Phase 4:
COVID_19_CASE ANALYTICS

Model Building:

Clustering Analysis:

Use unsupervised learning techniques like K-Means clustering or DBSCAN to group your data into clusters based on the available features (Day, Month, Deaths). This can help identify patterns or similarities in covid_19_case.

Importing Libraries:

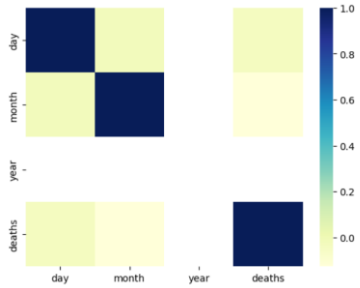
The code begins by importing the necessary Python libraries, including Pandas for data handling, NumPy for numerical operations, Scikit-Learn for machine learning, and Matplotlib for data visualization.

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

Feature Selection:

The code selects the features (independent variables) to be used for clustering, which are 'Day,' 'Month,' and 'Deathes.' These features will be used to determine the clusters.

```
import seaborn as sns
sns.heatmap(data.corr(),cmap='YlGnBu')
```




```
X = data[['Day', 'Month', 'Deaths']]
```

Feature Standardization:

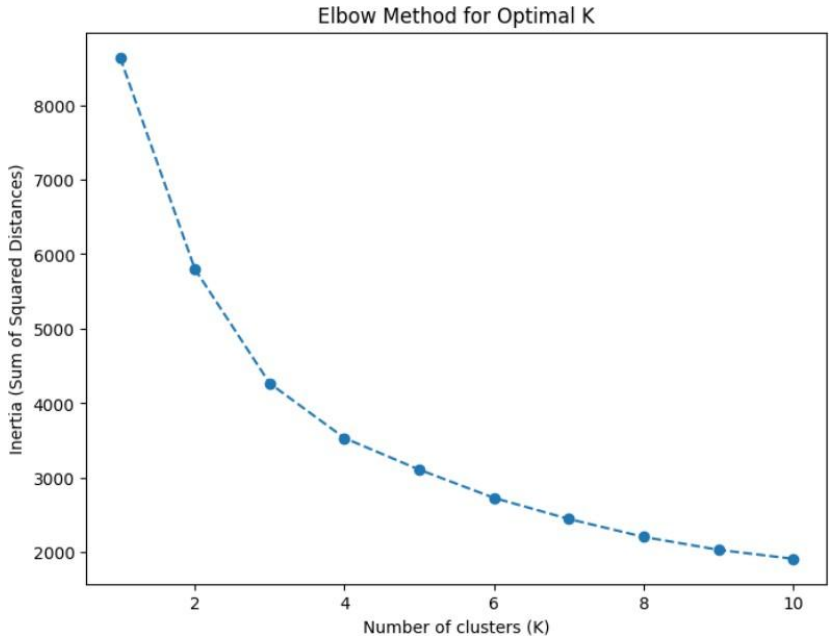
The features are standardized using the StandardScaler from Scikit-Learn. Standardization ensures that all features have a mean of 0 and a standard deviation of 1, which is important for K-Means clustering.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=0).fit(X)
    inertia.append(kmeans.inertia_)
```

Determine the Optimal Number of Clusters:

The code then uses the Elbow method to find the optimal number of clusters (K). It iterates through different values of K and calculates the inertia, which is the sum of squared distances from data points to their assigned cluster centers. The Elbow method plots these inertias for various K values to help you identify the "elbow point" where increasing K doesn't significantly reduce the inertia.

```
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), inertia, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of clusters (K)')
plt.ylabel('Inertia (Sum of Squared Distances)')
plt.show()
```



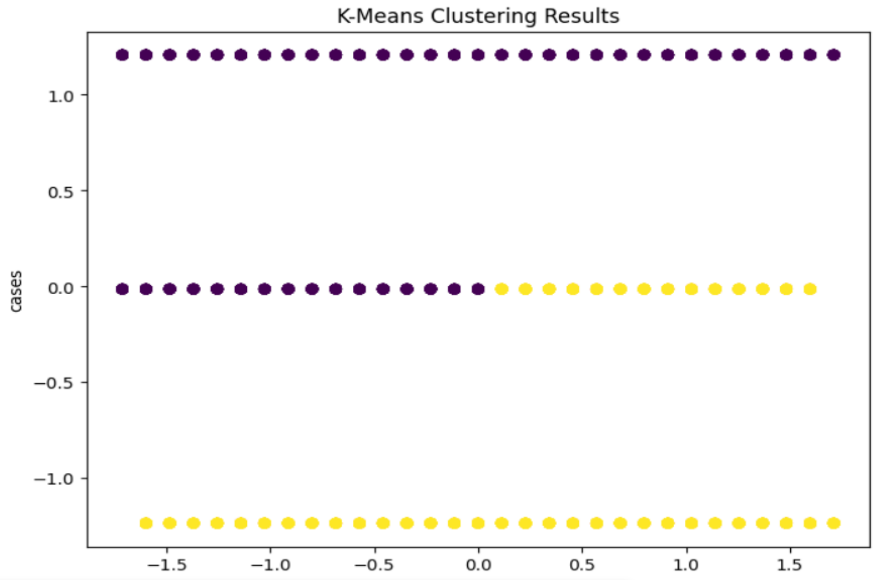
K-Means Clustering:

After determining the optimal K (in this case, K = 3), the code performs K-Means clustering using the KMeans algorithm from Scikit-Learn. The clusters are assigned to the 'Cluster' column in the dataset.

```
kmeans = KMeans(n_clusters=2, random_state=0)
data['Air Quality'] = kmeans.fit_predict(X)
```

```
0       0
1       0
2       0
3       0
4       0
       ..
2725    1
2726    1
2727    1
2728    1
2729    1
Name: covid_19, Length: 2730, dtype: int32
```

```
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=data['covid_19'], cmap='viridis')
plt.title('K-Means Clustering Results')
plt.xlabel('Deaths')
plt.ylabel('Cases')
plt.show()
```

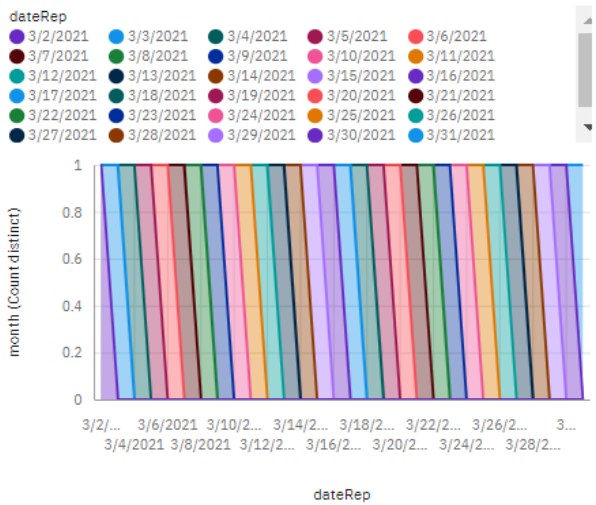


Visualization and Insights:

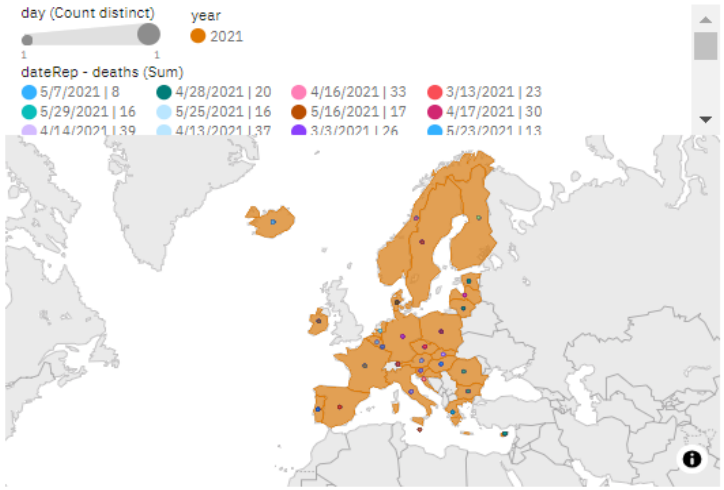
1) The visualizations presented in this analysis provide valuable insights into the trajectory and impact of the COVID-19 pandemic
2) The heatmap visualizes the correlation between various variables in the COVID-19 case analysis dataset.
3) The scatter plot shows distinct clusters of data points based on the number of COVID-19 cases and deaths. This suggests that K-Means has successfully grouped regions with similar COVID-19 statistics into clusters.
4) In conclusion, the K-Means clustering analysis provides a useful method for grouping regions based on COVID-19 cases and deaths.
5) In the code snippet, a new data frame named "new_data" has been created. This data frame consists of three columns: "day," "month," and "deaths," each containing three numerical values.

Tab 1 +

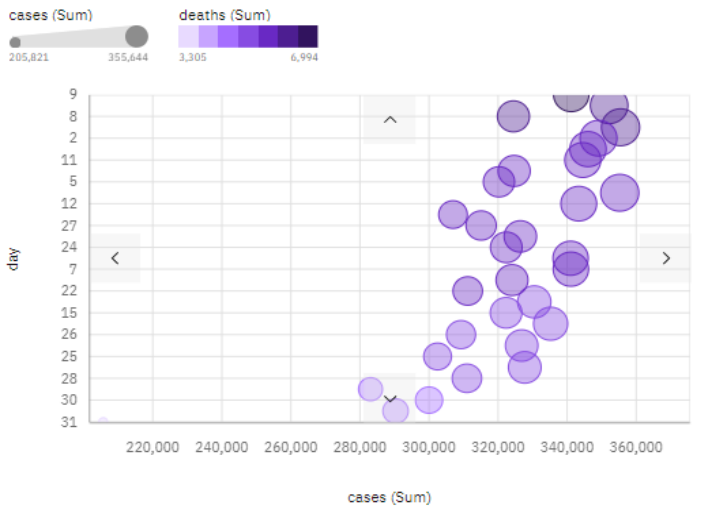
month by dateRep colored by dateRep



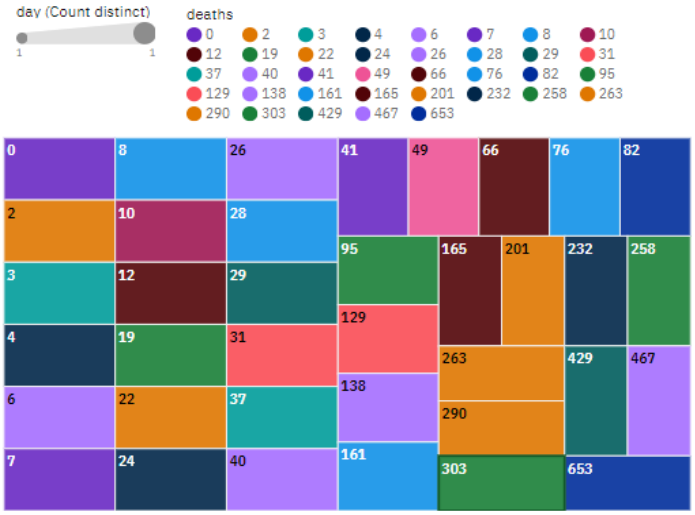
year for countriesAndTerritories regions, dateRep and deaths and day for countriesAndTerritories points



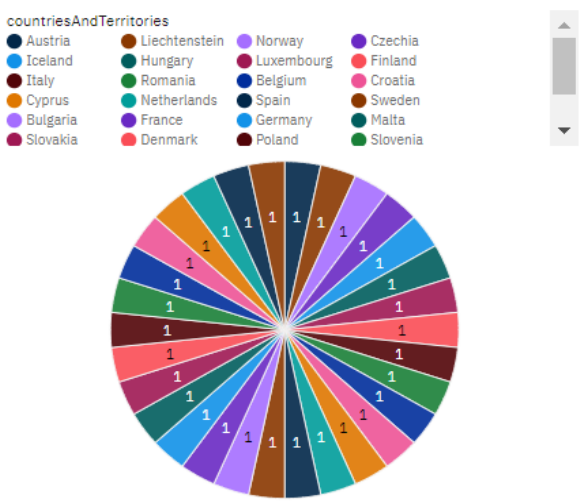
cases by day colored by deaths and sized by cases



day for deaths hierarchy



year by countriesAndTerritories



dateRep by deaths