MINI PROJECT REPORT
ON
COVID-19 DATA ANALYSIS

UNDER THE GUIDENCE OF

Mr. Amir Khan

Technical Trainer

T&D dept.

GLA University, Mathura.

Under taken by

NIMISHA PACHAURI 181500430

PARAKH TIWARI 181500450

CONTENT

Abstract

This project is entitled to “Covid-19 Data Analysis”.

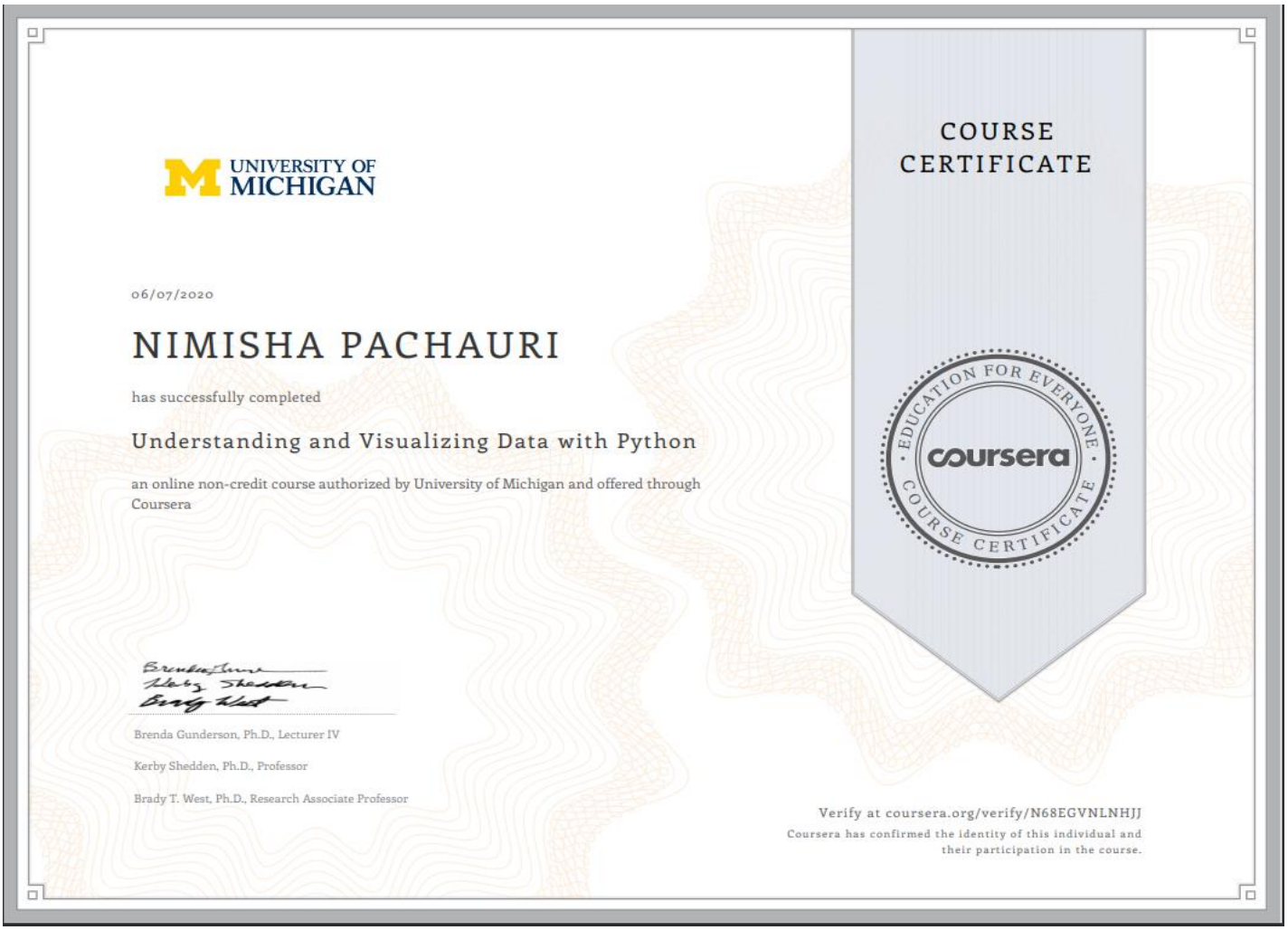
This project is an outcome of summer training done on “Introduction of Data Analysis for Business”. The training was based on how to play with data to extract useful information. A large amount of data has been short summarised and thoroughly analysed using data analysis.

The main objective of the project is to analyse the COVID-19 data of India and the world. In the process, using Jupyter Notebook, the data was accessed which was stored in a CSV file and then different queries were generated for analysing the data in the form of graphs, charts and different other diagrammatic format which made it easy to understand the spread of the disease.

The Covid data has been recorded in a CSV file which was quite difficult to be read and analyse for which this project has been done. Various tools have been used for analysing the data such as python pandas and Jupyter Notebook, after which a conclusion has been drawn about the data set.

With the help of this project, a huge amount of data has been easily analysed for making a brief report on the spread and contain of covid and the process of this has been explained in this report and the objective has been fulfilled.

Certificates



University of Colorado
Boulder | Colorado Springs | Denver | Anschutz Medical Campus

COURSE CERTIFICATE

05/25/2020

Parakh Tiwari

has successfully completed

Introduction to Data Analytics for Business

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

David A. Torgerson

David Torgerson

EDUCATION FOR EVERYONE · coursera · COURSE CERTIFICATE

Verify at coursera.org/verify/MQBQ7P5HJHNJ
Coursera has confirmed the identity of this individual and their participation in the course.

Objective

To analyse the data of covid19 contain and spread across the country and the world among the population using various diagrammatic formats to understand the spread of the pandemic covid-19 and make a brief report on the same in order help the authorities to do the possible to rein the flow of the disease and save lives. This project is nothing less than a contribution for our government and concerned authorities to understand the disease containment and bring back the normal routine life as it was before corona.

Introduction

The project aims at deeply analysing the contain and spread of the world pandemic Corona virus, also known as Covid19 in Indian subcontinent and other parts of the world, which has made a foot fall in December 2019 from the city of Wuhan, China. The project uses various technologies for analysing the spread of the virus through charts, graphs, tables, maps and various representations methods.

Today, as we all know how much the virus is spreading and making its roots across the country and the world, the data that we will represent in this project, will give us a thorough study of the numbers upto which the virus is spreading among the locals, how steeply the number of patients are rising, the number of recoveries, active patients and the number of deceased. It also gives a thorough study of the data of specified places where the cases are more, places where the cases are less, states which are able to control the disease, the population effected, groups of ages of the people who are effected, who among males and females are effected, who are at higher risk of infection, who has lower risk of infection and many other deep studies related to the pandemic.

The project also gives a time series analysis of the pandemic which helps us to predict the future projection of the disease, which will help us to plan the future actions that should be taken for controlling and stopping its contain.

This could help us to understand the extent of corona virus spread in the country and the world and, in turn, it could help us to study what effects

the measures have done, taken in the past, and now what shall be done in future to control the spread.

The project will reduce the searching and analysing of covid19 data for the users and analysts who has to search at different places for such data, maintain the record of the data and has to generate reports of the contain and spread of the data. The project will also help the government, doctors, police, media, journalists and other concerned authorities who require the data for analysing purpose and report making and to place the future actions that should be taken for controlling the same, knowing the past mistakes.

Problem

The problem is a large set of given data of patient details of covid-19 effected people of whom some are confirmed cases, some are active, some recovered and some were unfortunate deaths. To analyse the amount of people effected and those who are non-effected was difficult to understand. Those who are active and those who recovered from the disease were difficult to compare. Also to analyse the age groups effected the most, the people who succumbed to the disease was difficult to be known. To make the task easy, this project has been implemented.

With the implementation of this project, the problem of analysation has been resolved as the projection of every available data or information has been done separately in form of easily understandable formats to reduce the confusion generated by the data.

Implementation

The project uses python as a working methodology. Various libraries of python is been used in this project to make this project working more effective, efficient and apprehensive. The project works on Jupyter Notebook along with other technologies where each tech used in this project has a different purpose. Some techs give us data analysed in graphical form, whereas some techs represent data tabular form and all techs used in this project gives a better cumulative result and the purpose of the project is been fulfilled.

The main platform on which the project works if Jupyter Notebook. The other techs include are matplotlib, plotly, plotly.express , plotly.graph_objects , cufflinks, plotly.offline , folium.

What is Pandas?

It is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables. There are several ways to create a DataFrame.

What is NumPy?

NumPy is a python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

What is DataFrame?

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

Time Series

A **time series** is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data. Time Series analysis can be useful to see how a given asset, security or economic variable changes over time.

Time series are very frequently plotted via run charts (a temporal line chart). Time series are used in statistics, signal processing, pattern recognition, econometrics, mathematical finance, weather forecasting, earthquake prediction, electroencephalography, control engineering, astronomy, communications engineering, and largely in any domain of applied science and engineering which involves temporal measurements.

Time Series Analysis

Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values. While regression analysis is often employed in such a way as to test theories that the current values of one or more independent time series affect the current value of another time series, this type of analysis of time series is not called "time series analysis", which focuses on comparing values of a single time series or multiple dependent time series at different points in time. Interrupted analysis is the analysis of interventions on a single time series.

Time series data have a natural temporal ordering. This makes time series analysis distinct from cross-sectional studies, in which there is no natural ordering of the observations (e.g. explaining people's wages by reference to their respective education levels, where the individuals' data could be entered in any order). Time series analysis is also distinct from spatial data analysis where the observations typically relate to geographical locations (e.g. accounting for house prices by the location as well as the intrinsic characteristics of the houses). A stochastic model for a time series will generally reflect the fact that observations close together in time will be more closely related than observations further apart. In addition, time series models will often make use of the natural one-way ordering of time so that values for a given period will be expressed as deriving in some way from past values, rather than from future values (see time reversibility.)

Methods of Analysis

Methods for time series analysis may be divided into two classes: frequency domain methods and time-domain methods. The former include spectral analysis and wavelet analysis; the latter include auto-corrected and cross-correlation analysis. In the time domain, correlation and analysis can be made in a filter-like manner using scaled correlation, thereby mitigating the need to operate in the frequency domain.

Additionally, time series analysis techniques may be divided into parametric and non-parametric methods. The parametric approaches assume that the underlying stationary schastic has a certain structure which can be described using a small number of parameters (for example, using an autoregressive or moving average model). In these approaches, the task is to estimate the parameters of the model that describes the stochastic process. By contrast, non-parametric approaches explicitly estimate the covariance or the spectrum of the process without assuming that the process has any particular structure.

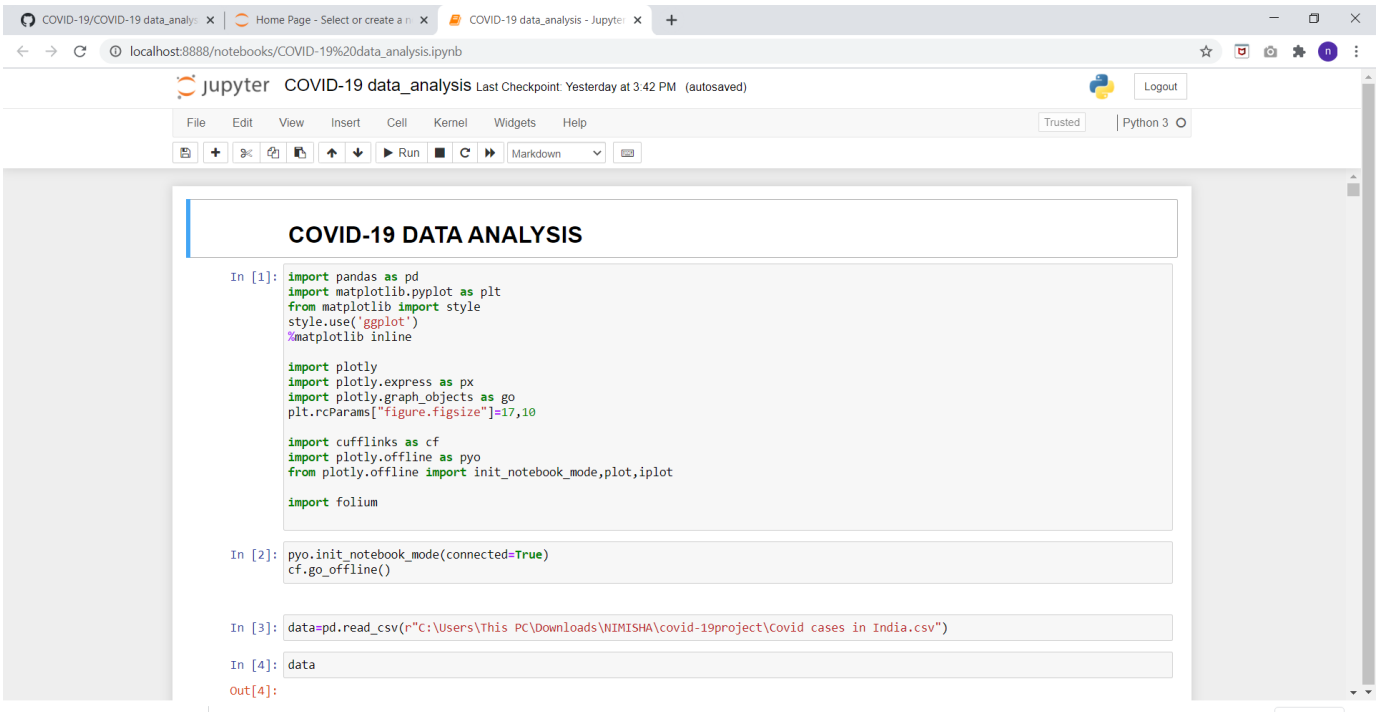
Methods of time series analysis may also be divided into linear and non-linear, and univariate and multivariate.

Implementation of the project work :-

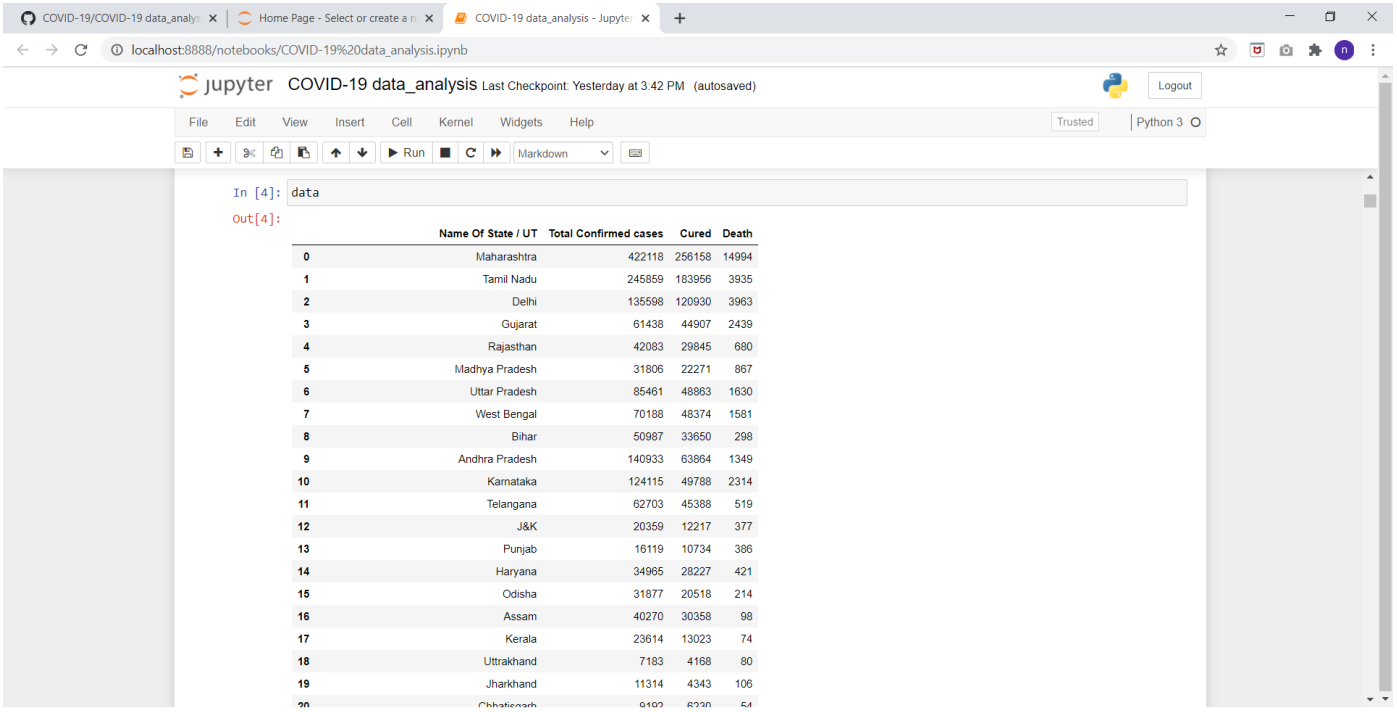
Screenshots

Importing libraries of python

- Pandas
- Matplotlib
- Plotly
- Plotly.express
- Cufflinks
- Folium



Data that has been taken to work upon in this project.



COVID-19/COVID-19 data_analys | Home Page - Select or create a n | COVID-19 data_analysis - Jupyter

localhost:8888/notebooks/COVID-19%20data_analysis.ipynb

jupyter COVID-19 data_analysis Last Checkpoint: Yesterday at 3:42 PM (autosaved) Logout

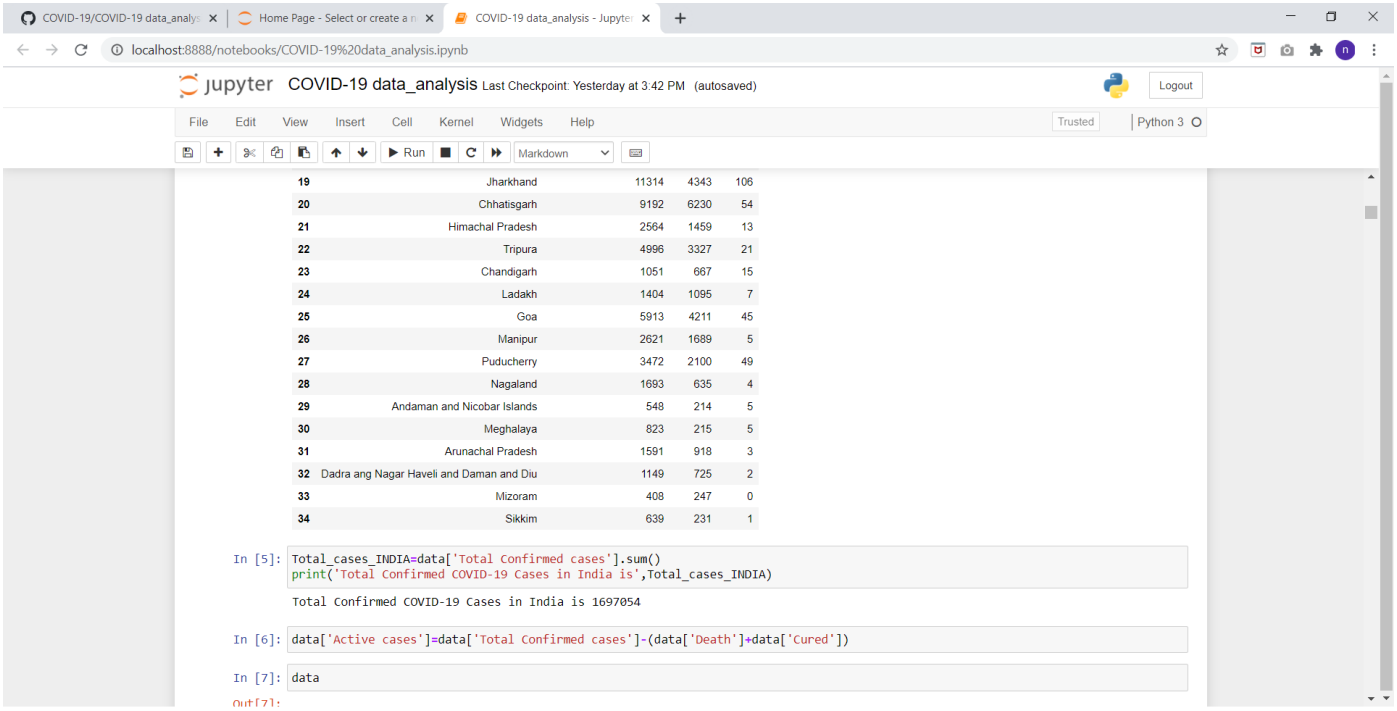
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Markdown

In [4]:
```
data
```

Out[4]:

	Name Of State / UT	Total Confirmed cases	Cured	Death
0	Maharashtra	422118	256158	14994
1	Tamil Nadu	245859	183956	3935
2	Delhi	135598	120930	3963
3	Gujarat	61438	44907	2439
4	Rajasthan	42083	29845	680
5	Madhya Pradesh	31806	22271	867
6	Uttar Pradesh	85461	48863	1630
7	West Bengal	70188	48374	1581
8	Bihar	50987	33650	298
9	Andhra Pradesh	140933	63864	1349
10	Karnataka	124115	49788	2314
11	Telangana	62703	45388	519
12	J&K	20359	12217	377
13	Punjab	16119	10734	386
14	Haryana	34965	28227	421
15	Odisha	31877	20518	214
16	Assam	40270	30358	98
17	Kerala	23614	13023	74
18	Uttrakhand	7183	4168	80
19	Jharkhand	11314	4343	106
20	Chhatisgarh	9192	6230	54



COVID-19/COVID-19 data_analys | Home Page - Select or create a n | COVID-19 data_analysis - Jupyter

localhost:8888/notebooks/COVID-19%20data_analysis.ipynb

jupyter COVID-19 data_analysis Last Checkpoint: Yesterday at 3:42 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Markdown

19	Jharkhand	11314	4343	106
20	Chhatisgarh	9192	6230	54
21	Himachal Pradesh	2564	1459	13
22	Tripura	4996	3327	21
23	Chandigarh	1051	667	15
24	Ladakh	1404	1095	7
25	Goa	5913	4211	45
26	Manipur	2621	1689	5
27	Puducherry	3472	2100	49
28	Nagaland	1693	635	4
29	Andaman and Nicobar Islands	548	214	5
30	Meghalaya	823	215	5
31	Arunachal Pradesh	1591	918	3
32	Dadra ang Nagar Haveli and Daman and Diu	1149	725	2
33	Mizoram	408	247	0
34	Sikkim	639	231	1

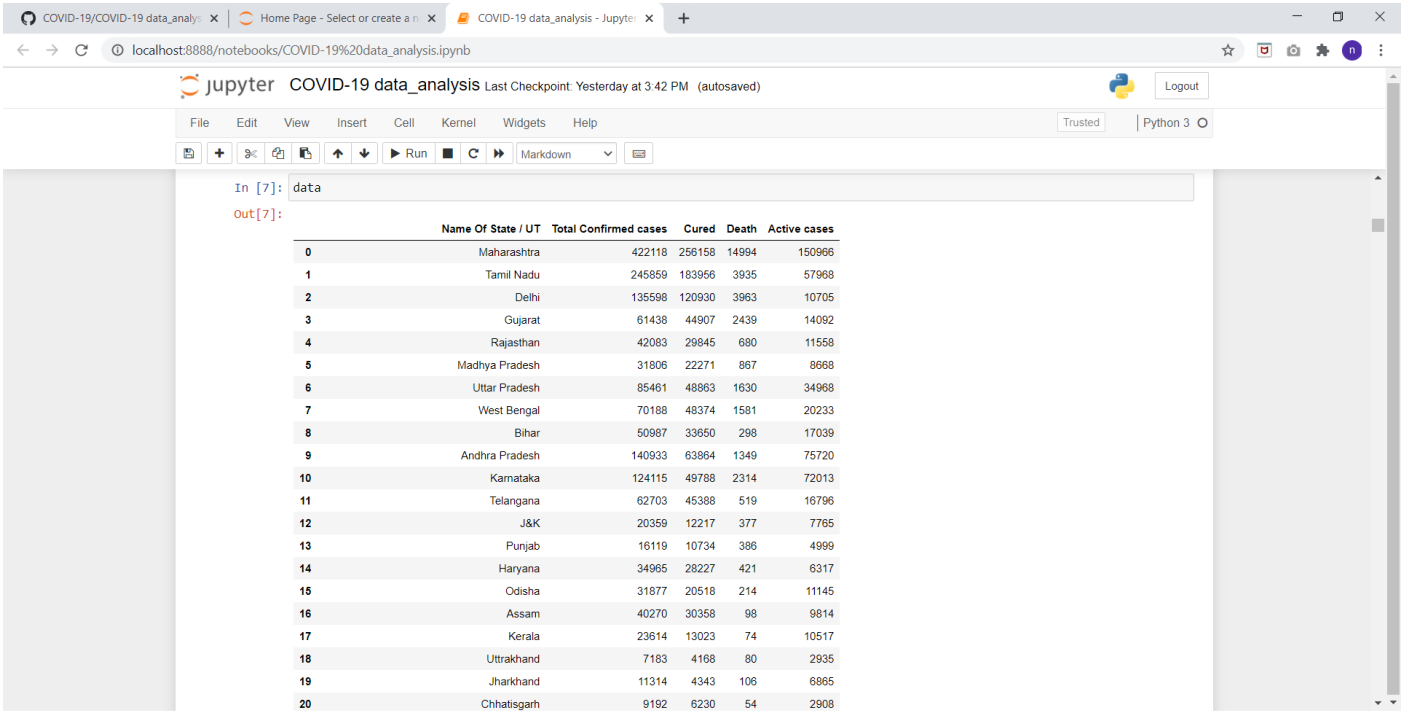
In [5]:
```
Total_cases_INDIA=data['Total Confirmed cases'].sum()
print('Total Confirmed COVID-19 Cases in India is',Total_cases_INDIA)
```

```
Total Confirmed COVID-19 Cases in India is 1697054
```

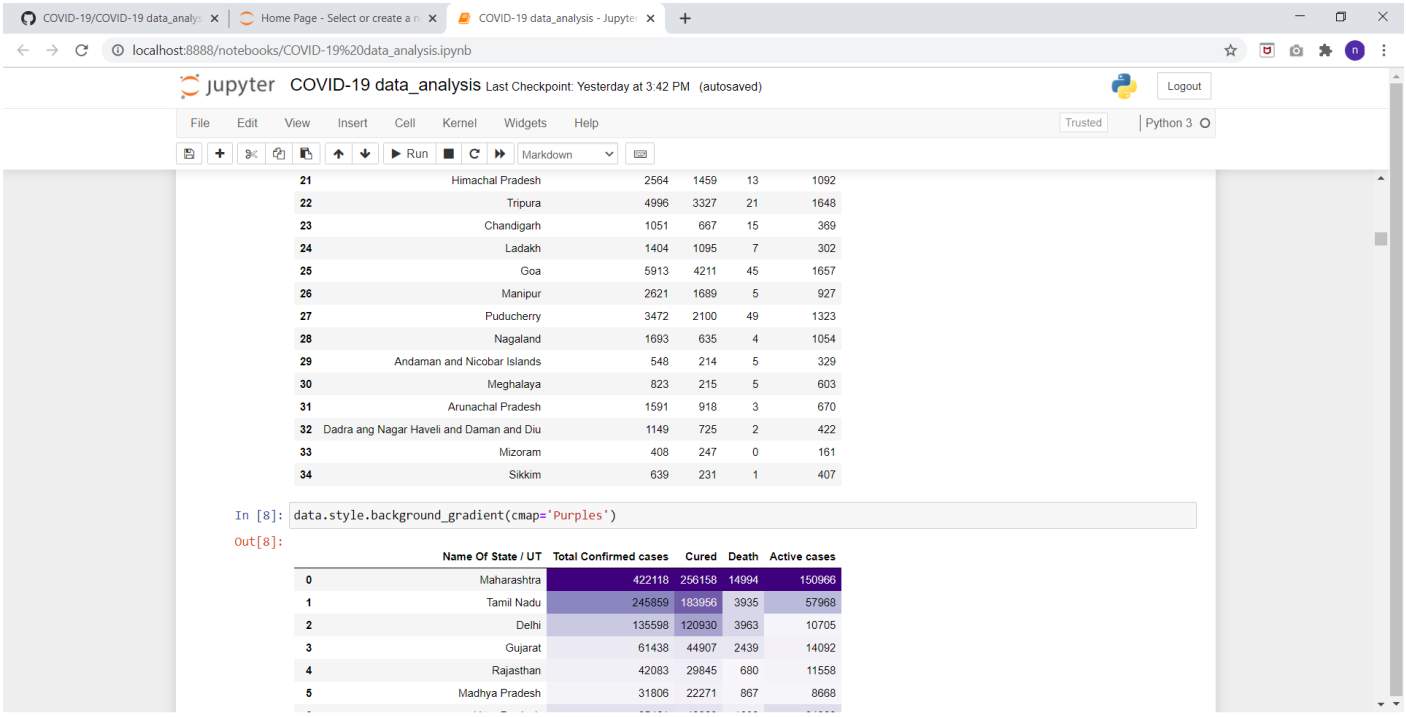
In [6]:
```
data['Active cases']=data['Total Confirmed cases']-(data['Death']+data['Cured'])
```

In [7]:
```
data
```

Out[7]:

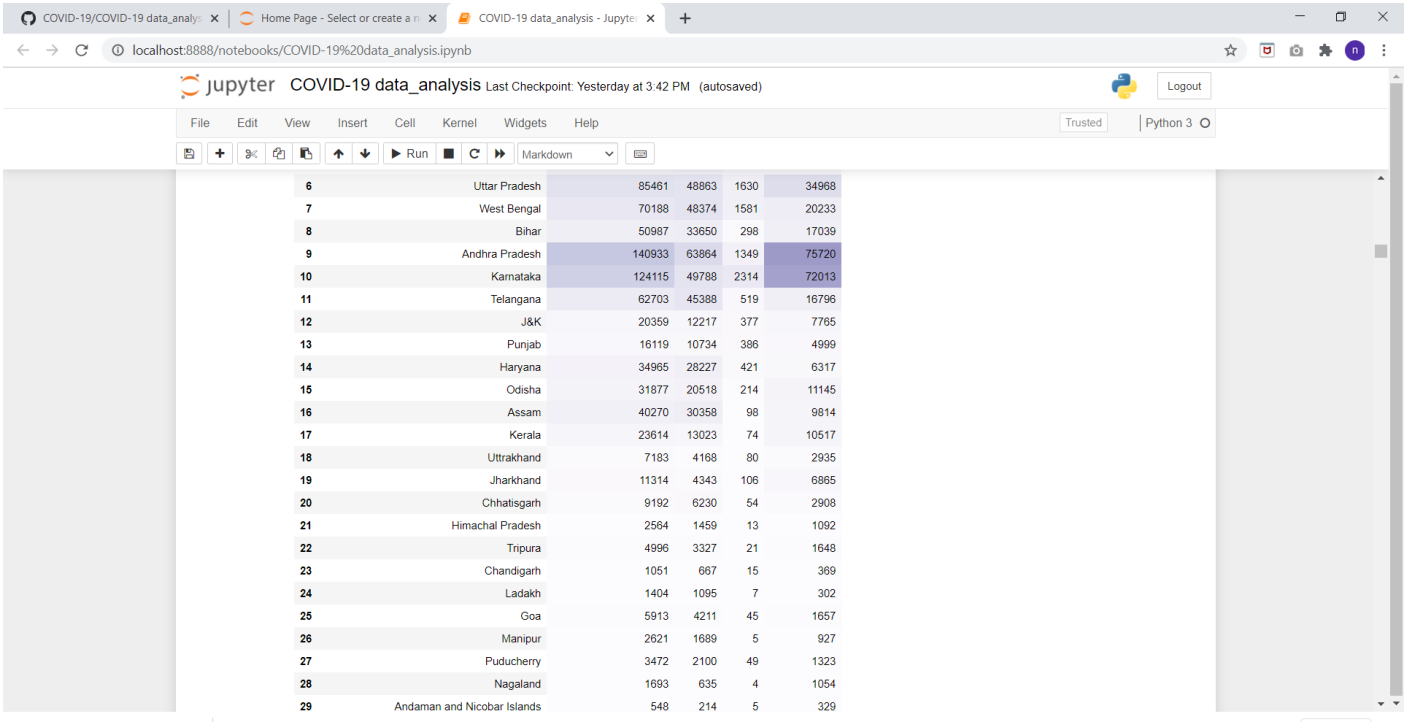


	Name Of State / UT	Total Confirmed cases	Cured	Death	Active cases
0	Maharashtra	422118	256158	14994	150966
1	Tamil Nadu	245859	183956	3935	57968
2	Delhi	135598	120930	3963	10705
3	Gujarat	61438	44907	2439	14092
4	Rajasthan	42083	29845	680	11558
5	Madhya Pradesh	31806	22271	867	8668
6	Uttar Pradesh	85461	48863	1630	34968
7	West Bengal	70188	48374	1581	20233
8	Bihar	50987	33650	298	17039
9	Andhra Pradesh	140933	63864	1349	75720
10	Karnataka	124115	49788	2314	72013
11	Telangana	62703	45388	519	16796
12	J&K	20359	12217	377	7765
13	Punjab	16119	10734	386	4999
14	Haryana	34965	28227	421	6317
15	Odisha	31877	20518	214	11145
16	Assam	40270	30358	98	9814
17	Kerala	23614	13023	74	10517
18	Uttrakhand	7183	4168	80	2935
19	Jharkhand	11314	4343	106	6865
20	Chhatisgarh	9192	6230	54	2908

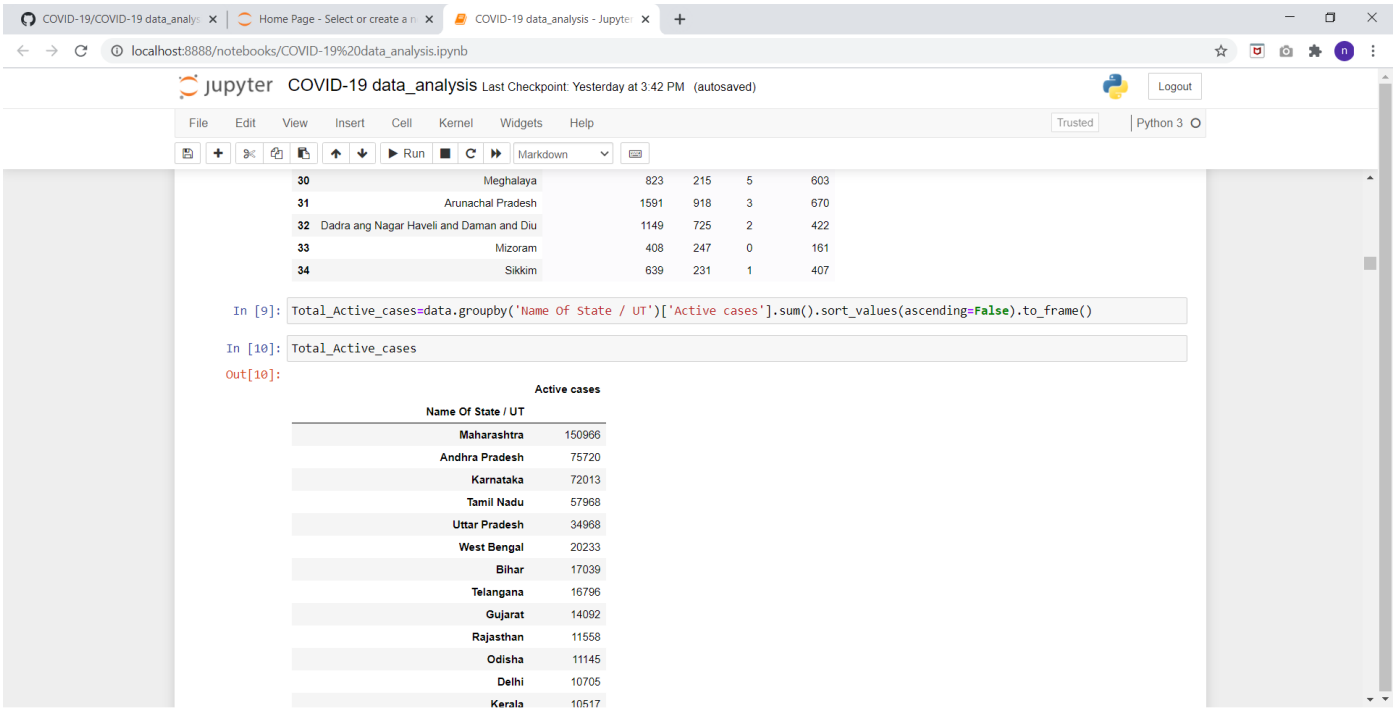


21	Himachal Pradesh	2564	1459	13	1092
22	Tripura	4996	3327	21	1648
23	Chandigarh	1051	667	15	369
24	Ladakh	1404	1095	7	302
25	Goa	5913	4211	45	1657
26	Manipur	2621	1689	5	927
27	Puducherry	3472	2100	49	1323
28	Nagaland	1693	635	4	1054
29	Andaman and Nicobar Islands	548	214	5	329
30	Meghalaya	823	215	5	603
31	Arunachal Pradesh	1591	918	3	670
32	Dadra ang Nagar Haveli and Daman and Diu	1149	725	2	422
33	Mizoram	408	247	0	161
34	Sikkim	639	231	1	407

	Name Of State / UT	Total Confirmed cases	Cured	Death	Active cases
0	Maharashtra	422118	256158	14994	150966
1	Tamil Nadu	245859	183956	3935	57968
2	Delhi	135598	120930	3963	10705
3	Gujarat	61438	44907	2439	14092
4	Rajasthan	42083	29845	680	11558
5	Madhya Pradesh	31806	22271	867	8668



6	Uttar Pradesh	85461	48863	1630	34968
7	West Bengal	70188	48374	1581	20233
8	Bihar	50987	33650	298	17039
9	Andhra Pradesh	140933	63864	1349	75720
10	Karnataka	124115	49788	2314	72013
11	Telangana	62703	45388	519	16796
12	J&K	20359	12217	377	7765
13	Punjab	16119	10734	386	4999
14	Haryana	34965	28227	421	6317
15	Odisha	31877	20518	214	11145
16	Assam	40270	30358	98	9814
17	Kerala	23614	13023	74	10517
18	Uttrakhand	7183	4168	80	2935
19	Jharkhand	11314	4343	106	6865
20	Chhatisgarh	9192	6230	54	2908
21	Himachal Pradesh	2564	1459	13	1092
22	Tripura	4996	3327	21	1648
23	Chandigarh	1051	667	15	369
24	Ladakh	1404	1095	7	302
25	Goa	5913	4211	45	1657
26	Manipur	2621	1689	5	927
27	Puducherry	3472	2100	49	1323
28	Nagaland	1693	635	4	1054
29	Andaman and Nicobar Islands	548	214	5	329



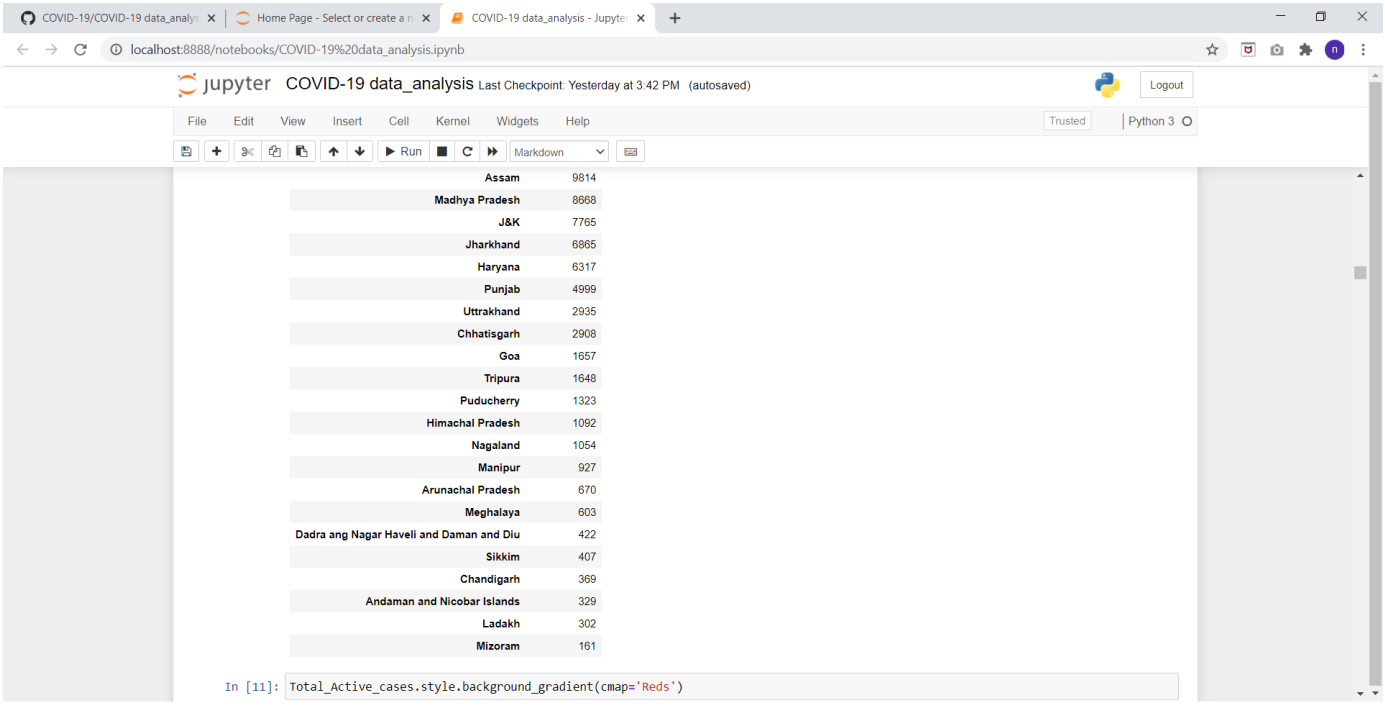
30	Meghalaya	823	215	5	603
31	Arunachal Pradesh	1591	918	3	670
32	Dadra ang Nagar Haveli and Daman and Diu	1149	725	2	422
33	Mizoram	408	247	0	161
34	Sikkim	639	231	1	407

In [9]:
```
Total_Active_cases=data.groupby('Name Of State / UT')['Active cases'].sum().sort_values(ascending=False).to_frame()
```

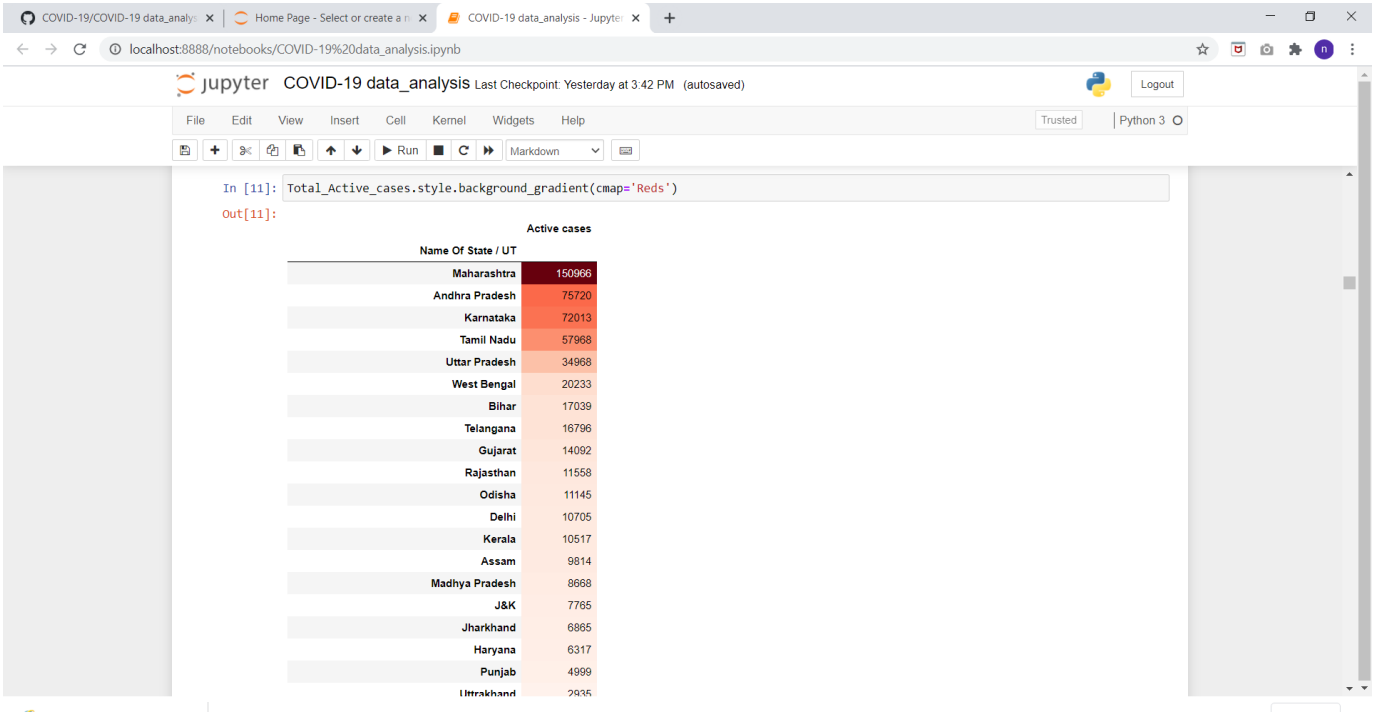
In [10]:
```
Total_Active_cases
```

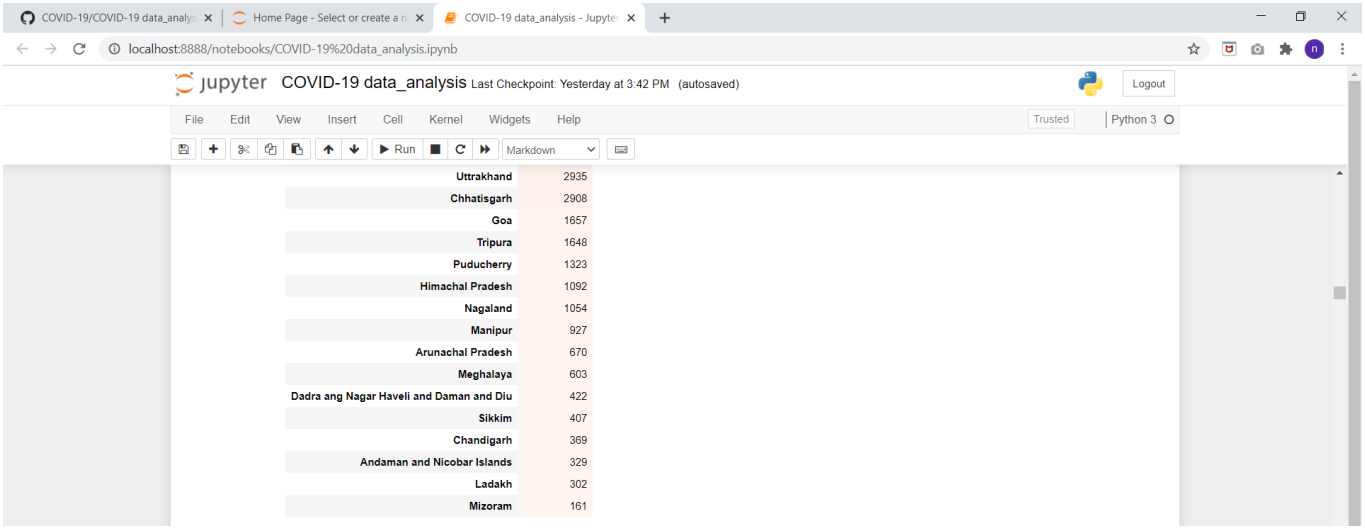
Out[10]:

Name Of State / UT	Active cases
Maharashtra	150966
Andhra Pradesh	75720
Karnataka	72013
Tamil Nadu	57968
Uttar Pradesh	34968
West Bengal	20233
Bihar	17039
Telangana	16796
Gujarat	14092
Rajasthan	11558
Odisha	11145
Delhi	10705
Kerala	10517



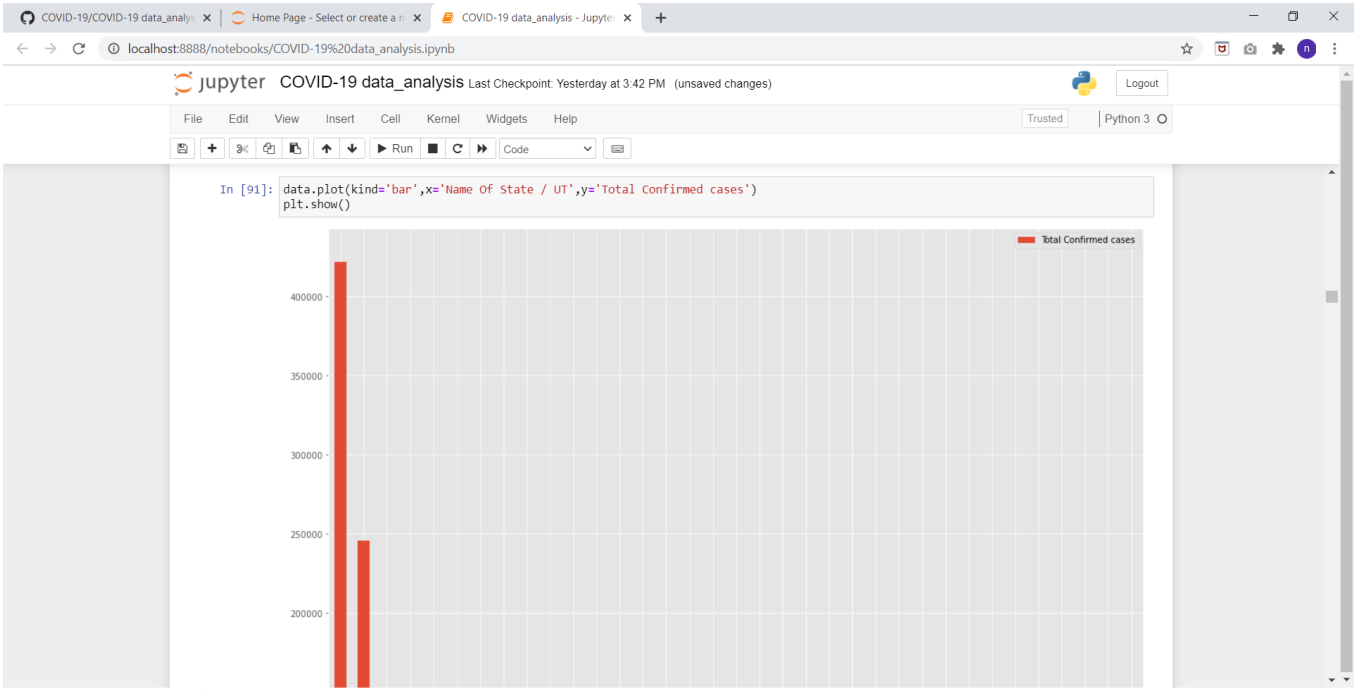
Representation of active cases of corona virus.

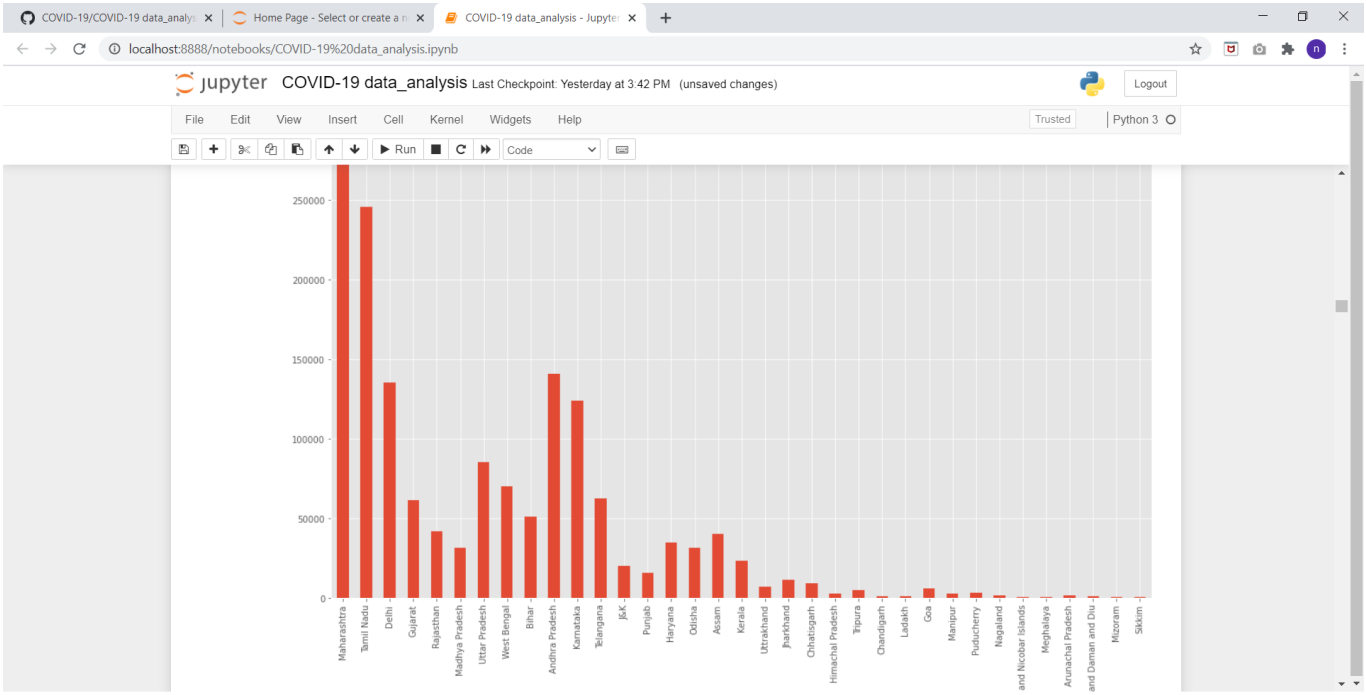


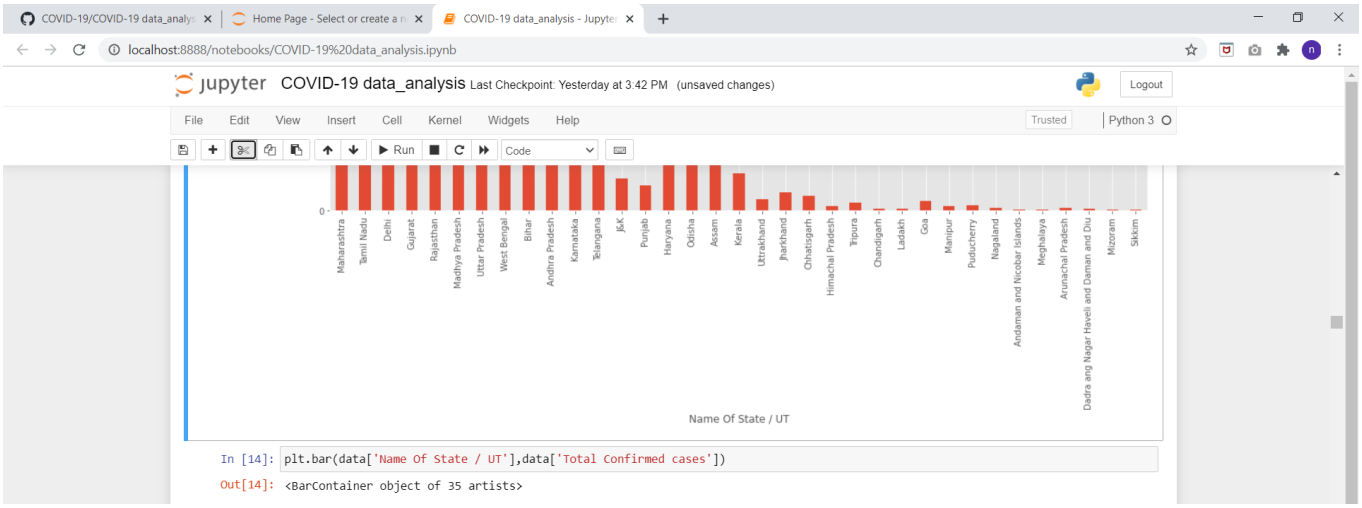


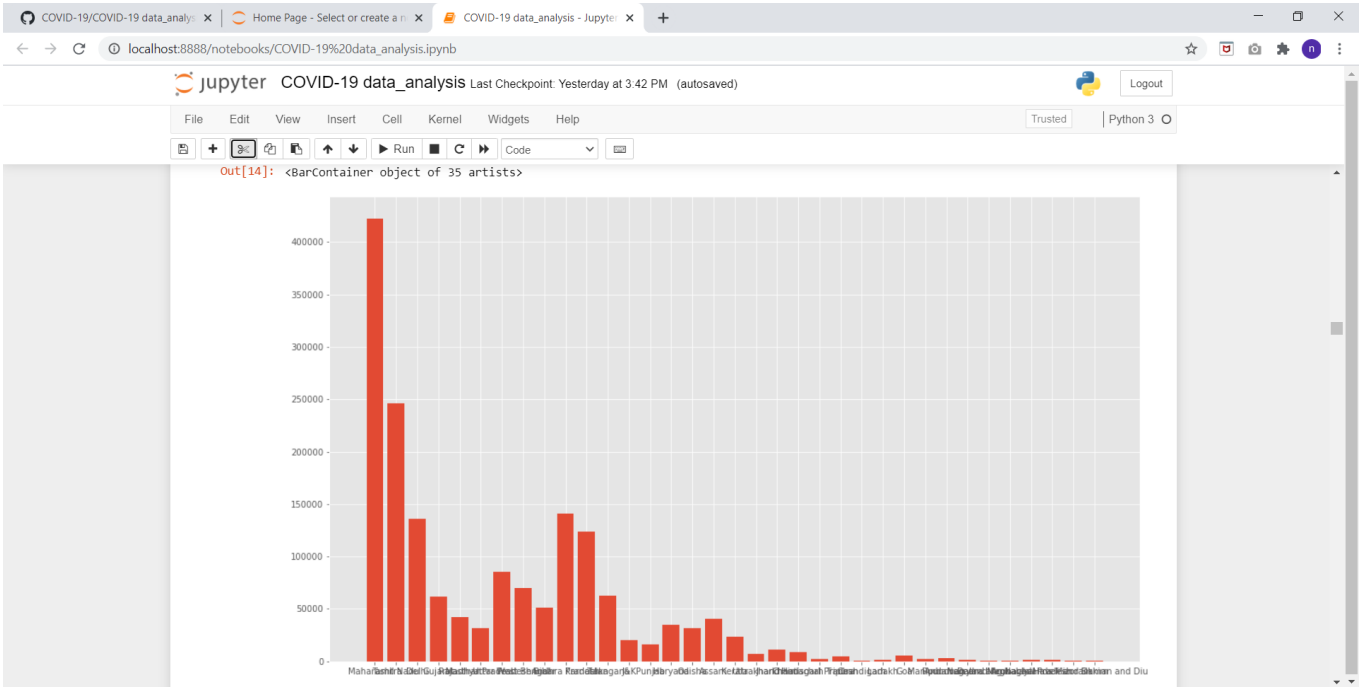
Uttrakhand	2935
Chhatisgarh	2908
Goa	1657
Tripura	1648
Puducherry	1323
Himachal Pradesh	1092
Nagaland	1054
Manipur	927
Arunachal Pradesh	670
Meghalaya	603
Dadra ang Nagar Haveli and Daman and Diu	422
Sikkim	407
Chandigarh	369
Andaman and Nicobar Islands	329
Ladakh	302
Mizoram	161

Bar Graph representation of confirmed cases of covid-19.

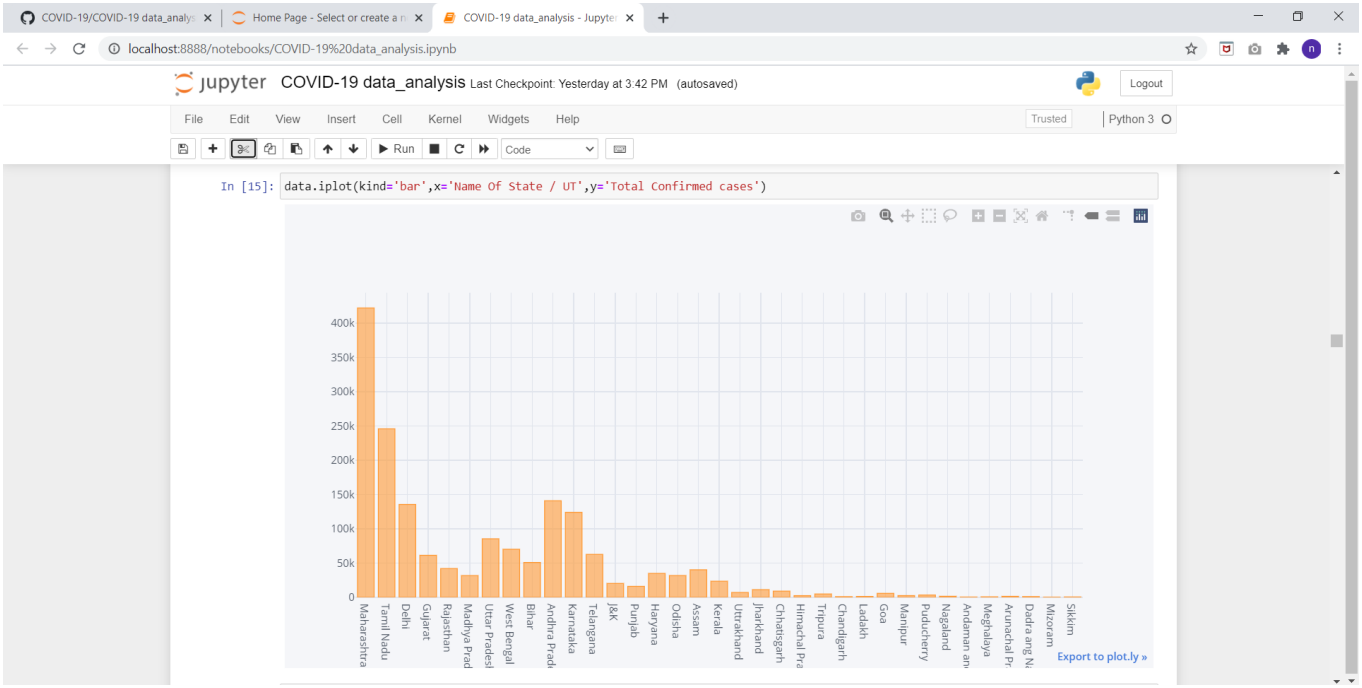


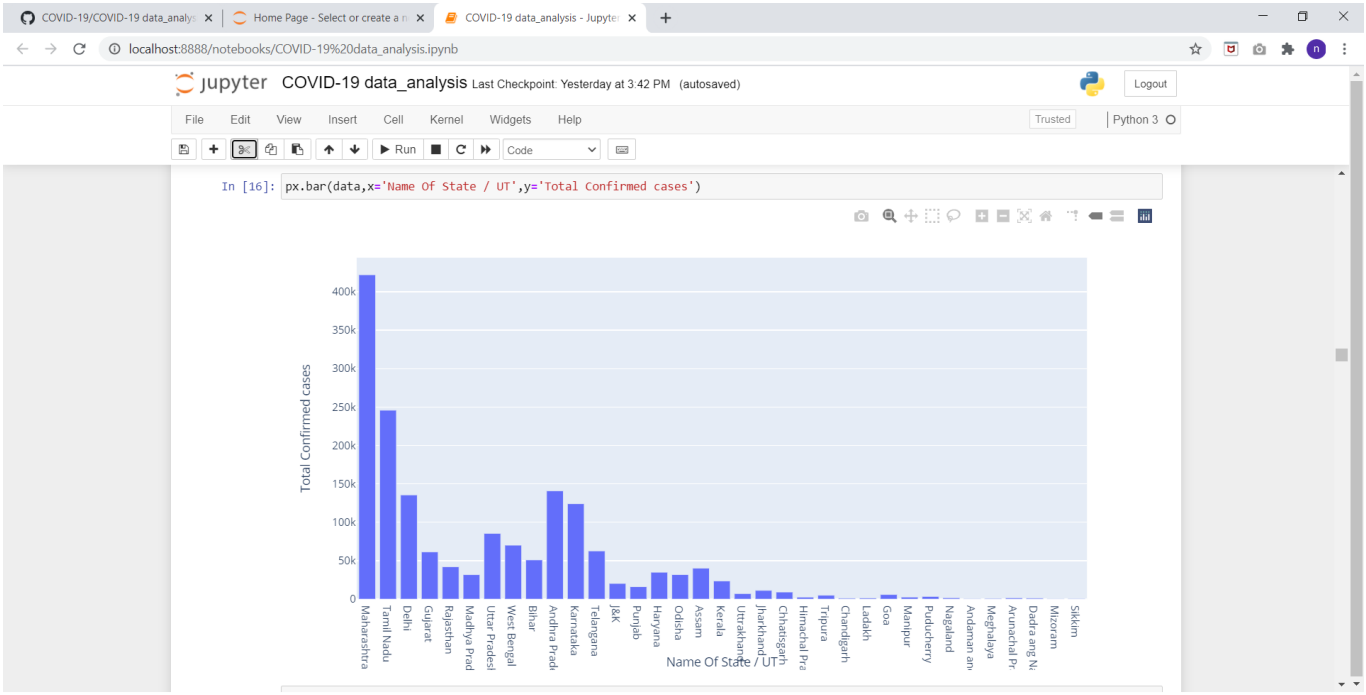

COVID-19/COVID-19 data_analys
Home Page - Select or create a n
COVID-19 data_analysis - Jupyter
localhost:8888/notebooks/COVID-19%20data_analysis.ipynb
jupyter COVID-19 data_analysis Last Checkpoint: Yesterday at 3:42 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
250000
200000
150000
100000
50000
0
Maharashtra
Tamil Nadu
Delhi
Gujarat
Rajasthan
Madhya Pradesh
Uttar Pradesh
West Bengal
Bihar
Andhra Pradesh
Karnataka
Telangana
J&K
Punjab
Haryana
Odisha
Assam
Kerala
Uttrakhand
Jharkhand
Chhatisgarh
Himachal Pradesh
Tripura
Chandigarh
Ladakh
Goa
Manipur
Puducherry
Nagaland
and Nicobar Islands
Meghalaya
Arunachal Pradesh
and Daman and Diu
Mizoram
Sikkim


COVID-19/COVID-19 data_analys
Home Page - Select or create a n
COVID-19 data_analysis - Jupyter
localhost:8888/notebooks/COVID-19%20data_analysis.ipynb
jupyter COVID-19 data_analysis Last Checkpoint: Yesterday at 3:42 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
0
Maharashtra
Tamil Nadu
Delhi
Gujarat
Rajasthan
Madhya Pradesh
Uttar Pradesh
West Bengal
Bihar
Andhra Pradesh
Karnataka
Telangana
J&K
Punjab
Haryana
Odisha
Assam
Kerala
Uttrakhand
Jharkhand
Chhatisgarh
Himachal Pradesh
Tripura
Chandigarh
Ladakh
Goa
Manipur
Puducherry
Nagaland
Andaman and Nicobar Islands
Meghalaya
Arunachal Pradesh
Dadra and Nagar Haveli and Daman and Diu
Mizoram
Sikkim
Name Of State / UT
In [14]: plt.bar(data['Name Of State / UT'],data['Total Confirmed cases'])
Out[14]: <BarContainer object of 35 artists>

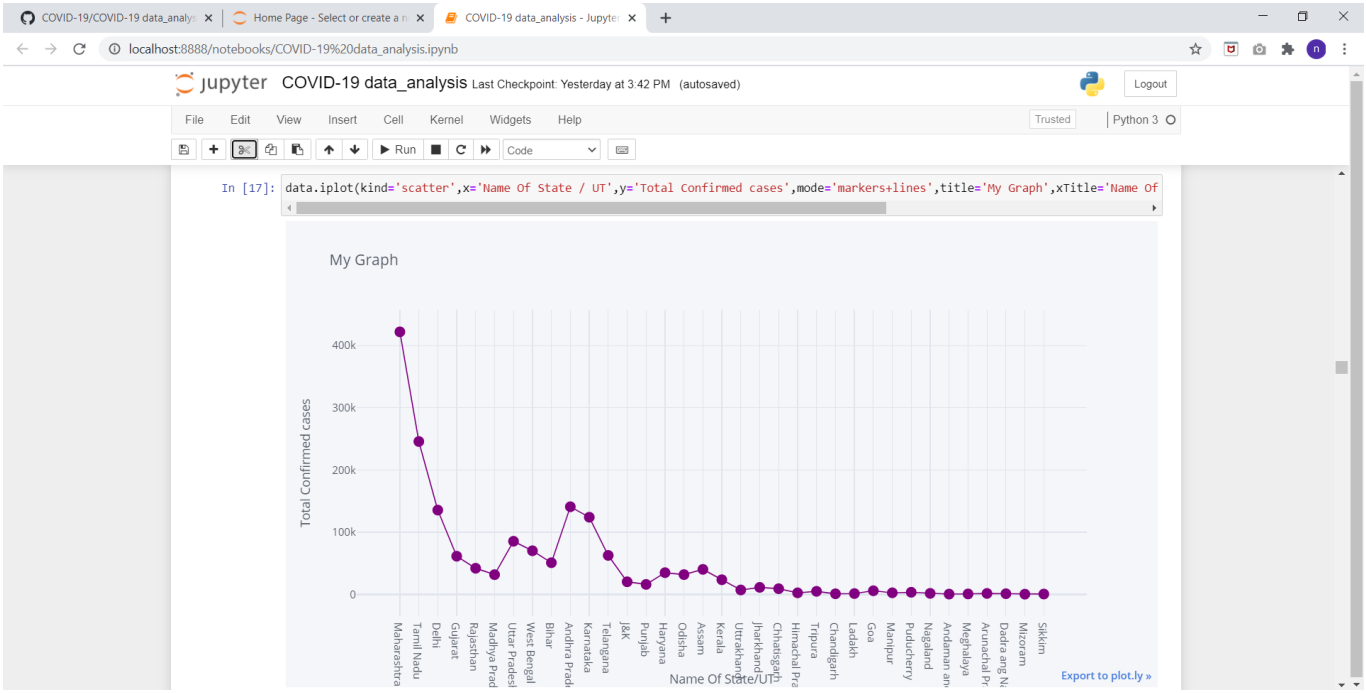


Bar Graph representation of confirmed cases of covid-19 using iplot.

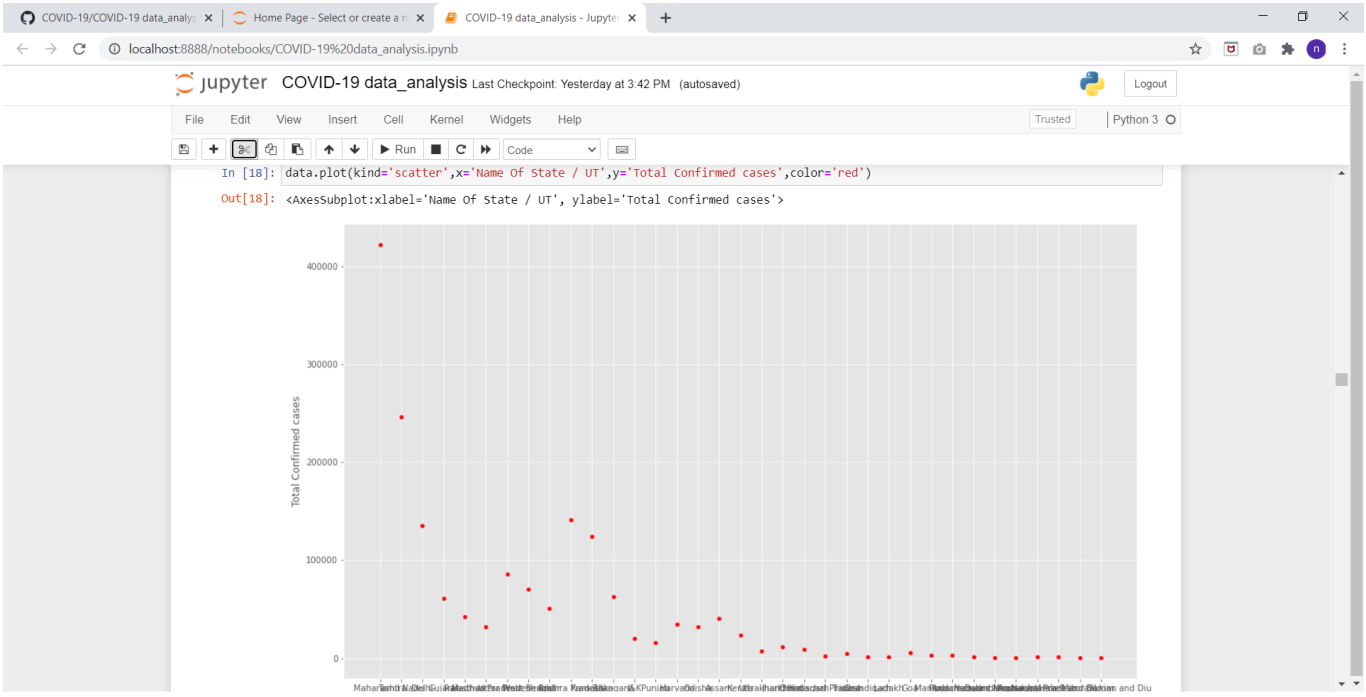


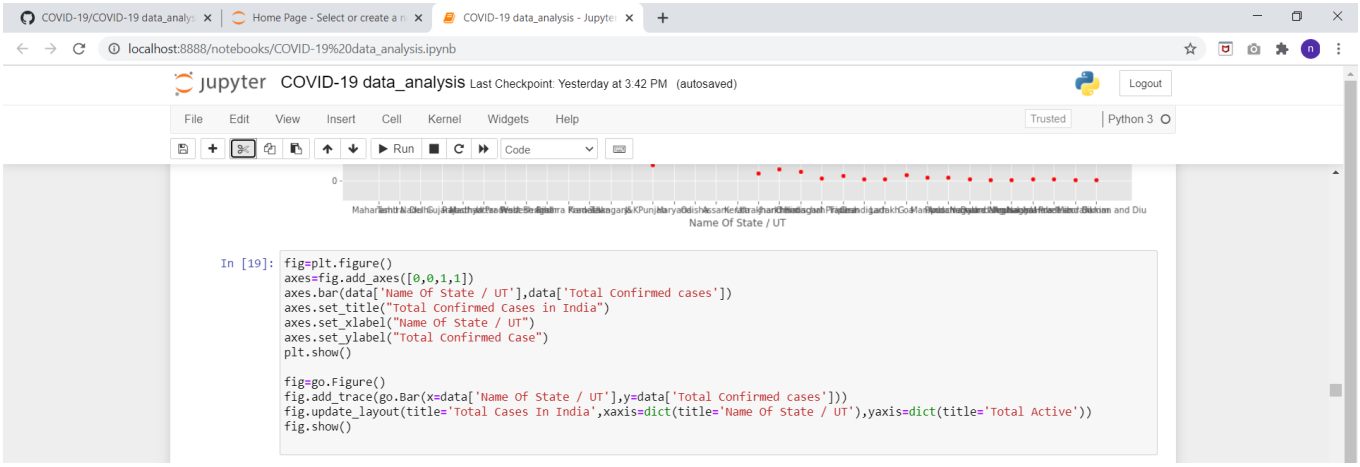


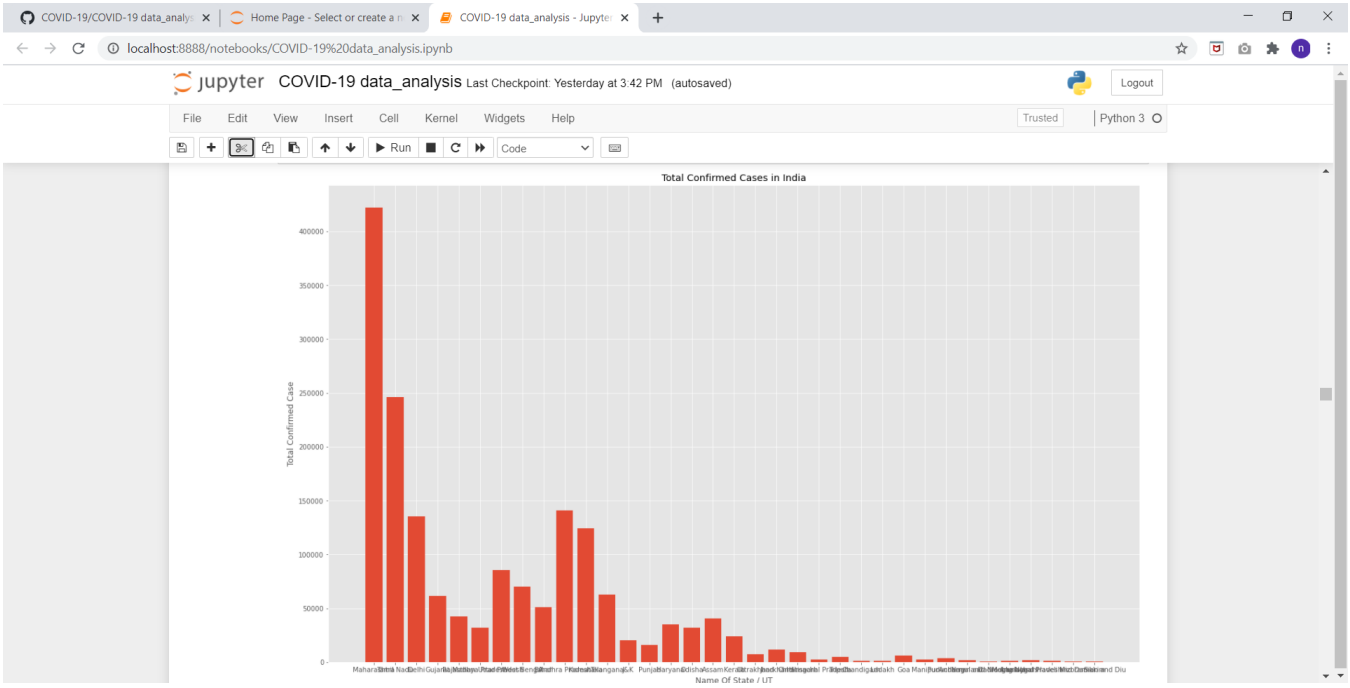
Line Graph representation of confirmed cases of covid-19.

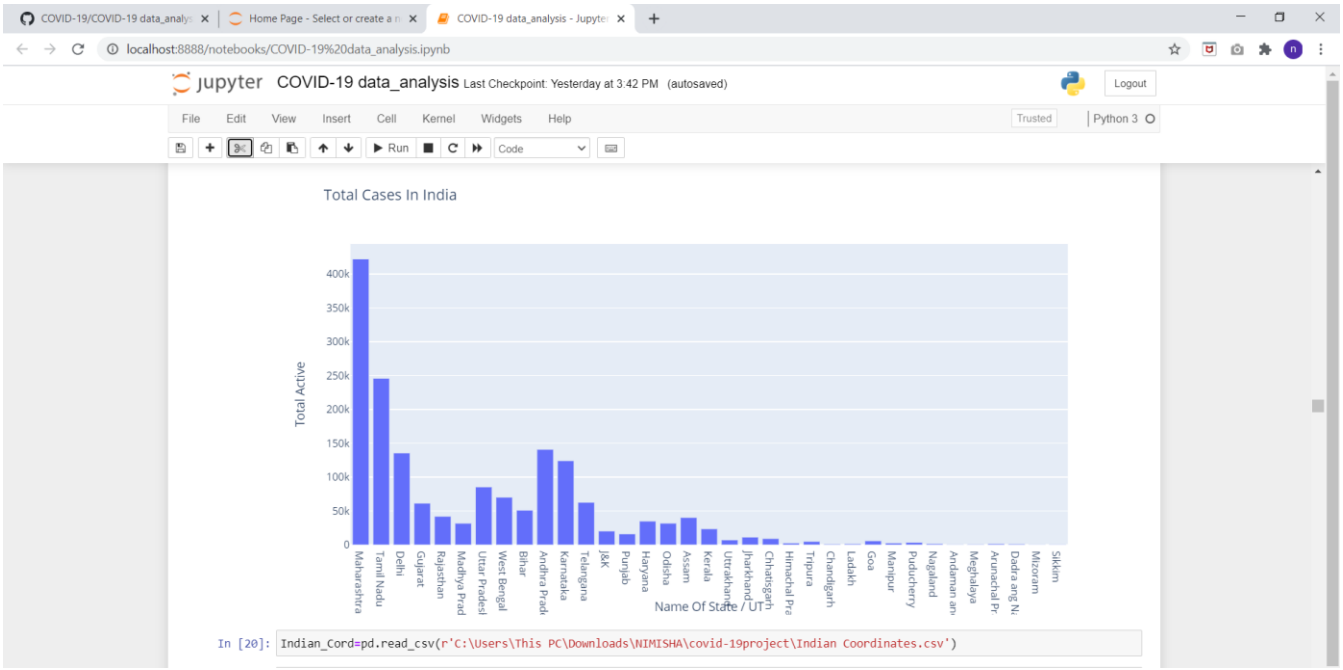


Dotted representation of the data x-axis=states, y-axis= total confirmed cases.

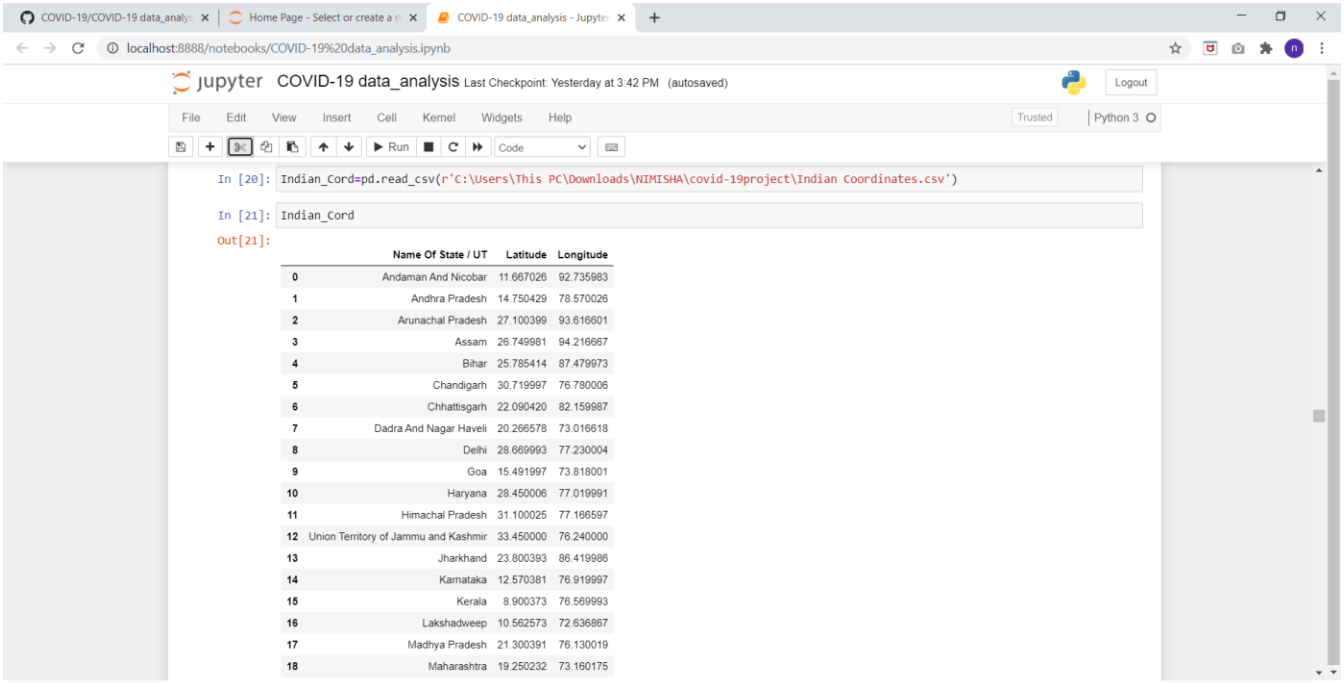

COVID-19/COVID-19 data_analys
Home Page - Select or create a n
COVID-19 data_analysis - Jupyter
localhost:8888/notebooks/COVID-19%20data_analysis.ipynb
jupyter COVID-19 data_analysis Last Checkpoint: Yesterday at 3:42 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [18]: data.plot(kind='scatter',x='Name Of State / UT',y='Total Confirmed cases',color='red')
Out[18]: <AxesSubplot:xlabel='Name Of State / UT', ylabel='Total Confirmed cases'>
Total Confirmed cases
400000
300000
200000
100000
0


COVID-19/COVID-19 data_analys
Home Page - Select or create a n
COVID-19 data_analysis - Jupyter
localhost:8888/notebooks/COVID-19%20data_analysis.ipynb
jupyter COVID-19 data_analysis Last Checkpoint: Yesterday at 3:42 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
0
Name Of State / UT
In [19]:
fig=plt.figure()
axes=fig.add_axes([0,0,1,1])
axes.bar(data['Name Of State / UT'],data['Total Confirmed cases'])
axes.set_title("Total Confirmed Cases in India")
axes.set_xlabel("Name Of State / UT")
axes.set_ylabel("Total Confirmed Case")
plt.show()
fig=go.Figure()
fig.add_trace(go.Bar(x=data['Name Of State / UT'],y=data['Total Confirmed cases']))
fig.update_layout(title='Total Cases In India',xaxis=dict(title='Name Of State / UT'),yaxis=dict(title='Total Active'))
fig.show()

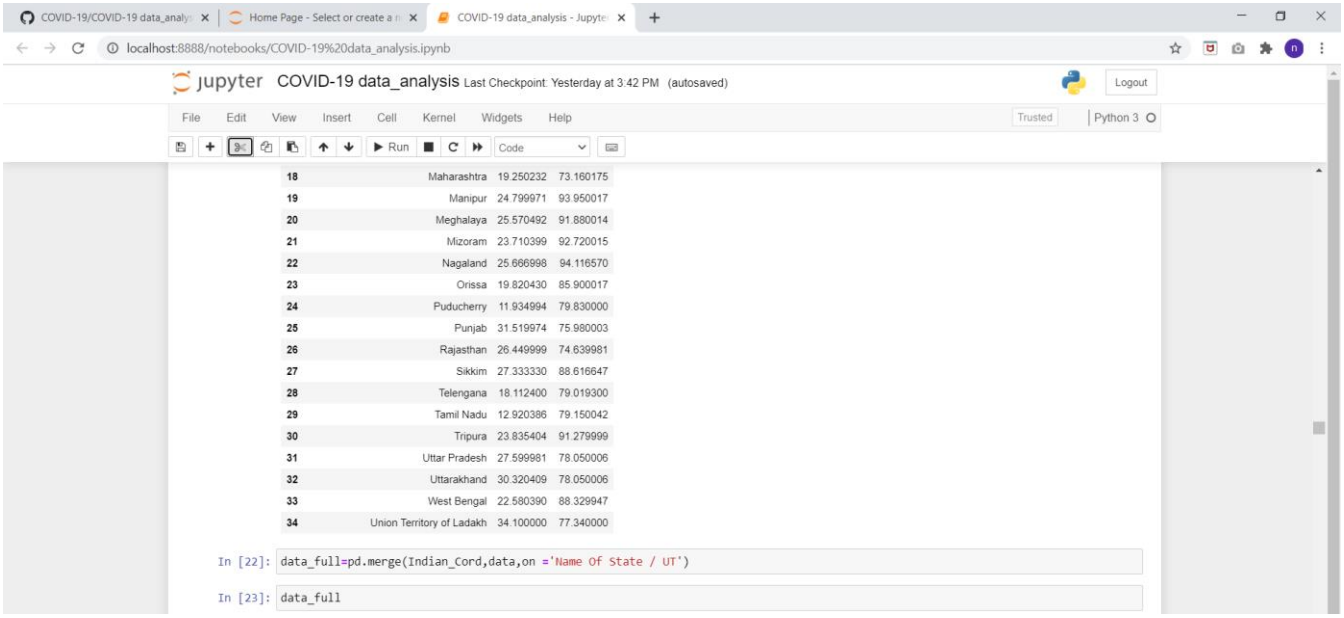




Latitude and longitude data of the places the data is taken of.

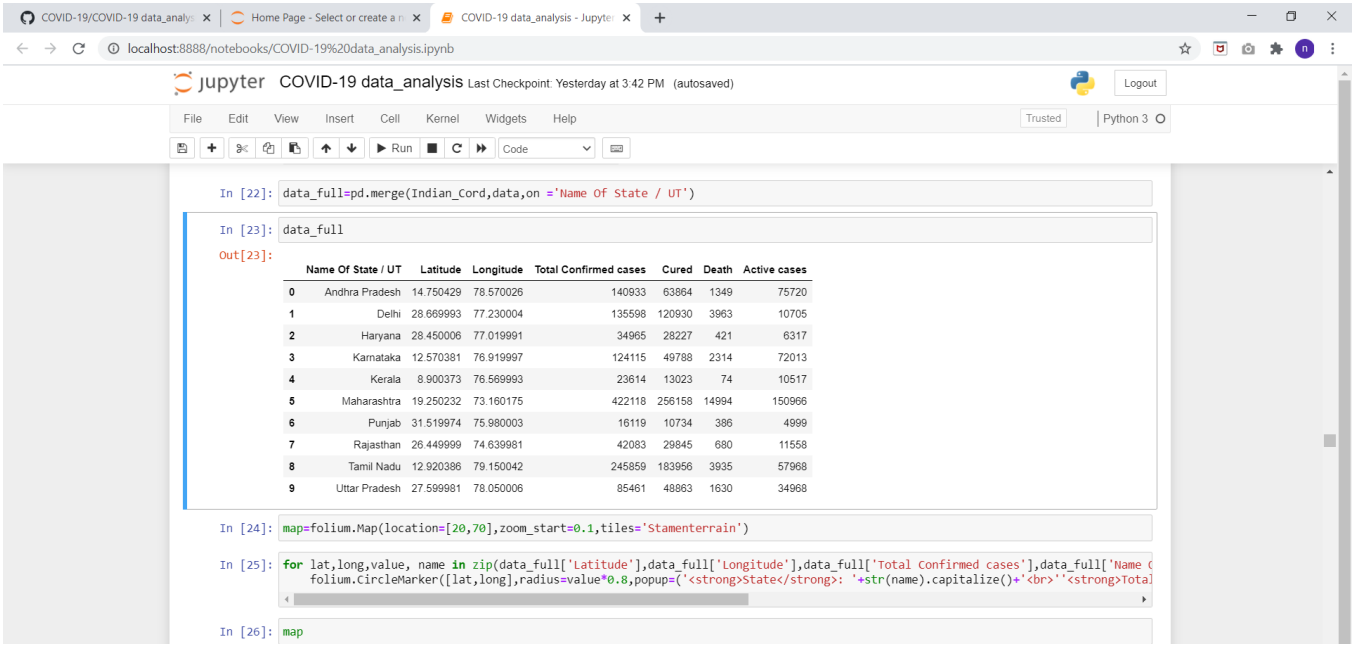


	Name Of State / UT	Latitude	Longitude
0	Andaman And Nicobar	11.667026	92.735983
1	Andhra Pradesh	14.750429	78.570026
2	Arunachal Pradesh	27.100399	93.616601
3	Assam	26.749981	94.216667
4	Bihar	25.785414	87.479973
5	Chandigarh	30.719997	76.780006
6	Chhattisgarh	22.090420	82.159987
7	Dadra And Nagar Haveli	20.266578	73.016618
8	Delhi	28.669993	77.230004
9	Goa	15.491997	73.818001
10	Haryana	28.450006	77.019991
11	Himachal Pradesh	31.100025	77.166597
12	Union Territory of Jammu and Kashmir	33.450000	76.240000
13	Jharkhand	23.800393	86.419986
14	Karnataka	12.570381	76.919997
15	Kerala	8.900373	76.569993
16	Lakshadweep	10.562573	72.636867
17	Madhya Pradesh	21.300391	76.130019
18	Maharashtra	19.250232	73.160175



18	Maharashtra	19.250232	73.160175
19	Manipur	24.799971	93.950017
20	Meghalaya	25.570492	91.880014
21	Mizoram	23.710399	92.720015
22	Nagaland	25.666998	94.116570
23	Orissa	19.820430	85.900017
24	Puducherry	11.934994	79.830000
25	Punjab	31.519974	75.980003
26	Rajasthan	26.449999	74.639981
27	Sikkim	27.333330	88.616647
28	Telengana	18.112400	79.019300
29	Tamil Nadu	12.920386	79.150042
30	Tripura	23.835404	91.279999
31	Uttar Pradesh	27.599981	78.050006
32	Uttarakhand	30.320409	78.050006
33	West Bengal	22.580390	88.329947
34	Union Territory of Ladakh	34.100000	77.340000

Merging the data sheets of covid-19 death,recovery,active and total cases with the data sheet of longitude and latitude.



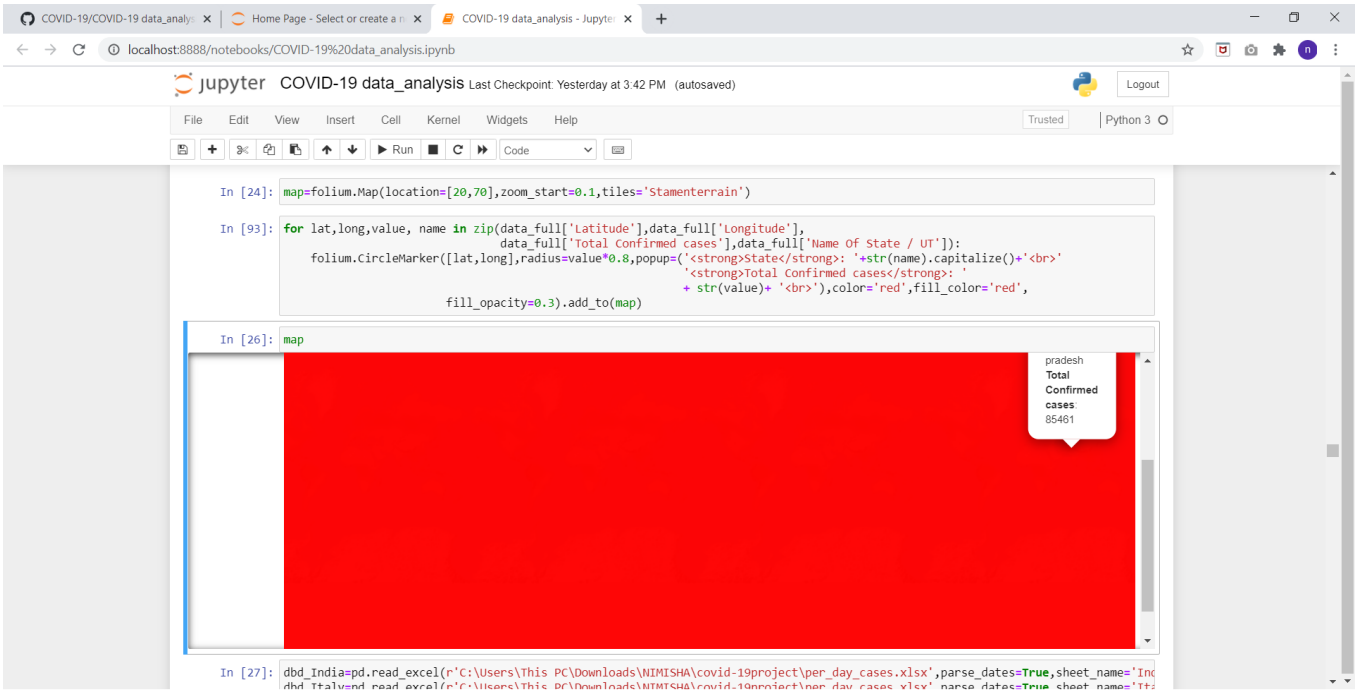
```
In [22]: data_full=pd.merge(Indian_Cord,data,on ='Name Of State / UT')

In [23]: data_full
```

Out[23]:

	Name Of State / UT	Latitude	Longitude	Total Confirmed cases	Cured	Death	Active cases
0	Andhra Pradesh	14.750429	78.570026	140933	63864	1349	75720
1	Delhi	28.669993	77.230004	135598	120930	3963	10705
2	Haryana	28.450006	77.019991	34965	28227	421	6317
3	Karnataka	12.570381	76.919997	124115	49788	2314	72013
4	Kerala	8.900373	76.569993	23614	13023	74	10517
5	Maharashtra	19.250232	73.160175	422118	256158	14994	150966
6	Punjab	31.519974	75.980003	16119	10734	386	4999
7	Rajasthan	26.449999	74.639981	42083	29845	680	11558
8	Tamil Nadu	12.920386	79.150042	245859	183956	3935	57968
9	Uttar Pradesh	27.599981	78.050006	85461	48863	1630	34968

```
In [24]: map=folium.Map(location=[20,70],zoom_start=0.1,tiles='Stamenterrain')

In [25]: for lat,long,value, name in zip(data_full['Latitude'],data_full['Longitude'],data_full['Total Confirmed cases'],data_full['Name
             folium.CircleMarker([lat,long],radius=value*0.8,popup=('<strong>State</strong>: '+str(name).capitalize()+'<br>''<strong>Total

In [26]: map
```

Global Map representation of the collected data.

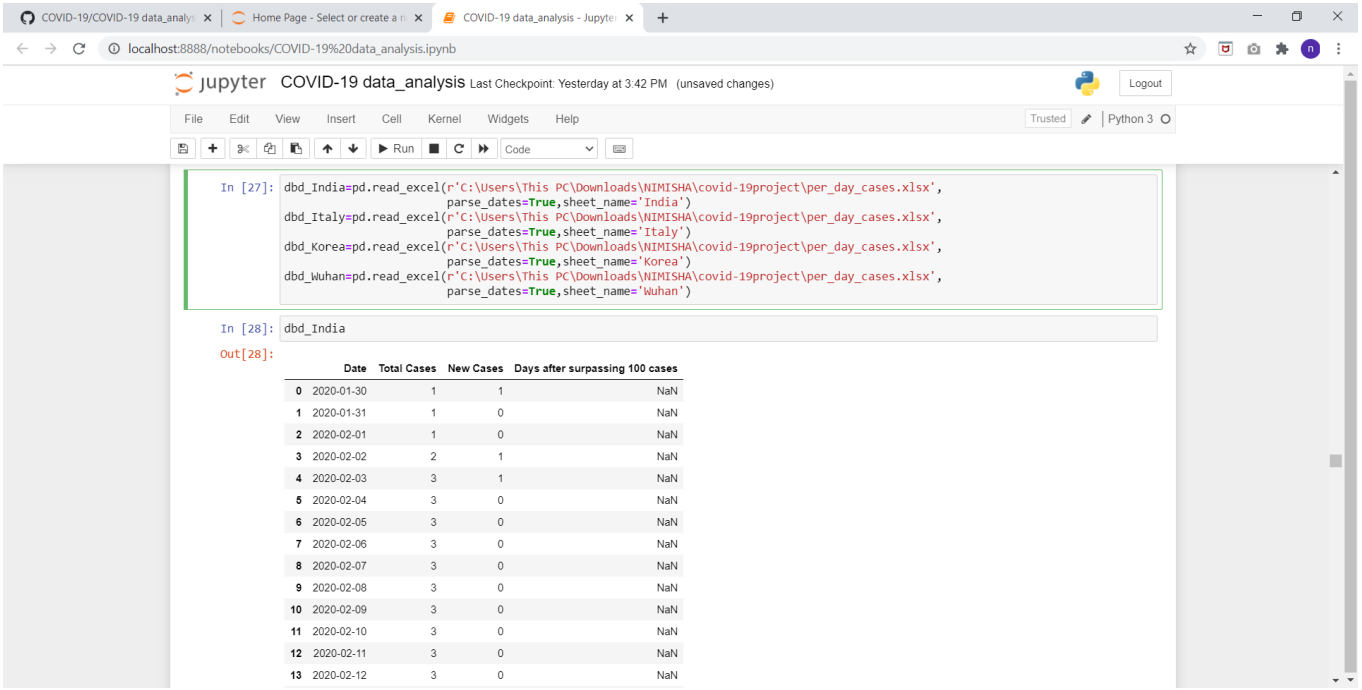


```
In [24]: map=folium.Map(location=[20,70],zoom_start=0.1,tiles='Stamenterrain')

In [93]: for lat,long,value, name in zip(data_full['Latitude'],data_full['Longitude'],
                                         data_full['Total Confirmed cases'],data_full['Name Of State / UT']):
             folium.CircleMarker([lat,long],radius=value*0.8,popup=('<strong>State</strong>: '+str(name).capitalize()+'<br>'
                                                                    '<strong>Total Confirmed cases</strong>: '
                                                                    + str(value)+ '<br>'),color='red',fill_color='red',
                                 fill_opacity=0.3).add_to(map)

In [26]: map
```

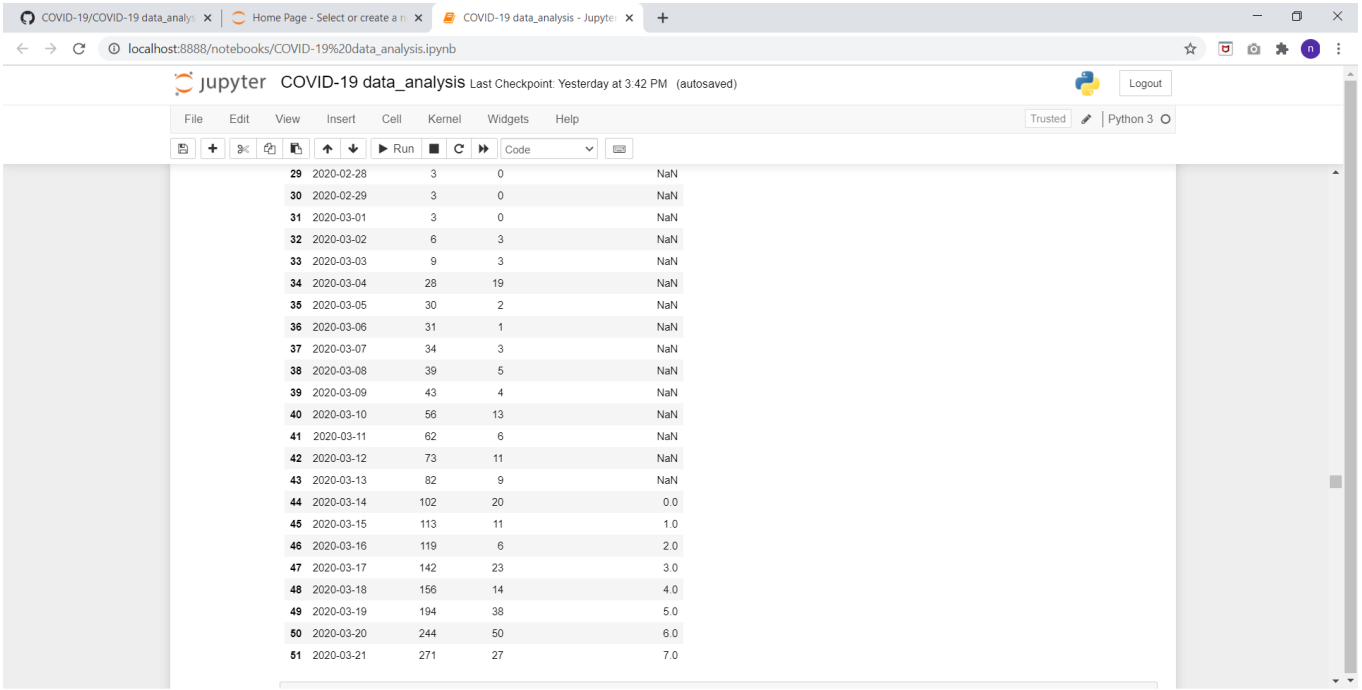
pradesh
Total Confirmed cases
85461

```
In [27]: dbd_India=pd.read_excel(r'C:\Users\This PC\Downloads\NIMISHA\covid-19project\per_day_cases.xlsx',parse_dates=True,sheet_name='Inc
```

Taking up the data of India,Italy,Korea,Wuhan.

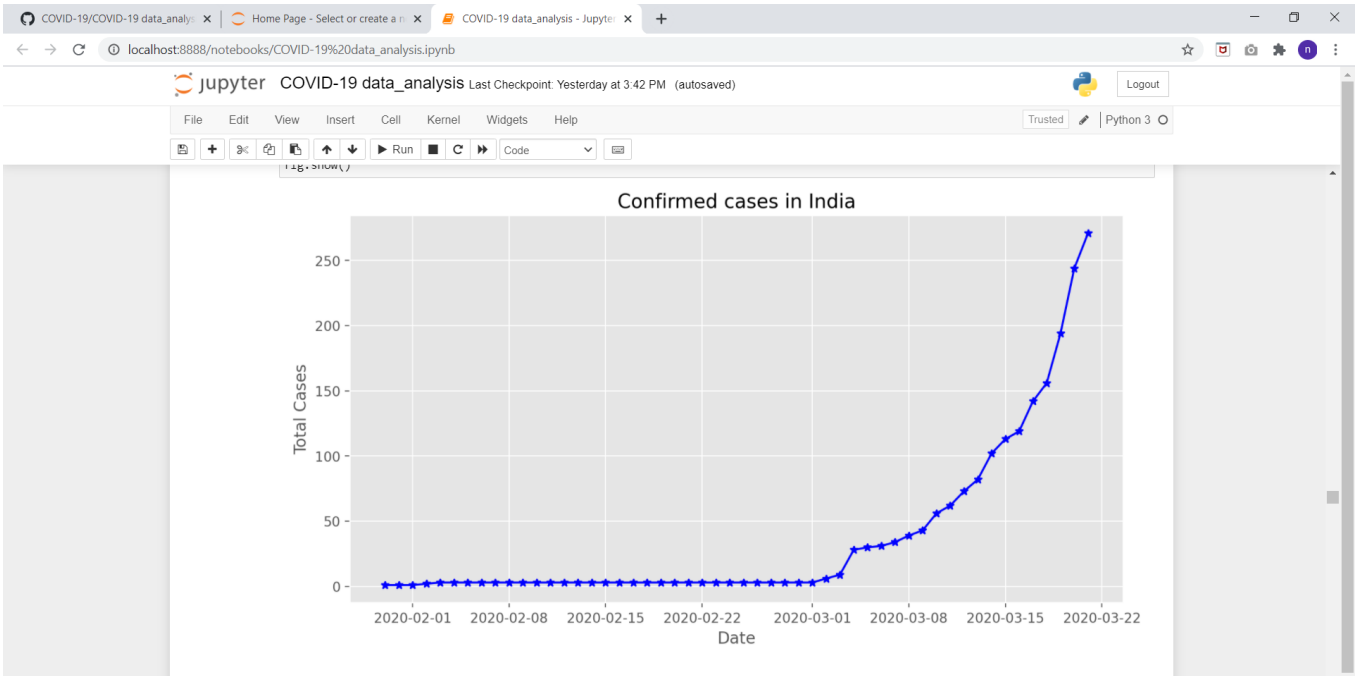


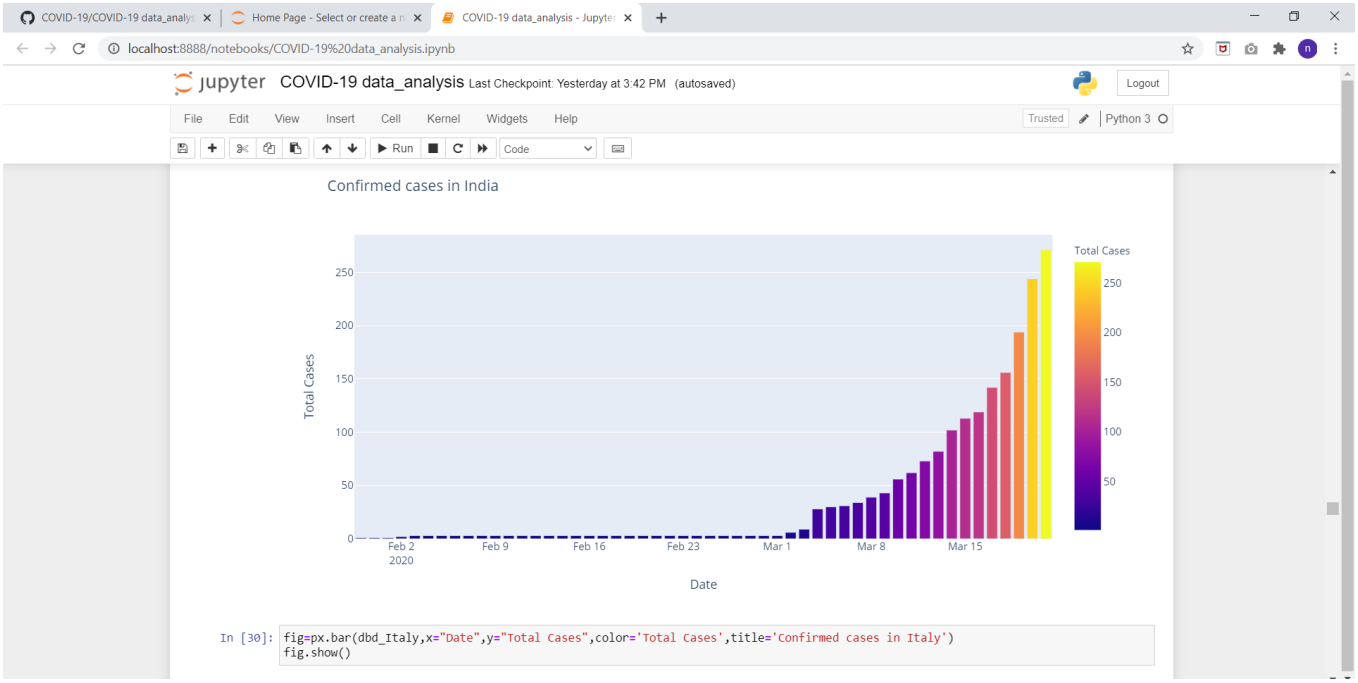
	Date	Total Cases	New Cases	Days after surpassing 100 cases
0	2020-01-30	1	1	NaN
1	2020-01-31	1	0	NaN
2	2020-02-01	1	0	NaN
3	2020-02-02	2	1	NaN
4	2020-02-03	3	1	NaN
5	2020-02-04	3	0	NaN
6	2020-02-05	3	0	NaN
7	2020-02-06	3	0	NaN
8	2020-02-07	3	0	NaN
9	2020-02-08	3	0	NaN
10	2020-02-09	3	0	NaN
11	2020-02-10	3	0	NaN
12	2020-02-11	3	0	NaN
13	2020-02-12	3	0	NaN



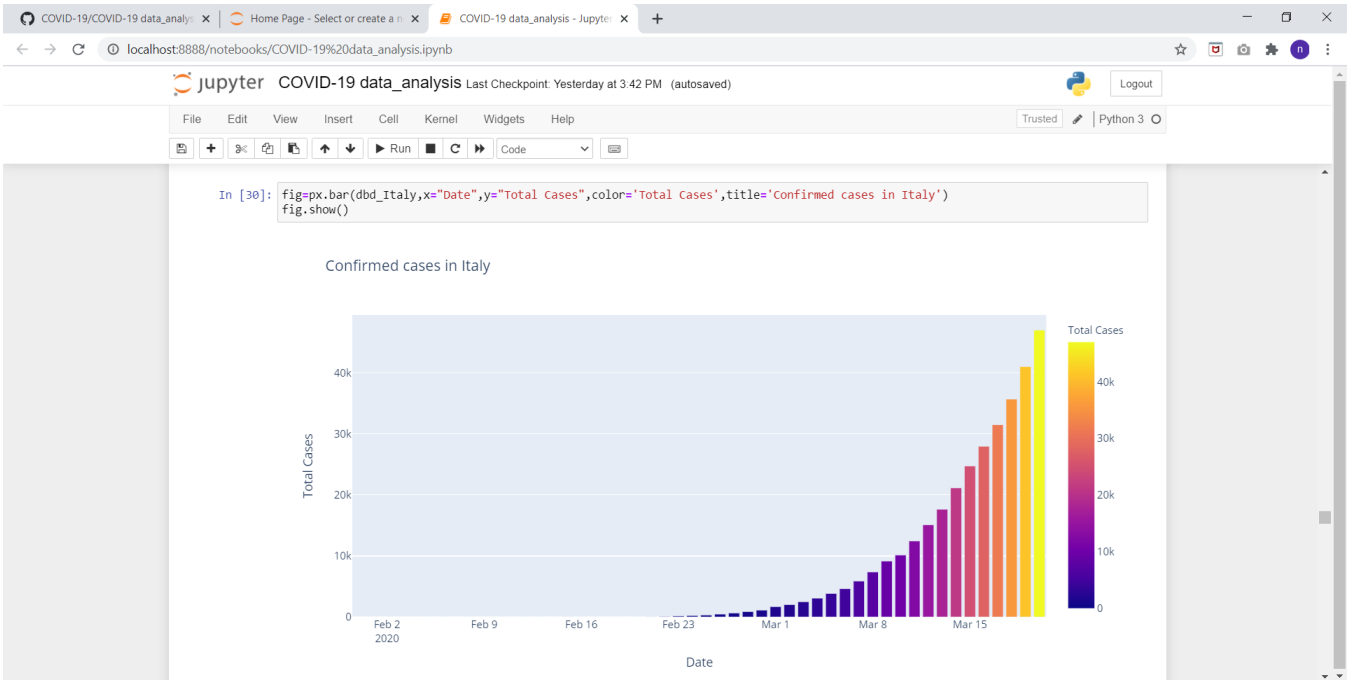
29	2020-02-28	3	0	NaN
30	2020-02-29	3	0	NaN
31	2020-03-01	3	0	NaN
32	2020-03-02	6	3	NaN
33	2020-03-03	9	3	NaN
34	2020-03-04	28	19	NaN
35	2020-03-05	30	2	NaN
36	2020-03-06	31	1	NaN
37	2020-03-07	34	3	NaN
38	2020-03-08	39	5	NaN
39	2020-03-09	43	4	NaN
40	2020-03-10	56	13	NaN
41	2020-03-11	62	6	NaN
42	2020-03-12	73	11	NaN
43	2020-03-13	82	9	NaN
44	2020-03-14	102	20	0.0
45	2020-03-15	113	11	1.0
46	2020-03-16	119	6	2.0
47	2020-03-17	142	23	3.0
48	2020-03-18	156	14	4.0
49	2020-03-19	194	38	5.0
50	2020-03-20	244	50	6.0
51	2020-03-21	271	27	7.0

Confirmed cases in India.

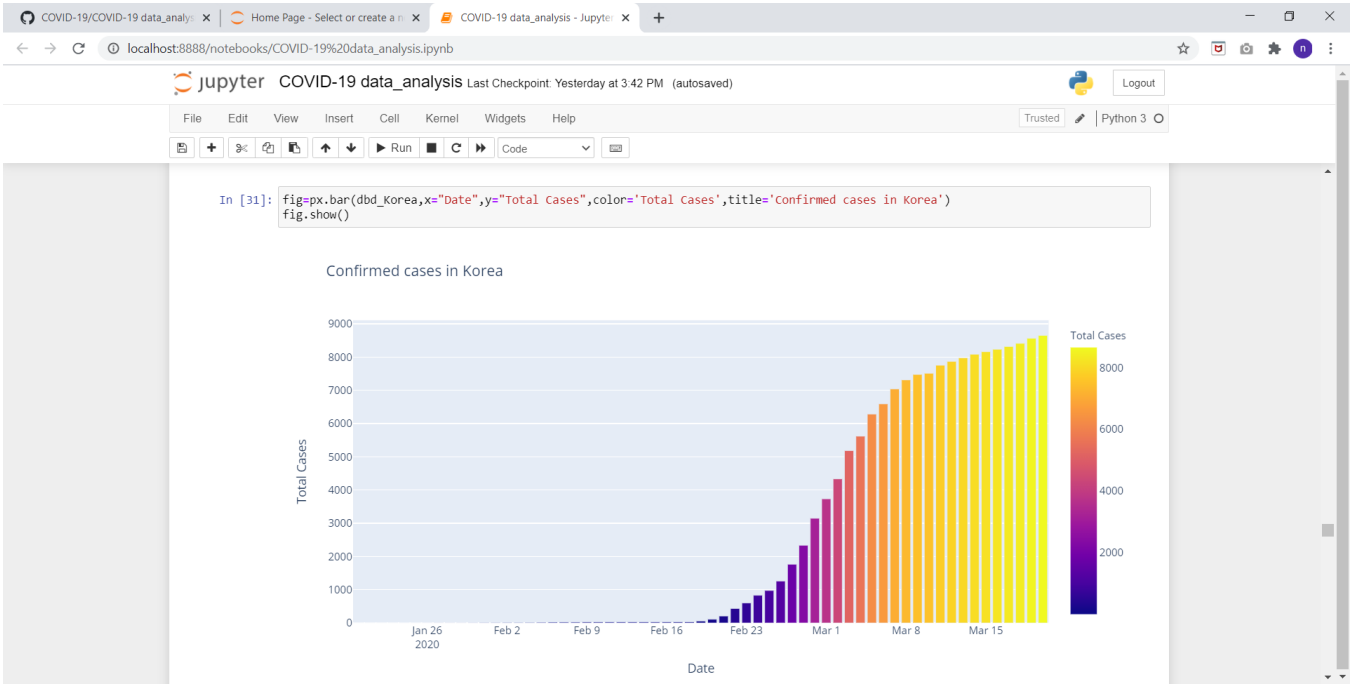




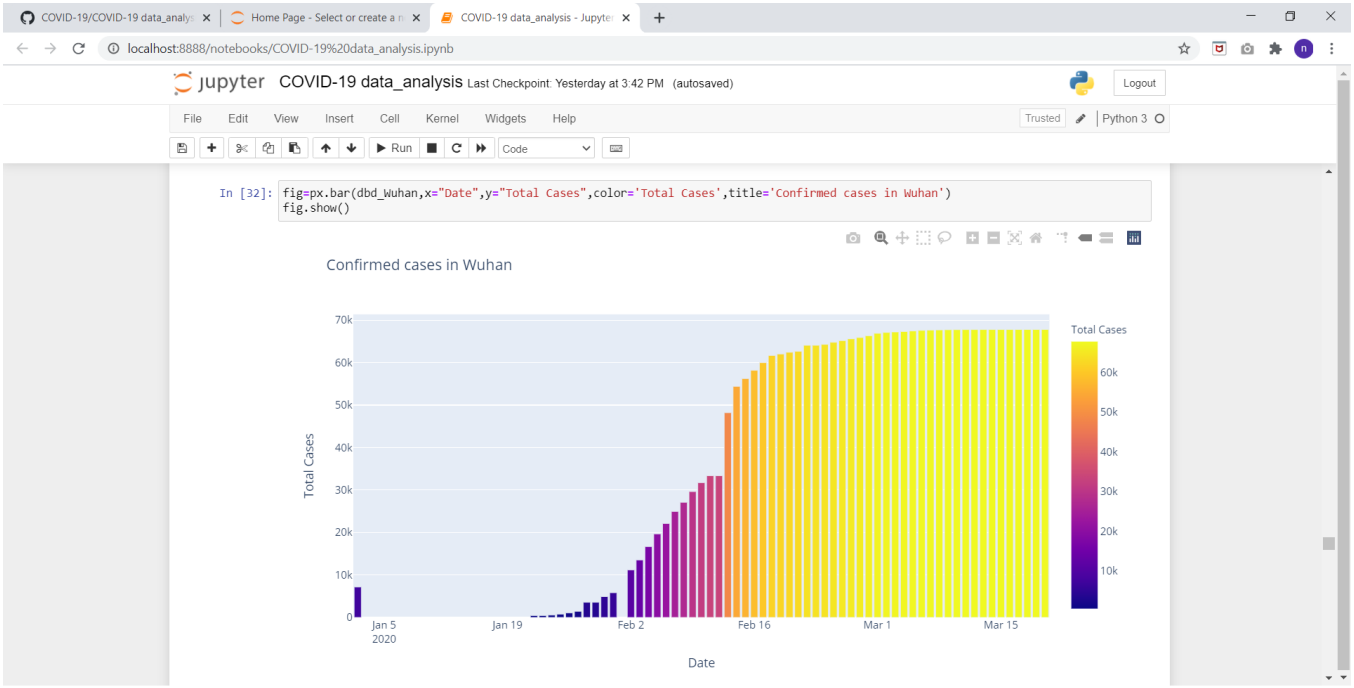
Confirmed cases in Italy.

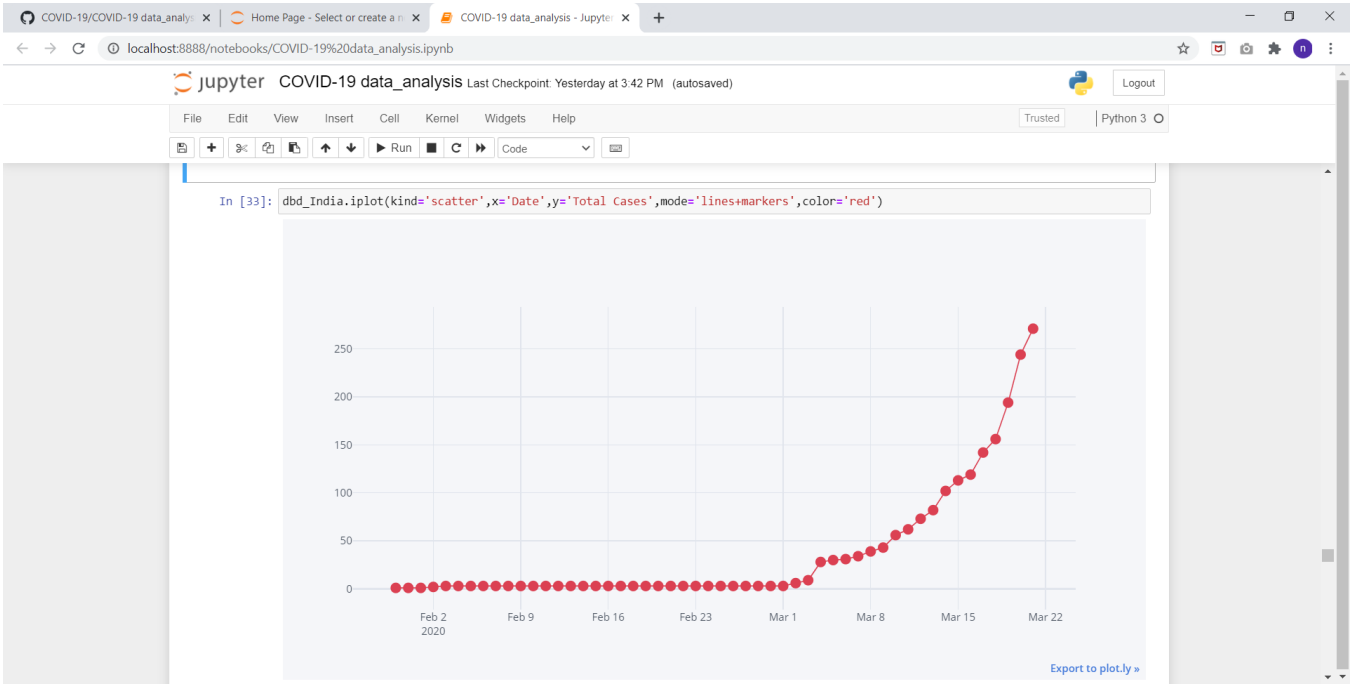


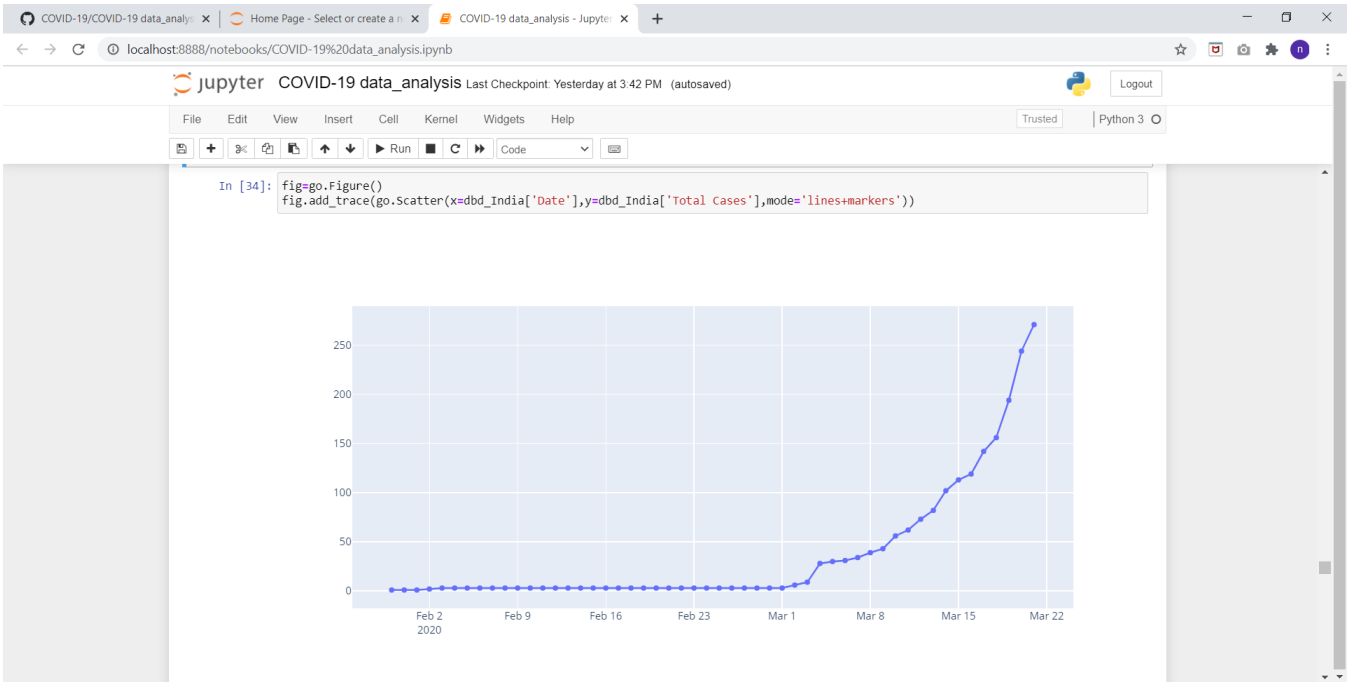
Confirmed Cases in Korea.



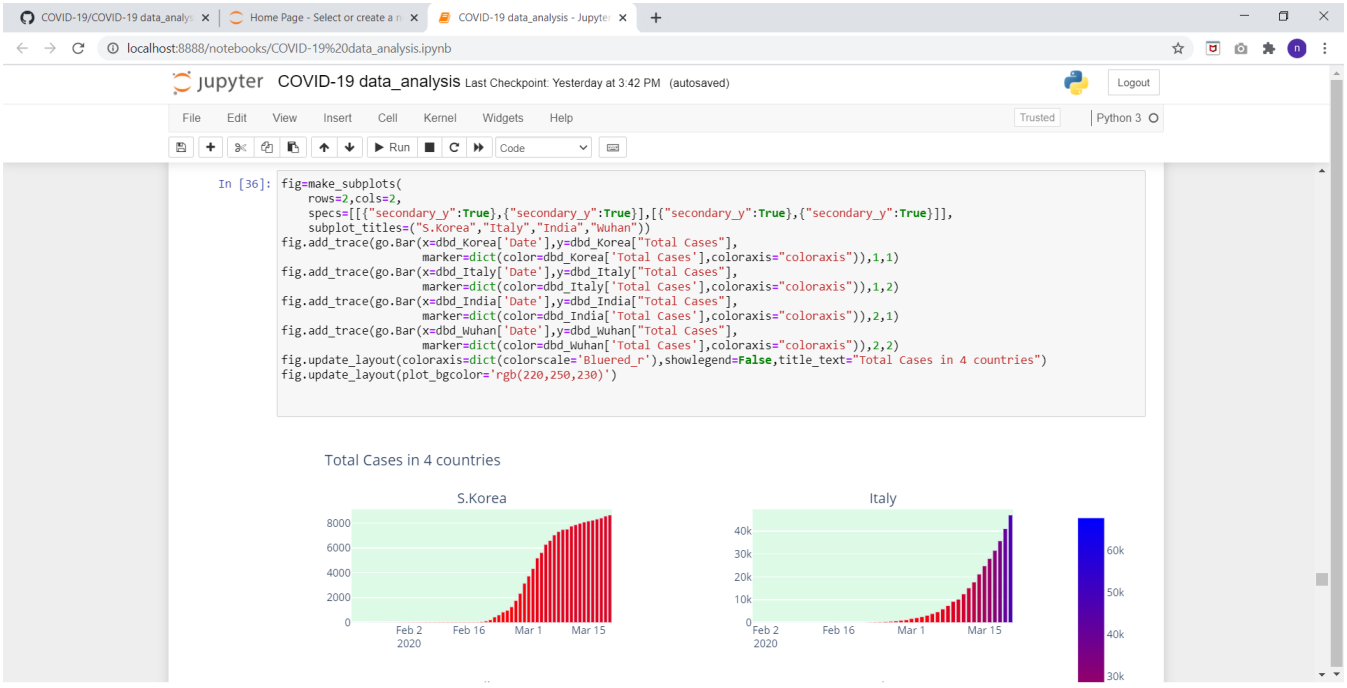
Confirmed Cases in Wuhan.

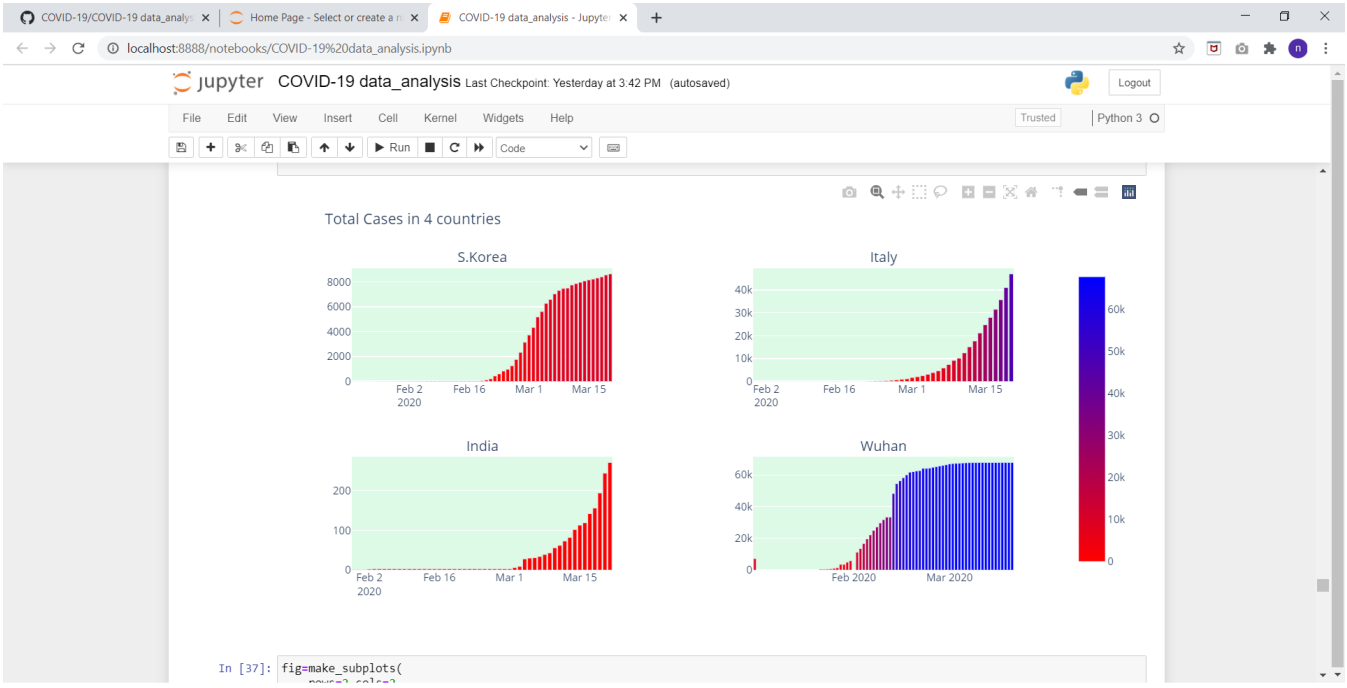




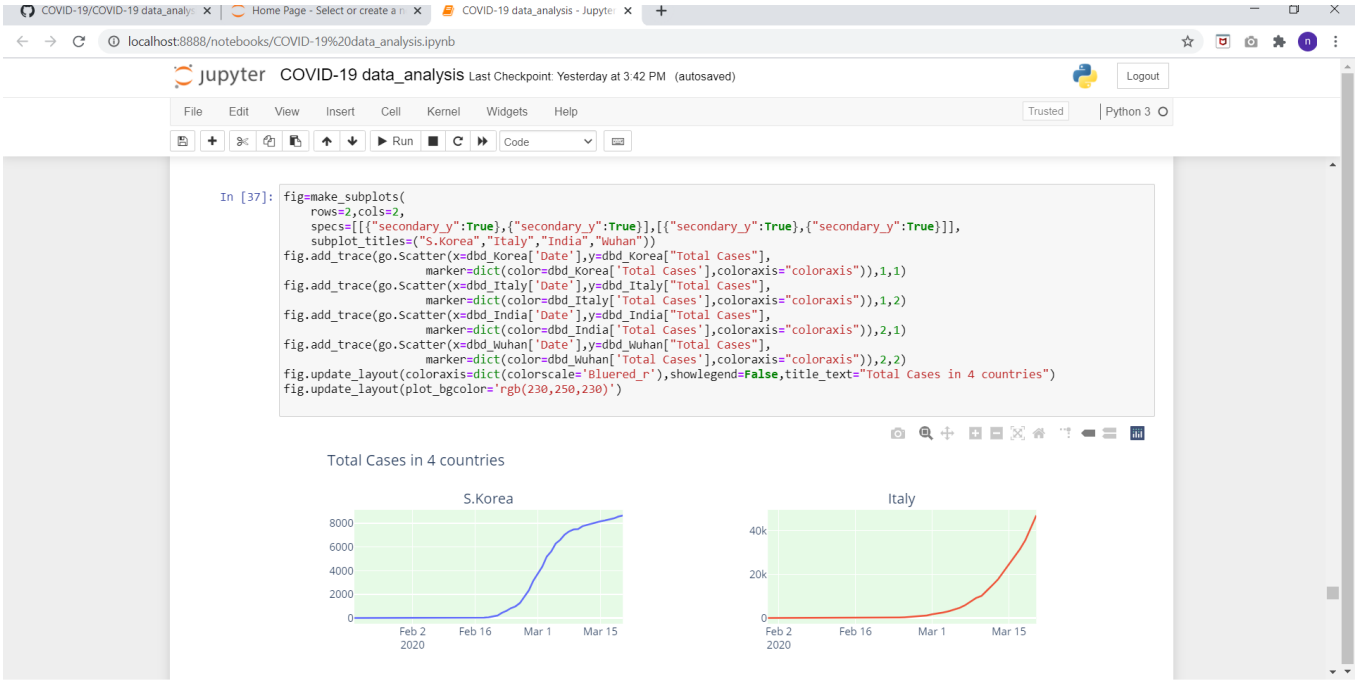


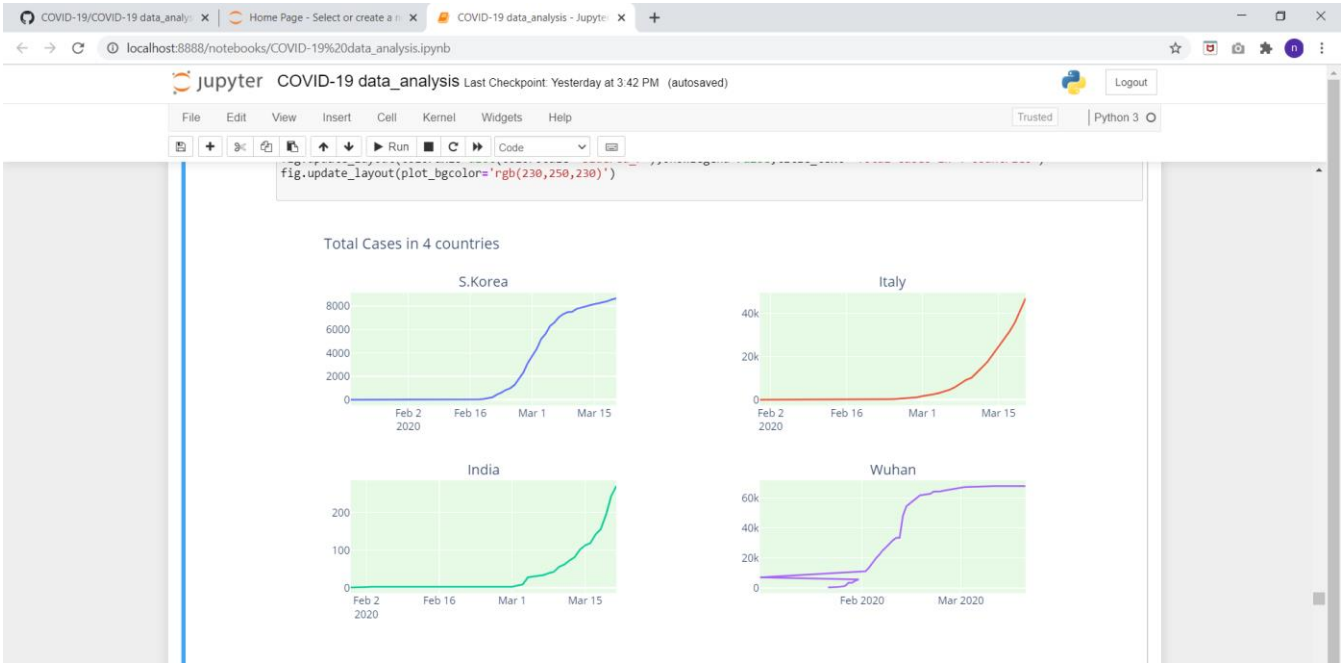
Comparing the cases between India,Wuhan,Italy Korea, using data set using bar graph.

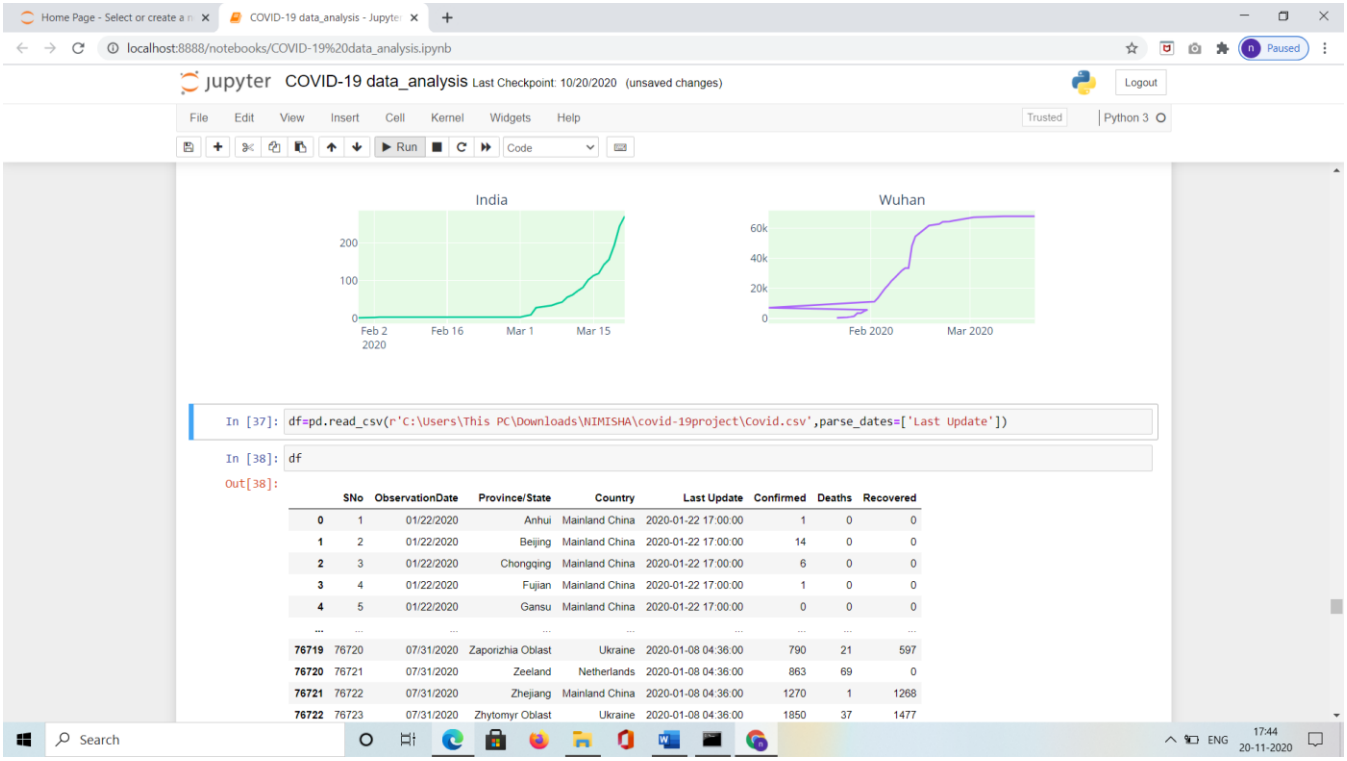


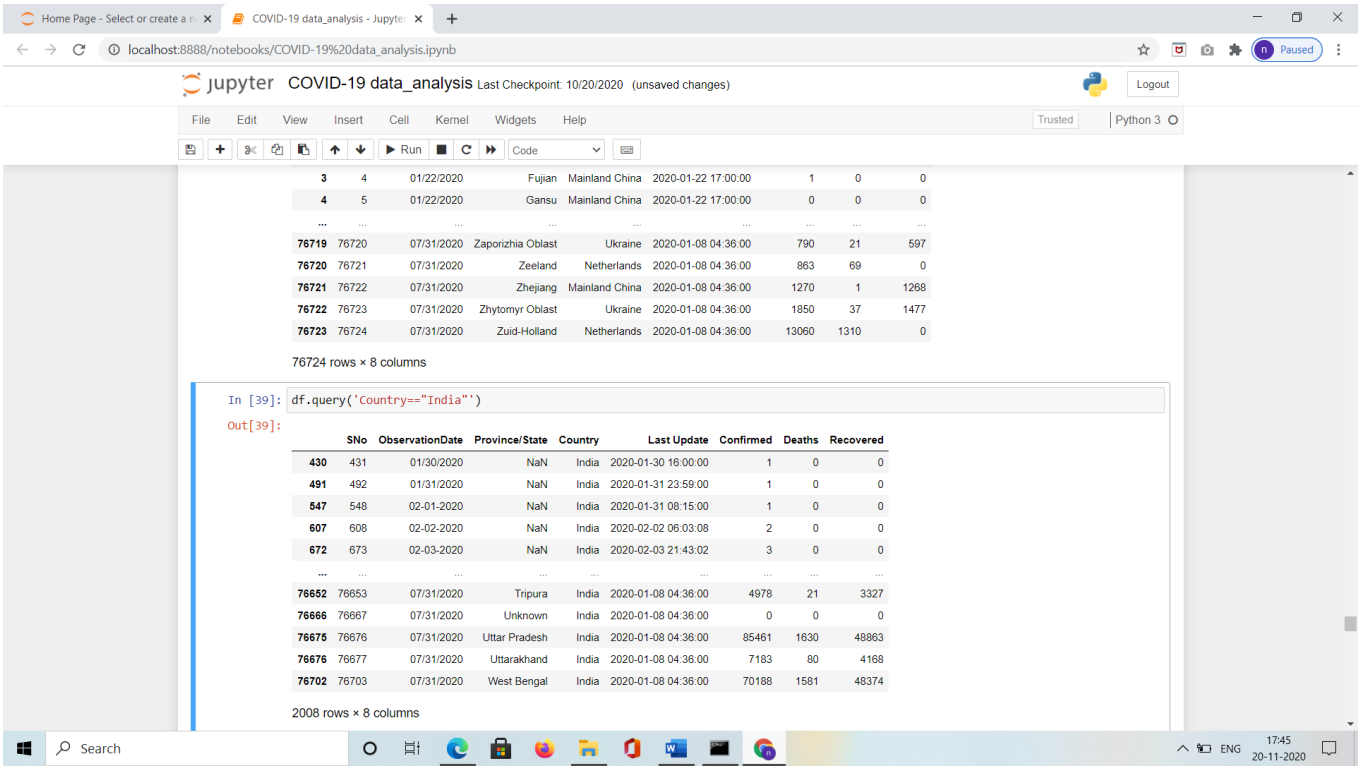


Comparing the cases between India,Wuhan,Italy Korea, using data set using iplot.

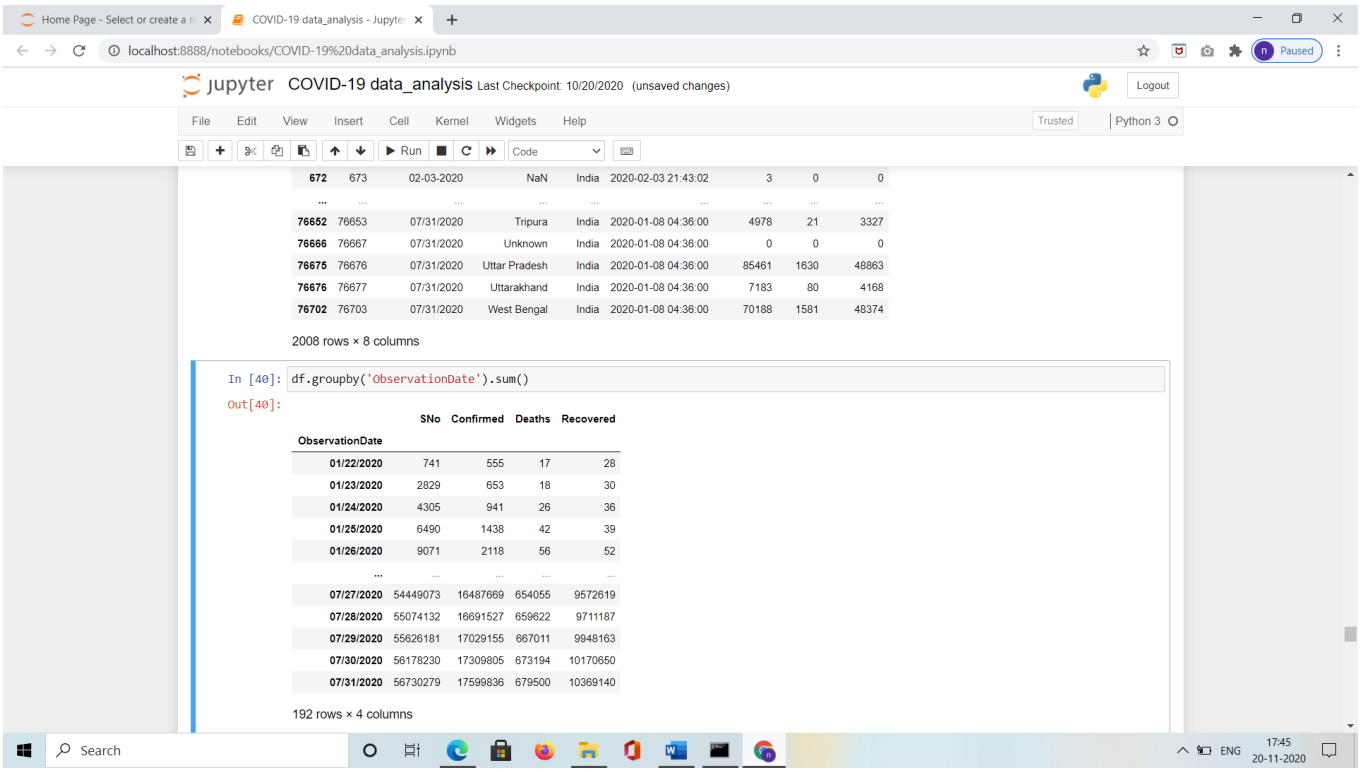

localhost:8888/notebooks/COVID-19%20data_analysis.ipynb
jupyter COVID-19 data_analysis Last Checkpoint: Yesterday at 3:42 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [37]:
fig=make_subplots(
rows=2,cols=2,
specs=[[{"secondary_y":True},{"secondary_y":True}],[{"secondary_y":True},{"secondary_y":True}]],
subplot_titles=("S.Korea","Italy","India","Wuhan"))
fig.add_trace(go.Scatter(x=dbd_Korea['Date'],y=dbd_Korea["Total Cases"],
marker=dict(color=dbd_Korea['Total Cases'],coloraxis="coloraxis")),1,1)
fig.add_trace(go.Scatter(x=dbd_Italy['Date'],y=dbd_Italy["Total Cases"],
marker=dict(color=dbd_Italy['Total Cases'],coloraxis="coloraxis")),1,2)
fig.add_trace(go.Scatter(x=dbd_India['Date'],y=dbd_India["Total Cases"],
marker=dict(color=dbd_India['Total Cases'],coloraxis="coloraxis")),2,1)
fig.add_trace(go.Scatter(x=dbd_Wuhan['Date'],y=dbd_Wuhan["Total Cases"],
marker=dict(color=dbd_Wuhan['Total Cases'],coloraxis="coloraxis")),2,2)
fig.update_layout(coloraxis=dict(colorscale='Bluered_r'),showlegend=False,title_text="Total Cases in 4 countries")
fig.update_layout(plot_bgcolor='rgb(230,250,230)')
Total Cases in 4 countries
S.Korea
Italy
8000
6000
4000
2000
0
40k
20k
Feb 2
2020
Feb 16
Mar 1
Mar 15


localhost:8888/notebooks/COVID-19%20data_analysis.ipynb
jupyter COVID-19 data_analysis Last Checkpoint: Yesterday at 3:42 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
fig.update_layout(plot_bgcolor='rgb(230,250,230)')
Total Cases in 4 countries
S.Korea
Italy
India
Wuhan
8000
6000
4000
2000
0
40k
20k
200
100
60k
Feb 2
2020
Feb 16
Mar 1
Mar 15
Feb 2020
Mar 2020


Home Page - Select or create a n
COVID-19 data_analysis - Jupyter
localhost:8888/notebooks/COVID-19%20data_analysis.ipynb
jupyter COVID-19 data_analysis Last Checkpoint: 10/20/2020 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
India
200
100
0
Feb 2 2020
Feb 16
Mar 1
Mar 15
Wuhan
60k
40k
20k
0
Feb 2020
Mar 2020
In [37]: df=pd.read_csv(r'C:\Users\This PC\Downloads\NIMISHA\covid-19project\Covid.csv',parse_dates=['Last Update'])
In [38]: df
Out[38]:
SNo ObservationDate Province/State Country Last Update Confirmed Deaths Recovered
0 1 01/22/2020 Anhui Mainland China 2020-01-22 17:00:00 1 0 0
1 2 01/22/2020 Beijing Mainland China 2020-01-22 17:00:00 14 0 0
2 3 01/22/2020 Chongqing Mainland China 2020-01-22 17:00:00 6 0 0
3 4 01/22/2020 Fujian Mainland China 2020-01-22 17:00:00 1 0 0
4 5 01/22/2020 Gansu Mainland China 2020-01-22 17:00:00 0 0 0
76719 76720 07/31/2020 Zaporizhia Oblast Ukraine 2020-01-08 04:36:00 790 21 597
76720 76721 07/31/2020 Zeeland Netherlands 2020-01-08 04:36:00 863 69 0
76721 76722 07/31/2020 Zhejiang Mainland China 2020-01-08 04:36:00 1270 1 1268
76722 76723 07/31/2020 Zhytomyr Oblast Ukraine 2020-01-08 04:36:00 1850 37 1477
Search
ENG
17:44
20-11-2020


Home Page - Select or create a n
COVID-19 data_analysis - Jupyter
localhost:8888/notebooks/COVID-19%20data_analysis.ipynb
jupyter COVID-19 data_analysis Last Checkpoint: 10/20/2020 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
3 4 01/22/2020 Fujian Mainland China 2020-01-22 17:00:00 1 0 0
4 5 01/22/2020 Gansu Mainland China 2020-01-22 17:00:00 0 0 0
76719 76720 07/31/2020 Zaporizhia Oblast Ukraine 2020-01-08 04:36:00 790 21 597
76720 76721 07/31/2020 Zeeland Netherlands 2020-01-08 04:36:00 863 69 0
76721 76722 07/31/2020 Zhejiang Mainland China 2020-01-08 04:36:00 1270 1 1268
76722 76723 07/31/2020 Zhytomyr Oblast Ukraine 2020-01-08 04:36:00 1850 37 1477
76723 76724 07/31/2020 Zuid-Holland Netherlands 2020-01-08 04:36:00 13060 1310 0
76724 rows × 8 columns
In [39]: df.query('Country=="India"')
Out[39]:
SNo ObservationDate Province/State Country Last Update Confirmed Deaths Recovered
430 431 01/30/2020 NaN India 2020-01-30 16:00:00 1 0 0
491 492 01/31/2020 NaN India 2020-01-31 23:59:00 1 0 0
547 548 02-01-2020 NaN India 2020-01-31 08:15:00 1 0 0
607 608 02-02-2020 NaN India 2020-02-02 06:03:08 2 0 0
672 673 02-03-2020 NaN India 2020-02-03 21:43:02 3 0 0
76652 76653 07/31/2020 Tripura India 2020-01-08 04:36:00 4978 21 3327
76666 76667 07/31/2020 Unknown India 2020-01-08 04:36:00 0 0 0
76675 76676 07/31/2020 Uttar Pradesh India 2020-01-08 04:36:00 85461 1630 48863
76676 76677 07/31/2020 Uttarakhand India 2020-01-08 04:36:00 7183 80 4168
76702 76703 07/31/2020 West Bengal India 2020-01-08 04:36:00 70188 1581 48374
2008 rows × 8 columns
Search
ENG
17:45
20-11-2020

Sorting the data as per the need on the bases of deaths, recoveries, confirmed cases.



672	673	02-03-2020	NaN	India	2020-02-03 21:43:02	3	0	0
...	...	...	...	...	...	...	...	...
76652	76653	07/31/2020	Tripura	India	2020-01-08 04:36:00	4978	21	3327
76666	76667	07/31/2020	Unknown	India	2020-01-08 04:36:00	0	0	0
76675	76676	07/31/2020	Uttar Pradesh	India	2020-01-08 04:36:00	85461	1630	48863
76676	76677	07/31/2020	Uttarakhand	India	2020-01-08 04:36:00	7183	80	4168
76702	76703	07/31/2020	West Bengal	India	2020-01-08 04:36:00	70188	1581	48374

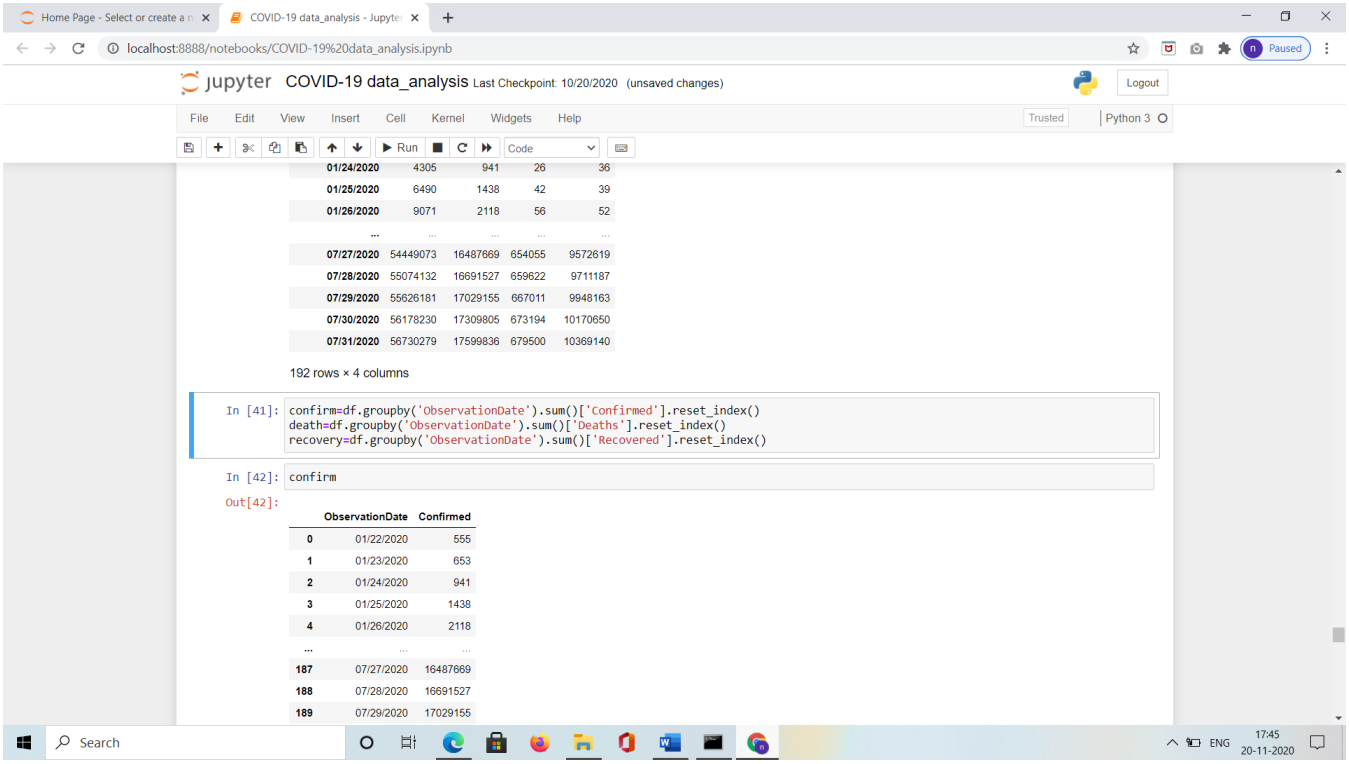
2008 rows × 8 columns

```
In [40]: df.groupby('ObservationDate').sum()
```

Out[40]:

ObservationDate	SNo	Confirmed	Deaths	Recovered
01/22/2020	741	555	17	28
01/23/2020	2829	653	18	30
01/24/2020	4305	941	26	36
01/25/2020	6490	1438	42	39
01/26/2020	9071	2118	56	52
...	...	...	...	...
07/27/2020	54449073	16487669	654055	9572619
07/28/2020	55074132	16691527	659622	9711187
07/29/2020	55626181	17029155	667011	9948163
07/30/2020	56178230	17309805	673194	10170650
07/31/2020	56730279	17599836	679500	10369140

192 rows × 4 columns



01/24/2020	4305	941	26	36
01/25/2020	6490	1438	42	39
01/26/2020	9071	2118	56	52
...	...	...	...	...
07/27/2020	54449073	16487669	654055	9572619
07/28/2020	55074132	16691527	659622	9711187
07/29/2020	55626181	17029155	667011	9948163
07/30/2020	56178230	17309805	673194	10170650
07/31/2020	56730279	17599836	679500	10369140

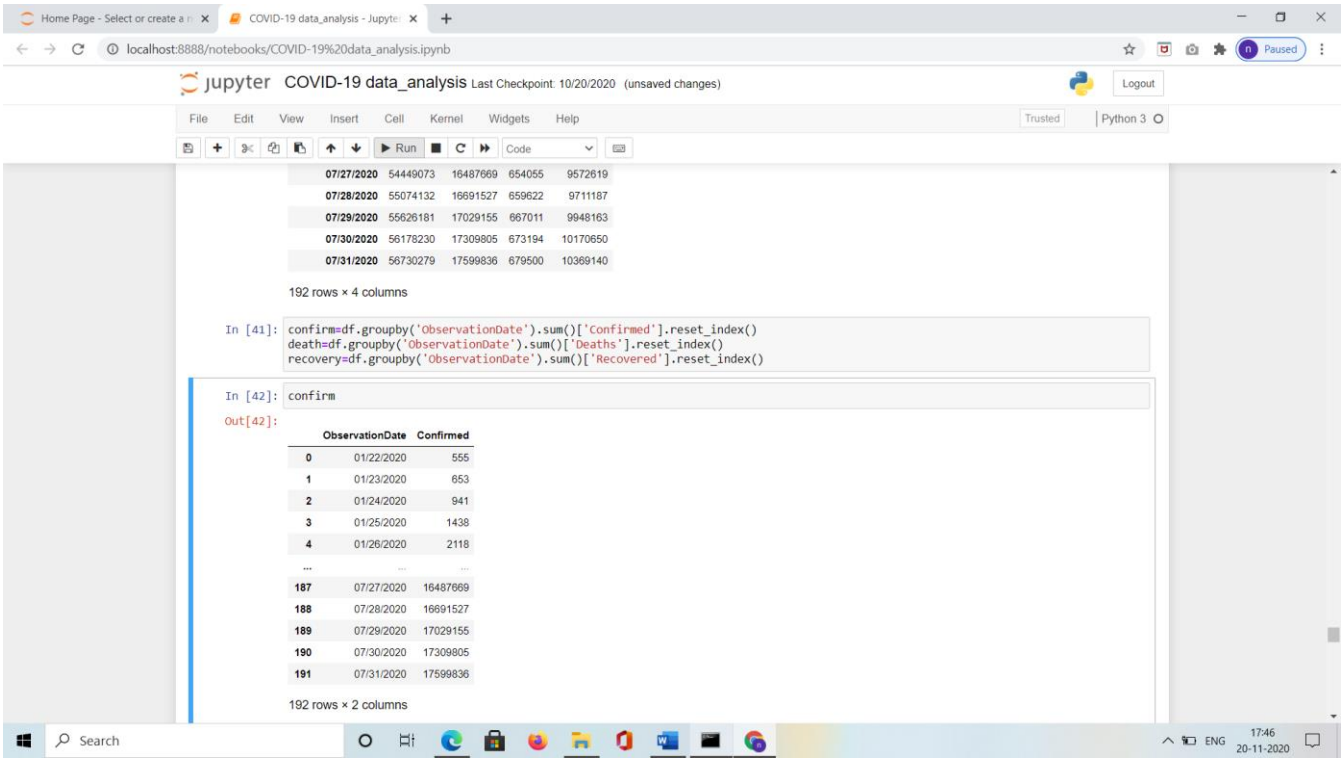
192 rows × 4 columns

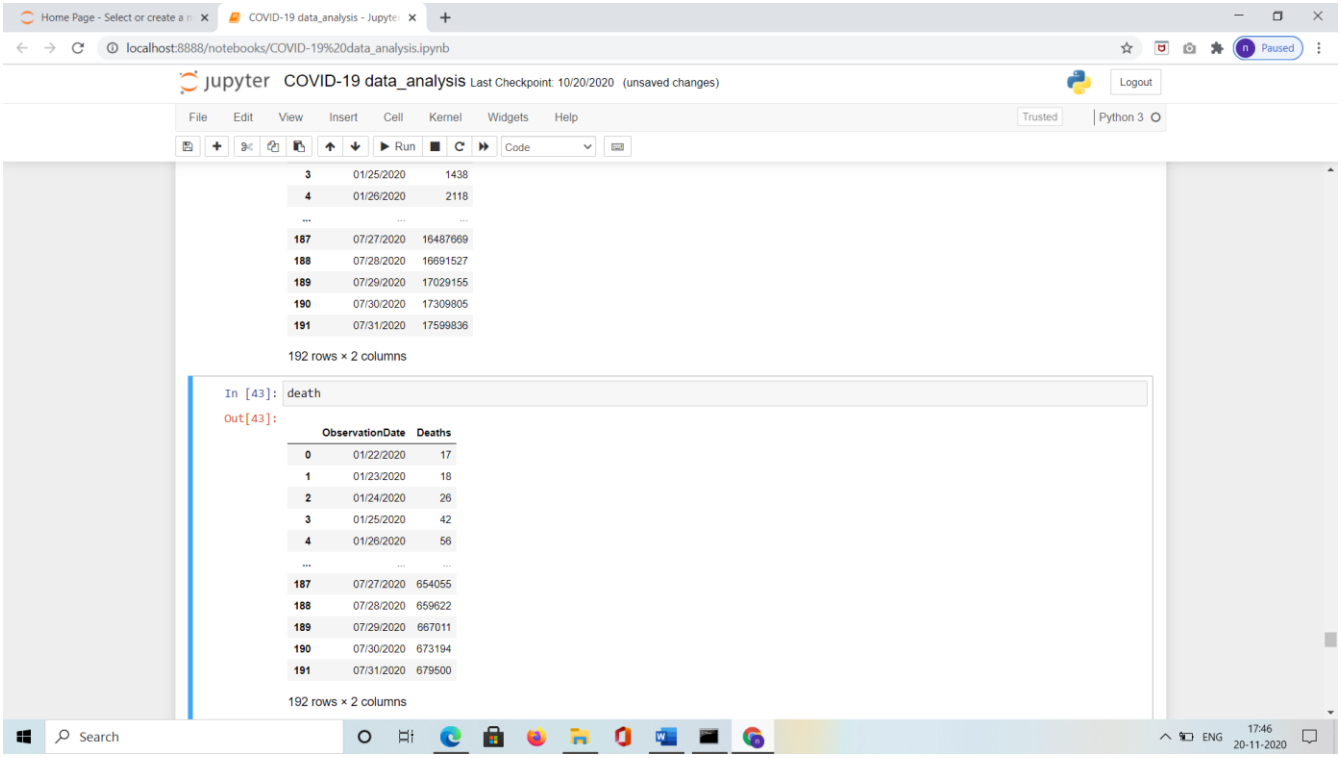
```
In [41]: confirm=df.groupby('ObservationDate').sum()['Confirmed'].reset_index()
         death=df.groupby('ObservationDate').sum()['Deaths'].reset_index()
         recovery=df.groupby('ObservationDate').sum()['Recovered'].reset_index()
```

```
In [42]: confirm
```

Out[42]:

	ObservationDate	Confirmed
0	01/22/2020	555
1	01/23/2020	653
2	01/24/2020	941
3	01/25/2020	1438
4	01/26/2020	2118
...	...	...
187	07/27/2020	16487669
188	07/28/2020	16691527
189	07/29/2020	17029155


Home Page - Select or create a n
COVID-19 data_analysis - Jupyte
localhost:8888/notebooks/COVID-19%20data_analysis.ipynb
Paused
jupyter COVID-19 data_analysis Last Checkpoint: 10/20/2020 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
07/27/2020 54449073 16487669 654055 9572619
07/28/2020 55074132 16691527 659622 9711187
07/29/2020 55626181 17029155 667011 9948163
07/30/2020 56178230 17309805 673194 10170650
07/31/2020 56730279 17599836 679500 10369140
192 rows × 4 columns
In [41]:
confirm=df.groupby('ObservationDate').sum()['Confirmed'].reset_index()
death=df.groupby('ObservationDate').sum()['Deaths'].reset_index()
recovery=df.groupby('ObservationDate').sum()['Recovered'].reset_index()
In [42]:
confirm
Out[42]:
ObservationDate Confirmed
0 01/22/2020 555
1 01/23/2020 653
2 01/24/2020 941
3 01/25/2020 1438
4 01/26/2020 2118
187 07/27/2020 16487669
188 07/28/2020 16691527
189 07/29/2020 17029155
190 07/30/2020 17309805
191 07/31/2020 17599836
192 rows × 2 columns
Search
ENG
17:46
20-11-2020



3	01/25/2020	1438
4	01/26/2020	2118
...	...	...
187	07/27/2020	16487669
188	07/28/2020	16691527
189	07/29/2020	17029155
190	07/30/2020	17309805
191	07/31/2020	17599836

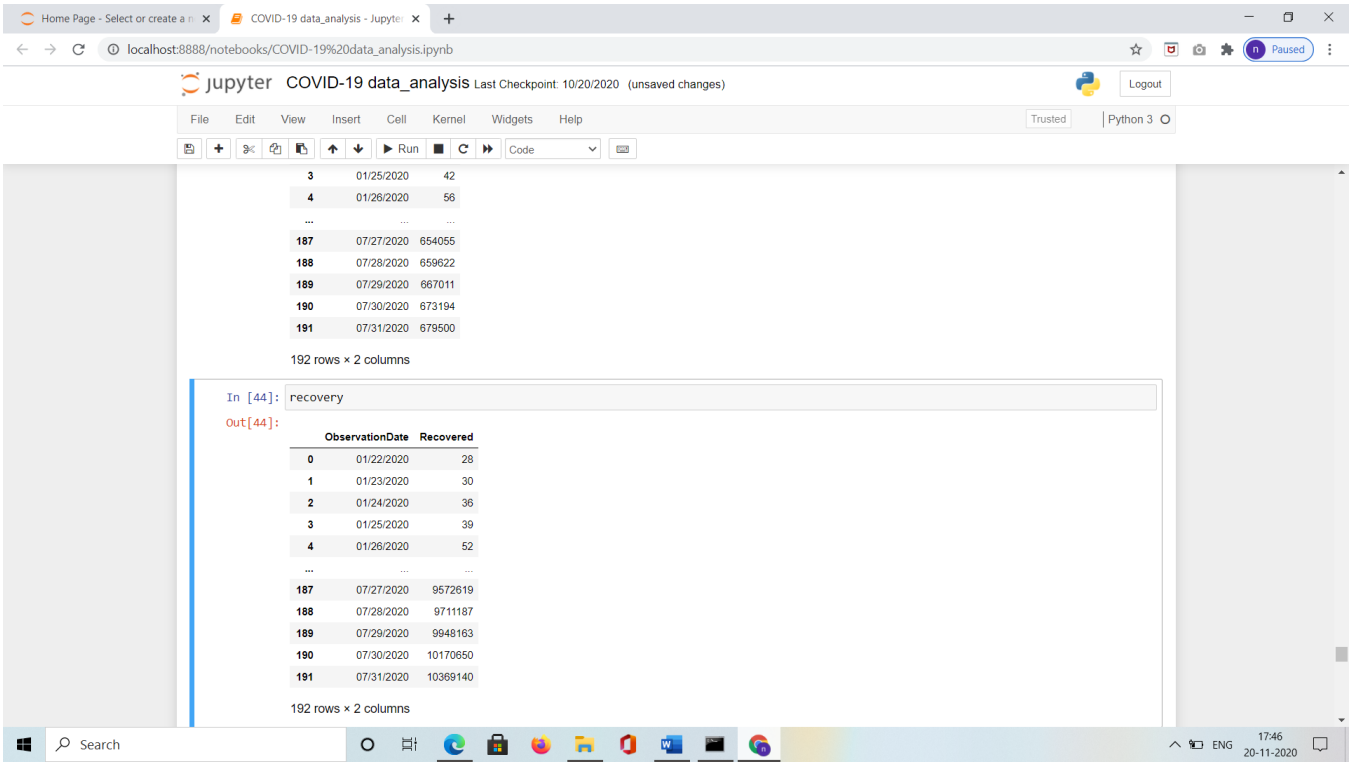
192 rows × 2 columns

```
In [43]: death
```

Out[43]:

	ObservationDate	Deaths
0	01/22/2020	17
1	01/23/2020	18
2	01/24/2020	26
3	01/25/2020	42
4	01/26/2020	56
...	...	...
187	07/27/2020	654055
188	07/28/2020	659622
189	07/29/2020	667011
190	07/30/2020	673194
191	07/31/2020	679500

192 rows × 2 columns



3	01/25/2020	42
4	01/26/2020	56
...	...	...
187	07/27/2020	654055
188	07/28/2020	659622
189	07/29/2020	667011
190	07/30/2020	673194
191	07/31/2020	679500

192 rows × 2 columns

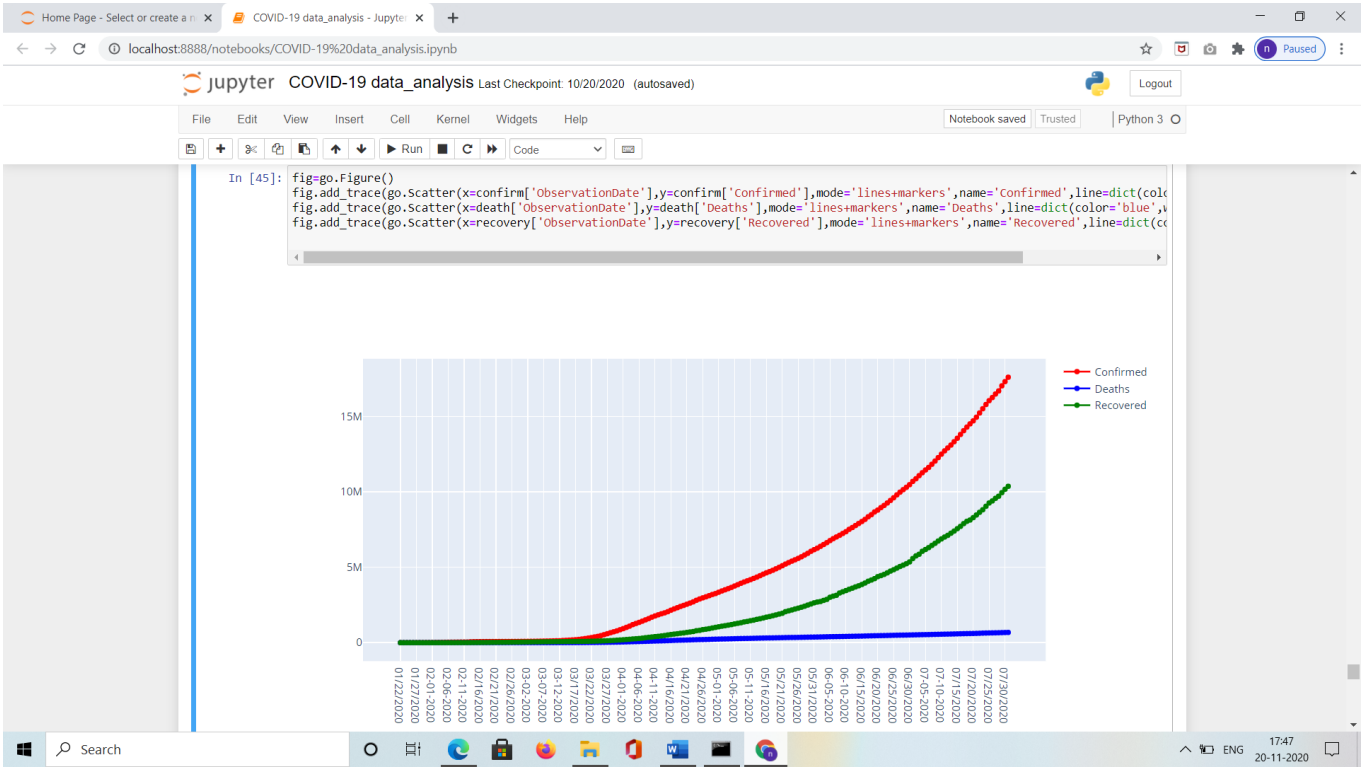
```
In [44]: recovery
```

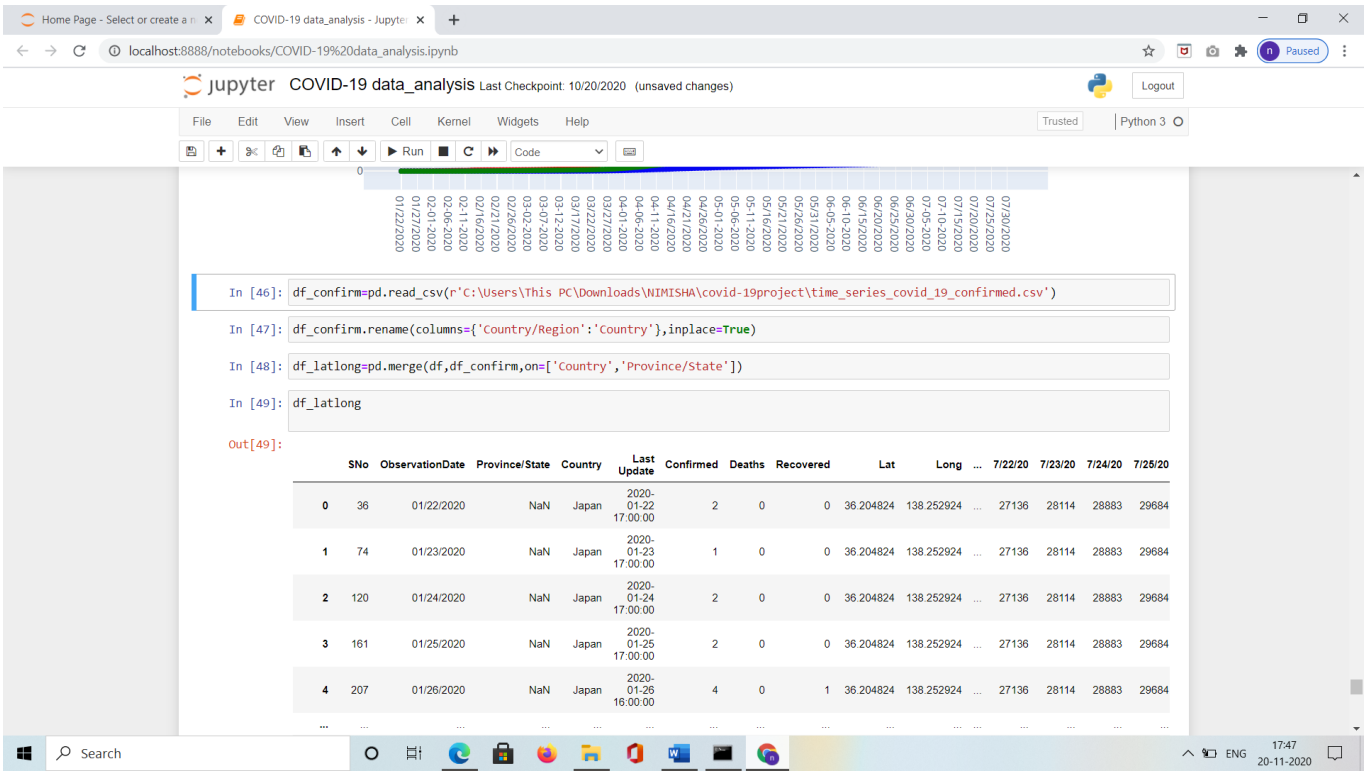
Out[44]:

	ObservationDate	Recovered
0	01/22/2020	28
1	01/23/2020	30
2	01/24/2020	36
3	01/25/2020	39
4	01/26/2020	52
...	...	...
187	07/27/2020	9572619
188	07/28/2020	9711187
189	07/29/2020	9948163
190	07/30/2020	10170650
191	07/31/2020	10369140

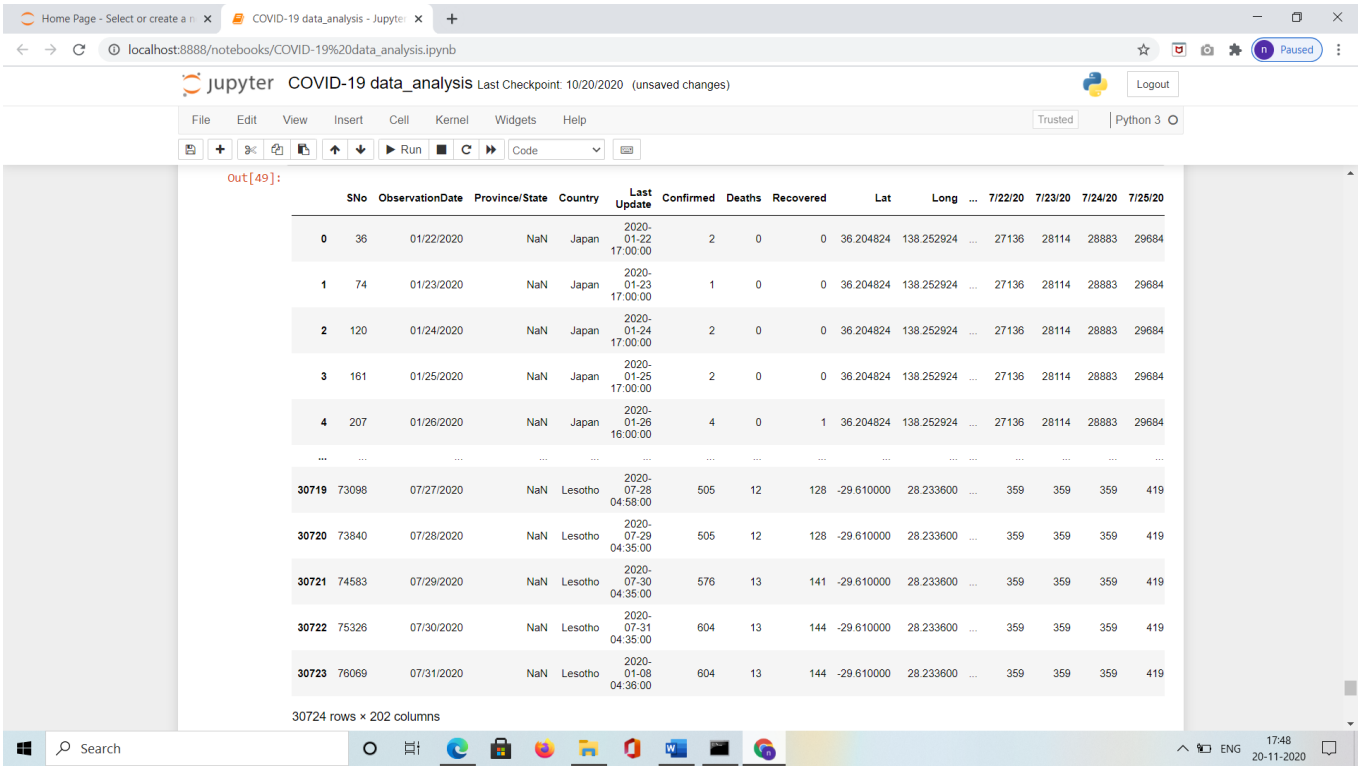
192 rows × 2 columns

Confirmed, Deaths, Recovered cases in a single view on same graph.





	SNo	ObservationDate	Province/State	Country	Last Update	Confirmed	Deaths	Recovered	Lat	Long	...	7/22/20	7/23/20	7/24/20	7/25/20
0	36	01/22/2020	NaN	Japan	2020-01-22 17:00:00	2	0	0	36.204824	138.252924	...	27136	28114	28883	29684
1	74	01/23/2020	NaN	Japan	2020-01-23 17:00:00	1	0	0	36.204824	138.252924	...	27136	28114	28883	29684
2	120	01/24/2020	NaN	Japan	2020-01-24 17:00:00	2	0	0	36.204824	138.252924	...	27136	28114	28883	29684
3	161	01/25/2020	NaN	Japan	2020-01-25 17:00:00	2	0	0	36.204824	138.252924	...	27136	28114	28883	29684
4	207	01/26/2020	NaN	Japan	2020-01-26 16:00:00	4	0	1	36.204824	138.252924	...	27136	28114	28883	29684

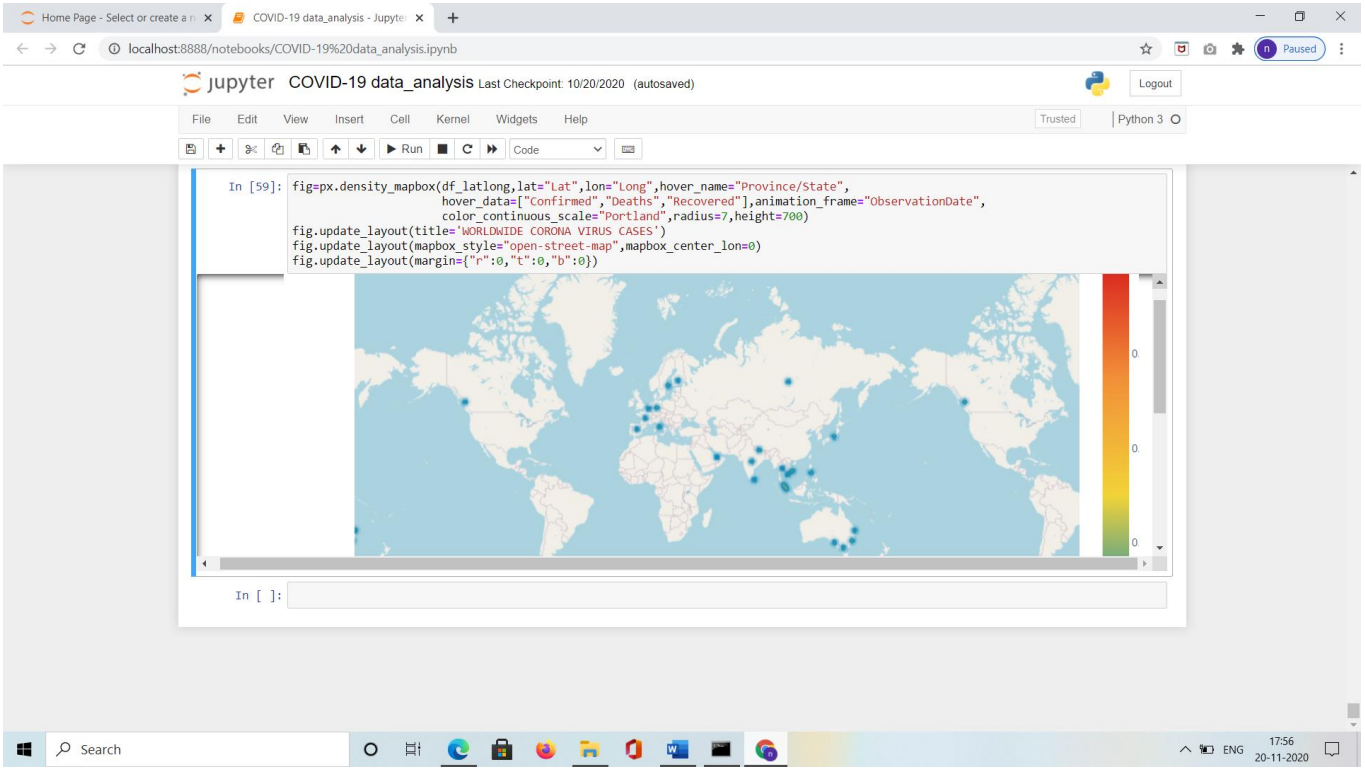


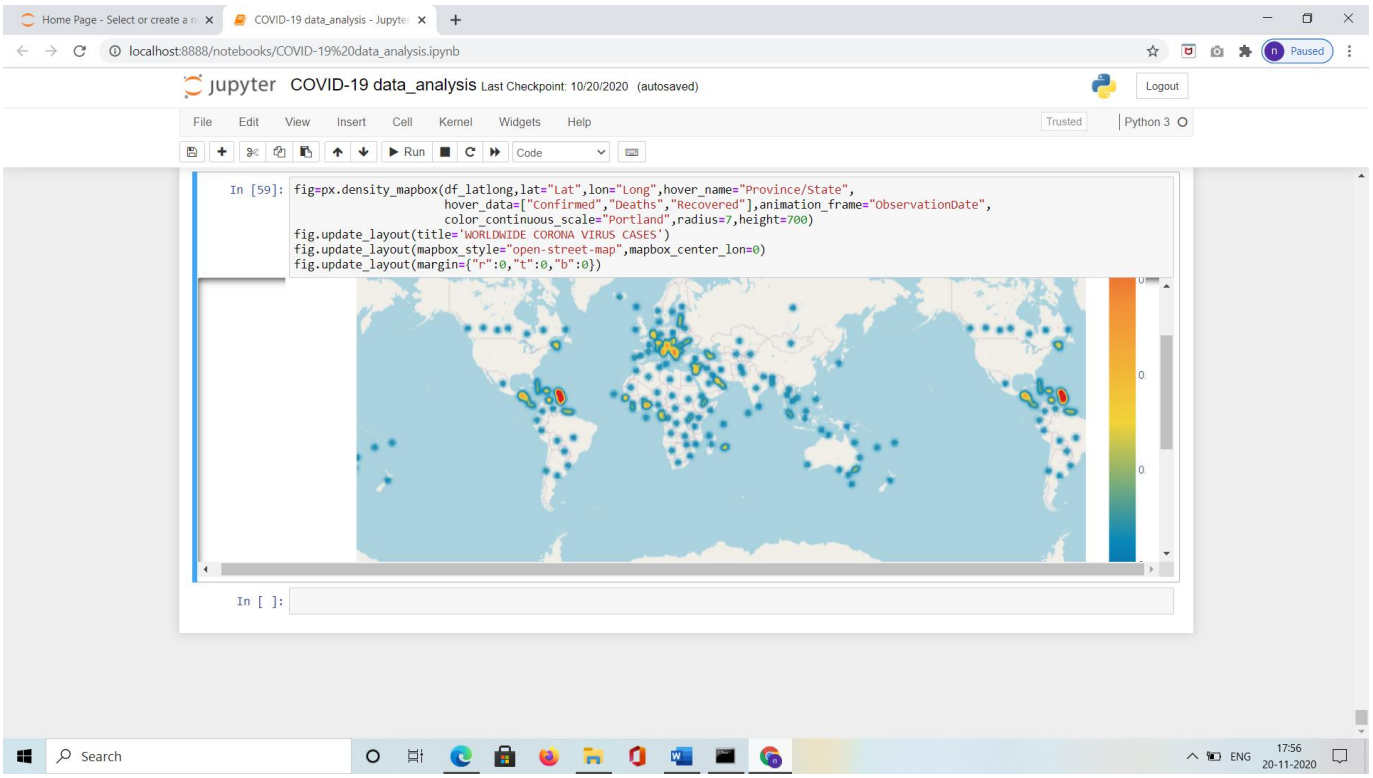
Out[49]:

	SNo	ObservationDate	Province/State	Country	Last Update	Confirmed	Deaths	Recovered	Lat	Long	...	7/22/20	7/23/20	7/24/20	7/25/20
0	36	01/22/2020	NaN	Japan	2020-01-22 17:00:00	2	0	0	36.204824	138.252924	...	27136	28114	28883	29684
1	74	01/23/2020	NaN	Japan	2020-01-23 17:00:00	1	0	0	36.204824	138.252924	...	27136	28114	28883	29684
2	120	01/24/2020	NaN	Japan	2020-01-24 17:00:00	2	0	0	36.204824	138.252924	...	27136	28114	28883	29684
3	161	01/25/2020	NaN	Japan	2020-01-25 17:00:00	2	0	0	36.204824	138.252924	...	27136	28114	28883	29684
4	207	01/26/2020	NaN	Japan	2020-01-26 16:00:00	4	0	1	36.204824	138.252924	...	27136	28114	28883	29684
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
30719	73098	07/27/2020	NaN	Lesotho	2020-07-28 04:56:00	505	12	128	-29.610000	28.233600	...	359	359	359	419
30720	73840	07/28/2020	NaN	Lesotho	2020-07-29 04:35:00	505	12	128	-29.610000	28.233600	...	359	359	359	419
30721	74583	07/29/2020	NaN	Lesotho	2020-07-30 04:35:00	576	13	141	-29.610000	28.233600	...	359	359	359	419
30722	75326	07/30/2020	NaN	Lesotho	2020-07-31 04:35:00	604	13	144	-29.610000	28.233600	...	359	359	359	419
30723	76069	07/31/2020	NaN	Lesotho	2020-01-08 04:36:00	604	13	144	-29.610000	28.233600	...	359	359	359	419

30724 rows × 202 columns

Map that shows the gradual increase or decrease in the cases of COVID-19 based ob the given data set.


Home Page - Select or create a n
COVID-19 data_analysis - Jupyter
localhost:8888/notebooks/COVID-19%20data_analysis.ipynb
Paused
jupyter COVID-19 data_analysis Last Checkpoint: 10/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [59]:
fig=px.density_mapbox(df_latlong,lat="Lat",lon="Long",hover_name="Province/State",
hover_data=["Confirmed","Deaths","Recovered"],animation_frame="ObservationDate",
color_continuous_scale="Portland",radius=7,height=700)
fig.update_layout(title='WORLDWIDE CORONA VIRUS CASES')
fig.update_layout(mapbox_style="open-street-map",mapbox_center_lon=0)
fig.update_layout(margin={"r":0,"t":0,"b":0})
In [ ]:
Search
ENG
17:56
20-11-2020


Home Page - Select or create a n
COVID-19 data_analysis - Jupyter
localhost:8888/notebooks/COVID-19%20data_analysis.ipynb
Paused
jupyter COVID-19 data_analysis Last Checkpoint: 10/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [59]:
fig=px.density_mapbox(df_latlong,lat="Lat",lon="Long",hover_name="Province/State",
hover_data=["Confirmed","Deaths","Recovered"],animation_frame="ObservationDate",
color_continuous_scale="Portland",radius=7,height=700)
fig.update_layout(title='WORLDWIDE CORONA VIRUS CASES')
fig.update_layout(mapbox_style="open-street-map",mapbox_center_lon=0)
fig.update_layout(margin={"r":0,"t":0,"b":0})
In [ ]:
Search
ENG
17:56
20-11-2020

Overall Experience

The work responsibilty that we took for completing this project is of a Data Analyst. During our project completion, we have learned about how it feel to bs an analyst, how an analyst is needed for answering various questions from a given dataset and how conclusions are drawn. The work can that we have done for this training could be done better if we would have gone deeper into the studies. We have implemented lot of more things like maps, progressions, bar graphs and other such representations if I had more time to go further into the studies and become a reliable data analyst for an organisation.

We must say that we had already stepped into the world of a data analyst and have started experiencing the long to go journey but we could have gained more experience into this if we had a personal guide with me who has an experience into this field, so that we would have done better and submitted a better report of our training and project work. I learned a lot from this training and project which would help us to go for further studies into this with an experience of a beginner so that we could do a specialization in this field and do much better than what we have done in this project and could learn more in this field and become a specialist in the field of data analysis.

References

The source of the data used in the project is:

- google.com
- mohfw.gov.in (The official site of Ministry Of Health And Family Welfare of India)
- covid19india.org
- worldometers.info/coronavirus
- Some other sites as well for collecting data.
- ***(Data upto 31st July, 2020)***