COMP5721M: Programming for Data Science

Coursework 3: Data Analysis Project

Last modified: 15 November 2023

Analysis Of Youtube Trending Data

Give names and emails of group members here:

- Asish Panda, mm23ap@leeds.ac.uk
- Mohamed Imthiyas Abdul Rasheeth, mm23m2ia@leeds.ac.uk
- Naveen Sabarinath Babu, mm23nsb@leeds.ac.uk
- Roshan ., mm23rt@leeds.ac.uk

Project Requirements

PLEASE DELETE THIS WHOLE CELL BEFORE SUBMITTING YOUR PROJECT

The purpose of this assignment is to develop your skills in organising and presenting a Data Science project.

Since most of the marks will be awarded for organisation and presentation, it is suggested that you do not initially attempt anything too complicated. However, once you have managed to get a basic pipeline working that fits the guidelines, you are encouraged to extend and elaborate your analysis.

Your project should entirely be contained within this template file. You should keep the basic structure indicated below. To facilitate grading according to the marking scheme.

You *may* import any module that is provided with Anaconda3 Python.

Marking Scheme

The marking scheme is as follows:

- Project Plan:
  - Description of data to be used (10)
  - Overview of Project Aims (5)
  - Design (5)
- Program Code: (15)
  Code should be laid out in steps with explanations and intermediate output with comments. You should ensure that the steps do not require a large amount of processing time.

- Project Outcome:
    - Explanation of Results (10)
      This should include a qualitative description of the results as well as key figures and tables of results.
    - Results visualisation (10)
      This should be graphical representations of the results with brief explanations (ordinary tables will be graded as part of the explanation of results)
    - Conclusion (5)

Data Resources

You can use any data you like. Many useful resources are available.

The Data Resources section of the module (Unit 4.3 on Minerva) has links to several example data sets.

As a starting point you could browse the following:

- Kaggle
- Our World in Data
- scikit-learn datasets
- scikit-learn tutorial

Using this Notebook Template

Please use this notebook as a template for your project file. In the following cells of the notebook, *italic text* giving explanations and examples should be either deleted, or, in most cases, replaced by appropriate text describing your project. Text that is not in italic (which is mostly headings) should be left as it is. **Your project report notebook should the same overall structure as this template notebook.** An exception to this is the current markup cell describing the project requiements. You should delete this before submitting your notebook.

Project Plan

The Data (10 marks)

Youtube provides an API that provides information about trendng data country-wise. The dataset on Kaggle is a real-time (daily) updating dataset derived from the API consisting of attruibutes for various contries which is used for analysis in this project. The scope of this project limits our use to a specific daterange and limited the country to Great Britain.

The dataset consists of two files , _GB_category_id.json and GB_youtube_trending*data.csv*. The files contain the follwoing column ID's

CSV:

['video_id', 'title', 'publishedAt', 'channelId', 'channelTitle', 'categoryId', 'trending_date', 'tags', 'view_count', 'likes', 'dislikes', 'comment_count', 'thumbnail_link', 'comments_disabled', 'ratings_disabled', 'description']

JSON:

The json file contains a data structure that links the categoryId column in each file. The json file alos ha sinformation about each files category i.e ['family','Entertainment','Education']. The structure of the file is as follows

```
{
    "kind": "youtube#videoCategoryListResponse",
    "etag": "kBCr3I9kLHHU79W4Ip5196LDptI",
    "items": [
        {
            "kind": "youtube#videoCategory",
            "etag": "IfWa37JGcqZs-jZeAyFGkbeh6bc",
            "id": "1",
            "snippet": {
                "title": "{{category_string}}",
                "assignable":{{boolean}},
                "channelId": "{{string}}"
            }
        },
        {
            "kind": "youtube#videoCategory",
            "etag": "5XGylIs7zkjHh5940dsT5862m1Y",
            "id": "2",
            "snippet": {
                "title": "{{category_string}}",
                "assignable": {{boolean}},
                "channelId": "{{string}}"
            }
        },
    ]
}
```

Project Aim and Objectives (5 marks)

The primary objective of this project is to perform an in-depth analysis of the "Trending Youtube API Dataset" to identify and understand the important attributes that results in the uploaded video to be "trending" in the platform. The project aims to find the patterns in the dataset that would push the videos, uploaded by the users, into the trending category; thus helping the content creators to optimize their video uploads accordingly that would contribute to the Youtube Algorithm.

The aim of the project is achieved by making use of data analysis techniques and machine learning algorithms to find out the correlations between the different attributes in the dataset. We will make use of columns such as {categoryId, title,view_count,likes,dislikes etc.} to recognize the patterns resulting in a trending video. After thorough analysis of the various aspects, we aim to discern the commonalities

among the trending videos, while also identifying the factors that may vary across the different genres of Youtube.

In summary, the project aims to provide insight into the dynamics of Youtube Trending videos, thereby helping content creators in sharing content that would satisfy the Youtube algorithm and results in the video being in the Youtube Trending. The results will help in improving the experience of the users and Youtubers on the Youtube Platform.

_Here you should describe the general aim of your project in

around 200-300 words._

_This can can be anything from classifying items according to

their characteristic features (which mushrooms are poisonous?) to simulating an evolving process (will the rabbits eat all the carrots or get eaten by the foxes?)_

_Here some ideas of general types of processing functionality

that you could implement:_

- *Classification: separate data items into classes according to their charactersitics (can be either a definite or a statistical kind of classification)*
- *Corellation: find correspondences between different attributes within a dataset*
- *Search: find solutions matching some criteria*
- *Visualisation: find informative ways to display the structure of a large and/or complex dataset*
- *Query Answering: create a system that enables one to retrieve information by evaluating some form of query representation*
- *Simulation: model the evolution of a complex process*

Specific Objective(s)

*You should chose and list **up to 4** specific objectives suited to the data you will be working with and the type of project you wish to carry out. There should be **at least one per person doing the project**. There is no need for the objectives them to be completely different. They could be different stages of the processing requirements, or different processing functions that the system provides. Or just different aspects of data analysis that will be conducted. Typically, it is expected that there would be one objective per person. Replace the following examples with your own objectives:*

- **Objective 1:** *Visualize the genres that trend in the United Kingdom*
- **Objective 2:** *Checks the key words to be used in the Title and/or Description for a video to be in the trending category*
- **Objective 3:** *Calculates whether the Likes, Dislikes and View Count influences a video to trend*
- **Objective 4:** *Check whether the upload date and time results in a video to trend*

System Design (5 marks)

Describe your code in terms of the following two sections.

Architecture

Typically this would be a pipeline in which data goes through several stages of transformation and analysis, but other architectures are possible. This does not need to be particularly complicated. A simple diagram with 100-150 words of explanation would be a good way to present your architecture.

Processing Modules and Algorithms

Briefly list and describe the most significant computational components of your system and the algorithms you will use to implement them. This could include things like:

- *cleaning the data by removing outliers*
- *combining different datasets*
- *converting samples to a special representaion (such as feature vectors)*
- *constructing a special data-structure (such as a decision tree)*
- *running some kind of analysis*

Your list can be presented in similar form to the one just given, but should include a brief but more specific description of the components and/or algorithms. Probably three or four components is sufficient for most projects, but you may want to have more.

Program Code (15 marks)

Module Imports

We are importing `pandas` since it is used for DataFrame operations. Along with Pandas, we are also using the built-in `JSON` to import our `GB_category_id.json`

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

Reading And Combining the two files

In the following code block, we read the CSV as `dfc` and transform it into a pandas data frame called `df`. We then read the JSON file into a variable called `categories`. We then select the category title from `categories` and append it into a new column called `category` in the `df` data frame.

We use the `head()` method to see how our data frame looks.

In [ ]:
```
dfc= pd.read_csv('GB_youtube_trending_data.csv')
df = pd.DataFrame(dfc)

# Load the categories
with open('GB_category_id.json') as f:
    categories = json.load(f)

# Create a dictionary to map category IDs to category names
category_dict = {int(item['id']): item['snippet']['title'] for item in categ

# Map the category IDs in the dataframe to the category names
df['category'] = df['categoryId'].map(category_dict)

df.head()
```

Out[ ]:

	video_id	title	publishedAt	channelId	channelTitle	cate
0	J78aPJ3VyNs	I left youtube for a month and THIS is what ha...	2020-08-11T16:34:06Z	UCYzPXprvl5Y-Sf0g4vX-m6g	jacksepticeye	
1	9nidKH8cM38	TAXI CAB SLAYER KILLS 'TO KNOW HOW IT FEELS'	2020-08-11T20:00:45Z	UCFMbX7frWZfuWdjAML0babA	Eleanor Neale	
2	M9Pmf9AB4Mo	Apex Legends | Stories from the Outlands – “Th...	2020-08-11T17:00:10Z	UC0ZV6M2THA81QT9hrVWJG3A	Apex Legends	
3	kgUV1MaD_M8	Nines - Clout (Official Video)	2020-08-10T18:30:28Z	UCvDkzrj8ZPlBqRd6flxdhTw	Nines	
4	49Z6Mv4_WCA	i don't know what im doing anymore	2020-08-11T20:24:34Z	UCtinbF-Q-fVthA0qrFQTgXQ	CaseyNeistat	

Since the data frame looks fine, we are ready for the data exploration part.

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

Out[ ]:
```
(238191, 17)
```

We see that there are `17` features (columns) and `238191` trending videos (rows). To ensure correct analysis, we check if we do not have null items in any column by using isna(), which returns a boolean value and then sums it column-wise.

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
video_id                0
title                   0
publishedAt             0
channelId               0
channelTitle            0
categoryId              0
trending_date           0
tags                    0
view_count              0
likes                   0
dislikes                0
comment_count           0
thumbnail_link          0
comments_disabled       0
ratings_disabled        0
description          4279
category              102
dtype: int64
```

The above output shows that `4279` missing descriptions and `102` missing categories. Since missing values in these columns will not cause problems in our analysis, we will not drop those rows.

Duplicate Check

We also check for duplicate rows in the data frame using the pandas' `duplicated()` function, as duplicate values could lead to bias. We then sum up the total to see how many rows have duplicates. From the result, we see that 124 rows have duplicates.

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

Out[ ]:
```
124
```

We now use the `drop_duplicates()` function to drop those rows. We can then check the data frame's shape to cross-verify the deletion of duplicates.

In [ ]:
```
df = df.drop_duplicates()
df.shape
```

Out[ ]:
```
(238067, 17)
```

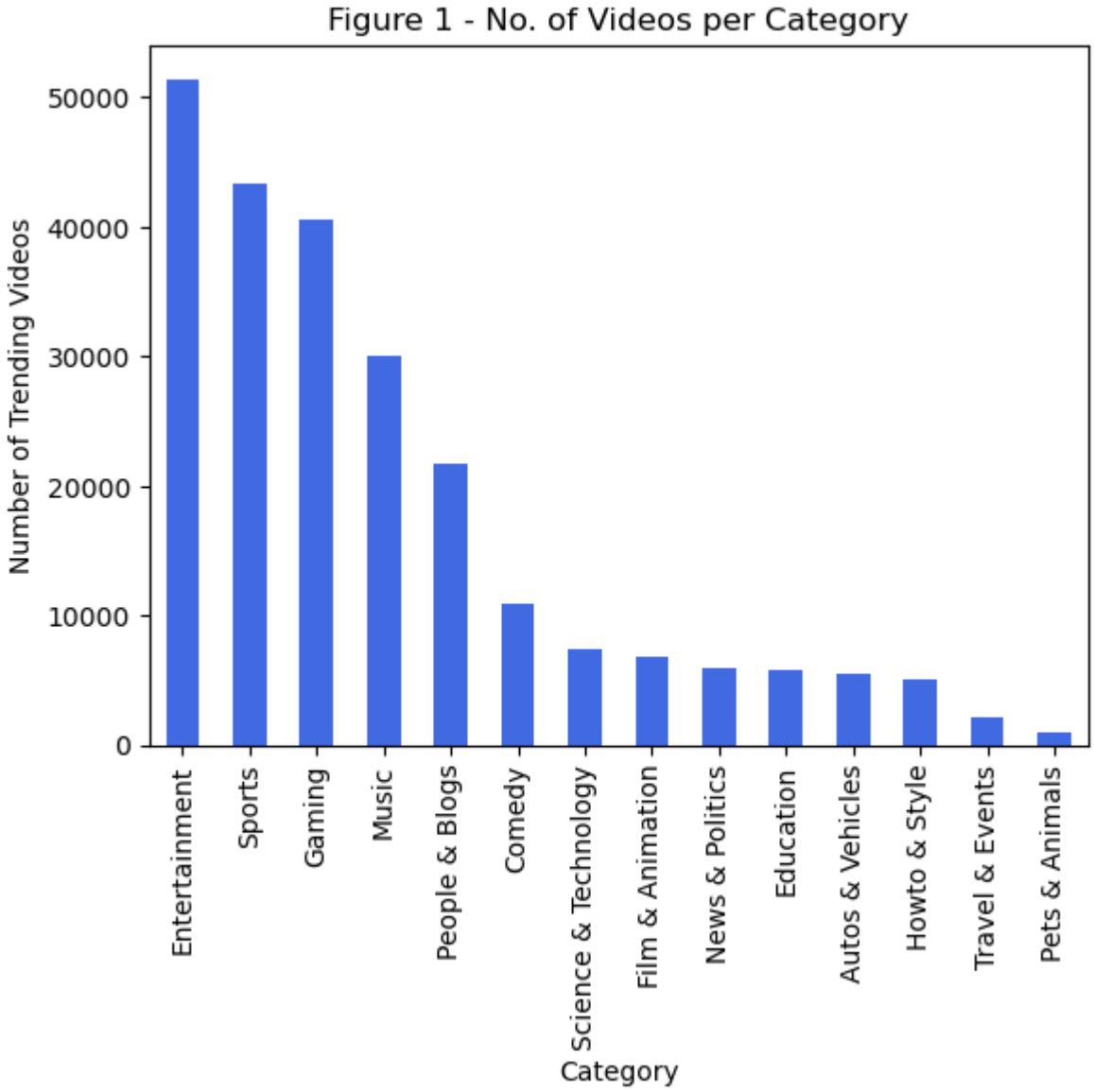
Since the data is clean, we will start exploring the data and working on our objectives.

Objective 1 Code Cells: *Visualize the genres that trend in the United Kingdom*

Firstly, we will explore which category of video trends to know what kind of videos the viewers are intrigued to watch. So, we plot a barplot that shows the number of videos trending in each category.

In [ ]:
```
df['category'].value_counts().plot.bar(title='Figure 1 - No. of Videos per C
```

Out[ ]:
```
<Axes: title={'center': 'Figure 1 - No. of Videos per Category'}, xlabel='C
ategory', ylabel='Number of Trending Videos'>
```



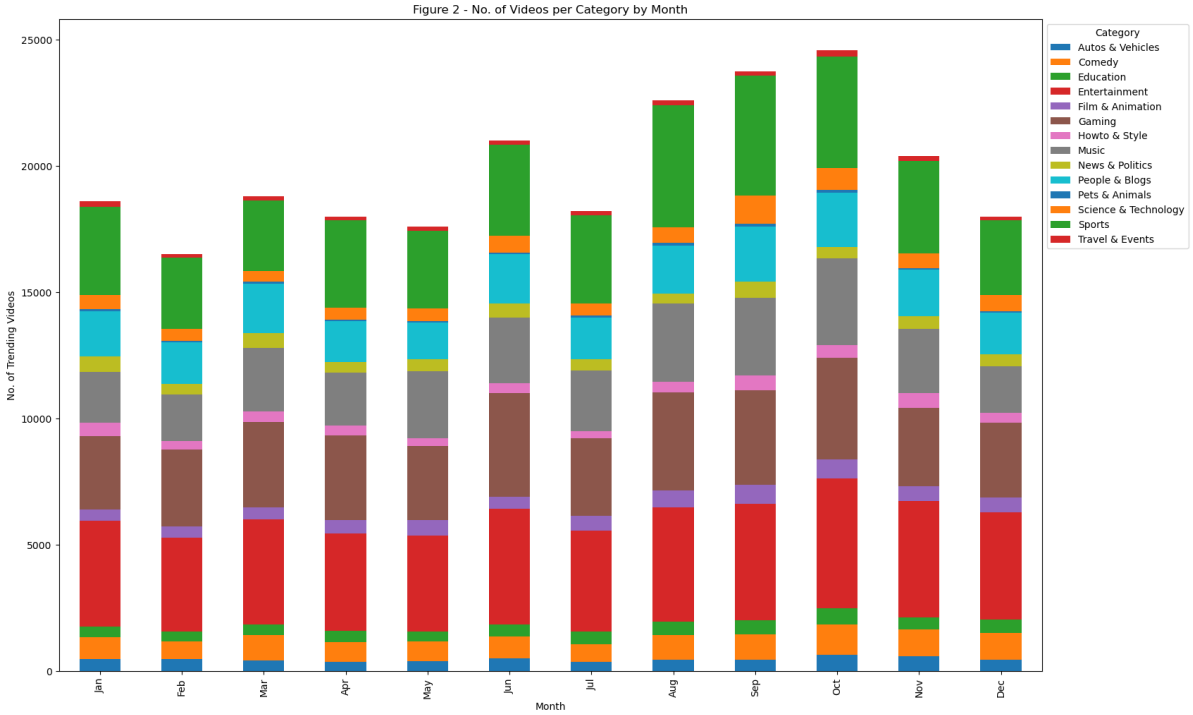
As we can see from Figure 1, Entertainment tops the list, followed by Sports, Gaming, Music, and People & Blogs as the Top 5 Categories across the dataset for the year between Aug 2020 and Nov 2023.

Now that we know the ranking of the categories that trend, we further investigate the nature of the trending categories month-wise. We can do this by converting the trending_date column in the data frame into the date-time datatype and storing it in a new column called date. Further columns are created based on year and month to aid in plotting. We can then label the month from integers to their names in English to enhance clarity. A graph is then plotted based on category and month.

In [ ]:
```
# plot bar graph based on category and date by month
df['date'] = pd.to_datetime(df['trending_date'])
df['month'] = df['date'].dt.month
```

```
df['year'] = df['date'].dt.year
df['month'] = df['month'].replace((1,2,3,4,5,6,7,8,9,10,11,12),('Jan','Feb',
df['month'] = pd.Categorical(df['month'], categories=['Jan','Feb','Mar','Ap
#plot bar graph based on category and date by month
df.groupby(['month','category']).size().unstack().plot(kind='bar',stacked=Tr
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x154bae150>
```



We are using a stacked barplot to compare the number of trending videos each month. As shown in Figure 2, October has the highest number of trending videos, followed by September and August. The least number of trending videos is for February. The top 5 categories from Figure 1 have almost a similar spread across the plot.

The spike in the number of trending videos between August and November may be because of the period range we have in our dataset. That is, our range is between August 2020 to November 2023. This incomplete data in 2020 and 2023 could be a reason to see the spike. We will further probe into finding how big of a difference it is between each year by comparing only the top 5 categories.

The data frame is further filtered to the Top 5 categories per Figure 1.

In [ ]:
```
# plot grouped bar graph for top 5 category per year
df.groupby(['year','category']).size().groupby(level=0).nlargest(5).unstack(
plt.xticks(np.array([0,1,2,3]), ('2020','2021','2022','2023'),rotation=0)
```

Out[ ]:
```
([<matplotlib.axis.XTick at 0x154f83710>,
  <matplotlib.axis.XTick at 0x154f8a190>,
  <matplotlib.axis.XTick at 0x154febd50>,
  <matplotlib.axis.XTick at 0x15717fb90>],
 [Text(0, 0, '2020'),
  Text(1, 0, '2021'),
  Text(2, 0, '2022'),
  Text(3, 0, '2023')])
```

Figure 3 - Top 5 Categories per Year

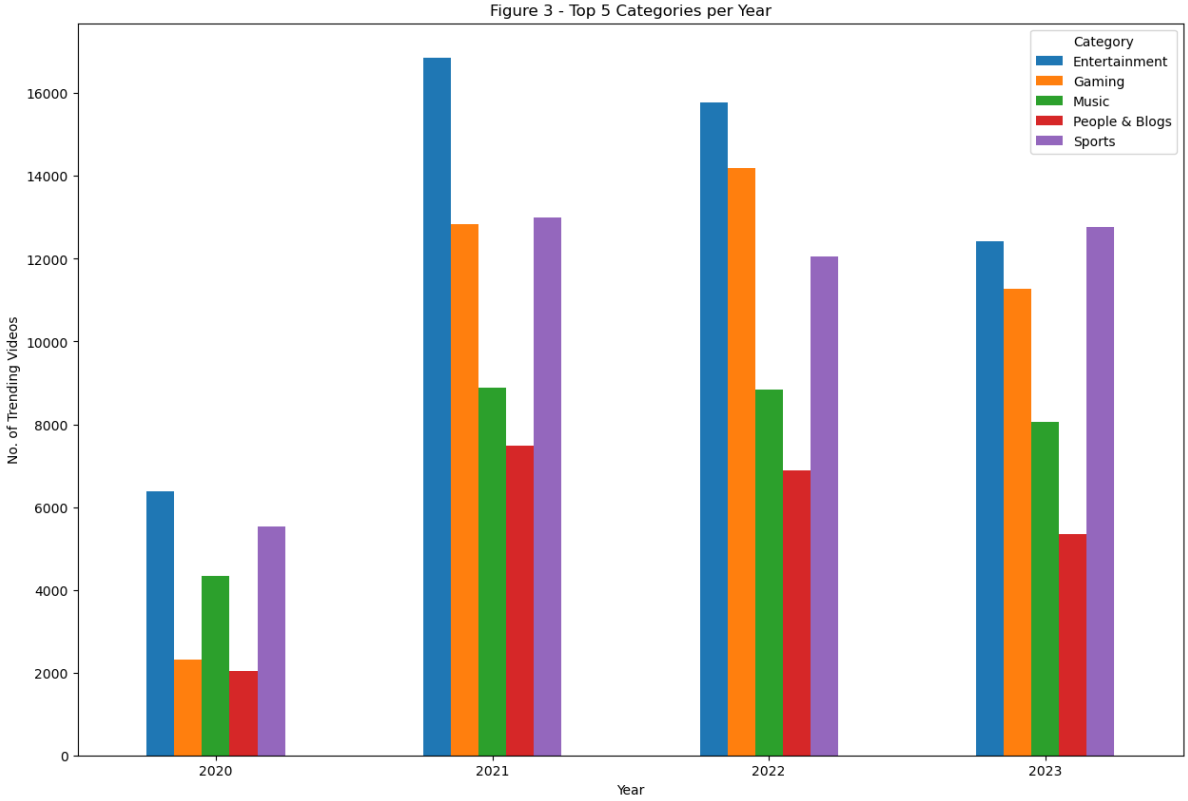


Figure 3 shows that in 2020, there were fewer trending videos than in any other year. As for 2023, the number of trending videos is also less compared to 2021 and 2022. This shows that since we have incomplete data for 2020 and 2023, the height is reduced, which made it spike between August and November in Figure 2.

Objective 2 Code Cells: *Checks the key words to be used in the Title and/or Description for a video to be in the trending category*

To aide in our understanding of how keywords in title affect a videos abilty to trend, we use an external python module called `worldcloud`. This will allow us to visualize frequently mentioned words in the title.

In [ ]:
```
from wordcloud import WordCloud
```

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

Out[ ]:

	video_id	title	publishedAt	channelId	channelTitle	cate
0	J78aPJ3VyNs	I left youtube for a month and THIS is what ha...	2020-08-11T16:34:06Z	UCYzPXprvl5Y-Sf0g4vX-m6g	jacksepticeye	
1	9nidKH8cM38	TAXI CAB SLAYER KILLS 'TO KNOW HOW IT FEELS'	2020-08-11T20:00:45Z	UCFMbX7frWZfuWdjAML0babA	Eleanor Neale	
2	M9Pmf9AB4Mo	Apex Legends | Stories from the Outlands – “Th...	2020-08-11T17:00:10Z	UC0ZV6M2THA81QT9hrVWJG3A	Apex Legends	
3	kgUV1MaD_M8	Nines - Clout (Official Video)	2020-08-10T18:30:28Z	UCvDkzrj8ZPlBqRd6flxdhTw	Nines	
4	49Z6Mv4_WCA	i don't know what im doing anymore	2020-08-11T20:24:34Z	UCtinbF-Q-fVthA0qrFQTgXQ	CaseyNeistat	

We created a df_test3 variable containing df(main data set) where the lengths of the titles are stored in a new column name title_lenght using the .split() function to split them based on spaces and count their size using .len() function. And displayed the mean of title_lenght to know the average length of the title.

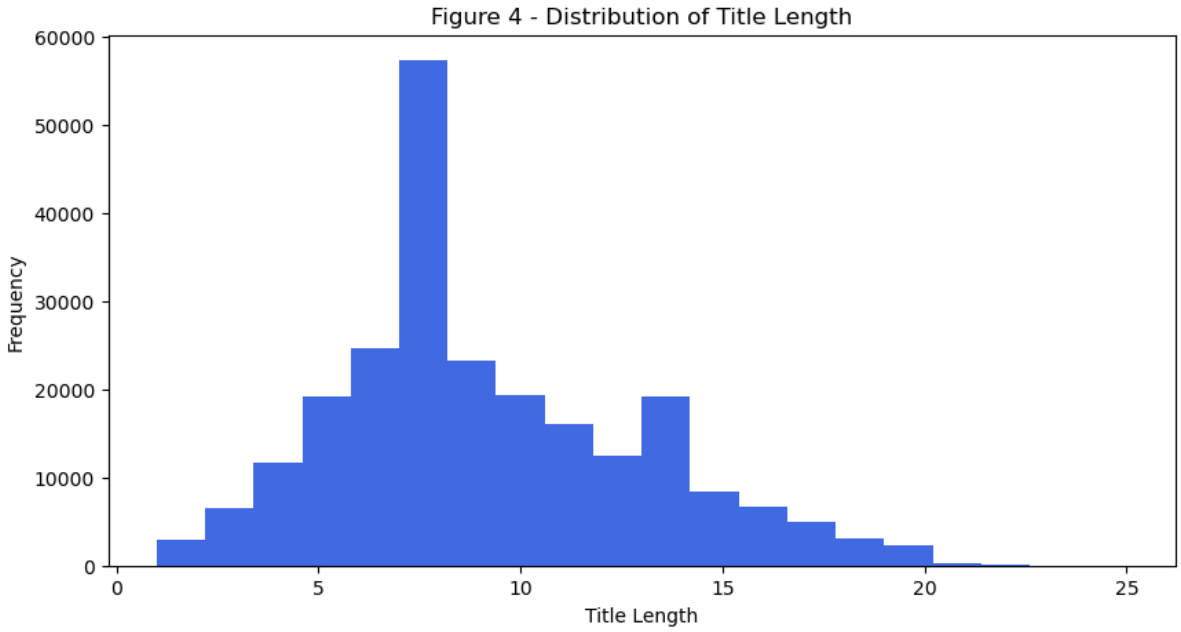
In [ ]:
```
#find the average length of title in words
df_objective2 = df.copy()
df_objective2['title_length'] = df_objective2['title'].str.split().str.len()
df_objective2['title_length'].mean()
```

Out[ ]:
```
9.045142753930616
```

We plot a histogram for title_lenght with a bin size of 20. It explains to divide into 20 equal intervals where the data points are collected in the given interval and labeled with the title and their axis(x="Title Length," y="Frequency"), names

In [ ]:
```
df_objective2['title_length'].plot.hist(bins=20,figsize=(10,5),title="Figure
```

Out[ ]:
```
<Axes: title={'center': 'Figure 4 – Distribution of Title Length'}, xlabel
='Title Length', ylabel='Frequency'>
```



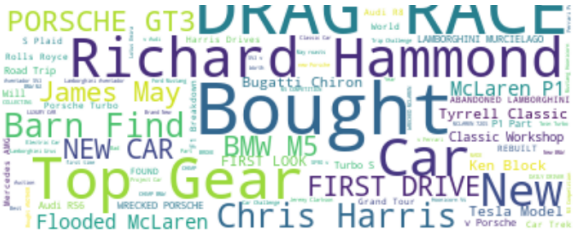
Here, the plot makes more sense due to a higher frequency value closer to 10, which is assumed to be a normal distribution with some outliers.

here we are creating a wordcloud to show the trending video title by category .firstly we created a fig and axes variable with subplots of 7 rows and 2 column . and created categoris variable to contail cotegory names

In [ ]:
```
# Multiple graphs in the same cell
fig, axes = plt.subplots(7, 2, figsize=(15, 30))

# Get unique categories
categories = df_objective2['category'].unique()

# Limit the number of categories to the number of subplots
categories = categories[:len(axes.flatten())]

# Plot the wordcloud for each category
for i, ax in enumerate(axes.flatten()):
    category = categories[i]
    title = df_objective2[df_objective2['category'] == category]['title']
    text = ' '.join(title)
    if text != ' ':
        wordcloud = WordCloud(max_font_size=50, max_words=10000, background_
        ax.imshow(wordcloud, interpolation="bilinear")
        ax.set_title(category)
        ax.axis("off")
        fig.suptitle('Figure 5 - Wordcloud of Trending Video Titles by Categ
        fig.tight_layout(rect=[0, 0.03, 1, 0.95])
```

```
/var/folders/y6/rbs796bx78jcl293lzj57pnm0000gn/T/ipykernel_55561/395844268
8.py:21: UserWarning: The figure layout has changed to tight
  fig.tight_layout(rect=[0, 0.03, 1, 0.95])
```

Figure 5 - Wordcloud of Trending Video Titles by Category



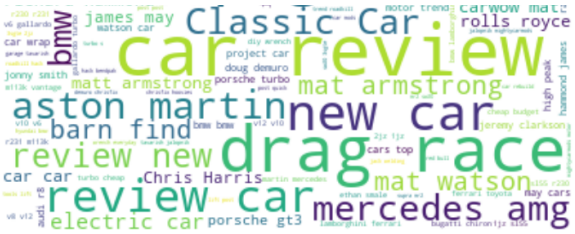
In [ ]:
```
df_objective2_tags = df.copy()
df_objective2_tags['tag_length'] = df_objective2['tags'].str.split().str.len
df_objective2_tags['tag_length'].mean()
```

Out[ ]:
```
17.73617511036809
```

In [ ]:
```
# Multiple graphs in the same cell
fig, axes = plt.subplots(7, 2, figsize=(15, 30))

# Get unique categories
categories = df_objective2['category'].unique()

# Limit the number of categories to the number of subplots
categories = categories[:len(axes.flatten())]

# Plot the wordcloud for each category
for i, ax in enumerate(axes.flatten()):
    category = categories[i]
    tags = df_objective2[df_objective2['category'] == category]['tags']
    text = ' '.join(tags)
    if text != ' ' or text == 'None':
        wordcloud = WordCloud(max_font_size=50, max_words=10000, background_
        ax.imshow(wordcloud, interpolation="bilinear")
        ax.set_title(category)
        ax.axis("off")
        fig.suptitle('Figure 6 - Wordcloud of Trending Video tags by Categor
        fig.tight_layout(rect=[0, 0.03, 1, 0.95])
```

```
/var/folders/y6/rbs796bx78jcl293lzj57pnm0000gn/T/ipykernel_55561/394838813
5.py:21: UserWarning: The figure layout has changed to tight
  fig.tight_layout(rect=[0, 0.03, 1, 0.95])
```

Figure 6 - Wordcloud of Trending Video tags by Category



In [ ]:
```
def wordcloud_gen_v2(category):

    text1 = ' '.join(df_objective2[df_objective2['category'] == category]['t
    text2 = ' '.join(df_objective2[df_objective2['category'] == category]['t

    from collections import Counter

    # Generate word frequency dictionaries
    word_freq1 = Counter(text1.split())
    word_freq2 = Counter(text2.split())

    # Find common words
    common_words = word_freq1 & word_freq2

    # Generate a word cloud using only the common words
    common_text = ' '.join(common_words.elements())
    if common_text != ' ' or common_text == 'None':
        wordcloud = WordCloud(max_font_size = 50,max_words=10000, backgroun
    return wordcloud
```

In [ ]:
```
# Multiple graphs in the same cell
fig, axes = plt.subplots(7, 2, figsize=(15, 30))

# Get unique categories
categories = df_objective2['category'].unique()

# Limit the number of categories to the number of subplots
categories = categories[:len(axes.flatten())]

# Plot the wordcloud for each category
for i, ax in enumerate(axes.flatten()):
    category = categories[i]
    #tags = df_test3[df_test3['category'] == category]['tags']
    #text = ' '.join(tags)

    if text != ' ' or text == 'None':
        ax.imshow(wordcloud_gen_v2(category), interpolation="bilinear")
        ax.set_title(category)
        ax.axis("off")
        fig.suptitle('Figure 7 - Wordcloud of Common Words in Trending Video
        fig.tight_layout(rect=[0, 0.03, 1, 0.95])
```

```
/var/folders/y6/rbs796bx78jcl293lzj57pnm0000gn/T/ipykernel_55561/311149007
3.py:21: UserWarning: The figure layout has changed to tight
  fig.tight_layout(rect=[0, 0.03, 1, 0.95])
```

Figure 7 - Wordcloud of Common Words in Trending Video Tags and Title by Category

Objective 3 Code Cells

We have to see all the columns but use only the ones we need for objective 3. Thus, we use the column method that returns the column names in the data frame.

In [ ]:
```
#find column names of dataframe
df.columns
#As you can see in the figure 3, we have taken the top five trending categor
#We can see a downtrend in Entertainment from year 2021 to year 2023, while
```

Out[ ]:
```
Index(['video_id', 'title', 'publishedAt', 'channelId', 'channelTitle',
       'categoryId', 'trending_date', 'tags', 'view_count', 'likes',
       'dislikes', 'comment_count', 'thumbnail_link', 'comments_disabled',
       'ratings_disabled', 'description', 'category', 'date', 'month', 'yea
r'],
      dtype='object')
```

Since we now have 20 columns, we drop the unnecessary columns to explore the data using the drop() method.

In [ ]:
```
#drop columns that are not needed - video_id, channelId, tags, channelTitle,
df_objective3 = df.copy()
df_objective3 = df.drop(['video_id','channelId','tags','channelTitle','comme
df_objective3.head()
#dfme = df_ob3[df_ob3['category'] == 'Education'].sort_values(by='likes', as
#dfme.head(30)
```

Out[ ]:

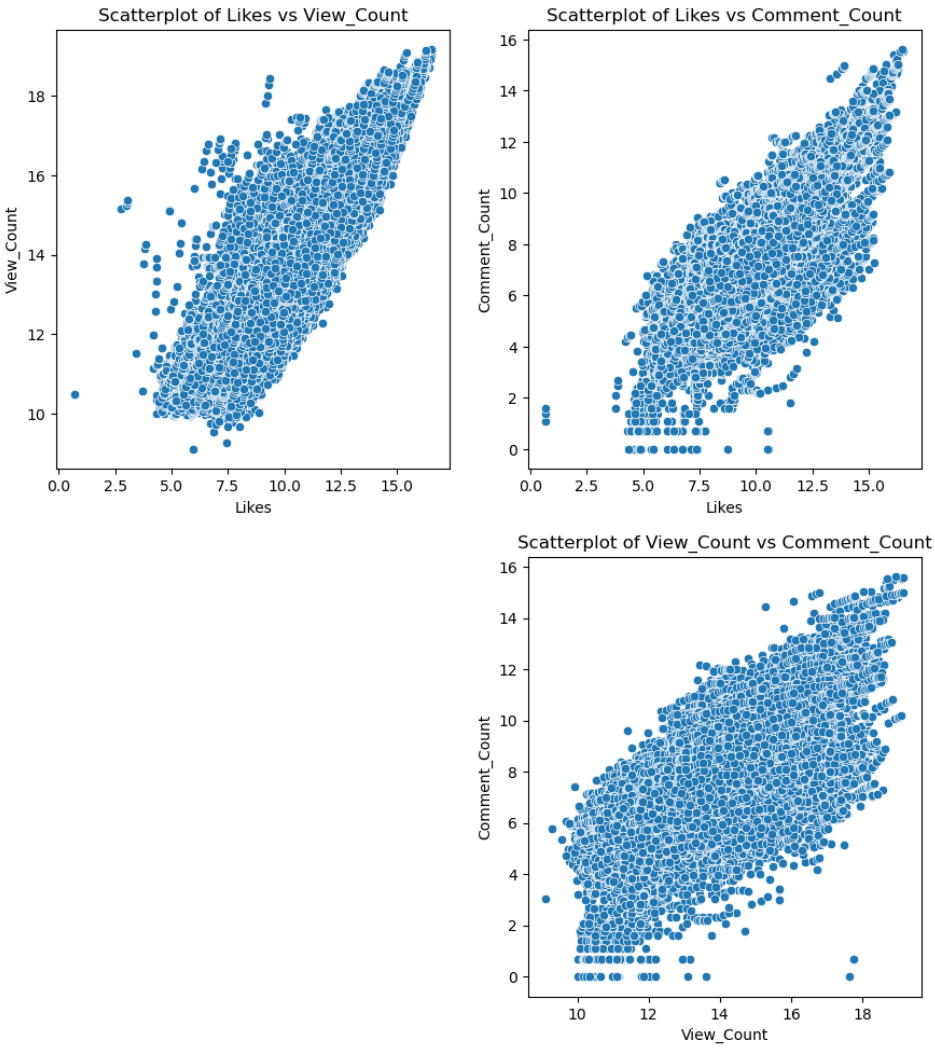
	title	publishedAt	categoryId	trending_date	view_count	likes	dislikes	cat
0	I left youtube for a month and THIS is what ha...	2020-08-11T16:34:06Z	24	2020-08-12T00:00:00Z	2038853	353790	2628	Entertair
1	TAXI CAB SLAYER KILLS 'TO KNOW HOW IT FEELS'	2020-08-11T20:00:45Z	27	2020-08-12T00:00:00Z	236830	16423	209	Educ
2	Apex Legends | Stories from the Outlands – "Th...	2020-08-11T17:00:10Z	20	2020-08-12T00:00:00Z	2381688	146739	2794	Ga
3	Nines - Clout (Official Video)	2020-08-10T18:30:28Z	24	2020-08-12T00:00:00Z	613785	37567	669	Entertair
4	i don't know what im doing anymore	2020-08-11T20:24:34Z	22	2020-08-12T00:00:00Z	940036	87113	1860	Pec

We now check the relation between likes, views, and comment count using scatterplot.

In [ ]:
```
columns = ['likes', 'view_count', 'comment_count']
plt.figure(figsize=(16, 18))
plt.suptitle('Figure 8 - Scatterplot of Likes, View Count and Comment Count
for i in range(len(columns)):
    for j in range(i+1, len(columns)):
        plt.subplot(len(columns), len(columns), i*len(columns) + j + 1)
        sns.scatterplot(x=np.log(df[columns[i]]), y=np.log(df[columns[j]]))
```

```
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Figure 8 - Scatterplot of Likes, View Count and Comment Count



In Figure 8, we see that likes, view count, and comment count have a linear relationship, which means if the likes increase, the view count increases, and then the comment count also increases.

We then look deeper into likes and view counts for each category and look for the similarity. However, we also dropped the null rows.

Relation between view count and category

In [ ]:
```
df_objective3 = df_objective3.dropna()

plt.scatter(df_objective3['view_count'],df_objective3['category'].astype(str
plt.xlabel('View Count [in 100 million]')
plt.ylabel('Category')
plt.title('Figure 9 - View count vs. Category')
plt.show()
```

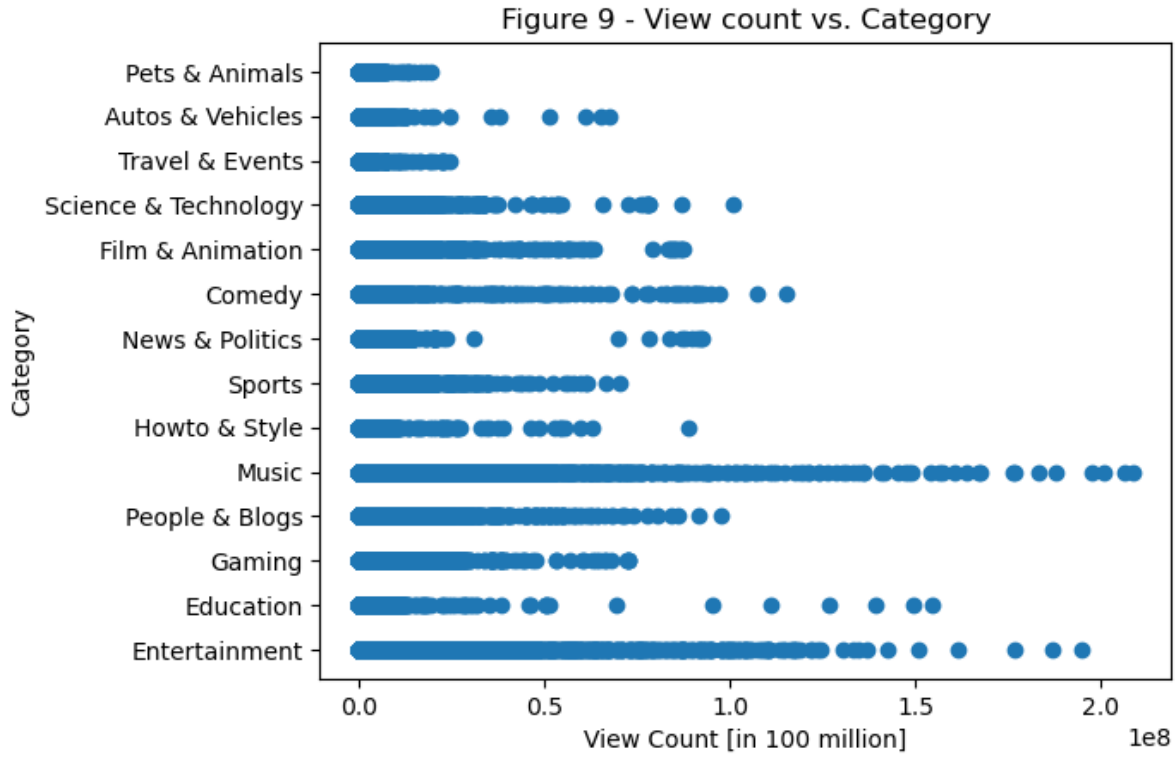


Figure 9 shows the view count in millions for each category, where Music has the highest number of views, followed by Entertainment and Education.

Relation between likes and category

In [ ]:
```
plt.scatter(df_objective3['likes'],df_objective3['category'].astype(str))
plt.xlabel('Likes [in 100 million]')
plt.ylabel('Category')
plt.title('Figure 10 - Likes vs. Category')
plt.show()
```

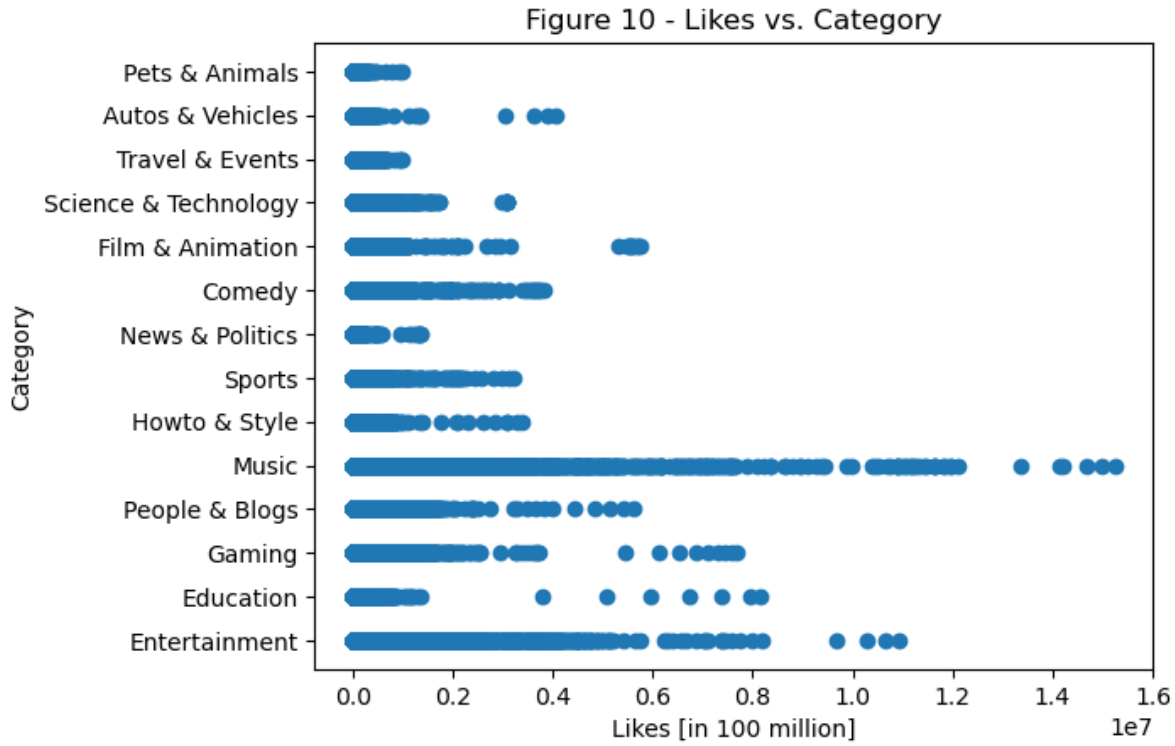


Figure 10 shows the likes in millions for each category, where Music has the most likes, followed by Entertainment and Education. These two graphs show similar results, so

looking from the perspective of view_count alone provides similar results. We chose view count since the values are higher than that of likes.

We then count the number of videos that are higher than the mean of the view count.

In [ ]:
```
df_objective3[df_objective3['view_count'] > df_objective3['view_count'].mean
```

Out[ ]:
```
title            51028
publishedAt      51028
categoryId       51028
trending_date    51028
view_count       51028
likes            51028
dislikes         51028
category         51028
dtype: int64
```

We also check the number of videos that have 0 counts by using the count() method.

In [ ]:
```
df_objective3['title'][df_objective3['view_count'] == 0].count()
```

Out[ ]:
```
94
```

Here, we see 94 videos that have made it to the trending dataset with 0 view count, so these can be considered anomalies. From the graphs, the higher the view count, the more likely it is recommended to users in that particular category than the ones with a lower view count.

We also now count the number of videos higher than the mean of the likes for further investigation.

In [ ]:
```
df_objective3['title'][df_objective3['likes'] > df_objective3['likes'].mean(
```

Out[ ]:
```
49245
```

We find the maximum value of likes the dataset has using the max() method.

In [ ]:
```
df_objective3['likes'].max()
```

Out[ ]:
```
15246514
```

We also enquired about the mean of the likes and view count for each category.

In [ ]:
```
df_objective3.groupby('category')['likes'].mean()
```

Out[ ]:
```
category
Autos & Vehicles         41388.621114
Comedy                  115153.003197
Education                96781.728007
Entertainment           130497.046563
Film & Animation        116169.649217
Gaming                   88443.437217
Howto & Style            65767.418930
Music                   284905.892479
News & Politics          19181.273624
People & Blogs           91500.477865
Pets & Animals           50546.548057
Science & Technology     96778.695623
Sports                   27900.184374
Travel & Events          33063.363850
Name: likes, dtype: float64
```

In [ ]:
```
df_objective3.groupby('category')['view_count'].mean()
```

Out[ ]:
```
category
Autos & Vehicles        9.365756e+05
Comedy                  1.841453e+06
Education               1.830250e+06
Entertainment           2.697683e+06
Film & Animation        3.202573e+06
Gaming                  1.625279e+06
Howto & Style           1.327054e+06
Music                   4.267132e+06
News & Politics         1.419264e+06
People & Blogs          1.792461e+06
Pets & Animals          1.265167e+06
Science & Technology    2.442207e+06
Sports                  1.146422e+06
Travel & Events         7.483659e+05
Name: view_count, dtype: float64
```

Now that we have all the necessary information, we create a new column called likes_mean, where we get a boolean value of 1 if the like count is greater than the mean of the likes; if not, we get 0.

In [ ]:
```
#create a new column where, if the number of likes is greater than the mean,

df_objective3['likes_mean'] = (df_objective3['likes'] > df_objective3['likes
df_objective3.head()
```

Out[ ]:

	title	publishedAt	categoryId	trending_date	view_count	likes	dislikes	cat
0	I left youtube for a month and THIS is what ha...	2020-08-11T16:34:06Z	24	2020-08-12T00:00:00Z	2038853	353790	2628	Entertair
1	TAXI CAB SLAYER KILLS 'TO KNOW HOW IT FEELS'	2020-08-11T20:00:45Z	27	2020-08-12T00:00:00Z	236830	16423	209	Educ
2	Apex Legends | Stories from the Outlands – "Th...	2020-08-11T17:00:10Z	20	2020-08-12T00:00:00Z	2381688	146739	2794	Ga
3	Nines - Clout (Official Video)	2020-08-10T18:30:28Z	24	2020-08-12T00:00:00Z	613785	37567	669	Entertair
4	i don't know what im doing anymore	2020-08-11T20:24:34Z	22	2020-08-12T00:00:00Z	940036	87113	1860	Pec

We plot graphs to find out the following: (i) Relation between Likes_mean and category (Figure 11) (ii) Relation between the 50th quantile of likes and category (Figure 12) (iii) Relation between view counts mean and category (Figure 13) (iv) Relation between the 50th quantile of view counts and category (Figure 14)

In [ ]:
```
#plot graph between likes_mean and category
df_objective3.groupby(['likes_mean','category']).size().unstack().plot(kind=
```

Out[ ]:
```
[Text(0, 0, 'Likes Lesser than Mean'), Text(1, 0, 'Lkes Greater than Mea
n')]
```

Figure 11 - Likes Mean vs. Category

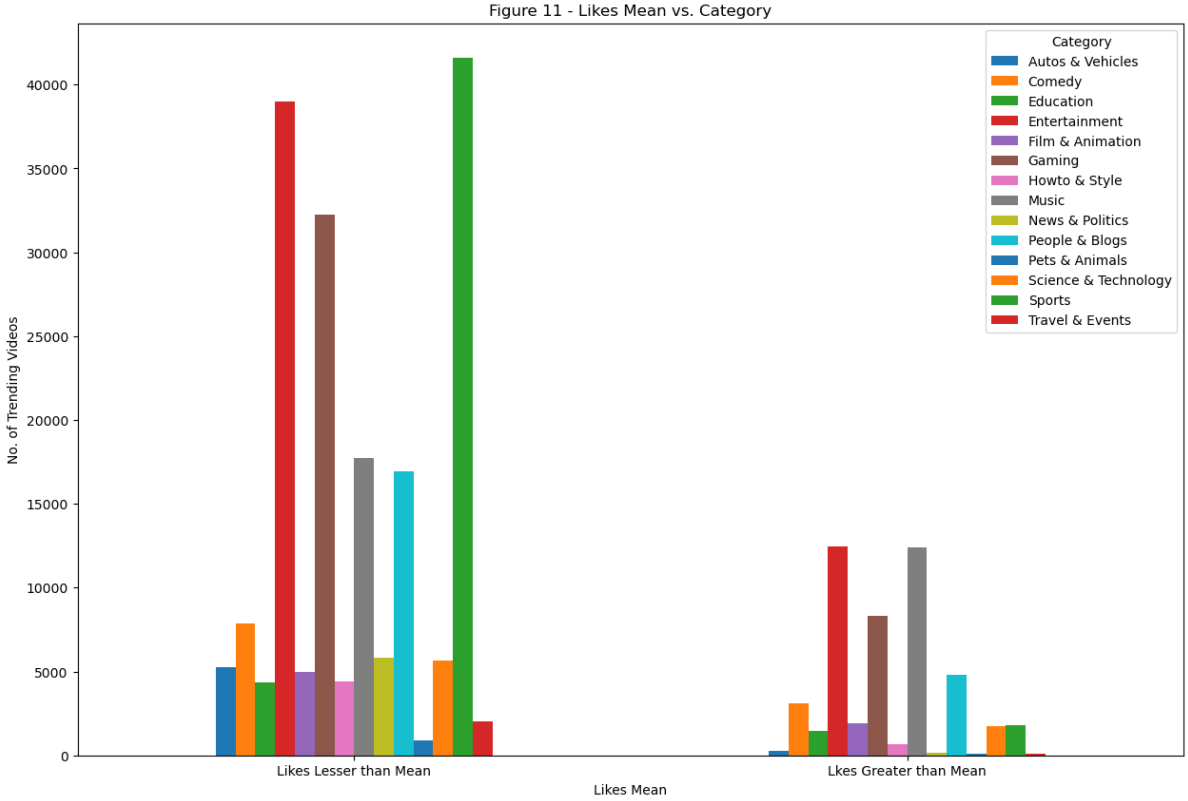


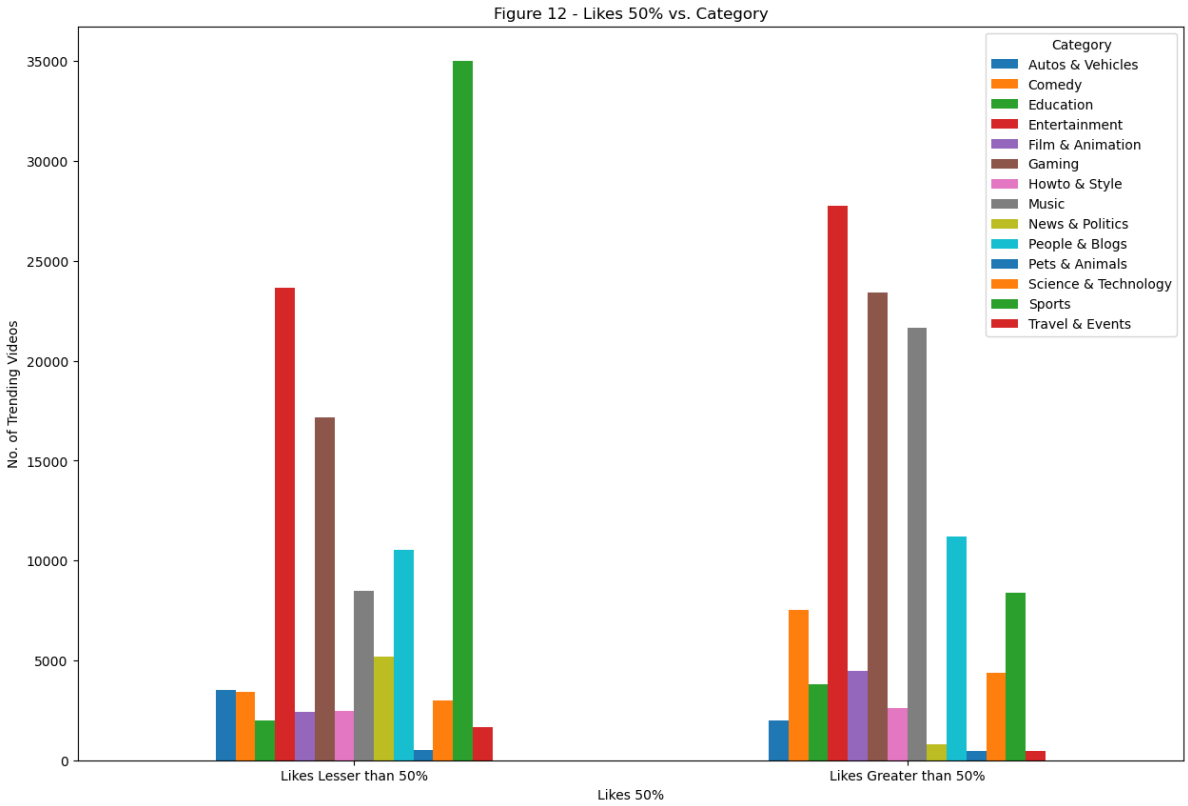
Figure 17 shows several video trends even if the likes exceed the mean. Let us then take a 50% quantile to better visualize the data.

In [ ]:
```
df_objective3['likes_50'] = (df_objective3['likes'] > df_objective3['likes']
```

In [ ]:
```
df_objective3.groupby(['likes_50','category']).size().unstack().plot(kind='b
```

Out[ ]:
```
[Text(0, 0, 'Likes Lesser than 50%'), Text(1, 0, 'Likes Greater than 50%')]
```

Figure 12 - Likes 50% vs. Category



In Figure 12, we see more trending videos above the 50 percentile in the Entertainment, Gaming, and Music categories, while there are more trending videos in Sports below the

50 percentile.

We follow the same procedure to further know about view count.

In [ ]:
```
df_objective3['view_count_mean'] = (df_objective3['view_count'] > df_object
```

In [ ]:
```
df_objective3['view_count_50'] = (df_objective3['view_count'] > df_objective
df_objective3.head()
```

Out[ ]:

	title	publishedAt	categoryId	trending_date	view_count	likes	dislikes	cat
0	I left youtube for a month and THIS is what ha...	2020-08-11T16:34:06Z	24	2020-08-12T00:00:00Z	2038853	353790	2628	Entertair
1	TAXI CAB SLAYER KILLS 'TO KNOW HOW IT FEELS'	2020-08-11T20:00:45Z	27	2020-08-12T00:00:00Z	236830	16423	209	Educ
2	Apex Legends | Stories from the Outlands – "Th...	2020-08-11T17:00:10Z	20	2020-08-12T00:00:00Z	2381688	146739	2794	Ga
3	Nines - Clout (Official Video)	2020-08-10T18:30:28Z	24	2020-08-12T00:00:00Z	613785	37567	669	Entertair
4	i don't know what im doing anymore	2020-08-11T20:24:34Z	22	2020-08-12T00:00:00Z	940036	87113	1860	Peo

We will also take a 25 percentile.

In [ ]:
```
df_objective3['view_count_25'] = (df_objective3['view_count'] > df_objective
df_objective3.head()
```

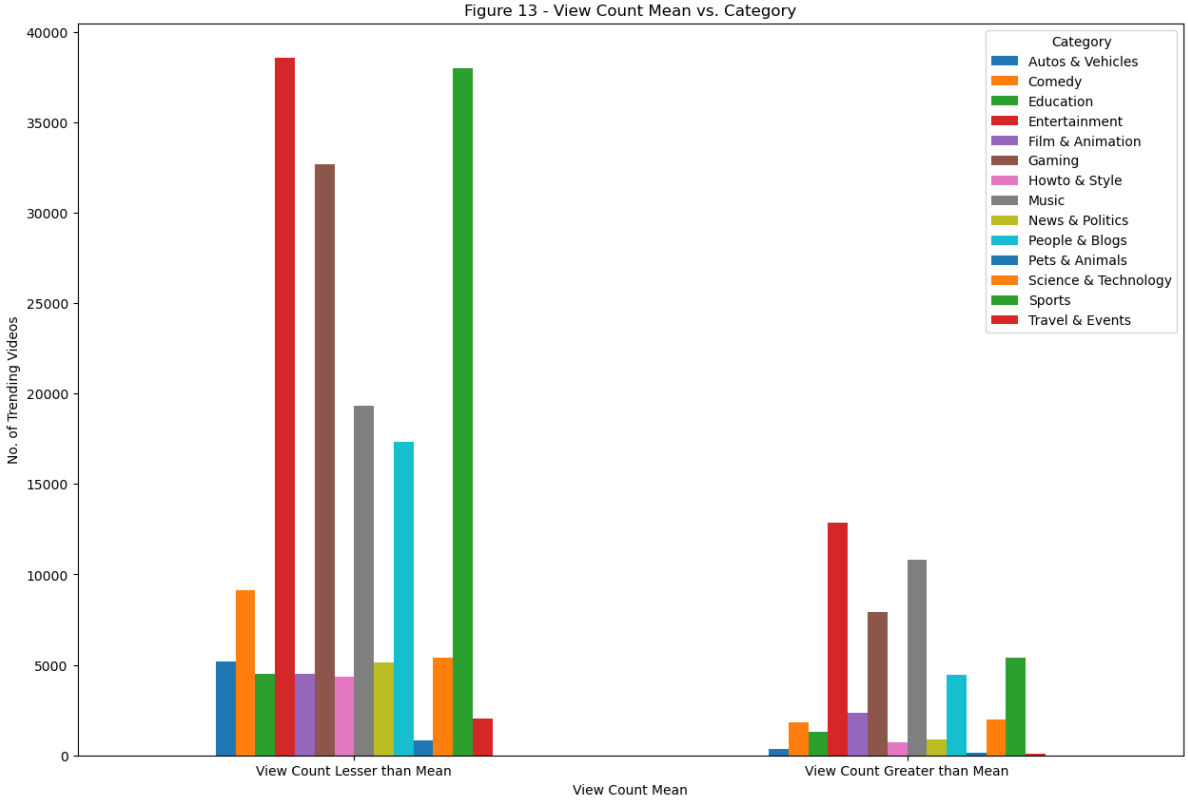
Out[ ]:

	title	publishedAt	categoryId	trending_date	view_count	likes	dislikes	cat
0	I left youtube for a month and THIS is what ha...	2020-08-11T16:34:06Z	24	2020-08-12T00:00:00Z	2038853	353790	2628	Entertair
1	TAXI CAB SLAYER KILLS 'TO KNOW HOW IT FEELS'	2020-08-11T20:00:45Z	27	2020-08-12T00:00:00Z	236830	16423	209	Educ
2	Apex Legends | Stories from the Outlands – "Th...	2020-08-11T17:00:10Z	20	2020-08-12T00:00:00Z	2381688	146739	2794	Ga
3	Nines - Clout (Official Video)	2020-08-10T18:30:28Z	24	2020-08-12T00:00:00Z	613785	37567	669	Entertair
4	i don't know what im doing anymore	2020-08-11T20:24:34Z	22	2020-08-12T00:00:00Z	940036	87113	1860	Pec

In [ ]:
```
df_objective3.groupby(['view_count_mean','category']).size().unstack().plot(
```

Out[ ]:
```
[Text(0, 0, 'View Count Lesser than Mean'),
 Text(1, 0, 'View Count Greater than Mean')]
```

Figure 13 - View Count Mean vs. Category



We see the same trend in Figure 13, just like in Figure 11. Most of the videos trending below the mean of the view count.

In [ ]:
```
df_objective3.groupby(['view_count_50','category']).size().unstack().plot(k
```

Out[ ]:
```
[Text(0, 0, 'View Count Lesser than 50%'),
 Text(1, 0, 'View Count Greater than 50%')]
```

Figure 14 - View Count 50% vs. Category

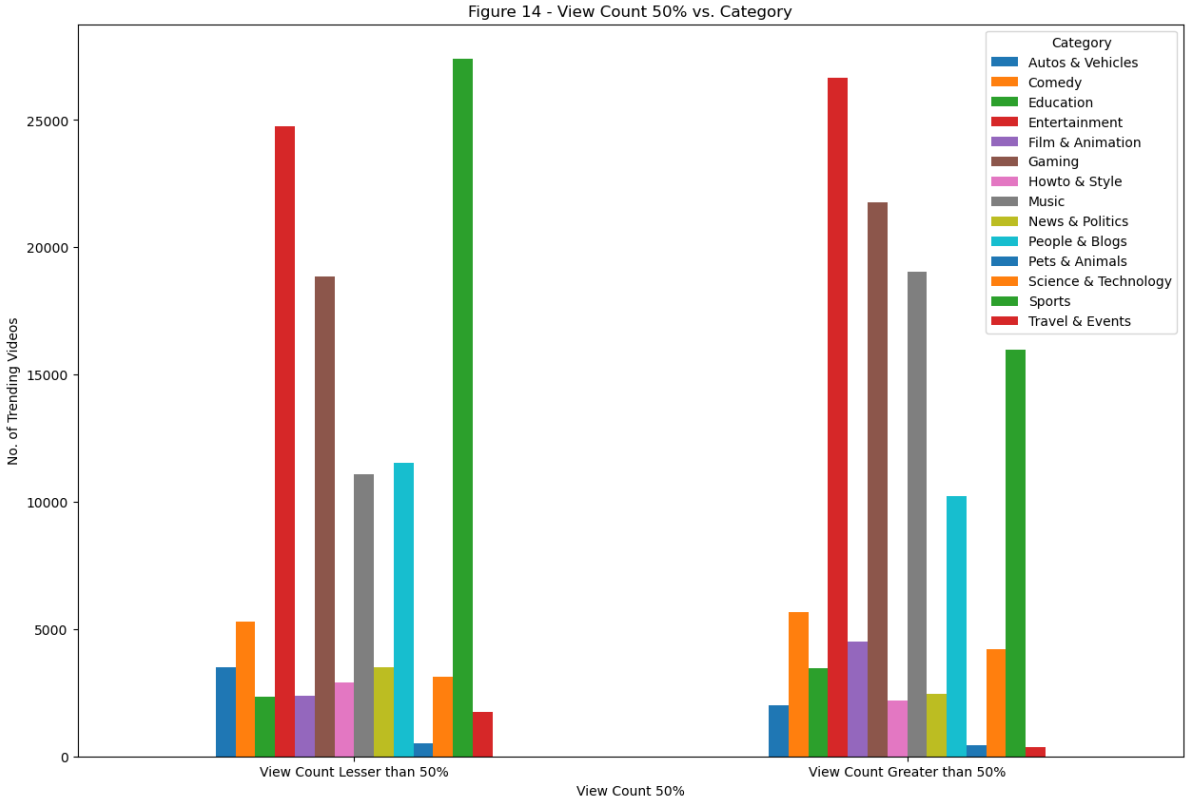
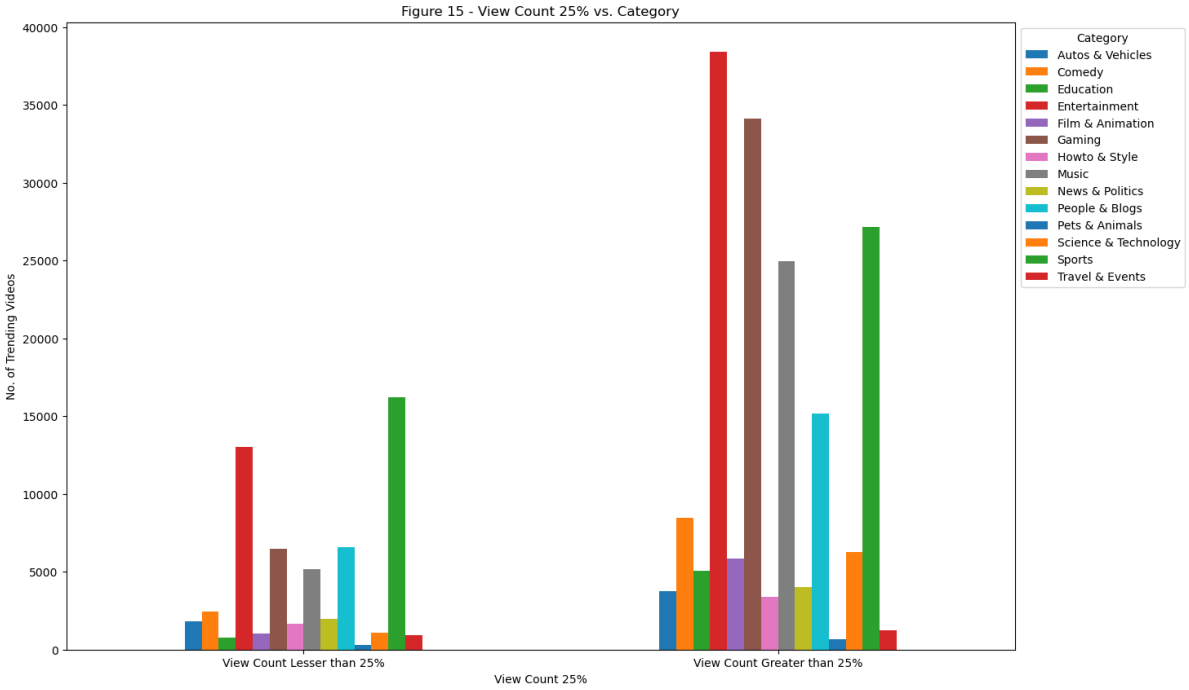


Figure 14 shows the 50 percentile in view count. Some have the same number of videos on both sides, while more trending Sports videos are less than the 50 percentile, and more Gaming, Entertainment, and Music videos are above the 50 percentile.

We see a massive number of likes and view counts for the top 5 categories compared to the rest.

In [ ]:
```
df_objective3.groupby(['view_count_25','category']).size().unstack().plot(k
```

Out[ ]:
```
[Text(0, 0, 'View Count Lesser than 25%'),
 Text(1, 0, 'View Count Greater than 25%')]
```



Then, Figure 15 suggests the graph for the 25 percentile, and we see a similar trend here as well.

We further explore the ratio of view count to likes and the ratio of like to view count and save it in two new columns called `view_to_like_ratio` and `like_to_view_ratio` respectively.

In [ ]:
```
#find the ratio of view_count to likes
df_objective3['view_to_like_ratio'] = df_objective3['view_count']/df_object
df_objective3['like_to_view_ratio'] = df_objective3['likes']/df_objective3[
df_objective3.head()
```

Out[ ]:

	title	publishedAt	categoryId	trending_date	view_count	likes	dislikes	cat
0	I left youtube for a month and THIS is what ha...	2020-08-11T16:34:06Z	24	2020-08-12T00:00:00Z	2038853	353790	2628	Entertair
1	TAXI CAB SLAYER KILLS 'TO KNOW HOW IT FEELS'	2020-08-11T20:00:45Z	27	2020-08-12T00:00:00Z	236830	16423	209	Educ
2	Apex Legends | Stories from the Outlands – "Th...	2020-08-11T17:00:10Z	20	2020-08-12T00:00:00Z	2381688	146739	2794	Ga
3	Nines - Clout (Official Video)	2020-08-10T18:30:28Z	24	2020-08-12T00:00:00Z	613785	37567	669	Entertair
4	i don't know what im doing anymore	2020-08-11T20:24:34Z	22	2020-08-12T00:00:00Z	940036	87113	1860	Pec

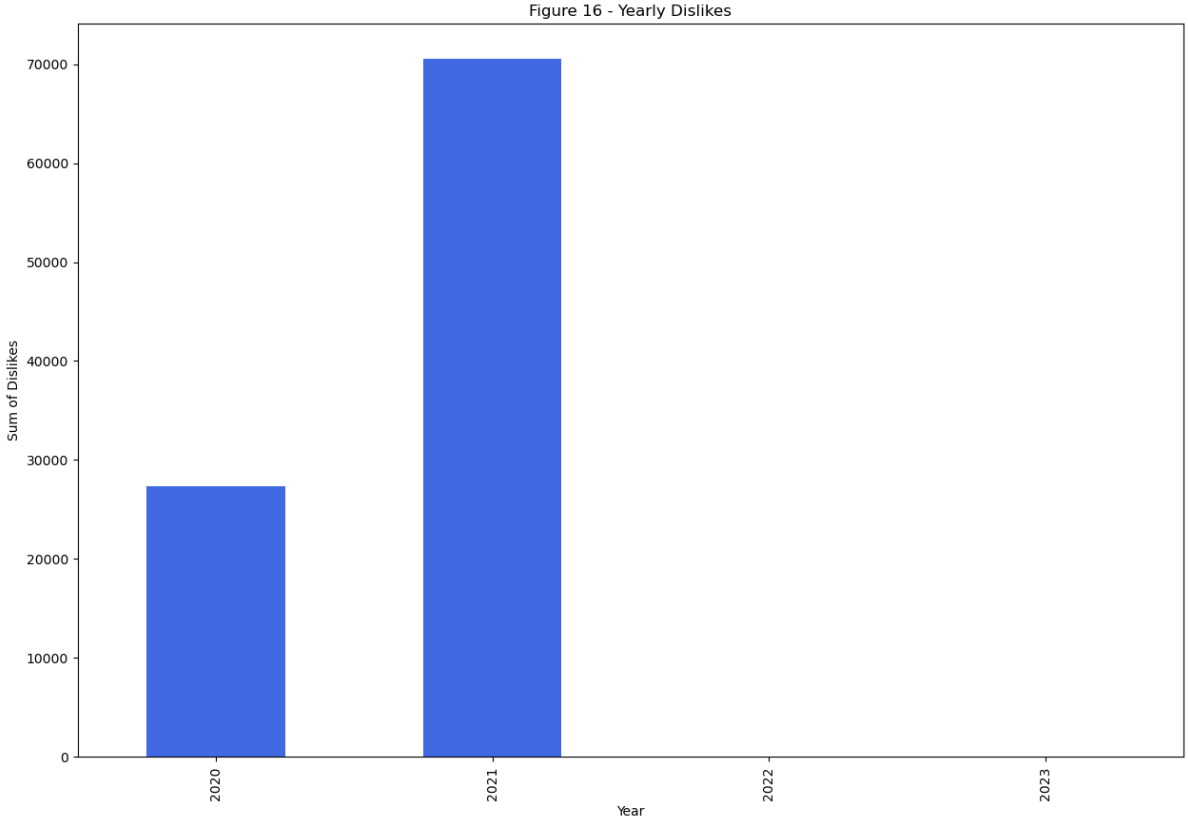
With the help of the ratios, we see that we get a single like for multiple view counts, which means we do not get likes for every view count. This is even true with the reality where a music video is played repeatedly, but the user can like the video only once.

While we are on the subject of likes, we do not use dislikes for our analysis. This is because Youtube's API has stopped registering dislikes count from 2022. We will plot a barplot to visualize if this is true.

In [ ]:
```
# Extract year from 'date' column
df_yearly_dislikes = df.copy()
df_yearly_dislikes['year'] = df_yearly_dislikes['date'].dt.year
df_yearly_dislikes['dislikes_nonzero'] = df['dislikes'].apply(lambda x: 0 if

# Group by year and calculate sum of 'dislikes_zero'
yearly_dislikes_nonzero = df_yearly_dislikes.groupby('year')['dislikes_nonze

# Plot bar graph
yearly_dislikes_nonzero.plot(kind='bar', figsize=(15,10), title="Figure 16 -
```

Out[ ]:
```
<Axes: title={'center': 'Figure 16 - Yearly Dislikes'}, xlabel='Year', ylab
el='Sum of Dislikes'>
```

Figure 16 - Yearly Dislikes



Thus, from Figure 17, we can see there are dislikes in 2020 and 2021 alone. The empty bins for 2022 and 2023 suggest no data for dislikes count. The reason behind the dislikes count for 2020 being less than 2021 is that we need the complete data for 2020 (From January to December).

Test visualisations

We will now deeply explore into the dataset to find about other variables such as the time and the month the video published if it affects the trend of a video. We use the `info()` method to see if we have any null items, and see the datatypes and columns.

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

```
<class 'pandas.core.frame.DataFrame'>
Index: 238067 entries, 0 to 238190
Data columns (total 20 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   video_id           238067 non-null  object
 1   title              238067 non-null  object
 2   publishedAt        238067 non-null  object
 3   channelId          238067 non-null  object
 4   channelTitle       238067 non-null  object
 5   categoryId         238067 non-null  int64
 6   trending_date      238067 non-null  object
 7   tags               238067 non-null  object
 8   view_count         238067 non-null  int64
 9   likes              238067 non-null  int64
 10  dislikes           238067 non-null  int64
 11  comment_count      238067 non-null  int64
 12  thumbnail_link     238067 non-null  object
 13  comments_disabled  238067 non-null  bool
 14  ratings_disabled   238067 non-null  bool
 15  description        233793 non-null  object
 16  category           237965 non-null  object
 17  date               238067 non-null  datetime64[ns, UTC]
 18  month              238067 non-null  category
 19  year               238067 non-null  int32
dtypes: bool(2), category(1), datetime64[ns, UTC](1), int32(1), int64(5), o
bject(10)
memory usage: 32.5+ MB
```

Now that there is not any null items, we will convert the `publishedAt` column to Date-time datatype, since it is in object datatype currently. Then, we will take the published month and hour and append them in the two new columns called `num_month` and `hour` respectively.

In [ ]:
```
df['publishedAt'] = pd.to_datetime(df['publishedAt'])
```

In [ ]:
```
df['num_month'] = df['publishedAt'].dt.month
df['hour'] = df['publishedAt'].dt.hour
df.head()
```

Out[ ]:

	video_id	title	publishedAt	channelId	channelTitle	ca
0	J78aPJ3VyNs	I left youtube for a month and THIS is what ha...	2020-08-11 16:34:06+00:00	UCYzPXprvl5Y-Sf0g4vX-m6g	jacksepticeye	
1	9nidKH8cM38	TAXI CAB SLAYER KILLS 'TO KNOW HOW IT FEELS'	2020-08-11 20:00:45+00:00	UCFMbX7frWZfuWdjAML0babA	Eleanor Neale	
2	M9Pmf9AB4Mo	Apex Legends | Stories from the Outlands – "Th...	2020-08-11 17:00:10+00:00	UC0ZV6M2THA81QT9hrVWJG3A	Apex Legends	
3	kgUV1MaD_M8	Nines - Clout (Official Video)	2020-08-10 18:30:28+00:00	UCvDkzrj8ZPlBqRd6flxdhTw	Nines	
4	49Z6Mv4_WCA	i don't know what im doing anymore	2020-08-11 20:24:34+00:00	UCtinbF-Q-fVthA0qrFQTgXQ	CaseyNeistat	

5 rows × 22 columns

From the above first five rows, we can cross-check if the month and hour published are correctly placed. Then, we create a new list called `corr_list` to see the correlation among the listed variables.

Correlation means that there is a relationship between two things, but does not always mean that one causes the other. It ranges from -1 to 1, where 0 denotes 'No correlation' and 1 denotes positively correlated (If one variable increases, the other variable also increases), while -1 denotes negatively correlated (If one variable increases, the other variable decreases). We aim to see how close it is to -1 and 1 for correlation, the closer the more correlated.

We use the `corr()` method to get the correlation.

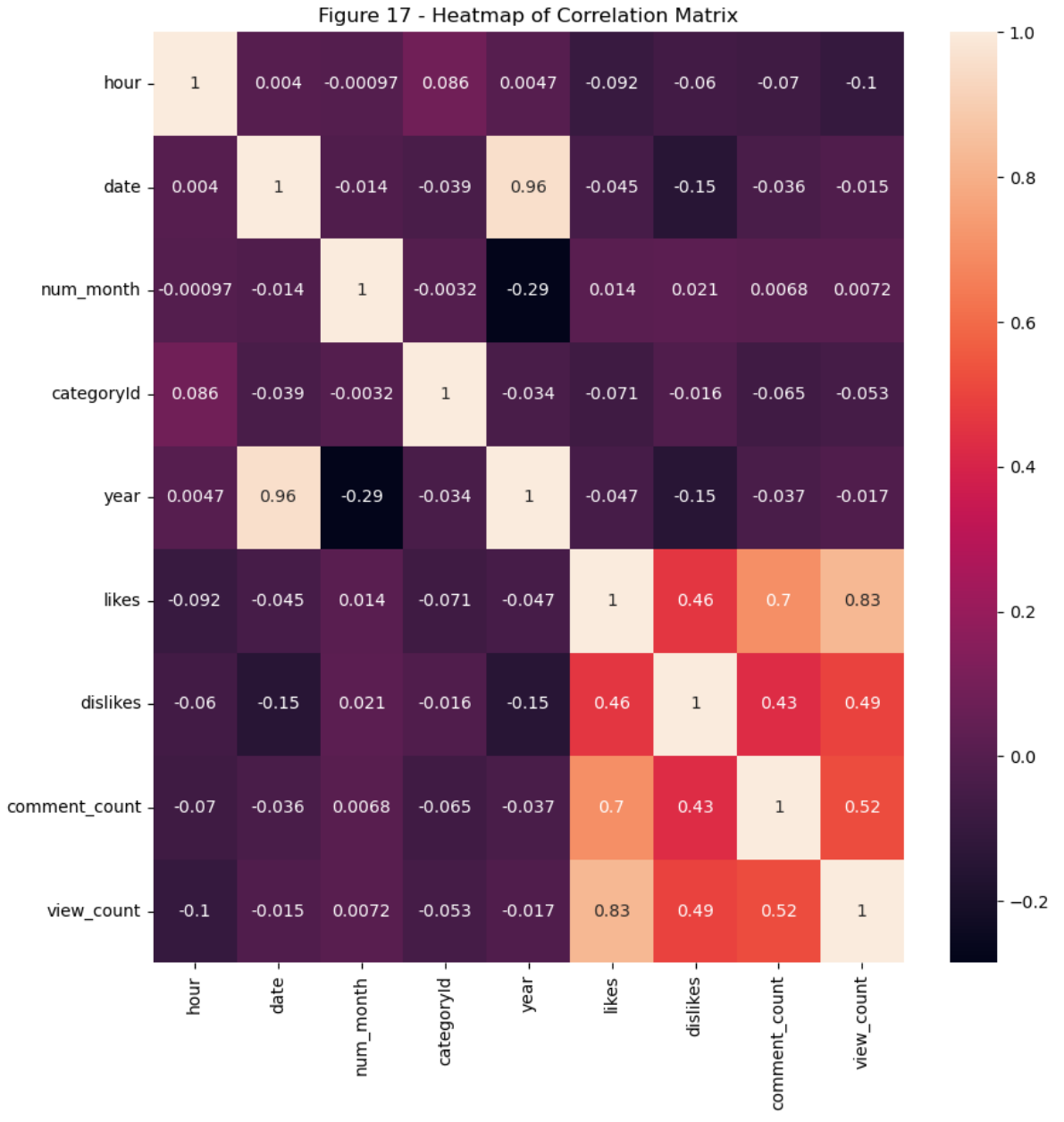
In [ ]:
```
corr_list = ['hour','date','num_month','categoryId','year','likes','dislikes
df[corr_list].corr()
```

Out[ ]:

	hour	date	num_month	categoryId	year	likes	di
hour	1.000000	0.003967	-0.000968	0.086224	0.004656	-0.092060	-0.0
date	0.003967	1.000000	-0.014301	-0.038648	0.957254	-0.044968	-0.1
num_month	-0.000968	-0.014301	1.000000	-0.003229	-0.285082	0.014485	0.0
categoryId	0.086224	-0.038648	-0.003229	1.000000	-0.034486	-0.070575	-0.0
year	0.004656	0.957254	-0.285082	-0.034486	1.000000	-0.047119	-0.1
likes	-0.092060	-0.044968	0.014485	-0.070575	-0.047119	1.000000	0.4
dislikes	-0.059514	-0.146859	0.021026	-0.016366	-0.147075	0.461104	1.0
comment_count	-0.070440	-0.036412	0.006841	-0.064965	-0.037127	0.699778	0.4
view_count	-0.104156	-0.015463	0.007157	-0.052603	-0.016800	0.829249	0.4

Now that we have the correlation, we will use seaborn's feature called `heatmap()` to better visualize the correlation among them.

In [ ]:
```
plt.figure(figsize=(10,10))
fig = sns.heatmap(data = df[corr_list].corr(), annot=True).set_title("Figure
```



From Figure 18, we see that the likes and view count is high-positively correlated with 0.83 value. The dislikes and view count, the comment count and view count, and the dislikes and comment count are also somewhat correlated. The rest of the variables are neglibly correlated as the value is closer to 0.

We can see a very high positive correlation between year and date, and further explore on it by comparing it with likes, comment count and view count. We will use the `lineplot` method to plot the graph.

In [ ]:
```
f, ax = plt.subplots(2, 3, figsize=(25, 20))
variables = ['likes', 'view_count', 'comment_count']
x_values = ['hour','month']
plt.suptitle('Figure 19 - Lineplot of Likes, View Count and Comment Count vs
plt.subplots_adjust(top=0.85)

for i, x in enumerate(x_values):
    for j, var in enumerate(variables):
        sns.lineplot(x = df[x], y = df[var], data = df, ax = ax[i,j], marke
```

Figure 19 - Lineplot of Likes, View Count and Comment Count vs Hour and Month

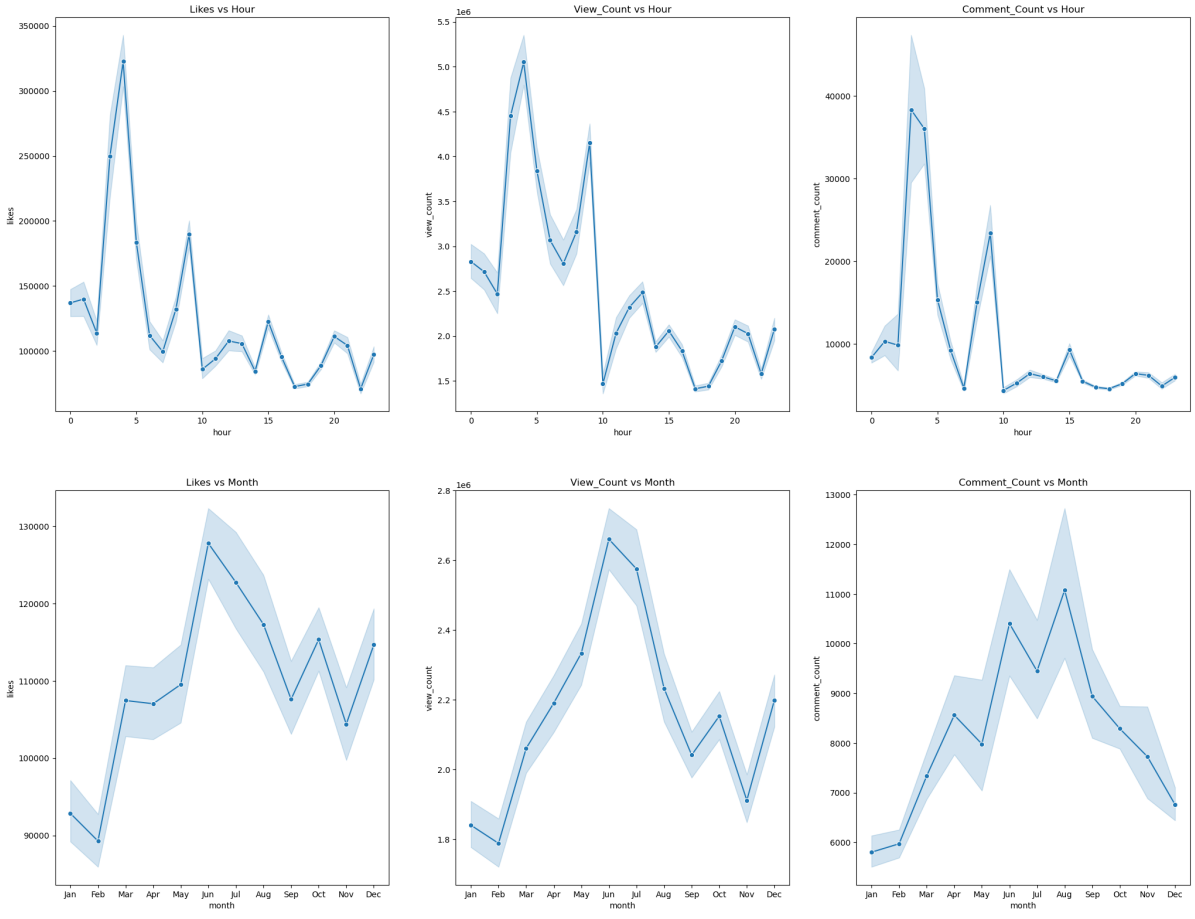


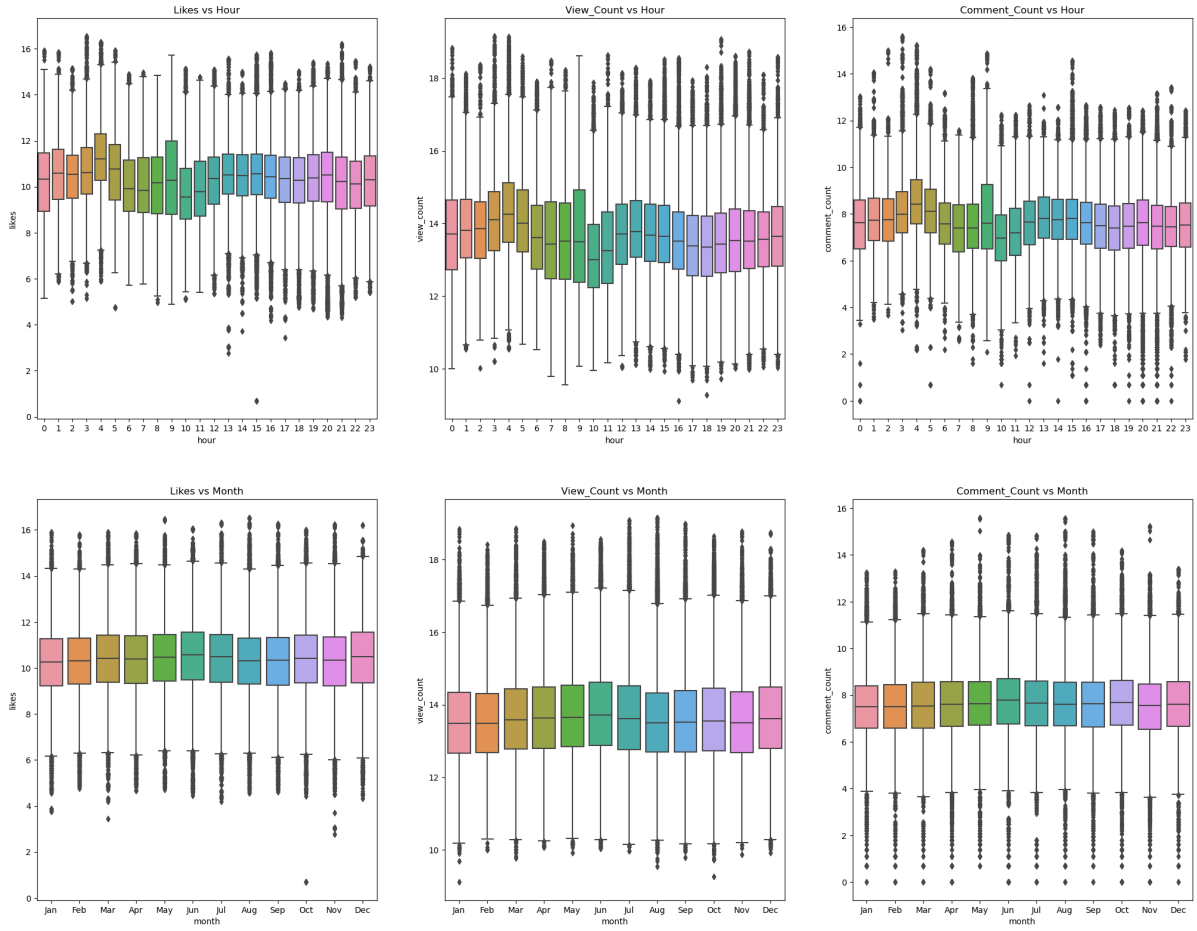
Figure 19 shows us the line plot for the number of likes, view count and comment count hourwise with respect to hours and month. From the figure 19, we see the majority of likes, view counts and comments are accrued for the videos pubished between 2 am and 7 am. We can also see the same trend for the videos published in the months between May and September.

Although we have a time frame to see a spike of the video, we will now use the `boxplot()` to better visualize the spike hourly and monthly.

In [ ]:
```
f, ax = plt.subplots(2, 3, figsize=(25, 20))
variables = ['likes', 'view_count', 'comment_count']
x_values = ['hour','month']
plt.suptitle('Figure 20 - Boxplot of Likes, View Count and Comment Count', 
plt.subplots_adjust(top=0.85)

for i, x in enumerate(x_values):
    for j, var in enumerate(variables):
        sns.boxplot(x = df[x], y = np.log(df[var]), data = df, ax = ax[i,j]
```

```
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Figure 20 - Boxplot of Likes, View Count and Comment Count

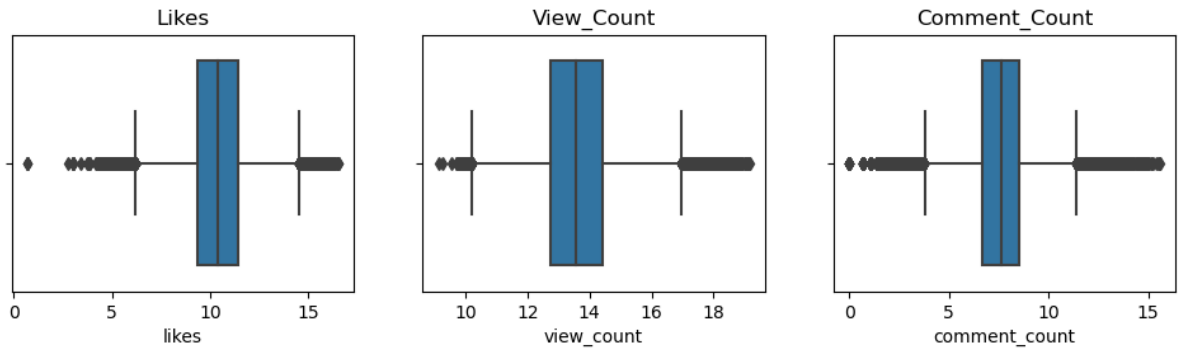


From figure 20, we can now see the exact hour and month better. Thus, as per the graph, 5am is the best time and June is the best month to publish the video.

In [ ]:
```
columns = ['likes', 'view_count', 'comment_count']
plt.figure(figsize=(16, 6))
plt.suptitle('Figure 21 - Boxplot of Likes, View Count and Comment Count', t
plt.subplots_adjust(top=0.85)
for i, column in enumerate(columns):
    plt.subplot(2, 4, i + 1)
    sns.boxplot(x=np.log(df[column]))
    plt.title(column.title())
```

```
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/Applications/anaconda3/lib/python3.11/site-packages/pandas/core/arraylike.
py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Figure 21 - Boxplot of Likes, View Count and Comment Count



Tried plotting with the actual count of likes, view_count and comment_count, but the code cell couldnt be compiled. So, we went for the log of each values.

Importing Regression models and creating Test/Train Datasets.

To begin with our analysis, we import the sci-kit-learn Python library. Scikit-learn contains a variety of practical classes that can help us with regression modelling. We import train_test_split, LinearRegression and RandomForestRegressor for modelling. We also import mean_squared_error and r2_score to help us in evaluating the model.

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import MinMaxScaler
```

Encoding Categorical Variables and Scaling

Since our dataset contains categorical data like `categoryId`, `Month`, `Hour` and `year`, we use OneHotEncoding function to create dummy variables.

Similarly, our dataset contains numerical values in `likes`, `dislikes` and `comment_count` which have varying numerical scale. We can thus use `MinMaxScaler` from Scikit-Learn to scale our numerical columns. Scikit-Learn alos contains `StandardScaler`, but we are using `MinMaxScaler` since it scales the data from `[0,1]` and preserves the shape of the dataset.

Finally, we drop all unnecessary columns in our dataframe to reduce computation complexity and then create a 20:80 test/train datasets.

In [ ]:
```
df_oneHotEncode = df.copy()
#list of columns to be one hot encoded
ohe_columns = ['categoryId','num_month','hour','year']
#one hot encode the columns
df_oneHotEncode = pd.get_dummies(df_oneHotEncode, columns=ohe_columns)

# drop columns that are not needed - title, publishedAt, trending_date, date
df_oneHotEncode = df_oneHotEncode.drop(['title','publishedAt','trending_date

# scale the numerical columns
```

```
scaler = MinMaxScaler()
df_oneHotEncode[['likes', 'dislikes', 'comment_count']] = scaler.fit_transfo
df_oneHotEncode.head()

target = df_oneHotEncode.drop(['view_count'], axis=1)
feature = df_oneHotEncode['view_count']

X_train, X_test, y_train, y_test = train_test_split(target, feature, test_si
```

Creating and evaluating a LinearRegression model

Secondly, we instantiate a linear regression model from the Scikit-Learn library and assign it to a variable called model. The object's fit method is used to train and create a linear regression model using our training dataset X_train. Finally, using the created model, we use our testing dataset X_test to predict our view count.

In [ ]:
```
#fit the model
model = LinearRegression()
model.fit(X_train, y_train)

#predict the model
y_pred = model.predict(X_test)
```

Evaluating the model

We can then print the Mean Squared Error, R**2 and Root Mean Squared Error.

In [ ]:
```
#evaluate the model
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
print('Variance score: %.2f' % r2_score(y_test, y_pred))
print('Root Mean squared error: %.2f' % mean_squared_error(y_test, y_pred, s
```

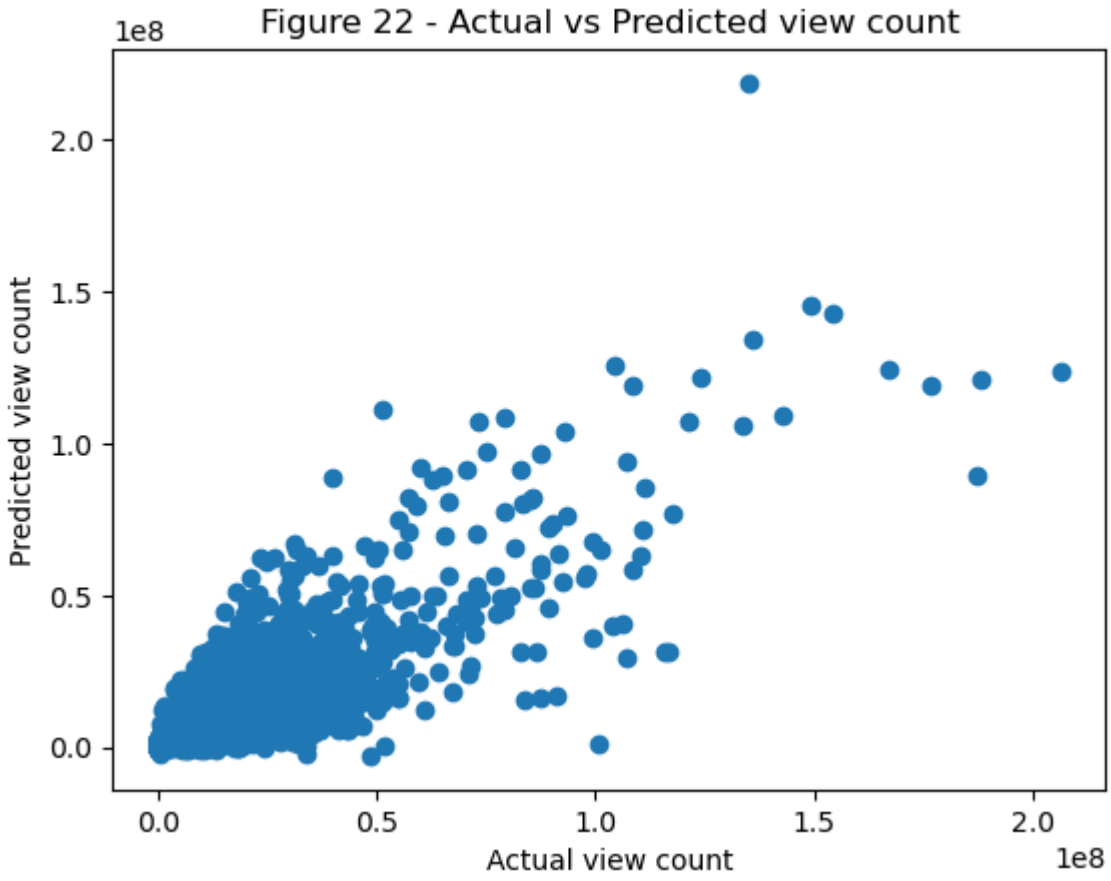
```
Mean squared error: 10033078753476.97
Variance score: 0.73
Root Mean squared error: 3167503.55
```

From the output above, we get a variance score of 0.72. While this score is close to 1, we can try another model to ensure we get the best possible score.

Plotting Predicted vs Actual View Count For Linear Regression

When we plot the graph for Linear Regression, we can observe that while Linear Regression can predict view counts for trending videos up to 1e8 views, it fails to accurately predict higher view counts, thus lowering our variance score.

In [ ]:
```
#plot the model
plt.scatter(y_test, y_pred)
plt.xlabel("Actual view count")
plt.ylabel("Predicted view count")
plt.title("Figure 22 - Actual vs Predicted view count")
plt.show()
```



Using Random Forest Regressor

We can now use Linear Regression as our baseline model to evaluate our next model. We are using RandomForestRegressor from Scikit-Learn to implement our model.

RandomForestRegressor needs an argument called n_estimtor that determines the number of decision tree classifiers to use. To check the appropriate integer for the n_estimator, we can create a function that takes an n as the number estimator. The function returns MSE, R2 and RMSE values.

In [ ]:
```
# def RandomForestList(n):
#     model = RandomForestRegressor(n_estimators=n, random_state=201750985)
#     model.fit(X_train, y_train.values.ravel())
#     y_pred = model.predict(X_test)
#     return mean_squared_error(y_test, y_pred), r2_score(y_test, y_pred), r
```

We can then pass the function through a for loop that iterates over the list of estimators as defined below. We can then append the resultant scores against the n_estimators to a new data frame.

In [ ]:
```
# rf_df = pd.DataFrame(columns=['n_estimators', 'MSE', 'RMSE', 'R2'])
# n_estimates = [1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50,55 ,60 ,65 ,70, 75
# for i in n_estimates:
#     mse,r2,rmse = RandomForestList(i)
#     rf_df = pd.concat([rf_df, pd.DataFrame([{'n_estimators': i, 'MSE': mse
#     #rf_df = rf_df.append(, ignore_index=True)
```

In [ ]:
```
# rf_df.head(100)
```

Selection of Estimator value

We can now plot RMSE vs n_estimator and R2 vs n_estimator. From the graph, we see that an increase in n_estimnator initially causes a rapid rise in R2 value, but only a marginal increase from n_estimator greater than 15. Furthermore, any increase in the value of n_estimnator does not cause any change in the value of R2. Thus, we decide that the value of n_estimator should be 50 to maximise our R2 while also minimising compute time.

In [ ]:
```
# fig, axes = plt.subplots(1, 2, figsize=(15, 5))
# axes[0].plot(rf_df['n_estimators'], rf_df['RMSE'])
# axes[0].set_xlabel('n_estimators')
# axes[0].set_ylabel('RMSE')
# axes[0].set_title('Figure 9 - RMSE vs n_estimators')
# axes[0].axvline(x=50, color='r', linestyle='--')
# axes[1].plot(rf_df['n_estimators'], rf_df['R2'], color='orange')
# axes[1].set_xlabel('n_estimators')
# axes[1].set_ylabel('R2')
# axes[1].set_title('Figure 10 - R2 vs n_estimators')
# axes[1].axvline(x=50, color='r', linestyle='--')
# plt.show()
```

Training RandomForest Modedel using Selected n_estimator

After determining 50 as the ideal value of n_estimator, we can start training our Random Forest Model. AFter training with X_train and y_train, we can test our model with the X_test dataset.

In [ ]:
```
# # create regressor object
# regressor = RandomForestRegressor(n_estimators = 50, random_state = 201750
# regressor.fit(X_train, y_train.values.ravel())
# # predict the result
# y_pred = regressor.predict(X_test)
```

Evaluating the Model

Printing the scores for our model shows a significant increase in the R2 value from 0.72 to 0.95. Thus, we can choose the RandomForestRegressor with n_estimator as 50 for our prediction Model.

In [ ]:
```
# print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
# print('Variance score: %.2f' % r2_score(y_test, y_pred))
# print('Root Mean squared error: %.2f' % mean_squared_error(y_test, y_pred,
```

Plotting the graph of actual vs prediction shows our model has improved over just Linear Regression in determining View_count of Videos given a set of attributes about the video.

In [ ]:
```
# #plot the model
# plt.scatter(y_test, y_pred)
# plt.xlabel("Actual view count")
# plt.ylabel("Predicted view count")
```

```
# plt.title("Actual vs Predicted view count")
# plt.show()
```

Project Outcome (10 + 10 marks)

This section should describe the outcome of the project by means of both explanation of the results and by graphical visualisation in the form of graphs, charts or or other kinds of diagram

The section should begin with a general overview of the results and then have a section for each of the project objectives. For each of these objectives an explanation of more specific results relating to that objective shoud be given, followed by a section presenting some visualisation of the results obtained. (In the case where the project had just one objective, you should still have a section describing the results from a general perspective followed by a section that focuses on the particular objective.)

The marks for this section will be divided into 10 marks for Explanation and 10 marks for Visualisation. These marks will be awarded for the Project Outcome section as a whole, not for each objective individually. Hence, you do not have to pay equal attention to each. However, you are expected to have a some explanation and visualisation for each. It is suggested you have 200-400 words explanation for each objective.

Overview of Results

Give a general overview of the results (around 200 words).

Objective 1

We can deduce two conclusions:

1. In 2020, the number of bins in the plot is comparatively smaller than that of other bins. This is mainly because we only have the data for the last five months of 2020 (i.e., August to December 2020).
2. We also notice a change in trend across 2021 to 2023, where the videos in the gaming category are more trending, while the Entertainment, and People and blog categories are decreasing year by year. Sports and Music categories stay in almost similar trends.

Objective 2

Checks the key words to be used in the Title and/or Description for a video to be in the trending category

Now, we will probe into investigating the keywords used in the title, description and how frequent the keywords are among the list of trending categories in each category. We

first created a new variable 'title_length' to store the number of words in each title. As we can see in Figure 4, most of the trending videos have 8 words in the title, and there are almost negligible trending videos that have over 20 words.

We then studied into finding actual keywords that are significant in the title and Video tags. We used the word cloud to plot the keywords as Figure 5 for the title and Figure 6 for the video tags respectively, for each category. In Figure 5, we can see the ones that are dark and big are more used keywords and of the highest significance. Thus, in the Entertainment category, keywords like 'Official', and 'Trailer' are used more, while 'Slow Mo' is of lesser significance. Similarly in Figure 6, 'Manchester United' has the highest significance in Entertainment category.

The same goes for the rest of the categories, where 'Official Video' tops the list across a few categories, followed by the local keywords such as 'Prime Minister' for News & Politics, 'League Highlights' for Sports, 'iPhone Pro' for Science & Technology, etc. in regards to the keywords used in the title.

Now that we observed the keywords used, we then generated a word cloud that has the common keywords between the title and the video tags. From Figure 7, we spot that "Season" is the most commonly used keyword in the title as well as in the video tags in the Entertainment category.

Based on the observations made, we can deduce the following key points,

1. The audience is more intrigued to watch videos that have keywords relating to current affairs, in almost every category.
2. The keywords used in the title are also used in the video tags, with which one could assume that the frequency has an impact on the reach of the videos.

Objective 3

Objective 4

Conclusion (5 marks)

Your concluding section should be around 200-400 words. It is recommended that you divide it into the following sections.

Achievements

As we had expected, the most popular fridge magnets were of the 'meme' kind. We were surprised that 'smiley' fridge magnets were less common than expected. We conjecture that this is because, although they are apparently very popular, few fridges display more than one smiley. However, 'meme' based magnets can be found in large numbers, even on quite small fridges.

Limitations

The limitations include the following,

1. Insufficient Data: As we only have data from August 2020 to November 2023, we can only see the trend for the recent years. If we had enough data for atleast 5 years, we could have better predicted trend and accuracy.
2. Dislikes from 2022: Youtube has stopped publishing the dislikes count since 2022. If we had dislikes count for our entire dataset, we would have then investigated to see if the controversy have any imoact on the trending of the videos.
3. Time Complexity: The functions like randomforest takes a lot of time to compute. In order to enhance this problem, we can use VPC/VPS for better computational capability to further make more complex models.

Modelling RandomFOirest for large scale datasets like the Yotube APi , mention O notation for models

Future Work

For future works, we can enhance the project by further probing into the following,

1. Adding More Countries
2. Make it more user configurable
3.