In [1]:
```
import pandas as pd #data manipulation
import numpy as np  #numerical
import matplotlib.pyplot as plt #data visualization
import seaborn as sns #data visualization
import plotly.express as px #data visualization
from plotly.subplots import make_subplots #data visualization,plotting
from datetime import datetime
```

In [2]:
```
covid_df = pd.read_csv('C:/Users/admin/Desktop/covid_19_india.csv')
```

In [3]:
```
covid_df.head(10)
```

Out[3]:

| | Sno | Date | Time | State/UnionTerritory | ConfirmedIndianNational | ConfirmedForeignNational | Cured | De |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 19-05-21 | 6:00 PM | Kerala | 1 | 0 | 0 | |
| **1** | 2 | 19-05-21 | 6:00 PM | Kerala | 1 | 0 | 0 | |
| **2** | 3 | 19-05-21 | 6:00 PM | Kerala | 2 | 0 | 0 | |
| **3** | 4 | 19-05-21 | 6:00 PM | Kerala | 3 | 0 | 0 | |
| **4** | 5 | 19-05-21 | 6:00 PM | Kerala | 3 | 0 | 0 | |
| **5** | 6 | 19-05-21 | 6:00 PM | Kerala | 3 | 0 | 0 | |
| **6** | 7 | 19-05-21 | 6:00 PM | Kerala | 3 | 0 | 0 | |
| **7** | 8 | 19-05-21 | 6:00 PM | Kerala | 3 | 0 | 0 | |
| **8** | 9 | 19-05-21 | 6:00 PM | Kerala | 3 | 0 | 0 | |
| **9** | 10 | 19-05-21 | 6:00 PM | Kerala | 3 | 0 | 0 | |

In [4]:
```
covid_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15086 entries, 0 to 15085
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Sno                      15086 non-null  int64 
 1   Date                     15086 non-null  object
 2   Time                     15086 non-null  object
 3   State/UnionTerritory     15086 non-null  object
 4   ConfirmedIndianNational  15086 non-null  object
 5   ConfirmedForeignNational 15086 non-null  object
 6   Cured                    15086 non-null  int64 
 7   Deaths                   15086 non-null  int64 
 8   Confirmed                15086 non-null  int64 
dtypes: int64(4), object(5)
memory usage: 1.0+ MB
```

In [5]:
```
covid_df.describe()
```

Out[5]:

| | Sno | Cured | Deaths | Confirmed |
|---|---|---|---|---|
| **count** | 15086.000000 | 1.508600e+04 | 15086.000000 | 1.508600e+04 |
| **mean** | 7543.500000 | 1.747937e+05 | 2721.084449 | 1.942820e+05 |
| **std** | 4355.097416 | 3.648330e+05 | 7182.672358 | 4.095184e+05 |
| **min** | 1.000000 | 0.000000e+00 | 0.000000 | 0.000000e+00 |
| **25%** | 3772.250000 | 1.685000e+03 | 12.000000 | 2.935500e+03 |
| **50%** | 7543.500000 | 1.964700e+04 | 364.000000 | 2.608150e+04 |
| **75%** | 11314.750000 | 2.087552e+05 | 2170.000000 | 2.216012e+05 |
| **max** | 15086.000000 | 4.927480e+06 | 83777.000000 | 5.433506e+06 |

In [6]:
```
vaccine_df=pd.read_csv('C:/Users/admin/Downloads/covid_vaccine_statewise.csv')
```

In [7]:
```
vaccine_df.head()
```

Out[7]:

| | Updated On | State | Total Doses Administered | Sessions | Sites | First Dose Administered | Second Dose Administered | Male (Doses Administered) | Fe (D Administ |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 16-01-21 | India | 48276.0 | 3455.0 | 2957.0 | 48276.0 | 0.0 | NaN | |
| **1** | 17-01-21 | India | 58604.0 | 8532.0 | 4954.0 | 58604.0 | 0.0 | NaN | |
| **2** | 18-01-21 | India | 99449.0 | 13611.0 | 6583.0 | 99449.0 | 0.0 | NaN | |
| **3** | 19-01-21 | India | 195525.0 | 17855.0 | 7951.0 | 195525.0 | 0.0 | NaN | |
| **4** | 20-01-21 | India | 251280.0 | 25472.0 | 10504.0 | 251280.0 | 0.0 | NaN | |

5 rows × 24 columns

# Exploration summary

- Drop unnecessary columns from 1st dataset like time, sno, ConfirmedIndianNational, ConfirmedForeignNational

In [8]:
```
covid_df.drop(["Sno","Time","ConfirmedIndianNational","ConfirmedForeignNational",],inplace=True,
covid_df.head()
```

Out[8]:

| | Date | State/UnionTerritory | Cured | Deaths | Confirmed |
|---|---|---|---|---|---|
| **0** | 19-05-21 | Kerala | 0 | 0 | 1 |
| **1** | 19-05-21 | Kerala | 0 | 0 | 1 |
| **2** | 19-05-21 | Kerala | 0 | 0 | 2 |
| **3** | 19-05-21 | Kerala | 0 | 0 | 3 |
| **4** | 19-05-21 | Kerala | 0 | 0 | 3 |

# Format the date column

In [9]:
```
covid_df['Date'] = pd.to_datetime(covid_df['Date'], format='%y-%m-%d')
covid_df.head()
```

Out[9]:

| | Date | State/UnionTerritory | Cured | Deaths | Confirmed |
|---|---|---|---|---|---|
| **0** | 2019-05-21 | Kerala | 0 | 0 | 1 |
| **1** | 2019-05-21 | Kerala | 0 | 0 | 1 |
| **2** | 2019-05-21 | Kerala | 0 | 0 | 2 |
| **3** | 2019-05-21 | Kerala | 0 | 0 | 3 |
| **4** | 2019-05-21 | Kerala | 0 | 0 | 3 |

# Find active cases

- Active cases = Total no. of confirmed cases - cured cases + death reported

In [10]:
```
covid_df['Active_Cases'] = covid_df['Confirmed'] - (covid_df['Cured'] + covid_df['Deaths'])
covid_df.tail()
```

Out[10]:

| | Date | State/UnionTerritory | Cured | Deaths | Confirmed | Active_Cases |
|---|---|---|---|---|---|---|
| **15081** | 2003-02-20 | Telangana | 485644 | 3012 | 536766 | 48110 |
| **15082** | 2002-02-20 | Tripura | 36402 | 450 | 42776 | 5924 |
| **15083** | 2001-02-20 | Uttarakhand | 214426 | 5132 | 295790 | 76232 |
| **15084** | 2031-01-20 | Uttar Pradesh | 1483249 | 18072 | 1637663 | 136342 |
| **15085** | 2030-01-20 | West Bengal | 1026492 | 13576 | 1171861 | 131793 |

# Create Pivot tables using pandas library

- Here in this table will be summing all the confirmed, deaths and cured cases for each of the states and union territories

In [11]:
```
statewise = pd.pivot_table(covid_df, values = ["Confirmed", "Deaths", "Cured"],
                           index = "State/UnionTerritory", aggfunc = "max")
```

## Find the recovery rate = total no. of cured cases / total no. of confirmed cases * 100

In [12]:
```
statewise["Recovery_rate"] = statewise["Cured"]*100/statewise["Confirmed"]
```

## Find moratlity rate = Total no. of deaths/ total no. of confirmed cases * 100

In [13]:
```
statewise["Mortality_rate"] = statewise["Deaths"]*100/statewise["Confirmed"]
```

In [14]:
```
# sort the values based on confirmed cases value
```

In [15]:
```
statewise = statewise.sort_values(by= "Confirmed", ascending = False)
```

- Plot our pivot table using visual for that use background_gradient function and inside that function will pass c_map parameter
- c_map stands for color map, present inside matplotlib provided by matplotlib.org

In [16]:
```
statewise.style.background_gradient(cmap = "cubehelix")
```

Out[16]:

| State/UnionTerritory | Confirmed | Cured | Deaths | Recovery_rate | Mortality_rate |
|---|---|---|---|---|---|
| Maharashtra | 5433506 | 4927480 | 83777 | 90.686934 | 1.541859 |
| Karnataka | 2272374 | 1674487 | 22838 | 73.688882 | 1.005028 |
| Kerala | 2200706 | 1846105 | 6612 | 83.886944 | 0.300449 |
| Tamil Nadu | 1664350 | 1403052 | 18369 | 84.300297 | 1.103674 |
| Uttar Pradesh | 1637663 | 1483249 | 18072 | 90.571076 | 1.103524 |
| Andhra Pradesh | 1475372 | 1254291 | 9580 | 85.015237 | 0.649328 |
| Delhi | 1402873 | 1329899 | 22111 | 94.798246 | 1.576123 |
| West Bengal | 1171861 | 1026492 | 13576 | 87.595030 | 1.158499 |
| Chhattisgarh | 925531 | 823113 | 12036 | 88.934136 | 1.300443 |
| Rajasthan | 879664 | 713129 | 7080 | 81.068340 | 0.804853 |
| Gujarat | 766201 | 660489 | 9269 | 86.203098 | 1.209735 |
| Madhya Pradesh | 742718 | 652612 | 7139 | 87.868074 | 0.961199 |
| Haryana | 709689 | 626852 | 6923 | 88.327704 | 0.975498 |
| Bihar | 664115 | 595377 | 4039 | 89.649684 | 0.608178 |
| Odisha | 633302 | 536595 | 2357 | 84.729718 | 0.372176 |
| Telangana | 536766 | 485644 | 3012 | 90.475924 | 0.561138 |
| Punjab | 511652 | 427058 | 12317 | 83.466497 | 2.407300 |
| Telengana | 443360 | 362160 | 2312 | 81.685312 | 0.521472 |
| Assam | 340858 | 290774 | 2344 | 85.306491 | 0.687676 |
| Jharkhand | 320934 | 284805 | 4601 | 88.742545 | 1.433628 |
| Uttarakhand | 295790 | 214426 | 5132 | 72.492647 | 1.735015 |
| Jammu and Kashmir | 251919 | 197701 | 3293 | 78.478003 | 1.307166 |
| Himachal Pradesh | 166678 | 129330 | 2460 | 77.592724 | 1.475900 |
| Goa | 138776 | 112633 | 2197 | 81.161728 | 1.583127 |
| Puducherry | 87749 | 69060 | 1212 | 78.701752 | 1.381212 |
| Chandigarh | 56513 | 48831 | 647 | 86.406667 | 1.144869 |
| Tripura | 42776 | 36402 | 450 | 85.099121 | 1.051992 |
| Manipur | 40683 | 33466 | 612 | 82.260404 | 1.504314 |
| Meghalaya | 24872 | 19185 | 355 | 77.134931 | 1.427308 |
| Arunachal Pradesh | 22462 | 19977 | 88 | 88.936871 | 0.391773 |
| Nagaland | 18714 | 14079 | 228 | 75.232446 | 1.218339 |
| Ladakh | 16784 | 15031 | 170 | 89.555529 | 1.012869 |
| Sikkim | 11689 | 8427 | 212 | 72.093421 | 1.813671 |
| Dadra and Nagar Haveli and Daman and Diu | 9652 | 8944 | 4 | 92.664733 | 0.041442 |

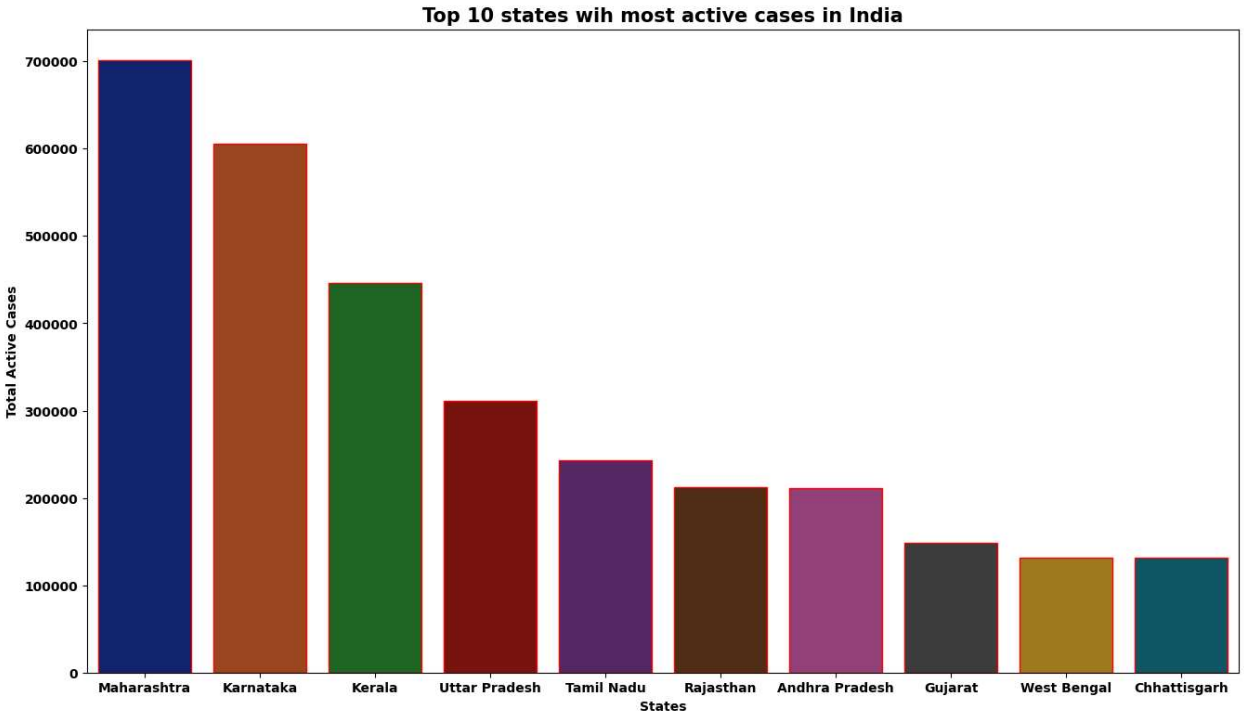
| State/UnionTerritory | Confirmed | Cured | Deaths | Recovery_rate | Mortality_rate |
|---|---|---|---|---|---|
| Cases being reassigned to states | 9265 | 0 | 0 | 0.000000 | 0.000000 |
| Mizoram | 9252 | 7094 | 29 | 76.675313 | 0.313446 |
| Andaman and Nicobar Islands | 6674 | 6359 | 92 | 95.280192 | 1.378484 |
| Lakshadweep | 5212 | 3915 | 15 | 75.115119 | 0.287797 |
| Unassigned | 77 | 0 | 0 | 0.000000 | 0.000000 |
| Daman & Diu | 2 | 0 | 0 | 0.000000 | 0.000000 |

In [17]:
```
##top 10 states based on active status or top 10 active cases across states0

top_10_active_cases = covid_df.loc[covid_df.groupby('State/UnionTerritory')['Active_Cases'].idxma
```

In [18]:
```
fig = plt.figure(figsize=(16,9)  )
```

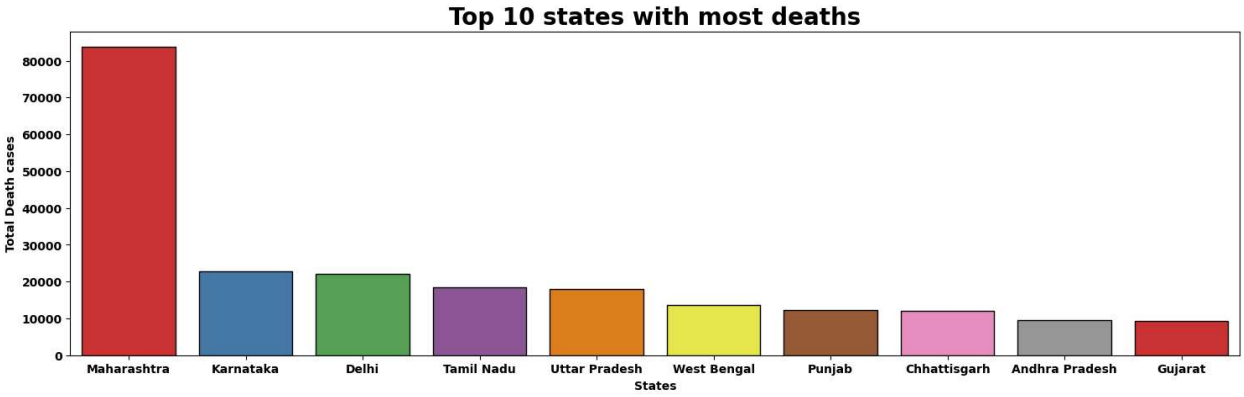
```
<Figure size 1600x900 with 0 Axes>
```

In [19]:
```
##top 10 states based on active status or top 10 active cases across states0

top_10_active_cases = covid_df.loc[covid_df.groupby('State/UnionTerritory')['Active_Cases'].idxma
fig = plt.figure(figsize=(16,9)  )
plt.title("Top 10 states wih most active cases in India",size=15,fontweight='bold')  ##title to t
sx = sns.barplot(data = top_10_active_cases.iloc[:10],
                 y= "Active_Cases",
                 x = "State/UnionTerritory",
                 linewidth = 1,
                edgecolor = 'red',
                 hue="State/UnionTerritory",
                 palette = "dark",  ##can use Purples,Reds,Blues, Greens, Oranges,deep,muted,past
                 dodge = False) ##iloc for index location

## As labels are overlapping and to change the bar color
plt.xlabel("States",fontweight='bold')
plt.ylabel("Total Active Cases",fontweight='bold')
plt.xticks(fontweight='bold') ## to make the name of states as bold
plt.yticks(fontweight='bold')
plt.show()
##lets collite all the lines of of that we have witten for most active cases in one cell
```




# Top 10 cases with highest deaths

In [20]:
```
### Strip whitespaces and tabs from column names
covid_df.columns = covid_df.columns.str.strip()

# Plot top 10 deaths
top_10_deaths = covid_df.groupby(by='State/UnionTerritory').max()[['Deaths', 'Date']].sort_values

# Create the plot
fig = plt.figure(figsize=(18, 5))
sns.barplot(data=top_10_deaths.iloc[:10],
            y="Deaths",
            x="State/UnionTerritory",
            linewidth=1,
            edgecolor="black",
           hue="State/UnionTerritory",
           palette="Set1") ##"Blues", "BuGn", "Oranges", "RdBu", "coolwarm" ,"Set1", "Set2", "Set.

# Add titles and labels
plt.title('Top 10 states with most deaths', size=20, fontweight='bold')
plt.xlabel("States",fontweight='bold')
plt.xticks(fontweight='bold')
plt.yticks(fontweight='bold')
plt.ylabel("Total Death cases",fontweight='bold')
plt.show()
```




# Using second dataset vaccine_df

In [21]:
```
vaccine_df.head()
```

Out[21]:

| | Updated On | State | Total Doses Administered | Sessions | Sites | First Dose Administered | Second Dose Administered | Male (Doses Administered) | Fe (D Administ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 16-01-21 | India | 48276.0 | 3455.0 | 2957.0 | 48276.0 | 0.0 | NaN | |
| 1 | 17-01-21 | India | 58604.0 | 8532.0 | 4954.0 | 58604.0 | 0.0 | NaN | |
| 2 | 18-01-21 | India | 99449.0 | 13611.0 | 6583.0 | 99449.0 | 0.0 | NaN | |
| 3 | 19-01-21 | India | 195525.0 | 17855.0 | 7951.0 | 195525.0 | 0.0 | NaN | |
| 4 | 20-01-21 | India | 251280.0 | 25472.0 | 10504.0 | 251280.0 | 0.0 | NaN | |

5 rows × 24 columns

# Rename the Updated on column to vaccine_date

In [22]:
```
## use rename() in pandas library

vaccine_df.rename(columns = {'Updated On' : 'Vaccine_date'}, inplace = True)
```

In [23]:
```
vaccine_df.head()
```

Out[23]:

| | Vaccine_date | State | Total Doses Administered | Sessions | Sites | First Dose Administered | Second Dose Administered | Male (Doses Administered) | Adm |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 16-01-21 | India | 48276.0 | 3455.0 | 2957.0 | 48276.0 | 0.0 | NaN | |
| 1 | 17-01-21 | India | 58604.0 | 8532.0 | 4954.0 | 58604.0 | 0.0 | NaN | |
| 2 | 18-01-21 | India | 99449.0 | 13611.0 | 6583.0 | 99449.0 | 0.0 | NaN | |
| 3 | 19-01-21 | India | 195525.0 | 17855.0 | 7951.0 | 195525.0 | 0.0 | NaN | |
| 4 | 20-01-21 | India | 251280.0 | 25472.0 | 10504.0 | 251280.0 | 0.0 | NaN | |

5 rows × 24 columns

In [24]:
```
vaccine_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7845 entries, 0 to 7844
Data columns (total 24 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   Vaccine_date                          7845 non-null   object 
 1   State                                 7845 non-null   object 
 2   Total Doses Administered              7621 non-null   float64
 3   Sessions                              7621 non-null   float64
 4    Sites                                7621 non-null   float64
 5   First Dose Administered               7621 non-null   float64
 6   Second Dose Administered              7621 non-null   float64
 7   Male (Doses Administered)             7461 non-null   float64
 8   Female (Doses Administered)           7461 non-null   float64
 9   Transgender (Doses Administered)      7461 non-null   float64
 10   Covaxin (Doses Administered)         7621 non-null   float64
 11  CoviShield (Doses Administered)       7621 non-null   float64
 12  Sputnik V (Doses Administered)        2995 non-null   float64
 13  AEFI                                  5438 non-null   float64
 14  18-44 Years (Doses Administered)      1702 non-null   float64
 15  45-60 Years (Doses Administered)      1702 non-null   float64
 16  60+ Years (Doses Administered)        1702 non-null   float64
 17  18-44 Years(Individuals Vaccinated)   3733 non-null   float64
 18  45-60 Years(Individuals Vaccinated)   3734 non-null   float64
 19  60+ Years(Individuals Vaccinated)     3734 non-null   float64
 20  Male(Individuals Vaccinated)          160 non-null    float64
 21  Female(Individuals Vaccinated)        160 non-null    float64
 22  Transgender(Individuals Vaccinated)   160 non-null    float64
 23  Total Individuals Vaccinated          5919 non-null   float64
dtypes: float64(22), object(2)
memory usage: 1.4+ MB
```

## Find sum of missing values for each column

In [25]:
```
vaccine_df.isnull().sum()
```

Out[25]:
```
Vaccine_date                          0
State                                 0
Total Doses Administered            224
Sessions                            224
 Sites                              224
First Dose Administered             224
Second Dose Administered            224
Male (Doses Administered)           384
Female (Doses Administered)         384
Transgender (Doses Administered)    384
 Covaxin (Doses Administered)       224
CoviShield (Doses Administered)     224
Sputnik V (Doses Administered)     4850
AEFI                               2407
18-44 Years (Doses Administered)   6143
45-60 Years (Doses Administered)   6143
60+ Years (Doses Administered)     6143
18-44 Years(Individuals Vaccinated) 4112
45-60 Years(Individuals Vaccinated) 4111
60+ Years(Individuals Vaccinated)  4111
Male(Individuals Vaccinated)       7685
Female(Individuals Vaccinated)     7685
Transgender(Individuals Vaccinated) 7685
Total Individuals Vaccinated       1926
dtype: int64
```

# Drop few missing value columns

- Almost all the columns have missing values..but columns such as (Male Individuals Vaccinated ) ansd others have a lot of missing values

In [26]:
```
## Immutability in Pandas: By default, the drop() method does not modify the original DataFrame (
## Instead, it returns a new DataFrame with the specified columns removed. If you assign this new
## you're essentially creating a new copy of the data, without the dropped columns.

## When you call vaccination.head(), it reflects the changes because it is based on the modified
vaccination = vaccine_df.drop(columns = ['Sputnik V (Doses Administered)', 'AEFI','18-44 Years (D
                                         '45-60 Years (Doses Administered)'],axis = 1)
```

In [27]:
```
vaccination.head()
```

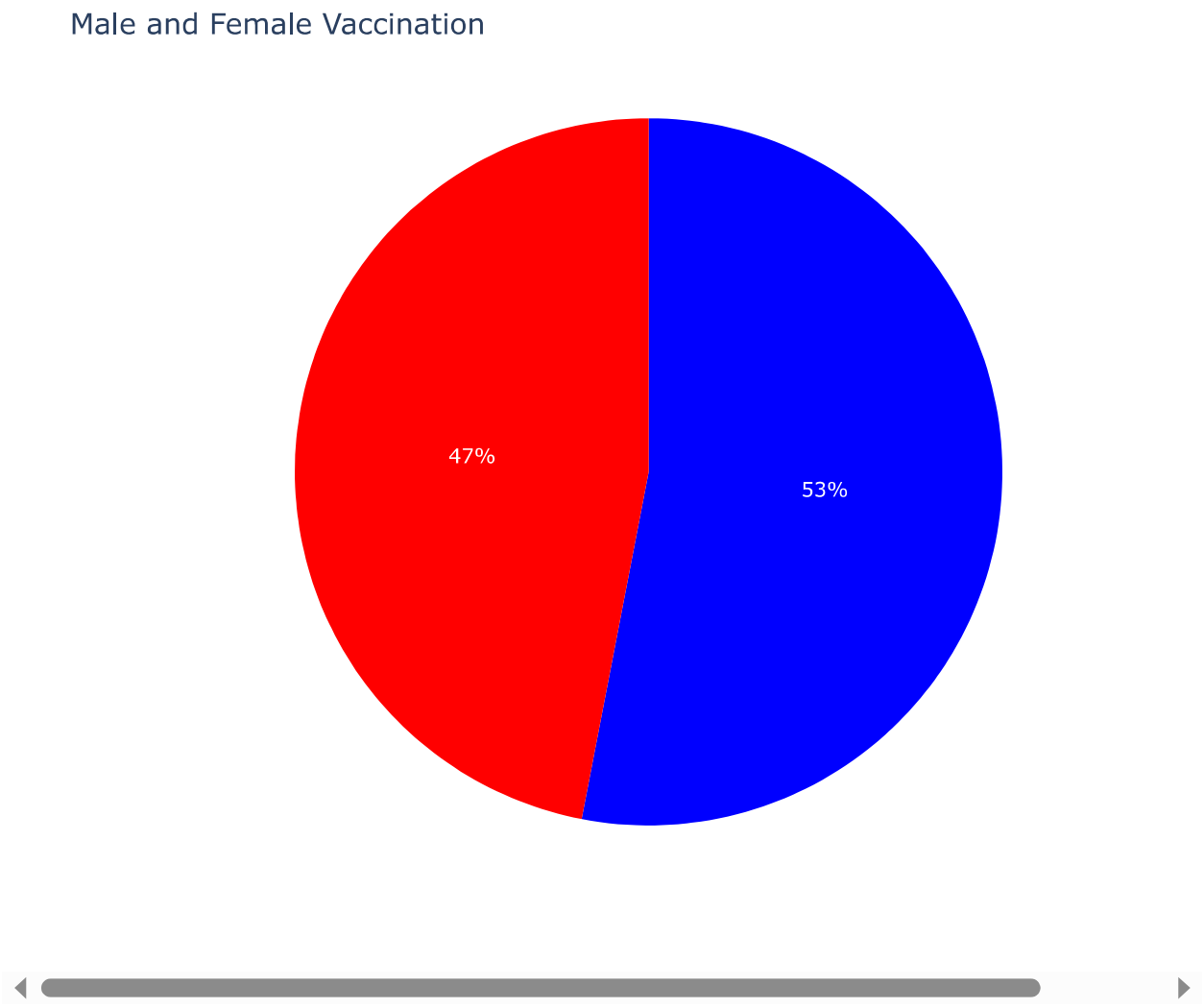
Out[27]:

| | Vaccine_date | State | Total Doses Administered | Sessions | Sites | First Dose Administered | Second Dose Administered | Male (Doses Administered) | Adm |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 16-01-21 | India | 48276.0 | 3455.0 | 2957.0 | 48276.0 | 0.0 | NaN | |
| **1** | 17-01-21 | India | 58604.0 | 8532.0 | 4954.0 | 58604.0 | 0.0 | NaN | |
| **2** | 18-01-21 | India | 99449.0 | 13611.0 | 6583.0 | 99449.0 | 0.0 | NaN | |
| **3** | 19-01-21 | India | 195525.0 | 17855.0 | 7951.0 | 195525.0 | 0.0 | NaN | |
| **4** | 20-01-21 | India | 251280.0 | 25472.0 | 10504.0 | 251280.0 | 0.0 | NaN | |

# Plot for male vs female vaccination using plotly library

In [28]:
```
## first will filter out male individuals from the data frame
male = vaccination["Male(Individuals Vaccinated)"].sum()
```

```
female =  vaccination["Female(Individuals Vaccinated)"].sum()

fig = px.pie(names=["Male", "Female"],
             values=[male, female],
             title = "Male and Female Vaccination",
            color=["Male", "Female"],  # This is to match the names to colors
            color_discrete_map={"Male": "blue", "Female": "red"}  # Custom colors)  ##pie() funct
            )

# Update the layout to increase the size of the chart
fig.update_layout(
    width=800,  # Set the width of the pie chart
    height=600  # Set the height of the pie chart
)
```




## Remove rows where state = India

- To get the state-wise instead of country

In [29]:
```
vaccine = vaccine_df[vaccine_df.State!='India']
vaccine  ## The filtered DataFrame is assigned to a new variable called vaccine.
```

Out[29]:

| | Vaccine_date | State | Total Doses Administered | Sessions | Sites | First Dose Administered | Second Dose Administered | Male (Doses Administered) |
|---|---|---|---|---|---|---|---|---|
| **212** | 16-01-21 | Andaman and Nicobar Islands | 23.0 | 2.0 | 2.0 | 23.0 | 0.0 | 12.0 |
| **213** | 17-01-21 | Andaman and Nicobar Islands | 23.0 | 2.0 | 2.0 | 23.0 | 0.0 | 12.0 |
| **214** | 18-01-21 | Andaman and Nicobar Islands | 42.0 | 9.0 | 2.0 | 42.0 | 0.0 | 29.0 |
| **215** | 19-01-21 | Andaman and Nicobar Islands | 89.0 | 12.0 | 2.0 | 89.0 | 0.0 | 53.0 |
| **216** | 20-01-21 | Andaman and Nicobar Islands | 124.0 | 16.0 | 3.0 | 124.0 | 0.0 | 67.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **7840** | 11-08-21 | West Bengal | NaN | NaN | NaN | NaN | NaN | NaN |
| **7841** | 12-08-21 | West Bengal | NaN | NaN | NaN | NaN | NaN | NaN |
| **7842** | 13-08-21 | West Bengal | NaN | NaN | NaN | NaN | NaN | NaN |
| **7843** | 14-08-21 | West Bengal | NaN | NaN | NaN | NaN | NaN | NaN |
| **7844** | 15-08-21 | West Bengal | NaN | NaN | NaN | NaN | NaN | NaN |

7633 rows × 24 columns

# Rename the last column "Total Individual Vaccinated" to "Total"

In [30]:
```
vaccine.rename(columns = {"Total Individuals Vaccinated" : "Total"}, inplace = True)
vaccine.head()
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_13480\3627174491.py:1: SettingWithCopyWarning:


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/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[30]:

| | Vaccine_date | State | Total Doses Administered | Sessions | Sites | First Dose Administered | Second Dose Administered | Male (Doses Administered) | A |
|---|---|---|---|---|---|---|---|---|---|
| **212** | 16-01-21 | Andaman and Nicobar Islands | 23.0 | 2.0 | 2.0 | 23.0 | 0.0 | 12.0 | |
| **213** | 17-01-21 | Andaman and Nicobar Islands | 23.0 | 2.0 | 2.0 | 23.0 | 0.0 | 12.0 | |
| **214** | 18-01-21 | Andaman and Nicobar Islands | 42.0 | 9.0 | 2.0 | 42.0 | 0.0 | 29.0 | |
| **215** | 19-01-21 | Andaman and Nicobar Islands | 89.0 | 12.0 | 2.0 | 89.0 | 0.0 | 53.0 | |
| **216** | 20-01-21 | Andaman and Nicobar Islands | 124.0 | 16.0 | 3.0 | 124.0 | 0.0 | 67.0 | |

5 rows × 24 columns

# Visualization

In [31]:
```
## Find states with most number of individuals vaccinated

max_vac = vaccine.groupby('State')['Total'].sum().to_frame('Total')
max_vac = max_vac.sort_values('Total',ascending = False)[:5]
max_vac
```

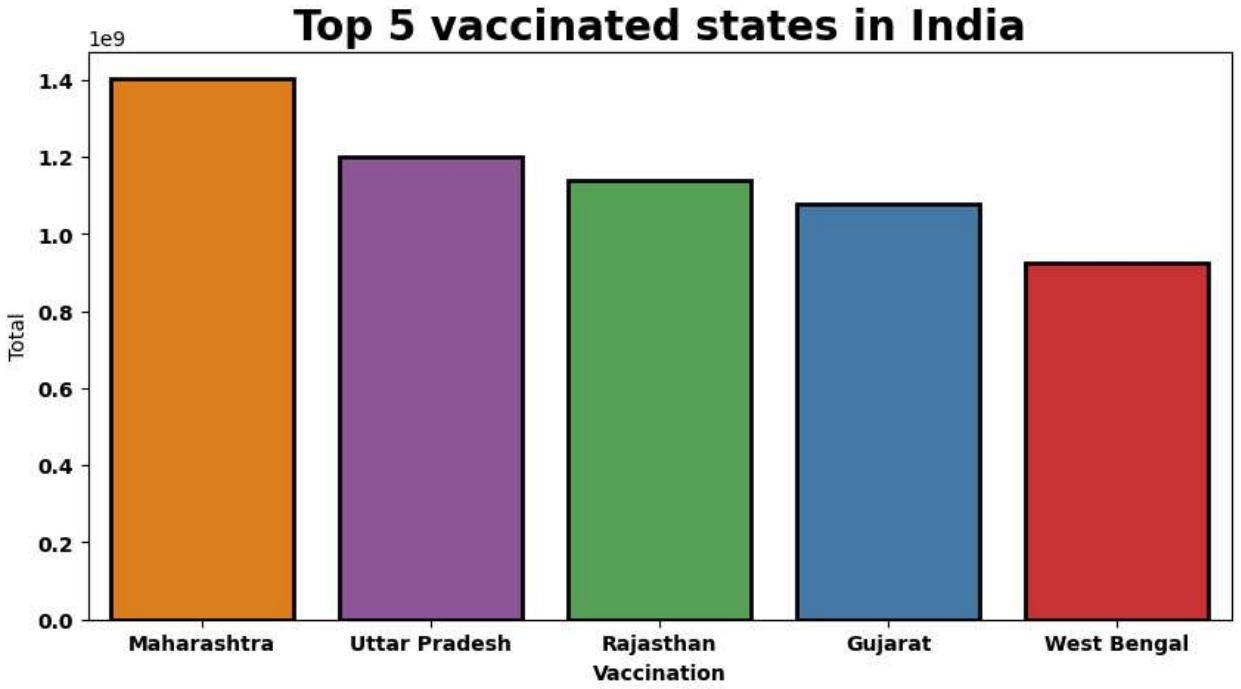
Out[31]:

| State | Total |
|---|---|
| **Maharashtra** | 1.403075e+09 |
| **Uttar Pradesh** | 1.200575e+09 |
| **Rajasthan** | 1.141163e+09 |
| **Gujarat** | 1.078261e+09 |
| **West Bengal** | 9.250227e+08 |

# Convert the above table into Bar chart

In [32]:
```
fig = plt.figure(figsize=(10,5))
plt.title("Top 5 vaccinated states in India", size = 20,fontweight="bold")
x = sns.barplot(data = max_vac.iloc[:10],y=max_vac.Total,
x=max_vac.index, linewidth = 2,
                edgecolor="black",
```

```
                 hue="Total",
                palette="Set1"
               )
plt.legend().set_visible(False)
plt.xlabel("States", fontweight="bold")
plt.xlabel("Vaccination", fontweight="bold")
plt.xticks(fontweight='bold')
plt.yticks(fontweight='bold')
plt.show()
```



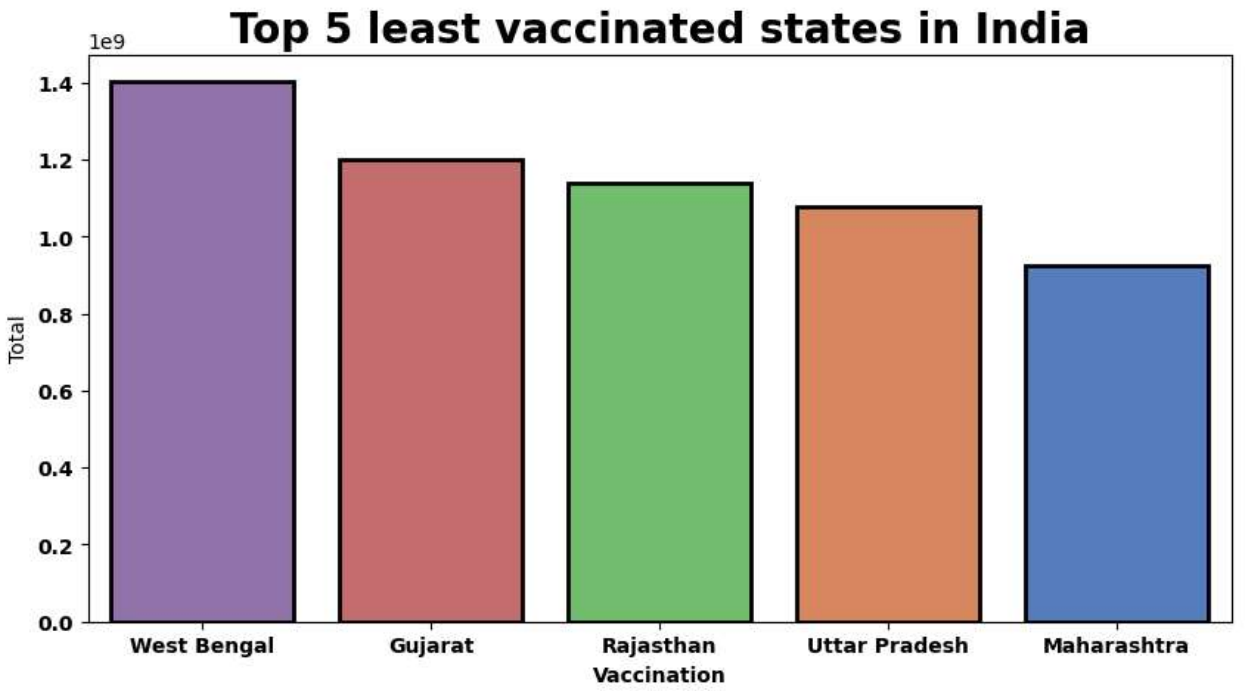

In [33]:
```
## Find states with least number of individuals vaccinated

min_vac = vaccine.groupby('State')['Total'].sum().to_frame('Total')
min_vac = max_vac.sort_values('Total',ascending = True)[:5]
min_vac
```

Out[33]:

| State | Total |
|---|---|
| West Bengal | 9.250227e+08 |
| Gujarat | 1.078261e+09 |
| Rajasthan | 1.141163e+09 |
| Uttar Pradesh | 1.200575e+09 |
| Maharashtra | 1.403075e+09 |

In [35]:
```
fig = plt.figure(figsize=(10,5))
plt.title("Top 5 least vaccinated states in India", size = 20,fontweight="bold")
x = sns.barplot(data = max_vac.iloc[:10],y=max_vac.Total,
x=min_vac.index, linewidth = 2,
                edgecolor="black",
                 hue="Total",
                palette="muted"
               )
plt.legend().set_visible(False)
plt.xlabel("States", fontweight="bold")
```

```
plt.xlabel("Vaccination", fontweight="bold")
plt.xticks(fontweight='bold')
plt.yticks(fontweight='bold')
plt.show()
```




In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: