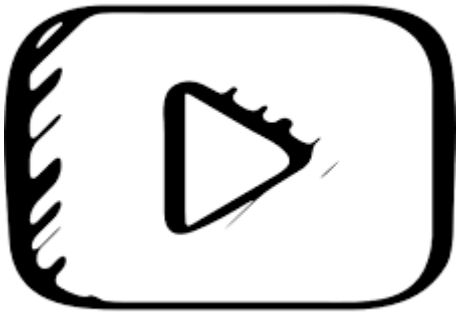


Youtube Statystics - Test ¶

The following test consist of three different tasks

- Finding what are categorical differences in viewing patterns betweene Germans and Indians
- Finding the ratios between views/likes/dislikes for different countries/categories
- Finding channels that are popular in most countries

Before we start with the tasks, we will import the essential packages for Data Science

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docke
r-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list al
l files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets p
reserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved out
side of the current session
```

```
/kaggle/input/youtube-new/GBvideos.csv
/kaggle/input/youtube-new/MXvideos.csv
/kaggle/input/youtube-new/KRvideos.csv
/kaggle/input/youtube-new/MX_category_id.json
/kaggle/input/youtube-new/GB_category_id.json
/kaggle/input/youtube-new/US_category_id.json
/kaggle/input/youtube-new/IN_category_id.json
/kaggle/input/youtube-new/DEvideos.csv
/kaggle/input/youtube-new/KR_category_id.json
/kaggle/input/youtube-new/RU_category_id.json
/kaggle/input/youtube-new/FRvideos.csv
/kaggle/input/youtube-new/USvideos.csv
/kaggle/input/youtube-new/INvideos.csv
/kaggle/input/youtube-new/RUvideos.csv
/kaggle/input/youtube-new/CA_category_id.json
/kaggle/input/youtube-new/DE_category_id.json
/kaggle/input/youtube-new/JP_category_id.json
/kaggle/input/youtube-new/JPvideos.csv
/kaggle/input/youtube-new/FR_category_id.json
/kaggle/input/youtube-new/CAvideos.csv
```

Task nr1 - What are the categorical differences in viewing patterns between Germans and Indians ?

First we have to lad the data, as we are evaluating differences between Germans and Indians we will name first dataframe "germans" and second "indians". We will also take a look at first rows of both dataframes and their shapes to avoid complications related to structure. For that reason, I we will also extract column names from the data.

In [2]:

```
# We will load the data from the CSVs into two separate dataframes, get their shap
e to check the difference in structure
germans = pd.read_csv('../input/youtube-new/DEvideos.csv', sep=',')
indians = pd.read_csv('../input/youtube-new/INvideos.csv', sep=',')


print(f'Germans - shape: {germans.shape}')
print(f'Indians - shape: {indians.shape}')
```

```
Germans - shape: (40840, 16)
Indians - shape: (37352, 16)
```

In [3]:

```
germans.head()
```

Out[3]:

	video_id	trending_date	title	channel_title	category_id	pub
0	LgVi6y5QljM	17.14.11	Sing zu Ende! | Gesangseinlagen vom Feinsten |...	inscope21	24	201 13T
1	Bayt7uQith4	17.14.11	Kinder ferngesteuert im Kiosk! Erwachsene abzo...	LUKE! Die Woche und ich	23	201 12T
2	1ZAPwfrtAFY	17.14.11	The Trump Presidency: Last Week Tonight with J...	LastWeekTonight	24	201 13T
3	AHtypnRk7JE	17.14.11	Das Fermi-Paradoxon	100SekundenPhysik	27	201 12T
4	ZJ9We4bjcg0	17.14.11	18 SONGS mit Kelly MissesVlog (Sing-off)	rezo	24	201 12T

In [4]:

```
# We will also look if the columns are the same for both dataframes
print(f'Germans - columns: {germans.columns}')
print('')
print(f'Indians - columns: {indians.columns}')

#germans_columns = germans.columns.to_list
#indians_columns = indians.columns.to_list
```

```
Germans - columns: Index(['video_id', 'trending_date', 'title', 'ch
annel_title', 'category_id',
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comme
nt_count',
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description'],
      dtype='object')

Indians - columns: Index(['video_id', 'trending_date', 'title', 'ch
annel_title', 'category_id',
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comme
nt_count',
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description'],
      dtype='object')
```

So far we know that there is a difference in shapes of both dataframe ***Germans - shape: (40840, 16) and Indians - shape: (37352, 16)*** Therefore we know that both dataframes have 16 columns but the germans have 3,5k records (that is videos) more than indians. We know hovewer that the structure of both dataframes is very simillar and that the column names are the same in both of them.

As my task is to get insights about viewing patterns, the only thing I will focus on is analyzing the parts of our data frames directly correlated to those patterns (columns: trending_date, views, likes, dislikes, comment_count)

In [5]:

```
# First we will examine the germans average number of likes and comments under vid
eo to know how strongly they are engaging with video we will also create and compa
re ratios of those engagemenst
germans[['views','comment_count','likes','dislikes']].mean()
```

Out[5]:

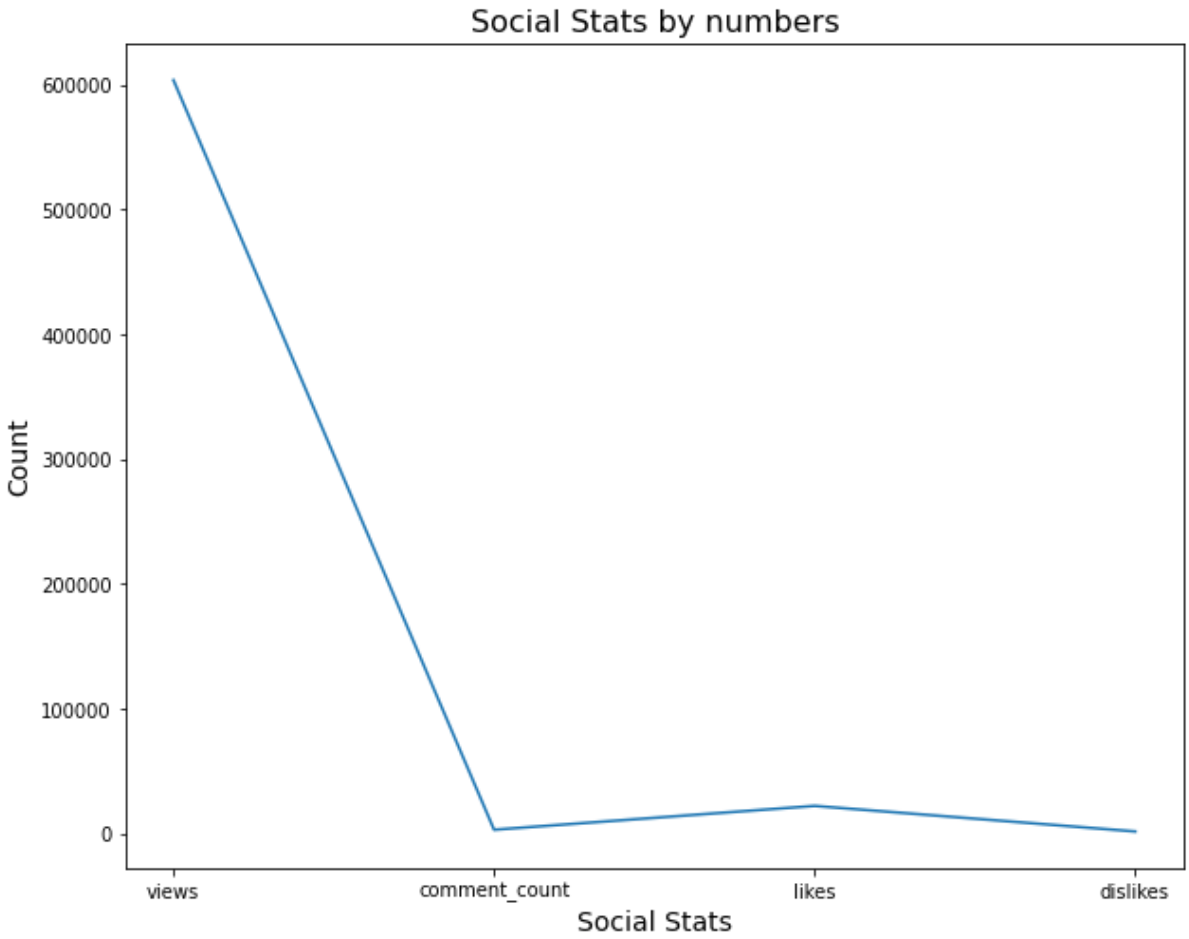
```
views           603455.318438
comment_count     2785.856513
likes            21875.502889
dislikes          1397.135921
dtype: float64
```

In [6]:

```
plt.figure(figsize = (10,8))
plt.plot(germans[['views','comment_count','likes','dislikes']].mean())
plt.ylabel('Count', fontsize=14)
plt.xlabel('Social Stats', fontsize =14)
plt.title("Social Stats by numbers", fontsize=16)
```

Out[6]:

```
Text(0.5, 1.0, 'Social Stats by numbers')
```



In [7]:

```
#We will find average engagement ratio for germany by comeparing average numbner o
f interactions to average number of views
germans_avg_interactions = sum(germans[['comment_count','likes','dislikes']].mea
n())
germans_avg_views = germans['views'].mean()

germans_engagement_ratio = (germans_avg_interactions/germans_avg_views) * 100

germans_engagement_ratio
```

Out[7]:

```
4.318214543318868
```

In [8]:

```
#And we will also do the same for indians
pd.set_option('display.float_format', lambda x: '%.6f' % x)
indians[['views','comment_count','likes','dislikes']].mean()
```

Out[8]:

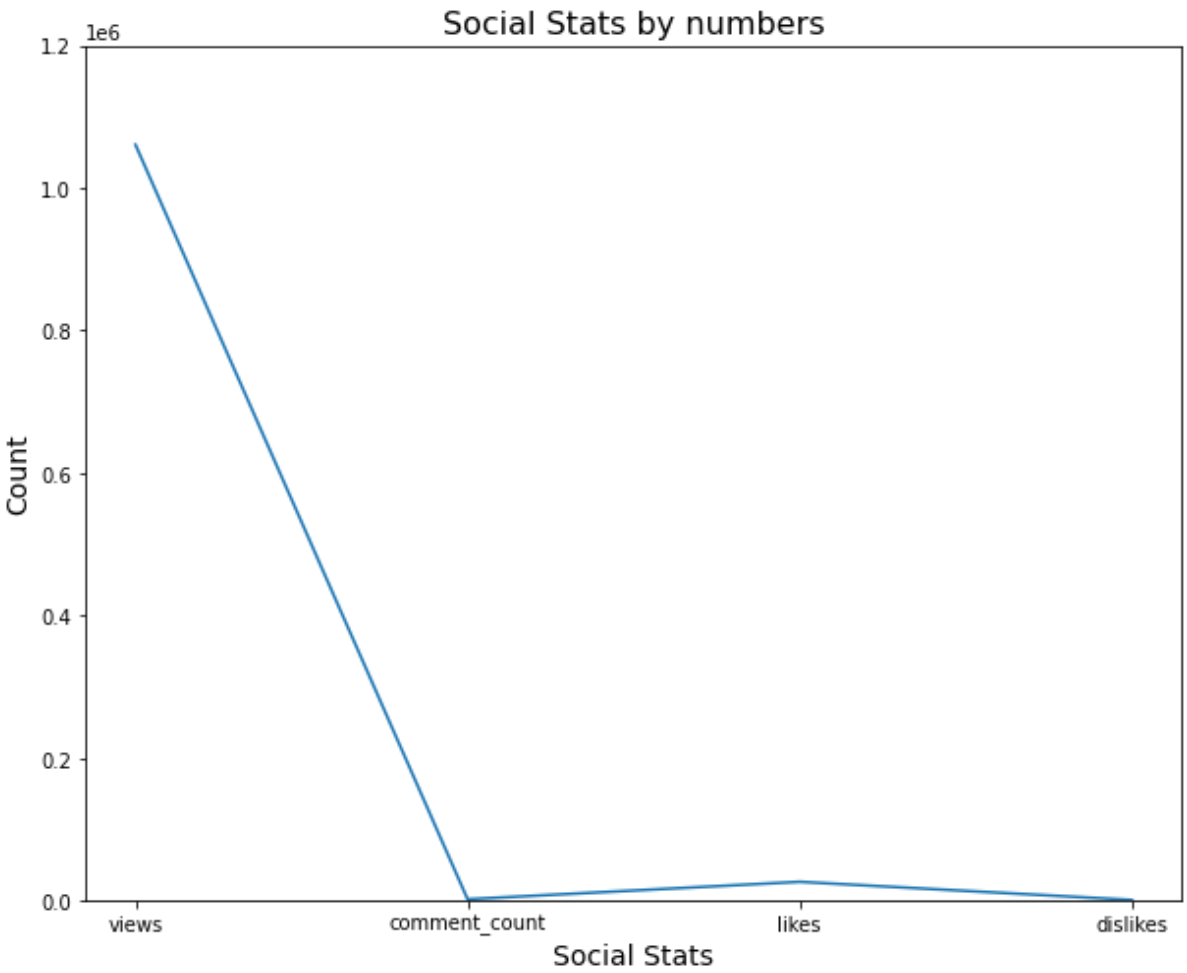
```
views           1060477.645882
comment_count      2676.997430
likes             27082.717659
dislikes           1665.081977
dtype: float64
```

In [9]:

```
plt.figure(figsize = (10,8))
plt.plot(['views','comment_count','likes','dislikes'],indians[['views','comment_
count','likes','dislikes']].mean())
plt.ylim(bottom=0, top=1200000)
plt.ylabel('Count', fontsize=14)
plt.xlabel('Social Stats', fontsize =14)
plt.title("Social Stats by numbers", fontsize=16)
```

Out[9]:

```
Text(0.5, 1.0, 'Social Stats by numbers')
```



In [10]:

```
indians_avg_interactions = sum(indians[['comment_count','likes','dislikes']].mea
n())
indians_avg_views = indians['views'].mean()

indians_engagement_ratio = (indians_avg_interactions/indians_avg_views) * 100

indians_engagement_ratio
```

Out[10]:

```
2.9632682204823357
```

In [11]:

```
#Now let's compare average engagement ratio for both countries
print(f'{germans_engagement_ratio/indians_engagement_ratio}')
```

```
1.4572472763251874
```

Above analysis shows us, that engagement ratio for germans is equal to **4.31%** and for indians around **2.96%**. Therefore on average **german videos recieve 46% more engagement than indian videos .**

Now we will explore if there is a difference, when those videos are trending.

In [12]:

```
# We are going to sort values in the datasets by trending_date column to detect if
videos in our dataframes were trending in the same time
germans = germans.sort_values(by="trending_date")
indians = indians.sort_values(by="trending_date")

indians
```

Out[12]:

	video_id	trending_date	title	channel_title	category_id	publish_
3256	6ZfuNTqbHE8	17.01.12	Marvel Studios' Avengers: Infinity War Officia...	Marvel Entertainment	24	2017-1 29T13:
3384	8Lvskj12y7k	17.01.12	வள்ளி | VALLI | Sun TV | Tamil Serial | Episod...	Saregama TVShows	24	2017-1 29T17:
3385	MM6d97DKOiw	17.01.12	Anbuchezhiyan is neither good nor bad : SV Sek...	IndiaGlitz Tamil Movies | Interviews | Shootin...	24	2017-1 29T10:
3386	CtSZLqdO-y4	17.01.12	A Boss Fight at the Consumer Financial Protect...	The Daily Show with Trevor Noah	23	2017-1 30T04:
3387	1vmhLprZYBg	17.01.12	Arsenal vs Huddersfield 5-0 All Goals & Highli...	Wrsh98	17	2017-1 29T22:
...	...	...	...	...	...	...
34779	ln53LeOFVuM	18.31.05	Lok Sabha Bypoll Result: BJP ahead of Shiv Sen...	Zee News	25	2018-0 31T08:
34778	nT96UHb74Ys	18.31.05	Taarak Mehta Ka Ooltah Chashmah - तारक मेहता -...	Sony PAL	24	2018-0 29T19:
34777	DjVdzb5sryI	18.31.05	Telangana Formation Day 2018 Special Song | By...	TeluguOne	10	2018-0 31T11:
34775	gyJsjJ0GXzM	18.31.05	News Fuse 30 May 2018 | ବିଜେଡିକୁ ବଳଦ ଧୋକା ଦେଲା...	OTV	25	2018-0 30T17:
34722	MZO8lgL2mN4	18.31.05	Result Prank | Fun Panrom | Black Sheep	Black Sheep	24	2018-0 29T15:

37352 rows × 16 columns

After sorting the dataframes we have to check if both dataframes have the time horizon of measurment.

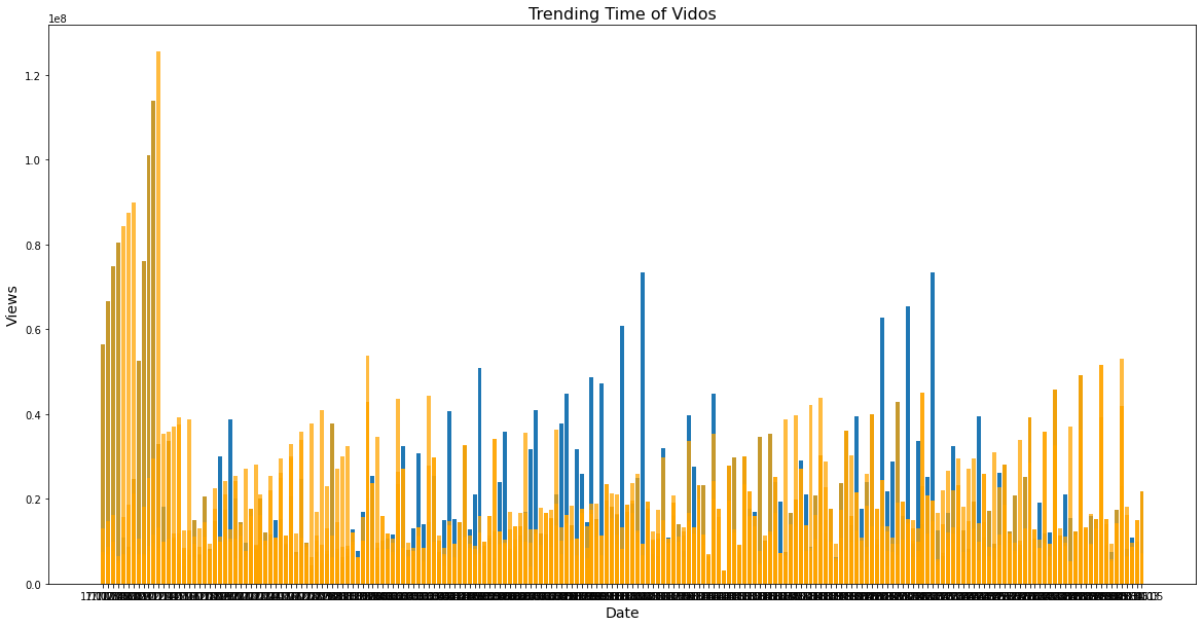
In [13]:

```
# We will chceck if germans and indians dataframes start on the same date
print('Indians')
print(indians['trending_date'].head(1))
print(indians['trending_date'].tail(1))
print('<----------------->')
print('Germans')
print(germans['trending_date'].head(1))
print(germans['trending_date'].tail(1))
```

```
Indians
3256    17.01.12
Name: trending_date, dtype: object
34722    18.31.05
Name: trending_date, dtype: object
<----------------->
Germans
3399    17.01.12
Name: trending_date, dtype: object
38031    18.31.05
Name: trending_date, dtype: object
```

In [14]:

```
import matplotlib.dates as mdates

plt.figure(figsize=(20,10))
plt.bar(germans['trending_date'],germans['views'], label='german')
plt.bar(indians['trending_date'], indians['views'], label='indian', color='orang
e', alpha=0.75)



plt.ylabel('Views', fontsize=14)
plt.xlabel('Date', fontsize =14)
plt.title("Trending Time of Vidos", fontsize=16)

plt.show()
```



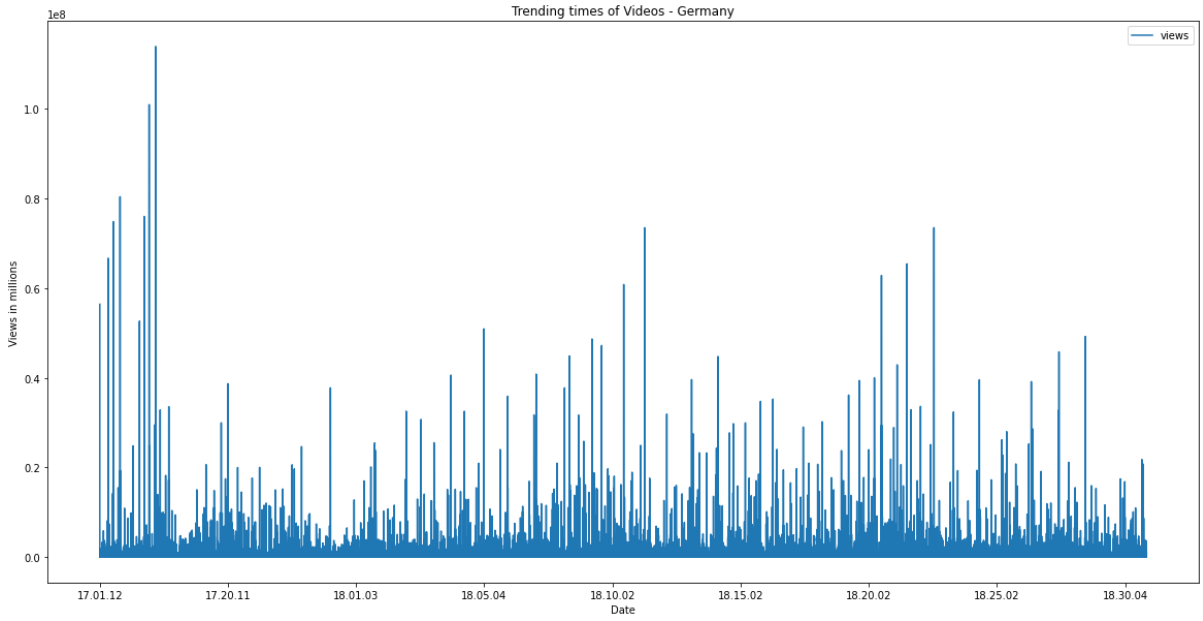
As we can see, the periods overlay each other, however for greater examination we will examine views for trading data on separate plots

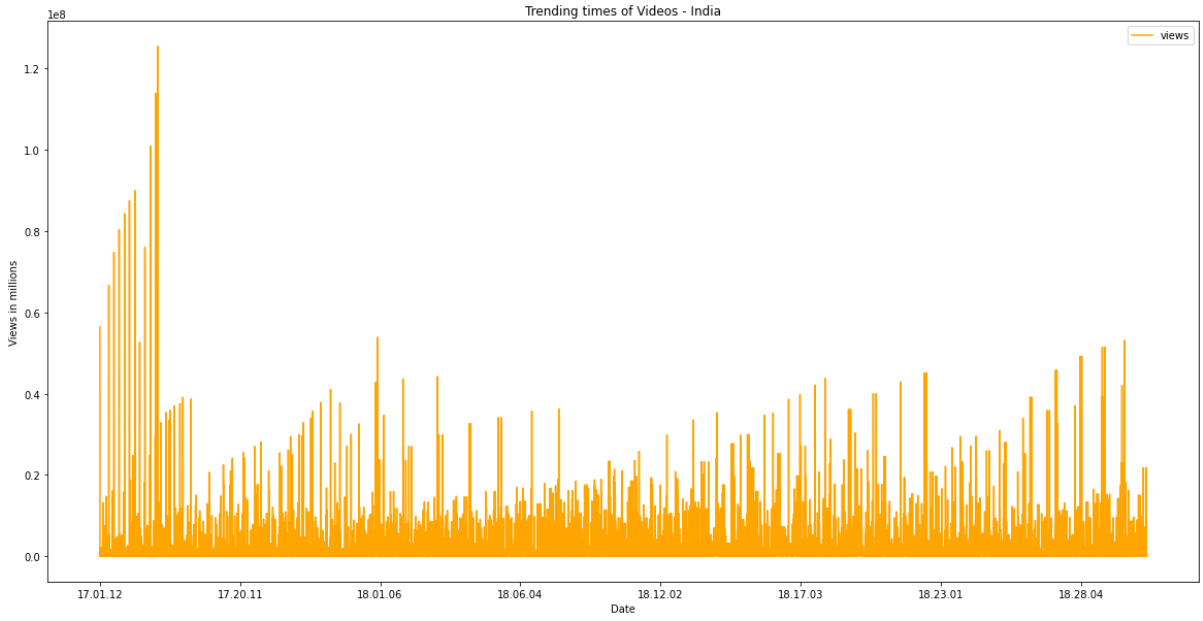
In [15]:

```
ax1 = germans.plot(y='views', x='trending_date', figsize = (20,10))
ax1.set(title='Trending times of Videos - Germany',ylabel = 'Views in millions',
xlabel = 'Date')

ax2 = indians.plot(y='views', x='trending_date', figsize = (20,10), color='orang
e')
ax2.set(title='Trending times of Videos - India', ylabel = 'Views in millions',
xlabel = 'Date')
```

Out[15]:

```
[Text(0.5, 1.0, 'Trending times of Videos - India'),
 Text(0, 0.5, 'Views in millions'),
 Text(0.5, 0, 'Date')]
```





As we can see, videos weren't trending exactly in the same time for all the time. In individual dates Germany as well as India registered sudden jumps in their view count. The first two quaters of 2017 recodred maximum view counts for individual dates for both countries.

We will also check the amount of views for a different measures of the view structure to see precisely the difference between those two countries.

In [16]:

```
print("Germans:")

print(f"Germans max views : {germans['views'].max()}")
print(f"Germans min views : {germans['views'].min()}")
print(f"Germans cumulative views : {germans['views'].sum()}")
print(f"Germans average views : {germans['views'].mean()}")
print(f"Germans median views : {germans['views'].median()}")

print("")
print("Indians: ")

print(f"Indians max views : {indians['views'].max()}")
print(f"Indians min views : {indians['views'].min()}")
print(f"Indians cumulative views : {indians['views'].sum()}")
print(f"Indians average views : {indians['views'].mean()}")
print(f"Indians median views : {indians['views'].median()}")


#We will also print differences
print("")
print("Differences: ")

print(f"Difference max views : {indians['views'].max() - germans['views'].max()}
")
print(f"Difference min views : {indians['views'].min() - germans['views'].min()}
")
print(f"Difference cumulative views : {indians['views'].sum() - germans['views']
.sum()} | in percentage {(germans['views'].sum() / indians['views'].sum()) *100}
%")
print(f"Difference average views : {indians['views'].mean() - germans['views'].m
ean()} | in percentage {(germans['views'].mean() / indians['views'].mean()) *100
}%")
print(f"Difference median views : {indians['views'].median() - germans['views'].
median()} | in percentage {(germans['views'].median() / indians['views'].median
()) *100}%")

#We will also print percentages for germans

print("")

print(f"Germans obtained {round((germans['views'].max() / indians['views'].max
()),2) *100}% of indians max views ")
```

```
print(f"Germans obtained {round((germans['views'].min() / indians['views'].min
()),2) *100}% of indians min views ")
print(f"Germans obtained {round((germans['views'].sum() / indians['views'].sum
()),2) *100}% of indians cummulative views ")
print(f"Germans obtained {round((germans['views'].mean() / indians['views'].mean
()),1) *100}% of indians mean views ")
print(f"Germans obtained {round((germans['views'].median() / indians['views'].me
dian()),2) *100}% of indians median views ")
```

```
Germans:
Germans max views : 113876217
Germans min views : 518
Germans cumulative views : 24645115205
Germans average views : 603455.3184378061
Germans median views : 119277.0

Indians:
Indians max views : 125432237
Indians min views : 4024
Indians cumulative views : 39610961029
Indians average views : 1060477.645882416
Indians median views : 304586.0

Differences:
Difference max views : 11556020
Difference min views : 3506
Difference cumulative views : 14965845824 | in percentage 62.217917
881257165%
Difference average views : 457022.32744460995 | in percentage 56.90
410550197643%
Difference median views : 185309.0 | in percentage 39.1603685001937
05%

Germans obtained 91.0% of indians max views
Germans obtained 13.0% of indians min views
Germans obtained 62.0% of indians cummulative views
Germans obtained 60.0% of indians mean views
Germans obtained 39.0% of indians median views
```

The statystics shows that there is significant difference between conuntry's most and worst performing videos. Indians best performing video has 11556020 more views than German best performing video, Indians also have 3,5k more views under their worst performing video. There is almost **15 billion views** difference in cummulative number of views for both countries. Indians also have higher number of average views and the median view count for them is larger by almost 20k.

Those statistics are not suprising though. According to worldometers.io (https://www.worldometers.info/) , India is the country with 1.4 billion people while germany has 84 million people. The average age for an indian is 29 years old while in germany the average age is 45 years old. Younger genereations have a tendency to adopt to new technologies (including mobile) that are the main trafic driving sources for social media platforms in todays world (research.com (http://https://research.com/software/mobile-vs-desktop-usage)).

Neverthless, germany's audience is stronger in sense as they are able to obtain almost 40% of indians median views and 60% of their mean views, even though india has 15 times more people than germany.

We can also examine how those statistics look in different percentiles. **Below table also show us standard deviation** (mesurment of diference from average value), we can see that its a little bit higher for germans

In [17]:

```
germans.describe()
```

Out[17]:

	category_id	views	likes	dislikes	comm
count	40840.000000	40840.000000	40840.000000	40840.000000	4084(
mean	20.705828	603455.318438	21875.502889	1397.135921	2785.
std	6.975813	2348962.768206	101799.997726	14577.383851	17458
min	1.000000	518.000000	0.000000	0.000000	0.000
25%	20.000000	27068.750000	533.000000	29.000000	79.00
50%	24.000000	119277.000000	2699.000000	134.000000	376.0
75%	24.000000	443101.500000	11796.250000	532.000000	1376.
max	44.000000	113876217.000000	4924056.000000	1470386.000000	10844

In [18]:

```
indians.describe()
```

Out[18]:

	category_id	views	likes	dislikes	comment
count	37352.000000	37352.000000	37352.000000	37352.000000	37352.0
mean	21.576596	1060477.645882	27082.717659	1665.081977	2676.99
std	6.556593	3184932.053381	97145.095131	16076.174539	14868.3
min	1.000000	4024.000000	0.000000	0.000000	0.00000
25%	23.000000	123915.500000	864.000000	108.000000	81.0000
50%	24.000000	304586.000000	3069.000000	326.000000	329.000
75%	24.000000	799291.250000	13774.250000	1019.250000	1285.00
max	43.000000	125432237.000000	2912710.000000	1545017.000000	827755.

Task nr 2

- **Compare ratios of views,likes,dislikes for different categories/countries.**
- **Is it easier for musicians to collect likes than for politicians ?**
- **What is the sentiment to videos between Mexicans and Brits ?**

Before anything else, we would have to open multiple csvs and json files. We will also create a lists of unique codes and unique countries.

In [19]:

```
# To open all csv's at once we are gona use loop thrugh list of all the csv files
 in our directory to access each of them, then we will create dataframe from them.
import glob
files = [i for i in glob.glob('../input/youtube-new/*.csv')]
sorted(files)
```

Out[19]:

```
['../input/youtube-new/CAvideos.csv',
 '../input/youtube-new/DEvideos.csv',
 '../input/youtube-new/FRvideos.csv',
 '../input/youtube-new/GBvideos.csv',
 '../input/youtube-new/INvideos.csv',
 '../input/youtube-new/JPvideos.csv',
 '../input/youtube-new/KRvideos.csv',
 '../input/youtube-new/MXvideos.csv',
 '../input/youtube-new/RUvideos.csv',
 '../input/youtube-new/USvideos.csv']
```

In [20]:

```
dfs = list()
for csv in files:
    df = pd.read_csv(csv,encoding='latin1')
    df['country'] = csv[21:23]
    dfs.append(df)

all_df = pd.concat(dfs)
all_df.head(3)
```

Out[20]:

	video_id	trending_date	title	channel_title	category_id	publish_time
0	Jw1Y-zhQURU	17.14.11	John Lewis Christmas Ad 2017 - #MozTheMonster	John Lewis	26	2017-11-10T07:38:2
1	3s1rvMFUweQ	17.14.11	Taylor Swift: â ¦Ready for It? (Live) - SNL	Saturday Night Live	24	2017-11-12T06:24:4
2	n1WpP7iowLc	17.14.11	Eminem - Walk On Water (Audio) ft. BeyoncÃ©	EminemVEVO	10	2017-11-10T17:00:0

In [21]:

```
#List of unique countries

countries = all_df.country.unique()

countries = countries.tolist()
countries
```

Out[21]:

```
['GB', 'MX', 'KR', 'DE', 'FR', 'US', 'IN', 'RU', 'JP', 'CA']
```

In [22]:

```
#List of unique codes
categories = all_df.category_id.unique()

categories = categories.tolist()
categories = sorted(categories)
categories
```

Out[22]:

```
[1, 2, 10, 15, 17, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 43,
44]
```

In [23]:

```
#Load the json the files:
files = [i for i in glob.glob('../input/youtube-new/*.json')]
sorted(files)

dfs = list()
for json in files:
    df = pd.read_json(json)
    df['country'] = json[21:23]
    dfs.append(df)

jsons_df = pd.concat(dfs)
jsons_df.head(3)
```

Out[23]:

	kind	etag
0	youtube#videoCategoryListResponse	"XI7nbFXulYBIpL0ayR_gDh3eu1k/1v2mrzYSYG6onNLt2
1	youtube#videoCategoryListResponse	"XI7nbFXulYBIpL0ayR_gDh3eu1k/1v2mrzYSYG6onNLt2
2	youtube#videoCategoryListResponse	"XI7nbFXulYBIpL0ayR_gDh3eu1k/1v2mrzYSYG6onNLt2

In [24]:

```
#As categories_id is sorted and values from cv are
cn = {category['id']: category['snippet']['title'] for category in jsons_df['ite
ms']}

categories_names = list()

for key in categories:
    categories_names.append(cn.get(f'{key}'))

categories_names
```

Out[24]:

```
['Film & Animation',
 'Autos & Vehicles',
 'Music',
 'Pets & Animals',
 'Sports',
 'Travel & Events',
 'Gaming',
 'People & Blogs',
 'Comedy',
 'Entertainment',
 'News & Politics',
 'Howto & Style',
 'Education',
 'Science & Technology',
 'Nonprofits & Activism',
 'Movies',
 'Shows',
 'Trailers']
```

Now when we have both lists, we can group the dataframes, extract values for views,likes and dislikes from them and create new dataframe with our ratios

We will mark the ratios as follows:

- L2V - Likes to views
- D2V - Dislikes to views
- L2D - Likes to Dislikes

In [25]:

```
#Compare ratios for different countries

gcountry_df = all_df.groupby('country', as_index=False).sum()


gcountry_df.loc[gcountry_df['country']=='CA']

list_of_views = list()
list_of_likes = list()
list_of_dislikes = list()

Ratio_df = pd.DataFrame()


for c in range(len(countries)):
    x = int(gcountry_df[gcountry_df['country']==f'{countries[c]}'].views)
    y = int(gcountry_df[gcountry_df['country']==f'{countries[c]}'].likes)
    z = int(gcountry_df[gcountry_df['country']==f'{countries[c]}'].dislikes)
    list_of_views.append(x)
    list_of_likes.append(y)
    list_of_dislikes.append(z)

ratio_df = pd.DataFrame({'country':countries, 'views': list_of_views, 'likes': l
ist_of_likes, 'dislikes': list_of_dislikes})
ratio_df.head()

ratio_df['L2V'] = ratio_df['likes']/ratio_df['views']
ratio_df['D2V'] = ratio_df['dislikes']/ratio_df['views']
ratio_df['L2D'] = ratio_df['likes']/ratio_df['dislikes']


ratio_df
```

Out[25]:

	country	views	likes	dislikes	L2V	D2V	L2D
0	GB	230069198174	5234962944	296250384	0.022754	0.001288	17.67073
1	MX	13849692994	641627186	30223385	0.046328	0.002182	21.22949
2	KR	14689152313	421247912	18634999	0.028677	0.001269	22.60520
3	DE	24645115205	893395538	57059031	0.036250	0.002315	15.65739
4	FR	17100897444	708144090	33188528	0.041410	0.001941	21.33701
5	US	96671770152	3041147198	151978155	0.031458	0.001572	20.01042
6	IN	39610961029	1011593670	62194142	0.025538	0.001570	16.26509
7	RU	9806494525	506598491	60098157	0.051659	0.006128	8.429518
8	JP	5377466630	165406898	7528321	0.030759	0.001400	21.97128
9	CA	46891975069	1618179878	82137919	0.034509	0.001752	19.70076

As we can see in the first two rows, british have significantly lower count of likes to dislikes (they have 17.5 times more likes than dislikes) in commparison to the Mexicans (they have 21.2 times more likes than dislikes)

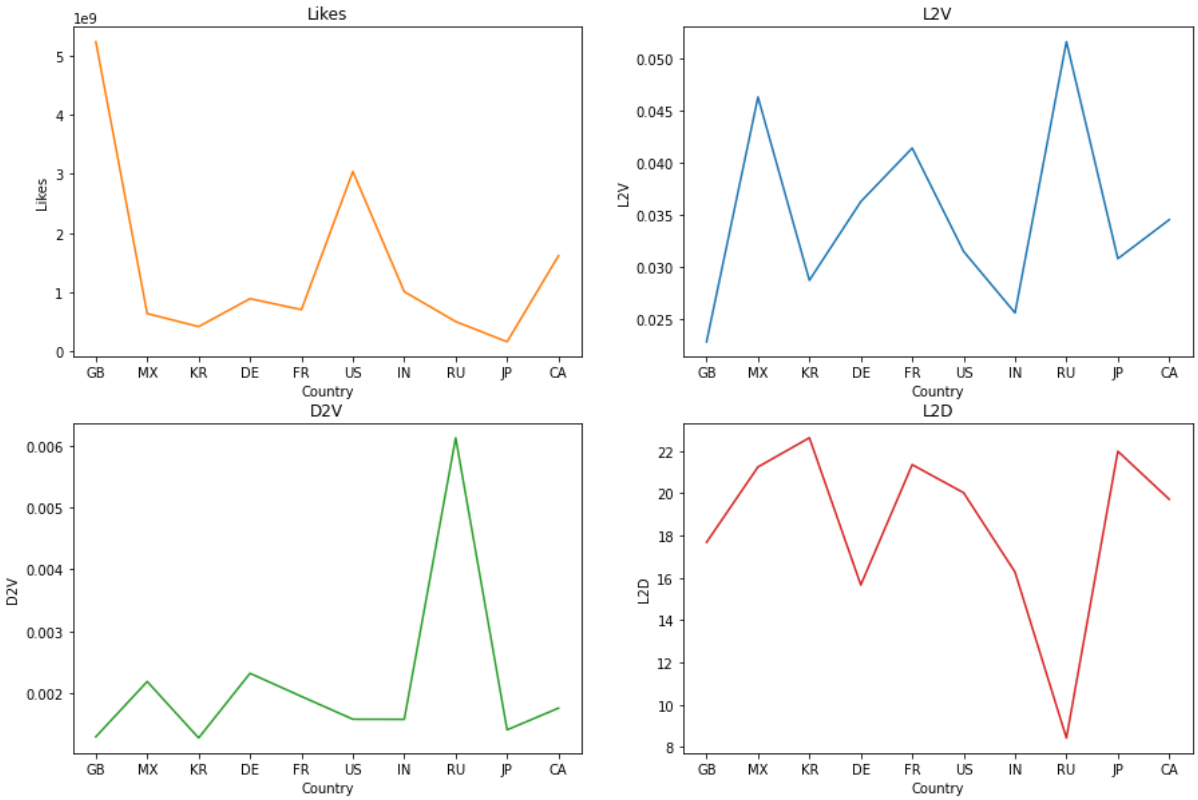
Well we can even visualize those ratios using multiple line charts

In [26]:

```
fig, axs = plt.subplots(2, 2, figsize=(15,10))


axs[0,0].plot(ratio_df['country'], ratio_df['likes'], 'tab:orange')
axs[0,0].set(xlabel = 'Country',ylabel = 'Likes', title='Likes')

axs[0,1].plot(ratio_df['country'], ratio_df['L2V'])
axs[0,1].set(xlabel = 'Country',ylabel = 'L2V', title = 'L2V')

axs[1,0].plot(ratio_df['country'], ratio_df['D2V'], 'tab:green')
axs[1,0].set(xlabel = 'Country',ylabel = 'D2V', title = 'D2V')

axs[1,1].plot(ratio_df['country'], ratio_df['L2D'], 'tab:red')
axs[1,1].set(xlabel = 'Country',ylabel = 'L2D', title = 'L2D')
```

Out[26]:

```
[Text(0.5, 0, 'Country'), Text(0, 0.5, 'L2D'), Text(0.5, 1.0, 'L2
D')]
```



In [27]:

```
# We need a list of all categories
category_df = all_df.groupby('category_id', as_index=False).sum()
category_df.head(5)
```

Out[27]:

	category_id	views	likes	dislikes	comment_count	comments_dis
0	1	27619347901	589885590	25279207	65387125	536
1	2	1661853766	45461895	2571460	5957385	73
2	10	255967088943	7227198427	294657819	620030515	236
3	15	2008474231	56601492	1503766	8103678	94
4	17	18972425164	399630743	26536025	46998109	513

In [28]:

```
category_df = all_df.groupby('category_id', as_index=False).sum()


list_of_views = list()
list_of_likes = list()
list_of_dislikes = list()


for c in range(len(categories)):
    x = (category_df[category_df['category_id'] == categories[c]]).views.values
    y = (category_df[category_df['category_id'] == categories[c]]).likes
    z = (category_df[category_df['category_id'] == categories[c]]).dislikes

    list_of_views.append(int(x))
    list_of_likes.append(int(y))
    list_of_dislikes.append(int(z))


ratio_categories_df = pd.DataFrame({'category_id':categories, 'views': list_of_v
iews, 'likes': list_of_likes, 'dislikes': list_of_dislikes})

ratio_categories_df['L2V'] = ratio_categories_df['likes']/ratio_categories_df['v
iews']
ratio_categories_df['D2V'] = ratio_categories_df['dislikes']/ratio_categories_df
['views']
ratio_categories_df['L2D'] = ratio_categories_df['likes']/ratio_categories_df['d
islikes']


ratio_categories_df = ratio_categories_df.sort_values(by='category_id')
ratio_categories_df.insert(1,'category_name', categories_names)

ratio_categories_df
```

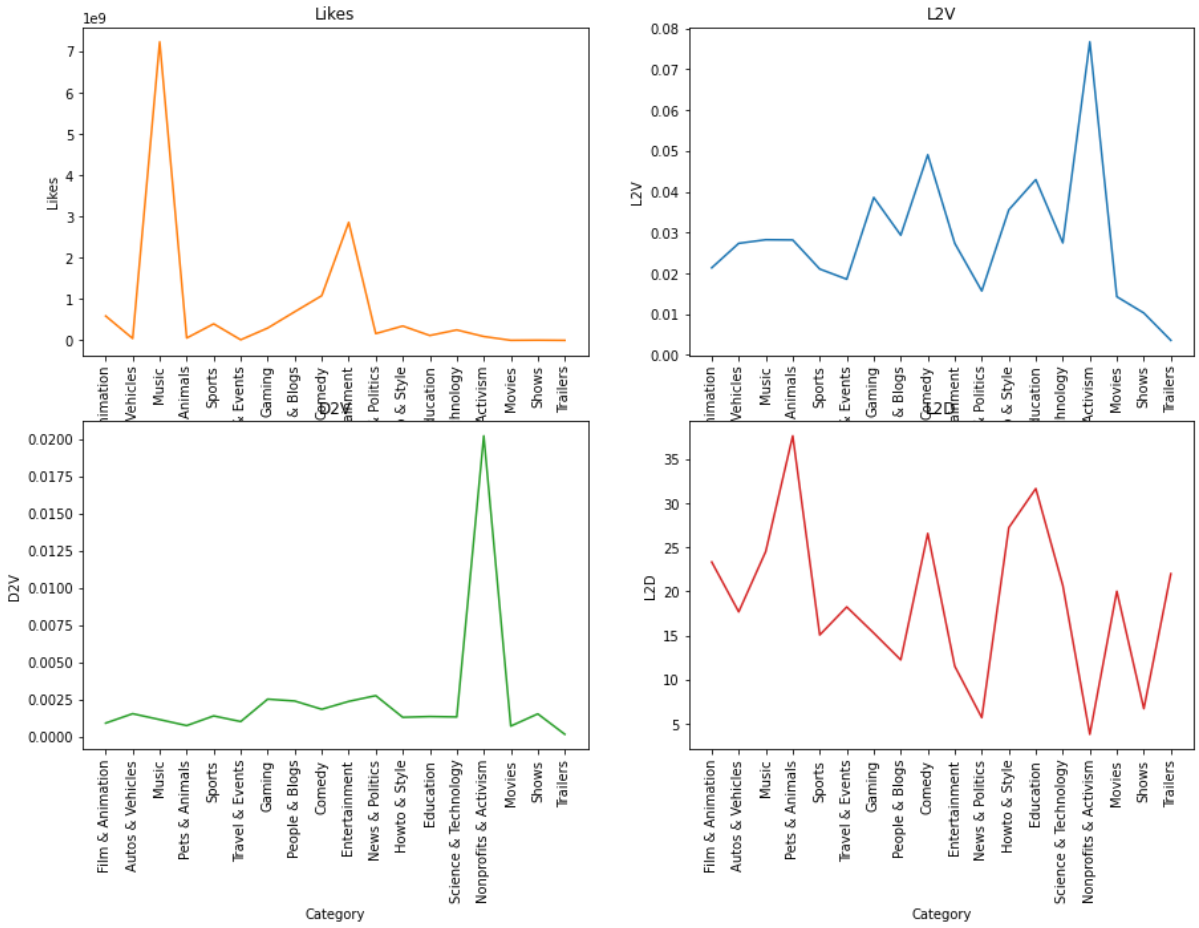
Out[28]:

	category_id	category_name	views	likes	dislikes	L2V
0	1	Film & Animation	27619347901	589885590	25279207	0.021358
1	2	Autos & Vehicles	1661853766	45461895	2571460	0.027356
2	10	Music	255967088943	7227198427	294657819	0.028235
3	15	Pets & Animals	2008474231	56601492	1503766	0.028181
4	17	Sports	18972425164	399630743	26536025	0.021064
5	19	Travel & Events	726674959	13494079	739962	0.018570
6	20	Gaming	7730729502	298337663	19534374	0.038591
7	22	People & Blogs	23600365409	692550961	56634003	0.029345
8	23	Comedy	22050866339	1081392644	40698333	0.049041
9	24	Entertainment	104517467253	2857743591	248270342	0.027342
10	25	News & Politics	10422502991	163503422	28778398	0.015688
11	26	Howto & Style	9771031927	347338295	12756984	0.035548
12	27	Education	2734841410	117479047	3710565	0.042956
13	28	Science & Technology	9194715151	252570921	12218574	0.027469
14	29	Nonprofits & Activism	1219859213	93538593	24670453	0.076680
15	30	Movies	70359777	1005417	50242	0.014290
16	43	Shows	444064556	4570827	682505	0.010293
17	44	Trailers	55043	198	9	0.003597

Now we can visualize statystics for categories

In [29]:

```
fig, axs = plt.subplots(2, 2, figsize=(15,10))



axs[0,0].plot(ratio_categories_df['category_name'], ratio_categories_df['likes'
], 'tab:orange')
axs[0,0].set(xlabel = 'Category',ylabel = 'Likes', title='Likes')
axs[0,0].tick_params(axis='x', rotation=90)

axs[0,1].plot(ratio_categories_df['category_name'], ratio_categories_df['L2V'])
axs[0,1].set(xlabel = 'Category',ylabel = 'L2V', title = 'L2V')
axs[0,1].tick_params(axis='x', rotation=90)

axs[1,0].plot(ratio_categories_df['category_name'], ratio_categories_df['D2V'],
'tab:green')
axs[1,0].set(xlabel = 'Category',ylabel = 'D2V', title = 'D2V')
axs[1,0].tick_params(axis='x', rotation=90)

axs[1,1].plot(ratio_categories_df['category_name'], ratio_categories_df['L2D'],
'tab:red')
axs[1,1].set(xlabel = 'Category',ylabel = 'L2D', title = 'L2D')
axs[1,1].tick_params(axis='x', rotation=90)
```



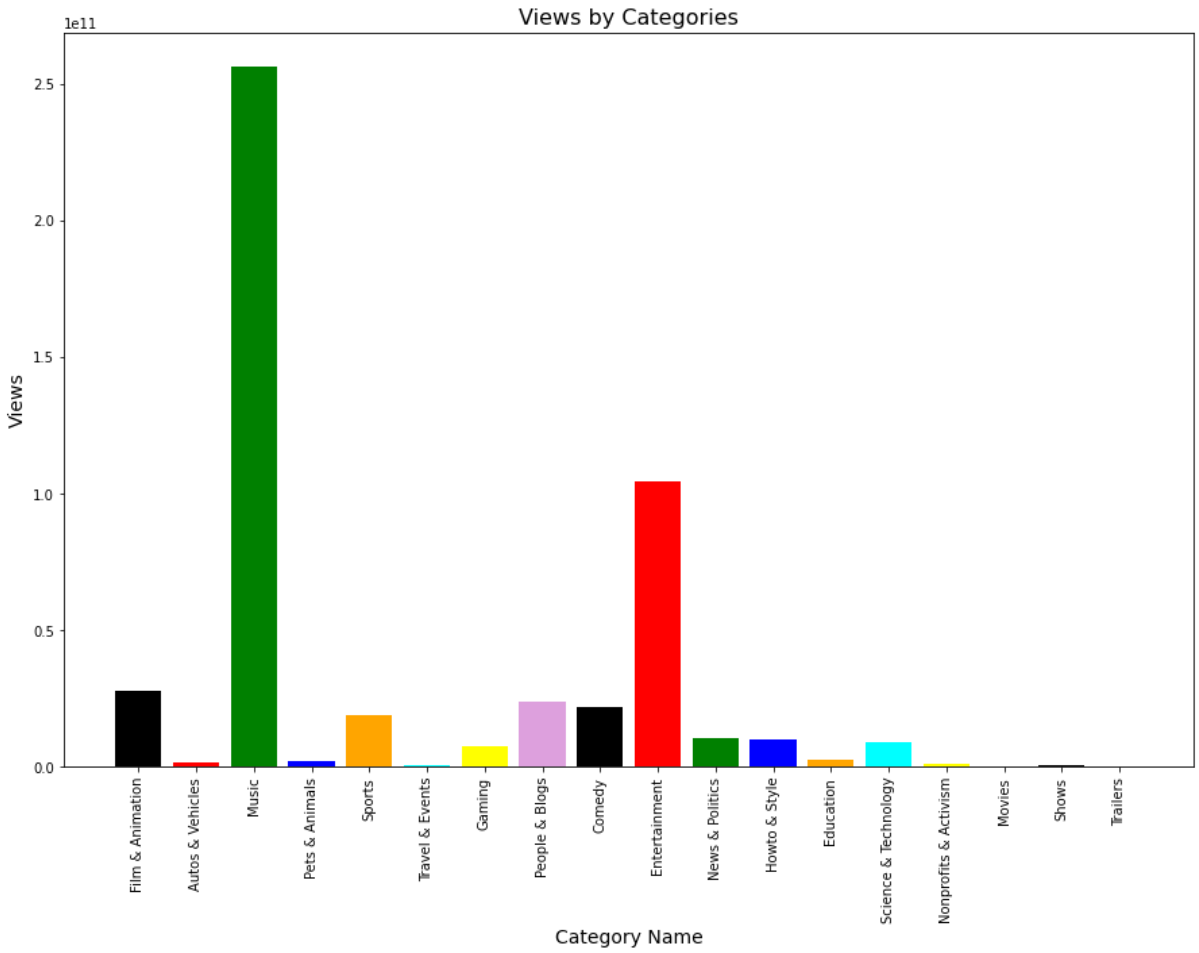
Let's examine in which category is the easiest to get views

In [30]:

```
plt.figure(figsize = (15,10))
plt.xticks(rotation='vertical')
plt.bar(ratio_categories_df['category_name'], ratio_categories_df['views'], colo
r=['black', 'red', 'green', 'blue','orange', 'cyan', 'yellow', 'plum'])
plt.ylabel('Views', fontsize=14)
plt.xlabel('Category Name', fontsize =14)
plt.title("Views by Categories", fontsize=16)
```

Out[30]:

```
Text(0.5, 1.0, 'Views by Categories')
```



As shown on the chart, the music category surpases everything else but for task completion we will examine how does the music category compare to politics.

In [31]:

```
#Check if it is easier to collect likes for musicians than for politicians ?
#To do that we have to detect all music channels and all political channels

music = int(ratio_categories_df[ratio_categories_df['category_name'] == 'Music']
['views'].values)

politics = int(ratio_categories_df[ratio_categories_df['category_name'] == 'News
& Politics']['views'].values)


print(f'Cumulative views in Music category : {music}')

print("")

print(f'Cumulative views in Politics category: {politics}')
print("")

if music > politics:
    print(f'Its easier to get views in Music, than in Politics ')
else:
    print(f'Its easier to get views in Politics, than in Music')
```

```
Cumulative views in Music category : 255967088943

Cumulative views in Politics category: 10422502991

Its easier to get views in Music, than in Politics
```

Musicians obtained 255967088943 views and **politicians 10422502991**. Based on that data we can conclude, that it is significantly harder for politicians to obtain views in comparison to musicians. **About 24,5 times more difficult to be exact**.

Running task nr 3

Select channels that are popular in most countries.

> There is essentialy couple of ways we can approach this problem. One of them is to find top channels from a given contry and check if any of those channels are also popular in other countries. The other method is to set some benchmark (cumulative views for the channel), that will help to estimate if a channel is popular or not. Both approaches have theirs pros and cons as the first one lets us find channels that recive their views from truly international audience but cannot gurantee that the channel will apear in majority of examined countries so the research might not be worth the effort. Second method however doesn't specify the country from which the views are comeing from so we can only guess that the channel is internationally popular bcause of its amount outstanding amount of views. We will therefore combine both approaches for the best result

In [32]:

```
#Ratios agin

gcountry_df = all_df.groupby('country', as_index=False).sum()
gcountry_df.head()
```

Out[32]:

	country	category_id	views	likes	dislikes	comment_count
0	CA	850143	46891975069	1618179878	82137919	206161849
1	DE	845626	24645115205	893395538	57059031	113774380
2	FR	819522	17100897444	708144090	33188528	74624804
3	GB	654876	230069198174	5234962944	296250384	509346351
4	IN	805929	39610961029	1011593670	62194142	99991208

In [33]:

```
#Find channels with most uploads (One row in our dataframe is one video)

top_uploaders = all_df['channel_title'].value_counts()
top_uploaders.head(10)
```

Out[33]:

```
The Late Show with Stephen Colbert        984
WWE                                       804
Late Night with Seth Meyers               773
VikatanTV                                 763
TheEllenShow                              743
Jimmy Kimmel Live                         707
The Tonight Show Starring Jimmy Fallon    705
PewDiePie                                 652
RadaanMedia                               651
The Late Late Show with James Corden      583
Name: channel_title, dtype: int64
```

We will try to find our benchmark by using percentiles. 90th percentile is a good estimate as it will separate 10% of most popular videos from the rest.

In [34]:

```
df = all_df.sort_values(by='views', ascending=True)
df['views'].quantile(0.90)
```

Out[34]:

```
2093287.0
```

We can see that the 90th percentile is around 2million, therefore that is gonna be our benchmark for entry

In [35]:

```
df = all_df.sort_values(by='views', ascending=True)

gruped = df.groupby(['country','channel_title'], as_index=False).sum()
gruped.drop(['comment_count','comments_disabled','ratings_disabled','video_error
_or_removed'], axis=1, inplace=True)



list_of_df = []

# The following function will help us find channels from a given country with a sp
ecific condition

def FindPopular():
    for c in countries:
        #This conditions specifies that the channel must at least ten million view
s to be considered popular
        condition = gruped['views'] >= 20000000
        c = gruped[(gruped['country'] == f'{c}') & (condition)]
        df = pd.DataFrame(c).sort_values(by='views', ascending= False).head(100)
        list_of_df.append(df)

FindPopular()
```

Now, when we have a list of dataframes with top 10% of channels by views from every country, we can create one big dataframe out of them and find which channels are occurning the most in it. Therefore we will find countries that are popular in m

In [36]:

```
#Lets find 30 most viewed channels in  multiple countries
concat_df = pd.concat(list_of_df)
most_viewed = concat_df['channel_title'].value_counts()
most_viewed.head(30)
```

Out[36]:

```
TaylorSwiftVEVO                    10
20th Century Fox                   10
Marvel Entertainment               10
Sony Pictures Entertainment        10
ibighit                             9
Universal Pictures                  9
Maroon5VEVO                         9
YouTube Spotlight                   9
Dude Perfect                        9
Warner Bros. Pictures               9
ChildishGambinoVEVO                 9
SpaceX                              9
Clash Royale                        8
Kylie Jenner                        8
Ed Sheeran                          8
Paramount Pictures                  8
ArianaGrandeVevo                    8
PewDiePie                           8
DrakeVEVO                           8
MLG Highlights                      7
justintimberlakeVEVO                7
AsapSCIENCE                         7
Disneyâ ¢Pixar                      7
EminemVEVO                          7
SMTOWN                              7
WWE                                 7
jypentertainment                    7
BuzzFeedVideo                       7
5-Minute Crafts                     6
The Late Show with Stephen Colbert  6
Name: channel_title, dtype: int64
```

Now we can visualize those channel names in a form of word cloud

In [37]:

```
to_word_cloud = most_viewed.head(30).index.to_list()
to_word_cloud
```

Out[37]:

```
['TaylorSwiftVEVO',
 '20th Century Fox',
 'Marvel Entertainment',
 'Sony Pictures Entertainment',
 'ibighit',
 'Universal Pictures',
 'Maroon5VEVO',
 'YouTube Spotlight',
 'Dude Perfect',
 'Warner Bros. Pictures',
 'ChildishGambinoVEVO',
 'SpaceX',
 'Clash Royale',
 'Kylie Jenner',
 'Ed Sheeran',
 'Paramount Pictures',
 'ArianaGrandeVevo',
 'PewDiePie',
 'DrakeVEVO',
 'MLG Highlights',
 'justintimberlakeVEVO',
 'AsapSCIENCE',
 'Disneyâ\x80¢Pixar',
 'EminemVEVO',
 'SMTOWN',
 'WWE',
 'jypentertainment',
 'BuzzFeedVideo',
 '5-Minute Crafts',
 'The Late Show with Stephen Colbert']
```

In [38]:

```
#convert it to dictionary with values and its occurences
from wordcloud import WordCloud
from collections import Counter



word_could_dict=Counter(to_word_cloud)
wordcloud = WordCloud(width = 1200, height = 500, background_color='white').gene
rate_from_frequencies(word_could_dict)


plt.figure(figsize=(15,8))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```