 **GROUP NO. 7**

**Data Analysis Project on Covid-19**

PRESENTED BY – ADITYA PRATAP VERMA( 3rd Sem CSE)

* TABLE OF CONTENT

|  |  |
| --- | --- |
| 1. | TABLE OF FIGURES |
| 2. | ABSTRACT |
| 3. | CHAPTERS |
| 4. | COMPLETE CODE  (attached ipynb file) |
| 5. | CONCLUSION |

* TABLES OF FIGURES/GRAPHS

|  |
| --- |
| 1. **STATEWISE COVID DATA IN INDIA**   Figure 1………………………………………………………………….  Figure 2………………………………………………………………….  Figure 3………………………………………………………………….   1. **OVERALL COVID 19 STATUS IN INDIA**   Figure 1………………………………………………………………….   1. **VISUALISING THE SPREADS GEOGRAPHICALLY**   Map 1………………………………………………………………….  Map 2………………………………………………………………….  Map 3………………………………………………………………….   1. **TREND OF VIRUS**   Graph 1………………………………………………………………….  Graph 2………………………………………………………………….  Graph 3………………………………………………………………….  Graph 4………………………………………………………………….   1. **FORECAST USING** *fbprophet*   Graph 1………………………………………………………………….  Graph 2………………………………………………………………….  Graph 3…………………………………………………………………. |

* **ABSTRACT**

We are creating a data analysis Project on Spread of Novel Corona Virus in India. The project uses raw data in the form of .csv and .xls files and transforms it into Data Analysis. This project is an attempt of data analysing Coronavirus(COVID-19) spread in India with the help of data science and data analytics in python code. This analysis will help us to find the basis behind common notions about the spread from purely a dataset perspective.

The data used for this project is split across two files:-

1. Covid\_19\_india.csv
2. Indian Coordinates.xls

For the analysis we use some libraries such as pandas, numpy, matplotlib, seaborn, plotly, stdlib, folium, warnings and prophet.

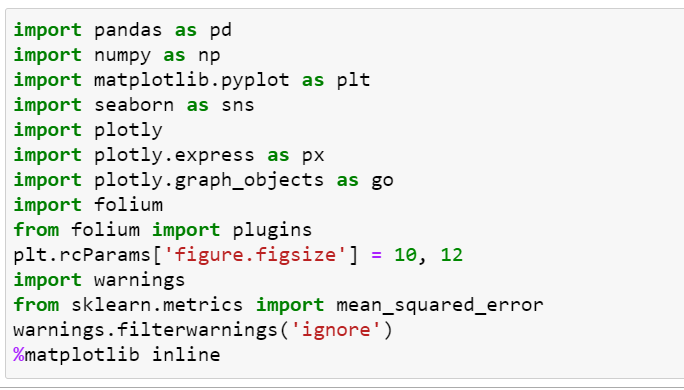
In the project we import files and explore the parameters in both files. By using some user defined functions we clean the data.

We also filter off unwanted entries, thus making our data perfect for analysis.

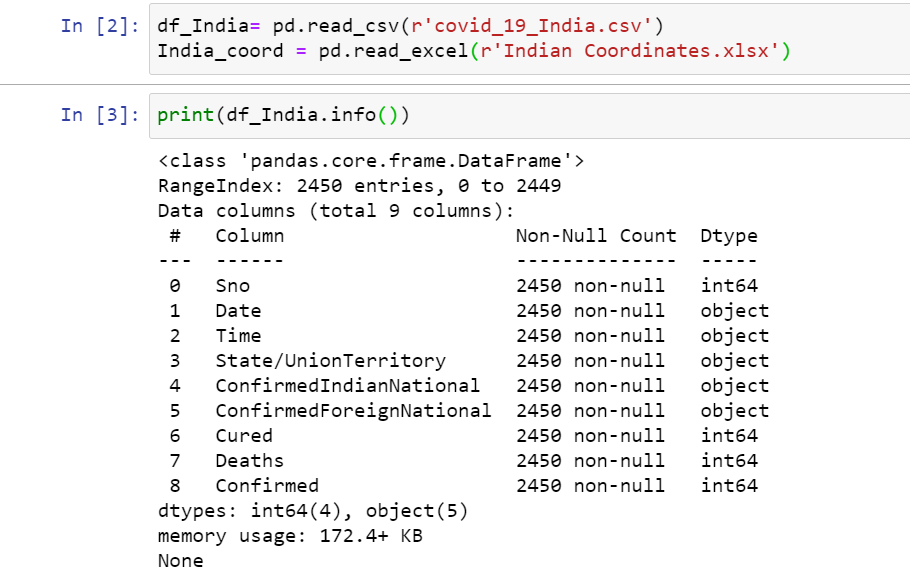
Then we merge the data from dataset into DataFrames for further analysis.

* **CHAPTERS**

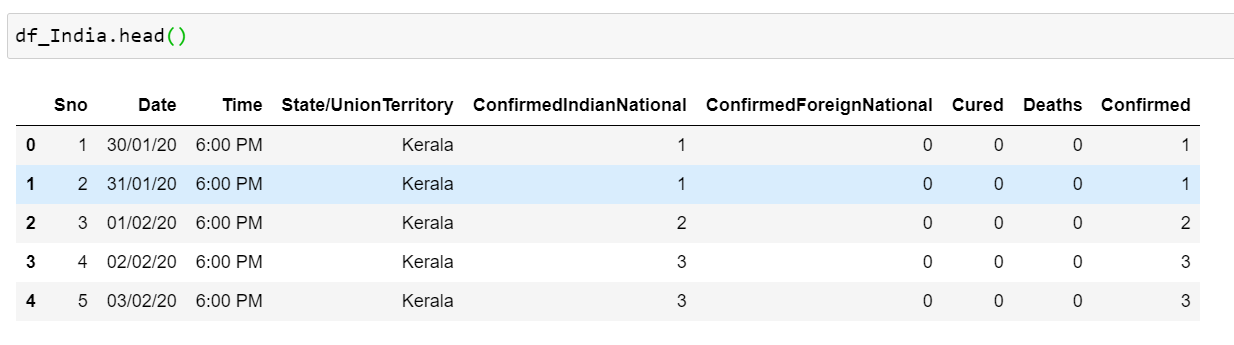
**Importing Data Visualization libraries in python**



* Pandas is a must for data-science. Import pandas as “pd” instead of using the name pandas it’s instructed as pd. It provides fast, expressive, and flexible data structures to easily (and intuitively) work with structured (tabular, multidimensional, potentially heterogeneous) and time-series data.
* NumPy is a Pythonlibrary 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. NumPy stands for Numerical Python.
* Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It allows you to generate high quality line plots, scatter plots, histograms, bar charts, and much more.
* Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
* The **Plotly Python**library is an interactive open-source library. This can be a very helpful tool for data visualization and understanding the data simply and easily. It can plot various types of graphs and charts like scatter plots, line charts, bar charts, box plots, histograms, pie charts, etc.
* Folium is a Pythonlibrary used for visualizing geospatial data. It is easy to use and a powerful library.
* The plotly.express module (imported as px) contains functions that can create entire figures at once, and is referred to as Plotly Express or PX. Plotly Express is a built-in part of the plotly library, and is the recommended starting point for creating most common figures. Every Plotly Express function uses graph objects internally and returns a plotly.graph\_objects.
* plt.rcParams['figure.figsize'] integer If not provided, a new figure will be created, and the figure number will be incremented.
* Import warnings - program to display warning a message.
* Mean squared error - the average squared difference between the estimated values and true value.

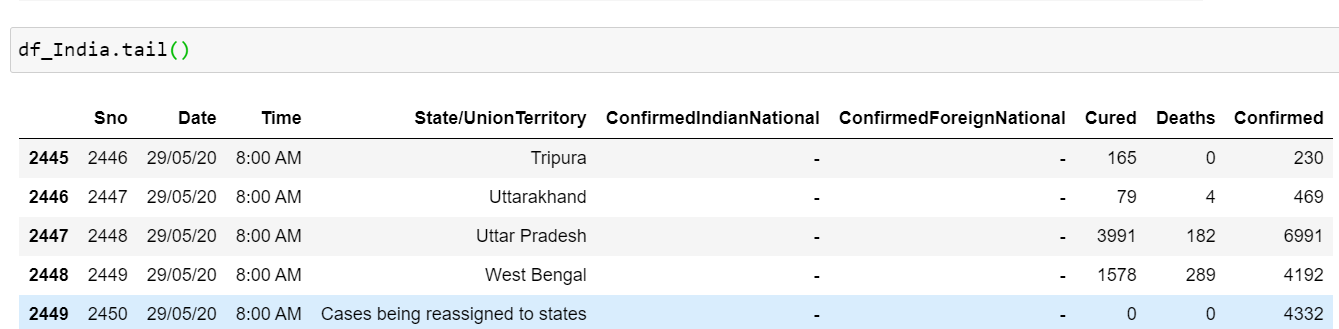


* We use the read\_csv() function from pandas to import data from our system. And similarly We can use the pandas module read**\_**excel() function to read the excel file data into a DataFrame object.



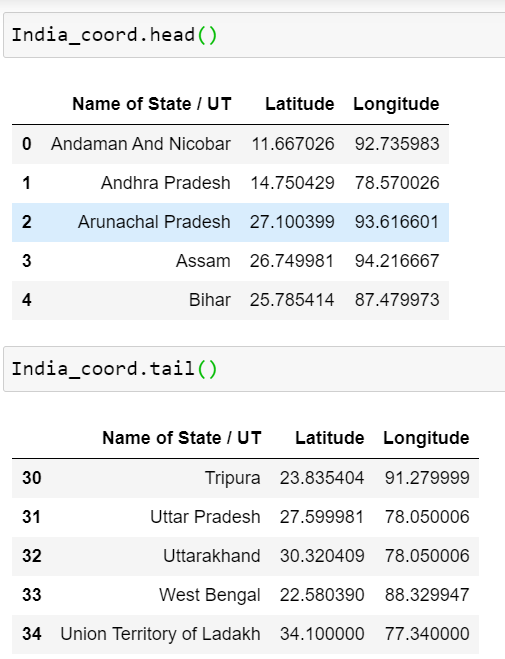
* df\_India.head()

Pandas head() method is used to return top 5 rows(by default) of a data frame or series.



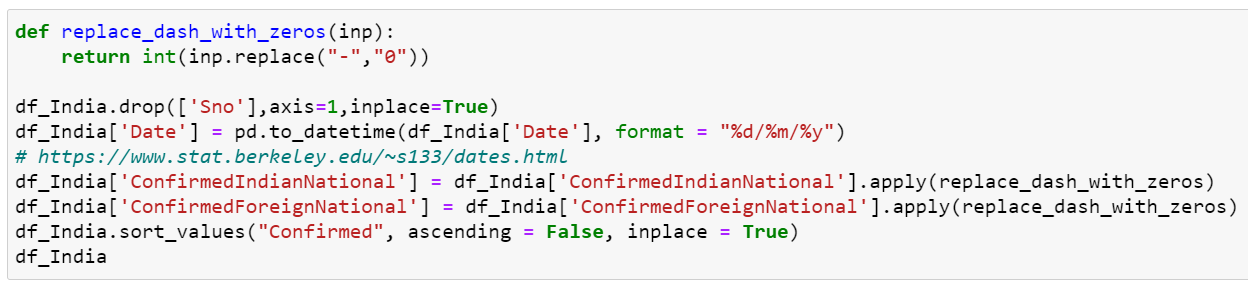
* df\_India.tail()

This function returns last n rows from the object based on position. It is useful for quickly verifying data, for example, after sorting or aappending rows.



* Similarly we get the first 5 state’s coordinates from India\_coord.head() and the last 5 coordinates of from India\_coord.tail().

**DATA CLEANING**



* We have defined a function named “replace\_dash\_with\_zeros”

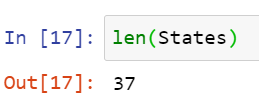
It replaces all the dashes present in the data set with zero.

* df\_India.drop(['Sno'],axis=1,inplace=True)

This line drops the serial number column(axis = 1 = column1) present in the data set.



* We have defined a variable “States” that will fetch data of State/UnionTerritory from .csv file and unique() method is used to know all type of unique values in State/UnionTerritory column. And .tolist() will present the data in a form of list.

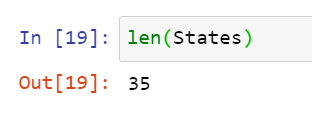


* len(States) **prints** the number of states present in the list.
* States.remove("Cases being reassigned to states")

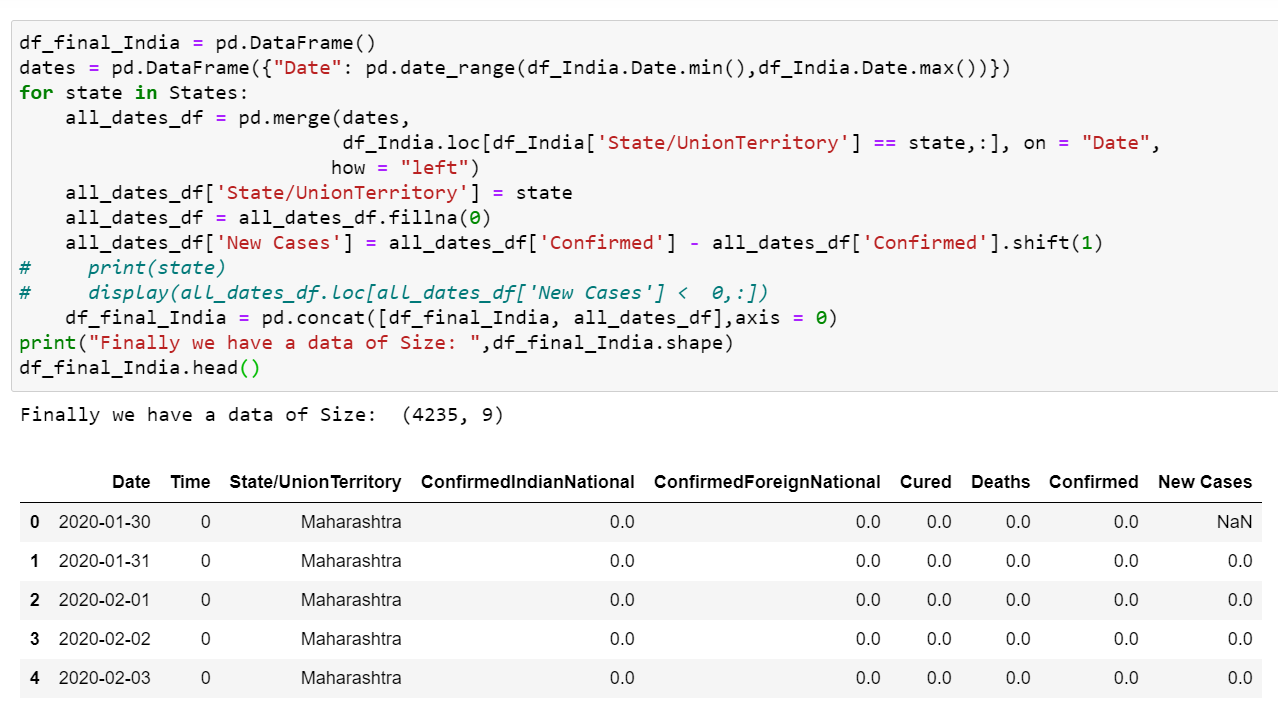
States.remove("Unassigned")

States

* From this statement the states will be removed where the cases were reassigned and Unassigned.

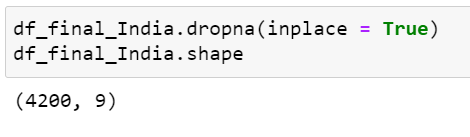
Hence the number of states will be 35.

**MERGING DATA FRAMES**

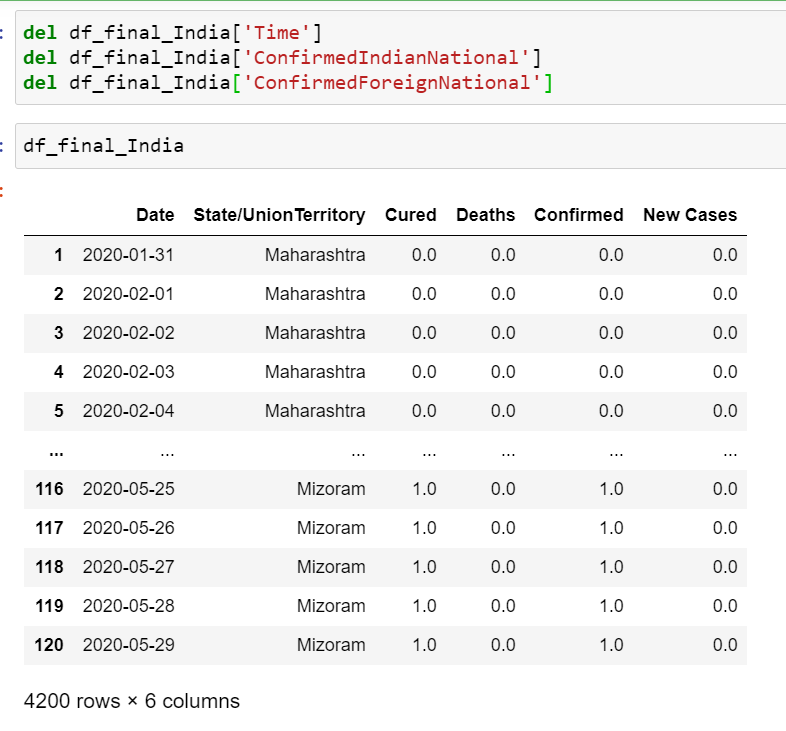
****

* **df\_final\_India = pd.DataFrame()** – Creating an empty data set using .DataFrame().
* **dates = pd.DataFrame({"Date": pd.date\_range(df\_India.Date.min(),df\_India.Date.max())}) –** Storing dates in the variable ranging from the start date to the end date.

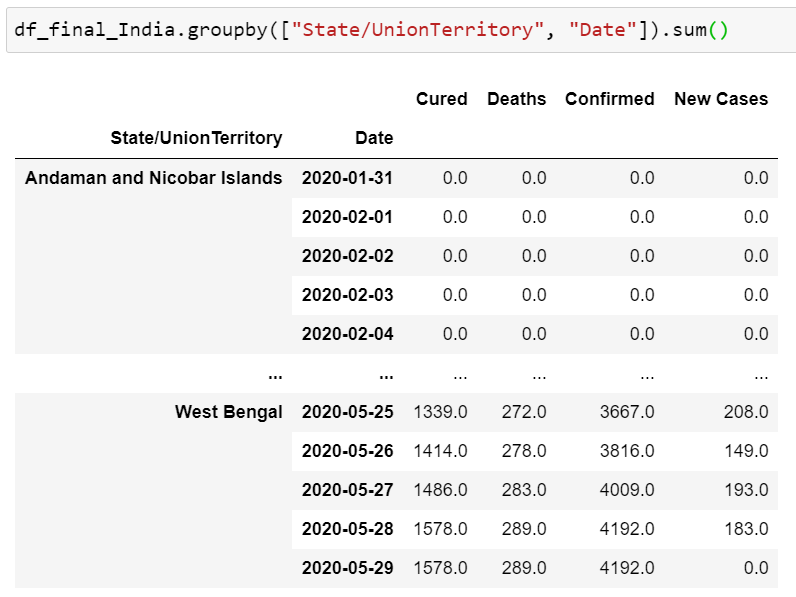
**Pd.merge() -** Merge DataFrame or named Series objects with a database-style join.  If joining columns on columns, the DataFrame indexes  be ignored. Otherwise if joining indexes on indexes or indexes on a column or columns, the index will be passed on.

****

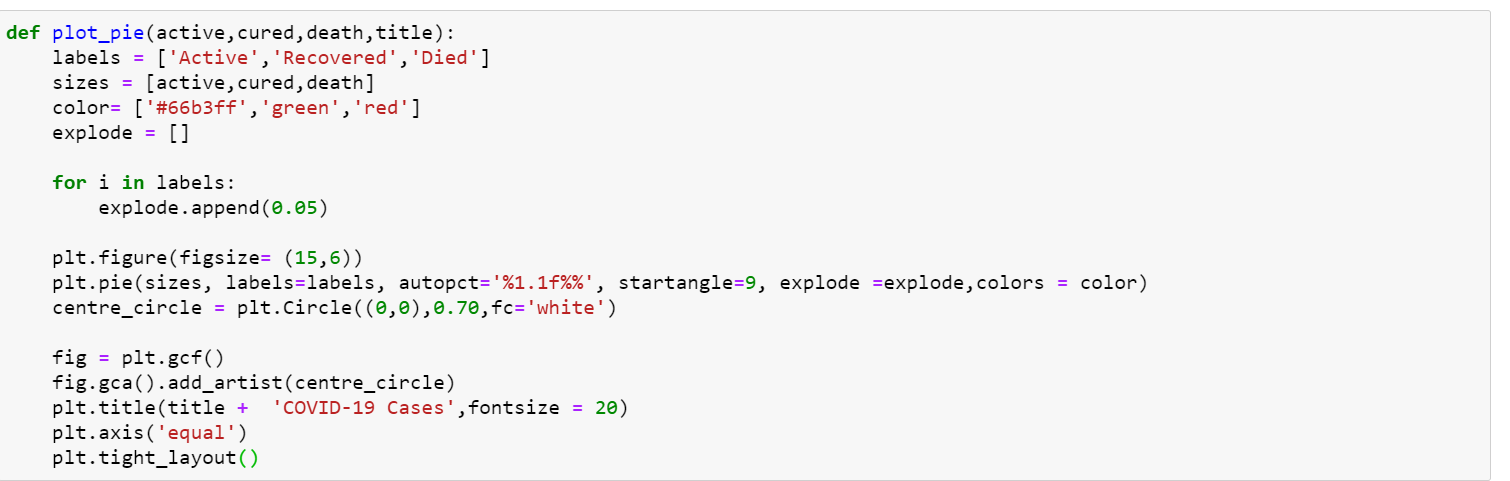
* **.dropna(inplace = True) -** Pandas dropna() method allows the user to analyze and drop Rows/Columns with Null values in different ways.  inplace=True returns None inplace=False returns a copy of the object with the operation performed.
* **.shape() -** The **shape** attribute for numpy arrays returns the dimensions of the array. Here the dimesions are 4200, 9

****

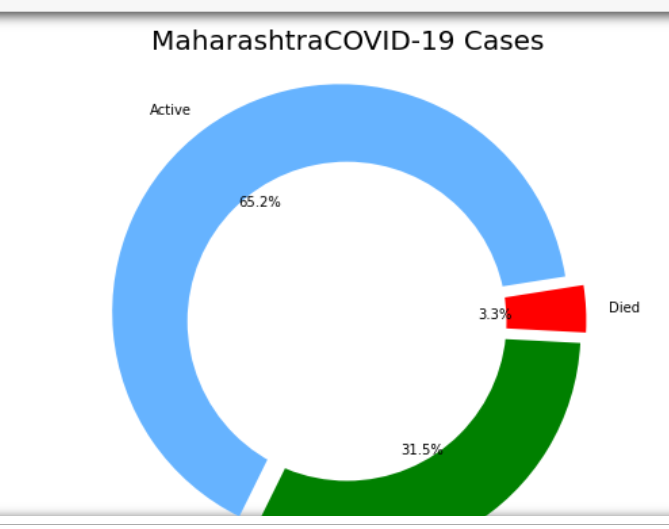
* We have deleted 3 rows i.e “Time”, “ConfirmIndianNational”, “ConfirmForiegnNational” using the keyword **del.**

****

* .groupby() - This can be used to group large amounts of data and compute operations on these groups. Here we have grouped according to the date.

****

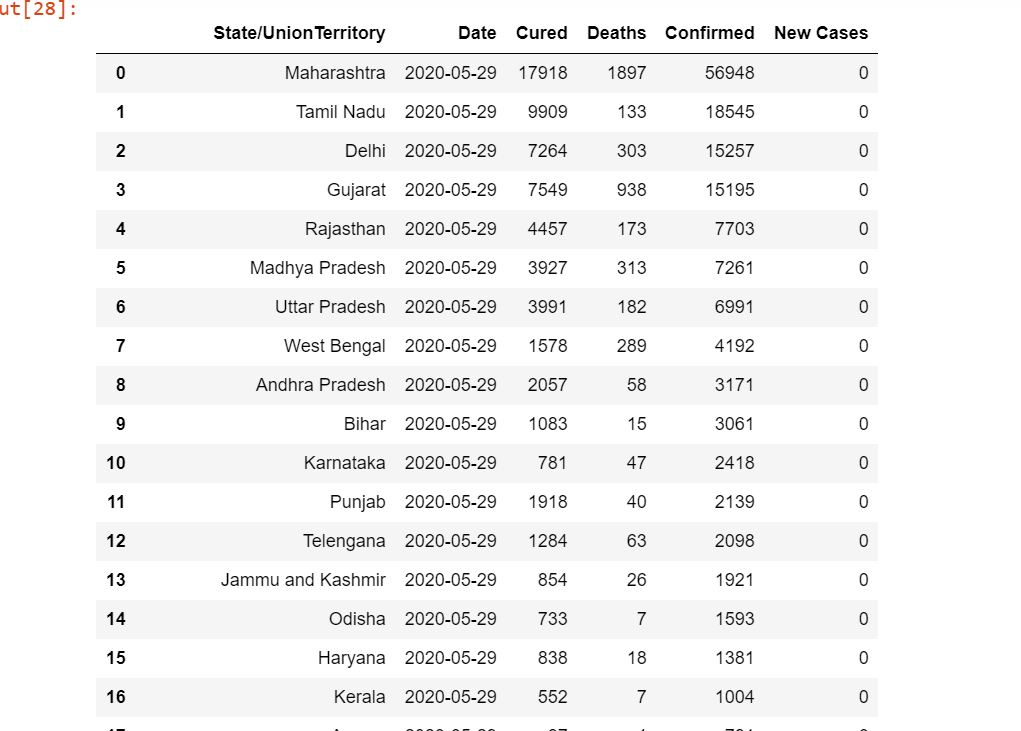




* We have defined a function named plot\_pie which will plot a pie chart for data “Active”, “Recovered”, “Died”.
* Matplotlib allows the aspect ratio, **figure size** to be specified when the Figure object is created here we have figsize(15,6)
* **plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=9, explode =explode,colors = color) - autopct** enables you to display the percent value using **Python** string formatting. For example, if **autopct**='%. 2f' , then for each pie wedge, the format string is '%. 2f' and the numerical percent value for that wedge is pct , so the wedge label is set to the string '%.
* **centre\_circle = plt.Circle((0,0),0.70,fc='white') –** This line defines the coordinated of the pie plot and the color assigned to it.

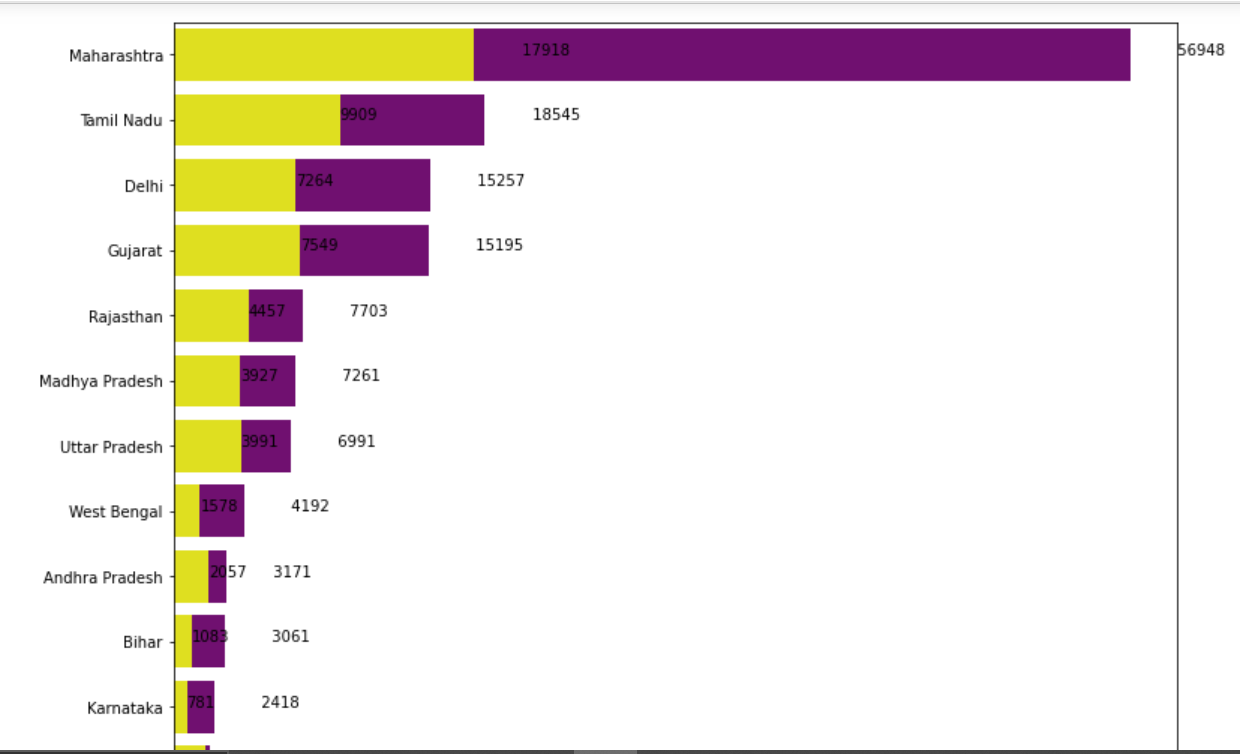
****

* **reset\_index()** is a method to reset index of a Data Frame. reset\_index() method sets a list of integer ranging from 0 to length of data as index.





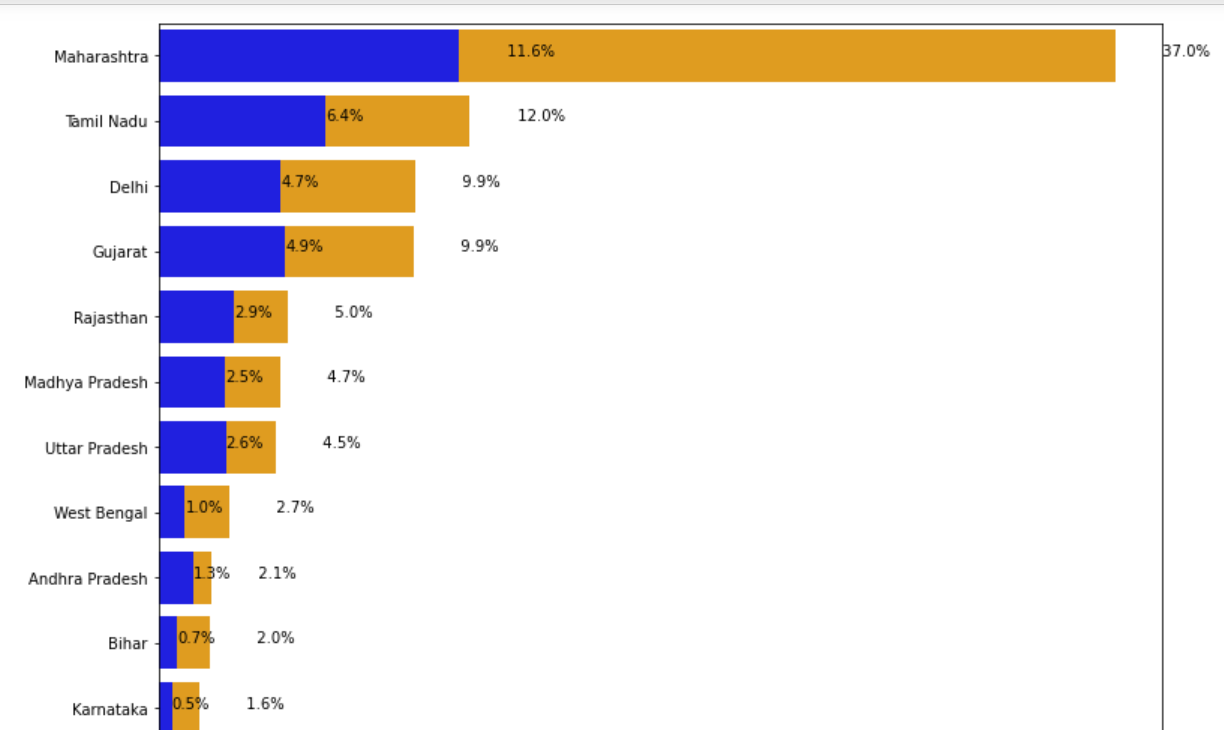
* Here we are plotting a bar-graph for number of Total cases and number of cured cases.
* .patches is used to give a rectangle patch to a plot with lower left at xy = (x, y) with specified width, height and rotation angle.
* Matplotlib.legend - The legend module defines the Legend class, which is responsible for drawing legends associated with axes and/or figures.



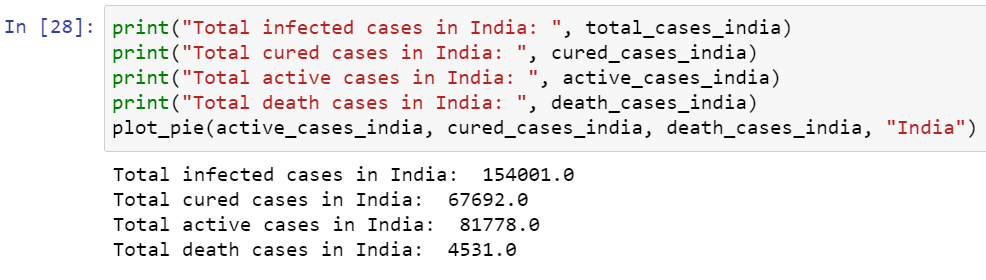


* Here data Total case vs Cured cases is represented in percentage format.

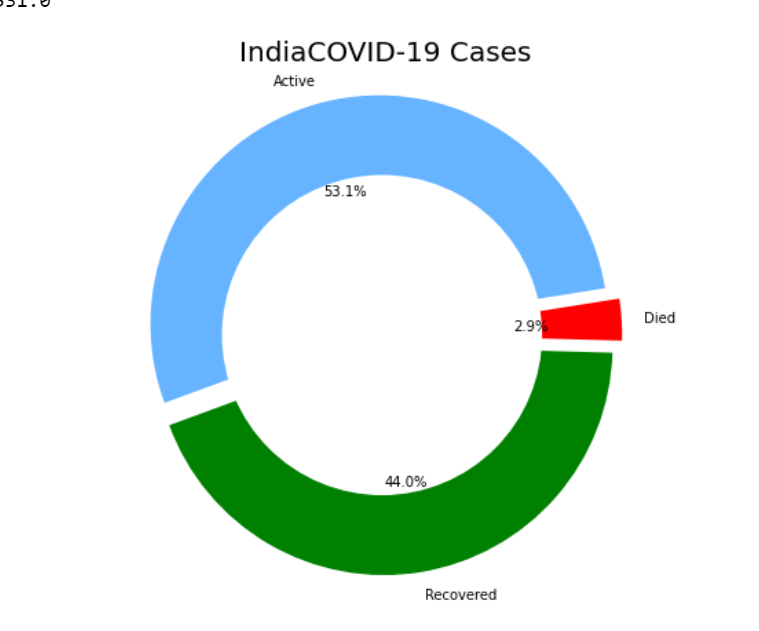
**percentage = '{:.1f}%'.format(100 \* p.get\_width()/total)**

****

**Overall Covid19 Status in India**

****

* We have printed the overall covid cases happened in India from a pie plot i.e total infected cases, total cured cases, total active cases and total deaths. Active cases are 53.1%, Recovered cases are 44% and deaths are 2.9% of the total population of India.



# VISUALISING THE SPREADS GEOGRAPHICALLY

# 

# Here we want to visualise geographically so we have returned values using .values from India\_coord.

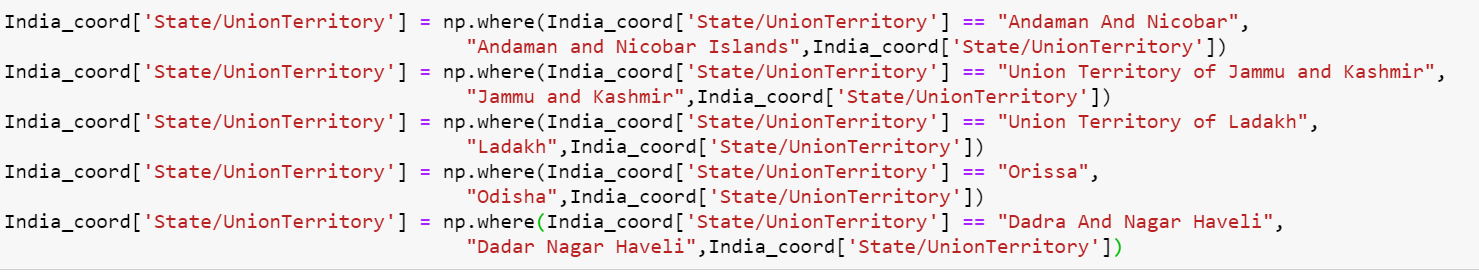
# 

# The strip() method removes any leading (spaces at the beginning) and trailing (spaces at the end) characters.

# 

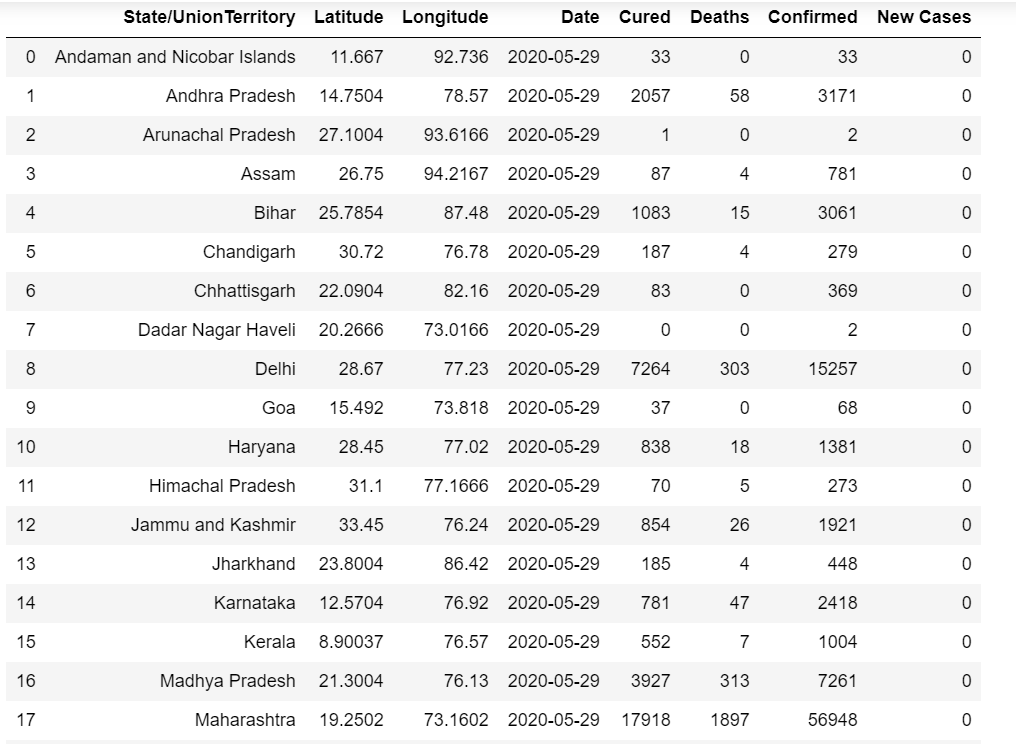
# We have used set to store multiple items in a single variable.

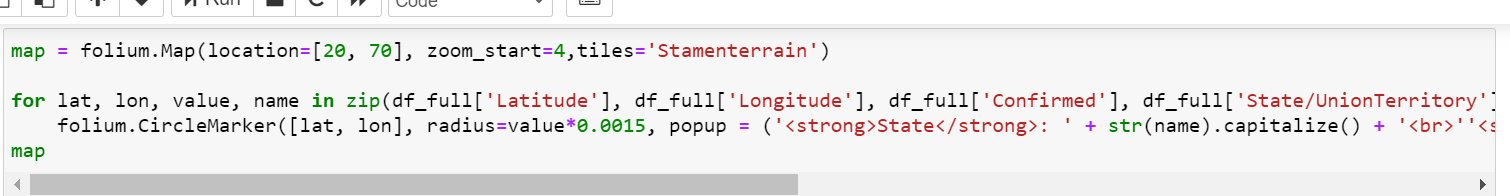
* **India\_coord.loc[India\_coord.shape[0]] - .loc**() method is a method that takes only index labels and returns row or dataframe if the index label exists in the caller data frame.



* The **np.where()** function returns the indices of elements in an input array where the given condition is satisfied.

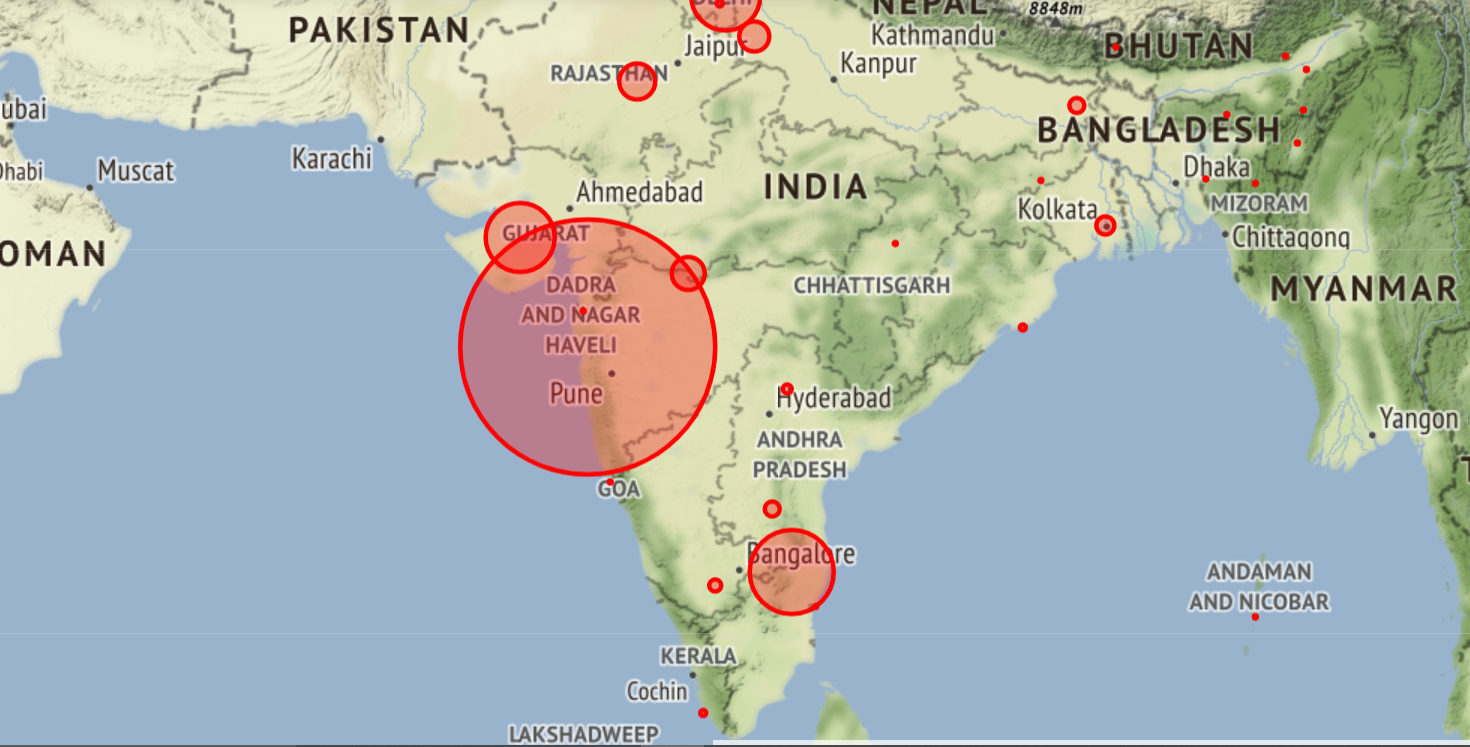
***condition :****When True, yield x, otherwise yield y.****x, y :****Values from which to choose. x, y and condition need to be broadcastable to some shape.*

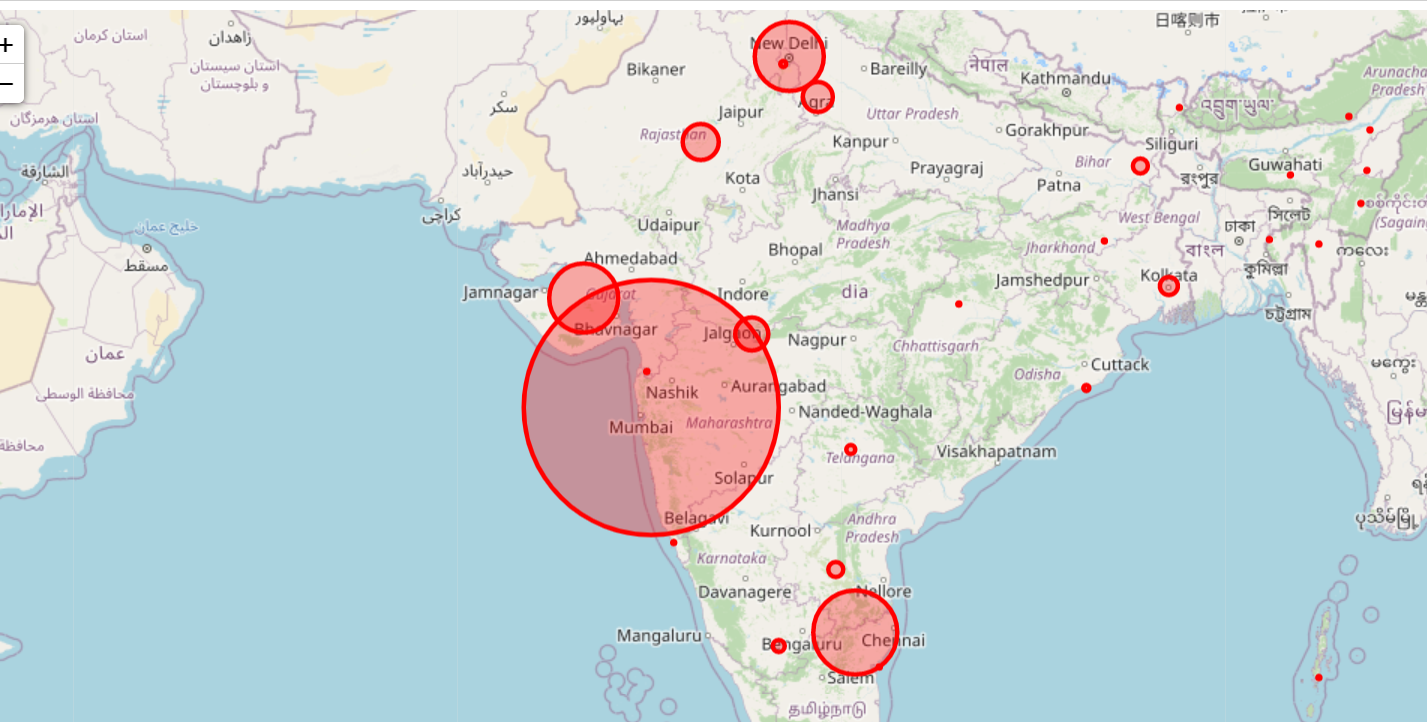
* 
* This resets the index values of the data showing Latitude and Longitude, datewise, cured cases, Deaths, Confirmed cases and New cases.

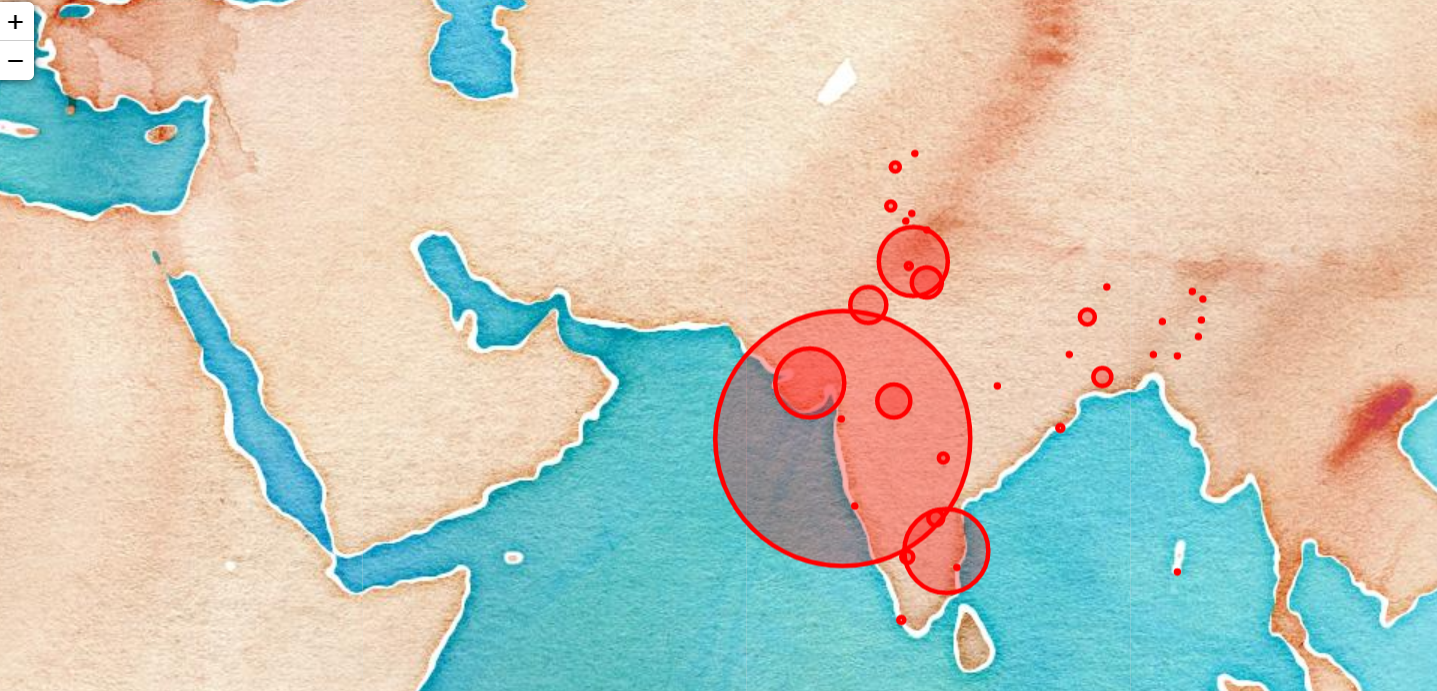


We use folium package of python to draw maps.

* Folium makes It easy to visualize data that’s been manipulated in python on an interactive leaflet map. It enables both the binding of data to a map for choropleth visualizations as well as passing rich vector/raster/HTML visualizations as markers on the map.
* Map is the variable where store the code for displaying map. Location means we have to locate this coordinate on map. Zoom\_start will tell us till what level can we zoom in the map and tiles are of different types of maps. Here we use ‘stamenterrain’.
* For all values of lat i.e latitude lon i.e longitude and name in zip.
* The purpose of zip() is to**map the similar index of multiple containers** so that they can be used just using as single entity.





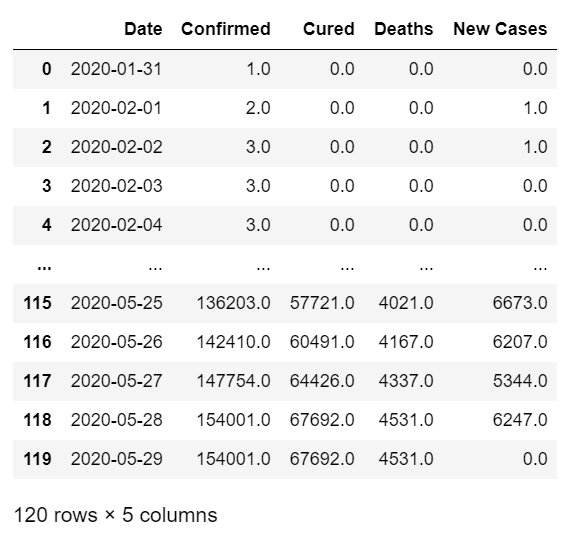


**In these three maps, these are the different geographical representation of virus spread.**

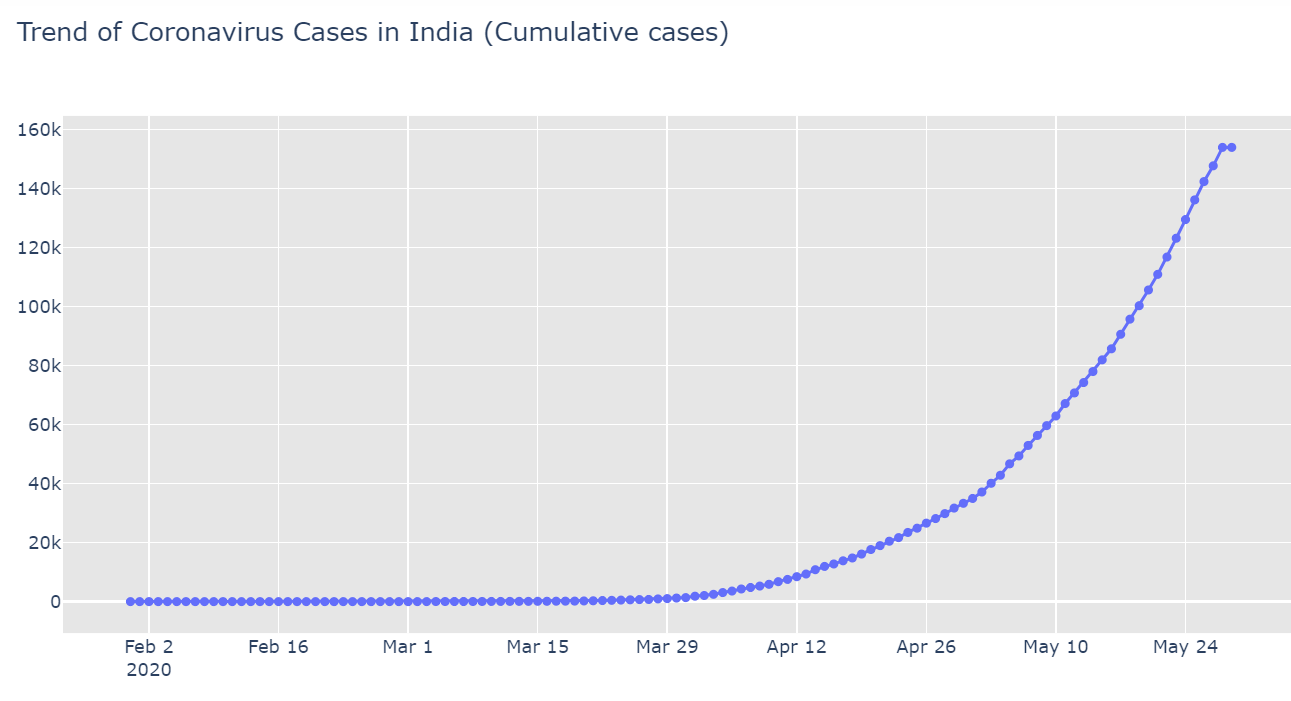
# Lets check the trend of the virus

# 

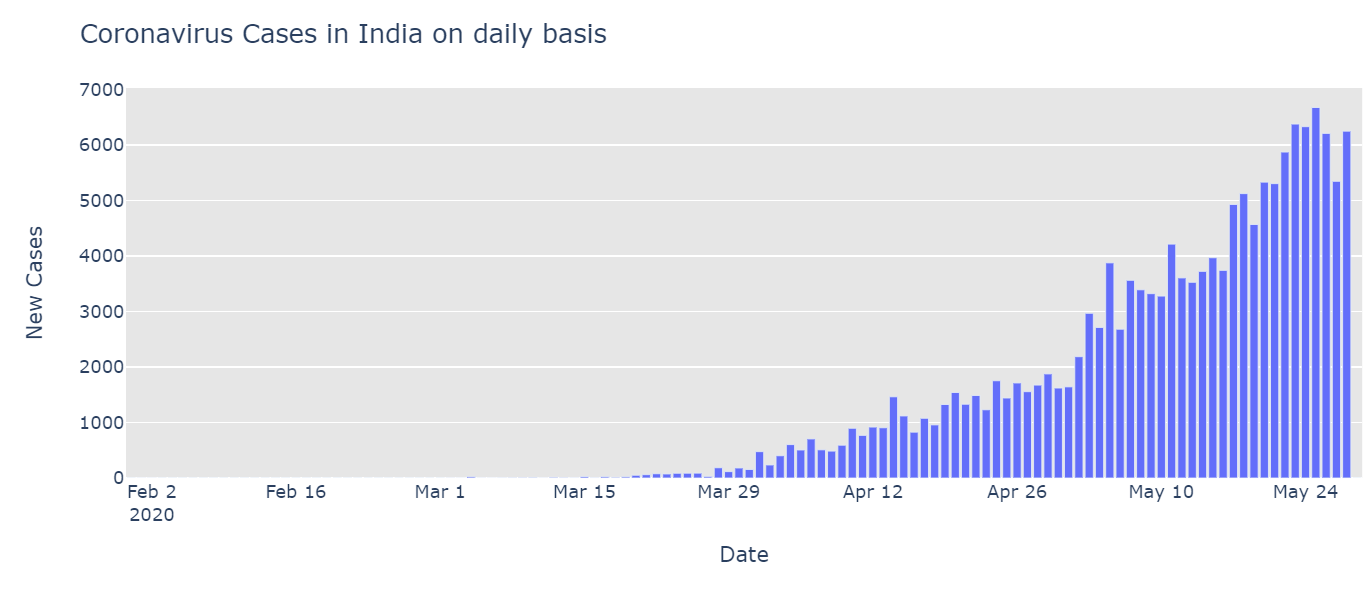
# Now We want to check the trend of the virus, for this we are taking the day-wise report. We are using .groupby() Pandas****dataframe.groupby()**** function is used to split the data into groups based on some criteria. pandas objects can be split on any of their axes. The abstract definition of grouping is to provide a mapping of labels to group names.



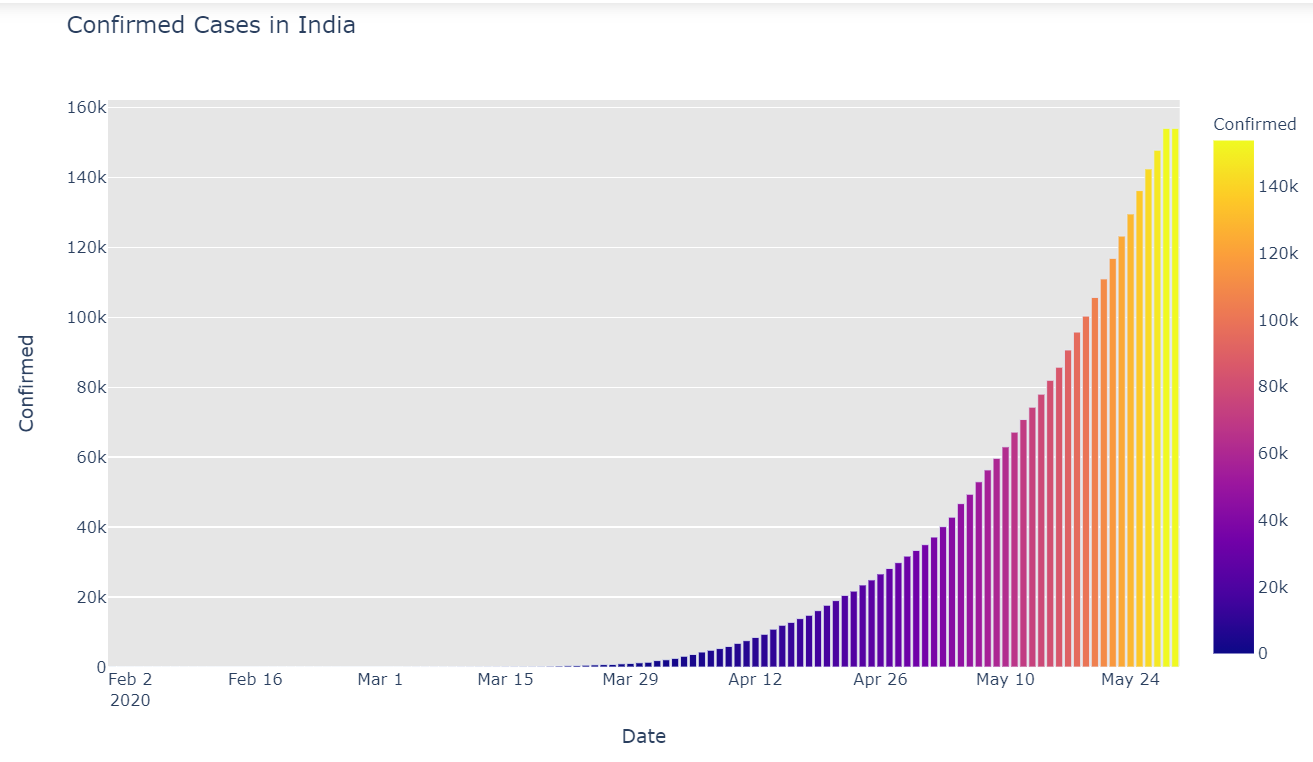
For 1st Jan 2020 there was 1 confirmed case and none were cured, deceased or new. Similarly other data is grouped together using .groupby()



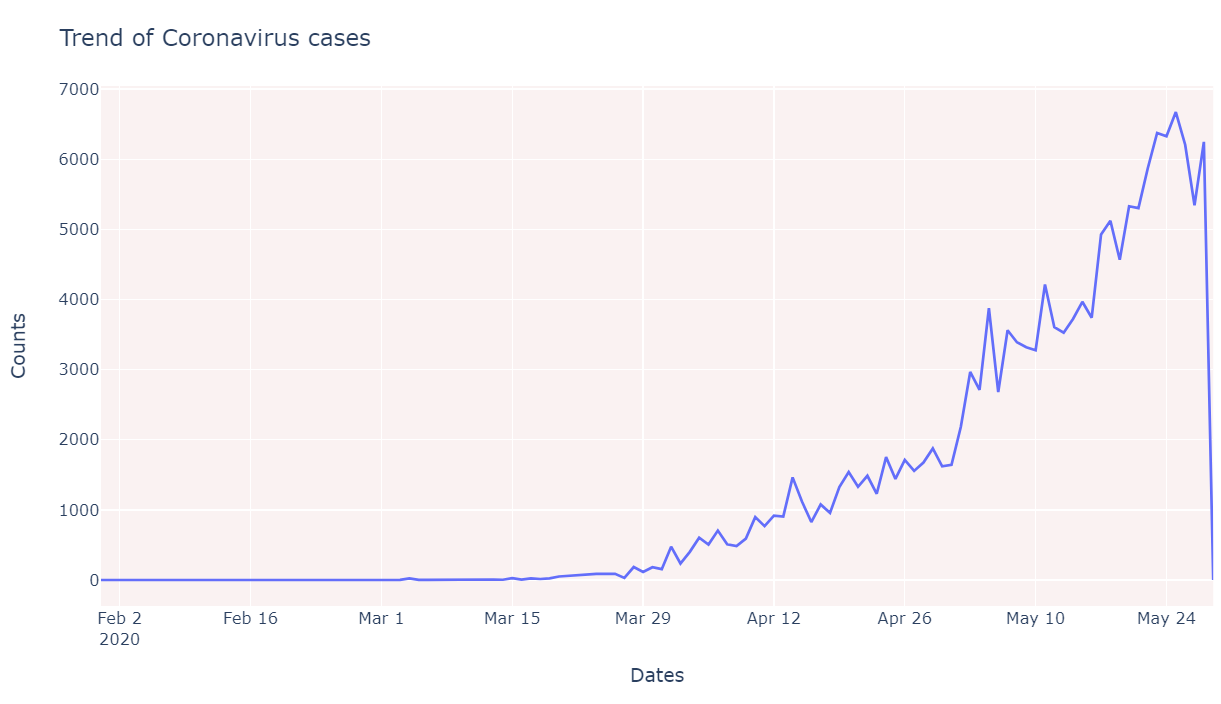
**fig.add\_trace()** Here is an example of day-wise trend of creating a figure that includes a single trend of scatter traces in the subplot layout.



* A bar plot mode which are grouped shows COVID-19 cases in India on daily basis. New cases vs Date. Defined as on x axis="Date", y axis="New Cases", barmode='group', with max height=400.



* A better representation of Confirmed cases date-wise is shown using barplot with a max height of 600. Matplotlib is the most popular graphing and data visualization library for Python.



 With **px.line**, each data point is represented as a vertex (which location is given by the x and y columns) of a **polyline mark** in 2D space.

# Forecasting Using fbprophet

# It's a tool intended to help you to do time series forecasting at a scale with ease.

# Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data.

# 

# Dataframe.iloc() method is used when the index label of a data frame is something other than numeric series of 0, 1, 2, 3…. n or in case the user doesn't know the index label. Rows can be extracted using an imaginary index position which isn't visible in the data frame.

# Df\_train, df\_test - Separating data into training and testing sets is an important part of evaluating data models.

# After a model has been processed by using the training set, you test the model by making predictions against the test set. Because the data in the testing set already contains known values for the attribute that you want to predict, it is easy to determine whether the model's guesses are correct.

# 

# 

# predict() : predicts the label of a new set of data. This method accepts one argument, the new data X\_new (e.g. .predict(future) ), and returns the learned label for each object in the array.

# 

# 

# Result and the predicted values are shown with trend, additive terms, multiplicative terms and weekly.

# We

# 

# Forecast result for only ‘ds’, ‘yhat’, ‘yhat\_lower’, ‘yhat\_upper’ Is shown using .tail(5) where last 5 row’s data will be displayed.

# 

# Here on x axis we have dates and on y axis we have results for prophet.

# The line graph shows data of Actual and predicted cases.

# 

# recovered\_train.columns = ['ds','y']

# recovered\_train.tail() – here after recovering the train values and showing ‘ds’, ‘y’.

# 

# 

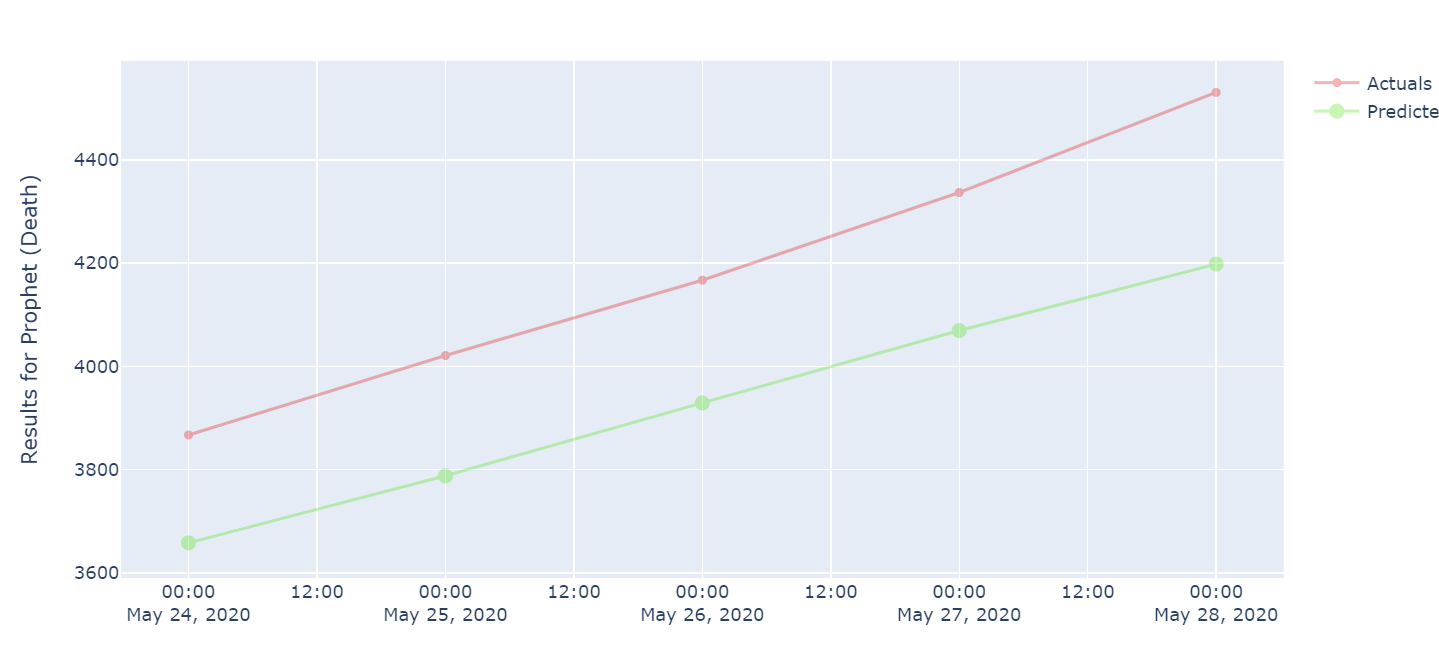
# When cases were Recovered, the predicted and actual values varied – the variance in line plot is shown below:

# 

# Graph shows results for recovered cases Actual vs predicted values.

# 

# This test data shows the death which were actual and predicted.



**The above 3 line plots shows the results for prophet (Total cases, Recovered and Deaths) due to the virus spread.**