Covid-19 Vaccines Analysis

Importing Necessary Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from sklearn import metrics
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

Data Loading and Extraction

```
df = pd.read_csv('country_vaccinations.csv')
df0 = pd.read_csv('country_vaccinations.csv')
df.head()
```

```
       country iso_code        date  total_vaccinations  
people_vaccinated  \
0  Afghanistan      AFG  2021-02-22                 0.0  
0.0
1  Afghanistan      AFG  2021-02-23                 NaN  
NaN
2  Afghanistan      AFG  2021-02-24                 NaN  
NaN
3  Afghanistan      AFG  2021-02-25                 NaN  
NaN
4  Afghanistan      AFG  2021-02-26                 NaN  
NaN

   people_fully_vaccinated  daily_vaccinations_raw  daily_vaccinations 
\
0                      NaN                     NaN                 NaN

1                      NaN                     NaN              1367.0

2                      NaN                     NaN              1367.0

3                      NaN                     NaN              1367.0

4                      NaN                     NaN              1367.0
```

```
   total_vaccinations_per_hundred  people_vaccinated_per_hundred  \
0                             0.0                            0.0
1                             NaN                            NaN
2                             NaN                            NaN
3                             NaN                            NaN
4                             NaN                            NaN

   people_fully_vaccinated_per_hundred  daily_vaccinations_per_million
\
0                                  NaN                             NaN

1                                  NaN                            34.0

2                                  NaN                            34.0

3                                  NaN                            34.0

4                                  NaN                            34.0


                                            vaccines  \
0  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
1  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
2  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
3  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
4  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...

                source_name             source_website
0  World Health Organization  https://covid19.who.int/
1  World Health Organization  https://covid19.who.int/
2  World Health Organization  https://covid19.who.int/
3  World Health Organization  https://covid19.who.int/
4  World Health Organization  https://covid19.who.int/
```

```
df.isnull().sum()
```

```
country                                    0
iso_code                                   0
date                                       0
total_vaccinations                     42905
people_vaccinated                      45218
people_fully_vaccinated                47710
daily_vaccinations_raw                 51150
daily_vaccinations                       299
total_vaccinations_per_hundred         42905
people_vaccinated_per_hundred          45218
people_fully_vaccinated_per_hundred    47710
daily_vaccinations_per_million           299
vaccines                                   0
```

```
source_name                                0
source_website                             0
dtype: int64
```

```
df1 = df.copy()
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86512 entries, 0 to 86511
Data columns (total 15 columns):
 #   Column                               Non-Null Count  Dtype

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

 0   country                              86512 non-null  object

 1   iso_code                             86512 non-null  object

 2   date                                 86512 non-null
datetime64[ns]
 3   total_vaccinations                   86512 non-null  float64

 4   people_vaccinated                    86512 non-null  float64

 5   people_fully_vaccinated              86512 non-null  float64

 6   daily_vaccinations_raw               86512 non-null  float64

 7   daily_vaccinations                   86512 non-null  float64

 8   total_vaccinations_per_hundred       86512 non-null  float64

 9   people_vaccinated_per_hundred        86512 non-null  float64

 10  people_fully_vaccinated_per_hundred  86512 non-null  float64

 11  daily_vaccinations_per_million       86512 non-null  float64

 12  vaccines                             86512 non-null  object

 13  source_name                          86512 non-null  object

 14  source_website                       86512 non-null  object

dtypes: datetime64[ns](1), float64(9), object(5)
memory usage: 9.9+ MB
```

Handling Missing Values

```
df.fillna({'total_vaccinations': 0,
'people_vaccinated': 0,
'people_fully_vaccinated':0,
'daily_vaccinations_raw':0,
'daily_vaccinations':0,
'total_vaccinations_per_hundred': 0,
'people_vaccinated_per_hundred': 0,
'people_fully_vaccinated_per_hundred' :0,
'daily_vaccinations_per_million':0}, inplace=True)

df.isnull().sum()
```

```
country                                0
iso_code                               0
date                                   0
total_vaccinations                     0
people_vaccinated                      0
people_fully_vaccinated                0
daily_vaccinations_raw                 0
daily_vaccinations                     0
total_vaccinations_per_hundred         0
people_vaccinated_per_hundred          0
people_fully_vaccinated_per_hundred    0
daily_vaccinations_per_million         0
vaccines                               0
source_name                            0
source_website                         0
dtype: int64
```

```
df['date'] = pd.to_datetime(df['date'])
df
```

```
           country iso_code       date  total_vaccinations
people_vaccinated  \
0      Afghanistan      AFG 2021-02-22                 0.0
0.0
1      Afghanistan      AFG 2021-02-23                 0.0
0.0
2      Afghanistan      AFG 2021-02-24                 0.0
0.0
3      Afghanistan      AFG 2021-02-25                 0.0
0.0
4      Afghanistan      AFG 2021-02-26                 0.0
0.0
...            ...      ...        ...                 ...
...
86507     Zimbabwe      ZWE 2022-03-25           8691642.0
4814582.0
```

```
86508     Zimbabwe      ZWE 2022-03-26           8791728.0
4886242.0
86509     Zimbabwe      ZWE 2022-03-27           8845039.0
4918147.0
86510     Zimbabwe      ZWE 2022-03-28           8934360.0
4975433.0
86511     Zimbabwe      ZWE 2022-03-29           9039729.0
5053114.0

       people_fully_vaccinated  daily_vaccinations_raw
daily_vaccinations  \
0                          0.0                     0.0
0.0
1                          0.0                     0.0
1367.0
2                          0.0                     0.0
1367.0
3                          0.0                     0.0
1367.0
4                          0.0                     0.0
1367.0
...                        ...                     ...
...
86507                3473523.0                139213.0
69579.0
86508                3487962.0                100086.0
83429.0
86509                3493763.0                 53311.0
90629.0
86510                3501493.0                 89321.0
100614.0
86511                3510256.0                105369.0
103751.0

       total_vaccinations_per_hundred
people_vaccinated_per_hundred  \
0                                0.00                          0.00

1                                0.00                          0.00

2                                0.00                          0.00

3                                0.00                          0.00

4                                0.00                          0.00

...                               ...                           ...

86507                           57.59                         31.90
```

```
86508                                    58.25                              32.38

86509                                    58.61                              32.59

86510                                    59.20                              32.97

86511                                    59.90                              33.48


       people_fully_vaccinated_per_hundred
daily_vaccinations_per_million  \
0                                     0.00
0.0
1                                     0.00
34.0
2                                     0.00
34.0
3                                     0.00
34.0
4                                     0.00
34.0
...                                    ...
...
86507                                23.02
4610.0
86508                                23.11
5528.0
86509                                23.15
6005.0
86510                                23.20
6667.0
86511                                23.26
6874.0

                                                vaccines  \
0      Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
1      Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
2      Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
3      Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
4      Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
...                                                  ...
86507  Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
86508  Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
86509  Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
86510  Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
86511  Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...

                     source_name  \
0      World Health Organization
1      World Health Organization
```

```
2      World Health Organization
3      World Health Organization
4      World Health Organization
...                          ...
86507         Ministry of Health
86508         Ministry of Health
86509         Ministry of Health
86510         Ministry of Health
86511         Ministry of Health

                                          source_website
0                                https://covid19.who.int/
1                                https://covid19.who.int/
2                                https://covid19.who.int/
3                                https://covid19.who.int/
4                                https://covid19.who.int/
...                                                  ...
86507  https://www.arcgis.com/home/webmap/viewer.html...
86508  https://www.arcgis.com/home/webmap/viewer.html...
86509  https://www.arcgis.com/home/webmap/viewer.html...
86510  https://www.arcgis.com/home/webmap/viewer.html...
86511  https://www.arcgis.com/home/webmap/viewer.html...

[86512 rows x 15 columns]

#Checking For Duplicate values
df.drop_duplicates(inplace=True)
df[df.duplicated()]

Empty DataFrame
Columns: [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]
Index: []
```

All the Duplicates values are dropped.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86512 entries, 0 to 86511
Data columns (total 15 columns):
 #   Column                               Non-Null Count  Dtype

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

 0   country                              86512 non-null  object
```

```
 1   iso_code                               86512 non-null  object

 2   date                                   86512 non-null
datetime64[ns]
 3   total_vaccinations                     86512 non-null  float64

 4   people_vaccinated                      86512 non-null  float64

 5   people_fully_vaccinated                86512 non-null  float64

 6   daily_vaccinations_raw                 86512 non-null  float64

 7   daily_vaccinations                     86512 non-null  float64

 8   total_vaccinations_per_hundred         86512 non-null  float64

 9   people_vaccinated_per_hundred          86512 non-null  float64

 10  people_fully_vaccinated_per_hundred    86512 non-null  float64

 11  daily_vaccinations_per_million         86512 non-null  float64

 12  vaccines                               86512 non-null  object

 13  source_name                            86512 non-null  object

 14  source_website                         86512 non-null  object

dtypes: datetime64[ns](1), float64(9), object(5)
memory usage: 9.9+ MB

#Dropping unwanted
df1 = df1.drop((['vaccines','source_name','source_website']), axis=1)
df1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86512 entries, 0 to 86511
Data columns (total 12 columns):
 #   Column                                 Non-Null Count  Dtype

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

 0   country                                86512 non-null  object

 1   iso_code                               86512 non-null  object

 2   date                                   86512 non-null
datetime64[ns]
 3   total_vaccinations                     86512 non-null  float64

 4   people_vaccinated                      86512 non-null  float64
```

```
 5   people_fully_vaccinated                86512 non-null   float64
 6   daily_vaccinations_raw                 86512 non-null   float64
 7   daily_vaccinations                     86512 non-null   float64
 8   total_vaccinations_per_hundred         86512 non-null   float64
 9   people_vaccinated_per_hundred          86512 non-null   float64
 10  people_fully_vaccinated_per_hundred    86512 non-null   float64
 11  daily_vaccinations_per_million         86512 non-null   float64
dtypes: datetime64[ns](1), float64(9), object(2)
memory usage: 7.9+ MB
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df2['country']=le.fit_transform(df2['country'])
df2['iso_code']=le.fit_transform(df2['iso_code'])

df2['vaccines']=le.fit_transform(df2['vaccines'])

df2['source_name']=le.fit_transform(df2['source_name'])

df2['source_website']=le.fit_transform(df2['source_website'])
# df2['date'] = df2['date'].str.replace('-', ' ')
```

Machine Learning in Python

Testing and Training

```
x=df2[['country', 'iso_code','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']]
y=df2[['total_vaccinations']]

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,y,
```

```
random_state=42)
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((64884, 13), (21628, 13), (64884, 1), (21628, 1))
```

Linear Regression Model

```
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
LR.fit(x_train, y_train)
y_pred_LR=LR.predict(x_test)
y_pred_LR
```

```
array([[-1.59766070e+07],
       [-9.63050813e+06],
       [-3.70713648e+07],
       ...,
       [ 1.09876415e+08],
       [-1.43512829e+07],
       [-2.66557595e+06]])
```

```
# Model Evaluation
print('R^2:',metrics.r2_score(y_test, y_pred_LR))
print('MAE:',metrics.mean_absolute_error(y_test, y_pred_LR))
print('MSE:',metrics.mean_squared_error(y_test, y_pred_LR))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, y_pred_LR)))
```

```
R^2: 0.6376719832015039
MAE: 22774656.10221463
MSE: 1.0086802875248108e+16
RMSE: 100433076.59953521
```

R^2 : It is a measure of the linear relationship between X and Y. It is interpreted as the proportion of the variance in the dependent variable that is predictable from the independent variable.

Adjusted R^2 :The adjusted R-squared compares the explanatory power of regression models that contain different numbers of predictors.

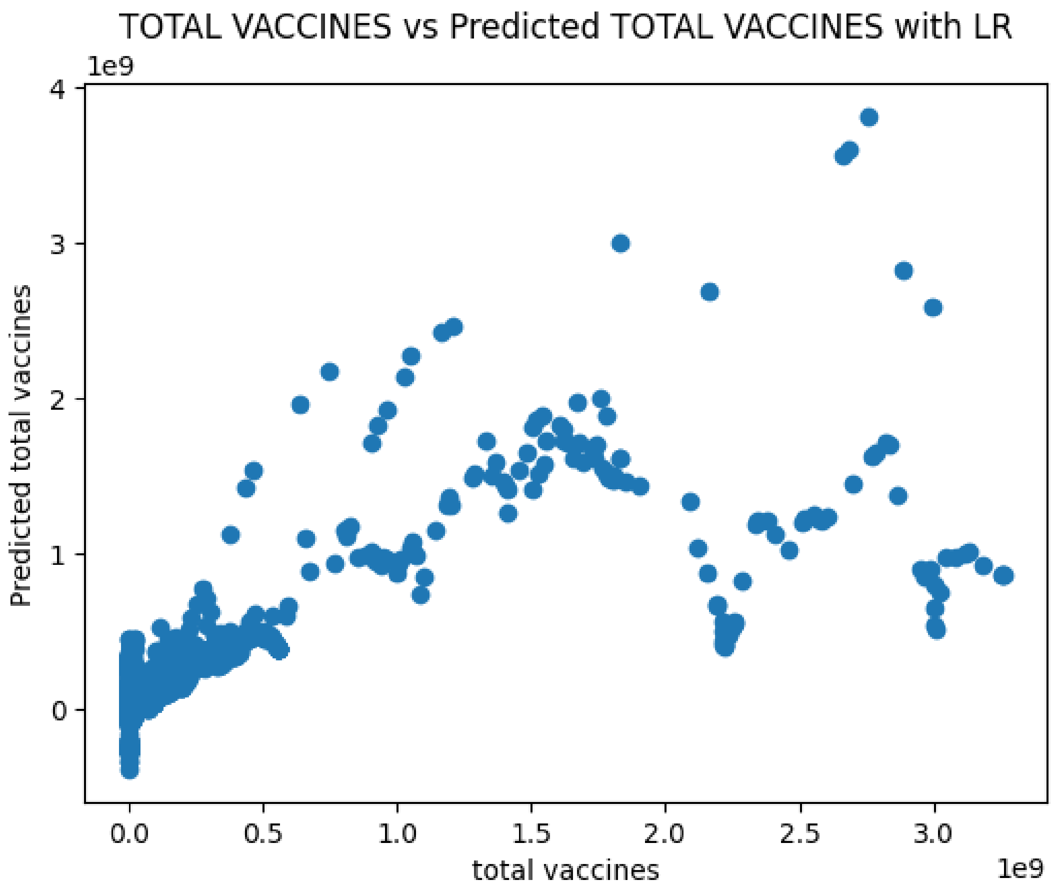
MAE : It is the mean of the absolute value of the errors. It measures the difference between two continuous variables, here actual and predicted values of y.

MSE: The mean square error (MSE) is just like the MAE, but squares the difference before summing them all instead of using the absolute value.

RMSE: The mean square error (MSE) is just like the MAE, but squares the difference before summing them all instead of using the absolute value.

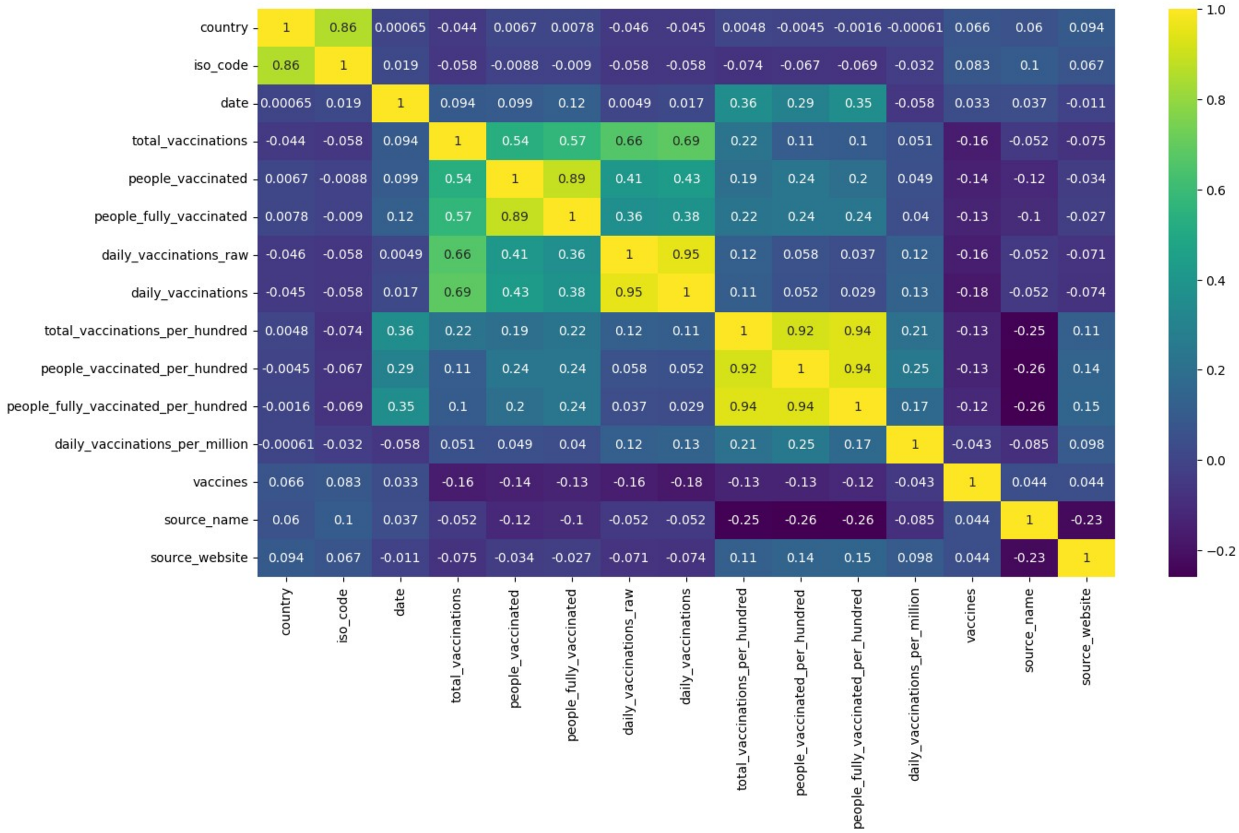
```
plt.scatter(y_test, y_pred_LR)
plt.xlabel("total vaccines")
```

```
plt.ylabel("Predicted total vaccines")
plt.title("TOTAL VACCINES vs Predicted TOTAL VACCINES with LR")
plt.show()
```




```
corr = df2.corr()
plt.figure(figsize=(15,8))
sns.heatmap(corr, cmap='viridis', annot=True)
```

```
<Axes: >
```



Exploratory Data Analysis

```
df0.rename(columns =
{'total_vaccinations_per_hundred':'total_vaccinations_percent',

'people_fully_vaccinated_per_hundred':'people_fully_vaccinated_percent
',

'people_vaccinated_per_hundred':'people_vaccinated_percent'},
inplace=True)
df0.columns
```

```
Index(['country', 'iso_code', 'date', 'total_vaccinations',
       'people_vaccinated', 'people_fully_vaccinated',
       'daily_vaccinations_raw', 'daily_vaccinations',
       'total_vaccinations_percent', 'people_vaccinated_percent',
       'people_fully_vaccinated_percent',
'daily_vaccinations_per_million',
       'vaccines', 'source_name', 'source_website'],
      dtype='object')
```

Basic info about Dataset

```python
print('Data point starts from:',df0.date.min(),'\n')
print('Data point ends at:',df0.date.max(),'\n')
print('Total no of Countries in the data
set:',len(df0.country.unique()),'\n')
print('Total no of unique Vaccine Schemes in the data
set:',len(df0.vaccines.unique()),'\n')
```

```
Data point starts from: 2020-12-02

Data point ends at: 2022-03-29

Total no of Countries in the data set: 223

Total no of unique Vaccine Schemes in the data set: 84
```

```python
df0.country.unique()
```

```
array(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
       'Anguilla', 'Antigua and Barbuda', 'Argentina', 'Armenia',
'Aruba',
       'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
       'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize',
'Benin',
       'Bermuda', 'Bhutan', 'Bolivia', 'Bonaire Sint Eustatius and
Saba',
       'Bosnia and Herzegovina', 'Botswana', 'Brazil',
       'British Virgin Islands', 'Brunei', 'Bulgaria', 'Burkina Faso',
       'Burundi', 'Cambodia', 'Cameroon', 'Canada', 'Cape Verde',
       'Cayman Islands', 'Central African Republic', 'Chad', 'Chile',
       'China', 'Colombia', 'Comoros', 'Congo', 'Cook Islands',
       'Costa Rica', "Cote d'Ivoire", 'Croatia', 'Cuba', 'Curacao',
       'Cyprus', 'Czechia', 'Democratic Republic of Congo', 'Denmark',
       'Djibouti', 'Dominica', 'Dominican Republic', 'Ecuador',
'Egypt',
       'El Salvador', 'England', 'Equatorial Guinea', 'Estonia',
       'Eswatini', 'Ethiopia', 'Faeroe Islands', 'Falkland Islands',
       'Fiji', 'Finland', 'France', 'French Polynesia', 'Gabon',
'Gambia',
       'Georgia', 'Germany', 'Ghana', 'Gibraltar', 'Greece',
'Greenland',
       'Grenada', 'Guatemala', 'Guernsey', 'Guinea', 'Guinea-Bissau',
       'Guyana', 'Haiti', 'Honduras', 'Hong Kong', 'Hungary',
'Iceland',
       'India', 'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Isle of Man',
       'Israel', 'Italy', 'Jamaica', 'Japan', 'Jersey', 'Jordan',
       'Kazakhstan', 'Kenya', 'Kiribati', 'Kosovo', 'Kuwait',
       'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon', 'Lesotho',
'Liberia',
       'Libya', 'Liechtenstein', 'Lithuania', 'Luxembourg', 'Macao',
```

```
       'Madagascar', 'Malawi', 'Malaysia', 'Maldives', 'Mali',
'Malta',
       'Mauritania', 'Mauritius', 'Mexico', 'Moldova', 'Monaco',
       'Mongolia', 'Montenegro', 'Montserrat', 'Morocco',
'Mozambique',
       'Myanmar', 'Namibia', 'Nauru', 'Nepal', 'Netherlands',
       'New Caledonia', 'New Zealand', 'Nicaragua', 'Niger',
'Nigeria',
       'Niue', 'North Macedonia', 'Northern Cyprus', 'Northern
Ireland',
       'Norway', 'Oman', 'Pakistan', 'Palestine', 'Panama',
       'Papua New Guinea', 'Paraguay', 'Peru', 'Philippines',
'Pitcairn',
       'Poland', 'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda',
       'Saint Helena', 'Saint Kitts and Nevis', 'Saint Lucia',
       'Saint Vincent and the Grenadines', 'Samoa', 'San Marino',
       'Sao Tome and Principe', 'Saudi Arabia', 'Scotland', 'Senegal',
       'Serbia', 'Seychelles', 'Sierra Leone', 'Singapore',
       'Sint Maarten (Dutch part)', 'Slovakia', 'Slovenia',
       'Solomon Islands', 'Somalia', 'South Africa', 'South Korea',
       'South Sudan', 'Spain', 'Sri Lanka', 'Sudan', 'Suriname',
'Sweden',
       'Switzerland', 'Syria', 'Taiwan', 'Tajikistan', 'Tanzania',
       'Thailand', 'Timor', 'Togo', 'Tokelau', 'Tonga',
       'Trinidad and Tobago', 'Tunisia', 'Turkey', 'Turkmenistan',
       'Turks and Caicos Islands', 'Tuvalu', 'Uganda', 'Ukraine',
       'United Arab Emirates', 'United Kingdom', 'United States',
       'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela', 'Vietnam',
       'Wales', 'Wallis and Futuna', 'Yemen', 'Zambia', 'Zimbabwe'],
      dtype=object)

# All the different kinds of vaccines
df0.vaccines.unique()

array(['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,
```

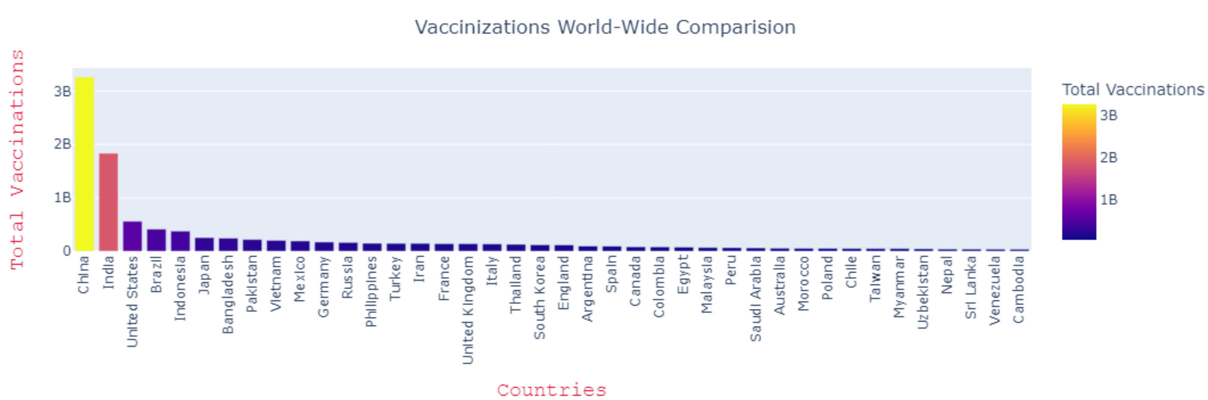
```
Sinopharm/Beijing, Sputnik Light, Sputnik V',
       'Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech,
Sinopharm/Beijing, Sinovac',
       'Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing',
       'Sinopharm/Beijing, Sputnik V',
       'Johnson&Johnson, Moderna, Oxford/AstraZeneca,
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'], dtype=object)

# Here we are creating `country_data` which store basic info about a
country, like the vaccine scheme it uses, total
# vaccinations completed and its percentage with the population

country_data = df0.copy()
cols = ['country', 'total_vaccinations', 'iso_code', 'vaccines',
'total_vaccinations_percent']

country_data =
country_data[cols].groupby('country').max().sort_values('total_vaccina
tions', ascending=False)
country_data.reset_index(inplace = True)

country_data.columns = ['Country', 'Total Vaccinations', 'iso_code',
'Vaccines', 'Total Vaccinations Percentage']
country_data

              Country  Total Vaccinations iso_code  \
0               China        3.263129e+09      CHN
1               India        1.834501e+09      IND
2       United States        5.601818e+08      USA
3              Brazil        4.135596e+08      BRA
4           Indonesia        3.771089e+08      IDN
..                ...                 ...      ...
218  Falkland Islands        4.407000e+03      FLK
219        Montserrat        4.211000e+03      MSR
220              Niue        4.161000e+03      NIU
221           Tokelau        1.936000e+03      TKL
222          Pitcairn        9.400000e+01      PCN

                                              Vaccines  \
0    CanSino, Sinopharm/Beijing, Sinopharm/Wuhan, S...
1                Covaxin, Oxford/AstraZeneca, Sputnik V
2             Johnson&Johnson, Moderna, Pfizer/BioNTech
3    Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
4    Johnson&Johnson, Moderna, Novavax, Oxford/Astr...
..                                                 ...
218                                 Oxford/AstraZeneca
219                                 Oxford/AstraZeneca
```

```
220                                             Pfizer/BioNTech
221                                             Pfizer/BioNTech
222                                         Oxford/AstraZeneca

     Total Vaccinations Percentage
0                           225.94
1                           131.66
2                           168.72
3                           193.26
4                           136.45
..                             ...
218                         124.91
219                          84.54
220                         257.81
221                         141.52
222                         200.00

[223 rows x 5 columns]
```

```
fig = px.bar(country_data[:40], x = 'Country', y = 'Total
Vaccinations', color = 'Total Vaccinations')

fig.update_layout(title = dict(text = 'Vaccinizations World-Wide
Comparision', x=0.5, y=0.95))
fig.update_xaxes(title = 'Countries', title_font = dict(size=18,
family='Courier', color='crimson'), tickangle=-90)
fig.update_yaxes(title = 'Total Vaccinations', title_font =
dict(size=18, family='Courier', color='crimson'))

fig.show()
```



From the plot, some interesting facts stand out:

• The United States, despite having the highest number of people affected by Covid-19, has the highest number of vaccinated people.

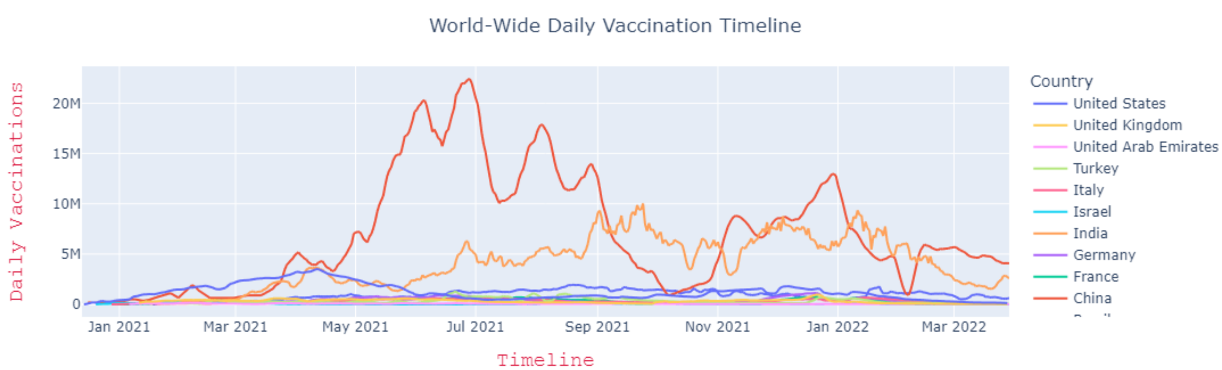
• China, from where the virus started spreading, is at second.

• India, who has been supplying vaccines to the world is at 3th position.

• UK, where we have found a new variant strain of the virus, is right next.

• Following that, we have Israel, UAE, Brazil, Germany and others

```
top_countries =
['USA','CHN','GBR','IND','ISR','ARE','BRA','DEU','TUR','ITA','FRA']
fig = px.line(df0[df0.iso_code.isin(top_countries)], x='date',
y='daily_vaccinations', color='country')

fig.update_layout(title = dict(text = 'World-Wide Daily Vaccination
Timeline', x=0.5, y=0.95),
                  legend = dict(title = 'Country', traceorder =
'reversed'))
fig.update_xaxes(title = 'Timeline', title_font = dict(size=18,
family='Courier', color='crimson'))
fig.update_yaxes(title = 'Daily Vaccinations', title_font =
dict(size=18, family='Courier', color='crimson'))

fig.show()
#Country wise daily vaccination
```



From the plot, we can deduce:

• The Line plot for China is composed entirely of straight lines. This can be attributed to the CCP which tries to restrict flow of information in and out of China. Thus, information from China usually comes in intervals and can be taken with a grain of salt.

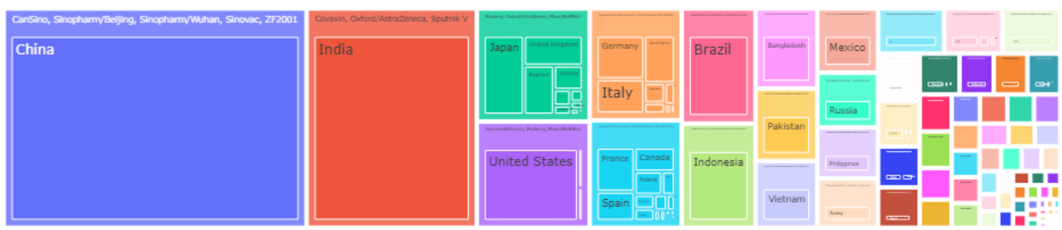
• Comparatively, the plot of vaccinations in the USA is better plotted. We can also see that while the USA was heavily affected by the virus, its vaccination drive is highly effective.

• Others like the UK have a steady increase in Daily Vaccinations and India, while supplying to many countries, maintains a respectable 3th position.

```
fig = px.treemap(country_data, path = ['Vaccines', 'Country'], values
= 'Total Vaccinations', height = 650,
                 custom_data = ['Country', 'Vaccines', 'Total
```

```
Vaccinations'])

fig.update_layout(title = dict(text = 'Total vaccinations per country, grouped by Vaccine Scheme', x=0.5, y=0.95))
fig.update_traces(hovertemplate = 'Country: %{customdata[0]}<br>Vaccine: %{customdata[1]}<br>Total Vaccinations: %{customdata[2]}')
fig.show()
```

Total vaccinations per country, grouped by Vaccine Scheme



• From the above Treemap we can realise that a Bar and Pie Plot may often only show a part of the information that can be observed, whereas a Treemap can accurately show the share of a particular vaccine world-wide, the countries that are using the said vaccine and can even show comparisons between all the countries.

• As the Treemap shows so much information at a time, it can help one understand the data much more accurately.

```
fig = px.choropleth(country_data, locations = 'Country', color = 'Total Vaccinations',
                    locationmode = 'country names', color_continuous_scale = 'rainbow',
                    hover_name = 'Country', projection = 'natural earth')

fig.update_layout(title = dict(text = 'Total Vaccinations in every Country', x=0.5, y=0.95),
                 geo = dict(showocean = True, oceancolor = "#7af8ff", showland = True,
                            landcolor = "white",showlakes = False, showframe = False))

fig.show()
```

• In the above visualisation, we can see the countries and the total vaccinations they have completed.

Statistical Analysis

Which countries started vaccinations first?

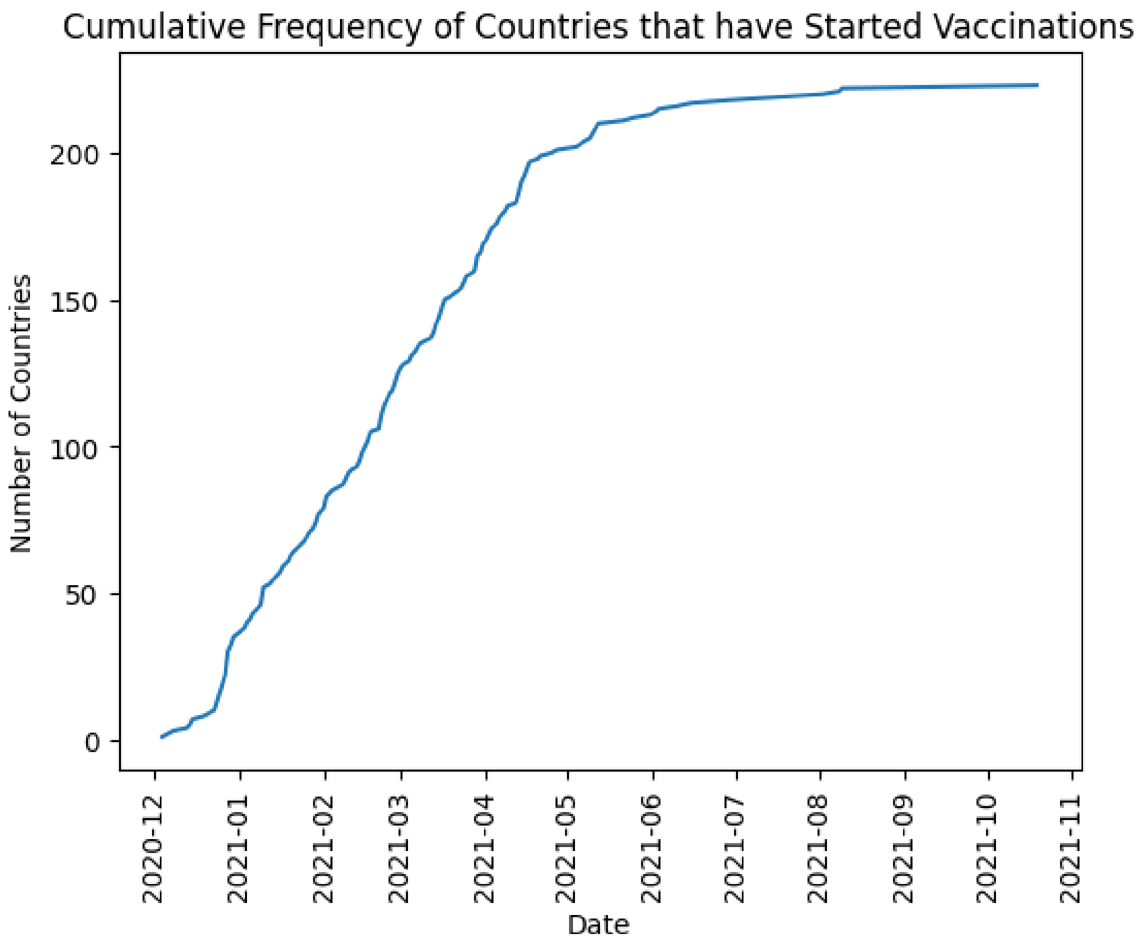
```
# Find out which countries started vaccinations earliest
df0['date'] = pd.to_datetime(df0['date'], utc=True)
vacc_start = df0.loc[df0[df0.total_vaccinations >
0].groupby('country')['date'].idxmin()].sort_values('date')
vacc_start.head(5)
```

```
             country iso_code                      date
total_vaccinations  \
43117         Latvia      LVA 2020-12-04 00:00:00+00:00
1.0
58523         Norway      NOR 2020-12-08 00:00:00+00:00
5.0
20826        Denmark      DNK 2020-12-08 00:00:00+00:00
1.0
82360  United States      USA 2020-12-13 00:00:00+00:00
30288.0
13403         Canada      CAN 2020-12-14 00:00:00+00:00
5.0

       people_vaccinated  people_fully_vaccinated
daily_vaccinations_raw  \
43117                1.0                      NaN
NaN
58523                5.0                      NaN
5.0
20826                1.0                      NaN
NaN
82360            25125.0                   5897.0
```

```
NaN
13403                5.0                      NaN
NaN

       daily_vaccinations  total_vaccinations_percent  \
43117                 NaN                        0.00
58523                 1.0                        0.00
20826                 NaN                        0.00
82360                 NaN                        0.01
13403                 NaN                        0.00

       people_vaccinated_percent  people_fully_vaccinated_percent  \
43117                       0.00                              NaN
58523                       0.00                              NaN
20826                       0.00                              NaN
82360                       0.01                              0.0
13403                       0.00                              NaN

       daily_vaccinations_per_million  \
43117                             NaN
58523                             0.0
20826                             NaN
82360                             NaN
13403                             NaN

                                                vaccines  \
43117  Johnson&Johnson, Moderna, Novavax, Pfizer/BioN...
58523                          Moderna, Pfizer/BioNTech
20826          Johnson&Johnson, Moderna, Pfizer/BioNTech
82360          Johnson&Johnson, Moderna, Pfizer/BioNTech
13403  Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...

                                             source_name  \
43117                            National Health Service
58523                Norwegian Institute of Public Health
20826                            Statens Serum Institute
82360          Centers for Disease Control and Prevention
13403  Official data from provinces via covid19tracke...

                                          source_website
43117  https://data.gov.lv/dati/eng/dataset/covid19-v...
58523  https://github.com/folkehelseinstituttet/surve...
20826  https://covid19.ssi.dk/overvagningsdata/downlo...
82360  https://data.cdc.gov/Vaccinations/COVID-19-Vac...
13403  https://covid19tracker.ca/vaccinationtracker.html
```

How have the cumulative number of countries adopting covid-19 vaccinations evolved over time? How is this trend?

```
# Cumulative distribution of vaccination start dates
events = pd.Series(vacc_start.date.value_counts())
events.index = pd.to_datetime(events.index)
events.sort_index(inplace=True)

plt.plot(events.cumsum())
plt.xticks(rotation=90)
plt.title('Cumulative Frequency of Countries that have Started
Vaccinations')
plt.xlabel('Date')
plt.ylabel('Number of Countries')
plt.show()
```



What are the top 20 countries in terms of total number of vaccines administered?

```
vacc_total = df0.loc[df0.groupby('country')
['total_vaccinations'].idxmax()].sort_values('total_vaccinations',asce
nding=False)
vacc_total = pd.concat((vacc_total,vacc_total["vaccines"].str.split(",
", expand = True)),axis=1)
```

```
# Plot out which countries have performed most vaccinations in
descending order
plt.figure(figsize=(10, 4))
plt.bar(vacc_total.country[0:20], vacc_total.total_vaccinations[0:20])
plt.xticks(rotation=90)
plt.title('Top 20 Countries by Total Vaccinations')
plt.xlabel('Country')
plt.ylabel('Total Vaccinations (x 10Million)')
plt.show()
```