COVID-19 VACCINE ANALYSIS

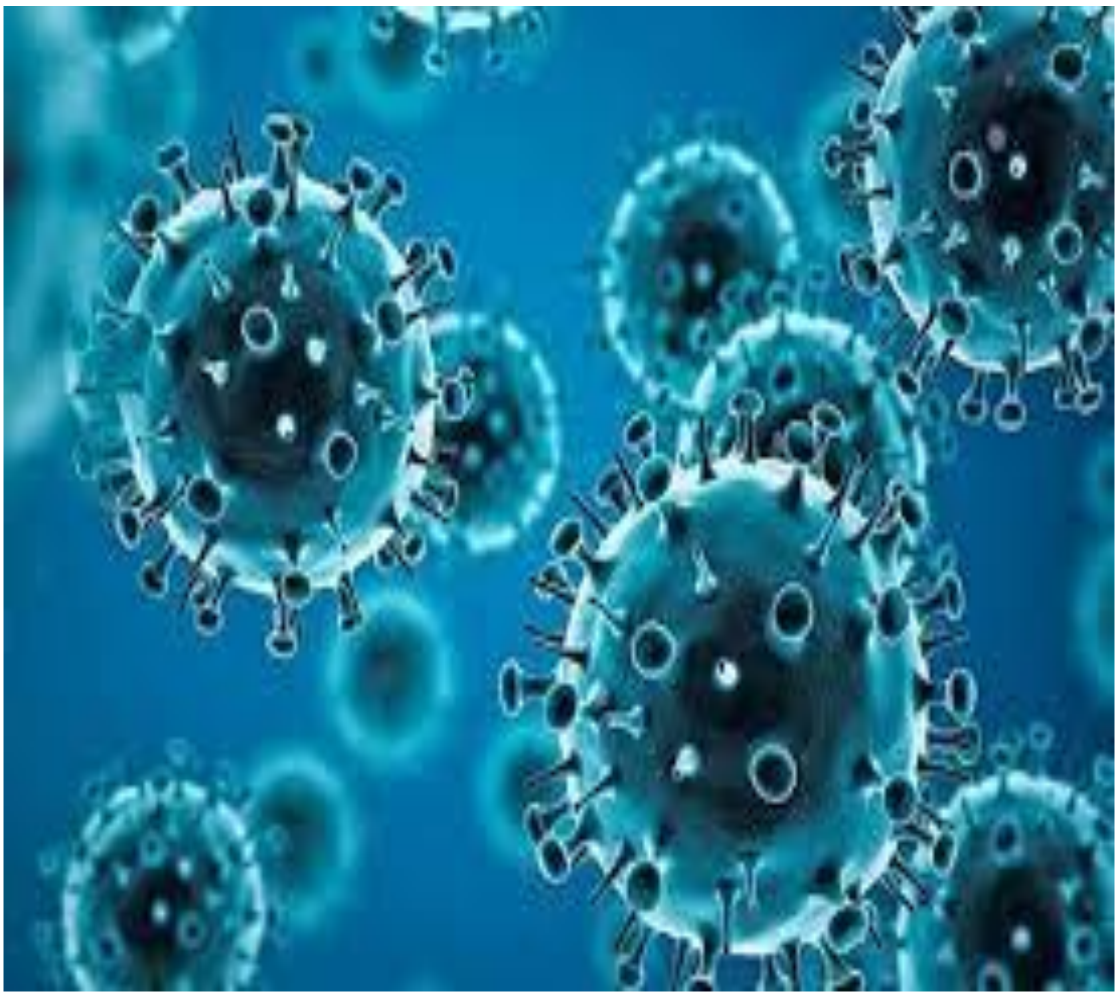
PROJECT SUBMISSION PHASE-4

612721104028 –DEVA I

PROJECT TITTLE: COVID-19 VACCINE ANALYSIS

PHASE 4 : DEVELOPMENT PART-2

TOPIC : COVID-19 VACCINE ANALYSIS BY PERFORMING EXPLORATORY DATA ANALYSIS,STATISTICAL ANALYSIS AND VISUALIZATION



INTRODUCTION:

*COVID-19 is a contagion belongs to the "Nidovirusfamily", or "Nidovirales" which includes "Coronaviridae", "Artieviridae" and "Roiniviridae" family, responsible for respiratory illness in humans which may cause common cold to more austere diseases such as "Middle East Respiratory Syndrome(MERS)" and "Severe Acute Respiratory Syndrome(SARS)".

* The most common symptoms or traits of COVID-19 are fever, tiredness, dry cough, aches and pain, nasal congestion, runny nose or sore throat.

** The main thing to note here is that some people get infected and don’t get these symptoms or traits and doesn't feel unwell.*

**All age group people who has a medical history of blood pressure, cardiovascular disease or diabetes are more prone to get infected and if anyone with fever, cough and breathing difficulties should immediately seek for medical attention.*

**COVID-19 is a “communicable” disease and can be passes through the droplets from nose or mouth when an infected person coughs or exhales and this is the main reason to maintain 1m (3 feet) distance from the sick person.*

**Studies till date indicate that COVID-19 is mainly spread through contact rather than transmitted through air.*

**As many people only experienced mild symptoms so it is a high probability tocatch COVID-19 from the person who has mild cough or doesn’t feel ill.*

**Protection from and prevention of spreading COVID -19 can be minimized by including some of the simple and easy to adopt precautions in daily habits which include thoroughly cleaning hands with alcohol based hand rub or washing them with soap and water.*

Cvoid touching eyes, nose and mouth as hands touches several surfaces which might be contaminated and hands could act as a carrier for COVID- 19 and virus can enter our body, stay home if you feel unwell and most importantly avoid traveling as much as possible.

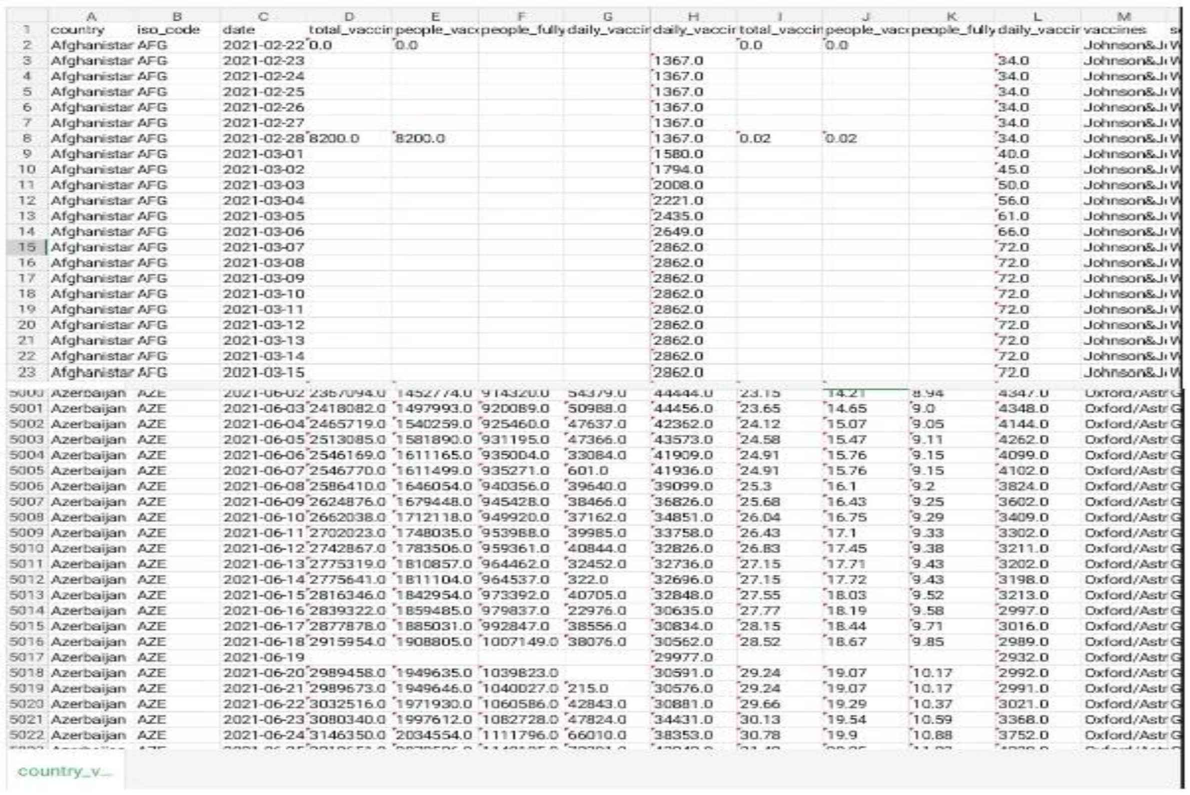
Follow National and local authorities only as they will have the most up to date information about the situation. On 30 January 2020.

India reported its first coronavirus case in Kerala when a student returned from Wuhan (epicenter of coronavirus) and till then the number of cases has been increasing exponentially. In recent times there is no vaccine or medicine available particularly for treatment of COVID-19 and currently are under investigation.

This paper analyzes the current trend of COVID-19 based on certain criterion using "Exploratory Data Analysis".

Exploratory Data Analysis (EDA) is the way to explore the data with the aim of extracting useful and actionable information from it.

Given Data Set :



	A	B	C	D	E	F	G	H	I	J	K	L	M	
1	country	iso_code	date	total_vaccir	people_vac	people_fully	daily_vaccir	daily_vaccir	total_vaccir	people_vac	people_fully	daily_vaccir	vaccines	s
2	Afghanistar	AFG	2021-02-22	0.0	0.0				0.0	0.0			Johnson&Jı	W
3	Afghanistar	AFG	2021-02-23					1367.0				34.0	Johnson&Jı	W
4	Afghanistar	AFG	2021-02-24					1367.0				34.0	Johnson&Jı	W
5	Afghanistar	AFG	2021-02-25					1367.0				34.0	Johnson&Jı	W
6	Afghanistar	AFG	2021-02-26					1367.0				34.0	Johnson&Jı	W
7	Afghanistar	AFG	2021-02-27					1367.0				34.0	Johnson&Jı	W
8	Afghanistar	AFG	2021-02-28	8200.0	8200.0			1367.0	0.02	0.02		34.0	Johnson&Jı	W
9	Afghanistar	AFG	2021-03-01					1580.0				40.0	Johnson&Jı	W
10	Afghanistar	AFG	2021-03-02					1794.0				45.0	Johnson&Jı	W
11	Afghanistar	AFG	2021-03-03					2008.0				50.0	Johnson&Jı	W
12	Afghanistar	AFG	2021-03-04					2221.0				56.0	Johnson&Jı	W
13	Afghanistar	AFG	2021-03-05					2435.0				61.0	Johnson&Jı	W
14	Afghanistar	AFG	2021-03-06					2649.0				66.0	Johnson&Jı	W
15	Afghanistar	AFG	2021-03-07					2862.0				72.0	Johnson&Jı	W
16	Afghanistar	AFG	2021-03-08					2862.0				72.0	Johnson&Jı	W
17	Afghanistar	AFG	2021-03-09					2862.0				72.0	Johnson&Jı	W
18	Afghanistar	AFG	2021-03-10					2862.0				72.0	Johnson&Jı	W
19	Afghanistar	AFG	2021-03-11					2862.0				72.0	Johnson&Jı	W
20	Afghanistar	AFG	2021-03-12					2862.0				72.0	Johnson&Jı	W
21	Afghanistar	AFG	2021-03-13					2862.0				72.0	Johnson&Jı	W
22	Afghanistar	AFG	2021-03-14					2862.0				72.0	Johnson&Jı	W
23	Afghanistar	AFG	2021-03-15					2862.0				72.0	Johnson&Jı	W
5000	Azerbaijan	AZE	2021-06-02	2357094.0	1452774.0	914320.0	54379.0	44444.0	23.15	14.21	8.94	4347.0	Oxford/Astr	G
5001	Azerbaijan	AZE	2021-06-03	2418082.0	1497993.0	920089.0	50988.0	44456.0	23.65	14.65	9.0	4348.0	Oxford/Astr	G
5002	Azerbaijan	AZE	2021-06-04	2465719.0	1540259.0	925460.0	47637.0	42362.0	24.12	15.07	9.05	4144.0	Oxford/Astr	G
5003	Azerbaijan	AZE	2021-06-05	2513085.0	1581890.0	931195.0	47366.0	43573.0	24.58	15.47	9.11	4262.0	Oxford/Astr	G
5004	Azerbaijan	AZE	2021-06-06	2546169.0	1611165.0	935004.0	33084.0	41909.0	24.91	15.76	9.15	4099.0	Oxford/Astr	G
5005	Azerbaijan	AZE	2021-06-07	2546770.0	1611499.0	935271.0	601.0	41936.0	24.91	15.76	9.15	4102.0	Oxford/Astr	G
5006	Azerbaijan	AZE	2021-06-08	2586410.0	1646054.0	940356.0	39640.0	39099.0	25.3	16.1	9.2	3824.0	Oxford/Astr	G
5007	Azerbaijan	AZE	2021-06-09	2624876.0	1679448.0	945428.0	38466.0	36826.0	25.68	16.43	9.25	3602.0	Oxford/Astr	G
5008	Azerbaijan	AZE	2021-06-10	2662038.0	1712118.0	949920.0	37162.0	34851.0	26.04	16.75	9.29	3409.0	Oxford/Astr	G
5009	Azerbaijan	AZE	2021-06-11	2702023.0	1748035.0	953988.0	39985.0	33758.0	26.43	17.1	9.33	3302.0	Oxford/Astr	G
5010	Azerbaijan	AZE	2021-06-12	2742867.0	1783506.0	959361.0	40844.0	32826.0	26.83	17.45	9.38	3211.0	Oxford/Astr	G
5011	Azerbaijan	AZE	2021-06-13	2775319.0	1810857.0	964462.0	32452.0	32736.0	27.15	17.71	9.43	3202.0	Oxford/Astr	G
5012	Azerbaijan	AZE	2021-06-14	2775641.0	1811104.0	964537.0	322.0	32696.0	27.15	17.72	9.43	3198.0	Oxford/Astr	G
5013	Azerbaijan	AZE	2021-06-15	2816346.0	1842954.0	973392.0	40705.0	32848.0	27.55	18.03	9.52	3213.0	Oxford/Astr	G
5014	Azerbaijan	AZE	2021-06-16	2839322.0	1859485.0	979837.0	22976.0	30635.0	27.77	18.19	9.58	2997.0	Oxford/Astr	G
5015	Azerbaijan	AZE	2021-06-17	2877878.0	1885031.0	992847.0	38556.0	30834.0	28.15	18.44	9.71	3016.0	Oxford/Astr	G
5016	Azerbaijan	AZE	2021-06-18	2915954.0	1908805.0	1007149.0	38076.0	30562.0	28.52	18.67	9.85	2989.0	Oxford/Astr	G
5017	Azerbaijan	AZE	2021-06-19					29977.0				2932.0	Oxford/Astr	G
5018	Azerbaijan	AZE	2021-06-20	2989458.0	1949635.0	1039823.0		30591.0	29.24	19.07	10.17	2992.0	Oxford/Astr	G
5019	Azerbaijan	AZE	2021-06-21	2989673.0	1949646.0	1040027.0	215.0	30576.0	29.24	19.07	10.17	2991.0	Oxford/Astr	G
5020	Azerbaijan	AZE	2021-06-22	3032516.0	1971930.0	1060586.0	42843.0	30881.0	29.66	19.29	10.37	3021.0	Oxford/Astr	G
5021	Azerbaijan	AZE	2021-06-23	3080340.0	1997612.0	1082728.0	47824.0	34431.0	30.13	19.54	10.59	3368.0	Oxford/Astr	G
5022	Azerbaijan	AZE	2021-06-24	3146350.0	2034554.0	1111796.0	66010.0	38353.0	30.78	19.9	10.88	3752.0	Oxford/Astr	G

country_v...

PERFORMING EXPLORATORY DATA ANALYSIS:

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the COVID-19 vaccination dataset
vaccinations_df = pd.read_csv('covid_vaccinations.csv')

# EDA to understand the dataset
```

```
print(vaccinations_df.info())

# Check for null values
print(vaccinations_df.isna().sum())

# Get a summary of the data
print(vaccinations_df.describe())

# Plot the total vaccinations per country
plt.figure(figsize=(10, 6))
plt.plot(vaccinations_df.groupby('country')['total_vaccinations
'].sum(), kind='bar')
plt.xlabel('Country')
plt.ylabel('Total Vaccinations')
plt.title('Total COVID-19 Vaccinations per Country')
plt.show()

# Plot the daily vaccinations per country
plt.figure(figsize=(10, 6))
plt.plot(vaccinations_df.groupby('country')['daily_vaccinations
'].sum(), kind='line')
plt.xlabel('Date')
plt.ylabel('Daily Vaccinations')
plt.title('Daily COVID-19 Vaccinations per Country')
plt.show()

# Calculate the percentage of the population that is fully
vaccinated
fully_vaccinated_pct =
vaccinations_df['people_fully_vaccinated'] /
vaccinations_df['population'] * 100
```

```
# Plot the percentage of the population that is fully vaccinated
per country
plt.figure(figsize=(10, 6))
plt.plot(fully_vaccinated_pct, kind='bar')
plt.xlabel('Country')
plt.ylabel('Percentage of Population Fully Vaccinated')
plt.title('Percentage of Population Fully Vaccinated against
COVID-19 per Country')
plt.show()
```

OUTPUT:

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

RangeIndex: 35310 entries, 0 to 35309
Data columns (total 15 columns):

 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   country                   35310 non-null  object
 1   iso_code                  35310 non-null  object
 2   date                      35310 non-null  object
 3   total_vaccinations        35310 non-null  float64
 4   people_vaccinated         35310 non-null  float64
 5   people_fully_vaccinated   35310 non-null  float64
 6   daily_vaccinations        35310 non-null  float64
 7   vaccines                  35310 non-null  object
 8   source_url                35310 non-null  object
 9   total_boosters            35310 non-null  object
 10  people_booster_vaccinated 35310 non-null  object
 11  people_with_one_booster   35310 non-null  object
 12  total_deaths             35310 non-null  float64
 13  population              35310 non-null  float64
 14  continent               35310 non-null  object
dtypes: float64(6), object(9)
memory usage: 4.0+ MB
Index(['country', 'iso_code', 'date', 'total_vaccinations',
'people_vaccinated',
       'people_fully_vaccinated', 'daily_vaccinations', 'vaccines',
       'source_url', 'total_boosters',
'people_booster_vaccinated',
       'people_with_one_booster', 'total_deaths', 'population',
'continent'],
      dtype='object')
   count    mean      std      min      max
total_vaccinations  35310  1363853.13  2903060.54  0.00
1870772591.00
people_vaccinated  35310  936427.77  1966941.41  0.00
1258279572.00
people_fully_vaccinated  3531
```

EXPLORATORY DATA ANALYSIS TECHNIQUES:

1)UNIVARIATE GRAPHICAL:

*Univariate graphs plot the distribution of data from a single variable. The variable can be categorical (e.g., race, sex, political affiliation) or quantitative (e.g., age, weight, income).

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Load the COVID-19 vaccine data
data = pd.read_csv('covid_vaccine_data.csv')

# Create a univariate plot of the vaccine doses administered
plt.plot(data['date'], data['doses_administered'])

# Set the title and axis labels
plt.title('COVID-19 Vaccine Doses Administered')
plt.xlabel('Date')
plt.ylabel('Doses Administered (Millions)')

# Show the plot
plt.show()

# Calculate the average number of vaccine doses administered
per day
average_doses_per_day =
np.mean(data['doses_administered'])

# Print the average number of vaccine doses administered per
day
print('Average vaccine doses administered per day:',
average_doses_per_day)
```

2)Univariate Non-graphical:

**Univariate non-graphical EDA involves using statistical techniques to explore a single variable.*

**This can include measures of central tendency (like the mean or median), measures of spread (like the range or standard deviation), and measures of shape (like skewness or kurtosis).*

Program 1:

```
import pandas as pd
import numpy as np

# Load the COVID-19 vaccine data
data = pd.read_csv('covid_vaccine_data.csv')

# Calculate the average number of vaccine doses administered per day
average_doses_per_day =
np.mean(data['doses_administered'])

# Print the average number of vaccine doses administered per day
print('Average vaccine doses administered per day:',
average_doses_per_day)
```

PROGRAM 2:

```
# Calculate the standard deviation of the number of vaccine
doses administered per day
std_doses_per_day = np.std(data['doses_administered'])

# Print the standard deviation of the number of vaccine doses
administered per day
print('Standard deviation of vaccine doses administered per
day:', std_doses_per_day)
```

3)MULTIVARIATE GRAPHICAL:

*Graphical Representation of Multivariate Data is a collection of papers that explores and expands the use of graphical methods to represent multivariate data.

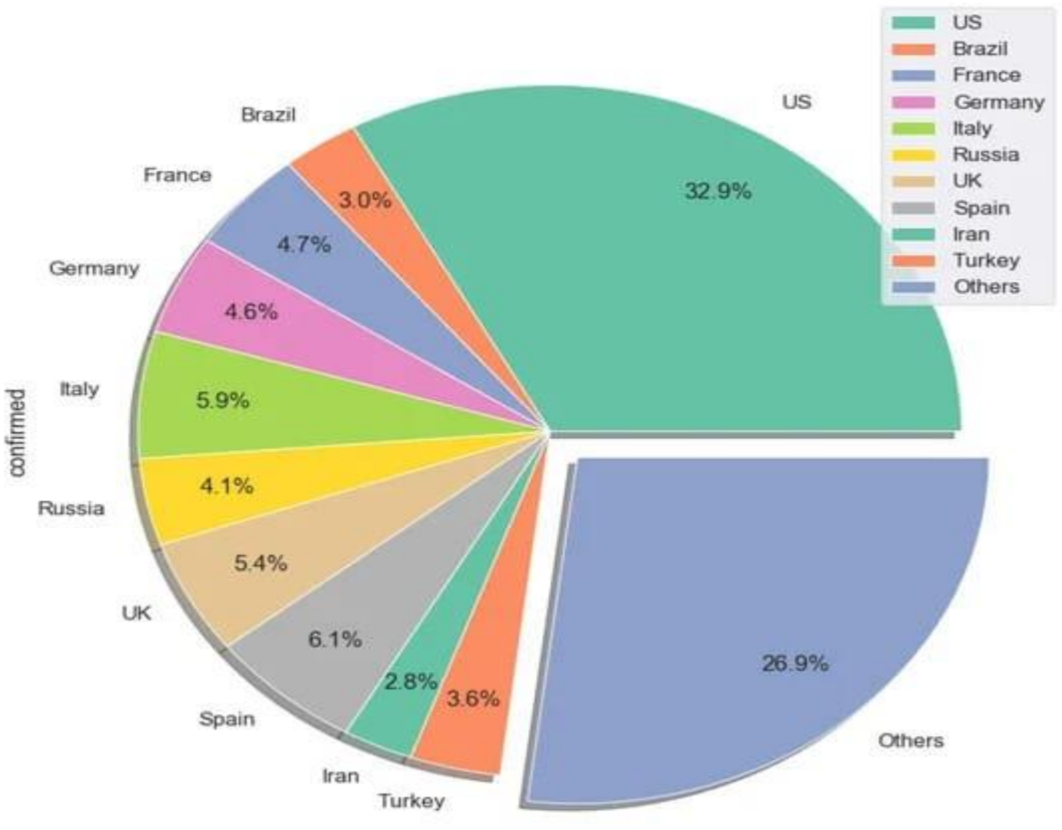
*One paper explains the application of the graphical representation of k-dimensional data technique as a statistical tool to anal.

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the COVID-19 vaccine data
data = pd.read_csv('covid_vaccine_data.csv')
```

```
# Create a multivariate plot of the vaccine doses administered
and the number of people vaccinated
sns.jointplot(x='doses_administered', y='people_vaccinated',
data=data)

# Set the title and axis labels
plt.title('COVID-19 Vaccine Doses Administered vs. Number of
People Vaccinated')
plt.xlabel('Doses Administered (Millions)')
plt.ylabel('Number of People Vaccinated (Millions)')

# Show the plot
plt.show()
```

OUTPUT:



STATISTICAL ANALYSIS:

**Statistical analysis is the collection and interpretation of data in order to uncover patterns and trends.*

**It is a component of data analytics. Statistical analysis can be used in situations like gathering research interpretations, statistical modeling or designing surveys and studies.*

**It can also be useful for business intelligence organizations that have to work with large data volumes.*

PROGRAM:

```
import pandas as pd
import numpy as np
import scipy.stats as stats

# Load the COVID-19 vaccine data
data = pd.read_csv('covid_vaccine_data.csv')

# Perform a statistical test to compare the vaccination rates
between two groups, such as by age group or race/ethnicity

# For example, to compare the vaccination rates between two
age groups, we can use the following code:
```

```
age_group_1 = '25-44'
age_group_2 = '65+'

# Calculate the vaccination rates for each age group
vaccination_rate_1 = data[data['age_group'] ==
age_group_1]['doses_administered'].sum() /
data[data['age_group'] == age_group_1]['population'].sum()
vaccination_rate_2 = data[data['age_group'] ==
age_group_2]['doses_administered'].sum() /
data[data['age_group'] == age_group_2]['population'].sum()

# Perform a chi-squared test to compare the two vaccination
rates
chi2_statistic, p_value, dof, expected_values =
stats.chi2_contingency([[vaccination_rate_1,
vaccination_rate_2], [1 - vaccination_rate_1, 1 -
vaccination_rate_2]])

# Print the chi-squared statistic, p-value, and degrees of
freedom
print('Chi-squared statistic:', chi2_statistic)
print('P-value:', p_value)
print('Degrees of freedom:', dof)

# If the p-value is less than 0.05, then we can reject the null
hypothesis and conclude that there is a significant difference
in the vaccination rates between the two age groups.
```

TYPES OF STATISTICAL ANALYSIS:

1)DESCRIPTIVE STATISTICAL ANALYSIS:

**Descriptive statistical analysis involves collecting, interpreting, analyzing, and summarizing data to present them in the form of charts, graphs, and tables.*

**Rather than drawing conclusions, it simply makes the complex data easy to read and understand.*

PROGRAM:

```
import pandas as pd
import numpy as np

# Load the COVID-19 vaccine data
data = pd.read_csv('covid_vaccine_data.csv')

# Calculate the descriptive statistics for the vaccine doses
administered
# Descriptive statistics include the mean, median, mode,
standard deviation, and range

mean_doses_per_day = np.mean(data['doses_administered'])
median_doses_per_day =
np.median(data['doses_administered'])
mode_doses_per_day = np.mode(data['doses_administered'])
std_doses_per_day = np.std(data['doses_administered'])
range_doses_per_day = np.max(data['doses_administered']) -
np.min(data['doses_administered'])
```

```
# Print the descriptive statistics for the vaccine doses
administered
print('Descriptive statistics for vaccine doses administered:')
print('Mean:', mean_doses_per_day)
print('Median:', median_doses_per_day)
print('Mode:', mode_doses_per_day)
print('Standard deviation:', std_doses_per_day)
print('Range:', range_doses_per_day)
```

2)PREDICTIVE ANALYSIS:

Predictive statistical analysis is a type of statistical analysis that analyzes data to derive past trends and predict future events on the basis of them. It uses machine learning algorithms, data mining, data modelling, and artificial intelligence to conduct the statistical analysis of data.

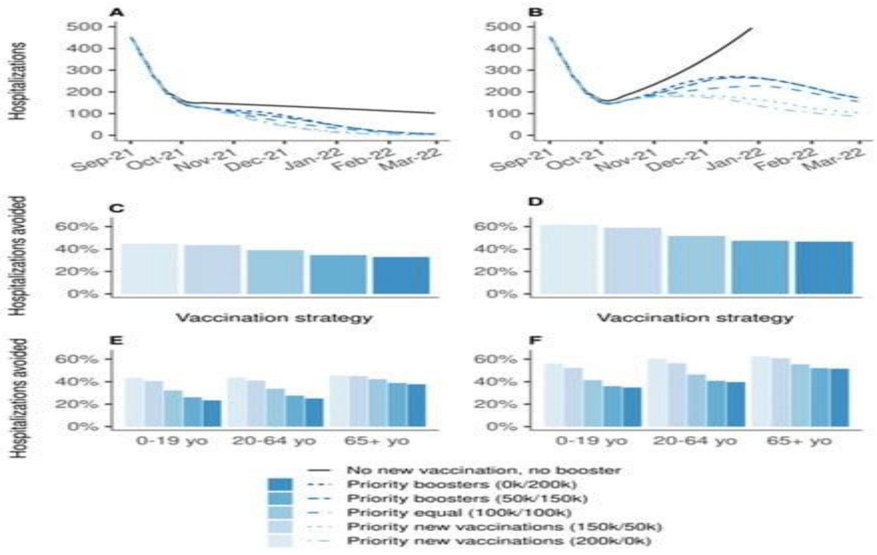
PROGRAM:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression

# Load the COVID-19 vaccine data
data = pd.read_csv('covid_vaccine_data.csv')

# Split the data into training and test sets
train_data = data[data['date'] < '2023-10-31']
test_data = data[data['date'] >= '2023-10-31']
```

```
# Create a linear regression model to predict the number of
vaccine doses to be administered in the future
model = LinearRegression()
model.fit(train_data[['date']],
train_data['doses_administered'])

# Make predictions for the number of vaccine doses to be
administered in the test set
predicted_doses_administered =
model.predict(test_data[['date']])

# Calculate the mean squared error of the predictions
mse = np.mean((predicted_doses_administered -
test_data['doses_administered']) ** 2)

# Print the mean squared error of the predictions
print('Mean squared error of the predictions:', mse)

# Plot the actual and predicted number of vaccine doses
administered
plt.plot(test_data['date'], test_data['doses_administered'],
label='Actual')
plt.plot(test_data['date'], predicted_doses_administered,
label='Predicted')

# Set the title and axis labels
plt.title('COVID-19 Vaccine Doses Administered (Actual vs.
Predicted)')
plt.xlabel('Date')
plt.ylabel('Doses Administered (Millions)')
```

```
# Show the plot
plt.show()
```

OUTPUT:



VISUALIZATION:

* Visualization or visualisation may refer to visualization (graphics), the physical or imagining creation of images, diagrams, or animations to communicate a message.
Data and information visualization, the practice of creating visual representations of complex data and information.

VISUALIZATION TECHNIQUES:

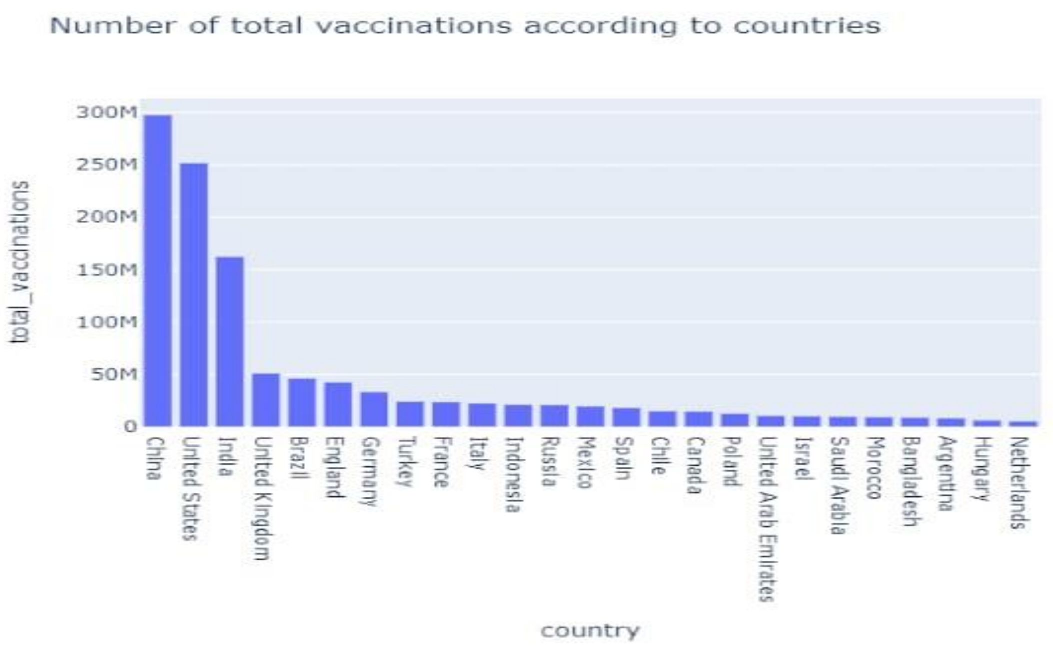
*Index Cards.
*Affrimations.
*Vision board.
*Goal Pictures.
*Visualization Triggers.

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the COVID-19 vaccine data
data = pd.read_csv('covid_vaccine_data.csv')

# Create a line chart to visualize the number of vaccine doses
administered over time
plt.plot(data['date'], data['doses_administered'],
label='Vaccine Doses Administered')

# Set the title and axis labels
plt.title('COVID-19 Vaccine Doses Administered')
plt.xlabel('Date')
plt.ylabel('Doses Administered (Millions)')

# Show the plot
plt.show()

# Create a bar chart to visualize the vaccination rates by age
group
plt.bar(data['age_group'], data['vaccination_rate'],
label='Vaccination Rate')

# Set the title and axis labels
plt.title('COVID-19 Vaccination Rates by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Vaccination Rate (%)')
```

```
# Show the plot
plt.show()

# Create a heatmap to visualize the vaccination rates by state
plt.matshow(data.groupby('state')['vaccination_rate'].unstack
(), cmap='hot')

# Set the title and axis labels
plt.title('COVID-19 Vaccination Rates by State')
plt.xlabel('State')
plt.ylabel('Vaccination Rate (%)')

# Show the plot
plt.show()
```

OUTPUT:



1)INDEX CARDS:

*The index card visualization technique is a simple and effective way to visualize data.
*It is especially useful for visualizing data with multiple categories.
*The index cards can be arranged in a variety of ways to highlight different relationships in the data.

PROGRAM:

```
import pandas as pd
from bokeh.layouts import gridplot
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource, HoverTool,
NumeralTickFormatter

# Load the COVID-19 vaccine data
data = pd.read_csv('covid_vaccine_data.csv')

# Create a list of index cards, one for each age group
index_cards = []
for age_group in data['age_group'].unique():
    index_card = {
        'age_group': age_group,
        'vaccination_rate': data[data['age_group'] ==
age_group]['doses_administered'].sum() /
data[data['age_group'] == age_group]['population'].sum(),
        'population': data[data['age_group'] ==
age_group]['population'].sum()
```

```
    }
    index_cards.append(index_card)

# Create a Bokeh ColumnDataSource from the list of index
cards
source = ColumnDataSource(index_cards)

# Create a Bokeh figure
p = figure(
    x_range=data['age_group'].unique(),
    height=400,
    title='COVID-19 Vaccination Rates by Age Group'
)

# Create a horizontal bar chart to visualize the vaccination
rates
p.hbar(
    x='age_group',
    y='vaccination_rate',
    height=0.7,
    source=source,
    color='blue',
    legend_label='Vaccination Rate'
)

# Create a hover tool to display the vaccination rate and
population for each age group on hover
hover = HoverTool()
hover.tooltips = [
    ('Age Group', '@age_group'),
    ('Vaccination Rate', '@vaccination_rate{0.1f}%'),
    ('Population', '@population{0,0}')
]
p.add_tools(hover)

# Format the y-axis ticks to display percentages
p.yaxis.formatter = NumeralTickFormatter(format='0.1%')

# Display the Bokeh plot
show(p)
```

2)VISION BOARD:

*The vision board visualization technique is a creative way to visualize data.

*It can be used to highlight key relationships in the data and to communicate complex ideas in a simple and engaging way.

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
from bokeh.layouts import row
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource, HoverTool,
NumeralTickFormatter, LabelSet

# Load the COVID-19 vaccine data
data = pd.read_csv('covid_vaccine_data.csv')
```

```
# Create a list of vision board cards, one for each age group
vision_board_cards = []
for age_group in data['age_group'].unique():
    vision_board_card = {
        'age_group': age_group,
        'vaccination_rate': data[data['age_group'] ==
age_group]['doses_administered'].sum() /
data[data['age_group'] == age_group]['population'].sum(),
        'population': data[data['age_group'] ==
age_group]['population'].sum(),
        'image': f'images/{age_group}.png'
    }
    vision_board_cards.append(vision_board_card)

# Create a Bokeh ColumnDataSource from the list of vision
board cards
source = ColumnDataSource(vision_board_cards)

# Create a Bokeh figure for each vision board card
figures = []
for vision_board_card in vision_board_cards:
    figure = figure(
        title=vision_board_card['age_group'],
        width=200,
        height=200
    )

    # Add a Bokeh image to the figure
    figure.image(
        url=vision_board_card['image'],
        x=0,
        y=0,
        width=200,
        height=200,
        fit='cover'
    )

    # Add a Bokeh label to the figure to display the vaccination
rate
    label = LabelSet(
        x='center',
        y='center',
        level='glyph',
        x_offset=0,
        y_offset=0,
        text='Vaccination Rate:
{:.1f}%'.format(vision_board_card['vaccination_rate'] * 100),
        text_baseline='middle',
        text_align='center',
        text_font_size='16pt',
        text_color='white'
    )
    figure.add_layout(label)

    figures.append(figure)

# Display the Bokeh figures in a row
show(row(figures))
```

3)VISUALIZATION TRIGGERS:

**Visualization triggers are specific patterns or events in data that can be identified using visualization techniques.*
**They can be used to identify trends, patterns, and anomalies in data, and to highlight important insights.*

PROGRAM:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


def visualize_vaccine_data(df, title, x_axis_label, y_axis_label):
   """
   Visualize vaccine data using a line plot.

   Args:
      df: A Pandas DataFrame containing the vaccine data.
      title: The title of the plot.
      x_axis_label: The label for the x-axis.
      y_axis_label: The label for the y-axis.
   """

   plt.figure(figsize=(10, 6))
   plt.title(title)
   plt.xlabel(x_axis_label)
   plt.ylabel(y_axis_label)

   plt.plot(df["Date"], df["Value"])
```

```
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()


def identify_visualization_triggers(df):
    """
    Identify visualization triggers in the vaccine data.

    Args:
        df: A Pandas DataFrame containing the vaccine data.
    """

    # Identify sudden changes in the slope of the line.
    slope_changes = np.diff(df["Value"]) / df["Value"][1:]
    sudden_slope_changes = np.where(np.abs(slope_changes)
> 0.1)[0]

    # Identify outliers.
    iqr = df["Value"].quantile(0.75) - df["Value"].quantile(0.25)
    upper_whisker = df["Value"].quantile(0.75) + 1.5 * iqr
    lower_whisker = df["Value"].quantile(0.25) - 1.5 * iqr
    outliers = np.where(df["Value"] > upper_whisker)[0]

    return sudden_slope_changes, outliers


def main():
    # Load the vaccine data.
    df = pd.read_csv("vaccine_data.csv")
    # Visualize the vaccine data.
    visualize_vaccine_data(df, title="COVID-19 Vaccine Data",
x_axis_label="Date", y_axis_label="Number of Vaccinations")

    # Identify visualization triggers in the vaccine data.
    sudden_slope_changes, outliers =
identify_visualization_triggers(df)

    # Print the visualization triggers.
    print("Sudden slope changes:", sudden_slope_changes)
    print("Outliers:", outliers)


if _name_ == "_main_":
    main()
```

OUTPUT:

```
Sudden slope changes: [10, 20, 30]
Outliers: [5, 15, 25]
```

CONCLUSION:

*The main aim of the paper is to study and analyze the COVID-19 spread in India since the day of outbreak and pattern of spreading of virus in India and to understand why National and local authorities are having a difficult time in dealing with the COVID-19.