About Dataset

Context This dataset contains over 80,000 reports of UFO sightings over the last century.

Content

There are two versions of this dataset: scrubbed and complete. The complete data includes entries where the location of the sighting was not found or blank (0.8146%) or have an erroneous or blank time (8.0237%). Since the reports date back to the 20th century, some older data might be obscured. Data contains city, state, time, description, and duration of each sighting.

https://www.kaggle.com/datasets/NUFORC/ufo-sightings/data?select=scrubbed.csv (https://www.kaggle.com/datasets/NUFORC/ufo-sightings/data?select=scrubbed.csv)

In [488]:

```
#import required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import seaborn as sns
sns.set_style('whitegrid')
%matplotlib inline
```

In [489]:

```
#import data

df = pd.read_csv('Desktop/data/scrubbed.csv')
```

In [490]:

```
#obtain summary of data frame

df.info
```

Out[490]:
```
<bound method DataFrame.info of                datetime                    city state country     shape  \
0      10/10/1949 20:30              san marcos    tx      us  cylinder
1      10/10/1949 21:00            lackland afb    tx     NaN     light
2      10/10/1955 17:00  chester (uk/england)     NaN      gb    circle
3      10/10/1956 21:00                    edna    tx      us    circle
4      10/10/1960 20:00                 kaneohe    hi      us     light
...                 ...                     ...   ...     ...       ...
80327    9/9/2013 21:15               nashville    tn      us     light
80328    9/9/2013 22:00                   boise    id      us    circle
80329    9/9/2013 22:00                    napa    ca      us     other
80330    9/9/2013 22:20                  vienna    va      us    circle
80331    9/9/2013 23:00                  edmond    ok      us     cigar

      duration (seconds) duration (hours/min)  \
0                   2700           45 minutes
1                   7200              1-2 hrs
2                     20           20 seconds
3                     20             1/2 hour
4                    900           15 minutes
...                  ...                  ...
80327              600.0           10 minutes
80328             1200.0           20 minutes
80329             1200.0                 hour
80330                5.0            5 seconds
80331             1020.0           17 minutes

                                                comments date posted  \
0      This event took place in early fall around 194...   4/27/2004
1      1949 Lackland AFB&#44 TX.  Lights racing acros...  12/16/2005
2      Green/Orange circular disc over Chester&#44 En...   1/21/2008
3      My older brother and twin sister were leaving ...   1/17/2004
4      AS a Marine 1st Lt. flying an FJ4B fighter/att...   1/22/2004
...                                                  ...         ...
80327  Round from the distance/slowly changing colors...   9/30/2013
80328  Boise&#44 ID&#44 spherical&#44 20 min&#44 10 r...   9/30/2013
80329                                      Napa UFO&#44   9/30/2013
80330  Saw a five gold lit cicular craft moving fastl...   9/30/2013
80331  2 witnesses 2  miles apart&#44 Red &amp; White...   9/30/2013

         latitude  longitude
0      29.8830556 -97.941111
1        29.38421 -98.581082
2            53.2  -2.916667
3      28.9783333 -96.645833
4      21.4180556 -157.803611
...           ...        ...
80327   36.165833  -86.784444
80328   43.613611 -116.202500
80329   38.297222 -122.284444
80330   38.901111  -77.265556
80331   35.652778  -97.477778

[80332 rows x 11 columns]>
```

In [491]:

```
#displays first few rows of the data frame

df.head()
```

Out[491]:

	datetime	city	state	country	shape	duration (seconds)	duration (hours/min)	comments	
0	10/10/1949 20:30	san marcos	tx	us	cylinder	2700	45 minutes	This event took place in early fall around 194...	4.
1	10/10/1949 21:00	lackland afb	tx	NaN	light	7200	1-2 hrs	1949 Lackland AFB&#44 TX. Lights racing acros...	12.
2	10/10/1955 17:00	chester (uk/england)	NaN	gb	circle	20	20 seconds	Green/Orange circular disc over Chester&#44 En...	1.
3	10/10/1956 21:00	edna	tx	us	circle	20	1/2 hour	My older brother and twin sister were leaving ...	1.
4	10/10/1960 20:00	kaneohe	hi	us	light	900	15 minutes	AS a Marine 1st Lt. flying an FJ4B fighter/att...	1.

In [492]:
```
#displays last few rows of the data frame

df.tail()
```

Out[492]:

	datetime	city	state	country	shape	duration (seconds)	duration (hours/min)	comments	p
80327	9/9/2013 21:15	nashville	tn	us	light	600.0	10 minutes	Round from the distance/slowly changing colors...	9/30
80328	9/9/2013 22:00	boise	id	us	circle	1200.0	20 minutes	Boise&#44 ID&#44 spherical&#44 20 min&#44 10 r...	9/30
80329	9/9/2013 22:00	napa	ca	us	other	1200.0	hour	Napa UFO&#44	9/30
80330	9/9/2013 22:20	vienna	va	us	circle	5.0	5 seconds	Saw a five gold lit cicular craft moving fastl...	9/30
80331	9/9/2013 23:00	edmond	ok	us	cigar	1020.0	17 minutes	2 witnesses 2 miles apart&#44 Red & White...	9/30

In [493]:
```
#variability shows how spread out the data is from each other in a particul
#dropping low and high variability is good because it does not contribute w

variability= pd.DataFrame(unique).sort_values(by=0, ascending=False)
variability
```

Out[493]:

	0
datetime	69586
latitude	23312
city	19900
longitude	19455
duration (hours/min)	8349
duration (seconds)	706
date posted	317
state	67
shape	29
country	5

In [494]:
```
#drop comments because it has a lot of unique variables that we don't need

df.drop(['comments'], axis=1, inplace=True)
```

In [495]:
```
#calculates the number of unique values for each column
unique=(df.nunique(axis=0))
unique
```

Out[495]:
```
datetime                  69586
city                      19900
state                        67
country                       5
shape                        29
duration (seconds)          706
duration (hours/min)       8349
date posted                 317
latitude                  23312
longitude                 19455
dtype: int64
```

In [496]:
```
#check data types for each columns
df.dtypes
```

Out[496]:
```
datetime                   object
city                       object
state                      object
country                    object
shape                      object
duration (seconds)         object
duration (hours/min)       object
date posted                object
latitude                   object
longitude                 float64
dtype: object
```

In [497]:
```
#determine why latitude is not a float  like longitude
#scan unique values and determine anything out of place
#output shows normal float values

unique_latitudes = df['latitude'].unique()
print(unique_latitudes)
```

```
['29.8830556' '29.38421' '53.2' ... 50.465843 34.367594 34.1013889]
```

In [498]:
```
#convert latitude column into a float type

df['latitude'] = pd.to_numeric(df['latitude'], errors='coerce')
```

In [499]:
```
#convert datetime to datetime object

df['datetime'] = pd.to_datetime(df['datetime'], errors='coerce')
```

In [500]:
```
#convert state and country into categorical types

df['state'] = df['state'].astype('category')
df['country'] = df['country'].astype('category')

df.dtypes
```

Out[500]:
```
datetime                  datetime64[ns]
city                              object
state                           category
country                         category
shape                             object
duration (seconds)                object
duration (hours/min)              object
date posted                       object
latitude                         float64
longitude                        float64
dtype: object
```

In [501]:
```
#checks for missing values

df.isnull().sum()
```

Out[501]:
```
datetime                 694
city                       0
state                   5797
country                 9670
shape                   1932
duration (seconds)         0
duration (hours/min)       0
date posted                0
latitude                   1
longitude                  0
dtype: int64
```

In [502]:
```
#checking shape distribution

df['shape'].value_counts()
```

Out[502]:
```
shape
light        16565
triangle      7865
circle        7608
fireball      6208
other         5649
unknown       5584
sphere        5387
disk          5213
oval          3733
formation     2457
cigar         2057
changing      1962
flash         1328
rectangle     1297
cylinder      1283
diamond       1178
chevron        952
egg            759
teardrop       750
cone           316
cross          233
delta            7
round            2
crescent         2
pyramid          1
flare            1
hexagon          1
dome             1
changed          1
Name: count, dtype: int64
```

In [503]:
```
# Uppercase the state and country columns

df['state'] = df['state'].str.upper()
df['country'] = df['country'].str.upper()
```

In [504]:
```
# Capitalize all column titles

df.columns = df.columns.str.capitalize()
```

In [505]:

```
# Convert datetime column to a datetime object

df['Datetime'] = pd.to_datetime(df['Datetime'], errors='coerce')

# Create new date and time columns
#this will create 2 columns, one for date and the other time

df['Date'] = df['Datetime'].dt.date
df['Time'] = df['Datetime'].dt.time

#drop the original Datetime column

df = df.drop(['Datetime'], axis=1)
```

In [506]:

```
# this code reorders the data frame to get the last two columns to be moved
#columns from the original order

df = df[['Date', 'Time'] + [col for col in df.columns if col not in ['Date'
```

In [507]:

```
df.head()
```

Out[507]:

	Date	Time	City	State	Country	Shape	Duration (seconds)	Duration (hours/min)	Date posted
0	1949-10-10	20:30:00	san marcos	TX	US	cylinder	2700	45 minutes	4/27/2004
1	1949-10-10	21:00:00	lackland afb	TX	NaN	light	7200	1-2 hrs	12/16/2005
2	1955-10-10	17:00:00	chester (uk/england)	NaN	GB	circle	20	20 seconds	1/21/2008
3	1956-10-10	21:00:00	edna	TX	US	circle	20	1/2 hour	1/17/2004
4	1960-10-10	20:00:00	kaneohe	HI	US	light	900	15 minutes	1/22/2004

In [508]:
```
#Check if there are any missing values

print(df.isnull().sum())
```

```
Date                     694
Time                     694
City                       0
State                   5797
Country                 9670
Shape                   1932
Duration (seconds)         0
Duration (hours/min)       0
Date posted                0
Latitude                   1
Longitude                  0
dtype: int64
```

What areas of the country are most likely to have UFO sightings?

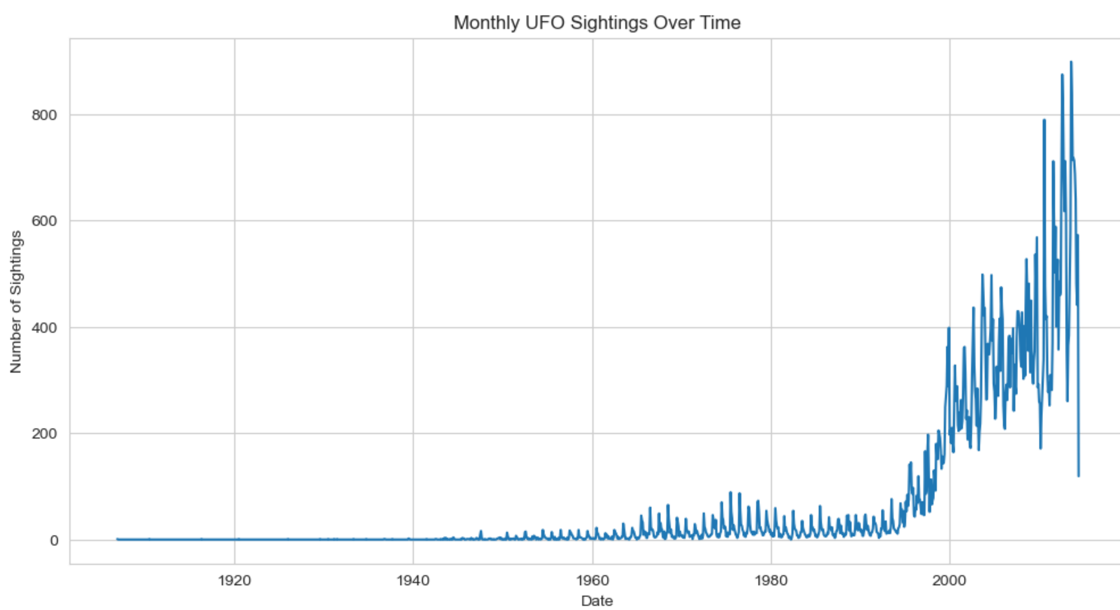
In [509]:
```
df['Country'].value_counts().head(5)

#shows that United States, Canada, United Kingdom of Great Britain, Austrai
```

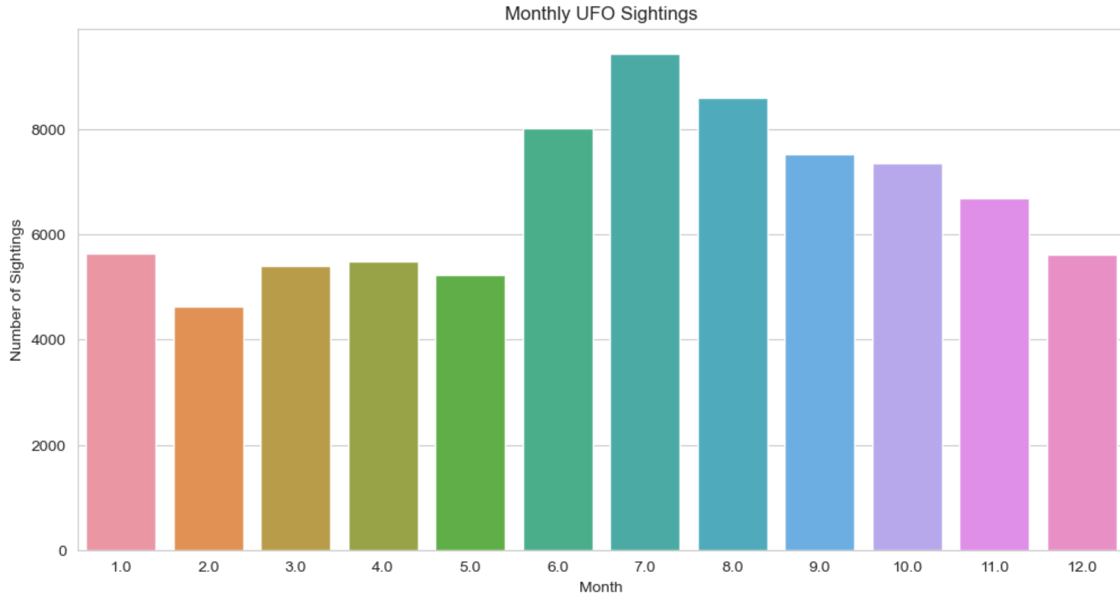
Out[509]:
```
Country
US    65114
CA     3000
GB     1905
AU      538
DE      105
Name: count, dtype: int64
```

Are there any trends in UFO sightings over time? Do they tend to be clustered or seasonal?

In [510]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'Date' is in datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Extract year and month for additional analysis
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
#--------------------------------

# Create a new DataFrame with counts per month
monthly_counts = df.resample('M', on='Date').size().reset_index(name='Numbe

# Plot the number of sightings over time (monthly)
plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Number of Sightings', data=monthly_counts)
plt.title('Monthly UFO Sightings Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Sightings')
plt.show()


# Plot monthly sightings
plt.figure(figsize=(12, 6))
sns.countplot(x='Month', data=df)
plt.title('Monthly UFO Sightings')
plt.xlabel('Month')
plt.ylabel('Number of Sightings')
plt.show()
```



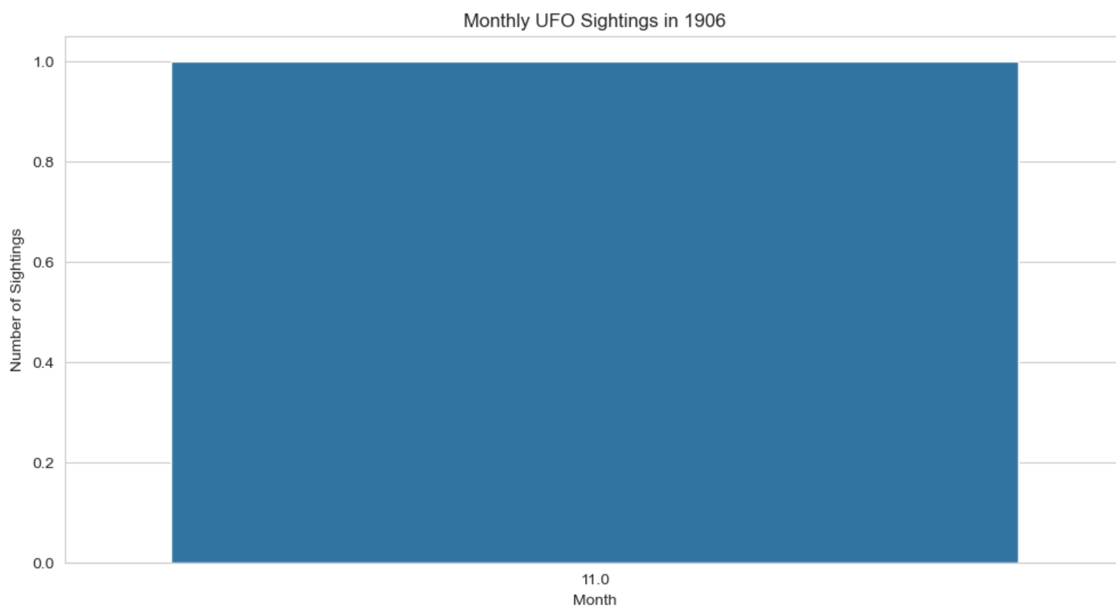


For the first graph: There looks to be a trend in UFO sightings over the years. The graph shows that around the late 1990s the number of UFO sightings has increase by a large amount since the start of the first sighting collected in 1906.

For the second graph: The second graph shows a collection of all the years in the data set starting from 1906 all the way to 2014, and taking the average of UFO sighting and inputing them into each month. This graph shows that for the month of June, July, and August there have been more sightings than any other months. This suggests that UFO sightings coulds be a summer trend.

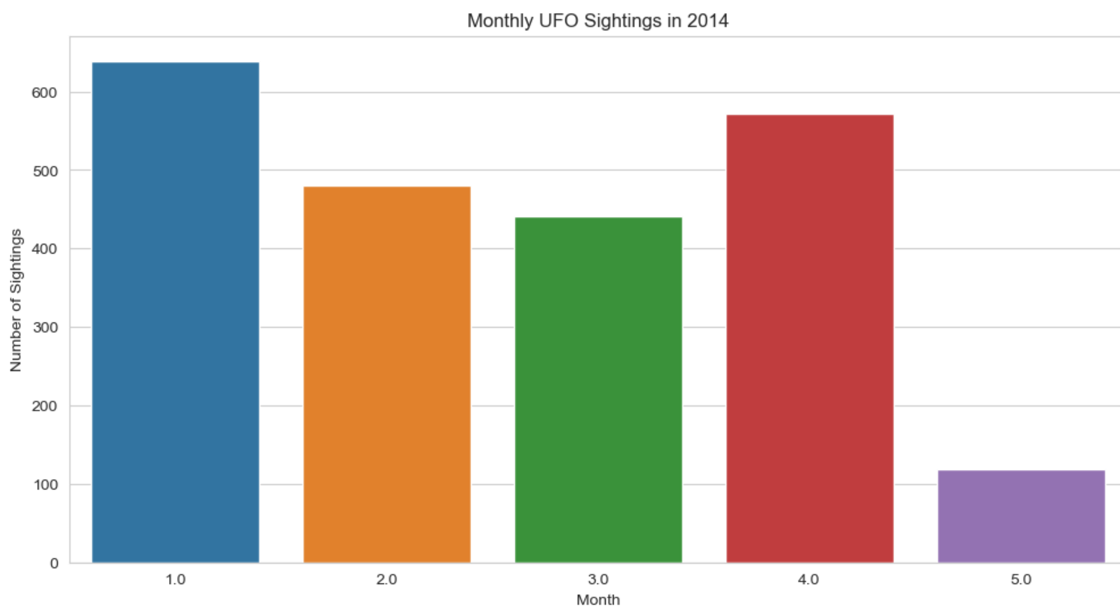
Is there a difference in monthly sightings between the start of this data collect(1906) vs the last set of data collected(2014)?

In [511]:
```
# Assuming 'Date' is in datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Extract year information
df['Year'] = df['Date'].dt.year

# Filter data for 1906
df_year_1906 = df[df['Year'] == 1906]

# Plot monthly sightings for the specific year
plt.figure(figsize=(12, 6))
sns.countplot(x='Month', data=df_year_1906)
plt.title('Monthly UFO Sightings in 1906')
plt.xlabel('Month')
plt.ylabel('Number of Sightings')
plt.show()
```



This shows that there has only been one UFO sighting in the month of June.

In [512]:
```
# Assuming 'Date' is in datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Extract year information
df['Year'] = df['Date'].dt.year

# Filter data for 2013
df_year_2014 = df[df['Year'] == 2014]

# Plot monthly sightings for the specific year
plt.figure(figsize=(12, 6))
sns.countplot(x='Month', data=df_year_2014)
plt.title('Monthly UFO Sightings in 2014')
plt.xlabel('Month')
plt.ylabel('Number of Sightings')
plt.show()
```



This shows that January, February, March, April, and May has the most UFO sightins in the year 2014. Also, there is a significant amount of more sightings in 2014 compared to the one in 1906.

Is there a specific year with the most UFO sightings?

In [513]:
```
df['Date'].max()
```

Out[513]: Timestamp('2014-05-08 00:00:00')

In [514]:
```
df['Date'].min()
```

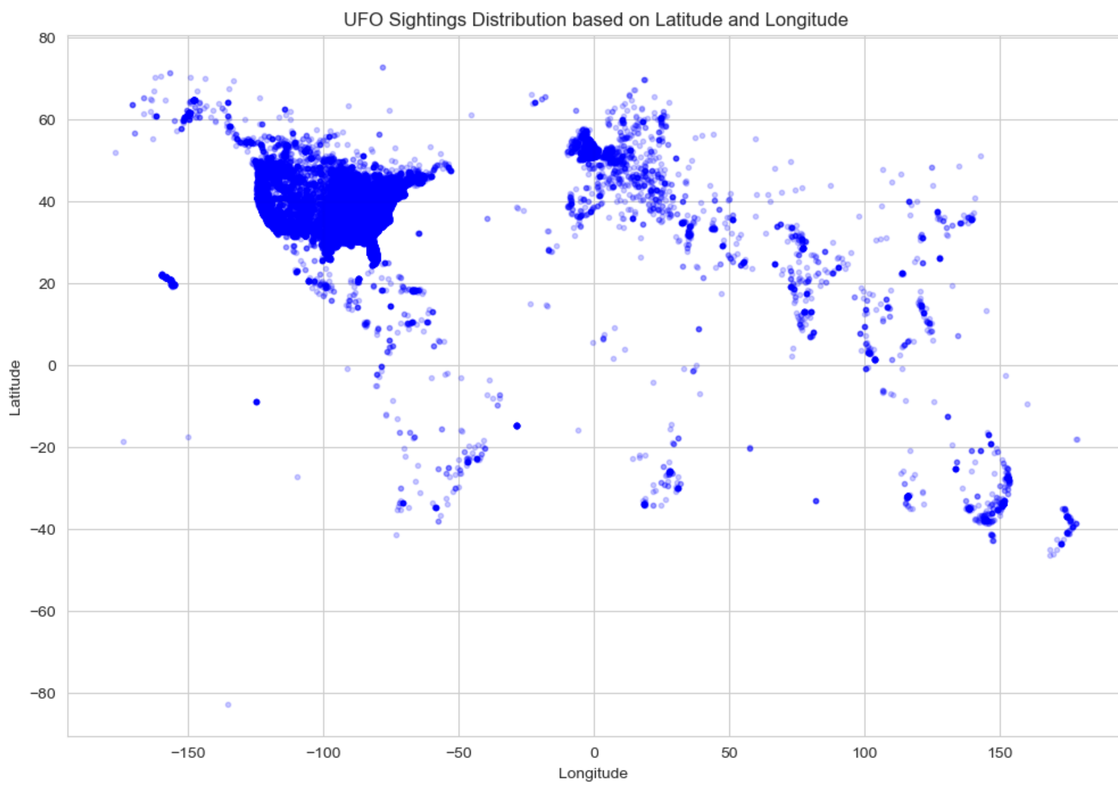
Out[514]: Timestamp('1906-11-11 00:00:00')

This shows that the year with the most UFO sightings is 2014 and year with the least sightings is 1906.

Show what continents has the most UFO sightings?

In [515]:

```
#dremove rows with missing latitude and longitude

df = df.dropna(subset=['Latitude', 'Longitude '])

#Plot a scatter plot plotting the latitude and longitude from the dataset

plt.figure(figsize=(12, 8))
plt.scatter(df['Longitude '], df['Latitude'], alpha=0.2, marker='.', color=
plt.title('UFO Sightings Distribution based on Latitude and Longitude')
plt.xlabel('Longitude ')
plt.ylabel('Latitude')
plt.grid(True)
plt.show()
```



This scatter plot graphs the longitude and latitude and shows the continents with the most sightings are: North America, Europe, some in Asia, and some in Australia.

statistical description

In [516]:

```
#includes both numeric and non-numeric columns for statistical description
df.describe(include='all')
```

Out[516]:

	Date	Time	City	State	Country	Shape	Duration (seconds)	Duration (hours/min)
count	79637	79637	80331	74534	70662	78399	80331	80331
unique	NaN	1390	19899	67	5	29	706	8349
top	NaN	22:00:00	seattle	CA	US	light	300	5 minutes
freq	NaN	4617	525	9655	65114	16565	7070	4716
mean	2004-06-01 15:32:34.895337600	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	1906-11-11 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	2001-08-11 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	2006-11-28 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	2011-06-25 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	2014-05-08 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Correlation Coefficient

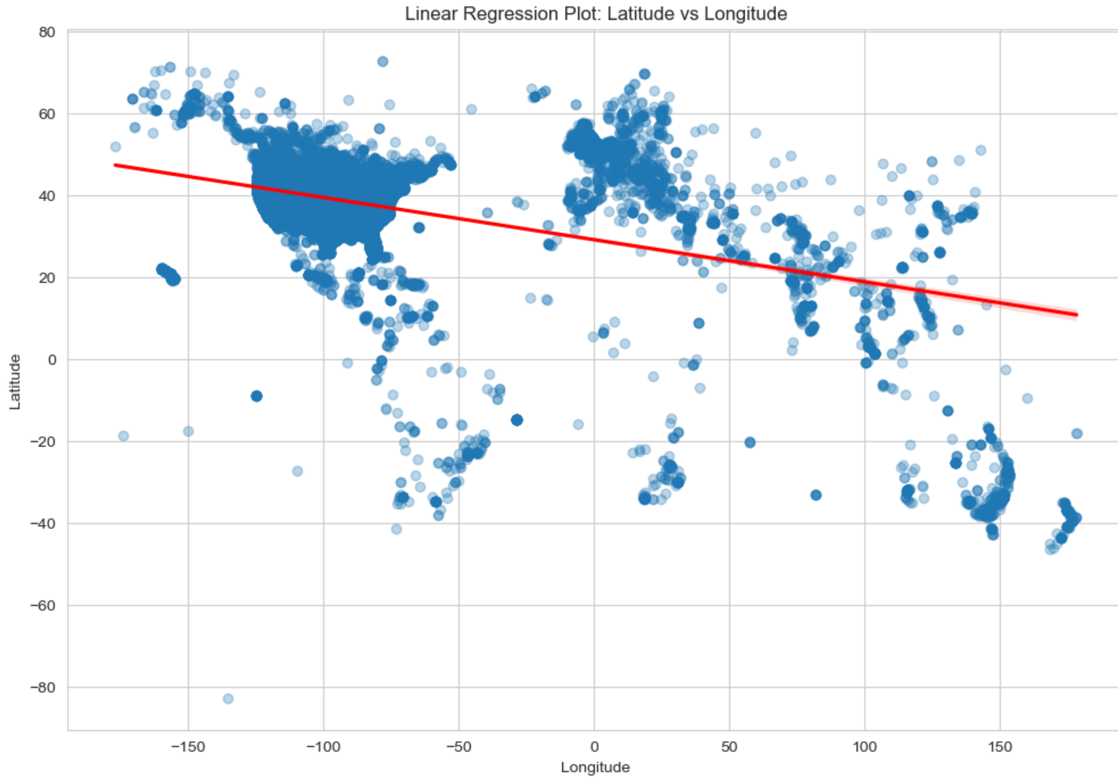
In [517]:

```
import pandas as pd

# Assuming 'df' is your DataFrame
numeric_columns = ['Latitude', 'Longitude']  # Exclude 'Duration (seconds)'

# Remove extra space from column names
df.columns = df.columns.str.strip()

# Convert the specified numeric columns to numeric type, coercing errors
for col in numeric_columns:
    df[col] = pd.to_numeric(df[col], errors='coerce')

# Drop rows with NaN values after conversion
df = df.dropna(subset=numeric_columns)

# Calculate correlation matrix
correlation_matrix = df[numeric_columns].corr()

# Display the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)
```

```
Correlation Matrix:
           Latitude  Longitude
Latitude   1.000000  -0.390219
Longitude -0.390219   1.000000
```

In [518]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

#assign the 3 columns to numeric_columns
numeric_columns = ['Latitude', 'Longitude', 'Duration (seconds)']

# Remove extra space from column names
df.columns = df.columns.str.strip()

# Convert the specified numeric columns to numeric type, coercing errors(cc
for col in numeric_columns:
    df[col] = pd.to_numeric(df[col], errors='coerce')

# Drop rows with NaN values after conversion
df = df.dropna(subset=numeric_columns)

# Create a linear regression plot
plt.figure(figsize=(12, 8))

# You can choose any two numeric columns for x and y
x_column = 'Longitude'
y_column = 'Latitude'

# Plot the scatter plot with the regression line
sns.regplot(x=x_column, y=y_column, data=df, scatter_kws={'alpha':0.3}, lir

# Set labels and title
plt.title(f'Linear Regression Plot: {y_column} vs {x_column}')
plt.xlabel(x_column)
plt.ylabel(y_column)

plt.show()
```



Hypothesis Testing

Null Hypothesis:

The mean latitude of UFO sightings is the same across all geographic locations.

Alternate Hypothesis:

There is a significant difference in the mean latitude of UFO sightings across different geographic locations.

In [519]:

```
from scipy.stats import ttest_1samp

# Assuming 'df' is your DataFrame with columns 'latitude' and 'longitude'
# Replace with your actual column names

# Perform one-sample t-test
t_stat, p_value = ttest_1samp(df['Latitude'], popmean=df['Latitude'].mean()

# Display the results
print("One-Sample T-Test Results:")
print(f"T-statistic: {t_stat}")
print(f"P-value: {p_value}")

# Interpret the results
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis. There is a significant difference ir
else:
    print("Fail to reject the null hypothesis. There is no significant diff
```

```
One-Sample T-Test Results:
T-statistic: 0.0
P-value: 1.0
Fail to reject the null hypothesis. There is no significant difference in
mean latitude.
```

The t-statistic measures how far the sample mean (mean latitude in this case) is from the null hypothesis mean (a specified value or the population mean). A t-statistic of 0.0 suggests that the sample mean is exactly equal to the null hypothesis mean.

A p-value of 1.0 means that there is a very high probability of observing a t-statistic as extreme as the one obtained, even if there is no actual difference between the sample mean and the null hypothesis mean.

Not have enough evidence to conclude that there is a significant difference in the mean latitude of UFO sightings

Conclusion

The analysis of latitude and longitude does not provide sufficient evidence to support a significant pattern or clustering of UFO sightings. The data indicates that UFO sightings are spread out fairly evenly across different locations on the map.

In [ ]: