Midterm_Youtube_API_Project

January 6, 2024

1 Data Programming Project: Decoding Success on YouTube through the Lens of Mr. Beast's Journey

1.1 1. Introduction to research topic

This research investigates the transformative journey of Mr. Beast on YouTube, from a humble start in his room to becoming a multi-millionaire with a massive fan base. The significance lies in providing a valuable guide for content creators to enhance their appeal, foster engagement, and navigate the competitive landscape of YouTube, drawing inspiration from Mr. Beast's remarkable ascent in the digital realm.

1.2 1.1 Why this field of interest

As technology advances, achieving fame and amassing a net worth of over 20 billion dollars through an online platform was previously unimaginable in our society. It is fascinating to analyze how he translates his thought processes into videos that capture global attention and generate such an astronomical income.

1.3 1.2 Topic has been poorly explored previously

While MrBeast has become a global sensation, the existing articles and analyses on the internet have been lacking in-depth exploration and often rely on limited statistical data (such as only analyzing five videos) or provide only brief textual explanations of his success without substantial supporting evidence.

- https://techcrunch.com/2023/12/14/mrbeasts-analytics-platform-viewstats-is-out-in-beta/
- https://medium.com/@pjvillasista/leveraging-data-analytics-on-mrbeasts-youtube-videos-a-deep-dive-into-engagement-sentiment-and-3fa21d4fcbea
- https://www.tubefilter.com/2023/12/13/view-stats-analytics-statistics-mrbeast-jimmy-donaldson-chucky-appleby/

1.4 1.3 Scope of Work

In this data analysis project, the following tasks and analyses will be undertaken: * Analysis of Top 20 Videos by View Count * Analysis of Bottom 20 Videos by View Count * Distribution of View Count * Correlation of Comment Count and Like Count with View Count * Correlation of Top 10 Comments per Video with Video Title * Channel Growth Analysis * Correlation of Published Day Frequency with View Count

1.5 1.3.1 Steps and Stages in Analytical Data Processing Pipeline

1. Acquiring Dataset
2. Data pre-Processing
3. Data Analysis
4. Data Visualization
5. Conclusion
6. References

1.6 1.4 Aims & Objective

The focus is on decoding the key strategies and content approaches that propelled his success, as well as analyzing audience engagement patterns and growth factors. By employing a mixed-methods approach, including qualitative analysis of Mr. Beast's videos and quantitative data on subscriber growth and engagement metrics, the study aims to distill practical insights for aspiring YouTubers.

1.7 1.4.1 Benefits of Analysis

This analysis holds valuable advantages, not only for Mr. Beast himself but also for a broader audience interested in YouTube content creation.

Mr. Beast can leverage this analysis to identify trends within his channel. By recognizing these patterns, he can make informed decisions about the content of his future videos. This, in turn, can enhance audience engagement and satisfaction, ultimately contributing to the growth of his channel.

Individuals looking to embark on a YouTube career or those already established in the YouTube scene can find immense value in this analysis. By delving into the statistics of a successful channel like Mr. Beast's, they gain valuable insights and learn from the experiences of others. This knowledge can significantly expedite their path to success and help them achieve their goals more efficiently.

1.8 1.5 Acquiring Dataset

To ensure the reliability and legality of our dataset, this research will utilize the official YouTube API in conjunction with the Python request library. Leveraging the official API not only guarantees the authenticity of the data but also ensures compliance with YouTube's terms of service. This ethical and lawful approach not only reinforces the integrity of our research but also provides a robust foundation for a thorough analysis of Mr. Beast's YouTube journey. By adhering to official channels for data acquisition, we prioritize accuracy and respect the platform's guidelines, contributing to the credibility of our findings.

1.9 1.6 Utilizing of Dataset

I will be importing necessary libraries such as pandas and seaborn for data manipulation and data visualization

1.10 2. Installation of Google API Python Client Library

[1]:
```
!pip install --upgrade google-api-python-client
!pip install pandas
!pip install python-dateutil
!pip install requests
!pip install nltk
!pip install emoji
!pip install seaborn
!pip install matplotlib
!pip install wordcloud
!pip install langdetect
!pip install tqdm
!pip install isodate
```

```
Requirement already satisfied: google-api-python-client in
/Users/roscoe/myenv/lib/python3.11/site-packages (2.112.0)
Requirement already satisfied: httplib2<1.dev0,>=0.15.0 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from google-api-python-client)
(0.22.0)
Requirement already satisfied: google-auth<3.0.0.dev0,>=1.19.0 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from google-api-python-client)
(2.26.1)
Requirement already satisfied: google-auth-httplib2>=0.1.0 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from google-api-python-client)
(0.2.0)
Requirement already satisfied: google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from google-api-python-client)
(2.15.0)
Requirement already satisfied: uritemplate<5,>=3.0.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from google-api-python-client)
(4.1.1)
Requirement already satisfied: googleapis-common-protos<2.0.dev0,>=1.56.2 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-
client) (1.62.0)
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.2
1.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0.dev0,>=3.19.5 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-
client) (4.25.1)
Requirement already satisfied: requests<3.0.0.dev0,>=2.18.0 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-
client) (2.31.0)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from google-
auth<3.0.0.dev0,>=1.19.0->google-api-python-client) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from google-
auth<3.0.0.dev0,>=1.19.0->google-api-python-client) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from google-
auth<3.0.0.dev0,>=1.19.0->google-api-python-client) (4.9)
Requirement already satisfied:
pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
httplib2<1.dev0,>=0.15.0->google-api-python-client) (3.1.1)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
pyasn1-modules>=0.2.1->google-auth<3.0.0.dev0,>=1.19.0->google-api-python-
client) (0.5.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
requests<3.0.0.dev0,>=2.18.0->google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-
client) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
requests<3.0.0.dev0,>=2.18.0->google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-
client) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
requests<3.0.0.dev0,>=2.18.0->google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-
client) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
requests<3.0.0.dev0,>=2.18.0->google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-
client) (2023.11.17)
Requirement already satisfied: pandas in
/Users/roscoe/myenv/lib/python3.11/site-packages (2.1.4)
Requirement already satisfied: numpy<2,>=1.23.2 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from pandas) (1.26.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from pandas) (2023.4)
Requirement already satisfied: six>=1.5 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from python-
dateutil>=2.8.2->pandas) (1.16.0)
```

```
Requirement already satisfied: python-dateutil in
/Users/roscoe/myenv/lib/python3.11/site-packages (2.8.2)
Requirement already satisfied: six>=1.5 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from python-dateutil) (1.16.0)
Requirement already satisfied: requests in
/Users/roscoe/myenv/lib/python3.11/site-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from requests) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from requests) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from requests) (2023.11.17)
Requirement already satisfied: nltk in /Users/roscoe/myenv/lib/python3.11/site-
packages (3.8.1)
Requirement already satisfied: click in /Users/roscoe/myenv/lib/python3.11/site-
packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in
/Users/roscoe/myenv/lib/python3.11/site-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from nltk) (2023.12.25)
Requirement already satisfied: tqdm in /Users/roscoe/myenv/lib/python3.11/site-
packages (from nltk) (4.66.1)
Requirement already satisfied: emoji in /Users/roscoe/myenv/lib/python3.11/site-
packages (2.9.0)
Requirement already satisfied: seaborn in
/Users/roscoe/myenv/lib/python3.11/site-packages (0.13.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from seaborn) (1.26.3)
Requirement already satisfied: pandas>=1.2 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from seaborn) (2.1.4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from seaborn) (3.8.2)
Requirement already satisfied: contourpy>=1.0.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (4.47.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (23.2)
Requirement already satisfied: pillow>=8 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from pandas>=1.2->seaborn)
(2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from pandas>=1.2->seaborn)
(2023.4)
Requirement already satisfied: six>=1.5 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from python-
dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Requirement already satisfied: matplotlib in
/Users/roscoe/myenv/lib/python3.11/site-packages (3.8.2)
Requirement already satisfied: contourpy>=1.0.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib) (4.47.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy<2,>=1.21 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib) (1.26.3)
Requirement already satisfied: packaging>=20.0 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=8 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from python-
dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: wordcloud in
/Users/roscoe/myenv/lib/python3.11/site-packages (1.9.3)
Requirement already satisfied: numpy>=1.6.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from wordcloud) (1.26.3)
Requirement already satisfied: pillow in
/Users/roscoe/myenv/lib/python3.11/site-packages (from wordcloud) (10.2.0)
```

```
Requirement already satisfied: matplotlib in
/Users/roscoe/myenv/lib/python3.11/site-packages (from wordcloud) (3.8.2)
Requirement already satisfied: contourpy>=1.0.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib->wordcloud)
(1.2.0)
Requirement already satisfied: cycler>=0.10 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib->wordcloud)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib->wordcloud)
(4.47.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib->wordcloud)
(1.4.5)
Requirement already satisfied: packaging>=20.0 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib->wordcloud)
(23.2)
Requirement already satisfied: pyparsing>=2.3.1 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib->wordcloud)
(3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from matplotlib->wordcloud)
(2.8.2)
Requirement already satisfied: six>=1.5 in
/Users/roscoe/myenv/lib/python3.11/site-packages (from python-
dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Requirement already satisfied: langdetect in
/Users/roscoe/myenv/lib/python3.11/site-packages (1.0.9)
Requirement already satisfied: six in /Users/roscoe/myenv/lib/python3.11/site-
packages (from langdetect) (1.16.0)
Requirement already satisfied: tqdm in /Users/roscoe/myenv/lib/python3.11/site-
packages (4.66.1)
Requirement already satisfied: isodate in
/Users/roscoe/myenv/lib/python3.11/site-packages (0.6.1)
Requirement already satisfied: six in /Users/roscoe/myenv/lib/python3.11/site-
packages (from isodate) (1.16.0)
```

[35]:
```python
# Run the pip freeze command to generate the requirements.txt file in the␣
↪target directory
!pip freeze > requirements.txt

# Read and print the contents of the requirements.txt file
with open("requirements.txt", "r") as file:
    requirements_content = file.read()
    print("Contents of requirements.txt:")
    print(requirements_content)
```

```
Contents of requirements.txt:
```

```
anyio==4.2.0
appnope==0.1.3
argon2-cffi==23.1.0
argon2-cffi-bindings==21.2.0
arrow==1.3.0
asttokens==2.4.1
async-lru==2.0.4
attrs==23.2.0
Babel==2.14.0
beautifulsoup4==4.12.2
bleach==6.1.0
cachetools==5.3.2
certifi==2023.11.17
cffi==1.16.0
charset-normalizer==3.3.2
click==8.1.7
comm==0.2.1
contourpy==1.2.0
cycler==0.12.1
debugpy==1.8.0
decorator==5.1.1
defusedxml==0.7.1
emoji==2.9.0
executing==2.0.1
fastjsonschema==2.19.1
fonttools==4.47.0
fqdn==1.5.1
google-api-core==2.15.0
google-api-python-client==2.112.0
google-auth==2.26.1
google-auth-httplib2==0.2.0
googleapis-common-protos==1.62.0
httplib2==0.22.0
idna==3.6
ipykernel==6.28.0
ipython==8.19.0
ipywidgets==8.1.1
isodate==0.6.1
isoduration==20.11.0
jedi==0.19.1
Jinja2==3.1.2
joblib==1.3.2
json5==0.9.14
jsonpointer==2.4
jsonschema==4.20.0
jsonschema-specifications==2023.12.1
jupyter==1.0.0
jupyter-console==6.6.3
```

```
jupyter-events==0.9.0
jupyter-lsp==2.2.1
jupyter_client==8.6.0
jupyter_core==5.7.0
jupyter_server==2.12.2
jupyter_server_terminals==0.5.1
jupyterlab==4.0.10
jupyterlab-widgets==3.0.9
jupyterlab_pygments==0.3.0
jupyterlab_server==2.25.2
kiwisolver==1.4.5
langdetect==1.0.9
MarkupSafe==2.1.3
matplotlib==3.8.2
matplotlib-inline==0.1.6
matplotlib-venn==0.11.9
mistune==3.0.2
nbclient==0.9.0
nbconvert==7.14.0
nbformat==5.9.2
nest-asyncio==1.5.8
nltk==3.8.1
notebook==7.0.6
notebook_shim==0.2.3
numpy==1.26.3
overrides==7.4.0
packaging==23.2
pandas==2.1.4
pandocfilters==1.5.0
parso==0.8.3
pexpect==4.9.0
pillow==10.2.0
platformdirs==4.1.0
prometheus-client==0.19.0
prompt-toolkit==3.0.43
protobuf==4.25.1
psutil==5.9.7
ptyprocess==0.7.0
pure-eval==0.2.2
pyasn1==0.5.1
pyasn1-modules==0.3.0
pycparser==2.21
Pygments==2.17.2
pyparsing==3.1.1
python-dateutil==2.8.2
python-json-logger==2.0.7
pytz==2023.3.post1
PyYAML==6.0.1
```

```
pyzmq==25.1.2
qtconsole==5.5.1
QtPy==2.4.1
referencing==0.32.0
regex==2023.12.25
requests==2.31.0
rfc3339-validator==0.1.4
rfc3986-validator==0.1.1
rpds-py==0.16.2
rsa==4.9
scipy==1.11.4
seaborn==0.13.1
Send2Trash==1.8.2
six==1.16.0
sniffio==1.3.0
soupsieve==2.5
stack-data==0.6.3
terminado==0.18.0
tinycss2==1.2.1
tornado==6.4
tqdm==4.66.1
traitlets==5.14.1
types-python-dateutil==2.8.19.14
tzdata==2023.4
uri-template==1.3.0
uritemplate==4.1.1
urllib3==2.1.0
wcwidth==0.2.12
webcolors==1.13
webencodings==0.5.1
websocket-client==1.7.0
widgetsnbextension==4.0.9
wordcloud==1.9.3
```

1.11 2.1 Importing Essential Libraries for Analysis

[2]:
```
#Google API-related libraries
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError

#Data manipulation and analysis libraries
import pandas as pd

#Date parsing library
from dateutil import parser
```

```
#Web scraping libraries
import requests  # Requests for making HTTP requests

#Natural Language Processing (NLP) libraries
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

#Emoji library
import emoji

#Regular expression library
import re

#JSON manipulation library
import json

#Data visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib
import matplotlib.ticker as ticker
from matplotlib_venn import venn2
from wordcloud import WordCloud  # WordCloud for creating word clouds

#Language detection library
from langdetect import detect, LangDetectException

#Provides a progress meter for loops
from tqdm import tqdm
#Provides time-related functions
import time
#Parses ISO 8601 dates
import isodate
#Helps safely evaluate strings containing Python expressions
import ast
```

1.12 2.2 Dataset Comparison

[3]:
```
#An overview of the second dataset in panda
Second_Dataset = pd.read_csv('MrBeast_Youtube_Stats.csv')
Second_Dataset
```

[3]:
```
             id                                       title  \
0   TQHEJj68Jew             I Got Hunted By A Real Bounty Hunter
1   00NgUctWoLQ                 Extreme $1,000,000 Hide And Seek
```

```
2           NaN                                            MrBeast
3    ayXxwJJId_c  I Bought The World&#39;s Largest Mystery Box! …
4    cExLQ1o2pDw                    First To Rob Bank Wins $100,000
..           …                                                  …
242  yeHjsYQ076A                         Remember When COD Was Fun?
243  X7doE4h6W64                  Insane Gun Sync - 7 Hours To Make
244  AIytwdufwW8   MY MESSAGE TO COD YOUTUBERS (Watch till end plz)
245  56OkH6ocYZU    L0114R - Biblical Creeper Post for Post @L0114R
246  zi7tZ-2PhLk  How Much Money Do You Make As An Uber Driver??…

                                           description  \
0    Sign up for Current w/ my Creator Code "BEAST"…
1    I didn't expect that to happen at the end I wa…
2    Accomplishments - Raised $20000000 To Plant 20…
3    I cant believe I spent over $500000 on mystery…
4    I didnt think he would actually rob the bank…
..                                                …
242  Yup, another very pointless video  Subscribe?…
243  Look at this - http://gyazo.com/8852509d2350db…
244  The beginning of the video is a little off top…
245  His channel - https://www.youtube.com/user/the…
246  In this video I talk about being an uber drive…

                   publishTime     kind_stats  duration_seconds    viewCount  \
0    2021-04-24 20:00:00+00:00  youtube#video               861   84717282.0
1    2021-12-18 21:00:00+00:00  youtube#video               729   32090178.0
2    2012-02-20 00:43:50+00:00            NaN                 0          NaN
3    2021-04-03 20:00:01+00:00  youtube#video               709  101745632.0
4    2021-09-26 20:00:06+00:00  youtube#video               482   50008942.0
..                         …              …                 …            …
242  2015-04-26 21:26:36+00:00  youtube#video               216      16312.0
243  2015-06-21 12:22:34+00:00  youtube#video               134      15740.0
244  2015-05-29 20:07:48+00:00  youtube#video               292      18502.0
245  2015-05-15 16:48:54+00:00  youtube#video               109      16627.0
246  2015-05-31 18:31:25+00:00  youtube#video               304      18773.0

     likeCount  commentCount  \
0    2876493.0      128922.0
1    2125183.0       73593.0
2          NaN           NaN
3    3110824.0      162796.0
4    2359606.0      120621.0
..           …             …
242      574.0         145.0
243      637.0         105.0
244      508.0         117.0
245      430.0         134.0
```

```
246      370.0          59.0

                              thumbnails.default.url  …  \
0       https://i.ytimg.com/vi/TQHEJj68Jew/default.jpg  …
1       https://i.ytimg.com/vi/OONgUctWoLQ/default.jpg  …
2    https://yt3.ggpht.com/ytc/AKedOLTctGKJ32CdDLiS…  …
3       https://i.ytimg.com/vi/ayXxwJJId_c/default.jpg  …
4       https://i.ytimg.com/vi/cExLQ1o2pDw/default.jpg  …
..                                                …  …
242     https://i.ytimg.com/vi/yeHjsYQ076A/default.jpg  …
243     https://i.ytimg.com/vi/X7doE4h6W64/default.jpg  …
244     https://i.ytimg.com/vi/AIytwdufwW8/default.jpg  …
245     https://i.ytimg.com/vi/56OkH6ocYZU/default.jpg  …
246     https://i.ytimg.com/vi/zi7tZ-2PhLk/default.jpg  …

     thumbnails.high.width  thumbnails.high.height contentDetails.duration  \
0                    480.0                   360.0                PT14M21S
1                    480.0                   360.0                 PT12M9S
2                      NaN                     NaN                     NaN
3                    480.0                   360.0                PT11M49S
4                    480.0                   360.0                  PT8M2S
..                     …                     …                     …
242                  480.0                   360.0                 PT3M36S
243                  480.0                   360.0                 PT2M14S
244                  480.0                   360.0                 PT4M52S
245                  480.0                   360.0                 PT1M49S
246                  480.0                   360.0                  PT5M4S

     contentDetails.dimension  \
0                          2d
1                          2d
2                         NaN
3                          2d
4                          2d
..                        …
242                        2d
243                        2d
244                        2d
245                        2d
246                        2d

                          topicDetails.topicCategories  \
0    ['https://en.wikipedia.org/wiki/Lifestyle_(soc…
1    ['https://en.wikipedia.org/wiki/Lifestyle_(soc…
2                                                NaN
3    ['https://en.wikipedia.org/wiki/Lifestyle_(soc…
4    ['https://en.wikipedia.org/wiki/Lifestyle_(soc…
..                                              …
242  ['https://en.wikipedia.org/wiki/Action-adventu…
243  ['https://en.wikipedia.org/wiki/Action-adventu…
244  ['https://en.wikipedia.org/wiki/Action-adventu…
245  ['https://en.wikipedia.org/wiki/Action-adventu…
246  ['https://en.wikipedia.org/wiki/Action-adventu…

    snippet.defaultLanguage            localizations.en.title  \
0                       NaN                               NaN
1                        en  Extreme $1,000,000 Hide And Seek
2                       NaN                               NaN
3                       NaN                               NaN
4                       NaN                               NaN
..                      …                               …
242                     NaN                               NaN
243                     NaN                               NaN
244                     NaN                               NaN
245                     NaN                               NaN
246                     NaN                               NaN

                          localizations.en.description  \
0                                                  NaN
1    I didn't expect that to happen at the end I wa…
2                                                  NaN
3                                                  NaN
4                                                  NaN
..                                                 …
242                                                NaN
243                                                NaN
244                                                NaN
245                                                NaN
246                                                NaN

                                          snippet.tags  \
0                                                  NaN
1                                                  NaN
2                                                  NaN
3                                                  NaN
4                                                  NaN
..                                                 …
242  ['How', 'much', 'money', 'does', 'make', 'blac…
243  ['How', 'much', 'money', 'does', 'make', 'blac…
244  ['How', 'much', 'money', 'does', 'make', 'blac…
245  ['How', 'much', 'money', 'does', 'make', 'blac…
246  ['How', 'much', 'money', 'does', 'make', 'blac…

    contentDetails.contentRating.ytRating
```

```
0                                        NaN
1                                        NaN
2                                        NaN
3                                        NaN
4                                        NaN
..                                       …
242                                      NaN
243                                      NaN
244                                      NaN
245                                      NaN
246                                      NaN

[247 rows x 26 columns]
```

I obtained another dataset mentioned above from Kaggle, which is available for public use. However, it's important to note that this dataset is only up to date until the year 2022 and contains some missing data, particularly regarding video comments. With its outdated nature and missing information, it poses unreliability when identifying recent trends changes in content strategy and audience engagement. Hence, this data would not be used for reference.

The reason behind my decision to utilize the YouTube API to extract data instead of relying solely on the aforementioned dataset is twofold. Firstly, the dataset's irregular update schedule may lead to outdated trends and analyses. Secondly, by using the YouTube API, I have the advantage of accessing data ranging from Mr. Beast's earliest videos to the most recent ones. Furthermore, this approach allows me to selectively extract specific types of data for use in visual analysis, providing greater flexibility in my research.

1.13 2.3 Data Relevance and Origin

The dataset utilized in this project is acquired directly from the Official YouTube API using the Python request library, aligning with the YouTube API documentation for retrieving public data. The choice of using an API key for authentication is intentional, as it caters to the specific needs of the project.

Given that the project focuses on retrieving public information already accessible on the YouTube platform, the decision to opt for an API key over OAuth Authentication is justified. This approach ensures compliance with laws and regulations, as it restricts access to non-sensitive and does not involve making changes on behalf of the YouTube channel. The chosen authentication method aligns seamlessly with the intended scope of the research, emphasizing transparency and adherence to ethical standards.

1.14 2.4 Authentication Key for request to Youtube API

[4]:
```
# API Key essential for Youtube API to recognise and access
API_KEY = ["AIzaSyD1iXZ4T9fEM6DF7bSl6T8RCf_mrCfWhmI",␣
 ↪"AIzaSyBRQUKuNgq02x7cmld0vo4jp31IbEBTC1c"]
youtube = build("youtube", "v3", developerKey=API_KEY[0])
```

[5]:
```python
# Initialize a list to store channel data
channel_data = []

# Request the next page of results
request = youtube.channels().list(
    part="snippet,contentDetails,statistics",
    id="UCX6OQ3DkcsbYNE6H8uQQuVA"
).execute()

# Extract information from the request
for item in request['items']:
    data = {
        'channelName': item['snippet']['title'],
        'subscribers': item['statistics']['subscriberCount'],
        'views': item['statistics']['viewCount'],
        'totalVideos': item['statistics']['videoCount'],
        'playlistID': item['contentDetails']['relatedPlaylists']['uploads']
    }

    # Append data to channel_data
    channel_data.append(data)

# Print the channel data
print(pd.DataFrame(channel_data))
```

```
  channelName subscribers        views totalVideos                playlistID
0     MrBeast   227000000  40453544002         774  UUX6OQ3DkcsbYNE6H8uQQuVA
```

[6]:
```python
# Initialize a list to store Video ID
videoIDs = []
next_page_token = None

while True:
    #Obtain videoID from playlistID
    request = youtube.playlistItems().list(
        part="snippet,contentDetails",
        playlistId="UUX6OQ3DkcsbYNE6H8uQQuVA",
        maxResults = 50,
        pageToken = next_page_token
    ).execute()

    # Extract information from the request
    for item in request['items']:
        videoIDs.append(item['contentDetails']['videoId'])

    next_page_token = request.get('nextPageToken')

    if not next_page_token:
        break
#Print videoIDs
print(pd.DataFrame(videoIDs))
```

```
               0
0    K_CbgLpvH9E
1    lOKASgtr6kU
2    9RhWXPcKBI8
3    ZVt9ZJfWV1c
4    rWBOITBjitE
..           …
769  7qj3nuF9Dzw
770  Y74b7WlcEpk
771  Z8nEEdXTaX0
772  jP82d277Cc8
773  2XVcLrB7B3Y

[774 rows x 1 columns]
```

[7]:
```
# Initialize a list to store Video Data
video_data = []

# Request statistics of video
for i in range(0, len(videoIDs), 50):
    request = youtube.videos().list(
        part="snippet,contentDetails,statistics",
        id=','.join(videoIDs[i:i+50])
    ).execute()

    # Extract information from the request
    for video in request['items']:
        snippet = video['snippet']
        content_details = video['contentDetails']
        statistics = video.get('statistics',{})

        title = snippet['title']
        channelTitle = snippet['channelTitle']
        description = snippet['description']
        tags = snippet.get('tags','NA')
        publishedAt = snippet['publishedAt']

        video_id = video['id']
        duration = content_details['duration']
        definition = content_details['definition']
        caption = content_details['caption']

        view_count = statistics.get('viewCount', 'NA')
        like_count = statistics.get('likeCount', 'NA')
        favourite_count = statistics.get('favouriteCount', 'NA')
        comment_count = statistics.get('commentCount', 'NA')

        #Append data to video_data
        video_data.append({
            'Title': title,
            'Channel Title': channelTitle,
            'Description': description,
            'Tags': tags,
            'Published At': publishedAt,
            'Video ID': video_id,
            'Duration': duration,
            'Definition': definition,
            'Caption': caption,
            'View Count': view_count,
            'Like Count': like_count,
            'Favourite Count': favourite_count,
            'Comment Count': comment_count,
        })

#Print video_data
print(pd.DataFrame(video_data))
```

```
                                                 Title Channel Title  \
0                I Spent 7 Days In Solitary Confinement       MrBeast
1                          I Rescued 100 Abandoned Dogs!       MrBeast
2                Survive 100 Days Trapped, Win $500,000       MrBeast
3                     Feeding A Dog $1 vs $10,000 Steak       MrBeast
4                       Could You Walk Up A Skyscraper?       MrBeast
..                                                 ...           ...
769                 Most Epic minecraft skin EVER  (Psy)       MrBeast
770                           More birds IN MINECRAFT!!       MrBeast
771                      Boxy item mod Minecraft.  EPIC       MrBeast
772  Harry Potter Mod In Minecraft! EPIC MUST SEE M…       MrBeast
773                     Worst Minecraft Saw Trap Ever???       MrBeast

                                           Description  \
0    I started going insane at the end of this chal…
1    I’m so happy all of these dogs will be going t…
2    This video got crazier the longer we kept film…
3
4
..                                                 …
769  Psy in minecraft!!!   drop a like for psy's mo…
770  Basically what this mod does is adds more bird…
771  At the begining i said i was mrbeast6000… i…
772  One of the coolest mods i have ever seen\n\nMo…
```

```
773  This is the worst saw trap ever done in minecr…

                                                       Tags          Published At  \
0                                                        NA  2023-12-30T17:00:03Z
1                                                        NA  2023-12-23T17:00:00Z
2                                                        NA  2023-12-16T17:00:01Z
3                                                        NA  2023-12-14T18:00:00Z
4                                                        NA  2023-12-05T18:00:00Z
..                                                      …                   …
769           [psy, minecraft, epic, skin, most, ever]  2013-01-13T01:59:21Z
770      [birds, minecraft, in, more, must, see, epic]  2013-01-12T23:35:45Z
771                [boxy, item, mod, minecraft, epic]  2013-01-12T22:34:11Z
772  [Harry Potter minecraft, minecraft, minecraft …  2012-03-09T23:29:03Z
773  [minecraft, saw trap, minecraft saw trap, wors…  2012-02-20T22:42:32Z

        Video ID  Duration Definition Caption View Count Like Count  \
0    K_CbgLpvH9E  PT20M16S         hd    true   81908968    3479698
1    1OKASgtr6kU   PT15M3S         hd    true   93302539    4440585
2    9RhWXPcKBI8   PT27M8S         hd    true  141598201    4505961
3    ZVt9ZJfWV1c     PT27S         hd   false  106220186    5282025
4    rWBOITBjitE     PT50S         hd   false   68296629    4738966
..           …         …        …       …          …          …
769  7qj3nuF9Dzw     PT31S         hd   false     873637      34407
770  Y74b7W1cEpk    PT2M6S         hd   false    1009673      39803
771  Z8nEEdXTaX0   PT1M30S         hd   false    1194378      46939
772  jP82d277Cc8   PT3M59S         hd   false    4288541         NA
773  2XVcLrB7B3Y   PT2M37S         hd   false   21142524     990140

    Favourite Count Comment Count
0                NA         66066
1                NA         87488
2                NA         78901
3                NA         13580
4                NA         13185
..              …             …
769              NA          3208
770              NA          3494
771              NA          4144
772              NA          8287
773              NA        115854

[774 rows x 13 columns]
```

[8]:
```
# Initialize a list to store Comments Data
comments_data = []

# Assuming you have the total number of videos in the variable total_videos
```

```python
for video_id in tqdm(videoIDs, desc="Processing Videos", unit="video"):
    try:
        request = youtube.commentThreads().list(
            part="snippet,replies",
            videoId=video_id,
            maxResults=10  # Limit the number of comments per video to 10
        ).execute()

        comments_in_video =␣
 ↪[comment['snippet']['topLevelComment']['snippet']['textOriginal'] for␣
 ↪comment in request.get('items', [])]
        comments_in_video_info = {'Video_id': video_id, 'Comments':␣
 ↪comments_in_video}

        comments_data.append(comments_in_video_info)

        # For demonstration purposes, simulate some processing time
        time.sleep(0.1)

    except Exception as e:
        if isinstance(e, HttpError) and e.resp.status in (403, 404):
            print(f"Video {video_id} not found. Skipping...")
        else:
            # Print other exceptions
            print(f"Could not get comments for video {video_id}. Error:␣
 ↪{str(e)}")

print(pd.DataFrame(comments_data))
```

```
Processing Videos:  40%|               | 308/774 [01:32<02:08,  3.63video/s]

Video AS5CxLCWq-Q not found. Skipping…

Processing Videos: 100%|          | 774/774 [03:38<00:00,  3.55video/s]

        Video_id                                           Comments
0    K_CbgLpvH9E  [watch until the end for good luck, My favorit…
1    lOKASgtr6kU  [Every family who adopted a dog was fully vett…
2    9RhWXPcKBI8  [Subscribe for a chance to be in one of my fut…
3    ZVt9ZJfWV1c  [         ,                        …
4    rWBOITBjitE  [Deque pais eres, Help me bro, Hindi   ,  , Ta…
..           …                                              …
768  7qj3nuF9Dzw  [Don't ask, Say, Cool music, Love u mr \nLove …
769  Y74b7WlcEpk  [I just got home from baseball practice lol, M…
770  Z8nEEdXTaX0  [I bought a new mic but it turned out to be wo…
771  jP82d277Cc8  [I remember filming this with my horrible lapt…
772  2XVcLrB7B3Y  [I was 13 when this was filmed lol. Go watch m…

[773 rows x 2 columns]
```

1.15 2.5 Export Request Data to CSV File

To optimize the data retrieval process and prevent alterations to existing requests, the decision has been made to export the data to a CSV file. This approach minimizes the need for repetitive API requests, ensuring resource efficiency and avoiding unnecessary strain on the YouTube API. Storing the data locally in a CSV format not only enhances the speed of data analysis, particularly with substantial datasets or frequent analyses, but also ensures consistency and reproducibility of results. By utilizing a static data file, the dataset remains unchanged across multiple analyses, contributing to result reliability.

Moreover, this approach aligns with compliance considerations, preventing the risk of surpassing the YouTube API's daily quota limits, especially during debugging or code rewriting scenarios. In summary, exporting data to a CSV file strikes a balance between efficiency, stability, and compliance, fostering a smoother and more dependable analytical process.

Additionally, exporting to a CSV file serves as a precautionary measure against potential errors caused by alterations in the API response or connectivity issues during analysis, providing a stable data source.

[9]:
```
#Convert JSON list to panda DataFrame
video = pd.DataFrame(video_data)
comments = pd.DataFrame(comments_data)

# Specify the CSV file path
video_csv_file_path = 'video_data.csv'
comments_csv_file_path = 'comments_data.csv'

# Write the DataFrame to a CSV file
video.to_csv(video_csv_file_path, index=False)
comments.to_csv(comments_csv_file_path, index=False)
```

1.16 2.6 Retrieve Data from CSV File

[10]:
```
#Retrive Data in panda format
vd = pd.read_csv('video_data.csv')
cd = pd.read_csv('comments_data.csv')
```

1.17 2.7 Data Pre-Processing

The meticulous examination of null values and their types is a pivotal step in ensuring data quality and preventing errors. This process offers a clear understanding of each variable's storage, laying a foundation for seamless data exploration. It supports effective data cleaning by allowing for the imputation of missing values or the removal of instances with missing data from the analysis. ertain columns undergo necessary conversions to accommodate specific libraries and enable optimal analysis. This comprehensive pre-processing ensures a reliable and accurate basis for the subsequent stages of the analytical pipeline.

[11]:
```
#Check for NULL values
vd.isnull().any()
```

[11]:
```
Title               False
Channel Title       False
Description          True
Tags                 True
Published At        False
Video ID            False
Duration            False
Definition          False
Caption             False
View Count          False
Like Count           True
Favourite Count      True
Comment Count        True
dtype: bool
```

[12]:
```
#Check data types
vd.dtypes
```

[12]:
```
Title                object
Channel Title        object
Description          object
Tags                 object
Published At         object
Video ID             object
Duration             object
Definition           object
Caption                bool
View Count            int64
Like Count          float64
Favourite Count     float64
Comment Count       float64
dtype: object
```

[13]:
```
#Check for NULL values
cd.isnull().any()
```

[13]:
```
Video_id    False
Comments    False
dtype: bool
```

[14]:
```
#Check for types
cd.dtypes
```

```
[14]: Video_id    object
      Comments    object
      dtype: object
```

1.18 2.8 Data Type Conversion

The conversion of 'Count' columns to numerical values, 'Published At' column to datetime format and 'Duration' column to seconds is undertaken to facilitate seamless data analysis in the later stages of the project.

```
[15]: #Convert count columns to Numeric
      numeric_cols = ['View Count', 'Like Count', 'Favourite Count', 'Comment Count']
      vd[numeric_cols] = vd[numeric_cols].apply(pd.to_numeric, errors = 'coerce',␣
        ↪axis = 1)
```

```
[16]: #Convert Published At to DataTime Type
      vd['Published At'] = pd.to_datetime(vd['Published At'])
      vd['Publish Day Name'] = vd['Published At'].dt.strftime("%A")

      vd[['Publish Day Name','Published At']]
```

```
[16]:     Publish Day Name              Published At
      0           Saturday 2023-12-30 17:00:03+00:00
      1           Saturday 2023-12-23 17:00:00+00:00
      2           Saturday 2023-12-16 17:00:01+00:00
      3           Thursday 2023-12-14 18:00:00+00:00
      4            Tuesday 2023-12-05 18:00:00+00:00
      ..               …                        …
      769           Sunday 2013-01-13 01:59:21+00:00
      770         Saturday 2013-01-12 23:35:45+00:00
      771         Saturday 2013-01-12 22:34:11+00:00
      772           Friday 2012-03-09 23:29:03+00:00
      773           Monday 2012-02-20 22:42:32+00:00

      [774 rows x 2 columns]
```

```
[17]: #Convert duration to seconds
      vd['DurationSecs'] = vd['Duration'].apply(lambda x: isodate.parse_duration(x))
      vd['DurationSecs'] = vd['Duration'].astype('timedelta64[s]')

      vd[['DurationSecs','Duration']]
```

```
[17]:        DurationSecs  Duration
      0   0 days 00:20:16  PT20M16S
      1   0 days 00:15:03   PT15M3S
      2   0 days 00:27:08   PT27M8S
      3   0 days 00:00:27     PT27S
      4   0 days 00:00:50     PT50S
```

```
..                 …       …
769 0 days 00:00:31     PT31S
770 0 days 00:02:06    PT2M6S
771 0 days 00:01:30   PT1M30S
772 0 days 00:03:59   PT3M59S
773 0 days 00:02:37   PT2M37S

[774 rows x 2 columns]
```

1.19 2.9 Lemmatization & StopWords

WordNet lemmatizer to convert words in their base or dictionary form. This helps in standardizing word variations.

The preprocess_text function performs several preprocessing tasks on the text data:

- Lowercasing
- Removal of Emojis
- Removal of Non-Alphanumeric Characters
- Punctuation Removal
- Tokenization
- Stopword Removal
- Custom Stopwords

[18]:
```python
def lemmatize_tokens(tokens):
    # Initialize the WordNet lemmatizer
    lemmatizer = WordNetLemmatizer()

    # Lemmatize each token
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]

    return lemmatized_tokens
```

[19]:
```python
def preprocess_text(text):
    text = text.lower()

    #Remove emojis
    text = emoji.demojize(text)

    #Remove non-alphanumeric characters
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)

    #Remove certain punctuation using regex.
    text = re.sub('\.|\,|\"|\"|\"|\-|\-|\-|\_|\?|\!|\:|\;|\/|\(|\)', '', text)

    #Tokenize the text
    tokens = word_tokenize(text)

    #Remove stopwords using NLTK
    stop_words = set(nltk.corpus.stopwords.words('english'))

    #Remove tokens with only punctuation or suffixes separated from words
    punctuation = ["''", "'''", '``', '(', ')', '%', '&', '...', '…', "‘", "’"]
    suffixes = ["'s", "n't", "'ve", "'ll", "'re", "'d"]
    custom_stopwords = ["youtuber", "youtube", "youtubers", "much", "please",␣
 ↪"mr", "subscribe",
                       "redheart", "video", "videos", "comment", "im"]  # Add␣
 ↪any custom stopwords
    remove_words = punctuation + suffixes + custom_stopwords + list(stop_words)

    #Create a new list with stopwords removed from the text
    filtered_tokens = [word for word in tokens if word not in remove_words]

    #Reconstruct text as a string
    filtered_text = ' '.join(filtered_tokens)

    return filtered_text

def clean_text(text):
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)

    return text
```

[20]:
```
#Apply the preprocess_text function to the 'Title' column
vd['title_no_stopwords'] = vd['Title'].apply(preprocess_text)

#Print the result
print(vd[['Title', 'title_no_stopwords']])
```

```
                                                 Title  \
0                I Spent 7 Days In Solitary Confinement
1                         I Rescued 100 Abandoned Dogs!
2                Survive 100 Days Trapped, Win $500,000
3                     Feeding A Dog $1 vs $10,000 Steak
4                           Could You Walk Up A Skyscraper?
..                                                  …
769                 Most Epic minecraft skin EVER  (Psy)
770                            More birds IN MINECRAFT!!
771                      Boxy item mod Minecraft.  EPIC
772  Harry Potter Mod In Minecraft! EPIC MUST SEE M…
773                    Worst Minecraft Saw Trap Ever???

                                  title_no_stopwords
0                spent 7 days solitary confinement
1                         rescued 100 abandoned dogs
2              survive 100 days trapped win 500000
```

```
3                    feeding dog 1 vs 10000 steak
4                           could walk skyscraper
..                                            …
769                  epic minecraft skin ever psy
770                               birds minecraft
771                  boxy item mod minecraft epic
772  harry potter mod minecraft epic must see mod
773                 worst minecraft saw trap ever

[774 rows x 2 columns]
```

[21]:
```
# Convert string representation of lists to actual lists
cd['Comments'] = cd['Comments'].apply(lambda x: ast.literal_eval(x) if␣
 ↪isinstance(x, str) else x)

# Extract individual comments
individual_comment = [comment for comments_list in cd['Comments'] for comment␣
 ↪in comments_list]

individual_c = pd.DataFrame(individual_comment, columns=['Individual Comments'])

individual_c
```

[21]:
```
                                    Individual Comments
0                        watch until the end for good luck
1                                         My favorite vid
2                                                    Help
3                                               Mr. Beast
4     Ayuda , porque siento que yo si aguantaría más…
…                                                     …
7721                                 Aquí nació una leyenda
7722                                   LEGEND SEE IN 2024
7723        Then first now it is insane how far he came
7724
7725                          2024'ten izliyen turkler

[7726 rows x 1 columns]
```

[22]:
```
#Apply the preprocess_text function to the 'Title' column
individual_c['Comments_no_stopwords'] = individual_c['Individual Comments'].
 ↪apply(preprocess_text)

#Print the result
print(individual_c[['Comments_no_stopwords', 'Individual Comments']])
```

```
                                 Comments_no_stopwords  \
0                                  watch end good luck
1                                         favorite vid
```

```
2                                                      help
3                                                     beast
4     ayuda porque siento que yo si aguantara ms de …
…                                                         …
7721                                   aqu naci una leyenda
7722  legend see 2024facewithtearsofjoysmilingfacewi…
7723                                  first insane far came
7724                                                   goat
7725          2024ten izliyen turklerTurkeyTurkeyTurkey

                                    Individual Comments
0                          watch until the end for good luck
1                                          My favorite vid
2                                                      Help
3                                                 Mr. Beast
4     Ayuda , porque siento que yo si aguantaría más…
…                                                         …
7721                                 Aquí nació una leyenda
7722                                   LEGEND SEE IN 2024
7723        Then first now it is insane how far he came
7724
7725                            2024'ten izliyen turkler

[7726 rows x 2 columns]
```

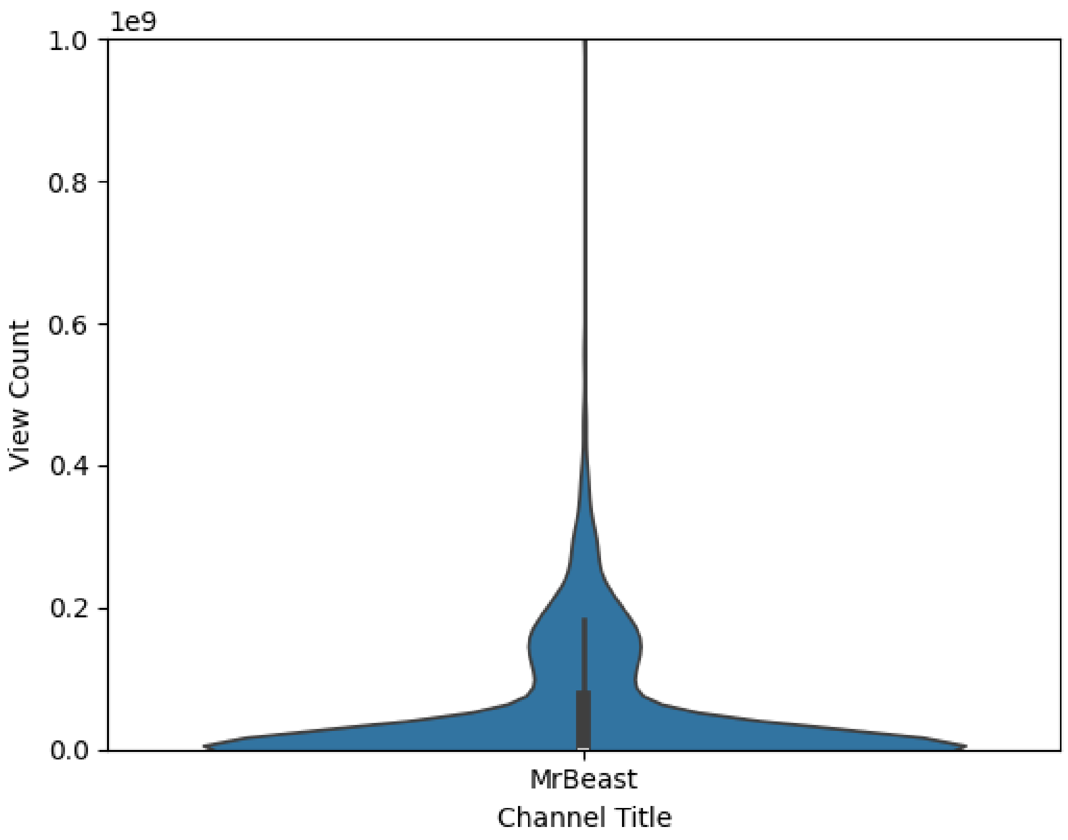
1.20 3. Data Analysis & Visualization

In the Data Analysis & Visualization section, we utilize a variety of powerful visualization techniques to gain insights from the dataset: * Violinplot * Barplot * Lineplot * Scatterplot * WordCloud * Venn Diagram.

1.21 3.1 View Count Distribution

[23]:
```
#View Count Distribution in Violin Plot
sns.violinplot(x=vd['Channel Title'], y=vd['View Count'])
plt.ylim(0, 1000000000)
```

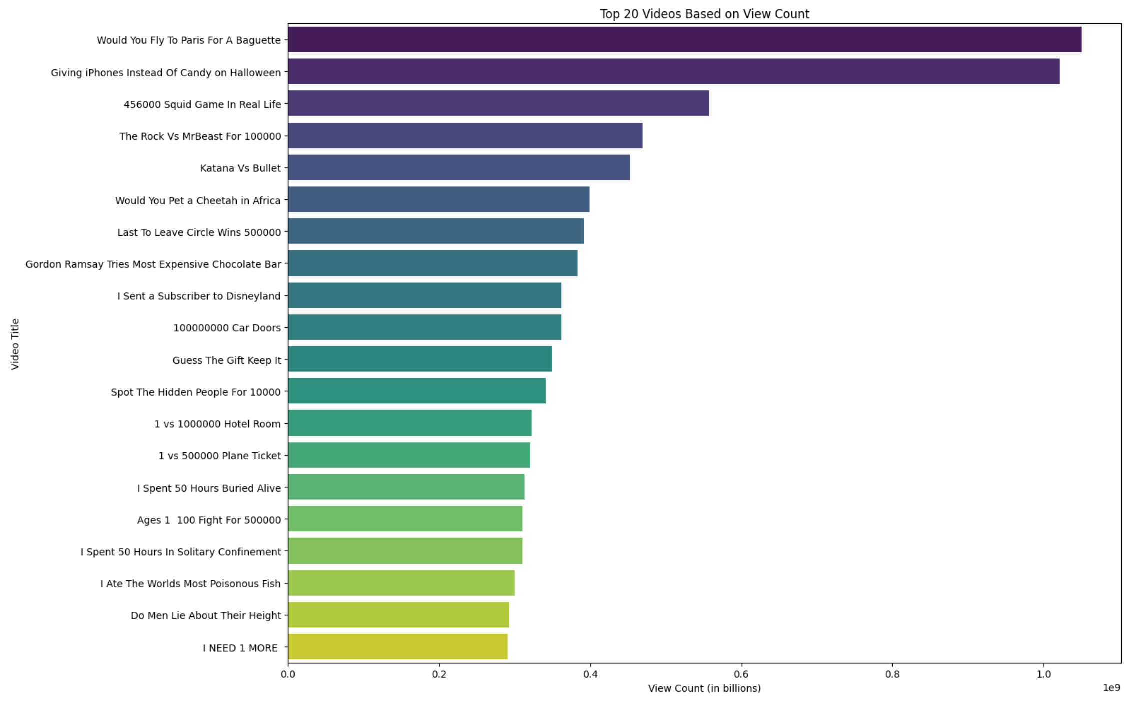
[23]:
```
(0.0, 1000000000.0)
```



The violin plot illustrates the distribution of Mr Beast's video view counts. Notably, a significant portion of his videos garners fewer than 40 million views, with over 50% falling below the 10 million mark. Additionally, there are noteworthy outliers, showcasing videos with view counts ranging from above 40 million to an impressive 1 billion. This analysis provides insights into the varying levels of popularity within Mr Beast's video catalog, highlighting both the majority distribution and exceptional performance of select videos.

1.22 3.2 Correlation of Titles and View Count

```
[24]: #View Count in Descending Order in Barplot
vd_sorted = vd.sort_values(by='View Count', ascending=False)
top_20_videos = vd_sorted.head(20).copy()

top_20_videos['Title'] = top_20_videos['Title'].apply(clean_text)

plt.figure(figsize=(16, 10))
ax = sns.barplot(x='View Count', y='Title',hue ='Title', data=top_20_videos,␣
 ↪palette='viridis')
ax.set_xlabel('View Count (in billions)')
ax.set_ylabel('Video Title')
ax.set_title('Top 20 Videos Based on View Count')

plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

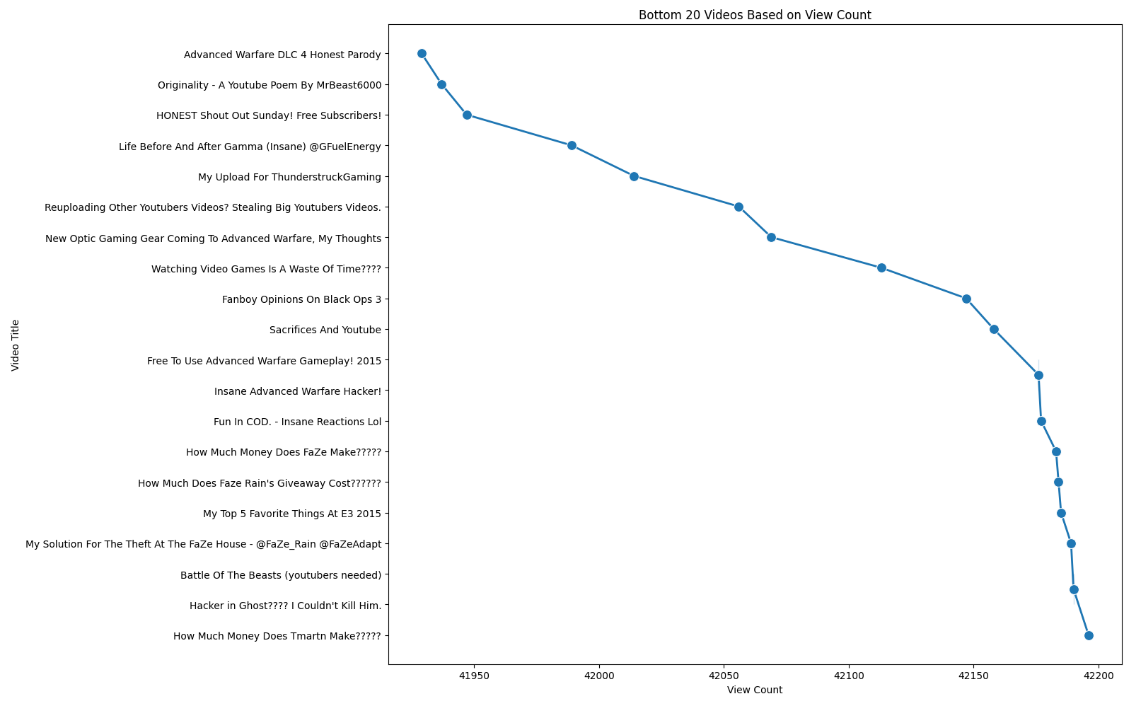


This analysis above focuses on the top 20 videos from Mr Beast's YouTube channel, shedding light on the content that has garnered the highest view counts. The majority of Mr Beast's top-performing videos fall into the challenge category, except for one where he onboarded a celebrity chef in his video.

This observation underscores the significant interest in reality shows, hinting at a promising opportunity to delve deeper into this genre.

[25]:
```
#View Count in Ascending Order in Lineplot
vd_sorted = vd.sort_values(by='View Count', ascending=True)
top_20_videos = vd_sorted.head(20)

plt.figure(figsize=(16, 10))
ax = sns.lineplot(x='View Count', y='Title', data=top_20_videos, marker='o',␣
  ↪markersize=10, linewidth=2)
ax.set_xlabel('View Count')
ax.set_ylabel('Video Title')
ax.set_title('Bottom 20 Videos Based on View Count')
```

```
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



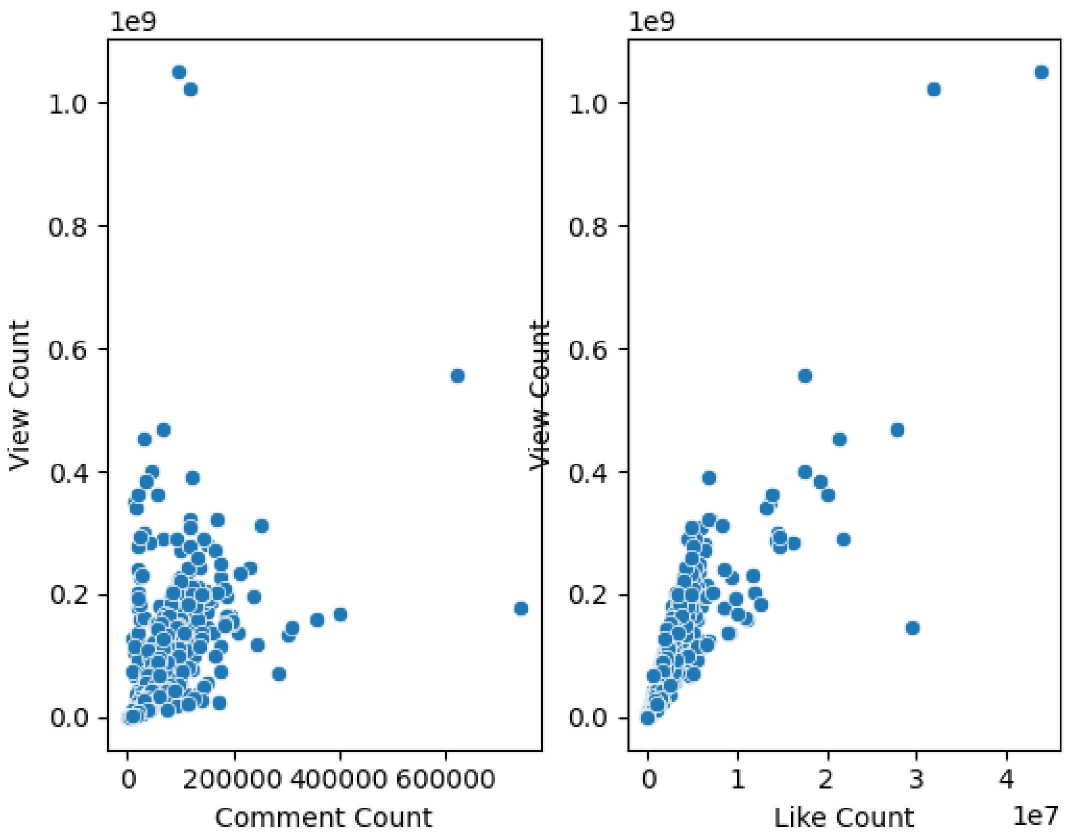
The analysis above delves into Mr Beast's videos with the lowest view counts (hovering at 42,000 view), aiming to understand the content themes. Notably, a majority of these videos fall under the gaming category. It's worth noting that the lower view counts may not necessarily indicate a lack of audience or appeal for the gaming genre. Instead, it could be attributed to the specific games featured, which might not be as widely popular.

Additionally, there are instances where Mr Beast explores 'How-to' type videos. While this content category may not resonate with his audience, there might be other external factors contributing to this view count. This insight underscores the importance of carefully selecting 'How-to' topics to ensure broader audience engagement and interest."

1.23 3.3 Correlation of Comment Count and Like Count with View Count

[26]:
```
#Correlation of Comment Count & Like Count with View Count in Scatterplot
fig, ax = plt.subplots(1,  2)
sns.scatterplot(data = vd, x = 'Comment Count', y = 'View Count', ax = ax[0])
sns.scatterplot(data = vd, x = 'Like Count', y = 'View Count', ax = ax[1])
```

[26]:
```
<Axes: xlabel='Like Count', ylabel='View Count'>
```



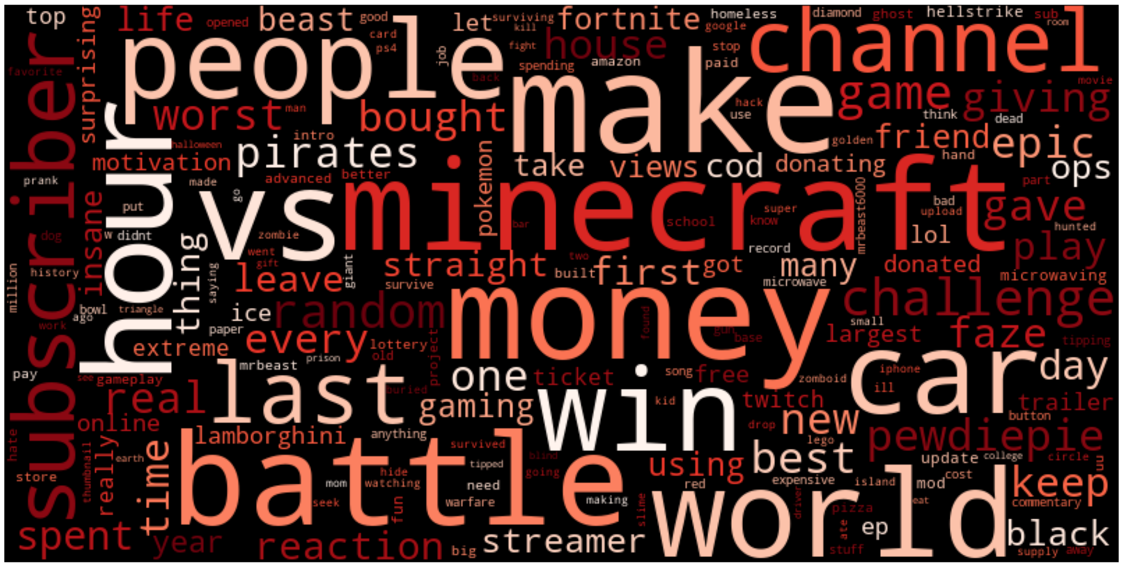
For these two graphs, the goal is to explore the relationship between view count and both comment and like counts. The scatter plots display a consistent pattern where the points fan out from the bottom left to the top right. This suggests a positive correlation between view count and both comments and likes.

While the majority of points follow this trend, there are two notable outliers with high view counts but relatively low comment counts. Despite these exceptions, the overall distribution implies that videos with higher view counts tend to attract more engagement, as indicated by comments and likes.

1.24 3.4 Correlation of Comments with Video Title

[27]:
```
#Most common words used in Title
all_words = [word for title in vd['title_no_stopwords'] for word in title.
 ↪split()]
all_words_str = ' '.join(all_words)

def plot_cloud(wordcloud):
    plt.figure(figsize=(30, 20))
    plt.imshow(wordcloud)
    plt.axis("off")
```

```
wordcloud = WordCloud(width=800, height=400, background_color='black',
                      colormap='Reds', random_state=1, max_words=200,
                      max_font_size=100, contour_width=3, contour_color='white',
                      collocations=False).generate(all_words_str)

plot_cloud(wordcloud)
```



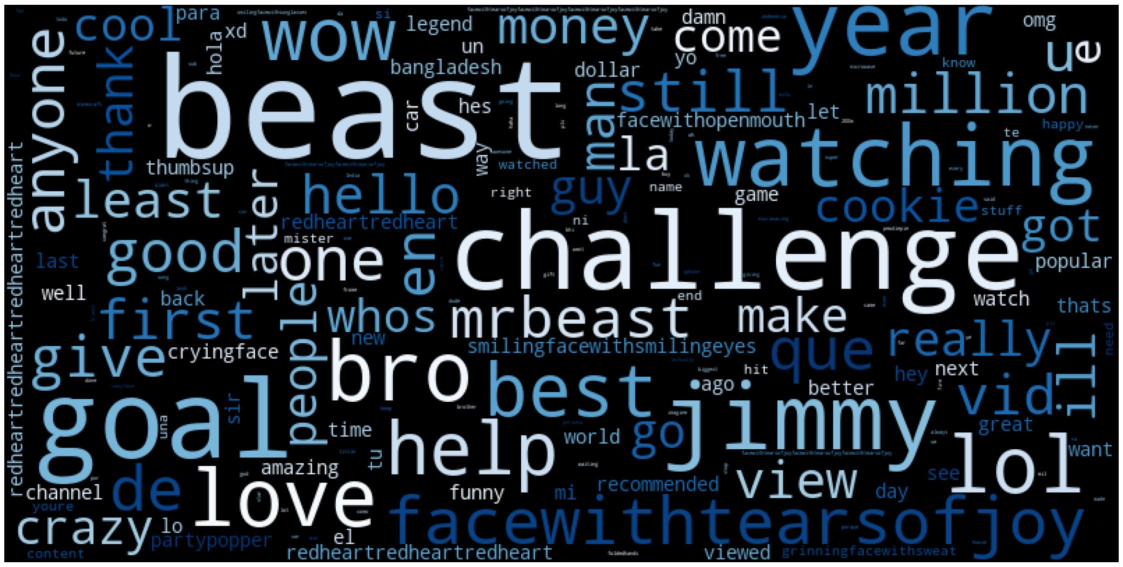
The presented word cloud reveals the most frequently used words in Mr Beast's video titles. Notably, terms like "challenge," "battle," "money," "win," "hour," and "vs" prominently stand out, aligning with our earlier analysis highlighting Mr Beast's inclination towards challenge-based content. This emphasizes his significant presence in the challenges genre.

Interestingly, another set of keywords, including "gaming," "minecraft," "gameplay," and "ops," surfaces, suggesting a distinct presence of gaming-related content. Unlike our previous analysis, popular game names are now evident, indicating a potential avenue within the gaming genre.

In essence, this analysis provides valuable insights into the diverse content themes Mr Beast explores, particularly emphasizing the challenges and gaming genres as prominent categories.

```
[28]: #Most common words used in Comments
      all_words = [word for title in individual_c['Comments_no_stopwords'] for word␣
       ↪in title.split()]
      all_words_str = ' '.join(all_words)

      def plot_cloud(wordcloud):
          plt.figure(figsize=(30, 20))
          plt.imshow(wordcloud)
          plt.axis("off")
```

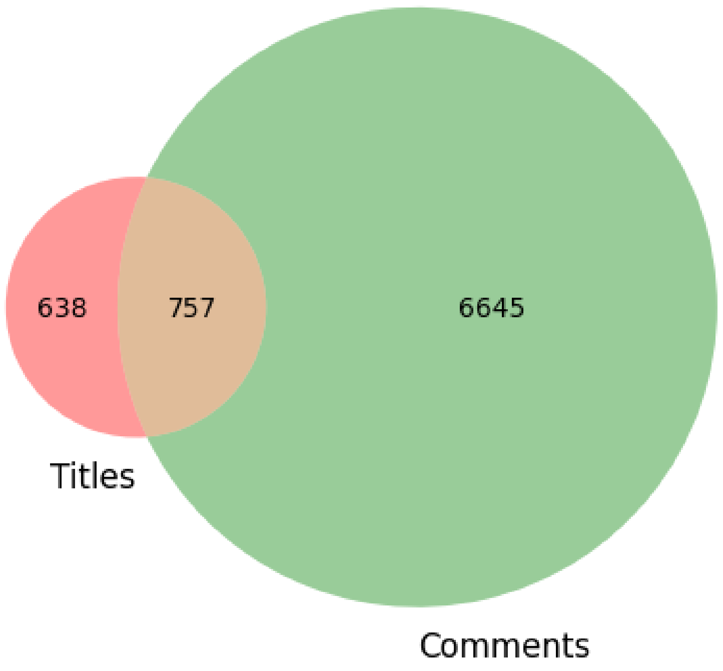
```
wordcloud = WordCloud(width=800, height=400, background_color='black',
                      colormap='Blues', random_state=1, max_words=200,
                      max_font_size=100, contour_width=3, contour_color='white',
                      collocations=False).generate(all_words_str)

plot_cloud(wordcloud)
```



The depicted word cloud showcases the top 10 comments from each video, aiming to discern any correlation with the video titles. However, it becomes apparent that there is little to no direct relevance between the words in the comments and the video titles. Instead, the comments predominantly consist of words of support and encouragement from Mr Beast's audience.

While a direct correlation might not be evident, the prevalence of positive and encouraging comments underscores the importance of audience engagement. Such words of encouragement contribute to fostering a supportive and positive online community. For a content creator like Mr Beast, these comments serve as a motivating factor, reinforcing the idea that creating meaningful content is appreciated by the audience.

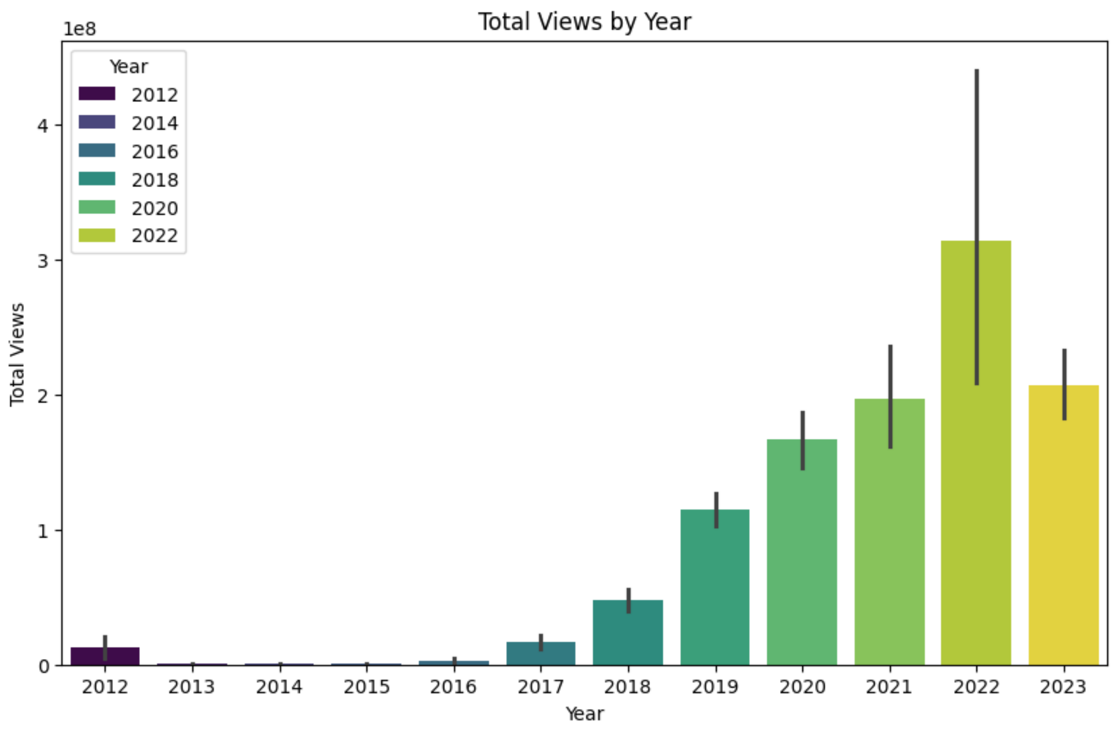
[29]:
```
#Word Count that coincides between Title and Comments using Venn Diagram
# Convert titles and comments to sets of words
title_set = set(vd['title_no_stopwords'].str.split().sum())
comment_set = set(individual_c['Comments_no_stopwords'].str.split().sum())

# Create a Venn diagram
venn2([title_set, comment_set], set_labels=('Titles', 'Comments'))

plt.show()
```



Although there are 752 words that are the same in the venn diagram, we can conclude from both word cloud that the comments from the videos do not have much correlation to the title of the video.

1.25 3.5 Channel Growth Analysis

[30]:
```
#Total Views over the years
vd['Year'] = vd['Published At'].dt.year
# Assuming you have a DataFrame called 'df' with 'Year' and 'Total Views'␣
 ↪columns
# Replace 'df' with the actual name of your DataFrame
plt.figure(figsize=(10, 6))
sns.barplot(x='Year', y='View Count', hue='Year', data=vd, palette='viridis')
plt.xlabel('Year')
plt.ylabel('Total Views')
plt.title('Total Views by Year')
plt.show()
```



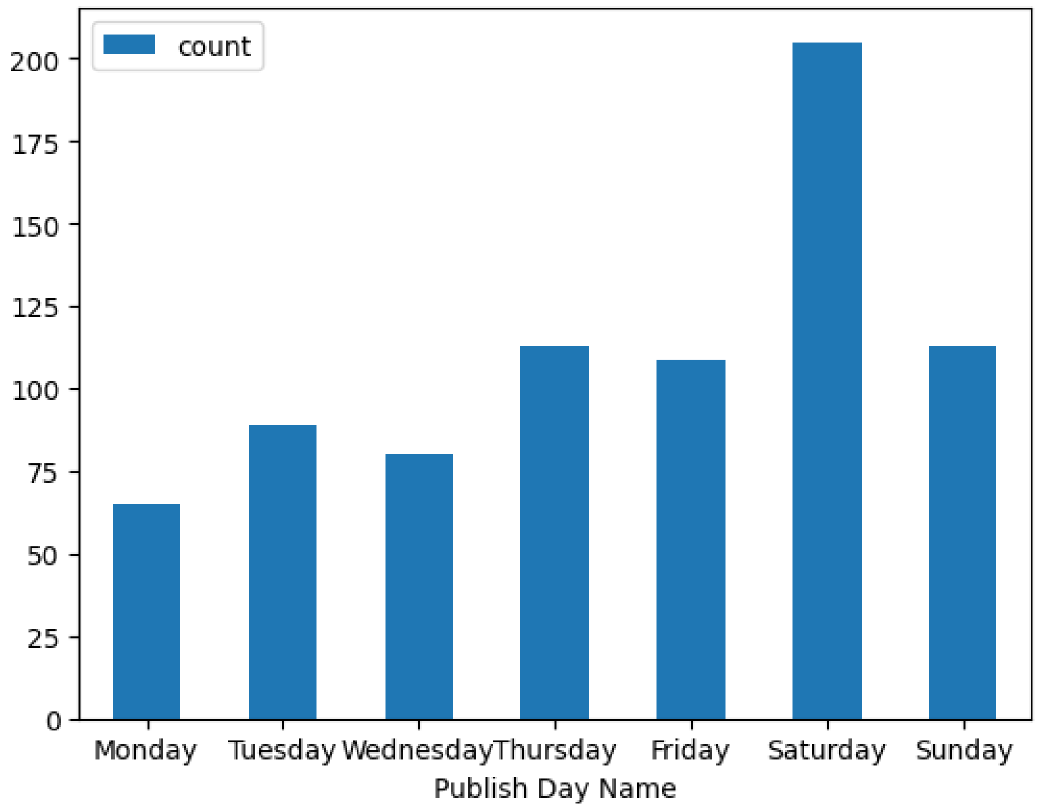
The above graph illustrates the channel's growth trajectory over the years based on total view count. Upon excluding the outlier of 2012, a consistent upward trend is observed from 2016 to 2022, with a progressive increase in view count. Notably, there is a substantial spike in view count in 2022, followed by a decline in 2023.

The black line running through the middle of the graph represents the confidence interval (CI). For most years, the CI is relatively short, indicating a higher level of confidence in the estimated mean view count. However, in 2023, the CI is noticeably longer, suggesting increased uncertainty in the estimate. This is likely attributed to outliers—videos with extremely high view counts—resulting in a larger mean value.

Despite the presence of outliers in 2023, examining the bottom of the confidence interval reveals a value similar to the heights of 2021 and 2022. This suggests that, although extreme view counts may influence the mean, the lower bound of the CI remains consistent, leading to the conclusion that the growth is slowing down during these three years.

1.26 3.6 Correlation of Published Day Frequency with View Count

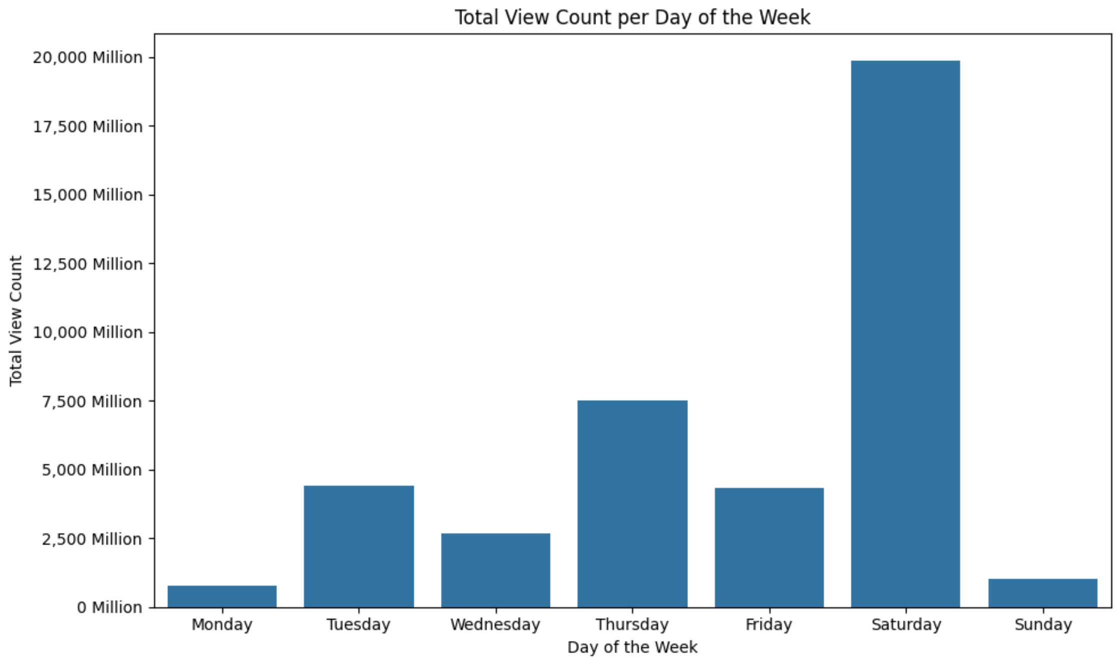
[31]:
```
#Published Frequency throughout the week
day_df = pd.DataFrame(vd['Publish Day Name'].value_counts())
weekdays = [ 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
 ↪'Saturday', 'Sunday']
day_df = day_df.reindex(weekdays)
ax = day_df.reset_index().plot.bar(x='Publish Day Name', y='count', rot=0)
```



The bar graph illustrates the frequency of video publications on different days. On average, Mr Beast tends to post videos on Thursday, Friday, and Sunday. Notably, Saturday emerges as the day with the highest frequency of video releases, while Monday sees the lowest frequency. This distribution provides insights into Mr Beast's content release schedule, possibly aligning with strategic considerations or audience engagement patterns on specific days.

[32]:
```
#View Count per Day
# Group by 'Publish Day Name' and sum the 'View Count'
view_count_per_day = vd.groupby('Publish Day Name')['View Count'].sum()

# Reorder the days of the week
view_count_per_day = view_count_per_day.reindex(weekdays)

# Plot the view count per day
plt.figure(figsize=(10, 6))
ax = sns.barplot(x=view_count_per_day.index, y=view_count_per_day.values,␣
 ↪order=weekdays)  # Removed 'palette' argument
ax.set_xticks(ax.get_xticks())
ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '{:,.0f}␣
 ↪Million'.format(x / 1000000)))
```

```
plt.xlabel('Day of the Week')
plt.ylabel('Total View Count')
plt.title('Total View Count per Day of the Week')

plt.tight_layout()
plt.show()
```



This analysis reveals a noteworthy connection between the choice of publishing days and Mr Beast's video view counts. Particularly, Saturdays emerge as a standout day, consistently garnering the highest audience engagement. The subsequent high rankings on Thursdays and Fridays further affirm a pattern of viewer interest during these periods.

However, the anomaly observed on Sundays is well-explained by the presence of both exceptionally high and low view counts, underscoring the impact of outliers on the overall average. It's crucial to consider such extremes when interpreting the data.

The key takeaway from this observation is the significance of strategic publishing in maximizing audience interaction. Content creators can benefit from understanding the nuanced patterns of viewer engagement and optimizing their video release schedule accordingly.

1.27 3.7 Conclusions

Base on the above data visual analysis, we can conlude that Mr Beast being one of the top youtube channel globally has certain trends that he is adhering to: Diverse Content Themes, Strategic Publishing and Positive Audience Engagement.

The findings of this research highlighted the importance of several key aspects which contributed continuous growth for a Youtube Channel.These insights provide invaluable lessons for aspiring

YouTubers and content creators seeking to navigate the competitive world of online media.

1.28 4. References

YouTube Data API - Search: https://developers.google.com/youtube/v3/docs/search/list

YouTube Data API Documentation: https://developers.google.com/youtube/registering_an_application

Seaborn Documentation: https://seaborn.pydata.org/

Matplotlib Documentation: https://matplotlib.org/

Plotly Bar Charts in Python: https://plotly.com/python/bar-charts/

TQDM Documentation: https://tqdm.github.io/

Kaggle Dataset - MrBeast YouTube Video Statistics: https://www.kaggle.com/datasets/robikscube/mr-beast-youtube-video-statistics?resource=download

Appleby, C. (2023, December 14). MrBeast's analytics platform ViewStats is out in beta. TechCrunch. https://techcrunch.com/2023/12/14/mrbeasts-analytics-platform-viewstats-is-out-in-beta/

Villasista, P. J. (2023, October 4). Leveraging data analytics on MrBeast's YouTube videos: A deep dive into engagement, sentiment, and more. Medium. https://medium.com/@pjvillasista/leveraging-data-analytics-on-mrbeasts-youtube-videos-a-deep-dive-into-engagement-sentiment-and-3fa21d4fcbea

Tubefilter Staff. (2023, December 13). View Stats: The new analytics platform co-founded by MrBeast. Tubefilter. https://www.tubefilter.com/2023/12/13/view-stats-analytics-statistics-mrbeast-jimmy-donaldson-chucky-appleby/