```
#%% md
# Data Preparation Notebook

## Purpose
This notebook is dedicated to preparing all datasets for subsequent analysis. The focus is on cleaning, structuring, and standardizing data to ensure consistency across analyses.

## Datasets Overview
This section lists all the datasets involved, such as:
- Russian Losses as Documented by 3rd Party
- Ukrainian Losses as Reported by Ukrainian State
- [Add others as applicable]

## Tools and Libraries
This notebook utilizes Python libraries including pandas for data manipulation and NumPy for numerical operations.
#%% md
# Setup

## Import Libraries
Here, we import all necessary Python libraries needed for data preparation tasks.


#%%
# Import necessary libraries
import pandas as pd
import numpy as np
from sqlalchemy import create_engine
# TODO: Import any additional libraries needed

#%% md
# Data Loading

## Load Data
Each dataset is loaded from its respective source. Detailed instructions and code for loading each
```



```
specific dataset are provided below.


#%%
import pandas as pd

# Load the first dataset (Documented Losses for both Ukraine and Russia)
file_path_documented = '/Users/jaytonschmeeckle/Local/Repos/Math-CSCE-440-Final-Project/MATH440-Project/Data/Russia-Ukraine Equipment Losses - Original.csv'
df_documented = pd.read_csv(file_path_documented)
documented_headers = df_documented.columns.tolist()

# Load the second dataset (Russian Losses as Claimed by Ukraine)
file_path_claimed = '/Users/jaytonschmeeckle/Local/Repos/Math-CSCE-440-Final-Project/MATH440-Project/Data/Russian_Losses-ClaimedbyUkraine.csv'
df_claimed = pd.read_csv(file_path_claimed)
claimed_headers = df_claimed.columns.tolist()

# Display headers from both datasets with explicit descriptions
print("Headers from Documented Losses (both Ukraine and Russia):")
print(documented_headers)
print("\nHeaders from Russian Losses as Claimed by Ukraine:")
print(claimed_headers)

#%% md
# Data Structuring

## Reshape Data
Data is organized into a consistent format across all datasets for easier analysis.

## Merge/Concatenate
Data from different sources is combined if necessary to create a unified dataset for analysis.
```



```


#%%
# Initialize a new DataFrame for standardized data
df_standardized = pd.DataFrame()

# Date Column
df_standardized['Date'] = df_documented['Date']

# Tanks
df_standardized['RUS-Tanks-Documented'] = df_documented['Russia_Tanks']
df_standardized['RUS-Tanks-Claimed-UKR'] = df_claimed['tanks']

# Aircraft
df_standardized['RUS-Aircraft-Documented'] = df_documented['Russia_Aircraft']
df_standardized['RUS-Aircraft-Claimed-UKR'] = df_claimed['planes'] + df_claimed['helicopters'] + df_claimed['uav']

# Artillery
df_standardized['RUS-Artillery-Documented'] = df_documented['Russia_Artillery']
df_standardized['RUS-Artillery-Claimed-UKR'] = df_claimed['artillery_systems'] + df_claimed['mlrs']

# Air Defense
df_standardized['RUS-AirDefense-Documented'] = df_documented['Russia_Antiair']
df_standardized['RUS-AirDefense-Claimed-UKR'] = df_claimed['air_defense_equipment']

# Vehicles
df_standardized['RUS-Vehicles-Documented'] = df_documented['Russia_Vehicles']
df_standardized['RUS-Vehicles-Claimed-UKR'] = df_claimed['cars_and_tank_trucks']

# Personnel
```



```
df_standardized['RUS-Personnel-Documented'] = df_documented['Russia_Infantry']
df_standardized['RUS-Personnel-Claimed-UKR'] = df_claimed['personnel']

#%% md
# Output

## Save Clean Data
The cleaned and structured data is saved to new files or databases for easy access in subsequent analysis phases.

#%%
# Save the standardized data to a new CSV file
df_standardized.to_csv('standardized_data.csv', index=False)

# Or analyze directly within the notebook
print(df_standardized.head())


#%% md

#%% md
## Data Cleaning Function

This function is designed to streamline the process of checking and cleaning data for any selected category of losses (e.g., tanks, aircraft). It handles missing values, corrects data types, and ensures that the dataset is ready for further analysis. By parameterizing the function, we can apply the same cleaning logic to different data categories efficiently.

#%%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error
```



```
from scipy.optimize import newton
from scipy.interpolate import CubicSpline
import seaborn as sns

def clean_data(category):
    """Cleans data for the specified category of losses.

    Args:
        category (str): The category to clean, e.g., 'Tanks', 'Aircraft'.
    """
    documented_col = f'RUS-{category}-Documented'
    claimed_col = f'RUS-{category}-Claimed-UKR'

    # Check for missing values
    print(f"Missing values before cleaning for {category}:")
    print(df_standardized[[documented_col, claimed_col]].isnull().sum())

    # Handle missing values by forward filling
    df_standardized[documented_col].fillna(method='ffill', inplace=True)
    df_standardized[claimed_col].fillna(method='ffill', inplace=True)

    # Convert data types to integer
    df_standardized[documented_col] = df_standardized[documented_col].astype(int)
    df_standardized[claimed_col] = df_standardized[claimed_col].astype(int)

    # Verify changes
    print(f"Data types after conversion for {category}:")
    print(df_standardized[[documented_col, claimed_col]].dtypes)
    print("\nSample data after cleaning:")
    print(df_standardized[[documented_col, claimed_col]].head())
```



```

clean_data('Tanks')  # Modify 'Tanks' to any other category needed

#%% md
## Verifying Changes Across Categories

After applying the cleaning function to different categories, this section is intended for final verification. You can run diagnostics, visualize data, or apply further checks to ensure that all categories have been cleaned and are ready for in-depth analysis.

#%%
# Example of how to check data across multiple categories
categories = ['Tanks', 'Aircraft', 'Artillery', 'AirDefense', 'Vehicles', 'Personnel']
for category in categories:
    documented_col = f'RUS-{category}-Documented'
    claimed_col = f'RUS-{category}-Claimed-UKR'
    print(f"{category} - Missing values check:")
    print(df_standardized[[documented_col, claimed_col]].isnull().sum())
    print(f"{category} - Data type check:")
    print(df_standardized[[documented_col, claimed_col]].dtypes)
    print()

#%% md
## Error Analysis

Calculate the Mean Absolute Error (MAE) and Mean Squared Error (MSE) between the documented and claimed data to quantify discrepancies.

#%%
def calculate_errors(documented, claimed):
    mae = mean_absolute_error(documented, claimed)
    mse = mean_squared_error(documented, claimed)
```

```
178     return mae, mse
179
180 documented_tanks = df_standardized['RUS-Tanks-
    Documented']
181 claimed_tanks = df_standardized['RUS-Tanks-Claimed
    -UKR']
182 tanks_mae, tanks_mse = calculate_errors(
    documented_tanks, claimed_tanks)
183
184 print(f"Mean Absolute Error for Tanks: {tanks_mae}
    ")
185 print(f"Mean Squared Error for Tanks: {tanks_mse}"
    )
186
187 #%% md
188 ## Linear Regression Analysis
189
190 Model the relationship between documented and
    claimed tank losses using linear regression to
    assess predictability and consistency of the
    reported data.
191
192 #%%
193 x = documented_tanks.values.reshape(-1, 1)
194 y = claimed_tanks.values.reshape(-1, 1)
195 A = np.vstack([x.T, np.ones(len(x))]).T
196 m, c = np.linalg.lstsq(A, y, rcond=None)[0]
197
198 plt.figure(figsize=(20, 10))
199 plt.plot(x, y, 'o', label='Original data',
    markersize=10)
200 plt.plot(x, m*x + c, 'r', label='Fitted line')
201 plt.xlabel('Documented Tank Losses')
202 plt.ylabel('Claimed Tank Losses by Ukraine')
203 plt.title('Linear Regression Analysis of Tank
    Losses')
204 plt.legend()
205 plt.show()
206
207 print(f"Slope: {m[0]}, Intercept: {c[0]}")
208
```