Project Report

Course Name (NICF)	Advanced Certificate in Data Science
Product Name (Marketing & Sales)	Advanced Certificate in Data Science
Module Name (NICF)	**WSQ- Data Analysis and Visualization(SF)**
Product Name (Marketing & Sales)	**WSQ- Data Analysis and Visualization(SF)**

Student name	**Assessor name**
MEGARAJ MRITTIKA	Ryan Suryanto

Date issued	**Completion date**	**Submitted on**
27.09.2023	09.10.2023	11.10.2023

Project title	**Russia_ Ukraine: The War Analysis and Visualization**

Learner declaration
I certify that the work submitted for this assignment is my own and research sources are fully acknowledged. Student signature: Megaraj Mrittika Date: 11/10/2023

Project Overview: Describe the Project along with Project Outcomes (Explain the Project in your own words in 15 – 20 lines)

Project Definition

Ukraine has been at war with Russia and its separatists since February 2014 in what is known as the Russo-Ukrainian War. This war broke out soon after the Ukrainian Revolution for Dignity and has since remained a major global problem. The political status of Crimea and the Donbas, which are still regarded as being a part of Ukraine, are two major focus points of this conflict. The battle has seen a variety of occurrences, from cyberwarfare to maritime mishaps, all supported by rising political tensions.

The acquisition of Crimea by Russia marked a turning point in the conflict, which was then quickly followed by military confrontations in the Donbas, where insurgents backed by Russia clashed with Ukrainian government forces. As Russian forces surrounded Ukrainian territory in 2021, tensions between the two countries grew. Finally, on February 24, 2022, Russia began a full-scale assault on the Ukrainian mainland, ushering in a new chapter of the battle. In this project, we will explore the data related to the Russo-Ukrainian War using the Pandas, NumPy, Matplotlib, and Seaborn libraries. Our main goal is to analyze and provide this data thoroughly.

Project Goal

Using the Pandas, NumPy, Matplotlib, and Seaborn libraries, this project's main objective is to analyze and display data related to the Russo-Ukrainian War, which started in February 2014. Armed clashes, maritime accidents, cyber warfare, and severe political unrest are all part of this complex conflict between Russia and Ukraine. Our project will concentrate on using data analysis to comprehend the mechanics of the war. The dataset might include details about the political climate in the internationally recognized Ukrainian regions of Crimea and the Donbas. We will carefully examine and alter the data using Pandas and NumPy to derive valuable insights. Afterward, we'll use the Matplotlib and Seaborn tools to create visual representations of our findings. Finding patterns, trends, and connections within the context of the dispute will be greatly helped by visualization. Our goal is to produce visually appealing and educational plots, charts, and graphs that provide a comprehensive understanding of the Russo-Ukrainian War.

Our goal is to better comprehend this conflict and all of its complex aspects by using these cutting-edge data analysis and visualization approaches. The conclusions we draw from this effort may clarify the chronology, geographic details, and other causes that have aided in the conflict's intensification. Ultimately, our project's objective is to provide a comprehensive overview of the Russo-Ukrainian War through the lens of data analysis and visualization, leveraging Pandas, NumPy, Matplotlib, and Seaborn libraries.

In summary, this project will serve as an integrated platform for comprehending the Russo-Ukrainian War by harnessing the power of data analysis and visualization tools, shedding light on its complexities and nuances.

1. Project Technical Environment: (Describe the Architecture with Tools used)

Various tools and technologies are accessible and can be used depending on the particular project requirements.

Architecture:

In order to handle and present data in a meaningful way, the architecture for data analysis and visualization often requires several components working together. The elements that data scientists frequently employ are listed below.

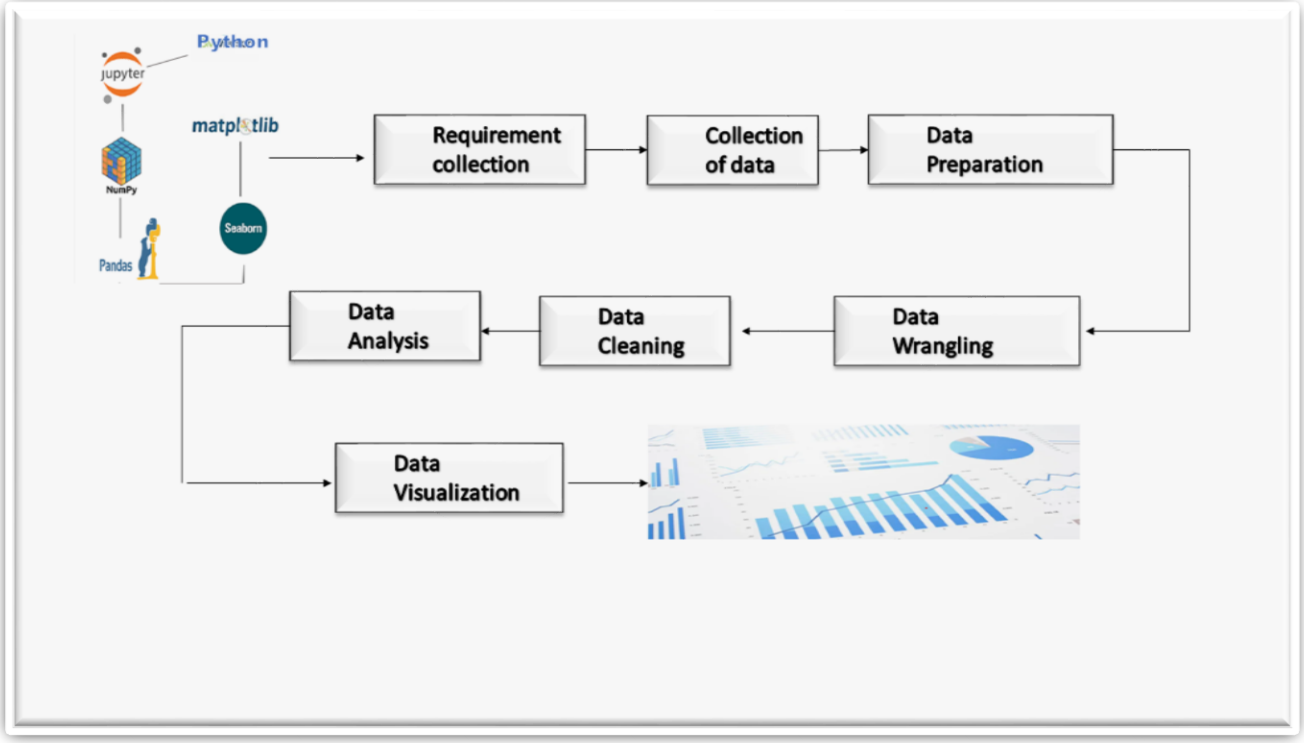


Fig1.1: Architecture of Russia_ Ukraine: The War Analysis and Visualization

- **Requirements Collection**

The project requirements must be gathered and made clear before moving on to the technical parts. In this instance, our main objective is to thoroughly analyze and provide data pertaining to the Russo-Ukrainian War. Understanding the dynamics of the conflict, including elements like political changes, occurrences, and geographic patterns, is necessary for this. We'll need to determine the data's breadth, specify our goals, and pinpoint our important stakeholders.

Python: Python, our go-to programming language, will serve as the foundation for all of our data analysis and visualization work. It is a powerful asset thanks to its adaptability and wide-ranging libraries.

Jupyter: For our data analysis, Jupyter notebooks offer an interactive and group-based environment. They help us to record our work so that it is available to and understandable to others.

NumPy: The Swiss Army knife of numerical operations is NumPy. It offers support for effective array operations and mathematical functions, both of which are necessary for manipulating data.

Pandas: Pandas is our toolbox for data manipulation. It enables us to manage data extraction, transformation, and cleaning with ease, guaranteeing that our data is spotless and prepared for analysis.

Matplotlib and Seaborn: These libraries are our artistic tools, enabling us to create visually compelling representations of our insights. They offer a wide range of chart types to convey our findings in a meaningful and engaging manner.

- **Collection of Data**

The sources we used to collect our data from—databases, spreadsheets, and CSV files—offer a variety of viewpoints on the Russo-Ukrainian War. To make sure that our analysis accurately captures the dynamic character of the dispute, we won't be restricted to just static sources; instead, we'll also look at real-time data from news sources and social media. Our strategy will be governed by data ethics, with a focus on privacy and security while managing sensitive information. The goal of this thorough data collection strategy is to present a complete picture of the Russo-Ukrainian War.

- **Data Preparation**

Data preparation is a crucial step in the data analysis process as it lays the foundation for accurate and reliable insights. In addition to cleaning and transforming the data, this stage also involves dealing with missing values, handling outliers, and standardizing variables. These steps further enhance the quality and consistency of the data, ensuring that it is ready for analysis.

- **Data Wrangling**

Data wrangling is a critical component of data preparation. It encompasses activities such as handling missing values, dealing with outliers, and structuring the data for analysis. During this phase, we'll leverage Python libraries like Pandas and NumPy to streamline the data into a usable format.

- **Data Cleaning**

Data cleaning is an essential step in the data analysis process as it ensures that the data is accurate and reliable. By removing duplicates and addressing errors, we can trust that the insights derived from the data will be valid and meaningful. Additionally, standardizing data formats allows for easier comparison and integration of different datasets, further enhancing the quality of analysis.

- **Data Analysis**

Now that our data has been cleaned up, data analysis is the major event. In this stage, Pandas and NumPy, two reliable Python libraries, are used to mine the data for informational nuggets. The basis of our understanding of the Russo-Ukrainian War will be built using statistical methods, data mining, and potentially even machine learning.

- **Data Visualization**

After extracting meaningful insights from the data, we'll proceed to visualize our findings. Data visualization tools like Matplotlib, Seaborn, and Plotly will be employed to create visually appealing and informative charts, graphs, and plots. Visualization aids in presenting complex data in an understandable and engaging manner.

- **Reporting and Presentation**

Presenting the analyzed and visualized data to the stakeholders and end-users is the last phase in our process. We can create reports that clearly describe our findings by using specialized reporting tools like Jupyter Notebooks. These studies will be a useful tool for policymakers and anybody seeking information about the Russo-Ukrainian War.

An organized process comprising data collection, preparation, cleaning, analysis, visualization, and presentation is included in the project's technical environment, to sum up. The project's goal of thoroughly comprehending the Russo-Ukrainian War through data analysis and visualization depends on the success of each phase.

2. Analytical Technique & Tools used: Describe the Analytical Technique and Tools used in the Project

We use a comprehensive set of analytical approaches and tools in our effort to understand the complexity of the Russo-Ukrainian War through data analysis and visualization. Our project's foundation is made up of these techniques and tools, which allow us to draw conclusions from the data that are both instructive and visually appealing.

- **Descriptive Statistics**

  Analytical Technique:

  Descriptive statistics serve as our compass, guiding us through the vast sea of data. This technique involves the summarization and elucidation of key data characteristics. Metrics like mean, median, mode, standard deviation, range, and percentiles empower us to grasp the central tendencies and variability within our dataset.

  Tools Utilized:

  Our trusty companions in this phase are Python libraries, namely NumPy and Pandas. NumPy, renowned for its numerical prowess, facilitates precise calculations, while Pandas offers a haven for data manipulation and statistical computations. Additionally, we might enlist the assistance of conventional tools like Microsoft Excel for specific descriptive statistics.

- **Data Mining**

  Analytical Technique:

  Data mining, akin to a skilled prospector, helps us unearth valuable nuggets within our expansive dataset. This technique involves the exploration of hidden patterns, relationships, and trends. Clustering, classification, association rule mining, and anomaly detection are our guiding lights in this venture.

  Tools Employed:

  Python takes center stage once again. Its expansive ecosystem boasts a plethora of libraries designed for data mining, with Scikit-Learn standing tall. Scikit-Learn equips us with the tools needed for machine learning tasks like classification and clustering, enabling us to unearth intricate data relationships.

- **Sentiment Analysis**

Sentiment analysis, also known as opinion mining, is the process of determining the sentiment or emotional tone expressed in text data. In the context of our project, we can apply sentiment analysis to news articles, social media posts, or other textual sources related to the war.

This technique helps us understand public sentiment, reactions, and opinions surrounding the conflict.

- **Data Visualization**

  Analytical Technique:

  Data visualization acts as our artistic canvas, transforming raw data into vivid and comprehensible visuals. Through visualization, we can unravel complex narratives, identify trends, and spot anomalies, rendering the Russo-Ukrainian War data accessible and engaging.

  Tools Utilized:

  Python's arsenal includes Matplotlib, Seaborn, and Plotly, our artist's palette. These libraries enable us to craft a rich tapestry of charts, graphs, and interactive visualizations. With these tools, we turn data into captivating visuals that convey the story of the conflict.

- **Presentation and Reporting**

  Analytical Technique:

  Our journey culminates in the presentation and reporting phase, where we translate our insights into actionable information. This phase is the bridge that connects our analytical work to the end-users and stakeholders, offering them a clear path to understanding the Russo-Ukrainian War data.

  Tools Leveraged:

  To construct our bridge, we employ Jupyter Notebooks as our architectural blueprint. These notebooks allow us to seamlessly merge code, visualizations, and explanatory text into cohesive and insightful reports. Our reports serve as beacons, guiding decision-makers and those seeking deeper comprehension of the Russo-Ukrainian War.

In summary, our project harnesses an arsenal of analytical techniques and tools to navigate the complexities of the Russo-Ukrainian War data landscape. Through the lens of descriptive statistics, data mining, Sentiment analysis, data visualization, and thoughtful reporting, we aim to provide a holistic and enlightening perspective on this intricate conflict. These techniques and tools act as our guiding compass, empowering us to explore, analyze, and convey the data's essence, shedding light on the dynamics of an ongoing historical event.

3. Project Task List

Activity 1: Task Summary

1.1 Create a Word document Name of the file**: ACDS-DAV-0823-Megaraj Mrittika Russia_ Ukraine: The War Analysis and Visualization**

1.2 Determine the software and programming scripting language required to develop this project

To develop this project, we will need the following software and programming scripting languages:

- **Programming Language: Python**

Python is a versatile and widely used language in data analysis and visualization. It offers a rich ecosystem of libraries and tools that are essential for this project.

- **Integrated Development Environment (IDE): Jupyter Notebook**

Jupyter Notebook provides an interactive and collaborative environment for data analysis, code documentation, and report generation. It's an excellent choice for combining code, visualizations, and explanatory text in one document.

- **Python Libraries:**

1. **NumPy:** NumPy is used for numerical operations and efficient array handling.
2. **Pandas:** Pandas are crucial for data manipulation, including data extraction, transformation, and cleaning.
3. **Matplotlib:** Matplotlib helps create static, animated, or interactive visualizations.
4. **Seaborn:** Seaborn is a high-level interface for creating attractive and informative statistical graphics.

1.3 Analyze the flow and design the step-by-step flow

Our project is an expedition into the depths of the Russo-Ukrainian War data landscape. The journey is meticulously planned with distinct phases and innovative subheadings, each contributing to our quest for insights and understanding.

1. Collection and Data Gathering

- Quest for Knowledge: Collect project requirements and define objectives.
- Harvesting Data: Gather data related to the Russo-Ukrainian War from various sources, including databases, spreadsheets, and CSV files.

2. **Data Preparation and Cleaning**
    - Data Alchem: Use Python libraries (Pandas) to clean and preprocess the raw data.
    - The Art of Refinement: Handle missing values, outliers, and data formatting issues.
    - Structural Engineering: Transform and structure the data for analysis.
3. **Data Analysis and Data Mining**
    - The Analytical Arsenal: Apply analytical techniques, including descriptive statistics and data mining algorithms, to extract insights from the prepared data.
    - Mining the Data Veins: Utilize Python libraries (NumPy, Scikit-Learn) for data mining tasks like clustering and classification.
4. **Data Visualization**
    - The Visual Odyssey: Create meaningful visual representations of the insights gained using Python libraries (Matplotlib, Seaborn, Plotly).
    - Artistry in Data: Develop various charts, graphs, and interactive visualizations to convey patterns, trends, and correlations.
5. **Presentation and Reporting**
    - Storytelling with Code: Use Jupyter Notebook to combine code, visualizations, and explanatory text into comprehensive reports.
    - Illuminating the Path: Present the analyzed and visualized data to end-users and stakeholders.
6. **Version Control and Collaboration**
    - Guardians of Code: Implement version control using Git and a platform like GitHub to facilitate collaboration and track changes.
7. **Testing and Quality Assurance**
    - The Vigilant Sentry: Test the project components for accuracy and consistency.
    - The Code Crusade: Ensure the code follows best practices and adheres to coding standards.
8. **Deployment**
    - Sharing the Treasures: If applicable, deploy interactive visualizations or dashboards to share insights with a broader audience.
9. **Documentation**
    - The Chronicles: Document the project thoroughly, including data sources, methodologies, and code comments for future reference.
10. **Maintain and Update**
    - Guardians of Continuity: Regularly update and maintain the development, including Python libraries and tools, to leverage new features and security updates.

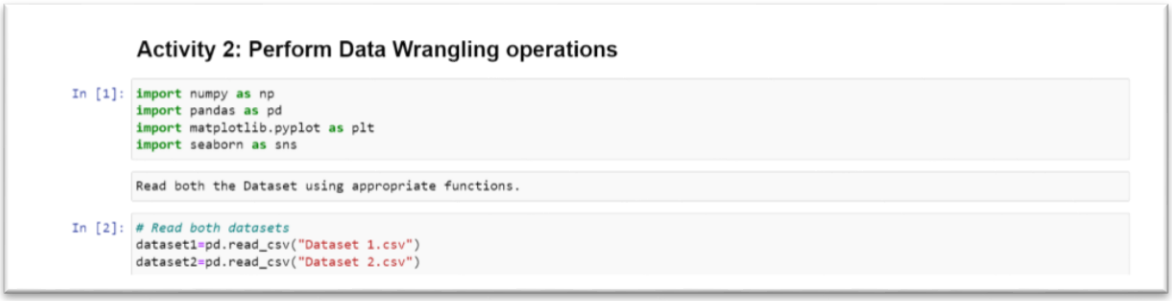
By following this step-by-step flow and leveraging the specified software and scripting languages, we can effectively analyze and visualize data related to the Russo-Ukrainian War, providing valuable insights and a comprehensive overview of the conflict.

Activity 2: Perform Data Wrangling operations

1. Develop a program with reusable functions with the Name of the file: **ACDS-DAV-0823-Megaraj Mrittika Russia_ Ukraine: The War Analysis and Visualization**

2. Import required packages Numpy, OS, Pandas, Matplotlib, Numpy

2.1 Read both the Dataset using appropriate functions



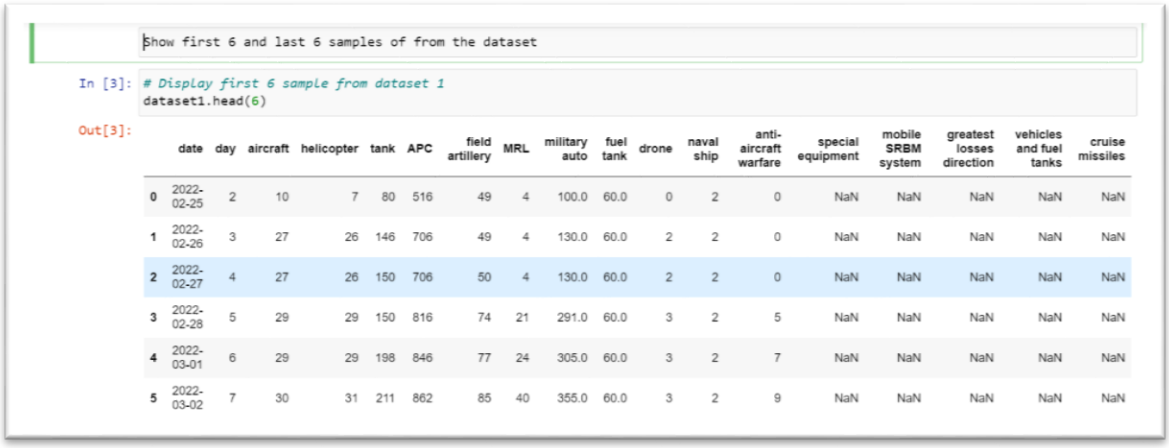
Activity 2: Perform Data Wrangling operations

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns

        Read both the Dataset using appropriate functions.

In [2]: # Read both datasets
        dataset1=pd.read_csv("Dataset 1.csv")
        dataset2=pd.read_csv("Dataset 2.csv")
```

2.2 Show the first 6 and last 6 samples from the dataset

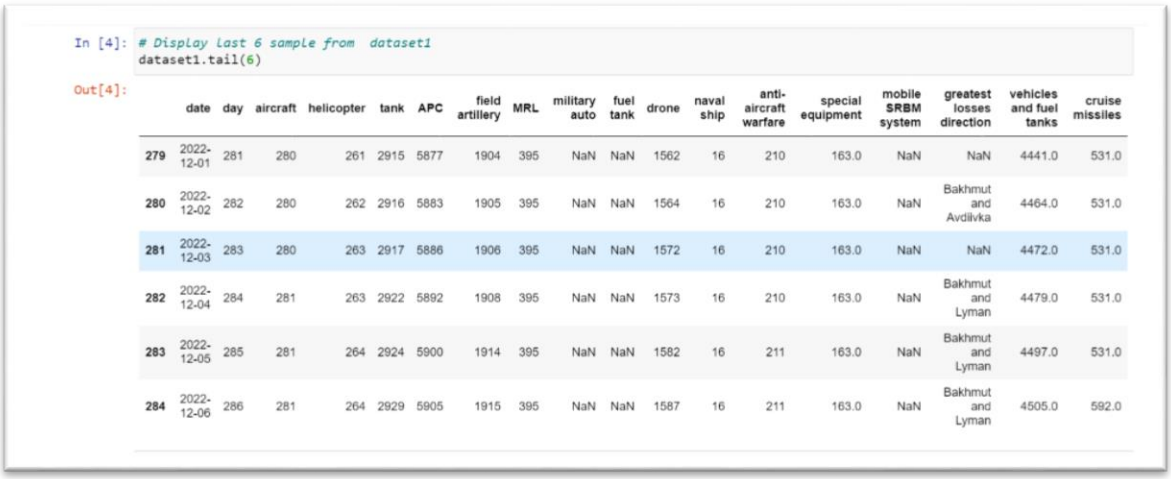


```
Show first 6 and last 6 samples of from the dataset

In [3]: # Display first 6 sample from dataset 1
        dataset1.head(6)
```

Out[3]:

	date	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	naval ship	anti-aircraft warfare	special equipment	mobile SRBM system	greatest losses direction	vehicles and fuel tanks	cruise missiles
0	2022-02-25	2	10	7	80	516	49	4	100.0	60.0	0	2	0	NaN	NaN	NaN	NaN	NaN
1	2022-02-26	3	27	26	146	706	49	4	130.0	60.0	2	2	0	NaN	NaN	NaN	NaN	NaN
2	2022-02-27	4	27	26	150	706	50	4	130.0	60.0	2	2	0	NaN	NaN	NaN	NaN	NaN
3	2022-02-28	5	29	29	150	816	74	21	291.0	60.0	3	2	5	NaN	NaN	NaN	NaN	NaN
4	2022-03-01	6	29	29	198	846	77	24	305.0	60.0	3	2	7	NaN	NaN	NaN	NaN	NaN
5	2022-03-02	7	30	31	211	862	85	40	355.0	60.0	3	2	9	NaN	NaN	NaN	NaN	NaN



```
In [4]: # Display last 6 sample from  dataset1
        dataset1.tail(6)
```

Out[4]:

	date	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	naval ship	anti-aircraft warfare	special equipment	mobile SRBM system	greatest losses direction	vehicles and fuel tanks	cruise missiles
279	2022-12-01	281	280	261	2915	5877	1904	395	NaN	NaN	1562	16	210	163.0	NaN	NaN	4441.0	531.0
280	2022-12-02	282	280	262	2916	5883	1905	395	NaN	NaN	1564	16	210	163.0	NaN	Bakhmut and Avdiivka	4464.0	531.0
281	2022-12-03	283	280	263	2917	5886	1906	395	NaN	NaN	1572	16	210	163.0	NaN	NaN	4472.0	531.0
282	2022-12-04	284	281	263	2922	5892	1908	395	NaN	NaN	1573	16	210	163.0	NaN	Bakhmut and Lyman	4479.0	531.0
283	2022-12-05	285	281	264	2924	5900	1914	395	NaN	NaN	1582	16	211	163.0	NaN	Bakhmut and Lyman	4497.0	531.0
284	2022-12-06	286	281	264	2929	5905	1915	395	NaN	NaN	1587	16	211	163.0	NaN	Bakhmut and Lyman	4505.0	592.0

```
In [5]: # Display first 6 sample  from dataset2
        dataset2.head(6)
```

Out[5]:

	date	day	personnel	personnel*	POW
0	2022-02-25	2	2800	about	0.0
1	2022-02-26	3	4300	about	0.0
2	2022-02-27	4	4500	about	0.0
3	2022-02-28	5	5300	about	0.0
4	2022-03-01	6	5710	about	200.0
5	2022-03-02	7	5840	about	200.0

```
In [6]: # Display last 6 sample from  dataset2
        dataset2.tail(6)
```

Out[6]:

	date	day	personnel	personnel*	POW
279	2022-12-01	281	89440	about	NaN
280	2022-12-02	282	90090	about	NaN
281	2022-12-03	283	90600	about	NaN
282	2022-12-04	284	91150	about	NaN
283	2022-12-05	285	91690	about	NaN

2.3 Find the Dataset information

Dataset1 information:

```
To get dataset1 information

In [7]: dataset1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 18 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   date                       285 non-null    object
 1   day                        285 non-null    int64
 2   aircraft                   285 non-null    int64
 3   helicopter                 285 non-null    int64
 4   tank                       285 non-null    int64
 5   APC                        285 non-null    int64
 6   field artillery            285 non-null    int64
 7   MRL                        285 non-null    int64
 8   military auto              65 non-null     float64
 9   fuel tank                  65 non-null     float64
 10  drone                      285 non-null    int64
 11  naval ship                 285 non-null    int64
 12  anti-aircraft warfare      285 non-null    int64
 13  special equipment          266 non-null    float64
 14  mobile SRBM system         36 non-null     float64
 15  greatest losses direction  195 non-null    object
 16  vehicles and fuel tanks    220 non-null    float64
 17  cruise missiles            220 non-null    float64
dtypes: float64(6)  int64(10)  object(2)
```

```
In [8]: dataset1.describe()
```

Out[8]:

	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	naval ship	anti-aircraft warfare	e
count	285.000000	285.000000	285.000000	285.000000	285.000000	285.000000	285.000000	65.000000	65.000000	285.000000	285.000000	285.000000	2
mean	144.000000	210.021053	185.912281	1665.957895	3720.287719	938.136842	243.217544	1047.507692	69.323077	707.547368	12.519298	121.119298	
std	82.416625	62.500419	53.116693	782.804694	1427.345027	541.040253	105.130544	466.162060	7.545917	473.079321	4.238964	56.446595	
min	2.000000	10.000000	7.000000	80.000000	516.000000	49.000000	4.000000	100.000000	60.000000	0.000000	2.000000	0.000000	
25%	73.000000	199.000000	155.000000	1122.000000	2713.000000	509.000000	172.000000	600.000000	60.000000	341.000000	11.000000	84.000000	
50%	144.000000	220.000000	188.000000	1684.000000	3879.000000	846.000000	248.000000	1178.000000	73.000000	688.000000	15.000000	110.000000	
75%	215.000000	260.000000	224.000000	2290.000000	4857.000000	1369.000000	330.000000	1437.000000	76.000000	970.000000	15.000000	172.000000	1
max	286.000000	281.000000	264.000000	2929.000000	5905.000000	1915.000000	395.000000	1701.000000	76.000000	1587.000000	16.000000	211.000000	1

```
In [9]: dataset1.dtypes

Out[9]: date          object
        day            int64
        aircraft       int64
        helicopter     int64
        tank           int64
```

```
In [10]: dataset1.shape
Out[10]: (285, 18)

In [11]: dataset1.columns
Out[11]: Index(['date', 'day', 'aircraft', 'helicopter', 'tank', 'APC',
                'field artillery', 'MRL', 'military auto', 'fuel tank', 'drone',
                'naval ship', 'anti-aircraft warfare', 'special equipment',
                'mobile SRBM system', 'greatest losses direction',
                'vehicles and fuel tanks', 'cruise missiles'],
               dtype='object')

In [12]: dataset1.index
Out[12]: RangeIndex(start=0, stop=285, step=1)
```

Dataset2 information:

```
To get dataset2 information

In [13]: dataset2.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   date        285 non-null    object
 1   day         285 non-null    int64
 2   personnel   285 non-null    int64
 3   personnel*  285 non-null    object
 4   POW         62 non-null     float64
dtypes: float64(1), int64(2), object(2)
memory usage: 11.3+ KB

In [14]: dataset2.describe()
```

Out[14]:

	day	personnel	POW
count	285.000000	285.000000	62.000000
mean	144.000000	42256.722807	386.387097
std	82.416625	22123.640833	131.440363
min	2.000000	2800.000000	0.000000

```
In [15]: dataset2.dtypes
Out[15]: date           object
         day             int64
         personnel       int64
         personnel*     object
         POW           float64
         dtype: object

In [16]: dataset2.shape
Out[16]: (285, 5)

In [17]: dataset2.columns
Out[17]: Index(['date', 'day', 'personnel', 'personnel*', 'POW'], dtype='object')

In [18]: dataset2.index
Out[18]: RangeIndex(start=0, stop=285, step=1)
```

2.4 Find the null values from the dataset

Null values of Dataset1:

```
In [19]: dataset1.isnull().sum()
Out[19]: date                        0
         day                         0
         aircraft                    0
         helicopter                  0
         tank                        0
         APC                         0
         field artillery             0
         MRL                         0
         military auto             220
         fuel tank                 220
         drone                       0
         naval ship                  0
         anti-aircraft warfare       0
         special equipment          19
         mobile SRBM system        240
         greatest losses direction  90
         vehicles and fuel tanks    65
         cruise missiles            65
         dtype: int64
```

Null values of Dataset2:

```
In [20]: dataset2.isnull().sum()

Out[20]: date            0
         day             0
         personnel       0
         personnel*      0
         POW           223
         dtype: int64
```

2.5 Drop the column with maximum NULL Values

To drop the column in dataset 1 and description:

```
To drop the column in dataset 1 and description

In [21]: dataset1.drop("mobile SRBM system",axis=1,inplace=True)
         dataset1
```

Out[21]:

	date	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	naval ship	anti-aircraft warfare	special equipment	greatest losses direction	vehicles and fuel tanks	cruise missiles
0	2022-02-25	2	10	7	80	516	49	4	100.0	60.0	0	2	0	NaN	NaN	NaN	NaN
1	2022-02-26	3	27	26	146	706	49	4	130.0	60.0	2	2	0	NaN	NaN	NaN	NaN
2	2022-02-27	4	27	26	150	706	50	4	130.0	60.0	2	2	0	NaN	NaN	NaN	NaN
3	2022-02-28	5	29	29	150	816	74	21	291.0	60.0	3	2	5	NaN	NaN	NaN	NaN
4	2022-03-01	6	29	29	198	846	77	24	305.0	60.0	3	2	7	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
280	2022-12-02	282	280	262	2916	5883	1905	395	NaN	NaN	1564	16	210	163.0	Bakhmut and Avdiivka	4464.0	531.0
281	2022-12-03	283	280	263	2917	5886	1906	395	NaN	NaN	1572	16	210	163.0	NaN	4472.0	531.0

```
In [22]: dataset1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 17 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   date                       285 non-null    object
 1   day                        285 non-null    int64
 2   aircraft                   285 non-null    int64
 3   helicopter                 285 non-null    int64
 4   tank                       285 non-null    int64
 5   APC                        285 non-null    int64
 6   field artillery            285 non-null    int64
 7   MRL                        285 non-null    int64
 8   military auto              65 non-null     float64
 9   fuel tank                  65 non-null     float64
 10  drone                      285 non-null    int64
 11  naval ship                 285 non-null    int64
 12  anti-aircraft warfare      285 non-null    int64
 13  special equipment          266 non-null    float64
 14  greatest losses direction  195 non-null    object
 15  vehicles and fuel tanks    220 non-null    float64
 16  cruise missiles            220 non-null    float64
dtypes: float64(5), int64(10), object(2)
memory usage: 38.0+ KB
```

```
In [25]: dataset1.shape

Out[25]: (285, 17)

In [26]: dataset1.columns

Out[26]: Index(['date', 'day', 'aircraft', 'helicopter', 'tank', 'APC',
                'field artillery', 'MRL', 'military auto', 'fuel tank', 'drone',
                'naval ship', 'anti-aircraft warfare', 'special equipment',
                'greatest losses direction', 'vehicles and fuel tanks',
                'cruise missiles'],
               dtype='object')

In [27]: dataset1.index

Out[27]: RangeIndex(start=0, stop=285, step=1)
```

To drop the column in dataset 2 and description:

```
To drop the column in dataset 2 and description
```

```
In [28]: dataset2.drop("POW",axis=1,inplace=True)
         dataset2
```

Out[28]:

	date	day	personnel	personnel*
0	2022-02-25	2	2800	about
1	2022-02-26	3	4300	about
2	2022-02-27	4	4500	about
3	2022-02-28	5	5300	about
4	2022-03-01	6	5710	about
...	...	...	...	...
280	2022-12-02	282	90090	about
281	2022-12-03	283	90600	about
282	2022-12-04	284	91150	about
283	2022-12-05	285	91690	about
284	2022-12-06	286	92200	about

285 rows × 4 columns

```
In [29]: dataset2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   date        285 non-null    object
 1   day         285 non-null    int64 
 2   personnel   285 non-null    int64 
 3   personnel*  285 non-null    object
dtypes: int64(2), object(2)
memory usage: 9.0+ KB
```

```
In [30]: dataset2.describe()
```

Out[30]:

	day	personnel
count	285.000000	285.000000
mean	144.000000	42256.722807
std	82.416625	22123.640833
min	2.000000	2800.000000
25%	73.000000	25100.000000
50%	144.000000	38300.000000

```
In [31]: dataset2.dtypes
```

```
Out[31]: date          object
         day            int64
         personnel      int64
         personnel*    object
         dtype: object
```

```
In [32]: dataset2.shape
```

```
Out[32]: (285, 4)
```

```
In [33]: dataset2.columns
```

```
Out[33]: Index(['date', 'day', 'personnel', 'personnel*'], dtype='object')
```

```
In [34]: dataset2.index
```

```
Out[34]: RangeIndex(start=0, stop=285, step=1)
```

Activity 3: Perform Data Sorting operations

3.1 A data frame can be sorted by the value of one of the variables. For example, you can sort by navel ship (use ascending=False to sort in descending order):

```
A DataFrame can be sorted by the value of one of the variables (i.e columns). For example, you can sort by
navel ship (use ascending=False to sort in descending order):
```

```
In [35]: # Sort the dataset1 by the 'navel_ship' column in descending order
dataset1_sorted = dataset1.sort_values(by='naval ship', ascending=False)
dataset1_sorted
```

Out[35]:

	date	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	naval ship	anti-aircraft warfare	special equipment	greatest losses direction	vehicles and fuel tanks	cruise missiles
284	2022-12-06	286	281	264	2929	5905	1915	395	NaN	NaN	1587	16	211	163.0	Bakhmut and Lyman	4505.0	592.0
257	2022-11-09	259	278	260	2801	5666	1802	393	NaN	NaN	1483	16	205	159.0	Bakhmut and Avdiivka	4227.0	399.0
255	2022-11-07	257	277	260	2771	5630	1782	391	NaN	NaN	1472	16	202	157.0	Lyman, Bakhmut and Avdiivka	4199.0	399.0
254	2022-11-06	256	277	260	2765	5611	1781	391	NaN	NaN	1465	16	202	155.0	Lyman and Avdiivka	4191.0	399.0
253	2022-11-05	255	277	260	2758	5601	1776	391	NaN	NaN	1462	16	202	155.0	Lyman and Avdiivka	4184.0	399.0

3.2 You can also sort by multiple columns ['Tank',' naval ship']

```
You can also sort by multiple columns [ 'Tank',' naval ship']
```

```
In [36]: # Sort the dataset1 by the 'navel_ship' column in descending order
dataset1_sorted = dataset1.sort_values(by=['tank','naval ship'], ascending=False)
dataset1_sorted
```

Out[36]:

	date	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	naval ship	anti-aircraft warfare	special equipment	greatest losses direction	vehicles and fuel tanks	cruise missiles
284	2022-12-06	286	281	264	2929	5905	1915	395	NaN	NaN	1587	16	211	163.0	Bakhmut and Lyman	4505.0	592.0
283	2022-12-05	285	281	264	2924	5900	1914	395	NaN	NaN	1582	16	211	163.0	Bakhmut and Lyman	4497.0	531.0
282	2022-12-04	284	281	263	2922	5892	1908	395	NaN	NaN	1573	16	210	163.0	Bakhmut and Lyman	4479.0	531.0
281	2022-12-03	283	280	263	2917	5886	1906	395	NaN	NaN	1572	16	210	163.0	NaN	4472.0	531.0
280	2022-12-02	282	280	262	2916	5883	1905	395	NaN	NaN	1564	16	210	163.0	Bakhmut and Avdiivka	4464.0	531.0

3.3 Find the proportion of days in our data frame

```
A DataFrame can be indexed in a few different ways. To get a single column, you can use a
DataFrame['Name'] construction. Let's use this to answer a question about that column alone: what
is the proportion of day in our dataframe?
```

```
In [37]: day= dataset1['day']
# Calculate the proportion of non-null values in the 'day' column
proportion_day = day.count() / dataset1.count().sum()*100
proportion_day
```

```
Out[37]: 6.841094575132021
```

3.4 What are the average aircraft(Aircraft feature) has been used ?

```
What are the average aircraft(Aircraft feature) has been used ?
```

```
In [38]: dataset1.aircraft.mean()
```

```
Out[38]: 210.02105263157895
```

3.5 What are mean value and standard deviation of the APC used after 50th day

```
What are mean value and standard deviation of the APC used after 50th day

In [39]: dataset1[dataset1.day>50]['APC'].mean()
Out[39]: 4195.533898305085

In [40]: dataset1[dataset1.day>50]['APC'].std()
Out[40]: 1052.7955868933843
```

3.6 Use crosstab method on “MRL” and “military auto”to show relation between them

```
Use crosstab method on "MRL" and "military auto"to show relation between them

In [41]: crosstab= pd.crosstab(dataset1['MRL'], dataset1['military auto'])
         crosstab
```

Out[41]:

military auto	100.0	130.0	291.0	305.0	355.0	374.0	404.0	409.0	447.0	454.0	...	1508.0	1523.0	1543.0	1557.0	1566.0	1643.0	1666.0	1688.0	1695.0	1701.0
MRL																					
4	1	2	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
21	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
24	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	1	1	1	1	...	0	0	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

3.7 DataFrames can be indexed by column name (label) or row name (index) or by the serial number of a row. The loc method is used for indexing by name, while iloc() is used for indexing by number. In the first case, we say "give us the values of the rows with index from 0 to 5 (inclusive) and columns labeled from aircraft to tank (inclusive)".

```
In [42]: dataset1.loc[0:5, 'aircraft':'tank']
```

Out[42]:

	aircraft	helicopter	tank
0	10	7	80
1	27	26	146
2	27	26	150
3	29	29	150
4	29	29	198
5	30	31	211

```
In [43]: # Accesses rows 0 to 5 and columns 0 to 3
         dataset1.iloc[0:6, 0:4]
```

Out[43]:

	date	day	aircraft	helicopter
0	2022-02-25	2	10	7
1	2022-02-26	3	27	26
2	2022-02-27	4	27	26
3	2022-02-28	5	29	29
4	2022-03-01	6	29	29
5	2022-03-02	7	30	31

3.8 In the second case, we say "give us the values of the first five rows in the first three columns" (as in a typical Python slice: the maximal value is not included).

```
In [44]: # Select the values of the first five rows in the first three columns
         dataset1.iloc[0:5, 0:3]
```

Out[44]:

	date	day	aircraft
0	2022-02-25	2	10
1	2022-02-26	3	27
2	2022-02-27	4	27
3	2022-02-28	5	29
4	2022-03-01	6	29

Activity 5: Data Visualization with Matplotlib

5.1 Task 1

5.1.1 View the content of the dataset1.csv file, it contains 285 entries and 18 column

```
View the content of the dataset1.csv file, it is containing 285 entries and 18 columns.

In [45]: dataset1=pd.read_csv("Dataset 1.csv")

In [46]: dataset1.shape
Out[46]: (285, 18)
```

5.1.2 Here you have to use import seaborn as sns library and sns.lineplot method

```
Here you have to use import seaborn as sns library and sns.lineplot method

In [47]: import seaborn as sns
```

5.1.3 Display the list of columns associated with the dataset1 by using .columns property

```
Display the list of columns associated with the dataset1 by using .columns property

In [48]: dataset1.columns
Out[48]: Index(['date', 'day', 'aircraft', 'helicopter', 'tank', 'APC',
       'field artillery', 'MRL', 'military auto', 'fuel tank', 'drone',
       'naval ship', 'anti-aircraft warfare', 'special equipment',
       'mobile SRBM system', 'greatest losses direction',
       'vehicles and fuel tanks', 'cruise missiles'],
      dtype='object')

In [49]: dataset1["date"]=pd.to_datetime(dataset1.date)

In [50]: dataset1_monthly=dataset1.resample("M",on="date").sum(numeric_only=True)
         dataset1_monthly
```

Out[50]:

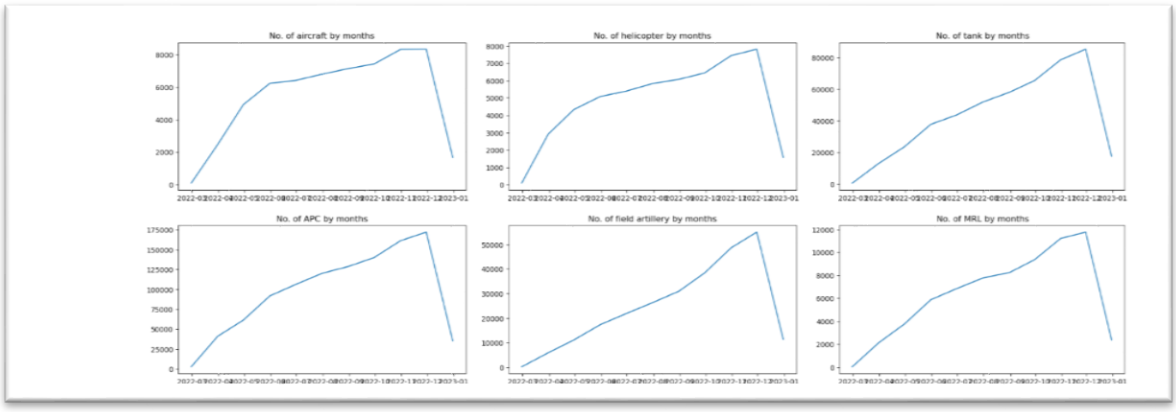
date	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	naval ship	anti-aircraft warfare	special equipment	mobile SRBM system	vehicles and fuel tanks	cruise missiles
2022-02-28	14	93	88	526	2744	222	33	651.0	240.0	7	8	5	0.0	0.0	0.0	0.0
2022-03-31	651	2472	2913	12998	41083	5928	2123	23740.0	1986.0	774	115	1129	256.0	22.0	0.0	0.0
2022-																

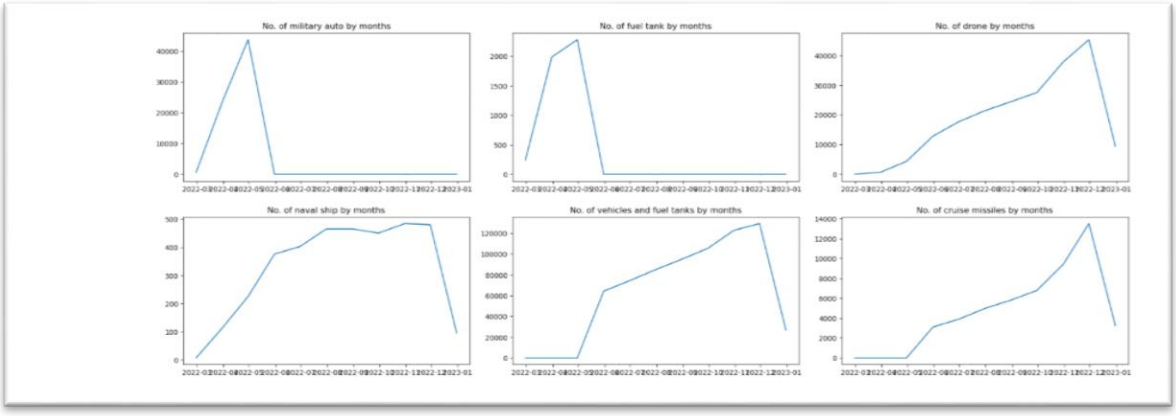
5.1.4 Use plot() and show() method to visualize and identify the pattern of each attribute from the Dataset

```
Use plot() and show() method to visualize and identify the pattern of each attribute from the
dataset.

In [53]: # Assuming you have a list of attribute names to plot
attribute_names = ["aircraft", "helicopter", "tank", "APC", "field artillery", "MRL", "military auto",
                   "fuel tank", "drone", "naval ship", "vehicles and fuel tanks", "cruise missiles"]
# Create a 4x3 grid of subplots
fig, axes = plt.subplots(4, 3, figsize=(20, 15))
fig.tight_layout(pad=3.0)  # Add some padding between subplots
# Flatten the 2D array of axes for easy iteration
axes = axes.flatten()
# Iterate through attribute names and plot them
for i, attribute in enumerate(attribute_names):
    ax = axes[i]  # Get the current subplot
    ax.plot(dataset1_monthly.index, dataset1_monthly[attribute])
    ax.set_title(f"No. of {attribute} by months")
# Hide any remaining empty subplots
for i in range(len(attribute_names), len(axes)):
    axes[i].axis('off')

plt.show()
```





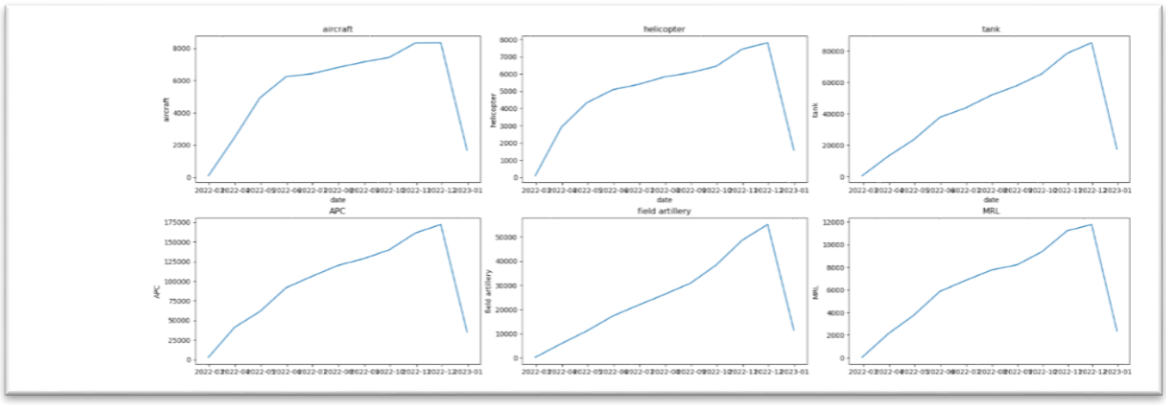
5.1.5 Air vehicles such as aircraft and helicopters are the primary weapons of assault for Russia in the battle; yet, as we have seen from the dataset pattern, the Russians have been experiencing a significant loss of air vehicles in recent conflicts. Use line charts/plot plot 1 by 1 and all in one to provide a visual representation of the loss of air vehicles, such as aircraft and helicopters with all the possible parameters of sns.lineplot.

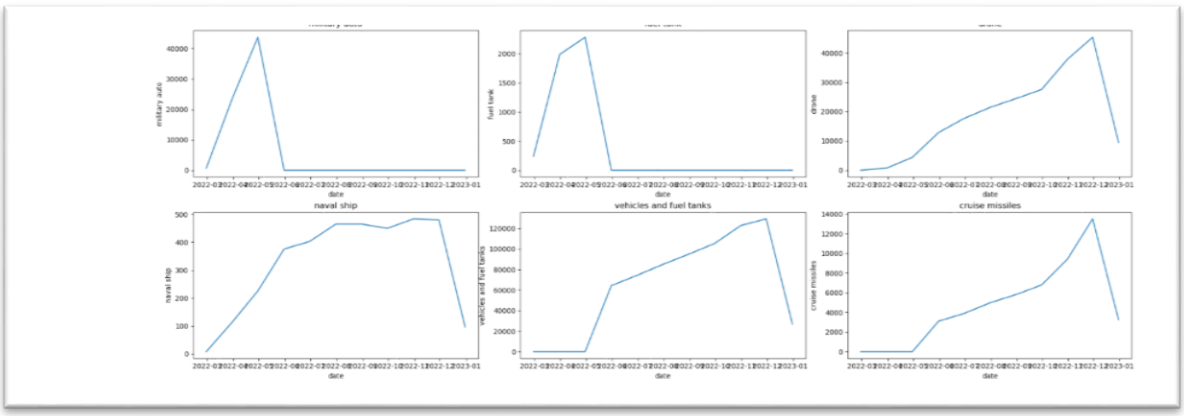
line plot 1 by 1:

```
line plot 1 by 1
```

```
In [54]: # Assuming you have a list of attribute names to plot
attribute_names = ["aircraft", "helicopter", "tank", "APC", "field artillery", "MRL", "military auto",
                   "fuel tank", "drone", "naval ship", "vehicles and fuel tanks", "cruise missiles"]
# Create a 4x3 grid of subplots

fig, axes = plt.subplots(4, 3, figsize=(20, 15))
fig.tight_layout(pad=3.0)  # Add some padding between subplots
# Flatten the 2D array of axes for easy iteration
axes = axes.flatten()
# Iterate through attribute names and plot them using sns.lineplot
for i, attribute in enumerate(attribute_names):
    ax = axes[i]  # Get the current subplot
    sns.lineplot(x=dataset1_monthly.index, y=dataset1_monthly[attribute], data=dataset1_monthly, ax=ax)
    ax.set_title(f"{attribute}")
# Hide any remaining empty subplots
for i in range(len(attribute_names), len(axes)):
    axes[i].axis('off')
plt.show()
```

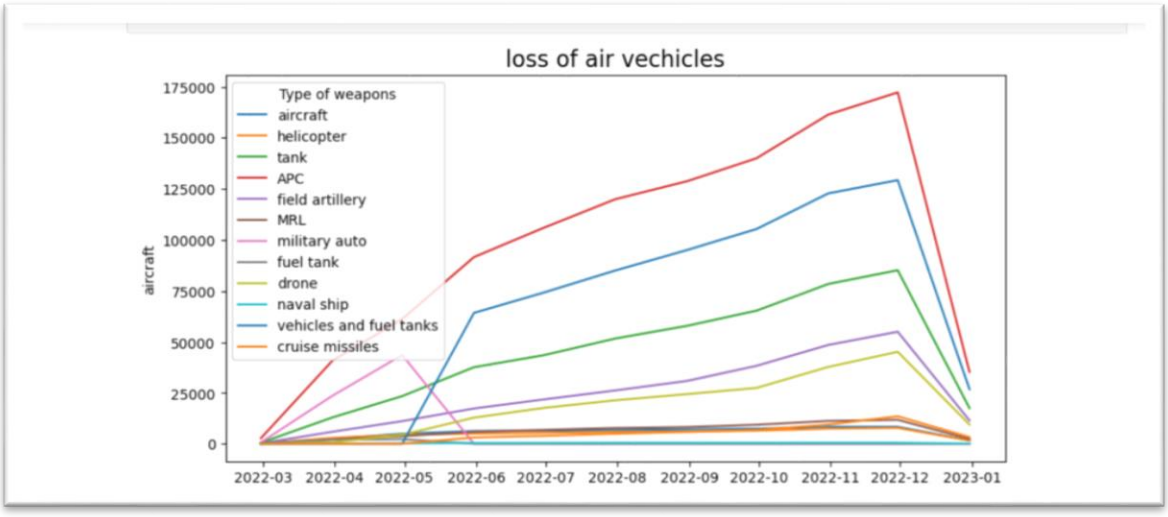




line plot all in one:

```
line plot all in one

In [55]: # Assuming you have a list of attribute names to plot
attribute_names = ["aircraft", "helicopter", "tank", "APC", "field artillery", "MRL", "military auto",
                   "fuel tank", "drone", "naval ship", "vehicles and fuel tanks", "cruise missiles"]
# Create a figure with a specified size
plt.figure(figsize=(10,10))
# Iterate through attribute names and plot them using sns.lineplot
for attribute in attribute_names:
    sns.lineplot(x=dataset1_monthly.index, y=dataset1_monthly[attribute], data=dataset1_monthly, label=attribute)
plt.title("loss of air vechicles", fontsize=16)
plt.legend(title="Type of weapons",fontsize='medium')
plt.show()
```



5.2 Task 2

5.2.1 Define what scatter plot with syntax and the parameter associated with it.

```
Define what scatter plot with syntax and the parameter associate with it.
      Data points are graphically represented as scatter plots on a two-dimensional plane. For illustrating the link between
two continuous variables, it is especially helpful. With one variable shown on the x-axis and another on the y-axis, each data
point is shown as a dot or marker. A scatter plot is necessary for:
1.Finding Relationships: Scatter plots aid in figuring out the nature of a link between two variables. Is the association
substantial, negative, or positive (when one variable rises, the other rises as well)?
2.Finding Outliers: On a scatter plot, outliers, or data points that differ markedly from the norm, are simple to spot.
Data points tend to gather together in patterns called clustering, which can be seen.
# Syntax:
plt.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, edgecolors=None,
linewidths=None, label=None)
Parameters:
x: A sequence of values representing the x-coordinates of the data points.
y: A sequence of values representing the y-coordinates of the data points.
s (optional): The size of the markers (dots). It can be a scalar or an array specifying the size of each marker.
c (optional): The color of the markers. It can be a single color or a sequence of colors.
marker (optional): The marker style to use for the data points (e.g., 'o' for circles, 's' for squares).
cmap (optional): A colormap for mapping data values to colors when 'c' is an array of numeric values.
norm (optional): A Normalize instance for scaling data values to the interval [0, 1] when 'c' is specified.
vmin, vmax (optional): The minimum and maximum values for colormap normalization when 'c' is specified.
alpha (optional): The transparency of the markers (0.0 for fully transparent, 1.0 for fully opaque).
edgecolors (optional): The color of the marker edges.
linewidths (optional): The width of the marker edges.
label (optional): A label for the data points, used in the legend when creating multiple plots.
```

5.2.2 In addition to the Air-Vehicle, Russia has lost a large number of other weapons, including Tanks, Armoured Personnel Carriers (APCs), Field Artillery, Multiple Rocket Launchers, military automobiles, aircraft, and helicopters; visualize all of these weapons using scatter plot 1 by 1 and all in one scatterplot that includes all of the possible parameters of Scatter plot

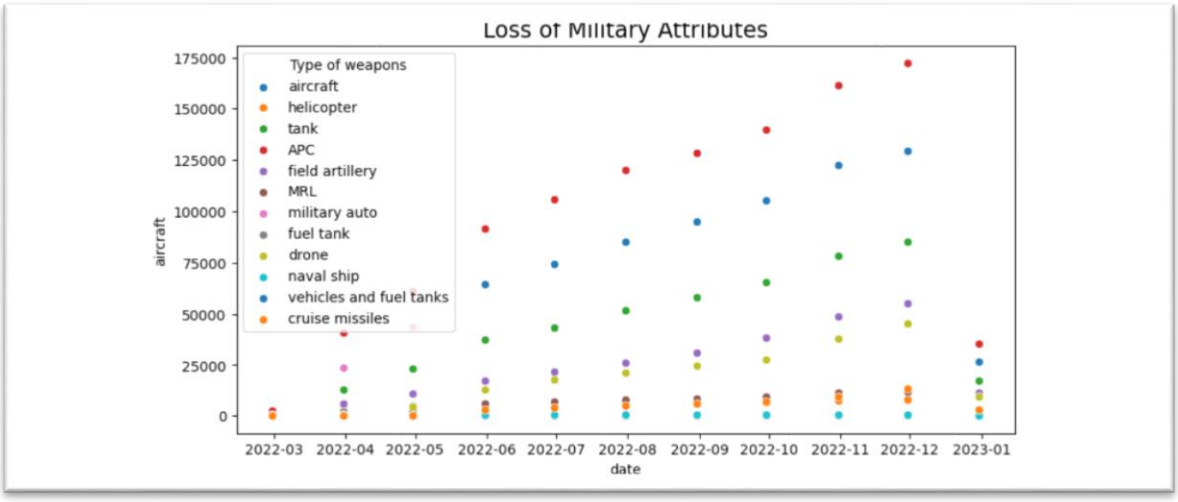
scatter plot all in one:

```
In addition to the Air-Vehicle, Russia has lost a large number of other weapons, including Tanks,
Armoured Personnel Carriers (APCs), field Artillery, Multiple Rocket Launchers, militaryautomobiles, aircraft, and helicopters;
visualize all of these weapons using a scatter plot 1 by 1 and all in one scatterplot that includes all of the possible
parameters of Scatter plot
```

```
scatter plot all in one
```

```
In [56]: # Assuming you have a list of attribute names to plot
attribute_names = ["aircraft", "helicopter", "tank", "APC", "field artillery", "MRL", "military auto",
                   "fuel tank", "drone", "naval ship", "vehicles and fuel tanks", "cruise missiles"]
# Create a figure with a specified size
plt.figure(figsize=(10, 10))
# Iterate through attribute names and plot them using sns.scatterplot
for attribute in attribute_names:
    sns.scatterplot(x=dataset1_monthly.index, y=dataset1_monthly[attribute], data=dataset1_monthly, label=attribute)

plt.title("Loss of Military Attributes", fontsize=16)
plt.legend(title="Type of weapons",fontsize='medium')
plt.show()
```

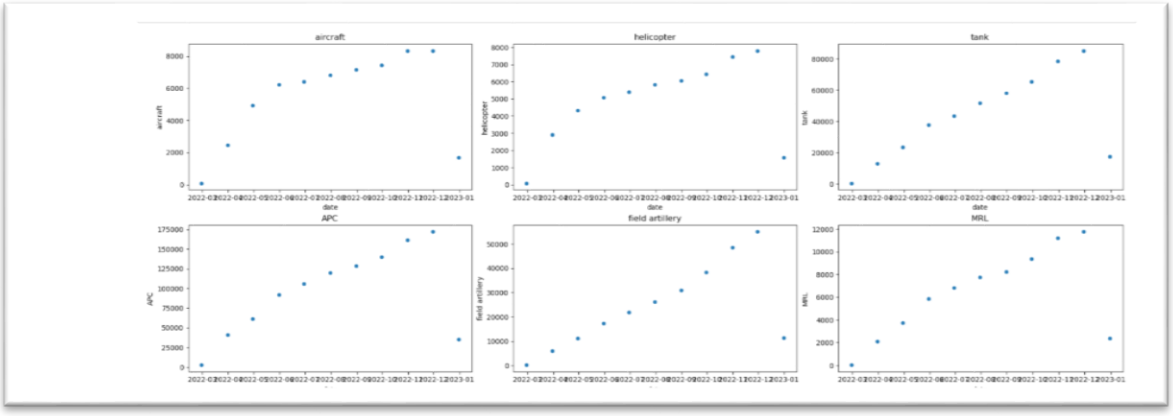


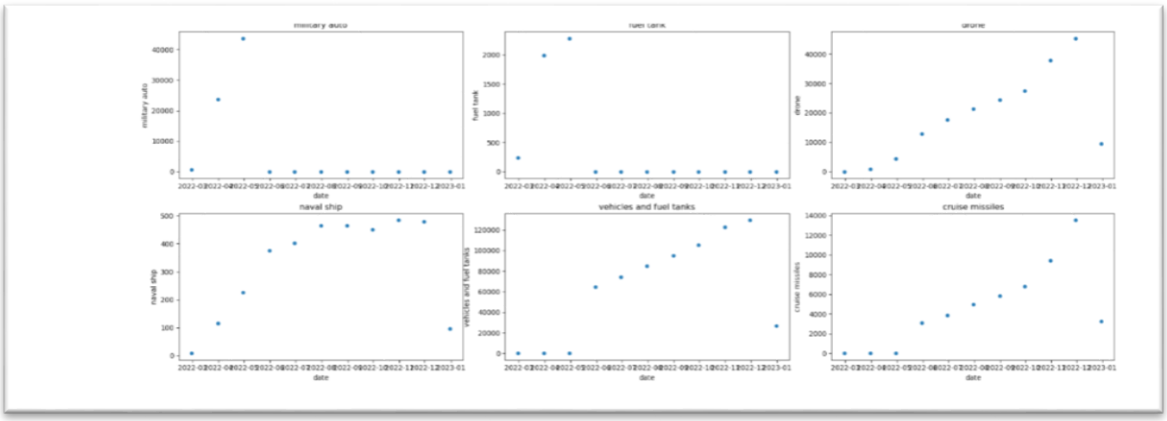
scatter plot 1 by 1:

```
scatter plot 1 by 1
```

```
In [57]: # Assuming you have a list of attribute names to plot
attribute_names = ["aircraft", "helicopter", "tank", "APC", "field artillery", "MRL", "military auto",
                   "fuel tank", "drone", "naval ship", "vehicles and fuel tanks", "cruise missiles"]
# Create a 4x3 grid of subplots

fig, axes = plt.subplots(4, 3, figsize=(20, 15))
fig.tight_layout(pad=3.0)  # Add some padding between subplots
# Flatten the 2D array of axes for easy iteration
axes = axes.flatten()
# Iterate through attribute names and plot them using sns.lineplot
for i, attribute in enumerate(attribute_names):
    ax = axes[i]  # Get the current subplot
    sns.scatterplot(x=dataset1_monthly.index, y=dataset1_monthly[attribute], data=dataset1_monthly, ax=ax)
    ax.set_title(f"{attribute}")
# Hide any remaining empty subplots
for i in range(len(attribute_names), len(axes)):
    axes[i].axis('off')
plt.show()
```





5.3 Task 3

5.3.1Define whatis correlation according to you in 1-2 line

```
Define whatis correlation according to you in 1-2 line
Correlation is a statistical measure that quantifies the degree and direction of the linear relationship between two or more
variables. It indicates how changes in one variable are associated with changes in another.

1.A positive correlation implies that as one variable increases, the other also tends to increase.
2.A negative correlation suggests that as one variable increases, the other tends to decrease.
3.A correlation of zero indicates no linear relationship between the variables.

Correlation coefficients, such as the Pearson correlation coefficient, are commonly used to express the strength and direction
of correlation, ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation), with 0 indicating no
correlation. Correlation analysis is essential in statistics, data analysis, and various fields to understand relationships
between variables.
```

5.3.2 To solve this tasks you have to use Sns.heatmap along with merge and corr

```
In [75]: #merge the dataset1,dataset2
dataset_merge=pd.merge(dataset1,dataset2,on='day')
dataset_merge
```

Out[75]:

	date_x	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	...	anti-aircraft warfare	special equipment	mobile SRBM system	greatest losses direction	vehicles and fuel tanks	cruise missiles	date_y	personne
0	2022-02-25	2	10	7	80	516	49	4	100.0	60.0	...	0	NaN	NaN	NaN	NaN	NaN	2022-02-25	280
1	2022-02-26	3	27	26	146	706	49	4	130.0	60.0	...	0	NaN	NaN	NaN	NaN	NaN	2022-02-26	430
2	2022-02-27	4	27	26	150	706	50	4	130.0	60.0	...	0	NaN	NaN	NaN	NaN	NaN	2022-02-27	450
3	2022-02-28	5	29	29	150	816	74	21	291.0	60.0	...	5	NaN	NaN	NaN	NaN	NaN	2022-02-28	530
4	2022-03-01	6	29	29	198	846	77	24	305.0	60.0	...	7	NaN	NaN	NaN	NaN	NaN	2022-03-01	571
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
280	2022-12-02	282	280	262	2916	5883	1905	395	NaN	NaN	...	210	163.0	NaN	Bakhmut and Avdiivka	4464.0	531.0	2022-12-02	9009
281	2022-12-03	283	280	263	2917	5886	1906	395	NaN	NaN	...	210	163.0	NaN	NaN	4472.0	531.0	2022-12-03	9060
	~~														Bakhmut			~~~~	

```
In [61]: dataset_numeric=dataset_merge.select_dtypes(include=["int64","float64"])
In [62]: dataset_numeric
```

Out[62]:

	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	naval ship	anti-aircraft warfare	special equipment	mobile SRBM system	vehicles and fuel tanks	cruise missiles	personnel	POW
0	2	10	7	80	516	49	4	100.0	60.0	0	2	0	NaN	NaN	NaN	NaN	2800	0.0
1	3	27	26	146	706	49	4	130.0	60.0	2	2	0	NaN	NaN	NaN	NaN	4300	0.0
2	4	27	26	150	706	50	4	130.0	60.0	2	2	0	NaN	NaN	NaN	NaN	4500	0.0
3	5	29	29	150	816	74	21	291.0	60.0	3	2	5	NaN	NaN	NaN	NaN	5300	0.0
4	6	29	29	198	846	77	24	305.0	60.0	3	2	7	NaN	NaN	NaN	NaN	5710	200.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
280	282	280	262	2916	5883	1905	395	NaN	NaN	1564	16	210	163.0	NaN	4464.0	531.0	90090	NaN
281	283	280	263	2917	5886	1906	395	NaN	NaN	1572	16	210	163.0	NaN	4472.0	531.0	90600	NaN
282	284	281	263	2922	5892	1908	395	NaN	NaN	1573	16	210	163.0	NaN	4479.0	531.0	91150	NaN
283	285	281	264	2924	5900	1914	395	NaN	NaN	1582	16	211	163.0	NaN	4497.0	531.0	91690	NaN
284	286	281	264	2929	5905	1915	395	NaN	NaN	1587	16	211	163.0	NaN	4505.0	592.0	92200	NaN

5.3.3Find out how each column is connected to the others since correlations are nothing more than determining how closely two variables are related. Since a correlation is nothing more than determining how closely two variables are related, find out how each column is related to the others by using corr method

```
In [63]: dataset_numeric.corr()
```

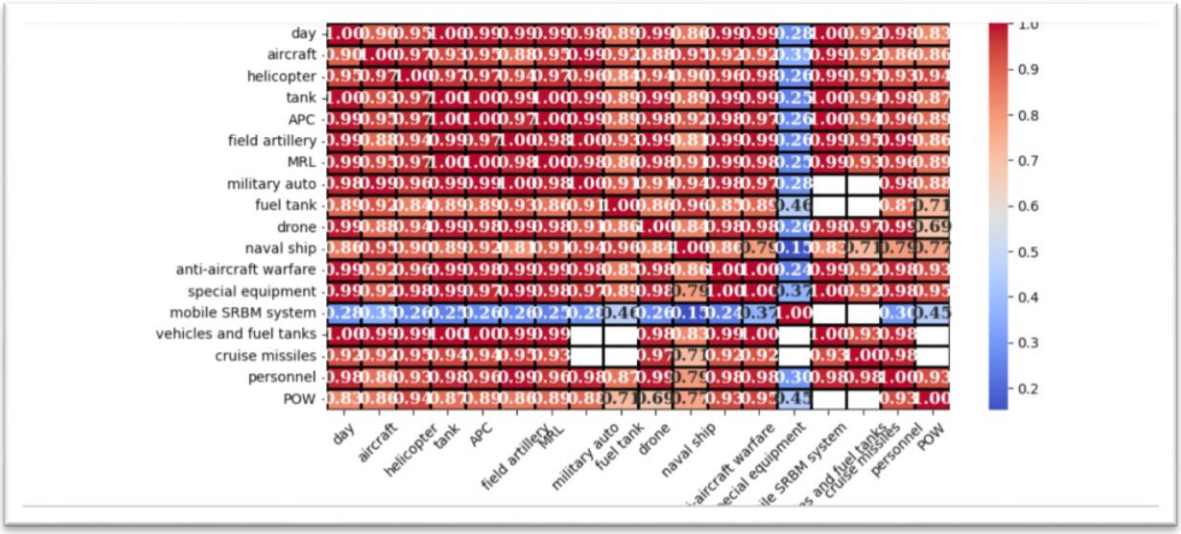
Out[63]:

	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	naval ship	anti-aircraft warfare	special equipment	mobile SRBM system	vehi and t
day	1.000000	0.902627	0.945223	0.995043	0.987652	0.990050	0.985816	0.979363	0.890240	0.985304	0.860192	0.991955	0.992070	0.284747	0.99
aircraft	0.902627	1.000000	0.974623	0.931273	0.948758	0.881600	0.948378	0.993953	0.921081	0.883633	0.952457	0.920741	0.921511	0.347026	0.99
helicopter	0.945223	0.974623	1.000000	0.965292	0.971078	0.941753	0.973233	0.962985	0.841115	0.936588	0.903286	0.963746	0.982749	0.258891	0.98
tank	0.995043	0.931273	0.965292	1.000000	0.996569	0.988838	0.996170	0.988725	0.887154	0.988083	0.886850	0.993345	0.987678	0.247350	0.99
APC	0.987652	0.948758	0.971078	0.996569	1.000000	0.974949	0.997560	0.993803	0.886196	0.979055	0.917941	0.984740	0.974112	0.263685	0.99
field artillery	0.990050	0.881600	0.941753	0.988838	0.974949	1.000000	0.977971	0.996674	0.929310	0.992124	0.814901	0.989687	0.994727	0.262446	0.99
MRL	0.985816	0.948378	0.973233	0.996170	0.997560	0.977971	1.000000	0.979341	0.864099	0.979980	0.910359	0.986309	0.976808	0.245700	0.99
military auto	0.979363	0.993953	0.962985	0.988725	0.993803	0.996674	0.979341	1.000000	0.912906	0.913692	0.941556	0.981397	0.971746	0.283902	
fuel tank	0.890240	0.921081	0.841115	0.887154	0.886196	0.929310	0.864099	0.912906	1.000000	0.857450	0.958101	0.854913	0.889583	0.464120	
drone	0.985304	0.883633	0.936588	0.988083	0.979055	0.992124	0.979980	0.913692	0.857450	1.000000	0.841406	0.977920	0.978551	0.261989	0.97
naval ship	0.860192	0.952457	0.903286	0.886850	0.917941	0.814901	0.910359	0.941556	0.958101	0.841406	1.000000	0.855097	0.786761	0.151186	0.82
anti-aircraft warfare	0.991955	0.920741	0.963746	0.993345	0.984740	0.989687	0.986309	0.981397	0.854913	0.977920	0.855097	1.000000	0.996686	0.238928	0.99

5.3.4You can plot heatmap to identify the correlation by using heatmap method

```
You can plot heatmap to identify the correlation by using heatmap method
```

```
In [64]: plt.figure(figsize=(16,10))
sns.heatmap(dataset_numeric.corr(),cmap="coolwarm",annot=True,fmt="0.2f",annot_kws={"fontsize":12,"fontweight":"bold","fontfamily
plt.xticks(rotation=45)
plt.title("correlation HeatMap")
plt.show()
```



5.4 Task 4

5.4.1 Define hist method with syntax

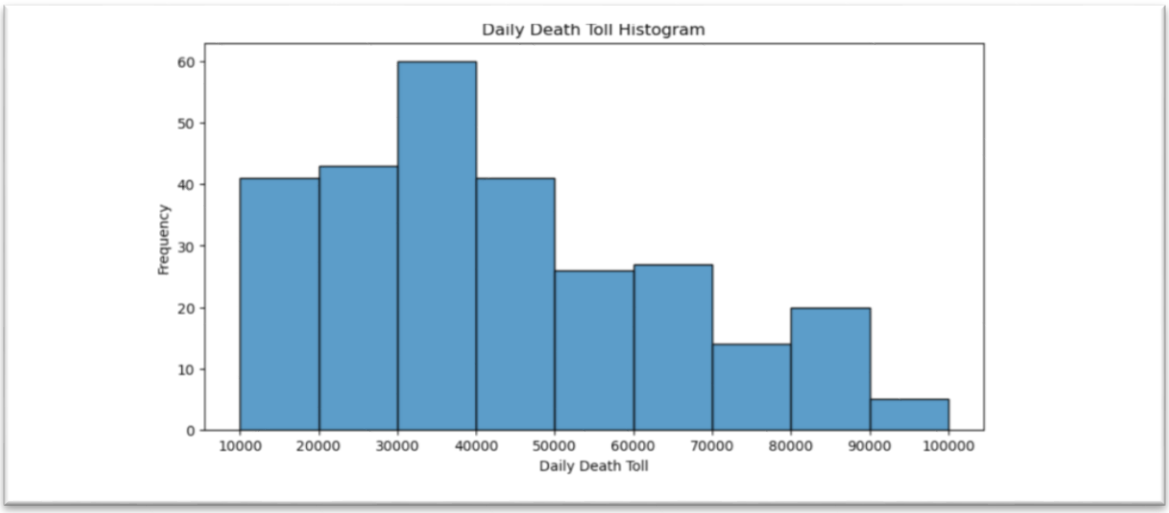
```
Define hist method with syntax
A histogram is a graphical representation of data distribution, displaying the frequency of data points within predefined
intervals or bins. The x-axis represents the data range divided into bins, while the y-axis shows the count or frequency of data
points in each bin. Histograms help analyze data shape, central tendencies, spread, and identify outliers. They are crucial for
visualizing data distributions, making data-driven decisions, and spotting patterns or anomalies. Histograms are widely used in
statistics, data analysis, and data visualization.
syntax:

plt.hist(x, bins=None, range=None, density=False, cumulative=False, histtype='bar', align='mid', color=None, label=None,
stacked=False)

Parameters:
x: Numeric data to create the histogram from.
bins (optional): Number of bins or bin edges for the histogram.
range (optional): The range of values to include in the histogram.
density (optional): If True, the histogram represents a probability density.
cumulative (optional): If True, the histogram is cumulative.
histtype (optional): Type of histogram ('bar', 'barstacked', 'step', 'stepfilled').
align (optional): Alignment of the bins ('left', 'mid', 'right').
color (optional): Color of the bars.
label (optional): Label for the histogram (used in legends).
stacked (optional): If True, multiple histograms are stacked on top of each other.
```

5.4.2 Since we have also read the other dataset, which is designated as Dataset2, Determine the dailydeath toll(the number of people who died in the war) by taking into account the date and the attribute personnel

```
In [66]: plt.figure(figsize=(10,5))
sns.histplot(dataset2.personnel,bins=[10000,20000,30000,40000,50000,60000,70000,80000,90000,100000])
plt.xticks([10000,20000,30000,40000,50000,60000,70000,80000,90000,100000])
plt.title("Daily Death Toll Histogram")
plt.xlabel("Daily Death Toll")
plt.ylabel("Frequency")
plt.show()
```



5.5 Task 5

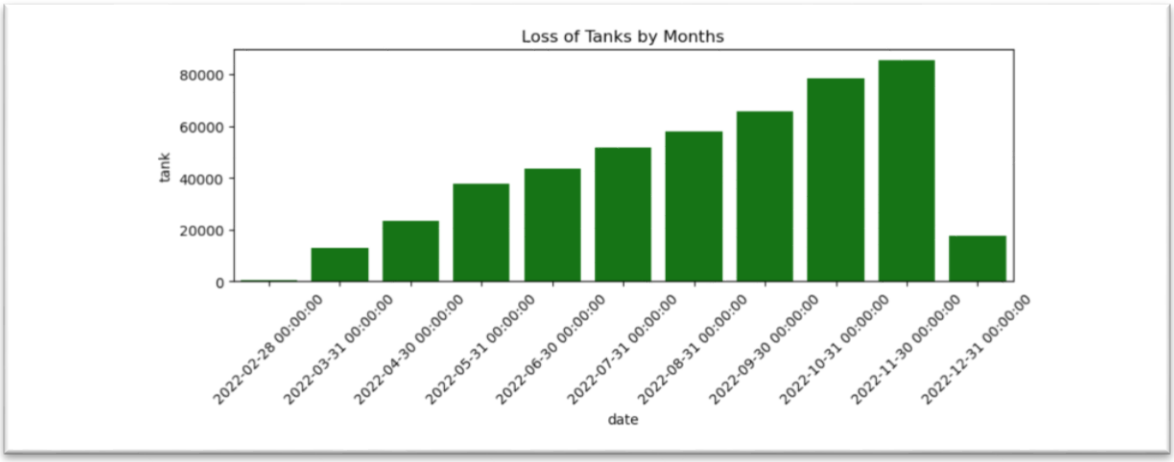
5.5.1Define bar plot with syntax

```
Define bar plot with syntax
A bar plot is a graphical representation of data in which rectangular bars are used to represent categories or data points. The
length or height of each bar is proportional to the value it represents. Bar plots are typically used for displaying categorical
data or comparing values across different categories.
syntax:
sns.barplot(x=None, y=None, hue=None, data=None, palette=None, ci=None, capsize=None, orient=None)
Parameters:
x: Categorical data to be displayed on the x-axis.
y: Numeric data to be displayed on the y-axis (height of bars).
hue (optional): Categorical data to create grouped bars based on a third variable.
data: DataFrame or data source containing the data.
palette (optional): Color palette for the bars.
ci (optional): Confidence interval for error bars.
capsize (optional): Width of the caps at the end of error bars.
orient (optional): Orientation of the plot ('v' for vertical, 'h' for horizontal).
```

5.5.2 Find the loss of tanks and APC through the Bar plot

Loss byTanks:

```
In [68]: #Loss of tanks
plt.figure(figsize=(10,5))
sns.barplot(x=dataset1_monthly.index,y=dataset1_monthly.tank,data=dataset1_monthly,color="green")
plt.xticks(rotation=45)
plt.title("Loss of Tanks by Months")
plt.show()
```



Loss by APC:

```
In [69]: #Loss of APC
         plt.figure(figsize=(10,5))
         sns.barplot(x=dataset1_monthly.index,y=dataset1_monthly.APC,data=dataset1_monthly,color="red")
         plt.xticks(rotation=45)
         plt.title("Loss of APC by Months")
         plt.show()
```