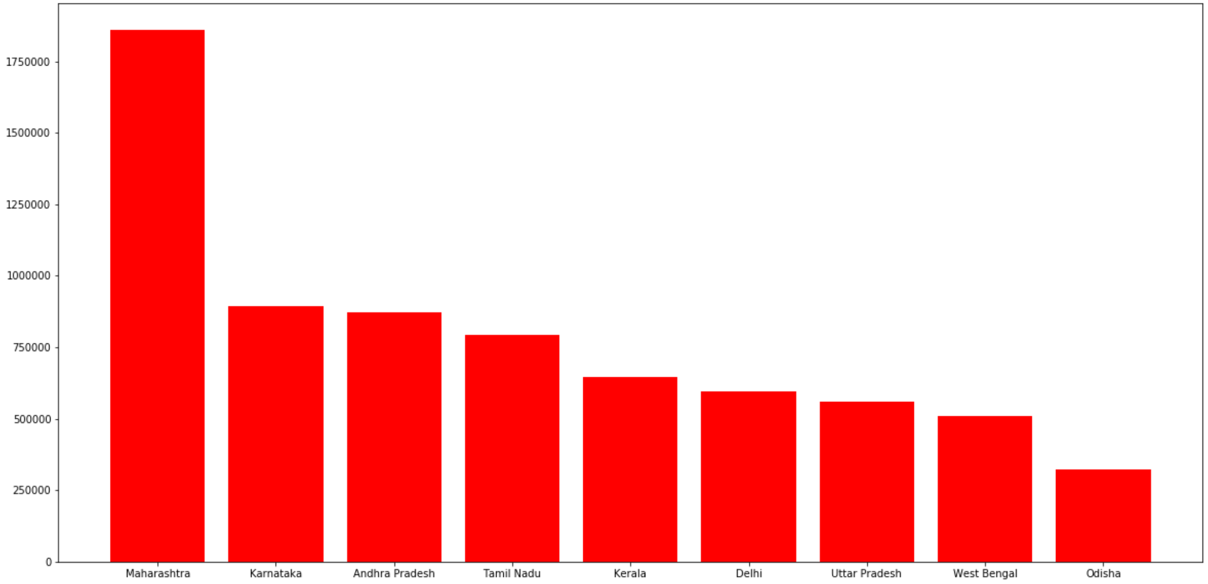


In [18]:

```
#10 states with most confirmed cases
last = df.tail(35)
most_confirmed = last.sort_values(by='Confirmed', ascending=False).head
(10)
plt.figure(figsize=(20,10))
plt.bar(most_confirmed['State'],height= most_confirmed['Confirmed'],col
or='red')
```

Out[18]:

```
<BarContainer object of 10 artists>
```

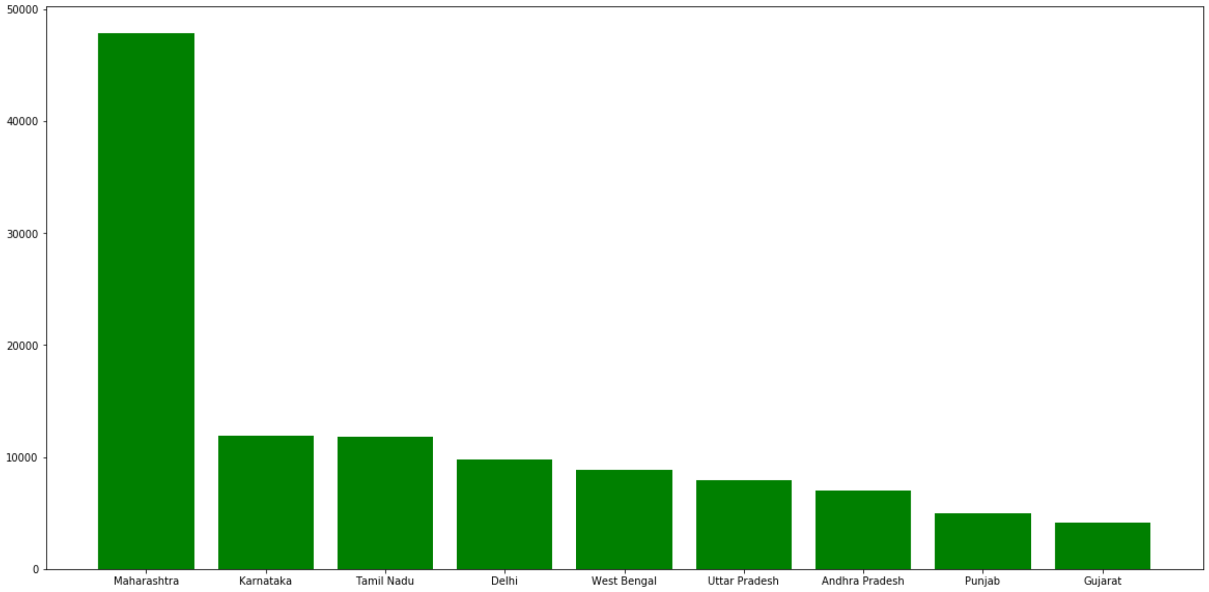


In [19]:

```
#10 states with most number of deaths

most_deaths = last.sort_values(by='Deaths', ascending=False).head(10)

plt.figure(figsize=(20,10))

plt.bar(most_deaths['State'],height= most_deaths['Deaths'],color='gree
n')
```

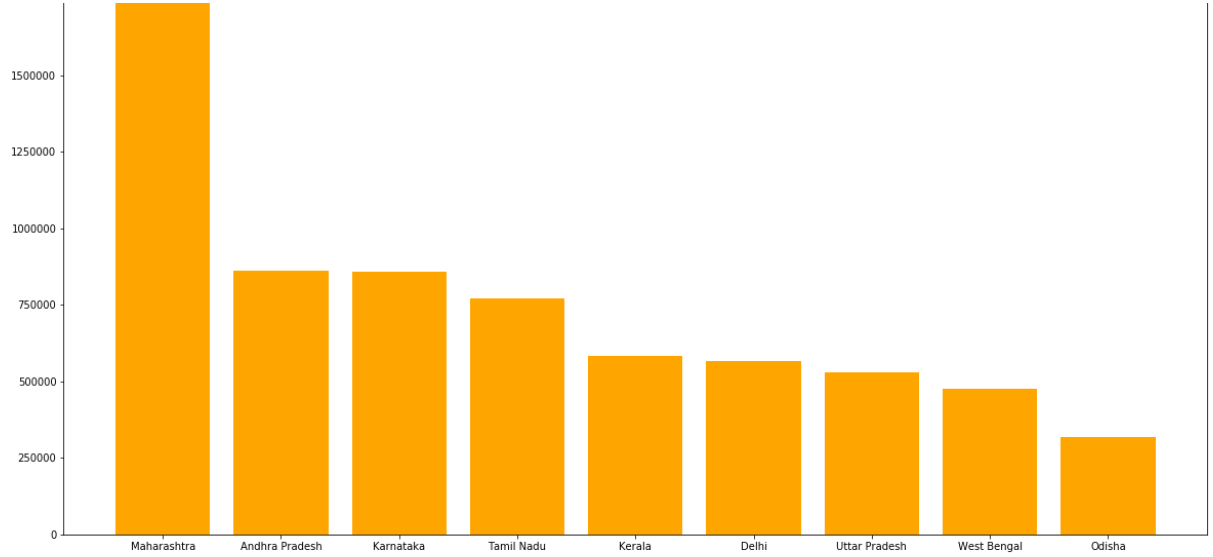
Out[19]:

```
<BarContainer object of 10 artists>
```



```
In [20]: #top 10 states with the most number of cured people

         most_cured = last.sort_values(by='Cured', ascending=False).head(10)

         plt.figure(figsize=(20,10))

         plt.bar(most_cured['State'],height= most_cured['Cured'],color='orange')
```

```
Out[20]: <BarContainer object of 10 artists>
```

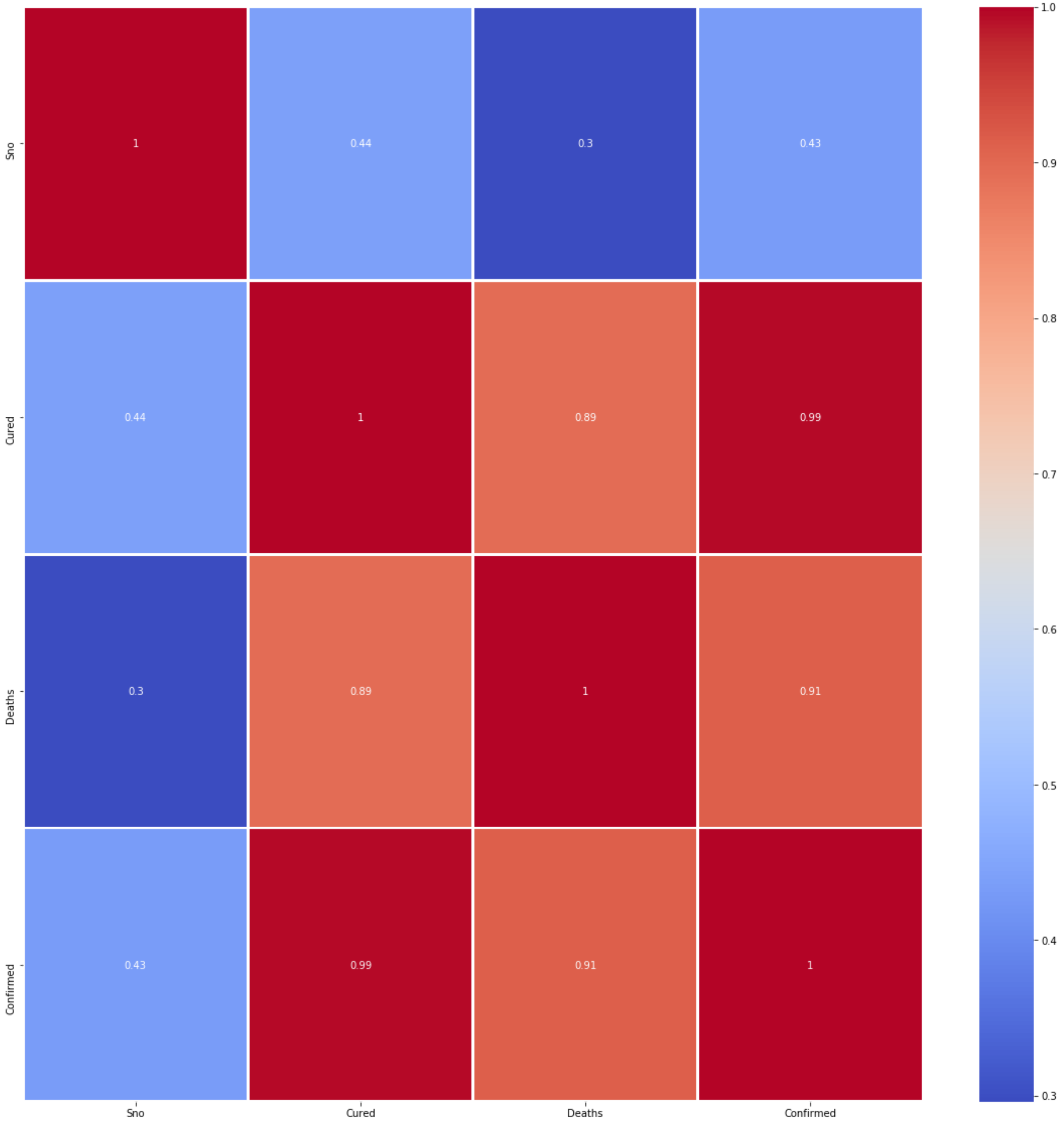


In [22]:

```
#Heatmap
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(), annot = True, cmap ='coolwarm', linewidths=2)
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24e5e88a348>
```


Sno
Cured
Deaths
Confirmed
1
0.44
0.3
0.43
0.44
1
0.89
0.99
0.3
0.89
1
0.91
0.43
0.99
0.91
1
Sno
Cured
Deaths
Confirmed
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

In [23]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

label = le.fit_transform(df["State"])
data = df.drop("State",axis = 1)
data["States"] = label


X = data[['States','Cured', 'Confirmed']]
y = data[['Deaths']]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
3, random_state=42)
print(X_train.shape)
print(X_test.shape)
```

```
(6061, 3)
(2986, 3)
```

In [24]:

```
from sklearn.linear_model import LinearRegression
linear_model = LinearRegression(normalize = True, fit_intercept=True)
linear_model.fit(X_train,y_train)
test_linear_predict = linear_model.predict(X_test)
linear_predict = linear_model.predict(X_train)
```

In [27]:

```
print('MAE: ',mean_absolute_error(test_linear_predict,y_test))
print('MSE: ',mean_squared_error(test_linear_predict,y_test))
```

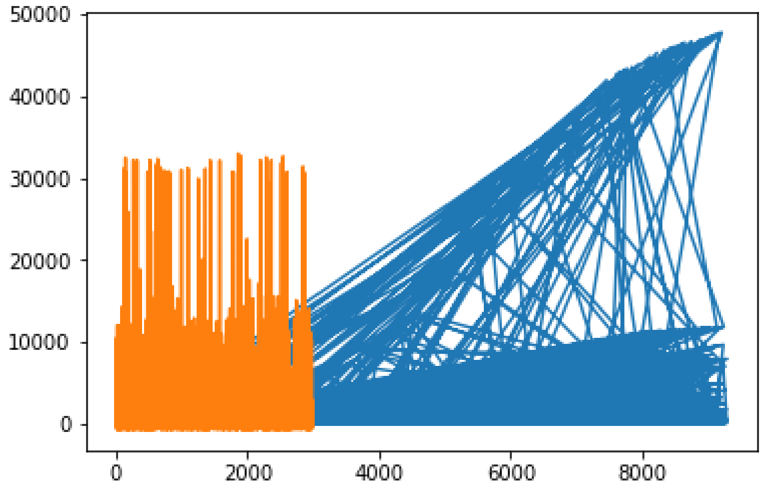
```
MAE:  911.6781691592545
MSE:  3647437.0438132533
```

In [30]:

```
plt.plot(y_test)
plt.plot(test_linear_predict)
```

Out[30]:

```
[<matplotlib.lines.Line2D at 0x24e5f7461c8>]
```



In [31]:

```
r2_score = linear_model.score(X_test,y_test)
print(r2_score*100,'%')
```

```
87.08220544196492 %
```

In [ ]: