Megaraj Mrittika(Russia Ukraine -The War Analysis and Visualization)

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")
```

```
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	a v
0	2022-02-25	2	10	7	80	516	49	4	100.0	60.0	0	2	
1	2022-02-26	3	27	26	146	706	49	4	130.0	60.0	2	2	
2	2022-02-27	4	27	26	150	706	50	4	130.0	60.0	2	2	
3	2022-02-28	5	29	29	150	816	74	21	291.0	60.0	3	2	
4	2022-03-01	6	29	29	198	846	77	24	305.0	60.0	3	2	
5	2022-03-02	7	30	31	211	862	85	40	355.0	60.0	3	2	

```
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
279	2022-12-01	281	280	261	2915	5877	1904	395	NaN	NaN	1562	16
280	2022-12-02	282	280	262	2916	5883	1905	395	NaN	NaN	1564	16
281	2022-12-03	283	280	263	2917	5886	1906	395	NaN	NaN	1572	16
282	2022-12-04	284	281	263	2922	5892	1908	395	NaN	NaN	1573	16
283	2022-12-05	285	281	264	2924	5900	1914	395	NaN	NaN	1582	16
284	2022-12-06	286	281	264	2929	5905	1915	395	NaN	NaN	1587	16

```
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
284	2022-12-06	286	92200	about	NaN

```
Find the Dataset 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)
memory usage: 40.2+ KB
```

In [8]:

```
dataset1.describe()
```

Out[8]:

	day	aircraft	helicopter	tank	APC	field artillery	MI
count	285.000000	285.000000	285.000000	285.000000	285.000000	285.000000	285.0000
mean	144.000000	210.021053	185.912281	1665.957895	3720.287719	938.136842	243.2175
std	82.416625	62.500419	53.116693	782.804694	1427.345027	541.040253	105.1305
min	2.000000	10.000000	7.000000	80.000000	516.000000	49.000000	4.0000
25%	73.000000	199.000000	155.000000	1122.000000	2713.000000	509.000000	172.0000
50%	144.000000	220.000000	188.000000	1684.000000	3879.000000	846.000000	248.0000
75%	215.000000	260.000000	224.000000	2290.000000	4857.000000	1369.000000	330.0000
max	286.000000	281.000000	264.000000	2929.000000	5905.000000	1915.000000	395.0000

In [9]:

```
dataset1.dtypes
```

Out[9]:

```
date                         object
day                           int64
aircraft                      int64
helicopter                    int64
tank                          int64
APC                           int64
field artillery               int64
MRL                           int64
military auto               float64
fuel tank                   float64
drone                         int64
naval ship                    int64
anti-aircraft warfare         int64
special equipment           float64
mobile SRBM system          float64
greatest losses direction    object
vehicles and fuel tanks     float64
cruise missiles             float64
dtype: object
```

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)
```

```
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
25%	73.000000	25100.000000	389.000000
50%	144.000000	38300.000000	421.000000
75%	215.000000	57200.000000	474.500000
max	286.000000	92200.000000	496.000000

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)
```

```
Null values sometimes decrees the performance and give us bad result to
avoid this please
perform the exploratory Data analysis i.e. find the null values from the
dataset.
```

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           249
greatest losses direction     90
vehicles and fuel tanks       65
cruise missiles               65
dtype: int64
```

In [20]:

```
dataset2.isnull().sum()
```

Out[20]:

```
date            0
day             0
personnel       0
personnel*      0
POW           223
dtype: int64
```

Drop the column with maximum NULL Values (if any) and Get the Dataset 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  MRL  military  fuel  drone  naval
                                                       artillery         auto  tank          ship
  0  2022-02-25    2        10           7    80   516        49    4     100.0  60.0      0      2
  1  2022-02-26    3        27          26   146   706        49    4     130.0  60.0      2      2
  2  2022-02-27    4        27          26   150   706        50    4     130.0  60.0      2      2
  3  2022-02-28    5        29          29   150   816        74   21     291.0  60.0      3      2
  4  2022-03-01    6        29          29   198   846        77   24     305.0  60.0      3      2
...         ...  ...       ...         ...   ...   ...       ...  ...       ...   ...    ...    ...
280  2022-12-02  282       280         262  2916  5883      1905  395       NaN   NaN   1564     16
281  2022-12-03  283       280         263  2917  5886      1906  395       NaN   NaN   1572     16
282  2022-12-04  284       281         263  2922  5892      1908  395       NaN   NaN   1573     16
283  2022-12-05  285       281         264  2924  5900      1914  395       NaN   NaN   1582     16
284  2022-12-06  286       281         264  2929  5905      1915  395       NaN   NaN   1587     16

285 rows × 17 columns
```

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 [23]:
```
dataset1.describe()
```

Out[23]:

	day	aircraft	helicopter	tank	APC	field artillery	MI
count	285.000000	285.000000	285.000000	285.000000	285.000000	285.000000	285.0000
mean	144.000000	210.021053	185.912281	1665.957895	3720.287719	938.136842	243.2175
std	82.416625	62.500419	53.116693	782.804694	1427.345027	541.040253	105.1305
min	2.000000	10.000000	7.000000	80.000000	516.000000	49.000000	4.0000
25%	73.000000	199.000000	155.000000	1122.000000	2713.000000	509.000000	172.0000
50%	144.000000	220.000000	188.000000	1684.000000	3879.000000	846.000000	248.0000
75%	215.000000	260.000000	224.000000	2290.000000	4857.000000	1369.000000	330.0000
max	286.000000	281.000000	264.000000	2929.000000	5905.000000	1915.000000	395.0000

In [24]:

```
dataset1.dtypes
```

Out[24]:

```
date                          object
day                            int64
aircraft                       int64
helicopter                     int64
tank                           int64
APC                            int64
field artillery                int64
MRL                            int64
military auto                float64
fuel tank                    float64
drone                          int64
naval ship                     int64
anti-aircraft warfare          int64
special equipment            float64
greatest losses direction     object
vehicles and fuel tanks      float64
cruise missiles              float64
dtype: object
```

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

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
75%	215.000000	57200.000000
max	286.000000	92200.000000

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)
```

Perform data sorting operations

```
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
284	2022-12-06	286	281	264	2929	5905	1915	395	NaN	NaN	1587	16
257	2022-11-09	259	278	260	2801	5666	1802	393	NaN	NaN	1483	16
255	2022-11-07	257	277	260	2771	5630	1782	391	NaN	NaN	1472	16
254	2022-11-06	256	277	260	2765	5611	1781	391	NaN	NaN	1465	16
253	2022-11-05	255	277	260	2758	5601	1776	391	NaN	NaN	1462	16
...	...	...	...	...	...	...	...	...	...	...	...	...
6	2022-03-03	8	30	31	217	900	90	42	374.0	60.0	3	2
7	2022-03-04	9	33	37	251	939	105	50	404.0	60.0	3	2
8	2022-03-05	10	39	40	269	945	105	50	409.0	60.0	3	2
9	2022-03-06	11	44	48	285	985	109	50	447.0	60.0	4	2
0	2022-02-25	2	10	7	80	516	49	4	100.0	60.0	0	2

285 rows × 17 columns

```
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
dataset1_sorted
```

	12-04											
281	2022-12-03	283	280	263	2917	5886	1906	395	NaN	NaN	1572
280	2022-12-02	282	280	262	2916	5883	1905	395	NaN	NaN	1564
...	...	...	...	...	...	...	...	...	...	...	...
4	2022-03-01	6	29	29	198	846	77	24	305.0	60.0	3
2	2022-02-27	4	27	26	150	706	50	4	130.0	60.0	2
3	2022-02-28	5	29	29	150	816	74	21	291.0	60.0	3
1	2022-02-26	3	27	26	146	706	49	4	130.0	60.0	2
	2022										

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
```

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

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

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

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
```

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	152
MRL													
4	1	2	0	0	0	0	0	0	0	0	...	0	
21	0	0	1	0	0	0	0	0	0	0	...	0	
24	0	0	0	1	0	0	0	0	0	0	...	0	
40	0	0	0	0	1	0	0	0	0	0	...	0	
42	0	0	0	0	0	1	0	0	0	0	...	0	
50	0	0	0	0	0	0	1	1	1	1	...	0	
56	0	0	0	0	0	0	0	0	0	0	...	0	
58	0	0	0	0	0	0	0	0	0	0	...	0	
62	0	0	0	0	0	0	0	0	0	0	...	0	
64	0	0	0	0	0	0	0	0	0	0	...	0	
70	0	0	0	0	0	0	0	0	0	0	...	0	
72	0	0	0	0	0	0	0	0	0	0	...	0	
74	0	0	0	0	0	0	0	0	0	0	...	0	
80	0	0	0	0	0	0	0	0	0	0	...	0	
82	0	0	0	0	0	0	0	0	0	0	...	0	
90	0	0	0	0	0	0	0	0	0	0	...	0	
91	0	0	0	0	0	0	0	0	0	0	...	0	
93	0	0	0	0	0	0	0	0	0	0	...	0	
95	0	0	0	0	0	0	0	0	0	0	...	0	
96	0	0	0	0	0	0	0	0	0	0	...	0	
100	0	0	0	0	0	0	0	0	0	0	...	0	
105	0	0	0	0	0	0	0	0	0	0	...	0	
107	0	0	0	0	0	0	0	0	0	0	...	0	
108	0	0	0	0	0	0	0	0	0	0	...	0	
111	0	0	0	0	0	0	0	0	0	0	...	0	
115	0	0	0	0	0	0	0	0	0	0	...	0	
122	0	0	0	0	0	0	0	0	0	0	...	0	
125	0	0	0	0	0	0	0	0	0	0	...	0	
127	0	0	0	0	0	0	0	0	0	0	...	0	
130	0	0	0	0	0	0	0	0	0	0	...	0	
132	0	0	0	0	0	0	0	0	0	0	...	0	
136	0	0	0	0	0	0	0	0	0	0	...	1	
138	0	0	0	0	0	0	0	0	0	0	...	0	
143	0	0	0	0	0	0	0	0	0	0	...	0	
147	0	0	0	0	0	0	0	0	0	0	...	0	
149	0	0	0	0	0	0	0	0	0	0	...	0	
151	0	0	0	0	0	0	0	0	0	0	...	0	

37 rows × 64 columns

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

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

Data Visualization with Matplotlib

```
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)
```

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

In [47]:

```
import seaborn as sns
```

```
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]:

	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	nav sh
date											
2022-02-28	14	93	88	526	2744	222	33	651.0	240.0	7	
2022-03-31	651	2472	2913	12998	41083	5928	2123	23740.0	1986.0	774	1
2022-04-30	1545	4926	4331	23478	61343	11136	3749	43697.0	2280.0	4397	2
2022-05-31	2542	6237	5086	37713	91653	17371	5864	0.0	0.0	12856	3
2022-06-30	3375	6412	5391	43604	105916	21822	6823	0.0	0.0	17638	4
2022-07-31	4433	6800	5828	51821	119939	26277	7758	0.0	0.0	21419	4
2022-08-31	5394	7145	6076	58007	128750	30960	8230	0.0	0.0	24476	4
2022-09-30	6135	7428	6442	65436	139969	38386	9369	0.0	0.0	27511	4
2022-10-31	7285	8324	7442	78491	161345	48678	11219	0.0	0.0	37836	4
2022-11-30	7965	8336	7811	85201	172197	55137	11779	0.0	0.0	45297	4
2022-12-31	1701	1683	1577	17523	35343	11452	2370	0.0	0.0	9440	

In [51]:
```
dataset1_monthly.shape
```

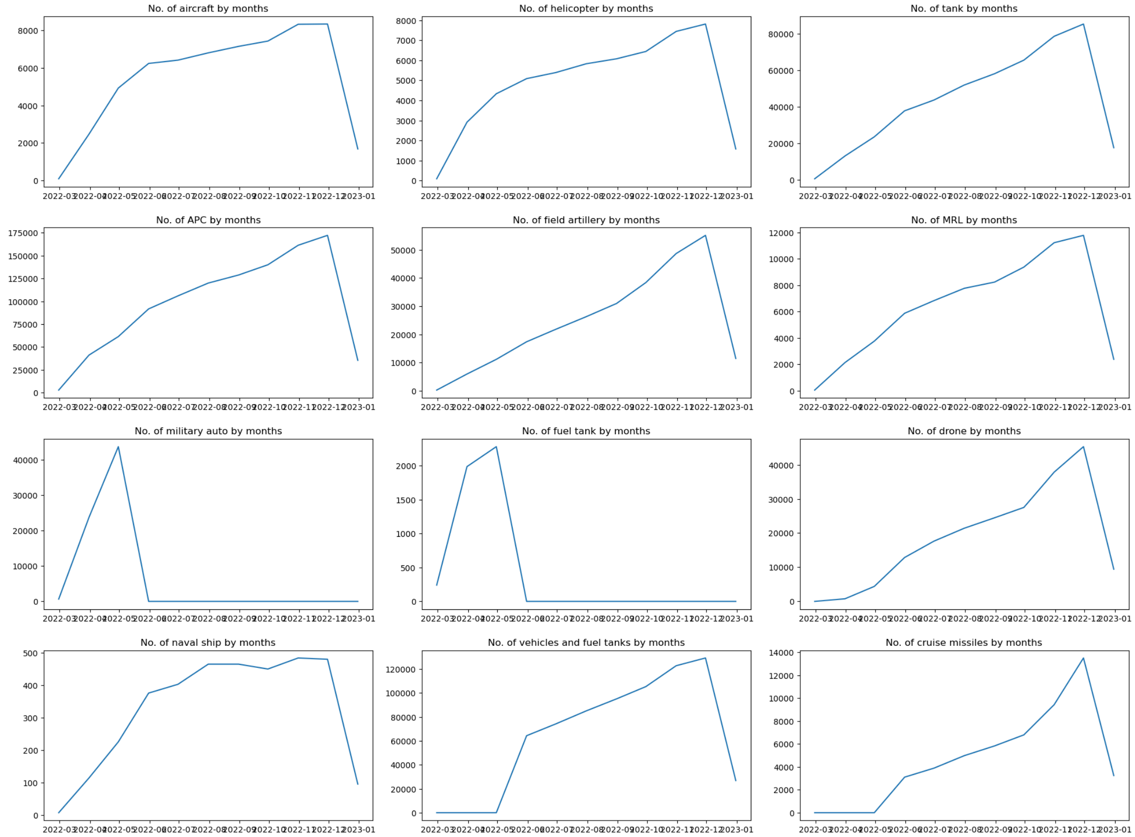
Out[51]:
```
(11, 16)
```

In [52]:
```
dataset1_monthly.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 11 entries, 2022-02-28 to 2022-12-31
Freq: M
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   day                     11 non-null     int64  
 1   aircraft                11 non-null     int64  
 2   helicopter              11 non-null     int64  
 3   tank                    11 non-null     int64  
 4   APC                     11 non-null     int64  
 5   field artillery         11 non-null     int64  
 6   MRL                     11 non-null     int64  
 7   military auto           11 non-null     float64
 8   fuel tank               11 non-null     float64
 9   drone                   11 non-null     int64  
 10  naval ship              11 non-null     int64  
 11  anti-aircraft warfare   11 non-null     int64  
 12  special equipment       11 non-null     float64
 13  mobile SRBM system      11 non-null     float64
 14  vehicles and fuel tanks 11 non-null     float64
 15  cruise missiles         11 non-null     float64
dtypes: float64(6), int64(10)
memory usage: 1.5 KB
```

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

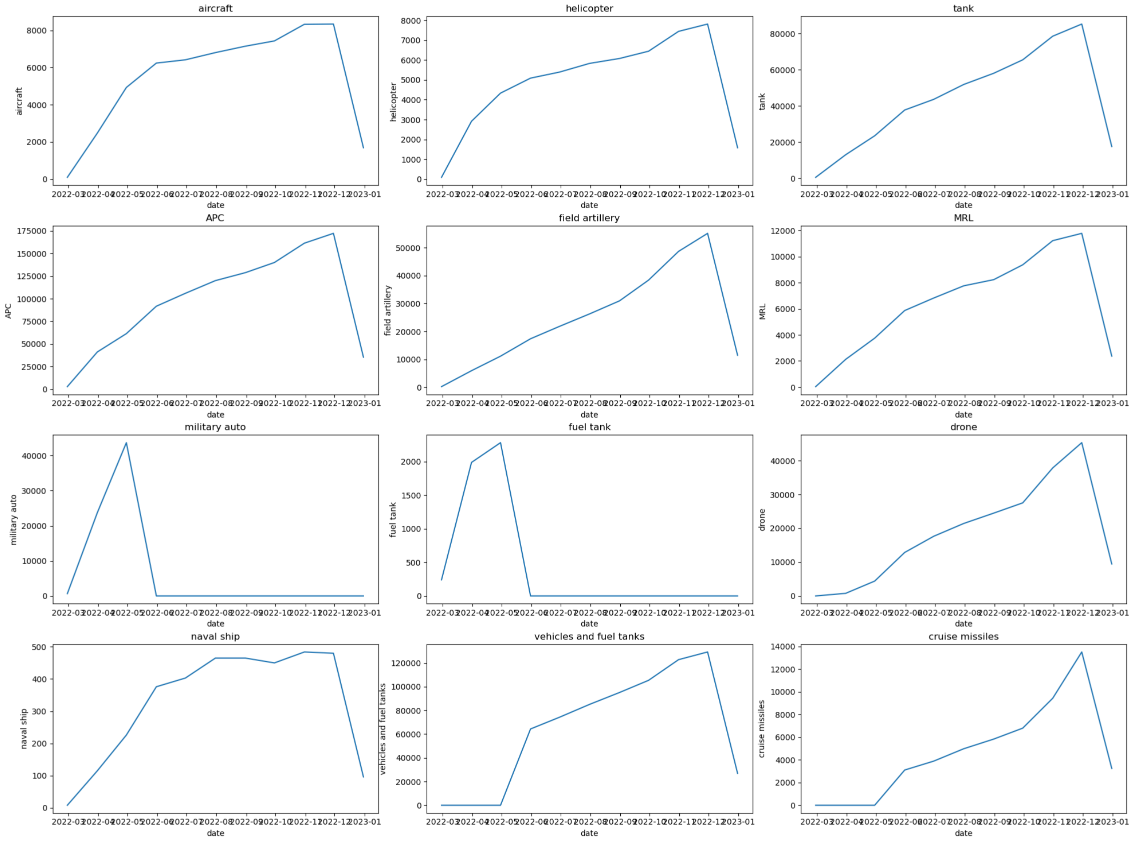
In [53]:
```
# Assuming you have a list of attribute names to plot
attribute_names = ["aircraft", "helicopter", "tank", "APC", "field artiller
                   "fuel tank", "drone", "naval ship", "vehicles and fuel t
# Create a 4x3 grid of subplots
fig, axes = plt.subplots(4, 3, figsize=(20, 15))
fig.tight_layout(pad=3.0)  # 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()
```



Air-vehicles such as aircraft and helicopters are the primary weapon of
assault for Russia in the
battle; yet, as we have seen 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 aircrafts and
helicopters with all the
possible parameters of sns.lineplot.

line plot 1 by 1

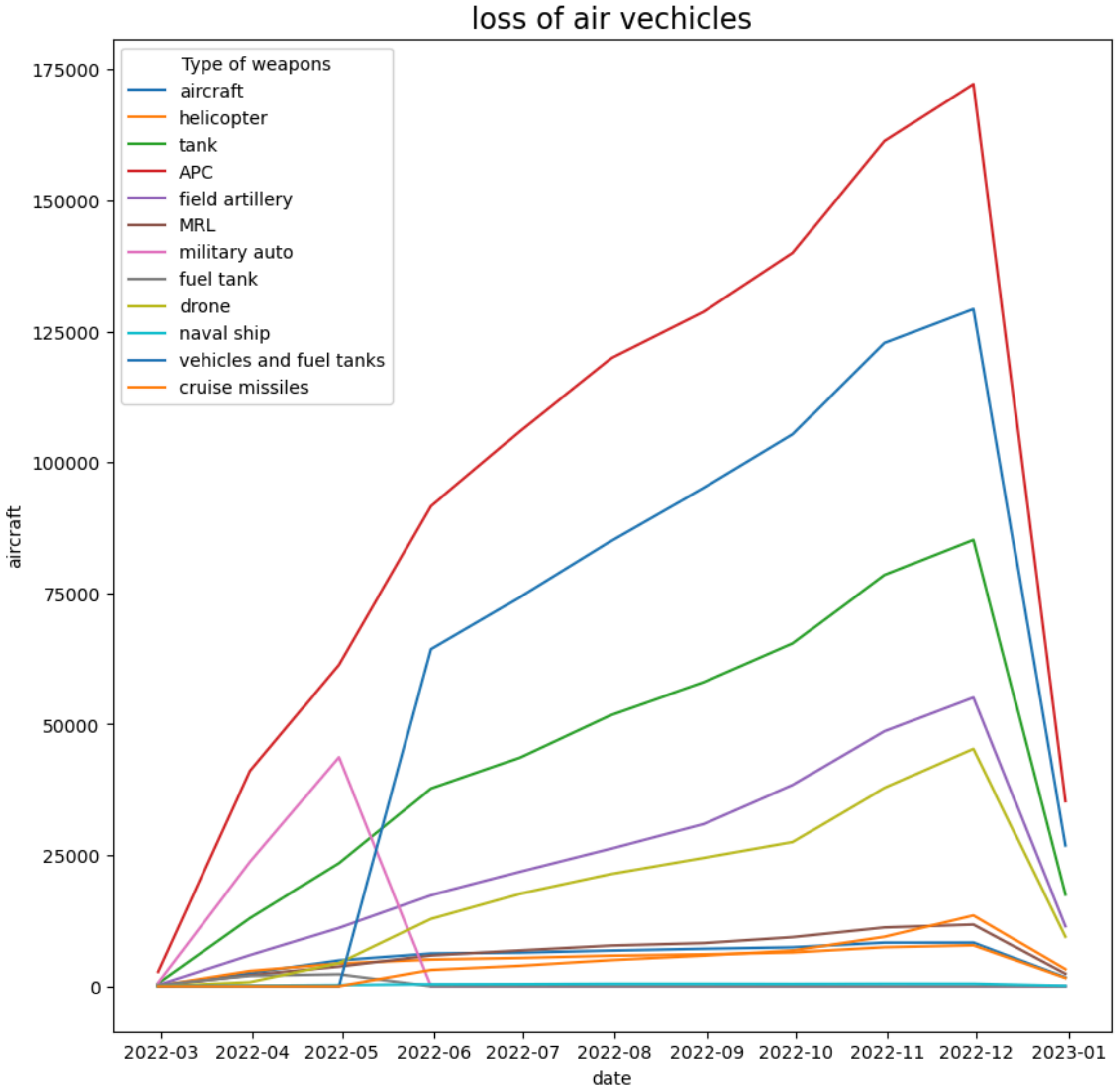
In [54]:
```
# Assuming you have a list of attribute names to plot
attribute_names = ["aircraft", "helicopter", "tank", "APC", "field artiller
                   "fuel tank", "drone", "naval ship", "vehicles and fuel t
# Create a 4x3 grid of subplots

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], c
    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

In [55]:

```
# Assuming you have a list of attribute names to plot
attribute_names = ["aircraft", "helicopter", "tank", "APC", "field artiller
                   "fuel tank", "drone", "naval ship", "vehicles and fuel t
# 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], c
plt.title("loss of air vechicles", fontsize=16)
plt.legend(title="Type of weapons",fontsize='medium')
plt.show()
```



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.

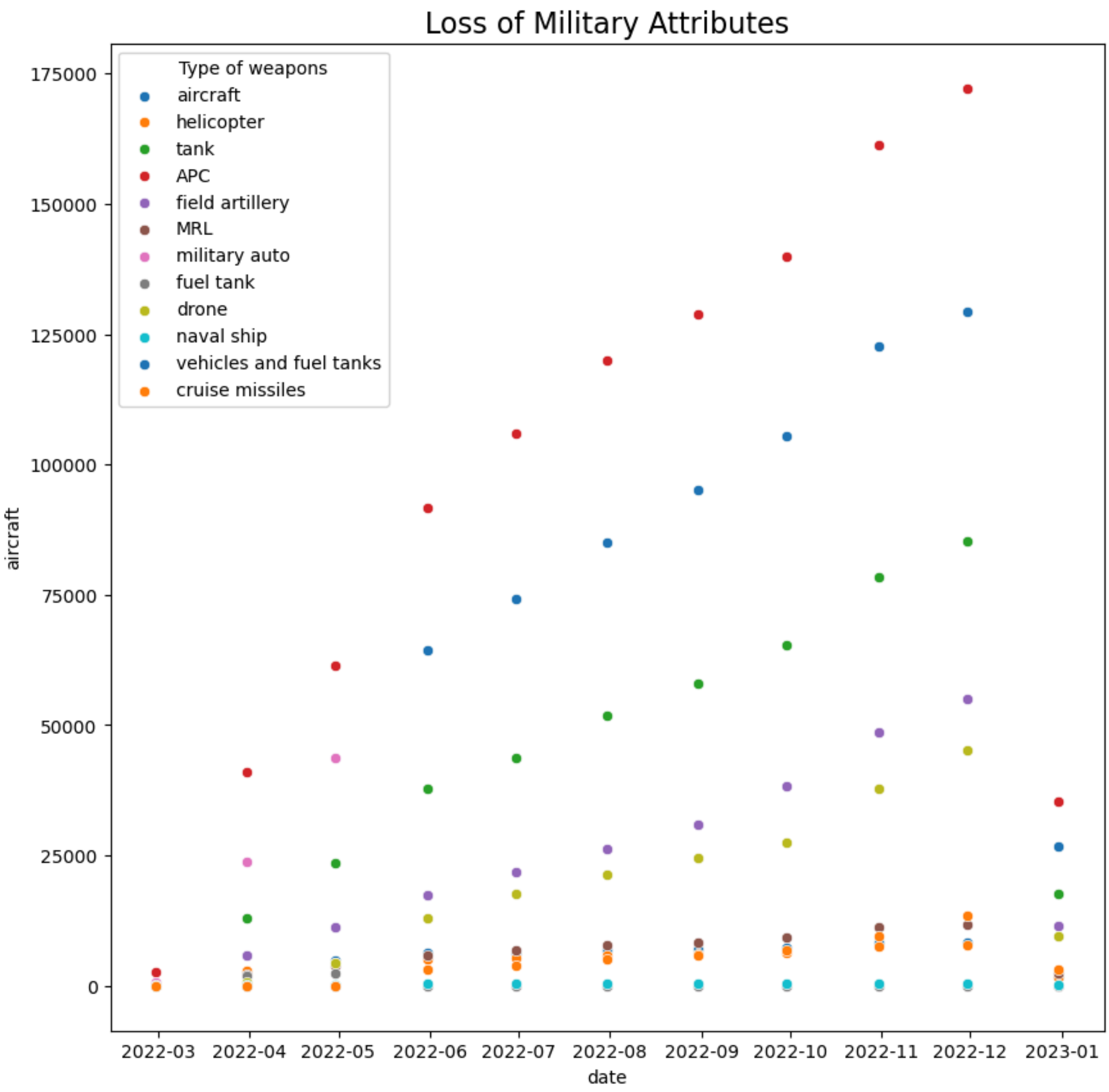
```
Data points tend to gather together in patterns called clustering, which
can be seen.
# Syntax:
plt.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None,
vmin=None, vmax=None, alpha=None, edgecolors=None, linewidths=None,
label=None)
Parameters:
x: A sequence of values representing the 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.
```

```
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]:

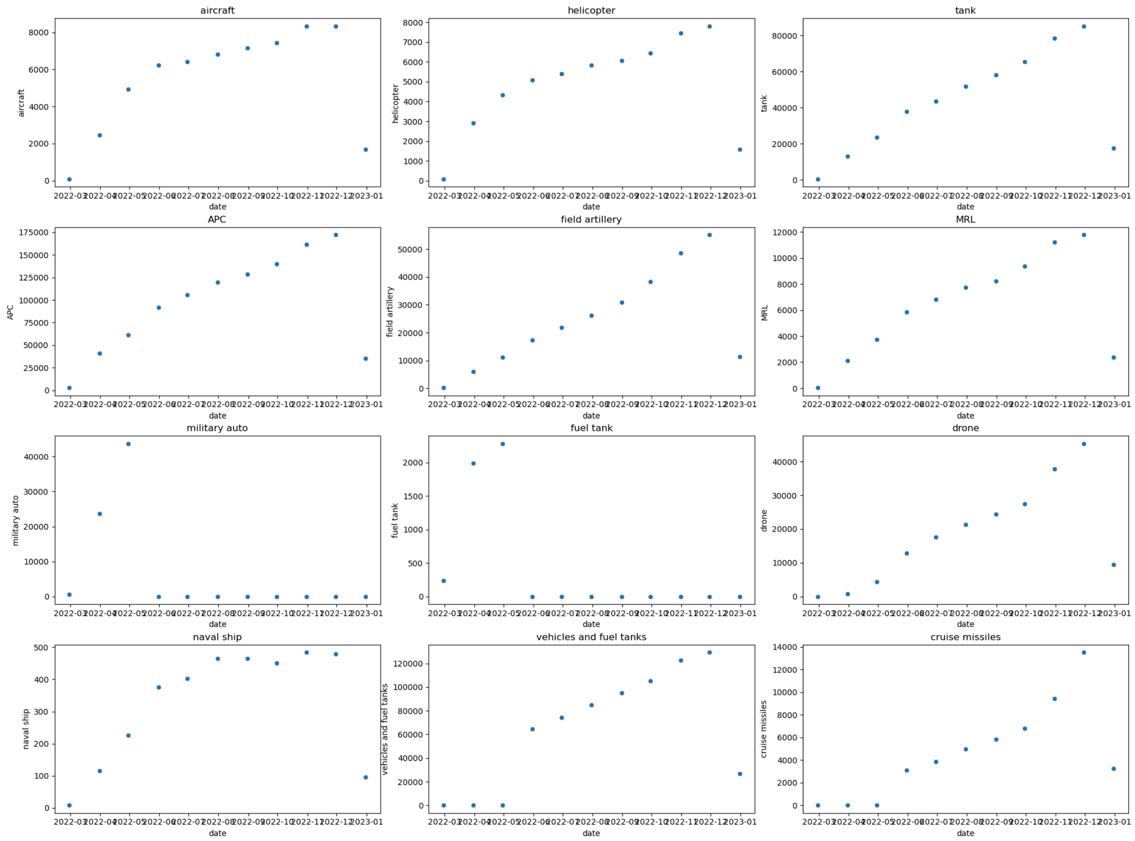
```
# Assuming you have a list of attribute names to plot
attribute_names = ["aircraft", "helicopter", "tank", "APC", "field artiller
                   "fuel tank", "drone", "naval ship", "vehicles and fuel t
# Create a figure with a specified size
plt.figure(figsize=(10, 10))
# Iterate through attribute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

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



scatter plot 1 by 1

In [57]:

```
# Assuming you have a list of attribute names to plot
attribute_names = ["aircraft", "helicopter", "tank", "APC", "field artiller
                   "fuel tank", "drone", "naval ship", "vehicles and fuel t
# 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]
    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()
```



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.

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

In [58]:

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

In [59]:

```
#merge the dataset1,dataset2
dataset_merge=pd.merge(dataset1,dataset2,on='day')
dataset_merge
```

Out[59]:

	date_x	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	...	anti-aircraft warfare
0	2022-02-25	2	10	7	80	516	49	4	100.0	60.0	...	0
1	2022-02-26	3	27	26	146	706	49	4	130.0	60.0	...	0
2	2022-02-27	4	27	26	150	706	50	4	130.0	60.0	...	0
3	2022-02-28	5	29	29	150	816	74	21	291.0	60.0	...	5
4	2022-03-01	6	29	29	198	846	77	24	305.0	60.0	...	7
...	...	...	...	...	...	...	...	...	...	...	...	...
280	2022-12-02	282	280	262	2916	5883	1905	395	NaN	NaN	...	210
281	2022-12-03	283	280	263	2917	5886	1906	395	NaN	NaN	...	210
282	2022-12-04	284	281	263	2922	5892	1908	395	NaN	NaN	...	210
283	2022-12-05	285	281	264	2924	5900	1914	395	NaN	NaN	...	211
284	2022-12-06	286	281	264	2929	5905	1915	395	NaN	NaN	...	211

285 rows × 22 columns

In [60]:

```
dataset_numeric=dataset_merge.select_dtypes(include=["int64","float64"])
```

In [61]:

```
dataset_numeric
```

Out[61]:

	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	naval ship	an aircra warfa
0	2	10	7	80	516	49	4	100.0	60.0	0	2	
1	3	27	26	146	706	49	4	130.0	60.0	2	2	
2	4	27	26	150	706	50	4	130.0	60.0	2	2	
3	5	29	29	150	816	74	21	291.0	60.0	3	2	
4	6	29	29	198	846	77	24	305.0	60.0	3	2	
...	...	...	...	...	...	...	...	...	...	...	...	
280	282	280	262	2916	5883	1905	395	NaN	NaN	1564	16	2
281	283	280	263	2917	5886	1906	395	NaN	NaN	1572	16	2
282	284	281	263	2922	5892	1908	395	NaN	NaN	1573	16	2
283	285	281	264	2924	5900	1914	395	NaN	NaN	1582	16	2
284	286	281	264	2929	5905	1915	395	NaN	NaN	1587	16	2

285 rows × 18 columns

```
Find 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 [62]:
```
dataset_numeric.corr()
```

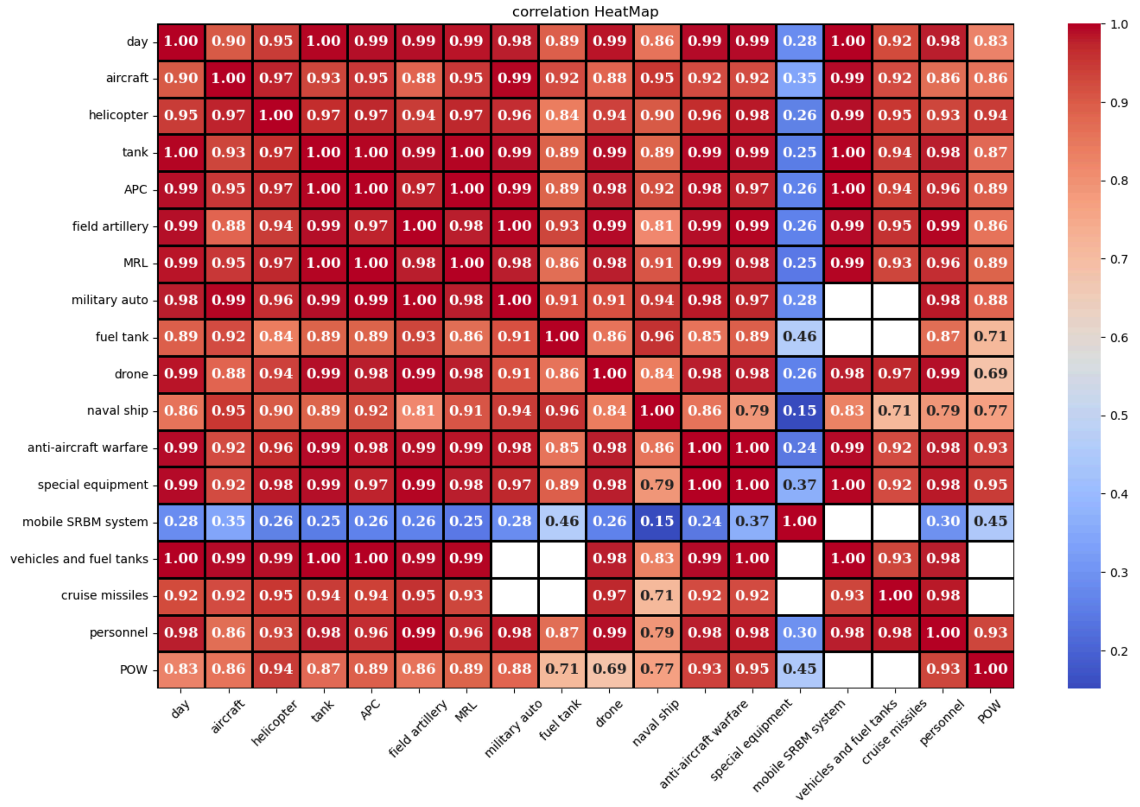
Out[62]:

	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto
day	1.000000	0.902627	0.945223	0.995043	0.987652	0.990050	0.985816	0.979363
aircraft	0.902627	1.000000	0.974623	0.931273	0.948758	0.881600	0.948378	0.993953
helicopter	0.945223	0.974623	1.000000	0.965292	0.971078	0.941753	0.973233	0.962985
tank	0.995043	0.931273	0.965292	1.000000	0.996569	0.988838	0.996170	0.988725
APC	0.987652	0.948758	0.971078	0.996569	1.000000	0.974949	0.997560	0.993803
field artillery	0.990050	0.881600	0.941753	0.988838	0.974949	1.000000	0.977971	0.996674
MRL	0.985816	0.948378	0.973233	0.996170	0.997560	0.977971	1.000000	0.979341
military auto	0.979363	0.993953	0.962985	0.988725	0.993803	0.996674	0.979341	1.000000
fuel tank	0.890240	0.921081	0.841115	0.887154	0.886196	0.929310	0.864099	0.912906
drone	0.985304	0.883633	0.936588	0.988083	0.979055	0.992124	0.979980	0.913692
naval ship	0.860192	0.952457	0.903286	0.886850	0.917941	0.814901	0.910359	0.941556
anti-aircraft warfare	0.991955	0.920741	0.963746	0.993345	0.984740	0.989687	0.986309	0.981397
special equipment	0.992070	0.921511	0.982749	0.987678	0.974112	0.994727	0.976808	0.971746
mobile SRBM system	0.284747	0.347026	0.258891	0.247350	0.263685	0.262446	0.245700	0.283902
vehicles and fuel tanks	0.996736	0.992912	0.989197	0.998231	0.997478	0.992978	0.993832	NaN
cruise missiles	0.923865	0.921398	0.951585	0.942891	0.937318	0.954875	0.929510	NaN
personnel	0.979639	0.864087	0.933072	0.978383	0.962325	0.994139	0.963493	0.982241
POW	0.826973	0.860039	0.941204	0.871803	0.887960	0.855041	0.887171	0.881444

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

In [63]:

```
plt.figure(figsize=(16,10))
sns.heatmap(dataset_numeric.corr(),cmap="coolwarm",annot=True,fmt="0.2f",ar
plt.xticks(rotation=45)
plt.title("correlation HeatMap")
plt.show()
```



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.

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

In [64]:

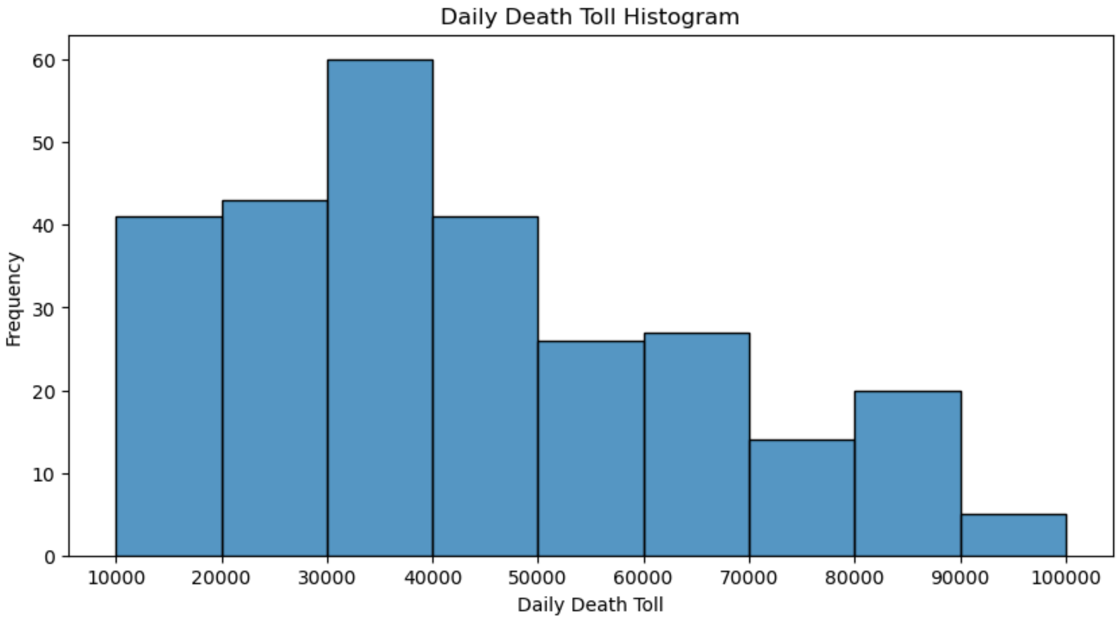
```
dataset2.head()
```

Out[64]:

	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

In [65]:

```
plt.figure(figsize=(10,5))
sns.histplot(dataset2.personnel,bins=[10000,20000,30000,40000,50000,60000,7
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()
```



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).
```

```
Find the loss of tanks and APC through the Bar plot.
```

In [66]:

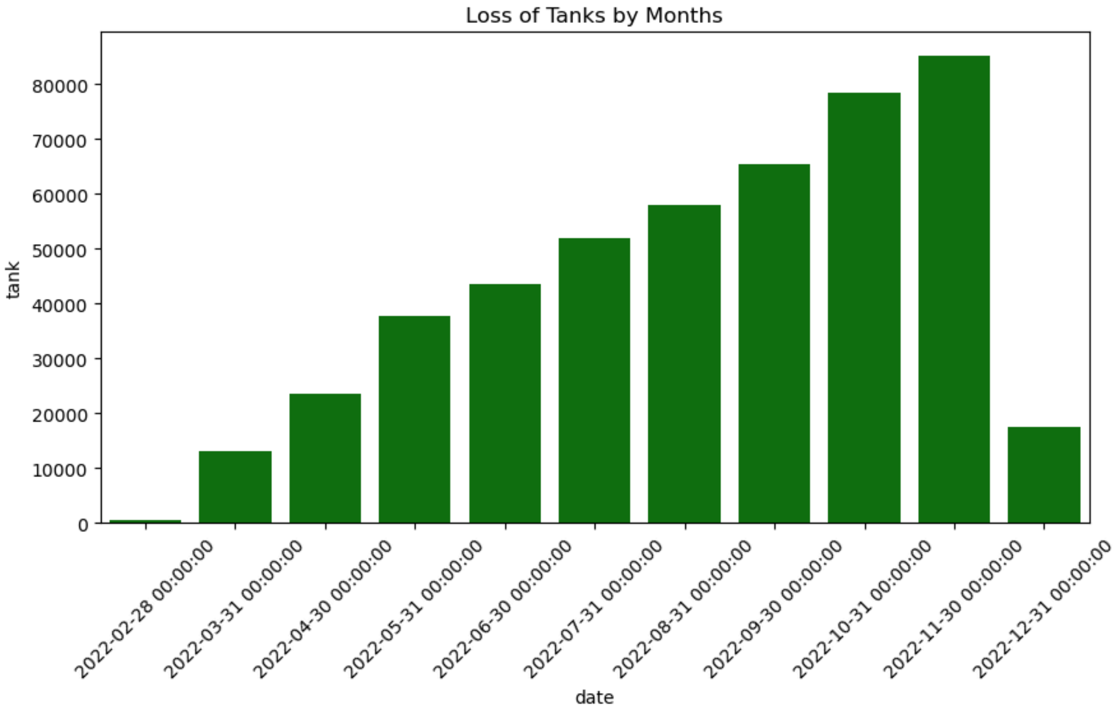
```
dataset1_monthly.head()
```

Out[66]:

	day	aircraft	helicopter	tank	APC	field artillery	MRL	military auto	fuel tank	drone	nava shi
date											
2022-02-28	14	93	88	526	2744	222	33	651.0	240.0	7	
2022-03-31	651	2472	2913	12998	41083	5928	2123	23740.0	1986.0	774	11
2022-04-30	1545	4926	4331	23478	61343	11136	3749	43697.0	2280.0	4397	22
2022-05-31	2542	6237	5086	37713	91653	17371	5864	0.0	0.0	12856	37
2022-06-30	3375	6412	5391	43604	105916	21822	6823	0.0	0.0	17638	40

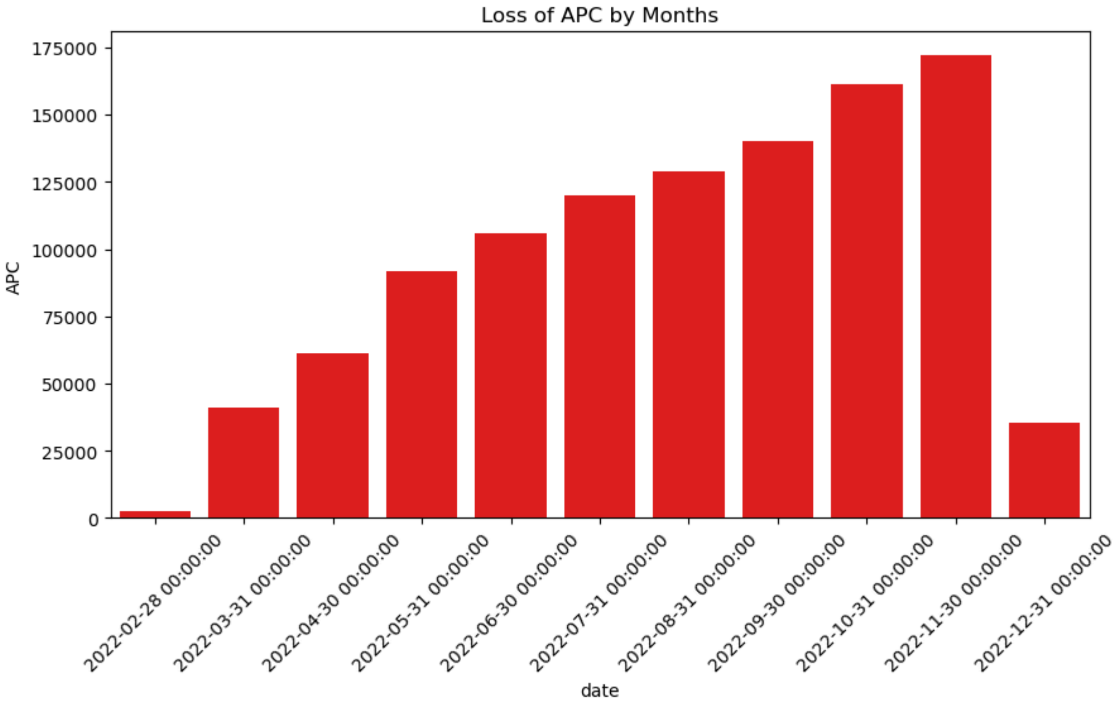
In [67]:

```
#loss of tanks
plt.figure(figsize=(10,5))
sns.barplot(x=dataset1_monthly.index,y=dataset1_monthly.tank,data=dataset1_
plt.xticks(rotation=45)
plt.title("Loss of Tanks by Months")
plt.show()
```



In [68]:

```
#loss of APC
plt.figure(figsize=(10,5))
sns.barplot(x=dataset1_monthly.index,y=dataset1_monthly.APC,data=dataset1_m
plt.xticks(rotation=45)
plt.title("Loss of APC by Months")
plt.show()
```



In [ ]: