In [1]:

```
# libraries for data and visualizations
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# library for wordcloud and image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from PIL import Image

# library for json
import json

# library for os
import os

# library for datetime
import datetime as dt

#os.getcwd()
```

In [2]:
```
dat = pd.read_csv("CAvideos.csv")

# Info about the data
dat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40881 entries, 0 to 40880
Data columns (total 16 columns):
video_id                  40881 non-null object
trending_date             40881 non-null object
title                     40881 non-null object
channel_title             40881 non-null object
category_id               40881 non-null int64
publish_time              40881 non-null object
tags                      40881 non-null object
views                     40881 non-null int64
likes                     40881 non-null int64
dislikes                  40881 non-null int64
comment_count             40881 non-null int64
thumbnail_link            40881 non-null object
comments_disabled         40881 non-null bool
ratings_disabled          40881 non-null bool
video_error_or_removed    40881 non-null bool
description               39585 non-null object
dtypes: bool(3), int64(5), object(8)
memory usage: 4.2+ MB
```

There are 16 columns with 40881 rows.

In [59]:
```
# Info about the data
dat.head()
```

Out[59]:

	video_id	trending_date	title	channel_title	category_id	publish_time	tags	vie
0	n1WpP7iowLc	17.14.11	Eminem - Walk On Water (Audio) ft. Beyoncé	EminemVEVO	10	2017-11-10 17:00:03	Eminem|"Walk"|"On"|"Water"|"Aftermath/Shady/In...	171585
1	0dBIkQ4Mz1M	17.14.11	PLUSH - Bad Unboxing Fan Mail	iDubbbzTV	23	2017-11-13 17:00:00	plush|"bad unboxing"|"unboxing"|"fan mail"|"id...	10146
2	5qpjK5DgCt4	17.14.11	Racist Superman | Rudy Mancuso, King Bach & Le...	Rudy Mancuso	23	2017-11-12 19:05:24	racist superman|"rudy"|"mancuso"|"king"|"bach"...	31914
3	d380meD0W0M	17.14.11	I Dare You: GOING BALD!?	nigahiga	24	2017-11-12 18:01:41	ryan|"higa"|"higatv"|"nigahiga"|"i dare you"|"...	20958
4	2Vv-BfVoq4g	17.14.11	Ed Sheeran - Perfect (Official Music Video)	Ed Sheeran	10	2017-11-09 11:04:14	edsheeran|"ed sheeran"|"acoustic"|"live"|"cove...	335236

Preview of the first 5 rows of the dataset

Adding categories column

In [3]:
```
# read json file for category names
with open('CA_category_id.json', 'r') as f:
    data = json.load(f)

# create a dictionary to map 'category_id' as 'category'
dic_cat = {}
for category in data['items']:
    dic_cat[category['id']] = category['snippet']['title']
dic_cat
```

Out[3]:
```
{'1': 'Film & Animation',
 '2': 'Autos & Vehicles',
 '10': 'Music',
 '15': 'Pets & Animals',
 '17': 'Sports',
 '18': 'Short Movies',
 '19': 'Travel & Events',
 '20': 'Gaming',
 '21': 'Videoblogging',
 '22': 'People & Blogs',
 '23': 'Comedy',
 '24': 'Entertainment',
 '25': 'News & Politics',
 '26': 'Howto & Style',
 '27': 'Education',
 '28': 'Science & Technology',
 '30': 'Movies',
 '31': 'Anime/Animation',
 '32': 'Action/Adventure',
```

Since `category_id` column contains a list of integers, we convert them into strings to map the `categories` column.

In [4]:
```
dat['category_id'] = dat['category_id'].astype(str)
dat.insert(4, 'categories', dat['category_id'].map(dic_cat))
dat.head()[['categories', 'category_id']]
```

Out[4]:

	categories	category_id
0	Music	10
1	Comedy	23
2	Comedy	23
3	Entertainment	24
4	Music	10

Preview of the first 5 rows of the `categories` and the `category_id` column.

Top 10 Most Views

In [5]:
```
# top 10 most views
top10_views = dat[['views', 'title', 'likes', 'dislikes', 'comment_count', 'publish_time', 'trending_dat
top10_views = top10_views.drop_duplicates(subset="title", keep='first')
top10_views = top10_views.head(10)
top10_views[['views', 'title']].head(10)
```

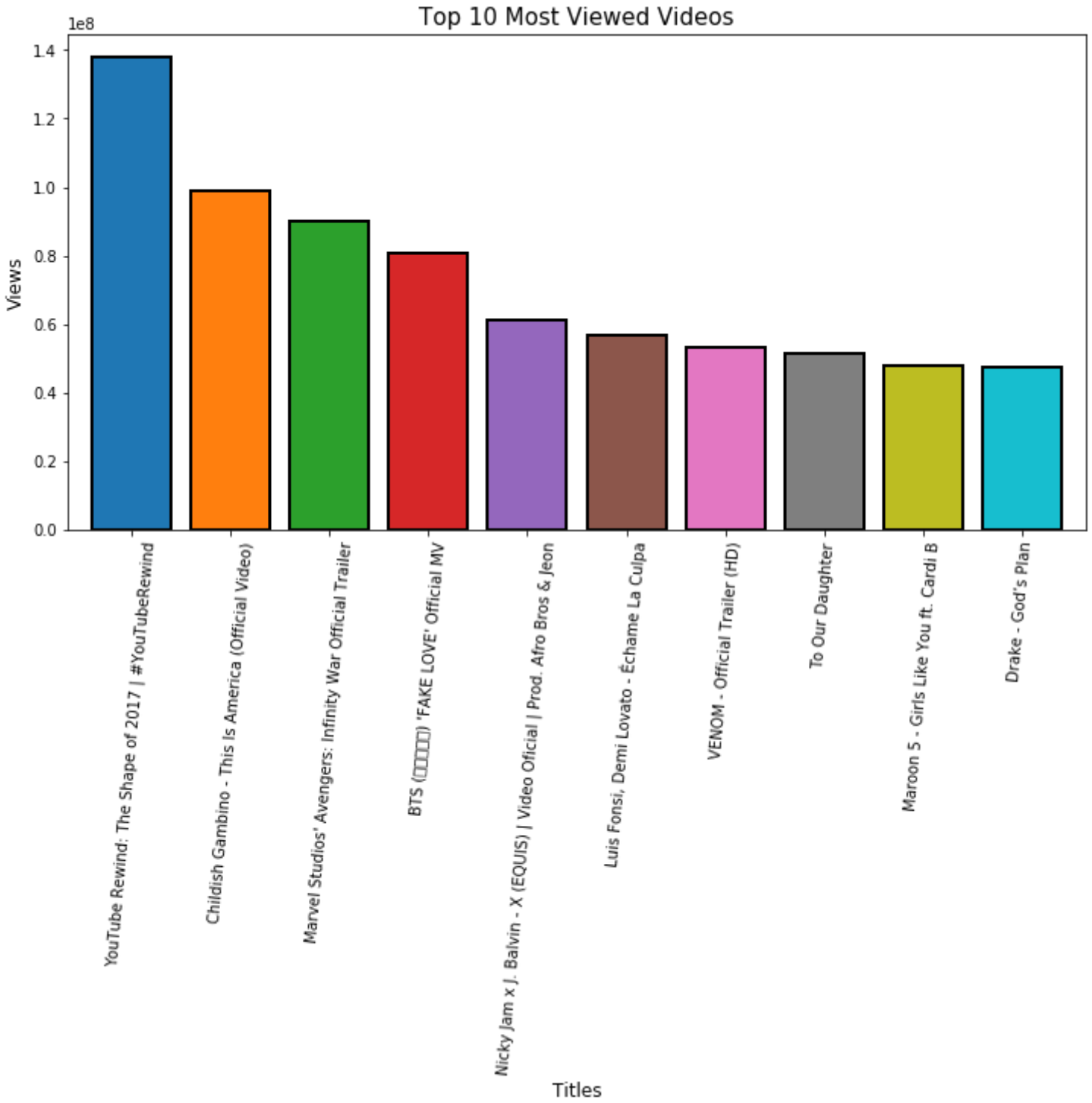
Out[5]:

	views	title
5900	137843120	YouTube Rewind: The Shape of 2017 | #YouTubeRe...
34361	98938809	Childish Gambino - This Is America (Official V...
4699	89930713	Marvel Studios' Avengers: Infinity War Officia...
36453	80738011	BTS (방탄소년단) 'FAKE LOVE' Official MV
22029	61163906	Nicky Jam x J. Balvin - X (EQUIS) | Video Ofic...
1712	56843038	Luis Fonsi, Demi Lovato - Échame La Culpa
31796	53071887	VENOM - Official Trailer (HD)
17237	51243149	To Our Daughter
39208	47778378	Maroon 5 - Girls Like You ft. Cardi B
20055	47362934	Drake - God's Plan

- List of top 10 Youtube videos with the most views.
- We see that 'YouTube Rewind: The Shape of 2017' had the most views.

In [14]:

```
# Label list
label = list(top10_views['title'])

# Plot data
ax = top10_views.plot.bar(x='title', y='views', rot=85, edgecolor='black',
                          linewidth=2, align='center', figsize=(12,6), width=0.8, legend=None)

# Set axis labels / title / legend
ax.set_ylabel("Views", size=12)
ax.set_xlabel("Titles", size=12)
ax.set_title("Top 10 Most Viewed Videos", size=15)

plt.show()
```


Top 10 Most Viewed Videos
1e8
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Views
YouTube Rewind: The Shape of 2017 | #YouTubeRewind
Childish Gambino - This Is America (Official Video)
Marvel Studios' Avengers: Infinity War Official Trailer
BTS (□□□□) 'FAKE LOVE' Official MV
Nicky Jam x J. Balvin - X (EQUIS) | Video Oficial | Prod. Afro Bros & Jeon
Luis Fonsi, Demi Lovato - Échame La Culpa
VENOM - Official Trailer (HD)
To Our Daughter
Maroon 5 - Girls Like You ft. Cardi B
Drake - God's Plan
Titles

- Barplot of the top 10 mostly viewed videos on Youtube.
- We can see that the y-axis is shown in *one hundred millions*, i.e. the most viewed video had 1.4 hundred million views.
- We can also see that there is a relatively steady decreasing trend in the number of views from the left to the right.

Top 10 Most Likes

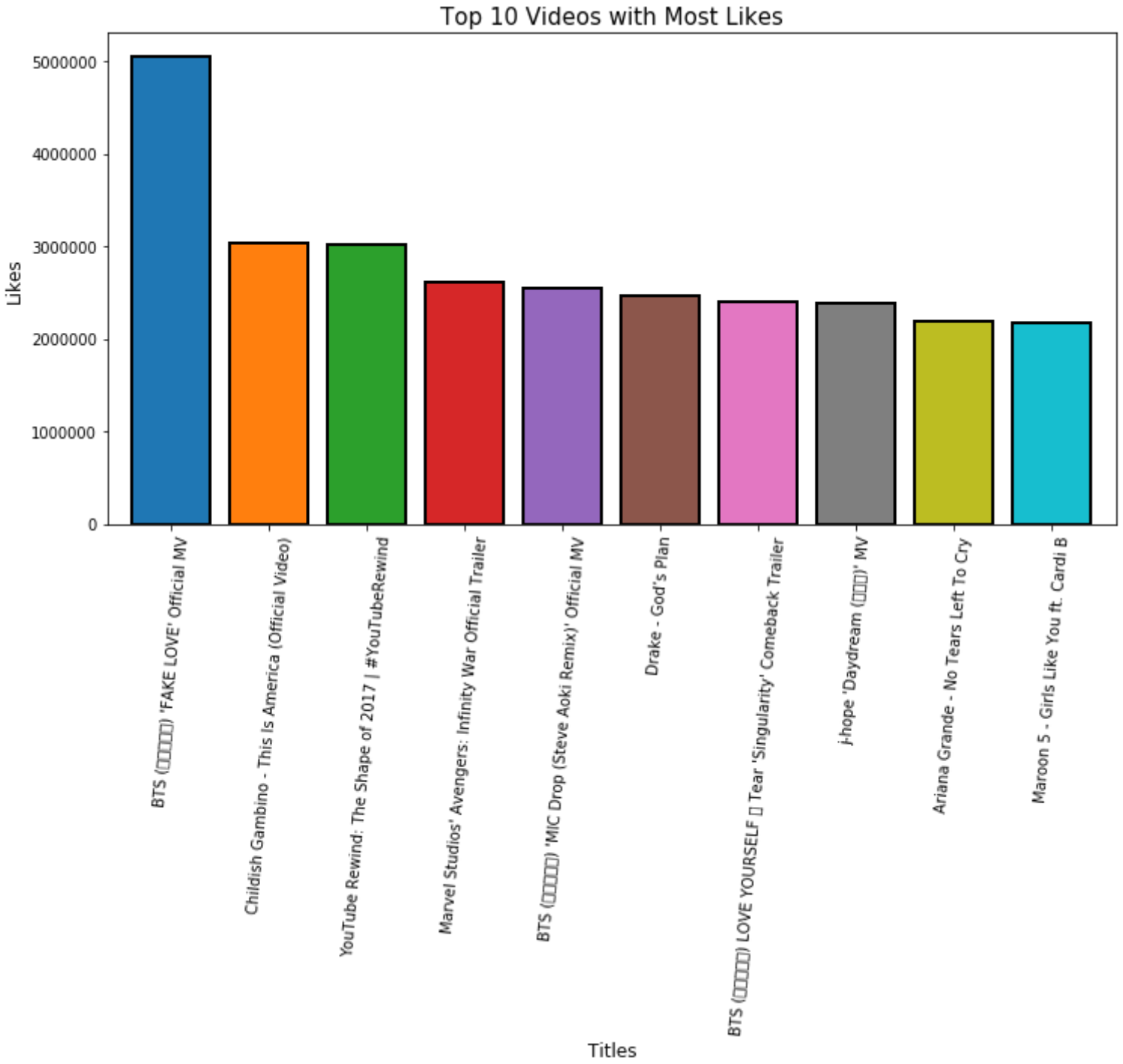
In [16]:
```
# top 10 most likes
top10_likes = dat[['likes', 'title', 'views', 'dislikes', 'comment_count', 'publish_time', 'trending_da
top10_likes = top10_likes.drop_duplicates(subset="title", keep='first')
top10_likes = top10_likes.head(10)
top10_likes[['likes','title']].head(10)
```

Out[16]:

	likes	title
36453	5053338	BTS (방탄소년단) 'FAKE LOVE' Official MV
34361	3037318	Childish Gambino - This Is America (Official V...
5900	3014479	YouTube Rewind: The Shape of 2017 | #YouTubeRe...
4699	2606665	Marvel Studios' Avengers: Infinity War Officia...
2873	2542863	BTS (방탄소년단) 'MIC Drop (Steve Aoki Remix)' Offi...
20055	2469057	Drake - God's Plan
33633	2407419	BTS (방탄소년단) LOVE YOURSELF 轉 Tear 'Singularity'...
21466	2392594	j-hope 'Daydream (백일몽)' MV
30900	2195120	Ariana Grande - No Tears Left To Cry
39208	2178332	Maroon 5 - Girls Like You ft. Cardi B

- List of top 10 Youtube videos with the most likes.
- We see that 'BTS (방탄소년단) 'FAKE LOVE' Official MV' had the most views.

In [17]:
```
# Plot data
ax = top10_likes.plot.bar(x='title', y='likes', rot=85, edgecolor='black',
                          linewidth=2, align='center', figsize=(12,6), width=0.8, legend=None)

# Set axis labels and title
ax.set_ylabel("Likes", size=12)
ax.set_xlabel("Titles", size=12)
ax.set_title("Top 10 Videos with Most Likes", size=15)

plt.show()
```


Top 10 Videos with Most Likes
Likes
5000000
4000000
3000000
2000000
1000000
0
BTS (□□□□) 'FAKE LOVE' Official MV
Childish Gambino - This Is America (Official Video)
YouTube Rewind: The Shape of 2017 | #YouTubeRewind
Marvel Studios' Avengers: Infinity War Official Trailer
BTS (□□□□) 'MIC Drop (Steve Aoki Remix)' Official MV
Drake - God's Plan
BTS (□□□□) LOVE YOURSELF □ Tear 'Singularity' Comeback Trailer
j-hope 'Daydream (□□)' MV
Ariana Grande - No Tears Left To Cry
Maroon 5 - Girls Like You ft. Cardi B
Titles

- Barplot of the top 10 videos with the most likes on Youtube.
- Video wth the most likes is by 'BTS'.
- Except the video with the most likes, the rest of the videos tend to have a steady decreasing number of likes.

Top 10 Most Dislikes

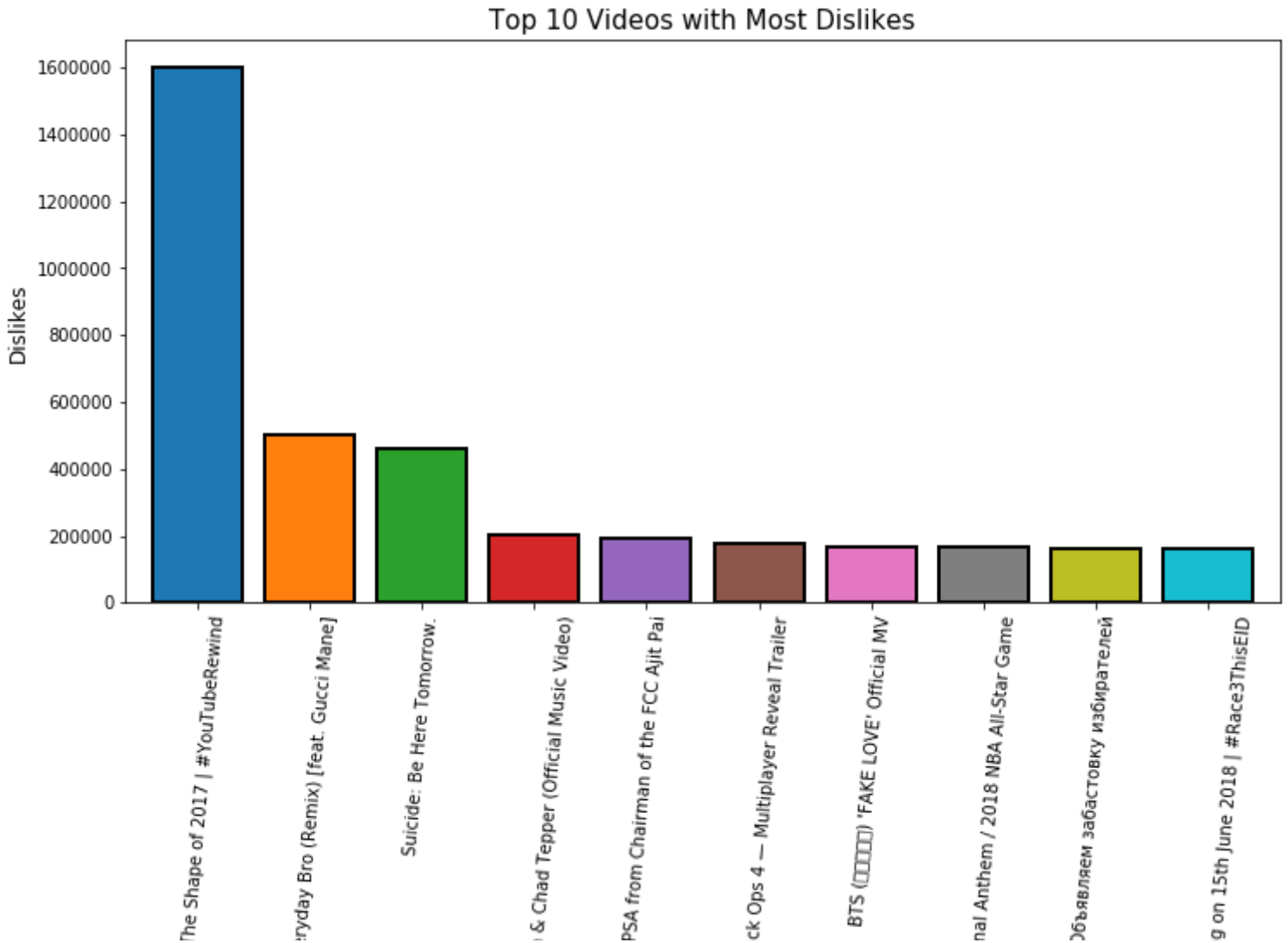
In [18]:
```
# top 10 most dislikes
top10_dlikes = dat[['dislikes', 'title', 'views', 'likes', 'comment_count', 'publish_time', 'trending_da
top10_dlikes = top10_dlikes.drop_duplicates(subset='title', keep='first')
top10_dlikes = top10_dlikes.head(10)
top10_dlikes[['dislikes','title']].head(10)
```

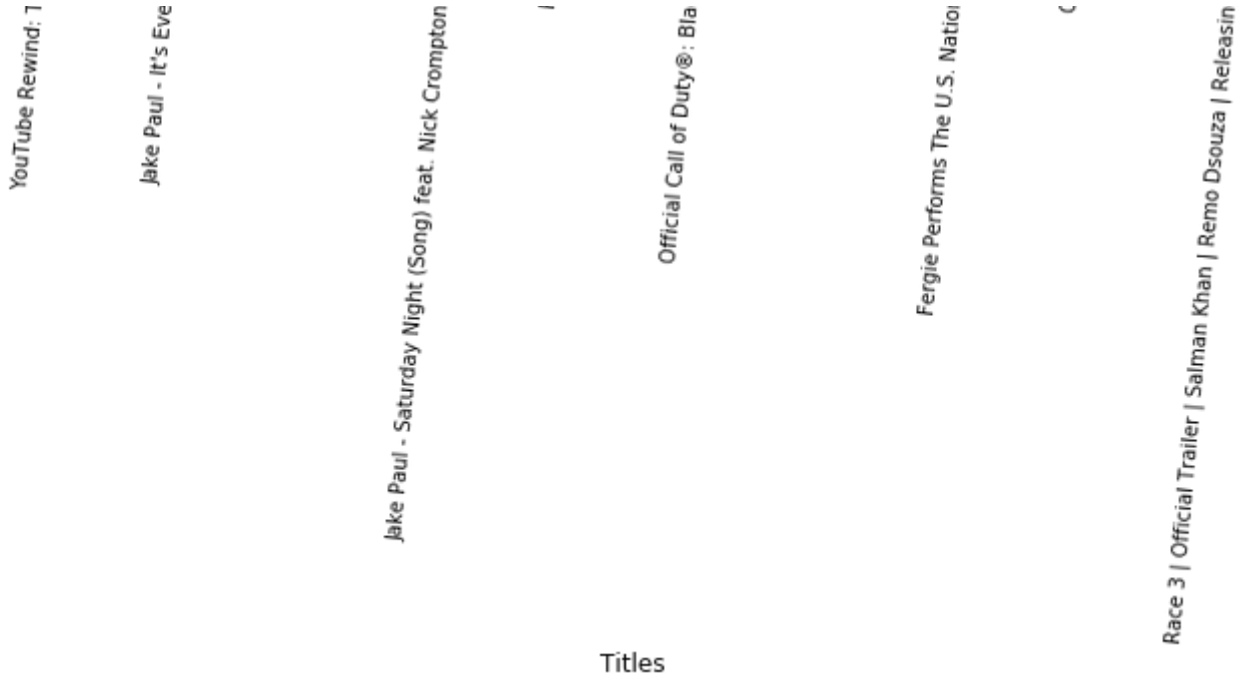
Out[18]:

	dislikes	title
5900	1602383	YouTube Rewind: The Shape of 2017 | #YouTubeRe...
2898	504340	Jake Paul - It's Everyday Bro (Remix) [feat. G...
14852	461660	Suicide: Be Here Tomorrow.
305	200391	Jake Paul - Saturday Night (Song) feat. Nick C...
6686	190227	PSA from Chairman of the FCC Ajit Pai
35970	174645	Official Call of Duty®: Black Ops 4 — Multipla...
36453	165854	BTS (방탄소년단) 'FAKE LOVE' Official MV
20325	164693	Fergie Performs The U.S. National Anthem / 201...
8769	163586	Объявляем забастовку избирателей
35735	162731	Race 3 | Official Trailer | Salman Khan | Remo...

- List of top 10 Youtube videos with the most dislikes.
- We see that 'YouTube Rewind: The Shape of 2017 | #YouTubeRe...' had the most views.

In [20]:
```
# Plot data
ax = top10_dlikes.plot.bar(x='title', y='dislikes', rot=85, edgecolor='black',
                           linewidth=2, align='center', figsize=(12,6), width=0.8, legend=None)

# Set axis labels and title
ax.set_ylabel("Dislikes", size=12)
ax.set_xlabel("Titles", size=12)
ax.set_title("Top 10 Videos with Most Dislikes", size=15)

plt.show()
```





- Barplot of the top 10 videos with the most dislikes on Youtube.
- Video with the most dislikes is also the video with the most views!
- We see that video with the most likes is unusually greater in the number of dislikes than the other 9 videos with the most dislikes.
- We can group the bar graphs into 3:
  - Video with the most dislikes is unusually greater than the other 9 videos.
  - 2nd and 3rd video with the most dislikes with more than 400,000 dislikes.
  - From 4th to the 10th video with the most dislikes being near 200,000 dislikes.

Top 10 Most Comments

In [21]:
```
# top 10 most comments
top10_comments = dat[['comment_count', 'title', 'views','likes', 'dislikes', 'publish_time', 'trending_
top10_comments = top10_comments.drop_duplicates(subset='title', keep='first')
top10_comments = top10_comments.head(10)
top10_comments[['comment_count', 'title']].head(10)
```

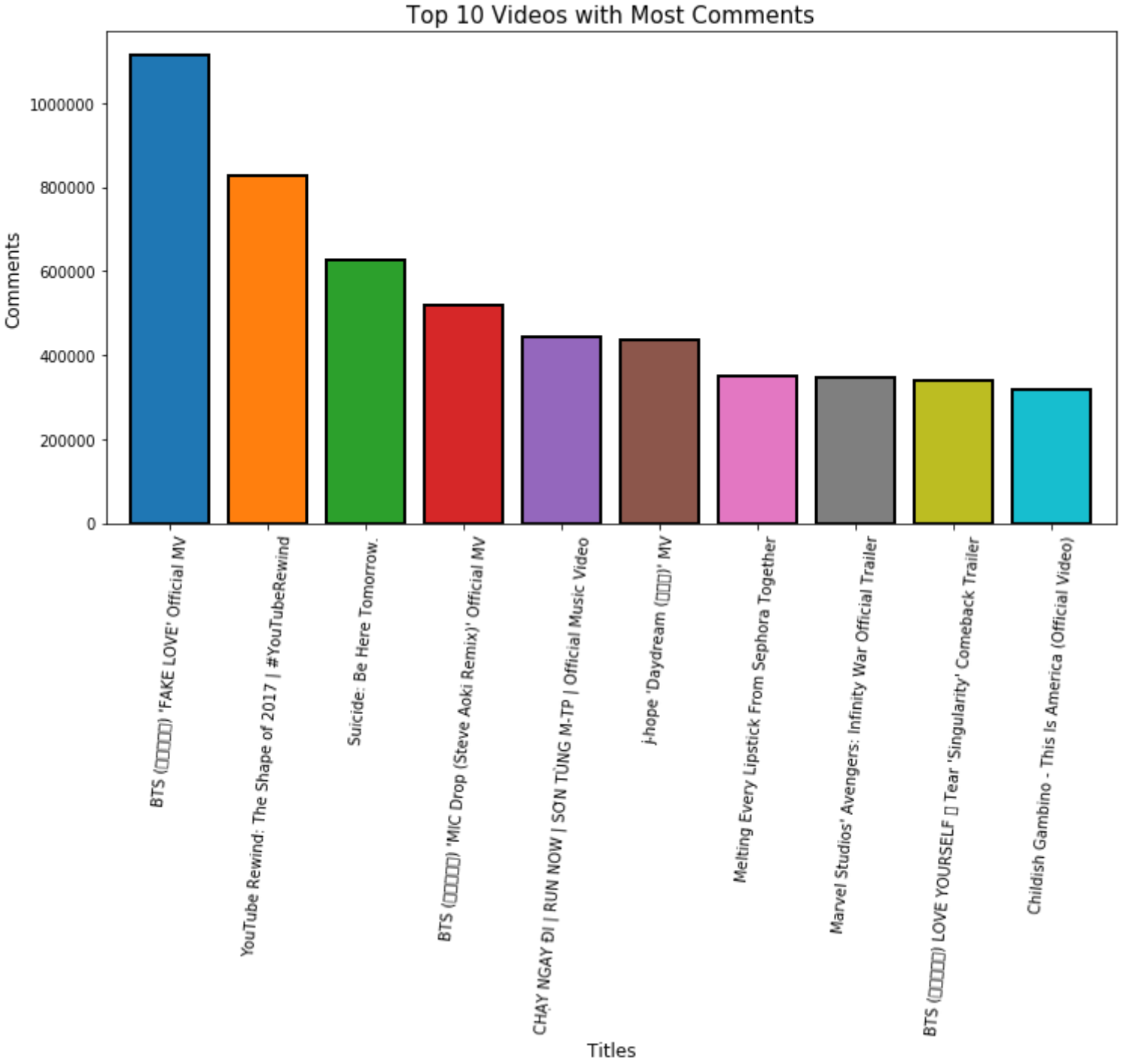
Out[21]:

	comment_count	title
36453	1114800	BTS (방탄소년단) 'FAKE LOVE' Official MV
4996	827755	YouTube Rewind: The Shape of 2017 | #YouTubeRe...
14852	625010	Suicide: Be Here Tomorrow.
2873	519092	BTS (방탄소년단) 'MIC Drop (Steve Aoki Remix)' Offi...
34599	445251	CHẠY NGAY ĐI | RUN NOW | SƠN TÙNG M-TP | Offic...
21466	437036	j-hope 'Daydream (백일몽)' MV
29241	349112	Melting Every Lipstick From Sephora Together
4699	347982	Marvel Studios' Avengers: Infinity War Officia...
33633	340125	BTS (방탄소년단) LOVE YOURSELF 轉 Tear 'Singularity'...
34361	319502	Childish Gambino - This Is America (Official V...

- List of top 10 Youtube videos with the most comments.
- We see that 'BTS (방탄소년단) 'FAKE LOVE' Official MV' had the most comments.
- Note that 'BTS (방탄소년단) 'FAKE LOVE' Official MV' is also the video with the most likes!

In [22]:

```
# Plot data
ax = top10_comments.plot.bar(x='title', y='comment_count', rot=85, edgecolor='black',
                             linewidth=2, align='center', figsize=(12,6), width=0.8, legend=None)

# Set axis labels and title
ax.set_ylabel("Comments", size=12)
ax.set_xlabel("Titles", size=12)
ax.set_title("Top 10 Videos with Most Comments", size=15)

plt.show()
```


Top 10 Videos with Most Comments
Comments
1000000
800000
600000
400000
200000
0
BTS (□□□□□) 'FAKE LOVE' Official MV
YouTube Rewind: The Shape of 2017 | #YouTubeRewind
Suicide: Be Here Tomorrow.
BTS (□□□□□) 'MIC Drop (Steve Aoki Remix)' Official MV
CHẠY NGAY ĐI | RUN NOW | SƠN TÙNG M-TP | Official Music Video
j-hope 'Daydream (□□□)' MV
Melting Every Lipstick From Sephora Together
Marvel Studios' Avengers: Infinity War Official Trailer
BTS (□□□□□) LOVE YOURSELF □ Tear 'Singularity' Comeback Trailer
Childish Gambino - This Is America (Official Video)
Titles

- Barplot of the top 10 vidoes with the most comments on Youtube.
- There is a gradually decreasing trend from the left to the right.
- We also see that the video with the most dislikes had the second most comments, 'YouTube Rewind: The Shape of 2017 | #YouTubeRe...'.

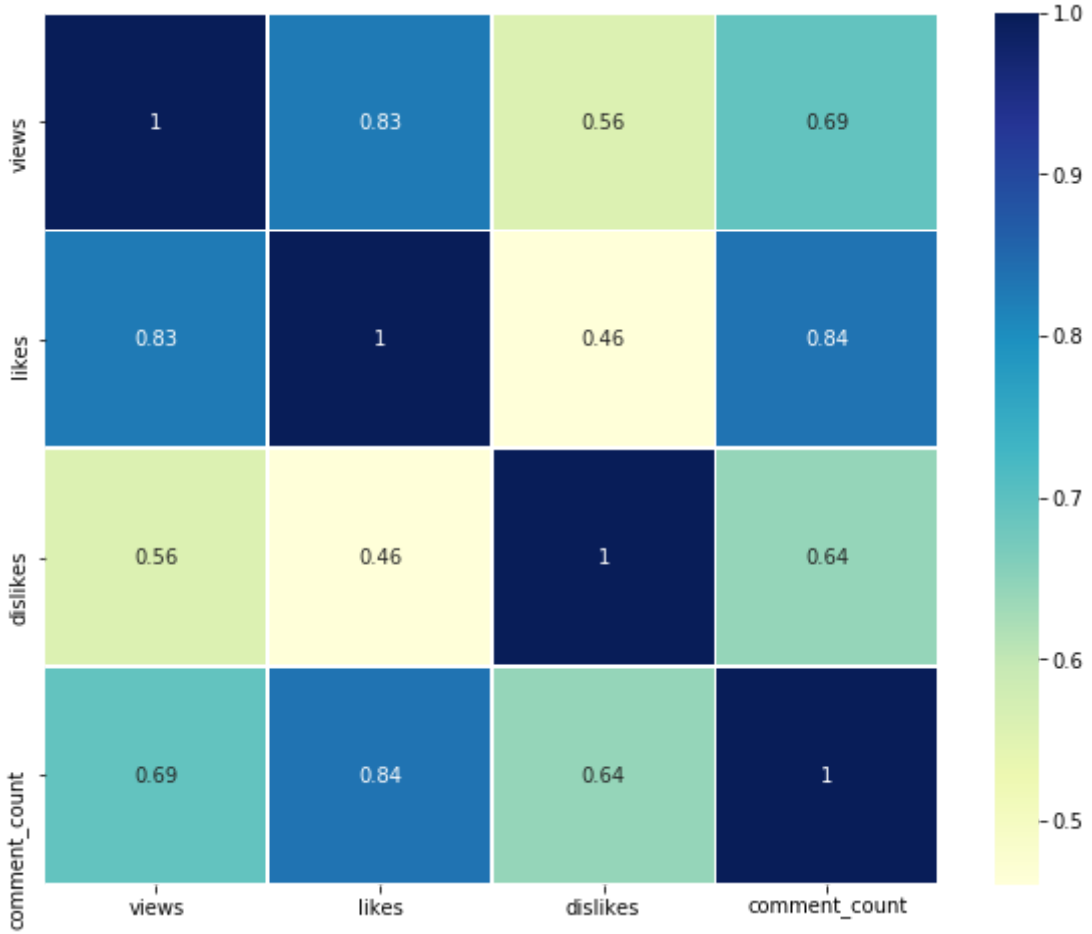
Correlation Matrix and Heatmap

In [75]:
```
# First calculate for the correlation matrix
htmp = dat[['views', 'likes', 'dislikes', 'comment_count']]
htmp = htmp.corr()
htmp
```

Out[75]:

	views	likes	dislikes	comment_count
views	1.000000	0.828964	0.557621	0.693107
likes	0.828964	1.000000	0.460427	0.836585
dislikes	0.557621	0.460427	1.000000	0.643494
comment_count	0.693107	0.836585	0.643494	1.000000

In [76]:
```
# Set figure size
fig, ax = plt.subplots(figsize=(10,8))

# Plot heatmap
sns.heatmap(htmp, annot=True, linewidths=0.5, cmap="YlGnBu")
plt.show()
```



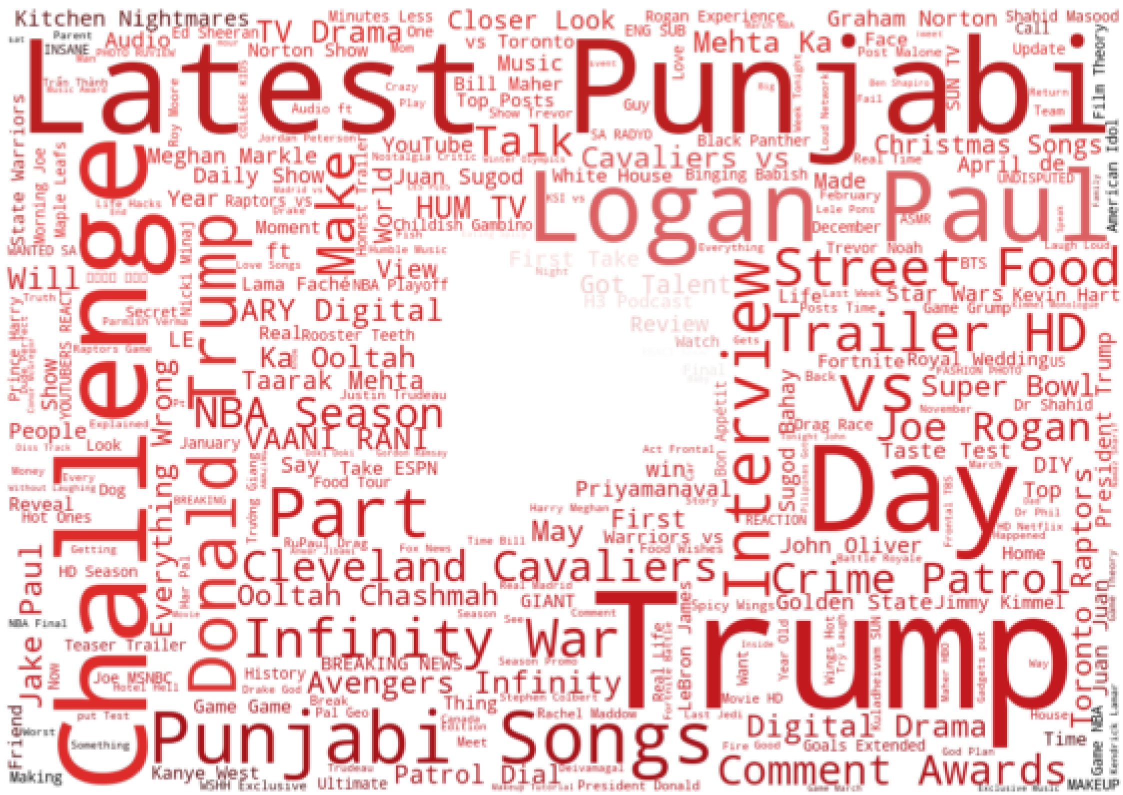
- We see above that there is a **positive correlation** between all the columns ($0 < r < 1$).
  - Note that **positive correlation** means that when **one variable increases, so does the other**, or **when one variable decreases, so does the other**.
- There is a strong positive correlation between `likes` and `views` ($r = 0.83$), and `likes` and `comments` ($r = 0.84$).
- Lowest positive correlation exists between `likes` and `dislikes` ($r = 0.46$).

Word Cloud

Word Cloud of Youtube video titles

In [23]:
```
# combine all the titles of the videos in one string variable
text = " ".join(title for title in dat.title)
print("There are a total of {} words in the titles of the videos.".format(len(text)))

# List of words to be filtered out, i.e. 'video, official'
stopwords = set(STOPWORDS)
stopwords.update(['video', 'official', 'song', 'full', 'episode',
                  'ep', 'highlights', 'new', 'latest punjabi', 'best', 'live'])


# Create wordcloud image and convert mask image into pixels
mask = np.array(Image.open("youtubelogo.png").convert('RGB'))
wc_youtube = WordCloud(stopwords=stopwords, background_color="white",
                       mode="RGB", max_words=300, mask=mask).generate(text)

# Display/plot the image
colors = ImageColorGenerator(mask)
plt.subplots(figsize=(10,8))
plt.imshow(wc_youtube.recolor(color_func=colors), interpolation="bilinear")
plt.axis("off")
plt.show();

# save img to img folder:
# wordcloud.to_file("img/word_cloud.png")
```

```
There are a total of 2236568 words in the titles of the videos.

/anaconda3/lib/python3.7/site-packages/PIL/Image.py:952: UserWarning: Palette images with Transparency
expressed in bytes should be converted to RGBA images
  ' expressed in bytes should be converted ' +
```

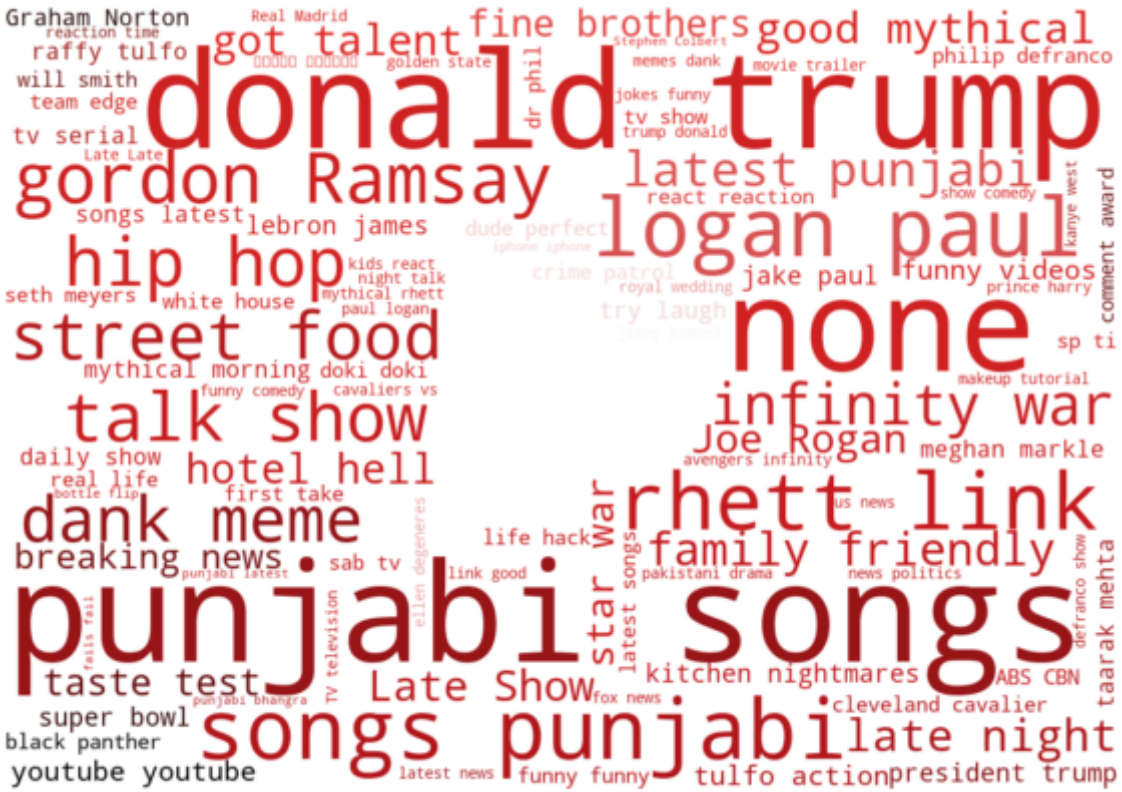


- Word cloud of Youtube video titles in the Youtube logo shape.
- Titles with the biggest presence is the videos with most presence (i.e. number of videos with the same title).
- We see that '*Trump*', '*Latest Punjabi*', '*Challenge*', '*Logan Paul*', '*Day*' are the top 5 most occuring video titles.

Word Cloud for tags

In [78]:
```
# combine all the tags of the videos in one string variable
text = " ".join(title for title in dat.tags)
print("There are a total of {} words in the tags of the videos.".format(len(text)))

# List of words to be filtered out, i.e. 'video, official'
stopwords = set(STOPWORDS)
stopwords.update(['video', 'official', 'song', 'full', 'episode',
                  'ep', 'highlights', 'new', 'latest punjabi', 'best', 'live'])


# Create wordcloud image and convert mask image into pixels
mask = np.array(Image.open("youtubelogo.png").convert('RGB'))
wc_youtube = WordCloud(stopwords=stopwords, background_color="white",
                       mode="RGB", max_words=100, mask=mask).generate(text)

# Display/plot the image
colors = ImageColorGenerator(mask)
plt.subplots(figsize=(10,8))
plt.imshow(wc_youtube.recolor(color_func=colors), interpolation="bilinear")
plt.axis("off")
plt.show()

# save img to img folder:
# wordcloud.to_file("img/word_cloud.png")
```

```
There are a total of 11408278 words in the tags of the videos.

/anaconda3/lib/python3.7/site-packages/PIL/Image.py:952: UserWarning: Palette images with Transparency
expressed in bytes should be converted to RGBA images
  ' expressed in bytes should be converted ' +
```



- Word cloud of Youtube video tags in the Youtube logo shape.
- We see that 'punjabi songs', 'donald trump', 'none', 'songs punjabi', 'logan paul' are the top 5 most occuring tags.

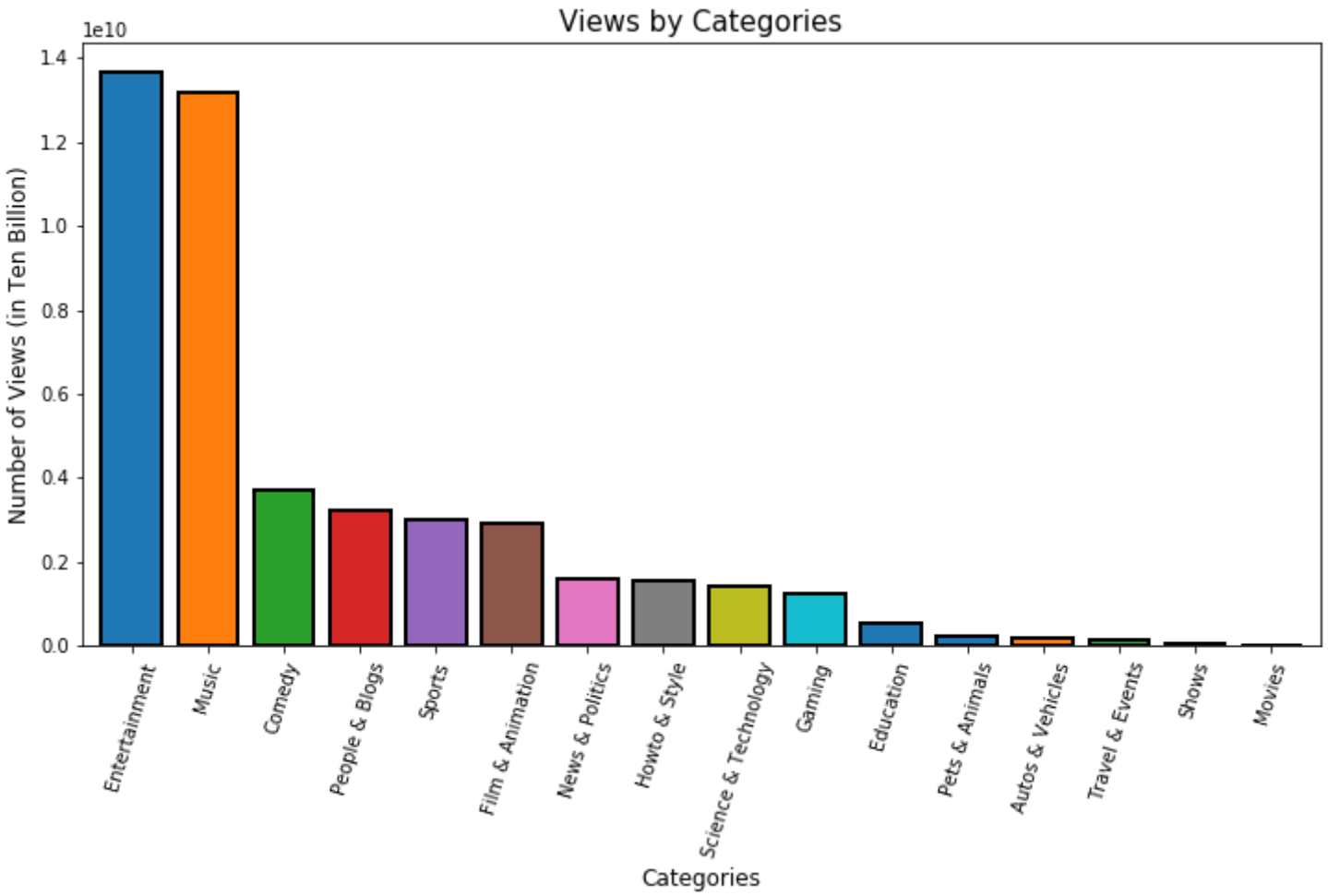
Views by Categories

In [24]:
```
# Group data by 'categories'
views_by_cat = dat.groupby('categories')
cat_list = list(views_by_cat.first().index.values)

# Loop to make a list of views per category
views = []
for category in cat_list:
    views.append(views_by_cat.get_group(category)['views'].sum()) # Add up all the views

# Make a DataFrame and sort 'views' in descending order
data = {"Categories": cat_list, "Views": views}
data = pd.DataFrame(data).sort_values(ascending=False, by='Views')
data.head(5)
```

Out[24]:

	Categories	Views
3	Entertainment	13671215509
8	Music	13179850194
1	Comedy	3708438785
10	People & Blogs	3228227926
14	Sports	2997652188

In [26]:
```
# Plot data
ax = data.plot.bar(x='Categories', y='Views', rot=73, edgecolor='black',
                   linewidth=2, align='center', figsize=(12,6), width=0.8, legend=None)

# Set axis labels and title
ax.set_ylabel("Number of Views (in Ten Billion)", size=12)
ax.set_xlabel("Categories", size=12)
ax.set_title("Views by Categories", size=15)

plt.show()
```



- We see that `Entertainment` and `Music` are the two categories with the most views (Both greater than *thirteen billion* views).

- We see that `Entertainment` and `Music` are unusually greater than other categories, i.e. `Comedy`, `People & Blogs`, and etc...
- `Movies`, `Shows`, `Travel & Events` are the categories with the lowest number of views.

Likes by Categories

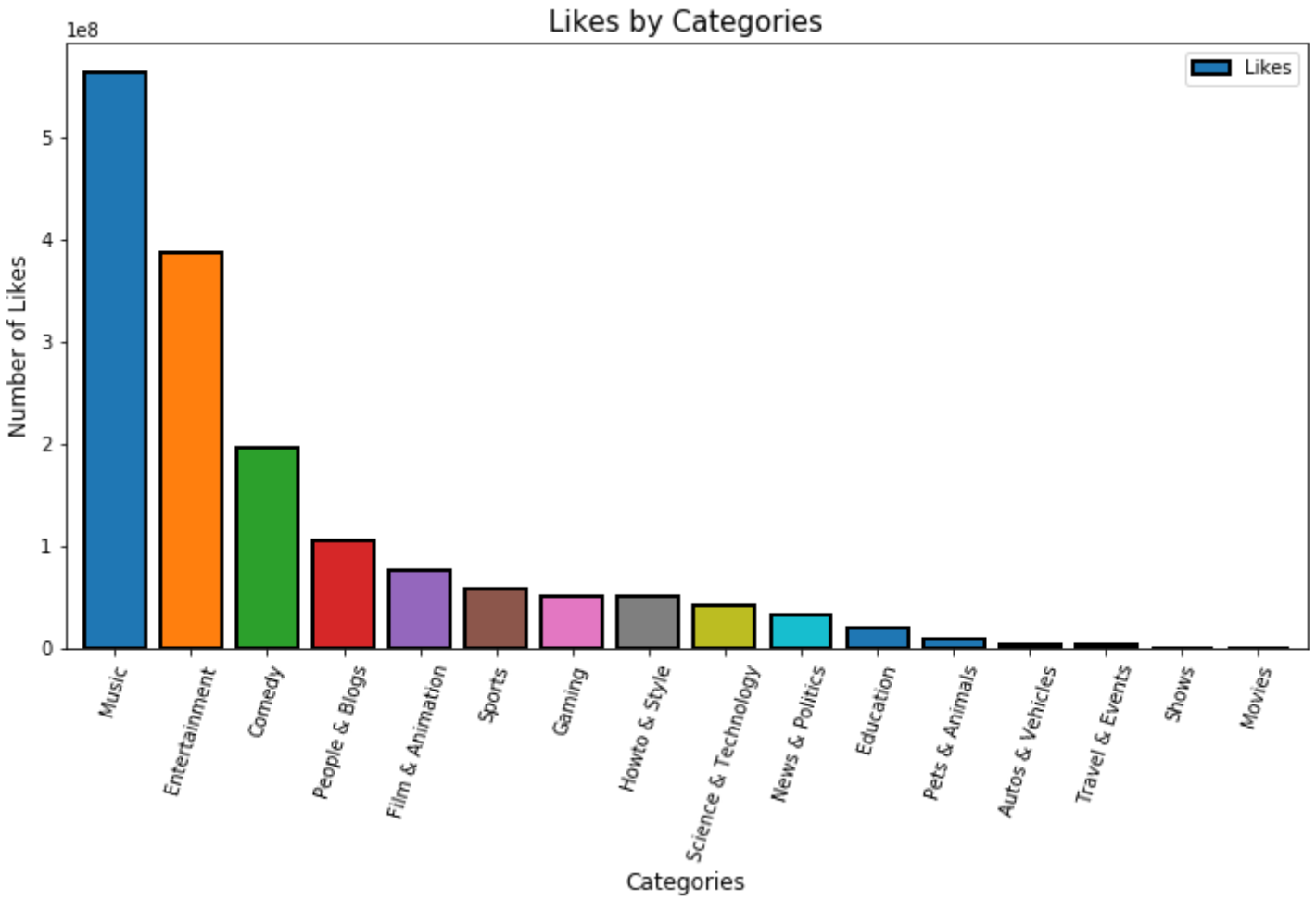
In [732]:

```
# Loop to make a list of likes per category
likes = []
for category in cat_list:
    likes.append(views_by_cat.get_group(category)['likes'].sum()) # Add up all the likes

# Make a DataFrame and sort 'views' in descending order
data = {"Categories": cat_list, "Likes": likes}
data = pd.DataFrame(data).sort_values(ascending=False, by='Likes')
data.head(5)
```

Out[732]:

	Categories	Likes
8	Music	564447530
3	Entertainment	387245433
1	Comedy	196046674
10	People & Blogs	105388564
4	Film & Animation	77802003

In [733]:
```
# Plot data
ax = data.plot.bar(x='Categories', y='Likes', rot=73, edgecolor='black',
                   linewidth=2, align='center', figsize=(12,6), width=0.8)

# Set axis labels and title
ax.set_ylabel("Number of Likes", size=12)
ax.set_xlabel("Categories", size=12)
ax.set_title("Likes by Categories", size=15)

plt.show()
```



- We see again that `Entertainment` and `Music` are the categories with the most likes.

- However, note that this time, `Music` outnumbered `Entertainment` in terms of number of likes.
- Again, we see that `Movies` and `Shows` has the lowest likes.
  - This result is parallel to the correlation coefficient above in the heatmap above since views and likes had relatively high value ($r = 0.83$).

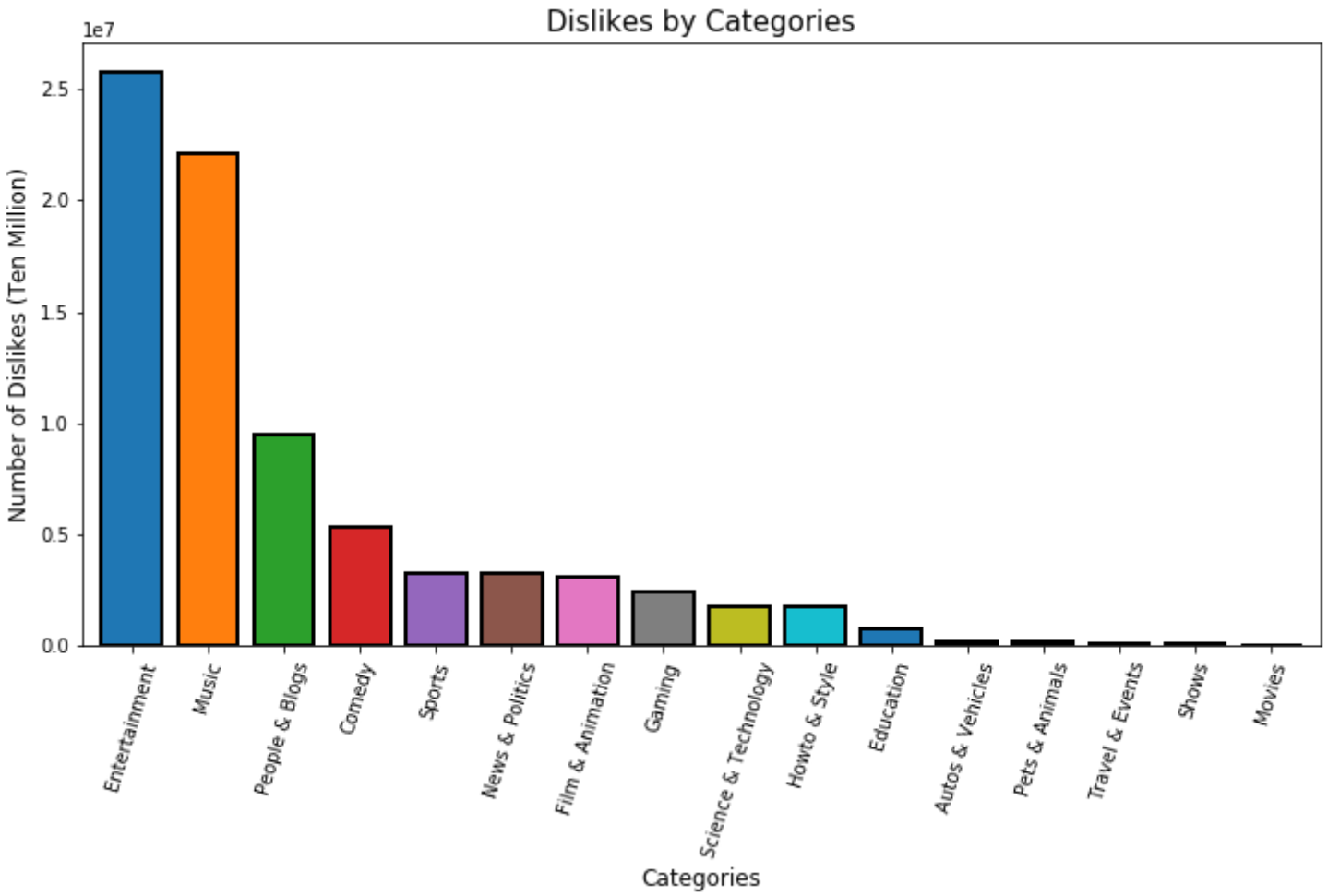
Dislikes by Categories

In [27]:
```
# Loop to make a list of dislikes per category
dislikes = []
for category in cat_list:
    dislikes.append(views_by_cat.get_group(category)['dislikes'].sum()) # Add up all the dislikes

# Make a DataFrame and sort 'views' in descending order
data = {"Categories": cat_list, "Dislikes": dislikes}
data = pd.DataFrame(data).sort_values(ascending=False, by='Dislikes')
data.head(5)
```

Out[27]:

	Categories	Dislikes
3	Entertainment	25791583
8	Music	22098190
10	People & Blogs	9480796
1	Comedy	5372515
14	Sports	3286369

In [29]:
```
# Plot data
ax = data.plot.bar(x='Categories', y='Dislikes', rot=73, edgecolor='black',
                   linewidth=2, align='center', figsize=(12,6), width=0.8, legend=None)

# Set axis labels and title
ax.set_ylabel("Number of Dislikes (Ten Million)", size=12)
ax.set_xlabel("Categories", size=12)
ax.set_title("Dislikes by Categories", size=15)

plt.show()
```



- We see again that `Entertainment` and `Music` had the most dislikes.

- `Movies` and `Shows` had the least number of dislikes.
  - Again, we can relate this result with the heatmap above, as there was a positive correlation between `views` and `dislikes` ($r = 0.56$), we see that the order in which each of the categories are aligned on the graphs are similar to that of the `views` and the `dislikes` .

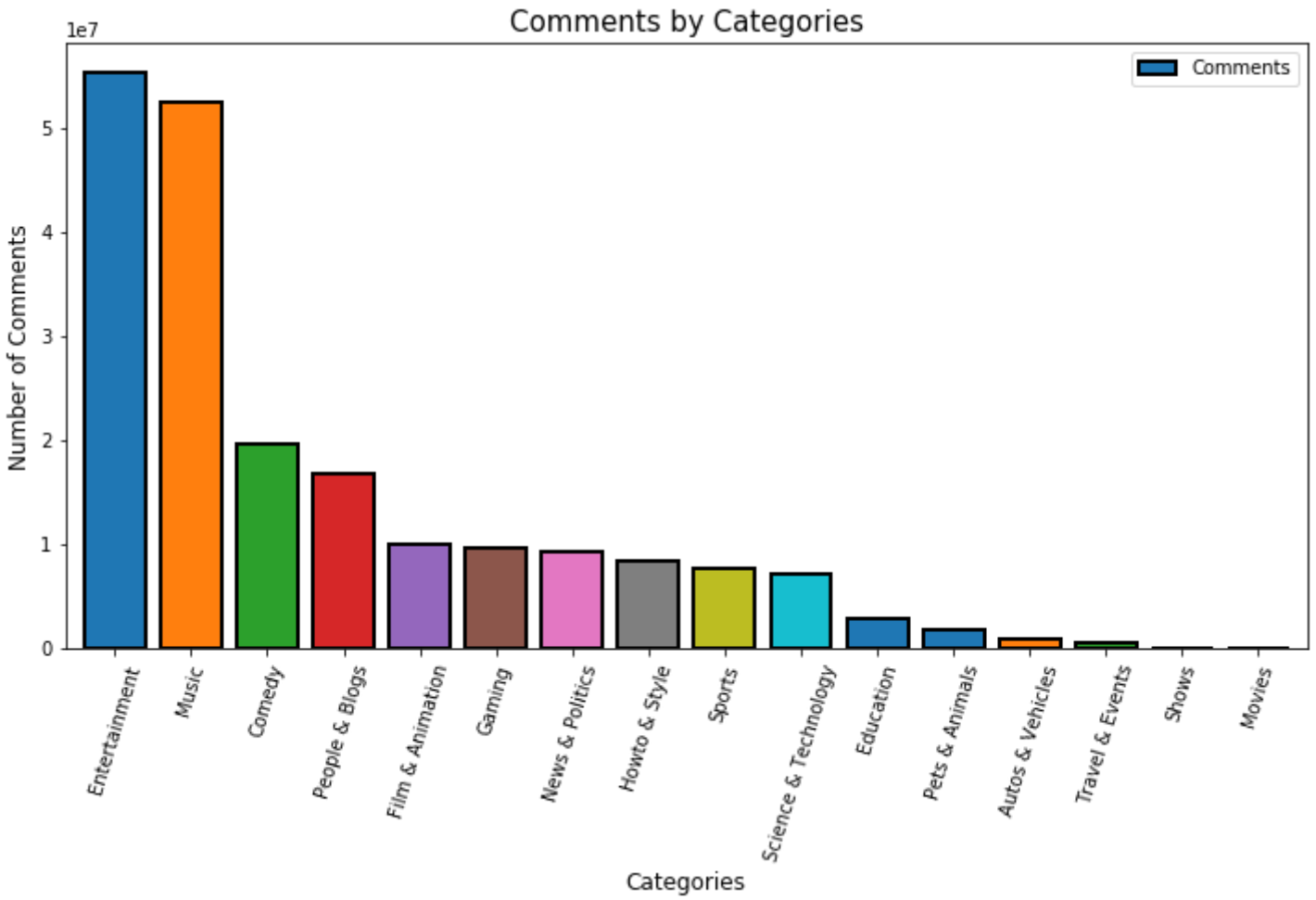
Comments by Categories

In [736]:
```
# Loop to make a list of likes per category
comments = []
for category in cat_list:
    comments.append(views_by_cat.get_group(category)['comment_count'].sum()) # Add up all the comments

# Make a DataFrame and sort 'views' in descending order
data = {"Categories": cat_list, "Comments": comments}
data = pd.DataFrame(data).sort_values(ascending=False, by='Comments')
data.head(5)
```

Out[736]:

	Categories	Comments
3	Entertainment	55313036
8	Music	52435252
1	Comedy	19638776
10	People & Blogs	16818201
4	Film & Animation	10086573

In [737]:
```
# Plot data
ax = data.plot.bar(x='Categories', y='Comments', rot=73, edgecolor='black',
                   linewidth=2, align='center', figsize=(12,6), width=0.8)

# Set axis labels and title
ax.set_ylabel("Number of Comments", size=12)
ax.set_xlabel("Categories", size=12)
ax.set_title("Comments by Categories", size=15)

plt.show()
```



- We see again that `Entertainment` and `Music` had the most comments.

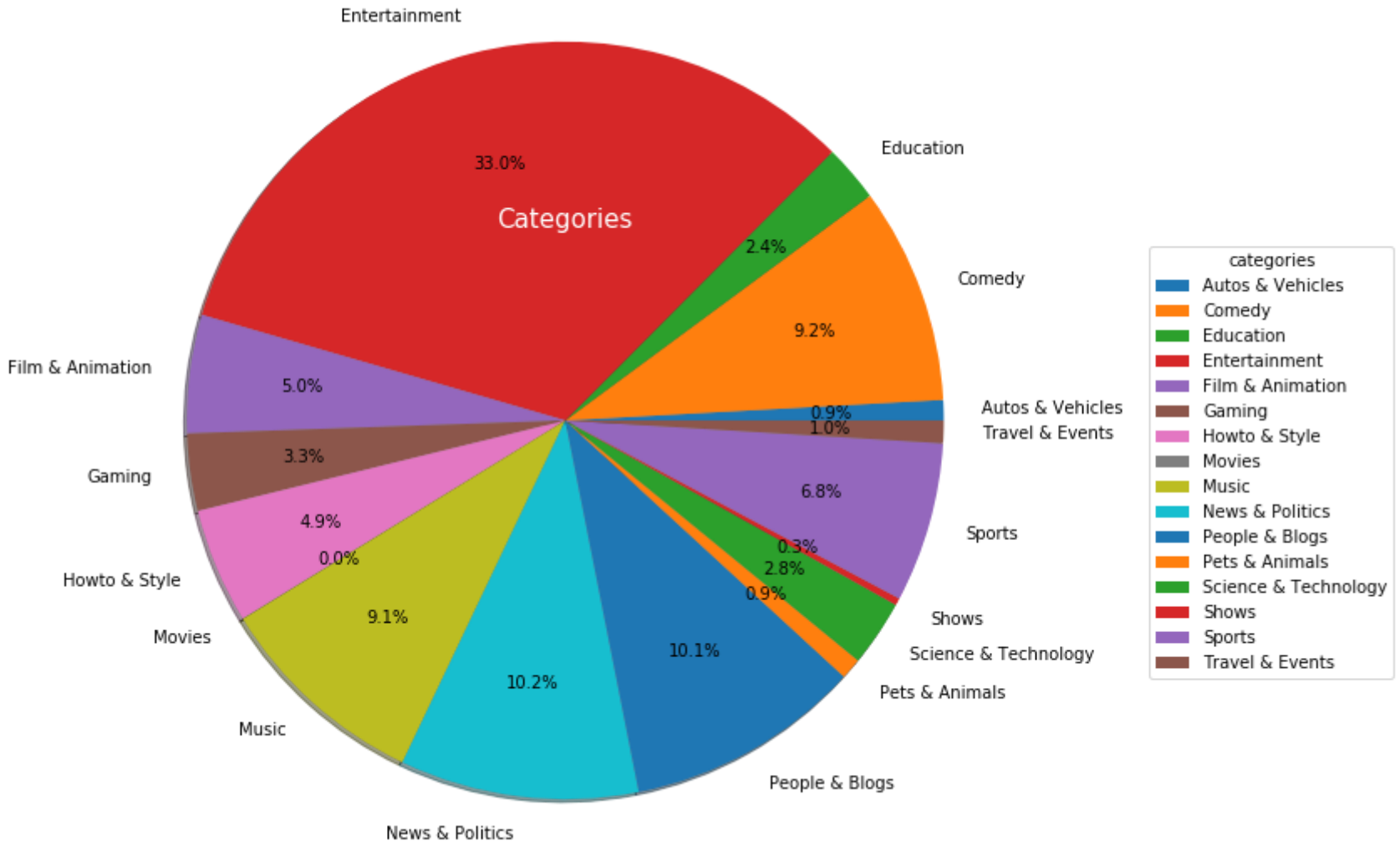
- `Entertainment` and `Music` had the most comments.
- `Shows` and `Movies` had the least comments.

Pie Chart of Categories

In [30]:
```
# Group data by 'categories'
cat_count = dat.groupby('categories')
cat_count = cat_count['categories'].count()
cat_count
```

Out[30]:
```
categories
Autos & Vehicles          353
Comedy                   3773
Education                 991
Entertainment           13451
Film & Animation         2060
Gaming                   1344
Howto & Style            2007
Movies                      6
Music                    3731
News & Politics          4159
People & Blogs           4105
Pets & Animals            369
Science & Technology     1155
Shows                     124
Sports                   2787
Travel & Events           392
Name: categories, dtype: int64
```

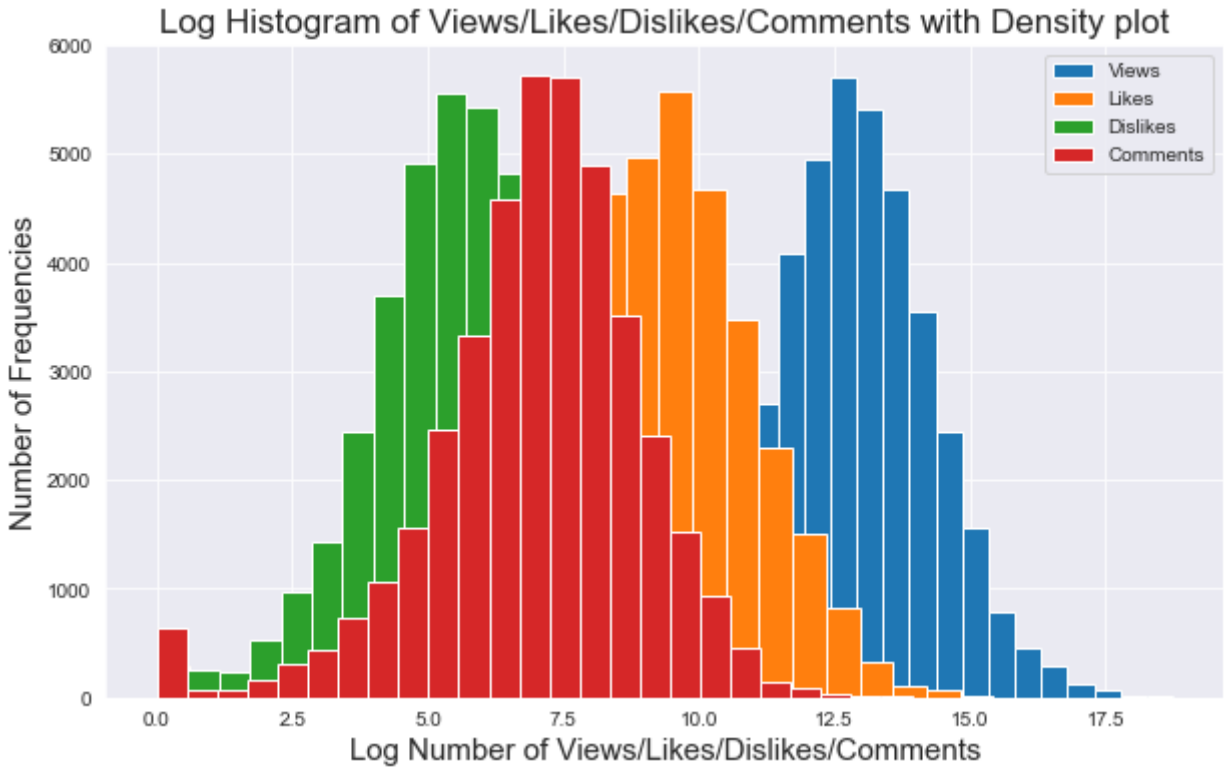
In [32]:
```
# Get lists of category lists and counts
# cat_list = list(cat_count.index.values)
cat_count = list(cat_count)

# Plot Pie Plot
plt.pie(cat_count, labels=cat_list, autopct='%.1f%%', pctdistance=0.7, shadow=True, radius=2.6)
plt.title("Categories", size=15, color='white')
plt.legend(loc='upper right', bbox_to_anchor=(1, 0, 1.8, 1), title="categories")

plt.show()
```



- Proportion of videos in different categories.
- We see `Entertainment` had the biggest proprotion, close to a 3rd of all the videos (33.0%).

Log-Histogram of Views/Likes/Dislikes/Comments

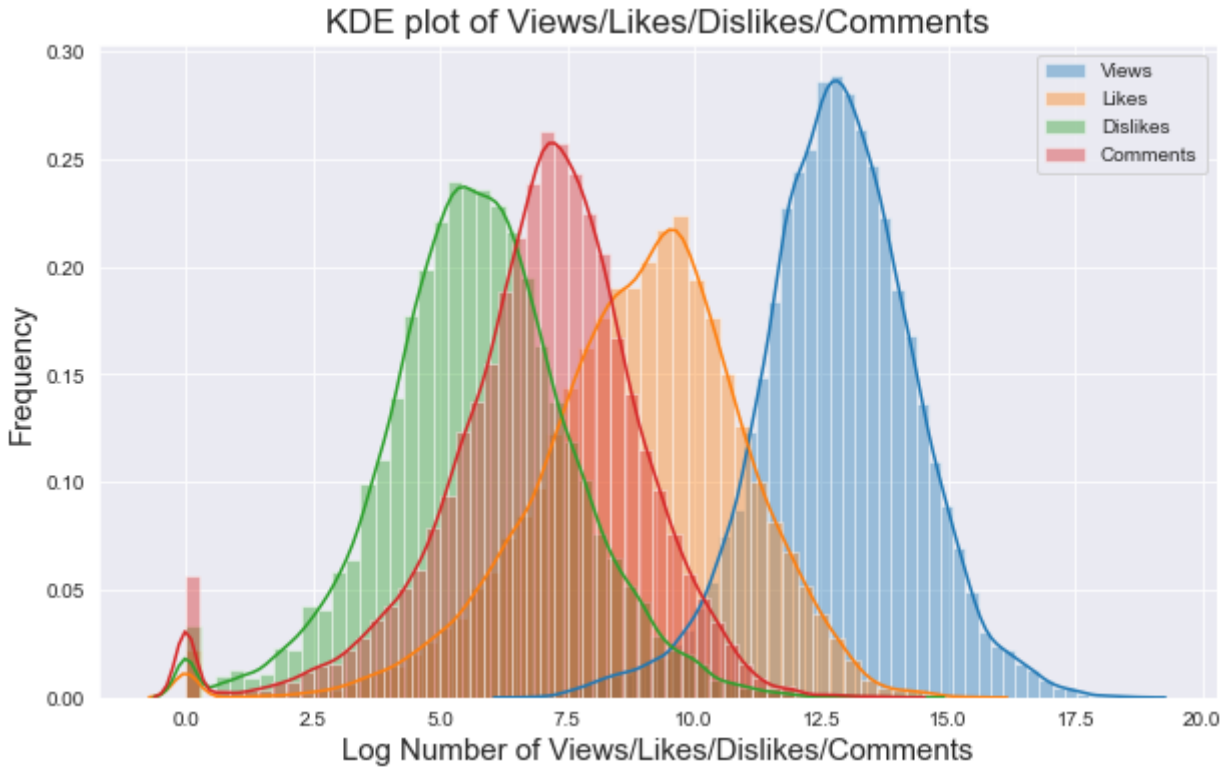
In [331]:
```
# Set figure size
fig, ax = plt.subplots(figsize=(10,6))

# Set theme
sns.set_style('darkgrid')

# Plot all four histograms
plt.hist(np.log(dat['views']+1), bins=25)
plt.hist(np.log(dat['likes']+1), bins=25)
plt.hist(np.log(dat['dislikes']+1), bins=25)
plt.hist(np.log(dat['comment_count']+1), bins=25)

# Grid/Labels/Title/Legend
plt.xlabel('Log Number of Views/Likes/Dislikes/Comments', size=15)
plt.ylabel('Number of Frequencies', size=15)
plt.title('Log Histogram of Views/Likes/Dislikes/Comments with Density plot', size=17)
plt.legend(['Views','Likes','Dislikes','Comments'])

plt.show()
```



- Because the number of `views` greatly outweighs those of `likes/dislikes/comments` , we take the log of the number of `views/likes/dislikes/comments` to compare them together.
- We see that histogram of `views/likes/dislikes/comments` looks normal, i.e.bell-shaped curves.
- Mean value of `views` is the greatest, while the mean values of `dislikes` is the lowest.

KDE Density plot of Views/Likes/Dislikes/Comments

In [332]:
```
# Set figure size and theme
fig, ax = plt.subplots(figsize=(10,6))
sns.set_style('darkgrid')

# Plot all four data
sns.distplot(np.log(dat['views']+1))
sns.distplot(np.log(dat['likes']+1))
sns.distplot(np.log(dat['dislikes']+1))
sns.distplot(np.log(dat['comment_count']+1))

# Add legend/labels/title
plt.legend(['Views','Likes','Dislikes','Comments'])
plt.xlabel('Log Number of Views/Likes/Dislikes/Comments', size=15)
plt.ylabel('Frequency', size=15)
plt.title('KDE plot of Views/Likes/Dislikes/Comments', size=17)

plt.show()
```

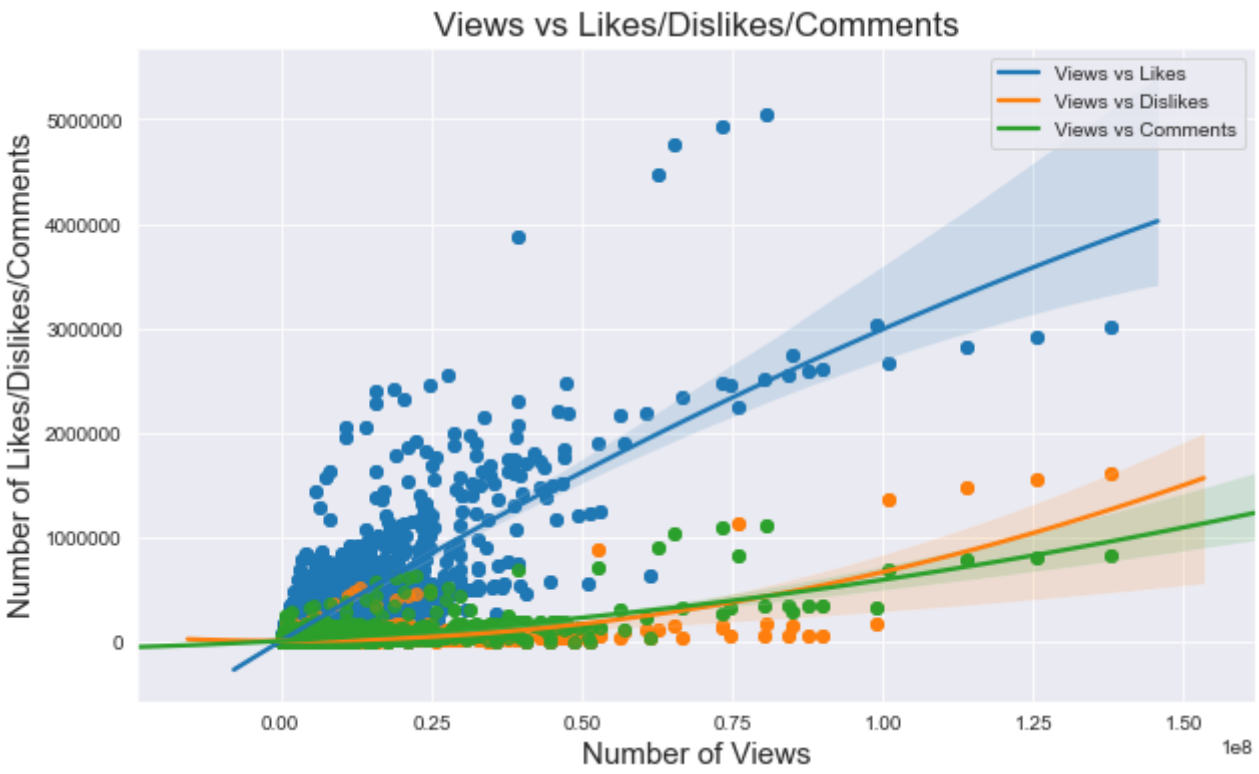


- KDE density plot of `views/likes/dislikes/comments`.
- Based on the curve outside the histogram, we see that the curve is normal, i.e. bell-shaped.

Scatterplot Comparison of Views/Likes/Dislikes/Comments with Regression Lines

In [14]:

```
# Set figure size and theme
fig, ax = plt.subplots(figsize=(10,6))
sns.set_style('darkgrid')

# Plot Scatter Plots
plt.scatter(dat['views'], dat['likes'])
plt.scatter(dat['views'], dat['dislikes'])
plt.scatter(dat['views'], dat['comment_count'])

# Regression plot
sns.regplot(x=dat['views'], y=dat['likes'], fit_reg=True, order=2)
sns.regplot(x=dat['views'], y=dat['dislikes'], fit_reg=True, order=2)
sns.regplot(x=dat['views'], y=dat['comment_count'], fit_reg=True, order=2)

# Legend/Axis labels/Title
plt.legend(['Views vs Likes', 'Views vs Dislikes', 'Views vs Comments'])
plt.title('Views vs Likes/Dislikes/Comments', size=17)
plt.xlabel('Number of Views', size=15)
plt.ylabel('Number of Likes/Dislikes/Comments', size=15)

plt.show()
```



`Views vs Likes` :

- There is a *fanning-out* effect on the scatter plot.
- Regression line is *non-linear* and rather a *power regression* line.
- Number of `views` increase much faster than that of `likes` .

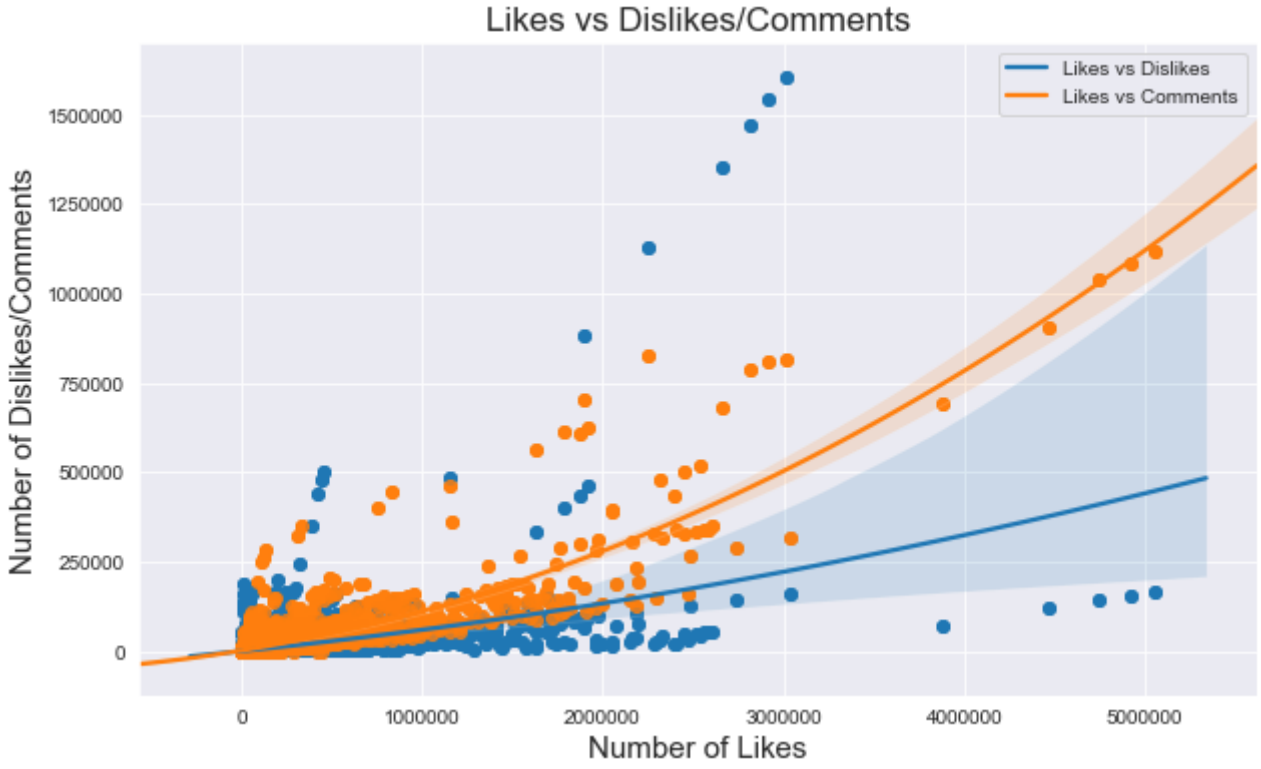
`Views vs Dislikes` :

- Regression line is *non-linear* and rather an *exponential* line.
- Number of `views` increase slower than that of `dislikes` .

`Views vs Comments` :

- Regression line is *non-linear* and rather *exponential* line.
- Number of `views` increase slower than that of `comments` .

In [22]:
```
# Set figure size and theme
fig, ax = plt.subplots(figsize=(10,6))
sns.set_style('darkgrid')

# Plot Scatter Plots
# plt.scatter(dat['likes'], dat['views'])
plt.scatter(dat['likes'], dat['dislikes'])
plt.scatter(dat['likes'], dat['comment_count'])

# Regression plot
sns.regplot(x=dat['likes'], y=dat['dislikes'], fit_reg=True, order=2)
sns.regplot(x=dat['likes'], y=dat['comment_count'], fit_reg=True, order=2)

# Legend/Axis labels/Title
plt.legend(['Likes vs Dislikes', 'Likes vs Comments'])
plt.title('Likes vs Dislikes/Comments', size=17)
plt.xlabel('Number of Likes', size=15)
plt.ylabel('Number of Dislikes/Comments', size=15)
plt.show()
```



Note: Rule out `views` since the number of `views` are significantly greater than `likes` / `dislikes` / `comments` , this makes it harder to see the relationship between just `likes` vs `dislikes` / `comments` .

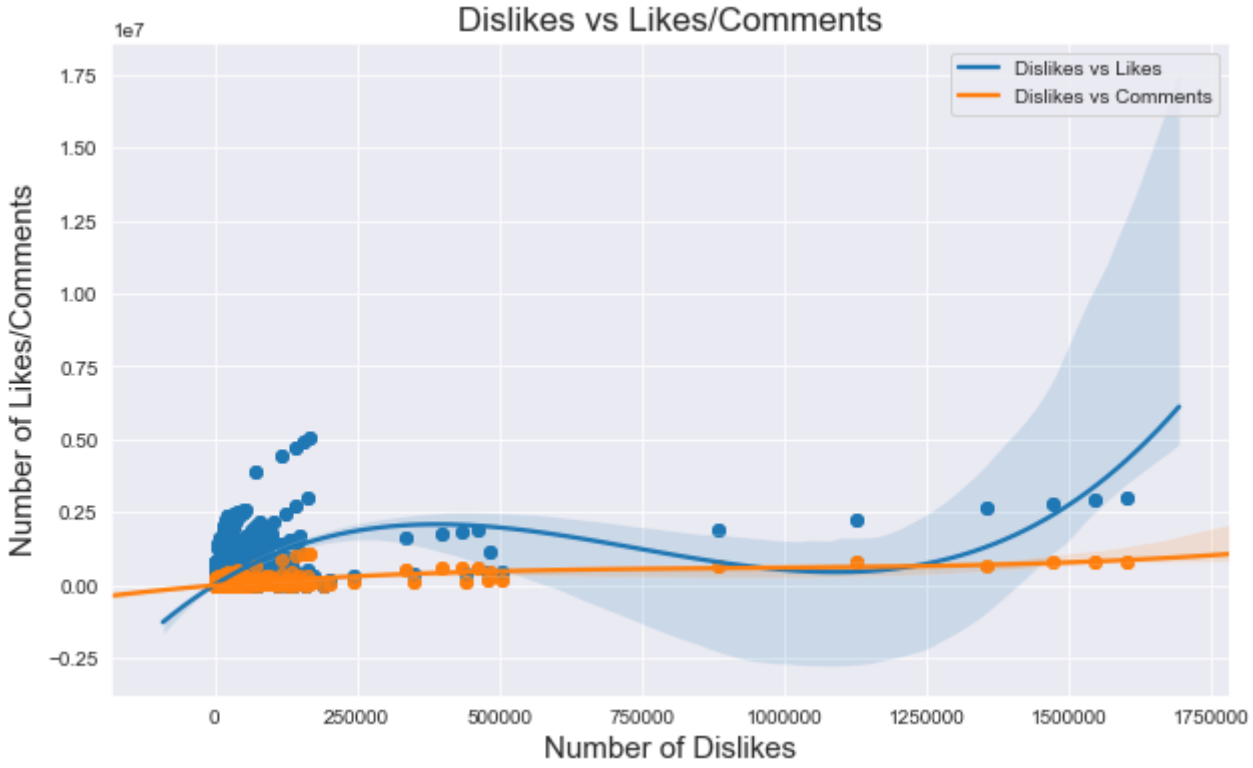
`Views vs Disikes` :

- Regression line is *non-linear* and rather an *exponential regression* line.
- Number of `likes` increase much slower than that of `dislikes` .

`Likes vs Comments` :

- Regression line is *non-linear* and rather an *exponential* line.
- Number of `likes` increase slower than that of `comments` .

In [24]:
```
# Set figure size and theme
fig, ax = plt.subplots(figsize=(10,6))
sns.set_style('darkgrid')

# Plot Scatter Plots
# plt.scatter(dat['dislikes'], dat['views'])
plt.scatter(dat['dislikes'], dat['likes'])
plt.scatter(dat['dislikes'], dat['comment_count'])

# Regression plot
sns.regplot(x=dat['dislikes'], y=dat['likes'], fit_reg=True, order=3)
sns.regplot(x=dat['dislikes'], y=dat['comment_count'], fit_reg=True, order=3)

# Legend/Axis labels/Title
plt.legend(['Dislikes vs Likes', 'Dislikes vs Comments'])
plt.title('Dislikes vs Likes/Comments', size=17)
plt.xlabel('Number of Dislikes', size=15)
plt.ylabel('Number of Likes/Comments', size=15)
plt.show()
```



Note: Again, rule out `views` for same reason.

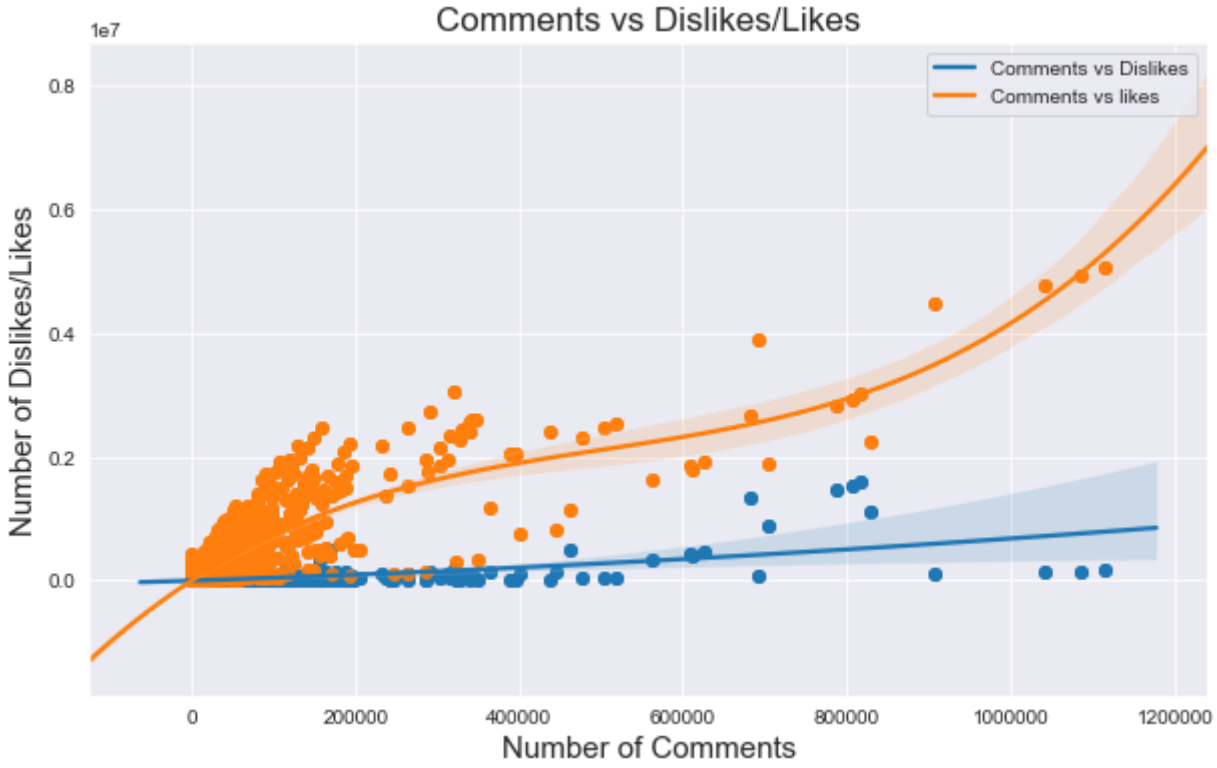
`Dislikes` vs `Likes`:

- Regression line is *non-linear* and rather an *logistic regression* line (inverse).
- As the number of `likes` increases, the number of `dislikes` levels off.

`Dislikes` vs `Comments`:

- Regression line looks somewhat linear.
- Number of `dislikes` increase faster than that of `comments`.

In [25]:
```
# Set figure size and theme
fig, ax = plt.subplots(figsize=(10,6))
sns.set_style('darkgrid')

# Plot Scatter Plots
# plt.scatter(dat['comment_count'], dat['views'])
plt.scatter(dat['comment_count'], dat['dislikes'])
plt.scatter(dat['comment_count'], dat['likes'])

# Regression plot
sns.regplot(x=dat['comment_count'], y=dat['dislikes'], fit_reg=True, order=2)
sns.regplot(x=dat['comment_count'], y=dat['likes'], fit_reg=True, order=3)

# Legend/Axis labels/Title
plt.legend(['Comments vs Dislikes', 'Comments vs likes'])
plt.title('Comments vs Dislikes/Likes', size=17)
plt.xlabel('Number of Comments', size=15)
plt.ylabel('Number of Dislikes/Likes', size=15)
plt.show()
```



Note: Rule out `views`.

`Comments` vs `Dislikes`:

- Regression line is *non-linear* and rather an *logistic regression* line (inverse).
- As the number of `dislikes` increases, the number of `comments` levels off.

`Comments` vs `Likes`:

- Regression line is *non-linear* and rather an *exponential* line.
- Number of `comments` increase slower than that of `likes`.