FreedomWings Presents:

UFO Sightings Exploratory Data Analysis

Project Objective:

* The following outlines the aim of conducting an analysis on this dataset.

UFO Reports

[i] This dataset comprises eyewitness UFO reports collected by the National UFO Research Center (NUFORC). It includes various attributes associated with the reports, making it an intriguing subject for data analytics exploration. I have selected this dataset for three reasons:

- Data Cleaning techniques:

  > This serves as an illustrative demonstration of effective techniques for tidying up a large dataset, encompassing over 141,000 lines. It also provides insights into troubleshooting issues related to missing data, data challenges, and more

- Data Analytics:

  > I will demonstrate how to extract valuable insights from this dataset and analyze the data to make sense of it, focusing on identifying key areas where UFOs have been spotted.

> The objective is to discern patterns or insights from the gathered information.
> Visualizations within this project will play a crucial role in facilitating this analysis

UFO's & UAP's

> If you have a interest in this subject, check this out

Dataset Overview

* Below is an overview of the dataset pertinent to this project.

1) Dataset:

> Name: nuforc_reports.csv
> Owner: Timothy Renner
> Shared: With Everyone, Public Domain
> Size (csv): 412.79mb
> Nr Columns: 13
> Nr rows: 141359
> Original Dataset can be found HERE

2) About:

This dataset contains eye witness ufo reports that was collected by The National UFO Research Center.

3) Data Features:

Describing the columns:

`summary:` This column is a summary of the column 'text', describing the details of the ufo encounter.
`city:` The city in which the encounter occured.
`state:` The US State in which the encounter occured.
`date_time:` The date and time the encounter occured.
`shape:` The UFO shape.
`duration:` The time duration in which the encounter occured.
`stats:` The summary of the encounter: date, time, shape, location ect.
`report_link:` The hyperlink to the original data captured (National UFO Reporting Center Sighting Report).
`text:` The eye witness details describing the encounter.
`posted:` When the encounter was posted/reported.
`city_latitude & city_longitude:` Co-ordinates of the encounter.

Step 1: Importing Modules:

In [844…

```
#====== Importing Modules needed for Notebook:
#==== Operations:
import numpy as np        # Numerical operations library
import pandas as pd       # Data manipulation library

import warnings
warnings.filterwarnings("ignore")   # Suppressing warnings for cleaner output
from datetime import datetime   # Library for working with date and time
import re   # Regular expression library for text manipulation

#==== Visualisations:
import PIL.Image   # Python Imaging Library for working with images
import seaborn as sns     # Statistical data visualization library
import plotly.express as px   # Interactive plotting library
import plotly.graph_objects as go

import matplotlib.pyplot as plt   # Plotting library
%matplotlib inline
from matplotlib import font_manager as fm


#====== Geospatial:
import geopandas as gpd      # Geospatial data manipulation library
import fiona
fiona.drvsupport.supported_drivers['ESRI Shapefile'] = 'rw'  # Configuring support for E

#====== NLP:
import nltk   # Natural Language Toolkit library for text processing
from nltk.tokenize import sent_tokenize   # Tokenizing sentences

from collections import Counter   # Collection data type for counting occurrences

from nltk.corpus import stopwords   # Stopwords for text processing
```

Step 2: Creating a Dataframe:

ⁱ I will create a dataframe using the CSV file "nuforc_reports.csv". The main goal of this section is to read the raw data and understand its features: what the data reveals, whether feature engineering is needed, if there are any missing values that require attention, and any other aspects that need consideration. It's crucial in any Exploratory Data Analysis (EDA) to first and foremost understand the nature of the data and make initial observations before delving into further analysis. My methodology for this is demonstrated within this notebook through the systematic approach I've adopted, along with detailed notes and explanations.

Encoding:

In [845…

```
#====== List of encodings to try when reading the CSV file:
try_encodings = ['utf-8', 'iso-8859-1', 'utf-16']
```

```
#====== Iterating through the encodings to find out encoding format for csv file:
for encoding in try_encodings:
    try:
        ufo_df = pd.read_csv("ufo_sightings.csv", encoding = encoding)
        encoding_type = encoding
        print(f'''============ Feedback: ============
        \rSuccessfully read using {encoding} encoding.''')

        break
    except UnicodeDecodeError:
        print(f"Failed to read using {encoding} encoding.")
```

```
============ Feedback: ============
Successfully read using utf-8 encoding.
```

Dataframe:

In [846…
```
#====== Generating a dataframe:
ufo_df = pd.read_csv("ufo_sightings.csv", encoding = encoding_type)
```

In [847…
```
#====== Setting a styled format to read the dataframe:
def style_dataframe(df):

    styled_df = df.style.set_properties(**{ # Parameters listed below:
        'border': '2px solid green',
        'color': 'white',
        'background-color': '#0B164B',
        'font-weight': 'bold',
        'font-size': '14px',
        'text-align': 'left',
        'font-family': 'Arial',
        'padding': '10px',
        'margin': '5px',
        'white-space': 'nowrap',
        'text-decoration': 'underline'
    })

    return styled_df
```

In [848…
```
#====== Reading dataframe with styled format:
style_dataframe(ufo_df.head(10))
```

Out[848]:

0	MADAR Node 100
1	Steady flashing object with three lights hovered in sky
2	Group of several orange lights, seemingly circular. Lights did not blink. ((anonymous report))
3	Dropped in flashed a few times and shot off 5 or 6 balls of light then shot back up extermly fast. ((a
4	Location: While traveling in a TGV, from Lille to Charles du Gaule airport - Paris, after the outskirts
5	Llike a star at first glance, got brighter and bigger, then dimmer and small again. Stationary the wh
6	Light in the sky moving from south to north with weird up and down movements. Watch it for abou
7	Glowing circle moving through the sky. Canton, CT.
8	The crew of an airliner at 34,000' witnesses a metallic sphere streak by their aircraft; reported to FA

Describing:

In [849…
```
#===== Checking column names and data types:
ufo_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 141261 entries, 0 to 141260
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   summary         141189 non-null  object 
 1   country         140944 non-null  object 
 2   city            140779 non-null  object 
 3   state           131681 non-null  object 
 4   date_time       138601 non-null  object 
 5   shape           134962 non-null  object 
 6   duration        133642 non-null  object 
 7   stats           141261 non-null  object 
 8   report_link     141261 non-null  object 
 9   text            141227 non-null  object 
 10  posted          138601 non-null  object 
 11  city_latitude   115440 non-null  float64
 12  city_longitude  115440 non-null  float64
dtypes: float64(2), object(11)
memory usage: 14.0+ MB
```

Step 3: Checking Missing Values::

Nan Values:

In [850…
```
#====== Checking sum of missing values per column:
print("====== \u274f Missing Values: =======")

ufo_df.isna().sum()
```

Out[850]:
```
====== ❏ Missing Values: =======
summary             72
country            317
city               482
state             9580
date_time         2660
shape             6299
duration          7619
stats                0
report_link          0
text                34
posted            2660
city_latitude    25821
city_longitude   25821
dtype: int64
```

Missing ratio:

In [851…
```
#====== Check total number of rows within dataset:
num_rows = len(ufo_df)

#====== Checking total number of missing values:
total_NaN = sum(ufo_df.isna().sum())
```

```python
#====== Check how many rows have missing values:
total_missing_rows = ufo_df.isna().any(axis=1).sum()

#====== Calculate the missing value ratio percentage:
missing_ratio = (total_missing_rows / num_rows) * 100

#====== Printing out Findings:
print(f'''====================== Findings: ======================
Total number of rows:                     {num_rows}
Total number of missing values:           {total_NaN}
Total number of rows with missing values: {total_missing_rows}
Ratio percentage:                         {round(missing_ratio)}%
=========================================================
''')
```

```
====================== Findings: ======================
Total number of rows:                     141261
Total number of missing values:           81365
Total number of rows with missing values: 36098
Ratio percentage:                         26%
=========================================================
```

Data Cleaning Overview

Missing Values Observations:

1. There is a notable prevalence of missing values within the columns, and the corresponding missing ratio stands at a significant 26%. Consequently, I am inclined not to remove these missing values, as doing so could potentially undermine the accuracy of the analysis I intend to conduct.

1. Among the columns, those pertaining to city, state, and coordinates exhibit the most substantial occurrences of missing values. To address this issue without eliminating any missing data, I will need to employ imputation techniques.

====== □ Missing Values: =======
summary 72
country 317
city 478
state 9580
date_time 2660
shape 6299
duration 7616
stats 0
report_link 0
text 34
posted 2660
city_latitude 25821
city_longitude 25821

Describing the columns:

- We will analyze the details within each column and explore potential approaches to address the presence of missing values.

> `Summary`
> Initially, the 'summary' column exhibits fewer missing values compared to the 'text' column.
> Therefore, I intend to drop the 'text' column.
> Given that the summarization format within the 'summary' column is inadequate, I plan to eliminate this column, create a new one, and employ sentiment analysis instead-extracting the data from the 'summary' column before deleting.
>
> `city`
> *The city in which encounter occured.*
> The general format of this column is as example: Caloocan City (Philippines). It has the city and the country name in brackets.
> I want the city and country to be seperate, so I will therfore extract the country from this column and impute it into the country column.
>
> `state`
> *The US State in which encounter occured.*
> This is expected to be a large amount of missing values since not all countries have states, ie- unique only to US.
> I will therefore perform imputation and label the missing values as 'unknown' and use the state codes to fill data in country column for USA.
>
> `date_time`
> *The date and time encounter occured.*
> The general format of this column is as example: (YYYY-MM-DDTHH:mm:ss)1988-08-10T15:00:00.
> I will therfore change the date format to exclude the time.
>
> `shape`
> *The UFO shape.*
> I will identify the distinct/unique shapes mentioned, and I have the option to either use the average described shape for filling in the gaps or label the absent values as 'other'.
>
> `duration`
> *The time duration in which encounter occured.*
> The format of this data is too mixed making it difficut to convert to a specific duration format.
> I will therfore drop this column aswell
>
> `stats`
> *The summary of event: date,time,shape,location ect.*
> I will drop this column as I do not need the summary as I will be performing my own EDA on this dataset.
>
> `report_link`
> *The hyperlink to the original data captured (National UFO Reporting Center Sighting Report).*
> I will drop this column as wont be needed for data analysis

`text`

The details describing the encounter

As mentioned above will also drop this column

`posted`

When the event was posted/reported.

I will also drop this column as I dont need to know when the report was posted, its irrelevant.

`city_latitude`

The latitude location

I will refine this column with merging with another dataset

`city_longitude`

The longitude location

I will refine this column with merging with another dataset

Step 4: Data Clean and Preparing:

Checking Unique columns:

In [852…

```
#====== Checking columns unique values and frequency/counts:
for col in ufo_df.iloc[:,1:].select_dtypes(include = 'object').columns:
    print(col)

    print("-------------------------------------")
    print(ufo_df[col].value_counts(normalize = True))
    print("-------------------------------------")
```

```
country
-------------------------------------
country
USA                 0.893007
Canada              0.040449
United Kingdom      0.022193
Australia           0.006286
India               0.003143
                      ...
Republic of Korea   0.000007
Ochorios            0.000007
Tyrrhenian Sea      0.000007
none                0.000007
USVI                0.000007
Name: proportion, Length: 439, dtype: float64
-------------------------------------
city
-------------------------------------
city
New York                          0.006187
Phoenix                           0.005619
Seattle                           0.005555
Las Vegas                         0.004781
Portland                          0.004660
                                    ...
Ein Geidi Beach (Israel)          0.000007
Chatham Maritime (UK/England)     0.000007
Coastline                         0.000007
Szekesfehervar (Hungary)          0.000007
```

```
Cappagh                          0.000007
Name: proportion, Length: 23863, dtype: float64
-----------------------------------
state
-----------------------------------
state
CA                           0.117359
FL                           0.059409
WA                           0.052369
TX                           0.044167
NY                           0.042709
                               ...
SOUTH GYEONGSANG PROVINCE    0.000008
BURGAS PROVINCE              0.000008
TA                           0.000008
BUCHAREST                    0.000008
COUNTY WATERFORD             0.000008
Name: proportion, Length: 292, dtype: float64
-----------------------------------
date_time
-----------------------------------
date_time
2015-11-07T18:00:00    0.000750
2014-07-04T22:00:00    0.000332
2010-07-04T22:00:00    0.000267
2012-07-04T22:00:00    0.000245
1999-11-16T19:00:00    0.000202
                           ...
2006-12-22T07:11:00    0.000007
2006-12-22T04:45:00    0.000007
2006-12-21T22:00:00    0.000007
2006-12-21T20:18:00    0.000007
2019-12-24T18:30:00    0.000007
Name: proportion, Length: 118057, dtype: float64
-----------------------------------
shape
-----------------------------------
shape
light        0.211215
circle       0.111757
triangle     0.093411
other        0.071709
fireball     0.071709
unknown      0.070879
sphere       0.067982
disk         0.061780
oval         0.045316
formation    0.034617
changing     0.027030
cigar        0.026185
rectangle    0.018161
flash        0.017598
cylinder     0.017012
diamond      0.015063
chevron      0.011981
egg          0.009010
teardrop     0.008906
cone         0.004283
cross        0.003468
star         0.000674
delta        0.000252
Name: proportion, dtype: float64
-----------------------------------
duration
-----------------------------------
duration
```

```
5 minutes                0.064044
2 minutes                0.046632
10 minutes               0.045136
1 minute                 0.040122
3 minutes                0.034375
                           ...
30 minutes on and off    0.000007
Just over 7 minutes      0.000007
1second-12 min           0.000007
2 min. (est.)            0.000007
1/12                     0.000007
Name: proportion, Length: 14008, dtype: float64
-----------------------------------
stats
-----------------------------------
stats
Occurred : 3/29/2020 21:50  (Entered as : 3/29/20 21:50) Reported: 4/1/2020 2:26:47 AM 0
2:26 Posted: 6/25/2020 Location: Smilax, KY Shape: Sphere Duration: 1 minute

                                                                                0.
000014
Occurred : 7/15/1995 15:20 Reported: 7/15/1995 15:29 Posted: 11/2/1999 Location: Seattl
e, WA Shape:  Duration: 2 min.


0.000014
Occurred :   Reported: 6/19/2021 12:09:37 PM 12:09 Posted: 8/16/2021 Location: Corinth,
NY Shape: Circle Duration: 30 seconds Characteristics: There was an aura or haze around
the object
                                                                                0.
000014
Occurred : 6/19/2014 23:00  (Entered as : 06/19/14 23:00) Reported: 6/25/2014 6:29:57 PM
18:29 Posted: 6/27/2014 Location: Austin, TX Shape: Cylinder Duration: 10 minutes Charac
teristics: There were lights on the object, There was an aura or haze around the object,
The object emitted beams, The object changed color, Animals reacted to the event    0.00
0014
Occurred : 4/1/2019 20:00  (Entered as : 4/2019 20:00) Reported: 12/29/2019 9:27:11 AM 0
9:27 Posted: 2/7/2020 Location: Manitowoc, WI Shape: Unknown Duration: 20 minutes

                                                                                0.
000014




...
Occurred : 6/20/1985 14:00  (Entered as : 06/20/85 14:00) Reported: 6/25/2006 9:14:09 AM
09:14 Posted: 7/16/2006 Location: Salem, NH Shape: Sphere Duration: seconds Characterist
ics: There was an aura or haze around the object, The object emitted beams, The object c
hanged color                                                                    0.0
00007
Occurred : 2/15/1985 00:00  (Entered as : 021585 0:00) Reported: 6/30/2006 2:58:37 PM 1
4:58 Posted: 7/16/2006 Location: Hinesville, GA Shape: Sphere Duration: 30minutes


0.000007
Occurred : 6/20/1983 13:00  (Entered as : 06/20/1983 13:00) Reported: 6/13/2006 2:03:21
PM 14:03 Posted: 7/16/2006 Location: Balderton Newark on Trent, Nottinghamshire (UK/Engl
and),  Shape: Other
                                                                                0.
000007
Occurred : 5/17/1983 05:45  (Entered as : 05/17/83 5:45) Reported: 7/3/2006 10:17:50 PM
22:17 Posted: 7/16/2006 Location: Tallahassee, FL Shape: Unknown Duration: 10 minutes Ch
aracteristics: There were aircraft in the vicinity or aircraft chasing the object
                                                                                0.
000007
```

Occurred : 12/24/2019 18:30 (Entered as : 012419 18:30) Reported: 5/13/2019 6:25:41 PM 18:25 Posted: 5/14/2019 Location: Fredericksburg, VA Shape: Sphere Duration: 4 seconds Characteristics: There were lights on the object, There were aircraft in the vicinity or aircraft chasing the object 0.000007
Name: proportion, Length: 141247, dtype: float64
---
report_link
---
report_link
http://www.nuforc.org/webreports/reports/147/S147087.html 0.000007
http://www.nuforc.org/webreports/reports/107/S107748.html 0.000007
http://www.nuforc.org/webreports/reports/107/S107751.html 0.000007
http://www.nuforc.org/webreports/reports/107/S107676.html 0.000007
http://www.nuforc.org/webreports/reports/107/S107679.html 0.000007
... ...
http://www.nuforc.org/webreports/reports/051/S51075.html 0.000007
http://www.nuforc.org/webreports/reports/051/S51177.html 0.000007
http://www.nuforc.org/webreports/reports/050/S50935.html 0.000007
http://www.nuforc.org/webreports/reports/051/S51205.html 0.000007
http://www.nuforc.org/webreports/reports/146/S146103.html 0.000007
Name: proportion, Length: 141261, dtype: float64
---
text
---
text
MADAR Node 119

0.000404

MADAR Node 142

0.000375

MADAR Node 143

0.000347

MADAR Node 100

0.000333

MADAR Node 106

0.000304

...

Two objects were observed moving eratically in the early morning sky, each multi-lighted. \n \nI got up in the middle of the night to use the bathroom and a bright light was shining from the sky into my eighth floor apartment. The light was so different from anything I have ever seen that I went onto our balcony and using binoculars observed it. I then woke up my husband and each of us noticed that there were two strange lights in the sky, one very big and a smaller one to the right. My husband called the police and the officer viewed the lights from our balcony. He told me to write to you. All of us thought it was very unusual. \n \n The officer left and I continued watching with wonder. The next evening, I got up around the same time and saw the lights again. This time I took photos. \n \n \n((NUFORC Note: Witness elects to remain totally anonymous; provides no contact information. PD))

0.000007

Bright starlike light seen hovering then speeding out of sight. \n \nAt approximately 2:25 A.M. on the morning of 30 Nov. 2007, I saw in the northern sky a bright whiteish/orange light that looked like a bright star that was stationary in the sky. I was traveling in my car with my wife and daughter. The intensity of the light was that of the brightest star you've ever seen. Probably 2 or 3 times that of Jupiter. As we continued traveling eastward the light remained the same for approximately 30 seconds. We then turned and traveled northward for approximately .25 mile. I then stopped the car and rolled the window down to more closely observe. It was then the object started traveling westward. The intensity of the light diminished to a faint light. It picked up speed as it traveled and went out of sight on the horizon in approximately 3 to 4 seconds. Things to note: There was no sound heard, there were no flashing lights like that of a normal flying aircraft, there was cloud cover with a ceiling of around 10,000 feet. No precipitation. \n \n \n((NUFORC Note: Witness elects to remain totally anonymous. PD))

0.000007

A cluster of lights interchanging and changing color. \n \nThe other night, I was drivin g home in the farm lands. I was taking a shortcut home. As I was driving, I believe I sa w something. This something was hovering, about 20-30 feet above ground. It was just off the road maybe only 50 ft or so. I didn’t know what it was, due to the fact, I have neve r seen anything like this in my life. I didn’t know what to do, whether I should stop wh at I was doing and try to get a better look at this object, or if I should’ve fled away being afraid of what could possibly happen. I was somewhat scared, because it was abnorm al and it was so close to me. This object was so close to me, that I could have thrown a rock at it and I would’ve definitely hit it. Being so afraid and uncertain of what I was seeing, I just kept driving, but only faster. I didn’t tell anybody because I didn’t wan t people to think I was hallucinating or something. \n \nNow to describe this object. I saw a cluster of lights. The lights were not in a row or any formation. They were not in a sequence or in any pattern to form a shape of the object. They were spread out. They c hanged color. All of the lights changed color at the same time. At first the lights were a whitish color. Then they changed to a blue-grean or teal color. Next they appeared in a reddish or red-orange color. They flashed or should I say switched position. One light would go out and another one would light up. They switched from light to light extremely fast and again, not in any pattern or sequence. This flashing of lights made it hard to identify its shape. I could not make out the body of the object because of the scramblin g lights. When the lights changed from light to light, it was like there was a streak of light following behind it, kind of like a tracer. This object didn’t appear to be on th e ground, but very close to it. And it wasn’t moving, and if it was it was cruising very slowly. \n \nAfter seeing this, I was not sure of what to do. I was so astonished, I cou ldn’t believe what I had just seen. I thought there was some kind of mistake or I was ju st seeing things. I never really put too much thought into this sighting. I never really believed in these sightings that others reported. I was just online at MSN.COM and I saw an article about a recent sighting in Texas. There was a video of the witness talking to the news crew. He was describing what he had seen. The way he described the object was a very accurate description of what I saw. I should have made this report a lot sooner. Bu t me seeing this video of the witness and him describing the sighting, pushed me to make a report. Maybe my info could help you all out in some way. \n \n \n((NUFORC Note: Witn ess indicates that the date of the event is approximate. PD)) 0.000007

Glowing V moves across sky \n \nI walked outside to get something from my car and looked up. The night was clear and I started to admire the stars when I saw right in the middl e of my field of vision a v shaped object moving steadily across the sky. I thought to video it with my cell phone, but the glow was so dim I knew it wouldn't come through so I just stared at it as it moved out of my field of vision. There were lights from the c ity (I'm close to Baltimore) so as it got close to the horizon the lights overpowered th e dim glow. \n \n

0.000007

While driving at night, I watched two blue-green light-emitting spheres flying fast \n \nWhile driving west at night, on rt. 3, about five miles east of Fredericksburg, Virginia, I watched two blue-green light-emitting spheres, flying fast, in tight formation, at about 600 feet above the ground, from the southwest toward the northeast. They were followed immediately by a pursuing aircraft with familiar, man-made looking navigation lights. This all took about four seconds. I never heard a sound from any of the lights' sources.

0.000007

```
Name: proportion, Length: 139328, dtype: float64
-----------------------------------
posted
-----------------------------------
posted
2020-06-25T00:00:00    0.013153
2009-12-12T00:00:00    0.011580
2006-10-30T00:00:00    0.011075
2019-12-01T00:00:00    0.010671
2010-11-21T00:00:00    0.009675
                         ...
2003-01-20T00:00:00    0.000014
1999-05-23T00:00:00    0.000014
2019-12-06T00:00:00    0.000014
2002-03-25T00:00:00    0.000007
2003-01-10T00:00:00    0.000007
Name: proportion, Length: 617, dtype: float64
-----------------------------------
```

Duplicates:

In [853…
```
#====== Checking duplicates:
ufo_df.duplicated().sum()
```

Out[853]:
```
0
```

Columns drop:

In [854…
```
#====== Drop the columns not needed:
columns_to_drop = ["duration","report_link","text","posted"]
ufo_df.drop(columns_to_drop, axis = 1, inplace = True)

#====== Displaying updated columns:
ufo_df.columns.to_list()
```

Out[854]:
```
['summary',
 'country',
 'city',
 'state',
 'date_time',
 'shape',
 'stats',
 'city_latitude',
 'city_longitude']
```

Column: summary

`Summary clean:` * Replacing all character strings to empty spaces in order to read text more efficiently. -Checking the total number of hoaxes picked up by the National UFO Reporting Center Sighting Report. -Checking the word hoax by setting the conditions to convert all text to lowercase. -Setting the expressions r'\bhoax\b' in order to search of instances of the complete word, ignoring cases where it might be part of a larger word. -Remove or filter out rows from the DataFrame where the 'summary' column contains the word 'Hoax' (case-insensitive) as a whole word.It then updates the df to contain only the rows where this condition is not met.

In [855…
```
#====== Checking missing values in 'summary':
ufo_df['summary'].isna().sum()
```

Out[855]:
```
72
```

In [856…
```
#====== Filling missing values with 'unknown':
ufo_df['summary'].fillna('unknown', inplace = True)
```

In [857…
```
#====== Defining a function to replace characters in strings:
def clean_summary(summary):

    cleaned_summary = summary.replace('(', '').replace(')', '').replace('/', ' ').replac
    return cleaned_summary

#====== Apply the function to the summary column:
ufo_df['summary'] = ufo_df['summary'].apply(clean_summary)
```

In [858…
```
#====== Checking total values of hoaxes declared:
hoax_count = ufo_df['summary'].str.lower().str.count(r'\bhoax\b').sum() # Counting nr of
print("Total occurrences of 'HOAX' in the summary column:\n",hoax_count)
```

```
Total occurrences of 'HOAX' in the summary column:
 1446
```

In [859…
```
#====== Removing rows containing 'hoax':
ufo_df = ufo_df[~ufo_df['summary'].str.contains(r'\bHoax\b', case = False)]

#====== Confirming that hoaxes have been removed:
hoax_count = ufo_df['summary'].str.lower().str.count(r'\bhoax\b').sum()
print("Total occurrences of 'HOAX' in the summary column:\n",hoax_count)
```

```
Total occurrences of 'HOAX' in the summary column:
 0
```

Column: state

`Clean State:` * Fill in missing values with 'unspecified'.

In [860…
```
#====== Checking missing values again:
ufo_df['state'].isna().sum()
```

Out[860]:
```
9432
```

In [861…
```
#====== Filling in missing values:
ufo_df['state'].fillna('unspecified', inplace = True)
```

Column: Country

`Clean Country:` >Creating a new DataFrame by opening the file named 'worldcities_refined.' > * Dropping all missing values in that new dataframe > * Extracting values from the 'city' and 'country' columns, pairing them using the zip function, and storing the pairs in a dictionary. This will enable mapping cities to their respective countries using the data in this new DataFrame. The mapping will be applied to the existing ufo_df DataFrame, replacing all 'unspecified' values. * The remaining 'unspecified' values that I was not able to map, will be changed to the average country

In [862…
```
#====== Checking missing values again:
ufo_df['country'].isna().sum()
```

Out[862]:
```
297
```

In [863…
```
#====== Filling missing values to 'unspecified':
ufo_df['country'].fillna('unspecified', inplace = True)

'''
Saving total nr of 'unspecified' countries to a variable:
I am saving the total to check later for before and after mapping
'''
before_mapping = ufo_df['country'].value_counts().get('unspecified', 0)

#====== Confirming if all missing values have been replaced:
ufo_df['country'].isna().sum()
```

Out[863]:
```
0
```

1) New Dataset (country mapping):

Utilizing a global city dataset, which serves as a valuable resource for various mapping techniques. This dataset comprises approximately 43,000 distinct cities and towns spanning every country worldwide, complete with latitude and longitude coordinates. The dataset's most recent update was completed on March 31, 2023. I have `refined the original csv dataset by deleting the columns and creating 2 columns showing city and country respectably.`

> Name: worldcities.csv
> Owner: Juanma Hernández
> Shared: With Everyone, Public Domain
> Size (csv):2mb
> Original Dataset can be found HERE

In [864…
```
#====== Creating new Data frame:
city_country_df = pd.read_csv('worldcities_refined.csv')

#====== Checking missing values:
city_country_df.isna().sum()
```

Out[864]:
```
city       316
country      0
dtype: int64
```

In [865…
```
#====== Dropping all missing values:
city_country_df.dropna(inplace = True)
```

In [866…
```
#====== Creating a dictionary mapping cities to their respective countries:
city_to_country_mapping = dict(zip(city_country_df['city'], city_country_df['country']))
```

In [867…
```
#======Identifying all rows with 'unspecified' in the 'country' column:
unspecified_rows = ufo_df[ufo_df['country'] == 'unspecified']

#====== Iterating through rows with 'unspecified' in the 'country' column and updating b
for index, row in unspecified_rows.iterrows():
    city = row['city']   # Extracting the city value from the current row
    if city in city_to_country_mapping:   # Checking if city is present in the mapping d
        # Updating the 'country' column with the mapped country for the current city
        ufo_df.at[index, 'country'] = city_to_country_mapping[city]

#====== Saving results to variables:
after_mapping = ufo_df['country'].value_counts().get('unspecified', 0)

#====== Printing Results:
print(f'''
Country column values BEFORE mapping:{before_mapping}
Country column AFTER before mapping:{after_mapping}
Total nr of unspecified countries replaced: {before_mapping - after_mapping}
''')
```

```
Country column values BEFORE mapping:297
Country column AFTER before mapping:223
Total nr of unspecified countries replaced: 74
```

In [868…
```
'''
Although we've handled 74 unspecified values, there are still 223 remaining to address:
'''

#====== Calculating the average country value:
average_country = ufo_df['country'].mode()[0] #returns a Series object(first value)

#====== Replacing 'unspecified' with the average country value:
```

```
ufo_df['country'].replace('unspecified', average_country, inplace=True)

#====== Confirm that all unspecified countries have been replaced:
final_mapping = ufo_df['country'].value_counts().get('unspecified', 0)
print(final_mapping)
```

```
0
```

Column: city

`Clean City:` * Fill in missing values with 'unspecified'. - Setting a function to replace the string characters with empty spaces in order to read more effectively. - Applying function to the city column

In [869…
```
#====== Checking missing values:
ufo_df['city'].isna().sum()
```

Out[869]:
```
449
```

In [870…
```
#====== Filling missing values:
ufo_df['city'].fillna('unspecified', inplace=True)
```

In [871…
```
#====== Defining a function to replace characters in strings:
def clean_city(city):

    cleaned_city = city.replace('(', '').replace(')', '').replace('/', ' ').replace(',',
    return cleaned_city

#====== Applyimg the above function to the 'city' column:
ufo_df['city'] = ufo_df['city'].apply(clean_city)
```

In [872…
```
#====== Viewing total nr of occurences of unique cities:
ufo_df['city'].value_counts()
```

Out[872]:
```
city
New York                  854
Phoenix                   785
Seattle                   779
Las Vegas                 668
Portland                  649
                         ...
Pleasant Lake               1
Bucerias Nayarit Mexico     1
Tarrant City                1
Mantova Italy               1
Cappagh                     1
Name: count, Length: 23649, dtype: int64
```

Column: date_time

`Clean Date:` * Checking Missing values - Converting the date_time column to datetime objects. - Formatting 'date_time' to include only the date, excluding time entries. - Parsing the 'date_time' values into a new column 'date_only'. > To address missing values, I plan to impute them by extracting data from the 'stats' column and using it to replace the missing values. Since the 'stats' column may contain dates, some rows might lack dates in the 'date_time' column but have them in the 'stats' column. I intend to match and impute the missing values accordingly. - Defining a function to search for datetime values in the 'stats' column using the re.search function from the re module for regex pattern matching. If a match is found, it will replace the missing value; otherwise, it will return pd.NaT (used in pandas to

represent missing or undefined datetime values). - Executing this function within a temporary new column 'date_extracted', then using it to replace the values in the original 'date_only' column. - After executing this code, I will drop the remaining missing values.

In [873…
```
#====== Checking missing values:
ufo_df['date_time'].isnull().sum()
```

Out[873]:
```
2622
```

In [874…
```
#====== Converting to date time format:
ufo_df['date_time'] = pd.to_datetime(ufo_df['date_time'])

#====== Extracting date only and parsing to a new column:
ufo_df['date_only'] = ufo_df['date_time'].dt.date

#====== Dropping the original 'date_time' column:
ufo_df.drop(columns = ['date_time'], inplace = True)
```

In [875…
```
#====== Defining a function to extract the date from the 'stats' column:
def extract_date(stats):
    match = re.search(r'(\d{1,2}/\d{1,2}/\d{4})', stats)
    if match:
        return match.group(0)
    else:
        return pd.NaT

#====== Applying the function to extract the date and create a new 'date_extracted' colu
ufo_df['date_extracted'] = ufo_df['stats'].apply(extract_date)

#====== Convert the 'date_extracted' column to datetime:
ufo_df['date_extracted'] = pd.to_datetime(ufo_df['date_extracted'], errors = 'coerce')

#====== Replace missing values in 'date_only' with the extracted dates:
ufo_df['date_only'].fillna(ufo_df['date_extracted'], inplace = True)

#====== Drop the temporary 'date_extracted' column
ufo_df.drop(columns = ['date_extracted'], inplace=True)
```

In [876…
```
#====== Checking new missing values total:
ufo_df['date_only'].isna().sum()
```

Out[876]:
```
5
```

In [877…
```
#====== Dropping final rows with missing dates:
ufo_df.dropna(subset = ['date_only'], inplace = True)

#====== Checking new missing values total:
ufo_df['date_only'].isna().sum()
```

Out[877]:
```
0
```

Column: shape

`Clean Shape:`

- Checking for missing and unique values.
- Performing imputation for missing values:

- Defining a function to group unique shape values and categorize them to reduce uniqueness and simplify the data.
- Creating a variable named 'ufo_shapes' to store the resulting list of unique shape values.
- Executing code to search for matching keywords within the 'ufo_shapes' list in the 'summary' column. When a match is found, it replaces the 'unspecified' values with the corresponding keyword from the 'ufo_shapes' list. The function uses a lambda row function to apply the specified condition to each row in the DataFrame. The line next((shape for shape in ufo_shapes if shape in row['summary'].lower()), row['shape']) is a generator expression that iterates over the 'ufo_shapes' list and checks if any of the shapes are present in the lowercase version of the 'summary' column (row['summary'].lower()). The next function returns the first matching shape or row['shape'] if no match is found. Calculating the most common shape per city by grouping the data by the 'city' column and categorizing by 'shape'.
- Merging the grouped 'avg_shape_per_city' data back into the 'ufo_df' DataFrame.
- Applying a function to replace the 'unspecified' values with the most common shape for each city.

In [878…
```
#====== Checking missing values:
ufo_df['shape'].isnull().sum()
```

Out[878]:
```
6218
```

In [879…
```
#====== Filling missing values:
ufo_df['shape'].fillna('unspecified', inplace = True)
```

In [880…
```
#====== Viewing Unique values for shape:
ufo_df['shape'].value_counts()
```

Out[880]:
```
shape
light          28341
circle         14932
triangle       12463
fireball        9592
other           9553
unknown         9439
sphere          9109
disk            8200
unspecified     6218
oval            6056
formation       4641
changing        3599
cigar           3502
rectangle       2417
flash           2347
cylinder        2275
diamond         2004
chevron         1604
egg             1189
teardrop        1180
cone             570
cross            461
star              91
```

```
delta             34
Name: count, dtype: int64
```

In [881…
```
#====== Defining fuction that replaces values:
def replace_column_value(df, column_name, old_value, new_value):
    df[column_name] = df[column_name].replace(old_value, new_value)

#====== Executing function to categorise the ufo shapes:
replace_column_value(ufo_df, 'shape', 'egg', 'oval')
replace_column_value(ufo_df, 'shape', 'circle', 'sphere')
replace_column_value(ufo_df, 'shape', ['flash','changing','star'], 'light')
replace_column_value(ufo_df, 'shape', ['unknown','cross','other'], 'unspecified')
replace_column_value(ufo_df, 'shape', 'cylinder', 'cigar')
replace_column_value(ufo_df, 'shape', ['delta','chevron'], 'triangle')
```

In [882…
```
#====== Viewing Unique values for shape:
ufo_df['shape'].value_counts()
```

Out[882]:
```
shape
light          34378
unspecified    25671
sphere         24041
triangle       14101
fireball        9592
disk            8200
oval            7245
cigar           5777
formation       4641
rectangle       2417
diamond         2004
teardrop        1180
cone             570
Name: count, dtype: int64
```

In [883…
```
#====== Setting variable containing unique categories:
ufo_shapes = ['light', 'sphere', 'cigar', 'unspecified', 'triangle', 'oval',
              'teardrop', 'rectangle', 'formation', 'fireball', 'disk',
              'diamond', 'cone']

#====== Applying custom function to update the shape column.
ufo_df['shape'] = ufo_df.apply(lambda row: next((shape for
                                                 shape in ufo_shapes if shape in row['s
                                                , row['shape']) if row['shape'] == 'uns
```

In [884…
```
#====== Viewing Unique values for shape:
ufo_df['shape'].value_counts()
```

Out[884]:
```
shape
light          43798
sphere         24220
unspecified    15231
triangle       14221
fireball        9722
disk            8232
oval            7303
cigar           5869
formation       4979
rectangle       2436
diamond         2033
teardrop        1187
cone             586
Name: count, dtype: int64
```

In [885…
```
#====== Calculating avg shape per city:
avg_shape_per_city = ufo_df.groupby('city')['shape'].agg(lambda x: x.mode().iloc[0]).re
```

```
avg_shape_per_city.columns = ['city', 'avg_shape']

#====== Merging the average shapes back into the original DataFrame:
ufo_df = ufo_df.merge(avg_shape_per_city, on='city', how = 'left')

#====== Replacing 'unspecified' shapes with the corresponding average shape:
ufo_df['shape'] = ufo_df.apply(lambda row: row['avg_shape'] if row['shape'] == 'unspeci

#====== Dropping the temporary 'avg_shape' column:
ufo_df = ufo_df.drop('avg_shape', axis=1)
```

In [886…

```
#====== Viewing final unique values for shape:
ufo_df['shape'].value_counts()
```

Out[886]:

```
shape
light          53045
sphere         25118
triangle       14431
fireball        9930
disk            8487
oval            7401
cigar           6037
formation       5062
unspecified     3979
rectangle       2453
diamond         2085
teardrop        1194
cone             595
Name: count, dtype: int64
```

Column: Latitude & Longitude

`Clean Co-ordinates:`

- Checking for missing values.
- Performing imputation for missing values:
- Merge ufo_df with world_cities_df to enrich the UFO data with latitude and longitude information. The merge is based on the 'city' column using a left join strategy. This ensures all rows from ufo_df are included, with matching rows from world_cities_df added based on the 'city' column.
- Update 'city_latitude' in ufo_df by filling missing values with latitude information from world_cities_df. The 'combine_first' method is used to prioritize latitude information from world_cities_df when available, while falling back to the original 'city_latitude' values from ufo_df for non-matching rows.

In [887…

```
#====== Checking missing values:
ufo_df[['city_latitude','city_longitude']].isna().sum()
```

Out[887]:

```
city_latitude     25458
city_longitude    25458
dtype: int64
```

In [888…

```
#====== Setting a new DataFrame:
world_cities_df = pd.read_csv('worldcities.csv')
```

In [889…

```
#====== Merge DataFrames based on 'city' column:
merged_df = pd.merge(ufo_df, world_cities_df[['city', 'lat', 'lng']], on = 'city', how

#====== Replace latitude and longitude columns in ufo_df with values from world_cities
ufo_df['city_latitude'] = merged_df['lat'].combine_first(ufo_df['city_latitude'])
ufo_df['city_longitude'] = merged_df['lng'].combine_first(ufo_df['city_longitude'])
```

In [890…
```
#====== Checking missing values:
ufo_df[['city_latitude','city_longitude']].isna().sum()
```

Out[890]:
```
city_latitude     4544
city_longitude    4544
dtype: int64
```

In [891…
```
#====== Drop rows with missing values in 'city_latitude' and 'city_longitude':
ufo_df.dropna(subset=['city_latitude', 'city_longitude'], inplace = True)
```

Final check on missing values:

In [892…
```
ufo_df.isna().sum()
```

Out[892]:
```
summary           0
country           0
city              0
state             0
shape             0
stats             0
city_latitude     0
city_longitude    0
date_only         0
dtype: int64
```

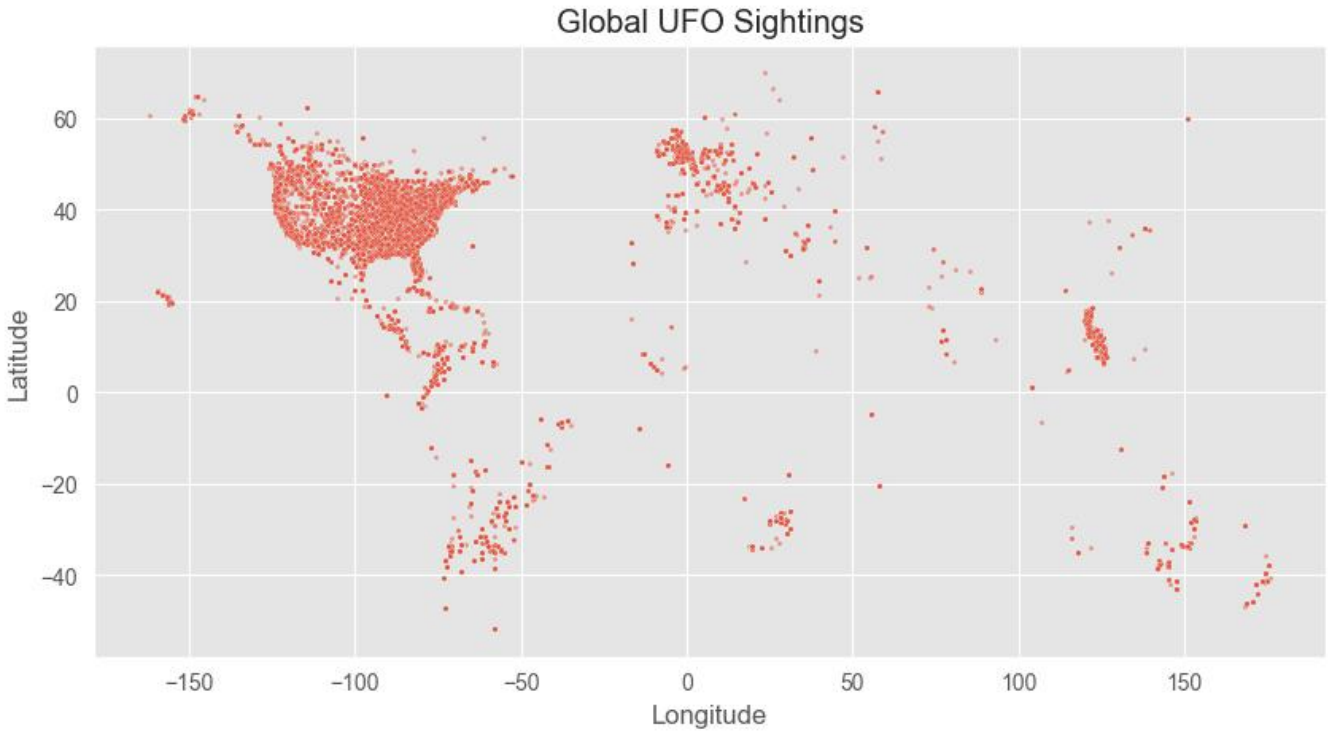
In [893…
```
#======== ML COPY:
ml_df = ufo_df.copy()
```

Data assessment post-cleaning: `With all the data now cleaned, prepared, and missing values addressed, I can proceed to data analysis and visualization.`

Data Visualisations:

In [894…
```
#================ Creating a scatter plot of coordinates with world-map overlay

#====== Extract latitude and longitude columns:
latitude = ufo_df['city_latitude']
longitude = ufo_df['city_longitude']

#====== Create a scatter plot using Seaborn:
plt.figure(figsize = (10, 5))
sns.scatterplot(x = longitude, y = latitude, s = 5, alpha = 0.5)

#====== Add labels and title:
```

```
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Global UFO Sightings')

#====== Show the plot:
plt.grid(True)
plt.show()
```



In [895…

```
#================ Creating a scatter plot of coordinates with a world-map overlay usir

#====== Plotting the graph:
fig = px.scatter(
    ufo_df,
    x = 'city_longitude',
    y = 'city_latitude' )

#====== Create a map trace with a world map:
# Original source (https://github.com/johan/world.geo.json/blob/master/UNLICENSE)
map_trace = go.Choropleth(
    z = [0],
    showscale = False,
    geojson = 'https://raw.githubusercontent.com/johan/world.geo.json/master/countries
    locations = [],
    hoverinfo = 'location+z',
)

#====== Create a figure with both the scatter plot and the map trace:
fig = go.Figure(data = [fig.data[0], map_trace])

#====== Customize the layout of the combined figure:
fig.update_layout(
    title = 'UFO Sightings with World Map Overlay',
    xaxis_title = 'Longitude',
    yaxis_title = 'Latitude',
    xaxis = dict(showgrid = True),
    yaxis = dict(showgrid = True),
    geo = dict(
        showland = True,
        landcolor = 'black',
    ),
)
```

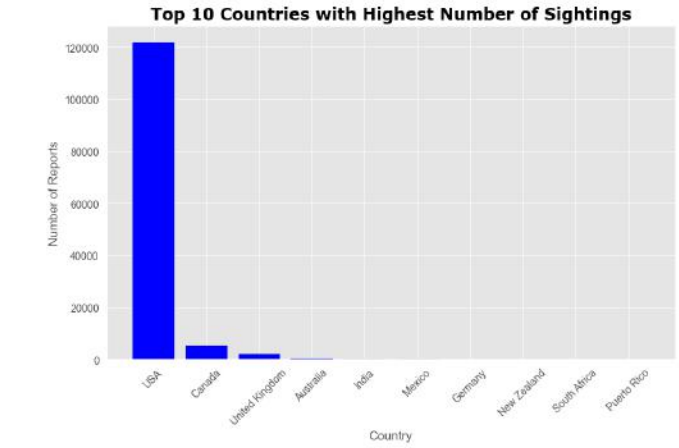
```
#====== Display:
fig.show()
```

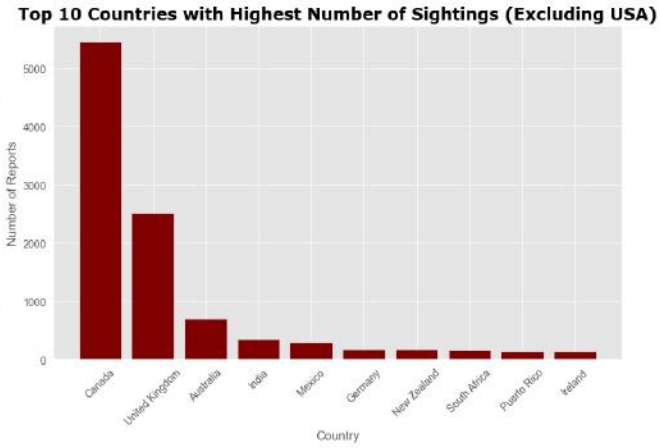
Data Observation: > `By plotting the data points as coordinates, a distinct pattern emerges, revealing the primary locations of UFO sightings around the world. The United States stands out as the top country with the highest number of reported sightings. This observation may suggest either a heightened interest in UFO visitations in the USA or a potential reluctance among individuals to report such incidents compared to the global average.` > `Additionally, a noteworthy trend is observed between the Northern and Southern Hemispheres, with a higher frequency of UFO encounters reported in the Northern Hemisphere.` > `Conversely, Africa appears to have fewer reported sightings, indicating a possible lower reporting rate or a lack of significant public attention to UFO phenomena in the region.`

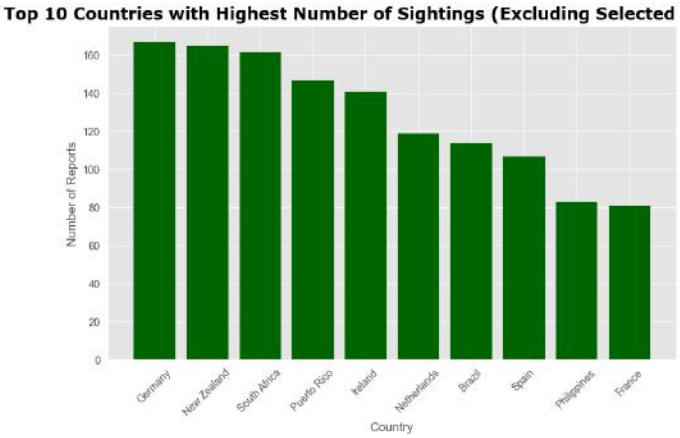
In [896…

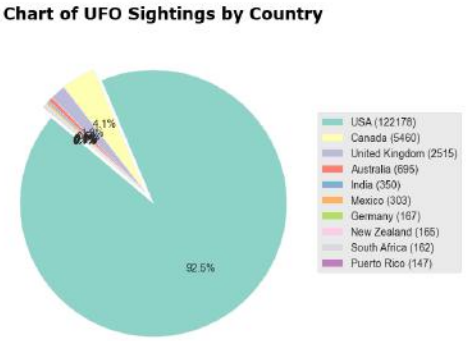
```
#====== Setting subplots showing total nr Ufo's per country

#====== Set Fig size:
fig, axes = plt.subplots(2, 2, figsize=(18, 12))

#====== Set title font settings
title_font = {
    'fontname': 'Verdana',
    'color': 'black',
    'weight': 'bold',
    'size': 16,
}

#====== Plot for Top 10 UFO Reports by Country:
country_counts = ufo_df['country'].value_counts()
top_10_countries = country_counts.head(10)

axes[0, 0].bar(top_10_countries.index, top_10_countries.values, color='blue')
axes[0, 0].set_xlabel('Country')
axes[0, 0].set_ylabel('Number of Reports')
axes[0, 0].set_title('Top 10 Countries with Highest Number of Sightings', fontdict=tit
axes[0, 0].tick_params(axis='x', rotation=45)

#====== Plot for Top 10 UFO Reports by Country (Excluding USA):
country_counts = ufo_df['country'].value_counts()
country_counts = country_counts.drop('USA', errors='ignore')
top_10_countries = country_counts.head(10)

axes[0, 1].bar(top_10_countries.index, top_10_countries.values, color='maroon')
axes[0, 1].set_xlabel('Country')
axes[0, 1].set_ylabel('Number of Reports')
axes[0, 1].set_title('Top 10 Countries with Highest Number of Sightings (Excluding USA
axes[0, 1].tick_params(axis='x', rotation=45)

#====== Plot for Top 10 UFO Reports by Country (Excluding Selected Countries):
country_counts = ufo_df['country'].value_counts()
countries_to_exclude = ['USA', 'Canada', 'United Kingdom', 'Australia', 'India', 'Mexi
filtered_country_counts = country_counts.drop(countries_to_exclude, errors='ignore')
top_10_countries = filtered_country_counts.head(10)

axes[1, 0].bar(top_10_countries.index, top_10_countries.values, color='darkgreen')
axes[1, 0].set_xlabel('Country')
axes[1, 0].set_ylabel('Number of Reports')
axes[1, 0].set_title('Top 10 Countries with Highest Number of Sightings (Excluding Sel
axes[1, 0].tick_params(axis='x', rotation=45)

#====== Plot Pie Chart of UFO Sightings by Country with Exploded Largest Portion:
country_counts = ufo_df['country'].value_counts().reset_index()
country_counts.columns = ['Country', 'Count']

#====== Select the top 10 countries for the pie chart:
top_10_countries = country_counts.head(10)

#====== Create a list to specify explosion for each slice (0 for all slices except the
explode = [0.1 if country == top_10_countries['Country'].iloc[0] else 0 for country in

#====== Create the pie chart with explosion in the lower-right subplot:
ax = axes[1, 1]
wedges, texts, autotexts = ax.pie(top_10_countries['Count'], labels=None, startangle=1

#====== Create a legend based on the country names:
legend_labels = [f"{country} ({count})" for country, count in zip(top_10_countries['Co
ax.legend(wedges, legend_labels, loc='center left', bbox_to_anchor=(1, 0, 0.5, 1))

#====== Set title:
```

```
ax.set_title('Pie Chart of UFO Sightings by Country', fontdict=title_font)

#====== Adjust subplot spacing:
plt.tight_layout()

#====== Show the subplots:
plt.show()
```







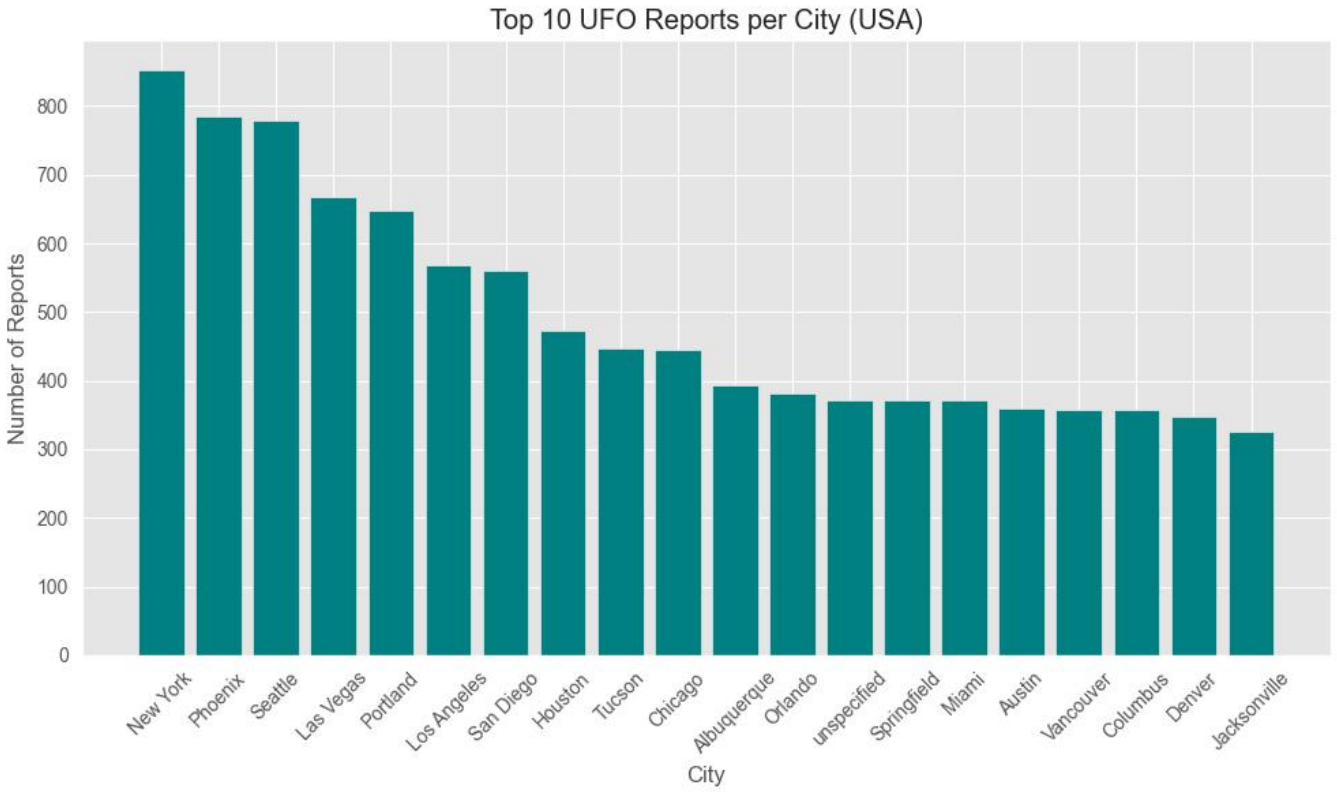


Data Observation: > `The visualizations unmistakably highlight the USA as the predominant country for UFO sightings, displaying a substantial margin compared to other nations. Canada, neighboring the USA, emerges as the second-highest in reported sightings. Notably, other countries exhibit closer margins among themselves, suggesting a discernible pattern in UFO visitations or encounters. This similarity in sighting numbers across different countries implies a consistent trend or phenomenon in UFO activities worldwide, beyond the significant prevalence observed in the USA.`

In [897…

```
#========== Plotting chart showing top cities with Ufo activity:

#====== Getting city counts:
city_counts = ufo_df['city'].value_counts()
```

```
#====== Select the top 10 cities:
top_10_cities = city_counts.head(20)

#====== Setting plot style:
plt.style.use('ggplot')

#====== Plotting bar chart:
plt.figure(figsize = (10, 6))
bars = plt.bar(x=top_10_cities.index, height=top_10_cities.values, color='teal')  # Ad

#====== Set titles and labels:
plt.xlabel('City')
plt.ylabel('Number of Reports')
plt.title('Top 10 UFO Reports per City (USA)')

#====== Rotate x-axis labels:
plt.xticks(rotation=45)

#====== Displaying:
plt.tight_layout()
plt.show()
```



Data Observation: > `The preeminent cities for UFO sightings are unsurprisingly located in the USA, consistent with the country's overall higher number of UFO reports compared to others. An intriguing observation is that these top cities are predominantly highly populated urban centers. This challenges the skepticism that busy cities might hinder the noticeability of UFO phenomena, as the data suggests otherwise.` > `Furthermore, the prominence of busy cities, particularly during the night, raises questions about the connection between UFO sightings and nightlife. The prevalence of sightings in cities with active nighttime activities indicates that UFOs, often described as lights, are more conspicuous during nocturnal hours. This prompts speculation about the potential existence of UFOs that go unnoticed during daylight due to sunlight.` > `Additionally, the mention of Phoenix brings to mind the widely reported Phoenix Lights incident on March 13, 1997. The collective sighting of unidentified flying objects in the skies over Arizona and Nevada could contribute to heightened awareness and more UFO reports in the area. The geographical characteristics of Phoenix, situated

amidst open spaces and desert terrain with minimal light pollution, might also play a role in enhancing visibility and facilitating UFO sightings`

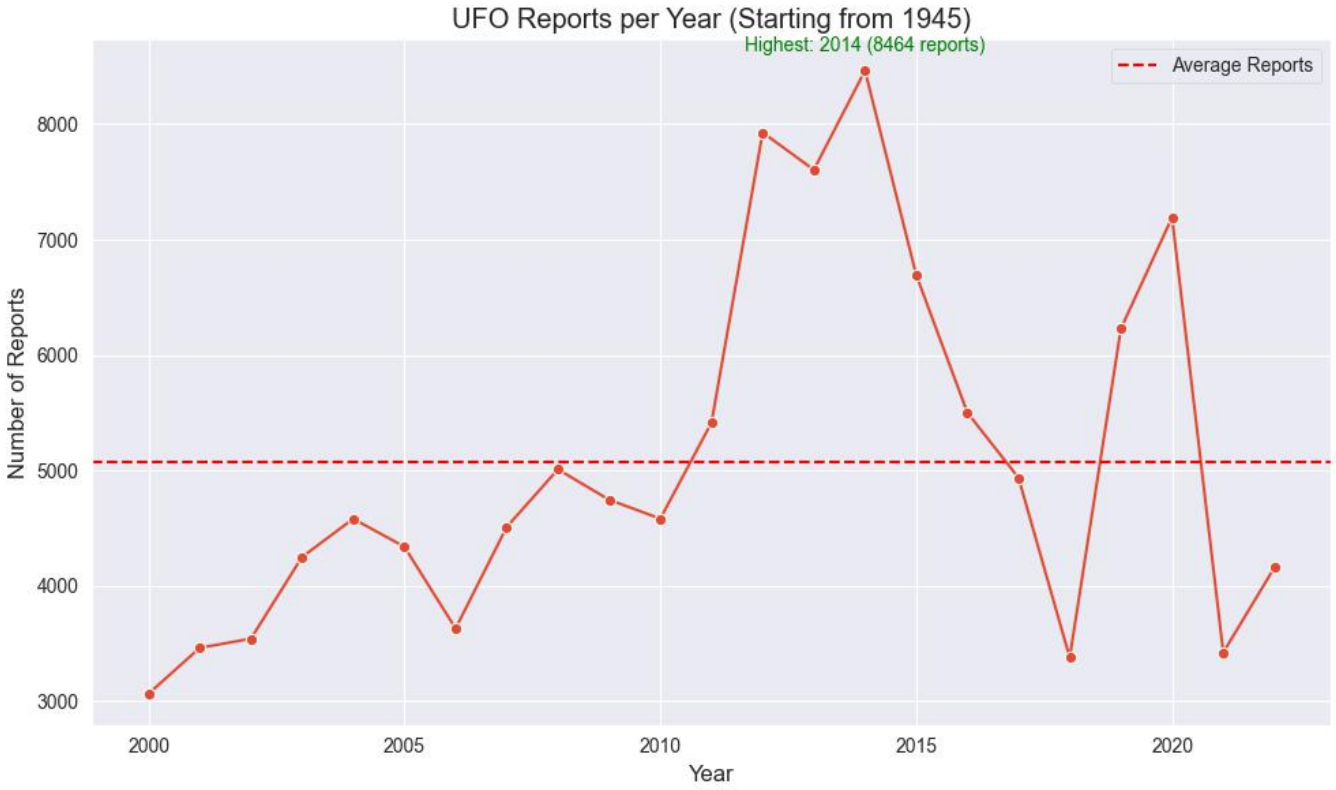
In [898…

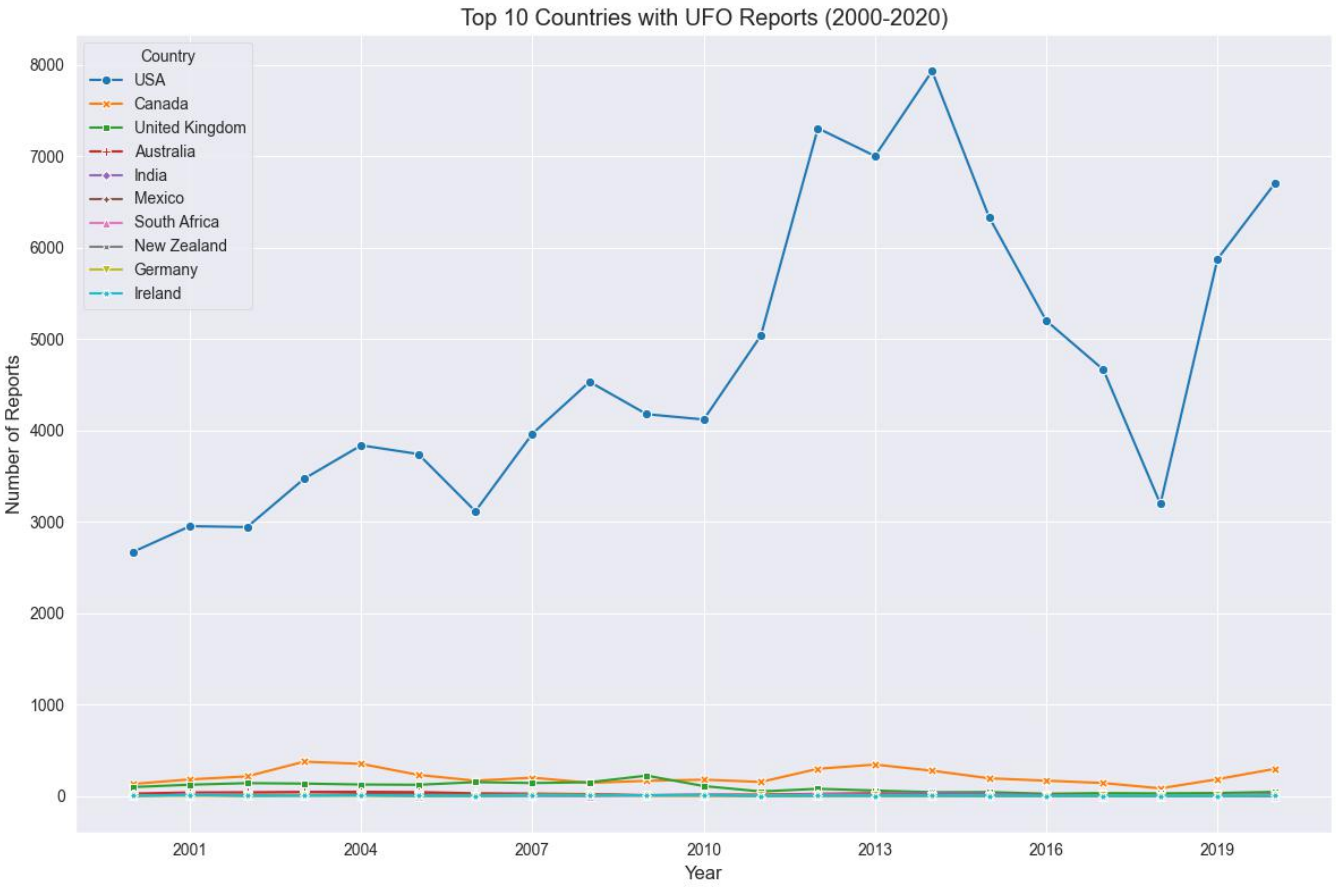
```
#========== Setting line chart to show UFO encounters timeline:

#====== Convert 'date_only' column to datetime format:
ufo_df['date_only'] = pd.to_datetime(ufo_df['date_only'])

#====== Extract the year from the 'date_only' column:
ufo_df['year'] = ufo_df['date_only'].dt.year

#====== Filter data to include years from 1945 and onwards:
ufo_df = ufo_df[ufo_df['year'] >= 1945]

#====== Count the number of reports per year:
reports_per_year = ufo_df['year'].value_counts().sort_index()

#====== Calculate the average number of reports per year:
average_reports = reports_per_year.mean()

#====== Setting the chart style:
sns.set_style("darkgrid")

#====== Plotting line chart:
plt.figure(figsize = (10, 6))
lineplot = sns.lineplot(x = reports_per_year.index, y = reports_per_year.values, marke

#====== Add a horizontal line for the average number of reports:
plt.axhline(average_reports, color = 'red', linestyle = '--', label = 'Average Reports

#====== Set title and labels:
plt.xlabel('Year')
plt.ylabel('Number of Reports')
plt.title('UFO Reports per Year (1945 - 2021)')

#====== Find the year with the highest number of reports:
highest_year = reports_per_year.idxmax()
highest_count = reports_per_year.max()

#====== Annotate the highest point with the year and count:
plt.annotate(f'Highest: {highest_year} ({highest_count} reports)', (highest_year, high
             textcoords="offset points", xytext = (0, 10), ha = 'center',color = 'red'

#====== Display the chart:
plt.legend()  # Add a legend
plt.tight_layout()
plt.show()
```

In [899…

```
#========== Plotting first chart showing UFO encounters per year (2020 upwwards):

#====== Convert 'date_only' column to datetime format:
ufo_df['date_only'] = pd.to_datetime(ufo_df['date_only'])

#====== Extract the year from the 'date_only' column:
ufo_df['year'] = ufo_df['date_only'].dt.year

#====== Filter data to include years from 2000 and onwards:
ufo_df = ufo_df[ufo_df['year'] >= 2000]

#====== Count the number of reports per year:
reports_per_year = ufo_df['year'].value_counts().sort_index()

#====== Calculate the average number of reports per year:
average_reports = reports_per_year.mean()

#====== set the chart style:
sns.set_style("darkgrid")

#====== Plotting line chart:
plt.figure(figsize = (10, 6))
lineplot = sns.lineplot(x = reports_per_year.index, y = reports_per_year.values, marke

#====== Add a horizontal line for the average number of reports:
plt.axhline(average_reports, color='red', linestyle='--', label = 'Average Reports')

#====== Setting plot titae and labels:
plt.xlabel('Year')
plt.ylabel('Number of Reports')
plt.title('UFO Reports per Year (Starting from 1945)')

#====== Find the year with the highest number of reports:
highest_year = reports_per_year.idxmax()
highest_count = reports_per_year.max()

#====== Annotate the highest point with the year and count:
plt.annotate(f'Highest: {highest_year} ({highest_count} reports)', (highest_year, high
             textcoords="offset points", xytext = (0, 10), ha = 'center',color = 'gree
```

```
#====== Display the plot:
plt.legend()  # Add a legend
plt.tight_layout()
plt.show()

#========== Plotting 2nd chart showing UFO encounters per year (between 2000 - 2020) a
#====== Convert the 'date_only' column to a datetime type:
ufo_df['date_only'] = pd.to_datetime(ufo_df['date_only'])

#====== Extract the year from the 'date_only' column:
ufo_df['year'] = ufo_df['date_only'].dt.year

#====== Filter data to include years from 2000 to 2020:
ufo_df_filtered = ufo_df[(ufo_df['year'] >= 2000) & (ufo_df['year'] <= 2020)]

#====== Count the number of reports per year and per country:
reports_per_year_and_country = ufo_df_filtered.groupby(['year', 'country']).size().uns

#====== Get the top 10 countries with the most reports over the years 2000-2020:
top_10_countries = reports_per_year_and_country.sum().nlargest(10).index

#====== Filter the data to include only the top 10 countries:
reports_per_year_and_country = reports_per_year_and_country[top_10_countries]

#====== Plotting line chart:
plt.figure(figsize = (12, 8))
sns.set_style("darkgrid")

#====== Use palette to control line colors:
palette = sns.color_palette("tab10", len(top_10_countries))
sns.lineplot(data = reports_per_year_and_country, dashes = False, markers = True, pale

#====== Set title and label:
plt.xlabel('Year')
plt.ylabel('Number of Reports')
plt.title('Top 10 Countries with UFO Reports (2000-2020)')

#====== Set the x-axis tick locator to integers
plt.gca().xaxis.set_major_locator(plt.MaxNLocator(integer = True))

#====== Display the plot:
plt.legend(title = 'Country')
plt.tight_layout()
plt.show()
```



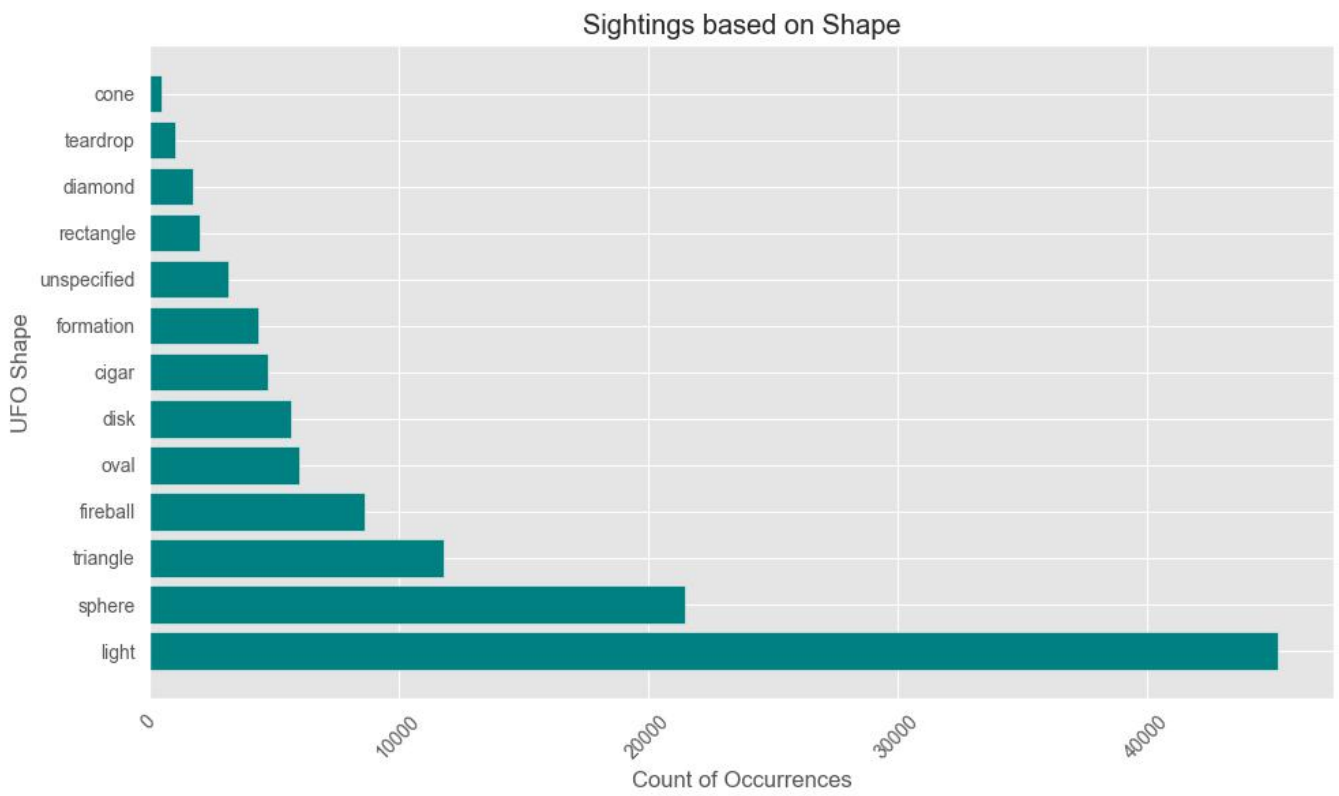


Data Observation: > `The surge in UFO sightings in the 21st century, especially leading into the 2000s, can be attributed to several factors, reflecting changes in technology, societal awareness, and global events. The sudden advancement of technology, including faster internet speeds and the ability to report sightings digitally, likely played a role. Additionally, the influence of popular culture, with movies like "Independence Day" (1996) portraying UFOs and aliens, may have sparked increased interest and awareness in the public.` > `The notable spike in sightings in 2014 could be linked to a combination of factors. The increasing prevalence of advanced video cameras and smartphones during that time may have contributed to more individuals capturing and reporting

sightings. Moreover, the peak aligns with a period of heightened global events, such as increased military activity and geopolitical tensions.` > `The period from 2010 to 2015 saw a substantial increase in sightings, prompting speculation about potential influencing factors. The aftermath of 9/11 is mentioned as a potential trigger, with people being more vigilant and alert to objects in the sky due to fear of potential terrorist attacks. Another theory suggests that the global military pressure, especially in the USA with troops deployed to the Middle East, might have attracted attention from unidentified beings or led to misinterpretations of military activities as UFO sightings.` > `It's interesting to note that this spike in sightings is more pronounced in the USA, indicating a unique pattern compared to other countries. The absence of a similar anomaly in other nations suggests that the influencing factors might be specific to the USA, potentially related to its geopolitical situation and global military involvement.` > `The mention of President Barack Obama's order in June 2014, responding to the Northern Iraq offensive, adds another layer of complexity. While it might be challenging to establish a direct correlation, geopolitical events and military activities could contribute to heightened awareness and observation of the skies, impacting the number of reported UFO sightings.`

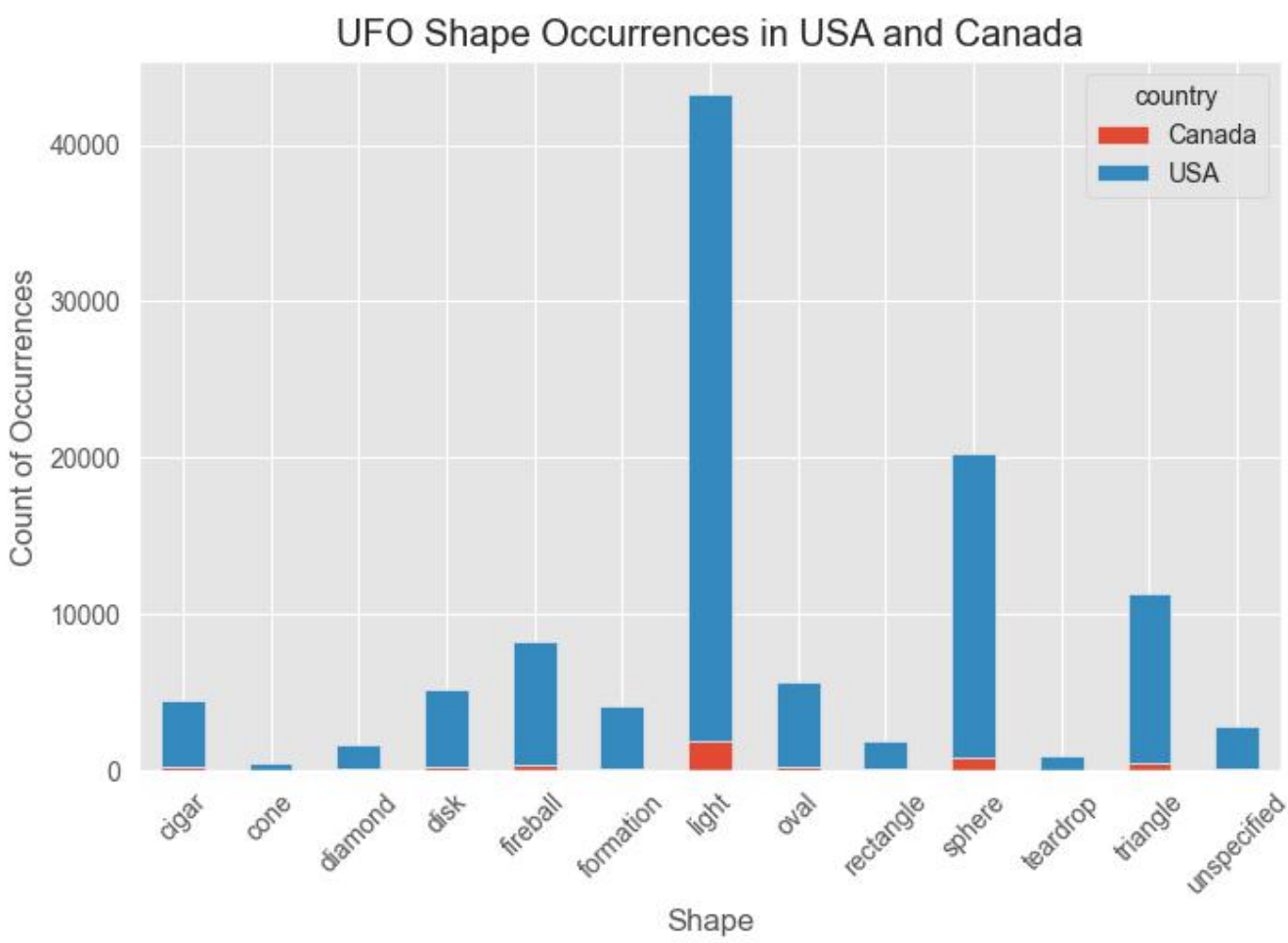
In [900…

```
#========== Plotting chart showing Ufo shape counts:

#====== Getting shapes count:
shape_counts = ufo_df['shape'].value_counts()

plt.style.use('ggplot')  # Setting plot style

#====== Plotting a horizontal bar chart:
plt.figure(figsize = (10, 6))
bars = plt.barh(y = shape_counts.index, width=shape_counts.values, color='teal')  # Ad

#====== Setting titles and labels:
plt.xlabel('Count of Occurrences')
plt.ylabel('UFO Shape')
plt.title('Sightings based on Shape')

#====== Rotate x-axis labels:
plt.xticks(rotation=45)

# ====== Display:
plt.tight_layout()
plt.show()
```



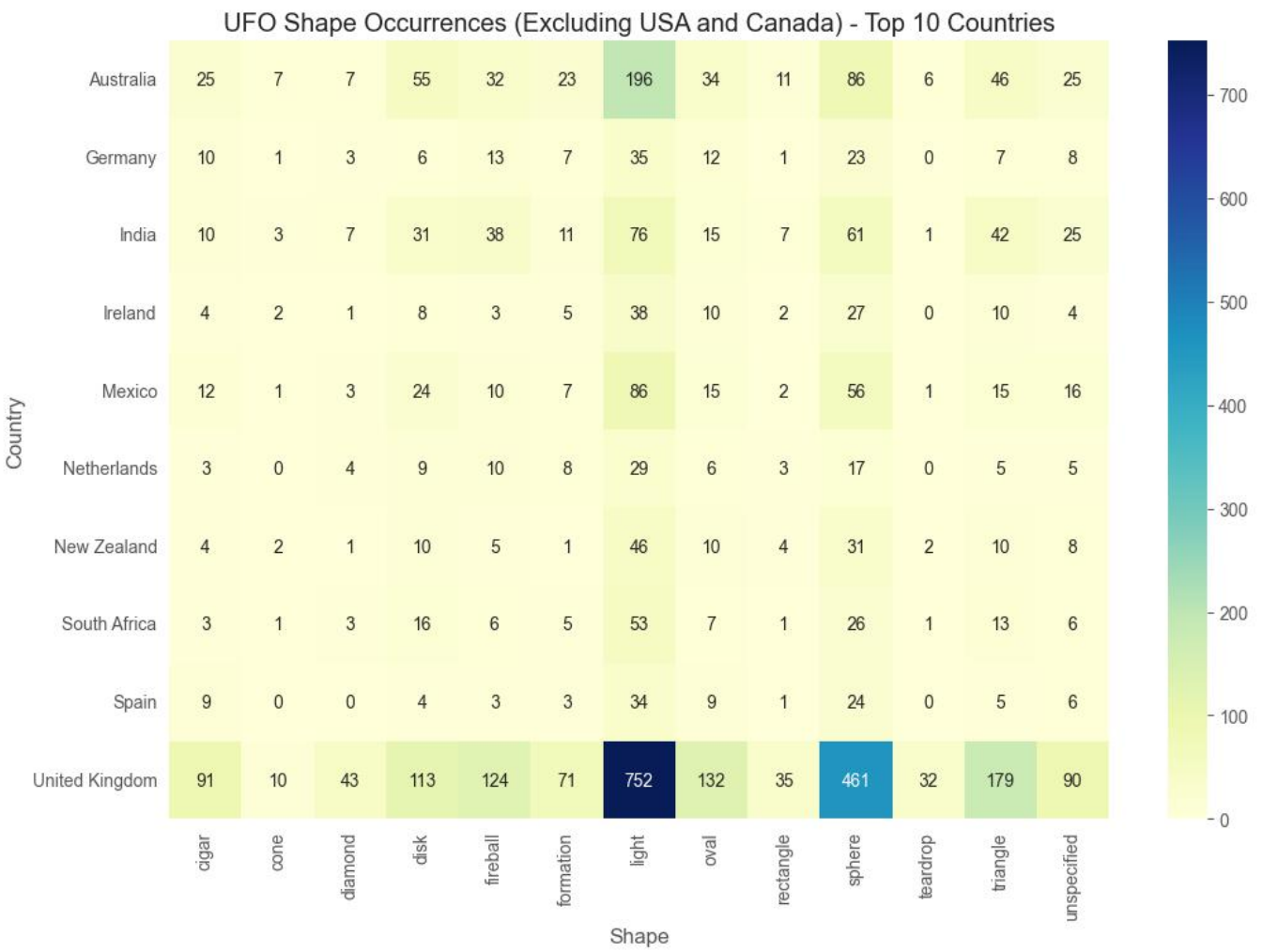
In [901…

```
#========== Setting chart displaying nr of UFO sightings per country:

#====== Extract top cities counts:
top_10_countries = ufo_df['country'].value_counts().head(10).index

#====== Filter the DataFrame for 'USA' and 'Canada':
filtered_df = ufo_df[ufo_df['country'].isin(['USA', 'Canada'])]

#====== Pivot the data to count UFO shape occurrences by country and transpose:
pivot_df = filtered_df.pivot_table(index='country', columns = 'shape', aggfunc = 'size

#====== Plotting bar chart:
ax = pivot_df.plot(kind = 'bar', stacked = True, figsize = (8, 5))
plt.title('UFO Shape Occurrences in USA and Canada')
plt.xlabel('Shape')
plt.ylabel('Count of Occurrences')
plt.xticks(rotation = 45)

#====== Display plot:
plt.show()
```



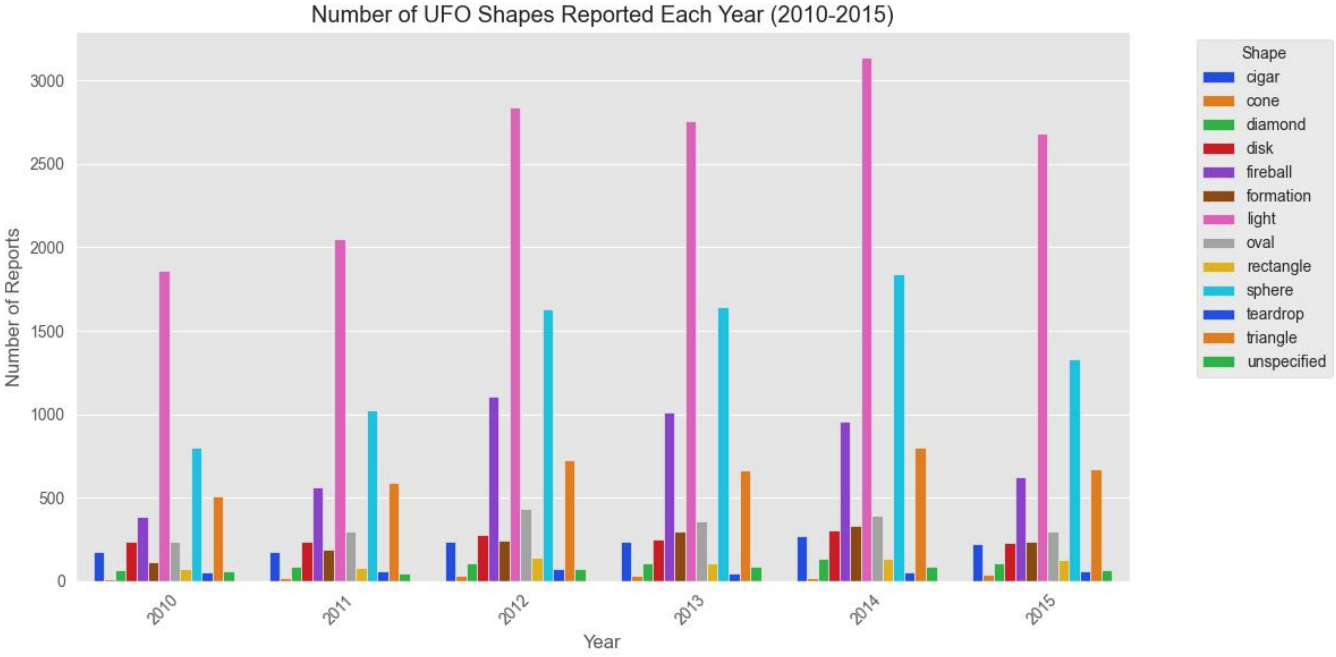
In [902…

```python
import seaborn as sns

#========== Setting chart displaying nr of UFO sightings per country:
#====== Extract top cities counts:
top_10_countries = ufo_df['country'].value_counts().head(12).index  # Include the top

#====== Filter the DataFrame for all countries except 'USA' and 'Canada':
filtered_df = ufo_df[~ufo_df['country'].isin(['USA', 'Canada'])]

#====== Extract top 10 countries after excluding 'USA' and 'Canada':
remaining_top_10_countries = filtered_df['country'].value_counts().head(10).index

#====== Filter the DataFrame for the top 10 countries after exclusion:
final_filtered_df = filtered_df[filtered_df['country'].isin(remaining_top_10_countries

#====== Pivot the data to count UFO shape occurrences by country:
pivot_df = final_filtered_df.pivot_table(index='country', columns='shape', aggfunc='si

#====== Plotting heatmap:
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_df, cmap="YlGnBu", annot=True, fmt="d", cbar=True)
plt.title('UFO Shape Occurrences (Excluding USA and Canada) - Top 10 Countries')
plt.xlabel('Shape')
plt.ylabel('Country')

#====== Display plot:
plt.show()
```



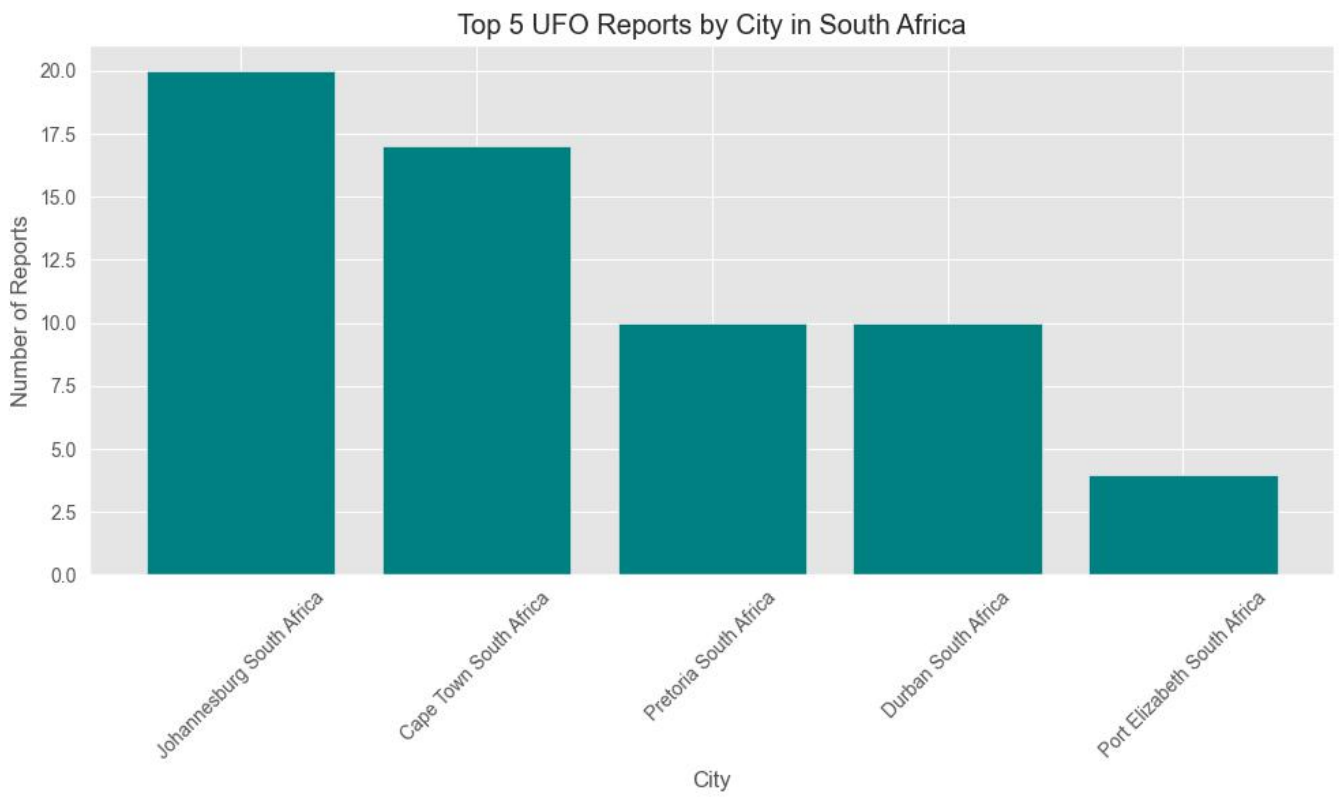
In [903…

```
#========= Plotting another plot showing UFO shapes per year:

#====== Convert the 'date_only' column to a datetime type:
ufo_df['date_only'] = pd.to_datetime(ufo_df['date_only'])

#====== Filtering records(year range 2010-2015):
filtered_df = ufo_df[(ufo_df['date_only'].dt.year >= 2010) & (ufo_df['date_only'].dt.y

#====== Group by 'year' and 'shape' and calculate counts:
year_shape_counts = filtered_df.groupby([filtered_df['date_only'].dt.year, 'shape']).s

#====== Setting barplot:
plt.figure(figsize = (12, 6))
sns.barplot(x = 'date_only', y='count', hue='shape', data = year_shape_counts, palette
plt.xlabel('Year')
plt.ylabel('Number of Reports')
plt.title('Number of UFO Shapes Reported Each Year (2010-2015)')

#====== Legend re-adjust position:
plt.legend(title='Shape', loc='upper right', bbox_to_anchor = (1.2, 1.0))

plt.xticks(rotation = 45)

#====== Display:
plt.tight_layout()
plt.show()
```



In [904…

```
#========== Plotting chart showing Ufo sightings per city in South Africa:

#====== Filtering to extract values from country South Africa only:
south_africa_df = ufo_df[ufo_df['country'] == 'South Africa']

#====== Count nr of reports per city:
city_counts = south_africa_df['city'].value_counts()

#====== Select the top 5 cities in South Africa:
top_5_cities = city_counts.head(5)

#====== Setting style for chart:
plt.style.use('ggplot')

#===== Plotting barplot:
plt.figure(figsize = (10, 6))
bars = plt.bar(x = top_5_cities.index, height = top_5_cities.values, color = 'teal')

#===== Add labels and title
plt.xlabel('City')
plt.ylabel('Number of Reports')
plt.title('Top 5 UFO Reports by City in South Africa')

#====== Rotate x-axis labels:
plt.xticks(rotation = 45)

#====== Display the plot:
plt.tight_layout()
plt.show()
```

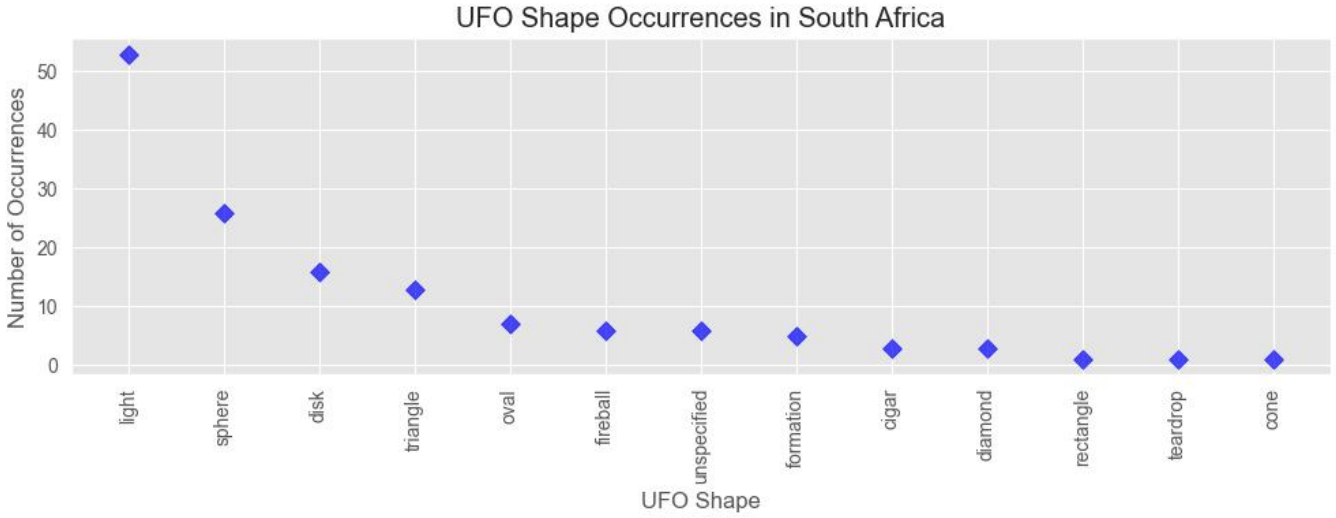


In [905…

```python
#====== Defining function that returns most common Ufo shape:
def find_most_common_shape(df, target_country):
    # Filter the DataFrame to include only the target country
    country_data = df[df['country'] == target_country]

    if country_data.empty:
        return None, 0  # Return None if no data for the country

    # Find the mode of the 'shape' column (most common shape)
    most_common_shape = country_data['shape'].mode().values[0]

    # Count the occurrences of the most common shape
    occurrences = country_data[country_data['shape'] == most_common_shape]['shape'].co

    return most_common_shape, occurrences
```

In [906…

```python
#======== Initiatilising function above:

#====== Filter the DataFrame to include only 'South Africa'
sa_data = ufo_df[ufo_df['country'] == 'South Africa']

#====== Count the occurrences of each unique shape in South Africa
shape_counts = sa_data['shape'].value_counts()

#====== Plotting scatterplot:
plt.figure(figsize = (10, 4))
plt.scatter(shape_counts.index, shape_counts.values, s=50, c='blue', alpha=0.7, marker
plt.xlabel('UFO Shape')
plt.ylabel('Number of Occurrences')
plt.title('UFO Shape Occurrences in South Africa')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.grid(True)

#====== Display:
plt.tight_layout()
plt.show()
```



In [907…

```python
#========== Plotting heat map/area chart for USA States and nr of ufo encounters:

#====== Filtering data for USA only:
usa_ufo_df = ufo_df[ufo_df['country'] == 'USA']

#====== Calculate the number of UFO reports per state
state_counts = usa_ufo_df['state'].value_counts().reset_index()
state_counts.columns = ['state', 'num_reports']

#====== Create the choropleth map
fig = px.choropleth(
    state_counts,
    locations='state',  # Column containing state codes or names
    locationmode='USA-states',  # Use USA state abbreviations as location mode
    color='num_reports',  # Column containing the values to be plotted
    scope='usa',  # Map scope (USA)
    hover_name='state',  # Column for hover text (state names)
    title='UFO Reports per USA State',  # Title of the map
    color_continuous_scale='Viridis'
)

#====== Display the map:
fig.show()
```

In [908…

```
#========== Plotting heat map/area chart for USA States and nr of ufo encounters per t

#====== Extract the year from the 'date_time' column:
usa_ufo_df['year'] = pd.to_datetime(usa_ufo_df['date_only']).dt.year

#====== Calculate the number of UFO reports per state and year
state_year_counts = usa_ufo_df.groupby(['state', 'year']).size().reset_index(name='num

#====== Create the choropleth map
fig = px.choropleth(
    state_year_counts,
    locations='state',  # Column containing state codes or names
    locationmode='USA-states',  # Use USA state abbreviations as location mode
    color='num_reports',  # Column containing the values to be plotted
    scope='usa',  # Map scope (USA)
    hover_name='state',  # Column for hover text (state names)
    animation_frame='year',  # Column for animation (year)
    title='UFO Reports per USA State Over Time'  # Title of the map
)

#====== Display map:
fig.show()
```

In [909…

```
#====== Using NL Toolkit to display frequency of ufo shape texts

#====== Download stopwords library:
nltk.download('stopwords')

#====== Tokenizing the 'summary' text:
ufo_df['summary'] = ufo_df['summary'].str.lower().str.split() # Convert to lowercase

#====== Removing stopwords:
stop_words = set(stopwords.words('english'))
ufo_df['summary'] = ufo_df['summary'].apply(lambda tokens: [word for word in tokens if

#====== Setting counter to count word frequecies:
word_counts = Counter()

for summary_tokens in ufo_df['summary']:
    word_counts.update(summary_tokens) # Apply tokens

#====== Finding the most common words:
most_common_words = word_counts.most_common(30)  # Setting to 20 most common

#====== Display the most common words and their frequencies
print("Top 50 most common words:\n")

for word, count in most_common_words:
    print(f"\n{word}: {count}")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\warri\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Top 50 most common words:


lights: 25226

light: 24958

bright: 20041

sky: 19449

object: 18275

moving: 14079

orange: 12538

white: 12494

red: 11641
```

```
saw: 9769

seen: 7886

flying: 7757

shaped: 7576

like: 7437

sky.: 6473

craft: 6356

two: 6204

objects: 5832

ufo: 5828

large: 5642

3: 5596

one: 5356

note:: 5290

pd: 5223

nuforc: 5203

across: 5198

green: 5184

2: 5077

hovering: 4998

night: 4929
```

Final closing:

* The following outlines the final summary of findings:

UFO Reports

[i] In conclusion, this analysis of UFO reports from the National UFO Research Center highlights compelling insights gained through rigorous data cleaning and analytics. Visualizations reveal distinct patterns, with the USA prominently leading in sightings, possibly reflecting heightened reporting or genuine interest. Contrary to expectations, urban areas, particularly at night, exhibit significant UFO activity, challenging assumptions about visibility amidst city lights. Events like the Phoenix Lights and

societal factors such as technological advancements contribute to spikes in reports, notably in the early 2000s and around 2014, especially within the USA. These findings underscore the complex interplay of socio-cultural influences and technological developments on global UFO sightings