In [1]:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
```

In [2]:

```
train = pd.read_csv('train_tweet.csv')
test = pd.read_csv('test_tweets.csv')

print(train.shape)
print(test.shape)
```

```
(31962, 3)
(17197, 2)
```

In [3]:

```
train.head()
```

Out[3]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

In [4]:

```
test.head()
```

Out[4]:

	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedic...
1	31964	@user #white #supremacists want everyone to s...
2	31965	safe ways to heal your #acne!! #altwaystohe...
3	31966	is the hp and the cursed child book up for res...
4	31967	3rd #bihday to my amazing, hilarious #nephew...

In [5]:

```
train.isnull().any()
test.isnull().any()
```

Out[5]:

```
id       False
tweet    False
dtype: bool
```

In [6]:

```
# checking out the negative comments from the train set

train[train['label'] == 0].head(10)
```

Out[6]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation
5	6	0	[2/2] huge fan fare and big talking before the...
6	7	0	@user camping tomorrow @user @user @user @use...
7	8	0	the next school year is the year for exams.ð□□...
8	9	0	we won!!! love the land!!! #allin #cavs #champ...
9	10	0	@user @user welcome here ! i'm it's so #gr...

In [7]:

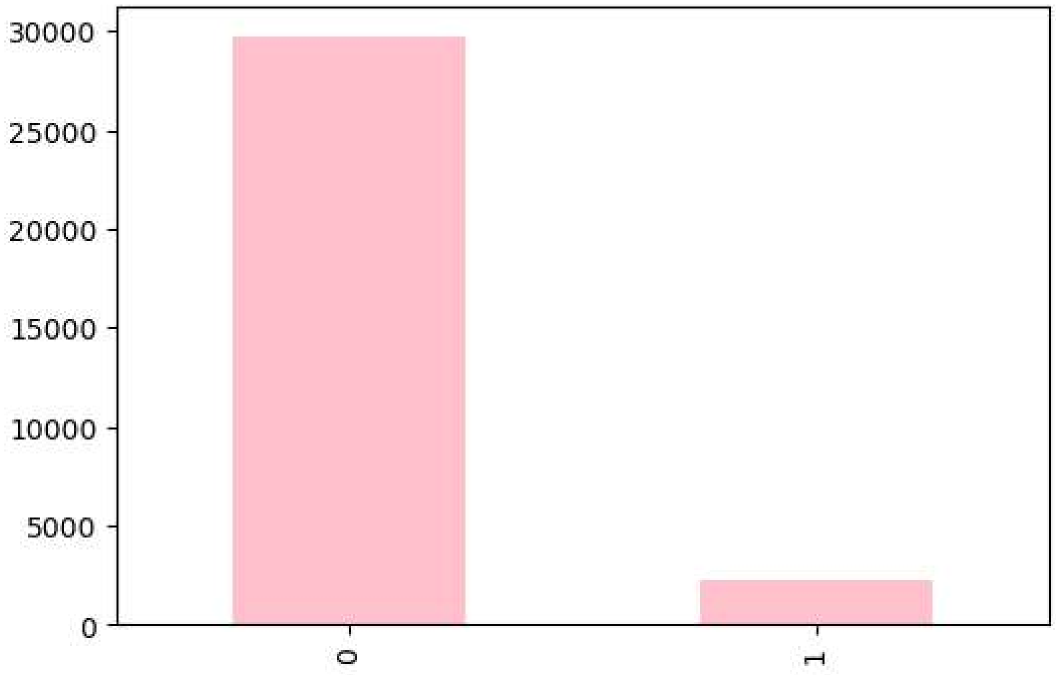
```
# checking out the postive comments from the train set

train[train['label'] == 1].head(10)
```

Out[7]:

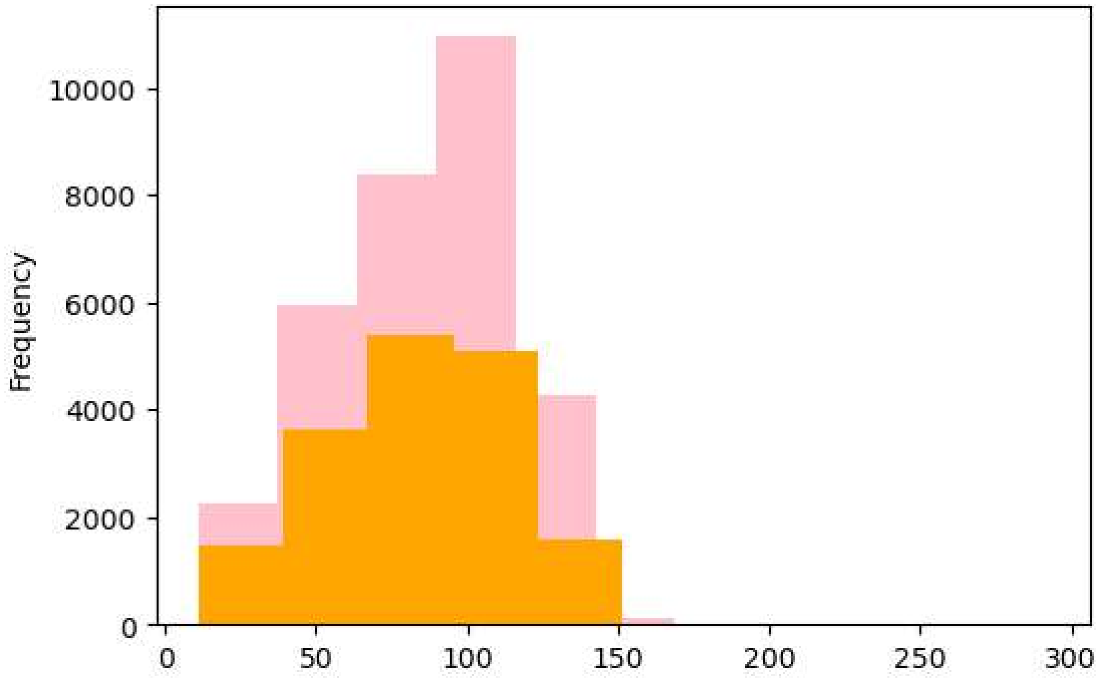
	id	label	tweet
13	14	1	@user #cnn calls #michigan middle school 'buil...
14	15	1	no comment! in #australia #opkillingbay #se...
17	18	1	retweet if you agree!
23	24	1	@user @user lumpy says i am a . prove it lumpy.
34	35	1	it's unbelievable that in the 21st century we'...
56	57	1	@user lets fight against #love #peace
68	69	1	ð□□©the white establishment can't have blk fol...
77	78	1	@user hey, white people: you can call people '...
82	83	1	how the #altright uses &amp; insecurity to lu...
111	112	1	@user i'm not interested in a #linguistics tha...

In [8]:
```
train['label'].value_counts().plot.bar(color = 'pink', figsize = (6, 4))
```

Out[8]:
```
<Axes: >
```



In [9]:
```
# checking the distribution of tweets in the data

length_train = train['tweet'].str.len().plot.hist(color = 'pink', figsiz
length_test = test['tweet'].str.len().plot.hist(color = 'orange', figsiz
```



In [10]:
```
# adding a column to represent the length of the tweet

train['len'] = train['tweet'].str.len()
test['len'] = test['tweet'].str.len()

train.head(10)
```

Out[10]:

	id	label	tweet	len
0	1	0	@user when a father is dysfunctional and is s...	102
1	2	0	@user @user thanks for #lyft credit i can't us...	122
2	3	0	bihday your majesty	21
3	4	0	#model i love u take with u all the time in ...	86
4	5	0	factsguide: society now #motivation	39
5	6	0	[2/2] huge fan fare and big talking before the...	116
6	7	0	@user camping tomorrow @user @user @user @use...	74
7	8	0	the next school year is the year for exams.ð□□...	143
8	9	0	we won!!! love the land!!! #allin #cavs #champ...	87
9	10	0	@user @user welcome here ! i'm it's so #gr...	50

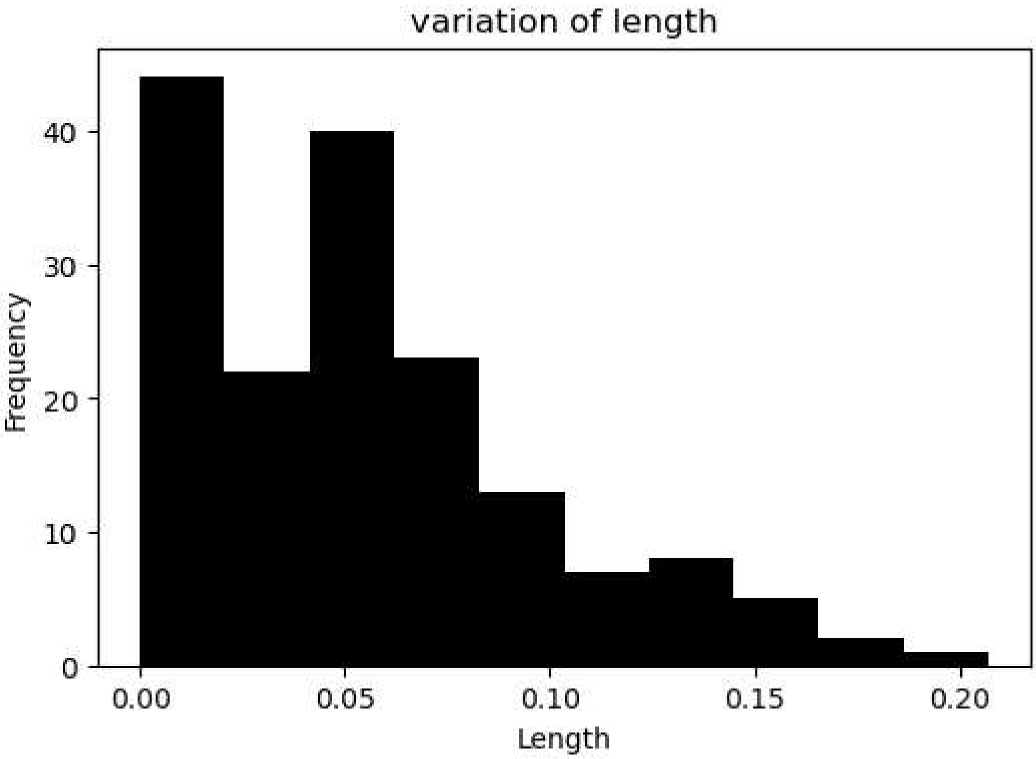
In [11]:
```
train.groupby('label').describe()
```

Out[11]:

	id								len
	count	mean	std	min	25%	50%	75%	max	cour
label									
0	29720.0	15974.454441	9223.783469	1.0	7981.75	15971.5	23965.25	31962.0	2972
1	2242.0	16074.896075	9267.955758	14.0	8075.25	16095.0	24022.00	31961.0	224

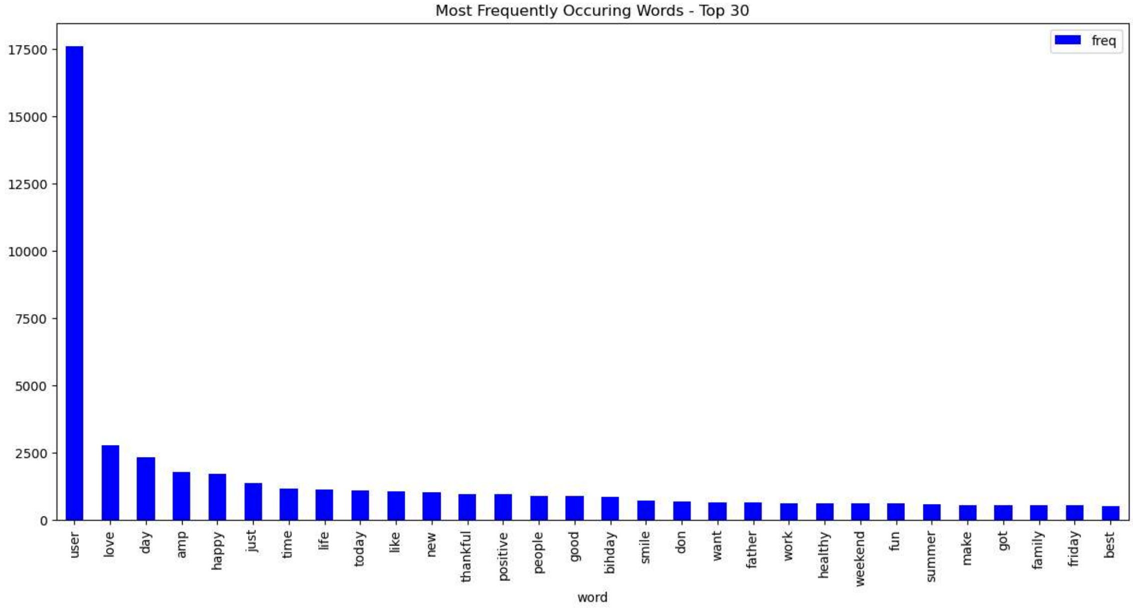
In [12]:
```
train.groupby('len').mean()['label'].plot.hist(color = 'black', figsize
plt.title('variation of length')
plt.xlabel('Length')
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_26068\3421818104.py:1: Futu
reWarning: The default value of numeric_only in DataFrameGroupBy.mean i
s deprecated. In a future version, numeric_only will default to False.
Either specify numeric_only or select only columns which should be vali
d for the function.
  train.groupby('len').mean()['label'].plot.hist(color = 'black', figsi
ze = (6, 4),)
```



In [13]:
```
from sklearn.feature_extraction.text import CountVectorizer


cv = CountVectorizer(stop_words = 'english')
words = cv.fit_transform(train.tweet)

sum_words = words.sum(axis=0)

words_freq = [(word, sum_words[0, i]) for word, i in cv.vocabulary_.item
words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)

frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])

frequency.head(30).plot(x='word', y='freq', kind='bar', figsize=(15, 7),
plt.title("Most Frequently Occuring Words - Top 30")
```

Out[13]:
```
Text(0.5, 1.0, 'Most Frequently Occuring Words - Top 30')
```

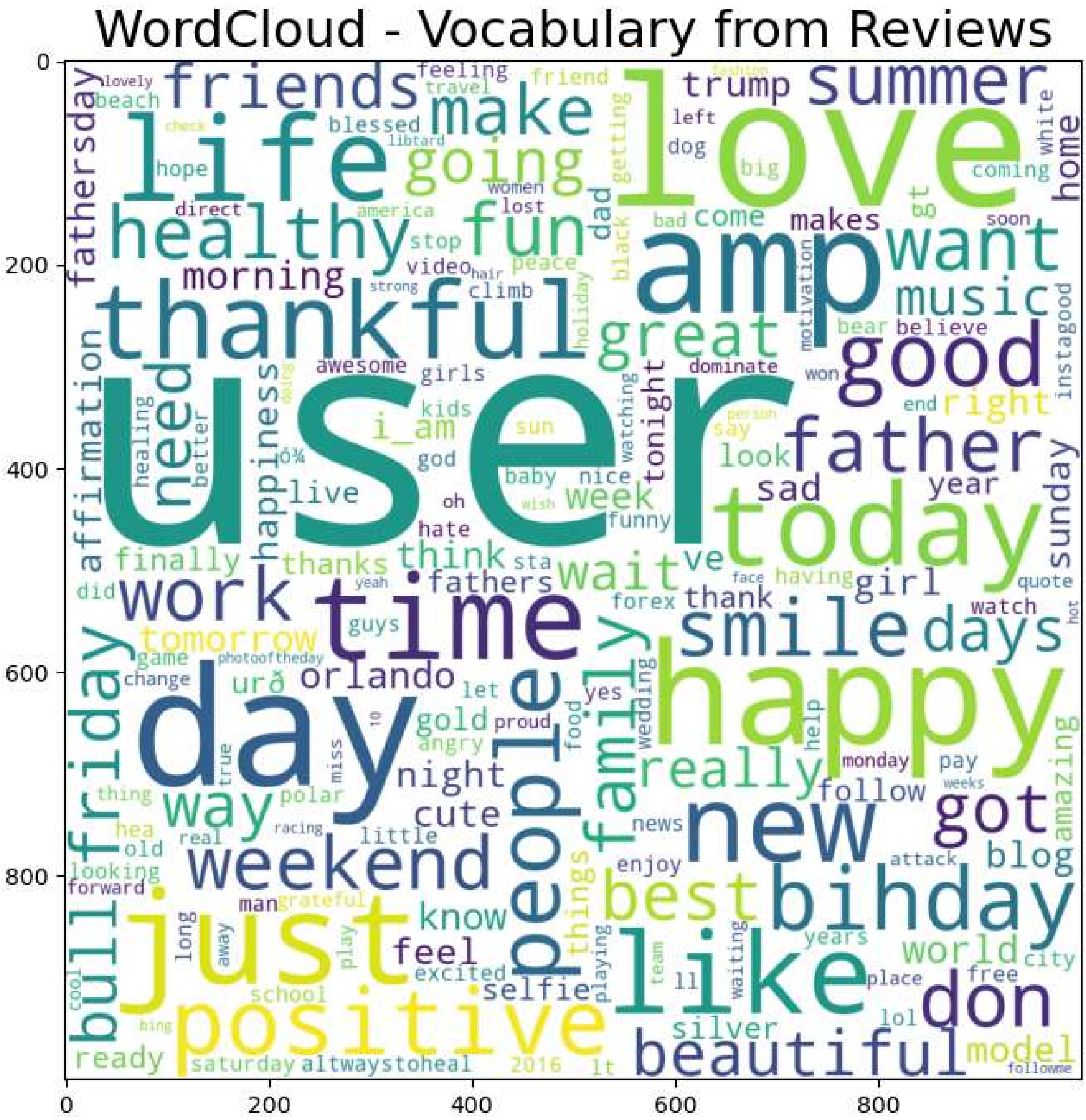


In [14]:

```
from wordcloud import WordCloud

wordcloud = WordCloud(background_color = 'white', width = 1000, height =

plt.figure(figsize=(10,8))
plt.imshow(wordcloud)
plt.title("WordCloud - Vocabulary from Reviews", fontsize = 22)
```

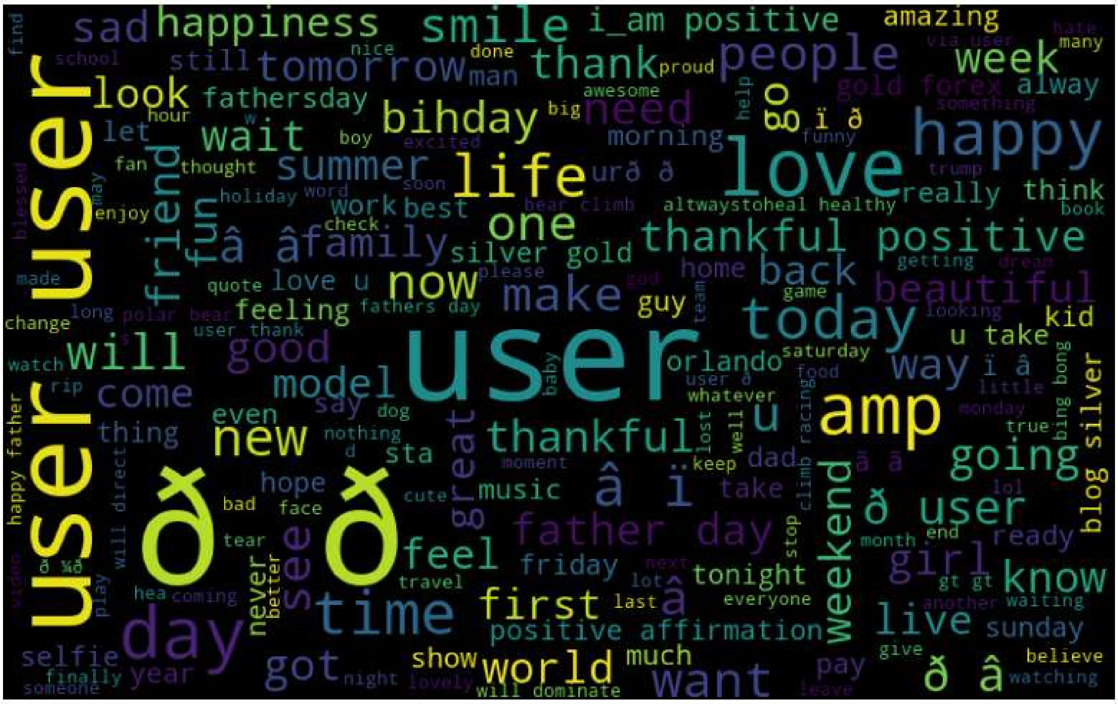
Out[14]:

```
Text(0.5, 1.0, 'WordCloud - Vocabulary from Reviews')
```

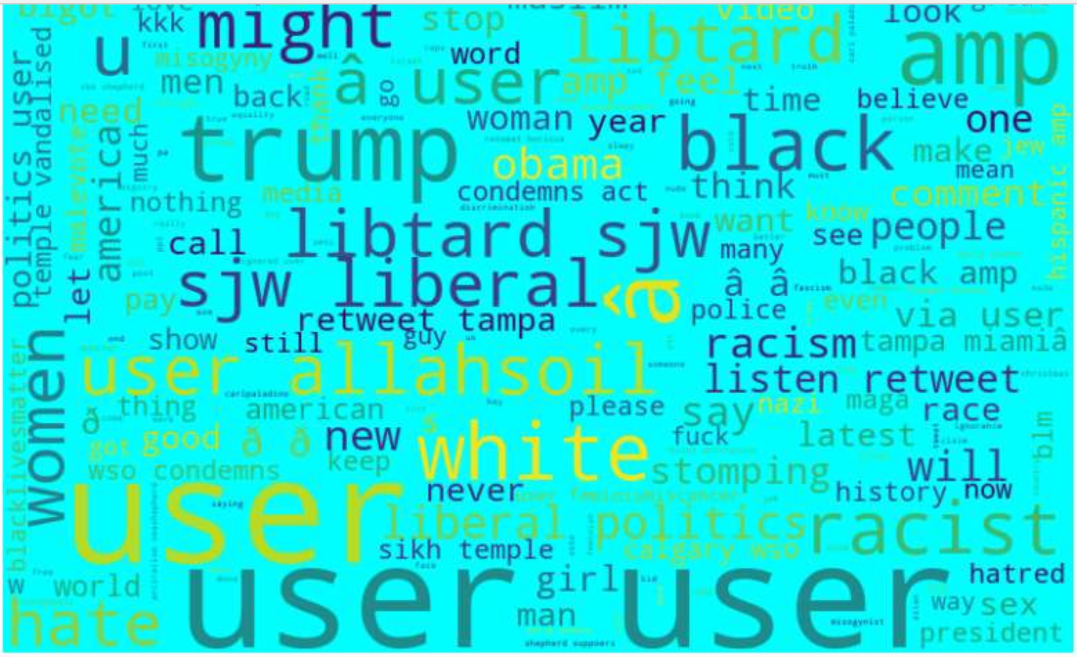


In [15]:
```
normal_words =' '.join([text for text in train['tweet'][train['label'] =

wordcloud = WordCloud(width=800, height=500, random_state = 0, max_font_
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title('The Neutral Words')
plt.show()
```

The Neutral Words



In [16]:
```
negative_words =' '.join([text for text in train['tweet'][train['label']

wordcloud = WordCloud(background_color = 'cyan', width=800, height=500,
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title('The Negative Words')
plt.show()
```



In [ ]:

In [ ]:

In [17]:
```
# removing unwanted patterns from the data

import re
import nltk

nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [ ]:

In [18]:

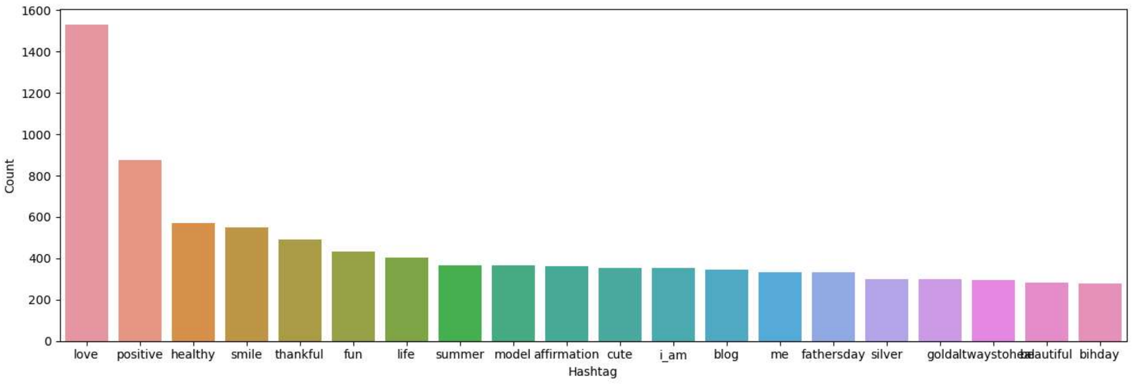
```
# collecting the hashtags

def hashtag_extract(x):
    hashtags = []

    for i in x:
        ht = re.findall(r"#(\w+)", i)
        hashtags.append(ht)

    return hashtags
```

In [19]:

```
# extracting hashtags from non racist/sexist tweets
HT_regular = hashtag_extract(train['tweet'][train['label'] == 0])

# extracting hashtags from racist/sexist tweets
HT_negative = hashtag_extract(train['tweet'][train['label'] == 1])

# unnesting list
HT_regular = sum(HT_regular,[])
HT_negative = sum(HT_negative,[])
```

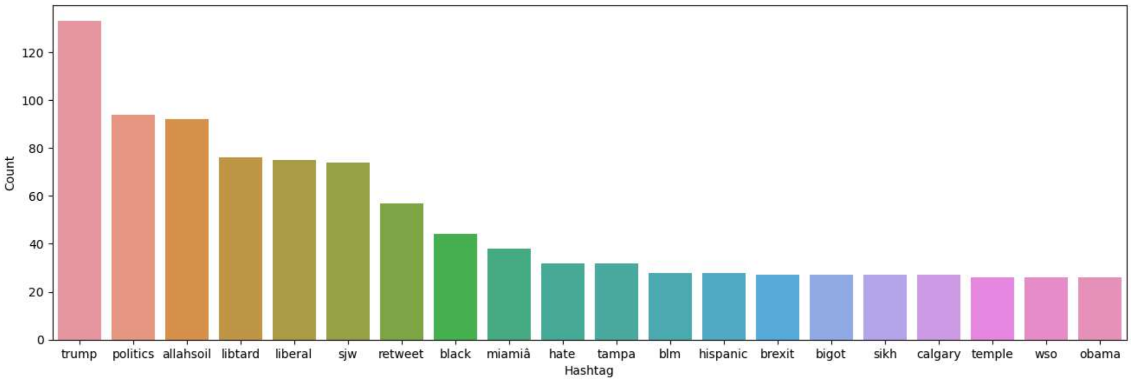
In [20]:

```
a = nltk.FreqDist(HT_regular)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})

# selecting top 20 most frequent hashtags
d = d.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```



In [21]:

```
a = nltk.FreqDist(HT_negative)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})

# selecting top 20 most frequent hashtags
d = d.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```



In [22]:

```
# tokenizing the words present in the training set
tokenized_tweet = train['tweet'].apply(lambda x: x.split())

# importing gensim
import gensim

# creating a word to vector model
model_w2v = gensim.models.Word2Vec(
            tokenized_tweet,
            vector_size=200, # desired no. of features/independent varia
            window=5, # context window size
            min_count=2,
            sg = 1, # 1 for skip-gram model
            hs = 0,
            negative = 10, # for negative sampling
            workers= 2, # no.of cores
            seed = 34)

model_w2v.train(tokenized_tweet, total_examples= len(train['tweet']), ep


# !pip install gensim
```

Out[22]:

```
(6109226, 8411580)
```

```
In [23]: model_w2v.wv.most_similar(positive = "dinner")
```

```
Out[23]: [('spaghetti', 0.6443722248077393),
          ('#prosecco', 0.5974183082580566),
          ('#boardgames', 0.5951496958732605),
          ('7!', 0.582740843296051),
          ('shopping!', 0.582472026348114),
          ('enroute', 0.5815898776054382),
          ('podium', 0.5800884366035461),
          ('sister!!', 0.5694329142570496),
          ('#wanderlust', 0.5684548616409302),
          ('fluffy', 0.5654929876327515)]
```

```
In [24]: model_w2v.wv.most_similar(positive = "cancer")
```

```
Out[24]: [('champion,', 0.7116358876228333),
          ('#merica', 0.6990123987197876),
          ('tolerance', 0.699009895324707),
          ('absurd.', 0.6967269778251648),
          ('level.', 0.691489040851593),
          ('clubs', 0.69086092710495),
          ('speeches', 0.6872279047966003),
          ('tragedies', 0.683964729309082),
          ('ownership', 0.682731568813324),
          ('#prayfororlandoâ\x80¦', 0.6822460889816284)]
```

```
In [25]: model_w2v.wv.most_similar(positive = "apple")
```

```
Out[25]: [('"mytraining"', 0.7063566446304321),
          ('mytraining', 0.7032891511917114),
          ('training"', 0.6900411248207092),
          ('app,', 0.6482236981391907),
          ('"my', 0.6160722374916077),
          ('ios', 0.6056389212608337),
          ('app', 0.5827821493148804),
          ('humans.', 0.581159770488739),
          ('mp3', 0.5621538758277893),
          ('ta', 0.5579870939254761)]
```

```
In [26]: model_w2v.wv.most_similar(negative = "hate")
```

```
Out[26]: [('#foodie', 0.03966078534722328),
          ('@', 0.038726236671209335),
          ('#babies', 0.0276285782456398),
          ('#hungry', 0.023657528683543205),
          ('ð\x9f\x8e\x93', 0.011807414703071117),
          ('#relax', 0.00909979734569788),
          ('â\x99¡', 0.008922450244426727),
          ('â\x9c\x88ï¸\x8f', 0.004844261333346367),
          ('#wine', 0.0035755163989961147),
          ('board', 0.003491251962259412)]
```

In [27]:
```
# from gensim.models.deprecated.doc2vec import LabeledSentence

from tqdm import tqdm
tqdm.pandas(desc="progress-bar")
from gensim.models.doc2vec import TaggedDocument
```

In [28]:
```
def add_label(twt):
    output = []
    for i, s in zip(twt.index, twt):
        output.append(TaggedDocument(s, ["tweet_" + str(i)]))
    return output

# label all the tweets
labeled_tweets = add_label(tokenized_tweet)

labeled_tweets[:6]
```

Out[28]:
```
[TaggedDocument(words=['@user', 'when', 'a', 'father', 'is', 'dysfuncti
onal', 'and', 'is', 'so', 'selfish', 'he', 'drags', 'his', 'kids', 'int
o', 'his', 'dysfunction.', '#run'], tags=['tweet_0']),
 TaggedDocument(words=['@user', '@user', 'thanks', 'for', '#lyft', 'cre
dit', 'i', "can't", 'use', 'cause', 'they', "don't", 'offer', 'wheelcha
ir', 'vans', 'in', 'pdx.', '#disapointed', '#getthanked'], tags=['tweet
_1']),
 TaggedDocument(words=['bihday', 'your', 'majesty'], tags=['tweet_2']),
 TaggedDocument(words=['#model', 'i', 'love', 'u', 'take', 'with', 'u',
'all', 'the', 'time', 'in', 'urð\x9f\x93±!!!', 'ð\x9f\x98\x99ð\x9f\x98
\x8eð\x9f\x91\x84ð\x9f\x91', 'ð\x9f\x92¦ð\x9f\x92¦ð\x9f\x92¦'], tags=
['tweet_3']),
 TaggedDocument(words=['factsguide:', 'society', 'now', '#motivation'],
tags=['tweet_4']),
 TaggedDocument(words=['[2/2]', 'huge', 'fan', 'fare', 'and', 'big', 't
alking', 'before', 'they', 'leave.', 'chaos', 'and', 'pay', 'disputes',
'when', 'they', 'get', 'there.', '#allshowandnogo'], tags=['tweet_5'])]
```

In [29]:
```
# # removing unwanted patterns from the data

# import re
# import nltk

# nltk.download('stopwords')
# from nltk.corpus import stopwords
# from nltk.stem.porter import PorterStemmer
```

In [30]:

```
train_corpus = []

for i in range(0, 31962):
  review = re.sub('[^a-zA-Z]', ' ', train['tweet'][i])
  review = review.lower()
  review = review.split()

  ps = PorterStemmer()

  # stemming
  review = [ps.stem(word) for word in review if not word in set(stopword

  # joining them back with space
  review = ' '.join(review)
  train_corpus.append(review)
```

In [31]:

```
test_corpus = []

for i in range(0, 17197):
  review = re.sub('[^a-zA-Z]', ' ', test['tweet'][i])
  review = review.lower()
  review = review.split()

  ps = PorterStemmer()

  # stemming
  review = [ps.stem(word) for word in review if not word in set(stopword

  # joining them back with space
  review = ' '.join(review)
  test_corpus.append(review)
```

In [32]:

```
# creating bag of words

from sklearn.feature_extraction.text import CountVectorizer

cv  = CountVectorizer(max_features = 2500)
x = cv.fit_transform(train_corpus).toarray()
y = train.iloc[:, 1]

print(x.shape)
print(y.shape)
```

```
(31962, 2500)
(31962,)
```

In [33]:

```
# creating bag of words

from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features = 2500)
x_test = cv.fit_transform(test_corpus).toarray()

print(x_test.shape)
```

```
(17197, 2500)
```

In [34]:

```
# splitting the training data into train and valid sets

from sklearn.model_selection import train_test_split

x_train, x_valid, y_train, y_valid = train_test_split(x, y, test_size =

print(x_train.shape)
print(x_valid.shape)
print(y_train.shape)
print(y_valid.shape)
```

```
(23971, 2500)
(7991, 2500)
(23971,)
(7991,)
```

In [35]:

```
# standardization

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_train = sc.fit_transform(x_train)
x_valid = sc.transform(x_valid)
x_test = sc.transform(x_test)
```

In [36]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score

model = RandomForestClassifier()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("F1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

```
Training Accuracy : 0.999123941429227
Validation Accuracy : 0.9515705168314353
F1 score : 0.6071065989847715
[[7305  127]
 [ 260  299]]
```

In [37]:

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("f1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\linear_model\_logist
ic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression (https://scikit-learn.org/stable/modules/linear_model.html#l
ogistic-regression)
  n_iter_i = _check_optimize_result(
```

```
Training Accuracy : 0.9851487213716574
Validation Accuracy : 0.9416843949443123
f1 score : 0.5933682373472949
[[7185  247]
 [ 219  340]]
```

In [38]:

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("f1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

```
Training Accuracy : 0.9991656585040257
Validation Accuracy : 0.9334251032411462
f1 score : 0.5413793103448276
[[7145  287]
 [ 245  314]]
```

In [39]:

```
from sklearn.svm import SVC

model = SVC()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("f1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

```
Training Accuracy : 0.978181969880272
Validation Accuracy : 0.9521962207483419
f1 score : 0.4986876640419947
[[7419   13]
 [ 369  190]]
```

In [40]:

```
from xgboost import XGBClassifier

model = XGBClassifier()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("f1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

```
Training Accuracy : 0.9608693838388053
Validation Accuracy : 0.9550744587661119
f1 score : 0.575147928994083
[[7389   43]
 [ 316  243]]
```