DataScienceProjectPart1 (1)

February 3, 2026

[1]:
```
## Step 1: Load the data using pandas and print the dataframe
import pandas as pd
df = pd.read_csv('project_data.csv')
df
```

[1]:

	Unnamed: 0	Tweet	Followers	Friends	Num_tweets	\
0	0	1441497018807906305	1198	605.0	6166	
1	1	1431812786099613699	608	1259.0	1811	
2	2	1426644707313135617	173	167.0	4306	
3	3	1431814908425998337	2540	222.0	6008	
4	4	1432862687533441027	4439	11.0	9985	
…	…	…	…	…	…	
28818	28815	1431029315274162181	16182	1150.0	58245	
28819	28816	1437970083087851520	688077	636.0	12308	
28820	28817	1428110541093052418	57068	1225.0	25131	
28821	28818	1430722665514377219	66	121.0	549	
28822	28819	1441165693974503442	169	276.0	2910	

	Verified	Listed_count	Location	Age	Length	Num_users	\
0	False	1	True	7	6	4	
1	False	5	True	2	2	2	
2	False	0	True	0	4	2	
3	False	0	True	9	2	2	
4	False	55	True	2	4	3	
…	…	…	…	…	…	…	
28818	True	685	False	12	2	2	
28819	True	361	False	10	247	218	
28820	False	0	True	5	22	21	
28821	False	3	True	1	4	3	
28822	False	7	True	0	2	2	

	Num_author_replies	TOXICITY_x	Num_toxic_direct_replies	\
0	2	0.235235	0.0	
1	0	0.085582	0.0	
2	2	0.076877	0.0	
3	0	0.095684	0.0	
4	1	0.165919	0.0	

```
…                    …          …                          …
28818                    0    0.042905                        0.0
28819                    3    0.039698                        8.0
28820                    0    0.018346                        2.0
28821                    1    0.924899                        0.0
28822                    0    0.035456                        0.0

       Num_toxic_nested_replies  Num_author_toxic_replies  Num_toxic_replies  \
0                           0.0                       0.0                0.0
1                           0.0                       0.0                0.0
2                           0.0                       0.0                0.0
3                           0.0                       0.0                0.0
4                           0.0                       0.0                0.0
…                           …                         …                  …
28818                       0.0                       0.0                0.0
28819                       2.0                       0.0               10.0
28820                       0.0                       0.0                2.0
28821                       0.0                       0.0                0.0
28822                       0.0                       0.0                0.0

       Toxic
0          0
1          0
2          0
3          0
4          0
…      …
28818      0
28819      1
28820      1
28821      0
28822      0

[28823 rows x 18 columns]
```

[2]:
```
## Step 2: Print the column names and dataframe shape
print(list(df))
print("Dataframe Shape:", df.shape)
```

```
['Unnamed: 0', 'Tweet', 'Followers', 'Friends', 'Num_tweets', 'Verified',
'Listed_count', 'Location', 'Age', 'Length', 'Num_users', 'Num_author_replies',
'TOXICITY_x', 'Num_toxic_direct_replies', 'Num_toxic_nested_replies',
'Num_author_toxic_replies', 'Num_toxic_replies', 'Toxic']
Dataframe Shape: (28823, 18)
```

Step 2 Interpretation: How many observations and columns are there in the dataset? There are 28823 different rows or tweets, and 18 different labeled columns providing information on each tweet.

[3]:
```
## Step 3: Inspect the dataframe and print out the data types for all the␣
  ↪columns
print(df.dtypes)
```

```
Unnamed: 0                  int64
Tweet                       int64
Followers                   int64
Friends                   float64
Num_tweets                  int64
Verified                     bool
Listed_count                int64
Location                   object
Age                         int64
Length                      int64
Num_users                   int64
Num_author_replies          int64
TOXICITY_x                float64
Num_toxic_direct_replies  float64
Num_toxic_nested_replies  float64
Num_author_toxic_replies  float64
Num_toxic_replies         float64
Toxic                       int64
dtype: object
```

Interpretation for step 3: What are your observations about the dataset and variable data types (which variables are numeric, which ones are categorical, which ones are binary, etc.)? Things that should be stored as whole numbers or integers, such as Tweet id, followers, num_tweets, are stored as an int64 in this dataframe. Columns that need representaion as a number with decimals are stored as float64, such as the toxicity rating of tweets. It seems to me that some of the columns that are represented as a float64 should actually be stored as an int64 instead, because the categories like Friends and the number of replies will always be an integer.The only boolean data type in this dataframe is whether the user is verified or not, which makes sense to use a true/false data type here. Location is stored as an object which can be used to represent a string or word, but it does say in about the data set that the location is supposed to be a binary variable indicating whether the user who posted the original tweet provided the location on their profiles or not so this may need to be changed.

[4]:
```
## Step 4, Remove the rows containing missing values
df = df.dropna(axis=0) ## removes rows containing NaN values
df.shape
```

[4]: (28821, 18)

Step 4 interpretation: How many rows contain missing values? In other words, how many rows did you remove? After removing rows containing NaN values and looking at the shape of the dataframe again, I see that there are two less rows than we started with, which means I removed two rows from the orginal dataframe.

[5]:
```
## Step 5, remove dupliacte rows (just keep one)
df = df.drop_duplicates()
df.shape
```

[5]:
```
(28818, 18)
```

Step 5 interpretation: How many duplicated rows have you removed? After checking the shape of the data frame again after removing any duplicate rows and keeping one of each, the new data frame has 3 less rows which means 3 duplicated rows were removed.

[6]:
```
## Step 6: Transform the Toxicity column. Make it binary if the value is equal␣
 ↪or higher than 0.5, make the value 1, if it is lower than 0.5, make it 0
## TOXICITY_x is read as a boolean statement based on wehether the float64␣
 ↪rating of toxicty was >= 0.5, true or false were then transformed to 1 and 0␣
 ↪respectively using .astype command.
df['TOXICITY_x'] = (df['TOXICITY_x']>= 0.5).astype(int)
print(df.dtypes)
```

```
Unnamed: 0                  int64
Tweet                       int64
Followers                   int64
Friends                   float64
Num_tweets                  int64
Verified                     bool
Listed_count                int64
Location                   object
Age                         int64
Length                      int64
Num_users                   int64
Num_author_replies          int64
TOXICITY_x                  int32
Num_toxic_direct_replies  float64
Num_toxic_nested_replies  float64
Num_author_toxic_replies  float64
Num_toxic_replies         float64
Toxic                       int64
dtype: object
```

[6]:

	Unnamed: 0	Tweet	Followers	Friends	Num_tweets	\
0	0	1441497018807906305	1198	605.0	6166	
1	1	1431812786099613699	608	1259.0	1811	
2	2	1426644707313135617	173	167.0	4306	
3	3	1431814908425998337	2540	222.0	6008	
4	4	1432862687533441027	4439	11.0	9985	
…	…	…	…	…	…	
28818	28815	1431029315274162181	16182	1150.0	58245	
28819	28816	1437970083087851520	688077	636.0	12308	

```
28820       28817  1428110541093052418      57068   1225.0       25131
28821       28818  1430722665514377219         66    121.0         549
28822       28819  1441165693974503442        169    276.0        2910

       Verified  Listed_count Location  Age  Length  Num_users  \
0         False             1     True    7       6          4
1         False             5     True    2       2          2
2         False             0     True    0       4          2
3         False             0     True    9       2          2
4         False            55     True    2       4          3
…         …           …       …  …     …         …
28818      True           685    False   12       2          2
28819      True           361    False   10     247        218
28820     False             0     True    5      22         21
28821     False             3     True    1       4          3
28822     False             7     True    0       2          2

       Num_author_replies  TOXICITY_x  Num_toxic_direct_replies  \
0                       2           0                       0.0
1                       0           0                       0.0
2                       2           0                       0.0
3                       0           0                       0.0
4                       1           0                       0.0
…                   …         …                       …
28818                   0           0                       0.0
28819                   3           0                       8.0
28820                   0           0                       2.0
28821                   1           1                       0.0
28822                   0           0                       0.0

       Num_toxic_nested_replies  Num_author_toxic_replies  Num_toxic_replies  \
0                           0.0                       0.0                0.0
1                           0.0                       0.0                0.0
2                           0.0                       0.0                0.0
3                           0.0                       0.0                0.0
4                           0.0                       0.0                0.0
…                         …                       …                …
28818                       0.0                       0.0                0.0
28819                       2.0                       0.0               10.0
28820                       0.0                       0.0                2.0
28821                       0.0                       0.0                0.0
28822                       0.0                       0.0                0.0

       Toxic
0          0
1          0
2          0
```

```
3          0
4          0
…        …
28818      0
28819      1
28820      1
28821      0
28822      0

[28818 rows x 18 columns]
```

[7]:
```
## Step 7: Create a new column named Total_toxic_replies which would be the sum␣
 ↪of Num_toxic_direct_replies and Num_toxic_nested_replies.
df['Total_toxic_replies'] = df['Num_toxic_direct_replies'] +␣
 ↪df['Num_toxic_nested_replies']
df
```

[7]:
```
       Unnamed: 0                Tweet  Followers  Friends  Num_tweets  \
0               0  1441497018807906305       1198    605.0        6166
1               1  1431812786099613699        608   1259.0        1811
2               2  1426644707313135617        173    167.0        4306
3               3  1431814908425998337       2540    222.0        6008
4               4  1432862687533441027       4439     11.0        9985
…             …                    …        …       …          …
28818       28815  1431029315274162181      16182   1150.0       58245
28819       28816  1437970083087851520     688077    636.0       12308
28820       28817  1428110541093052418      57068   1225.0       25131
28821       28818  1430722665514377219         66    121.0         549
28822       28819  1441165693974503442        169    276.0        2910

       Verified  Listed_count Location  Age  Length  Num_users  \
0         False             1     True    7       6          4
1         False             5     True    2       2          2
2         False             0     True    0       4          2
3         False             0     True    9       2          2
4         False            55     True    2       4          3
…         …              …       …  …     …         …
28818      True           685    False   12       2          2
28819      True           361    False   10     247        218
28820     False             0     True    5      22         21
28821     False             3     True    1       4          3
28822     False             7     True    0       2          2

       Num_author_replies  TOXICITY_x  Num_toxic_direct_replies  \
0                       2           0                       0.0
1                       0           0                       0.0
2                       2           0                       0.0
```

```
3                      0           0                      0.0
4                      1           0                      0.0
…                    …           …                      …
28818                  0           0                      0.0
28819                  3           0                      8.0
28820                  0           0                      2.0
28821                  1           1                      0.0
28822                  0           0                      0.0

       Num_toxic_nested_replies  Num_author_toxic_replies  Num_toxic_replies  \
0                           0.0                       0.0                0.0
1                           0.0                       0.0                0.0
2                           0.0                       0.0                0.0
3                           0.0                       0.0                0.0
4                           0.0                       0.0                0.0
…                         …                         …                  …
28818                       0.0                       0.0                0.0
28819                       2.0                       0.0               10.0
28820                       0.0                       0.0                2.0
28821                       0.0                       0.0                0.0
28822                       0.0                       0.0                0.0

       Toxic  Total_toxic_replies
0          0                 0.0
1          0                 0.0
2          0                 0.0
3          0                 0.0
4          0                 0.0
…      …                   …
28818      0                 0.0
28819      1                10.0
28820      1                 2.0
28821      0                 0.0
28822      0                 0.0

[28818 rows x 19 columns]
```

[8]:
```
## Step 8: Create a new column named Toxic_conversation where the value would be 1 if Total_toxic_replies is greater than 0,
## while the value would be 0 otherwise.
df['Toxic_conversation'] = (df['Total_toxic_replies'] > 0).astype(int)
df
```

[8]:
```
       Unnamed: 0                Tweet  Followers  Friends  Num_tweets  \
0               0  1441497018807906305       1198    605.0        6166
1               1  1431812786099613699        608   1259.0        1811
2               2  1426644707313135617        173    167.0        4306
```

```
3              3  1431814908425998337       2540    222.0        6008
4              4  1432862687533441027       4439     11.0        9985
…            …                   …         …       …          …
28818      28815  1431029315274162181      16182   1150.0       58245
28819      28816  1437970083087851520     688077    636.0       12308
28820      28817  1428110541093052418      57068   1225.0       25131
28821      28818  1430722665514377219         66    121.0         549
28822      28819  1441165693974503442        169    276.0        2910

       Verified  Listed_count Location  Age  Length  Num_users  \
0         False             1     True    7       6          4
1         False             5     True    2       2          2
2         False             0     True    0       4          2
3         False             0     True    9       2          2
4         False            55     True    2       4          3
…         …           …      …  …     …         …
28818      True           685    False   12       2          2
28819      True           361    False   10     247        218
28820     False             0     True    5      22         21
28821     False             3     True    1       4          3
28822     False             7     True    0       2          2

       Num_author_replies  TOXICITY_x  Num_toxic_direct_replies  \
0                       2           0                       0.0
1                       0           0                       0.0
2                       2           0                       0.0
3                       0           0                       0.0
4                       1           0                       0.0
…                   …         …                     …
28818                   0           0                       0.0
28819                   3           0                       8.0
28820                   0           0                       2.0
28821                   1           1                       0.0
28822                   0           0                       0.0

       Num_toxic_nested_replies  Num_author_toxic_replies  Num_toxic_replies  \
0                           0.0                       0.0                0.0
1                           0.0                       0.0                0.0
2                           0.0                       0.0                0.0
3                           0.0                       0.0                0.0
4                           0.0                       0.0                0.0
…                         …                         …                  …
28818                       0.0                       0.0                0.0
28819                       2.0                       0.0               10.0
28820                       0.0                       0.0                2.0
28821                       0.0                       0.0                0.0
28822                       0.0                       0.0                0.0
```

```
       Toxic  Total_toxic_replies  Toxic_conversation
0          0                  0.0                   0
1          0                  0.0                   0
2          0                  0.0                   0
3          0                  0.0                   0
4          0                  0.0                   0
…        …                    …                   …
28818      0                  0.0                   0
28819      1                 10.0                   1
28820      1                  2.0                   1
28821      0                  0.0                   0
28822      0                  0.0                   0

[28818 rows x 20 columns]
```

[9]:
```
## Step 9: Get the descriptive statistics of conversation lengths.
df['Length'].describe()
```

[9]:
```
count    28818.000000
mean         8.528906
std         29.007414
min          1.000000
25%          2.000000
50%          4.000000
75%          7.000000
max       1689.000000
Name: Length, dtype: float64
```

Step 9 interpretation: Discuss the mean value. The mean value we get from using .describe() shows us the average of the total number of tweets in conversations, from this we can see that the average number of tweets per conversation is 8-9.

[10]:
```
## Step 10: Get the descriptive statistics of number of unique users in the␣
 ↪conversation.
df['Num_users'].describe()
```

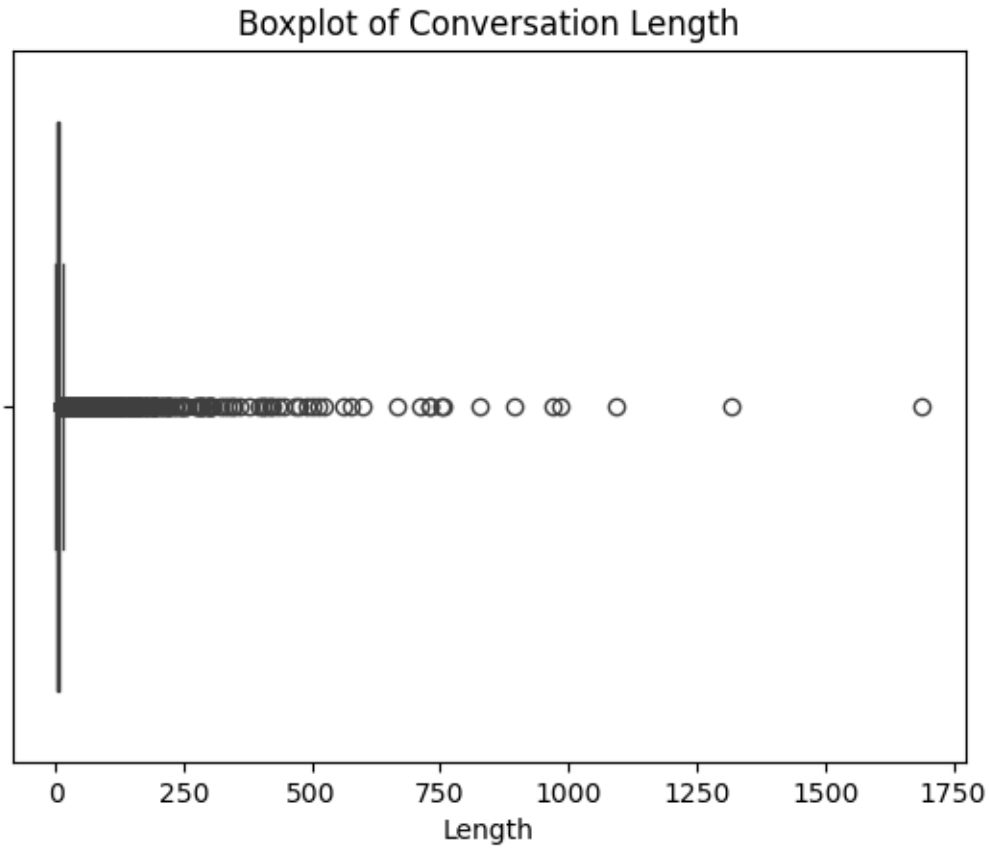
[10]:
```
count    28818.000000
mean         5.664064
std         20.054350
min          1.000000
25%          2.000000
50%          2.000000
75%          4.000000
max        925.000000
Name: Num_users, dtype: float64
```

Step 10 interpretation: Discuss the quartile values. The quartile values we get from using .de-

scribe() on the num_users column can show us the distribution of the number of unique users in conversations, from this we can see that 50 % of conversations in this data set had 2 or fewer unique users, which tells us that most conversations did not involve many different unique users, and only 25% of conversations had more than 4 showing only so many conversations involved more than a couple unique users.

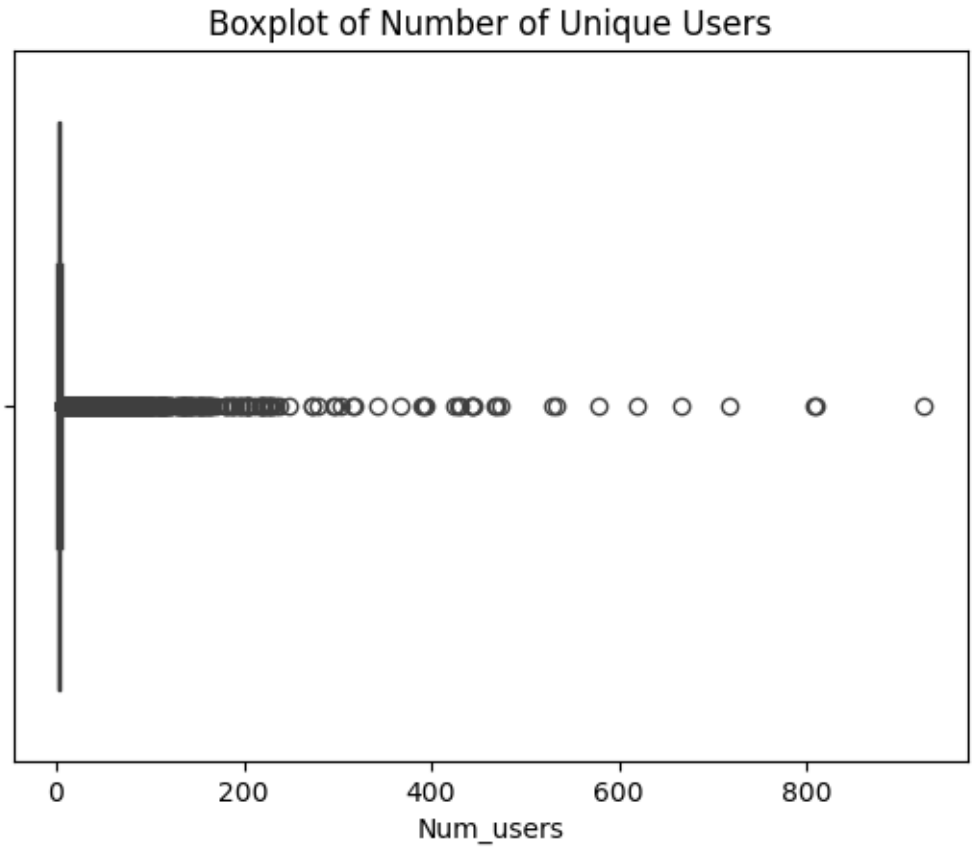
[11]:
```
## Step 11: Create boxplots for length and number of unique users in the␣
 ↪conversation and display
## Conversation Length box plot
%pip install seaborn
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(data = df, x = df['Length'])
plt.title("Boxplot of Conversation Length")
plt.show()
```

```
Matplotlib is building the font cache; this may take a moment.
```

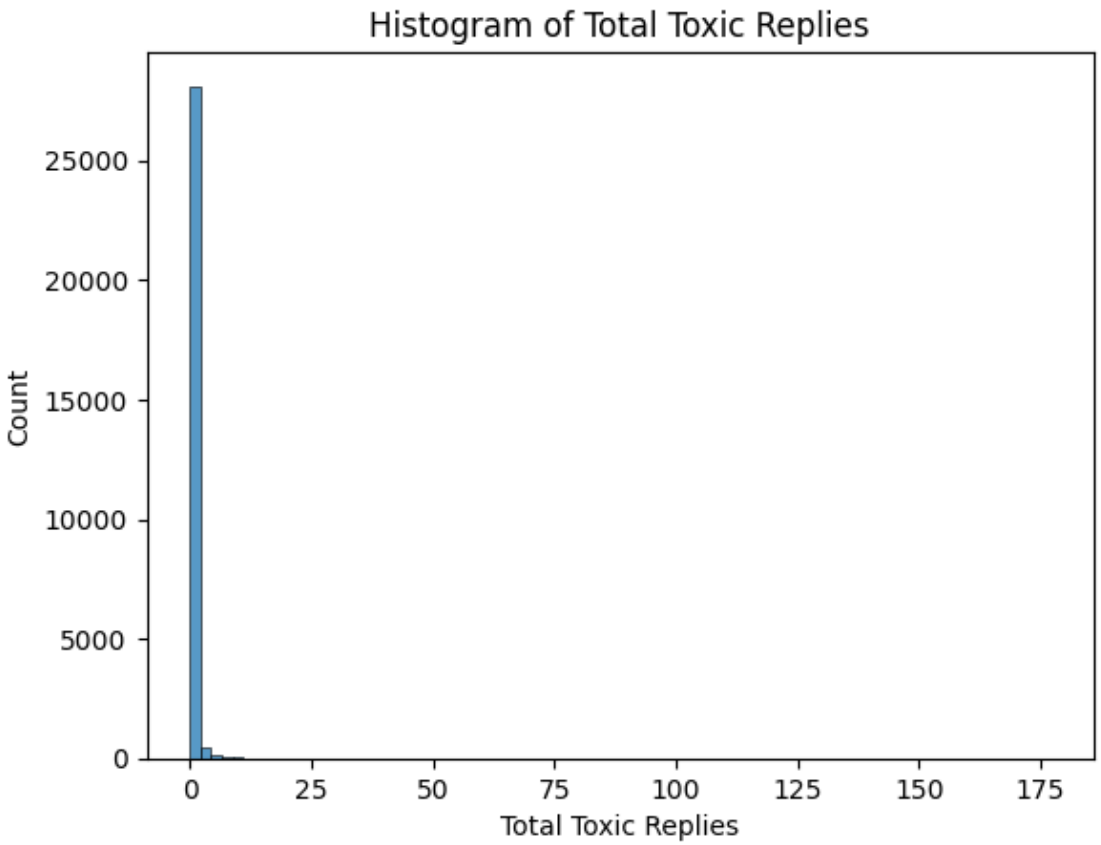


[12]:
```
## Number of unique users boxplot
sns.boxplot(data = df, x = df['Num_users'])
```

```
plt.title("Boxplot of Number of Unique Users in conversations")
plt.show()
```



[13]:
```
## Step 12: Create a histogram of Total_toxic_replies.
sns.histplot(data = df, x ="Total_toxic_replies", bins = 80)
plt.xlabel('Total Toxic Replies')
plt.title('Histogram of Total Toxic Replies')
plt.show()
```



Step 12 Interpretation: What do you observe? From this histogram I can see that the distribution is right skewed, meaning that a most of the conversations had very few total toxic replies, also noticed how a couple values are pretty big outliers as there are not many values over 25, none are even visible on the box plot, but the maximum is 177 after checking using .describe().

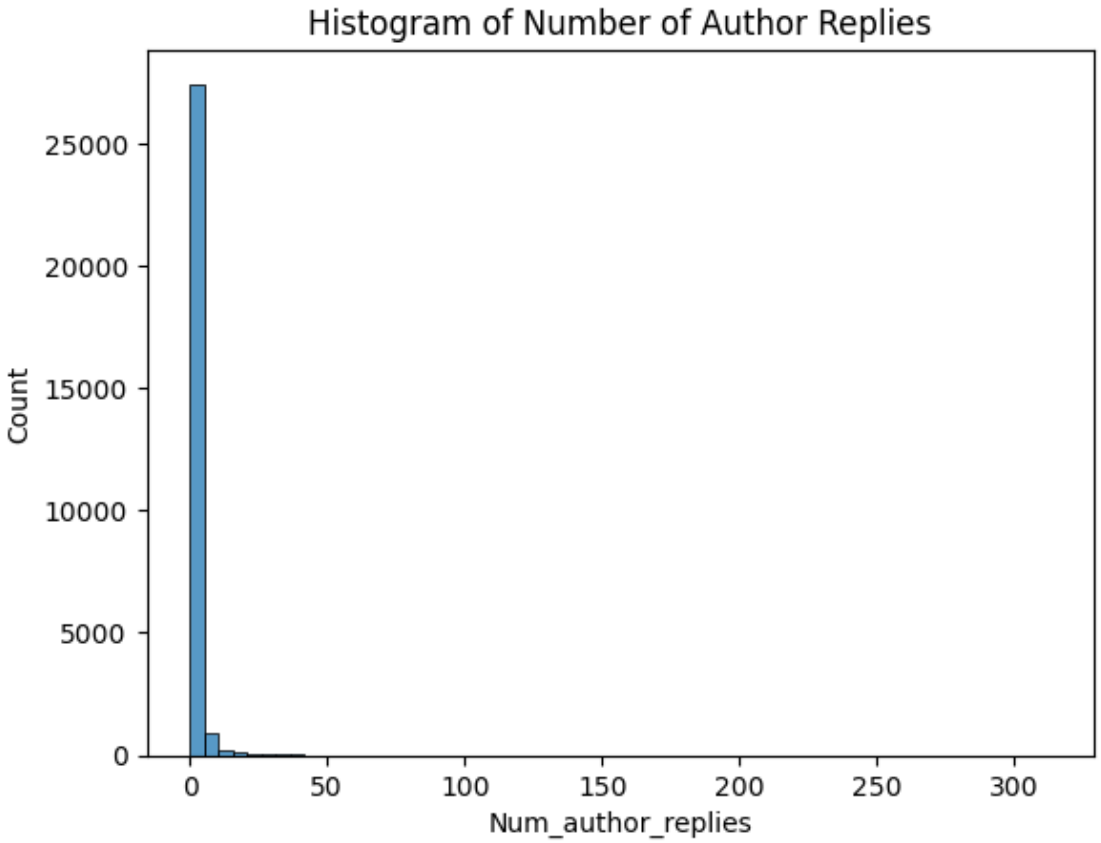
[14]:
```
## Step 13: Get the value count for Toxic_conversation
df['Toxic_conversation'].value_counts()
```

[14]:
```
Toxic_conversation
0    21613
1     7205
Name: count, dtype: int64
```

Step 13 interpretation: What do you observe? From these values we know a 0 represents a non toxic conversation, while a 1 represents a toxic conversation, we can see that a majority of the conversations were not toxic around 75%.

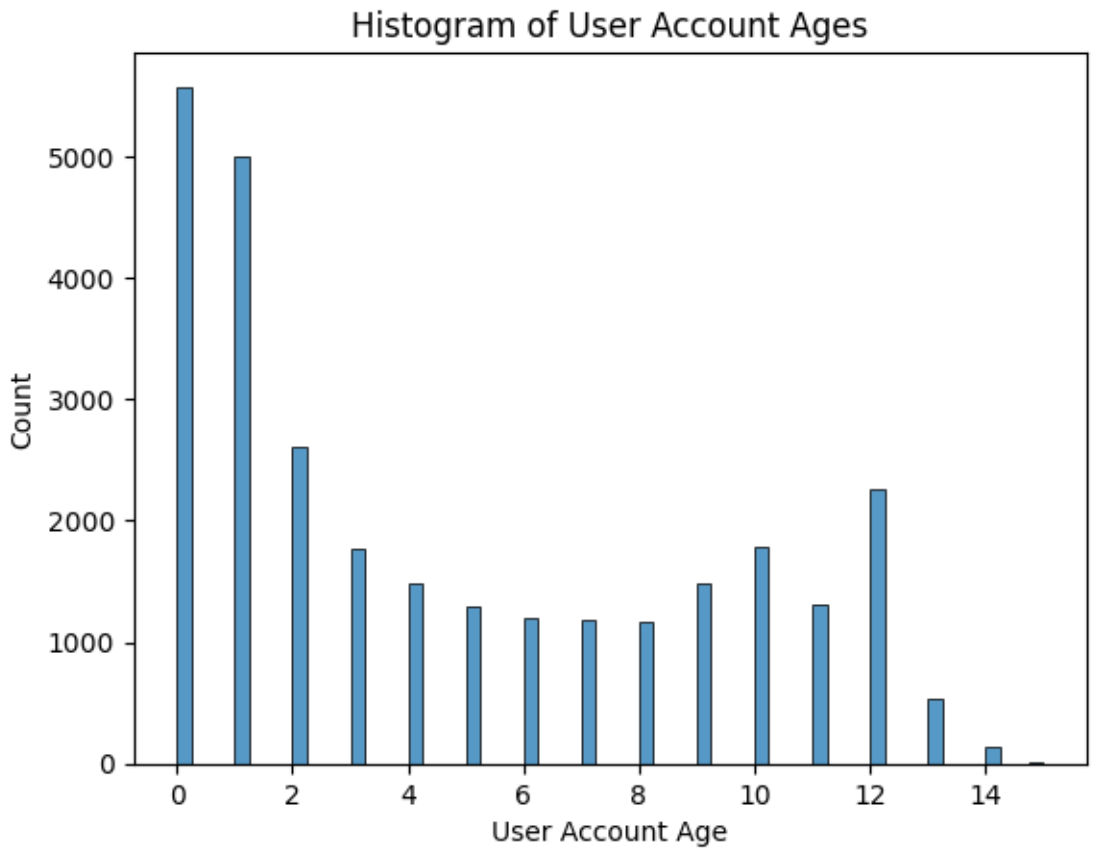
[15]:
```
## Step 14: Create a histogram of Num_author_replies.
sns.histplot(data = df, x= "Num_author_replies", bins=60)
plt.xlabel('Num_author_replies')
plt.title('Histogram of Number of Author Replies')
```

```
plt.show()
```



Step 14 interpretation: What do you observe? I notice how this histogram is very similarly distributed to the total toxic replies one, this shows that in most conversations besides some outliers the number of tweets from the original author are pretty much all below 50, and a a vast majority of those are well below 50 closer to 0 showing how most tweets do not have a ton of replies from the original author.

[16]:
```
## Step 15: Create a histogram of users ages.
sns.histplot(data = df, x= "Age", bins=60)
plt.xlabel('User Account Age')
plt.title('Histogram of User Account Ages')
plt.show()
```



Step 15 Interpetation: What do you observe? From this histogram, I can see that this is also right skewed, this shows there is a big cluster of new accounts in this data set as most fall under 3 years old, also seems like there are not as many older accounts meaning people either make new accounts or maybe stop being active, or maybe this data set was from a big surge in twitter use based on there being more new users than old ones.

[17]:
```
## Step 16: By using groupby function, find the mean length of toxic vs. non-toxic conversations.
## Groups the data based on whether the conversation is toxic (1) or non-toxic (0).
## Calculates the average length of conversations for each group.
df.groupby(df['Toxic_conversation'])['Length'].mean()
```

[17]:
```
Toxic_conversation
0     5.412483
1    17.877307
Name: Length, dtype: float64
```

Step 16 Interpretation: What do you observe? From these numbers, I can see that on average the length of the toxic conversations was far greter than the non toxic ones, meaning the toxic

conversations are getting a lot more interaction.

[18]:
```
## Step 17: By using groupby function, find the mean length of toxic vs.␣
 ↪non-toxic conversations.
## Group by whether the account is verified or not
## Compare the two groups based on the mean of their follower count
df.groupby(df['Verified'])['Followers'].mean()
```

[18]:
```
Verified
False      6022.879054
True     572177.384670
Name: Followers, dtype: float64
```

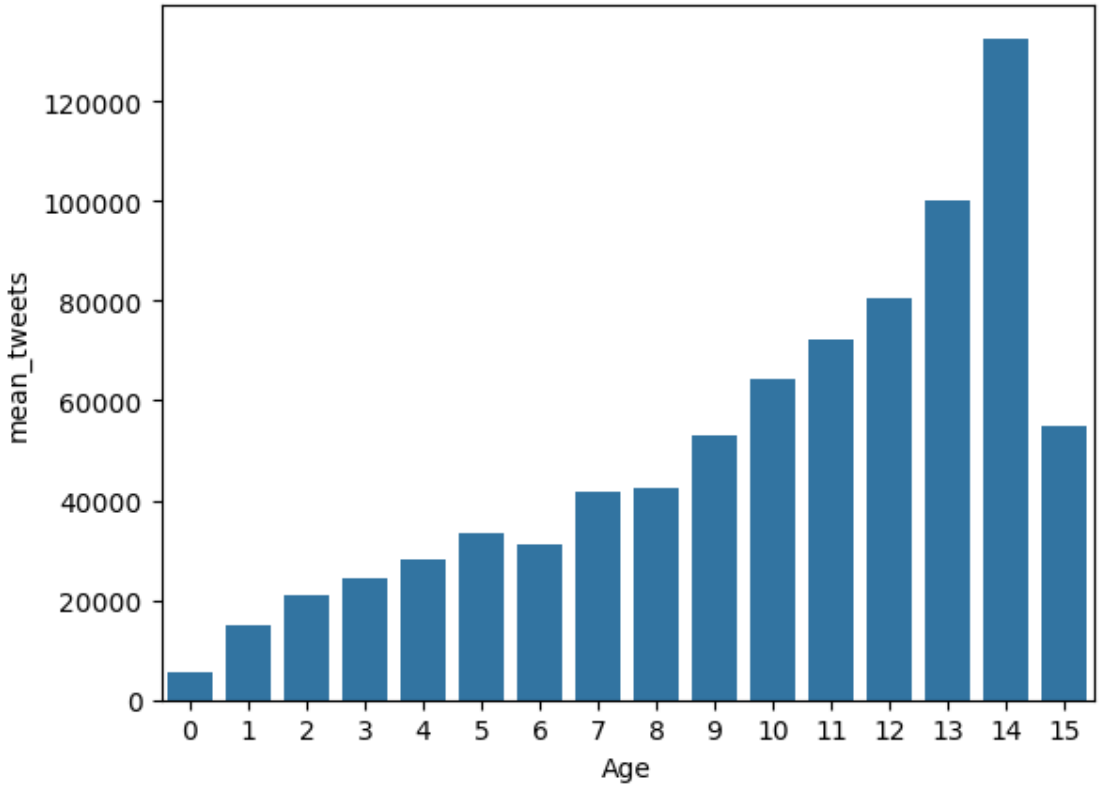
Step 17 Interpretation: What do you observe? From the means of the followers of these two groups, it is clear that verified accounts on average have a significantly larger follower account than those who are not verified.

[19]:
```
## Step 18: By using a groupby function, get the mean number of posted tweets␣
 ↪by main user for each account age.
## By resetting the index, name the new column as 'mean_tweets'.
mean_tweets_by_age = df.groupby('Age')['Num_tweets'].mean().reset_index()

## Rename the column for clarity
mean_tweets_by_age.rename(columns={'Num_tweets': 'mean_tweets'}, inplace=True)

## Display the result
print(mean_tweets_by_age)
```

```
    Age    mean_tweets
0     0    5373.940649
1     1   15108.871605
2     2   20949.909438
3     3   24357.895632
4     4   28330.858776
5     5   33349.213127
6     6   31184.233864
7     7   41906.631313
8     8   42550.933219
9     9   52964.504365
10   10   64442.248458
11   11   72244.098473
12   12   80569.417699
13   13  100239.100379
14   14  132627.958333
15   15   54874.250000
```

[20]:
```
## Plot the bar chart for age and mean tweets
sns.barplot(x = 'Age', y = 'mean_tweets', data = mean_tweets_by_age)
```

[20]: <Axes: xlabel='Age', ylabel='mean_tweets'>



Step 18 Interpretation: What do you observe from the bar chart? This bar chart shows us how the older an account is, the more tweets that account has likely posted and younger accounts have likely posted less. The 15 year old accounts average is lower than what might have been expected based on the rest of the graph, there may be some outliers or maybe a lot of those accounts went inactive or made a new account or maybe there was not many acccounts created at that time. There is a pretty exponential increase by age, which shows evidence there might be some sort of a positive relation between account age and the mean number of tweets.

[21]:
```
## Step 19: Create a crosstab of Toxic_conversation and verified columns.
toxic_verified_crosstab = pd.crosstab(df['Toxic_conversation'], df['Verified'])

## Display the result
print(toxic_verified_crosstab)
```

```
Verified            False  True
Toxic_conversation
0                   20802    811
1                    6607    598
```

Step 19 Interpretation: What do you observe? From this crosstab I can see that most conversations are from non-verified users and they more toxic and non toxic replies mostly because of the fact that

verified users have 1/20th the amount of tweets as non verified users in this data set. If we look at the rate of toxicty though, it is seen that a much higher percentage of verified users conversations are toxic in this data set than those who are not verified. This data suggests that verified users might create toxic interactions at a higher rate than usual.

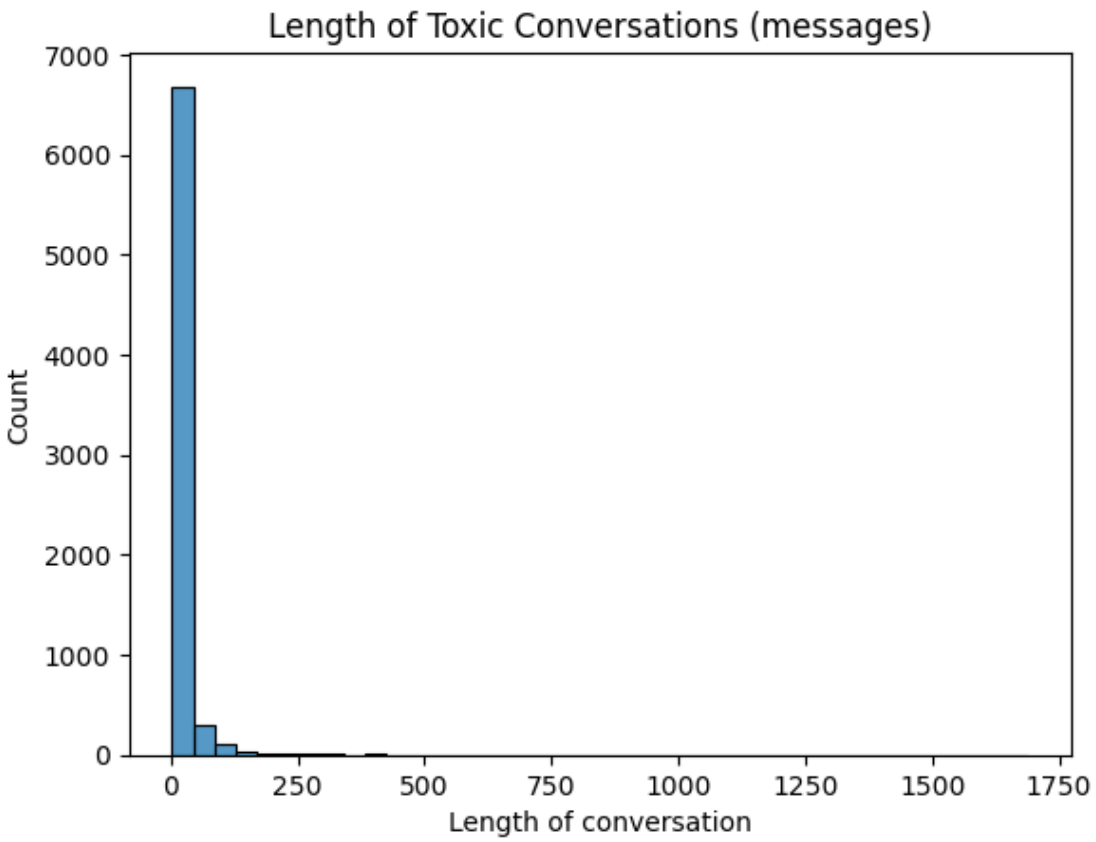
[22]:
```
## Step 20: Using created frequency table, run the Chi-square test
## compare the observed and expected number of verified users in toxic vs.␣
 ↪non-toxicconversations
from scipy.stats import chi2_contingency

chi2, p, dof, ex = chi2_contingency(toxic_verified_crosstab)

print("chi2 = ", chi2)
print("p = ", p)
print("degrees of freedom = ", dof)
print("expected = ", ex)
```

```
chi2 =  239.3148979065939
p =  5.547396807333776e-54
degrees of freedom =  1
expected =  [[20556.27444653  1056.72555347]
 [ 6852.72555347   352.27444653]]
```

Step 20 Interpretation: State your null and alternative hypotheses. Then, run the test. State what do the results suggest in terms of rejecting or failure to reject H0. H0: There is no association between user verification status and toxicity level of conversations. HA: There is a significant association between user verification status and toxicty level of conversations. The results from the chi2 test give us a P value that is basically 0, telling us we should reject the null hypothesis that there is no association between verification and toxicty level of conversations, showing us there is a significant association between the two.

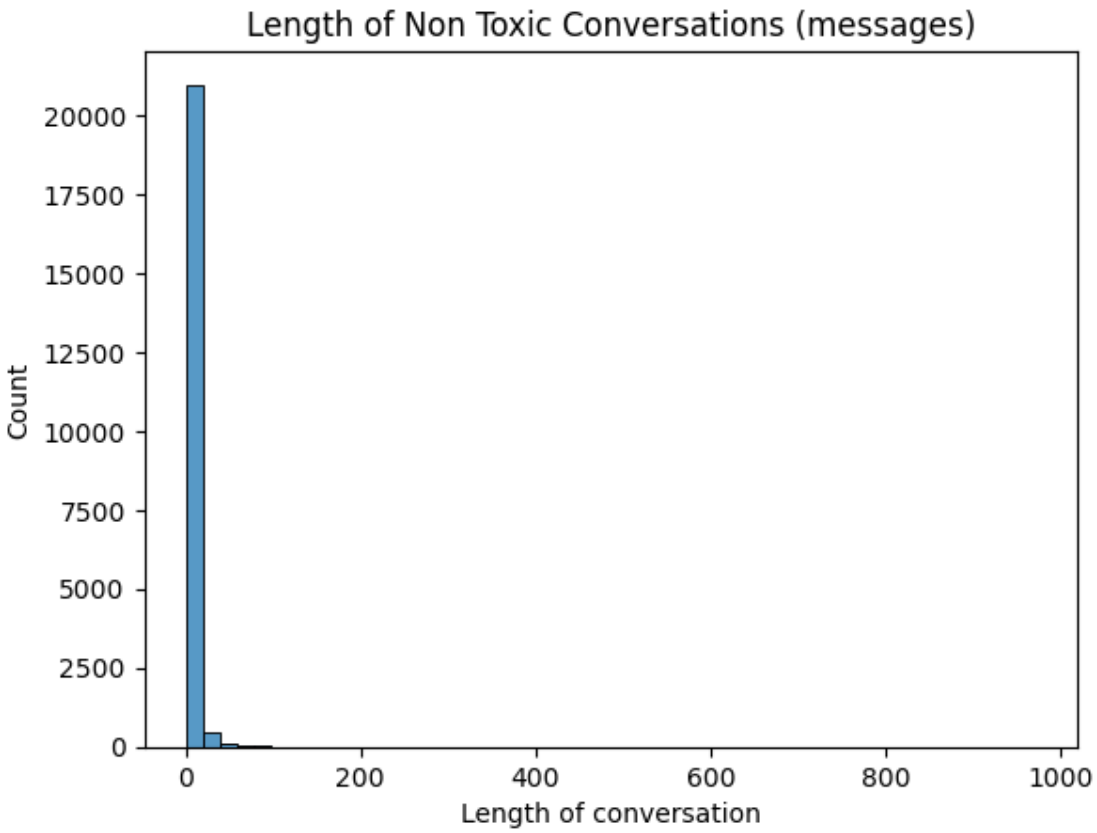
[23]:
```
## Step 21: Split the dataframe into two distinct dataframes: one only␣
 ↪containing toxic conversations and the other containing non-toxic
## Create a df containing only toxic conversations (Toxic_conversation = 1)
toxic_df = df[df['Toxic_conversation'] == 1]

## Create a df containing only non-toxic conversations (Toxic_conversation = 0)
non_toxic_df = df[df['Toxic_conversation'] == 0]
```

[24]:
```
## Step 22: Plot the histogram of conversation length of toxic conversations
sns.histplot(data = toxic_df, x= "Length", bins =40)
plt.xlabel('Length of conversation')
plt.title('Length of Toxic Conversations (messages)')
plt.show()
```



Step 22 Interpretation: What is the distribution? The distribution of this histogram is skewed right, meaning most of the conversations are shorter, but there is some outliers such as one conversation with 1689 messages, this distrubtion also suggests that most toxic conversations die out pretty quickly, besides some outliers.

[25]:
```
## Step 23: Plot the histogram of conversation length of non toxic conversations
sns.histplot(data = non_toxic_df, x= "Length", bins =50)
plt.xlabel('Length of conversation')
plt.title('Length of Non Toxic Conversations (messages)')
plt.show()
```



Step 23 Interpretation: What is the distribution? The histogram is very similar to the toxic conversation one, it shows a right-skewed distribution, meaning most non-toxic conversations are short, but there are also some outliers, though not as extreme or as often as the toxic conversation histogram, through these histograms the distribution of the length of non toxic conversations and toxic conversations seem to be pretty similar aside from Toxic conversations having more conversations with a length of 200-500 messages, and higher outliers which could suggest that the toxicicity of conversations might have an impact on the amount of interaction a tweet recieves.

[26]:
```
## Step 24: Run a Mann-Whitney test to compare lengths of toxic vs. non-toxic␣
↪conversations
from scipy.stats import mannwhitneyu
mannwhitneyu(toxic_df['Length'],non_toxic_df['Length'])
```

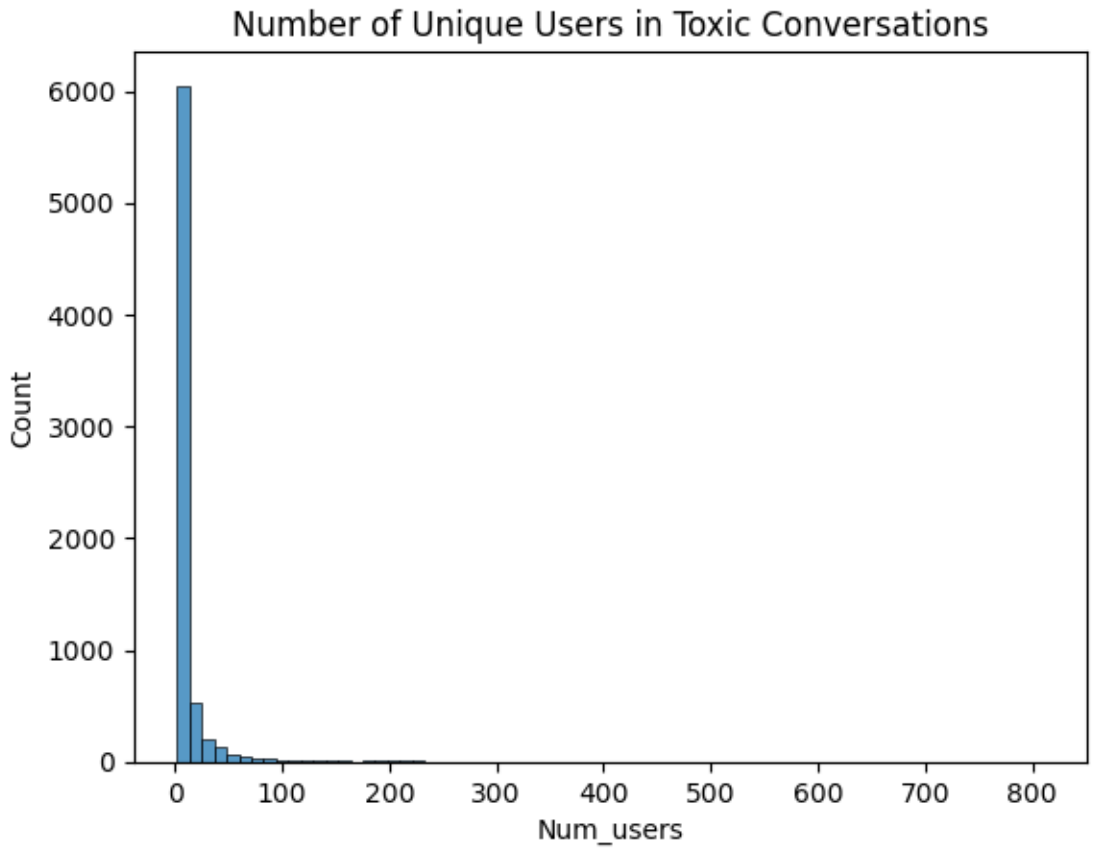
[26]:
```
MannwhitneyuResult(statistic=np.float64(113183967.0), pvalue=np.float64(0.0))
```

Step 24 Interpretation: State your null and alternative hypotheses. Then, run the test. State what do the results suggest in terms of rejecting or failure to reject H0. H0: There is no significant difference in conversation lengths between toxic and non toxic conversations. HA: There is a significant difference in conversation lengths between toxic and non-toxic conversations.

The extremely low P value of 0 suggests that we should reject the null hypothesis that there is

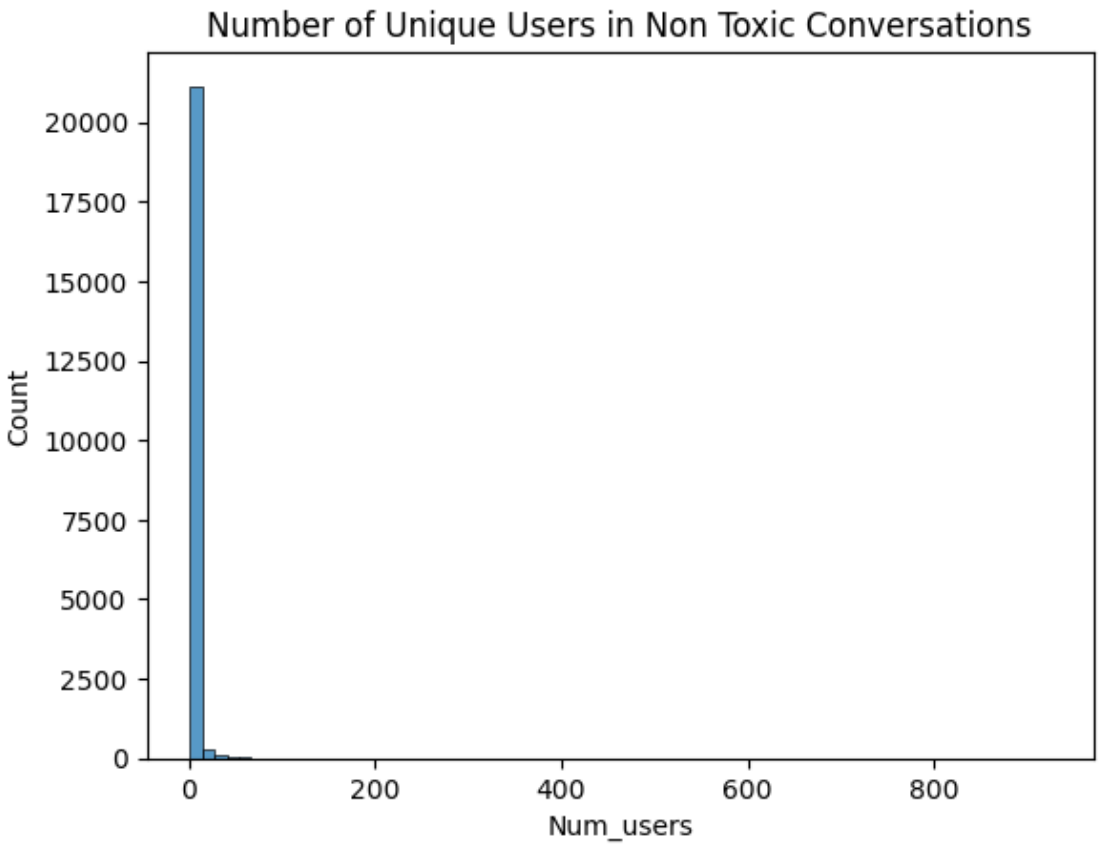
no significant difference in conversaiton length between toxic and non toxic conversations. This data suggests that there is a statistically significant difference in the lengths of non toxic and toxic conversations.

[27]:
```
## Step 25: Plot the histogram of number of unique users of toxic conversations.
sns.histplot(data = toxic_df, x= "Num_users", bins = 70)
plt.xlabel('Num_users')
plt.title('Number of Unique Users in Toxic Conversations')
plt.show()
```



Step 25 Interpretation: What is the distribution? The distribution of this histogram is heavily skewed right, most of the conversations have fewer number of users, but there are definitely some high outliers and quite a bit of conversations with 100-200 unique users, more than non toxic conversations.

[28]:
```
## Step 26: Plot the histogram of number of unique users of non-toxic␣
 ↪conversations.
sns.histplot(data = non_toxic_df, x= "Num_users", bins = 70)
plt.xlabel('Num_users')
plt.title('Number of Unique Users in Non Toxic Conversations')
plt.show()
```



Step 26 Interpretation: What is the distribution? The distribution of this histogram is also skewed right, by the distribution of the histograms it also might suggest that the non toxic conversations in general have less unique users interacting in the conversation.

[29]:
```
## Step 27: Run a Mann-Whitney test to compare number of unique users of toxic␣
↪vs. non- toxic conversations.
from scipy.stats import mannwhitneyu
mannwhitneyu(toxic_df['Num_users'],non_toxic_df['Num_users'])
```

[29]:
```
MannwhitneyuResult(statistic=np.float64(108388312.5), pvalue=np.float64(0.0))
```

Step 27 Interpretation: State your null and alternative hypotheses. Then, run the test. State what do the results suggest in terms of rejecting or failure to reject H0 H0: There is no statistically significant difference in the number of unique users in toxic and non toxic conversations. HA: There is a statistically significant difference in the number of unique users in toxic and non toxic conversations. With an extremely low P value from the Mann-Whitney test, we should reject the null hypothesis that there is no statistically significant difference between the number of unique users in toxic and non toxic conversations, the data suggests that there is a statistically significant difference between the number of unique in a conversation based on whether the conversation is toxic or not.

[30]:
```
## Step 28: Run t-tests for the tasks in Step 27 and Step 24.
## T- test for step 24
from scipy.stats import ttest_ind

ttest_ind(toxic_df['Length'],non_toxic_df['Length'])
```

[30]:
```
TtestResult(statistic=np.float64(32.1487580990876),
pvalue=np.float64(7.98320672131386e-223), df=np.float64(28816.0))
```

[31]:
```
## T- test for step 27
ttest_ind(toxic_df['Num_users'],non_toxic_df['Num_users'])
```

[31]:
```
TtestResult(statistic=np.float64(28.717472597708117),
pvalue=np.float64(7.660097368029594e-179), df=np.float64(28816.0))
```

Step 28 Interpretation: State whether the results are similar to Mann-Whitney tests. Which test is more appropriate in these scenarios and why? The results from the t-test are not similar to the Mann-Whitney test at all, the P value is not even close to 0 from the t-test, in statistics you must determine what tests you should run on the data based on it's distribution. If this data set was normally distributed, we could have used a t test and came up with statistically significant results, but because the data is far from normal and right skewed, the Mann-Whitney test is what was needed to analyze the data.