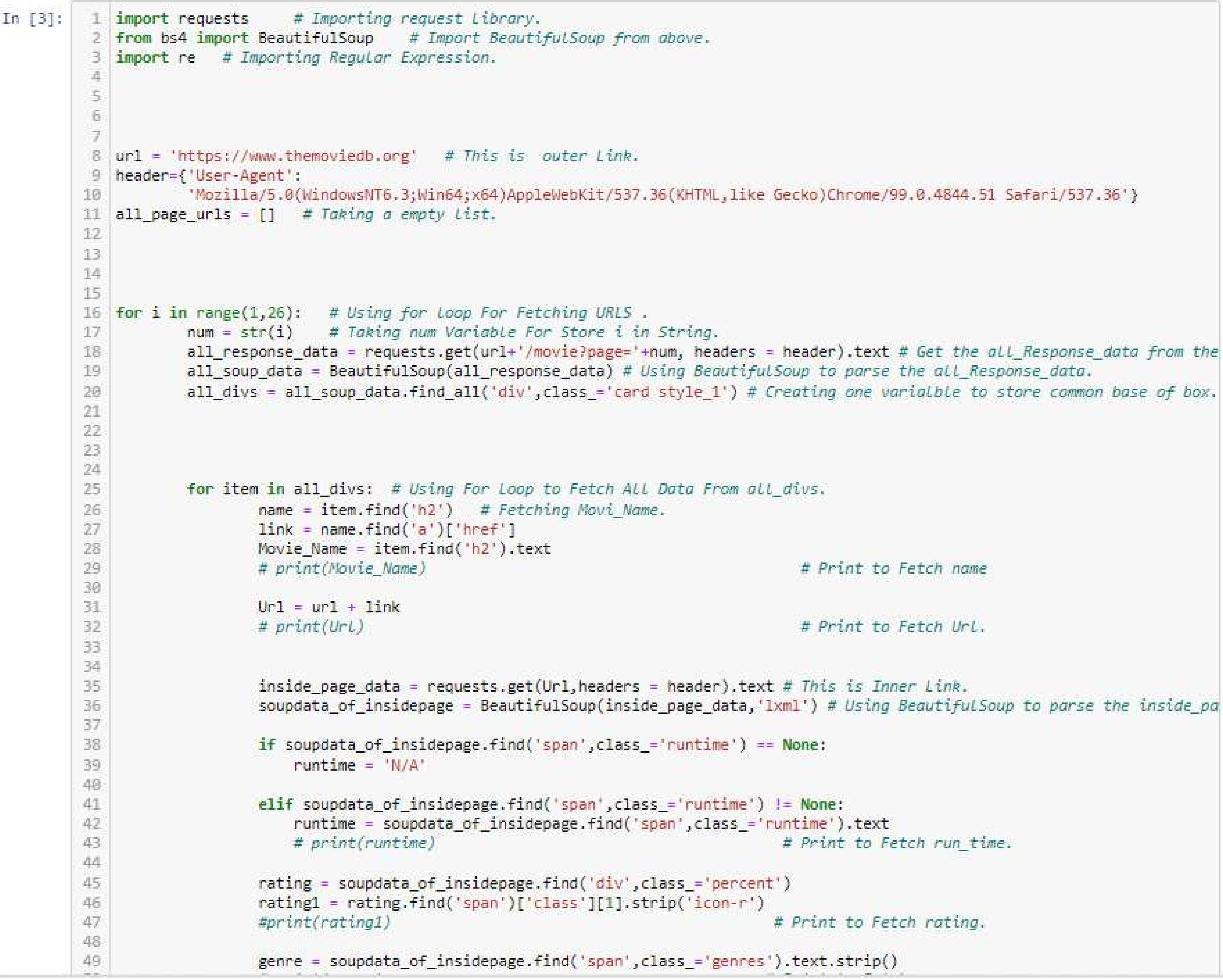
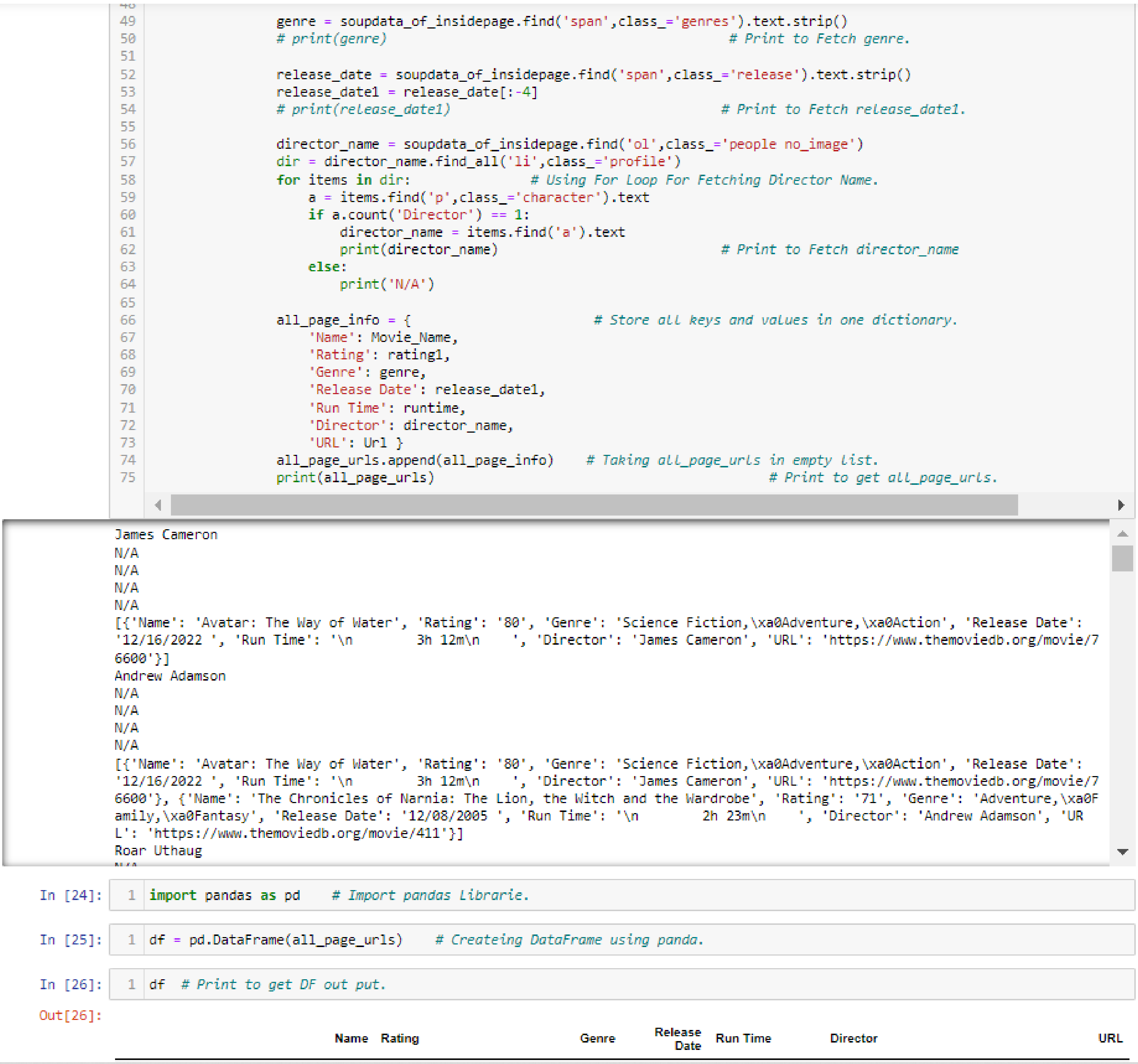
Project On TMDB



In [3]:

```
import requests     # Importing request Library.
from bs4 import BeautifulSoup    # Import BeautifulSoup from above.
import re   # Importing Regular Expression.




url = 'https://www.themoviedb.org'   # This is  outer Link.
header={'User-Agent':
        'Mozilla/5.0(WindowsNT6.3;Win64;x64)AppleWebKit/537.36(KHTML,like Gecko)Chrome/99.0.4844.51 Safari/537.36'}
all_page_urls = []   # Taking a empty List.




for i in range(1,26):   # Using for Loop For Fetching URLS .
        num = str(i)    # Taking num Variable For Store i in String.
        all_response_data = requests.get(url+'/movie?page='+num, headers = header).text # Get the all_Response_data from the
        all_soup_data = BeautifulSoup(all_response_data) # Using BeautifulSoup to parse the all_Response_data.
        all_divs = all_soup_data.find_all('div',class_='card style_1') # Creating one varialble to store common base of box.




        for item in all_divs:  # Using For Loop to Fetch All Data From all_divs.
                name = item.find('h2')   # Fetching Movi_Name.
                link = name.find('a')['href']
                Movie_Name = item.find('h2').text
                # print(Movie_Name)                                     # Print to Fetch name

                Url = url + link
                # print(Url)                                            # Print to Fetch Url.


                inside_page_data = requests.get(Url,headers = header).text # This is Inner Link.
                soupdata_of_insidepage = BeautifulSoup(inside_page_data,'lxml') # Using BeautifulSoup to parse the inside_pa

                if soupdata_of_insidepage.find('span',class_='runtime') == None:
                    runtime = 'N/A'

                elif soupdata_of_insidepage.find('span',class_='runtime') != None:
                    runtime = soupdata_of_insidepage.find('span',class_='runtime').text
                    # print(runtime)                                    # Print to Fetch run_time.

                rating = soupdata_of_insidepage.find('div',class_='percent')
                rating1 = rating.find('span')['class'][1].strip('icon-r')
                #print(rating1)                                         # Print to Fetch rating.

                genre = soupdata_of_insidepage.find('span',class_='genres').text.strip()
```



```
                genre = soupdata_of_insidepage.find('span',class_='genres').text.strip()
                # print(genre)                                          # Print to Fetch genre.

                release_date = soupdata_of_insidepage.find('span',class_='release').text.strip()
                release_date1 = release_date[:-4]
                # print(release_date1)                                  # Print to Fetch release_date1.

                director_name = soupdata_of_insidepage.find('ol',class_='people no_image')
                dir = director_name.find_all('li',class_='profile')
                for items in dir:                # Using For Loop For Fetching Director Name.
                    a = items.find('p',class_='character').text
                    if a.count('Director') == 1:
                        director_name = items.find('a').text
                        print(director_name)                            # Print to Fetch director_name
                    else:
                        print('N/A')

                all_page_info = {                                       # Store all keys and values in one dictionary.
                    'Name': Movie_Name,
                    'Rating': rating1,
                    'Genre': genre,
                    'Release Date': release_date1,
                    'Run Time': runtime,
                    'Director': director_name,
                    'URL': Url }
                all_page_urls.append(all_page_info)    # Taking all_page_urls in empty list.
                print(all_page_urls)                                          # Print to get all_page_urls.
```

```
James Cameron
N/A
N/A
N/A
N/A
[{'Name': 'Avatar: The Way of Water', 'Rating': '80', 'Genre': 'Science Fiction,\xa0Adventure,\xa0Action', 'Release Date':
'12/16/2022 ', 'Run Time': '\n        3h 12m\n    ', 'Director': 'James Cameron', 'URL': 'https://www.themoviedb.org/movie/7
6600'}]
Andrew Adamson
N/A
N/A
N/A
N/A
[{'Name': 'Avatar: The Way of Water', 'Rating': '80', 'Genre': 'Science Fiction,\xa0Adventure,\xa0Action', 'Release Date':
'12/16/2022 ', 'Run Time': '\n        3h 12m\n    ', 'Director': 'James Cameron', 'URL': 'https://www.themoviedb.org/movie/7
6600'}, {'Name': 'The Chronicles of Narnia: The Lion, the Witch and the Wardrobe', 'Rating': '71', 'Genre': 'Adventure,\xa0F
amily,\xa0Fantasy', 'Release Date': '12/08/2005 ', 'Run Time': '\n        2h 23m\n    ', 'Director': 'Andrew Adamson', 'UR
L': 'https://www.themoviedb.org/movie/411'}]
Roar Uthaug
```

In [24]:

```
import pandas as pd    # Import pandas Librarie.
```

In [25]:

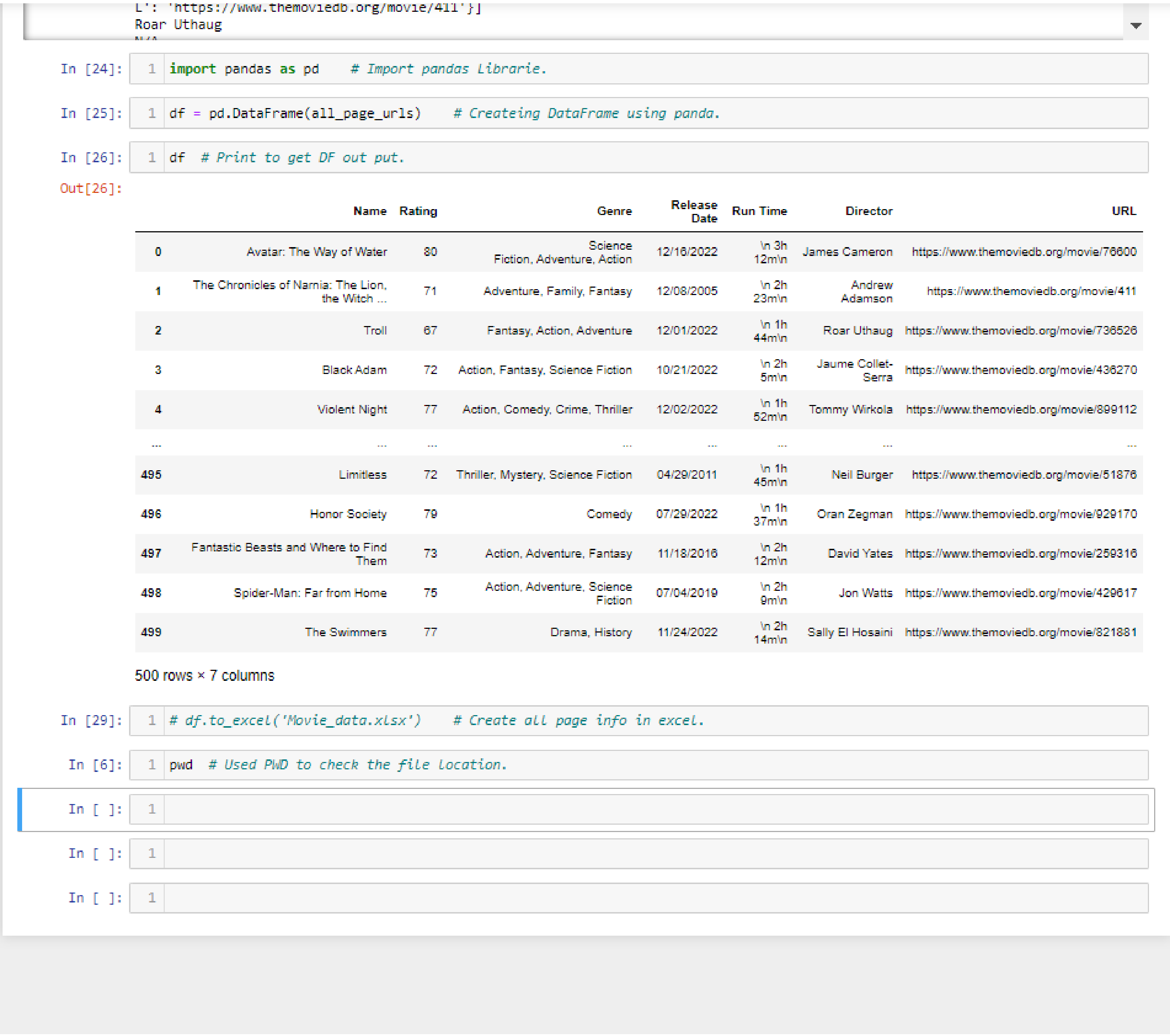
```
df = pd.DataFrame(all_page_urls)    # Createing DataFrame using panda.
```

In [26]:

```
df  # Print to get DF out put.
```

Out[26]:

	Name	Rating	Genre	Release Date	Run Time	Director	URL



```
L': 'https://www.themoviedb.org/movie/411'}]
Roar Uthaug
N/A
```

```
In [24]: import pandas as pd     # Import pandas Librarie.
```

```
In [25]: df = pd.DataFrame(all_page_urls)     # Createing DataFrame using panda.
```

```
In [26]: df  # Print to get DF out put.
```

Out[26]:

	Name	Rating	Genre	Release Date	Run Time	Director	URL
0	Avatar: The Way of Water	80	Science Fiction, Adventure, Action	12/16/2022	\n 3h 12m\n	James Cameron	https://www.themoviedb.org/movie/76600
1	The Chronicles of Narnia: The Lion, the Witch ...	71	Adventure, Family, Fantasy	12/08/2005	\n 2h 23m\n	Andrew Adamson	https://www.themoviedb.org/movie/411
2	Troll	67	Fantasy, Action, Adventure	12/01/2022	\n 1h 44m\n	Roar Uthaug	https://www.themoviedb.org/movie/736526
3	Black Adam	72	Action, Fantasy, Science Fiction	10/21/2022	\n 2h 5m\n	Jaume Collet-Serra	https://www.themoviedb.org/movie/436270
4	Violent Night	77	Action, Comedy, Crime, Thriller	12/02/2022	\n 1h 52m\n	Tommy Wirkola	https://www.themoviedb.org/movie/899112
...	...	...	...	...	...	...	...
495	Limitless	72	Thriller, Mystery, Science Fiction	04/29/2011	\n 1h 45m\n	Neil Burger	https://www.themoviedb.org/movie/51876
496	Honor Society	79	Comedy	07/29/2022	\n 1h 37m\n	Oran Zegman	https://www.themoviedb.org/movie/929170
497	Fantastic Beasts and Where to Find Them	73	Action, Adventure, Fantasy	11/18/2016	\n 2h 12m\n	David Yates	https://www.themoviedb.org/movie/259316
498	Spider-Man: Far from Home	75	Action, Adventure, Science Fiction	07/04/2019	\n 2h 9m\n	Jon Watts	https://www.themoviedb.org/movie/429617
499	The Swimmers	77	Drama, History	11/24/2022	\n 2h 14m\n	Sally El Hosaini	https://www.themoviedb.org/movie/821881

500 rows × 7 columns

```
In [29]: # df.to_excel('Movie_data.xlsx')     # Create all page info in excel.
```

```
In [6]: pwd  # Used PWD to check the file location.
```

```
In [ ]: 
```

```
In [ ]: 
```

```
In [ ]: 
```