```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import time
import nltk
import re
import string
from nltk.corpus import stopwords
nltk.download('punkt')
nltk.download('stopwords')
from nltk.tokenize import word_tokenize

stop_words = stopwords.words()
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
t1 = time.time()
df = pd.read_csv('/content/drive/MyDrive/Dataset/data_science.csv')
t2 = time.time()
print('Elapsed time [s]: ', np.round(t2-t1,2))
```

```
Elapsed time [s]:  6.56
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 228487 entries, 0 to 228486
Data columns (total 36 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   id               228487 non-null  float64
 1   conversation_id  228487 non-null  float64
 2   created_at       228487 non-null  object 
 3   date             228487 non-null  object 
 4   time             228487 non-null  object 
 5   timezone         228487 non-null  int64  
 6   user_id          228487 non-null  float64
 7   username         228487 non-null  object 
 8   name             228487 non-null  object 
 9   place            348 non-null     object 
 10  tweet            228487 non-null  object 
 11  language         228487 non-null  object 
 12  mentions         228487 non-null  object 
 13  urls             228487 non-null  object 
 14  photos           228487 non-null  object 
 15  replies_count    228487 non-null  int64  
 16  retweets_count   228487 non-null  int64  
 17  likes_count      228487 non-null  int64  
 18  hashtags         228487 non-null  object 
 19  cashtags         228487 non-null  object 
 20  link             228487 non-null  object 
 21  retweet          228487 non-null  bool   
 22  quote_url        9222 non-null    object 
 23  video            228487 non-null  int64  
 24  thumbnail        104585 non-null  object 
 25  near             0 non-null       float64
 26  geo              0 non-null       float64
 27  source           0 non-null       float64
 28  user_rt_id       0 non-null       float64
 29  user_rt          0 non-null       float64
 30  retweet_id       0 non-null       float64
 31  reply_to         228487 non-null  object 
 32  retweet_date     0 non-null       float64
 33  translate        0 non-null       float64
 34  trans_src        0 non-null       float64
 35  trans_dest       0 non-null       float64
dtypes: bool(1), float64(13), int64(5), object(17)
memory usage: 61.2+ MB
```

```
df.isnull().sum()
```

```
id                    0
conversation_id       0
```

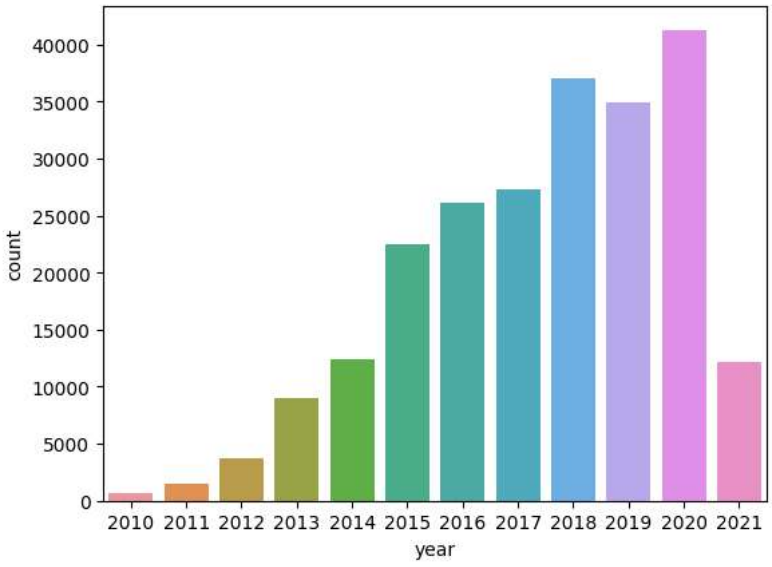
```
created_at              0
date                    0
time                    0
timezone                0
user_id                 0
username                0
name                    0
place              228139
tweet                   0
language                0
mentions                0
urls                    0
photos                  0
replies_count           0
retweets_count          0
likes_count             0
hashtags                0
cashtags                0
link                    0
retweet                 0
quote_url          219265
video                   0
thumbnail          123902
near               228487
geo                228487
source             228487
user_rt_id         228487
user_rt            228487
retweet_id         228487
reply_to                0
retweet_date       228487
translate          228487
trans_src          228487
trans_dest         228487
dtype: int64
```

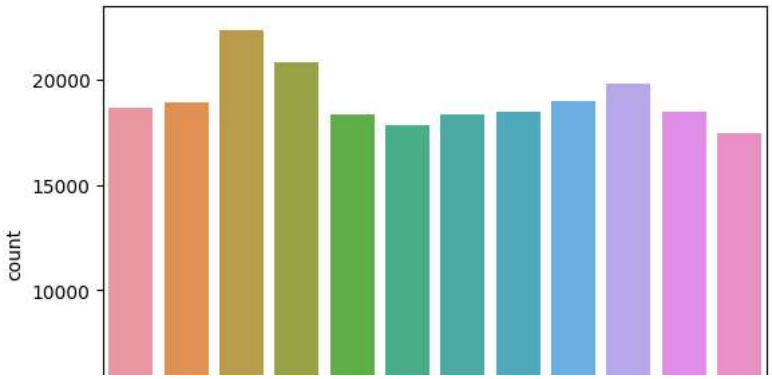
```
df.drop(['place','quote_url','thumbnail','near','geo','source','user_rt_id','user_rt','retweet_id','retweet_date','translate','trans_src','tr
```

```
df.isnull().sum()
```

```
id                 0
conversation_id    0
created_at         0
date               0
time               0
timezone           0
user_id            0
username           0
name               0
tweet              0
language           0
mentions           0
urls               0
photos             0
replies_count      0
retweets_count     0
likes_count        0
hashtags           0
cashtags           0
link               0
retweet            0
video              0
reply_to           0
dtype: int64
```

```
df['year'] = pd.DatetimeIndex(df['date']).year
sns.countplot(x ='year', data = df)
plt.show()
df['month'] = pd.DatetimeIndex(df['date']).month
sns.countplot(x ='month', data = df)
plt.show()
```






```
apostrophe_dict = {
"ain't": "am not / are not",
"aren't": "are not / am not",
"can't": "cannot",
"can't've": "cannot have",
"'cause": "because",
"could've": "could have",
"couldn't": "could not",
"couldn't've": "could not have",
"didn't": "did not",
"doesn't": "does not",
"don't": "do not",
"hadn't": "had not",
"hadn't've": "had not have",
"hasn't": "has not",
"haven't": "have not",
"he'd": "he had / he would",
"he'd've": "he would have",
"he'll": "he shall / he will",
"he'll've": "he shall have / he will have",
"he's": "he has / he is",
"how'd": "how did",
"how'd'y": "how do you",
"how'll": "how will",
"how's": "how has / how is",
"i'd": "I had / I would",
"i'd've": "I would have",
"i'll": "I shall / I will",
"i'll've": "I shall have / I will have",
"i'm": "I am",
"i've": "I have",
"isn't": "is not",
"it'd": "it had / it would",
"it'd've": "it would have",
"it'll": "it shall / it will",
"it'll've": "it shall have / it will have",
"it's": "it has / it is",
"let's": "let us",
```

```
"ma'am": "madam",
"mayn't": "may not",
"might've": "might have",
"mightn't": "might not",
"mightn't've": "might not have",
"must've": "must have",
"mustn't": "must not",
"mustn't've": "must not have",
"needn't": "need not",
"needn't've": "need not have",
"o'clock": "of the clock",
"oughtn't": "ought not",
"oughtn't've": "ought not have",
"shan't": "shall not",
"sha'n't": "shall not",
"shan't've": "shall not have",
"she'd": "she had / she would",
"she'd've": "she would have",
"she'll": "she shall / she will",
"she'll've": "she shall have / she will have",
"she's": "she has / she is",
"should've": "should have",
"shouldn't": "should not",
"shouldn't've": "should not have",
"so've": "so have",
"so's": "so as / so is",
"that'd": "that would / that had",
"that'd've": "that would have",
"that's": "that has / that is",
"there'd": "there had / there would",
"there'd've": "there would have",
"there's": "there has / there is",
"they'd": "they had / they would",
"they'd've": "they would have",
"they'll": "they shall / they will",
"they'll've": "they shall have / they will have",
"they're": "they are",
"they've": "they have",
"to've": "to have",
"wasn't": "was not",
"we'd": "we had / we would",
"we'd've": "we would have",
"we'll": "we will",
"we'll've": "we will have",
"we're": "we are",
"we've": "we have",
"weren't": "were not",
"what'll": "what shall / what will",
"what'll've": "what shall have / what will have",
"what're": "what are",
"what's": "what has / what is",
"what've": "what have",
"when's": "when has / when is",
"when've": "when have",
"where'd": "where did",
"where's": "where has / where is",
"where've": "where have",
"who'll": "who shall / who will",
"who'll've": "who shall have / who will have",
"who's": "who has / who is",
"who've": "who have",
"why's": "why has / why is",
"why've": "why have",
"will've": "will have",
"won't": "will not",
"won't've": "will not have",
"would've": "would have",
"wouldn't": "would not",
"wouldn't've": "would not have",
"y'all": "you all",
"y'all'd": "you all would",
"y'all'd've": "you all would have",
"y'all're": "you all are",
"y'all've": "you all have",
"you'd": "you had / you would",
"you'd've": "you would have",
"you'll": "you shall / you will",
```

```
"you'll've": "you shall have / you will have",
"you're": "you are",
"you've": "you have",
"rt":"",
"RT":"",
"http":"",
"https":""
}

def lookup_dict(text, dictionary):
    for word in str(text).split():
        if word.lower() in dictionary:
            if word.lower() in str(text).split():
                text = text.replace(word, dictionary[word.lower()])
    return text

df['tweet'] = df['tweet'].apply(lambda x: lookup_dict(x,apostrophe_dict))


df.drop(['id','conversation_id',
'created_at',
'time',
'timezone','user_id','username','name','language','mentions','urls',
'photos','replies_count','retweets_count','likes_count','hashtags','link','retweet','video','reply_to','cashtags'],axis=1,inplace=True)


df['tweet'] =df['tweet'].apply(lambda x: word_tokenize(str(x)))
stop_words = set(stopwords.words('english'))
df['tweet'] = df['tweet'].apply(lambda x: [word for word in x if not word in stop_words])


def cleaning(text):
    text = str(text)
    text = text.lower()
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('[’“”…]', '', text)

    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"
                           u"\U0001F300-\U0001F5FF"
                           u"\U0001F680-\U0001F6FF"
                           u"\U0001F1E0-\U0001F1FF"
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+", flags=re.UNICODE)
    text = emoji_pattern.sub(r'', text)

    # removing the stop-words
    text_tokens = word_tokenize(text)
    tokens_without_sw = [word for word in text_tokens if not word in stop_words]
    filtered_sentence = (" ").join(tokens_without_sw)
    text = filtered_sentence

    return text

dt = df['tweet'].apply(cleaning)


from textblob import TextBlob
def GetTextSubjectivity(txt):
    txt=str(txt)
    return TextBlob(txt).sentiment.subjectivity

def GetTextPolarity(txt):
    txt=str(txt)
    return TextBlob(txt).sentiment.polarity

df['Subjectivity'] = df['tweet'].apply(GetTextSubjectivity)
df['Polarity'] = df['tweet'].apply(GetTextPolarity)

def GetTextAnalysis(a):
    if a<0:
        return "Negative"
    elif a==0:
        return "Neutral"
```

```
    else:
        return "Positive"

df["Score"]=df['Polarity'].apply(GetTextAnalysis)


positive=df[df['Score']=='Positive']
print (str(round(positive.shape[0]/(df.shape[0])*100)) + "% Positive tweets")
pos = positive.shape[0]/(df.shape[0])*100

negative=df[df['Score']=='Negative']
print (str(round(negative.shape[0]/(df.shape[0])*100)) + "% Negative tweets")
neg = negative.shape[0]/df.shape[0]*100

neutral=df[df['Score']=='Neutral']
print (str(round((neutral.shape[0])/(df.shape[0])*100)) + "% neutral tweets")
neu = neutral.shape[0]/df.shape[0]*100


explode=(0.05,0,0)
labels = 'positive','negative','neutral'
sizes = [pos,neg,neu]
colours=['#9BBFE0','#E8A09A','#FBE29F']
plt.pie(sizes,explode=explode,colors=colours,autopct='%.2f%%',startangle=0)
plt.title('Percentage of Sentiment in all Tweets \n', fontsize=24)
plt.legend(labels,loc=(-0.05,0.05),shadow=True)
plt.axis('equal')
```