Capstone 1 Project Milestone Report:

YouTube Trending Videos:

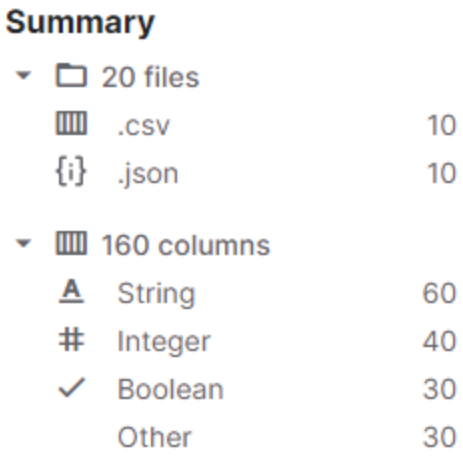
Project Goal:

This capstone project we attempt to analyze what YouTube videos are trending. Find common characteristics of trending videos and attempt to predict and reproduce results. Our target clients are companies and/or individuals who are interested in using YouTube as an avenue for content creation.

This information could be useful for these content creators who are interested in increasing their view rates.

The data we will start with is from a kaggle dataset: [Trending YouTube Video Statistics]

We have a set of 10 CSV files and 10 JSON files



Initial exploration of the data columns give us a few ideas:

```
ctry = united_states
ctry.columns
```

```
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id',
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count',
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description'],
      dtype='object')
```

We can find out which videos have the most views by using sort_values() to show which videos in descending order. We could also use value_counts() to show which category type has the

most views. This will be useful for our client/s to guide them in creating content that will get more views from the data. We could see if there are correlations between views and numbers of likes and dislikes, comment count, etc … We could create some predictive models to test whether a particular video and/or video category will get many views. We could attempt to predict exactly how many average daily views a view will get.

Data Wrangling

The original data comes from a kaggle project in the following link: Trending YouTube Video Statistics

The original data were 10 csv files organized by country and one json file with the keys to the category type for the YouTube videos. We start by reading these in:

```
canada = pd.read_csv('CAvideos.csv', encoding='latin-1')
denmark = pd.read_csv('DEvideos.csv', encoding='latin-1')
france = pd.read_csv('FRvideos.csv', encoding='latin-1')
great_britain = pd.read_csv('GBvideos.csv', encoding='latin-1')
india = pd.read_csv('INvideos.csv', encoding='latin-1')
japan = pd.read_csv('JPvideos.csv', encoding='latin-1')
south_korea = pd.read_csv('KRvideos.csv', encoding='latin-1')
mexico = pd.read_csv('MXvideos.csv', encoding='latin-1')
russia = pd.read_csv('RUvideos.csv', encoding='latin-1')
united_states = pd.read_csv('USvideos.csv', encoding='latin-1')

categories = pd.read_json('CA_category_id.json')
```

We start to explore the data looking at t:
he .columns and .head()

```
ctry = united_states
ctry.columns
```

```
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id',
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count',
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description'],
      dtype='object')
```

```
ctry.td2 = ctry.trending_date

for index, row in ctry.iterrows():
    ctry.td2[index] = ctry.td2[index][6:8] + "." + ctry.td2[index][3:5] + "." + ctry.td2[index][0:2]

ctry.head()
```

	video_id	trending_date	title	channel_title	category_id	publish_time	tags	views	likes	dislikes
0	2kyS6SvSYSE	11.14.17	WE WANT TO TALK ABOUT OUR MARRIAGE	CaseyNeistat	22	2017-11-13T17:13:01.000Z	SHANtell martin	748374	57527	2966
1	1ZAPwfrtAFY	11.14.17	The Trump Presidency: Last Week Tonight with J...	LastWeekTonight	24	2017-11-13T07:30:00.000Z	last week tonight trump presidency|"last week ...	2418783	97185	6146
2	5qpjK5DgCt4	11.14.17	Racist Superman | Rudy Mancuso, King Bach & Le...	Rudy Mancuso	23	2017-11-12T19:05:24.000Z	racist superman|"rudy"|"mancuso"|"king"|"bach"...	3191434	146033	5339

We notice there are many useful columns that such as trending and publish dates. Category ID is a number which we will need to couple with the information category type found in the json file. Other useful information is the views.

One thing we notice is the dates are in different formats. Trending date is ordered by mm.dd.yy we rearrange to yy.mm.dd before converting the string to Date Time object.

```
ctry.td2 = ctry.trending_date

for index, row in ctry.iterrows():
    ctry.td2[index] = ctry.td2[index][6:8] + "." + ctry.td2[index][3:5] + "." + ctry.td2[index][0:2]

ctry.head()
```

	video_id	trending_date	title	channel_title	category_id	publish_time	tags	views	likes	dislikes
0	2kyS6SvSYSE	11.14.17	WE WANT TO TALK ABOUT OUR MARRIAGE	CaseyNeistat	22	2017-11-13T17:13:01.000Z	SHANtell martin	748374	57527	2966
1	1ZAPwfrtAFY	11.14.17	The Trump Presidency: Last Week Tonight with J...	LastWeekTonight	24	2017-11-13T07:30:00.000Z	last week tonight trump presidency|"last week ...	2418783	97185	6146
2	5qpjK5DgCt4	11.14.17	Racist Superman | Rudy Mancuso, King Bach & Le...	Rudy Mancuso	23	2017-11-12T19:05:24.000Z	racist superman|"rudy"|"mancuso"|"king"|"bach"...	3191434	146033	5339

Now we convert to Date Time object using pd.to_datetime() and dt.date

```
ctry.trending_date = pd.to_datetime(ctry.trending_date)
```

```
ctry.trending_date = ctry.trending_date.dt.date
ctry.trending_date
```

```
0        2017-11-14
1        2017-11-14
2        2017-11-14
3        2017-11-14
4        2017-11-14
```

We do the same for the publish_time column:

```
ctry.publish_time = pd.to_datetime(ctry.publish_time)
ctry.publish_time.head()
```

```
0   2017-11-13 17:13:01+00:00
1   2017-11-13 07:30:00+00:00
2   2017-11-12 19:05:24+00:00
3   2017-11-13 11:00:04+00:00
4   2017-11-12 18:01:41+00:00
Name: publish_time, dtype: datetime64[ns, UTC]
```

```
ctry.publish_time = ctry.publish_time.dt.date
ctry.publish_time
```

```
0        2017-11-13
1        2017-11-13
2        2017-11-12
3        2017-11-13
4        2017-11-12
```

We create a new column to get time elapsed from published and trending date in order to get average daily view count for each video below:

```
ctry.time_elapse = (ctry.publish_time - ctry.trending_date).abs()
ctry.time_elapse = ctry.time_elapse.dt.days
ctry.time_elapse
```

```
0        1
1        1
2        2
3        1
4        2
```

```
av_daily_views = ctry.views / ctry.time_elapse
av_daily_views = av_daily_views.replace([np.inf, -np.inf], np.nan)
av_daily_views.dropna(inplace=True)
av_daily_views.sort_values(ascending=False).head(10)
```

```
35550    3.934993e+07
3200     3.773628e+07
35749    3.139820e+07
3400     2.818364e+07
30750    2.797321e+07
4801     2.630586e+07
5020     2.532316e+07
5236     2.522789e+07
4600     2.478216e+07
5452     2.277493e+07
dtype: float64
```

We create a new column for average daily views column to the dataframe:

```
ctry['daily_views'] = av_daily_views
ctry['time_elapse'] = ctry.time_elapse
ctry_sorted = ctry.sort_values(by=['daily_views'], ascending=False)
ctry_sorted.head(10)
```

...s	comment_count	thumbnail_link	comments_disabled	ratings_disabled	video_error_or_removed	description	daily_views	time_elapse
...46	905912	https://i.ytimg.com/vi/7C2z4GqqS5E/default.jpg	False	False	False	BTS (ë°©íŒ□□ì□□ë□ □ë□¨) 'FAKE LOVE' Official MVD...	2.093213e+07	3
...47	682890	https://i.ytimg.com/vi/FlsCjmMhFmw/default.jpg	False	False	False	YouTube Rewind 2017. Celebrating the videos, p...	2.018231e+07	5
...'07	692305	https://i.ytimg.com/vi/7C2z4GqqS5E/default.jpg	False	False	False	BTS (ë°©íŒ□□ì□□ë□ □ë□¨) 'FAKE LOVE' Official MVD...	1.967496e+07	2

This is a good start. We want to look at the category_id column and the corresponding data from the json file as we mentioned before. We note that as we do a value count we see:

```
ctry.category_id.value_counts()
```

```
24    9964
10    6472
26    4146
23    3457
22    3210
25    2487
28    2401
1     2345
17    2174
27    1656
15     920
20     817
19     402
2      384
29      57
43      57
Name: category_id, dtype: int64
```

We may want to pick from the 1st 5 or 10 categories if we want to produce a video that will get the most views. Certainly, we'll avoid category #43 whatever it may be.

For this we will have to look at the json file to give us a clue to what these categories are by their category id numbers.

Exploring the JSON file

```
with open ('CA_category_id.json') as f:
    data = json.load(f)

data
```

```
{'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',
```

We can see the categories are assigned pairing each ‘id’ and ‘title’.

We create a dict() to hold this pairing and convert to dict() to a series and the series to a panda dataframe:

```
category_dict = dict()

for item in data['items']:
    idnum = item['id']
    title = item['snippet']['title']
    category_dict[idnum] = title

s = pd.Series(category_dict, name = 'title')
s.index.name = 'id'

categories_df = pd.DataFrame(s)
categories_df.head()
```

id	title
1	Film & Animation
2	Autos & Vehicles
10	Music
15	Pets & Animals
17	Sports

Finally we clean up this data get the category_id #'s alongside the title of the category:

```
c3 = c2.rename(columns={'id2_x':'category_id'})
c3 = c3.drop_duplicates()
c3.sort_values(by=['category_id'])
c3
```

	category_id	title
0	1	Film & Animation
1	2	Autos & Vehicles
2	10	Music
3	15	Pets & Animals
4	17	Sports
5	18	Short Movies
6	19	Travel & Events
7	20	Gaming
8	21	Videoblogging
9	22	People & Blogs
10	23	Comedy
12	34	Comedy
14	24	Entertainment
15	25	News & Politics
16	26	Howto & Style
17	27	Education
18	28	Science & Technology
19	30	Movies
20	31	Anime/Animation
21	32	Action/Adventure
22	33	Classics
23	35	Documentary
24	36	Drama
25	37	Family
26	38	Foreign
27	39	Horror
28	40	Sci-Fi/Fantasy
29	41	Thriller
30	42	Shorts
31	43	Shows
32	44	Trailers

We note that category #24, getting the highest viewcount is the Entertainment category and #10 is Music, so if you plan to do anything related to the entertainment or music industry, it should gather the most viewcount. While #43 is called "Shows" - it may be a good one to avoid.

Trending YouTube Videos

The YouTube trending video data is a rich source to study the making of a trending video. YouTube, being a global phenomenon, makes this study a worthwhile pursuit.

Just starting with a look at the data. We can see there are some variables of interest.

```
ctry.describe()
```

	category_id	views	likes	dislikes	comment_count	daily_views	time_elapse
count	40949.000000	4.094900e+04	4.094900e+04	4.094900e+04	4.094900e+04	4.082800e+04	40949.000000
mean	19.972429	2.360785e+06	7.426670e+04	3.711401e+03	8.446804e+03	4.067646e+05	16.810423
std	7.568327	7.394114e+06	2.288853e+05	2.902971e+04	3.743049e+04	1.095361e+06	146.014303
min	1.000000	5.490000e+02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000
25%	17.000000	2.423290e+05	5.424000e+03	2.020000e+02	6.140000e+02	4.760675e+04	3.000000
50%	24.000000	6.818610e+05	1.809100e+04	6.310000e+02	1.856000e+03	1.300480e+05	5.000000
75%	25.000000	1.823157e+06	5.541700e+04	1.938000e+03	5.755000e+03	3.518692e+05	9.000000
max	43.000000	2.252119e+08	5.613827e+06	1.674420e+06	1.361580e+06	3.934993e+07	4215.000000

We have total view count as well as daily view count. There's also a time elapse from the date of upload. The daily view count is derived from the total view divided by time elapsed. We found that there is a flaw in this method because although the time elapse is 4000 days, the video could only be trending for only 9 days and the daily view count is skewed. This deserves a more in depth look to get more reliable numbers for these columns.

Some of the **questions** we want to answer are:

1. What video types / categories will have the most view counts?

2. What view count can we expect for each video type/category?

3. Who has the most view counts?

```
ctry['category_id'].value_counts()
```

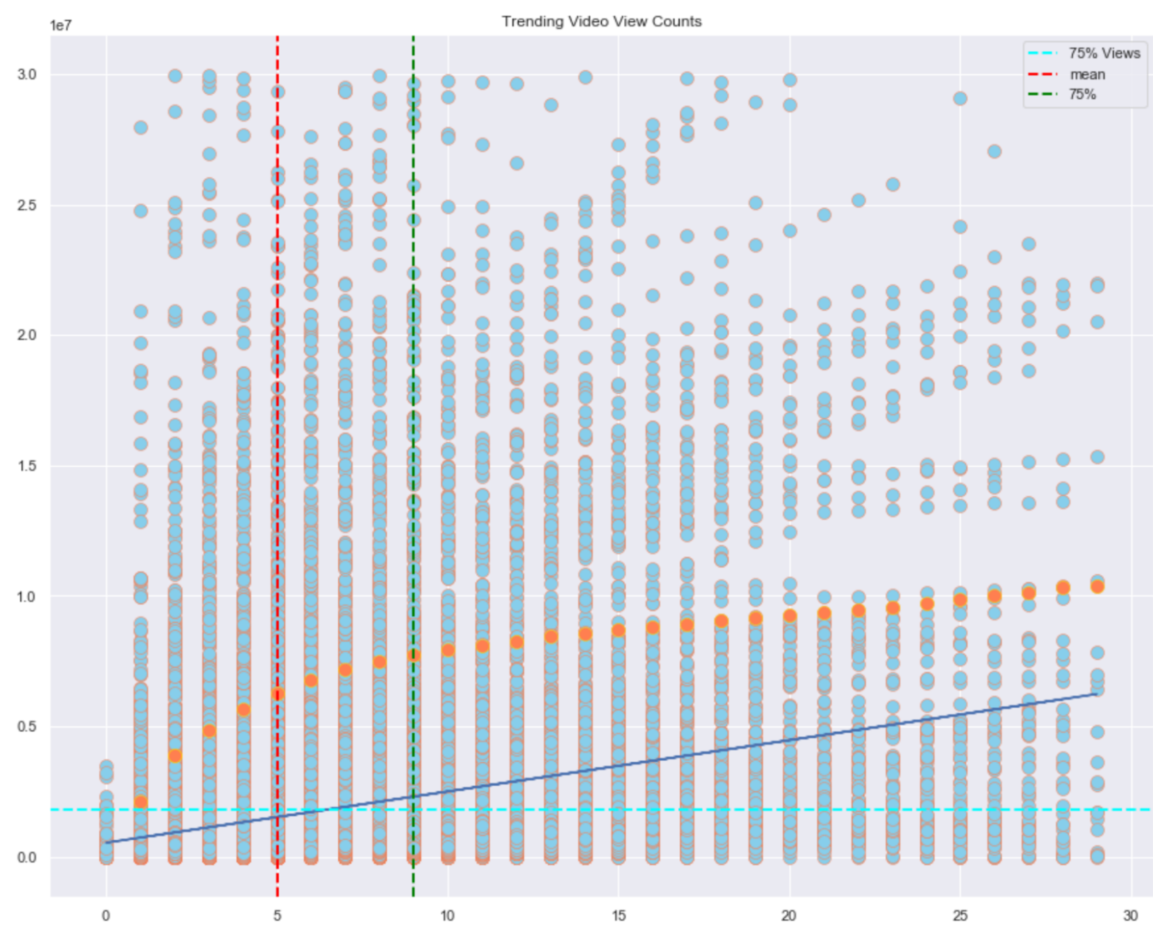
```
24    9964
10    6472
26    4146
23    3457
22    3210
25    2487
28    2401
1     2345
17    2174
27    1656
15     920
20     817
19     402
2      384
29      57
43      57
Name: category_id, dtype: int64
```

To the left we see categories that have the most trending videos. We could see categories #24 (Entertainment) and #10 (Music) occurred the most.

Here are the top videos with more that 15 million daily views. It's pretty staggering to think that it's 15 million views in a single day! That's the the rate of 173.6 views every second.

	title	category_id	daily_views	channel_title
35550	BTS (ë°©í□□ì□□ë□□ë□¨) 'FAKE LOVE' Official MV	10	39349927.0	ibighit
3200	Marvel Studios' Avengers: Infinity War Officia...	24	37736281.0	Marvel Entertainment
30750	VENOM - Official Trailer (HD)	24	27973210.0	Sony Pictures Entertainment
4600	YouTube Rewind: The Shape of 2017 | #YouTubeRe...	24	24782158.0	YouTube Spotlight
16181	To Our Daughter	22	20921796.0	Kylie Jenner
24156	Marvel Studios' Avengers: Infinity War - Offic...	24	19716689.0	Marvel Entertainment
30752	Sanju | Official Teaser | Ranbir Kapoor | Rajk...	24	18639195.0	FoxStarHindi
801	Luis Fonsi, Demi Lovato - Ã□chame La Culpa	10	18558186.0	LuisFonsiVEVO
5001	Jurassic World: Fallen Kingdom - Official Trai...	24	18184886.0	Universal Pictures
39149	we broke up	22	16884972.0	David Dobrik
29951	Ariana Grande - No Tears Left To Cry	10	15873034.0	ArianaGrandeVevo

75% of the trending videos trending are in the 1st 9 days. 75% of the view counts are about 200 thousand or less daily views.



Techniques and Steps:

- We will be looking to use a combination of Frequentist and Bootstrap Inferential techniques and perhaps some Bayesian Inferential techniques to gain some insight from our data.

- We will want to go through video titles and descriptions to additionally look at words or description categories that generate the most views.

- From these we could then try to quantify elements that produce videos that will generate more views.