**TERRORISM HOTSPOT AROUND THE WORLD DATA ANALYSIS USING PYTHON**

A minor project

### 

### TABLE OF CONTENT

Abstract

Problem Statement

1. Introduction
2. Motivation & Objective
   1. Motivation
   2. Objective
3. Software and Hardware Requirements
   1. Software Requirements
   2. Hardware Requirements
4. Data Analysis
   1. Defining a Question ?
   2. Data Set Generation
   3. CRUD Operations
   4. Multi-dimensional Data Models
   5. Data Pre-Processing Techniques
   6. Data Visualization
5. Result
6. Conclusion

## ABSTRACT

The Global Terrorism Database (GTD) documents more than 200,000 international and domestic terrorist attacks that occurred worldwide since 1970. With details on various dimensions of each attack, the GTD familiarizes analysts, policymakers, scholars, and journalists with patterns of terrorism. The GTD defines terrorist attacks as: The threatened or actual use of illegal force and violence by a non-state actor to attain a political, economic, religious, or social goal through fear, coercion, or intimidation.

**Primary Findings:**

Some general findings derived from the GTD involve the nature and distribution of terrorist attacks. For example, about half of all terrorist attacks in the GTD are non-lethal, and although approximately one percent of attacks involve 25 or more fatalities, these highly lethal attacks killed more than 140,000 people in total between 1970 and 2018. The attacks in the GTD are attributed to more than 2,000 named perpetrator organizations and more than 700 additional generic groupings such as "Tamil separatists." However, two-thirds of these groups are active for less than a year and carry out fewer than four total attacks. Likewise, only 20 perpetrator groups are responsible for half of all attacks from 1970 to 2018 for which a perpetrator was identified. In general, patterns of terrorist attacks are very diverse across time and place and the GTD supports in-depth analysis of these patterns.

START researchers use the GTD to conduct statistical analyses of patterns of terrorist attacks, perpetrator groups, and responses to terrorism using innovative analytical strategies. Selected findings from these analyses include: (1) the vast majority of terrorist attacks, including those attributed to organizations that represent the most serious foreign threat to the US, mostly attack domestic targets in their own countries; (2) conciliatory actions by the government are sometimes more effective at reducing terrorist attacks than are repressive actions, (3) perpetrator organizations can be classified into those that desist rapidly and those that desist gradually, if at all, based on the shape of their activity over time; and (4) the groups most likely to persist are those with a rapid pattern of onset, while those with a gradual pattern of onset are more likely to decline quickly.

## PROBLEM STATEMENT

Problem Statement:

To conduct a comprehensive data analysis of terrorism hotspots worldwide, identifying trends, patterns, and key contributing factors. This project aims to uncover geographical, temporal, and socio-political factors influencing the prevalence of terrorism, enabling policymakers, security agencies, and researchers to develop more effective strategies for prevention and mitigation.

Terrorism continues to pose a significant threat globally, with various regions experiencing heightened levels of activity. The project aims to analyze a comprehensive dataset of terrorism incidents to identify hotspots around the world. This analysis will delve into factors such as geographical location, frequency of attacks, types of attacks, targets, perpetrators, and socio-political contexts. By examining these data points, the project seeks to uncover underlying patterns, trends, and correlations that can shed light on the dynamics of terrorism hotspots. The ultimate goal is to provide actionable insights to policymakers, security agencies, and researchers, facilitating the development of more targeted and effective counterterrorism strategies.

## INTRODUCTION

In today's data-driven world, understanding and addressing the global phenomenon of terrorism requires sophisticated data engineering techniques and tools. The project "Terrorism Hotspot All Over the World Data Analysis" leverages the power of Python in data engineering to process, transform, and analyze large-scale datasets related to terrorism incidents worldwide.

Python, with its rich ecosystem of libraries and frameworks, offers unparalleled flexibility and efficiency in handling diverse data sources, formats, and structures. In this project, Python serves as the primary language for data acquisition, cleaning, integration, and preparation, laying the foundation for rigorous analysis and insight generation.

The data engineering pipeline begins with the acquisition of terrorism-related datasets from various sources, including public repositories, government agencies, and research organizations. Python's libraries such as Pandas, Requests, and Beautiful Soup facilitate seamless web scraping, API integration, and file handling, enabling the retrieval of comprehensive and up-to-date data.

Once the raw data is obtained, Python scripts are employed to clean and preprocess the data, addressing issues such as missing values, inconsistencies, and outliers. Techniques such as data imputation, standardization, and normalization ensure the integrity and quality of the dataset, preparing it for subsequent analysis.

Python's capabilities in data transformation and manipulation are harnessed to integrate disparate datasets, merge relevant variables, and derive new features that provide deeper insights into terrorism hotspots. Whether it's geospatial analysis to identify geographic clusters of terrorist activity or temporal analysis to detect emerging trends over time, Python enables the creation of sophisticated data pipelines tailored to the specific needs of the project.

By harnessing the power of Python in data engineering, the project "Terrorism Hotspot All Over the World Data Analysis" aims to provide actionable insights and intelligence to stakeholders, including policymakers, security agencies, and researchers. Through rigorous data processing, analysis, and visualization, Python enables us to unravel the complexities of terrorism dynamics and contribute to the global efforts in countering terrorism and promoting peac

## 2.MOTIVATION & OBJECTIVE

The motivation behind the project "terrorism hotspot all over the world data analysis" is likely to better understand patterns, trends, and factors contributing to terrorism globally. The objective could be to identify high-risk areas, analyze causes and impacts of terrorism. By analyzing data on terrorism hotspots, researchers or policymakers can potentially enhance security measures, allocate resources more effectively, and contribute to efforts in countering terrorism.

## MOTIVATION

## The motivation behind the project "Terrorism Hotspots All Over the World Data Analysis" lies in the urgent need to confront and mitigate the pervasive threat posed by terrorism on a global scale. Terrorism transcends borders, ideologies, and cultures, exacting a heavy toll on human lives, societal stability, and economic prosperity. Understanding the distribution, patterns, and underlying factors driving terrorism hotspots is essential for developing effective strategies to combat this complex phenomenon.

## 2.2 OBJECTIVE

* + - **Data Acquisition**: Utilize Python libraries such as APIs for data retrieval, or Pandas for importing datasets from various sources to collect comprehensive data on terrorism incidents worldwide.
    - **Data Cleaning**: Employ Python's Pandas library to clean and preprocess the raw data, handling missing values, inconsistencies, and outliers to ensure data integrity and reliability for analysis.
    - **Statistical Modeling**: Apply statistical techniques and machine learning algorithms using Python libraries . Predict terrorism hotspots based on historical data, enabling the identification of high-risk regions.
    - **Visualization**: Use Python libraries to create informative charts, graphs, and interactive visualizations that communicate the findings of the data analysis, facilitating interpretation and decision-making.
    - **Interpretation and Insights**: Analyze the results of data analytics using Python to extract actionable insights, such as identifying key drivers of terrorism hotspots, assessing risk factors, and recommending strategic interventions for policymakers and stakeholders
    - **Documentation and Reporting**: Document the data analysis process, methodologies, and findings using Python tools l

## 3. SOFTWARE & HARDWARE REQURIMENTS

* 1. **SOFTWARE REQURIMENTS**

**Operating System :** Windows

**Programming Language :** Python

**Modules Required :** Pandas , Matplotlib

**Modules :** Create own Dataset and perform all Data pre- processing operations and visualize the data.

**IDE’s :** Python Google Colab & Spyder

## HARDWARE REQURIMENTS

**Processor :** 11th Gen Intel(R) core (TM) i5-1155G7@ 2.50GH

**RAM :** 8.00GB

**Version**

## 4 DATA ANALYSIS

### 4.1 Defining a Question ?

The problem statement could be: "Developing a data analytics framework to identify and analyze terrorism hotspots worldwide, enabling timely and effective interventions to mitigate security risks and protect lives."

**4.2 Dataset Generation**

We generate the dataset on basis of terrorism hotspots world wide. We consider the key column as Rank world wide and dataset generation code is as follows

import pandas as pd

# Existing data

set= {

"Ranking in world":pd.Series([1,2,3,5,4,6,8,9,10,11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons":(["Sword","Axe","Bow","Crossbow","Spear","Dagger","Mace", "Staff","Throwing Knives","Warhammer","Pistol"]),

"weapons\_cost":([2000,1500,2500,3500,1200,1000,1800,2200,1300,3000,4000]),

"Attacks in 2024" : pd.Series(['120','130','170','130','60','90','90','80','110','140','30']),

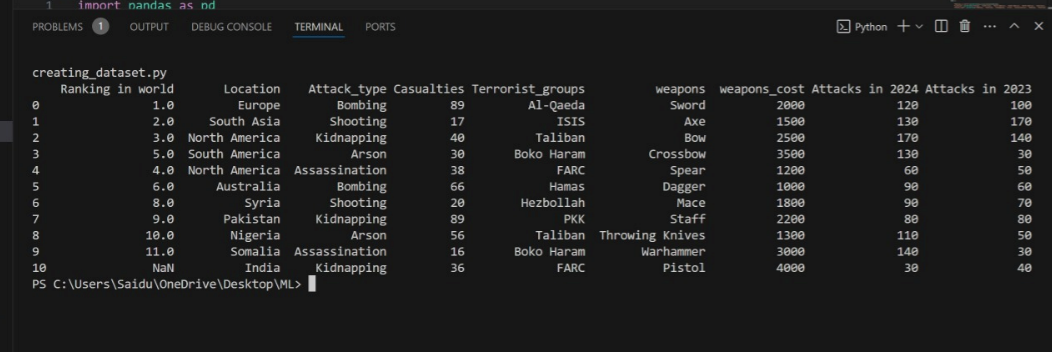
"Attacks in 2023":pd.Series(['100','170','140','30','50','60','70','80','50','30','40']),

}

result = pd.DataFrame(set)

print(result)

**The generated dataset of Terriorism hotspots all over the world is as follows..**

****

**4.3 CRUD Operations**

CRUD stands for Create, Read, Update, and Delete. In Python, CRUD operations refer to basic operations performed on data such as creating new records, reading existing records, updating records, and deleting records in a database or any other data storage system. These operations are fundamental in working with databases or data structures in Python.

**4.31 Create**

Here we can perform insertion operations for both rows and colums

1. **Column insertion :**

The code for the column insertion is as follows

import pandas as pd

# Existing data

set = {

# "Date": pd.Series(['2022-05-20', '2021-02-20', '2019-07-12', '2023-12-09', '2017-04-15', '2016-15-10', '2015-06-19', '2024-03-12', '2013-02-30', '2012-01-17', '2011-09-20']),

"Ranking in world":pd.Series([1,2,3,5,4,6,8,9,10,11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons":(["Sword","Axe","Bow","Crossbow","Spear","Dagger","Mace", "Staff","Throwing Knives","Warhammer","Pistol"]),

"weapons\_cost":([2000,1500,2500,3500,1200,1000,1800,2200,1300,3000,4000]),

"Attacks in 2024" : pd.Series(['120','130','170','130','60','90','90','80','110','140','30']),

"Attacks in 2023":pd.Series(['100','170','140','30','50','60','70','80','50','30','40'])

}

#CRUD --->1. create 2. read 3.delete 4.update

#1.create or insert

# column insertion

print('----before Modified-----')

print()

result = pd.DataFrame(set)

print(result)

set['Pincode']=pd.Series([23111,523231,523232,523234,523325,523325,523236,523327,523238,523239,523260])

b=pd.DataFrame(set)

print('----after Modified-----')

print()

print(b)import pandas as pd

result = pd.DataFrame(set)

print(result)

set['Pincode']=pd.Series([23111,523231,523232,523234,523325,523325,523236,523327,523238,523239,523260])

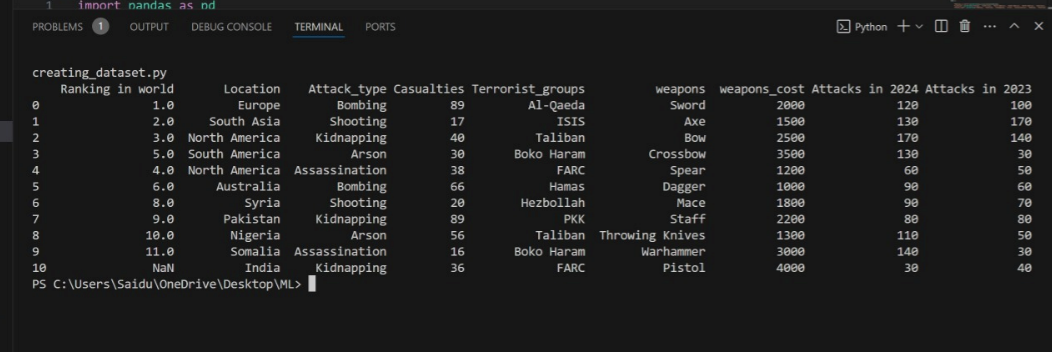
b=pd.DataFrame(set)

print('----after Modified-----')

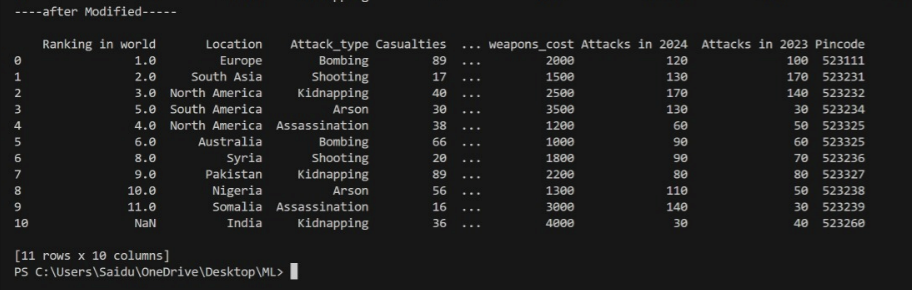
print()

print(b)

**Output Before insertion:**

****

**After appending a column ,the output is as follows**

****

**b.Row insertion :**

The code for a new row insertion is as follows

import pandas as pd

# Existing data

set = {

#"Date": pd.Series(['2022-05-20', '2021-02-20', '2019-07-12', '2023-12-09', '2017-04-15', '2016-15-10', '2015-06-19', '2024-03-12', '2013-02-30', '2012-01-17', '2011-09-20']),

"Ranking in world":pd.Series([1,2,3,5,4,6,8,9,10,11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons":(["Sword","Axe","Bow","Crossbow","Spear","Dagger","Mace", "Staff","Throwing Knives","Warhammer","Pistol"]),

"weapons\_cost":([2000,1500,2500,3500,1200,1000,1800,2200,1300,3000,4000]),

"Attacks in 2024" : pd.Series(['120','130','170','130','60','90','90','80','110','140','30']),

"Attacks in 2023":pd.Series(['100','170','140','30','50','60','70','80','50','30','40'])

}

# Convert dictionary to DataFrame

b = pd.DataFrame(set)

print()

print('----before Modified-----')

print()

print(b)

# New row data

new\_row\_data = {

#"Date": '2025-04-30',

"Location": 'Asia',

"Attack\_type": 'Bombing',

"Casualties": '50',

"Terrorist\_groups": 'New Group',

"weapons":'AK64',

"weapons\_cost":3000,

"Attacks in 2024" :70,

"Attacks in 2023":25

}

# Append new row to the DataFrame

print()

print('---after Modified-----')

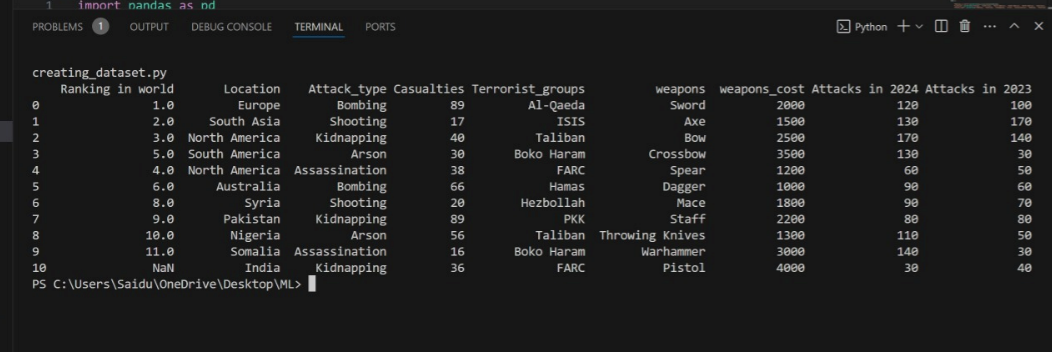
print()e

b = pd.concat([b, pd.DataFrame([new\_row\_data])], ignore\_index=True)

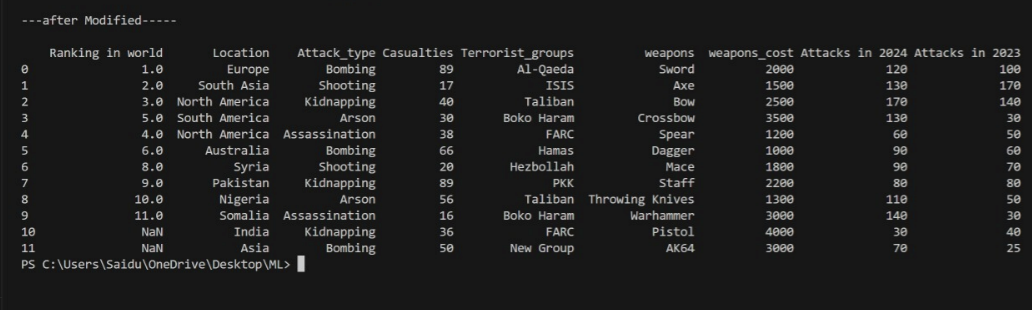
# Print the updated DataFrame

print(b)

**The dataset before row insertion is:**

****

**The data set after insertion of a row is :**

****

**4.32 Read or Retrieve**

We can perform read operation for the values from a dataset .we can select a row or a column or both .

**a.Read column values:**

To read a particular column in our dataset , the code is as follows

import pandas as pd

# Existing data

set = {

# "Date": pd.Series(['2022-05-20', '2021-02-20', '2019-07-12', '2023-12-09', '2017-04-15', '2016-15-10', '2015-06-19', '2024-03-12', '2013-02-30', '2012-01-17', '2011-09-20']),

"Ranking in world":pd.Series([1,2,3,5,4,6,8,9,10,11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons":(["Sword","Axe","Bow","Crossbow","Spear","Dagger","Mace", "Staff","Throwing Knives","Warhammer","Pistol"]),

"weapons\_cost":([2000,1500,2500,3500,1200,1000,1800,2200,1300,3000,4000]),

"Attacks in 2024" : pd.Series(['120','130','170','130','60','90','90','80','110','140','30']),

"Attacks in 2023":pd.Series(['100','170','140','30','50','60','70','80','50','30','40'])

}

#red entai column data

result = pd.DataFrame(set)

print('---before Modified-----')

print()

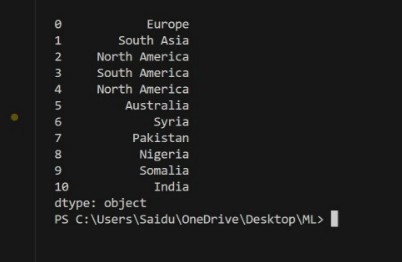
print(result)

print('---after Modified-----')

print()

print(set["Location"])

**OUTPUT:**

****

1. **Read row values:**

To read a particular column in our dataset , we use loc() method and the code is as follows

a

import pandas as pd

# Existing data

set = {

# "Date": pd.Series(['2022-05-20', '2021-02-20', '2019-07-12', '2023-12-09', '2017-04-15', '2016-15-10', '2015-06-19', '2024-03-12', '2013-02-30', '2012-01-17', '2011-09-20']),

"Ranking in world":pd.Series([1,2,3,5,4,6,8,9,10,11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons":(["Sword","Axe","Bow","Crossbow","Spear","Dagger","Mace", "Staff","Throwing Knives","Warhammer","Pistol"]),

"weapons\_cost":([2000,1500,2500,3500,1200,1000,1800,2200,1300,3000,4000]),

"Attacks in 2024" : pd.Series(['120','130','170','130','60','90','90','80','110','140','30']),

"Attacks in 2023":pd.Series(['100','170','140','30','50','60','70','80','50','30','40'])

}

#read entire row data

df = pd.DataFrame(set)

print()

print('----before Modified-----')

print()

print(df)

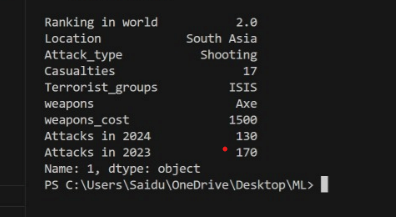
print()

print('----after Modified-----')

print()

print(df.loc[1])

**OUTPUT:**

****

**c.Read a particular record:**

****

**OUTPUT:**

****

**4.33 Update**

This means changing an element without selecting. We can update both column update as well as row update also.

1. **column update**

To update a particular column the code is as follows

import pandas as pd

# Existing data

data = {

"Ranking in world": pd.Series([1, 2, 3, 5, 4, 6, 8, 9, 10, 11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons": (["Sword", "Axe", "Bow", "Crossbow", "Spear", "Dagger", "Mace", "Staff", "Throwing Knives", "Warhammer", "Pistol"]),

"weapons\_cost": ([2000, 1500, 2500, 3500, 1200, 1000, 1800, 2200, 1300, 3000, 4000]),

"Attacks in 2024": pd.Series(['120', '130', '170', '130', '60', '90', '90', '80', '110', '140', '30']),

"Attacks in 2023": pd.Series(['100', '170', '140', '30', '50', '60', '70', '80', '50', '30', '40'])

}

# Convert dictionary to DataFrame

df = pd.DataFrame(data)

print('----before Modification-----')

print(df)

# Update the "Location" value in the row with index 1

df.iloc[1, df.columns.get\_loc('Location')] = 'ANDRA PRADESH'

print('----after Modification-----')

print(df)

**OUTPUT:**



1. **Row update**

import pandas as pd

# Existing data

set\_data = {

"Ranking in world": pd.Series([1, 2, 3, 5, 4, 6, 8, 9, 10, 11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons": (["Sword", "Axe", "Bow", "Crossbow", "Spear", "Dagger", "Mace", "Staff", "Throwing Knives", "Warhammer", "Pistol"]),

"weapons\_cost": ([2000, 1500, 2500, 3500, 1200, 1000, 1800, 2200, 1300, 3000, 4000]),

"Attacks in 2024": pd.Series(['120', '130', '170', '130', '60', '90', '90', '80', '110', '140', '30']),

"Attacks in 2023": pd.Series(['100', '170', '140', '30', '50', '60', '70', '80', '50', '30', '40'])

}

# Convert dictionary to DataFrame

set\_df = pd.DataFrame(set\_data)

print('----before Modified-----')

print(set\_df)

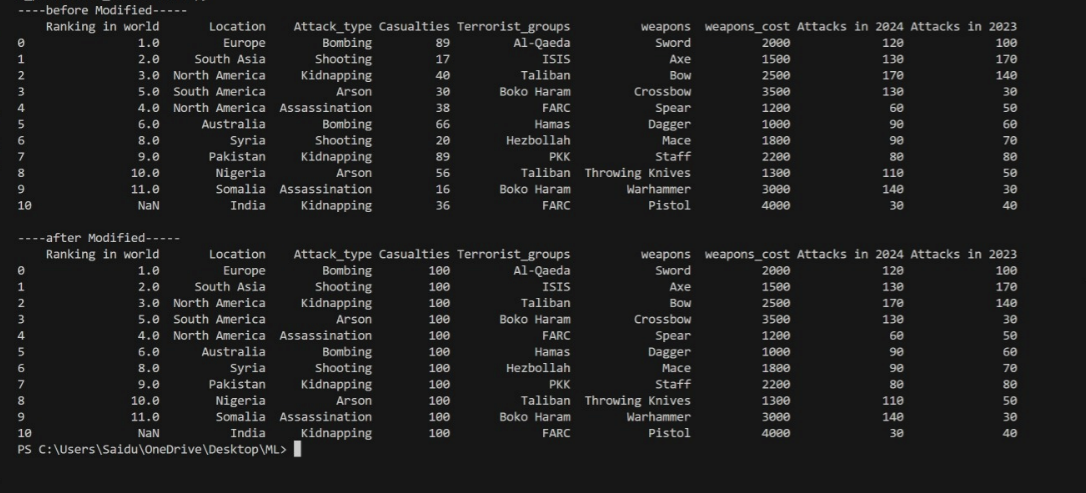
# Update the entire "Date" column

set\_df["Casualties"] = '100'

print('\n----after Modified-----')

print(set\_df)

**OUTPUT:**

****

**4.34 Deletion**

We can perform delete operation in column with the help of pop() and in row with the help of drop().

**a.column deletion:**

import pandas as pd

# Existing data

set = {

#"Date": pd.Series(['2022-05-20', '2021-02-20', '2019-07-12', '2023-12-09', '2017-04-15', '2016-15-10', '2015-06-19', '2024-03-12', '2013-02-30', '2012-01-17', '2011-09-20']),

"Ranking in world":pd.Series([1,2,3,5,4,6,8,9,10,11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons":(["Sword","Axe","Bow","Crossbow","Spear","Dagger","Mace", "Staff","Throwing Knives","Warhammer","Pistol"]),

"weapons\_cost":([2000,1500,2500,3500,1200,1000,1800,2200,1300,3000,4000]),

"Attacks in 2024" : pd.Series(['120','130','170','130','60','90','90','80','110','140','30']),

"Attacks in 2023":pd.Series(['100','170','140','30','50','60','70','80','50','30','40'])

}

# Convert dictionary to DataFrame

set\_df = pd.DataFrame(set)

print('---before Modified-----')

print()

print(set\_df)

# Drop the "Location" column

set\_df = set\_df.drop(columns=['Location'])

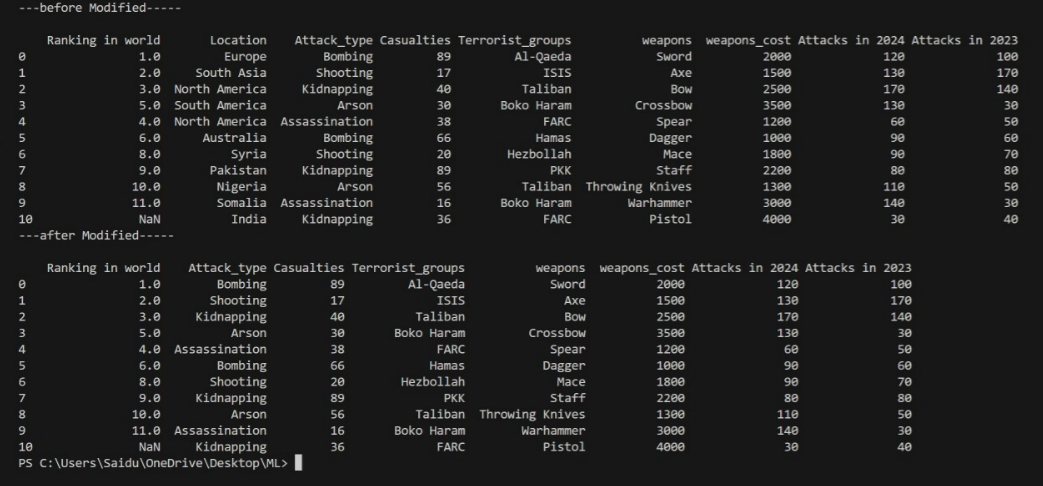
print('---after Modified-----')

print()

# Print the updated DataFrame

print(set\_df)

**OUTPUT:**

****

**b.row deletion :**

import pandas as pd

# Existing data

set\_data = {

"Ranking in world": pd.Series([1, 2, 3, 5, 4, 6, 8, 9, 10, 11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons": (["Sword", "Axe", "Bow", "Crossbow", "Spear", "Dagger", "Mace", "Staff", "Throwing Knives", "Warhammer", "Pistol"]),

"weapons\_cost": ([2000, 1500, 2500, 3500, 1200, 1000, 1800, 2200, 1300, 3000, 4000]),

"Attacks in 2024": pd.Series(['120', '130', '170', '130', '60', '90', '90', '80', '110', '140', '30']),

"Attacks in 2023": pd.Series(['100', '170', '140', '30', '50', '60', '70', '80', '50', '30', '40'])

}

# Convert dictionary to DataFrame

set\_df = pd.DataFrame(set\_data)

print('----before Modified-----')

print(set\_df)

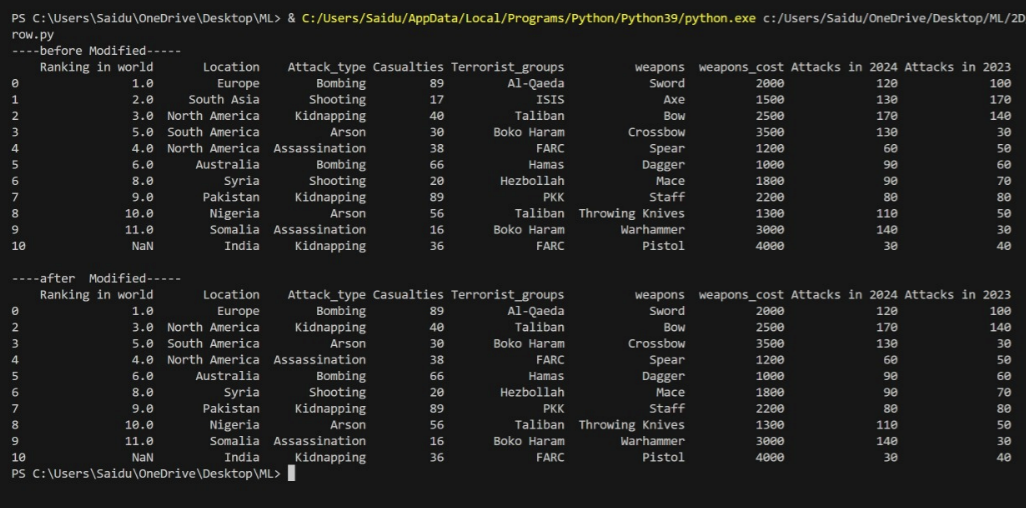
# Drop the row with index 1

set\_df = set\_df.drop(index=1)

print('\n----after Modified-----')

print(set\_df)

**OUTPUT:**

****

**4.5 Data Pre-Processing**

Data preprocessing is a crucial step in preparing raw data for analysis or modeling. It involves cleaning, transforming, and organizing data to make it suitable for further processing. This typically includes tasks such as handling missing values, removing duplicates, scaling features, encoding categorical variables, and splitting data into training and testing sets. The goal is to ensure that the data is consistent, accurate, and ready for analysis or machine learning algorithms.

**4.51 Data Collection**

Data collection is the process of gathering raw information or observations for analysis or research purposes. It involves identifying relevant sources, such as databases, surveys, sensors, or web scraping, and systematically collecting data from these sources. Depending on the nature of the data, collection methods may include manual entry, automated extraction, or real-time monitoring. It's essential to ensure that data collection methods are ethical, reliable, and appropriate for the intended analysis or research objectives. Data collected must also be carefully documented to maintain its integrity and facilitate further processing and analysis.

**4.52 Data Cleaning**

It is a process of replacing the missing Values with user needed values while performing dacta cleaning operation Fint we need to check the particular value is miss missing we need to use two methods or not. For this we need **isnull( )** and **notnull( )**

**a.isnull( )**

import pandas as pd

import numpy as np

# Existing data

set= {

#"Date": pd.Series(['2022-05-20', '2021-02-20', '2019-07-12', '2023-12-09',np.nan, '2016-15-10', '2015-06-19', '2024-03-12', '2013-02-30', '2012-01-17', '2011-09-20']),

"Ranking in world":pd.Series([1,2,3,5,4,6,8,9,10,11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', np.nan, 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons":(["Sword","Axe","Bow","Crossbow","Spear","Dagger","Mace", "Staff","Throwing Knives","Warhammer","Pistol"]),

"weapons\_cost":([2000,1500,np.nan,3500,1200,np.nan,1800,2200,np.nan,3000,4000]),

"Attacks in 2024" : pd.Series(['120','130','170','130','60','90','90','80','110','140','30']),

"Attacks in 2023":pd.Series(['100','170','140','30','50','60','70','80','50','30','40'])

}

result = pd.DataFrame(set)

print(result)

print()

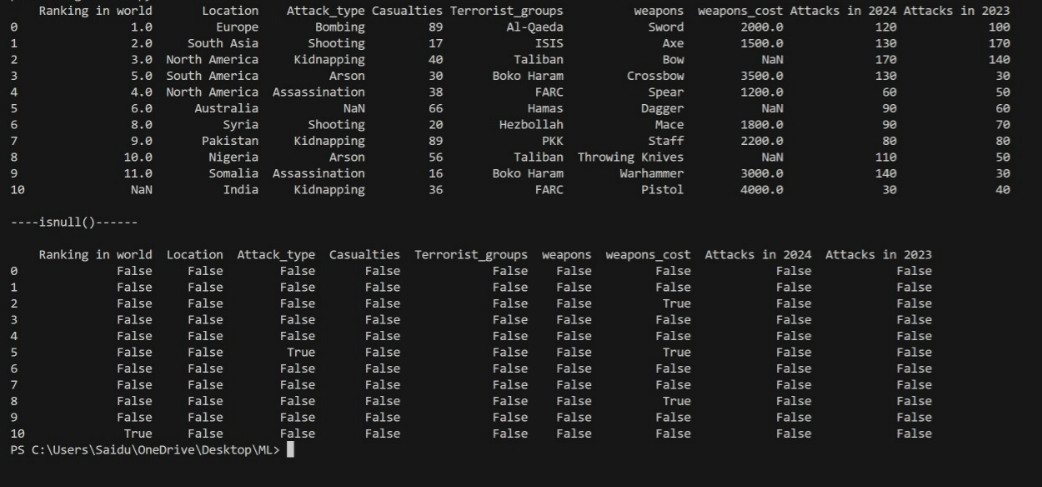
result = result.isnull()

print('----isnull()------')

print()

print(result)

**OUTPUT:**

****

**b.notnull( )**

import pandas as pd

import numpy as np

# Existing data

set= {

# "Date": pd.Series(['2022-05-20', '2021-02-20', '2019-07-12', '2023-12-09',np.nan, '2016-15-10', '2015-06-19', '2024-03-12', '2013-02-30', '2012-01-17', '2011-09-20']),

"Ranking in world":pd.Series([1,2,3,5,4,6,8,9,10,11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', np.nan, 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons":(["Sword","Axe","Bow","Crossbow","Spear","Dagger","Mace", "Staff","Throwing Knives","Warhammer","Pistol"]),

"weapons\_cost":([2000,1500,np.nan,3500,1200,np.nan,1800,2200,np.nan,3000,4000]),

"Attacks in 2024" : pd.Series(['120','130','170','130','60','90','90','80','110','140','30']),

"Attacks in 2023":pd.Series(['100','170','140','30','50','60','70','80','50','30','40'])

}

result = pd.DataFrame(set)

print(result)

print()

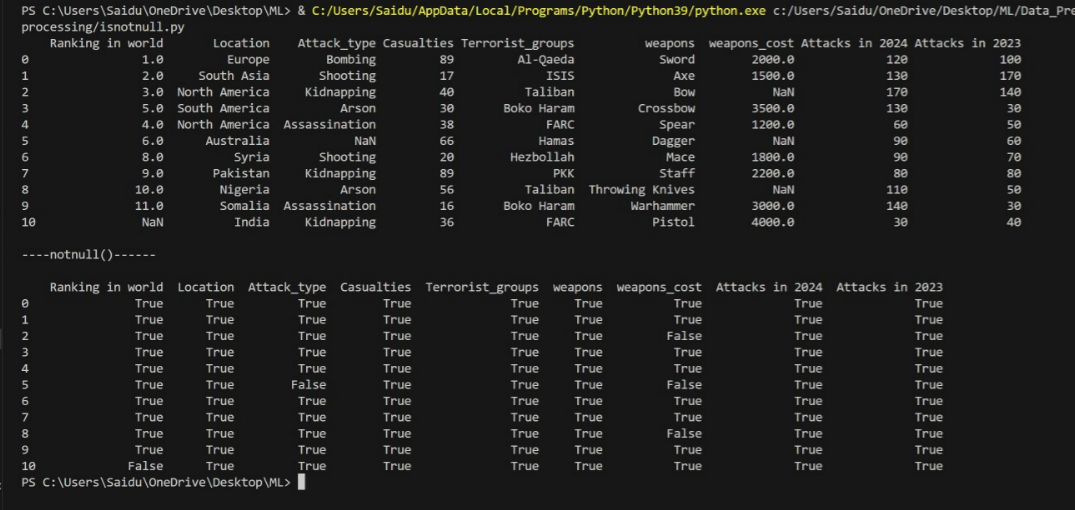
result = result.notnull()

print('----notnull()------')

print()

print(result)

**OUTPUT:**

****

**4.53 Data Integration**

Data integration is nothing nut filling the missing values with proper values using **fill( )** method

import pandas as pd

import numpy as np

# Existing data

data = {

# "Date": pd.Series(['2022-05-20', '2021-02-20', '2019-07-12', '2023-12-09',np.nan, '2016-15-10', '2015-06-19', '2024-03-12', '2013-02-30', '2012-01-17', '2011-09-20']),

"Ranking in world":pd.Series([1,2,3,5,4,6,8,9,10,11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', np.nan, 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons":(["Sword","Axe","Bow","Crossbow","Spear","Dagger","Mace", "Staff","Throwing Knives","Warhammer","Pistol"]),

"weapons\_cost":([2000,1500,np.nan,3500,1200,np.nan,1800,2200,np.nan,3000,4000]),

"Attacks in 2024" : pd.Series(['120','130','170','130','60','90','90','80','110','140','30']),

"Attacks in 2023":pd.Series(['100','170','140','30','50','60','70','80','50','30','40'])

}

# Creating DataFrame

df = pd.DataFrame(data)

# Filling missing values

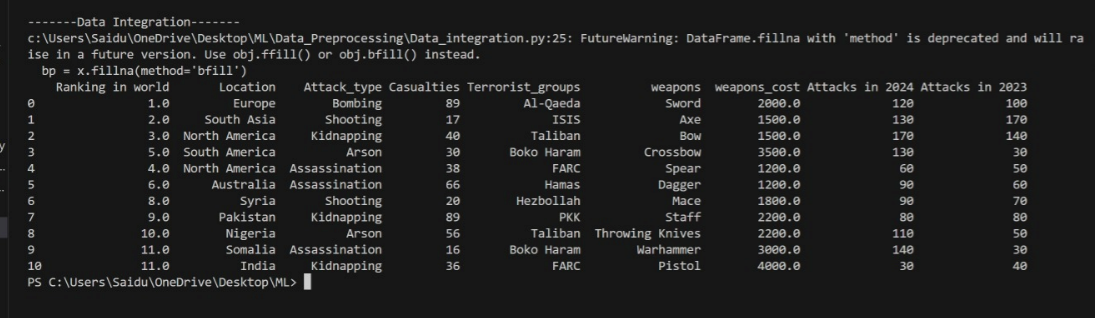
x = df.fillna(method='ffill')

print("-------Data Integration-------")

bp = x.fillna(method='bfill')

print(bp)

**OUTPUT:**

****

**4.54 Data Reduction**

Data reduction is nothing but removing the duplicates.It is a process of remove Values in a data set

With method drop-duplicate( )

import pandas as pd

import numpy as np

# Existing data

data = {

#"Date": pd.Series(['2022-05-20', '2021-02-20', '2019-07-12', '2023-12-09',np.nan, '2016-15-10', '2015-06-19', '2024-03-12', '2013-02-30', '2012-01-17', '2011-09-20']),

"Ranking in world":pd.Series([1,2,3,5,4,6,8,9,10,11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Attack\_type": pd.Series(['Bombing', 'Shooting', 'Kidnapping', 'Arson', 'Assassination', np.nan, 'Shooting', 'Kidnapping', 'Arson', 'Assassination', 'Kidnapping']),

"Casualties": pd.Series(['89', '17', '40', '30', '38', '66', '20', '89', '56', '16', '36']),

"Terrorist\_groups": pd.Series(['Al-Qaeda', 'ISIS', 'Taliban', 'Boko Haram', 'FARC', 'Hamas', 'Hezbollah', 'PKK', 'Taliban', 'Boko Haram', 'FARC']),

"weapons": pd.Series(["Sword","Axe","Bow","Crossbow","Spear","Dagger","Mace", "Staff","Throwing Knives","Warhammer","Pistol"]),

"Attacks in 2024" : pd.Series(['120','130','170','130','60','90','90','80','110','140','30']),

"Attacks in 2023":pd.Series(['100','170','140','30','50','60','70','80','50','30','40'])

}

df = pd.DataFrame(data)

print('----before Data Reduction--------')

print()

print(df)

# Creating DataFrame

df = pd.DataFrame(data)

print("--------Data Reduction---------")

result = df.drop\_duplicates()

print(result)

**OUTPUT:**



**4.55 Data Transformation**

Data transformation involves converting data from one format, structure, or representation to another, with the goal of making it suitable for specific analysis or processing tasks. This process can include various operations such as normalization, standardization, encoding, scaling, and aggregation.

Data transformation is a critical step in data preprocessing, as it helps prepare the data for analysis, modeling, and visualization by ensuring that it is in a suitable format and representation for the intended use case.

**4.56 Data Discretization**

Data discretization is the process of converting continuous variables into discrete categories or intervals. This is often done to simplify data analysis, reduce complexity, and improve the performance of certain algorithms that work better with categorical or ordinal data.

Data discretization can be applied to various types of data, including numerical and ordinal variables. It's essential to choose an appropriate discretization method based on the nature of the data and the specific requirements of the analysis or modeling task. Discretization helps simplify the interpretation of results, reduce computational complexity, and improve the performance of certain algorithms, especially those that rely on categorical or ordinal inputs**.**

**4.6 Data Visualization**

Data visualization is the graphical representation of data and information. It involves creating visualizations such as charts, graphs, maps, and dashboards to communicate insights, patterns, and trends in the data. Data visualization aims to make complex datasets more understandable, interpretable, and actionable for a wide range of audiences.

Common types of data visualizations include bar charts, line charts, scatter plots, histograms, heatmaps, pie charts, and geographical maps. Choosing the right visualization depends on the nature of the data, the relationships to be explored, and the intended audience.

Overall, data visualization plays a crucial role in data analysis, storytelling, and decision-making processes, helping to unlock the value of data and drive actionable insights.

**4.61 bar( )**

import pandas as pd

import matplotlib.pyplot as plt

# Existing data

data = {

"Ranking in world": pd.Series([1, 2, 3, 5, 4, 6, 8, 9, 10, 11]),

"Location": pd.Series(['Europe', 'South Asia', 'North America', 'South America', 'North America', 'Australia', 'Syria', 'Pakistan', 'Nigeria', 'Somalia', 'India']),

"Casualties": pd.Series([89, 17, 40, 30, 38, 66, 20, 89, 56, 16, 36]),

"Attacks in 2024": pd.Series([120, 130, 170, 130, 60, 90, 90, 80, 110, 140, 30]),

"Attacks in 2023": pd.Series([100, 170, 140, 30, 50, 60, 70, 80, 50, 30, 40])

}

# Creating DataFrame

df = pd.DataFrame(data)

# Scatter plot

plt.figure(figsize=(8, 6))

plt.scatter(df['Ranking in world'], df['Casualties'], color='blue', label='Casualties')

plt.xlabel('Ranking in world')

plt.ylabel('Casualties')

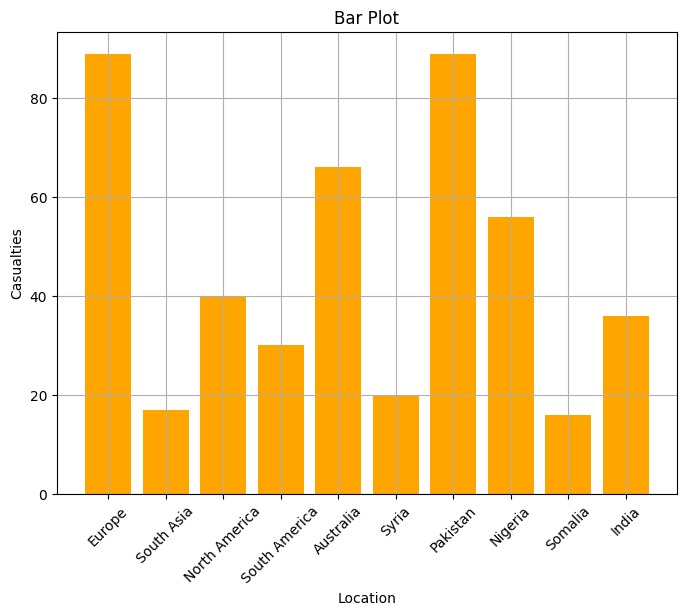
plt.title('Scatter Plot')

plt.legend()

plt.grid(True)

plt.show()

**OUTPUT**:



**4.62 Line Plot**

# Line plot

plt.figure(figsize=(8, 6))

plt.plot(df['Ranking in world'], df['Attacks in 2024'], marker='o', color='green', label='Attacks in 2024')

plt.plot(df['Ranking in world'], df['Attacks in 2023'], marker='s', color='red', label='Attacks in 2023')

plt.xlabel('Ranking in world')

plt.ylabel('Number of Attacks')

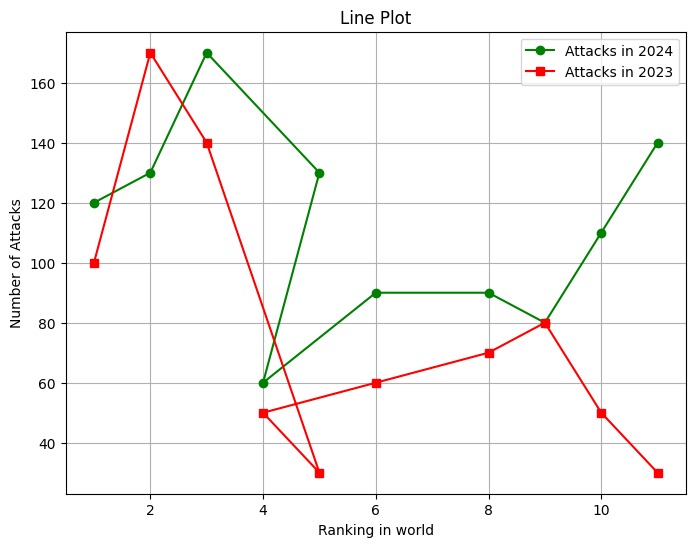
plt.title('Line Plot')

plt.legend()

plt.grid(True)

plt.show()

**OUTPUT:**



**4.63 Scatter ( )**

# Bar plot

plt.figure(figsize=(8, 6))

plt.bar(df['Location'], df['Casualties'], color='orange')

plt.xlabel('Location')

plt.ylabel('Casualties')

plt.title('Bar Plot')

plt.xticks(rotation=45)

plt.grid(True)

plt.show()

**OUTPUT:**

# 

## 5.RESULT

I have come to end of my project on Terrorism Hotspot all over the world data analysis. On the time working on my project I have learned many new things on terrorism regions. I obtained a lot of experience on working of projects. The data provides valuable insights into the demographic distribution of the insured population, including terrorism groups, attacks , and possibly geographical

location. Understanding these demographics can help in world peace and harmony to better suit the needs of different demographic segments.

## 6 CONCLUSION

## In this data analysis we have made CRUD operations , data pre-processing techniques ,data visualization on the data set of Terrorism Hotspot data analysis .

In today's data-driven world, understanding and addressing the global phenomenon of terrorism requires sophisticated data engineering techniques and tools. The project "Terrorism Hotspot All Over the World Data Analysis" leverages the power of Python in data engineering to process, transform, and analyze large-scale datasets related to terrorism incidents worldwide.

Python, with its rich ecosystem of libraries and frameworks, offers unparalleled flexibility and efficiency in handling diverse data sources, formats, and structures. In this project, Python serves as the primary language for data acquisition, cleaning, integration, and preparation, laying the foundation for rigorous analysis and insight generation.

## 6 CONCLUSION

## In this data analysis we have made CRUD operations , data pre-processing techniques ,data visualization on the data set of Terrorism Hotspot data analysis .

In today's data-driven world, understanding and addressing the global phenomenon of terrorism requires sophisticated data engineering techniques and tools. The project "Terrorism Hotspot All Over the World Data Analysis" leverages the power of Python in data engineering to process, transform, and analyze large-scale datasets related to terrorism incidents worldwide.

Python, with its rich ecosystem of libraries and frameworks, offers unparalleled flexibility and efficiency in handling diverse data sources, formats, and structures. In this project, Python serves as the primary language for data acquisition, cleaning, integration, and preparation, laying the foundation for rigorous analysis and insight generation.