Aim to predict conformed covid 19 cases

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

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

```
df = pd.read_csv('/content/drive/MyDrive/data/covid_19_india.csv')
```

```
df.head()
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
0	1	30/01/20	6:00 PM	Kerala	1	0	0	0	1
1	2	31/01/20	6:00 PM	Kerala	1	0	0	0	1
2	3	01/02/20	6:00 PM	Kerala	2	0	0	0	2
3	4	02/02/20	6:00 PM	Kerala	3	0	0	0	3
4	5	03/02/20	6:00 PM	Kerala	3	0	0	0	3

Next steps: Generate code with df View recommended plots

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1478 entries, 0 to 1477
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Sno                       1478 non-null   int64 
 1   Date                      1478 non-null   object
 2   Time                      1478 non-null   object
 3   State/UnionTerritory      1478 non-null   object
 4   ConfirmedIndianNational   1478 non-null   object
 5   ConfirmedForeignNational  1478 non-null   object
 6   Cured                     1478 non-null   int64 
 7   Deaths                    1478 non-null   int64 
 8   Confirmed                 1478 non-null   int64 
dtypes: int64(4), object(5)
memory usage: 104.0+ KB
```

```
df.shape
```

```
(1478, 9)
```

```
df['State/UnionTerritory'].unique()
```

```
array(['Kerala', 'Telengana', 'Delhi', 'Rajasthan', 'Uttar Pradesh',
       'Haryana', 'Ladakh', 'Tamil Nadu', 'Karnataka', 'Maharashtra',
       'Punjab', 'Jammu and Kashmir', 'Andhra Pradesh', 'Uttarakhand',
       'Odisha', 'Puducherry', 'West Bengal', 'Chhattisgarh',
       'Chandigarh', 'Gujarat', 'Himachal Pradesh', 'Madhya Pradesh',
       'Bihar', 'Manipur', 'Mizoram', 'Andaman and Nicobar Islands',
       'Goa', 'Unassigned', 'Assam', 'Jharkhand', 'Arunachal Pradesh',
       'Tripura', 'Nagaland', 'Meghalaya', 'Nagaland#', 'Jharkhand#'],
      dtype=object)
```

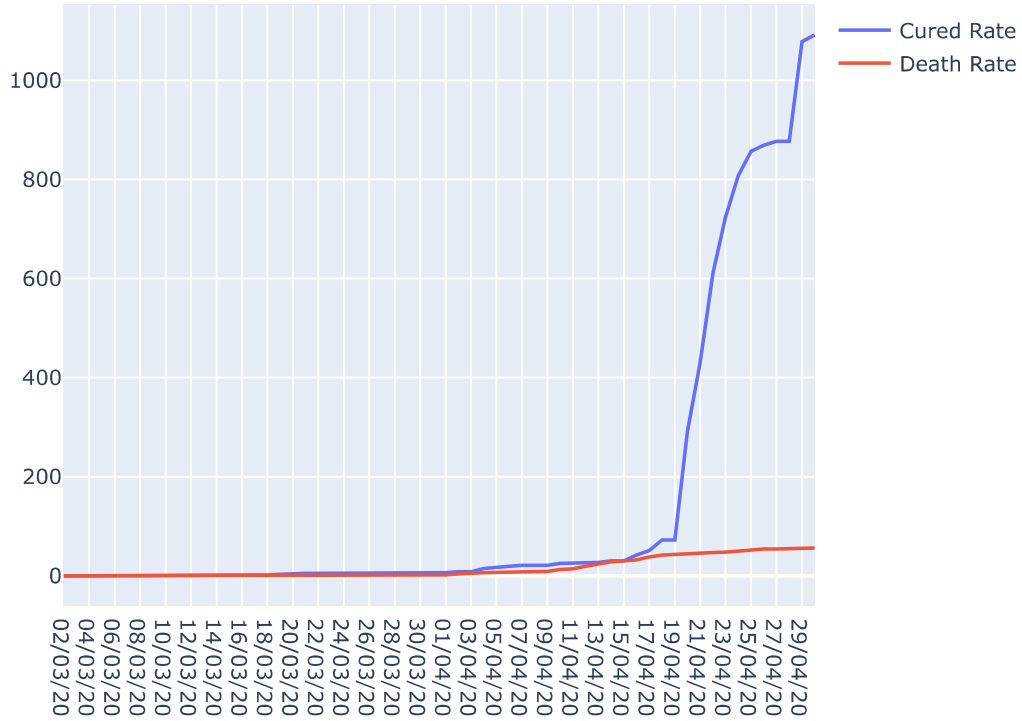
```
df_Delhi = df.loc[df['State/UnionTerritory']=='Delhi']
```

```
df_Delhi.head()
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
34	35	02/03/20	6:00 PM	Delhi	1	0	0	0	1
38	39	03/03/20	6:00 PM	Delhi	1	0	0	0	1
42	43	04/03/20	6:00 PM	Delhi	1	0	0	0	1
45	46	05/03/20	6:00 PM	Delhi	2	0	0	0	2
51	52	06/03/20	6:00 PM	Delhi	3	0	0	0	3

```
import plotly.offline as py
import plotly.graph_objs as go


cured_rate = go.Scatter(x=df_Delhi['Date'],y=df_Delhi['Cured'],name='Cured Rate')
death_rate =go.Scatter(x=df_Delhi['Date'],y=df_Delhi['Deaths'],name='Death Rate')
py.iplot([cured_rate,death_rate])
```




```
df_Delhi.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 60 entries, 34 to 1453
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Sno                       60 non-null     int64 
 1   Date                      60 non-null     object
 2   Time                      60 non-null     object
 3   State/UnionTerritory      60 non-null     object
 4   ConfirmedIndianNational   60 non-null     object
 5   ConfirmedForeignNational  60 non-null     object
 6   Cured                     60 non-null     int64 
```

```
 7   Deaths                    60 non-null     int64
 8   Confirmed                 60 non-null     int64
dtypes: int64(4), object(5)
memory usage: 6.7+ KB
```

```
df1 = df_Delhi[['Confirmed']]
```

```
df1
```

	Confirmed
34	1
38	1
42	1
45	2
51	3
62	3
71	3
84	4
87	4
98	5
109	6
122	6
135	7
149	7
163	7
178	8
193	10
211	12
230	17
250	26
273	29
296	29
319	30
343	31
370	36

397	36
424	39
451	49
478	87
506	97
535	152
564	219
594	219
625	445
655	503
685	523
715	576
746	576
777	669
808	898
839	999
870	1069
901	1154
933	1510
966	1561
999	1578
1032	1640
1065	1707
1098	1893
1131	2003
1164	2081

Next

Generate code with `df1`

View recommended plots

1197	2156
1229	2248
1261	2376
1293	2514
1325	2625
1357	2918
1389	3108
1421	3314
1453	3439

```
df1=df1.values
```

```
type(df1)
```

```
numpy.ndarray
```

```
df1
```

```
array([[   1],
       [   2],
       [   3],
       [   3],
       [   3],
       [   4],
       [   4],
       [   5],
       [   6],
       [   6],
       [   7],
       [   7],
       [   7],
       [   8],
       [  10],
       [  12],
       [  17],
       [  26],
```

```
[  30],
[  31],
[  36],
[  36],
[  39],
[  49],
[  87],
[  97],
[ 152],
[ 219],
[ 219],
[ 445],
[ 503],
[ 523],
[ 576],
[ 576],
[ 669],
[ 898],
[ 903],
[1069],
[1154],
[1510],
[1561],
[1578],
[1640],
[1707],
[1893],
[2003],
[2081],
[2156],
[2248],
[2376],
[2514],
[2625],
[2918],
[3108],
[3314],
[3439]])
```

```
train_size = int(len(df1)*0.8)
test_size = len(df1)-train_size
```

```
df1.shape
```

```
(60, 1)
```

```
train_size
```

```
48
```

```
train,test = df1[0:train_size],df1[train_size:len(df1)]
```

```
train
```

```
array([[   1],
       [   1],
       [   1],
       [   2],
       [   3],
       [   3],
       [   3],
       [   4],
       [   4],
       [   5],
       [   6],
       [   6],
       [   7],
       [   7],
       [   7],
       [   8],
       [  10],
       [  12],
       [  17],
       [  26],
       [  29],
       [  29],
       [  30],
       [  31],
       [  36],
       [  36],
       [  39],
       [  49],
       [  87],
       [  97],
       [ 152],
       [ 219],
       [ 219],
       [ 445],
       [ 503],
       [ 523],
       [ 576],
```

```
            [ 576],
            [ 669],
            [ 898],
            [ 903],
            [1069],
            [1154],
            [1510],
            [1561],
            [1578],
            [1640],
            [1707]])
```

```
test
```

```
array([1893, 2003, 2081, 2156, 2248, 2376, 2514, 2625, 2918, 3108, 3314,
       3439])
```

```
def create_dataset(dataset,look_back=1):
  datax,datay=[],[]
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back),0]
    datax.append(a)
    datay.append(dataset[i+look_back,0])
  return np.array(datax),np.array(datay)
```

```
look_back=2
trainx,trainy=create_dataset(train,look_back)
testx,testy=create_dataset(test,look_back)
```

```
#Regression
from sklearn.linear_model import LinearRegression
```

```
model = LinearRegression()
```

```
model.fit(trainx,trainy)
```

```
▾ LinearRegression
LinearRegression()
```

```
predict_value = model.predict(testx)

df_Delhi = pd.DataFrame({'Actual ':testy.flatten(),'Predicted ':predict_value.flatten()})

df_Delhi
```

	Actual	Predicted
0	2081	2182.279931
1	2156	2271.654773
2	2248	2353.780449
3	2376	2451.682692
4	2514	2586.199614
5	2625	2735.180545
6	2918	2859.823186
7	3108	3154.070347
8	3314	3375.216898