1.Indian Covid-19 Data Analysis

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from plotly.subplots import make_subplots
from datetime import datetime
```

In [2]:
```
##import first dataset
cov19_df=pd.read_csv("C:\\Users\\Ankit\\Desktop\\Data Science\\Covid-19 Data Analy
```

In [3]:
```
cov19_df.head(30)
```

Out[3]:

	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
5	6	2020-02-04	6:00 PM	Kerala	3	0
6	7	2020-02-05	6:00 PM	Kerala	3	0
7	8	2020-02-06	6:00 PM	Kerala	3	0
8	9	2020-02-07	6:00 PM	Kerala	3	0
9	10	2020-02-08	6:00 PM	Kerala	3	0
10	11	2020-02-09	6:00 PM	Kerala	3	0
11	12	2020-02-10	6:00 PM	Kerala	3	0
12	13	2020-02-11	6:00 PM	Kerala	3	0
13	14	2020-02-12	6:00 PM	Kerala	3	0
14	15	2020-02-13	6:00 PM	Kerala	3	0
15	16	2020-02-14	6:00 PM	Kerala	3	0
16	17	2020-02-15	6:00 PM	Kerala	3	0
17	18	2020-02-16	6:00 PM	Kerala	3	0
18	19	2020-02-17	6:00 PM	Kerala	3	0
19	20	2020-02-18	6:00 PM	Kerala	3	0
20	21	2020-02-19	6:00 PM	Kerala	3	0
21	22	2020-02-20	6:00 PM	Kerala	3	0
22	23	2020-02-21	6:00 PM	Kerala	3	0

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational
23	24	2020-02-22	6:00 PM	Kerala	3	0
24	25	2020-02-23	6:00 PM	Kerala	3	0
25	26	2020-02-24	6:00 PM	Kerala	3	0
26	27	2020-02-25	6:00 PM	Kerala	3	0
27	28	2020-02-26	6:00 PM	Kerala	3	0
28	29	2020-02-27	6:00 PM	Kerala	3	0
29	30	2020-	6:00	Kerala	3	0

In [4]:
```
cov19_df.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
```

In [5]:
```
cov19_df.describe
```

Out[5]:
```
<bound method NDFrame.describe of          Sno        Date     Time State/UnionTer
ritory  \
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]>
```

In [6]:
```python
cov19_df.drop(["Sno","Time","ConfirmedIndianNational","ConfirmedForeignNational"]
```

In [7]:
```python
cov19_df.head()
```

Out[7]:

	Date	State/UnionTerritory	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

In [8]:
```python
##Find the active cases
cov19_df['Active Cases']=cov19_df['Confirmed']-(cov19_df['Cured']  + cov19_df['Dea
cov19_df.tail()
```

Out[8]:

	Date	State/UnionTerritory	Cured	Deaths	Confirmed	Active Cases
18105	2021-08-11	Telangana	638410	3831	650353	8112
18106	2021-08-11	Tripura	77811	773	80660	2076
18107	2021-08-11	Uttarakhand	334650	7368	342462	444
18108	2021-08-11	Uttar Pradesh	1685492	22775	1708812	545
18109	2021-08-11	West Bengal	1506532	18252	1534999	10215

In [9]:
```
cov19_df.tail(15)
```

Out[9]:

	Date	State/UnionTerritory	Cured	Deaths	Confirmed	Active Cases
18095	2021-08-11	Manipur	96776	1664	105424	6984
18096	2021-08-11	Meghalaya	64157	1185	69769	4427
18097	2021-08-11	Mizoram	33722	171	46320	12427
18098	2021-08-11	Nagaland	26852	585	28811	1374
18099	2021-08-11	Odisha	972710	6565	988997	9722
18100	2021-08-11	Puducherry	119115	1800	121766	851
18101	2021-08-11	Punjab	582791	16322	599573	460
18102	2021-08-11	Rajasthan	944700	8954	953851	197
18103	2021-08-11	Sikkim	25095	356	28018	2567
18104	2021-08-11	Tamil Nadu	2524400	34367	2579130	20363
18105	2021-08-11	Telangana	638410	3831	650353	8112
18106	2021-08-11	Tripura	77811	773	80660	2076
18107	2021-08-11	Uttarakhand	334650	7368	342462	444
18108	2021-08-11	Uttar Pradesh	1685492	22775	1708812	545
18109	2021-08-11	West Bengal	1506532	18252	1534999	10215

In [10]:
```
cov19_df.tail(30)
```

Out[10]:

	Date	State/UnionTerritory	Cured	Deaths	Confirmed	Active Cases
18080	2021-08-11	Chhattisgarh	988189	13544	1003356	1623
18081	2021-08-11	Dadra and Nagar Haveli and Daman and Diu	10646	4	10654	4
18082	2021-08-11	Delhi	1411280	25068	1436852	504
18083	2021-08-11	Goa	167978	3164	172085	943
18084	2021-08-11	Gujarat	814802	10077	825085	206
18085	2021-08-11	Haryana	759790	9652	770114	672
18086	2021-08-11	Himachal Pradesh	202761	3537	208616	2318
18087	2021-08-11	Jammu and Kashmir	317081	4392	322771	1298
18088	2021-08-11	Jharkhand	342102	5130	347440	208
18089	2021-08-11	Karnataka	2861499	36848	2921049	22702
18090	2021-08-11	Kerala	3396184	18004	3586693	172505
18091	2021-08-11	Ladakh	20130	207	20411	74
18092	2021-08-11	Lakshadweep	10165	51	10263	47
18093	2021-08-11	Madhya Pradesh	781330	10514	791980	136
18094	2021-08-11	Maharashtra	6159676	134201	6363442	69565
18095	2021-08-11	Manipur	96776	1664	105424	6984
18096	2021-08-11	Meghalaya	64157	1185	69769	4427
18097	2021-08-11	Mizoram	33722	171	46320	12427
18098	2021-08-11	Nagaland	26852	585	28811	1374
18099	2021-08-11	Odisha	972710	6565	988997	9722
18100	2021-08-11	Puducherry	119115	1800	121766	851
18101	2021-08-11	Punjab	582791	16322	599573	460

	Date	State/UnionTerritory	Cured	Deaths	Confirmed	Active Cases
18102	2021-08-11	Rajasthan	944700	8954	953851	197
18103	2021-08-11	Sikkim	25095	356	28018	2567
18104	2021-08-11	Tamil Nadu	2524400	34367	2579130	20363
18105	2021-08-11	Telangana	638410	3831	650353	8112
18106	2021-08-11	Tripura	77811	773	80660	2076
18107	2021-08-11	Uttarakhand	334650	7368	342462	444
18108	2021-08-11	Uttar Pradesh	1685492	22775	1708812	545
18109	2021-08-11	West Bengal	1506532	18252	1534999	10215

In [11]:
```
## Creating a Pivot Table using Pandas libraries
## creating a variable for statewise
swise = pd.pivot_table(cov19_df , values=["Confirmed","Cured","Deaths"], index="St
```

In [12]:
```
##Recovery rates
swise["Recovery Rates"]=swise["Cured"]*100/swise["Confirmed"]
```

In [13]:
```
##Death Rates
swise["Mortality Rates"]=swise["Deaths"]*100/swise["Confirmed"]
```

In [14]:
```
#sort values based on confirmed case
swise=swise.sort_values(by="Confirmed",ascending=False)
```

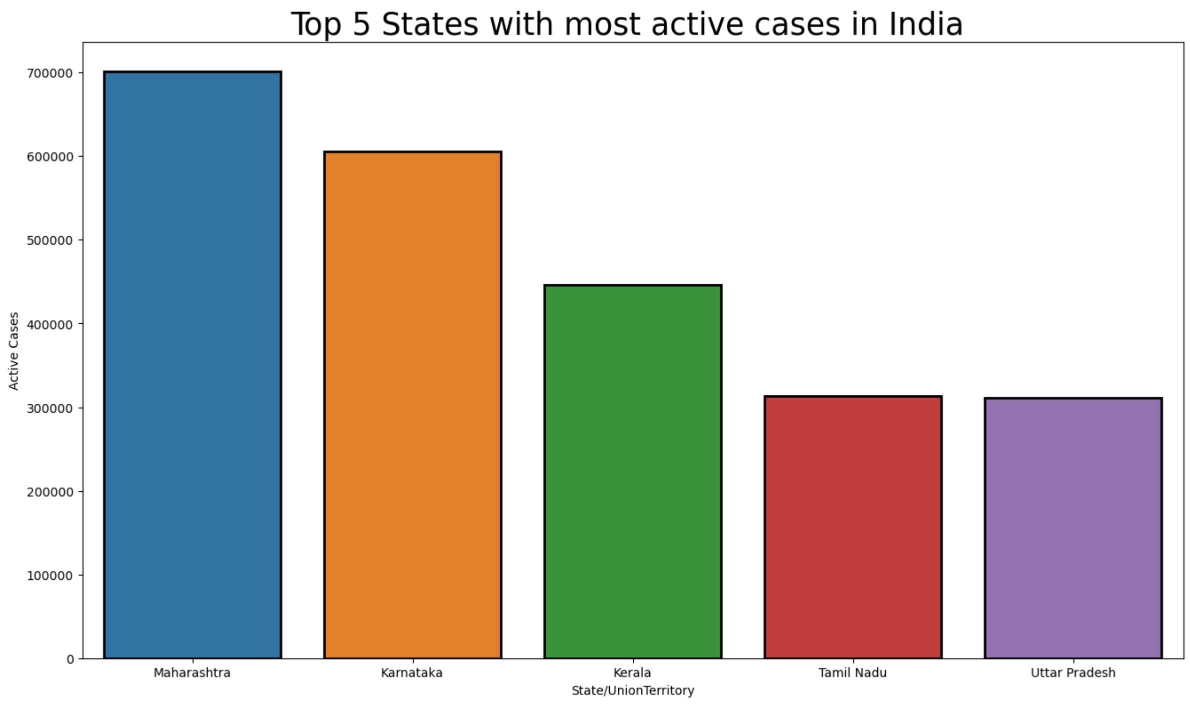
In [15]:
```
swise.style.background_gradient(cmap ="magma")
```

Out[15]:

State/UnionTerritory	Confirmed	Cured	Deaths	Recovery Rates	Mortality Rates
Maharashtra	6363442	6159676	134201	96.797865	2.108937
Maharashtra***	6229596	6000911	130753	96.329056	2.098900
Kerala	3586693	3396184	18004	94.688450	0.501967
Karnataka	2921049	2861499	36848	97.961349	1.261465
Karanataka	2885238	2821491	36197	97.790581	1.254559
Tamil Nadu	2579130	2524400	34367	97.877967	1.332504
Andhra Pradesh	1985182	1952736	13564	98.365591	0.683262
Uttar Pradesh	1708812	1685492	22775	98.635309	1.332797
West Bengal	1534999	1506532	18252	98.145471	1.189056
Delhi	1436852	1411280	25068	98.220276	1.744647
Chhattisgarh	1003356	988189	13544	98.488373	1.349870
Odisha	988997	972710	6565	98.353180	0.663804
Rajasthan	953851	944700	8954	99.040626	0.938721
Gujarat	825085	814802	10077	98.753704	1.221329
Madhya Pradesh	791980	781330	10514	98.655269	1.327559
Madhya Pradesh***	791656	780735	10506	98.620487	1.327092
Haryana	770114	759790	9652	98.659419	1.253321
Bihar	725279	715352	9646	98.631285	1.329971
Bihar****	715730	701234	9452	97.974655	1.320610
Telangana	650353	638410	3831	98.163613	0.589065
Punjab	599573	582791	16322	97.201008	2.722271
Assam	576149	559684	5420	97.142232	0.940729
Telengana	443360	362160	2312	81.685312	0.521472
Jharkhand	347440	342102	5130	98.463620	1.476514
Uttarakhand	342462	334650	7368	97.718871	2.151480
Jammu and Kashmir	322771	317081	4392	98.237140	1.360717
Himachal Pradesh	208616	202761	3537	97.193408	1.695460
Himanchal Pradesh	204516	200040	3507	97.811418	1.714780
Goa	172085	167978	3164	97.613389	1.838626
Puducherry	121766	119115	1800	97.822873	1.478245
Manipur	105424	96776	1664	91.796934	1.578388
Tripura	80660	77811	773	96.467890	0.958344
Meghalaya	69769	64157	1185	91.956313	1.698462
Chandigarh	61992	61150	811	98.641760	1.308233

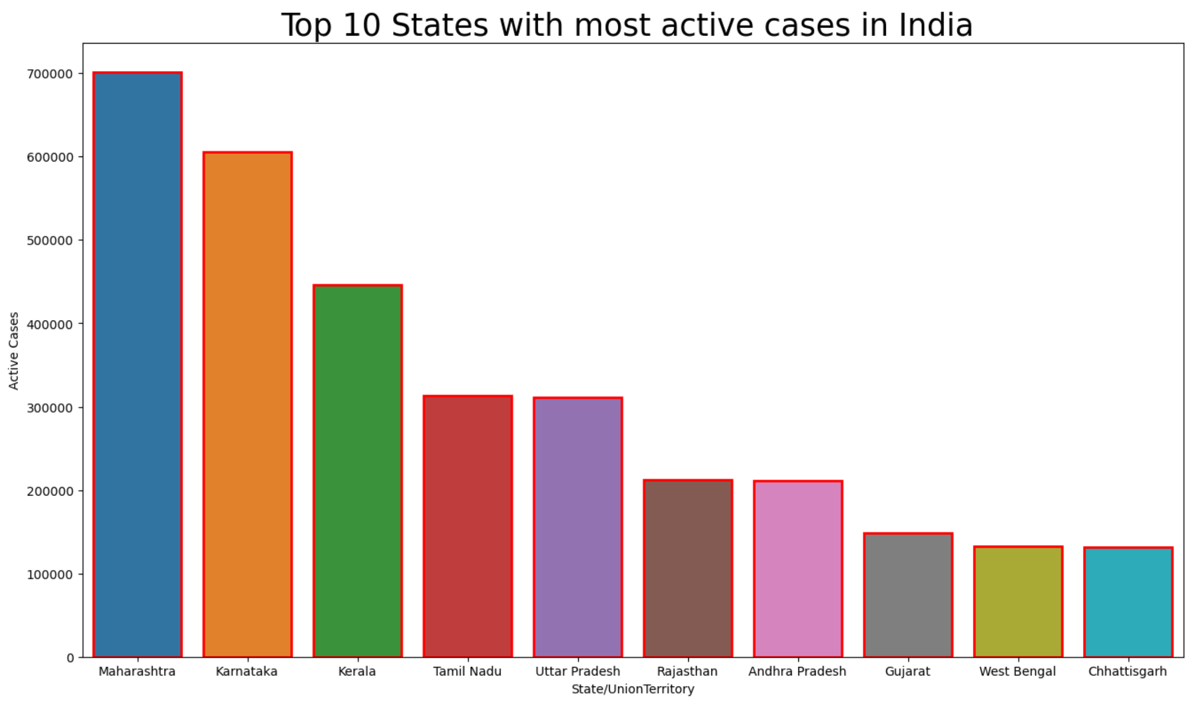
	Confirmed	Cured	Deaths	Recovery Rates	Mortality Rates
State/UnionTerritory					
Arunachal Pradesh	50605	47821	248	94.498567	0.490070
Mizoram	46320	33722	171	72.802245	0.369171
Nagaland	28811	26852	585	93.200514	2.030474
Sikkim	28018	25095	356	89.567421	1.270612
Ladakh	20411	20130	207	98.623291	1.014159
Dadra and Nagar Haveli and Daman and Diu	10654	10646	4	99.924911	0.037545
Dadra and Nagar Haveli	10377	10261	4	98.882143	0.038547
Lakshadweep	10263	10165	51	99.045114	0.496931
Cases being reassigned to states	9265	0	0	0.000000	0.000000
Andaman and Nicobar Islands	7548	7412	129	98.198198	1.709062
Unassigned	77	0	0	0.000000	0.000000
Daman & Diu	2	0	0	0.000000	0.000000

In [17]:
```
## Data Visualization of Covid Case
##Box Plot
# Creating a bar plots based on top 5 states
##top 5 active states
top_5_active_cases = cov19_df.groupby(by='State/UnionTerritory').max()[['Active Ca
fig=plt.figure(figsize=(16,9))
plt.title("Top 5 States with most active cases in India ",size=25)
ax=sns.barplot(data=top_5_active_cases.iloc[:5] , y="Active Cases",x="State/UnionTe
```

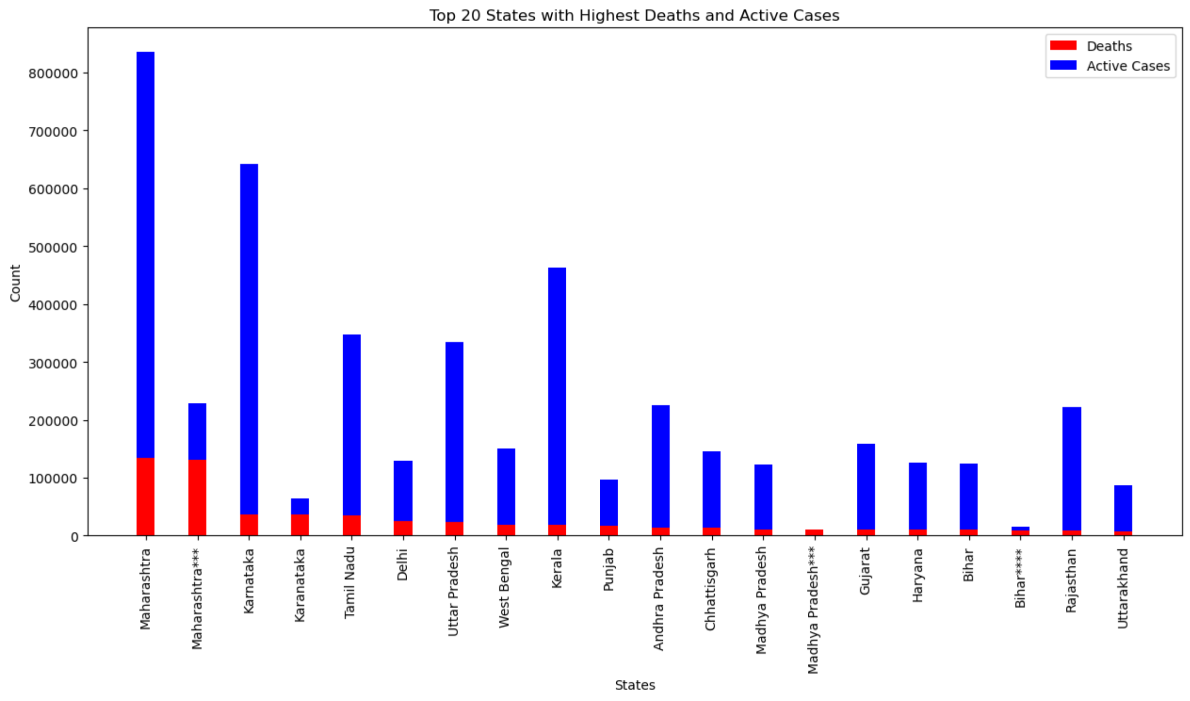


In [18]:
```
# Creating a bar plots based on top 5 states
##top 10 active states
top_10_active_cases = cov19_df.groupby(by='State/UnionTerritory').max()[['Active Ca
fig=plt.figure(figsize=(16,9))
```

```
plt.title("Top 10 States with most active cases in India ",size=25)
ax=sns.barplot(data=top_10_active_cases.iloc[:10] , y="Active Cases",x="State/Unio
```

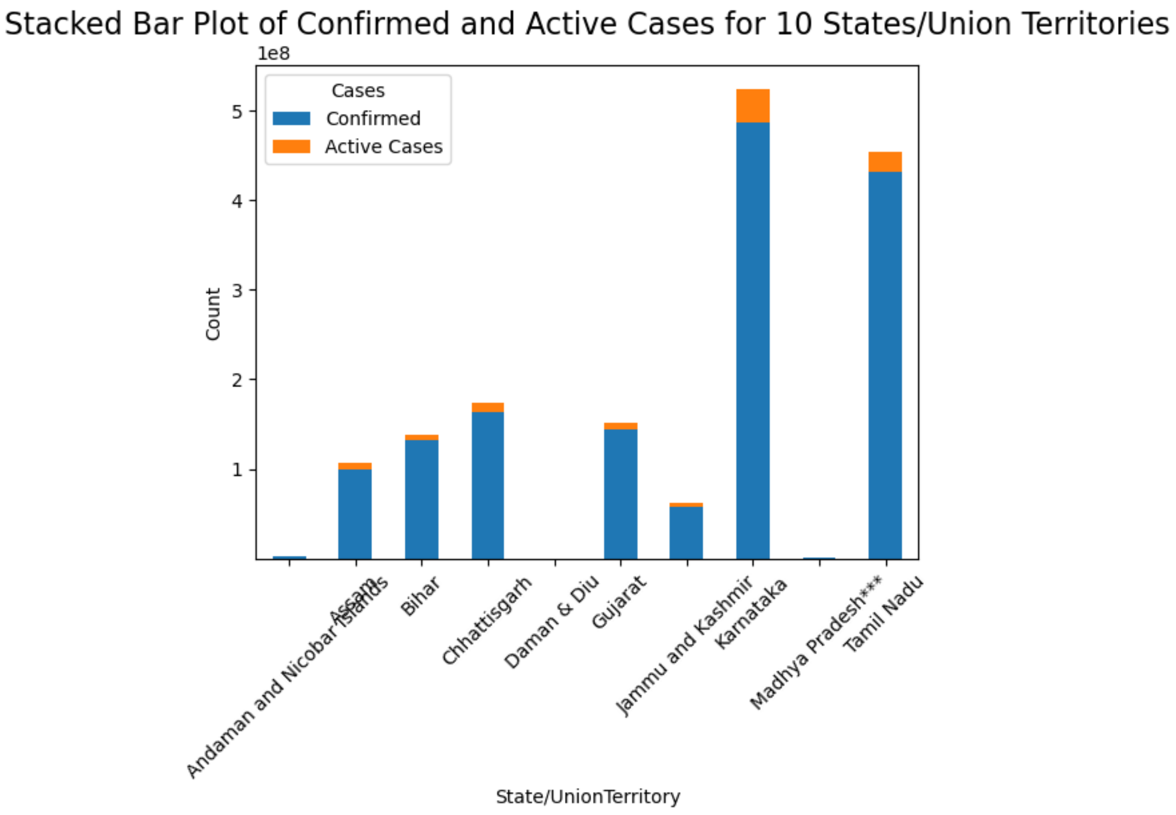


```
In [19]: # Group bar Chart

top_20_death_cases = cov19_df.groupby(by='State/UnionTerritory').max()[['Deaths',
# Create a grouped bar chart
fig, ax = plt.subplots(figsize=(15, 7))
width = 0.35
x = range(len(top_20_death_cases))
ax.bar(x, top_20_death_cases['Deaths'], width, label='Deaths', color='red')
ax.bar(x, top_20_death_cases['Active Cases'], width, label='Active Cases', bottom=
ax.set_xlabel('States')
ax.set_ylabel('Count')
ax.set_title('Top 20 States with Highest Deaths and Active Cases')
ax.set_xticks(x)
ax.set_xticklabels(top_20_death_cases.index, rotation=90)
ax.legend()
plt.show()
```

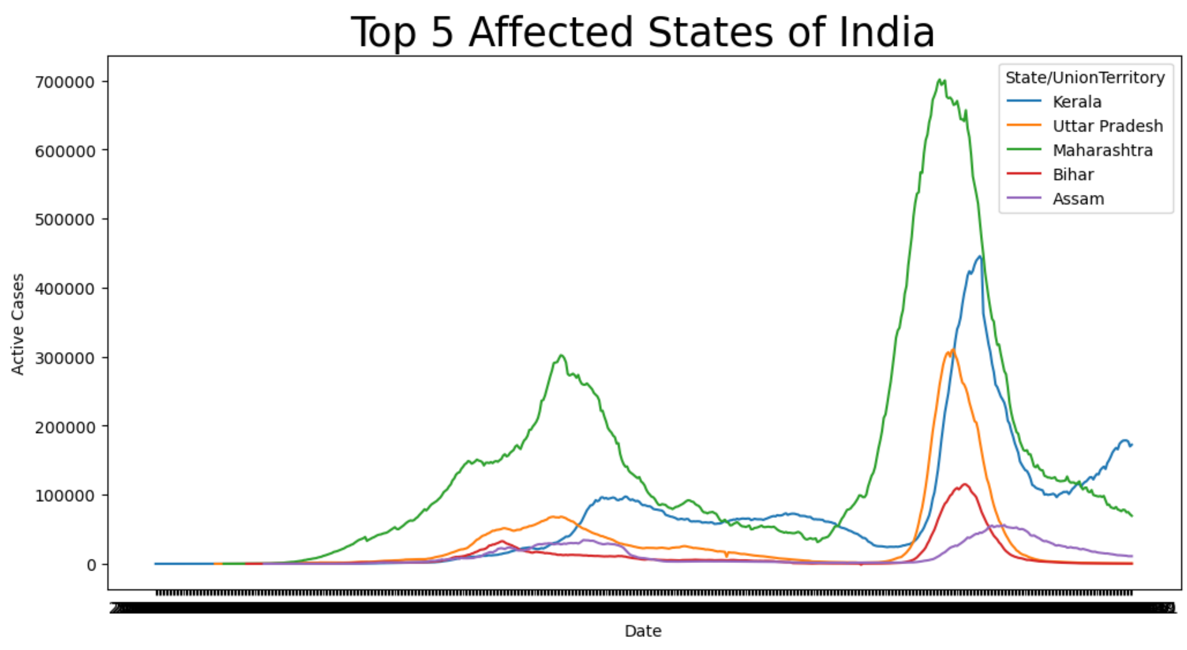


In [20]:
```
##stacked bar plot
import random
random_states = random.sample(cov19_df['State/UnionTerritory'].unique().tolist(), 1
selected_states_data = cov19_df[cov19_df['State/UnionTerritory'].isin(random_state
statewise_totals = selected_states_data.groupby('State/UnionTerritory').agg({'Conf
plt.figure(figsize=(12, 6))
ax = statewise_totals.plot(kind='bar', stacked=True)
plt.title('Stacked Bar Plot of Confirmed and Active Cases for 10 States/Union Terr
plt.xlabel('State/UnionTerritory')
plt.ylabel('Count')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.legend(title='Cases', labels=['Confirmed', 'Active Cases'], loc='upper left')
plt.show()
```

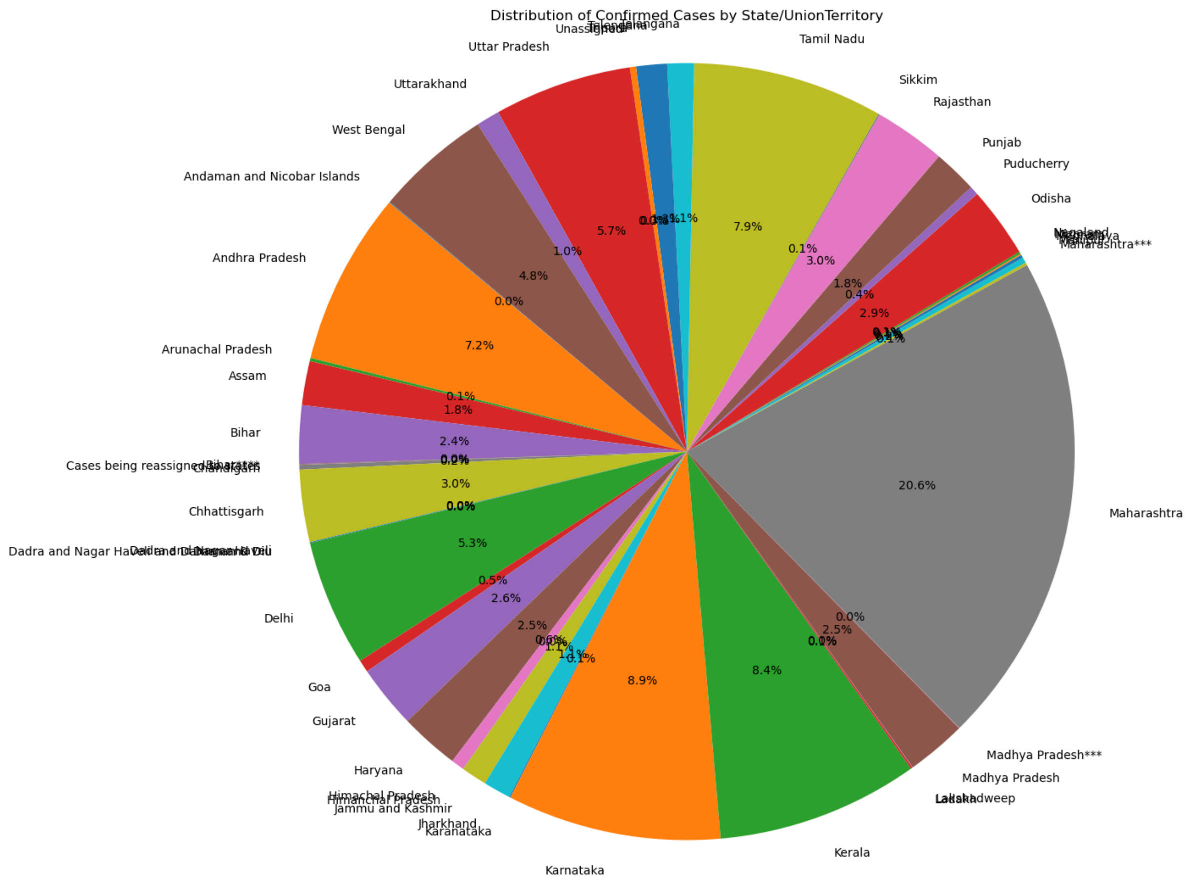
```
<Figure size 1200x600 with 0 Axes>
```



In [21]:
```
#Line Plot
## Covid case Growth trend
fig = plt.figure(figsize=(12, 6))
ax = sns.lineplot(data=cov19_df[cov19_df['State/UnionTerritory'].isin(['Maharashtra
ax.set_title("Top 5 Affected States of India", size=25)

plt.show()  # Use plt.show() to display the plot
```

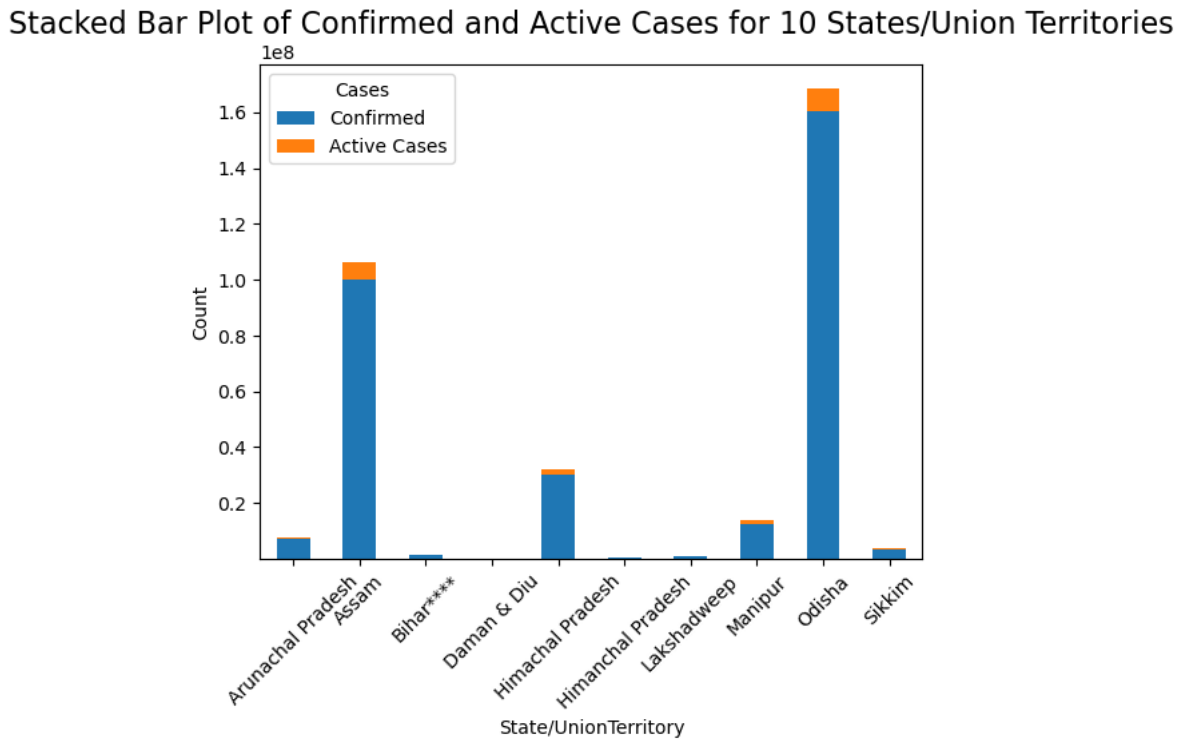


In [22]:
```
##Pie Chart
# Group the data by State/UnionTerritory and calculate the total confirmed and deat
statewise_totals = cov19_df.groupby('State/UnionTerritory').agg({'Confirmed': 'sum
# Create a pie chart
plt.figure(figsize=(13, 13))
plt.pie(statewise_totals['Confirmed'], labels=statewise_totals.index, autopct='%1.
plt.title('Distribution of Confirmed Cases by State/UnionTerritory')
plt.axis('equal')
plt.show()
```

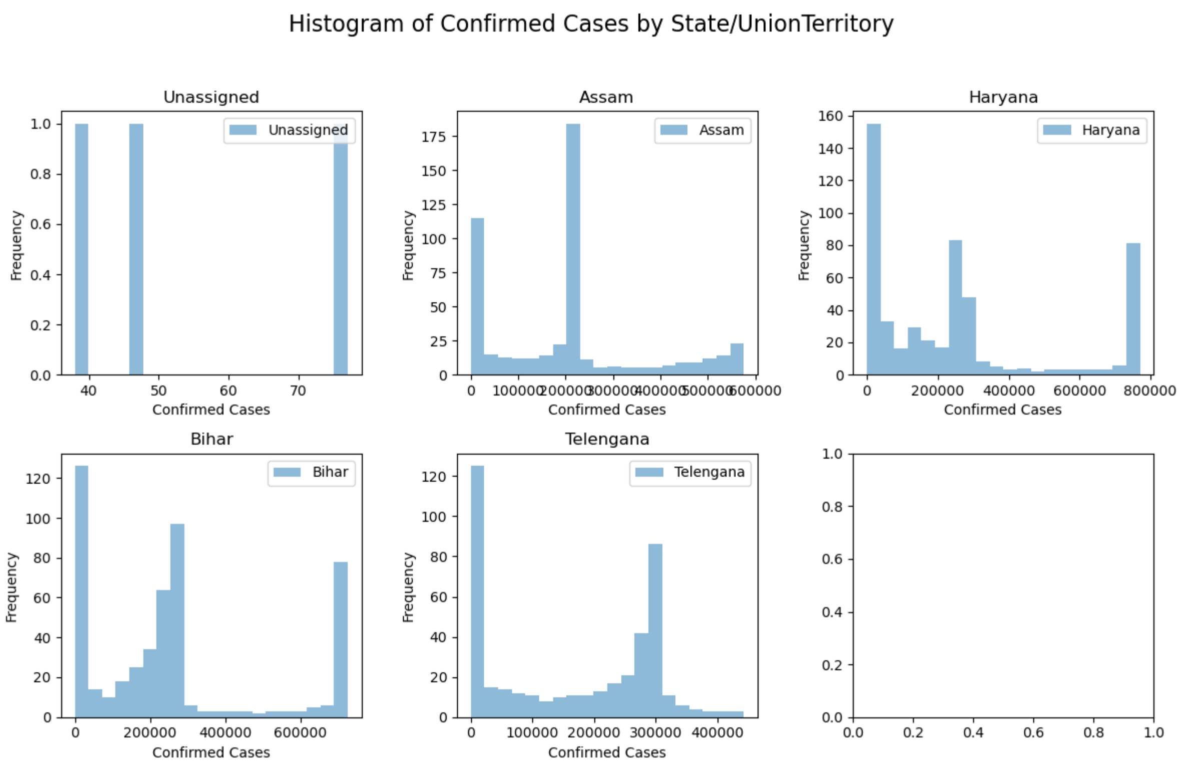


In [23]:
```
##Horizontal Bar Plot using Barh
##stacked bar plot
import random
random_states = random.sample(cov19_df['State/UnionTerritory'].unique().tolist(), 
selected_states_data = cov19_df[cov19_df['State/UnionTerritory'].isin(random_state
statewise_totals = selected_states_data.groupby('State/UnionTerritory').agg({'Conf
plt.figure(figsize=(12, 6))
ax = statewise_totals.plot(kind='bar', stacked=True)
plt.title('Stacked Bar Plot of Confirmed and Active Cases for 10 States/Union Terr
plt.xlabel('State/UnionTerritory')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Cases', labels=['Confirmed', 'Active Cases'], loc='upper left')
plt.show()
```

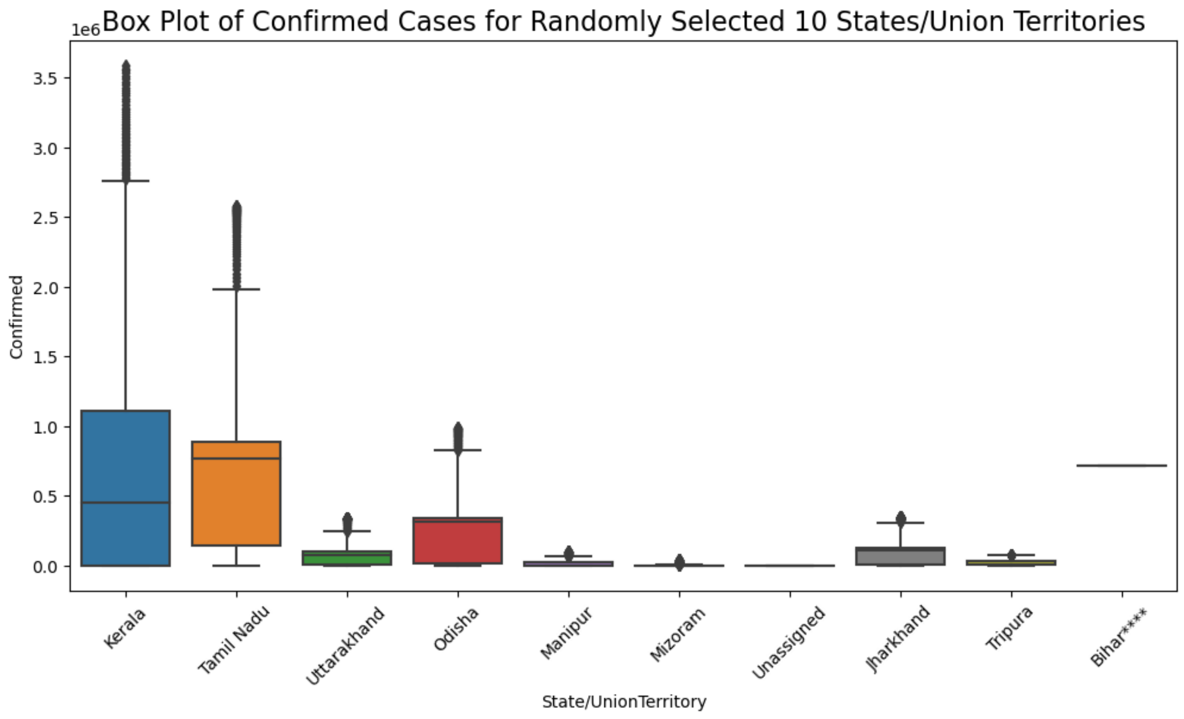
```
<Figure size 1200x600 with 0 Axes>
```



In [24]:
```
#histogram
random_states = random.sample(cov19_df['State/UnionTerritory'].unique().tolist(), 
# Create a figure with subplots for each state
fig, axes = plt.subplots(2, 3, figsize=(12, 8))
fig.suptitle('Histogram of Confirmed Cases by State/UnionTerritory', fontsize=16)
# Loop through the selected states and create histograms
for i, state in enumerate(random_states):
    ax = axes[i // 3, i % 3]
    state_data = cov19_df[cov19_df['State/UnionTerritory'] == state]['Confirmed']
    ax.hist(state_data, bins=20, alpha=0.5, label=state)
    ax.set_title(state)
    ax.set_xlabel('Confirmed Cases')
    ax.set_ylabel('Frequency')
    ax.legend(loc='upper right')

# Adjust layout
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

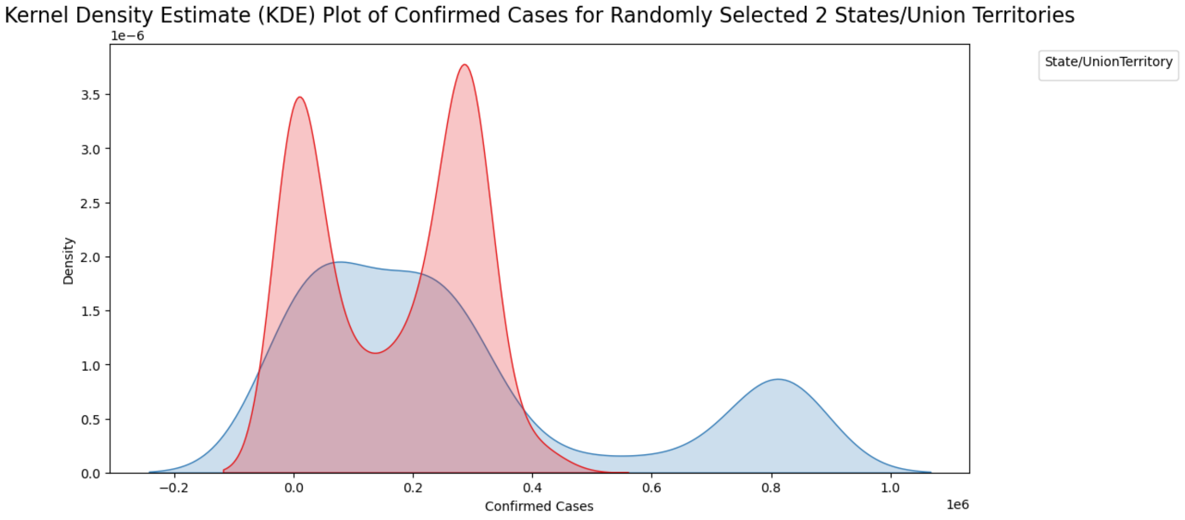


In [25]:
```python
#Box Plot
# Randomly select 10 'State/UnionTerritory' values
random_states = random.sample(cov19_df['State/UnionTerritory'].unique().tolist(), 

# Filter the DataFrame to include only the selected states
selected_states_data = cov19_df[cov19_df['State/UnionTerritory'].isin(random_state

# Create a box plot
plt.figure(figsize=(12, 6))
ax = sns.boxplot(data=selected_states_data, x='State/UnionTerritory', y='Confirmed
ax.set_title('Box Plot of Confirmed Cases for Randomly Selected 10 States/Union Te

plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



```
In [26]: ## Kernel Density Estimate plot using Gaussian kernels.
         import pandas as pd
         import random
         import seaborn as sns
         import matplotlib.pyplot as plt

         cov19_df=pd.read_csv("C:\\Users\\Ankit\\Desktop\\Data Science\\Covid-19 Data Analy
         random_states = random.sample(cov19_df['State/UnionTerritory'].unique().tolist(), 

         # Filter the DataFrame to include only the selected states
         selected_states_data = cov19_df[cov19_df['State/UnionTerritory'].isin(random_state

         # Create a KDE plot
         plt.figure(figsize=(12, 6))
         ax = sns.kdeplot(data=selected_states_data, x='Confirmed', hue='State/UnionTerrito
         ax.set_title('Kernel Density Estimate (KDE) Plot of Confirmed Cases for Randomly Se

         plt.xlabel('Confirmed Cases')
         plt.ylabel('Density')
         plt.legend(title='State/UnionTerritory', loc='upper right', bbox_to_anchor=(1.25, 

         plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label star
t with an underscore are ignored when legend() is called with no argument.
```


Kernel Density Estimate (KDE) Plot of Confirmed Cases for Randomly Selected 2 States/Union Territories
1e−6
Density
1e6
Confirmed Cases
State/UnionTerritory