YOUTUBE ANALYSIS PROJECT

Introduction:

YouTube, founded in 2005, has emerged as the dominant force in online video sharing, boasting over 2 billion monthly users. This case study focuses on text analysis within the YouTube ecosystem, aiming to uncover trends, sentiments, and user behaviors. By examining comments, video descriptions, and titles, we seek to understand audience engagement, content dynamics, and the platform's societal impact. Through this analysis, we aim to provide insights into YouTube's role in shaping online culture and its implications for content creators and society at large.

Study case:

This case study focuses on analyzing text data from YouTube content in four main areas. Firstly, it examines positive sentiment to understand the prevalence of positive reactions in comments and titles. Secondly, it studies emoji usage to gauge user emotions and engagement. Thirdly, it investigates the relationship between dislikes and views to uncover factors impacting engagement. Lastly, it analyzes trending video text to identify common themes and strategies used by creators to attract viewers. These analyses provide insights into user behavior and content dynamics on YouTube.

- ➢ Positive Sentiment Analysis
- ➢ Emoji Analysis
- ➢ Dislikes vs views Analysis
- ➢ Trending Video Text on YouTube

Step1: How to read csv data or how to load data.

- Here we are performing ETL pipeline which means to extract Data, Transform Data , Load Data.
- We have Raw data which includes null values and duplicate values , so we need to clean the data and then we will get featured data.
- We have 5 modules to import data and to perform analysis.

  - ✓ pandas- mainly used to read data, modify data, Manipulate data.
  - ✓ Numpy - used for mathematical purpose like to find mean, median, percentile
  - ✓ Matplotlib- base module for visualization purpose.
  - ✓ Seaborn – used for Data visualization
  - ✓ plotly – used for Data visualization

*** import all the modules**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

***read the file:**

comments = pd.read_csv(r'File path' , error_bad_lines=False)

- ⇨ r is used to reduce Unicode error.
- ⇨ r makes the normal string to raw string to avoid problems with different os(Mac/windows)
- ⇨ Error_bad_lines = False is used to ignore unwanted message while running.

***To check if there is any null values or not :**

comments.isnull().sum()

- ⇨ if there is less number of null values then we can drop null values using below command.
- ⇨ comments.dropna(inplace=True)

Step:2: Perform sentiment analysis:

- Sentiment Analysis means Analysis Sentiment of user
- Polarity ranges between [-1, +1]
- We need to analysis the comment and need to create a column with polarity of comment.
- For this we are installing textblob

  !pip install textblob

  from textblob import TextBlob

Step:3: Perform Wordcloud analysis for data:

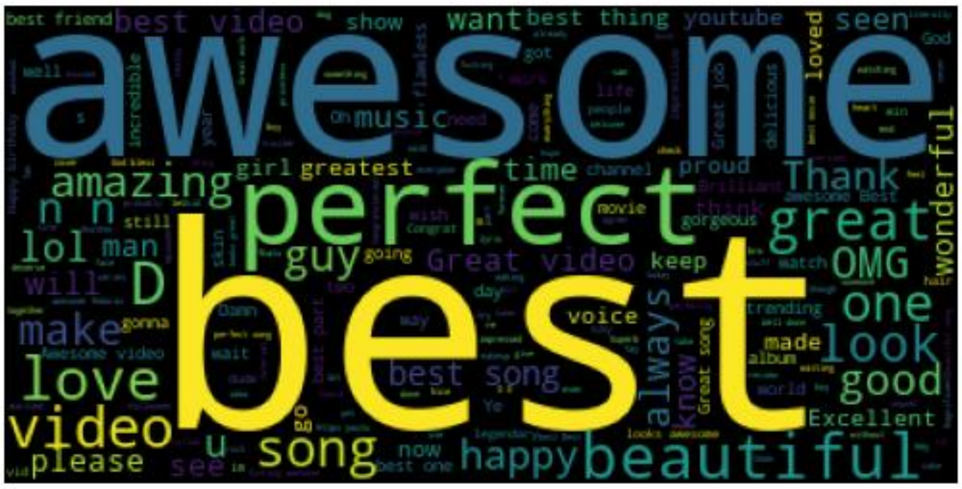
- Graphical representation of text frequency
- Most important keyword
- Word cloud contain all positive keywords
- We will divide into two database based on polarity 1 , -1.
- To create word cloud we need to install wordcloud.

  !pip install wordcloud

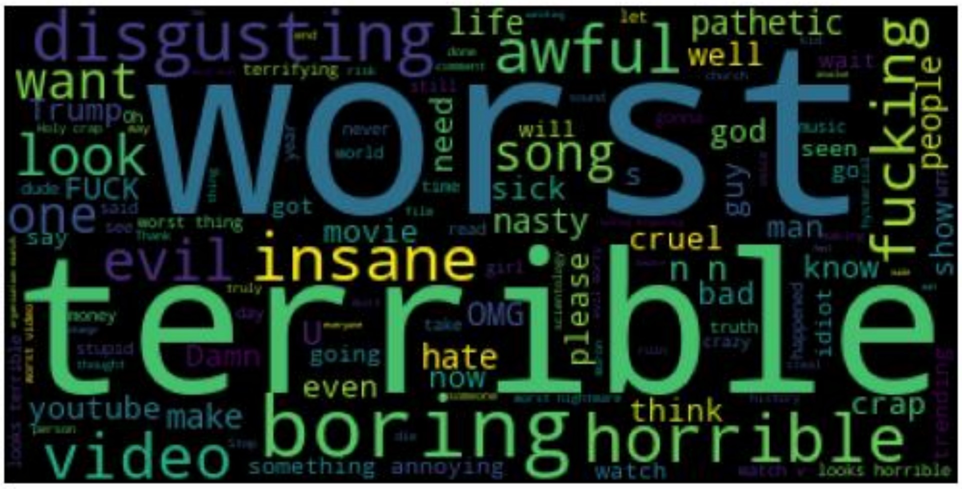
  from wordcloud import WordCloud , STOPWORDS

- Stopwords used to neglect meaning less words.

- **Positive Sentiment Analysis**



- **Negative Sentiment Analysis**



Step4: perform Emoji Analysis:

- It is a part of EDA process
- We need to collect the Emoji counts and need to perform visualization of count of emoji's with bar chat
- For this we need to install emoji version 2.2.0

```
!pip install emoji==2.2.0

import emoji
```

- to get emoji from comments , we use below code

```
all_emojis_list = []
```

```
for comment in comments['comment_text'].dropna(): ## in case u have missing values
, call dropna()

    for char in comment:

        if char in emoji.EMOJI_DATA:

            all_emojis_list.append(char)
```

- To collect all top 10 emoji we will use counter from collections

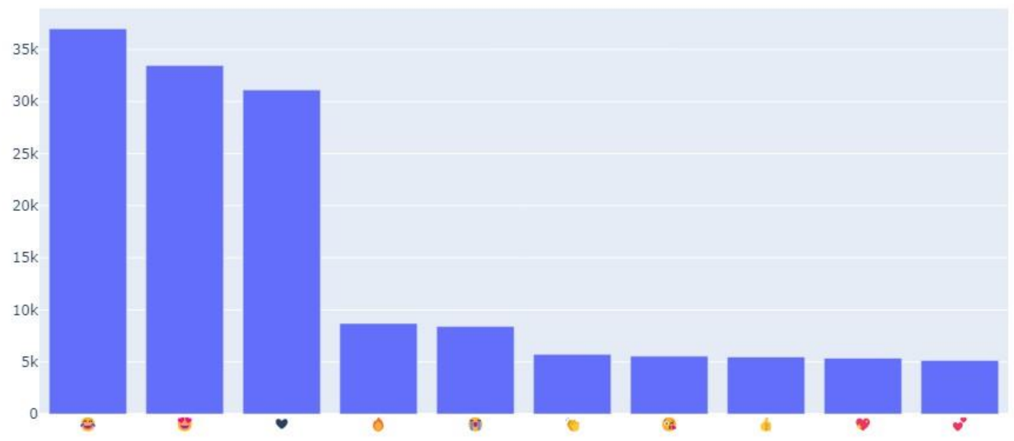
```
from collections import Counter

Counter(all_emojis_list).most_common(10)
```

- To plot graph we will divide emoji and count of emoji as two rows

```
emojis = [Counter(all_emojis_list).most_common(10)[i][0] for i in range(10)]

freqs = [Counter(all_emojis_list).most_common(10)[i][1] for i in range(10)]
```

- To plot graph we will import pliotly

```
import plotly.graph_objs as go

from plotly.offline import iplot

trace = go.Bar(x=emojis , y=freqs)

iplot([trace])
```



Step5: collect entire data of youtube:

- Collect entire data from India, us etc; to do data transformation, data cleaning, data featurization

  import os

  files= os.listdir(r'File path')

  files

  files_csv = [file for file in files if '.csv' in file]

  files_csv

Step6: How to export data into csv,json,db:

- We have raw data and we need to check duplicated, Null values etc,

  full_df[full_df.duplicated()].shape

  full_df = full_df.drop_duplicates() ## lets drop duplicate rows ..

  full_df.shape

- Storing data into csv ..

  full_df[0:1000].to_csv(r'file path.csv' , index=False) # you can consider sample of data depending on how efficient your system is..

- Storing data into json

  full_df[0:1000].to_json(r'file path.json')

- Storing data into database

  from sqlalchemy import create_engine

  engine = create_engine(r'sqlite:///youtube_sample.sqlite')

  full_df[0:1000].to_sql('Users' , con=engine , if_exists='append')

Step7: Which category has maximum likes:

- This code is a common way to enrich a Data Frame with additional information from an external source. By mapping category IDs to category names, you make the data more readable and interpretable.

```
full_df['category_id'].unique()

json_df = pd.read_json(r'file path.json')

json_df

cat_dict = {}

for item in json_df['items'].values:

 ## cat_dict[key] = value (Syntax to insert key:value in dictionary)

cat_dict[int(item['id'])] = item['snippet']['title']

full_df['category_name'] = full_df['category_id'].map(cat_dict)
```

Step8: To find out whether audience are engaged or not:

- These calculations can help provide insights into the engagement levels of the items in your dataset by normalizing the likes, dislikes, and comment counts by the total views.

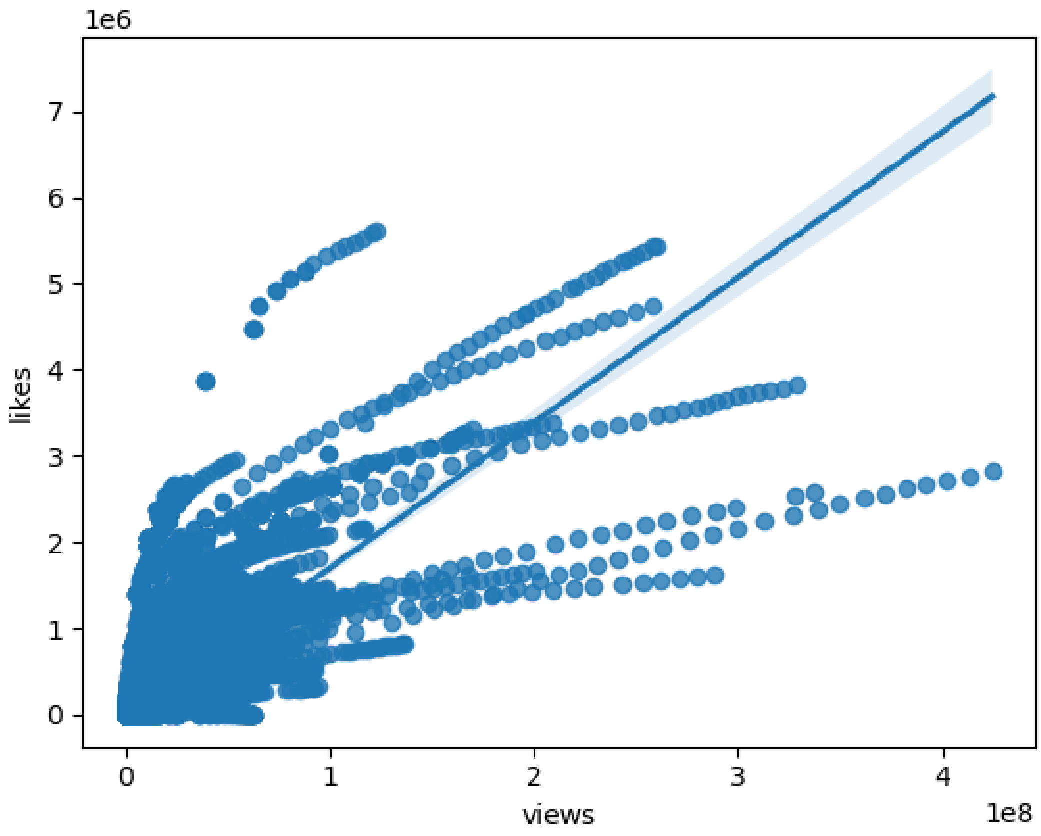
```
full_df['like_rate'] = (full_df['likes']/full_df['views'])*100

full_df['dislike_rate'] = (full_df['dislikes']/full_df['views'])*100

full_df['comment_count_rate'] = (full_df['comment_count']/full_df['views'])*100
```

- to get boxplot between category and like rate

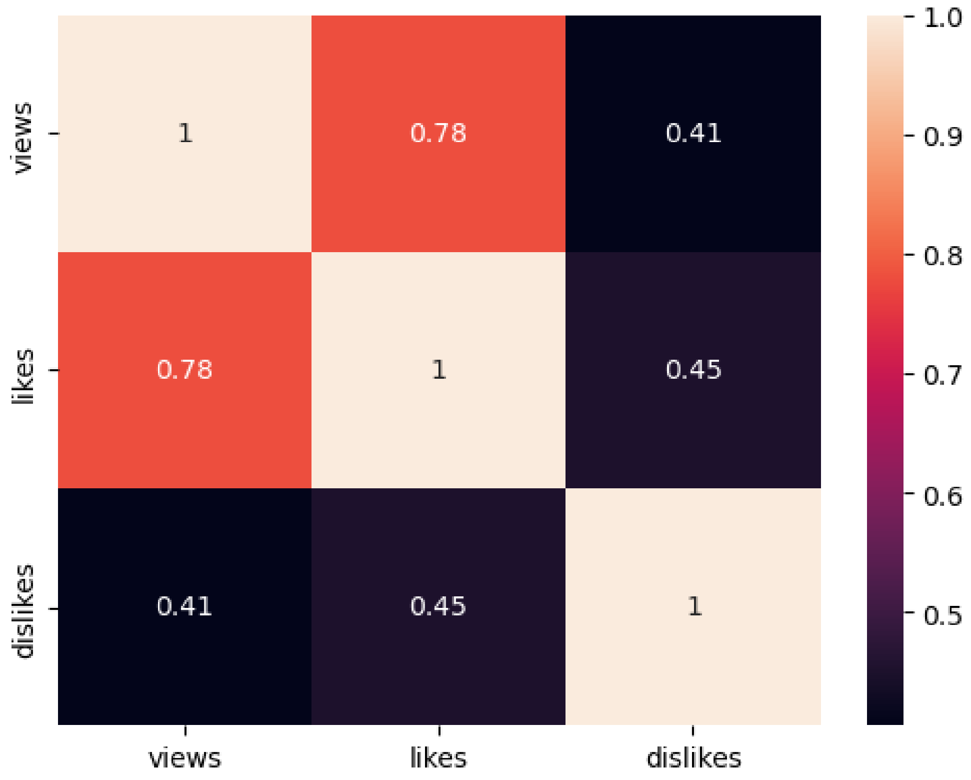
```
plt.figure(figsize=(8,6))

sns.boxplot(x='category_name' , y='like_rate' , data=full_df)

plt.xticks(rotation='vertical')

plt.show()
```

- regression plot between views and likes

```
sns.regplot(x='views' , y='likes' , data = full_df)
```



- Heap map to check correlation between views, likes, dislikes



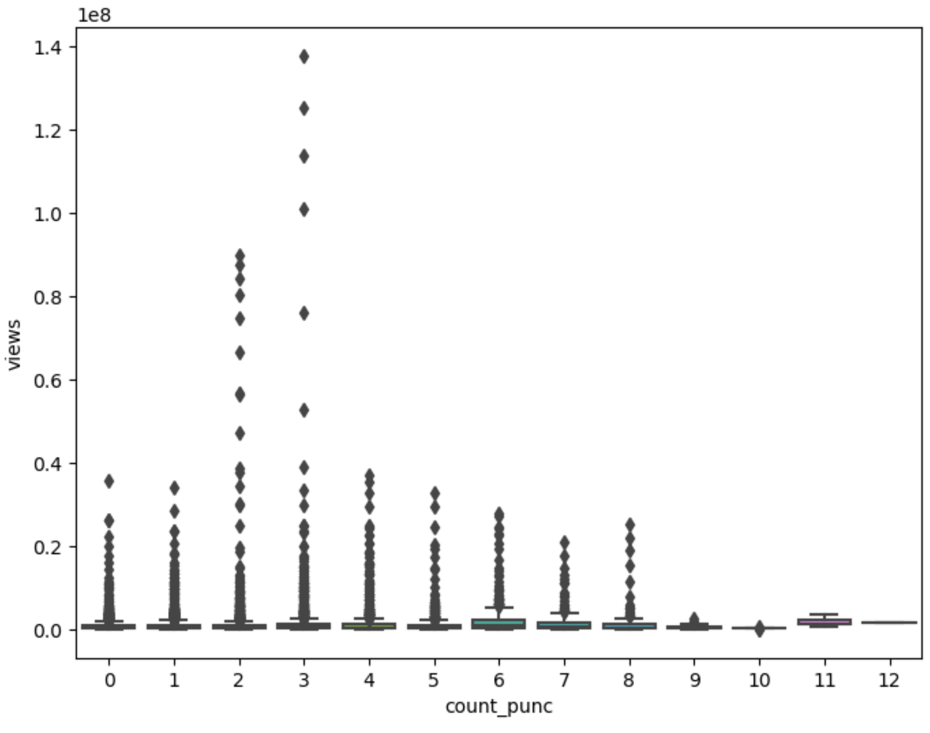
Step9: Which channels have the largest number of trending videos:

```
full_df['channel_title'].value_counts()

cdf = full_df.groupby(['channel_title']).size().sort_values(ascending=False).reset_index()

cdf = cdf.rename(columns={0:'total_videos'})
```

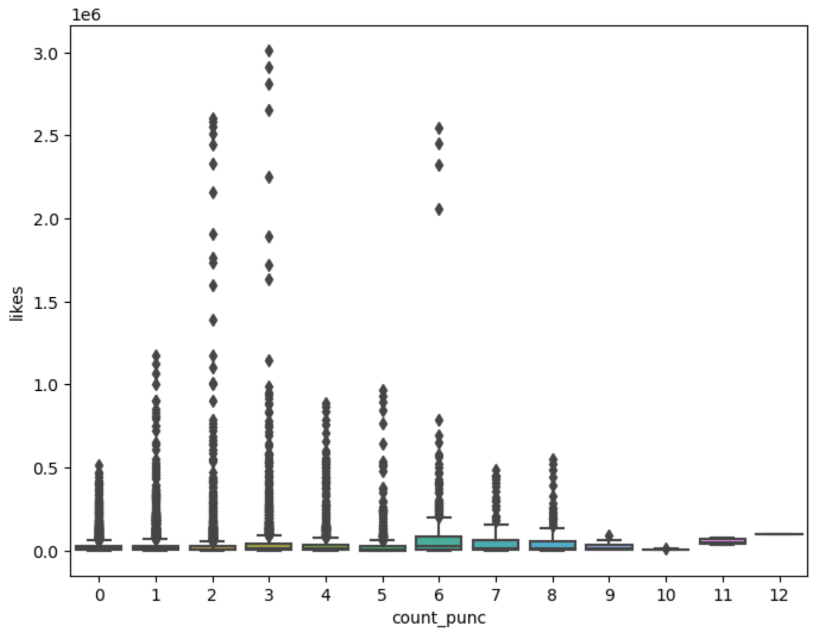
Step10:Does punctuations in title and tags have any relation with views, likes, dislikes, comments?

```
import string

string.punctuation

len([char for char in full_df['title'][0] if char in string.punctuation])

def punc_count(text):

    return len([char for char in text if char in string.punctuation])

sample = full_df[0:10000]

sample['count_punc'] = sample['title'].apply(punc_count)

sample['count_punc']
```

- Boxplot for count punctuations and views



- Boxplot for count punctuations and likes


1e6
likes
count_punc