In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data = pd.read_csv("C:/Users/windows/Desktop/LWDA/r1/sort-minRange.csv")
```

In [4]:
```
data.describe()
```

Out[4]:

	NORAD_CAT_ID_1	DSE_1	NORAD_CAT_ID_2	DSE_2	TCA_RANGE	TCA_REI
count	67507.00000	67507.000000	67507.000000	67507.000000	67507.000000	
mean	48482.29551	3.940780	43392.833691	4.298445	3.333688	
std	7032.87838	2.061438	22680.649614	2.625188	1.176109	
min	900.00000	0.096000	12.000000	0.115000	0.000000	
25%	43887.00000	2.185000	32066.000000	2.372000	2.501000	
50%	48861.00000	3.938000	46273.000000	4.183000	3.535000	
75%	53868.00000	5.689500	56186.000000	5.952000	4.324000	
max	58201.00000	32.331000	270287.000000	36.209000	5.000000	

In [7]:
```
nullvalues=pd.DataFrame(df.isnull().sum(),columns=['count'])
nullvalues['percent']=round((nullvalues['count']/df.shape[0])*100,2)
nullvalues.sort_values('percent',ascending=False)
```

Out[7]:

	count	percent
NORAD_CAT_ID_1	0	0.0
OBJECT_NAME_1	0	0.0
DSE_1	0	0.0
NORAD_CAT_ID_2	0	0.0
OBJECT_NAME_2	0	0.0
DSE_2	0	0.0
TCA	0	0.0
TCA_RANGE	0	0.0
TCA_RELATIVE_SPEED	0	0.0
MAX_PROB	0	0.0
DILUTION	0	0.0
TCA_time	0	0.0

In [ ]:
```
# Hence There is no null values.
```

In [ ]:
```
# It seems like all the Continuous variables have Outliers, But we can't delete
# As it can lead to the result distortion.
```

In [8]:
```
data.OBJECT_NAME_1.value_counts()
```

Out[8]:
```
OBJECT_NAME_1
OBJECT C [+]             164
OBJECT A [+]             154
MOVE-II [+]              142
AAUSAT-II [P]            140
OBJECT B [+]             134
                        ...
ONEWEB-0637 [+]            1
STARLINK-5099 [+]          1
SHERPA-LTE1 [+]            1
NUSAT-7 (SOPHIE) [+]       1
STARLINK-2118 [+]          1
Name: count, Length: 7693, dtype: int64
```

In [9]:
```
data.OBJECT_NAME_2.value_counts()
```

Out[9]:
```
OBJECT_NAME_2
FENGYUN 1C DEB [-]     5396
COSMOS 2251 DEB [-]    3140
UNKNOWN [-]            1073
DELTA 1 DEB [-]        1044
CZ-6A DEB [-]          1024
                       ...
INS-2TD [+]               1
STARLINK-4643 [+]         1
STARLINK-5383 [+]         1
DUBAISAT-1 [+]            1
STARLINK-5347 [+]         1
Name: count, Length: 6787, dtype: int64
```

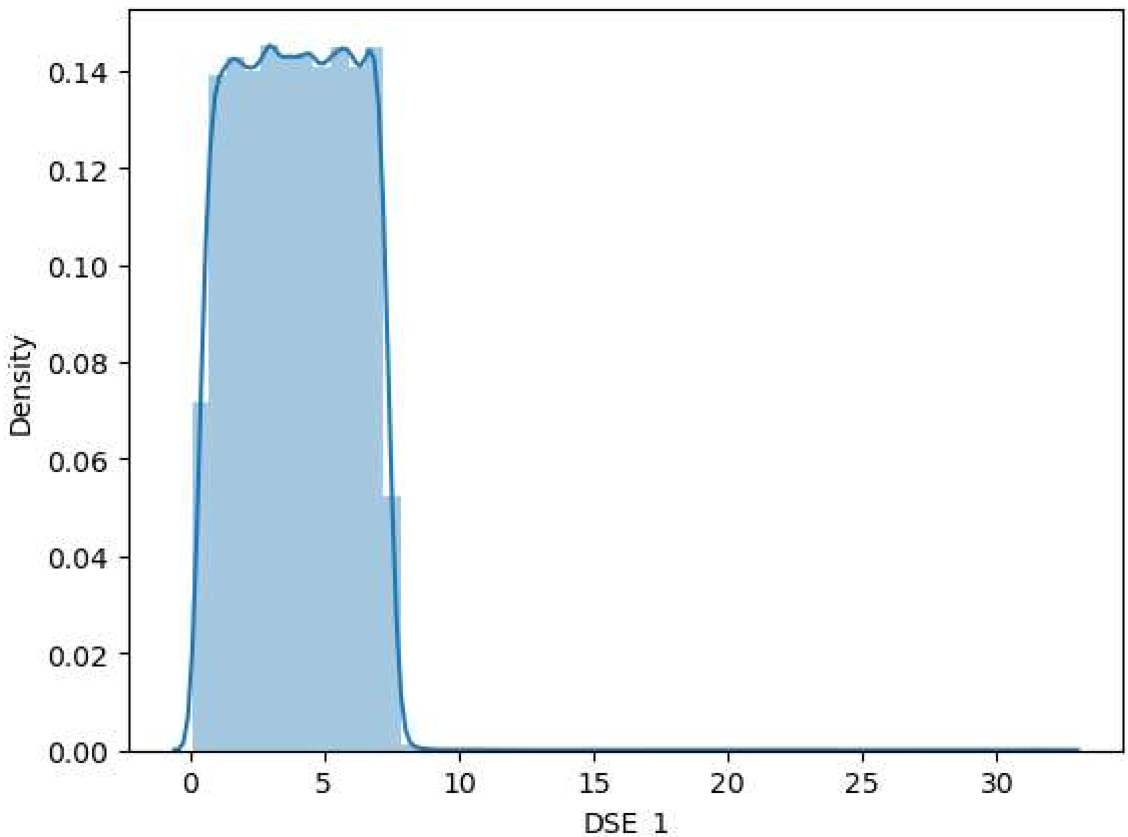
In [ ]:
```
#So,Here we understand that so many satellites are repeated again and again.
```

Exploratory Data Analysis

In [10]:
```
sns.distplot(data.DSE_1, kde = 'true')
```

```
C:\Users\windows\AppData\Local\Temp\ipykernel_11892\622275522.py:1: UserWarni
ng:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gis
t.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(data.DSE_1, kde = 'true')
```

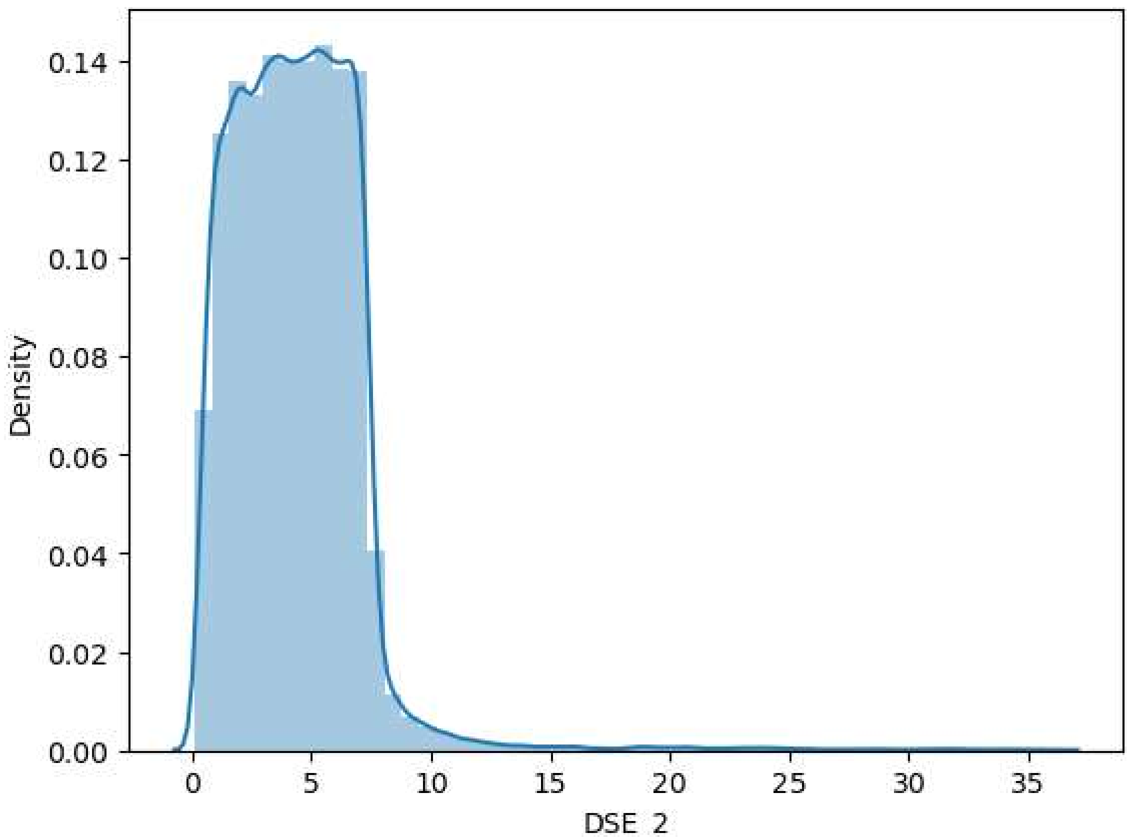
Out[10]:
```
<Axes: xlabel='DSE_1', ylabel='Density'>
```



In [11]:
```
sns.distplot(data.DSE_2, kde = 'true')
```

```
C:\Users\windows\AppData\Local\Temp\ipykernel_11892\2379141943.py:1: UserWarn
ing:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gis
t.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(data.DSE_2, kde = 'true')
```

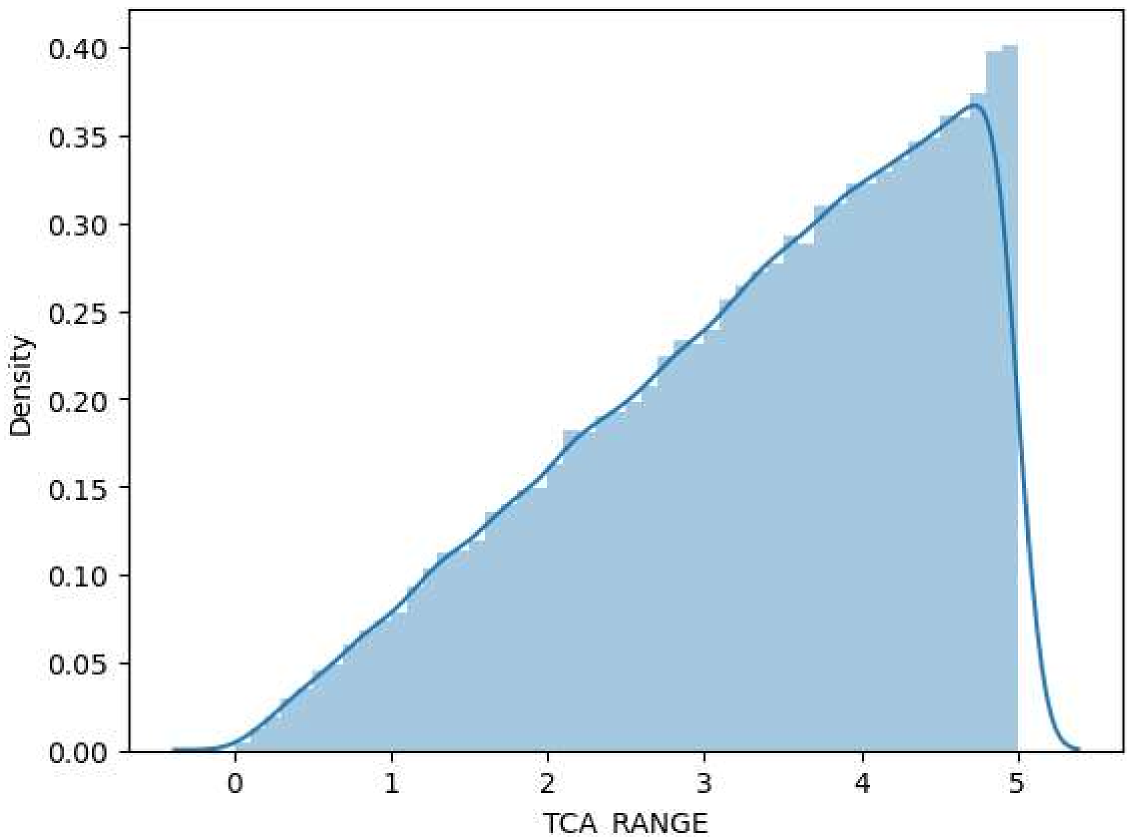
Out[11]:
```
<Axes: xlabel='DSE_2', ylabel='Density'>
```



In [12]:
```
sns.distplot(data.TCA_RANGE, kde = 'true')
```

```
C:\Users\windows\AppData\Local\Temp\ipykernel_11892\880883894.py:1: UserWarni
ng:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gis
t.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(data.TCA_RANGE, kde = 'true')
```

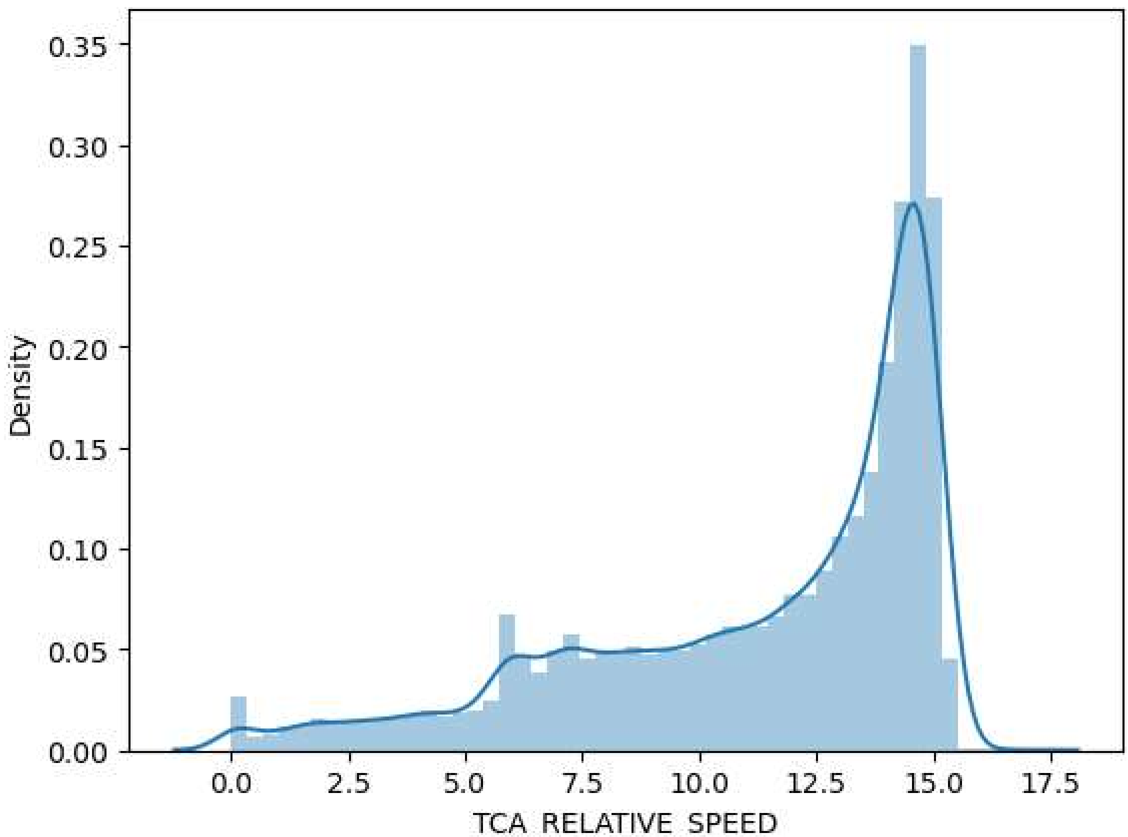
Out[12]:
```
<Axes: xlabel='TCA_RANGE', ylabel='Density'>
```



In [13]:
```
sns.distplot(data.TCA_RELATIVE_SPEED, kde = 'true')
```

```
C:\Users\windows\AppData\Local\Temp\ipykernel_11892\1019884293.py:1: UserWarn
ing:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gis
t.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(data.TCA_RELATIVE_SPEED, kde = 'true')
```

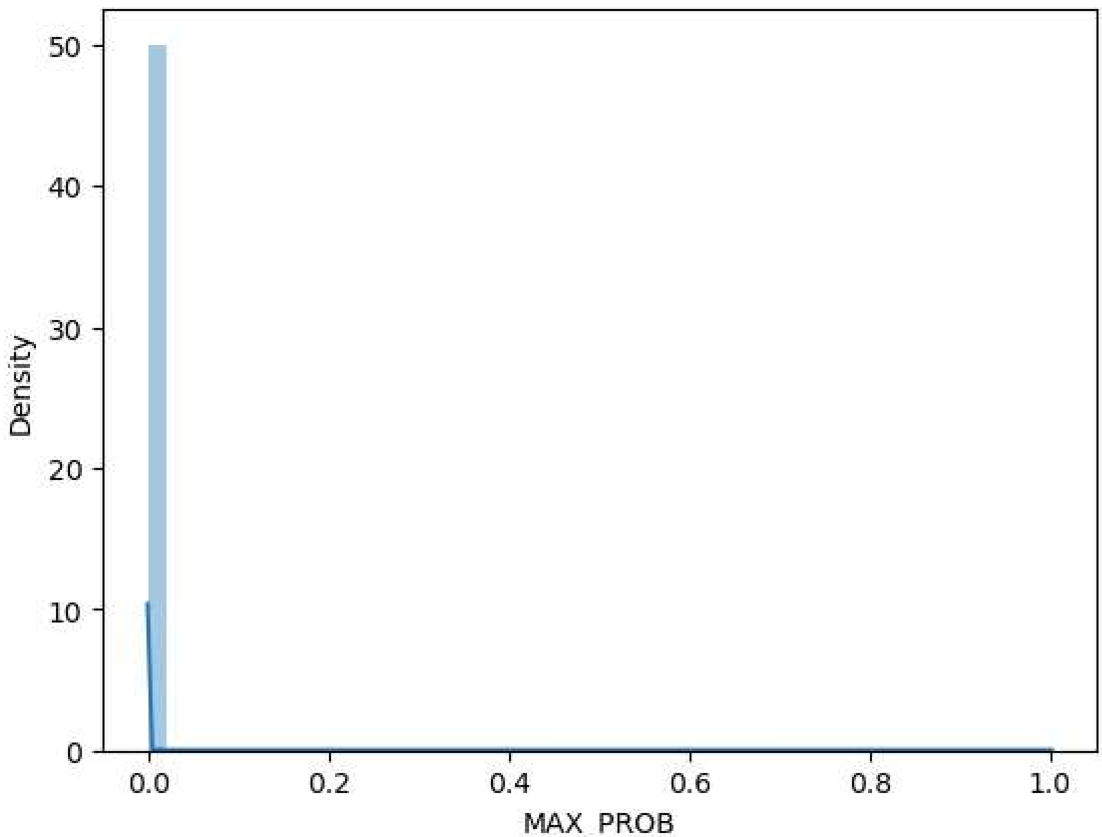
Out[13]:
```
<Axes: xlabel='TCA_RELATIVE_SPEED', ylabel='Density'>
```



In [15]:
```
sns.distplot(data.MAX_PROB, kde = 'true')
```

```
C:\Users\windows\AppData\Local\Temp\ipykernel_11892\3669083729.py:1: UserWarn
ing:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gis
t.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(data.MAX_PROB, kde = 'true')
```

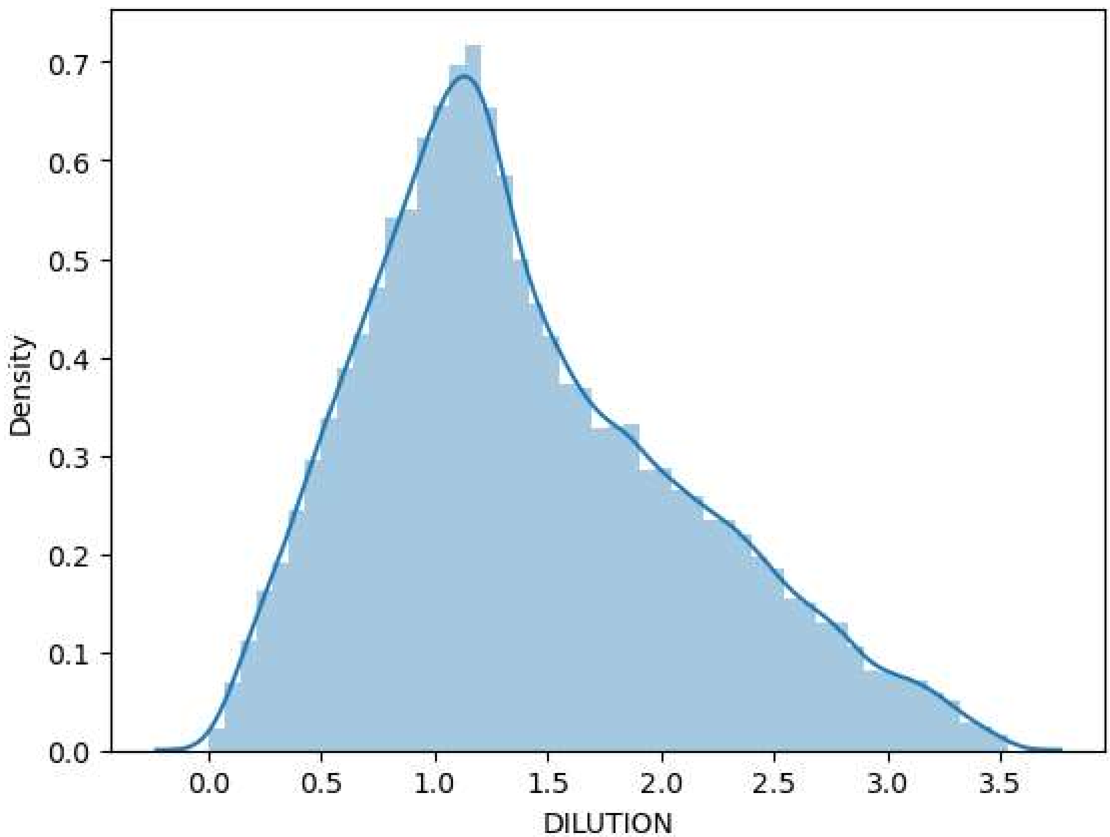
Out[15]: <Axes: xlabel='MAX_PROB', ylabel='Density'>



In [16]:
```
sns.distplot(data.DILUTION, kde = 'true')
```

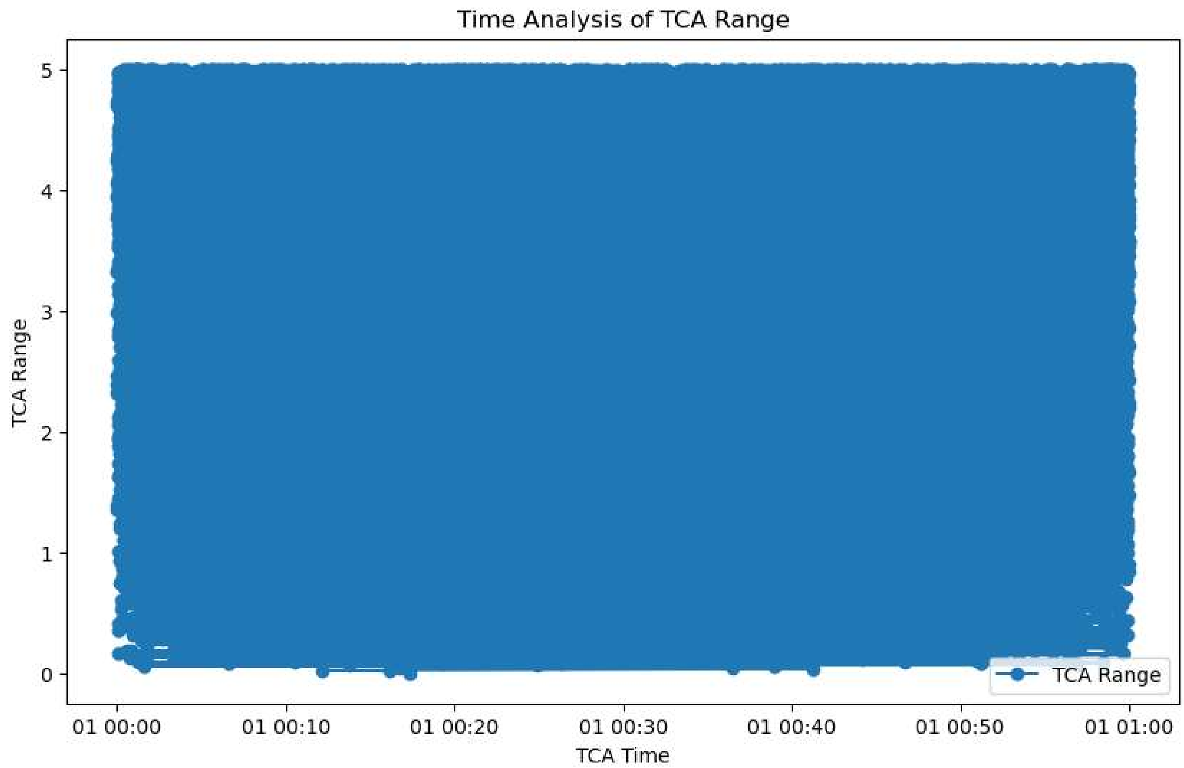
```
C:\Users\windows\AppData\Local\Temp\ipykernel_11892\1094680811.py:1: UserWarn
ing:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gis
t.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(data.DILUTION, kde = 'true')
```

Out[16]: <Axes: xlabel='DILUTION', ylabel='Density'>



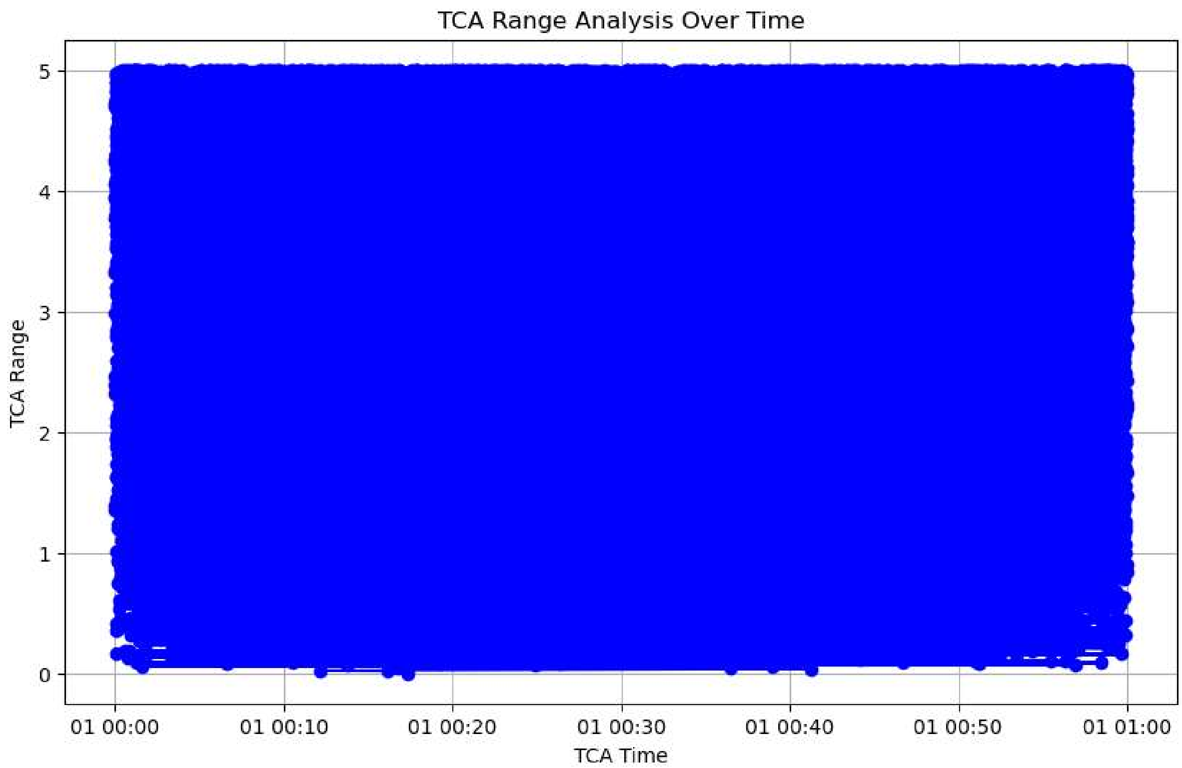
In [ ]:
```
##Time Analysis: Investigate the distribution of conjunctions over time. Identi
```

In [17]:
```
df = pd.DataFrame(data)

# Convert 'TCA' column to datetime format
df['TCA'] = pd.to_datetime(df['TCA'], format='%M:%S.%f')

# Plotting time analysis
plt.figure(figsize=(10, 6))
plt.plot(df['TCA'], df['TCA_RANGE'], marker='o', linestyle='-', label='TCA Rang
plt.xlabel('TCA Time')
plt.ylabel('TCA Range')
plt.title('Time Analysis of TCA Range')
plt.legend()
plt.show()
```



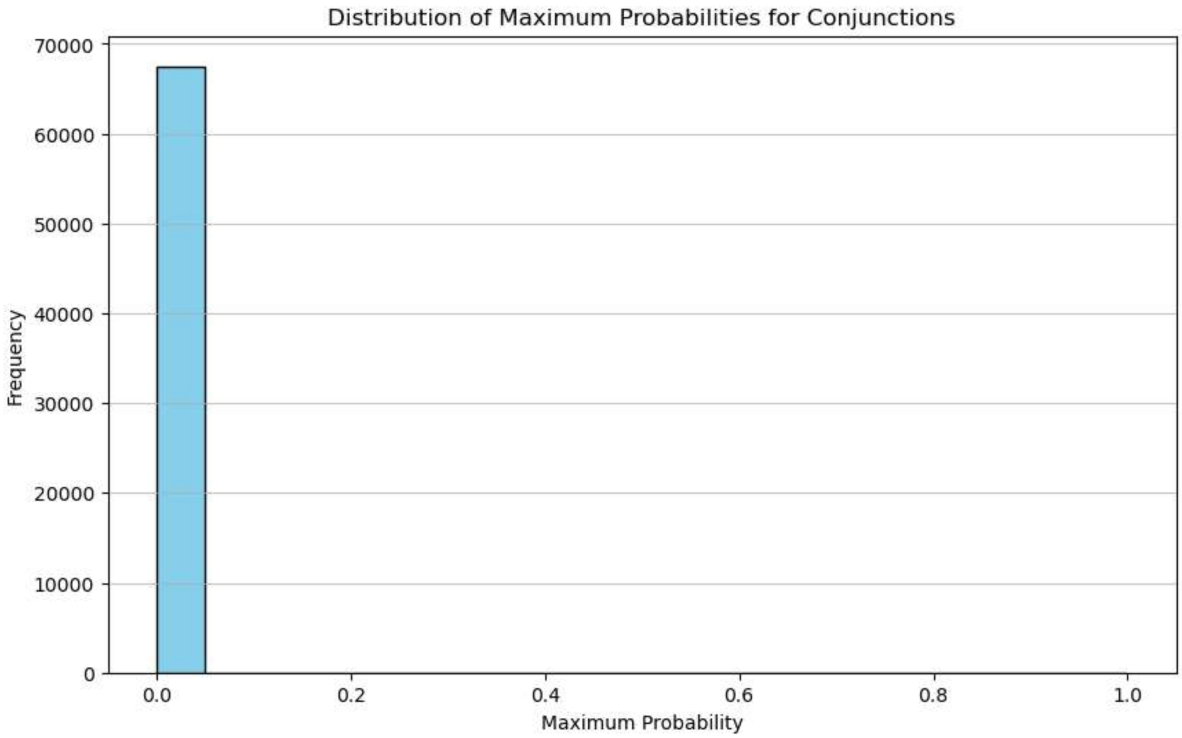
In [ ]:
```
##TCA Range Analysis: Analyze the TCA range values to understand how closely th
```

In [18]:
```
df = pd.DataFrame(data)

# Convert 'TCA' column to datetime format
df['TCA'] = pd.to_datetime(df['TCA'], format='%M:%S.%f')

# Plotting TCA Range analysis
plt.figure(figsize=(10, 6))
plt.plot(df['TCA'], df['TCA_RANGE'], marker='o', linestyle='-', color='b')
plt.xlabel('TCA Time')
plt.ylabel('TCA Range')
plt.title('TCA Range Analysis Over Time')
plt.grid(True)
plt.show()
```



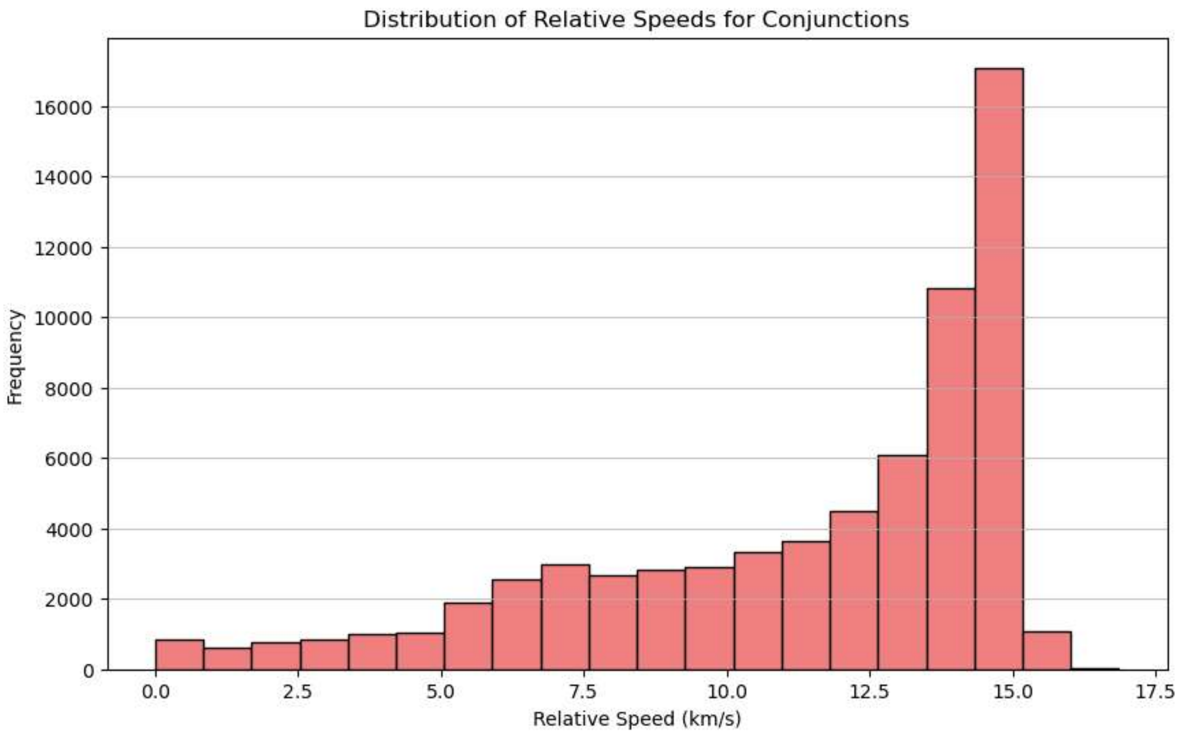
In [ ]:
```
#Maximum Probability Analysis: Examine the 'MAX_PROB' column to assess the maxi
```

In [19]:
```
df = pd.DataFrame(data)

# Plot histogram for Maximum Probabilities
plt.figure(figsize=(10, 6))
plt.hist(df['MAX_PROB'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Maximum Probabilities for Conjunctions')
plt.xlabel('Maximum Probability')
plt.ylabel('Frequency')
plt.grid(axis='y', alpha=0.75)
plt.show()
```



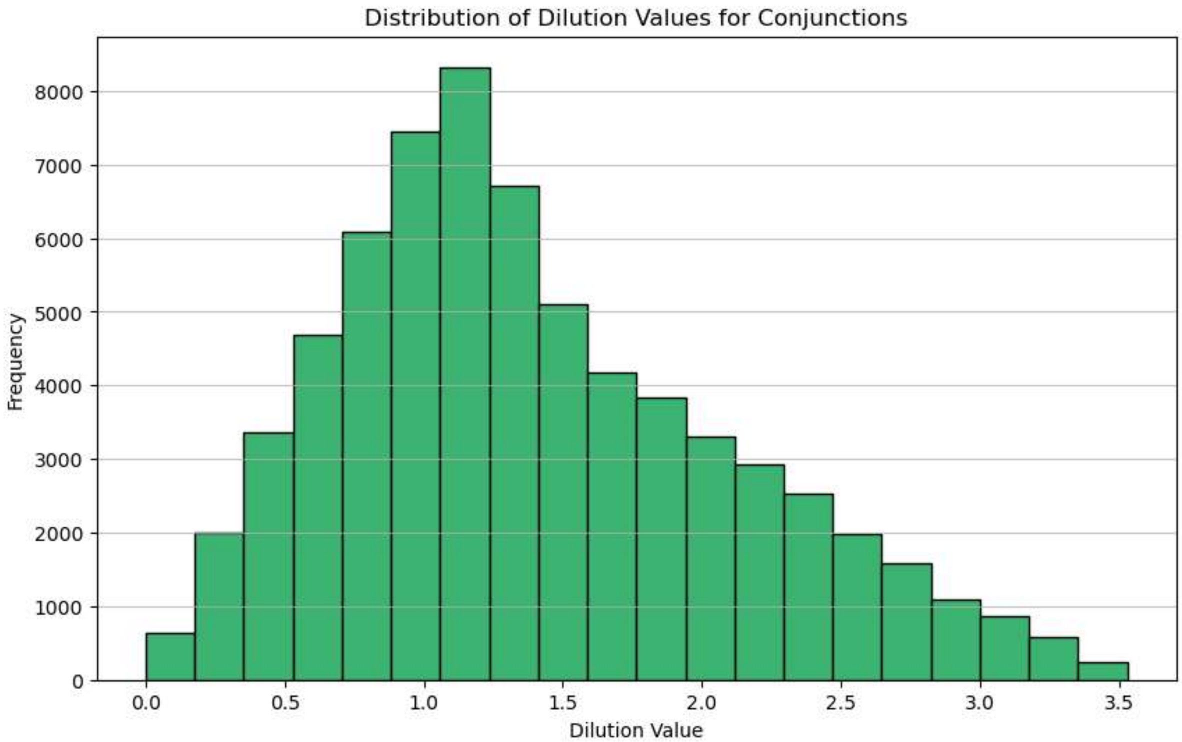
In [ ]:
```
##Relative Speed Analysis: Investigate the 'TCA_RELATIVE_SPEED' column to under
```

In [20]:
```
df = pd.DataFrame(data)

# Plot histogram for Relative Speeds
plt.figure(figsize=(10, 6))
plt.hist(df['TCA_RELATIVE_SPEED'], bins=20, color='lightcoral', edgecolor='blac
plt.title('Distribution of Relative Speeds for Conjunctions')
plt.xlabel('Relative Speed (km/s)')
plt.ylabel('Frequency')
plt.grid(axis='y', alpha=0.75)
plt.show()
```



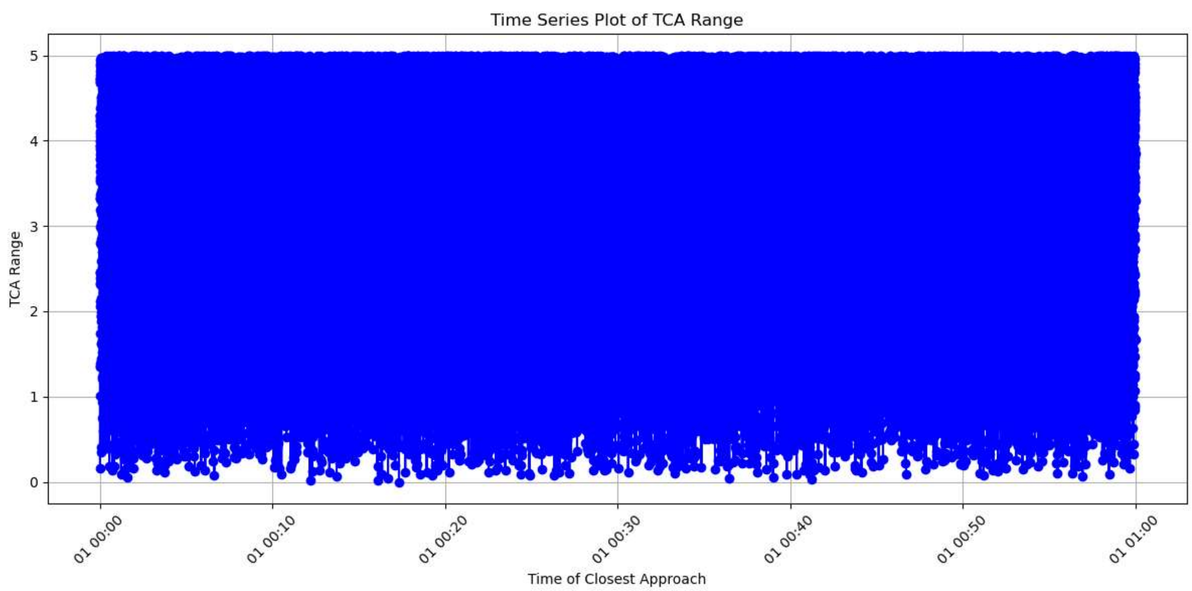
In [ ]:
```
##Dilution Analysis: Examine the 'DILUTION' column to understand the dilution j
```

In [21]:
```
df = pd.DataFrame(data)

# Plot histogram for Dilution values
plt.figure(figsize=(10, 6))
plt.hist(df['DILUTION'], bins=20, color='mediumseagreen', edgecolor='black')
plt.title('Distribution of Dilution Values for Conjunctions')
plt.xlabel('Dilution Value')
plt.ylabel('Frequency')
plt.grid(axis='y', alpha=0.75)
plt.show()
```

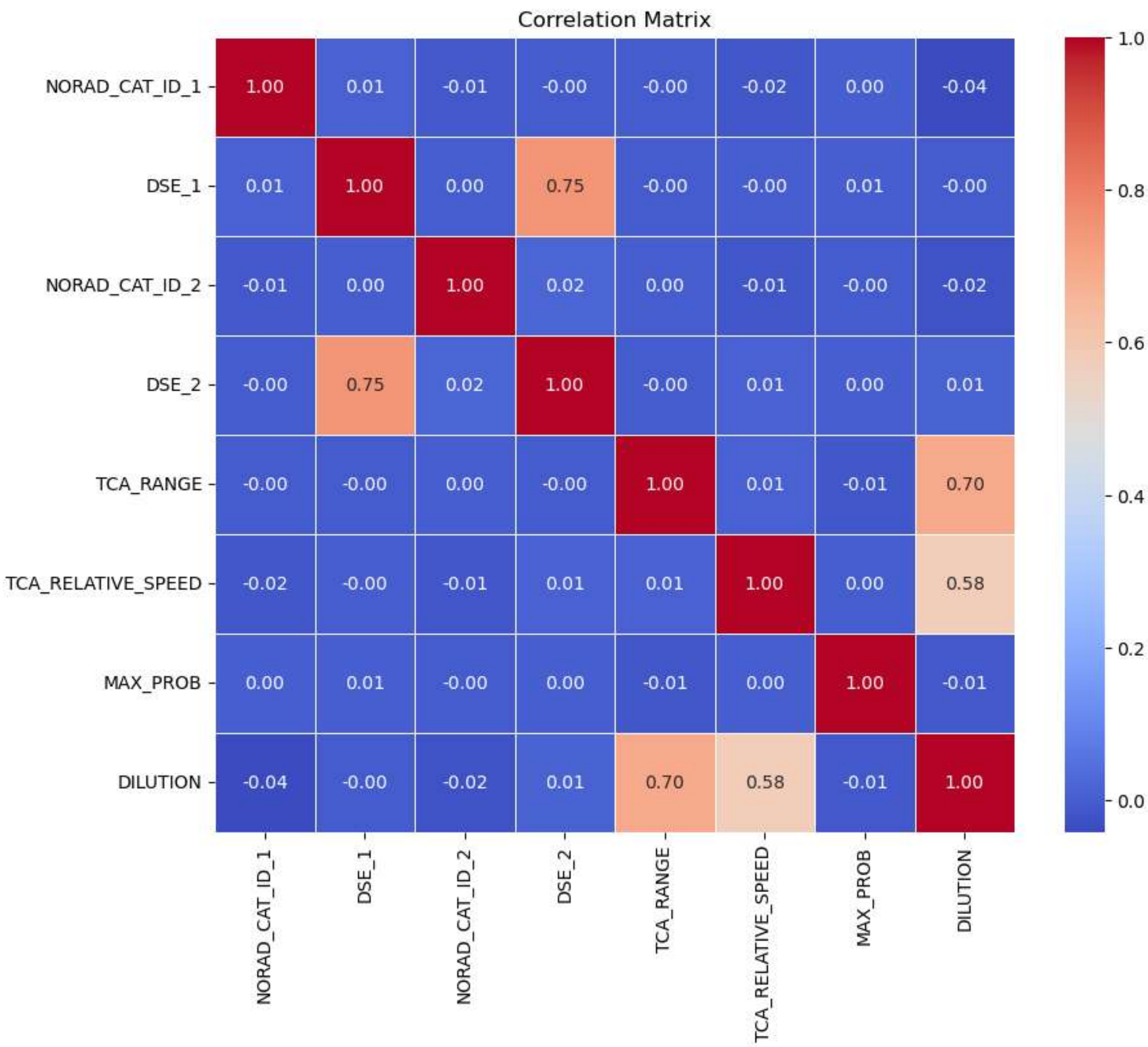


In [ ]:
```
##Time Series Plot: Visualize the occurrence of conjunctions over time using a
```

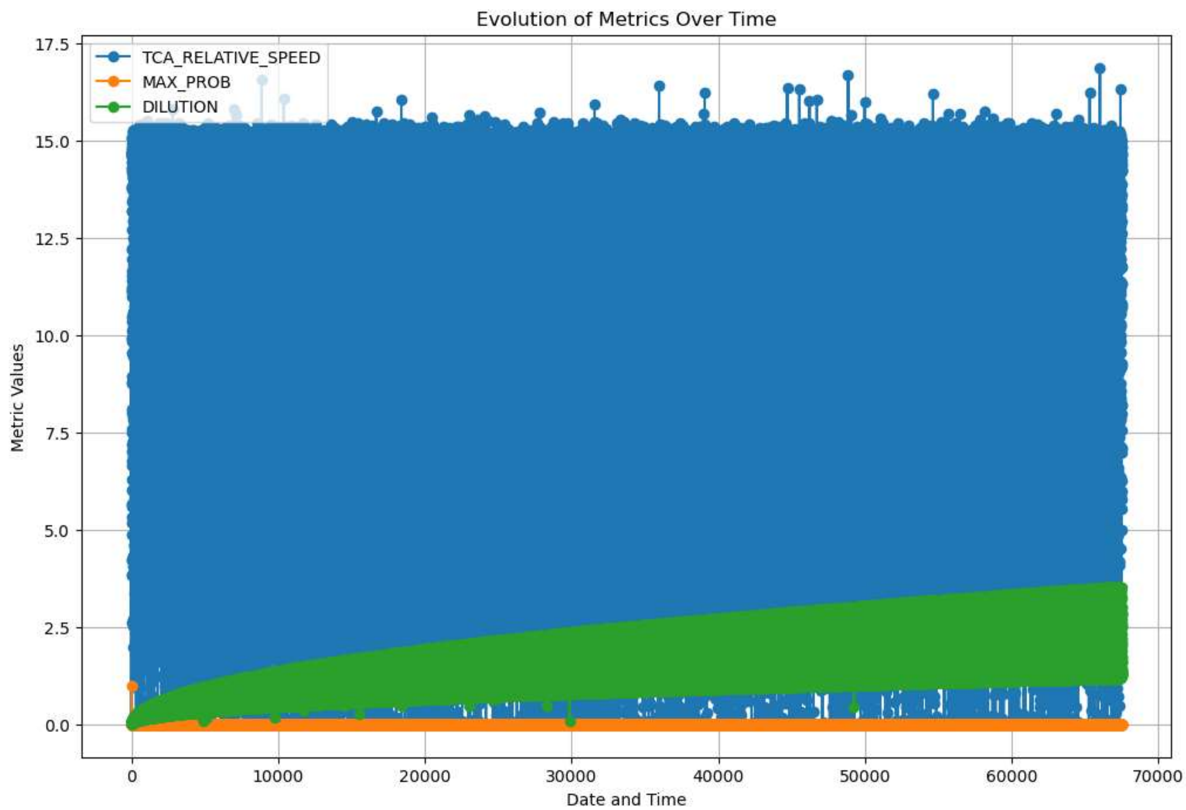
In [25]:
```
df = pd.DataFrame(data)

# Convert 'TCA' column to datetime format
df['TCA'] = pd.to_datetime(df['TCA'], format='%M:%S.%f')

# Sort DataFrame by 'TCA' for correct time series plotting
df = df.sort_values(by='TCA')

# Plotting
plt.figure(figsize=(12, 6))
plt.plot(df['TCA'], df['TCA_RANGE'], marker='o', linestyle='-', color='b')
plt.title('Time Series Plot of TCA Range')
plt.xlabel('Time of Closest Approach')
plt.ylabel('TCA Range')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



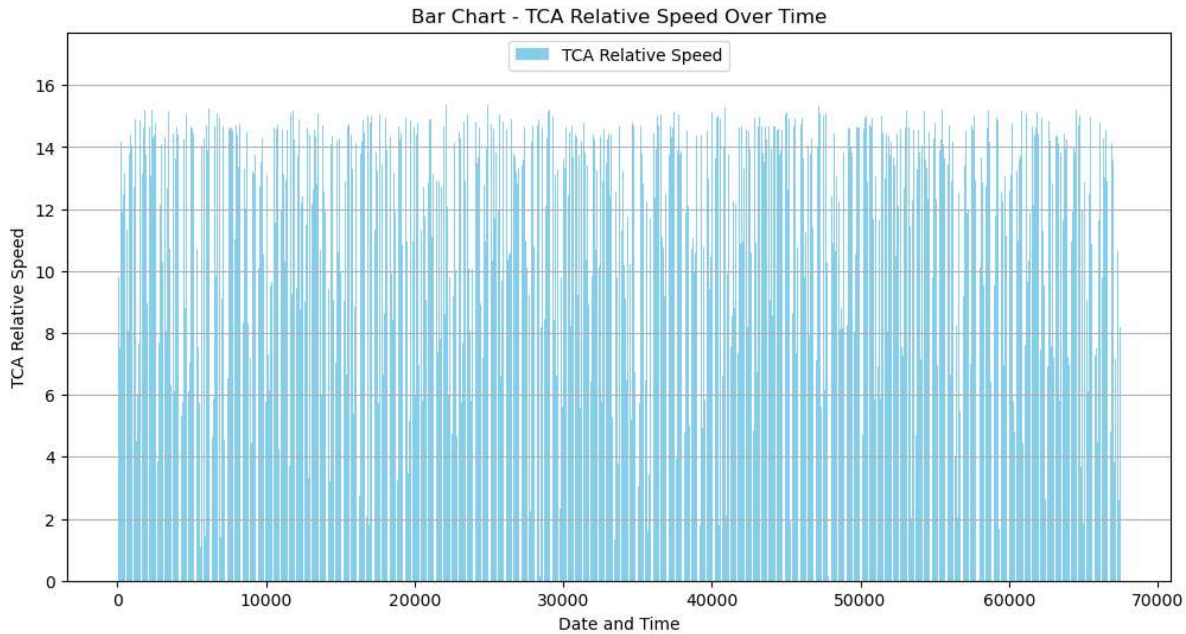
In [26]:
```python
df = pd.DataFrame(data)

# Select only numeric columns
numeric_columns = df.select_dtypes(include=['float64', 'int64'])

# Compute the correlation matrix
correlation_matrix = numeric_columns.corr()

# Create a heatmap using seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewi
plt.title('Correlation Matrix')
plt.show()
```



In [27]:
```
metrics = ['TCA_RELATIVE_SPEED', 'MAX_PROB', 'DILUTION']

plt.figure(figsize=(12, 8))
for metric in metrics:
    plt.plot(df[metric], label=metric, marker='o', linestyle='-')

plt.xlabel('Date and Time')
plt.ylabel('Metric Values')
plt.title('Evolution of Metrics Over Time')
plt.legend()
plt.grid(True)
plt.show()
```



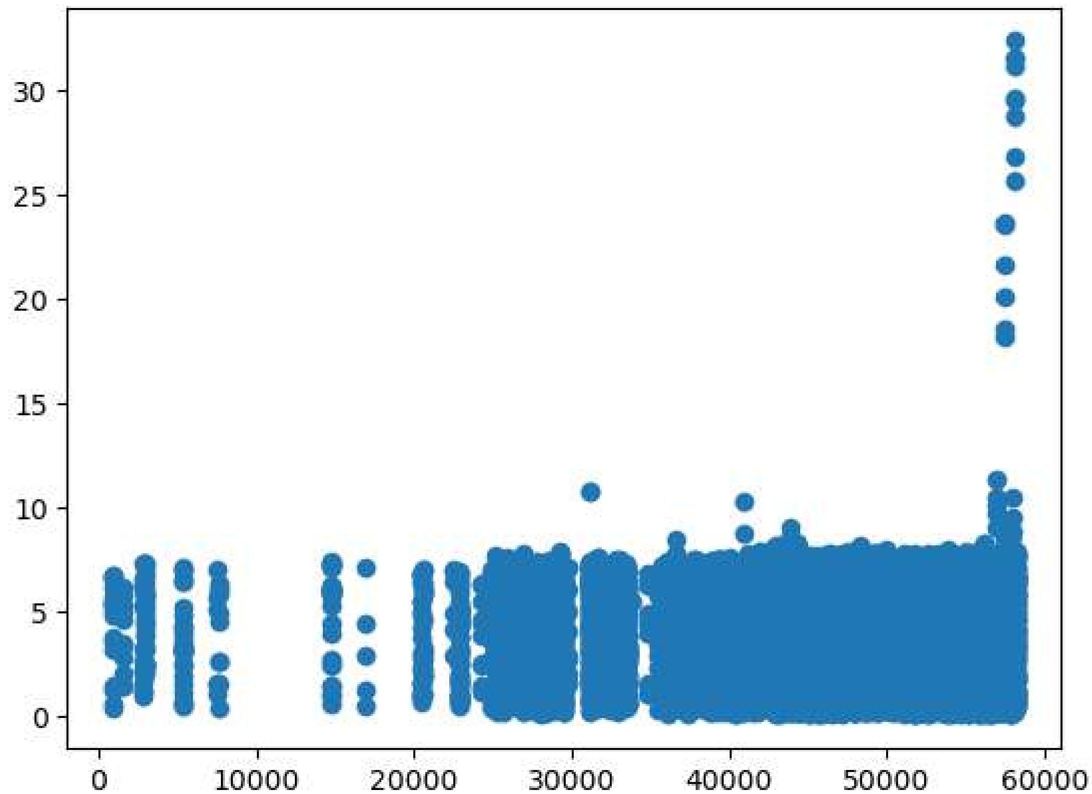
In [28]:
```
plt.figure(figsize=(12, 6))
plt.bar(df.index, df['TCA_RELATIVE_SPEED'], color='skyblue', label='TCA Relati

plt.xlabel('Date and Time')
plt.ylabel('TCA Relative Speed')
plt.title('Bar Chart - TCA Relative Speed Over Time')
plt.legend()
plt.grid(axis='y')
plt.show()
```



In [ ]:
```
#Scatter plots can be useful for visualizing relationships between two variable
```

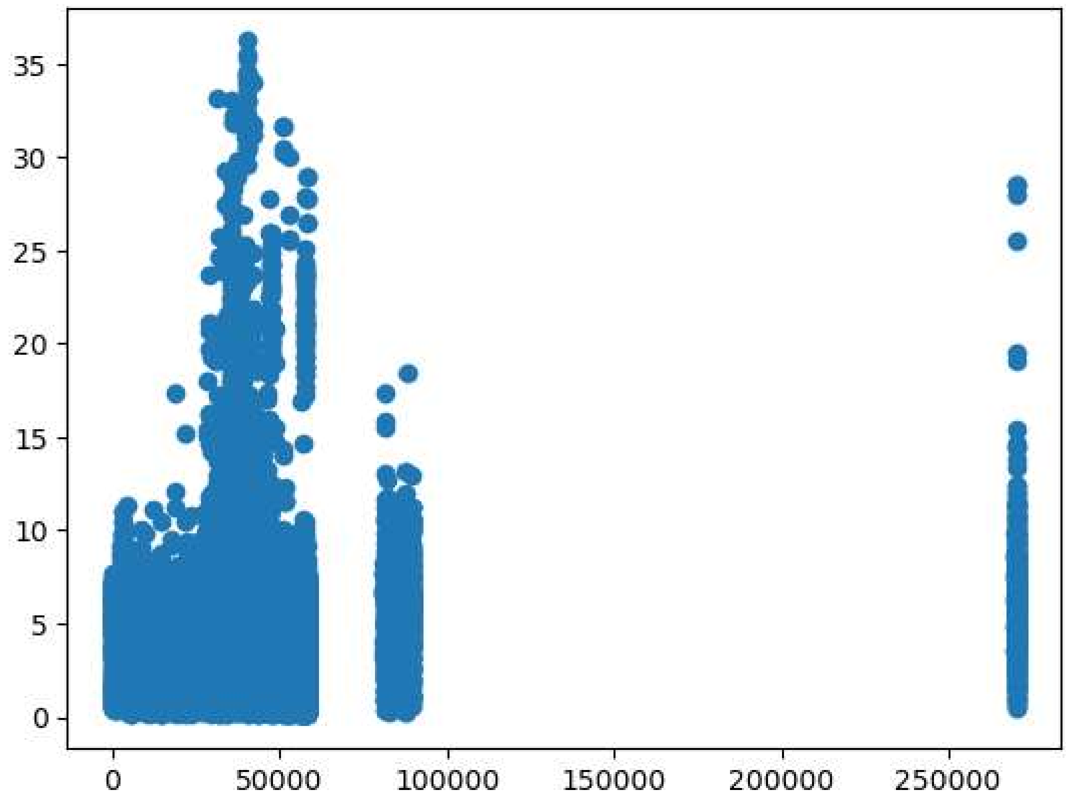
In [4]:
```
plt.scatter(data['NORAD_CAT_ID_1'],data['DSE_1'])
```

Out[4]:
```
<matplotlib.collections.PathCollection at 0x1ce18667a50>
```



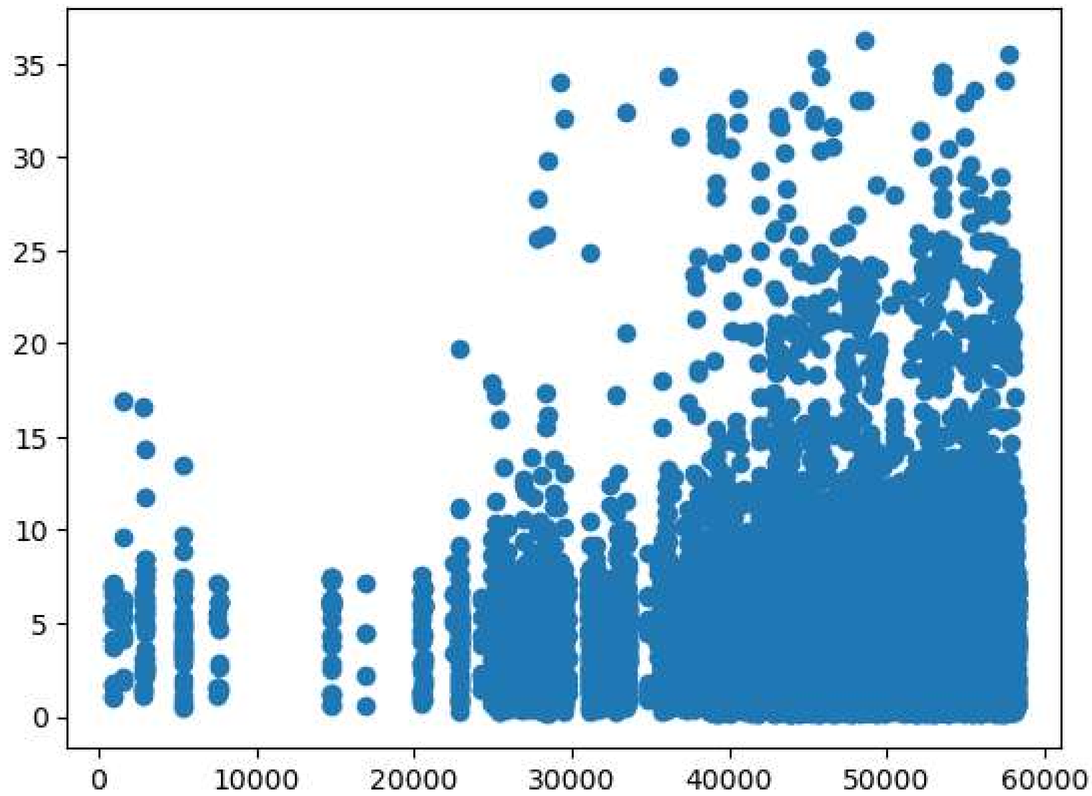
In [5]:
```
plt.scatter(data['NORAD_CAT_ID_2'],data['DSE_2'])
```

Out[5]: <matplotlib.collections.PathCollection at 0x1ce19034090>



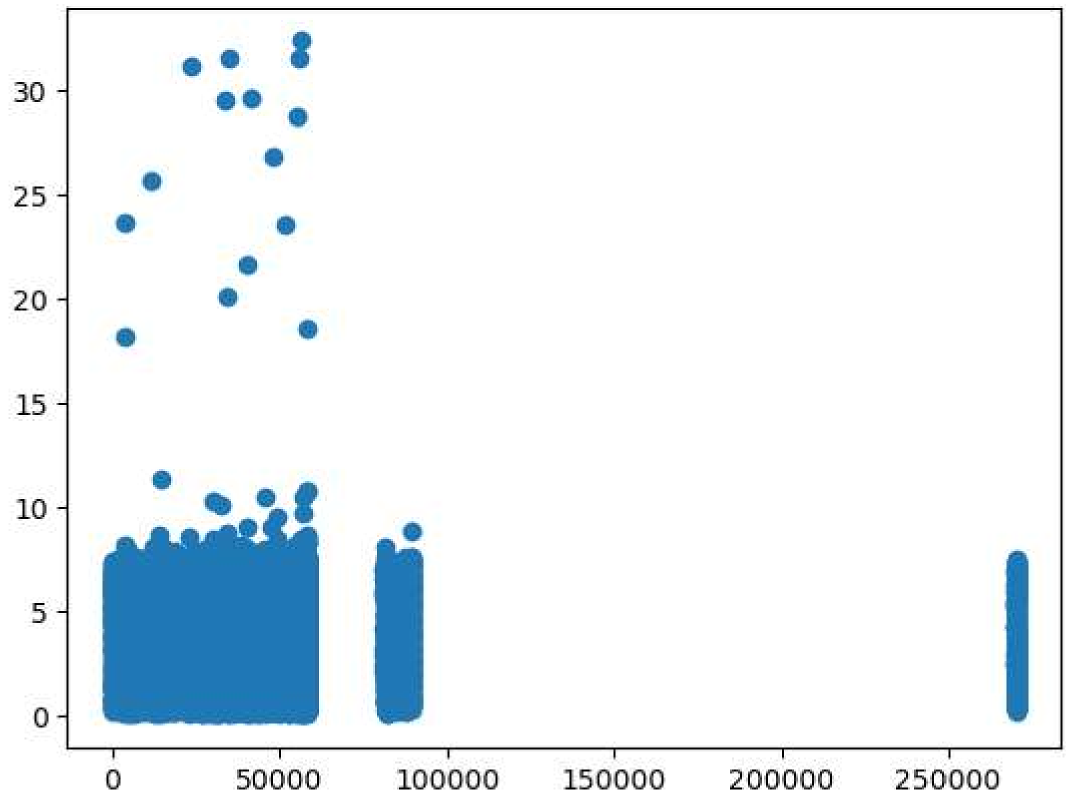
In [6]:
```
plt.scatter(data['NORAD_CAT_ID_1'],data['DSE_2'])
```

Out[6]:
```
<matplotlib.collections.PathCollection at 0x1ce190a76d0>
```



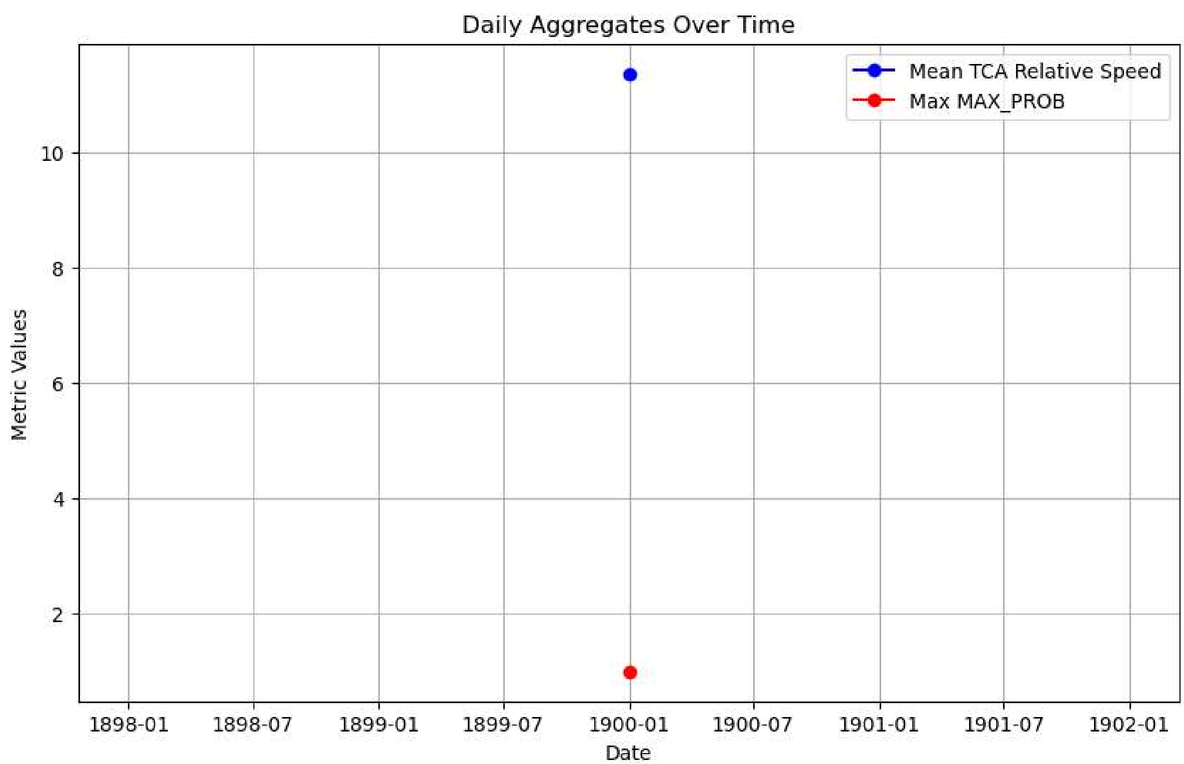
In [7]:
```
plt.scatter(data['NORAD_CAT_ID_2'],data['DSE_1'])
```

Out[7]: <matplotlib.collections.PathCollection at 0x1ce19130550>



In [ ]:

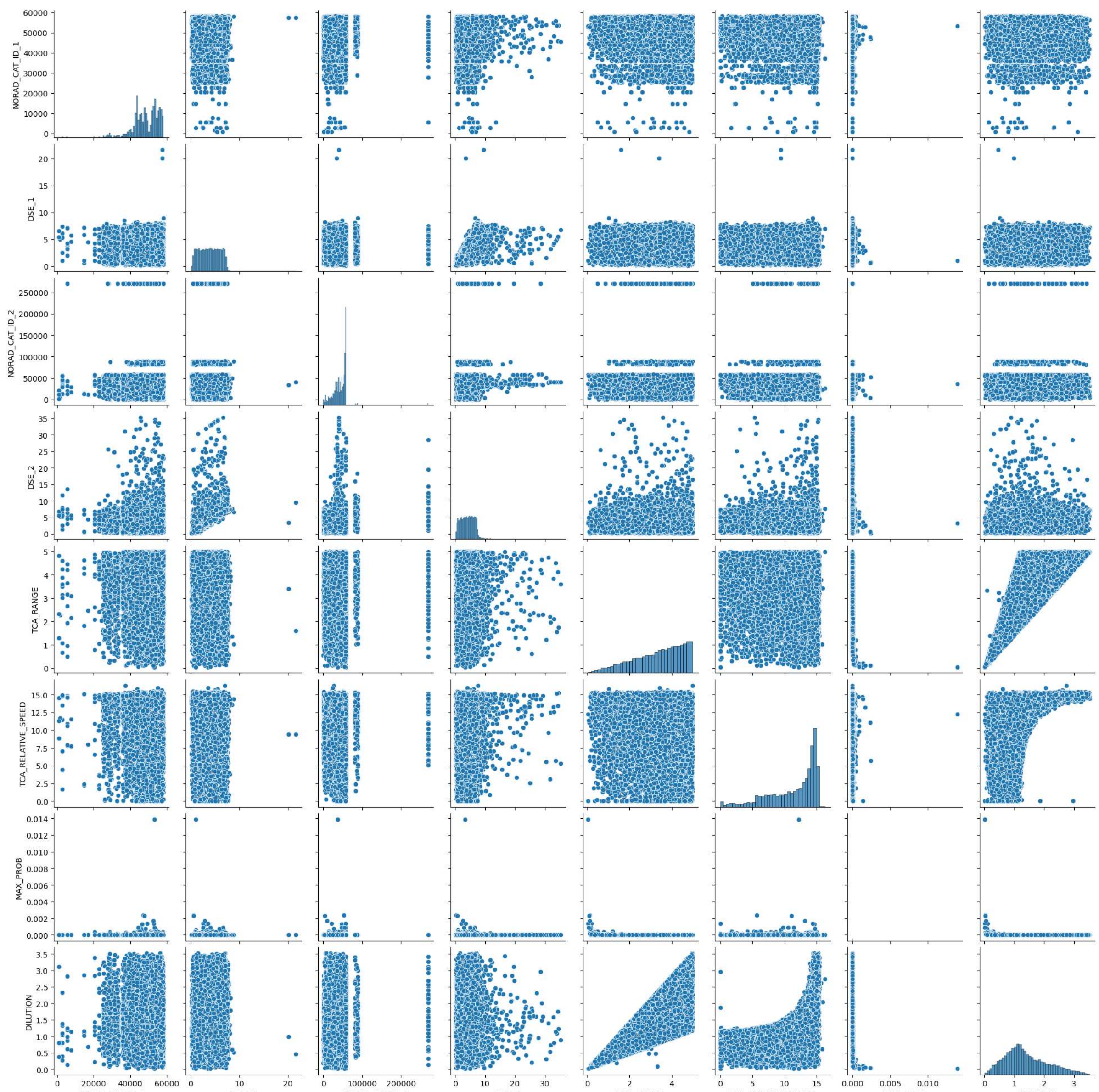
In [31]:
```python
df['TCA'] = pd.to_datetime(df['TCA'], format='%M:%S.%f')

# Create a new column for the date
df['Date'] = df['TCA'].dt.date

# Calculate daily aggregates
daily_aggregates = df.groupby('Date').agg({
    'TCA_RELATIVE_SPEED': 'mean',  # Adjust this based on the metric you want t
    'MAX_PROB': 'max',              # Another example metric
    # Add more metrics as needed
}).reset_index()

# Plotting daily aggregates
plt.figure(figsize=(10, 6))
plt.plot(daily_aggregates['Date'], daily_aggregates['TCA_RELATIVE_SPEED'], mark
plt.plot(daily_aggregates['Date'], daily_aggregates['MAX_PROB'], marker='o', l:
# Add more plots for additional metrics

plt.xlabel('Date')
plt.ylabel('Metric Values')
plt.title('Daily Aggregates Over Time')
plt.legend()
plt.grid(True)
plt.show()
```



In [13]:
```
import seaborn as sns
#df = pd.read_csv("output.csv ")

# Sample a fraction of your data (e.g., 20%)
df_sampled = data.sample(frac=0.2, random_state=42)

# Create pair plot with the sampled data
sns.pairplot(df_sampled)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[13]: <seaborn.axisgrid.PairGrid at 0x1ce1906b010>



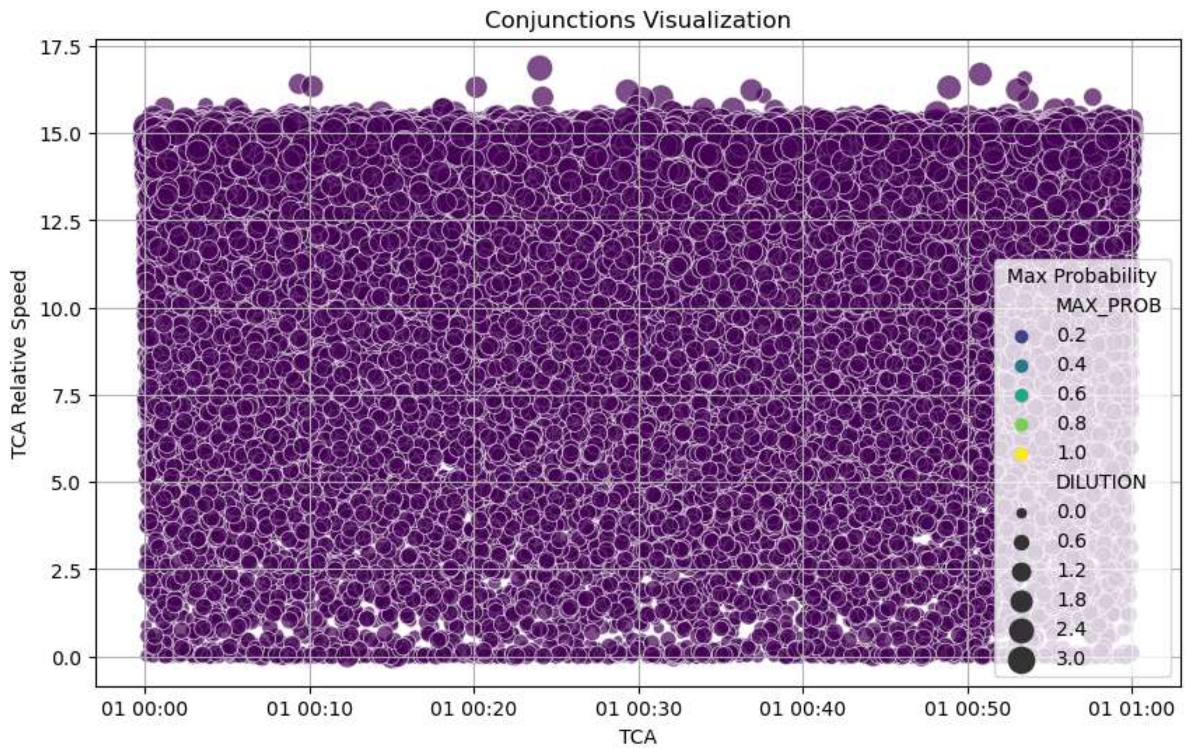
number of conjunction among active satellites.

In [16]:
```
df = pd.DataFrame(data)

# Convert 'TCA' column to datetime format
df['TCA'] = pd.to_datetime(df['TCA'], format='%M:%S.%f', errors='coerce')

# Extract time component
df['TCA_time'] = df['TCA'].dt.time

# Specify the comparison time (e.g., midnight)
comparison_time = pd.to_datetime('00:00:00').time()

# Filter rows where both objects are active satellites and have a non-zero TCA
conjunctions = df[(df['DSE_1'] > 0) & (df['DSE_2'] > 0) & (df['TCA_time'] > cor

# Print the number of conjunctions
num_conjunctions = len(conjunctions)
print("Number of conjunctions among active satellites:", num_conjunctions)
```

```
Number of conjunctions among active satellites: 67505
```

In [ ]:
```
#Result in visual form
```

In [17]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'conjunctions' is the DataFrame containing the conjunction data

# Scatter plot of TCA values
plt.figure(figsize=(10, 6))
sns.scatterplot(x='TCA', y='TCA_RELATIVE_SPEED', data=conjunctions, hue='MAX_PI
plt.title('Conjunctions Visualization')
plt.xlabel('TCA')
plt.ylabel('TCA Relative Speed')
plt.legend(title='Max Probability')
plt.grid(True)
plt.show()
```



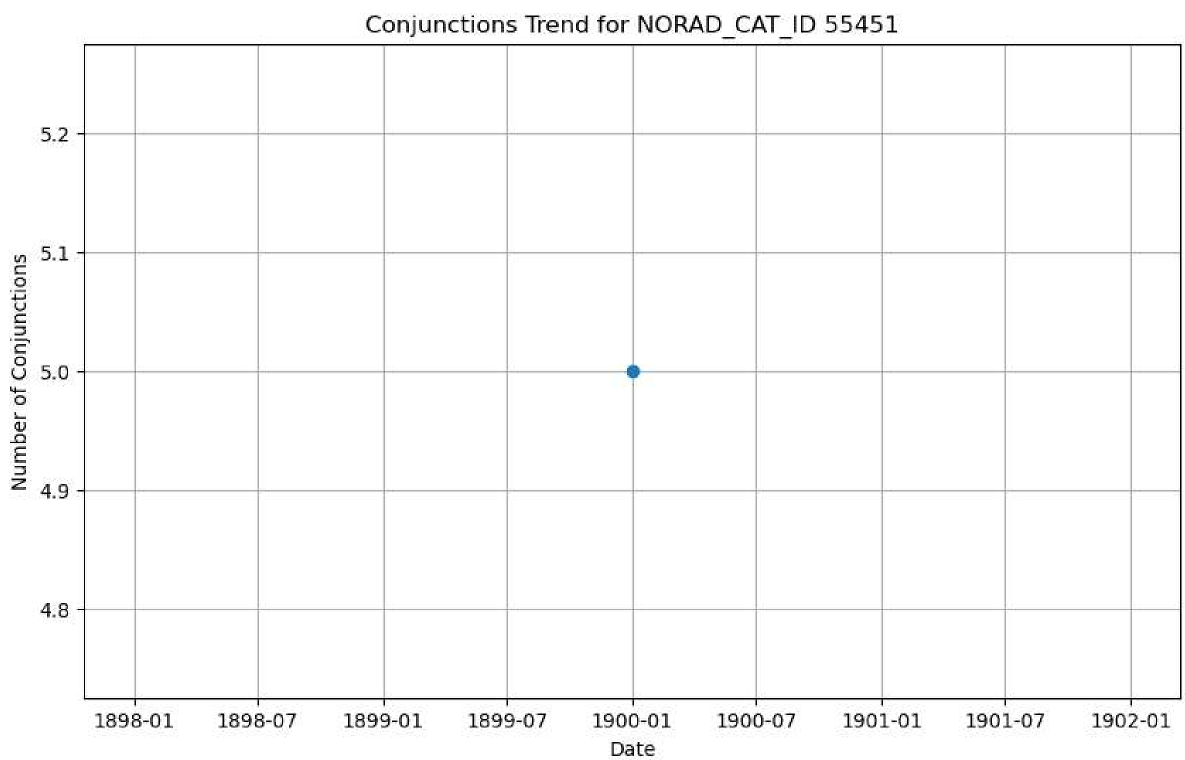
In [ ]:
```
#Represent the conjunctions data of a single satellite or a satellite constell
#The analytics should be intuitive, represented in an easily understandable for
#should enable decision making from a satellite operator’s point of view.
```

In [9]:
```
df = pd.DataFrame(data)
selected_satellite_id = 55451

# Filter data for the selected satellite
selected_satellite_data = df[df['NORAD_CAT_ID_1'] == selected_satellite_id]

# Display the selected satellite's conjunction data
print("Conjunctions Data for NORAD_CAT_ID:", selected_satellite_id)
print(selected_satellite_data)
```

```
Conjunctions Data for NORAD_CAT_ID: 55451
       NORAD_CAT_ID_1      OBJECT_NAME_1  DSE_1  NORAD_CAT_ID_2  \
0               55451  STARLINK-5656 [+]  6.780           23608
12507           55451  STARLINK-5656 [+]  0.699            4724
32290           55451  STARLINK-5656 [+]  6.767           54430
45738           55451  STARLINK-5656 [+]  4.040           57183
56689           55451  STARLINK-5656 [+]  0.813           57191

                         OBJECT_NAME_2  DSE_2      TCA  TCA_RANGE  \
0                    ARIANE 40+3 R/B [-]  6.545  17:20.4      0.000
12507                 COSMOS 375 DEB [-]  0.497  20:47.3      2.156
32290                      CZ-6A DEB [-]  6.676  58:21.1      3.457
45738                       OBJECT T [+]  3.830  31:36.2      4.109
56689  POLYTECH-UNIVERSE 3 (RS46S) [+]  0.606  04:43.3      4.581

       TCA_RELATIVE_SPEED  MAX_PROB  DILUTION
0                  14.299  1.000000     0.000
12507              12.215  0.000006     0.770
32290              12.743  0.000002     1.331
45738              12.104  0.000002     1.472
56689              10.890  0.000001     1.467
```

In [10]:
```
#Use the whole dataset that spans about five days. Derive analytics and visuali
#data/analytics accounting for the evolution from the first day (for e.g. the r
#conjunctions of the RSO having NORAD ID 12345 over 7 days of analysis)
```

In [11]:
```
#df = pd.DataFrame(data)

# Convert 'TCA' column to datetime format
df['TCA'] = pd.to_datetime(df['TCA'], format='%M:%S.%f')

# Extract date from 'TCA' column for grouping
df['Date'] = df['TCA'].dt.date

# Choose a specific NORAD_CAT_ID for analysis (e.g., 55451)
selected_satellite_id = 55451

# Filter data for the selected satellite
selected_satellite_data = df[df['NORAD_CAT_ID_1'] == selected_satellite_id]

# Calculate the number of conjunctions per day
conjunctions_per_day = selected_satellite_data.groupby('Date').size()

# Plotting the trend
plt.figure(figsize=(10, 6))
plt.plot(conjunctions_per_day.index, conjunctions_per_day.values, marker='o', l
plt.title(f'Conjunctions Trend for NORAD_CAT_ID {selected_satellite_id}')
plt.xlabel('Date')
plt.ylabel('Number of Conjunctions')
plt.grid(True)
plt.show()
```



In [ ]:

In [ ]:

In [ ]: