YT Data Analysis Final

August 5, 2021

[1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
import sqlite3
import json
from datetime import datetime
```

1 YouTube Analysis - YouTube Trending Across the World

Since its inception in 2005, YouTube has grown to be a household name and one of the biggest successes of the social media and tech age. Each year, YouTube collects statistics on the videos on its platform in order to create lists of the top trending videos. To do this, they collect a number of statistics about each video

The data used in this analysis is a daily record of the top trending videos in ten different countries over six months:

- Canada (CA)
- Germany (DE)
- France (FR)
- United Kingdom (GB)
- India (IN)
- Japan (JP)
- South Korea (KR)
- Mexico (MX)
- Russia (RU)
- USA (US)

The main data for each country is stored in separate `CSV` files. Each country also has an associated `JSON` file, which contains information scraped from YouTube about the category of each video and its associated category number.

1.1 What can we find out from the data?

What are people in each of these countries using YouTube for? There are many categories of video on YouTube, so what categories are more likely to produce trending videos in each country? There might be some interesting insights into YouTube viewing/interaction habits across the world.

To paint a picture of the overall data and then viewing habits and user interaction between countries, the following questions will be answered:

1. How many videos are in each dataset? This will be the number of records in each dataset.
2. How many video views are there for each country?
3. How many likes/dislikes are there for the videos for each country?
4. What is the like to dislike ratio for each country?
5. For each country, what is the percentage of video views that result in a like?
6. For each country, how are these percentages distributed?
7. What are the categories of videos in the datasets? Are the categories the same for all of the countries?
8. What is the distribution of views amongst the different categories for each country?

1.2 Implementing ML - predicting video category

1. Can we accurately predict which category a video will belong to based on some features of the data? And will this translate across the different countries?
2. If not, what is likely the cause of the poor classification?

2 Findings

2.1 Data Statistics

- The datasets span the 6 month period between 01/12/17 to 31/05/18, except for Japan which starts at 01/03/18
- There are similar numbers of records (~40,000) for each country, except Japan, which has ~20,000
- Trending videos in the UK account for approximately 46% of the total video views from all countries combined. It is not clear whether this is an anomaly due to data acquisition or if it is something more significant. To contrast - the USA accounts for only around 19% of the total views
- There are also significantly more likes & dislikes for videos in the UK dataset, with almost double the USA, the country with the second most likes & dislikes
- Russia has by far the lowest like to dislike ratio, with some other countries having a ratio which is almost three times higher
- Despite this, Russia actually has the highest percentage of views resulting in a like for all the countries. So it seems that Russian viewers are more liberal with giving out both likes and dislikes
- In contrast, the UK has the lowest percentage of views that result in a like

2.2 Trending Categories

If you have a YouTube channel and you want your video to become trending, **YOU SHOULD** create a video which belongs in one of these categories:

- *'Film & Animation'*
- *'Autos & Vehicles'*
- *'Music'*
- *'Pets & Animals'*

- *'Sports'*
- *'Travel & Events'*
- *'Gaming'*
- 'People & Blogs'*
- *'Comedy'*
- *'Entertainment'*
- *'News & Politics'*
- *'Howto & Style'*
- *'Education'*
- *'Science & Technology'*
- *'Nonprofits & Activism'*
- *'Movies'*
- *'Shows'*
- *'Trailers'*

Similarly, **YOU SHOULDN'T** create a video which belongs in one of these categories:

- *'Short Movies'*
- *'Videoblogging'*
- *'Anime/Animation'*
- *'Action/Adventure'*
- *'Classics'*
- *'Documentary'*
- *'Drama'*
- *'Family'*
- *'Foreign'*
- *'Horror'*
- *'Sci-Fi/Fantasy'*
- *'Thriller'*
- *'Shorts'*

The top 5 most common categories for videos that become trending are (in descending order):

1. Entertainment
2. People and Blogs
3. Music
4. News and Politics
5. Comedy

2.3 Machine Learning

The chosen prediction task was not well suited to the data using a Random Forest with random search cross-validation. The number of classes (18) and the small number of features suitable (5) to try to define each class is insufficient; the data points are clustered together with inadequate separation.

Projecting the 5-D data onto the top 2 and 3 eigenvectors allows for the visualisation of this point. It can be seen that the data are clustered together and so cannot be accurately separated.

3 Data Ingestion - csv files

The csv files will be loaded in first, with the JSON files being loaded in further in the notebook.

Some of the CSV files have different encoding due to the different languages present in the datasets. The following cell was used to determine the encoding:

[2]:
```
# creating a list containing the abbreviations for each country in the datasets␣
↪- will be used throughout
countries_list = ['CA', 'DE', 'FR', 'GB', 'IN', 'JP', 'KR', 'MX', 'RU', 'US']

# chardet detects the type of encoding used in files
import chardet

countries_encoding = {}
for i in countries_list:
    with open(i+'videos.csv', 'rb') as rawdata:
        result = chardet.detect(rawdata.read(100000))
        countries_encoding[i] = 'Encoding is ' + str(result['encoding'])

countries_encoding
```

[2]:
```
{'CA': 'Encoding is Windows-1254',
 'DE': 'Encoding is Windows-1254',
 'FR': 'Encoding is Windows-1254',
 'GB': 'Encoding is Windows-1254',
 'IN': 'Encoding is Windows-1254',
 'JP': 'Encoding is None',
 'KR': 'Encoding is None',
 'MX': 'Encoding is Windows-1254',
 'RU': 'Encoding is utf-8',
 'US': 'Encoding is Windows-1254'}
```

This created an issue when trying to load in the data with a script. Instead, the data for each country will be loaded manually to allow for different encoding parameters in the `pd.read_csv` method. We could use `latin1` encoding for all of the `CSV`s, but this affects the legibility of some of the columns in the Western European language `CSV`s

[3]:
```
# load all of the CSV files into DataFrames

canada = pd.read_csv('CAvideos.csv')
germany = pd.read_csv('DEvideos.csv')
france = pd.read_csv('FRvideos.csv')
uk = pd.read_csv('GBvideos.csv')
india = pd.read_csv('INvideos.csv')
japan = pd.read_csv('JPvideos.csv', encoding="latin1")
korea = pd.read_csv('KRvideos.csv', encoding="latin1")
mexico = pd.read_csv('MXvideos.csv', encoding="latin1")
```

```
russia = pd.read_csv('RUvideos.csv', encoding="latin1")
usa = pd.read_csv('USvideos.csv')
```

We should see if the tables match and investigate the features.

[4]:
```
# check to see that the column numbers match
# this will also tell us how many records exist in each DataFrame
# each record is a video, and so this tells us how many videos are present in␣
↪each dataset

canada.shape, germany.shape, france.shape, uk.shape, india.shape, japan.shape,␣
↪korea.shape, mexico.shape, russia.shape, usa.shape
```

[4]:
```
((40881, 16),
 (40840, 16),
 (40724, 16),
 (38916, 16),
 (37352, 16),
 (20523, 16),
 (34567, 16),
 (40451, 16),
 (40739, 16),
 (40949, 16))
```

[5]:
```
# check to ensure that the data have the same columns (done one at a time)
canada.columns == germany.columns
canada.columns == france.columns
canada.columns == uk.columns
canada.columns == india.columns
canada.columns == japan.columns
canada.columns == korea.columns
canada.columns == mexico.columns
canada.columns == russia.columns
canada.columns == usa.columns
```

[5]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True])
```

[6]:
```
# check the top of the DataFrame
canada.head()
```

[6]:
```
     video_id trending_date  \
0  n1WpP7iowLc      17.14.11
1  0dBIkQ4Mz1M      17.14.11
2  5qpjK5DgCt4      17.14.11
3  d380meD0W0M      17.14.11
4  2Vv-BfVoq4g      17.14.11
```

```
                                                title channel_title  \
0         Eminem - Walk On Water (Audio) ft. Beyoncé    EminemVEVO
1                      PLUSH - Bad Unboxing Fan Mail     iDubbbzTV
2  Racist Superman | Rudy Mancuso, King Bach & Le…  Rudy Mancuso
3                           I Dare You: GOING BALD!?      nigahiga
4        Ed Sheeran - Perfect (Official Music Video)    Ed Sheeran

   category_id              publish_time  \
0           10  2017-11-10T17:00:03.000Z
1           23  2017-11-13T17:00:00.000Z
2           23  2017-11-12T19:05:24.000Z
3           24  2017-11-12T18:01:41.000Z
4           10  2017-11-09T11:04:14.000Z

                                                tags     views    likes  \
0  Eminem|"Walk"|"On"|"Water"|"Aftermath/Shady/In…  17158579   787425
1  plush|"bad unboxing"|"unboxing"|"fan mail"|"id…   1014651   127794
2  racist superman|"rudy"|"mancuso"|"king"|"bach"…   3191434   146035
3  ryan|"higa"|"higatv"|"nigahiga"|"i dare you"|"…   2095828   132239
4  edsheeran|"ed sheeran"|"acoustic"|"live"|"cove…  33523622  1634130

   dislikes  comment_count                                  thumbnail_link  \
0     43420         125882  https://i.ytimg.com/vi/n1WpP7iowLc/default.jpg
1      1688          13030  https://i.ytimg.com/vi/0dBIkQ4Mz1M/default.jpg
2      5339           8181  https://i.ytimg.com/vi/5qpjK5DgCt4/default.jpg
3      1989          17518  https://i.ytimg.com/vi/d380meD0W0M/default.jpg
4     21082          85067  https://i.ytimg.com/vi/2Vv-BfVoq4g/default.jpg

   comments_disabled  ratings_disabled  video_error_or_removed  \
0              False             False                   False
1              False             False                   False
2              False             False                   False
3              False             False                   False
4              False             False                   False

                                         description
0  Eminem's new track Walk on Water ft. Beyoncé i…
1  STill got a lot of packages. Probably will las…
2  WATCH MY PREVIOUS VIDEO  \n\nSUBSCRIBE  http…
3  I know it's been a while since we did this sho…
4   : https://ad.gt/yt-perfect\n : https://atlant…
```

[7]:
```python
# checking the bottom of the DataFrame
canada.tail()
```

[7]:
```
          video_id trending_date  \
40876  sGolxsMSGfQ      18.14.06
40877  8HNuRNi8t70      18.14.06
40878  GWlKEM3m2EE      18.14.06
40879  lbMKLzQ4cNQ      18.14.06
40880  POTgw38-m58      18.14.06

                                                    title     channel_title  \
40876                          HOW2: How to Solve a Mystery    Annoying Orange
40877                      Eli Lik Lik Episode 13 Partie 01  Elhiwar Ettounsi
40878  KINGDOM HEARTS III – SQUARE ENIX E3 SHOWCASE 2…   Kingdom Hearts
40879                      Trump Advisor Grovels To Trudeau   The Young Turks
40880                                   2018.06.13

       category_id              publish_time  \
40876           24  2018-06-13T18:00:07.000Z
40877           24  2018-06-13T19:01:18.000Z
40878           20  2018-06-11T17:30:53.000Z
40879           25  2018-06-13T04:00:05.000Z
40880           24  2018-06-13T16:00:03.000Z

                                                     tags   views  likes  \
40876  annoying orange|"funny"|"fruit"|"talking"|"ani…   80685   1701
40877  hkayet tounsia|"elhiwar ettounsi"|"denya okhra…  103339    460
40878  Kingdom Hearts|"KH3"|"Kingdom Hearts 3"|"Froze…  773347  25900
40879  180612__TB02SorryExcuse|"News"|"Politics"|"The…  115225   2115
40880    |"  "|"    "|"Sandy"|"Jacky wu"|"  "|" …  107392    300

       dislikes  comment_count  \
40876        99           1312
40877        66             51
40878       224           3881
40879       182           1672
40880        62            251

                                       thumbnail_link  comments_disabled  \
40876  https://i.ytimg.com/vi/sGolxsMSGfQ/default.jpg              False
40877  https://i.ytimg.com/vi/8HNuRNi8t70/default.jpg              False
40878  https://i.ytimg.com/vi/GWlKEM3m2EE/default.jpg              False
40879  https://i.ytimg.com/vi/lbMKLzQ4cNQ/default.jpg              False
40880  https://i.ytimg.com/vi/POTgw38-m58/default.jpg              False

       ratings_disabled  video_error_or_removed  \
40876             False                   False
40877             False                   False
40878             False                   False
40879             False                   False
```

```
40880                False                        False

                                                      description
40876    NEW MERCH! http://amzn.to/annoyingorange    …
40877    Retrouvez vos programmes préférés : https://…
40878   Find out more about Kingdom Hearts 3: https://…
40879   Peter Navarro isn’t talking so tough now. Ana …
40880          LaLa       ( ) Wendy(  ) …
```

3.1 Timespan of data

Before digging into the data, it will be useful to know exactly over which time the data were collected for each country. If there is a large discrepancy between datasets then it might skew results in the analysis.

To do this we will use python’s `datetime.strptime` method to find the `max_date` and `min_date` for each country, and then create a list that will contain the `date_ranges` for each country.

[8]:
```python
# creating a list containing the DataFrames of each country - will be useful␣
↪going forward
countries_df_list=[canada,
                   germany,
                   france,
                   uk,
                   india,
                   japan,
                   korea,
                   mexico,
                   russia,
                   usa]
```

[9]:
```python
# initialise the lists that will contain the relevant date data for each country
max_dates = []
min_dates = []
date_ranges = []

# loop through each country's DataFrame and append to the lists
for country in countries_df_list:
    max_date = datetime.strptime(country['trending_date'].max(), "%y.%d.%m")
    max_dates.append(max_date)
    min_date = datetime.strptime(country['trending_date'].min(), "%y.%d.%m")
    min_dates.append(min_date)
    date_ranges.append(max_date - min_date)

for i in range(len(countries_list)):
    print(countries_list[i]+" latest date: "+str(max_dates[i]))
    print(countries_list[i]+" earliest date: "+str(min_dates[i]))
    print(countries_list[i]+" number of days in dataset: "+str(date_ranges[i]))
```

```
    print("-"*45)
```

```
CA latest date: 2018-05-31 00:00:00
CA earliest date: 2017-12-01 00:00:00
CA number of days in dataset: 181 days, 0:00:00
---------------------------------------------
DE latest date: 2018-05-31 00:00:00
DE earliest date: 2017-12-01 00:00:00
DE number of days in dataset: 181 days, 0:00:00
---------------------------------------------
FR latest date: 2018-05-31 00:00:00
FR earliest date: 2017-12-01 00:00:00
FR number of days in dataset: 181 days, 0:00:00
---------------------------------------------
GB latest date: 2018-05-31 00:00:00
GB earliest date: 2017-12-01 00:00:00
GB number of days in dataset: 181 days, 0:00:00
---------------------------------------------
IN latest date: 2018-05-31 00:00:00
IN earliest date: 2017-12-01 00:00:00
IN number of days in dataset: 181 days, 0:00:00
---------------------------------------------
JP latest date: 2018-05-31 00:00:00
JP earliest date: 2018-03-01 00:00:00
JP number of days in dataset: 91 days, 0:00:00
---------------------------------------------
KR latest date: 2018-05-31 00:00:00
KR earliest date: 2017-12-01 00:00:00
KR number of days in dataset: 181 days, 0:00:00
---------------------------------------------
MX latest date: 2018-05-31 00:00:00
MX earliest date: 2017-12-01 00:00:00
MX number of days in dataset: 181 days, 0:00:00
---------------------------------------------
RU latest date: 2018-05-31 00:00:00
RU earliest date: 2017-12-01 00:00:00
RU number of days in dataset: 181 days, 0:00:00
---------------------------------------------
US latest date: 2018-05-31 00:00:00
US earliest date: 2017-12-01 00:00:00
US number of days in dataset: 181 days, 0:00:00
---------------------------------------------
```

[10]:
```
max_dates
```

[10]:
```
[datetime.datetime(2018, 5, 31, 0, 0),
 datetime.datetime(2018, 5, 31, 0, 0),
```

```
 datetime.datetime(2018, 5, 31, 0, 0),
 datetime.datetime(2018, 5, 31, 0, 0),
 datetime.datetime(2018, 5, 31, 0, 0),
 datetime.datetime(2018, 5, 31, 0, 0),
 datetime.datetime(2018, 5, 31, 0, 0),
 datetime.datetime(2018, 5, 31, 0, 0),
 datetime.datetime(2018, 5, 31, 0, 0),
 datetime.datetime(2018, 5, 31, 0, 0)]
```

[11]:
```
min_dates
```

[11]:
```
[datetime.datetime(2017, 12, 1, 0, 0),
 datetime.datetime(2017, 12, 1, 0, 0),
 datetime.datetime(2017, 12, 1, 0, 0),
 datetime.datetime(2017, 12, 1, 0, 0),
 datetime.datetime(2017, 12, 1, 0, 0),
 datetime.datetime(2018, 3, 1, 0, 0),
 datetime.datetime(2017, 12, 1, 0, 0),
 datetime.datetime(2017, 12, 1, 0, 0),
 datetime.datetime(2017, 12, 1, 0, 0),
 datetime.datetime(2017, 12, 1, 0, 0)]
```

All of the datasets contain 181 days of trending YouTube video statistics, except Japan which contains only 91 days as the data begins three months after the other countries.

So of course this will impact the number of records in the `japan` data, but there is still a significant amount of data to be worked with.

Bearing this in mind, we will begin to investigate the statistics of each dataset.

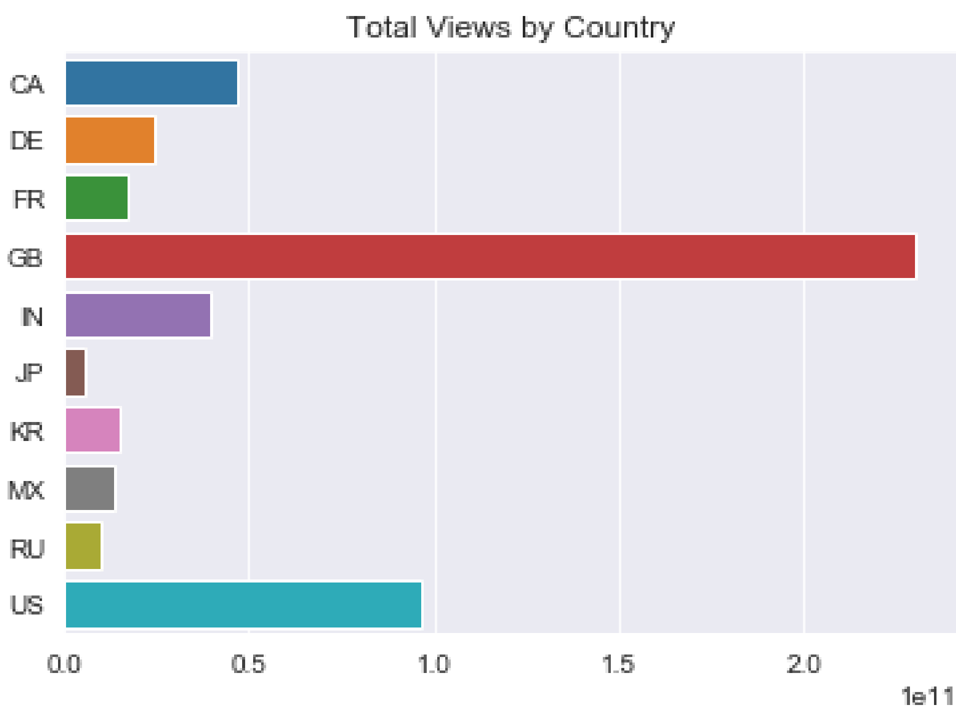
3.2 YouTube video views by country

[88]:
```
# finding the total views in each country

total_views = sum(country_total_views.values())
country_total_views = {}
country_proportion_views = {}
for country in range(len(countries_df_list)):
    view_count = countries_df_list[country]['views'].sum()
    country_total_views[countries_list[country]] = view_count
    country_proportion_views[countries_list[country]] = round(((view_count/
↪total_views)*100), 2)

country_total_views, country_proportion_views
```

[88]:
```
({'CA': 46891975069,
  'DE': 24645115205,
  'FR': 17100897444,
```

```
  'GB': 230069198174,
  'IN': 39610961029,
  'JP': 5377466630,
  'KR': 14689152313,
  'MX': 13849692994,
  'RU': 9806494525,
  'US': 96671770152},
 {'CA': 9.4,
  'DE': 4.94,
  'FR': 3.43,
  'GB': 46.13,
  'IN': 7.94,
  'JP': 1.08,
  'KR': 2.95,
  'MX': 2.78,
  'RU': 1.97,
  'US': 19.38})
```

[13]:
```
# plotting the above result with Seaborn
sns.set_style('darkgrid')

sns.barplot(x=list(country_total_views.values()), y=countries_list).
↪set_title('Total Views by Country')
plt.savefig('ViewsByCountry.png')
```



3.2.1 One thing of note

So, already we can see that the UK has far more views in the data collected between 2017-12-01 and 2018-05-31. It's not clear at this point whether the UK generally has far more views for trending videos, if there is some discrepancy in the data collection process, or if this is due to some other unforeseen reason.

For a country of only around 65 million people, it would be expected that the UK would not account for such a large proportion of the views.

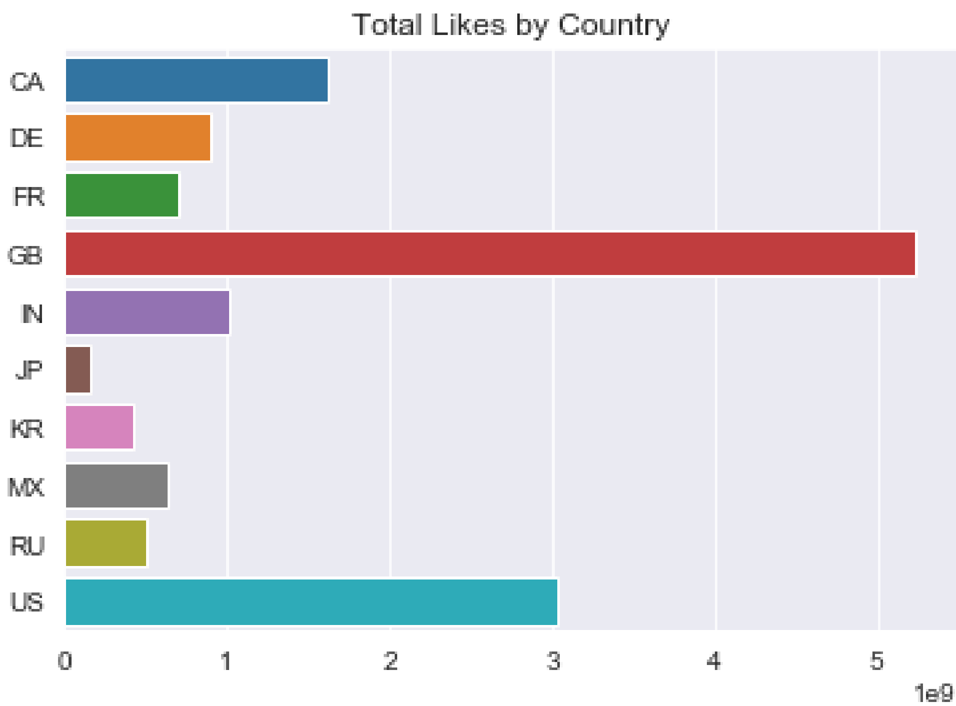
3.3 Sum of likes and dislikes across each country

3.3.1 Total likes by country

[14]:
```python
# now doing the same with likes
country_total_likes = {}
for country in range(len(countries_df_list)):
    like_count = countries_df_list[country]['likes'].sum()
    country_total_likes[countries_list[country]] = like_count

country_total_likes
```

[14]:
```
{'CA': 1618179878,
 'DE': 893395538,
 'FR': 708144090,
 'GB': 5234962944,
 'IN': 1011593670,
 'JP': 165406898,
 'KR': 421247912,
 'MX': 641627186,
 'RU': 506598491,
 'US': 3041147198}
```

[15]:
```python
sns.barplot(x=list(country_total_likes.values()), y=countries_list).
 ↪set_title('Total Likes by Country')
plt.savefig('LikesByCountry.png')
```



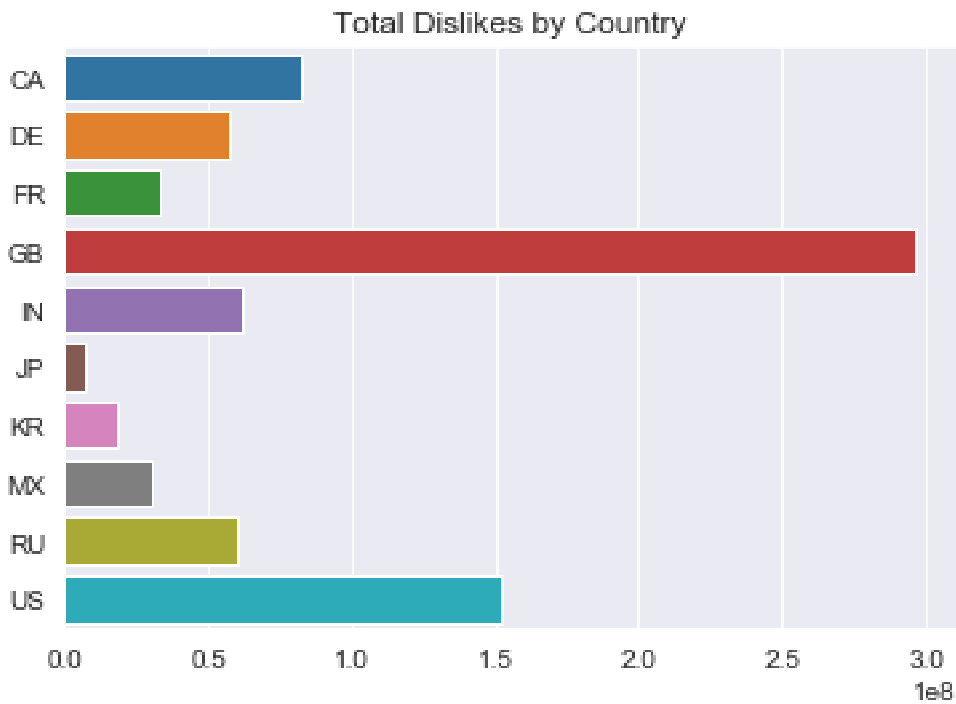
3.3.2 Total Dislikes by country

[16]:
```
# and doing the same again for dislikes
country_total_dislikes = {}
for country in range(len(countries_df_list)):
    like_count = countries_df_list[country]['dislikes'].sum()
    country_total_dislikes[countries_list[country]] = like_count

country_total_dislikes
```

[16]:
```
{'CA': 82137919,
 'DE': 57059031,
 'FR': 33188528,
 'GB': 296250384,
 'IN': 62194142,
 'JP': 7528321,
 'KR': 18634999,
 'MX': 30223385,
 'RU': 60098157,
 'US': 151978155}
```

[17]:
```
sns.barplot(x=list(country_total_dislikes.values()), y=countries_list).
 ↪set_title('Total Dislikes by Country')
```

```
plt.savefig('DislikesByCountry.png')
```



[18]:
```
# make a new directory to contain these figures (and any future figures)
#!mkdir figures
```

[19]:
```
# move the figures to the directory
!move *.png figures
```

```
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\DislikesByCountry.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\LikesByCountry.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\ViewsByCountry.png
        3 file(s) moved.
```

3.4 Ratio of `likes/dislikes` and `likes/views` by country

To deepen the understanding of user-video interaction between countries, it would be insightful to see how the **ratio** of likes to dislikes varies between countries.

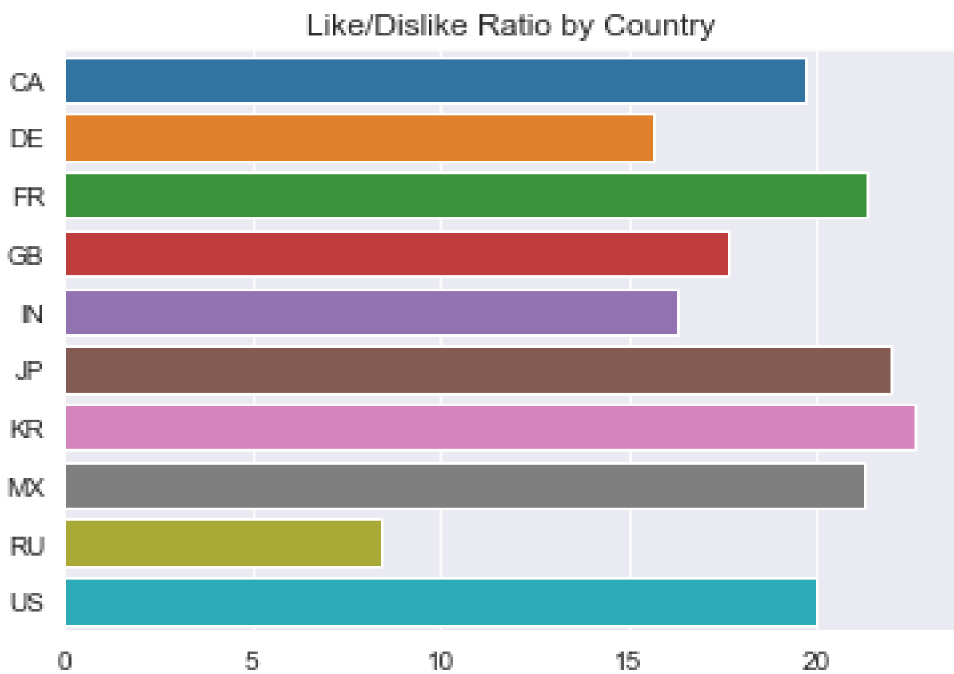
Some videos have `TRUE` in the `ratings_disabled` column - meaning these will have zero likes and zero dislikes. However, for the time being we will use the entire dataset for each country to get a picture across the whole dataset.

3.4.1 Like/Dislike ratio for each country

[20]:
```
# finding the ratio of likes to dislikes on YouTube videos for each country
# adding a column to each country DataFrame with the like/dislike ratio for␣
↪each video

like_dislike_ratio_list = []
for country in countries_df_list:
    ratio = country['likes'].sum() / country['dislikes'].sum()
    ratio = np.round(ratio, decimals=2)
    like_dislike_ratio_list.append(ratio)
    country['like/dislike ratio'] = np.round(country['likes'] /␣
↪country['dislikes'], decimals=2)

like_dislike_ratio_list
```

[20]:
```
[19.7, 15.66, 21.34, 17.67, 16.27, 21.97, 22.61, 21.23, 8.43, 20.01]
```

[21]:
```
sns.barplot(x=like_dislike_ratio_list, y=countries_list).set_title('Like/
↪Dislike Ratio by Country')
plt.savefig('LikeDislikeRatioByCountry.png')
```



From this it looks like the Russians aren't too keen on giving out likes! Maybe this isn't representative though, as it doesn't take the **number of likes per view** into account. It could also mean that Russians give out more dislikes than the other countries as a proportion of total likes/likes.

To get a better idea of how likely people are to like a video in each country, it is a good idea to find the percentage of views that result in a like (see `1.5.2` further in the notebook)

3.4.2 Percentage of views resulting in a like for each country

To continue deepening our understanding of how viewers are interacting with videos in each country, it is useful to see the rate at which videos are liked in each country. We can do this by calculating the percentage of views that result in a like.

[22]:
```
# now getting the percentage of views that result in a like in each country
# also adding a new column to each DataFrame that will have the %views/like␣
↪ratio (rounded to 2 decimal places)

likes_per_view = []
for country in countries_df_list:
    ratio_pct = (country['likes'].sum() / country['views'].sum())*100
    ratio_pct = np.round(ratio_pct, decimals=2)
    likes_per_view.append(ratio_pct)
    country['%_Views Resulting in Like'] = np.round((country['likes']/
↪country['views'])*100, decimals=2)

likes_per_view
```

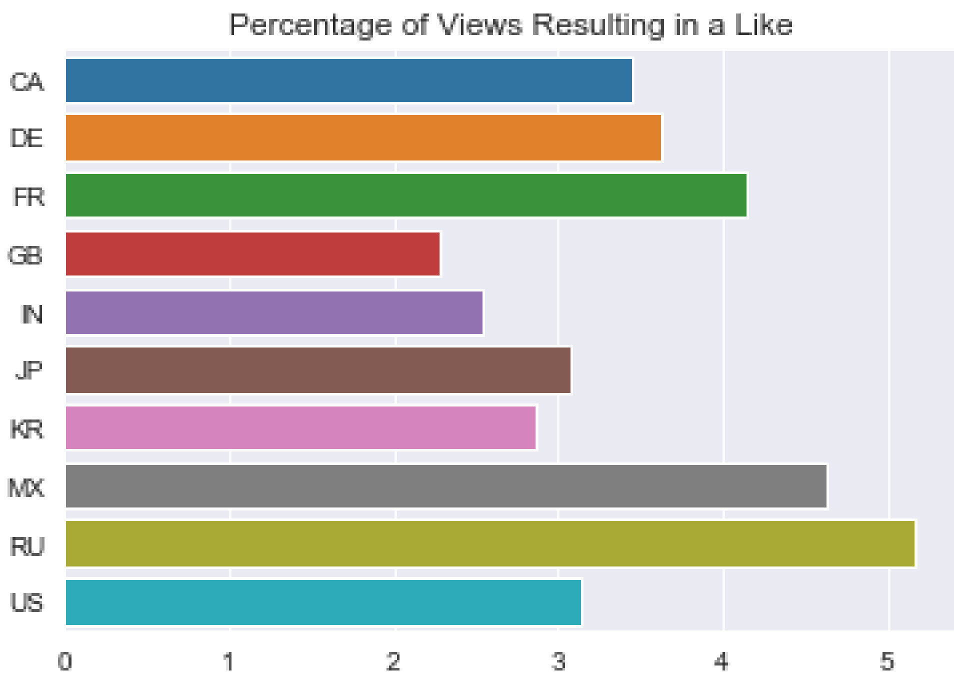
[22]:
```
[3.45, 3.63, 4.14, 2.28, 2.55, 3.08, 2.87, 4.63, 5.17, 3.15]
```

[23]:
```
canada.head(5)
```

[23]:
```
      video_id trending_date  \
0  n1WpP7iowLc      17.14.11
1  0dBIkQ4Mz1M      17.14.11
2  5qpjK5DgCt4      17.14.11
3  d380meD0W0M      17.14.11
4  2Vv-BfVoq4g      17.14.11

                                                title channel_title  \
0         Eminem - Walk On Water (Audio) ft. Beyoncé    EminemVEVO
1                      PLUSH - Bad Unboxing Fan Mail     iDubbbzTV
2  Racist Superman | Rudy Mancuso, King Bach & Le…  Rudy Mancuso
3                           I Dare You: GOING BALD!?      nigahiga
4        Ed Sheeran - Perfect (Official Music Video)    Ed Sheeran

   category_id              publish_time  \
0           10  2017-11-10T17:00:03.000Z
1           23  2017-11-13T17:00:00.000Z
2           23  2017-11-12T19:05:24.000Z
3           24  2017-11-12T18:01:41.000Z
4           10  2017-11-09T11:04:14.000Z
```

```
                                                tags     views    likes  \
0  Eminem|"Walk"|"On"|"Water"|"Aftermath/Shady/In…  17158579   787425
1  plush|"bad unboxing"|"unboxing"|"fan mail"|"id…   1014651   127794
2  racist superman|"rudy"|"mancuso"|"king"|"bach"…   3191434   146035
3  ryan|"higa"|"higatv"|"nigahiga"|"i dare you"|"…   2095828   132239
4  edsheeran|"ed sheeran"|"acoustic"|"live"|"cove…  33523622  1634130

   dislikes  comment_count                                  thumbnail_link  \
0     43420         125882  https://i.ytimg.com/vi/n1WpP7iowLc/default.jpg
1      1688          13030  https://i.ytimg.com/vi/0dBIkQ4Mz1M/default.jpg
2      5339           8181  https://i.ytimg.com/vi/5qpjK5DgCt4/default.jpg
3      1989          17518  https://i.ytimg.com/vi/d380meD0W0M/default.jpg
4     21082          85067  https://i.ytimg.com/vi/2Vv-BfVoq4g/default.jpg

   comments_disabled  ratings_disabled  video_error_or_removed  \
0              False             False                   False
1              False             False                   False
2              False             False                   False
3              False             False                   False
4              False             False                   False

                                         description  like/dislike ratio  \
0  Eminem's new track Walk on Water ft. Beyoncé i…               18.14
1  STill got a lot of packages. Probably will las…               75.71
2  WATCH MY PREVIOUS VIDEO   \n\nSUBSCRIBE   http…               27.35
3  I know it's been a while since we did this sho…               66.49
4   : https://ad.gt/yt-perfect\n : https://atlant…               77.51

   %_Views Resulting in Like
0                       4.59
1                      12.59
2                       4.58
3                       6.31
4                       4.87
```

[24]:
```python
sns.barplot(x=likes_per_view, y=countries_list).set_title('Percentage of Views Resulting in a Like')
plt.savefig('PctOfViewsResultingInLike.png')
```



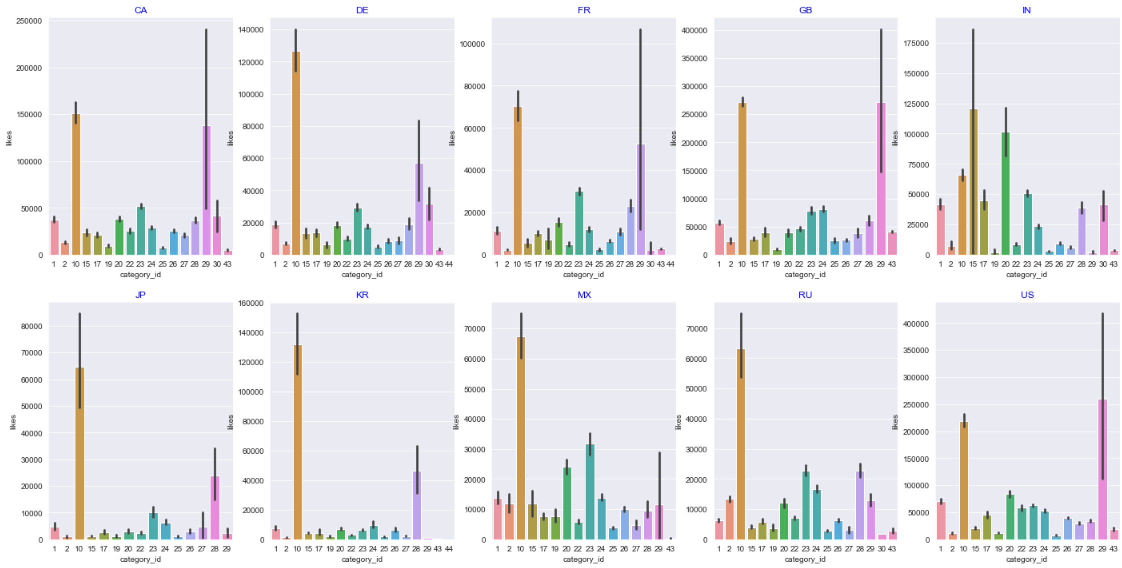
So in fact, Russia actually has the **most likes per view**, with the UK having the least.

3.4.3 Visualising the distribution of likes in each country

The first visualisation will show the mean likes for each `category_id`. Note that the graphs will include error bars, which provide a bit more insight into the spread of the data around the mean within each country and `category_id`.

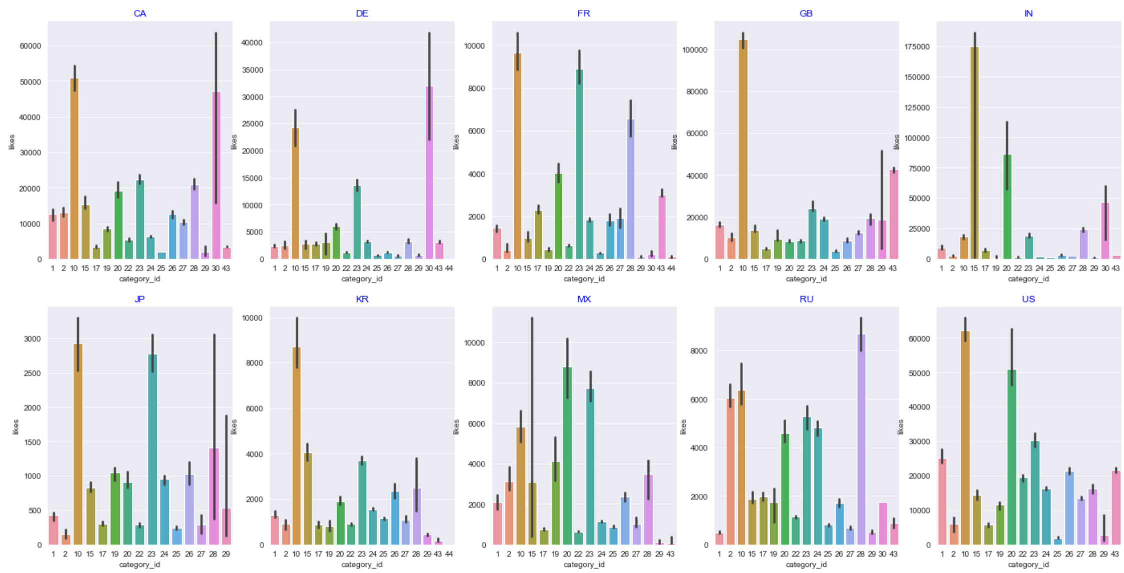
Barplot of mean views

[25]:
```
# the following plot shows the mean likes for each category

fig, axes = plt.subplots(2, 5, figsize = (24, 12))
for i in range(1, (len(countries_df_list))+1):
    ax = plt.subplot(2, 5, i)
    sns.barplot(x=countries_df_list[i-1]['category_id'],\
                y=countries_df_list[i-1]['likes'])\
                .set_title(countries_list[i-1],\
                color='blue')

plt.savefig('MeanLikesPerCat.png')
```



Barplot of median views

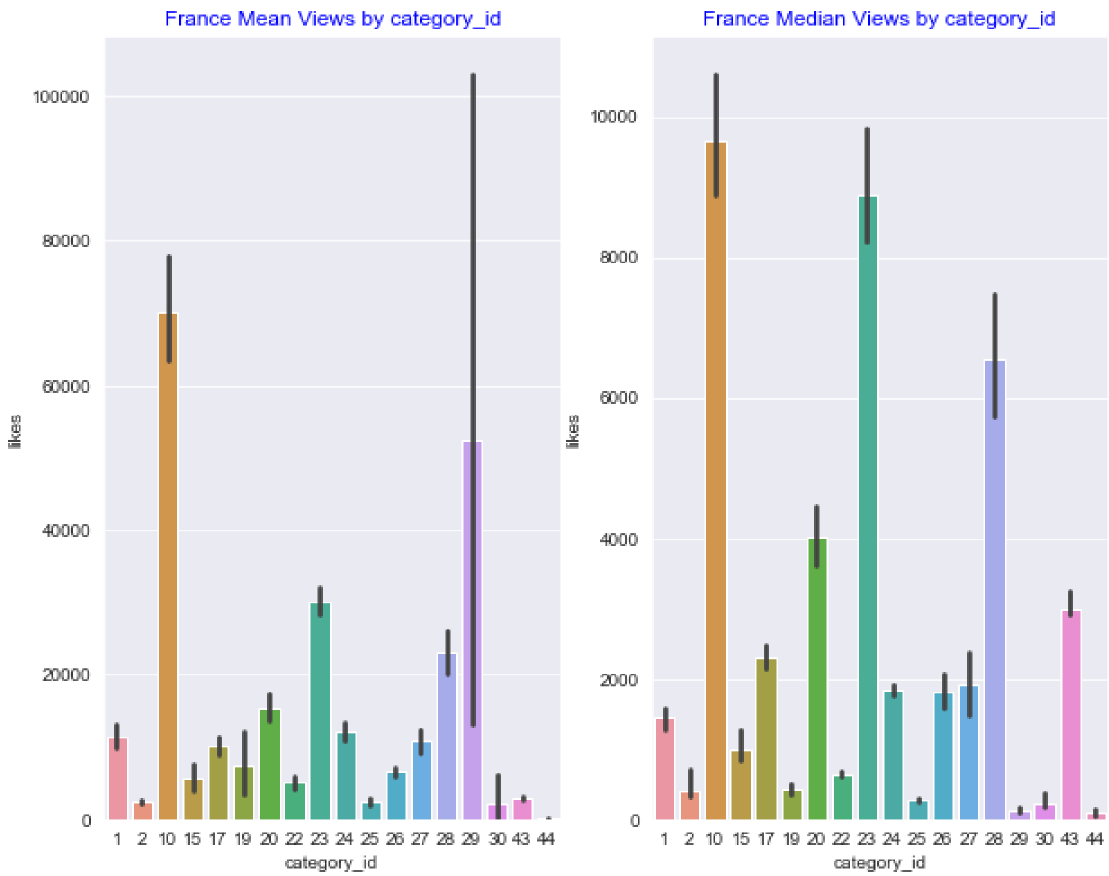
[26]:
```
# the following plot shows the median likes for each category
# this is done by including the "estimator" parameter in the sns.barplot method and specifying "median"
from numpy import median

fig, axes = plt.subplots(2, 5, figsize = (24, 12))
for i in range(1, (len(countries_df_list))+1):
    ax = plt.subplot(2, 5, i)
    sns.barplot(x=countries_df_list[i-1]['category_id'],\
                y=countries_df_list[i-1]['likes'], estimator=median)\
                .set_title(countries_list[i-1],\
                color='blue')

plt.savefig('MedianLikesPerCat.png')
```



Interpreting the results There is quite a lot going on when looking at the above charts - 10 countries with at least 16 categories each, and each with different scales for the y-axis (number of likes).

Rather than going into detail regarding every category for every country, we will look at one interesting example.

[27]:
```python
# plotting the mean views by category next to the median views by category for France.

fig, axes = plt.subplots(1, 2, figsize = (10, 8))

plt.subplot(1, 2, 1)
ax = sns.barplot(x=france['category_id'], y=france['likes']).set_title('France Mean Views by category_id', color='blue')

plt.subplot(1, 2, 2)
sns.set_palette('pastel')
ax = sns.barplot(x=france['category_id'], y=france['likes'], estimator=median).set_title('France Median Views by category_id', color='blue')

plt.savefig('FranceMeanVsMedianLikes.png')
```



If attention is restricted to just **category 29** we can see that the mean number of views is:

1. Largely spread around the mean - some videos have very large numbers of likes and many have very few
2. Much larger than the median (note: scales on each y-axis are not equal)

What could account for such a spread? The large mean vs median would suggest that there is a small number of videos that are attracting large numbers of likes, and there are many other videos in the category that attract only very few likes.

Perhaps this is a controversial category, or perhaps there are only a small number of viral videos that come from this category and so there are only very few videos that make it into the trending datasets.

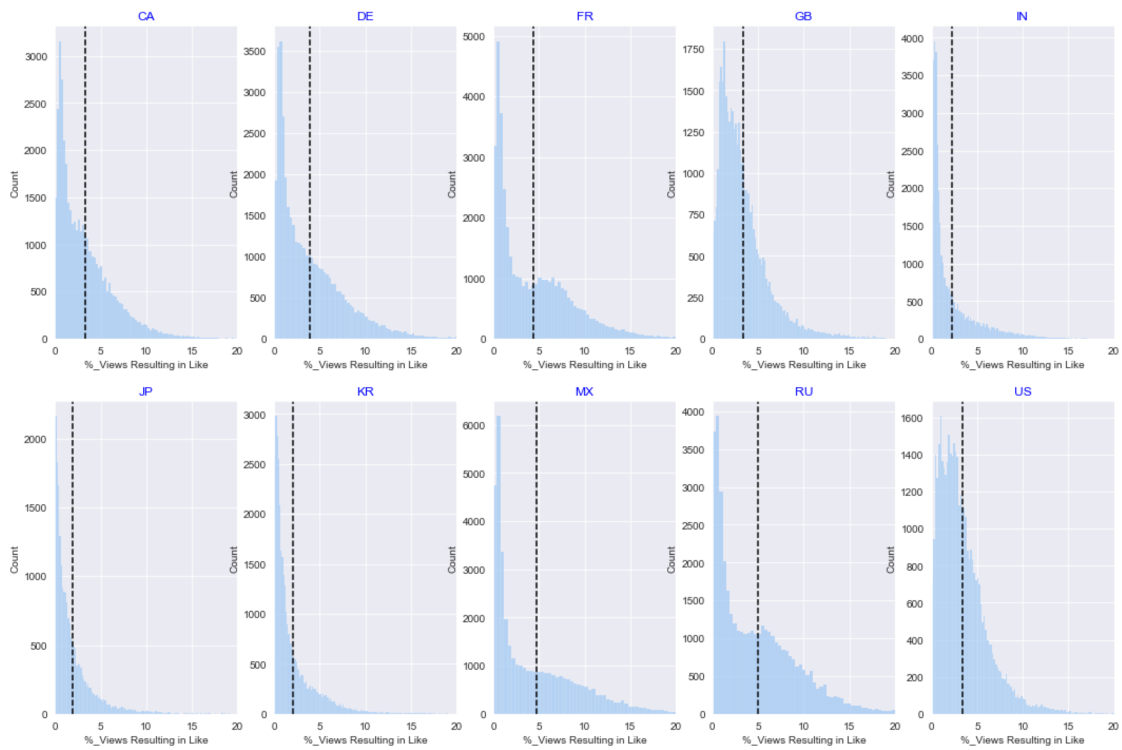
One other thing to note is that many of the other categories are much more stable between different countries, and have smaller uncertainty (e.g. category 10).

3.4.4 Visualising the distribution of percentage of views resulting in a like across each country

[28]:
```
# create a figure that will contain a subplot for each country
# each subplot will have the distribution of the percentage of views resulting␣
↪in a like
# We will also print the mean and standard deviation of the distributions
# the plots will have the mean for each country displayed as a vertical broken␣
↪line on each plot

fig, ax = plt.subplots(2, 5, figsize = (18, 12))
for i in range(1, (len(countries_df_list))+1):
    ax = plt.subplot(2, 5, i)
    sns.histplot(countries_df_list[i-1]['%_Views Resulting in Like'], ax=ax).
↪set_title(countries_list[i-1], color='blue')
    ax.set_xlim(0, 20)
    #sns.violinplot(x=countries_df_list[i-1]['%_Views Resulting in Like'],␣
↪ax=ax).set_title(countries_list[i-1], color='blue')
    #ax.set_xlim(0,20)
    plt.axvline(countries_df_list[i-1]['%_Views Resulting in Like'].mean(),␣
↪color='k', linestyle='dashed', label='mean')
    print("Mean for " + countries_list[i-1] +": {}"\
          .format(np.round(countries_df_list[i-1]['%_Views Resulting in Like'].
↪mean(), decimals=2))), \
    print("Std for " +countries_list[i-1] +": {}"\
          .format(np.round(countries_df_list[i-1]['%_Views Resulting in Like'].
↪std(), decimals=2)))
    print("-"*20)

plt.savefig('DistOfViewsWithLike.png')
```

```
Mean for CA: 3.34
Std for CA: 3.08
--------------------
Mean for DE: 3.94
Std for DE: 3.78
--------------------
Mean for FR: 4.41
Std for FR: 4.24
--------------------
Mean for GB: 3.38
Std for GB: 2.8
--------------------
Mean for IN: 2.2
Std for IN: 2.86
--------------------
Mean for JP: 1.95
```

```
Std for JP: 2.49
--------------------
Mean for KR: 2.07
Std for KR: 2.63
--------------------
Mean for MX: 4.79
Std for MX: 5.06
--------------------
Mean for RU: 4.96
Std for RU: 4.6
--------------------
Mean for US: 3.44
Std for US: 2.7
--------------------
```



4 Analysis of Video Categories

4.1 Analysing using SQL

SQL's functionality will make it easy to aggregate the data in each of the DataFrames and provide quick insight.

We will create a SQL table for each of the countries using the Pandas DataFrames.

[29]:
```
# initialise the database that will store the tables
conn = sqlite3.connect('youtube_stats.db')

# then create tables for each country and add them to the database
canada_table = canada.to_sql('canada_table', conn)
germany_table = germany.to_sql('germany_table', conn)
france_table = france.to_sql('france_table', conn)
uk_table = uk.to_sql('uk_table', conn)
india_table = india.to_sql('india_table', conn)
japan_table = japan.to_sql('japan_table', conn)
korea_table = korea.to_sql('korea_table', conn)
mexico_table = mexico.to_sql('mexico_table', conn)
russia_table = russia.to_sql('russia_table', conn)
usa_table = usa.to_sql('usa_table', conn)
```

```
C:\Users\Stewa\Documents\Miniconda\lib\site-
packages\pandas\core\generic.py:2663: UserWarning: The spaces in these column
names will not be changed. In pandas versions < 0.14, spaces were converted to
underscores.
  method=method,
```

[30]:
```
# load the sql extension
%load_ext sql
```

[31]:
```
# connect sqlite to the newly created database
%sql sqlite:///youtube_stats.db
```

[31]:
```
'Connected: @youtube_stats.db'
```

[32]:
```
%%sql
SELECT *
FROM canada_table
LIMIT 5
```

```
 * sqlite:///youtube_stats.db
Done.
```

[32]:
```
[(0, 'n1WpP7iowLc', '17.14.11', 'Eminem - Walk On Water (Audio) ft. Beyoncé',
'EminemVEVO', 10, '2017-11-10T17:00:03.000Z',
'Eminem|"Walk"|"On"|"Water"|"Aftermath/Shady/Interscope"|"Rap"', 17158579,
787425, 43420, 125882, 'https://i.ytimg.com/vi/n1WpP7iowLc/default.jpg', 0, 0,
0, "Eminem's new track Walk on Water ft. Beyoncé is available everywhere:
http://shady.sr/WOWEminem \\nPlaylist Best of Eminem: https://goo.gl/AquNpo\\nS
… (313 characters truncated) …
m/shadyrecords\\nhttp://trustshady.tumblr.com\\n\\nMusic video by Eminem
performing Walk On Water. (C) 2017 Aftermath Records\\nhttp://vevo.ly/gA7xKt",
18.14, 4.59),
```

(1, '0dBIkQ4Mz1M', '17.14.11', 'PLUSH - Bad Unboxing Fan Mail', 'iDubbbzTV', 23, '2017-11-13T17:00:00.000Z', 'plush|"bad unboxing"|"unboxing"|"fan mail"|"idubbbztv"|"idubbbztv2"|"things"|"best"|"packages"|"plushies"|"chontent chop"', 1014651, 127794, 1688, 13030, 'https://i.ytimg.com/vi/0dBIkQ4Mz1M/default.jpg', 0, 0, 0, "STill got a lot of packages. Probably will last for another year. On a side note, more 2nd channel vids soon. editing with premiere from now on, gon' … (422 characters truncated) … stagram.com/idubbbz/\\nTwitter https://twitter.com/Idubbbz\\nFacebook http://www.facebook.com/IDubbbz\\nTwitch http://www.twitch.tv/idubbbz\\n_", 75.71, 12.59),
(2, '5qpjK5DgCt4', '17.14.11', 'Racist Superman | Rudy Mancuso, King Bach & Lele Pons', 'Rudy Mancuso', 23, '2017-11-12T19:05:24.000Z', 'racist superman|"rudy"|"mancuso"|"king"|"bach"|"racist"|"superman"|"love"|"rudy mancuso poo bear black white official music video"|"iphone x by pinea … (17 characters truncated) … "hannahstocking"|"rudymancuso"|"inanna"|"anwar"|"sarkis"|"shots"|"shotsstudios"|"alesso"|"anitta"|"brazil"|"Getting My Driver\'s License | Lele Pons"', 3191434, 146035, 5339, 8181, 'https://i.ytimg.com/vi/5qpjK5DgCt4/default.jpg', 0, 0, 0, 'WATCH MY PREVIOUS VIDEO \\n\\nSUBSCRIBE https://www.youtube.com/channel/UC5jkXpfnBhlDjqh0ir5FsIQ?sub_confirmation=1\\n\\nTHANKS FOR WATCHING! LIK … (916 characters truncated) … p://youtube.com/c/miketyson \\nRudy Mancuso | http://youtube.com/c/rudymancuso\\nShots Studios | http://youtube.com/c/shots\\n\\n#Rudy\\n#RudyMancuso', 27.35, 4.58),
(3, 'd380meD0W0M', '17.14.11', 'I Dare You: GOING BALD!?', 'nigahiga', 24, '2017-11-12T18:01:41.000Z', 'ryan|"higa"|"higatv"|"nigahiga"|"i dare you"|"idy"|"rhpc"|"dares"|"no truth"|"comments"|"comedy"|"funny"|"stupid"|"fail"', 2095828, 132239, 1989, 17518, 'https://i.ytimg.com/vi/d380meD0W0M/default.jpg', 0, 0, 0, "I know it's been a while since we did this show, but we're back with what might be the best episode yet!\\nLeave your dares in the comment section! \ … (364 characters truncated) … w.higatv.com\\n\\nInstagram\\nhttp://www.instagram.com/notryanhiga\\n\\nSend us mail or whatever you want here!\\nPO Box 232355\\nLas Vegas, NV 89105", 66.49, 6.31),
(4, '2Vv-BfVoq4g', '17.14.11', 'Ed Sheeran - Perfect (Official Music Video)', 'Ed Sheeran', 10, '2017-11-09T11:04:14.000Z', 'edsheeran|"ed sheeran"|"acoustic"|"live"|"cover"|"official"|"remix"|"official video"|"lyrics"|"session"', 33523622, 1634130, 21082, 85067, 'https://i.ytimg.com/vi/2Vv-BfVoq4g/default.jpg', 0, 0, 0, " : https://ad.gt/yt-perfect\\n : https://atlanti.cr/yt-album\\nSubscribe to Ed's channel: http://bit.ly/SubscribeToEdSheeran\\n\\nFollow Ed on…\\nF … (996 characters truncated) … rian Casting: Ursula Kiplinger\\n \\nAdditional VFX: Zoic\\n\\nSpecial Thanks to: The Hintertux Glacier, Austria;\\nThe Tenne, and Hotel Neuhintertux", 77.51, 4.87)]

[33]:
```
%%sql
SELECT category_id, SUM(views) as views
FROM canada_table
GROUP BY category_id
ORDER BY category_id ASC
```

```
 * sqlite:///youtube_stats.db
Done.
```

[33]:
```
[(1, 2939060844),
 (2, 200066074),
 (10, 13179850194),
 (15, 235592173),
 (17, 2997652188),
 (19, 143746952),
 (20, 1241532385),
 (22, 3228227926),
 (23, 3708438785),
 (24, 13671215509),
 (25, 1614610043),
 (26, 1570846611),
 (27, 531773343),
 (28, 1425090421),
 (29, 115601623),
 (30, 17120490),
 (43, 71549508)]
```

So, above we can see that we have category numbers and the associated total views for each of them.

[34]:
```
%%sql
SELECT category_id, SUM(views) as views
FROM canada_table
GROUP BY category_id
ORDER BY views DESC
```

```
 * sqlite:///youtube_stats.db
Done.
```

[34]:
```
[(24, 13671215509),
 (10, 13179850194),
 (23, 3708438785),
 (22, 3228227926),
 (17, 2997652188),
 (1, 2939060844),
 (25, 1614610043),
 (26, 1570846611),
 (28, 1425090421),
```

```
(20, 1241532385),
(27, 531773343),
(15, 235592173),
(2, 200066074),
(19, 143746952),
(29, 115601623),
(43, 71549508),
(30, 17120490)]
```

Here, we can see that the top two `category_id`s contain by far the most views, but we have no idea what these categories are.

To discover what each `category_id` is, we will need the JSON files.

We will start by saving each query as into a pandas DataFrame that we constructed to get the above table

[35]:
```python
# save the query for each country in a new DataFrame
canada_views_by_category = pd.read_sql("""
                                SELECT DISTINCT category_id, SUM(views) as views
                                FROM canada_table
                                GROUP BY category_id
                                ORDER BY category_id asc
                                """, con = conn)

germany_views_by_category = pd.read_sql("""
                                SELECT DISTINCT category_id, SUM(views) as views
                                FROM germany_table
                                GROUP BY category_id
                                ORDER BY category_id asc
                                """, con = conn)

france_views_by_category = pd.read_sql("""
                                SELECT DISTINCT category_id, SUM(views) as views
                                FROM france_table
                                GROUP BY category_id
                                ORDER BY category_id asc
                                """, con = conn)

uk_views_by_category = pd.read_sql("""
                                SELECT DISTINCT category_id, SUM(views) as views
                                FROM uk_table
                                GROUP BY category_id
                                ORDER BY category_id asc
                                """, con = conn)

india_views_by_category = pd.read_sql("""
                                SELECT DISTINCT category_id, SUM(views) as views
```

```
                                FROM india_table
                                GROUP BY category_id
                                ORDER BY category_id asc
                                """, con = conn)

japan_views_by_category = pd.read_sql("""
                                SELECT DISTINCT category_id, SUM(views) as views
                                FROM japan_table
                                GROUP BY category_id
                                ORDER BY category_id asc
                                """, con = conn)

korea_views_by_category = pd.read_sql("""
                                SELECT DISTINCT category_id, SUM(views) as views
                                FROM korea_table
                                GROUP BY category_id
                                ORDER BY category_id asc
                                """, con = conn)

mexico_views_by_category = pd.read_sql("""
                                SELECT DISTINCT category_id, SUM(views) as views
                                FROM mexico_table
                                GROUP BY category_id
                                ORDER BY category_id asc
                                """, con = conn)

russia_views_by_category = pd.read_sql("""
                                SELECT DISTINCT category_id, SUM(views) as views
                                FROM russia_table
                                GROUP BY category_id
                                ORDER BY category_id asc
                                """, con = conn)

usa_views_by_category = pd.read_sql("""
                                SELECT DISTINCT category_id, SUM(views) as views
                                FROM usa_table
                                GROUP BY category_id
                                ORDER BY category_id asc
                                """, con = conn)
```

4.2 Dealing with differences in `category_ids`

The datasets for each country may have different `category_ids` or different numbers of categories.

[36]:
```
# the following list will be useful in the upcoming analysis
countries_views_by_category = [canada_views_by_category,
                              germany_views_by_category,
```

```
                              france_views_by_category,
                              uk_views_by_category,
                              india_views_by_category,
                              japan_views_by_category,
                              korea_views_by_category,
                              mexico_views_by_category,
                              russia_views_by_category,
                              usa_views_by_category]

# count the number of unique category_ids in each dataset
for country in countries_views_by_category:
    print(country['category_id'].nunique())
```

```
17
18
18
16
17
15
17
16
17
16
```

Above we can see that the different countries do not necessarily have the same number of unique `category_ids` - we need to find which `category_ids` are missing from the datasets with fewer unique `category_ids`. The following will do this:

[37]:
```
# the following will print the category_ids present in each country's dataset
for i in range(len(countries_views_by_category)):
    print(countries_list[i] +":", countries_views_by_category[i]['category_id'].
↪unique())
```

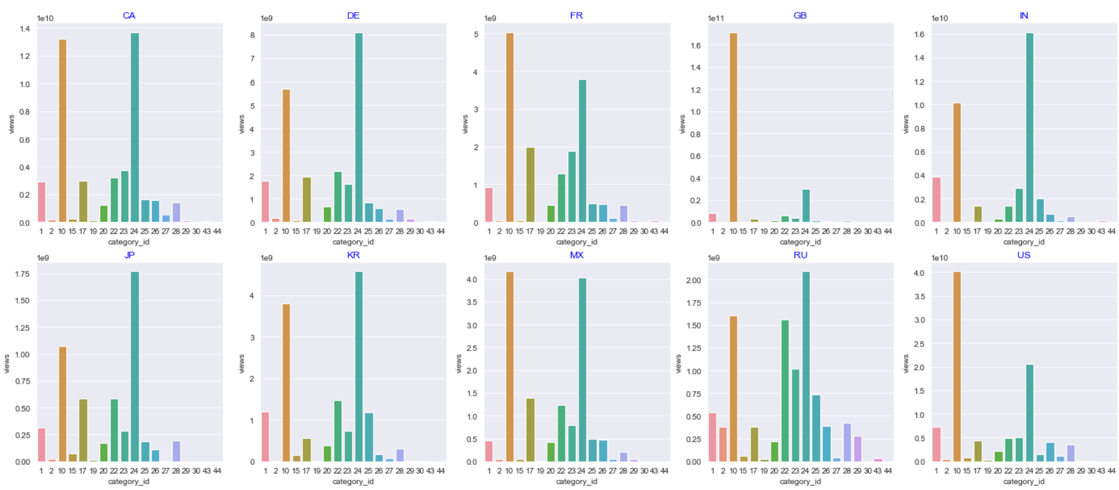
```
CA: [ 1  2 10 15 17 19 20 22 23 24 25 26 27 28 29 30 43]
DE: [ 1  2 10 15 17 19 20 22 23 24 25 26 27 28 29 30 43 44]
FR: [ 1  2 10 15 17 19 20 22 23 24 25 26 27 28 29 30 43 44]
GB: [ 1  2 10 15 17 19 20 22 23 24 25 26 27 28 29 43]
IN: [ 1  2 10 15 17 19 20 22 23 24 25 26 27 28 29 30 43]
JP: [ 1  2 10 15 17 19 20 22 23 24 25 26 27 28 29]
KR: [ 1  2 10 15 17 19 20 22 23 24 25 26 27 28 29 43 44]
MX: [ 1  2 10 15 17 19 20 22 23 24 25 26 27 28 29 43]
RU: [ 1  2 10 15 17 19 20 22 23 24 25 26 27 28 29 30 43]
US: [ 1  2 10 15 17 19 20 22 23 24 25 26 27 28 29 43]
```

So, both `germany` and `france` contain a complete list of `category_ids` - we will use this complete list when visualising the `views` by `category_id` in the following sections.

4.3 Visualising the views per numerical `category_id` by country

Although we do not yet know what these `category_id`s represent, it will still be insightful to visualise the views by `category_id` before going through the process of matching them to names.

[38]:
```
fig, ax = plt.subplots(2, 5, figsize = (24, 10))
for i in range(1, (len(countries_views_by_category))+1):
    ax = plt.subplot(2, 5, i)
    # use countries_views_by_category[1] to choose the full list of category_ids from the germany data
    # this will give the plots matching x-axes
    sns.barplot(x=countries_views_by_category[1]['category_id'], y=countries_views_by_category[i-1]['views'])\
                .set_title(countries_list[i-1], color='blue')

plt.savefig('ViewsByCategoryID.png')
```



5 Data Ingestion - JSON files

The JSON files contain the metadata of the YouTube videos. We will load all of the files, investigate them, and see how they can assist the analysis.

[39]:
```
# opening the JSON files in read mode
ca_json=open('CA_category_id.json', 'r')
de_json=open('DE_category_id.json', 'r')
fr_json=open('FR_category_id.json', 'r')
gb_json=open('GB_category_id.json', 'r')
in_json=open('IN_category_id.json', 'r')
jp_json=open('JP_category_id.json', 'r')
kr_json=open('KR_category_id.json', 'r')
mx_json=open('MX_category_id.json', 'r')
```

```
ru_json=open('RU_category_id.json', 'r')
us_json=open('US_category_id.json', 'r')
```

[40]:
```
# converting the JSON files into ojects, one for each of the countries:
canada_json = json.loads(ca_json.read())
germany_json = json.loads(de_json.read())
france_json = json.loads(fr_json.read())
uk_json = json.loads(gb_json.read())
india_json = json.loads(in_json.read())
japan_json = json.loads(jp_json.read())
korea_json = json.loads(kr_json.read())
mexico_json = json.loads(mx_json.read())
russia_json = json.loads(ru_json.read())
usa_json = json.loads(us_json.read())
```

[41]:
```
# The following list will be useful to perform operations on all of the␣
↪countries' JSON data
countries_json_list = [canada_json,
                       germany_json,
                       france_json,
                       uk_json,
                       india_json,
                       japan_json,
                       korea_json,
                       mexico_json,
                       russia_json,
                       usa_json]
```

Having a look at one of the JSON files:

[42]:
```
canada_json
```

[42]:

```
{'kind': 'youtube#videoCategoryListResponse',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/1v2mrzYSYG6onNLt2qTj13hkQZk"',
 'items': [{'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/Xy1mB4_yLrHy_BmKmPBggty2mZQ"',
   'id': '1',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Film & Animation',
    'assignable': True}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/UZ1oLIIz2dxIhO45ZTFR3a3NyTA"',
   'id': '2',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Autos & Vehicles',
    'assignable': True}},
  {'kind': 'youtube#videoCategory',
```

```
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/nqRIq97-xe5XRZTxbknKFVe5Lmg"',
 'id': '10',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Music',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/HwXKamM1Q20q9BN-oBJavSGkfDI"',
 'id': '15',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Pets & Animals',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/9GQMSRjrZdHeb1OEM1XVQ9zbGec"',
 'id': '17',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Sports',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/FJwVpGCVZ1yiJrqZbpqe68Sy_OE"',
 'id': '18',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Short Movies',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/M-3iD9dwK7YJCafRf_DkLN8CouA"',
 'id': '19',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Travel & Events',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/WmA0qYEfjWsAoyJFSw2zinhn2wM"',
 'id': '20',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Gaming',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/EapFaGYG7K0StIXVf8aba249tdM"',
 'id': '21',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Videoblogging',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/xId8RX7vRN8rqkbYZbNIytUQDRo"',
 'id': '22',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'People & Blogs',
  'assignable': True}},
```

```
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/G9LHzQmx44rX2S5yaga_Aqtwz8M"',
 'id': '23',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Comedy',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/UVB9oxX2Bvqa_w_y3vXSLVK5E_s"',
 'id': '24',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Entertainment',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/QiLK0ZIrFoORdk_g2l_XR_ECjDc"',
 'id': '25',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'News & Politics',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/r6Ck6Z0_L0rG37VJQR200SGNA_w"',
 'id': '26',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Howto & Style',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/EoYkczo9I3RCf96RveKTOgOPkUM"',
 'id': '27',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Education',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/w5HjcTD82G_XA3xBctS30zS-JpQ"',
 'id': '28',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Science & Technology',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/lL7uWDr_071CHxifjYG1tJrp4Uo"',
 'id': '30',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Movies',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/WnuVfjO-PyFLO7NTRQIbrGE62nk"',
 'id': '31',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Anime/Animation',
```

```
  'assignable': False}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/ctpH2hGA_UZ3volJT_FTlOg9M00"',
  'id': '32',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Action/Adventure',
   'assignable': False}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/L0kR3-g1BAo5UD1PLVbQ7LkkDtQ"',
  'id': '33',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Classics',
   'assignable': False}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/pUZOAC_s9sfiwar639qr_wAB-aI"',
  'id': '34',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Comedy',
   'assignable': False}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/Xb5JLhtyNRN3AQq021Ds-OV50Jk"',
  'id': '35',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Documentary',
   'assignable': False}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/u8WXzF4HIhtEi805__sqjuA4lEk"',
  'id': '36',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Drama',
   'assignable': False}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/D04PP4Gr7wc4IV_O9G66Z4A8KWQ"',
  'id': '37',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Family',
   'assignable': False}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/i5-_AceGXQCEEMWU0V8CcQm_vLQ"',
  'id': '38',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Foreign',
   'assignable': False}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/rtlxd0zOixA9QHdIZB26-St5qgQ"',
  'id': '39',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
```

```
    'title': 'Horror',
    'assignable': False}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/N1TrDFLRppxZgBowCJfJCvh0Dpg"',
   'id': '40',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Sci-Fi/Fantasy',
    'assignable': False}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/7UMGi6zRySqXopr_rv4sZq6Za2E"',
   'id': '41',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Thriller',
    'assignable': False}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/RScXhi324h8usyIetreAVb-uKeM"',
   'id': '42',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Shorts',
    'assignable': False}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/0n9MJVCDLpA8q7aiGVrFsuFsd0A"',
   'id': '43',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Shows',
    'assignable': False}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/x5NxSf5fz8hn4loSN4rvhwzD_pY"',
   'id': '44',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Trailers',
    'assignable': False}}]}
```


[43]:
```
canada_json.keys()
```

[43]:
```
dict_keys(['kind', 'etag', 'items'])
```

In the JSON file we can see that we have nested dictionaries, and in one of the dictionaries we have a key names "`id`" and another named "`title`". This will allow us to associate all of the `category_ids` with lexical titles.

Apart from this, there does not appear to be much interesting information. The `channelId` key is the same for every category, which suggest this is the `channelId` of the user who scraped the data using the YouTube API.

[44]:
```
germany_json
```

[44]:

```
{'kind': 'youtube#videoCategoryListResponse',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/1v2mrzYSYG6onNLt2qTj13hkQZk"',
 'items': [{'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/Xy1mB4_yLrHy_BmKmPBggty2mZQ"',
   'id': '1',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Film & Animation',
    'assignable': True}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/UZ1oLIIz2dxIh045ZTFR3a3NyTA"',
   'id': '2',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Autos & Vehicles',
    'assignable': True}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/nqRIq97-xe5XRZTxbknKFVe5Lmg"',
   'id': '10',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Music',
    'assignable': True}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/HwXKamM1Q20q9BN-oBJavSGkfDI"',
   'id': '15',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Pets & Animals',
    'assignable': True}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/9GQMSRjrZdHeb10EM1XVQ9zbGec"',
   'id': '17',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Sports',
    'assignable': True}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/FJwVpGCVZ1yiJrqZbpqe68Sy_0E"',
   'id': '18',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Short Movies',
    'assignable': False}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/M-3iD9dwK7YJCafRf_DkLN8CouA"',
   'id': '19',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Travel & Events',
    'assignable': True}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/WmA0qYEfjWsAoyJFSw2zinhn2wM"',
   'id': '20',
```

```
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Gaming',
   'assignable': True}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/EapFaGYG7K0StIXVf8aba249tdM"',
  'id': '21',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Videoblogging',
   'assignable': False}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/xId8RX7vRN8rqkbYZbNIytUQDRo"',
  'id': '22',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'People & Blogs',
   'assignable': True}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/G9LHzQmx44rX2S5yaga_Aqtwz8M"',
  'id': '23',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Comedy',
   'assignable': True}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/UVB9oxX2Bvqa_w_y3vXSLVK5E_s"',
  'id': '24',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Entertainment',
   'assignable': True}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/QiLK0ZIrFoORdk_g2l_XR_ECjDc"',
  'id': '25',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'News & Politics',
   'assignable': True}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/r6Ck6Z0_L0rG37VJQR200SGNA_w"',
  'id': '26',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Howto & Style',
   'assignable': True}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/EoYkczo9I3RCf96RveKTOgOPkUM"',
  'id': '27',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Education',
   'assignable': True}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/w5HjcTD82G_XA3xBctS30zS-JpQ"',
```

```
 'id': '28',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Science & Technology',
  'assignable': True}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/lL7uWDr_071CHxifjYG1tJrp4Uo"',
 'id': '30',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Movies',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/WnuVfjO-PyFLO7NTRQIbrGE62nk"',
 'id': '31',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Anime/Animation',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/ctpH2hGA_UZ3volJT_FTlOg9M00"',
 'id': '32',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Action/Adventure',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/L0kR3-g1BAo5UD1PLVbQ7LkkDtQ"',
 'id': '33',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Classics',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/pUZOAC_s9sfiwar639qr_wAB-aI"',
 'id': '34',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Comedy',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/Xb5JLhtyNRN3AQq021Ds-OV50Jk"',
 'id': '35',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Documentary',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/u8WXzF4HIhtEi805__sqjuA4lEk"',
 'id': '36',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Drama',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
```

```
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/D04PP4Gr7wc4IV_O9G66Z4A8KWQ"',
 'id': '37',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Family',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/i5-_AceGXQCEEMWU0V8CcQm_vLQ"',
 'id': '38',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Foreign',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/rtlxd0zOixA9QHdIZB26-St5qgQ"',
 'id': '39',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Horror',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/N1TrDFLRppxZgBowCJfJCvh0Dpg"',
 'id': '40',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Sci-Fi/Fantasy',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/7UMGi6zRySqXopr_rv4sZq6Za2E"',
 'id': '41',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Thriller',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/RScXhi324h8usyIetreAVb-uKeM"',
 'id': '42',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Shorts',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/0n9MJVCDLpA8q7aiGVrFsuFsd0A"',
 'id': '43',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Shows',
  'assignable': False}},
{'kind': 'youtube#videoCategory',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/x5NxSf5fz8hn4loSN4rvhwzD_pY"',
 'id': '44',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Trailers',
  'assignable': False}}]}
```

Again, this `channelId` matches the one in the `canada_json` file and so isn't of any analytical use.

Now, we need to access the part of the `JSON` object that relates the category_id to a name. The following shows how to do this:

[45]:
```
# accessing the first id number, then its corresponding title

print(canada_json['items'][0]['id'])

print(canada_json['items'][0]['snippet']['title'])
```

```
1
Film & Animation
```

5.1 Function to create (key, value) pairs of category_id and title

We can create a function that will match each `category_id` to a `title` and add these to a dictionary as `key:value` pairs

[46]:
```
def category_names(dictionary):
    categories = {}

    for i in range(len(dictionary['items'])):
        categories[dictionary['items'][i]['id']] =␣
 ↪dictionary['items'][i]['snippet']['title']

    return categories
```

[47]:
```
# check that all the JSON files have the same number of category titles
for country in countries_json_list:
    print(len(category_names(country)))
```

```
31
31
31
31
31
31
31
31
31
32
```

So, the final one, which is the `usa_json` file, contains one more category than the other countries.

Another thing to note is that there are **almost double the number of category `titles` present in the `JSON` files than there are unique `category_ids` in the datasets** (this ranged from 16-18 distinct `category_ids`).

This means that for some of the category `titles` present in the `JSON` files, there are no views in the datasets.

This itself is an interesting insight - we can see the names of the categories that are not present in the datasets and hence the names of the categories that are very unlikely to have trending videos (over the six month period that the data were collected, there were no trending videos from this category.

We will need to filter the extra category `title`s out when visualising the views by category `title`.

[48]:
```
# use the category_names function to show the usa category titles as this is␣
↪the most complete collection
category_names(usa_json)
```

[48]:
```
{'1': 'Film & Animation',
 '2': 'Autos & Vehicles',
 '10': 'Music',
 '15': 'Pets & Animals',
 '17': 'Sports',
 '18': 'Short Movies',
 '19': 'Travel & Events',
 '20': 'Gaming',
 '21': 'Videoblogging',
 '22': 'People & Blogs',
 '23': 'Comedy',
 '24': 'Entertainment',
 '25': 'News & Politics',
 '26': 'Howto & Style',
 '27': 'Education',
 '28': 'Science & Technology',
 '29': 'Nonprofits & Activism',
 '30': 'Movies',
 '31': 'Anime/Animation',
 '32': 'Action/Adventure',
 '33': 'Classics',
 '34': 'Comedy',
 '35': 'Documentary',
 '36': 'Drama',
 '37': 'Family',
 '38': 'Foreign',
 '39': 'Horror',
 '40': 'Sci-Fi/Fantasy',
 '41': 'Thriller',
 '42': 'Shorts',
 '43': 'Shows',
 '44': 'Trailers'}
```

5.1.1 Finding the category title that is present only in the USA data

[49]:
```
# create two temporary dictionaries that can be used to find the key, value␣
↪pair that exists only in the USA JSON file
usa_temp_dict = category_names(usa_json)
canada_temp_dict = category_names(canada_json)

for key, value in usa_temp_dict.items():
    if value not in canada_temp_dict.values():
        print("Category_ID: "+key)
        print("Title: "+value)
```

```
Category_ID: 29
Title: Nonprofits & Activism
```

[50]:
```
# for the usa, there are exactly double the number of category titles in the␣
↪JSON files vs category_ids in the dataset
print((len(category_names(usa_json)), len(usa_views_by_category)))
```

```
(32, 16)
```

5.1.2 Filtering the categories

As seen earlier, `germany` and `france` both have the most complete lists of `category_ids`, and `usa` has the most complete list of category `titles`. So we now have a full list of the maximum number of `category_ids` present in any of the datasets, plus a complete list of all the category `titles`.

We can use these to create a filter that will remove all of the category `titles` that have no data.

[51]:
```
# storing the full list of category_ids in a new object called category_ids
category_ids = list((germany_views_by_category)['category_id'])
category_ids
```

[51]:
```
[1, 2, 10, 15, 17, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 43, 44]
```

[52]:
```
# storing the full list of categories in a new object called categories
categories = category_names(usa_json)
categories
```

[52]:
```
{'1': 'Film & Animation',
 '2': 'Autos & Vehicles',
 '10': 'Music',
 '15': 'Pets & Animals',
 '17': 'Sports',
 '18': 'Short Movies',
 '19': 'Travel & Events',
 '20': 'Gaming',
 '21': 'Videoblogging',
 '22': 'People & Blogs',
```

```
 '23': 'Comedy',
 '24': 'Entertainment',
 '25': 'News & Politics',
 '26': 'Howto & Style',
 '27': 'Education',
 '28': 'Science & Technology',
 '29': 'Nonprofits & Activism',
 '30': 'Movies',
 '31': 'Anime/Animation',
 '32': 'Action/Adventure',
 '33': 'Classics',
 '34': 'Comedy',
 '35': 'Documentary',
 '36': 'Drama',
 '37': 'Family',
 '38': 'Foreign',
 '39': 'Horror',
 '40': 'Sci-Fi/Fantasy',
 '41': 'Thriller',
 '42': 'Shorts',
 '43': 'Shows',
 '44': 'Trailers'}
```

[53]:
```
# now filtering the categories to only contain the same ones present in the␣
↪datasets
filtered_cats = [categories[str(category_ids[i])] for i in␣
↪range(len(category_ids))]
filtered_cats
```

[53]:
```
['Film & Animation',
 'Autos & Vehicles',
 'Music',
 'Pets & Animals',
 'Sports',
 'Travel & Events',
 'Gaming',
 'People & Blogs',
 'Comedy',
 'Entertainment',
 'News & Politics',
 'Howto & Style',
 'Education',
 'Science & Technology',
 'Nonprofits & Activism',
 'Movies',
 'Shows',
 'Trailers']
```

5.1.3 The categories which are not present in any of the datasets

This will be a list of categories which are unlikely to produce trending videos.

[54]:
```
missing_categories = []
for value in categories.values():
    if value not in filtered_cats:
        missing_categories.append(value)

missing_categories
```

[54]:
```
['Short Movies',
 'Videoblogging',
 'Anime/Animation',
 'Action/Adventure',
 'Classics',
 'Documentary',
 'Drama',
 'Family',
 'Foreign',
 'Horror',
 'Sci-Fi/Fantasy',
 'Thriller',
 'Shorts']
```

6 Visualising the views per category name

Now that we have associated each of the `category_ids` with `titles`, we can plot the total views per category title for each country

[55]:
```
fig, ax = plt.subplots(2, 5, figsize = (24, 28))
for i in range(1, (len(countries_views_by_category))+1):
    ax = plt.subplot(5, 2, i)
    sns.barplot(x=list(countries_views_by_category[i-1]['views']),␣
 ↪y=filtered_cats[:len(countries_views_by_category[i-1])])\
                .set_title(countries_list[i-1], color='blue')

plt.savefig("CountryViewsByCategoryName.png")
```

7 Aggregating across all datasets

We can see that there are significant differences in the videos people from these 10 different countries watch.

Now we will concatenate the datasets for each of the countries and visualise the viewing statistics for all of the datasets as a whole.

Rather than use `SQL` this time, we will use pandas to perform the aggregation.

[56]:
```
# use the list created at the start of the notebook which holds all of the␣
↪countries' DataFrames
concatenated_df = pd.concat(countries_df_list)
```

[57]:
```
concatenated_df.shape
```

[57]:
```
(375942, 18)
```

[58]:
```
concatenated_df.head()
```

[58]:
```
      video_id trending_date  \
0  n1WpP7iowLc      17.14.11
1  0dBIkQ4Mz1M      17.14.11
2  5qpjK5DgCt4      17.14.11
3  d380meD0W0M      17.14.11
4  2Vv-BfVoq4g      17.14.11

                                               title channel_title  \
0         Eminem - Walk On Water (Audio) ft. Beyoncé    EminemVEVO
1                      PLUSH - Bad Unboxing Fan Mail     iDubbbzTV
2  Racist Superman | Rudy Mancuso, King Bach & Le…  Rudy Mancuso
3                           I Dare You: GOING BALD!?      nigahiga
4        Ed Sheeran - Perfect (Official Music Video)    Ed Sheeran

   category_id              publish_time  \
0           10  2017-11-10T17:00:03.000Z
1           23  2017-11-13T17:00:00.000Z
2           23  2017-11-12T19:05:24.000Z
3           24  2017-11-12T18:01:41.000Z
4           10  2017-11-09T11:04:14.000Z

                                                tags     views    likes  \
0  Eminem|"Walk"|"On"|"Water"|"Aftermath/Shady/In…  17158579   787425
1  plush|"bad unboxing"|"unboxing"|"fan mail"|"id…   1014651   127794
2  racist superman|"rudy"|"mancuso"|"king"|"bach"…   3191434   146035
3  ryan|"higa"|"higatv"|"nigahiga"|"i dare you"|"…   2095828   132239
4  edsheeran|"ed sheeran"|"acoustic"|"live"|"cove…  33523622  1634130

   dislikes  comment_count                                  thumbnail_link  \
0     43420         125882  https://i.ytimg.com/vi/n1WpP7iowLc/default.jpg
1      1688          13030  https://i.ytimg.com/vi/0dBIkQ4Mz1M/default.jpg
2      5339           8181  https://i.ytimg.com/vi/5qpjK5DgCt4/default.jpg
3      1989          17518  https://i.ytimg.com/vi/d380meD0W0M/default.jpg
4     21082          85067  https://i.ytimg.com/vi/2Vv-BfVoq4g/default.jpg

   comments_disabled  ratings_disabled  video_error_or_removed  \
0              False             False                   False
```

```
1              False            False                    False
2              False            False                    False
3              False            False                    False
4              False            False                    False

                                          description  like/dislike ratio  \
0  Eminem's new track Walk on Water ft. Beyoncé i…               18.14
1  STill got a lot of packages. Probably will las…               75.71
2  WATCH MY PREVIOUS VIDEO  \n\nSUBSCRIBE  http…                27.35
3  I know it's been a while since we did this sho…               66.49
4   : https://ad.gt/yt-perfect\n : https://atlant…              77.51

   %_Views Resulting in Like
0                       4.59
1                      12.59
2                       4.58
3                       6.31
4                       4.87
```

[59]:
```python
df_grouped_catid = concatenated_df.groupby('category_id').sum()
```

[60]:
```python
df_grouped_catid.head()
```

[60]:
```
                    views       likes   dislikes  comment_count  \
category_id
1             27619347901   589885590   25279207       65387125
2              1661853766    45461895    2571460        5957385
10           255967088943  7227198427  294657819      620030515
15             2008474231    56601492    1503766        8103678
17            18972425164   399630743   26536025       46998109

             comments_disabled  ratings_disabled  video_error_or_removed  \
category_id
1                        536.0             519.0                    85.0
2                         73.0              75.0                     2.0
10                       236.0             231.0                    60.0
15                        94.0              88.0                     1.0
17                       513.0             448.0                     5.0

             like/dislike ratio  %_Views Resulting in Like
category_id
1                           inf                   58722.56
2                           inf                   20065.55
10                          inf                  202981.30
15                          inf                   20907.39
17                          inf                   49454.84
```

[61]:

```python
overall_views_by_cat = df_grouped_catid['views']
```

[62]:

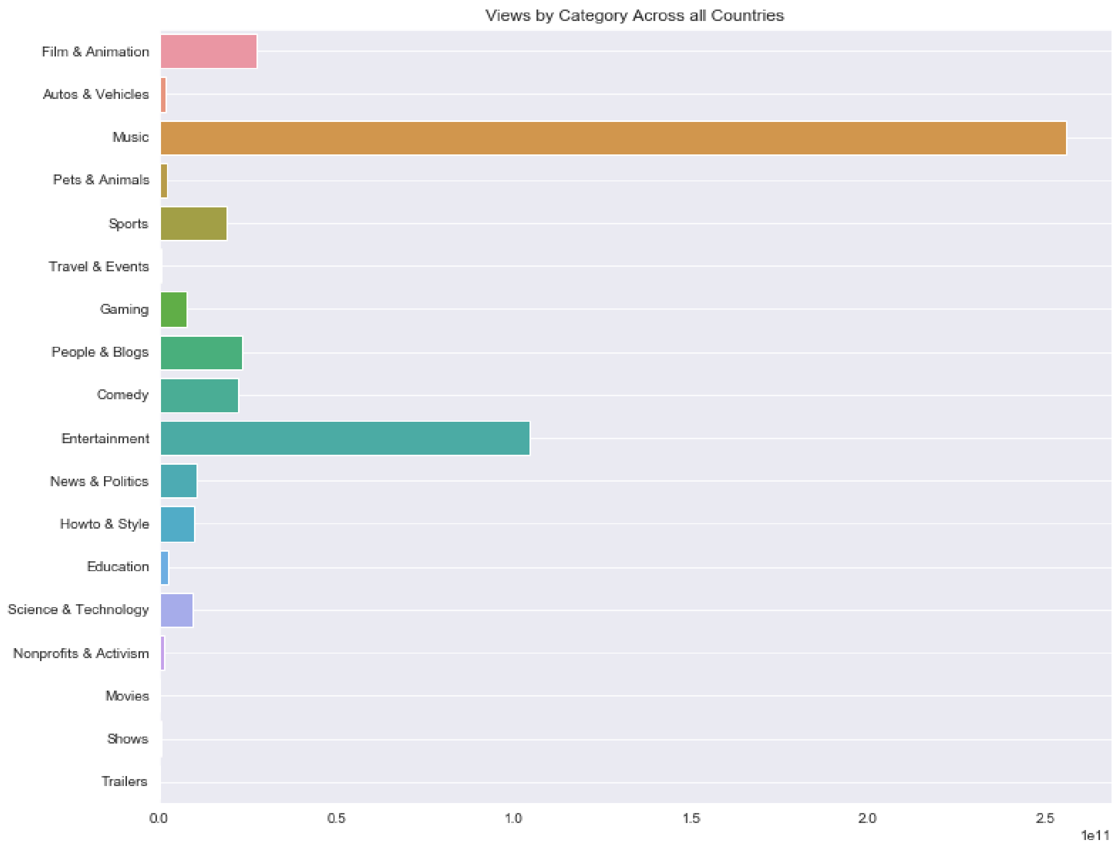
```python
overall_views_by_cat
```

[62]:

```
category_id
1       27619347901
2        1661853766
10     255967088943
15       2008474231
17      18972425164
19        726674959
20       7730729502
22      23600365409
23      22050866339
24     104517467253
25      10422502991
26       9771031927
27       2734841410
28       9194715151
29       1219859213
30         70359777
43        444064556
44            55043
Name: views, dtype: int64
```

[63]:

```python
plt.figure(figsize=(12, 10))
sns.barplot(x=list(overall_views_by_cat), y=filtered_cats).set_title('Views by␣
↪Category Across all Countries')
plt.grid()
plt.savefig('OverallViewsByCategory.png')
```



8 Predicting `category_id`

Now that we have a good idea of how statistics vary across each country, and have aggregated the data into one DataFrame, we will see if we can accurately predict the numerical `category_id` of videos based on other features in the dataset.

This is a multi-class classification problem, and so an algorithm that is suitable for such a task must be used.

8.1 Random Forest Classifier

This algorithm is naturally well suited to this problem, and is simple to implement.

8.1.1 Choosing features

The features that will be used are `likes`, `dislikes`, `views`, `comments_disabled`, and `comment_count`.

There is no guarantee that `category_id` can be successfully predicted using this data, but we will find out.

8.1.2 Selecting out the labels and features

We will be using the full concatenated dataset to select out both the labels and features.

In order to mitigate sampling bias, we will use random sampling from the full dataset. This means there are approximately 376,000 samples to choose from.

1. We will select out features from `concatenated_df` - two separate DataFrames for `features` and `labels` will be created
2. These DataFrames will then be converted to numpy arrays
3. The full dataset will be used to train and test the model - around 375,000 records - split into 70% train and 30% test sets

Selecting out the target variable

[64]:
```
all_labels = concatenated_df['category_id']
len(all_labels)
#all_labels = np.array(all_labels, dtype='int64')
all_labels = np.array(all_labels)
all_labels[:5]
```

[64]:
```
array([10, 23, 23, 24, 10], dtype=int64)
```

Selecting out the model features

[65]:
```
all_features = concatenated_df[['views', 'likes', 'dislikes',␣
↪'comments_disabled', 'comment_count']]
len(all_features)
all_features = np.array(all_features).astype(np.int64) #convert the boolean␣
↪comments_disabled column into binary
all_features[:5]
```

[65]:
```
array([[17158579,   787425,    43420,        0,   125882],
       [ 1014651,   127794,     1688,        0,    13030],
       [ 3191434,   146035,     5339,        0,     8181],
       [ 2095828,   132239,     1989,        0,    17518],
       [33523622,  1634130,    21082,        0,    85067]], dtype=int64)
```

[66]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

[67]:
```
# split the data into test and train datasets
X_train, X_test, y_train, y_test = train_test_split(all_features, all_labels,␣
↪test_size=0.3)
```

[68]:
```
len(X_train)
X_train.shape, y_train.shape
```

[68]:
```
((263159, 5), (263159,))
```

[69]:
```
X_train[:5]
```

[69]:
```
array([[284404,  17392,    520,      0,   3228],
       [200202,   9285,    441,      0,   1810],
       [  4991,     79,     24,      0,     26],
       [218516,  10231,    193,      0,   1374],
       [ 45367,   2442,     51,      0,    150]], dtype=int64)
```

8.2 Hyperparameter Tuning

8.2.1 RandomizedSearchCV

There are several hyperparameters that can be specified when training the RandomForestClassifier. It is not possible to know beforehand exactly which combination of hyperparameter settings will deliver optimal classification for these data.

To address this, sklearn has the `RandomizedSearchCV` method (one of a number of cv methods) which allows us to use **cross validation** to select the model parameters which provide the highest classification accuracy. As the name suggests, this uses "randomized search" to select random combinations of hyperparameters out of a given range specified by the programmer. The parameter `n_iter` specifies how many different combinations of parameters to try from the given lists.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

Creating the `random_grid` To implement `RandomizedSearchCV`, we must:

1. Decide which hyperparameters we want to vary
2. Define ranges for these hyperparameters
3. Add these to a dictionary that will be passed as a parameter to the `RandomizedSearchCV` method
4. We also pass the instance of the `RandomForestClassifier` we create as a parameter, and this allows us to build the model

We also must choose sensible value ranges for each of the hyperparameters (mainly balancing overfitting with classification accuracy)

[70]:
```
# creating the random_grid dictionary
from sklearn.model_selection import RandomizedSearchCV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 200, num = 10)]
# Number of features to consider at every split
max_features = [3, 4, 5]
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(2, 6, num = 3)]
# Minimum number of samples required to split a node
min_samples_split = [3, 5, 7, 9]
# Minimum number of samples required at each leaf node
min_samples_leaf = [10, 15, 20, 25]
# Method of selecting samples for training each tree
```

```
bootstrap = [True]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
```

8.3 Building model, training, and testing

[71]:
```
# instantiating the RandomForestClassifier
rf = RandomForestClassifier()
```

[72]:
```
# implementing RandomizedSearchCV for our instance of the rf classifier
rf_random = RandomizedSearchCV(estimator = rf,
                              param_distributions = random_grid, # use the␣
 ↪random grid for hyperparameters
                              n_iter=10,
                              cv = 5,
                              n_jobs = -1) # commit all compute cores
```

[73]:
```
# timing the model training
import time
t0 = time.time()

# creating the model by fitting it to the training data
model = rf_random.fit(X_train, y_train)

print("time taken= {}".format(time.time()-t0))
```

```
time taken= 1347.6702308654785
```

[74]:
```
y_pred = model.predict(X_test)
print(y_pred[:20], y_test[:20])
acc = model.score(X_train, y_train)
print(acc)
```

```
[24 24 24 24 10 24 24 24 24 22 24 24 24 24 22 24 24 24 24 22] [26 23 10 24 10 22
26 24 25 24 22 26 25 10 24 22 25 24 22 23]
0.3353333915997553
```

[75]:
```
model.best_estimator_
```

[75]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=6, max_features=4,
                       max_leaf_nodes=None, max_samples=None,
```

```
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=20, min_samples_split=3,
                        min_weight_fraction_leaf=0.0, n_estimators=200,
                        n_jobs=None, oob_score=False, random_state=None,
                        verbose=0, warm_start=False)
```

It can be seen that this has not produced an effective classifier as the accuracy is approximately only 33%.

It looks like the model is over-predicting class 24 (entertainment). It drowns out the rest of the classes and there is no significant separation or defining feature about this class relative to other classes.

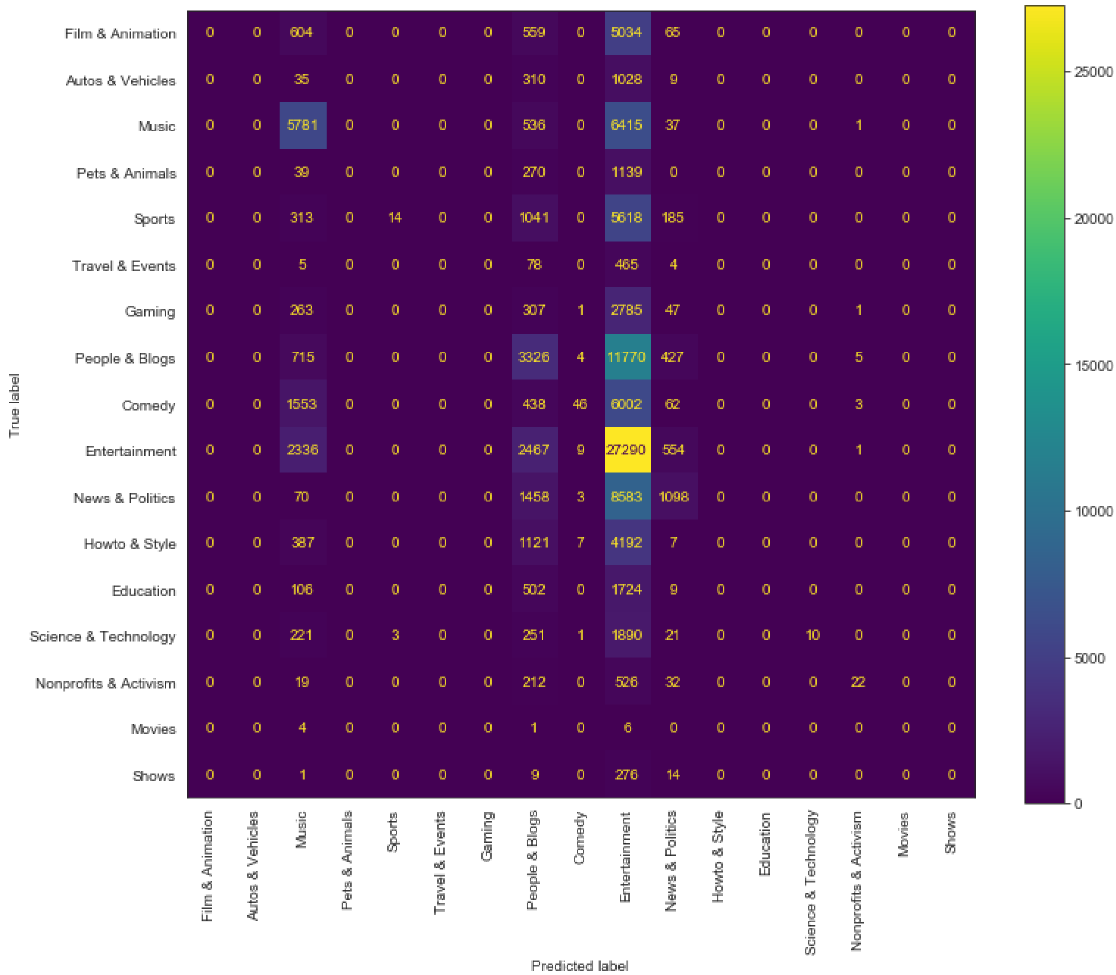
9 Understanding the Misclassifications

We can create a confusion matrix that will show the classification accuracy by category, as well as show which classes are being confused with each other.

9.1 Confusion Matrix

[76]:
```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
```

[77]:
```python
confusion = confusion_matrix(y_test, y_pred)

sns.set_style('white')
fig, ax = plt.subplots(figsize=(12,10))
cm_display = ConfusionMatrixDisplay(confusion, display_labels=filtered_cats).
 ↪plot(xticks_rotation='vertical', ax=ax, values_format=".0f")
plt.savefig('ConfusionMatrix.png')
```



The confusion matrix shows that the category "entertainment" is the one that causes most of the misclassification.

This is most likely due to there being many more videos with this `category_id` in the dataset and hence it is influencing the classifier more than the other classes.

[78]:
```python
# count the number of occurrences of each category_id in the dataset

category_id_counts = concatenated_df.groupby('category_id').count()
category_id_counts = category_id_counts['video_id']
category_id_counts
```

[78]:
```
category_id
1        20932
2         4734
10       42514
15        4863
```

```
17     23684
19      1776
20     11498
22     54052
23     26970
24    109006
25     37288
26     18856
27      7788
28      8171
29      2795
30        36
43       974
44         5
Name: video_id, dtype: int64
```

The top 5 most common `category_ids` in the dataset are (in descending order):

1. 24 - Entertainment
2. 22 - People and Blogs
3. 10 - Music
4. 25 - News and Politics
5. 23 - Comedy

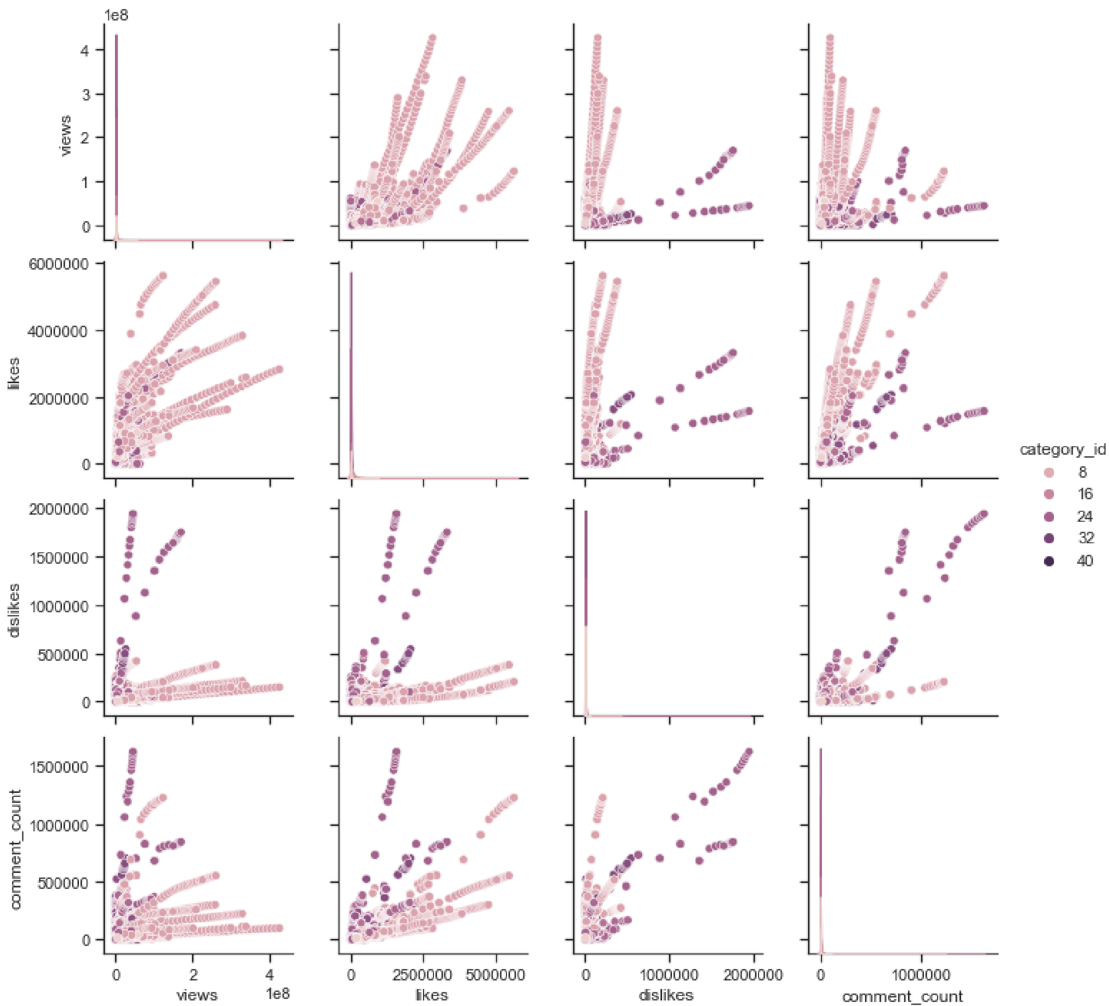
As can be seen from the confusion matrix, it is these categories which are responsible for the vast majority of the misclassification. As these `category_ids` are much more prevalent in the dataset, the classifier is predicting them more often.

The fact that this is a multiclass classification problem with **18 classes** also accounts for the low accuracy of the classifier. When there are so many classes, and few defining features upon which to split, the result is that the classes are all grouped together and there is little distinction between them.

To show this, we can plot the data.

9.2 Scatter Plots

[79]:
```
sns.set_theme(style='ticks')

sns.pairplot(concatenated_df[['views', 'likes', 'dislikes', 'comment_count',␣
 ↪'category_id']], hue='category_id')

plt.savefig('ScatterPlotsPair.png')
```



9.3 PCA Projections

Using the 5-dimensional data created a classifier that was inaccurate.

Projecting the data onto the top n eigenvectors could help to show that there is little separation between category_id's and hence help explain why the classifier is not able to successfully predict the video categories.

1. Project on to top 2 eigenvectors to make a 2-D plot and show if there is any separation between classes
2. Project on to top 3 eigenvectors and make a 3-D interactive plot to show if the data is clustered and hence difficult to separate in 3 dimensions.

If there is not significant separation between classes then this further shows that the data is not well suited to modelling as-is.

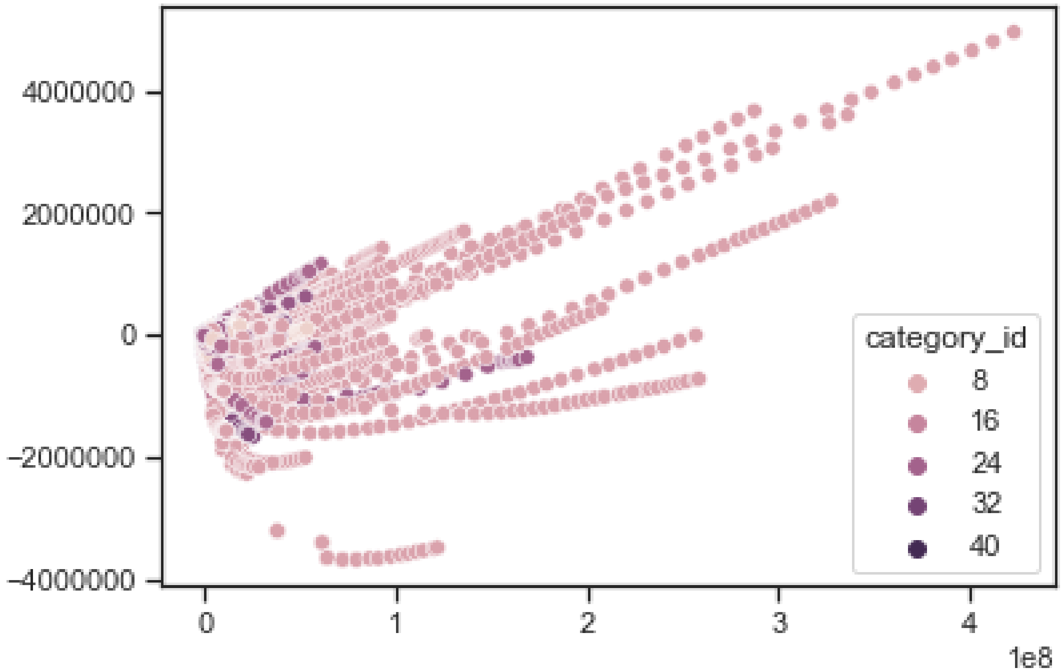
[80]:
```
n_components=3

from sklearn.decomposition import PCA
```

[81]:
```
pca = PCA(n_components=n_components).fit(all_features)
```

[82]:
```
all_features_pca = pca.transform(all_features)
all_features_pca[:2]
```

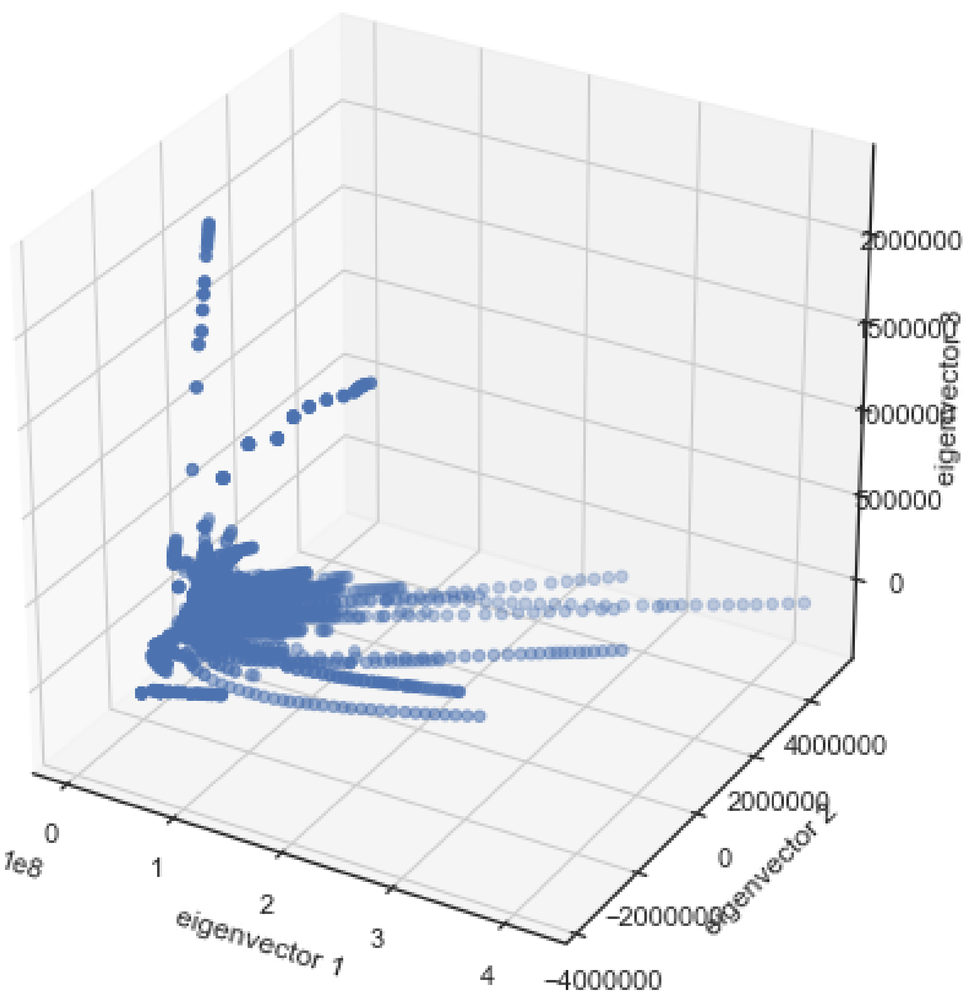
[82]:
```
array([[ 1.58432327e+07, -4.71691286e+05,  8.52596502e+03],
       [-3.10220096e+05, -9.57219087e+04, -6.97911973e+03]])
```

[83]:
```
# scatter plot of projection on to top 2 eigenvectors

sns.scatterplot(x=all_features_pca[:,0], y=all_features_pca[:,1],␣
 ↪hue=concatenated_df['category_id'])
plt.savefig('2DPCA.png')
```



[84]:
```
#%matplotlib widget
# interactive scatter plot of projection on to top 3 eigenvectors

from mpl_toolkits.mplot3d import Axes3D

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

```
ax = fig.add_subplot(111, projection='3d')

x = all_features_pca[:,0]
y = all_features_pca[:,1]
z = all_features_pca[:,2]

ax.set_xlabel('eigenvector 1')
ax.set_ylabel('eigenvector 2')
ax.set_zlabel('eigenvector 3')

ax.scatter(x, y, z)

plt.show()

plt.savefig('3DPCA.png')
```



<Figure size 432x288 with 0 Axes>

[85]:
```
!move *.png figures
```

```
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\2DPCA.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\3DPCA.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\ConfusionMatrix.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\CountryViewsByCategoryName.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\DistOfViewsWithLike.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\FranceMeanVsMedianLikes.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\LikeDislikeRatioByCountry.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\MeanLikesPerCat.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\MedianLikesPerCat.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\OverallViewsByCategory.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\PctOfViewsResultingInLike.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\ScatterPlotsPair.png
C:\Users\Stewa\Documents\All things data\Data Sets\YouTube Trending
Data\ViewsByCategoryID.png
       13 file(s) moved.
```