# Term Project - Scraping a Webpage

In [38]:
```
!pip install lxml --user
```

```
Requirement already satisfied: lxml in c:\programdata\anaconda3\lib\site-packages
(4.3.4)
```

In [14]:
```
import requests

# Cheking the robots.txt for IMDB to verify that we are not scraping a disallowed sect
ion of the site
robot = requests.get('https://www.imdb.com/robots.txt')

print(str(robot.content))
```

```
b'# robots.txt for https://www.imdb.com properties\nUser-agent: *\nDisallow: /OnThis
Day\nDisallow: /ads/\nDisallow: /ap/\nDisallow: /mymovies/\nDisallow: /r/\nDisallow:
/register\nDisallow: /registration/\nDisallow: /search/name-text\nDisallow: /search/
title-text\nDisallow: /find\nDisallow: /find$\nDisallow: /find/\nDisallow: /tvschedu
le\nDisallow: /updates\nDisallow: /watch/_ajax/option\nDisallow: /_json/video/mon\nD
isallow: /_json/getAdsForMediaViewer/\nDisallow: /list/ls*/_ajax\nDisallow: /*/*/rg*
/mediaviewer/rm*/tr\nDisallow: /*/rg*/mediaviewer/rm*/tr\nDisallow: /*/mediaviewer/*
/tr\nDisallow: /title/tt*/mediaviewer/rm*/tr\nDisallow: /name/nm*/mediaviewer/rm*/t
r\nDisallow: /gallery/rg*/mediaviewer/rm*/tr\nDisallow: /tr/\nDisallow: /title/tt*/w
atchoptions'
```

In [35]:
```
# Finding the inner folders disallowed to scrape by the site admin
y = str(robot.content).split('\\n')
for n in y:
    if 'Disallow' in n:
        print(n)
```

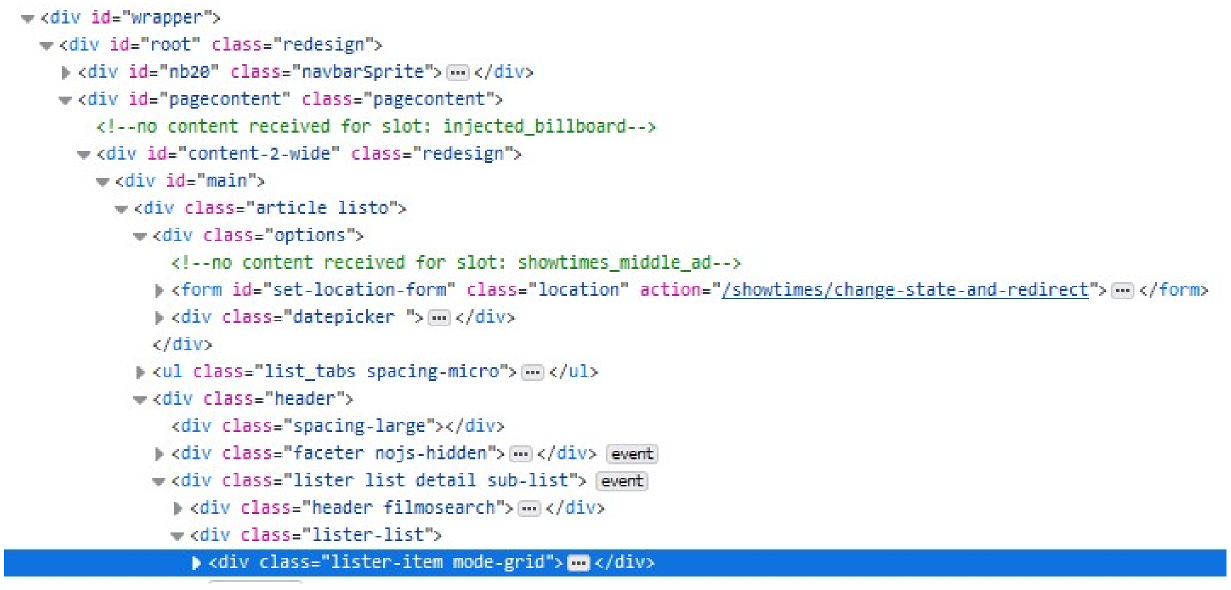
```
Disallow: /OnThisDay
Disallow: /ads/
Disallow: /ap/
Disallow: /mymovies/
Disallow: /r/
Disallow: /register
Disallow: /registration/
Disallow: /search/name-text
Disallow: /search/title-text
Disallow: /find
Disallow: /find$
Disallow: /find/
Disallow: /tvschedule
Disallow: /updates
Disallow: /watch/_ajax/option
Disallow: /_json/video/mon
Disallow: /_json/getAdsForMediaViewer/
Disallow: /list/ls*/_ajax
Disallow: /*/*/rg*/mediaviewer/rm*/tr
Disallow: /*/rg*/mediaviewer/rm*/tr
Disallow: /*/mediaviewer/*/tr
Disallow: /title/tt*/mediaviewer/rm*/tr
Disallow: /name/nm*/mediaviewer/rm*/tr
Disallow: /gallery/rg*/mediaviewer/rm*/tr
Disallow: /tr/
Disallow: /title/tt*/watchoptions'
```

In [114]:
```python
from lxml import html

resp = requests.get('https://www.imdb.com/showtimes/location?ref_=sh_lc')
page = html.document_fromstring(resp.content)

body = page.find('body')
divs = body.findall('div')
```

In [120]:
```python
for x in range(len(divs)):
    if divs[x].attrib['id'] == 'wrapper':
        content_div = divs[x]

print(content_div.attrib)
```

```
{'id': 'wrapper'}
```

## Here is a section of the source code for the above HTML page. First I have to get to the innermost DIV tag to get the list of movies. Then I follow each of those links and extract movie details to a dataframe.

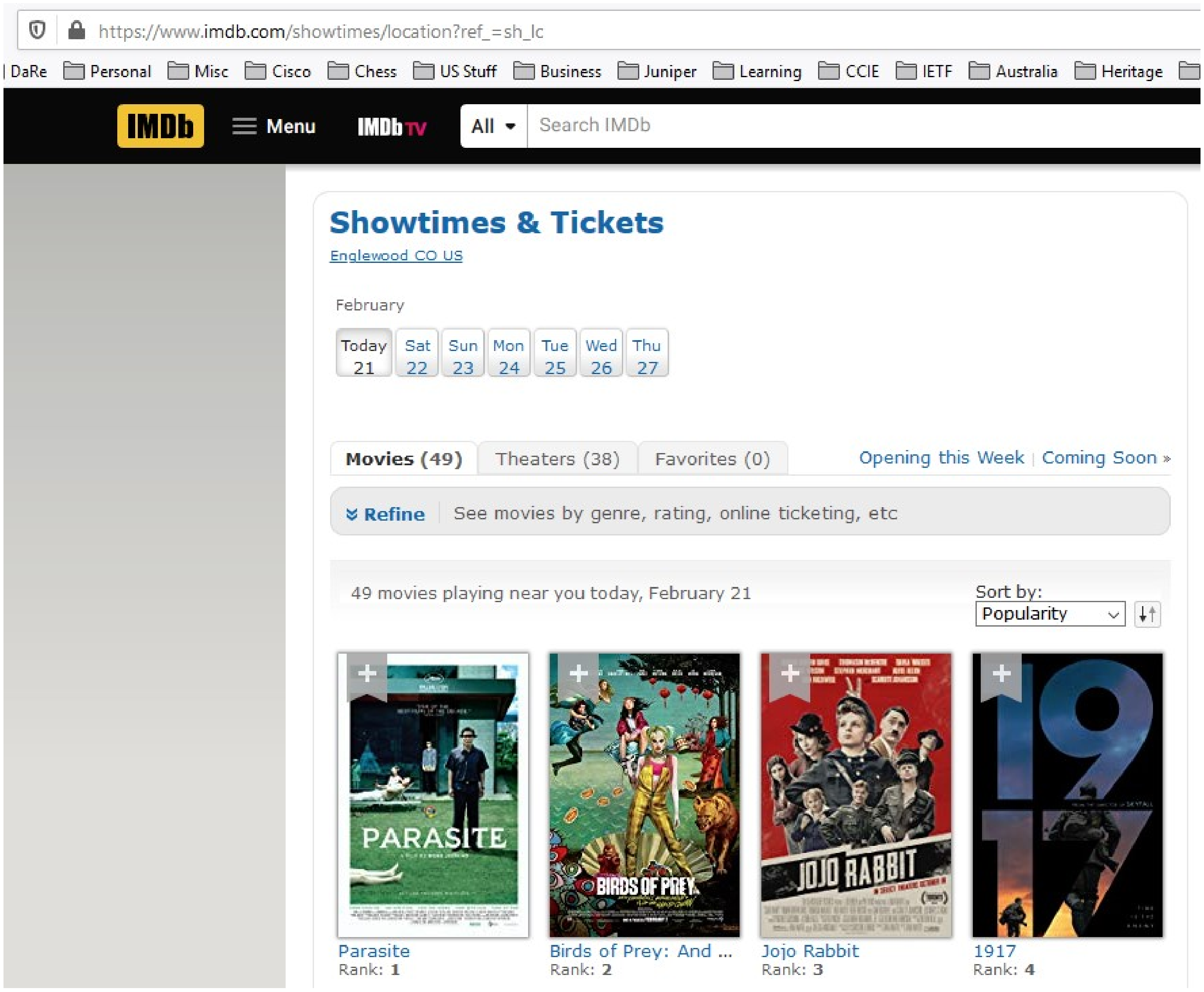


```
▾<div id="wrapper">
  ▾<div id="root" class="redesign">
    ▸<div id="nb20" class="navbarSprite">…</div>
    ▾<div id="pagecontent" class="pagecontent">
      <!--no content received for slot: injected_billboard-->
      ▾<div id="content-2-wide" class="redesign">
        ▾<div id="main">
          ▾<div class="article listo">
            ▾<div class="options">
              <!--no content received for slot: showtimes_middle_ad-->
              ▸<form id="set-location-form" class="location" action="/showtimes/change-state-and-redirect">…</form>
              ▸<div class="datepicker ">…</div>
            </div>
            ▸<ul class="list_tabs spacing-micro">…</ul>
            ▾<div class="header">
              <div class="spacing-large"></div>
              ▸<div class="faceter nojs-hidden">…</div> event
              ▾<div class="lister list detail sub-list"> event
                ▸<div class="header filmosearch">…</div>
                ▾<div class="lister-list">
                  ▸<div class="lister-item mode-grid">…</div>
```

In [154]:
```python
# Getting the list of movies from the 'Movies' page
temp = content_div.find('div')     # getting the 'root' div
temp = temp.findall('div')         # getting the 'pagecontent' div
temp = temp[1].find('div')         # getting the 'content-2-wide' div
temp = temp.find('div')            # getting the 'main' div
temp = temp.find('div')            # getting the 'article listo' div
temp = temp.findall('div')         # getting the 'header' div
temp = temp[1].findall('div')      # getting the 'lister list detail sublist' div
temp = temp[2].findall('div')      # getting the 'lister list' div
movie_div_list = temp[1].findall('div')

# Checking how many movies are listed
print(len(movie_div_list))
```

```
49
```

## Above number matches with what's on the web page



https://www.imdb.com/showtimes/location?ref_=sh_lc

Showtimes & Tickets

Englewood CO US

February

49 movies playing near you today, February 21

Sort by: Popularity

Parasite
Rank: 1

Birds of Prey: And ...
Rank: 2

Jojo Rabbit
Rank: 3

1917
Rank: 4

In [299]:

```
# Creating the dataframe with required columns
import pandas as pd

movies = pd.DataFrame(columns=['Title_ID', 'Title_page', 'Name', 'Release_Date', 'Pag
e_Rank', 'User_Rating','Rating_Count',
                               'Metascore', 'Runtime', 'Genre', 'Description', 'Direc
tor', 'Director_Page', 'Stars',
                               'Star_Pages', 'Image_Url', 'MPAA_Rating'])
```

In [300]:

```
movies
```

Out[300]:

| Title_ID | Title_page | Name | Release_Date | Page_Rank | User_Rating | Rating_Count | Metascore | Runtime | Genre | D |
|---|---|---|---|---|---|---|---|---|---|---|

In [301]:
```
# Extracting information for individual movies

row = {} # defining an empty dictionary

for n in movie_div_list: # movie_div_list has 49 divs in it. One for each movie.
    sub_divs = n.findall('div')
    # Each sub_div has 3 divs in it. We will be extracting information from those.
    # This is first div
    for n in sub_divs[0].findall('span'):
        if n.attrib['name'] == "moviemeter":
            row['Page_Rank'] = n.attrib['data-value']
        elif n.attrib['name'] == "alpha":
            row['Name'] = n.attrib['data-value']
        elif n.attrib['name'] == "user_rating":
            row['User_Rating'] = n.attrib['data-value']
        elif n.attrib['name'] == "runtime":
            row['Runtime'] = n.attrib['data-value']

    # second div
    row['Title_page'] = 'https://www.imdb.com' + sub_divs[1].find('a').attrib['href']
    row['Title_ID'] = sub_divs[1].find('a').attrib['href'].split('/')[3]
    row['Image_Url'] = sub_divs[1].find('a').find('img').attrib['src']

    for x in sub_divs[1].findall('div'):
        if 'id' in x.attrib and x.attrib['id'] == 'release_date':
            row['Release_Date'] = x.find('strong').text

    # third div
    divs = sub_divs[2].findall('div')
    divs = divs[1].findall('div')

    # This part needs a try/except block as some movies don't have a metascore tag. S
o, we need to catch the exception
    try:
        row['Metascore'] = divs[2].find('span').text
    except:
        row['Metascore'] = 99

    ps = sub_divs[2].findall('p')
    span = ps[0].findall('span')

    for n in span:
        if n.attrib['class'] == "certificate":
            row['MPAA_Rating'] = n.text
        elif n.attrib['class'] == "genre":
            row['Genre'] = n.text

    row['Description'] = ps[1].text

    # This part needs a try/except block as some movies don't have a Rating Count ta
g. So, we need to catch the exception
    try:
        row['Rating_Count'] = ps[3].findall('span')[1].attrib['data-value']
    except:
        row['Rating_Count'] = 0

    anchors = ps[2].findall('a')
    Director = True
    star_list = []
    star_page_list = []

    for n in anchors:
        if Director:
            row['Director'] = n.text
            row['Director_Page'] = 'https://www.imdb.com' + n.attrib['href']
            Director = False
```

```
In [302]: movies
```

Out[302]:

| | Title_ID | Title_page | Name | Release_Date | Page_Rank | User_Rating | Rating_Count | Metascor |
|---|---|---|---|---|---|---|---|---|
| **0** | tt6751668 | https://www.imdb.com /showtimes/title /tt6751668/ | Parasite | 08 Nov 2019 | 1 | 8.6 | 260438 | 9 |
| **1** | tt7713068 | https://www.imdb.com /showtimes/title /tt7713068/ | Birds of Prey: And the Fantabulous Emancipatio... | 07 Feb 2020 | 2 | 6.6 | 45084 | 6 |
| **2** | tt2584384 | https://www.imdb.com /showtimes/title /tt2584384/ | Jojo Rabbit | 08 Nov 2019 | 3 | 8 | 129008 | 5 |
| **3** | tt8579674 | https://www.imdb.com /showtimes/title /tt8579674/ | 1917 | 10 Jan 2020 | 4 | 8.5 | 188219 | 7 |
| **4** | tt7286456 | https://www.imdb.com /showtimes/title /tt7286456/ | Joker | 04 Oct 2019 | 5 | 8.6 | 684646 | 5 |
| **5** | tt3794354 | https://www.imdb.com /showtimes/title /tt3794354/ | Sonic the Hedgehog | 14 Feb 2020 | 6 | 6.9 | 12843 | 4 |
| **6** | tt8946378 | https://www.imdb.com /showtimes/title /tt8946378/ | Knives Out | 27 Nov 2019 | 9 | 8 | 180585 | 8 |
| **7** | tt3281548 | https://www.imdb.com /showtimes/title /tt3281548/ | Little Women | 25 Dec 2019 | 10 | 8 | 61866 | 9 |
| **8** | tt1950186 | https://www.imdb.com /showtimes/title /tt1950186/ | Ford v Ferrari | 15 Nov 2019 | 12 | 8.2 | 144443 | 8 |
| **9** | tt8367814 | https://www.imdb.com /showtimes/title /tt8367814/ | The Gentlemen | 24 Jan 2020 | 14 | 8.1 | 36192 | 5 |
| **10** | tt0983946 | https://www.imdb.com /showtimes/title /tt0983946/ | Fantasy Island | 14 Feb 2020 | 16 | 4.6 | 3183 | 2 |
| **11** | tt6673612 | https://www.imdb.com /showtimes/title /tt6673612/ | Dolittle | 17 Jan 2020 | 17 | 5.5 | 14829 | 2 |
| **12** | tt6394270 | https://www.imdb.com /showtimes/title /tt6394270/ | Bombshell | 20 Dec 2019 | 21 | 6.8 | 31678 | 6 |
| | | https://www.imdb.com | Bad Boys For | | | | | |

### Looking at the dataset, it seems that there are quite a few things to fix. Below are some of the issues I noticed.

- 'Rating_Count' column had one movie without a count
- Quite a few missing values in 'Metascore' column were replaced by the arbitrary value '99'. Need to check if that values really makes sense with the rest of the data
- 'Genre' & 'Description' columns has "\n" character at the beginnning
- 'MPAA_Rating' column has some values as 'unrated' while another says 'not rated'. Need to use a single value to identify missing values.

In [303]:
```
# Since there is only one movie without a rating count, I will keep it at 0

# I have used the arbitrary value '99' to replace missing values in 'metascore' colum
n

count = 0

for x in movies['Metascore']:
    if x == 99:
        count += 1

print(count)
```
```
11
```

In [304]:
```
# Taking the average of other values

total = 0

for x in movies['Metascore']:
    if x != 99:
        total += int(x)

print("Average is ", total/38) # 11 values with '99'. Total rows are 49. 49 - 11 = 3
8.
```
```
Average is  60.13157894736842
```

In [305]:
```
# Replacing '99' in 'metascore' column with '60'

movies['Metascore'] = movies['Metascore'].replace(99, 60)

movies['Metascore']
```

Out[305]:
```
0     96
1     60
2     58
3     78
4     59
5     47
6     82
7     91
8     81
9     51
10    21
11    26
12    64
13    59
14    64
15    58
16    80
17    53
18    64
19    49
20    49
21    95
22    63
23    48
24    68
25              60
26    64
27    24
28    41
29    54
30    77
31    32
32    77
33    55
34    63
35              60
36              60
37    25
38              60
39              60
40              60
41    68
42              60
43              60
44              60
45    70
46              60
47    71
48              60
Name: Metascore, dtype: object
```

```
In [306]: # Removing "\n" character at the beginnning from 'Genre' column

          movies['Genre'] = movies['Genre'].str.lstrip('\n')

          movies['Genre']
```

```
Out[306]: 0          Comedy, Drama, Thriller
          1         Action, Adventure, Crime
          2              Comedy, Drama, War
          3                      Drama, War
          4           Crime, Drama, Thriller
          5        Action, Adventure, Comedy
          6            Comedy, Crime, Drama
          7                  Drama, Romance
          8        Action, Biography, Drama
          9                   Action, Crime
          10       Adventure, Comedy, Horror
          11       Action, Adventure, Comedy
          12               Biography, Drama
          13          Action, Comedy, Crime
          14    Animation, Adventure, Comedy
          15       Action, Adventure, Comedy
          16               Biography, Drama
          17      Action, Adventure, Fantasy
          18       Fantasy, Horror, Thriller
          19                  Comedy, Drama
          20        Adventure, Drama, Family
          21                 Drama, Romance
          22                 Drama, Romance
          23          Action, Drama, Horror
          24                          Drama
          25       Horror, Mystery, Thriller
          26         Drama, Horror, Thriller
          27                 Comedy, Family
          28                Horror, Mystery
          29    Animation, Action, Adventure
          30      Adventure, Family, Fantasy
          31                         Comedy
          32                          Drama
          33          Comedy, Romance, Sport
          34         Biography, Crime, Drama
          35                          Drama
          36                Comedy, Romance
          37    Animation, Adventure, Comedy
          38                         Comedy
          39                         Horror
          40             Drama, Reality-TV
          41      Adventure, Drama, Thriller
          42               Animation, Drama
          43                    Documentary
          44                        Romance
          45                          Drama
          46                  Action, Crime
          47                    Documentary
          48                          Drama
          Name: Genre, dtype: object
```

In [307]:
```
# Removing "\n" character at the beginnning from 'Description' column

movies['Description'] = movies['Description'].str.lstrip('\n ')

movies['Description']
```

Out[307]:
```
0     A poor family, the Kims, con their way into be...
1     After splitting with the Joker, Harley Quinn j...
2     A young boy in Hitler's army finds out his mot...
3     April 6th, 1917. As a regiment assembles to wa...
4     In Gotham City, mentally troubled comedian Art...
5     After discovering a small, blue, fast hedgehog...
6     A detective investigates the death of a patria...
7     Jo March reflects back and forth on her life, ...
8                                American car designer
9     An American expat tries to sell off his highly...
10    A horror adaptation of the popular '70s TV sho...
11    A physician who can talk to animals embarks on...
12              A group of women take on Fox News head
13    The Bad Boys Mike Lowrey and Marcus Burnett ar...
14    Anna, Elsa, Kristoff, Olaf and Sven leave Aren...
15    In Jumanji: The Next Level, the gang is back b...
16    Based on the true story of a real-life friends...
17    The surviving members of the resistance face t...
18    A long time ago in a distant fairy tale countr...
19    Barely escaping an avalanche during a family s...
20    A sled dog struggles for survival in the wilds...
21    On an isolated island in Brittany at the end o...
22    A series of intertwining love stories set in t...
23    A crew of aquatic researchers work to get to s...
24    World-renowned civil rights defense attorney B...
25    After a family moves into the Heelshire Mansio...
26    A soon-to-be stepmom is snowed in with her fia...
27    A crew of rugged firefighters meet their match...
28    A house is cursed by a vengeful ghost that doo...
29    When the world's best spy is turned into a pig...
30    While home sick in bed, a young boy's grandfat...
31    Two friends with very different ideals start a...
32    A searing look at a day in the life of an assi...
33    In the Olympic Athlete Village, a young cross-...
34    The real life of Tommaso Buscetta, the so-call...
35    Neena is a French teacher and single parent to...
36    The road to achieving a happy ending is a litt...
37    Animated feature film inspired by the Playmobi...
38    A woman's island getaway with her boyfriend is...
39    Part 1 of 2 Part Horror film Starring Vicky Ka...
40    2020 Oscar Nominated Short Films Live Action: ...
41    Four unfortunate men from different parts of t...
42    Violet Evergarden, a former soldier returned f...
43    Pushed to his breaking point, a master welder ...
44    The film revolves around Bheeshma, a man makin...
45    A teenager in a family shelter, wages war agai...
46    Mafia Chapter 1 is a Tamil drama starring Arun...
47    A profile of giraffe researcher Anne Dagg who,...
48
Name: Description, dtype: object
```

In [310]:
```
# Standardizing 'MPAA_Rating' column with a single value for unrated movies

for x in movies['MPAA_Rating']:
    if x not in ['R', 'PG', 'PG-13']:
        print(x)
```

```
Not Rated
Unrated
Unrated
Unrated
Not Rated
```

In [311]:
```
# We will use the value 'Unrated' for all movies that are not rated.
# Converting 2 "Not Rated" values with "Unrated"

movies['MPAA_Rating'] = movies['MPAA_Rating'].replace('Not Rated', 'Unrated')

movies['MPAA_Rating']
```

Out[311]:
```
0            R
1            R
2        PG-13
3            R
4            R
5           PG
6        PG-13
7           PG
8        PG-13
9            R
10       PG-13
11          PG
12           R
13           R
14          PG
15       PG-13
16          PG
17       PG-13
18       PG-13
19           R
20          PG
21           R
22       PG-13
23       PG-13
24       PG-13
25       PG-13
26           R
27          PG
28           R
29          PG
30          PG
31           R
32           R
33       PG-13
34           R
35           R
36           R
37          PG
38           R
39           R
40     Unrated
41          PG
42          PG
43          PG
44          PG
45     Unrated
46     Unrated
47     Unrated
48     Unrated
Name: MPAA_Rating, dtype: object
```

In [313]:
```
# Checking for null values
movies.isnull().sum()
```

Out[313]:
```
Title_ID         0
Title_page       0
Name             0
Release_Date     0
Page_Rank        0
User_Rating      0
Rating_Count     0
Metascore        0
Runtime          0
Genre            0
Description      0
Director         0
Director_Page    0
Stars            0
Star_Pages       0
Image_Url        0
MPAA_Rating      0
dtype: int64
```

In [334]:
```
# Checking for empty strings as they are not interpreted as NULL
for x in movies.iterrows():
    for y in range(len(x[1])):
        if x[1][y] == '':
            print(x[1])
```

```
Title_ID                                                 tt0572720
Title_page         https://www.imdb.com/showtimes/title/tt0572720/
Name                                                   Kwaad bloed
Release_Date                                           02 Feb 2002
Page_Rank                                                  1000000
User_Rating                                                      0
Rating_Count                                                     0
Metascore                                                       60
Runtime                                                          0
Genre                                                Drama
Description
Director                                            Vivian Pieters
Director_Page                  https://www.imdb.com/name/nm0682859/
Stars              [Hans Ligtvoet, Janni Goslinga, Geert Lageveen...
Star_Pages         [https://www.imdb.com/name/nm0989566/, https:/...
Image_Url          https://m.media-amazon.com/images/G/01/imdb/im...
MPAA_Rating                                                Unrated
Name: 48, dtype: object
```

In [335]:
```
# Seems that the 'Description' column is empty for the last movie.
# Let's replace it with "THERE IS NO DESCRIPTION AVAILABLE FOR THIS MOVIE"

movies['Description'] = movies['Description'].replace('', 'THERE IS NO DESCRIPTION AV
AILABLE FOR THIS MOVIE')

movies['Description'][48]
```

Out[335]:
```
'THERE IS NO DESCRIPTION AVAILABLE FOR THIS MOVIE'
```

```
In [336]: # Final Dataset
          movies
```

Out[336]:

| | Title_ID | Title_page | Name | Release_Date | Page_Rank | User_Rating | Rating_Count | Metascor |
|---|---|---|---|---|---|---|---|---|
| **0** | tt6751668 | https://www.imdb.com/showtimes/title/tt6751668/ | Parasite | 08 Nov 2019 | 1 | 8.6 | 260438 | 9 |
| **1** | tt7713068 | https://www.imdb.com/showtimes/title/tt7713068/ | Birds of Prey: And the Fantabulous Emancipatio... | 07 Feb 2020 | 2 | 6.6 | 45084 | 6 |
| **2** | tt2584384 | https://www.imdb.com/showtimes/title/tt2584384/ | Jojo Rabbit | 08 Nov 2019 | 3 | 8 | 129008 | 5 |
| **3** | tt8579674 | https://www.imdb.com/showtimes/title/tt8579674/ | 1917 | 10 Jan 2020 | 4 | 8.5 | 188219 | 7 |
| **4** | tt7286456 | https://www.imdb.com/showtimes/title/tt7286456/ | Joker | 04 Oct 2019 | 5 | 8.6 | 684646 | 5 |
| **5** | tt3794354 | https://www.imdb.com/showtimes/title/tt3794354/ | Sonic the Hedgehog | 14 Feb 2020 | 6 | 6.9 | 12843 | 4 |
| **6** | tt8946378 | https://www.imdb.com/showtimes/title/tt8946378/ | Knives Out | 27 Nov 2019 | 9 | 8 | 180585 | 8 |
| **7** | tt3281548 | https://www.imdb.com/showtimes/title/tt3281548/ | Little Women | 25 Dec 2019 | 10 | 8 | 61866 | 9 |
| **8** | tt1950186 | https://www.imdb.com/showtimes/title/tt1950186/ | Ford v Ferrari | 15 Nov 2019 | 12 | 8.2 | 144443 | 8 |
| **9** | tt8367814 | https://www.imdb.com/showtimes/title/tt8367814/ | The Gentlemen | 24 Jan 2020 | 14 | 8.1 | 36192 | 5 |
| **10** | tt0983946 | https://www.imdb.com/showtimes/title/tt0983946/ | Fantasy Island | 14 Feb 2020 | 16 | 4.6 | 3183 | 2 |
| **11** | tt6673612 | https://www.imdb.com/showtimes/title/tt6673612/ | Dolittle | 17 Jan 2020 | 17 | 5.5 | 14829 | 2 |
| **12** | tt6394270 | https://www.imdb.com/showtimes/title/tt6394270/ | Bombshell | 20 Dec 2019 | 21 | 6.8 | 31678 | 6 |
| **13** | tt1502397 | https://www.imdb.com/showtimes/title/tt1502397/ | Bad Boys For Life | 17 Jan 2020 | 25 | 7.2 | 39493 | 5 |

## Appendix

In [211]:
```
# Finding which movie doesn't have the 'metascore' div tag
x = 0
for n in movie_div_list:
    x += 1
    print(x)
    sub_divs = n.findall('div')
    divs = sub_divs[2].findall('div')
    divs = divs[1].findall('div')
    print(divs[2].attrib)
```

```
1
{'class': 'inline-block ratings-metascore'}
2
{'class': 'inline-block ratings-metascore'}
3
{'class': 'inline-block ratings-metascore'}
4
{'class': 'inline-block ratings-metascore'}
5
{'class': 'inline-block ratings-metascore'}
6
{'class': 'inline-block ratings-metascore'}
7
{'class': 'inline-block ratings-metascore'}
8
{'class': 'inline-block ratings-metascore'}
9
{'class': 'inline-block ratings-metascore'}
10
{'class': 'inline-block ratings-metascore'}
11
{'class': 'inline-block ratings-metascore'}
12
{'class': 'inline-block ratings-metascore'}
13
{'class': 'inline-block ratings-metascore'}
14
{'class': 'inline-block ratings-metascore'}
15
{'class': 'inline-block ratings-metascore'}
16
{'class': 'inline-block ratings-metascore'}
17
{'class': 'inline-block ratings-metascore'}
18
{'class': 'inline-block ratings-metascore'}
19
{'class': 'inline-block ratings-metascore'}
20
{'class': 'inline-block ratings-metascore'}
21
{'class': 'inline-block ratings-metascore'}
22
{'class': 'inline-block ratings-metascore'}
23
{'class': 'inline-block ratings-metascore'}
24
{'class': 'inline-block ratings-metascore'}
25
{'class': 'inline-block ratings-metascore'}
26

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-211-f1d20f61feb4> in <module>
      7     divs = sub_divs[2].findall('div')
      8     divs = divs[1].findall('div')
----> 9     print(divs[2].attrib)

IndexError: list index out of range
```

In [214]:
```
# Finding which movie doesn't have the 'Rating Count' div tag
x = 0
for n in movie_div_list:
    x += 1
    print(x)
    sub_divs = n.findall('div')
    ps = sub_divs[2].findall('p')
    print(ps[3].findall('span')[1].attrib)
```

```
1
{'name': 'nv', 'data-value': '260438'}
2
{'name': 'nv', 'data-value': '45084'}
3
{'name': 'nv', 'data-value': '129008'}
4
{'name': 'nv', 'data-value': '188219'}
5
{'name': 'nv', 'data-value': '684646'}
6
{'name': 'nv', 'data-value': '12843'}
7
{'name': 'nv', 'data-value': '180585'}
8
{'name': 'nv', 'data-value': '61866'}
9
{'name': 'nv', 'data-value': '144443'}
10
{'name': 'nv', 'data-value': '36192'}
11
{'name': 'nv', 'data-value': '3183'}
12
{'name': 'nv', 'data-value': '14829'}
13
{'name': 'nv', 'data-value': '31678'}
14
{'name': 'nv', 'data-value': '39493'}
15
{'name': 'nv', 'data-value': '72932'}
16
{'name': 'nv', 'data-value': '82085'}
17
{'name': 'nv', 'data-value': '26241'}
18
{'name': 'nv', 'data-value': '257249'}
19
{'name': 'nv', 'data-value': '3904'}
20
{'name': 'nv', 'data-value': '892'}
21
{'name': 'nv', 'data-value': '549'}
22
{'name': 'nv', 'data-value': '16349'}
23
{'name': 'nv', 'data-value': '752'}
24
{'name': 'nv', 'data-value': '9414'}
25
{'name': 'nv', 'data-value': '10721'}
26
{'name': 'nv', 'data-value': '206'}
27
{'name': 'nv', 'data-value': '2050'}
28
{'name': 'nv', 'data-value': '5552'}
29
{'name': 'nv', 'data-value': '6268'}
30
{'name': 'nv', 'data-value': '6690'}
31
{'name': 'nv', 'data-value': '373004'}
32
{'name': 'nv', 'data-value': '2143'}
33
{'name': 'nv', 'data-value': '616'}
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-214-5b216f71d537> in <module>
      6     sub_divs = n.findall('div')
      7     ps = sub_divs[2].findall('p')
----> 8     print(ps[3].findall('span')[1].attrib)

IndexError: list index out of range
```

```
In [ ]:
```