COVID 19 Analysis

In [7]:

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

In [8]:

```
# Load the dataset

data = pd.read_csv("D:/Data Analysis Project/Corona Virus Dataset.csv")
```

In [9]:

```
# Displaying first 10 rows
data.head(10)
```

Out[9]:

	Province	Country/Region	Latitude	Longitude	Date	Confirmed	Deaths	Recovered
0	Afghanistan	Afghanistan	33.93911	67.709953	22-01-2020	0.0	0.0	0.0
1	Afghanistan	Afghanistan	33.93911	67.709953	23-01-2020	0.0	0.0	0.0
2	Afghanistan	Afghanistan	33.93911	67.709953	24-01-2020	0.0	0.0	0.0
3	Afghanistan	Afghanistan	33.93911	67.709953	25-01-2020	0.0	0.0	0.0
4	Afghanistan	Afghanistan	33.93911	67.709953	26-01-2020	0.0	0.0	0.0
5	Afghanistan	Afghanistan	33.93911	67.709953	27-01-2020	0.0	0.0	0.0
6	Afghanistan	Afghanistan	33.93911	67.709953	28-01-2020	0.0	0.0	0.0
7	Afghanistan	Afghanistan	33.93911	67.709953	29-01-2020	0.0	0.0	0.0
8	Afghanistan	Afghanistan	33.93911	67.709953	30-01-2020	0.0	0.0	0.0
9	Afghanistan	Afghanistan	33.93911	67.709953	31-01-2020	0.0	0.0	0.0

In [10]:

```
# Displaying last 10 rows
data.tail(10)
```

Out[10]:

	Province	Country/Region	Latitude	Longitude	Date	Confirmed	Deaths	Recovered
78376	Zimbabwe	Zimbabwe	-19.015438	29.154857	04-06-2021	52.0	1.0	10.0
78377	Zimbabwe	Zimbabwe	-19.015438	29.154857	05-06-2021	24.0	0.0	8.0
78378	Zimbabwe	Zimbabwe	-19.015438	29.154857	06-06-2021	21.0	1.0	30.0
78379	Zimbabwe	Zimbabwe	-19.015438	29.154857	07-06-2021	49.0	5.0	18.0
78380	Zimbabwe	Zimbabwe	-19.015438	29.154857	08-06-2021	83.0	6.0	10.0
78381	Zimbabwe	Zimbabwe	-19.015438	29.154857	09-06-2021	111.0	5.0	161.0
78382	Zimbabwe	Zimbabwe	-19.015438	29.154857	10-06-2021	64.0	4.0	23.0
78383	Zimbabwe	Zimbabwe	-19.015438	29.154857	11-06-2021	192.0	3.0	30.0
78384	Zimbabwe	Zimbabwe	-19.015438	29.154857	12-06-2021	164.0	3.0	22.0
78385	Zimbabwe	Zimbabwe	-19.015438	29.154857	13-06-2021	107.0	0.0	12.0

In [11]:

```
# Information about the dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 78386 entries, 0 to 78385
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Province        78386 non-null  object 
 1   Country/Region  78386 non-null  object 
 2   Latitude        78386 non-null  float64
 3   Longitude       78386 non-null  float64
 4   Date            78386 non-null  object 
 5   Confirmed       78366 non-null  float64
 6   Deaths          78379 non-null  float64
 7   Recovered       78378 non-null  float64
dtypes: float64(5), object(3)
memory usage: 4.8+ MB
```

In [12]:

```
# Changing datatype of Date column to Date Time
data['Date'] = pd.to_datetime(data['Date'])
```

```
C:\Users\ROHITH DP\AppData\Local\Temp\ipykernel_2376\4069544646.py:2: UserWarning:
Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specifie
d. This may lead to inconsistently parsed dates! Specify a format to ensure consis
tent parsing.
  data['Date'] = pd.to_datetime(data['Date'])
```

In [13]:

```
# Checking the datatype whether changed or not for the Date column
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 78386 entries, 0 to 78385
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   Province        78386 non-null  object        
 1   Country/Region  78386 non-null  object        
 2   Latitude        78386 non-null  float64       
 3   Longitude       78386 non-null  float64       
 4   Date            78386 non-null  datetime64[ns]
 5   Confirmed       78366 non-null  float64       
 6   Deaths          78379 non-null  float64       
 7   Recovered       78378 non-null  float64       
dtypes: datetime64[ns](1), float64(5), object(2)
memory usage: 4.8+ MB
```

In [14]:

```
# Checking null values
data.isnull().sum()
```

Out[14]:

```
Province           0
Country/Region     0
Latitude           0
Longitude          0
Date               0
Confirmed         20
Deaths             7
Recovered          8
dtype: int64
```

In [15]:

```
# Fill 0 for empty values
data.fillna(0,inplace=True)
```

In [16]:

```
# Checking null values after handling the empty values
data.isnull().sum()
```

Out[16]:

```
Province          0
Country/Region    0
Latitude          0
Longitude         0
Date              0
Confirmed         0
Deaths            0
Recovered         0
dtype: int64
```

In [17]:

```
# Size of the columns and Rows
data.shape
```

Out[17]:

```
(78386, 8)
```

In [18]:

```
# Displays the columns name
data.columns
```

Out[18]:

```
Index(['Province', 'Country/Region', 'Latitude', 'Longitude', 'Date',
       'Confirmed', 'Deaths', 'Recovered'],
      dtype='object')
```

In [19]:

```
# Dropping unwanted columns
data.drop(columns=['Province'],inplace = True)
```

In [20]:

```
# Displays the columns name after dropping the unwanted columns
data.columns
```

Out[20]:

```
Index(['Country/Region', 'Latitude', 'Longitude', 'Date', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
```

In [21]:

```
# Checking information about the data after preprocessing
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 78386 entries, 0 to 78385
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   Country/Region  78386 non-null  object        
 1   Latitude        78386 non-null  float64       
 2   Longitude       78386 non-null  float64       
 3   Date            78386 non-null  datetime64[ns]
 4   Confirmed       78386 non-null  float64       
 5   Deaths          78386 non-null  float64       
 6   Recovered       78386 non-null  float64       
dtypes: datetime64[ns](1), float64(5), object(1)
memory usage: 4.2+ MB
```

In [22]:

```
# Again displays the first 5 rows
data.head()
```

Out[22]:

	Country/Region	Latitude	Longitude	Date	Confirmed	Deaths	Recovered
0	Afghanistan	33.93911	67.709953	2020-01-22	0.0	0.0	0.0
1	Afghanistan	33.93911	67.709953	2020-01-23	0.0	0.0	0.0
2	Afghanistan	33.93911	67.709953	2020-01-24	0.0	0.0	0.0
3	Afghanistan	33.93911	67.709953	2020-01-25	0.0	0.0	0.0
4	Afghanistan	33.93911	67.709953	2020-01-26	0.0	0.0	0.0

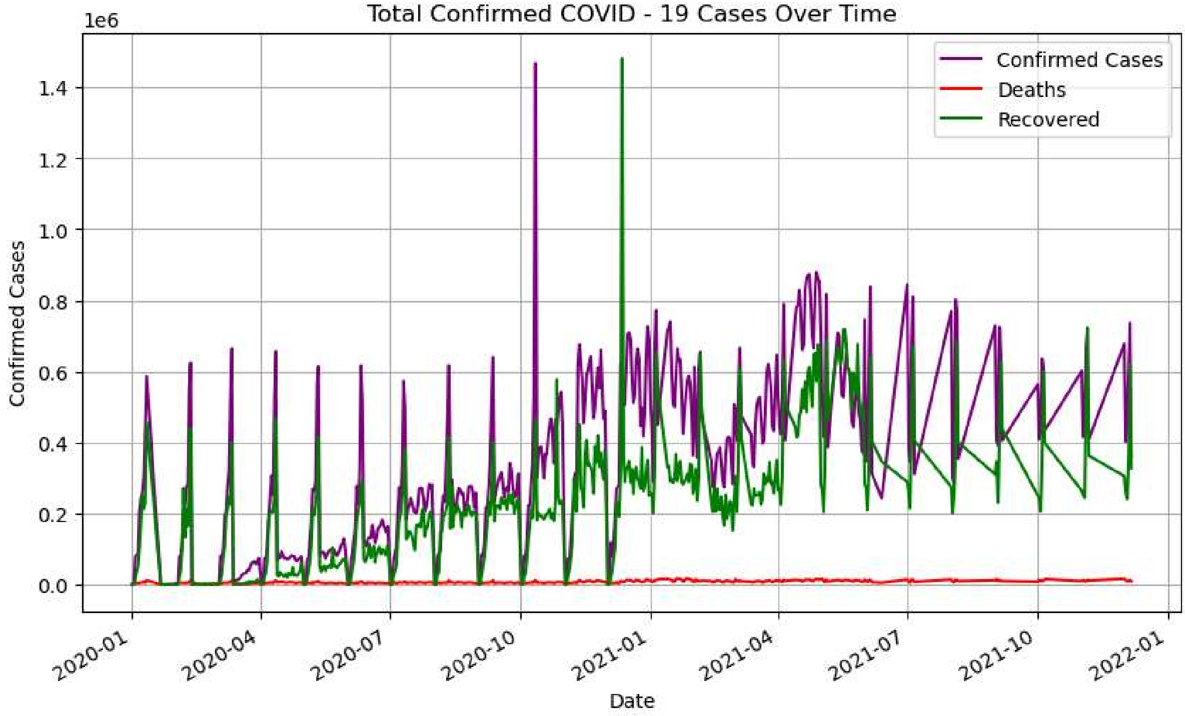
Exploratory Data Analysis (EDA)

In [23]:

```
# Group by Dates
confirmed_over_time = data.groupby('Date')['Confirmed'].sum()
deaths_over_time = data.groupby('Date')['Deaths'].sum()
recovered_over_time = data.groupby('Date')['Recovered'].sum()
```

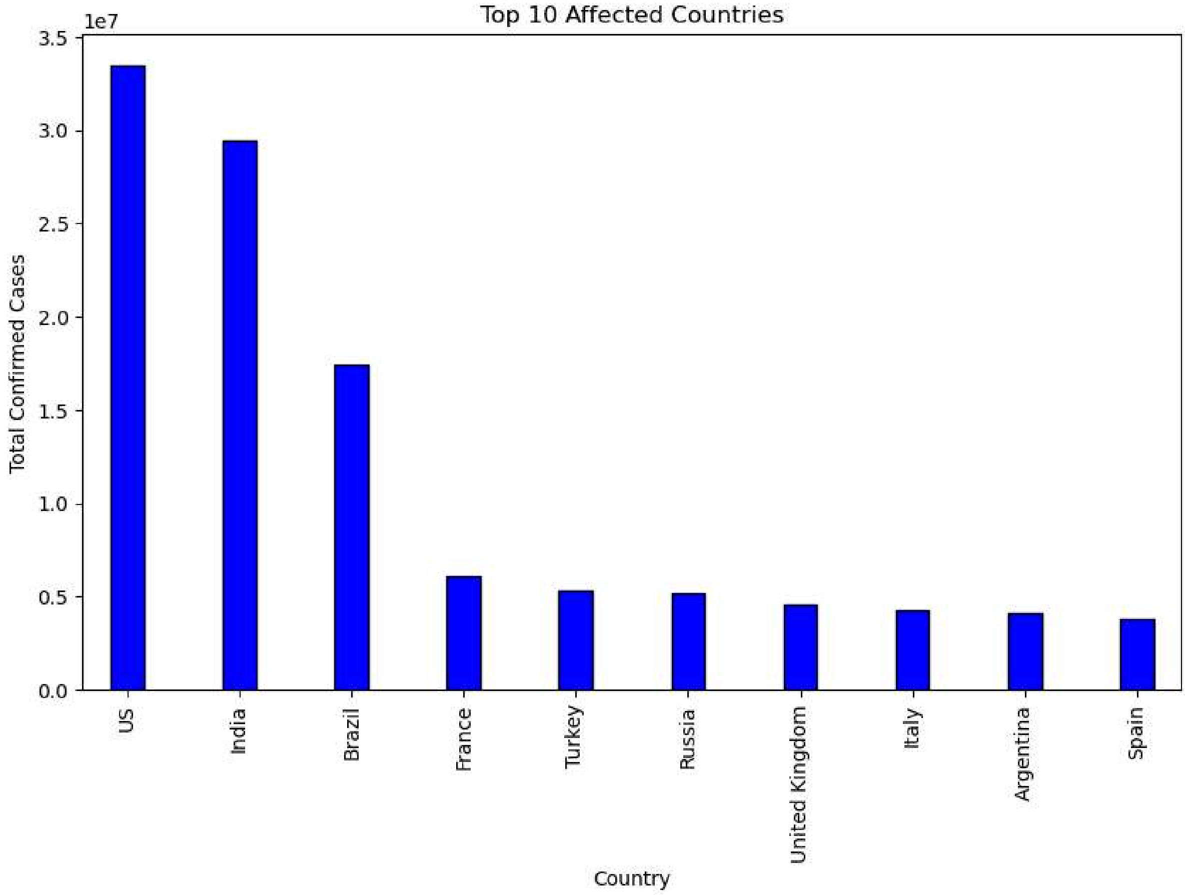
In [24]:

```
# Plotting Total confirmed cases over time
plt.figure(figsize=(10,6))
confirmed_over_time.plot(label='Confirmed Cases',color='purple')
deaths_over_time.plot(label='Deaths',color='Red')
recovered_over_time.plot(label='Recovered',color='green')
plt.title("Total Confirmed COVID - 19 Cases Over Time")
plt.xlabel("Date")
plt.ylabel("Confirmed Cases")
plt.legend()
plt.grid(True)
plt.show()
```



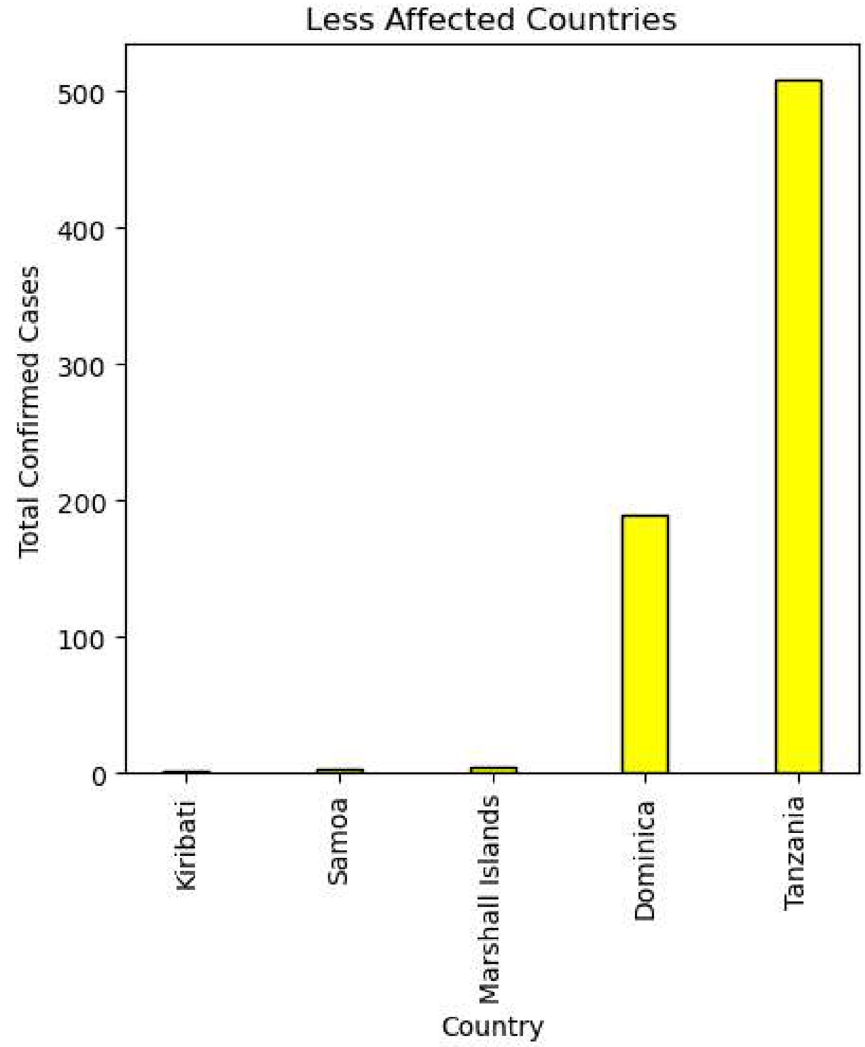
In [25]:

```
# Top 10 affected Countries
top_affected = data.groupby('Country/Region')['Confirmed'].sum().sort_values(ascend

# Bar chart for top affected countries
plt.figure(figsize=(10,6))
top_affected.plot.bar(color='blue',edgecolor = 'black',width=0.3)
plt.title("Top 10 Affected Countries")
plt.xlabel("Country")
plt.ylabel("Total Confirmed Cases")
plt.show()
```



In [26]:

```
# Five less affected countries
less_affected = data.groupby('Country/Region')['Confirmed'].sum().sort_values().hea

# Bar plot for less affected countries
plt.figure(figsize=(5,5))
less_affected.plot.bar(color='yellow',edgecolor='black',width=0.3)
plt.title("Less Affected Countries")
plt.xlabel("Country")
plt.ylabel("Total Confirmed Cases")
plt.show()
```



In [27]:

```
# Visulaize and explore the data
```

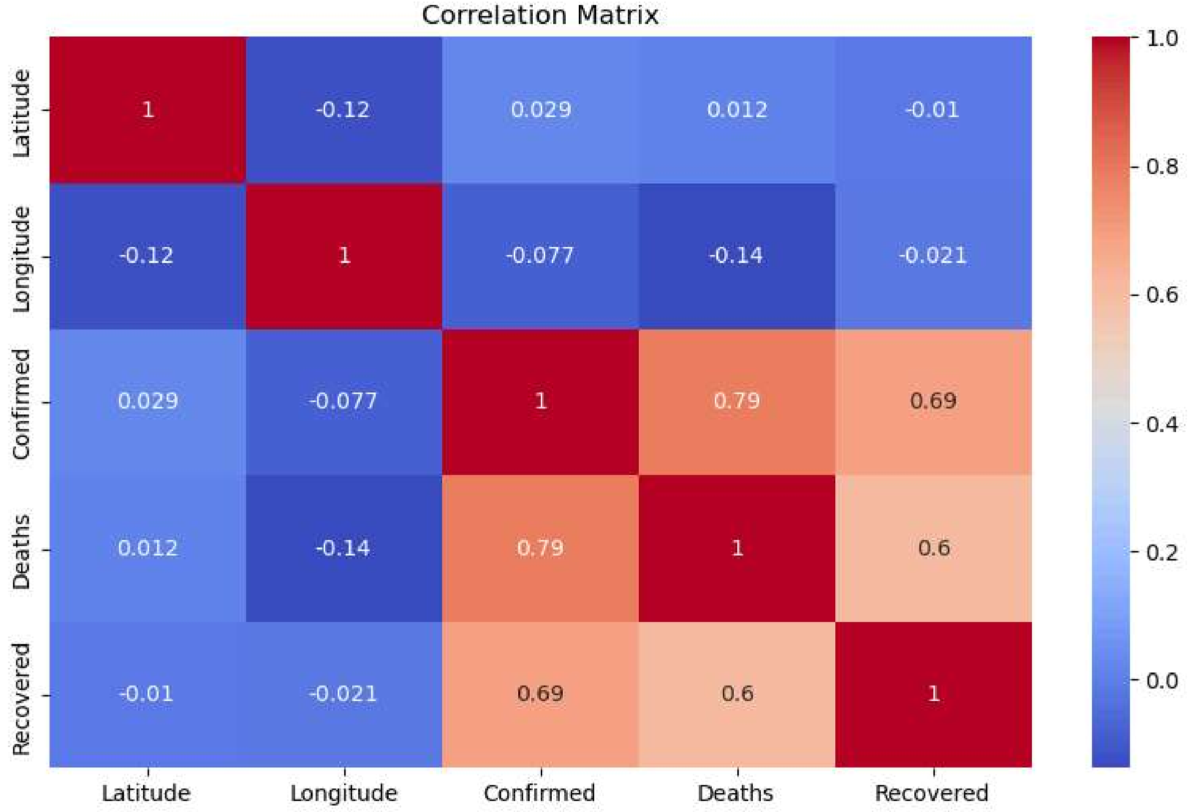
In [28]:

```
print("\n              Summary Statistics")
print(data.describe())
```

```
              Summary Statistics
           Latitude     Longitude      Confirmed        Deaths     Recovered
count  78386.000000  78386.000000   78386.000000  78386.000000  7.838600e+04
mean      21.645431     27.499549    2156.291391     46.537086  1.442675e+03
std       26.790357     71.401825   12541.577492    214.225689  1.034558e+04
min      -42.882100   -172.104600       0.000000      0.000000  0.000000e+00
25%        6.611100     -7.092600       0.000000      0.000000  0.000000e+00
50%       26.207000     26.798741      23.000000      0.000000  2.000000e+00
75%       42.315400     84.250000     643.750000     10.000000  3.070000e+02
max       71.706900    174.886000  823225.000000   7374.000000  1.123456e+06
```

In [29]:
```
# Correlation Matrix
plt.figure(figsize=(10,6))
sns.heatmap(data.corr(),annot=True,cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```

```
C:\Users\ROHITH DP\AppData\Local\Temp\ipykernel_2376\3267824859.py:3: FutureWarnin
g: The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  sns.heatmap(data.corr(),annot=True,cmap='coolwarm')
```



Classification

In [30]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
```

In [43]:
```
# Filter the top 10 countries
```

In [40]:
```
country_data = data.groupby('Country/Region')['Confirmed'].sum().reset_index()
```

In [41]:
```
top_10_countries = country_data.sort_values(by='Confirmed',ascending = False).head(
```

In [42]:
```
data_top_10 = data[data['Country/Region'].isin(top_10_countries['Country/Region'])]
```

In [46]:

```
# Create Labels
median_confirmed = data_top_10['Confirmed'].median()
data_top_10['Severity'] = ( data_top_10['Confirmed'] > median_confirmed).astype(int
```

```
C:\Users\ROHITH DP\AppData\Local\Temp\ipykernel_2376\3630555613.py:3: SettingWithC
opyWarning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.o
rg/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  data_top_10['Severity'] = ( data_top_10['Confirmed'] > median_confirmed).astype
(int)
```

In [47]:

```
# Features and Target
x = data_top_10[['Deaths','Recovered']]
y = data_top_10['Severity']

# Split the data into train and testing sets
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=42)
```

Random Forest Classifier

In [52]:

```
RFC = RandomForestClassifier()
RFC.fit(x_train,y_train)
pred_RFC = RFC.predict(x_test)
print('Random Forest Accuracy:',accuracy_score(y_test,pred_RFC))
print('Random Forest Classification Report:\n',classification_report(y_test,pred_RF
print("Random Forest Confusion Matrix:\n",confusion_matrix(y_test,pred_RFC))
```

```
Random Forest Accuracy: 0.9627659574468085
Random Forest Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.96      0.96      1224
           1       0.96      0.97      0.96      1220

    accuracy                           0.96      2444
   macro avg       0.96      0.96      0.96      2444
weighted avg       0.96      0.96      0.96      2444

Random Forest Confusion Matrix:
 [[1169   55]
 [  36 1184]]
```

Logistic Regression

In [53]:

```
LR = LogisticRegression()
LR.fit(x_train,y_train)
pred_LR = LR.predict(x_test)
print("Logistic Regression Accuracy:",accuracy_score(y_test,pred_LR))
print("Logistic Regression Classification Report:\n",classification_report(y_test,p
print("Logistic Regression Confusion Matrix:\n",confusion_matrix(y_test,pred_LR))
```

```
Logistic Regression Accuracy: 0.9279869067103109
Logistic Regression Classification Report:
               precision    recall  f1-score   support

           0       0.88      0.99      0.93      1224
           1       0.98      0.87      0.92      1220

    accuracy                           0.93      2444
   macro avg       0.93      0.93      0.93      2444
weighted avg       0.93      0.93      0.93      2444

Logistic Regression Confusion Matrix:
 [[1206   18]
 [ 158 1062]]
```

Decision Tree Classifier

In [54]:

```
DTC = DecisionTreeClassifier()
DTC.fit(x_train,y_train)
pred_DTC = DTC.predict(x_test)
print("Decision Tree Accuracy:",accuracy_score(y_test,pred_DTC))
print("Decision Tree Classificatio Report:\n",classification_report(y_test,pred_DTC
print("Decision Tree Confusion Matrix:\n",confusion_matrix(y_test,pred_DTC))
```

```
Decision Tree Accuracy: 0.953355155482815
Decision Tree Classificatio Report:
               precision    recall  f1-score   support

           0       0.95      0.96      0.95      1224
           1       0.96      0.95      0.95      1220

    accuracy                           0.95      2444
   macro avg       0.95      0.95      0.95      2444
weighted avg       0.95      0.95      0.95      2444

Decision Tree Confusion Matrix:
 [[1172   52]
 [  62 1158]]
```

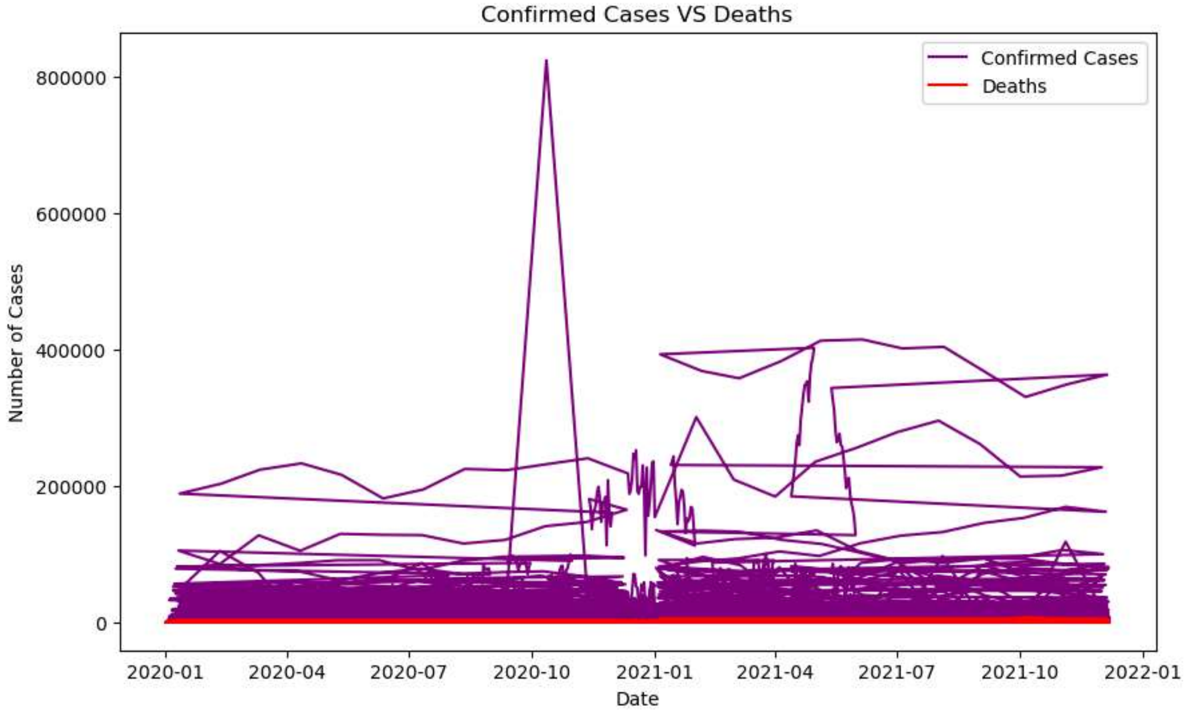
In [56]:

```
SVM = SVC()
SVM.fit(x_train,y_train)
pred_SVM = SVM.predict(x_test)
print("SVM Accuracy:",accuracy_score(y_test,pred_SVM))
print("SVM Classification Report:\n",classification_report(y_test,pred_SVM))
print("SVM Confusion Matrix:\n",confusion_matrix(y_test,pred_SVM))
```

```
SVM Accuracy: 0.838379705400982
SVM Classification Report:
               precision    recall  f1-score   support

           0       0.76      0.99      0.86      1224
           1       0.98      0.69      0.81      1220

    accuracy                           0.84      2444
   macro avg       0.87      0.84      0.83      2444
weighted avg       0.87      0.84      0.83      2444

SVM Confusion Matrix:
 [[1211   13]
 [ 382  838]]
```

In [59]:

```
plt.figure(figsize=(10,6))
plt.plot(data['Date'],data['Confirmed'],label='Confirmed Cases',color = 'purple')
plt.plot(data['Date'],data['Deaths'],label='Deaths',color='red')
plt.title("Confirmed Cases VS Deaths")
plt.xlabel("Date")
plt.ylabel("Number of Cases")
plt.legend()
plt.show()
```


Confirmed Cases VS Deaths
Confirmed Cases
Deaths
Number of Cases
Date