**COVID Vaccines Analysis**

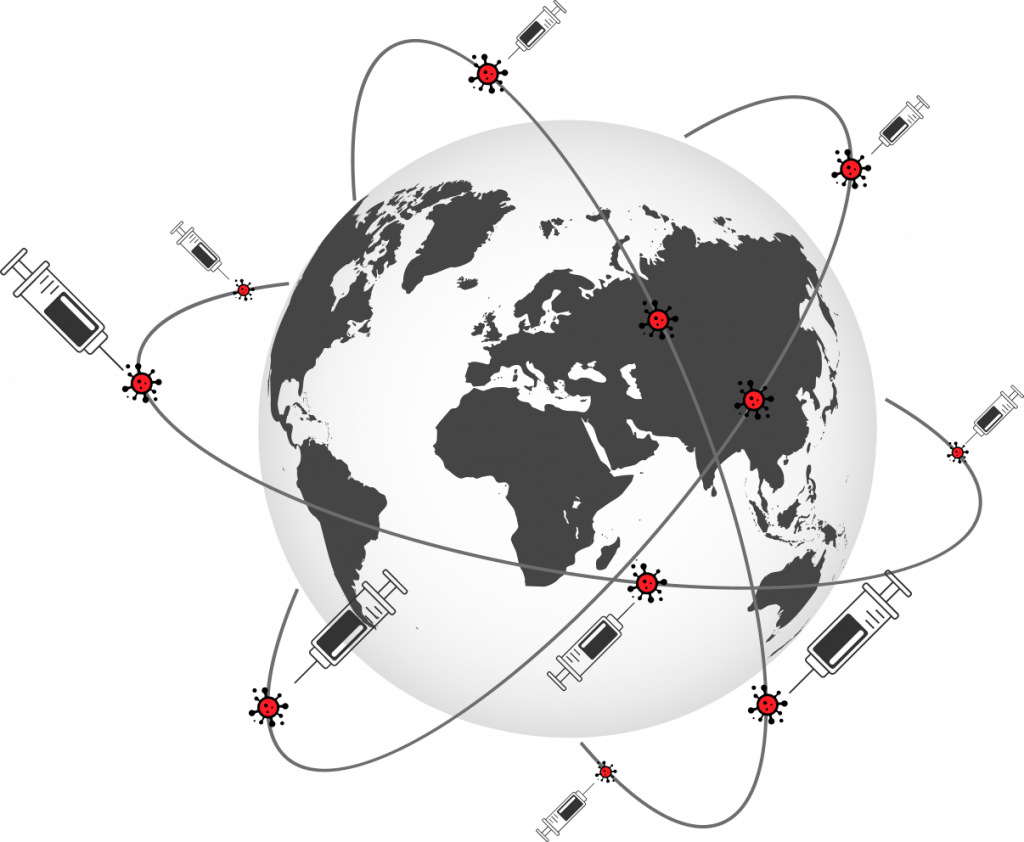
**Phase 4 Submission Document**

**Project Title:** COVID Vaccines Analysis

**Phase 3 : *Development part 2***

**Topic :** Covid-19 vaccines analysis by performing: EDA, Statistical analysis, Visualization.

****

****

COVID Vaccines Analysis

**COVID Vaccines Analysis**

**Introduction:**

* The "COVID Vaccines Analysis" project is a comprehensive exploration of critical aspects related to COVID-19 vaccines. This project delves into the data surrounding vaccine distribution, administration, and the occurrence of adverse effects, aiming to provide insights that contribute to informed decision-making and public health strategies.
* Our analysis encompasses a wide range of methodologies, from data mining and machine learning techniques to statistical modelling and trend analysis. By leveraging advanced data analytics, we seek to uncover hidden patterns, identify areas of concern, and make meaningful predictions regarding the trajectory of the pandemic and the success of vaccination efforts.

**Given data set:**

**Dataset Link:** <https://www.kaggle.com/datasets/gpreda/covid-world-vaccination-progress>

**Exploratory data analysis:**

"In this comprehensive exploratory data analysis (EDA), we delve into the wealth of information provided by the 'COVID-19 World Vaccination Progress' dataset from Kaggle. Our aim is to gain a deep understanding of the global landscape of COVID-19 vaccination efforts. Through this analysis, we explore the trends, patterns, and insights that emerge from the data. We examine the distribution of vaccine types, demographics, and regional disparities in vaccination rates. We also investigate potential correlations between vaccine distribution and COVID-19 infection rates. Our data-driven approach not only provides a snapshot of the progress made in the fight against the pandemic but also offers valuable insights to inform public health policies and interventions. Join us on this data journey as we uncover the narrative behind the numbers and visualize the path towards global herd immunity."

**Program:**

In [1]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

from plotly.subplots import make\_subplots

from datetime import datetime

## **Exploratory Data Analysis of Covid\_19\_India Dataset**

In [2]:

covid\_data = pd.read\_csv("../input/covid19-in-india/covid\_19\_india.csv")

covid\_data['Date'] = covid\_data['Date'].astype('datetime64[ns]')

covid\_data.head()

Out[2]:

|  | Sno | Date | Time | State/UnionTerritory | ConfirmedIndianNational | ConfirmedForeignNational | Cured | Deaths | Confirmed |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2020-01-30 | 6:00 PM | Kerala | 1 | 0 | 0 | 0 | 1 |
| 1 | 2 | 2020-01-31 | 6:00 PM | Kerala | 1 | 0 | 0 | 0 | 1 |
| 2 | 3 | 2020-02-01 | 6:00 PM | Kerala | 2 | 0 | 0 | 0 | 2 |
| 3 | 4 | 2020-02-02 | 6:00 PM | Kerala | 3 | 0 | 0 | 0 | 3 |
| 4 | 5 | 2020-02-03 | 6:00 PM | Kerala | 3 | 0 | 0 | 0 | 3 |

In [3]:

covid\_data.shape

Out[3]:

(14150, 9)

In [4]:

covid\_data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 14150 entries, 0 to 14149

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Sno 14150 non-null int64

1 Date 14150 non-null datetime64[ns]

2 Time 14150 non-null object

3 State/UnionTerritory 14150 non-null object

4 ConfirmedIndianNational 14150 non-null object

5 ConfirmedForeignNational 14150 non-null object

6 Cured 14150 non-null int64

7 Deaths 14150 non-null int64

8 Confirmed 14150 non-null int64

dtypes: datetime64[ns](1), int64(4), object(4)

memory usage: 995.0+ KB

In [5]:

covid\_data.describe()

Out[5]:

|  | Sno | Cured | Deaths | Confirmed |
| --- | --- | --- | --- | --- |
| count | 14150.00000 | 1.415000e+04 | 14150.000000 | 1.415000e+04 |
| mean | 7075.50000 | 1.540126e+05 | 2471.903816 | 1.681579e+05 |
| std | 4084.89749 | 3.105335e+05 | 6607.935316 | 3.384063e+05 |
| min | 1.00000 | 0.000000e+00 | 0.000000 | 0.000000e+00 |
| 25% | 3538.25000 | 1.225250e+03 | 10.000000 | 2.355250e+03 |
| 50% | 7075.50000 | 1.677250e+04 | 318.000000 | 2.102550e+04 |
| 75% | 10612.75000 | 1.782590e+05 | 1915.000000 | 2.059975e+05 |
| max | 14150.00000 | 3.330747e+06 | 62479.000000 | 4.094840e+06 |

## **Statewise Analysis**

In [6]:

state\_wise = covid\_data.groupby('State/UnionTerritory')['Confirmed','Cured','Deaths'].sum().reset\_index()

state\_wise["Death\_percentage"] = ((state\_wise["Deaths"] / state\_wise["Confirmed"]) \* 100)

state\_wise.style.background\_gradient(cmap='magma')

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.

"""Entry point for launching an IPython kernel.

Out[6]:

|  | State/UnionTerritory | Confirmed | Cured | Deaths | Death\_percentage |
| --- | --- | --- | --- | --- | --- |
| 0 | Andaman and Nicobar Islands | 1161624 | 1095239 | 14794 | 1.273562 |
| 1 | Andhra Pradesh | 205627560 | 195009537 | 1684138 | 0.819023 |
| 2 | Arunachal Pradesh | 3582775 | 3335949 | 10816 | 0.301889 |
| 3 | Assam | 51095731 | 47806964 | 231434 | 0.452942 |
| 4 | Bihar | 59502973 | 56326201 | 325733 | 0.547423 |
| 5 | Cases being reassigned to states | 345565 | 0 | 0 | 0.000000 |
| 6 | Chandigarh | 4497154 | 4098897 | 67640 | 1.504062 |
| 7 | Chhattisgarh | 60074333 | 53723225 | 694144 | 1.155475 |
| 8 | Dadra and Nagar Haveli and Daman and Diu | 866005 | 817775 | 582 | 0.067205 |
| 9 | Daman & Diu | 2 | 0 | 0 | 0.000000 |
| 10 | Delhi | 134827188 | 126242510 | 2399310 | 1.779545 |
| 11 | Goa | 11678197 | 10837849 | 161118 | 1.379648 |
| 12 | Gujarat | 58279923 | 52900802 | 1177367 | 2.020193 |
| 13 | Haryana | 55563899 | 51919242 | 605094 | 1.089006 |
| 14 | Himachal Pradesh | 10373705 | 9370148 | 166062 | 1.600797 |
| 15 | Jammu and Kashmir | 26971141 | 24677307 | 420238 | 1.558102 |
| 16 | Jharkhand | 26446302 | 24562965 | 236267 | 0.893384 |
| 17 | Karnataka | 209493180 | 192754571 | 2813076 | 1.342801 |
| 18 | Kerala | 166898288 | 151310286 | 657076 | 0.393698 |
| 19 | Ladakh | 2018868 | 1830434 | 25147 | 1.245599 |
| 20 | Lakshadweep | 46403 | 34913 | 57 | 0.122837 |
| 21 | Madhya Pradesh | 54165387 | 49471433 | 857399 | 1.582928 |
| 22 | Maharashtra | 494842275 | 432346106 | 12168696 | 2.459106 |
| 23 | Manipur | 5691040 | 5196442 | 63765 | 1.120445 |
| 24 | Meghalaya | 2721780 | 2485346 | 26987 | 0.991520 |
| 25 | Mizoram | 877555 | 807083 | 1442 | 0.164320 |
| 26 | Nagaland | 2557096 | 2337373 | 15235 | 0.595793 |
| 27 | Odisha | 72946835 | 69441871 | 384313 | 0.526840 |
| 28 | Puducherry | 8743641 | 8055805 | 146461 | 1.675057 |
| 29 | Punjab | 39895309 | 35785710 | 1215922 | 3.047782 |
| 30 | Rajasthan | 64864781 | 59970518 | 606176 | 0.934523 |
| 31 | Sikkim | 1219550 | 1111371 | 23817 | 1.952933 |
| 32 | Tamil Nadu | 198338601 | 186328161 | 2969358 | 1.497116 |
| 33 | Telengana | 66667750 | 61921079 | 383400 | 0.575091 |
| 34 | Tripura | 7596122 | 7049046 | 84812 | 1.116517 |
| 35 | Unassigned | 161 | 0 | 0 | 0.000000 |
| 36 | Uttar Pradesh | 133829373 | 122698961 | 1930554 | 1.442549 |
| 37 | Uttarakhand | 19218385 | 17580606 | 311929 | 1.623076 |
| 38 | West Bengal | 115907271 | 108036513 | 2097080 | 1.809274 |

In [7]:

px.bar(x=state\_wise.nlargest(10,"Confirmed")["State/UnionTerritory"],

y = state\_wise.nlargest(10,"Confirmed")["Confirmed"],

color\_discrete\_sequence=px.colors.diverging.Picnic,

title="Top 10 states with highest number of Confirmed cases")

MaharashtraKarnatakaAndhra PradeshTamil NaduKeralaDelhiUttar PradeshWest BengalOdishaTelengana0100M200M300M400M500M

Top 10 states with highest number of Confirmed casesxy

In [8]:

px.bar(x=state\_wise.nlargest(10,"Cured")["State/UnionTerritory"],

y = state\_wise.nlargest(10,"Cured")["Cured"],

color\_discrete\_sequence=px.colors.sequential.Sunset,

title="Top 10 states with highest number of Cured cases")

MaharashtraAndhra PradeshKarnatakaTamil NaduKeralaDelhiUttar PradeshWest BengalOdishaTelengana050M100M150M200M250M300M350M400M450M

Top 10 states with highest number of Cured casesxy

In [9]:

px.bar(x=state\_wise.nlargest(10,"Deaths")["State/UnionTerritory"],

y = state\_wise.nlargest(10,"Deaths")["Deaths"],

color\_discrete\_sequence=px.colors.diverging.curl,

title="Top 10 states with highest number of Deaths")

MaharashtraTamil NaduKarnatakaDelhiWest BengalUttar PradeshAndhra PradeshPunjabGujaratMadhya Pradesh02M4M6M8M10M12M

Top 10 states with highest number of Deathsxy

In [10]:

px.bar(x=state\_wise.nlargest(10,"Death\_percentage")["State/UnionTerritory"],

y = state\_wise.nlargest(10,"Death\_percentage")["Death\_percentage"],

color\_discrete\_sequence=px.colors.diverging.Portland,

title="Top 10 states with highest of Death percentage")

PunjabMaharashtraGujaratSikkimWest BengalDelhiPuducherryUttarakhandHimachal PradeshMadhya Pradesh00.511.522.53

Top 10 states with highest of Death percentagexy

## **Monthwise Analysis**

In [11]:

month\_wise = covid\_data.groupby(pd.Grouper(key='Date',freq='M')).sum()

month\_wise = month\_wise.drop(['Sno'], axis = 1)

month\_wise['Date'] = month\_wise.index

first\_column = month\_wise.pop('Date')

month\_wise.insert(0, 'Date', first\_column)

index = [x for x **in** range(len(month\_wise))]

month\_wise['index'] = index

month\_wise = month\_wise.set\_index('index')

second\_column = month\_wise.pop('Confirmed')

month\_wise.insert(1, 'Confirmed', second\_column)

month\_wise["Death\_percentage"] = ((month\_wise["Deaths"] / month\_wise["Confirmed"]) \* 100)

month\_wise.style.background\_gradient(cmap='twilight\_shifted')

Out[11]:

|  | Date | Confirmed | Cured | Deaths | Death\_percentage |
| --- | --- | --- | --- | --- | --- |
| index |  |  |  |  |  |
| 0 | 2020-01-31 00:00:00 | 2 | 0 | 0 | 0.000000 |
| 1 | 2020-02-29 00:00:00 | 86 | 0 | 0 | 0.000000 |
| 2 | 2020-03-31 00:00:00 | 9687 | 808 | 202 | 2.085269 |
| 3 | 2020-04-30 00:00:00 | 422442 | 75443 | 13270 | 3.141260 |
| 4 | 2020-05-31 00:00:00 | 2938234 | 1133341 | 89834 | 3.057415 |
| 5 | 2020-06-30 00:00:00 | 10558374 | 5668946 | 319690 | 3.027834 |
| 6 | 2020-07-31 00:00:00 | 31726501 | 19980130 | 793511 | 2.501098 |
| 7 | 2020-08-31 00:00:00 | 80749620 | 58580895 | 1553468 | 1.923808 |
| 8 | 2020-09-30 00:00:00 | 149113758 | 118592934 | 2443374 | 1.638597 |
| 9 | 2020-10-31 00:00:00 | 226770312 | 198824412 | 3457615 | 1.524721 |
| 10 | 2020-11-30 00:00:00 | 264556412 | 246213201 | 3894165 | 1.471960 |
| 11 | 2020-12-31 00:00:00 | 307177353 | 292244085 | 4457379 | 1.451077 |
| 12 | 2021-01-31 00:00:00 | 326469747 | 315332019 | 4709167 | 1.442451 |
| 13 | 2021-02-28 00:00:00 | 305631803 | 297133802 | 4359434 | 1.426368 |
| 14 | 2021-03-31 00:00:00 | 356305616 | 342610397 | 4935253 | 1.385118 |
| 15 | 2021-04-30 00:00:00 | 317003781 | 282887825 | 3951077 | 1.246382 |

pIn [12]:

fig = px.bar(month\_wise, x='Date', y='Confirmed',

hover\_data=['Cured', 'Deaths'], color='Date',

labels={'Date':'Date(monthwise)'},

title="Monthwise Increase in Confirmed cases")

fig.show()

Jul 2020Jan 2021050M100M150M200M250M300M350M

Date(monthwise)2020-01-31 00:00:002020-02-29 00:00:002020-03-31 00:00:002020-04-30 00:00:002020-05-31 00:00:002020-06-30 00:00:002020-07-31 00:00:002020-08-31 00:00:002020-09-30 00:00:002020-10-31 00:00:002020-11-30 00:00:002020-12-31 00:00:002021-01-31 00:00:002021-02-28 00:00:002021-03-31 00:00:002021-04-30 00:00:00Monthwise Increase in Confirmed casesDate(monthwise)Confirmed

In [13]:

fig = px.bar(month\_wise, x='Date', y='Cured',

hover\_data=['Confirmed','Deaths'], color='Date',

labels={'Date':'Date(monthwise)'},

title="Monthwise Increase in Cured cases")

fig.show()

Jul 2020Jan 2021050M100M150M200M250M300M350M

Date(monthwise)2020-01-31 00:00:002020-02-29 00:00:002020-03-31 00:00:002020-04-30 00:00:002020-05-31 00:00:002020-06-30 00:00:002020-07-31 00:00:002020-08-31 00:00:002020-09-30 00:00:002020-10-31 00:00:002020-11-30 00:00:002020-12-31 00:00:002021-01-31 00:00:002021-02-28 00:00:002021-03-31 00:00:002021-04-30 00:00:00Monthwise Increase in Cured casesDate(monthwise)Cured

In [14]:

fig = px.bar(month\_wise, x='Date', y='Deaths',

hover\_data=['Confirmed','Cured'], color='Date',

labels={'Date':'Date(monthwise)'},

title="Monthwise Increase in Deaths cases")

fig.show()

Jul 2020Jan 202101M2M3M4M5M

Date(monthwise)2020-01-31 00:00:002020-02-29 00:00:002020-03-31 00:00:002020-04-30 00:00:002020-05-31 00:00:002020-06-30 00:00:002020-07-31 00:00:002020-08-31 00:00:002020-09-30 00:00:002020-10-31 00:00:002020-11-30 00:00:002020-12-31 00:00:002021-01-31 00:00:002021-02-28 00:00:002021-03-31 00:00:002021-04-30 00:00:00Monthwise Increase in Deaths casesDate(monthwise)Deaths

In [15]:

fig = px.bar(month\_wise ,

x='Date',

y='Death\_percentage' ,

hover\_data=['Confirmed','Deaths'],color='Date',

labels={'Death\_percentage':'Death percentage'},

title="Top 10 states with highest of Death percentage")

fig.show()

Jul 2020Jan 202100.511.522.53

Date2020-01-31 00:00:002020-02-29 00:00:002020-03-31 00:00:002020-04-30 00:00:002020-05-31 00:00:002020-06-30 00:00:002020-07-31 00:00:002020-08-31 00:00:002020-09-30 00:00:002020-10-31 00:00:002020-11-30 00:00:002020-12-31 00:00:002021-01-31 00:00:002021-02-28 00:00:002021-03-31 00:00:002021-04-30 00:00:00Top 10 states with highest of Death percentageDateDeath percentage

## **Exploratory Data Analysis of StatewiseTestingDetails Dataset**

In [16]:

covid\_testing = pd.read\_csv("../input/covid19-in-india/StatewiseTestingDetails.csv")

covid\_testing['Date'] = covid\_testing['Date'].astype('datetime64[ns]')

covid\_testing.head()

Out[16]:

|  | Date | State | TotalSamples | Negative | Positive |
| --- | --- | --- | --- | --- | --- |
| 0 | 2020-04-17 | Andaman and Nicobar Islands | 1403.0 | 1210.0 | 12.0 |
| 1 | 2020-04-24 | Andaman and Nicobar Islands | 2679.0 | NaN | 27.0 |
| 2 | 2020-04-27 | Andaman and Nicobar Islands | 2848.0 | NaN | 33.0 |
| 3 | 2020-05-01 | Andaman and Nicobar Islands | 3754.0 | NaN | 33.0 |
| 4 | 2020-04-02 | Andhra Pradesh | 1800.0 | 1175.0 | 132.0 |

In [17]:

covid\_testing.shape

Out[17]:

(926, 5)

In [18]:

covid\_testing.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 926 entries, 0 to 925

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 926 non-null datetime64[ns]

1 State 926 non-null object

2 TotalSamples 926 non-null float64

3 Negative 756 non-null float64

4 Positive 918 non-null float64

dtypes: datetime64[ns](1), float64(3), object(1)

memory usage: 36.3+ KB

In [19]:

covid\_testing['Negative'] = covid\_testing['TotalSamples'] - covid\_testing['Positive']

covid\_testing = covid\_testing.dropna()

covid\_testing.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 918 entries, 0 to 925

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 918 non-null datetime64[ns]

1 State 918 non-null object

2 TotalSamples 918 non-null float64

3 Negative 918 non-null float64

4 Positive 918 non-null float64

dtypes: datetime64[ns](1), float64(3), object(1)

memory usage: 43.0+ KB

## **Statewise Analysis**

In [20]:

covid\_testing\_state = covid\_testing.groupby('State')['TotalSamples','Negative','Positive'].max().reset\_index()

covid\_testing\_state["Positive\_percentage"] = ((covid\_testing["Positive"] / covid\_testing["TotalSamples"]) \* 100)

covid\_testing\_state.style.background\_gradient(cmap='gist\_earth\_r')

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:1: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.

Out[20]:

|  | State | TotalSamples | Negative | Positive | Positive\_percentage |
| --- | --- | --- | --- | --- | --- |
| 0 | Andaman and Nicobar Islands | 3754.000000 | 3721.000000 | 33.000000 | 0.855310 |
| 1 | Andhra Pradesh | 173735.000000 | 171755.000000 | 1980.000000 | 1.007839 |
| 2 | Arunachal Pradesh | 1823.000000 | 1821.000000 | 2.000000 | 1.158708 |
| 3 | Assam | 18002.000000 | 17940.000000 | 62.000000 | 0.879062 |
| 4 | Bihar | 36053.000000 | 35346.000000 | 707.000000 | 7.333333 |
| 5 | Chandigarh | 2142.000000 | 1969.000000 | 173.000000 | 5.726388 |
| 6 | Chhattisgarh | 25282.000000 | 25223.000000 | 59.000000 | 5.475711 |
| 7 | Delhi | 97678.000000 | 90445.000000 | 7233.000000 | 5.820638 |
| 8 | Goa | 4848.000000 | 4841.000000 | 7.000000 | 4.934323 |
| 9 | Gujarat | 113493.000000 | 105298.000000 | 8195.000000 | 4.502618 |
| 10 | Haryana | 56983.000000 | 56280.000000 | 703.000000 | 4.520796 |
| 11 | Himachal Pradesh | 10791.000000 | 10736.000000 | 55.000000 | 2.638992 |
| 12 | Jammu and Kashmir | 47080.000000 | 46219.000000 | 861.000000 | 2.826785 |
| 13 | Jharkhand | 20832.000000 | 20675.000000 | 157.000000 | 2.811189 |
| 14 | Karnataka | 107311.000000 | 106463.000000 | 848.000000 | 2.400030 |
| 15 | Kerala | 37464.000000 | 36952.000000 | 512.000000 | 2.349266 |
| 16 | Ladakh | 3503.000000 | 3461.000000 | 42.000000 | 2.117186 |
| 17 | Madhya Pradesh | 72069.000000 | 68455.000000 | 3614.000000 | 1.958470 |
| 18 | Maharashtra | 225524.000000 | 205296.000000 | 20228.000000 | 1.859177 |
| 19 | Meghalaya | 2287.000000 | 2275.000000 | 13.000000 | 1.757518 |
| 20 | Mizoram | 201.000000 | 200.000000 | 1.000000 | 1.658342 |
| 21 | Nagaland | 862.000000 | 862.000000 | 0.000000 | 1.612429 |
| 22 | Odisha | 59780.000000 | 59418.000000 | 362.000000 | 1.578785 |
| 23 | Puducherry | 4364.000000 | 4347.000000 | 17.000000 | 1.567207 |
| 24 | Punjab | 40962.000000 | 39139.000000 | 1823.000000 | 1.512588 |
| 25 | Rajasthan | 166424.000000 | 162610.000000 | 3814.000000 | 1.483745 |
| 26 | Sikkim | 219.000000 | 219.000000 | 0.000000 | 1.427874 |
| 27 | Tamil Nadu | 243037.000000 | 235833.000000 | 7204.000000 | 1.406788 |
| 28 | Telangana | 19278.000000 | 18262.000000 | 1016.000000 | 1.377276 |
| 29 | Tripura | 9091.000000 | 8955.000000 | 136.000000 | 1.317586 |
| 30 | Uttar Pradesh | 129955.000000 | 126488.000000 | 3467.000000 | 1.286219 |
| 31 | Uttarakhand | 9668.000000 | 9600.000000 | 68.000000 | 1.257839 |
| 32 | West Bengal | 43414.000000 | 41475.000000 | 1939.000000 | 1.227228 |

In [21]:

px.bar(x=covid\_testing\_state.nlargest(10,"TotalSamples")["State"],

y = covid\_testing\_state.nlargest(10,"TotalSamples")["TotalSamples"],

labels={'y':'Total Samples','x':'State'},

color\_discrete\_sequence=px.colors.sequential.haline,

title="Top 10 states with highest number of Total Samples")

Tamil NaduMaharashtraAndhra PradeshRajasthanUttar PradeshGujaratKarnatakaDelhiMadhya PradeshOdisha050k100k150k200k250k

Top 10 states with highest number of Total SamplesStateTotal Samples

In [22]:

px.bar(x=covid\_testing\_state.nlargest(10,"Negative")["State"],

y = covid\_testing\_state.nlargest(10,"Negative")["Negative"],

labels={'y':'Total Negative cases','x':'State'},

color\_discrete\_sequence=px.colors.sequential.turbid,

title="Top 10 states with highest number of Negative cases")

Tamil NaduMaharashtraAndhra PradeshRajasthanUttar PradeshKarnatakaGujaratDelhiMadhya PradeshOdisha050k100k150k200k

Top 10 states with highest number of Negative casesStateTotal Negative cases

In [23]:

px.bar(x=covid\_testing\_state.nlargest(10,"Positive")["State"],

y = covid\_testing\_state.nlargest(10,"Positive")["Positive"],

labels={'y':'Total Positive Cases','x':'State'},

color\_discrete\_sequence=px.colors.sequential.solar,

title="Top 10 states with highest number of Positive cases")

MaharashtraGujaratDelhiTamil NaduRajasthanMadhya PradeshUttar PradeshAndhra PradeshWest BengalPunjab05k10k15k20k

Top 10 states with highest number of Positive casesStateTotal Positive Cases

In [24]:

px.bar(x=covid\_testing\_state.nlargest(10,"Positive\_percentage")["State"],

y = covid\_testing\_state.nlargest(10,"Positive\_percentage")["Positive\_percentage"],

labels={'y':'Positive Percentage','x':'State'},

color\_discrete\_sequence=px.colors.sequential.Aggrnyl,

title="Top 10 states with highest Positive percentage",

height = 420)

BiharDelhiChandigarhChhattisgarhGoaHaryanaGujaratJammu and KashmirJharkhandHimachal Pradesh0246

Top 10 states with highest Positive percentageStatePositive Percentage

**Statistical analysis:**

"In addition to the exploratory data analysis, we performed a comprehensive statistical analysis to extract meaningful insights from the 'COVID-19 World Vaccination Progress' dataset. This analysis involved conducting various statistical tests, such as t-tests and chi-squared tests, to evaluate significant differences in vaccination rates between different demographic groups and regions. We also employed correlation analysis to assess the relationships between variables, including vaccine distribution and COVID-19 infection rates. Time series analysis allowed us to identify trends and seasonality in vaccination progress over time. Furthermore, we used regression analysis to model the factors influencing vaccination rates, shedding light on the impact of various variables, such as healthcare infrastructure, population density, and socioeconomic factors. Our statistical analysis aims to provide a quantitative perspective on the patterns and determinants of COVID-19 vaccine distribution, contributing to evidence-based decision-making in the ongoing fight against the pandemic."

Top of Form

**Visualization:**

"As part of our analysis of the 'COVID-19 World Vaccination Progress' dataset, we leveraged various visualization techniques to bring the data to life. We employed bar charts and pie charts to vividly illustrate the distribution of vaccine types, age groups, and gender among those receiving vaccinations. Time series line plots and heatmaps were utilized to visualize trends and correlations over time and between variables. Choropleth maps enabled us to portray the geographic distribution of vaccination rates, highlighting regional disparities. We also created interactive dashboards that offer a user-friendly interface for exploring the data visually and dynamically. These visualizations serve as powerful tools to convey the intricate narrative within the dataset, making it more accessible and informative for a broad audience. By combining these techniques, we aim to provide a holistic and insightful perspective on the global COVID-19 vaccination progress."

**Program:**

*# This Python 3 environment comes with many helpful analytics libraries installed*

*# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python*

*# For example, here's several helpful packages to load*

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

*# Input data files are available in the read-only "../input/" directory*

*# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory*

import os

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

*# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"*

*# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session*

/kaggle/input/covid-world-vaccination-progress/country\_vaccinations\_by\_manufacturer.csv

/kaggle/input/covid-world-vaccination-progress/country\_vaccinations.csv

In [2]:

*#importing essential libraries*

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

In [3]:

*#reading dataset*

df= pd.read\_csv("../input/covid-world-vaccination-progress/country\_vaccinations.csv")

In [4]:

df.head()

Out[4]:

|  | country | iso\_code | date | total\_vaccinations | people\_vaccinated | people\_fully\_vaccinated | daily\_vaccinations\_raw | daily\_vaccinations | total\_vaccinations\_per\_hundred | people\_vaccinated\_per\_hundred | people\_fully\_vaccinated\_per\_hundred | daily\_vaccinations\_per\_million | vaccines | source\_name | source\_website |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Afghanistan | AFG | 2021-02-22 | 0.0 | 0.0 | NaN | NaN | NaN | 0.0 | 0.0 | NaN | NaN | Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi... | World Health Organization | https://covid19.who.int/ |
| 1 | Afghanistan | AFG | 2021-02-23 | NaN | NaN | NaN | NaN | 1367.0 | NaN | NaN | NaN | 34.0 | Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi... | World Health Organization | https://covid19.who.int/ |
| 2 | Afghanistan | AFG | 2021-02-24 | NaN | NaN | NaN | NaN | 1367.0 | NaN | NaN | NaN | 34.0 | Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi... | World Health Organization | https://covid19.who.int/ |
| 3 | Afghanistan | AFG | 2021-02-25 | NaN | NaN | NaN | NaN | 1367.0 | NaN | NaN | NaN | 34.0 | Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi... | World Health Organization | https://covid19.who.int/ |
| 4 | Afghanistan | AFG | 2021-02-26 | NaN | NaN | NaN | NaN | 1367.0 | NaN | NaN | NaN | 34.0 | Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi... | World Health Organization | https://covid19.who.int/ |

In [5]:

df.describe()

Out[5]:

|  | total\_vaccinations | people\_vaccinated | people\_fully\_vaccinated | daily\_vaccinations\_raw | daily\_vaccinations | total\_vaccinations\_per\_hundred | people\_vaccinated\_per\_hundred | people\_fully\_vaccinated\_per\_hundred | daily\_vaccinations\_per\_million |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| count | 4.360700e+04 | 4.129400e+04 | 3.880200e+04 | 3.536200e+04 | 8.621300e+04 | 43607.000000 | 41294.000000 | 38802.000000 | 86213.000000 |
| mean | 4.592964e+07 | 1.770508e+07 | 1.413830e+07 | 2.705996e+05 | 1.313055e+05 | 80.188543 | 40.927317 | 35.523243 | 3257.049157 |
| std | 2.246004e+08 | 7.078731e+07 | 5.713920e+07 | 1.212427e+06 | 7.682388e+05 | 67.913577 | 29.290759 | 28.376252 | 3934.312440 |
| min | 0.000000e+00 | 0.000000e+00 | 1.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 5.264100e+05 | 3.494642e+05 | 2.439622e+05 | 4.668000e+03 | 9.000000e+02 | 16.050000 | 11.370000 | 7.020000 | 636.000000 |
| 50% | 3.590096e+06 | 2.187310e+06 | 1.722140e+06 | 2.530900e+04 | 7.343000e+03 | 67.520000 | 41.435000 | 31.750000 | 2050.000000 |
| 75% | 1.701230e+07 | 9.152520e+06 | 7.559870e+06 | 1.234925e+05 | 4.409800e+04 | 132.735000 | 67.910000 | 62.080000 | 4682.000000 |
| max | 3.263129e+09 | 1.275541e+09 | 1.240777e+09 | 2.474100e+07 | 2.242429e+07 | 345.370000 | 124.760000 | 122.370000 | 117497.000000 |

In [6]:

df.dtypes

Out[6]:

country object

iso\_code object

date object

total\_vaccinations float64

people\_vaccinated float64

people\_fully\_vaccinated float64

daily\_vaccinations\_raw float64

daily\_vaccinations float64

total\_vaccinations\_per\_hundred float64

people\_vaccinated\_per\_hundred float64

people\_fully\_vaccinated\_per\_hundred float64

daily\_vaccinations\_per\_million float64

vaccines object

source\_name object

source\_website object

dtype: object

In [7]:

*#converting date column datatype to date*

df["date"]= pd.to\_datetime(df.date)

In [8]:

df["Total\_vaccinations(count)"]= df.groupby("country").total\_vaccinations.tail(1)

In [9]:

*#Top countries with most vaccinations*

df.groupby("country")["Total\_vaccinations(count)"].mean().sort\_values(ascending= False).head(20)

Out[9]:

country

China 3.263129e+09

India 1.834501e+09

United States 5.601818e+08

Brazil 4.135596e+08

Indonesia 3.771089e+08

Japan 2.543456e+08

Bangladesh 2.436427e+08

Pakistan 2.193686e+08

Vietnam 2.031444e+08

Mexico 1.919079e+08

Germany 1.719400e+08

Russia 1.636012e+08

Philippines 1.487991e+08

Turkey 1.468819e+08

Iran 1.467926e+08

France 1.416662e+08

United Kingdom 1.409683e+08

Italy 1.358709e+08

Thailand 1.288824e+08

South Korea 1.206045e+08

Name: Total\_vaccinations(count), dtype: float64

In [10]:

*#barplot visualization of top countries with most vaccinations*

x= df.groupby("country")["Total\_vaccinations(count)"].mean().sort\_values(ascending= False).head(20)

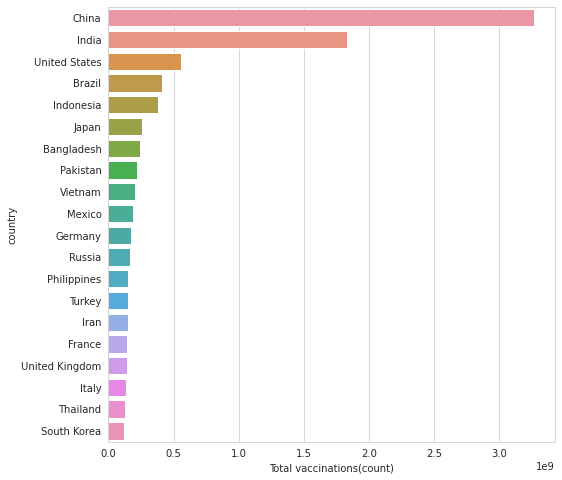
sns.set\_style("whitegrid")

plt.figure(figsize= (8,8))

ax= sns.barplot(x.values,x.index)

ax.set\_xlabel("Total vaccinations(count)")

plt.show()



In [11]:

*#Top countries with fully vaccinated peoples*

df["Full\_vaccinations(count)"]= df.groupby("country").people\_fully\_vaccinated.tail(1)

df.groupby("country")["Full\_vaccinations(count)"].mean().sort\_values(ascending= False).head(20)

Out[11]:

country

India 828229455.0

United States 217498967.0

Brazil 160272858.0

Indonesia 158830466.0

Bangladesh 107712737.0

Pakistan 101881176.0

Japan 100633737.0

Mexico 79711762.0

Vietnam 77754108.0

Russia 72841232.0

Philippines 65804988.0

Germany 63142649.0

Iran 56810058.0

Turkey 52968985.0

France 52438706.0

Thailand 50159803.0

United Kingdom 49404026.0

Italy 47817555.0

South Korea 44482876.0

England 41501690.0

Name: Full\_vaccinations(count), dtype: float64

In [12]:

*#barplot visualization of top countries with most full vaccinations*

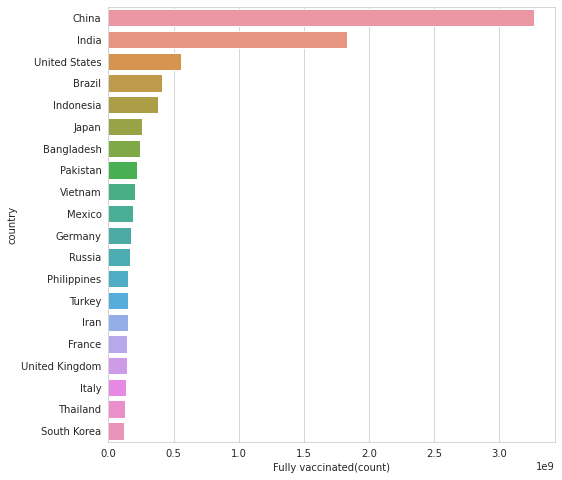
sns.set\_style("whitegrid")

plt.figure(figsize= (8,8))

ax= sns.barplot(x.values,x.index)

ax.set\_xlabel("Fully vaccinated(count)")

plt.show()



In [13]:

*#Vaccine types*

x=df.vaccines.unique()

y= list(x)

for i **in** y: print(i)

Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing

Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V

Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac, Sputnik V

Moderna, Oxford/AstraZeneca, Pfizer/BioNTech

Oxford/AstraZeneca

Oxford/AstraZeneca, Pfizer/BioNTech

Oxford/AstraZeneca, Pfizer/BioNTech, Sputnik V

CanSino, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V

Moderna, Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac, Sputnik V

Pfizer/BioNTech

Johnson&Johnson, Moderna, Novavax, Oxford/AstraZeneca, Pfizer/BioNTech

Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech

Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sputnik Light, Sputnik V

Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac

Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing

Sinopharm/Beijing, Sputnik V

Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech

Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac

Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing

Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V

Moderna, Pfizer/BioNTech

Covaxin, Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac

Johnson&Johnson, Oxford/AstraZeneca

Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing

Johnson&Johnson, Oxford/AstraZeneca, Sinopharm/Beijing

Sinopharm/Beijing

Johnson&Johnson, Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac

Covaxin, Oxford/AstraZeneca

CanSino, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac

CanSino, Sinopharm/Beijing, Sinopharm/Wuhan, Sinovac, ZF2001

Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac

Covaxin, Oxford/AstraZeneca, Sinopharm/Beijing

Moderna, Oxford/AstraZeneca, Sinopharm/Beijing, Sputnik V

Abdala, Soberana Plus, Soberana02

Johnson&Johnson, Moderna, Pfizer/BioNTech

Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac, Sputnik V

Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac

Covaxin, Johnson&Johnson, Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac

Johnson&Johnson, Pfizer/BioNTech

Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V

Oxford/AstraZeneca, Sputnik V

Moderna

Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sputnik V

Oxford/AstraZeneca, Sinopharm/Beijing

Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V

Johnson&Johnson, Moderna

Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sputnik V

Pfizer/BioNTech, Sinovac

Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V

Covaxin, Oxford/AstraZeneca, Sputnik V

Johnson&Johnson, Moderna, Novavax, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac

COVIran Barekat, Covaxin, FAKHRAVAC, Oxford/AstraZeneca, Razi Cov Pars, Sinopharm/Beijing, Soberana02, SpikoGen, Sputnik V

Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V

QazVac, Sinopharm/Beijing, Sputnik V

Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac, Sputnik Light, Sputnik V

Johnson&Johnson, Moderna, Novavax, Pfizer/BioNTech

Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac, Sputnik V

Pfizer/BioNTech, Sinopharm/Beijing

CanSino, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac

CanSino, Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V

Abdala, Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Soberana02, Sputnik Light, Sputnik V

Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac

CanSino, Covaxin, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac, Sputnik V

Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac, Sputnik Light, Sputnik V

Covaxin, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac, Sputnik V

EpiVacCorona, Sputnik V

Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac, Sputnik V

Pfizer/BioNTech, Sputnik V

Oxford/AstraZeneca, Sinopharm/Beijing, Sputnik V

Moderna, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac

Johnson&Johnson, Moderna, Novavax, Oxford/AstraZeneca, Pfizer/BioNTech, Sputnik V

Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac

Johnson&Johnson, Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac, Sputnik Light, Sputnik V

Medigen, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech

Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V

Johnson&Johnson, Pfizer/BioNTech, Sinopharm/Beijing

Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac

Pfizer/BioNTech, Sinovac, Turkovac

EpiVacCorona, Oxford/AstraZeneca, QazVac, Sinopharm/Beijing, Sputnik V, ZF2001

Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinopharm/Wuhan, Sputnik V

Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik Light, Sputnik V, ZF2001

Abdala, Sinopharm/Beijing, Sinovac, Soberana02, Sputnik Light, Sputnik V

Abdala, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V

Johnson&Johnson, Oxford/AstraZeneca, Sinovac

In [14]:

*#most common vaccines*

df.vaccines.value\_counts()

Out[14]:

Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech 7608

Moderna, Oxford/AstraZeneca, Pfizer/BioNTech 6263

Oxford/AstraZeneca 6022

Oxford/AstraZeneca, Pfizer/BioNTech 4629

Johnson&Johnson, Moderna, Novavax, Oxford/AstraZeneca, Pfizer/BioNTech 3564

...

Johnson&Johnson, Oxford/AstraZeneca, Sinovac 312

Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V 311

Johnson&Johnson, Moderna 251

Johnson&Johnson, Pfizer/BioNTech, Sinopharm/Beijing 228

EpiVacCorona, Oxford/AstraZeneca, QazVac, Sinopharm/Beijing, Sputnik V, ZF2001 190

Name: vaccines, Length: 84, dtype: int64

In [15]:

from wordcloud import WordCloud, STOPWORDS

plt.figure(figsize= (20,20))

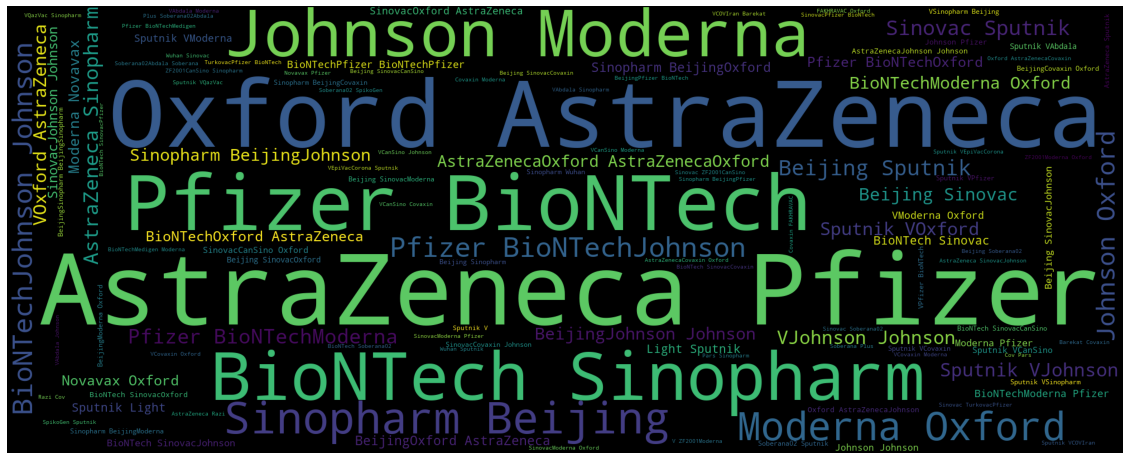
words= "".join(df["vaccines"])

final = WordCloud(width = 2000, height = 800, background\_color ="black",min\_font\_size = 10).generate(words)

plt.imshow(final)

plt.axis("off")

plt.show()

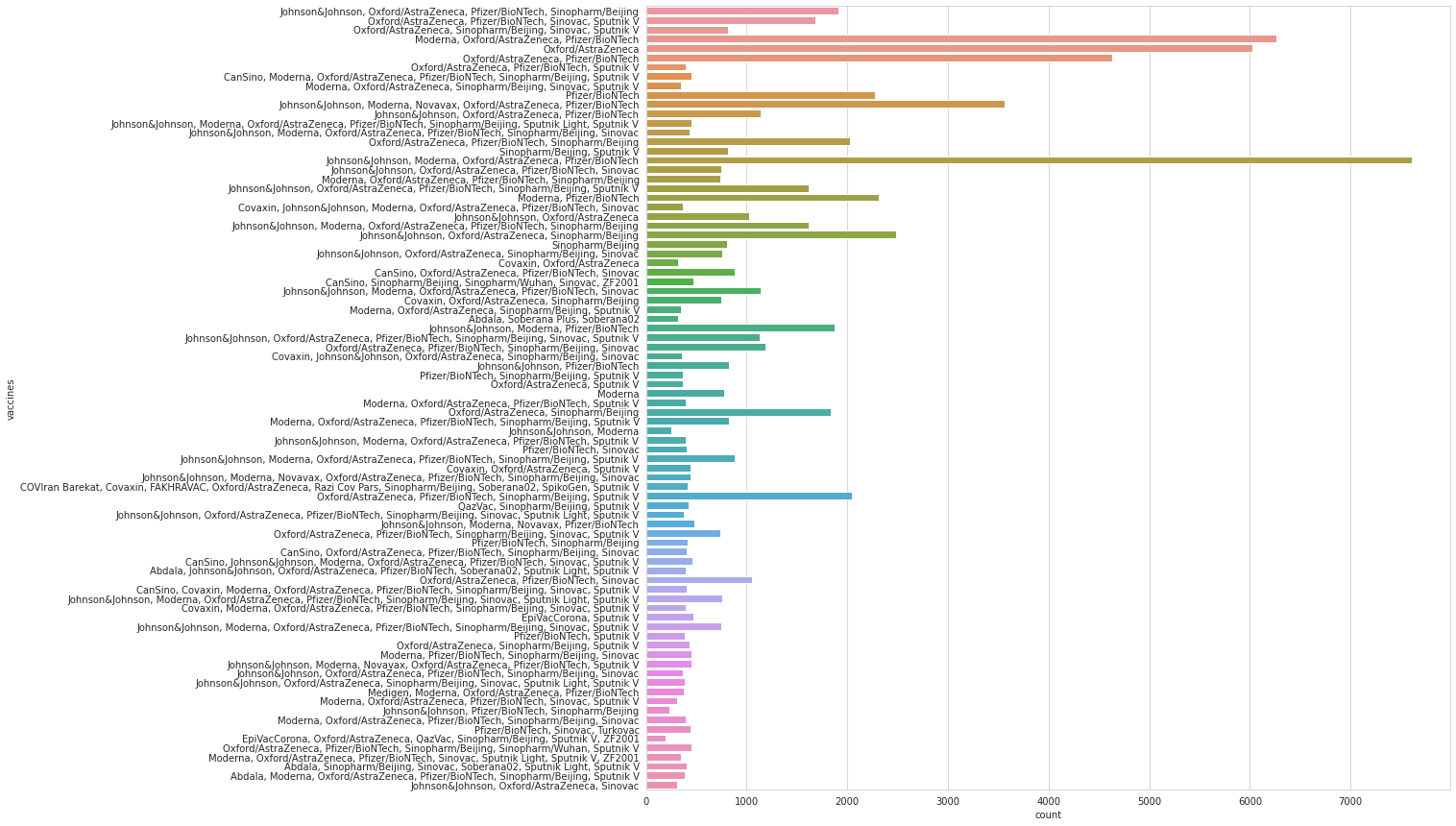


In [16]:

plt.figure(figsize=(15,15))

sns.countplot(y= "vaccines",data= df)

plt.show()



In [17]:

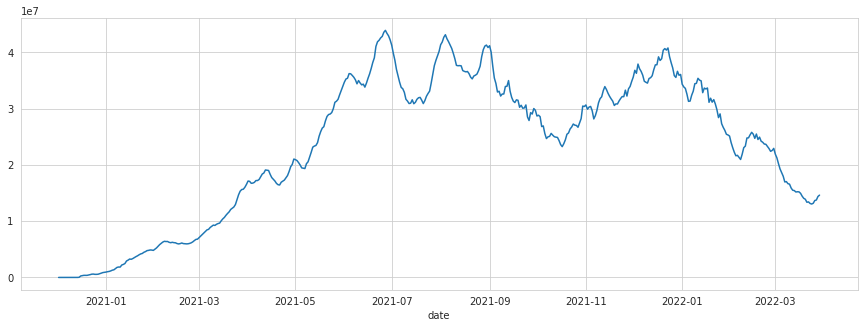
*#daily vaccinations*

x= df.groupby("date").daily\_vaccinations.sum()

plt.figure(figsize= (15,5))

sns.lineplot(x.index,x.values)

plt.show()



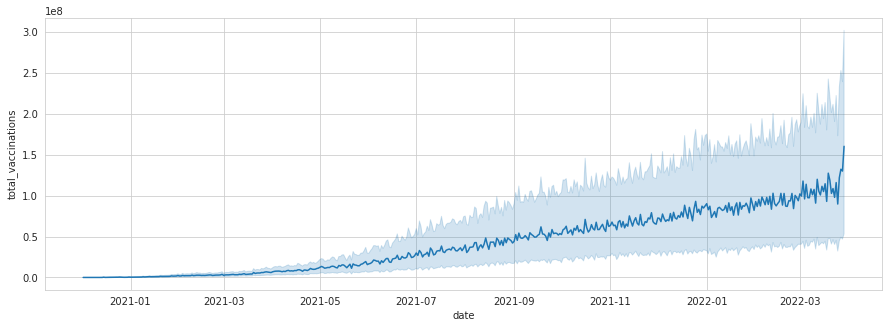
In [18]:

*#total vaccinations*

plt.figure(figsize= (15,5))

sns.lineplot(x= "date",y= "total\_vaccinations",data= df)

plt.show()



In [19]:

*#Countries with best daily average vaccinations*

x= df.groupby("country").daily\_vaccinations.mean().sort\_values(ascending= False).head(20)

x

Out[19]:

country

China 6.930368e+06

India 4.175994e+06

United States 1.191727e+06

Brazil 9.435287e+05

Indonesia 8.462893e+05

Japan 6.215795e+05

Bangladesh 5.453055e+05

Pakistan 5.430051e+05

Vietnam 5.310949e+05

Mexico 4.134253e+05

Germany 3.761575e+05

Philippines 3.665658e+05

Iran 3.535194e+05

Russia 3.480843e+05

Turkey 3.351917e+05

Thailand 3.251471e+05

United Kingdom 3.140841e+05

France 3.104963e+05

South Korea 3.042512e+05

Italy 2.970580e+05

Name: daily\_vaccinations, dtype: float64

In [20]:

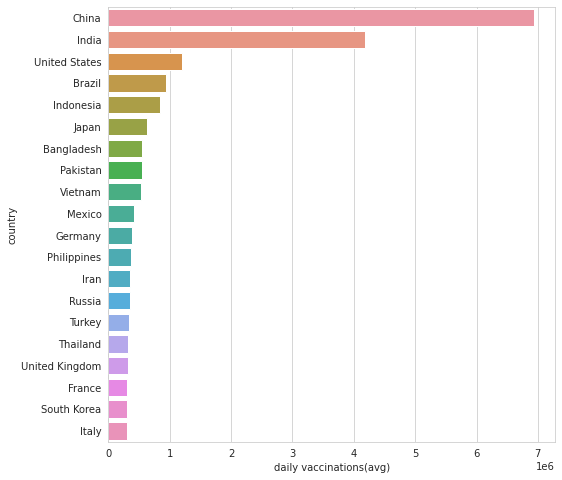
*#daily vaccinations barplot*

plt.figure(figsize= (8,8))

ax= sns.barplot(x.values,x.index)

ax.set\_xlabel("daily vaccinations(avg)")

plt.show()



In [21]:

*#vaccination per hundred top countries*

df["Total\_vaccinations\_per\_hundred"]= df.groupby("country").total\_vaccinations\_per\_hundred.tail(1)

In [22]:

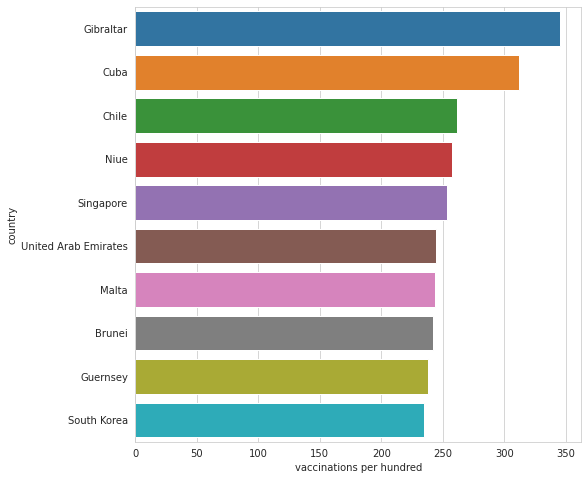
x= df.groupby("country")["Total\_vaccinations\_per\_hundred"].mean().sort\_values(ascending= False).head(10)

plt.figure(figsize= (8,8))

ax= sns.barplot(x.values,x.index)

ax.set\_xlabel("vaccinations per hundred")

plt.show()



In [23]:

*#daily vaccinations per million top countries*

df.groupby("country")["daily\_vaccinations\_per\_million"].mean().sort\_values(ascending= False).head(20)

Out[23]:

country

Falkland Islands 21185.393939

Saint Helena 13915.164835

Tokelau 12718.106195

Pitcairn 10891.797619

Niue 10109.509434

Cuba 9955.943333

Gibraltar 8000.463470

Bonaire Sint Eustatius and Saba 7412.000000

Bhutan 7241.676880

Brunei 6906.782857

Turkmenistan 6618.888889

South Korea 5930.227273

Uruguay 5829.491139

Chile 5764.154525

Singapore 5585.536424

Malta 5553.986207

Taiwan 5545.517426

Guernsey 5437.624113

Australia 5422.241895

Vietnam 5410.000000

Name: daily\_vaccinations\_per\_million, dtype: float64

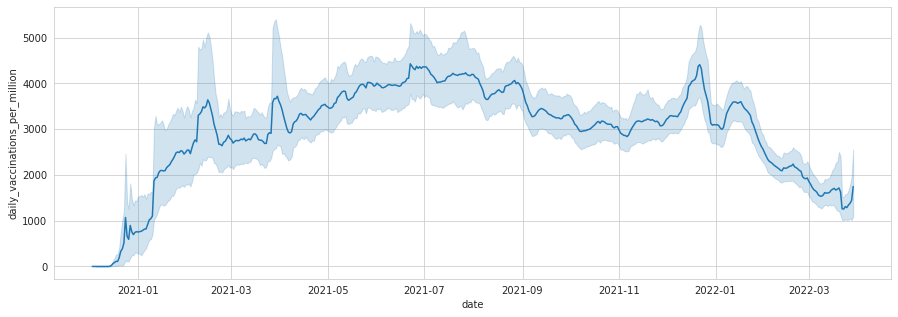
In [24]:

*#daily vaccination per million*

plt.figure(figsize= (15,5))

sns.lineplot(x= "date",y= "daily\_vaccinations\_per\_million",data= df)

plt.show()



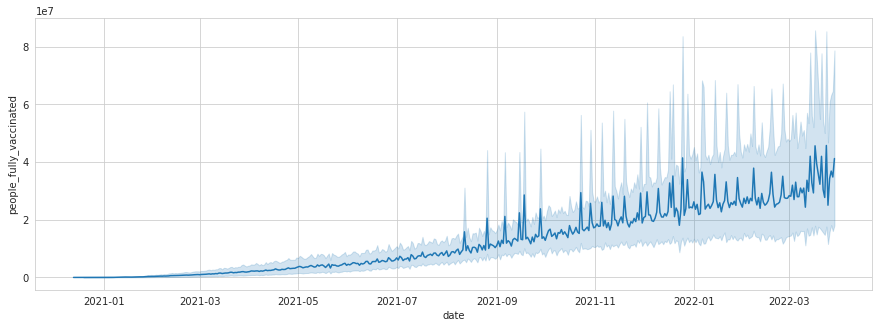
In [25]:

*#people fully vaccinated*

plt.figure(figsize= (15,5))

sns.lineplot(x= "date",y= "people\_fully\_vaccinated",data= df)

plt.show()



In [26]:

*#covid 19 vaccinations INDIA*

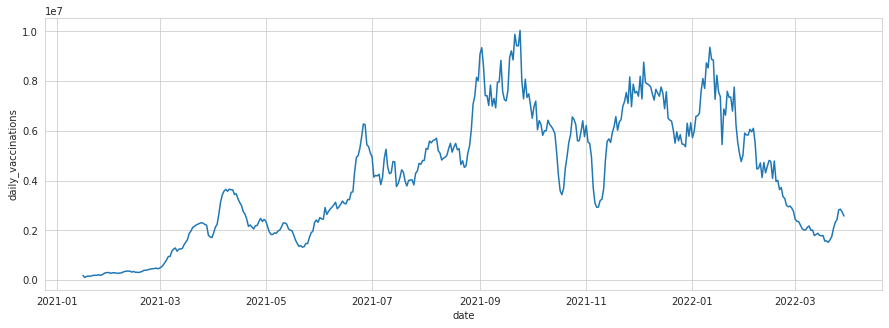
In [27]:

*#daily vaccinations in India*

plt.figure(figsize= (15,5))

sns.lineplot(x= "date",y= "daily\_vaccinations",data= df[df.country== "India"])

plt.show()



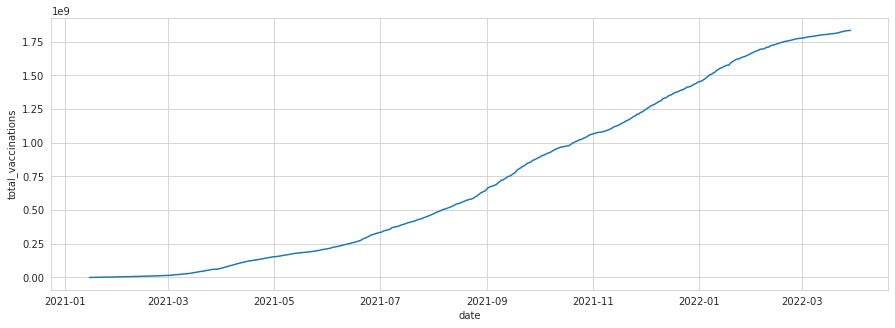
In [28]:

*#Total vaccinations in India*

plt.figure(figsize= (15,5))

sns.lineplot(x= "date",y= "total\_vaccinations",data= df[df["country"]=="India"])

plt.show()



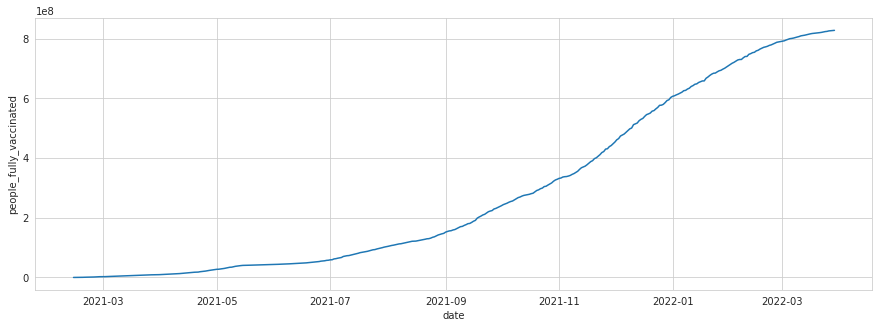
In [29]:

*#full vaccinations in India*

plt.figure(figsize= (15,5))

sns.lineplot(x= "date",y= "people\_fully\_vaccinated",data= df[df["country"]=="India"])

plt.show()



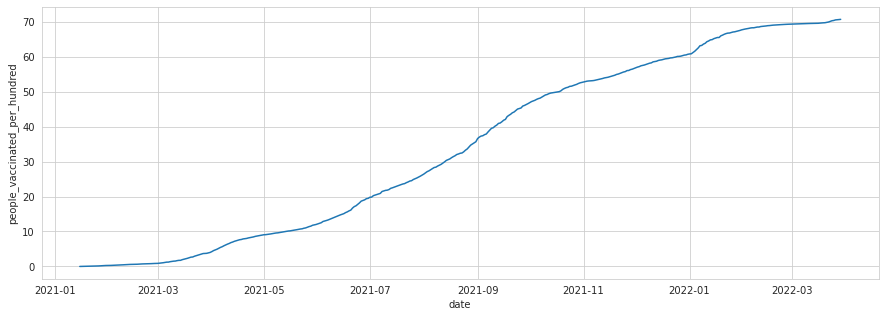
In [30]:

*#people\_vaccinated per hundred in India*

plt.figure(figsize= (15,5))

sns.lineplot(x= "date",y= "people\_vaccinated\_per\_hundred",data= df[df["country"]=="India"])

plt.show()



In [31]:

*#preferred vaccine in India*

x= df[df["country"]=="India"]

z= x.vaccines.value\_counts()

c= list(z.index)

c

Out[31]:

['Covaxin, Oxford/AstraZeneca, Sputnik V']

In [32]:

*#COMPARING TOP 5 COUNTRIES WITH MOST VACCINATIONS*

In [33]:

df.groupby("country")["Total\_vaccinations(count)"].mean().sort\_values(ascending= False).head()

Out[33]:

country

China 3.263129e+09

India 1.834501e+09

United States 5.601818e+08

Brazil 4.135596e+08

Indonesia 3.771089e+08

Name: Total\_vaccinations(count), dtype: float64

In [34]:

*#creating dataframe for top 5 vaccinated countries*

x= df.loc[(df.country== "United States") | (df.country== "China")| (df.country== "India")| (df.country== "Unted Kingdom")|(df.country== "England")]

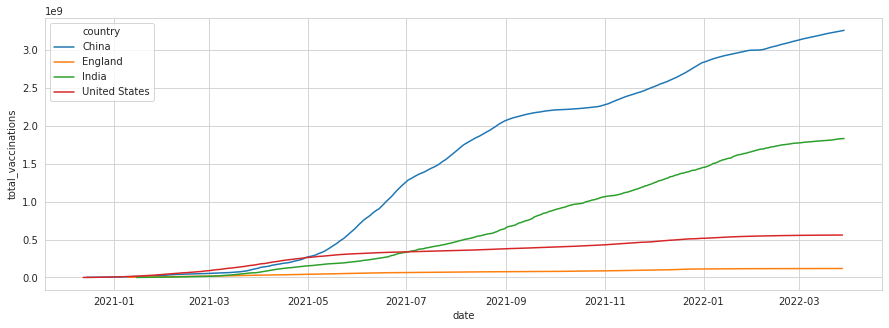
In [35]:

*#total vaccination comparison*

plt.figure(figsize= (15,5))

sns.lineplot(x= "date",y= "total\_vaccinations" ,data= x,hue= "country")

plt.show()



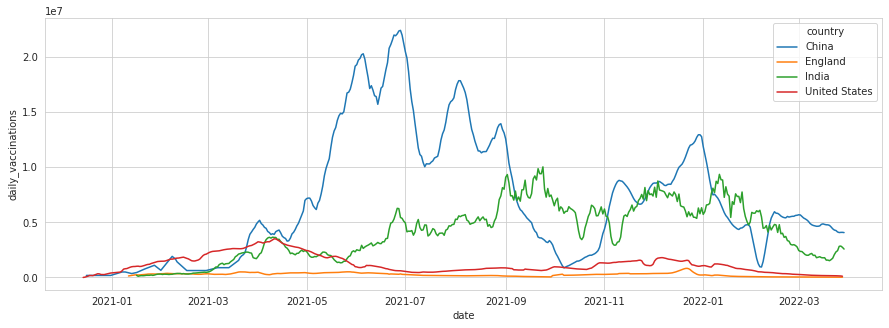
In [36]:

*#daily vaccination comparison*

plt.figure(figsize= (15,5))

sns.lineplot(x= "date",y= "daily\_vaccinations" ,data= x,hue= "country")

plt.show()



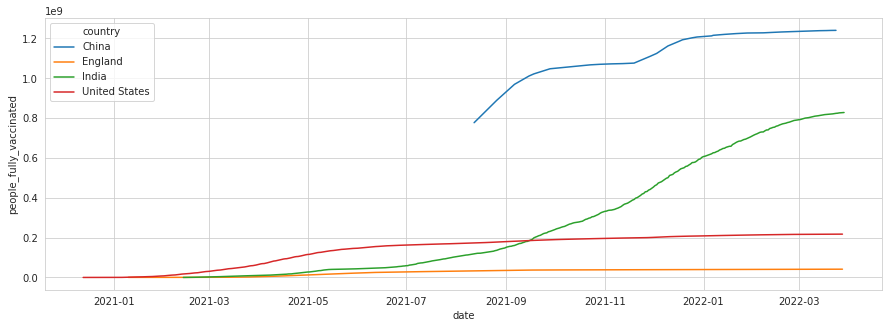
In [37]:

*#full vaccinations comparison*

plt.figure(figsize= (15,5))

sns.lineplot(x= "date",y= "people\_fully\_vaccinated" ,data= x,hue= "country")

plt.show()



In [38]:

*#daily vaccination per million comparison*

plt.figure(figsize= (15,5))

sns.lineplot(x= "date",y= "daily\_vaccinations\_per\_million" ,data= x,hue= "country")

plt.show()

