Problem statement:

We were given the following data sets each with a purpose, this datasets contains data of youtube, each data base has different titles and info, our task is to Extract this data, clean it analysis it and answer the following.

1. Perform Sentiment Analysis, and checkwhat are the most used positive words and negitive words
2. Check top 10 most usedemoji's in the comments.
3. Ceking for the most liked category
4. Find out whether audience is engaged or not

. Use UScomments dataset for analysing the sentiment analysis and emojy tasks

. Use the datasets in the additional data file for remaining task

1. Perform Sentiment Analysis, and check the most used positive words and negitive words in comments

Importing data for analysis

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

In [2]:
```
df = pd.read_csv("UScomments.csv", error_bad_lines=False)  # Keeping error_bad_lines as false to skip the
                                                           # lines with improper data
```

```
C:\Users\PC\AppData\Local\Temp\ipykernel_10728\1061535121.py:1: FutureWarning: The error_bad_lines argument has been
deprecated and will be removed in a future version. Use on_bad_lines in the future.


  df = pd.read_csv("UScomments.csv", error_bad_lines=False)  # Keeping error_bad_lines as false to skip the
Skipping line 41589: expected 4 fields, saw 11
Skipping line 51628: expected 4 fields, saw 7
Skipping line 114465: expected 4 fields, saw 5

Skipping line 142496: expected 4 fields, saw 8
Skipping line 189732: expected 4 fields, saw 6
Skipping line 245218: expected 4 fields, saw 7

Skipping line 388430: expected 4 fields, saw 5

C:\Users\PC\AppData\Local\Temp\ipykernel_10728\1061535121.py:1: DtypeWarning: Columns (2,3) have mixed types. Specify
dtype option on import or set low_memory=False.
  df = pd.read_csv("UScomments.csv", error_bad_lines=False)  # Keeping error_bad_lines as false to skip the
```

Need to clean data while checking for duplicate and empty rows

In [3]:
```
df.info()  #checking the details of the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 691400 entries, 0 to 691399
Data columns (total 4 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   video_id      691400 non-null  object
 1   comment_text  691375 non-null  object
 2   likes         691400 non-null  object
 3   replies       691400 non-null  object
dtypes: object(4)
memory usage: 21.1+ MB
```

In [4]:
```
df.isnull().sum()   # checking for total null values per column
```

Out[4]:
```
video_id         0
comment_text    25
likes            0
replies          0
dtype: int64
```

In [5]:
```
df.dropna(inplace=True)    #dropping all the rows with null values
```

In [6]:
```
df.isnull().sum()         # verifying whether we got any null values or not
```

Out[6]:
```
video_id        0
comment_text    0
likes           0
replies         0
dtype: int64
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 691375 entries, 0 to 691399
Data columns (total 4 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   video_id      691375 non-null  object
 1   comment_text  691375 non-null  object
 2   likes         691375 non-null  object
 3   replies       691375 non-null  object
dtypes: object(4)
memory usage: 26.4+ MB
```

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

Out[8]:

	video_id	comment_text	likes	replies
0	XpVt6Z1Gjjo	Logan Paul it's yo big day !!!!!	4	0
1	XpVt6Z1Gjjo	I've been following you from the start of your...	3	0
2	XpVt6Z1Gjjo	Say hi to Kong and maverick for me	3	0
3	XpVt6Z1Gjjo	MY FAN . attendance	3	0
4	XpVt6Z1Gjjo	trending 😉	3	0

In [9]:
```
from textblob import TextBlob as tb  # importing textblob to serigate the positive and negitive comments
```

polarity will give us values between -1(negitive) to 0(neutral) to 1(Positive) values based on the polarity we can confirm whether a comment is positive or not

In [10]:
```
tb("Logan Paul it's yo big day !!!!!").sentiment.polarity   # Testing textBlob
```

Out[10]:
```
0.0
```

In [11]:
```
polarity = []
for i in df["comment_text"]:
    polarity.append(tb(i).sentiment.polarity)  # Appending polarity values of each comment in to a list
```

In [13]:
```
polarity
```

Out[13]:
```
[0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.8,
 -0.13571428571428573,
 0.0,
 0.2,
 -0.023333333333333352,
 0.5,
 0.0,
 0.8,
 -0.2916666666666667,
 0.0,
 0.25,
```

In [14]:
```
df["polarity"]=polarity
```

In [15]:
```
positive_comment_filter = df['polarity']==1   # seperating the comments which are positive
```

In [16]:
```
negitive_comment_filter = df['polarity']==-1   # seperating the comments which are negitive
```

In [17]:
```
import wordcloud
from wordcloud import STOPWORDS, WordCloud     # importing wordcloud to seperate the words which dosent have sentiment
```

In [18]:
```
set(STOPWORDS)
```

```
 'each',
 'else',
 'ever',
 'few',
 'for',
 'from',
 'further',
 'get',
 'had',
 "hadn't",
 'has',
 "hasn't",
 'have',
 "haven't",
 'having',
 'he',
 "he'd",
 "he'll",
 "he's",
 'hence'
```

In [19]:
```
positive_comment = df[positive_comment_filter]   # Getting comments which are positive
```

In [20]:
```
negitive_comment = df[negitive_comment_filter]   # Getting comments which are positive
```

In [21]:
```
total_positive_comment = ' '.join(positive_comment['comment_text'])   # Converting all the comments to strings
```

In [22]:
```
total_negitive_comment = ' '.join(negitive_comment['comment_text'])   # Converting all the comments to Strings
```

In [23]:
```
wordcloud = WordCloud(stopwords=set(STOPWORDS)).generate(total_positive_comment)  # generating wordcloud with all the
```

In [24]:
```
plt.imshow(wordcloud)      # showing wordcloud
plt.axis(False)
```

Out[24]:
```
(-0.5, 399.5, 199.5, -0.5)
```

Here are the most used positive words in a poster

In [25]:
```
wordcloud = WordCloud(stopwords=set(STOPWORDS)).generate(total_negitive_comment) # generating wordcloud with all the p
```

In [26]:
```
plt.imshow(wordcloud)
plt.axis(False)
```

Out[26]:
```
(-0.5, 399.5, 199.5, -0.5)
```

Here are the most used negitive words in a poster

2. Check top 10 most usedemoji's in the comments.

In [27]:
```
import emoji               # importing the emoji to handle emojis
from emoji import EMOJI_DATA
```

In [28]:
```
emo = []
for i in df['comment_text']:                    # seperating all the emojis from comments
    if i in emoji.EMOJI_DATA:
        emo.append(i)
```

In [29]:
```
from collections import Counter
```

In [30]:
```
a = []
for i in range(10):                                    # Seperating top 10 repeted emoji's(just fig)
    a.append(Counter(emo).most_common(10)[i][0])
```

In [31]:
```
a           # Most used emoji's
```

Out[31]: ['💔', '❤', '👍', '😍', '😂', '♥', '💔', '💜', '🔥', '🐍']

In [32]:
```
c = []
for i in range(10):                                   # Seperating top 10 repeted emoji's(Just counts)
    c.append(Counter(emo).most_common(10)[i][1])
```

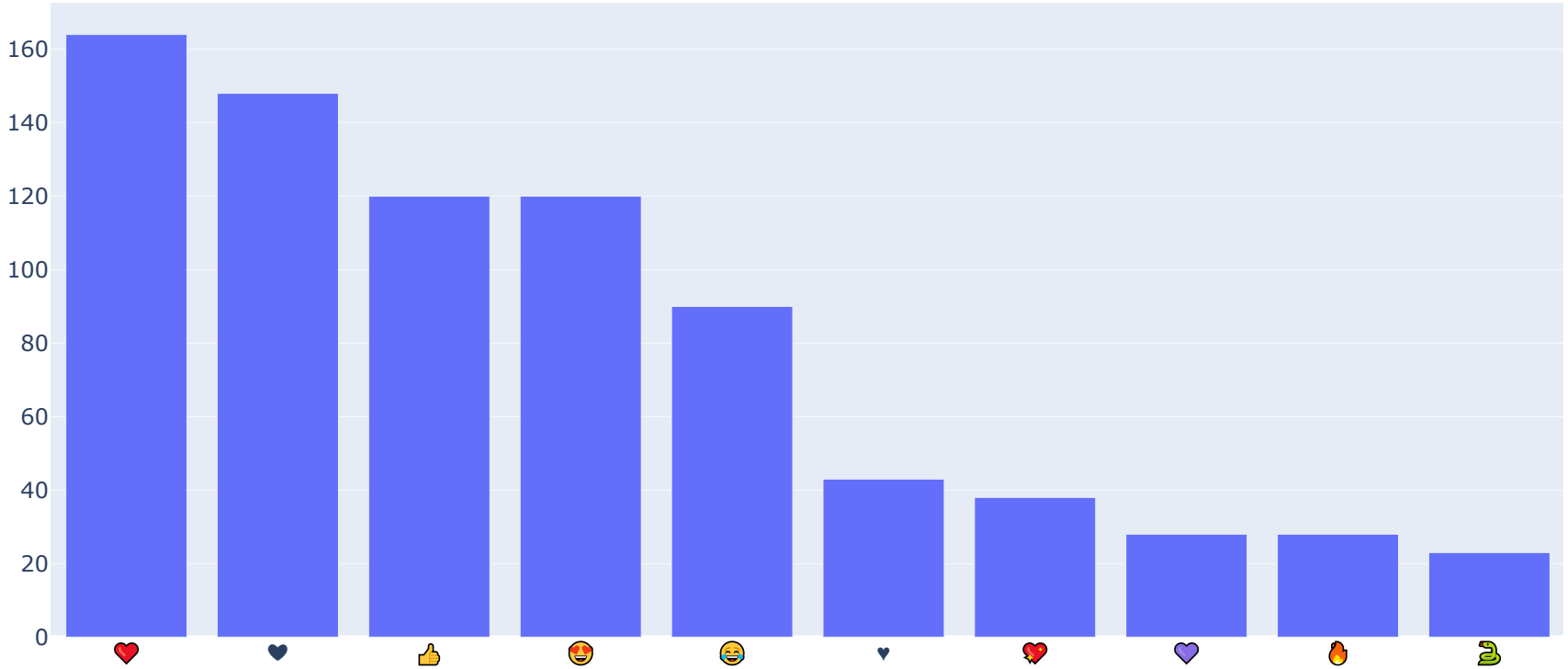
In [33]:
```
c
```

Out[33]: [164, 148, 120, 120, 90, 43, 38, 28, 28, 23]

In [34]:
```
import plotly.graph_objs as go
from plotly.offline import iplot              # importing plotty to plot a bar graph to compare the counts of top 10 em
```

In [35]:
```
trace = go.Bar(x=a, y=c)
```

In [36]:
```
iplot([trace])
```



Here is a report of top 10 most used emoji's