wnfmvyugk

December 31, 2024

1 Descripción de las columnas del dataset de bombardeos de EE.UU. en Corea

1.1 Datos generales de la misión

- **ROW_NUMBER**: Número único de fila en el dataset (índice del registro).
- **MISSION_NUMBER**: Identificador de la misión de bombardeo.
- **OP_ORDER**: Código del orden operacional asociado a la misión.
- **UNIT**: Unidad militar encargada del bombardeo (e.g., "98th Bomb Wing").
- **MISSION_DATE**: Fecha de la misión.

1.2 Información sobre la aeronave

- **AIRCRAFT_TYPE_MDS**: Tipo o modelo de aeronave utilizada (e.g., B-29).
- **NBR_ATTACK_EFFEC_AIRCRAFT**: Número de aeronaves efectivas en el ataque.
- **SORTIE_DUPE**: Número de salidas duplicadas (operaciones repetidas).
- **NBR_ABORT_AIRCRAFT**: Número de aeronaves que abortaron la misión.
- **NBR_LOST_AIRCRAFT**: Número de aeronaves perdidas durante la misión.

1.3 Objetivo de la misión

- **TARGET_NAME**: Nombre del objetivo bombardeado.
- **TGT_TYPE**: Tipo de objetivo (e.g., infraestructura, instalaciones militares).
- **SOURCE_UTM_JAPAN_B**: Sistema de referencia UTM utilizado (en Japón, datos escasos).
- **SOURCE_TGT_UTM**: Coordenadas UTM del objetivo.
- **TGT_MGRS**: Coordenadas del objetivo en formato MGRS (Military Grid Reference System).
- **TGT_LATITUDE_WGS84**: Latitud del objetivo (formato WGS84).
- **TGT_LONGITUDE_WGS84**: Longitud del objetivo (formato WGS84).
- **SOURCE_TGT_LAT**: Latitud del objetivo obtenida de otra fuente.
- **SOURCE_TGT_LONG**: Longitud del objetivo obtenida de otra fuente.

1.4 Información sobre el armamento

- **NBR_OF_WEAPONS**: Número de armas utilizadas.
- **WEAPONS_TYPE**: Tipo de armas utilizadas (e.g., bombas, cohetes).
- **BOMB_SIGHTING_METHOD**: Método utilizado para apuntar las bombas (e.g., visual, radar).

- **TOTAL_BOMBLOAD_IN_LBS**: Carga total de bombas en libras.
- **NOSE_FUZE**: Tipo de espoleta en la nariz de las bombas.
- **TAIL_FUZE**: Tipo de espoleta en la cola de las bombas.
- **CALCULATED_BOMBLOAD_LBS**: Carga calculada de bombas en libras.

1.5 Evaluación y resultados

- **TOT**: Hora sobre el objetivo ("Time Over Target").
- **MISSION_TYPE**: Tipo de misión (e.g., ataque estratégico, reconocimiento).
- **ALTITUDE_FT**: Altitud en pies durante el bombardeo.
- **BDA**: Evaluación de daños del bombardeo ("Bomb Damage Assessment").

1.6 Información adicional

- **CALLSIGN**: Indicativo de llamada del avión (datos escasos).
- **RECORD_SOURCE**: Fuente del registro (e.g., "EXETER", sistema o archivo de origen).

2

---

3 IMPORTAR LIBRERIAS

[173]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import folium
from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut
import time
#from googletrans import Translator
import re
import warnings
warnings.filterwarnings("ignore", category=pd.errors.SettingWithCopyWarning)
```

4

---

5 LECTURA BASES DE DATOS

[186]:
```
file_path = './data/THOR_KOREAN_EXTR_DATA.csv'
df_bombardeos = pd.read_csv(file_path, sep=',')
df_bombardeos.head()
```

[186]:
```
   ROW_NUMBER MISSION_NUMBER OP_ORDER             UNIT MISSION_DATE  \
0           2            433   174-51   98th Bomb Wing       6/1/51
1           3            433   174-51  307th Bomb Wing       6/1/51
2           4            433   174-51  307th Bomb Wing       6/1/51
3           5            433   174-51   98th Bomb Wing       6/1/51
4           6            433   174-51   98th Bomb Wing       6/1/51

  AIRCRAFT_TYPE_MDS  NBR_ATTACK_EFFEC_AIRCRAFT  SORTIE_DUPE  \
0              B-29                        1.0          NaN
1              B-29                        NaN          1.0
2              B-29                        1.0          1.0
3              B-29                        1.0          NaN
4              B-29                        1.0          NaN

   NBR_ABORT_AIRCRAFT NBR_LOST_AIRCRAFT  … TOTAL_BOMBLOAD_IN_LBS  TOT  \
0                 NaN               NaN  …               12000.0  NaN
1                 NaN               NaN  …                   NaN  NaN
2                 NaN               NaN  …                   NaN  NaN
3                 NaN               NaN  …               16000.0  NaN
4                 NaN               NaN  …               16000.0  NaN

  MISSION_TYPE    ALTITUDE_FT CALLSIGN  \
0          NaN          19750      NaN
1          NaN            NaN      NaN
2          NaN  21000 - 22500      NaN
3          NaN          21500      NaN
4          NaN          16000      NaN

                                              BDA NOSE_FUZE  TAIL_FUZE  \
0          Bombs fell on the east end of the tracks      0.01  Non-delay
1                                              NaN      0.01  Non-delay
2  1 aircraft due to a bomb rack malfunction drop…      0.01  Non-delay
3                                              NaN      0.01  Non-delay
4                                              NaN      0.01  Non-delay

  CALCULATED_BOMBLOAD_LBS RECORD_SOURCE
0                 12000.0        EXETER
1                  4000.0        EXETER
2                  8000.0        EXETER
3                 16000.0        EXETER
4                 16000.0        EXETER

[5 rows x 32 columns]
```

[187]:
```
file_path2 = './data/THOR_KOREAN_DATA.csv'
df_bombardeos_2 = pd.read_csv(file_path2, sep=',')
df_bombardeos_2.head()
```

[187]:
```
   KOREAN_ID MSN_DATE UNIT_ID UNIT_ID_2 UNIT_ID_CODE  \
0          1   2/1/51   003BL     008BL   003BL008BL
1          2  6/28/50   003BL       NaN        003BL
2          3  6/28/50   003BL       NaN        003BL
3          4  6/29/50   003BL       NaN        003BL
4          5  6/29/50   003BL       NaN        003BL

         GROUP_OR_HIGHER_UNIT_ID                    SQUADRON_ID AIRFIELD_ID  \
0                            NaN  3rd Bombardment Group (Light)         NaN
1  3rd Bombardment Group (Light)                            NaN        G841
2  3rd Bombardment Group (Light)                            NaN        G841
3  3rd Bombardment Group (Light)                            NaN        G841
4  3rd Bombardment Group (Light)                            NaN        G841

  LAUNCH_BASE LAUNCH_COUNTRY  …  AC_LOST_TO_OTHER  AC_DAMAGED  KIA  WIA  \
0         NaN            NaN  …               NaN         NaN  NaN  NaN
1         NaN            NaN  …               NaN         NaN  4.0  NaN
2         NaN            NaN  …               NaN         NaN  4.0  NaN
3         NaN            NaN  …               NaN         NaN  NaN  NaN
4         NaN            NaN  …               NaN         NaN  NaN  NaN

    MIA  EAC_CONFIRMED_DESTROYED  EAC_PROB_DESTROYED  TOTAL_TONS  ROCKETS  \
0   NaN                      NaN                 NaN        25.0     10.0
1   NaN                      NaN                 NaN        28.0      NaN
2   NaN                      NaN                 NaN        28.0      NaN
3  20.0                      NaN                 NaN        24.0      NaN
4  20.0                      NaN                 NaN        24.0      NaN

   BULLETS
0     74.0
1    132.0
2    132.0
3    113.0
4    113.0

[5 rows x 29 columns]
```

[176]:
```
df_bombardeos_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12878 entries, 0 to 12877
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   KOREAN_ID               12878 non-null  int64
 1   MSN_DATE                12878 non-null  object
 2   UNIT_ID                 12822 non-null  object
```

```
 3   UNIT_ID_2                12847 non-null  object
 4   UNIT_ID_CODE             12875 non-null  object
 5   GROUP_OR_HIGHER_UNIT_ID  2992 non-null   object
 6   SQUADRON_ID              5088 non-null   object
 7   AIRFIELD_ID              12870 non-null  object
 8   LAUNCH_BASE              2098 non-null   object
 9   LAUNCH_COUNTRY           2098 non-null   object
 10  LAUNCH_LAT               0 non-null      float64
 11  LAUNCH_LONG              0 non-null      float64
 12  AC_TYPE                  12877 non-null  object
 13  AC_DISPATCHED            12869 non-null  float64
 14  AC_EFFECTIVE             12766 non-null  float64
 15  AC_ABORT                 2537 non-null   float64
 16  AC_LOST_TO_EAC           23 non-null     float64
 17  AC_LOST_TO_AAA           148 non-null    float64
 18  AC_LOST_TO_UNKNOWN_EA    91 non-null     float64
 19  AC_LOST_TO_OTHER         158 non-null    float64
 20  AC_DAMAGED               449 non-null    float64
 21  KIA                      55 non-null     float64
 22  WIA                      74 non-null     float64
 23  MIA                      190 non-null    float64
 24  EAC_CONFIRMED_DESTROYED  67 non-null     float64
 25  EAC_PROB_DESTROYED       38 non-null     float64
 26  TOTAL_TONS               5155 non-null   float64
 27  ROCKETS                  3846 non-null   float64
 28  BULLETS                  6197 non-null   float64
dtypes: float64(18), int64(1), object(10)
memory usage: 2.8+ MB
```

[123]:
```
df_bombardeos_2.columns
```

[123]:
```
Index(['KOREAN_ID', 'MSN_DATE', 'UNIT_ID', 'UNIT_ID_2', 'UNIT_ID_CODE',
       'GROUP_OR_HIGHER_UNIT_ID', 'SQUADRON_ID', 'AIRFIELD_ID', 'LAUNCH_BASE',
       'LAUNCH_COUNTRY', 'LAUNCH_LAT', 'LAUNCH_LONG', 'AC_TYPE',
       'AC_DISPATCHED', 'AC_EFFECTIVE', 'AC_ABORT', 'AC_LOST_TO_EAC',
       'AC_LOST_TO_AAA', 'AC_LOST_TO_UNKNOWN_EA', 'AC_LOST_TO_OTHER',
       'AC_DAMAGED', 'KIA', 'WIA', 'MIA', 'EAC_CONFIRMED_DESTROYED',
       'EAC_PROB_DESTROYED', 'TOTAL_TONS', 'ROCKETS', 'BULLETS'],
      dtype='object')
```

[188]:
```
valores_columna = df_bombardeos_2['LAUNCH_COUNTRY']

valores_unicos = df_bombardeos_2['LAUNCH_COUNTRY'].unique()
valores_unicos
```

[188]:
```
array([nan, 'Okinawa', 'Japan', 'Philippines'], dtype=object)
```

[189]:
```
valores_columna = df_bombardeos_2['LAUNCH_BASE']

valores_unicos = df_bombardeos_2['LAUNCH_BASE'].unique()
valores_unicos
```

[189]:
```
array([nan, 'Kadena AFB', 'Yokota AFB', 'Clark AFB'], dtype=object)
```

[150]:
```
df_bombardeos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11052 entries, 0 to 11051
Data columns (total 32 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ROW_NUMBER                  11052 non-null  int64
 1   MISSION_NUMBER              11042 non-null  object
 2   OP_ORDER                    11043 non-null  object
 3   UNIT                        11040 non-null  object
 4   MISSION_DATE                11052 non-null  object
 5   AIRCRAFT_TYPE_MDS           10428 non-null  object
 6   NBR_ATTACK_EFFEC_AIRCRAFT   11012 non-null  float64
 7   SORTIE_DUPE                 6663 non-null   float64
 8   NBR_ABORT_AIRCRAFT          379 non-null    float64
 9   NBR_LOST_AIRCRAFT           32 non-null     object
 10  TARGET_NAME                 6036 non-null   object
 11  TGT_TYPE                    7758 non-null   object
 12  SOURCE_UTM_JAPAN_B          12 non-null     object
 13  SOURCE_TGT_UTM              8564 non-null   object
 14  TGT_MGRS                    7534 non-null   object
 15  TGT_LATITUDE_WGS84          7534 non-null   object
 16  TGT_LONGITUDE_WGS84         7534 non-null   object
 17  SOURCE_TGT_LAT              592 non-null    object
 18  SOURCE_TGT_LONG             585 non-null    object
 19  NBR_OF_WEAPONS              9956 non-null   object
 20  WEAPONS_TYPE                9958 non-null   object
 21  BOMB_SIGHTING_METHOD        8108 non-null   object
 22  TOTAL_BOMBLOAD_IN_LBS       5 non-null      float64
 23  TOT                         1559 non-null   object
 24  MISSION_TYPE                9925 non-null   object
 25  ALTITUDE_FT                 9219 non-null   object
 26  CALLSIGN                    1 non-null      object
 27  BDA                         6680 non-null   object
 28  NOSE_FUZE                   5208 non-null   object
 29  TAIL_FUZE                   4771 non-null   object
 30  CALCULATED_BOMBLOAD_LBS     9876 non-null   float64
 31  RECORD_SOURCE               11052 non-null  object
dtypes: float64(5), int64(1), object(26)
```

```
memory usage: 2.7+ MB
```

[127]:
```
df_bombardeos.columns
```

[127]:
```
Index(['ROW_NUMBER', 'MISSION_NUMBER', 'OP_ORDER', 'UNIT', 'MISSION_DATE',
       'AIRCRAFT_TYPE_MDS', 'NBR_ATTACK_EFFEC_AIRCRAFT', 'SORTIE_DUPE',
       'NBR_ABORT_AIRCRAFT', 'NBR_LOST_AIRCRAFT', 'TARGET_NAME', 'TGT_TYPE',
       'SOURCE_UTM_JAPAN_B', 'SOURCE_TGT_UTM', 'TGT_MGRS',
       'TGT_LATITUDE_WGS84', 'TGT_LONGITUDE_WGS84', 'SOURCE_TGT_LAT',
       'SOURCE_TGT_LONG', 'NBR_OF_WEAPONS', 'WEAPONS_TYPE',
       'BOMB_SIGHTING_METHOD', 'TOTAL_BOMBLOAD_IN_LBS', 'TOT', 'MISSION_TYPE',
       'ALTITUDE_FT', 'CALLSIGN', 'BDA', 'NOSE_FUZE', 'TAIL_FUZE',
       'CALCULATED_BOMBLOAD_LBS', 'RECORD_SOURCE'],
      dtype='object')
```

[128]:
```
df_bombardeos.head()
```

[128]:

	ROW_NUMBER	MISSION_NUMBER	OP_ORDER	UNIT	MISSION_DATE
0	2	433	174-51	98th Bomb Wing	6/1/51
1	3	433	174-51	307th Bomb Wing	6/1/51
2	4	433	174-51	307th Bomb Wing	6/1/51
3	5	433	174-51	98th Bomb Wing	6/1/51
4	6	433	174-51	98th Bomb Wing	6/1/51

	AIRCRAFT_TYPE_MDS	NBR_ATTACK_EFFEC_AIRCRAFT	SORTIE_DUPE
0	B-29	1.0	NaN
1	B-29	NaN	1.0
2	B-29	1.0	1.0
3	B-29	1.0	NaN
4	B-29	1.0	NaN

	NBR_ABORT_AIRCRAFT	NBR_LOST_AIRCRAFT	…	TOTAL_BOMBLOAD_IN_LBS	TOT
0	NaN	NaN	…	12000.0	NaN
1	NaN	NaN	…	NaN	NaN
2	NaN	NaN	…	NaN	NaN
3	NaN	NaN	…	16000.0	NaN
4	NaN	NaN	…	16000.0	NaN

	MISSION_TYPE	ALTITUDE_FT	CALLSIGN
0	NaN	19750	NaN
1	NaN	NaN	NaN
2	NaN	21000 - 22500	NaN
3	NaN	21500	NaN
4	NaN	16000	NaN

	BDA	NOSE_FUZE	TAIL_FUZE
0	Bombs fell on the east end of the tracks	0.01	Non-delay

```
1                                                   NaN      0.01  Non-delay
2  1 aircraft due to a bomb rack malfunction drop…      0.01  Non-delay
3                                                   NaN      0.01  Non-delay
4                                                   NaN      0.01  Non-delay

  CALCULATED_BOMBLOAD_LBS RECORD_SOURCE
0                 12000.0        EXETER
1                  4000.0        EXETER
2                  8000.0        EXETER
3                 16000.0        EXETER
4                 16000.0        EXETER

[5 rows x 32 columns]
```

[190]:
```
# EXTRAER DATOS DE DIA, MES Y AÑO DEL BOMBARDEO
df_bombardeos['MISSION_DATE'] = pd.to_datetime(df_bombardeos['MISSION_DATE'],␣
 ↪format='%m/%d/%y')

# Crear nuevas columnas para día, mes y año
df_bombardeos['DAY'] = df_bombardeos['MISSION_DATE'].dt.day
df_bombardeos['MONTH'] = df_bombardeos['MISSION_DATE'].dt.month
df_bombardeos['YEAR'] = df_bombardeos['MISSION_DATE'].dt.year
```

[191]:
```
# Corregir el año para reflejar el siglo XX
df_bombardeos['YEAR'] = df_bombardeos['YEAR'].apply(lambda x: x - 100 if x >␣
 ↪2000 else x)
```

[193]:
```
df_bombardeos.head()
```

[193]:
```
   ROW_NUMBER MISSION_NUMBER OP_ORDER             UNIT MISSION_DATE  \
0           2            433   174-51   98th Bomb Wing   2051-06-01
1           3            433   174-51  307th Bomb Wing   2051-06-01
2           4            433   174-51  307th Bomb Wing   2051-06-01
3           5            433   174-51   98th Bomb Wing   2051-06-01
4           6            433   174-51   98th Bomb Wing   2051-06-01

  AIRCRAFT_TYPE_MDS  NBR_ATTACK_EFFEC_AIRCRAFT  SORTIE_DUPE  \
0              B-29                        1.0          NaN
1              B-29                        NaN          1.0
2              B-29                        1.0          1.0
3              B-29                        1.0          NaN
4              B-29                        1.0          NaN

   NBR_ABORT_AIRCRAFT NBR_LOST_AIRCRAFT  …    ALTITUDE_FT CALLSIGN  \
0                 NaN               NaN  …          19750      NaN
1                 NaN               NaN  …            NaN      NaN
2                 NaN               NaN  …  21000 - 22500      NaN
```

```
3                 NaN               NaN  …          21500      NaN
4                 NaN               NaN  …          16000      NaN

                                                      BDA NOSE_FUZE  TAIL_FUZE  \
0           Bombs fell on the east end of the tracks      0.01  Non-delay
1                                                   NaN      0.01  Non-delay
2  1 aircraft due to a bomb rack malfunction drop…      0.01  Non-delay
3                                                   NaN      0.01  Non-delay
4                                                   NaN      0.01  Non-delay

  CALCULATED_BOMBLOAD_LBS RECORD_SOURCE DAY MONTH  YEAR
0                 12000.0        EXETER   1     6  1951
1                  4000.0        EXETER   1     6  1951
2                  8000.0        EXETER   1     6  1951
3                 16000.0        EXETER   1     6  1951
4                 16000.0        EXETER   1     6  1951

[5 rows x 35 columns]
```

[181]:
```
df_bombardeos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11052 entries, 0 to 11051
Data columns (total 35 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ROW_NUMBER                  11052 non-null  int64
 1   MISSION_NUMBER              11042 non-null  object
 2   OP_ORDER                    11043 non-null  object
 3   UNIT                        11040 non-null  object
 4   MISSION_DATE                11052 non-null  datetime64[ns]
 5   AIRCRAFT_TYPE_MDS           10428 non-null  object
 6   NBR_ATTACK_EFFEC_AIRCRAFT   11012 non-null  float64
 7   SORTIE_DUPE                 6663 non-null   float64
 8   NBR_ABORT_AIRCRAFT          379 non-null    float64
 9   NBR_LOST_AIRCRAFT           32 non-null     object
 10  TARGET_NAME                 6036 non-null   object
 11  TGT_TYPE                    7758 non-null   object
 12  SOURCE_UTM_JAPAN_B          12 non-null     object
 13  SOURCE_TGT_UTM              8564 non-null   object
 14  TGT_MGRS                    7534 non-null   object
 15  TGT_LATITUDE_WGS84          7534 non-null   object
 16  TGT_LONGITUDE_WGS84         7534 non-null   object
 17  SOURCE_TGT_LAT              592 non-null    object
 18  SOURCE_TGT_LONG             585 non-null    object
 19  NBR_OF_WEAPONS              9956 non-null   object
 20  WEAPONS_TYPE                9958 non-null   object
```

```
 21  BOMB_SIGHTING_METHOD       8108 non-null   object
 22  TOTAL_BOMBLOAD_IN_LBS      5 non-null      float64
 23  TOT                        1559 non-null   object
 24  MISSION_TYPE               9925 non-null   object
 25  ALTITUDE_FT                9219 non-null   object
 26  CALLSIGN                   1 non-null      object
 27  BDA                        6680 non-null   object
 28  NOSE_FUZE                  5208 non-null   object
 29  TAIL_FUZE                  4771 non-null   object
 30  CALCULATED_BOMBLOAD_LBS    9876 non-null   float64
 31  RECORD_SOURCE              11052 non-null  object
 32  DAY                        11052 non-null  int32
 33  MONTH                      11052 non-null  int32
 34  YEAR                       11052 non-null  int64
dtypes: datetime64[ns](1), float64(5), int32(2), int64(2), object(25)
memory usage: 2.9+ MB
```

[194]:
```
file_path = './data/df_bombardeos_actualizado.csv'
df_bombardeos = pd.read_csv(file_path, sep=',')
df_bombardeos.head()
```

[194]:

	ROW_NUMBER	MISSION_NUMBER	OP_ORDER	UNIT	AIRCRAFT_TYPE_MDS	\
0	2	433	174-51	98th Bomb Wing	B-29	
1	3	433	174-51	307th Bomb Wing	B-29	
2	4	433	174-51	307th Bomb Wing	B-29	
3	5	433	174-51	98th Bomb Wing	B-29	
4	6	433	174-51	98th Bomb Wing	B-29	

	NBR_ATTACK_EFFEC_AIRCRAFT	SORTIE_DUPE	NBR_ABORT_AIRCRAFT	\
0	1.0	NaN	NaN	
1	NaN	1.0	NaN	
2	1.0	1.0	NaN	
3	1.0	NaN	NaN	
4	1.0	NaN	NaN	

	NBR_LOST_AIRCRAFT	TARGET_NAME	…	ALTITUDE_FT	CALLSIGN	\
0	NaN	Changdo-ri	…	19750	NaN	
1	NaN	NaN	…	NaN	NaN	
2	NaN	NaN	…	21000 - 22500	NaN	
3	NaN	Anju	…	21500	NaN	
4	NaN	Hamhung	…	16000	NaN	

	BDA	NOSE_FUZE	TAIL_FUZE	\
0	Bombs fell on the east end of the tracks	0.01	Non-delay	
1	NaN	0.01	Non-delay	
2	1 aircraft due to a bomb rack malfunction drop…	0.01	Non-delay	
3	NaN	0.01	Non-delay	

```
4                                                    NaN      0.01  Non-delay

  CALCULATED_BOMBLOAD_LBS RECORD_SOURCE DAY MONTH  YEAR
0                 12000.0        EXETER   1     6  1951
1                  4000.0        EXETER   1     6  1951
2                  8000.0        EXETER   1     6  1951
3                 16000.0        EXETER   1     6  1951
4                 16000.0        EXETER   1     6  1951

[5 rows x 34 columns]
```

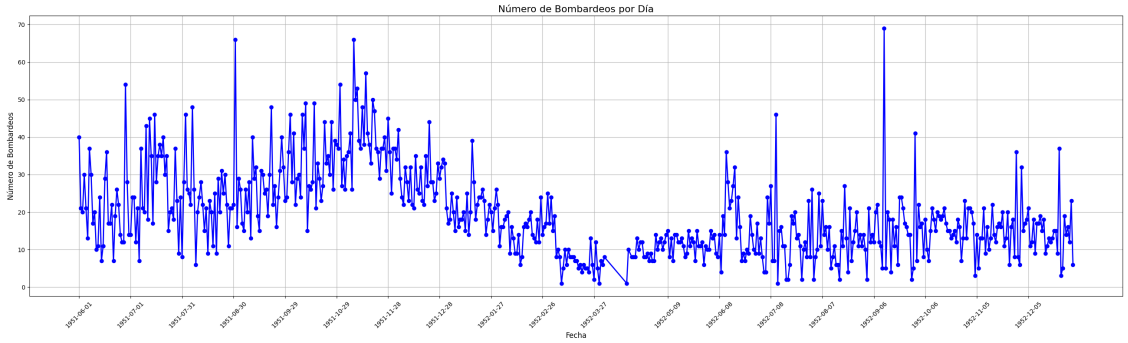
[195]:
```
df_bombardeos['MISSION_DATE'] = pd.to_datetime(
    df_bombardeos[['YEAR', 'MONTH', 'DAY']],
    errors='coerce')

bombing_counts_by_date = df_bombardeos.groupby('MISSION_DATE').size()

plt.figure(figsize=(26, 8))
plt.plot(
    bombing_counts_by_date.index,
    bombing_counts_by_date.values,
    marker='o', linestyle='-', linewidth=2, color='blue'
)

# Configurar los ticks para que se muestren todos los meses
plt.xticks(bombing_counts_by_date.index[::30], rotation=45)  # Mostrar una␣
 ↪etiqueta aproximadamente cada mes

# Configurar título y etiquetas de ejes
plt.title("Número de Bombardeos por Día", fontsize=16)
plt.xlabel("Fecha", fontsize=12)
plt.ylabel("Número de Bombardeos", fontsize=12)
plt.grid(True)
plt.tight_layout();
```



[155]:
```
valores_columna = df_bombardeos['TGT_LATITUDE_WGS84']

valores_unicos = df_bombardeos['TGT_LATITUDE_WGS84'].unique()
valores_unicos.tolist()
```

[155]:
```
['38.49881N',
 nan,
 '39.66879N',
 '39.60215N',
 '39.91217N',
 '38.24362N',
 '52.89048N',
 '40.77223N',
 '39.65877N',
 '38.16582N',
 '38.20858N',
 '38.20318N',
 '38.25461N',
 '38.20173N',
 '38.15706N',
 '38.18240N',
 '38.29953N',
 '38.25448N',
 '38.22268N',
 '38.16118N',
 '36.66510N',
 '38.29038N',
 '40.11406N',
 '39.68349N',
 '39.70554N',
 '39.53857N',
 '39.52939N',
 '40.72265N',
 '38.01361N',
 '38.08702N',
 '38.17484N',
 '38.07934N',
 '38.18409N',
 '38.22616N',
 '38.13879N',
 '38.27193N',
 '38.20136N',
 '38.18385N',
 '39.67750N',
 '40.79618N',
 '38.45245N',
 '38.50132N',
```

```
'39.84011N',
'39.54758N',
'38.09821N',
'38.10910N',
'38.20017N',
'38.39237N',
'38.20837N',
'38.26555N',
'38.26321N',
'38.57547N',
'38.12569N',
'38.17545N',
'38.28164N',
'38.51032N',
'39.53840N',
'38.09396N',
'38.19176N',
'38.20248N',
'38.25920N',
'38.19723N',
'38.17445N',
'38.21062N',
'38.21037N',
'38.30812N',
'38.19347N',
'38.17782N',
'38.21377N',
'38.13422N',
'38.20134N',
'39.90272N',
'38.73373N',
'38.23718N',
'38.11448N',
'38.20212N',
'38.21113N',
'38.27675N',
'38.22049N',
'37.98682N',
'37.92558N',
'38.13898N',
'38.37577N',
'38.20957N',
'38.20977N',
'38.31279N',
'38.21125N',
'38.28623N',
'38.31308N',
```

```
'38.88885N',
'38.37762N',
'38.39481N',
'38.42150N',
'38.34977N',
'38.39547N',
'38.33992N',
'38.36728N',
'38.32190N',
'38.26751N',
'38.54568N',
'38.34076N',
'38.22556N',
'38.30710N',
'38.36812N',
'38.35107N',
'38.36924N',
'38.39563N',
'38.24403N',
'38.27754N',
'38.26608N',
'38.25652N',
'38.26870N',
'38.43134N',
'38.42133N',
'38.37729N',
'38.27397N',
'38.11262N',
'40.44359N',
'38.28428N',
'38.76497N',
'38.36008N',
'38.41712N',
'38.27581N',
'38.22281N',
'38.20229N',
'38.30587N',
'38.29621N',
'38.33091N',
'38.32173N',
'38.21481N',
'38.33777N',
'38.19853N',
'38.27686N',
'38.28410N',
'38.36828N',
'38.29572N',
```

```
'38.31148N',
'38.26892N',
'38.37635N',
'38.50105N',
'38.72496N',
'38.40805N',
'38.44475N',
'38.29763N',
'38.36724N',
'38.28923N',
'38.12326N',
'38.25911N',
'38.28785N',
'38.26497N',
'38.29701N',
'38.29748N',
'38.24825N',
'38.00460N',
'38.02447N',
'38.34488N',
'38.89947N',
'38.33238N',
'38.28694N',
'38.25019N',
'38.24971N',
'38.30368N',
'38.26456N',
'38.42580N',
'38.51005N',
'38.38017N',
'38.37436N',
'38.34751N',
'38.35652N',
'38.40792N',
'38.37228N',
'39.96485N',
'38.43723N',
'38.25872N',
'38.26783N',
'38.19440N',
'38.50078N',
'38.41210N',
'38.33446N',
'38.48736N',
'38.40265N',
'38.09180N',
'38.28113N',
```

```
'38.32047N',
'38.26667N',
'38.26764N',
'38.30252N',
'38.39710N',
'38.39068N',
'38.42184N',
'38.43952N',
'38.50756N',
'39.81265N',
'38.39229N',
'38.32933N',
'38.31108N',
'38.31131N',
'38.50769N',
'38.50782N',
'38.36314N',
'38.39003N',
'38.33542N',
'38.37472N',
'38.35478N',
'38.46588N',
'38.41679N',
'38.27593N',
'38.62358N',
'38.40046N',
'38.36287N',
'38.40858N',
'38.73827N',
'39.01412N',
'40.02442N',
'39.41014N',
'39.01246N',
'38.40130N',
'38.36571N',
'38.25863N',
'39.43032N',
'39.18731N',
'38.36300N',
'38.38047N',
'38.69635N',
'38.28566N',
'38.29515N',
'38.36837N',
'38.46211N',
'38.40778N',
'38.59270N',
```

```
'38.49994N',
'38.39891N',
'38.48967N',
'38.35050N',
'38.46264N',
'38.42593N',
'38.48900N',
'40.03549N',
'40.22907N',
'38.51734N',
'38.43085N',
'38.38019N',
'38.49031N',
'38.40270N',
'38.41759N',
'38.30407N',
'38.29525N',
'38.41943N',
'38.43077N',
'38.49019N',
'38.26692N',
'38.48914N',
'38.38615N',
'38.42904N',
'38.03347N',
'38.49907N',
'38.39970N',
'38.22587N',
'38.52496N',
'39.40759N',
'39.31018N',
'38.43801N',
'38.38572N',
'39.83209N',
'39.53275N',
'38.64426N',
'38.41665N',
'38.49117N',
'38.37092N',
'38.33183N',
'38.47766N',
'38.49746N',
'39.29870N',
'38.32440N',
'38.32183N',
'38.36549N',
'40.02452N',
```

```
'38.57555N',
'38.55261N',
'38.47999N',
'38.40605N',
'38.05909N',
'38.08942N',
'38.37932N',
'38.37856N',
'38.25237N',
'38.25258N',
'38.24358N',
'38.28002N',
'38.30904N',
'38.30426N',
'38.25901N',
'38.21656N',
'38.29823N',
'38.42194N',
'38.29467N',
'38.00574N',
'38.40409N',
'38.47735N',
'38.44702N',
'38.30338N',
'40.02463N',
'38.41049N',
'38.39611N',
'38.17009N',
'38.63766N',
'39.15357N',
'38.63688N',
'38.27655N',
'38.46328N',
'38.38742N',
'38.64453N',
'38.38155N',
'39.83181N',
'38.15483N',
'38.81500N',
'38.52584N',
'38.80193N',
'38.44019N',
'39.82281N',
'38.26812N',
'38.38892N',
'38.02420N',
'37.92263N',
```

```
'38.41022N',
'38.48886N',
'38.33156N',
'39.83195N',
'38.24580N',
'38.40354N',
'38.23498N',
'38.60171N',
'38.61982N',
'38.46251N',
'38.73790N',
'38.38206N',
'38.38033N',
'38.39562N',
'38.24030N',
'38.46170N',
'38.45269N',
'38.47350N',
'38.29761N',
'38.28840N',
'38.22759N',
'38.39056N',
'38.29740N',
'39.11472N',
'39.41542N',
'40.17170N',
'39.19631N',
'38.28642N',
'38.30416N',
'38.33110N',
'38.29944N',
'38.30865N',
'39.53249N',
'38.41811N',
'38.25421N',
'38.28556N',
'38.30023N',
'38.46238N',
'38.69621N',
'38.41594N',
'38.37146N',
'38.16314N',
'38.18095N',
'38.29884N',
'38.27142N',
'38.28651N',
'38.41824N',
```

```
'38.34922N',
'38.16334N',
'38.23956N',
'38.28882N',
'38.75942N',
'38.69008N',
'38.40310N',
'38.32801N',
'38.33714N',
'38.35453N',
'38.36340N',
'38.19977N',
'38.31269N',
'38.34002N',
'38.34347N',
'38.19957N',
'38.32723N',
'38.34538N',
'38.41977N',
'38.35775N',
'38.29925N',
'38.24952N',
'38.50051N',
'38.37992N',
'38.19056N',
'38.22598N',
'38.47071N',
'38.29552N',
'38.34940N',
'38.30359N',
'39.31465N',
'38.28063N',
'38.28123N',
'38.25947N',
'38.23478N',
'38.41019N',
'38.24460N',
'38.15433N',
'38.15454N',
'38.39043N',
'38.34525N',
'38.24676N',
'38.44147N',
'38.26322N',
'38.43151N',
'38.09027N',
'38.34039N',
```

```
'38.49894N',
'38.34950N',
'38.40984N',
'38.25938N',
'38.02261N',
'38.93099N',
'38.43522N',
'39.33897N',
'39.69585N',
'38.17153N',
'38.12731N',
'38.11810N',
'38.46902N',
'39.32312N',
'39.41195N',
'38.38934N',
'38.56663N',
'38.63886N',
'38.39863N',
'38.29905N',
'38.49583N',
'39.59447N',
'38.27750N',
'38.38075N',
'38.41772N',
'38.41706N',
'38.24619N',
'38.45964N',
'38.38142N',
'38.27713N',
'38.50024N',
'38.25037N',
'38.37241N',
'38.24070N',
'38.24090N',
'38.25010N',
'38.24981N',
'38.41885N',
'38.39957N',
'38.30894N',
'38.26848N',
'38.53499N',
'38.49953N',
'38.36778N',
'38.27758N',
'38.43996N',
'38.54014N',
```

```
'38.26802N',
'38.26839N',
'38.99694N',
'39.62212N',
'39.77476N',
'38.50702N',
'38.40071N',
'38.41693N',
'38.27731N',
'38.18294N',
'38.31038N',
'38.43494N',
'38.40923N',
'36.53758N',
'38.25361N',
'39.25216N',
'38.40764N',
'38.44340N',
'38.38102N',
'38.40819N',
'38.65129N',
'38.32281N',
'38.28984N',
'39.61988N',
'39.14203N',
'39.08695N',
'38.35199N',
'38.69676N',
'38.45973N',
'38.74284N',
'38.50795N',
'39.43049N',
'38.52318N',
'39.63842N',
'38.52297N',
'39.82267N',
'39.39830N',
'38.49843N',
'38.31863N',
'38.49163N',
'38.18808N',
'39.31438N',
'39.78578N',
'38.33071N',
'38.29719N',
'38.24378N',
'39.88471N',
```

```
'38.49868N',
'38.48980N',
'39.15390N',
'39.01494N',
'38.32357N',
'39.16336N',
'38.34186N',
'39.42114N',
'38.45296N',
'38.45310N',
'38.27694N',
'38.79586N',
'38.75996N',
'39.00540N',
'39.13526N',
'38.45044N',
'38.79626N',
'38.76883N',
'38.26713N',
'38.28595N',
'39.40643N',
'38.25832N',
'39.21257N',
'39.82308N',
'38.42840N',
'38.19136N',
'38.24911N',
'38.24209N',
'39.63973N',
'39.15374N',
'42.23786N',
'38.29487N',
'38.35554N',
'38.44193N',
'38.26681N',
'39.39859N',
'39.22180N',
'39.27170N',
'39.14504N',
'38.24991N',
'38.25892N',
'38.36526N',
'39.65953N',
'38.31874N',
'38.27722N',
'38.26733N',
'38.28576N',
```

```
'38.27665N',
'38.43904N',
'38.31862N',
'38.37368N',
'38.32821N',
'38.30453N',
'38.16126N',
'38.24274N',
'39.31556N',
'39.43313N',
'38.34603N',
'38.35466N',
'38.31363N',
'38.25882N',
'38.33740N',
'38.28183N',
'38.32802N',
'38.30462N',
'39.64874N',
'38.39996N',
'39.62147N',
'38.41798N',
'38.61015N',
'39.14473N',
'39.68371N',
'38.36354N',
'38.37625N',
'38.30578N',
'38.40009N',
'38.36203N',
'38.09047N',
'38.03745N',
'38.69909N',
'38.30935N',
'38.13487N',
'38.18870N',
'38.50496N',
'38.09800N',
'38.74942N',
'38.61916N',
'38.33814N',
'38.14303N',
'38.25801N',
'39.15737N',
'40.14133N',
'39.37556N',
'38.26571N',
```

```
'38.44462N',
'38.22758N',
'38.47836N',
'39.14489N',
'38.63001N',
'38.28163N',
'38.28354N',
'38.33664N',
'38.27603N',
'38.71125N',
'38.45413N',
'38.31882N',
'38.46142N',
'38.38606N',
'38.11665N',
'38.46230N',
'38.51906N',
'38.39507N',
'38.34733N',
'38.15182N',
'38.15474N',
'38.63902N',
'39.22469N',
'38.33120N',
'38.26575N',
'38.22118N',
'38.42621N',
'38.27416N',
'38.27262N',
'38.26401N',
'38.25480N',
'38.34903N',
'39.02839N',
'38.88874N',
'39.38529N',
'37.92286N',
'37.91872N',
'37.46648N',
'38.81131N',
'38.72875N',
'39.17813N',
'38.35002N',
'38.40392N',
'38.43711N',
'38.42185N',
'38.43979N',
'39.34008N',
```

```
'38.29237N',
'38.26516N',
'38.36169N',
'38.25542N',
'38.75843N',
'38.97894N',
'38.19156N',
'38.14633N',
'38.50821N',
'38.78039N',
'39.61247N',
'39.74885N',
'38.32200N',
'38.33101N',
'38.37380N',
'38.34159N',
'38.45622N',
'38.50744N',
'38.40736N',
'38.33343N',
'39.73999N',
'38.26659N',
'38.30378N',
'38.38536N',
'38.43095N',
'38.21799N',
'38.25634N',
'38.45837N',
'38.43050N',
'38.38050N',
'38.22515N',
'38.44162N',
'38.24695N',
'39.44183N',
'38.06154N',
'38.12437N',
'38.08187N',
'38.14367N',
'38.04585N',
'38.41955N',
'39.60373N',
'39.68685N',
'39.73357N',
'38.10764N',
'38.04396N',
'39.25658N',
'39.69509N',
```

```
'38.46699N',
'38.31259N',
'38.33900N',
'38.42453N',
'38.44646N',
'38.33129N',
'40.05278N',
'39.42413N',
'39.21233N',
'38.30358N',
'38.25843N',
'38.26744N',
'38.30348N',
'38.27635N',
'38.32246N',
'38.48943N',
'38.98767N',
'38.72927N',
'38.17214N',
'38.78688N',
'38.16354N',
'38.29161N',
'38.39821N',
'38.27741N',
'39.69897N',
'39.43283N',
'39.49597N',
'38.38526N',
'37.20715N',
'38.32415N',
'38.09596N',
'38.49568N',
'39.90287N',
'38.75740N',
'36.36997N',
'38.66281N',
'39.92073N',
'38.40610N',
'38.35741N',
'38.45073N',
'38.19945N',
'38.46997N',
'38.78363N',
'39.91173N',
'39.84745N',
'38.85156N',
'39.42383N',
```

```
'38.48716N',
'38.07225N',
'38.48695N',
'38.66254N',
'39.01217N',
'39.03185N',
'39.48671N',
'38.42266N',
'38.13692N',
'38.01779N',
'39.00512N',
'38.83831N',
'38.79264N',
'38.48018N',
'38.41365N',
'38.42931N',
'38.05121N',
'39.00346N',
'38.79755N',
'38.41895N',
'38.33973N',
'38.40480N',
'38.40496N',
'38.38373N',
'38.93411N',
'39.19684N',
'38.20918N',
'38.10679N',
'38.25744N',
'38.09757N',
'38.37567N',
'39.27806N',
'38.61833N',
'39.52375N',
'39.74955N',
'38.32525N',
'38.30620N',
'38.15331N',
'38.12371N',
'38.33467N',
'38.27101N',
'38.78061N',
'38.24066N',
'38.34060N',
'39.55881N',
'38.52721N',
'38.41119N',
```

```
'38.32119N',
'38.36686N',
'38.16396N',
'39.08481N',
'38.53806N',
'38.41988N',
'38.17355N',
'38.17315N',
'38.20817N',
'38.21758N',
'38.56406N',
'38.25406N',
'38.09349N',
'39.48619N',
'39.74984N',
'38.12629N',
'38.00879N',
'38.48335N',
'38.42747N',
'38.74883N',
'38.57495N',
'38.40218N',
'38.48475N',
'38.28482N',
'38.22365N',
'38.18371N',
'38.14573N',
'38.74597N',
'38.48251N',
'38.47958N',
'38.25630N',
'38.32850N',
'38.54319N',
'38.71451N',
'38.36461N',
'38.22639N',
'39.00195N',
'38.40399N',
'38.55662N',
'38.40382N',
'38.30382N',
'37.85462N',
'38.26390N',
'38.34443N',
'38.78995N',
'38.24900N',
'38.80004N',
```

```
'38.79976N',
'37.97937N',
'38.44368N',
'38.42064N',
'38.45685N',
'38.56512N',
'38.85617N',
'39.15419N',
'38.50671N',
'39.41248N',
'39.16275N',
'39.69288N',
'38.10914N',
'38.56123N',
'38.53970N',
'37.99231N',
'37.98867N',
'37.96395N',
'38.52239N',
'38.43514N',
'40.00989N',
'40.24344N',
'40.44361N',
'41.08105N',
'41.20927N',
'40.96568N',
'39.89386N',
'41.78294N',
'39.83153N',
'39.70485N',
'38.25279N',
'38.28777N',
'38.65720N',
'39.34797N',
'39.24814N',
'39.69556N',
'39.41572N',
'39.59473N',
'39.69272N',
'39.79716N',
'41.40791N',
'39.01467N',
'39.80279N',
'39.65564N',
'38.65704N',
'38.22915N',
'38.13063N',
```

```
'38.18555N',
'37.83017N',
'38.34464N',
'39.14497N',
'38.86518N',
'39.54723N',
'38.49178N',
'38.74753N',
'39.92703N',
'37.67231N',
'36.78516N',
'39.82238N',
'40.95918N',
'38.03298N',
'38.21354N',
'38.44449N',
'38.24420N',
'38.03373N',
'38.41859N',
'38.50695N',
'38.22577N',
'38.38355N',
'38.30337N',
'38.38426N',
'38.45770N',
'38.65340N',
'40.17163N',
'34.50205N',
'40.65969N',
'37.94950N',
'38.79885N',
'38.47886N',
'38.04641N',
'38.66401N',
'38.97631N',
'38.34343N',
'38.52207N',
'38.39617N',
'39.90333N',
'38.08921N',
'38.42099N',
'38.31317N',
'39.58610N',
'38.33138N',
'38.35842N',
'38.36145N',
'39.14574N',
```

```
'38.88919N',
'39.72126N',
'40.19477N',
'40.15933N',
'38.43935N',
'40.53406N',
'39.14347N',
'38.48523N',
'38.57964N',
'38.48066N',
'38.43273N',
'38.05156N',
'38.42798N',
'38.41950N',
'38.79728N',
'38.21955N',
'38.52709N',
'40.52541N',
'41.57712N',
'39.91803N',
'39.98960N',
'38.35944N',
'38.35878N',
'38.38547N',
'38.38614N',
'38.35227N',
'40.59299N',
'39.92725N',
'40.51436N',
'39.48771N',
'38.48429N',
'38.48273N',
'38.52123N',
'38.39133N',
'38.55858N',
'38.39807N',
'38.49440N',
'38.56314N',
'38.54513N',
'38.31920N',
'40.12701N',
'41.71175N',
'38.39642N',
'38.79015N',
'38.39291N',
'38.48284N',
'38.43874N',
```

```
 '39.75061N',
 '40.21671N',
 '38.21448N',
 '38.37060N',
 '38.24932N',
 '38.23113N',
 '38.24949N',
 '39.89433N',
 '38.14862N',
 '38.60665N',
 '38.33975N',
 '38.81333N',
 '38.47213N',
 '38.49155N',
 '39.14457N',
 '38.18880N',
 '38.33165N',
 '38.42822N',
 …]
```

[156]:
```
valores_columna = df_bombardeos['TGT_LONGITUDE_WGS84']

valores_unicos = df_bombardeos['TGT_LONGITUDE_WGS84'].unique()
valores_unicos.tolist()
```

[156]:
```
[' 127.66974E',
 nan,
 ' 125.50653E',
 ' 125.66717E',
 ' 127.56091E',
 ' 126.41753E',
 ' 126.55293E',
 ' 125.61128E',
 ' 125.55279E',
 ' 127.71006E',
 ' 126.92125E',
 ' 127.83503E',
 ' 126.97708E',
 ' 127.04706E',
 ' 127.73304E',
 ' 127.58418E',
 ' 127.58190E',
 ' 127.58278E',
 ' 127.99466E',
 ' 128.20096E',
 ' 128.22789E',
 ' 127.57065E',
```

```
' 126.26600E',
' 125.22724E',
' 125.00633E',
' 127.25447E',
' 127.24307E',
' 126.48985E',
' 126.61924E',
' 126.68527E',
' 127.70990E',
' 126.75391E',
' 127.73257E',
' 127.48051E',
' 127.71054E',
' 127.53671E',
' 127.66375E',
' 127.70974E',
' 125.06382E',
' 126.55820E',
' 125.89926E',
' 125.74071E',
' 127.56241E',
' 127.25425E',
' 126.79895E',
' 126.90126E',
' 126.95575E',
' 127.13354E',
' 126.90984E',
' 127.76545E',
' 127.55975E',
' 127.32395E',
' 127.37992E',
' 127.09342E',
' 127.59368E',
' 125.74105E',
' 127.24284E',
' 126.58243E',
' 126.99024E',
' 127.09272E',
' 128.05130E',
' 128.20057E',
' 127.67566E',
' 127.68643E',
' 127.66359E',
' 127.54742E',
' 127.09295E',
' 128.01811E',
' 128.00620E',
```

```
' 128.21267E',
' 127.02423E',
' 127.52601E',
' 125.41599E',
' 127.01184E',
' 126.70721E',
' 127.73226E',
' 127.73211E',
' 127.99391E',
' 127.09225E',
' 126.63150E',
' 126.72447E',
' 126.58095E',
' 127.98109E',
' 126.97833E',
' 126.98974E',
' 127.99341E',
' 127.74353E',
' 128.05095E',
' 128.02772E',
' 127.91615E',
' 127.35142E',
' 127.29378E',
' 127.27024E',
' 127.29484E',
' 127.33957E',
' 127.23786E',
' 127.26009E',
' 127.23830E',
' 127.21675E',
' 127.14106E',
' 127.29505E',
' 127.43483E',
' 127.46739E',
' 127.31730E',
' 127.38636E',
' 127.39740E',
' 127.35101E',
' 127.46871E',
' 127.28510E',
' 127.12535E',
' 127.09131E',
' 127.29674E',
' 127.32729E',
' 127.25879E',
' 127.32853E',
' 127.05656E',
```

```
' 126.20547E',
' 128.88208E',
' 127.13631E',
' 127.34254E',
' 127.38615E',
' 126.99546E',
' 127.17082E',
' 127.24069E',
' 127.08130E',
' 127.37592E',
' 127.33040E',
' 127.23808E',
' 127.22686E',
' 127.30941E',
' 127.10062E',
' 127.43541E',
' 127.23939E',
' 127.12488E',
' 127.32874E',
' 127.29610E',
' 127.14705E',
' 128.14263E',
' 128.04977E',
' 125.75216E',
' 125.40419E',
' 127.61414E',
' 127.67073E',
' 127.43328E',
' 128.03845E',
' 126.89608E',
' 127.96180E',
' 128.03987E',
' 127.37632E',
' 127.05680E',
' 127.38755E',
' 127.42185E',
' 127.13723E',
' 125.79042E',
' 125.71144E',
' 125.48310E',
' 125.23711E',
' 126.35718E',
' 128.14242E',
' 128.05142E',
' 127.99428E',
' 127.98210E',
' 127.67403E',
```

```
' 127.59089E',
' 125.75250E',
' 126.48136E',
' 127.13400E',
' 127.14613E',
' 127.14590E',
' 127.60269E',
' 127.63772E',
' 127.45443E',
' 125.78410E',
' 127.99416E',
' 128.00547E',
' 126.64762E',
' 125.76362E',
' 125.70298E',
' 126.90622E',
' 126.89033E',
' 126.69814E',
' 126.47989E',
' 126.48479E',
' 126.64355E',
' 126.65672E',
' 127.98261E',
' 127.85632E',
' 126.88151E',
' 127.03054E',
' 127.29314E',
' 127.26981E',
' 127.64664E',
' 127.52794E',
' 127.82055E',
' 127.83301E',
' 127.81043E',
' 127.83330E',
' 127.65811E',
' 127.66958E',
' 127.62645E',
' 127.61449E',
' 127.56976E',
' 127.86663E',
' 127.68383E',
' 127.97983E',
' 127.59107E',
' 127.90246E',
' 127.55257E',
' 127.74026E',
' 127.60356E',
```

```
' 127.65994E',
' 125.62321E',
' 125.77195E',
' 127.94527E',
' 125.78758E',
' 125.84117E',
' 127.82041E',
' 127.86677E',
' 127.98273E',
' 127.24556E',
' 125.68614E',
' 127.61500E',
' 127.56888E',
' 127.60860E',
' 127.98235E',
' 128.03940E',
' 128.18724E',
' 127.61311E',
' 127.59124E',
' 128.05841E',
' 127.77293E',
' 127.60287E',
' 127.65844E',
' 128.21033E',
' 127.65894E',
' 127.60235E',
' 127.60113E',
' 128.20296E',
' 126.17911E',
' 127.71528E',
' 127.29293E',
' 127.54598E',
' 127.71576E',
' 127.96926E',
' 127.65978E',
' 128.02784E',
' 128.05083E',
' 127.83157E',
' 128.09489E',
' 127.70430E',
' 127.90260E',
' 127.61259E',
' 128.15269E',
' 126.25058E',
' 125.71178E',
' 127.69267E',
' 127.67156E',
```

```
' 126.42958E',
' 125.48927E',
' 125.89199E',
' 125.42410E',
' 127.88856E',
' 128.09545E',
' 127.64438E',
' 125.54817E',
' 125.79210E',
' 127.57962E',
' 127.79601E',
' 126.47023E',
' 128.11900E',
' 127.41793E',
' 125.51126E',
' 125.53966E',
' 126.40323E',
' 126.71213E',
' 126.64209E',
' 127.95699E',
' 128.17343E',
' 127.64580E',
' 127.60130E',
' 127.45385E',
' 126.64056E',
' 126.81061E',
' 127.47731E',
' 127.42009E',
' 126.85144E',
' 126.86286E',
' 126.86312E',
' 126.88492E',
' 126.99844E',
' 126.73556E',
' 126.72553E',
' 126.86391E',
' 126.89582E',
' 128.11791E',
' 127.98223E',
' 126.67644E',
' 128.14103E',
' 127.39501E',
' 127.88842E',
' 127.94780E',
' 127.96871E',
' 126.22838E',
' 127.38535E',
```

```
' 127.33328E',
' 128.04633E',
' 126.03745E',
' 127.95443E',
' 127.97105E',
' 127.71624E',
' 127.40845E',
' 125.78062E',
' 127.66044E',
' 127.62102E',
' 126.47772E',
' 125.40722E',
' 127.66924E',
' 126.84654E',
' 127.31563E',
' 127.62120E',
' 128.03975E',
' 126.46961E',
' 125.72282E',
' 125.42332E',
' 126.21694E',
' 127.58966E',
' 128.08468E',
' 127.63270E',
' 126.98875E',
' 126.74391E',
' 126.88622E',
' 128.05829E',
' 128.06954E',
' 127.64749E',
' 128.19458E',
' 127.05367E',
' 127.55743E',
' 126.80141E',
' 127.94871E',
' 127.57873E',
' 127.57891E',
' 127.83069E',
' 126.86154E',
' 126.85038E',
' 126.97783E',
' 127.66028E',
' 126.85011E',
' 125.93877E',
' 128.37754E',
' 125.68648E',
' 128.07382E',
```

```
' 128.03928E',
' 128.02748E',
' 126.96440E',
' 126.97558E',
' 125.55979E',
' 127.70559E',
' 126.95423E',
' 127.97092E',
' 127.01012E',
' 127.63603E',
' 127.59710E',
' 127.52236E',
' 127.56905E',
' 126.89972E',
' 126.88779E',
' 126.93011E',
' 126.90803E',
' 128.08525E',
' 127.71704E',
' 128.03868E',
' 126.91112E',
' 127.86873E',
' 126.87323E',
' 127.60737E',
' 125.75939E',
' 125.70264E',
' 127.70719E',
' 127.71847E',
' 127.66094E',
' 127.64933E',
' 126.93292E',
' 127.98198E',
' 128.01592E',
' 126.90596E',
' 126.92151E',
' 127.63856E',
' 127.64967E',
' 127.86593E',
' 127.98135E',
' 126.95297E',
' 127.97143E',
' 125.77507E',
' 126.46992E',
' 126.92176E',
' 126.88649E',
' 127.57856E',
' 128.08514E',
```

```
' 128.06157E',
' 126.70127E',
' 125.63301E',
' 126.91920E',
' 126.95348E',
' 128.08559E',
' 126.87481E',
' 127.80881E',
' 126.92022E',
' 126.91138E',
' 126.92279E',
' 127.64883E',
' 127.63823E',
' 127.04586E',
' 127.40726E',
' 126.95398E',
' 127.33874E',
' 127.42631E',
' 128.06169E',
' 127.68121E',
' 128.07301E',
' 127.77445E',
' 128.07416E',
' 125.38105E',
' 127.63863E',
' 127.62508E',
' 126.25006E',
' 125.89233E',
' 126.86523E',
' 126.91215E',
' 126.90101E',
' 126.04278E',
' 125.65653E',
' 127.22279E',
' 127.55725E',
' 127.33563E',
' 127.34545E',
' 127.57997E',
' 126.94154E',
' 127.42900E',
' 125.60865E',
' 128.08536E',
' 127.59177E',
' 127.67123E',
' 127.61397E',
' 127.01159E',
' 127.41832E',
```

```
' 127.64900E',
' 128.03964E',
' 125.78652E',
' 128.07427E',
' 127.64916E',
' 127.99441E',
' 128.01726E',
' 128.03999E',
' 128.00571E',
' 127.77430E',
' 127.66011E',
' 127.61603E',
' 128.08548E',
' 127.03808E',
' 127.07344E',
' 128.10712E',
' 128.09679E',
' 128.11769E',
' 126.81996E',
' 128.02833E',
' 128.07405E',
' 125.73664E',
' 127.42713E',
' 127.38863E',
' 127.60078E',
' 127.76316E',
' 127.60252E',
' 128.06250E',
' 127.00190E',
' 127.07846E',
' 127.60217E',
' 127.71720E',
' 126.08295E',
' 125.08003E',
' 125.60750E',
' 127.57979E',
' 126.69864E',
' 127.55617E',
' 127.61466E',
' 127.62559E',
' 126.98895E',
' 128.11911E',
' 126.93037E',
' 125.67950E',
' 125.69598E',
' 125.74012E',
' 127.45500E',
```

```
' 127.64308E',
' 126.03170E',
' 124.26574E',
' 128.02845E',
' 127.68104E',
' 127.25718E',
' 126.87782E',
' 125.65691E',
' 126.86635E',
' 127.60951E',
' 125.90322E',
' 126.53457E',
' 127.03249E',
' 127.84187E',
' 126.35099E',
' 125.64460E',
' 127.54019E',
' 127.98172E',
' 126.83868E',
' 126.87454E',
' 127.52640E',
' 127.65828E',
' 127.66991E',
' 127.44918E',
' 125.73734E',
' 126.80358E',
' 127.48370E',
' 128.26764E',
' 127.23418E',
' 127.60182E',
' 127.61328E',
' 128.01677E',
' 127.64121E',
' 127.65340E',
' 125.76006E',
' 127.40331E',
' 126.02062E',
' 127.67574E',
' 127.64172E',
' 127.92546E',
' 128.01665E',
' 125.93839E',
' 127.94845E',
' 126.64875E',
' 127.64456E',
' 127.13261E',
' 126.96741E',
```

```
' 127.92572E',
' 126.78319E',
' 126.20958E',
' 125.59870E',
' 127.43761E',
' 130.29683E',
' 128.00510E',
' 127.75248E',
' 127.44162E',
' 127.89117E',
' 125.89162E',
' 126.66002E',
' 126.46132E',
' 127.46095E',
' 128.01714E',
' 128.01702E',
' 127.82099E',
' 125.51786E',
' 127.68448E',
' 128.05107E',
' 127.94832E',
' 127.99379E',
' 127.98248E',
' 128.00312E',
' 127.67304E',
' 127.76362E',
' 127.06655E',
' 128.08502E',
' 126.79705E',
' 126.81745E',
' 127.41082E',
' 125.95113E',
' 127.70687E',
' 127.67238E',
' 128.09635E',
' 128.00559E',
' 127.07775E',
' 126.98775E',
' 127.05511E',
' 128.09646E',
' 125.59904E',
' 127.69446E',
' 125.60967E',
' 127.69413E',
' 127.98927E',
' 127.43781E',
' 125.21558E',
```

```
' 127.66078E',
' 128.03833E',
' 126.81556E',
' 127.70591E',
' 127.53490E',
' 126.86759E',
' 126.92610E',
' 125.75974E',
' 126.54099E',
' 126.83207E',
' 126.81906E',
' 126.86689E',
' 126.78755E',
' 124.47298E',
' 127.98915E',
' 127.12349E',
' 126.78619E',
' 127.91416E',
' 126.59273E',
' 126.27664E',
' 126.68972E',
' 127.10250E',
' 127.65928E',
' 127.59473E',
' 126.89060E',
' 127.44938E',
' 127.35714E',
' 126.97633E',
' 127.09060E',
' 127.03200E',
' 127.91389E',
' 127.36679E',
' 125.83058E',
' 127.04392E',
' 127.55582E',
' 128.14125E',
' 126.82121E',
' 125.86528E',
' 125.75285E',
' 128.14114E',
' 127.13470E',
' 126.77451E',
' 126.93419E',
' 127.35694E',
' 127.35495E',
' 128.03892E',
' 126.61103E',
```

```
' 126.63534E',
' 127.62525E',
' 127.06798E',
' 126.97658E',
' 126.99968E',
' 126.98850E',
' 128.01580E',
' 127.49814E',
' 127.90463E',
' 127.24669E',
' 125.41195E',
' 125.16156E',
' 125.25023E',
' 125.57982E',
' 125.64586E',
' 127.25185E',
' 128.14167E',
' 128.11813E',
' 127.79691E',
' 128.10646E',
' 128.09478E',
' 127.23622E',
' 127.07893E',
' 127.06822E',
' 126.91688E',
' 127.66276E',
' 124.47317E',
' 125.73595E',
' 126.97882E',
' 126.96866E',
' 127.70397E',
' 126.66301E',
' 125.60933E',
' 127.47094E',
' 128.01617E',
' 128.01604E',
' 127.09968E',
' 126.80304E',
' 126.67352E',
' 126.53426E',
' 127.55689E',
' 126.84905E',
' 127.48280E',
' 127.86831E',
' 127.99354E',
' 128.04966E',
' 128.11780E',
```

```
' 126.94383E',
' 127.07988E',
' 127.32666E',
' 128.06052E',
' 126.96213E',
' 126.84082E',
' 128.00460E',
' 127.41871E',
' 127.05728E',
' 125.96311E',
' 126.69553E',
' 126.76585E',
' 126.75253E',
' 126.89064E',
' 128.28104E',
' 126.89168E',
' 127.84302E',
' 125.59735E',
' 125.89196E',
' 125.82390E',
' 126.82147E',
' 126.78918E',
' 126.19534E',
' 125.08767E',
' 128.11736E',
' 127.97054E',
' 127.90151E',
' 126.90362E',
' 127.83113E',
' 128.05036E',
' 127.30004E',
' 125.95075E',
' 126.63717E',
' 127.97067E',
' 127.95987E',
' 127.95975E',
' 127.95923E',
' 127.94819E',
' 128.07336E',
' 126.53488E',
' 125.74783E',
' 125.62288E',
' 126.89946E',
' 125.85521E',
' 126.92253E',
' 127.03322E',
' 127.54562E',
```

```
' 128.07393E',
' 125.76419E',
' 125.96273E',
' 125.58171E',
' 128.03821E',
' 125.14109E',
' 126.39180E',
' 126.26311E',
' 127.41753E',
' 127.53771E',
' 128.44756E',
' 128.82164E',
' 125.76984E',
' 127.52563E',
' 126.88125E',
' 127.20310E',
' 126.03207E',
' 128.65737E',
' 126.92523E',
' 127.38816E',
' 127.52582E',
' 127.43369E',
' 125.78867E',
' 125.96235E',
' 126.87888E',
' 126.85672E',
' 126.86742E',
' 125.78132E',
' 125.85271E',
' 125.78419E',
' 125.59299E',
' 127.35040E',
' 126.94610E',
' 126.83553E',
' 125.77160E',
' 127.43301E',
' 127.38796E',
' 126.52373E',
' 127.35060E',
' 126.26202E',
' 125.72383E',
' 125.84081E',
' 125.78656E',
' 125.79485E',
' 127.98160E',
' 127.36226E',
' 127.37371E',
```

```
' 127.86649E',
' 127.93854E',
' 125.66334E',
' 126.95550E',
' 126.77588E',
' 127.14843E',
' 126.76476E',
' 127.96965E',
' 127.30733E',
' 127.89726E',
' 125.54784E',
' 125.90625E',
' 126.89504E',
' 126.83842E',
' 126.85435E',
' 126.71833E',
' 126.91765E',
' 126.88518E',
' 126.67452E',
' 127.22883E',
' 127.28361E',
' 126.03191E',
' 127.79541E',
' 127.91187E',
' 127.92466E',
' 127.99266E',
' 127.55029E',
' 127.69349E',
' 127.99028E',
' 127.87738E',
' 126.97932E',
' 126.95650E',
' 126.89842E',
' 126.92099E',
' 127.16354E',
' 127.54850E',
' 127.68853E',
' 125.61622E',
' 125.89459E',
' 126.85514E',
' 126.83579E',
' 127.18852E',
' 127.07536E',
' 127.48100E',
' 128.09308E',
' 127.91201E',
' 128.08275E',
```

```
' 127.17060E',
' 127.29779E',
' 127.04754E',
' 126.93445E',
' 126.31899E',
' 127.83055E',
' 127.56692E',
' 127.74275E',
' 127.75294E',
' 127.61155E',
' 127.61975E',
' 127.08847E',
' 126.90932E',
' 127.54489E',
' 127.30502E',
' 127.26701E',
' 127.29357E',
' 126.71270E',
' 126.78348E',
' 127.61689E',
' 127.56958E',
' 127.20381E',
' 127.91430E',
' 128.06720E',
' 128.03266E',
' 127.99798E',
' 127.57909E',
' 127.21299E',
' 127.22356E',
' 127.23237E',
' 127.42110E',
' 126.89370E',
' 125.50011E',
' 127.25762E',
' 127.43741E',
' 126.00881E',
' 128.60076E',
' 126.99146E',
' 126.36108E',
' 128.71528E',
' 128.03202E',
' 128.37387E',
' 127.39401E',
' 126.99496E',
' 127.45341E',
' 128.31810E',
' 128.90567E',
```

```
' 128.28570E',
' 128.84493E',
' 129.33276E',
' 127.53790E',
' 129.81830E',
' 127.59765E',
' 125.89271E',
' 126.87428E',
' 126.81610E',
' 127.36802E',
' 126.24971E',
' 126.21885E',
' 125.90398E',
' 125.92717E',
' 125.59702E',
' 125.21587E',
' 124.91556E',
' 126.52346E',
' 125.74888E',
' 125.55811E',
' 126.38010E',
' 127.35653E',
' 127.73179E',
' 127.79055E',
' 127.16166E',
' 127.52276E',
' 126.30980E',
' 125.49453E',
' 126.88241E',
' 127.42090E',
' 127.23098E',
' 125.76327E',
' 125.61204E',
' 125.21171E',
' 126.67555E',
' 125.91360E',
' 127.58615E',
' 126.59972E',
' 126.68697E',
' 126.70411E',
' 127.64782E',
' 125.10262E',
' 125.70039E',
' 126.59455E',
' 126.51135E',
' 126.87507E',
' 127.14521E',
```

```
' 126.68984E',
' 127.19099E',
' 127.28084E',
' 126.62119E',
' 128.36579E',
' 129.62090E',
' 129.20111E',
' 125.83385E',
' 126.68544E',
' 127.50961E',
' 127.51842E',
' 127.96553E',
' 127.66085E',
' 127.48951E',
' 127.37107E',
' 125.61079E',
' 125.49167E',
' 126.79922E',
' 127.23589E',
' 128.03916E',
' 125.95761E',
' 128.06180E',
' 128.06145E',
' 127.48913E',
' 126.08406E',
' 126.27903E',
' 125.57843E',
' 125.74899E',
' 126.27592E',
' 127.25836E',
' 127.58311E',
' 125.19854E',
' 127.31458E',
' 127.64529E',
' 127.65861E',
' 126.03281E',
' 128.00840E',
' 127.78560E',
' 126.22803E',
' 125.79806E',
' 127.03516E',
' 127.78394E',
' 125.46736E',
' 129.59978E',
' 125.21142E',
' 125.23715E',
' 127.34039E',
```

```
 ' 127.29463E',
 ' 127.27110E',
 ' 127.31688E',
 ' 126.89426E',
 ' 125.24533E',
 ' 125.20002E',
 ' 125.93904E',
 ' 125.93018E',
 ' 128.02543E',
 ' 127.85347E',
 ' 128.13964E',
 ' 127.72896E',
 ' 127.40468E',
 ' 127.53417E',
 ' 127.32583E',
 ' 127.10618E',
 ' 127.10665E',
 ' 127.06679E',
 ' 128.44837E',
 ' 129.68517E',
 ' 127.40825E',
 ' 127.96371E',
 ' 127.16787E',
 ' 127.86494E',
 ' 127.96875E',
 ' 127.61096E',
 ' 124.57478E',
 ' 127.28658E',
 ' 127.50039E',
 ' 127.20577E',
 ' 127.19479E',
 ' 127.20532E',
 ' 127.21719E',
 ' 125.49135E',
 ' 127.10553E',
 ' 127.01316E',
 ' 127.22642E',
 ' 127.59481E',
 …]
```

[ ]:
```
# Paso 1: Convertir la columna a valores numéricos eliminando el sufijo 'N' y␣
 ↪'E'
df_bombardeos['TGT_LATITUDE_WGS84'] = df_bombardeos['TGT_LATITUDE_WGS84'].str.
 ↪rstrip('N')
df_bombardeos['TGT_LATITUDE_WGS84'] = pd.
 ↪to_numeric(df_bombardeos['TGT_LATITUDE_WGS84'], errors='coerce')
```

```
df_bombardeos['TGT_LONGITUDE_WGS84'] = df_bombardeos['TGT_LONGITUDE_WGS84'].str.
 ↪rstrip('E')
df_bombardeos['TGT_LONGITUDE_WGS84'] = pd.
 ↪to_numeric(df_bombardeos['TGT_LONGITUDE_WGS84'], errors='coerce')

# Paso 2: Crear una copia para trabajar
df_bombardeos_filled = df_bombardeos.copy()

# Filtrar los valores válidos
valid_coords = df_bombardeos_filled.dropna(subset=['TGT_LATITUDE_WGS84',␣
 ↪'TGT_LONGITUDE_WGS84'])

# Paso 3: Crear una función para rellenar los valores faltantes
def fill_missing_coordinates(row):
    if pd.isna(row['TGT_LATITUDE_WGS84']) or pd.
 ↪isna(row['TGT_LONGITUDE_WGS84']):
        # Filtrar valores dentro de un rango cercano (en días)
        nearby_points = valid_coords[
            (valid_coords['YEAR'] == row['YEAR']) &  # Mismo año
            (valid_coords['MONTH'] == row['MONTH']) &  # Mismo mes
            (abs(valid_coords['DAY'] - row['DAY']) <= 3)  # Rango de 3 días
        ]

        # Calcular la media de latitud y longitud si hay puntos cercanos
        if not nearby_points.empty:
            lat_mean = nearby_points['TGT_LATITUDE_WGS84'].mean()
            lon_mean = nearby_points['TGT_LONGITUDE_WGS84'].mean()
            return lat_mean, lon_mean
    # Si no hay puntos cercanos, devolver los valores originales
    return row['TGT_LATITUDE_WGS84'], row['TGT_LONGITUDE_WGS84']

# Paso 4: Aplicar la función para rellenar los valores faltantes
df_bombardeos_filled['TGT_LATITUDE_WGS84'],␣
 ↪df_bombardeos_filled['TGT_LONGITUDE_WGS84'] = zip(
    *df_bombardeos_filled.apply(fill_missing_coordinates, axis=1)
)

# Paso 5: Volver a agregar los sufijos 'N' para latitud y 'E' para longitud
df_bombardeos_filled['TGT_LATITUDE_WGS84'] =␣
 ↪df_bombardeos_filled['TGT_LATITUDE_WGS84'].apply(
    lambda x: f"{x}N" if not pd.isna(x) else x
)
df_bombardeos_filled['TGT_LONGITUDE_WGS84'] =␣
 ↪df_bombardeos_filled['TGT_LONGITUDE_WGS84'].apply(
    lambda x: f"{x}E" if not pd.isna(x) else x  # Cambiado a 'E' para longitud
)
```

[197]:
```
df_bombardeos_filled.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11052 entries, 0 to 11051
Data columns (total 35 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ROW_NUMBER                  11052 non-null  int64
 1   MISSION_NUMBER              11042 non-null  object
 2   OP_ORDER                    11043 non-null  object
 3   UNIT                        11040 non-null  object
 4   AIRCRAFT_TYPE_MDS           10428 non-null  object
 5   NBR_ATTACK_EFFEC_AIRCRAFT   11012 non-null  float64
 6   SORTIE_DUPE                 6663 non-null   float64
 7   NBR_ABORT_AIRCRAFT          379 non-null    float64
 8   NBR_LOST_AIRCRAFT           32 non-null     object
 9   TARGET_NAME                 6036 non-null   object
 10  TGT_TYPE                    7758 non-null   object
 11  SOURCE_UTM_JAPAN_B          12 non-null     object
 12  SOURCE_TGT_UTM              8564 non-null   object
 13  TGT_MGRS                    7534 non-null   object
 14  TGT_LATITUDE_WGS84          11052 non-null  object
 15  TGT_LONGITUDE_WGS84         11052 non-null  object
 16  SOURCE_TGT_LAT              592 non-null    object
 17  SOURCE_TGT_LONG             585 non-null    object
 18  NBR_OF_WEAPONS              9956 non-null   object
 19  WEAPONS_TYPE                9958 non-null   object
 20  BOMB_SIGHTING_METHOD        8108 non-null   object
 21  TOTAL_BOMBLOAD_IN_LBS       5 non-null      float64
 22  TOT                         1559 non-null   object
 23  MISSION_TYPE                9925 non-null   object
 24  ALTITUDE_FT                 9219 non-null   object
 25  CALLSIGN                    1 non-null      object
 26  BDA                         6680 non-null   object
 27  NOSE_FUZE                   5208 non-null   object
 28  TAIL_FUZE                   4771 non-null   object
 29  CALCULATED_BOMBLOAD_LBS     9876 non-null   float64
 30  RECORD_SOURCE               11052 non-null  object
 31  DAY                         11052 non-null  int64
 32  MONTH                       11052 non-null  int64
 33  YEAR                        11052 non-null  int64
 34  MISSION_DATE                11052 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(5), int64(4), object(25)
memory usage: 3.0+ MB
```

[160]:
```
# atencion
```

[142]:
```
# Convertir latitud y longitud a formato numérico eliminando sufijos
df_bombardeos_ubicaciones_validas['TGT_LATITUDE_WGS84'] = pd.
 ↪to_numeric(df_bombardeos_ubicaciones_validas['TGT_LATITUDE_WGS84'].
 ↪astype(str).str.rstrip('N'), errors='coerce')
df_bombardeos_ubicaciones_validas['TGT_LONGITUDE_WGS84'] = pd.
 ↪to_numeric(df_bombardeos_ubicaciones_validas['TGT_LONGITUDE_WGS84'].
 ↪astype(str).str.rstrip('E'), errors='coerce')

# Filtrar filas con valores NaN en latitud y longitud
df_bombardeos_ubicaciones_validas = df_bombardeos_ubicaciones_validas.
 ↪dropna(subset=['TGT_LATITUDE_WGS84', 'TGT_LONGITUDE_WGS84'])

# Calcular el centro del mapa basado en el promedio de las coordenadas
centro_lat = df_bombardeos_ubicaciones_validas['TGT_LATITUDE_WGS84'].mean()
centro_lon = df_bombardeos_ubicaciones_validas['TGT_LONGITUDE_WGS84'].mean()

# Crear un mapa centrado
mapa_bombardeos = folium.Map(location=[centro_lat, centro_lon], zoom_start=6)

# Agregar marcadores para cada ubicación
for _, row in df_bombardeos_ubicaciones_validas.iterrows():
    folium.Marker(
        location=[row['TGT_LATITUDE_WGS84'], row['TGT_LONGITUDE_WGS84']],
        popup=f"Fecha: {row['YEAR']}-{row['MONTH']}-{row['DAY']}<br>Unidad:␣
 ↪{row.get('UNIT', 'N/A')}",
        icon=folium.Icon(color='blue', icon='info-sign')
    ).add_to(mapa_bombardeos)

# Guardar el mapa como archivo HTML
map_path = 'mapa_bombardeos.html'
mapa_bombardeos.save(map_path)

# Mostrar enlace para descargar el mapa interactivo
map_path
```

[142]:
```
'mapa_bombardeos.html'
```

[77]:
```
# # Configurar el geolocalizador con un agente de usuario único
# geolocator = Nominatim(user_agent="my_python_script_v1")

# # Caché para almacenar resultados ya calculados
# location_cache = {}

# # Función para obtener el nombre del lugar
# def get_location_name_with_cache(lat, lon):
#     key = (lat, lon)
#     if key in location_cache:
```

```
#         return location_cache[key]
#     try:
#         location = geolocator.reverse((lat, lon), language='en', timeout=10)
#         location_name = location.raw.get('address', {}).get('town') or \
#                         location.raw.get('address', {}).get('city') or \
#                         location.raw.get('address', {}).get('village') or␣
 ↪"Unknown"
#         location_cache[key] = location_name
#         return location_name
#     except (GeocoderTimedOut, GeocoderServiceError):  # Manejar errores
#         return "Unknown"

# # Aplicar geocodificación inversa con un retraso entre solicitudes
# def apply_geocoding(row):
#     if not pd.isna(row['TGT_LATITUDE_WGS84']) and not pd.
 ↪isna(row['TGT_LONGITUDE_WGS84']):
#         time.sleep(2)  # Aumentar el retraso a 2 segundos
#         return get_location_name_with_cache(row['TGT_LATITUDE_WGS84'],␣
 ↪row['TGT_LONGITUDE_WGS84'])
#     else:
#         return "Unknown"

# # Aplicar la función al DataFrame
# df_bombardeos['City/Town'] = df_bombardeos.apply(apply_geocoding, axis=1)

# # Guardar los resultados en un archivo CSV
# df_bombardeos.to_csv("df_bombardeos_actualizado.csv", index=False)
```

[70]:
```
# PRUEBA CON LOS PRIMEROS 100 REGISTROS

# Configurar el geolocalizador con un agente de usuario único
geolocator = Nominatim(user_agent="my_python_script_v1")

# Caché para almacenar resultados ya calculados
location_cache = {}

# Función para obtener el nombre del lugar con caché
def get_location_name_with_cache(lat, lon):
    key = (lat, lon)
    if key in location_cache:
        return location_cache[key]
    try:
        location = geolocator.reverse((lat, lon), language='en', timeout=10)
        location_name = location.raw.get('address', {}).get('town') or \
                        location.raw.get('address', {}).get('city') or \
                        location.raw.get('address', {}).get('village') or␣
 ↪"Unknown"
```

```
        location_cache[key] = location_name
        return location_name
    except (GeocoderTimedOut, GeocoderServiceError):  # Manejar errores
        return "Unknown"

# Función para aplicar geocodificación inversa con retraso entre solicitudes
def apply_geocoding(row):
    if not pd.isna(row['TGT_LATITUDE_WGS84']) and not pd.
 ↪isna(row['TGT_LONGITUDE_WGS84']):
        time.sleep(2)  # Retraso de 2 segundos para respetar límites de la API
        return get_location_name_with_cache(row['TGT_LATITUDE_WGS84'],␣
 ↪row['TGT_LONGITUDE_WGS84'])
    else:
        return "Unknown"


# Seleccionar las primeras 100 filas para prueba
df_bombardeos_sample = df_bombardeos.head(100).copy()  # Usar .copy() para␣
 ↪evitar SettingWithCopyWarning

# Aplicar la función de geocodificación
df_bombardeos_sample.loc[:, 'City/Town'] = df_bombardeos_sample.
 ↪apply(apply_geocoding, axis=1)

# Guardar los resultados en un archivo CSV
df_bombardeos_sample.to_csv("df_bombardeos_sample_actualizado.csv", index=False)

# Mostrar un fragmento del DataFrame actualizado
print(df_bombardeos_sample[['TGT_LATITUDE_WGS84', 'TGT_LONGITUDE_WGS84', 'City/
 ↪Town']].head())
```

```
   TGT_LATITUDE_WGS84  TGT_LONGITUDE_WGS84   City/Town
0            38.49881            127.66974  Changdo-ri
1                 NaN                  NaN     Unknown
2            39.66879            125.50653
3            39.60215            125.66717     Anju-si
4            39.91217            127.56091  Hamhung-si
```

[159]:
```
file_path = './data/df_bombardeos_sample_actualizado.csv'
df_bombardeos_ubicaciones = pd.read_csv(file_path, sep=',')
df_bombardeos_ubicaciones.head()
```

[159]:

	ROW_NUMBER	MISSION_NUMBER	OP_ORDER	UNIT	AIRCRAFT_TYPE_MDS	\
0	2	433	174-51	98th Bomb Wing	B-29	
1	3	433	174-51	307th Bomb Wing	B-29	
2	4	433	174-51	307th Bomb Wing	B-29	
3	5	433	174-51	98th Bomb Wing	B-29	

```
4           6             433   174-51   98th Bomb Wing              B-29

   NBR_ATTACK_EFFEC_AIRCRAFT  SORTIE_DUPE  NBR_ABORT_AIRCRAFT  \
0                        1.0          NaN                 NaN
1                        NaN          1.0                 NaN
2                        1.0          1.0                 NaN
3                        1.0          NaN                 NaN
4                        1.0          NaN                 NaN

   NBR_LOST_AIRCRAFT TARGET_NAME  …  \
0                NaN  Changdo-ri  …
1                NaN         NaN  …
2                NaN         NaN  …
3                NaN        Anju  …
4                NaN     Hamhung  …

                                              BDA  NOSE_FUZE  TAIL_FUZE  \
0          Bombs fell on the east end of the tracks       0.01  Non-delay
1                                              NaN       0.01  Non-delay
2  1 aircraft due to a bomb rack malfunction drop…       0.01  Non-delay
3                                              NaN       0.01  Non-delay
4                                              NaN       0.01  Non-delay

  CALCULATED_BOMBLOAD_LBS  RECORD_SOURCE  DAY MONTH  YEAR  MISSION_DATE  \
0                 12000.0         EXETER    1     6  1951    1951-06-01
1                  4000.0         EXETER    1     6  1951    1951-06-01
2                  8000.0         EXETER    1     6  1951    1951-06-01
3                 16000.0         EXETER    1     6  1951    1951-06-01
4                 16000.0         EXETER    1     6  1951    1951-06-01

    City/Town
0  Changdo-ri
1     Unknown
2
3     Anju-si
4  Hamhung-si

[5 rows x 36 columns]
```

[79]:
```
# # lo hacemos con todos los datos, los 11000 registros

# # Configurar el geolocalizador con un agente de usuario único
# geolocator = Nominatim(user_agent="my_python_script_v1")

# # Caché para almacenar resultados ya calculados
# location_cache = {}
```

```
# # Función para obtener el nombre del lugar con caché
# def get_location_name_with_cache(lat, lon):
#     key = (lat, lon)
#     if key in location_cache:
#         return location_cache[key]
#     try:
#         location = geolocator.reverse((lat, lon), language='en', timeout=10)
#         location_name = location.raw.get('address', {}).get('town') or \
#                         location.raw.get('address', {}).get('city') or \
#                         location.raw.get('address', {}).get('village') or␣
 ↪"Unknown"
#         location_cache[key] = location_name
#         return location_name
#     except (GeocoderTimedOut, GeocoderServiceError):  # Manejar errores
#         return "Unknown"

# # Función para aplicar geocodificación inversa con retraso entre solicitudes
# def apply_geocoding(row):
#     if not pd.isna(row['TGT_LATITUDE_WGS84']) and not pd.
 ↪isna(row['TGT_LONGITUDE_WGS84']):
#         time.sleep(2)  # Retraso de 2 segundos para respetar límites de la API
#         return get_location_name_with_cache(row['TGT_LATITUDE_WGS84'],␣
 ↪row['TGT_LONGITUDE_WGS84'])
#     else:
#         return "Unknown"


# # Procesar por lotes para guardar progreso parcial
# batch_size = 100  # Tamaño del lote
# output_file = "df_bombardeos_actualizado_completo.csv"

# # Verificar si existe un archivo de salida para continuar
# try:
#     processed_df = pd.read_csv(output_file)
#     start_idx = len(processed_df)  # Continuar desde donde quedó
#     print(f"Reanudando desde el registro {start_idx}")
# except FileNotFoundError:
#     processed_df = pd.DataFrame()
#     start_idx = 0

# # Procesar en lotes
# for i in range(start_idx, len(df_bombardeos), batch_size):
#     batch = df_bombardeos.iloc[i:i + batch_size].copy()  # Crear un lote
#     print(f"Procesando registros {i} a {i + len(batch) - 1}")
#     batch.loc[:, 'City/Town'] = batch.apply(apply_geocoding, axis=1)  #␣
 ↪Aplicar geocodificación
```

```
#     processed_df = pd.concat([processed_df, batch], ignore_index=True)  #␣
 ↪Agregar al conjunto procesado

#     # Guardar resultados parciales en un archivo CSV
#     processed_df.to_csv(output_file, index=False)

# print("Geocodificación completada para todos los registros.")
```

[202]:
```
# Verificar si las columnas contienen valores de tipo cadena
if df_bombardeos_filled['TGT_LATITUDE_WGS84'].dtype == 'object':
    df_bombardeos_filled['TGT_LATITUDE_WGS84'] =␣
 ↪df_bombardeos_filled['TGT_LATITUDE_WGS84'].str.rstrip('N')
    df_bombardeos_filled['TGT_LATITUDE_WGS84'] = pd.
 ↪to_numeric(df_bombardeos_filled['TGT_LATITUDE_WGS84'], errors='coerce')

if df_bombardeos_filled['TGT_LONGITUDE_WGS84'].dtype == 'object':
    df_bombardeos_filled['TGT_LONGITUDE_WGS84'] =␣
 ↪df_bombardeos_filled['TGT_LONGITUDE_WGS84'].str.rstrip('E')
    df_bombardeos_filled['TGT_LONGITUDE_WGS84'] = pd.
 ↪to_numeric(df_bombardeos_filled['TGT_LONGITUDE_WGS84'], errors='coerce')
```

[203]:
```
print(df_bombardeos_filled['TGT_LATITUDE_WGS84'].head())
print(df_bombardeos_filled['TGT_LATITUDE_WGS84'].dtype)
print(df_bombardeos_filled['TGT_LONGITUDE_WGS84'].head())
print(df_bombardeos_filled['TGT_LONGITUDE_WGS84'].dtype)
```

```
0    38.498810
1    38.935944
2    39.668790
3    39.602150
4    39.912170
Name: TGT_LATITUDE_WGS84, dtype: float64
float64
0    127.669740
1    126.781969
2    125.506530
3    125.667170
4    127.560910
Name: TGT_LONGITUDE_WGS84, dtype: float64
float64
```

[204]:
```
print(df_bombardeos_filled[['TGT_LATITUDE_WGS84', 'TGT_LONGITUDE_WGS84']].
 ↪describe())
```

```
       TGT_LATITUDE_WGS84  TGT_LONGITUDE_WGS84
count        11052.000000         11052.000000
mean            39.126479           126.686361
```

```
std              0.703574            0.928440
min             34.502050          121.278270
25%             38.628055          125.939040
50%             39.122121          126.700727
75%             39.594470          127.307850
max             52.890480          130.692550
```

[215]:
```python
import time
from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut, GeocoderServiceError
import pandas as pd

# Configurar el geolocalizador con un agente de usuario único
geolocator = Nominatim(user_agent="my_python_script_v1")

# Caché para almacenar resultados ya calculados
location_cache = {}

# Función para obtener el nombre del lugar con caché
def get_location_name_with_cache(lat, lon):
    key = (lat, lon)
    if key in location_cache:
        return location_cache[key]
    try:
        if pd.notna(lat) and pd.notna(lon):  # Ensure coordinates are valid
            location = geolocator.reverse((lat, lon), language='en', timeout=10)
            if location is not None:
                location_name = location.raw.get('address', {}).get('town') or \
                                location.raw.get('address', {}).get('city') or \
                                location.raw.get('address', {}).get('village')␣
 ↪or "Unknown"
                location_cache[key] = location_name
                return location_name
        return "Unknown"
    except (GeocoderTimedOut, GeocoderServiceError):  # Manejar errores
        return "Unknown"

# Función para aplicar geocodificación inversa con retraso entre solicitudes
def apply_geocoding(row):
    if not pd.isna(row['TGT_LATITUDE_WGS84']) and not pd.
 ↪isna(row['TGT_LONGITUDE_WGS84']):
        time.sleep(2)  # Retraso de 2 segundos para respetar límites de la API
        return get_location_name_with_cache(row['TGT_LATITUDE_WGS84'],␣
 ↪row['TGT_LONGITUDE_WGS84'])
    else:
        return "Unknown"
```

```python
# Procesar por lotes para guardar progreso parcial
batch_size = 100  # Tamaño del lote
output_file = "df_bombardeos_actualizado_completo.csv"

# Verificar si existe un archivo de salida para continuar
try:
    processed_df = pd.read_csv(output_file)
    start_idx = len(processed_df)  # Continuar desde donde quedó
    print(f"Reanudando desde el registro {start_idx}")
except FileNotFoundError:
    processed_df = pd.DataFrame()
    start_idx = 0

# Procesar en lotes
for i in range(start_idx, len(df_bombardeos_filled), batch_size):
    batch = df_bombardeos_filled.iloc[i:i + batch_size].copy()  # Crear un lote
    print(f"Procesando registros {i} a {i + len(batch) - 1}")
    batch.loc[:, 'City/Town'] = batch.apply(apply_geocoding, axis=1)  # Aplicar␣
 ↪geocodificación
    processed_df = pd.concat([processed_df, batch], ignore_index=True)  #␣
 ↪Agregar al conjunto procesado

    # Guardar resultados parciales en un archivo CSV
    processed_df.to_csv(output_file, index=False)

print("Geocodificación completada para todos los registros.")
```

```
Procesando registros 0 a 99
Procesando registros 100 a 199
Procesando registros 200 a 299
Procesando registros 300 a 399
Procesando registros 400 a 499
Procesando registros 500 a 599
Procesando registros 600 a 699
Procesando registros 700 a 799
Procesando registros 800 a 899
Procesando registros 900 a 999
Procesando registros 1000 a 1099
Procesando registros 1100 a 1199
Procesando registros 1200 a 1299
Procesando registros 1300 a 1399
Procesando registros 1400 a 1499
Procesando registros 1500 a 1599
Procesando registros 1600 a 1699
Procesando registros 1700 a 1799
Procesando registros 1800 a 1899
```

```
Procesando registros 1900 a 1999
Procesando registros 2000 a 2099
Procesando registros 2100 a 2199
Procesando registros 2200 a 2299
Procesando registros 2300 a 2399
Procesando registros 2400 a 2499
Procesando registros 2500 a 2599
Procesando registros 2600 a 2699
Procesando registros 2700 a 2799
Procesando registros 2800 a 2899
Procesando registros 2900 a 2999
Procesando registros 3000 a 3099
Procesando registros 3100 a 3199
Procesando registros 3200 a 3299
Procesando registros 3300 a 3399
Procesando registros 3400 a 3499
Procesando registros 3500 a 3599
Procesando registros 3600 a 3699
Procesando registros 3700 a 3799
Procesando registros 3800 a 3899
Procesando registros 3900 a 3999
Procesando registros 4000 a 4099
Procesando registros 4100 a 4199
Procesando registros 4200 a 4299
Procesando registros 4300 a 4399
Procesando registros 4400 a 4499
Procesando registros 4500 a 4599
Procesando registros 4600 a 4699
Procesando registros 4700 a 4799
Procesando registros 4800 a 4899
Procesando registros 4900 a 4999
Procesando registros 5000 a 5099
Procesando registros 5100 a 5199
Procesando registros 5200 a 5299
Procesando registros 5300 a 5399
Procesando registros 5400 a 5499
Procesando registros 5500 a 5599
Procesando registros 5600 a 5699
Procesando registros 5700 a 5799
Procesando registros 5800 a 5899
Procesando registros 5900 a 5999
Procesando registros 6000 a 6099
Procesando registros 6100 a 6199
Procesando registros 6200 a 6299
Procesando registros 6300 a 6399
Procesando registros 6400 a 6499
Procesando registros 6500 a 6599
Procesando registros 6600 a 6699
```

```
Procesando registros 6700 a 6799
Procesando registros 6800 a 6899
Procesando registros 6900 a 6999
Procesando registros 7000 a 7099
Procesando registros 7100 a 7199
Procesando registros 7200 a 7299
Procesando registros 7300 a 7399
Procesando registros 7400 a 7499
Procesando registros 7500 a 7599
Procesando registros 7600 a 7699
Procesando registros 7700 a 7799
Procesando registros 7800 a 7899
Procesando registros 7900 a 7999
Procesando registros 8000 a 8099
Procesando registros 8100 a 8199
Procesando registros 8200 a 8299
Procesando registros 8300 a 8399
Procesando registros 8400 a 8499
Procesando registros 8500 a 8599
Procesando registros 8600 a 8699
Procesando registros 8700 a 8799
Procesando registros 8800 a 8899
Procesando registros 8900 a 8999
Procesando registros 9000 a 9099
Procesando registros 9100 a 9199
Procesando registros 9200 a 9299
Procesando registros 9300 a 9399
Procesando registros 9400 a 9499
Procesando registros 9500 a 9599
Procesando registros 9600 a 9699
Procesando registros 9700 a 9799
Procesando registros 9800 a 9899
Procesando registros 9900 a 9999
Procesando registros 10000 a 10099
Procesando registros 10100 a 10199
Procesando registros 10200 a 10299
Procesando registros 10300 a 10399
Procesando registros 10400 a 10499
Procesando registros 10500 a 10599
Procesando registros 10600 a 10699
Procesando registros 10700 a 10799
Procesando registros 10800 a 10899
Procesando registros 10900 a 10999
Procesando registros 11000 a 11051
Geocodificación completada para todos los registros.
```

```
[ ]: file_path = './data/df_bombardeos_actualizado_completo.csv'
     df_bombardeos_ubicaciones_validas = pd.read_csv(file_path, sep=',')
     df_bombardeos_ubicaciones_validas.head()
```

```
[ ]:    ROW_NUMBER MISSION_NUMBER OP_ORDER             UNIT AIRCRAFT_TYPE_MDS  \
     0           2            433   174-51   98th Bomb Wing              B-29
     1           3            433   174-51  307th Bomb Wing              B-29
     2           4            433   174-51  307th Bomb Wing              B-29
     3           5            433   174-51   98th Bomb Wing              B-29
     4           6            433   174-51   98th Bomb Wing              B-29

        NBR_ATTACK_EFFEC_AIRCRAFT  SORTIE_DUPE  NBR_ABORT_AIRCRAFT  \
     0                        1.0          NaN                 NaN
     1                        NaN          1.0                 NaN
     2                        1.0          1.0                 NaN
     3                        1.0          NaN                 NaN
     4                        1.0          NaN                 NaN

       NBR_LOST_AIRCRAFT TARGET_NAME  …  \
     0               NaN  Changdo-ri  …
     1               NaN         NaN  …
     2               NaN         NaN  …
     3               NaN        Anju  …
     4               NaN     Hamhung  …

                                                 BDA NOSE_FUZE  TAIL_FUZE  \
     0          Bombs fell on the east end of the tracks      0.01  Non-delay
     1                                              NaN      0.01  Non-delay
     2  1 aircraft due to a bomb rack malfunction drop…      0.01  Non-delay
     3                                              NaN      0.01  Non-delay
     4                                              NaN      0.01  Non-delay

       CALCULATED_BOMBLOAD_LBS  RECORD_SOURCE  DAY MONTH  YEAR MISSION_DATE  \
     0                 12000.0         EXETER    1     6  1951   1951-06-01
     1                  4000.0         EXETER    1     6  1951   1951-06-01
     2                  8000.0         EXETER    1     6  1951   1951-06-01
     3                 16000.0         EXETER    1     6  1951   1951-06-01
     4                 16000.0         EXETER    1     6  1951   1951-06-01

         City/Town
     0  Changdo-ri
     1     Unknown
     2
     3     Anju-si
     4  Hamhung-si

     [5 rows x 36 columns]
```

[217]:
```
df_bombardeos_ubicaciones_validas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11052 entries, 0 to 11051
Data columns (total 36 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ROW_NUMBER                  11052 non-null  int64
 1   MISSION_NUMBER              11042 non-null  object
 2   OP_ORDER                    11043 non-null  object
 3   UNIT                        11040 non-null  object
 4   AIRCRAFT_TYPE_MDS           10428 non-null  object
 5   NBR_ATTACK_EFFEC_AIRCRAFT   11012 non-null  float64
 6   SORTIE_DUPE                 6663 non-null   float64
 7   NBR_ABORT_AIRCRAFT          379 non-null    float64
 8   NBR_LOST_AIRCRAFT           32 non-null     object
 9   TARGET_NAME                 6036 non-null   object
 10  TGT_TYPE                    7758 non-null   object
 11  SOURCE_UTM_JAPAN_B          12 non-null     object
 12  SOURCE_TGT_UTM              8564 non-null   object
 13  TGT_MGRS                    7534 non-null   object
 14  TGT_LATITUDE_WGS84          11052 non-null  float64
 15  TGT_LONGITUDE_WGS84         11052 non-null  float64
 16  SOURCE_TGT_LAT              592 non-null    object
 17  SOURCE_TGT_LONG             585 non-null    object
 18  NBR_OF_WEAPONS              9956 non-null   object
 19  WEAPONS_TYPE                9958 non-null   object
 20  BOMB_SIGHTING_METHOD        8108 non-null   object
 21  TOTAL_BOMBLOAD_IN_LBS       5 non-null      float64
 22  TOT                         1559 non-null   object
 23  MISSION_TYPE                9925 non-null   object
 24  ALTITUDE_FT                 9219 non-null   object
 25  CALLSIGN                    1 non-null      object
 26  BDA                         6680 non-null   object
 27  NOSE_FUZE                   5208 non-null   object
 28  TAIL_FUZE                   4771 non-null   object
 29  CALCULATED_BOMBLOAD_LBS     9876 non-null   float64
 30  RECORD_SOURCE               11052 non-null  object
 31  DAY                         11052 non-null  int64
 32  MONTH                       11052 non-null  int64
 33  YEAR                        11052 non-null  int64
 34  MISSION_DATE                11052 non-null  object
 35  City/Town                   11052 non-null  object
dtypes: float64(7), int64(4), object(25)
memory usage: 3.0+ MB
```

[219]:
```
valores_columna = df_bombardeos_ubicaciones_validas['City/Town']

valores_unicos = df_bombardeos_ubicaciones_validas['City/Town'].unique()
valores_unicos.tolist()
```

[219]:
```
['Changdo-ri',
 'Unknown',
 '  ',
 'Anju-si',
 'Hamhung-si',
 'Pyongsan County',
 'Ushumun',
 '  ',
 '  ',
 'Hwacheon',
 '   ',
 'Komsa-ri',
 'Sinseo-myeon',
 'Sangseo',
 'Jinhyeon-ri',
 'Bangsan-myeon',
 'Buk-myeon',
 'Mungyeong-si',
 'Geunnam-myeon',
 'Huichon-si',
 'Chongju-si',
 '  ',
 'Kumya-up',
 '  ',
 '    ',
 'Kaesong',
 '  ',
 'Songgan-up',
 '  ',
 'Sariwon-si',
 'Otan-ri',
 'Ryonghak-ri',
 '  ',
 'Jung-myeon',
 'Dong-myeon',
 'Seohwa-myeon',
 'Geundong-myeon',
 'Yanggu-eup',
 'Nampo',
 'Paju-si',
 '  ',
 'Paekhyon-ri',
```

```
'  ',
'  ',
'Pyonggang-up',
'Haebang-ni',
'Junggangri',
'Junggang-ri',
'  ',
'Cheorwon-eup',
'  ',
'Seo-myeon',
'Amjeong-ri',
'  ',
'  ',
'Yangji-ri',
'Daema-ri',
'Dongsong-eup',
'   ',
'  ',
'Tomil-ri',
'  ',
'Tanchon-si',
'Jungse-ri',
'  ',
'Woram-ri',
'Odeok-ri',
'Geunbuk-myeon',
'Jeongyeon-ri',
'Hongwon-ri',
'Naedae-ri',
'Odong-ri',
'Gwanu-ri',
'Gadan-ri',
'Igil-ri',
'Samsa-ri',
'Haean-myeon',
'Chongdu-ri',
'Wonnam-ri',
'Sinchang-ri',
'Eumnae-ri',
'Puap-ri',
'Yugok-ri',
'Haeju-si',
'Sinchon-up',
'Kimhwa-up',
'  ',
'Pomak-ri',
'Kubong-ri',
```

```
'Kahak-ri',
'  ',
'Ryongdang-ri',
'  ',
'Unpa-up',
'  ',
'Chorwon-up',
'Ichon-up',
'  ',
'  ',
'  ',
'Sudong-ri',
'  ',
'Hasikjom-ri',
'  ',
'  ',
'Deungdae-ri',
'Sogon-ri',
'  ',
'Wondong-ri',
'Songnim-si',
"P'yŏngyang",
'Hongwon-up',
'Sunchon-si',
'Kowon County',
'Sunan District',
'Hoeyang-up',
'  ',
'  ',
'   ',
'Sinpo-si',
'  ',
'  ',
'  ',
'Sutae-ri',
'Songgo-ri',
'Sohung-up',
'Anak-up',
'Sukchon County',
'Ochon-ri',
'  ',
'Tongsan-ri',
'  ',
'  ',
'Tosan-up',
'  ',
'Seongsan-ri',
```

```
'  ',
'  ',
'Mundong-ri',
'Ipo-ri',
'Hasong-ri',
'Munhye-ri',
'Kumgang-up',
'  ',
'Ryonghyon-ri',
'Kumchon-up',
'  ',
'  ',
'  ',
'Onjin County',
'Kosong-up',
'  ',
'Sinpung-ri',
'  ',
'   ',
'Ripsok-ri',
'Majang-ri',
'  ',
'Miram-ri',
'Tapgeo-ri',
'Hwangju-up',
'  ',
'Wondong-myeon',
'Pyongwon-up',
'Sangmasan-ri',
'Pocheon-si',
'Oho-ri',
'  ',
'  ',
'    ',
'Kaechon',
'Kuyon-ri',
'    ',
'  ',
'   ',
'Sinanju',
'  ',
'Kalhyon-ri',
'  ',
'    ',
'  ',
'  ',
'  ',
```

```
'  ',
'  ',
'Dokgeom-ri',
'Choseo-ri',
'Changyon-up',
'Gangdon-ri',
'  ',
'Yangdok County',
'Singye-up',
'  ',
'  ',
'  ',
'   ',
'Wonsan-si',
'  ',
'  ',
'Orang-ri',
'  ',
'Sudong',
'  ',
'Hwasan-ri',
'    ',
'Rason',
'Jakdong-ri',
'Sinyang-up',
'Unjon-up',
'Naemun-ri',
'Munchon-si',
'  ',
'Misan-myeon',
'  ',
'Sepo-up',
'  ',
'  ',
'Pongsan-up',
'  ',
'  ',
'Apgangnodongja-gu',
'Bangpo-ri',
'Ryongdamnodongja-gu',
'Chotan-ri',
'Ryongpo-ri',
'Koksan-up',
'  ',
'  ',
'Pansok-ri',
'  ',
```

```
'Ryudaepo-ri',
'  ',
'  ',
'Soksa-ri',
'Chongdong-ri',
'Wangjing-myeon',
'Baekhak-myeon',
'Geonji-dong',
'  ',
'Songchon-up',
'Kwaksan-up',
'  ',
'  ',
'Mundok-up',
'  ',
'Guseong-ri',
'Hoechang-up',
'  ',
'Hoesan-ri',
'Sokcho-si',
'  ',
'Tongga-ri',
'Hamju-up',
'Chunghwa-up',
'Aptong-ni',
'Jangnam-myeon',
'Juchon-ri',
'Hukgyo-ri',
'  ',
'Okpyeong',
'  ',
'Daewi-ri',
'  ',
'  ',
'Jasan-ri',
'Hyon-ri',
'Choksan-ri',
'  ',
'  ',
'  ',
'Geumgok-ri',
'Suan-up',
'Sehyeon-ri',
'Songpo-ri',
'Munsan-eup',
'Unjon-ri',
'Nam-myeon',
```

```
'  ',
'Pamgashi',
'Kosan-up',
'Geumpung-ri',
'  ',
'Bukbun-ri',
'Girin-myeon',
'Yonggwang-up',
'Pukchong-up',
'Kapsan-up',
'Kilju-up',
'Chongjin-si',
'  ',
'  ',
'  ',
'  ',
'Sonchon-up',
'  ',
'   ',
'Gapyeong',
'Kusong-si',
'Yangchon',
'Kanggye-si',
'Changpung-up',
'  ',
'Oehak-ri',
'  ',
'  ',
'Tsushima',
'  ',
'Sojeong-ri',
'  ',
'Beopsu-ri',
'Sanae-myeon',
'Kumchon-ri',
'  ',
'Taechon-up',
'  ',
'  ',
'Pakchon-up',
'   ',
'   ',
'Kamdul-li',
'  ',
'  ',
'Guktojeongjungang-myeon',
'  ',
```

```
'Kyongsong County',
'Unyang-dong',
'  ',
'  ',
'  ',
"Sinch'ang",
'Rungdong-ri',
'Gwanjeon-ri',
'Wolha-ri',
'  ',
'Nyongbyon-up',
'Ryangchaek-rodongjagu',
'Namyangni',
'Wacho-ri',
'  ',
'  ',
'   ',
'Taesuk-ri',
'Unryul-up',
'Dochan-ri',
'  ',
'Inje-eup',
'  ',
'Anbyon-up',
'  ',
'  ',
'Gwangjeon-ri',
'  ',
'  ',
'  ',
'Pyokdong-up',
'Santhan-ri',
'  ',
'  ',
'  ',
'  ',
'Ganseong-eup',
'  ',
'Gwanin',
'Hanam',
'Sinbuk',
'Tokchon-si',
'  ',
'Sinhung-up',
'Jinseo-ri',
'  ',
'  ',
```

```
'  ',
'Jeongok-eup',
'Hyeonnae-myeon',
'  ',
'  ',
'Ungok-ri',
'Chuncheon-si',
'  ',
'Sinuiju-si',
'Gueup-ri',
'  ',
'  ',
'   ',
'Hahoe-ri',
'  ',
'Pangmok-ri',
'  ',
'  ',
'  ',
'Taephyong-ri',
'  ',
'   ',
'  ',
'Ryangsa-ri',
'Sacheon-si',
'Pukjom-ri',
'Sudong-myeon',
'Singye-ri',
'  ',
'Hoeryong-si',
'  ',
'Tongchon-up',
'  ',
'Yueup-ri',
'  ',
'Sayo-ri',
'Gamdun-ri',
'  ',
'Naehyeon-ri',
'Ryong-am-ri',
'Pukchang County',
'  ',
'  ',
'Jangheung-ri',
'  ',
'  ',
'  ',
```

```
'  ',
'  ',
'  ',
'Sinbok-ri',
'Gyoju-ri',
'    ',
'Sungap-ri',
'Pyoksong-up',
'  ',
'  ',
'  ',
'  ',
'  ',
'Judun-ri',
'  ',
'Galmal-eup',
'  ',
'Oktok-ri',
'  ',
'    ',
'  ',
'Inhung-ri',
'  ',
'Sokdam-ri',
"Ch'ontae-ri",
'Chungsan-up',
'Sadong-ri',
'  ',
'Hwachon-ri',
'Geojin-eup',
'Myeongu-ri',
'   ',
'  ',
'Hyondong-ri',
'  ',
'  ',
'  ',
'  ',
'  ',
'  ',
'    ',
'  ',
'Gomun-ri',
'  ',
'  ',
'Gandong',
'  ',
```

```
'  ',
'  ',
' ',
'  ',
'  ',
'Oji-ri',
'Unchon-up',
'  ',
'  ',
'Toseong-ri',
'  ',
'Pangyo-up',
'  ',
'  ',
'  ',
'  ',
'  ',
'  ',
'  ',
'Jungcheon-ri',
'  ',
'Taejon-ri',
'Punghyeon-ri',
'  ',
'Yangsa-myeon',
'Kyongdo-ri',
'  ',
'  ',
'Rotan-ri',
'  ',
'Goseong-ri',
'Sinphyong-up',
'  ',
'  ',
'  ',
'   ',
'  ',
'  ',
'  ',
'  ',
'Pyongsong-si',
'  ',
'  ',
'  ',
'Unsan-up',
'  ',
'  ',
```

```
'  ',
'  ',
'  ',
'  ',
'  ',
'Gangneung-si',
'Tongsin-up',
'  ',
'  ',
'Mingyue',
'  ',
'  ',
'Nyongwon-up',
'Tukchang',
'  ',
'  ',
'Geum Pyeongni',
'  ',
'    ',
'Sangeum-ri',
'    ',
'    ',
'  ',
'  ',
'   ',
'  ',
'  ',
'  ',
'  ',
'  ',
'    ',
'  ',
'  ',
'  ',
'    ',
'    ',
'  ',
'  ',
'  ',
'  ',
'  ',
'Kujang-up',
'  ',
'  ',
'Ohyon-ri',
'  ',
'  ',
```

```
'  ',
'  ',
'Ryongsan-ri',
'  ',
'   ',
'Yonsan-up',
'  ',
'Kasung-ri',
'  ',
'Jangsan-ri',
'  ',
'Wafangdian City',
'  ',
'Gisan-ri',
'Seungjeon-ri',
'Kumphung-ri',
'Chonnae-up',
'   ',
'  ',
'  ',
'  ',
'Paegwŏn',
'Rosang-ri',
'Sinphung-ri',
'  ',
'  ',
'Dongsifangtai',
'  ',
'Jeokseong-myeon',
'  ',
'   ',
'  ',
'  ',
'  ',
'   ',
'Gushan',
'  ',
'Suphung-rodongjagu',
'  ',
'Dashiqiao City',
'   ',
'Kumbu-ri',
'  ',
'  ',
'  ',
'Chongpyong-up',
'Ryŏnpho-ri',
```

```
'  ',
'  ',
'Hyangsan-up',
'  ',
'Chongsu-rodongjagu',
'  ',
'Namsa-rodongjagu',
'Sa Pyeongni',
'    ',
'    ',
'  ',
'  ',
'  ',
'    ',
'  ',
'Miyang-myeon',
'Jonggok-ri',
'Bongsan-ri',
'Baehwa-ri',
'  ',
'  ',
'Yomju-up',
'  ',
'Ryukwa-ri',
'  ',
'  ',
'  ',
'  ',
'    ',
'Namwon-si',
'  ',
'Sindang-ri',
'  ',
'  ',
'Cholsan-up',
'  ',
'  ',
'Õgung Workers District',
'Tongphyong-ri',
'  ',
'Uiju-up',
'  ',
'Songsan-ri',
'Cheonsam-ri',
'  ',
'Hwararodongja-gu',
'  ',
```

```
 '  ',
 '  ',
 '  ',
 '  ',
 '  ',
 '  ',
 '  ',
 '  ',
 '   ',
 '  ',
 '  ',
 '  ',
 'Pongryon-ri',
 'Kwangmyong-ri',
 '  ',
 'Chonma-up',
 '  ',
 '  ',
 '  ',
 'Komam-ri',
 'Onjong-ri',
 'Phungsong-ri',
 'Hancheon-dong',
 'Sinsi-ri',
 '  ',
 'Kuup-ri',
 '  ',
 'Oncheon-ri',
 'Ryeonho-dong',
 '  ',
 'Phungmi-ri']
```

[220]:
```
from deep_translator import GoogleTranslator
import re

# Crear un objeto traductor
translator = GoogleTranslator(source='auto', target='en')

# Función para verificar si un texto contiene caracteres no latinos
def is_non_english(text):
    return bool(re.search(r'[^\x00-\x7F]', str(text)))

# Filtrar los valores únicos no ingleses
valores_unicos = df_bombardeos_ubicaciones_validas['City/Town'].dropna().
 ↪unique()
non_english_values = [value for value in valores_unicos if␣
 ↪is_non_english(value)]
```

```
# Crear un diccionario de traducción
translations = {}

for value in non_english_values:
    try:
        translated = translator.translate(value)
        translations[value] = translated
    except Exception as e:
        translations[value] = "Unknown"  # Manejar errores de traducción

# Aplicar las traducciones al DataFrame
df_bombardeos_ubicaciones_validas['City/Town'] =␣
 ↪df_bombardeos_ubicaciones_validas['City/Town'].replace(translations)

# Guardar los resultados actualizados en un archivo CSV
df_bombardeos_ubicaciones_validas.to_csv("df_bombardeos_traducido.csv",␣
 ↪index=False)

# Mostrar un fragmento del DataFrame actualizado
print(df_bombardeos_ubicaciones_validas[['City/Town']].head())
```

```
    City/Town
0  Changdo-ri
1     Unknown
2   Seosam-ri
3     Anju-si
4  Hamhung-si
```

[221]:
```
file_path = './data/df_bombardeos_traducido.csv'
df_bombardeos_traducido = pd.read_csv(file_path, sep=',')
df_bombardeos_traducido.head()
```

[221]:

	ROW_NUMBER	MISSION_NUMBER	OP_ORDER	UNIT	AIRCRAFT_TYPE_MDS	\
0	2	433	174-51	98th Bomb Wing	B-29	
1	3	433	174-51	307th Bomb Wing	B-29	
2	4	433	174-51	307th Bomb Wing	B-29	
3	5	433	174-51	98th Bomb Wing	B-29	
4	6	433	174-51	98th Bomb Wing	B-29	

	NBR_ATTACK_EFFEC_AIRCRAFT	SORTIE_DUPE	NBR_ABORT_AIRCRAFT	\
0	1.0	NaN	NaN	
1	NaN	1.0	NaN	
2	1.0	1.0	NaN	
3	1.0	NaN	NaN	
4	1.0	NaN	NaN	

```
  NBR_LOST_AIRCRAFT TARGET_NAME  …  \
0               NaN  Changdo-ri  …
1               NaN         NaN  …
2               NaN         NaN  …
3               NaN        Anju  …
4               NaN     Hamhung  …

                                                BDA NOSE_FUZE  TAIL_FUZE  \
0           Bombs fell on the east end of the tracks      0.01  Non-delay
1                                                NaN      0.01  Non-delay
2  1 aircraft due to a bomb rack malfunction drop…      0.01  Non-delay
3                                                NaN      0.01  Non-delay
4                                                NaN      0.01  Non-delay

  CALCULATED_BOMBLOAD_LBS  RECORD_SOURCE  DAY MONTH  YEAR MISSION_DATE  \
0                 12000.0         EXETER    1     6  1951   1951-06-01
1                  4000.0         EXETER    1     6  1951   1951-06-01
2                  8000.0         EXETER    1     6  1951   1951-06-01
3                 16000.0         EXETER    1     6  1951   1951-06-01
4                 16000.0         EXETER    1     6  1951   1951-06-01

    City/Town
0  Changdo-ri
1     Unknown
2   Seosam-ri
3     Anju-si
4  Hamhung-si

[5 rows x 36 columns]
```

[222]:
```
df_bombardeos_traducido.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11052 entries, 0 to 11051
Data columns (total 36 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   ROW_NUMBER                 11052 non-null  int64
 1   MISSION_NUMBER             11042 non-null  object
 2   OP_ORDER                   11043 non-null  object
 3   UNIT                       11040 non-null  object
 4   AIRCRAFT_TYPE_MDS          10428 non-null  object
 5   NBR_ATTACK_EFFEC_AIRCRAFT  11012 non-null  float64
 6   SORTIE_DUPE                6663 non-null   float64
 7   NBR_ABORT_AIRCRAFT         379 non-null    float64
 8   NBR_LOST_AIRCRAFT          32 non-null     object
 9   TARGET_NAME                6036 non-null   object
```

```
 10  TGT_TYPE                  7758 non-null   object
 11  SOURCE_UTM_JAPAN_B        12 non-null     object
 12  SOURCE_TGT_UTM            8564 non-null   object
 13  TGT_MGRS                  7534 non-null   object
 14  TGT_LATITUDE_WGS84        11052 non-null  float64
 15  TGT_LONGITUDE_WGS84       11052 non-null  float64
 16  SOURCE_TGT_LAT            592 non-null    object
 17  SOURCE_TGT_LONG           585 non-null    object
 18  NBR_OF_WEAPONS            9956 non-null   object
 19  WEAPONS_TYPE              9958 non-null   object
 20  BOMB_SIGHTING_METHOD      8108 non-null   object
 21  TOTAL_BOMBLOAD_IN_LBS     5 non-null      float64
 22  TOT                       1559 non-null   object
 23  MISSION_TYPE              9925 non-null   object
 24  ALTITUDE_FT               9219 non-null   object
 25  CALLSIGN                  1 non-null      object
 26  BDA                       6680 non-null   object
 27  NOSE_FUZE                 5208 non-null   object
 28  TAIL_FUZE                 4771 non-null   object
 29  CALCULATED_BOMBLOAD_LBS   9876 non-null   float64
 30  RECORD_SOURCE             11052 non-null  object
 31  DAY                       11052 non-null  int64
 32  MONTH                     11052 non-null  int64
 33  YEAR                      11052 non-null  int64
 34  MISSION_DATE              11052 non-null  object
 35  City/Town                 11052 non-null  object
dtypes: float64(7), int64(4), object(25)
memory usage: 3.0+ MB
```

[223]:
```
valores_columna = df_bombardeos_ubicaciones_validas['City/Town']

valores_unicos = df_bombardeos_ubicaciones_validas['City/Town'].unique()
valores_unicos.tolist()
```

[223]:
```
['Changdo-ri',
 'Unknown',
 'Seosam-ri',
 'Anju-si',
 'Hamhung-si',
 'Pyongsan County',
 'Ushumun',
 'Green River Village',
 'Colonel Lee',
 'Hwacheon',
 'Baekrosanri',
 'Komsa-ri',
 'Sinseo-myeon',
```

```
'Sangseo',
'Jinhyeon-ri',
'Bangsan-myeon',
'Buk-myeon',
'Mungyeong-si',
'Geunnam-myeon',
'Huichon-si',
'Chongju-si',
'Sentence Lee',
'Kumya-up',
'Jungnam-ri',
'Dongsan Labor Union',
'Kaesong',
'Chojangri',
'Songgan-up',
'Gacheon-ri',
'Sariwon-si',
'Otan-ri',
'Ryonghak-ri',
'Seongsan-ri',
'Jung-myeon',
'Dong-myeon',
'Seohwa-myeon',
'Geundong-myeon',
'Yanggu-eup',
'Nampo',
'Paju-si',
'Reasonable price',
'Paekhyon-ri',
'Jeokdong-ri',
'Cheonam-ri',
'Pyonggang-up',
'Haebang-ni',
'Junggangri',
'Junggang-ri',
'Bokmanri',
'Cheorwon-eup',
'Jungsamri',
'Seo-myeon',
'Amjeong-ri',
'Geoncheon-ri',
'Apdong-ri',
'Yangji-ri',
'Daema-ri',
'Dongsong-eup',
'Lee Su-deok-ri',
'Bokgye-ri',
```

```
'Tomil-ri',
'Large fish',
'Tanchon-si',
'Jungse-ri',
'Okdong-ri',
'Woram-ri',
'Odeok-ri',
'Geunbuk-myeon',
'Jeongyeon-ri',
'Hongwon-ri',
'Naedae-ri',
'Odong-ri',
'Gwanu-ri',
'Gadan-ri',
'Igil-ri',
'Samsa-ri',
'Haean-myeon',
'Chongdu-ri',
'Wonnam-ri',
'Sinchang-ri',
'Eumnae-ri',
'Puap-ri',
'Yugok-ri',
'Haeju-si',
'Sinchon-up',
'Kimhwa-up',
'Yongheung-ri',
'Pomak-ri',
'Kubong-ri',
'Kahak-ri',
'Red Star',
'Ryongdang-ri',
'Yongpo-ri',
'Unpa-up',
'Ryu Jeong-ri',
'Chorwon-up',
'Ichon-up',
'Ryongseong-ri',
'Song Se-ri',
'Songcheon-ri',
'Sudong-ri',
'Chukdong-ri',
'Hasikjom-ri',
'Gusan-ri',
'Lihari',
'Deungdae-ri',
'Sogon-ri',
```

```
'Gui Rak Ri',
'Wondong-ri',
'Songnim-si',
"P'yŏngyang",
'Hongwon-up',
'Sunchon-si',
'Kowon County',
'Sunan District',
'Hoeyang-up',
'Wonbuk-ri',
'My Gang-ri',
'Hakbang Labor Union',
'Sinpo-si',
'Saengyang-ri',
'Seongdori',
'Street noise',
'Sutae-ri',
'Songgo-ri',
'Sohung-up',
'Anak-up',
'Sukchon County',
'Ochon-ri',
'Sangbuk-dong-ri',
'Tongsan-ri',
'Gunhan-ri',
'Daeam-ri',
'Tosan-up',
'Shinsung-ri',
'Yangsari',
'Suhapri',
'Mundong-ri',
'Ipo-ri',
'Hasong-ri',
'Munhye-ri',
'Kumgang-up',
'Mountain geography',
'Ryonghyon-ri',
'Kumchon-up',
'Pangyo-ri',
'Langhari',
'Seonam-ri',
'Onjin County',
'Kosong-up',
'Umiri',
'Sinpung-ri',
'Wolseong-ri',
'Yangcheonseori',
```

```
'Ripsok-ri',
'Majang-ri',
'Flexible',
'Miram-ri',
'Tapgeo-ri',
'Hwangju-up',
'Yonggyo-ri',
'Wondong-myeon',
'Pyongwon-up',
'Sangmasan-ri',
'Pocheon-si',
'Oho-ri',
'Wolhyeon-ri',
'Naesan-ri',
'Shinsungcheon Labor Union',
'Kaechon',
'Kuyon-ri',
'Aparodongjagu',
'New Physiology',
'Forwarding Management',
'Sinanju',
'Geumpyeong-ri',
'Kalhyon-ri',
'Top plate',
'Jangchon Labor Union',
'Songjeong-ri',
'Seokam-ri',
'Jeongsan-ri',
'Moonbong-ri',
'Hyangdong-ri',
'Dokgeom-ri',
'Choseo-ri',
'Changyon-up',
'Gangdon-ri',
'Gaecheon-ri',
'Yangdok County',
'Singye-up',
'geography',
'Foguri',
'Mundeungri',
'Churan Jeonri',
'Wonsan-si',
'Hwanggang-ri',
'Jeongok-ri',
'Orang-ri',
'Picnic',
'Sudong',
```

```
'Yongam-ri',
'Hwasan-ri',
'Maengjung Labor Union',
'Rason',
'Jakdong-ri',
'Sinyang-up',
'Unjon-up',
'Naemun-ri',
'Munchon-si',
'Geundong-ri',
'Misan-myeon',
'Baekyang-ri',
'Sepo-up',
'Sambangri',
'Masan-ri',
'Pongsan-up',
'Namjeong-ri',
'Midang-ri',
'Apgangnodongja-gu',
'Bangpo-ri',
'Ryongdamnodongja-gu',
'Chotan-ri',
'Ryongpo-ri',
'Koksan-up',
'Baekilri',
'Seondeok-ri',
'Pansok-ri',
'Jungsan-ri',
'Ryudaepo-ri',
'Donghori',
'Munsan-ri',
'Soksa-ri',
'Chongdong-ri',
'Wangjing-myeon',
'Baekhak-myeon',
'Geonji-dong',
'Seohwari',
'Songchon-up',
'Kwaksan-up',
'Go Yeon-ri',
'Ma Bang-ri',
'Mundok-up',
'Hwangryong-ri',
'Guseong-ri',
'Hoechang-up',
'Deokryun-ri',
'Hoesan-ri',
```

```
'Sokcho-si',
'Tongga-ri',
'Hamju-up',
'Chunghwa-up',
'Aptong-ni',
'Jangnam-myeon',
'Juchon-ri',
'Hukgyo-ri',
'Geumran-ri',
'Okpyeong',
'Ryongbanri',
'Daewi-ri',
'Resources',
'Misong-ri',
'Jasan-ri',
'Hyon-ri',
'Choksan-ri',
'Jeongdong-ri',
'Eoryong-ri',
'Hwaam-ri',
'Geumgok-ri',
'Suan-up',
'Sehyeon-ri',
'Songpo-ri',
'Munsan-eup',
'Unjon-ri',
'Nam-myeon',
'Bongrae-ri',
'Pamgashi',
'Kosan-up',
'Geumpung-ri',
'Hwayari',
'Bukbun-ri',
'Girin-myeon',
'Yonggwang-up',
'Pukchong-up',
'Kapsan-up',
'Kilju-up',
'Chongjin-si',
'Anbong-ri',
'Namcheonri',
'Daegok-ri',
'Steel Dong',
'Sonchon-up',
'Hwanhyeon-ri',
'Info Labor Union',
'Gapyeong',
```

```
'Kusong-si',
'Yangchon',
'Kanggye-si',
'Changpung-up',
'Recommended',
'Oehak-ri',
'Rimokri',
'Haepori',
'Tsushima',
'Suwon-dong',
'Sojeong-ri',
'Munyang-ri',
'Beopsu-ri',
'Sanae-myeon',
'Kumchon-ri',
'Gosan-ri',
'Taechon-up',
'Diagonal',
'Songchon-ri',
'Pakchon-up',
'Bukjin Labor Union',
"Lakeside Workers' Union",
'Kamdul-li',
'Daebaekri',
'Three thousand miles',
'Guktojeongjungang-myeon',
'Namjung-ri',
'Kyongsong County',
'Unyang-dong',
'Jeonseungri',
'Songryeon-ri',
'Yongdaeri',
"Sinch'ang",
'Rungdong-ri',
'Gwanjeon-ri',
'Wolha-ri',
'Danghyeon-ri',
'Nyongbyon-up',
'Ryangchaek-rodongjagu',
'Namyangni',
'Wacho-ri',
'Jeongbong-ri',
'Sangduri',
'Hwajeon Labor Union',
'Taesuk-ri',
'Unryul-up',
'Dochan-ri',
```

```
'Sadong-ri',
'Inje-eup',
'Seorim-ri',
'Anbyon-up',
'Gwangbok-dong',
'Kijeongri',
'Gwangjeon-ri',
'Jangdong-ri',
'Pungcheon-ri',
'Song Sam-ri',
'Pyokdong-up',
'Santhan-ri',
'Bongui-ri',
'Wonduk-ri',
'Separi',
'Jang Gook-li',
'Ganseong-eup',
'Thousand Horses',
'Gwanin',
'Hanam',
'Sinbuk',
'Tokchon-si',
'Bukpori',
'Sinhung-up',
'Jinseo-ri',
'Shinchang-ri',
'Taepyeong-ri',
'Ryangwolli',
'Jeongok-eup',
'Hyeonnae-myeon',
'Shinpo-ri',
'Shinwon-ri',
'Ungok-ri',
'Chuncheon-si',
'Hwatong-ri',
'Sinuiju-si',
'Gueup-ri',
'Namsan-ri',
'Dokjeong-ri',
'City Labor Union',
'Hahoe-ri',
'Gisan-ri',
'Pangmok-ri',
'Deokchon-ri',
'Daeseong-ri',
'Thousand-year-old',
'Taephyong-ri',
```

```
'Bongpo-ri',
'Chahoro Labor Union',
'Byeolhari',
'Ryangsa-ri',
'Sacheon-si',
'Pukjom-ri',
'Sudong-myeon',
'Singye-ri',
'Donghari',
'Hoeryong-si',
'Guhang-ri',
'Tongchon-up',
'Geumcheol-ri',
'Yueup-ri',
'Seongpyeong-ri',
'Sayo-ri',
'Gamdun-ri',
'Doeumri',
'Naehyeon-ri',
'Ryong-am-ri',
'Pukchang County',
'Moonseong-ri',
'A hundred horses',
'Jangheung-ri',
'Jang Jae-ri',
'Guryong-ri',
'Chowon-ri',
'Bonghari',
'Cheongjeong-ri',
'Songhwari',
'Sinbok-ri',
'Gyoju-ri',
"Standing Workers' Union",
'Sungap-ri',
'Pyoksong-up',
'Water vapor',
'Jiseok-ri',
'An Miri',
'Dangsan-ri',
'Geumbong-ri',
'Judun-ri',
'Honam-ri',
'Galmal-eup',
'Daeryong-ri',
'Oktok-ri',
'Jinheung-ri',
'Shinsang Labor Union',
```

```
'Samdori',
'Inhung-ri',
'Okhyun-ri',
'Sokdam-ri',
"Ch'ontae-ri",
'Chungsan-up',
'Ryongcheon-ri',
'Hwachon-ri',
'Geojin-eup',
'Myeongu-ri',
'Bank reorganization',
'Daemun-ri',
'Hyondong-ri',
'Baeksan-ri',
'Pocheon-ri',
'Sacheongri',
'Haptan-ri',
'Seokbong-ri',
'Gaeryeonri',
'Ryongcheol Labor Union',
'Jinam-ri',
'Gomun-ri',
'Hoam-ri',
'Accounting',
'Gandong',
'Deokheung-ri',
'Byeolsang-ri',
'Goeup-ri',
'Yuli',
'Daegunri',
'Construction site',
'Oji-ri',
'Unchon-up',
'Rim Seong-ri',
'Seochon-ri',
'Toseong-ri',
'Sanchamri',
'Pangyo-up',
'Hanam-ri',
'Seowon-ri',
'Geumok-ri',
'Gwanseong-ri',
'Ryongyeon-ri',
'Sangsulli',
'Jungcheon-ri',
'Wonhari',
'Taejon-ri',
```

```
'Punghyeon-ri',
'Yaksuri',
'Yangsa-myeon',
'Kyongdo-ri',
'Shindong-ri',
'Obong-ri',
'Rotan-ri',
'Forward vibration',
'Goseong-ri',
'Sinphyong-up',
'Hagyori',
'Bud',
'Geumseong-ri',
'Myeokmi Labor Union',
'Taekinri',
'Shinpung-ri',
'Jehyeon-ri',
'Wonam-ri',
'Pyongsong-si',
'Yongjeon-ri',
'Dongnim-ri',
'Daecheong-ri',
'Unsan-up',
'Mukbang-dong',
'Yongjin-dong',
'Joyang-dong',
'Inam-ri',
'Induri',
'Jeongpyeong-ri',
'Shinsang-ri',
'Gangneung-si',
'Tongsin-up',
'Sujeon-ri',
'Unheung-ri',
'Mingyue',
'Jungheung-ri',
'Nyongwon-up',
'Tukchang',
'Gwisang-ri',
'Union Lee',
'Geum Pyeongni',
'Gyesok-ri',
'Buraesan Labor Union',
'Sangeum-ri',
'Cheon Seong Labor Union',
'Gubong Labor Union',
'Deok-eum-ri',
```

```
'Inheung-ri',
'Saemaeul-ri',
'Daeheung-ri',
'Gwanghwari',
'Lingyunri',
'Yoo Seung-ri',
'Moonsamri',
"The deceased worker's ward",
'Pungjeon-ri',
'Limheung-ri',
'Self-operation',
'Jaedong Labor Union',
'Songnam Labor Union',
'Sangchori',
'Samsung-ri',
'Maenghari',
'Dopyeong-ri',
'Jiseong-ri',
'Kujang-up',
'Sangyang-ri',
'Ohyon-ri',
'Rotary wheel',
'Songjeon-ri',
'Unpori',
'Ryongsan-ri',
'Yeonhong-ri',
'Sujeon Labor Union',
'Yonsan-up',
'Sangap-ri',
'Kasung-ri',
'Namwon-ri',
'Jangsan-ri',
'Wonseori',
'Wafangdian City',
'Sudeok-ri',
'Seungjeon-ri',
'Kumphung-ri',
'Chonnae-up',
'Hwacheon Labor Union',
'Gownley',
'Jeonsan-ri',
'Seopyeong-ri',
'Paegwŏn',
'Rosang-ri',
'Sinphung-ri',
'Yeomtan-ri',
'Dongchang-ri',
```

'Dongsifangtai',
'Deokam-ri',
'Jeokseong-myeon',
'Sanghari',
'Eungok Labor Union',
'Jidong-ri',
'Yongsan-ri',
'Joyang-ri',
'Namcheon Labor Union',
'Gushan',
'Yang Chun-ri',
'Suphung-rodongjagu',
'Alildong',
'Dashiqiao City',
'Haksan Labor Union',
'Kumbu-ri',
'Wolbong-ri',
'Dongsari',
'Lip stone',
'Chongpyong-up',
'Ryŏnpho-ri',
'Ryanggyo-ri',
'Cheonsuri',
'Hyangsan-up',
'Daedeok-ri',
'Chongsu-rodongjagu',
'Jigyeong-ri',
'Namsa-rodongjagu',
'Sa Pyeongni',
'Yongmun Labor Union',
'Ryongsu Labor Union',
'Hoeun-ri',
'Hachori',
'Songdong-ri',
'Daeyuro Labor Union',
'Suncheon-ri',
'Miyang-myeon',
'Jonggok-ri',
'Bongsan-ri',
'Baehwa-ri',
'Middle East',
'Jangsa-ri',
'Yomju-up',
'Lee Cheon-ri',
'Ryukwa-ri',
'Dongyang-ri',
'Naedong-ri',

```
'Cheolsan-ri',
'Raw interest rate',
'Holdong Labor Union',
'Namwon-si',
'Cultural Center',
'Sindang-ri',
'Jangcheon-ri',
'Donggori',
'Cholsan-up',
'Cheongok-ri',
'Ayang-ri',
'Ŏgung Workers District',
'Tongphyong-ri',
'Hongnam-ri',
'Uiju-up',
'Ryongnam-ri',
'Songsan-ri',
'Cheonsam-ri',
'Ssangryong-ri',
'Hwararodongja-gu',
'Civilization',
'Buksinhyeon-ri',
'Gwangheung-ri',
'Yeonsam-ri',
'Giyang-ri',
'Chang Gaeri',
'Outer frost',
'Bongheung-ri',
'Unbong-ri',
'Palwon Labor Union',
'Approval',
'Gwansang-ri',
'Gwanbong-ri',
'Pongryon-ri',
'Kwangmyong-ri',
'Songgang-ri',
'Chonma-up',
'Dongsam-ri',
'Daecheonri',
'Long-term interest rate',
'Komam-ri',
'Onjong-ri',
'Phungsong-ri',
'Hancheon-dong',
'Sinsi-ri',
'Moon Ok-ri',
'Kuup-ri',
```

```
 'Jangseong-ri',
 'Oncheon-ri',
 'Ryeonho-dong',
 'Yangjiri',
 'Phungmi-ri']
```

[224]:
```
# Contar el número de valores "Unknown" en la columna "City/Town"
unknown_count = df_bombardeos_traducido['City/Town'].value_counts().
 ↪get('Unknown', 0)

print(f"Número de valores 'Unknown' en la columna 'City/Town': {unknown_count}")
```

```
Número de valores 'Unknown' en la columna 'City/Town': 1770
```

[225]:
```
# Calcular el porcentaje de valores "Unknown"
total_values = df_bombardeos_traducido['City/Town'].notna().sum()  # Total de␣
 ↪valores no nulos
unknown_percentage = (unknown_count / total_values) * 100

print(f"Porcentaje de valores 'Unknown': {unknown_percentage:.2f}%")
```

```
Porcentaje de valores 'Unknown': 16.02%
```

[226]:
```
# Filtrar los registros con "Unknown"
unknowns_df = df_bombardeos_traducido[df_bombardeos_traducido['City/Town'] ==␣
 ↪'Unknown']
```

[227]:
```
# Estadísticas básicas de latitud y longitud de las ubicaciones desconocidas
unknown_lat_stats = unknowns_df['TGT_LATITUDE_WGS84'].describe()
unknown_lon_stats = unknowns_df['TGT_LONGITUDE_WGS84'].describe()
print("Estadísticas de Latitud (Unknown):")
print(unknown_lat_stats)
print("\nEstadísticas de Longitud (Unknown):")
print(unknown_lon_stats)
```

```
Estadísticas de Latitud (Unknown):
count    1770.000000
mean       38.904033
std         0.555895
min        35.276700
25%        38.522150
50%        38.870323
75%        39.122121
max        41.407910
Name: TGT_LATITUDE_WGS84, dtype: float64

Estadísticas de Longitud (Unknown):
count    1770.000000
```

```
mean      126.888604
std         0.798579
min       121.278270
25%       126.551477
50%       126.827378
75%       127.242775
max       130.164860
Name: TGT_LONGITUDE_WGS84, dtype: float64
```

[228]:
```python
# Revisar si las ubicaciones están en Corea del Norte (37-43 latitud, 124-131 longitud)
def is_in_north_korea(lat, lon):
    return 37 <= lat <= 43 and 124 <= lon <= 131

unknowns_df['In_North_Korea'] = unknowns_df.apply(
    lambda row: is_in_north_korea(row['TGT_LATITUDE_WGS84'], row['TGT_LONGITUDE_WGS84']),
    axis=1
)

# Contar cuántos están dentro de Corea del Norte
north_korea_count = unknowns_df['In_North_Korea'].sum()
print(f"Registros Unknown en Corea del Norte: {north_korea_count}")
```

```
Registros Unknown en Corea del Norte: 1756
```

[229]:
```python
# Actualizar la columna 'City/Town' para los registros en Corea del Norte
df_bombardeos_traducido.loc[
    (df_bombardeos_traducido['City/Town'] == 'Unknown') &
    (df_bombardeos_traducido['TGT_LATITUDE_WGS84'].between(37, 43)) &
    (df_bombardeos_traducido['TGT_LONGITUDE_WGS84'].between(124, 131)),
    'City/Town'
] = 'North Korea'

# Guardar el DataFrame actualizado en un archivo CSV (opcional)
df_bombardeos_traducido.to_csv("df_bombardeos_traducido_actualizado.csv", index=False)

# Mostrar un fragmento del DataFrame actualizado
df_bombardeos_traducido[['TGT_LATITUDE_WGS84', 'TGT_LONGITUDE_WGS84', 'City/Town']].head()
```

[229]:

	TGT_LATITUDE_WGS84	TGT_LONGITUDE_WGS84	City/Town
0	38.498810	127.669740	Changdo-ri
1	38.935944	126.781969	North Korea
2	39.668790	125.506530	Seosam-ri
3	39.602150	125.667170	Anju-si

```
 4           39.912170           127.560910   Hamhung-si
```

[230]:
```
df_bombardeos_traducido.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11052 entries, 0 to 11051
Data columns (total 36 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ROW_NUMBER                  11052 non-null  int64
 1   MISSION_NUMBER              11042 non-null  object
 2   OP_ORDER                    11043 non-null  object
 3   UNIT                        11040 non-null  object
 4   AIRCRAFT_TYPE_MDS           10428 non-null  object
 5   NBR_ATTACK_EFFEC_AIRCRAFT   11012 non-null  float64
 6   SORTIE_DUPE                 6663 non-null   float64
 7   NBR_ABORT_AIRCRAFT          379 non-null    float64
 8   NBR_LOST_AIRCRAFT           32 non-null     object
 9   TARGET_NAME                 6036 non-null   object
 10  TGT_TYPE                    7758 non-null   object
 11  SOURCE_UTM_JAPAN_B          12 non-null     object
 12  SOURCE_TGT_UTM              8564 non-null   object
 13  TGT_MGRS                    7534 non-null   object
 14  TGT_LATITUDE_WGS84          11052 non-null  float64
 15  TGT_LONGITUDE_WGS84         11052 non-null  float64
 16  SOURCE_TGT_LAT              592 non-null    object
 17  SOURCE_TGT_LONG             585 non-null    object
 18  NBR_OF_WEAPONS              9956 non-null   object
 19  WEAPONS_TYPE                9958 non-null   object
 20  BOMB_SIGHTING_METHOD        8108 non-null   object
 21  TOTAL_BOMBLOAD_IN_LBS       5 non-null      float64
 22  TOT                         1559 non-null   object
 23  MISSION_TYPE                9925 non-null   object
 24  ALTITUDE_FT                 9219 non-null   object
 25  CALLSIGN                    1 non-null      object
 26  BDA                         6680 non-null   object
 27  NOSE_FUZE                   5208 non-null   object
 28  TAIL_FUZE                   4771 non-null   object
 29  CALCULATED_BOMBLOAD_LBS     9876 non-null   float64
 30  RECORD_SOURCE               11052 non-null  object
 31  DAY                         11052 non-null  int64
 32  MONTH                       11052 non-null  int64
 33  YEAR                        11052 non-null  int64
 34  MISSION_DATE                11052 non-null  object
 35  City/Town                   11052 non-null  object
dtypes: float64(7), int64(4), object(25)
memory usage: 3.0+ MB
```

[231]:
```
# Revisar si hay ubicaciones nulas
null_coords_count = unknowns_df[['TGT_LATITUDE_WGS84', 'TGT_LONGITUDE_WGS84']].
 ↪isnull().any(axis=1).sum()
print(f"Registros Unknown con coordenadas nulas: {null_coords_count}")
```

```
Registros Unknown con coordenadas nulas: 0
```

[232]:
```
import geopandas as gpd
from shapely.geometry import Point
```

[233]:
```
shapefile_path = "./data/countries_mapas/ne_110m_admin_0_countries.shp"
world = gpd.read_file(shapefile_path)
```

[236]:
```
# Crear puntos para las ubicaciones desconocidas
unknowns_df['geometry'] = unknowns_df.apply(
    lambda row: Point(row['TGT_LONGITUDE_WGS84'], row['TGT_LATITUDE_WGS84'])
    if not pd.isna(row['TGT_LONGITUDE_WGS84']) and not pd.
 ↪isna(row['TGT_LATITUDE_WGS84']) else None,
    axis=1
)

# Convertir a GeoDataFrame
unknowns_gdf = gpd.GeoDataFrame(unknowns_df, geometry='geometry')

# Realizar un cruce espacial para verificar si están en tierra
unknowns_gdf['In_Land'] = unknowns_gdf.geometry.apply(
    lambda x: any(world.contains(x)) if x else False
)

# Contar cuántos están en el agua
in_water_count = (~unknowns_gdf['In_Land']).sum()
print(f"Registros Unknown en el agua: {in_water_count}")
```

```
Registros Unknown en el agua: 71
```

[237]:
```
# Contar el número de valores "Unknown" en la columna "City/Town"
unknown_count = df_bombardeos_traducido['City/Town'].value_counts().
 ↪get('Unknown', 0)

print(f"Número de valores 'Unknown' en la columna 'City/Town': {unknown_count}")
```

```
Número de valores 'Unknown' en la columna 'City/Town': 14
```

[240]:
```
import geopandas as gpd
from shapely.geometry import Point
```

[242]:
```
# Crear puntos para las ubicaciones desconocidas ('Unknown')
df_bombardeos_traducido['geometry'] = df_bombardeos_traducido.apply(
    lambda row: Point(row['TGT_LONGITUDE_WGS84'], row['TGT_LATITUDE_WGS84'])
    if not pd.isna(row['TGT_LONGITUDE_WGS84']) and not pd.
 ↪isna(row['TGT_LATITUDE_WGS84']) else None,
    axis=1
)

# Convertir el DataFrame en un GeoDataFrame
bombardeos_gdf = gpd.GeoDataFrame(df_bombardeos_traducido, geometry='geometry')

# Realizar el cruce espacial para verificar si están en tierra
bombardeos_gdf['In_Land'] = bombardeos_gdf.geometry.apply(
    lambda x: world.contains(x).any() if x else False
)

# Filtrar los registros 'Unknown' y en el agua (In_Land = False)
unknown_in_water = bombardeos_gdf[
    (bombardeos_gdf['City/Town'] == 'Unknown') & (~bombardeos_gdf['In_Land'])
]

# Sustituir 'Unknown' por 'Water' para estos registros
bombardeos_gdf.loc[unknown_in_water.index, 'City/Town'] = 'Water'

# Guardar el DataFrame actualizado
output_path = "./df_bombardeos_actualizado.csv"
bombardeos_gdf.to_csv(output_path, index=False)

print(f"Archivo actualizado guardado en: {output_path}")
```

```
Archivo actualizado guardado en: ./df_bombardeos_actualizado.csv
```

[243]:
```
file_path = './data/df_bombardeos_actualizado_02.csv'
df_bombardeos_processed = pd.read_csv(file_path, sep=',')
df_bombardeos_processed.head()
```

[243]:

	ROW_NUMBER	MISSION_NUMBER	OP_ORDER	UNIT	AIRCRAFT_TYPE_MDS	NBR_ATTACK_EFFEC_AIRCRAFT	SORTIE_DUPE	NBR_ABORT_AIRCRAFT
0	2	433	174-51	98th Bomb Wing	B-29	1.0	NaN	NaN
1	3	433	174-51	307th Bomb Wing	B-29	NaN	1.0	NaN
2	4	433	174-51	307th Bomb Wing	B-29	1.0	1.0	NaN
3	5	433	174-51	98th Bomb Wing	B-29			
4	6	433	174-51	98th Bomb Wing	B-29			

```
3                       1.0          NaN                 NaN
4                       1.0          NaN                 NaN

  NBR_LOST_AIRCRAFT TARGET_NAME  …  TAIL_FUZE CALCULATED_BOMBLOAD_LBS  \
0               NaN  Changdo-ri  …  Non-delay                 12000.0
1               NaN         NaN  …  Non-delay                  4000.0
2               NaN         NaN  …  Non-delay                  8000.0
3               NaN        Anju  …  Non-delay                 16000.0
4               NaN     Hamhung  …  Non-delay                 16000.0

  RECORD_SOURCE DAY  MONTH  YEAR MISSION_DATE    City/Town  \
0        EXETER   1      6  1951   1951-06-01   Changdo-ri
1        EXETER   1      6  1951   1951-06-01  North Korea
2        EXETER   1      6  1951   1951-06-01    Seosam-ri
3        EXETER   1      6  1951   1951-06-01      Anju-si
4        EXETER   1      6  1951   1951-06-01   Hamhung-si

                                    geometry In_Land
0                    POINT (127.66974 38.49881)    True
1  POINT (126.78196876190476 38.93594352380953)    True
2                    POINT (125.50653 39.66879)    True
3                    POINT (125.66717 39.60215)    True
4                    POINT (127.56091 39.91217)    True

[5 rows x 38 columns]
```

[244]:
```
# Contar el número de valores "Unknown" en la columna "City/Town"
unknown_count = df_bombardeos_processed['City/Town'].value_counts().
 ↪get('Unknown', 0)

print(f"Número de valores 'Unknown' en la columna 'City/Town': {unknown_count}")
```

```
Número de valores 'Unknown' en la columna 'City/Town': 1
```

[245]:
```
df_bombardeos_processed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11052 entries, 0 to 11051
Data columns (total 38 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   ROW_NUMBER                11052 non-null  int64
 1   MISSION_NUMBER            11042 non-null  object
 2   OP_ORDER                  11043 non-null  object
 3   UNIT                      11040 non-null  object
 4   AIRCRAFT_TYPE_MDS         10428 non-null  object
 5   NBR_ATTACK_EFFEC_AIRCRAFT 11012 non-null  float64
```

```
 6   SORTIE_DUPE                6663 non-null   float64
 7   NBR_ABORT_AIRCRAFT         379 non-null    float64
 8   NBR_LOST_AIRCRAFT          32 non-null     object
 9   TARGET_NAME                6036 non-null   object
 10  TGT_TYPE                   7758 non-null   object
 11  SOURCE_UTM_JAPAN_B         12 non-null     object
 12  SOURCE_TGT_UTM             8564 non-null   object
 13  TGT_MGRS                   7534 non-null   object
 14  TGT_LATITUDE_WGS84         11052 non-null  float64
 15  TGT_LONGITUDE_WGS84        11052 non-null  float64
 16  SOURCE_TGT_LAT             592 non-null    object
 17  SOURCE_TGT_LONG            585 non-null    object
 18  NBR_OF_WEAPONS             9956 non-null   object
 19  WEAPONS_TYPE               9958 non-null   object
 20  BOMB_SIGHTING_METHOD       8108 non-null   object
 21  TOTAL_BOMBLOAD_IN_LBS      5 non-null      float64
 22  TOT                        1559 non-null   object
 23  MISSION_TYPE               9925 non-null   object
 24  ALTITUDE_FT                9219 non-null   object
 25  CALLSIGN                   1 non-null      object
 26  BDA                        6680 non-null   object
 27  NOSE_FUZE                  5208 non-null   object
 28  TAIL_FUZE                  4771 non-null   object
 29  CALCULATED_BOMBLOAD_LBS    9876 non-null   float64
 30  RECORD_SOURCE              11052 non-null  object
 31  DAY                        11052 non-null  int64
 32  MONTH                      11052 non-null  int64
 33  YEAR                       11052 non-null  int64
 34  MISSION_DATE               11052 non-null  object
 35  City/Town                  11052 non-null  object
 36  geometry                   11052 non-null  object
 37  In_Land                    11052 non-null  bool
dtypes: bool(1), float64(7), int64(4), object(26)
memory usage: 3.1+ MB
```

[247]:
```
from geopy.distance import geodesic

# Filtrar el registro "Unknown"
unknown_row = df_bombardeos_processed[df_bombardeos_processed['City/Town'] ==␣
 ↪'Unknown']

# Filtrar registros válidos (sin "Unknown")
valid_rows = df_bombardeos_processed[df_bombardeos_processed['City/Town'] !=␣
 ↪'Unknown']

# Verificar si hay registros válidos para calcular la distancia
if not unknown_row.empty and not valid_rows.empty:
```

```
    # Extraer las coordenadas del registro "Unknown"
    unknown_coords = (unknown_row['TGT_LATITUDE_WGS84'].values[0],␣
 ↪unknown_row['TGT_LONGITUDE_WGS84'].values[0])

    # Calcular la distancia a cada registro válido
    valid_rows['distance'] = valid_rows.apply(
        lambda row: geodesic(unknown_coords, (row['TGT_LATITUDE_WGS84'],␣
 ↪row['TGT_LONGITUDE_WGS84'])).meters,
        axis=1
    )

    # Encontrar el registro más cercano
    closest_row = valid_rows.loc[valid_rows['distance'].idxmin()]

    # Reemplazar el valor "Unknown" con la ubicación del registro más cercano
    df_bombardeos_processed.loc[unknown_row.index, 'City/Town'] =␣
 ↪closest_row['City/Town']

# Guardar el DataFrame actualizado
df_bombardeos_processed.to_csv('./data/df_bombardeos_procesado.csv',␣
 ↪index=False)

print(f"Último valor 'Unknown' reemplazado con: {closest_row['City/Town']}")
```

```
Último valor 'Unknown' reemplazado con: Water
```

[248]:
```
file_path = './data/df_bombardeos_procesado.csv'
df_bombardeos_procesado_completo = pd.read_csv(file_path, sep=',')
df_bombardeos_procesado_completo.head()
```

[248]:

	ROW_NUMBER	MISSION_NUMBER	OP_ORDER	UNIT	AIRCRAFT_TYPE_MDS	\
0	2	433	174-51	98th Bomb Wing	B-29	
1	3	433	174-51	307th Bomb Wing	B-29	
2	4	433	174-51	307th Bomb Wing	B-29	
3	5	433	174-51	98th Bomb Wing	B-29	
4	6	433	174-51	98th Bomb Wing	B-29	

	NBR_ATTACK_EFFEC_AIRCRAFT	SORTIE_DUPE	NBR_ABORT_AIRCRAFT	\
0	1.0	NaN	NaN	
1	NaN	1.0	NaN	
2	1.0	1.0	NaN	
3	1.0	NaN	NaN	
4	1.0	NaN	NaN	

	NBR_LOST_AIRCRAFT	TARGET_NAME	…	TAIL_FUZE	CALCULATED_BOMBLOAD_LBS	\
0	NaN	Changdo-ri	…	Non-delay	12000.0	
1	NaN	NaN	…	Non-delay	4000.0	

```
2               NaN         NaN  …  Non-delay                  8000.0
3               NaN        Anju  …  Non-delay                 16000.0
4               NaN     Hamhung  …  Non-delay                 16000.0

  RECORD_SOURCE DAY  MONTH  YEAR MISSION_DATE    City/Town  \
0        EXETER   1      6  1951   1951-06-01   Changdo-ri
1        EXETER   1      6  1951   1951-06-01  North Korea
2        EXETER   1      6  1951   1951-06-01    Seosam-ri
3        EXETER   1      6  1951   1951-06-01      Anju-si
4        EXETER   1      6  1951   1951-06-01   Hamhung-si

                                    geometry In_Land
0                    POINT (127.66974 38.49881)    True
1  POINT (126.78196876190476 38.93594352380953)    True
2                    POINT (125.50653 39.66879)    True
3                    POINT (125.66717 39.60215)    True
4                    POINT (127.56091 39.91217)    True

[5 rows x 38 columns]
```

[251]:
```
# Contar el número de valores "Unknown" en la columna "City/Town"
unknown_count = df_bombardeos_procesado_completo['City/Town'].value_counts().
 ↪get('Unknown', 0)

print(f"Número de valores 'Unknown' en la columna 'City/Town': {unknown_count}")
```

```
Número de valores 'Unknown' en la columna 'City/Town': 0
```

[250]:
```
df_bombardeos_procesado_completo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11052 entries, 0 to 11051
Data columns (total 38 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   ROW_NUMBER                 11052 non-null  int64
 1   MISSION_NUMBER             11042 non-null  object
 2   OP_ORDER                   11043 non-null  object
 3   UNIT                       11040 non-null  object
 4   AIRCRAFT_TYPE_MDS          10428 non-null  object
 5   NBR_ATTACK_EFFEC_AIRCRAFT  11012 non-null  float64
 6   SORTIE_DUPE                6663 non-null   float64
 7   NBR_ABORT_AIRCRAFT         379 non-null    float64
 8   NBR_LOST_AIRCRAFT          32 non-null     object
 9   TARGET_NAME                6036 non-null   object
 10  TGT_TYPE                   7758 non-null   object
 11  SOURCE_UTM_JAPAN_B         12 non-null     object
```

```
 12  SOURCE_TGT_UTM              8564 non-null   object
 13  TGT_MGRS                    7534 non-null   object
 14  TGT_LATITUDE_WGS84          11052 non-null  float64
 15  TGT_LONGITUDE_WGS84         11052 non-null  float64
 16  SOURCE_TGT_LAT              592 non-null    object
 17  SOURCE_TGT_LONG             585 non-null    object
 18  NBR_OF_WEAPONS              9956 non-null   object
 19  WEAPONS_TYPE                9958 non-null   object
 20  BOMB_SIGHTING_METHOD        8108 non-null   object
 21  TOTAL_BOMBLOAD_IN_LBS       5 non-null      float64
 22  TOT                         1559 non-null   object
 23  MISSION_TYPE                9925 non-null   object
 24  ALTITUDE_FT                 9219 non-null   object
 25  CALLSIGN                    1 non-null      object
 26  BDA                         6680 non-null   object
 27  NOSE_FUZE                   5208 non-null   object
 28  TAIL_FUZE                   4771 non-null   object
 29  CALCULATED_BOMBLOAD_LBS     9876 non-null   float64
 30  RECORD_SOURCE               11052 non-null  object
 31  DAY                         11052 non-null  int64
 32  MONTH                       11052 non-null  int64
 33  YEAR                        11052 non-null  int64
 34  MISSION_DATE                11052 non-null  object
 35  City/Town                   11052 non-null  object
 36  geometry                    11052 non-null  object
 37  In_Land                     11052 non-null  bool
dtypes: bool(1), float64(7), int64(4), object(26)
memory usage: 3.1+ MB
```

6

---

7 VISUALIZACIONES - ANÁLISIS

[254]:
```
# CANTIDAD DE BOMBARDEOS POR CIUDAD O UBICACIÓN
pd.set_option('display.max_rows', None)  # Mostrar todas las filas
pd.set_option('display.max_columns', None)  # Mostrar todas las columnas, si es␣
 ↪necesario

# Volver a mostrar las estadísticas completas
bombing_stats = df_bombardeos_procesado_completo['City/Town'].value_counts()
print(bombing_stats)
```

```
City/Town
North Korea                  1756
Yangdok County                706
```

Hamhung-si	657
P'yŏngyang	370
Songnim-si	319
Nampo	299
Kowon County	280
Sunchon-si	249
Kaesong	217
Anju-si	201
Sariwon-si	126
Wonsan-si	120
Pukchang County	114
Chongju-si	112
Huichon-si	107
Chongjin-si	95
Bangsan-myeon	91
Hoechang-up	91
Dong-myeon	84
Shinsungcheon Labor Union	77
Kaechon	68
Sonchon-up	60
Kanggye-si	52
Sinanju	52
Sohung-up	51
geography	51
Koksan-up	51
Pyongsan County	49
Ryongcheol Labor Union	48
Steel Dong	48
Haeju-si	48
Unyang-dong	46
Suwon-dong	46
Shinsung-ri	45
Songchon-up	45
Pyongsong-si	43
Sinyang-up	43
Tokchon-si	41
Unpa-up	40
Seokam-ri	40
Lingyunri	40
Chojangri	38
Churan Jeonri	37
Sunan District	36
Pukchong-up	36
Maengjung Labor Union	35
Haksan Labor Union	35
Hoeyang-up	34
Pyonggang-up	32
Kilju-up	32

Wonbuk-ri	32
Sinuiju-si	31
Yonggwang-up	30
Tongsin-up	30
Sukchon County	29
Kusong-si	28
Gyesok-ri	28
Sepo-up	28
Jangdong-ri	27
Goeup-ri	26
Paegwŏn	26
Daedeok-ri	26
Chongdu-ri	26
Changdo-ri	26
Hwacheon	25
Deokheung-ri	25
Sinseo-myeon	25
Haean-myeon	24
Lihari	24
Kimhwa-up	24
Jidong-ri	23
Tosan-up	23
Songjeong-ri	23
Sinhung-up	23
Kumya-up	23
Kosan-up	22
Soksa-ri	22
Hwangju-up	22
Wondong-ri	22
Sangseo	22
Ochon-ri	21
Ryongdamnodongja-gu	21
Jakdong-ri	21
Saengyang-ri	20
Changyon-up	20
Suphung-rodongjagu	20
Gownley	20
Ryongpo-ri	20
Shinchang-ri	19
Hakbang Labor Union	18
Pakchon-up	18
Chang Gaeri	18
Sinphyong-up	18
Okpyeong	18
Ryongbanri	18
Anak-up	18
Ipo-ri	18
Sudong	18

Jung-myeon	18
Flexible	18
Sinchang-ri	17
Shinsang-ri	17
Ryonghyon-ri	17
Tanchon-si	17
Kyongsong County	17
Kubong-ri	17
Pongsan-up	16
Ryongdang-ri	16
Munchon-si	15
Seohwa-myeon	15
Tapgeo-ri	14
Wolseong-ri	14
Ripsok-ri	14
Water	14
Gaeryeonri	14
Taekinri	14
Ichon-up	14
Wonnam-ri	14
Jeokdong-ri	14
Namcheonri	14
Daemun-ri	14
Ryongsu Labor Union	14
Wangjing-myeon	14
Jungsamri	13
Inam-ri	13
Unheung-ri	13
Joyang-ri	13
Otan-ri	13
Mundong-ri	13
Hwajeon Labor Union	13
Jiseong-ri	13
Yangsari	13
Punghyeon-ri	12
Dongnim-ri	12
Unsan-up	12
Cheon Seong Labor Union	12
Apgangnodongja-gu	12
Yongam-ri	12
Oho-ri	12
The deceased worker's ward	12
Top plate	12
Wonseori	12
Hasong-ri	12
Baekrosanri	12
Pangyo-up	12
Kuup-ri	12

Deokryun-ri	12
Jehyeon-ri	11
Jeongsan-ri	11
Sinpo-si	11
Chorwon-up	11
My Gang-ri	11
Street noise	11
Eungok Labor Union	11
Pyongwon-up	11
Sangchori	11
Dochan-ri	10
Pocheon-si	10
Pangyo-ri	10
Geunnam-myeon	10
Singye-up	10
Inheung-ri	10
Yongheung-ri	10
Holdong Labor Union	10
Anbyon-up	10
Puap-ri	10
Jungnam-ri	10
Chukdong-ri	10
Forward vibration	10
Foguri	9
Tukchang	9
Chunghwa-up	9
Ryu Jeong-ri	9
Hoesan-ri	9
Nyongwon-up	9
Gamdun-ri	9
Seongsan-ri	9
Paekhyon-ri	9
Sinpung-ri	9
Forwarding Management	9
Majang-ri	9
Moonsamri	9
Jaedong Labor Union	9
Daegok-ri	8
Seochon-ri	8
Donghori	8
Taechon-up	8
Ryongcheon-ri	8
Hoeun-ri	8
Approval	8
Yoo Seung-ri	8
Cheonam-ri	8
Chonnae-up	8
Moonbong-ri	8

Dokgeom-ri	8
Kuyon-ri	8
Seungjeon-ri	8
Sudeok-ri	8
Onjin County	8
Kyongdo-ri	8
Misong-ri	8
Baekhak-myeon	8
Rason	8
Moonseong-ri	8
Gubong Labor Union	8
Shinwon-ri	8
Daegunri	8
Mukbang-dong	8
Induri	8
Doeumri	8
Jeongbong-ri	7
Taejon-ri	7
Kahak-ri	7
Giyang-ri	7
Unbong-ri	7
Hongwon-up	7
Sanghari	7
Sentence Lee	7
Deokam-ri	7
Namcheon Labor Union	7
Sinchon-up	7
Sogon-ri	7
Ssangryong-ri	7
Daebaekri	7
Bukjin Labor Union	7
Yongjin-dong	7
Yongjeon-ri	7
Hukgyo-ri	7
Construction site	7
Kapsan-up	7
Hamju-up	7
Tomil-ri	7
Suhapri	7
Naemun-ri	7
Yangcheonseori	7
Miram-ri	7
Daecheong-ri	6
Unryul-up	6
Cholsan-up	6
Songnam Labor Union	6
Tongsan-ri	6
Geumpyeong-ri	6

Jangsan-ri	6
Limheung-ri	6
Bukpori	6
Kujang-up	6
Dopyeong-ri	6
Daecheonri	6
Daeyuro Labor Union	6
Danghyeon-ri	6
Namyangni	6
Gwangheung-ri	6
Gosan-ri	6
Civilization	6
Hwasan-ri	6
Mundok-up	6
Goseong-ri	6
Rotan-ri	6
Bonghari	6
Hoam-ri	6
Haptan-ri	6
Geonji-dong	6
Wonam-ri	6
Mundeungri	6
Daeheung-ri	6
Ryangwolli	6
Sujeon-ri	6
Pangmok-ri	6
Yanggu-eup	5
Uiju-up	5
Komsa-ri	5
Junggangri	5
Ryŏnpho-ri	5
Geunbuk-myeon	5
Okdong-ri	5
Jinseo-ri	5
Ryonghak-ri	5
Sutae-ri	5
Buk-myeon	5
Bangpo-ri	5
Midang-ri	5
Langhari	5
Kalhyon-ri	5
Mountain geography	5
Kumgang-up	5
Cultural Center	5
Aptong-ni	5
Union Lee	5
Sinch'ang	5
Gisan-ri	5

Bokgye-ri	5
Geoncheon-ri	5
Seo-myeon	5
Cheorwon-eup	5
Saemaeul-ri	5
Obong-ri	5
Chotan-ri	5
Jeongpyeong-ri	5
Joyang-dong	5
Bokmanri	5
Gwanseong-ri	5
Anbong-ri	5
Gyoju-ri	5
Baeksan-ri	5
Oehak-ri	5
Jiseok-ri	5
Sungap-ri	5
Buksinhyeon-ri	5
Gandong	5
Wolha-ri	5
Songcheon-ri	5
Lee Cheon-ri	4
Baehwa-ri	4
Songdong-ri	4
Yongpo-ri	4
Pongryon-ri	4
Munyang-ri	4
Girin-myeon	4
Geumpung-ri	4
Byeolsang-ri	4
Kwaksan-up	4
Myeokmi Labor Union	4
Hwaam-ri	4
Pansok-ri	4
Baekilri	4
Geum Pyeongni	4
Daema-ri	4
Hwatong-ri	4
Hyeonnae-myeon	4
Jeongok-eup	4
Hwanggang-ri	4
Sambangri	4
Unjon-up	4
Maenghari	4
Picnic	4
Aparodongjagu	4
Naesan-ri	4
Daeam-ri	4

Namwon-ri	4
Songjeon-ri	4
Seongdori	4
Hwacheon Labor Union	4
Sangbuk-dong-ri	4
Pomak-ri	4
Yongmun Labor Union	4
Jigyeong-ri	4
Paju-si	4
Naedong-ri	4
Donggori	4
Dongyang-ri	3
Pungcheon-ri	3
Ryukwa-ri	3
Haebang-ni	3
Chongsu-rodongjagu	3
Deungdae-ri	3
Dongsan Labor Union	3
Jeongyeon-ri	3
Naedae-ri	3
Kumbu-ri	3
Hahoe-ri	3
Odong-ri	3
Seorim-ri	3
Song Se-ri	3
Sa Pyeongni	3
Chongpyong-up	3
Jangcheon-ri	3
Jungheung-ri	3
Lee Su-deok-ri	3
Nam-myeon	3
Pocheon-ri	3
Chongdong-ri	3
Ryudaepo-ri	3
Jungsan-ri	3
Jangnam-myeon	3
Tongga-ri	3
Sanchamri	3
Changpung-up	3
Chungsan-up	3
Deok-eum-ri	3
Beopsu-ri	3
Misan-myeon	3
Geundong-ri	3
Jang Jae-ri	3
Jangheung-ri	3
Shinsang Labor Union	3
Geumcheol-ri	3

Orang-ri	3
Pungjeon-ri	3
Sanae-myeon	3
Jeongok-ri	3
Wondong-myeon	3
Guryong-ri	3
New Physiology	3
Hoeryong-si	3
Gunhan-ri	3
Tongchon-up	3
Songgo-ri	3
Kumchon-up	3
Ryongnam-ri	3
Namwon-si	3
Yeonsam-ri	3
Hancheon-dong	3
Colonel Lee	3
Gushan	2
Yongdaeri	2
Guktojeongjungang-myeon	2
Lip stone	2
Separi	2
Kamdul-li	2
Nyongbyon-up	2
Namsa-rodongjagu	2
Song Sam-ri	2
Hyangsan-up	2
Ryanggyo-ri	2
Ryangchaek-rodongjagu	2
Wacho-ri	2
Sadong-ri	2
Pyokdong-up	2
Dongsam-ri	2
Yomju-up	2
Komam-ri	2
Hongnam-ri	2
Bongsan-ri	2
Jangsa-ri	2
Hasikjom-ri	2
Suncheon-ri	2
Junggang-ri	2
Gwangjeon-ri	2
Kijeongri	2
Gwangbok-dong	2
Jangseong-ri	2
Dashiqiao City	2
Hongwon-ri	2
Woram-ri	2

City Labor Union	2
Rungdong-ri	2
Ryongsan-ri	2
Unpori	2
Sangap-ri	2
Yonsan-up	2
Yeonhong-ri	2
Sujeon Labor Union	2
Kasung-ri	2
Chahoro Labor Union	2
Seongpyeong-ri	2
Ryong-am-ri	2
Chowon-ri	2
Songhwari	2
Seonam-ri	2
Jangchon Labor Union	2
Kosong-up	2
Umiri	2
Wafangdian City	2
Taephyong-ri	2
Munhye-ri	2
Rosang-ri	2
Yeomtan-ri	2
Kumphung-ri	2
Sudong-myeon	2
Sinphung-ri	2
Yongsan-ri	2
Jeokseong-myeon	2
Sangeum-ri	2
Gwanghwari	2
Rimokri	2
Gaecheon-ri	2
Choseo-ri	2
Gangdon-ri	2
Hyangdong-ri	2
Sojeong-ri	2
Inhung-ri	2
Sokdam-ri	2
Galmal-eup	2
Geumbong-ri	2
Ohyon-ri	2
Wolhyeon-ri	2
Self-operation	2
Sangyang-ri	2
Munsan-ri	2
Diagonal	2
Kumchon-ri	2
Juchon-ri	2

Guseong-ri	2
Ma Bang-ri	2
Go Yeon-ri	2
Sokcho-si	2
Geumran-ri	2
Wonhari	2
Sacheongri	2
Dongsong-eup	2
Seohwari	2
Seokbong-ri	2
Bongrae-ri	2
Buraesan Labor Union	2
Bud	2
Hwanhyeon-ri	2
Info Labor Union	2
Rim Seong-ri	2
Geumok-ri	2
Sangsulli	2
Yangsa-myeon	2
Yaksuri	2
Resources	2
Hyon-ri	2
Tongphyong-ri	2
Õgung Workers District	2
Eoryong-ri	2
Jeongdong-ri	2
Songpo-ri	2
Geumgok-ri	2
Raw interest rate	2
Sindang-ri	2
Ayang-ri	2
Cheongok-ri	2
Bongheung-ri	2
Outer frost	2
Palwon Labor Union	2
Eumnae-ri	2
Cheonsam-ri	2
Hwararodongja-gu	2
Chonma-up	2
Songsan-ri	2
Kwangmyong-ri	2
Gwanbong-ri	2
Gwansang-ri	2
Jinhyeon-ri	2
Green River Village	1
Songgang-ri	1
Onjong-ri	1
Phungsong-ri	1

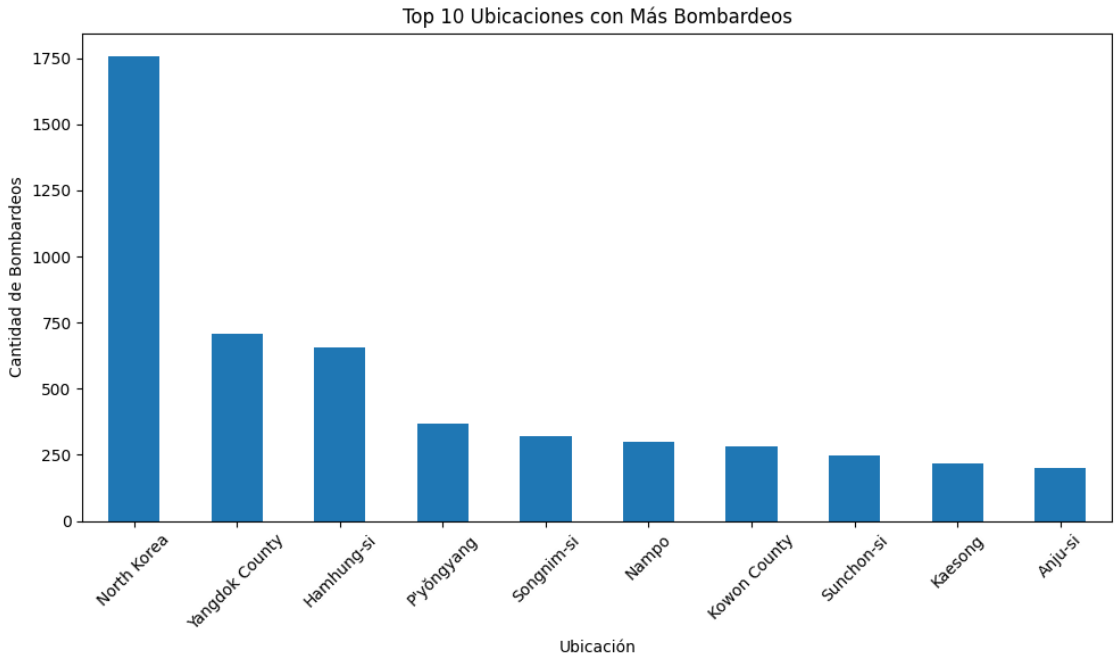
Sinsi-ri	1
Moon Ok-ri	1
Yugok-ri	1
Cheolsan-ri	1
Jasan-ri	1
Choksan-ri	1
Suan-up	1
Sehyeon-ri	1
Shindong-ri	1
Hagyori	1
Geumseong-ri	1
Oji-ri	1
Unchon-up	1
Toseong-ri	1
Hanam-ri	1
Seowon-ri	1
Ryongyeon-ri	1
Jungcheon-ri	1
Daewi-ri	1
Shinpung-ri	1
Hwangryong-ri	1
Jungse-ri	1
Seondeok-ri	1
Jinam-ri	1
Gomun-ri	1
Accounting	1
Yuli	1
Munsan-eup	1
Unjon-ri	1
Pamgashi	1
Hwayari	1
Bukbun-ri	1
Gangneung-si	1
Amjeong-ri	1
Apdong-ri	1
Yangji-ri	1
Large fish	1
Mingyue	1
Gwisang-ri	1
Hwachon-ri	1
Geojin-eup	1
Myeongu-ri	1
Bank reorganization	1
Hyondong-ri	1
Gapyeong	1
Yangchon	1
Recommended	1
Haepori	1

Tsushima	1
Baekyang-ri	1
Masan-ri	1
Namjeong-ri	1
Samsung-ri	1
Sangmasan-ri	1
Water vapor	1
An Miri	1
Dangsan-ri	1
Judun-ri	1
Honam-ri	1
Daeryong-ri	1
Oktok-ri	1
Jinheung-ri	1
Samdori	1
Okhyun-ri	1
Ch'ontae-ri	1
Yueup-ri	1
Sayo-ri	1
Naehyeon-ri	1
A hundred horses	1
Cheongjeong-ri	1
Sinbok-ri	1
Standing Workers' Union	1
Pyoksong-up	1
Yonggyo-ri	1
Rotary wheel	1
Deokchon-ri	1
Daeseong-ri	1
Thousand-year-old	1
Bongpo-ri	1
Byeolhari	1
Ryangsa-ri	1
Sacheon-si	1
Pukjom-ri	1
Singye-ri	1
Donghari	1
Guhang-ri	1
Igil-ri	1
Samsa-ri	1
Jeonsan-ri	1
Seopyeong-ri	1
Dongchang-ri	1
Dongsifangtai	1
Yang Chun-ri	1
Alildong	1
Mungyeong-si	1
Songgan-up	1

Gacheon-ri	1
Gui Rak Ri	1
Odeok-ri	1
Namjung-ri	1
Jeonseungri	1
Songryeon-ri	1
Gwanjeon-ri	1
Taepyeong-ri	1
Shinpo-ri	1
Ungok-ri	1
Chuncheon-si	1
Gueup-ri	1
Namsan-ri	1
Dokjeong-ri	1
Gwanu-ri	1
Gadan-ri	1
Wolbong-ri	1
Dongsari	1
Cheonsuri	1
Red Star	1
Ryongseong-ri	1
Santhan-ri	1
Bongui-ri	1
Wonduk-ri	1
Jang Gook-li	1
Ganseong-eup	1
Thousand Horses	1
Gwanin	1
Hanam	1
Sinbuk	1
Songchon-ri	1
Lakeside Workers' Union	1
Three thousand miles	1
Sangduri	1
Taesuk-ri	1
Inje-eup	1
Geundong-myeon	1
Reasonable price	1
Sudong-ri	1
Gusan-ri	1
Hachori	1
Miyang-myeon	1
Jonggok-ri	1
Middle East	1
Long-term interest rate	1
Ushumun	1
Seosam-ri	1
Oncheon-ri	1

```
Ryeonho-dong                    1
Yangjiri                       1
Phungmi-ri                     1
Name: count, dtype: int64
```

[ ]:
```
bombing_stats.head(10).plot(kind='bar', figsize=(10, 6))
plt.title("Top 10 Ubicaciones con Más Bombardeos")
plt.xlabel("Ubicación")
plt.ylabel("Cantidad de Bombardeos")
plt.xticks(rotation=45)
plt.tight_layout();                    # todo es Corea del Norte
```



[265]:
```
unique_values = df_bombardeos_procesado_completo['City/Town'].unique()
unique_values.tolist()
```

[265]:
```
['Changdo-ri',
 'North Korea',
 'Seosam-ri',
 'Anju-si',
 'Hamhung-si',
 'Pyongsan County',
 'Ushumun',
 'Green River Village',
 'Colonel Lee',
 'Hwacheon',
 'Baekrosanri',
```

```
'Komsa-ri',
'Sinseo-myeon',
'Sangseo',
'Jinhyeon-ri',
'Bangsan-myeon',
'Buk-myeon',
'Mungyeong-si',
'Geunnam-myeon',
'Huichon-si',
'Chongju-si',
'Sentence Lee',
'Kumya-up',
'Jungnam-ri',
'Dongsan Labor Union',
'Kaesong',
'Chojangri',
'Songgan-up',
'Gacheon-ri',
'Sariwon-si',
'Otan-ri',
'Ryonghak-ri',
'Seongsan-ri',
'Jung-myeon',
'Dong-myeon',
'Seohwa-myeon',
'Geundong-myeon',
'Yanggu-eup',
'Nampo',
'Paju-si',
'Reasonable price',
'Paekhyon-ri',
'Jeokdong-ri',
'Cheonam-ri',
'Pyonggang-up',
'Haebang-ni',
'Junggangri',
'Junggang-ri',
'Bokmanri',
'Cheorwon-eup',
'Jungsamri',
'Seo-myeon',
'Amjeong-ri',
'Geoncheon-ri',
'Apdong-ri',
'Yangji-ri',
'Daema-ri',
'Dongsong-eup',
```

```
'Lee Su-deok-ri',
'Bokgye-ri',
'Tomil-ri',
'Large fish',
'Tanchon-si',
'Jungse-ri',
'Okdong-ri',
'Woram-ri',
'Odeok-ri',
'Geunbuk-myeon',
'Jeongyeon-ri',
'Hongwon-ri',
'Naedae-ri',
'Odong-ri',
'Gwanu-ri',
'Gadan-ri',
'Igil-ri',
'Samsa-ri',
'Haean-myeon',
'Chongdu-ri',
'Wonnam-ri',
'Sinchang-ri',
'Eumnae-ri',
'Puap-ri',
'Yugok-ri',
'Haeju-si',
'Sinchon-up',
'Kimhwa-up',
'Yongheung-ri',
'Pomak-ri',
'Kubong-ri',
'Kahak-ri',
'Red Star',
'Ryongdang-ri',
'Yongpo-ri',
'Unpa-up',
'Ryu Jeong-ri',
'Chorwon-up',
'Ichon-up',
'Ryongseong-ri',
'Song Se-ri',
'Songcheon-ri',
'Sudong-ri',
'Chukdong-ri',
'Hasikjom-ri',
'Gusan-ri',
'Lihari',
```

```
'Deungdae-ri',
'Sogon-ri',
'Gui Rak Ri',
'Wondong-ri',
'Songnim-si',
"P'yŏngyang",
'Hongwon-up',
'Sunchon-si',
'Kowon County',
'Sunan District',
'Hoeyang-up',
'Wonbuk-ri',
'My Gang-ri',
'Hakbang Labor Union',
'Sinpo-si',
'Saengyang-ri',
'Seongdori',
'Street noise',
'Sutae-ri',
'Songgo-ri',
'Sohung-up',
'Anak-up',
'Sukchon County',
'Ochon-ri',
'Sangbuk-dong-ri',
'Tongsan-ri',
'Gunhan-ri',
'Daeam-ri',
'Tosan-up',
'Shinsung-ri',
'Yangsari',
'Suhapri',
'Mundong-ri',
'Ipo-ri',
'Hasong-ri',
'Munhye-ri',
'Kumgang-up',
'Mountain geography',
'Ryonghyon-ri',
'Kumchon-up',
'Pangyo-ri',
'Langhari',
'Seonam-ri',
'Onjin County',
'Kosong-up',
'Umiri',
'Sinpung-ri',
```

```
'Wolseong-ri',
'Yangcheonseori',
'Ripsok-ri',
'Majang-ri',
'Flexible',
'Miram-ri',
'Tapgeo-ri',
'Hwangju-up',
'Yonggyo-ri',
'Wondong-myeon',
'Pyongwon-up',
'Sangmasan-ri',
'Pocheon-si',
'Oho-ri',
'Wolhyeon-ri',
'Naesan-ri',
'Shinsungcheon Labor Union',
'Kaechon',
'Kuyon-ri',
'Aparodongjagu',
'New Physiology',
'Forwarding Management',
'Sinanju',
'Geumpyeong-ri',
'Kalhyon-ri',
'Top plate',
'Jangchon Labor Union',
'Songjeong-ri',
'Seokam-ri',
'Jeongsan-ri',
'Moonbong-ri',
'Hyangdong-ri',
'Dokgeom-ri',
'Choseo-ri',
'Water',
'Changyon-up',
'Gangdon-ri',
'Gaecheon-ri',
'Yangdok County',
'Singye-up',
'geography',
'Foguri',
'Mundeungri',
'Churan Jeonri',
'Wonsan-si',
'Hwanggang-ri',
'Jeongok-ri',
```

```
'Orang-ri',
'Picnic',
'Sudong',
'Yongam-ri',
'Hwasan-ri',
'Maengjung Labor Union',
'Rason',
'Jakdong-ri',
'Sinyang-up',
'Unjon-up',
'Naemun-ri',
'Munchon-si',
'Geundong-ri',
'Misan-myeon',
'Baekyang-ri',
'Sepo-up',
'Sambangri',
'Masan-ri',
'Pongsan-up',
'Namjeong-ri',
'Midang-ri',
'Apgangnodongja-gu',
'Bangpo-ri',
'Ryongdamnodongja-gu',
'Chotan-ri',
'Ryongpo-ri',
'Koksan-up',
'Baekilri',
'Seondeok-ri',
'Pansok-ri',
'Jungsan-ri',
'Ryudaepo-ri',
'Donghori',
'Munsan-ri',
'Soksa-ri',
'Chongdong-ri',
'Wangjing-myeon',
'Baekhak-myeon',
'Geonji-dong',
'Seohwari',
'Songchon-up',
'Kwaksan-up',
'Go Yeon-ri',
'Ma Bang-ri',
'Mundok-up',
'Hwangryong-ri',
'Guseong-ri',
```

```
'Hoechang-up',
'Deokryun-ri',
'Hoesan-ri',
'Sokcho-si',
'Tongga-ri',
'Hamju-up',
'Chunghwa-up',
'Aptong-ni',
'Jangnam-myeon',
'Juchon-ri',
'Hukgyo-ri',
'Geumran-ri',
'Okpyeong',
'Ryongbanri',
'Daewi-ri',
'Resources',
'Misong-ri',
'Jasan-ri',
'Hyon-ri',
'Choksan-ri',
'Jeongdong-ri',
'Eoryong-ri',
'Hwaam-ri',
'Geumgok-ri',
'Suan-up',
'Sehyeon-ri',
'Songpo-ri',
'Munsan-eup',
'Unjon-ri',
'Nam-myeon',
'Bongrae-ri',
'Pamgashi',
'Kosan-up',
'Geumpung-ri',
'Hwayari',
'Bukbun-ri',
'Girin-myeon',
'Yonggwang-up',
'Pukchong-up',
'Kapsan-up',
'Kilju-up',
'Chongjin-si',
'Anbong-ri',
'Namcheonri',
'Daegok-ri',
'Steel Dong',
'Sonchon-up',
```

```
'Hwanhyeon-ri',
'Info Labor Union',
'Gapyeong',
'Kusong-si',
'Yangchon',
'Kanggye-si',
'Changpung-up',
'Recommended',
'Oehak-ri',
'Rimokri',
'Haepori',
'Tsushima',
'Suwon-dong',
'Sojeong-ri',
'Munyang-ri',
'Beopsu-ri',
'Sanae-myeon',
'Kumchon-ri',
'Gosan-ri',
'Taechon-up',
'Diagonal',
'Songchon-ri',
'Pakchon-up',
'Bukjin Labor Union',
"Lakeside Workers' Union",
'Kamdul-li',
'Daebaekri',
'Three thousand miles',
'Guktojeongjungang-myeon',
'Namjung-ri',
'Kyongsong County',
'Unyang-dong',
'Jeonseungri',
'Songryeon-ri',
'Yongdaeri',
"Sinch'ang",
'Rungdong-ri',
'Gwanjeon-ri',
'Wolha-ri',
'Danghyeon-ri',
'Nyongbyon-up',
'Ryangchaek-rodongjagu',
'Namyangni',
'Wacho-ri',
'Jeongbong-ri',
'Sangduri',
'Hwajeon Labor Union',
```

```
'Taesuk-ri',
'Unryul-up',
'Dochan-ri',
'Sadong-ri',
'Inje-eup',
'Seorim-ri',
'Anbyon-up',
'Gwangbok-dong',
'Kijeongri',
'Gwangjeon-ri',
'Jangdong-ri',
'Pungcheon-ri',
'Song Sam-ri',
'Pyokdong-up',
'Santhan-ri',
'Bongui-ri',
'Wonduk-ri',
'Separi',
'Jang Gook-li',
'Ganseong-eup',
'Thousand Horses',
'Gwanin',
'Hanam',
'Sinbuk',
'Tokchon-si',
'Bukpori',
'Sinhung-up',
'Jinseo-ri',
'Shinchang-ri',
'Taepyeong-ri',
'Ryangwolli',
'Jeongok-eup',
'Hyeonnae-myeon',
'Shinpo-ri',
'Shinwon-ri',
'Ungok-ri',
'Chuncheon-si',
'Hwatong-ri',
'Sinuiju-si',
'Gueup-ri',
'Namsan-ri',
'Dokjeong-ri',
'City Labor Union',
'Hahoe-ri',
'Gisan-ri',
'Pangmok-ri',
'Deokchon-ri',
```

```
'Daeseong-ri',
'Thousand-year-old',
'Taephyong-ri',
'Bongpo-ri',
'Chahoro Labor Union',
'Byeolhari',
'Ryangsa-ri',
'Sacheon-si',
'Pukjom-ri',
'Sudong-myeon',
'Singye-ri',
'Donghari',
'Hoeryong-si',
'Guhang-ri',
'Tongchon-up',
'Geumcheol-ri',
'Yueup-ri',
'Seongpyeong-ri',
'Sayo-ri',
'Gamdun-ri',
'Doeumri',
'Naehyeon-ri',
'Ryong-am-ri',
'Pukchang County',
'Moonseong-ri',
'A hundred horses',
'Jangheung-ri',
'Jang Jae-ri',
'Guryong-ri',
'Chowon-ri',
'Bonghari',
'Cheongjeong-ri',
'Songhwari',
'Sinbok-ri',
'Gyoju-ri',
"Standing Workers' Union",
'Sungap-ri',
'Pyoksong-up',
'Water vapor',
'Jiseok-ri',
'An Miri',
'Dangsan-ri',
'Geumbong-ri',
'Judun-ri',
'Honam-ri',
'Galmal-eup',
'Daeryong-ri',
```

```
'Oktok-ri',
'Jinheung-ri',
'Shinsang Labor Union',
'Samdori',
'Inhung-ri',
'Okhyun-ri',
'Sokdam-ri',
"Ch'ontae-ri",
'Chungsan-up',
'Ryongcheon-ri',
'Hwachon-ri',
'Geojin-eup',
'Myeongu-ri',
'Bank reorganization',
'Daemun-ri',
'Hyondong-ri',
'Baeksan-ri',
'Pocheon-ri',
'Sacheongri',
'Haptan-ri',
'Seokbong-ri',
'Gaeryeonri',
'Ryongcheol Labor Union',
'Jinam-ri',
'Gomun-ri',
'Hoam-ri',
'Accounting',
'Gandong',
'Deokheung-ri',
'Byeolsang-ri',
'Goeup-ri',
'Yuli',
'Daegunri',
'Construction site',
'Oji-ri',
'Unchon-up',
'Rim Seong-ri',
'Seochon-ri',
'Toseong-ri',
'Sanchamri',
'Pangyo-up',
'Hanam-ri',
'Seowon-ri',
'Geumok-ri',
'Gwanseong-ri',
'Ryongyeon-ri',
'Sangsulli',
```

```
'Jungcheon-ri',
'Wonhari',
'Taejon-ri',
'Punghyeon-ri',
'Yaksuri',
'Yangsa-myeon',
'Kyongdo-ri',
'Shindong-ri',
'Obong-ri',
'Rotan-ri',
'Forward vibration',
'Goseong-ri',
'Sinphyong-up',
'Hagyori',
'Bud',
'Geumseong-ri',
'Myeokmi Labor Union',
'Taekinri',
'Shinpung-ri',
'Jehyeon-ri',
'Wonam-ri',
'Pyongsong-si',
'Yongjeon-ri',
'Dongnim-ri',
'Daecheong-ri',
'Unsan-up',
'Mukbang-dong',
'Yongjin-dong',
'Joyang-dong',
'Inam-ri',
'Induri',
'Jeongpyeong-ri',
'Shinsang-ri',
'Gangneung-si',
'Tongsin-up',
'Sujeon-ri',
'Unheung-ri',
'Mingyue',
'Jungheung-ri',
'Nyongwon-up',
'Tukchang',
'Gwisang-ri',
'Union Lee',
'Geum Pyeongni',
'Gyesok-ri',
'Buraesan Labor Union',
'Sangeum-ri',
```

```
'Cheon Seong Labor Union',
'Gubong Labor Union',
'Deok-eum-ri',
'Inheung-ri',
'Saemaeul-ri',
'Daeheung-ri',
'Gwanghwari',
'Lingyunri',
'Yoo Seung-ri',
'Moonsamri',
"The deceased worker's ward",
'Pungjeon-ri',
'Limheung-ri',
'Self-operation',
'Jaedong Labor Union',
'Songnam Labor Union',
'Sangchori',
'Samsung-ri',
'Maenghari',
'Dopyeong-ri',
'Jiseong-ri',
'Kujang-up',
'Sangyang-ri',
'Ohyon-ri',
'Rotary wheel',
'Songjeon-ri',
'Unpori',
'Ryongsan-ri',
'Yeonhong-ri',
'Sujeon Labor Union',
'Yonsan-up',
'Sangap-ri',
'Kasung-ri',
'Namwon-ri',
'Jangsan-ri',
'Wonseori',
'Wafangdian City',
'Sudeok-ri',
'Seungjeon-ri',
'Kumphung-ri',
'Chonnae-up',
'Hwacheon Labor Union',
'Gownley',
'Jeonsan-ri',
'Seopyeong-ri',
'Paegwŏn',
'Rosang-ri',
```

```
'Sinphung-ri',
'Yeomtan-ri',
'Dongchang-ri',
'Dongsifangtai',
'Deokam-ri',
'Jeokseong-myeon',
'Sanghari',
'Eungok Labor Union',
'Jidong-ri',
'Yongsan-ri',
'Joyang-ri',
'Namcheon Labor Union',
'Gushan',
'Yang Chun-ri',
'Suphung-rodongjagu',
'Alildong',
'Dashiqiao City',
'Haksan Labor Union',
'Kumbu-ri',
'Wolbong-ri',
'Dongsari',
'Lip stone',
'Chongpyong-up',
'Ryŏnpho-ri',
'Ryanggyo-ri',
'Cheonsuri',
'Hyangsan-up',
'Daedeok-ri',
'Chongsu-rodongjagu',
'Jigyeong-ri',
'Namsa-rodongjagu',
'Sa Pyeongni',
'Yongmun Labor Union',
'Ryongsu Labor Union',
'Hoeun-ri',
'Hachori',
'Songdong-ri',
'Daeyuro Labor Union',
'Suncheon-ri',
'Miyang-myeon',
'Jonggok-ri',
'Bongsan-ri',
'Baehwa-ri',
'Middle East',
'Jangsa-ri',
'Yomju-up',
'Lee Cheon-ri',
```

```
'Ryukwa-ri',
'Dongyang-ri',
'Naedong-ri',
'Cheolsan-ri',
'Raw interest rate',
'Holdong Labor Union',
'Namwon-si',
'Cultural Center',
'Sindang-ri',
'Jangcheon-ri',
'Donggori',
'Cholsan-up',
'Cheongok-ri',
'Ayang-ri',
'Ŏgung Workers District',
'Tongphyong-ri',
'Hongnam-ri',
'Uiju-up',
'Ryongnam-ri',
'Songsan-ri',
'Cheonsam-ri',
'Ssangryong-ri',
'Hwararodongja-gu',
'Civilization',
'Buksinhyeon-ri',
'Gwangheung-ri',
'Yeonsam-ri',
'Giyang-ri',
'Chang Gaeri',
'Outer frost',
'Bongheung-ri',
'Unbong-ri',
'Palwon Labor Union',
'Approval',
'Gwansang-ri',
'Gwanbong-ri',
'Pongryon-ri',
'Kwangmyong-ri',
'Songgang-ri',
'Chonma-up',
'Dongsam-ri',
'Daecheonri',
'Long-term interest rate',
'Komam-ri',
'Onjong-ri',
'Phungsong-ri',
'Hancheon-dong',
```

```
'Sinsi-ri',
'Moon Ok-ri',
'Kuup-ri',
'Jangseong-ri',
'Oncheon-ri',
'Ryeonho-dong',
'Yangjiri',
'Phungmi-ri']
```

8 Patrones Identificados en los Bombardeos durante la Guerra de Corea

8.1 Patrones Estratégicos Identificados

8.1.1 1. Concentración en Corea del Norte:

- La mayoría de las ubicaciones bombardeadas pertenecen a **Corea del Norte**, incluyendo ciudades principales como **Pyongyang**, **Hamhung-si**, **Nampo**, y regiones industriales o logísticas como **Songnim-si** y **Kaesong**.
- **Patrón**:
  - Corea del Norte fue el principal objetivo debido a su papel en el conflicto y la necesidad de debilitar sus capacidades militares e industriales.

8.1.2 2. Bombardeos en Áreas Estratégicas:

- **Ciudades Industriales y Portuarias**:
  - Ciudades como **Nampo** y **Songnim-si** son puertos y centros industriales clave.
- **Infraestructura Logística**:
  - Áreas como **Pyongsong-si** o **Kaesong**, ubicaciones cercanas a rutas de transporte importantes, reflejan un intento de interrumpir la logística enemiga.
- **Zonas Fronterizas**:
  - Áreas como **Kaesong** (cerca de la DMZ) sugieren operaciones para controlar territorios cercanos a Corea del Sur.

8.1.3 3. Bombardeos en el Agua:

- **Cantidad de Bombardeos en el Agua**: 14 bombardeos etiquetados como `"Water"`.
- **Posibles Razones**:
  - Ataques contra objetivos navales o costas.
  - Bloqueo de suministros marítimos o ataque a bases navales.

8.1.4 4. Áreas Repetidamente Bombardeadas:

- **Yangdok County** (706 bombardeos) y **Hamhung-si** (657 bombardeos):

- **Yangdok County**: Importante por rutas logísticas.
- **Hamhung-si**: Centro industrial clave.

8.2 Estrategias y Objetivos Militares

8.2.1 1. Debilitar Capacidades Industriales y Logísticas:

- Las ciudades más bombardeadas tenían importancia militar e industrial, reflejando un esfuerzo por destruir infraestructura esencial.

8.2.2 2. Aislamiento Estratégico:

- Muchas áreas bombardeadas eran nodos logísticos clave, sugiriendo un esfuerzo por cortar el suministro de recursos y movimientos de tropas.

8.2.3 3. Control de Zonas Urbanas y Portuarias:

- Bombardeos concentrados en ciudades portuarias como **Nampo** y **Songnim-si** indican intentos de controlar puntos de acceso marítimos.

8.3 Impacto en Zonas Rurales

8.3.1 1. Bombardeos en Áreas de Baja Densidad:

- Regiones rurales como **Yangdok County**, **Pyongsan County**, o **Sukchon County** también sufrieron ataques, probablemente para interrumpir el suministro agrícola o logístico.

8.3.2 2. Distribución Ampliada:

- Las ubicaciones menos bombardeadas (1-10 bombardeos) muestran la amplitud de las operaciones, cubriendo tanto objetivos estratégicos como tácticos.

8.4 Conclusiones

8.4.1 1. Prioridades del Conflicto:

- La infraestructura militar, industrial y logística de Corea del Norte fue el principal objetivo.

8.4.2 2. Estrategia Naval:

- Los ataques en el agua sugieren un intento por controlar rutas marítimas y bloquear suministros.

8.4.3 3. Amplitud del Conflicto:

- Aunque los bombardeos se concentraron en áreas estratégicas, también alcanzaron muchas ubicaciones pequeñas, reflejando la magnitud del conflicto.

8.5 Siguientes Pasos

1. **Análisis Geoespacial**:
   - Representar las ubicaciones en un mapa para observar patrones geográficos más claramente.
2. **Estudio Cronológico**:
   - Analizar los bombardeos por fechas para identificar cambios en las estrategias a lo largo del tiempo.
3. **Relación con Resultados Militares**:
   - Vincular los bombardeos con los avances o retrocesos militares para entender su efectividad.
