TWITTER RESEARCH

There were 3 datasets found after an exhaustive research.

DATASET 1:

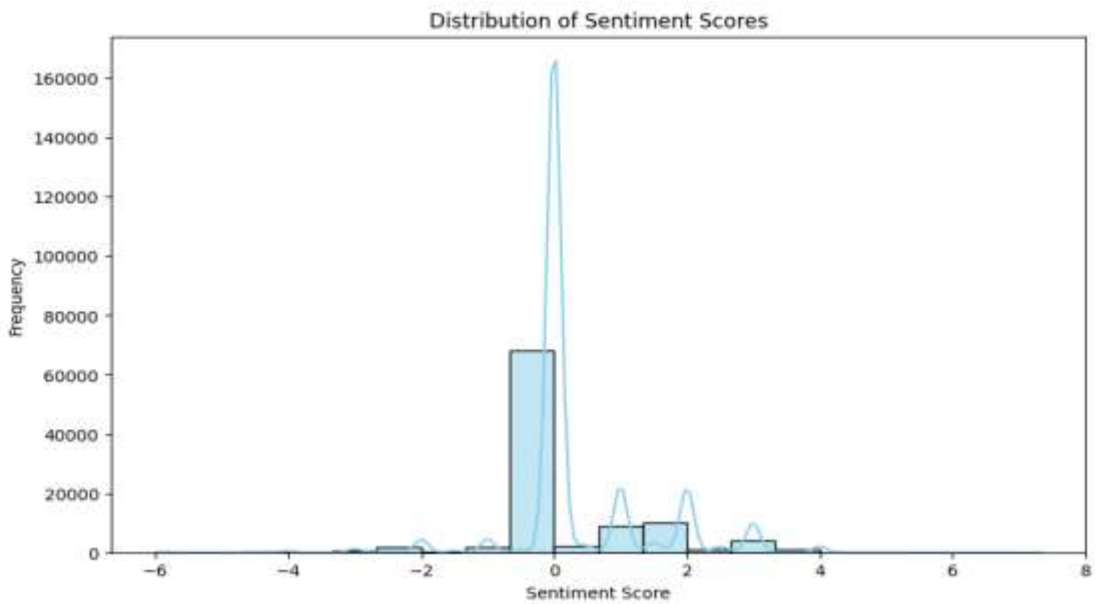
About the dataset:

```
Data columns (total 14 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   TweetID       100000 non-null  object
 1    Weekday      100000 non-null  object
 2    Hour         100000 non-null  float64
 3    Day          100000 non-null  float64
 4    Lang         100000 non-null  object
 5    IsReshare    100000 non-null  object
 6    Reach        100000 non-null  float64
 7    RetweetCount 100000 non-null  float64
 8    Likes        100000 non-null  float64
 9    Klout        100000 non-null  float64
 10   Sentiment    100000 non-null  float64
 11   text         100000 non-null  object
 12   LocationID   100000 non-null  float64
 13   UserID       100000 non-null  object
dtypes: float64(8), object(6)
```

These are the columns in the dataset.

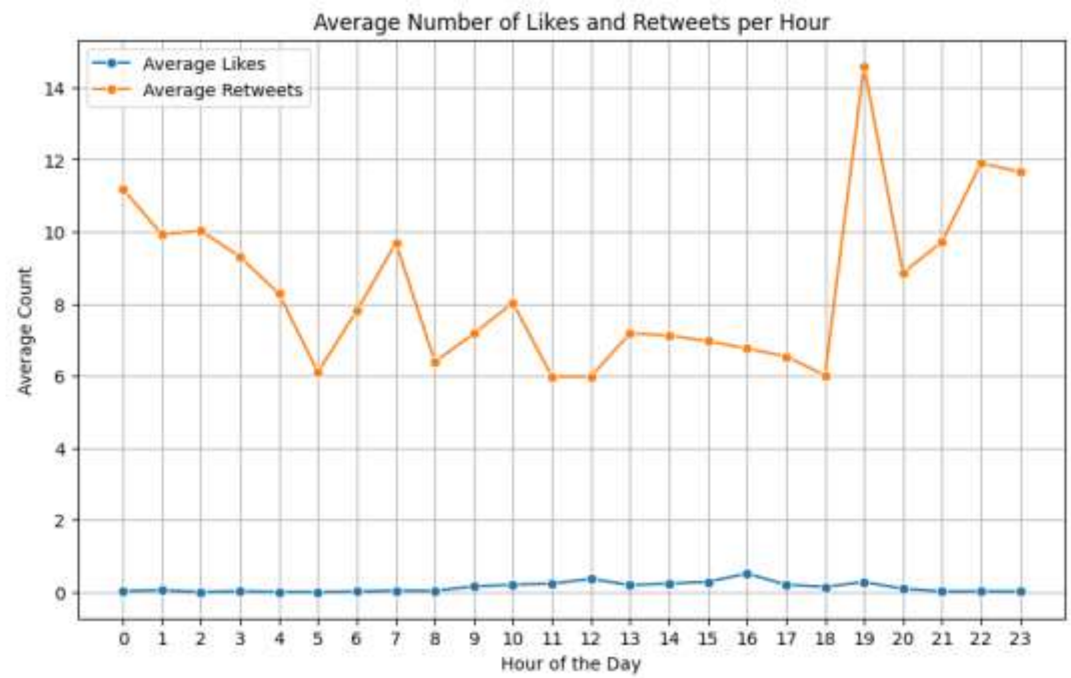
Exploring the dataset:

1.



The sentiment score 0 has the highest frequency indicating that the sentiment scores are neutral, and more of them being positive and less being negative.

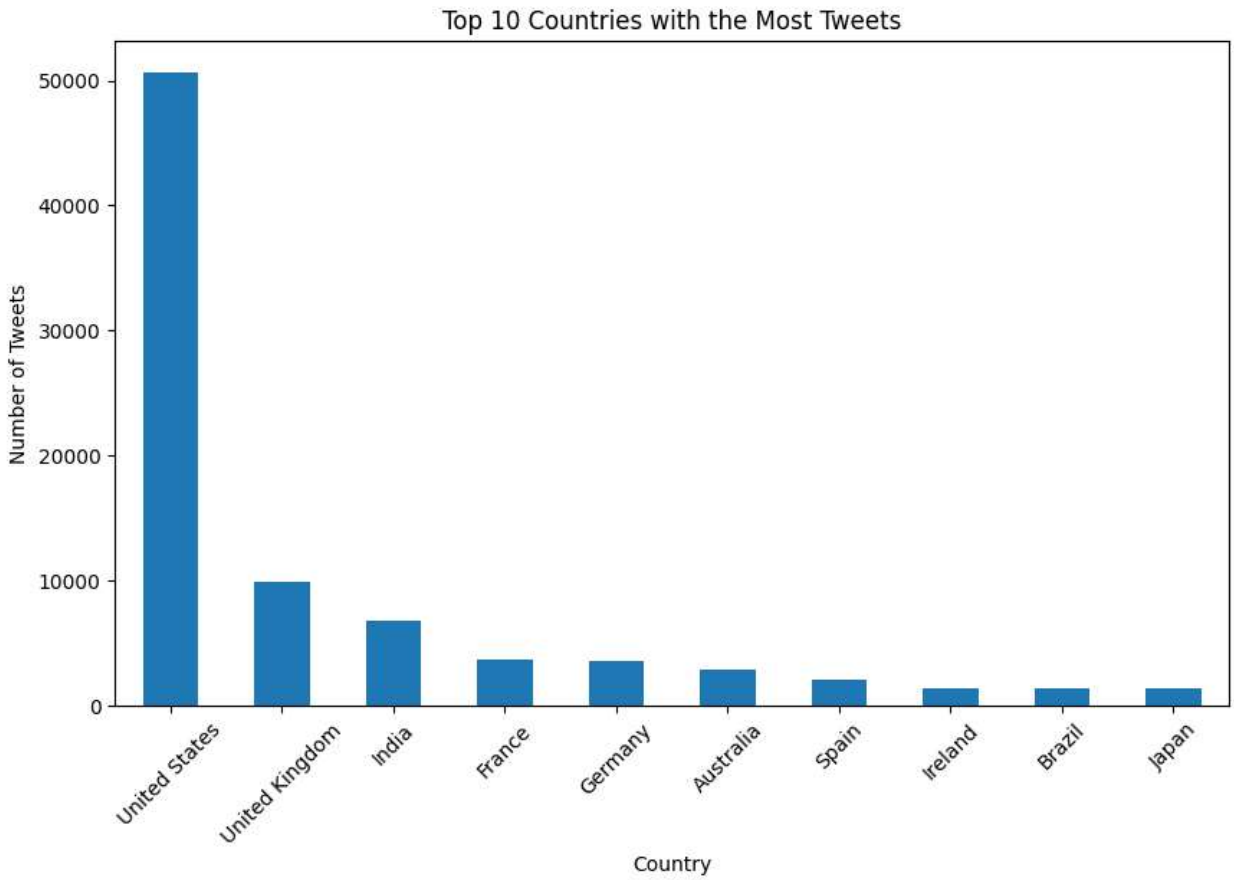
2.



As seen above the above graph we can see that the number of likes per hour are very less as compared to retweets so we check the max, min and mean scores of likes and retweets as show below:

	Hour	Day	Reach	RetweetCount	Likes	Klout	Sentiment	LocationID
count	100000.000000	100000.000000	1.000000e+05	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000
mean	11.412490	15.894960	8.542396e+03	8.052750	0.152770	40.389260	0.380921	2836.163440
std	6.053577	8.399852	8.867027e+04	97.863474	2.583633	13.636513	1.046559	1323.140242
min	0.000000	1.000000	0.000000e+00	0.000000	0.000000	0.000000	-6.000000	1.000000
25%	7.000000	9.000000	1.510000e+02	0.000000	0.000000	32.000000	0.000000	1601.000000
50%	11.000000	16.000000	4.485000e+02	0.000000	0.000000	43.000000	0.000000	3738.000000
75%	16.000000	23.000000	1.496000e+03	3.000000	0.000000	49.000000	0.666667	3775.000000
max	23.000000	31.000000	1.034245e+07	26127.000000	133.000000	99.000000	7.333333	6289.000000

3.



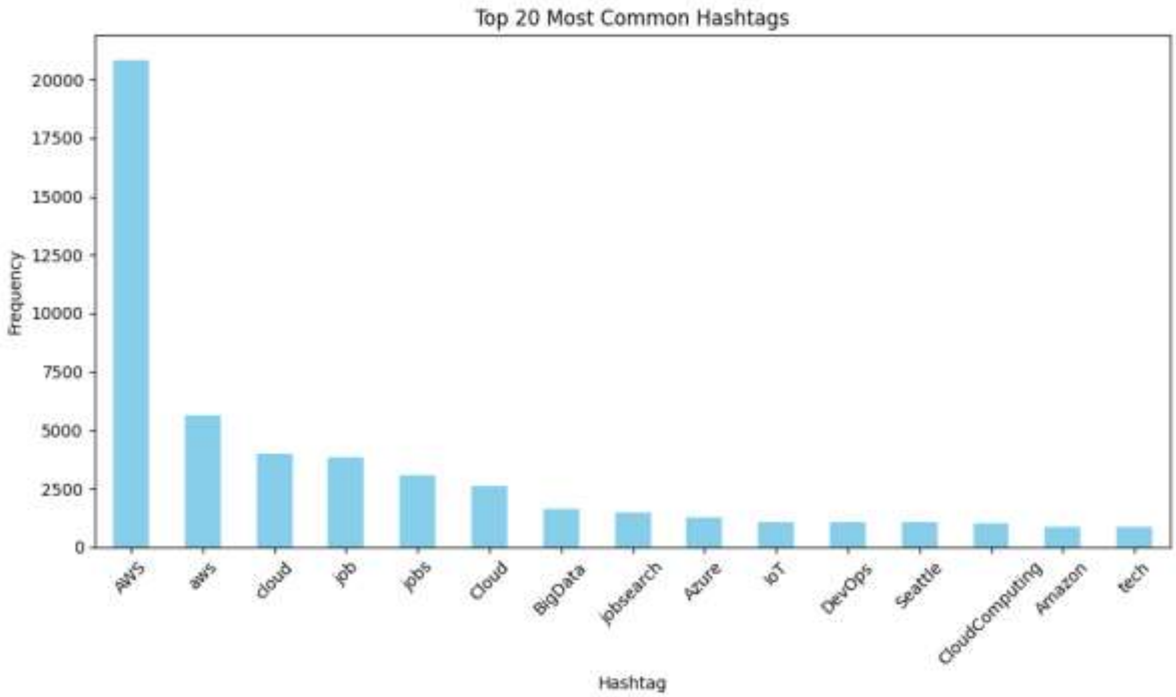
USA is the country having the most number of tweets.

```
Locations with the most tweets:
State
California        11567
Washington         8248
Greater London     5096
New York           3876
Massachusetts      3840
Texas              3107
Ile-de-France      2796
Maharashtra        1996
0                  1710
Illinois           1709
```

We also remove the data for most number of tweets in a state.

Engagement metrics:

1.



We extract the hashtags from the tweets and then look at the frequency of the hashtags in the tweets as seen above.

2.

A wordcloud indicating the most used words in the tweet.

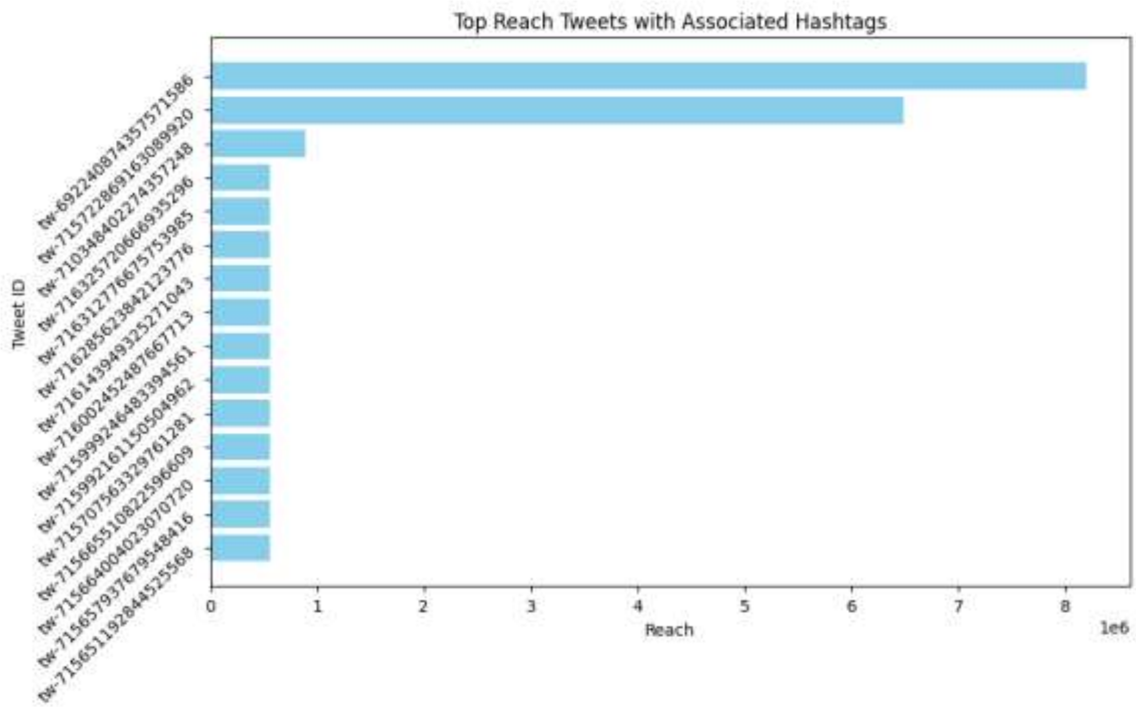
By looking at the most used hashtags and the most used words we can conclude that topics about tech (big data) likes AWS, Cloud, etc. are the highest.

3.

```
Top hashtags in tweets with the highest reach:
 text
AWSSummit         4
DataWarehouse     2
CloudComputing    2
BigData           2
5ab1db3649e1      1
Build2016         1
LiveWireTV        1
SLAPTV            1
Aurora            1
DevOps            1
Name: count, dtype: int64
```

These are the hashtags with the highest reach.

4.



These are the tweets which have the highest reach.

5.

We now we extract the 'Total Interactions' and 'Sharevoice':

```
interaction_metrics = [' Likes', ' RetweetCount', ' Reach', ' Klout']
df['TotalInteractions'] = df[interaction_metrics].sum(axis=1)
user_interactions = df.groupby('UserID')['TotalInteractions'].sum().reset_index()

user_interactions_sorted = user_interactions.sort_values(by='TotalInteractions',
ascending=False)

total_forum_interactions = user_interactions_sorted['TotalInteractions'].sum()

user_interactions_sorted['ShareVoice'] =
(user_interactions_sorted['TotalInteractions'] / total_forum_interactions) * 100

print("Most interactive public accounts of the forum/group:")
print(user_interactions_sorted[[' UserID', 'TotalInteractions',
'ShareVoice']].head(10))
```

Explaining the code:

The code calculates the total interactions for each row (account) in the dataset by summing up the values of the interaction metrics along the rows and storing the result in a new column called 'TotalInteractions'.

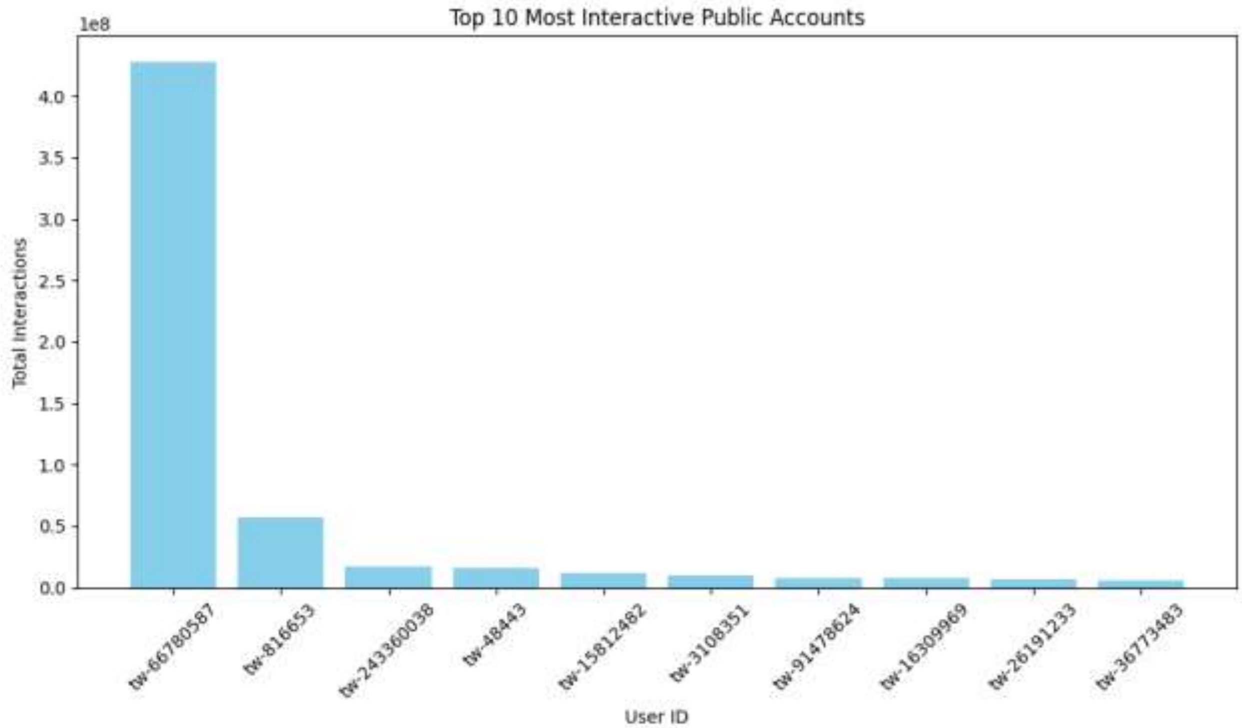
Then we we group the dataset by the 'UserID' column and calculate the sum of total interactions for each user. We then reset the index to make the 'UserID' a regular column instead of an index.

Then we sort the users based on their total interactions in descending order, so users with the highest total interactions appear first.

Then we calculate the total interactions for the forum or group by summing up the total interactions of all users. Then, we compute the share of voice for each user by dividing their total interactions by the total forum interactions and multiplying by 100 to get the percentage.

OUTPUT:

```
Most interactive public accounts of the forum/group:
             UserID  TotalInteractions  ShareVoice
29313   tw-66780587        427738375.0   49.789179
31215     tw-816653         56684294.0    6.598109
14149  tw-243360038         16651761.0    1.938282
26141      tw-48443         15531820.0    1.807920
6386    tw-15812482         12327628.0    1.434948
18995    tw-3108351         10342577.0    1.203886
32325   tw-91478624          8196520.0    0.954083
6985    tw-16309969          7838310.0    0.912387
15434   tw-26191233          6925877.0    0.806179
22165   tw-36773483          6014154.0    0.700054
```



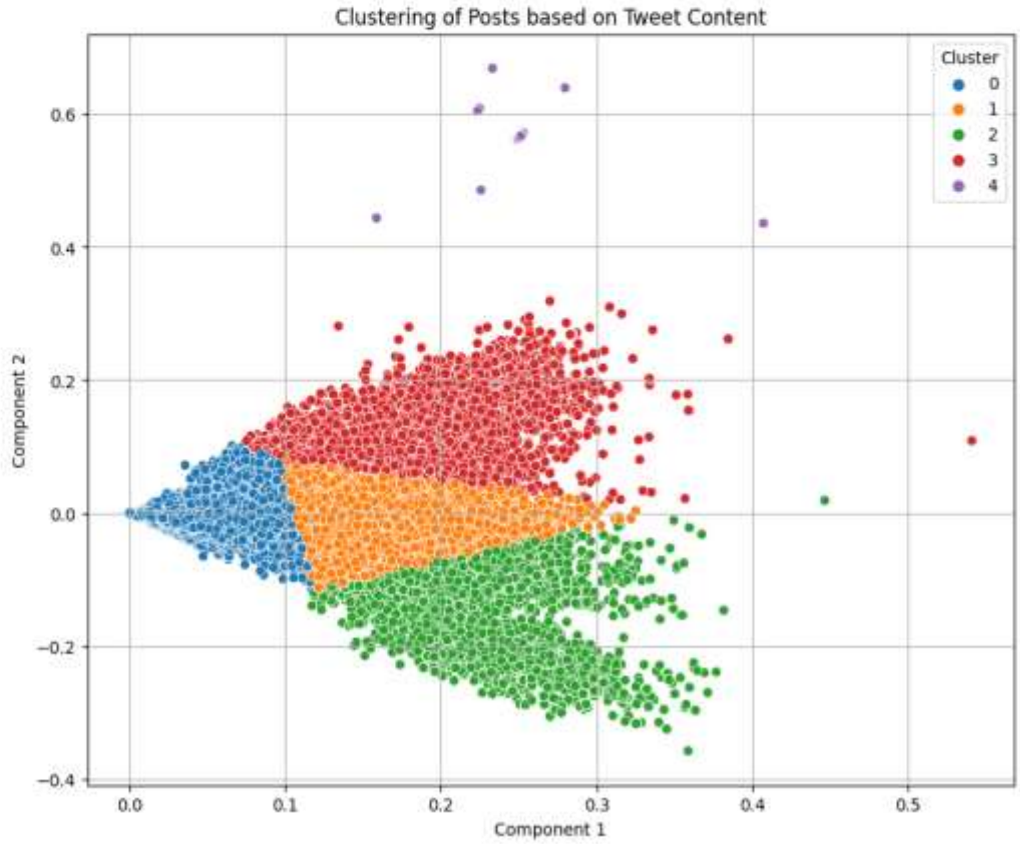
These are the most interactive public accounts based on the code we wrote above.

6.

```
Details for account 'tw-66780587':
Interactions: 427738375.0
Share voice: 49.79%
Posts: 845
Rate: 0.84%
```

These are the metrics for the most interactive twitter account in the dataset. We can do the same for as many accounts as we want based on the code written.

7.



We use the TFIDF vectorizer to convert the tweets into numerical vectors.

Then we cluster the data according to the similarity in the vectors, showing some similarity in the tweet.

I have also printed some example tweets in the code of each cluster.

Dataset 2:

The second dataset I found after extensive research was a dataset of verified users.

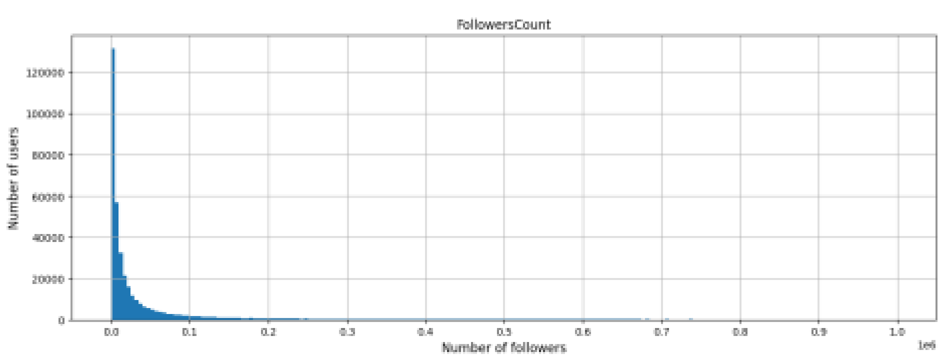
Exploring the dataset:

1.

```
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   #ID              384494 non-null  int64
 1   ScreenName       384493 non-null  object
 2   Protected        384494 non-null  bool
 3   Verified         384494 non-null  bool
 4   FriendsCount     384494 non-null  int64
 5   FollowersCount   384494 non-null  int64
 6   ListedCount      384494 non-null  int64
 7   StatusesCount    384494 non-null  int64
 8   CreatedAt        384494 non-null  object
 9   URL              384465 non-null  object
 10  ProfileImageURL  384485 non-null  object
 11  Location         311171 non-null  object
 12  Relation         384494 non-null  object
 13  Subject          384494 non-null  object
dtypes: bool(2), int64(5), object(7)
```

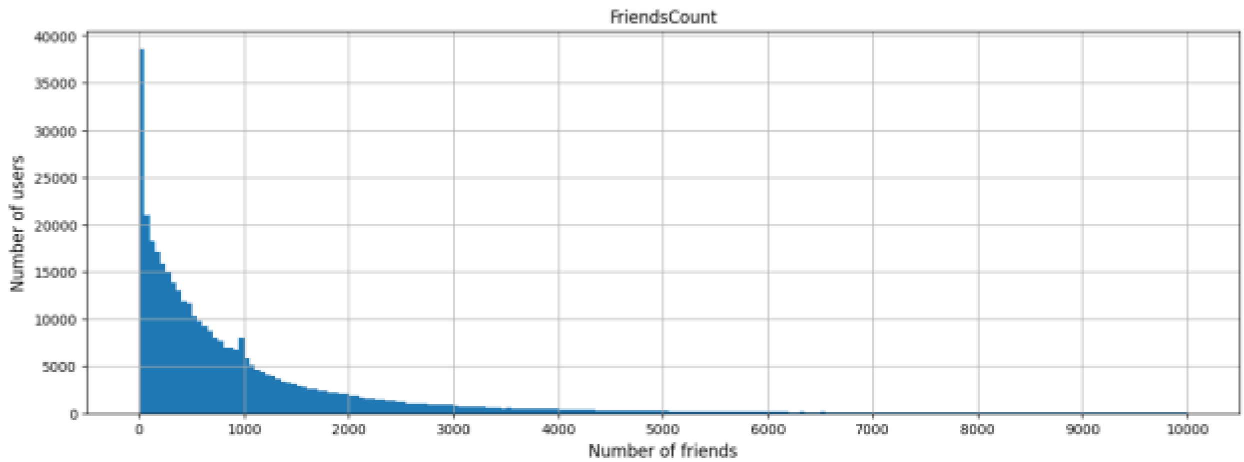
These were the columns in the dataset.

2.



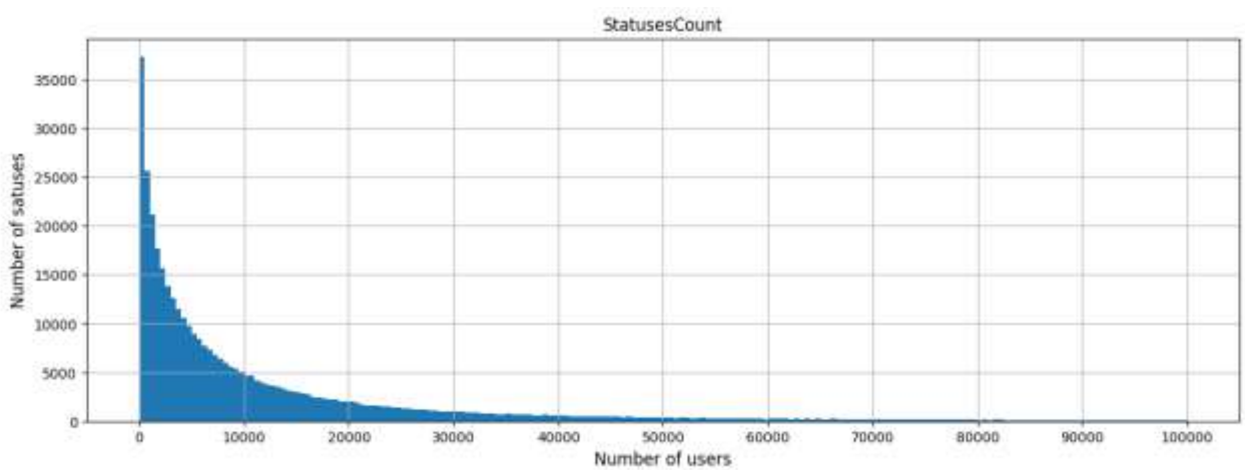
The above graph shows the distribution of the number of followers among users, providing insights into the popularity or reach of different users in the dataset.

3.



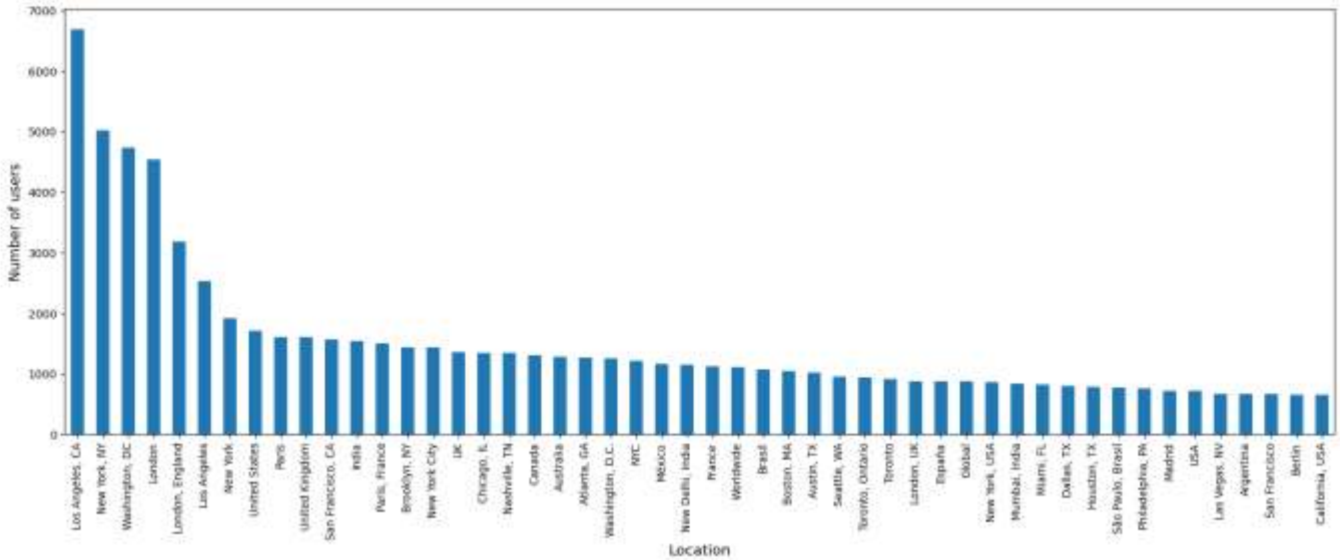
The same for the 'Friendcount'.

4.



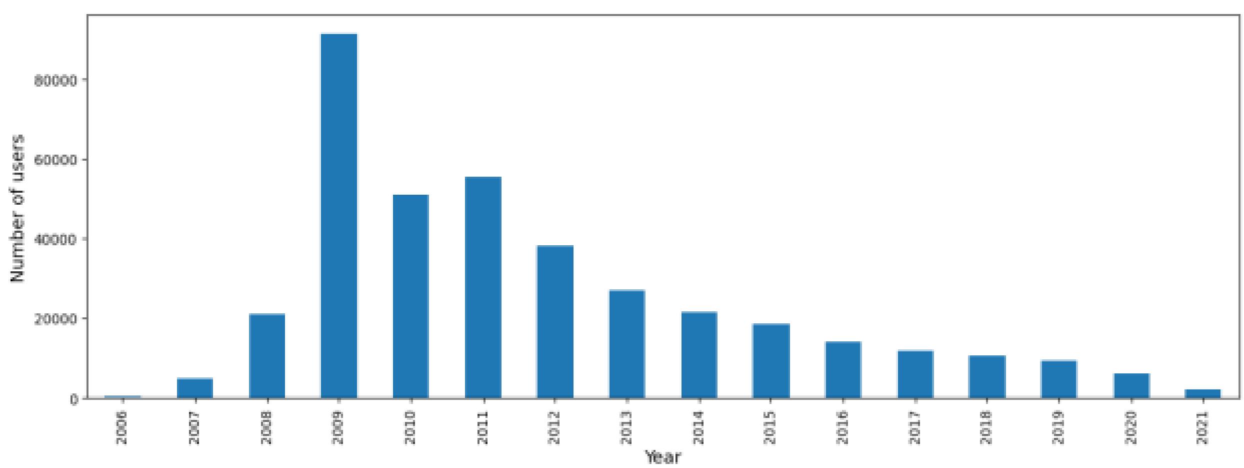
The same for 'StatusCount'.

5.



The above graph shows the frequency of users in a location.

6.



The above graph shows that a big chunk verified accounts were created in 2009, which was when Twitter was most popular. Since then there has been a steady decline, and in the last two years, this may be due to the fact that Twitter has paused public submissions for verification since early 2018.

7.

```
Number of nodes (users): 4989
Number of edges (follow relationships): 5000
Average degree: 2.0044097013429547
Average clustering coefficient: 0.0
```

I tried making a social network graph for the dataset but it was not complete and threw an error because the dataset doesn't contain such data where the path length can't be found, you can mention that in real-world social networks, it's common for not all users to be directly connected to each other. In this dataset of Twitter, users may follow others who don't follow them back, creating a directed graph where not all nodes are reachable from each other. This results in disconnected components in the graph, leading to the NetworkX error I encountered.

Engagement Metrics:

```
Top 5 accounts with the highest number of friends:
            ScreenName  FriendsCount
352916  6BillionPeople       4190657
312692  liferdefempire       2282592
146675  Karabo_Mokgoko       2077458
343468     Starbucks_J       1690545
316802       benlandis       1582312

Top 5 accounts with the highest number of followers:
          ScreenName  FollowersCount
384461   BarackObama       130027990
383943  justinbieber       114016239
322112     katyperry       108788824
381186       rihanna       103008128
382917     Cristiano        94420848

Top 5 accounts with the highest number of both friends and followers:
          ScreenName  TotalFollowersFriends
384461   BarackObama              130617140
383943  justinbieber              114303382
322112     katyperry              108789061
381186       rihanna              103009130
382917     Cristiano               94420905
```

I calculated the accounts with the highest number of friends, followers and both.

And as you can see the accounts with the highest number of followers are well known celebrities.

Dataset 3:

This was another twitter dataset found after research.

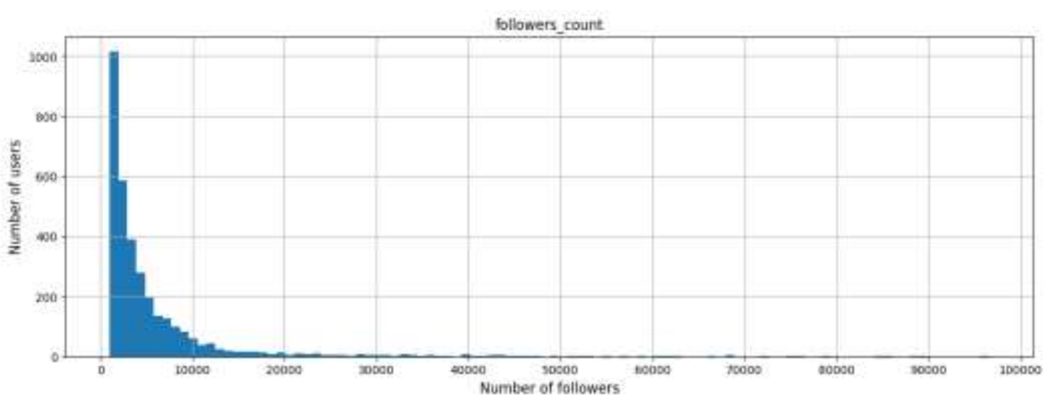
Exploring the dataset:

1.

```
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   uid              3368 non-null   int64
 1   name             3368 non-null   object
 2   friends_count    3368 non-null   int64
 3   followers_count  3368 non-null   int64
 4   listed_count     3368 non-null   int64
 5   statuses_count   3368 non-null   int64
 6   pff              3368 non-null   float64
 7   pfr              3368 non-null   float64
 8   gcf              3368 non-null   float64
 9   gcr              3368 non-null   float64
 10  description      3330 non-null   object
 11  tweets           3368 non-null   object
 12  total_fake       3368 non-null   float64
 13  total_real       3368 non-null   float64
 14  net_trust        3368 non-null   float64
 15  total_news       3368 non-null   float64
 16  fake_prob        3368 non-null   float64
 17  net_trust_norm   3368 non-null   float64
 18  fake             3368 non-null   float64
dtypes: float64(11), int64(5), object(3)
```

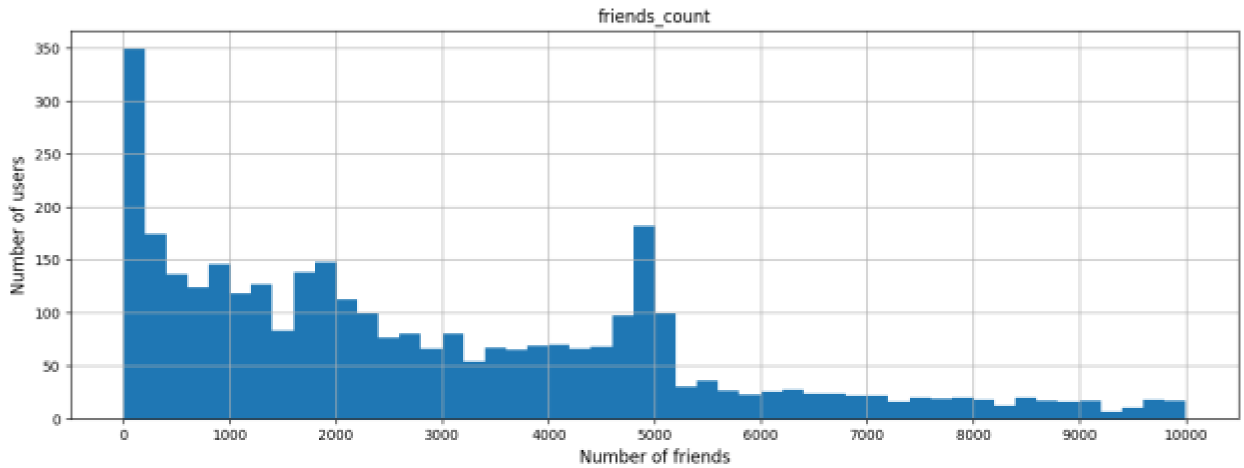
These are the columns in the dataset.

2.



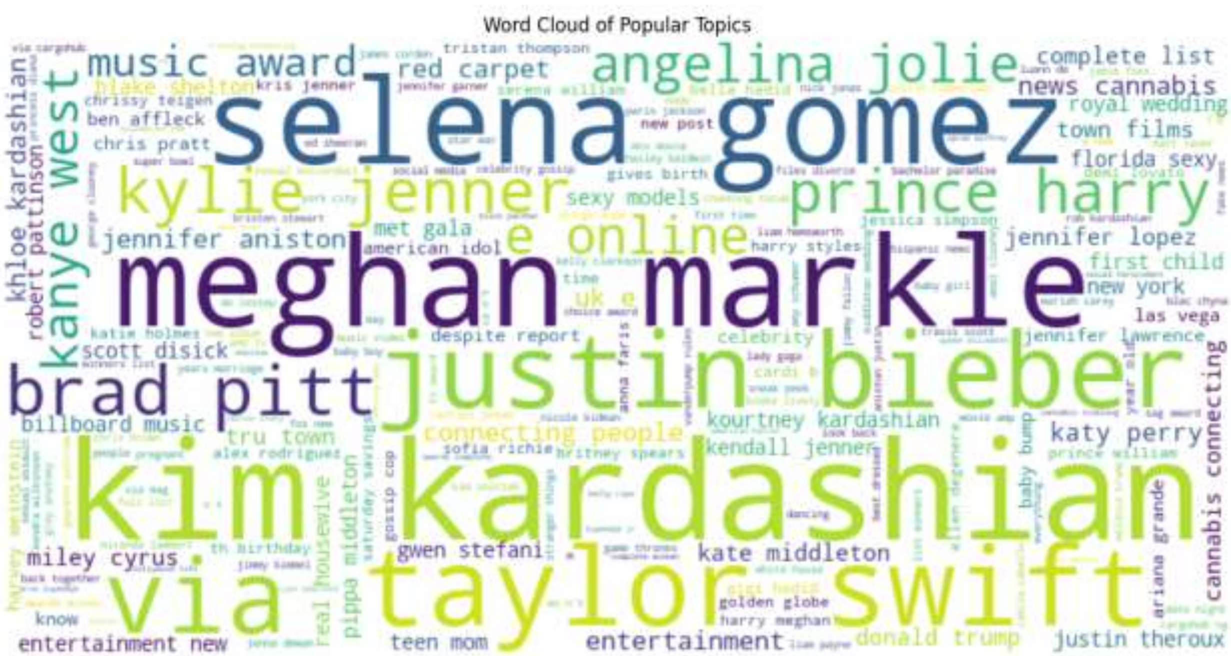
Like the previous dataset, we also find the number of followers.

3.



Graph for number of friends, and many more such graphs.

4.



A wordcloud of most used words in the tweets.

Engagement metrics:

1.

```
Top 5 accounts with the highest number of friends:
                name  friends_count
227    Texas Insider           9997
499     Dread Pirate           9994
433   ProudNavyMom56           9988
479  Ricky Roberts ❌           9987
360     🏳️‍🌈 TLeonard          9986

Top 5 accounts with the highest number of followers:
                   name  followers_count
0          DRUDGE REPORT          1378378
1           NYT Business           786607
2          Jeffrey Levin           610302
3    History Lovers Club           547308
4   GLAMOUR South Africa           500713
```

We find out the accounts with the most followers and the most friends.

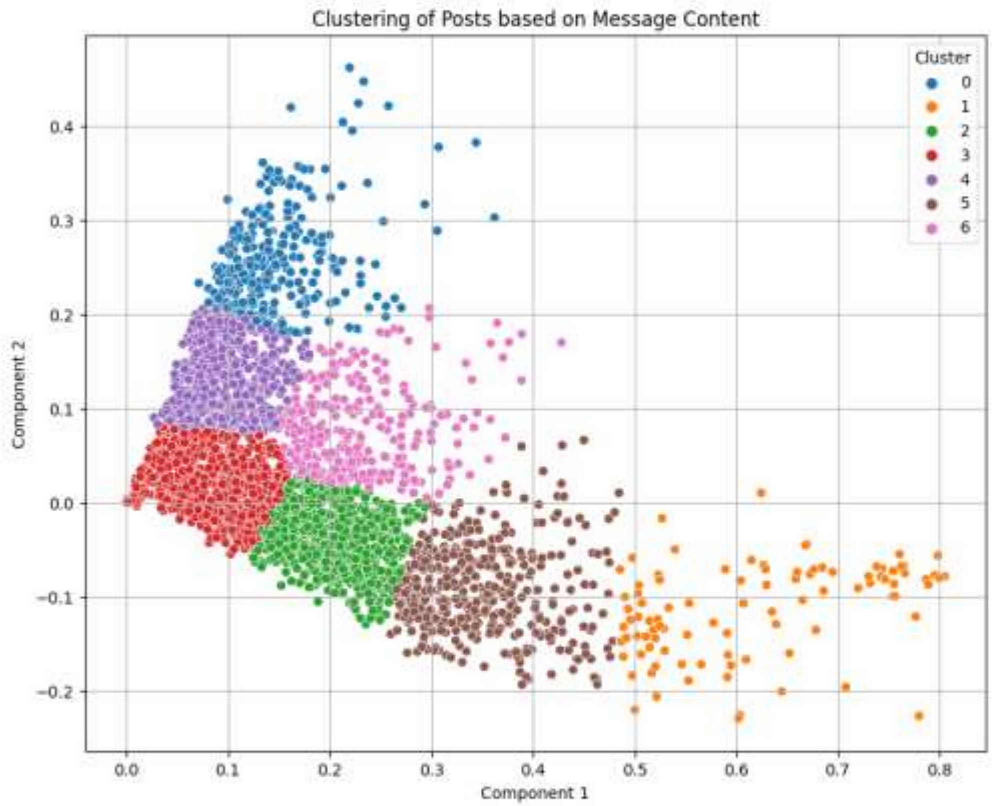
2.

```
Fake probability for top 5 accounts with highest followers:
DRUDGE REPORT: 0.375
NYT Business: 0.36363636363636
Jeffrey Levin: 0.3289473684210526
History Lovers Club: 0.9411764705882352
GLAMOUR South Africa: 0.5833333333333334
```

In this dataset, the values have % of fake probability more than 0.5 are classified fake, and those below 0.5 are classified as real.

We can see that "History Lovers Club" despite having 4th most followers have 94% fake news and also "GLAMOUR South Africa" has 58% fake news.

3.



Like for the dataset 1, we convert the text into numerical vectors and then cluster the data according to the similarity.