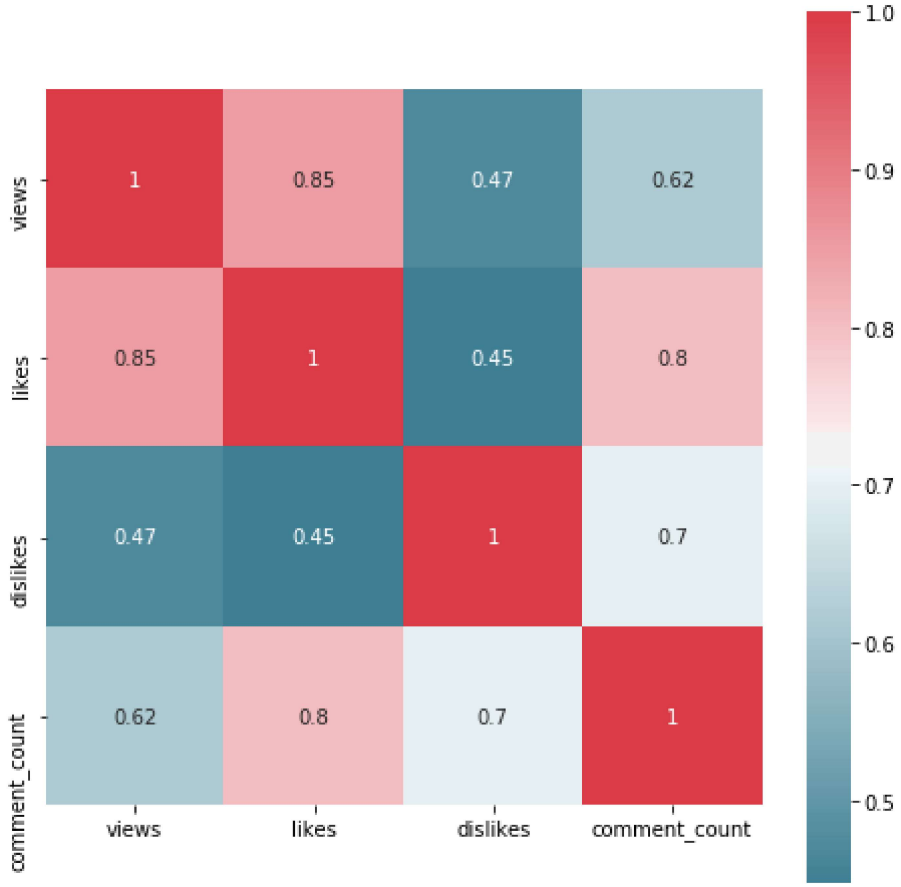
In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df_usa=pd.read_csv("D:/projects/Sentiment_Analysis(YouTube_Dataset)(NLP_WordCloud_ TextBlob)/USvideos.csv")
df_ca=pd.read_csv("D:/projects/Sentiment_Analysis(YouTube_Dataset)(NLP_WordCloud_ TextBlob)/CAvideos.csv")
df_de=pd.read_csv("D:/projects/Sentiment_Analysis(YouTube_Dataset)(NLP_WordCloud_ TextBlob)/DEvideos.csv")
df_fr=pd.read_csv("D:/projects/Sentiment_Analysis(YouTube_Dataset)(NLP_WordCloud_ TextBlob)/FRvideos.csv")
df_gb=pd.read_csv("D:/projects/Sentiment_Analysis(YouTube_Dataset)(NLP_WordCloud_ TextBlob)/GBvideos.csv")
```

In [3]:
```
#In the dataset, the Trending Date and Published Time are not in the Unix date-time format. Let's fix this first.
df_usa['trending_date'] = pd.to_datetime(df_usa['trending_date'], format='%y.%d.%m')
df_usa['publish_time'] = pd.to_datetime(df_usa['publish_time'], format='%Y-%m-%dT%H:%M:%S.%fZ')

# separates date and time into two columns from 'publish_time' column

df_usa.insert(4, 'publish_date', df_usa['publish_time'].dt.date)#Series.dt.dateReturns (namely, the date part of Timestamps without timezone information).
df_usa['publish_time'] = df_usa['publish_time'].dt.time
df_usa['publish_date']=pd.to_datetime(df_usa['publish_date']) #pandas.to_datetime Convert argument to datetime.
```

In [4]:
```
#To see the correlation between the likes, dislikes, comments, and views lets
 plot a correlation matrix
columns_show=['views', 'likes', 'dislikes', 'comment_count']
f, ax = plt.subplots(figsize=(8, 8))
corr = df_usa[columns_show].corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_
palette(220, 10, as_cmap=True),
            square=True, ax=ax,annot=True)#seaborn.heatmap Plot rectangular da
ta as a color-encoded matrix.  draw the heatmap

#In the above matrix, for USA dataset, the columns with :-
#    High Correlation - Views and Likes,, Comment_count and Dislikes
#    Medium Correlation - Views and Dislikes, Views and Comment_Count, Likes a
nd Comment_Count
#    Low Correlation - Likes and Dislike
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x10772210>



In [5]:
```
#Since, a video could be in trending for several days. There might be multiple
 rows of a particular video.
#In order to calculate the total Views, Comments, Likes, Dislikes of a video,
 we need to groupby with video_id.
#The below script will give you the total no. of views/comments/likes, and dis
likes of a video.

usa_video_views=df_usa.groupby(['video_id'])['views'].agg('sum')
usa_video_likes=df_usa.groupby(['video_id'])['likes'].agg('sum')
usa_video_dislikes=df_usa.groupby(['video_id'])['dislikes'].agg('sum')
usa_video_comment_count=df_usa.groupby(['video_id'])['comment_count'].agg('su
m')
```

In [6]:
```
#To get the numbers of videos on which the 'Comments Disabled/ Rating Disable
d/Video Error'.
#We need to remove the duplicates to get the correct numbers otherwise there w
ill be redundancy.

df_usa_single_day_trend=df_usa.drop_duplicates(subset='video_id', keep=False,
inplace=False)#implace-works on orignal data(faster)
df_usa_multiple_day_trend= df_usa.drop_duplicates(subset='video_id',keep='firs
t',inplace=False)#since multiple day(keep=first)

frames = [df_usa_single_day_trend, df_usa_multiple_day_trend]
df_usa_without_duplicates=pd.concat(frames)

#DataFrame.describe: Generates descriptive statistics that summarize the centr
al tendency,
#dispersion and shape of a dataset’s distribution, excluding NaN values.
df_usa_comment_disabled=df_usa_without_duplicates[df_usa_without_duplicates['c
omments_disabled']==True].describe()
df_usa_rating_disabled=df_usa_without_duplicates[df_usa_without_duplicates['ra
tings_disabled']==True].describe()
df_usa_video_error=df_usa_without_duplicates[df_usa_without_duplicates['video_
error_or_removed']==True].describe()
```

In [7]:
```
#How many videos were trended only for a single day?

df_usa_single_day_trend.head()#DataFrame.head(n=5) Return the first n rows.
```

Out[7]:

| | video_id | trending_date | title | channel_title | publish_date | category_id | |
|---|---|---|---|---|---|---|---|
| **10** | 9wRQljFNDW8 | 2017-11-14 | Dion Lewis' 103-Yd Kick Return TD vs. Denver! ... | NFL | 2017-11-13 | 17 | 0 |
| **36** | Om_zGhJLZ5U | 2017-11-14 | TL;DW - Every DCEU Movie Before Justice League | Screen Junkies | 2017-11-12 | 1 | 1 |
| **41** | goP4Z5wyOIM | 2017-11-14 | Iraq-Iran earthquake: Deadly tremor hits borde... | BBC News | 2017-11-12 | 25 | 2 |
| **55** | 8NHA23f7LvU | 2017-11-14 | Jason Momoa Wows Hugh Grant With Some Dothraki... | The Graham Norton Show | 2017-11-10 | 24 | 1 |
| **76** | IE-xepGLVt8 | 2017-11-14 | Mayo Clinic's first face transplant patient me... | Mayo Clinic | 2017-11-10 | 28 | 1 |

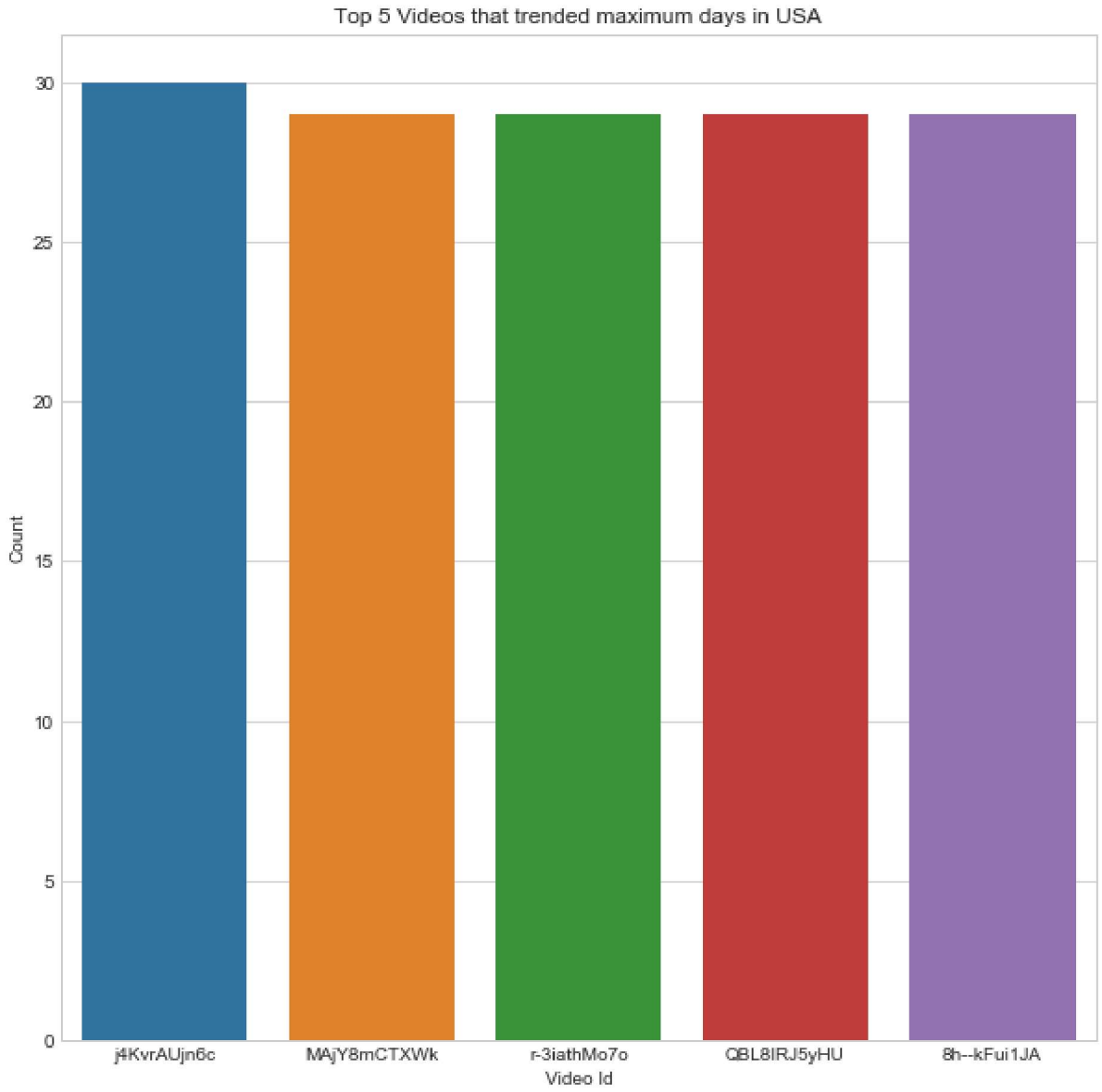
In [8]:
```
#Videos that were trended for more than 1 day

df_usa_multiple_day_trend.head()
```

Out[8]:

| | video_id | trending_date | title | channel_title | publish_date | category_i |
|---|---|---|---|---|---|---|
| **0** | 2kyS6SvSYSE | 2017-11-14 | WE WANT TO TALK ABOUT OUR MARRIAGE | CaseyNeistat | 2017-11-13 | 22 |
| **1** | 1ZAPwfrtAFY | 2017-11-14 | The Trump Presidency: Last Week Tonight with J... | LastWeekTonight | 2017-11-13 | 24 |
| **2** | 5qpjK5DgCt4 | 2017-11-14 | Racist Superman \| Rudy Mancuso, King Bach & Le... | Rudy Mancuso | 2017-11-12 | 23 |
| **3** | puqaWrEC7tY | 2017-11-14 | Nickelback Lyrics: Real or Fake? | Good Mythical Morning | 2017-11-13 | 24 |
| **4** | d380meD0W0M | 2017-11-14 | I Dare You: GOING BALD!? | nigahiga | 2017-11-12 | 24 |

In [9]:
```
#Which video trended on maximum days and what is the title, likes, dislikes, comments, and views.

df_usa_which_video_trended_maximum_days=df_usa.groupby(by=['video_id'],as_index=False).count().sort_values(by='title',ascending=False).head()

plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.barplot(x=df_usa_which_video_trended_maximum_days['video_id'],y=df_usa_which_video_trended_maximum_days['trending_date'], data=df_usa_which_video_trended_maximum_days)
plt.xlabel("Video Id")
plt.ylabel("Count")
plt.title("Top 5 Videos that trended maximum days in USA")
```

Out[9]: <matplotlib.text.Text at 0xfc820b0>

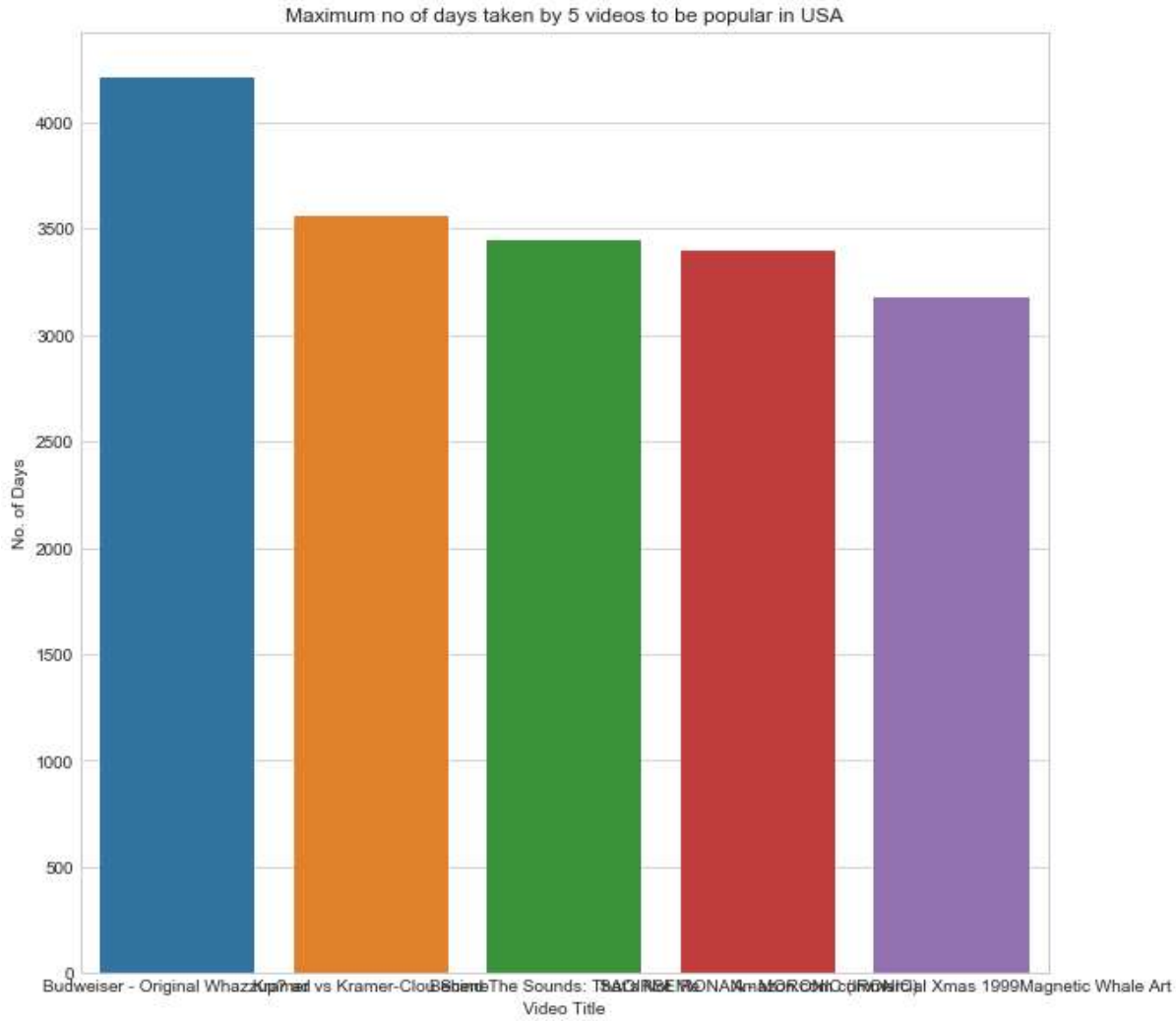


In [10]:
```
#Video which were trended for maximum days
#The maximum no. of days a video trended is 14 i.e. for 'sXP6vliZIHI' video i
d.
#Now, the below script gives its likes,dislikes, views, and comments.

df_usa_maximum_views=usa_video_views['sXP6vliZIHI']
df_usa_maximum_likes=usa_video_likes['sXP6vliZIHI']
df_usa_maximum_dislikes=usa_video_dislikes['sXP6vliZIHI']
df_usa_maximum_comment=usa_video_comment_count['sXP6vliZIHI']
```

In [11]:
```
#Which video took maximum no of days to be a Trending Video-
#Now, the below script will calculate the no of days taken by a video to be a
 Trending Video.
#The graph will show the top 5 videos that took maximum no. of days to be a tr
ending video.

df_usa_multiple_day_trend['Days_taken_to_be_trending_video'] =df_usa_multiple_
day_trend['trending_date'] - df_usa_multiple_day_trend['publish_date']

#You can convert it to a timedelta with a day precision. To extract the intege
r value of days you divide it with a timedelta of one day.
df_usa_multiple_day_trend['Days_taken_to_be_trending_video']= df_usa_multiple_
day_trend['Days_taken_to_be_trending_video'] / np.timedelta64(1, 'D')
usa_no_of_days_take_trend=df_usa_multiple_day_trend.sort_values(by='Days_taken
_to_be_trending_video',ascending=False).head(5)

plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.barplot(x=usa_no_of_days_take_trend['title'],y=usa_no_of_days_take_tr
end['Days_taken_to_be_trending_video'], data=usa_no_of_days_take_trend)
plt.xlabel("Video Title")
plt.ylabel("No. of Days")
plt.title("Maximum no of days taken by 5 videos to be popular in USA")
```

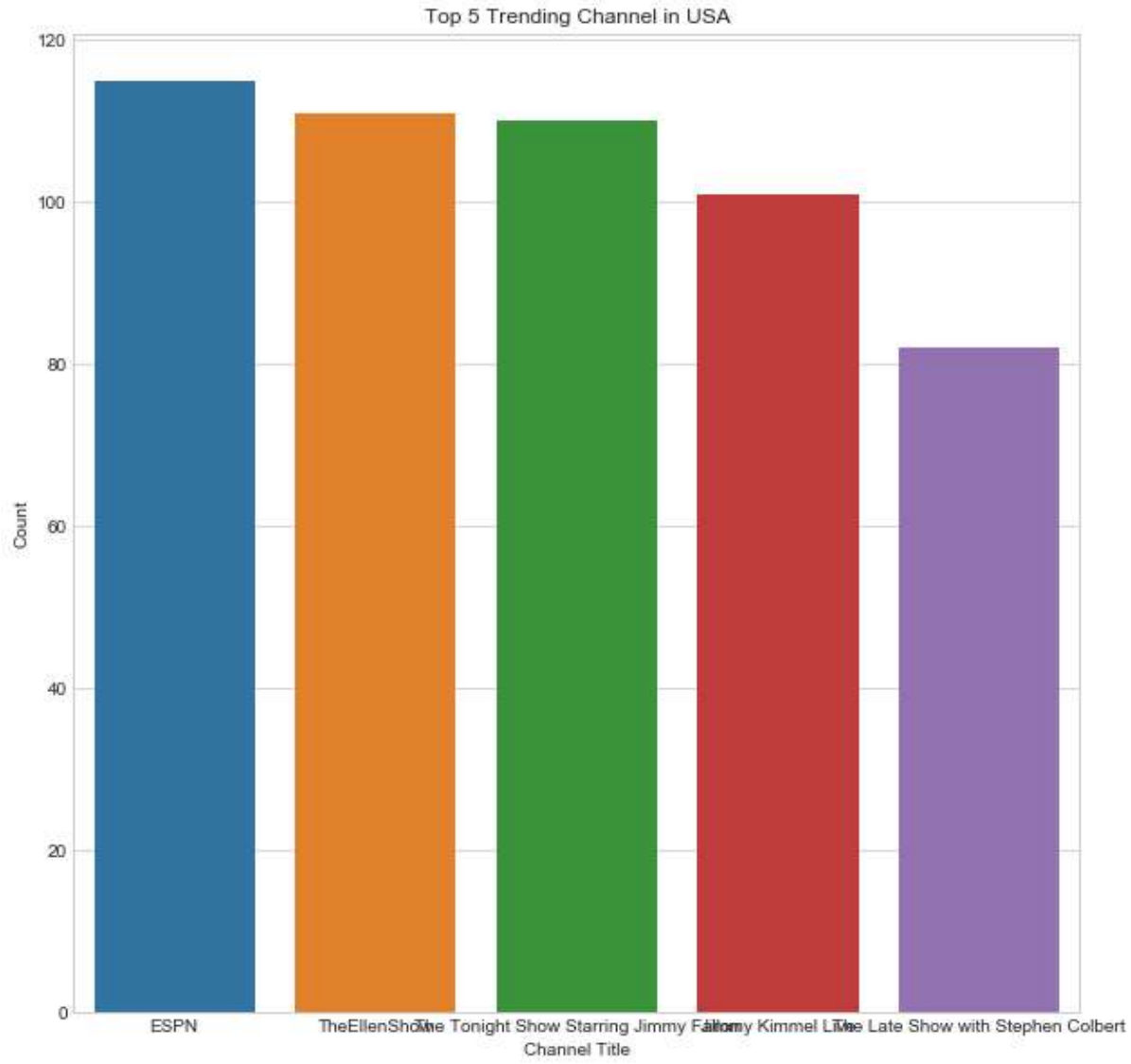
```
c:\users\ankur\appdata\local\programs\python\python36-32\lib\site-packages\ip
ykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  """
c:\users\ankur\appdata\local\programs\python\python36-32\lib\site-packages\ip
ykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
```

Out[11]: <matplotlib.text.Text at 0x106e0150>



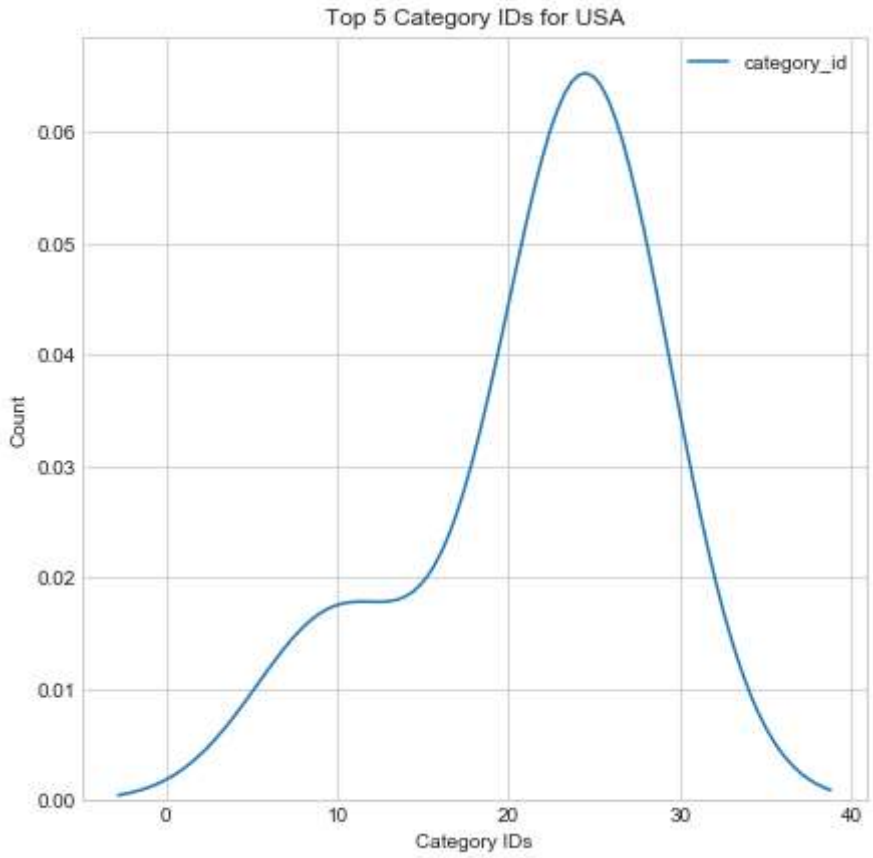
In [12]:
```
#Top 5 Trending Channel in USA

usa_trending_channel=df_usa_without_duplicates.groupby(by=['channel_title'],as
_index=False).count().sort_values(by='title',ascending=False).head()
#as_index can be set to true or false depending on if you want the column by w
hich you grouped to be the index of the output

plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.barplot(x=usa_trending_channel['channel_title'],y=usa_trending_channe
l['video_id'], data=usa_trending_channel)
plt.xlabel("Channel Title")
plt.ylabel("Count")
plt.title("Top 5 Trending Channel in USA")
```

Out[12]: <matplotlib.text.Text at 0x10a03930>



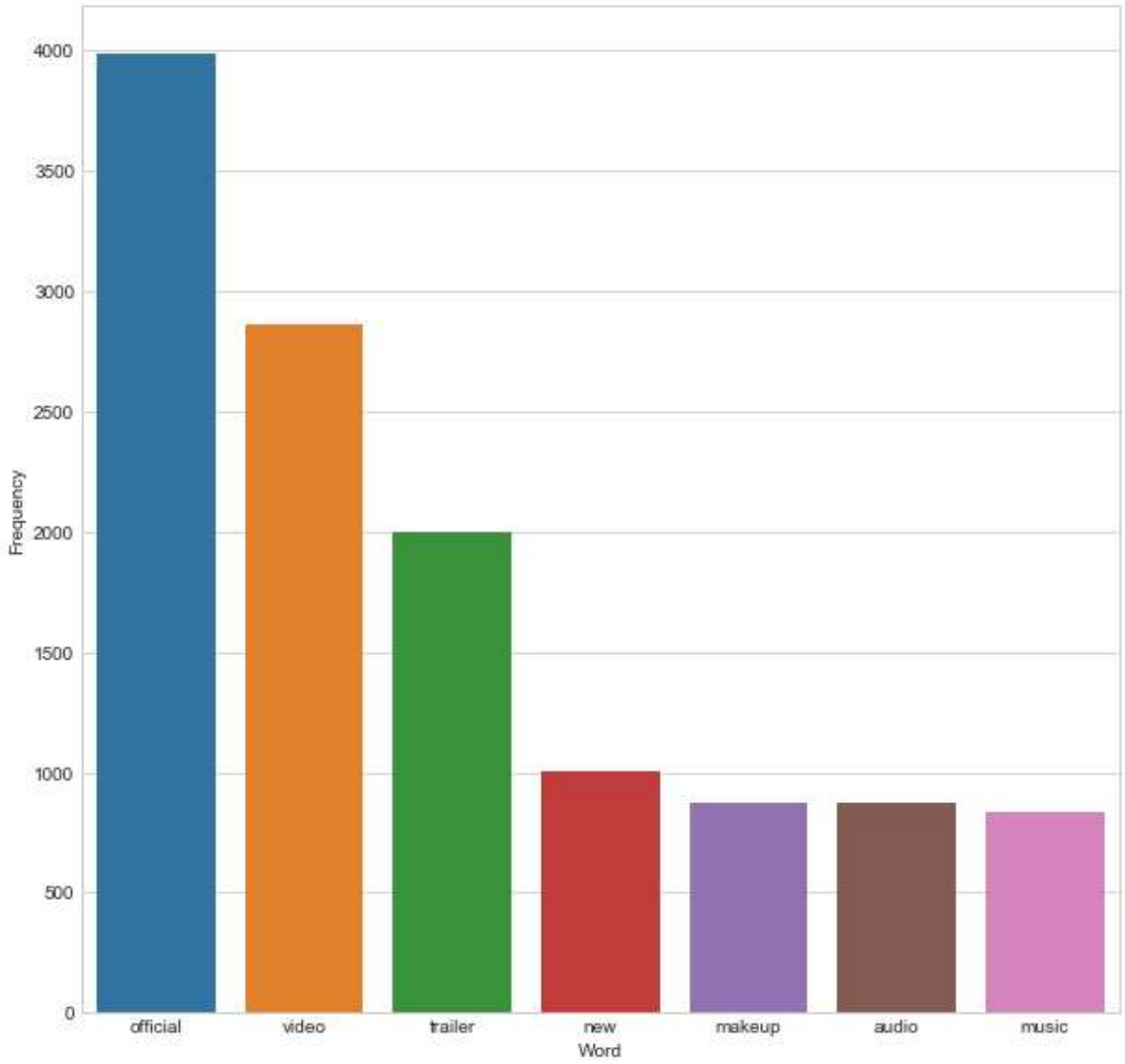
In [13]:
```
#Top 5 USA_Category_IDs

usa_category_id=df_usa_without_duplicates.groupby(by=['category_id'],as_index=
False).count().sort_values(by='title',ascending=False).head(5)

plt.figure(figsize=(7,7))
sns.kdeplot(usa_category_id['category_id']);
plt.xlabel("Category IDs")
plt.ylabel("Count")
plt.title("Top 5 Category IDs for USA")
```

Out[13]: <matplotlib.text.Text at 0x10a79290>



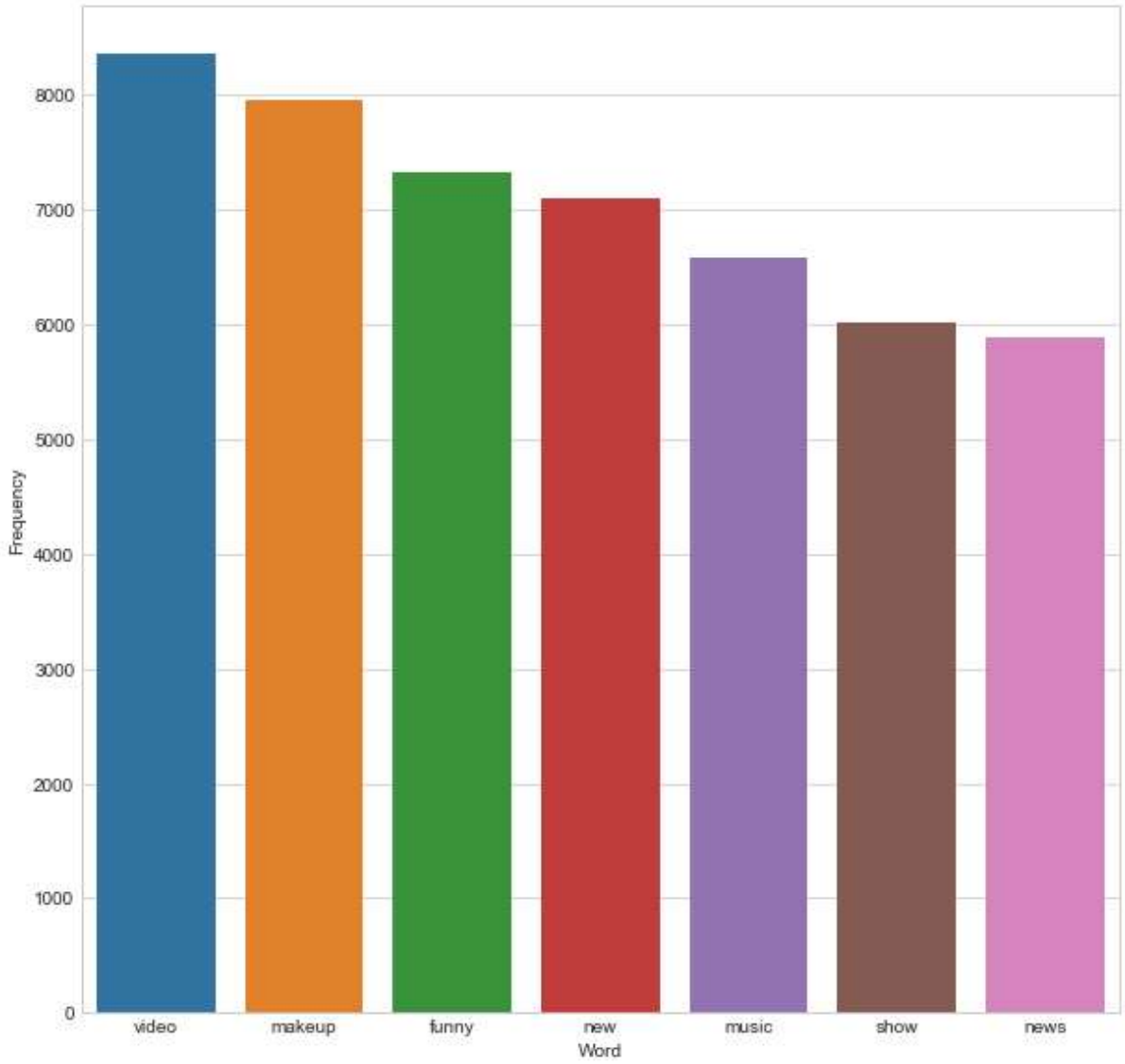

In [14]:
```
#Defining function for wordcloud

from wordcloud import WordCloud
import nltk
from nltk.corpus import stopwords
from nltk import sent_tokenize, word_tokenize
from wordcloud import WordCloud, STOPWORDS

def wc(data,bgcolor,title):
    plt.figure(figsize = (100,100))
    wc = WordCloud(background_color = bgcolor, max_words = 1000,  max_font_siz
e = 50)
    wc.generate(' '.join(data))
    plt.imshow(wc)
    plt.axis('off')
```

In [15]:
```
#To Count the frequency of words in Title column.

from collections import Counter
from nltk.tokenize import RegexpTokenizer
from stop_words import get_stop_words
import re

top_N = 100
#convert list of list into text
#a=''.join(str(r) for v in df_usa['title'] for r in v)

a = df_usa['title'].str.lower().str.cat(sep=' ')

# removes punctuation,numbers and returns list of words
b = re.sub('[^A-Za-z]+', ' ', a)

#remove all the stopwords from the text
stop_words = list(get_stop_words('en'))
nltk_words = list(stopwords.words('english'))
stop_words.extend(nltk_words)

word_tokens = word_tokenize(b)
filtered_sentence = [w for w in word_tokens if not w in stop_words]
filtered_sentence = []
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

# Remove characters which have length less than 2
without_single_chr = [word for word in filtered_sentence if len(word) > 2]

# Remove numbers
cleaned_data_title = [word for word in without_single_chr if not word.isnumeri
c()]

# Calculate frequency distribution
word_dist = nltk.FreqDist(cleaned_data_title)
rslt = pd.DataFrame(word_dist.most_common(top_N),
                    columns=['Word', 'Frequency'])

plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.barplot(x="Word",y="Frequency", data=rslt.head(7))
```

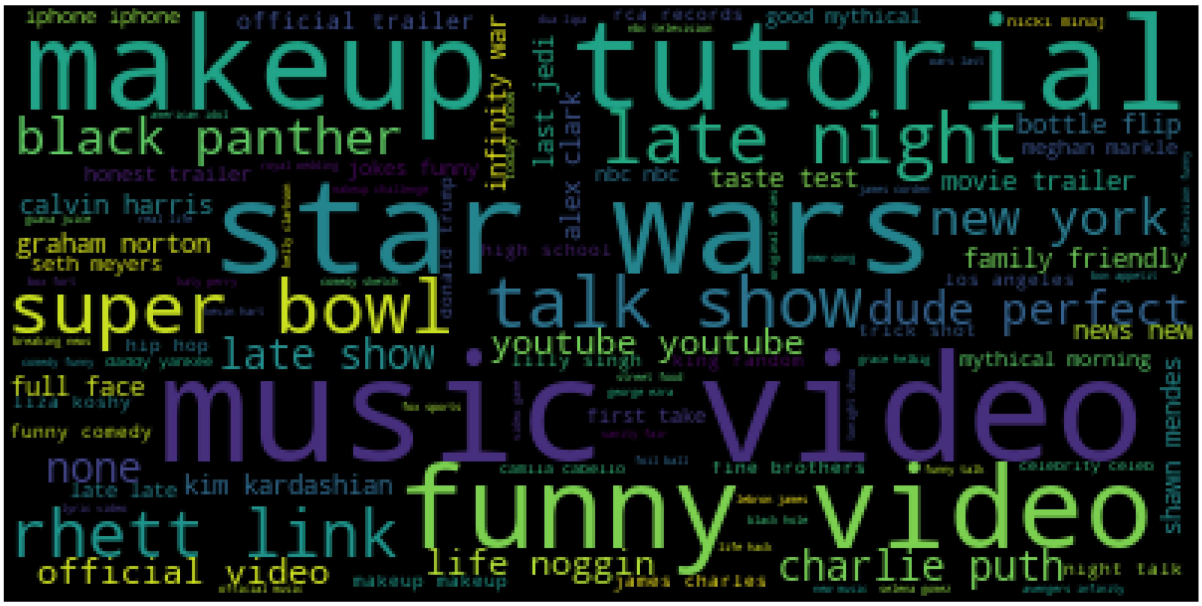



In [16]:
```
#WordCloud for Title Column

wc(cleaned_data_title,'black','Common Words' )
```

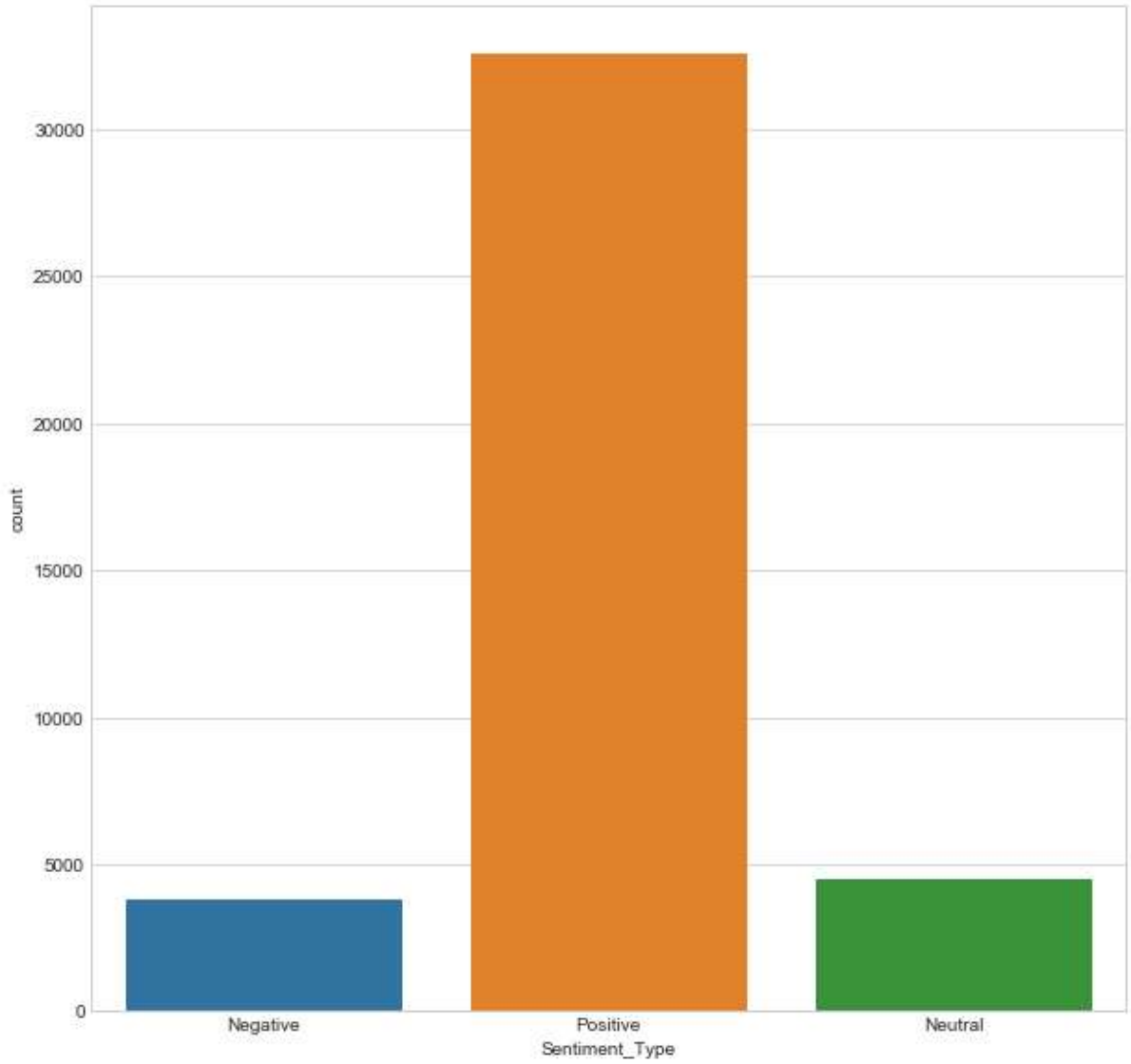
In [17]:
```
#To Count the frequency of words in Tags column.

from collections import Counter
from nltk.tokenize import RegexpTokenizer
from stop_words import get_stop_words
import re

top_N = 100
#convert list of list into text
#a=''.join(str(r) for v in df_usa['title'] for r in v)

tags_lower = df_usa['tags'].str.lower().str.cat(sep=' ')

# removes punctuation,numbers and returns list of words
tags_remove_pun = re.sub('[^A-Za-z]+', ' ', tags_lower)

#remove all the stopwords from the text
stop_words = list(get_stop_words('en'))
nltk_words = list(stopwords.words('english'))
stop_words.extend(nltk_words)

word_tokens_tags = word_tokenize(tags_remove_pun)
filtered_sentence_tags = [w_tags for w_tags in word_tokens_tags if not w_tags
in stop_words]
filtered_sentence_tags = []
for w_tags in word_tokens_tags:
    if w_tags not in stop_words:
        filtered_sentence_tags.append(w_tags)

# Remove characters which have length less than 2
without_single_chr_tags = [word_tags for word_tags in filtered_sentence_tags i
f len(word_tags) > 2]

# Remove numbers
cleaned_data_tags = [word_tags for word_tags in without_single_chr_tags if not
 word_tags.isnumeric()]

# Calculate frequency distribution
word_dist_tags = nltk.FreqDist(cleaned_data_tags)
rslt_tags = pd.DataFrame(word_dist_tags.most_common(top_N),
                    columns=['Word', 'Frequency'])

plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.barplot(x="Word",y="Frequency", data=rslt_tags.head(7))
```



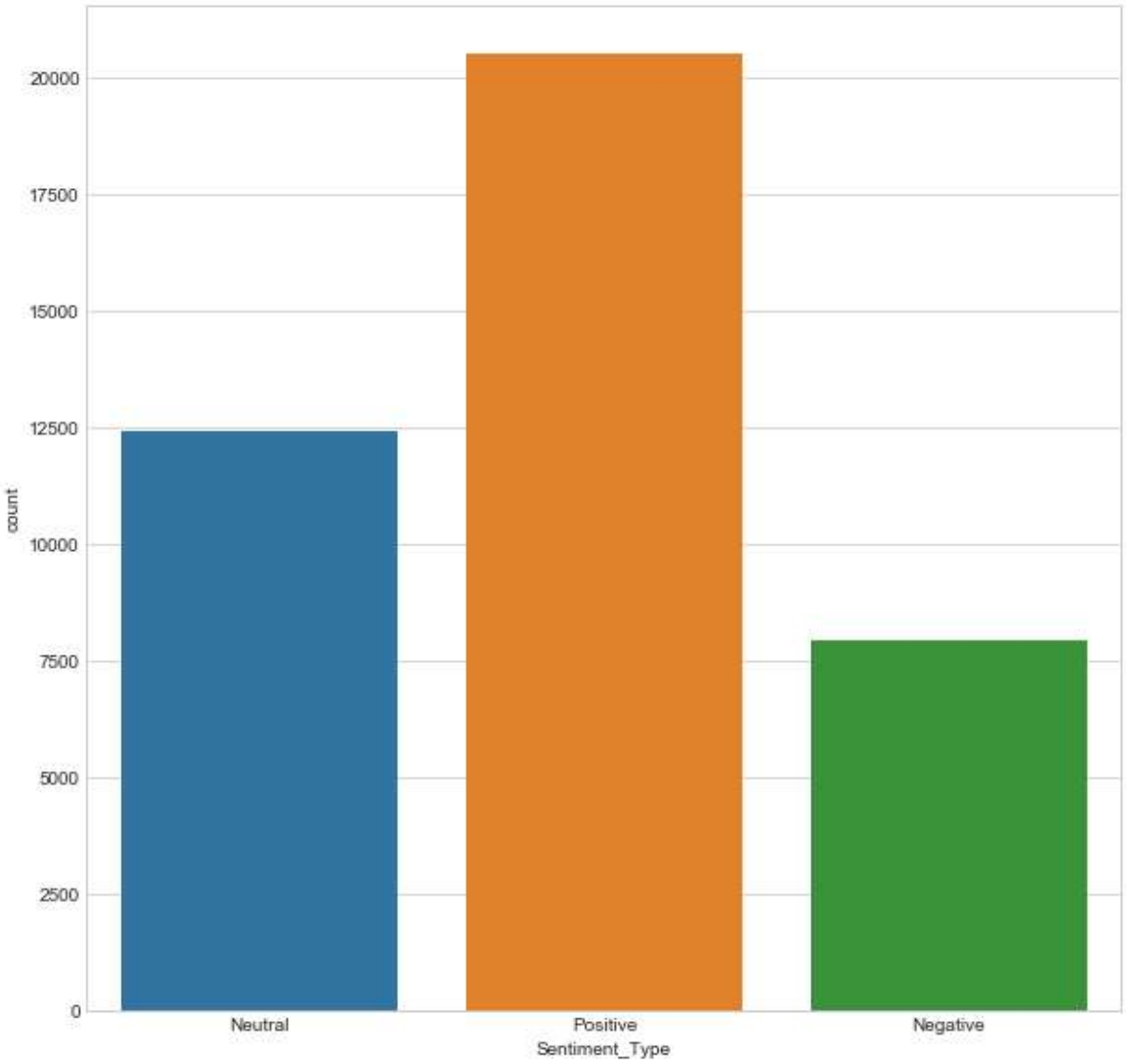

In [18]:
```
#WordCloud for Tags

wc(cleaned_data_tags,'black','Common Words' )
```



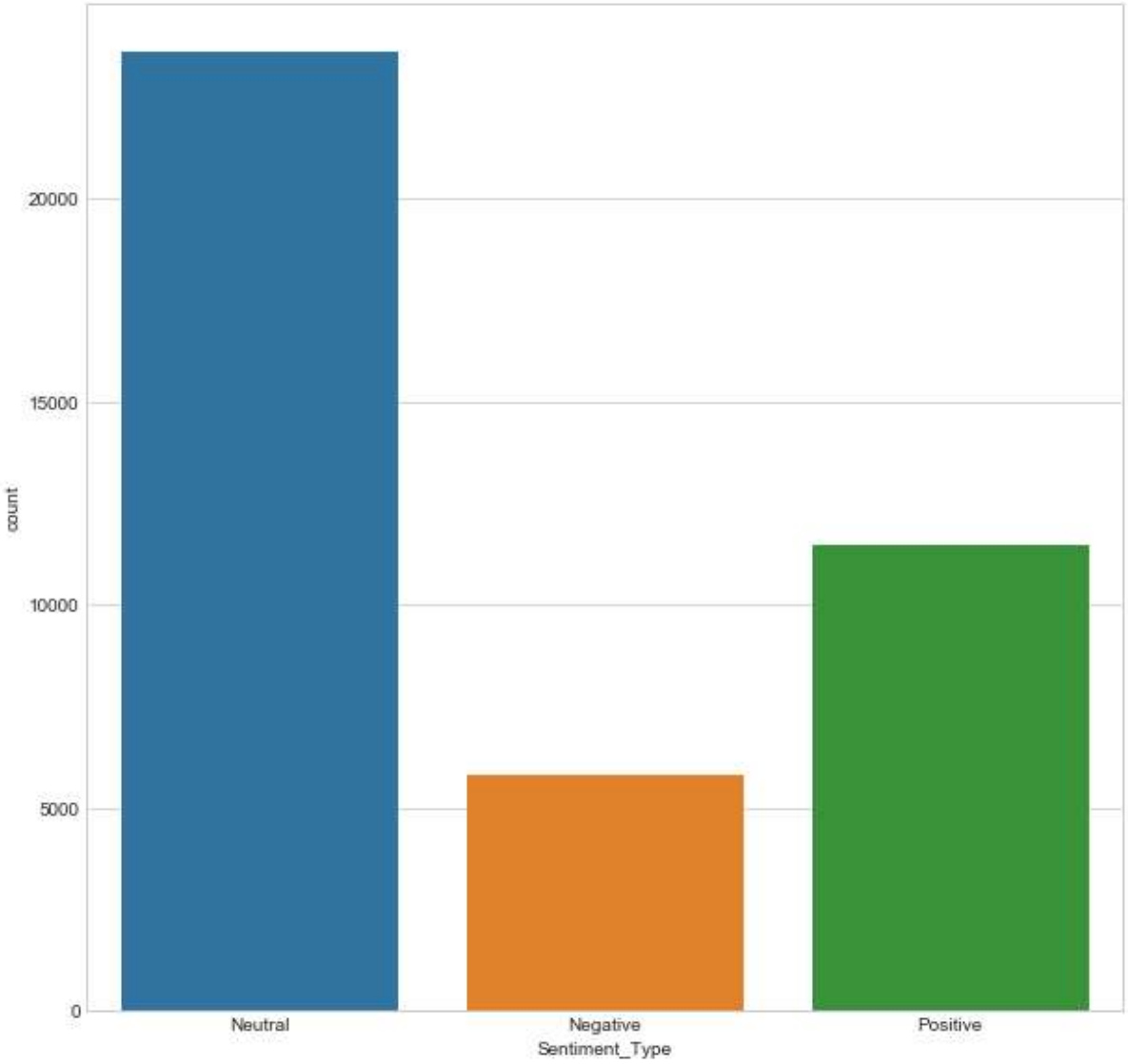
In [19]:
```
#Categorize the Description column into Positive and Negative sentiments using
 TextBlob
#The sentiment function of textblob returns two properties, polarity, and subj
ectivity.
#Polarity is float which lies in the range of [-1,1] where 1 means positive st
atement and -1 means a negative statement.
#Subjective sentences generally refer to personal opinion, emotion or judgment
 whereas objective refers to factual information.
#Subjectivity is also a float which lies in the range of [0,1].

from textblob import TextBlob

bloblist_desc = list()

df_usa_descr_str=df_usa['description'].astype(str)#When converting a pandas Se
ries object to type string using astype(str)
for row in df_usa_descr_str:
    blob = TextBlob(row)#helps to tokenize the words
    bloblist_desc.append((row,blob.sentiment.polarity, blob.sentiment.subjecti
vity))
    df_usa_polarity_desc = pd.DataFrame(bloblist_desc, columns = ['sentence',
'sentiment','polarity'])

def f(df_usa_polarity_desc):
    if df_usa_polarity_desc['sentiment'] > 0:
        val = "Positive"
    elif df_usa_polarity_desc['sentiment'] == 0:
        val = "Neutral"
    else:
        val = "Negative"
    return val

df_usa_polarity_desc['Sentiment_Type'] = df_usa_polarity_desc.apply(f, axis=1)

plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.countplot(x="Sentiment_Type", data=df_usa_polarity_desc)
```


30000
25000
20000
15000
10000
5000
0
count
Negative
Positive
Neutral
Sentiment_Type

In [20]:
```
#Categorize the Tags column into Positive and Negative sentiments using TextBl
ob

from textblob import TextBlob

bloblist_tags = list()

df_usa_tags_str=df_usa['tags']
for row in df_usa_tags_str:
    blob = TextBlob(row)#helps to tokenize the words
    bloblist_tags.append((row,blob.sentiment.polarity, blob.sentiment.subjecti
vity))
    df_usa_polarity_tags = pd.DataFrame(bloblist_tags, columns = ['sentence',
'sentiment','polarity'])

def f_tags(df_usa_polarity_tags):
    if df_usa_polarity_tags['sentiment'] > 0:
        val = "Positive"
    elif df_usa_polarity_tags['sentiment'] == 0:
        val = "Neutral"
    else:
        val = "Negative"
    return val

df_usa_polarity_tags['Sentiment_Type'] = df_usa_polarity_tags.apply(f_tags, ax
is=1)

plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.countplot(x="Sentiment_Type", data=df_usa_polarity_tags)
```


20000
17500
15000
12500
10000
7500
5000
2500
0
count
Neutral
Positive
Negative
Sentiment_Type

In [21]:
```
#Categorize the Title column into Positive and Negative sentiments using TextB
lob

from textblob import TextBlob

bloblist_title = list()

df_usa_title_str=df_usa['title']
for row in df_usa_title_str:
    blob = TextBlob(row)#helps to tokenize the words
    bloblist_title.append((row,blob.sentiment.polarity, blob.sentiment.subject
ivity))
    df_usa_polarity_title = pd.DataFrame(bloblist_title, columns = ['sentence'
,'sentiment','polarity'])

def f_title(df_usa_polarity_title):
    if df_usa_polarity_title['sentiment'] > 0:
        val = "Positive"
    elif df_usa_polarity_title['sentiment'] == 0:
        val = "Neutral"
    else:
        val = "Negative"
    return val

df_usa_polarity_title['Sentiment_Type'] = df_usa_polarity_title.apply(f_title,
 axis=1)

plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.countplot(x="Sentiment_Type", data=df_usa_polarity_title)
```


20000
15000
10000
5000
0
count
Neutral
Negative
Positive
Sentiment_Type