MSIN0143: Programming for Business Analytics

Group Coursework

Exploring YouTube's Trending Videos

Word Count: 2167 (Excluding Table of Contents and References)

December 2020

Table of Contents

1. Defining the Problem

Problem

Social media platforms are rapidly shaping the world's trends and culture. Socially Powerful is a UK-based social media agency company dedicated to helping influencers and companies achieve social media presence. Hypothetically, we work as data analysts at Socially Powerful. This project focuses on YouTube, the most acceleratingly popular content channel for influencers and firms.

The aim of this project is, after a thorough examination of YouTube trending video statistics, to be able to offer proper comprehensive advice to creators worldwide as to how minimize the time it takes for a video to get to the trending page and how to maximize their time there.

Possible outcomes of the project:

- Build awareness: tailored video content creation by category.
- Video publishing strategy: find the best time to attract views and become trending.
- Trending time prediction: trending video is a major KPI for a company to assess a successful marketing campaign. We would apply Ordinary Least Squares (OLS) regression to predict how long it took for a given video to be trending using variables such as views, likes, and dislikes.

Dataset

The dataset is taken from kaggle(Sharma, 2020). It should be noted that the dataset is updated daily, and this analysis uses dataset downloaded on 12/12/2020. The study is built based on the following logic: we start by doing a small analysis of what is happening in the world then focus on specific countries known for their popularity on the platforms. We take in account which are the biggest cultural poles as of right now and ended up with the following selection: Japan, UK, US and Russian. Those countries should be representative for the majority of popular content creators and their analysis should help us advice them properly on the matter.

2. Data Preparation

2.1 Importing Libraries

In [1]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"                    # Show all res
ults of execution.
from six.moves import urllib                                       # For data pre
paration.
import matplotlib.pyplot as plt                                    # For data vis
ualization.
plt.rcParams['figure.dpi'] = 300                                   # Set dpi.
import seaborn as sns                                              # For data vis
ualization.
import pandas as pd                                                # For data wra
ngling and exploratory data analysis.
import numpy as np                                                 # For data wra
ngling and exploratory data analysis.
import os                                                          # For data pre
paration.
import tarfile                                                     # For data pre
paration.
import json                                                        # For data wra
ngling.
from wordcloud import WordCloud                                    # For wordclou
d.
from google_trans_new import google_translator                     # For translat
ion.

%autosave 120
```

```
Autosaving every 120 seconds
```

2.2 Loading Data

2.2.1 WorldWide

In [2]:
```
# Loading all the data about trending videos in the world in one Data Frame called df
_ww.

countries = ['BR','CA','IN','KR','DE','US','GB','JP','FR','MX','RU']
#we notice that all the sets begin with the country code. We store it in the 'countri
es' variable
df_ww = pd.DataFrame()
df_1 = pd.DataFrame()
#we create two data frames: one will be our main dataset and the other one we use in
 the loop to store all countries

# We create a loop in order to save all the countries. The way the loop works: it tak
es the main part of the path
#then it takes one by one the country code, adds it to the path and gets the .csv fil
e for each country, adding them
#to our main dataset.
for c in countries:
    path = '/project/YouTube/trending_videos/' + c + '_youtube_trending_data.csv'
    df_1 = pd.read_csv(path, parse_dates=['publishedAt','trending_date'])
    df_1['country'] = c
    df_ww = pd.concat([df_ww, df_1])
```

2.2.2 For Japan, The United Kingdom, USA, and Russia

In [3]:
```
#Creating a sub data sets for Japan, The United Kingdom, The United States of America
and Russia.
df_jp = pd.read_csv('./JP_youtube_trending_data.csv')
df_gb = pd.read_csv('./GB_youtube_trending_data.csv')
df_us = pd.read_csv('./US_youtube_trending_data.csv')
df_ru = pd.read_csv('./RU_youtube_trending_data.csv')
```

In [4]:
```
# Function to import definition of the category id  # Reference :
def load_categoryid(countryname,dataframe):
    lo ="./_category_id.json"
    location= lo[:2] + countryname + lo[2:]
    with open (location) as f:
        categories = json.load(f)["items"]
    cat_dict = {}
    for cat in categories:
        cat_dict[int(cat["id"])] = cat["snippet"]["title"]
    dataframe['category_name'] = dataframe['categoryId'].map(cat_dict)
```

In [5]:
```
load_categoryid("JP",df_jp)
load_categoryid("GB",df_gb)
load_categoryid("RU",df_ru)
load_categoryid("US",df_us)
```

2.3 Data Wrangling

2.3.1 For the WorldWide Data Set

In [6]:
```
# Step 1: Checking the data structure and the type of each variable, number of rows a
nd columns
#Get data information
df_ww.info()

# Get the number of rows and columns
rows = len(df_ww.axes[0])
cols = len(df_ww.axes[1])

# Print the number of rows and columns
print("Number of Rows: " + str(rows))
print("Number of Columns: " + str(cols))
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 259462 entries, 0 to 23995
Data columns (total 17 columns):
 #   Column             Non-Null Count   Dtype              
---  ------             --------------   -----              
 0   video_id           259462 non-null  object             
 1   title              259462 non-null  object             
 2   publishedAt        259462 non-null  datetime64[ns, UTC]
 3   channelId          259462 non-null  object             
 4   channelTitle       259462 non-null  object             
 5   categoryId         259462 non-null  int64              
 6   trending_date      259462 non-null  datetime64[ns, UTC]
 7   tags               259462 non-null  object             
 8   view_count         259462 non-null  int64              
 9   likes              259462 non-null  int64              
 10  dislikes           259462 non-null  int64              
 11  comment_count      259462 non-null  int64              
 12  thumbnail_link     259462 non-null  object             
 13  comments_disabled  259462 non-null  bool               
 14  ratings_disabled   259462 non-null  bool               
 15  description        256436 non-null  object             
 16  country            259462 non-null  object             
dtypes: bool(2), datetime64[ns, UTC](2), int64(5), object(8)
memory usage: 32.2+ MB
Number of Rows: 259462
Number of Columns: 17
```

In [7]:
```
###Data contains the following information:

#video_id = Unique ID allocated to each video uploaded on YouTube
#title = Title of the YouTube video
#publishedAt = Time at which the video was published
#channelId = The unique ID belonging to the channel where the video was posted
#channelTitle = Title of the YouTube channel where the video was posted
#categoryId = Code allocated to a specific category
#trending_date = Date when the channel reached trending
#tags = Label attached to the video for the purpose of identification or to give othe
r information
#likes = Total number of likes of the video at the given date
#dislikes = Total number of dislikes of the video at the given date
#comment_count = Number of comments
#thumbnail_link = Link to the image thumbnail of the video
#comments_disabled = Function that allows to censor the comment section (on or off)
#ratings_disabled = Function that allows to disable the rating option (on or off)
#Description = Description of the video
```

In [8]:
```
# Step 2: Adding an unique ID to the countries for future plots
country = df_ww.country.unique().tolist()
df_ww['country_ID'] = df_ww.country.apply(lambda x: country.index(x))
```

In [9]:
```
# Step 3: Trying to identify the date which the first and the last videos were posted
#adding columns to the dataframe
df_ww['publishing_year'] = ''
df_ww['publishing_month'] = ''
df_ww['publishing_day'] = ''
df_ww['publishing_hour'] = ''
df_ww['trending_year']= ''
df_ww['trending_month']= ''
df_ww['trending_day']= ''
df_ww['trending_hour']= ''

#We store the 'publishedAt' variable in 'Pub_Date' for further manipulation
Pub_Date = df_ww['publishedAt']
#We convert our 'Pub_Date' variable into a datatime index
Pub_Date = pd.to_datetime(Pub_Date)
Pub_Date.head()
#We replace the 'publishedAt' variable with the new variable
df_ww['publishedAt']=Pub_Date

#We split our 'publishedAt' & 'trending_date' into year/month/day/hour
df_ww['publishing_year'] = pd.DatetimeIndex(df_ww['publishedAt']).year
#df_ww['publishing_month'] = pd.DatetimeIndex(df_ww['publishedAt']).month
#df_ww['publishing_day']= pd.DatetimeIndex(df_ww['publishedAt']).dayofweek
df_ww['publishing_hour'] = pd.DatetimeIndex(df_ww['publishedAt']).hour

df_ww['trending_year'] = pd.DatetimeIndex(df_ww['trending_date']).year
#df_ww['trending_month'] = pd.DatetimeIndex(df_ww['trending_date']).month
#df_ww['trending_day']= pd.DatetimeIndex(df_ww['trending_date']).dayofweek
df_ww['trending_hour'] = pd.DatetimeIndex(df_ww['trending_date']).hour

#for the name of the month
df_ww['publishing_month'] = pd.DatetimeIndex(df_ww['publishedAt']).month_name()
df_ww['trending_month'] = pd.DatetimeIndex(df_ww['trending_date']).month_name()

#for the name of the day
df_ww['publishing_day'] = pd.DatetimeIndex(df_ww['publishedAt']).day_name()
df_ww['trending_day'] = pd.DatetimeIndex(df_ww['trending_date']).day_name()

#checking the first date when the video was published in the dataset
first = df_ww['publishedAt'].min()
first_date = f"The first video was published on: {first}."
last = df_ww['publishedAt'].max()
last_date = f"The last video was published on: {last}"
print(first_date)
print(last_date)
```

Out[9]:
```
0   2020-08-11 22:21:49+00:00
1   2020-08-11 15:00:13+00:00
2   2020-08-10 14:59:00+00:00
3   2020-08-11 15:00:09+00:00
4   2020-08-11 20:04:02+00:00
Name: publishedAt, dtype: datetime64[ns, UTC]

The first video was published on: 2020-07-14 22:24:57+00:00.
The last video was published on: 2020-12-12 10:00:08+00:00
```

In [10]:
```
#Step 3: Identifying unneeded columns

#checking if the trending hour is always midnight
#creating a function to check all the unique entries in the 'trending_hour' column
def is_unique(s):
    a = s.to_numpy()
    return (a[0] == a).all()

is_unique(df_ww['trending_hour'])
#it returns true, hence the trending hour is always midnight. We can remove that colu
mn
```

Out[10]:
```
True
```

In [11]:
```
#Step 4: Removing columns which will not be used
df_ww.drop(columns= ['trending_hour', 'thumbnail_link','description'],inplace=True)
```

When doing the general analysis of the trending videos all over the world, we take in account all the videos that reached trending in different locations

2.3.2 For Individual Countries Data Sets

In [12]:
```
# Step 1: Check existence of null entries in each country's dataframe.
print(df_jp.isna().sum()/len(df_jp)) # check for null entries in df_jp, the 'descript
ion' column has 1.1% null values and category name with 0.03% in category name.
print(df_gb.isna().sum()/len(df_gb)) # Same finding for all other countries, only col
umns with null values is the description column and category name.
print(df_us.isna().sum()/len(df_us))
print(df_ru.isna().sum()/len(df_ru))
```

```
video_id             0.000000
title                0.000000
publishedAt          0.000000
channelId            0.000000
channelTitle         0.000000
categoryId           0.000000
trending_date        0.000000
tags                 0.000000
view_count           0.000000
likes                0.000000
dislikes             0.000000
comment_count        0.000000
thumbnail_link       0.000000
comments_disabled    0.000000
ratings_disabled     0.000000
description          0.012198
category_name        0.000374
dtype: float64
video_id             0.000000
title                0.000000
publishedAt          0.000000
channelId            0.000000
channelTitle         0.000000
categoryId           0.000000
trending_date        0.000000
tags                 0.000000
view_count           0.000000
likes                0.000000
dislikes             0.000000
comment_count        0.000000
thumbnail_link       0.000000
comments_disabled    0.000000
ratings_disabled     0.000000
description          0.006496
category_name        0.001122
dtype: float64
video_id             0.000000
title                0.000000
publishedAt          0.000000
channelId            0.000000
channelTitle         0.000000
categoryId           0.000000
trending_date        0.000000
tags                 0.000000
view_count           0.000000
likes                0.000000
dislikes             0.000000
comment_count        0.000000
thumbnail_link       0.000000
comments_disabled    0.000000
ratings_disabled     0.000000
description          0.006216
category_name        0.000000
dtype: float64
video_id             0.000000
title                0.000000
publishedAt          0.000000
channelId            0.000000
channelTitle         0.000000
categoryId           0.000000
trending_date        0.000000
tags                 0.000000
view_count           0.000000
likes                0.000000
dislikes             0.000000
comment_count        0.000000
thumbnail_link       0.000000
```

```
comments_disabled    0.000000
ratings_disabled     0.000000
description          0.019488
category_name        0.018413
dtype: float64
```

In [13]:
```
df1 = df_gb[df_gb["category_name"].isna()].drop_duplicates(subset=['title'], keep='last')
df1
```

Out[13]:

	video_id	title	publishedAt	channelId	channelTitle	categoryId
5764	P5urIeEcuvA	Miley Cyrus quits veganism	2020-09-05T18:17:10Z	UCVRrGAcUc7cblUzOhI1KfFg	Earthling Ed	29
14961	nLW1Acynvb4	RETURNING To YouTube? Our CHARITY FUNDRAISER! ...	2020-10-20T16:59:18Z	UC6kFD5xIFvWyLlytv5pTR1w	DissociaDID	29
15167	nGJGnXkWJ20	It Counts	2020-10-21T21:15:10Z	UCuQ9PbS08dQJNWfwjQE-Fnw	Annie LeBlanc	29
15779	vwSXOPq9u80	Happy Birthday, #TeamTrees!	2020-10-25T14:19:34Z	UCTzBfQF4z2ZdXRbfWazLlrQ	#TeamTrees	29
19135	rUv4U9P-dY0	The Joe Wicks 24 Hour PE Challenge I Part 1	2020-11-12T09:48:49Z	UCoFxbbdY3eOJsjdHH3TSClg	BBC Children in Need	29
19596	2vyWMBjMUFc	The Joe Wicks 24 Hour PE Challenge I Part 5 I ...	2020-11-13T09:33:48Z	UCoFxbbdY3eOJsjdHH3TSClg	BBC Children in Need	29

The column for **description** will be excluded from Analysis, hence we will not drop any null value for this column.On the other hand, the null values in category name is due to the difference in name format for category id 29 in the jason file. Therefore, the null value will be mannually updated as **Shows** for this category according the the dataset provider.

In [14]:
```
df_gb[df_gb['categoryId']==29] = df_gb[df_gb['categoryId']==29].fillna("Shows")
df_us[df_us['categoryId']==29] = df_us[df_us['categoryId']==29].fillna("Shows")
df_jp[df_jp['categoryId']==29] = df_jp[df_jp['categoryId']==29].fillna("Shows")
df_ru[df_ru['categoryId']==29] = df_ru[df_ru['categoryId']==29].fillna("Shows")
```

In [15]:
```
# Step 2: drop irrelevant columns
# Define a function for drop column.
def drop_column(dataframe):
    dataframe = dataframe.drop(columns=['video_id', 'channelId','channelTitle','tags'
,"thumbnail_link","description", "categoryId"],inplace=True)
```

In [16]:
```
drop_column(df_jp)
drop_column(df_gb)
drop_column(df_us)
drop_column(df_ru)
```

In [17]:
```
# Step 3: Drop duplicates.
# Function to remove duplicates.
def drop_duplicate(dataframe):
    dataframe = dataframe.drop_duplicates(subset=['title'], keep='last',inplace=True)
```

In [18]:
```
# Drop duplicated trending videos.
drop_duplicate(df_jp)
drop_duplicate(df_gb)
drop_duplicate(df_us)
drop_duplicate(df_ru)
```

In [19]:
```
print("The dataset after removing duplicates for Japan contains {} videos, for UK {}
 videos, for US {} videos and for Russian {} videos.".format(len(df_jp),len(df_gb),le
n(df_us),len(df_ru)))
#Since the same Youtube video can be trending for multiple times and the dataset cont
ains duplicated trending videos. As a result, we drop duplicated trending videos of t
he same title, and keep the latest one.
```

```
The dataset after removing duplicates for Japan contains 3602 videos, for UK 4863 vid
eos, for US 4290 videos and for Russian 13477 videos.
```

In [20]:
```
# Step 4: Create a variable for the time difference between published time and trendi
ng time.
# Function for creating time_diff.
def time_diff(dataframe):
    dataframe['trending_date'] = (pd.to_datetime(dataframe['trending_date'], format=
'%Y-%m-%dT%H:%M:%SZ'))+ np.timedelta64(24,"h")  # Convert to datetime datatype, trend
ing date is considered at the end of day, thus +24 hours.
    dataframe["publishedAt"] = pd.to_datetime(dataframe['publishedAt'], format='%Y-%m
-%dT%H:%M:%SZ')     # Convert to datetime datatype
    dataframe["time_diff"] = (dataframe.trending_date-dataframe.publishedAt ) / np.ti
medelta64(1, 'h')   # Convert time_diff in hours
```

In [21]:
```
time_diff(df_jp)
time_diff(df_gb)
time_diff(df_us)
time_diff(df_ru)
```

In [22]:
```
# Step 5: Define variable like to dislike ratio and comment rates for subsequent Expl
oratory analysis.
def func(df):
    df['% commented'] =  df['comment_count'] / df['view_count'] * 100
    df["ldratio"] = round(df["likes"] /df["dislikes"],2)
func(df_gb)
func(df_jp)
func(df_ru)
func(df_us)
```

3. Exploratory Analysis

3.1 Worldwide Anlaysis

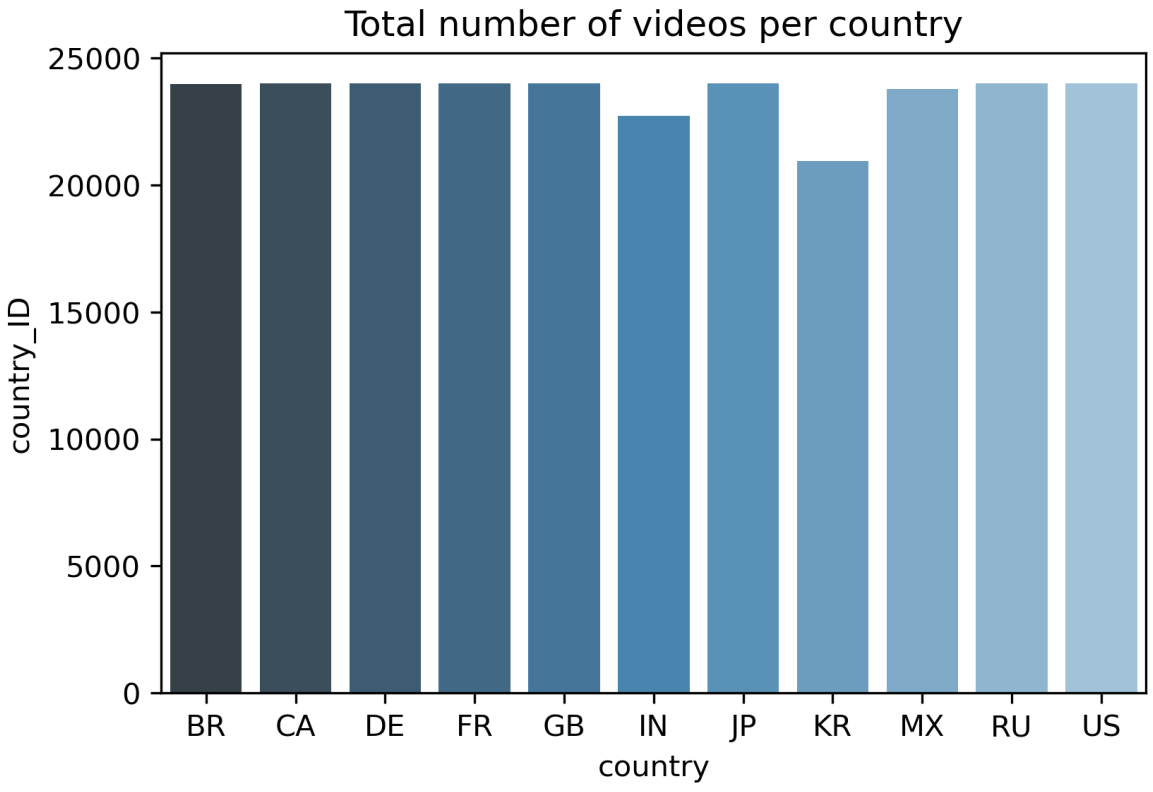
3.1.1 Total number of videos posted per country

In [29]:
```
#Plotting the total number of videos posted per country

#checking statistics of the videos per country worldwide
#checking how many videos we have per country worldwide
vid_count_ww = df_ww[['country_ID', 'country']].groupby('country').count()

#Plotting the total number of views per country worldwide
sns.barplot(x=vid_count_ww.index, y=vid_count_ww.country_ID,palette=("Blues_d")).set_
title('Total number of videos per country')
vid_count_ww
```

Out[29]: Text(0.5, 1.0, 'Total number of videos per country')

Out[29]:

country	country_ID
BR	23994
CA	24000
DE	23999
FR	23996
GB	23998
IN	22730
JP	23997
KR	20955
MX	23800
RU	23996
US	23997



Concerning the number of videos posted that reach the trending page, the US has most success with almost double the average number of videos posted in the rest of the countries considered in this study.

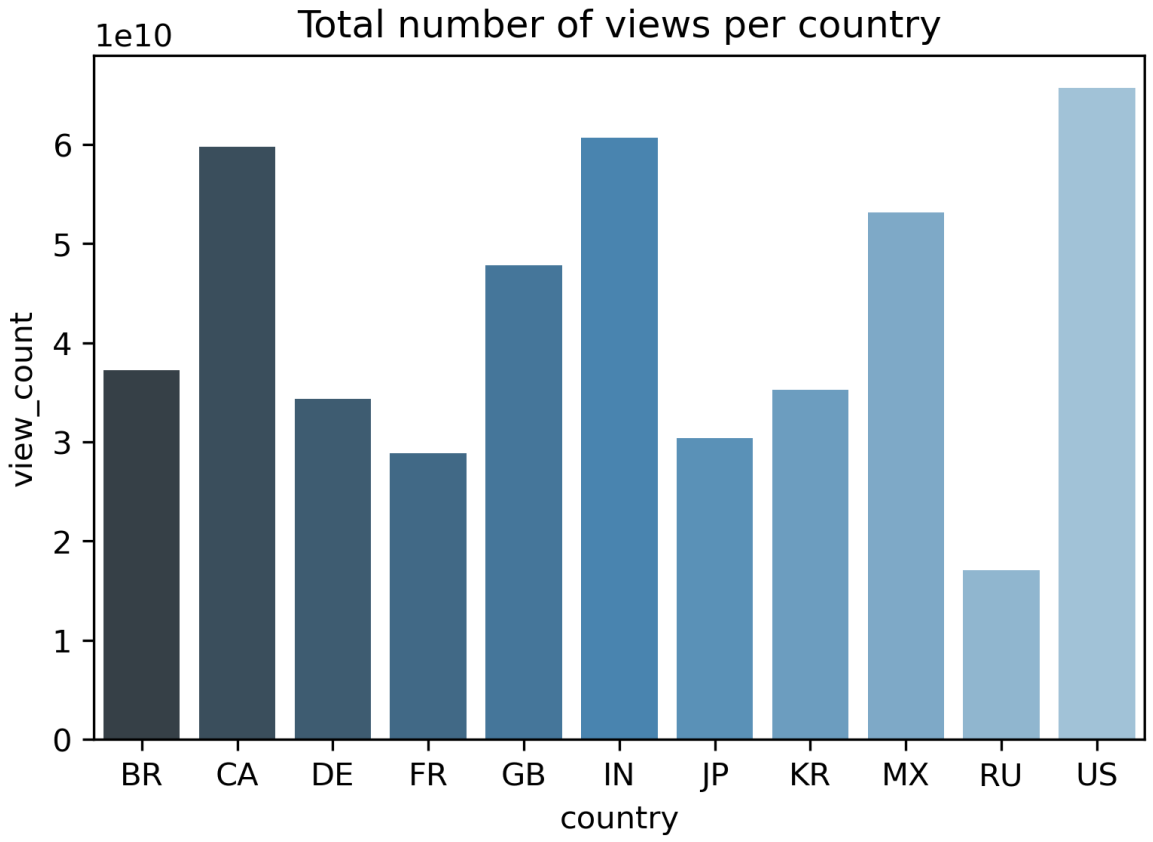
3.1.2 Total number of views per country

In [30]:
```
#Checking statistics of the videos per country worldwide: likes, dislikes, and number of views
stat_count_ww = df_ww[['likes', 'dislikes', 'view_count', 'country']].groupby('country').sum()

#Plotting the total number of views per country
sns.barplot(x=stat_count_ww.index, y=stat_count_ww.view_count,palette=("Blues_d")).set_title('Total number of views per country')
stat_count_ww
```

Out[30]:
```
Text(0.5, 1.0, 'Total number of views per country')
```

Out[30]:

country	likes	dislikes	view_count
BR	3096772326	57260839	37215484540
CA	3704926862	87107762	59774037346
DE	2286193613	63107110	34356627811
FR	2243249159	48379412	28868143072
GB	2980946614	69772934	47838660645
IN	3117077185	269157086	60673984989
JP	1697939006	43240295	30397068740
KR	2064072059	47221966	35261271327
MX	3999711819	87816953	53134870323
RU	1407112409	45282178	17076684148
US	3896341880	93806337	65690183105

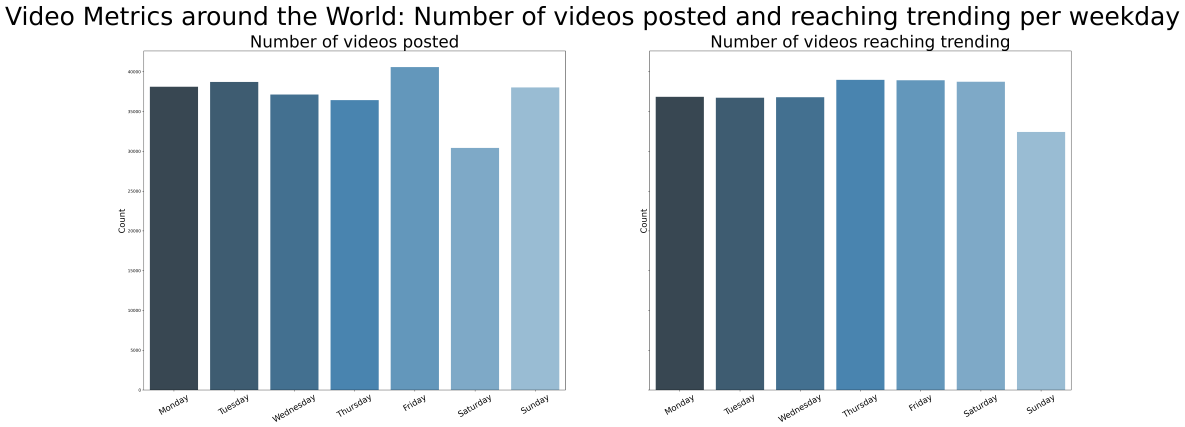


The US is still the leader in terms of views count, just like in terms of videos, but what is interesting to notice is the discrepancy between the other countries. For example, even though Russia and Great Britain have almost the exact same number of videos reaching trending, the activity of Russian people is way lower.

3.1.3 Video Metrics around the world: Most popular days for posting and reaching trending

In [31]:
```
#Plotting the day of the week when videos get posted and reach trending
#checking how much people post each day of the week
day_count_ww = df_ww[['video_id', 'publishing_day']].groupby('publishing_day').count
()

#reordering to start from Monday until Sunday
day_count_ww = day_count_ww.reindex(['Monday','Tuesday','Wednesday','Thursday','Frida
y','Saturday','Sunday'])

#Plotting the total number of videos posted per day of the week in the world
#checking statistics of the videos per country worldwide

fig, axes = plt.subplots(1, 2, figsize=(40, 15), sharey=True)
fig.suptitle('Video Metrics around the World: Number of videos posted and reaching tr
ending per weekday', fontsize = 60)

#Plotting the total number of videos posted per day of the week in the world
plt.figure(figsize = (20,15))
a = sns.barplot(ax=axes[0],x=day_count_ww.index, y=day_count_ww.video_id,palette=("Bl
ues_d"))
axes[0].set_xticklabels(a.get_xticklabels(), rotation=30, fontsize=20)
axes[0].set_title("Number of videos posted", fontsize=40)
axes[0].set_xlabel("")
axes[0].set_ylabel("Count", fontsize=20)

#Plotting the total number of videos reaching trending per day of the week in the wor
ld

#checking the most popular trending days (we assume it is during the weekend)
day_trending_ww = df_ww[['video_id', 'trending_day']].groupby('trending_day').count()

#reordering to start from Monday until Sunday
day_trending_ww = day_trending_ww.reindex(['Monday','Tuesday','Wednesday','Thursday',
'Friday','Saturday','Sunday'])

#Plotting the total number of videos reaching trending per day of the week in the wor
ld
plt.figure(figsize = (20,15))
b = sns.barplot(ax=axes[1], x=day_trending_ww.index, y=day_trending_ww.video_id,palet
te=("Blues_d"))
axes[1].set_xticklabels(a.get_xticklabels(), rotation=30, fontsize=20)
axes[1].set_title("Number of videos reaching trending", fontsize=40)
axes[1].set_xlabel("")
axes[1].set_ylabel("Count", fontsize=20)
```

```
Out[31]: Text(0.5, 0.98, 'Video Metrics around the World: Number of videos posted and reaching
         trending per weekday')

Out[31]: <Figure size 6000x4500 with 0 Axes>

Out[31]: [Text(0, 0, 'Monday'),
          Text(0, 0, 'Tuesday'),
          Text(0, 0, 'Wednesday'),
          Text(0, 0, 'Thursday'),
          Text(0, 0, 'Friday'),
          Text(0, 0, 'Saturday'),
          Text(0, 0, 'Sunday')]

Out[31]: Text(0.5, 1.0, 'Number of videos posted')

Out[31]: Text(0.5, 0, '')

Out[31]: Text(0, 0.5, 'Count')

Out[31]: <Figure size 6000x4500 with 0 Axes>

Out[31]: [Text(0, 0, 'Monday'),
          Text(0, 0, 'Tuesday'),
          Text(0, 0, 'Wednesday'),
          Text(0, 0, 'Thursday'),
          Text(0, 0, 'Friday'),
          Text(0, 0, 'Saturday'),
          Text(0, 0, 'Sunday')]

Out[31]: Text(0.5, 1.0, 'Number of videos reaching trending')

Out[31]: Text(0.5, 0, '')

Out[31]: Text(0, 0.5, 'Count')
```



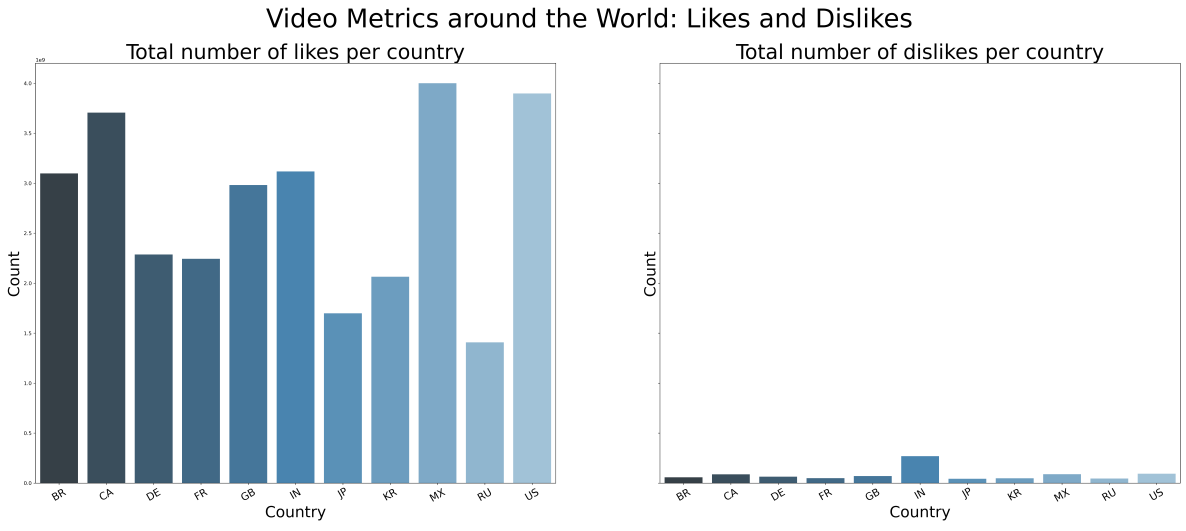

```
<Figure size 6000x4500 with 0 Axes>

<Figure size 6000x4500 with 0 Axes>
```

The trend is quite obvious: content creators post more on Friday as they anticipate the audience to be more active during the weekend when they have more free time which is, as the second graphic shows, indeed the case.

3.1.4 Video Metrics around the world: Likes and Dislikes

In [32]:
```
#Plotting the total number of likes and dislikes around the world for all videos

fig, axes = plt.subplots(1, 2, figsize=(40, 15), sharey=True)
fig.suptitle('Video Metrics around the World: Likes and Dislikes', fontsize = 50)

# Likes Count
c = sns.barplot(ax=axes[0], x=stat_count_ww.index, y=stat_count_ww.likes,palette=("Bl
ues_d"))
axes[0].set_title('Total number of likes per country', fontsize=40)
axes[0].set_xticklabels(c.get_xticklabels(), rotation=30, fontsize=20)
axes[0].set_xlabel("Country", fontsize=30)
axes[0].set_ylabel("Count", fontsize=30)

# Dislikes Count
d = sns.barplot(ax=axes[1],x=stat_count_ww.index, y=stat_count_ww.dislikes,palette=(
"Blues_d"))
axes[1].set_title('Total number of dislikes per country', fontsize=40)
axes[1].set_xticklabels(d.get_xticklabels(), rotation=30, fontsize=20)
axes[1].set_xlabel("Country", fontsize=30)
axes[1].set_ylabel("Count", fontsize=30)
```

```
Out[32]: Text(0.5, 0.98, 'Video Metrics around the World: Likes and Dislikes')

Out[32]: Text(0.5, 1.0, 'Total number of likes per country')

Out[32]: [Text(0, 0, 'BR'),
          Text(0, 0, 'CA'),
          Text(0, 0, 'DE'),
          Text(0, 0, 'FR'),
          Text(0, 0, 'GB'),
          Text(0, 0, 'IN'),
          Text(0, 0, 'JP'),
          Text(0, 0, 'KR'),
          Text(0, 0, 'MX'),
          Text(0, 0, 'RU'),
          Text(0, 0, 'US')]

Out[32]: Text(0.5, 0, 'Country')

Out[32]: Text(0, 0.5, 'Count')

Out[32]: Text(0.5, 1.0, 'Total number of dislikes per country')

Out[32]: [Text(0, 0, 'BR'),
          Text(0, 0, 'CA'),
          Text(0, 0, 'DE'),
          Text(0, 0, 'FR'),
          Text(0, 0, 'GB'),
          Text(0, 0, 'IN'),
          Text(0, 0, 'JP'),
          Text(0, 0, 'KR'),
          Text(0, 0, 'MX'),
          Text(0, 0, 'RU'),
          Text(0, 0, 'US')]

Out[32]: Text(0.5, 0, 'Country')

Out[32]: Text(0, 0.5, 'Count')
```



We first notice the high discrepancy between the number of likes and dislikes in the world. This must be linked to the YouTube recommendation feature. Also, the US, Great Britain, Mexico, India, and Canada are more active. This is probably also linked to the fact that in our list those countries are also amongst the coutries in the world with the largest populations. The only exception we see above is actually India, where the number of dislikes is surprisingly high compared to the rest.

3.1.5 The distribution of time taken for a video to be trending in 4 different countries.

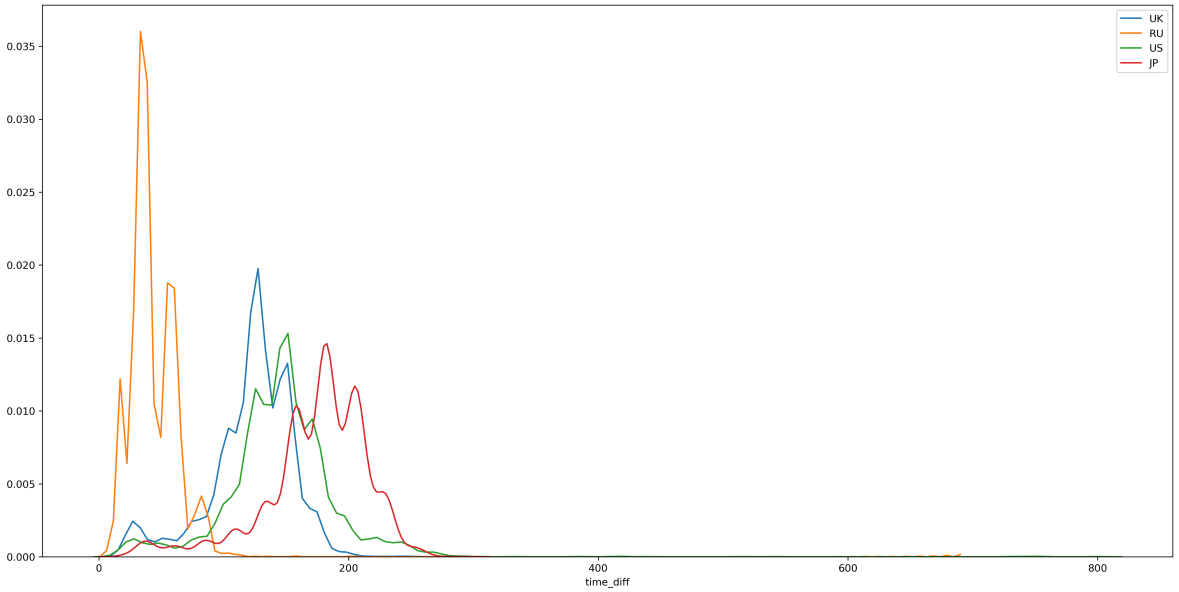
In [33]:
```
# Distplot for the time taken distribution for the following countries.
fig, ax = plt.subplots(figsize=(20,10))
sns.distplot(df_gb["time_diff"],bins = 100,hist = False,label= "UK")
sns.distplot(df_ru["time_diff"],bins = 100,hist = False,label= "RU")
sns.distplot(df_us["time_diff"],bins = 100,hist = False,label = "US")
sns.distplot(df_jp["time_diff"],bins = 100,hist = False,label = "JP")
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7ac192ba60>

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7ac192ba60>

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7ac192ba60>

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7ac192ba60>



Looking at the distribution of time taken for video to be trending, RU likes to select more recently published video to be trending. While in countries such as US and Japan, it takes longer for a published video to be on trending. In the following section, we will focus on each individual country and understanding factors that may impact time_diff

3.1.6 Rankings in the world

Top 10 most viewed videos

In [34]:
```
#Ranking the top 10 most viewed videos worldwide
most_dislikes = df_ww[['title', 'view_count']].groupby('title').sum().sort_values(by=
'view_count', ascending=False).head(10)
most_dislikes
```

Out[34]:

title	view_count
BTS (방탄소년단) 'Dynamite' Official MV	15982262277
BLACKPINK - 'Ice Cream (with Selena Gomez)' M/V	11410871867
BLACKPINK – 'Lovesick Girls' M/V	8966192820
BTS (방탄소년단) 'Life Goes On' Official MV	8900845648
BTS (방탄소년단) 'Dynamite' Official Teaser	3709983373
TWICE I CAN'T STOP ME M/V	3362665283
Apple Event — October 13	3015536639
NCT U 엔시티 유 'Make A Wish (Birthday Song)' MV	2928174974
BTS (방탄소년단) 'Dynamite' Official MV (B-side)	2749272734
Cardi B - WAP feat. Megan Thee Stallion [Official Music Video]	2553919588

In the list above we can see the ten most popular YouTube videos that reached the trending page from July until December. Nine out of ten are music videos and one is a tech related video linked to Apple's most recent event. As BTS has significantly grown in popularity in the past two years, maybe it would be interesting for content creators to do BTS related videos in order to reach the trending page. Reaction, dance videos, and covers are preferred options. In addition, we see also that a tech video is on the list. Tech reviews and gaming videos have tremendously grown in popularity and it seems like this type of content has more potential to capture the audience's attention.

Top 10 most channel with most views

In [35]:
```
#Ranking the top 10 most popular channels worldwide
most_popular_channel = df_ww[['channelTitle','view_count']].groupby('channelTitle').s
um().sort_values(by='view_count', ascending=False).head(10)
most_popular_channel
```

Out[35]:

channelTitle	view_count
Big Hit Labels	36766827398
BLACKPINK	26321079942
SMTOWN	11926678941
JYP Entertainment	11050193617
MrBeast	7522349414
Apple	6228326336
BANGTANTV	6138195002
JustinBieberVEVO	3879999371
T-Series	3879304928
ArianaGrandeVevo	3327269713

Five of the top 10 videos belong to the music category, three are entertainment & gaming related and one is tech related.

Top 10 most liked videos

In [36]:
```
#Ranking the top 10 most liked videos worldwide
most_likes = df_ww[['title', 'likes']].groupby('title').sum().sort_values(by='likes',
ascending=False).head(10)
most_likes
```

Out[36]:

title	likes
BTS (방탄소년단) 'Dynamite' Official MV	1207813075
BLACKPINK - 'Ice Cream (with Selena Gomez)' M/V	864573673
BTS (방탄소년단) 'Life Goes On' Official MV	743567714
BLACKPINK – 'Lovesick Girls' M/V	686947413
BTS (방탄소년단) 'Dynamite' Official MV (B-side)	403915496
BTS (방탄소년단) 'Dynamite' Official Teaser	392225747
BTS (방탄소년단) 'Savage Love' (Laxed – Siren Beat) [BTS Remix] Lyric Video	260001075
BTS (방탄소년단) 'Life Goes On' Official Teaser 1	198345682
[CHOREOGRAPHY] BTS (방탄소년단) 'Dynamite' Dance Practice	184530577
Billie Eilish - Therefore I Am (Official Music Video)	180362264

Top 10 most disliked videos

In [37]:
```
#Ranking the top 10 most disliked videos worldwide
most_dislikes = df_ww[['title', 'dislikes']].groupby('title').sum().sort_values(by='d
islikes', ascending=False).head(10)
most_dislikes
```

Out[37]:

title	dislikes
Sadak 2 I Official Trailer I Sanjay I Pooja I Alia I Aditya I Jisshu I Mahesh Bhatt I 28 Aug	113912935
BLACKPINK - 'Ice Cream (with Selena Gomez)' M/V	63389531
BTS (방탄소년단) 'Dynamite' Official MV	48889246
Cardi B - WAP feat. Megan Thee Stallion [Official Music Video]	12518006
Khaali Peeli I Teaser I Ishaan I Ananya Panday I Maqbool Khan I Coming Soon	11146126
BTS (방탄소년단) 'Dynamite' Official Teaser	9368424
BLACKPINK – 'Lovesick Girls' M/V	7340048
Chocolate - Tony Kakkar ft. Riyaz Aly & Avneet Kaur I Satti Dhillon I Anshul Garg	7144754
BTS (방탄소년단) 'Life Goes On' Official MV	6412683
BTS (방탄소년단) 'Dynamite' Official MV (B-side)	6220643

Six of the ten ranked most liked and most disliked videos around the world are in both categories. This means that if a video is popular people are simply more likely to react. We cannot assess whether there is a correlation in this statistic with the quality of the video or shift in public opinion.

3.1.7 Correlation Heat Map

In [26]:
```
# Creat a mask for the upper triangle
mask_ut=np.triu(np.ones(36).reshape(6,6).astype(np.bool), k=1)
mask_ut
df_ww.drop(columns= ['comments_disabled', 'ratings_disabled','country_ID","publishing
_year","trending_year"],inplace=True)
df1 = df_ww._get_numeric_data()
df1
corrMatrix1 = df1.corr()
plt.rcParams['figure.dpi'] = 100
plt.figure(figsize=(15,15))
sns.set(font_scale=1.8)

sns.heatmap(corrMatrix1, annot=True, fmt='.1g', cmap= "Blues", linewidths=1, linecolo
r='white',annot_kws={"size": 16},
            square=True,mask=mask_ut)

plt.show()
```

Out[26]:
```
array([[False,  True,  True,  True,  True,  True],
       [False, False,  True,  True,  True,  True],
       [False, False, False,  True,  True,  True],
       [False, False, False, False,  True,  True],
       [False, False, False, False, False,  True],
       [False, False, False, False, False, False]])
```

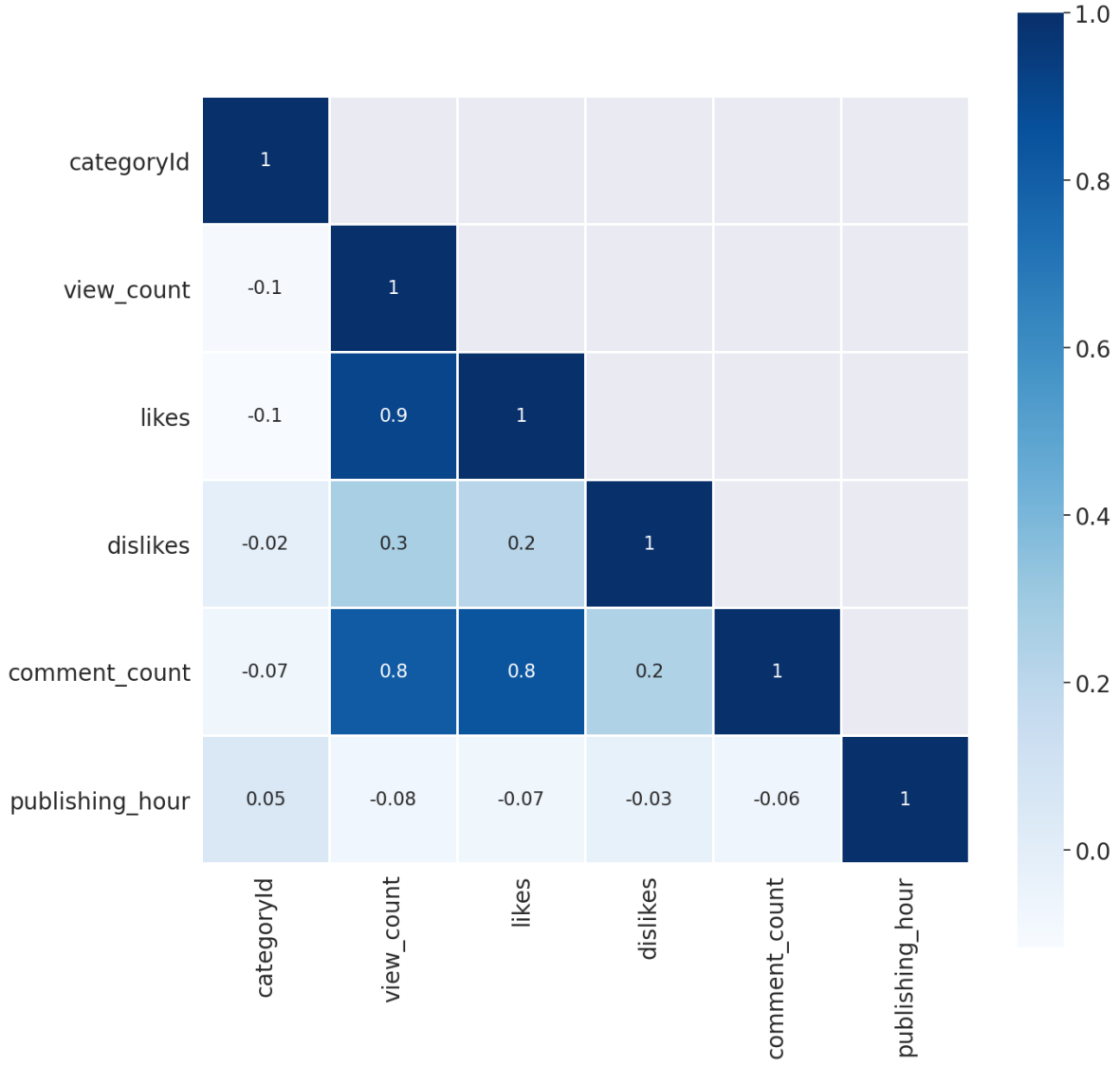
Out[26]:

	categoryId	view_count	likes	dislikes	comment_count	publishing_hour
0	22	263835	85095	487	4500	22
1	10	6000070	714310	15176	31040	15
2	22	2296748	39761	5484	0	14
3	20	300510	46222	242	2748	15
4	23	327235	22059	3972	2751	20
...	...	...	...	...	...	...
23991	10	25017	126	11	0	16
23992	1	394900	7182	353	1819	18
23993	22	213254	18050	426	1456	17
23994	25	160116	2083	202	643	12
23995	25	29410	4064	94	270	6

259462 rows × 6 columns

Out[26]:
```
<Figure size 1500x1500 with 0 Axes>
```

Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8d7d876340>
```



In [29]:
```
#We observe that the number of views, the likes, and the number of comments are stron
ly correlated to each other. Naturally, if a video gets many views it would
#very likely determine reactions from people. Since most videos which get to trending
are because their content is appreciated by the vast audience, we would expect
#them to get a big number of likes
#The rest of the relations between variables aren't as significant
```

3.2 Japan JP

3.2.1 Count by Categories

In [25]:
```
# Plot count of categories for Japan.
plt.figure(figsize = (25,10))
g = sns.countplot('category_name', data=df_jp, palette="Blues_d",order = df_jp['categ
ory_name'].value_counts().index)
g.set_xticklabels(g.get_xticklabels(), rotation=30, fontsize=15)
g.set_title("Count of Video Categories ", fontsize=20)
g.set_xlabel("")
g.set_ylabel("Count", fontsize=15)
```

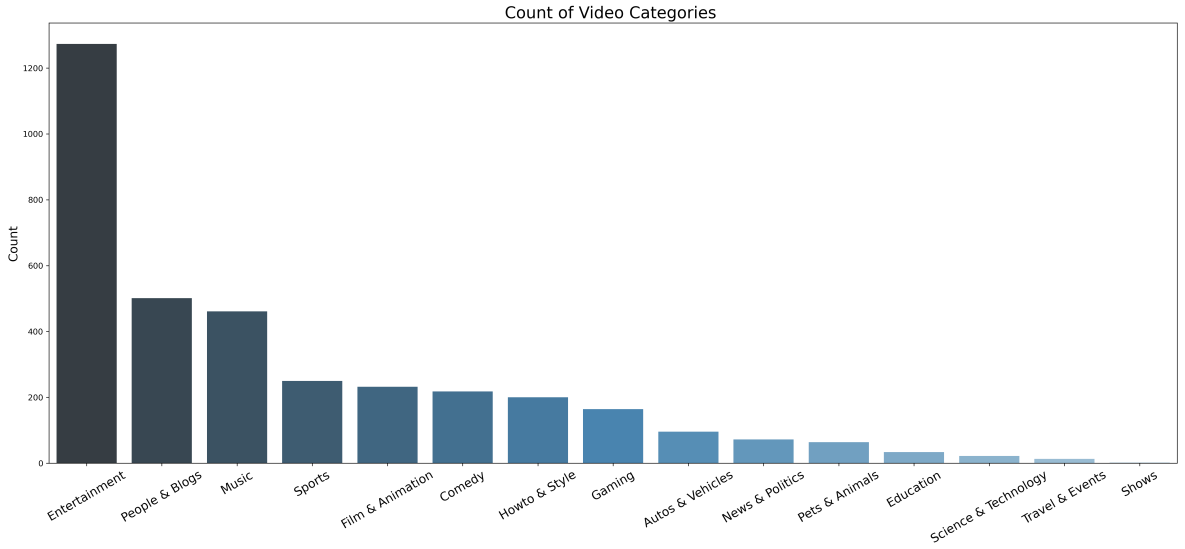
Out[25]:
```
<Figure size 7500x3000 with 0 Axes>
```

Out[25]:
```
[Text(0, 0, 'Entertainment'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'Music'),
 Text(0, 0, 'Sports'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Comedy'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'News & Politics'),
 Text(0, 0, 'Pets & Animals'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Shows')]
```

Out[25]:
```
Text(0.5, 1.0, 'Count of Video Categories ')
```

Out[25]:
```
Text(0.5, 0, '')
```

Out[25]:
```
Text(0, 0.5, 'Count')
```

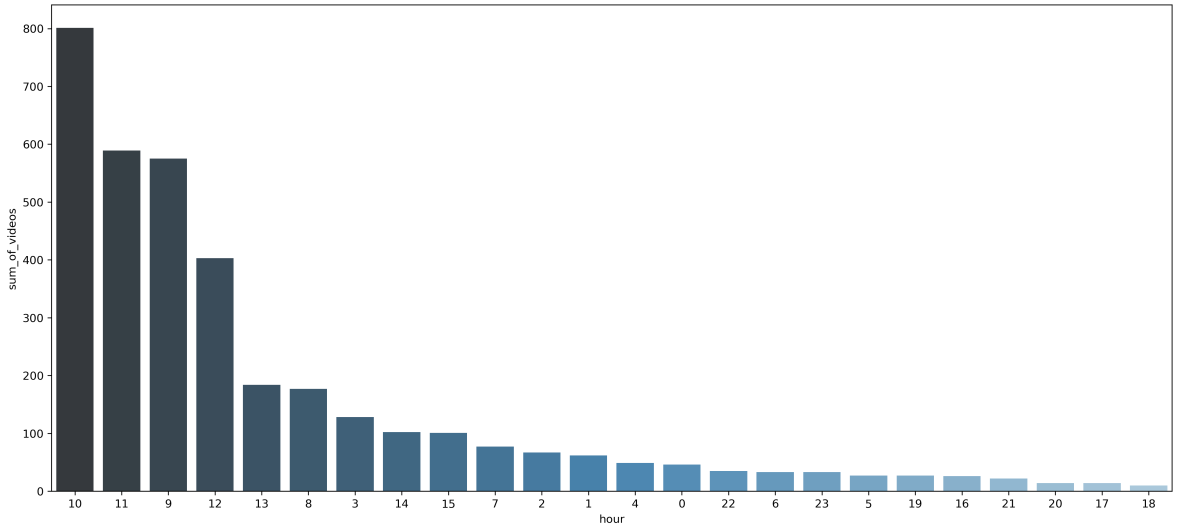


In [ ]:
```
#Entertainment videos account for the majority of the Japanese trending Youtube video
s, followed by People & Blogs.
```

3.2.2 Publishing Time

In [23]:
```
# Plot count of hourly trending videos.
jp_hour = df_jp["publishedAt"].dt.hour.value_counts().reset_index().rename(columns =
{"index": "hour", "publishedAt": "sum_of_videos"})
fig, ax = plt.subplots(figsize=(18,8))
sns.barplot(x="hour", y="sum_of_videos", data = jp_hour,palette=("Blues_d"),
            order=jp_hour.sort_values("sum_of_videos", ascending=False).hour)
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3f07912820>



In [ ]:
```
#Trending videos in Japan are mostly published at 10 a.m.
```

3.2.3 Publishing Day

In [83]:
```
week_plot_jp = df_jp['publishedAt'].apply(lambda x: x.weekday()).value_counts().reset
_index().rename(columns = {"index": "weekday", "publishedAt": "sum_of_videos"})
week_plot_jp["weekday"] = week_plot_jp["weekday"].replace({0: "Mon",1:"Tue",2:"Wed",3
:"Thu",4:"Fri",5:"Sat",6:"Sun"})
week_plot_jp
fig, ax = plt.subplots(figsize=(15,5))
sns.barplot(x="weekday", y="sum_of_videos", data = week_plot_jp,palette=("Blues_d"))
plt.title("Publication day before trending ")
plt.xlabel("")
```

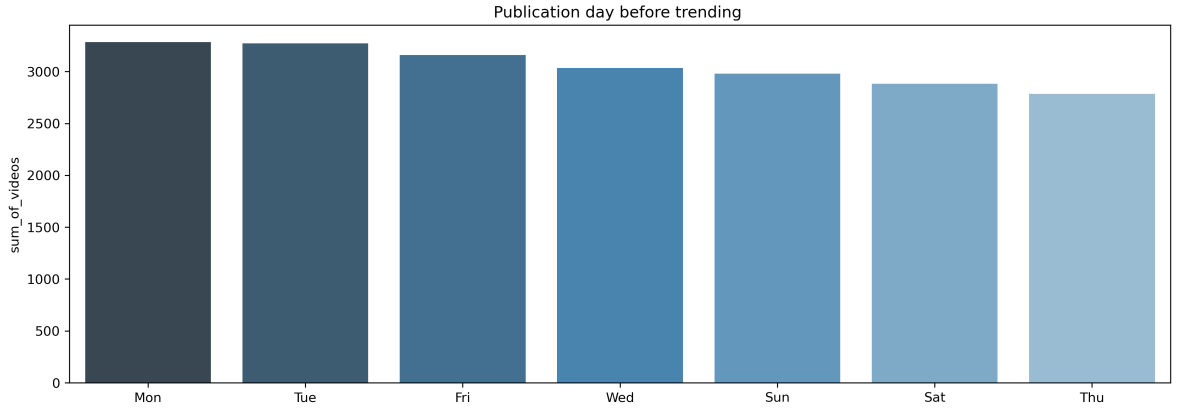
Out[83]:

	weekday	sum_of_videos
0	Mon	3283
1	Tue	3273
2	Fri	3159
3	Wed	3034
4	Sun	2980
5	Sat	2882
6	Thu	2786

Out[83]: `<matplotlib.axes._subplots.AxesSubplot at 0x7fd51b956130>`

Out[83]: `Text(0.5, 1.0, 'Publication day before trending ')`

Out[83]: `Text(0.5, 0, '')`



In [ ]:
```
#Mondays and Tuesdays seem to be the favoured days for posting videos that reach tren
ding in Japan.
```

3.2.4 Trending Day

In [85]:
```
week_plot_jp = df_jp['trending_date'].apply(lambda x: x.weekday()).value_counts().res
et_index().rename(columns = {"index": "weekday", "trending_date": "sum_of_videos"})
week_plot_jp["weekday"] = week_plot_jp["weekday"].replace({0: "Mon",1:"Tue",2:"Wed",3
:"Thu",4:"Fri",5:"Sat",6:"Sun"})
week_plot_jp
fig, ax = plt.subplots(figsize=(15,5))
sns.barplot(x="weekday", y="sum_of_videos", data = week_plot_jp,palette=("Blues_d"))
plt.xlabel("")
plt.ylabel("Sum of videos")
plt.title("Day become trending")
```

Out[85]:

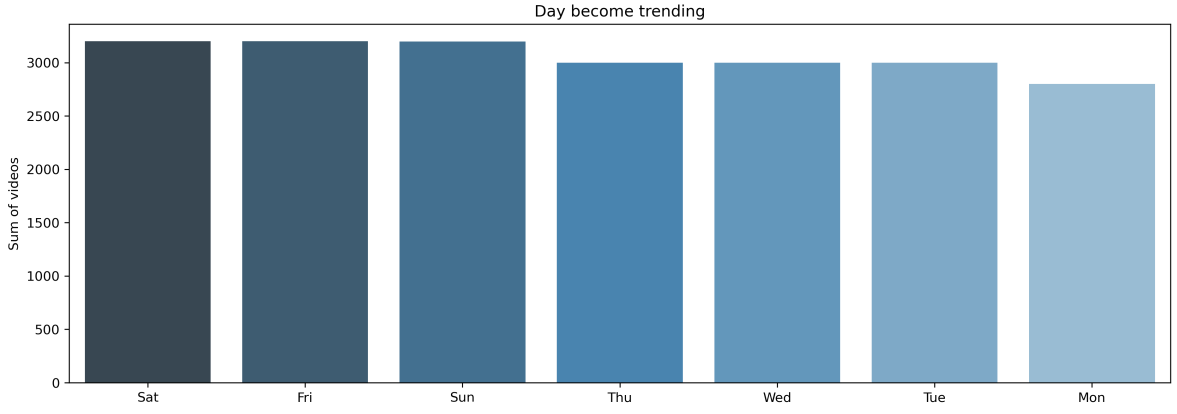
	weekday	sum_of_videos
0	Sat	3200
1	Fri	3200
2	Sun	3197
3	Thu	3000
4	Wed	3000
5	Tue	3000
6	Mon	2800

Out[85]: `<matplotlib.axes._subplots.AxesSubplot at 0x7fd51c8ab9d0>`

Out[85]: `Text(0.5, 0, '')`

Out[85]: `Text(0, 0.5, 'Sum of videos')`

Out[85]: `Text(0.5, 1.0, 'Day become trending')`



In [ ]:
```
#Saturdays and Fridays seem to be the most common days for videos to reach trending.
```

3.2.5 Likes to Dislikes Ratio per Category

In addition to the count of video categories, we have designed a few other measurements to gauge the popularity of a trending YouTube video's category. The first one is the ratio of likes to dislikes by category.

In [89]:
```
jp_ldratio =df_jp.groupby(['category_name'])['ldratio'].mean().reset_index()
jp_ldratio
plt.figure(figsize = (20,10))
g=sns.barplot(x="category_name", y="ldratio", data=jp_ldratio, palette="Blues_d",
           order=jp_ldratio.sort_values("ldratio", ascending=False).category_name)
g.set_xticklabels(g.get_xticklabels(), rotation=30, fontsize=11)
plt.ylabel("ldratio")
```

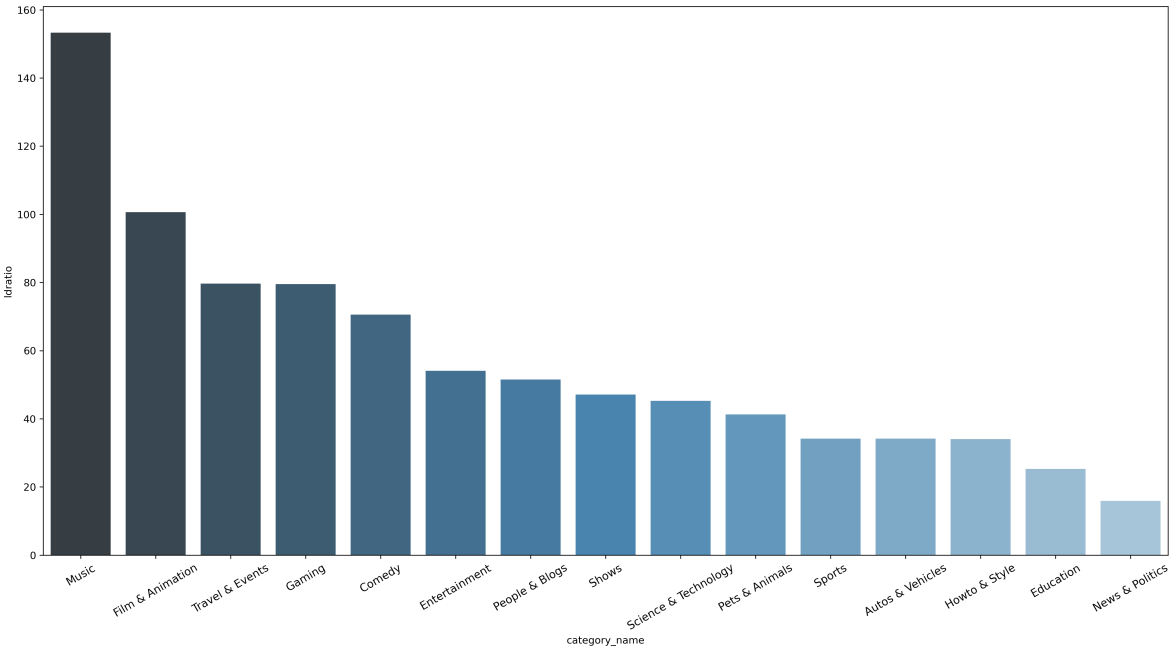
Out[89]:

	category_name	ldratio
0	Autos & Vehicles	34.147768
1	Comedy	70.580032
2	Education	25.289950
3	Entertainment	54.048406
4	Film & Animation	100.651774
5	Gaming	79.487768
6	Howto & Style	34.012200
7	Music	153.321151
8	News & Politics	15.892794
9	People & Blogs	51.523308
10	Pets & Animals	41.298029
11	Science & Technology	45.245573
12	Shows	47.076000
13	Sports	34.199169
14	Travel & Events	79.678354

Out[89]:
```
<Figure size 6000x3000 with 0 Axes>
```

Out[89]:
```
[Text(0, 0, 'Music'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'Comedy'),
 Text(0, 0, 'Entertainment'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'Shows'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'Pets & Animals'),
 Text(0, 0, 'Sports'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'News & Politics')]
```

Out[89]:
```
Text(0, 0.5, 'ldratio')
```



In [25]:
```
#Music has the highest ratio of likes to dislikes, News & politics the lowest, which
 implies music videos are commonly appreciated,
#while opinions on news and politics are usually very mixed.
```

3.2.6 Comment Ratio

Another mesure of popularity amongst trending Youtube is comment rates.

In [16]:
```
plt.figure(figsize = (18,10))

# Plot comment rate distribution
g2= sns.lvplot(y='% commented', x='category_name', data=df_jp)
g2.set_xticklabels(g2.get_xticklabels(),rotation=30)
g2.set_title("Distribution of comment rate", fontsize=20)
g2.set_xlabel("Category Names", fontsize=15)
g2.set_ylabel("Comment Rate", fontsize=15)
# Comment rate distribution
functions=['count','mean','max','min','var']
var = df_jp.groupby(['category_name'])
var['% commented'].agg(functions)
plt.annotate("Highest", xy = (2,0.3), xytext=(2.2,0.6), arrowprops={'color': 'red'})
plt.annotate("Lowest", xy = (0.95,0.15), xytext=(1,0.35), arrowprops={'color': 'red'
})

# 1. "Music" arouses the hottest discussion, "Autos and Vehicles" the coldest.
```

Out[16]: <Figure size 5400x3000 with 0 Axes>

```
/opt/anaconda/envs/Python3/lib/python3.8/site-packages/seaborn/categorical.py:2613: U
serWarning: The `lvplot` function has been renamed to `boxenplot`. The original name
will be removed in a future release. Please update your code.
  warnings.warn(msg)
```

Out[16]:
```
[Text(0, 0, 'Entertainment'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Music'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'Comedy'),
 Text(0, 0, 'News & Politics'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'Sports'),
 Text(0, 0, 'Pets & Animals'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Shows')]
```

Out[16]: Text(0.5, 1.0, 'Distribution of comment rate')

Out[16]: Text(0.5, 0, 'Category Names')

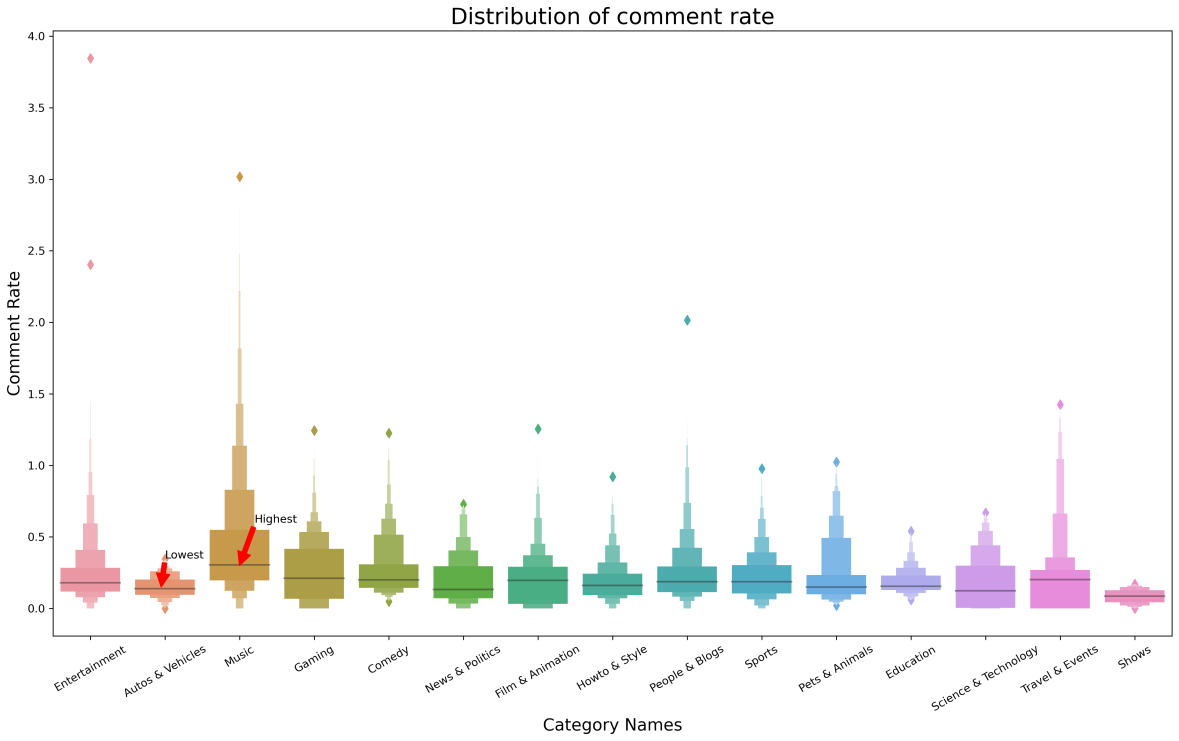
Out[16]: Text(0, 0.5, 'Comment Rate')

Out[16]:

category_name	count	mean	max	min	var
Autos & Vehicles	96	0.150293	0.344493	0.000000	0.005969
Comedy	218	0.266117	1.224845	0.048107	0.038550
Education	34	0.188429	0.538809	0.060020	0.009679
Entertainment	1273	0.239861	3.845743	0.000000	0.056506
Film & Animation	232	0.202775	1.255236	0.000000	0.035332
Gaming	164	0.255494	1.244685	0.000000	0.052258
Howto & Style	200	0.186587	0.921470	0.000000	0.020159
Music	461	0.434097	3.018907	0.000000	0.166151
News & Politics	72	0.201210	0.727505	0.000000	0.031729
People & Blogs	501	0.239673	2.014796	0.000000	0.044786
Pets & Animals	64	0.228623	1.023207	0.021232	0.048451
Science & Technology	22	0.186081	0.668700	0.000000	0.041785
Shows	2	0.084959	0.169918	0.000000	0.014436
Sports	250	0.222322	0.976956	0.000000	0.028184
Travel & Events	13	0.259028	1.425758	0.000000	0.140605

Out[16]: Text(2.2, 0.6, 'Highest')

Out[16]: Text(1, 0.35, 'Lowest')



In [ ]:
```
#The Music category tends to have most comments, the Autos and Vehicles category having the least.
```

3.2.7 Buzzwords Analysis

In [ ]:
```
#The following section contains wordclouds of the titles in the top 4 video categories according to category counts.
```

In [91]:
```
# Title is translated to gain better understanding of the trending video.
translator = google_translator()
translate_text_music = translator.translate(df_jp[df_jp.category_name=="Music"].title)
translate_text_entertainment = translator.translate(df_jp[df_jp.category_name=="Entertainment"].title)
translate_text_ppl = translator.translate(df_jp[df_jp.category_name=="People & Blogs"].title)
translate_text_sports = translator.translate(df_jp[df_jp.category_name=="Sports"].title)
```

In [92]:
```
# Word Cloud for Trending YouTube Video in the top 4 categories
wordcloud1 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(translate_text_music))


wordcloud2 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(translate_text_entertainment))

wordcloud3 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(translate_text_ppl))

wordcloud4 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(translate_text_sports))


# plot the WordCloud image
plt.figure(figsize = (10,10), facecolor = None)
plt.subplot(2, 2, 1)
plt.imshow(wordcloud1)
plt.axis("off")
plt.title("Music")
plt.subplot(2, 2, 2)
plt.imshow(wordcloud2)
plt.axis("off")
plt.title("Entertainment")
plt.subplot(2, 2, 3)
plt.imshow(wordcloud3)
plt.axis("off")
plt.title("People & Blogs")
plt.subplot(2, 2, 4)
plt.imshow(wordcloud4)
plt.axis("off")
plt.title("Sports")
plt.tight_layout(pad = 2)

plt.show()
```

Out[92]: <Figure size 3000x3000 with 0 Axes>

Out[92]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd51c895460>

Out[92]: <matplotlib.image.AxesImage at 0x7fd51a0cf130>

Out[92]: (-0.5, 799.5, 799.5, -0.5)

Out[92]: Text(0.5, 1.0, 'Music')

Out[92]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd51a0c6af0>

Out[92]: <matplotlib.image.AxesImage at 0x7fd51a0c6ca0>

Out[92]: (-0.5, 799.5, 799.5, -0.5)

Out[92]: Text(0.5, 1.0, 'Entertainment')

Out[92]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd51bc5c100>

Out[92]: <matplotlib.image.AxesImage at 0x7fd51bc5ca30>

Out[92]: (-0.5, 799.5, 799.5, -0.5)

Out[92]: Text(0.5, 1.0, 'People & Blogs')

Out[92]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd51a2e0dc0>

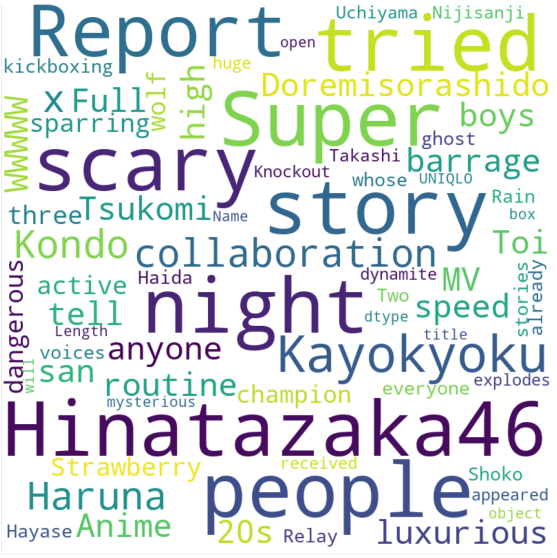
Out[92]: <matplotlib.image.AxesImage at 0x7fd51a2e0d60>

Out[92]: (-0.5, 799.5, 799.5, -0.5)

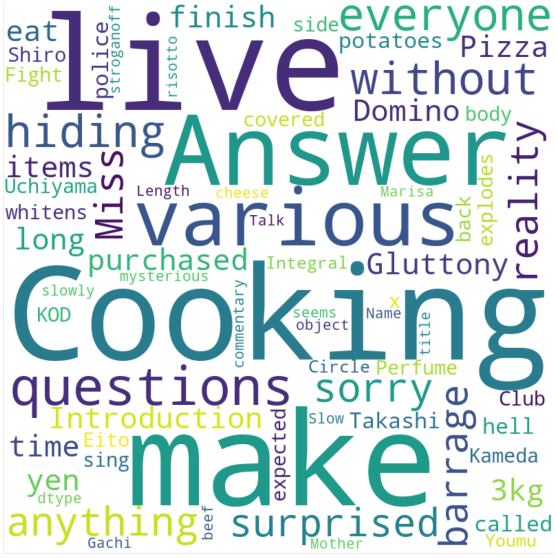
Out[92]: Text(0.5, 1.0, 'Sports')

Music

Entertainment



People & Blogs



Sports

Music

In addition to the descriptive words, such as 'official' and 'video', most often titles include the words X and Kenshi Yonezu. X is a famous Japanese rock band (Wikipedia, 2020a), and Kenshi Yonezu is a popular Japanese male singer (Wikipedia, 2020b).

Entertainment

In addition to the descriptive verbs, such as 'will' and 'make', most trending titles include words like "Kayokyoku" and "takoyaki". Kayokyoku is a music genre with a mix of Western and Japanese local musical scales (Wikipedia, 2020c). Takoyaki is also known as "octopus balls", a ball-shaped snack made of a wheat flour-based batter and cooked in a special molded pan.(Wikipedia, 2020d) Both are very common popular topics in Japan.

People & Blogs

The wordcloud is indicative of Japanese preferences for "cooking" and "live show".

Sports

1. Baseball is outstandingly popular in Japan. The main reason for the frequency of 'Kazushige Nagashima' word group because this person is a sports commentator and former professional baseball player (Wikipedia, 2020e). He is also the son of the Japanese baseball legend - Shigeo Nagashima.
2. Given the diverse eco-system in Japan, it is geographically ideal for fishing as the country is surrounded by water and has an abundance of different fish species. Therefore, 'fishing' shows up in the titles of trending Sports videos in Japan.

3.3 UK GB

3.3.1 Count by Categories

In [39]:
```
category_plot = df_gb["category_name"].value_counts().reset_index().rename(columns =
{"index": "category", "category_name": "sum_of_videos"})
category_plot
fig, ax = plt.subplots(figsize=(25,10))
g = sns.barplot(x="category", y="sum_of_videos", data = category_plot,palette=("Blues
_d"))
g.set_xticklabels(g.get_xticklabels(), rotation=30, fontsize=15)
g.set_title("Count of Video Categories ", fontsize=20)
g.set_xlabel("")
g.set_ylabel("Count", fontsize=15)
```

Out[39]:

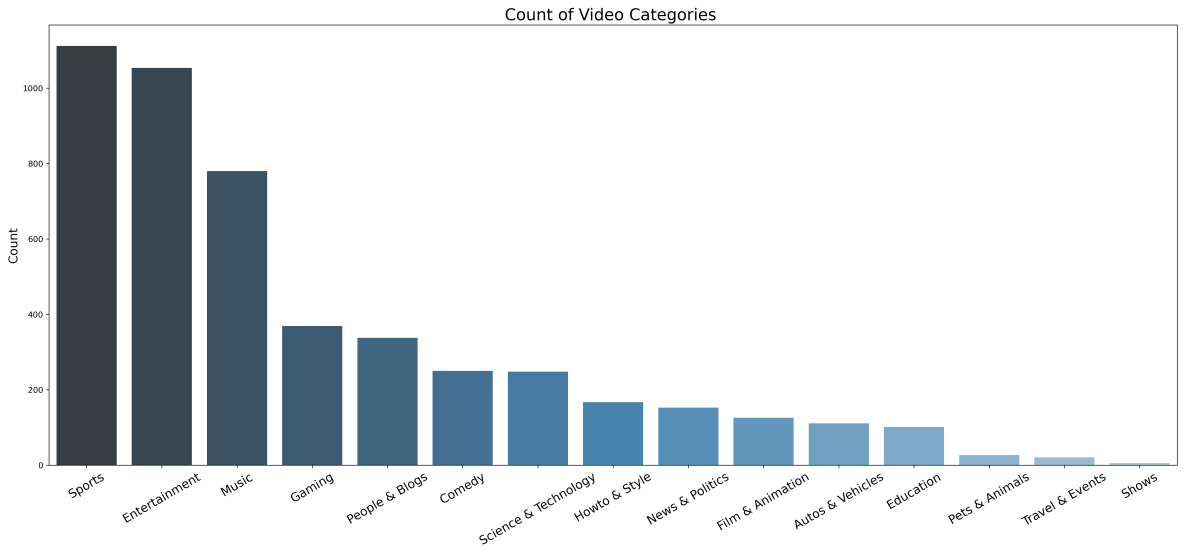
	category	sum_of_videos
0	Sports	1112
1	Entertainment	1054
2	Music	780
3	Gaming	369
4	People & Blogs	338
5	Comedy	250
6	Science & Technology	248
7	Howto & Style	167
8	News & Politics	153
9	Film & Animation	126
10	Autos & Vehicles	111
11	Education	101
12	Pets & Animals	27
13	Travel & Events	21
14	Shows	6

Out[39]:
```
[Text(0, 0, 'Sports'),
 Text(0, 0, 'Entertainment'),
 Text(0, 0, 'Music'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'Comedy'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'News & Politics'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Pets & Animals'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Shows')]
```

Out[39]:
```
Text(0.5, 1.0, 'Count of Video Categories ')
```

Out[39]:
```
Text(0.5, 0, '')
```

Out[39]:
```
Text(0, 0.5, 'Count')
```



In [40]:
```
df1=df_gb[df_gb["category_name"]=="Sports"]
df1["publishedAt"].value_counts()
```

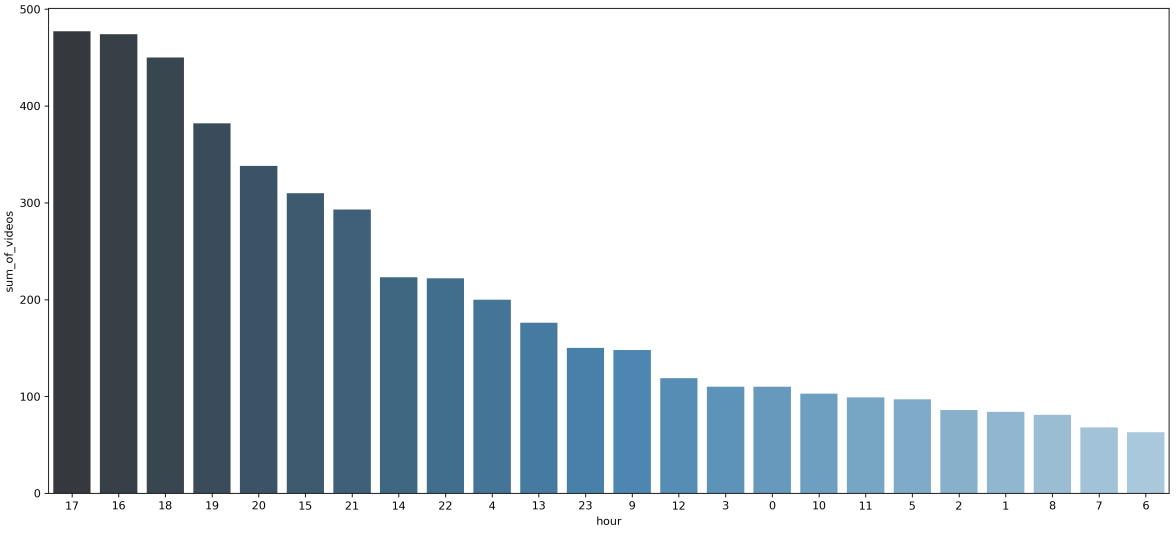
Out[40]:
```
2020-10-03 16:44:37    3
2020-11-18 17:31:14    2
2020-11-03 22:19:08    2
2020-10-23 20:16:34    2
2020-10-21 17:00:10    2
                      ..
2020-10-11 17:00:21    1
2020-11-11 17:00:22    1
2020-11-01 01:39:33    1
2020-09-16 17:08:06    1
2020-09-05 18:28:21    1
Name: publishedAt, Length: 1085, dtype: int64
```

In [26]:
```
#The number of trending videos is dominated by Sport, Entertainment and Music.
#Sport is the most popular category outperforming Entertainment and Music, which are
 in general are the most trending categories.
#After looking into the dates for the sports videos, we conclude they coincide with t
he Serie A TIM match days (Football Italia, 2020).
#This shows that English people have a high interest in soccer.
```

3.3.2 Publishing Time

In [41]:
```
gb_hour = df_gb["publishedAt"].dt.hour.value_counts().reset_index().rename(columns =
{"index": "hour", "publishedAt": "sum_of_videos"})
fig, ax = plt.subplots(figsize=(18,8))
sns.barplot(x="hour", y="sum_of_videos", data = gb_hour,palette=("Blues_d"),
            order=gb_hour.sort_values("sum_of_videos", ascending=False).hour)
```

Out[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f90928425b0>
```



In [27]:
```
#Most trending Videos are published from 4pm to 6pm.
#Therefore, releasing video around this time can increase its chance of reaching tren
ding.
#This could be because the general off-time for employees starts at 5 pm. (Wellbeing
 People,2020).
```

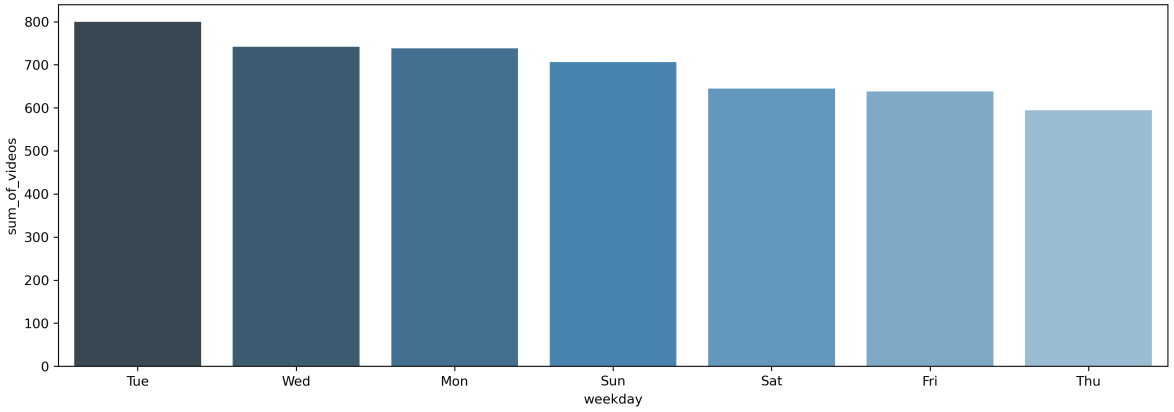
3.3.3 Publishing Day

In [23]:
```
week_plot = df_gb['publishedAt'].apply(lambda x: x.weekday()).value_counts().reset_in
dex().rename(columns = {"index": "weekday", "publishedAt": "sum_of_videos"})
week_plot["weekday"] = week_plot["weekday"].replace({0: "Mon",1:"Tue",2:"Wed",3:"Thu"
,4:"Fri",5:"Sat",6:"Sun"})
week_plot
fig, ax = plt.subplots(figsize=(15,5))
sns.barplot(x="weekday", y="sum_of_videos", data = week_plot,palette=("Blues_d"))
```

Out[23]:

	weekday	sum_of_videos
0	Tue	800
1	Wed	742
2	Mon	738
3	Sun	706
4	Sat	645
5	Fri	638
6	Thu	594

Out[23]: `<matplotlib.axes._subplots.AxesSubplot at 0x7fce6ab0f580>`



In [28]:
```
#Most trending videos are published on Tuesdays and Wednesdays
```

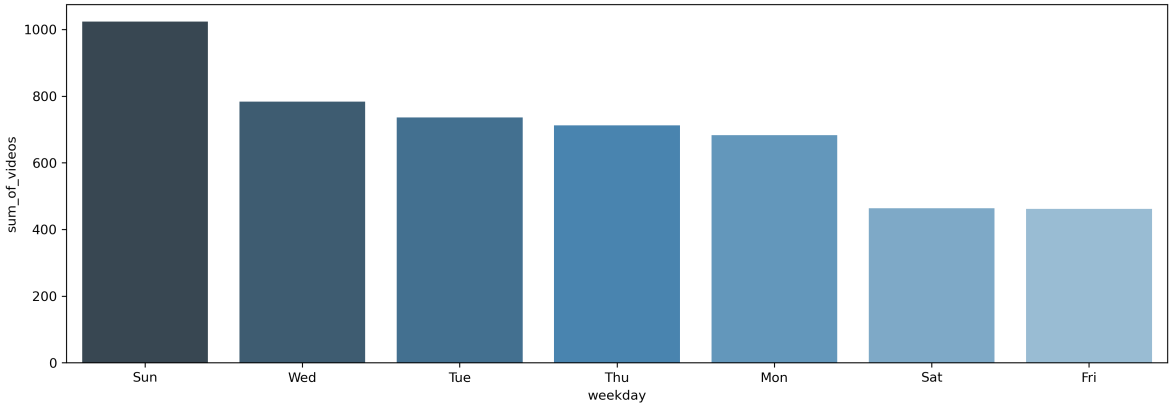
3.3.4 Trending Day

In [42]:
```
week_plot = df_gb['trending_date'].apply(lambda x: x.weekday()).value_counts().reset_
index().rename(columns = {"index": "weekday", "trending_date": "sum_of_videos"})
week_plot["weekday"] = week_plot["weekday"].replace({0: "Mon",1:"Tue",2:"Wed",3:"Thu"
,4:"Fri",5:"Sat",6:"Sun"})
week_plot
fig, ax = plt.subplots(figsize=(15,5))
sns.barplot(x="weekday", y="sum_of_videos", data = week_plot,palette=("Blues_d"))
```

Out[42]:

	weekday	sum_of_videos
0	Sun	1024
1	Wed	783
2	Tue	736
3	Thu	712
4	Mon	683
5	Sat	463
6	Fri	462

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x7f909201f730>



In [ ]:
```
#Most videos reach trending on Sunday.
```

In [43]:
```
print("The average time taken for a newly published video to be on trending is {} day
s.".format(round(df_gb["time_diff"].mean()/24,2)))
```

```
The average time taken for a newly published video to be on trending is 5.11 days.
```

In [29]:
```
#While most trending videos are published on Tuesdays and Wednesdays, reach trending
 on Sunday.
#This may be because it takes on average longer for a video to reach trending in the
 UK (5.11 days) in comparison to countries.
```

3.3.5 Likes to Dislikes Ratio per Category

In [44]:
```
gb_ldratio =df_gb.groupby(['category_name'])['ldratio'].mean().reset_index()
gb_ldratio
plt.figure(figsize = (20,10))
g=sns.barplot(x="category_name", y="ldratio", data=gb_ldratio, palette="Blues_d",
           order=gb_ldratio.sort_values("ldratio", ascending=False).category_name)
g.set_xticklabels(g.get_xticklabels(), rotation=30, fontsize=11)
plt.ylabel("ldratio")
```

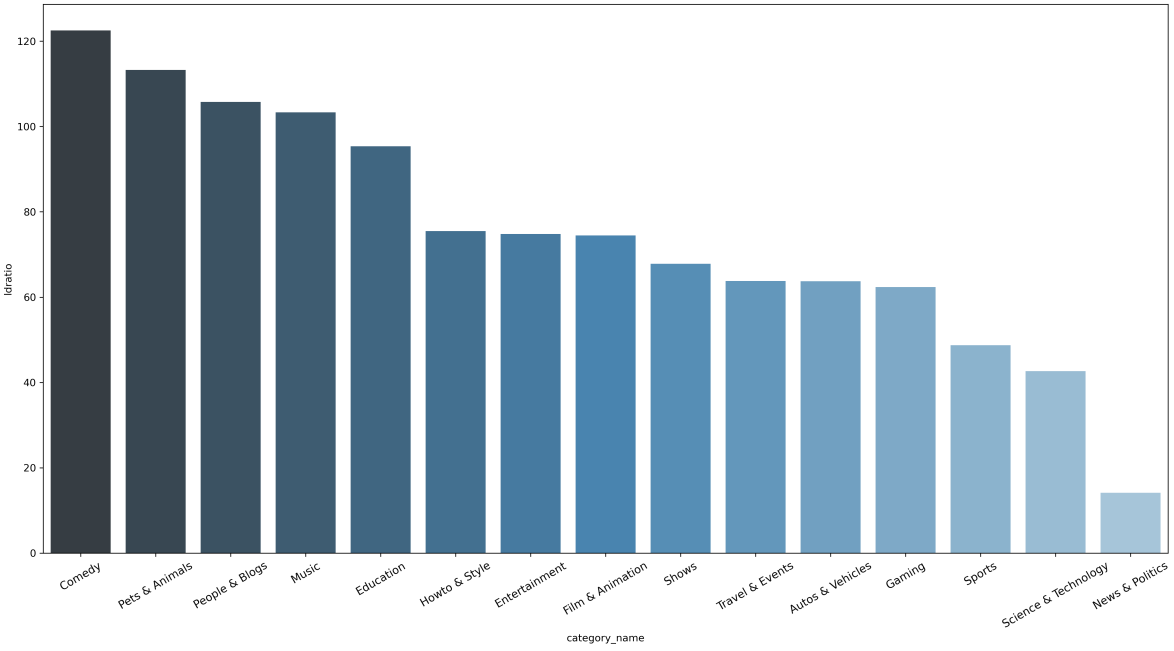
Out[44]:

	category_name	ldratio
0	Autos & Vehicles	63.741376
1	Comedy	122.546280
2	Education	95.350000
3	Entertainment	74.815281
4	Film & Animation	74.437302
5	Gaming	62.344575
6	Howto & Style	75.508503
7	Music	103.311082
8	News & Politics	14.149276
9	People & Blogs	105.738667
10	Pets & Animals	113.271852
11	Science & Technology	42.660041
12	Shows	67.796667
13	Sports	48.746664
14	Travel & Events	63.786667

Out[44]:
```
<Figure size 6000x3000 with 0 Axes>
```

Out[44]:
```
[Text(0, 0, 'Comedy'),
 Text(0, 0, 'Pets & Animals'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'Music'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'Entertainment'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Shows'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'Sports'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'News & Politics')]
```

Out[44]:
```
Text(0, 0.5, 'ldratio')
```



In [ ]:
```
#When considering people's category preference Comedy, Pets & Animal and People & Blo
gs are the top three.
#This means that categories such as entertainment and music, may have the highest num
ber in video trending, but they have higher number of dislikes.
#This means that even though videos in the Entertainment and Music categories are oft
en trending, they have a higher number of dislikes.
```

3.3.6 Comment Ratio

In [26]:
```
plt.figure(figsize = (18,10))

# Plot comment rate distribution
g2 = sns.lvplot(y='% commented', x='category_name', data=df_gb)
g2.set_xticklabels(g2.get_xticklabels(),rotation=30)
g2.set_title("Distribution of comment rate", fontsize=20)
g2.set_xlabel("Category Names", fontsize=15)
g2.set_ylabel("Comment Rate", fontsize=15)
# Comment rate distribution
functions=['count','mean','max','min','var']
var = df_gb.groupby(['category_name'])
var['% commented'].agg(functions)
plt.annotate("Highest", xy = (14,0.5), xytext=(14.2,1.5), arrowprops={'color': 'red'
})
plt.annotate("Lowest", xy = (11,0.5), xytext=(11.2,1.5), arrowprops={'color': 'red'})
```

Out[26]: <Figure size 5400x3000 with 0 Axes>

```
/opt/anaconda/envs/Python3/lib/python3.8/site-packages/seaborn/categorical.py:2613: U
serWarning: The `lvplot` function has been renamed to `boxenplot`. The original name
will be removed in a future release. Please update your code.
  warnings.warn(msg)
```

Out[26]:
```
[Text(0, 0, 'Sports'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'Entertainment'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Music'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'News & Politics'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Comedy'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Pets & Animals'),
 Text(0, 0, 'Shows')]
```

Out[26]: Text(0.5, 1.0, 'Distribution of comment rate')

Out[26]: Text(0.5, 0, 'Category Names')

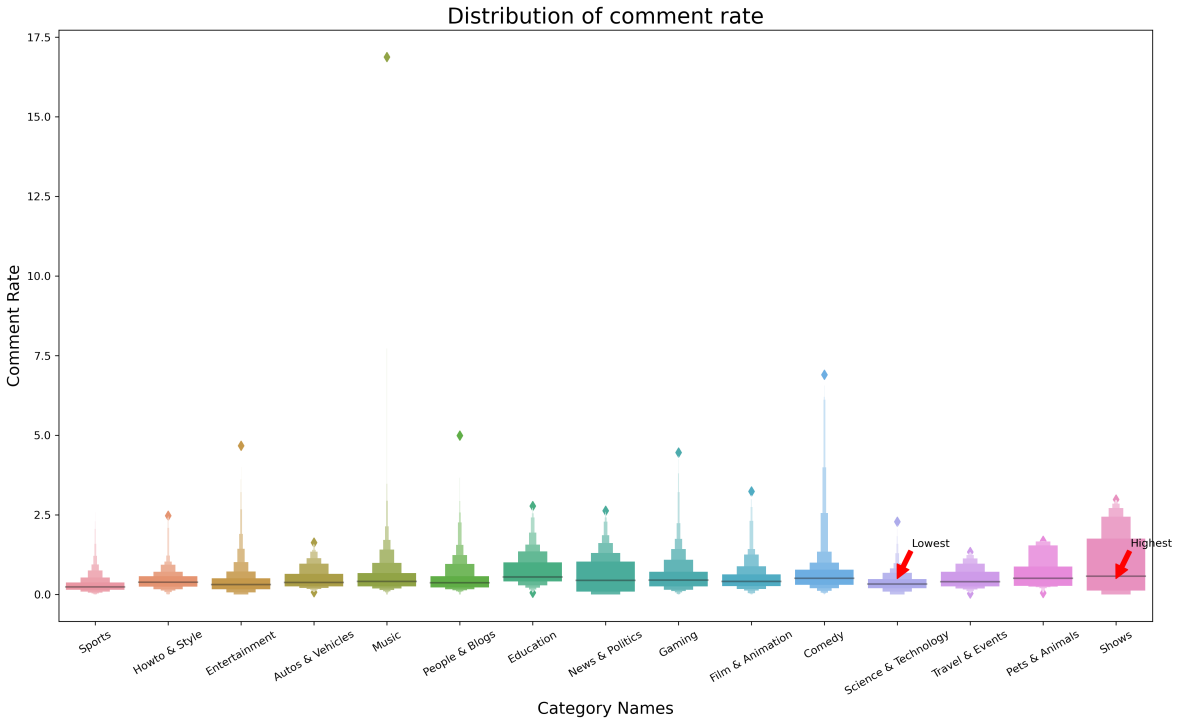
Out[26]: Text(0, 0.5, 'Comment Rate')

Out[26]:

category_name	count	mean	max	min	var
Autos & Vehicles	111	0.503497	1.628952	0.071514	0.122402
Comedy	250	0.685827	6.903758	0.000000	0.724456
Education	101	0.742928	2.777612	0.059277	0.265487
Entertainment	1054	0.410905	4.673902	0.000000	0.206227
Film & Animation	126	0.529517	3.236157	0.000000	0.237986
Gaming	369	0.586018	4.456775	0.000000	0.301181
Howto & Style	167	0.449848	2.472725	0.000000	0.109716
Music	780	0.586221	16.878641	0.000000	0.690104
News & Politics	153	0.622890	2.627783	0.000000	0.359726
People & Blogs	338	0.510517	4.995151	0.000000	0.252629
Pets & Animals	27	0.677231	1.684172	0.045729	0.277120
Science & Technology	248	0.382541	2.284074	0.000000	0.090211
Shows	6	1.041587	2.986602	0.000000	1.511398
Sports	1112	inf	inf	0.000000	NaN
Travel & Events	21	0.509885	1.331820	0.029860	0.128978

Out[26]: Text(14.2, 1.5, 'Highest')

Out[26]: Text(11.2, 1.5, 'Lowest')



In [30]:
```
#In the UK, people usually comment on videos from the Shows category, and comment lea
st in the Science & Technology category.
#It should be noted that while News & Politics' comment rate on average is not high,
 the standard deviation is quite high.
#This means that some news & politics related video spark a lot of discussions.
```

3.3.7 Buzzwords Analysis

In [28]:
```
df_gb_enter = df_gb[(df_gb['category_name'] == "Entertainment")]
df_gb_sports = df_gb[(df_gb['category_name'] == "Sports")]
df_gb_music = df_gb[(df_gb['category_name'] == "Music")]
df_gb_gaming = df_gb[(df_gb['category_name'] == "Gaming")]
text1 = df_gb_enter['title'].values
text2 = df_gb_sports['title'].values
text3 = df_gb_music['title'].values
text4 = df_gb_gaming['title'].values


wordcloud1 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(text1))


wordcloud2 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(text2))

wordcloud3 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(text3))

wordcloud4 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(text4))


# plot the WordCloud image
plt.figure(figsize = (10,10), facecolor = None)
plt.subplot(2, 2, 1)
plt.imshow(wordcloud1)
plt.axis("off")
plt.title("Entertainment")
plt.subplot(2, 2, 2)
plt.imshow(wordcloud2)
plt.axis("off")
plt.title("Sports")
plt.subplot(2, 2, 3)
plt.imshow(wordcloud3)
plt.axis("off")
plt.title("Music")
plt.subplot(2, 2, 4)
plt.imshow(wordcloud4)
plt.axis("off")
plt.title("Gaming")
plt.tight_layout(pad = 2)

plt.show()
```

Out[28]: <Figure size 3000x3000 with 0 Axes>

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7fce69c24910>

Out[28]: <matplotlib.image.AxesImage at 0x7fce69c407f0>

Out[28]: (-0.5, 799.5, 799.5, -0.5)

Out[28]: Text(0.5, 1.0, 'Entertainment')

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7fce6a507be0>

Out[28]: <matplotlib.image.AxesImage at 0x7fce6a507fd0>

Out[28]: (-0.5, 799.5, 799.5, -0.5)

Out[28]: Text(0.5, 1.0, 'Sports')

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7fce69e1c370>

Out[28]: <matplotlib.image.AxesImage at 0x7fce69e1c3d0>

Out[28]: (-0.5, 799.5, 799.5, -0.5)

Out[28]: Text(0.5, 1.0, 'Music')

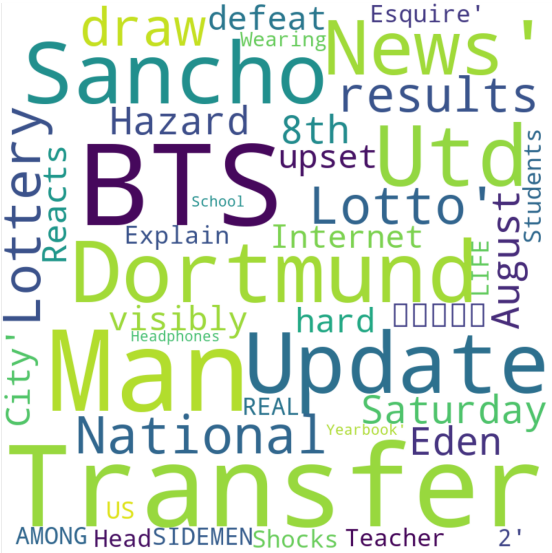
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7fce6a6b4ac0>

Out[28]: <matplotlib.image.AxesImage at 0x7fce6a6b4df0>

Out[28]: (-0.5, 799.5, 799.5, -0.5)

Out[28]: Text(0.5, 1.0, 'Gaming')

Entertainment



Sports

Music

Gaming

Entertainment

A few of the most frequent words in Entertainment are related to the rumours about the British born football star Jadon Sancho (such as Sancho, Dortmund, and transfer). Various videos of specialists expressing their opinion have been posted (Keith, 2020). Additionally, BTS released another music video on 20/11/2020, drawing a lot of attention (Big Hit Labels, 2020).

Sport

As mentioned, the high attention in the category of sport is due to the football leagues (UEFA). Many trending videos cover matches highlights, intervieos and live reactions.

Music

In the music category, there is no dominating theme, but official music video releases from various artist like Little Mix, Justin Bieber or Blackpink. The word cloud is a good representation for the genres preferred in the UK.

Gaming

1. Xbox series S was released on 10/11/2020 (Xbox,2020). In the gaming category, many trending videos linked to the new gaming console, either reviewing it or comparing it with other gaming consoles.
2. Live streams of Minecraft, Cyberpunk, and Apex video seem to be another popular type of video in the UK.

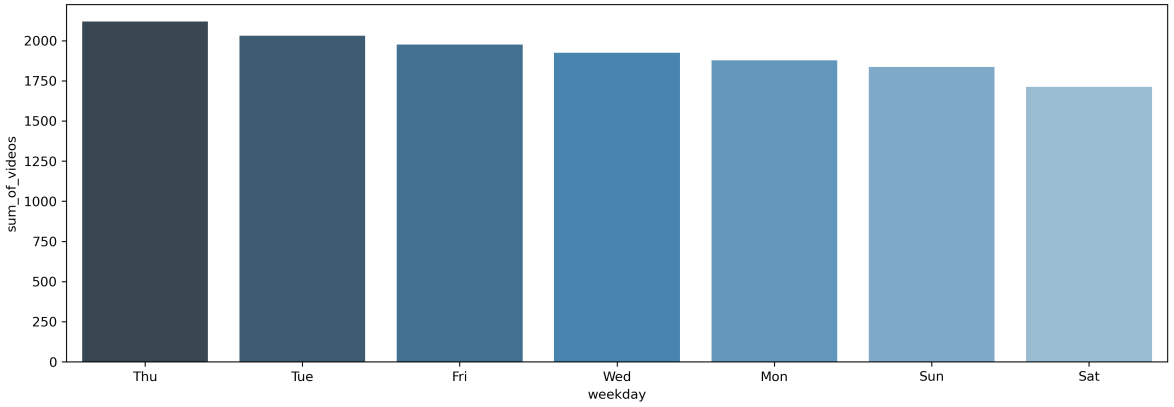
3.4 Russia RU

3.4.1 Publishing Day & Time

In [258]:
```
week_plot = df_ru['publishedAt'].apply(lambda x: x.weekday()).value_counts().reset_in
dex().rename(columns = {"index": "weekday", "publishedAt": "sum_of_videos"})
week_plot["weekday"] = week_plot["weekday"].replace({0: "Mon",1:"Tue",2:"Wed",3:"Thu"
,4:"Fri",5:"Sat",6:"Sun"})
week_plot
fig, ax = plt.subplots(figsize=(15,5))
g =sns.barplot(x="weekday", y="sum_of_videos", data = week_plot,palette=("Blues_d"))
```

Out[258]:

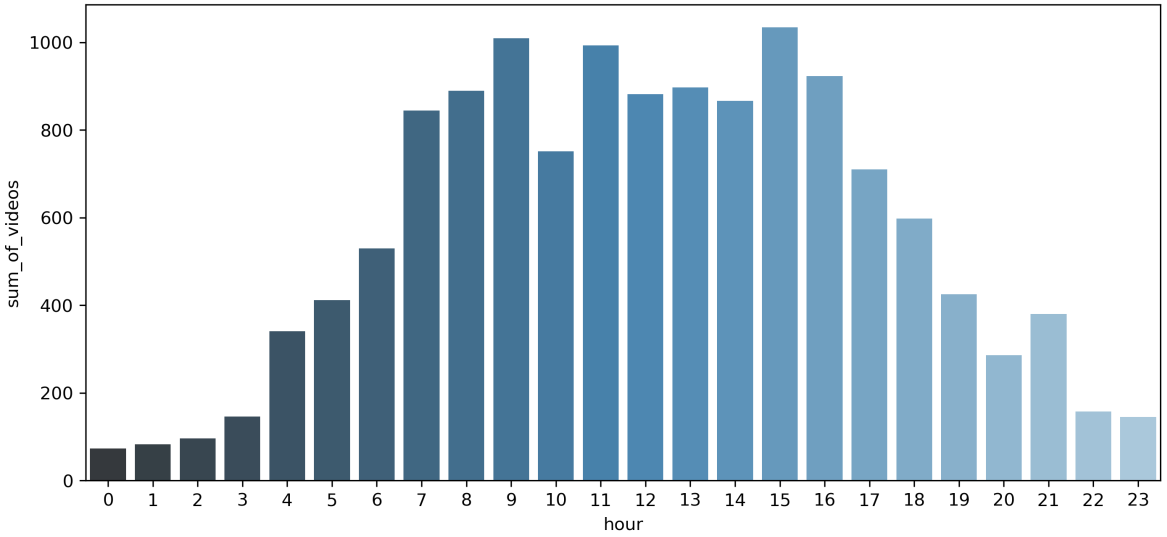
	weekday	sum_of_videos
0	Thu	2120
1	Tue	2031
2	Fri	1976
3	Wed	1924
4	Mon	1878
5	Sun	1836
6	Sat	1712



In [ ]:
```
#Trending videos are usually publised on Thursday.
```

In [259]:
```
df_ru['hour'] = df_ru["publishedAt"].dt.hour
hour_plot = df_ru["hour"].value_counts().reset_index().rename(columns = {"index": "hour", "hour": "sum_of_videos"})
fig, ax = plt.subplots(figsize=(11,5))
sns.barplot(x="hour", y="sum_of_videos", data = hour_plot,palette=("Blues_d"))
```

Out[259]: <matplotlib.axes._subplots.AxesSubplot at 0x7f655df966a0>



In [ ]:
```
#The majority of trending videos within a big time window from 9 am to 5pm.
```

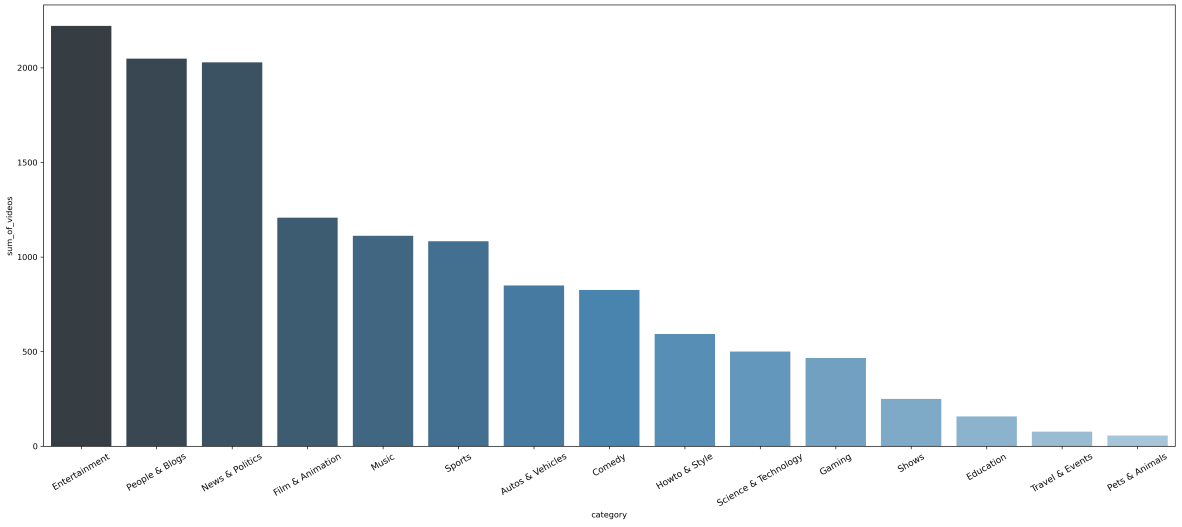
3.4.2 Count by Categories

In [260]:
```
category_plot = df_ru["category_name"].value_counts().reset_index().rename(columns =
{"index": "category", "category_name": "sum_of_videos"})
category_plot
fig, ax = plt.subplots(figsize=(25,10))
g = sns.barplot(x="category", y="sum_of_videos", data = category_plot,palette=("Blues
_d"))
g.set_xticklabels(g.get_xticklabels(), rotation=30, fontsize=11)
```

Out[260]:

	category	sum_of_videos
0	Entertainment	2222
1	People & Blogs	2048
2	News & Politics	2029
3	Film & Animation	1208
4	Music	1112
5	Sports	1083
6	Autos & Vehicles	850
7	Comedy	826
8	Howto & Style	593
9	Science & Technology	500
10	Gaming	466
11	Shows	250
12	Education	157
13	Travel & Events	77
14	Pets & Animals	56

Out[260]:
```
[Text(0, 0, 'Entertainment'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'News & Politics'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Music'),
 Text(0, 0, 'Sports'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Comedy'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'Shows'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Pets & Animals')]
```



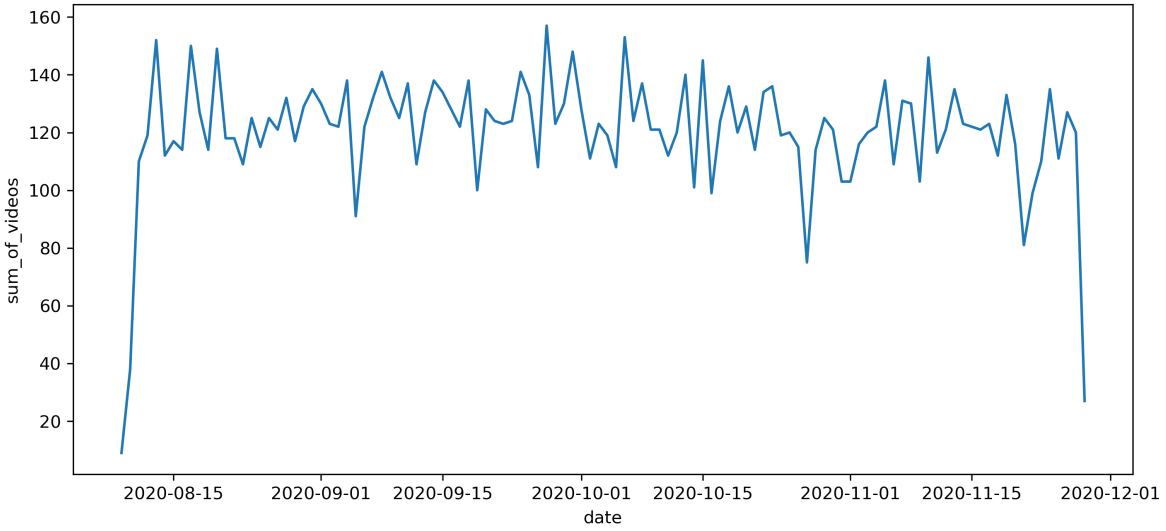
In [ ]:
```
#Entertainment has the highest number of trending videos, followed by People&Blogs an
d News&Politics
```

3.4.3 Trending Videos over Time

In [261]:
```
df_ru['date'] = df_ru["publishedAt"].dt.date
df_ru['date'].head()
date_plot = df_ru["date"].value_counts().reset_index().rename(columns = {"index": "da
te", "date": "sum_of_videos"})
fig, ax = plt.subplots(figsize=(11,5))
sns.lineplot(x="date", y="sum_of_videos", data = date_plot,palette=("Blues_d"))
plt.show()
date_plot.head()
```

Out[261]:
```
10    2020-08-11
12    2020-08-11
22    2020-08-11
29    2020-08-11
44    2020-08-12
Name: date, dtype: object
```

Out[261]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6560ae4f70>
```



Out[261]:

	date	sum_of_videos
0	2020-09-27	157
1	2020-10-06	153
2	2020-08-13	152
3	2020-08-17	150
4	2020-08-20	149

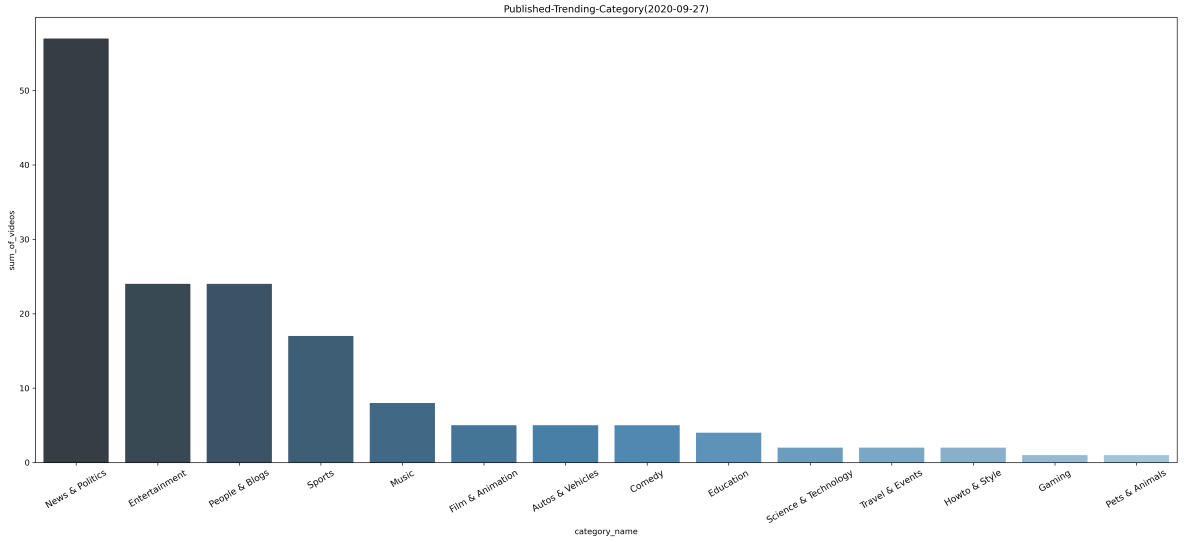
In [262]:
```
df1 = df_ru.set_index('publishedAt')
df1["2020-09-27"]["category_name"].value_counts()
```

Out[262]:
```
News & Politics         57
Entertainment           24
People & Blogs          24
Sports                  17
Music                    8
Film & Animation         5
Autos & Vehicles         5
Comedy                   5
Education                4
Science & Technology     2
Travel & Events          2
Howto & Style            2
Gaming                   1
Pets & Animals           1
Name: category_name, dtype: int64
```

In [263]:
```
category_plot = df1["2020-09-27"]["category_name"].value_counts().reset_index().renam
e(columns = {"index": "category_name", "category_name": "sum_of_videos"})
fig, ax = plt.subplots(figsize=(25,10))
g = sns.barplot(x ="category_name" ,y="sum_of_videos" ,data = category_plot,palette=(
"Blues_d"))
g.set_title("Published-Trending-Category(2020-09-27)")
g.set_xticklabels(g.get_xticklabels(), rotation=30, fontsize=11)
```

Out[263]:
```
Text(0.5, 1.0, 'Published-Trending-Category(2020-09-27)')
```

Out[263]:
```
[Text(0, 0, 'News & Politics'),
 Text(0, 0, 'Entertainment'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'Sports'),
 Text(0, 0, 'Music'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Comedy'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'Pets & Animals')]
```



On September 27th, the highest number of trending videos were under News&Politics category, surpassing the Entertainment and People&Blogs categories combined. On September 27th, 2020, Nagorno-Karabakh war, in which Russia was involved, started. This explains the popularity of the category.

3.4.4 Likes to Dislikes Ratio per Category

In [264]:
```
ru_ldratio =df_ru.groupby(['category_name'])['ldratio'].mean().reset_index()
ru_ldratio
plt.figure(figsize = (20,10))
g_ru = sns.barplot(x="category_name", y="ldratio", data=ru_ldratio, palette="Blues_d"
,
           order=ru_ldratio.sort_values("ldratio", ascending=False).category_name)
g_ru.set_xticklabels(g_ru.get_xticklabels(), rotation=30, fontsize=11)
plt.ylabel("ldratio")
```

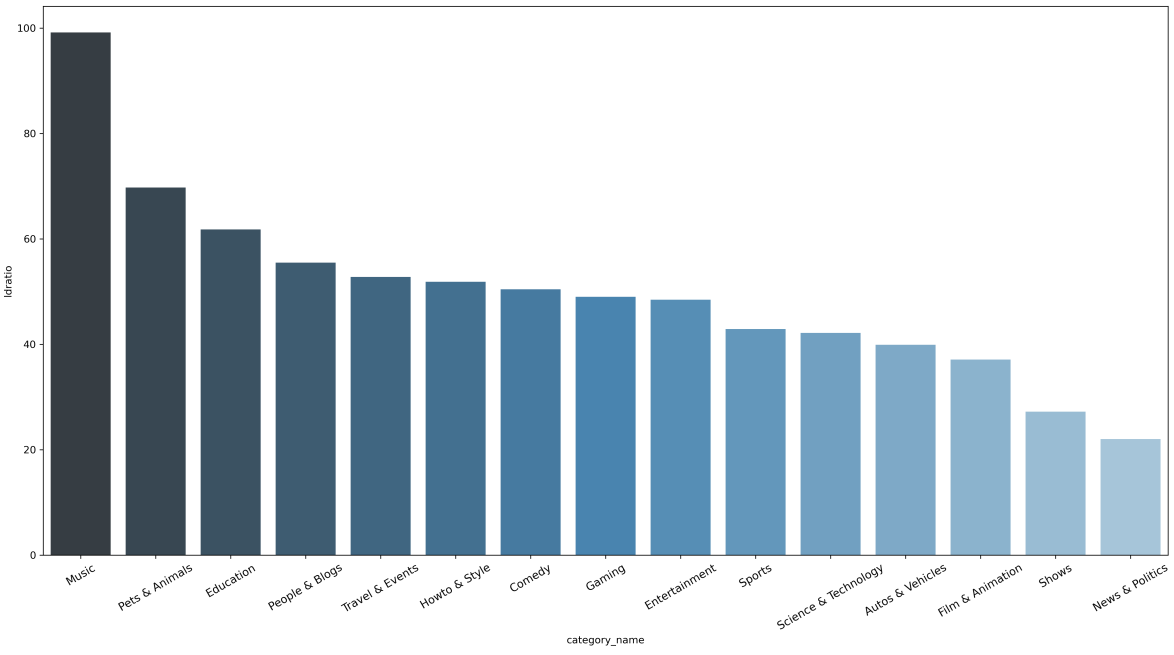
Out[264]:

	category_name	ldratio
0	Autos & Vehicles	39.885018
1	Comedy	50.400255
2	Education	61.782739
3	Entertainment	48.446444
4	Film & Animation	37.081387
5	Gaming	48.983297
6	Howto & Style	51.864814
7	Music	99.170411
8	News & Politics	21.993052
9	People & Blogs	55.467034
10	Pets & Animals	69.716964
11	Science & Technology	42.127565
12	Shows	27.219920
13	Sports	42.875880
14	Travel & Events	52.755455

Out[264]:
```
<Figure size 6000x3000 with 0 Axes>
```

Out[264]:
```
[Text(0, 0, 'Music'),
 Text(0, 0, 'Pets & Animals'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'Comedy'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'Entertainment'),
 Text(0, 0, 'Sports'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Shows'),
 Text(0, 0, 'News & Politics')]
```

Out[264]:
```
Text(0, 0.5, 'ldratio')
```



In [31]:
```
#Videos in Music category are most likely to get high like rates, followed by Pets&An
imals.
```

3.4.5 Comment Ratio

Comment rates analysis for Russia

In [27]:
```
plt.figure(figsize = (18,10))

# Plot comment rate distribution
g2= sns.lvplot(y='% commented', x='category_name', data=df_ru)
g2.set_xticklabels(g2.get_xticklabels(),rotation=30)
g2.set_title("Distribution of comment rate", fontsize=20)
g2.set_xlabel("Category Names", fontsize=15)
g2.set_ylabel("Comment Rate", fontsize=15)
# Comment rate distribution
functions=['count','mean','max','min','var']
var = df_ru.groupby(['category_name'])
var['% commented'].agg(functions)
plt.annotate("Highest", xy = (7,0.7), xytext=(7.2,1.9), arrowprops={'color': 'red'})
plt.annotate("Lowest", xy = (8,0.3), xytext=(8.2,1.5), arrowprops={'color': 'red'})
```

Out[27]: <Figure size 5400x3000 with 0 Axes>

```
/opt/anaconda/envs/Python3/lib/python3.8/site-packages/seaborn/categorical.py:2613: U
serWarning: The `lvplot` function has been renamed to `boxenplot`. The original name
will be removed in a future release. Please update your code.
  warnings.warn(msg)
```

Out[27]:
```
[Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'News & Politics'),
 Text(0, 0, 'Comedy'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Entertainment'),
 Text(0, 0, 'Music'),
 Text(0, 0, 'Sports'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'Pets & Animals'),
 Text(0, 0, 'Shows'),
 Text(0, 0, 'Travel & Events')]
```

Out[27]: Text(0.5, 1.0, 'Distribution of comment rate')

Out[27]: Text(0.5, 0, 'Category Names')

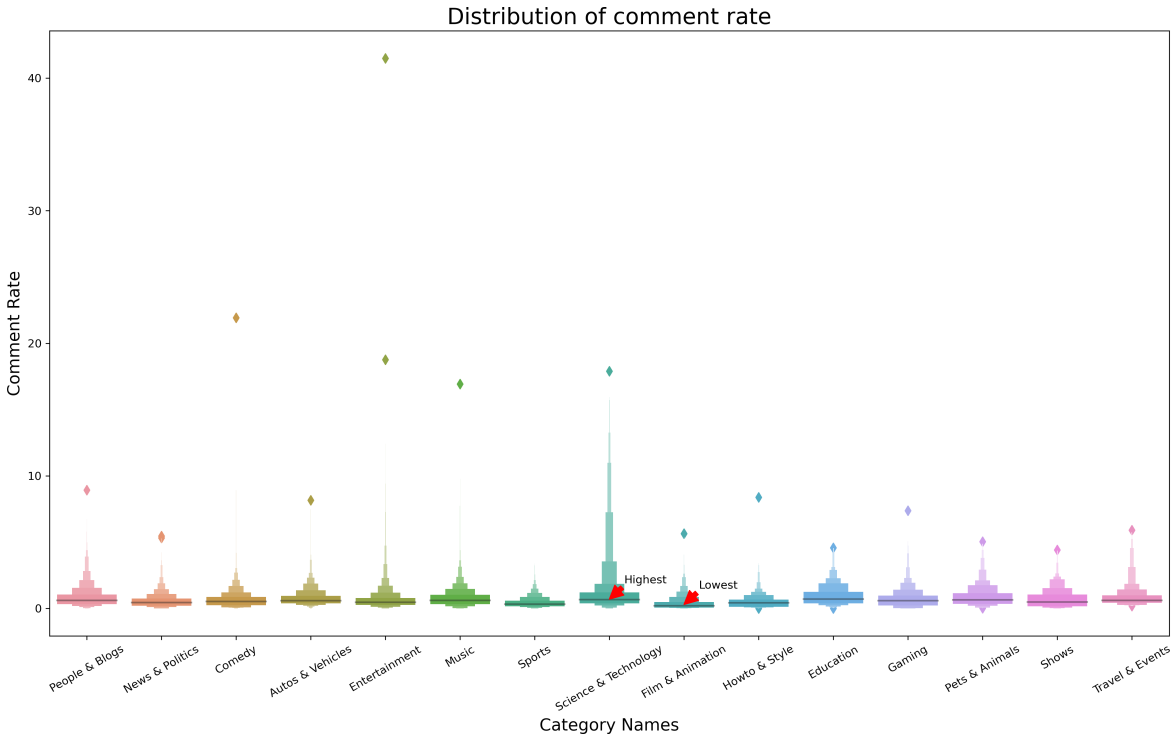
Out[27]: Text(0, 0.5, 'Comment Rate')

Out[27]:

category_name	count	mean	max	min	var
Autos & Vehicles	850	0.780784	8.167166	0.000000	0.519511
Comedy	826	0.681871	21.933187	0.000000	1.052180
Education	157	0.953172	4.570513	0.006975	0.659338
Entertainment	2222	0.699837	41.500051	0.000000	1.611839
Film & Animation	1208	0.390072	5.636734	0.000000	0.300310
Gaming	466	0.736066	7.368519	0.000000	0.604980
Howto & Style	593	0.517113	8.383743	0.000000	0.325454
Music	1112	0.796457	16.925526	0.000000	0.804354
News & Politics	2029	0.573066	5.455866	0.000000	0.329507
People & Blogs	2048	0.834759	8.934176	0.000000	0.680671
Pets & Animals	56	0.898520	5.027890	0.027140	0.840956
Science & Technology	500	1.259004	17.898064	0.000000	4.763142
Shows	250	0.722476	4.406527	0.000000	0.574311
Sports	1083	inf	inf	0.000000	NaN
Travel & Events	77	0.878554	5.908117	0.182249	0.863872

Out[27]: Text(7.2, 1.9, 'Highest')

Out[27]: Text(8.2, 1.5, 'Lowest')



Science & Technology videos are the most discussed while Film & Animation videos received the coldest response.

3.4.6 Buzzwords Analysis

In [266]:
```
df_ru_enter = df_ru[(df_ru['category_name'] == "Entertainment")]
df_ru_pb = df_ru[(df_ru['category_name'] == "People & Blogs")]
df_ru_np = df_ru[(df_ru['category_name'] == "News & Politics")]
df_ru_fa = df_ru[(df_ru['category_name'] == "Film & Animation")]
text1 = df_ru_enter['title'].values
text2 = df_ru_pb['title'].values
text3 = df_ru_np['title'].values
text4 = df_ru_fa['title'].values

from google_trans_new import google_translator
translator = google_translator()
translate_text1 = translator.translate(text1)
translate_text2 = translator.translate(text2)
translate_text3 = translator.translate(text3)
translate_text4 = translator.translate(text4)
```

In [267]:
```
wordcloud1 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(translate_text1))

wordcloud2 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(translate_text2))

wordcloud3 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(translate_text3))
wordcloud4 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(translate_text4))



# plot the WordCloud image
plt.figure(figsize = (10,10), facecolor = None)
plt.subplot(2, 2, 1)
plt.imshow(wordcloud1)
plt.axis("off")
plt.title("Entertainment")
plt.subplot(2, 2, 2)
plt.imshow(wordcloud2)
plt.axis("off")
plt.title("People & Blogs")
plt.subplot(2, 2, 3)
plt.imshow(wordcloud3)
plt.axis("off")
plt.title("News & Politics")
plt.subplot(2, 2, 4)
plt.imshow(wordcloud4)
plt.axis("off")
plt.title("Film & Animation")
plt.tight_layout(pad = 2)

plt.show()
```

Out[267]: <Figure size 3000x3000 with 0 Axes>

Out[267]: <matplotlib.axes._subplots.AxesSubplot at 0x7f655e10ff40>

Out[267]: <matplotlib.image.AxesImage at 0x7f655dd86ee0>

Out[267]: (-0.5, 799.5, 799.5, -0.5)

Out[267]: Text(0.5, 1.0, 'Entertainment')

Out[267]: <matplotlib.axes._subplots.AxesSubplot at 0x7f655dd9c3a0>

Out[267]: <matplotlib.image.AxesImage at 0x7f655dd9c7f0>

Out[267]: (-0.5, 799.5, 799.5, -0.5)

Out[267]: Text(0.5, 1.0, 'People & Blogs')

Out[267]: <matplotlib.axes._subplots.AxesSubplot at 0x7f655e054af0>

Out[267]: <matplotlib.image.AxesImage at 0x7f655e054f40>

Out[267]: (-0.5, 799.5, 799.5, -0.5)

Out[267]: Text(0.5, 1.0, 'News & Politics')

Out[267]: <matplotlib.axes._subplots.AxesSubplot at 0x7f655e0ca280>

Out[267]: <matplotlib.image.AxesImage at 0x7f655e0ca6d0>

Out[267]: (-0.5, 799.5, 799.5, -0.5)

Out[267]: Text(0.5, 1.0, 'Film & Animation')

As shown in the buzzwords, some features are specific to Russia.

Entertainment

Russians have a sweet tooth! In 2020, Russia increased their cocoa and chocolate imports (Seanews, 2020). We assume that this culinary trend was linked to the videos' popularity as well. In addition, TikTok became very popular in Russia. TikTok's Russian audience is around 8 million people/month (IQBAL, 2020), which explains why we have it as a trending category.

News & Politics

Russian people tend to be sensitive to politically-linked topics. Keywords like "Massacre", "Air", "Special", seem to be also identified by our function.

3.5 US US

3.5.1 Count by Categories

In [15]:
```
# Plot count of categories for USA.
plt.figure(figsize = (24,10))
g = sns.countplot('category_name', data=df_us, palette="Blues_d",order = df_us['categ
ory_name'].value_counts().index)
g.set_xticklabels(g.get_xticklabels(), rotation=30, fontsize=11)
g.set_title("Count of Video Categories ", fontsize=20)
g.set_xlabel("", fontsize=15)
g.set_ylabel("Count", fontsize=15)
```

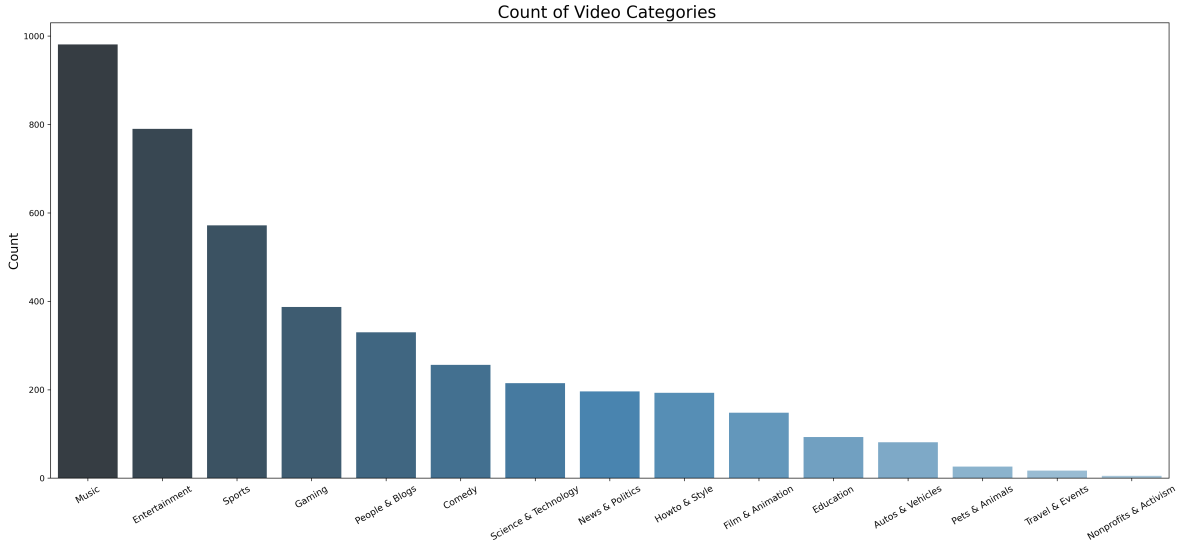
Out[15]:
```
<Figure size 7200x3000 with 0 Axes>
```

Out[15]:
```
[Text(0, 0, 'Music'),
 Text(0, 0, 'Entertainment'),
 Text(0, 0, 'Sports'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'Comedy'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'News & Politics'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Pets & Animals'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Nonprofits & Activism')]
```

Out[15]:
```
Text(0.5, 1.0, 'Count of Video Categories ')
```

Out[15]:
```
Text(0.5, 0, '')
```

Out[15]:
```
Text(0, 0.5, 'Count')
```

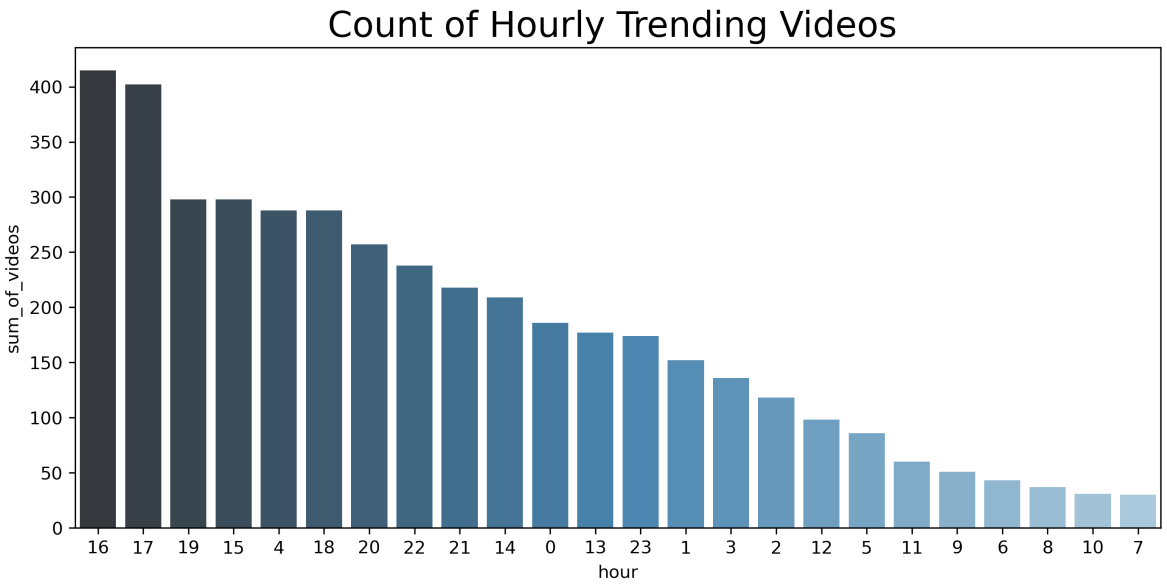


In [ ]:
```
#Music videos account for the majority of the USA trending videos, followed by Eterta
inment. Nonprofits & Activism has the least amount.
```

3.5.2 Publishing Time

In [17]:
```
# Plot count of hourly trending videos.
plt.rcParams['figure.dpi'] =300
us_hour = df_us["publishedAt"].dt.hour.value_counts().reset_index().rename(columns =
{"index": "hour", "publishedAt": "sum_of_videos"})
fig, ax = plt.subplots(figsize=(11,5))
g=sns.barplot(x="hour", y="sum_of_videos", data = us_hour,palette=("Blues_d"),
              order=us_hour.sort_values("sum_of_videos", ascending=False).hour)
g.set_title("Count of Hourly Trending Videos ", fontsize=20)
```

Out[17]:
```
Text(0.5, 1.0, 'Count of Hourly Trending Videos ')
```



In [ ]:
```
#Trending videos in the USA mostly get published after 4 p.m.
#Interestingly enough, 4 a.m. is another popular time to publish trending videos too.
#This might be because they anticipate the viewers will watch them early in the morni
ng
```

3.5.3 Publishing Day

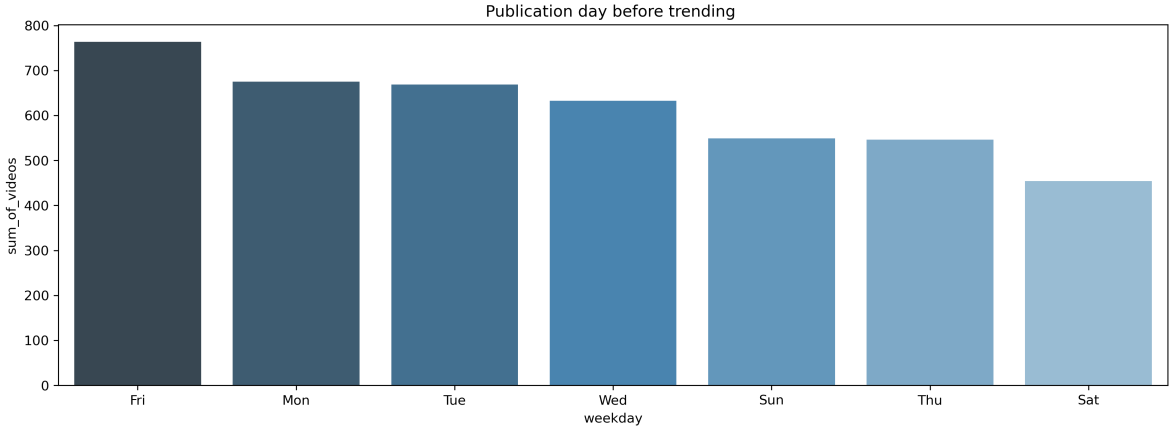
In [53]:
```
week_plot_us = df_us['publishedAt'].apply(lambda x: x.weekday()).value_counts().reset
_index().rename(columns = {"index": "weekday", "publishedAt": "sum_of_videos"})
week_plot_us["weekday"] = week_plot_us["weekday"].replace({0: "Mon",1:"Tue",2:"Wed",3
:"Thu",4:"Fri",5:"Sat",6:"Sun"})
week_plot_us
fig, ax = plt.subplots(figsize=(15,5))
sns.barplot(x="weekday", y="sum_of_videos", data = week_plot_us,palette=("Blues_d"))
plt.title("Publication day before trending ")
## Monday =0, sunday = 6
```

Out[53]:

	weekday	sum_of_videos
0	Fri	764
1	Mon	675
2	Tue	669
3	Wed	633
4	Sun	549
5	Thu	546
6	Sat	454

Out[53]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f99ae6dbcd0>`

Out[53]: `Text(0.5, 1.0, 'Publication day before trending ')`



In [ ]:
```
#In the USA, most videos get published on Fridays and Mondays.
```

3.5.4 Trending Day

In [18]:
```
week_plot_us = df_us['trending_date'].apply(lambda x: x.weekday()).value_counts().res
et_index().rename(columns = {"index": "weekday", "trending_date": "sum_of_videos"})
week_plot_us["weekday"] = week_plot_us["weekday"].replace({0: "Mon",1:"Tue",2:"Wed",3
:"Thu",4:"Fri",5:"Sat",6:"Sun"})
week_plot_us
fig, ax = plt.subplots(figsize=(15,5))
sns.barplot(x="weekday", y="sum_of_videos", data = week_plot_us,palette=("Blues_d"))
plt.xlabel("")
plt.ylabel("Sum of videos")
plt.title("Day become trending")
```

Out[18]:

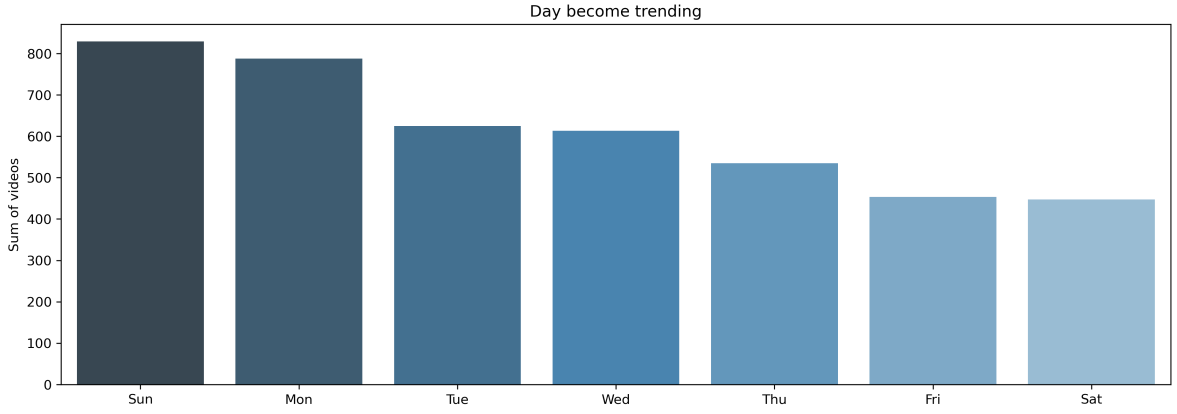
	weekday	sum_of_videos
0	Sun	829
1	Mon	788
2	Tue	625
3	Wed	613
4	Thu	535
5	Fri	453
6	Sat	447

Out[18]: `<matplotlib.axes._subplots.AxesSubplot at 0x7fa1407e0ca0>`

Out[18]: `Text(0.5, 0, '')`

Out[18]: `Text(0, 0.5, 'Sum of videos')`

Out[18]: `Text(0.5, 1.0, 'Day become trending')`



In [ ]:
```
#In the USA, the day videos become trending is according to our expectations.
#As the viewers have more free time during the weekend, they are more likely to get v
ideos to trending. It is therefore, advisable to post on Fridays.
```

3.5.5 Likes to Dislikes Ratio per Category

In [55]:
```
us_ldratio =df_us.groupby(['category_name'])['ldratio'].mean().reset_index()
us_ldratio
plt.figure(figsize = (25,10))
g = sns.barplot(x="category_name", y="ldratio", data=us_ldratio, palette="Blues_d",
           order=us_ldratio.sort_values("ldratio", ascending=False).category_name)
g.set_xticklabels(g.get_xticklabels(), rotation=30, fontsize=11)
plt.ylabel("ldratio")
```

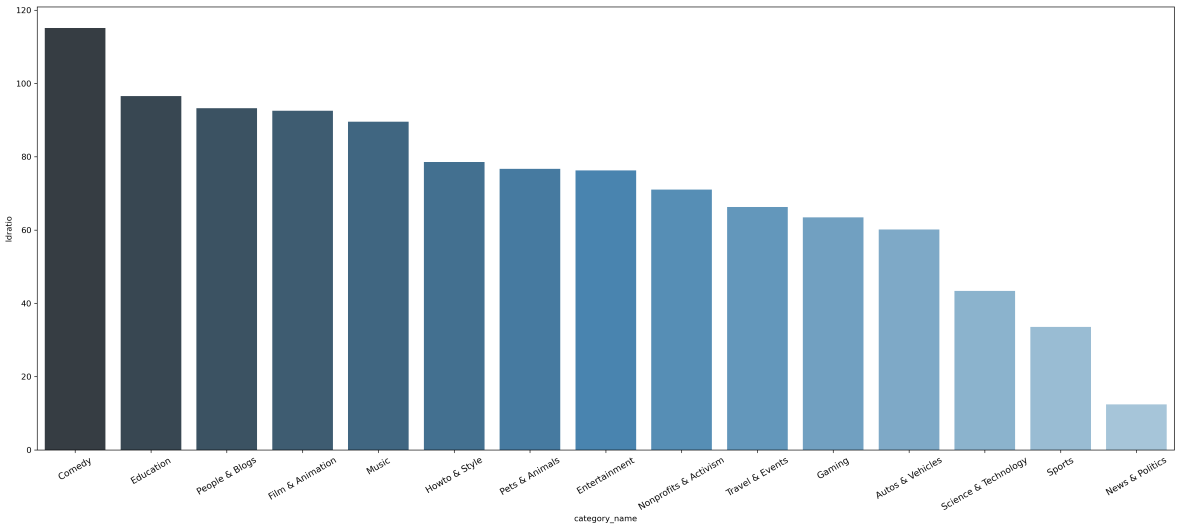
Out[55]:

	category_name	ldratio
0	Autos & Vehicles	60.152658
1	Comedy	115.147148
2	Education	96.554457
3	Entertainment	76.248104
4	Film & Animation	92.564459
5	Gaming	63.477552
6	Howto & Style	78.551250
7	Music	89.532267
8	News & Politics	12.405737
9	Nonprofits & Activism	71.012000
10	People & Blogs	93.245435
11	Pets & Animals	76.681538
12	Science & Technology	43.401033
13	Sports	33.568956
14	Travel & Events	66.297647

Out[55]: <Figure size 7500x3000 with 0 Axes>

Out[55]:
```
[Text(0, 0, 'Comedy'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Music'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'Pets & Animals'),
 Text(0, 0, 'Entertainment'),
 Text(0, 0, 'Nonprofits & Activism'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'Sports'),
 Text(0, 0, 'News & Politics')]
```

Out[55]: Text(0, 0.5, 'ldratio')



In [32]:
```
#The Comedy category has the highest likes to dislikes ration, which means that gener
ally the audience gives positive feedback to those videos.
#On the opposite side, News & Politics tends to receive mixed feedback, hence the rat
io is low. Other categories that generally receive appreciation are:
#Education, People & Blogs, and Film & Animation.
```

3.5.6 Comment Ratio

In [28]:
```
plt.figure(figsize = (18,10))

# Plot comment rate distribution
g2= sns.lvplot(y='% commented', x='category_name', data=df_us)
g2.set_xticklabels(g2.get_xticklabels(),rotation=30)
g2.set_title("Distribution of comment rate", fontsize=20)
g2.set_xlabel("Category Names", fontsize=15)
g2.set_ylabel("Comment Rate", fontsize=15)
# Comment rate distribution
functions=['count','mean','max','min','var']
var = df_us.groupby(['category_name'])
var['% commented'].agg(functions)
plt.annotate("Highest", xy = (14,0.6), xytext=(14.2,1.2), arrowprops={'color': 'red'
})
plt.annotate("Lowest", xy = (4,0.3), xytext=(4.2,0.7), arrowprops={'color': 'red'})
```

Out[28]: <Figure size 5400x3000 with 0 Axes>

```
/opt/anaconda/envs/Python3/lib/python3.8/site-packages/seaborn/categorical.py:2613: U
serWarning: The `lvplot` function has been renamed to `boxenplot`. The original name
will be removed in a future release. Please update your code.
  warnings.warn(msg)
```

Out[28]:
```
[Text(0, 0, 'Music'),
 Text(0, 0, 'Howto & Style'),
 Text(0, 0, 'Entertainment'),
 Text(0, 0, 'News & Politics'),
 Text(0, 0, 'Sports'),
 Text(0, 0, 'Film & Animation'),
 Text(0, 0, 'Comedy'),
 Text(0, 0, 'Autos & Vehicles'),
 Text(0, 0, 'Gaming'),
 Text(0, 0, 'People & Blogs'),
 Text(0, 0, 'Science & Technology'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Travel & Events'),
 Text(0, 0, 'Pets & Animals'),
 Text(0, 0, 'Nonprofits & Activism')]
```

Out[28]: Text(0.5, 1.0, 'Distribution of comment rate')

Out[28]: Text(0.5, 0, 'Category Names')

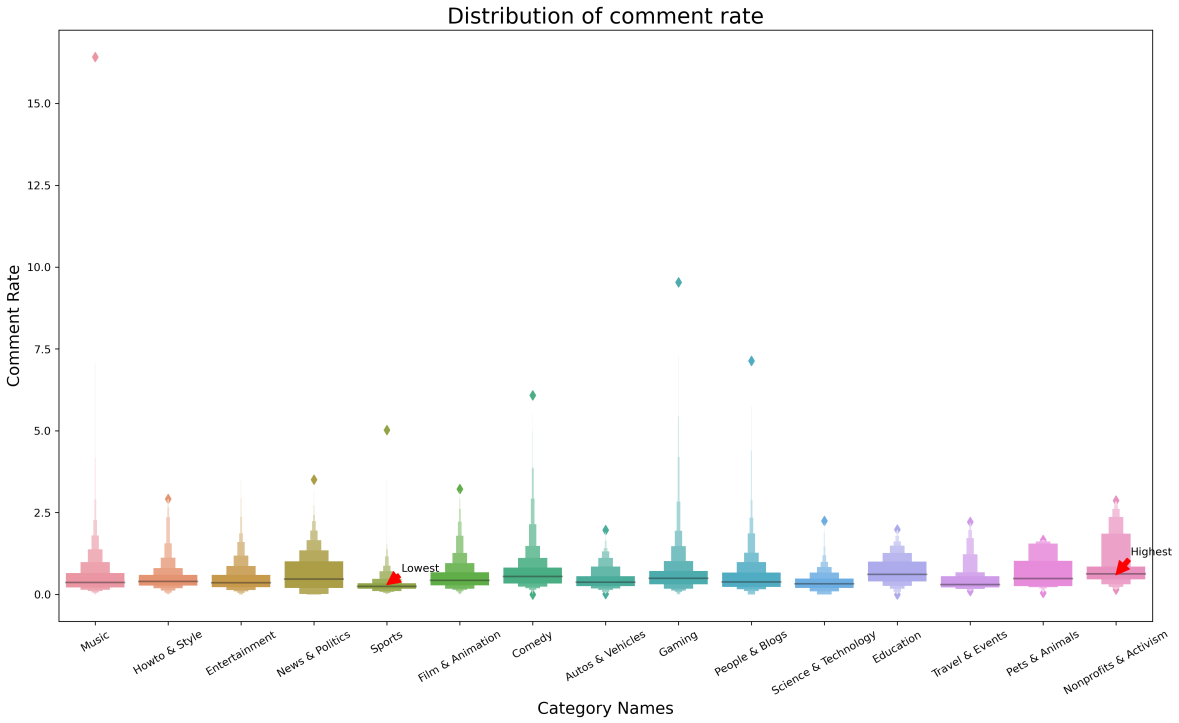
Out[28]: Text(0, 0.5, 'Comment Rate')

Out[28]:

category_name	count	mean	max	min	var
Autos & Vehicles	81	0.470510	1.968528	0.014718	0.125062
Comedy	256	0.702869	6.083203	0.000000	0.436205
Education	93	0.725318	1.981573	0.000000	0.195812
Entertainment	790	inf	inf	0.000000	NaN
Film & Animation	148	0.560622	3.224479	0.000000	0.227536
Gaming	387	0.646373	9.539418	0.000000	0.580868
Howto & Style	193	0.513694	2.918741	0.000000	0.206668
Music	980	0.553408	16.420872	0.000000	0.622499
News & Politics	196	0.653035	3.510026	0.000000	0.354874
Nonprofits & Activism	5	0.992525	2.871130	0.148522	1.168270
People & Blogs	330	0.554290	7.139718	0.000000	0.402078
Pets & Animals	26	0.714253	1.668774	0.046485	0.285235
Science & Technology	215	0.378174	2.249999	0.000000	0.089367
Sports	572	0.312669	5.025138	0.000000	0.107658
Travel & Events	17	0.498665	2.215122	0.103036	0.270099

Out[28]: Text(14.2, 1.2, 'Highest')

Out[28]: Text(4.2, 0.7, 'Lowest')



In [ ]:
```
#In the USA, News & Politics and Nonprofits & Activism videos are on average the most commented on.
#Comedy, Gaming, Education, and people & Blogs also receive a high number of comment s, consistent with like/dislike ration analysis.
```

3.5.7 Buzzwords Analysis

In [20]:
```
df_us_comedy = df_us[(df_us['category_name'] == "Comedy")]
df_us_peopleblog = df_us[(df_us['category_name'] == "People & Blogs")]
df_us_education = df_us[(df_us['category_name'] == "Education")]
df_us_game = df_us[(df_us['category_name'] == "Gaming")]
text1 = df_us_comedy['title'].values
text2 = df_us_peopleblog['title'].values
text3 = df_us_education['title'].values
text4 = df_us_game['title'].values


wordcloud1 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(text1))


wordcloud2 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(text2))

wordcloud3 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(text3))

wordcloud4 = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(str(text4))


# plot the WordCloud image
plt.figure(figsize = (10,10), facecolor = None)
plt.subplot(2, 2, 1)
plt.imshow(wordcloud1)
plt.axis("off")
plt.subplot(2, 2, 2)
plt.imshow(wordcloud2)
plt.axis("off")
plt.subplot(2, 2, 3)
plt.imshow(wordcloud3)
plt.axis("off")
plt.subplot(2, 2, 4)
plt.imshow(wordcloud4)
plt.axis("off")
plt.tight_layout(pad = 3)

plt.show()
```

Out[20]: <Figure size 3000x3000 with 0 Axes>

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa13f2c5760>

Out[20]: <matplotlib.image.AxesImage at 0x7fa13f2c3e20>

Out[20]: (-0.5, 799.5, 799.5, -0.5)

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa13fc81880>

Out[20]: <matplotlib.image.AxesImage at 0x7fa13fc81790>

Out[20]: (-0.5, 799.5, 799.5, -0.5)

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa13f2dba30>

Out[20]: <matplotlib.image.AxesImage at 0x7fa13f2db1c0>

Out[20]: (-0.5, 799.5, 799.5, -0.5)

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa13fe3f430>

Out[20]: <matplotlib.image.AxesImage at 0x7fa13fe3f0a0>

Out[20]: (-0.5, 799.5, 799.5, -0.5)

Based on the likes to dislikes ratio & the comment rate analysis: comedy, people & blogs, and gaming are the most popular categories in the USA.

Comedy

People tend to enjoy comedic vlogs or funny videos. Words like "People" and "Game" are very common. This might be due to the fact that Americans enjoy content they can relate to.

People & Blogs

Words like "Pregnant", "Family", "Baby", and "iphone" are identified as frequent. Again, this is linked to our previous assumption that people in the USA prefer content "close to home".

Education

"COVID" is currently a trending topic. People started watching educational Youtube videos on the "COVID" virus. Moreover, "Method", "Songs", and "Makeup" are common tags. It is consistent with our initial expectation that people prefer videos they can relate to and help with their day-by-day lives.

Gaming

Most popular video types are game trailers (as shown by the graphic). In addition, we also observe games being mentioned (Apex, Minecraft, Pokémon). The reasoning behind watching those videos is to find tutorials or the latest news in the gaming scene.

4. Regression

In [135]:
```
# Import related statistic models and split the largest dataset Russain into training and testing data.
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from statsmodels.formula.api import glm

train, test = train_test_split(df_ru, test_size=0.2)
train.info()
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10781 entries, 12315 to 17580
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   title              10781 non-null  object
 1   publishedAt        10781 non-null  datetime64[ns]
 2   trending_date      10781 non-null  datetime64[ns]
 3   view_count         10781 non-null  int64
 4   likes              10781 non-null  int64
 5   dislikes           10781 non-null  int64
 6   comment_count      10781 non-null  int64
 7   comments_disabled  10781 non-null  bool
 8   ratings_disabled   10781 non-null  bool
 9   category_name      10581 non-null  object
 10  time_diff          10781 non-null  float64
 11  ldratio            10673 non-null  float64
dtypes: bool(2), datetime64[ns](2), float64(2), int64(4), object(2)
memory usage: 947.5+ KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2696 entries, 715 to 4323
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   title              2696 non-null   object
 1   publishedAt        2696 non-null   datetime64[ns]
 2   trending_date      2696 non-null   datetime64[ns]
 3   view_count         2696 non-null   int64
 4   likes              2696 non-null   int64
 5   dislikes           2696 non-null   int64
 6   comment_count      2696 non-null   int64
 7   comments_disabled  2696 non-null   bool
 8   ratings_disabled   2696 non-null   bool
 9   category_name      2646 non-null   object
 10  time_diff          2696 non-null   float64
 11  ldratio            2668 non-null   float64
dtypes: bool(2), datetime64[ns](2), float64(2), int64(4), object(2)
memory usage: 237.0+ KB
```

In [136]:
```
#Setting up GLM models
formula  = "time_diff~view_count+likes+dislikes+comment_count"
family_LM = sm.families.Gaussian()
model_LM = glm(formula = formula, data = train, family = family_LM).fit()
print(model_LM.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              time_diff   No. Observations:                10781
Model:                            GLM   Df Residuals:                    10776
Model Family:                Gaussian   Df Model:                            4
Link Function:               identity   Scale:                          362.68
Method:                          IRLS   Log-Likelihood:                -47064.
Date:                Sun, 13 Dec 2020   Deviance:                   3.9082e+06
Time:                        19:21:44   Pearson chi2:                 3.91e+06
No. Iterations:                     3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept        18.2052      0.192     95.043      0.000      17.830      18.581
view_count     3.064e-06   1.89e-07     16.236      0.000    2.69e-06    3.43e-06
likes          -5.93e-06   1.84e-06     -3.223      0.001   -9.54e-06   -2.32e-06
dislikes       9.884e-05   1.99e-05      4.964      0.000    5.98e-05       0.000
comment_count -8.503e-05   5.31e-06    -16.010      0.000   -9.54e-05   -7.46e-05
==============================================================================
```

In [140]:
```
pred_lm = model_LM.predict(test)
predictions = pd.DataFrame({'Pred_LM': pred_lm })
all_data = pd.concat([test["time_diff"], predictions], axis = 1)
all_data.head(10)
```

Out[140]:

	time_diff	Pred_LM
715	6.149167	18.639012
2926	13.603333	18.281253
5458	9.999167	18.352792
888	-5.002500	18.464267
3084	11.626389	19.103044
11696	28.058889	18.441075
21206	11.996389	32.008150
11904	34.249444	18.800683
10698	32.496944	18.475872
16373	21.696389	19.023703

To understand the model accuracy, we used the following formula to calculate the pseudo R-squared (Portugués, 2020):

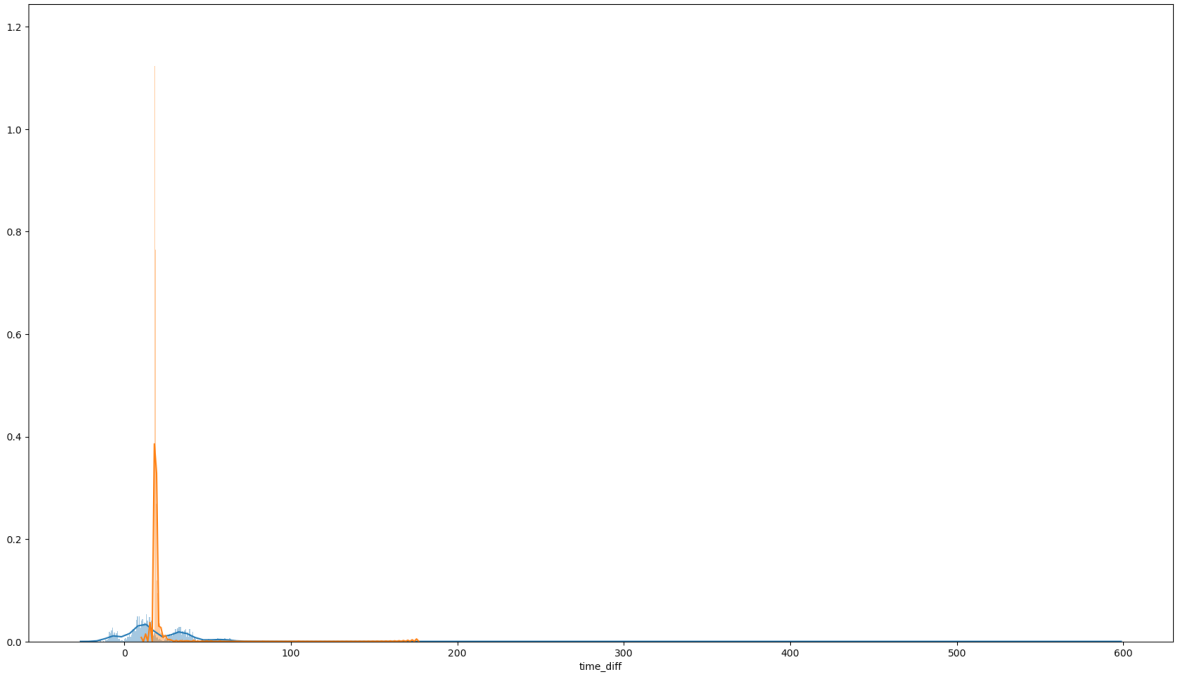
$$R^2 := 1 - \frac{D}{D_0} \overset{\text{linear model}}{=} 1 - \frac{\text{SSE}}{\text{SST}}.$$

In [142]:
```
model_LM.null_deviance
model_LM.deviance
print(100*((model_LM.null_deviance-model_LM.deviance)/model_LM.null_deviance))
fig, ax = plt.subplots(figsize=(21,12))
time = sns.distplot(test["time_diff"],bins = 1000)
time2 =sns.distplot(predictions,bins = 1000)
```

Out[142]: 4146264.9498166135

Out[142]: 3908239.008841418

5.740731570608458



A shown by both the R squared calculated (5.7%) and the distribution graph for both the prediction and test dataset. The model's performance is poor in predicting the outcome.

5. Conclusion and Recommendations

As stated in the beginning, we tried to provide insights to enhance the video content quality and increase their visibility. After conducting the analysis, we identified some key characteristics of trending videos in Japan, Russia, UK, and the US.

For Japan, content creators can tailor their YouTube content to the Entertainment and People & Blogs categories. According to the buzzword analysis, they can collaborate with local live shows in Japan to gain popularity. Alternative collaborations with local singers are desired for better content. A vigorous discussion convincingly helps attract more attention. The best time to publish a video seems to be 10 a.m., and our clients should reserve about four days before reaching trending.

Currently, in the UK, people are interested in sport (due to Series A and UEFA), entertainment, and gaming. Commentaries on football leagues, live streams, and live gameplays are recommended for content. The busy work-life means that around 5 p.m. is the best time to publishing a video. British people are slower to respond to recent video hence, Tuesday seems to be the best publishing day.

For the USA, comedy, people & blogs are good topics to choose from. These categories have high potential to be liked and comment on by viewers. Moreover, Americans seem to prefer videos related to their everyday lives. We suggest as content life related videos, awkward experiences confessions or comedic content. Our analysis indicates that videos designed this way have a better trending chance in the USA.

In Russia, publishing videos from 8 a.m. to 6 p.m. on Thursdays is suggested. Russians pay more attention to news and politics. Words such as “massacre” and “escape” are like to grab their attention. In addition, videos on candy and fashion are popular. Thus, Russian YouTubers should consider videos approaching those topics.

- Limitation and next steps

Predicting the time for a video to reach trending is difficult for the given dataset.

- News and special events such as the Nagorno-Karabakh war and Series A matches have high impacts on both what videos are selected and the duration it takes to reach trending. Our study does not conduct predictions on those events, as the data is limited. However, understanding what people are interested in now can be helpful when advising our clients.
- The selection process of the dataset is biased (survival bias). With the given dataset, we can only focus the analysis on videos that have reached trending, and the impact determined by our analysis may be overestimated or underestimated due to lack of information on non-trending videos. For further analysis, we would expand our analysis to a stratified randomly sampled dataset to eliminate the survival bias.
- The OLS model developed is insufficient to predict our desired outcomes. More sophisticated models should be used, like decision tree analysis.

- We only considered 12 countries and selected 4 for the in-depth analysis. This lowers our scope of options for international trends.

6. References

Big Hit Labels, 2020. [online] Youtube.com. Available at: <https://www.youtube.com/watch?v=-5q5mZbe3V8> [Accessed 4 December 2020].

Football-italia.net. 2020. Serie A Previews | Fixtures | Results | Football Italia. [online] Available at: <https://www.football-italia.net/SerieA> [Accessed 2 December 2020].

IQBAL, M., 2020. Tiktok Revenue And Usage Statistics (2020). [online] Business of Apps. Available at: <https://www.businessofapps.com/data/tik-tok-statistics/> [Accessed 5 December 2020].

Keith, F., 2020. Dortmund's New Boss Has Already Addressed Jadon Sancho To Man United Links. [online] Manchester Evening News. Available at: <https://www.manchestereveningnews.co.uk/sport/football/transfer-news/dortmund-jadon-sancho-man-utd-19450614> [Accessed 2 December 2020].

Mohit, 2020. Youtube Video Trending EDA And NLP. [online] Kaggle.com. Available at: <https://www.kaggle.com/mohitmanjaria/youtube-video-trending-eda-and-nlp> [Accessed 5 December 2020].

Portugués, E., 2020. 5.5 Deviance | Notes For Predictive Modeling. [online] Bookdown.org. Available at: <https://bookdown.org/egarpor/PM-UC3M/glm-deviance.html> [Accessed 14 December 2020].

Seanews, 2020. [online] Available at: <https://seanews.ru/en/2020/07/22/en-russia-imports-more-cocoa-and-chocolate-in-q1-2020/> [Accessed 5 December 2020].

Sharma, R., 2020. Youtube Trending Video Dataset (Updated Daily). [online] Kaggle.com. Available at: <https://www.kaggle.com/rsrishav/youtube-trending-video-dataset> [Accessed 4 December 2020].

Wellbeing People. 2018. Is The Traditional Working Hours Of 9Am To 5Pm Bad For Employee Wellbeing? | Wellbeing People. [online] Available at: <https://www.wellbeingpeople.com/2018/08/23/is-the-traditional-working-hours-of-9am-to-5pm-bad-for-employee-wellbeing/> [Accessed 4 December 2020].

Xbox, 2020. The All-New Xbox Series S | Xbox. [online] Xbox.com. Available at: <https://www.xbox.com/en-GB/consoles/xbox-series-s> [Accessed 10 December 2020].

In [34]:
```
import io
from IPython.nbformat import current
filepath = "Group Project F1.ipynb"

with io.open(filepath, 'r', encoding='utf-8') as f:
    nb = current.read(f, 'json')

word_count = 0
for cell in nb.worksheets[0].cells:
    if cell.cell_type == "markdown":
        word_count += len(cell['source'].replace('#', '').lstrip().split(' '))
print(word_count-22-208-60)
```

```
2167
```

In [ ]: