```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
data = pd.read_csv("/content/covid_19.csv")
print("Dataset Loaded Successfully!")
print("Shape of Data:", data.shape)
```

```
Dataset Loaded Successfully!
Shape of Data: (238, 9)
```

```
print("\nChecking Missing Values:")
print(data.isnull().sum())
```

```
Checking Missing Values:
country        0
continent      2
population     9
day            0
time           0
Cases          0
Recovered     48
Deaths         5
Tests         25
dtype: int64
```

```
data = data.fillna(0)
```

```
data = data.drop_duplicates()
print("Shape after cleaning:", data.shape)
```

```
Shape after cleaning: (238, 9)
```

```
print("\nFirst 5 Rows:")
print(data.head())
```

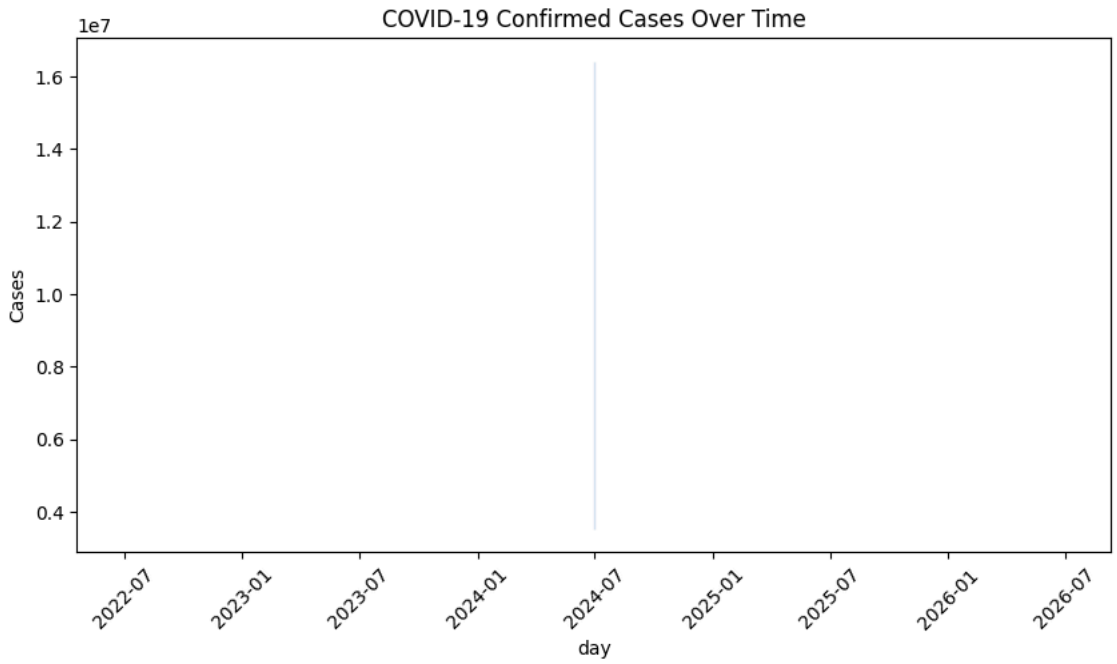
```
First 5 Rows:
            country      continent  population         day  \
0      Saint-Helena         Africa      6115.0  2024-06-30
1  Falkland-Islands  South-America      3539.0  2024-06-30
2        Montserrat  North-America      4965.0  2024-06-30
3  Diamond-Princess              0         0.0  2024-06-30
4      Vatican-City         Europe       799.0  2024-06-30

                       time  Cases  Recovered  Deaths    Tests
0  2024-06-30T16:15:16+00:00   2166        2.0     0.0      0.0
1  2024-06-30T16:15:16+00:00   1930     1930.0     0.0   8632.0
2  2024-06-30T16:15:16+00:00   1403     1376.0     8.0  17762.0
3  2024-06-30T16:15:16+00:00    712      699.0    13.0      0.0
4  2024-06-30T16:15:16+00:00     29       29.0     0.0      0.0
```

```
print("\nStatistical Summary:")
print(data.describe())
```

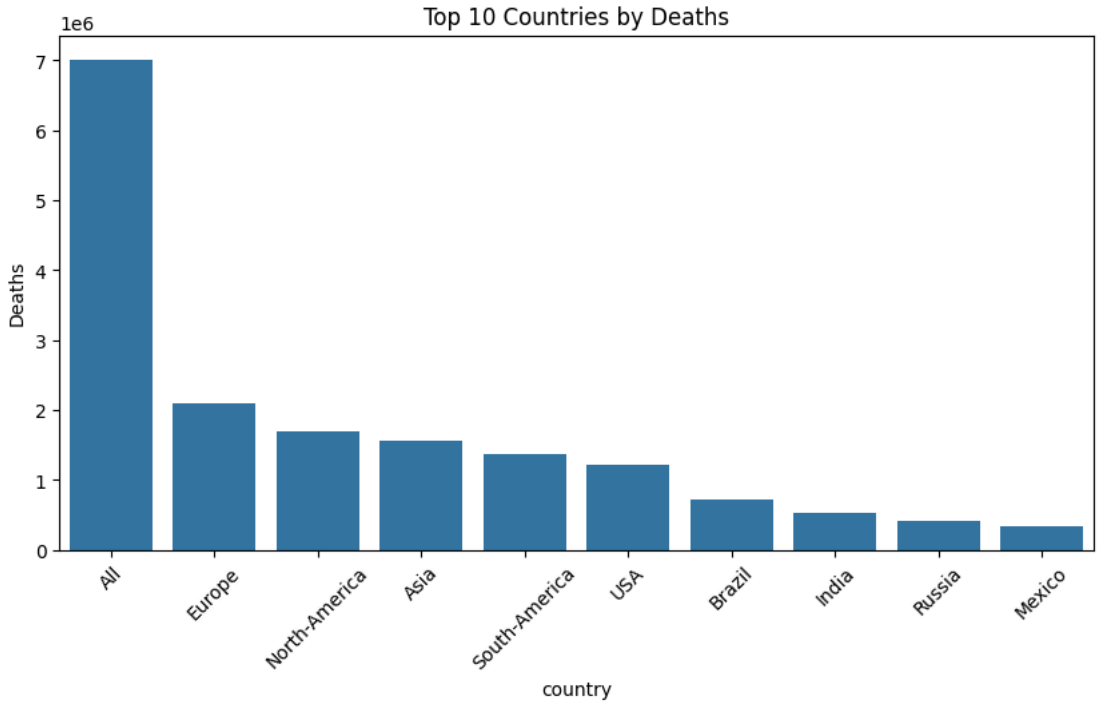
```
Statistical Summary:
         population         Cases     Recovered        Deaths         Tests
count  2.380000e+02  2.380000e+02  2.380000e+02  2.380000e+02  2.380000e+02
mean   3.338208e+07  8.883449e+06  8.032801e+06  8.836987e+04  2.952313e+07
std    1.361412e+08  5.193031e+07  4.977610e+07  5.110324e+05  1.138312e+08
min    0.000000e+00  9.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
25%    3.277222e+05  2.741875e+04  2.717500e+03  1.902500e+02  1.840850e+05
50%    5.334767e+06  2.320425e+05  6.530850e+04  2.267000e+03  1.551728e+06
75%    2.115053e+07  1.565481e+06  1.238110e+06  1.707325e+04  1.152990e+07
max    1.448471e+09  7.047539e+08  6.756198e+08  7.010681e+06  1.186852e+09
```

```
plt.figure(figsize=(10,5))
data['day'] = pd.to_datetime(data['day']) # Convert 'day' to datetime
```

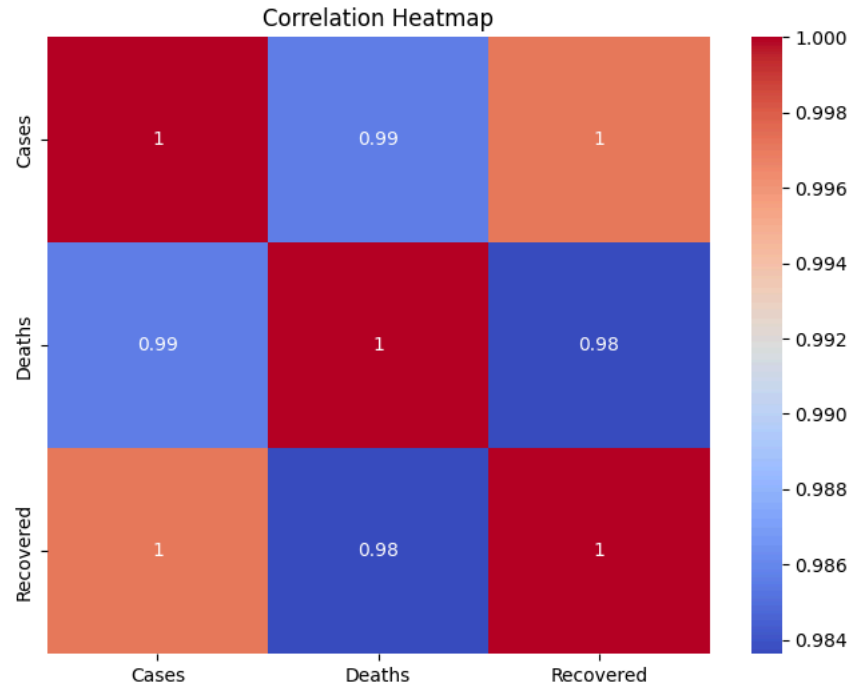
```
sns.lineplot(x="day", y="Cases", data=data)
plt.title("COVID-19 Confirmed Cases Over Time")
plt.xticks(rotation=45)
plt.show()
```




```
plt.figure(figsize=(10,5))
sns.barplot(x="country", y="Deaths", data=data.sort_values("Deaths", ascending=False).head(10))
plt.title("Top 10 Countries by Deaths")
plt.xticks(rotation=45)
plt.show()
```




```
import numpy as np
plt.figure(figsize=(8,6))
# Select only numeric columns for correlation calculation and include 'Cases', 'Deaths', and 'Recovered'
numeric_data = data[['Cases', 'Deaths', 'Recovered']]
sns.heatmap(numeric_data.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```




```
# Select features (excluding 'Deaths' as it's the target) and target
numeric_data = data.select_dtypes(include=np.number)
X = numeric_data.drop("Deaths", axis=1)
y = numeric_data["Deaths"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Shape of X:", X_scaled.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (238, 4)
Shape of y: (238,)
```

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

```
model = DecisionTreeClassifier(max_depth=5, random_state=42)
model.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, random_state=42)
```

```
y_pred = model.predict(X_test)
```

```
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
     1408.0       0.00      0.00      0.00         1
     1427.0       0.00      0.00      0.00         1
     1637.0       0.00      0.00      0.00         1
     2024.0       0.00      0.00      0.00         1
     2284.0       0.00      0.00      0.00         1
     2686.0       0.00      0.00      0.00         1
     6437.0       0.00      0.00      0.00         1
     6638.0       0.00      0.00      0.00         1
     8727.0       0.00      0.00      0.00         1
     9428.0       0.00      0.00      0.00         1
    12031.0       0.00      0.00      0.00         1
    13848.0       0.00      0.00      0.00         1
    16303.0       0.00      0.00      0.00         1
    18057.0       0.00      0.00      0.00         1
    19495.0       0.00      0.00      0.00         1
    20289.0       0.00      0.00      0.00         1
    28126.0       0.00      0.00      0.00         1
    43517.0       0.00      0.00      0.00         1
    66864.0       0.00      0.00      0.00         1
   143200.0       0.00      0.00      0.00         1
   167642.0       0.00      0.00      0.00         1
   183027.0       0.00      0.00      0.00         1
   334958.0       0.00      0.00      0.00         1
  1219487.0       0.00      0.00      0.00         1
  1695941.0       0.00      0.00      0.00         1

   accuracy                           0.02        48
  macro avg       0.01      0.02      0.01        48
weighted avg      0.01      0.02      0.01        48

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined an
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined an
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined an
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.show()
```