youtube-watch-history-analysis

April 24, 2023

1 YouTube Wrapped

I learned recently that Google allows its users to download complete data of our youtube account. This leads to interesting insights to be found in our streaming patter, like: ***Average daily watch time***, ***Favorite Video Category***, ***Favorite Channel***, etc. In order to explore these and other questions I decided to request my data and perform the present analysis.

The first step is requesting the YouTube watch history data which can be downloaded from Google Takeout. For performing data analysis we need the data in Json format.

1.1 Mounting G-Drive

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

1.2 Import requirements

[ ]:
```
from googleapiclient.discovery import build
import pandas as pd
import numpy as np
import seaborn as sns
import json
import matplotlib.pyplot as plt
import matplotlib.animation as anim
import dateutil
import random
from wordcloud import WordCloud
import requests
import time
import isodate
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

1.3 Read in JSON file from Google Tackout, convert to list, and build youtube API

For extracting the youtube video information from youtube API i took some references from here

[ ]:
```python
api_key = 'AIzaSyB1T-SPMB6PYg4jly7WB0kjFjeVCLJX9-I'
f = open(r'/content/drive/MyDrive/Colab Notebooks/Self practice projects/
 ↪Youtube/watch_history.json', encoding="UTF-8")
history = json.load(f)
history_list = []
total_videos = len(history)
for i in range(0,len(history)):
    if history[i]['header'] == 'YouTube':
        if 'titleUrl' in history[i]:
            video = history[i]['titleUrl'].split('=',1)[1]
            view_date = history[i]['time']
            if 'details' in history[i] and any(d.get('name') == 'From Google␣
 ↪Ads' for d in history[i]['details']):
            #The data is showing the count of videos including Ads, Hence we␣
 ↪remove the count of Ads to obtain accurate results.
                continue
            history_list.append(dict(
                watch_date = view_date,
                video_id = video
            ))

youtube = build('youtube', 'v3', developerKey=api_key)
```

The last line of code uses the `build()` function from the `googleapiclient.discovery` module to create a client object that can be used to interact with the YouTube Data API.

The `build()` function takes several arguments:

- `'youtube'` specifies the name of the API to use (in this case, the YouTube Data API).
- `'v3'` specifies the version of the API to use.
- `'developerKey=api_key'` specifies the API key to use for authentication. The resulting youtube object can then be used to make requests to the YouTube API, such as retrieving additional information about the videos in the user's watch history.

[ ]:
```python
print(len(history_list))
```

```
22132
```

2 Function to extract video data using Youtube API

[ ]:
```
def get_video_stats(youtube, sample_list):
    all_data = []
    all_ids = [sub['video_id'] for sub in sample_list]
    batched_ids = []
    n = 50
    for i in range(0,len(all_ids),n):
        batched_ids.append(all_ids[i:i + n])

    for i in range(len(batched_ids)):
        request = youtube.videos().list(
                    part='snippet,contentDetails,statistics',
                    id=batched_ids[i])

        response = request.execute()

        for i in range(len(response["items"])):
            data = dict(video_id = response["items"][i]["id"],
                    video_title = response["items"][i]["snippet"]['title'],
                    video_description =␣
 ↪response["items"][i]["snippet"]['description'],
                    published_at =␣
 ↪response["items"][i]["snippet"]['publishedAt'],
                    channel_id = response["items"][i]["snippet"]['channelId'],
                    category_id = response["items"][i]["snippet"]['categoryId'],
                    duration =␣
 ↪response["items"][i]["contentDetails"]['duration'],
                    favorite_count =␣
 ↪response["items"][i]["statistics"]['favoriteCount']
                    )
            if 'tags' in response["items"][i]["snippet"]:
             data['tag'] = response["items"][i]["snippet"]['tags']
            else:
             data['tag'] = 'NULL'

            if 'likeCount' in response["items"][i]["statistics"]:
             data['like_count'] =␣
 ↪response["items"][i]["statistics"]['likeCount']
            else:
             data['like_count'] = 'NULL'

            if 'commentCount' in response["items"][i]["statistics"]:
             data['comment_count'] =␣
 ↪response["items"][i]["statistics"]['commentCount']
            else:
             data['comment_count'] = 'NULL'
            if 'viewCount' in response["items"][i]["statistics"]:
             data['view_count'] =␣
↪response["items"][i]["statistics"]['viewCount']
            else:
             data['view_count'] = 'NULL'
            if data['video_title'] is None:
                print(i)
            all_data.append(data)
    return all_data
```

This code defines a function `get_video_stats` that takes two arguments `youtube` and `sample_list`. The function performs the following steps:

- Extracts the video IDs from `sample_list` and stores them in `all_ids` list.
- Breaks down the `all_ids` list into batches of 50 IDs (if there are more than 50) and stores each batch in the `batched_ids` list.
- For each batch in `batched_ids`, the function sends a request to the YouTube API to get the video statistics (snippet, contentDetails, and statistics) for each video in the batch.
- For each video in the batch, the function extracts the relevant information (video_id, video_title, video_description, published_at, channel_id, category_id, duration, favorite_count, tags, like_count, comment_count, and view_count) from the API response and stores it in a dictionary called `data`.
- The `data` dictionary is then appended to the `all_data` list.
- Once all batches have been processed, the function returns the `all_data` list containing the video statistics for all videos in `sample_list`.

```
[ ]: video_stats = get_video_stats(youtube, history_list)
     #print(video_stats)
```

```
[ ]: video_data = pd.DataFrame(video_stats)
     video_data.head()
```

```
[ ]:       video_id  \
     0  DSgJ1sejWtw
     1  CqYY_vSBfIc
     2  SkKrzGvcaZs
     3  5TecXk1l8sM
     4  dXvp-xMIruM

               video_title  \
     0
     Karate Kid - The Jacket
     1                                                                   Ep 2. Who Is
     Vedha? | Vikram Vedha
     2        Virumaandi - Panchayat scene | Kamal Haasan | Napoleon | Pasupathy |
     Abhiramy | 4K [Eng Subs]
     3  Soorarai Pottru - Deleted Scene 3- Arivu Threatens Maara | Sudha Kongara |
     Suriya | 2D Entertain…
     4    Soorarai Pottru - Deleted Scene 7  - Friendship Song  | Sudha Kongara |
     Suriya| 2D Entertainment

          video_description  \
     0                                                                    Karaté Kid
     2010 about the Jacket.
     1  YNOT Studios' #vikramvedha  [2017]\nWritten & Directed by Pushkar &
     Gayatri\nProduced by S. Sash…
     2  Stream the full movie now on Amazon Prime Video:-\n
     https://bit.ly/VirumaandionPrimeVideo\n\nD…
     3  Check out  Suriya's "Soorarai Pottru" deleted scene of Arivu Threatening
     Maara (Suriya) on 2D En…
     4  Maara has finally started an airline service. His team is recruiting
     candidates for an air hoste…

                published_at               channel_id category_id duration  \
     0  2010-08-13T19:50:45Z  UCwJXE75BbFV-0W91iACnnGA           1  PT7M39S
     1  2023-03-31T18:36:54Z  UCqVDSxEb7MNfYvddpbri4TA           1  PT6M46S
     2  2021-01-28T12:30:11Z  UC_gXhnzeF5_XIFn4gx_bocg          22  PT5M20S
     3  2021-02-20T08:37:20Z  UCj6rqKA33Ywu2GTFRDxHhnA          10  PT1M41S
     4  2021-02-21T09:30:49Z  UCj6rqKA33Ywu2GTFRDxHhnA          10  PT1M25S

       favorite_count  \
     0              0
     1              0
     2              0
     3              0
     4              0

                        tag  \
     0      [veste, karate, karaté, kid, jacket, smith, jaden, jackie, chan, 2010,
     trailer, take, off, on]
     1  [tamil movie scenes, vikram vedha, madhavan tamil movies, madhavan mass
     scene, r madhavan, vijay…
     2  [Ulaganayagan Tube, Kamal Haasan, Virumaandi, Virumaandi Trailer, virumandi
     songs, kamal haasan …
     3  [2d music, 2d entertainment, surya emotional scenes, soorarai pottru
     emotional scene, soorarai p…
     4  [2d music, 2d entertainment, surya emotional scenes, soorarai pottru
     emotional scene, soorarai p…

       like_count comment_count view_count
     0     201932          4430   16229286
     1       4599            32     578485
     2     106158          1837    5948038
     3      90541          1267    2820657
```

```
4      70662          1003    1550984
```

```
[ ]: video_view = pd.DataFrame(history_list)
     video_view.head(3)
```

```
[ ]:                  watch_date     video_id
     0  2023-04-15T07:52:11.689Z  DSgJ1sejWtw
     1  2023-04-15T07:49:17.922Z  CqYY_vSBfIc
     2  2023-04-15T07:45:02.091Z  SkKrzGvcaZs
```

3 Merge the two dataframes

```
[ ]: final_data = video_view.merge(video_data, how='left', on='video_id')
```

4 How many viewed videos have been taken down?

```
[ ]: final_data[final_data['video_title'].isna()].count() #check for missing values
```

```
[ ]: watch_date           2331
     video_id             2331
     video_title             0
     video_description       0
     published_at            0
     channel_id              0
     category_id             0
     duration                0
     favorite_count          0
     tag                     0
     like_count              0
     comment_count           0
     view_count              0
     dtype: int64
```

5 Clean Data: Remove NAs and duplicates(not replays, identitcal times)

```
[ ]: final_data_clean = final_data.dropna().copy() #Remove NAs
     final_data_clean = final_data_clean.drop_duplicates(['watch_date']) #remove␣
      ↪duplicates
     final_data_clean.head(3)
```

```
[ ]:                  watch_date     video_id  \
     0  2023-04-15T07:52:11.689Z  DSgJ1sejWtw
```

```
3  2023-04-15T07:49:17.922Z  CqYY_vSBfIc
4  2023-04-15T07:45:02.091Z  SkKrzGvcaZs

     video_title  \
0                                                                        Karate
Kid - The Jacket
3                                                             Ep 2. Who Is
Vedha? | Vikram Vedha
4  Virumaandi - Panchayat scene | Kamal Haasan | Napoleon | Pasupathy | Abhiramy
| 4K [Eng Subs]

     video_description  \
0                                                                    Karaté Kid
2010 about the Jacket.
3  YNOT Studios' #vikramvedha  [2017]\nWritten & Directed by Pushkar &
Gayatri\nProduced by S. Sash…
4  Stream the full movie now on Amazon Prime Video:-\n
https://bit.ly/VirumaandionPrimeVideo\n\nD…

           published_at                channel_id category_id duration  \
0  2010-08-13T19:50:45Z  UCwJXE75BbFV-0W91iACnnGA           1  PT7M39S
3  2023-03-31T18:36:54Z  UCqVDSxEb7MNfYvddpbri4TA           1  PT6M46S
4  2021-01-28T12:30:11Z  UC_gXhnzeF5_XIFn4gx_bocg          22  PT5M20S

  favorite_count  \
0              0
3              0
4              0

                   tag  \
0       [veste, karate, karaté, kid, jacket, smith, jaden, jackie, chan, 2010,
trailer, take, off, on]
3  [tamil movie scenes, vikram vedha, madhavan tamil movies, madhavan mass
scene, r madhavan, vijay…
4  [Ulaganayagan Tube, Kamal Haasan, Virumaandi, Virumaandi Trailer, virumandi
songs, kamal haasan …

  like_count comment_count view_count
0     201932          4430   16229286
3       4599            32     578485
4     106158          1837    5948038
```

6 Convert data types

```
[ ]: numeric_cols = ['view_count','like_count','favorite_count','comment_count']
     final_data_clean[numeric_cols] = final_data_clean[numeric_cols].apply(pd.
      ↪to_numeric,
                                                                                     ␣
      ↪errors='coerce',axis = 1) # from object to number


     final_data_clean['watch_date'] = pd.to_datetime(final_data_clean['watch_date'],
                                                     infer_datetime_format=True) #␣
      ↪from object to Date
     final_data_clean['published_at'] = pd.
      ↪to_datetime(final_data_clean['published_at'],
                                                       infer_datetime_format=True)  #␣
      ↪from object to Date

     final_data_clean['duration_sec'] = final_data_clean['duration'].apply(lambda x:␣
      ↪isodate.parse_duration(x)) # new column for Duration in seconds

     final_data_clean['duration_sec'] = final_data_clean['duration_sec'].
      ↪astype('timedelta64[s]') # conversion from int to type seconds
     final_data_clean.dtypes
```

```
[ ]: watch_date           datetime64[ns, UTC]
     video_id                          object
     video_title                       object
     video_description                 object
     published_at         datetime64[ns, UTC]
     channel_id                        object
     category_id                       object
     duration                          object
     favorite_count                   float64
     tag                               object
     like_count                       float64
     comment_count                    float64
     view_count                       float64
     duration_sec                     float64
     dtype: object
```

```
[ ]: final_data_clean.isnull().any()
```

```
[ ]: watch_date           False
     video_id             False
     video_title          False
     video_description    False
     published_at         False
```

```
channel_id          False
category_id         False
duration            False
favorite_count      False
tag                 False
like_count           True
comment_count        True
view_count           True
duration_sec        False
dtype: bool
```

7 What is my favorite Category?

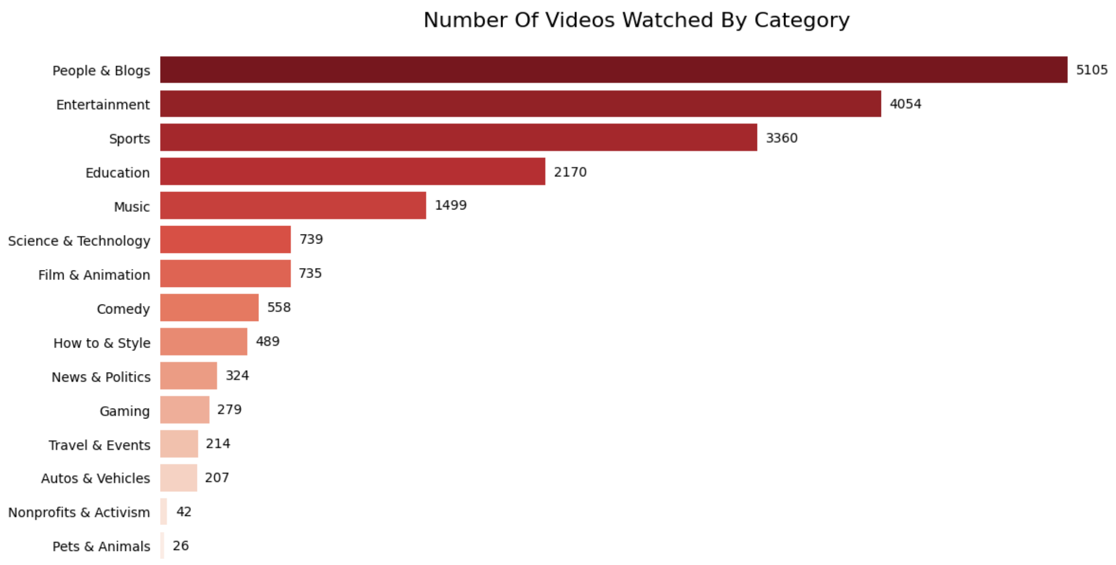
7.0.1 Replace default numerical representation of category to word.

You can get the category names list from here.

```
[ ]: final_data_clean['category_id'] = final_data_clean['category_id'].replace(['2',␣
      ↪'1','10','15','17','18','19','20','21','22','23','24','25',                    ␣
      ↪      '26','27','28','29','30','31','32','33','34','35','36','37',            ␣
      ↪      '38','39','40','41','42','43','44'],[
                     'Autos & Vehicles','Film & Animation', 'Music', 'Pets &␣
      ↪Animals', 'Sports', 'Short Movies',
                     'Travel & Events', 'Gaming', 'Videoblogging', 'People & Blogs',␣
      ↪'Comedy', 'Entertainment',
                     'News & Politics', 'How to & Style', 'Education', 'Science &␣
      ↪Technology', 'Nonprofits & Activism', 'Movies',
                     'Anime/Animation', 'Action/Adventure', 'Classics','Comedy',␣
      ↪'Documentary', 'Drama', 'Family', 'Foreign',
                     'Horror', 'Sci-Fi/Fantasy', 'Thriller', 'Shorts', 'Shows',␣
      ↪'Trailers']) #replace category id with category name
```

```
[ ]: final_data_groupedby_category = final_data_clean.
      ↪groupby(['category_id'])['category_id'].size().reset_index(name='counts') ␣
      ↪#group by category
     final_data_groupedby_category = final_data_groupedby_category.
      ↪sort_values(by=['counts'],ascending=False).reset_index(drop=True) #sort by␣
      ↪counts


     science_technology_educational_videos = final_data_clean[np.
      ↪logical_or(final_data_clean['category_id'] == 'Science & Technology' ,␣
      ↪final_data_clean['category_id'] == 'Education')]
     pd.set_option('display.max_colwidth', 100)
```

```
[ ]: fig = plt.figure(figsize=(13,7))
     font = {'family': 'sans-serif','color':  'black','weight': 'normal','size': 16}
     color_palette = sns.color_palette('Reds_r', len(final_data_groupedby_category))

     ax = sns.barplot(y='category_id', x='counts',␣
      ↪data=final_data_groupedby_category, orient='horizontal',␣
      ↪palette=color_palette)

     plt.ylabel('', labelpad=20, fontdict=font)
     plt.xlabel('', labelpad=20, fontdict=font)
     plt.title('Number Of Videos Watched By Category', fontdict=font, pad=20)

     # Add bar labels
     for i in ax.containers:
             ax.bar_label(i, padding=6)

     plt.tick_params(axis='x', which='both', bottom=False, top=False,␣
      ↪labelbottom=False)
     plt.tick_params(axis='y', which='both', right=False, left=False, labelleft=True)

     for pos in ['right', 'top', 'bottom', 'left']: # remove the frame
         plt.gca().spines[pos].set_visible(False)

     plt.show()
```



8 Whats my favorite day to watch YouTube?

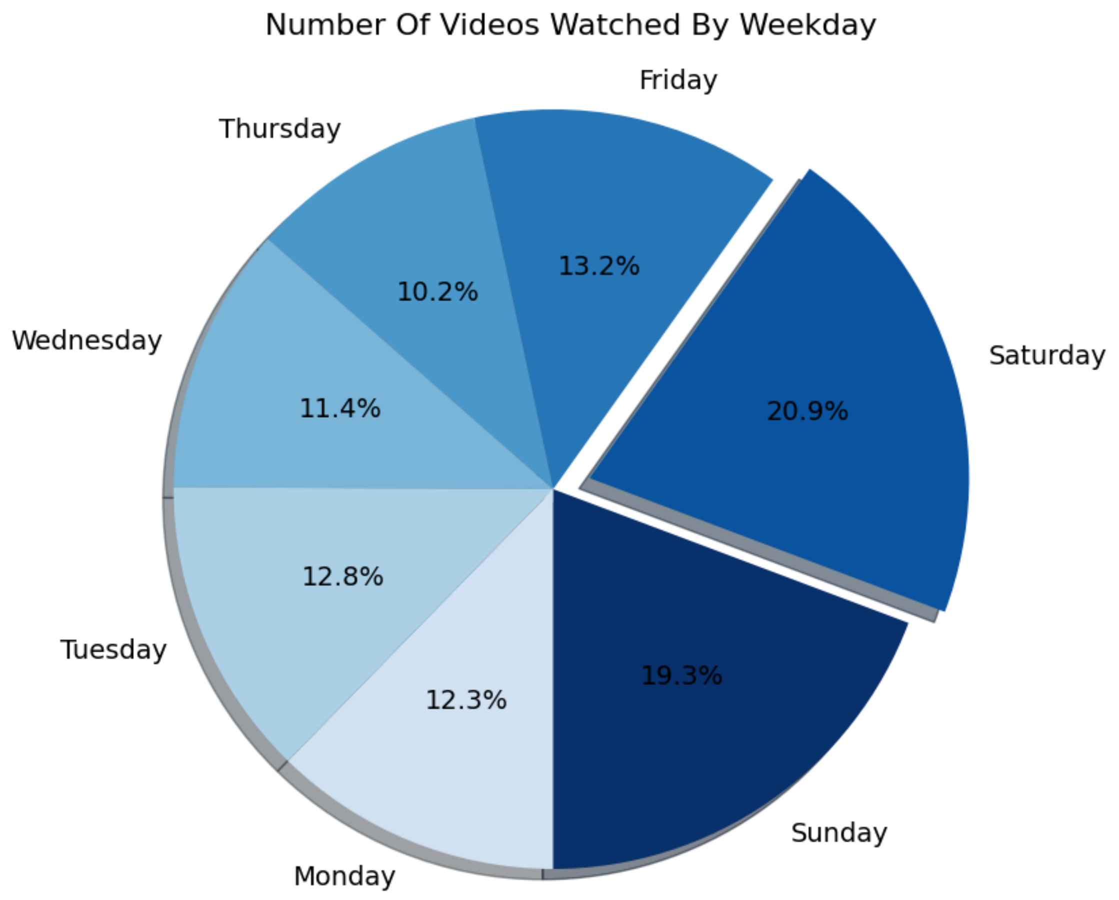
```
[ ]: day_data = final_data_clean.copy()
     day_data['watch_date'] = day_data['watch_date'].dt.day_name()
```

```
[ ]: days = [ 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',␣
      ↪'Sunday']

     final_data_groupedby_watch_day = day_data.groupby(['watch_date'])['watch_date'].
      ↪size().reset_index(name='count')
     final_data_groupedby_watch_day['watch_date'] = pd.
      ↪Categorical(final_data_groupedby_watch_day['watch_date'],
                                                                          categories=days,␣
      ↪ordered=True)
     final_data_groupedby_watch_day = final_data_groupedby_watch_day.
      ↪sort_values('watch_date')
```

```
[ ]: final_data_groupedby_watch_day
```

```
[ ]:   watch_date  count
     1     Monday   2438
     5    Tuesday   2528
     6  Wednesday   2253
     4   Thursday   2019
     0     Friday   2612
     2   Saturday   4135
     3     Sunday   3816
```

```
[ ]: import matplotlib.cm as cm

     fig = plt.figure(figsize=(8,8))
     font = {'color':  'black','weight': 'normal','size': 16}
     colors = cm.Blues(np.linspace(0.2, 1, len(final_data_groupedby_watch_day)))
     counts = final_data_groupedby_watch_day['count']
     labels = final_data_groupedby_watch_day['watch_date']

     # Get the index of the slice with the highest percentage
     max_index = counts.argmax()
     explode = [0] * len(counts)
     explode[max_index] = 0.1

     plt.pie(counts, labels=labels, colors=colors, autopct='%1.1f%%',␣
      ↪startangle=-90, textprops={'fontsize': 14}, explode=explode, shadow=True,␣
      ↪counterclock=False)

     plt.title('Number Of Videos Watched By Weekday', pad=20, fontdict=font)
     plt.axis('equal')
plt.show()
```



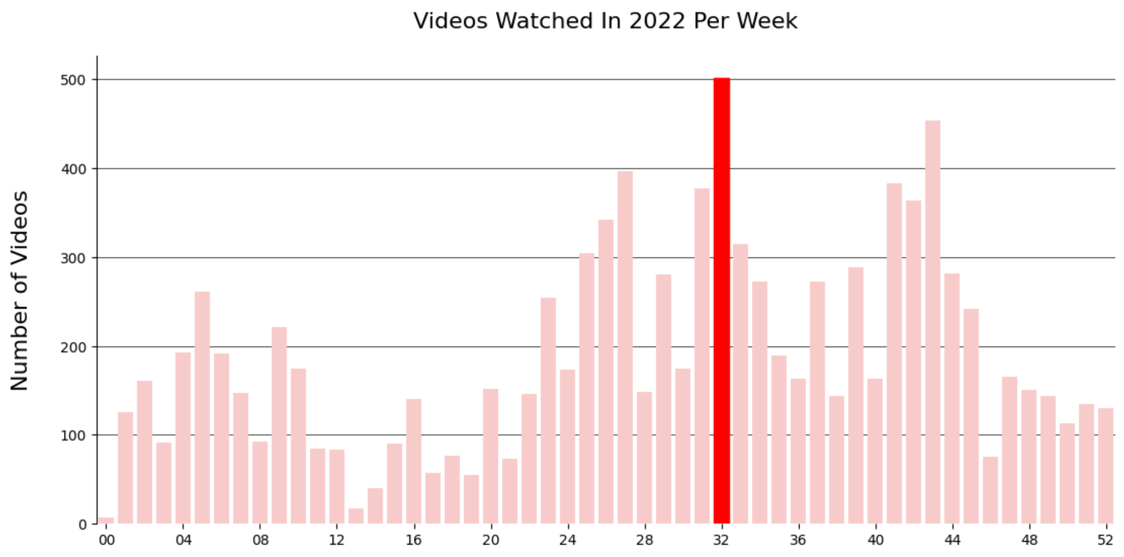
9 What week did I watch more videos last year?

[ ]:
```
copy = final_data_clean.copy()
ttone = copy[final_data_clean['watch_date'].dt.year == 2022]
by_week = ttone.sort_values(by='watch_date',ascending=True).
 ↪reset_index(drop=True)
by_week['week_num'] = by_week['watch_date'].dt.strftime('%U') #get week number
by_week.head(1)
```

[ ]:
```
                              watch_date     video_id  \
0 2022-01-01 09:52:27.060000+00:00  1UqQhQ7_Mm8

video_title  \
0  Vadivelu Bus Comedy | Aadhavan Comedy Scenes | Vadivelu Comedy | KalaignarTV
```

```
Movies

     video_description  \
0  Watch Vadivelu Bus comedy scene from the movie Aadhavan. For More latest
Tamil Movies, Subscribe…

               published_at                channel_id       category_id  \
0 2020-09-14 12:30:03+00:00  UC8-dWVwoMQNOqIdW04mIHPQ  Film & Animation

   duration  favorite_count  \
0  PT11M53S             0.0

                   tag  \
0  [Full length Tamil movies, Latest Tamil Films, Tamil movies online, Cult
hits, Super hit Tamizh …

   like_count  comment_count  view_count  duration_sec week_num
0     18340.0          136.0   2914912.0         713.0       00
```

[ ]:
```python
grouped_by_week = by_week.groupby(['week_num'])['week_num'].size().
 ↪reset_index(name='videos_per_week')
fig = plt.figure(figsize=(13,6))
watch_trend = sns.barplot(y='videos_per_week',␣
 ↪x='week_num',data=grouped_by_week,zorder=3, alpha=1, color='#FFC4C4')
highest_count = grouped_by_week['videos_per_week'].max()
for i in range(len(watch_trend.patches)):
    if watch_trend.patches[i].get_height() == highest_count:
        watch_trend.patches[i].set_color('#FF0000')

plt.title('Videos Watched In 2022 Per Week',pad=20,fontdict=font)
plt.ylabel('Number of Videos',labelpad=20,fontdict=font)
plt.xlabel('',labelpad=20,fontdict=font)
plt.grid(axis='y', color='black', linewidth=.5, zorder=0)
plt.xticks(range(0,53,4))

plt.tick_params(axis='x', which='both', bottom=True,
                top=False, labelbottom=True)

for pos in ['top', 'bottom','right']:
    plt.gca().spines[pos].set_visible(False)

plt.show()
```

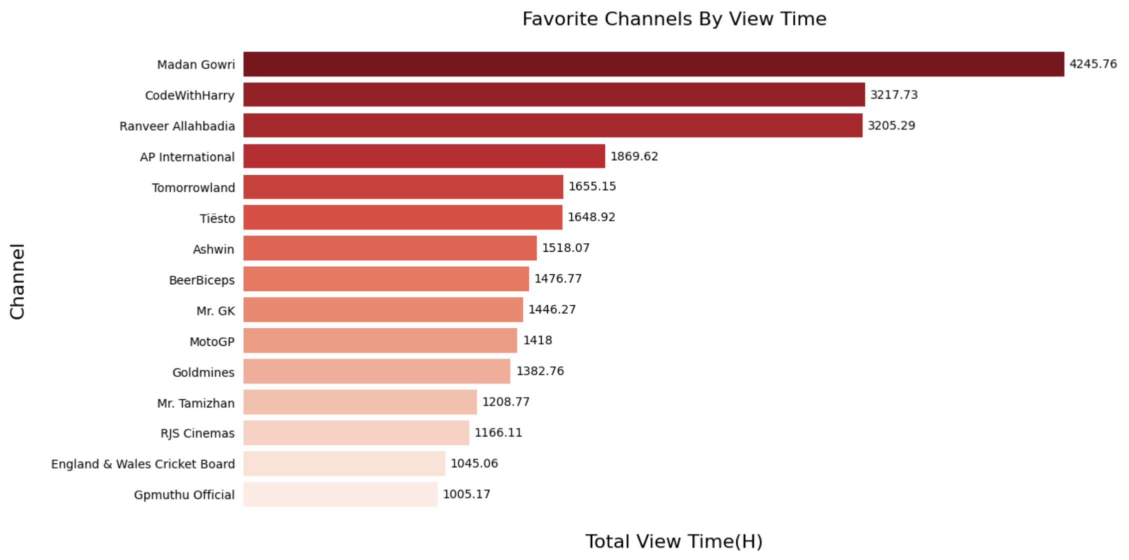


10 Favourite youtube channel

[ ]:
```
final_data_groupedby_channel =␣
↪final_data_clean[final_data_clean['duration_sec'] < 14400].copy()
final_data_groupedby_channel['duration_min'] =␣
↪final_data_groupedby_channel['duration_sec'].div(60).round(2)
final_data_groupedby_channel = final_data_groupedby_channel.
↪groupby(['channel_id'])['duration_min'].sum().reset_index(name='sum')
final_data_groupedby_channel = final_data_groupedby_channel.
↪sort_values(by=['sum'],ascending=False).reset_index(drop=True)
final_data_groupedby_channel =␣
↪final_data_groupedby_channel[final_data_groupedby_channel['sum'] > 1000]
final_data_groupedby_channel

def get_channel_title(cid):
    request1 = youtube.channels().list(
                    part='snippet,contentDetails,statistics',
                    id=cid)

    response = request1.execute()
    title = response['items'][0]['snippet']['title']
    return title

final_data_groupedby_channel['channel_title'] =␣
↪final_data_groupedby_channel['channel_id'].apply(get_channel_title)
final_data_groupedby_channel.head(2)
```

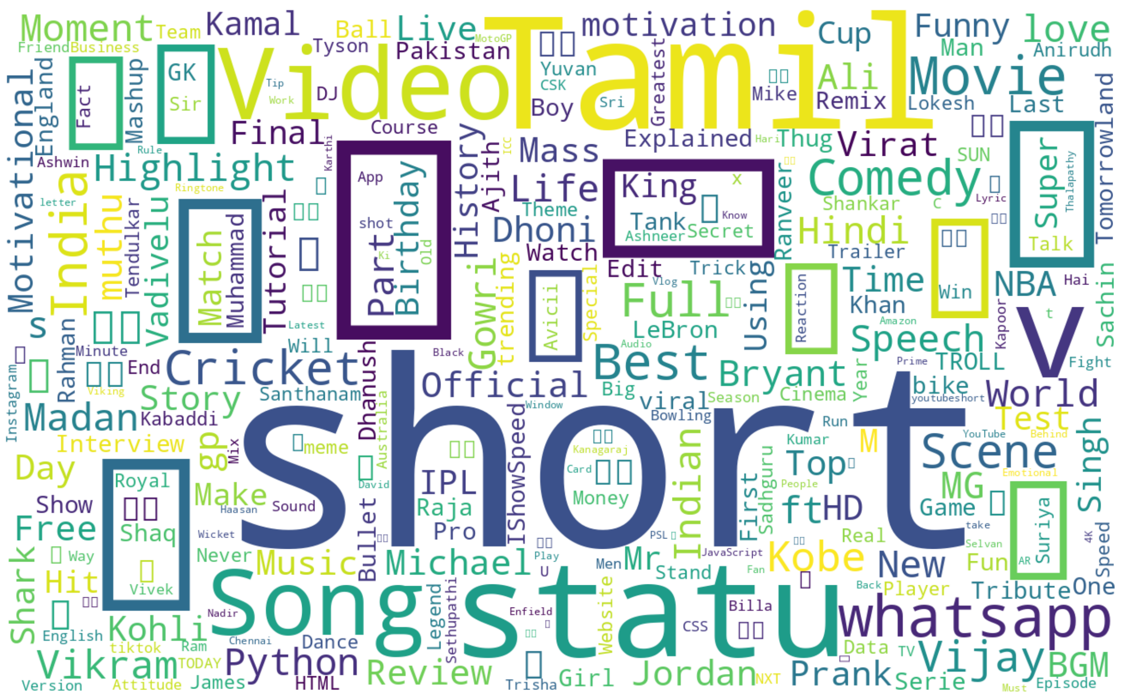
[ ]:
```
                 channel_id      sum  channel_title
0  UCY6KjrDBN_tIRFT_QNqQbRQ  4245.76    Madan Gowri
1  UCeVMnSShP_Iviwkknt83cww  3217.73  CodeWithHarry
```

[ ]:
```
fig = plt.figure(figsize=(13,7))
color_palette = sns.color_palette('Reds_r', len(final_data_groupedby_category))
watch_trend = sns.barplot(y='channel_title',␣
 ↪x='sum',data=final_data_groupedby_channel,orient='horizontal',palette=color_palette)
plt.title('Favorite Channels By View Time',pad=20,fontdict=font)
plt.ylabel('Channel',labelpad=20,fontdict=font)
plt.xlabel('Total View Time(H)',labelpad=20,fontdict=font)
plt.xticks(rotation=80)
for i in watch_trend.containers:
        watch_trend.bar_label(i,padding=4,)

plt.tick_params(axis='x', which='both', bottom=False,
                top=False, labelbottom=False)
plt.tick_params(axis='y', which='both', right=False,
                left=False, labelleft=True)

for pos in ['right', 'top', 'bottom', 'left']:
    plt.gca().spines[pos].set_visible(False)

plt.show()
```



11 What words are most common in the title of videos I have watched?

```
[ ]: stop_words = set(stopwords.words('english'))
     stop_words.update(['short','statu','GTI','VW','Thing'])
     final_data_clean['title_no_stopwords'] = final_data_clean['video_title'].
      ↪apply(lambda x:[item for item in str(x).split()
                                                                              ␣
      ↪        if item not in stop_words]).copy()

     all_words = list([a for b in final_data_clean['title_no_stopwords'].tolist()␣
      ↪for a in b])
     all_words_str = ' '.join(all_words)

     def plot_cloud(wordcloud):
         fig = plt.figure(figsize=(30,20))
         plt.imshow(wordcloud)
         plt.axis("off");

     wordcloud = WordCloud(width=1300, height=800,
                           background_color='white',
                           random_state=10,
                           max_words=300,
                           contour_width=3,
                           collocations=False).generate(all_words_str)
     plot_cloud(wordcloud)
```


short
Tamil
Video
Song
statu
whatsapp
Cricket
Official
Comedy
Movie
India
Highlight
Motivational
motivation
Vijay
Full
Best
Speech
Scene
Virat
Hindi
Dhoni
King
Review
Python
Vikram
Kohli
Kobe
Jordan
Prank
Bryant
World
Singh
NBA
Michael
Music
Madan
Story
Funny
love
Moment
Kamal
Live
Pakistan
Cup
Ali
Remix
Mass
Life
History
Tutorial
Birthday
Match
Time
Khan
IPL
Indian
Top
New
Free
Make
Day
Super
Tomorrowland
Sachin