COVID-19 VACCINES ANALYSIS PHASE 4

Member name: M.Gokula Kannan

Code:au723721205018

INTRODUCTION:

Exploratory data analysis (EDA) serves as a valuable method employed by data experts to gain insights into a dataset prior to embarking on modeling. EDA is sometimes colloquially known as data exploration. Its primary objective is to unveil the dataset's underlying attributes and patterns, aiding data analysts in forming predictions and making assumptions about the data. EDA frequently incorporates visualizations such as histograms, scatter plots, and box plots to facilitate this process.

Key Terms:

Value: A data value is a piece of information, such as a number or a date.

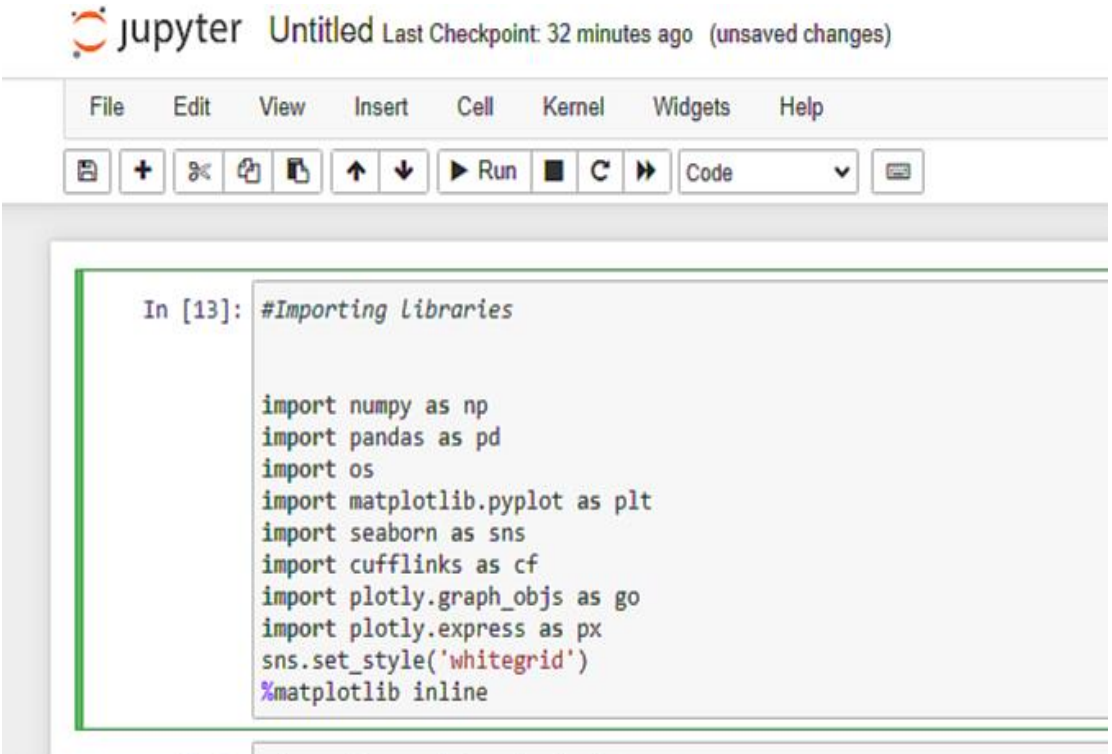
Variable: A data variable is a characteristic that you can measure, such as weight or income.

Distribution: The distribution of a dataset is how the dataset is spread out. You can visualize a dataset's distribution by observing its shape on a graph.

Outlier: An outlier is a data value that is significantly different, including much higher or lower, from the rest of a dataset.

Data model: A data model is a method of organizing data and relationships between values in a dataset.

Importing Necessary Libraries:



These libraries are requires for performing necessary operations and plots for analysis purpose.

Find the shape of your dataset:

Finding the shape of your dataset is another important step in the EDA process. This step is important because you can gather relevant information about your dataset by observing its shape. The shape of your dataset shows your data's distribution. You can also notice data features like skewness and gaps that can help you learn more about the dataset. It can also help you identify trends in your dataset.

```
In [17]: #importing covid analysis dataset

data= pd.read_csv("country_vaccinations.csv")

print("Loading Dataset.....")

print("dataset:%s"%(str(data.shape)))

Loading Dataset.....
dataset:(86512, 15)
```

Observe the dataset:

Commence your exploratory data analysis by taking a high-level overview of your dataset. Your first task should be to understand the dataset's size, which involves identifying the number of rows and columns it contains. This initial step can provide early insights into potential data-related issues that could emerge down the road.

The dataset has a total of 15 columns and 86512 records . Once the dataset is loaded we will

take a look at the dataset entries. head() is used to get the starting records from the dataset and tail() is used to get the ending records from the dataset. We can also specify how much records are to be displayed .

```
In [19]: data.head(6)
Out[19]:
```

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per
0	Afghanistan	AFG	2021-02-22	0.0	0.0	NaN	NaN	NaN	
1	Afghanistan	AFG	2021-02-23	NaN	NaN	NaN	NaN	1367.0	
2	Afghanistan	AFG	2021-02-24	NaN	NaN	NaN	NaN	1367.0	
3	Afghanistan	AFG	2021-02-25	NaN	NaN	NaN	NaN	1367.0	
4	Afghanistan	AFG	2021-02-26	NaN	NaN	NaN	NaN	1367.0	
5	Afghanistan	AFG	2021-02-27	NaN	NaN	NaN	NaN	1367.0	

In the above our dataset is defined as 'data' and data.head(5) has returned the top 5 records.

In [20]:
```
data.tail(6)
```

Out[20]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_
86506	Zimbabwe	ZWE	2022-03-24	8552429.0	4704720.0	3461926.0	137952.0	51151.0	
86507	Zimbabwe	ZWE	2022-03-25	8691642.0	4814582.0	3473523.0	139213.0	69579.0	
86508	Zimbabwe	ZWE	2022-03-26	8791728.0	4886242.0	3487962.0	100086.0	83429.0	
86509	Zimbabwe	ZWE	2022-03-27	8845039.0	4918147.0	3493763.0	53311.0	90629.0	
86510	Zimbabwe	ZWE	2022-03-28	8934360.0	4975433.0	3501493.0	89321.0	100614.0	
86511	Zimbabwe	ZWE	2022-03-29	9039729.0	5053114.0	3510256.0	105369.0	103751.0	

In the above figure data.tail(5) has returned the last 5 records.

Chech for null values :

After your initial dataset inspection, your next step is to detect and address any missing values. When you come across these gaps in the data, consider the underlying causes for their absence. Analyzing patterns within your dataset may enable you to make informed estimations and potentially replace some of the missing values.

- Stop Word Removal: Eliminate common, non-informative words.

- Lemmatization or Stemming: Reduce words to their base form.

```
In [21]: #checkingnull values

         data.isnull().sum(axis=0)

Out[21]: 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
```

isnull() returns true or false in presence or absence of NaN values in the records. Using sum() we can get the total number of null values in the records.

Removing null values:

```
In [23]: #Removing null values for totvacc and peopvacc
         data.dropna(subset=["total_vaccinations", "people_vaccinated"], how="all", inplace=True)
         data.head()

Out[23]:
```

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccin
0	Afghanistan	AFG	2021-02-22	0.0	0.0	NaN	
6	Afghanistan	AFG	2021-02-28	8200.0	8200.0	NaN	
22	Afghanistan	AFG	2021-03-16	54000.0	54000.0	NaN	
44	Afghanistan	AFG	2021-04-07	120000.0	120000.0	NaN	
59	Afghanistan	AFG	2021-04-22	240000.0	240000.0	NaN	

dropna() is used to remove NaN values from columns mentioned. Here we have removed

null values from columns 'total_vaccinations' and 'people_vaccinated'.

Removing unecessary columns:

In [26]:
```
#Dropping redundant columns

data.drop(['daily_vaccinations_raw','source_website'],axis=1)
```

Out[26]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_
0	Afghanistan	AFG	2021-02-22	0.0	0.0	NaN	
6	Afghanistan	AFG	2021-02-28	8200.0	8200.0	NaN	
22	Afghanistan	AFG	2021-03-16	54000.0	54000.0	NaN	
44	Afghanistan	AFG	2021-04-07	120000.0	120000.0	NaN	
59	Afghanistan	AFG	2021-04-22	240000.0	240000.0	NaN	

...

Since there is no use of 'daily_vaccinations_raw' and 'source_website' we have dropped these

columns using drop() by defining axis=1 which denotes the columns.

Displaying vaccines used by countries:

In [29]:
```
#vaccines used by country
vacc_country=data[['country','iso_code','vaccines']]
vacc_country
```

Out[29]:

	country	iso_code	vaccines
0	Afghanistan	AFG	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
6	Afghanistan	AFG	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
22	Afghanistan	AFG	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
44	Afghanistan	AFG	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
59	Afghanistan	AFG	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
...	...	...	...
86507	Zimbabwe	ZWE	Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
86508	Zimbabwe	ZWE	Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
86509	Zimbabwe	ZWE	Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
86510	Zimbabwe	ZWE	Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
86511	Zimbabwe	ZWE	Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...

44011 rows × 3 columns

Univariate Analysis :

Univariate Analysis is the process of examining a single variable within your dataset. Each variable represents an individual feature or column. You can perform this analysis using graphical or mathematical techniques, which may involve calculating specific numerical values from the data. Some visual techniques for Univariate Analysis include:

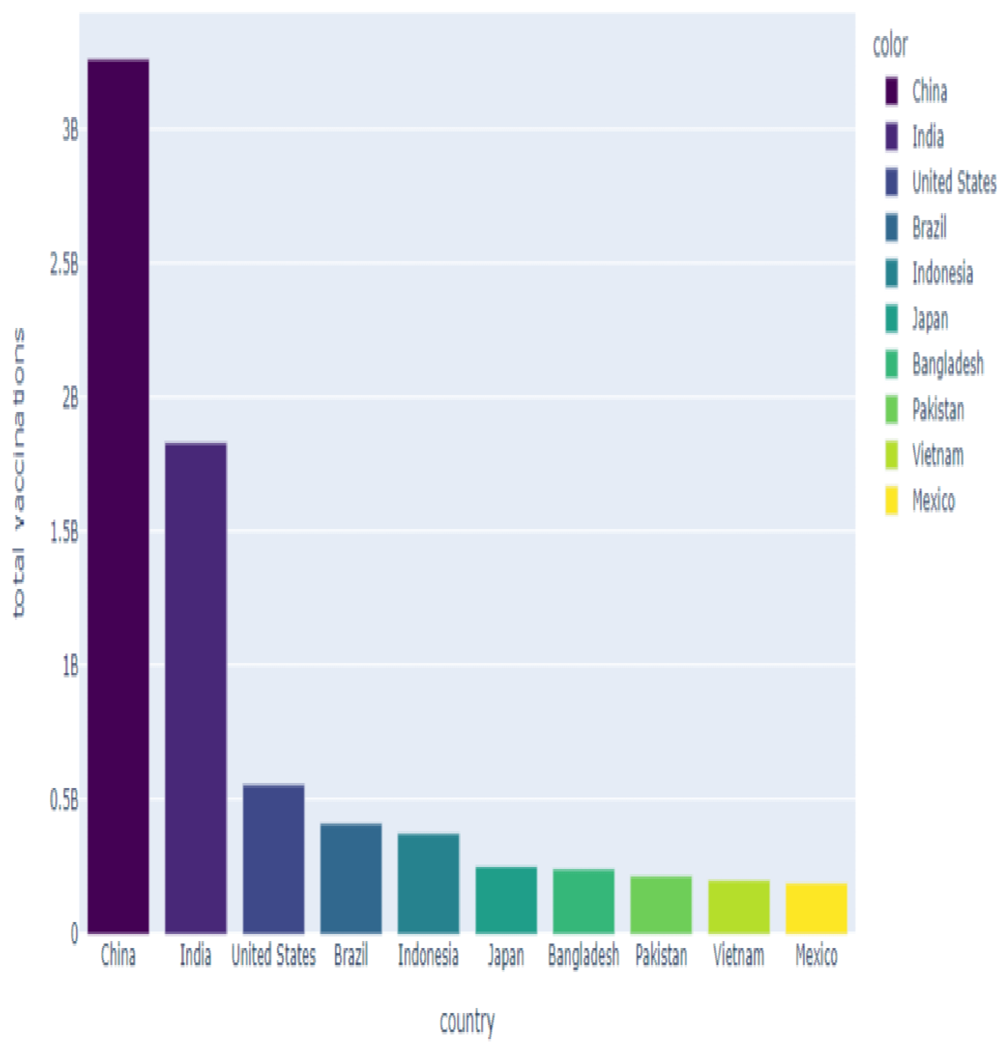
Bar chart for top 10 countries vaccinated:

```
In [*]: #bar chart for top 10 countries vaccinated

        fig = px.bar(x=total_vaccinations_top10.index ,y=total_vaccinations_top10.values,
                     color=total_vaccinations_top10.index,
                     labels={"x": "country", "y": "total vaccinations"},
                     color_discrete_sequence =px.colors.sequential.Viridis)

        fig.show()
```

Based on country and total vaccinations.



Map visualisation :

```
In [56]: #vaccinated per million
def create_choropleth(loc,z,text, title):
    fig = go.Figure(data=go.Choropleth(
          locations = loc,
           z=z,
           text=text,
           colorscale = 'viridis',
           autocolorscale=False,
           reversescale=True,
           marker_line_color='white',
           marker_line_width=0.5
    ))
    fig.update_geos(
         visible=True,
         resolution=50,
         showcountries=True,
         countrycolor = 'darkgrey'
         )
    fig.update_layout(
         title_text = title,
         geo=dict(
             showframe=False,
             showcoastlines=False,
             projection_type='natural earth'
     ))
    fig.show()
dayly_vac_million = data[['country', 'iso_code', 'daily_vaccinations_per_million']]
dayly_vac_million['daily_vaccinations_per_million'] = dayly_vac_million['daily_vaccinations_per_million'].fillna(0)
dayly_vac_million = dayly_vac_million.groupby(['country', 'iso_code']).max().reset_index()
create_choropleth(dayly_vac_million['iso_code'],
                  dayly_vac_million['daily_vaccinations_per_million'],
                  dayly_vac_million['country'],
                  'Daily vaccinations per million')
```

Vaccines used by each country:

```
In [65]: #vaccines used by each country


         vaccinebycountry_df = data[['country','iso_code','vaccines', 'total_vaccinations']]
         total_vaccinations= vaccinebycountry_df.groupby(['country']).max()[['total_vaccinations'
                           ,'vaccines','iso_code']].reset_index()


         fig = px.choropleth(total_vaccinations, locations = 'country', locationmode = 'country names',color = 'vaccines',
                            title = 'total Vaccines used for each country',hover_data= ['total_vaccinations'],
                             color_discrete_map=dict(zip(total_vaccinations['vaccines'], px.colors.sequential.Viridis)),
                            labels={'vaccines': 'Name of vaccine', 'country': 'Country',
                                   'total_vaccinations': 'Number of vaccinations'})
         fig.update_geos(
             visible=True,
             resolution=50,
             showcountries=True,
             countrycolor="darkgrey"
             )
         fig.update_layout(
             geo=dict(
                 showframe=False,
                 showcoastlines=False,
                 projection_type='equirectangular'
             ),
         )
         fig.show()
```

total Vaccines used for each country

Name of vaccine

- 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