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

import nltk
import re
nltk.download('stopwords')
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword=set(stopwords.words('english'))

from nltk.sentiment.vader import SentimentIntensityAnalyzer
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator


from nltk.stem import PorterStemmer, WordNetLemmatizer, porter

nltk.download('punkt')
```

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

Out[1]: True

In [2]:
```
df = pd.read_csv('IMDb Movies India(1).csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Acto
0		NaN	NaN	Drama	NaN	NaN	J.S. Randhawa	Manmauji	Birbal	Rajen Bha
1	#Gadhvi (He thought he was Gandhi)	-2019.0	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arv Jar
2	#Homecoming	-2021.0	90 min	Drama, Musical	NaN	NaN	Soumyajit Majumdar	Sayani Gupta	Plabita Borthakur	Angs
3	#Yaaram	-2019.0	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddh Kap
4	...And Once Again	-2010.0	105 min	Drama	NaN	NaN	Amol Palekar	Rajat Kapoor	Rituparna Sengupta	Ant

In [4]:
```
df.tail()
```

In [4]:
```
df.tail()
```

Out[4]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
15504	Zulm Ko Jala Doonga	-1988.0	NaN	Action	4.6	11	Mahendra Shah	Naseeruddin Shah	Sumeet Saigal	Suparna Anand
15505	Zulmi	-1999.0	129 min	Action, Drama	4.5	655	Kuku Kohli	Akshay Kumar	Twinkle Khanna	Aruna Irani
15506	Zulmi Raj	-2005.0	NaN	Action	NaN	NaN	Kiran Thej	Sangeeta Tiwari	NaN	NaN
15507	Zulmi Shikari	-1988.0	NaN	Action	NaN	NaN	NaN	NaN	NaN	NaN
15508	Zulm-O-Sitam	-1998.0	130 min	Action, Drama	6.2	20	K.C. Bokadia	Dharmendra	Jaya Prada	Arjun Sarja

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15509 entries, 0 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Name      15509 non-null  object 
 1   Year      14980 non-null  float64
 2   Duration  7241 non-null   object 
 3   Genre     13632 non-null  object 
 4   Rating    7919 non-null   float64
 5   Votes     7920 non-null   object 
 6   Director  14984 non-null  object 
 7   Actor 1   13892 non-null  object 
 8   Actor 2   13124 non-null  object 
 9   Actor 3   12356 non-null  object 
dtypes: float64(2), object(8)
memory usage: 1.2+ MB
```

In [6]:
```
df.describe()
```

Out[6]:

	Year	Rating
count	14980.000000	7919.000000
mean	-1987.012350	5.842758
std	25.417532	1.384783
min	-2022.000000	1.100000
25%	-2009.000000	4.900000
50%	-1991.000000	6.000000
75%	-1968.000000	6.800000
max	-1913.000000	14.000000

In [7]:
```
df.columns
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Name', 'Year', 'Duration', 'Genre', 'Rating', 'Votes', 'Director',
       'Actor 1', 'Actor 2', 'Actor 3'],
      dtype='object')
```

In [8]:
```
df.duplicated()
```

Out[8]:
```
0        False
1        False
2        False
3        False
4        False
         ...
15504    False
15505    False
15506    False
15507    False
15508    False
Length: 15509, dtype: bool
```

In [9]:
```
df.isnull().sum()/len(df)*100
```

Out[9]:
```
Name         0.000000
Year         3.410923
Duration    53.310981
Genre       12.102650
Rating      48.939326
Votes       48.932878
Director     3.385131
Actor 1     10.426204
Actor 2     15.378168
Actor 3     20.330131
dtype: float64
```

Pandas profiling report

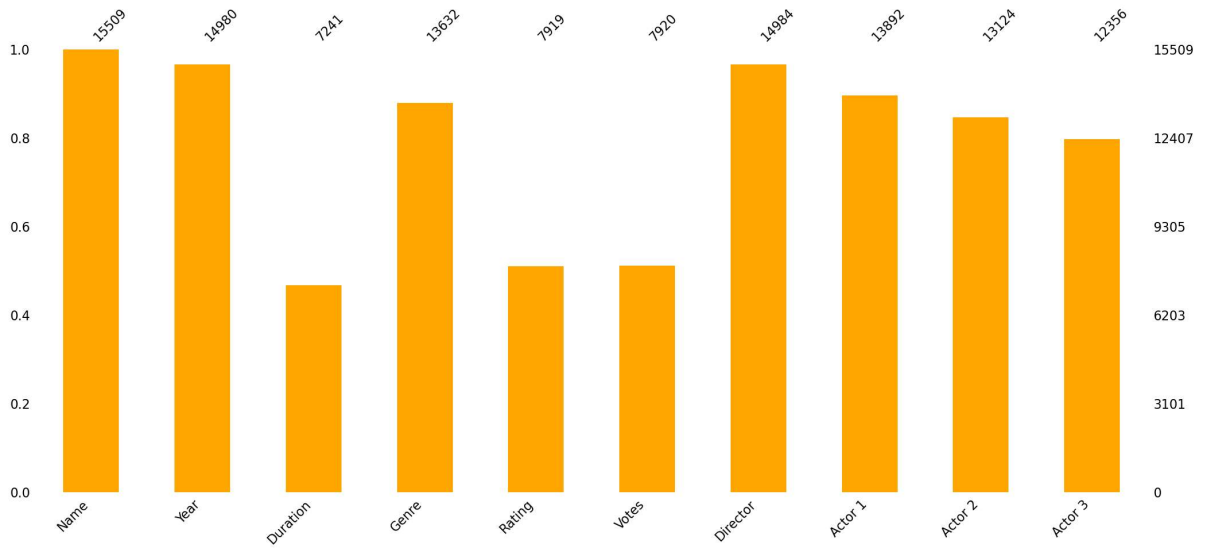
In [10]:
```
#import ydata_profiling as pandas_profiling  # Use ydata_profiling instead of par
#from IPython.display import display
#report = pandas_profiling.ProfileReport(df)
```

In [11]:
```
#sns.distplot(df['Director'],color='blue')
```

In [ ]:
```

```

In [12]:
```
import missingno as msn
```

In [12]:
```
import missingno as msn
msn.bar(df,color='orange')
```

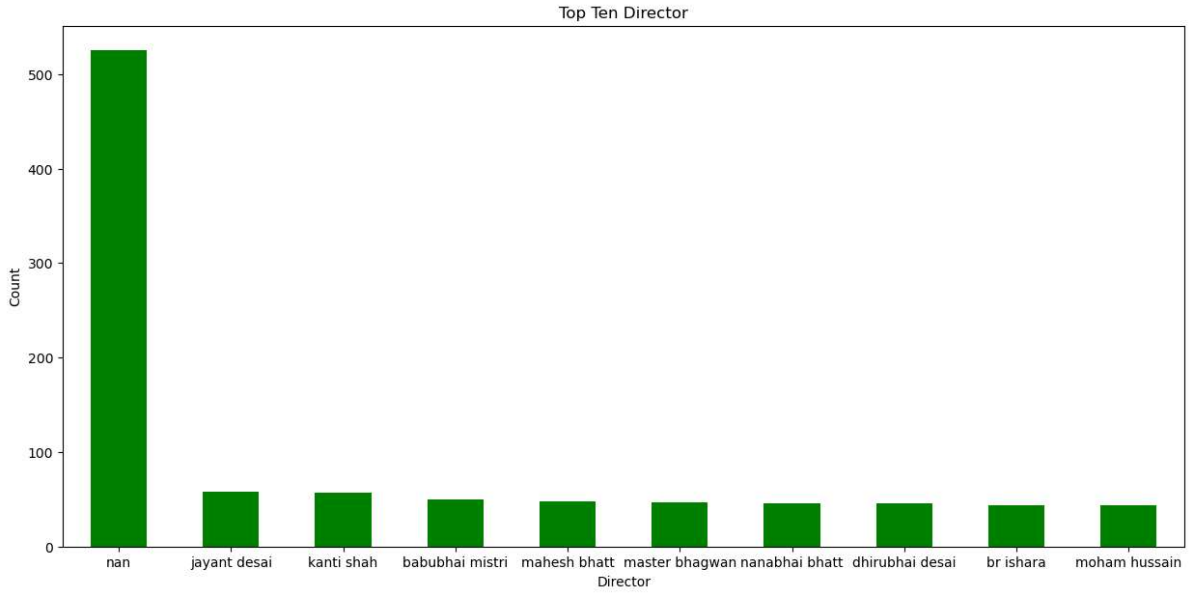


In [13]:
```
def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
df["Rating"] = df["Rating"].apply(clean)
df["Votes"] =  df["Votes"].apply(clean)
df["Director"] =  df["Director"].apply(clean)
df["Actor 1"] =  df["Actor 1"].apply(clean)
df["Actor 2"] =  df["Actor 2"].apply(clean)
df["Actor 3"] =  df["Actor 3"].apply(clean)
df["Genre"] =  df["Genre"].apply(clean)
df["Duration"] =  df["Duration"].apply(clean)
```

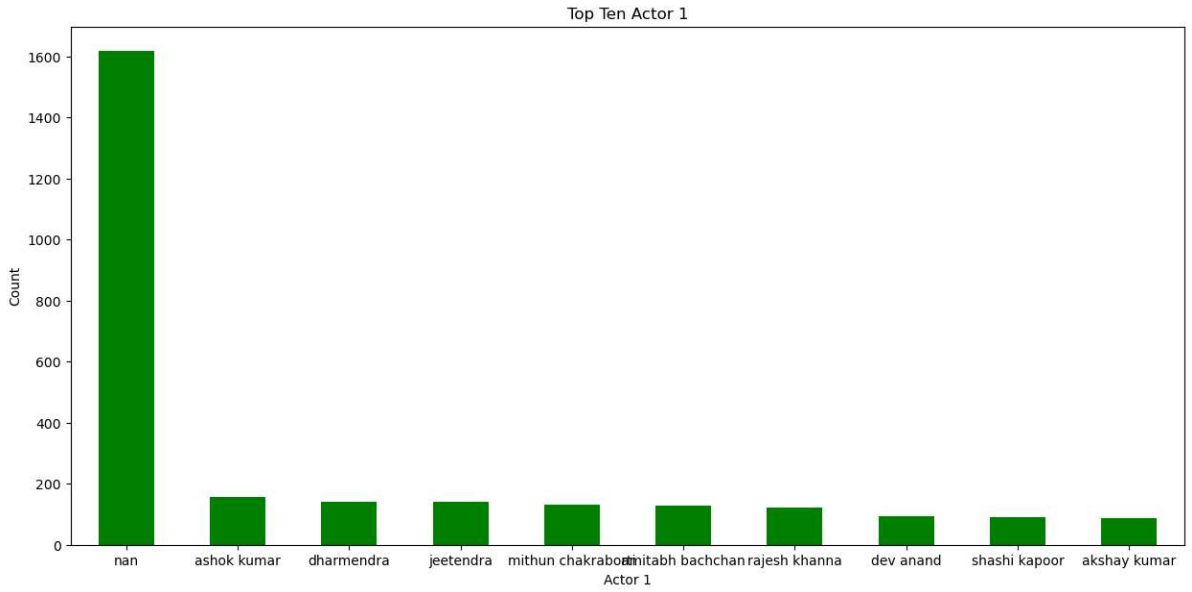
In [14]:
```
def TopTenPlot(column):
    global df
    df[column].value_counts().sort_values(ascending=False)[:10].plot(kind="bar",
    plt.xticks(rotation=0)
    plt.title("Top Ten {}".format(column))
    plt.xlabel(column)
    plt.ylabel("Count")
```

In [15]:
```
TopTenPlot("Director")
```

In [15]:
```
TopTenPlot("Director")
```

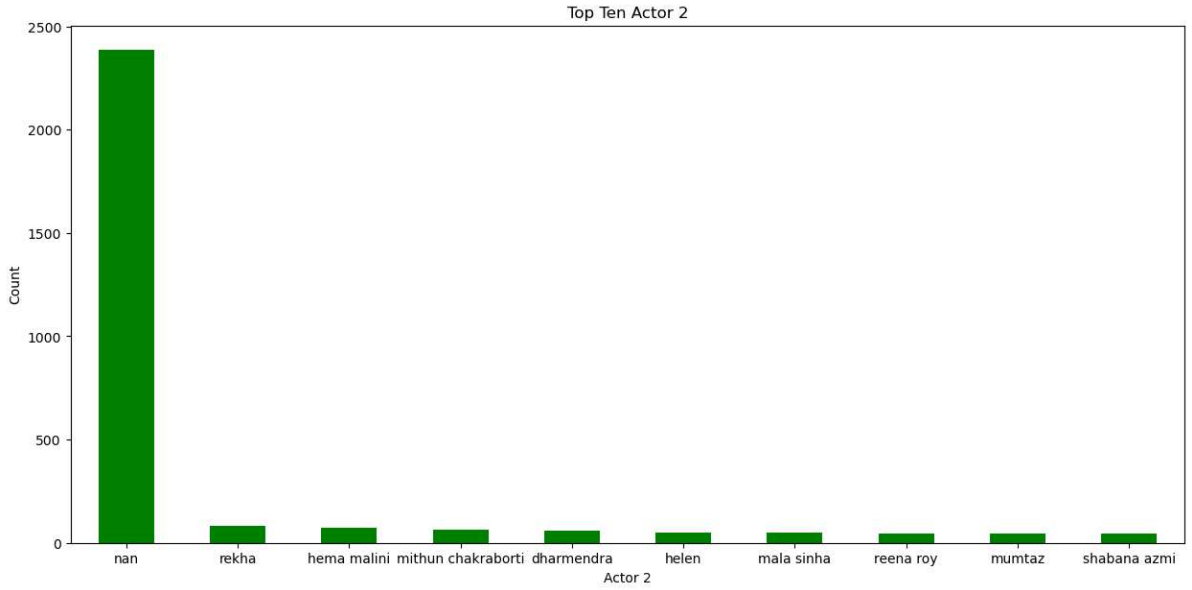


In [16]:
```
TopTenPlot("Actor 1")
```

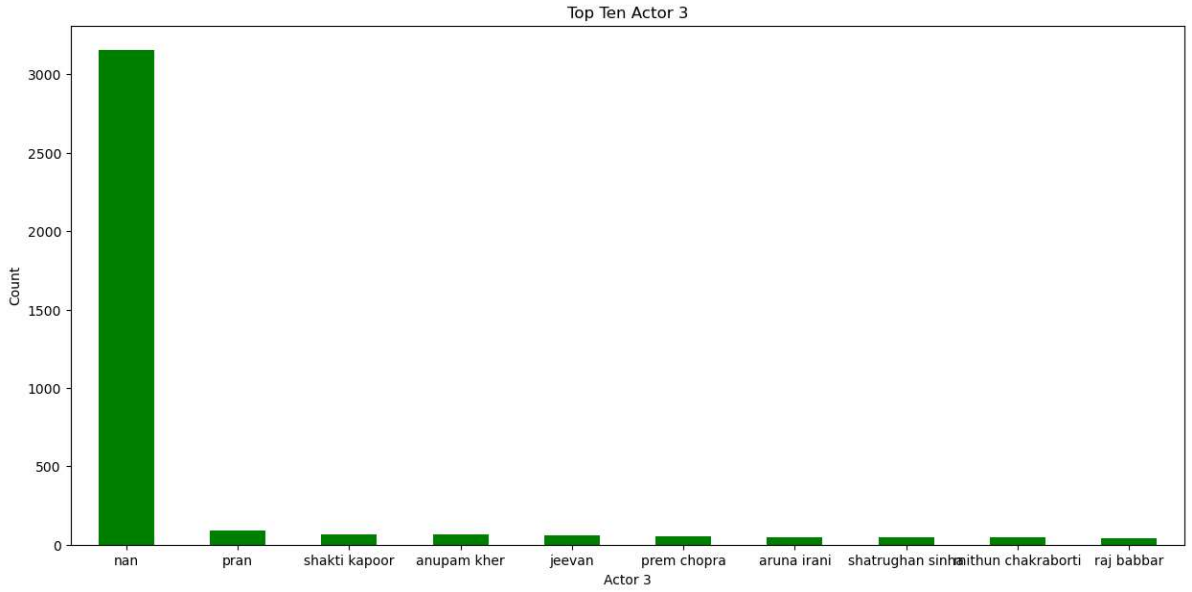


In [17]:
```
TopTenPlot("Actor 2")
```

In [17]:
```
TopTenPlot("Actor 2")
```



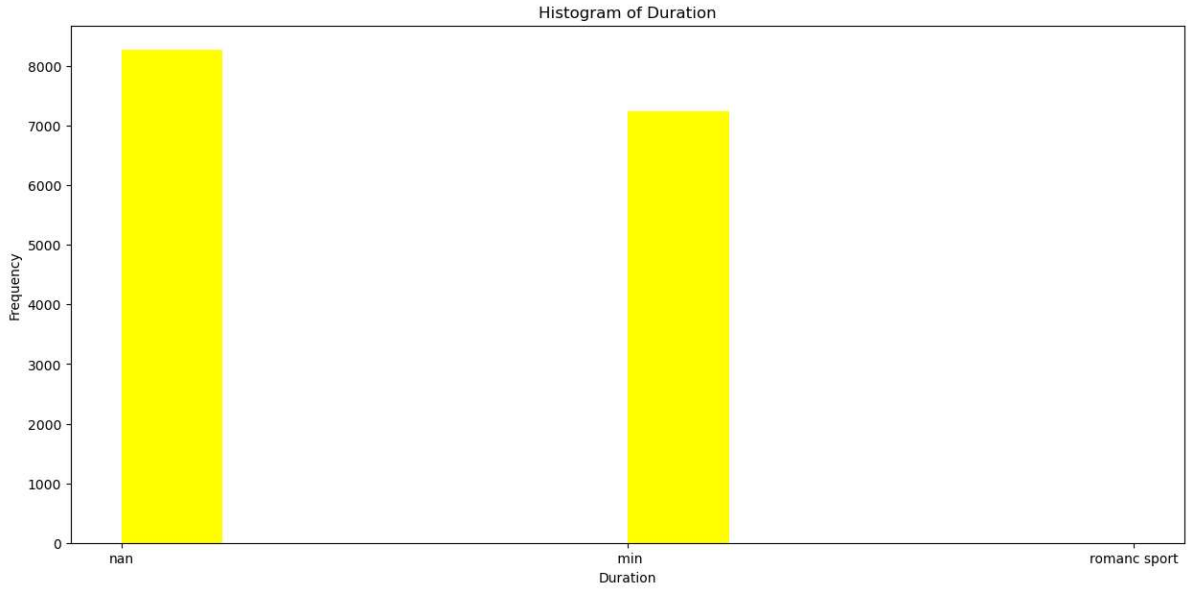
In [18]:
```
TopTenPlot("Actor 3")
```



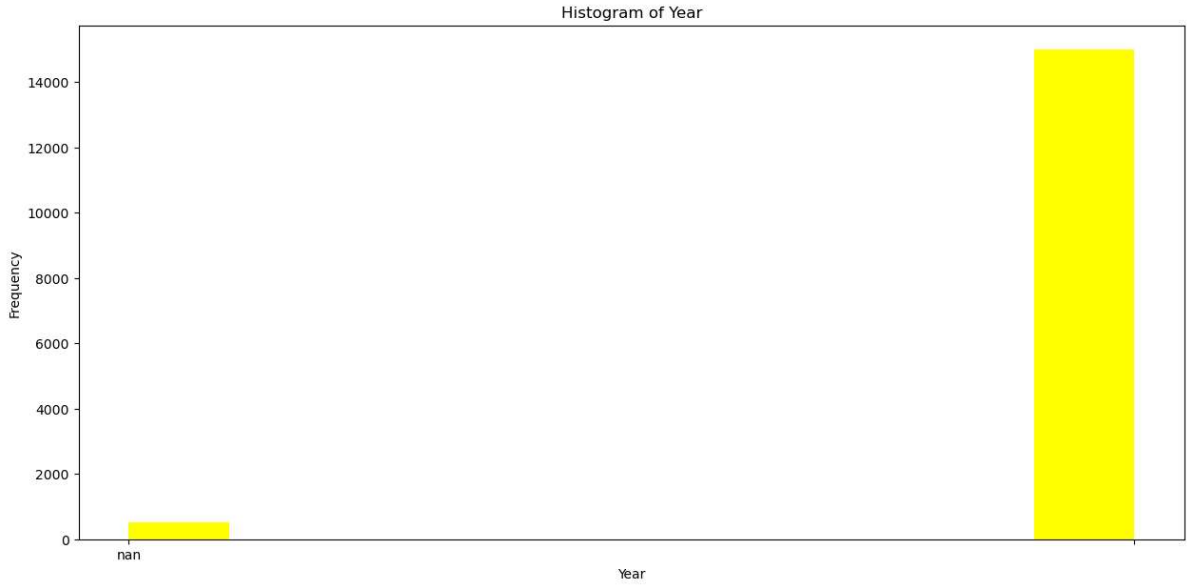
In [19]:
```
def Histogram(column):
    global df
    plt.figure(figsize=(15,7))
    plt.hist(df[column],color="yellow")
    plt.xticks(rotation=0)
    plt.title("Histogram of {}".format(column))
    plt.xlabel(column)
    plt.ylabel("Frequency")
```

In [20]:
```
Histogram("Duration")
```

In [20]:
```
Histogram("Duration")
```

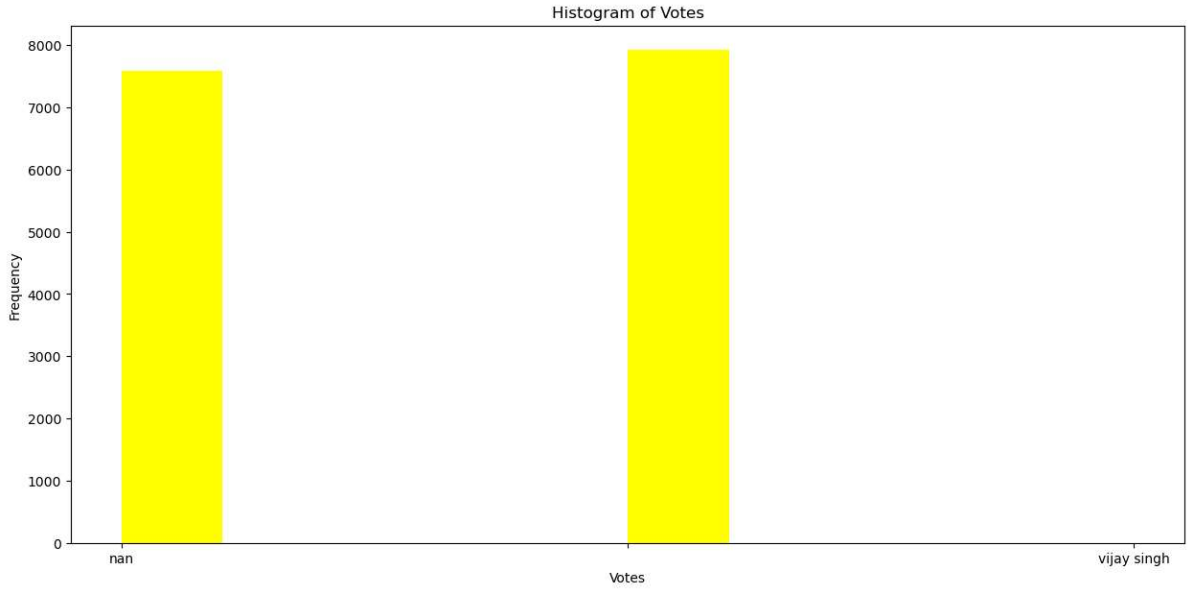


In [21]:
```
Histogram("Year")
```

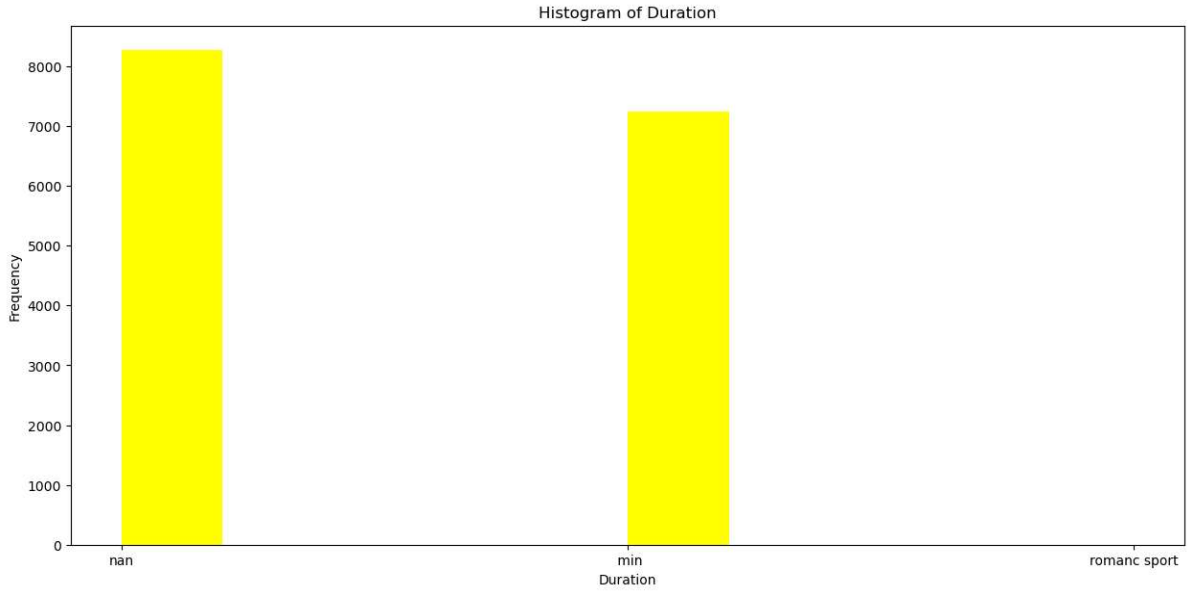


In [22]:
```
Histogram("Votes")
```

In [22]:
```
Histogram("Votes")
```

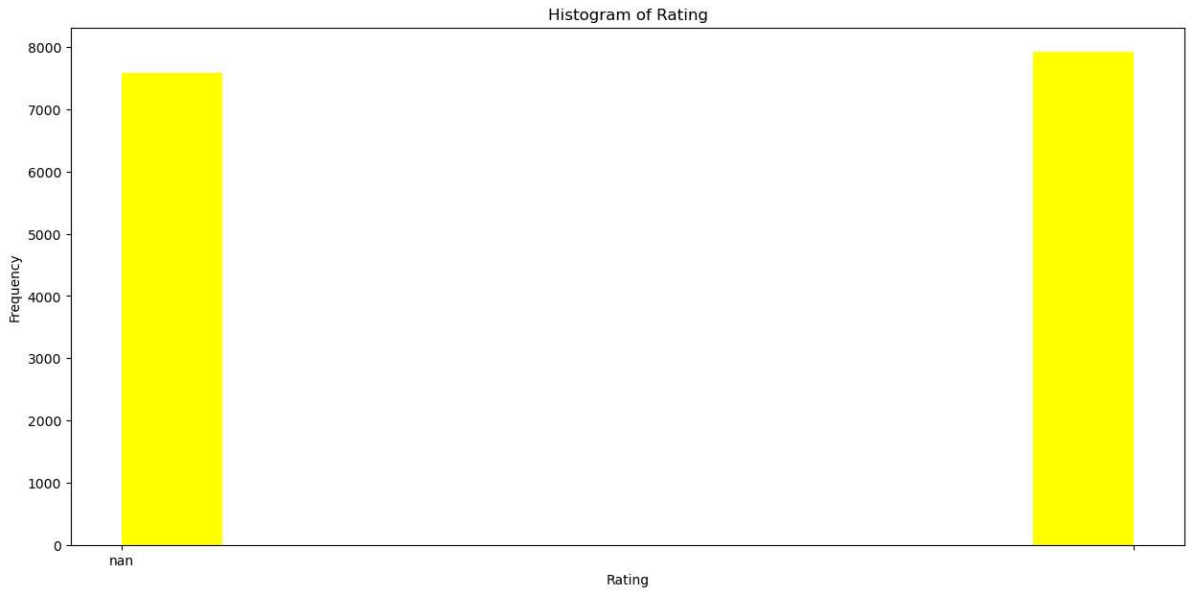


In [23]:
```
Histogram("Duration")
```



In [24]:
```
Histogram("Rating")
```

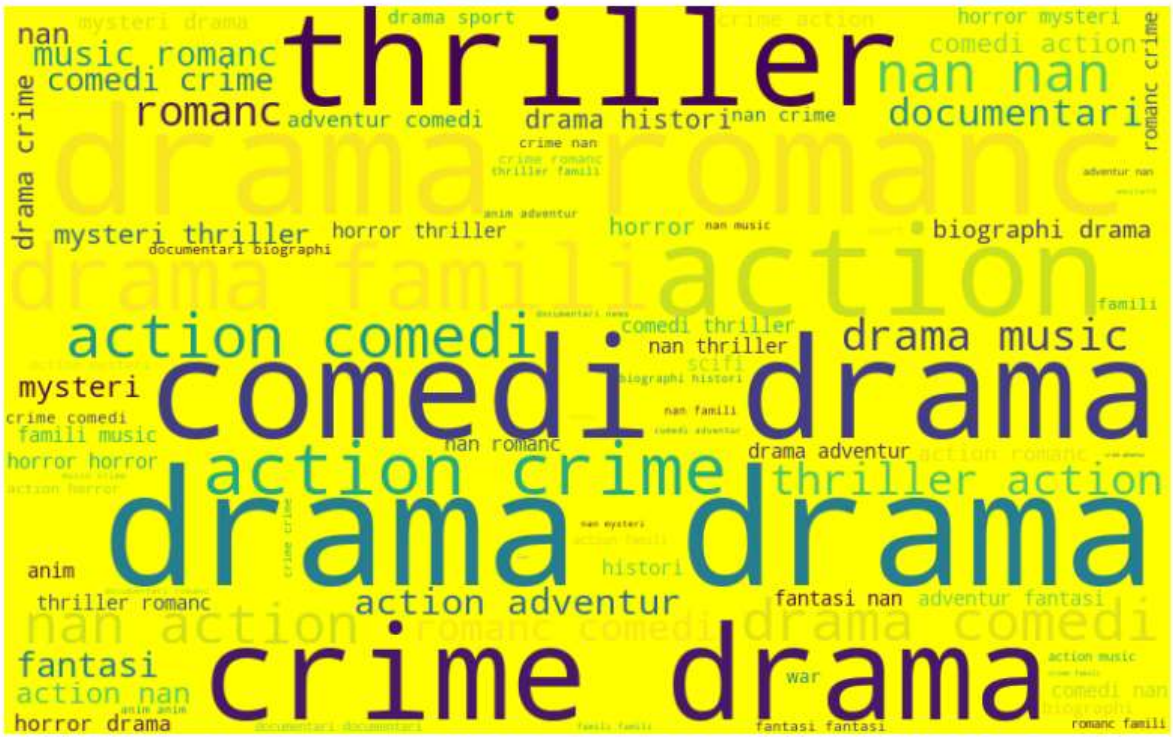
In [24]:
```
Histogram("Rating")
```



In [25]:
```
frequent_words = ' '.join([text for text in df['Name']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110,
 background_color='pink',colormap='viridis').generate(frequent_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
```

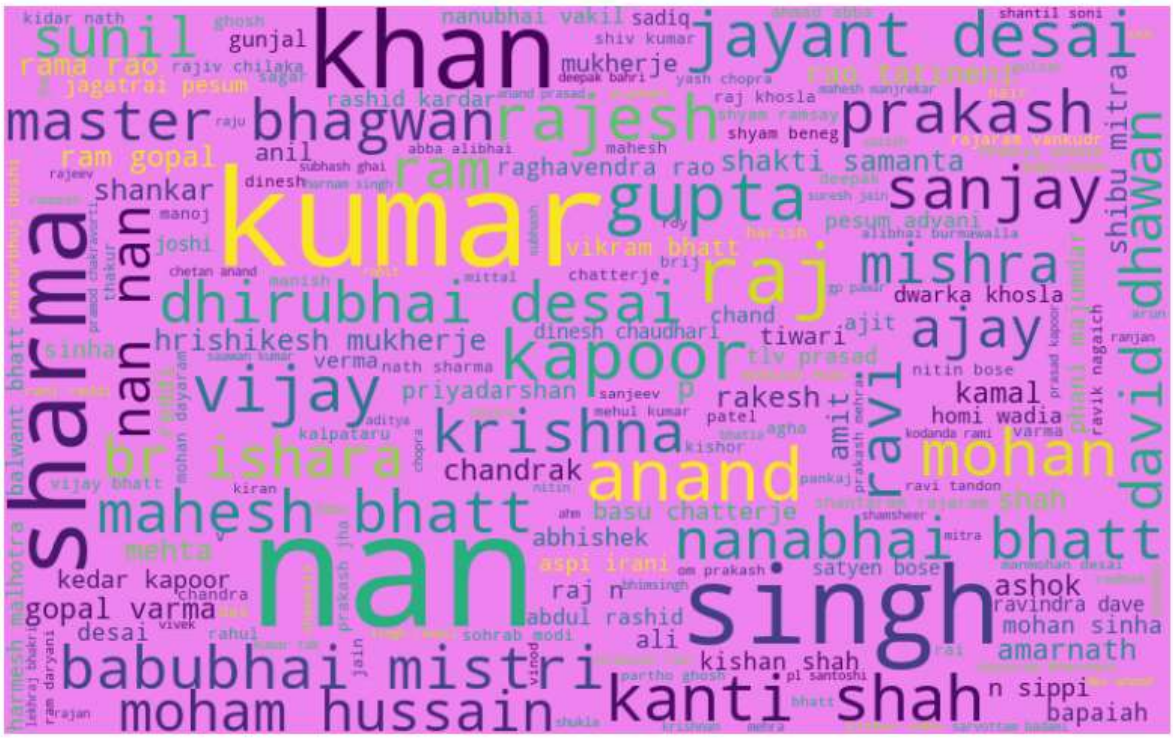
In [26]:
```
frequent_words = ' '.join([text for text in df['Genre']])
```

In [26]:
```
frequent_words = ' '.join([text for text in df['Genre']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110,
 background_color='yellow',colormap='viridis').generate(frequent_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
```



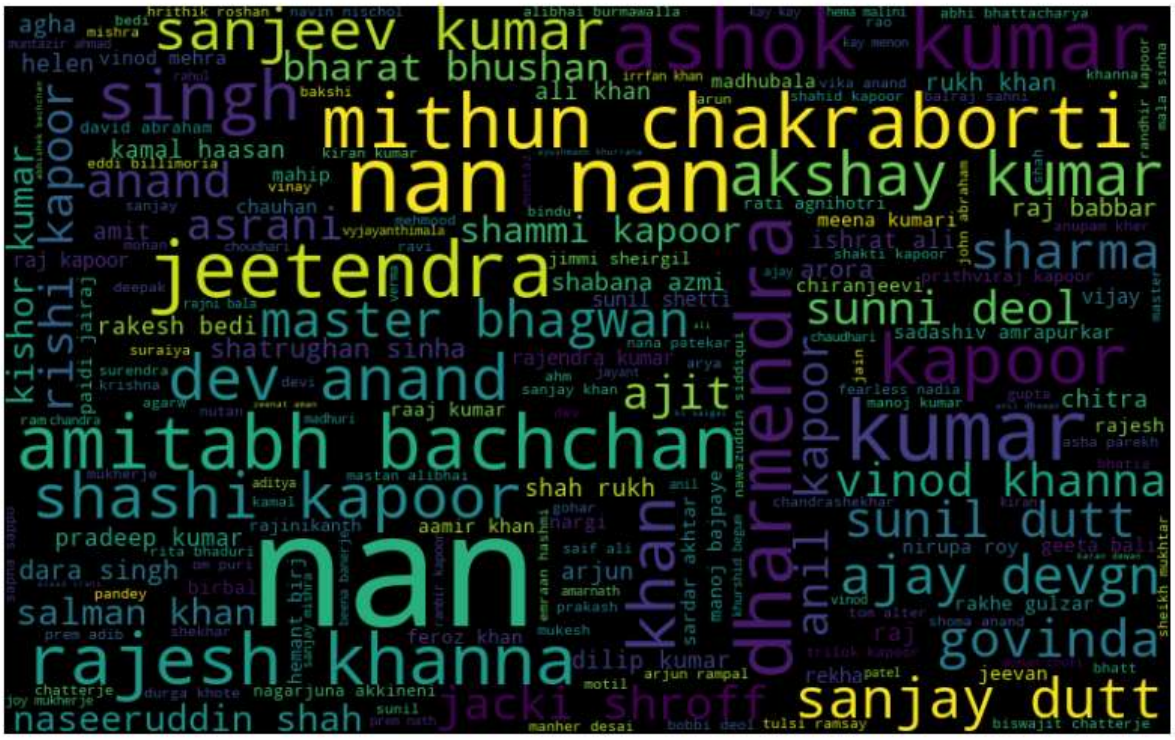
In [27]:
```
frequent_words = ' '.join([text for text in df['Director']])
```

In [27]:
```
frequent_words = ' '.join([text for text in df['Director']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110,
 background_color='violet',colormap='viridis').generate(frequent_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
```



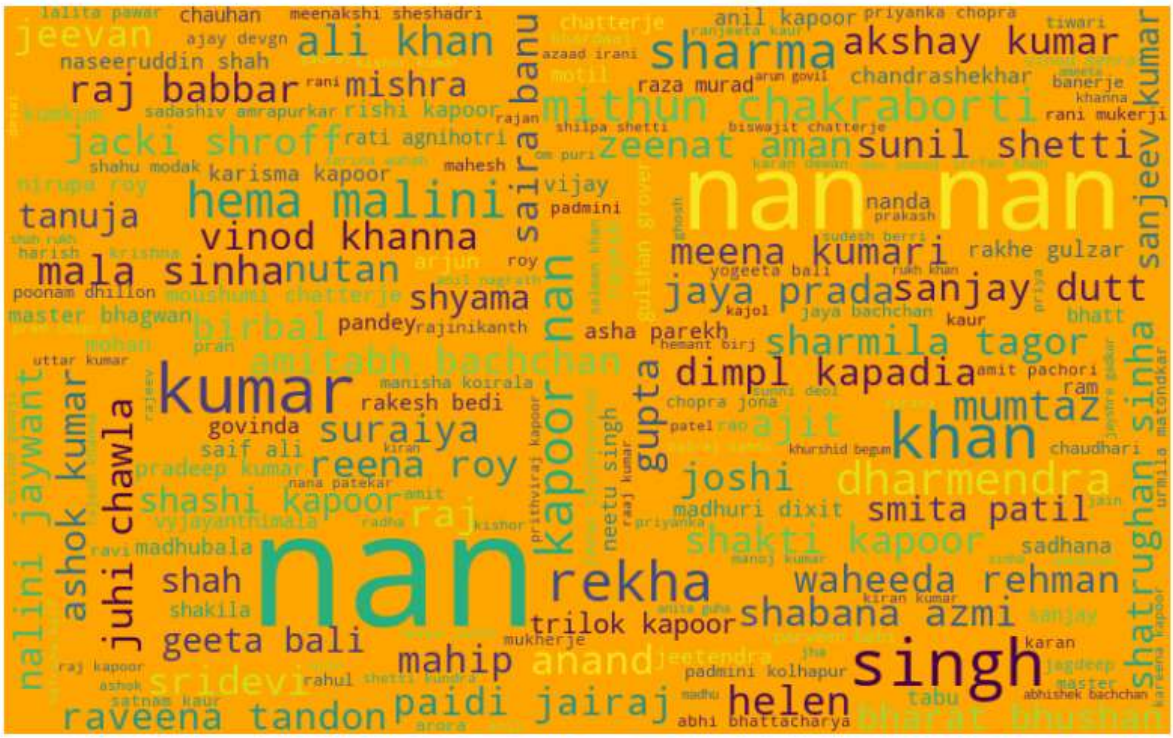
In [28]:
```
frequent_words = ' '.join([text for text in df['Actor 1']])
```

In [28]:
```
frequent_words = ' '.join([text for text in df['Actor 1']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110,
 background_color='black',colormap='viridis').generate(frequent_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
```



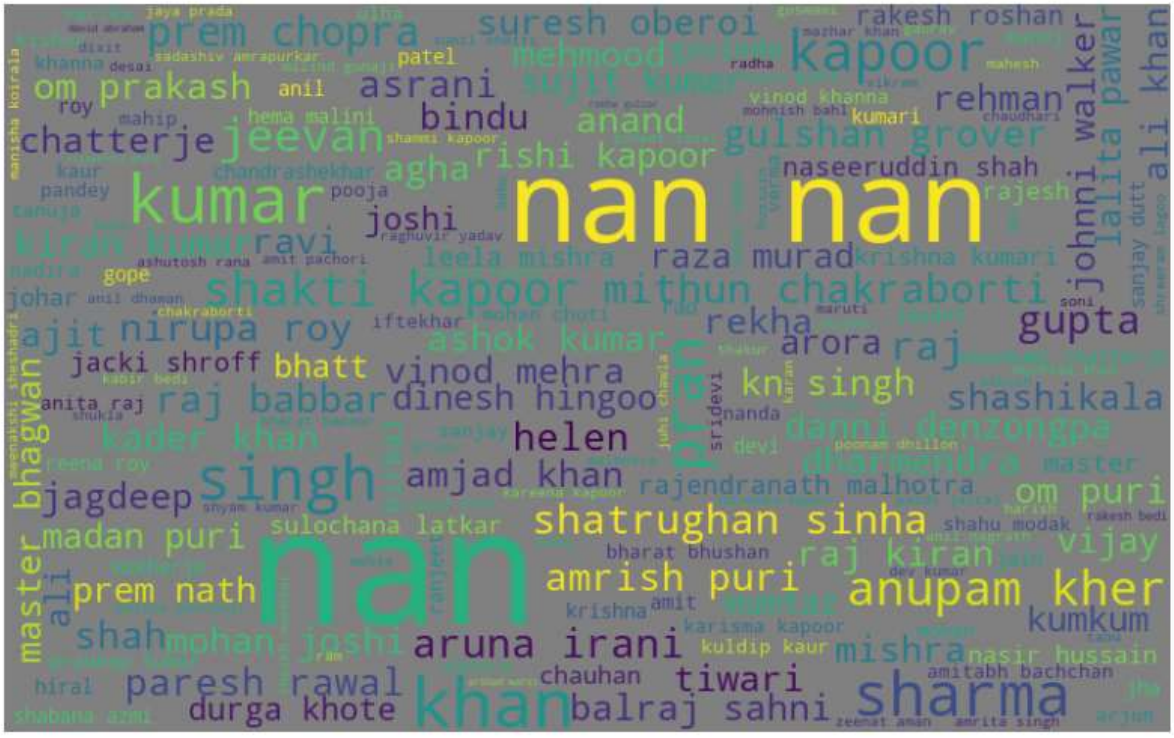
In [29]:
```
frequent_words = ' '.join([text for text in df['Actor 2']])
```

In [29]:
```
frequent_words = ' '.join([text for text in df['Actor 2']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110,
 background_color='orange',colormap='viridis').generate(frequent_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
```



In [30]:
```
frequent_words = ' '.join([text for text in df['Actor 3']])
```

In [30]:
```
frequent_words = ' '.join([text for text in df['Actor 3']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110,
 background_color='grey',colormap='viridis').generate(frequent_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
```



In [31]:
```
df.columns
```

Out[31]:
```
Index(['Name', 'Year', 'Duration', 'Genre', 'Rating', 'Votes', 'Director',
       'Actor 1', 'Actor 2', 'Actor 3'],
      dtype='object')
```

sentiment score

In [32]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

In [33]:
```
df['Rating'].value_counts()
```

Out[33]:
```
       7919
nan    7590
Name: Rating, dtype: int64
```

In [34]:
```
sia.polarity_scores(df.loc[0]['Rating'])
```

Out[34]:
```
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
```

In [35]:
```
df.loc[0]['Rating']
```

In [35]:
```
df.loc[0]['Rating']
```

Out[35]: 'nan'

In [36]:
```
df.shape
```

Out[36]: (15509, 10)

In [37]:
```
df['score']=df['Rating'].apply(lambda rating : sia.polarity_scores(rating) )
```

Out[37]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
0		nan	nan	drama	nan	nan	js randhawa	manmauji	birbal	rajendra bhatia
1	#Gadhvi (He thought he was Gandhi)		min	drama			gaurav bakshi	rasika dugal	vivek ghamand	arvind jangid
2	#Homecoming		min	drama music	nan	nan	soumyajit majumdar	sayani gupta	plabita borthakur	roy angana
3	#Yaaram		min	comedi romanc			ovai khan	prateik	ishita raj	siddhan kapoo
4	...And Once Again		min	drama	nan	nan	amol palekar	rajat kapoor	rituparna sengupta	antara ma
5	...Aur Pyaar Ho Gaya		min	comedi drama music			rahul rawail	bobbi deol	aishwarya rai bachchan	shamm kapoo
6	...Yahaan		min	drama romanc war			shoojit sircar	jimmi sheirgil	minissha lamba	yashpa sharma
7	.in for Motion		min	documentari	nan	nan	anirban datta	nan	nan	nan
8	?: A Question Mark		min	horror mysteri thriller			allyson patel	yash dave	muntazir ahmad	kiran bhatia
9	@Andheri		min	action crime thriller			biju bhaskar nair	augustin	fathima babu	byon

In [38]:
```
len(df['score'])
```

Out[38]: 15509

In [39]:
```
df['compound'] = df['score'].apply(lambda score_dict : score_dict['compound'])
```

Out[38]: 15509

In [39]:
```
df['compound'] = df['score'].apply(lambda score_dict : score_dict['compound'])
```

Out[39]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
0		nan	nan	drama	nan	nan	js randhawa	manmauji	birbal	rajendra bhatia
1	#Gadhvi (He thought he was Gandhi)		min	drama			gaurav bakshi	rasika dugal	vivek ghamand	arvind jangid
2	#Homecoming		min	drama music	nan	nan	soumyajit majumdar	sayani gupta	plabita borthakur	roy angana
3	#Yaaram		min	comedi romanc			ovai khan	prateik	ishita raj	siddhan kapoo
4	...And Once Again		min	drama	nan	nan	amol palekar	rajat kapoor	rituparna sengupta	antara ma
5	...Aur Pyaar Ho Gaya		min	comedi drama music			rahul rawail	bobbi deol	aishwarya rai bachchan	shamm kapoo
6	...Yahaan		min	drama romanc war			shoojit sircar	jimmi sheirgil	minissha lamba	yashpa sharma
7	.in for Motion		min	documentari	nan	nan	anirban datta	nan	nan	nan
8	?: A Question Mark		min	horror mysteri thriller			allyson patel	yash dave	muntazir ahmad	kiran bhatia
9	@Andheri		min	action crime thriller			biju bhaskar nair	augustin	fathima babu	byon

In [40]:
```
df['comp_score'] = df['compound'].apply(lambda c: 'Actor1' if c >=0 else 'Actor2'
```

In [40]:
```
df['comp_score'] = df['compound'].apply(lambda c: 'Actor1' if c >=0 else 'Actor2'
```

2	#Homecoming	min	drama music	nan	nan	soumyajit majumdar	sayani gupta	plabita borthakur	anç
3	#Yaaram	min	comedi romanc			ovai khan	prateik	ishita raj	siddl kaj
4	...And Once Again	min	drama	nan	nan	amol palekar	rajat kapoor	rituparna sengupta	an
5	...Aur Pyaar Ho Gaya	min	comedi drama music			rahul rawail	bobbi deol	aishwarya rai bachchan	sha kaj
6	...Yahaan	min	drama romanc war			shoojit sircar	jimmi sheirgil	minissha lamba	yas sha

Evaluation matrix

In [41]:
```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_sc
```

In [42]:
```
print(confusion_matrix(df['Rating'], df['comp_score']))
```

In [42]:
```
print(confusion_matrix(df['Rating'], df['comp_score']))
print("************"*10)
print(classification_report(df['Rating'], df['comp_score']))
print("************"*10)
```

```
[[   0 7919    0]
 [   0    0    0]
 [   0 7590    0]]
**********************************************************************************
**************************************

C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set t
o 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set t
o 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

              precision    recall  f1-score   support

                   0.00      0.00      0.00    7919.0
      Actor1       0.00      0.00      0.00       0.0
         nan       0.00      0.00      0.00    7590.0

    accuracy                           0.00   15509.0
   macro avg       0.00      0.00      0.00   15509.0
weighted avg       0.00      0.00      0.00   15509.0

**********************************************************************************
**************************************
0.0

C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set t
o 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [43]:
```
print(confusion_matrix(df['Votes'], df['comp_score']))
```

In [43]:
```
print(confusion_matrix(df['Votes'], df['comp_score']))
print("************"*10)
print(classification_report(df['Votes'], df['comp_score']))
print("************"*10)
```

```
[[   0 7919    0    0]
 [   0    0    0    0]
 [   0 7589    0    0]
 [   0    1    0    0]]
********************************************************************************
****************************************

C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set t
o 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set t
o 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set t
o 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

              precision    recall  f1-score   support

                   0.00      0.00      0.00    7919.0
      Actor1       0.00      0.00      0.00       0.0
         nan       0.00      0.00      0.00    7589.0
 vijay singh       0.00      0.00      0.00       1.0

    accuracy                           0.00   15509.0
   macro avg       0.00      0.00      0.00   15509.0
weighted avg       0.00      0.00      0.00   15509.0

********************************************************************************
****************************************
0.0
```

In [44]:
```
print(confusion_matrix(df['Votes'], df['comp_score']))
```

In [44]:
```
print(confusion_matrix(df['Votes'], df['comp_score']))
print("************"*10)
print(classification_report(df['Votes'], df['comp_score']))
print("************"*10)
```

```
[[   0 7919    0    0]
 [   0    0    0    0]
 [   0 7589    0    0]
 [   0    1    0    0]]
*******************************************************************************
*****************************************

C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set t
o 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set t
o 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set t
o 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

              precision    recall  f1-score   support

                   0.00      0.00      0.00    7919.0
      Actor1       0.00      0.00      0.00       0.0
         nan       0.00      0.00      0.00    7589.0
 vijay singh       0.00      0.00      0.00       1.0

    accuracy                           0.00   15509.0
   macro avg       0.00      0.00      0.00   15509.0
weighted avg       0.00      0.00      0.00   15509.0

*******************************************************************************
*****************************************
0.0
```

In [45]:
```
from tqdm import tqdm
```

In [45]:
```
from tqdm import tqdm

preprocessed_reviews = []

for sentence in tqdm(df['Rating'].values):
    sentence = re.sub('[^a-zA-Z]',' ',sentence)
    sentence = ' '.join(low.lower() for low in sentence.split() if low.lower() no
```

```
100%|██████████████████████████████████████████████████████████████████████████| 15509/15509 [00:03<00:00, 4083.65it/s]
```

In [46]:
```
df['Rating'].values
```

Out[46]:
```
array(['nan', '', 'nan', ..., 'nan', 'nan', ''], dtype=object)
```

In [47]:
```
preprocessed_reviews
```

```
 '',
 '',
 '',
 '',
 '',
 '',
 'nan',
 'nan',
 '',
 '',
 '',
 '',
 'nan',
 '',
 '',
 '',
 '',
 'nan',
 'nan',
 '',
 'nan',
 '',
 '',
```

In [48]:
```
nltk.download('vader_lexicon')
sentiments = SentimentIntensityAnalyzer()
df["Positive"] = [sentiments.polarity_scores(i)["pos"] for i in df["Rating"]]
df["Negative"] = [sentiments.polarity_scores(i)["neg"] for i in df["Rating"]]
df["Neutral"] = [sentiments.polarity_scores(i)["neu"] for i in df["Rating"]]
df = df[["Rating", "Positive", "Negative", "Neutral"]]
```

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

	Rating	Positive	Negative	Neutral
0	nan	0.0	0.0	1.0
1		0.0	0.0	0.0
2	nan	0.0	0.0	1.0
3		0.0	0.0	0.0
4	nan	0.0	0.0	1.0

In [49]:
```
A = sum(df["Positive"])
```

In [49]:
```
A = sum(df["Positive"])
B = sum(df["Negative"])
C = sum(df["Neutral"])

def sentiment_score(a, b, c):
    if (a>b):
        print("Positive 😊 ")
    elif (b>c):
        print("Negative 😠 ")
    else:
        print("Neutral 🙂 ")
```

```
Neutral 🙂
```

Feature Extraction

In [50]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
x = tfidf.fit_transform(preprocessed_reviews).toarray()
```

Out[50]:
```
(15509, 1)
```

In [51]:
```
pd.DataFrame(x).head()
```

Out[51]:

	0
0	1.0
1	0.0
2	1.0
3	0.0
4	1.0

In [52]:
```
df['Rating'].value_counts()
```

Out[52]:
```
       7919
nan    7590
Name: Rating, dtype: int64
```

In [53]:
```
# Split the data into training and test
from sklearn.model_selection import train_test_split
```

Applying Models

In [54]:
```
from sklearn.linear_model import LogisticRegression
```

In [54]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.ensemble import VotingClassifier
```

In [55]:
```
# LogisticRegression
```

In [55]:
```
# LogisticRegression
logistic = LogisticRegression()
lr = logistic.fit(x_train, y_train)
y_pred_lr = logistic.predict(x_test)
accuracy_lr = accuracy_score(y_test, y_pred_lr)

# DecisionTree
dtree = DecisionTreeClassifier()
dt = dtree.fit(x_train, y_train)
y_pred_dt = dtree.predict(x_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)

# RandomForest
rfmodel = RandomForestClassifier()
rf = rfmodel.fit(x_train, y_train)
y_pred_rf = rfmodel.predict(x_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)

# BaggingClassifier
bagg = BaggingClassifier()
bg = bagg.fit(x_train, y_train)
y_pred_bg = bagg.predict(x_test)
accuracy_bg = accuracy_score(y_test, y_pred_bg)

# AdaBoostClassifier
ada = AdaBoostClassifier()
ad = ada.fit(x_train, y_train)
y_pred_ad = ada.predict(x_test)
accuracy_ad = accuracy_score(y_test, y_pred_ad)

# GradientBoostingClassifier
gdb = GradientBoostingClassifier()
gd = gdb.fit(x_train, y_train)
y_pred_gd = gdb.predict(x_test)
accuracy_gd = accuracy_score(y_test, y_pred_gd)

# XGBClassifier = RF + GDBoosting - lambda - regularisation, gamma - autoprunning
#xgb = XGBClassifier()
#xg = xgb.fit(x_train, y_train)
#y_pred_xg = xgb.predict(x_test)
#accuracy_xg = accuracy_score(y_test, y_pred_xg)


# SVM
svc = SVC()
sv = svc.fit(x_train, y_train)
y_pred_sv = svc.predict(x_test)
accuracy_sv = accuracy_score(y_test, y_pred_sv)

# KNN
knn = KNeighborsClassifier()
kn = knn.fit(x_train, y_train)
y_pred_knn = knn.predict(x_test)
accuracy_knn = accuracy_score(y_test, y_pred_knn)

# GaussianNB
naive_gb = GaussianNB()
ngb = naive_gb.fit(x_train, y_train)
y_pred_ngb = naive_gb.predict(x_test)
accuracy_ngb = accuracy_score(y_test, y_pred_ngb)
```

```
ngb = naive_gb.fit(x_train, y_train)
y_pred_ngb = naive_gb.predict(x_test)
accuracy_ngb = accuracy_score(y_test, y_pred_ngb)

# BernoulliNB
naive_bn = BernoulliNB()
nbr = naive_bn.fit(x_train, y_train)
y_pred_nbr = naive_bn.predict(x_test)
accuracy_nbr = accuracy_score(y_test, y_pred_nbr)
```

In [56]:
```
from sklearn.ensemble import VotingClassifier
```

In [ ]:
```
```

In [57]:
```
evc = VotingClassifier(estimators=[('lr',lr),('dt',dt),('rf', rf),('bg', bg),('ad
                                   ('gd', gd),('sv', sv),('kn', kn),
                                   ('ngb', ngb),('nbr', nbr)], voting='hard')


model_evc = evc.fit(x_train, y_train)
pred_evc = evc.predict(x_test)
```

In [58]:
```
list1 = ['LogisticRegression','DecisionTree','RandomForest','Bagging','Adaboost',
         'GradientBoosting','SupportVector','KNearestNeighbors',
```

In [59]:
```
list2 = [accuracy_lr, accuracy_dt, accuracy_rf, accuracy_bg,accuracy_ad, accuracy
```

In [60]:
```
list3 = [logistic, dtree, rfmodel, bagg, ada, gdb, svc, knn, naive_gb,naive_bn, e
```

Classification Report

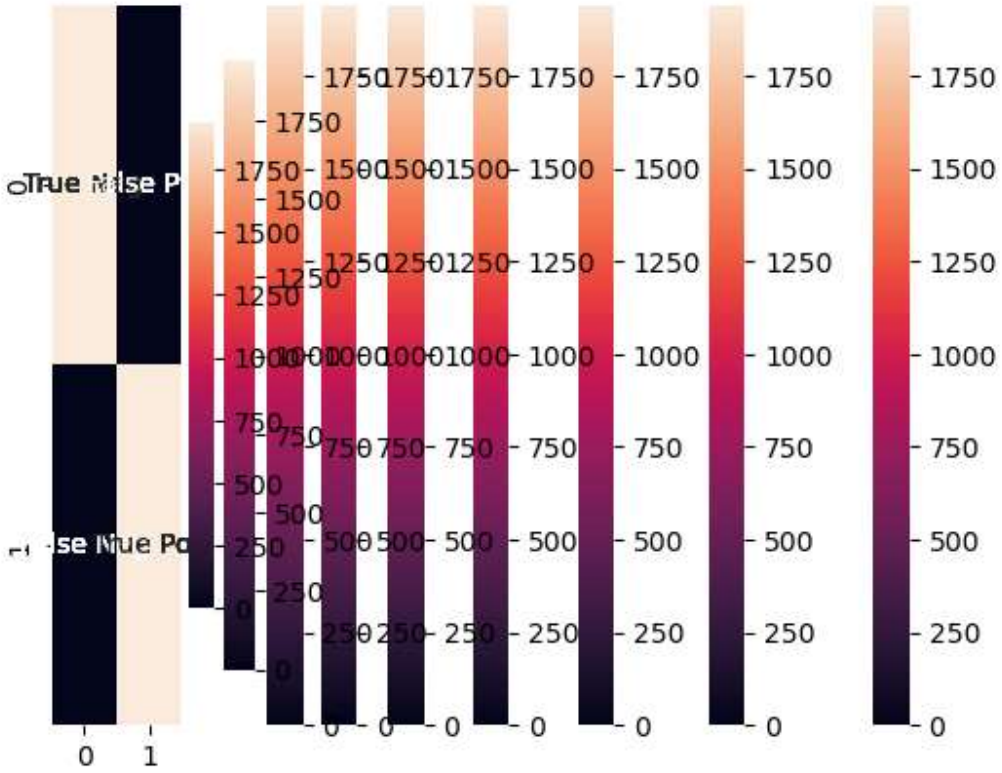
In [61]:
```
print('LogisticRegression: \n',classification_report(y_test,y_pred_lr))
```

In [61]:
```
print('LogisticRegression: \n',classification_report(y_test,y_pred_lr))
print('DecisionTreeClassifier: \n',classification_report(y_test,y_pred_dt))
print('RandomForestClassifier: \n',classification_report(y_test,y_pred_rf))
print('BaggingClassifier: \n',classification_report(y_test,y_pred_bg))
print('AdaboostClassifier: \n',classification_report(y_test,y_pred_ad))
print('SupportVectorClassifier: \n',classification_report(y_test,y_pred_sv))
print('KNearestNeighborsClassifier: \n',classification_report(y_test,y_pred_knn))
print('NaiveBayesGaussianClassifier: \n',classification_report(y_test,y_pred_ngb))
print('NaiveBayesBernoulliesClassifier: \n',classification_report(y_test,y_pred_r
```

```
DecisionTreeClassifier:
               precision    recall  f1-score   support

                   1.00      1.00      1.00      1938
         nan       1.00      1.00      1.00      1940

    accuracy                           1.00      3878
   macro avg       1.00      1.00      1.00      3878
weighted avg       1.00      1.00      1.00      3878

RandomForestClassifier:
               precision    recall  f1-score   support

                   1.00      1.00      1.00      1938
         nan       1.00      1.00      1.00      1940

    accuracy                           1.00      3878
   macro avg       1.00      1.00      1.00      3878
weighted avg       1.00      1.00      1.00      3878
```

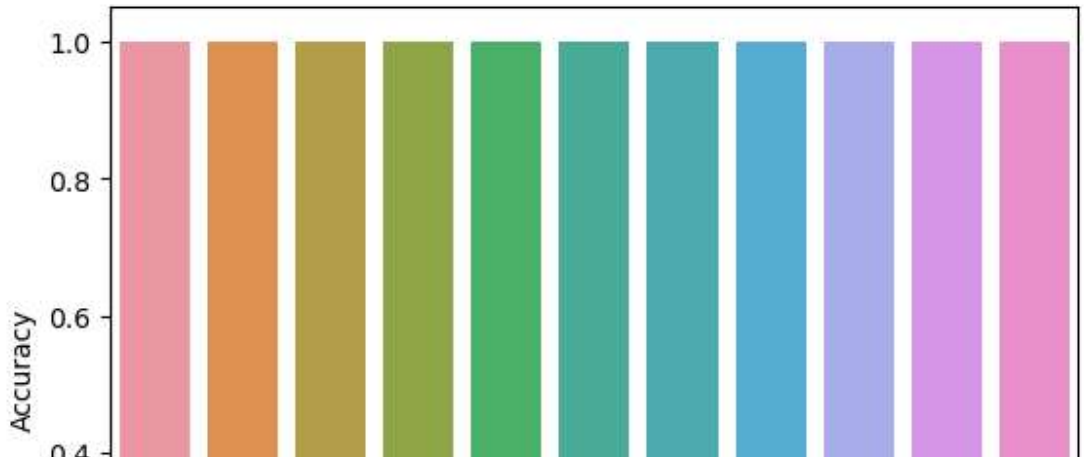
In [62]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [62]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)

sns.heatmap(confusion_matrix(y_test,y_pred_lr), annot=labels, fmt='')
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)

sns.heatmap(confusion_matrix(y_test,y_pred_dt), annot=labels, fmt='')
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)

sns.heatmap(confusion_matrix(y_test,y_pred_rf), annot=labels, fmt='')
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)

sns.heatmap(confusion_matrix(y_test,y_pred_bg), annot=labels, fmt='')
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)

sns.heatmap(confusion_matrix(y_test,y_pred_ad), annot=labels, fmt='')
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)

sns.heatmap(confusion_matrix(y_test,y_pred_sv), annot=labels, fmt='')
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)

sns.heatmap(confusion_matrix(y_test,y_pred_knn), annot=labels, fmt='')
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)

sns.heatmap(confusion_matrix(y_test,y_pred_ngb), annot=labels, fmt='')
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)

sns.heatmap(confusion_matrix(y_test,y_pred_nbr), annot=labels, fmt='')
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)
```



In [63]:
```
final_accuracy = pd.DataFrame({'Method Used': list1, "Accuracy": list2})
print(final_accuracy)
charts = sns.barplot(x="Method Used", y = 'Accuracy', data=final_accuracy)
charts.set_xticklabels(charts.get_xticklabels(), rotation=90)
```

```
7       KNearestNeighbors       1.0
8      NaiveBayesGaussian       1.0
9   NaiveBayesBernoullies       1.0
10       VotingClassifier       1.0
Axes(0.125,0.11;0.775x0.77)
```



In [64]:
```
nltk.download('vader_lexicon')
```

In [64]:
```
nltk.download('vader_lexicon')
sentiments = SentimentIntensityAnalyzer()
df["Positive"] = [sentiments.polarity_scores(i)["pos"] for i in df["Rating"]]
df["Negative"] = [sentiments.polarity_scores(i)["neg"] for i in df["Rating"]]
df["Neutral"] = [sentiments.polarity_scores(i)["neu"] for i in df["Rating"]]
df = df[["Rating", "Positive", "Negative", "Neutral"]]
```

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

	Rating	Positive	Negative	Neutral
0	nan	0.0	0.0	1.0
1		0.0	0.0	0.0
2	nan	0.0	0.0	1.0
3		0.0	0.0	0.0
4	nan	0.0	0.0	1.0

In [65]:
```
A = sum(df["Positive"])
B = sum(df["Negative"])
C = sum(df["Neutral"])

def sentiment_score(a, b, c):
    if (a>b):
        print("Positive 😊 ")
    elif (b>c):
        print("Negative 😠 ")
    else:
        print("Neutral 🙂 ")
```

```
Neutral 🙂
```

conclusion

```
  There are 7590 nan Rating.
  most of the review are in favor of neutarl Rating.
  with the help of sentiment analysis we conclude that most of te review are
neutarl.
```