INDIAN INSTITUTE OF INFORMATION TECHNOLOGY NAGPUR

COURSE: IT Workshop 1

TOPIC: COVID-19 ANALYZER

Raghav Agrawal	:	BT19CSE044
Aniket Shrivastav	:	BT19CSE079
Aseem Ranjan	:	BT19CSE085
Priyank Kumar Singh	:	BT19CSE090
Naveen Rathore	:	BT19CSE117

This project was performed under the guidance of Dr. Mayuri Digalwar

INDEX

Introduction:

This project aims to develop content and pictorial view of COVID-19. We intend to populate the site with practical, credible and thought-provoking information on all aspects of COVID 19. In regard to this, we have created Tkinter's GUI to demonstrate our project.

This project focuses on analysing COVID-19 situations and displays all data in a user-friendly interface by making use of graphs, charts, etc. Our application will read data from multiple datasets that contains information about new cases on a day, recovered cases on a day, deaths on a day, new cases till a day, recovered cases till a day, deaths till a day then analyse that data and on the basis of this we'll display information accordingly to the user.

In this project, we have also shown the risky and safest countries that were worth travelling in 4 months by analysing every month's data and also we have used 3 different types of matplotlib's graphs to showcase our results in the form of Line-graph, pie-charts and histogram (bar-graph) so as to gain a better understanding of our analysis.

Tools and Libraries Used-

We have used various python libraries and tools to attain an overall success in developing this project. These libraries acted as a basic building block of our project. Libraries that are user are:

1. **tkinter** - Tkinter library for developing GUI of our project

2. **pandas** - Used to manipulate numerical tables, series and dataFrames.

3. **numpy** - Used to perform various operations on numpy arrays

4. **matplotlib** - Used for plotting graphs, histogram and pie-charts.

5. **heapq** - This module is used for sorting and implementation of the Min heap.

6. **random** - Used to Generate pseudo-random numbers.

7. **math** - Used to play with decimal values.

8. **time** - Used this module for displaying current date and time.

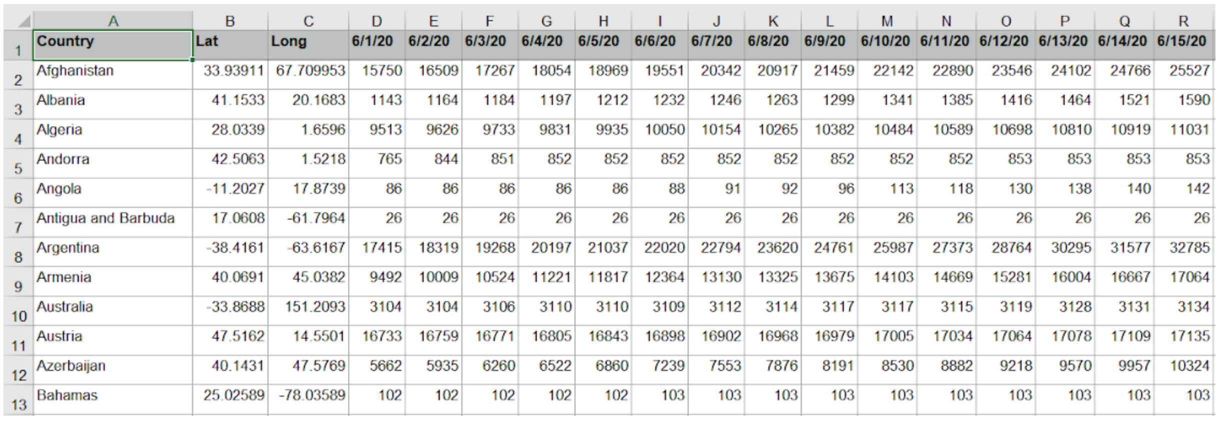
Setting Dataset for Analysis-

On searching the internet, we found 3 datasets of confirmed, recovered and death cases for the month of June, July, August and September. These 3 datasets contains cases till any date, ie. If we see the 7th July case, then that's the total case till 7th july.

We performed various operations on columns and finally ended with 3 more datasets that contain day wise cases of confirmed, recovered and death sections. So, finally our application works on multiple offline datasets that are divided in six files containing information about-

1. Confirmed new cases on a day
2. Recovered cases on a day
3. Deaths on a day
4. Confirmed new cases till a day
5. Recovered cases till a day
6. Deaths till a day

All these datasets have country names in the first column then in the next 2 columns we have geographical coordinates of respective countries and further columns are named as date to represent corresponding data in the respective dataset. Our dataset starts from 1st June 2020 and ends at 23rd September 2020, ie. We have data for 115 days.

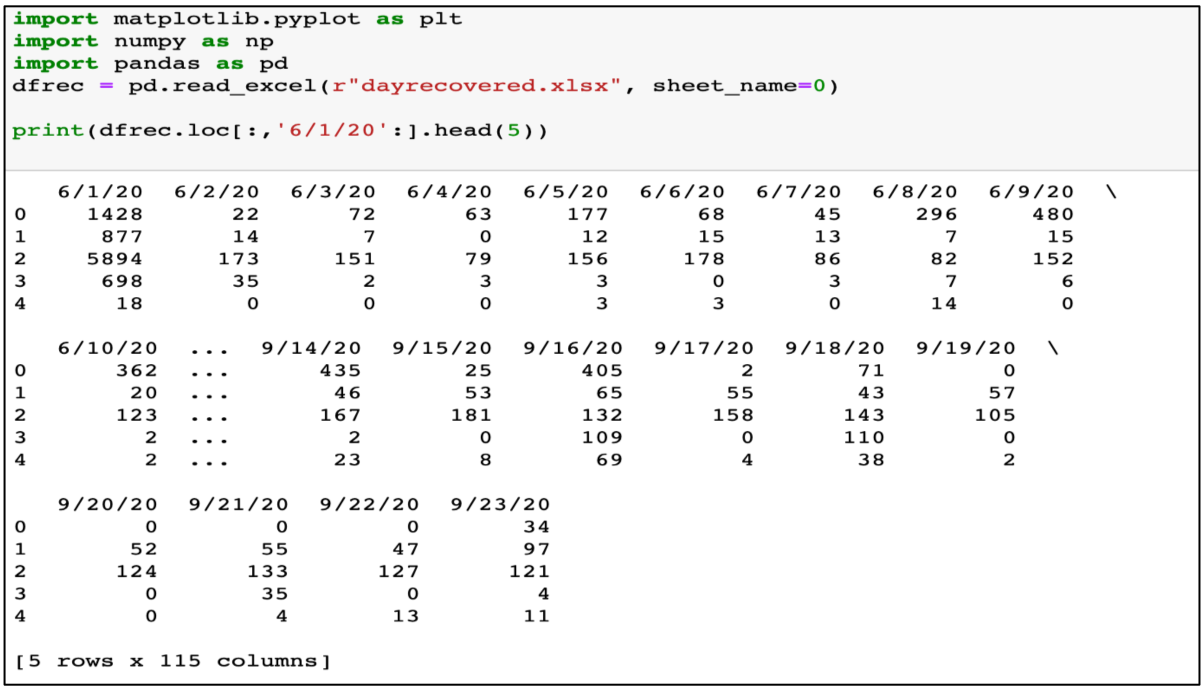


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	**Country**	**Lat**	**Long**	**6/1/20**	**6/2/20**	**6/3/20**	**6/4/20**	**6/5/20**	**6/6/20**	**6/7/20**	**6/8/20**	**6/9/20**	**6/10/20**	**6/11/20**	**6/12/20**	**6/13/20**	**6/14/20**	**6/15/20**
2	Afghanistan	33.93911	67.709953	15750	16509	17267	18054	18969	19551	20342	20917	21459	22142	22890	23546	24102	24766	25527
3	Albania	41.1533	20.1683	1143	1164	1184	1197	1212	1232	1246	1263	1299	1341	1385	1416	1464	1521	1590
4	Algeria	28.0339	1.6596	9513	9626	9733	9831	9935	10050	10154	10265	10382	10484	10589	10698	10810	10919	11031
5	Andorra	42.5063	1.5218	765	844	851	852	852	852	852	852	852	852	852	853	853	853	853
6	Angola	-11.2027	17.8739	86	86	86	86	86	88	91	92	96	113	118	130	138	140	142
7	Antigua and Barbuda	17.0608	-61.7964	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26
8	Argentina	-38.4161	-63.6167	17415	18319	19268	20197	21037	22020	22794	23620	24761	25987	27373	28764	30295	31577	32785
9	Armenia	40.0691	45.0382	9492	10009	10524	11221	11817	12364	13130	13325	13675	14103	14669	15281	16004	16667	17064
10	Australia	-33.8688	151.2093	3104	3104	3106	3110	3110	3109	3112	3114	3117	3117	3115	3119	3128	3131	3134
11	Austria	47.5162	14.5501	16733	16759	16771	16805	16843	16898	16902	16968	16979	17005	17034	17064	17078	17109	17135
12	Azerbaijan	40.1431	47.5769	5662	5935	6260	6522	6860	7239	7553	7876	8191	8530	8882	9218	9570	9957	10324
13	Bahamas	25.02589	-78.03589	102	102	102	102	102	103	103	103	103	103	103	103	103	103	103

Dataset lookout

Finally our application reads these datasets using **pandas** library's read_excel method and create dataframes for respective datasets. Now we can extract required data from these data-frames and convert them into a series or an array using **NumPy** library.

For analysis, we have used pd.matplotlib.loc [:, '6/1/20':] function for selecting column values from 6th June till 23rd August, thereby removing columns of geographical coordinates as they are of no use to us.



```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
dfrec = pd.read_excel(r"dayrecovered.xlsx", sheet_name=0)

print(dfrec.loc[:,'6/1/20':].head(5))
```

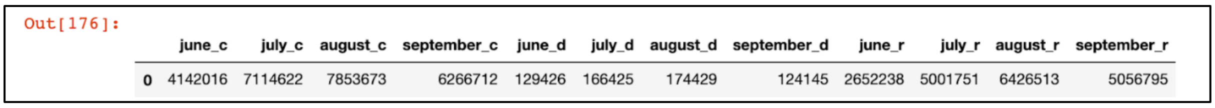
```
   6/1/20  6/2/20  6/3/20  6/4/20  6/5/20  6/6/20  6/7/20  6/8/20  6/9/20  \
0    1428      22      72      63     177      68      45     296     480   
1     877      14       7       0      12      15      13       7      15   
2    5894     173     151      79     156     178      86      82     152   
3     698      35       2       3       3       0       3       7       6   
4      18       0       0       0       3       3       0      14       0   

   6/10/20  ...  9/14/20  9/15/20  9/16/20  9/17/20  9/18/20  9/19/20  \
0      362  ...      435       25      405        2       71        0   
1       20  ...       46       53       65       55       43       57   
2      123  ...      167      181      132      158      143      105   
3        2  ...        2        0      109        0      110        0   
4        2  ...       23        8       69        4       38        2   

   9/20/20  9/21/20  9/22/20  9/23/20  
0        0        0        0       34  
1       52       55       47       97  
2      124      133      127      121  
3        0       35        0        4  
4        0        4       13       11  

[5 rows x 115 columns]
```

Modified dataset used for analysis

For plotting graphs, we have created a new dataFrame that contains total confirmed, death and recovered cases of a particular month with sum up of all the countries cases of that month, thereby giving us the total confirmed, death and recovered cases for June, July, august and September respectively. For this we have used day wise cases datasets.

dffinal = pd.DataFrame().

dffinal['july_c'] = [sum(sum(dfconfirmed.loc[i,'7/1/20':'7/31/20']) for i in range(187))]



Out[176]:

	june_c	july_c	august_c	september_c	june_d	july_d	august_d	september_d	june_r	july_r	august_r	september_r
0	4142016	7114622	7853673	6266712	129426	166425	174429	124145	2652238	5001751	6426513	5056795

dffinal DataFrame output

In the next section we'll be discussing how we have divided our analysis in three different sections? And how are these datasets being used in the analysis of COVID-19 pandemic?

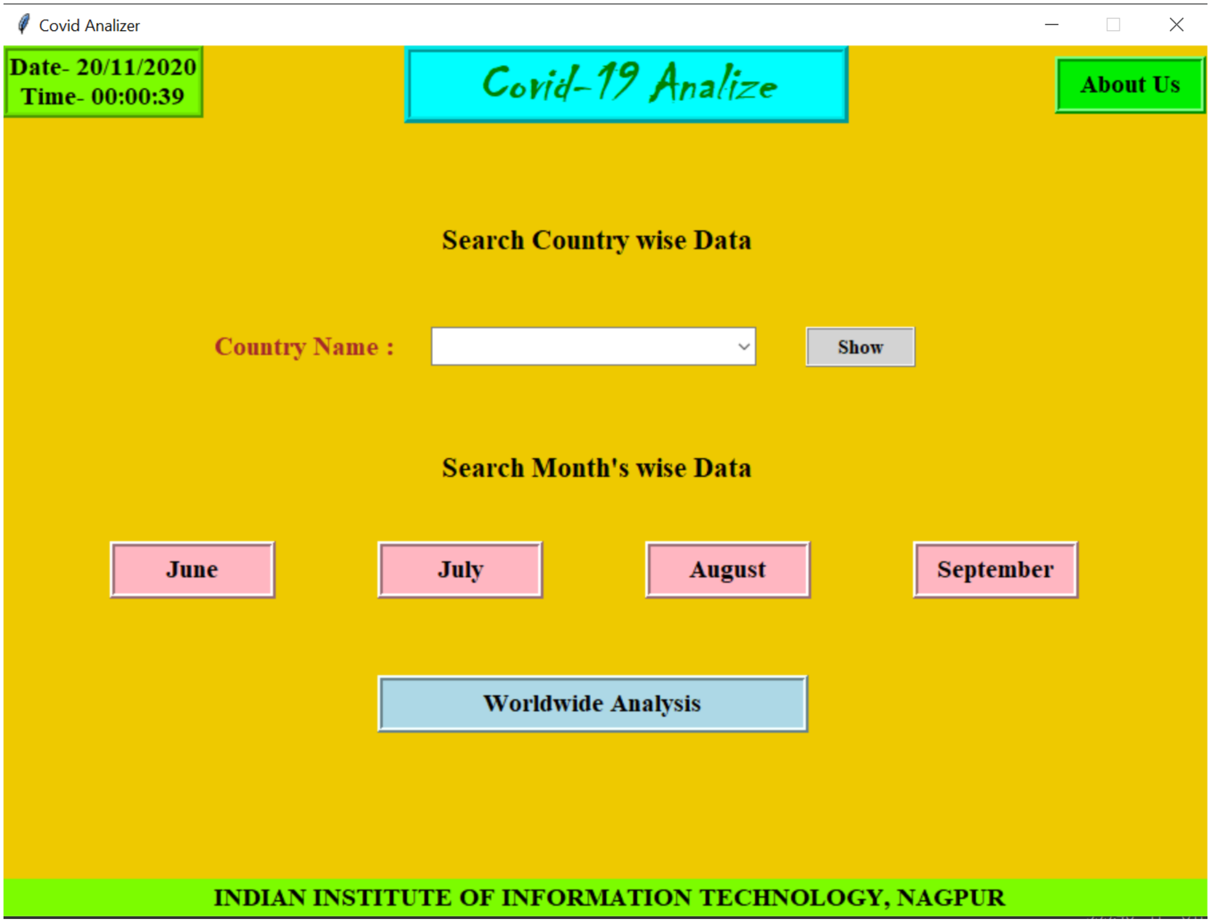
Working and Output-

Our application allows users to interact with a graphical user interface that is divided in four sections-

1. Homepage
2. Country Wise Analysis
3. Month Wise Analysis
4. World-Wide Analysis

A. Homepage-

This is the default GUI display which will be opened once we run our project's python file. Through this display we have displayed the results of all our analysis in the form of labelled texts, numbers, various graphs and charts.



GUI Display of Homepage

The Homepage has been categorised into 3 main features i.e. Country wise analysis, Month Wise Analysis and World Wide Analysis with 3 elements in header and a labelled name in footer.

In the country wise section, we have an entry field which has a drop-down menu list that helps us to find any particular country. At the adjacent of this Entry Field we give a show button which takes the entered country name and as a result we are shown up with that country page, whereas in the Month wise section, we have 4 buttons representing June, July, August and September. When any of the buttons is clicked, a function call is sent to the month's function and as a result a new window will open which will show the top 6 safest and most dangerous countries of that month with its cumulative pie-chart.

In the world-wide analyser button, we have displayed the analysis of 4 months in the form of a bar-graph with their total confirmed, recovered and death count.

Header of our GUI display has 3 elements, time module - that displays the time, Project Name Text Label and About Us button. Name label has been specially designed using Tkinter's label widget and using the python's random module, we gave out text a somewhat different style and colours.

At the left corner of the top of the Homepage, we have current date and time that are displayed using the Time Module.

Similarly, the right corner of the Homepage has About Us button and as a result the new window will pop-up which will show the name and roll numbers of the group members.

At last we have placed a footer at the bottom of our GUI display that contains a label which shows our college name with suitable text font and size.

```
footer = Label(root, text=qq, font=("times", 14, "bold"), bg="lawn green", width=82)

footer.place(x=0, y=622)
```

This placed out footer at appropriate location and by giving suitable name, font and text size.

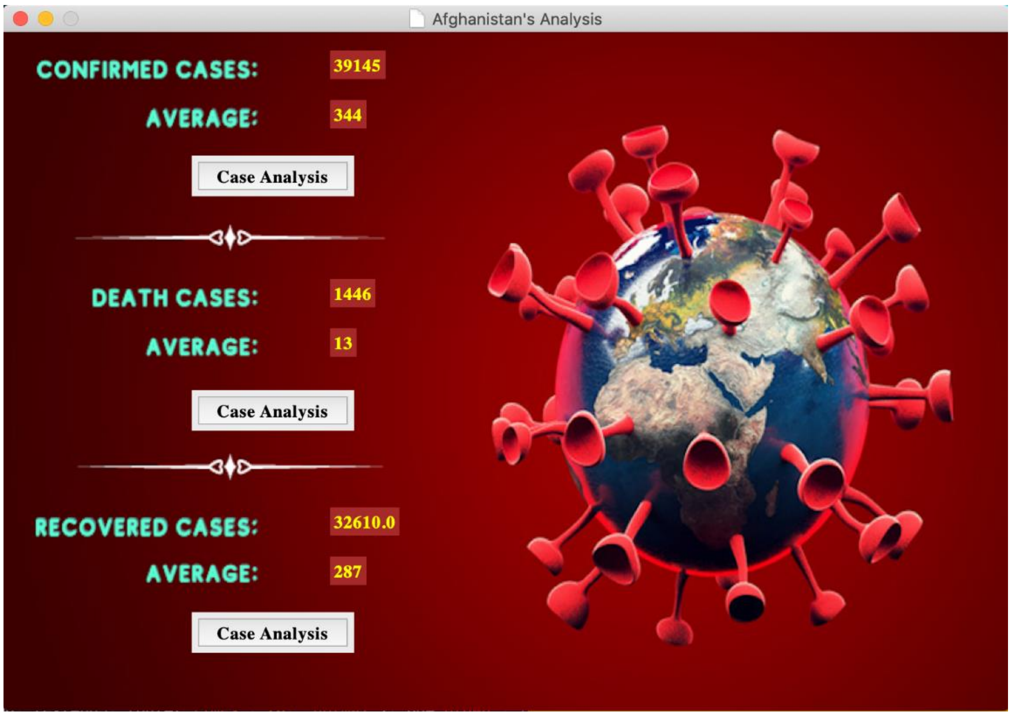
B. Country Wise Analysis-

Through this portion we have tried to display the results of 187 Countries individually by analysing their Confirmed, Recovered and Death cases and also plotted the graph between cases per day and number of days.

Every time we are selecting a country, we are shown with that country page with the results of all the analysis for that particular country. For each section of confirmed, death and recovered cases we are finding the total case of that country till 23 September because that's the last day record we have according to our dataset and also we are printing the average cases per day for each section by dividing the total cases by total number of days i.e. cases/114.

Using the Tkinter PhotoImage() function, we are reading the image and later by wrapping it in an appropriate Label() we are setting it as a background image. Also, we have added 6 Text Boxes for displaying our analysis data. The 3 Buttons that are shown in the image below are for showing the graphs for that particular section.

For finding total cases we are using pd.DataFrame.loc[r, '9/23/20'] function with r as the row index of that country and '9/23/20' as column as that's the last value we have in our dataset and it displays the total cases till that date.



CONFIRMED CASES: 39145
AVERAGE: 344

DEATH CASES: 1446
AVERAGE: 13

RECOVERED CASES: 32610.0
AVERAGE: 287

GUI of Afghanistan's Page

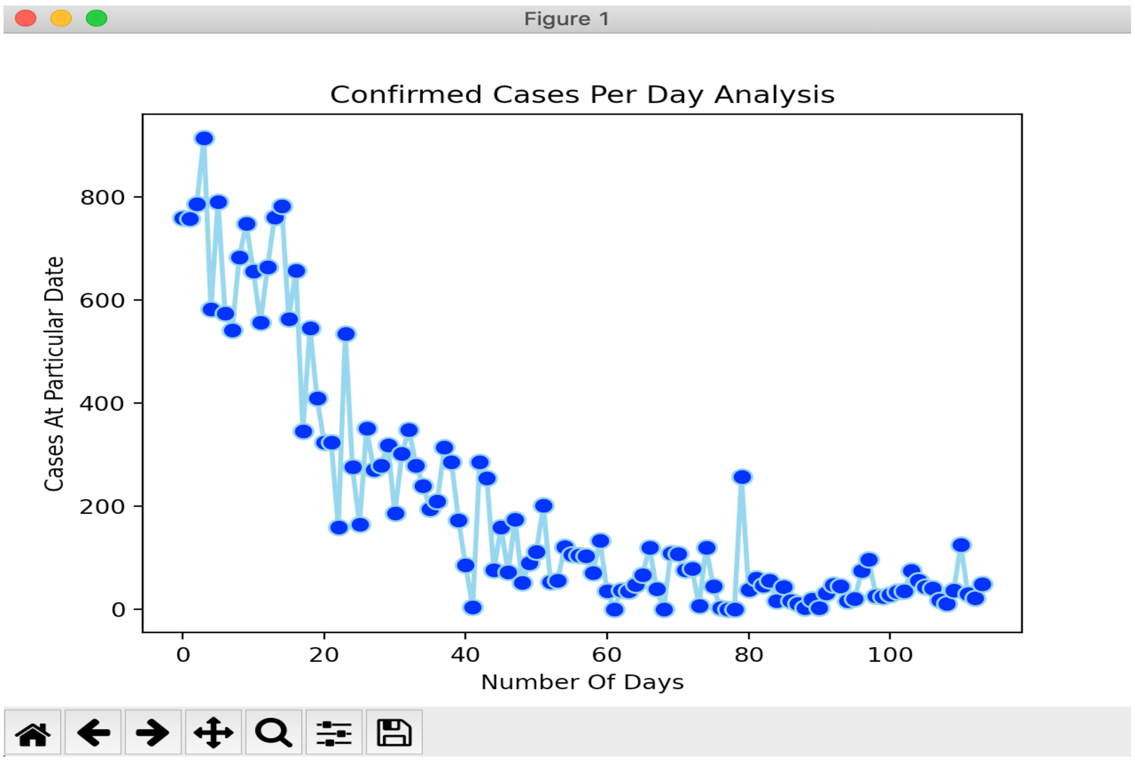
After using NumPy and pandas for our data analysis, we have used matplotlib for plotting the graphs as they helped us in understanding our data in a bit clearer full way. When we click on any of the 3 buttons, then the graph function will be called with the row index of that country as an argument to that function.

In the graph function, first we have created a new Dataframe with 2 columns as X-axand Y-ax and in X-ax column we have added values from 0-113 because that's the number of days which are going to be plotted on x-axis of our graph and in Y-ax column we have used pd.DataFrame.iloc[ind, :] function with ind as index of the selected country row and all values of thar row column.

df = pd.DataFrame({'x-ax' : range(0,114), 'y-ax' : df_name.iloc[ind, :]})

For plotting the data on graph, we have used inbuilt matplotlib.plot() function with all the necessary arguments for getting the desired results. Later we set down the names for the x and y axis and finally by using matplotlib.show() function we displayed the graph on our Tkinter GUI.

plt.plot('x-ax', 'y-ax', data=df, marker='o', markerfacecolor='blue', markersize=8, color='skyblue', linewidth=2)

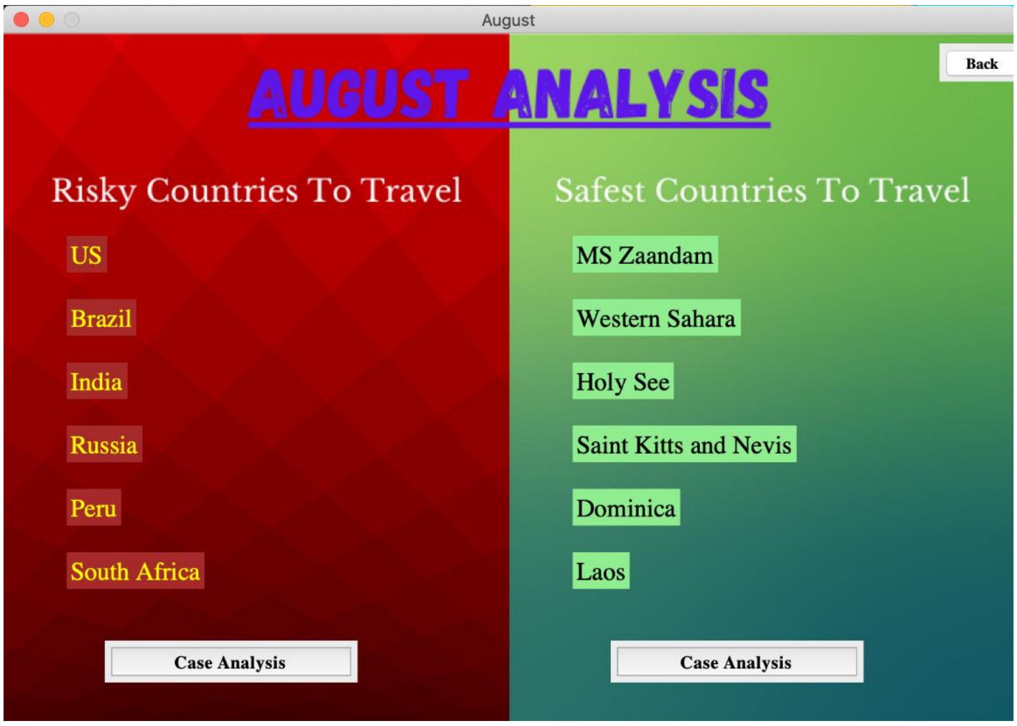


Afghanistan's Confirmed Cases Per Day Analysis

c. Month Wise Analysis-

Through this portion we displayed the Risky And Safe Countries that were worth travelling in June, July, August, September and plotted pie-chart using Matplotlib for getting the detailed information as how much safe/risky a particular country is.

Using the Tkinter PhotoImage() function, we are reading the image and later by wrapping it in an appropriate Label() we are setting it as a background image. Also, we have added 12 Text Boxes for displaying our analysis data and 3 Buttons, 2 for displaying Pie-Charts and 1 to go back to the home page.



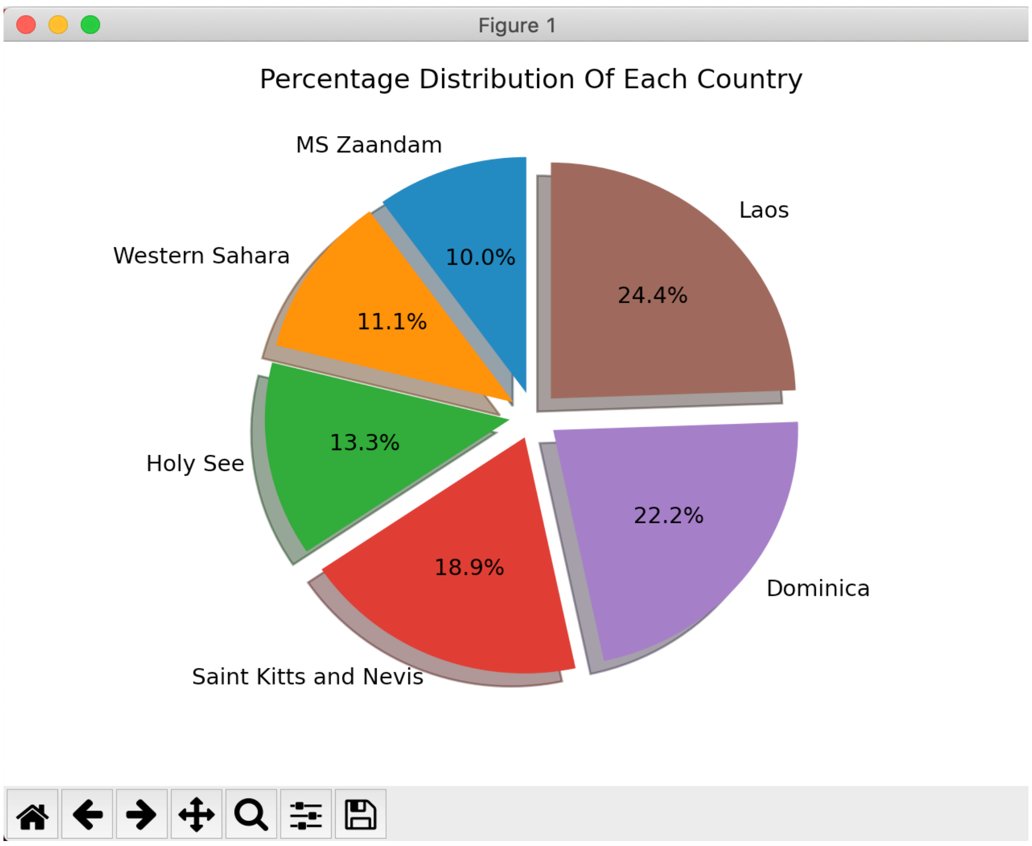
Risky Countries To Travel	Safest Countries To Travel
US	MS Zaandam
Brazil	Western Sahara
India	Holy See
Russia	Saint Kitts and Nevis
Peru	Dominica
South Africa	Laos

GUI of month analysis page

For Finding Top Risky and Top Safe countries we have used python's heapq module to find 6 Top/Bottom sorted values in our total confirmed dataset at the last date column of every month and using .loc[] we are searching for values in every row for that particular column.

heapq.nlargest(6, df_name.loc[:, date])

For detailed information, we have created a pie-chart that shows percentage wise distribution of every country.



Pie-chart Of Risky Countries in August

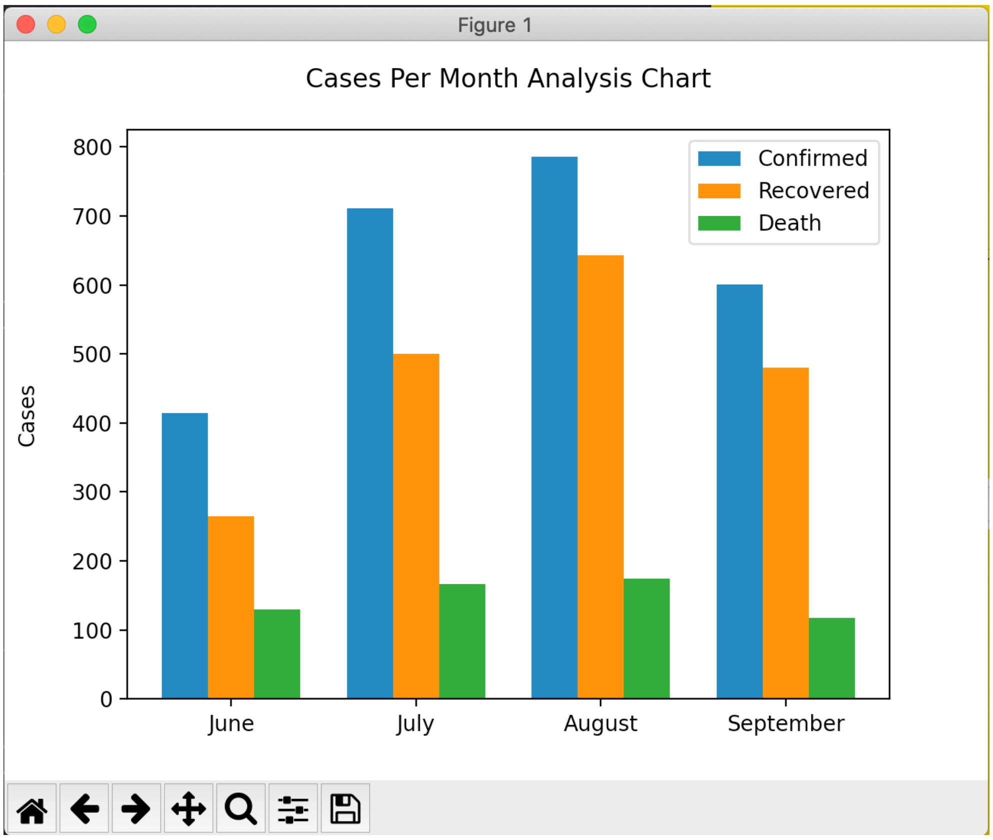
For plotting charts, we are using ax.pie() function where ax is defined as subplots of matplotlib library. To this ax.pie() function we are passing country name, values (total confirmed cases), autopct (to convert values to percentage format) and a few more parameters. Finally by using matplotlib.show() we are displaying our pie-chart.

ax.pie(values, labels=name, autopct='%1.1f%%', shadow=True, startangle=90)

This same procedure has been applied twice once for getting risky countries and another for safe countries and followed by a Back button which once clicked will destroy this window and will take us back to the main Home Page.

D. World-Wide Analysis-

Through this portion we have displayed the overall analysis of all the 4 months, i.e. June, July, August, September with the help of matplotlib's bar-graph/histogram.



Histogram Representing World-Wide Analysis of 4 Months

For plotting the histogram, we have created a new DataFrame that comprises 12 Columns, 3 for each month with total confirmed, recovered and death cases count of that particular month.

df_histo['june_c'] = [sum(sum(df_dc.loc[i, '6/2/20':'6/30/20']) for i in range(186))]

For creating the histogram, we are making 3 tuples wrapped in a list and each tuple containing 4 values i.e. Confirmed cases of all the 4 months, Recovered cases of all the 4 months, Death cases of all the 4 months. Then by using matplotlib.bar() function we are plotting these values (building shaped rectangles) and by using matplotlib.xticks() function we are labelling each section of graph as June, July, August and September.

Summary-

Covid analyser focuses on analysing COVID-19 situations and displays all data in a user-friendly interface by making use of graphs, charts, etc. We have used various libraries and modules like Tkinter,

Pandas, NumPy, Matplotlib, Heapq, Math etc to manipulate numerical tables and time series, for plotting graphs and histogram for covid-19 analysis etc. In GUI of our application we have created a Homepage,

Country Wise, Month wise, worldwide analyser. At the end we conclude that our application analyses all COVID-19 situations in a user-friendly way so that people can get proper data about their localized area so that they will be aware by seating at their homes.

Reference-

This project helped us in learning a lot about cleaning data, analysing data, and it has showed us the power of python programming language. We also took some material and reference from different websites, books, and other.

1. Kaggle
2. Pandas.pydata.org
3. Stack Overflow
4. GeeksforGeeks
5. W3Schools
6. Tutorialspoint
7. Medium