COVID-19 Database Analysis

In [1]:

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

Importing Database

In [2]:

```
df = pd.read_csv('./covid 19 data.csv')
```

In [3]:

```
df.head(5)
```

Out[3]:

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
0	1	30/01/20	6:00 PM	Kerala	1	0	0	0	1
1	2	31/01/20	6:00 PM	Kerala	1	0	0	0	1
2	3	01/02/20	6:00 PM	Kerala	2	0	0	0	2
3	4	02/02/20	6:00 PM	Kerala	3	0	0	0	3
4	5	03/02/20	6:00 PM	Kerala	3	0	0	0	3

In [4]:

```
df.tail(5)
```

Out[4]:

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirm
9286	9287	09/12/20	8:00 AM	Telengana	-	-	266120	1480	275
9287	9288	09/12/20	8:00 AM	Tripura	-	-	32169	373	32
9288	9289	09/12/20	8:00 AM	Uttarakhand	-	-	72435	1307	79
9289	9290	09/12/20	8:00 AM	Uttar Pradesh	-	-	528832	7967	558
9290	9291	09/12/20	8:00 AM	West Bengal	-	-	475425	8820	507

Dropping Un-neccesary Columns

In [5]:

```
df = df.drop(['Sno','Time','ConfirmedIndianNational','ConfirmedForeignNational'],axis=1)
```

In [6]:

```
df = df.rename(columns={'State/UnionTerritory':'State'})
```

In [7]:

```
df
```

Out[7]:

	Date	State	Cured	Deaths	Confirmed
0	30/01/20	Kerala	0	0	1
1	31/01/20	Kerala	0	0	1
2	01/02/20	Kerala	0	0	2
3	02/02/20	Kerala	0	0	3
4	03/02/20	Kerala	0	0	3
...	...	...	...	...	...
9286	09/12/20	Telengana	266120	1480	275261
9287	09/12/20	Tripura	32169	373	32945
9288	09/12/20	Uttarakhand	72435	1307	79141
9289	09/12/20	Uttar Pradesh	528832	7967	558173
9290	09/12/20	West Bengal	475425	8820	507995

9291 rows × 5 columns

In [8]:

```
df['Date'] = pd.to_datetime(df['Date'], dayfirst = True)
```

In [9]:

```
df
```

Out[9]:

	Date	State	Cured	Deaths	Confirmed
0	2020-01-30	Kerala	0	0	1
1	2020-01-31	Kerala	0	0	1
2	2020-02-01	Kerala	0	0	2
3	2020-02-02	Kerala	0	0	3
4	2020-02-03	Kerala	0	0	3
...	...	...	...	...	...
9286	2020-12-09	Telengana	266120	1480	275261
9287	2020-12-09	Tripura	32169	373	32945
9288	2020-12-09	Uttarakhand	72435	1307	79141
9289	2020-12-09	Uttar Pradesh	528832	7967	558173
9290	2020-12-09	West Bengal	475425	8820	507995

9291 rows × 5 columns

Extracing States Names from database

In [10]:

```
states = df['State'].unique()
```

In [11]:

```
states
```

Out[11]:

```
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', 'Dadar Nagar Haveli',
       'Cases being reassigned to states', 'Sikkim', 'Daman & Diu',
       'Dadra and Nagar Haveli and Daman and Diu', 'Telangana',
       'Telangana***', 'Telengana***', 'Maharashtra***', 'Chandigarh***',
       'Punjab***'], dtype=object)
```

Renaming Mis-spelled State names

In [12]:

```
df=df.replace('Telengana','Telangana')
df=df.replace('Telengana***','Telangana')
df=df.replace('Telangana***','Telangana')
df=df.replace('Maharashtra***','Maharashtra')
df=df.replace('Chandigarh***','Chandigarh')
df=df.replace('Punjab***','Punjab')
```

In [13]:

```
states = df['State'].unique()
```

In [14]:

```
states = list(states)    # Numpy array to python list
```

Removing Un-necessary entries from states list

Daman & Diu is removed because of lack of availability of data ---> database contains only 1 entry coresponding to Daman & Diu

In [15]:

```
df[df.State == 'Daman & Diu']
```

Out[15]:

	Date	State	Cured	Deaths	Confirmed
2890	**2020-06-11**	**Daman & Diu**	**0**	**0**	**2**

In [16]:

```
states.remove('Cases being reassigned to states')
states.remove('Unassigned')
states.remove('Daman & Diu')
```

In [17]:

```
states
```

Out[17]:

```
['Kerala',
 'Telangana',
 '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',
 'Assam',
 'Jharkhand',
 'Arunachal Pradesh',
 'Tripura',
 'Nagaland',
 'Meghalaya',
 'Dadar Nagar Haveli',
 'Sikkim',
 'Dadra and Nagar Haveli and Daman and Diu']
```

In [18]:

```
df.isnull().sum()
```

Out[18]:

```
Date          0
State         0
Cured         0
Deaths        0
Confirmed     0
dtype: int64
```

Adding Months column into database and removing Dates column

In [19]:

```
df['Month'] = pd.DatetimeIndex(df['Date']).month
```

In [20]:

```
df
```

Out[20]:

	Date	State	Cured	Deaths	Confirmed	Month
0	**2020-01-30**	**Kerala**	**0**	**0**	**1**	**1**
1	**2020-01-31**	**Kerala**	**0**	**0**	**1**	**1**

	Date	State	Cured	Deaths	Confirmed	Month
2	2020-02-01	Kerala	0	0	2	2
3	2020-02-02	Kerala	0	0	3	2
4	2020-02-03	Kerala	0	0	3	2
...	...	...	...	...	...	...
9286	2020-12-09	Telangana	266120	1480	275261	12
9287	2020-12-09	Tripura	32169	373	32945	12
9288	2020-12-09	Uttarakhand	72435	1307	79141	12
9289	2020-12-09	Uttar Pradesh	528832	7967	558173	12
9290	2020-12-09	West Bengal	475425	8820	507995	12

9291 rows × 6 columns

In [21]:

```
df.drop('Date',axis=1,inplace=True)
```

In [22]:

```
df
```

Out[22]:

	State	Cured	Deaths	Confirmed	Month
0	Kerala	0	0	1	1
1	Kerala	0	0	1	1
2	Kerala	0	0	2	2
3	Kerala	0	0	3	2
4	Kerala	0	0	3	2
...	...	...	...	...	...
9286	Telangana	266120	1480	275261	12
9287	Tripura	32169	373	32945	12
9288	Uttarakhand	72435	1307	79141	12
9289	Uttar Pradesh	528832	7967	558173	12
9290	West Bengal	475425	8820	507995	12

9291 rows × 5 columns

Creating a dictonary with keys as state names and their corresponding data

In [23]:

```
state_dct_df ={}
for i in states:
    state_dct_df[i] = df[df.State == i].copy().reset_index().drop(['index'],axis=1)
```

In [24]:

```
state_dct_df['Maharashtra']
```

Out[24]:

	State	Cured	Deaths	Confirmed	Month
0	Maharashtra	0	0	2	3
1	Maharashtra	0	0	5	3

	State	Cured	Deaths	Confirmed	Month
2	Maharashtra	0	0	2	3
3	Maharashtra	0	0	11	3
4	Maharashtra	0	0	14	3
...	...	...	...	...	...
271	Maharashtra	1710050	47599	1842587	12
272	Maharashtra	1715884	47694	1847509	12
273	Maharashtra	1723370	47734	1852266	12
274	Maharashtra	1730715	47774	1855341	12
275	Maharashtra	1737080	47827	1859367	12

276 rows × 5 columns

Creating a Dictonary to Keep count of number of entries in database corresponding to each state

In [25]:

```
row_count_dct = {}
for i in states:
    rows, columns = state_dct_df[i].shape
    row_count_dct[i] = rows
```

In [26]:

```
row_count_dct
```

Out[26]:

```
{'Kerala': 315,
 'Telangana': 283,
 'Delhi': 283,
 'Rajasthan': 282,
 'Uttar Pradesh': 281,
 'Haryana': 281,
 'Ladakh': 278,
 'Tamil Nadu': 278,
 'Karnataka': 276,
 'Maharashtra': 276,
 'Punjab': 276,
 'Jammu and Kashmir': 276,
 'Andhra Pradesh': 273,
 'Uttarakhand': 270,
 'Odisha': 269,
 'Puducherry': 267,
 'West Bengal': 267,
 'Chhattisgarh': 266,
 'Chandigarh': 266,
 'Gujarat': 265,
 'Himachal Pradesh': 264,
 'Madhya Pradesh': 264,
 'Bihar': 263,
 'Manipur': 261,
 'Mizoram': 260,
 'Andaman and Nicobar Islands': 259,
 'Goa': 259,
 'Assam': 253,
 'Jharkhand': 253,
 'Arunachal Pradesh': 251,
 'Tripura': 247,
 'Nagaland': 207,
 'Meghalaya': 240,
 'Dadar Nagar Haveli': 37,
 'Sikkim': 200,
 'Dadra and Nagar Haveli and Daman and Diu': 181}
```

Getting Back Original values of cured, deaths and confirmed from cumulative sum

In [27]:

```
state_dct_df['Maharashtra']
```

Out[27]:

	State	Cured	Deaths	Confirmed	Month
0	Maharashtra	0	0	2	3
1	Maharashtra	0	0	5	3
2	Maharashtra	0	0	2	3
3	Maharashtra	0	0	11	3
4	Maharashtra	0	0	14	3
...	...	...	...	...	...
271	Maharashtra	1710050	47599	1842587	12
272	Maharashtra	1715884	47694	1847509	12
273	Maharashtra	1723370	47734	1852266	12
274	Maharashtra	1730715	47774	1855341	12
275	Maharashtra	1737080	47827	1859367	12

276 rows × 5 columns

In [28]:

```
for s in states:
    for i in range(row_count_dct[s]-1,0,-1):
            state_dct_df[s].iloc[i,1] -= state_dct_df[s].iloc[i-1,1]
            state_dct_df[s].iloc[i,2] -= state_dct_df[s].iloc[i-1,2]
            state_dct_df[s].iloc[i,3] -= state_dct_df[s].iloc[i-1,3]
```

In [29]:

```
state_dct_df['Maharashtra']
```

Out[29]:

	State	Cured	Deaths	Confirmed	Month
0	Maharashtra	0	0	2	3
1	Maharashtra	0	0	3	3
2	Maharashtra	0	0	-3	3
3	Maharashtra	0	0	9	3
4	Maharashtra	0	0	3	3
...	...	...	...	...	...
271	Maharashtra	6776	127	5229	12
272	Maharashtra	5834	95	4922	12
273	Maharashtra	7486	40	4757	12
274	Maharashtra	7345	40	3075	12
275	Maharashtra	6365	53	4026	12

276 rows × 5 columns

As some of the entries in database turns out to be negative, now we will get back there indices and aftwerwards remove those entries

Negative entries can occur in cured, deaths and confirmed columns

In [30]:

```
check_neg_cured = {}
check_neg_deaths = {}
check_neg_confirmed = {}
for s in states:
    index_cured = []
    index_deaths = []
    index_confirmed = []

    for i in range(row_count_dct[s]):
        if(state_dct_df[s].iloc[i,1]<0):
            index_cured.append(i)
        if(state_dct_df[s].iloc[i,2]<0):
            index_deaths.append(i)
        if(state_dct_df[s].iloc[i,3]<0):
            index_confirmed.append(i)

    check_neg_cured[s] = index_cured
    check_neg_deaths[s] = index_deaths
    check_neg_confirmed[s] = index_confirmed
```

In [31]:

```
check_neg_cured
```

Out[31]:

```
{'Kerala': [],
 'Telangana': [],
 'Delhi': [],
 'Rajasthan': [],
 'Uttar Pradesh': [11],
 'Haryana': [],
 'Ladakh': [],
 'Tamil Nadu': [],
 'Karnataka': [],
 'Maharashtra': [],
 'Punjab': [],
 'Jammu and Kashmir': [],
 'Andhra Pradesh': [],
 'Uttarakhand': [],
 'Odisha': [],
 'Puducherry': [],
 'West Bengal': [16],
 'Chhattisgarh': [],
 'Chandigarh': [],
 'Gujarat': [],
 'Himachal Pradesh': [],
 'Madhya Pradesh': [],
 'Bihar': [],
 'Manipur': [],
 'Mizoram': [],
 'Andaman and Nicobar Islands': [],
 'Goa': [],
 'Assam': [],
 'Jharkhand': [],
 'Arunachal Pradesh': [],
 'Tripura': [],
 'Nagaland': [],
 'Meghalaya': [],
 'Dadar Nagar Haveli': [],
 'Sikkim': [],
```

```
 'Dadra and Nagar Haveli and Daman and Diu': []}
```

In [ ]:

```
```

Removing all the entries which are invalid/negative for calculation from Statewise database dictonary

In [32]:

```
for i in states:
    state_dct_df[i] = df[df.State == i].copy().reset_index().drop(['index'],axis=1)
```

In [33]:

```
for s in states:
    state_dct_df[s].drop(check_neg_cured[s],axis=0,inplace=True)
    state_dct_df[s].drop(check_neg_deaths[s],axis=0,inplace=True)
    state_dct_df[s].drop(check_neg_confirmed[s],axis=0,inplace=True)
```

In [34]:

```
state_dct_df['Maharashtra']
```

Out[34]:

	State	Cured	Deaths	Confirmed	Month
0	Maharashtra	0	0	2	3
1	Maharashtra	0	0	5	3
3	Maharashtra	0	0	11	3
4	Maharashtra	0	0	14	3
5	Maharashtra	0	0	14	3
...	...	...	...	...	...
271	Maharashtra	1710050	47599	1842587	12
272	Maharashtra	1715884	47694	1847509	12
273	Maharashtra	1723370	47734	1852266	12
274	Maharashtra	1730715	47774	1855341	12
275	Maharashtra	1737080	47827	1859367	12

274 rows × 5 columns

Calculating Updated row count for each state and putting it in dictonary

In [35]:

```
for i in states:
    rows, columns = state_dct_df[i].shape
    row_count_dct[i] = rows
```

In [36]:

```
row_count_dct
```

Out[36]:

```
{'Kerala': 315,
 'Telangana': 283,
 'Delhi': 283
```

```
 'Rajasthan': 281,
 'Uttar Pradesh': 280,
 'Haryana': 281,
 'Ladakh': 278,
 'Tamil Nadu': 278,
 'Karnataka': 276,
 'Maharashtra': 274,
 'Punjab': 276,
 'Jammu and Kashmir': 276,
 'Andhra Pradesh': 273,
 'Uttarakhand': 270,
 'Odisha': 269,
 'Puducherry': 266,
 'West Bengal': 266,
 'Chhattisgarh': 266,
 'Chandigarh': 266,
 'Gujarat': 265,
 'Himachal Pradesh': 264,
 'Madhya Pradesh': 264,
 'Bihar': 263,
 'Manipur': 261,
 'Mizoram': 260,
 'Andaman and Nicobar Islands': 259,
 'Goa': 259,
 'Assam': 253,
 'Jharkhand': 252,
 'Arunachal Pradesh': 251,
 'Tripura': 247,
 'Nagaland': 206,
 'Meghalaya': 240,
 'Dadar Nagar Haveli': 37,
 'Sikkim': 200,
 'Dadra and Nagar Haveli and Daman and Diu': 181}
```

Again Calculating Absolute values from cumulative sum

In [37]:

```
for s in states:
    for i in range(row_count_dct[s]-1,0,-1):
            state_dct_df[s].iloc[i,1] -= state_dct_df[s].iloc[i-1,1]
            state_dct_df[s].iloc[i,2] -= state_dct_df[s].iloc[i-1,2]
            state_dct_df[s].iloc[i,3] -= state_dct_df[s].iloc[i-1,3]
```

Summing confirmed cases monthwise

In [38]:

```
months = np.arange(1,13,1)
months
```

Out[38]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

In [39]:

```
state_dct_df['Maharashtra'].head()
```

Out[39]:

	State	Cured	Deaths	Confirmed	Month
0	**Maharashtra**	**0**	**0**	**2**	**3**
1	**Maharashtra**	**0**	**0**	**3**	**3**
3	**Maharashtra**	**0**	**0**	**6**	**3**

	State	Cured	Deaths	Confirmed	Month
4	Maharashtra	0	0	3	3
5	Maharashtra	0	0	0	3

In [40]:

```
month_wise_sum_of_confirmed_cases_dct={}
for i in months:
    month_wise_sum_of_confirmed_cases_dct[i] = 0
for s in states:
    for i in range(row_count_dct[s]):
        month_wise_sum_of_confirmed_cases_dct[state_dct_df[s].iloc[i,4]] +=  state_dct_d
f[s].iloc[i,3]
```

In [41]:

```
month_wise_sum_of_confirmed_cases_dct
```

Out[41]:

```
{1: 1,
 2: 2,
 3: 1356,
 4: 31971,
 5: 143322,
 6: 383210,
 7: 1079034,
 8: 1982375,
 9: 2604518,
 10: 1911356,
 11: 1294572,
 12: 304159}
```

Q1.Filter the month in which heighest people are get infected to Covid-19 virus?

In [42]:

```
peak_month = 0
peak_val = 0
for i in months:
    if month_wise_sum_of_confirmed_cases_dct[i]>peak_val:
        peak_val = month_wise_sum_of_confirmed_cases_dct[i]
        peak_month = i
```

In [43]:

```
print(f"Peak Month = {peak_month}\nPeak Cases = {peak_val}")
```

```
Peak Month = 9
Peak Cases = 2604518
```

Finding total number months for which data is available for each state

Used for rate calculations

In [44]:

```
total_months_dct = {}
for s in states:
    total_months_dct[s] = len(state_dct_df[s]['Month'].unique())
```

In [45]:

```
total_months_dct
```

Out[45]:

```
{'Kerala': 12,
 'Telangana': 10,
 'Delhi': 10,
 'Rajasthan': 10,
 'Uttar Pradesh': 10,
 'Haryana': 10,
 'Ladakh': 10,
 'Tamil Nadu': 10,
 'Karnataka': 10,
 'Maharashtra': 10,
 'Punjab': 10,
 'Jammu and Kashmir': 10,
 'Andhra Pradesh': 10,
 'Uttarakhand': 10,
 'Odisha': 10,
 'Puducherry': 10,
 'West Bengal': 10,
 'Chhattisgarh': 10,
 'Chandigarh': 10,
 'Gujarat': 10,
 'Himachal Pradesh': 10,
 'Madhya Pradesh': 10,
 'Bihar': 10,
 'Manipur': 10,
 'Mizoram': 10,
 'Andaman and Nicobar Islands': 10,
 'Goa': 10,
 'Assam': 9,
 'Jharkhand': 9,
 'Arunachal Pradesh': 9,
 'Tripura': 9,
 'Nagaland': 9,
 'Meghalaya': 9,
 'Dadar Nagar Haveli': 2,
 'Sikkim': 8,
 'Dadra and Nagar Haveli and Daman and Diu': 7}
```

Survival Rate = Total Cured/(Total Confirmed*Total number of months for which data is available)

Q2.Obtain state in which survival rate is high

In [46]:

```
survival_rate ={}
for s in states:
    total_confirmed = 0
    total_cured = 0
    for i in range(row_count_dct[s]):
       total_confirmed += state_dct_df[s].iloc[i,3]
       total_cured += state_dct_df[s].iloc[i,1]
    survival_rate[s] = (total_cured/(total_confirmed*total_months_dct[s]))
```

In [47]:

```
survival_rate
```

Out[47]:

```
{'Kerala': 0.07527462555995384,
 'Telangana': 0.09667915178684958,
 'Delhi': 0.09462864588217956,
 'Rajasthan': 0.09178398963803516,
 'Uttar Pradesh': 0.09474338601114708,
 'Haryana': 0.09409313318117878,
 'Ladakh': 0.08982049280856283,
 'Tamil Nadu': 0.09717326700202324,
```

```
 'Karnataka': 0.09587683908123008,
 'Maharashtra': 0.09342319187121209,
 'Punjab': 0.0922214948103044,
 'Jammu and Kashmir': 0.09407565899086269,
 'Andhra Pradesh': 0.09857121416435334,
 'Uttarakhand': 0.09152651596517608,
 'Odisha': 0.09846449195900756,
 'Puducherry': 0.09731178472836428,
 'West Bengal': 0.09358851957204303,
 'Chhattisgarh': 0.09097273116832666,
 'Chandigarh': 0.09310269203355447,
 'Gujarat': 0.09170086639306886,
 'Himachal Pradesh': 0.08197008722754918,
 'Madhya Pradesh': 0.09234337465830963,
 'Bihar': 0.09717577155464187,
 'Manipur': 0.08776329746931354,
 'Mizoram': 0.09484536082474226,
 'Andaman and Nicobar Islands': 0.09725826705734616,
 'Goa': 0.0958904669459487,
 'Assam': 0.10873749007746457,
 'Jharkhand': 0.10835841490493105,
 'Arunachal Pradesh': 0.10606152785382572,
 'Tripura': 0.10849395457074923,
 'Nagaland': 0.1043548121690817,
 'Meghalaya': 0.10462888351687707,
 'Dadar Nagar Haveli': 0.038461538461538464,
 'Sikkim': 0.11349472674976031,
 'Dadra and Nagar Haveli and Daman and Diu': 0.14196188770942575}
```

In [48]:

```
max_survival_rate_state = 0
max_survival_rate = 0
for i in states:
    if survival_rate[i]>max_survival_rate:
        max_survival_rate = survival_rate[i]
        max_survival_rate_state = i

print(f'State with maximum Survival Rate (per month) = "{max_survival_rate_state}"\nSurvival Rate (per month) = {max_survival_rate}')
```

```
State with maximum Survival Rate (per month) = "Dadra and Nagar Haveli and Daman and Diu"
Survival Rate (per month) = 0.14196188770942575
```

In [49]:

```
state_dct_df['Dadra and Nagar Haveli and Daman and Diu']
```

Out[49]:

	State	Cured	Deaths	Confirmed	Month
0	Dadra and Nagar Haveli and Daman and Diu	2	0	30	6
1	Dadra and Nagar Haveli and Daman and Diu	0	0	0	6
2	Dadra and Nagar Haveli and Daman and Diu	0	0	5	6
3	Dadra and Nagar Haveli and Daman and Diu	0	0	1	6
4	Dadra and Nagar Haveli and Daman and Diu	3	0	0	6
...	...	...	...	...	...
176	Dadra and Nagar Haveli and Daman and Diu	2	0	0	12
177	Dadra and Nagar Haveli and Daman and Diu	0	0	1	12

	State	Cured	Deaths	Confirmed	Month
178	Dadra and Nagar Haveli and Daman and Diu	0	0	1	12
179	Dadra and Nagar Haveli and Daman and Diu	2	0	4	12
180	Dadra and Nagar Haveli and Daman and Diu	2	0	5	12

181 rows × 5 columns

Death Rate = Total Deaths/(Total Confirmed*Total number of months for which data is available)

Q3.Check for state in which death rate is more than 1%

In [50]:

```
death_rate ={}
for s in states:
    total_confirmed = 0
    total_deaths = 0
    for i in range(row_count_dct[s]):
       total_confirmed += state_dct_df[s].iloc[i,3]
       total_deaths += state_dct_df[s].iloc[i,2]
    death_rate[s] = (total_deaths/(total_confirmed*total_months_dct[s]))
```

In [51]:

```
death_rate
```

Out[51]:

```
{'Kerala': 0.0003195304453571916,
 'Telangana': 0.0005376715190310288,
 'Delhi': 0.0016350366430418414,
 'Rajasthan': 0.0008686592800123893,
 'Uttar Pradesh': 0.001427335252690474,
 'Haryana': 0.0010637305972539212,
 'Ladakh': 0.0013602408295239157,
 'Tamil Nadu': 0.0014911931058492309,
 'Karnataka': 0.001326953234951144,
 'Maharashtra': 0.002572219470389654,
 'Punjab': 0.003155131537967724,
 'Jammu and Kashmir': 0.001544222101404795,
 'Andhra Pradesh': 0.0008067925470791291,
 'Uttarakhand': 0.0016514827965277163,
 'Odisha': 0.0005541870008356295,
 'Puducherry': 0.001648307469647021,
 'West Bengal': 0.0017362375613933208,
 'Chhattisgarh': 0.0012114585961497642,
 'Chandigarh': 0.0016228959921048302,
 'Gujarat': 0.001855589115683114,
 'Himachal Pradesh': 0.0016298348520594793,
 'Madhya Pradesh': 0.001545314815326136,
 'Bihar': 0.0005432012100851573,
 'Manipur': 0.0011782088195181088,
 'Mizoram': 0.0001508674880563238 7,
 'Andaman and Nicobar Islands': 0.0012766848053578903,
 'Goa': 0.001432512516603658,
 'Assam': 0.0005176072114054256,
 'Jharkhand': 0.0009922159254673107,
 'Arunachal Pradesh': 0.0003717899319286434,
 'Tripura': 0.0012579889040656986,
 'Nagaland': 0.0006485272623437968,
 'Meghalaya': 0.00109230906974662,
 'Dadar Nagar Haveli': 0.0,
 'Sikkim': 0.002804410354745925,
 'Dadra and Nagar Haveli and Daman and Diu': 8.526239502067613e-05}
```

In [52]:

```
states_with_death_rate_more_than_one_percent = []
for i in states:
    if death_rate[i]>0.01:
        states_with_death_rate_more_than_one_percent.append(i)
```

In [53]:

```
print('States with death rate of more than 1%  (per month) :\n')
if(not bool(states_with_death_rate_more_than_one_percent)):     # checking if dictonary i
s empty
    print("No states have death rate of more than 1%")
else:
    for i in states_with_death_rate_more_than_one_percent:
        print(i)
```

```
States with death rate of more than 1%  (per month) :

No states have death rate of more than 1%
```

Q. Filter the month in which heighest people are get infected to Covid-19 virus?

In [54]:

```
print(f"Peak Month = {peak_month}\nPeak Cases = {peak_val}")
```

```
Peak Month = 9
Peak Cases = 2604518
```

Q. Obtain state in which survival rate is high.

In [55]:

```
print(f'State with maximum Survival Rate (per month) = "{max_survival_rate_state}"\nSurvi
val Rate (per month) = {max_survival_rate}')
```

```
State with maximum Survival Rate (per month) = "Dadra and Nagar Haveli and Daman and Diu"
Survival Rate (per month) = 0.14196188770942575
```

Q. Check for state in which death rate is more than 1%

In [56]:

```
if(not bool(states_with_death_rate_more_than_one_percent)):     # checking if dictonary i
s empty
    print("No state has death rate of more than 1% (per month).")
else:
    print('States with death rate of more than 1% (per month):\n')
    for i in states_with_death_rate_more_than_one_percent:
        print(i)
```

```
No state has death rate of more than 1% (per month).
```