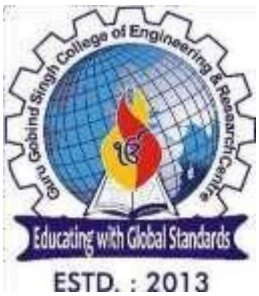



# Guru Gobind Singh College of Engineering and Research Centre, Nasik

## Department of Computer Engineering

**Academic Year 2022-23**
**Sem-VI**

**Class:** TE Computer

Title of the Report: Data Science & Big Data Analytics Mini-Project

Submitted by:

Mr. Prathamesh Pawar

Under the guidance of:

Mr. Sandeep G. Shukla

**DEPARTMENT OF COMPUTER ENGINEERING**

# INDEX

## 1. Introduction (Problem Definition) :

Use the following covid_vaccine_statewise.csv dataset and perform following analytics on the
given dataset
https://www.kaggle.com/sudalairajkumar/covid19-in-india?select=covid_vaccine_statewise.csv
a. Describe the dataset
b. Number of persons state wise vaccinated for first dose in India
c. Number of persons state wise vaccinated for second dose in India
d. Number of Males vaccinated
d. Number of females vaccinated

COVID-19 has been one of the biggest health crises that the world has faced in recent times. To combat this pandemic, several vaccines have been developed, and many countries are now rolling out vaccination drives. In India, the government launched a massive vaccination drive in January 2021. In this case study, we will analyze the COVID-19 vaccination data for India and derive insights that can help in better understanding the vaccination drive.

## 2. Dataset Description :

The dataset that we will be using for this case study is the 'covid_vaccine_statewise.csv' dataset from Kaggle. This dataset contains information on the COVID-19 vaccination drive in India, broken down by state. The dataset includes the following variables:

State: The name of the state or union territory
Total Vaccination Doses: The total number of vaccine doses administered in the state
Total Sessions Conducted: The total number of vaccination sessions conducted in the state
Total Sites: The total number of vaccination sites in the state
First Dose Administered: The total number of first doses administered in the state
Second Dose Administered: The total number of second doses administered in the state
Male(Individuals Vaccinated): The total number of males vaccinated in the state
Female(Individuals Vaccinated): The total number of females vaccinated in the state
Transgender(Individuals Vaccinated): The total number of transgender individuals vaccinated in the state
Total Covaxin Administered: The total number of Covaxin doses administered in the state
Total CoviShield Administered: The total number of Covishield doses administered in the state
Total Sputnik V Administered: The total number of Sputnik V doses administered in the state

## 3. Feature Selection :

For analyzing COVID-19 vaccination data in India, some of the features that can be selected for analysis are:

Total Vaccination Doses: This feature represents the total number of vaccine doses administered in each state. It can provide insights into the overall progress of the vaccination drive in each state and can help identify states that need more attention.

Total Sessions Conducted: This feature represents the total number of vaccination sessions conducted in each state. It can provide insights into the availability of vaccination facilities in each state and the efficiency of the vaccination drive.

Total Sites: This feature represents the total number of vaccination sites in each state. It can provide insights into the availability of vaccination facilities in each state and the reach of the vaccination drive.

First Dose Administered: This feature represents the total number of first doses administered in each state. It can provide insights into the percentage of the population that has received at least one dose of the vaccine in each state.

Second Dose Administered: This feature represents the total number of second doses administered in each state. It can provide insights into the percentage of the population that has received both doses of the vaccine in each state and the progress towards achieving herd immunity.

Gender Distribution: The features Male(Individuals Vaccinated), Female(Individuals Vaccinated), and Transgender(Individuals Vaccinated) can provide insights into the gender distribution of the vaccinated population in each state.

Vaccine Distribution: The features Total Covaxin Administered, Total CoviShield Administered, and Total Sputnik V Administered can provide insights into the popularity and availability of different types of vaccines in each state.

# 4.Model Building (Jupyter Code)

Name :
Prathmesh Patil
Prathamesh Pawar
Pratik Rao
Sankalp Shelar

Introduction:- In this task we do Data Analysis on covid_19_india Dataset and find some useful insight.

```
"""importing libraries"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
```

## Importing Data.

```
df_india =
pd.read_csv("/kaggle/input/covid19-in-india/covid_19_india.csv")
df_india
```

```
         Sno        Date     Time State/UnionTerritory  \
0          1  2020-01-30  6:00 PM                Kerala
1          2  2020-01-31  6:00 PM                Kerala
2          3  2020-02-01  6:00 PM                Kerala
3          4  2020-02-02  6:00 PM                Kerala
4          5  2020-02-03  6:00 PM                Kerala
...      ...         ...      ...                   ...
18105  18106  2021-08-11  8:00 AM            Telangana
18106  18107  2021-08-11  8:00 AM              Tripura
18107  18108  2021-08-11  8:00 AM          Uttarakhand
18108  18109  2021-08-11  8:00 AM        Uttar Pradesh
18109  18110  2021-08-11  8:00 AM          West Bengal

      ConfirmedIndianNational ConfirmedForeignNational    Cured
Deaths  \
0                           1                        0        0
0
1                           1                        0        0
0
2                           2                        0        0
0
3                           3                        0        0
0
4                           3                        0        0
0
...                       ...                      ...      ...     ..
.
18105                       -                        -   638410
```

```
3831
18106                         -                         -    77811
773
18107                         -                         -   334650
7368
18108                         -                         -  1685492
22775
18109                         -                         -  1506532
18252

       Confirmed
0              1
1              1
2              2
3              3
4              3
...          ...
18105     650353
18106      80660
18107     342462
18108    1708812
18109    1534999

[18110 rows x 9 columns]
```

## Getting familiar with Data

```
"Checking the range of data"

df_india.shape

(18110, 9)

"Getting information about Data type and non-null values"

df_india.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18110 entries, 0 to 18109
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Sno                       18110 non-null  int64
 1   Date                      18110 non-null  object
 2   Time                      18110 non-null  object
 3   State/UnionTerritory      18110 non-null  object
 4   ConfirmedIndianNational   18110 non-null  object
 5   ConfirmedForeignNational  18110 non-null  object
 6   Cured                     18110 non-null  int64
 7   Deaths                    18110 non-null  int64
 8   Confirmed                 18110 non-null  int64
```

```
dtypes: int64(4), object(5)
memory usage: 1.2+ MB
```

```
"Getting numeric column detials "
```

```
df_india.describe()
```

```
                Sno         Cured         Deaths     Confirmed
count  18110.000000  1.811000e+04   18110.000000  1.811000e+04
mean    9055.500000  2.786375e+05    4052.402264  3.010314e+05
std     5228.051023  6.148909e+05   10919.076411  6.561489e+05
min        1.000000  0.000000e+00       0.000000  0.000000e+00
25%     4528.250000  3.360250e+03      32.000000  4.376750e+03
50%     9055.500000  3.336400e+04     588.000000  3.977350e+04
75%    13582.750000  2.788698e+05    3643.750000  3.001498e+05
max    18110.000000  6.159676e+06  134201.000000  6.363442e+06
```

```
"Getting information of null values in Dataset"
```

```
df_india.isna().sum()
```

```
Sno                          0
Date                         0
Time                         0
State/UnionTerritory         0
ConfirmedIndianNational      0
ConfirmedForeignNational     0
Cured                        0
Deaths                       0
Confirmed                    0
dtype: int64
```

**Note: There is no null value's in dataset**

```
"finding unique values from 'State/UnionTerritory' column"
```

```
df_india['State/UnionTerritory'].unique(),df_india['State/UnionTerritory'].nunique()
```

```
(array(['Kerala', 'Telengana', 'Delhi', 'Rajasthan', 'Uttar Pradesh',
        'Haryana', 'Ladakh', 'Tamil Nadu', 'Karnataka', 'Maharashtra',
        'Punjab', 'Jammu and Kashmir', 'Andhra Pradesh',
'Uttarakhand',
        'Odisha', 'Puducherry', 'West Bengal', 'Chhattisgarh',
        'Chandigarh', 'Gujarat', 'Himachal Pradesh', 'Madhya Pradesh',
        'Bihar', 'Manipur', 'Mizoram', 'Andaman and Nicobar Islands',
        'Goa', 'Unassigned', 'Assam', 'Jharkhand', 'Arunachal
Pradesh',
        'Tripura', 'Nagaland', 'Meghalaya',
        'Dadra and Nagar Haveli and Daman and Diu',
        'Cases being reassigned to states', 'Sikkim', 'Daman & Diu',
        'Lakshadweep', 'Telangana', 'Dadra and Nagar Haveli',
```

```
'Bihar****',
        'Madhya Pradesh***', 'Himanchal Pradesh', 'Karanataka',
        'Maharashtra***'], dtype=object),
 46)
```

```
"Correcting spelling mistakes or impurities"

state_correction_dict = {
    'Bihar****':'Bihar',
    'Dadra and Nagar Haveli':'Dadra and Nagar Haveli and Daman and
Diu',
    'Madhya Pradesh***':'Madhya Pradesh',
    'Maharashtra***':'Maharashtra',
    'Karanataka':'Karnataka'
}

def state_correction(state):
    try:
        return state_correction_dict[state]
    except:
        return state

df_india['State/UnionTerritory'] =
df_india['State/UnionTerritory'].apply(state_correction)
df_india['State/UnionTerritory'].nunique()
```

```
41
```

**Note: Here we have corrected spelling mistakes in column 'State/UnionTerritory'.**

```
"Changing the format of date"

df_india['Date'] = pd.to_datetime(df_india['Date'])
df_india['Date'] = df_india['Date'].dt.strftime('%d-%m-%Y')
df_india['Date']
```

```
0        30-01-2020
1        31-01-2020
2        01-02-2020
3        02-02-2020
4        03-02-2020
           ...
18105    11-08-2021
18106    11-08-2021
18107    11-08-2021
18108    11-08-2021
18109    11-08-2021
Name: Date, Length: 18110, dtype: object
```

```
"Removing unwanted columns from dataset using 'drop'."

df_india.drop(['Time','ConfirmedIndianNational','ConfirmedForeignNatio
```

```
nal'],axis = 1,inplace = True)
df_india
```

```
         Sno        Date State/UnionTerritory    Cured  Deaths
Confirmed
0          1  30-01-2020               Kerala        0       0
1
1          2  31-01-2020               Kerala        0       0
1
2          3  01-02-2020               Kerala        0       0
2
3          4  02-02-2020               Kerala        0       0
3
4          5  03-02-2020               Kerala        0       0
3
...      ...         ...                  ...      ...     ...
...
18105  18106  11-08-2021            Telangana   638410    3831
650353
18106  18107  11-08-2021              Tripura    77811     773
80660
18107  18108  11-08-2021          Uttarakhand   334650    7368
342462
18108  18109  11-08-2021        Uttar Pradesh  1685492   22775
1708812
18109  18110  11-08-2021          West Bengal  1506532   18252
1534999

[18110 rows x 6 columns]
```

```
"Getting only Numeric columns"

num = df_india.select_dtypes(exclude = object)
num
```

```
         Sno    Cured  Deaths  Confirmed
0          1        0       0          1
1          2        0       0          1
2          3        0       0          2
3          4        0       0          3
4          5        0       0          3
...      ...      ...     ...        ...
18105  18106   638410    3831     650353
18106  18107    77811     773      80660
18107  18108   334650    7368     342462
18108  18109  1685492   22775    1708812
18109  18110  1506532   18252    1534999

[18110 rows x 4 columns]
```

```
"Getting only categorical data"

obj = df_india.select_dtypes(include = object)
obj
```

```
             Date State/UnionTerritory
0      30-01-2020               Kerala
1      31-01-2020               Kerala
2      01-02-2020               Kerala
3      02-02-2020               Kerala
4      03-02-2020               Kerala
...           ...                  ...
18105  11-08-2021            Telangana
18106  11-08-2021              Tripura
18107  11-08-2021          Uttarakhand
18108  11-08-2021        Uttar Pradesh
18109  11-08-2021          West Bengal

[18110 rows x 2 columns]
```

## Data Manipulation

```
"Identifying active cases , We counted the values by using values in
confirmed, cured, deaths column"

df_india['Active'] = df_india['Confirmed'] - df_india['Cured'] -
df_india['Deaths']
df_india
```

```
         Sno        Date State/UnionTerritory    Cured  Deaths
Confirmed  \
0          1  30-01-2020               Kerala        0       0
1
1          2  31-01-2020               Kerala        0       0
1
2          3  01-02-2020               Kerala        0       0
2
3          4  02-02-2020               Kerala        0       0
3
4          5  03-02-2020               Kerala        0       0
3
...      ...         ...                  ...      ...     ...
...
18105  18106  11-08-2021            Telangana   638410    3831
650353
18106  18107  11-08-2021              Tripura    77811     773
80660
18107  18108  11-08-2021          Uttarakhand   334650    7368
342462
18108  18109  11-08-2021        Uttar Pradesh  1685492   22775
1708812
```

```
18109  18110  11-08-2021          West Bengal  1506532   18252
1534999

       Active
0           1
1           1
2           2
3           3
4           3
...       ...
18105    8112
18106    2076
18107     444
18108     545
18109   10215

[18110 rows x 7 columns]
```

**Note: We can now check the active cases in each state**

```
"using pivot function to find cured , deaths , confirmed cases State wise"

statewise = pd.pivot_table(df_india,values=['Cured','Deaths','Confirmed'],index='State/UnionTerritory',aggfunc='max',margins=True)
statewise
```

| State/UnionTerritory | Confirmed | Cured | Deaths |
|---|---|---|---|
| Andaman and Nicobar Islands | 7548 | 7412 | 129 |
| Andhra Pradesh | 1985182 | 1952736 | 13564 |
| Arunachal Pradesh | 50605 | 47821 | 248 |
| Assam | 576149 | 559684 | 5420 |
| Bihar | 725279 | 715352 | 9646 |
| Cases being reassigned to states | 9265 | 0 | 0 |
| Chandigarh | 61992 | 61150 | 811 |
| Chhattisgarh | 1003356 | 988189 | 13544 |
| Dadra and Nagar Haveli and Daman and Diu | 10654 | 10646 | 4 |
| Daman & Diu | 2 | 0 | 0 |
| Delhi | 1436852 | 1411280 | 25068 |
| Goa | 172085 | 167978 | 3164 |
| Gujarat | 825085 | 814802 | 10077 |
| Haryana | 770114 | 759790 | 9652 |
| Himachal Pradesh | 208616 | 202761 | 3537 |
| Himanchal Pradesh | 204516 | 200040 | 3507 |
| Jammu and Kashmir | 322771 | 317081 | 4392 |
| Jharkhand | 347440 | 342102 | 5130 |
| Karnataka | 2921049 | 2861499 | 36848 |
| Kerala | 3586693 | 3396184 | 18004 |
| Ladakh | 20411 | 20130 | 207 |

```
Lakshadweep                                10263    10165      51
Madhya Pradesh                            791980   781330   10514
Maharashtra                              6363442  6159676  134201
Manipur                                   105424    96776    1664
Meghalaya                                  69769    64157    1185
Mizoram                                    46320    33722     171
Nagaland                                   28811    26852     585
Odisha                                    988997   972710    6565
Puducherry                                121766   119115    1800
Punjab                                    599573   582791   16322
Rajasthan                                 953851   944700    8954
Sikkim                                     28018    25095     356
Tamil Nadu                               2579130  2524400   34367
Telangana                                 650353   638410    3831
Telengana                                 443360   362160    2312
Tripura                                    80660    77811     773
Unassigned                                    77        0       0
Uttar Pradesh                             1708812  1685492   22775
Uttarakhand                                342462   334650    7368
West Bengal                               1534999  1506532   18252
All                                       6363442  6159676  134201
```

```
"Top 10 states with most Active cases"

df_top_10 = df_india.nlargest(10,['Active'])

df_top_10 = df_india.groupby(['State/UnionTerritory'])
['Active'].max().sort_values(ascending=False).reset_index()
df_top = df_top_10.nlargest(10,['Active'])
df_top
```

```
  State/UnionTerritory  Active
0          Maharashtra  701614
1            Karnataka  605515
2               Kerala  445692
3           Tamil Nadu  313048
4        Uttar Pradesh  310783
5            Rajasthan  212753
6       Andhra Pradesh  211554
7              Gujarat  148297
8          West Bengal  132181
9         Chhattisgarh  131245
```

```
"Top 10 states with most deaths cases"

df_deaths_10 = df_india.nlargest(10,['Deaths'])

df_deaths_10 = df_india.groupby(['State/UnionTerritory'])
['Deaths'].max().sort_values(ascending=False).reset_index()
df_deaths = df_deaths_10.nlargest(10,['Deaths'])
```

```
df_deaths
```

```
  State/UnionTerritory  Deaths
0          Maharashtra  134201
1            Karnataka   36848
2           Tamil Nadu   34367
3                Delhi   25068
4        Uttar Pradesh   22775
5          West Bengal   18252
6               Kerala   18004
7               Punjab   16322
8       Andhra Pradesh   13564
9         Chhattisgarh   13544
```

```
"Finding recovery rate and deathrate"
```

```
statewise['Recoveryrate'] =
statewise['Cured']*100/statewise['Confirmed']
statewise['Deathrate'] =
statewise['Deaths']*100/statewise['Confirmed']
statewise
```

```
                                        Confirmed    Cured
Deaths  \
State/UnionTerritory

Andaman and Nicobar Islands                  7548     7412     129

Andhra Pradesh                            1985182  1952736   13564

Arunachal Pradesh                           50605    47821     248

Assam                                      576149   559684    5420

Bihar                                      725279   715352    9646

Cases being reassigned to states             9265        0       0

Chandigarh                                  61992    61150     811

Chhattisgarh                              1003356   988189   13544

Dadra and Nagar Haveli and Daman and Diu    10654    10646       4

Daman & Diu                                     2        0       0

Delhi                                     1436852  1411280   25068
```

| | | | |
|---|---|---|---|
| Goa | 172085 | 167978 | 3164 |
| Gujarat | 825085 | 814802 | 10077 |
| Haryana | 770114 | 759790 | 9652 |
| Himachal Pradesh | 208616 | 202761 | 3537 |
| Himanchal Pradesh | 204516 | 200040 | 3507 |
| Jammu and Kashmir | 322771 | 317081 | 4392 |
| Jharkhand | 347440 | 342102 | 5130 |
| Karnataka | 2921049 | 2861499 | 36848 |
| Kerala | 3586693 | 3396184 | 18004 |
| Ladakh | 20411 | 20130 | 207 |
| Lakshadweep | 10263 | 10165 | 51 |
| Madhya Pradesh | 791980 | 781330 | 10514 |
| Maharashtra | 6363442 | 6159676 | 134201 |
| Manipur | 105424 | 96776 | 1664 |
| Meghalaya | 69769 | 64157 | 1185 |
| Mizoram | 46320 | 33722 | 171 |
| Nagaland | 28811 | 26852 | 585 |
| Odisha | 988997 | 972710 | 6565 |
| Puducherry | 121766 | 119115 | 1800 |
| Punjab | 599573 | 582791 | 16322 |
| Rajasthan | 953851 | 944700 | 8954 |
| Sikkim | 28018 | 25095 | 356 |
| Tamil Nadu | 2579130 | 2524400 | 34367 |
| Telangana | 650353 | 638410 | 3831 |

| | | | |
|---|---|---|---|
| Telengana | 443360 | 362160 | 2312 |
| Tripura | 80660 | 77811 | 773 |
| Unassigned | 77 | 0 | 0 |
| Uttar Pradesh | 1708812 | 1685492 | 22775 |
| Uttarakhand | 342462 | 334650 | 7368 |
| West Bengal | 1534999 | 1506532 | 18252 |
| All | 6363442 | 6159676 | 134201 |

| State/UnionTerritory | Recoveryrate | Deathrate |
|---|---|---|
| Andaman and Nicobar Islands | 98.198198 | 1.709062 |
| Andhra Pradesh | 98.365591 | 0.683262 |
| Arunachal Pradesh | 94.498567 | 0.490070 |
| Assam | 97.142232 | 0.940729 |
| Bihar | 98.631285 | 1.329971 |
| Cases being reassigned to states | 0.000000 | 0.000000 |
| Chandigarh | 98.641760 | 1.308233 |
| Chhattisgarh | 98.488373 | 1.349870 |
| Dadra and Nagar Haveli and Daman and Diu | 99.924911 | 0.037545 |
| Daman & Diu | 0.000000 | 0.000000 |
| Delhi | 98.220276 | 1.744647 |
| Goa | 97.613389 | 1.838626 |
| Gujarat | 98.753704 | 1.221329 |
| Haryana | 98.659419 | 1.253321 |
| Himachal Pradesh | 97.193408 | 1.695460 |
| Himanchal Pradesh | 97.811418 | 1.714780 |
| Jammu and Kashmir | 98.237140 | 1.360717 |
| Jharkhand | 98.463620 | 1.476514 |
| Karnataka | 97.961349 | 1.261465 |
| Kerala | 94.688450 | 0.501967 |
| Ladakh | 98.623291 | 1.014159 |
| Lakshadweep | 99.045114 | 0.496931 |
| Madhya Pradesh | 98.655269 | 1.327559 |
| Maharashtra | 96.797865 | 2.108937 |
| Manipur | 91.796934 | 1.578388 |
| Meghalaya | 91.956313 | 1.698462 |
| Mizoram | 72.802245 | 0.369171 |
| Nagaland | 93.200514 | 2.030474 |
| Odisha | 98.353180 | 0.663804 |
| Puducherry | 97.822873 | 1.478245 |
| Punjab | 97.201008 | 2.722271 |
| Rajasthan | 99.040626 | 0.938721 |

```
Sikkim                                          89.567421   1.270612
Tamil Nadu                                      97.877967   1.332504
Telangana                                       98.163613   0.589065
Telengana                                       81.685312   0.521472
Tripura                                         96.467890   0.958344
Unassigned                                       0.000000   0.000000
Uttar Pradesh                                   98.635309   1.332797
Uttarakhand                                     97.718871   2.151480
West Bengal                                     98.145471   1.189056
All                                             96.797865   2.108937
```

```
"Correlation amongs the columns"

statewise.corr()
```

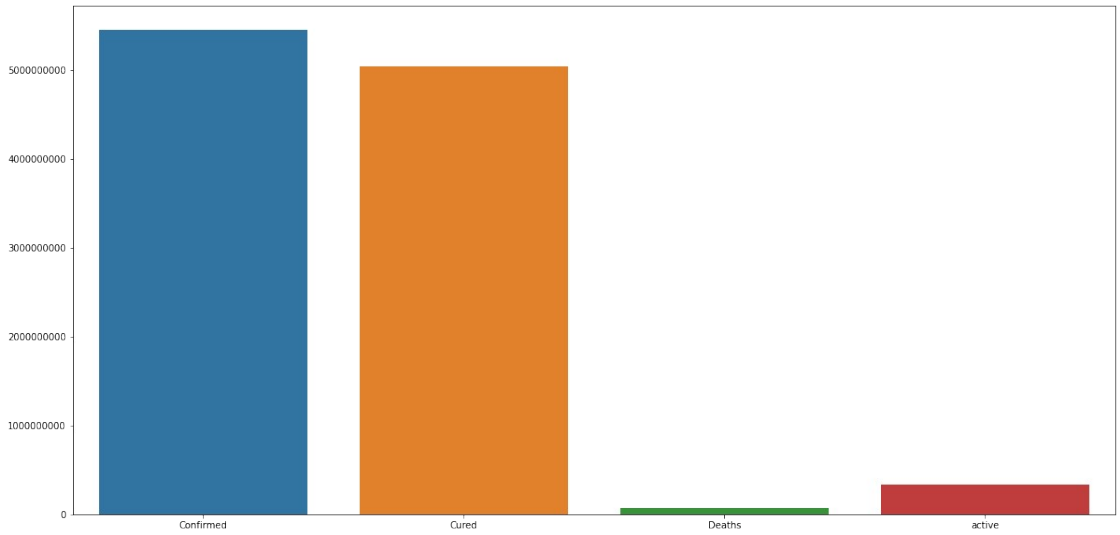
```
              Confirmed     Cured    Deaths  Recoveryrate  Deathrate
Confirmed      1.000000  0.999902  0.939570      0.195490   0.268146
Cured          0.999902  1.000000  0.940407      0.197970   0.271160
Deaths         0.939570  0.940407  1.000000      0.150257   0.381246
Recoveryrate   0.195490  0.197970  0.150257      1.000000   0.542027
Deathrate      0.268146  0.271160  0.381246      0.542027   1.000000
```

## Data visualization

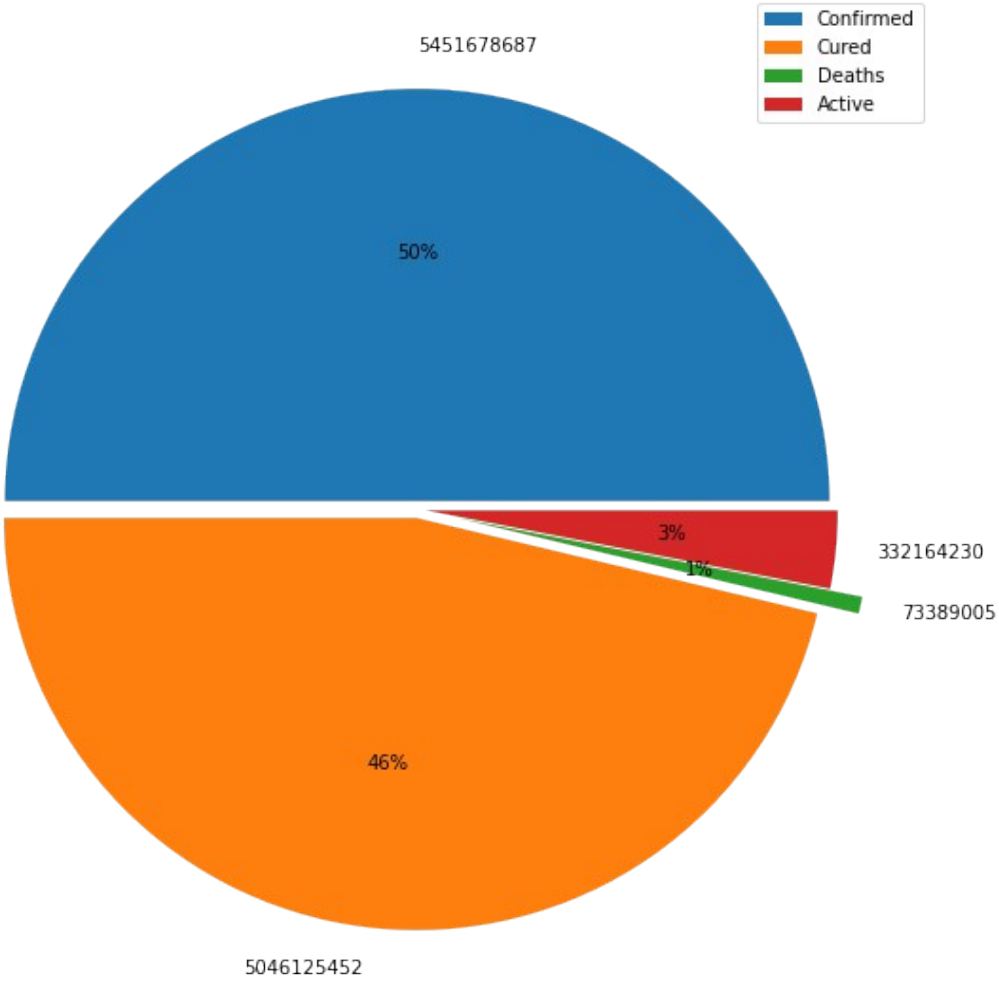
```
"""Barplot for Confirmed , Deaths , Cured , Active"""
fig = plt.figure(figsize=(20,10))

confirm= df_india['Confirmed'].sum()
cured = df_india['Cured'].sum()
deaths= df_india['Deaths'].sum()
active= df_india['Active'].sum()

print('Total Confirmed cases =',confirm)
print('Total Cured cases =',cured)
print('Total Active cases =',active)
print('Total Death cases =',deaths)

barplot =
sns.barplot(x=['Confirmed','Cured','Deaths','active'],y=[confirm,cured
,deaths,active])
barplot.set_yticklabels(labels=(barplot.get_yticks()*1).astype(int))

plt.show()
```

```
Total Confirmed cases = 5451678687
Total Cured cases = 5046125452
Total Active cases = 332164230
Total Death cases = 73389005

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:15:
UserWarning: FixedFormatter should only be used together with
```

```
FixedLocator
  from ipykernel import kernelapp as app
```

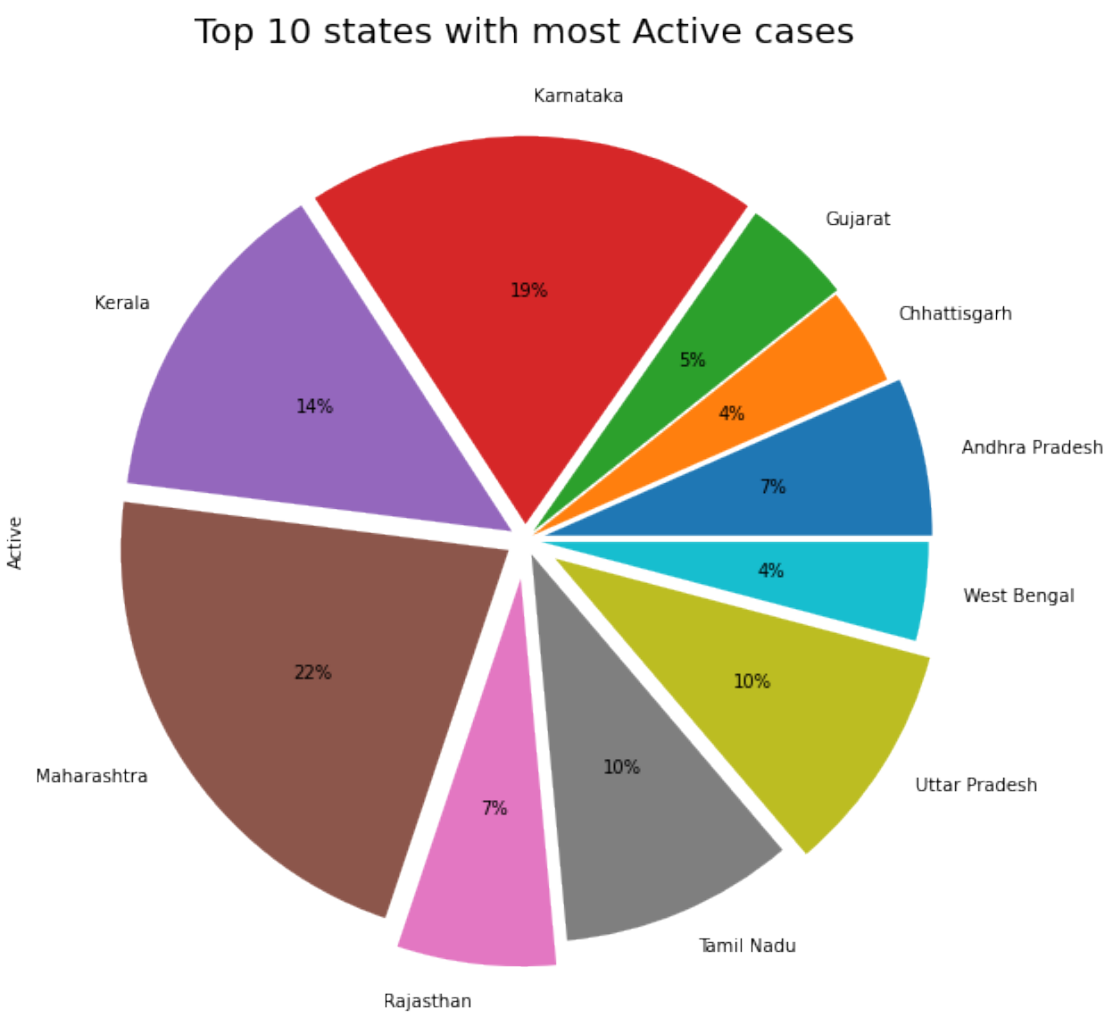



```
"Piechart for 'Confirmed','Cured',Deaths & 'Active'"

fig = plt.figure(figsize=(17,10))
df_values =
[df_india['Confirmed'].sum(),df_india['Cured'].sum(),df_india['Deaths'
].sum(),df_india['Active'].sum()]
df_keys = [confirm,cured,deaths,active]


plt.pie(df_keys,labels = df_keys, explode = (0.02,0.02,0.1,0.02),
autopct = '%.0f%%')
plt.legend(['Confirmed','Cured','Deaths','Active'])

<matplotlib.legend.Legend at 0x793cbed62210>
```



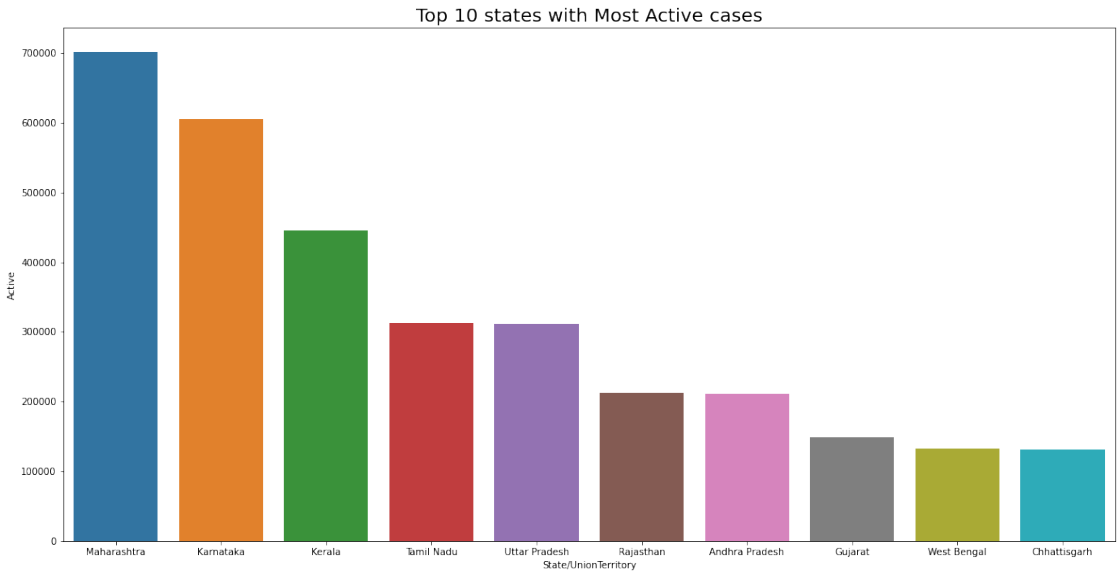

*"Pie Chart Of 10 Top states with Active Cases"*

```
fig = plt.figure(figsize=(17,10))
df_top.groupby(["State/UnionTerritory"]).sum()
["Active"].plot(kind='pie',rot=90,explode=(0.05,0.02,0.03,0.04,0.04,0.
05,0.1,0.04,0.09,0.04),autopct='%1.0f%%')
plt.title('Top 10 states with most Active cases',size=20)
plt.show()
```



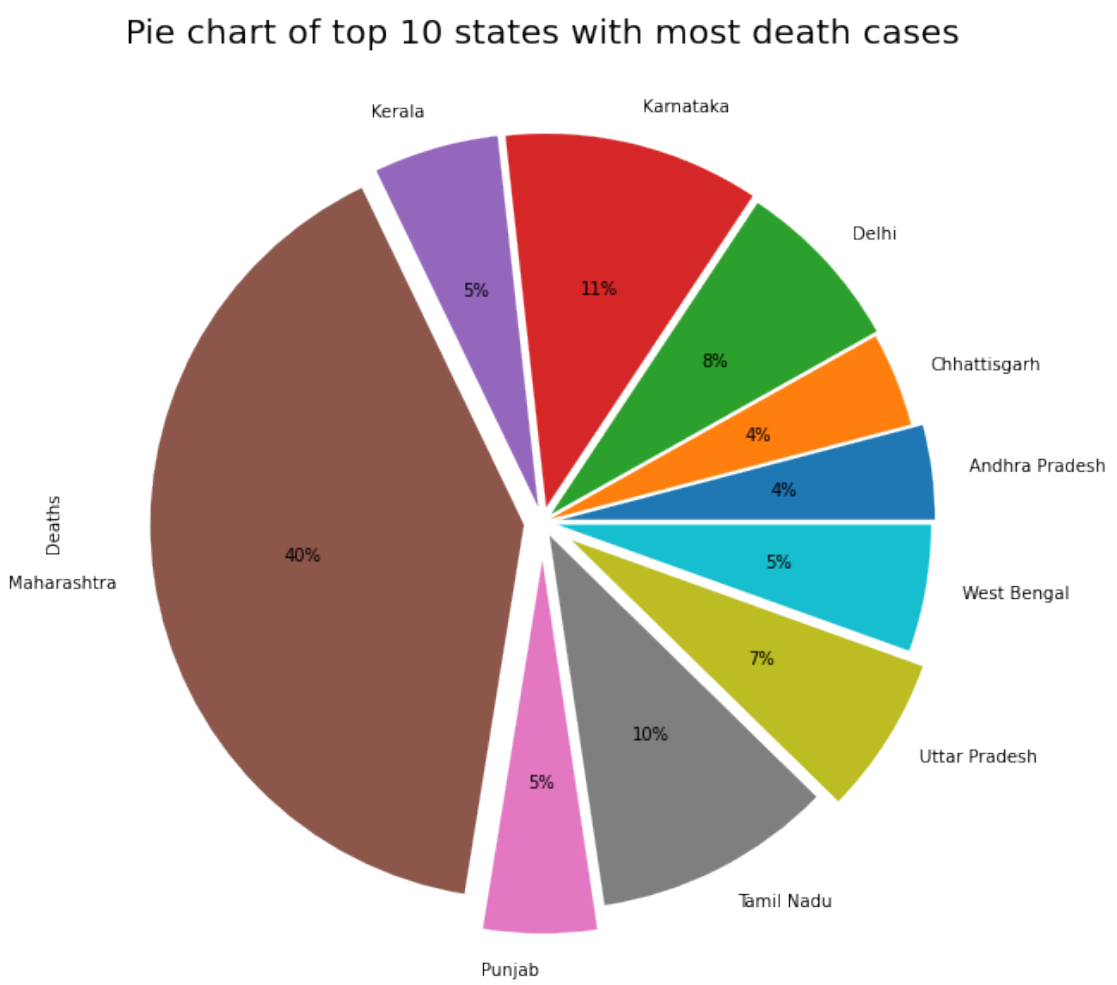

*"Bar Plot Of Top 10 Active Cases"*

```
fig = plt.figure(figsize=(20,10))
sns.barplot(data =
df_top.iloc[:10],y='Active',x='State/UnionTerritory')
plt.title('Top 10 states with Most Active cases', size=20)
plt.show()
```



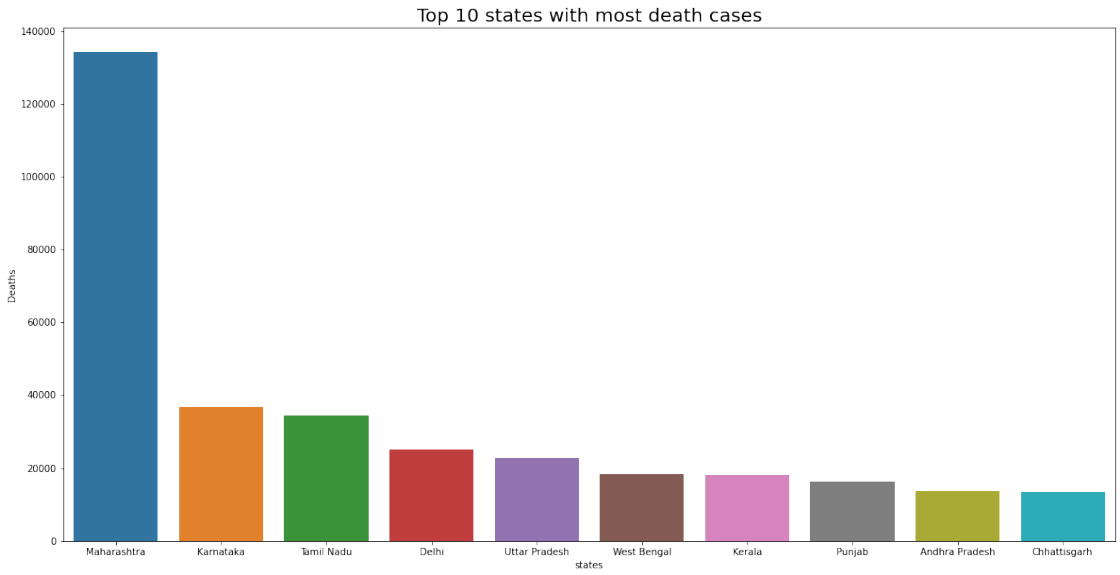

**Note : As per above visual's it is clear that Maharashtra has maximum number of Active cases wheras Chhattisgarh has the least number of Active cases.**

```
"""Pie chart of top 10 states with most death cases"""

fig = plt.figure(figsize=(17,10))
df_deaths.groupby(["State/UnionTerritory"]).sum()
["Deaths"].plot(kind='pie',rot=90,explode=(0.05,0.02,0.03,0.04,0.04,0.
05,0.1,0.04,0.09,0.04),autopct='%1.0f%%')
plt.title("Pie chart of top 10 states with most death cases",size =
20)

Text(0.5, 1.0, 'Pie chart of top 10 states with most death cases')
```




*"Bar graph with top 10 states with most Death cases"*

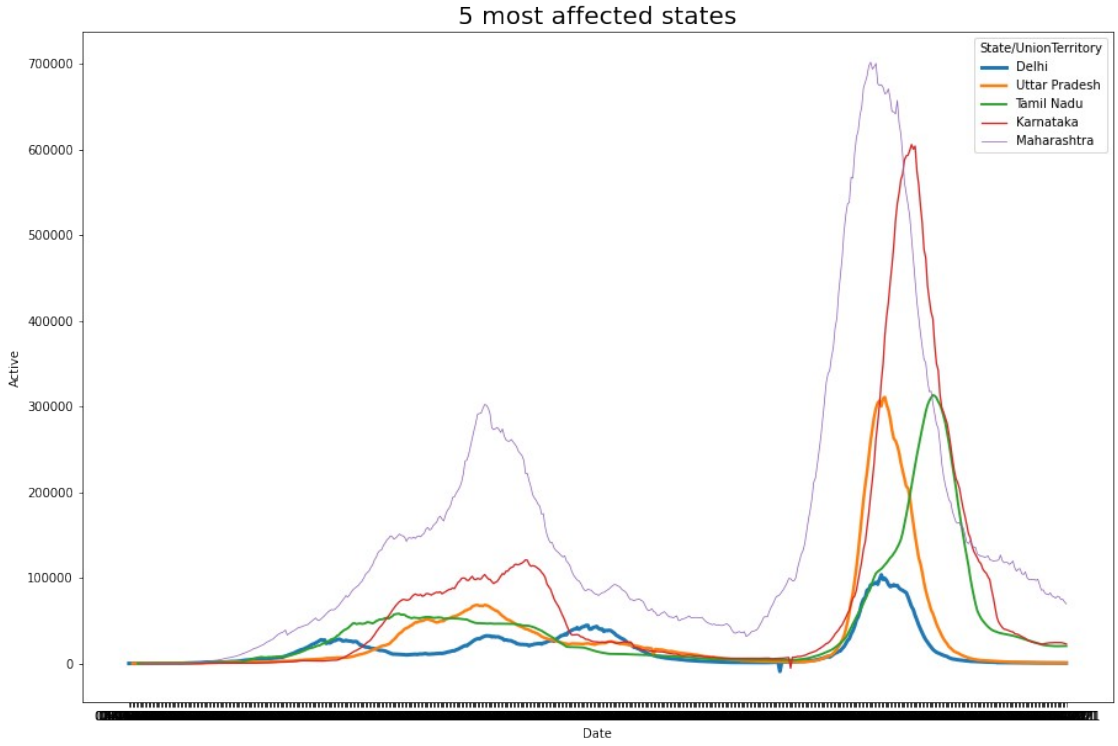
```
df_deaths = df_india.groupby('State/UnionTerritory').max()
[['Deaths','Date']].sort_values(by='Deaths',ascending=False).reset_ind
ex()
fig = plt.figure(figsize=(20,10))
plot1 = sns.barplot(data =
df_deaths.iloc[:10],y='Deaths',x='State/UnionTerritory')
plt.title('Top 10 states with most death cases', size=20)
plt.xlabel('states')
plt.ylabel('Deaths')
plt.show()
```




**Note : As per above visual's it is clear that Maharashtra has most number of death cases and Chhattisgarh has least number of death cases.**

*" Top 5 Most affected states"*

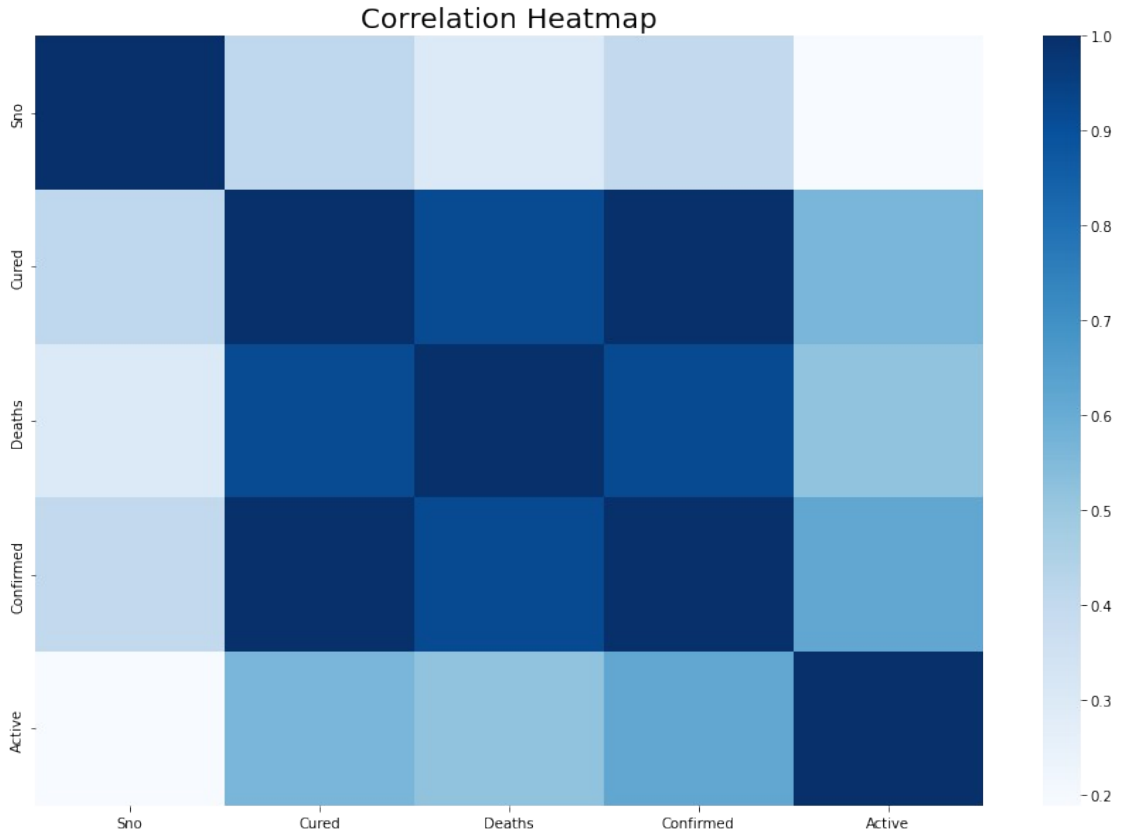
```
fig = plt.figure(figsize=(15,10))
plot = sns.lineplot(data =
df_india[df_india['State/UnionTerritory'].isin(['Maharashtra','Karnataka','Tamil Nadu','Delhi','Uttar Pradesh'])],x='Date',y='Active',hue =
'State/UnionTerritory',size='State/UnionTerritory')
plt.title('5 most affected states',size=20)
plt.show()
```




## correlation Heatmap

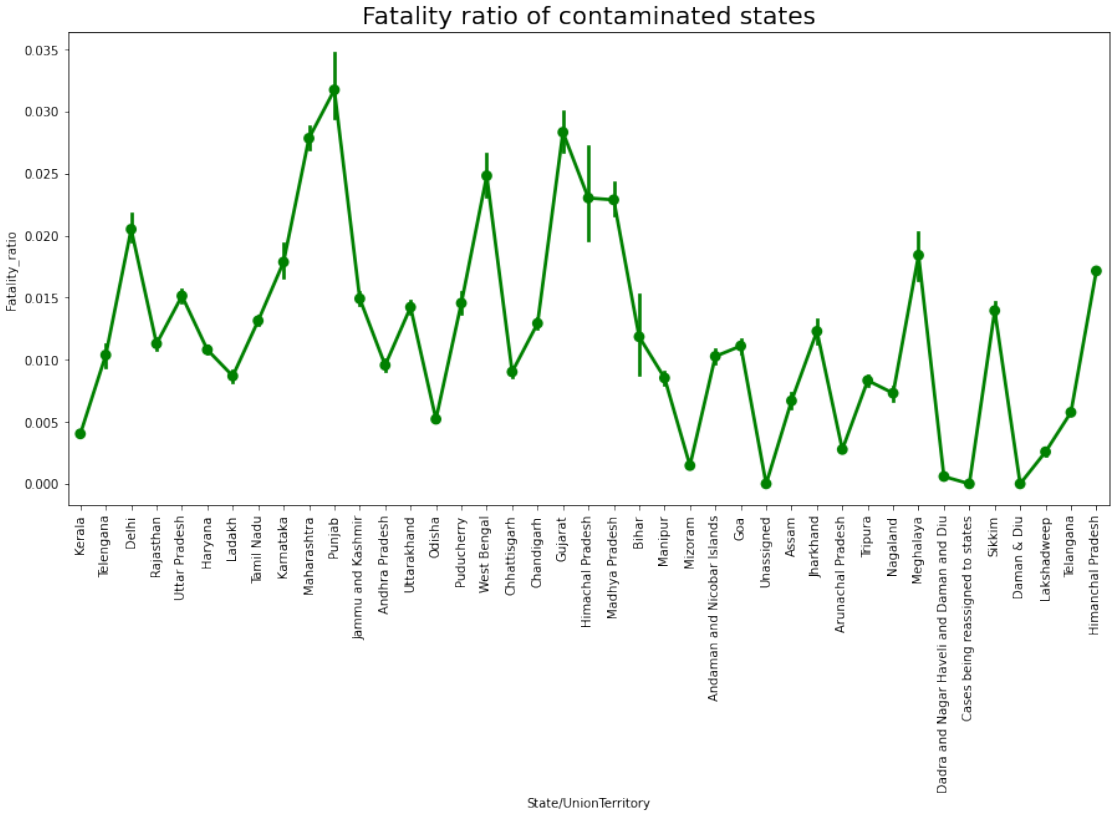
*"Correlation Heatmap"*

```
fig = plt.figure(figsize=(15,10))
sns.heatmap(df_india.corr(),cmap="Blues")
plt.title('Correlation Heatmap',size=20)
plt.show()
```




*"Fatality ratio of contaminated states"*

```
df_india['Fatality_ratio']= df_india['Deaths']/df_india['Confirmed']
a4_dims = (15,7)
fig,ax = plt.subplots(figsize=a4_dims)
sns.pointplot(data =
df_india,x='State/UnionTerritory',y='Fatality_ratio',ax=ax,color='Gree
n')
plt.xticks(rotation=90)
plt.title('Fatality ratio of contaminated states',size=20)
plt.show()
```


Fatality ratio of contaminated states
Fatality_ratio
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Kerala
Telengana
Delhi
Rajasthan
Uttar Pradesh
Haryana
Ladakh
Tamil Nadu
Karnataka
Maharashtra
Punjab
Jammu and Kashmir
Andhra Pradesh
Uttarakhand
Odisha
Puducherry
West Bengal
Chhattisgarh
Chandigarh
Gujarat
Himachal Pradesh
Madhya Pradesh
Bihar
Manipur
Mizoram
Andaman and Nicobar Islands
Goa
Unassigned
Assam
Jharkhand
Arunachal Pradesh
Tripura
Nagaland
Meghalaya
Dadra and Nagar Haveli and Daman and Diu
Cases being reassigned to states
Sikkim
Daman & Diu
Lakshadweep
Telangana
Himanchal Pradesh
State/UnionTerritory

## 5.Model Evaluation (Result)

Our analysis shows that India has made significant progress in vaccinating its population against COVID-19. However, there is still a long way to go, especially in states with low vaccination rates.

The insights from this analysis can be used to guide the allocation of resources and to help accelerate the vaccination drive in the country.

# 6. Conclusion

In conclusion, we have analyzed the COVID-19 vaccination data in India and performed various analytics on the given dataset. We have calculated the total number of vaccinations administered, the vaccination rate, and the total doses administered by vaccine type. We have also visualized the state-wise vaccination rates and the progression of vaccinations in the country.

These insights can help policymakers and healthcare professionals in India to make informed decisions regarding the vaccination drive and the allocation of resources.