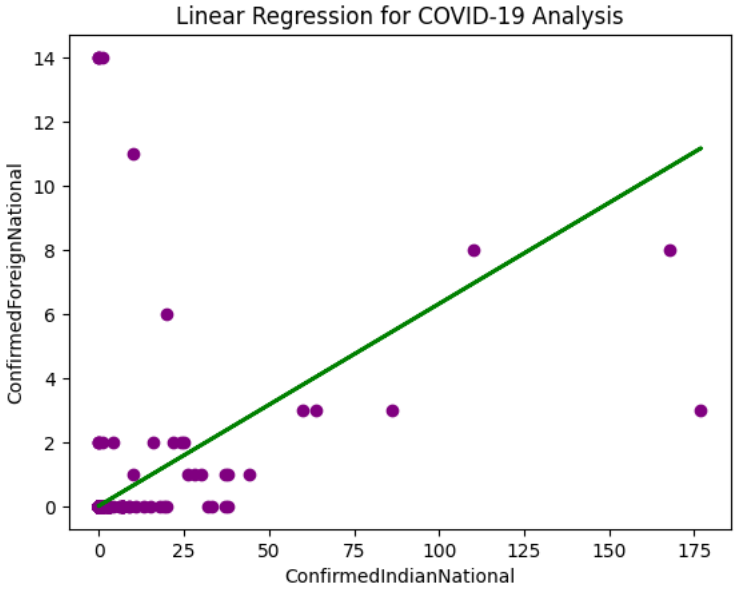
```
import pandas as pd
from matplotlib import pyplot as plt

covid_data = pd.read_csv('/content/covid_19_india (2) (1).csv')

print(covid_data.head())


columns_of_interest = ['ConfirmedForeignNational','Cured','Deaths','Confirmed']


subset_data = covid_data[columns_of_interest]


correlation_matrix = subset_data.corr()

print(correlation_matrix)
```

```
   Sno        Date     Time State/UnionTerritory ConfirmedIndianNational  \
0    1  30-01-2020  6:00 PM               Kerala                        1
1    2  31-01-2020  6:00 PM               Kerala                        1
2    3  01-02-2020  6:00 PM               Kerala                        2
3    4  02-02-2020  6:00 PM               Kerala                        3
4    5  03-02-2020  6:00 PM               Kerala                        3

  ConfirmedForeignNational  Cured  Deaths  Confirmed  Latitude  Longitude  \
0                        0      0       0          1   10.8505    76.2711
1                        0      0       0          1   10.8505    76.2711
2                        0      0       0          2   10.8505    76.2711
3                        0      0       0          3   10.8505    76.2711
4                        0      0       0          3   10.8505    76.2711

   Unnamed: 11
0          NaN
1          NaN
2          NaN
3          NaN
4          NaN
              Cured    Deaths  Confirmed
Cured      1.000000  0.917529   0.997751
Deaths     0.917529  1.000000   0.918346
Confirmed  0.997751  0.918346   1.000000
<ipython-input-14-0ef50ce02a86>:17: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
  correlation_matrix = subset_data.corr()
```

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt


df = pd.read_csv('/content/covid_19_india (2).csv',parse_dates=[0], dayfirst=True)
df.head(5)
data = df[['Date', 'ConfirmedForeignNational', 'ConfirmedIndianNational']]

data_copy['ConfirmedForeignNational'] = pd.to_numeric(data_copy['ConfirmedForeignNational'], errors='coerce').fillna(0)
data_copy['ConfirmedIndianNational'] = pd.to_numeric(data_copy['ConfirmedIndianNational'], errors='coerce').fillna(0)


data_copy['Date'] = pd.to_numeric(pd.to_datetime(data_copy['Date']))


X = data_copy[['ConfirmedIndianNational']]
y = data_copy['ConfirmedForeignNational']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


model = LinearRegression()


model.fit(X_train, y_train)


y_pred = model.predict(X_test)


plt.scatter(X_test, y_test, color='purple')
plt.plot(X_test, y_pred, color='green', linewidth=2)
plt.xlabel('ConfirmedIndianNational')
plt.ylabel('ConfirmedForeignNational')
plt.title('Linear Regression for COVID-19 Analysis')
plt.show()


print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
r_squared = metrics.r2_score(y_test, y_pred)
print('R-squared:', r_squared)
```

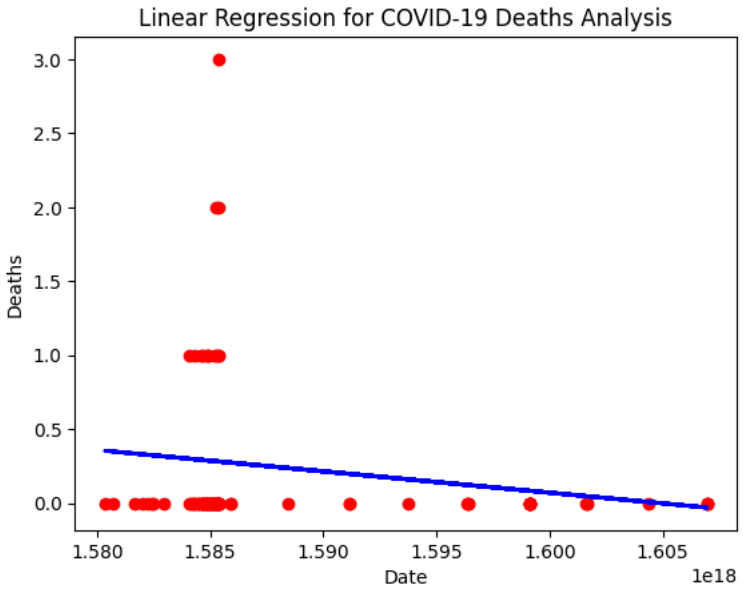
```
<ipython-input-16-78b4b3c5364c>:20: UserWarning: Parsing dates in DD/MM/YYYY format w
  data_copy['Date'] = pd.to_numeric(pd.to_datetime(data_copy['Date']))
```




```
Mean Absolute Error: 0.05610802638359297
Mean Squared Error: 0.29054636099174497
Root Mean Squared Error: 0.5390235254529666
R-squared: 0.07979759294071531
```

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt
df = pd.read_csv('/content/covid_19_india (2) (1).csv',parse_dates=[0], dayfirst=True)
data = df[['Date', 'Deaths', 'State/UnionTerritory', 'ConfirmedIndianNational']]
data['ConfirmedIndianNational'] = pd.to_numeric(data['ConfirmedIndianNational'], errors='coerce')
data = data.dropna(subset=['ConfirmedIndianNational'])
data['Date'] = pd.to_numeric(pd.to_datetime(data['Date']))
data['State/UnionTerritory'] = pd.Categorical(data['State/UnionTerritory'])
data['State/UnionTerritory'] = data['State/UnionTerritory'].cat.codes
X_death = data[['Date']]
y_death = data['Deaths']
X_train_death, X_test_death, y_train_death, y_test_death = train_test_split(X_death, y_death, test_size=0.2, random_state=42)
model_death = LinearRegression()
model_death.fit(X_train_death, y_train_death)
y_pred_death = model_death.predict(X_test_death)
plt.scatter(X_test_death, y_test_death, color='red')
plt.plot(X_test_death, y_pred_death, color='blue', linewidth=2)
plt.xlabel('Date')
plt.ylabel('Deaths')
plt.title('Linear Regression for COVID-19 Deaths Analysis')
plt.show()
print('Mean Absolute Error (Deaths):', metrics.mean_absolute_error(y_test_death, y_pred_death))
print('Mean Squared Error (Deaths):', metrics.mean_squared_error(y_test_death, y_pred_death))
print('Root Mean Squared Error (Deaths):', np.sqrt(metrics.mean_squared_error(y_test_death, y_pred_death)))
r_squared = metrics.r2_score(y_test_death, y_pred_death)
print('R-squared:', r_squared)
```

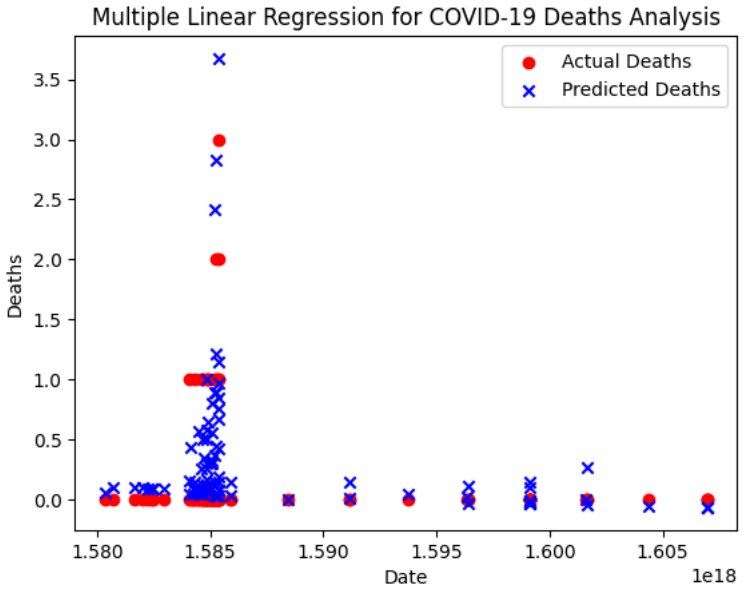
```
<ipython-input-2-10643f7a1882>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  data['ConfirmedIndianNational'] = pd.to_numeric(data['ConfirmedIndianNational'], er
<ipython-input-2-10643f7a1882>:19: UserWarning: Parsing dates in DD/MM/YYYY format wh
  data['Date'] = pd.to_numeric(pd.to_datetime(data['Date']))
```




```
Mean Absolute Error (Deaths): 0.35745622202363103
Mean Squared Error (Deaths): 0.2654817111880928
Root Mean Squared Error (Deaths): 0.5152491738839499
R-squared: 0.02076418004392
```

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt
data = df[['Date', 'Deaths', 'State/UnionTerritory', 'ConfirmedIndianNational']]
data['ConfirmedIndianNational'] = pd.to_numeric(data['ConfirmedIndianNational'], errors='coerce')
data = data.dropna(subset=['ConfirmedIndianNational'])
data['Date'] = pd.to_numeric(pd.to_datetime(data['Date']))
data['State/UnionTerritory'] = pd.Categorical(data['State/UnionTerritory'])
data['State/UnionTerritory'] = data['State/UnionTerritory'].cat.codes

# Prepare the features (X) and target variable (y) for Deaths
X_death = data[['Date', 'ConfirmedIndianNational']]
y_death = data['Deaths']

# Split the data into training and testing sets for Deaths
X_train_death, X_test_death, y_train_death, y_test_death = train_test_split(X_death, y_death, test_size=0.2, random_state=42)

# Create a linear regression model for Deaths
model_death = LinearRegression()

# Train the model for Deaths
model_death.fit(X_train_death, y_train_death)

# Make predictions on the testing set for Deaths
y_pred_death = model_death.predict(X_test_death)

# Visualize the results for Deaths
plt.scatter(X_test_death['Date'], y_test_death, color='red', label='Actual Deaths')
plt.scatter(X_test_death['Date'], y_pred_death, color='blue', label='Predicted Deaths', marker='x')
plt.xlabel('Date')
plt.ylabel('Deaths')
plt.title('Multiple Linear Regression for COVID-19 Deaths Analysis')
plt.legend()
plt.show()

# Evaluate the model for Deaths
print('Mean Absolute Error (Deaths):', metrics.mean_absolute_error(y_test_death, y_pred_death))
print('Mean Squared Error (Deaths):', metrics.mean_squared_error(y_test_death, y_pred_death))
print('Root Mean Squared Error (Deaths):', np.sqrt(metrics.mean_squared_error(y_test_death, y_pred_death)))
```

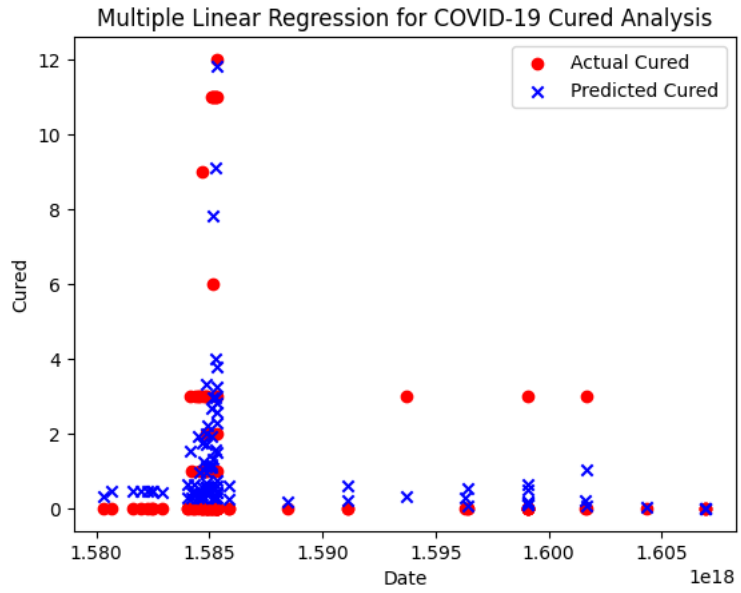
```
<ipython-input-4-fd923c2cad3b>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  data['ConfirmedIndianNational'] = pd.to_numeric(data['ConfirmedIndianNational'], er
<ipython-input-4-fd923c2cad3b>:19: UserWarning: Parsing dates in DD/MM/YYYY format wh
  data['Date'] = pd.to_numeric(pd.to_datetime(data['Date']))
```




```
Mean Absolute Error (Deaths): 0.37961096538543565
Mean Squared Error (Deaths): 0.5136193282176422
Root Mean Squared Error (Deaths): 0.7166723995087589
```

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt

# Assuming your DataFrame is named df
# Extract relevant columns
data = df[['Date', 'Cured', 'State/UnionTerritory', 'ConfirmedIndianNational']]

# Replace non-numeric values (e.g., '-') with NaN in 'ConfirmedIndianNational'
data['ConfirmedIndianNational'] = pd.to_numeric(data['ConfirmedIndianNational'], errors='coerce')

# Drop rows with NaN values in 'ConfirmedIndianNational' column
data = data.dropna(subset=['ConfirmedIndianNational'])

# Convert the 'Date' column to a numerical format for regression
data['Date'] = pd.to_numeric(pd.to_datetime(data['Date']))

# Convert 'State/UnionTerritory' to numerical format
data['State/UnionTerritory'] = pd.Categorical(data['State/UnionTerritory'])
data['State/UnionTerritory'] = data['State/UnionTerritory'].cat.codes

# Prepare the features (X) and target variable (y) for Cured
X_cured = data[['Date', 'ConfirmedIndianNational']]
y_cured = data['Cured']

# Split the data into training and testing sets for Cured
X_train_cured, X_test_cured, y_train_cured, y_test_cured = train_test_split(X_cured, y_cured, test_size=0.2, random_state=42)

# Create a linear regression model for Cured
model_cured = LinearRegression()

# Train the model for Cured
model_cured.fit(X_train_cured, y_train_cured)

# Make predictions on the testing set for Cured
y_pred_cured = model_cured.predict(X_test_cured)

# Visualize the results for Cured
plt.scatter(X_test_cured['Date'], y_test_cured, color='red', label='Actual Cured')
plt.scatter(X_test_cured['Date'], y_pred_cured, color='blue', label='Predicted Cured', marker='x')
plt.xlabel('Date')
plt.ylabel('Cured')
plt.title('Multiple Linear Regression for COVID-19 Cured Analysis')
plt.legend()
plt.show()

# Evaluate the model for Cured
print('Mean Absolute Error (Cured):', metrics.mean_absolute_error(y_test_cured, y_pred_cured))
print('Root Mean Squared Error (Cured):', np.sqrt(metrics.mean_squared_error(y_test_cured, y_pred_cured)))
r_squared = metrics.r2_score(y_test_cured, y_pred_cured)
print('R-squared:', r_squared)
```

```
<ipython-input-6-f5353814cab6>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  data['ConfirmedIndianNational'] = pd.to_numeric(data['ConfirmedIndianNational'], er
<ipython-input-6-f5353814cab6>:19: UserWarning: Parsing dates in DD/MM/YYYY format wh
  data['Date'] = pd.to_numeric(pd.to_datetime(data['Date']))
```




```
Mean Absolute Error (Cured): 1.2511001759027949
Root Mean Squared Error (Cured): 2.4535729173565337
R-squared: 0.39275015576136785
```

```
pip install pysal
```

```
Collecting pysal
  Downloading pysal-23.7-py3-none-any.whl (17 kB)
Collecting libpysal>=4.6.2 (from pysal)
  Downloading libpysal-4.9.2-py3-none-any.whl (2.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 22.0 MB/s eta 0:00:00
Collecting access>=1.1.8 (from pysal)
  Downloading access-1.1.9-py3-none-any.whl (21 kB)
Collecting esda>=2.4.1 (from pysal)
  Downloading esda-2.5.1-py3-none-any.whl (132 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.4/132.4 kB 18.3 MB/s eta 0:00:00
Collecting giddy>=2.3.3 (from pysal)
  Downloading giddy-2.3.4-py3-none-any.whl (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.1/61.1 kB 7.8 MB/s eta 0:00:00
Collecting inequality>=1.0.0 (from pysal)
  Downloading inequality-1.0.1-py3-none-any.whl (15 kB)
Collecting pointpats>=2.2.0 (from pysal)
  Downloading pointpats-2.4.0-py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.4/58.4 kB 8.2 MB/s eta 0:00:00
Collecting segregation>=2.3.1 (from pysal)
  Downloading segregation-2.5-py3-none-any.whl (141 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 141.3/141.3 kB 17.6 MB/s eta 0:00:00
Collecting spaghetti>=1.6.6 (from pysal)
  Downloading spaghetti-1.7.4-py3-none-any.whl (50 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.1/50.1 kB 7.5 MB/s eta 0:00:00
Collecting mgwr>=2.1.2 (from pysal)
  Downloading mgwr-2.2.1-py3-none-any.whl (47 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.9/47.9 kB 6.8 MB/s eta 0:00:00
Collecting momepy>=0.5.3 (from pysal)
  Downloading momepy-0.7.0-py3-none-any.whl (277 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 277.8/277.8 kB 27.5 MB/s eta 0:00:00
Collecting spglm>=1.0.8 (from pysal)
  Downloading spglm-1.1.0-py3-none-any.whl (41 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.4/41.4 kB 3.4 MB/s eta 0:00:00
Collecting spint>=1.0.7 (from pysal)
  Downloading spint-1.0.7.tar.gz (28 kB)
  Preparing metadata (setup.py) ... done
Collecting spreg>=1.2.4 (from pysal)
  Downloading spreg-1.4.2-py3-none-any.whl (331 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 331.8/331.8 kB 24.7 MB/s eta 0:00:00
Collecting spvcm>=0.3.0 (from pysal)
  Downloading spvcm-0.3.0.tar.gz (5.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.7/5.7 MB 45.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting tobler>=0.8.2 (from pysal)
  Downloading tobler-0.11.2-py3-none-any.whl (34 kB)
Collecting mapclassify>=2.4.3 (from pysal)
  Downloading mapclassify-2.6.1-py3-none-any.whl (38 kB)
```

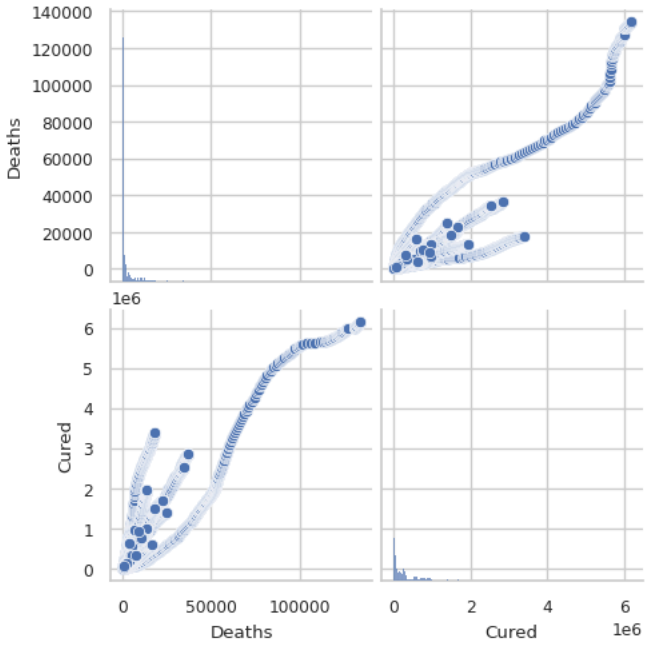
```
Collecting splot>=1.1.5.post1 (from pysal)
  Downloading splot-1.1.5.post1-py3-none-any.whl (39 kB)
Collecting spopt>=0.4.1 (from pysal)
  Downloading spopt-0.6.0-py3-none-any.whl (244 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 244.5/244.5 kB 37.0 MB/s eta 0:00:00
Requirement already satisfied: geopandas in /usr/local/lib/python3.10/dist-packages (from access>=1.1.8->pysal) (0.13.2)
Requirement already satisfied: numpy>=1.3 in /usr/local/lib/python3.10/dist-packages (from access>=1.1.8->pysal) (1.23.5)
Requirement already satisfied: pandas>=0.23.4 in /usr/local/lib/python3.10/dist-packages (from access>=1.1.8->pysal) (1.5.3)
Requirement already satisfied: requests>=2 in /usr/local/lib/python3.10/dist-packages (from access>=1.1.8->pysal) (2.31.0)
```

```
pip install geopandas matplotlib esda spreg
```

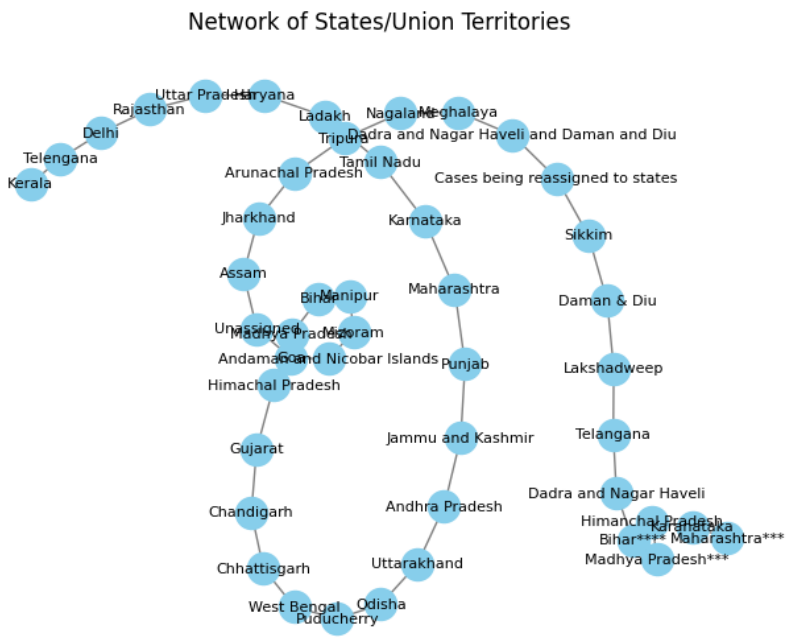
```
Requirement already satisfied: geopandas in /usr/local/lib/python3.10/dist-packages (0.13.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Collecting esda
  Downloading esda-2.5.1-py3-none-any.whl (132 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.4/132.4 kB 2.9 MB/s eta 0:00:00
Collecting spreg
  Downloading spreg-1.4.2-py3-none-any.whl (331 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 331.8/331.8 kB 9.9 MB/s eta 0:00:00
Requirement already satisfied: fiona>=1.8.19 in /usr/local/lib/python3.10/dist-packages (from geopandas) (1.9.5)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from geopandas) (23.2)
Requirement already satisfied: pandas>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from geopandas) (1.5.3)
Requirement already satisfied: pyproj>=3.0.1 in /usr/local/lib/python3.10/dist-packages (from geopandas) (3.6.1)
Requirement already satisfied: shapely>=1.7.1 in /usr/local/lib/python3.10/dist-packages (from geopandas) (2.0.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.47.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.23.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Collecting libpysal (from esda)
  Downloading libpysal-4.9.2-py3-none-any.whl (2.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 20.5 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn>=1.0 in /usr/local/lib/python3.10/dist-packages (from esda) (1.2.2)
Requirement already satisfied: scipy>=1.9 in /usr/local/lib/python3.10/dist-packages (from esda) (1.11.4)
Requirement already satisfied: attrs>=19.2.0 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (23.2.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (2023.11.17)
Requirement already satisfied: click~=8.0 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (8.1.7)
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (1.1.1
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (0.7.2)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (1.16.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (67.7.2)
Requirement already satisfied: beautifulsoup4>=4.10 in /usr/local/lib/python3.10/dist-packages (from libpysal->esda) (4.11.2)
Requirement already satisfied: platformdirs>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from libpysal->esda) (4.1.0)
Requirement already satisfied: requests>=2.27 in /usr/local/lib/python3.10/dist-packages (from libpysal->esda) (2.31.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.0->geopandas) (2023.3.post1
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0->esda) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0->esda) (3.2.0
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4>=4.10->libpysal->esda)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.27->libpysal->e
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.27->libpysal->esda) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.27->libpysal->esda)
Installing collected packages: libpysal, spreg, esda
Successfully installed esda-2.5.1 libpysal-4.9.2 spreg-1.4.2
```

```
sns.pairplot(data)
```

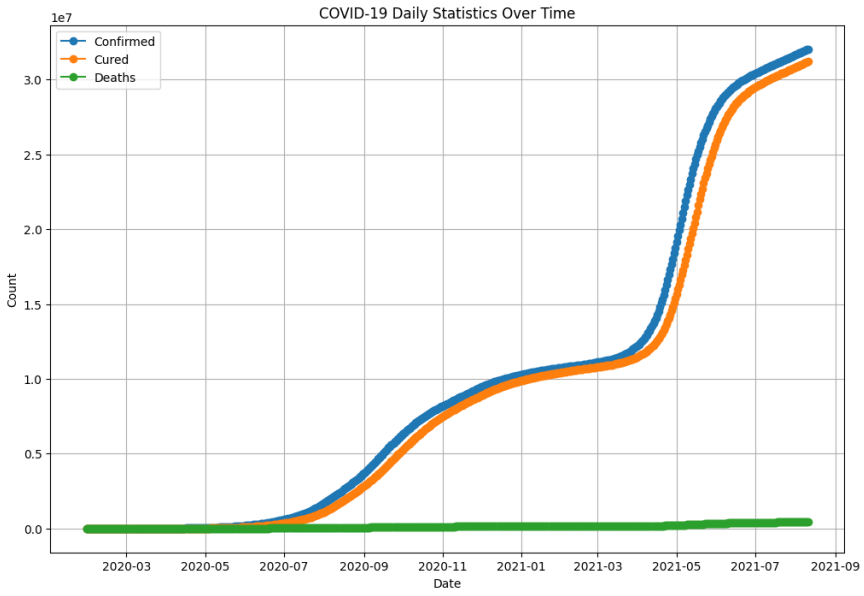
```
<seaborn.axisgrid.PairGrid at 0x7b36f3297760>
```



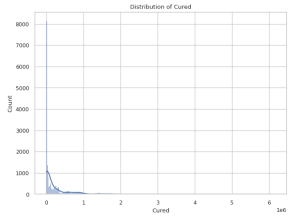

```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt

# Load the COVID-19 data from a CSV file into a DataFrame
covid_data = pd.read_csv('/content/covid_19_india (2).csv')

# Assuming you have a DataFrame named 'covid_data' with columns 'State/UnionTerritory', 'Deaths', 'Cured', 'Latitude', 'Longitude'
# Replace 'Region' with the actual column name that identifies different regions
covid_data = covid_data[['State/UnionTerritory', 'Deaths', 'Cured', 'Latitude', 'Longitude']]

# Create a graph
G = nx.Graph()

# Add nodes to the graph (regions)
for state in covid_data['State/UnionTerritory']:
    G.add_node(state)

# Add edges to the graph based on some criteria (e.g., spatial proximity, similar COVID-19 trends)
# For simplicity, let's connect states that are adjacent to each other alphabetically
states = covid_data['State/UnionTerritory'].unique()
for i in range(len(states)-1):
    G.add_edge(states[i], states[i+1])

# Draw the network
pos = nx.spring_layout(G)  # You can choose different layout algorithms based on your preference
nx.draw(G, pos, with_labels=True, font_size=8, node_size=300, font_color='black', node_color='skyblue', edge_color='gray')
plt.title('Network of States/Union Territories')
plt.show()
```

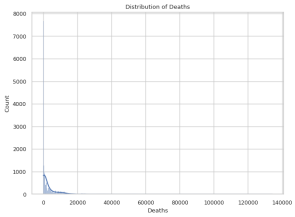


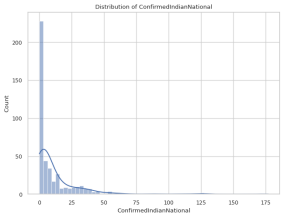

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the COVID-19 data from a CSV file into a DataFrame
covid_data = pd.read_csv('/content/covid_19_india (2).csv')

# Assuming you have a DataFrame named 'covid_data' with columns 'Date', 'State/UnionTerritory', 'Confirmed', 'Cured', 'Deaths'

# Convert 'Date' column to datetime format
covid_data['Date'] = pd.to_datetime(covid_data['Date'], format='%d-%m-%Y')  # Adjust the format based on your data

# Group data by date to get daily statistics
daily_stats = covid_data.groupby('Date').agg({
    'Confirmed': 'sum',
    'Cured': 'sum',
    'Deaths': 'sum'
}).reset_index()

# Plot daily statistics over time
plt.figure(figsize=(12, 8))

plt.plot(daily_stats['Date'], daily_stats['Confirmed'], label='Confirmed', marker='o')
plt.plot(daily_stats['Date'], daily_stats['Cured'], label='Cured', marker='o')
plt.plot(daily_stats['Date'], daily_stats['Deaths'], label='Deaths', marker='o')

plt.xlabel('Date')
plt.ylabel('Count')
plt.title('COVID-19 Daily Statistics Over Time')
plt.legend()
plt.grid(True)
plt.show()
```

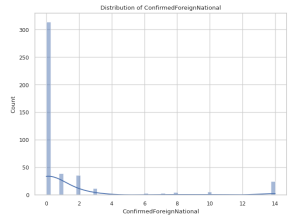



```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the COVID-19 data from a CSV file into a DataFrame
covid_data = pd.read_csv('/content/covid_19_india (2).csv')

# Assuming you have a DataFrame named 'covid_data' with numerical columns
# Replace non-numeric values (e.g., '-') with NaN
covid_data = covid_data.apply(pd.to_numeric, errors='coerce')

# Select relevant columns for distribution plots
columns_of_interest = ['Cured', 'Deaths', 'ConfirmedIndianNational', 'ConfirmedForeignNational']
selected_data = covid_data[columns_of_interest]

# Calculate the number of rows and columns for the grid
num_cols = len(selected_data.columns)
num_rows = (num_cols - 1) // 3 + 1

# Display distribution plots for selected columns
sns.set(style="whitegrid", font_scale=0.8)
plt.figure(figsize=(20, 5 * num_rows))

# Iterate through each selected column and create a distribution plot
for i, column in enumerate(selected_data.columns):
    plt.subplot(num_rows, 3, i + 1)
    sns.histplot(selected_data[column], kde=True)
    plt.title(f'Distribution of {column}')

plt.tight_layout()
plt.show()
```

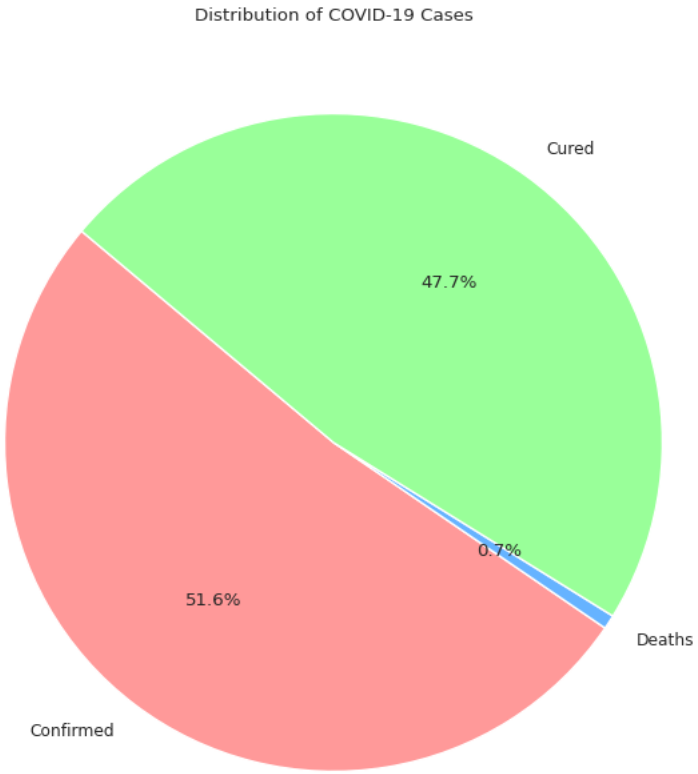








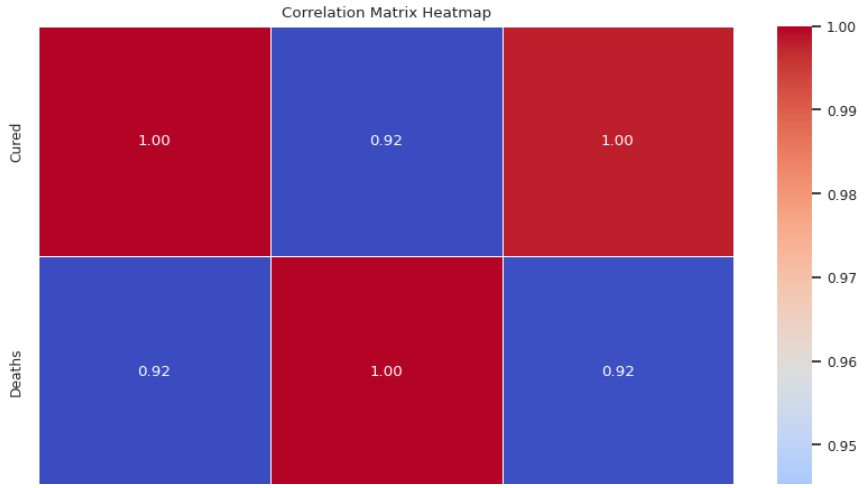

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have a COVID-19 dataset in a CSV file, load it into a Pandas DataFrame
# Replace 'your_dataset.csv' with the actual file name or URL of your dataset
covid_data = pd.read_csv('/content/covid_19_india (2).csv')

# Assuming your dataset has columns like 'Confirmed', 'Deaths', 'Recovered'
# Replace these with the actual column names in your dataset
cases_data = covid_data[['Confirmed', 'Deaths', 'Cured']].sum()

# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(cases_data, labels=cases_data.index, autopct='%1.1f%%', startangle=140, colors=['#ff9999', '#66b3ff', '#99ff99'])
plt.title('Distribution of COVID-19 Cases')
plt.show()
```




```
pip install libpysal
```

```
Requirement already satisfied: libpysal in /usr/local/lib/python3.10/dist-packages (4.9.2)
Requirement already satisfied: beautifulsoup4>=4.10 in /usr/local/lib/python3.10/dist-packages (from libpysal) (4.11.2)
Requirement already satisfied: geopandas>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from libpysal) (0.13.2)
Requirement already satisfied: numpy>=1.22 in /usr/local/lib/python3.10/dist-packages (from libpysal) (1.23.5)
Requirement already satisfied: packaging>=22 in /usr/local/lib/python3.10/dist-packages (from libpysal) (23.2)
Requirement already satisfied: pandas>=1.4 in /usr/local/lib/python3.10/dist-packages (from libpysal) (1.5.3)
Requirement already satisfied: platformdirs>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from libpysal) (4.1.0)
Requirement already satisfied: requests>=2.27 in /usr/local/lib/python3.10/dist-packages (from libpysal) (2.31.0)
Requirement already satisfied: scipy>=1.8 in /usr/local/lib/python3.10/dist-packages (from libpysal) (1.11.4)
Requirement already satisfied: shapely>=2.0.1 in /usr/local/lib/python3.10/dist-packages (from libpysal) (2.0.2)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4>=4.10->libpysal) (2.5)
Requirement already satisfied: fiona>=1.8.19 in /usr/local/lib/python3.10/dist-packages (from geopandas>=0.10.0->libpysal) (1.9.5)
Requirement already satisfied: pyproj>=3.0.1 in /usr/local/lib/python3.10/dist-packages (from geopandas>=0.10.0->libpysal) (3.6.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.4->libpysal) (2.8.2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.4->libpysal) (2023.3.post1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.27->libpysal)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.27->libpysal) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.27->libpysal) (2.0.7
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.27->libpysal) (2023.1
Requirement already satisfied: attrs>=19.2.0 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas>=0.10.0->libp
Requirement already satisfied: click~=8.0 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas>=0.10.0->libpysa
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas>=0.10.0
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas>=0.10.0->libpysa
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas>=0.10.0->libpysal) (1.1
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas>=0.10.0->libpysa
```

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

# Assuming you have a COVID-19 dataset in a CSV file, load it into a Pandas DataFrame
# Replace 'your_dataset.csv' with the actual file name or URL of your dataset
covid_data = pd.read_csv('/content/covid_19_india (2).csv')

# Assuming you want to analyze correlations between different numerical columns
# Replace 'column1', 'column2', etc., with the actual column names you are interested in
columns_of_interest = ['Cured', 'Deaths', 'Confirmed']

# Create a subset of the DataFrame with only the columns of interest
subset_data = covid_data[columns_of_interest]

# Calculate the correlation matrix
correlation_matrix = subset_data.corr()

# Create a heatmap using seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```




```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

# Load the COVID-19 data from a CSV file into a DataFrame
covid_data = pd.read_csv('/content/covid_19_india (2) (1).csv')

# Assuming you have a DataFrame named 'covid_data' with columns 'State/UnionTerritory', 'Deaths', 'Cured', 'Latitude', 'Longitude'
# Replace 'Region' with the actual column name that identifies different regions
covid_data = covid_data[['State/UnionTerritory', 'Deaths', 'Cured', 'Latitude', 'Longitude']]

# Replace non-numeric values (e.g., '-') with NaN
covid_data[['Deaths', 'Cured']] = covid_data[['Deaths', 'Cured']].apply(pd.to_numeric, errors='coerce')

# Fill null values with zero
covid_data.fillna(0, inplace=True)

# Create a GeoDataFrame for spatial plotting
geometry = gpd.points_from_xy(covid_data['Longitude'], covid_data['Latitude'])
gdf = gpd.GeoDataFrame(covid_data, geometry=geometry)

# India map shapefile (replace 'path_to_your_shapefile' with the actual path)
india_map = gpd.read_file('/content/india_st.shx')
```