In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
```

In [3]:
```
import plotly.express as px
```

In [4]:
```
data = pd.read_csv(r'C:\Users\Lenovo\Downloads\covid_19_india.csv')
data
```

Out[4]:

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational
0	1	2020-01-30	6:00 PM	Kerala	1	0
1	2	2020-01-31	6:00 PM	Kerala	1	0
2	3	2020-02-01	6:00 PM	Kerala	2	0
3	4	2020-02-02	6:00 PM	Kerala	3	0
4	5	2020-02-03	6:00 PM	Kerala	3	0
...	...	...	...	...	...	...
16845	16846	2021-07-07	8:00 AM	Telangana	-	-
16846	16847	2021-07-07	8:00 AM	Tripura	-	-
16847	16848	2021-07-07	8:00 AM	Uttarakhand	-	-
16848	16849	2021-07-07	8:00 AM	Uttar Pradesh	-	-
16849	16850	2021-07-07	8:00 AM	West Bengal	-	-

16850 rows × 9 columns

In [5]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16850 entries, 0 to 16849
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Sno                       16850 non-null  int64 
 1   Date                      16850 non-null  object
 2   Time                      16850 non-null  object
 3   State/UnionTerritory      16850 non-null  object
 4   ConfirmedIndianNational   16850 non-null  object
 5   ConfirmedForeignNational  16850 non-null  object
 6   Cured                     16850 non-null  int64 
 7   Deaths                    16850 non-null  int64 
 8   Confirmed                 16850 non-null  int64 
dtypes: int64(4), object(5)
memory usage: 1.2+ MB
```

In [6]:
```
1 data.isnull().sum()
```

Out[6]:
```
Sno                         0
Date                        0
Time                        0
State/UnionTerritory        0
ConfirmedIndianNational     0
ConfirmedForeignNational    0
Cured                       0
Deaths                      0
Confirmed                   0
dtype: int64
```

In [7]:
```
1 data['State/UnionTerritory'].unique()
```

Out[7]:
```
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****'],
      dtype=object)
```

In [ ]:
```
1
```

In [8]:
```
data['State/UnionTerritory']=data['State/UnionTerritory'].replace('Bihar****
data['State/UnionTerritory']=data['State/UnionTerritory'].replace('Daman & D
data['State/UnionTerritory']=data['State/UnionTerritory'].replace('Dadra and
data['State/UnionTerritory']=data['State/UnionTerritory'].replace('Telengana
```

In [9]:
```
data
```

Out[9]:

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational
0	1	2020-01-30	6:00 PM	Kerala	1	0
1	2	2020-01-31	6:00 PM	Kerala	1	0
2	3	2020-02-01	6:00 PM	Kerala	2	0
3	4	2020-02-02	6:00 PM	Kerala	3	0
4	5	2020-02-03	6:00 PM	Kerala	3	0
...	...	...	...	...	...	...
16845	16846	2021-07-07	8:00 AM	Telangana	-	-
16846	16847	2021-07-07	8:00 AM	Tripura	-	-
16847	16848	2021-07-07	8:00 AM	Uttarakhand	-	-
16848	16849	2021-07-07	8:00 AM	Uttar Pradesh	-	-
16849	16850	2021-07-07	8:00 AM	West Bengal	-	-

16850 rows × 9 columns

In [10]:
```
df = data.copy()
```

In [11]:
```
1 df
```

Out[11]:

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational
0	1	2020-01-30	6:00 PM	Kerala	1	0
1	2	2020-01-31	6:00 PM	Kerala	1	0
2	3	2020-02-01	6:00 PM	Kerala	2	0
3	4	2020-02-02	6:00 PM	Kerala	3	0
4	5	2020-02-03	6:00 PM	Kerala	3	0
...	...	...	...	...	...	...
16845	16846	2021-07-07	8:00 AM	Telangana	-	-
16846	16847	2021-07-07	8:00 AM	Tripura	-	-
16847	16848	2021-07-07	8:00 AM	Uttarakhand	-	-
16848	16849	2021-07-07	8:00 AM	Uttar Pradesh	-	-
16849	16850	2021-07-07	8:00 AM	West Bengal	-	-

16850 rows × 9 columns

In [12]:
```
df.drop(columns=['Sno','ConfirmedIndianNational','ConfirmedForeignNational'])
df
```

Out[12]:

	Date	Time	State/UnionTerritory	Cured	Deaths	Confirmed
0	2020-01-30	6:00 PM	Kerala	0	0	1
1	2020-01-31	6:00 PM	Kerala	0	0	1
2	2020-02-01	6:00 PM	Kerala	0	0	2
3	2020-02-02	6:00 PM	Kerala	0	0	3
4	2020-02-03	6:00 PM	Kerala	0	0	3
...	...	...	...	...	...	...
16845	2021-07-07	8:00 AM	Telangana	613124	3703	628282
16846	2021-07-07	8:00 AM	Tripura	63964	701	68612
16847	2021-07-07	8:00 AM	Uttarakhand	332006	7338	340882
16848	2021-07-07	8:00 AM	Uttar Pradesh	1682130	22656	1706818
16849	2021-07-07	8:00 AM	West Bengal	1472132	17834	1507241

16850 rows × 6 columns

In [13]:
```
df['Active_cases'] = df['Confirmed']-df['Deaths']-df['Cured']
df
```

Out[13]:

	Date	Time	State/UnionTerritory	Cured	Deaths	Confirmed	Active_cases
0	2020-01-30	6:00 PM	Kerala	0	0	1	1
1	2020-01-31	6:00 PM	Kerala	0	0	1	1
2	2020-02-01	6:00 PM	Kerala	0	0	2	2
3	2020-02-02	6:00 PM	Kerala	0	0	3	3
4	2020-02-03	6:00 PM	Kerala	0	0	3	3
...	...	...	...	...	...	...	...
16845	2021-07-07	8:00 AM	Telangana	613124	3703	628282	11455
16846	2021-07-07	8:00 AM	Tripura	63964	701	68612	3947
16847	2021-07-07	8:00 AM	Uttarakhand	332006	7338	340882	1538
16848	2021-07-07	8:00 AM	Uttar Pradesh	1682130	22656	1706818	2032
16849	2021-07-07	8:00 AM	West Bengal	1472132	17834	1507241	17275

16850 rows × 7 columns

In [14]:
```
df["Date"] = pd.to_datetime(df["Date"], format = "%Y-%m-%d")
df
```

Out[14]:

	Date	Time	State/UnionTerritory	Cured	Deaths	Confirmed	Active_cases
0	2020-01-30	6:00 PM	Kerala	0	0	1	1
1	2020-01-31	6:00 PM	Kerala	0	0	1	1
2	2020-02-01	6:00 PM	Kerala	0	0	2	2
3	2020-02-02	6:00 PM	Kerala	0	0	3	3
4	2020-02-03	6:00 PM	Kerala	0	0	3	3
...	...	...	...	...	...	...	...
16845	2021-07-07	8:00 AM	Telangana	613124	3703	628282	11455
16846	2021-07-07	8:00 AM	Tripura	63964	701	68612	3947
16847	2021-07-07	8:00 AM	Uttarakhand	332006	7338	340882	1538
16848	2021-07-07	8:00 AM	Uttar Pradesh	1682130	22656	1706818	2032
16849	2021-07-07	8:00 AM	West Bengal	1472132	17834	1507241	17275

16850 rows × 7 columns

In [15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16850 entries, 0 to 16849
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   Date                  16850 non-null  datetime64[ns]
 1   Time                  16850 non-null  object        
 2   State/UnionTerritory  16850 non-null  object        
 3   Cured                 16850 non-null  int64         
 4   Deaths                16850 non-null  int64         
 5   Confirmed             16850 non-null  int64         
 6   Active_cases          16850 non-null  int64         
dtypes: datetime64[ns](1), int64(4), object(2)
memory usage: 921.6+ KB
```

In [16]:
```
df2 = df.groupby('State/UnionTerritory').sum()
```

In [17]:
```
1 df2
```

Out[17]:

State/UnionTerritory	Cured	Deaths	Confirmed	Active_cases
Andaman and Nicobar Islands	1589935	22624	1675248	62689
Andhra Pradesh	303427899	2475816	324146783	18243068
Arunachal Pradesh	5150519	19303	5598324	428502
Assam	74011348	459575	80418492	5947569
Bihar	101533848	775163	108312449	6003438
Cases being reassigned to states	0	0	345565	345565
Chandigarh	7980284	119356	8691806	592166
Chhattisgarh	117163544	1591126	128751782	9997112
Dadra and Nagar Haveli and Daman and Diu	1491338	882	1587570	95350
Delhi	224062704	4066907	236972842	8843231
Goa	20224042	338359	22280065	1717664
Gujarat	103995131	1866811	114557615	8695673
Haryana	100010131	1166573	107408371	6231667
Himachal Pradesh	20682770	371931	23052151	1997450
Jammu and Kashmir	42295048	686680	46899925	3918197
Jharkhand	46083978	569298	49971564	3318288
Karnataka	345648926	4819018	387597335	37129391
Kerala	311127643	1327754	344319045	31863648
Ladakh	3059045	38578	3344131	246508
Lakshadweep	471712	2178	561459	87569
Madhya Pradesh	100169697	1427780	108712983	7115506
Maharashtra	813788907	19314532	908892470	75789031
Manipur	8420223	122089	9440912	898600
Meghalaya	4606548	66293	5221064	548223
Mizoram	1534630	5073	1822190	282487
Nagaland	3628619	39420	4089547	421508
Odisha	117984789	600149	126408397	7823459
Puducherry	14376916	249683	15858688	1232089
Punjab	71108712	2216735	78999515	5674068
Rajasthan	117312772	1159823	128998101	10525506
Sikkim	1983899	41530	2315519	290090
Tamil Nadu	317067499	4731627	342829697	21030571
Telangana	100211245	617882	108152726	7323599

State/UnionTerritory	Cured	Deaths	Confirmed	Active_cases
Tripura	10479169	124444	11397656	794043
Unassigned	0	0	161	161
Uttar Pradesh	232529439	3347656	252843682	16966587
Uttarakhand	36684388	728512	41179396	3766496
West Bengal	195296839	3214840	209822848	11311169

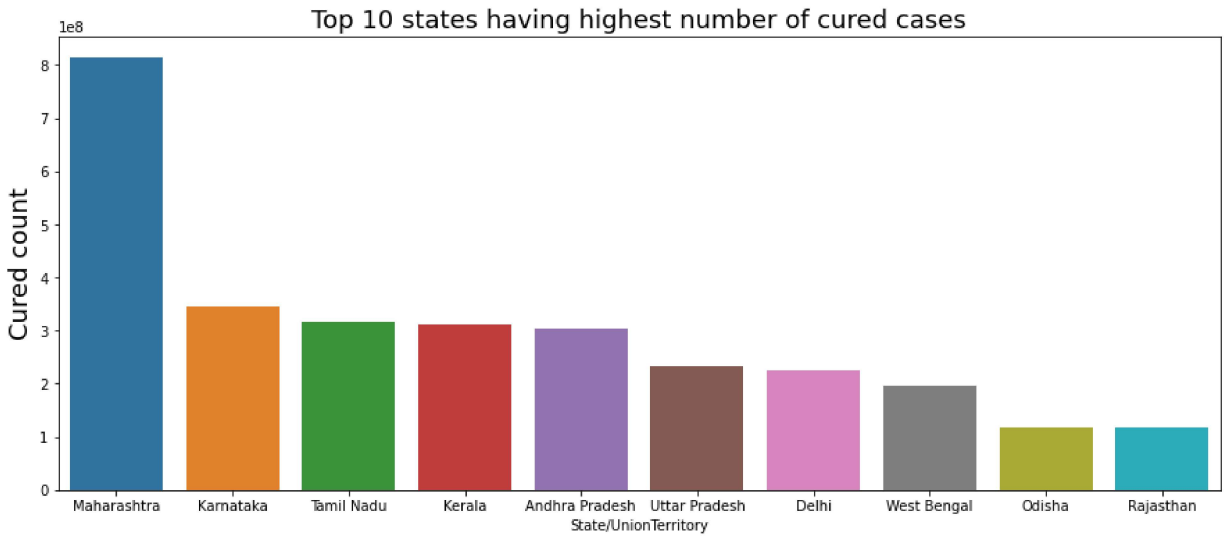
In [18]:
```
cured_count= df2.groupby(['State/UnionTerritory'])['Cured'].max()
max_cured_count = cured_count.sort_values(ascending=False)[0:10].reset_index
max_cured_count
```

Out[18]:

	State/UnionTerritory	Cured
0	Maharashtra	813788907
1	Karnataka	345648926
2	Tamil Nadu	317067499
3	Kerala	311127643
4	Andhra Pradesh	303427899
5	Uttar Pradesh	232529439
6	Delhi	224062704
7	West Bengal	195296839
8	Odisha	117984789
9	Rajasthan	117312772

In [19]:
```
fig = sns.barplot(x=max_cured_count['State/UnionTerritory'],y=max_cured_coun
plt.ylabel('Cured count',size=18)
plt.title('Top 10 states having highest number of cured cases',size=18 )
fig.figure.set_figheight(6)
fig.figure.set_figwidth(15)

plt.show()
```



Top 10 states having highest number of cases

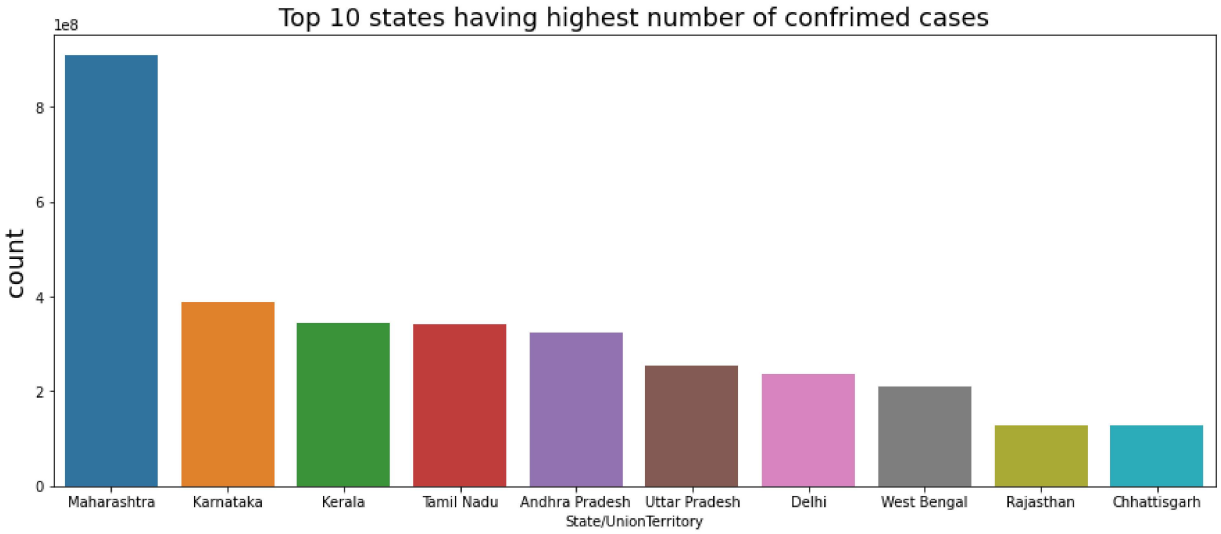
In [20]:
```
confirmed_count= df2.groupby(['State/UnionTerritory'])['Confirmed'].max()
max_confirmed_count = confirmed_count.sort_values(ascending=False)[0:10].res
max_confirmed_count
```

Out[20]:

	State/UnionTerritory	Confirmed
0	Maharashtra	908892470
1	Karnataka	387597335
2	Kerala	344319045
3	Tamil Nadu	342829697
4	Andhra Pradesh	324146783
5	Uttar Pradesh	252843682
6	Delhi	236972842
7	West Bengal	209822848
8	Rajasthan	128998101
9	Chhattisgarh	128751782

In [21]:
```
fig = sns.barplot(x=max_confirmed_count['State/UnionTerritory'],y=max_confir
plt.ylabel('count',size=18)
plt.title('Top 10 states having highest number of confrimed cases',size=18 )
fig.figure.set_figheight(6)
fig.figure.set_figwidth(15)

plt.show()
```



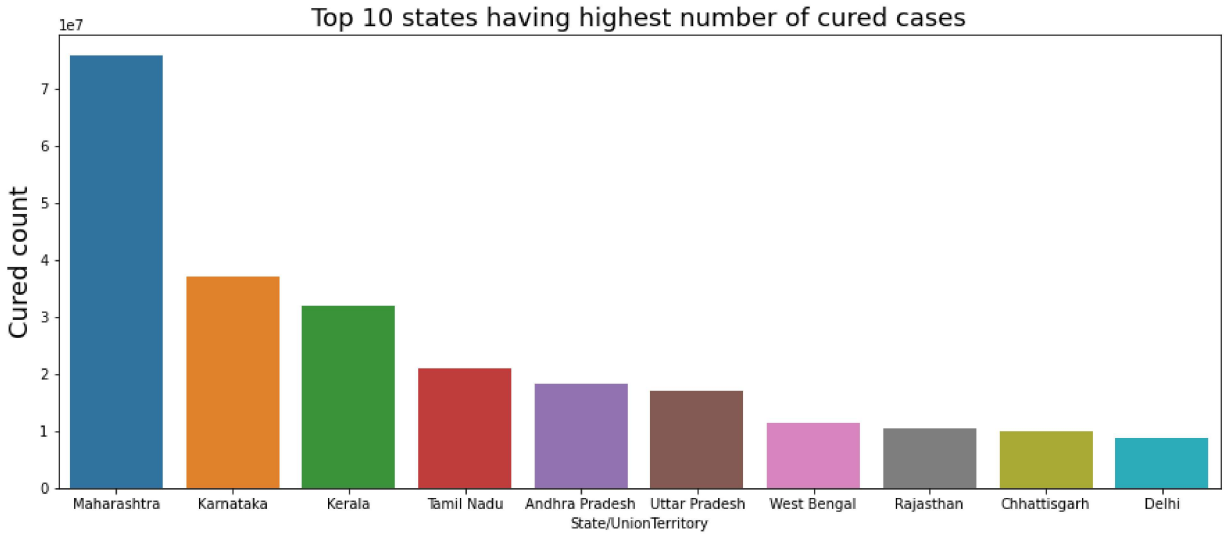
In [22]:
```
active_count= df2.groupby(['State/UnionTerritory'])['Active_cases'].max()
max_active_count = active_count.sort_values(ascending=False)[0:10].reset_ind
max_active_count
```

Out[22]:

	State/UnionTerritory	Active_cases
0	Maharashtra	75789031
1	Karnataka	37129391
2	Kerala	31863648
3	Tamil Nadu	21030571
4	Andhra Pradesh	18243068
5	Uttar Pradesh	16966587
6	West Bengal	11311169
7	Rajasthan	10525506
8	Chhattisgarh	9997112
9	Delhi	8843231

In [23]:
```
fig = sns.barplot(x=max_active_count['State/UnionTerritory'],y=max_active_co
plt.ylabel('Cured count',size=18)
plt.title('Top 10 states having highest number of cured cases',size=18 )
fig.figure.set_figheight(6)
fig.figure.set_figwidth(15)

plt.show()
```



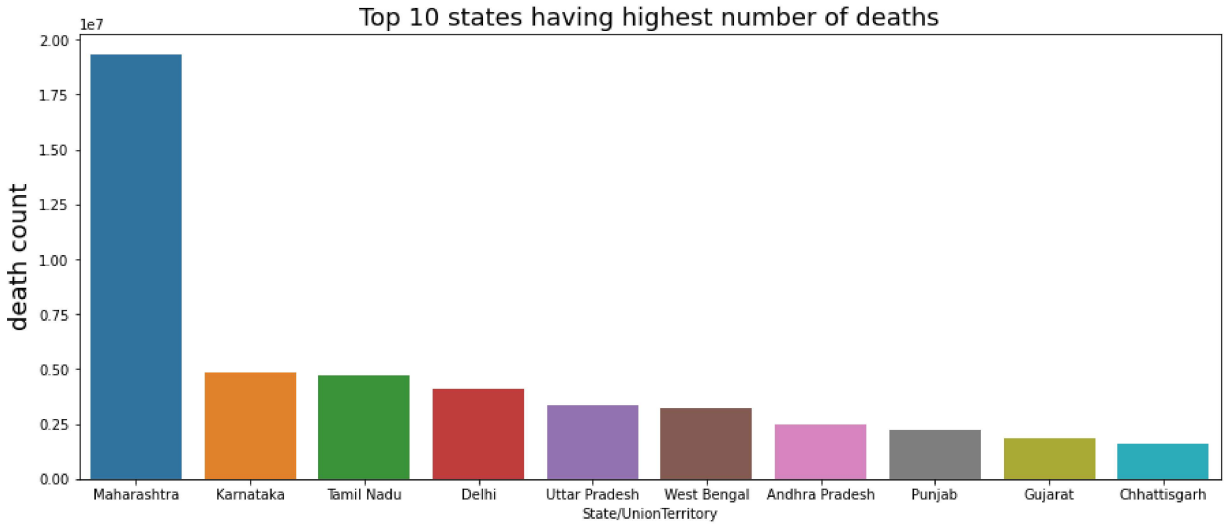
In [24]:
```
Death_count= df2.groupby(['State/UnionTerritory'])['Deaths'].max()
max_death_count = Death_count.sort_values(ascending=False)[0:10].reset_index
max_death_count
```

Out[24]:

	State/UnionTerritory	Deaths
0	Maharashtra	19314532
1	Karnataka	4819018
2	Tamil Nadu	4731627
3	Delhi	4066907
4	Uttar Pradesh	3347656
5	West Bengal	3214840
6	Andhra Pradesh	2475816
7	Punjab	2216735
8	Gujarat	1866811
9	Chhattisgarh	1591126

In [25]:
```
fig = sns.barplot(x=max_death_count['State/UnionTerritory'],y=max_death_coun
plt.ylabel('death count',size=18)
plt.title('Top 10 states having highest number of deaths',size=18 )
fig.figure.set_figheight(6)
fig.figure.set_figwidth(15)

plt.show()
```



In [26]:
```
max_cases = df.groupby(['State/UnionTerritory'])['Confirmed'].max().sort_val
max_cases
```

Out[26]:

	State/UnionTerritory	Confirmed
0	Maharashtra	6113335
1	Kerala	2996094
2	Karnataka	2859595
3	Tamil Nadu	2503481
4	Andhra Pradesh	1908065
5	Uttar Pradesh	1706818
6	West Bengal	1507241
7	Delhi	1434687
8	Chhattisgarh	996359
9	Rajasthan	952836
10	Odisha	927186
11	Gujarat	823964
12	Madhya Pradesh	790042
13	Haryana	769030
14	Bihar	722746
15	Telangana	628282
16	Punjab	596736
17	Assam	522267
18	Jharkhand	346038
19	Uttarakhand	340882
20	Jammu and Kashmir	317481
21	Himachal Pradesh	202945
22	Goa	167823
23	Puducherry	118227
24	Manipur	73581
25	Tripura	68612
26	Chandigarh	61752
27	Meghalaya	52358
28	Arunachal Pradesh	37879
29	Nagaland	25619
30	Mizoram	22155
31	Sikkim	21403
32	Ladakh	20137

	State/UnionTerritory	Confirmed
33	Dadra and Nagar Haveli and Daman and Diu	10575
34	Lakshadweep	9947
35	Cases being reassigned to states	9265
36	Andaman and Nicobar Islands	7487
37	Unassigned	77

MAXIMUM NUMBER OF CASES RECORED IN ONE DAY PER STATE

In [27]:
```
top_10max = max_cases.sort_values(by='Confirmed',ascending=False)[0:10]
```

In [28]:
```
top_10max
```

Out[28]:

	State/UnionTerritory	Confirmed
0	Maharashtra	6113335
1	Kerala	2996094
2	Karnataka	2859595
3	Tamil Nadu	2503481
4	Andhra Pradesh	1908065
5	Uttar Pradesh	1706818
6	West Bengal	1507241
7	Delhi	1434687
8	Chhattisgarh	996359
9	Rajasthan	952836

In [29]:
```
death_rate = (df2['Deaths']/df2['Confirmed']*100)
death_rate = pd.DataFrame(data=death_rate,columns=['Death_rate'])
death_rate
```

Out[29]:

State/UnionTerritory	Death_rate
Andaman and Nicobar Islands	1.350487
Andhra Pradesh	0.763795
Arunachal Pradesh	0.344800
Assam	0.571479
Bihar	0.715673
Cases being reassigned to states	0.000000
Chandigarh	1.373201
Chhattisgarh	1.235809
Dadra and Nagar Haveli and Daman and Diu	0.055557
Delhi	1.716191
Goa	1.518663
Gujarat	1.629583
Haryana	1.086110
Himachal Pradesh	1.613433
Jammu and Kashmir	1.464139
Jharkhand	1.139244
Karnataka	1.243305
Kerala	0.385617
Ladakh	1.153603
Lakshadweep	0.387918
Madhya Pradesh	1.313348
Maharashtra	2.125062
Manipur	1.293191
Meghalaya	1.269722
Mizoram	0.278401
Nagaland	0.963921
Odisha	0.474770
Puducherry	1.574424
Punjab	2.806011
Rajasthan	0.899101
Sikkim	1.793550

State/UnionTerritory	Death_rate
Tamil Nadu	1.380168
Telangana	0.571305
Tripura	1.091839
Unassigned	0.000000
Uttar Pradesh	1.324002
Uttarakhand	1.769118
West Bengal	1.532169

In [30]:
```
top_death_rate = death_rate.sort_values(by='Death_rate',ascending=False)[0:1
top_death_rate
```

Out[30]:

State/UnionTerritory	Death_rate
Punjab	2.806011
Maharashtra	2.125062
Sikkim	1.793550
Uttarakhand	1.769118
Delhi	1.716191
Gujarat	1.629583
Himachal Pradesh	1.613433
Puducherry	1.574424
West Bengal	1.532169
Goa	1.518663

In [31]:
```
cured_rate =df2['Cured']/df2['Confirmed']*100
cured_rate
```

Out[31]:
```
State/UnionTerritory
Andaman and Nicobar Islands                 94.907441
Andhra Pradesh                              93.608178
Arunachal Pradesh                           92.001088
Assam                                       92.032748
Bihar                                       93.741623
Cases being reassigned to states             0.000000
Chandigarh                                  91.813876
Chhattisgarh                                90.999551
Dadra and Nagar Haveli and Daman and Diu    93.938409
Delhi                                       94.552060
Goa                                         90.771917
Gujarat                                     90.779763
Haryana                                     93.112045
Himachal Pradesh                            89.721649
Jammu and Kashmir                           90.181483
Jharkhand                                   92.220404
Karnataka                                   89.177323
Kerala                                      90.360277
Ladakh                                      91.475035
Lakshadweep                                 84.015396
Madhya Pradesh                              92.141430
Maharashtra                                 89.536324
Manipur                                     89.188661
Meghalaya                                   88.230062
Mizoram                                     84.218989
Nagaland                                    88.729118
Odisha                                      93.336196
Puducherry                                  90.656402
Punjab                                      90.011580
Rajasthan                                   90.941472
Sikkim                                      85.678373
Tamil Nadu                                  92.485424
Telangana                                   92.657161
Tripura                                     91.941440
Unassigned                                   0.000000
Uttar Pradesh                               91.965691
Uttarakhand                                 89.084328
West Bengal                                 93.077013
dtype: float64
```

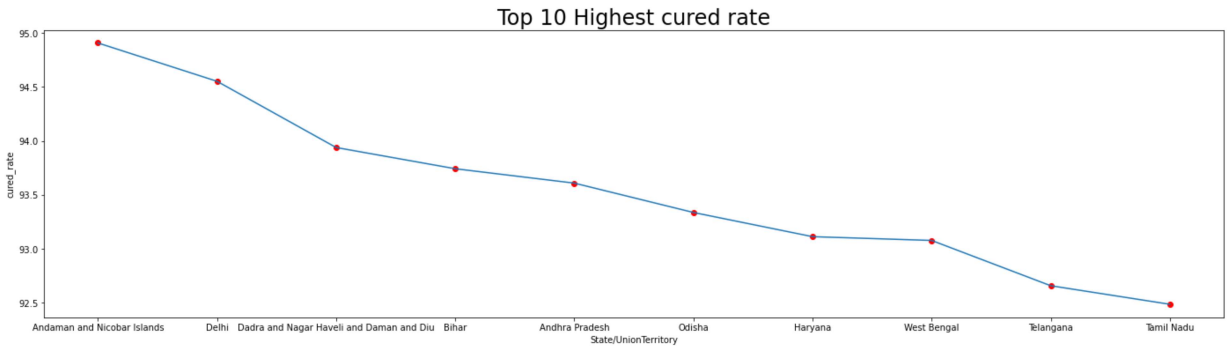
In [32]:
```
top_cured_rate = cured_rate.sort_values(ascending=False)[0:10]
top_cured_rate= pd.DataFrame(data=top_cured_rate,columns=['cured_rate'])
top_cured_rate
```

Out[32]:

State/UnionTerritory	cured_rate
Andaman and Nicobar Islands	94.907441
Delhi	94.552060
Dadra and Nagar Haveli and Daman and Diu	93.938409
Bihar	93.741623
Andhra Pradesh	93.608178
Odisha	93.336196
Haryana	93.112045
West Bengal	93.077013
Telangana	92.657161
Tamil Nadu	92.485424

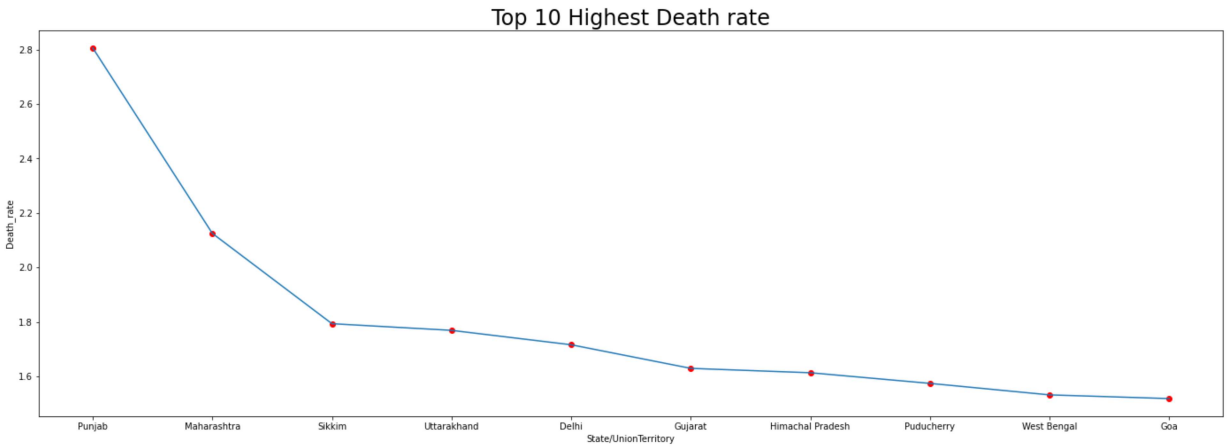
In [33]:
```
fig=sns.lineplot(x= top_cured_rate.index, y=top_cured_rate['cured_rate'])
plt.scatter(x=top_cured_rate.index, y=top_cured_rate.values , c='red')
fig.figure.set_figwidth(24)
fig.figure.set_figheight(6)
fig.set_title('Top 10 Highest cured rate',size=24)
```

Out[33]: Text(0.5, 1.0, 'Top 10 Highest cured rate')



In [34]:
```
fig=sns.lineplot(x= top_death_rate.index, y=top_death_rate['Death_rate'])
plt.scatter(x=top_death_rate.index, y=top_death_rate.values , c='red')
fig.figure.set_figwidth(24)
fig.figure.set_figheight(8)
fig.set_title('Top 10 Highest Death rate',size=24)
```

Out[34]: Text(0.5, 1.0, 'Top 10 Highest Death rate')



Among all the states PANJAB have the HEIGHTEST DEATH RATE

In [35]:
```
Ind_year_2020 = df[df['Date'].dt.year==2020]
Ind_year_2020
```

Out[35]:

	Date	Time	State/UnionTerritory	Cured	Deaths	Confirmed	Active_cases
0	2020-01-30	6:00 PM	Kerala	0	0	1	1
1	2020-01-31	6:00 PM	Kerala	0	0	1	1
2	2020-02-01	6:00 PM	Kerala	0	0	2	2
3	2020-02-02	6:00 PM	Kerala	0	0	3	3
4	2020-02-03	6:00 PM	Kerala	0	0	3	3
...	...	...	...	...	...	...	...
10077	2020-12-31	8:00 AM	Telangana	278839	1541	286354	5974
10078	2020-12-31	8:00 AM	Tripura	32751	385	33264	128
10079	2020-12-31	8:00 AM	Uttarakhand	84149	1504	90616	4963
10080	2020-12-31	8:00 AM	Uttar Pradesh	562459	8352	584966	14155
10081	2020-12-31	8:00 AM	West Bengal	528829	9683	550893	12381

10082 rows × 7 columns

In [36]:
```
total_2020=(Ind_year_2020['Cured'].sum(),Ind_year_2020['Deaths'].sum(),
            Ind_year_2020['Confirmed'].sum(),Ind_year_2020['Active_cases'].s
```

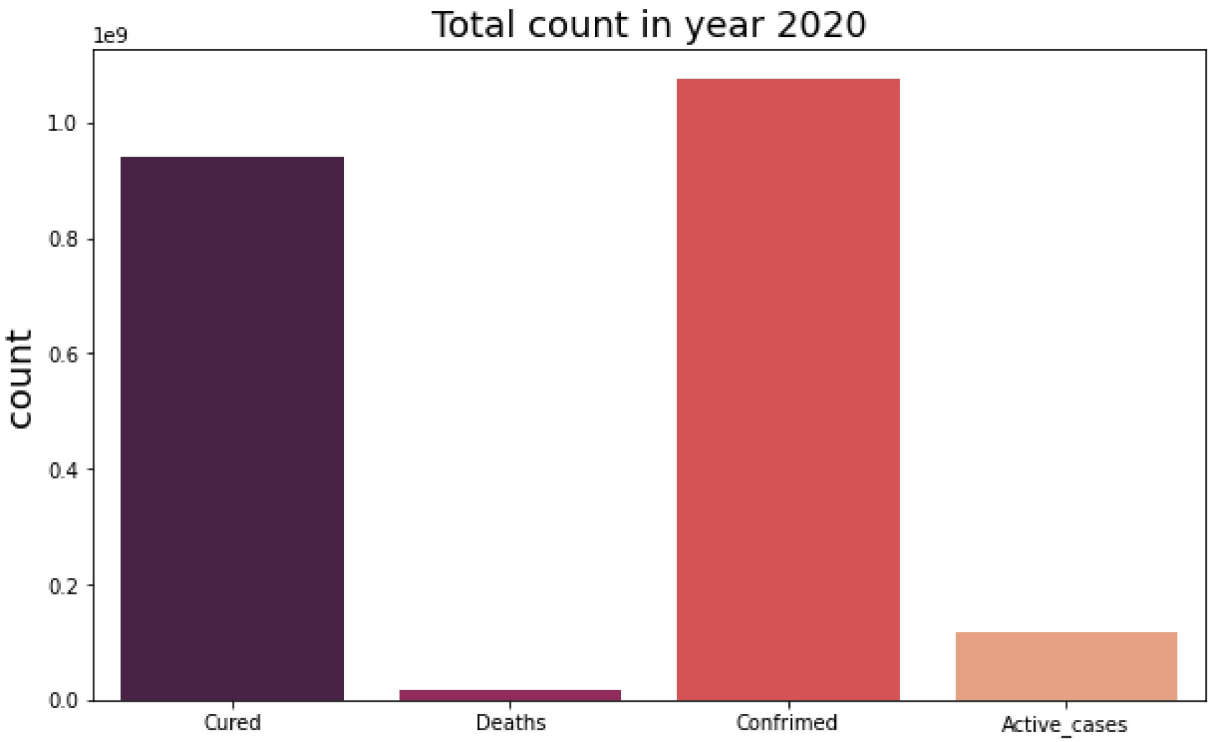
In [37]:
```
total_2020 = pd.DataFrame(data=total_2020, index=['Cured','Deaths','Confrime
total_2020
```

Out[37]:

	Total
Cured	941314195
Deaths	17022508
Confrimed	1074022781
Active_cases	115686078

In [38]:
```
fig = sns.barplot(x=total_2020.index,y=total_2020['Total'],palette='rocket')
plt.ylabel('count',size=18)
plt.title('Total count in year 2020',size=18 )
fig.figure.set_figheight(6)
fig.figure.set_figwidth(10)

plt.show()
```



In [39]:
```
Ind_year_2021 = df[df['Date'].dt.year==2021]
Ind_year_2021
```

Out[39]:

	Date	Time	State/UnionTerritory	Cured	Deaths	Confirmed	Active_cases
10082	2021-01-01	8:00 AM	Andhra Pradesh	871916	7108	882286	3262
10083	2021-01-01	8:00 AM	Andaman and Nicobar Islands	4826	62	4945	57
10084	2021-01-01	8:00 AM	Arunachal Pradesh	16564	56	16719	99
10085	2021-01-01	8:00 AM	Assam	211910	1045	216211	3256
10086	2021-01-01	8:00 AM	Bihar	245476	1397	251743	4870
...	...	...	...	...	...	...	...
16845	2021-07-07	8:00 AM	Telangana	613124	3703	628282	11455
16846	2021-07-07	8:00 AM	Tripura	63964	701	68612	3947
16847	2021-07-07	8:00 AM	Uttarakhand	332006	7338	340882	1538
16848	2021-07-07	8:00 AM	Uttar Pradesh	1682130	22656	1706818	2032
16849	2021-07-07	8:00 AM	West Bengal	1472132	17834	1507241	17275

6768 rows × 7 columns

In [40]:
```
total_2021=(Ind_year_2021['Cured'].sum(),Ind_year_2021['Deaths'].sum(),
            Ind_year_2021['Confirmed'].sum(),Ind_year_2021['Active_cases'].s
```

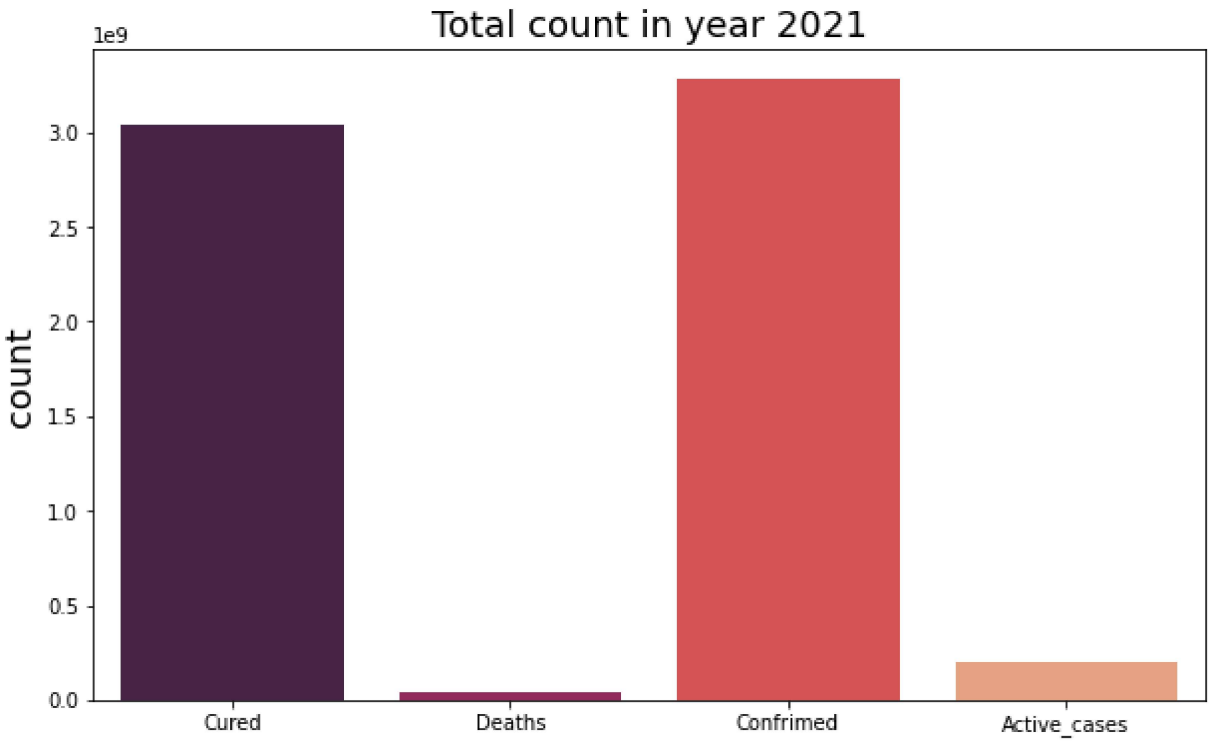
In [41]:
```
total_2021 = pd.DataFrame(data=total_2021, index=['Cured','Deaths','Confrime
total_2021
```

Out[41]:

	Total
Cured	3035879941
Deaths	41703492
Confrimed	3279455293
Active_cases	201871860

In [42]:
```
fig = sns.barplot(x=total_2021.index,y=total_2021['Total'],palette='rocket')
plt.ylabel('count',size=18)
plt.title('Total count in year 2021',size=18)
fig.figure.set_figheight(6)
fig.figure.set_figwidth(10)

plt.show()
```



In [43]:
```
cured_rate_2021 = Ind_year_2021['Cured'].sum()/Ind_year_2021['Confirmed'].su
cured_rate_2020 = Ind_year_2020['Cured'].sum()/Ind_year_2020['Confirmed'].su
print(f'Cured Rate of India in year 2020 is {cured_rate_2020} and in year 20
```

```
Cured Rate of India in year 2020 is 87.64378294877156 and in year 2021 is 92.57
268874742974.
```

Cured rate was high in year 2021

In [44]:
```
Death_rate_2021 = Ind_year_2021['Deaths'].sum()/Ind_year_2021['Confirmed'].s
Death_rate_2020 = Ind_year_2020['Deaths'].sum()/Ind_year_2020['Confirmed'].s
print(f'Cured Rate of India in year 2020 is {Death_rate_2020} and in year 20
```

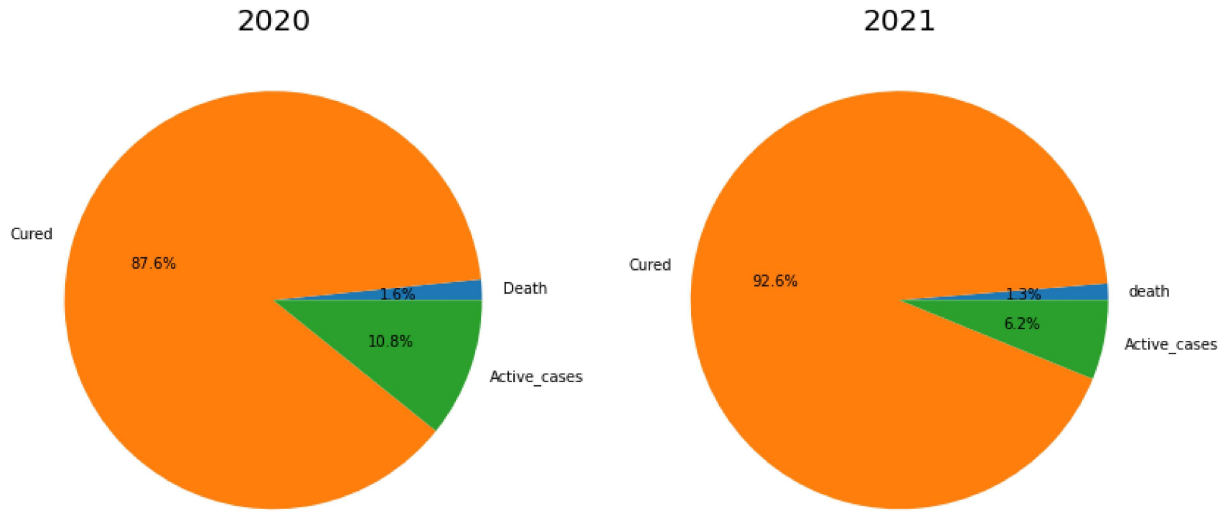
```
Cured Rate of India in year 2020 is 1.5849298824137326 and in year 2021 is 1.27
1659110249685.
```

Death rate was high in year 2020

In [ ]:
```
1
```

In [45]:
```
1  fig, (ax1,ax2) = plt.subplots(1,2,figsize=(14,14))
2  val = [Ind_year_2020['Deaths'].sum(),Ind_year_2020['Cured'].sum(),Ind_year_2
3  label = ['Death','Cured','Active_cases']
4  ax1.pie(val,labels=label,autopct='%1.1f%%')
5  ax1.set_title('2020',size=20)
6
7  val = [Ind_year_2021['Deaths'].sum(),Ind_year_2021['Cured'].sum(),Ind_year_2
8  label = ['death','Cured','Active_cases']
9  ax2.pie(val,labels=label,autopct='%1.1f%%')
10 ax2.set_title('2021',size=20)
11
```

Out[45]: Text(0.5, 1.0, '2021')



In year 2020 87.6% patient got cured and around 1.6% patient were dead ,whereas in year 2021 92.6% patient got cured and 1.3% patient were dead.

In [46]:
```
first_wave= Ind_year_2020[(Ind_year_2020['Date'].dt.month<=12)]
first_wave
```

Out[46]:

	Date	Time	State/UnionTerritory	Cured	Deaths	Confirmed	Active_cases
0	2020-01-30	6:00 PM	Kerala	0	0	1	1
1	2020-01-31	6:00 PM	Kerala	0	0	1	1
2	2020-02-01	6:00 PM	Kerala	0	0	2	2
3	2020-02-02	6:00 PM	Kerala	0	0	3	3
4	2020-02-03	6:00 PM	Kerala	0	0	3	3
...	...	...	...	...	...	...	...
10077	2020-12-31	8:00 AM	Telangana	278839	1541	286354	5974
10078	2020-12-31	8:00 AM	Tripura	32751	385	33264	128
10079	2020-12-31	8:00 AM	Uttarakhand	84149	1504	90616	4963
10080	2020-12-31	8:00 AM	Uttar Pradesh	562459	8352	584966	14155
10081	2020-12-31	8:00 AM	West Bengal	528829	9683	550893	12381

10082 rows × 7 columns

In [47]:
```
second_wave = Ind_year_2021[(Ind_year_2021['Date'].dt.month<=12)]
second_wave
```

Out[47]:

	Date	Time	State/UnionTerritory	Cured	Deaths	Confirmed	Active_cases
10082	2021-01-01	8:00 AM	Andhra Pradesh	871916	7108	882286	3262
10083	2021-01-01	8:00 AM	Andaman and Nicobar Islands	4826	62	4945	57
10084	2021-01-01	8:00 AM	Arunachal Pradesh	16564	56	16719	99
10085	2021-01-01	8:00 AM	Assam	211910	1045	216211	3256
10086	2021-01-01	8:00 AM	Bihar	245476	1397	251743	4870
...	...	...	...	...	...	...	...
16845	2021-07-07	8:00 AM	Telangana	613124	3703	628282	11455
16846	2021-07-07	8:00 AM	Tripura	63964	701	68612	3947
16847	2021-07-07	8:00 AM	Uttarakhand	332006	7338	340882	1538
16848	2021-07-07	8:00 AM	Uttar Pradesh	1682130	22656	1706818	2032
16849	2021-07-07	8:00 AM	West Bengal	1472132	17834	1507241	17275

6768 rows × 7 columns

In [48]:
```
cured_rate_1st = first_wave['Cured'].sum()/first_wave['Confirmed'].sum()*100
cured_rate_2nd = second_wave['Cured'].sum()/second_wave['Confirmed'].sum()*1
print(f'Cured Rate of India in 1st wave is {cured_rate_1st} and in 2nd wave

```

```
Cured Rate of India in 1st wave is 87.64378294877156 and in 2nd wave is 92.5726
8874742974.
```

In [49]:
```
death_rate_1st = first_wave['Deaths'].sum()/first_wave['Confirmed'].sum()*10
death_rate_2nd = second_wave['Deaths'].sum()/second_wave['Confirmed'].sum()*
print(f'Death Rate of India in 1st wave is {death_rate_1st} and in 2nd wave

```

```
Death Rate of India in 1st wave is 1.5849298824137326 and in 2nd wave is 1.2716
59110249685.
```

In [70]:
```
monthly = first_wave.groupby('Date')[['Cured','Active_cases','Deaths','Confi
monthly
```

Out[70]:

Date	Cured	Active_cases	Deaths	Confirmed
2020-03-09	0	2	0	2
2020-03-10	0	5	0	5
2020-03-11	0	2	0	2
2020-03-12	0	11	0	11
2020-03-13	0	14	0	14
...	...	...	...	...
2020-12-27	1807824	59223	49189	1916236
2020-12-28	1809948	60347	49255	1919550
2020-12-29	1814449	58294	49305	1922048
2020-12-30	1820021	55672	49373	1925066
2020-12-31	1824934	54206	49463	1928603

298 rows × 4 columns

In [69]:
```
monthly = second_wave.groupby('Date')[['Cured','Active_cases','Deaths','Conf
monthly
```

Out[69]:

Date	Cured	Active_cases	Deaths	Confirmed
2021-04-01	2400727	357604	54649	2812980
2021-04-02	2433368	367897	54898	2856163
2021-04-03	2457494	391203	55379	2904076
2021-04-04	2495315	402552	55656	2953523
2021-04-05	2522823	431896	55878	3010597
...	...	...	...	...
2021-05-27	5241833	317733	91341	5650907
2021-05-28	5276203	303752	92225	5672180
2021-05-29	5307874	291848	93198	5692920
2021-05-30	5339838	279347	94030	5713215
2021-05-31	5362370	274601	94844	5731815

61 rows × 4 columns

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [50]:
```
Maha_data = df[df['State/UnionTerritory']=='Maharashtra'].reset_index()
Maha_data.drop(columns=['State/UnionTerritory','Time','index'],inplace=True)
Maha_data
```

Out[50]:

	Date	Cured	Deaths	Confirmed	Active_cases
0	2020-03-09	0	0	2	2
1	2020-03-10	0	0	5	5
2	2020-03-11	0	0	2	2
3	2020-03-12	0	0	11	11
4	2020-03-13	0	0	14	14
...	...	...	...	...	...
481	2021-07-03	5836920	122353	6079352	120079
482	2021-07-04	5845315	122724	6088841	120802
483	2021-07-05	5848693	123030	6098177	126454
484	2021-07-06	5861720	123136	6104917	120061
485	2021-07-07	5872268	123531	6113335	117536

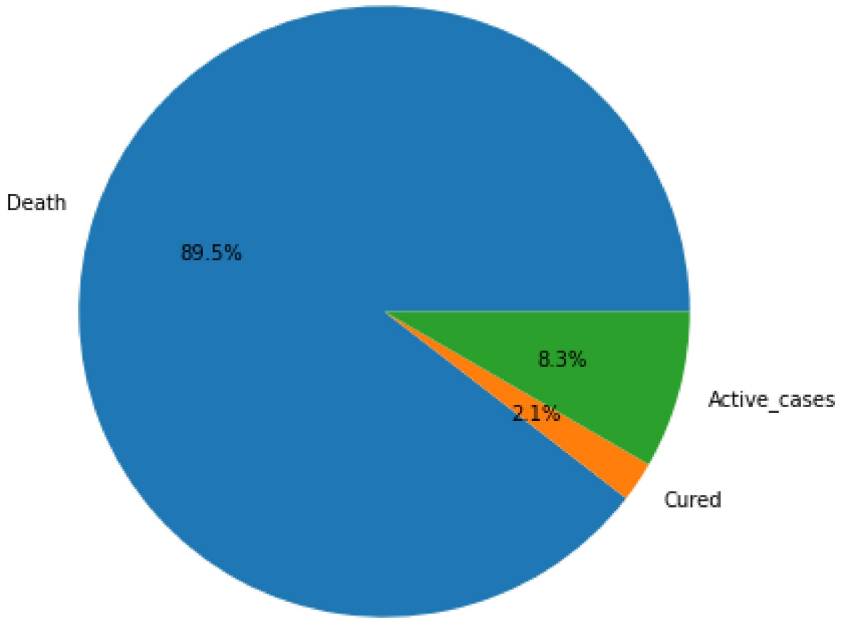
486 rows × 5 columns

In [51]:
```
label =['cured','deaths','active']
```

In [52]:

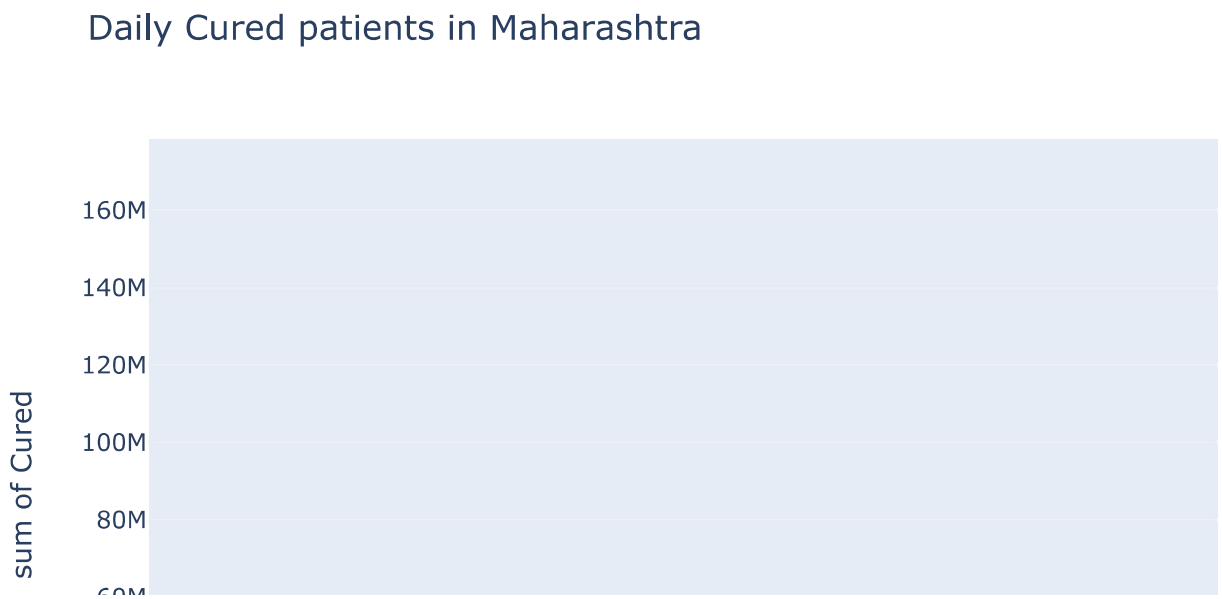
```
fig= plt.figure()
fig.set_figwidth(10)
fig.set_figheight(7)

val = [Maha_data['Cured'].sum(),Maha_data['Deaths'].sum(),Maha_data['Active_
label = ['Death','Cured','Active_cases']
plt.pie(val,labels=label,autopct='%1.1f%%')
plt.show()
```

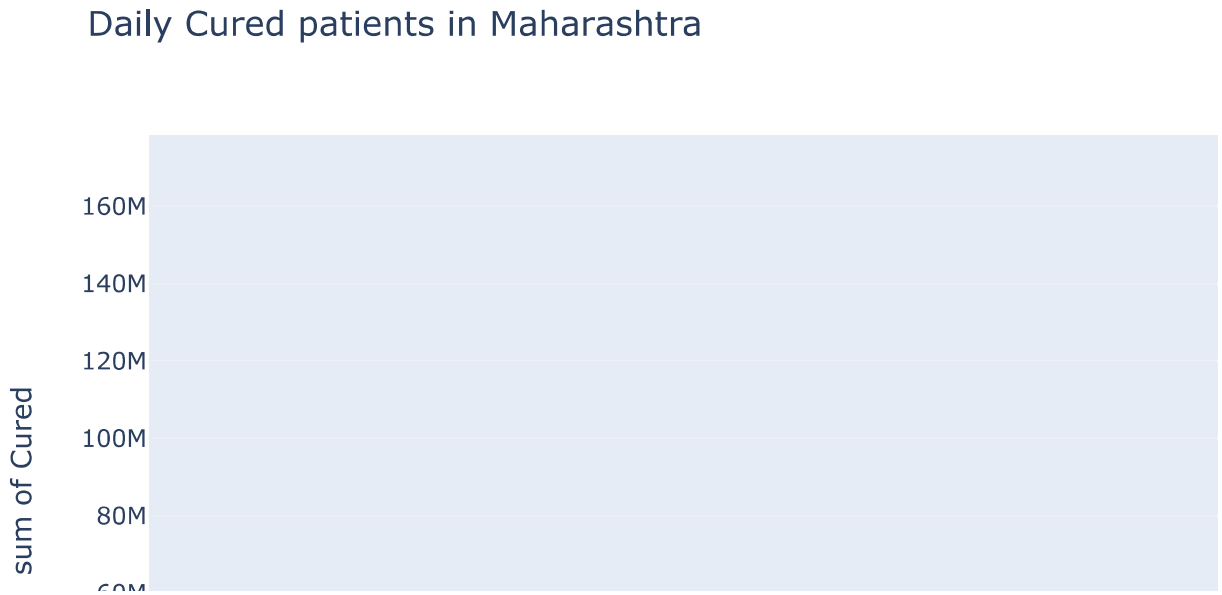


In [53]:
```
fig = px.histogram(Maha_data,x="Date", y="Confirmed",title="Daily Confirmed
fig.update_traces(marker_line_width=1,marker_line_color='black')
fig.show()
```

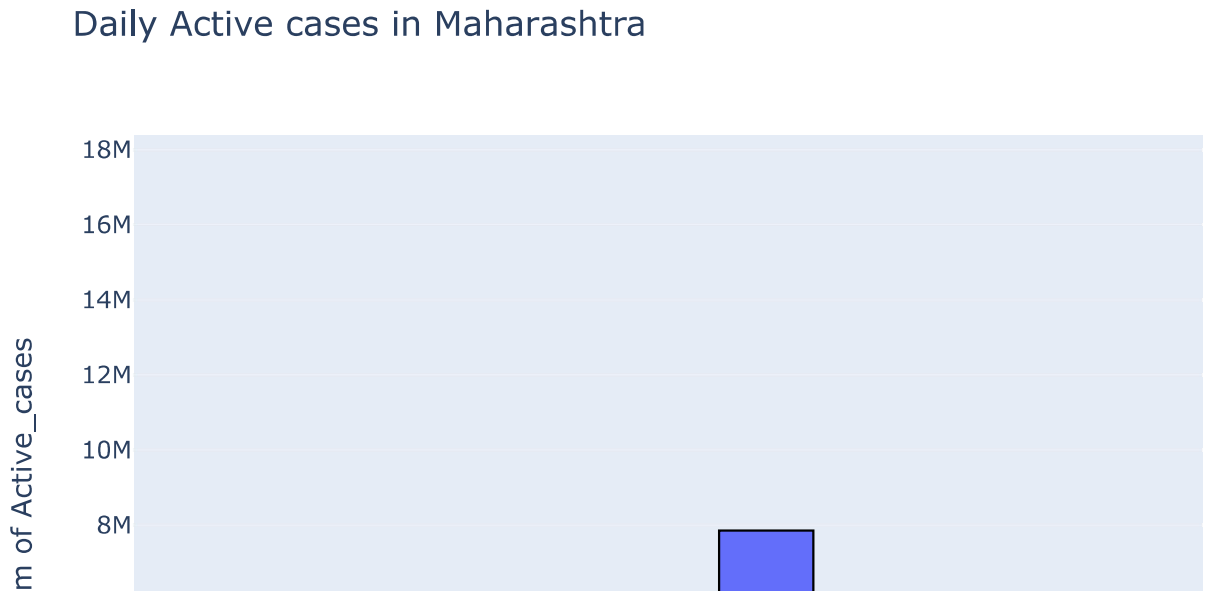
In [54]:
```
fig = px.histogram(Maha_data,x="Date", y="Cured",title="Daily Cured patients
fig.update_traces(marker_line_width=1,marker_line_color='black')
fig.show()
```



In [55]:
```
line = px.histogram(Maha_data,x="Date", y="Deaths",title="Daily no of Deaths
fig.update_traces(marker_line_width=1,marker_line_color='black')
fig.show()
```



In [56]:
```
fig = px.histogram(Maha_data,x="Date", y="Active_cases",title="Daily Active
fig.update_traces(marker_line_width=1,marker_line_color='black')
fig.show()
```



In [57]:
```
year_2020 = Maha_data[Maha_data['Date'].dt.year==2020]
year_2020
```

Out[57]:

	Date	Cured	Deaths	Confirmed	Active_cases
0	2020-03-09	0	0	2	2
1	2020-03-10	0	0	5	5
2	2020-03-11	0	0	2	2
3	2020-03-12	0	0	11	11
4	2020-03-13	0	0	14	14
...	...	...	...	...	...
293	2020-12-27	1807824	49189	1916236	59223
294	2020-12-28	1809948	49255	1919550	60347
295	2020-12-29	1814449	49305	1922048	58294
296	2020-12-30	1820021	49373	1925066	55672
297	2020-12-31	1824934	49463	1928603	54206

298 rows × 5 columns

In [58]:
```
year_2021 = Maha_data[Maha_data['Date'].dt.year==2021]
year_2021
```

Out[58]:

	Date	Cured	Deaths	Confirmed	Active_cases
298	2021-01-01	1828546	49521	1932112	54045
299	2021-01-02	1832825	49580	1935636	53231
300	2021-01-03	1834935	49631	1938854	54288
301	2021-01-04	1836999	49666	1942136	55471
302	2021-01-05	1847361	49695	1947011	49955
...	...	...	...	...	...
481	2021-07-03	5836920	122353	6079352	120079
482	2021-07-04	5845315	122724	6088841	120802
483	2021-07-05	5848693	123030	6098177	126454
484	2021-07-06	5861720	123136	6104917	120061
485	2021-07-07	5872268	123531	6113335	117536

188 rows × 5 columns

In [59]:
```
first_wave= year_2020[(year_2020['Date'].dt.month<=12)]
first_wave
```

Out[59]:

	Date	Cured	Deaths	Confirmed	Active_cases
0	2020-03-09	0	0	2	2
1	2020-03-10	0	0	5	5
2	2020-03-11	0	0	2	2
3	2020-03-12	0	0	11	11
4	2020-03-13	0	0	14	14
...	...	...	...	...	...
293	2020-12-27	1807824	49189	1916236	59223
294	2020-12-28	1809948	49255	1919550	60347
295	2020-12-29	1814449	49305	1922048	58294
296	2020-12-30	1820021	49373	1925066	55672
297	2020-12-31	1824934	49463	1928603	54206

298 rows × 5 columns

In [60]:
```
total_1 = first_wave.sum()
total_1
```

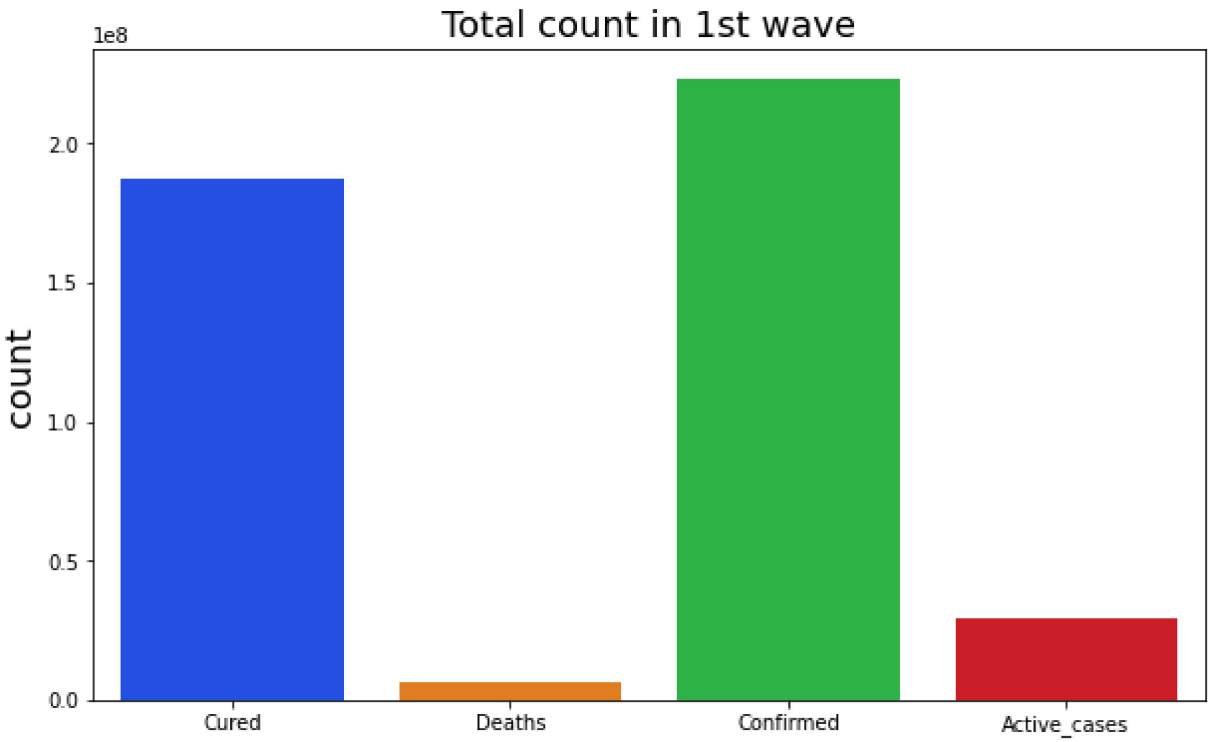
```
C:\Users\Lenovo\AppData\Local\Temp/ipykernel_5608/2509566703.py:1: FutureWarnin
g:

Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only vali
d columns before calling the reduction.
```

Out[60]:
```
Cured           187034270
Deaths            6184938
Confirmed       222900632
Active_cases     29681424
dtype: int64
```

In [61]:
```
fig = sns.barplot(x=total_1.index,y=total_1.values,palette='bright')
plt.ylabel('count',size=18)
plt.title('Total count in 1st wave',size=18)
fig.figure.set_figheight(6)
fig.figure.set_figwidth(10)

plt.show()
```



In [ ]:
```
1
```

In [62]:
```
cured_rate_1st = first_wave['Cured'].sum()/first_wave['Confirmed'].sum()*100
cured_rate_1st
```

Out[62]: 83.90925962022395

In [63]:
```
second_wave = year_2021[(year_2021['Date'].dt.month>3) & (year_2021['Date'].
second_wave
```

Out[63]:

	Date	Cured	Deaths	Confirmed	Active_cases
388	2021-04-01	2400727	54649	2812980	357604
389	2021-04-02	2433368	54898	2856163	367897
390	2021-04-03	2457494	55379	2904076	391203
391	2021-04-04	2495315	55656	2953523	402552
392	2021-04-05	2522823	55878	3010597	431896
...	...	...	...	...	...
444	2021-05-27	5241833	91341	5650907	317733
445	2021-05-28	5276203	92225	5672180	303752
446	2021-05-29	5307874	93198	5692920	291848
447	2021-05-30	5339838	94030	5713215	279347
448	2021-05-31	5362370	94844	5731815	274601

61 rows × 5 columns

In [64]:
```
total_2=second_wave.sum()
total_2
```

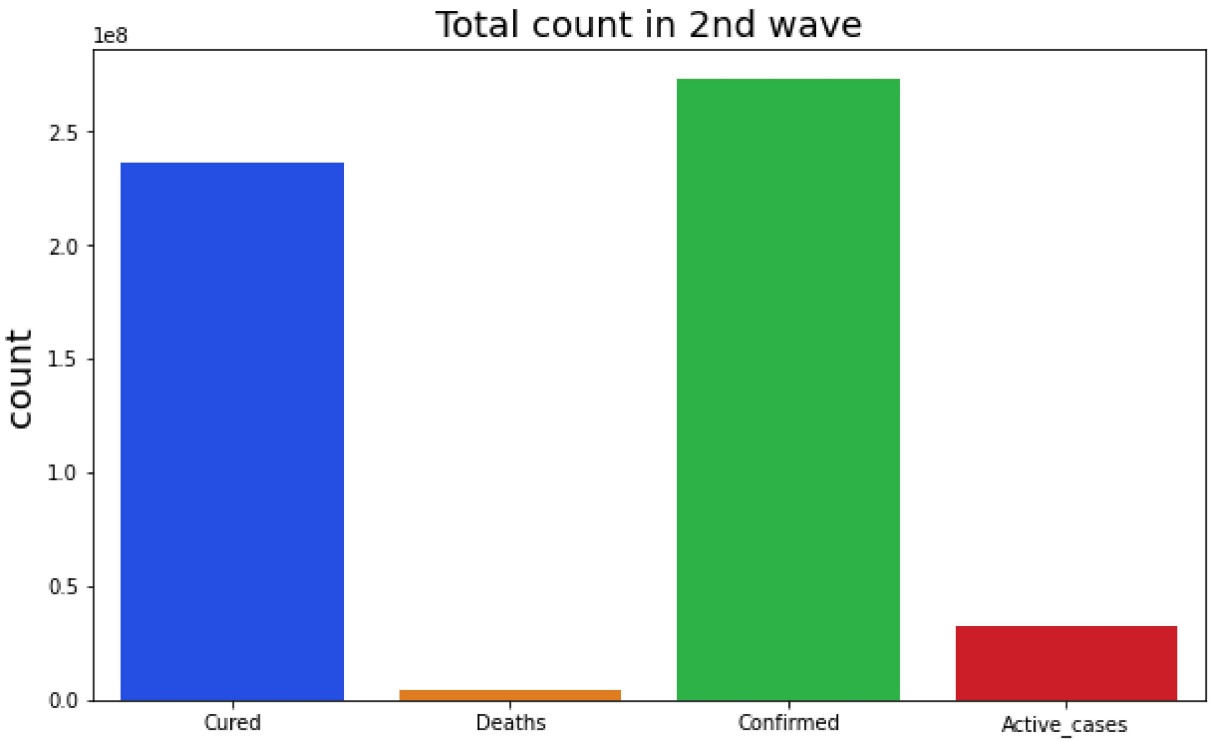
```
C:\Users\Lenovo\AppData\Local\Temp/ipykernel_5608/1753343790.py:1: FutureWarnin
g:

Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only vali
d columns before calling the reduction.
```

Out[64]:
```
Cured           235813480
Deaths            4314088
Confirmed       272653834
Active_cases     32526266
dtype: int64
```

In [65]:
```
fig = sns.barplot(x=total_2.index,y=total_2.values,palette='bright')
plt.ylabel('count',size=18)
plt.title('Total count in 2nd wave',size=18)
fig.figure.set_figheight(6)
fig.figure.set_figwidth(10)

plt.show()
```



In [66]:
```
death_rate_1st = first_wave['Deaths'].sum()/first_wave['Confirmed'].sum()*10
death_rate_2nd = second_wave['Deaths'].sum()/second_wave['Confirmed'].sum()*
print(f'1st wave rate is {death_rate_1st}\n 2nd wave rate is {death_rate_2nd
```

```
1st wave rate is 2.774751217394485
 2nd wave rate is 1.5822583298058448
```

In Maharashtra Death rate in 1st was 2.7747 which was high compare to 2nd wave

In [67]:
```
cured_rate_1st = first_wave['Cured'].sum()/first_wave['Confirmed'].sum()*100
cured_rate_2nd = second_wave['Cured'].sum()/second_wave['Confirmed'].sum()*1
print(f'1st wave rate is {cured_rate_1st}\n 2nd wave rate is {cured_rate_2nd
```

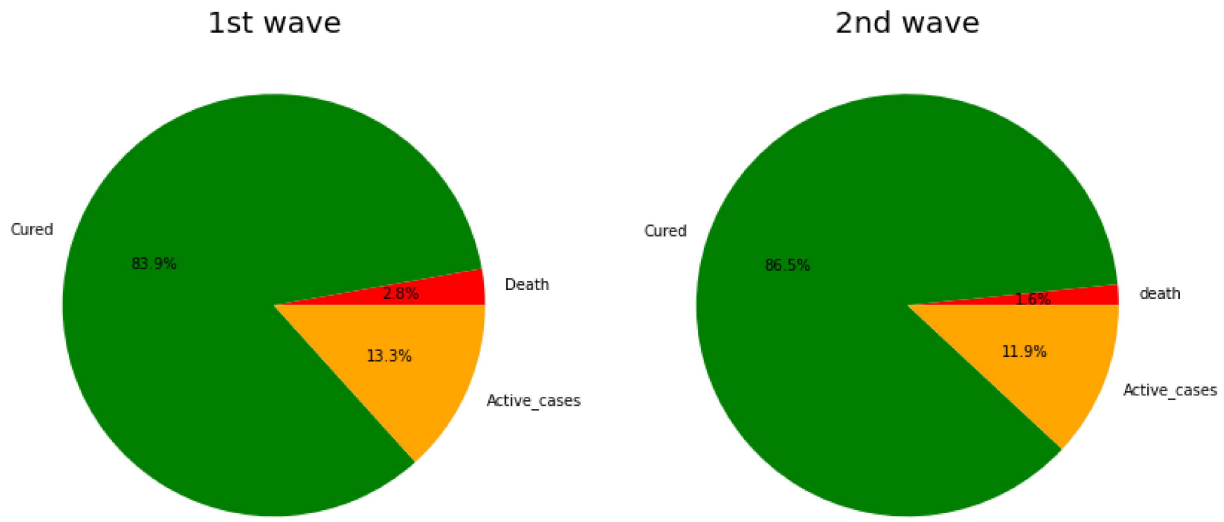
```
1st wave rate is 83.90925962022395
 2nd wave rate is 86.48823181411782
```

In Maharashtra Cured rate in 2nd was 86.488 which was high compare to 1st wave

In [68]:
```
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(14,14))
val = [first_wave['Deaths'].sum(),first_wave['Cured'].sum(),first_wave['Acti
label = ['Death','Cured','Active_cases']
ax1.pie(val,labels=label,autopct='%1.1f%%',colors=['red','green','orange'])
ax1.set_title('1st wave',size=20)

val = [second_wave['Deaths'].sum(),second_wave['Cured'].sum(),second_wave['A
label = ['death','Cured','Active_cases']
ax2.pie(val,labels=label,autopct='%1.1f%%',colors=['red','green','orange'])
ax2.set_title('2nd wave',size=20)
```

Out[68]: Text(0.5, 1.0, '2nd wave')



In [ ]:
```
1
```