```
In [1]: import requests
        from bs4 import BeautifulSoup
        from pandas import DataFrame
        from json import loads
```

```
In [2]: url = 'https://www.youtube.com/@PW-Foundation/videos'
```

```
In [3]: url
```

```
Out[3]: 'https://www.youtube.com/@PW-Foundation/videos'
```

```
In [4]: # Get the html by get method
        r = requests.get(url)
        print(r)

        <Response [200]>
```

```
In [5]: #Create BeautifulSoup object
        youtube_html = BeautifulSoup(r.text , 'html.parser')
```

```
In [6]: big_box = youtube_html.find_all('script')
```

```
In [7]: len(big_box)
```

```
Out[7]: 42
```

So far, this code imports necessary modules for web scraping, sends an HTTP GET request to a YouTube channel URL, creates a BeautifulSoup object from the HTML code of the web page, and finds all the script tags in the web page.

```
In [8]: def script_to_json(tags: list) -> dict:
            for tag in reversed(tags):
                text: str = tag.text
                if 'ytInitialData = {"responseContext"' in text:
                    return loads(text[20:-1])

            raise ValueError('Required script tag is not found in the given tags ')
```

This code defines a function script_to_json() that converts a script tag to a JSON object.

The function takes a list of script tags as input and searches for the tag that contains the JSON data that we are interested in. It does this by iterating over the list of tags in reverse order (starting from the end of the list), and searching for the ytInitialData JSON object in the text attribute of each tag.

If the required ytInitialData JSON object is found, the function extracts the JSON string from the text attribute of the tag (excluding the first 20 and last 1 characters, which contain some unwanted characters), and uses the loads() method from the json module to convert the string to a Python dictionary.

If the required ytInitialData JSON object is not found in any of the tags, the function raises a ValueError.

The resulting dictionary contains the data that we are interested in, which can be used for further analysis or manipulation.

In [10]:
```
youtube_data = script_to_json(big_box)

#Return data from videos
def get_contents_dict(data):
    return data['contents']['twoColumnBrowseResultsRenderer']['tabs'][1]['tabRen
```

This code defines a function get_contents_dict() that extracts the contents dictionary from the ytInitialData JSON object.

The function takes the ytInitialData dictionary as input and returns the contents dictionary that contains the data for the videos on the channel.

The contents dictionary can be found by navigating through the various keys in the ytInitialData dictionary. In this case, we can find it by following this path:

Q1. Write a python program to extract the video URL of the first five videos.

In [11]:
```
def get_videoUrl(data:dict, n: int = 5):
    contents = get_contents_dict(youtube_data)

    if n > 30:
        raise ValueError('Max Limit is 30.')

    result = []
    for i in range(n):
        result.append('https://www.youtube.com/watch?v=' +
                      contents[i]['richItemRenderer']['content']['videoRendere
    return result

get_videoUrl(youtube_data)
```

Out[11]:
```
['https://www.youtube.com/watch?v=FjqfizRZths',
 'https://www.youtube.com/watch?v=mdTrjYtwCXg',
 'https://www.youtube.com/watch?v=Acv0Nyb5CTc',
 'https://www.youtube.com/watch?v=3cJ3TnHl22U',
 'https://www.youtube.com/watch?v=1CzStq6B52o']
```

This code defines a function get_videoUrl() that extracts the URLs of the first n videos on the channel.

The function takes the ytInitialData dictionary as input and an optional parameter n that specifies the number of videos to retrieve (default value is 5). The function first extracts the contents dictionary using the get_contents_dict() function, and then iterates over the first n items in the contents list.

For each item, the function extracts the videoId from the videoRenderer dictionary and concatenates it with the YouTube video URL to create the full URL for the video. The resulting list result contains the URLs of the first n videos on the channel.

Note that the function raises a ValueError if the value of n is greater than 30, which is the maximum number of videos that can be retrieved from the YouTube API in a single request. This limit is enforced to prevent excessive usage of the API, which could result in rate limiting or other errors.

Q2. Write a python program to extract the URL of the video thumbnails of the first five videos.

In [12]:
```
def get_thumbnails(data:dict , n:int = 5):
    contents =get_contents_dict(youtube_data)

    if n>30:
        raise ValueError('Max Limit is 30')

    result = []
    for i in range(n):
        result.append(contents[i]['richItemRenderer']['content']['videoRenderer'

    return result
get_thumbnails(youtube_data)
```

Out[12]:
```
['https://i.ytimg.com/vi/FjqfizRZths/hqdefault.jpg?sqp=-oaymwEjCNACELwBSFryq4qp
AxUIARUAAAAAGAElAADIQj0AgKJDeAE=&rs=AOn4CLA8PSrOzXF_tfXY9VjcO9IqvXYU_w',
 'https://i.ytimg.com/vi/mdTrjYtwCXg/hqdefault.jpg?sqp=-oaymwEjCNACELwBSFryq4qp
AxUIARUAAAAAGAElAADIQj0AgKJDeAE=&rs=AOn4CLB0D-4cniwCdY8mEvzglKxLK0KESg',
 'https://i.ytimg.com/vi/Acv0Nyb5CTc/hqdefault.jpg?sqp=-oaymwEjCNACELwBSFryq4qp
AxUIARUAAAAAGAElAADIQj0AgKJDeAE=&rs=AOn4CLCJbyAkE3Nv1UjjWGzy7FdqSZk9AQ',
 'https://i.ytimg.com/vi/3cJ3TnHl22U/hqdefault.jpg?sqp=-oaymwEjCNACELwBSFryq4qp
AxUIARUAAAAAGAElAADIQj0AgKJDeAE=&rs=AOn4CLDxiH7vjvzmzxI7diHF7urc7W88gQ',
 'https://i.ytimg.com/vi/lCzStq6B52o/hqdefault.jpg?sqp=-oaymwEjCNACELwBSFryq4qp
AxUIARUAAAAAGAElAADIQj0AgKJDeAE=&rs=AOn4CLATlL1t1b7YoHF_9jOCzDyVIo3rXw']
```

This code defines a function get_thumbnails() that extracts the URLs of the thumbnail images for the first n videos on the channel.

The function takes the ytInitialData dictionary as input and an optional parameter n that specifies the number of thumbnails to retrieve (default value is 5). The function first extracts the contents dictionary using the get_contents_dict() function, and then iterates over the first n items in the contents list.

For each item, the function extracts the URL of the highest resolution thumbnail image by selecting the last item in the thumbnails list of the thumbnail dictionary.

The resulting list result contains the URLs of the thumbnail images for the first n videos on the channel. These URLs can be used to download and display the thumbnail images for each video.

Q3. Write a python program to extract the title of the first five videos.

In [13]:
```
def get_title(data:dict , n:int=5):
    contents=get_contents_dict(youtube_data)

    if n>30:
        raise ValueError('Max Limit is 30')

    result=[]
    for i in range(n):
        result.append(contents[i]['richItemRenderer']['content']['videoRenderer'

    return result
get_title(youtube_data)
```

Out[13]:

```
["Samriddhi Ma'am & Rakshak Sir क्या बाते कर रहे है?? 😱😍",
 "Launching India's BEST Batch for Class 10th !! UDAAN - Guaranteed 95% in Boar
ds 2025 🔥",
 'Last 4 Days Strategy for SST Board Exam | Presentation Tips | Score 95%+ in C
lass 10th 🤩',
 '4 Days Hard CHALLENGE !! Class 10th SST Marathon Score 100/100 || CBSE Board
🔥',
 'Class 10th BIOLOGY 50+ Most Important Question | Complete Science | Paper यही
से आएगा !! 🔥']
```

This code defines a function get_title() that extracts the titles of the first n videos on the channel.

The function takes the ytInitialData dictionary as input and an optional parameter n that specifies the number of titles to retrieve (default value is 5). The function first extracts the contents dictionary using the get_contents_dict() function, and then iterates over the first n items in the contents list.

For each item, the function extracts the title of the video by selecting the last item in the runs list of the title dictionary.

The resulting list result contains the titles of the first n videos on the channel. These titles can be used to identify each video and to display the titles in a list or table.

Q4. Write a python program to extract the number of views of the first five videos.

In [14]:

```
def get_views(data: dict , n :int = 5):
    contents = get_contents_dict(youtube_data)

    if n > 30:
        raise ValueError('MAx Limit is 30')

    result = []
    for i in range(n):
        result.append(int(contents[i]['richItemRenderer']['content']['videoRende
                         [:-6].replace(',' , '')))

    return result
get_views(youtube_data)
```

Out[14]:

```
[52596, 61310, 63564, 64507, 42708]
```

This code defines a function get_views() that extracts the view counts of the first n videos on the channel.

The function takes the ytInitialData dictionary as input and an optional parameter n that specifies the number of view counts to retrieve (default value is 5). The function first extracts the contents dictionary using the get_contents_dict() function, and then iterates over the first n items in the contents list.

For each item, the function extracts the view count of the video by selecting the simpleText value of the viewCountText dictionary. The view count is then converted to an integer by removing the last six characters (which represent " views") and any commas.

The resulting list result contains the view counts of the first n videos on the channel. These view counts can be used to measure the popularity of each video and to display the view counts in a list or table.

Q5. Write a python program to extract the time of posting of video for the first five videos.

```
In [15]: def get_time_of_post(data: dict , n : int=5):
             contents = get_contents_dict(youtube_data)

             if n > 30:
                 raise ValueError('Max Limit is 30')

             result = []
             for i in range(n):
                 result.append(contents[i]['richItemRenderer']['content']['videoRenderer'

             return result
         get_time_of_post(youtube_data)
```

```
Out[15]: ['22 hours ago', '2 days ago', '2 days ago', '3 days ago', '3 days ago']
```

This code defines a function get_time_of_posting() that extracts the publication dates of the first n videos on the channel.

The function takes the ytInitialData dictionary as input and an optional parameter n that specifies the number of publication dates to retrieve (default value is 5). The function first extracts the contents dictionary using the get_contents_dict() function, and then iterates over the first n items in the contents list.

For each item, the function extracts the publication date of the video by selecting the simpleText value of the publishedTimeText dictionary.

The resulting list result contains the publication dates of the first n videos on the channel. These dates can be used to determine when each video was published and to display the publication dates in a list or table.

Note: Save all the data scraped in the above questions in a CSV file. Save data in CSV format.

```
In [16]: def get_channel_video_details(data: dict, n: int):
             thumbnails = get_thumbnails(data, n)
             time_of_posting = get_time_of_post(data, n)
             titles = get_title(data, n)
             video_urls = get_videoUrl(data, n)
             views = get_views(data , n)

             main_data = list(zip(video_urls, titles, thumbnails, time_of_posting , views
             df = DataFrame.from_dict(main_data)
             df.rename(
                 columns={
                     0: 'video_urls',
                     1: 'title',
                     2: 'thumbnail_url',
                     3: 'time_of_posting',
                     4: 'views'
                 }, inplace=True)

             return df

         channel_data = get_channel_video_details(youtube_data, 10)
         channel_data.to_csv('bhi-PW-Foundation.csv', index=False)
```

This code defines a function get_channel_video_details() that extracts the video details of the first n videos on the channel.

The function takes the ytInitialData dictionary as input and an integer n that specifies the number of videos to retrieve. The function calls the get_thumbnails(), get_time_of_posting(), get_title(), and get_videoUrl() functions to extract the thumbnail URLs, publication dates, titles, and video URLs of the first n videos on the channel. It then zips these lists together into a single list of tuples called main_data.

The function then creates a Pandas DataFrame from main_data and renames the columns to video_urls, title, thumbnail_url, and time_of_posting. Finally, the function returns the DataFrame.

The resulting DataFrame channel_data contains the video details of the first n videos on the channel, including the video URLs, titles, thumbnail URLs, and publication dates. The DataFrame is saved to a CSV file called PW-Foundation.csv with the to_csv() method.

In [ ]:

```

```