PHASE-2

Data Preprocessing

Sentiment Analysis for Marketing:

Date	10 October 2023
TEAM ID	Proj_212173_Team_1
Project Name	Sentiment analysis for Marketing

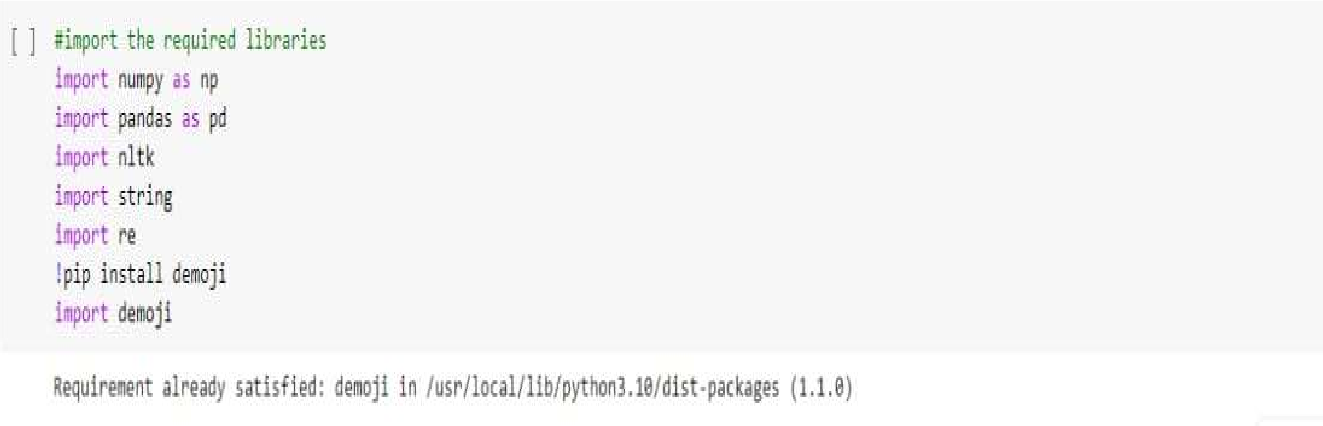
Data preprocessing refers to the cleaning, transforming, and integrating of data in order to make it ready for further analysis.Data preprocessing is a process of preparing the raw data and making it suitable for a further analysis

Program:

Import Package

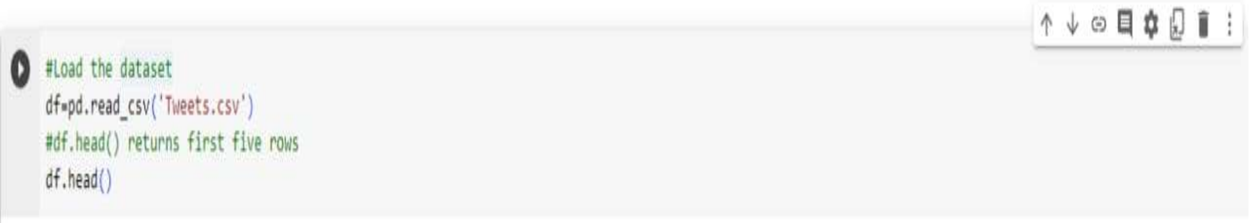
Explanation:

- Numpy-Numpy is an open source python library used for working with arrays.
- Pandas-It is a Powerful data manipulation library in python.
- Matplotlib- Matplotlib is a python library used to create 2D graphs and plots
- Nltk-It is a flexible library for sentiment analysis and NLP.
- String- The string module contains a number of functions to process standard Python strings.
- Re- RegEx(re) can be used to check if a string contains the specified search pattern.
- Demoji- Accurately find or remove emojis from a blob of text.



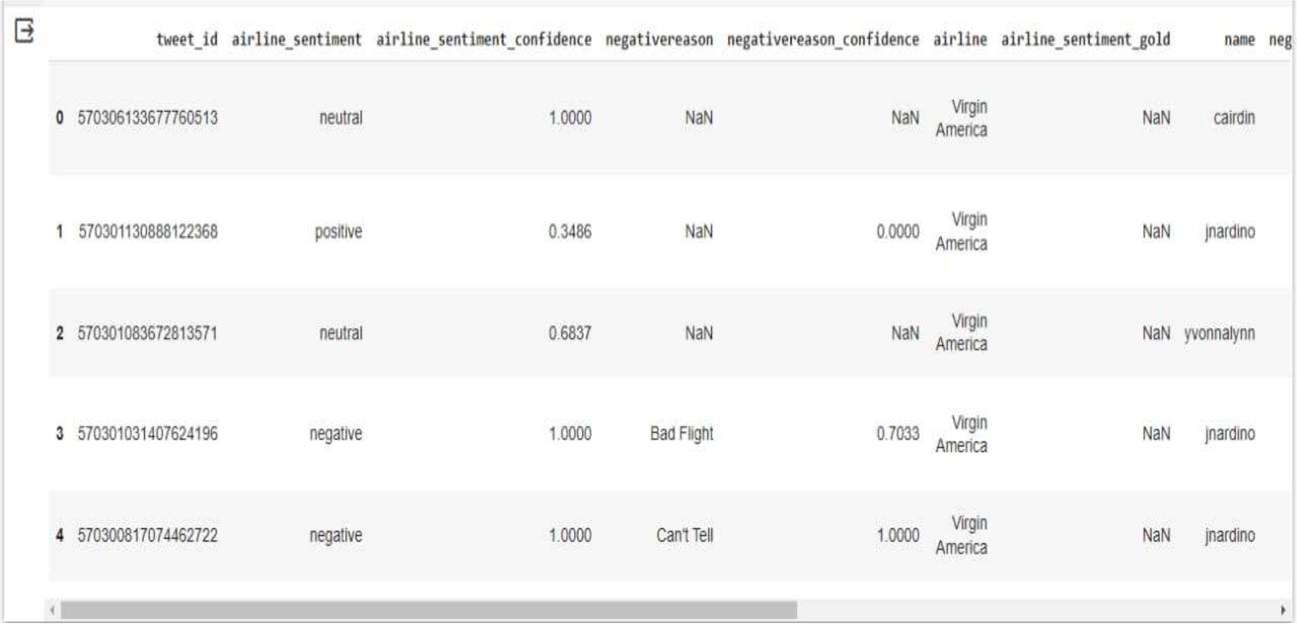
```
[ ] #import the required libraries
    import numpy as np
    import pandas as pd
    import nltk
    import string
    import re
    !pip install demoji
    import demoji

    Requirement already satisfied: demoji in /usr/local/lib/python3.10/dist-packages (1.1.0)
```

Pd.read_csv is used to load the dataset and stores it in a variable name called df



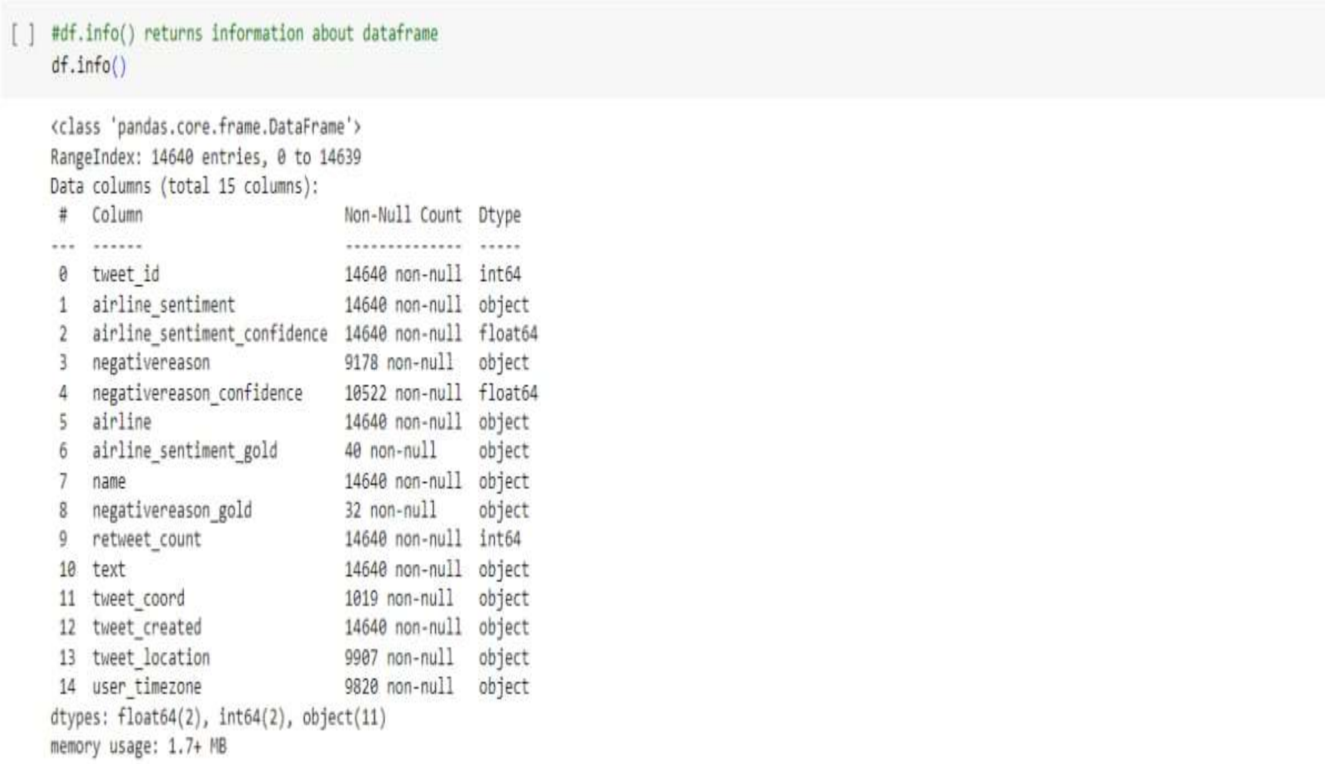
```
#Load the dataset
df=pd.read_csv('Tweets.csv')
#df.head() returns first five rows
df.head()
```

df.head() to print the first 5 values from the dataset



	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	neg
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdin	
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardino	
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonnalynn	
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardino	
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardino	

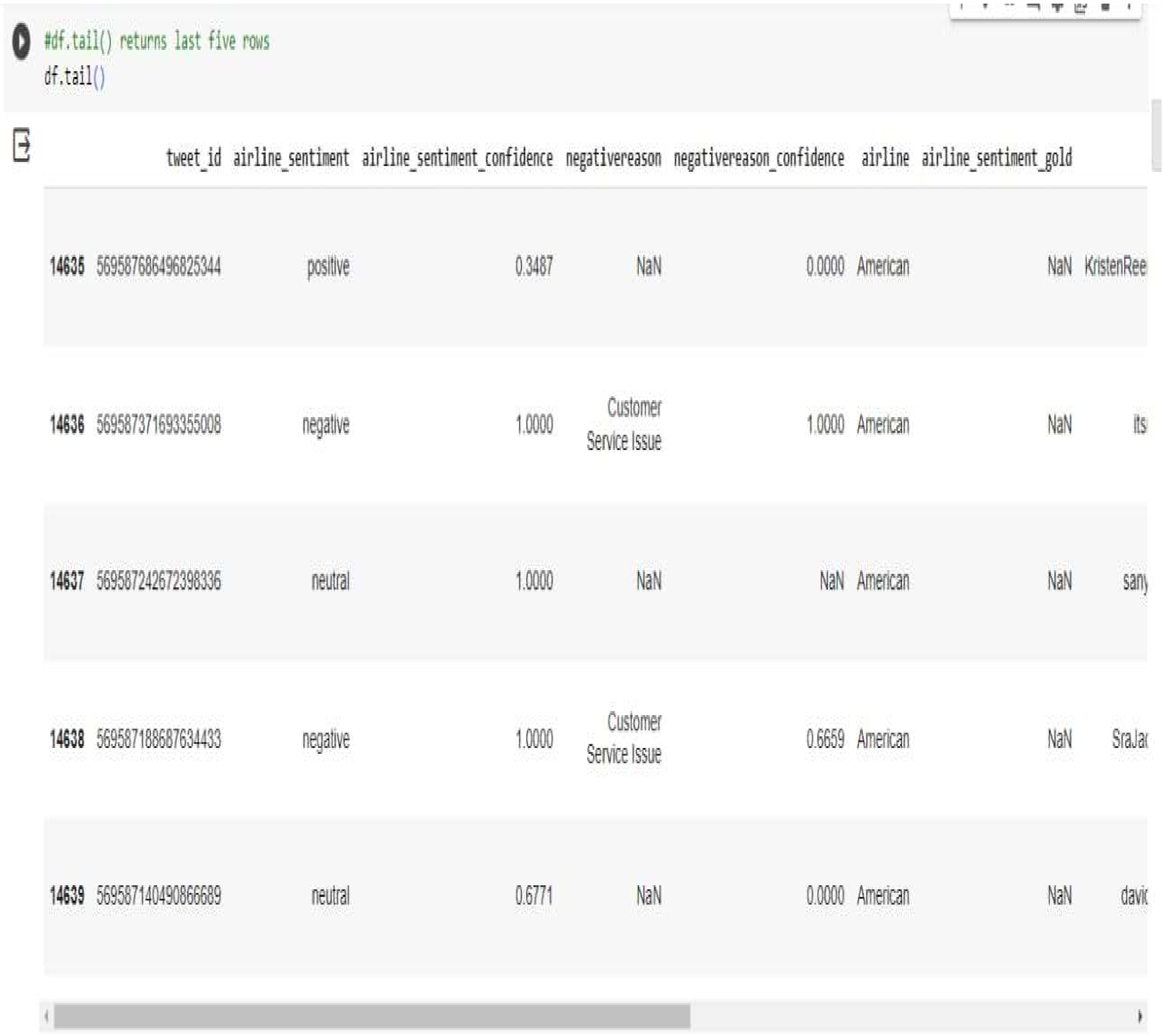
info() returns the information about the dataframe



```
[ ] #df.info() returns information about dataframe
    df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 15 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   tweet_id                      14640 non-null  int64
 1   airline_sentiment             14640 non-null  object
 2   airline_sentiment_confidence  14640 non-null  float64
 3   negativereason                9178 non-null   object
 4   negativereason_confidence     10522 non-null  float64
 5   airline                       14640 non-null  object
 6   airline_sentiment_gold        40 non-null     object
 7   name                          14640 non-null  object
 8   negativereason_gold           32 non-null     object
 9   retweet_count                 14640 non-null  int64
 10  text                          14640 non-null  object
 11  tweet_coord                   1019 non-null   object
 12  tweet_created                 14640 non-null  object
 13  tweet_location                9907 non-null   object
 14  user_timezone                 9820 non-null   object
dtypes: float64(2), int64(2), object(11)
memory usage: 1.7+ MB
```

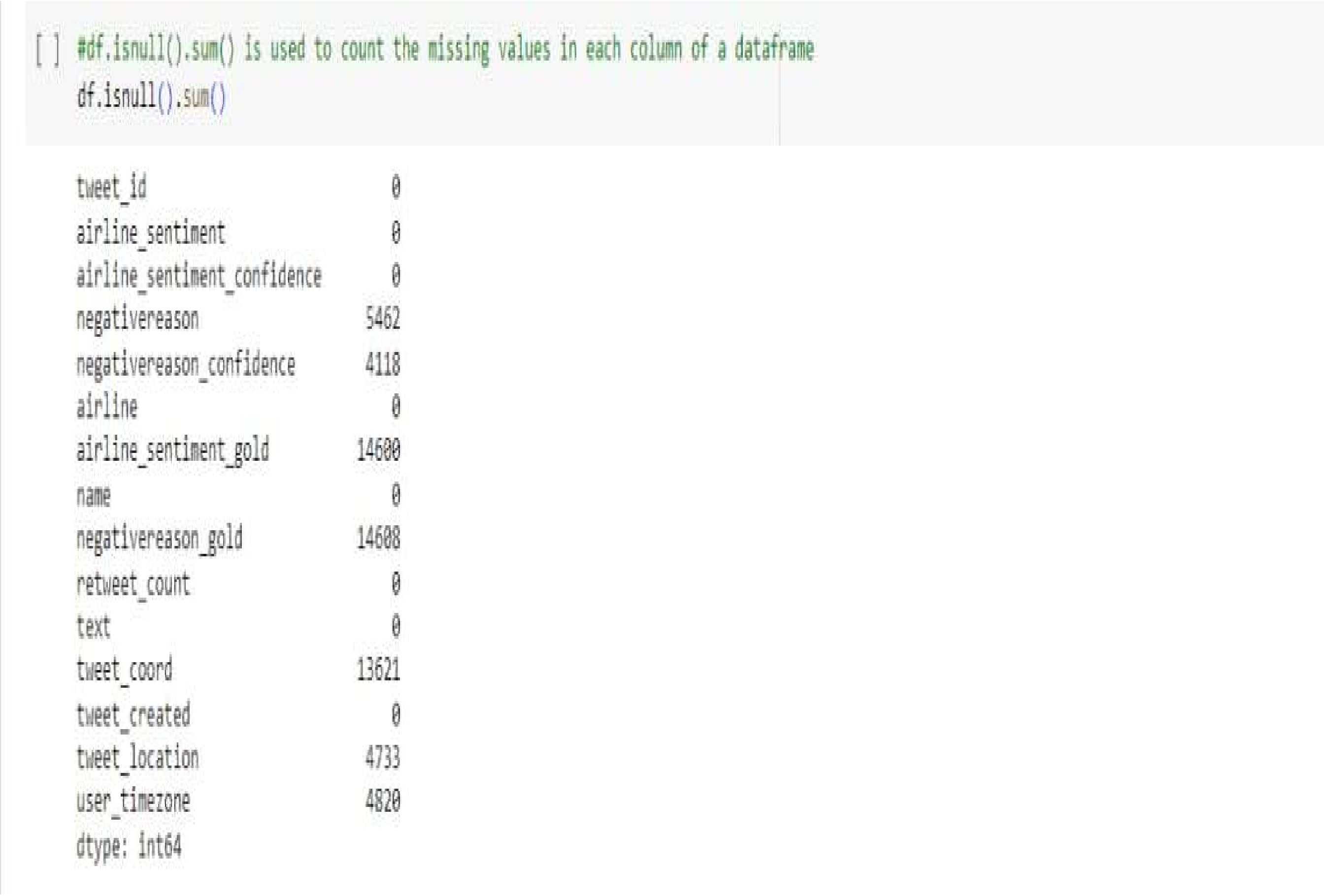
tail() is used to return last five rows



```
#df.tail() returns last five rows
df.tail()
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	
14635	569587686496825344	positive	0.3487	NaN	0.0000	American	NaN	KristenRee
14636	569587371693355008	negative	1.0000	Customer Service Issue	1.0000	American	NaN	its
14637	569587242672398336	neutral	1.0000	NaN	NaN	American	NaN	sany
14638	569587188687634433	negative	1.0000	Customer Service Issue	0.6659	American	NaN	SraJa
14639	569587140490866689	neutral	0.6771	NaN	0.0000	American	NaN	davi

isnull() is to check the values is null or not

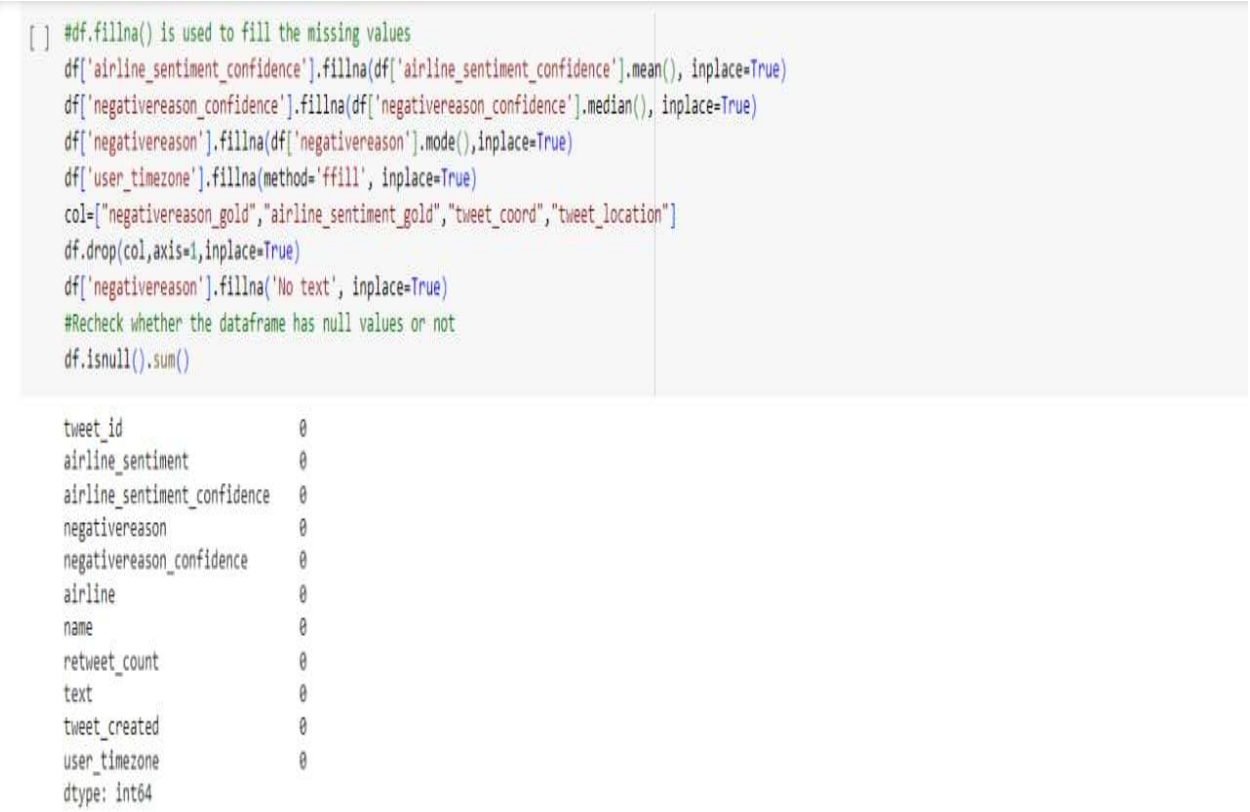


```
[ ] #df.isnull().sum() is used to count the missing values in each column of a dataframe
    df.isnull().sum()
```

```
tweet_id                            0
airline_sentiment                   0
airline_sentiment_confidence        0
negativereason                   5462
negativereason_confidence        4118
airline                             0
airline_sentiment_gold          14600
name                                0
negativereason_gold             14608
retweet_count                       0
text                                0
tweet_coord                     13621
tweet_created                       0
tweet_location                   4733
user_timezone                    4820
dtype: int64
```

The above code returns the count of NULL values.

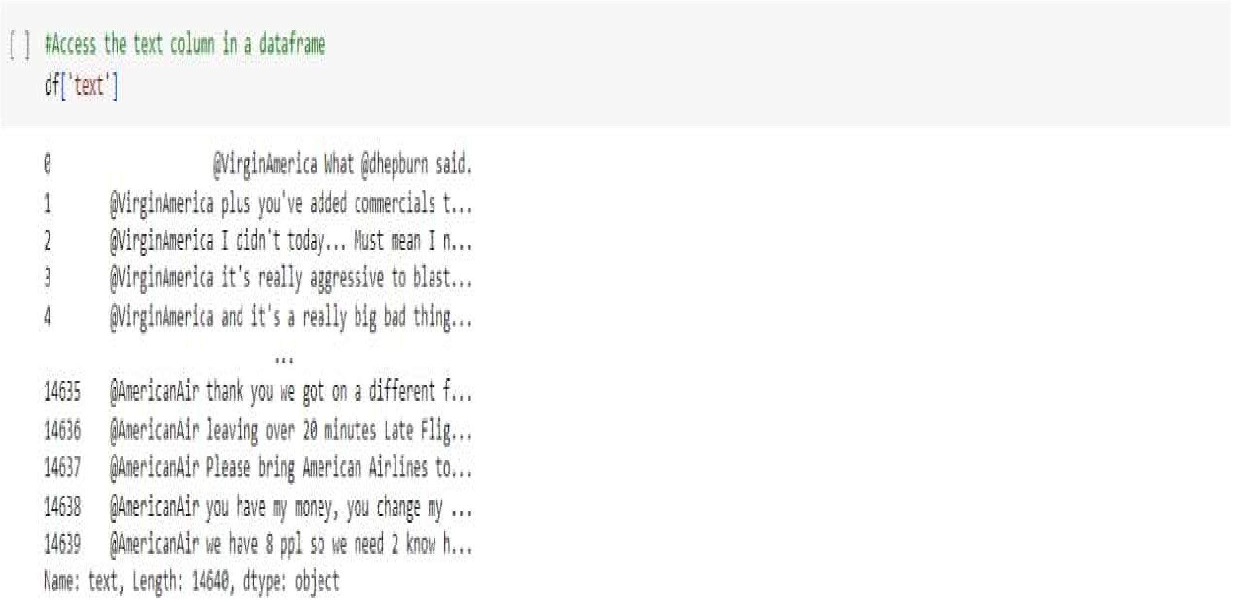
fillna() is used to fill the missing values with mean mode etc.



```
[ ] #df.fillna() is used to fill the missing values
    df['airline_sentiment_confidence'].fillna(df['airline_sentiment_confidence'].mean(), inplace=True)
    df['negativereason_confidence'].fillna(df['negativereason_confidence'].median(), inplace=True)
    df['negativereason'].fillna(df['negativereason'].mode(),inplace=True)
    df['user_timezone'].fillna(method='ffill', inplace=True)
    col=["negativereason_gold","airline_sentiment_gold","tweet_coord","tweet_location"]
    df.drop(col,axis=1,inplace=True)
    df['negativereason'].fillna('No text', inplace=True)
    #Recheck whether the dataframe has null values or not
    df.isnull().sum()
```

```
tweet_id                        0
airline_sentiment               0
airline_sentiment_confidence    0
negativereason                  0
negativereason_confidence       0
airline                         0
name                            0
retweet_count                   0
text                            0
tweet_created                   0
user_timezone                   0
dtype: int64
```

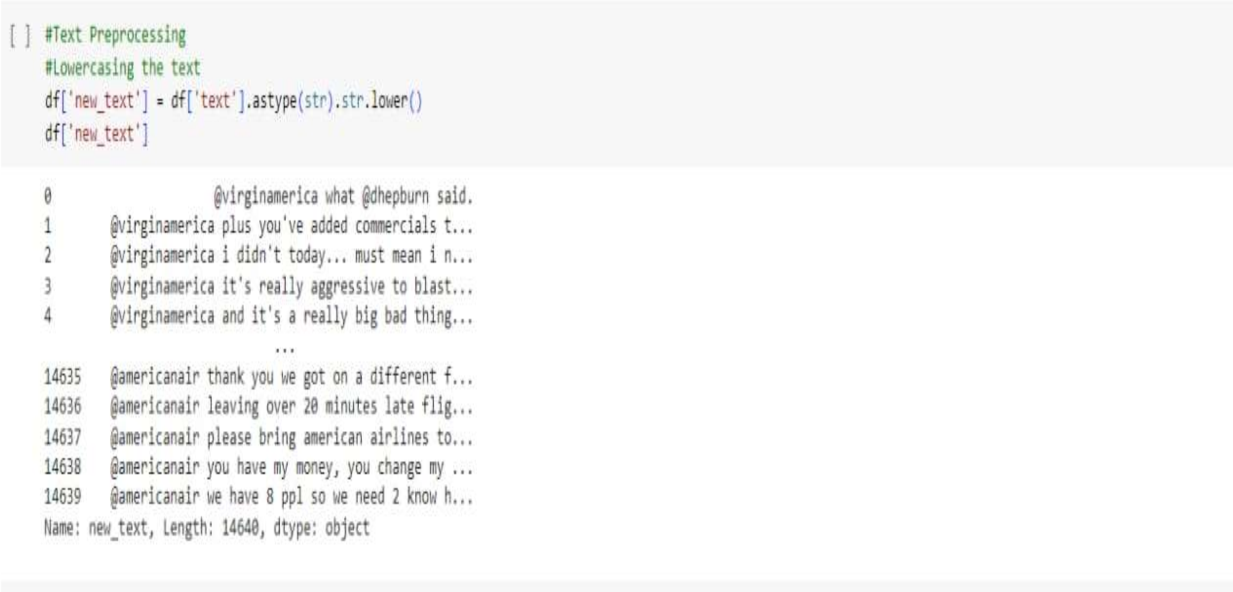
After running above code, the DataFrame df should have missing values filled and you can use df.isnull().sum() to confirm that there are no more missing values in the DataFrame.



```
[ ] #Access the text column in a dataframe
    df['text']
```

```
0                     @VirginAmerica What @dhepburn said.
1        @VirginAmerica plus you've added commercials t...
2        @VirginAmerica I didn't today... Must mean I n...
3        @VirginAmerica it's really aggressive to blast...
4        @VirginAmerica and it's a really big bad thing...
                               ...
14635    @AmericanAir thank you we got on a different f...
14636    @AmericanAir leaving over 20 minutes Late Flig...
14637    @AmericanAir Please bring American Airlines to...
14638    @AmericanAir you have my money, you change my ...
14639    @AmericanAir we have 8 ppl so we need 2 know h...
Name: text, Length: 14640, dtype: object
```

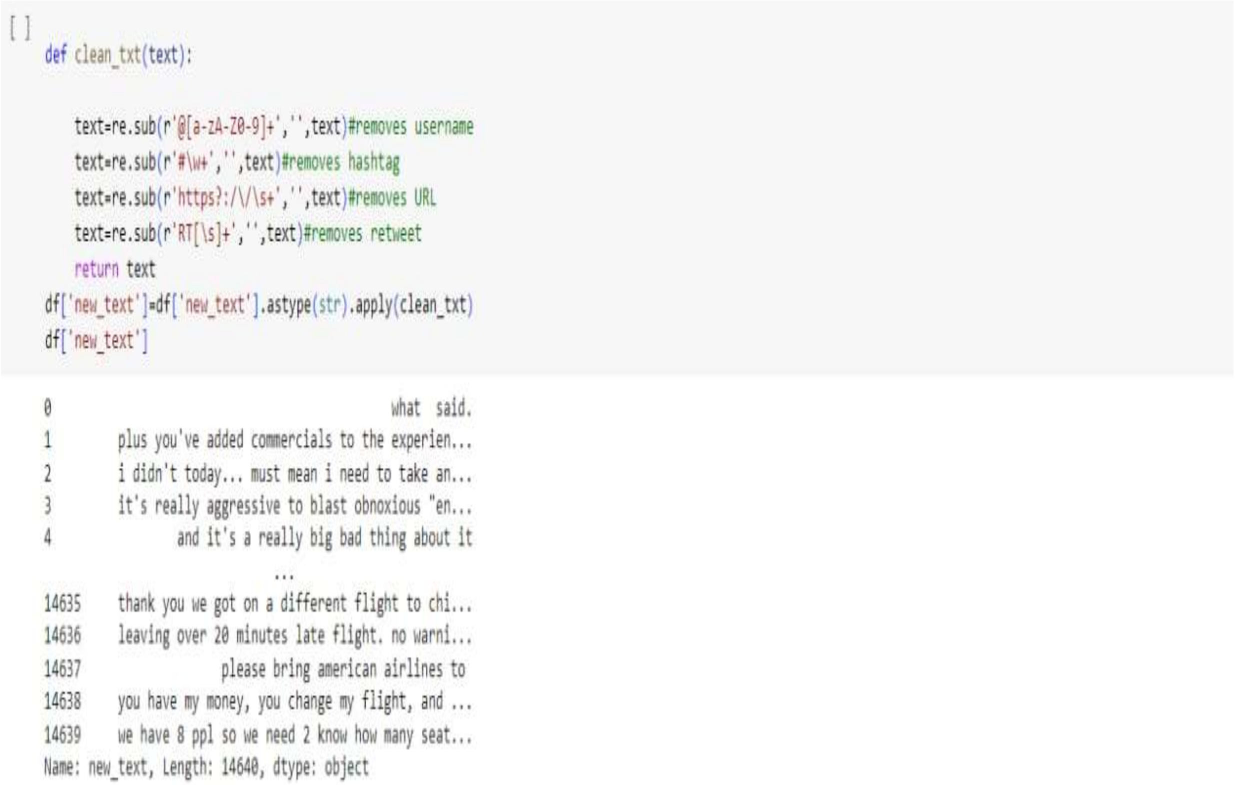
Text Preprocessing:



```
[ ] #Text Preprocessing
    #Lowercasing the text
    df['new_text'] = df['text'].astype(str).str.lower()
    df['new_text']
```

```
0                      @virginamerica what @dhepburn said.
1        @virginamerica plus you've added commercials t...
2        @virginamerica i didn't today... must mean i n...
3        @virginamerica it's really aggressive to blast...
4        @virginamerica and it's a really big bad thing...
                               ...
14635    @americanair thank you we got on a different f...
14636    @americanair leaving over 20 minutes late flig...
14637    @americanair please bring american airlines to...
14638    @americanair you have my money, you change my ...
14639    @americanair we have 8 ppl so we need 2 know h...
Name: new_text, Length: 14640, dtype: object
```

The re.sub(pat, replacement, str) function searches for all the instances of pattern in the given string, and replaces them.



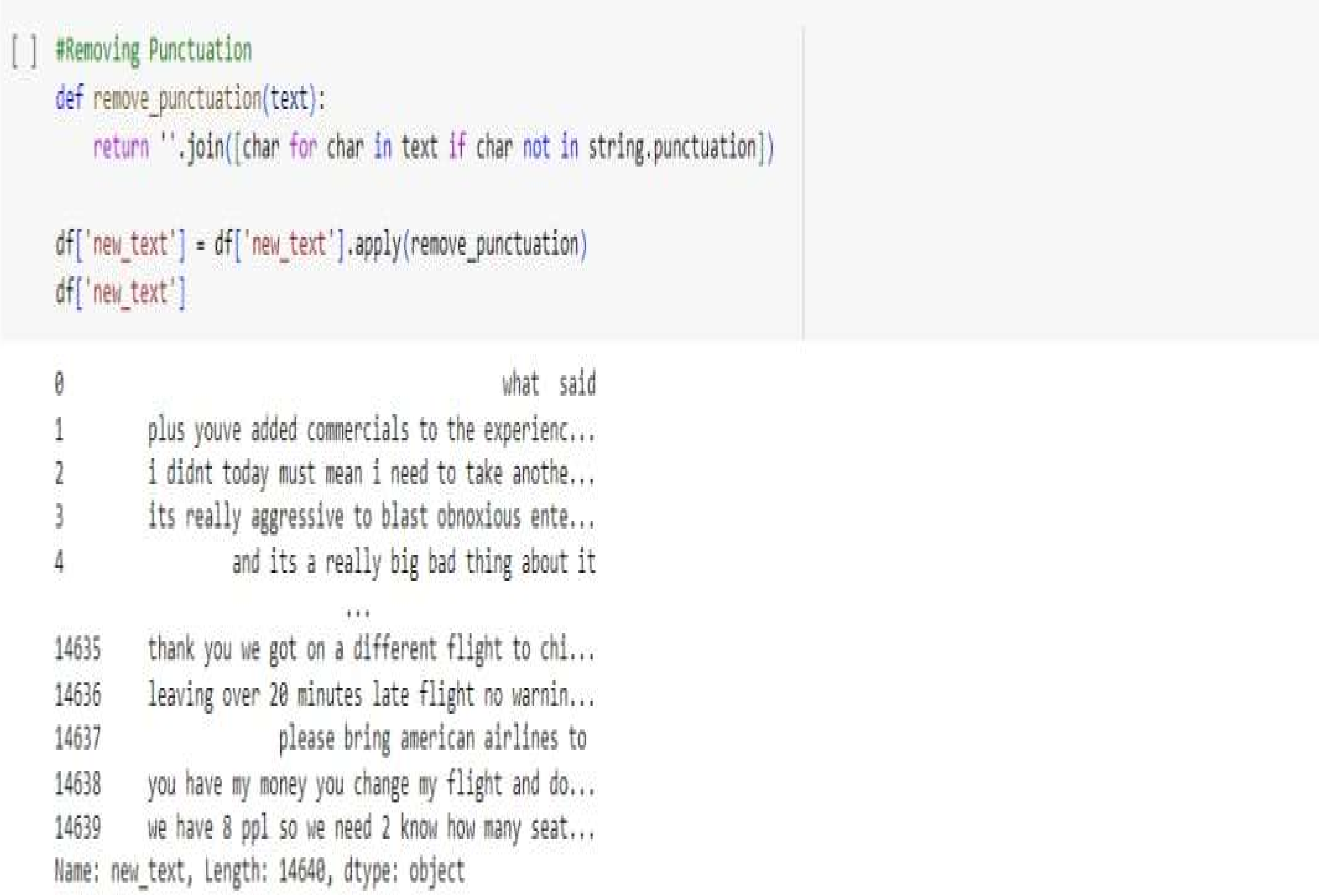
```
[ ]
    def clean_txt(text):


        text=re.sub(r'@[a-zA-Z0-9]+','',text)#removes username
        text=re.sub(r'#\w+','',text)#removes hashtag
        text=re.sub(r'https?:/\/\s+','',text)#removes URL
        text=re.sub(r'RT[\s]+','',text)#removes retweet
        return text
    df['new_text']=df['new_text'].astype(str).apply(clean_txt)
    df['new_text']
```

```
0                                           what  said.
1               plus you've added commercials to the experien...
2               i didn't today... must mean i need to take an...
3               it's really aggressive to blast obnoxious "en...
4                       and it's a really big bad thing about it
                                     ...
14635   thank you we got on a different flight to chi...
14636   leaving over 20 minutes late flight. no warni...
14637                  please bring american airlines to 
14638   you have my money, you change my flight, and ...
14639   we have 8 ppl so we need 2 know how many seat...
Name: new_text, Length: 14640, dtype: object
```

The above code is designed to clean a text column in a Dataframe by removing Twitter-specific elements like usernames,hashtags and URLs.

Removing punctuation from the dataset:

This code is used to remove stopwords from text data in a DataFrame using the Natural Language Toolkit (NLTK) library in Python.

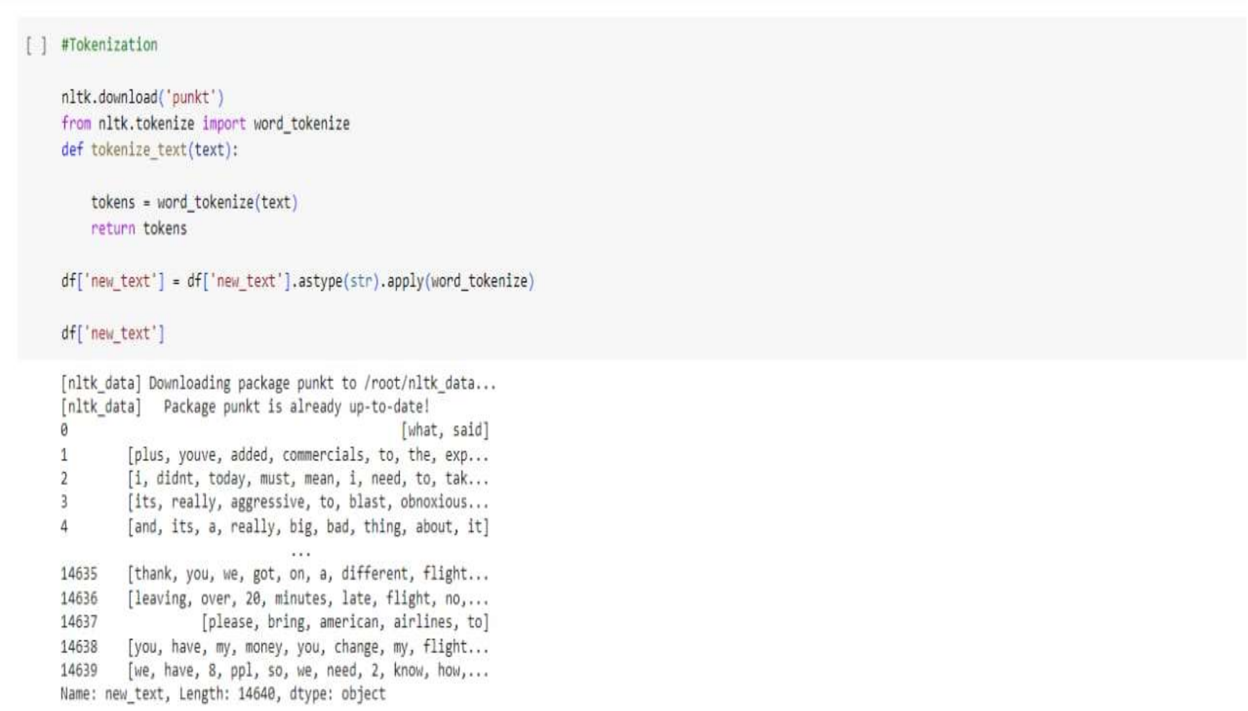


```
[ ] #Removing Punctuation
    def remove_punctuation(text):
        return ''.join([char for char in text if char not in string.punctuation])

    df['new_text'] = df['new_text'].apply(remove_punctuation)
    df['new_text']
```

```
0                                     what  said
1        plus youve added commercials to the experienc...
2        i didnt today must mean i need to take anothe...
3        its really aggressive to blast obnoxious ente...
4                 and its a really big bad thing about it
                              ...
14635    thank you we got on a different flight to chi...
14636    leaving over 20 minutes late flight no warnin...
14637                 please bring american airlines to
14638    you have my money you change my flight and do...
14639    we have 8 ppl so we need 2 know how many seat...
Name: new_text, Length: 14640, dtype: object
```

Tokenization basically refers to splitting up a larger body of text into smaller lines, words.

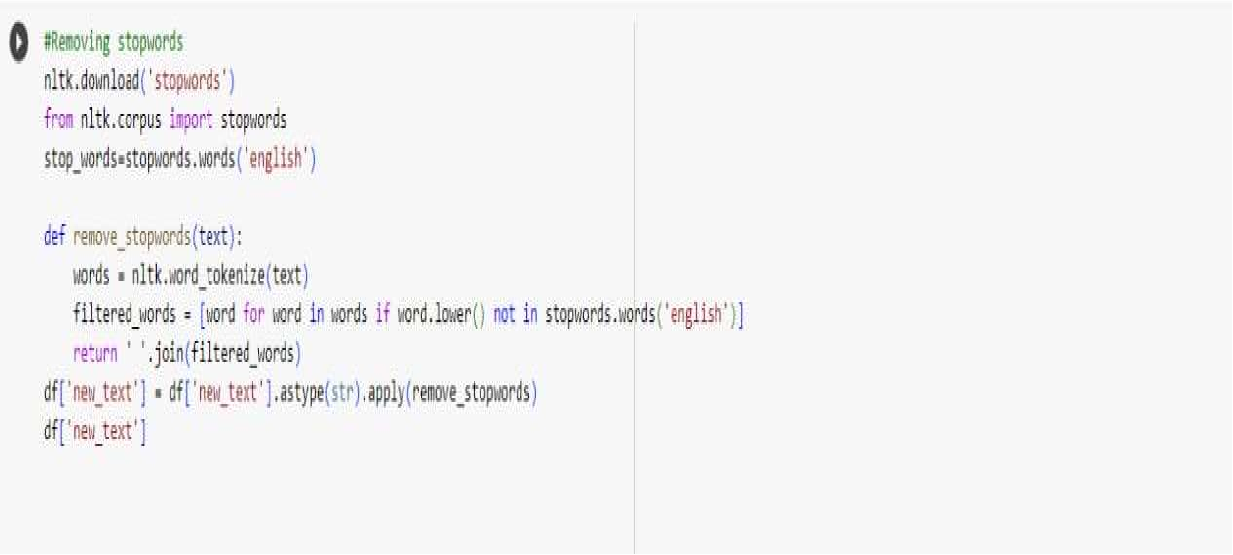
This code, the 'new_text' column in the DataFrame df will contain lists of tokens, where each list represents the tokenized version of the corresponding text entry



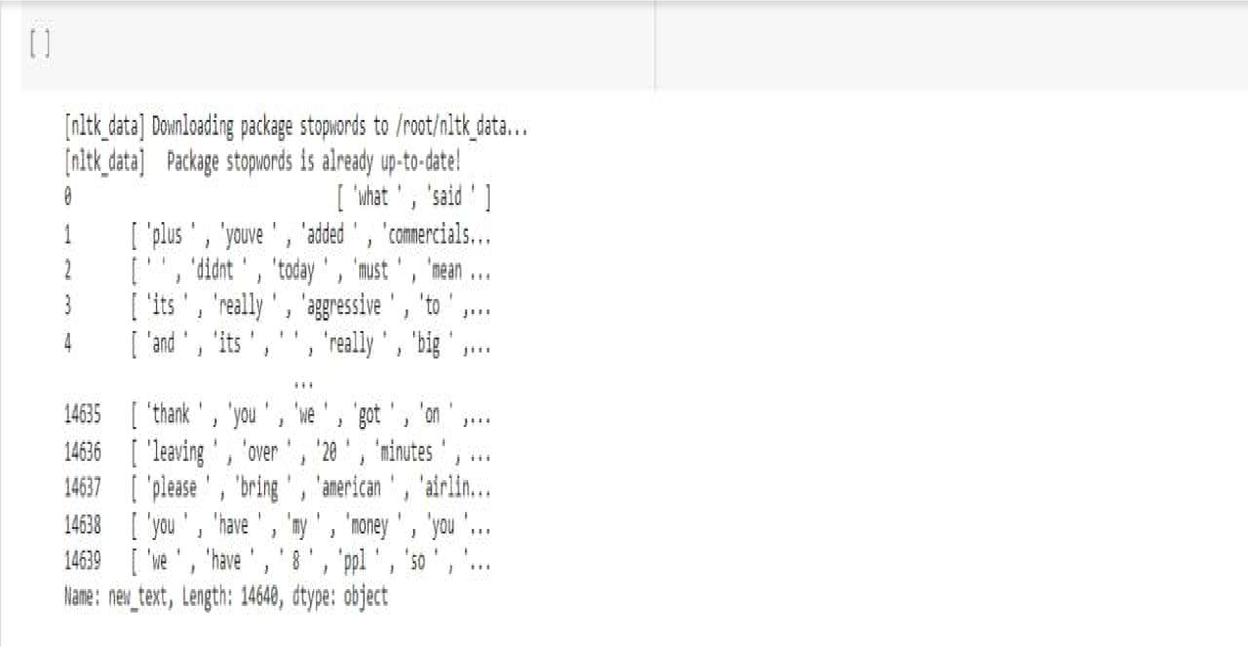
```
[ ]  #Tokenization

     nltk.download('punkt')
     from nltk.tokenize import word_tokenize
     def tokenize_text(text):

         tokens = word_tokenize(text)
         return tokens

     df['new_text'] = df['new_text'].astype(str).apply(word_tokenize)

     df['new_text']
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
0                                              [what, said]
1        [plus, youve, added, commercials, to, the, exp...
2        [i, didnt, today, must, mean, i, need, to, tak...
3        [its, really, aggressive, to, blast, obnoxious...
4        [and, its, a, really, big, bad, thing, about, it]
                               ...
14635    [thank, you, we, got, on, a, different, flight...
14636    [leaving, over, 20, minutes, late, flight, no,...
14637              [please, bring, american, airlines, to]
14638    [you, have, my, money, you, change, my, flight...
14639    [we, have, 8, ppl, so, we, need, 2, know, how,...
Name: new_text, Length: 14640, dtype: object
```

Removing stopwords:



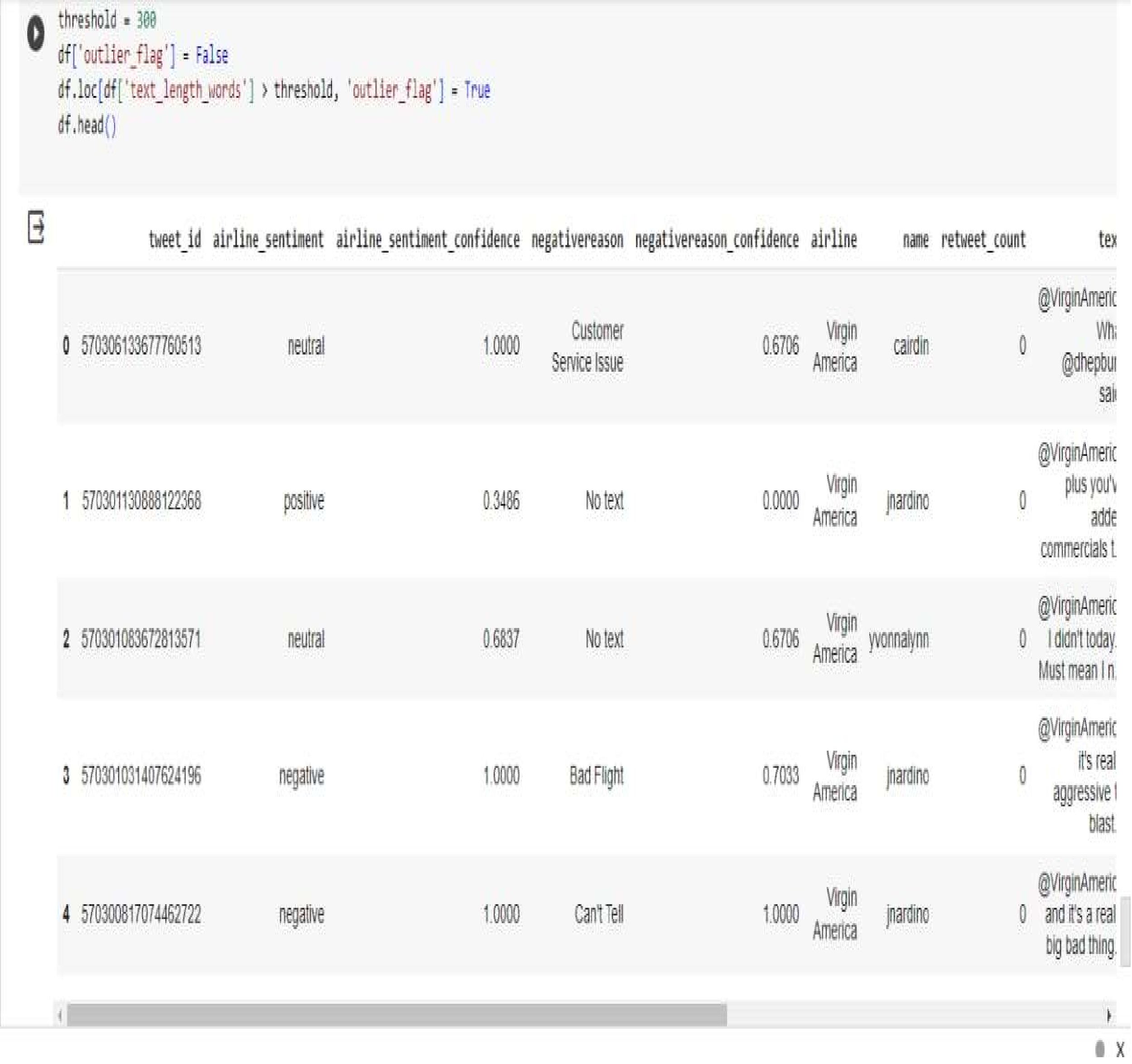
```
#Removing stopwords
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words=stopwords.words('english')

def remove_stopwords(text):
    words = nltk.word_tokenize(text)
    filtered_words = [word for word in words if word.lower() not in stopwords.words('english')]
    return ' '.join(filtered_words)
df['new_text'] = df['new_text'].astype(str).apply(remove_stopwords)
df['new_text']
```



```
[ ]

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
0                            [ 'what ' , 'said ' ]
1        [ 'plus ' , 'youve ' , 'added ' , 'commercials...
2        [ ' ' , 'didnt ' , 'today ' , 'must ' , 'mean ...
3        [ 'its ' , 'really ' , 'aggressive ' , 'to ' ,...
4        [ 'and ' , 'its ' , ' ' , 'really ' , 'big ' ,...
                            ...
14635    [ 'thank ' , 'you ' , 'we ' , 'got ' , 'on ' ,...
14636    [ 'leaving ' , 'over ' , '20 ' , 'minutes ' , ...
14637    [ 'please ' , 'bring ' , 'american ' , 'airlin...
14638    [ 'you ' , 'have ' , 'my ' , 'money ' , 'you '...
14639    [ 'we ' , 'have ' , ' 8 ' , 'ppl ' , 'so ' , '...
Name: new_text, Length: 14640, dtype: object
```

This code is used to remove stopwords from text data in a DataFrame using the Natural Language Toolkit (NLTK) library in Python.

An Outlier is a data-item/object that deviates significantly from the rest of the objects.

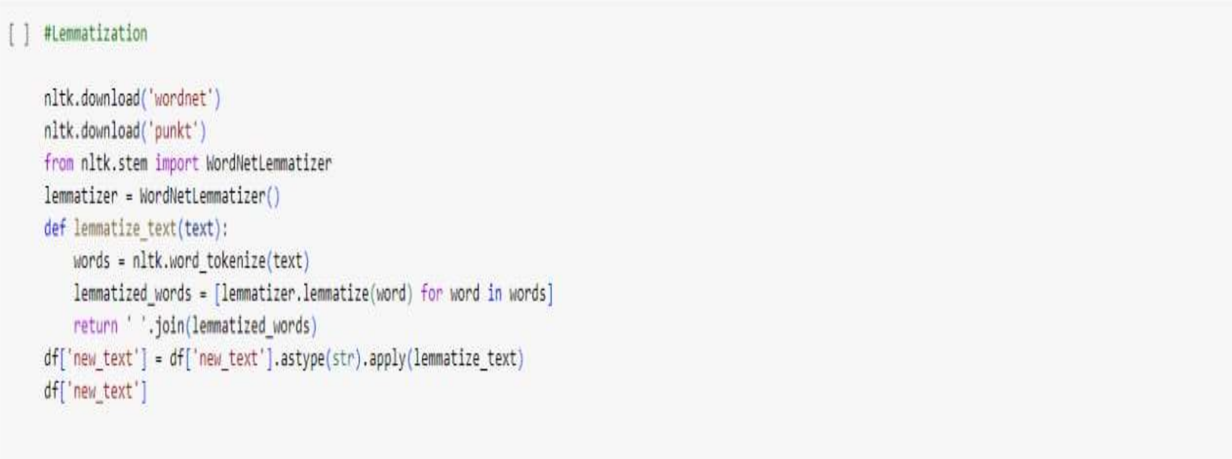


```
threshold = 300
df['outlier_flag'] = False
df.loc[df['text_length_words'] > threshold, 'outlier_flag'] = True
df.head()
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	name	retweet_count	tex
0	570306133677760513	neutral	1.0000	Customer Service Issue	0.6706	Virgin America	cairdin	0	@VirginAmeric Wh @dhepbu sai
1	570301130888122368	positive	0.3486	No text	0.0000	Virgin America	jnardino	0	@VirginAmeric plus you'v adde commercials t
2	570301083672813571	neutral	0.6837	No text	0.6706	Virgin America	yvonnalynn	0	@VirginAmeric I didn't today. Must mean I n
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	jnardino	0	@VirginAmeric it's real aggressive t blast
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	jnardino	0	@VirginAmeric and it's a real big bad thing

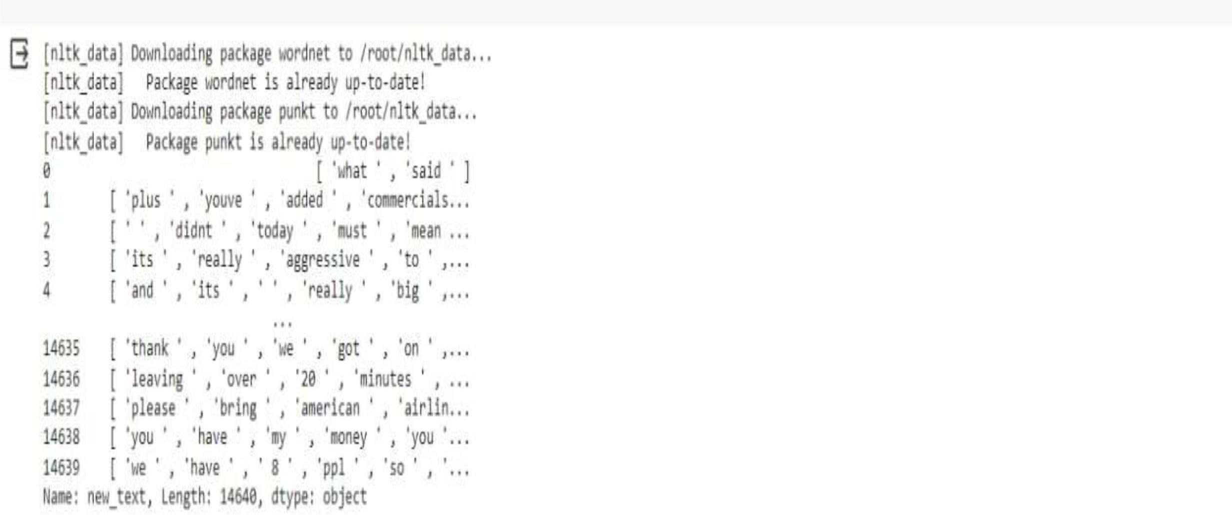
The above code creates an 'outlier_flag' column in the DataFrame df and sets it to True for rows where the 'text_length_words' column exceeds the specified threshold of 300 words. This allows you to flag or identify data points in the DataFrame that are considered outliers based on the text length criterion.

Lemmatization-It is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item.



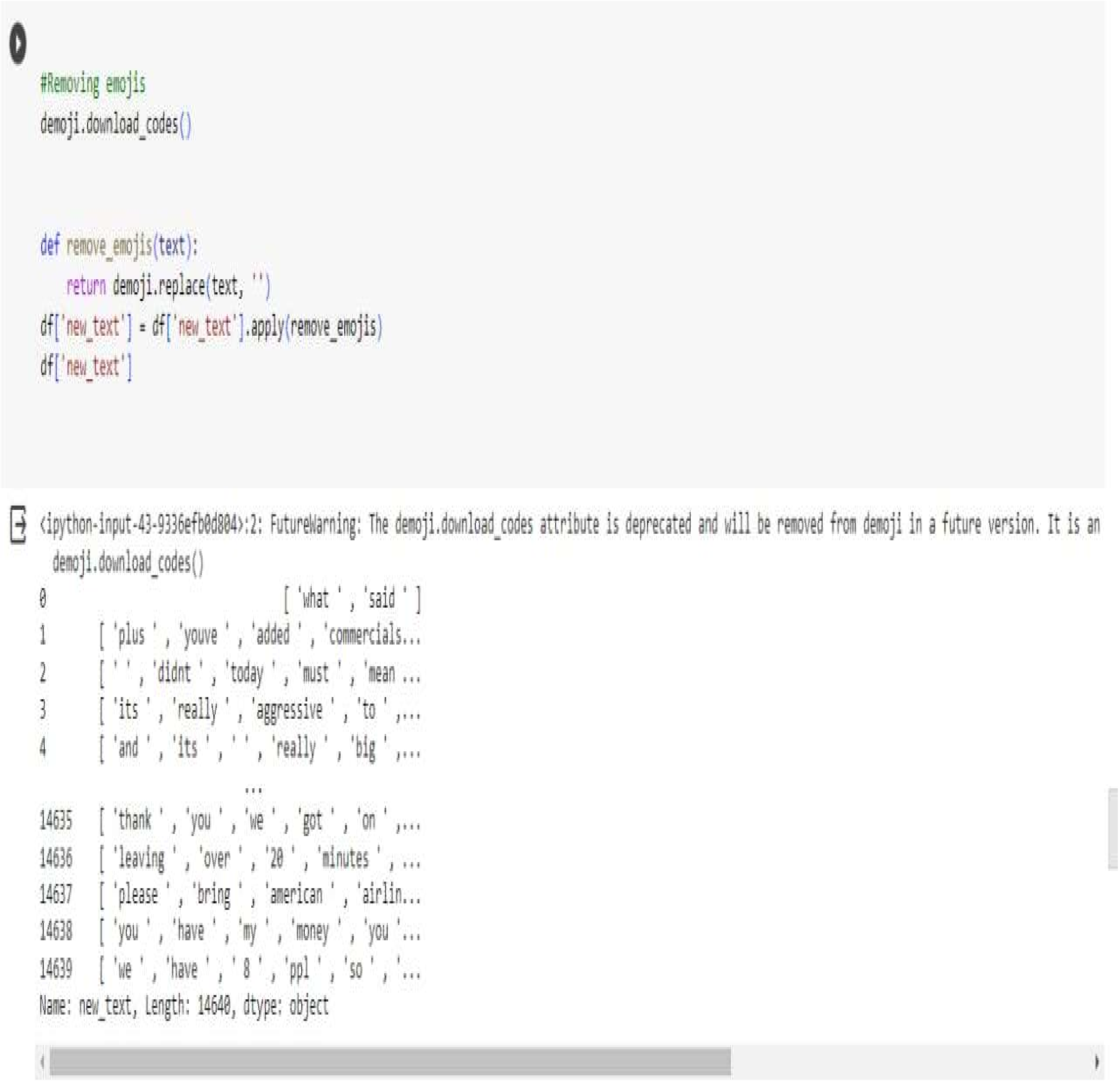
```
[ ] #Lemmatization

    nltk.download('wordnet')
    nltk.download('punkt')
    from nltk.stem import WordNetLemmatizer
    lemmatizer = WordNetLemmatizer()
    def lemmatize_text(text):
        words = nltk.word_tokenize(text)
        lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
        return ' '.join(lemmatized_words)
    df['new_text'] = df['new_text'].astype(str).apply(lemmatize_text)
    df['new_text']
```

After running this code, the 'new_text' column in the DataFrame df will contain the original text data with words lemmatized.



```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
0                               [ 'what ' , 'said ' ]
1        [ 'plus ' , 'youve ' , 'added ' , 'commercials...
2        [ ' ' , 'didnt ' , 'today ' , 'must ' , 'mean ...
3        [ 'its ' , 'really ' , 'aggressive ' , 'to ' ,...
4        [ 'and ' , 'its ' , ' ' , 'really ' , 'big ' ,...
                               ...
14635    [ 'thank ' , 'you ' , 'we ' , 'got ' , 'on ' ,...
14636    [ 'leaving ' , 'over ' , '20 ' , 'minutes ' , ...
14637    [ 'please ' , 'bring ' , 'american ' , 'airlin...
14638    [ 'you ' , 'have ' , 'my ' , 'money ' , 'you '...
14639    [ 'we ' , 'have ' , ' 8 ' , 'ppl ' , 'so ' , '...
Name: new_text, Length: 14640, dtype: object
```

Removing the emojis:

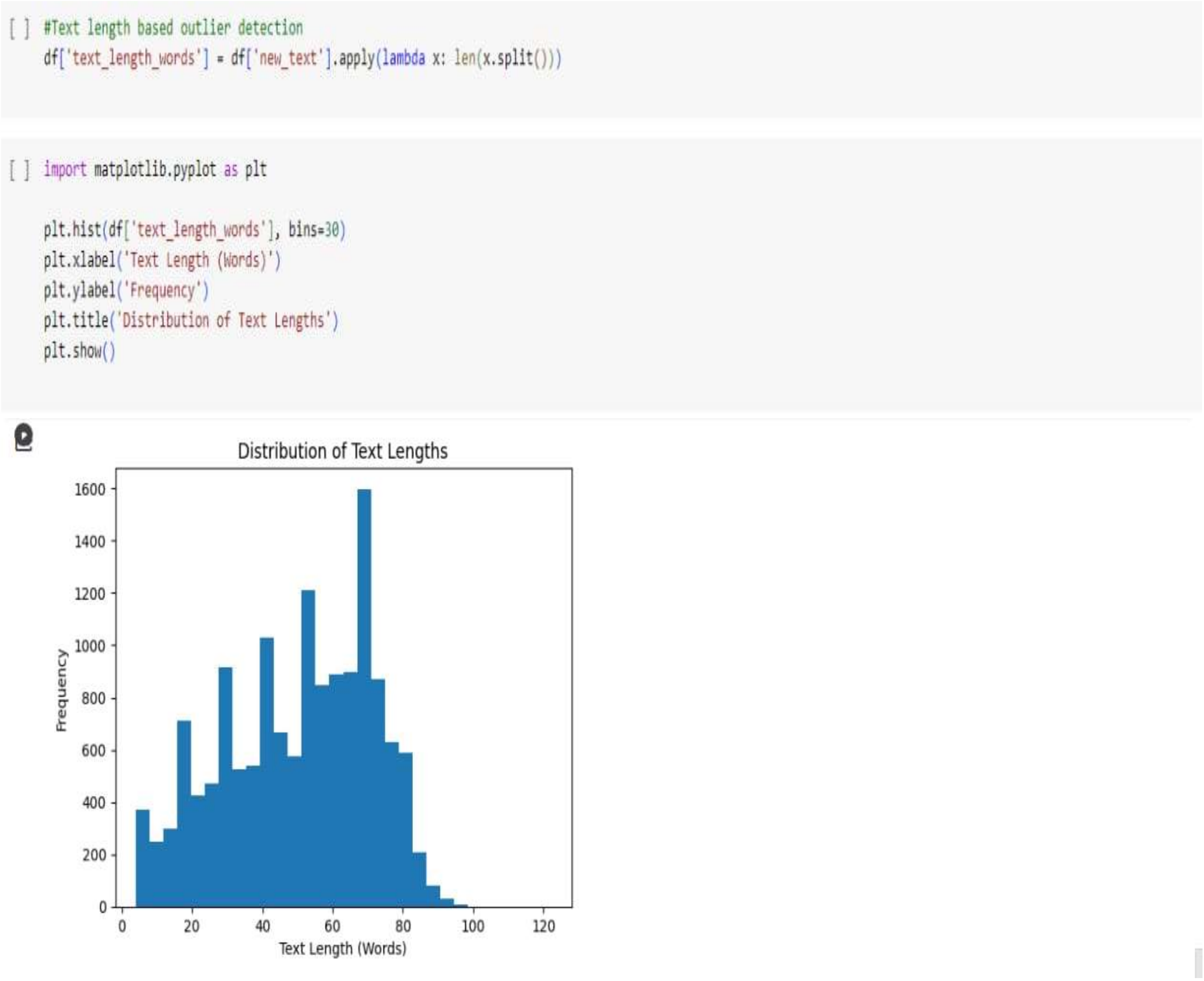


```
#Removing emojis
demoji.download_codes()



def remove_emojis(text):
    return demoji.replace(text, '')
df['new_text'] = df['new_text'].apply(remove_emojis)
df['new_text']
```

```
<ipython-input-43-9336efb0d804>:2: FutureWarning: The demoji.download_codes attribute is deprecated and will be removed from demoji in a future version. It is an
  demoji.download_codes()
0                            [ 'what ' , 'said ' ]
1        [ 'plus ' , 'youve ' , 'added ' , 'commercials...
2        [ ' ' , 'didnt ' , 'today ' , 'must ' , 'mean ...
3        [ 'its ' , 'really ' , 'aggressive ' , 'to ' ,...
4        [ 'and ' , 'its ' , ' ' , 'really ' , 'big ' ,...
                                ...
14635    [ 'thank ' , 'you ' , 'we ' , 'got ' , 'on ' ,...
14636    [ 'leaving ' , 'over ' , '20 ' , 'minutes ' , ...
14637    [ 'please ' , 'bring ' , 'american ' , 'airlin...
14638    [ 'you ' , 'have ' , 'my ' , 'money ' , 'you '...
14639    [ 'we ' , 'have ' , ' 8 ' , 'ppl ' , 'so ' , '...
Name: new_text, Length: 14640, dtype: object
```

After running this code, the 'new_text' column in the DataFrame df will contain the original text data with emojis removed. Emojis will be replaced with an empty string.

Hist()-It is used to create histograms



```
[ ] #Text length based outlier detection
    df['text_length_words'] = df['new_text'].apply(lambda x: len(x.split()))
```

```
[ ] import matplotlib.pyplot as plt

    plt.hist(df['text_length_words'], bins=30)
    plt.xlabel('Text Length (Words)')
    plt.ylabel('Frequency')
    plt.title('Distribution of Text Lengths')
    plt.show()
```

The code generates a histogram to visually represent the distribution of text lengths in the 'text_length_words' column of the DataFrame.