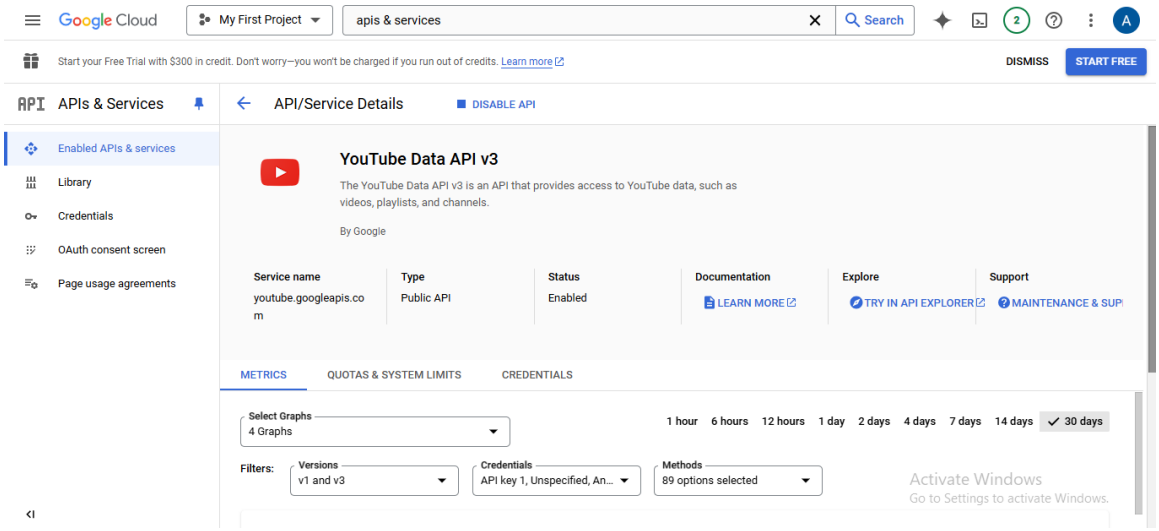
3. Social Network API

Youtube Vedios Dataset

Src: From Youtube using YouTube Data API v3 from Gogole cloud



Description:

The YouTube Video Dataset contains information about various videos collected from YouTube channels. It includes features such as video title, description, publish date, publish time, likes, views, category, and channel ID. This data can be used for analysis and modeling related to video performance, audience engagement, and content trends.

Data Collection and Understanding:

1. Attributes / Feature Description

Column Name	Description	Data Type	Attribute Type
index	The index number for each entry.	int64	Discrete
title	The title of the video.	object	Nominal
description	The description of the video.	object	Nominal
publish_date	The date when the video was published.	object	Nominal
publish_time	The time when the video was published.	object	Nominal
likes	The number of likes the video has received.	int64	Numeric (Ratio-Scaled)

views	The number of views the video has received.	int64	Numeric (Ratio-Scaled)
category	The category the video belongs to [news , sport .. etc]	object	Nominal
channel_id	The ID of the channel that published the video.	object	Nominal

2. Dataset Collection : Collected YouTube video data using the YouTube Data API and saved it to `youtube_videos.csv`

In this Part I collect data using Social Network API and "Python":

1. Import Libraries used in interact with youtube API

```
[1]: from googleapiclient.discovery import build #to interact with YouTube Data API
     import pandas as pd
```

&

download nedded librarys

```
[1]:
pip install google-api-python-client
```

```
   ---------------------------------------- - 12.1/12.6 MB 630.7 kB/s eta 0:00:01
   ---------------------------------------  12.3/12.6 MB 634.2 kB/s eta 0:00:01
   ---------------------------------------  12.3/12.6 MB 634.2 kB/s eta 0:00:01
   ---------------------------------------- 12.6/12.6 MB 629.8 kB/s eta 0:00:00
Downloading google_api_core-2.23.0-py3-none-any.whl (156 kB)
Downloading google_auth-2.36.0-py2.py3-none-any.whl (209 kB)
Downloading google_auth_httplib2-0.2.0-py2.py3-none-any.whl (9.3 kB)
Downloading httplib2-0.22.0-py3-none-any.whl (96 kB)
Downloading uritemplate-4.1.1-py2.py3-none-any.whl (10 kB)
Downloading googleapis_common_protos-1.66.0-py2.py3-none-any.whl (221 kB)
Downloading proto_plus-1.25.0-py3-none-any.whl (50 kB)
Downloading rsa-4.9-py3-none-any.whl (34 kB)
Installing collected packages: uritemplate, rsa, proto-plus, httplib2, googleapis-common-proto
s, google-auth, google-auth-httplib2, google-api-core, google-api-python-client
Successfully installed google-api-core-2.23.0 google-api-python-client-2.154.0 google-auth-2.3
6.0 google-auth-httplib2-0.2.0 googleapis-common-protos-1.66.0 httplib2-0.22.0 proto-plus-1.25.
0 rsa-4.9 uritemplate-4.1.1
Note: you may need to restart the kernel to use updated packages.
```

2. Setting YouTube Data API Client: set up and authenticate a YouTube Data API client using an API key generated from the Google Cloud Console. **build** from the googleapiclient.discovery module used to create an instance of the YouTube Data API client.

```
[2]: # YouTube API credentials
     # generated on  Google Cloud Console for youtube service
     api_key = 'AIzaSyBEfx6qOYZIovmdhQZk7mcBYIt8zhMpmtQ'

     # Build the YouTube Data API client
     # 'youtube' is the name of the API service
     # 'v3' is the version of the API
     # 'developerKey' is the API key used for authenticating requests
     youtube = build('youtube', 'v3', developerKey=api_key)

     # Print the youtube object to see if it was created successfully
     print(youtube)

     <googleapiclient.discovery.Resource object at 0x00000192123C0980>
```

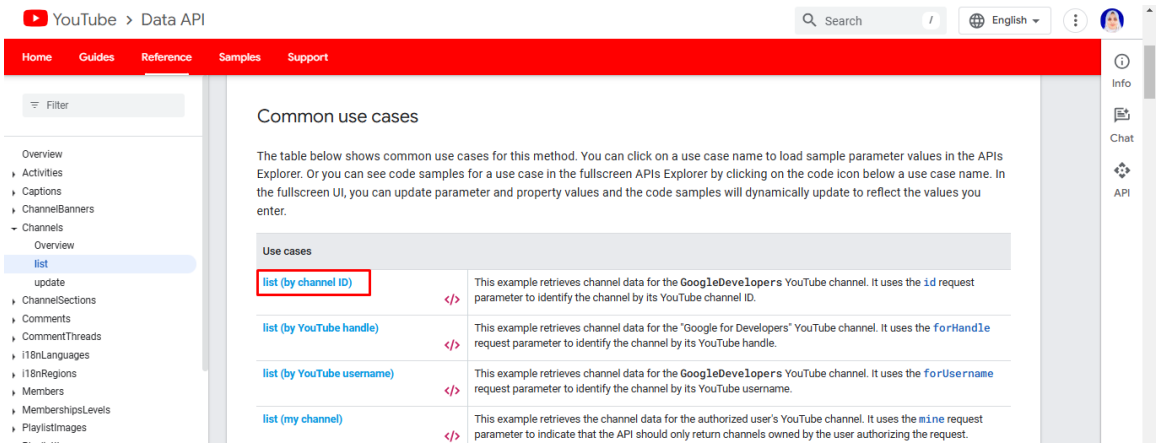
3. Define channels to read from: specify channels to read vedios data from it in a dictionary {key is categoy : value is channel name}.

```
[3]: # specific channels for several categories
     channels = {
         'news': ['BBCNews' ,'NBCNews'],
         'education': ['FITiug'],
         'cooking': ['gordonramsay'],
         'sport': ['SkatingISU'],
         'children': ['Osratouna']
     }
     channels

[3]: {'news': ['BBCNews', 'NBCNews'],
      'education': ['FITiug'],
      'cooking': ['gordonramsay'],
      'sport': ['SkatingISU'],
      'children': ['Osratouna']}
```

4. Get Channel id: use channel name to get id .. send request to list channel with specific name

```
[4]: # get the channel ID using the username
     def get_channel_id(username):
         # Create a request to YouTube Data API to list channels
         request = youtube.channels().list(
             part='id',  # only retern id  of the channel resource
             forUsername=username  #username -  which fetch there ID
         )
         # request excuction
         response = request.execute()
         # if the response contains any items (channels)
         if response['items']:
             # channel ID of the first item in the response
             return response['items'][0]['id']
         # if no items return None
         return None
```

All apis function I used available in **Youtube Data API Documentaition**



Required parameters	
part	string The part parameter specifies a comma-separated list of one or more channel resource properties that the API response will include. If the parameter identifies a property that contains child properties, the child properties will be included in the response. For example, in a channel resource, the contentDetails property contains other properties, such as the uploads properties. As such, if you set part=contentDetails, the API response will also contain all of those nested properties. The following list contains the part names that you can include in the parameter value: • auditDetails • brandingSettings • contentDetails • contentOwnerDetails • id • localizations • snippet • statistics • status • topicDetails

Filters (specify exactly one of the following parameters)	
categoryId	string This parameter has been deprecated. The categoryId parameter specified a YouTube guide category and could be used to request YouTube channels associated with that category.
forHandle	string The forHandle parameter specifies a YouTube handle, thereby requesting the channel associated with that handle. The parameter value can be prepended with an @ symbol. For example, to retrieve the resource for the "Google for Developers" channel, set the forHandle parameter value to either GoogleDevelopers or @GoogleDevelopers.
forUsername	string The forUsername parameter specifies a YouTube username, thereby requesting the channel associated with that username.
id	string The id parameter specifies a comma-separated list of the YouTube channel ID(s) for the resource(s) that are being retrieved. In a channel resource, the id property specifies the channel's YouTube channel ID.

Response return

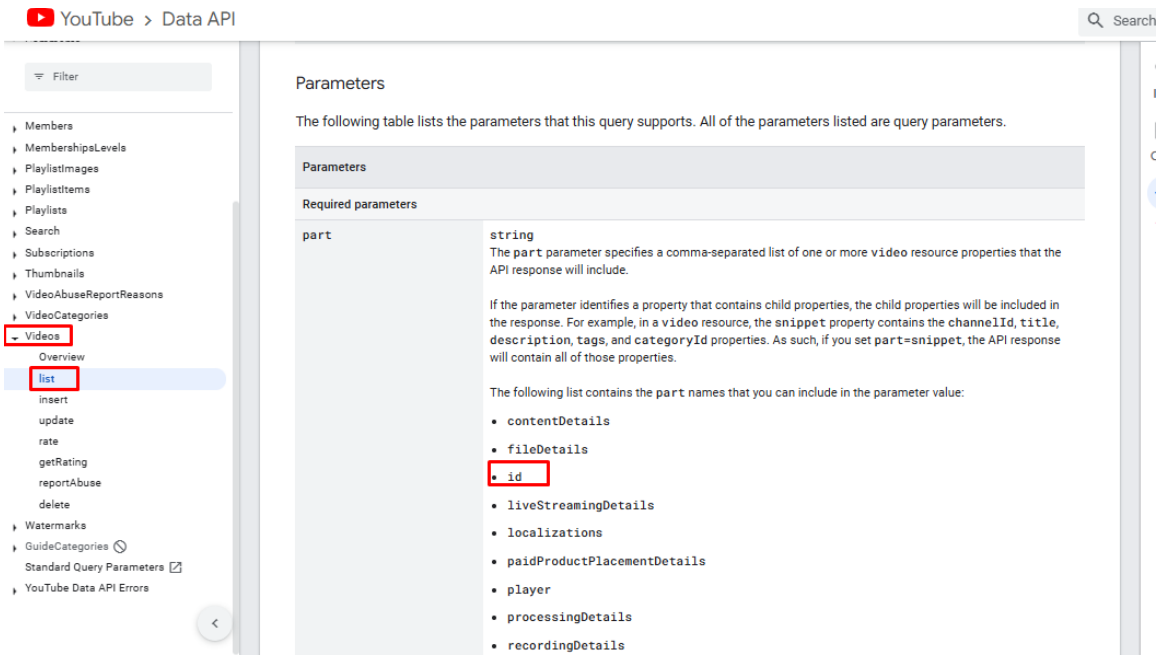
```
get_channel_id('FITiug')
```

```
{'kind': 'youtube#channelListResponse', 'etag': '3T44DmxSwIhLjc3eEvbMAzErQt8', 'pageInfo': {'totalResults': 1, 'resultsPerPage': 5}, 'items': [{'kind':
'youtube#channel', 'etag': 'UDe6YWMbHGisYjFf5FBa_b35VYY', 'id': 'UCpifcArFGkbQMpsZzj29eFg'}]}
```

[5]: `'UCpifcArFGkbQMpsZzj29eFg'`

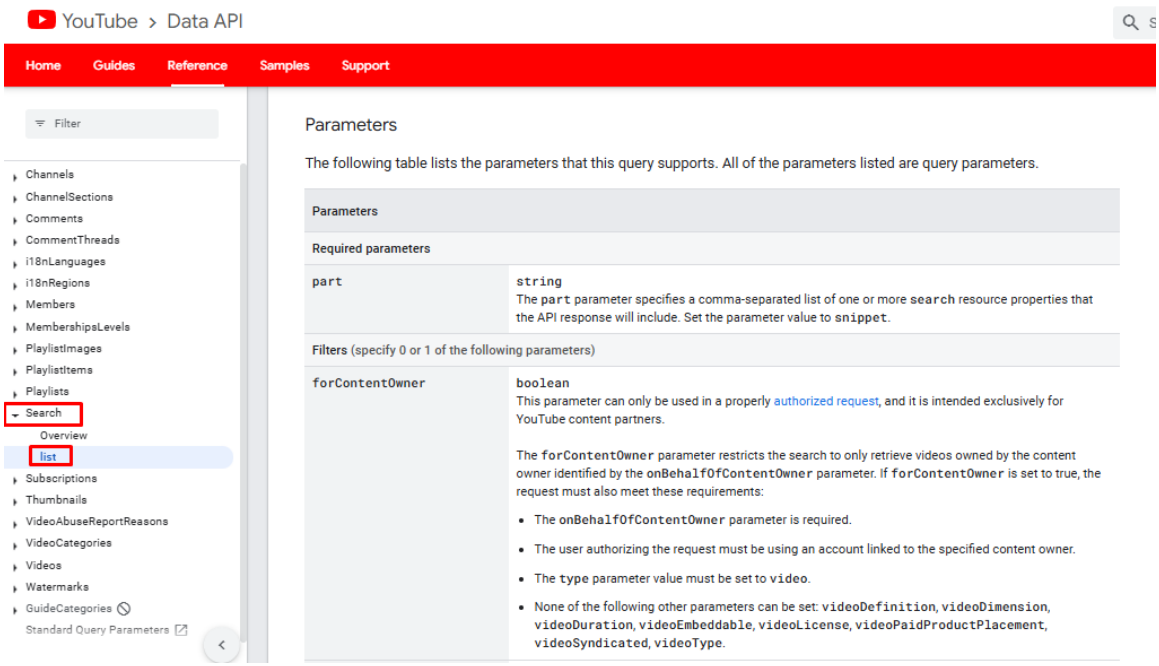
5. Get Vedio Details: for each vedio collected from chnnel list data about it ‘statistics for likes , views’ & ‘snippet for title, description , publiseted details'

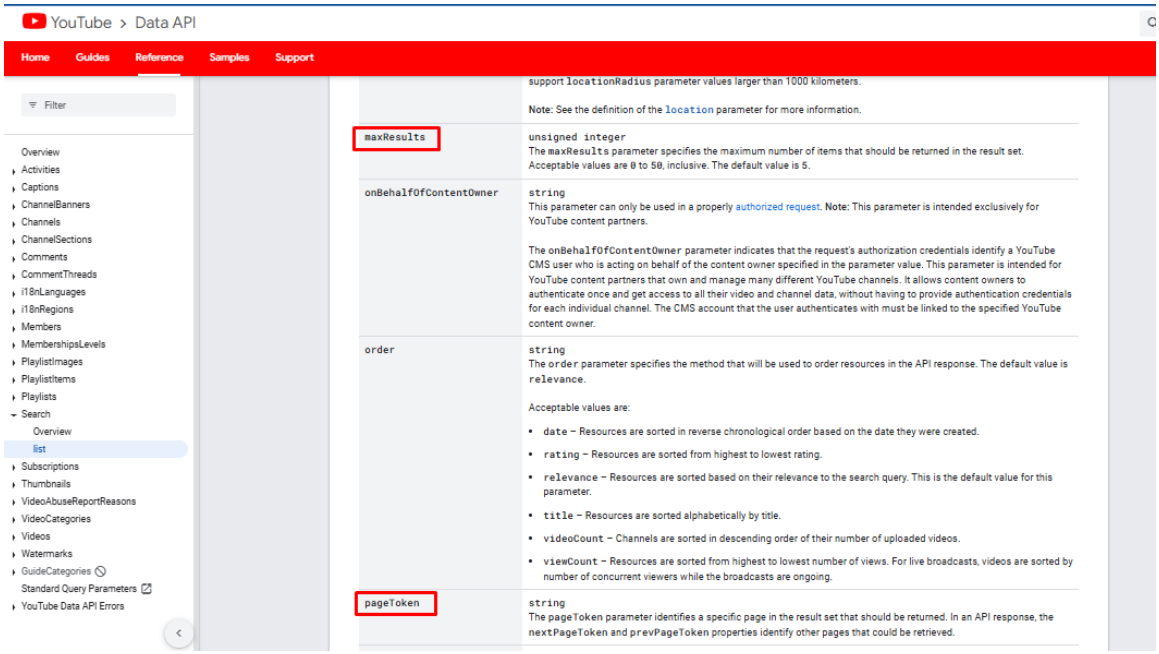
[5]:
```python
#  get the details of a video using video ID
def get_video_details(video_id):
    #  request to the YouTube Data API to list videos
    request = youtube.videos().list(
        part='snippet,statistics',  # retrieve: snippet and statistics
        id=video_id  # which vedion get details
    )
    # request excution
    response = request.execute()
    # if the response contains any items (videos)
    if response['items']:
        #  first video in the response
        video = response['items'][0]
        # dictionary of video details
        published_at = video['snippet']['publishedAt']
        publish_date, publish_time = published_at.split('T')
        publish_time = publish_time.split('Z')[0]
        return {
            'title': video['snippet']['title'],  # title of the video
            'description': video['snippet']['description'],  # description of the video
            'publish_date': publish_date,
            'publish_time': publish_time,
            'likes': video['statistics'].get('likeCount', 0),  # The number of likes the video has received
            'views': video['statistics'].get('viewCount', 0)  # The number of views the video has received
        }
    # if no items return None
    return None
```



6. Collect Vedio: collect 200 vedio data from each channel using channel id retened using function in 4 step.

```
[6]: # collect all videos from a channel
def collect_videos(channel_id, max_results=200):
    video_list = []  # list to store video details
    next_page_token = None  # token for the next page of results

    # Loop until max_results or there are no more videos
    while len(video_list) < max_results:
        # request to the YouTube Data API to list videos for a channel
        request = youtube.search().list(
            part='id',  # id part of the video resource
            channelId=channel_id,  # want to collect videos form this channel
            maxResults=min(max_results - len(video_list), 50),  # Limit the number of results per request
            pageToken=next_page_token,  # Token for the next page of results (if any)
            type='video'  # want video results
        )
        # request excution
        response = request.execute()

        #for all items in the response
        for item in response['items']:
            # details of each video using the video ID
            video_details = get_video_details(item['id']['videoId'])
            # If video details ... append them to the video list
            if video_details:
                video_list.append(video_details)

        # token for the next page of results (if any)
        next_page_token = response.get('nextPageToken')
        # if no next page token, break
        if not next_page_token:
            break

    # list of collected video details
    return video_list
```





7. Collect data: collect vedios data from diferent channels using all privios functions

```
[7]: all_data = []  # list to store all the collected video data

# Loop through each category
for category, usernames in channels.items():
    for username in usernames: # category can contain many channels/user name
        print(f"Collecting data for : {username} channel in : {category} category")

        # get channel ID using the username
        channel_id = get_channel_id(username)

        # if the channel retrieved
        if channel_id:
            # extract videos from the channel using the channel ID
            videos = collect_videos(channel_id)

            # Loop through all collected videos
            for video in videos:
                # Add additional information to each video (category and channel ID)
                video['category'] = category
                video['channel_id'] = channel_id

                # append the video details to the all_data list
                all_data.append(video)
        else:
            # if the channel ID was not found
            print(f"Channel ID not found for username: {username}")
```

```
Collecting data for : BBCNews channel in : news category
Collecting data for : NBCNews channel in : news category
Collecting data for : FITiug channel in : education category
Collecting data for : gordonramsay channel in : cooking category
Collecting data for : SkatingISU channel in : sport category
Collecting data for : Osratouna channel in : children category
```

8. Finally store data in csv file using pandas DataFrame and reading info()

```
[8]: # Save data to a CSV file
df = pd.DataFrame(all_data)
df.index += 1  # index from 1
df.to_csv('youtube_videos.csv', index_label='index')
print("Data saved to youtube_videos.csv")
```

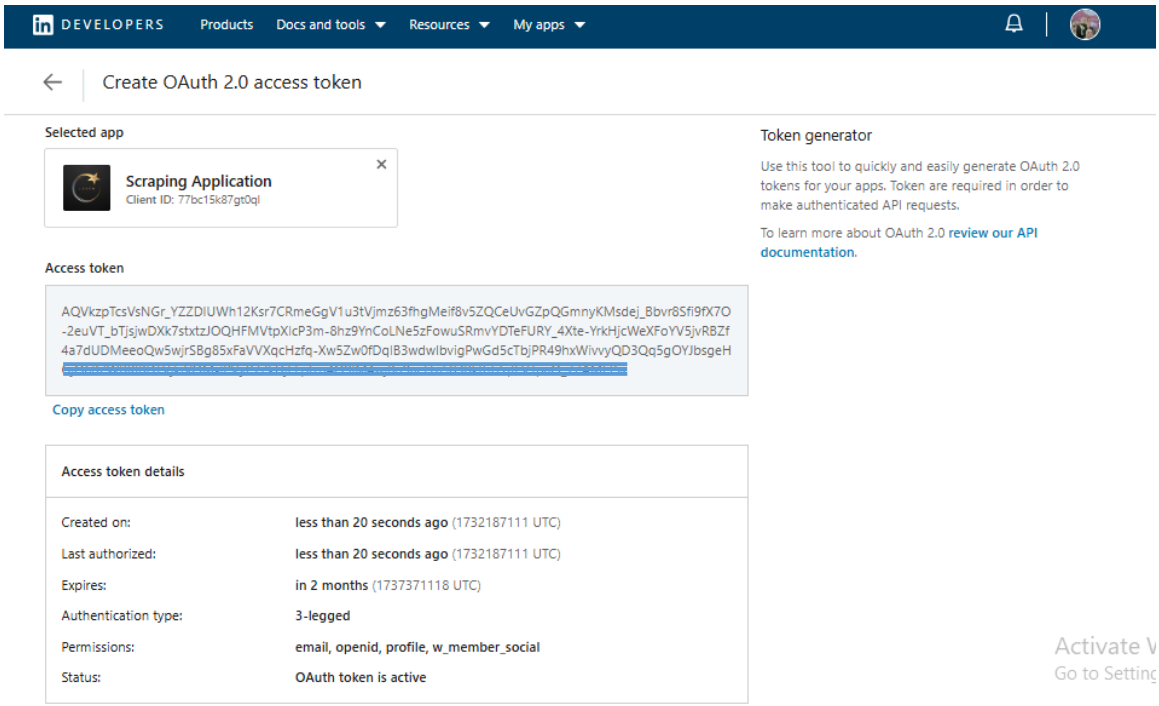
```
Data saved to youtube_videos.csv
```

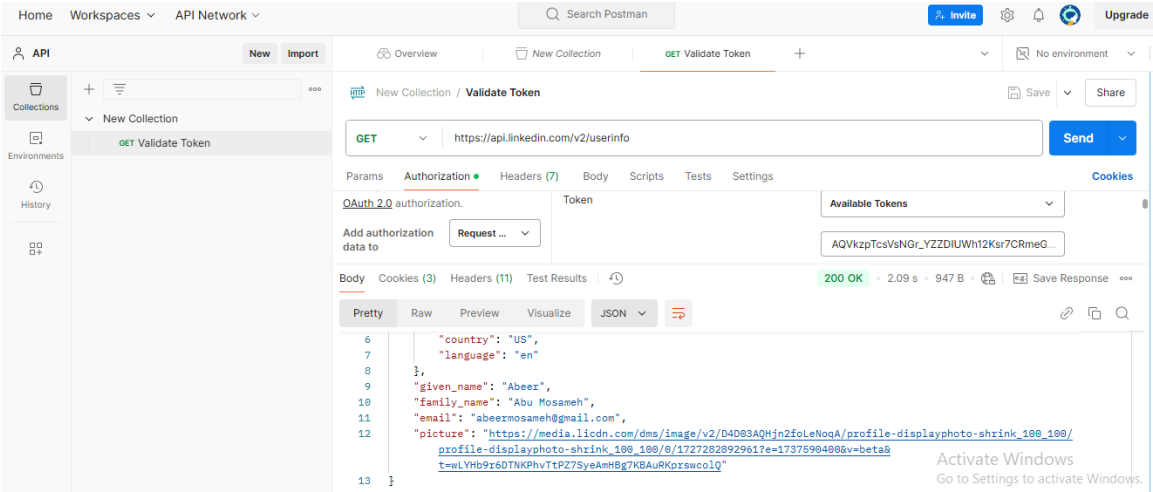
```
[9]: data = pd.read_csv('youtube_videos.csv')
     data.info() #info about data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   index         1200 non-null   int64
 1   title         1200 non-null   object
 2   description   1167 non-null   object
 3   publish_date  1200 non-null   object
 4   publish_time  1200 non-null   object
 5   likes         1200 non-null   int64
 6   views         1200 non-null   int64
 7   category      1200 non-null   object
 8   channel_id    1200 non-null   object
dtypes: int64(3), object(6)
memory usage: 84.5+ KB
```

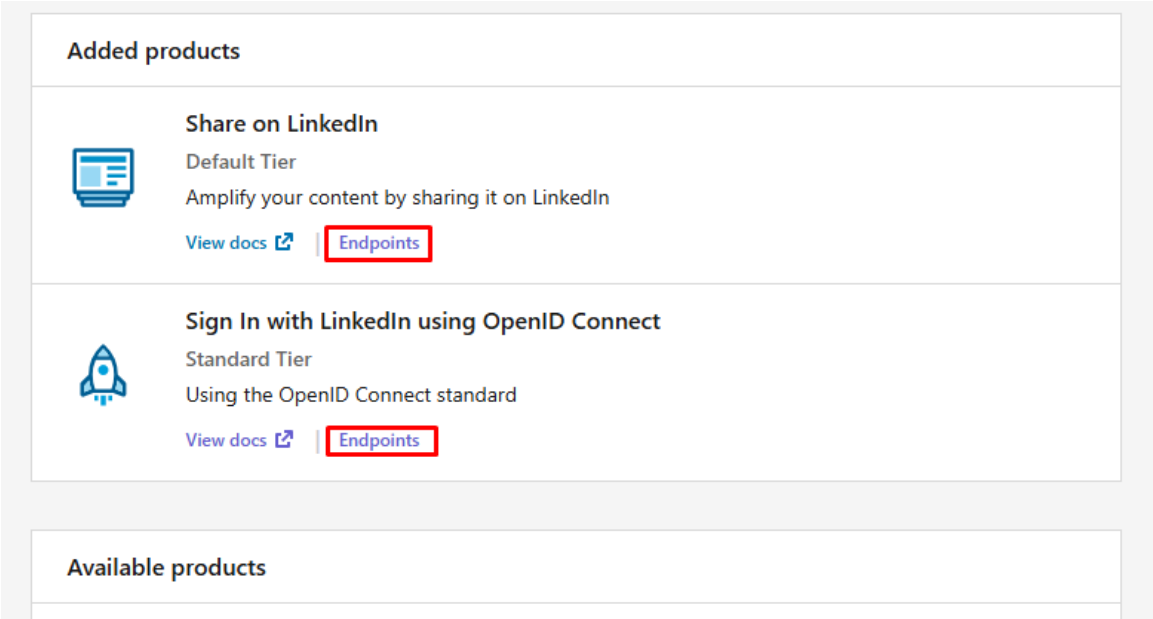
limitiation of Social API

I statrt to scap Linked in Jobs “ title , desc , location and so on)



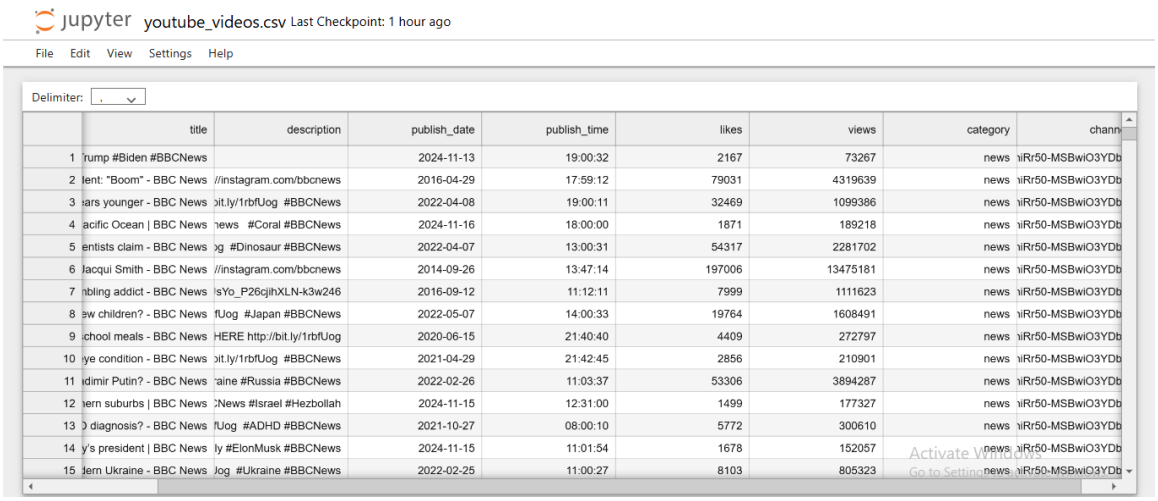


But avalible free end point can’t help in that



I need more products to access or company/busniss formal document to request access to this API’s

And same thing in X “twitter” I can’t read general data just for specific user with unlegal ways becous in 2024 be not free in legal way So finally I choos youtube API to complete assignment



	title	description	publish_date	publish_time	likes	views	category	chann
1	rump #Biden #BBCNews		2024-11-13	19:00:32	2167	73267	news	iRr50-MSBwiO3YDb
2	lent: "Boom" - BBC News	//instagram.com/bbcnews	2016-04-29	17:59:12	79031	4319639	news	iRr50-MSBwiO3YDb
3	ars younger - BBC News	it.ly/1rbfUog #BBCNews	2022-04-08	19:00:11	32469	1099386	news	iRr50-MSBwiO3YDb
4	acific Ocean | BBC News	ews #Coral #BBCNews	2024-11-16	18:00:00	1871	189218	news	iRr50-MSBwiO3YDb
5	entists claim - BBC News	g #Dinosaur #BBCNews	2022-04-07	13:00:31	54317	2281702	news	iRr50-MSBwiO3YDb
6	Jacqui Smith - BBC News	//instagram.com/bbcnews	2014-09-26	13:47:14	197006	13475181	news	iRr50-MSBwiO3YDb
7	nbling addict - BBC News	sYo_P26cjihXLN-k3w246	2016-09-12	11:12:11	7999	1111623	news	iRr50-MSBwiO3YDb
8	ew children? - BBC News	fUog #Japan #BBCNews	2022-05-07	14:00:33	19764	1608491	news	iRr50-MSBwiO3YDb
9	chool meals - BBC News	HERE http://bit.ly/1rbfUog	2020-06-15	21:40:40	4409	272797	news	iRr50-MSBwiO3YDb
10	ye condition - BBC News	it.ly/1rbfUog #BBCNews	2021-04-29	21:42:45	2856	210901	news	iRr50-MSBwiO3YDb
11	idimir Putin? - BBC News	aine #Russia #BBCNews	2022-02-26	11:03:37	53306	3894287	news	iRr50-MSBwiO3YDb
12	ern suburbs | BBC News	News #Israel #Hezbollah	2024-11-15	12:31:00	1499	177327	news	iRr50-MSBwiO3YDb
13	diagnosis? - BBC News	fUog #ADHD #BBCNews	2021-10-27	08:00:10	5772	300610	news	iRr50-MSBwiO3YDb
14	's president | BBC News	ly #ElonMusk #BBCNews	2024-11-15	11:01:54	1678	152057	news	iRr50-MSBwiO3YDb
15	ern Ukraine - BBC News	log #Ukraine #BBCNews	2022-02-25	11:00:27	8103	805323	news	iRr50-MSBwiO3YDb

Data Preparation:

1. Data Exploration

Explore the dataset to understand its structure and content.

1. Import Necessary Libraries

```
[1]: # Import libraries used in data preparation
     import pandas as pd
```

2. Load dataset & Read it

```
[2]: # Load the youtube_videos dataset .. in same folder
     dataset = pd.read_csv("youtube_videos.csv") #Load dataset into DataFrame
     dataset # ensure read it successfully
```

[2]:

	index	title	description	publish_date	publish_time	likes	views	category	channel_id
0	1	Joe Biden congratulates Donald Trump as pair s...	NaN	2024-11-13	19:00:32	2167	73267	news	UC16niRr50-MSBwiO3YDb3RA
1	2	The Queen vs The President: "Boom" - BBC News	The Queen and Prince Harry have responded to B...	2016-04-29	17:59:12	79031	4319639	news	UC16niRr50-MSBwiO3YDb3RA
2	3	Science rejuvenates woman's skin cells to 30 y...	Researchers have rejuvenated a 53-year-old wom...	2022-04-08	19:00:11	32469	1099386	news	UC16niRr50-MSBwiO3YDb3RA
3	4	World's largest coral found in Pacific Ocean |...	The largest coral ever recorded has been found...	2024-11-16	18:00:00	1871	189218	news	UC16niRr50-MSBwiO3YDb3RA
4	5	Dinosaur fossil from asteroid strike that caus...	A dinosaur's leg, stunningly preserved, has be...	2022-04-07	13:00:31	54317	2281702	news	UC16niRr50-MSBwiO3YDb3RA
...	...	...	...	...	...	...	...	...	...
1195	1196	If You're Happy arabic no music (learn arabic)...	If You're Happy arabic no music (learn arabic)...	2018-01-07	08:25:45	1991	1204353	children	UCWIdqSQekeGmUWlSFeCiEnA

3. Dataset Size

```
[3]: # get the dataset Size
     size = dataset.shape
     size
```

```
[3]: (1200, 9)
```

4. Dataset dimensions

```
[4]: # get the dataset dimension
dimension = dataset.ndim
dimension
```

```
[4]: 2
```

5. Get dataset columns

```
[5]: # get columns' titles
titles = dataset.columns
titles
```

```
[5]: Index(['index', 'title', 'description', 'publish_date', 'publish_time',
       'likes', 'views', 'category', 'channel_id'],
      dtype='object')
```

6. Head of dataset “”first rows”

```
[6]: # Step 1 : Data Exploration
# Display the first 5 rows of the dataset using head method
dataset.head()
```

[6]:

	index	title	description	publish_date	publish_time	likes	views	category	channel_id
0	1	Joe Biden congratulates Donald Trump as pair s...	NaN	2024-11-13	19:00:32	2167	73267	news	UC16niRr50-MSBwiO3YDb3RA
1	2	The Queen vs The President: "Boom" - BBC News	The Queen and Prince Harry have responded to B...	2016-04-29	17:59:12	79031	4319639	news	UC16niRr50-MSBwiO3YDb3RA
2	3	Science rejuvenates woman's skin cells to 30 y...	Researchers have rejuvenated a 53-year-old wom...	2022-04-08	19:00:11	32469	1099386	news	UC16niRr50-MSBwiO3YDb3RA
3	4	World's largest coral found in Pacific Ocean |...	The largest coral ever recorded has been found...	2024-11-16	18:00:00	1871	189218	news	UC16niRr50-MSBwiO3YDb3RA
4	5	Dinosaur fossil from asteroid strike that caus...	A dinosaur's leg, stunningly preserved, has be...	2022-04-07	13:00:31	54317	2281702	news	UC16niRr50-MSBwiO3YDb3RA

7. Explore the Structure using info()

```
[7]: # Display information about the dataset
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   index         1200 non-null   int64 
 1   title         1200 non-null   object
 2   description   1167 non-null   object
 3   publish_date  1200 non-null   object
 4   publish_time  1200 non-null   object
 5   likes         1200 non-null   int64 
 6   views         1200 non-null   int64 
 7   category      1200 non-null   object
 8   channel_id    1200 non-null   object
dtypes: int64(3), object(6)
memory usage: 84.5+ KB
```

8. Display descriptive statistic using describe()

```
[8]: # Display descriptive statistics of the dataset "for numeric columns"
dataset.describe()
```

[8]:

	index	likes	views
count	1200.000000	1.200000e+03	1.200000e+03
mean	600.500000	3.221288e+04	7.381537e+06
std	346.554469	1.145511e+05	3.574763e+07
min	1.000000	0.000000e+00	9.000000e+00
25%	300.750000	2.322500e+02	1.650100e+04
50%	600.500000	2.140000e+03	2.709995e+05
75%	900.250000	1.114850e+04	1.226443e+06
max	1200.000000	1.662526e+06	6.675060e+08

9. Data Types

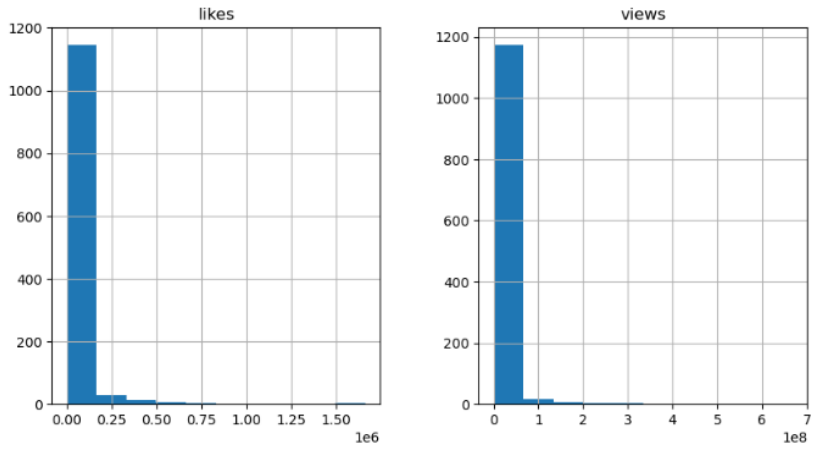
```
[9]: dataset.dtypes
```

```
[9]: index            int64
     title           object
     description     object
     publish_date    object
     publish_time    object
     likes            int64
     views            int64
     category        object
     channel_id      object
     dtype: object
```

Data Exploration befor cleaning & Transformation Steps

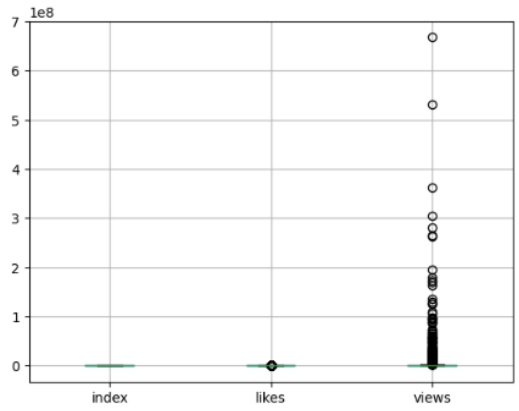
```
[10]: dataset[['likes', 'views']].hist(figsize=(10, 5))
```

```
[10]: array([[<Axes: title={'center': 'likes'}>,
              <Axes: title={'center': 'views'}>]], dtype=object)
```



```
[11]: dataset.boxplot()
```

```
[11]: <Axes: >
```



2. Data Cleaning

1. Missing/Incomplete Values

```
[12]: # Step 2 : Data Cleaning

#count missing values in each attribute
dataset.isnull().sum() #or pd.DataFrame({'missing': dataset.isnull().sum()})
```

```
[12]: index           0
title           0
description    33
publish_date    0
publish_time    0
likes           0
views           0
category        0
channel_id      0
dtype: int64
```

Handel Missing Values

```
[13]: # handel missing values in description
dataset['description'] = dataset['description'].fillna('No description available')

dataset.isnull().sum()
```

```
[13]: index           0
title           0
description     0
publish_date    0
publish_time    0
likes           0
views           0
category        0
channel_id      0
dtype: int64
```

2. Duplicated Values

```
[14]: dataset.duplicated()
```

```
[14]: 0       False
      1       False
      2       False
      3       False
      4       False
              ...
      1195    False
      1196    False
      1197    False
      1198    False
      1199    False
      Length: 1200, dtype: bool
```

```
[15]: # Remove duplicates if exsist but we ensure no dupliaction
      dataset = dataset.drop_duplicates()

      dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   index         1200 non-null   int64
 1   title         1200 non-null   object
 2   description   1200 non-null   object
 3   publish_date  1200 non-null   object
 4   publish_time  1200 non-null   object
 5   likes         1200 non-null   int64
 6   views         1200 non-null   int64
 7   category      1200 non-null   object
 8   channel_id    1200 non-null   object
dtypes: int64(3), object(6)
memory usage: 84.5+ KB
```

No Duplicated Values

3. Noisy Values

In Image below we notice no error, noisy or outlier “can be found later”

```
[16]: # Display summary statistics for all columns
      dataset.describe(include='all')
```

[16]:

	index	title	description	publish_date	publish_time	likes	views	category	channel_id
count	1200.000000	1200	1200	1200	1200	1.200000e+03	1.200000e+03	1200	1200
unique	NaN	1163	966	769	1020	NaN	NaN	5	6
top	NaN	فيلم 3D	To license ISU footage: https://bit.ly/3M7bcaF...	2024-11-22	17:00:02	NaN	NaN	news	UC16niRr50-MSBwiO3YDb3RA
freq	NaN	4	53	28	10	NaN	NaN	400	200
mean	600.500000	NaN	NaN	NaN	NaN	3.221288e+04	7.381537e+06	NaN	NaN
std	346.554469	NaN	NaN	NaN	NaN	1.145511e+05	3.574763e+07	NaN	NaN
min	1.000000	NaN	NaN	NaN	NaN	0.000000e+00	9.000000e+00	NaN	NaN
25%	300.750000	NaN	NaN	NaN	NaN	2.322500e+02	1.650100e+04	NaN	NaN
50%	600.500000	NaN	NaN	NaN	NaN	2.140000e+03	2.709995e+05	NaN	NaN
75%	900.250000	NaN	NaN	NaN	NaN	1.114850e+04	1.226443e+06	NaN	NaN
max	1200.000000	NaN	NaN	NaN	NaN	1.662526e+06	6.675060e+08	NaN	NaN

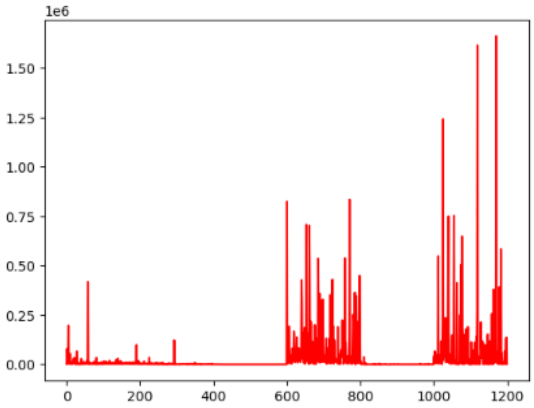
3. Data Transformation

1. Z-score Normalization (StandardScaler)

we need do transformation “normalization” in likes , views column using StandardScaler() : depend on mean and standard deviation of column

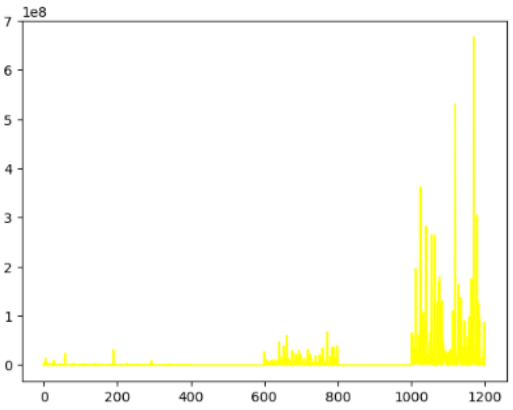
```
[17]: dataset['likes'].plot.line(color = 'red')
```

```
[17]: <Axes: >
```




```
[18]: dataset['views'].plot.line(color = 'yellow')
```

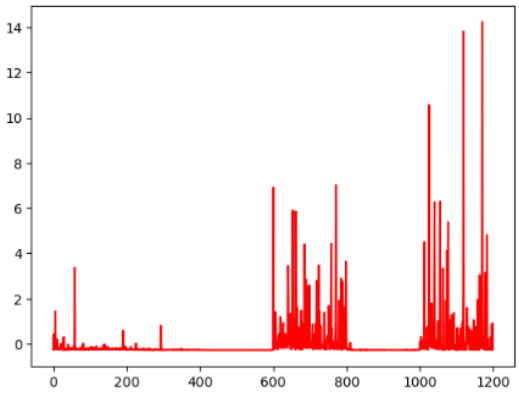
```
[18]: <Axes: >
```




```
[19]: # Step 3 : Data Transformation

from sklearn.preprocessing import StandardScaler

# numeric features
numeric_features = ['likes', 'views']

# Normlize numeric features using StandardScaler
numeric_transformer = StandardScaler()
scaler = StandardScaler()

# Apply the fit_transform method to the numeric columns
dataset[numeric_features] = scaler.fit_transform(dataset[numeric_features])
```

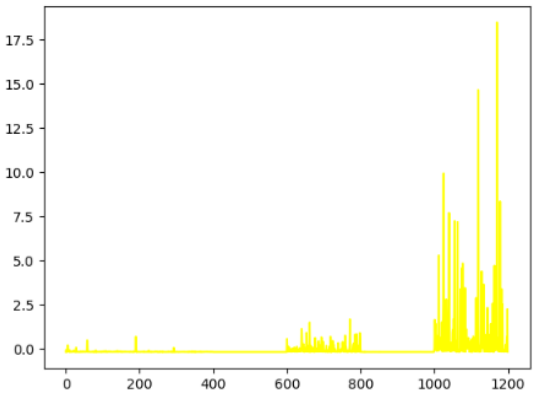
[20]:
```
dataset['likes'].plot.line(color = 'red')
```

[20]: <Axes: >



[21]:
```
dataset['views'].plot.line(color = 'yellow')
```

[21]: <Axes: >



2. Categorical Encoding - Ordinal/Label Encoder

[23]:
```
from sklearn.preprocessing import LabelEncoder


# LabelEncoder
label_encoder = LabelEncoder()
# Fit and transform the category column
dataset['category_encoded'] = label_encoder.fit_transform(dataset['category'])

#  mapping of categories to encoded integers ad dictionary
category_mapping = dict(zip(label_encoder.classes_, range(len(label_encoder.classes_))))
#label_encoder.classes_ :  array of unique categories found in category column
#range(len(label_encoder.classes_)) : range of integers from 0 to the number of unique categories.
#zip(label_encoder.classes_, range(len(label_encoder.classes_))) Pairs each category with its corresponding integer.

category_mapping
```

[23]: {'children': 0, 'cooking': 1, 'education': 2, 'news': 3, 'sport': 4}

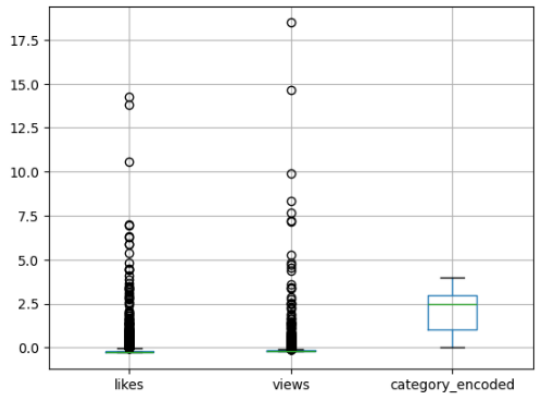
```
[24]: dataset.head()
```

[24]:

	index	title	description	publish_date	publish_time	likes	views	category	channel_id	category_encoded
0	1	Joe Biden congratulates Donald Trump as pair s...	No description available	2024-11-13	19:00:32	-0.262402	-0.204526	news	UC16niRr50-MSBwiO3YDb3RA	3
1	2	The Queen vs The President: "Boom" - BBC News	The Queen and Prince Harry have responded to B...	2016-04-29	17:59:12	0.408880	-0.085689	news	UC16niRr50-MSBwiO3YDb3RA	3
2	3	Science rejuvenates woman's skin cells to 30 y...	Researchers have rejuvenated a 53-year-old wom...	2022-04-08	19:00:11	0.002237	-0.175809	news	UC16niRr50-MSBwiO3YDb3RA	3
3	4	World's largest coral found in Pacific Ocean |...	The largest coral ever recorded has been found...	2024-11-16	18:00:00	-0.264987	-0.201281	news	UC16niRr50-MSBwiO3YDb3RA	3
4	5	Dinosaur fossil from asteroid strike that caus...	A dinosaur's leg, stunningly preserved, has be...	2022-04-07	13:00:31	0.193043	-0.142722	news	UC16niRr50-MSBwiO3YDb3RA	3

```
[25]: dataset.boxplot()
```

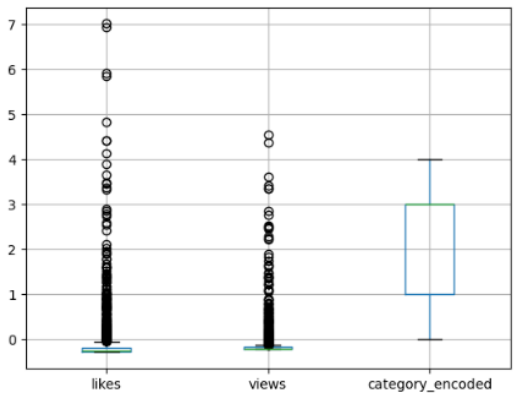
[25]: <Axes: >



We can renove outlier on views and likes:

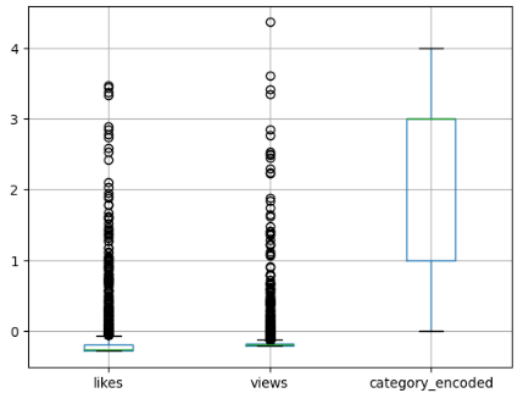
```
[26]: # indices of the top largest views
indices = dataset['views'].nlargest(10).index

# drop this 10 rows from the dataset
dataset = dataset.drop(indices)

dataset.boxplot()
```

[26]: <Axes: >



```
[27]: # indices of the top largest likes
indices = dataset['likes'].nlargest(10).index

# drop this 10 rows from the dataset
dataset = dataset.drop(indices)

dataset.boxplot()
```

```
[27]: <Axes: >
```



4. Feature Engineering

1. ***Create a New Feature: likes_to_views_ratio "ratio of likes to views for each video"***

Indicate that higher ratio suggests that the video is well-received by its viewers and used in Comparative Analysis

```
[29]: # step 4 : Feature Engineering
# new features
dataset['title_length'] = dataset['title'].apply(len)
dataset['description_length'] = dataset['description'].apply(len)

# show new features
dataset[['title', 'title_length', 'description', 'description_length']].head()
```

```
[29]:
```

	title	title_length	description	description_length
0	Joe Biden congratulates Donald Trump as pair s...	80	No description available	24
1	The Queen vs The President: "Boom" - BBC News	45	The Queen and Prince Harry have responded to B...	379
2	Science rejuvenates woman's skin cells to 30 y...	69	Researchers have rejuvenated a 53-year-old wom...	480
3	World's largest coral found in Pacific Ocean |...	55	The largest coral ever recorded has been found...	520
4	Dinosaur fossil from asteroid strike that caus...	94	A dinosaur's leg, stunningly preserved, has be...	680

2. ***Create a New Feature: description_length and title_length***

Useful in SEO Optimization "how long of it" .. insights into how these lengths might impact search engine rankings and viewer engagement"

[30]:
```
# Calculate the likes to views ratio feature
dataset['likes_to_views_ratio'] = dataset['likes'] / dataset['views']

# Display the new feature and some other columns to verify
dataset[['title', 'likes', 'views', 'likes_to_views_ratio']].head()
```

[30]:

	title	likes	views	likes_to_views_ratio
0	Joe Biden congratulates Donald Trump as pair s...	-0.262402	-0.204526	1.282975
1	The Queen vs The President: "Boom" - BBC News	0.408880	-0.085689	-4.771676
2	Science rejuvenates woman's skin cells to 30 y...	0.002237	-0.175809	-0.012723
3	World's largest coral found in Pacific Ocean |...	-0.264987	-0.201281	1.316501
4	Dinosaur fossil from asteroid strike that caus...	0.193043	-0.142722	-1.352586

Data Exploration after all **Data Preparation**

[31]:
```
# Generate histograms for each column
dataset.hist(figsize=(10, 8))
```

[31]:
```
array([[<Axes: title={'center': 'likes'}>,
        <Axes: title={'center': 'views'}>],
       [<Axes: title={'center': 'category_encoded'}>,
        <Axes: title={'center': 'title_length'}>],
       [<Axes: title={'center': 'description_length'}>,
        <Axes: title={'center': 'likes_to_views_ratio'}>]], dtype=object)
```