In [3]:
```
#Importing necessary libraries
import pandas as pd
```

In [5]:
```
import numpy as np
```

In [7]:
```
import matplotlib.pyplot as plt
```

In [8]:
```
import seaborn as sns
```

In [10]:
```
!pip install missingno
```

```
Requirement already satisfied: missingno in c:\users\lucil\anaconda3\lib\site-pac
kages (0.5.2)
Requirement already satisfied: numpy in c:\users\lucil\anaconda3\lib\site-package
s (from missingno) (1.26.4)
Requirement already satisfied: matplotlib in c:\users\lucil\anaconda3\lib\site-pa
ckages (from missingno) (3.8.4)
Requirement already satisfied: scipy in c:\users\lucil\anaconda3\lib\site-package
s (from missingno) (1.13.1)
Requirement already satisfied: seaborn in c:\users\lucil\anaconda3\lib\site-packa
ges (from missingno) (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\lucil\anaconda3\lib\s
ite-packages (from matplotlib->missingno) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\lucil\anaconda3\lib\site-
packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\lucil\anaconda3\lib
\site-packages (from matplotlib->missingno) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\lucil\anaconda3\lib
\site-packages (from matplotlib->missingno) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\lucil\anaconda3\lib\si
te-packages (from matplotlib->missingno) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\lucil\anaconda3\lib\site-pac
kages (from matplotlib->missingno) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\lucil\anaconda3\lib\s
ite-packages (from matplotlib->missingno) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\lucil\anaconda3\l
ib\site-packages (from matplotlib->missingno) (2.9.0.post0)
Requirement already satisfied: pandas>=1.2 in c:\users\lucil\anaconda3\lib\site-p
ackages (from seaborn->missingno) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\lucil\anaconda3\lib\site-
packages (from pandas>=1.2->seaborn->missingno) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\lucil\anaconda3\lib\sit
e-packages (from pandas>=1.2->seaborn->missingno) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\lucil\anaconda3\lib\site-pack
ages (from python-dateutil>=2.7->matplotlib->missingno) (1.16.0)
```

In [11]:
```
import missingno as msno
```

In [12]:
```
#Loading the dataset
df = pd.read_csv(r"C:\Users\lucil\Downloads\AviationData.csv.zip", encoding="lat
```

In [14]:
```
# Check if the DataFrame is loaded
print(df.head())  # Shows the first few rows of df
```

```
        Event.Id Investigation.Type Accident.Number  Event.Date  \
0  20001218X45444           Accident      SEA87LA080  1948-10-24
1  20001218X45447           Accident      LAX94LA336  1962-07-19
2  20061025X01555           Accident      NYC07LA005  1974-08-30
3  20001218X45448           Accident      LAX96LA321  1977-06-19
4  20041105X01764           Accident      CHI79FA064  1979-08-02

          Location        Country   Latitude   Longitude Airport.Code  \
0  MOOSE CREEK, ID  United States        NaN         NaN          NaN
1   BRIDGEPORT, CA  United States        NaN         NaN          NaN
2    Saltville, VA  United States  36.922223  -81.878056          NaN
3       EUREKA, CA  United States        NaN         NaN          NaN
4       Canton, OH  United States        NaN         NaN          NaN

  Airport.Name  ... Purpose.of.flight Air.carrier Total.Fatal.Injuries  \
0          NaN  ...          Personal         NaN                  2.0
1          NaN  ...          Personal         NaN                  4.0
2          NaN  ...          Personal         NaN                  3.0
3          NaN  ...          Personal         NaN                  2.0
4          NaN  ...          Personal         NaN                  1.0

  Total.Serious.Injuries Total.Minor.Injuries Total.Uninjured  \
0                    0.0                  0.0             0.0
1                    0.0                  0.0             0.0
2                    NaN                  NaN             NaN
3                    0.0                  0.0             0.0
4                    2.0                  NaN             0.0

  Weather.Condition  Broad.phase.of.flight   Report.Status Publication.Date
0               UNK                 Cruise  Probable Cause             NaN
1               UNK                Unknown  Probable Cause      19-09-1996
2               IMC                 Cruise  Probable Cause      26-02-2007
3               IMC                 Cruise  Probable Cause      12-09-2000
4               VMC               Approach  Probable Cause      16-04-1980

[5 rows x 31 columns]
```

In [15]:
```
#Taking a first look at the dataset
df.head()
```

Out[15]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Cou
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	U S
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	U S
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	U S
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	U S
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	U S

5 rows × 31 columns

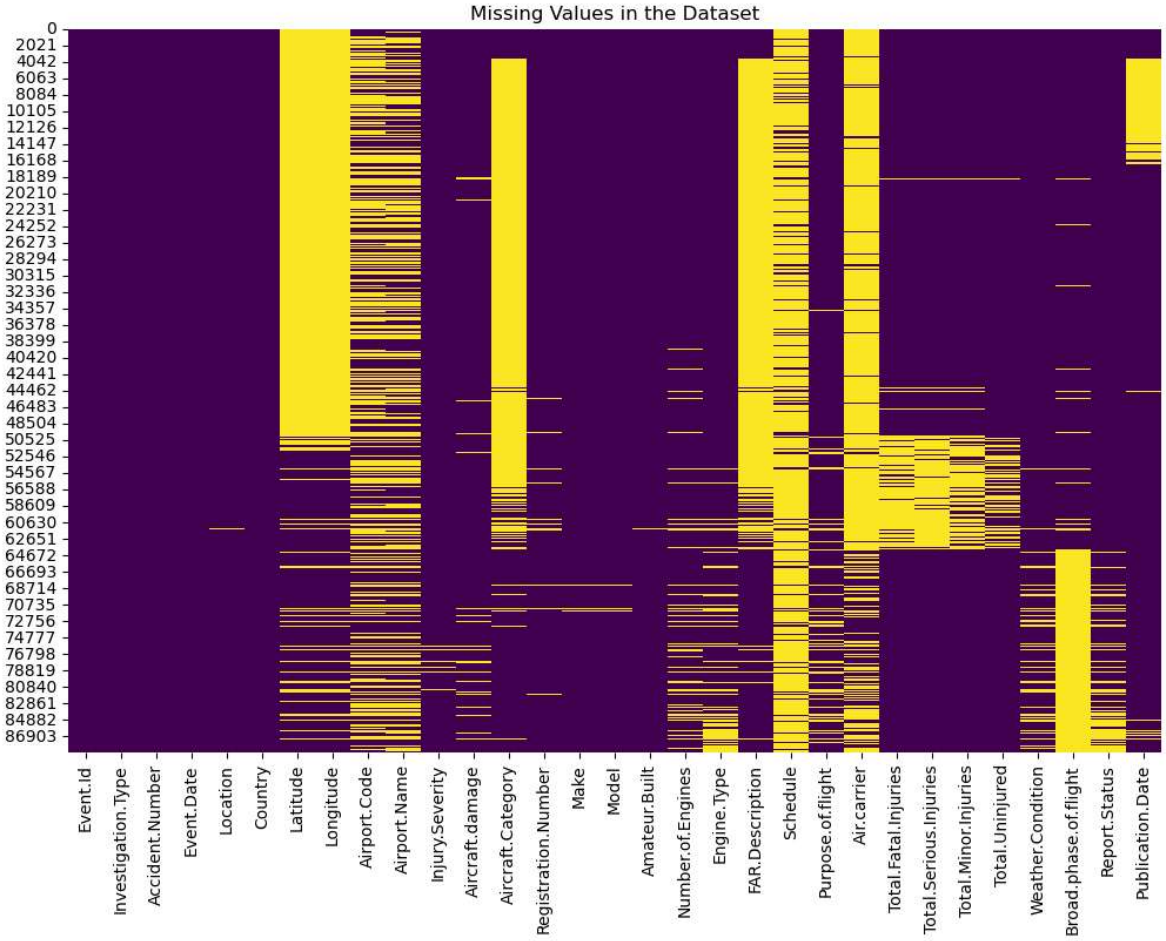
In [21]:
```
#To check details and structure of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      88889 non-null  object 
 2   Accident.Number         88889 non-null  object 
 3   Event.Date              88889 non-null  object 
 4   Location                88837 non-null  object 
 5   Country                 88663 non-null  object 
 6   Latitude                34382 non-null  object 
 7   Longitude               34373 non-null  object 
 8   Airport.Code            50132 non-null  object 
 9   Airport.Name            52704 non-null  object 
 10  Injury.Severity         87889 non-null  object 
 11  Aircraft.damage         85695 non-null  object 
 12  Aircraft.Category       32287 non-null  object 
 13  Registration.Number     87507 non-null  object 
 14  Make                    88826 non-null  object 
 15  Model                   88797 non-null  object 
 16  Amateur.Built           88787 non-null  object 
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81793 non-null  object 
 19  FAR.Description         32023 non-null  object 
 20  Schedule                12582 non-null  object 
 21  Purpose.of.flight       82697 non-null  object 
 22  Air.carrier             16648 non-null  object 
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object 
 28  Broad.phase.of.flight   61724 non-null  object 
 29  Report.Status           82505 non-null  object 
 30  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

In [23]:
```
#To check for null values in the dataset
df.isnull().sum()
```

Out[23]:
```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                   52
Country                   226
Latitude                54507
Longitude               54516
Airport.Code            38757
Airport.Name            36185
Injury.Severity          1000
Aircraft.damage          3194
Aircraft.Category       56602
Registration.Number      1382
Make                       63
Model                      92
Amateur.Built             102
Number.of.Engines        6084
Engine.Type              7096
FAR.Description         56866
Schedule                76307
Purpose.of.flight        6192
Air.carrier             72241
Total.Fatal.Injuries    11401
Total.Serious.Injuries  12510
Total.Minor.Injuries    11933
Total.Uninjured          5912
Weather.Condition        4492
Broad.phase.of.flight   27165
Report.Status            6384
Publication.Date        13771
dtype: int64
```

In [25]:
```
#To show visual overview of null values
plt.figure(figsize=(12, 8))
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values in the Dataset")
plt.show()
```



In [26]:
```
#To check for duplicates in the dataset
duplicates_count = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicates_count}")
```

```
Number of duplicate rows: 0
```

In [27]:
```
#To show the spread and shape of a dataset's distribution
df.describe()
```

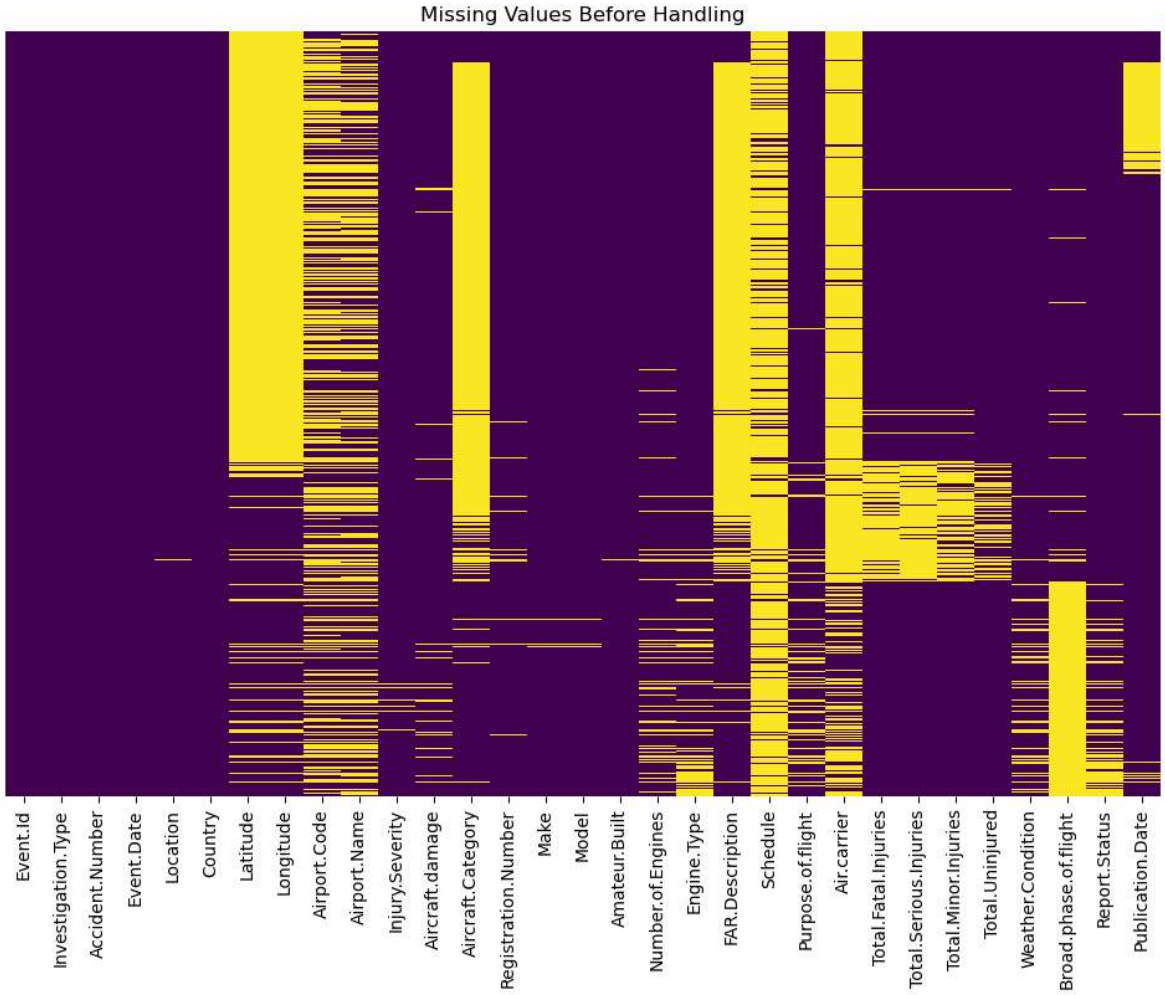
Out[27]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries
count	82805.000000	77488.000000	76379.000000	76956.000000
mean	1.146585	0.647855	0.279881	0.357061
std	0.446510	5.485960	1.544084	2.235625
min	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000
75%	1.000000	0.000000	0.000000	0.000000
max	8.000000	349.000000	161.000000	380.000000

In [31]:
```
#To check for missing values in each coloumn
df.isnull().sum()
```

Out[31]:
```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                   52
Country                   226
Latitude                54507
Longitude               54516
Airport.Code            38757
Airport.Name            36185
Injury.Severity          1000
Aircraft.damage          3194
Aircraft.Category       56602
Registration.Number      1382
Make                       63
Model                      92
Amateur.Built             102
Number.of.Engines        6084
Engine.Type              7096
FAR.Description         56866
Schedule                76307
Purpose.of.flight        6192
Air.carrier             72241
Total.Fatal.Injuries    11401
Total.Serious.Injuries  12510
Total.Minor.Injuries    11933
Total.Uninjured          5912
Weather.Condition        4492
Broad.phase.of.flight   27165
Report.Status            6384
Publication.Date        13771
dtype: int64
```

In [33]:
```
# Visualizing missing values with a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df.isnull(), cbar=False, cmap='viridis', yticklabels=False)
plt.title("Missing Values Before Handling")
plt.show()
```



In [35]:
```
#Returns boolean if each coloumn has missing values
df.isnull().any()
```

Out[35]:
```
Event.Id                 False
Investigation.Type       False
Accident.Number          False
Event.Date               False
Location                  True
Country                   True
Latitude                  True
Longitude                 True
Airport.Code              True
Airport.Name              True
Injury.Severity           True
Aircraft.damage           True
Aircraft.Category         True
Registration.Number       True
Make                      True
Model                     True
Amateur.Built             True
Number.of.Engines         True
Engine.Type               True
FAR.Description           True
Schedule                  True
Purpose.of.flight         True
Air.carrier               True
Total.Fatal.Injuries      True
Total.Serious.Injuries    True
Total.Minor.Injuries      True
Total.Uninjured           True
Weather.Condition         True
Broad.phase.of.flight     True
Report.Status             True
Publication.Date          True
dtype: bool
```

In [37]:
```
#To drop rows where all values are missing
df_cleaned = df.dropna(how='all')
```

In [39]:
```
# Drop rows with missing 'Event.Id' or 'Accident.Number' since they are critical
df.dropna(subset=['Event.Id', 'Accident.Number'], inplace=True)
```

In [47]:
```
# Using mode for categorical columns, median for numerical where applicable
# Location and Country: Impute with the most common value (mode)
df['Location'] = df['Location'].fillna(df['Location'].mode()[0])
df['Country'] = df['Country'].fillna(df['Country'].mode()[0])
```

In [51]:
```
# Injury Severity: Impute with the most common injury severity(mode)
df['Injury.Severity'] = df['Injury.Severity'].fillna(df['Injury.Severity'].mode(
```

In [43]:
```
# Fill with mode for specific categorical columns
categorical_mode_cols = ['Make', 'Model', 'Registration.Number']
for col in categorical_mode_cols:
    df[col] = df[col].fillna(df[col].mode()[0])
```

In [69]:
```
# Engine Type: Impute based on the most common values
df['Engine.Type'] = df['Engine.Type'].fillna(df['Engine.Type'].mode()[0])
```

In [55]:
```
# Latitude and Longitude: Consider dropping if they are non-essential due to hig
df.drop(columns=['Latitude', 'Longitude'], inplace=True)
```

In [57]:
```
# Airport Code and Airport Name: Drop if non-essential, or fill with "Unknown" i
df['Airport.Code'] = df['Airport.Code'].fillna("Unknown")
df['Airport.Name'] = df['Airport.Name'].fillna("Unknown")
```

In [55]:
```
# Aircraft Category, FAR Description, Schedule, Purpose of Flight, Air Carrier:
df.drop(columns=['Aircraft.Category', 'FAR.Description', 'Schedule', 'Purpose.of
```

In [59]:
```
# Injury Columns: If missing, fill with zero (assuming missing implies no injuri
df['Total.Fatal.Injuries'] = df['Total.Fatal.Injuries'].fillna(0)
df['Total.Serious.Injuries'] = df['Total.Serious.Injuries'].fillna(0)
df['Total.Minor.Injuries'] = df['Total.Minor.Injuries'].fillna(0)
df['Total.Uninjured'] = df['Total.Uninjured'].fillna(0)
```

In [61]:
```
 # Weather Condition: Impute with the most common weather condition
df['Weather.Condition'] = df['Weather.Condition'].fillna(df['Weather.Condition']
```

In [63]:
```
 #Broad Phase of Flight: Impute with the most common phase
df['Broad.phase.of.flight'] = df['Broad.phase.of.flight'].fillna(df['Broad.phase
```

In [65]:
```
# Report Status: Impute with the most common report status
df['Report.Status'] = df['Report.Status'].fillna(df['Report.Status'].mode()[0])
```

In [67]:
```
# Publication Date: Drop rows with missing publication dates if it's necessary f
df = df.dropna(subset=['Publication.Date'])
```

In [73]:
```
#For numerical coloumns with few missing values
numerical_mode_cols = ['Amateur.Built']
for col in numerical_mode_cols:
    df[col] = df[col].fillna(df[col].mode()[0])
```

In [75]:
```
# For 'Number.of.Engines', fill with the median
df['Number.of.Engines'] = df['Number.of.Engines'].fillna(df['Number.of.Engines']
```

In [79]:
```
# Aircraft.damage: Impute with the most frequent damage type
df['Aircraft.damage'] = df['Aircraft.damage'].fillna(df['Aircraft.damage'].mode(
```

In [85]:
```
# Purpose.of.flight: Impute with the most common purpose of flight
df['Purpose.of.flight'] = df['Purpose.of.flight'].fillna(df['Purpose.of.flight']
```

In [89]:
```
#Drop due go high missingness
df.drop(['Air.carrier'], axis=1, inplace=True)
```

In [91]:
```
#Drop due to high missingness
df.drop(['Aircraft.Category', 'FAR.Description'], axis=1, inplace=True, errors='
```

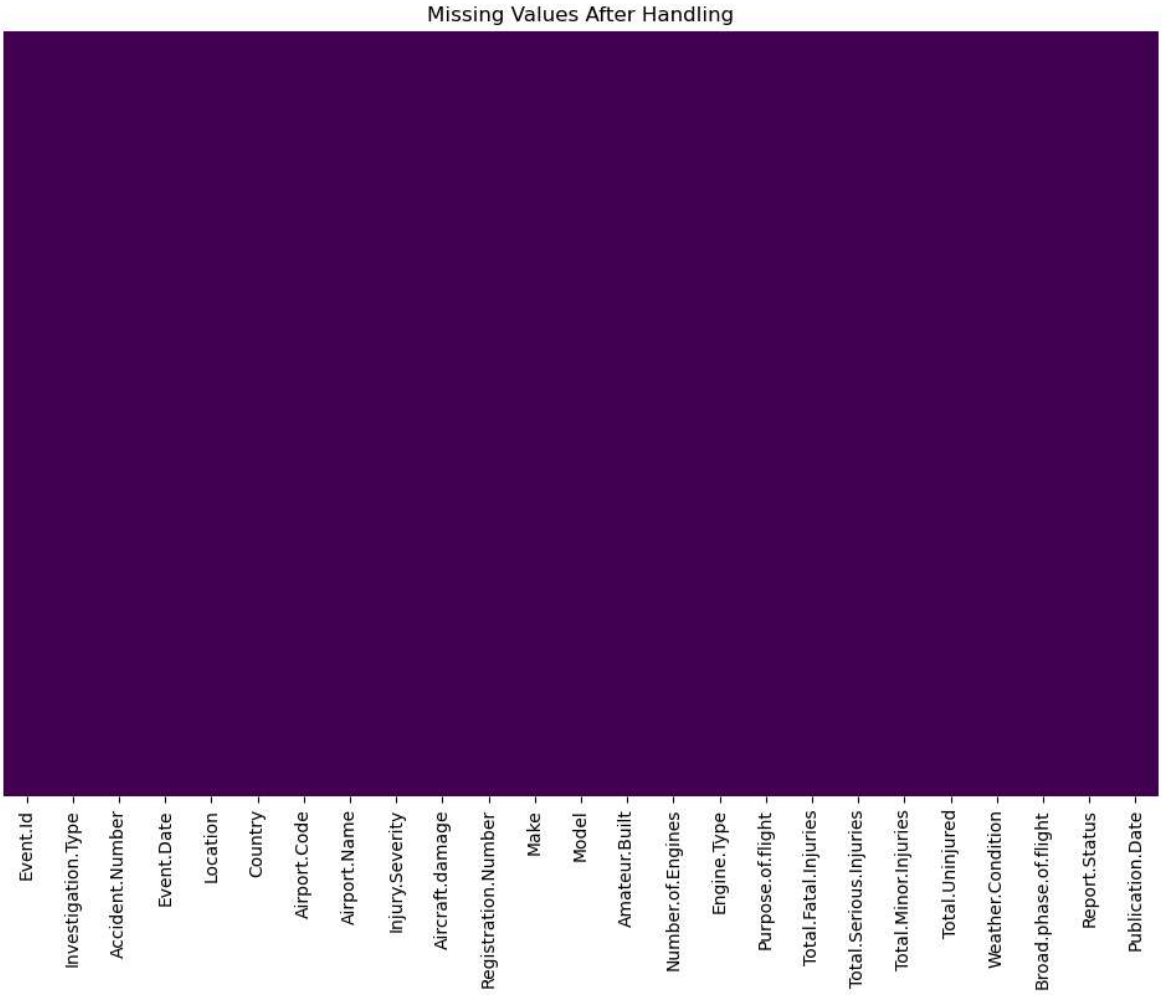
In [99]:
```
# Schedule: Drop this column due to high missingness unless essential
df.drop(['Schedule'], axis=1, inplace=True, errors='ignore')
```

In [101…
```
# Check remaining missing values to ensure all are handled
print("Remaining Missing Values:\n", df.isnull().sum())
```

```
Remaining Missing Values:
 Event.Id                 0
Investigation.Type        0
Accident.Number           0
Event.Date                0
Location                  0
Country                   0
Airport.Code              0
Airport.Name              0
Injury.Severity           0
Aircraft.damage           0
Registration.Number       0
Make                      0
Model                     0
Amateur.Built             0
Number.of.Engines         0
Engine.Type               0
Purpose.of.flight         0
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
Weather.Condition         0
Broad.phase.of.flight     0
Report.Status             0
Publication.Date          0
dtype: int64
```

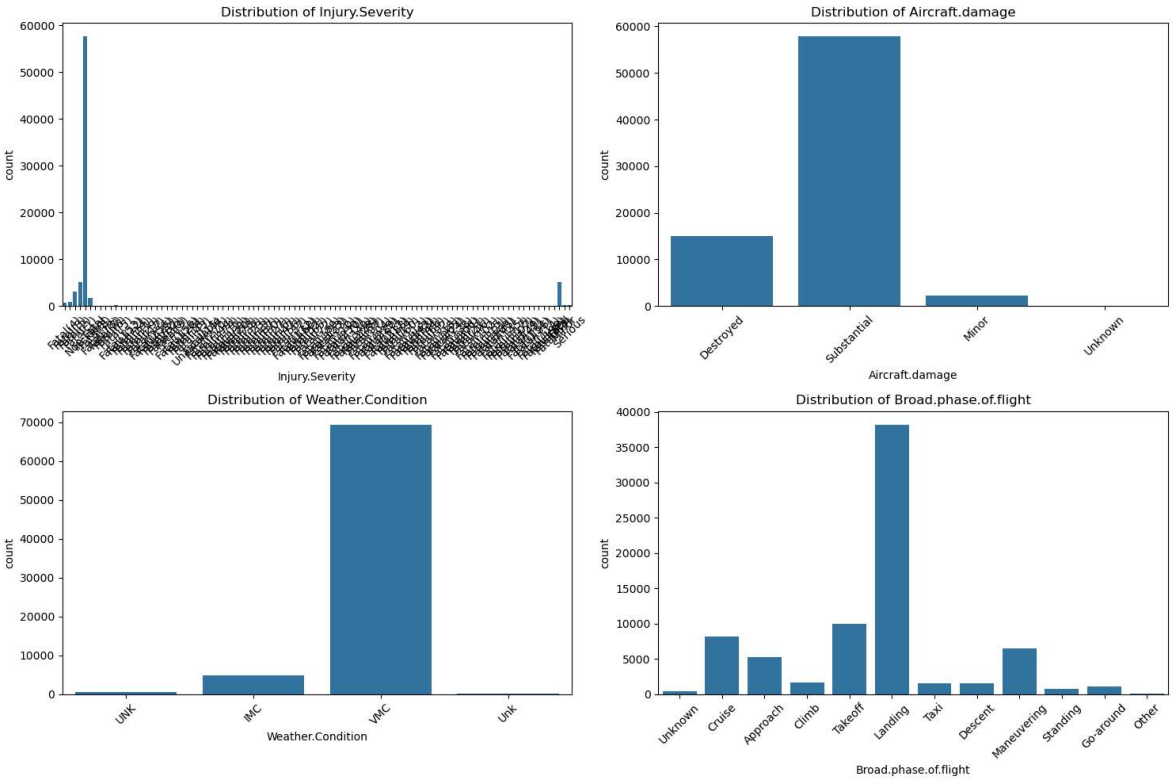
In [103…

```
# Heatmap to confirm no missing values remain (if all missing values are handled
plt.figure(figsize=(12, 8))
sns.heatmap(df.isnull(), cbar=False, cmap='viridis', yticklabels=False)
plt.title("Missing Values After Handling")
plt.show()
```



In [104…

```
# Plot categorical columns
plt.figure(figsize=(15, 10))
categorical_columns = ['Injury.Severity', 'Aircraft.damage', 'Weather.Condition'
for i, col in enumerate(categorical_columns, 1):
    plt.subplot(2, 2, i)
    sns.countplot(data=df, x=col)
    plt.xticks(rotation=45)
    plt.title(f"Distribution of {col}")
plt.tight_layout()
plt.show()
```



In [105…

```
# Plot numerical columns (e.g., Injury counts)
plt.figure(figsize=(12, 6))
injury_columns = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor
for col in injury_columns:
    sns.histplot(df[col], kde=True, label=col)
plt.title("Distribution of Injury Columns")
plt.legend()
plt.show()
```