nltk: natural language toolkit

In [1]:
```
# !pip install nltk
```

In [2]:
```
import nltk
```

Important libraries

In [3]:
```
#for tokenization
from nltk import word_tokenize, sent_tokenize, WhitespaceTokenizer
#for lemmitization
from nltk import WordNetLemmatizer, LancasterStemmer
#for handling stopwords
from nltk.corpus import stopwords
#frequency distribution
from nltk.probability import FreqDist
#Handling punctuations
from string import punctuation
#for contraction mapping
import contractions
#for wordcloud
from wordcloud import WordCloud
```

In [5]:
```
Text = """
Natural language processing (NLP) is a field of artificial intelligence (AI) that focuses on the int
computers and humans through natural language. The ultimate objective of NLP is to enable computers
interpret, and generate human language in a way that is both meaningful and useful.NLP techniques ar
text, extract relevant information, and derive insights from unstructured data.
"""
```

In [6]:
```
#word tokenization
word_token=word_tokenize(Text)
# word_token
```

In [7]:
```
#sentence tokenization
sent_token=sent_tokenize(Text)
sent_token
```

Out[7]:
```
['\nNatural language processing (NLP) is a field of artificial intelligence (AI) that focuses on the in
teraction between \ncomputers and humans through natural language.',
 'The ultimate objective of NLP is to enable computers to understand, \ninterpret, and generate human l
anguage in a way that is both meaningful and useful.NLP techniques are used to analyze \ntext, extract
relevant information, and derive insights from unstructured data.']
```

In [8]:
```
#to tokenize on the basis of whitespace
whitespace_token=WhitespaceTokenizer().tokenize(Text)
whitespace_token
```

Out[8]:
```
['Natural',
 'language',
 'processing',
 '(NLP)',
 'is',
 'a',
 'field',
 'of',
 'artificial',
 'intelligence',
 '(AI)',
 'that',
 'focuses',
 'on',
 'the',
 'interaction',
 'between',
 'computers',
 'and',
 'humans',
 'through',
 'natural',
 'language.',
 'The',
 'ultimate',
 'objective',
 'of',
 'NLP',
 'is',
 'to',
 'enable',
 'computers',
 'to',
 'understand,',
 'interpret,',
 'and',
 'generate',
 'human',
 'language',
 'in',
 'a',
 'way',
 'that',
 'is',
 'both',
 'meaningful',
 'and',
 'useful.NLP',
 'techniques',
 'are',
 'used',
 'to',
 'analyze',
 'text,',
 'extract',
 'relevant',
 'information,',
 'and',
 'derive',
 'insights',
 'from',
 'unstructured',
 'data.']
```

In [9]:

```
#formating text to same case
lower_text=[x.lower() for x in word_token]
lower_text
```

Out[9]:
```
['natural',
 'language',
 'processing',
 '(',
 'nlp',
 ')',
 'is',
 'a',
 'field',
 'of',
 'artificial',
 'intelligence',
 '(',
 'ai',
 ')',
 'that',
 'focuses',
 'on',
 'the',
 'interaction',
 'between',
 'computers',
 'and',
 'humans',
 'through',
 'natural',
 'language',
 '.',
 'the',
 'ultimate',
 'objective',
 'of',
 'nlp',
 'is',
 'to',
 'enable',
 'computers',
 'to',
 'understand',
 ',',
 'interpret',
 ',',
 'and',
 'generate',
 'human',
 'language',
 'in',
 'a',
 'way',
 'that',
 'is',
 'both',
 'meaningful',
 'and',
 'useful.nlp',
 'techniques',
 'are',
 'used',
 'to',
 'analyze',
 'text',
 ',',
 'extract',
 'relevant',
 'information',
 ',',
 'and',
 'derive',
 'insights',
 'from',
 'unstructured',
```

```
 'data',
 '.']
```

In [10]:

```
#Handling Punctuations
punct_rem=punctuation
print(punct_rem)
punct_rem1=[word for word in lower_text if word not in punctuation ]
punct_rem1
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Out[10]:
```
['natural',
 'language',
 'processing',
 'nlp',
 'is',
 'a',
 'field',
 'of',
 'artificial',
 'intelligence',
 'ai',
 'that',
 'focuses',
 'on',
 'the',
 'interaction',
 'between',
 'computers',
 'and',
 'humans',
 'through',
 'natural',
 'language',
 'the',
 'ultimate',
 'objective',
 'of',
 'nlp',
 'is',
 'to',
 'enable',
 'computers',
 'to',
 'understand',
 'interpret',
 'and',
 'generate',
 'human',
 'language',
 'in',
 'a',
 'way',
 'that',
 'is',
 'both',
 'meaningful',
 'and',
 'useful.nlp',
 'techniques',
 'are',
 'used',
 'to',
 'analyze',
 'text',
 'extract',
 'relevant',
 'information',
 'and',
 'derive',
 'insights',
 'from',
 'unstructured',
 'data']
```

In [11]:
```
#Contraction mapping
text1= "We can't like to attend lecture"
contract_text=contractions.fix(text1)
contract_text
```

Out[11]:
```
'We cannot like to attend lecture'
```

In [12]:
```
stopwords_list=stopwords.words("English")
print(stopwords_list)
stopwords_rem1=[word for word in punct_rem1 if word not in stopwords_list ]
stopwords_rem1
```

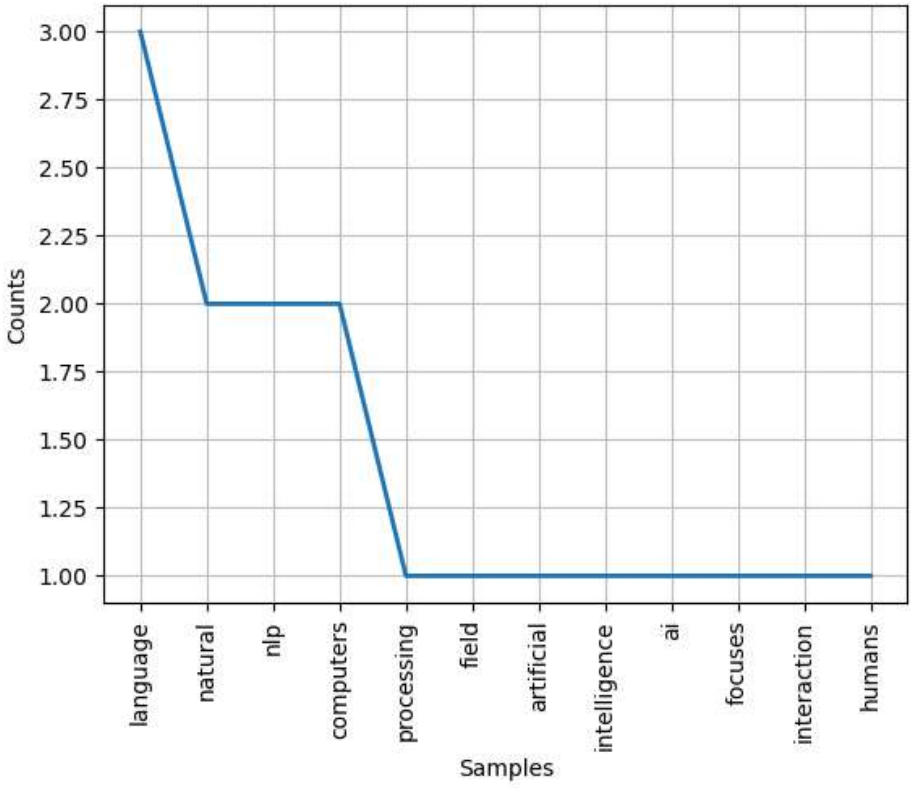
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "yo
u'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her',
'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'wha
t', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'wer
e', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'th
e', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'abou
t', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'fro
m', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'her
e', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'othe
r', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't',
'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've',
'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "had
n't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "must
n't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "were
n't", 'won', "won't", 'wouldn', "wouldn't"]
```

Out[12]:
```
['natural',
 'language',
 'processing',
 'nlp',
 'field',
 'artificial',
 'intelligence',
 'ai',
 'focuses',
 'interaction',
 'computers',
 'humans',
 'natural',
 'language',
 'ultimate',
 'objective',
 'nlp',
 'enable',
 'computers',
 'understand',
 'interpret',
 'generate',
 'human',
 'language',
 'way',
 'meaningful',
 'useful.nlp',
 'techniques',
 'used',
 'analyze',
 'text',
 'extract',
 'relevant',
 'information',
 'derive',
 'insights',
 'unstructured',
 'data']
```

In [13]:
```
frequency_distribution=FreqDist(stopwords_rem1)
print(frequency_distribution)

# To find most common words using Frequency Distribution, add the following lines in above code :
print(frequency_distribution.most_common())
```

```
<FreqDist with 33 samples and 38 outcomes>
[('language', 3), ('natural', 2), ('nlp', 2), ('computers', 2), ('processing', 1), ('field', 1), ('artificial', 1), ('intelligence', 1), ('ai', 1), ('focuses', 1), ('interaction', 1), ('humans', 1), ('ultimate', 1), ('objective', 1), ('enable', 1), ('understand', 1), ('interpret', 1), ('generate', 1), ('human', 1), ('way', 1), ('meaningful', 1), ('useful.nlp', 1), ('techniques', 1), ('used', 1), ('analyze', 1), ('text', 1), ('extract', 1), ('relevant', 1), ('information', 1), ('derive', 1), ('insights', 1), ('unstructured', 1), ('data', 1)]
```

In [14]:
```
import matplotlib.pyplot as plt
frequency_distribution.plot(12,cumulative=False)
plt.show()
```



In [15]:
```
text_lem=WordNetLemmatizer()
text_stem=LancasterStemmer()
for word in stopwords_rem1:
    text_lem1=text_lem.lemmatize(word)
    text_stem1=text_stem.stem(word)
    print(f"original text: {word}")
    print(f"lemmitize text: {text_lem1}")
    print(f"steming text: {text_stem1}")
    print("."*20)
```

```
original text: information
lemmitize text: information
steming text: inform
....................
original text: derive
lemmitize text: derive
steming text: der
....................
original text: insights
lemmitize text: insight
steming text: insight
....................
original text: unstructured
lemmitize text: unstructured
steming text: unstruct
....................
original text: data
lemmitize text: data
steming text: dat
....................
```

In [16]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
obj=SentimentIntensityAnalyzer()
Text="I like tea"
sent_text=obj.polarity_scores(Text)
sent_text
```

Out[16]: {'neg': 0.0, 'neu': 0.286, 'pos': 0.714, 'compound': 0.3612}

In [17]:
```
Text="very bad!"
sent_text=obj.polarity_scores(Text)
sent_text
```

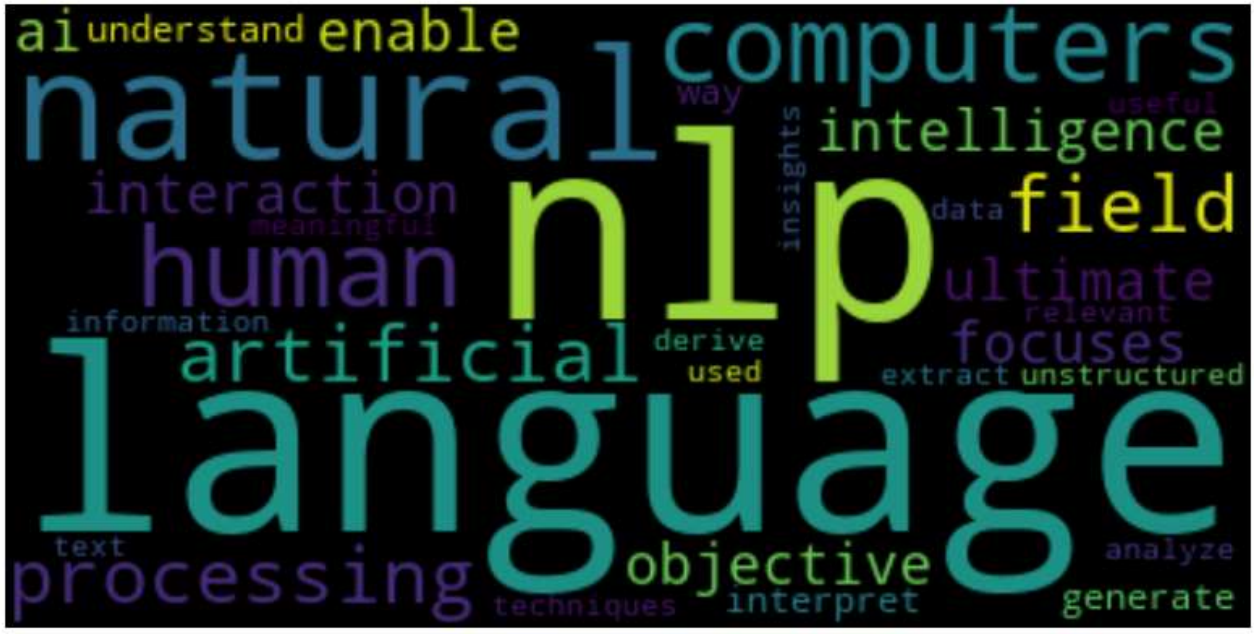
Out[17]: {'neg': 0.803, 'neu': 0.197, 'pos': 0.0, 'compound': -0.623}

In [18]:
```
from wordcloud import WordCloud
def plot_wordcloud(stopwords_rem1):
    text = ' '.join(stopwords_rem1)
    wordcloud = WordCloud(width=500, height=300, background_color='white').generate(text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
#     plt.axis('off')
    plt.show()
```

wordcloud

In [19]:
```
text = ' '.join(stopwords_rem1)   #processed text(after removing stopwords,punctuations i.e. noise j
print(text)
wordcloud = WordCloud().generate(text)
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
natural language processing nlp field artificial intelligence ai focuses interaction computers humans n
atural language ultimate objective nlp enable computers understand interpret generate human language wa
y meaningful useful.nlp techniques used analyze text extract relevant information derive insights unstr
uctured data
```



for text summarization

In [20]:
```
# Tokenize the text into sentences
sentences = sent_tokenize(text)

# Tokenize each sentence into words and remove stopwords
stop_words = set(stopwords.words('english'))
word_frequencies = {}
for sentence in sentences:
    word_tokens = word_tokenize(sentence)
    filtered_words = [word.lower() for word in word_tokens if word.isalnum() and word.lower() not i
    for word in filtered_words:
        if word in word_frequencies:
            word_frequencies[word] += 1
        else:
            word_frequencies[word] = 1
word_frequencies
```

Out[20]:
```
{'natural': 2,
 'language': 3,
 'processing': 1,
 'nlp': 2,
 'field': 1,
 'artificial': 1,
 'intelligence': 1,
 'ai': 1,
 'focuses': 1,
 'interaction': 1,
 'computers': 2,
 'humans': 1,
 'ultimate': 1,
 'objective': 1,
 'enable': 1,
 'understand': 1,
 'interpret': 1,
 'generate': 1,
 'human': 1,
 'way': 1,
 'meaningful': 1,
 'techniques': 1,
 'used': 1,
 'analyze': 1,
 'text': 1,
 'extract': 1,
 'relevant': 1,
 'information': 1,
 'derive': 1,
 'insights': 1,
 'unstructured': 1,
 'data': 1}
```

In [27]:

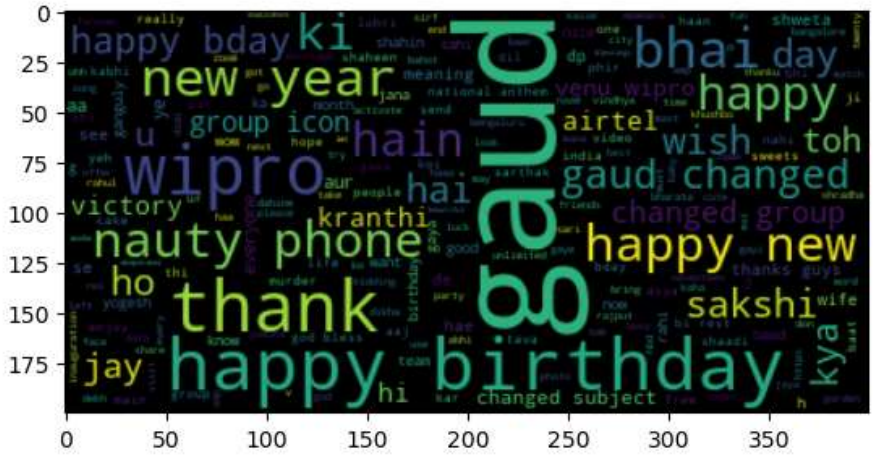
```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

# Sample text
text = """
Natural language processing (NLP) is a field of artificial intelligence (AI) that focuses on the int
computers and humans through natural language.The ultimate objective of NLP is to enable computers t
interpret, and generate human language in a way that is both meaningful and useful.NLP techniques ar
text, extract relevant information, and derive insights from unstructured data.
"""

# Tokenize the text into sentences
sentences = sent_tokenize(text)

# Tokenize each sentence into words and remove stopwords
stop_words = set(stopwords.words('english'))
word_frequencies = {}
for sentence in sentences:
    word_tokens = word_tokenize(sentence)
    filtered_words = [word.lower() for word in word_tokens if word.isalnum() and word.lower() not in
    for word in filtered_words:
        if word in word_frequencies:
            word_frequencies[word] += 1
        else:
            word_frequencies[word] = 1

# Calculate sentence scores based on word frequencies
sentence_scores = {}
for sentence in sentences:
    for word in word_tokenize(sentence.lower()):
        if word in word_frequencies:
            if len(sentence.split(' ')) < 30:
                if sentence in sentence_scores:
                    sentence_scores[sentence] += word_frequencies[word]
                else:
                    sentence_scores[sentence] = word_frequencies[word]

# Get top N sentences with highest scores
# Get top N sentences with highest scores
num_sentences = 2
summary_sentences = sorted(sentence_scores, key=sentence_scores.get, reverse=True)[:num_sentences]
summary = ' '.join(summary_sentences)
print("Summary: ", summary)

print("\noriginal text legth:",len(text))
print("summarize text length:", len(summary))
```

```
Summary:

original text legth: 433
summarize text length: 0
```

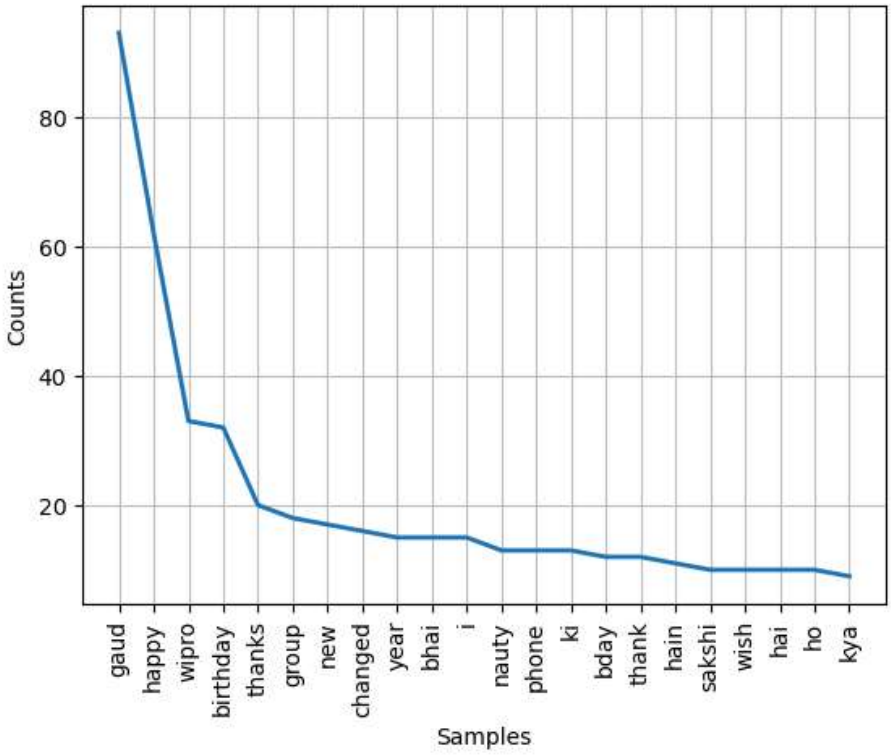
Set A: 4 (whatsup chat analysis)

In [30]:

```
# reading text file
import re
import nltk
with open(r"C:\Users\hp\Desktop\TYCS\DATA ANALYTICS\_chat.txt", encoding="utf8") as chat:
    chat_text1 = chat.read()

### Cleaning and Preparing Data

# removing punct
from nltk.tokenize import word_tokenize
from string import punctuation

#removing all elements other than alphabets
chat_text=re.sub('[^a-zA-Z]', ' ', chat_text1)
# chat_text

#word tokenization
words = word_tokenize(chat_text)

#removing all stopwords
from nltk.corpus import stopwords,re
sw = set(stopwords.words("english"))
filterd_words = [w.lower() for w in words if w not in sw]

# removing names and other anomalies like am,pm
unwanted =  {'vishnu','shahain','raghavan','preeti','pankaj','sinha','sharma',
             'sahil','phatania','neha','mukti','omitted','image','kushbhu','am','pm'}
filterd_words = [ele for ele in filterd_words if ele not in unwanted]
# filterd_words

#wordcloud
import matplotlib.pyplot as plt
from wordcloud import WordCloud
text = ' '.join(filterd_words)
processed_text=re.sub('[^a-zA-Z]', ' ', text)
wordcloud = WordCloud().generate(processed_text)
# plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.show()

# showing frequency distribution
from nltk import FreqDist
frequency_distribution= FreqDist(filterd_words)
frequency_distribution.most_common()

#sentiment analysis
from nltk.sentiment.vader import SentimentIntensityAnalyzer
vader_analyzer=SentimentIntensityAnalyzer()
print(vader_analyzer.polarity_scores( chat_text))

import matplotlib.pyplot as plt
frequency_distribution.plot(22,cumulative=False)
plt.show()
```




```
{'neg': 0.02, 'neu': 0.762, 'pos': 0.218, 'compound': 1.0}
```



Set B

Q1

In [61]:
```
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from collections import Counter
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Read the dataset
df = pd.read_csv(r"C:\Users\hp\Desktop\TYCS\DATA ANALYTICS\instagram_global_top_1000.csv")
df
```

Out[61]:

	Country	Rank	Account	Title	Link	Category	Followers
0	All	1	cristiano	Cristiano Ronaldo	https://www.instagram.com/cristiano/	Sports with a ball	400100000.0
1	All	2	kyliejenner	Kylie ♡	https://www.instagram.com/kyliejenner/	Fashion|Modeling|Beauty	308800000.0
2	All	3	leomessi	Leo Messi	https://www.instagram.com/leomessi/	Sports with a ball|Family	306300000.0
3	All	4	kendalljenner	Kendall	https://www.instagram.com/kendalljenner/	Modeling|Fashion	217800000.0
4	All	5	selenagomez	Selena Gomez	https://www.instagram.com/selenagomez/	Music|Lifestyle	295800000.0
...	...	...	...	...	...	...	...
995	All	996	senoritasaeva	Dina Saeva	https://www.instagram.com/senoritasaeva/	Lifestyle|Music|Modeling	7700000.0
996	All	997	manuelneuer	Manuel Neuer	https://www.instagram.com/manuelneuer/	Sports with a ball	11500000.0
997	All	998	sahilkhan	India's Youth & Fitness IC⭕N®	https://www.instagram.com/sahilkhan/	Fitness|Gym	10100000.0
998	All	999	mohanshakti	Shakti Mohan	https://www.instagram.com/mohanshakti/	Art|Artists|Cinema|Actors/actresses	13700000.0
999	All	1000	eduincaz	Eduin Caz	https://www.instagram.com/eduincaz/	Lifestyle	6200000.0

1000 rows × 11 columns

In [62]:
```
# i. Find top 5 Instagram influencers from India
top_5_influencers_india = df[df['Audience Country'] == 'India'].nlargest(5, 'Followers')
print("Top 5 Instagram influencers from India:")
sorted_top_5_influencers_india = top_5_influencers_india.sort_values(by='Followers', ascending=Fals

print(sorted_top_5_influencers_india[['Account', 'Followers']])
```

```
Top 5 Instagram influencers from India:
         Account    Followers
28     instagram  469600000.0
0      cristiano  400100000.0
27       therock  295800000.0
19  justinbieber  219800000.0
14   virat.kohli  182600000.0
```

In [63]:
```
# ii. Find Instagram account with least number of Followers
least_Followers_account = df.nsmallest(1, 'Followers')
print("\nInstagram account with least number of Followers:")
print(least_Followers_account[['Account', 'Followers']])
```

```
Instagram account with least number of Followers:
     Account  Followers
747  yooncy1  2800000.0
```

In [64]:
```
# Tokenize each string in the 'Category' column
categories = df['Category'].astype(str)
```

In [65]:
```
df["Category"]
```

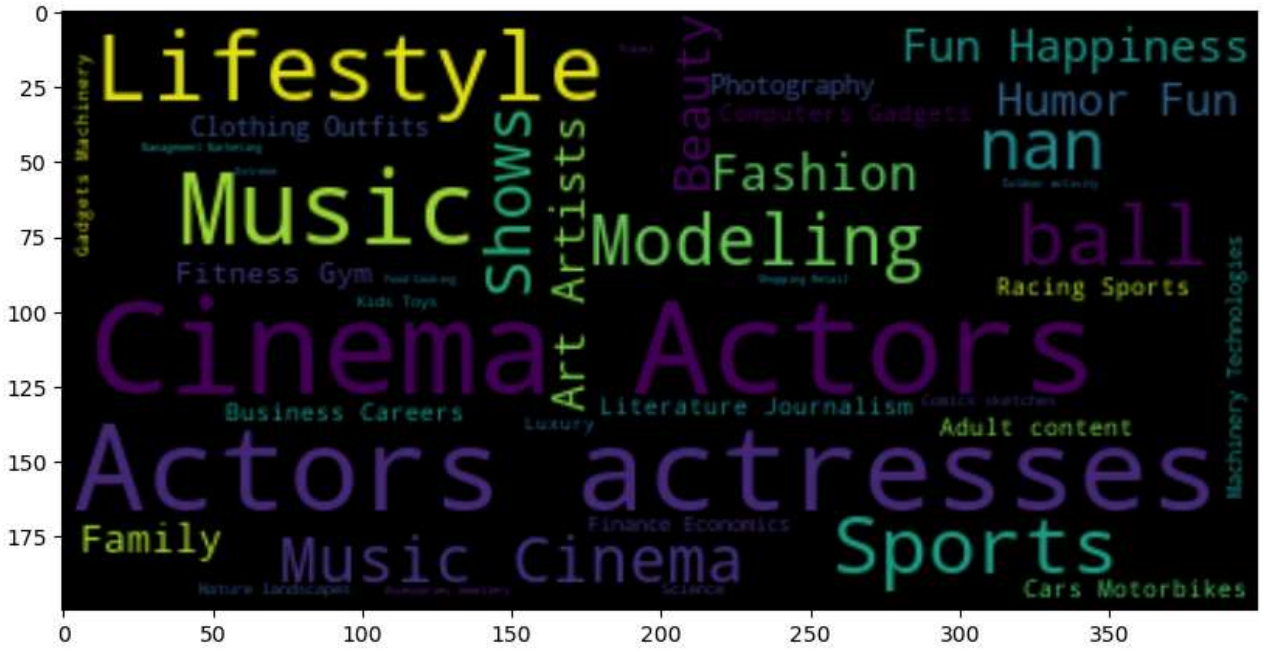
Out[65]:
```
0                       Sports with a ball
1                  Fashion|Modeling|Beauty
2                Sports with a ball|Family
3                          Modeling|Fashion
4                           Music|Lifestyle
                        ...
995                Lifestyle|Music|Modeling
996                       Sports with a ball
997                              Fitness|Gym
998    Art|Artists|Cinema|Actors/actresses
999                               Lifestyle
Name: Category, Length: 1000, dtype: object
```

In [66]:
```
# Tokenize each string in the 'Category' column
categories = df['Category'].astype(str)
# Assuming 'categories' is the column you want to tokenize
# Tokenize each string in the 'categories' column
word_tokens = categories.apply(lambda x: word_tokenize(str(x)))
word_tokens
```

Out[66]:
```
0                    [Sports, with, a, ball]
1                  [Fashion|Modeling|Beauty]
2              [Sports, with, a, ball|Family]
3                          [Modeling|Fashion]
4                           [Music|Lifestyle]
                        ...
995                [Lifestyle|Music|Modeling]
996                   [Sports, with, a, ball]
997                              [Fitness|Gym]
998    [Art|Artists|Cinema|Actors/actresses]
999                               [Lifestyle]
Name: Category, Length: 1000, dtype: object
```

In [68]:
```
stop_words = set(stopwords.words('english'))

# Remove stopwords from word tokens
filtered_text1 = [[word for word in tokens if word not in stop_words] for tokens in word_tokens]

filtered_text=text = ' '.join(categories)
# # Generate word cloud
wordcloud = WordCloud().generate(filtered_text)
# Plot the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
# plt.axis('off')
# plt.title('Wordcloud for Categories')
# plt.show()
```

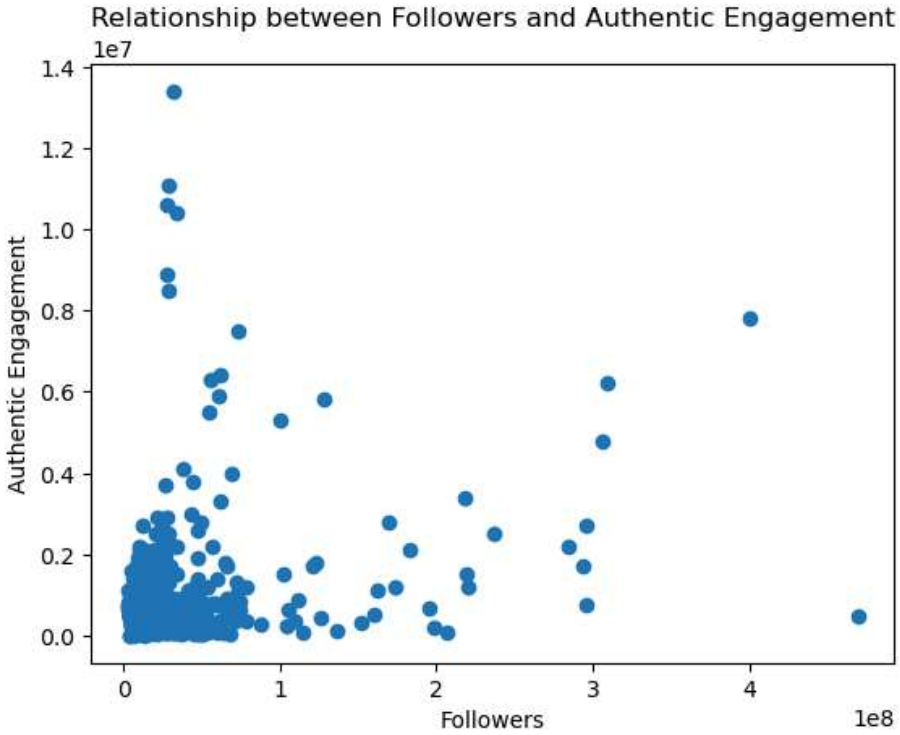
Out[68]: <matplotlib.figure.AxesImage at 0x1a19e79cca0>



In [69]:
```
# iv. Group Instagram accounts category wise
grouped_categories = df.groupby('Category').size().reset_index(name='counts')
print("\nInstagram accounts grouped by category:")
print(grouped_categories)
```

```
Instagram accounts grouped by category:
                                          Category  counts
0    Accessories|Jewellery|Clothing|Outfits|Fashion       1
1                                     Adult content       7
2                                           Animals       1
3                                       Art|Artists       3
4                                 Art|Artists|Beauty       1
..                                               ...     ...
160                       Sports with a ball|Family       2
161                    Sports with a ball|Lifestyle       4
162          Sports with a ball|Lifestyle|Kids|Toys       1
163        Sports with a ball|Literature|Journalism       1
164                         Sports with a ball|Shows       1

[165 rows x 2 columns]
```

In [70]:
```
# v. Visualize relationship between Followers and Authentic engagement
plt.scatter(df['Followers'], df['Authentic engagement'])
plt.title('Relationship between Followers and Authentic Engagement')
plt.xlabel('Followers')
plt.ylabel('Authentic Engagement')
plt.show()

```



In [71]:
```
correlation = df['Followers'].corr(df['Authentic engagement'])
correlation
```

Out[71]: 0.2870667818696664

Q2

In [53]:
```
import pandas as pd
from nltk.tokenize import word_tokenize
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
# nltk.download('vader_lexicon')

# Load the dataset
df = pd.read_csv(r"C:\Users\hp\Desktop\TYCS\DATA ANALYTICS\covid_2021_1.csv")
print(df.head())
```

```
                                query  \
0  coronavirus|covid%7C19|pandemic|vaccine
1  coronavirus|covid%7C19|pandemic|vaccine
2  coronavirus|covid%7C19|pandemic|vaccine
3  coronavirus|covid%7C19|pandemic|vaccine
4  coronavirus|covid%7C19|pandemic|vaccine

                                     url  \
0  https://www.youtube.com/watch?v=LyRoIKjmk9o (https://www.youtube.com/watch?v=LyRoIKjmk9o)
1  https://www.youtube.com/watch?v=LyRoIKjmk9o (https://www.youtube.com/watch?v=LyRoIKjmk9o)
2  https://www.youtube.com/watch?v=LyRoIKjmk9o (https://www.youtube.com/watch?v=LyRoIKjmk9o)
3  https://www.youtube.com/watch?v=LyRoIKjmk9o (https://www.youtube.com/watch?v=LyRoIKjmk9o)
4  https://www.youtube.com/watch?v=dwMWM1F7NMI (https://www.youtube.com/watch?v=dwMWM1F7NMI)

                                          title           upload_date  \
0  Which Coronavirus Vaccine Is More Effective? C...  2021-01-01T09:04:16Z
1  Which Coronavirus Vaccine Is More Effective? C...  2021-01-01T09:04:16Z
2  Which Coronavirus Vaccine Is More Effective? C...  2021-01-01T09:04:16Z
3  Which Coronavirus Vaccine Is More Effective? C...  2021-01-01T09:04:16Z
4  Growing Number Of Americans Going Hungry Durin...  2021-01-01T11:59:49Z

       channel  views  likes  dislikes  comment_count  \
0  India Today   8276     58         7              5
1  India Today   8276     58         7              5
2  India Today   8276     58         7              5
3  India Today   8276     58         7              5
4    NBC News  74546    874        62            601

                                      comment_text  \
0                                      OMG 😍💋 💝💖❤️
1                I love my Indian Mumbai me welcome
2                        What about indian vaccine?
3                           2:32 I like that😍💋 💝💖❤️
4  And all the rich companies that have profited ...

               comment_author          comment_date  comment_likes  \
0                Brendan Eric  2021-01-01T09:23:59Z              0
1  Bhallesingh Rathore rathore  2021-01-01T09:06:51Z              1
2      Kudip singh hong kong.  2021-02-03T08:17:14Z              0
3                   Bo Brice  2021-01-01T09:55:34Z              0
4                 Poppy Jalto  2021-01-01T13:17:56Z             93

                       DATE
0  2021-01-01 09:04:16+00:00
1  2021-01-01 09:04:16+00:00
2  2021-01-01 09:04:16+00:00
3  2021-01-01 09:04:16+00:00
4  2021-01-01 11:59:49+00:00
```

In [54]:
```
# i. Data Cleaning
# Drop rows with missing values
df.dropna(inplace=True)
```

In [55]:
```
df['tokenized_comments'] = df['comment_text'].apply(word_tokenize)
df['tokenized_comments']
```

Out[55]:
```
0                                          [OMG, 😍💋, 💝💖❤️]
1                 [I, love, my, Indian, Mumbai, me, welcome]
2                          [What, about, indian, vaccine, ?]
3                              [2:32, I, like, that😍💋, 💝💖❤️]
4         [And, all, the, rich, companies, that, have, p...
                                ...
41583                                        [Thanks, da]
41584                       [👍👍, very, helpful, information]
41585                                                [👍]
41586    [Very, very, much, informative, vlog, .., Sunn...
41587                [Thanking, you, for, this, information]
Name: tokenized_comments, Length: 41588, dtype: object
```

In [56]:
```
# iii. Sentiment Analysis
def analyze_sentiment(comment):
    analyzer = SentimentIntensityAnalyzer()
    sentiment_scores = analyzer.polarity_scores(comment)
    if sentiment_scores['compound'] > 0:
        return 'positive'
    elif sentiment_scores['compound'] < 0:
        return 'negative'
    else:
        return 'neutral'

df['sentiment'] = df['comment_text'].apply(analyze_sentiment)
df['sentiment']
```

Out[56]:
```
0         neutral
1        positive
2         neutral
3        positive
4        positive
           ...
41583    positive
41584    positive
41585     neutral
41586    positive
41587     neutral
Name: sentiment, Length: 41588, dtype: object
```

In [ ]:
```
# Calculate percentage of positive, negative, and neutral comments
sentiment_counts = df['sentiment'].value_counts(normalize=True) * 100

print("Percentage of Comments by Sentiment:")
print(sentiment_counts)

```

Q3

In [73]:
```
import pandas as pd
import matplotlib.pyplot as plt

# i. Read the dataset and perform data cleaning operations
df = pd.read_csv('INvideos.csv')

# Drop duplicates and missing values
df.drop_duplicates(inplace=True)
df.dropna(inplace=True)


```

In [74]:
```
# ii. Find total views, likes, dislikes, and comment count
total_views = df['views'].sum()
total_likes = df['likes'].sum()
total_dislikes = df['dislikes'].sum()
total_comments = df['comment_count'].sum()

print("Total Views:", total_views)
print("Total Likes:", total_likes)
print("Total Dislikes:", total_dislikes)
print("Total Comments:", total_comments)
```

```
Total Views: 32792912293
Total Likes: 845128501
Total Dislikes: 51994375
Total Comments: 83413188
```

In [75]:
```
# iii. Find the least and topmost liked and commented videos

 #row corresponding to the least number of likes
least_liked_video = df.loc[df['likes'].idxmin()]
#row corresponding to the max number of likes
top_liked_video = df.loc[df['likes'].idxmax()]

 #row corresponding to the least number of comments
least_commented_video = df.loc[df['comment_count'].idxmin()]
 #row corresponding to the max comments
top_commented_video = df.loc[df['comment_count'].idxmax()]
```

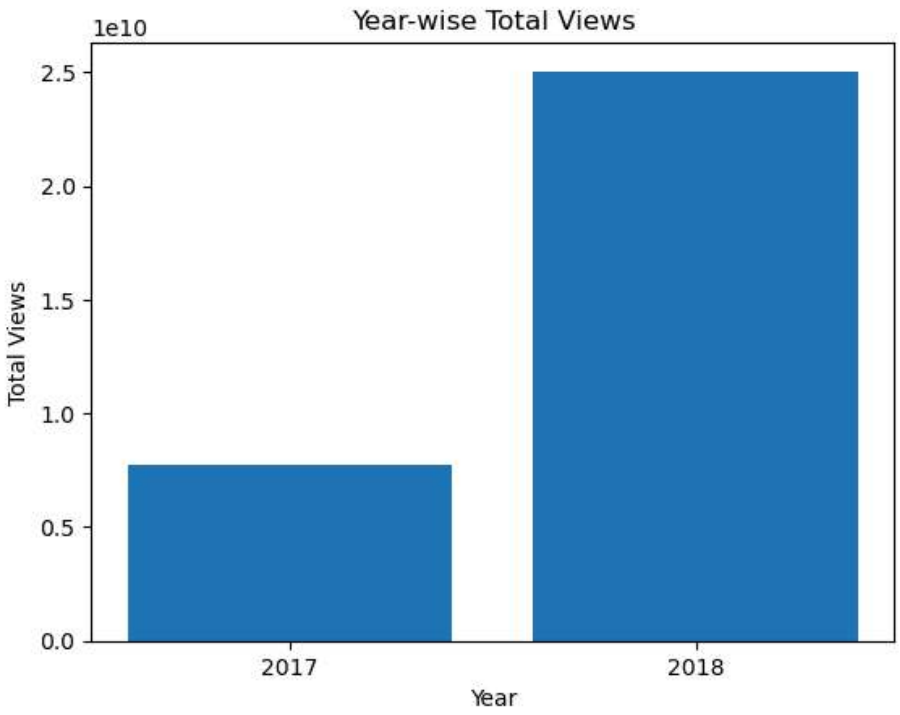
In [76]:
```
print("\nLeast Liked Video:\n",least_liked_video[['title', 'likes']])
print("\nTop Liked Video:\n",top_liked_video[['title', 'likes']])
print("\nLeast Commented Video:\n",least_commented_video[['title', 'comment_count']])
print("\nTop Commented Video:\n",top_commented_video[['title', 'comment_count']])
```

```
Least Liked Video:
 title    Breaking News IT Raid - நடந்தது என்ன? சிக்கியத...
likes                                                    0
Name: 34, dtype: object

Top Liked Video:
 title    YouTube Rewind: The Shape of 2017 | #YouTubeRe...
likes                                              2912710
Name: 5408, dtype: object

Least Commented Video:
 title           पीरियड्स के समय, पेट पर पति करता ऐसा, देखकर दं...
comment_count                                                    0
Name: 1, dtype: object

Top Commented Video:
 title           YouTube Rewind: The Shape of 2017 | #YouTubeRe...
comment_count                                              827755
Name: 4743, dtype: object
```

In [79]:
```
# iv. Perform year-wise statistics for views and plot the analyzed data
df['trending_date'] = pd.to_datetime(df['trending_date'], format='%y.%d.%m')
df['year'] = df['trending_date'].dt.year
yearly_views = df.groupby('year')['views'].sum()
df['trending_date']
```

Out[79]:
```
0       2017-11-14
1       2017-11-14
2       2017-11-14
3       2017-11-14
4       2017-11-14
           ...
37300   2018-06-14
37301   2018-06-14
37302   2018-06-14
37319   2018-06-14
37330   2018-06-14
Name: trending_date, Length: 32562, dtype: datetime64[ns]
```

In [80]:
```
plt.bar(yearly_views.index, yearly_views.values)
plt.xlabel('Year')
plt.ylabel('Total Views')
plt.title('Year-wise Total Views')
plt.xticks(yearly_views.index)
plt.show()
```



In [ ]:
```
# v. Plot the viewers who reacted on videos
reacted_viewers = df[['likes', 'dislikes']]
reacted_viewers.plot(kind='bar', stacked=True)
plt.xlabel('Video Index')
plt.ylabel('Number of Viewers')
plt.title('Viewers who Reacted on Videos')
plt.show()

```

In [ ]:
```

```