Visual Analysis of Covid-19 in world from 22-01-2020 to 12-06-2020 using Machine Learning

Karanam Santoshachandra Rao
Asst.Professor, Dept of CSE
Centurion University of Technology and Management, Odisa, India.

Avinash Alugolu
Asst.Professor, Dept of CSE
Centurion University of Technology and Management, Andhra Pradesh, India.

Revalla Vidyasri
Centurion University of Technology and Management, Odisa, India.

Talasu Greeshma
Centurion University of Technology and Management, Odisa, India.

Sai Kotturu
Centurion University of Technology and Management, Odisa, India.

Vysyaraju Sai Sirisha
Centurion University of Technology and Management, Odisa, India.

Abstract

Novel coronavirus disease (COVID-19)[1] was first identified in China, which eventually became a major global health concern due to its pathogenicity and widespread distribution around the world. In this paper Authors have analyzed Covid-19 cases from January 22nd 2020 to June 12th 2020 using machine learning visualization techniques based on some constrains like: 1)Countries having higher covid-19 confirmed cases from 22/01/2020 to 12/06/2020, 2) Total number of covid-19 confirmed cases on 12/06/2020, 3)Total number of death cases of each country on 12/06/2020, 4) Top 10 Countries with Covid-19 Confirmed cases and Death cases, 5) Total date wise covid-19 cases from 22/01/2020 to 12/06/2020,6) Top 10 Countries Recovered Cases and Confirmed Cases,7)Total number of covid_19 cases in India on 12/06/2020, 8) Countries having higher covid-19 cases from 22/01/2020 to 12/06/2020 using plotly,9) Top 10 Countries having higher covid-19 cases from 01/04/20 to 12/06/2020 using Matplotlib,10) Top countries affected by covid_19 till 12th June 2020.

Keywords: COVID-19, Coronavirus, Machine Learning, Visualization, Plotting, Python, Packages, Numpy, Pandas, Seaborn, Matplotlib, Plotly.

Introduction

Analysis of mentioned 10 points are based on python [2] best and top libraries like numpy[3], pandas and visualization libraries like matplotlib[4], plotlly[5]. Generally, Python is high-level general purpose programming and an interpreted language. Developed by Guido van Rossum and primarily released in 1991, Python's design intention emphasizes code readability & understanding with its easy use of significant whitespace or indentation. Its object-oriented approach targets to help developers write unambiguous, very logical code for tiny and large-scale projects.

"Visualization is worth a many thousand words". We are all aware of this expression. This especially applies when we trying to deliver the insight received from the analysis of large datasets. Data visualization plays a major role in the visualization of both tiny and very large-scale data.

One of the important skills of a data scientist or programmer is the ability to express a compelling story, displaying data visually and findings in a possible and stimulating way. Studying how to use a software tool to visualize data-sets will also permits you to fetch information, clear understand the data, and make more perfect decisions.

The main motto of this Data Visualization with Python is to analyze above mentioned 10 points and present that information in the form that makes sense to people. Various techniques have been used for presenting data visually with the help of python libraries namely Matplotlib, Seaborn and Plotly.

Following python library[6] code is required to import primarily to analyze the data with the help of various function and methods.

```
import numpy as np
import pandas as pd
from datetime import date
import matplotlib.pyplot as plt
import matplotlib.style as style
import matplotlib.ticker as ticker
import matplotlib.animation as animation from IPython.display
import HTML import plotly.express as px
import plotly.express as px
importplotly.graph_objects as go
```

Understanding the dataset

Here, we have considered the data from the dataset[7] “covid_19_clean_complete_updated” for analysis. Basic idea on dataset is given by following python code[8].

Step1: Reads the Murder_Victim_age_sex.csv file
covid_df=pd.read_csv("covid_19_clean_complete_updated.csv")
print(covid_df.head())

This above code gives the first 5 rows from the data set[9].

Fig 1 shows the output of the following python code for dataset summarizing the columns.
Code: covid_df.describe()

```
covid_df.describe()
```

	Lat	Long	Confirmed	Deaths	Recovered
count	23580.000000	23580.000000	23580.000000	23580.000000	23580.000000
mean	21.433571	22.597991	1890.858863	108.744826	464.683885
std	24.740944	70.570993	18541.984499	1187.076831	4229.313260
min	-51.796300	-135.000000	-1.000000	-1.000000	0.000000
25%	7.000000	-19.020800	0.000000	0.000000	0.000000
50%	23.659750	20.921188	3.000000	0.000000	0.000000
75%	41.204400	81.000000	143.000000	1.000000	13.000000
max	71.706900	178.065000	784326.000000	42094.000000	91500.000000

Fig 1: Dataset summarizing the columns

Following python code is used to check for null values in the dataset
Code: covid_df.isnull().any()

```
covid_df.isnull().any()
```

```
Province/State     True
Country/Region    False
Lat               False
Long              False
Date              False
Confirmed         False
Deaths            False
Recovered         False
dtype: bool
```

Fig 2: Identifying null values in the dataset

Following python code is used to replace the null values in the dataset
Code: df2 = covid_df.fillna(method="bfill", axis="columns")

```
df2 = covid_df.fillna(method="bfill", axis="columns")
print(df2[["Province/State","Country/Region"]])
```

```
                Province/State         Country/Region
0                  Afghanistan            Afghanistan
1                      Albania                Albania
2                      Algeria                Algeria
3                      Andorra                Andorra
4                       Angola                 Angola
...                        ...                    ...
23575  Saint Pierre and Miquelon                France
23576                South Sudan            South Sudan
23577             Western Sahara         Western Sahara
23578      Sao Tome and Principe  Sao Tome and Principe
23579                      Yemen                  Yemen

[23580 rows x 2 columns]
```

Fig 3: replacing null values in the dataset

Following code is to identify shape of the dataset
Code: covid_df.shape

```
: covid_df.shape
: (37323, 8)
```

Fig 4: Identifying shape of the dataset

1. Countries having higher covid-19 confirmed cases from 22/01/2020 to 12/06/2020.

Takes Province/State, Country/Region, Date, Confirmed considered as input for this task.Fig 5 is its visualization.

Code:

```
df2=covid_df.drop(["Lat","Long","Deaths","Recovered"],axis=1)
df2.Date=pd.to_datetime(df2.Date)
df2.rename(columns= {"Province/State":'state','Country/Region':"Country",},inplace =True)

#assigning each color to each country
colors = dict(zip(["Afghanistan","Albania","Algeria","Andorra","Angola","Antigua and Barbuda","Argentina","Armenia","Australia","Austria","Azerbaijan","Bahamas","Bahrain","Bangladesh","Barbados","Belarus","Belgium","Benin","Bhutan","Bolivia","Bosnia and Herzegovina","Brazil","Brunei","Bulgaria","Burkina Faso","Cabo Verde","Cambodia","Cameroon","Canada","Central African Republic","Chad","Chile","China","Colombia","CongoBrazzaville)","Congo (Kinshasa)","Costa
```

Rica","Coted'Ivoire","Croatia","DiamondPrincess","Cuba","Cyprus","Czechia","Denmark","Djibouti","DominicanRepublic","Ecuador","Egypt","ElSalvador","EquatorialGuinea","Eritrea","Estonia","Eswatini","Ethiopia","Fiji","Finland","France","Gabon","Gambia","Georgia","Germany","Ghana","Greece","Guatemala","Guinea","Guyana","Haiti","HolySee","Honduras","Hungary","Iceland","India","Indonesia","Iran","Iraq","Ireland","Israel","Italy","Jamaica","Japan","Jordan","Kazakhstan","Kenya","SouthKorea","Kuwait","Kyrgyzstan","Latvia","Lebanon","Liberia","Liechtenstein","Lithuania","Luxembourg","Madagascar","Malaysia","Maldives","Malta","Mauritania","Mauritius","Mexico","Moldova","Monaco","Mongolia","Montenegro","Morocco","Namibia","Nepal","Neherlands","Nicaragua","Niger","Nigeria","NorthMacedonia","Norway","Oman","Pakistan","Panama","PapuaNewGuinea","Paraguay","Peru","Philippines","Poland","Portugal","Qatar","Romania","Russia","Rwanda","Saint Lucia","Saint Vincent and the Grenadines","San Marino","SaudiArabia","Senegal","Serbia","Seychelles","Singapore","Slovakia","Slovenia","Somalia","SouthAfrica","Spain","SriLanka","Sudan","Suriname","Sweden","Switzerland","Taiwan*","Tanzania","Thailand","Togo","Trinidad and Tobago", "Tunisia", "Turkey", "Uganda", "Ukraine","United Arab Emirates","United Kingdom","Uruguay","US","Uzbekistan", "Venezuela","Vietnam","Zambia","Zimbabwe","Canada","Dominica","Grenada","Mozambique, "Syria","Timor-Leste","Belize","Laos","Libya","West Bank and Gaza","Guinea-Bissau","Mali", "Saint Kitts and Nevis","Kosovo","Burma","MS Zaandam","Botswana","Burundi","Sierra Leone","Malawi","South Sudan","Western Sahara","Sao Tome and Principe","Yemen"],

["#adb0ff", "#ffb3ff", "#90d595", "#e48381", "#aafbff", "#f7bb5f", "#eafb50""g", "mediumpurple","lime","teal","liime","crimson","c","y","lightskyblue","y","m","g","rosybrown","lightcoral","indianred","brown","firebrick","maroon","darkred","red","mistyrose","salmon","tomato","darksalmon","coral","orangered","lightsalmon","sienna","seashell" ,"chocolate", "saddlebrown","sandybrown","peachpuff","peru","linen","purple","darkorange","burlywood","m","tan" ,"c" ,"blanchedalmond" ,"papayawhip", "#adb0ff","#ffb3ff", "#90d595", "#e48381", "#aafbff","#f7bb5f","#eafb50","darkorange","pink","magenta","cyan","g","m","hotpink","olive", "maroon","y","magenta","limegreen","goldenrod","deeppink","lime","lightskyblue","m","lime", "teal","palevioletred","crimson","salmon","lightgray","c","gainsboro","m","g","rosybrown","lightcoral","indianred","brown","firebrick","maroon","darkred","red","mistyrose","salmon","tomato","darksalmon","coral","orangered","lightsalmon","sienna", "seashell" ,"chocolate", "saddlebrown","sandybrown","peachpuff","peru","g","linen","darkorange","burlywood","g","tan" ,"lightskyblue" ,"blanchedalmond","papayawhip", "darkorange", "pink" ,"magenta","cyan" ,"purple","blue","hotpink","olive","deepskyblue","y","magenta","limegreen","goldenrod","deeppink","lightskyblue","crimson","c","salmon","palevioletred","lime","teal","m","c","palevioletred", "gainsboro","m","g","rosybrown","lightcoral","indianred","brown","firebrick","maroon","darkred","red","mistyrose","salmon","tomato","darksalmon","coral","orangered","lightsalmon","sienna","seashell","chocolate","saddlebrown","sandybrown","peru","linen","purple","darkorange","burlywood", "y","tan" ,"c" ,"blanchedalmond" ,"papayawhip","#adb0ff","#ffb3ff", "#90d595","coral",

```
"orangered","lightsalmon","sienna","seashell"  ,"chocolate","saddlebrown","sandybrown"
,"peachpuff ","peru" ]

))
group_lk = df2.set_index('state')['Country'].to_dict()

#for only 12/06/2020 date
fig, ax = plt.subplots(figsize=(15, 8))

def draw_barchart(current_date):
   dff = df2[df2['Date'].eq(current_date)].sort_values(by='Confirmed', ascending=True).tail(10)
   ax.clear()
   ax.barh(dff['state'], dff['Confirmed'], color=[colors[group_lk[x]] for x in dff['state']])
   dx = dff['Confirmed'].max() / 200
   for i, (Deaths, state) in enumerate(zip(dff['Confirmed'], dff['state'])):
      ax.text(Deaths-dx, i,     state,          size=14, weight=600, ha='right', va='bottom')
      ax.text(Deaths-dx, i-.25, group_lk[state], size=10, color='#444444', ha='right', va='baseline')
      ax.text(Deaths+dx, i,     f'{Deaths:,.0f}',  size=14,weight=600, ha='left',  va='center')
   ax.text(1, 0.4, current_date, transform=ax.transAxes, color='#777777', size=46, ha='right',
weight=800)
   ax.text(0, 1.06, 'Population (thousands)', transform=ax.transAxes, size=12, color='#777777')
   ax.xaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))
   ax.xaxis.set_ticks_position('top')
   ax.tick_params(axis='x', colors='#777777', labelsize=12)
   ax.set_yticks([])
   ax.margins(0, 0.01)
   ax.grid(which='major', axis='x', linestyle='-')
   ax.set_axisbelow(True)
   ax.text(0, 1.15, 'The top most countries in the world affected by corona from 22/1/2020 to
12/06/2020',transform=ax.transAxes, size=24, weight=600, ha='left', va='top')
   ax.text(1, 0, 'by @AIMLdomain', transform=ax.transAxes, color='#777777', ha='right',
         bbox=dict(facecolor='white', alpha=0.8, edgecolor='white'))
   plt.box(False)
date1="12-06-2020"
```

draw_barchart(date1)

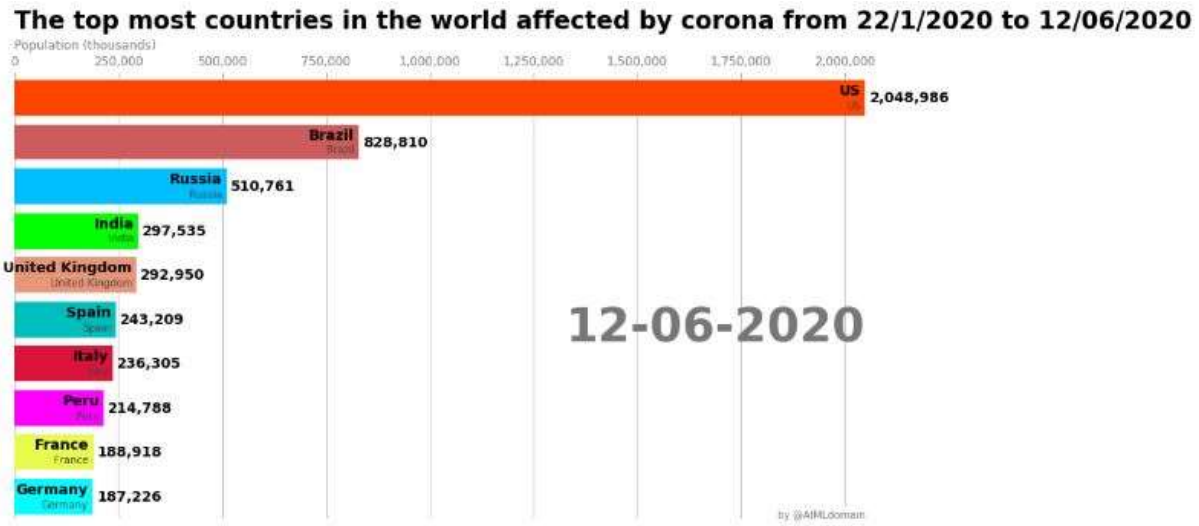

Fig 5: visualization of top 10 countries having highest covid-19 confirmed cases on 12/06/2020

Fig 5 Displays top 10 countries with number of covid confirmed cases. Here US is having higher Covid cases.On the X-axis total number of covid confirmed cases and On Y-axis Country on 12/06/2020.

#for Jan 22nd to June 12th

```
start_date=date(2020,1,22)
end_date=date(2020,6,12)
daterange=pd.date_range(start_date,end_date).strftime("%Y-%b-%d")
fig, ax = plt.subplots(figsize=(15, 8))
animator = animation.FuncAnimation(fig, draw_barchart,frames=daterange)
HTML(animator.to_jshtml())
```

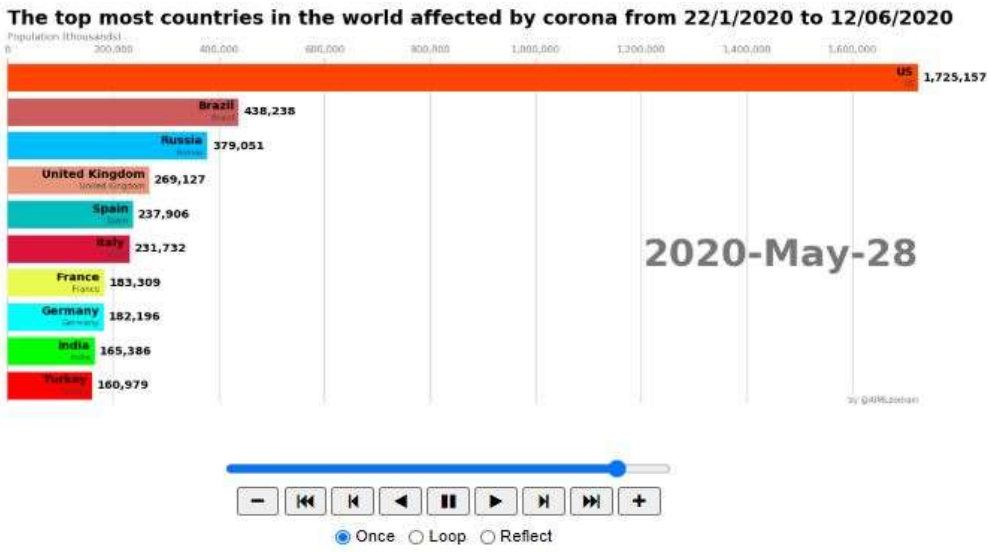

Fig 6: visualization of top 10 countries having highest covid-19 confirmed cases from 22/01/2020 to 12/06/2020

2. Total number of covid-19 confirmed cases on 12/06/2020.

Country/Region, Confirmed considered as input for this task.Fig 7 is its visualization.

Code:

```
df1=covid_df[covid_df["Date"]=="12-06-2020"]
fig = px.pie(df1,values='Confirmed', names='Country/Region',
          title='Infected rates across countries',
           hover_data=['Country/Region'], labels={'Country/Region':'Country/Region'})
fig.update_traces(textposition='inside')
fig.show()
```

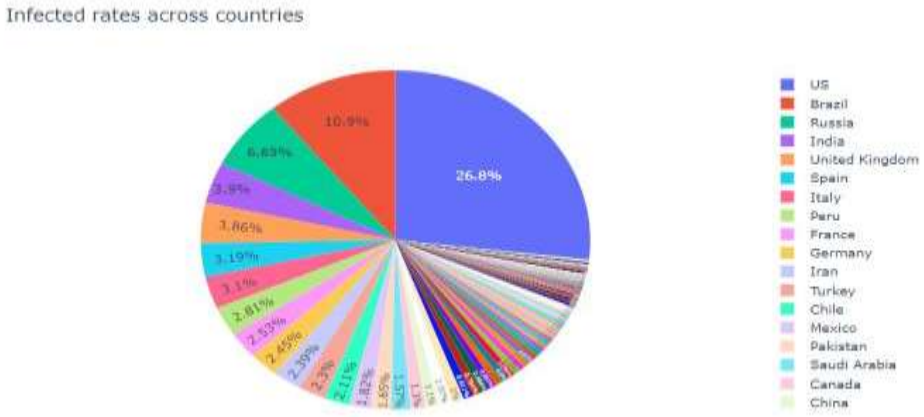

Fig 7: Total number of covid_19 confirmed cases on 12/06/2020.

3. Total number of death cases of each country on 12/06/2020.

Country/Region, Deaths considered as input for this task.Fig 8 is its visualization.

Code:

```
df1=covid_df[covid_df["Date"]=="12-06-2020"]
fig = px.bar(df1, x='Country/Region', y = 'Deaths', color='Country/Region', title ='Number of reported deaths across countries')
fig.show()
```

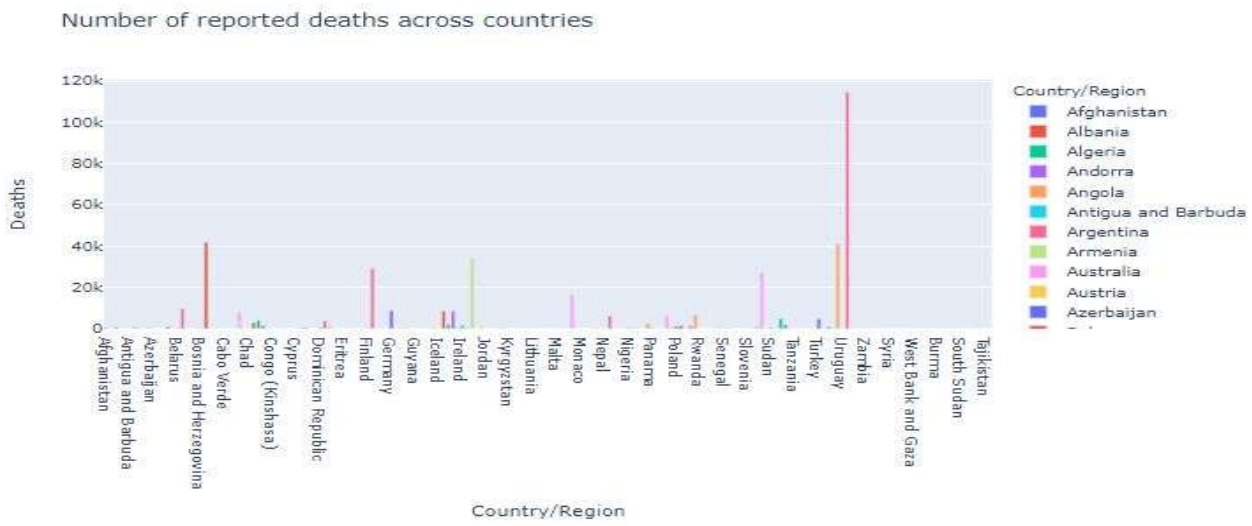

Fig 8: country wise covid_19 Death cases on 12/06/2020

4. Top 10 Countries with Covid-19 Confirmed cases and Death cases.

Country/Region, Confirmed, Deaths considered as input for this task.Fig 9 is its visualization.

Code:

```
df2=covid_df.drop(["Province/State","Lat","Long"],axis=1)
df2.Date=pd.to_datetime(df2.Date)
dff3=df2[df2["Date"]=='2020-12-06']
dff4=dff3.groupby("Country/Region").sum()
dff5=dff4.sort_values(by='Confirmed', ascending=True).tail(10)
fig = plt.figure(figsize=(14,8))
ax = fig.gca()
xs=dff5.index
ys=dff5.Confirmed
ax.plot(xs,ys,label="Confirmed", linestyle='dashed', linewidth = 3, marker='o',
markerfacecolor='red', markersize=10,)
for x,y in zip(xs,ys):
   label = "{:.0f}".format(y)
   plt.annotate(label, # this is the text
            (x,y), # this is the point to label
            textcoords="offset points", # how to position the text
            xytext=(0,10), # distance from text to points (x,y)
            ha='center', fontweight = 'bold', fontsize=12)
ax.bar(dff5.index,dff5.Deaths,color="red",label="Deaths")
for p1 in ax.patches:
```

```
    if (int(p1.get_height() != 0)):
        ax.text(p1.get_x() + p1.get_width()/2., p1.get_height(), '%d' % int(p1.get_height()),
            fontsize=12, color='black', fontweight = 'bold', ha='center', va='bottom')
plt.xlabel("Countries",fontsize=20,fontweight='bold')
plt.ylabel("No of cases",fontsize=20,fontweight='bold')
ax.set_yticks(np.arange(0, 2500000, 250000))
ax.set_yticklabels(np.arange(0, 2500000, 250000), minor=False, fontsize=12,fontweight='bold')
ax.set_xticklabels(xs,rotation=70,fontsize=16,fontweight='bold')
plt.title("Top 10 Countries with Covid-19 Confirmed cases vs Death
cases",fontsize=25,fontweight='bold')
plt.legend(fontsize=18)
plt.show()
```

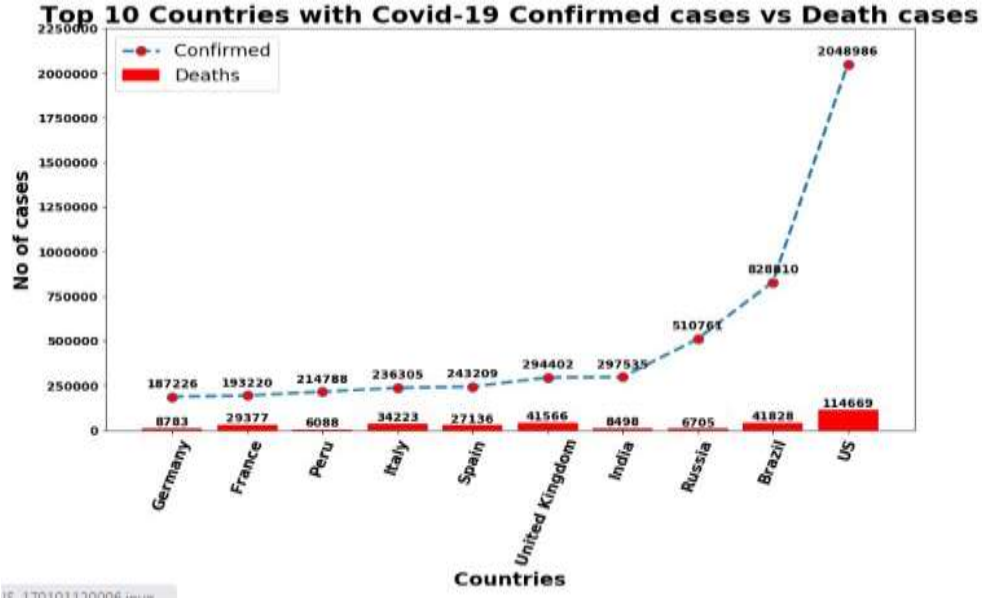

Fig 9: country wise covid_19 Confirmed cases and Death cases on 12/06/2020

5. Total date wise covid-19 cases from 22/01/2020 to 12/06/2020.

Date, Confirmed, Recovered, Deaths considered as input for this task. Fig 10 is its visualization.

Code:

```
temp = df1.groupby('Date')['Recovered', 'Deaths','Confirmed'].sum().reset_index()

from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import matplotlib
matplotlib.rc('xtick',labelsize=18)
matplotlib.rc('ytick',labelsize=18)
x=temp.Date
```

```
y=temp.Confirmed
y1=temp.Recovered
y2=temp.Deaths

fig,(ax1,ax2,ax3)=plt.subplots(nrows=3,figsize=(15,15))
fig.suptitle("Datewise Covid-19 Cases in the World",fontsize=25,fontweight='bold')
ax1.plot(x,y,marker='o',markerfacecolor="r",color='b')
ax2.plot(x,y1,marker='o',markerfacecolor="r",color='g')
ax3.plot(x,y2,marker='o',markerfacecolor="k",color='r')
ax1.legend({'Confirmed'},fontsize=20)
ax2.legend({'Recovered'},fontsize=20)
ax3.legend({'Deaths'},fontsize=20)
ax1.set_xlabel('Date',fontsize=22)
ax1.set_ylabel('No of Confirmed Cases',fontsize=18)
ax2.set_xlabel('Date',fontsize=22)
ax2.set_ylabel('No of Recovered Cases',fontsize=18)
ax3.set_xlabel('Date',fontsize=22)
ax3.set_ylabel('No of Death Cases',fontsize=18)
plt.show()
```

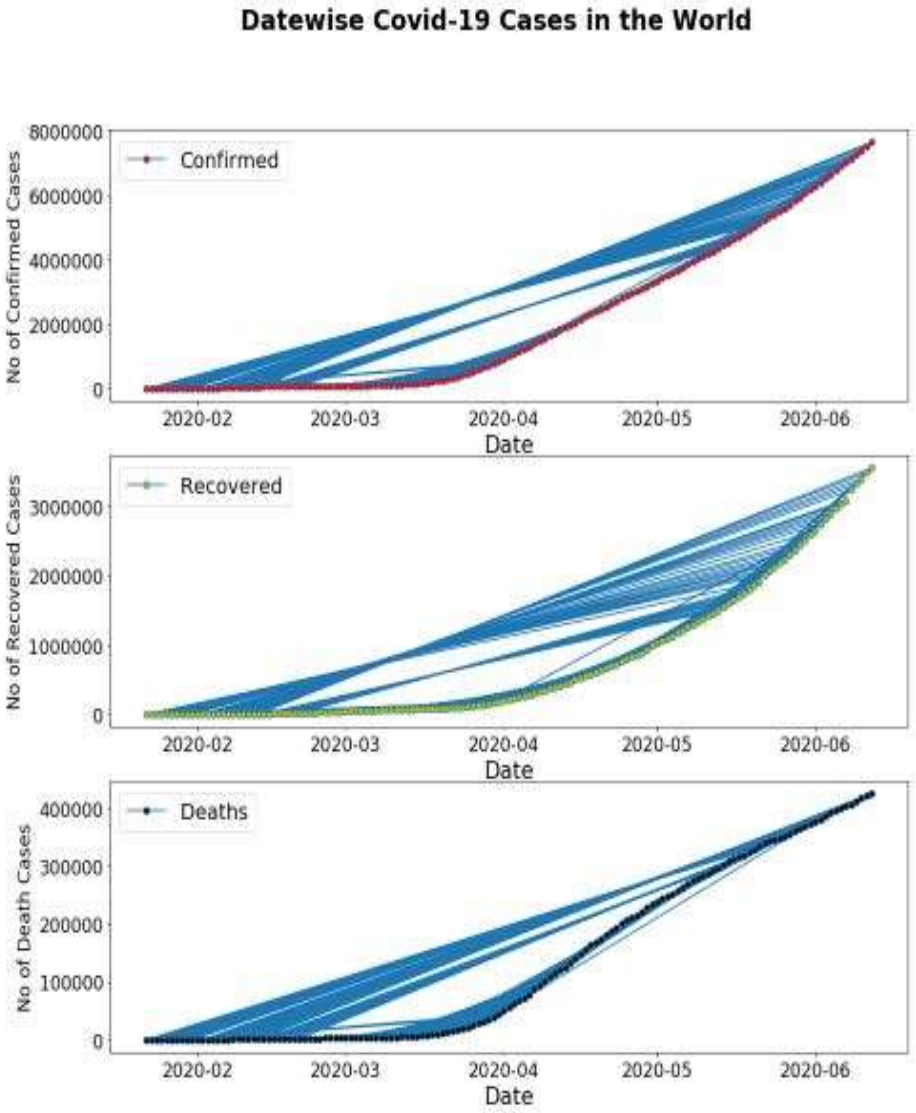

Fig 10: country wise covid_19 cases from 22/01/2020 to 12/06/2020

6. Top 10 Countries Recovered Cases and Confirmed Cases.

Country/Region, Recovered and Confirmed considered as input for this task.Fig11 is its visualization.

Code:

```
df1=covid_df.drop(["Province/State","Lat","Long"],axis=1)
df1.rename(columns= {'Country/Region':"Country"},inplace =True)
df1.Date=pd.to_datetime(df1.Date)
df2=df1[df1["Date"]=="2020-04-20"]
g1 = df2.groupby("Country")
df2=g1.sum()
df2.sort_values("Recovered",ascending=True,inplace=True)
df2=df2.tail(10)
```

```
x=df2.index
y=df2.Recovered
fig, ax = plt.subplots(figsize=(15,8))
plt.barh(x,y,color="g")
ax.set_xlabel('Count', fontsize=20)
ax.set_xticks(np.arange(0, 600000, 50000))
ax.set_xticklabels(np.arange(0,600000,50000), rotation=70,fontsize=16)
ax.set_yticks(x)
ax.set_ylabel('Recovered', fontsize=20)
ax.set_yticklabels(x, minor=False, fontsize=16)
for i in ax.patches:
   ax.text(i.get_width()+.3,i.get_y()+.38,i.get_width(),
         fontsize=15, color='k', fontweight = 'bold')

for i in ax.patches:
   ax.text(i.get_width()+.3,i.get_y()+.38,i.get_width(),
         fontsize=15, color='k', fontweight = 'bold')
m=y.max()+55000
l1=list(df2.Confirmed)
l=len(l1)
for i in range(len(l1)):
   ax.text(m, i, l1[i], bbox={'facecolor': 'red', 'alpha': 0.5, 'pad': 8},fontsize=15)

ax.text(600000, 10,"Confirmed cases",fontsize=15,fontweight='bold')
plt.title("Top 10 Countires Recovered Cases vs Confirmed Cases", fontsize = 20,
fontweight='bold')
plt.show()
```

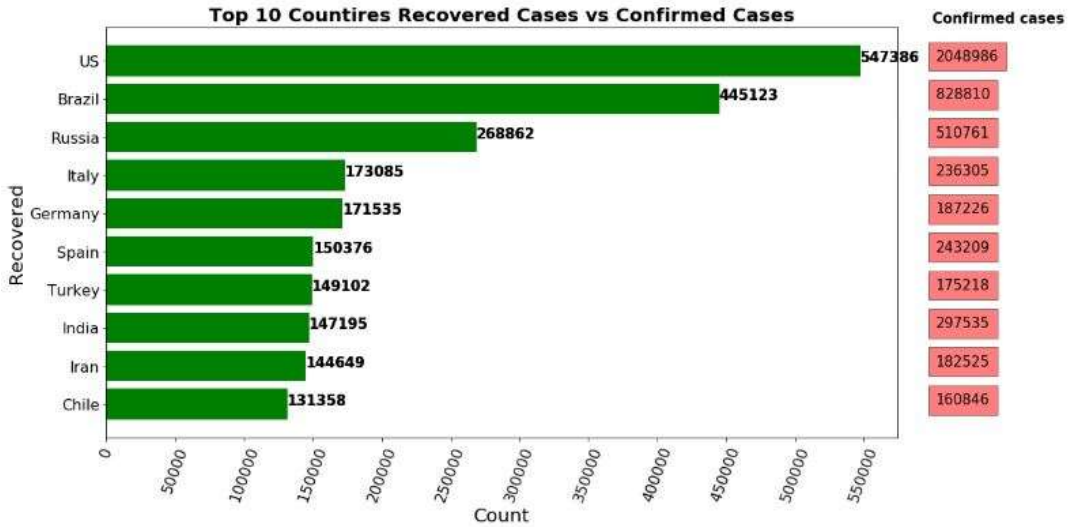

Fig 11: country wise covid_19 Recovered cases and Confirmed cases on 12/06/2020

7. Total number of covid_19 cases on India on 12/06/2020.

Country/Region, Confirmed, Recovered, Deaths and Date considered as input for this task.Fig12 is its visualization.

Code:

```
df2=df1[df1["Date"]=="2020-12-06"]
g1 = df2.groupby("Country")
df2=g1.sum()
df2.sort_values("Recovered",ascending=True,inplace=True)
x=df2[df2.index=="India"]

fig = plt.figure(figsize=(10,8))
ax = fig.add_axes([0,0,1,1])
#=df2.tail(5)
ax.axis('equal')
cases = ["Confirmed","Deaths","Recovered"]
x1=x.Confirmed
x2=x.Deaths
x3=x.Recovered
c = [x1,x2,x3]
explode = [0, 0, 0.1]
ax.pie(c, labels = cases,autopct='%1.2f%%',explode = explode,textprops={'fontsize':20})
plt.legend(fontsize=20)
plt.title("Covid_19 Cases in India",fontsize=30)
plt.show()
```

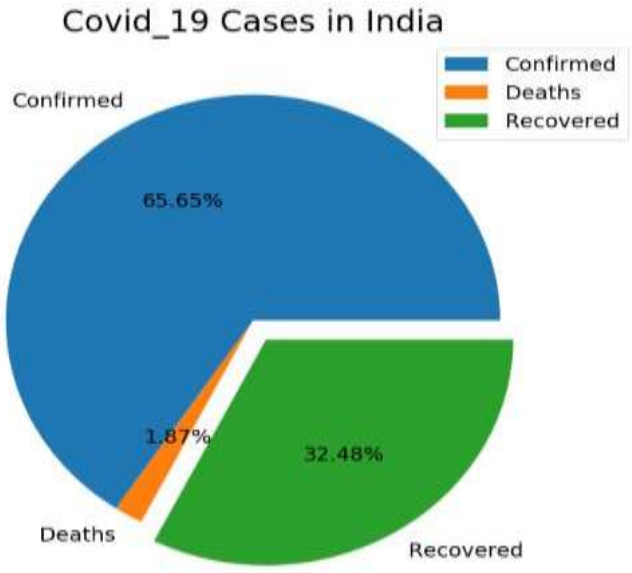

Fig 12: Total number of Covid_19 cases in India on 12/06/2020

8. Countries having higher covid-19 cases from 22/01/2020 to 12/06/2020 using plotly.

Takes Province/State, Country/Region, Lat, Long,Date, Confirmed,Recovered, Deaths considered as input for this task.Fig 13 is its visualization

Code:

```
df5=covid_df
normlised_data_C=[value/df5['Confirmed'].mean()+10forvalueindf5['Confirmed']]fig_map=px.scatter_mapbox(df5,lat="Lat",lon="Long",color="Confirmed",size=normlised_data_C,color_continuous_scale="Rainbow",size_max=50,animation_frame='Date',center=dict({'lat':32,'lon':4}),zoom=0.7,hover_data=['Country/Region'])
fig_map.update_layout(mapbox_style="carto-positron",width=900,title='Covid-19 Confirmed Cases 22/01/20 - 12/06/20',height=700)\
fig_map.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig_map.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"]=200
fig_map.layout.sliders[0].currentvalue.xanchor="left"
fig_map.layout.sliders[0].currentvalue.offset=-100
fig_map.layout.sliders[0].currentvalue.prefix=""
fig_map.layout.sliders[0].len=.9
fig_map.layout.sliders[0].currentvalue.font.color="indianred"
fig_map.layout.sliders[0].currentvalue.font.size=20
fig_map.layout.sliders[0].y=1.1
fig_map.layout.sliders[0].x=0.15
fig_map.layout.updatemenus[0].y=1.27
                                    fig_map.show()
```

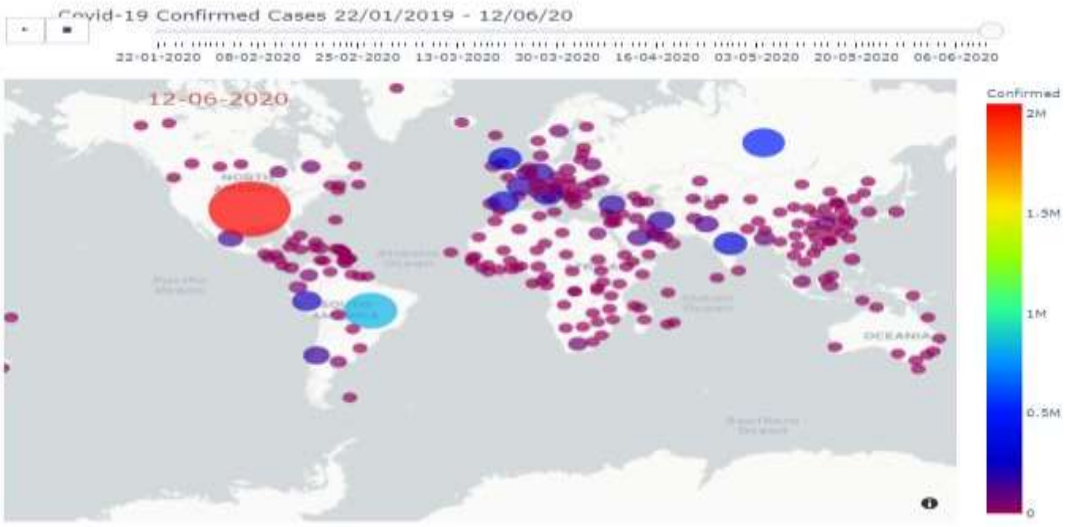

Fig 13: visualization of countries having highest covid-19 cases from 22/01/2020 to 12/06/2020

9. Top 10 Countries having higher covid-19 cases from 22/01/2020 to 20/04/2020 using Matplotlib.

Takes Province/State, Country/Region, Date, Confirmed, Recovered, Deaths considered as input for this task.Fig 14 is its visualization.

Code:

```
df1=covid_df.drop(["Lat","Long"],axis=1)
df2=df1.fillna(method="bfill",axis="columns")# axis is either "index" or "columns" (filling with backward column value)
df2.Date=pd.to_datetime(df2.Date)
df2.rename(columns={"Province/State":'name','Country/Region':"group",},inplace=True)
importnumpyasnp
fig,ax=plt.subplots(figsize=(15,12))


defdraw_barchart1(current_date):
dff1=df2[df2['Date'].eq(current_date)].sort_values(by='Confirmed',ascending=True).tail(10)
#print(dff1)
ax.clear()
N=10
ind=np.arange(N)# the x locations for the groups
width=0.35
rects1=ax.barh(ind,dff1.Deaths,width,color='r',label="Deaths")
rects2=ax.barh(ind+width,dff1.Recovered,width,color='limegreen',label="Recovered")
rects3=ax.barh(ind+width*2,dff1.Confirmed,width,color='royalblue',label="Confirmed")


defautolabel(rects):
foriinrects:
ax.text(i.get_width()+.3,i.get_y()+.38,i.get_width(),fontsize=14,color='k',fontweight='bold')


autolabel(rects3)
autolabel(rects2)
autolabel(rects1)
ax.text(1,0.4,current_date,transform=ax.transAxes,color='#777777',size=46,ha='right',weight=800)
ax.text(0,1.06,'Population (thousands)',transform=ax.transAxes,size=12,color='#777777')
ax.xaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))
ax.xaxis.set_ticks_position('top')
ax.tick_params(axis='x',colors='#777777',labelsize=12)
ax.set_yticks(ind+width)
ax.set_yticklabels(dff1.group,fontsize=16,fontweight='bold')
ax.margins(0,0.01)
ax.grid(which='major',axis='x',linestyle='-')
```

```
ax.set_axisbelow(True)
ax.text(0,1.15,'The top most countries Covid-19 Cases 22/1/2020 - 12/06/202020',
transform=ax.transAxes,size=24,weight=600,ha='left',va='top')
ax.text(1,0,'by @AIMLdomain',transform=ax.transAxes,color='#777777',ha='right',
bbox=dict(facecolor='white',alpha=0.8,edgecolor='white'))
plt.box(False)

ax.legend(fontsize=20)

date1="2020-04-01"
draw_barchart1(date1)
```

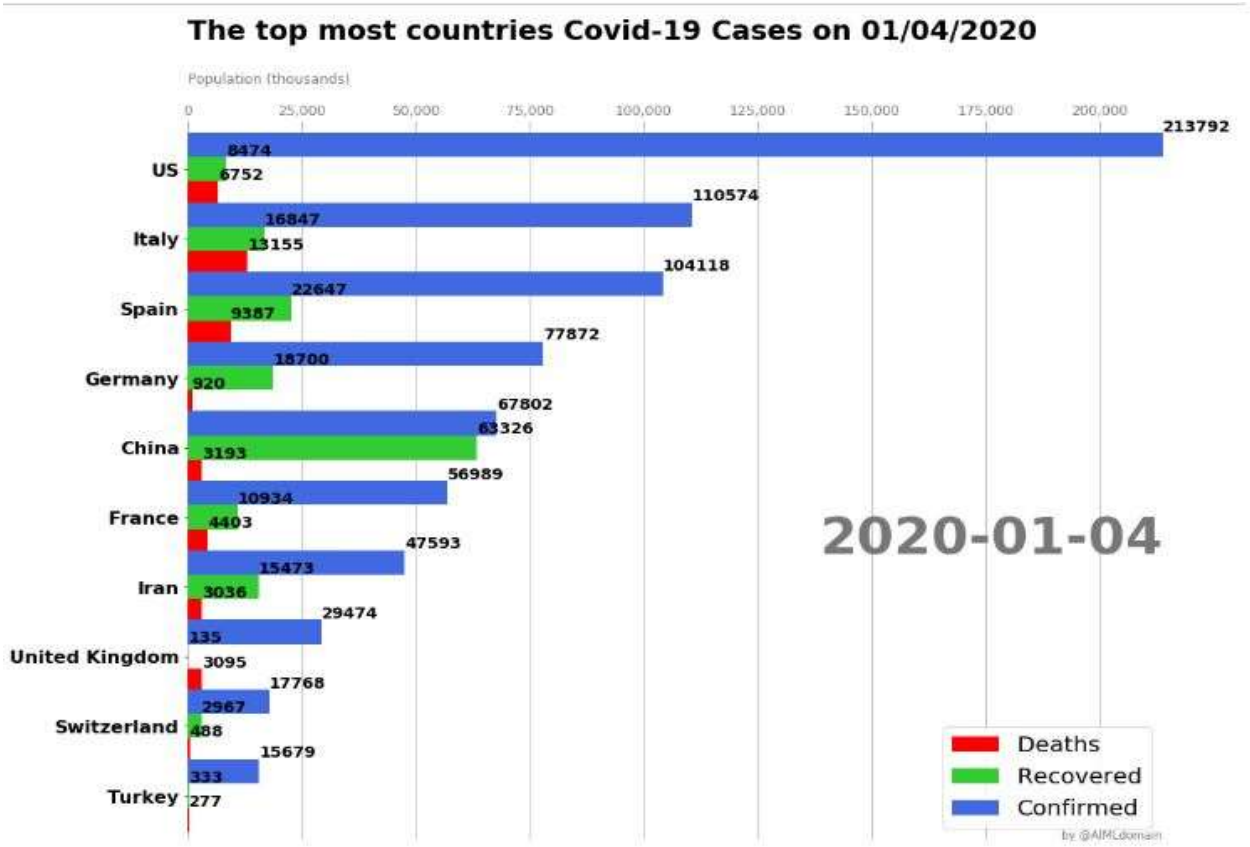

Fig 14: visualization of top 10 countries having highest covid-19 cases on 01/04/20

```
start_date=date(2020,1,22)
end_date=date(2020,4,20)
daterange=pd.date_range(start_date,end_date).strftime("%Y-%b-%d")
#print(daterange)
fig,ax=plt.subplots(figsize=(15,8))
animator1=animation.FuncAnimation(fig,draw_barchart1,frames=daterange)
HTML(animator1.to_jshtml())
```

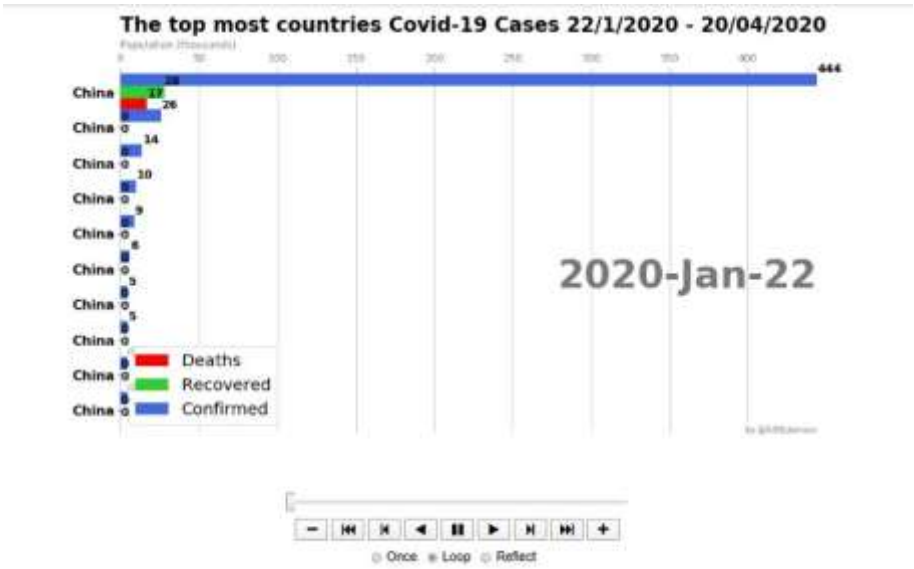

Fig 14: visualization of top countries having highest covid-19 cases from 22/01/20 to 20/04/2020

10.Top countries affected by covid_19 up to 20thApril.

Takes Province/State, Country/Region, Date, Confirmed, Recovered, Deaths considered as input for this task.Fig 15 is its visualization

Code:

```
N = 10
ind = np.arange(N)  # the x locations for the groups
width = 0.35   # the width of the bars

fig = plt.figure(figsize=(18,10))
ax = fig.add_subplot(111)
xvals=dff5.index
yvals = dff5.Confirmed
rects1 = ax.bar(ind, yvals, width, color='darkorchid',label="Confirmed")
zvals = dff5.Deaths
rects2 = ax.bar(ind+width, zvals, width, color='r',label="Deaths")
kvals = dff5.Recovered
rects3 = ax.bar(ind+width*2, kvals, width, color='g',label="Recovered")
ax.set_ylabel('No of cases',fontsize=16,fontweight='bold')
ax.set_xticks(ind+width)
ax.set_yticks(np.arange(0, 2300000, 150000))
ax.set_ylabel('Number of cases', fontsize=20)
ax.set_xlabel('Countries', fontsize=20)
ax.set_yticklabels(np.arange(0, 2300000, 150000), minor=False, fontsize=16,fontweight='bold')
ax.set_xticklabels(xvals,rotation=70,fontsize=16,fontweight='bold')
ax.legend(fontsize=20)
def autolabel(rects):
    for rect in rects:
```

```
        h = rect.get_height()
        ax.text(rect.get_x()+rect.get_width()/2., 1.00*h, '%d'%int(h),fontsize=12,fontweight='bold',
                ha='center', va='bottom')
autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
plt.title("Top ten Covid-19 cases in the world",fontsize=30,fontweight='bold')
plt.show()
```

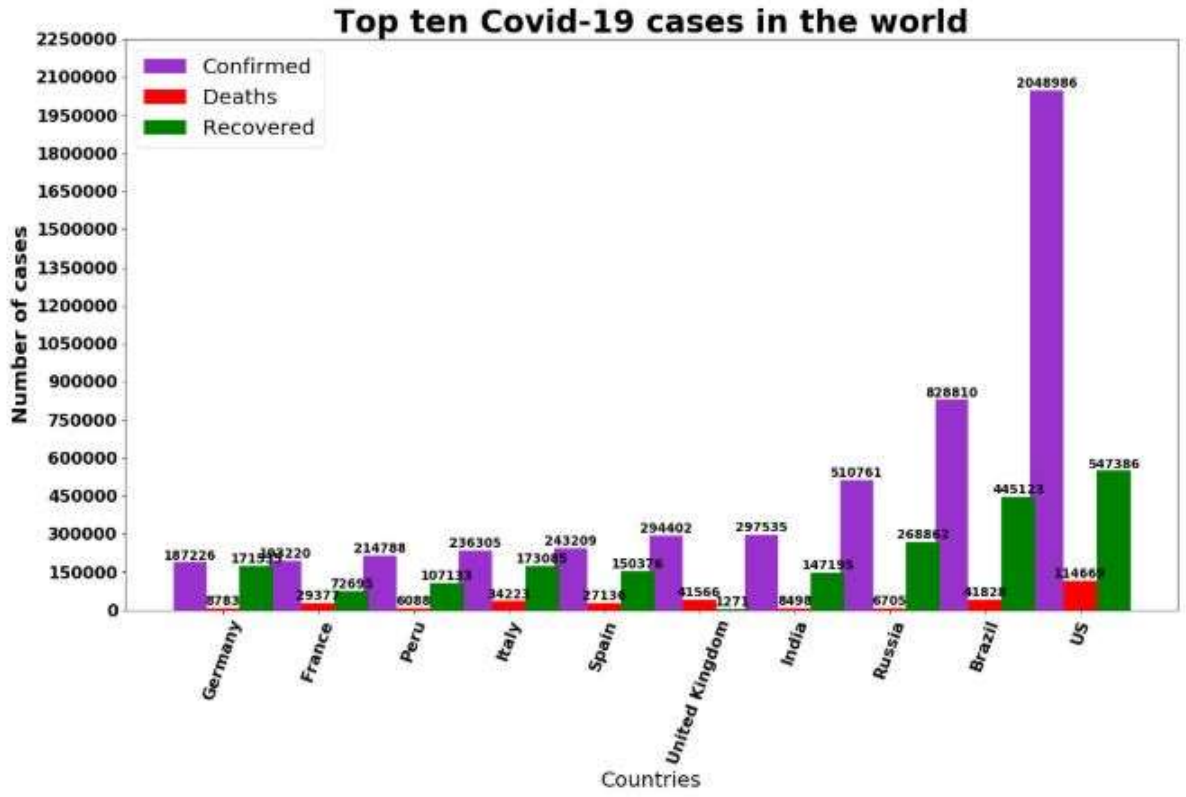

Fig 15: visualization of top 10 countries having highest covid-19 cases on 12/06/2020

Conclusion

Taking country and confirmed covid cases as an input here we can get to know that US is having more confirmed covid cases compared to all other countries. The top 10 countries having more confirmed covid cases are US, Brazil, Russia, India, United Kingdom, Spain, Italy, Peru, France and Germany till June 12th 2020. In specific date 12/06/2020 the covid confirmed cases are represented in pie plot and death cases are represented in bar plot. The covid growth in total countries is represented in line plot and here we can get to know that the confirmed cases are within the range of 0 to 8000000 and the recovered cases are within the range of 0 to 3000000 and the death cases are within the range of 0 to 400000 till June 12th 2020. The total 180 countries covid cases here the US is having 20, 48,986 covid confirmed cases and lowest is Saint Pierre and Miquelon is having 1.Based on Recovered cases the covid cases are represented here and here US is having the more Recovered cases and here we can get know that even though the confirmed cases are not that much higher but the recovered cases are more in china because it may be having best facilities for patients to recover from covid compared to other countries.

References

1. https://en.m.wikipedia.org/wiki/Portal:Coronavirus_disease_2019
2. https://www.python.org/
3. van der Walt, Stéfan & Colbert, S. & Varoquaux, Gael. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering. 13. 22 - 30. 10.1109/MCSE.2011.37
4. Barrett, Paul & Hunter, J. & Miller, J.T. & Hsu, J.-C & Greenfield, P.. (2005).matplotlib -- A Portable Python Plotting Package.
5. https://plotly.com/
6. https://packaging.python.org/tutorials/installing-packages/
7. https://statinfer.com/104-2-2-practice-working-with-datasets-in-python/
8. https://www.shanelynn.ie/python-pandas-read_csv-load-data-from-csv-files/
9. https://jakevdp.github.io/PythonDataScienceHandbook/03.03-operations-in-pandas.html
10. https://www.datacamp.com/community/tutorials/matplotlib-tutorial-python