

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award
of Bachelor of Technology Degree in

Information Technology

By

YUVASRI J

(Reg. No.38120097)



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

JANUARY 2021



**SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY**

(Established under Section 3 of UGC Act, 1956)

**Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai
600119**



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **YUVASRI J(Reg No. 38120097)** who carried out the project entitled “**COVID-19 DATA ANALYSIS AND VISUALIZATION**” under our supervision from Jan 2019 to April 2019.

Internal Guide

Dr. V. NIRMALRANI.,M.E.,Ph.D.,

Head of the Department

Dr. R. SUBHASHINI M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **YUVASRI J (Reg No:38120097)** hereby declare that the Project Report entitled “**COVID-19 DATA ANALYSIS AND VISUALIZATION**” done by me under the guidance of **Dr. V.**

NIRMALRANI, M.E., Ph.D.,

is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology. _____

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA M.E., Ph.D.**, Dean, School of Computing and **Dr. R. SUBHASHINI M.E.**, Head of the Department, Department of Computer Science and Engineering for providing us the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. V. NIRMALRANI** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **INFORMATION TECHNOLOGY** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



RETECH
SOLUTIONS PVT. LTD.
An ISO 9001:2015 certified company

INT: RE20S042

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **YUVASRI J** student of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, BTECH/IT** has undergone the Internship Training through online in our concern entitled **PYTHON** from **12th Aug 2020** to **26th Aug 2020** in relevant departments related to their academic studies.

During the above period, the performance was good & we wish great success in all your future endeavours.

Technology drives the world. We are the people who are behind the technology.

Authorised Signature



No-31, 1st Floor, Alagesan Street, Tambaram West, Chennai-45.

Office : 044-42077204 | +91-8122266653

www.retechsolutions.in



October 1, 2020

YUVASRI.J

has successfully completed

**Programming for Everybody (Getting
Started with Python)**

an online non-credit course authorized by University of Michigan and offered
through Coursera

A stylized, handwritten signature in black ink, appearing to read 'Signature'.

PROFESSOR JOHN DOE

COURSE
CERTIFICATE



ABSTRACT

This project is about analyse the Coronavirus disease 2019 (Covid -19) data's and visualization .This runs in python .The map's and plot's can be created by using the folium and plotly modules. The dataset includes time series data tracking the number of people affected by Coronavirus disease 2019 (COVID-19) worldwide, including: confirmed tested cases of Coronavirus infection , the number of people who have reportedly died while sick with Coronavirus and the number of people who have reportedly recovered from it. we can see daywise, countrywise and country_daywise reports. The overall modules used in this project are FOLIUM and PLOTLY.

TABLE OF CONTENTS

CHAPTER NO.	TITLE NAME	PG.NO
	ABSTRACT	i
1.	INTRODUCTION	1
	1.1 GENERAL	1
	1.2 ORGANIZATION OF THE PROJECT	1
2.	AIM & SCOPE	2
	2.1 AIM	2
	2.2 SCOPE	2
3.	ALGORITHMS AND METHODS	3
4.	SYSTEM IMPLEMENTATION	4
5.	RESULTS	5
6.	CONCLUSION AND FUTURE WORK	14
	6.1 CONCLUSION	14
	6.2 FUTURE WORK	14
	REFERENCES	15
	APPENDIX	16

CHAPTER 1

INTRODUCTION

1.1 GENERAL

With the rapid spread in the novel corona-virus across countries, the World Health Organisation (WHO) and several countries have published latest results on the impact of COVID-19 over the past few months. The entire project is easily understood by a normal person who knows simple english and python. The language used in this project is python.

1.2 ORGANIZATION OF THE REPORT

Chapter 2 deals with Aim, Scope of the present investigation. Chapter 3 details with General, Algorithm, Project description. Chapter 4 deals with Requirement, Role of Python and Results And Discussion. Chapter 5 deals with Conclusion and Future work.

CHAPTER 2

AIM & SCOPE

2.1 AIM

The main objective of this project is analyse the COVID-19 data's and how visualization helps to derive informative insights from data sources..

2.2 SCOPE

Python is one of the most prevalent coding languages. Alongside with being a high-level and general-purpose programming language, Python is also an object oriented and open source. Thus, analyse the data and visualization with python is easily implemented here.

CHAPTER 3

ALGORITHMS AND METHODS

1. Install the modules FOLIUM and PLOTLY which are needed to generate map and plot.
2. Import the libraries such as numpy,pandas,matplotlib ,etc...
3. Connect with dataset.
4. Read daywise, countrywise and country_daywise reports.
5. Print Confirmed, Recovered and Deaths cases.
6. Generate the FOLIUM and get the result in maps format.
7. Generate the PLOTLY and get the result in plots format.
8. Using PLOTS and FOLIUM display Death Cases, Recoverd Cases and Confirmed Cases in the format of plots and maps.

³ **CHAPTER 4**

SYSTEM REQUIREMENTS

REQUIREMENTS

1. HARDWARE REQUIREMENTS

- * A system with minimum 2GB of RAM.
- * Keyboard.
- * Monitor.

2. SOFTWARE REQUIREMENTS

- * Python 3.7 version in the system.
- * Folium & Plotly module using 'pip' command.
- * Jupyter Notebook.

4 CHAPTER 5 RESULTS

Worldwide Total Confirmed, recovered, and Deaths

```
In [13]: confirmed.tail()
```

```
Out[13]:
```

	Date	Confirmed
269	2020-10-17	39670828
270	2020-10-18	39955637
271	2020-10-19	40395527
272	2020-10-20	40783425
273	2020-10-21	41227176

```
In [14]: deaths.tail()
```

```
Out[14]:
```

	Date	Deaths
269	2020-10-17	1109833
270	2020-10-18	1113178
271	2020-10-19	1118159
272	2020-10-20	1124745
273	2020-10-21	1131308

Fig 5.1: WORLDWIDE TOTAL CONFIRMED,DEATHS CASES

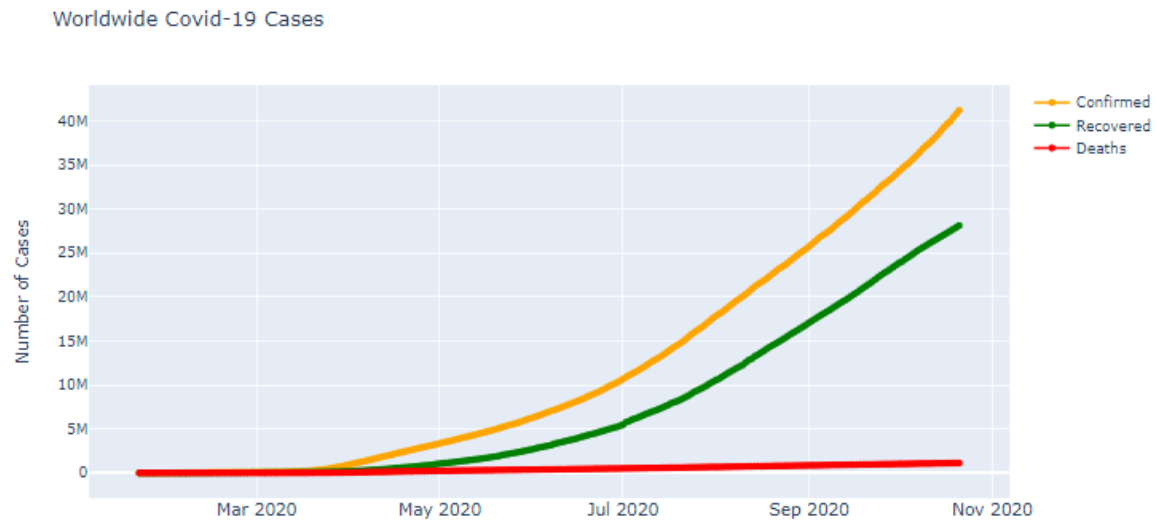


Fig 5.2: WORLDWIDE COVID-19 CASES

Worldwide Covid-19 Cases with Time Laps

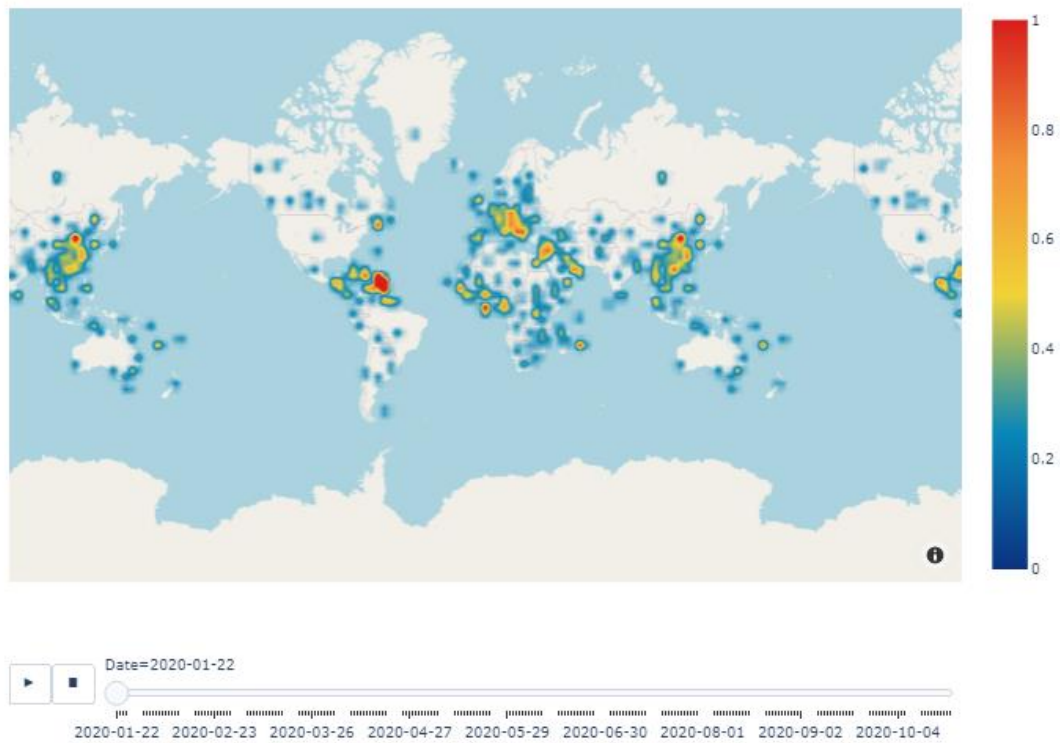


Fig 5.3: WORLDWIDE COVID-19 CASES WITH TIME LAPS

	Date	Province/State	Country	Lat	Long	Confirmed	Recovered	Deaths	Active
11507	2020-10-21 00:00:00	Diamond Princess	Canada	0.000000	0.000000	0	0	1	-1
11781	2020-10-21 00:00:00	Grand Princess	Canada	0.000000	0.000000	13	0	0	13
28221	2020-10-21 00:00:00	Diamond Princess		0.000000	0.000000	712	659	13	40
46305	2020-10-21 00:00:00	MS Zaandam		0.000000	0.000000	9	0	2	7

Fig 5.4: TOTAL CASES ON SHIPS



Fig 5.5: CASES OVER THE TIME WITH AREA PLOT

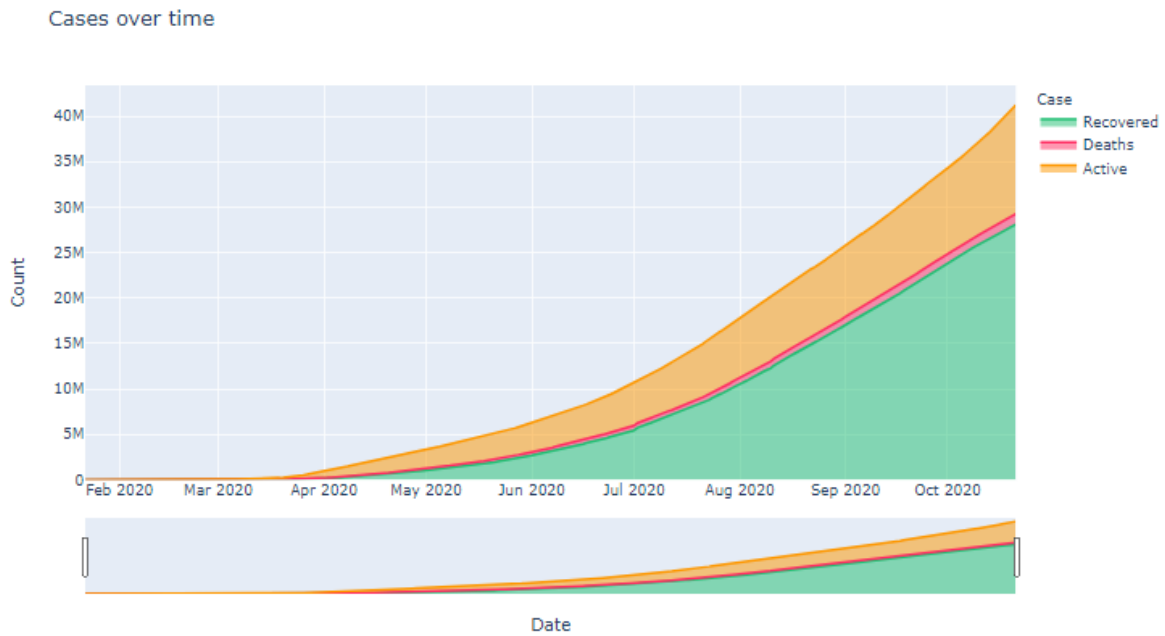


Fig 5.6: CASES OVER TIME WITH AREA PLOT

Cases over time

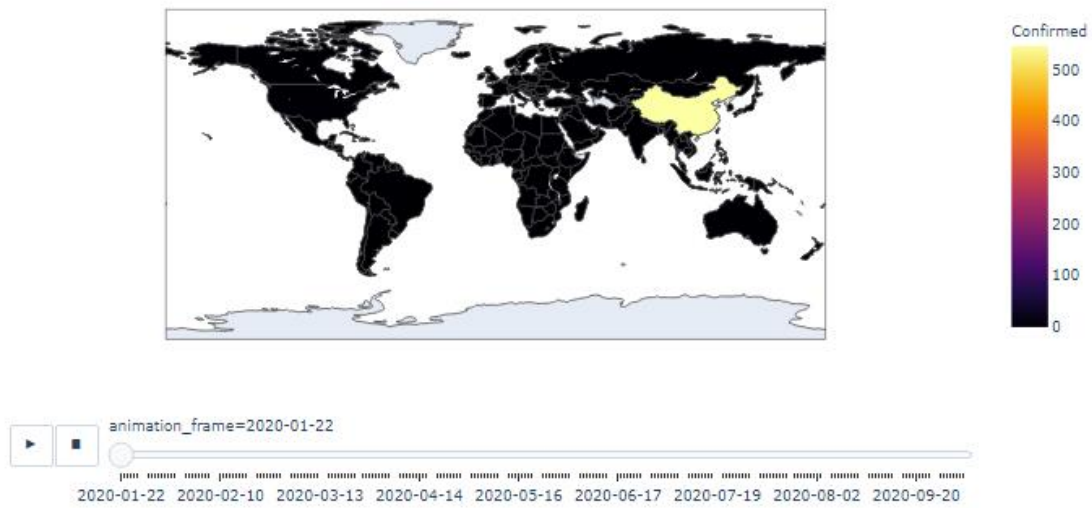


Fig 5.7: CONFIRMED CASES WITH CHOROPLETH MAP

	Date	Confirmed	Deaths	Recovered	Active	New Cases	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	No. of Countries
0	2020-01-22	555	17	28	510	0	3.06	5.05	60.71	6
1	2020-01-23	654	18	30	606	99	2.75	4.59	60.00	8
2	2020-01-24	941	26	36	879	287	2.76	3.83	72.22	9
3	2020-01-25	1434	42	39	1353	493	2.93	2.72	107.69	11
4	2020-01-26	2118	56	52	2010	684	2.64	2.46	107.69	13

Fig 5.8: DEATHS AND RECOVERIES PER 100 CASES

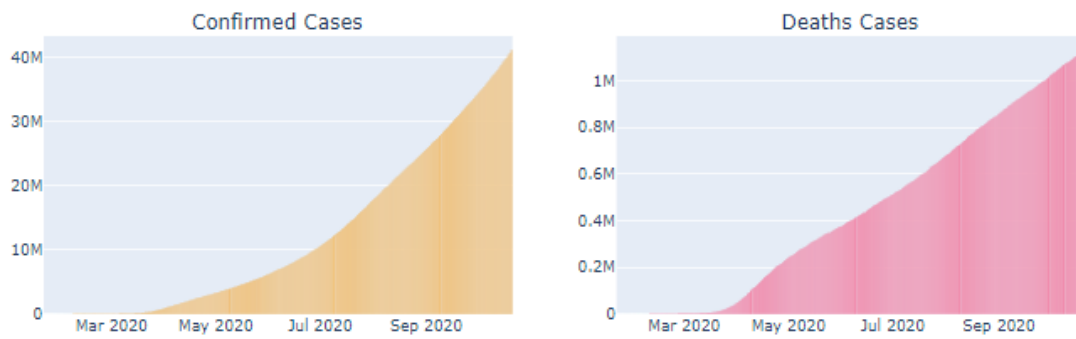


Fig 5.9: CONFIRMED AND DEATHS PER 100 CASES

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

This project was successfully completed, within the time span allotted. It is very simple. It is easily understood to the users who have knowledge in English and Python. To conclude this, we thank all the people who help us to complete this project.

6.2 FUTURE WORK

This project describes analysing the COVID-19 data of the algorithm. In the future we can determine the efficient algorithm based on their accuracy metrics that will help to choose an efficient algorithm for analysing the data.

REFERENCES

[1] PYTHON : <https://www.w3schools.com/python/>

[2] COVID 19 Data analysis : Coursera

[3] <https://jupyter.org/try>

APPENDICES

SOURCE CODE

```
pip install folium
```

```
pip install plotly
```

```
# imports
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
```

```
import folium
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
%matplotlib inline
```

```
import math
import random
from datetime import timedelta
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
#color palette
cnf = '#393e46'
dth = '#ff2e63'
rec = '#21bf73'
act = '#fe9801'
```

```
#Dataset Preparation
```

```
import plotly as py
py.offline.init_notebook_mode(connected = True)
```

```
try:
    os.system("rm -rf Covid-19-Preprocessed-Dataset")
except:
    print('File does not exist')
```

```
!git clone https://github.com/laxmimerit/Covid-19-Preprocessed-Dataset.git
```

```
df = pd.read_csv('Covid-19-Preprocessed-
Dataset/preprocessed/covid_19_data_cleaned.csv',parse_dates=['Date'])
```

```
country_daywise = pd.read_csv('Covid-19-Preprocessed-
Dataset/preprocessed/country_daywise.csv',parse_dates=['Date'])
countrywise = pd.read_csv('Covid-19-Preprocessed-Dataset/preprocessed/countrywise.csv')
daywise=pd.read_csv('Covid-19-Preprocessed-Dataset/preprocessed/daywise.csv',parse_dates=
['Date'])
```

```

df['Province/State'] = df['Province/State'].fillna("")
df.head()

confirmed = df.groupby('Date').sum()['Confirmed'].reset_index()
recovered = df.groupby('Date').sum()['Recovered'].reset_index()
deaths = df.groupby('Date').sum()['Deaths'].reset_index()
deaths.head()

df.isnull().sum()

df.info()

df.query('Country == "US"')

confirmed.tail()

deaths.tail()

fig = go.Figure()
fig.add_trace(go.Scatter(x = confirmed['Date'], y = confirmed['Confirmed'], mode = 'lines+markers',
name = 'Confirmed', line = dict(color = "Orange", width = 2)))
fig.add_trace(go.Scatter(x = recovered['Date'], y = recovered['Recovered'], mode =
'lines+markers', name = 'Recovered', line = dict(color = "Green", width = 2)))
fig.add_trace(go.Scatter(x = deaths['Date'], y = deaths['Deaths'], mode = 'lines+markers', name =
'Deaths', line = dict(color = "Red", width = 2)))
fig.update_layout(title = 'Worldwide Covid-19 Cases', xaxis_tickfont_size = 14, yaxis = dict(title =
'Number of Cases'))
fig.show()

df.info()

df['Date'] = df['Date'].astype(str)

df.info()

df.head()

fig = px.density_mapbox(df, lat = 'Lat', lon = 'Long', hover_name = 'Country', hover_data =
['Confirmed', 'Recovered', 'Deaths'], animation_frame='Date', color_continuous_scale='Portland',
radius = 7, zoom = 0, height = 700)
fig.update_layout(title = 'Worldwide Covid-19 Cases with Time Laps')
fig.update_layout(mapbox_style = 'open-street-map', mapbox_center_lon = 0)
fig.show()

df['Date'] = pd.to_datetime(df['Date'])
df.info()

#ships
ship_rows = df['Province/State'].str.contains('Grand Princess') |
df['Province/State'].str.contains('Diamond Princess') | df['Country'].str.contains('Grand Princess') |
df['Country'].str.contains('Diamond Princess') | df['Country'].str.contains('MS Zaandam')
ship = df[ship_rows]
df = df[~ship_rows]

ship_latest = ship[ship['Date'] == max(ship['Date'])]
ship_latest
ship_latest.style.background_gradient(cmap = 'Pastel1_r')

```

```
temp = df.groupby('Date')['Confirmed', 'Deaths', 'Recovered', 'Active'].sum().reset_index()
temp = temp[temp['Date']==max(temp['Date'])].reset_index(drop = True)
```

```
tm = temp.melt(id_vars = 'Date', value_vars = ['Active', 'Deaths', 'Recovered'])
fig = px.treemap(tm, path = ['variable'], values = 'value', height = 250, width = 800,
color_discrete_sequence=[act, rec, dth])
```

```
fig.data[0].textinfo = 'label+text+value'
fig.show()
```

```
temp = df.groupby('Date')['Recovered', 'Deaths', 'Active'].sum().reset_index()
temp = temp.melt(id_vars = 'Date', value_vars = ['Recovered', 'Deaths', 'Active'], var_name =
'Case', value_name = 'Count')
```

```
fig = px.area(temp, x = 'Date', y = 'Count', color = 'Case', height = 600, title = 'Cases over time',
color_discrete_sequence=[rec, dth, act])
fig.update_layout(xaxis_rangeslider_visible=True)
fig.show()
```

```
country_daywise.head()
```

```
fig = px.choropleth(country_daywise, locations = 'Country', locationmode='country names', color =
country_daywise['Confirmed'],
    hover_name = 'Country', animation_frame=country_daywise['Date'].dt.strftime('%Y-
%m-%d'),
    title='Cases over time', color_continuous_scale=px.colors.sequential.Inferno)
fig.update(layout_coloraxis_showscale = True)
fig.show()
daywise.head()
```

```
fig_c = px.bar(daywise, x = 'Date', y = 'Confirmed', color_discrete_sequence=[act])
fig_d = px.bar(daywise, x = 'Date', y = 'Deaths', color_discrete_sequence=[dth])
```

```
fig = make_subplots(rows = 1, cols = 2, shared_xaxes=False, horizontal_spacing=0.1,
    subplot_titles=('Confirmed Cases', 'Deaths Cases'))
```

```
fig.add_trace(fig_c['data'][0], row = 1, col = 1)
fig.add_trace(fig_d['data'][0], row = 1, col = 2)
```

```
fig.update_layout(height = 400)
```

```
fig.show()
```

```
fig1 = px.line(daywise, x = 'Date', y = 'Deaths / 100 Cases', color_discrete_sequence=[dth])
fig2 = px.line(daywise, x = 'Date', y = 'Recovered / 100 Cases', color_discrete_sequence=[rec])
fig3 = px.line(daywise, x = 'Date', y = 'Deaths / 100 Recovered', color_discrete_sequence=[dth])
```

```
fig = make_subplots(rows = 1, cols = 3, shared_xaxes=False,
    subplot_titles=('Deaths / 100 Cases', 'Recovered / 100 Cases', 'Deaths / Recovered'))
```

```
fig.add_trace(fig1['data'][0], row = 1, col = 1)
fig.add_trace(fig2['data'][0], row = 1, col = 2)
fig.add_trace(fig3['data'][0], row = 1, col = 3)
```

```
fig.update_layout(height = 400)
fig.show()
```

```

fig_c = px.bar(daywise, x = 'Date', y = 'Confirmed', color_discrete_sequence=[act])
fig_d = px.bar(daywise, x = 'Date', y = 'No. of Countries', color_discrete_sequence=[dth])

fig = make_subplots(rows = 1, cols = 2, shared_xaxes=False, horizontal_spacing=0.1,
                    subplot_titles=('No. of New Cases per Day', 'No. of Countries'))

fig.add_trace(fig_c['data'][0], row = 1, col = 1)
fig.add_trace(fig_d['data'][0], row = 1, col = 2)

fig.show()

top = 15

fig_c = px.bar(countrywise.sort_values('Confirmed').tail(top), x = 'Confirmed', y = 'Country',
               text = 'Confirmed', orientation='h', color_discrete_sequence=[cnf])
fig_d = px.bar(countrywise.sort_values('Deaths').tail(top), x = 'Deaths', y = 'Country',
               text = 'Deaths', orientation='h', color_discrete_sequence=[dth])

fig_a = px.bar(countrywise.sort_values('Active').tail(top), x = 'Active', y = 'Country',
               text = 'Active', orientation='h', color_discrete_sequence=[act])
fig_r = px.bar(countrywise.sort_values('Recovered').tail(top), x = 'Recovered', y = 'Country',
               text = 'Recovered', orientation='h', color_discrete_sequence=[rec])

fig_dc = px.bar(countrywise.sort_values('Deaths / 100 Cases').tail(top), x = 'Deaths / 100 Cases', y = 'Country',
               text = 'Deaths / 100 Cases', orientation='h', color_discrete_sequence=[#f84351])
fig_rc = px.bar(countrywise.sort_values('Recovered / 100 Cases').tail(top), x = 'Recovered / 100 Cases', y = 'Country',
               text = 'Recovered / 100 Cases', orientation='h', color_discrete_sequence=[#a45398])

fig_nc = px.bar(countrywise.sort_values('New Cases').tail(top), x = 'New Cases', y = 'Country',
               text = 'New Cases', orientation='h', color_discrete_sequence=[#f04341])

temp = countrywise[countrywise['Population']>1000000]
fig_p = px.bar(temp.sort_values('Cases / Million People').tail(top), x = 'Cases / Million People', y = 'Country',
               text = 'Cases / Million People', orientation='h', color_discrete_sequence=[#b40398])

fig_wc = px.bar(countrywise.sort_values('1 week change').tail(top), x = '1 week change', y = 'Country',
               text = '1 week change', orientation='h', color_discrete_sequence=[#c04041])

temp = countrywise[countrywise['Confirmed']>100]
fig_wi = px.bar(temp.sort_values('1 week % increase').tail(top), x = '1 week % increase', y = 'Country',
               text = '1 week % increase', orientation='h', color_discrete_sequence=[#b00398])

fig = make_subplots(rows = 5, cols = 2, shared_xaxes=False, horizontal_spacing=0.2,
                    vertical_spacing=0.05,
                    subplot_titles=('Confirmed Cases', 'Deaths Reported', 'Recovered Cases', 'Active

```

```

Cases',
        'Deaths / 100 Cases', 'Recovered / 100 Cases',
        'New Cases', 'Cases / Million People',
        '1 week change', '1 week % increase'))

fig.add_trace(fig_c['data'][0], row = 1, col = 1)
fig.add_trace(fig_d['data'][0], row = 1, col = 2)

fig.add_trace(fig_r['data'][0], row = 2, col = 1)
fig.add_trace(fig_a['data'][0], row = 2, col = 2)

fig.add_trace(fig_dc['data'][0], row = 3, col = 1)
fig.add_trace(fig_rc['data'][0], row = 3, col = 2)

fig.add_trace(fig_nc['data'][0], row = 4, col = 1)
fig.add_trace(fig_p['data'][0], row = 4, col = 2)

fig.add_trace(fig_wc['data'][0], row = 5, col = 1)
fig.add_trace(fig_wi['data'][0], row = 5, col = 2)

fig.update_layout(height = 2000)
fig.show()

top = 15
fig = px.scatter(countrywise.sort_values('Deaths', ascending = False).head(top),
                x = 'Confirmed', y = 'Deaths', color = 'Country', size = 'Confirmed', height = 600,
                text = 'Country', log_x = True, log_y = True, title='Deaths vs Confirmed Cases (Cases
are on log10 scale)')

fig.update_traces(textposition = 'top center')
fig.update_layout(showlegend = True)
fig.update_layout(xaxis_rangeslider_visible = True)
fig.show()

countrywise.sort_values('Deaths', ascending = False).head(15)

fig = px.bar(country_daywise, x = 'Date', y = 'Confirmed', color = 'Country', height = 600,
             title='Confirmed', color_discrete_sequence=px.colors.cyclical.mygbm)
fig.show()
fig = px.bar(country_daywise, x = 'Date', y = 'Deaths', color = 'Country', height = 600,
             title='Deaths', color_discrete_sequence=px.colors.cyclical.mygbm)
fig.show()

fig = px.bar(country_daywise, x = 'Date', y = 'Recovered', color = 'Country', height = 600,
             title='Recovered', color_discrete_sequence=px.colors.cyclical.mygbm)
fig.show()

fig = px.bar(country_daywise, x = 'Date', y = 'New Cases', color = 'Country', height = 600,
             title='New Cases', color_discrete_sequence=px.colors.cyclical.mygbm)
fig.show()

fig = px.line(country_daywise, x = 'Date', y = 'Confirmed', color = 'Country', height = 600,
             title='Confirmed', color_discrete_sequence = px.colors.cyclical.mygbm)

```

```
fig.show()
```

```
fig = px.line(country_daywise, x = 'Date', y = 'Deaths', color = 'Country', height = 600,  
              title='Deaths', color_discrete_sequence = px.colors.cyclical.mygbm)  
fig.show()
```

```
fig = px.line(country_daywise, x = 'Date', y = 'Recovered', color = 'Country', height = 600,  
              title='Recovered', color_discrete_sequence = px.colors.cyclical.mygbm)  
fig.show()
```

```
gt_100 = country_daywise[country_daywise['Confirmed']>100]['Country'].unique()  
temp = df[df['Country'].isin(gt_100)]
```

```
temp = temp.groupby(['Country', 'Date'])['Confirmed'].sum().reset_index()  
temp = temp[temp['Confirmed']>100]
```

```
min_date = temp.groupby('Country')['Date'].min().reset_index()  
min_date.columns = ['Country', 'Min Date']
```

```
from_100th_case = pd.merge(temp, min_date, on = 'Country')  
from_100th_case['N days'] = (from_100th_case['Date'] - from_100th_case['Min Date']).dt.days
```

```
fig = px.line(from_100th_case, x = 'N days', y = 'Confirmed', color = 'Country', title = 'N days from  
100 case', height = 600)  
fig.show()
```

```
gt_1000 = country_daywise[country_daywise['Confirmed']>1000]['Country'].unique()  
temp = df[df['Country'].isin(gt_1000)]
```

```
temp = temp.groupby(['Country', 'Date'])['Confirmed'].sum().reset_index()  
temp = temp[temp['Confirmed']>1000]
```

```
min_date = temp.groupby('Country')['Date'].min().reset_index()  
min_date.columns = ['Country', 'Min Date']
```

```
from_1000th_case = pd.merge(temp, min_date, on = 'Country')  
from_1000th_case['N days'] = (from_1000th_case['Date'] - from_1000th_case['Min  
Date']).dt.days
```

```
fig = px.line(from_1000th_case, x = 'N days', y = 'Confirmed', color = 'Country', title = 'N days  
from 1000 case', height = 600)  
fig.show()
```

```
first_date = df[df['Confirmed']>0]
```

```
first_date = first_date.groupby('Country')['Date'].agg(['min']).reset_index()
```

```
last_date = df.groupby(['Country', 'Date'])['Confirmed', 'Deaths', 'Recovered']
```

```

last_date = last_date.sum().diff().reset_index()

mask = last_date['Country'] != last_date['Country'].shift(1)

last_date.loc[mask, 'Confirmed'] = np.nan
last_date.loc[mask, 'Deaths'] = np.nan
last_date.loc[mask, 'Recovered'] = np.nan

last_date = last_date[last_date['Confirmed'] > 0]
last_date = last_date.groupby('Country')['Date'].agg(['max']).reset_index()

first_last = pd.concat([first_date, last_date['max']], axis = 1)
first_last['max'] = first_last['max'] + timedelta(days = 1)

first_last['Days'] = first_last['max'] - first_last['min']
first_last['Task'] = first_last['Country']

first_last.columns = ['Country', 'Start', 'Finish', 'Days', 'Task']

first_last = first_last.sort_values('Days')

colors = ['#' + ''.join([random.choice('0123456789ABCDEF') for j in range(6)]) for i in range(len(first_last))]

fig = ff.create_gantt(first_last, index_col = 'Country', colors = colors, show_colorbar = False,
                      bar_width=0.2, showgrid_x = True, showgrid_y = True, height = 2500)
fig.show()

```

