importing required libraries for webscraping

In [12]:

```
import numpy as np
import pandas as pd
from bs4 import BeautifulSoup
import requests
from urllib.request import urlretrieve,urlopen,Request
```

extracting url

In [2]:

```
url = "https://m.imdb.com/chart/top/?ref_=nv_mv_250"
url
```

Out[2]:

```
'https://m.imdb.com/chart/top/?ref_=nv_mv_250'
```

In [3]:

```
response = requests.get(url)
response
```

Out[3]:

```
<Response [403]>
```

trying to fix the response code 403

In [4]:

```
 fix_response = {"User_Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64)
```

In [5]:

```
response = requests.get(url,headers =  fix_response)
response
```

Out[5]:

```
<Response [200]>
```

extract data from html using beautiful soup

In [13]:
```python
soup = BeautifulSoup(response.content,'html.parser')
soup
```

Out[13]:
```
<!DOCTYPE html>
<html lang="en-US" xmlns:fb="http://www.facebook.com/2008/fbml" xmlns:og
="http://opengraphprotocol.org/schema/"><head><meta charset="utf-8"/><meta
content="width=device-width" name="viewport"/><script>if(typeof uet === 'f
unction'){ uet('bb', 'LoadTitle', {wb: 1}); }</script><script>window.addEv
entListener('load', (event) => {
        if (typeof window.csa !== 'undefined' && typeof window.csa === 'fu
nction') {
            var csaLatencyPlugin = window.csa('Content', {
                element: {
                    slotId: 'LoadTitle',
                    type: 'service-call'
                }
            });
            csaLatencyPlugin('mark', 'clickToBodyBegin', 1701194537971);
        }
    })</script><title>IMDb Top 250 Movies</title><meta content="As rated b
y regular IMDb voters." data-id="main" name="description"/><meta content
="IMDb" property="og:site_name"/><meta content="IMDb Top 250 Movies" prope
```

In [14]:
```python
soup.find("title").getText()
```

Out[14]:
```
'IMDb Top 250 Movies'
```

extracting movie names

In [143]:

```python
import re
top_250 = []
movie_name = soup.findAll('h3')
for movies in movie_name:
    x=movies.getText()
    top_250.append(x)

top_250= [re.sub(r'\d+\. ', '', item) for item in top_250]

print(top_250)
```

```
['IMDb Charts', 'The Shawshank Redemption', 'The Godfather', 'The Dark Knigh
t', 'The Godfather: Part II', '12 Angry Men', "Schindler's List", 'The Lord o
f the Rings: The Return of the King', 'Pulp Fiction', 'The Lord of the Rings:
The Fellowship of the Ring', 'Il Buono, Il Brutto, Il Cattivo', 'Forrest Gum
p', 'Fight Club', 'The Lord of the Rings: The Two Towers', 'Inception', 'Star
Wars: Episode V - The Empire Strikes Back', 'The Matrix', 'GoodFellas', "One
Flew Over the Cuckoo's Nest", 'Se7en', "It's a Wonderful Life", 'Shichinin No
Samurai', 'Interstellar', 'The Silence of the Lambs', 'Saving Private Ryan',
'City of God', 'Life Is Beautiful', 'Spider-man: Across the Spider-verse', 'T
he Green Mile', 'Star Wars: Episode IV - A New Hope', 'Terminator 2: Judgment
Day', 'Back to the Future', 'Spirited Away', 'The Pianist', 'Psycho', 'Parasi
te', 'Gladiator', 'The Lion King', 'Léon', 'American History X', 'The Departe
d', 'Whiplash', 'The Prestige', 'The Usual Suspects', 'Grave of the Fireflie
s', 'Seppuku', 'Casablanca', 'Intouchables', 'Modern Times', 'Cinema Paradis
o', "C'era Una Volta Il West", 'Rear Window', 'Alien', 'City Lights', 'Apocal
ypse Now', 'Django Unchained', 'Memento', 'Raiders of the Lost Ark', 'WALL·
E', 'Das Leben der Anderen', 'Oppenheimer', 'Sunset Blvd.', 'Paths of Glory',
'Avengers: Infinity War', 'The Shining', 'The Great Dictator', 'Spider-Man: I
nto the Spider-Verse', 'Witness for the Prosecution', 'Alien 2', 'Inglourious
Basterds', 'The Dark Knight Rises', 'American Beauty', 'Dr. Strangelove or: H
ow I Learned to Stop Worrying and Love the Bomb', 'Oldeuboi', 'Coco', 'Amadeu
s', 'Toy Story', 'Das Boot', 'Braveheart', 'Avengers: Endgame', 'Joker', 'Mon
onoke-hime', 'Good Will Hunting', 'Kimi No Na Wa.', 'Once Upon a Time in Amer
ica', 'Tengoku to Jigoku', '3 Idiots', "Singin' in the Rain", 'Capharnaüm',
'Requiem for a Dream', 'Idi I Smotri', 'Toy Story 3', 'Star Wars: Episode VI
- Return of the Jedi', 'Eternal Sunshine of the Spotless Mind', '2001: A Spac
e Odyssey', 'Jagten', 'Reservoir Dogs', 'Ikiru', 'Lawrence of Arabia', 'The A
partment', 'Citizen Kane', 'M - Eine Stadt sucht einen Mörder', 'North by Nor
thwest', 'Vertigo', 'Double Indemnity', "Le fabuleux destin d'Amélie Poulai
n", 'Scarface', 'Full Metal Jacket', 'A Clockwork Orange', 'Incendies', 'Hea
t', 'Up', 'To Kill a Mockingbird', 'Hamilton', 'The Sting', 'Jodaeiye Nader A
z Simin', 'Indiana Jones and the Last Crusade', 'Metropolis', 'Die Hard', 'Ta
are Zameen Par', 'Snatch', 'Ladri Di Biciclette', 'L.A. Confidential', 'Taxi
Driver', '1917', 'Der Untergang', 'Dangal', 'Per qualche dollaro in più', 'Ba
tman Begins', 'Top Gun: Maverick', 'Some Like It Hot', 'The Kid', 'The Wolf o
f Wall Street', 'The Father', 'Green Book', 'All About Eve', 'Judgment at Nur
emberg', 'The Truman Show', 'There Will Be Blood', 'Casino', 'Shutter Islan
d', 'Ran', 'El Laberinto Del Fauno', 'Jurassic Park', 'The Sixth Sense', 'Unf
orgiven', 'A Beautiful Mind', 'No Country for Old Men', 'The Treasure of the
Sierra Madre', 'Yôjinbô', 'Kill Bill: Vol. 1', 'The Thing', 'Monty Python and
the Holy Grail', 'The Great Escape', 'Finding Nemo', 'Rashōmon', 'The Elephan
t Man', 'Chinatown', 'Hauru No Ugoku Shiro', 'Dial M for Murder', 'Gone with
the Wind', 'V for Vendetta', 'Prisoners', 'Raging Bull', 'Lock, Stock and Two
Smoking Barrels', 'El Secreto De Sus Ojos', 'Inside Out', 'Spider-Man: No Way
Home', 'Three Billboards Outside Ebbing, Missouri', 'Trainspotting', 'The Bri
dge on the River Kwai', 'Fargo', 'Warrior', 'Catch Me If You Can', 'Gran Tori
no', 'My Neighbour Totoro', 'Klaus', 'Million Dollar Baby', 'Harry Potter and
the Deathly Hallows: Part 2', 'Bacheha-Ye Aseman', 'Blade Runner', '12 Years
a Slave', 'Before Sunrise', 'The Grand Budapest Hotel', 'Ben-Hur', 'The Gold
Rush', 'Gone Girl', 'Barry Lyndon', 'Hacksaw Ridge', 'In the Name of the Fath
er', 'On the Waterfront', 'Salinui Chueok', 'The General', 'The Deer Hunter',
'Smultronstället', 'Relatos Salvajes', 'The Third Man', 'Dead Poets Society',
'Le Salaire De La Peur', 'Sherlock Jr.', 'Mad Max: Fury Road', 'Monsters, In
c.', 'Mr. Smith Goes to Washington', 'Jaws', 'How to Train Your Dragon', 'Mar
y and Max', 'Ford v. Ferrari', 'Det Sjunde Inseglet', 'Room', 'The Big Lebows
ki', 'Ratatouille', 'Tokyo Story', 'Rocky', 'Hotel Rwanda', 'Logan', 'Spotlig
ht', 'Platoon', "La passion de Jeanne d'Arc", 'The Terminator', 'Jai Bhim',
'Before Sunset', 'Rush', 'Network', 'The Best Years of Our Lives', 'The Exorc
ist', 'Stand by Me', 'La haine', 'Pirates of the Caribbean: The Curse of the
Black Pearl', 'The Wizard of Oz', 'The Incredibles', 'Into the Wild', "Hachi:
A Dog's Tale", 'To Be or Not to Be', 'Ah-ga-ssi', 'My Father and My Son', 'La
battaglia di Algeri', 'Groundhog Day', 'The Grapes of Wrath', 'Amores perro
s', 'The Sound of Music', 'Rebecca', 'Cool Hand Luke', 'The Iron Giant', 'Pat
her Panchali', 'It Happened One Night', 'The Help', 'The 400 Blows', 'Aladdi
n', 'Dances with Wolves', 'Life of Brian', 'Persona', 'You have rated', 'More
to explore', 'Charts', 'Top Box Office (US)', 'Most Popular Movies', 'Top Rat
ed English Movies', 'Most Popular TV Shows', 'Top 250 TV Shows', 'Lowest Rate
d Movies', 'Most Popular Celebs', 'Top Rated Movies by Genre', 'Recently view
ed']
```

extracting release year

In [136]:

```python
year=[]
release = soup.findAll('span',{'class':'sc-479faa3c-8 bNrEFi cli-title-met
for i in release:
        x= i.getText()
        year.append(x)
years = [item for item in year if item.isdigit()]

print(years[0:201])

```

```
['1994', '1972', '2008', '1974', '1957', '1993', '2003', '1994', '2001', '196
6', '1994', '1999', '2002', '2010', '1980', '1999', '1990', '1975', '1995',
'1946', '1954', '2014', '1991', '1998', '2002', '1997', '2023', '1999', '197
7', '1991', '1985', '2001', '2002', '13', '1960', '2019', '2000', '1994', '19
94', '1998', '2006', '2014', '2006', '1995', '1988', '1962', '1942', '2011',
'1936', '1988', '1968', '1954', '1979', '1931', '1979', '2012', '2000', '198
1', '2008', '2006', '2023', '1950', '1957', '2018', '1980', '1940', '2018',
'1957', '1986', '2009', '2012', '1999', '1964', '2003', '2017', '1984', '199
5', '1981', '1995', '2019', '2019', '1997', '1997', '2016', '1984', '1963',
'2009', '1952', '2018', '2000', '1985', '2010', '1983', '2004', '1968', '201
2', '7', '1992', '1952', '1962', '1960', '1941', '1931', '1959', '1958', '194
4', '2001', '1983', '1987', '1971', '2010', '18', '1995', '2009', '1962', '20
20', '1973', '2011', '7', '1989', '1927', '1988', '2007', '2000', '1948', '19
97', '1976', '2019', '2004', '2016', '1965', '2005', '2022', '1959', '1921',
'2013', '2020', '2018', '1950', '1961', '1998', '2007', '1995', '2010', '198
5', '2006', '1993', '1999', '1992', '2001', '2007', '18', '1948', '1961', '20
03', '1982', '1975', '1963', '2003', '1950', '1980', '1974', '2004', '1954',
'1939', '2005', '2013', '1980', '1998', '2009', '2015', '2021', '2017', '199
6', '1957', '1996', '2011', '2002', '2008', '1988', '2019', '7', '2004', '201
1', '1997', '1982', '2013', '1995', '16', '2014', '1959', '1925', '2014', '19
75', '2016', '1993', '1954', '2003', '1926', '1978', '1957']
```

extracting run time

In [137]:

```python
import re
duration=[]
time = soup.findAll('span',{'class':'sc-479faa3c-8 bNrEFi cli-title-metada
for i in time:
         x= i.getText()
         duration.append(x)
Duration = [item for item in duration if re.match(r'\d+h \d+m', item)]

print(Duration[0:201])

```

```
['2h 22m', '2h 55m', '2h 32m', '3h 22m', '1h 36m', '3h 15m', '3h 21m', '2h 34
m', '2h 58m', '2h 41m', '2h 22m', '2h 19m', '2h 59m', '2h 28m', '2h 4m', '2h
16m', '2h 25m', '2h 13m', '2h 7m', '2h 10m', '3h 27m', '2h 49m', '1h 58m', '2
h 49m', '2h 10m', '1h 56m', '2h 20m', '3h 9m', '2h 1m', '2h 17m', '1h 56m',
'2h 5m', '2h 30m', '1h 49m', '2h 12m', '2h 35m', '1h 28m', '1h 50m', '1h 59
m', '2h 31m', '1h 46m', '2h 10m', '1h 46m', '1h 29m', '2h 13m', '1h 42m', '1h
52m', '1h 27m', '2h 35m', '2h 46m', '1h 52m', '1h 57m', '1h 27m', '2h 27m',
'2h 45m', '1h 53m', '1h 55m', '1h 38m', '2h 17m', '1h 50m', '1h 28m', '2h 29
m', '2h 26m', '2h 5m', '1h 57m', '1h 56m', '2h 17m', '2h 33m', '2h 44m', '2h
2m', '1h 35m', '1h 41m', '1h 45m', '2h 40m', '1h 21m', '2h 29m', '2h 58m', '3
h 1m', '2h 2m', '2h 14m', '2h 6m', '1h 46m', '3h 49m', '2h 23m', '2h 50m', '1
h 43m', '2h 6m', '1h 42m', '2h 22m', '1h 43m', '2h 11m', '1h 48m', '2h 29m',
'1h 55m', '1h 39m', '2h 23m', '3h 38m', '2h 5m', '1h 59m', '1h 57m', '2h 16
m', '2h 8m', '1h 47m', '2h 2m', '2h 50m', '1h 56m', '2h 16m', '2h 11m', '2h 5
0m', '1h 36m', '2h 9m', '2h 40m', '2h 9m', '2h 3m', '2h 7m', '2h 33m', '2h 12
m', '2h 42m', '1h 44m', '1h 29m', '2h 18m', '1h 54m', '1h 59m', '2h 36m', '2h
41m', '2h 12m', '2h 20m', '2h 10m', '2h 1m', '1h 8m', '1h 37m', '2h 10m', '2h
18m', '2h 59m', '1h 43m', '2h 38m', '2h 58m', '2h 18m', '2h 40m', '1h 58m',
'2h 7m', '1h 47m', '2h 10m', '2h 15m', '2h 2m', '2h 6m', '1h 50m', '1h 51m',
'1h 49m', '1h 31m', '2h 52m', '1h 40m', '1h 28m', '2h 4m', '2h 10m', '1h 59
m', '1h 45m', '3h 58m', '2h 12m', '2h 33m', '2h 9m', '1h 47m', '2h 9m', '1h 3
5m', '2h 28m', '1h 55m', '1h 33m', '2h 41m', '1h 38m', '2h 20m', '2h 21m', '1
h 56m', '1h 26m', '1h 36m', '2h 12m', '2h 10m', '1h 29m', '1h 57m', '2h 14m',
'1h 41m', '1h 39m', '3h 32m', '1h 35m', '2h 29m', '3h 5m', '2h 19m', '2h 13
m', '1h 48m', '2h 11m', '1h 18m', '3h 3m', '1h 31m', '2h 2m', '1h 44m', '2h 8
m', '2h 11m', '1h 32m', '2h 9m', '2h 4m', '1h 38m', '1h 32m']
```

extracting certification/rating category

In [138]:

```python
certificate=[]
certified = soup.findAll('span',{'class':'sc-479faa3c-8 bNrEFi cli-title-m
for i in certified:
         x= i.getText()
         certificate.append(x)


certifications = [item for item in certificate if item.isalpha()]

print(certifications[0:201])
```

```
['A', 'A', 'UA', 'A', 'U', 'A', 'U', 'A', 'U', 'A', 'UA', 'A', 'UA', 'UA', 'U
A', 'A', 'A', 'A', 'A', 'PG', 'U', 'UA', 'A', 'A', 'A', 'U', 'U', 'UA', 'U',
'A', 'U', 'U', 'A', 'A', 'UA', 'U', 'A', 'R', 'A', 'A', 'U', 'A', 'U', 'U',
'UA', 'G', 'U', 'U', 'U', 'R', 'G', 'R', 'A', 'UA', 'A', 'U', 'A', 'R', 'Pass
ed', 'A', 'UA', 'A', 'G', 'U', 'U', 'U', 'A', 'UA', 'UA', 'A', 'A', 'U', 'P
G', 'U', 'A', 'UA', 'A', 'U', 'U', 'U', 'A', 'UA', 'G', 'A', 'A', 'A', 'U',
'U', 'UA', 'U', 'U', 'U', 'U', 'UA', 'Passed', 'U', 'A', 'Passed', 'U', 'A',
'UA', 'A', 'A', 'U', 'U', 'U', 'U', 'A', 'U', 'UA', 'U', 'A', 'A', 'R', 'UA',
'U', 'U', 'UA', 'UA', 'U', 'Passed', 'A', 'UA', 'UA', 'Passed', 'A', 'U',
'A', 'A', 'A', 'R', 'UA', 'A', 'A', 'UA', 'Passed', 'U', 'A', 'A', 'U', 'U',
'U', 'U', 'UA', 'UA', 'U', 'A', 'U', 'UA', 'A', 'A', 'A', 'R', 'U', 'UA',
'A', 'A', 'U', 'A', 'UA', 'A', 'UA', 'U', 'UA', 'UA', 'PG', 'UA', 'A', 'UA',
'U', 'Passed', 'A', 'U', 'A', 'UA', 'A', 'UA', 'Passed', 'A', 'U', 'U', 'U',
'U', 'Passed', 'UA', 'U', 'U', 'A', 'U', 'U', 'UA', 'A', 'U', 'UA', 'U', 'U',
'U', 'UA', 'A', 'A', 'UA']
```

extracting ratings

In [139]:

```
rating=[]
rate = soup.findAll('div',{'class':"sc-e3e7b191-0 jlKVfJ sc-479faa3c-2 eUI
for i in rate:
    x = i.getText().replace('\xa0',' ').replace('Rate','')
    rating.append(x)
rating[0:201]
```

Out[139]:

```
['9.3 (2.8M)',
 '9.2 (2M)',
 '9.0 (2.8M)',
 '9.0 (1.3M)',
 '9.0 (842K)',
 '9.0 (1.4M)',
 '9.0 (1.9M)',
 '8.9 (2.2M)',
 '8.8 (2M)',
 '8.8 (796K)',
 '8.8 (2.2M)',
 '8.8 (2.3M)',
 '8.8 (1.7M)',
 '8.8 (2.5M)',
 '8.7 (1.4M)',
 '8.7 (2M)',
 '8.7 (1.2M)',
 '8.7 (1.1M)',
 '8.6 (1.8M)',
```

creating a dataframe imdb_top_200 using all the extracted information

In [148]:

```
df =  pd.DataFrame({'movie_name':top_250[1:202],'Release_year':years[0:20
```

In [151]:

```
df.head(10)
```

Out[151]:

	movie_name	Release_year	Run_Time	Rating	Certification
0	The Shawshank Redemption	1994	2h 22m	9.3 (2.8M)	A
1	The Godfather	1972	2h 55m	9.2 (2M)	A
2	The Dark Knight	2008	2h 32m	9.0 (2.8M)	UA
3	The Godfather: Part II	1974	3h 22m	9.0 (1.3M)	A
4	12 Angry Men	1957	1h 36m	9.0 (842K)	U
5	Schindler's List	1993	3h 15m	9.0 (1.4M)	A
6	The Lord of the Rings: The Return of the King	2003	3h 21m	9.0 (1.9M)	U
7	Pulp Fiction	1994	2h 34m	8.9 (2.2M)	A
8	The Lord of the Rings: The Fellowship of the Ring	2001	2h 58m	8.8 (2M)	U
9	Il Buono, Il Brutto, Il Cattivo	1966	2h 41m	8.8 (796K)	A

In [150]:

```
df.to_csv('imdb_top_200.csv', index=False)
```

In [ ]:

```

```