# Ukraine Russia War Data Analysis

## About Dataset

The 2022 Ukraine–Russia War dataset tracks daily cumulative Russian military losses, including personnel and equipment like tanks, aircraft, and drones. Verified through official sources and visual evidence, it enables time-series analysis of conflict intensity and patterns.

In [3]:
```
# IMPORT LIBRARIES
import pandas as pd
```

In [5]:
```
# LOAD THE DATASET
data = pd.read_csv('russia_losses_equipment.csv')
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1214 entries, 0 to 1213
Data columns (total 19 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   date                       1214 non-null   object 
 1   day                        1214 non-null   int64  
 2   aircraft                   1214 non-null   int64  
 3   helicopter                 1214 non-null   int64  
 4   tank                       1214 non-null   int64  
 5   APC                        1214 non-null   int64  
 6   field artillery            1214 non-null   int64  
 7   MRL                        1214 non-null   int64  
 8   military auto              65 non-null     float64
 9   fuel tank                  65 non-null     float64
 10  drone                      1214 non-null   int64  
 11  naval ship                 1214 non-null   int64  
 12  anti-aircraft warfare      1214 non-null   int64  
 13  special equipment          1195 non-null   float64
 14  mobile SRBM system         36 non-null     float64
 15  greatest losses direction  203 non-null    object 
 16  vehicles and fuel tanks    1149 non-null   float64
 17  cruise missiles            1149 non-null   float64
 18  submarines                 648 non-null    float64
dtypes: float64(7), int64(10), object(2)
memory usage: 180.3+ KB
```

In [9]:
```
data.shape
```

Out[9]:
```
(1214, 19)
```

In [11]:
```
data.describe()
```

Out[11]:

| | day | aircraft | helicopter | tank | APC | field artillery |
|---|---|---|---|---|---|---|
| **count** | 1214.000000 | 1214.000000 | 1214.000000 | 1214.000000 | 1214.000000 | 1214.00000C |
| **mean** | 608.500000 | 309.400329 | 286.542010 | 5595.590610 | 11194.628501 | 9747.99093S |
| **std** | 350.595921 | 68.163105 | 63.348728 | 3168.560315 | 6400.082849 | 8810.954283 |
| **min** | 2.000000 | 10.000000 | 7.000000 | 80.000000 | 516.000000 | 49.00000C |
| **25%** | 305.250000 | 283.000000 | 267.000000 | 3012.250000 | 6011.750000 | 1992.25000C |
| **50%** | 608.500000 | 320.000000 | 324.000000 | 5108.500000 | 9683.000000 | 7103.00000C |
| **75%** | 911.750000 | 367.000000 | 328.000000 | 8532.000000 | 16591.000000 | 17295.75000C |
| **max** | 1215.000000 | 416.000000 | 337.000000 | 10964.000000 | 22867.000000 | 29432.00000C |

## Overall Equipment Loss Summary

### Convert 'date' column to datetime

In [15]:
```
data['date'] = pd.to_datetime(data['date'])
```

### Get final values (latest date) for each equipment type

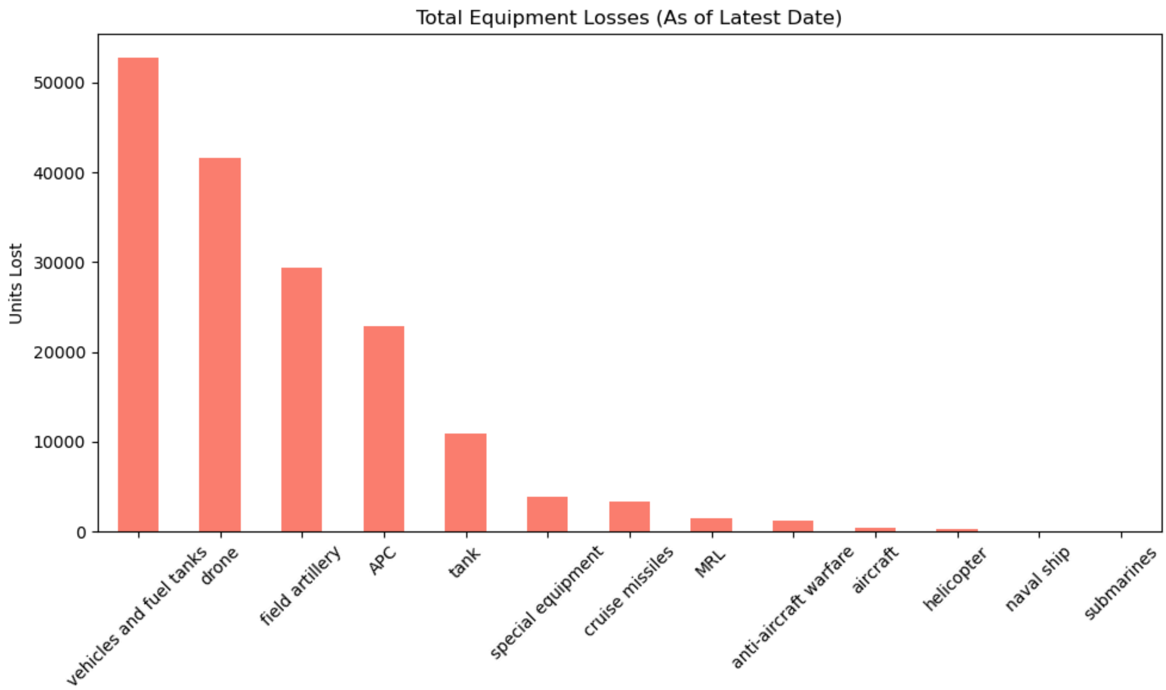
- The dataset might contain many columns (date, region, notes, etc.), but we are only interested in military equipment loss counts.
- This list helps us focus on the key features (weapons/equipment types) that are relevant to this analysis.
- It allows us to filter out only these columns from the row.

In [22]:
```
latest_row = data.iloc[0]
equipment_columns = ['aircraft', 'helicopter', 'tank', 'APC', 'field artillery',
                     'anti-aircraft warfare', 'special equipment', 'vehicles and
```

In [26]:
```
overall_losses = latest_row[equipment_columns].dropna().sort_values(ascending=Fa
overall_losses
```

Out[26]:
```
vehicles and fuel tanks    52734.0
drone                        41579
field artillery              29432
APC                          22867
tank                         10964
special equipment           3920.0
cruise missiles             3376.0
MRL                           1421
anti-aircraft warfare         1188
aircraft                       416
helicopter                     337
naval ship                      28
submarines                     1.0
Name: 0, dtype: object
```

In [30]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
overall_losses.plot(kind='bar', color='salmon')
plt.title('Total Equipment Losses (As of Latest Date)')
plt.ylabel('Units Lost')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

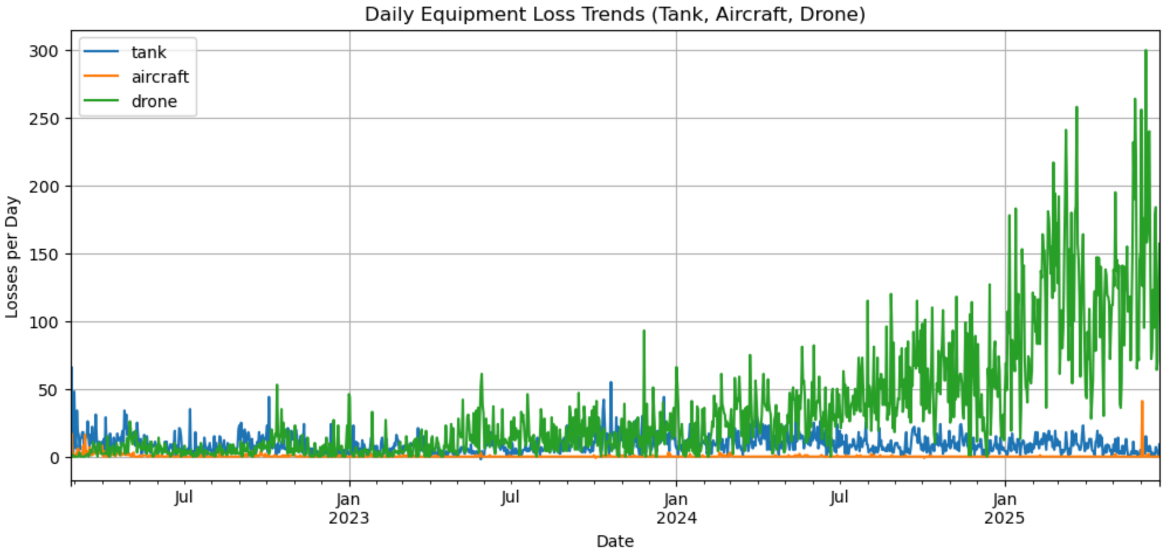



- This visualization provides a clear comparison of how many units of each major military equipment type have been lost.
- It shows vehicles and fuel tanks (52734) have suffered the most losses and highlights the scale of impact in modern warfare.
- Such visual analytics are crucial in defense analysis, strategic planning, and resource allocation.

## Daily Trend of Equipment Losses

In [38]:
```
# Calculate daily loss differences (day-to-day)
daily_losses = data.sort_values('date')[['date'] + equipment_columns].set_index(
```

In [40]:
```
# Plot trend for selected equipment
daily_losses[['tank', 'aircraft', 'drone']].plot(figsize=(12, 5))
plt.title('Daily Equipment Loss Trends (Tank, Aircraft, Drone)')
plt.ylabel('Losses per Day')
plt.xlabel('Date')
plt.grid(True)
plt.show()
```

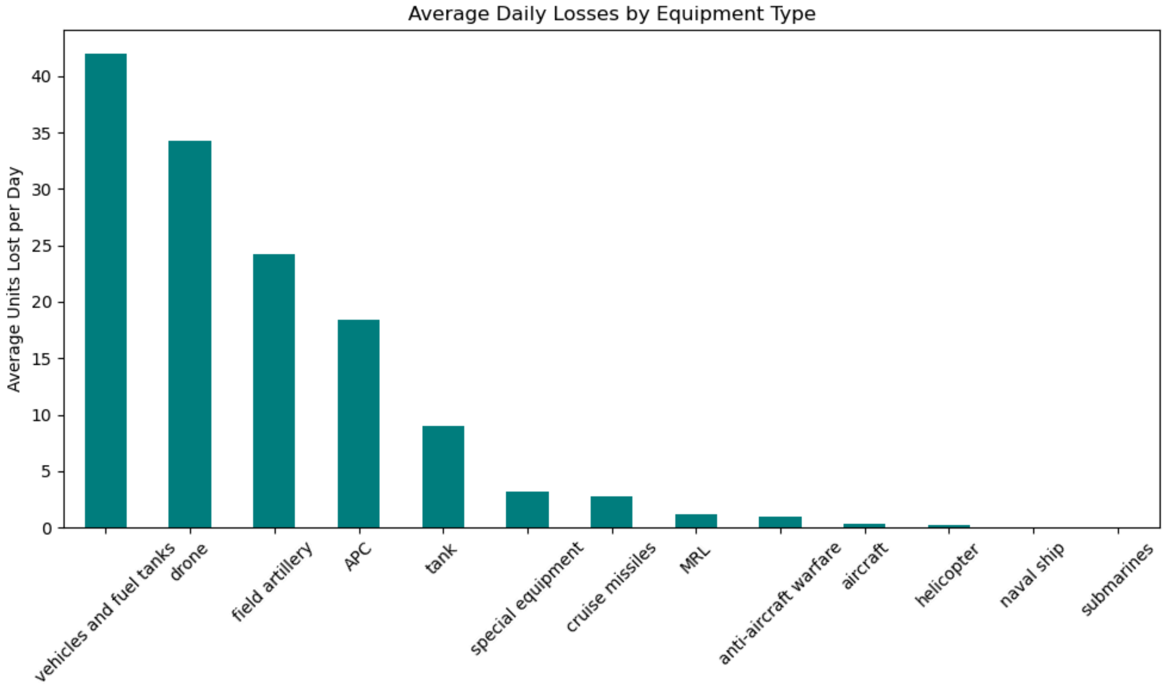



- Drone losses have sharply increased over time, especially from mid-2024 onward
- Tank losses remain steady but moderate, with occasional spikes reflecting ground battle surges.
- Aircraft losses stay consistently low, suggesting limited use or high preservation of manned air assets compared to drones.
- The year 2025 shows a dramatic rise in drone losses, likely pointing to a strategic shift or escalation in drone-based warfare.
- Drones show high volatility, unlike tanks and aircraft, highlighting their rapid deployment and destruction cycle in modern warfare.

## Most Frequently Lost Equipment Types (Daily)

In [43]:
```
# Sum of all daily losses
avg_daily_loss = daily_losses.mean().sort_values(ascending=False)
```

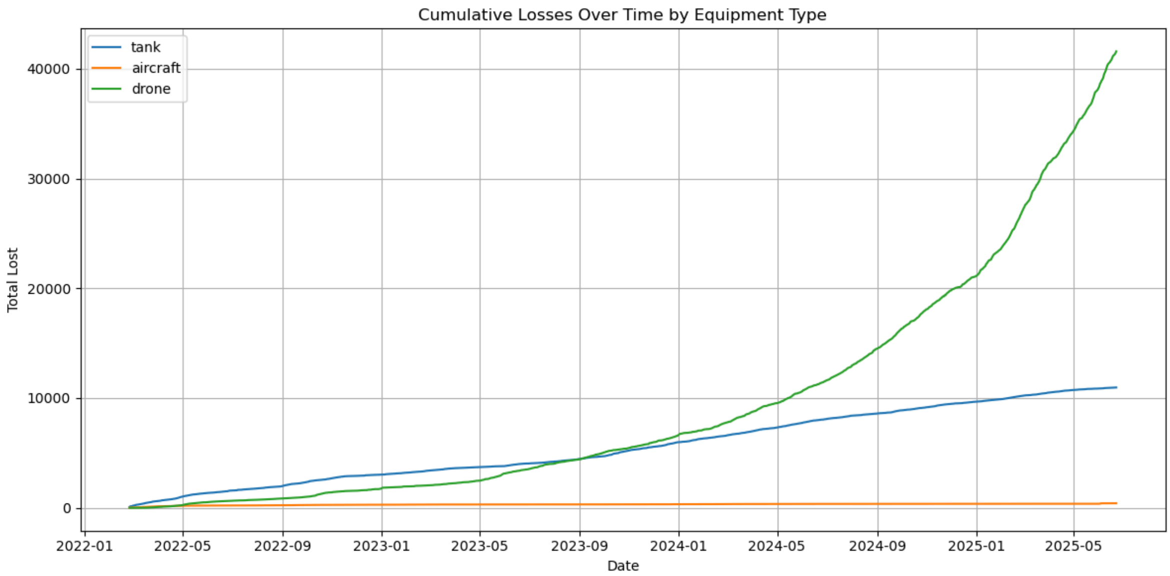
In [45]:
```
plt.figure(figsize=(10, 6))
avg_daily_loss.plot(kind='bar', color='teal')
plt.title('Average Daily Losses by Equipment Type')
plt.ylabel('Average Units Lost per Day')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




- Vehicles and fuel tanks have the highest average daily losses (~42 units/day), indicating frequent ground mobility destruction.
- Drones follow closely, emphasizing their widespread use and vulnerability in modern warfare.
- Field artillery, APCs, and tanks also suffer notable daily losses, showing active ground warfare.
- Aircraft, helicopters, naval ships, and submarines show minimal daily losses, reflecting less frequent or more protected deployment.
- The data highlights a shift toward high drone and vehicle attrition over heavier equipment.

## Compare loss types over time

In [47]:
```
# Cumulative plots of multiple equipment types
data_sorted = data.sort_values('date')
plt.figure(figsize=(12, 6))
for col in ['tank', 'aircraft', 'drone']:
    plt.plot(data_sorted['date'], data_sorted[col], label=col)

plt.title("Cumulative Losses Over Time by Equipment Type")
plt.xlabel("Date")
plt.ylabel("Total Lost")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```




- Drone losses (green line) have surged exponentially since mid-2023, crossing 40,000 units by mid-2025.
- Tank losses show a steady linear increase, now exceeding 10,000 units.
- Aircraft losses remain very low, suggesting either strategic conservation or fewer air battles.

## Correlation Between Losses and Time

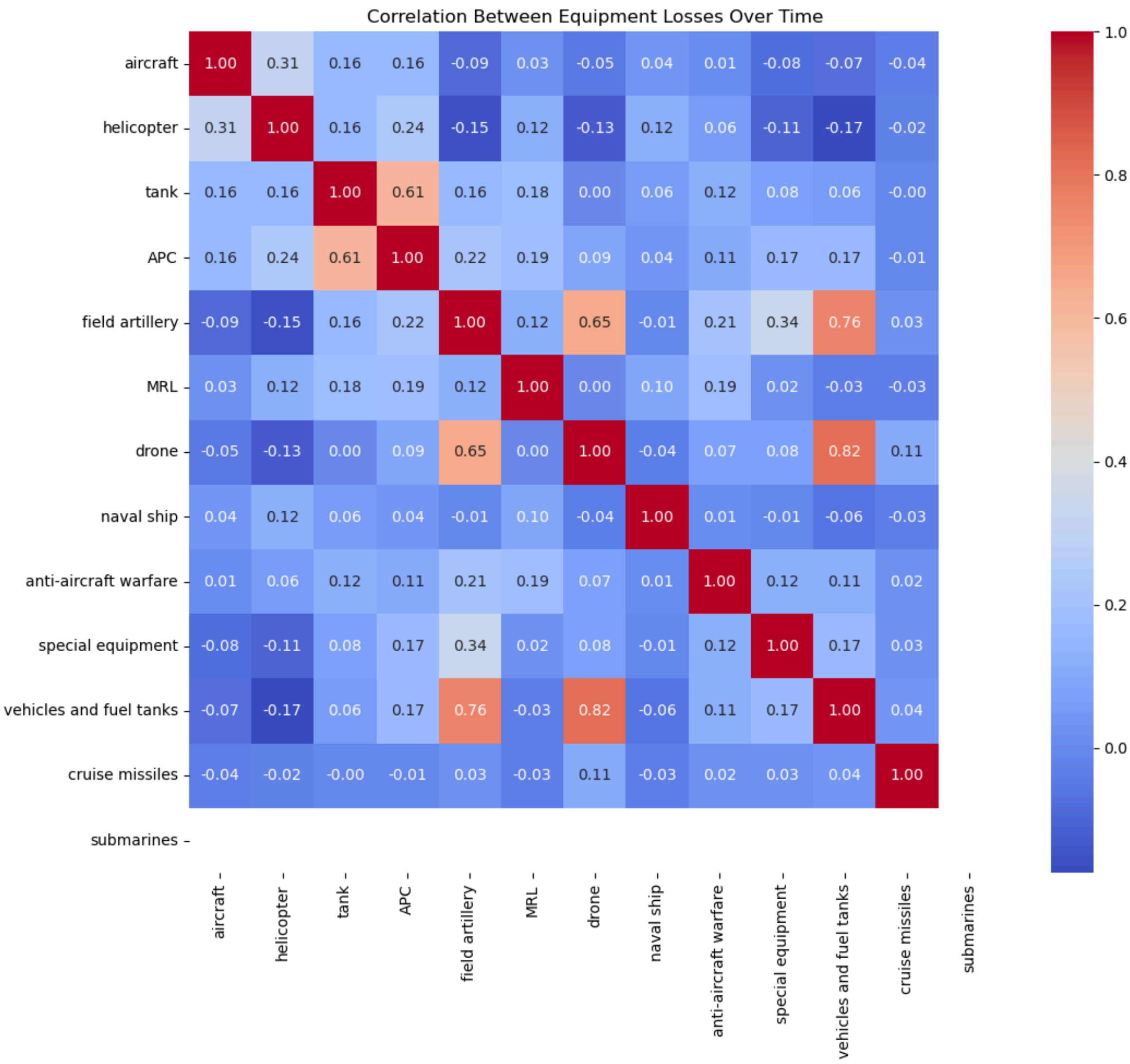
In [50]:
```
import seaborn as sns
import numpy as np
```

In [56]:
```
# Prepare cleaned daily_losses again
daily_losses = data.sort_values('date')[['date'] + equipment_columns].set_index(
```

In [58]:
```
# Ensure no negative values due to data corrections
daily_losses = daily_losses.clip(lower=0)
```

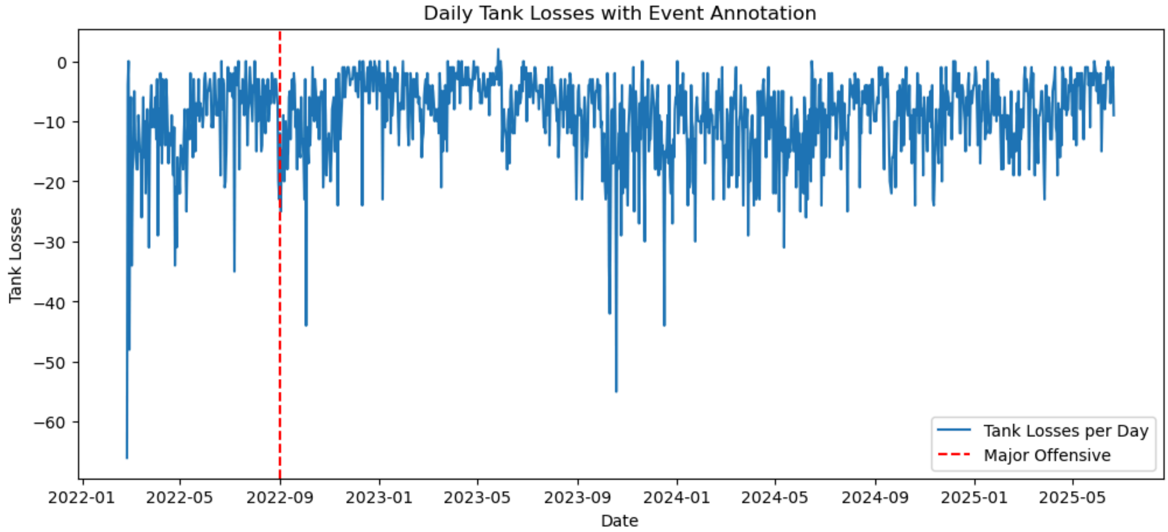
In [60]:
```
# Correlation matrix for daily losses
correlation_matrix = daily_losses.corr()
```

In [62]:
```
# Plot heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Between Equipment Losses Over Time")
plt.show()
```




## Impact of Specific Events

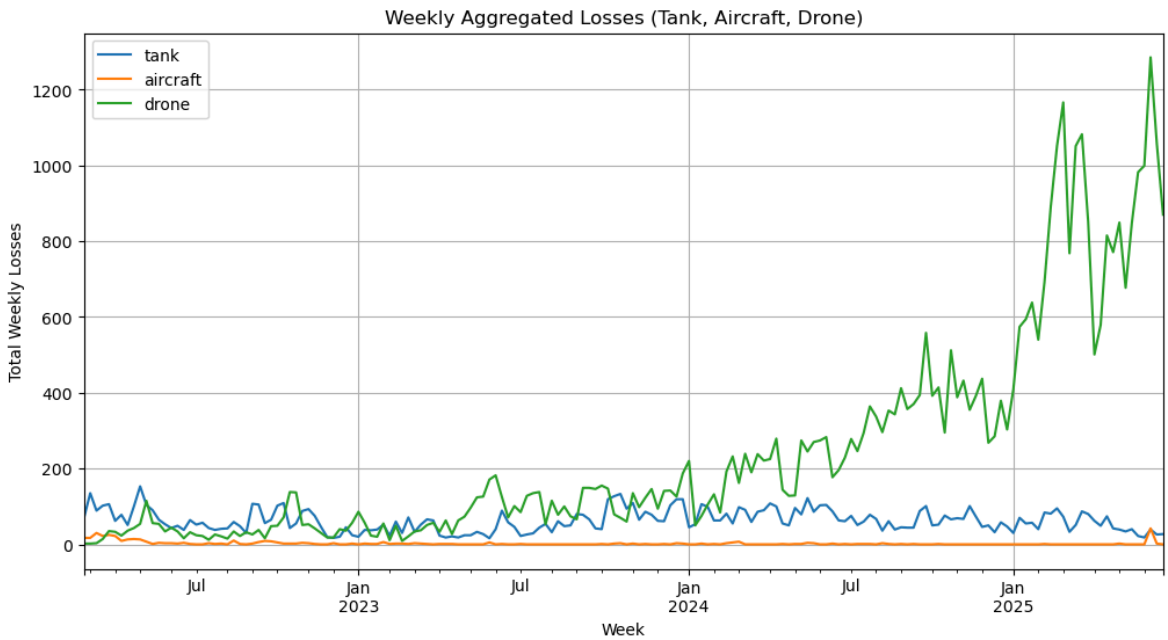
In [67]:
```python
# Example: Annotate spikes around a known date (replace with actual events)
highlight_date = '2022-09-01'  # Sample major offensive date
plt.figure(figsize=(12, 5))
plt.plot(data['date'], data['tank'].diff(), label='Tank Losses per Day')
plt.axvline(pd.to_datetime(highlight_date), color='red', linestyle='--', label='
plt.title("Daily Tank Losses with Event Annotation")
plt.xlabel("Date")
plt.ylabel("Tank Losses")
plt.legend()
plt.show()
```




- The red dashed line (Sept 1, 2022) marks a major offensive, after which tank losses appear more volatile.
- Several extreme dips up to indicate intense combat days post-offensive.
- Pre-offensive tank losses were relatively stable but post-offensive losses show greater variability.
- This suggests the event triggered a strategic shift or escalation in ground combat.

## Weekly or Monthly Loss Aggregation

In [70]:
```
# Resample by week
weekly_losses = daily_losses.resample('W').sum()

# Plot
weekly_losses[['tank', 'aircraft', 'drone']].plot(figsize=(12, 6))
plt.title("Weekly Aggregated Losses (Tank, Aircraft, Drone)")
plt.ylabel("Total Weekly Losses")
plt.xlabel("Week")
plt.grid(True)
plt.show()
```



- Drone losses have escalated rapidly, especially from mid-2024, surpassing 1,200 weekly losses in 2025.
- Tank losses remained moderate but consistent, with small fluctuations across weeks.
- Aircraft losses were minimal and steady, indicating either limited deployment or successful defense mechanisms.
- The gap between drone and other equipment losses widened, highlighting a shift in warfare strategy.

## Trend Before and After Key Dates

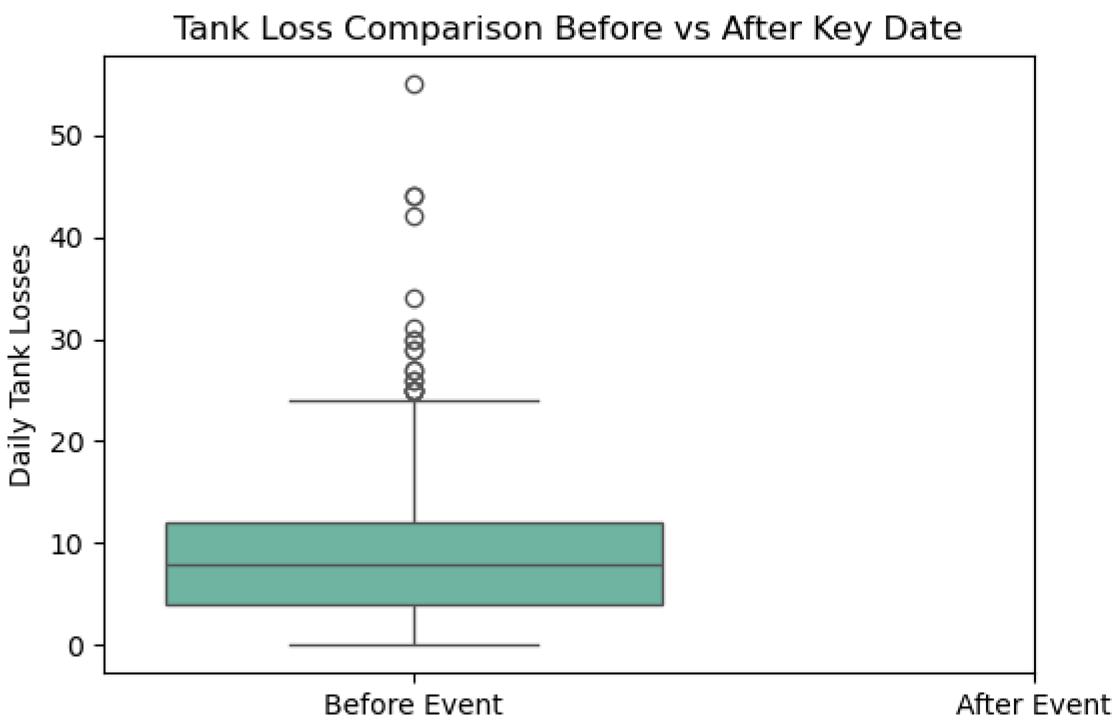
In [73]:
```
from scipy.stats import ttest_ind
```

In [75]:
```
# Choose a key event date
cutoff = pd.to_datetime('2022-09-01')
before = daily_losses.loc[daily_losses.index < cutoff]['tank']
after = daily_losses.loc[daily_losses.index >= cutoff]['tank']
```

In [77]:
```
# T-test
t_stat, p_value = ttest_ind(before.dropna(), after.dropna())
print(f"T-test result: t-stat={t_stat:.3f}, p-value={p_value:.4f}")
```

```
T-test result: t-stat=2.417, p-value=0.0158
```

In [79]:
```
# Boxplot
plt.figure(figsize=(6, 4))
sns.boxplot(data=[before, after], palette='Set2')
plt.xticks([0, 1], ['Before Event', 'After Event'])
plt.title("Tank Loss Comparison Before vs After Key Date")
plt.ylabel("Daily Tank Losses")
plt.show()
```



- Tank losses increased after the key date, as seen by the higher median and wider spread.
- Pre-event tank losses were lower and more stable, suggesting a build-up phase before a major conflict.
- Outliers before the event show some early spikes, but post-event losses were consistently higher, indicating intensified ground operations.

## Equipment Recovery or Capture Rates

In [82]:
```
# Placeholder logic if 'captured' or similar columns existed
if 'captured_tank' in data.columns:
    data['capture_ratio'] = data['captured_tank'] / data['tank']
    data[['date', 'capture_ratio']].plot(x='date', figsize=(10, 5), title="Captu
```

## Missing Data Analysis

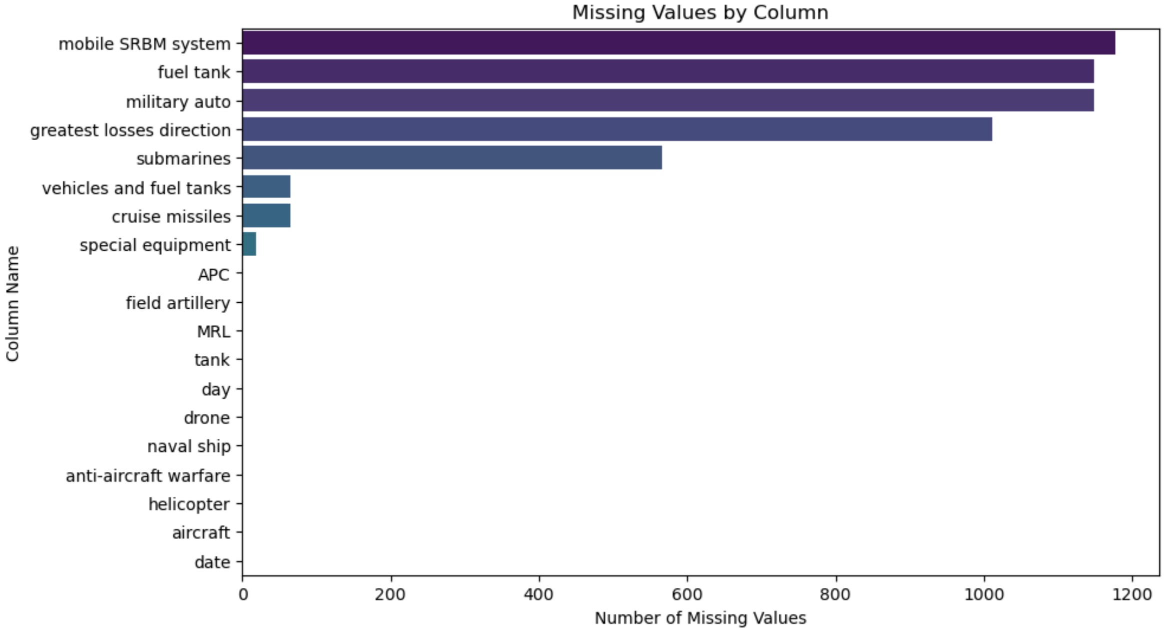
In [85]:
```
# Check for missing values in each column
missing_data = data.isnull().sum().sort_values(ascending=False)
print("Missing Data Summary:\n", missing_data)
```

```
Missing Data Summary:
 mobile SRBM system          1178
fuel tank                   1149
military auto               1149
greatest losses direction   1011
submarines                   566
vehicles and fuel tanks       65
cruise missiles               65
special equipment             19
APC                            0
field artillery                0
MRL                            0
tank                           0
day                            0
drone                          0
naval ship                     0
anti-aircraft warfare          0
helicopter                     0
aircraft                       0
date                           0
dtype: int64
```

In [87]:
```
# Visualize missing values
import seaborn as sns
import matplotlib.pyplot as plt
```

In [89]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x=missing_data.values, y=missing_data.index, palette="viridis")
plt.title("Missing Values by Column")
plt.xlabel("Number of Missing Values")
plt.ylabel("Column Name")
plt.show()
```

```
C:\Users\Nishita Bala\AppData\Local\Temp\ipykernel_55768\3103161547.py:2: FutureW
arning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=missing_data.values, y=missing_data.index, palette="viridis")
```

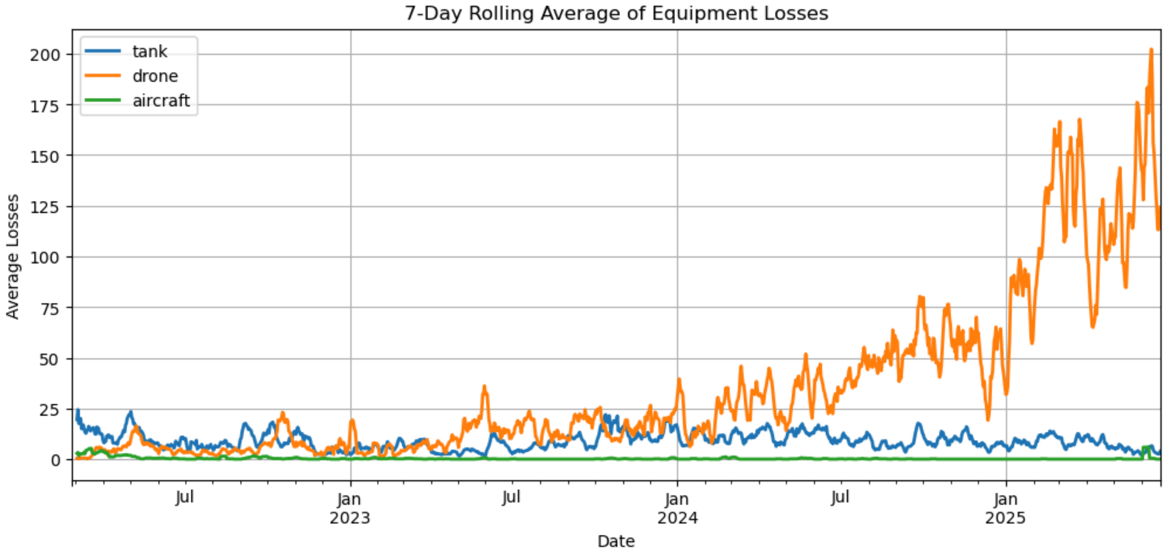



- Submarines and greatest losses direction are also sparsely recorded, limiting their analytical use.

## Rolling Average Analysis

In [92]:
```
# 7-day rolling average for selected equipment
rolling_avg = daily_losses[['tank', 'drone', 'aircraft']].rolling(window=7).mean
```

In [94]:
```
# Plot rolling averages
rolling_avg.plot(figsize=(12, 5), linewidth=2)
plt.title("7-Day Rolling Average of Equipment Losses")
plt.xlabel("Date")
plt.ylabel("Average Losses")
plt.grid(True)
plt.show()
```

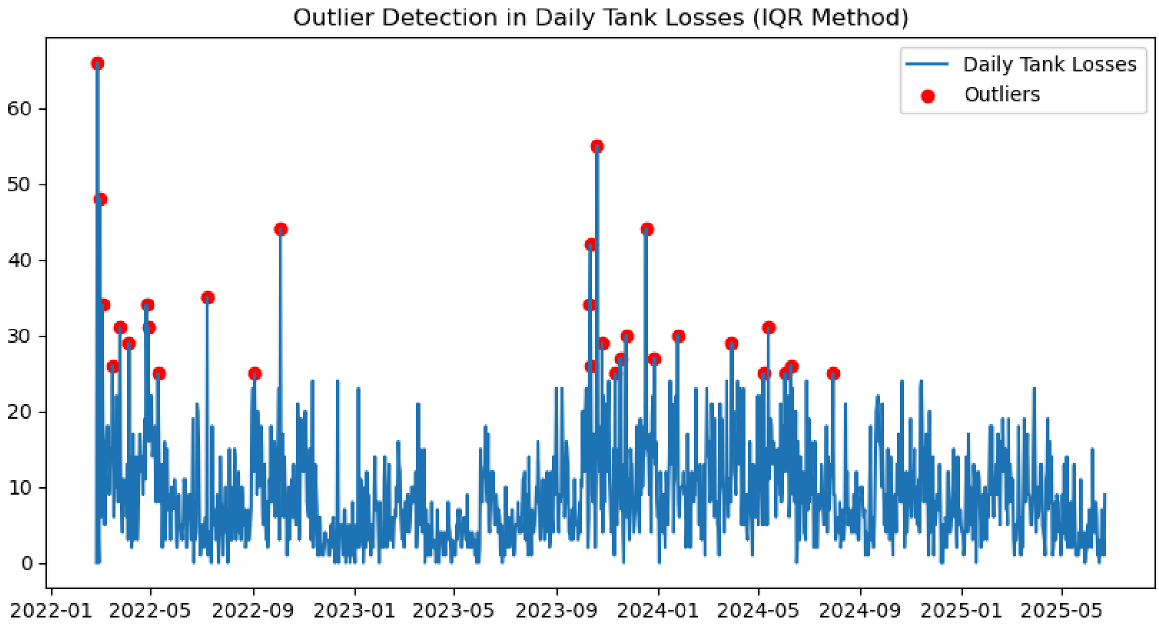



- Drone losses show a steep upward trend, peaking around early 2025 with volatile daily averages over 200.
- Tank losses follow a flatter trend, with periodic surges but no long-term acceleration.
- Aircraft losses are consistently low, indicating drones have largely replaced manned aircraft in frontline losses.

## Anomalies or Outlier Detection

In [97]:
```
# Detect outliers in daily tank losses using IQR method
Q1 = daily_losses['tank'].quantile(0.25)
Q3 = daily_losses['tank'].quantile(0.75)
IQR = Q3 - Q1
outliers = daily_losses[(daily_losses['tank'] < Q1 - 1.5 * IQR) | (daily_losses[
```

In [99]:
```
# Plot
plt.figure(figsize=(10, 5))
plt.plot(daily_losses.index, daily_losses['tank'], label='Daily Tank Losses')
plt.scatter(outliers.index, outliers['tank'], color='red', label='Outliers')
plt.title("Outlier Detection in Daily Tank Losses (IQR Method)")
plt.legend()
plt.show()
```




- Multiple outlier spikes in tank losses are visible, especially around late 2022 and late 2023.
- These outliers suggest intense battle periods or specific offensives leading to high single-day losses.
- The overall trend shows variability, but these outliers help flag extreme combat days worth further analysis.

## Seasonal or Temporal Patterns

In [115…

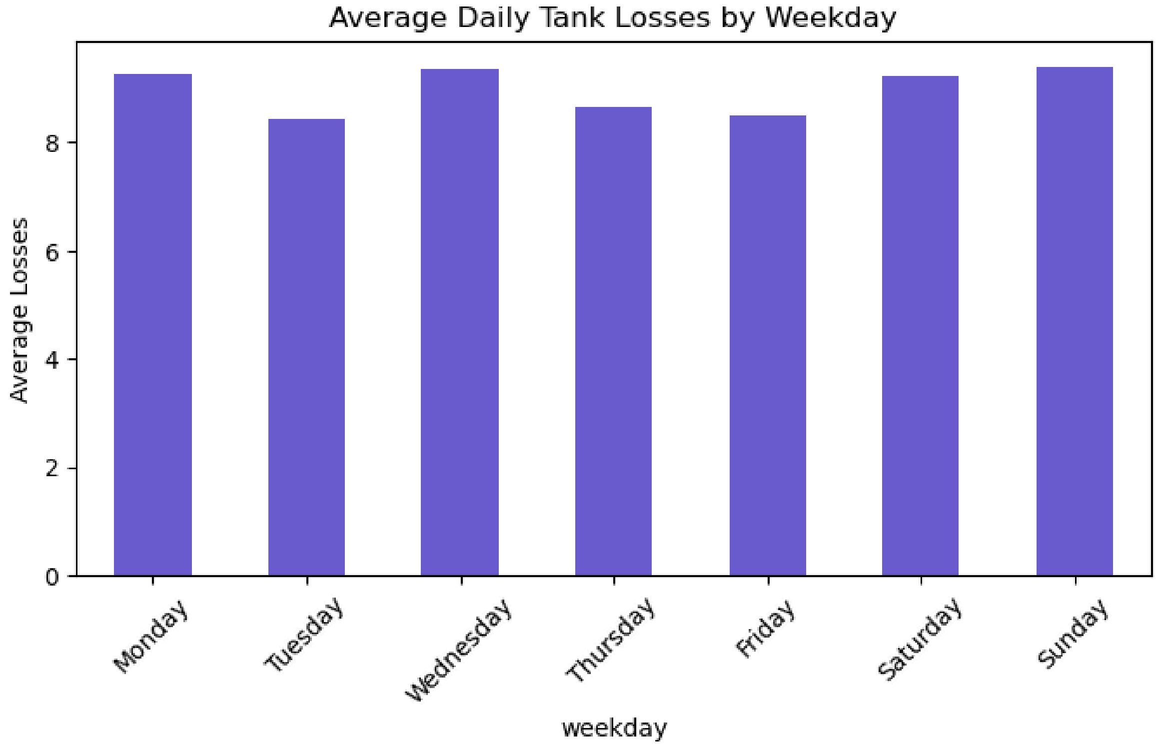
```
# Add weekday/month columns (if not already added)
data['weekday'] = data['date'].dt.day_name()
data['month'] = data['date'].dt.month_name()

# Add weekday column to daily_losses to group properly
daily_losses = daily_losses.copy()
daily_losses['weekday'] = data['weekday'].values  # align index
```

In [117…

```
# Group by weekday and calculate mean
weekday_avg = daily_losses.groupby('weekday').mean()

# Reorder weekdays correctly
weekday_avg = weekday_avg.reindex(['Monday', 'Tuesday', 'Wednesday', 'Thursday',
```

In [119…

```
# Plot
weekday_avg['tank'].plot(kind='bar', color='slateblue', figsize=(8, 4))
plt.title("Average Daily Tank Losses by Weekday")
plt.ylabel("Average Losses")
plt.xticks(rotation=45)
plt.show()
```



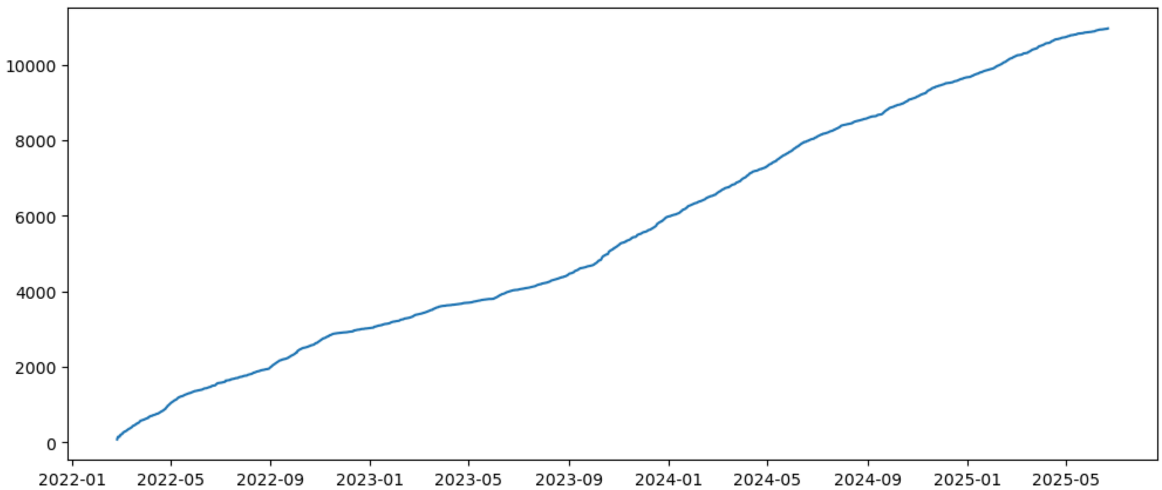

## Visual War Timeline

In [109…

```
# Plot total tank losses over time
plt.figure(figsize=(12, 5))
plt.plot(data['date'], data['tank'], label='Cumulative Tank Losses')
```

Out[109…

```
[<matplotlib.lines.Line2D at 0x20d9cfa06e0>]
```



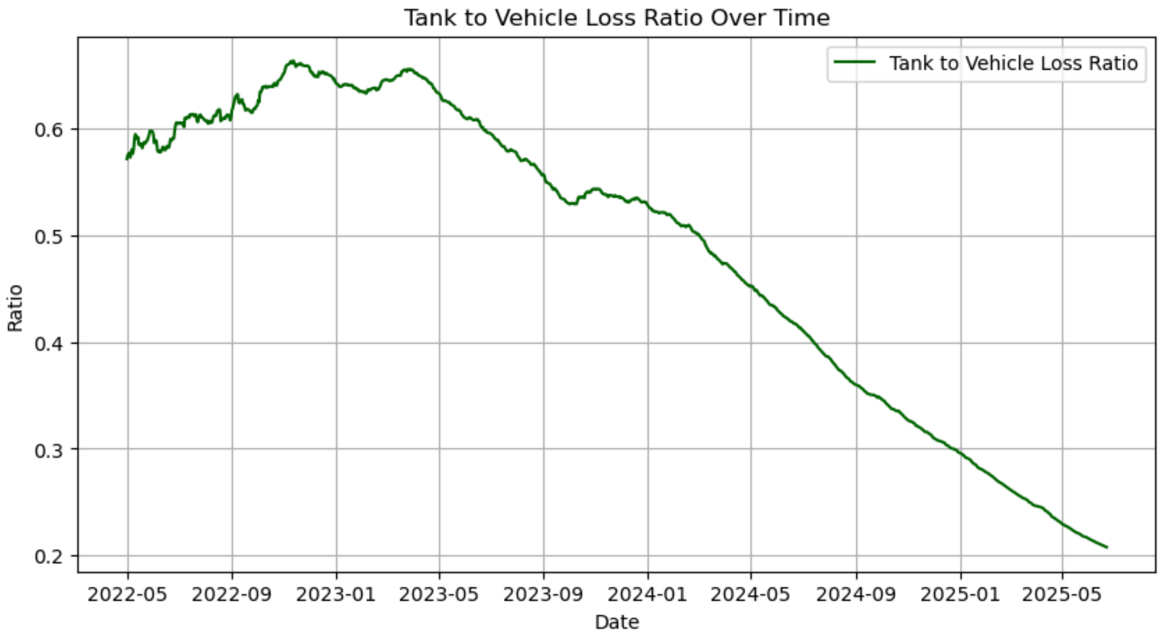

- Notable spikes around:
  - Feb 24, 2022 (start of war)
  - Sept 1, 2022 (counter-offensive)
- Shows how warfare escalated and intensified at key points.

## Mortality to Equipment Loss Ratio

In [122…

```
# Estimate "loss ratio" using proxies (e.g., tank vs. vehicles/fuel tanks)
data['loss_ratio'] = data['tank'] / data['vehicles and fuel tanks']
```

```
# Plot ratio over time
plt.figure(figsize=(10, 5))
plt.plot(data['date'], data['loss_ratio'], label='Tank to Vehicle Loss Ratio', c
plt.title("Tank to Vehicle Loss Ratio Over Time")
plt.xlabel("Date")
plt.ylabel("Ratio")
plt.grid(True)
plt.legend()
plt.show()
```

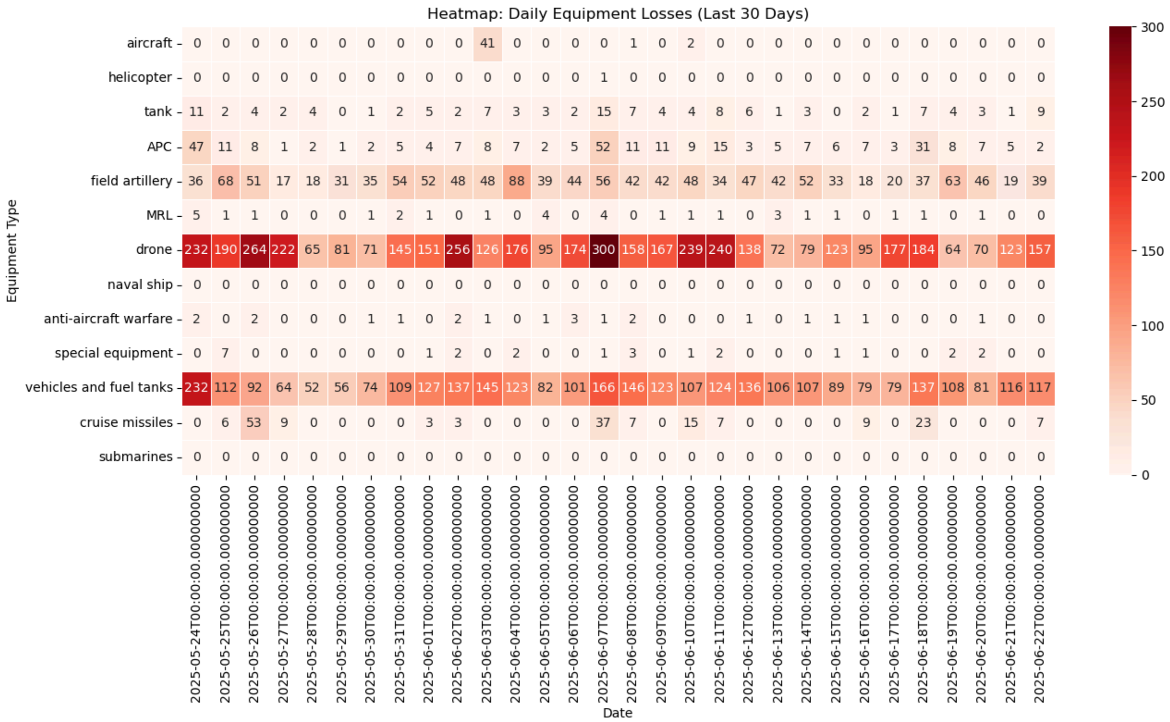



- Declining ratio indicates:
  - Tanks are increasingly outnumbered by overall vehicle losses
  - Or increased use of lighter or support vehicles

## Heatmap of Daily Losses per Equipment Type

In [130…

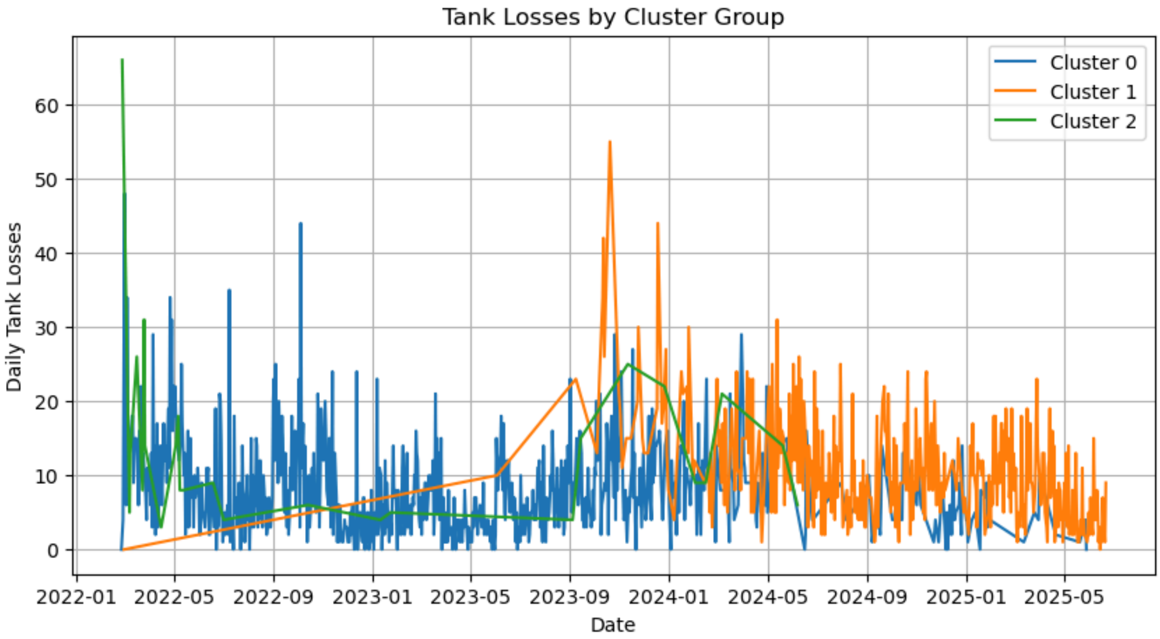
```
# Select only numeric columns (equipment types)
numeric_daily_losses = daily_losses.select_dtypes(include='number')

# Last 30 days of numeric data
last_30 = numeric_daily_losses.tail(30)

# Plot heatmap
plt.figure(figsize=(14, 8))
sns.heatmap(last_30.T, cmap='Reds', annot=True, fmt=".0f", linewidths=.5)
plt.title("Heatmap: Daily Equipment Losses (Last 30 Days)")
plt.xlabel("Date")
plt.ylabel("Equipment Type")
plt.tight_layout()
plt.show()
```




## Clustering of Loss Patterns

In [132…

```python
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Drop non-numeric or unrelated columns (like 'cluster' if previously added)
numeric_data = daily_losses.select_dtypes(include='number').copy()

# Prepare and scale data
scaler = StandardScaler()
daily_scaled = scaler.fit_transform(numeric_data.fillna(0))

# KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(daily_scaled)

# Assign cluster labels
daily_losses['cluster'] = clusters

# Plot cluster trends (e.g., tank losses by cluster)
plt.figure(figsize=(10, 5))
for c in range(3):
    cluster_data = daily_losses[daily_losses['cluster'] == c]
    plt.plot(cluster_data.index, cluster_data['tank'], label=f'Cluster {c}')
plt.title("Tank Losses by Cluster Group")
plt.ylabel("Daily Tank Losses")
plt.xlabel("Date")
plt.legend()
plt.grid(True)
plt.show()
```

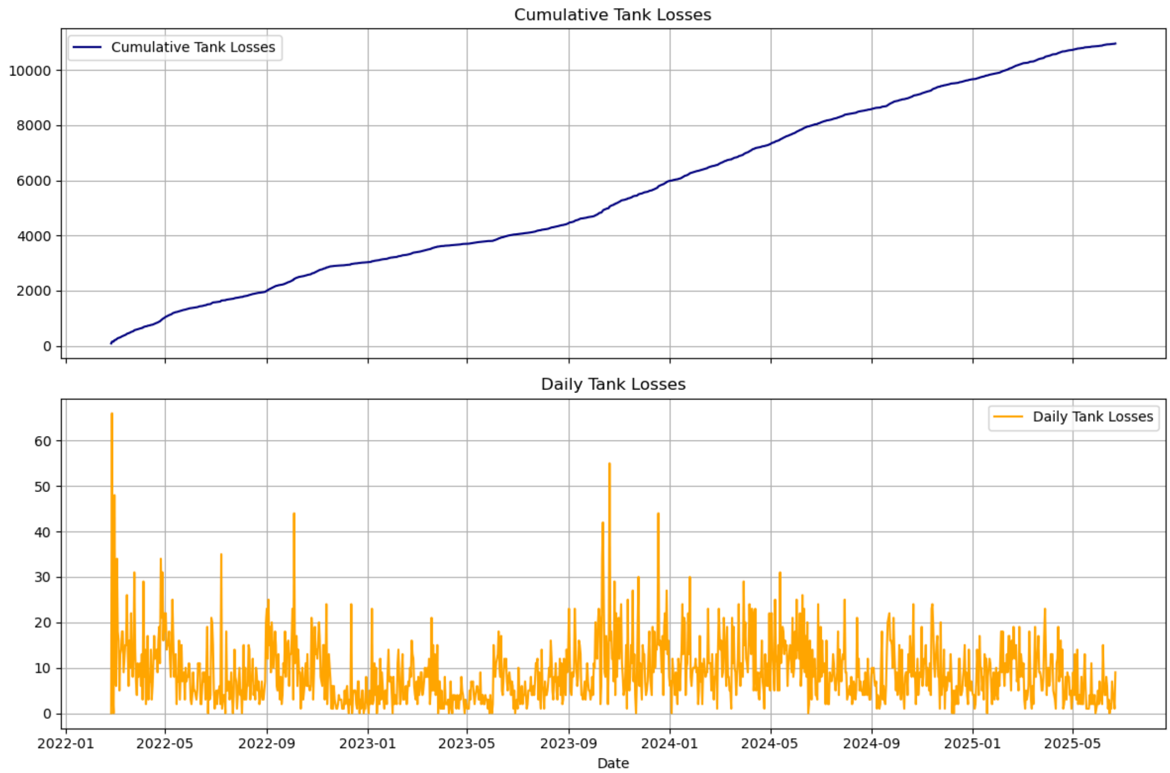



- Losses grouped into 3 clusters:
  - Each cluster corresponds to a distinct warfare intensity level.
  - Helps identify periods of escalation or calm.

## Cumulative vs. Daily Loss Plot

In [139…

```
# Compare cumulative vs. daily losses
fig, ax = plt.subplots(2, 1, figsize=(12, 8), sharex=True)

# Cumulative
ax[0].plot(data['date'], data['tank'], label='Cumulative Tank Losses', color='na
ax[0].set_title("Cumulative Tank Losses")

# Daily
ax[1].plot(daily_losses.index, daily_losses['tank'], label='Daily Tank Losses',
ax[1].set_title("Daily Tank Losses")

plt.xlabel("Date")
for a in ax:
    a.legend()
    a.grid(True)
plt.tight_layout()
plt.show()
```




- Daily plot highlights operational tempo and spikes.