TOBIAS & ASHISHA LLC

PROPOSAL TO ESTABLISH OUR CLIENTS REACH ON YOUTUBE

For Kyunghee Yoon – Rising Star in Music World

OVERVIEW

We are 20-year veterans in the Music Analytics industry who offer young, rising stars like yourself an opportunity to establish and be recognized in the world of YouTube, possibly even beyond. We have had success with stars like PSY, Taylor Swift, Drake and many such music sensations who once used our creative directives and strategies that lead them to where they are today.

Tobias & Ashisha LLC is pleased to offer insights into this seemingly arbitrary world of YouTube video trends. We aim to help you, our client, determine the path and actions needed to take to be successful in the long run. We will use data from 2017 & 2018 YouTube videos that are trending from US alone.

These data will be used for Descriptive Analysis and Diagnostic Analysis to chart trends and pull out information that you as a Musician would love to know about what the viewers would love to see and hear.

The Data

You will be pleased to hear that our source data[1] is sufficiently comprehensive and accurate. With 15 Columns and 40,949 attributes. Even though there are only 5 Integer values (int64), our company's expertise in accurately deciphering data from other object type data such dates, title and channel title. Our data can be found here[2]. Here are some of the things we will do:

- Correlation between Likes, Dislikes, Views and Comment Count
- Segregate data into different Categories and Industries and further analyze
- No. of days the video took to get into trending
- No. of views per video by each category, especially Music
- Provide insights on long term success vs short term popularity

```
#few null entries in the DESCRIPTION Column and in TAGS Column
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40949 entries, 0 to 40948
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   video_id                40949 non-null  object
 1   trending_date           40949 non-null  object
 2   title                   40949 non-null  object
 3   channel_title           40949 non-null  object
 4   category_id             40949 non-null  int64
 5   publish_time            40949 non-null  object
 6   tags                    40674 non-null  object
 7   views                   40949 non-null  int64
 8   likes                   40949 non-null  int64
 9   dislikes                40949 non-null  int64
 10  comment_count           40949 non-null  int64
 11  thumbnail_link          40949 non-null  object
 12  comments_disabled       40949 non-null  bool
 13  ratings_disabled        40949 non-null  bool
 14  video_error_or_removed  40949 non-null  bool
 15  description             40379 non-null  object
dtypes: bool(3), int64(5), object(8)
memory usage: 4.2+ MB
```

Table 1 – YouTube Data set

The Objective

As a rising star like yourself, you will want to know that we have the data covered from several industries: Comedy, Education, Entertainment, Film/Animation, How to & Style, Music, News/Politics, Blogs, Science/Tech and finally, Sports.

Miss Kyunghee Yoon, we understand the plight of an artist like yourself, who is now facing distress in this COVID-19 situation and facing lockdown. But like many others, where one door shuts, another golden opportunity opens in the social media industry.

An artist's digital presence has never been more important than it is today. We can also project how much you can make while producing content and being popular. While YouTube is very secretive about how much money they pay creators per view, they have confirmed that they use an RPM (Revenue per Mile) system to calculate[3].

One thing of note is that according to YouTube's monetization policy *"channels must have 1,000 subscribers and 4,000 watch hours in the previous 12-month period in order to be eligible…"* (Academy, n.d.) However, the amount of money that someone would be making with 1,000 subscribers and 4,000 watch hours is not enough to live off, and prospective clients such as yourself who would be in a situation to use our services should have already passed that threshold for this conversation to even be relevant to them.

Our Approach

Now that you almost at the edge of your seat and want to know how we are doing this. Its time to show you the Magicians trick. While the data that we showed above in Table 1 has many data. These Data consists of information for video that made it to the 'Trending' tab on YouTube. We believe that this will still be a good indicator for what our channels and content will be successful on the platform, especially if we can see any trends at the channel level within or maybe even across industries.

Besides removing null values and videos that were published outside of our time frame (videos that were published in 2016 and before but are still trending in year 2017 and 2018). Furthermore, certain insights may be industry specific. For instance, the upload schedule of a successful gaming YouTube channel may be unfeasible for an education or comedy YouTube channel. For now, we will be grouping our data by industry to preserve any nuances that may appear.

REFERENCES

Academy, C. (n.d.). Retrieved from Creators Academy: https://creatoracademy.youtube.com/page/lesson/m10n-analytics#strategies-zippy-link-1 (https://creatoracademy.youtube.com/page/lesson/m10n-analytics#strategies-zippy-link-1)

Kaggle. (n.d.). Trending Youtube video Statistics. Retrieved from: https://www.kaggle.com/datasnaek/youtube-new (https://www.kaggle.com/datasnaek/youtube-new)

Google. (n.d.). Google Support for YouTube. Retrieved from: https://support.google.com/youtube/answer/9314357?hl=en (https://support.google.com/youtube/answer/9314357?hl=en)

In [1]:
```
################
### IMPORT  ###
################

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf

import warnings
from collections import Counter
import datetime
from pathlib import Path

import matplotlib as mpl
from matplotlib import pyplot as plt
from collections import Counter

import json
```

In [2]:
```
################
### Set Path ###
################

df = pd.read_csv(r"C:\Users\tobys\Downloads\YouTube\USvideos.csv")
```

In [3]:
```
########################
### Category Naming ###
########################

df['category_name'] = np.nan

df.loc[(df["category_id"] == 1),"category_name"] = 'Film and Animation'
df.loc[(df["category_id"] == 2),"category_name"] = 'Cars and Vehicles'
df.loc[(df["category_id"] == 10),"category_name"] = 'Music'
df.loc[(df["category_id"] == 15),"category_name"] = 'Pets and Animals'
df.loc[(df["category_id"] == 19),"category_name"] = 'Travel and Events'
df.loc[(df["category_id"] == 20),"category_name"] = 'Gaming'
df.loc[(df["category_id"] == 22),"category_name"] = 'People and Blogs'
df.loc[(df["category_id"] == 23),"category_name"] = 'Comedy'
df.loc[(df["category_id"] == 24),"category_name"] = 'Entertainment'
df.loc[(df["category_id"] == 25),"category_name"] = 'News and Politics'
df.loc[(df["category_id"] == 26),"category_name"] = 'How to and Style'
df.loc[(df["category_id"] == 27),"category_name"] = 'Education'
df.loc[(df["category_id"] == 28),"category_name"] = 'Science and Technology'
df.loc[(df["category_id"] == 29),"category_name"] = 'Non Profits and Activism'
df.loc[(df["category_id"] == 25),"category_name"] = 'News & Politics'
```

In [4]:
```
df.head()
```

Out[4]:

	video_id	trending_date	title	channel_title	category_id	publish_time	
0	2kyS6SvSYSE	17.14.11	WE WANT TO TALK ABOUT OUR MARRIAGE	CaseyNeistat	22	2017-11-13T17:13:01.000Z	
1	1ZAPwfrtAFY	17.14.11	The Trump Presidency: Last Week Tonight with J...	LastWeekTonight	24	2017-11-13T07:30:00.000Z	
2	5qpjK5DgCt4	17.14.11	Racist Superman | Rudy Mancuso, King Bach & Le...	Rudy Mancuso	23	2017-11-12T19:05:24.000Z	sup
3	puqaWrEC7tY	17.14.11	Nickelback Lyrics: Real or Fake?	Good Mythical Morning	24	2017-11-13T11:00:04.000Z	
4	d380meD0W0M	17.14.11	I Dare You: GOING BALD!?	nigahiga	24	2017-11-12T18:01:41.000Z	

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40949 entries, 0 to 40948
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   video_id                40949 non-null  object
 1   trending_date           40949 non-null  object
 2   title                   40949 non-null  object
 3   channel_title           40949 non-null  object
 4   category_id             40949 non-null  int64 
 5   publish_time            40949 non-null  object
 6   tags                    40949 non-null  object
 7   views                   40949 non-null  int64 
 8   likes                   40949 non-null  int64 
 9   dislikes                40949 non-null  int64 
 10  comment_count           40949 non-null  int64 
 11  thumbnail_link          40949 non-null  object
 12  comments_disabled       40949 non-null  bool  
 13  ratings_disabled        40949 non-null  bool  
 14  video_error_or_removed  40949 non-null  bool  
 15  description             40379 non-null  object
 16  category_name           38718 non-null  object
dtypes: bool(3), int64(5), object(9)
memory usage: 4.5+ MB
```

Observation:

We can see that the file has 40,949 entries in the dataset and there are a few null entries in the DESCRIPTION Column and in TAGS Column. There are blanks in Column : video_id, tags and description.These blanks do not affect the intergrity or analysis of the data

To find Null Data

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
video_id                     0
trending_date                0
title                        0
channel_title                0
category_id                  0
publish_time                 0
tags                         0
views                        0
likes                        0
dislikes                     0
comment_count                0
thumbnail_link               0
comments_disabled            0
ratings_disabled             0
video_error_or_removed       0
description                570
category_name             2231
dtype: int64
```

Display Data in Time Series and Analysis

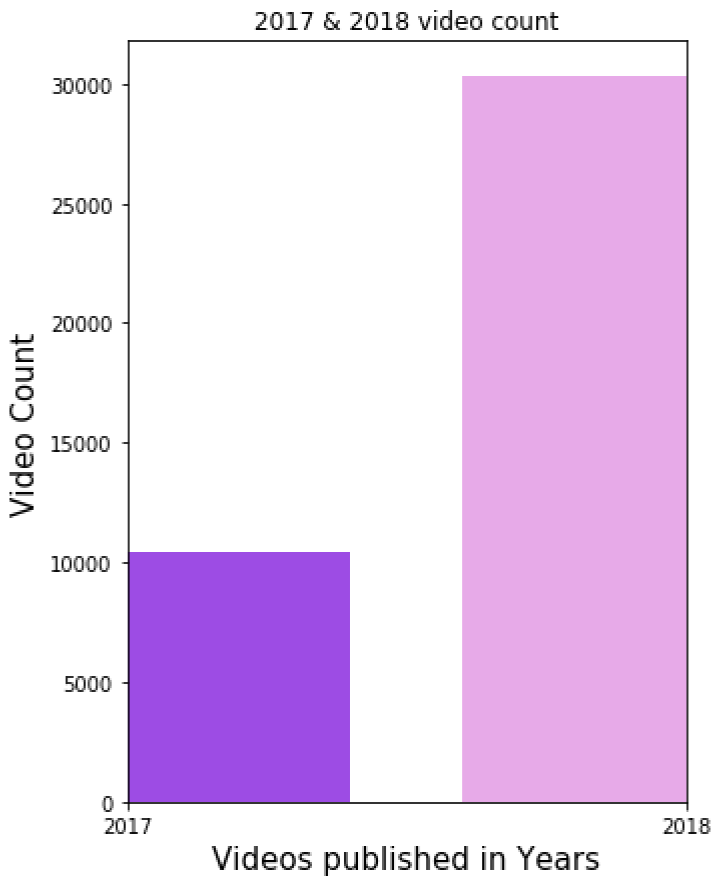
Data Cleansing of the Date Time series in "publish_time"

In [7]:
```
# To Seprate the 'publish_time' field into yyyy mm dd
YMD = pd.to_datetime(df["publish_time"])
US_video = df.assign(
publish_day = YMD.dt.day,
publish_month = YMD.dt.month,
publish_year = YMD.dt.year
)

publish_day = df["publish_time"].apply(lambda x: datetime.datetime.strptime(x
[:10], "%Y-%m-%d").date().strftime('%a'))
publish_hour = df["publish_time"].apply(lambda x: x[11:13])
```

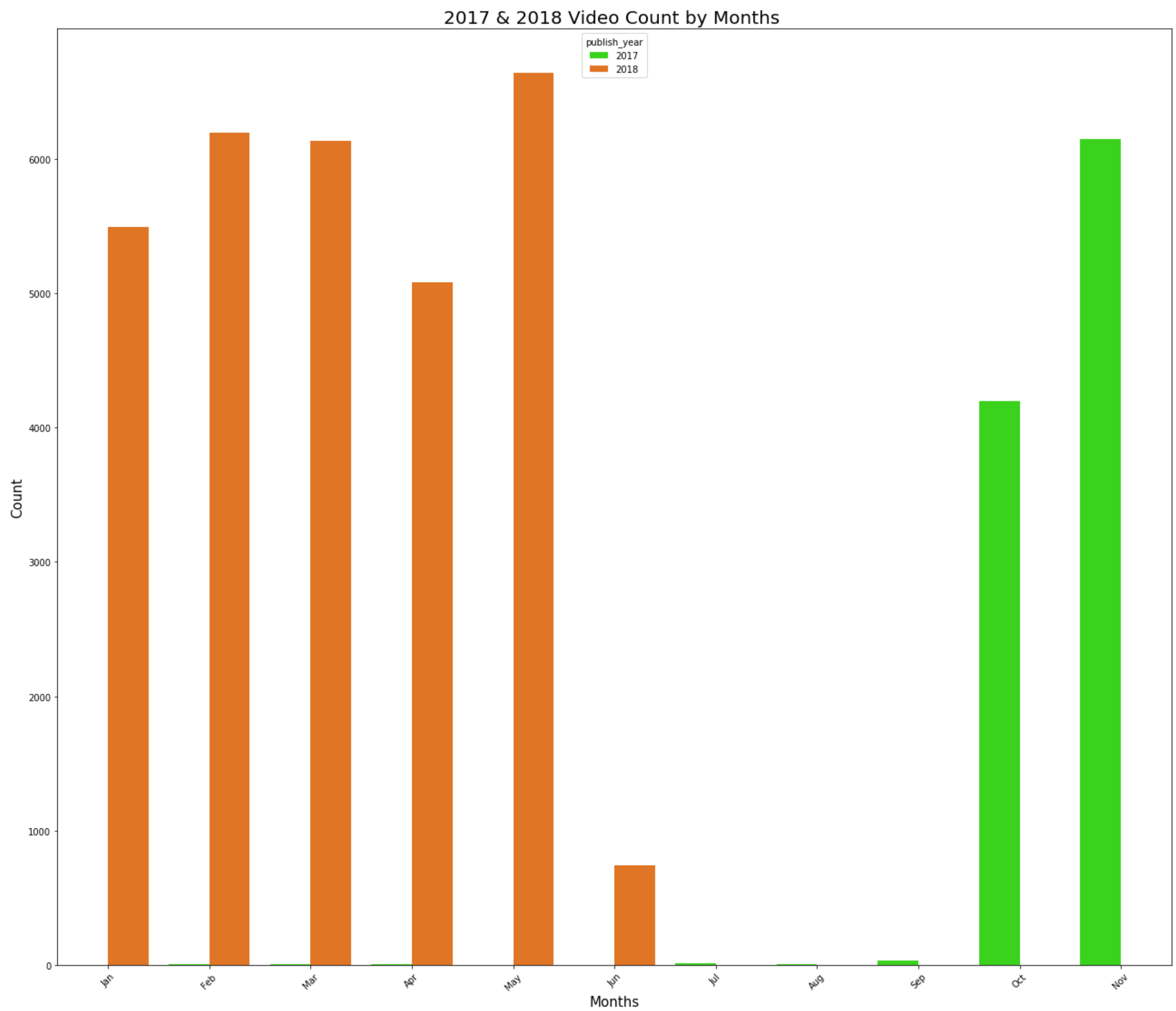
Bar chart showing Number of Videos published in 2017 and 2018 alone

In [8]:
```
plt.figure(figsize=(5,7))
sns.countplot(x = US_video["publish_year"],data = df,palette="gist_ncar")
plt.title("2017 & 2018 video count")
plt.xlim(10,11) ## these show only 2017 and 2018 years
plt.xlabel("Videos published in Years", fontsize=15)
plt.ylabel("Video Count", fontsize=15)
plt.show()
```



In [10]:
```
dfnew = US_video.loc[(US_video['publish_year']==2017) | (US_video['publish_year']==2018)]
xlabels = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']

plt.figure(figsize = (22,19))
g = sns.countplot( x = dfnew['publish_month'],hue='publish_year',data = dfnew, palette="gist_ncar")
g.set_xticklabels(xlabels,rotation=45)
g.set_title("2017 & 2018 Video Count by Months ", fontsize=20)
g.set_xticklabels(xlabels)
g.set_xlabel("Months", fontsize=15)
g.set_ylabel("Count", fontsize=15)
plt.show()
```

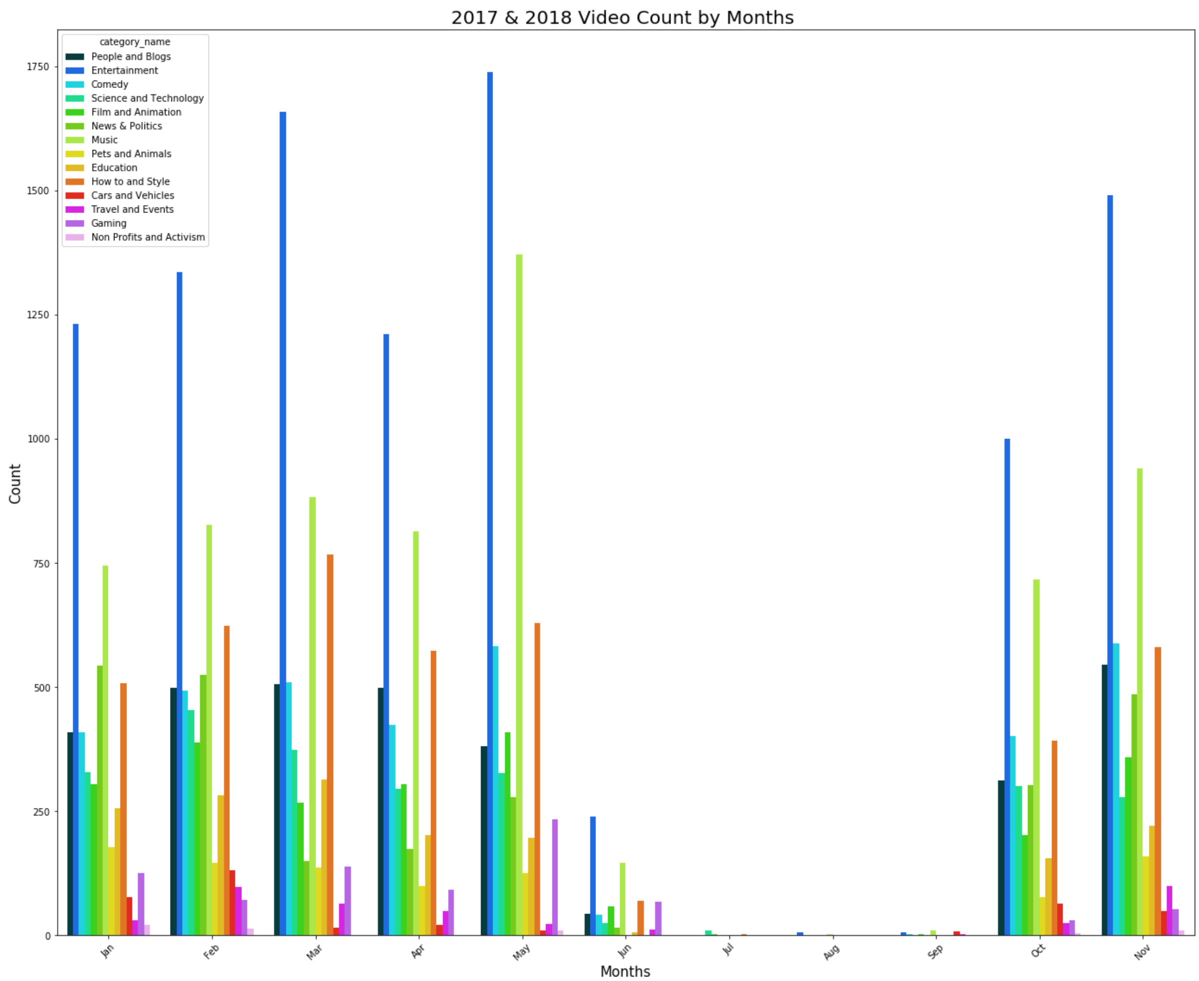


Observation:

This graph shows of all the videos that where published in 2017 and 2018 and made it to the trending in 2017 and 2018. This way we are comparing like vs like, to see how long the videos take to make it to the trending from the time they are published.

Bar plot of data into their categories across the months for videos published in 2017 and 2018

In [11]:
```
plt.figure(figsize = (22,18))
g = sns.countplot( x = dfnew['publish_month'],hue='category_name',data = US_vi
deo, palette="gist_ncar")
g.set_xticklabels(xlabels,rotation = 45)
g.set_title("2017 & 2018 Video Count by Months ", fontsize=20)
g.set_xlabel("Months", fontsize=15)
g.set_ylabel("Count", fontsize=15)
plt.show()
```



Observation:

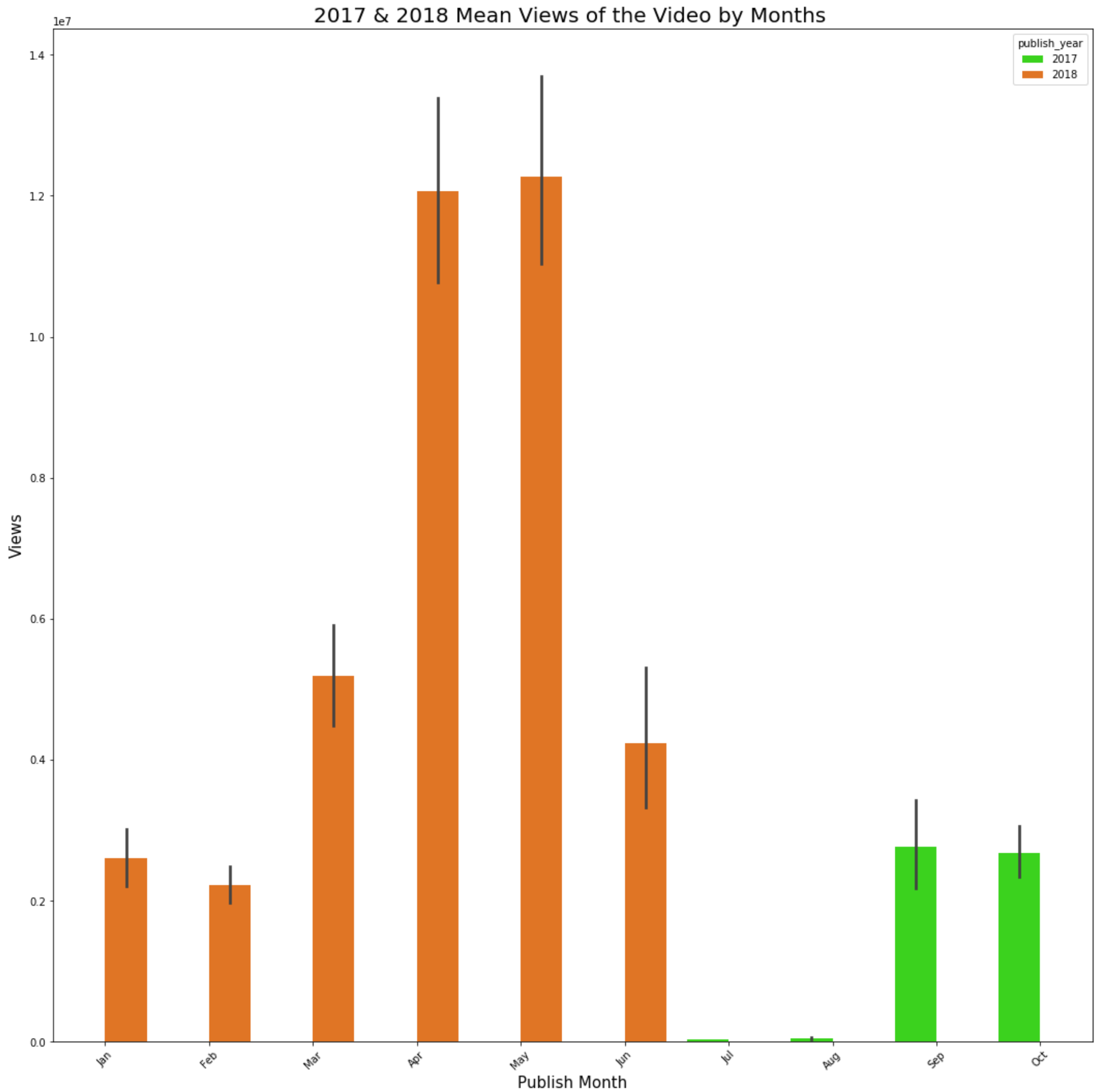
Looking at the MUSIC barchart, we can see that MUSIC video that got published in 2017 started increasing in mumber of videos that got uploaded since October 2017 and kept increasing in video count into 2018. In May 2018, there was the highest number of video upload among all the months.

Plotting Mean Number of Views against the Publish Month for MUSIC Category

In [12]:
```
dfnew = US_video.loc[(US_video['publish_year']==2017) | (US_video['publish_yea
r']==2018)]
dfnew_Music = dfnew.loc[(dfnew['category_id']==10)]

df_US_video_views=dfnew.groupby(['views']).mean()
df_US_video_likes=dfnew.groupby(['likes']).mean()

plt.figure(figsize = (18,18))
f = sns.barplot(data = dfnew_Music, x = dfnew_Music['publish_month'],y = 'view
s',hue = 'publish_year',palette="gist_ncar")
f.set_xticklabels(xlabels,rotation = 45)
f.set_title("2017 & 2018 Mean Views of the Video by Months ", fontsize=20)
f.set_xlabel("Publish Month", fontsize=15)
f.set_ylabel("Views", fontsize=15)
plt.show()
```



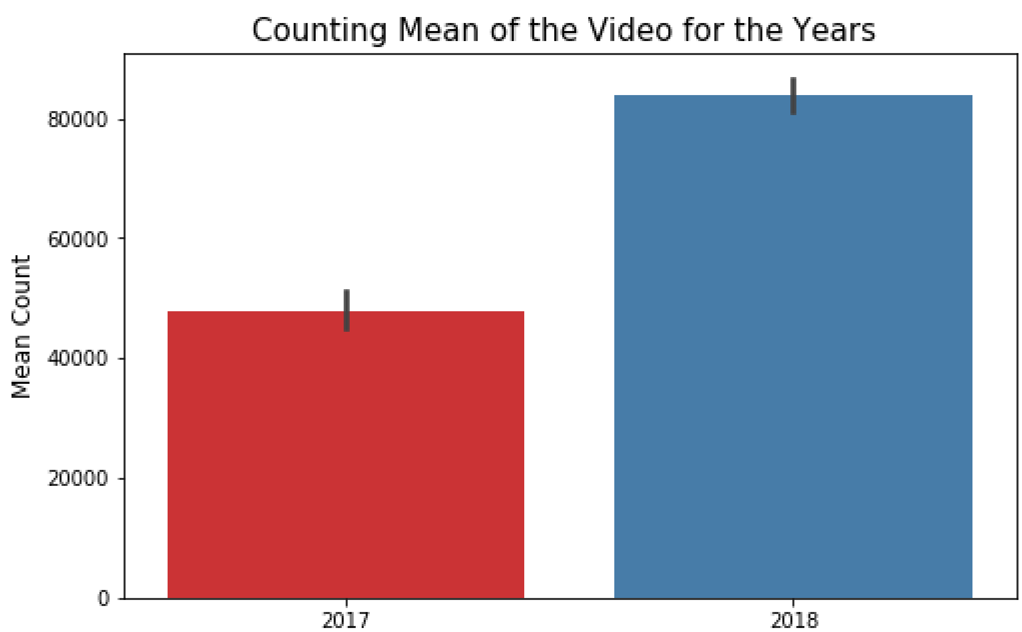
Average No. of Videos published in 2017 and 2018

In [13]:
```
## Dropping years from 2006 to 2016 from the count as it not the focus of our
 analysis
US_video.drop(US_video[US_video["publish_year"]<2017].index,inplace=True)
```

In [14]:
```
plt.figure(figsize = (8,5))
US_video.groupby("publish_year")["likes","dislikes","views","comment_count"].m
ean()
PublishYear = sns.barplot(x = US_video["publish_year"],y = df["likes"],palette
="Set1")

PublishYear.set_title("Counting Mean of the Video for the Years ", fontsize=15
)
PublishYear.set_xlabel("", fontsize=12)
PublishYear.set_ylabel("Mean Count", fontsize=12)
plt.show()
```

```
C:\Users\tobys\anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWar
ning: Indexing with multiple keys (implicitly converted to a tuple of keys) w
ill be deprecated, use a list instead.
```



Statistical Description - How many days did it take for a video to make it to trending from the time it was published

In [15]:
```
# Transform trending_date to datetime date format
df['trending_date'] = pd.to_datetime(df['trending_date'], format='%y.%d.%m').d
t.date
df.trending_date.value_counts().sort_index(inplace=True)

# Transform publish_time to datetime
publish_time = pd.to_datetime(df.publish_time, format='%Y-%m-%dT%H:%M:%S.%fZ')

# Create Variable publish_date
df['publish_date'] = publish_time.dt.date
```

In [16]:
```
# Create New Variable Counting Days to Achieving Trending Status
df['days_to_trending'] = (df.trending_date - df.publish_date).dt.days
df['days_to_trending'].describe()
```

Out[16]:
```
count    40949.000000
mean        16.810423
std        146.014303
min          0.000000
25%          3.000000
50%          5.000000
75%          9.000000
max       4215.000000
Name: days_to_trending, dtype: float64
```

Observation:

We can tell that, on Average, videos took appox 17 days to make it to Trending from the time they were published. This is very important finding for our client. The Median of the data suggest the No. of days to make it to Trend is 5 days, which is a huge departure from the Mean Data. This suggests the data is positively Skewed. This could be because the data has varying range with vast range of views.

None the less, this is important information to present to our client of the days to trend once she decides to upload a video, given the right mix of content.

In [17]:
```
DFFF = df.loc[(df.category_id == 10) & (df.views > 1000000) & (df.days_to_tren
ding <= 17) & (df.likes > 0)]
## categorising for views of 1M and above for 'days_to_trending' 17 days or le
ss and
##eliminate videos that had 0 likes(these videos are usually LIVE Telecast vid
eos such as MUSIC Festival)
g = DFFF[['days_to_trending', 'category_id','title','channel_title','views','l
ikes','dislikes']]
g.sort_values(by=['days_to_trending','views'], ascending=False)
```

Out[17]:

	days_to_trending	category_id	title	channel_title	views	likes
36505	17	10	Childish Gambino - This Is America (Official V...	ChildishGambinoVEVO	173478072	4360121
33339	17	10	Ariana Grande - No Tears Left To Cry	ArianaGrandeVevo	112904452	2875001
33334	17	10	Becky G, Natti Natasha - Sin Pijama (Official ...	BeckyGVEVO	94016241	1214283
31143	17	10	TWICE What is Love? M/V	jypentertainment	67478328	1315733
31726	17	10	The Weeknd - Call Out My Name (Official Video)	TheWeekndVEVO	62934266	1110336
...	...	...	...	...	...	...
13002	1	10	NF - NO NAME	NFVEVO	1029332	104232
29956	1	10	The Chainsmokers, Drew Love - Somebody (Offici...	ChainsmokersVEVO	1023661	101975
13046	1	10	BTS Exclusive Interview #BTSonBBCR1	BBC Radio 1	1007920	129819
5042	1	10	G-Eazy - Sober (Audio) ft. Charlie Puth	GEazyMusicVEVO	1006354	81200
37949	0	10	Maroon 5 - Girls Like You ft. Cardi B	Maroon5VEVO	3057987	406604

3292 rows × 7 columns

Observation:

There are a total of 40 videos here that under Music Category and have made it Trending in 17 days(Average 'days_to_trending) and with 10M views.

And close to 98 videos (not shown here) that made it to the Trending in 17 days(Average 'days_to_trending) and with 10M views.

Above all, there are total of 3292 videos that took 17 days or less to get 1M views on their Channel.

In Conclusion, our analysis shows that video that have original music and are part of music industry can easily get the influence on their YouTube Channel with millions of views.

Detailed views on the non-numerical values

In [18]:
```
df.describe(include = ['O'])
```

Out[18]:

	video_id	trending_date	title	channel_title	publish_time	tags	
count	40949	40949	40949	40949	40949	40949	
unique	6282	205	6455	2207	6269	6055	
top	#NAME?	2017-12-22	WE MADE OUR MOM CRY...HER DREAM CAME TRUE!	ESPN	2018-05-18T14:00:04.000Z	[none]	https://i.ytimg.c
freq	397	200	30	203	50	1535	

Observation:

From the table above, we can see that there are 205 unique dates under the 'trending_date', which means that our dataset contains collected data about trending videos over 205 days.

From video_id description, we can see that there are 40,552 videos (which is expected because our dataset contains 40949 entries), but we can see also that there are only 6281 unique videos which means that some videos appeared on the trending videos list on more than one day. The table also tells us that the top frequent title is 'WE MADE OUR MOM CRY...HER DREAM CAME TRUE!' and that it appeared 30 times on the trending videos list.

But there is something strange in the description table above: Because there are 6281 unique video IDs, we expect to have 6281 unique video titles also, because we assume that each ID is linked to a corresponding title. One possible interpretation is that a trending video had some title when it appeared on the trending list, then it appeared again on another day but with a modified title. Similar explaination applies for description column as well. For publish_time column, the unique values are less than 62811, but there is nothing strange here, because two different videos may be published at the same time.

To verify our interpretation for title column, let's take a look at an example where a trending video appeared more than once on the trending list but with different titles

In [19]:
```
grouped = df.groupby("video_id")
groups = []
wanted_groups = []
for key, item in grouped:
    groups.append(grouped.get_group(key))

for g in groups:
    if len(g['title'].unique()) != 1:
        wanted_groups.append(g)

wanted_groups[0]
```

Out[19]:

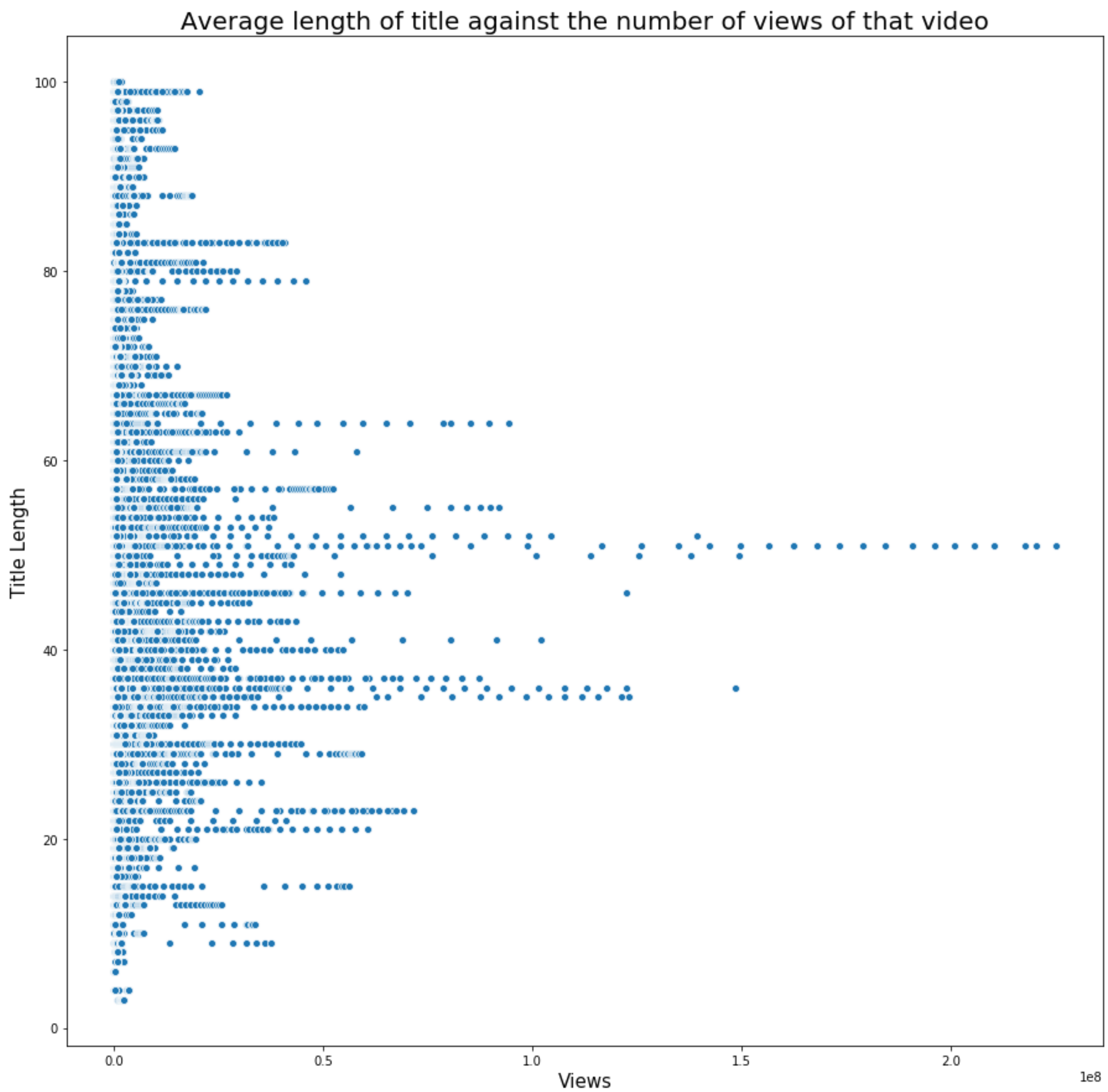
	video_id	trending_date	title	channel_title	category_id	publish_time	
92	#NAME?	2017-11-14	Animal Adventure Park Giraffe Cam	Animal Adventure Park	15	2017-11-12T00:18:43.000Z	
352	#NAME?	2017-11-15	Animal Adventure Park Giraffe Cam	Animal Adventure Park	15	2017-11-12T00:18:43.000Z	
433	#NAME?	2017-11-16	24 Facts about Koalas - mental_floss List Show...	Mental Floss	27	2017-11-15T16:00:00.000Z	
546	#NAME?	2017-11-16	Coach Taggart Monday Presser Ahead of Arizona	GoDucksdotcom	17	2017-11-13T20:41:45.000Z	
578	#NAME?	2017-11-16	Animal Adventure Park Giraffe Cam	Animal Adventure Park	15	2017-11-12T00:18:43.000Z	
...	...	...	...	...	...	...	
40133	#NAME?	2018-06-10	We Bought A House	JennaMarbles	23	2018-05-16T22:33:29.000Z	jenna
40333	#NAME?	2018-06-11	We Bought A House	JennaMarbles	23	2018-05-16T22:33:29.000Z	jenna
40546	#NAME?	2018-06-12	We Bought A House	JennaMarbles	23	2018-05-16T22:33:29.000Z	jenna
40747	#NAME?	2018-06-13	We Bought A House	JennaMarbles	23	2018-05-16T22:33:29.000Z	jenna
40749	#NAME?	2018-06-14	Dumbo Official Teaser Trailer	Disney Movie Trailers	1	2018-06-13T07:00:00.000Z	

397 rows × 19 columns

Observation:

We can see that this video appeared on the list with two different titles.

In [20]:
```
df["title_length"] = df["title"].apply(lambda x: len(x))
plt.figure(figsize = (15,15))
e = sns.scatterplot(data=df,x=df['views'],y=df['title_length'])
e.set_title("Average length of title against the number of views of that vide
o", fontsize=20)
e.set_xlabel("Views",fontsize=15)
e.set_ylabel("Title Length",fontsize=15)
plt.show()
```



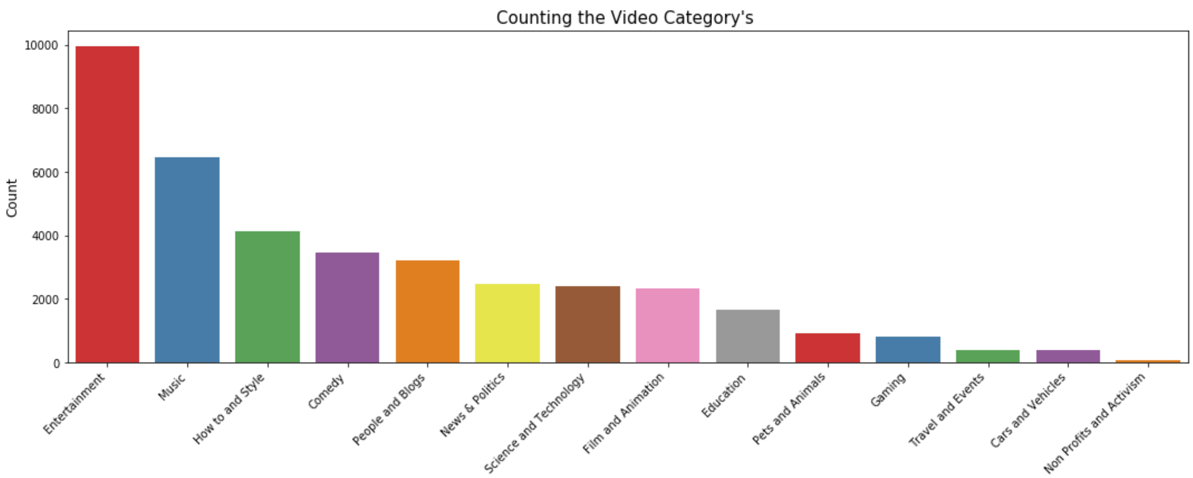
Observation:

By looking at the scatter plot, we can say that there is no relationship between the title length and the number of views. However, we notice an interesting thing: videos that have 100,000,000 views and more have title length between 33 and 55 characters approximately.

To analyse videos into their Category Type

In [21]:
```
plt.figure(figsize = (18,12))

plt.subplot(211)
CatName = sns.countplot('category_name', data=df, palette="Set1",order = df['category_name'].value_counts().index)
CatName.set_xticklabels(CatName.get_xticklabels(),rotation=45, ha ='right')
CatName.set_title("Counting the Video Category's ", fontsize=15)
CatName.set_xlabel("", fontsize=12)
CatName.set_ylabel("Count", fontsize=12)
plt.show()
```



Observation:

From the graph here we can tell we can tell much of the trending video history in 2017 and 2018. Entertainment, Music, How to Style videos. These are the top 3 video categories. And our Projects focus is on Music Category which has substantial number of Data.

Correlation between Like, Dislikes, Views and Comment Count

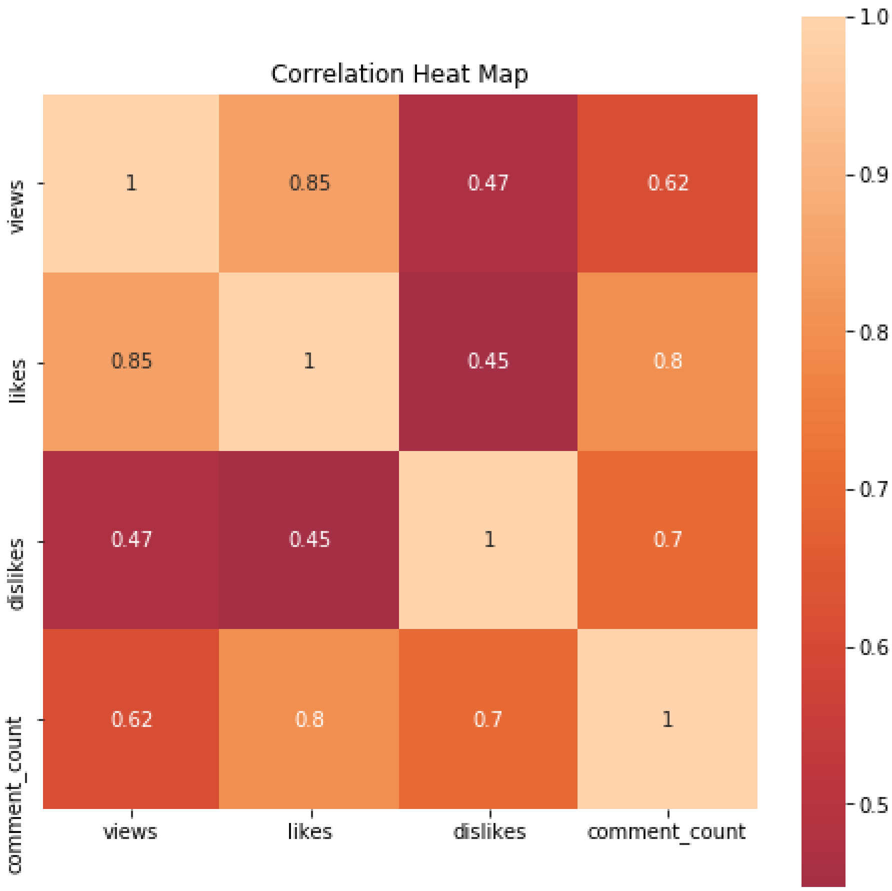
In [22]:
```
correlation_values = ['views', 'likes','dislikes','comment_count']
correlation_table = df[correlation_values].corr()
correlation_table.head()
```

Out[22]:

	views	likes	dislikes	comment_count
views	1.000000	0.849177	0.472213	0.617621
likes	0.849177	1.000000	0.447186	0.803057
dislikes	0.472213	0.447186	1.000000	0.700184
comment_count	0.617621	0.803057	0.700184	1.000000

In [23]:
```
plt.figure(figsize=(8,8))
sns.heatmap(correlation_table, annot=True, square = True,vmax=1,center=0)
plt.title("Correlation Heat Map")
```

Out[23]: `Text(0.5, 1, 'Correlation Heat Map')`



Observation

This is the correlation values where: 1 is strong positive correlation, 0 is no correlation, -1 is strong negative correlation.

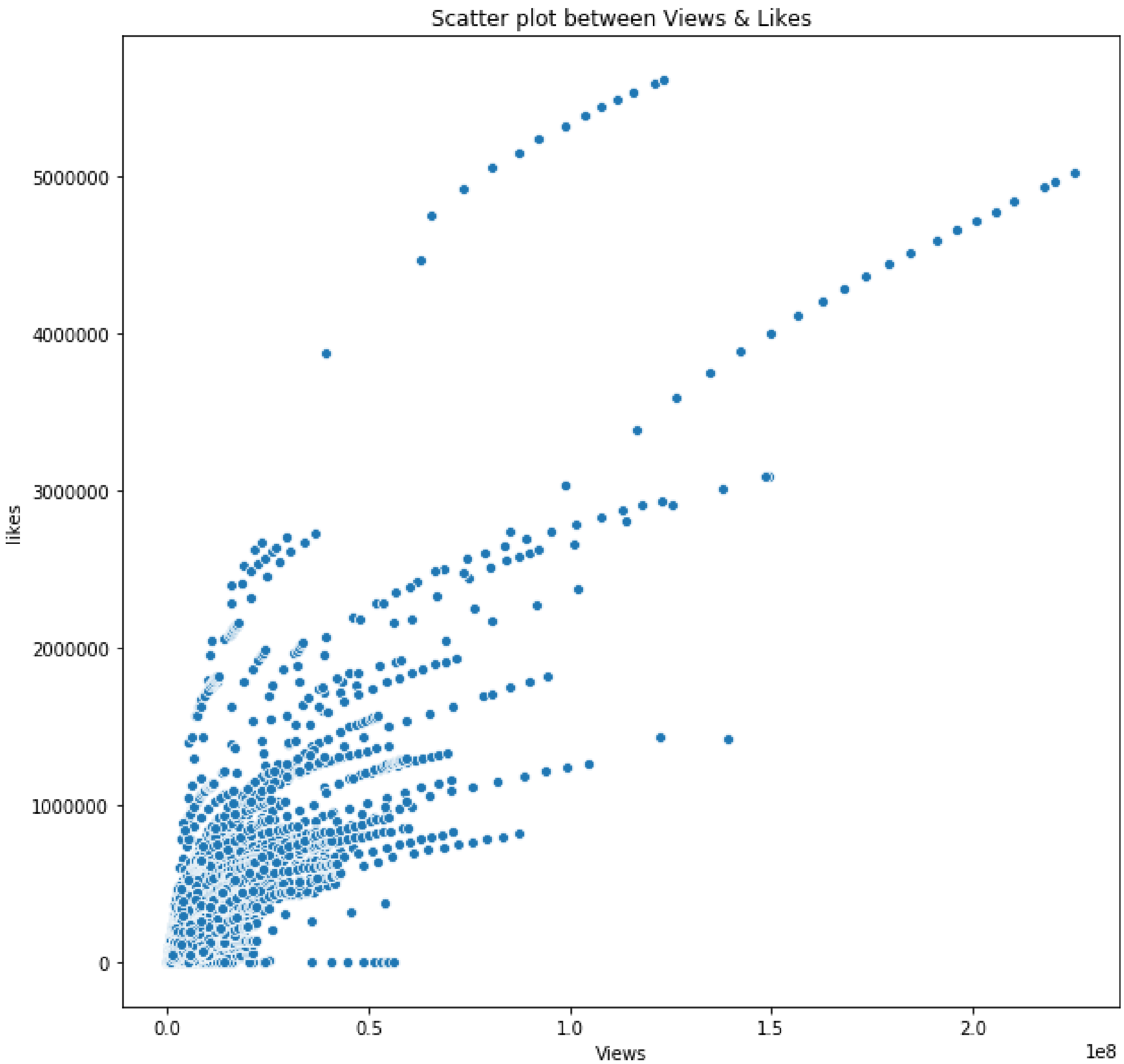
There is a strong positive correlation between likes and views, likes and comments, and views and comments. Dislikes have a lower correlation, which makes sense. For future testing, we may combine likes and dislikes as one term, because creators such as Logan Paul have proven that being unpopular and bringing public hate upon oneself still may be profitable over all. Further testing will need to be done to prove the significance of these findings.

Scatterplot between Likes and Views

Further narrow down to see Correlation between Likes and Views only

In [24]:
```
likesandviews = df[['views','likes']]
corr = likesandviews.corr()
```

In [25]:
```
plt.figure(figsize=(10,10))
sns.scatterplot(df["views"], df["likes"])
plt.title("Scatter plot between Views & Likes")
plt.ylabel("likes")
plt.xlabel("Views")
plt.show()
```

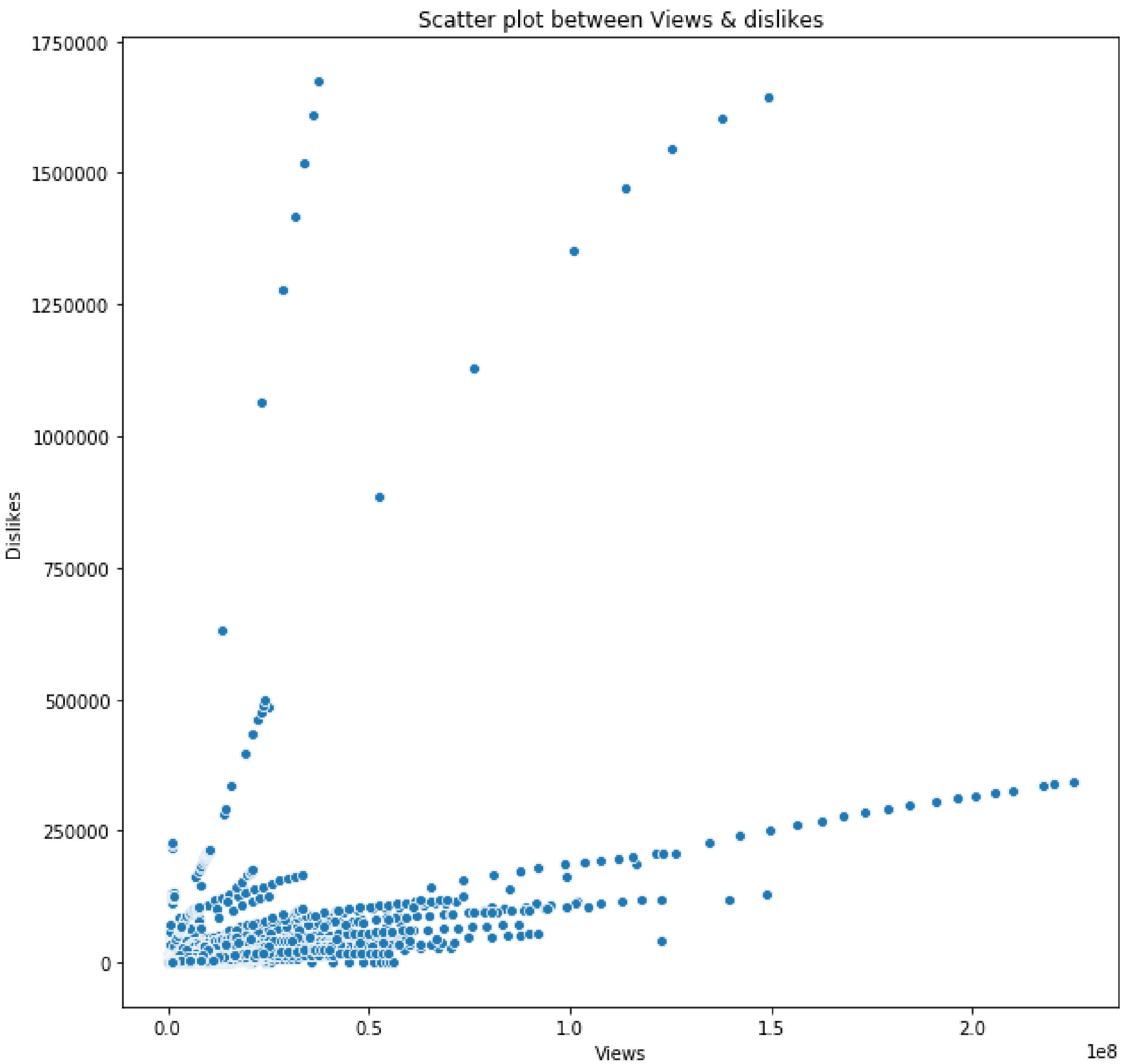


Observation:

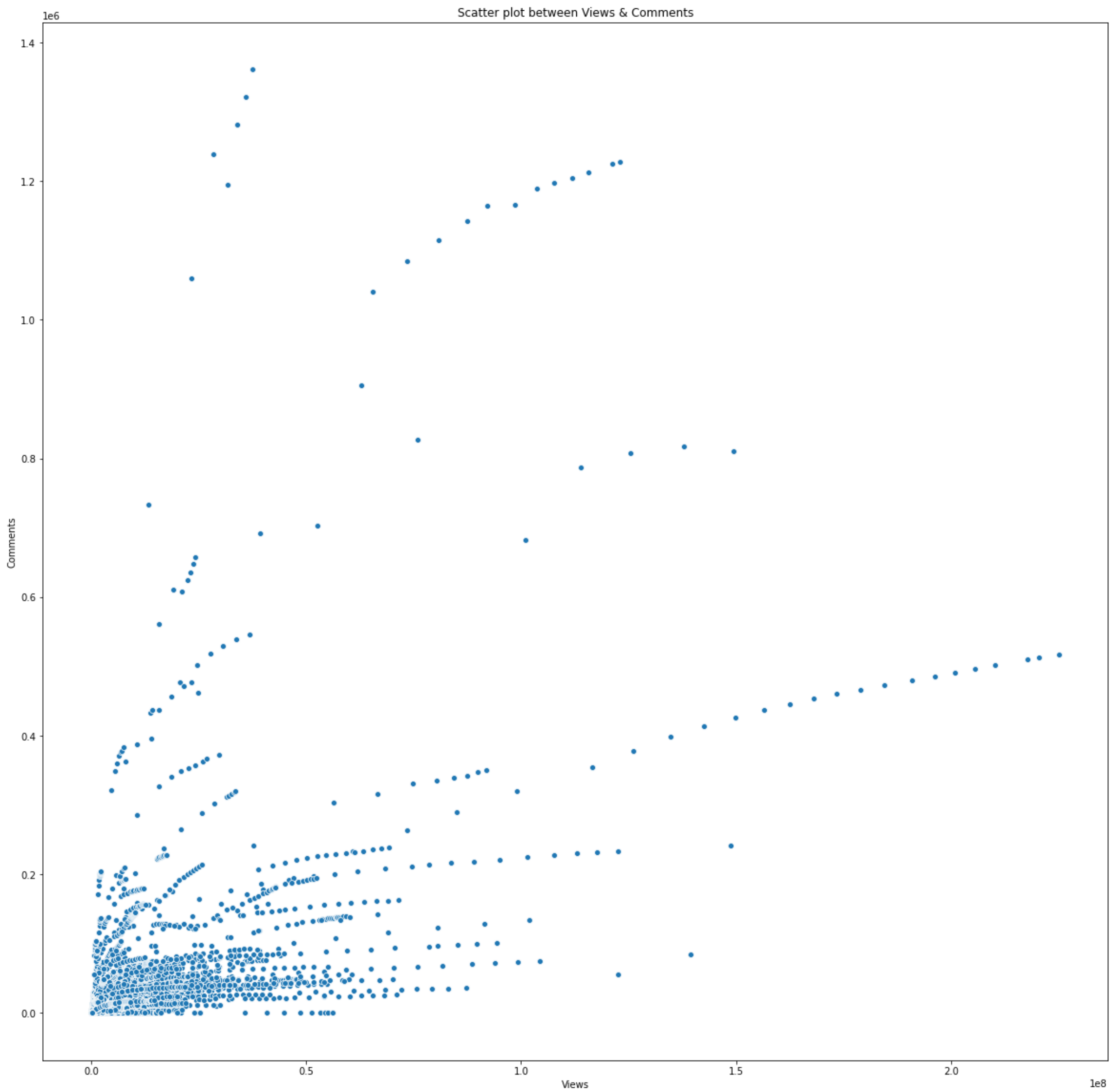
A strong exponential growth between the likes and views. However, as it approaches 250,000,000 views the number of likes become disproportionately lesser. We can also see that most videos within 500,000,000 views and 1,000,000 likes

But there are videos which surpass these views counts and the maximum is around 200 million views.

In [26]:
```
plt.figure(figsize=(10,10))
sns.scatterplot(df["views"], df["dislikes"])
plt.title("Scatter plot between Views & dislikes")
plt.ylabel("Dislikes")
plt.xlabel("Views")
plt.show()
```



In [26]:
```
plt.figure(figsize=(20,20))
sns.scatterplot(df["views"], df["comment_count"])
plt.title("Scatter plot between Views & Comments")
plt.ylabel("Comments")
plt.xlabel("Views")
plt.show()
```



Observation:

there may be multicollinearity issues here, might have to re-frame variables moving forwards

In [27]:
```
df['impressions']=df['likes']+df['comment_count']
model=smf.ols(formula='views ~ impressions + dislikes',data=df)
results=model.fit()
print(results.summary())
```

```
                            OLS Regression Results                            
==============================================================================
=
Dep. Variable:                  views   R-squared:                       0.70
5
Model:                            OLS   Adj. R-squared:                  0.70
5
Method:                 Least Squares   F-statistic:                 4.886e+0
4
Date:                Wed, 02 Dec 2020   Prob (F-statistic):               0.0
0
Time:                        13:23:54   Log-Likelihood:            -6.8079e+0
5
No. Observations:               40949   AIC:                         1.362e+0
6
Df Residuals:                   40946   BIC:                         1.362e+0
6
Df Model:                           2                                         
Covariance Type:            nonrobust                                         
==============================================================================
==
                  coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Intercept    4.086e+05   2.08e+04     19.597      0.000    3.68e+05    4.49e+
05
impressions    22.7199      0.088    258.447      0.000      22.548      22.8
92
dislikes       19.6581      0.787     24.977      0.000      18.115      21.2
01
==============================================================================
=
Omnibus:                    37240.278   Durbin-Watson:                   1.90
3
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         27296916.57
3
Skew:                           3.366   Prob(JB):                         0.0
0
Kurtosis:                     129.306   Cond. No.                    2.87e+0
5
==============================================================================
=

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
[2] The condition number is large, 2.87e+05. This might indicate that there a
re
strong multicollinearity or other numerical problems.
```

In [28]:
```
model2=smf.ols(formula='views ~ likes + comment_count + dislikes',data=df)
results=model2.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                  views   R-squared:                       0.78
3
Model:                            OLS   Adj. R-squared:                  0.78
3
Method:                 Least Squares   F-statistic:                 4.928e+0
4
Date:                Wed, 02 Dec 2020   Prob (F-statistic):              0.0
0
Time:                        13:23:57   Log-Likelihood:           -6.7447e+0
5
No. Observations:               40949   AIC:                         1.349e+0
6
Df Residuals:                   40945   BIC:                         1.349e+0
6
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
====
                    coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
Intercept      2.374e+05   1.79e+04     13.247      0.000    2.02e+05    2.73
e+05
likes            35.5501      0.130    274.299      0.000      35.296       3
5.804
comment_count   -97.7268      0.993    -98.430      0.000     -99.673      -9
5.781
dislikes         83.1609      0.853     97.505      0.000      81.489       8
4.833
==============================================================================
=
Omnibus:                    35876.481   Durbin-Watson:                   1.91
7
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         13801392.45
7
Skew:                           3.318   Prob(JB):                        0.0
0
Kurtosis:                      92.693   Cond. No.                    2.56e+0
5
==============================================================================
=

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
[2] The condition number is large, 2.56e+05. This might indicate that there a
re
strong multicollinearity or other numerical problems.
```

In [29]:
```
#subsetting only music based videos
dfM = df[df["category_id"]==10]
```

In [30]:
```
dfM.head()
```

Out[30]:

	video_id	trending_date	title	channel_title	category_id	publish_time	
12	5E4ZBSInqUU	2017-11-14	Marshmello - Blocks (Official Music Video)	marshmello	10	2017-11-13T17:00:00.000Z	ma
32	n1WpP7iowLc	2017-11-14	Eminem - Walk On Water (Audio) ft. Beyoncé	EminemVEVO	10	2017-11-10T17:00:03.000Z	Emi
37	e_7zHm7GsYc	2017-11-14	Hunter Hayes - You Should Be Loved (Part One O...	Hunter Hayes	10	2017-11-13T15:01:18.000Z	Hu
39	zZ9FciUx6gs	2017-11-14	Nickelback - The Betrayal Act III [Official Vi...	Nickelback	10	2017-11-13T15:31:44.000Z	Nick
40	PaJCFHXcWmM	2017-11-14	U2 - The Blackout	U2VEVO	10	2017-11-13T17:00:04.000Z	

5 rows × 21 columns

In [31]:
```
#cleaning out song titles not in the right format of "artist" - "song title"
DFA = dfM[dfM.title.str.contains(" - ")]
DFA.head()
DFA.head(50)
```

Out[31]:

	video_id	trending_date	title	channel_title	category_id	publi
12	5E4ZBSInqUU	2017-11-14	Marshmello - Blocks (Official Music Video)	marshmello	10	13T17:00
32	n1WpP7iowLc	2017-11-14	Eminem - Walk On Water (Audio) ft. Beyoncé	EminemVEVO	10	10T17:00
37	e_7zHm7GsYc	2017-11-14	Hunter Hayes - You Should Be Loved (Part One O...	Hunter Hayes	10	13T15:01
39	zZ9FciUx6gs	2017-11-14	Nickelback - The Betrayal Act III [Official Vi...	Nickelback	10	13T15:31
40	PaJCFHXcWmM	2017-11-14	U2 - The Blackout	U2VEVO	10	13T17:00
43	0tO_I_Ed5Rs	2017-11-14	Matthew Santoro - FACTS (Official Music Video)...	MatthewSantoro	10	11T16:00
53	9t9u_yPEidY	2017-11-14	Jennifer Lopez - Amor, Amor, Amor (Official Vi...	JenniferLopezVEVO	10	10T15:00
63	ujyTQNNjjDU	2017-11-14	G-Eazy - The Plan (Official Video)	GEazyMusicVEVO	10	10T05:00
70	2Vv-BfVoq4g	2017-11-14	Ed Sheeran - Perfect (Official Music Video)	Ed Sheeran	10	09T11:04
74	lY_0mkYDZDU	2017-11-14	Foster The People - Sit Next to Me (Official V...	fosterthepeopleVEVO	10	10T17:00
77	ObIQ0s02UHg	2017-11-14	Jason Derulo - Tip Toe feat. French Montana (O...	Jason Derulo	10	10T14:40

	video_id	trending_date	title	channel_title	category_id	publi
87	_Iz83-Cmt6A	2017-11-14	Little Big Town with Jimmy Webb - Wichita Line...	CMAVEVO	10	09T22:23
88	YlvCVbfS9M0	2017-11-14	Alan Walker - All Falls Down (Behind The Scenes)	Alan Walker	10	11T15:11
95	e4FApt6z55c	2017-11-14	Kimbra - Top of the World (Official Music Video)	kimbramusic	10	10T15:43
99	RkHuWjiR-LM	2017-11-14	Neck Deep - Parachute (Official Music Video)	Hopeless Records	10	10T18:03
104	pz95u3UVpaM	2017-11-14	Camila Cabello - Havana (Vertical Video) ft. Y...	CamilaCabelloVEVO	10	10T05:01
109	viyRD5z6ilQ	2017-11-14	Luke Bryan - O Holy Night (Audio)	LukeBryanVEVO	10	10T05:00
110	QFfEtKvXMAs	2017-11-14	Niall Horan - Too Much To Ask (Acoustic)	NiallHoranVEVO	10	10T17:00
111	xg9ebVTL9yE	2017-11-14	Empire Of The Sun - Way To Go	empireofthesunvevo	10	10T05:00
112	fbHbTBP_u7U	2017-11-14	NF - Let You Down	NFVEVO	10	09T05:00
114	QOksZ8VogRw	2017-11-14	Bastille - World Gone Mad (from Bright: The Al...	Atlantic Records	10	09T17:03
124	J_QGZspO4gg	2017-11-14	Sia - Snowman	SiaVEVO	10	09T15:45
126	ozkqm2ifMw8	2017-11-14	2CELLOS - Cinema Paradiso [OFFICIAL VIDEO]	2CELLOS	10	10T13:24

	video_id	trending_date	title	channel_title	category_id	publi
127	htvR_dBs3eg	2017-11-14	Sam Smith - The Thrill of It All ALBUM REVIEW	theneedledrop	10	10T21:38
129	1Wk8ZRgXQnY	2017-11-14	Andy Grammer - The Good Parts (Official Audio)	Andy Grammer	10	09T22:27
130	8l_e6bx8UG8	2017-11-14	Remy Ma - Wake Me Up (Audio) ft. Lil' Kim	RemyMaVEVO	10	08T17:00
134	cYw-oyJ7AEY	2017-11-14	Pitbull, Stereotypes - Jungle (Lyric Video) ft...	PitbullVEVO	10	10T08:00
136	5x1FAiIq_pQ	2017-11-14	Alicia Keys - When You Were Gone	Alicia Keys	10	09T15:49
137	LMCuKltaY3M	2017-11-14	Elbow - Golden Slumbers (John Lewis Advert 2017)	ElbowVEVO	10	10T08:00
140	7fm7mll2qvg	2017-11-14	Sigrid - Strangers (Lyric Video)	SigridVEVO	10	10T00:00
141	5gFpcEKayz4	2017-11-14	MØ - When I Was Young (Official Video)	MOMOMOYOUTHVEVO	10	09T09:00
142	eHIY3HNNqzM	2017-11-14	The Script - Arms Open (Acoustic) [Audio]	TheScriptVEVO	10	10T08:00
148	44NYFvhXmW8	2017-11-14	Thirty Seconds To Mars - Walk On Water (Offici...	ThirtySecondsToMarsVEVO	10	08T13:00
149	9wg3v-01yKQ	2017-11-14	Harry Styles - Kiwi	HarryStylesVEVO	10	08T13:00
152	afgvlR9WmIQ	2017-11-14	Maroon 5 - What Lovers Do (Live On The Ellen D...	Maroon5VEVO	10	09T02:21

	video_id	trending_date	title	channel_title	category_id	publi
162	9ymjcSvEyhk	2017-11-14	Enter Shikari - The Sights (Official Video)	Enter Shikari	10	09T14:00
164	HNa_UWX51_s	2017-11-14	Last Friday Night - Katy Perry ('40s Jazz Vibe...	PostmodernJukebox	10	09T18:46
165	08nkwgZIE4I	2017-11-14	P!nk - Barbies (Audio)	PinkVEVO	10	08T22:04
170	_w58R1OGQFA	2017-11-14	Train - Have Yourself a Merry Little Christmas	TrainVEVO	10	09T15:00
188	UFPSIa1cLRQ	2017-11-14	Phillip Phillips - Magnetic (Audio)	PhilPhillipsVEVO	10	09T05:00
190	iNf6VErGDLI	2017-11-14	Ozuna - Música Sin Fronteras (A YouTube Docume...	Ozuna	10	08T14:00
222	qDAxDcjgn-8	2017-11-15	Taylor Swift - Reputation ALBUM REVIEW	theneedledrop	10	14T23:06
234	5E4ZBSInqUU	2017-11-15	Marshmello - Blocks (Official Music Video)	marshmello	10	13T17:00
245	zZ9FciUx6gs	2017-11-15	Nickelback - The Betrayal Act III [Official Vi...	Nickelback	10	13T15:31
252	PaJCFHXcWmM	2017-11-15	U2 - The Blackout	U2VEVO	10	13T17:00
258	KhdecX0qjNA	2017-11-15	Demi Lovato - Sorry Not Sorry in the Live Lounge	BBCRadio1VEVO	10	13T20:11
269	e_7zHm7GsYc	2017-11-15	Hunter Hayes - You Should Be Loved (Part One O...	Hunter Hayes	10	13T15:01

	video_id	trending_date	title	channel_title	category_id	publi
278	kqmhCaF5_44	2017-11-15	HANSON - Finally It's Christmas (Official Lyri...	HANSON	10	14T16:10
298	n1WpP7iowLc	2017-11-15	Eminem - Walk On Water (Audio) ft. Beyoncé	EminemVEVO	10	10T17:00
305	0tO_l_Ed5Rs	2017-11-15	Matthew Santoro - FACTS (Official Music Video)...	MatthewSantoro	10	11T16:00

50 rows × 21 columns

In [49]:
```
#creating a cleaned version of artist, title variable to match with tags later
artist=[]
songname=[]
for row in range(len(DFA)):
    artistt,songnamet=DFA.iloc[row]['title'].split(" - ", 1)
    artistt=artistt.lower()
    songnamet=songnamet.lower()
    artistt=artistt.replace(' ','')
    songnamet=songnamet.replace(' ','')
    if songnamet.find("(")!=-1:
        songnamet =songnamet.split("(")[0]
    if songnamet.find("[")!=-1:
        songnamet =songnamet.split("[")[0]
    if songnamet.find("inthelive")!=-1:
        songnamet =songnamet.split("inthelive")[0]
    if artistt.find("[OFFICIAL VIDEO]")!=-1:
        artistt=artistt.split("[OFFICIAL VIDEO] ")[1]
    if artistt.find(",")!=-1:
        artistt=artistt.split(",")[0]
    if artistt.find("(")!=-1:
        artistt=artistt.split(" (")[0]

    songname.append(songnamet)
    artist.append(artistt)
```

In [33]:
```
DFA['artist']=artist
```

```
C:\Users\tobys\anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [34]:
```
DFA['songname']=songname
```

```
C:\Users\tobys\anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [36]:
```
# Recoding to find the "number of tags used in a video"
temp=[]
for row in range(len(DFA)):
    x = len(DFA.iloc[row]['tags'].split("|"))
    if x<=1:
        if str(DFA.iloc[row]['tags'].split("|"))=="[none]":
            temp.append(0)
        else:
            temp.append(x)
        # since we are searching for the separator value, we will need to add
 one to our counts if the
        # count itself is above 1 (i.e one | being found will mean that there
 are two tags being used)
    else: temp.append(x+1)
DFA['numtags']=temp
```

```
C:\Users\tobys\anaconda3\lib\site-packages\ipykernel_launcher.py:13: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  del sys.path[0]
```

In [38]:
```
#creating an aggregate of all the tags into one string per video
temp2=[]
for row in range(len(DFA)):
    x = DFA.iloc[row]['tags'].split("|")
    y=''.join(x)
    y=y.replace('"','')
    y=y.replace(' ','')
    y=y.lower()
    temp2.append(y)
DFA['totaltag']=temp2
```

```
C:\Users\tobys\anaconda3\lib\site-packages\ipykernel_launcher.py:10: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```

In [40]:
```
#now coding to find whether or not artist name/song title/genre/etc... is in t
ags. will code them individually
#Coding for tags may have some errors in it, since we're working with unstruct
ured text and so there is no good way to
#search for things besides tags. However, the vast majority of videos are corr
ectly tagged, so this should not
#create an issue for our model over all.
artistTag=[]
songTag=[]
for row in range(len(DFA)):
    #when using the .find() function, returning an index of -1 means that the
 substring has not been found. in real world terms,
    #this means that the thing we are searching for is not found within the ta
gs. The tags do not contain artist or song name in
    #this situation.
    if DFA.iloc[row]['totaltag'].find(DFA.iloc[row]['artist'])==-1:
        artistTag.append(False)
    else:
        artistTag.append(True)
    if DFA.iloc[row]['totaltag'].find(DFA.iloc[row]['songname'])==-1:
        songTag.append(False)
    else:
        songTag.append(True)
DFA['artistintags']=artistTag
DFA['songintags']=songTag
```

```
C:\Users\tobys\anaconda3\lib\site-packages\ipykernel_launcher.py:19: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
C:\Users\tobys\anaconda3\lib\site-packages\ipykernel_launcher.py:20: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [42]:
```
#Searching for certain record labels. Will only be looking at big names in the industry, since they're the only
#companies that put their record label in their tags. Some quick background research will show that
#universal, Warner media, and columbia records make up the majority of the record label indsutry.
labelTag=[]
for row in range(len(DFA)):
    #when using the .find() function, returning an index of -1 means that the substring has not been found. in real world terms,
    #this means that the thing we are searching for is not found within the tags. The tags do not contain artist or song name in
    #this situation.
    if DFA.iloc[row]['totaltag'].find('records')!=-1:
        labelTag.append(True)
    elif DFA.iloc[row]['totaltag'].find('columbia')!=-1:
        labelTag.append(True)
    elif DFA.iloc[row]['totaltag'].find('sony')!=-1:
        labelTag.append(True)
    elif DFA.iloc[row]['totaltag'].find('universal')!=-1:
        labelTag.append(True)
    elif DFA.iloc[row]['totaltag'].find('interscope')!=-1:
        labelTag.append(True)
    elif DFA.iloc[row]['totaltag'].find('rca')!=-1:
        labelTag.append(True)
    else:
        labelTag.append(False)
DFA['haslabel']=labelTag
```

```
C:\Users\tobys\anaconda3\lib\site-packages\ipykernel_launcher.py:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [43]:
```
#searching for "music video" in tags, to see if music videos do better than au
dio only videos
videoTag=[]
for row in range(len(DFA)):
    #when using the .find() function, returning an index of -1 means that the
 substring has not been found. in real world terms,
    #this means that the thing we are searching for is not found within the ta
gs. The tags do not contain artist or song name in
    #this situation.
    if DFA.iloc[row]['totaltag'].find('musicvideo')==-1:
        videoTag.append(False)
    elif DFA.iloc[row]['totaltag'].find('video')==-1:
        videoTag.append(False)
    else:
        videoTag.append(True)
DFA['musicvideo']=videoTag
```

```
C:\Users\tobys\anaconda3\lib\site-packages\ipykernel_launcher.py:13: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  del sys.path[0]
```

In [45]:
```
model4=smf.ols(formula='views ~ impressions +days_to_trending+ numtags + hasla
bel + musicvideo + artistintags + songintags',data=DFA)
resultssss=model4.fit()
print(resultssss.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                  views   R-squared:                       0.85
8
Model:                            OLS   Adj. R-squared:                  0.85
8
Method:                 Least Squares   F-statistic:                     453
9.
Date:                Wed, 02 Dec 2020   Prob (F-statistic):               0.0
0
Time:                        13:25:11   Log-Likelihood:                -8988
5.
No. Observations:                5277   AIC:                         1.798e+0
5
Df Residuals:                    5269   BIC:                         1.798e+0
5
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
===========
                          coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-----------
Intercept           -1.674e+06   3.03e+05     -5.533      0.000   -2.27e+06
-1.08e+06
haslabel[T.True]    -7.564e+05   1.84e+05     -4.115      0.000   -1.12e+06
-3.96e+05
musicvideo[T.True]   3.159e+05   2.62e+05      1.206      0.228   -1.97e+05
8.29e+05
artistintags[T.True]  1.094e+06   2.57e+05      4.253      0.000     5.9e+05
1.6e+06
songintags[T.True]   1.919e+05   2.14e+05      0.897      0.370   -2.27e+05
6.11e+05
impressions            33.3368      0.188    176.959      0.000      32.968
33.706
days_to_trending      966.8282    676.238      1.430      0.153    -358.879
2292.535
numtags              8606.1446   9323.125      0.923      0.356   -9671.044
2.69e+04
==============================================================================
=
Omnibus:                     3777.587   Durbin-Watson:                   1.77
0
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           245988.96
8
Skew:                           2.781   Prob(JB):                         0.0
0
Kurtosis:                      35.982   Cond. No.                     2.18e+0
6
==============================================================================
=

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
```

```
[2] The condition number is large, 2.18e+06. This might indicate that there a
re
strong multicollinearity or other numerical problems.
```

Observation

From our information, the number of impressions is important, as well as whether an artist is part of a record label, and the artist name is in the tags. Notably, the title of the video, as well as whether or not the video in question is a music video, are not significant. This means that people are searching for the artist tag more, than anything else. For every new impression on a video, the video will gain 33 views on average. Creating music videos does not add views, and with people’s current mindset about record labels, it may be best to leave record label tags out.

The main issue with this model is that impressions seems to greatly outweigh the effects of all other variables. To improve our model, we will most likely need to remove it for our next regressions.

In [46]:
```
model5=smf.ols(formula='views ~  days_to_trending+numtags + haslabel + musicvi
deo + artistintags + songintags',data=DFA)
resultsss=model5.fit()
print(resultsss.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                  views   R-squared:                       0.01
2
Model:                            OLS   Adj. R-squared:                  0.01
1
Method:                 Least Squares   F-statistic:                     11.0
2
Date:                Wed, 02 Dec 2020   Prob (F-statistic):           3.05e-1
2
Time:                        13:25:12   Log-Likelihood:                -9499
8.
No. Observations:                5277   AIC:                         1.900e+0
5
Df Residuals:                    5270   BIC:                         1.901e+0
5
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
===========
                          coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-----------
Intercept             1.86e+06   7.95e+05      2.339      0.019    3.01e+05
3.42e+06
haslabel[T.True]      2.82e+06   4.81e+05      5.858      0.000    1.88e+06
3.76e+06
musicvideo[T.True]   5.795e+05    6.9e+05      0.840      0.401   -7.73e+05
1.93e+06
artistintags[T.True]  2.354e+06   6.77e+05      3.475      0.001    1.03e+06
3.68e+06
songintags[T.True]   8.526e+05   5.63e+05      1.513      0.130   -2.52e+05
1.96e+06
days_to_trending       57.1606   1781.655      0.032      0.974   -3435.621
3549.942
numtags              5.312e+04    2.46e+04      2.163      0.031    4983.629
1.01e+05
==============================================================================
=
Omnibus:                     6462.997   Durbin-Watson:                   1.86
5
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           880256.86
8
Skew:                           6.630   Prob(JB):                         0.0
0
Kurtosis:                      64.868   Cond. No.                         54
5.
==============================================================================
=

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
```

Observation

From this output, we can see that the record label, artist name, number of tags they use are important. Interestingly, having a record label now has a positive effect on the number of views a video gets. This may be because of the vast resources working with a record label gets an artist access too, regardless of public opinion. Furthermore, the number of tags a video has on it is now statistically significant, as well. This means that the more tags are used, the more views a video will get, due to search engine optimization. However, because some of our tags that we have searched for have come up as insignificant, further exploration would need to be done, and one must also be careful when choosing tags to use

Most Common Words used in TAGS for Music Category

In [39]:
```
title_words = list(dfM["title"].apply(lambda x: x.split())) ## dfM is defined
 earlier for Category 10
title_words = [x for y in title_words for x in y]
Counter(title_words).most_common(20)
```

Out[39]:
```
[('-', 5378),
 ('Video)', 1710),
 ('(Official', 1415),
 ('The', 927),
 ('ft.', 740),
 ('Music', 481),
 ('[Official', 466),
 ('&', 432),
 ('(Audio)', 427),
 ('Me', 422),
 ('You', 411),
 ('(Lyric', 330),
 ('Video]', 313),
 ('the', 307),
 ('|', 293),
 ('I', 239),
 ('Love', 215),
 ('My', 186),
 ('To', 181),
 ('(Live', 171)]
```

Observation:

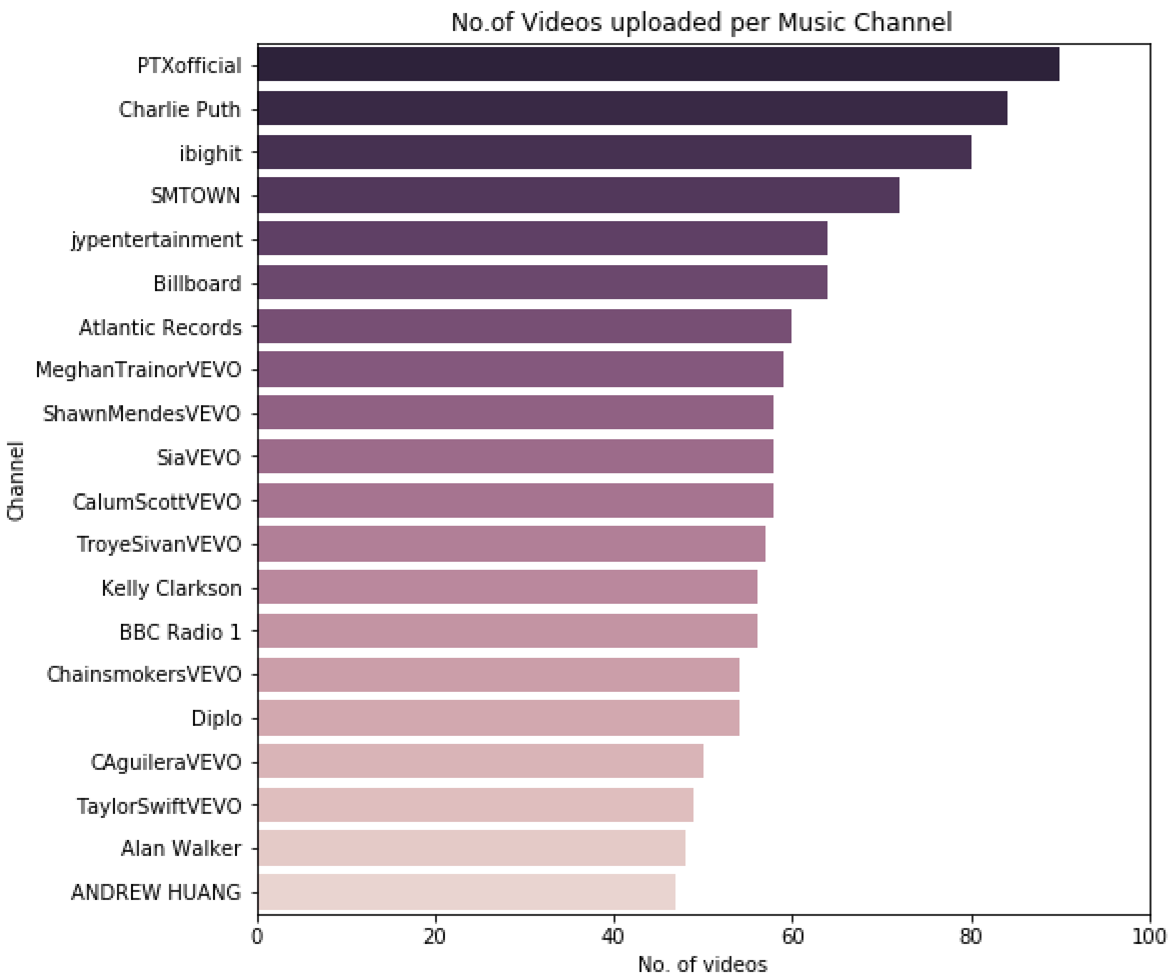
We can see that "-" is very commonly found in the Music Tagging. And further analysing this shows that video publisher use this "-" to further elaborate on their video title and description. Among other observation, "&" "Lyric" "I" "Love" are very popular words associated with the listeners and streamers. Which is something that we can further recommend our client to impart these tags in her video and songs she would sing.

The Genre of Music listened by the YouTube listeners has got to do with "Love","I","Me","You","My","Live" some are which possessive words and expression of feelings of Love and Live.

Top Music Channels with largest number of Trending Videos

In [47]:
```
cdf = dfM.groupby("channel_title").size().reset_index(name="video_count") \
    .sort_values("video_count", ascending=False).head(20)

fig, ax = plt.subplots(figsize=(8,8))
_ = sns.barplot(x="video_count", y="channel_title", data=cdf,
                palette=sns.cubehelix_palette(n_colors=20, reverse=True), ax=a
x)
_ = ax.set(xlabel="No. of videos", ylabel="Channel", title = "No.of Videos upl
oaded per Music Channel", xlim=(0,100))
```



Observation:

'PTXofficial' is the largest music video publisher who accounts for 90 videos that made it to trending in 2017 & 2018. 'VEVO' if put together will account for the largest Music video Channel as it holds MeghanTrainorVEVO, SiaVEVO, ShawMendesVEVO and TaylorSwiftVEVO under its heavyweight belt.

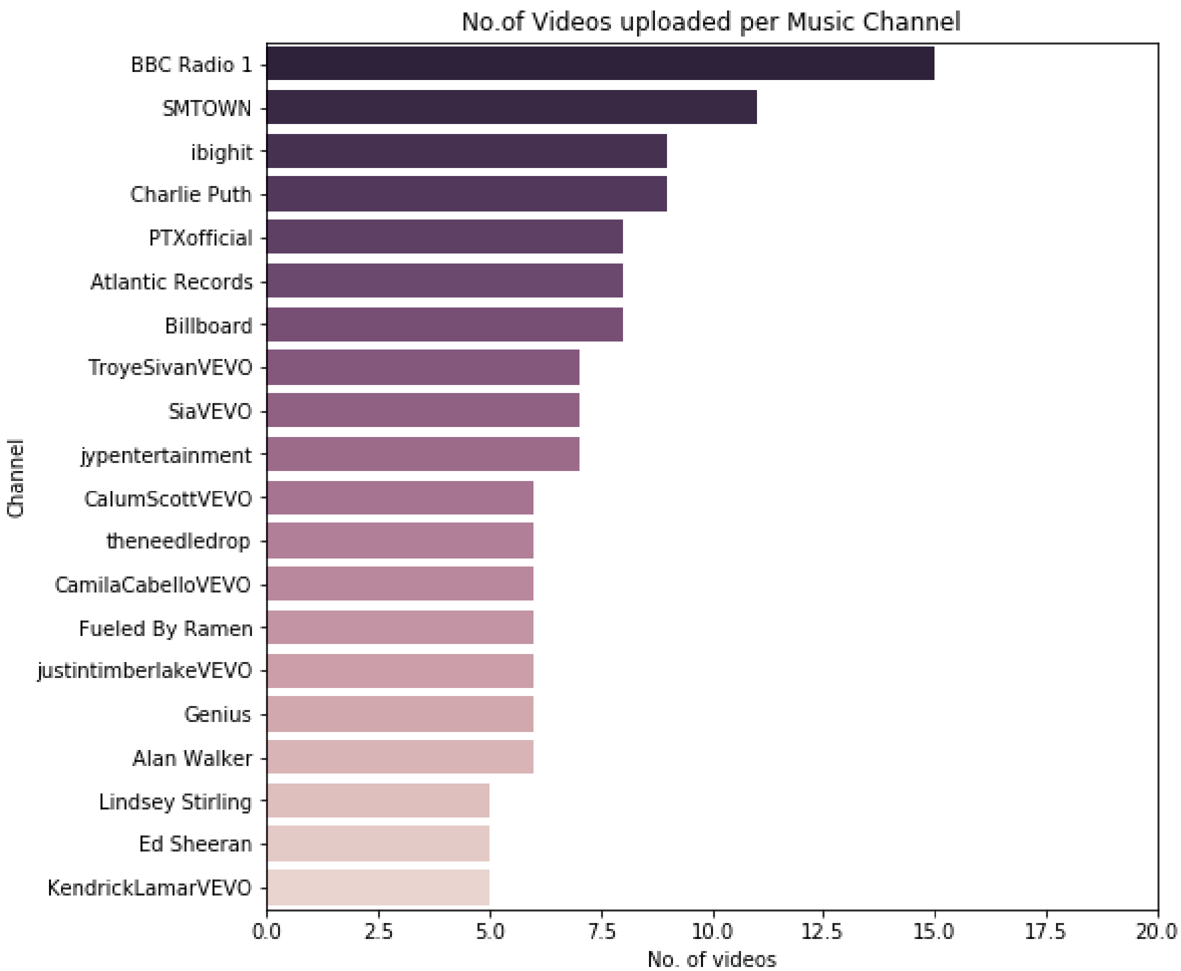
'Charlie Puth' is one such artist who has made a big name for himself and has published several videos (85 videos appox) that have made it to the trending.

Top Music Channels with largest number of Trending Videos that are Unique

In [48]:
```
DFM_duplicates = dfM.drop_duplicates(subset=['title'])

cdf = DFM_duplicates.groupby("channel_title").size().reset_index(name="video_count") \
    .sort_values("video_count", ascending=False).head(20)

fig, ax = plt.subplots(figsize=(8,8))
_ = sns.barplot(x="video_count", y="channel_title", data=cdf,
                palette=sns.cubehelix_palette(n_colors=20, reverse=True), ax=ax)
_ = ax.set(xlabel="No. of videos", ylabel="Channel", title = "No.of Videos uploaded per Music Channel", xlim=(0,20))
```



Observation:

Comparing the earlier chart to the one here. In terms of Unique Video title that got into Trending are from Music Channel 'BBC Radio 1'. 'PTXofficial' who had several of same video title that were gaining traction in trending actually has only 8 unique videos that made it to the trending. 'Charlie Puth' who is an artist also has 9 videos that were unique and raved by the YouTube Community.

The Contrast between the 2 charts show that even though 1 video by a channel can have short view count. Reuploading videos from time to time with different can increase the view counts.

RIGOROUSNESS

Working with this data has been more difficult than we anticipated. Not only have the numerical coefficients of our variables changed over time, but some variables have changed in levels of significance over time. However, as we refine our model, we see that tags become more and more important, both in which tags are used, and how many tags are used.

INSIGHTS

In conclusion, as a record label, you will be able to use this data to prove to up and coming artists that not only is YouTube a good investment as a platform, but working with a record label is still a good idea in 2020. For amateur musicians themselves, learning how to use the YouTube algorithm to favor their videos is a key strategy needed for online growth. Alternatively, depending on the size of record label, it may be a good idea to create an in-house analytics team to be able to have insights into what tags to use, length of videos time to upload, and other metrics.

Future research opportunities:

Aggregating more data on the channel level, to have more insights on what individual channels have done, and what has proven to work for channels as a business, instead of individual videos taken in a vacuum. Furthermore, collecting information on videos that have not made it to trending will be a good idea, since the fact that they made it to trending in the first place may have an effect on views, and may confound the effects of other variables themselves.

In [ ]:

```

```