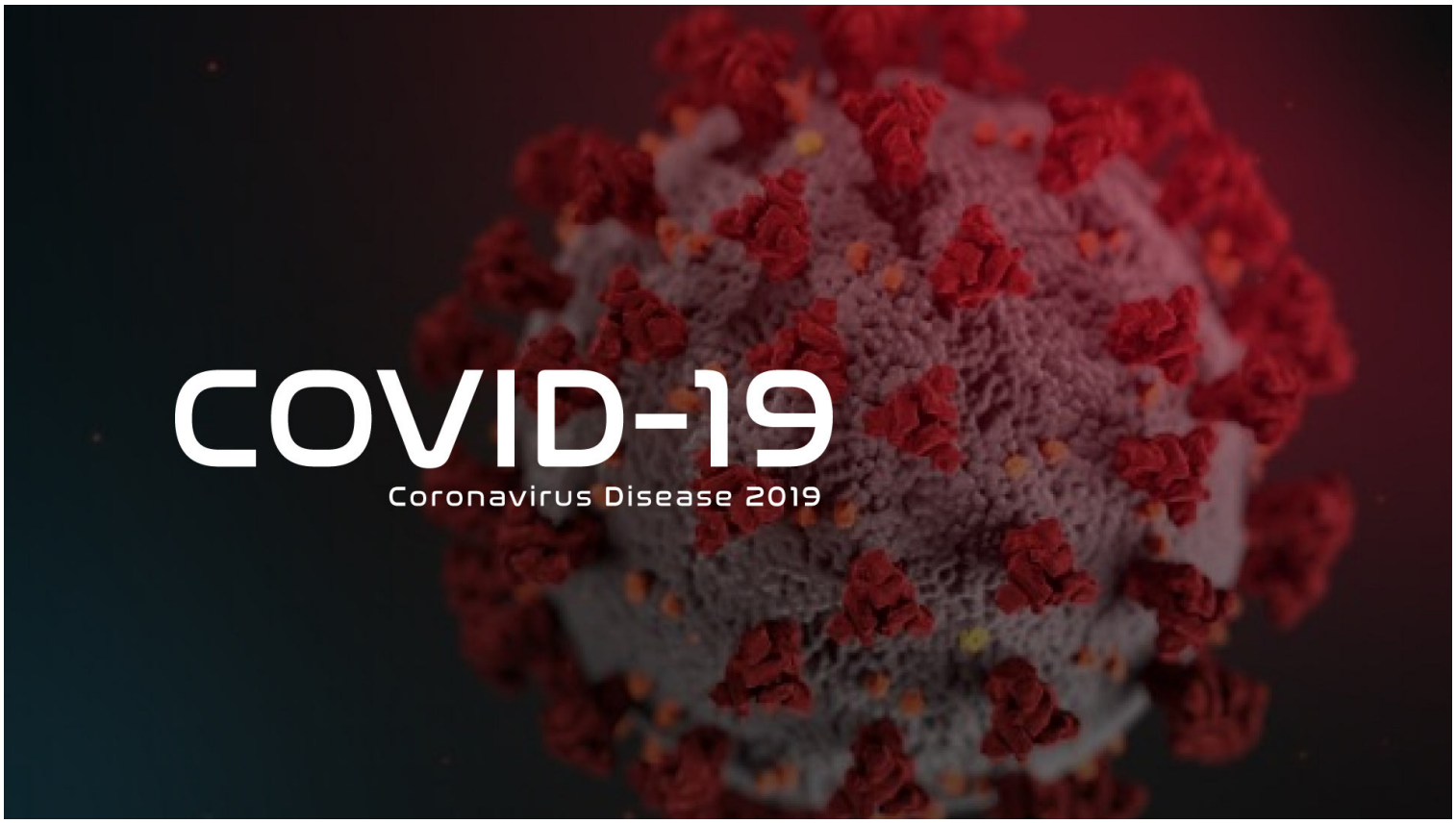


In [1]:
```
#path = 'https://raw.githubusercontent.com/umangkejriwal1122/Machine-Learning/master/Data%20Sets/covid_19_clean_complete.csv'
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

In [3]:
```
df = pd.read_csv('C:/Users/VAIO/Downloads/COVID-19 AI ML/covid_19_clean_complete.csv')
df.head()
```

Out[3]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered
0	NaN	Afghanistan	33.0000	65.0000	1/22/20	0	0	0
1	NaN	Albania	41.1533	20.1683	1/22/20	0	0	0
2	NaN	Algeria	28.0339	1.6596	1/22/20	0	0	0
3	NaN	Andorra	42.5063	1.5218	1/22/20	0	0	0
4	NaN	Angola	-11.2027	17.8739	1/22/20	0	0	0

In [4]:
```
df.drop(['Province/State'],axis=1,inplace=True)
df.head()
```

Out[4]:

	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered
0	Afghanistan	33.0000	65.0000	1/22/20	0	0	0
1	Albania	41.1533	20.1683	1/22/20	0	0	0
2	Algeria	28.0339	1.6596	1/22/20	0	0	0
3	Andorra	42.5063	1.5218	1/22/20	0	0	0
4	Angola	-11.2027	17.8739	1/22/20	0	0	0

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27984 entries, 0 to 27983
Data columns (total 7 columns):
Country/Region    27984 non-null object
Lat               27984 non-null float64
Long              27984 non-null float64
Date              27984 non-null object
Confirmed         27984 non-null int64
Deaths            27984 non-null int64
Recovered         27984 non-null int64
dtypes: float64(2), int64(3), object(2)
memory usage: 1.5+ MB
```

In [6]:
```
df['Date'] = pd.to_datetime(df['Date'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27984 entries, 0 to 27983
Data columns (total 7 columns):
Country/Region    27984 non-null object
Lat               27984 non-null float64
Long              27984 non-null float64
Date              27984 non-null datetime64[ns]
Confirmed         27984 non-null int64
Deaths            27984 non-null int64
Recovered         27984 non-null int64
dtypes: datetime64[ns](1), float64(2), int64(3), object(1)
memory usage: 1.5+ MB
```

In [7]:
```
df.rename(columns={"Country/Region":"Country"},inplace=True)
df.head()
```

Out[7]:

	Country	Lat	Long	Date	Confirmed	Deaths	Recovered
0	Afghanistan	33.0000	65.0000	2020-01-22	0	0	0
1	Albania	41.1533	20.1683	2020-01-22	0	0	0
2	Algeria	28.0339	1.6596	2020-01-22	0	0	0
3	Andorra	42.5063	1.5218	2020-01-22	0	0	0
4	Angola	-11.2027	17.8739	2020-01-22	0	0	0

In [8]:
```
df['Active'] = df['Confirmed'] - df['Recovered'] - df['Deaths']
df.tail()
```

Out[8]:

	Country	Lat	Long	Date	Confirmed	Deaths	Recovered	Active
27979	Western Sahara	24.215500	-12.885800	2020-05-06	6	0	5	1
27980	Sao Tome and Principe	0.186360	6.613081	2020-05-06	174	3	4	167
27981	Yemen	15.552727	48.516388	2020-05-06	25	5	0	20
27982	Comoros	-11.645500	43.333300	2020-05-06	8	1	0	7
27983	Tajikistan	38.861034	71.276093	2020-05-06	379	8	0	371

Latest Date Data

In [9]:
```
#### Latest Data
latest_date = df[df['Date']==df['Date'].max()]
latest_date.head()
```

Out[9]:

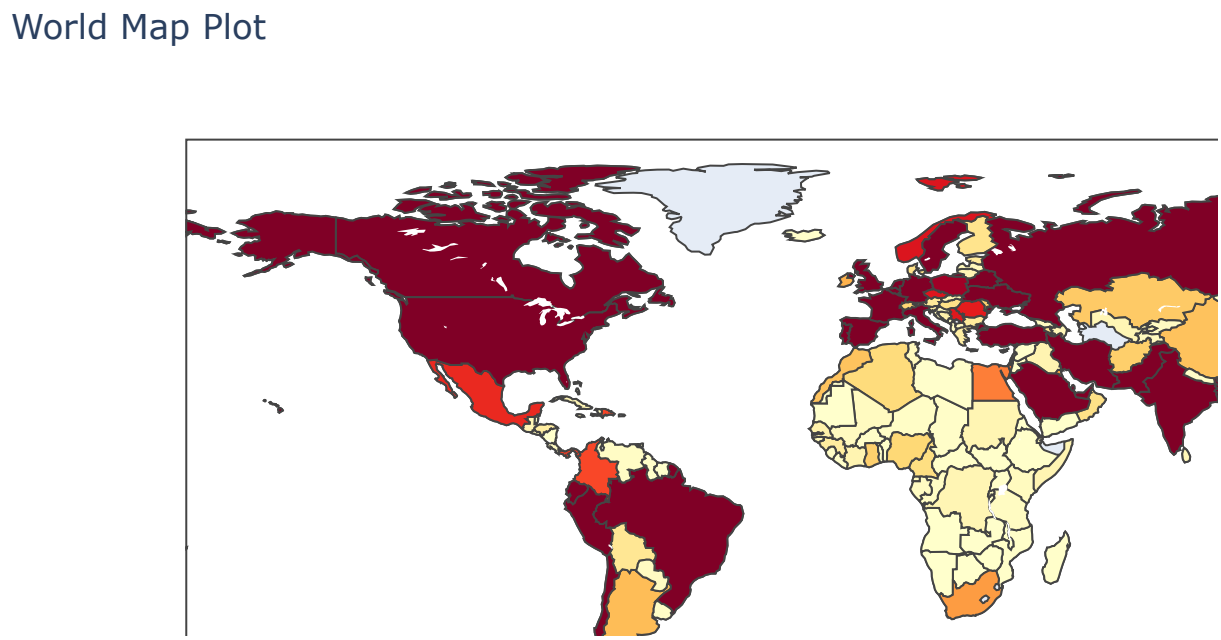
	Country	Lat	Long	Date	Confirmed	Deaths	Recovered	Active
27720	Afghanistan	33.0000	65.0000	2020-05-06	3392	104	458	2830
27721	Albania	41.1533	20.1683	2020-05-06	832	31	595	206
27722	Algeria	28.0339	1.6596	2020-05-06	4997	476	2197	2324
27723	Andorra	42.5063	1.5218	2020-05-06	751	46	521	184
27724	Angola	-11.2027	17.8739	2020-05-06	36	2	11	23

In [10]:
```
world = latest_date.groupby('Country')['Confirmed','Deaths','Recovered','Active'].sum
()
world = world.reset_index()
world.head()
```

Out[10]:

	Country	Confirmed	Deaths	Recovered	Active
0	Afghanistan	3392	104	458	2830
1	Albania	832	31	595	206
2	Algeria	4997	476	2197	2324
3	Andorra	751	46	521	184
4	Angola	36	2	11	23

In [11]:
```
#### Plot on World Map (Active Cases)
figure = px.choropleth(world,locations='Country',locationmode='country names',
                       color='Active',range_color=[1,10000]
                       ,color_continuous_scale='ylorrd',title='World Map Plot')
figure.show()
```



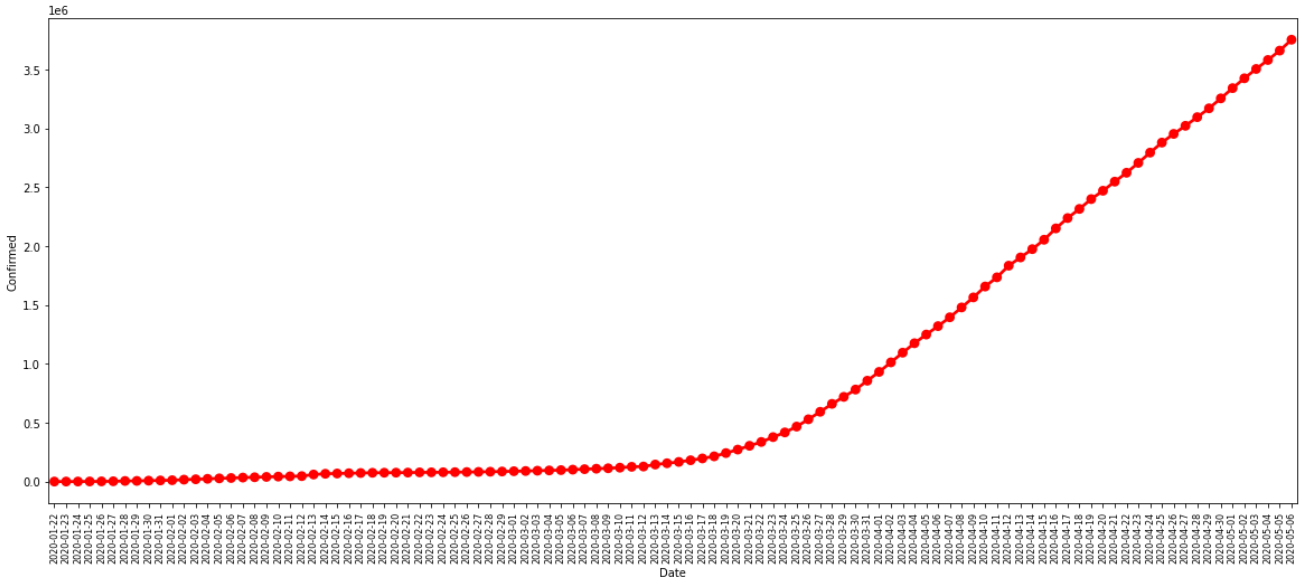
World - Total Confirmed Cases

In [12]:
```
#### Plot WorldWide Confirmed Cases Over Data
World_Total_Confirmed = df.groupby('Date')['Confirmed'].sum().reset_index()
World_Total_Confirmed.tail()
```

Out[12]:

	Date	Confirmed
101	2020-05-02	3427337
102	2020-05-03	3506723
103	2020-05-04	3583049
104	2020-05-05	3662685
105	2020-05-06	3755335

In [13]:
```
plt.figure(figsize=(20,8))
plt.xticks(rotation=90,fontsize=8)
sns.pointplot(World_Total_Confirmed['Date'].dt.date,World_Total_Confirmed['Confirmed'
]
                    ,color='red')
plt.show()
```

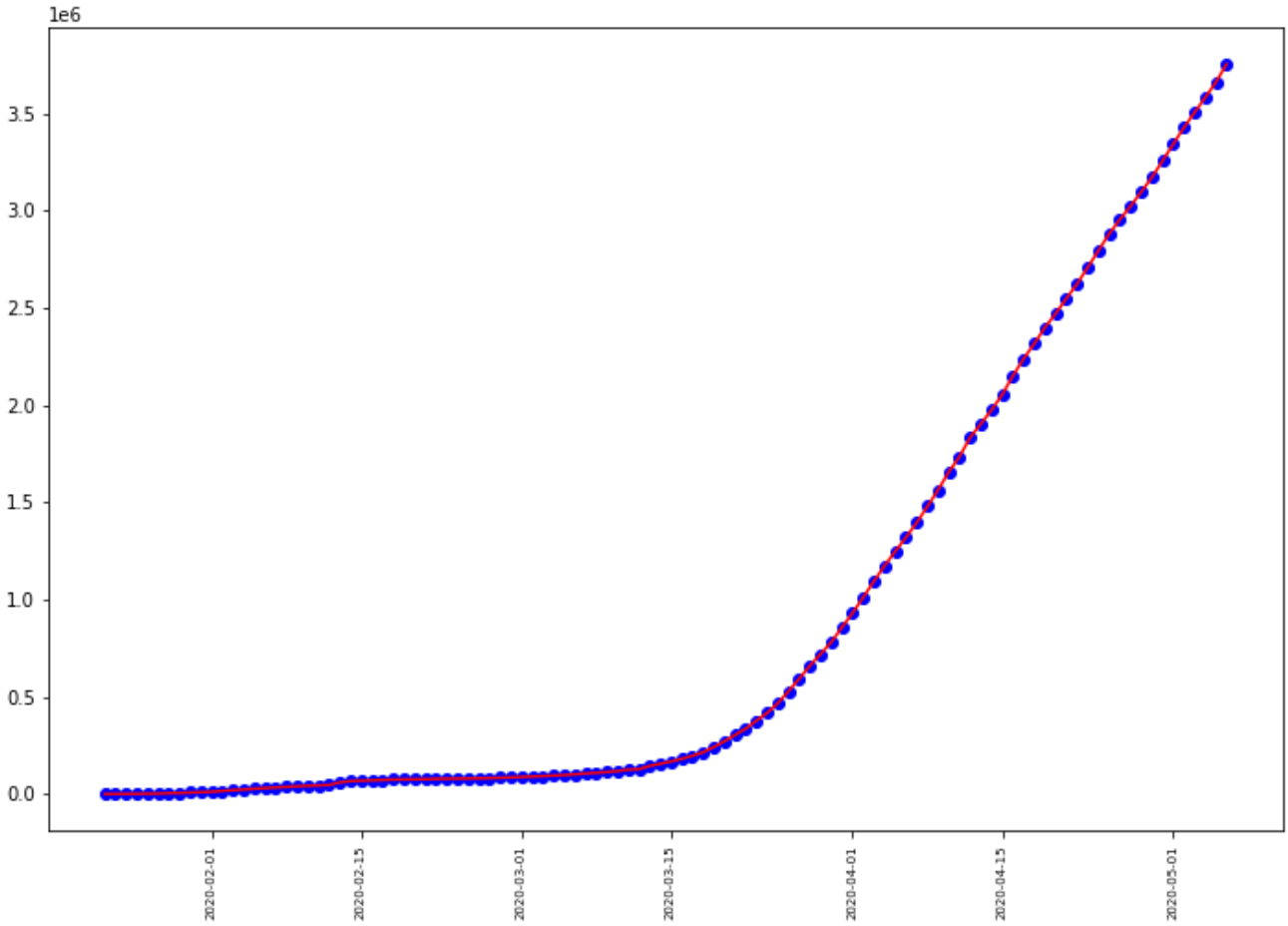


In [14]:
```
plt.figure(figsize=(12,8))
plt.xticks(rotation=90,fontsize=7)
plt.plot(World_Total_Confirmed['Date'].dt.date,World_Total_Confirmed['Confirmed']
                  ,color='red')
plt.scatter(World_Total_Confirmed['Date'].dt.date,World_Total_Confirmed['Confirmed']
                  ,color='blue')
plt.show()
```

```
C:\Users\VAIO\Anaconda3\lib\site-packages\pandas\plotting\_matplotlib\converter.py:1
03: FutureWarning:

Using an implicitly registered datetime converter for a matplotlib plotting method.
The converter was registered by pandas on import. Future versions of pandas will req
uire you to explicitly register matplotlib converters.

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
```



Active - Top 20 Countries

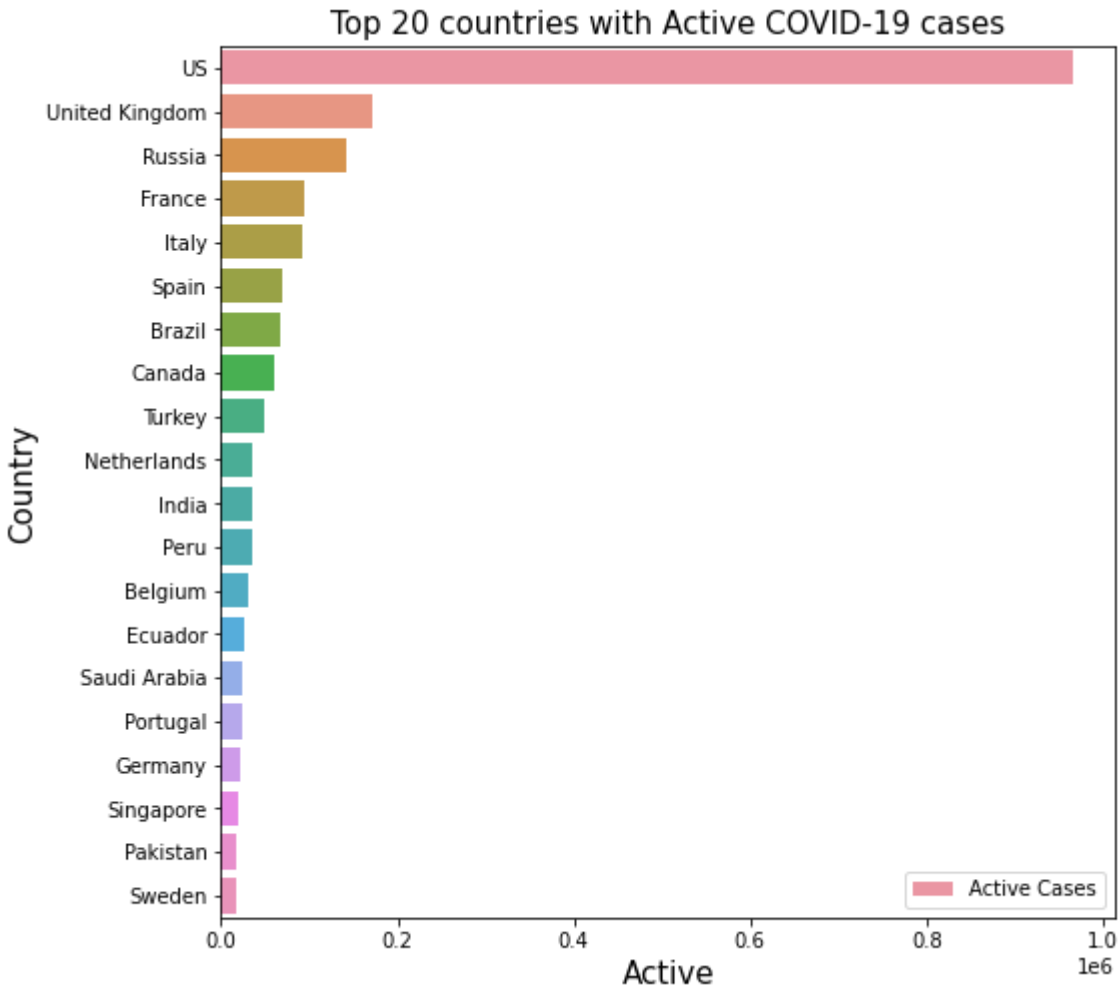
In [15]:
```
##### Top 20 Countries having most number of Active Cases
Active_Top_Countries = latest_date.groupby('Country')['Active'].sum().sort_values(asc
ending=False)
Active_Top_Countries = Active_Top_Countries.reset_index()
Active_Top_Countries.head()
```

Out[15]:

	Country	Active
0	US	965262
1	United Kingdom	171275
2	Russia	143065
3	France	94333
4	Italy	91528

In [16]:
```
Active_Top_20_Countries = Active_Top_Countries.head(20)
```

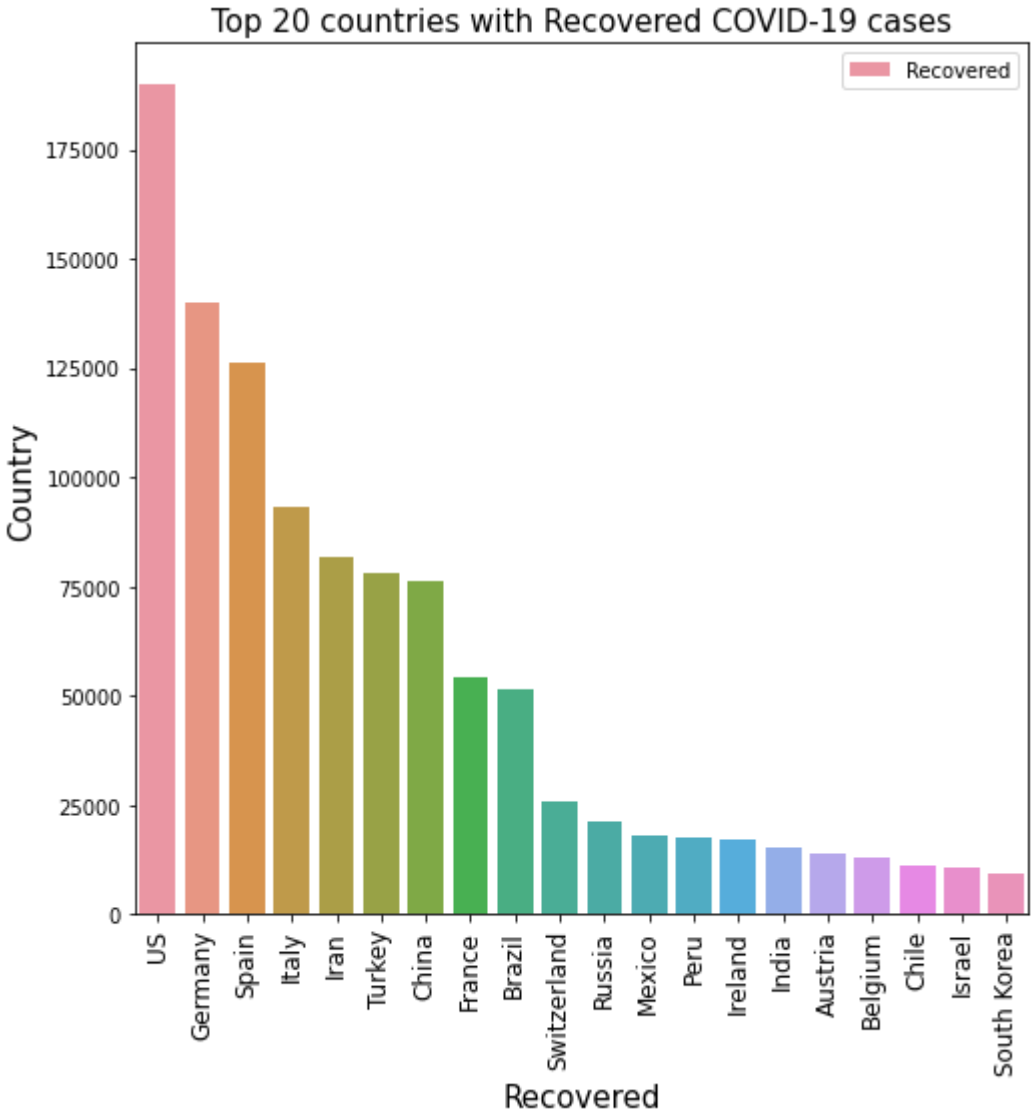
In [17]:
```
plt.figure(figsize=(8,8))
sns.barplot(Active_Top_20_Countries['Active'],Active_Top_20_Countries['Country'], lab
el = "Active Cases")
plt.xlabel("Active",  fontdict = {'fontsize' : 15})
plt.ylabel("Country",  fontdict = {'fontsize' : 15})
plt.title("Top 20 countries with Active COVID-19 cases",  fontdict = {'fontsize' : 15
})
plt.legend()
plt.show()
```



Recovered - Top 20 Countries

In [18]:
```
##### Top 20 Countries having most number of Recovered Cases
Recovered_Top_Countries = latest_date.groupby('Country')['Recovered'].sum().sort_valu
es(ascending=False)
Recovered_Top_Countries= Recovered_Top_Countries.reset_index()
Recovered_Top_20_Countries = Recovered_Top_Countries.head(20)
```

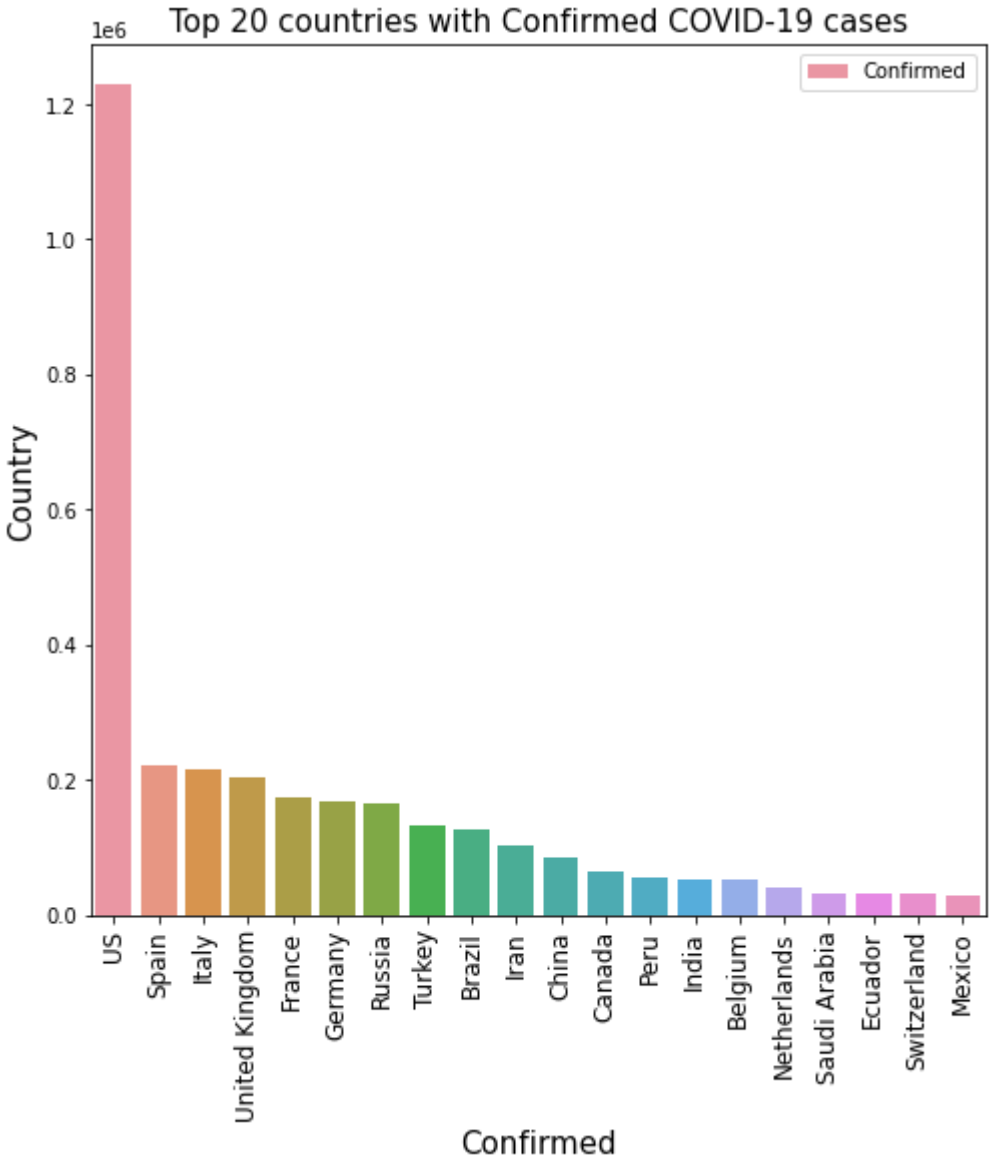
In [19]:
```
plt.figure(figsize=(8,8))
sns.barplot(Recovered_Top_20_Countries['Country'],Recovered_Top_20_Countries['Recover
ed'], label = "Recovered")
plt.tick_params(axis="x",labelrotation=90, labelsize = 12)
plt.xlabel("Recovered",  fontdict = {'fontsize' : 15})
plt.ylabel("Country",  fontdict = {'fontsize' : 15})
plt.title("Top 20 countries with Recovered COVID-19 cases",  fontdict = {'fontsize' :
15})
plt.legend()
plt.show()
plt.show()
```



Confirmed - Top 20 Countries

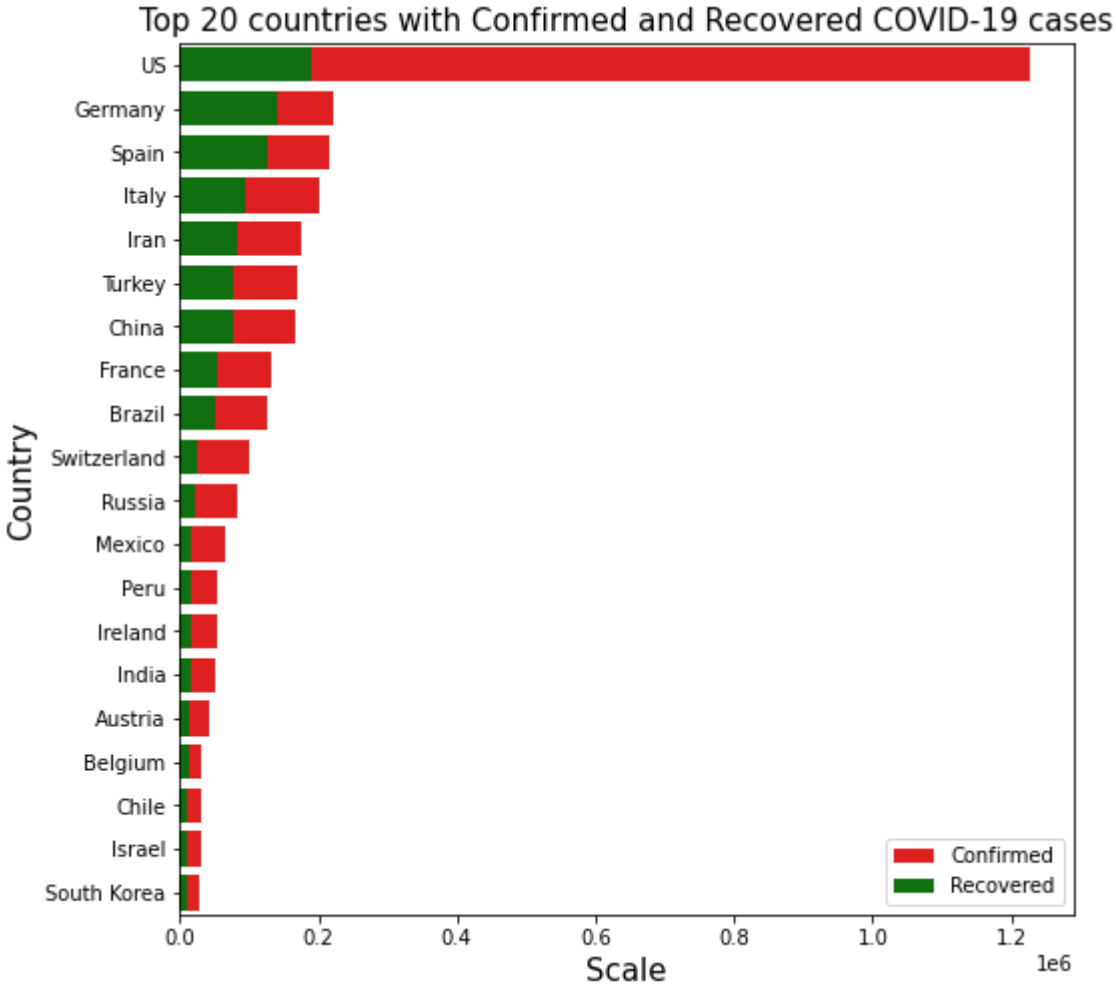
In [20]:
```
##### Top 20 Countries having most number of Confirmed Cases
Confirmed_Top_Countries  = latest_date.groupby('Country')['Confirmed'].sum().sort_val
ues(ascending=False)
Confirmed_Top_Countries= Confirmed_Top_Countries.reset_index()
Confirmed_Top_20_Countries = Confirmed_Top_Countries.head(20)
```

In [21]:
```
plt.figure(figsize=(8,8))
sns.barplot(Confirmed_Top_20_Countries['Country'],Confirmed_Top_20_Countries['Confirmed'], label = "Confirmed")
plt.tick_params(axis="x",labelrotation=90, labelsize = 12)
plt.xlabel("Confirmed",  fontdict = {'fontsize' : 15})
plt.ylabel("Country",  fontdict = {'fontsize' : 15})
plt.title("Top 20 countries with Confirmed COVID-19 cases",  fontdict = {'fontsize' : 15})
plt.legend()
plt.show()
```



Confirmed/Recovered - Top 20 Countries

In [22]:
```
plt.figure(figsize=(8,8))
sns.barplot(Confirmed_Top_20_Countries['Confirmed'],Confirmed_Top_20_Countries['Country'],color='red', label = "Confirmed")
sns.barplot(Recovered_Top_20_Countries['Recovered'],Recovered_Top_20_Countries['Country'],color='green', label = "Recovered")
plt.xlabel("Scale",  fontdict = {'fontsize' : 15})
plt.ylabel("Country",  fontdict = {'fontsize' : 15})
plt.title("Top 20 countries with Confirmed and Recovered COVID-19 cases",  fontdict = {'fontsize' : 15})
plt.legend()
plt.show()
```



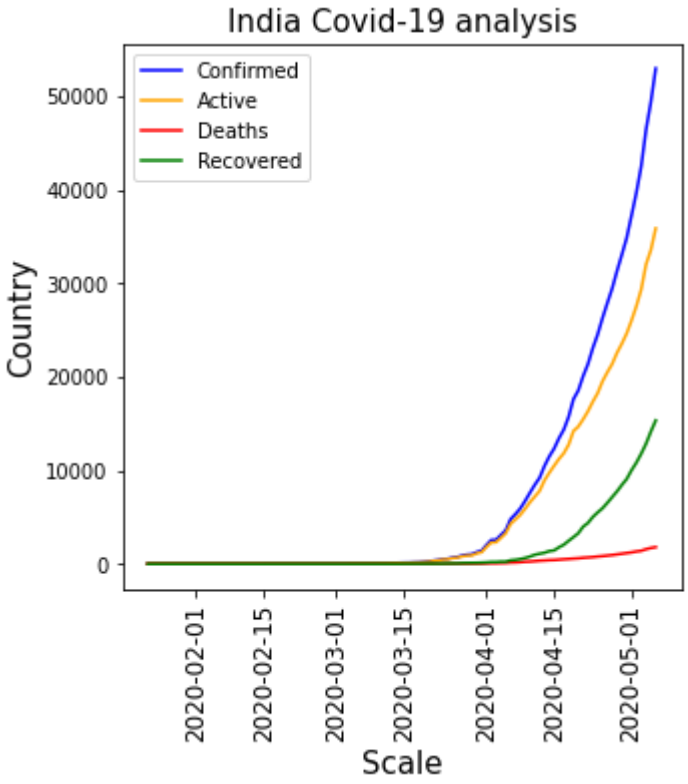
India - COVID-19 Analysis

In [23]:
```
#### Make some dataframes for some countries
India = df[df['Country']=='India']
India = India.groupby('Date')['Recovered','Deaths','Active','Confirmed'].sum().reset_index()
India.tail()
```

Out[23]:

	Date	Recovered	Deaths	Active	Confirmed
101	2020-05-02	10819	1323	27557	39699
102	2020-05-03	11775	1391	29339	42505
103	2020-05-04	12847	1566	32024	46437
104	2020-05-05	14142	1693	33565	49400
105	2020-05-06	15331	1785	35871	52987

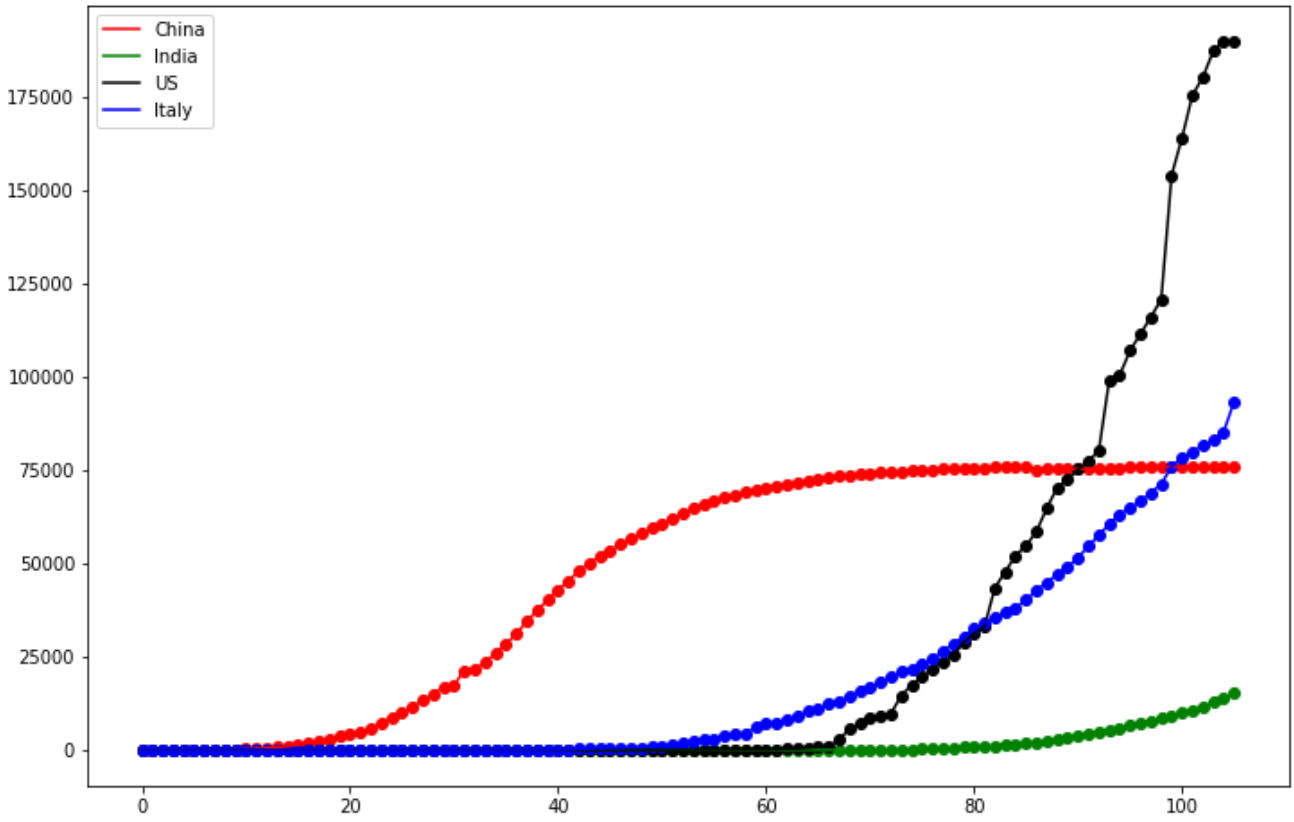
In [24]:
```
plt.figure(figsize=(5,5))
plt.plot(India['Date'],India['Confirmed'],color='blue', label = "Confirmed")
plt.plot(India['Date'],India['Active'],color='orange', label = "Active")
plt.plot(India['Date'],India['Deaths'],color='red', label = "Deaths")
plt.plot(India['Date'],India['Recovered'],color='green', label = "Recovered")
plt.tick_params(axis="x",labelrotation=90, labelsize = 12)
plt.xlabel("Scale",  fontdict = {'fontsize' : 15})
plt.ylabel("Country",  fontdict = {'fontsize' : 15})
plt.title("India Covid-19 analysis",  fontdict = {'fontsize' : 15})
plt.legend()
plt.show()
```



Recovered Comparision - India, Italy, China & US

In [25]:
```
### Italy , US, China & India
Italy = df[df['Country']=='Italy']
Italy = Italy.groupby('Date')['Recovered','Deaths','Active','Confirmed'].sum().reset_
index()
US = df[df['Country']=='US']
US = US.groupby('Date')['Recovered','Deaths','Active','Confirmed'].sum().reset_index
()
China = df[df['Country']=='China']
China = China.groupby('Date')['Recovered','Deaths','Active','Confirmed'].sum().reset_
index()
```

In [26]:
```
plt.figure(figsize=(12,8))
plt.plot(China.index,China['Recovered'],color='Red',label='China')
plt.plot(India.index,India['Recovered'],color='Green',label='India')
plt.plot(US.index,US['Recovered'],color='Black',label='US')
plt.plot(Italy.index,Italy['Recovered'],color='Blue',label='Italy')
plt.scatter(China.index,China['Recovered'],color='Red')
plt.scatter(India.index,India['Recovered'],color='Green')
plt.scatter(US.index,US['Recovered'],color='Black')
plt.scatter(Italy.index,Italy['Recovered'],color='Blue')
plt.legend(loc=2)
plt.show()
```



Prediction and Forecasting

In [27]:
```
#### Library - Fbprophet
#### Created by Facebook Company for time series analysis
#### The two column should be named as ds(date) and y(data)
#### Where to Apply time series -
#### 1. When the data is not constant
#### 2. When the data is not following any function
```

In [28]:
```
from fbprophet import Prophet
```

In [29]:
```
df.head()
```

Out[29]:

	Country	Lat	Long	Date	Confirmed	Deaths	Recovered	Active
0	Afghanistan	33.0000	65.0000	2020-01-22	0	0	0	0
1	Albania	41.1533	20.1683	2020-01-22	0	0	0	0
2	Algeria	28.0339	1.6596	2020-01-22	0	0	0	0
3	Andorra	42.5063	1.5218	2020-01-22	0	0	0	0
4	Angola	-11.2027	17.8739	2020-01-22	0	0	0	0

In [30]:
```
confirmed = df.groupby('Date')['Confirmed'].sum().reset_index()
confirmed.head()
```

Out[30]:

	Date	Confirmed
0	2020-01-22	555
1	2020-01-23	654
2	2020-01-24	941
3	2020-01-25	1434
4	2020-01-26	2118

In [31]:
```
confirmed.columns = ['ds','y']
```

In [32]:
```
#### Build The Model
model = Prophet(interval_width=0.95)
```

In [33]:
```
### Train the model
model.fit(confirmed)
```

```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=Tru
e to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True
to override this.
```

Out[33]:
```
<fbprophet.forecaster.Prophet at 0x26ad32a2c08>
```

In [34]:
```
future_dates = model.make_future_dataframe(periods=7)
future_dates.tail(10)
```

Out[34]:

	ds
103	2020-05-04
104	2020-05-05
105	2020-05-06
106	2020-05-07
107	2020-05-08
108	2020-05-09
109	2020-05-10
110	2020-05-11
111	2020-05-12
112	2020-05-13

In [35]:
```
forecast = model.predict(future_dates)
```

In [36]:
```
forecast[['ds','yhat','yhat_lower','yhat_upper']].tail(10)
```

Out[36]:

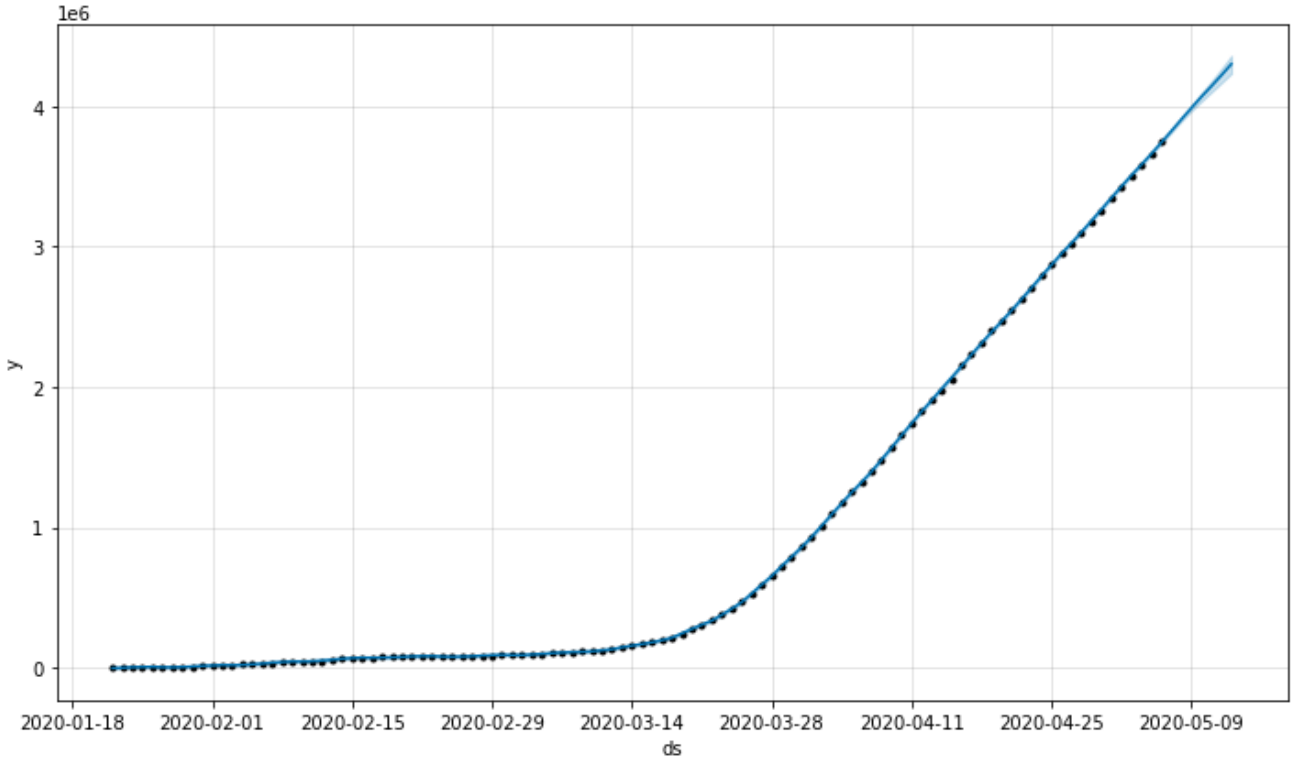
	ds	yhat	yhat_lower	yhat_upper
103	2020-05-04	3.584437e+06	3.575971e+06	3.592767e+06
104	2020-05-05	3.662043e+06	3.653052e+06	3.670132e+06
105	2020-05-06	3.742727e+06	3.734510e+06	3.751179e+06
106	2020-05-07	3.824092e+06	3.815124e+06	3.834171e+06
107	2020-05-08	3.907160e+06	3.895138e+06	3.918149e+06
108	2020-05-09	3.987840e+06	3.968499e+06	4.007947e+06
109	2020-05-10	4.067524e+06	4.036644e+06	4.095628e+06
110	2020-05-11	4.142802e+06	4.102453e+06	4.180020e+06
111	2020-05-12	4.220407e+06	4.168035e+06	4.268686e+06
112	2020-05-13	4.301092e+06	4.235185e+06	4.361759e+06

In [37]:
```
confirmed.tail()
```

Out[37]:

	ds	y
101	2020-05-02	3427337
102	2020-05-03	3506723
103	2020-05-04	3583049
104	2020-05-05	3662685
105	2020-05-06	3755335

In [38]:
```
confirmed_plot = model.plot(forecast)
```



In [39]:
```
##### Top 20 Countries having most number of Active Cases
# top_data = latest_date.groupby('Country')['Confirmed','Recovered'].sum()
# top_data = top_data.sort_values('Confirmed',ascending=False)
# top_data = top_data.reset_index()
# top_data.head()
# top_20 = top_data.head(20)
```

In [40]:
```
# plt.figure(figsize=(8,8))
# sns.barplot(top_20['Confirmed'],top_20['Country'],color='red')
# sns.barplot(top_20['Recovered'],top_20['Country'],color='green')
# plt.show()
```

INDIA - Covid Prediction

In [41]:
```
India_Prediction = df[df['Country']=='India']
India_Prediction.tail()
```

Out[41]:

	Country	Lat	Long	Date	Confirmed	Deaths	Recovered	Active
26795	India	21.0	78.0	2020-05-02	39699	1323	10819	27557
27059	India	21.0	78.0	2020-05-03	42505	1391	11775	29339
27323	India	21.0	78.0	2020-05-04	46437	1566	12847	32024
27587	India	21.0	78.0	2020-05-05	49400	1693	14142	33565
27851	India	21.0	78.0	2020-05-06	52987	1785	15331	35871

In [42]:
```
grouped_India_Prediction = India_Prediction.groupby('Date')['Confirmed'].sum().reset_
index()
grouped_India_Prediction.rename(columns={'Date':'ds','Confirmed':'y'},inplace=True)
grouped_India_Prediction.tail()
```

Out[42]:

	ds	y
101	2020-05-02	39699
102	2020-05-03	42505
103	2020-05-04	46437
104	2020-05-05	49400
105	2020-05-06	52987

In [43]:
```
#### Build The Model
india_model = Prophet(interval_width=0.95)
```

In [44]:
```
### Train the model
india_model.fit(grouped_India_Prediction)
```

```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=Tru
e to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True
to override this.
```

Out[44]:
```
<fbprophet.forecaster.Prophet at 0x26ad38c22c8>
```

In [45]:
```
future_prediction_dates = model.make_future_dataframe(periods=7)
future_prediction_dates.tail(7)
```

Out[45]:

	ds
106	2020-05-07
107	2020-05-08
108	2020-05-09
109	2020-05-10
110	2020-05-11
111	2020-05-12
112	2020-05-13

In [46]:
```
prediction = india_model.predict(future_prediction_dates)
prediction.head(3)
```

Out[46]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_term
0	2020-01-22	-114.608959	-1812.995443	2274.271553	-114.608959	-114.608959	251.411646	251
1	2020-01-23	-109.756882	-2477.435452	2046.006106	-109.756882	-109.756882	-160.604681	-160
2	2020-01-24	-104.904806	-2205.618824	1882.179924	-104.904806	-104.904806	-183.668113	-183

In [47]:
```
prediction_result = prediction[['ds','yhat','yhat_lower','yhat_upper']]
prediction_result.rename(columns={'yhat':'predicted','yhat_lower':'lower_limit','yhat_upper':'upper_limit'}, inplace=True)
prediction_result.tail(8)
```

```
C:\Users\VAIO\Anaconda3\lib\site-packages\pandas\core\frame.py:4223: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

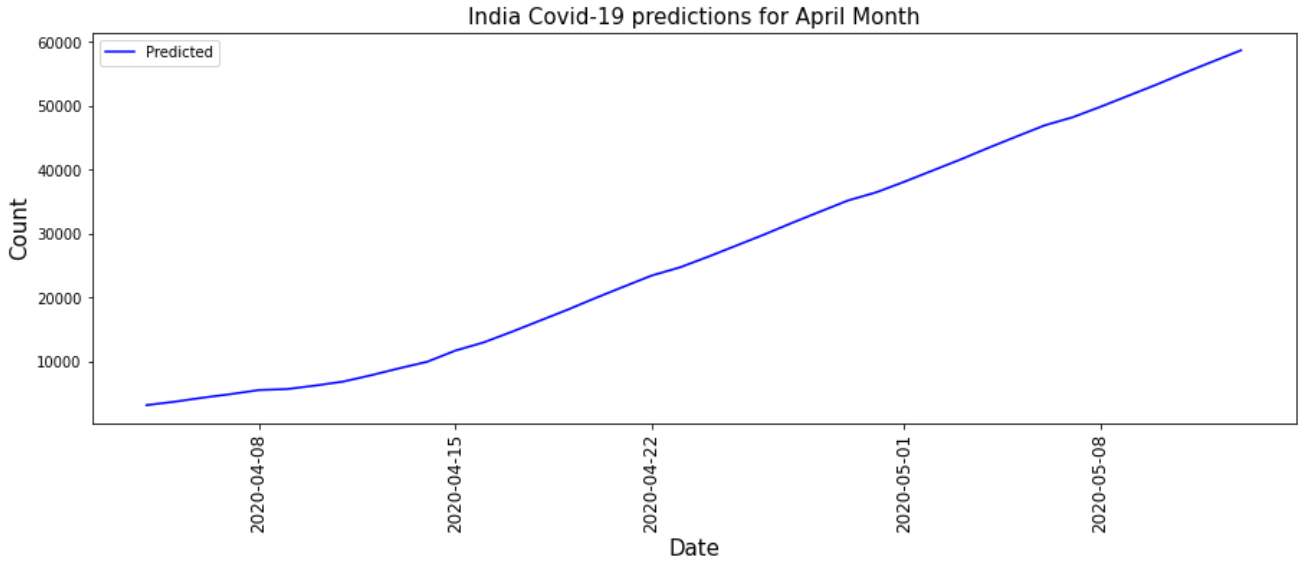
Out[47]:

	ds	predicted	lower_limit	upper_limit
105	2020-05-06	46955.527987	44889.099426	49107.128605
106	2020-05-07	48223.755251	46239.294201	50302.405435
107	2020-05-08	49880.935409	47734.298883	51894.250027
108	2020-05-09	51621.516891	49611.538592	53821.375868
109	2020-05-10	53362.234670	51248.314580	55465.249111
110	2020-05-11	55200.396047	52912.347405	57318.937536
111	2020-05-12	56969.563530	54696.073200	59175.735401
112	2020-05-13	58717.233119	56446.694700	61052.085453

Trend chart for month April till mentioned forcast day

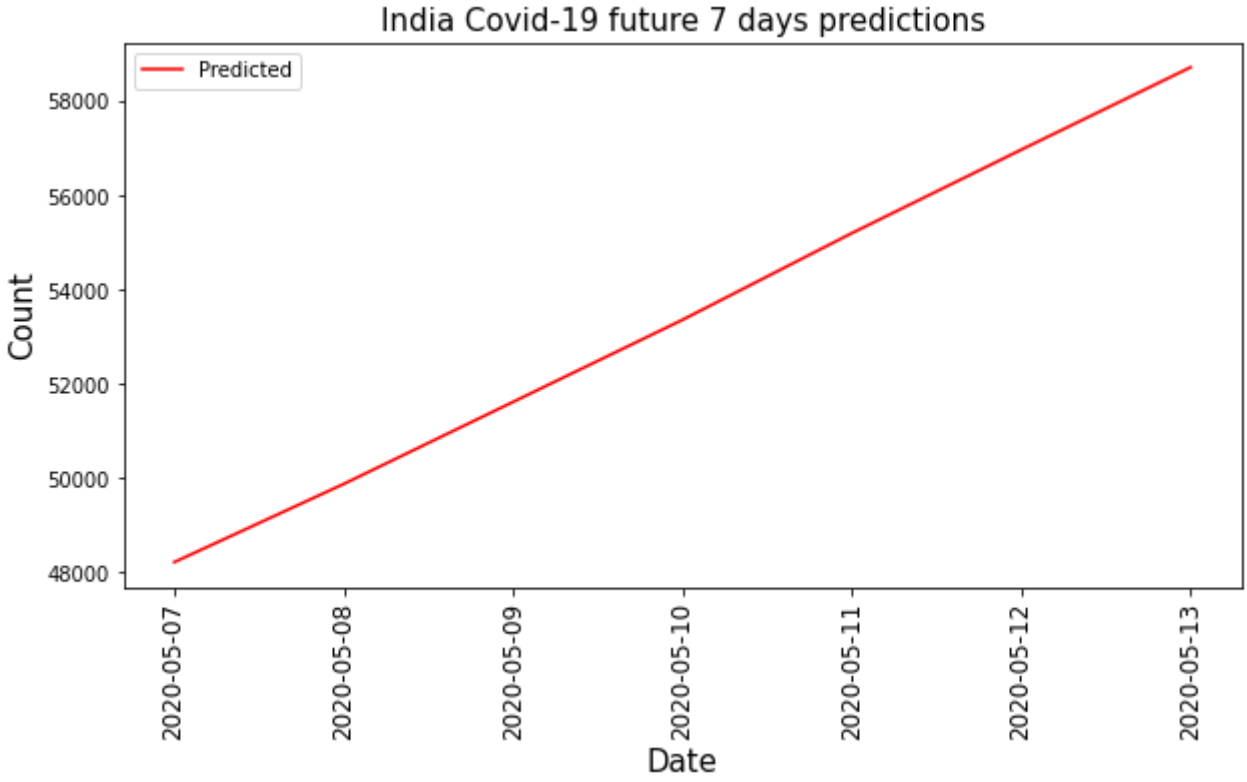
In [48]:
```
prediction_month = prediction_result.iloc[-40:,:]

#import datetime
plt.figure(figsize=(15,5))
plt.plot(prediction_month['ds'],prediction_month['predicted'],color='blue', label =
"Predicted")
plt.tick_params(axis="x",labelrotation=90, labelsize = 12)
#plt.xlim([datetime.date(2020,4,1), datetime.date(2020, 4, 15)])
plt.xlabel("Date",  fontdict = {'fontsize' : 15})
plt.ylabel("Count",  fontdict = {'fontsize' : 15})
plt.title("India Covid-19 predictions for April Month",  fontdict = {'fontsize' : 15
})
plt.legend()
plt.show()
```



In [49]:
```
prediction_dates = prediction_result.iloc[-7:,:]

plt.figure(figsize=(10,5))
plt.plot(prediction_dates['ds'],prediction_dates['predicted'],color='red', label = "P
redicted")
plt.tick_params(axis="x",labelrotation=90, labelsize = 12)
plt.xlabel("Date",  fontdict = {'fontsize' : 15})
plt.ylabel("Count",  fontdict = {'fontsize' : 15})
plt.title("India Covid-19 future 7 days predictions",  fontdict = {'fontsize' : 15})
plt.legend()
plt.show()
```



In [ ]: