Author: Ammar Alyousfi, Oct 2018

Analysis of more than 40,000 YouTube trending videos. Python is used with some packages like Pandas and Matplotlib to analyze a dataset that was collected over 205 days. For each of those days, the dataset contains data about the trending videos of that day. It contains data about more than 40,000 trending videos.

Introduction

YouTube is the most popular and most used video platfrom in the world today. YouTube has a list of trending videos that is updated constantly. Here we will use Python with some packages like Pandas and Matplotlib to analyze a dataset that was collected over 205 days. For each of those days, the dataset contains data about the trending videos of that day. It contains data about more than 40,000 trending videos. We will analyze this data to get insights into YouTube trending videos, to see what is common between these videos. Those insights might also be used by people who want to increase popularity of their videos on YouTube.

dataset_kaggle (https://www.kaggle.com/datasnaek/youtube-new)

We want to answer questions like:

- How many views do our trending videos have? Do most of them have a large number of views? Is having a large number of views required for a video to become trending?
- The same questions above, but applied to likes and comment count instead of views.
- Which video remained the most on the trendin-videos list?
- How many trending videos contain a fully-capitalized word in their titles?
- What are the lengths of trending video titles? Is this length related to the video becoming trendy?
- How are views, likes, dislikes, comment count, title length, and other attributes correlate with (relate to) each other? How are they connected?
- What are the most common words in trending video titles?
- Which YouTube channels have the largest number of trending videos?
- Which video category (e.g. Entertainment, Gaming, Comedy, etc.) has the largest number of trending videos?
- When were trending videos published? On which days of the week? at which times of the day?

In [3]:

```
#! pip install wordcloud
```

In [4]:

```
import os
import pandas as pd
import numpy as np
import matplotlib as mpl
from matplotlib import pyplot as plt
import seaborn as sns

import warnings
from collections import Counter
import datetime
import wordcloud
import json
```

In [5]:

```
# Hiding warnings for cleaner display
warnings.filterwarnings('ignore')

# Configuring some options
%matplotlib inline
%config InlineBackend.fiture_format = 'retina'

# If you want interactive plots, uncomment the next line
#%matplotlib notebook
```

just some visual options for plotting...not pivotal.

In [6]:

```
PLOT_COLORS = ["#268bd2", "#0052CC", "#FF5722", "#b58900", "#003f5c"]
pd.options.display.float_format = '{:.2f}'.format
sns.set(style="ticks")
plt.rc('figure', figsize=(8, 5), dpi=100)
plt.rc('axes', labelpad=20, facecolor="#ffffff", linewidth=0.4, grid=True, lab
elsize=14)
plt.rc('patch', linewidth=0)
plt.rc('xtick.major', width=0.2)
plt.rc('ytick.major', width=0.2)
plt.rc('grid', color='#9E9E9E', linewidth=0.4)
plt.rc('font', family='Arial', weight='400', size=10)
plt.rc('text', color='#282828')
plt.rc('savefig', pad_inches=0.3, dpi=300)
```

Reading the dataset

In [7]:

```
data_string = 'USvideos.csv'
json_string = "US_category_id.json" # you will find later
```

In [8]:

```
os.getcwd()
```

Out[8]:

```
'/Users/giovanni/Desktop/AI_ML_DL_DataScience/youtube video trendi
ng analysis'
```

In [9]:

```
df = pd.read_csv(data_string)
df.tail()
```

Out[9]:

	video_id	trending_date	title	channel_title	category_id	
40944	BZt0qjTWNhw	18.14.06	The Cat Who Caught the Laser	AaronsAnimals	15	2018 18T1
40945	1h7KV2sjUWY	18.14.06	True Facts : Ant Mutualism	zefrank1	22	2018 18T0
40946	D6Oy4LfoqsU	18.14.06	I GAVE SAFIYA NYGAARD A PERFECT HAIR MAKEOVER ...	Brad Mondo	24	2018 18T1
40947	oV0zkMe1K8s	18.14.06	How Black Panther Should Have Ended	How It Should Have Ended	1	2018 17T1
40948	ooyjaVdt-jA	18.14.06	Official Call of Duty®: Black Ops 4 — Multipla...	Call of Duty	20	2018 17T1

In [10]:

```
df.info() # see the variables, type, size
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40949 entries, 0 to 40948
Data columns (total 16 columns):
video_id                  40949 non-null object
trending_date             40949 non-null object
title                     40949 non-null object
channel_title             40949 non-null object
category_id               40949 non-null int64
publish_time              40949 non-null object
tags                      40949 non-null object
views                     40949 non-null int64
likes                     40949 non-null int64
dislikes                  40949 non-null int64
comment_count             40949 non-null int64
thumbnail_link            40949 non-null object
comments_disabled         40949 non-null bool
ratings_disabled          40949 non-null bool
video_error_or_removed    40949 non-null bool
description               40379 non-null object
dtypes: bool(3), int64(5), object(8)
memory usage: 4.2+ MB
```

We can see that there are 38916 entries in the dataset. We can see also that all columns in the dataset are complete (i.e. they have 38916 non-null entries) except description column which has some null values; it only has 38304 non-null values.

Data cleaning

The description column has some null values. These are some of the rows whose description values are null. We can see that null values are denoted by NaN

In [11]:

```
df[df["description"].apply(lambda x: pd.isna(x))].head(3)
```

Out[11]:

	video_id	trending_date	title	channel_title	category_id	p
42	NZFhMSgbKKM	17.14.11	Dennis Smith Jr. and LeBron James go back and ...	Ben Rohrbach	17	2017-13T1
47	sbcbvuitiTc	17.14.11	Stephon Marbury and Jimmer Fredette fight in C...	NBA Highlights · YouTube	17	2017-10T1
175	4d07RXYLsJE	17.14.11	Sphaera - demonstrating interaction	Jenny Hanell	28	2017-04T2

So to do some sort of data cleaning, and to get rid of those null values, we put an empty string in place of each null value in the description column

In [12]:

```
df.description = df.description.fillna(value="")
```

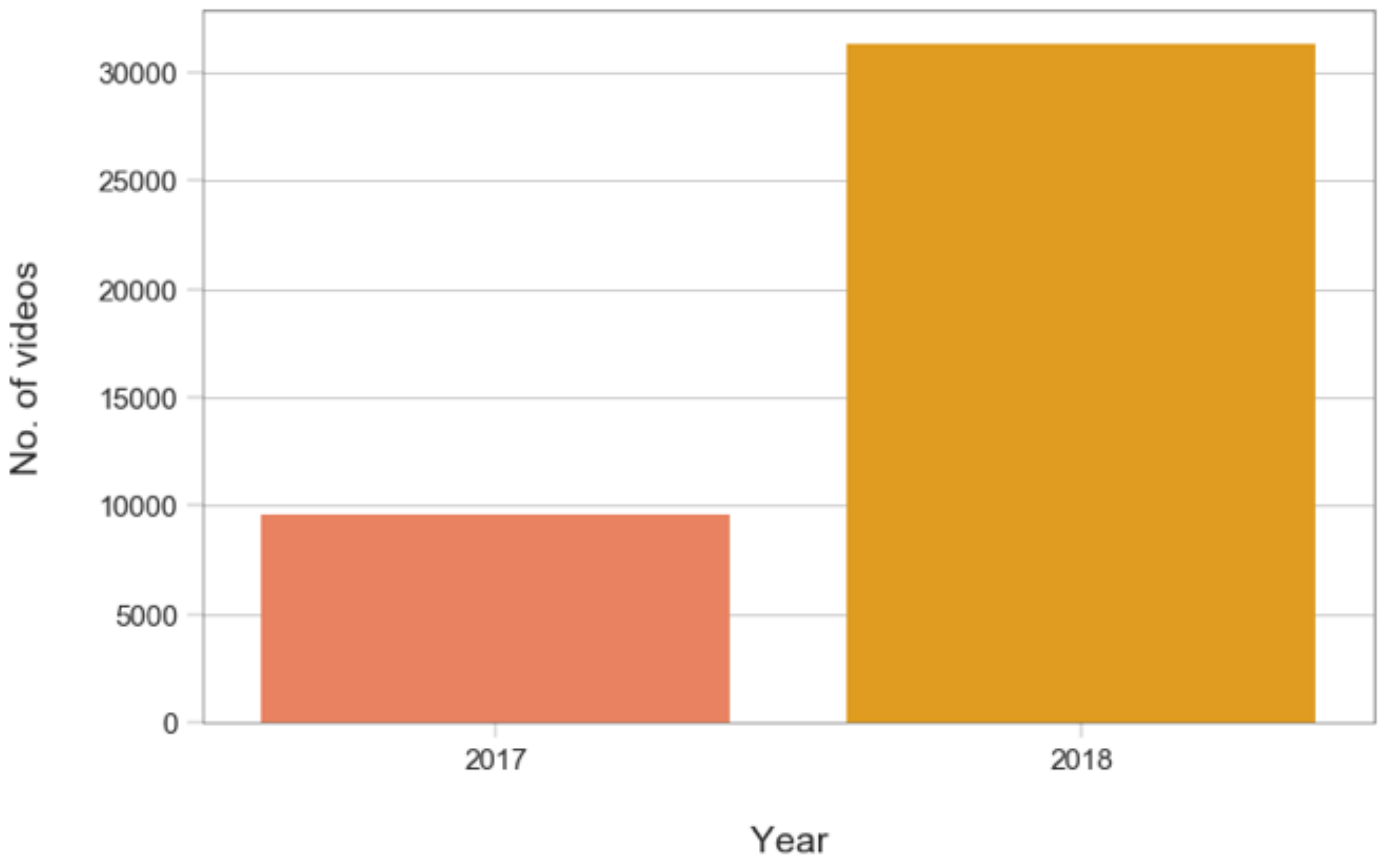
views histogram

Understand from which year the data comes from

In [13]:

```
cdf = df["trending_date"].apply(lambda x: '20' + x[:2]).value_counts() \
            .to_frame().reset_index() \
            .rename(columns={"index": "year", "trending_date": "No_of_videos"}
)

fig, ax = plt.subplots()
_ = sns.barplot(x="year", y="No_of_videos", data=cdf,
                palette=sns.color_palette(['#ff764a', '#ffa600'], n_colors=7),
ax=ax)
_ = ax.set(xlabel="Year", ylabel="No. of videos")
```



In [14]:

```
df["trending_date"].apply(lambda x: '20' + x[:2]).value_counts(normalize=True)
```

Out[14]:

```
2018   0.77
2017   0.23
Name: trending_date, dtype: float64
```

In [15]:

```
df.describe()
```

Out[15]:

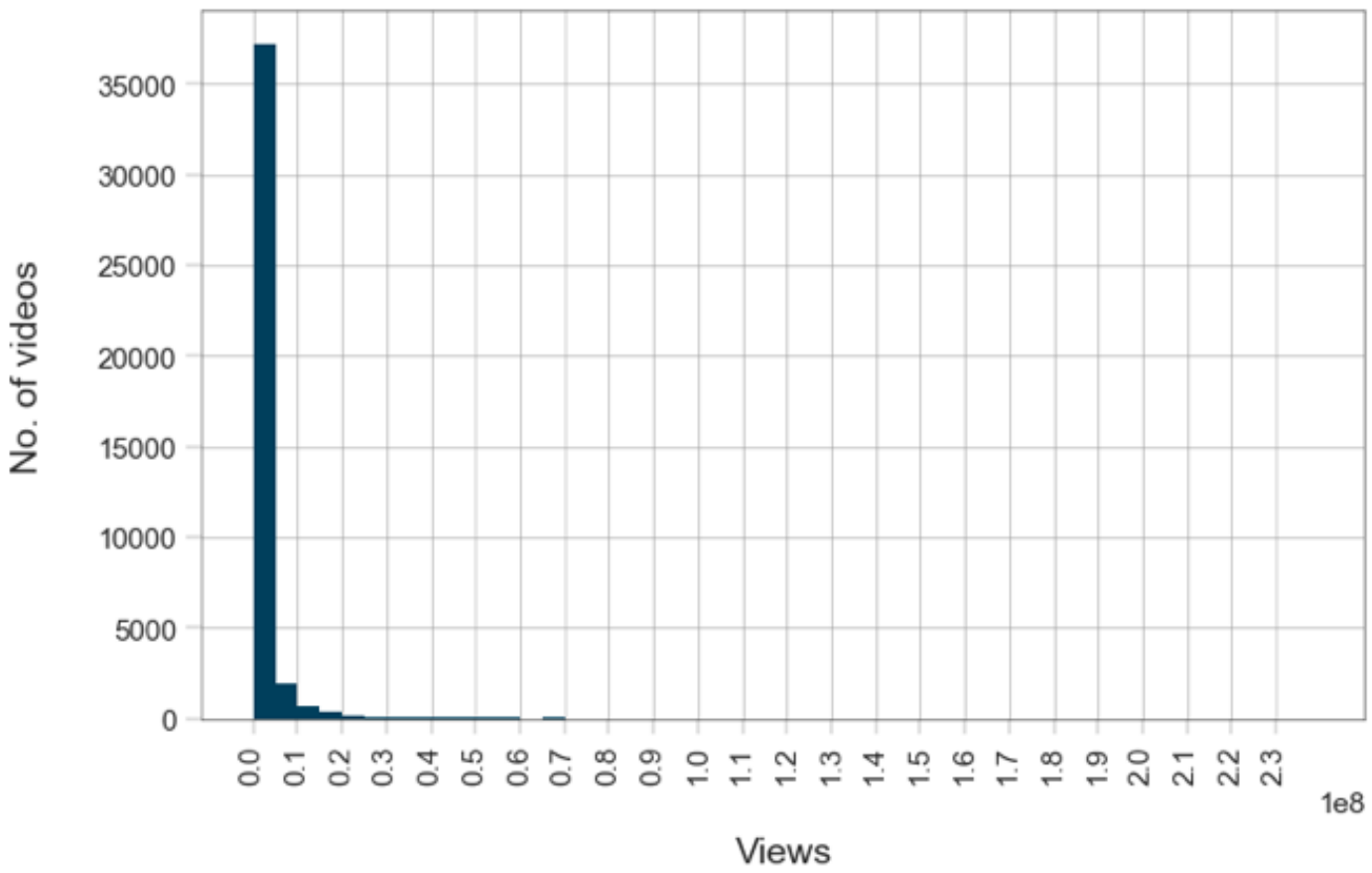
	category_id	views	likes	dislikes	comment_count
count	40949.00	40949.00	40949.00	40949.00	40949.00
mean	19.97	2360784.64	74266.70	3711.40	8446.80
std	7.57	7394113.76	228885.34	29029.71	37430.49
min	1.00	549.00	0.00	0.00	0.00
25%	17.00	242329.00	5424.00	202.00	614.00
50%	24.00	681861.00	18091.00	631.00	1856.00
75%	25.00	1823157.00	55417.00	1938.00	5755.00
max	43.00	225211923.00	5613827.00	1674420.00	1361580.00

average views of a trending video are 5911943 millions, average likes 134519 thousands and dislikes 7612 thousands. Are these statistics representative of the data?

let's plot a histogram for the views column to take a look at its distribution: to see how many videos have between 10 million and 20 million views, how many videos have between 20 million and 30 million views, and so on.

In [16]:

```
fig, ax = plt.subplots()
_ = sns.distplot(df["views"], kde=False, color=PLOT_COLORS[4],
                 hist_kws={'alpha': 1}, bins=np.linspace(0, 2.3e8, 47), ax=ax)
_ = ax.set(xlabel="Views", ylabel="No. of videos", xticks=np.arange(0, 2.4e8,
1e7))
_ = ax.set_xlim(right=2.5e8)
_ = plt.xticks(rotation=90)
```



In [17]:

```
#We note that the vast majority of trending videos have 5 million views or les
s. We get the 5 million number by calculating
(0.1*10e8)/2
```

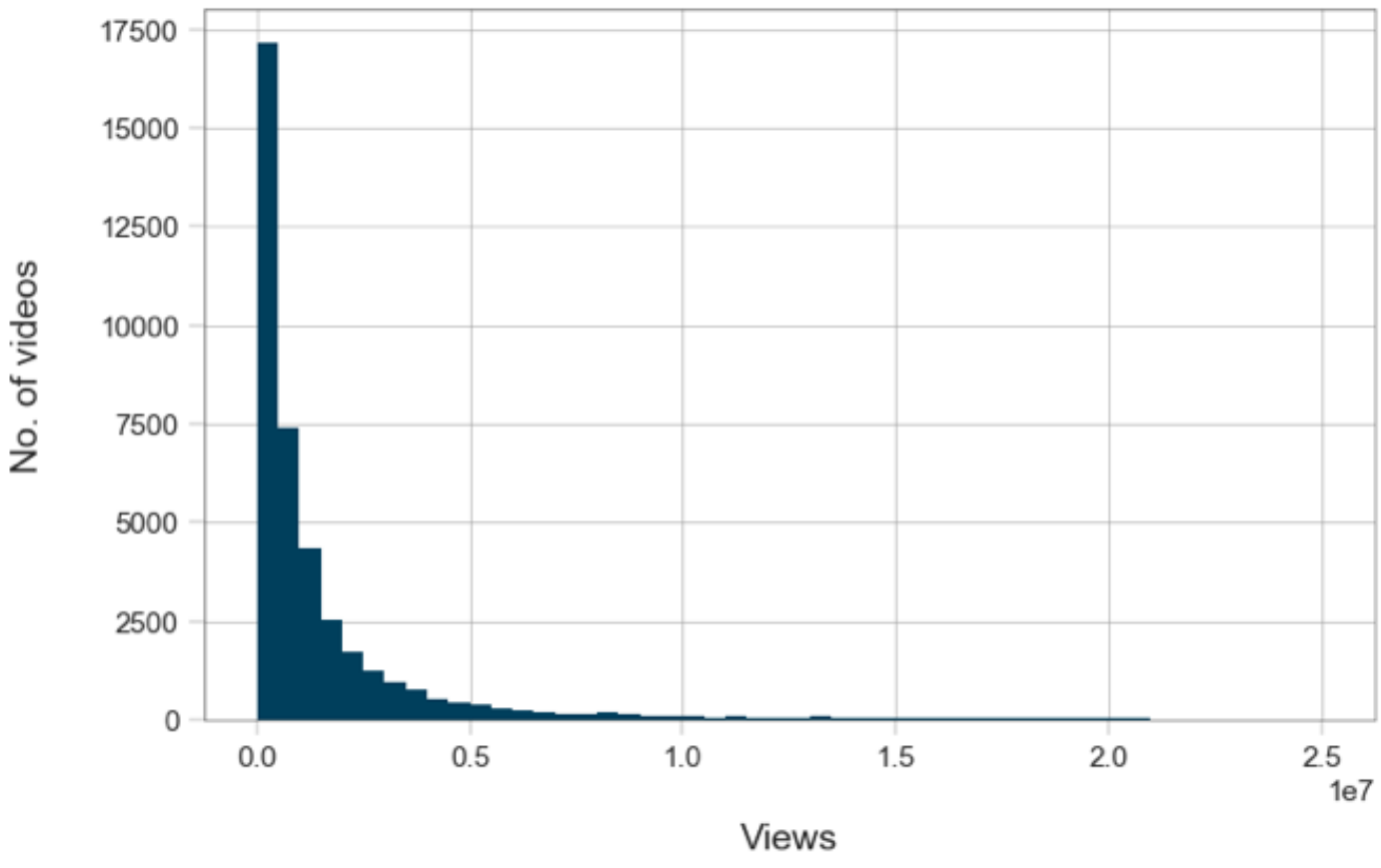
Out[17]:

```
50000000.0
```

Now let us plot the histogram just for videos with 25 million views or less to get a closer look at the distribution of the dat

In [18]:

```
fig, ax = plt.subplots()
_ = sns.distplot(df[df["views"] < 25e6]["views"], kde=False,
                 color=PLOT_COLORS[4], hist_kws={'alpha': 1}, ax=ax)
_ = ax.set(xlabel="Views", ylabel="No. of videos")
```



Now we see that the majority of trending videos have 1 million views or less. Let's see the exact percentage of videos less than 1 million views

In [19]:

```
print(df[df.views < 1e6].views.count() / df.views.count() * 100)
print(df[df.views<1.5e6].views.count() / df.views.count() * 100)
print(df[df.views<5e6].views.count() / df.views.count() * 100)
```

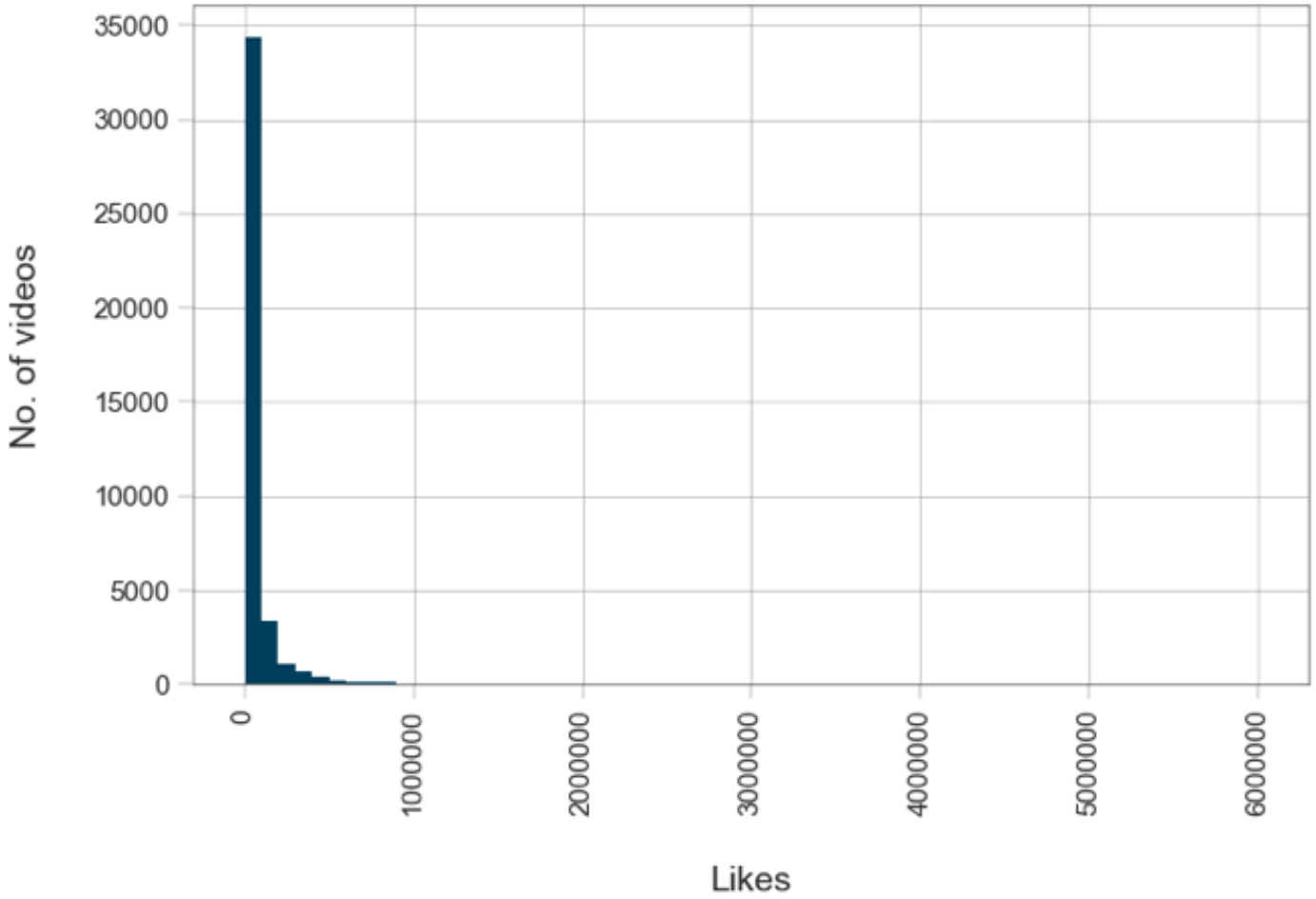
```
60.09426359618062
70.74653837700554
90.81052040342865
```

So, it is around 50%. Similarly, we can see that the percentage of videos with less than 1.5 million views is around 58%, and that the percentage of videos with less than 5 million views is around 79%.

likes histogram

In [20]:

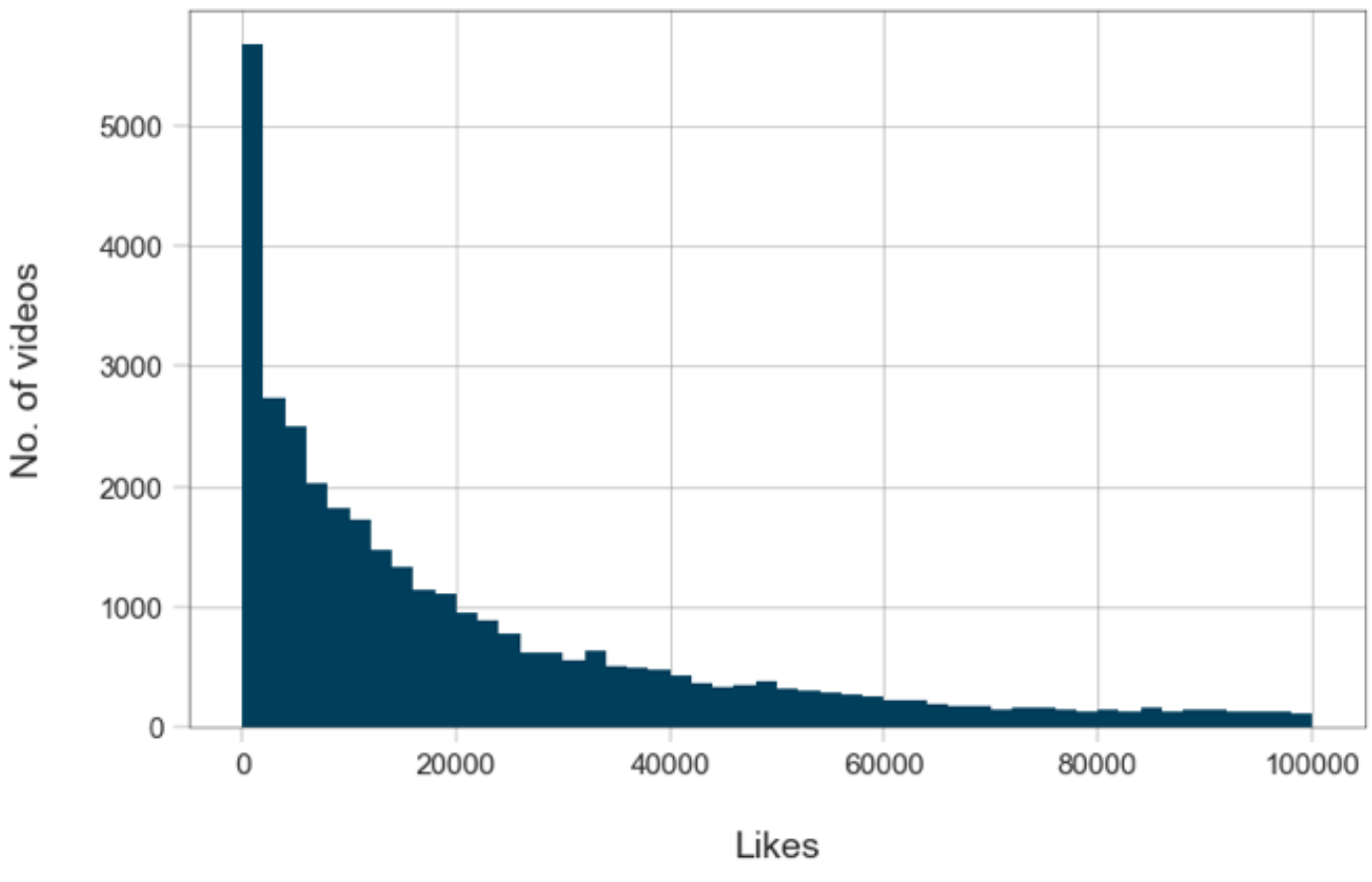
```
plt.rc('figure.subplot', wspace=0.9)
fig, ax = plt.subplots()
_ = sns.distplot(df["likes"], kde=False,
                 color=PLOT_COLORS[4], hist_kws={'alpha': 1},
                 bins=np.linspace(0, 6e6, 61), ax=ax)
_ = ax.set(xlabel="Likes", ylabel="No. of videos")
_ = plt.xticks(rotation=90)
```



We note that the vast majority of trending videos have between 0 and 100,000 likes. Let us plot the histogram just for videos with 1000,000 likes or less to get a closer look at the distribution of the data

In [21]:

```
fig, ax = plt.subplots()
_ = sns.distplot(df[df["likes"] <= 1e5]["likes"], kde=False,
                 color=PLOT_COLORS[4], hist_kws={'alpha': 1}, ax=ax)
_ = ax.set(xlabel="Likes", ylabel="No. of videos")
```



Now we can see that the majority of trending videos have 40000 likes or less with a peak for videos with 2000 likes or less. Let's see the exact percentage of videos with less than 20000 likes.

In [22]:

```
df[df.likes<4e4].likes.count()/df.likes.count() * 100
```

Out[22]:

```
68.4900730176561
```

Description on non-numerical columns

After we described numerical columns previously, we now describe non-numerical columns

In [23]:

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

Out[23]:

	video_id	trending_date	title	channel_title	publish_time	tag
count	40949	40949	40949	40949	40949	409
unique	6351	205	6455	2207	6269	605
top	j4KvrAUjn6c	17.06.12	WE MADE OUR MOM CRY...HER DREAM CAME TRUE!	ESPN	2018-05-18T14:00:04.000Z	[non
freq	30	200	30	203	50	153

there are 205 unique trending dates

From video_id description, we can see that there are 38916 videos (which is expected because our dataset contains 38916 entries), but we can see also that there are only 3369 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 38 times on the trending videos list.

But there is something strange in the description table above: Because there are 3369 unique video IDs, we expect to have 6351 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 3369, but there is nothing strange here, because two different videos may be published at the same time.

o 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 [24]:

```
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[24]:

	video_id	trending_date	title	channel_title	category_id	publisl
14266	0ufNmUyf2co	18.26.01	Here are the weirdest bikes I own - Freak Bike...	Seth's Bike Hacks	26	2018-01-25T00:00:0
14491	0ufNmUyf2co	18.27.01	Here are the weirdest bikes I own - Freak Bike...	Seth's Bike Hacks	26	2018-01-25T00:00:0
14706	0ufNmUyf2co	18.28.01	Bike Check - My freak bikes	Seth's Bike Hacks	26	2018-01-25T00:00:0
14931	0ufNmUyf2co	18.29.01	Bike Check - My freak bikes	Seth's Bike Hacks	26	2018-01-25T00:00:0
15175	0ufNmUyf2co	18.30.01	Bike Check - My freak bikes	Seth's Bike Hacks	26	2018-01-25T00:00:0
15385	0ufNmUyf2co	18.31.01	Bike Check - My freak bikes	Seth's Bike Hacks	26	2018-01-25T00:00:0

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

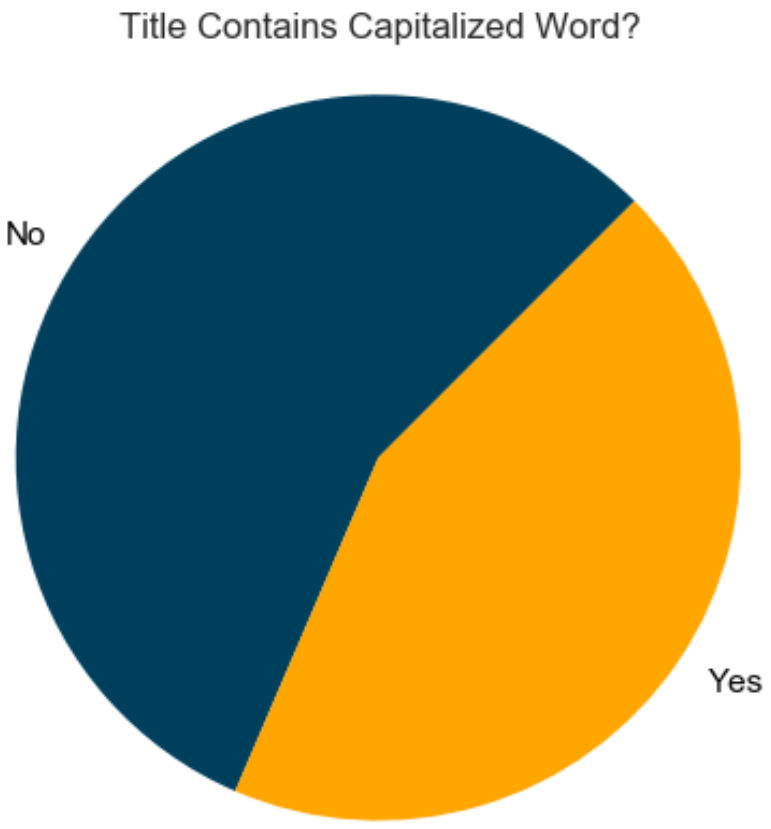
How many trending video titles contain capitalized word?

In [25]:

```
def contains_capitalized_word(s):
    for w in s.split():
        if w.isupper():
            return True
    return False


df["contains_capitalized"] = df["title"].apply(contains_capitalized_word)

value_counts = df["contains_capitalized"].value_counts().to_dict()
fig, ax = plt.subplots()
_ = ax.pie([value_counts[False], value_counts[True]], labels=['No', 'Yes'],
           colors=['#003f5c', '#ffa600'], textprops={'color': '#040204'}, star
tangle=45)
_ = ax.axis('equal')
_ = ax.set_title('Title Contains Capitalized Word?')
```



In [26]:

```
df["contains_capitalized"].value_counts(normalize=True)
```

Out[26]:

```
False   0.56
True    0.44
Name: contains_capitalized, dtype: float64
```

We can see that 47% of trending video titles contain at least a capitalized word. We will later use this added new column contains_capitalized in analyzing correlation between variables.

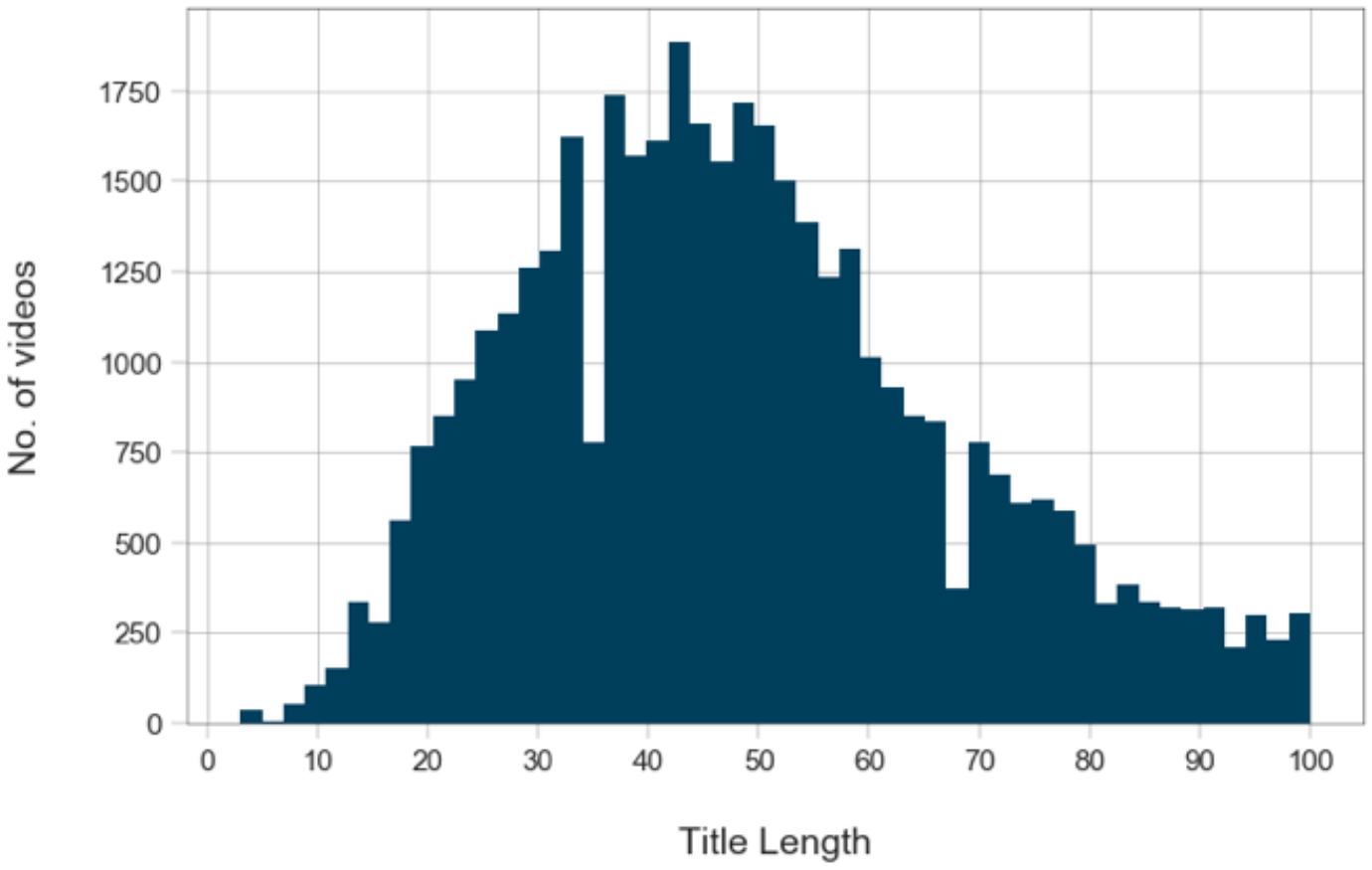
Video title lengths

Let's add another column to our dataset to represent the length of each video title, then plot the histogram of title length to get an idea about the lengths of trnding video titles

In [27]:

```
df['title_length'] = df.title.apply(lambda x: len(x))

fig, ax = plt.subplots()
_ = sns.distplot(df["title_length"], kde=False, rug=False,
                 color=PLOT_COLORS[4], hist_kws={'alpha': 1}, ax=ax)
_ = ax.set(xlabel="Title Length", ylabel="No. of videos", xticks=range(0, 110,
10))
```

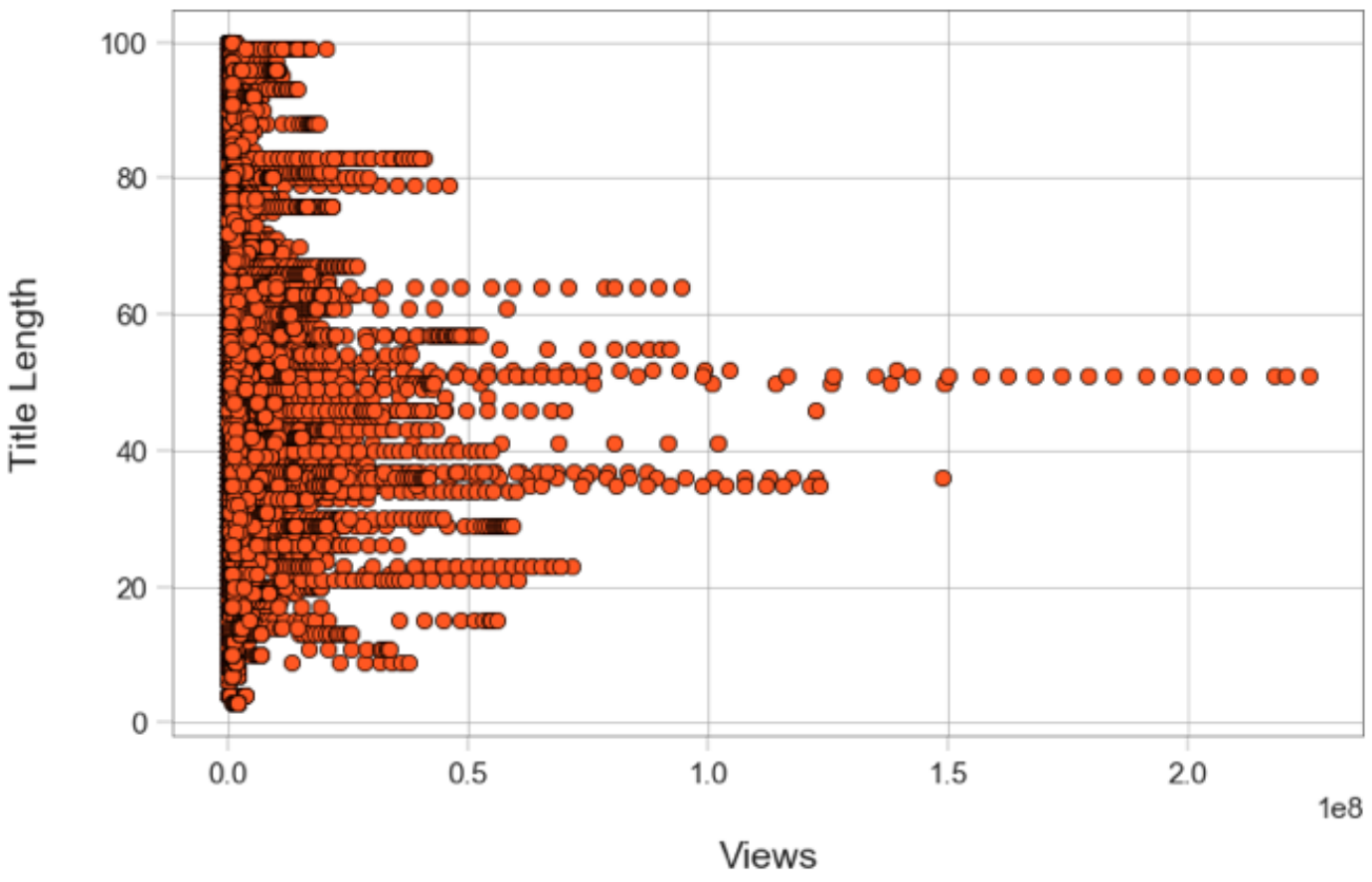


We can see that title-length distribution resembles a normal distribution, where most videos have title lengths between 25 and 55 character approximately.

Now let's draw a scatter plot between title length and number of views to see the relationship between these two variables

In [28]:

```
fig, ax = plt.subplots()
_ = ax.scatter(x=df['views'], y=df['title_length'], color=PLOT_COLORS[2], edge
colors="#000000", linewidths=0.5)
_ = ax.set(xlabel="Views", ylabel="Title Length")
```



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 of 20, 40, 50, 75 and 90 characters approximately. Which doesn't give much info..

Correlation between dataset variables

Now let's see how the dataset variables are correlated with each other: for example, we would like to see how views and likes are correlated, meaning do views and likes increase and decrease together (positive correlation)? Does one of them increase when the other decrease and vice versa (negative correlation)? Or are they not correlated?

Correlation is represented as a value between -1 and +1 where +1 denotes the highest positive correlation, -1 denotes the highest negative correlation, and 0 denotes that there is no correlation.

Let's see the correlation table between our dataset variables (numerical and boolean variables only)

In [29]:

```
df.corr()
```

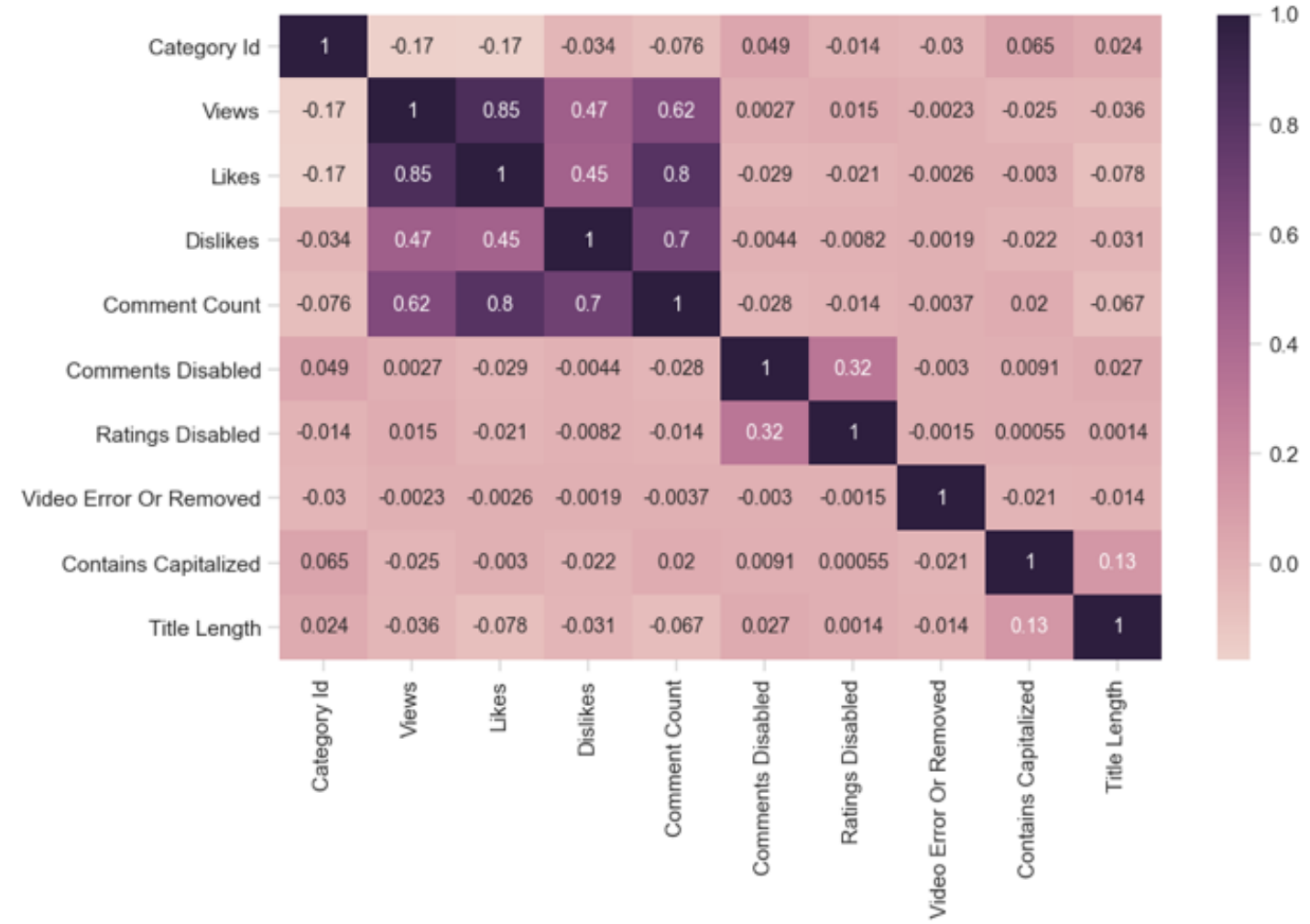
Out[29]:

	category_id	views	likes	dislikes	comment_count	comm
category_id	1.00	-0.17	-0.17	-0.03	-0.08	0.05
views	-0.17	1.00	0.85	0.47	0.62	0.00
likes	-0.17	0.85	1.00	0.45	0.80	-0.03
dislikes	-0.03	0.47	0.45	1.00	0.70	-0.00
comment_count	-0.08	0.62	0.80	0.70	1.00	-0.03
comments_disabled	0.05	0.00	-0.03	-0.00	-0.03	1.00
ratings_disabled	-0.01	0.02	-0.02	-0.01	-0.01	0.32
video_error_or_removed	-0.03	-0.00	-0.00	-0.00	-0.00	-0.00
contains_capitalized	0.06	-0.03	-0.00	-0.02	0.02	0.01
title_length	0.02	-0.04	-0.08	-0.03	-0.07	0.03

In [30]:

```
h_labels = [x.replace('_', ' ').title() for x in
            list(df.select_dtypes(include=['number', 'bool']).columns.values)]

fig, ax = plt.subplots(figsize=(10,6))
_ = sns.heatmap(df.corr(), annot=True, xticklabels=h_labels, yticklabels=h_lab
els, cmap=sns.cubehelix_palette(as_cmap=True), ax=ax)
```



The correlation map and correlation table above say that views and likes are highly positively correlated. Let's verify that by plotting a scatter plot between views and likes to visualize the relationship between these variables

We see that views and likes are truly positively correlated: as one increases, the other increases too—mostly.

Another verification of the correlation matrix and map is the scatter plot we drew above between views and title length as it shows that there is no correlation between them.

Most common words in video titles

Let's see if there are some words that are used significantly in trending video titles. We will display the 25 most common words in all trending video titles

In [31]:

```
title_words = list(df["title"].apply(lambda x: x.split()))
title_words = [x for y in title_words for x in y]
Counter(title_words).most_common(25)
```

Out[31]:

```
[('-', 11452),
 ('|', 10663),
 ('The', 5762),
 ('the', 3610),
 ('a', 2566),
 ('to', 2343),
 ('of', 2338),
 ('in', 2176),
 ('A', 2122),
 ('&', 2024),
 ('I', 1940),
 ('and', 1917),
 ('Video)', 1901),
 ('Trailer', 1868),
 ('How', 1661),
 ('with', 1655),
 ('2018', 1613),
 ('(Official', 1594),
 ('Official', 1554),
 ('on', 1552),
 ('To', 1397),
 ('You', 1254),
 ('My', 1080),
 ('for', 1020),
 ('ft.', 1017)]
```

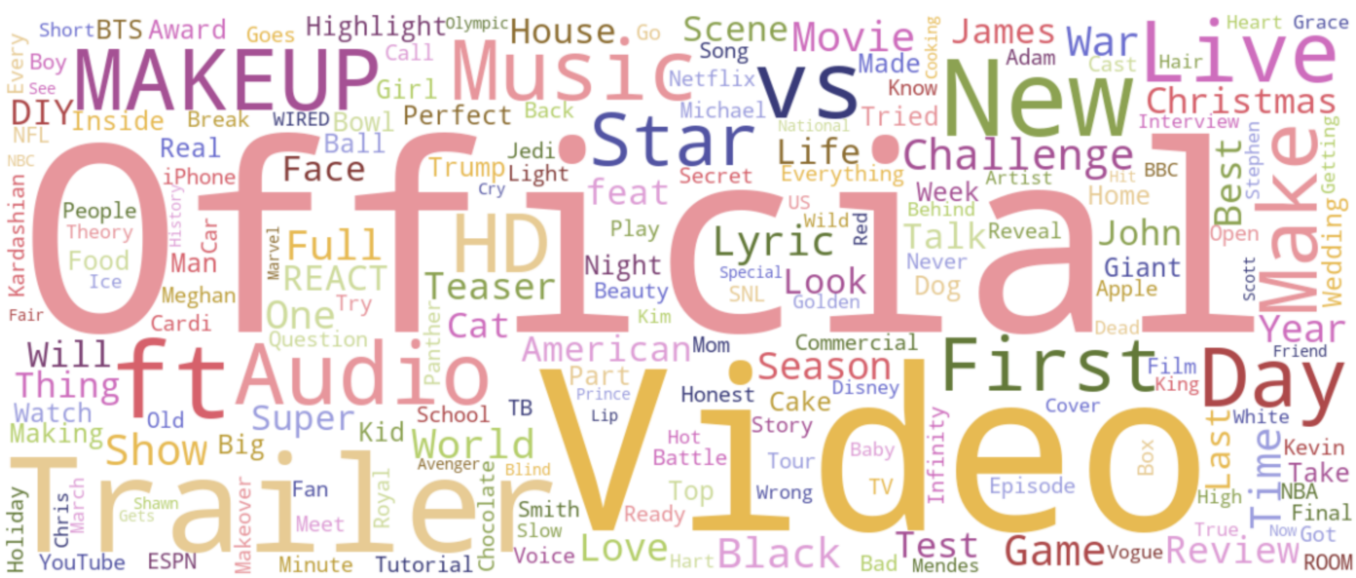
We can see that "-" and "|" symbols occured a lot trending video titles.

NB here you could use ML to remove "to","in" etc and vectorize to clearly analyse the text through NLP!

Let's draw a word cloud for the titles of our trending videos, which is a way to visualize most common words in the titles; the more common the word is, the bigger its font size is

In [32]:

```
# wc = wordcloud.WordCloud(width=1200, height=600, collocations=False, stopwor
ds=None, background_color="white", colormap="tab20b").generate_from_frequencie
s(dict(Counter(title_words).most_common(150)))
wc = wordcloud.WordCloud(width=1200, height=500,
                         collocations=False, background_color="white",
                         colormap="tab20b").generate(" ".join(title_words))
plt.figure(figsize=(15,10))
plt.imshow(wc, interpolation='bilinear')
_ = plt.axis("off")
```



Most common video using ML NLP by Giovanni

In [33]:

```
#df.title
```

In [34]:

```
from sklearn.feature_extraction.text import CountVectorizer

vect = CountVectorizer(stop_words='english', max_df=0.95)

# learn the 'vocabulary' of the training data (occurs in-place)
# learn vocabolary of training data
X = df.title
sparse_X = vect.fit_transform(X)

#vect.get_feature_names()
```

In [35]:

```
matrix = pd.DataFrame(sparse_X.toarray(), columns=vect.get_feature_names())
```

In [36]:

```
matrix.sum().sort_values(ascending=False).head(100)
```

Out[36]:

```
Out[30]:
official     3991
video        2864
2018         2295
trailer      2000
ft           1301
vs           1046
2017         1011
new          1008
makeup        876
audio         872
music         838
hd            796
live          790
day           697
10            672
challenge     670
game          604
black         593
star          572
world         571
lyric         569
make          569
time          562
love          535
movie         509
season        482
best          468
teaser        462
christmas     443
like          436
              ...
week          267
beauty        266
tutorial      263
cardi         263
snl           261
wedding       260
people        259
trailers      258
episode       257
got           251
box           251
youtube       250
voice         247
kids          246
questions     245
ball          240
war           239
apple         238
does          232
espn          229
cast          227
nba           227
honest        226
smith         224
jedi          224
tour          223
```

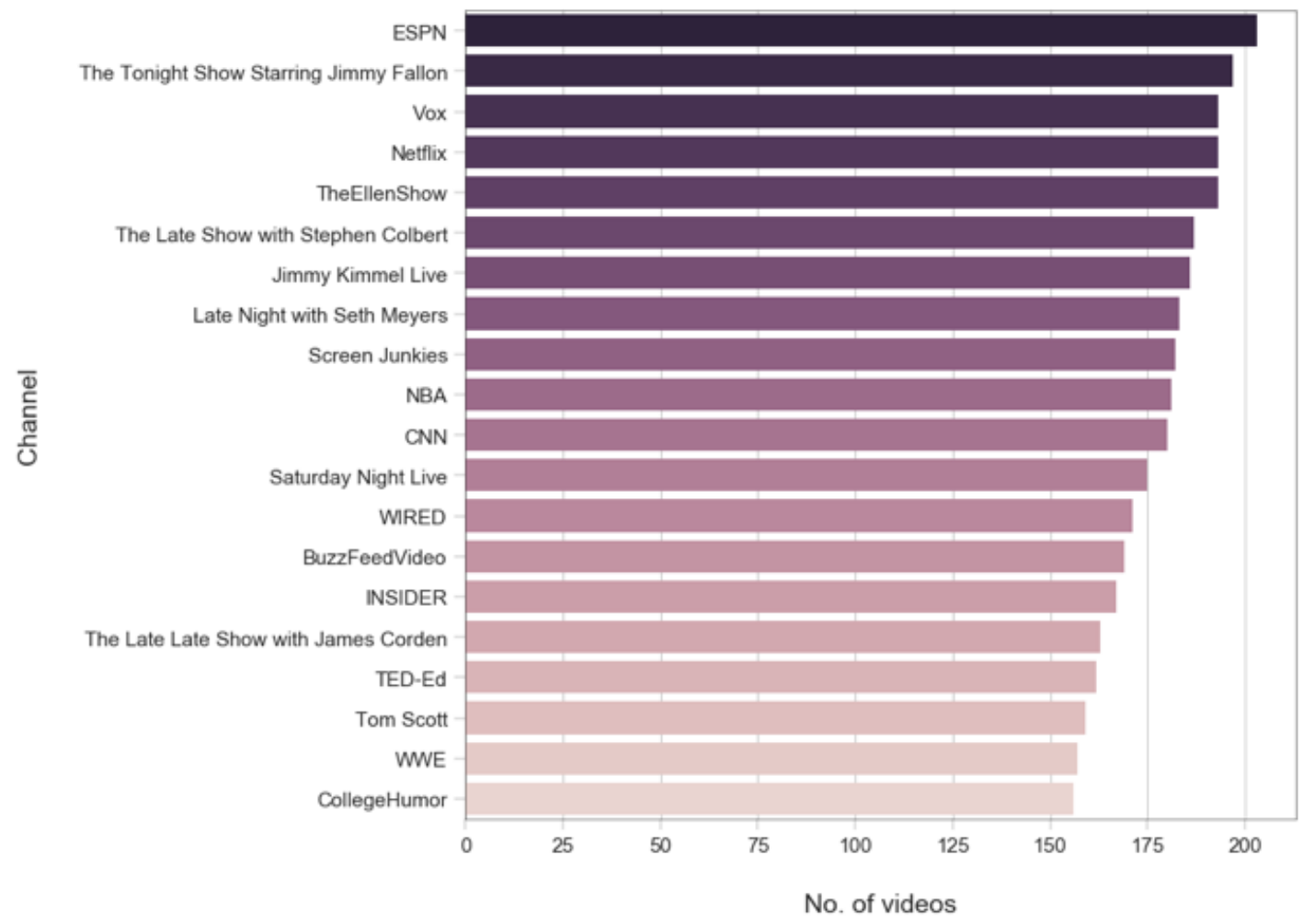
```
tour            223
mom             222
old             222
bad             222
netflix         222
Length: 100, dtype: int64
```

Which channel have the largest number of trending videos?

In [37]:

```
cdf = df.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")
```



Which video category has the largest number of trending videos?

First, we will add a column that contains category names based on the values in category_id column. We will use a category JSON file provided with the dataset which contains information about each category.

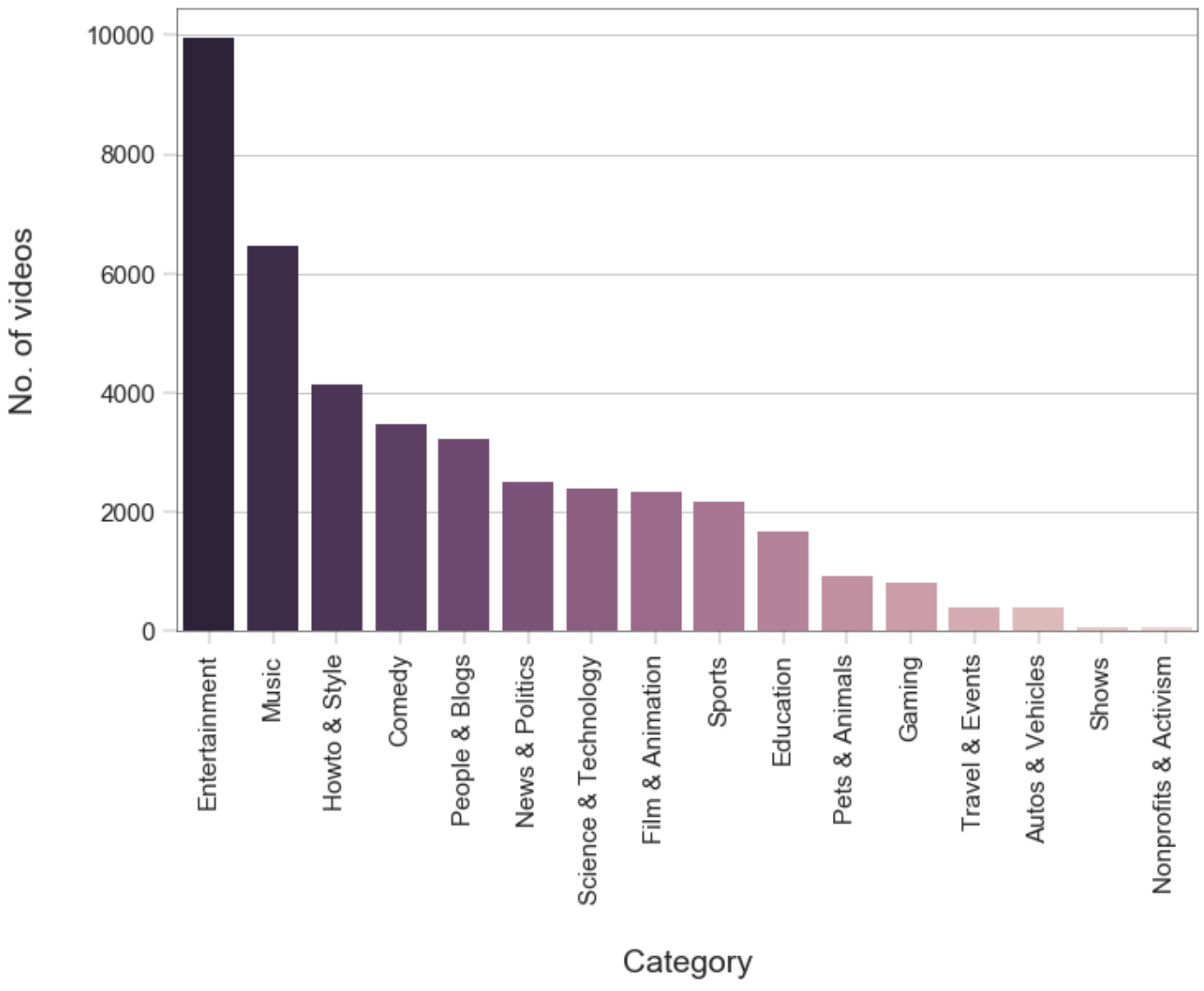
In [38]:

```
with open(json_string) as f:
    categories = json.load(f)["items"]
cat_dict = {}
for cat in categories:
    cat_dict[int(cat["id"])] = cat["snippet"]["title"]
df['category_name'] = df['category_id'].map(cat_dict)
```

Now we can see which category had the largest number of trending videos

In [39]:

```
cdf = df["category_name"].value_counts().to_frame().reset_index()
cdf.rename(columns={"index": "category_name", "category_name": "No_of_videos"}
, inplace=True)
fig, ax = plt.subplots()
_ = sns.barplot(x="category_name", y="No_of_videos", data=cdf,
                palette=sns.cubehelix_palette(n_colors=16, reverse=True), ax=a
x)
_ = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
_ = ax.set(xlabel="Category", ylabel="No. of videos")
```



We see that the Music category contains the largest number of trending videos among other categories: around 14,000 videos, followed by Entertainment category with around 8,500 videos, followed by People & Blogs category with around 2,500 videos, and so on.

Trending videos and their publishing time

An example value of the publish_time column in our dataset is 2017-11-13T17:13:01.000Z. And according to information on this page: https://www.w3.org/TR/NOTE-datetime (https://www.w3.org/TR/NOTE-datetime), this means that the date of publishing the video is 2017-11-13 and the time is 17:13:01 in Coordinated Universal Time (UTC) time zone.

Let's add two columns to represent the date and hour of publishing each video, then delete the original publish_time column because we will not need it anymore

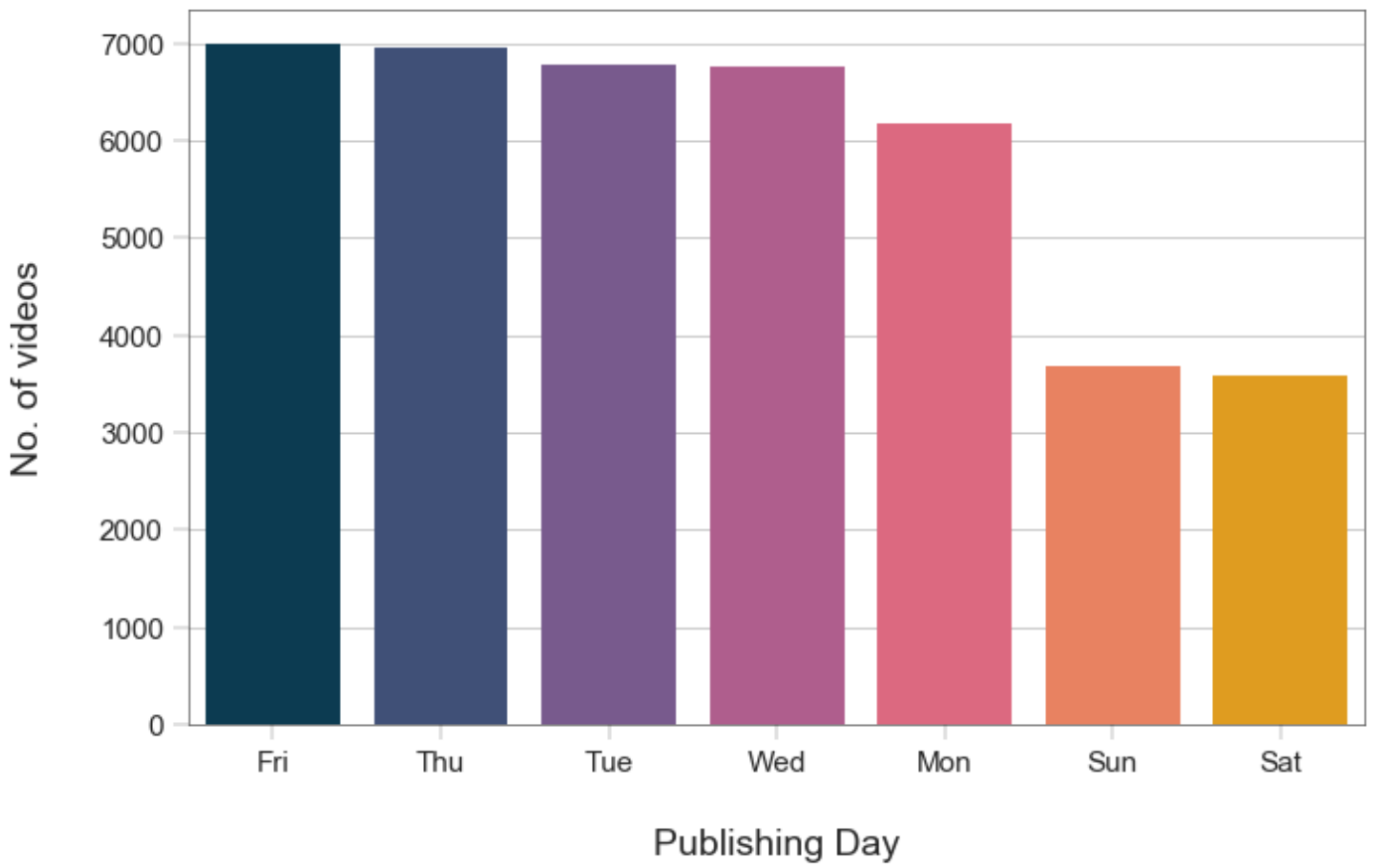
In [40]:

```
df["publishing_day"] = df["publish_time"].apply(
    lambda x: datetime.datetime.strptime(x[:10], "%Y-%m-%d").date().strftime('
%a'))
df["publishing_hour"] = df["publish_time"].apply(lambda x: x[11:13])
df.drop(labels='publish_time', axis=1, inplace=True)
```

Now we can see which days of the week had the largest numbers of trending videos

In [41]:

```
cdf = df["publishing_day"].value_counts()\
        .to_frame().reset_index().rename(columns={"index": "publishing_day", "
publishing_day": "No_of_videos"})
fig, ax = plt.subplots()
_ = sns.barplot(x="publishing_day", y="No_of_videos", data=cdf,
                palette=sns.color_palette(['#003f5c', '#374c80', '#7a5195',
                                           '#bc5090', '#ef5675', '#ff764a', '#
ffa600'], n_colors=7), ax=ax)
_ = ax.set(xlabel="Publishing Day", ylabel="No. of videos")
```

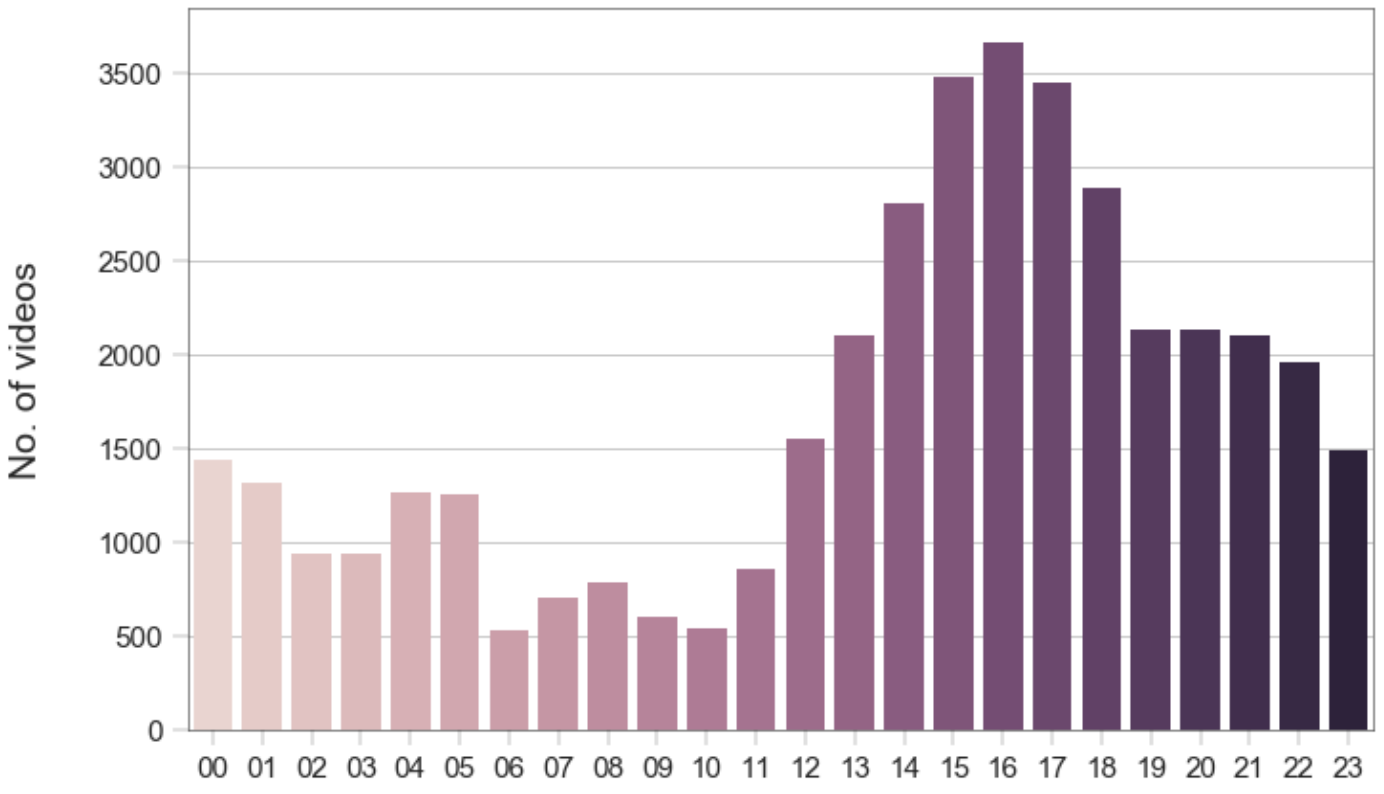


We can see that the number of trending videos published on Sunday and Saturday are noticeably less than the number of trending videos published on other days of the week.

Now let's use publishing_hour column to see which publishing hours had the largest number of trending videos

In [42]:

```
cdf = df["publishing_hour"].value_counts().to_frame().reset_index()\
        .rename(columns={"index": "publishing_hour", "publishing_hour": "No_of
_videos"})
fig, ax = plt.subplots()
_ = sns.barplot(x="publishing_hour", y="No_of_videos", data=cdf,
                palette=sns.cubehelix_palette(n_colors=24), ax=ax)
_ = ax.set(xlabel="Publishing Hour", ylabel="No. of videos")
```



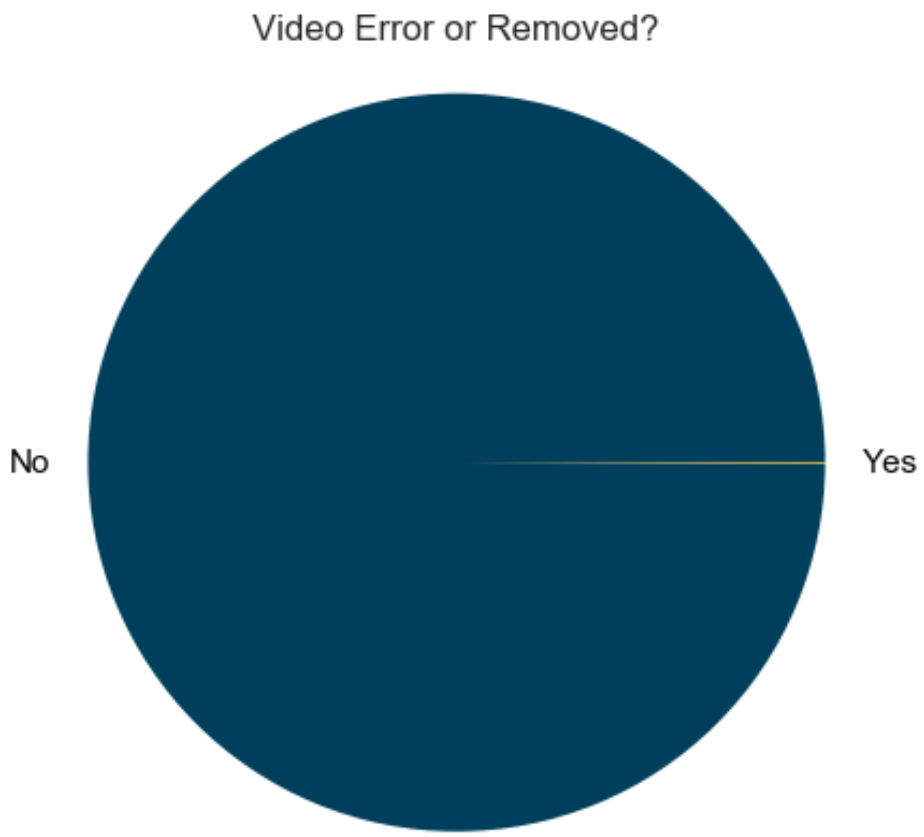
We can see that the period between 2PM and 7PM, peaking between 4PM and 5PM, had the largest number of trending videos. We notice also that the period between 12AM and 1PM has the smallest number of trending videos. But why is that? Is it because people publish a lot more videos between 2PM and 7PM? Is it because how YouTube algorithm chooses trending videos?

How many trending videos have an error?

To see how many trending videos got removed or had some error, we can use video_error_or_removed column in the dataset

In [43]:

```
value_counts = df["video_error_or_removed"].value_counts().to_dict()
fig, ax = plt.subplots()
_ = ax.pie([value_counts[False], value_counts[True]], labels=['No', 'Yes'],
        colors=['#003f5c', '#ffa600'], textprops={'color': '#040204'})
_ = ax.axis('equal')
_ = ax.set_title('Video Error or Removed?')
```



In [44]:

```
df["video_error_or_removed"].value_counts()
```

Out[44]:

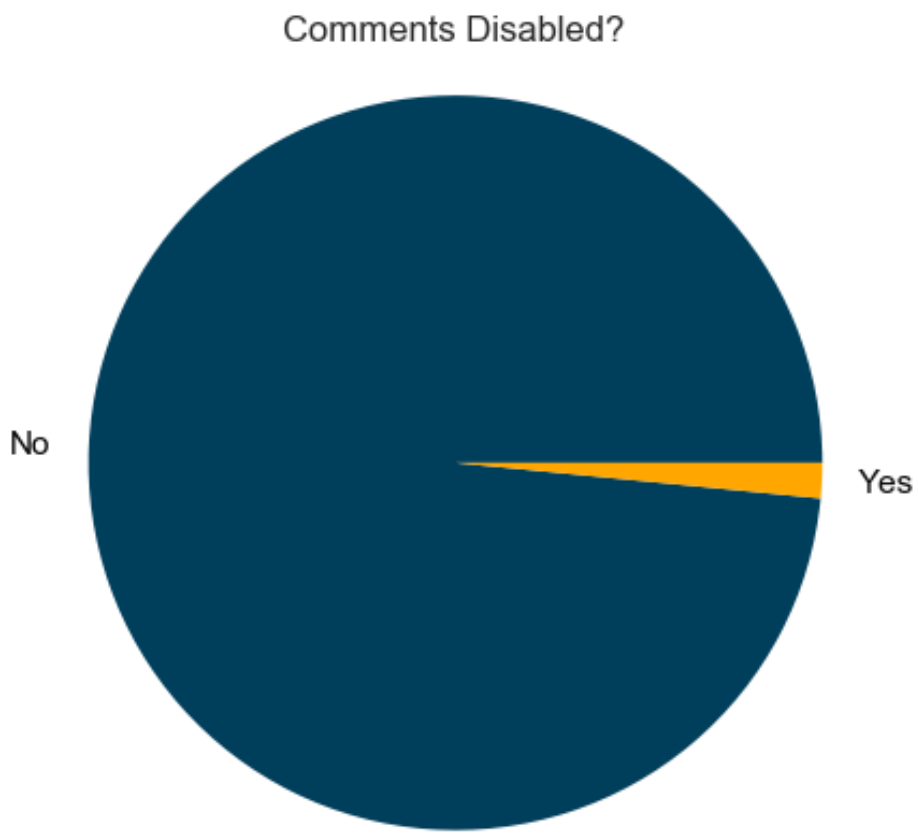
```
False    40926
True        23
Name: video_error_or_removed, dtype: int64
```

We can see that out of videos that appeared on trending list (38847 videos), there is a tiny portion (69 videos) with errors.

How many trending videos have their comments disabled?

In [45]:

```
value_counts = df["comments_disabled"].value_counts().to_dict()
fig, ax = plt.subplots()
_ = ax.pie(x=[value_counts[False], value_counts[True]], labels=['No', 'Yes'],
           colors=['#003f5c', '#ffa600'], textprops={'color': '#040204'})
_ = ax.axis('equal')
_ = ax.set_title('Comments Disabled?')
```



In [46]:

```
df["comments_disabled"].value_counts(normalize=True)
```

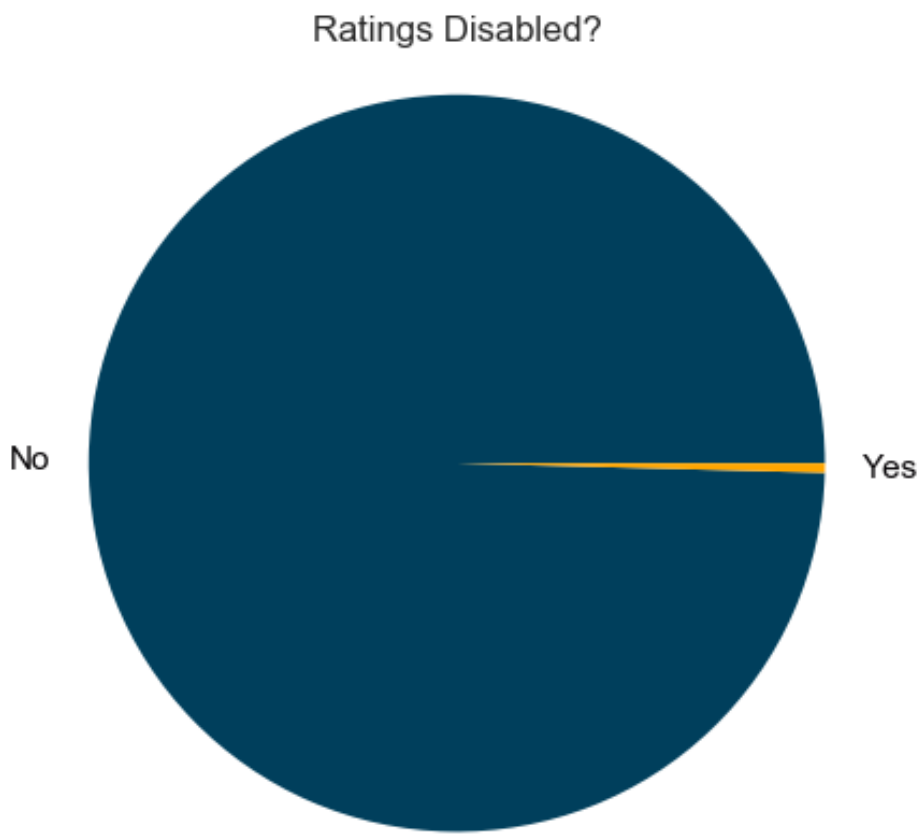
Out[46]:

```
False   0.98
True    0.02
Name: comments_disabled, dtype: float64
```

How many trending videos have their ratings disables?

In [47]:

```
value_counts = df["ratings_disabled"].value_counts().to_dict()
fig, ax = plt.subplots()
_ = ax.pie([value_counts[False], value_counts[True]], labels=['No', 'Yes'],
           colors=['#003f5c', '#ffa600'], textprops={'color': '#040204'})
_ = ax.axis('equal')
_ = ax.set_title('Ratings Disabled?')
```



In [48]:

```
df["ratings_disabled"].value_counts()
```

Out[48]:

```
False    40780
True       169
Name: ratings_disabled, dtype: int64
```

Only 272 has ratings disables

How many videos have both comments and ratings disabled?

In [49]:

```
len(df[(df["comments_disabled"] == True) & (df["ratings_disabled"] == True)].i
ndex)
```

Out[49]:

```
106
```

224 videos have ratings and comments disabled

Conclusions

NB: if you create a variable where you store the value of your analysis, you can run this notebook for all countries without having the trouble to actually analyse all of it and just focus here to what matters in the conclusion...Sophia is hungry and she wouldn't let me do it...the numbers in this notebook are from different countries...just look at the data imported and rely on the graphs for the numbers!

--> this is just an example of how to write the conclusions

Here are the some of the results we extracted from the analysis:

- We analyzed a dataset that contains information about YouTube trending videos for 205 days. The dataset was collected in 2017 and 2018. It contains 40949 video entry.
- 71% of trending videos have less than 1.5 million views, and 91% have less than 5 million views.
- 68% of trending videos have less than 40,000 likes, and 84% have less than 100,000 likes.
- 67% of trending videos have less than 4,000 comments, and 93% have less than 25,000 comments.
- Some videos may appear on the trending videos list on more than one day. Our dataset contains 40494 entries but not for 40494 unique videos but for 6351unique videos.
- Trending videos that have 100,000,000 views and more have title length between 33 and 55 characters approximately.
- The delimiters - and | were common in trending video titles.
- The words 'Official', 'Video', 'Trailer', 'How', and '2018' were common also in trending video titles.
- There is a strong positive correlation between the number of views and the number of likes of trending videos: As one of them increases, the other increases, and vice versa.
- There is a strong positive correlation also between the number of likes and the number of comments, and a slightly weaker one between the number of dislikes and the number of comments.
- The category that has the largest number of trending videos is 'Entertainment' with 9,964 videos, followed by 'Music' category with 6,472 videos, followed by 'Howto & Style' category with 4146 videos.
- On the opposite side, the category that has the smallest number of trending videos is 'Shows' with 57 videos, followed by 'Nonprofits & Activisim' with 57 videos, followed by 'Autos & Vehicles' with 384 videos.