1. Scrape all video URLs uploaded by the T-Series YouTube channel within the specified date range.

In [18]:

```
import requests

API_KEY = '' # HIDING MY YOU TUBE DATA API
CHANNEL_ID = 'UCq-Fj5jknLsUf-MWSy4_brA'  # T-Series channel ID

start_date = "2023-05-22T00:00:00Z"
end_date = "2023-08-08T23:59:59Z"

URL = f"https://www.googleapis.com/youtube/v3/search?key={API_KEY}&channelId={CHANNEL_ID

videos = []

# Fetch the video details
response = requests.get(URL)
result = response.json()

# Iterate through items and get video links
for item in result['items']:
    if item['id']['kind'] == 'youtube#video':  # To ensure the item is a video
        video_id = item['id']['videoId']
        video_link = f"https://www.youtube.com/watch?v={video_id}"
        videos.append(video_link)

print(videos)
```

['https://www.youtube.com/watch?v=qXEqLWgg9UY', 'https://www.youtube.com/watch?v=nhb4lCCKKjk', 'https://www.youtube.com/watch?v=kfSSwxS9aOs', 'https://www.youtube.com/watch?v=CO0YM4m5-hc', 'https://www.youtube.com/watch?v=NIKwvL4b0uE', 'https://www.youtube.com/watch?v=mZ-9dul2wak', 'https://www.youtube.com/watch?v=b2HqYOYTCgE', 'https://www.youtube.com/watch?v=nuaxkTOSPX8', 'https://www.youtube.com/watch?v=rIKSCTMIoss', 'https://www.youtube.com/watch?v=z0vOhkzefCE', 'https://www.youtube.com/watch?v=OdzSxyowu7s', 'https://www.youtube.com/watch?v=0GS_pfG-yaE', 'https://www.youtube.com/watch?v=5X6CNliwiAM', 'https://www.youtube.com/watch?v=xrsqaB3kwp0', 'https://www.youtube.com/watch?v=0ZfJUjtEeG8', 'https://www.youtube.com/watch?v=3ihp0OpjD04', 'https://www.youtube.com/watch?v=I-_LHvTeVy4', 'https://www.youtube.com/watch?v=xOHejhEwMDY', 'https://www.youtube.com/watch?v=AponoBIY3BU', 'https://www.youtube.com/watch?v=nCMpesZUIWA', 'https://www.youtube.com/watch?v=zNfc-q_BF-8', 'https://www.youtube.com/watch?v=IUffiaUue-U', 'https://www.youtube.com/watch?v=PEI3tn_0Smg', 'https://www.youtube.com/watch?v=2xg_luaPSnQ', 'https://www.youtube.com/watch?v=qs-Kvkds1ks', 'https://www.youtube.com/watch?v=AQEc4BwX6dk', 'https://www.youtube.com/watch?v=5rOqzd3KC5o', 'https://www.youtube.com/watch?v=UiUBPysceNY', 'https://www.youtube.com/watch?v=eG40RRp0Zas', 'https://www.youtube.com/watch?v=fKfqHDdWKKU', 'https://www.youtube.com/watch?v=BwhM9RMDL_4', 'https://www.youtube.com/watch?v=YsCeMprm2O0', 'https://www.youtube.com/watch?v=46QxVICzPB8', 'https://www.youtube.com/watch?v=VT5i6p1w0B0', 'https://www.youtube.com/watch?v=T0d8uvpXq9I', 'https://www.youtube.com/watch?v=EpdsVO9pt9g', 'https://www.youtube.com/watch?v=4-30eu39wRM', 'https://www.youtube.com/watch?v=2SuNLFYD41A', 'https://www.youtube.com/watch?v=Cybo2U_d0Xs', 'https://www.youtube.com/watch?v=Dm0Y_x6bOaQ', 'https://www.youtube.com/watch?v=8gbrGt4hbjc', 'https://www.youtube.com/watch?v=FkgnpnzeKx0', 'https://www.youtube.com/watch?v=IXa3sId1iF4', 'https://www.youtube.com/watch?v=OD1LhUu4bRI', 'https://www.youtube.com/watch?v=KbjwyBaz6Fs', 'https://www.youtube.com/watch?v=rB5QPIP5H6k', 'https://www.youtube.com/watch?v=uSb0M_UQE1o', 'https://www.youtube.com/watch?v=WzuIcK4IxA0', 'https://www.youtube.com/watch?v=VQ4javkA4MQ']

2 For each video URL, extract the exact upload date in the format "Month Day, Year" (e.g., "August 8, 2023").

In [19]:

```
# defining a functiuon to get upload date using You Tube Data Api

import requests

def get_upload_date(api_key, video_id):
    base_url = "https://www.googleapis.com/youtube/v3/videos"
    params = {
        "id": video_id,
        "key": api_key,
        "part": "snippet"
    }

    response = requests.get(base_url, params=params)
    data = response.json()

    # Extract the upload date from the video's snippet
    upload_date = data['items'][0]['snippet']['publishedAt']

    return upload_date
```

In [20]:

```
# Defing a funtion to extract video ID from URL using request

import re

def extract_video_id_from_url_regex(url):
    # Regular expression to match YouTube video IDs
    video_id_match = re.search(r"v=([0-9A-Za-z_-]{10}[048AEIMQUYcgkosw])", url)
    if video_id_match:
        return video_id_match.group(1)
    return None
```

In [21]:

```
# MAking list of video_id and upload date

UPLOAD_DATES=[]
VIDEO_IDS=[]
for i in range(len(videos)):
    UPLOAD_DATES.append(get_upload_date(API_KEY, extract_video_id_from_url_regex(videos[
    VIDEO_IDS.append(extract_video_id_from_url_regex(videos[i]))
```

In [22]:

```
UPLOAD_DATES
```

Out[22]:

```
['2023-08-08T10:30:16Z',
 '2023-08-08T09:30:13Z',
 '2023-08-08T03:29:07Z',
 '2023-08-07T15:00:07Z',
 '2023-08-07T14:30:08Z',
 '2023-08-07T12:00:39Z',
 '2023-08-07T06:49:14Z',
 '2023-08-06T12:30:25Z',
 '2023-08-06T09:30:13Z',
 '2023-08-06T05:30:18Z',
 '2023-08-05T15:00:30Z',
 '2023-08-05T13:33:34Z',
 '2023-08-05T13:22:20Z',
 '2023-08-05T11:31:56Z',
 '2023-08-05T07:30:00Z',
 '2023-08-04T11:00:03Z',
 '2023-08-04T05:29:12Z',
 '2023-08-03T13:00:51Z',
 '2023-08-03T12:40:18Z',
 '2023-08-02T08:32:38Z',
 '2023-08-02T05:30:17Z',
 '2023-08-01T14:00:11Z',
 '2023-08-01T13:06:41Z',
 '2023-08-01T11:30:02Z',
 '2023-08-01T08:30:25Z',
 '2023-07-31T07:19:08Z',
 '2023-07-31T06:25:42Z',
 '2023-07-31T03:29:09Z',
 '2023-07-29T15:30:08Z',
 '2023-07-29T12:30:31Z',
 '2023-07-29T11:20:01Z',
 '2023-07-29T09:30:08Z',
 '2023-07-29T07:30:00Z',
 '2023-07-29T06:07:31Z',
 '2023-07-29T05:30:12Z',
 '2023-07-28T14:30:04Z',
 '2023-07-28T05:29:08Z',
 '2023-07-26T14:45:00Z',
 '2023-07-26T14:15:02Z',
 '2023-07-26T13:50:00Z',
 '2023-07-26T13:45:01Z',
 '2023-07-26T13:40:00Z',
 '2023-07-25T09:30:09Z',
 '2023-07-25T05:30:08Z',
 '2023-07-24T11:31:55Z',
 '2023-07-24T06:30:09Z',
 '2023-07-24T03:30:12Z',
 '2023-07-23T13:30:17Z',
 '2023-07-23T08:30:13Z']
```

In [23]:

```
# converting upload date in required format


from datetime import datetime

def convert_rfc3339_to_custom_format(rfc_date):
    # Parse the RFC 3339 formatted date
    date_obj = datetime.fromisoformat(rfc_date.replace("Z", "+00:00"))

    # Convert to the desired format
    return date_obj.strftime('%B %d, %Y')

# Convert the list
converted_dates = [convert_rfc3339_to_custom_format(date) for date in UPLOAD_DATES]

print(converted_dates)  # Outputs: ['May 22, 2023', 'August 08, 2023']
```

```
['August 08, 2023', 'August 08, 2023', 'August 08, 2023', 'August 07, 202
3', 'August 07, 2023', 'August 07, 2023', 'August 07, 2023', 'August 06,
2023', 'August 06, 2023', 'August 06, 2023', 'August 05, 2023', 'August 0
5, 2023', 'August 05, 2023', 'August 05, 2023', 'August 05, 2023', 'Augus
t 04, 2023', 'August 04, 2023', 'August 03, 2023', 'August 03, 2023', 'Au
gust 02, 2023', 'August 02, 2023', 'August 01, 2023', 'August 01, 2023',
'August 01, 2023', 'August 01, 2023', 'July 31, 2023', 'July 31, 2023',
'July 31, 2023', 'July 29, 2023', 'July 29, 2023', 'July 29, 2023', 'July
29, 2023', 'July 29, 2023', 'July 29, 2023', 'July 29, 2023', 'July 28, 2
023', 'July 28, 2023', 'July 26, 2023', 'July 26, 2023', 'July 26, 2023',
'July 26, 2023', 'July 26, 2023', 'July 25, 2023', 'July 25, 2023', 'July
24, 2023', 'July 24, 2023', 'July 24, 2023', 'July 23, 2023', 'July 23, 2
023']
```

3 Collect all unique upload dates and the corresponding video URLs.

In [24]:

```
# zipping the urls with their upload date and creating key value pairs

result = dict(zip(videos, converted_dates))
```

In [25]:

```
result
```

Out[25]:


```
{'https://www.youtube.com/watch?v=qXEqLWgg9UY': 'August 08, 2023',
 'https://www.youtube.com/watch?v=nhb4lCCKKjk': 'August 08, 2023',
 'https://www.youtube.com/watch?v=kfSSwxS9aOs': 'August 08, 2023',
 'https://www.youtube.com/watch?v=CO0YM4m5-hc': 'August 07, 2023',
 'https://www.youtube.com/watch?v=NIKwvL4b0uE': 'August 07, 2023',
 'https://www.youtube.com/watch?v=mZ-9dul2wak': 'August 07, 2023',
 'https://www.youtube.com/watch?v=b2HqYOYTCgE': 'August 07, 2023',
 'https://www.youtube.com/watch?v=nuaxkTOSPX8': 'August 06, 2023',
 'https://www.youtube.com/watch?v=rIKSCTMIoss': 'August 06, 2023',
 'https://www.youtube.com/watch?v=z0vOhkzefCE': 'August 06, 2023',
 'https://www.youtube.com/watch?v=OdzSxyowu7s': 'August 05, 2023',
 'https://www.youtube.com/watch?v=0GS_pfG-yaE': 'August 05, 2023',
 'https://www.youtube.com/watch?v=5X6CNliwiAM': 'August 05, 2023',
 'https://www.youtube.com/watch?v=xrsqaB3kwp0': 'August 05, 2023',
 'https://www.youtube.com/watch?v=0ZfJUjtEeG8': 'August 05, 2023',
 'https://www.youtube.com/watch?v=3ihp0OpjD04': 'August 04, 2023',
 'https://www.youtube.com/watch?v=I-_LHvTeVy4': 'August 04, 2023',
 'https://www.youtube.com/watch?v=xOHejhEwMDY': 'August 03, 2023',
 'https://www.youtube.com/watch?v=AponoBIY3BU': 'August 03, 2023',
 'https://www.youtube.com/watch?v=nCMpesZUIWA': 'August 02, 2023',
 'https://www.youtube.com/watch?v=zNfc-q_BF-8': 'August 02, 2023',
 'https://www.youtube.com/watch?v=IUffiaUue-U': 'August 01, 2023',
 'https://www.youtube.com/watch?v=PEI3tn_0Smg': 'August 01, 2023',
 'https://www.youtube.com/watch?v=2xg_luaPSnQ': 'August 01, 2023',
 'https://www.youtube.com/watch?v=qs-Kvkds1ks': 'August 01, 2023',
 'https://www.youtube.com/watch?v=AQEc4BwX6dk': 'July 31, 2023',
 'https://www.youtube.com/watch?v=5rOqzd3KC5o': 'July 31, 2023',
 'https://www.youtube.com/watch?v=UiUBPysceNY': 'July 31, 2023',
 'https://www.youtube.com/watch?v=eG40RRp0Zas': 'July 29, 2023',
 'https://www.youtube.com/watch?v=fKfqHDdWKKU': 'July 29, 2023',
 'https://www.youtube.com/watch?v=BwhM9RMDL_4': 'July 29, 2023',
 'https://www.youtube.com/watch?v=YsCeMprm2O0': 'July 29, 2023',
 'https://www.youtube.com/watch?v=46QxVICzPB8': 'July 29, 2023',
 'https://www.youtube.com/watch?v=VT5i6p1w0B0': 'July 29, 2023',
 'https://www.youtube.com/watch?v=T0d8uvpXq9I': 'July 29, 2023',
 'https://www.youtube.com/watch?v=EpdsVO9pt9g': 'July 28, 2023',
 'https://www.youtube.com/watch?v=4-30eu39wRM': 'July 28, 2023',
 'https://www.youtube.com/watch?v=2SuNLFYD41A': 'July 26, 2023',
 'https://www.youtube.com/watch?v=Cybo2U_d0Xs': 'July 26, 2023',
 'https://www.youtube.com/watch?v=Dm0Y_x6bOaQ': 'July 26, 2023',
 'https://www.youtube.com/watch?v=8gbrGt4hbjc': 'July 26, 2023',
 'https://www.youtube.com/watch?v=FkgnpnzeKx0': 'July 26, 2023',
 'https://www.youtube.com/watch?v=IXa3sId1iF4': 'July 25, 2023',
 'https://www.youtube.com/watch?v=OD1LhUu4bRI': 'July 25, 2023',
 'https://www.youtube.com/watch?v=KbjwyBaz6Fs': 'July 24, 2023',
 'https://www.youtube.com/watch?v=rB5QPIP5H6k': 'July 24, 2023',
 'https://www.youtube.com/watch?v=uSb0M_UQE1o': 'July 24, 2023',
 'https://www.youtube.com/watch?v=WzuIcK4IxA0': 'July 23, 2023',
 'https://www.youtube.com/watch?v=VQ4javkA4MQ': 'July 23, 2023'}
```


4 Perform a case-insensitive search to find and count the most frequently repeated character in the video

URLs Video ID.

In [36]:

```
def most_repeated_character(strings):
    # Combine all strings into a single string
    combined_string = ''.join(strings)

    # Count characters
    char_count = {}
    for char in combined_string:
        char_count[char] = char_count.get(char, 0) + 1

    # Find the character(s) with the maximum count
    max_count = max(char_count.values())
    most_repeated_chars = [char for char, count in char_count.items() if count == max_co

    return most_repeated_chars, max_count


chars, count = most_repeated_character(VIDEO_IDS)

print(f"Most repeated character(s): {', '.join(chars)}")
print(f"Count: {count}")
```

```
Most repeated character(s): 0
Count: 21
```

5 Print the most repeated character and its count.

In [38]:

```
def most_repeated_character(strings):
    # Combine all strings into a single string
    combined_string = ''.join(strings)

    # Count characters
    char_count = {}
    for char in combined_string:
        char_count[char] = char_count.get(char, 0) + 1
    sorted_data = {k: v for k, v in sorted(char_count.items(), key=lambda item: item[1],

    return sorted_data
```

In [39]:

```
D=most_repeated_character(VIDEO_IDS)
```

In [41]:

```
for key, value in D.items():
    print(f"key: {key} Value: {value}")
```

key: 0 Value: 21
key: 4 Value: 17
key: s Value: 16
key: I Value: 16
key: U Value: 13
key: O Value: 13
key: p Value: 13
key: C Value: 12
key: K Value: 12
key: k Value: 12
key: w Value: 12
key: a Value: 12
key: u Value: 12
key: E Value: 11
key: S Value: 11
key: M Value: 11
key: e Value: 11
key: B Value: 11
key: Y Value: 10
key: b Value: 10
key: x Value: 10
key: d Value: 10
key: q Value: 9
key: f Value: 9
key: - Value: 9
key: z Value: 9
key: _ Value: 9
key: g Value: 8
key: 9 Value: 8
key: n Value: 8
key: h Value: 8
key: 3 Value: 8
key: Q Value: 8
key: X Value: 7
key: j Value: 7
key: 5 Value: 7
key: P Value: 7
key: o Value: 7
key: 6 Value: 7
key: i Value: 7
key: A Value: 7
key: D Value: 7
key: L Value: 6
key: c Value: 6
key: v Value: 6
key: 2 Value: 6
key: T Value: 6
key: 8 Value: 6
key: r Value: 6
key: y Value: 6
key: 1 Value: 6
key: m Value: 5
key: N Value: 5
key: H Value: 5
key: G Value: 5
key: V Value: 5
key: F Value: 5
key: R Value: 5
key: W Value: 4
key: l Value: 4
key: Z Value: 4

```
key: t Value: 4
key: 7 Value: 1
key: J Value: 1
```