In [1]:
```
# Importing libraries
```

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

In [3]:
```
df=pd.read_json('https://www.mohfw.gov.in/data/datanew.json')
# Reading dataset
```

This above cell was last run on 10-08-2022, The dataset below is as of 10-08-2022.

In [4]:
```
df
# Loading dataset
```

Out[4]:

	sno	state_name	active	positive	cured	death	new_active	new_positive	new_cured
0	1	Andaman and Nicobar Islands	34	10490	10327	129	40	10502	10333
1	2	Andhra Pradesh	1630	2333672	2317309	14733	1453	2333710	2317524
2	3	Arunachal Pradesh	322	66205	65587	296	295	66246	65655
3	4	Assam	4426	741502	729054	8022	4006	741541	729513
4	5	Bihar	1014	844858	831559	12285	1024	844997	831688
5	6	Chandigarh	769	97370	95430	1171	772	97442	95499
6	7	Chhattisgarh	3341	1169143	1151722	14080	3282	1169532	1152169
7	8	Dadra and Nagar Haveli and Daman and Diu	26	11556	11526	4	23	11557	11530
8	9	Delhi	8045	1969527	1935152	26330	7484	1970899	1937079
9	10	Goa*	1011	253042	248176	3855	1046	253162	248261
10	11	Gujarat	5895	1261261	1244388	10978	5862	1261922	1245080
11	12	Haryana	4685	1035974	1020641	10648	4598	1036795	1021547
12	13	Himachal Pradesh	5070	304629	295385	4174	4919	305383	296287
13	14	Jammu and Kashmir	5304	469749	459669	4776	5045	470201	460380
14	15	Jharkhand	897	440925	434700	5328	837	441010	434845
15	16	Karnataka	11898	4020087	3968029	40160	11252	4021106	3969691
16	17	Kerala***	10656	6729855	6648627	70582	10179	6730762	6650001
17	18	Ladakh	113	29004	28663	228	118	29021	28675

	sno	state_name	active	positive	cured	death	new_active	new_positive	new_cured
18	19	Lakshadweep	0	11415	11363	52	0	11415	11363
19	20	Madhya Pradesh	1355	1051278	1039161	10762	1349	1051447	1039336
20	21	Maharashtra	12011	8059732	7899582	148139	11968	8060737	7900626
21	22	Manipur	255	139348	136955	2138	244	139377	136995
22	23	Meghalaya	706	95909	93592	1611	686	95925	93628
23	24	Mizoram	1239	234144	232194	711	1214	234387	232461
24	25	Nagaland	62	35835	35001	772	54	35838	35012
25	26	Odisha	5904	1318875	1303823	9148	5851	1319527	1304527
26	27	Puducherry	701	171628	168960	1967	640	171651	169044
27	28	Punjab**	12155	777086	747101	17830	12429	777362	747101
28	29	Rajasthan	3438	1297262	1284234	9590	3813	1297814	1284411
29	30	Sikkim	568	42767	41725	474	476	42784	41834
30	31	Tamil Nadu	9889	3552698	3504776	38033	9408	3553670	3506229
31	32	Telangana	5910	825756	815735	4111	5667	826284	816506
32	33	Tripura	753	107344	105656	935	653	107419	105831
33	34	Uttarakhand	2584	444963	434661	7718	2378	445106	435009
34	35	Uttar Pradesh	4997	2107954	2079382	23575	5440	2108686	2079670
35	36	West Bengal	7847	2099056	2069814	21395	7302	2099433	2070731
36	37		135510	44161899	43499659	526740	131807	44174650	43516071

DataUnderstanding

In [5]:
```
df.shape
# There are 37 rows and 11 columns
```

Out[5]:
```
(37, 14)
```

In [6]:
```
df.isnull().sum()
# no null values
```

Out[6]:
```
sno                0
state_name         0
active             0
positive           0
cured              0
death              0
new_active         0
new_positive       0
new_cured          0
new_death          0
death_reconsille   0
total              0
state_code         0
```

```
actualdeath24hrs    0
dtype: int64
```

In [7]:
```
df.info()
# type of data present in the columns(int,object)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37 entries, 0 to 36
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   sno               37 non-null     int64 
 1   state_name        37 non-null     object
 2   active            37 non-null     int64 
 3   positive          37 non-null     int64 
 4   cured             37 non-null     int64 
 5   death             37 non-null     int64 
 6   new_active        37 non-null     int64 
 7   new_positive      37 non-null     int64 
 8   new_cured         37 non-null     int64 
 9   new_death         37 non-null     int64 
 10  death_reconsille  37 non-null     object
 11  total             37 non-null     object
 12  state_code        37 non-null     int64 
 13  actualdeath24hrs  37 non-null     int64 
dtypes: int64(11), object(3)
memory usage: 4.2+ KB
```

In [8]:
```
df.describe()
# Viewing the descriptive statistics of the data like mean, std deviation, min and m
```

Out[8]:

	sno	active	positive	cured	death	new_active	new_p
count	37.000000	37.000000	3.700000e+01	3.700000e+01	37.000000	37.000000	3.7000(
mean	19.000000	7324.864865	2.387130e+06	2.351333e+06	28472.432432	7124.702703	2.3878
std	10.824355	21994.360622	7.281370e+06	7.171866e+06	88317.856831	21395.312638	7.2834(
min	1.000000	0.000000	1.049000e+04	1.032700e+04	4.000000	0.000000	1.0502(
25%	10.000000	706.000000	1.073440e+05	1.056560e+05	1171.000000	653.000000	1.0741
50%	19.000000	2584.000000	7.415020e+05	7.290540e+05	7718.000000	2378.000000	7.4154
75%	28.000000	5904.000000	1.318875e+06	1.303823e+06	14733.000000	5851.000000	1.3195
max	37.000000	135510.000000	4.416190e+07	4.349966e+07	526740.000000	131807.000000	4.4174(

In [9]:
```
df.keys()
# Displaying all the column names present in data
```

Out[9]:
```
Index(['sno', 'state_name', 'active', 'positive', 'cured', 'death',
       'new_active', 'new_positive', 'new_cured', 'new_death',
       'death_reconsille', 'total', 'state_code', 'actualdeath24hrs'],
      dtype='object')
```

In [10]:
```
df.columns
# Different code but same purpose as above that is displaying all column names
```

Out[10]:
```
Index(['sno', 'state_name', 'active', 'positive', 'cured', 'death',
       'new_active', 'new_positive', 'new_cured', 'new_death',
```

```
       'death_reconsille', 'total', 'state_code', 'actualdeath24hrs'],
      dtype='object')
```

In [11]:
```
df.nunique()
# The nunique() function is used to count distinct observations over requested axis.
```

Out[11]:
```
sno                37
state_name         37
active             37
positive           37
cured              37
death              37
new_active         37
new_positive       37
new_cured          37
new_death          37
death_reconsille    2
total               9
state_code         37
actualdeath24hrs    8
dtype: int64
```

In [12]:
```
df.tail()
# As we can see there is an extra row in the end that may be the total of that colum
```

Out[12]:

	sno	state_name	active	positive	cured	death	new_active	new_positive	new_cured	n
32	33	Tripura	753	107344	105656	935	653	107419	105831	
33	34	Uttarakhand	2584	444963	434661	7718	2378	445106	435009	
34	35	Uttar Pradesh	4997	2107954	2079382	23575	5440	2108686	2079670	
35	36	West Bengal	7847	2099056	2069814	21395	7302	2099433	2070731	
36	37		135510	44161899	43499659	526740	131807	44174650	43516071	

In [13]:
```
df=df.iloc[:36]
# we apply the iloc function and select all rows except the last row
```

In [14]:
```
df
# This is the dataset that we are to work on
```

Out[14]:

	sno	state_name	active	positive	cured	death	new_active	new_positive	new_cured	nev
0	1	Andaman and Nicobar Islands	34	10490	10327	129	40	10502	10333	
1	2	Andhra Pradesh	1630	2333672	2317309	14733	1453	2333710	2317524	
2	3	Arunachal Pradesh	322	66205	65587	296	295	66246	65655	
3	4	Assam	4426	741502	729054	8022	4006	741541	729513	
4	5	Bihar	1014	844858	831559	12285	1024	844997	831688	
5	6	Chandigarh	769	97370	95430	1171	772	97442	95499	

	sno	state_name	active	positive	cured	death	new_active	new_positive	new_cured	nev
6	7	Chhattisgarh	3341	1169143	1151722	14080	3282	1169532	1152169	
7	8	Dadra and Nagar Haveli and Daman and Diu	26	11556	11526	4	23	11557	11530	
8	9	Delhi	8045	1969527	1935152	26330	7484	1970899	1937079	
9	10	Goa*	1011	253042	248176	3855	1046	253162	248261	
10	11	Gujarat	5895	1261261	1244388	10978	5862	1261922	1245080	
11	12	Haryana	4685	1035974	1020641	10648	4598	1036795	1021547	
12	13	Himachal Pradesh	5070	304629	295385	4174	4919	305383	296287	
13	14	Jammu and Kashmir	5304	469749	459669	4776	5045	470201	460380	
14	15	Jharkhand	897	440925	434700	5328	837	441010	434845	
15	16	Karnataka	11898	4020087	3968029	40160	11252	4021106	3969691	
16	17	Kerala***	10656	6729855	6648627	70582	10179	6730762	6650001	
17	18	Ladakh	113	29004	28663	228	118	29021	28675	
18	19	Lakshadweep	0	11415	11363	52	0	11415	11363	
19	20	Madhya Pradesh	1355	1051278	1039161	10762	1349	1051447	1039336	
20	21	Maharashtra	12011	8059732	7899582	148139	11968	8060737	7900626	
21	22	Manipur	255	139348	136955	2138	244	139377	136995	
22	23	Meghalaya	706	95909	93592	1611	686	95925	93628	
23	24	Mizoram	1239	234144	232194	711	1214	234387	232461	
24	25	Nagaland	62	35835	35001	772	54	35838	35012	
25	26	Odisha	5904	1318875	1303823	9148	5851	1319527	1304527	
26	27	Puducherry	701	171628	168960	1967	640	171651	169044	
27	28	Punjab**	12155	777086	747101	17830	12429	777362	747101	
28	29	Rajasthan	3438	1297262	1284234	9590	3813	1297814	1284411	
29	30	Sikkim	568	42767	41725	474	476	42784	41834	
30	31	Tamil Nadu	9889	3552698	3504776	38033	9408	3553670	3506229	
31	32	Telangana	5910	825756	815735	4111	5667	826284	816506	
32	33	Tripura	753	107344	105656	935	653	107419	105831	
33	34	Uttarakhand	2584	444963	434661	7718	2378	445106	435009	
34	35	Uttar Pradesh	4997	2107954	2079382	23575	5440	2108686	2079670	
35	36	West Bengal	7847	2099056	2069814	21395	7302	2099433	2070731	

In [15]:

```
df['active'].sum(axis = 0)
# The total number of active cases in India is 2,22,526
```

Out[15]: 135510

In [16]:
```
df['positive'].sum(axis = 0)
# The total number of positive cases in India are 10,466,595
```

Out[16]: 44161899

In [17]:
```
df['cured'].sum(axis = 0)
# The total number of cured cases in India is 10,092,909
```

Out[17]: 43499659

In [18]:
```
df['death'].sum(axis = 0)
# The total number of deaths in India are 1,51,160
```

Out[18]: 526740

In [19]:
```
df['new_active'].sum(axis = 0)
# The total number of new active cases are 2,16,558
```

Out[19]: 131807

In [20]:
```
df['new_positive'].sum(axis = 0)
# The total number of new positive cases are 10,479,179
```

Out[20]: 44174650

In [21]:
```
df['new_cured'].sum(axis = 0)
# The total number of new cured cases are 10,111,294
```

Out[21]: 43516071

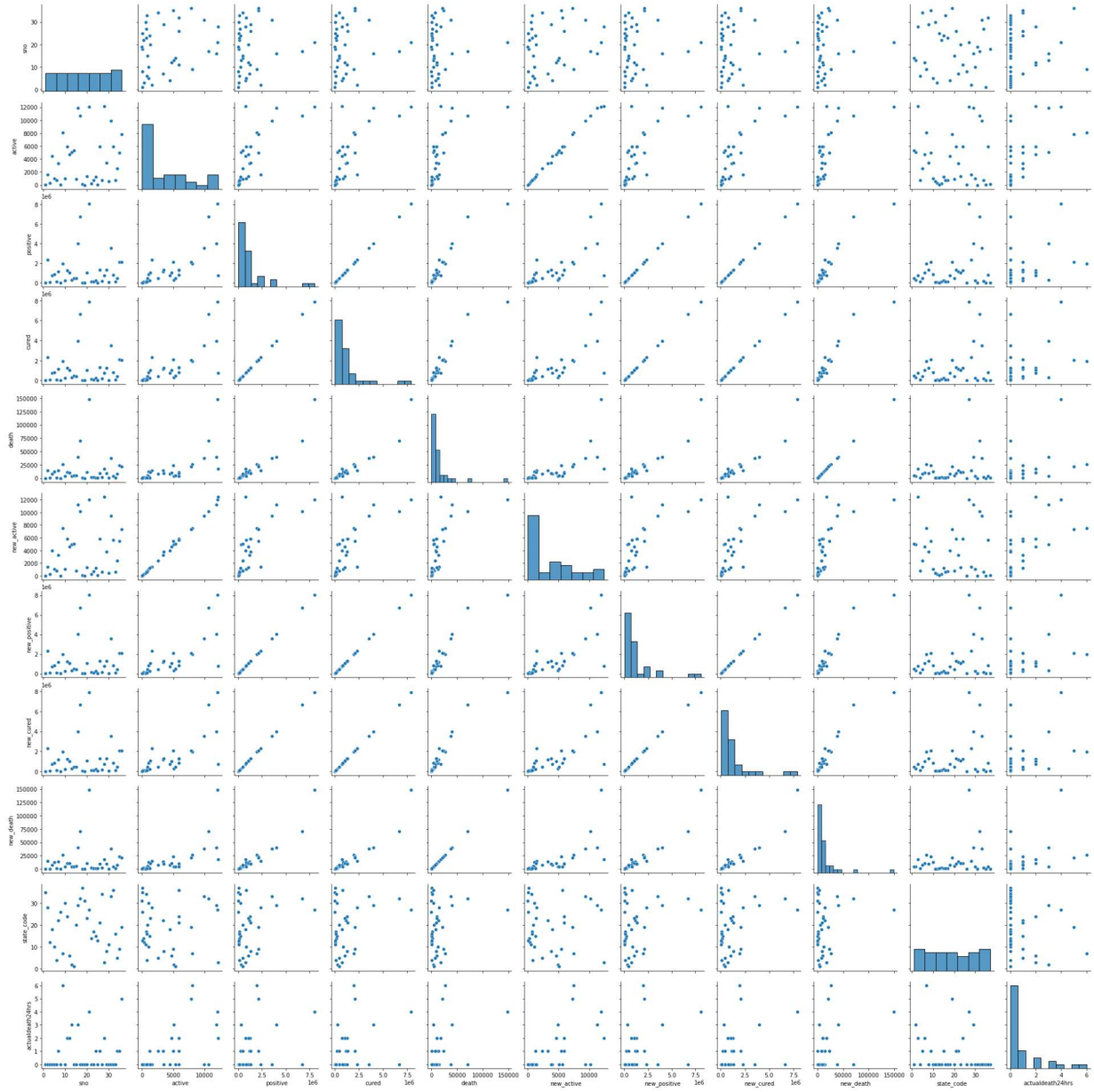
In [22]:
```
df['new_death'].sum(axis = 0)
# The total number of new death cases are 1,51,327
```

Out[22]: 526772

Data Visualization

In [23]:
```
sns.pairplot(data=df)
# plotting scatter plots of all data using pairplot
```

Out[23]: <seaborn.axisgrid.PairGrid at 0x1b088ea45b0>



In [24]:
```
cases_df=df.sum()
# Storing total cases in cases_df
```

In [25]:
```
cases_df
# Subset of original dataset
```

Out[25]:
```
sno                                                           666
state_name          Andaman and Nicobar IslandsAndhra PradeshAruna...
active                                                     135510
positive                                                 44161899
cured                                                    43499659
death                                                      526740
new_active                                                 131807
new_positive                                             44174650
new_cured                                                43516071
new_death                                                  526772
death_reconsille                                               10
total                                            162233104112115
state_code                                                   678
actualdeath24hrs                                              32
dtype: object
```

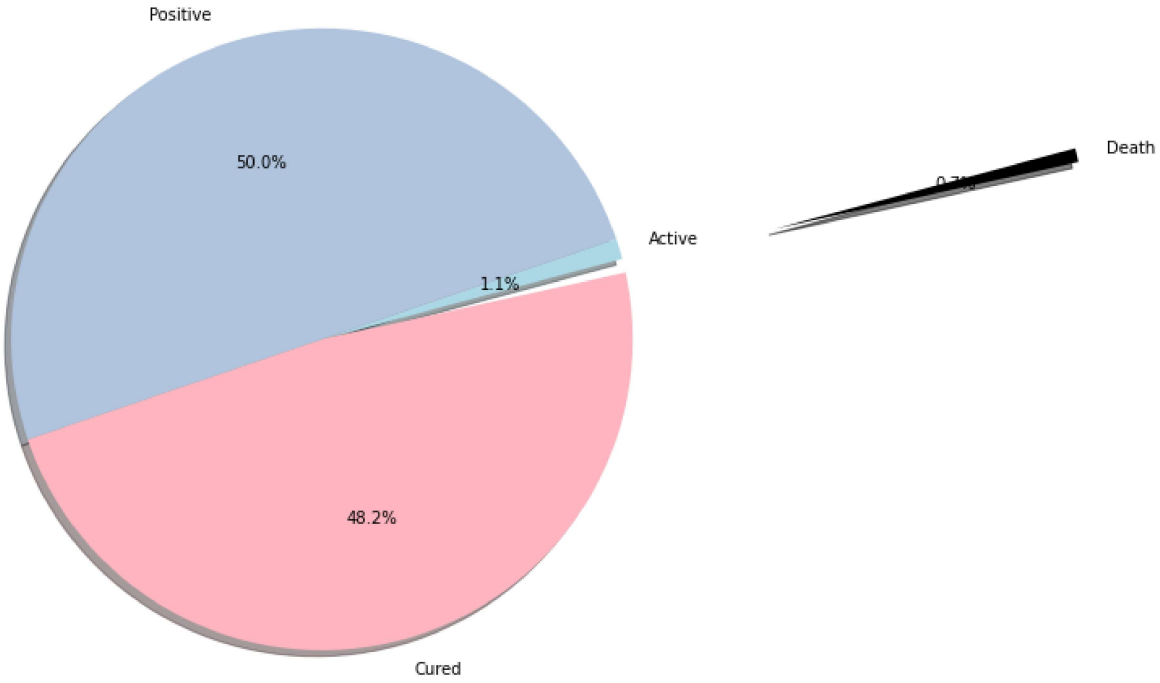
In [26]:
```
cases_df.drop(['sno','state_name','state_code'],inplace=True)
```

```
# Dropping unnecessary columns
```

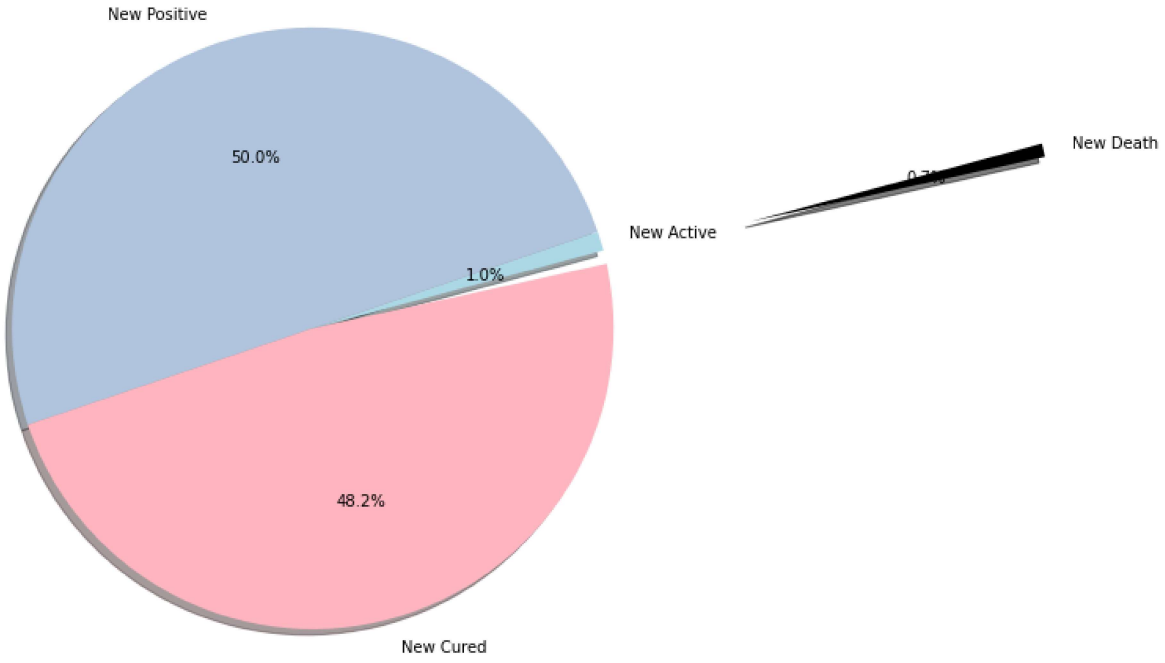
In [27]:
```
cases_df
# Name of case types and their total number
```

Out[27]:
```
active                     135510
positive                 44161899
cured                    43499659
death                      526740
new_active                 131807
new_positive             44174650
new_cured                43516071
new_death                  526772
death_reconsille               10
total              162233104112115
actualdeath24hrs               32
dtype: object
```

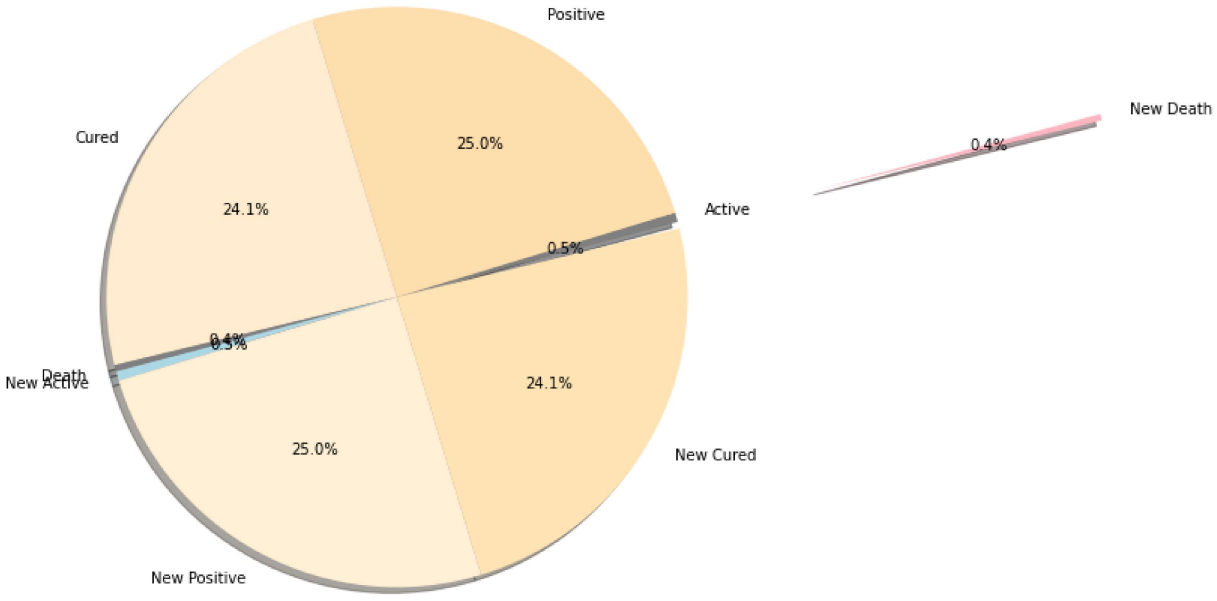
In [28]:
```
my_data = [222526,10466595,10092909,151160]
my_labels = 'Active','Positive','Cured','Death'
my_explode = (0,0,0,1.5)
my_colors = ['lightblue','lightsteelblue','lightpink','black']
fig1, ax1 = plt.subplots(figsize=(13, 8))
plt.pie(my_data, labels=my_labels, autopct='%1.1f%%', startangle=15, shadow = True,
plt.axis('equal')
plt.show()
# Pie chart visualization
```



In [29]:
```
my_data = [216558,10479179,10111294,151327]
my_labels = 'New Active','New Positive','New Cured','New Death'
my_explode = (0,0,0,1.5)
my_colors = ['lightblue','lightsteelblue','lightpink','black']
fig1, ax1 = plt.subplots(figsize=(13, 8))
plt.pie(my_data, labels=my_labels, autopct='%1.1f%%', startangle=15, shadow = True,
plt.axis('equal')
plt.show()
# Pie chart visualization
```



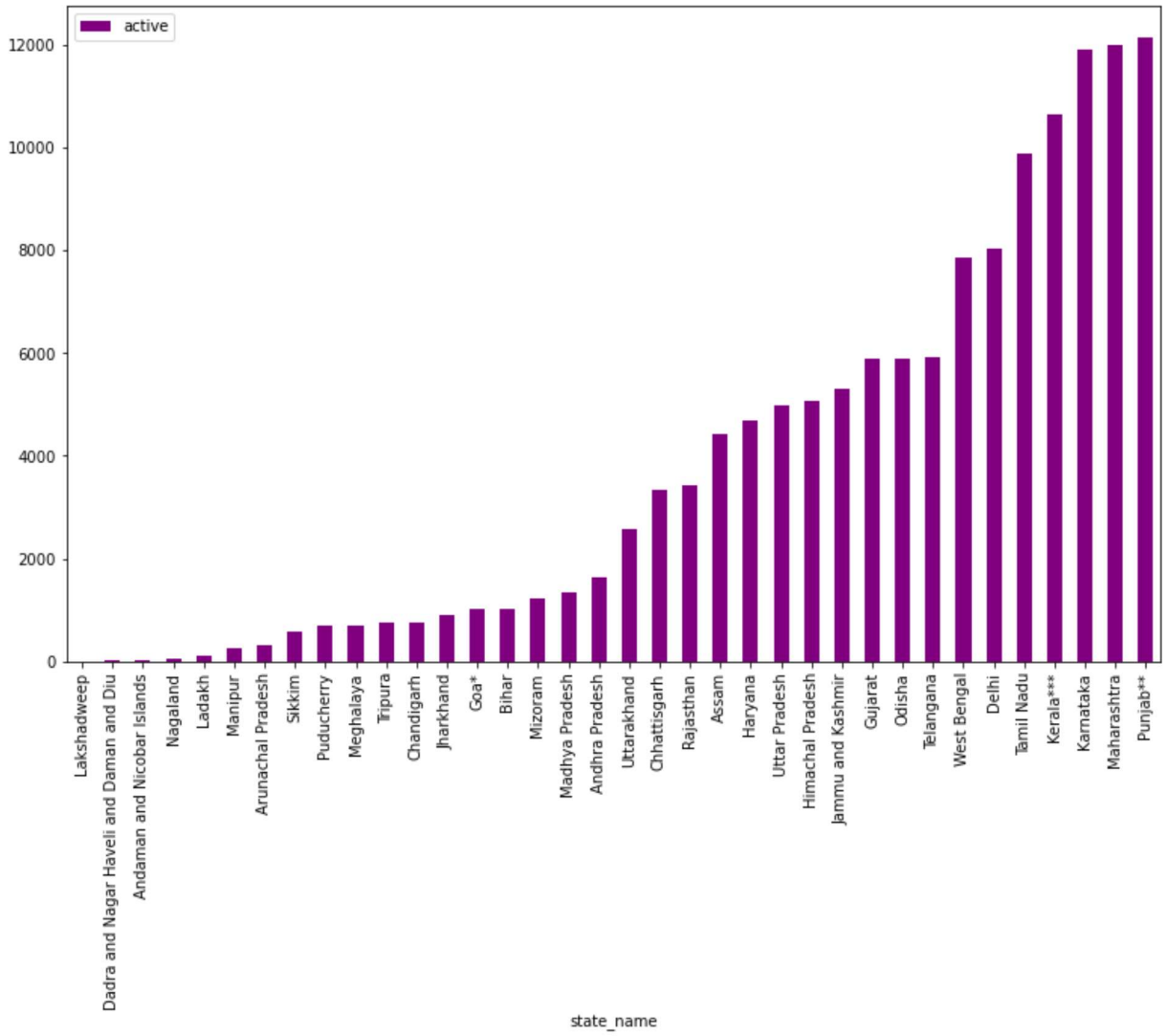
In [30]:
```
my_data = [222526,10466595,10092909,151160,216558,10479179,10111294,151327]
my_labels = 'Active','Positive','Cured','Death','New Active','New Positive','New Cur
my_explode = (0,0,0,0,0,0,0,1.5)
my_colors = ['gray','navajowhite','blanchedalmond','grey','lightblue','papayawhip','
fig1, ax1 = plt.subplots(figsize=(13, 8))
plt.pie(my_data, labels=my_labels, autopct='%1.1f%%', startangle=15, shadow = True,
plt.axis('equal')
plt.show()
# Pie chart visualization
```



In [31]:
```
plt.rcParams['figure.figsize']=(13,8)
# giving figure size
```

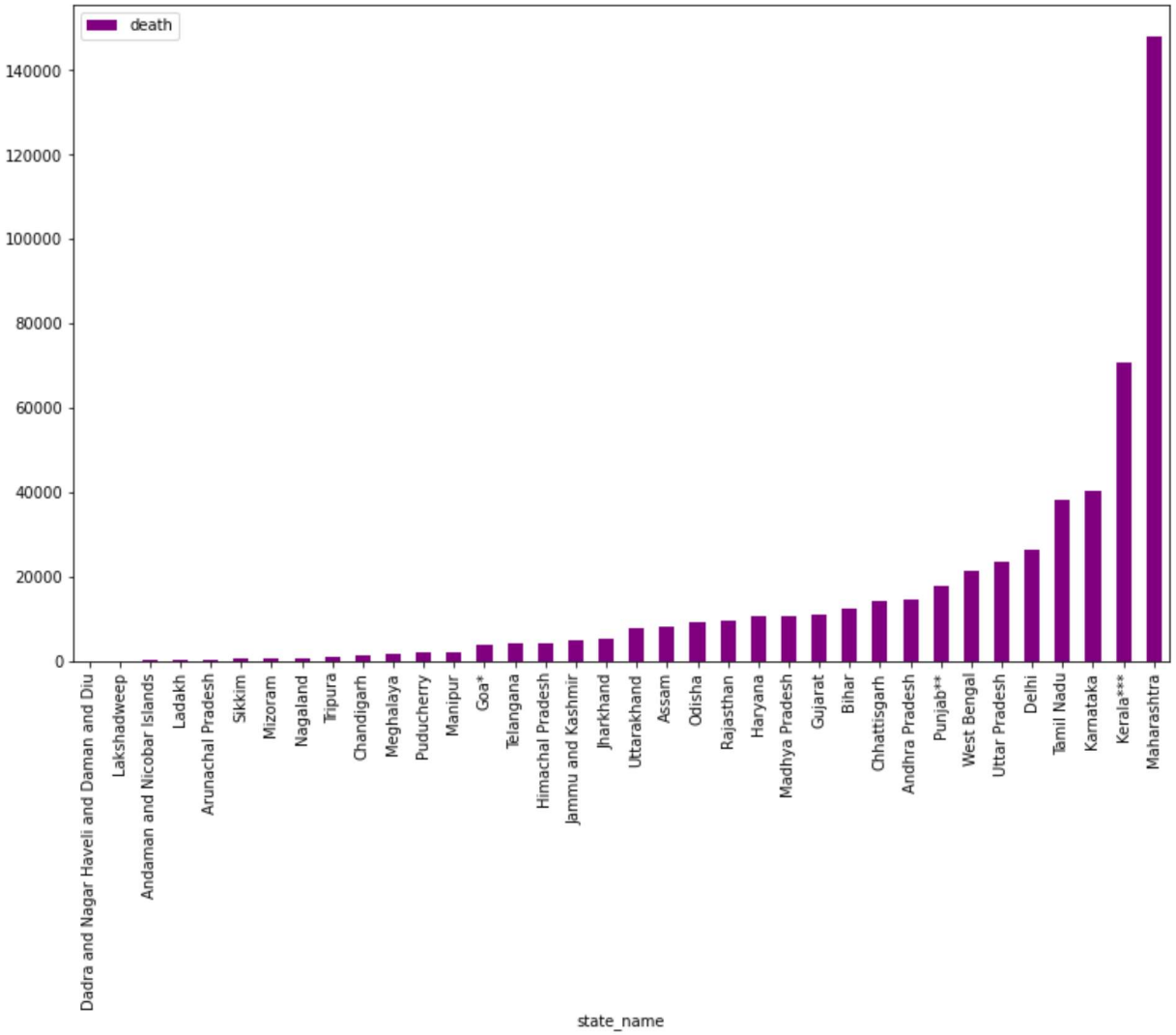
In [32]:
```
df[['state_name','active']].groupby(["state_name"]).mean().sort_values(by='active').
plt.show()
# We can also use the groupby function to sort values in an ascending order based on
```

```
# Below we get a clear picture of the states in an increasing order based on their a
# Kerela has a higher active cases compared to other states
```



In [33]:

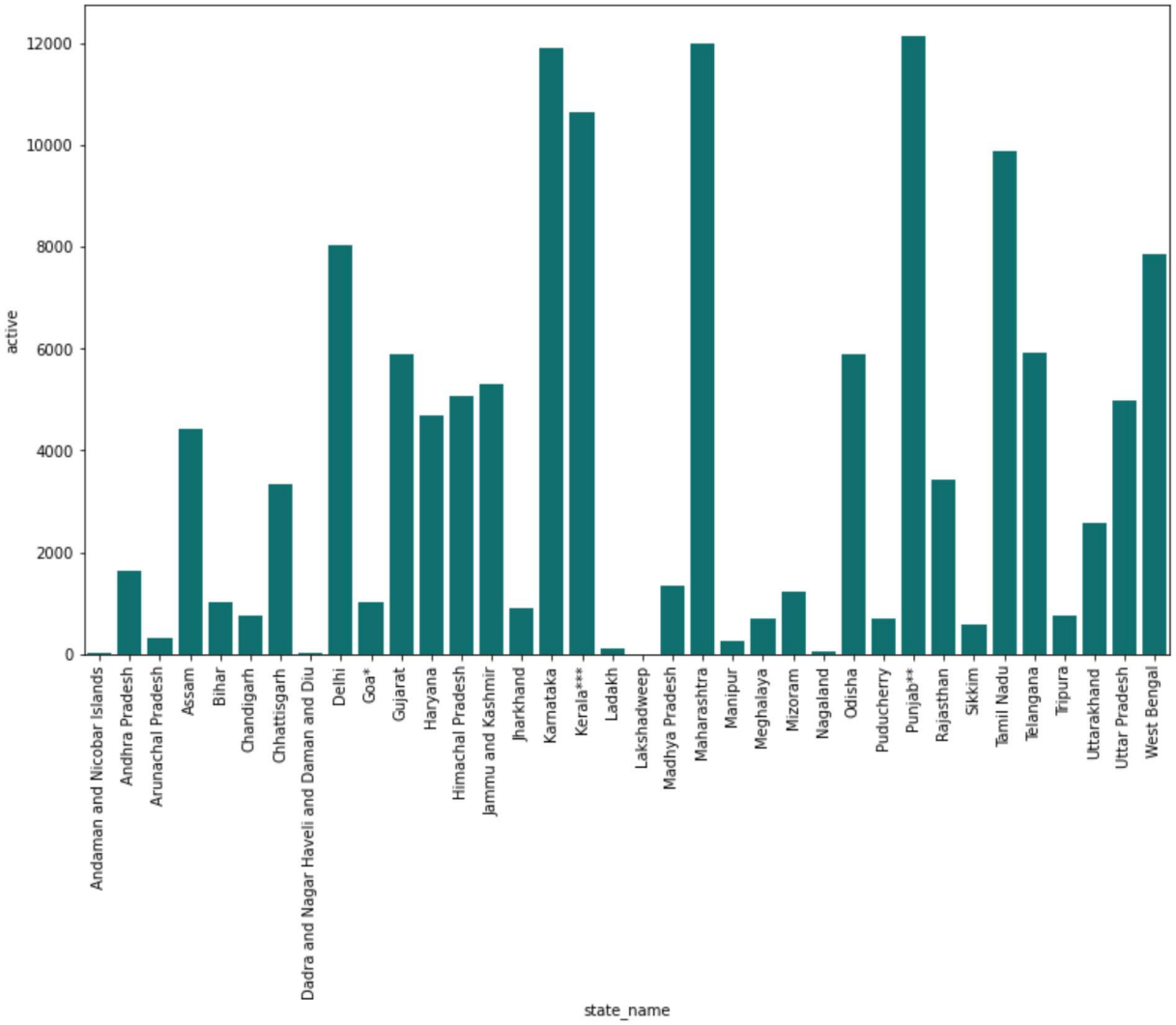
```
df[['state_name','death']].groupby(["state_name"]).mean().sort_values(by='death').pl
plt.show()
# We can also use the groupby function to sort values in an ascending order based on
# Below we get a clear picture of the states in an increasing order based on their d
# Maharashtra has a higher death cases compared to other states
```



In [34]:
```
df.columns
# Displaying column names
```

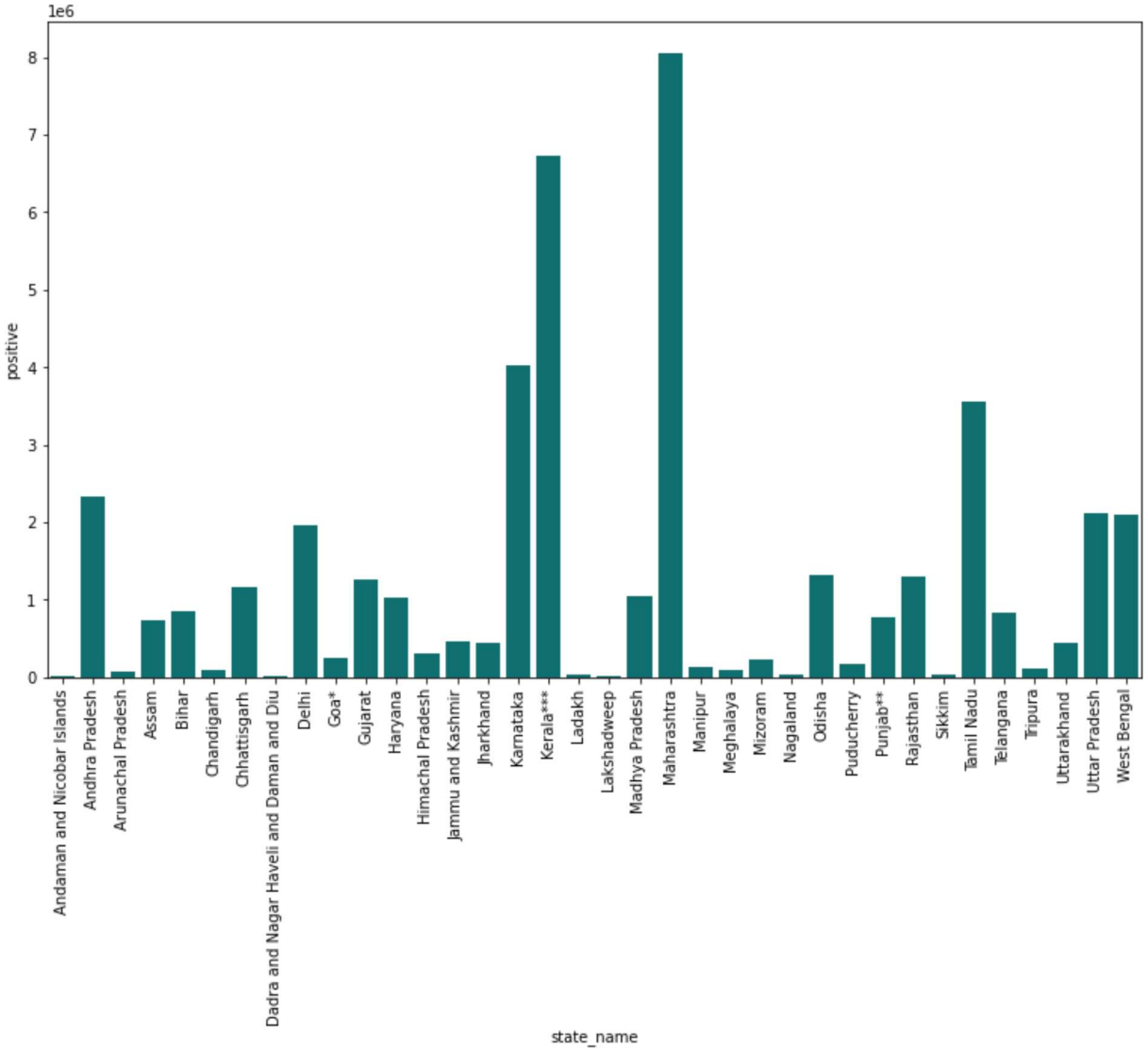
Out[34]:
```
Index(['sno', 'state_name', 'active', 'positive', 'cured', 'death',
       'new_active', 'new_positive', 'new_cured', 'new_death',
       'death_reconsille', 'total', 'state_code', 'actualdeath24hrs'],
      dtype='object')
```

In [35]:
```
plt.figure(figsize=(13, 8))
plt.xticks(rotation=90)
sns.barplot(x='state_name',y='active',color='teal',data=df);
# The visualization below shows us that high number of active cases are found in ker
```



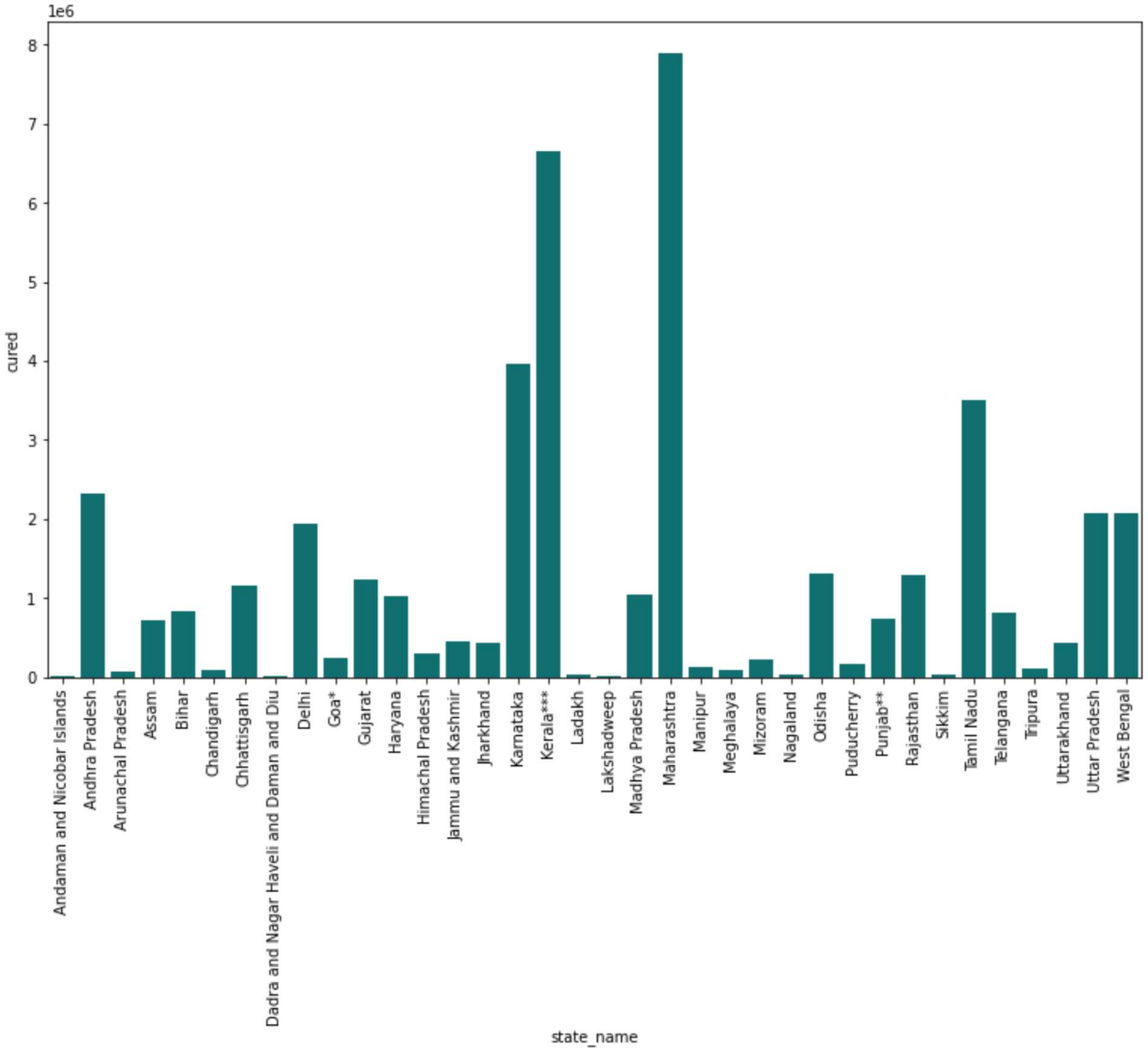
In [36]:

```
plt.figure(figsize=(13, 8))
plt.xticks(rotation=90)
sns.barplot(x='state_name',y='positive',color='teal',data=df);
# The below visualization shows us that high number of positive cases can be seen in
```



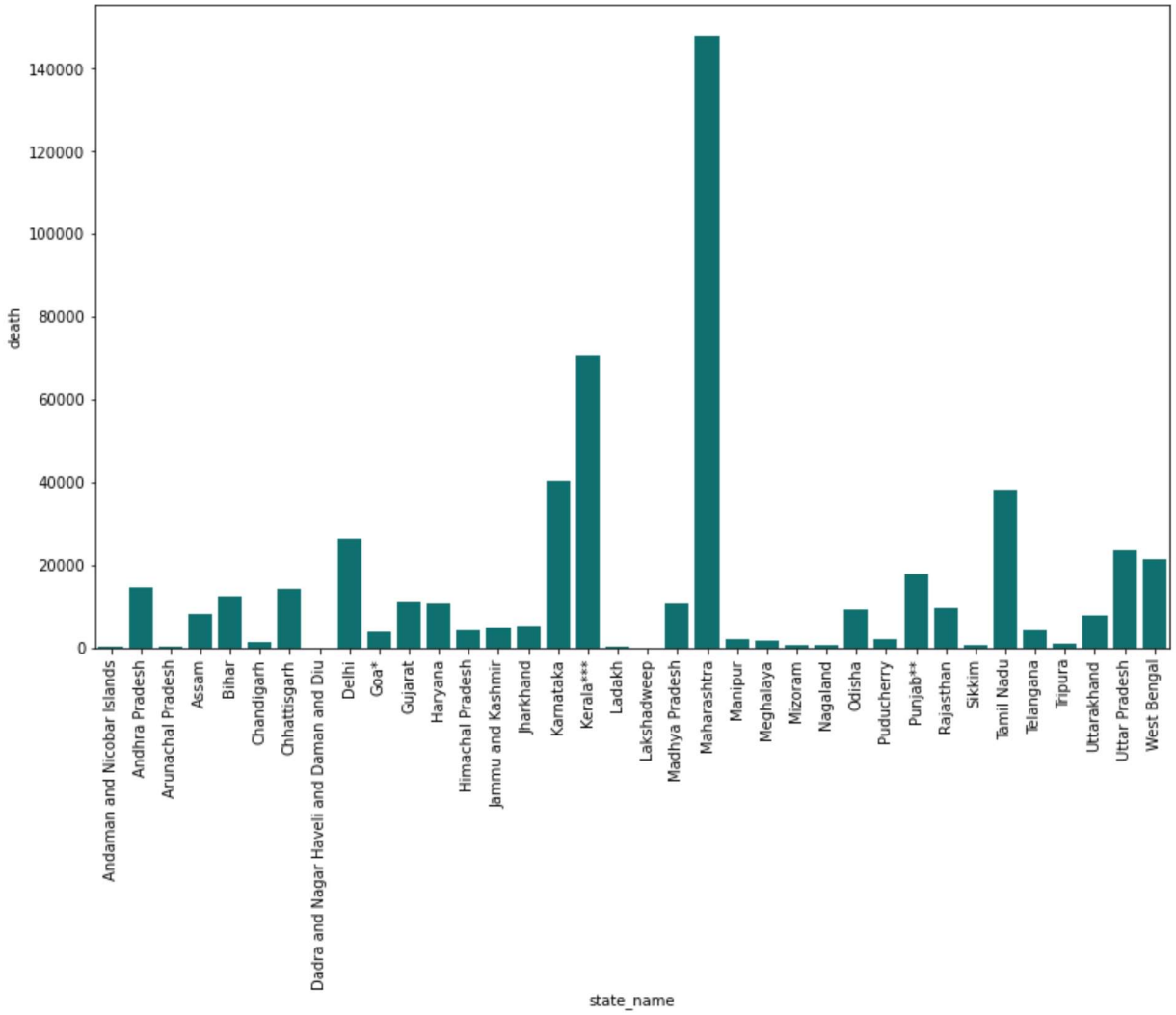
In [37]:

```
plt.figure(figsize=(13, 8))
plt.xticks(rotation=90)
sns.barplot(x='state_name',y='cured',color='teal',data=df);
# The visualization below shows us that high number of people have cured in the stat
```

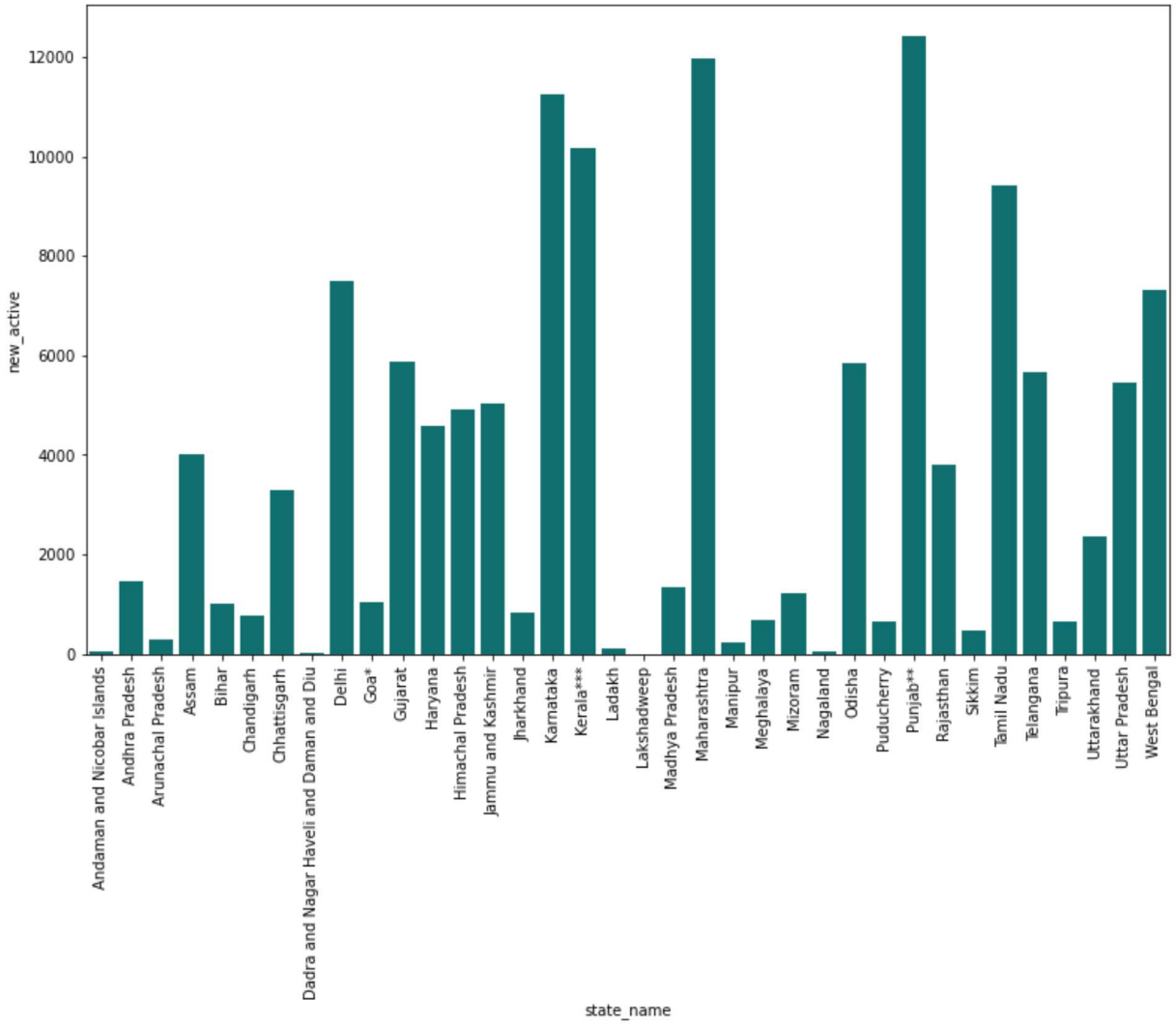


In [38]:

```
plt.figure(figsize=(13, 8))
plt.xticks(rotation=90)
sns.barplot(x='state_name',y='death',color='teal',data=df);
# The below visualization shows us that high number of deaths taking place in the st
```

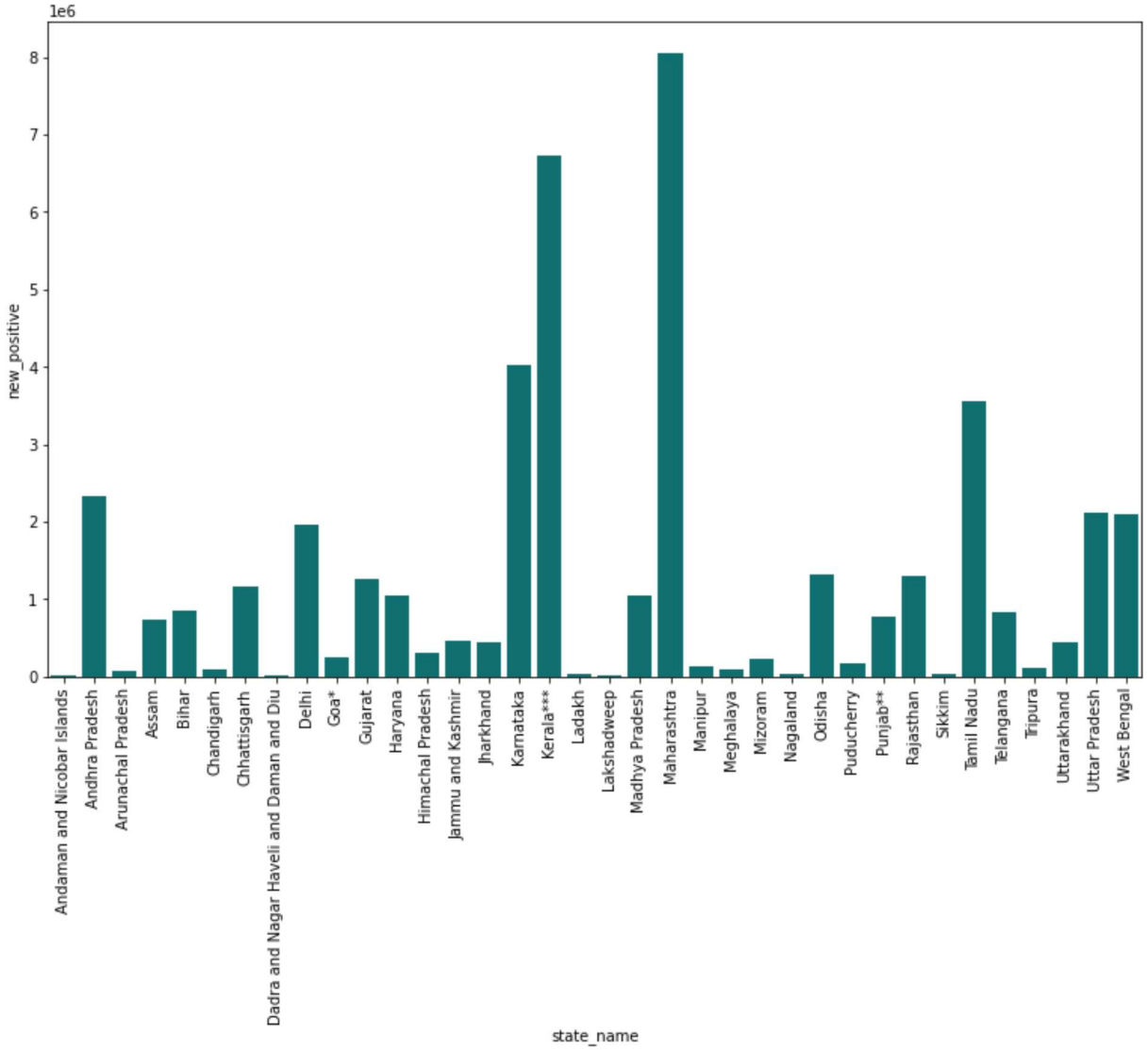


In [39]:
```
plt.figure(figsize=(13, 8))
plt.xticks(rotation=90)
sns.barplot(x='state_name',y='new_active',color='teal',data=df);
# The following visualization shows us the total number of new cases and kerela has
```



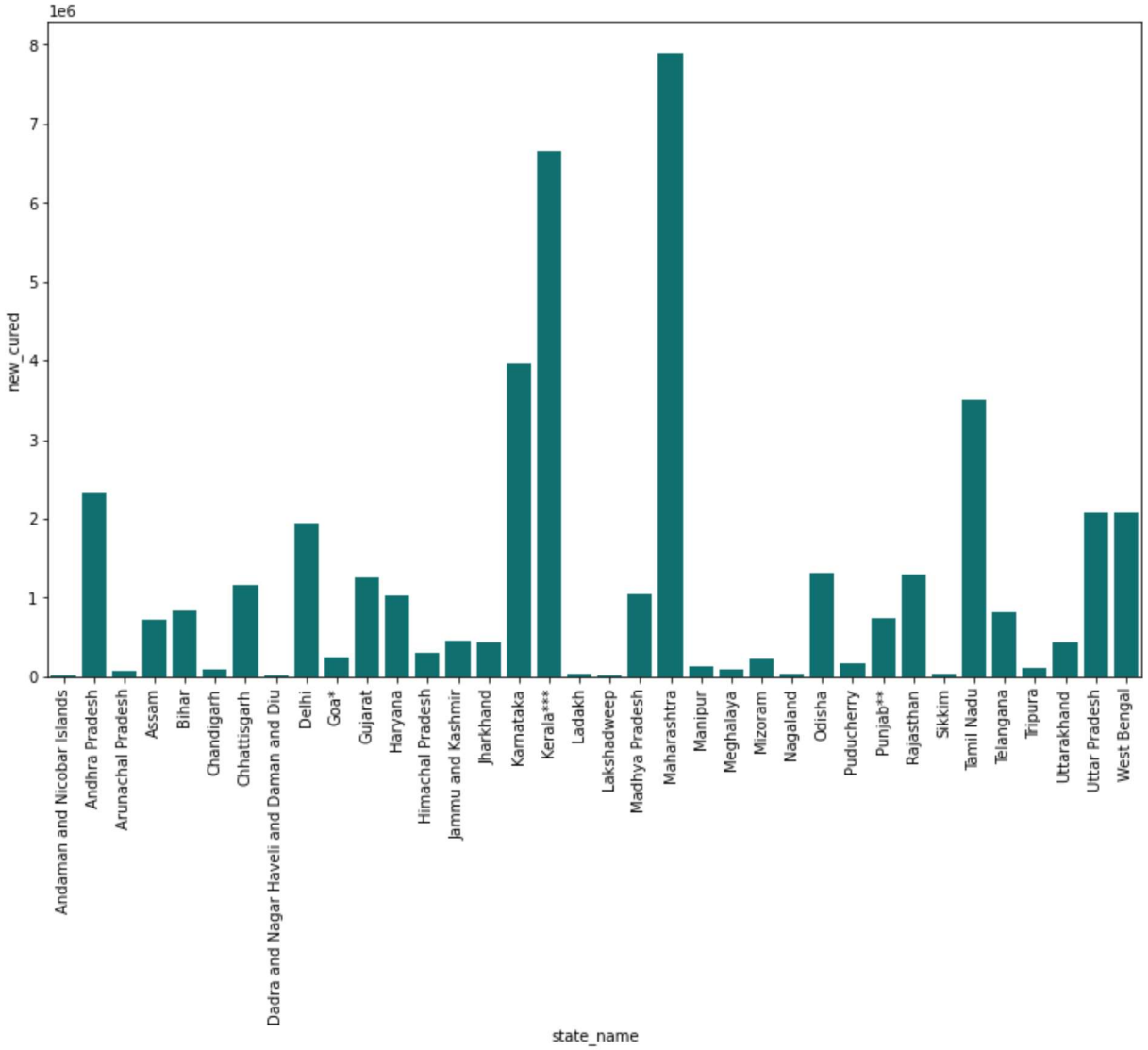
In [40]:

```
plt.figure(figsize=(13, 8))
plt.xticks(rotation=90)
sns.barplot(x='state_name',y='new_positive',color='teal',data=df);
# The following visualization shows us the total number of new positive cases and Ma
```



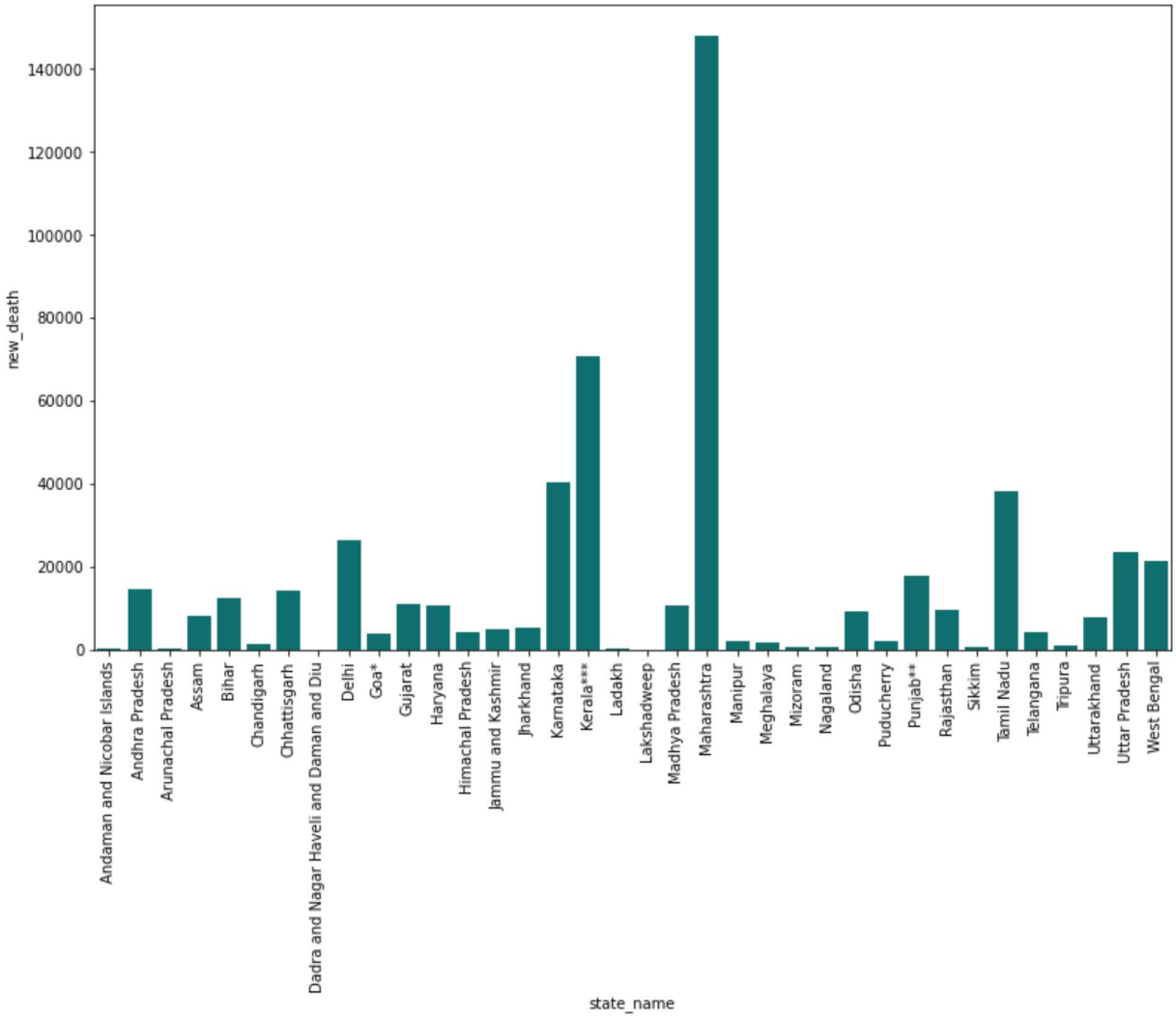
In [41]:

```
plt.figure(figsize=(13, 8))
plt.xticks(rotation=90)
sns.barplot(x='state_name',y='new_cured',color='teal',data=df);
# The following visualization shows us the total number of new cured cases and Mahar
```



In [42]:

```
plt.figure(figsize=(13, 8))
plt.xticks(rotation=90)
sns.barplot(x='state_name',y='new_death',color='teal',data=df);
# Maharashtra has the highest new death rates in India followed by Karnataka
```



In [43]:
```
df.columns
# column names
```

Out[43]:
```
Index(['sno', 'state_name', 'active', 'positive', 'cured', 'death',
       'new_active', 'new_positive', 'new_cured', 'new_death',
       'death_reconsille', 'total', 'state_code', 'actualdeath24hrs'],
      dtype='object')
```

In [44]:
```
df.drop(['sno','state_code'],axis=1,inplace=True)
# we have dropped columns like sno and statecode
```

```
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\pandas\core\frame.py:4308: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

In [45]:
```
df.head(5)
# This is the resulting data
```

Out[45]:

	state_name	active	positive	cured	death	new_active	new_positive	new_cured	new_death
0	Andaman and Nicobar Islands	34	10490	10327	129	40	10502	10333	129

	state_name	active	positive	cured	death	new_active	new_positive	new_cured	new_death
1	Andhra Pradesh	1630	2333672	2317309	14733	1453	2333710	2317524	14733
2	Arunachal Pradesh	322	66205	65587	296	295	66246	65655	296
3	Assam	4426	741502	729054	8022	4006	741541	729513	8022
4	Bihar	1014	844858	831559	12285	1024	844997	831688	12285

In [46]:
```
df=df.set_index('state_name')
# Setting index as statename
```

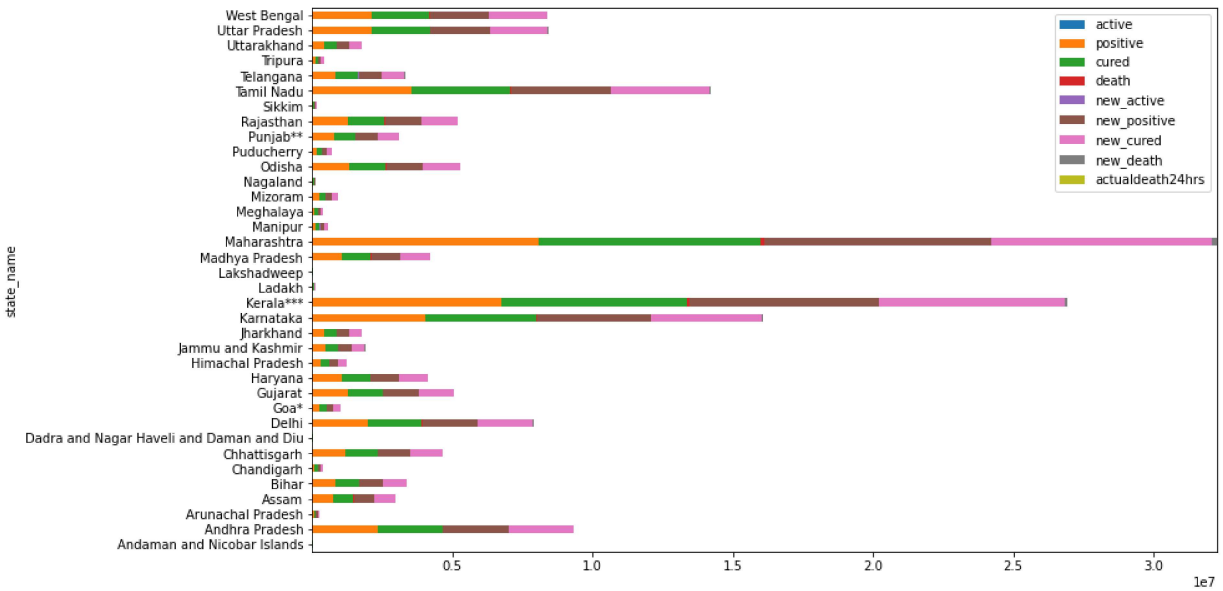
In [47]:
```
df.head(5)
```

Out[47]:

state_name	active	positive	cured	death	new_active	new_positive	new_cured	new_death	dea
Andaman and Nicobar Islands	34	10490	10327	129	40	10502	10333	129	
Andhra Pradesh	1630	2333672	2317309	14733	1453	2333710	2317524	14733	
Arunachal Pradesh	322	66205	65587	296	295	66246	65655	296	
Assam	4426	741502	729054	8022	4006	741541	729513	8022	
Bihar	1014	844858	831559	12285	1024	844997	831688	12285	

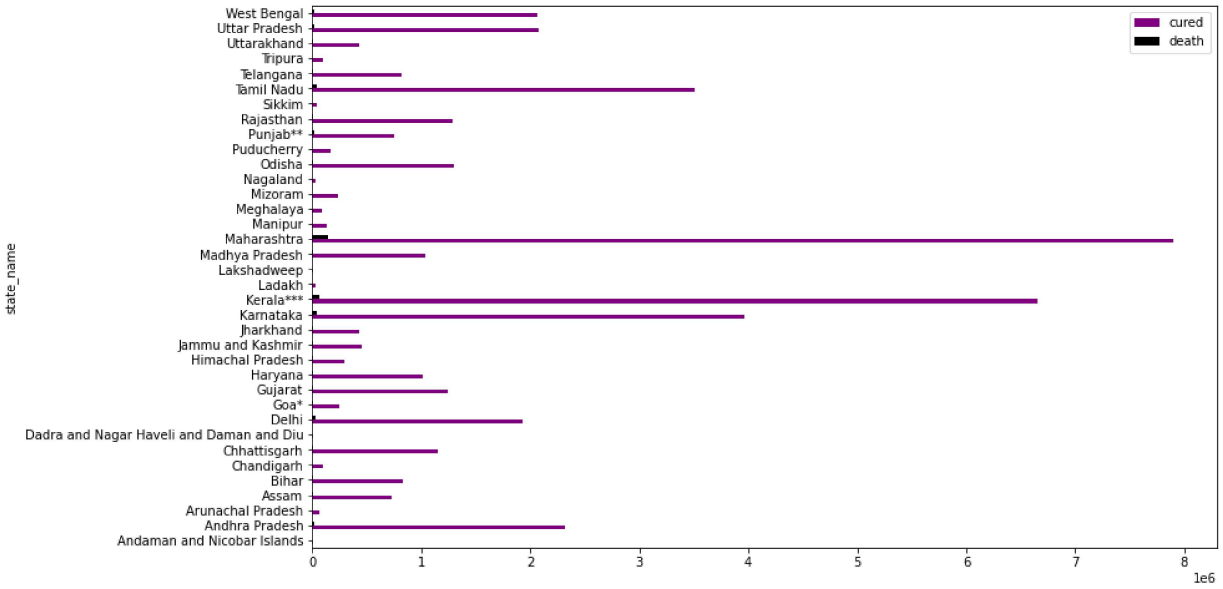
In [48]:
```
df.plot.barh(stacked=True,figsize=(13,8))
# Stacked bar plot
```

Out[48]: <AxesSubplot:ylabel='state_name'>



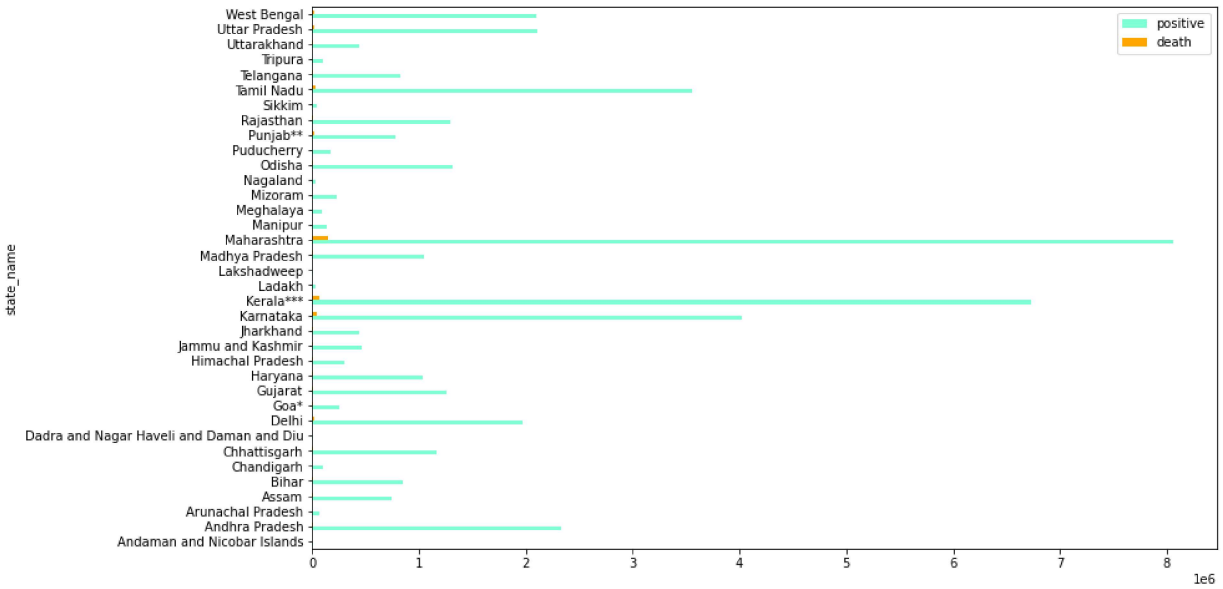
In [49]:
```
df1=df[['cured', 'death']]
df1.plot.barh(color={"cured": "purple", "death": "black"},figsize=(13,8))
# bar plot for cured and death
```

Out[49]: <AxesSubplot:ylabel='state_name'>



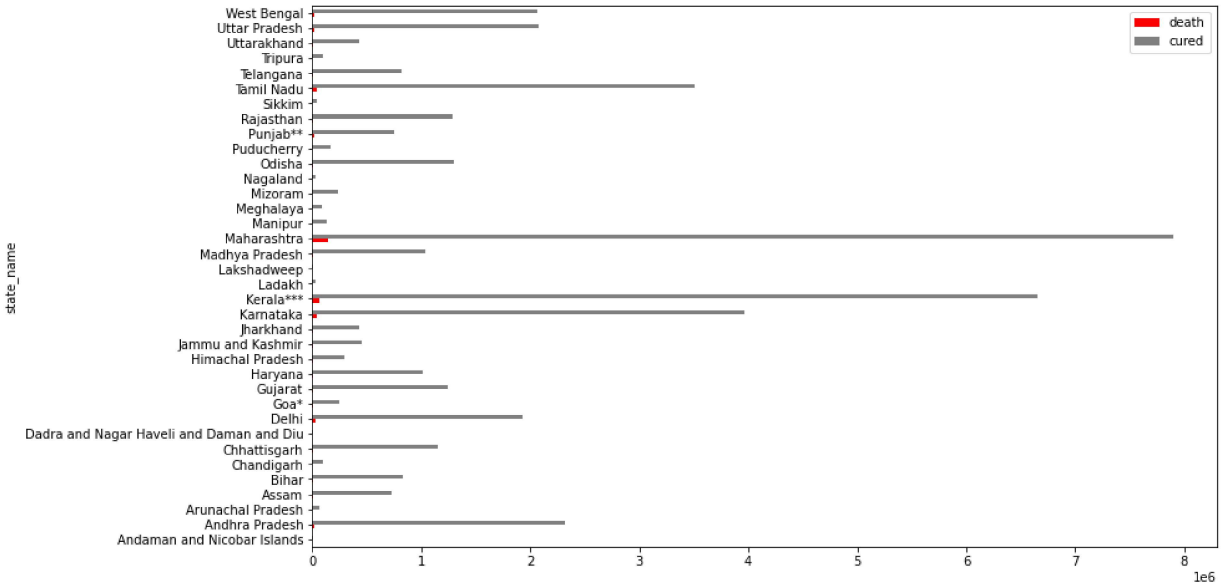
In [50]:
```
df2=df[['positive', 'death']]
df2.plot.barh(color={"positive": "aquamarine", "death": "orange"},figsize=(13,8))
# bar plot for positive cases and death rate
```

Out[50]: <AxesSubplot:ylabel='state_name'>



In [51]:
```
df3=df[['death', 'cured']]
df3.plot.barh(color={"death": "red", "cured": "grey"},figsize=(13,8))
# barplot for death cases and cured cases
```

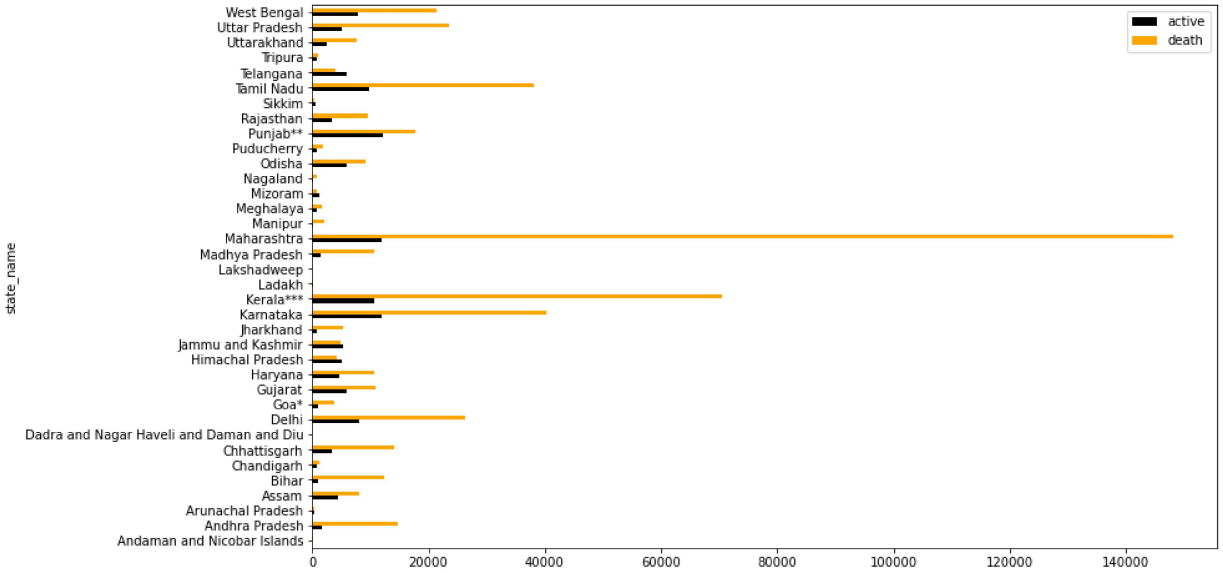
Out[51]: <AxesSubplot:ylabel='state_name'>



In [52]:

```
df4=df[['active', 'death']]
df4.plot.barh(color={"active": "black", "death": "orange"},figsize=(13,8))
# bar plot for active and death cases
```

Out[52]: <AxesSubplot:ylabel='state_name'>



In [ ]: