```
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

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

# Configuring some options
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
# If you want interactive plots, uncomment the next line
# %matplotlib notebook

df = pd.read_csv("../input/USvideos.csv")
```

We set some configuration options just for improving visualization graphs; nothing crucial

```
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, labelsize=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)

df.head()
```

```
      video_id                            ...
description
0  2kyS6SvSYSE                            ...
SHANTELL'S CHANNEL - https://www.youtube.com/s...
1  1ZAPwfrtAFY                            ...                                One
year after the presidential election, John...
2  5qpjK5DgCt4                            ...
WATCH MY PREVIOUS VIDEO ▶ \n\nSUBSCRIBE ► http...
3  puqaWrEC7tY                            ...
```

```
Today we find out if Link is a Nickelback amat...
4  d380meD0W0M                          ...                          I
know it's been a while since we did this sho...

[5 rows x 16 columns]
```

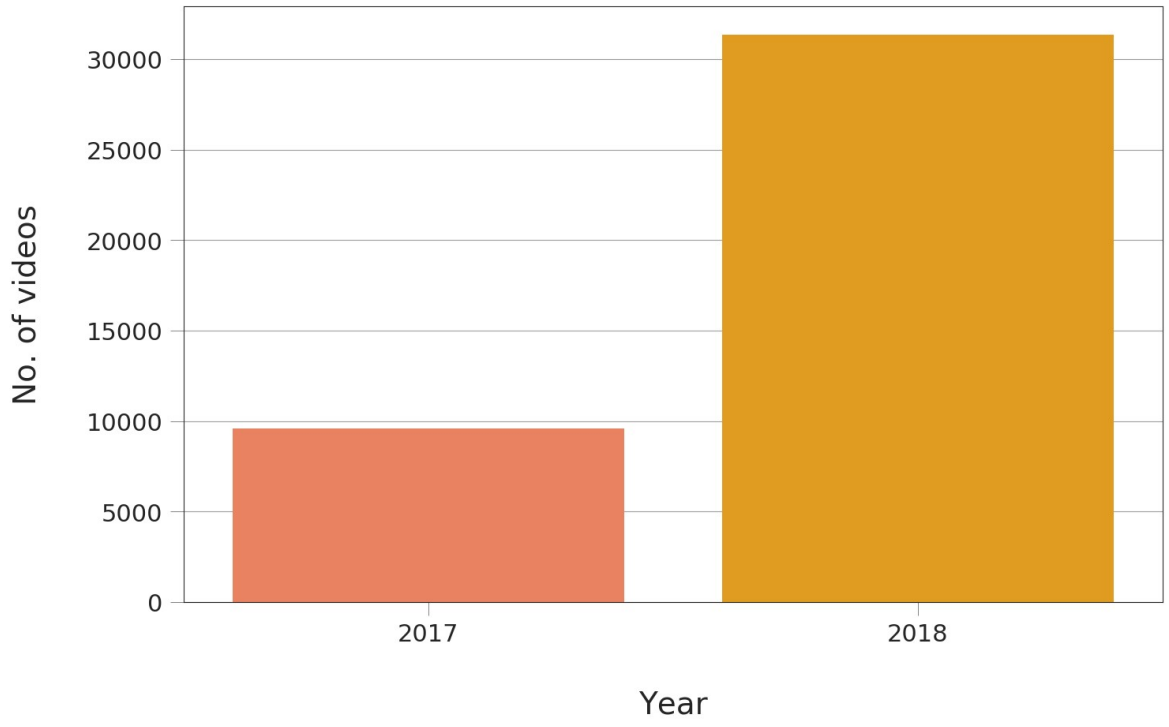
Now, let's see some information about our dataset

```
df.info()

<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

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

        video_id trending_date     ...     video_error_or_removed
description
42   NZFhMSgbKKM      17.14.11     ...                      False
NaN
47   sbcbvuitiTc      17.14.11     ...                      False
NaN
175  4d07RXYLsJE      17.14.11     ...                      False
NaN

[3 rows x 16 columns]
```

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

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

```
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")
```




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

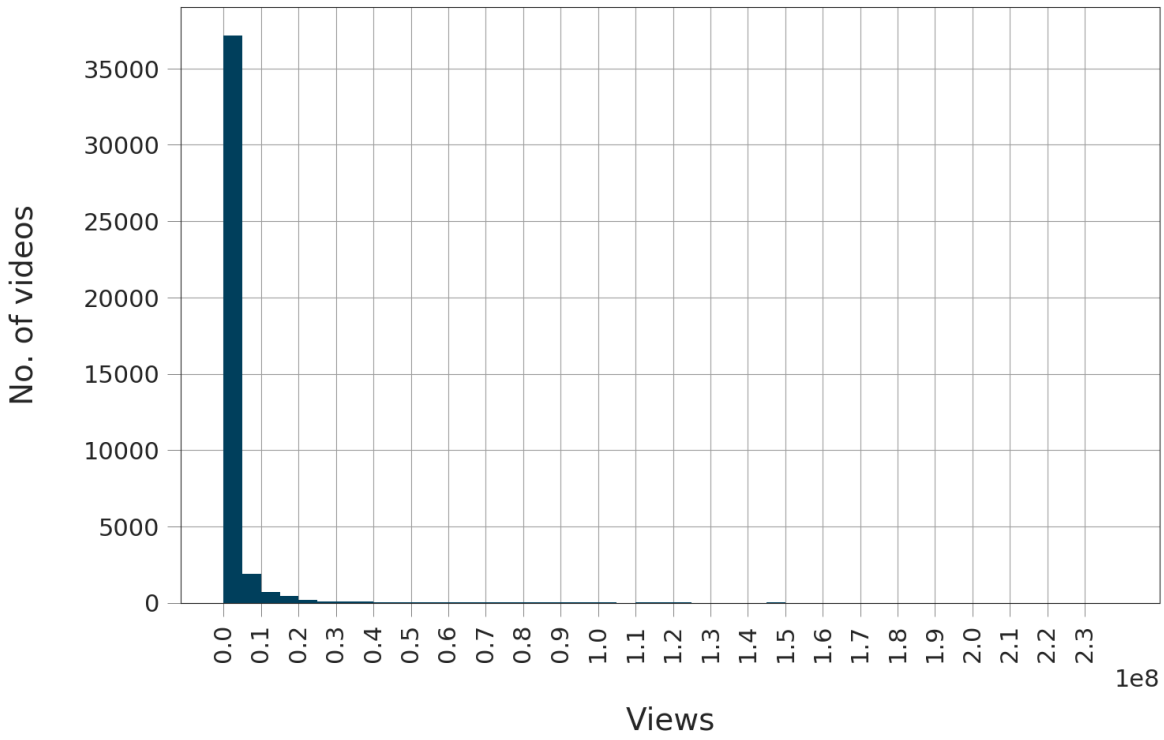
```
2018   0.77
2017   0.23
Name: trending_date, dtype: float64
```

We can see that the dataset was collected in 2017 and 2018 with `77%` of it in 2018 and `23%` in 2017.

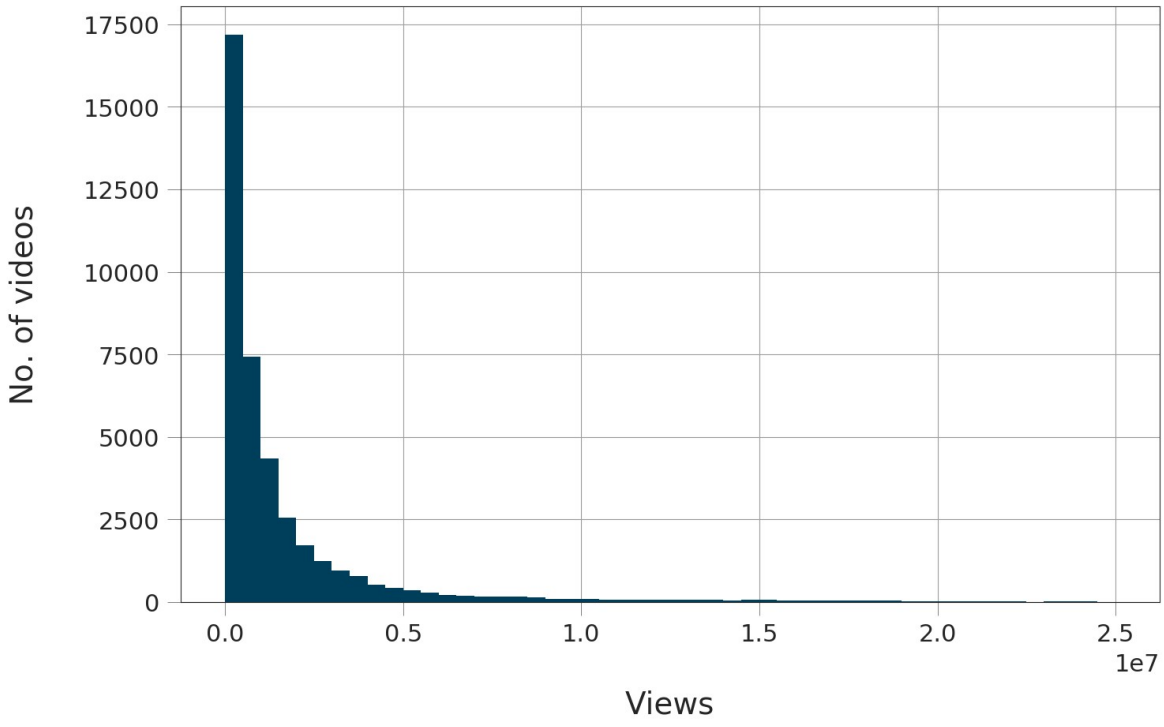
```
df.describe()
```

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

[8 rows x 5 columns]
```

```
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)
```



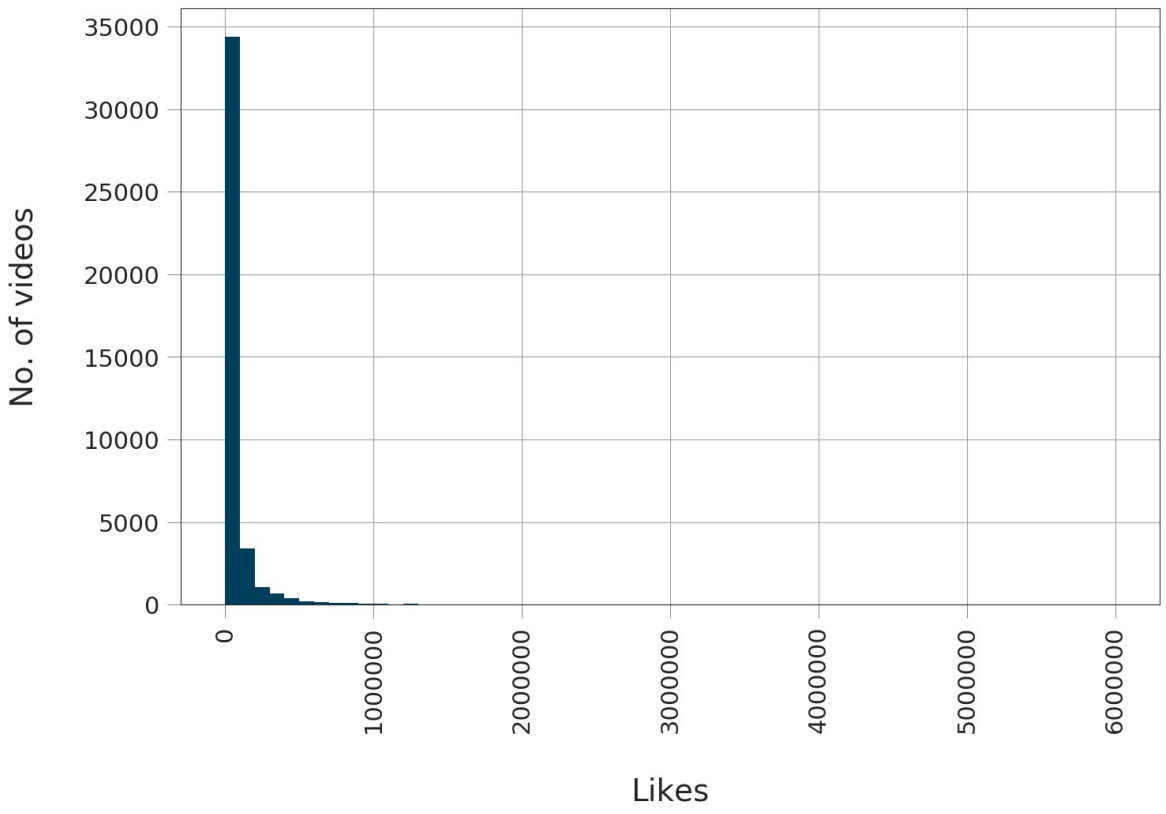

```
fig, ax = plt.subplots()
_ = sns.distplot(df[df["views"] < 25e6]["views"], kde=False,
                 color=PLOT_COLORS[4], hist_kws={'alpha': 1}, ax=ax)
_ = ax.set(xlabel="Views", ylabel="No. of videos")
```



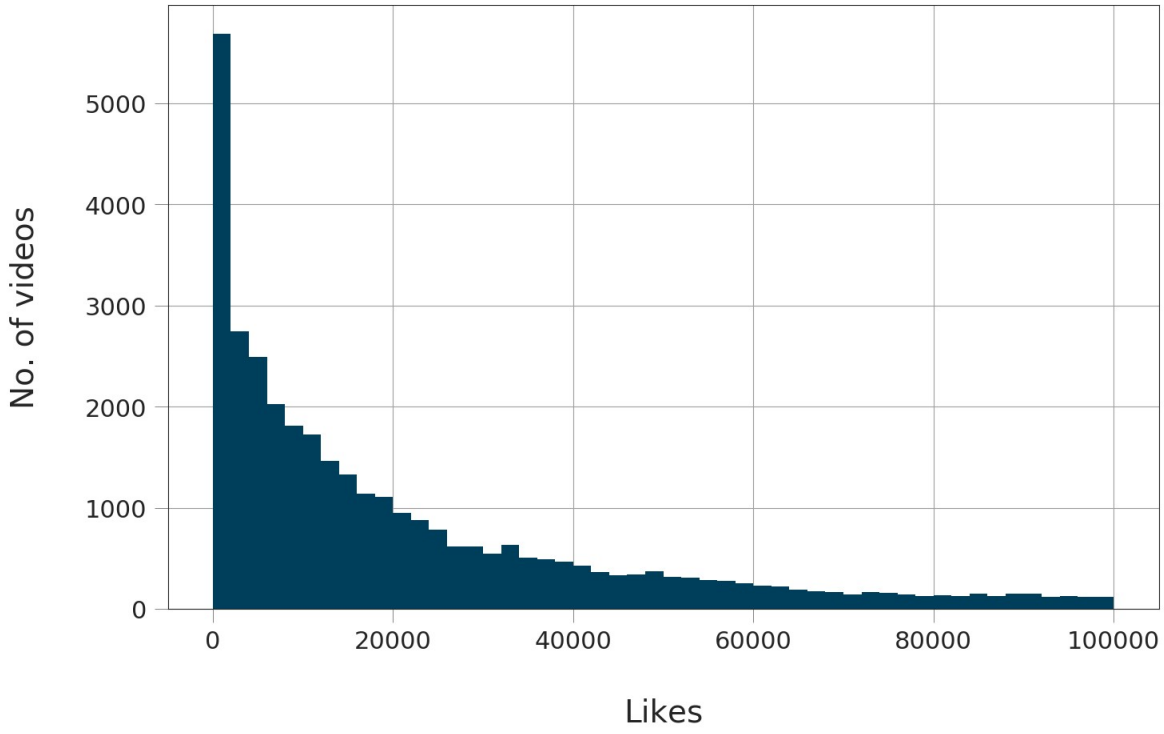

```
df[df['views'] < 1e6]['views'].count() / df['views'].count() * 100
```

```
60.09426359618062
```

```
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)
```



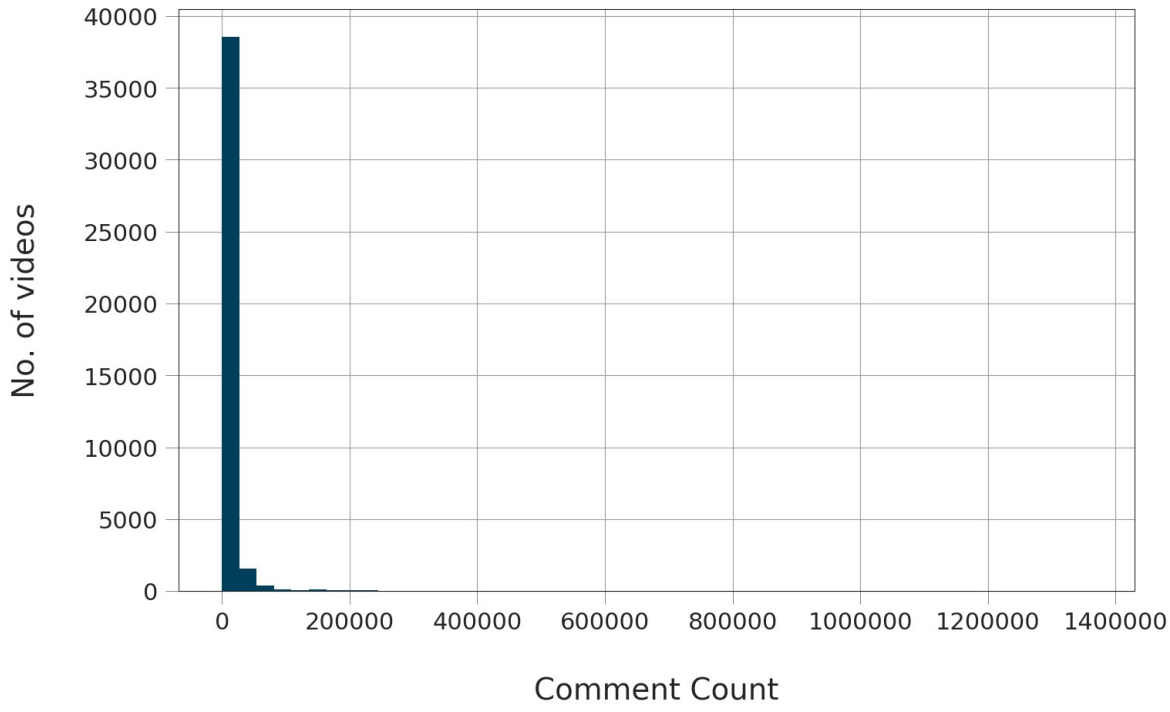

```
fig, ax = plt.subplots()
_ = sns.distplot(df[df["likes"] <= 1e5]["likes"], kde=False,
                 color=PLOT_COLORS[4], hist_kws={'alpha': 1}, ax=ax)
_ = ax.set(xlabel="Likes", ylabel="No. of videos")
```



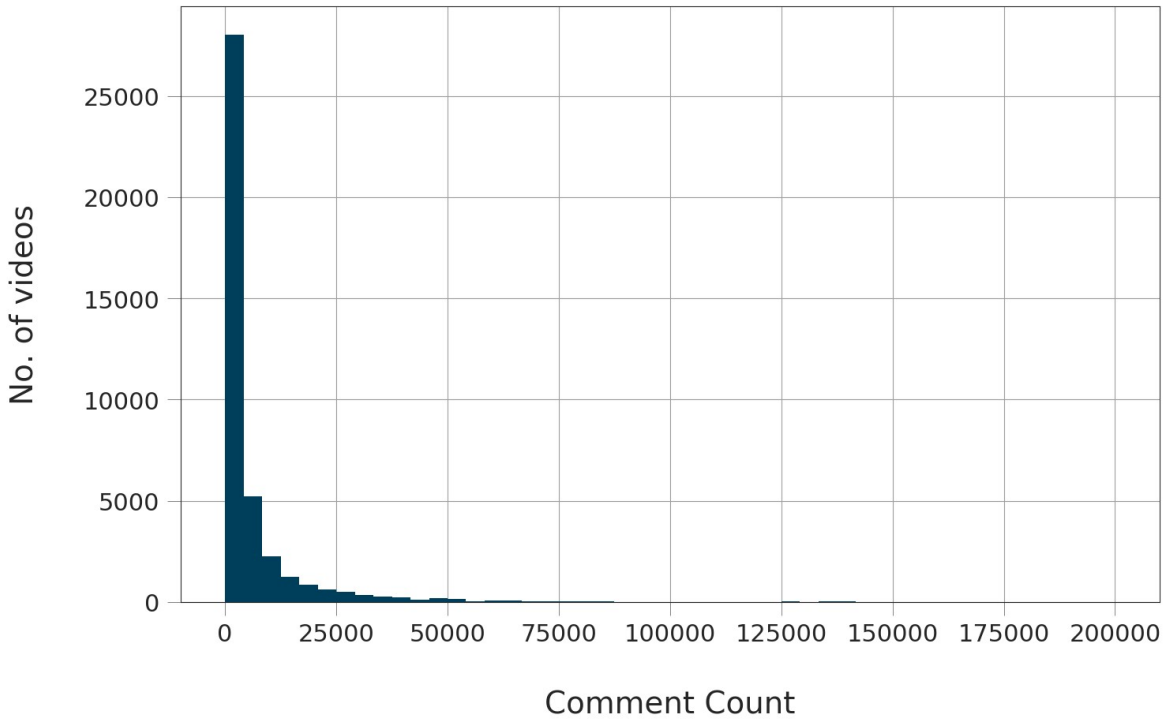

```
df[df['likes'] < 4e4]['likes'].count() / df['likes'].count() * 100
```

```
68.4900730176561
```

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




```
fig, ax = plt.subplots()
_ = sns.distplot(df[df["comment_count"] < 200000]["comment_count"],
kde=False, rug=False,
                 color=PLOT_COLORS[4], hist_kws={'alpha': 1},
                 bins=np.linspace(0, 2e5, 49), ax=ax)
_ = ax.set(xlabel="Comment Count", ylabel="No. of videos")
```




```
df[df['comment_count'] < 4000]['comment_count'].count() /
df['comment_count'].count() * 100
```

```
67.48882756599673
```

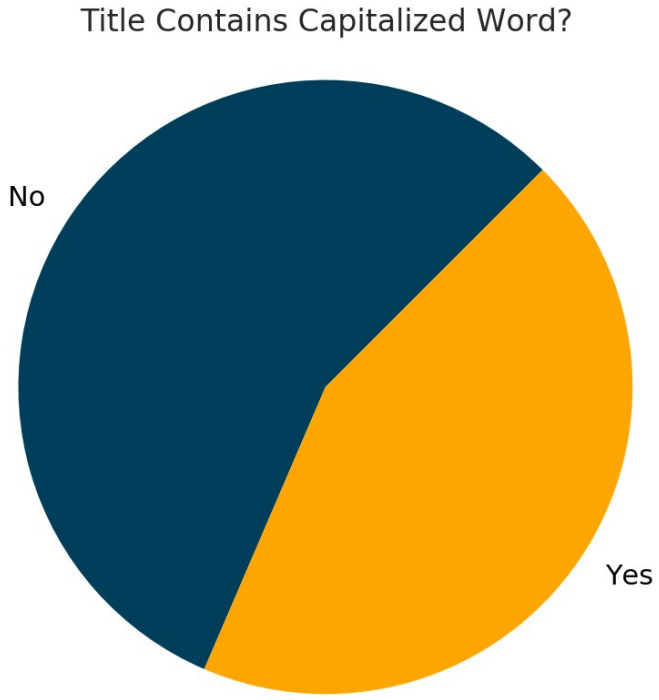
```
df.describe(include = ['O'])
```

```
           video_id     ...     description
count         40949     ...           40949
unique         6351     ...            6902
top     j4KvrAUjn6c     ...
freq             30     ...             570

[4 rows x 8 columns]
```

```
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]
```

```
          video_id                              ...
description
14266  0ufNmUyf2co                              ...
Some of the questions I get most are about my ...
14491  0ufNmUyf2co                              ...
Some of the questions I get most are about my ...
14706  0ufNmUyf2co                              ...
Some of the questions I get most are about my ...
14931  0ufNmUyf2co                              ...
Some of the questions I get most are about my ...
15175  0ufNmUyf2co                              ...
Some of the questions I get most are about my ...
15385  0ufNmUyf2co                              ...
Some of the questions I get most are about my ...

[6 rows x 16 columns]
```

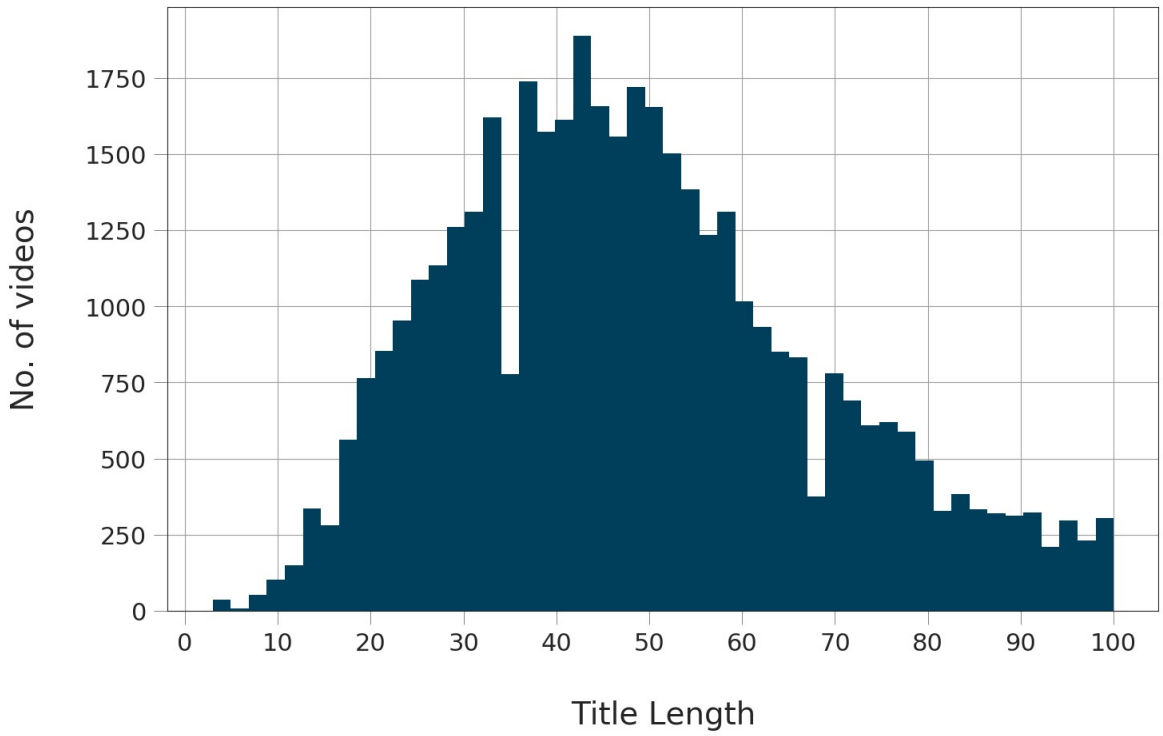
```
def contains_capitalized_word(s):
    for w in s.split():
        if w.isupper():
            return True
    return False


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'}, startangle=45)
_ = ax.axis('equal')
_ = ax.set_title('Title Contains Capitalized Word?')
```



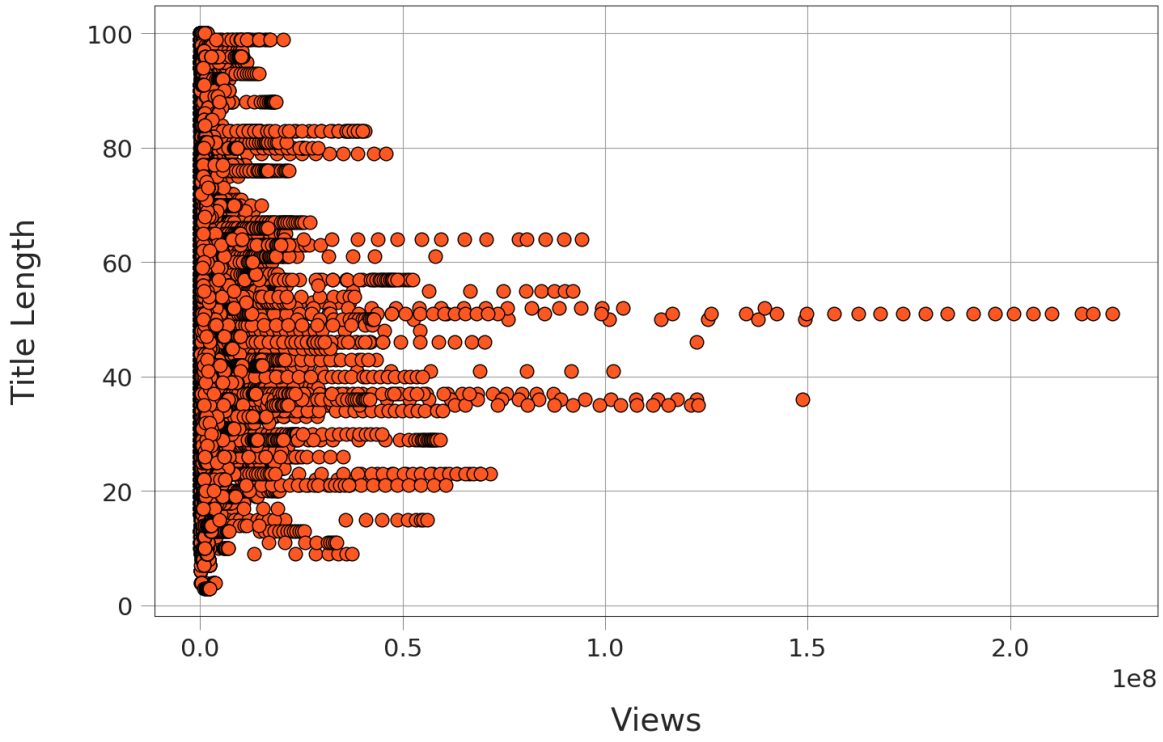

```
df["contains_capitalized"].value_counts(normalize=True)
```

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

```
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))
```




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

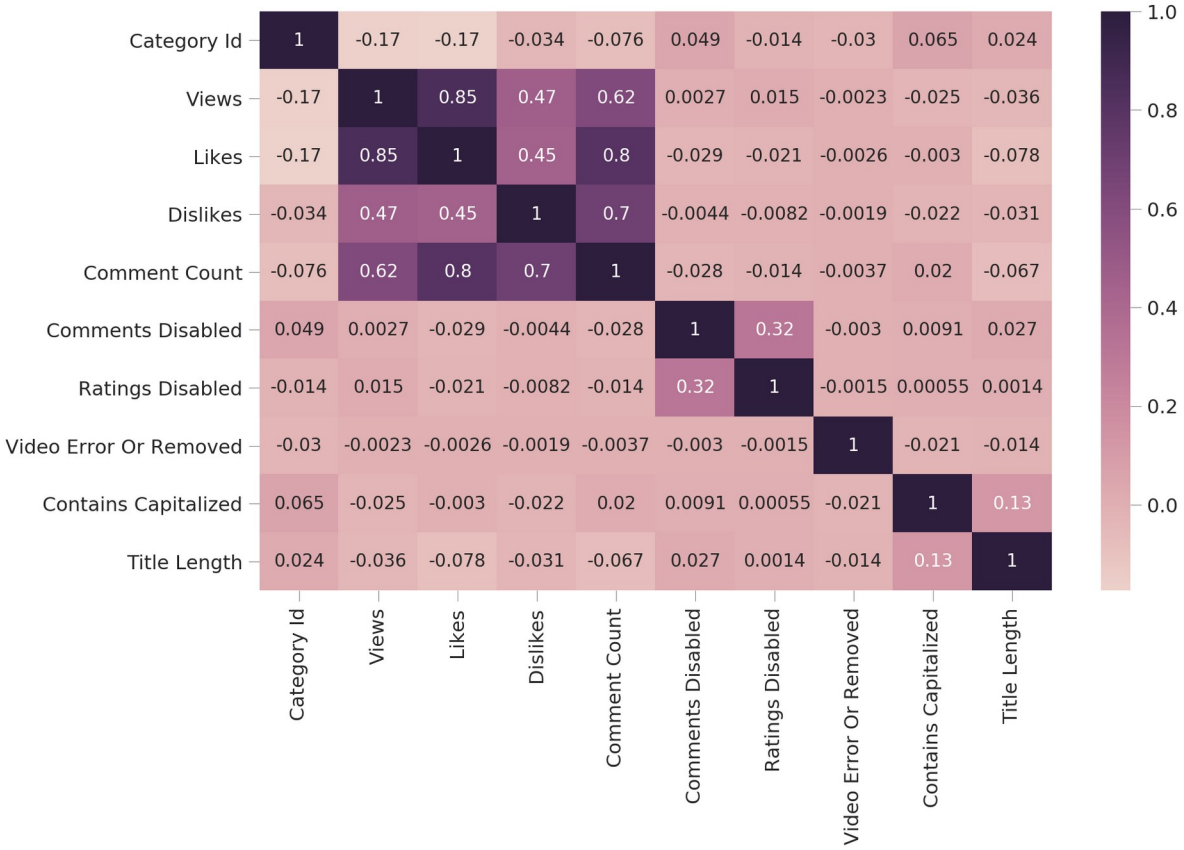



```
df.corr()
```

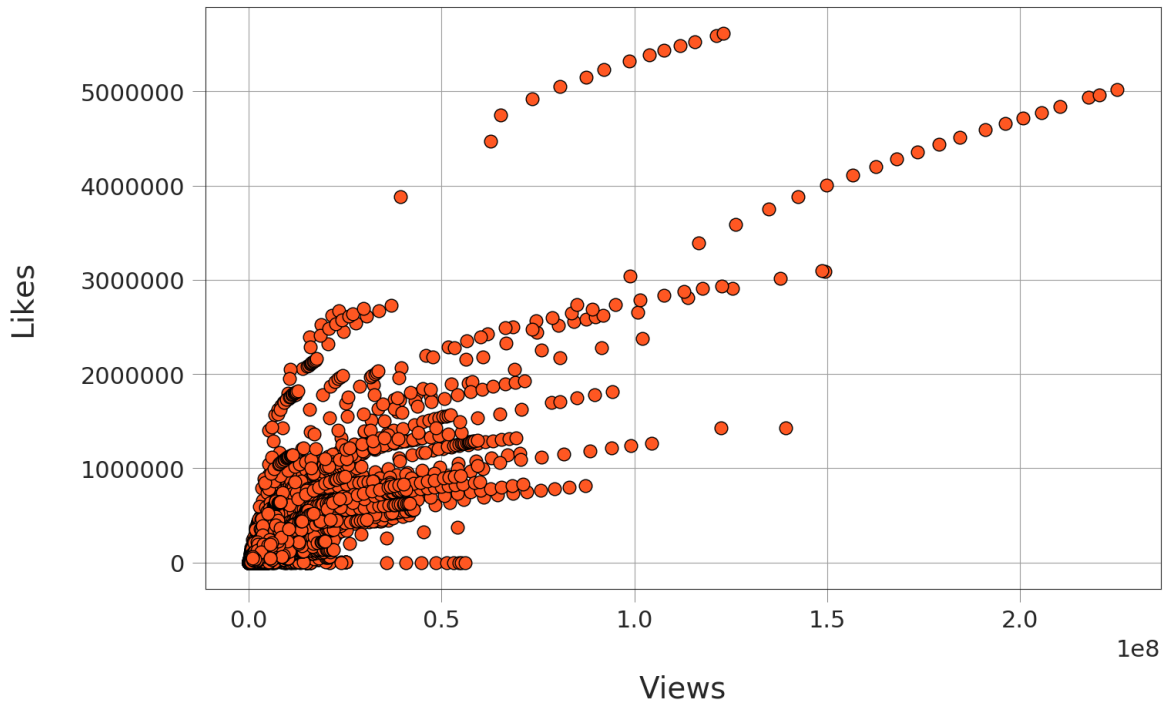
	category_id	...	title_length
category_id	1.00	...	0.02
views	-0.17	...	-0.04
likes	-0.17	...	-0.08
dislikes	-0.03	...	-0.03
comment_count	-0.08	...	-0.07
comments_disabled	0.05	...	0.03
ratings_disabled	-0.01	...	0.00
video_error_or_removed	-0.03	...	-0.01
contains_capitalized	0.06	...	0.13
title_length	0.02	...	1.00

[10 rows x 10 columns]

```
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_labels, cmap=sns.cubehelix_palette(as_cmap=True), ax=ax)
```




```
fig, ax = plt.subplots()
_ = plt.scatter(x=df['views'], y=df['likes'], color=PLOT_COLORS[2],
edgecolors="#000000", linewidths=0.5)
_ = ax.set(xlabel="Views", ylabel="Likes")
```

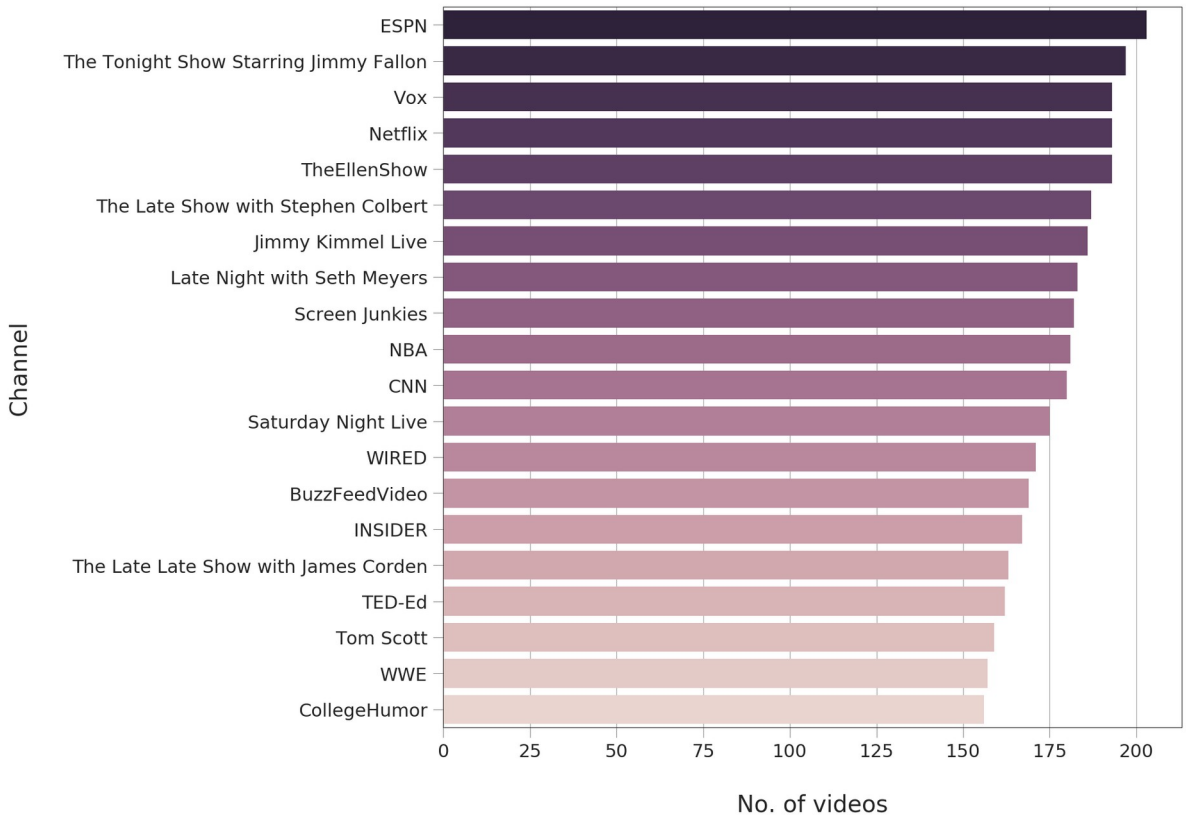



```
title_words = list(df["title"].apply(lambda x: x.split()))
title_words = [x for y in title_words for x in y]
Counter(title_words).most_common(25)
```

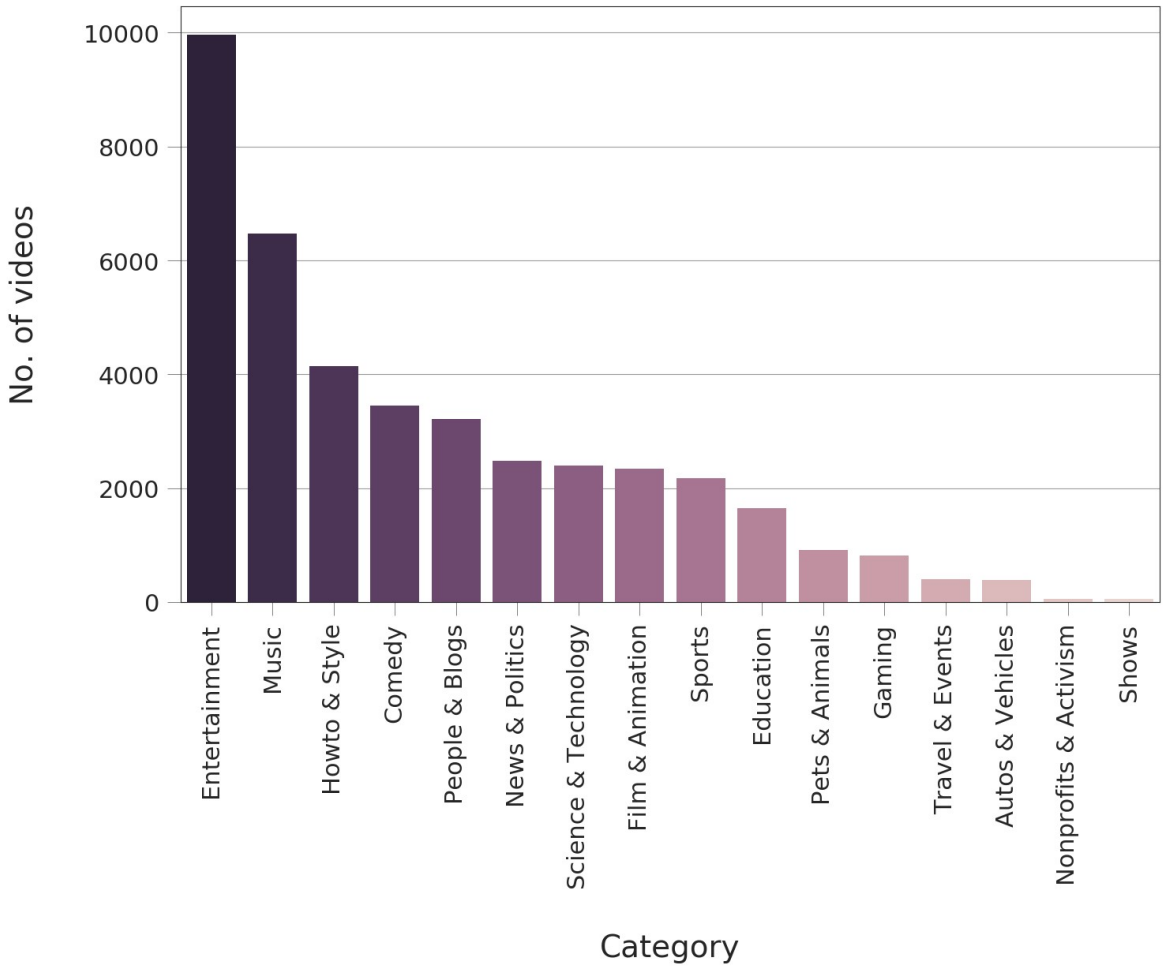
```
[('-', 11452),
 ('|', 10663),
 ('The', 5762),
 ('the', 3610),
 ('a', 2566),
 ('to', 2343),
 ('of', 2338),
 ('in', 2176),
 ('A', 2122),
 ('&', 2024),
 ('I', 1940),
 ('and', 1917),
 ('Video)', 1901),
 ('Trailer', 1868),
 ('How', 1661),
 ('with', 1655),
 ('2018', 1613),
 ('(Official', 1594),
 ('Official',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

```
 ('My', 1080),
 ('for', 1020),
 ('ft.', 1017)]

# wc = wordcloud.WordCloud(width=1200, height=600, collocations=False,
stopwords=None, background_color="white",
colormap="tab20b").generate_from_frequencies(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")
```

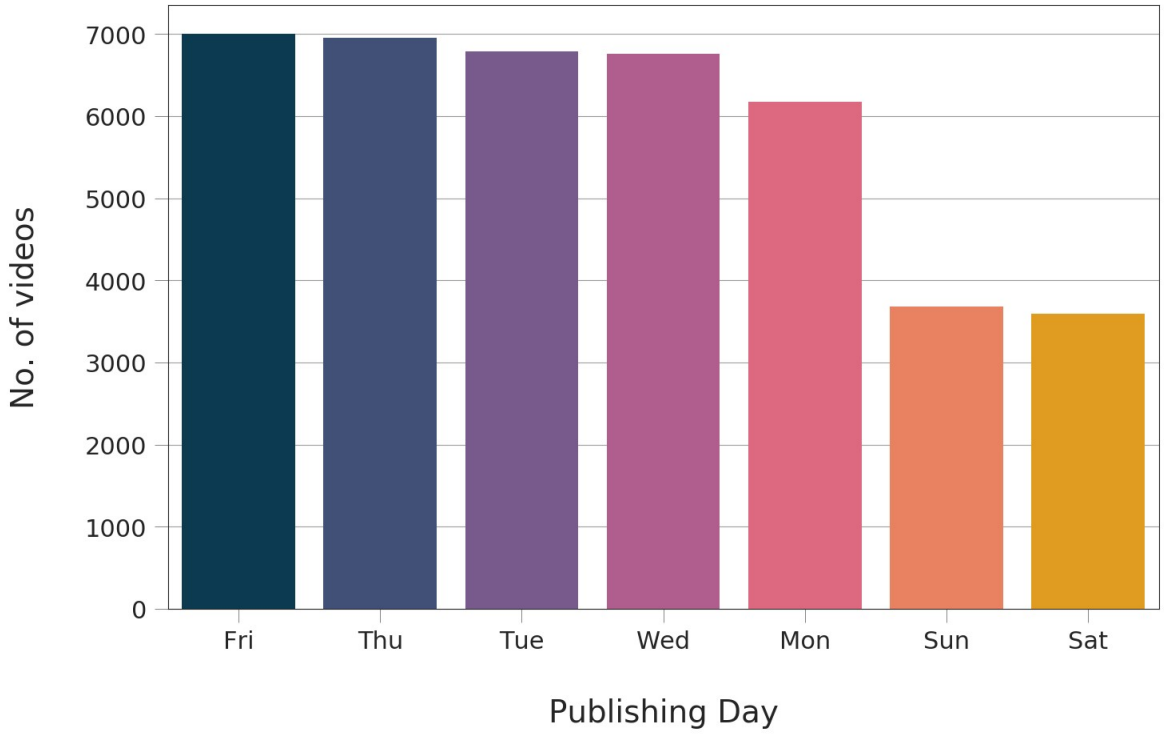

```
cdf =
df.groupby("channel_title").size().reset_index(name="video_count") \
    .sort_values("video_count", ascending=False).head(20)

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=ax)
_ = ax.set(xlabel="No. of videos", ylabel="Channel")
```



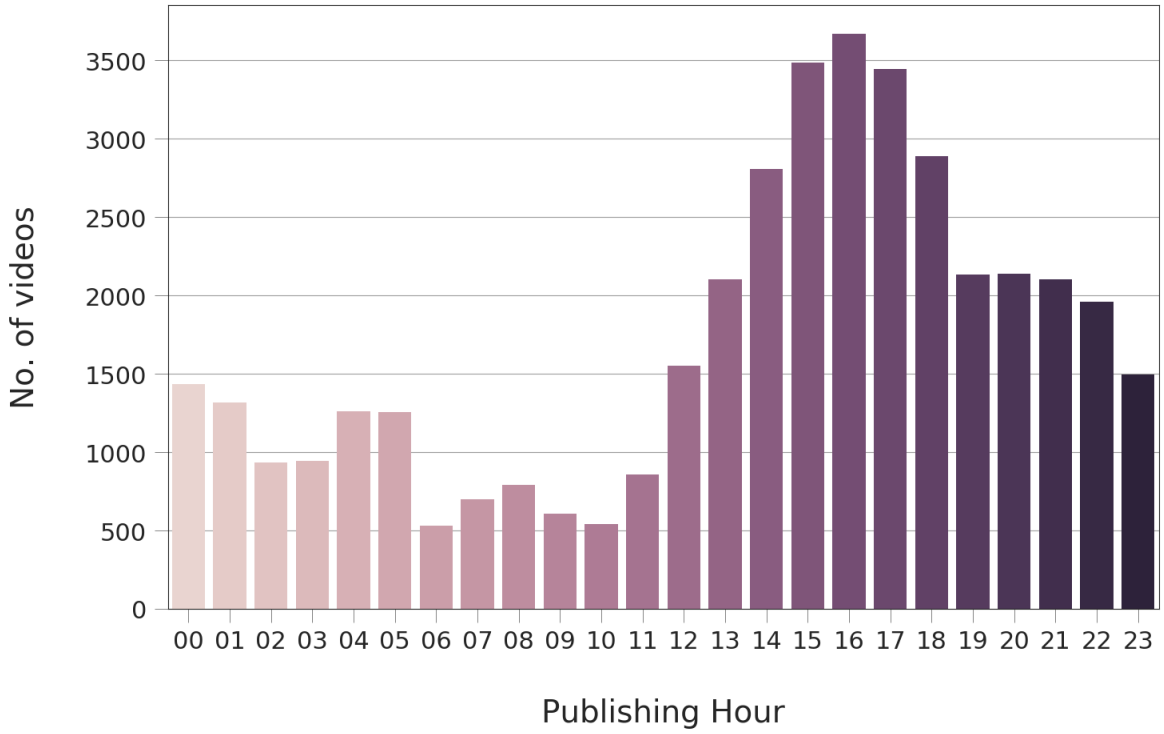

```
with open("../input/US_category_id.json") as f:
    categories = json.load(f)["items"]
cat_dict = {}
for cat in categories:
    cat_dict[int(cat["id"])] = cat["snippet"]["title"]
df['category_name'] = df['category_id'].map(cat_dict)

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=ax)
_ = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
_ = ax.set(xlabel="Category", ylabel="No. of videos")
```



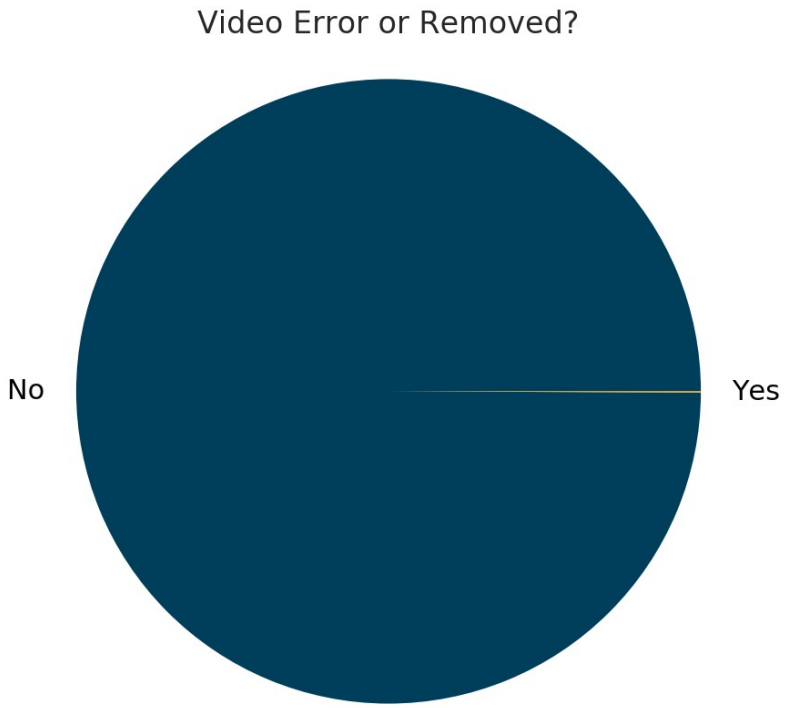

```
df["publishing_day"] = df["publish_time"].apply(
    lambda x: datetime.datetime.strptime(x[:10], "%Y-%m-
%d").date().strftime('%a'))
df["publishing_hour"] = df["publish_time"].apply(lambda x: x[11:13])
df.drop(labels='publish_time', axis=1, inplace=True)

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")
```




```
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")
```



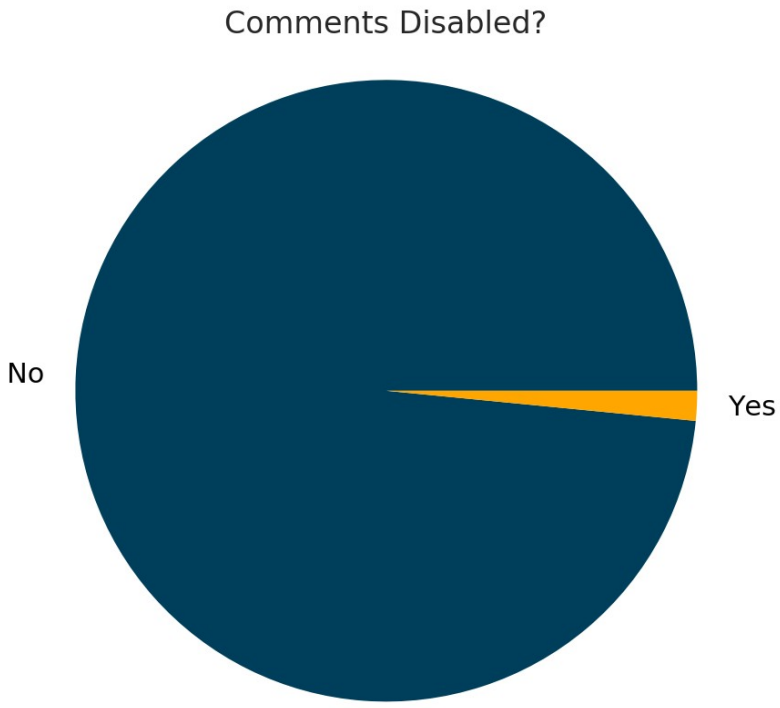

```
value_counts = df["video_error_or_removed"].value_counts().to_dict()
fig, ax = plt.subplots()
_ = ax.pie([value_counts[False], value_counts[True]], labels=['No',
'Yes'],
        colors=['#003f5c', '#ffa600'], textprops={'color': '#040204'})
_ = ax.axis('equal')
_ = ax.set_title('Video Error or Removed?')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




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

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

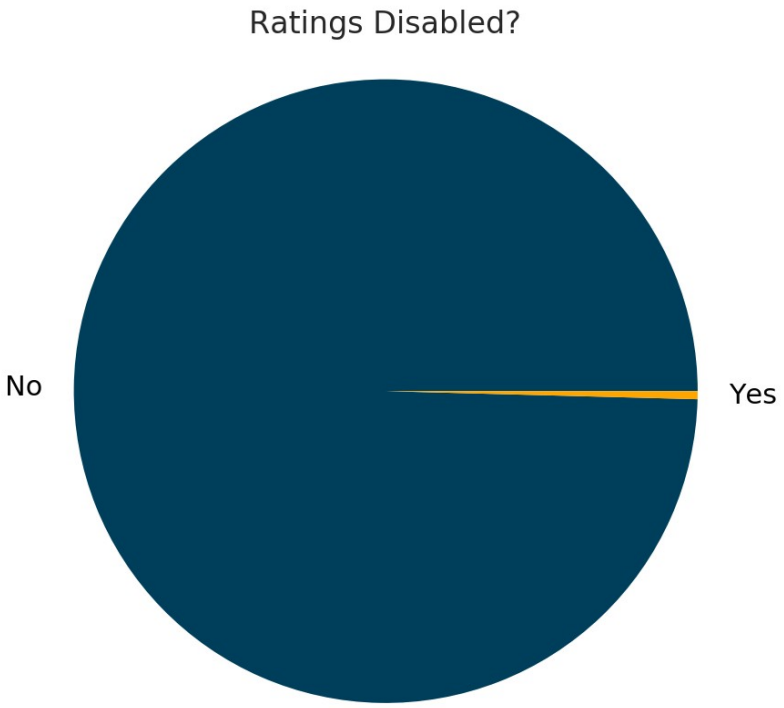
```
value_counts = df["comments_disabled"].value_counts().to_dict()
fig, ax = plt.subplots()
_ = ax.pie(x=[value_counts[False], value_counts[True]], labels=['No',
'Yes'],
           colors=['#003f5c', '#ffa600'], textprops={'color':
'#040204'})
_ = ax.axis('equal')
_ = ax.set_title('Comments Disabled?')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




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

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

```
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?')
```




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

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

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

```
106
```