In [29]:
```
# Libraries Used
import requests
from bs4 import BeautifulSoup
import numpy as np
import re
import pandas as pd
```

Scraping Data from the IMDB Website

In [14]:
```
page=requests.get("https://www.imdb.com/list/ls023757565/")
```

In [15]:
```
page
```

Out[15]:
```
<Response [200]>
```

In [16]:
```
soup=BeautifulSoup(page.text)
```

Extracting Columns From Web

In [18]:
```
movie=[]
for i in soup.find_all("h3",class_="lister-item-header"):
    a=i.text
    b=re.sub("[\n\d'.'\n\d+'()']","",a)
    movie.append(b)
```

In [20]:
```
len(movie)
```

Out[20]:
```
100
```

In [34]:
```
year=[]
for i in soup.find_all("span",class_="lister-item-year text-muted unbold"):
    y=i.text
    year.append(y)
len(year)
```

Out[34]:
```
100
```

In [64]:
```
# Cleaning Year Column By Removing () Braces
yr = []
for i in df["Release_Year"]:
    yr.append(re.sub("[()]","",i))
df["Release_Year"]=yr
```

In [70]:
```
# Cleaning Year Column By Removing "I"
y=[]
for i in df["Release_Year"]:
```

```
    i = i.replace("I","")
    y.append(i)
len(y)
```

Out[70]:
```
100
```

In [73]:
```
duration=[]
for i in soup.find_all("span",class_="runtime"):
    d=i.text
    duration.append(d)
len(duration)
```

Out[73]:
```
100
```

In [74]:
```
# Cleaning Duration Column By Removing "min"
dur=[]
for i in df["Duration(In Minutes)"]:
    dur.append(i.replace("min",""))
df["Duration(In Minutes)"] = dur
```

In [37]:
```
genre=[]
for i in soup.find_all("span",class_="genre"):
    g=i.text
    f=re.sub("[\n,']","",g)
    j=f.split()
    genre.append(j)
len(genre)
```

Out[37]:
```
100
```

In [ ]:
```

```

In [338…
```
# Cleaning Genre column By Removing "\n"
gen = []
for i in df["Genre"]:
    gen.append(i.replace("\n",""))
df["Genre"] = gen
```

In [77]:
```
rating=[]
rat=soup.find_all("span",class_="ipl-rating-star__rating")
for i in rat:
    pro=i.text
    rating.append(pro.split(",")[0].split("(")[0])
```

In [78]:
```
Name=[]
for i,char in enumerate(rating):
    if i%23==0:
        Name.append(char)
len(Name)
```

Out[78]:
```
100
```

In [40]:
```
gro=[]
for i in soup.find_all('p', class_='text-muted text-small'):
    a=i.text
    b=a.split('\n')[-2]

    gro.append(b)
```

In [41]:
```
gross=[]
for i,x in enumerate(gro):
    if i%3==2:
        gross.append(x)
```

In [42]:
```
len(gross)
```

Out[42]:
```
100
```

In [59]:
```
# Cleaning Gross column by Removing "$"
Grs=[]
for i in df["Collection(In Millions)"]:
    i = i.replace("$","")
    Grs.append(i)
len(Grs)
```

Out[59]:
```
100
```

In [61]:
```
# Cleaning Gross Column by Removing "M"
Gr=[]
for i in df["Collection(In Millions)"]:
    i = i.replace("M","")
    Gr.append(i)
len(Gr)
```

Out[61]:
```
100
```

In [43]:
```
vot=[]
for i in soup.find_all('p', class_='text-muted text-small'):
    a=i.text
    b=a.split('\n')[-4]
    vot.append(b)
```

In [44]:
```
votings=[]
for i,x in enumerate(vot):
    if i%3==2:
        votings.append(x)
```

In [283…
```
len(votings)
```

Out[283… 100

In [45]:
```
mt=[]
for i in soup.find_all("div",class_="inline-block ratings-metascore"):
    a=i.text.split('\n')[1]
    mot = a.strip()
    mt.append(mot)
len(mt)
```

Out[45]: 99

In [46]:
```
mt.extend([np.nan])
# Added Nan Value in MetaScore
```

In [47]:
```
len(mt)
```

Out[47]: 100

In [105…
```
df=pd.DataFrame({'Movie_Name':movie,
     'Release_Year':y,
     'Duration(In Minutes)':dur,
     'Genre':genre,
     'Ratings':Name,
     'Collection(In Millions)':Grs,
     'Votings':votings,
     'MetaScore':mt
    })
```

Converting dataframe with extracted file by using Pandas

In [106…
```
df
```

Out[106…

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
0	The Matrix	1999	136	[Action, Sci-Fi]	8.7	171.48	1,870,818	73
1	The Matrix Reloaded	2003	138	[Action, Sci-Fi]	7.2	281.49	587,402	62
2	The Matrix Revolutions	2003	129	[Action, Sci-Fi]	6.7	139.31	507,016	47
3	Alien	1979	117	[Horror, Sci-Fi]	8.5	78.90	860,202	89
4	Aliens	1986	137	[Action, Adventure, Sci-Fi]	8.4	85.16	703,791	84
…	…	…	…	…	…	…	…	…

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
95	The Truman Show	1998	103	[Comedy, Drama]	8.2	125.62	1,047,244	76
96	Guardians of the Galaxy	2014	121	[Action, Adventure, Comedy]	8	333.18	1,149,401	81
97	The Exorcist	1973	122	[Horror]	8.1	232.91	396,041	65
98	I Am Legend	2007	101	[Action, Drama, Sci-Fi]	7.2	256.39	743,427	79
99	Black Swan	2010	108	[Drama, Thriller]	8	106.95	751,216	NaN

100 rows × 8 columns

In [345…
```
page2=requests.get("https://www.imdb.com/list/ls023757565/?sort=list_order,asc&st_dt=&m
page2
```

Out[345…
```
<Response [200]>
```

In [177…
```
soup=BeautifulSoup(page2.text)
```

In [178…
```
movie=[]
for i in soup.find_all("h3",class_="lister-item-header"):
    a=i.text
    b=re.sub("[\n\d'.'\n\d+'()']","",a)
    movie.append(b)
len(movie)
```

Out[178…
```
100
```

In [179…
```
year=[]
for i in soup.find_all("span",class_="lister-item-year text-muted unbold"):
    y=i.text
    year.append(y)
len(year)
```

Out[179…
```
100
```

In [180…
```
yr = []
for i in df1["Release_Year"]:
    yr.append(re.sub("[()]","",i))
df1["Release_Year"]=yr
```

In [181…
```
y=[]
for i in df1["Release_Year"]:
```

```
    i = i.replace("I","")
    y.append(i)
len(y)
```

Out[181…
```
100
```

In [182…
```
duration=[]
for i in soup.find_all("span",class_="runtime"):
    d=i.text
    duration.append(d)
len(duration)
```

Out[182…
```
100
```

In [183…
```
dur=[]
for i in df1["Duration(In Minutes)"]:
    dur.append(i.replace("min",""))
df1["Duration(In Minutes)"] = dur
```

In [184…
```
genre=[]
for i in soup.find_all("span",class_="genre"):
    g=i.text
    genre.append(g)
len(genre)
```

Out[184…
```
100
```

In [185…
```
gen = []
for i in df1["Genre"]:
    gen.append(i.replace("\n",""))
df1["Genre"] = gen
```

In [186…
```
gro=[]
for i in soup.find_all('p', class_='text-muted text-small'):
    a=i.text
    b=a.split('\n')[-2]

    gro.append(b)

gross=[]
for i,x in enumerate(gro):
    if i%3==2:
        gross.append(x)

len(gross)
```

Out[186…
```
100
```

In [187…
```
Grs=[]
for i in df1["Collection(In Millions)"]:
    i = i.replace("$","")
```

```
    Grs.append(i)
len(Grs)
```

Out[187… 100

In [188…

```
Gr=[]
for i in df1["Collection(In Millions)"]:
    i = i.replace("M","")
    Gr.append(i)
len(Gr)
```

Out[188… 100

In [286…

```
vot=[]
for i in soup.find_all('p', class_='text-muted text-small'):
    a=i.text
    b=a.split('\n')[-4]

    vot.append(b)

votings=[]
for i,x in enumerate(vot):
    if i%3==2:
        votings.append(x)
len(votings)
```

Out[286… 100

In [190…

```
mt=[]
for i in soup.find_all("div",class_="inline-block ratings-metascore"):
    a=i.text.split('\n')[1]
    mot = a.strip()
    mt.append(mot)
len(mt)
```

Out[190… 97

In [191…

```
mt.extend([np.nan,np.nan,np.nan])
len(mt)
```

Out[191… 100

In [192…

```
df1=pd.DataFrame({'Movie_Name':movie,
     'Release_Year':y,
     'Duration(In Minutes)':dur,
     'Genre':gen,
     'Ratings':Name,
     'Collection(In Millions)':Gr,
     'Votings':votings,
     'MetaScore':mt
    })
```

In [193…
```
df1
```

Out[193…

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
0	Man of Steel	1980	136	Biography, Drama, Sport	8.7	171.48	757,672	55
1	The Thing	2019	138	Action, Drama, War	7.2	281.49	412,795	57
2	American Gangster	2003	129	Biography, Drama, Sport	6.7	139.31	421,342	76
3	The Banker	2004	117	Drama	8.5	78.90		59
4	Out of the Furnace	2006	137	Crime, Drama	8.4	85.16	115,216	63
...	...	...	...	...	...	...	...	...
95	Home Alone	2021	103	Action, Crime, Thriller	8.2	125.62	555,506	55
96	The Fly	2017	121	Drama, War	8	333.18	178,258	49
97	Weird Science	2018	122	Crime, Drama, Thriller	8.1	232.91	88,007	NaN
98	National Lampoons Vacation	2014	101	Adventure, Comedy, Drama	7.2	256.39	107,715	NaN
99	National Lampoons Christmas Vacation	2017	108	Horror, Mystery, Thriller	8	106.95	185,646	NaN

100 rows × 8 columns

In [346…
```
page3=requests.get("https://www.imdb.com/list/ls023757565/?sort=list_order,asc&st_dt=&m
page3
```

Out[346…
```
<Response [200]>
```

In [347…
```
soup=BeautifulSoup(page3.text)
```

In [348…
```
movie=[]
for i in soup.find_all("h3",class_="lister-item-header"):
    a=i.text
    b=re.sub("[\n\d'.'\n\d+'()']","",a)
```

```
    movie.append(b)
len(movie)
```

Out[348… 100

In [350…
```
year=[]
for i in soup.find_all("span",class_="lister-item-year text-muted unbold"):
    y=i.text
    year.append(y)
len(year)
```

Out[350… 100

In [359…
```
yr = []
for i in df2["Release_Year"]:
    yr.append(re.sub("[()]","",i))
df2["Release_Year"]=yr
```

In [ ]:
```
y=[]
for i in df2["Release_Year"]:
    i = i.replace("I","")
    y.append(i)
len(y)
```

In [351…
```
duration=[]
for i in soup.find_all("span",class_="runtime"):
    d=i.text
    duration.append(d)
len(duration)
```

Out[351… 100

In [360…
```
dur=[]
for i in df2["Duration(In Minutes)"]:
    dur.append(i.replace("min",""))
df2["Duration"] = dur
```

In [352…
```
genre=[]
for i in soup.find_all("span",class_="genre"):
    g=i.text
    genre.append(g)
len(genre)
```

Out[352… 100

In [361…
```
gen = []
for i in df2["Genre"]:
    gen.append(i.replace("\n",""))
df2["Genre"] = gen
```

In [353…
```
gro=[]
for i in soup.find_all('p', class_='text-muted text-small'):
    a=i.text
    b=a.split('\n')[-2]

    gro.append(b)

gross=[]
for i,x in enumerate(gro):
    if i%3==2:
        gross.append(x)

len(gross)
```

Out[353…
```
100
```

In [362…
```
Grs=[]
for i in df2["Collection(In Millions)"]:
    i = i.replace("$","")
    Grs.append(i)
len(Grs)
```

Out[362…
```
100
```

In [365…
```
Gr=[]
for i in df2["Collection(In Millions)"]:
    i = i.replace("M","")
    Gr.append(i)
len(Gr)
```

Out[365…
```
100
```

In [354…
```
vot=[]
for i in soup.find_all('p', class_='text-muted text-small'):
    a=i.text
    b=a.split('\n')[-4]

    vot.append(b)

votings=[]
for i,x in enumerate(vot):
    if i%3==2:
        votings.append(x)
len(votings)
```

Out[354…
```
100
```

In [355…
```
mt=[]
for i in soup.find_all("div",class_="inline-block ratings-metascore"):
    a=i.text.split('\n')[1]
    mot = a.strip()
    mt.append(mot)
len(mt)
```

Out[355… 97

In [356…
```
mt.extend([np.nan,np.nan,np.nan])
len(mt)
```

Out[356… 100

In [366…
```
df2=pd.DataFrame({'Movie_Name':movie,
     'Release_Year':yr,
     'Duration(In Minutes)':dur,
     'Genre':gen,
     'Ratings':Name,
     'Collection(In Millions)':Gr,
     'Votings':votings,
     'MetaScore':mt
    })
```

In [367…
```
df2
```

Out[367…

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
0	Raging Bull	1980	129	Biography, Drama, Sport	8.7	23.38	348,070	89
1		2019	119	Action, Drama, War	7.2	159.23	558,564	78
2	Radio	2003	109	Biography, Drama, Sport	6.7	52.28	40,698	38
3	Bullet Boy	2004	89	Drama	8.5	2,616		86
4	This Is England	2006	101	Crime, Drama	8.4	0.33	121,722	72
...	...	...	...	...	...	...	...	...
95	Wrath of Man	2021	119	Action, Crime, Thriller	8.2	166,111		68
96	Mudbound	2017	134	Drama, War	8	48,743		85
97	The Mule	2018	116	Crime, Drama, Thriller	8.1	103.80	134,569	NaN
98	Chef	2014	114	Adventure, Comedy, Drama	7.2	30.64	213,390	NaN
99	Get Outl	I 2017	104	Horror, Mystery, Thriller	8	176.04	567,120	NaN

100 rows × 8 columns

In [299…
```
page4=requests.get("https://www.imdb.com/list/ls023757565/?sort=list_order,asc&st_dt=&m
page4
```

Out[299…
```
<Response [200]>
```

In [300…
```
soup=BeautifulSoup(page4.text)
```

In [301…
```
movie=[]
for i in soup.find_all("h3",class_="lister-item-header"):
    a=i.text
    b=re.sub("[\n\d'.'\n\d+'()']","",a)
    movie.append(b)
len(movie)
```

Out[301…
```
100
```

In [302…
```
year=[]
for i in soup.find_all("span",class_="lister-item-year text-muted unbold"):
    y=i.text
    year.append(y)
len(year)
```

Out[302…
```
100
```

In [308…
```
yr = []
for i in df3["Release_Year"]:
    yr.append(re.sub("[()]","",i))
df3["Release_Year"]=yr
```

In [309…
```
y=[]
for i in df3["Release_Year"]:
    i = i.replace("I","")
    y.append(i)
len(y)
```

Out[309…
```
100
```

In [310…
```
duration=[]
for i in soup.find_all("span",class_="runtime"):
    d=i.text
    duration.append(d)
len(duration)
```

Out[310…
```
100
```

In [311…
```
dur=[]
```

```
for i in df3["Duration(In Minutes)"]:
    dur.append(i.replace("min",""))
df3["Duration"] = dur
```

In [312…

```
genre=[]
for i in soup.find_all("span",class_="genre"):
    g=i.text
    genre.append(g)
len(genre)
```

Out[312… 100

In [313…

```
gen = []
for i in df3["Genre"]:
    gen.append(i.replace("\n",""))
df3["Genre"] = gen
```

In [314…

```
gro=[]
for i in soup.find_all('p', class_='text-muted text-small'):
    a=i.text
    b=a.split('\n')[-2]

    gro.append(b)

gross=[]
for i,x in enumerate(gro):
    if i%3==2:
        gross.append(x)

len(gross)
```

Out[314… 100

In [315…

```
Grs=[]
for i in df3["Collection(In Millions)"]:
    i = i.replace("$","")
    Grs.append(i)
len(Grs)
```

Out[315… 100

In [316…

```
Gr=[]
for i in df3["Collection(In Millions)"]:
    i = i.replace("M","")
    Gr.append(i)
len(Gr)
```

Out[316… 100

In [317…

```
vot=[]
for i in soup.find_all('p', class_='text-muted text-small'):
```

```
    a=i.text
    b=a.split('\n')[-4]

    vot.append(b)

votings=[]
for i,x in enumerate(vot):
    if i%3==2:
        votings.append(x)
len(votings)
```

Out[317… 100

In [318…

```
mt=[]
for i in soup.find_all("div",class_="inline-block ratings-metascore"):
    a=i.text.split('\n')[1]
    mot = a.strip()
    mt.append(mot)
len(mt)
```

Out[318… 98

In [319…

```
mt.extend([np.nan,np.nan])
len(mt)
```

Out[319… 100

In [320…

```
df3=pd.DataFrame({'Movie_Name':movie,
     'Release_Year':y,
     'Duration(In Minutes)':dur,
     'Genre':gen,
     'Ratings':Name,
     'Collection(In Millions)':Gr,
     'Votings':votings,
     'MetaScore':mt
    })
```

In [321…

```
df3
```

Out[321…

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
0	Deliverance	2015	129	Biography, Drama, Sport	8.7	23.38	107,441	80
1	Goldfinger	2015	119	Action, Drama, War	7.2	159.23	188,485	87
2	The Omen	2015	109	Biography, Drama, Sport	6.7	52.28	117,321	62

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
3	Lone Survivor	2015	89	Drama	8.5	2,616	284,124	60
4	Wonderl	2015	101	Crime, Drama	8.4	0.33	161,012	66
...	...	...	...	...	...	...	...	...
95	Takers	2015	119	Action, Crime, Thriller	8.2	166,102	62,404	54
96	Triple	2015	134	Drama, War	8	48,743	71,934	69
97	*batteries not included	2015	116	Crime, Drama, Thriller	8.1	103.80	35,469	51
98	The Drop	2015	114	Adventure, Comedy, Drama	7.2	30.64	151,395	NaN
99	The Internl	2015	104	Horror, Mystery, Thriller	8	176.04	248,313	NaN

100 rows × 8 columns

Concating all the dataframes by using concat function

In [368...
```
dfj = pd.concat([df,df1], axis= 0)

dfj.reset_index(drop=True)
```

Out[368...

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
0	The Matrix	1999	109	Adventure, Drama, Thriller	8.7	7.06	107,441	80
1	The Matrix Reloaded	2003	110	Action, Adventure, Thriller	7.2	51.08	188,485	87
2	The Matrix Revolutions	2003	111	Horror, Mystery	6.7	4.27	117,321	62
3	Alien	1979	121	Action, Biography, Drama	8.5	125.10	284,124	60
4	Aliens	1986	113	Drama, Family	8.4	132.42	161,012	66
...	...	...	...	...	...	...	...	...

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
195	Home Alone	2021	103	Action, Crime, Thriller	8.2	125.62	555,506	55
196	The Fly	2017	121	Drama, War	8	333.18	178,258	49
197	Weird Science	2018	122	Crime, Drama, Thriller	8.1	232.91	88,007	NaN
198	National Lampoons Vacation	2014	101	Adventure, Comedy, Drama	7.2	256.39	107,715	NaN
199	National Lampoons Christmas Vacation	2017	108	Horror, Mystery, Thriller	8	106.95	185,646	NaN

200 rows × 8 columns

In [369…

```
dfj1 = pd.concat([dfj,df2], axis= 0)

dfj1.reset_index(drop=True)
```

Out[369…

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
0	The Matrix	1999	109	Adventure, Drama, Thriller	8.7	7.06	107,441	80
1	The Matrix Reloaded	2003	110	Action, Adventure, Thriller	7.2	51.08	188,485	87
2	The Matrix Revolutions	2003	111	Horror, Mystery	6.7	4.27	117,321	62
3	Alien	1979	121	Action, Biography, Drama	8.5	125.10	284,124	60
4	Aliens	1986	113	Drama, Family	8.4	132.42	161,012	66
...	...	...	...	...	...	...	...	...
295	Wrath of Man	2021	119	Action, Crime, Thriller	8.2	166,111		68
296	Mudbound	2017	134	Drama, War	8	48,743		85

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
297	The Mule	2018	116	Crime, Drama, Thriller	8.1	103.80	134,569	NaN
298	Chef	2014	114	Adventure, Comedy, Drama	7.2	30.64	213,390	NaN
299	Get Outl	I 2017	104	Horror, Mystery, Thriller	8	176.04	567,120	NaN

300 rows × 8 columns

Final Dataframe before cleaning

In [370…

```
dfj2 = pd.concat([dfj1,df3], axis= 0)

dfj2.reset_index(drop=True)
```

Out[370…

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
0	The Matrix	1999	109	Adventure, Drama, Thriller	8.7	7.06	107,441	80
1	The Matrix Reloaded	2003	110	Action, Adventure, Thriller	7.2	51.08	188,485	87
2	The Matrix Revolutions	2003	111	Horror, Mystery	6.7	4.27	117,321	62
3	Alien	1979	121	Action, Biography, Drama	8.5	125.10	284,124	60
4	Aliens	1986	113	Drama, Family	8.4	132.42	161,012	66
…	…	…	…	…	…	…	…	…
395	Takers	2015	119	Action, Crime, Thriller	8.2	166,102	62,404	54
396	Triple	2015	134	Drama, War	8	48,743	71,934	69
397	*batteries not included	2015	116	Crime, Drama, Thriller	8.1	103.80	35,469	51
398	The Drop	2015	114	Adventure, Comedy, Drama	7.2	30.64	151,395	NaN

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
399	The Internl	2015	104	Horror, Mystery, Thriller	8	176.04	248,313	NaN

400 rows × 8 columns

In [371…
```
dfj2.describe()
```

Out[371…

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
count	400	400	400	400	400	400	400	390
unique	397	70	82	83	27	302	271	64
top	The Hunger Games: Mockingjay - Part	2015	101	Drama	8.2	39.57		64
freq	2	101	14	29	44	3	35	19

In [372…
```
dfj2.dtypes
```

Out[372…
```
Movie_Name                object
Release_Year              object
Duration(In Minutes)      object
Genre                     object
Ratings                   object
Collection(In Millions)   object
Votings                   object
MetaScore                 object
dtype: object
```

Converting DataFrame into CSV File

In [ ]:
```
df.to_csv("IMDB List.csv")
```

In [32]:
```
df = pd.read_csv("IMDB List.csv")
```

In [33]:
```
df
```

Out[33]:

	Unnamed: 0	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	M
0	0	The Matrix	1980	129	Biography, Drama, Sport	8.7	171.48	348069	

	Unnamed: 0	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	M
1	1	The Matrix Reloaded	2019	119	Action, Drama, War	7.2	281.49	558558	
2	2	The Matrix Revolutions	2003	109	Biography, Drama, Sport	6.7	139.31	40698	
3	3	Alien	2004	89	Drama	8.5	78.90	0	
4	4	Aliens	2006	101	Crime, Drama	8.4	85.16	121722	
...	...	...	...	...	...	...	...	...	...
395	395	Takers	2021	119	Action, Crime, Thriller	8.2	166102.00	62404	
396	396	Triple	2017	134	Drama, War	8.0	48743.00	71934	
397	397	*batteries not included	2018	116	Crime, Drama, Thriller	8.1	103.80	35469	
398	398	The Drop	2014	114	Adventure, Comedy, Drama	7.2	30.64	151395	
399	399	The Internl	2017	104	Horror, Mystery, Thriller	8.0	176.04	248313	

400 rows × 9 columns

In [38]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Unnamed: 0              400 non-null    int64 
 1   Movie_Name              400 non-null    object
 2   Release_Year            400 non-null    int64 
 3   Duration(In Minutes)    400 non-null    int64 
 4   Genre                   400 non-null    object
 5   Ratings                 400 non-null    int32 
 6   Collection(In Millions)  400 non-null    int32 
 7   Votings                 400 non-null    int64 
 8   MetaScore               400 non-null    int64 
dtypes: int32(2), int64(5), object(2)
memory usage: 25.1+ KB
```

In [39]:
```
df.drop("Unnamed: 0",axis=1,inplace=True)
```

In [40]:
```
df
```

Out[40]:

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
0	The Matrix	1980	129	Biography, Drama, Sport	8	171	348069	89
1	The Matrix Reloaded	2019	119	Action, Drama, War	7	281	558558	78
2	The Matrix Revolutions	2003	109	Biography, Drama, Sport	6	139	40698	38
3	Alien	2004	89	Drama	8	78	0	86
4	Aliens	2006	101	Crime, Drama	8	85	121722	72
...	...	...	...	...	...	...	...	...
395	Takers	2021	119	Action, Crime, Thriller	8	166102	62404	54
396	Triple	2017	134	Drama, War	8	48743	71934	69
397	*batteries not included	2018	116	Crime, Drama, Thriller	8	103	35469	51
398	The Drop	2014	114	Adventure, Comedy, Drama	7	30	151395	0
399	The Internl	2017	104	Horror, Mystery, Thriller	8	176	248313	0

400 rows × 8 columns

In [8]:
```
df[(df["Ratings"]>9)&(df["Ratings"]<10)]
```

Out[8]:

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
37	The Godfather	2012	119	Action, Drama, Sci-Fi	9.2	134.97	569,473	70.0
137	Django Unchained	2012	175	Action, Drama, Sci-Fi	9.2	134.97	1,506,861	59.0

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
237	Looper	2012	119	Action, Drama, Sci-Fi	9.2	134.97	569,475	70.0
337	Fright Night	2012	119	Action, Drama, Sci-Fi	9.2	66.49	69,025	62.0

In [31]:
```
df[(df["Ratings"]>8)&(df["Ratings"]<9)]`
```

Out[31]:

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
0	The Matrix	1980	129	Biography, Drama, Sport	8.7	171.48	348,069	89.0
3	Alien	2004	89	Drama	8.5	78.9	0	86.0
4	Aliens	2006	101	Crime, Drama	8.4	85.16	121,722	72.0
7	Scarface	2005	116	Comedy, Romance	8.3	45.6	429,938	62.0
8	Inception	1997	150	Drama, Mystery, Sci-Fi	8.8	292.58	271,717	52.0
...	...	...	...	...	...	...	...	...
383	Jerry Maguire	1967	105	Drama	8.8	42.43	262,876	64.0
387	Cocoon	1984	116	Adventure, Mystery, Sci-Fi	8.2	40.2	63,035	53.0
393	Irrational Man	2011	101	Action, Crime, Thriller	8.5	4.11	62,984	45.0
395	Takers	2021	119	Action, Crime, Thriller	8.2	166,102	62,404	54.0
397	*batteries not included	2018	116	Crime, Drama, Thriller	8.1	103.8	35,469	51.0

180 rows × 8 columns

In [32]:
```
df[(df["Ratings"]>7)&(df["Ratings"]<8)]
```

Out[32]:

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
1	The Matrix Reloaded	2019	119	Action, Drama, War	7.2	281.49	558,558	78.0
9	Rise of the Planet of the Apes	1990	127	Drama, Fantasy, Romance	7.6	176.76	213,995	65.0
10	Dawn of the Planet of the Apes	1987	100	Drama, Music, Romance	7.6	208.55	227,221	70.0
11	War for the Planet of the Apes	1988	92	Comedy, Fantasy	7.4	146.88	294,012	50.0
15	ET the Extra-Terrestrial	1990	96	Comedy, Horror	7.9	435.11	136,310	75.0
...	...	...	...	...	...	...	...	...
390	Ouija: Origin of Evil	2019	83	Drama	7.6	5,275	63,383	33.0
391	Resident Evil	2019	122	Action, Adventure, Sci-Fi	7.5	85.71	267,261	53.0
392	Above the Rim	2020	115	Drama	7.2	76,345	15,515	62.0
394	The Woman in Black	2000	94	Biography, Crime, Drama	7.7	0.23	182,407	52.0
398	The Drop	2014	114	Adventure, Comedy, Drama	7.2	30.64	151,395	0.0

160 rows × 8 columns

In [33]:
```
df[(df["Ratings"]>6)&(df["Ratings"]<7)]
```

Out[33]:

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
2	The Matrix Revolutions	2003	109	Biography, Drama, Sport	6.7	139.31	40,698	38.0
5	Alien³	2008	116	Drama	6.4	55.47	765,172	51.0
43	Predator	2019	209	Biography, Crime, Drama	6.2	30.67	375,697	87.0

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
60	Signs	2019	113	Action, Comedy, Crime	6.8	227.97	0	92.0
67	The Conjuring: The Devil Made Me Do It	2008	92	Action, Drama, Fantasy	6.3	108,698	475,451	52.0
68	It Follows	2015	124	Biography, Drama	6.8	14.67	135,936	35.0
102	American Gangster	2003	129	Biography, Drama, Sport	6.7	139.31	421,342	76.0
105	Carlitos Way	2008	114	Drama	6.4	55.47	215,466	65.0
143	Shaun of the Dead	2019	108	Biography, Crime, Drama	6.2	30.67	549,342	59.0
160	Rogue One	2019	106	Action, Comedy, Crime	6.8	227.97	613,423	61.0
167	The Abyss	2008	112	Action, Drama, Fantasy	6.3	108,698	175,870	61.0
168	World War Z	2015	100	Biography, Drama	6.8	14.67	652,619	68.0
202	Radio	2003	109	Biography, Drama, Sport	6.7	139.31	40,698	38.0
205	Gran Torino	2008	116	Drama	6.4	55.47	765,174	51.0
243	The Irishman	2019	209	Biography, Crime, Drama	6.2	30.67	375,698	87.0
260	The Gentlemen	2019	113	Action, Comedy, Crime	6.8	227.97	0	92.0
267	Hancock	2008	92	Action, Drama, Fantasy	6.3	108,698	475,455	52.0
268	Joyl	2015	124	Biography, Drama	6.8	14.67	135,936	35.0
302	The Omen	2003	109	Biography, Drama, Sport	6.7	52.28	117,321	62.0
305	The Big Sick	2008	116	Drama	6.4	148.1	133,771	86.0

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
343	SuperI	2019	209	Biography, Crime, Drama	6.2	7	79,828	50.0
360	Kids	2019	113	Action, Comedy, Crime	6.8	323,054	78,255	63.0
367	The Hunger Games: Catching Fire	2008	92	Action, Drama, Fantasy	6.3	227.95	654,490	64.0
368	The Hunger Games: Mockingjay - Part	2015	124	Biography, Drama	6.8	56.45	447,797	65.0

In [10]:
```
tdf=df[(df["Collection(In Millions)"]>300)&(df["Collection(In Millions)"]<400)]
```

In [11]:
```
tdf
```

Out[11]:

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
27	Star Wars	2014	113	Action, Adventure, Sci-Fi	8.6	322.74	661851	85
29	Star Wars: Episode VI - Return of the Jedi	2012	110	Horror, Mystery, Thriller	8.3	309.13	245838	69
78	Forrest Gump	2019	161	Comedy, Drama	8.8	330.25	697146	91
96	Guardians of the Galaxy	2017	134	Drama, War	8.0	333.18	0	85
127	The Hobbit: An Unexpected Journey	2014	121	Action, Adventure, Sci-Fi	8.6	322.74	807325	66
129	The Hobbit: The Battle of the Five Armies	2012	131	Horror, Mystery, Thriller	8.3	309.13	517153	86
178	LucyI	2019	142	Comedy, Drama	8.8	330.25	487081	76
196	The Fly	2017	121	Drama, War	8.0	333.18	178258	49

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
227	Edge of Tomorrow	2014	113	Action, Adventure, Sci-Fi	8.6	322.74	661856	85
229	SinisterI	2012	110	Horror, Mystery, Thriller	8.3	309.13	245841	69
278	Once Upon a Time in Hollywood	2019	161	Comedy, Drama	8.8	330.25	697164	91
296	Mudbound	2017	134	Drama, War	8.0	333.18	0	85

In [58]:
```
df[(df["Collection(In Millions)"]>200)&(df["Collection(In Millions)"]<300)]
```

Out[58]:

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
1	The Matrix Reloaded	2019	119	Action, Drama, War	7	281	558558	78
8	Inception	1997	150	Drama, Mystery, Sci-Fi	8	292	271717	52
10	Dawn of the Planet of the Apes	1987	100	Drama, Music, Romance	7	208	227221	70
14	Terminator : Judgment Day	2004	96	Comedy	8	204	216317	65
22	Cast Away	1990	93	Action, Adventure, Comedy	7	233	93463	57
24	Batman Begins	1982	129	Action, Adventure, Fantasy	8	206	147841	53
28	The Empire Strikes Back	2006	120	Action, Drama, Horror	8	290	119759	53
31	Back to the Future	1992	126	Action, Crime, Fantasy	8	210	301378	68
52	Jaws	1999	122	Drama	8	260	1136372	75
59	300	2008	114	Action, Crime, Thriller	7	210	249882	60

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
60	Signs	2019	113	Action, Comedy, Crime	6	227	0	92
72	Raiders of the Lost Ark	2000	109	Action, Horror, Sci-Fi	8	248	237801	79
97	The Exorcist	2018	116	Crime, Drama, Thriller	8	232	134565	0
98	I Am Legend	2014	114	Adventure, Comedy, Drama	7	256	213386	0
101	The Thing	2019	138	Action, Drama, War	7	281	412795	57
108	Saving Private Ryan	1997	148	Drama, Mystery, Sci-Fi	8	292	1356389	50
110	Young Guns II	1987	130	Drama, Music, Romance	7	208	35990	58
114	Tenet	2004	137	Comedy	8	204	474311	58
122	Blade II	1990	143	Action, Adventure, Comedy	7	233	215823	74
124	The Lord of the Rings: The Fellowship of the Ring	1982	140	Action, Adventure, Fantasy	8	206	1809804	87
128	The Hobbit: The Desolation of Smaug	2006	124	Action, Drama, Horror	8	290	646410	59
131	Hereditary	1992	116	Action, Crime, Fantasy	8	210	297189	79
152	The Terminal	1999	124	Drama	8	260	449852	96
159	The Devils Rejects	2008	117	Action, Crime, Thriller	7	210	99371	72
160	Rogue One	2019	106	Action, Comedy, Crime	6	227	613423	61

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
172	Hell or High WaterII	2000	115	Action, Horror, Sci-Fi	8	248	227052	55
197	Weird Science	2018	122	Crime, Drama, Thriller	8	232	88007	0
198	National Lampoons Vacation	2014	101	Adventure, Comedy, Drama	7	256	107715	0
201	1917	2019	119	Action, Drama, War	7	281	558564	78
208	Contact	1997	150	Drama, Mystery, Sci-Fi	8	292	271721	52
210	Dirty DancingI	1987	100	Drama, Music, Romance	7	208	227224	70
214	Napoleon Dynamite	2004	96	Comedy	8	204	216320	65
222	Teenage Mutant Ninja Turtles	1990	93	Action, Adventure, Comedy	7	233	93464	57
224	Conan the Barbarian	1982	129	Action, Adventure, Fantasy	8	206	147842	53
228	Gwoemul	2006	120	Action, Drama, Horror	8	290	119761	53
231	Batman Returns	1992	126	Action, Crime, Fantasy	8	210	301379	68
252	American Beauty	1999	122	Drama	8	260	1136374	75
259	RocknRolla	2008	114	Action, Crime, Thriller	7	210	249883	60
260	The Gentlemen	2019	113	Action, Comedy, Crime	6	227	0	92
272	Pitch Black	2000	109	Action, Horror, Sci-Fi	8	248	237802	79
297	The Mule	2018	116	Crime, Drama, Thriller	8	232	134569	0

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
298	Chef	2014	114	Adventure, Comedy, Drama	7	256	213390	0
309	The FighterI	1990	127	Drama, Fantasy, Romance	7	217	362315	79
317	Falling Down	1984	105	Action, Comedy, Fantasy	8	238	187615	56
330	Reservoir Dogs	1989	126	Action, Adventure	7	251	996089	79
365	Juice	2018	135	Action, Adventure, Sci-Fi	7	213	27025	68
367	The Hunger Games: Catching Fire	2008	92	Action, Drama, Fantasy	6	227	654490	64

In [8]:
```
df[(df["Collection(In Millions)"]>100)&(df["Collection(In Millions)"]<200)]
```

Out[8]:

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
0	The Matrix	1980	129	Biography, Drama, Sport	8.7	171.48	348069	89
2	The Matrix Revolutions	2003	109	Biography, Drama, Sport	6.7	139.31	40698	38
6	PrometheusI	2016	117	Horror, Thriller	7.0	126.48	94168	73
9	Rise of the Planet of the Apes	1990	127	Drama, Fantasy, Romance	7.6	176.76	213995	65
11	War for the Planet of the Apes	1988	92	Comedy, Fantasy	7.4	146.88	294012	50
...	...	...	...	...	...	...	...	...
378	Flight of the Navigator	2019	161	Comedy, Drama	8.8	142.50	48017	59
380	Dick Tracy	1990	119	Comedy, Romance	7.5	178.41	61653	59
386	Hellraiser	1978	143	Action, Adventure, Sci-Fi	8.0	134.22	118765	65

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
397	*batteries not included	2018	116	Crime, Drama, Thriller	8.1	103.80	35469	51
399	The Internl	2017	104	Horror, Mystery, Thriller	8.0	176.04	248313	0

113 rows × 8 columns

In [9]:
```
df[(df["Collection(In Millions)"]>0)&(df["Collection(In Millions)"]<100)]
```

Out[9]:

	Movie_Name	Release_Year	Duration(In Minutes)	Genre	Ratings	Collection(In Millions)	Votings	MetaScore
3	Alien	2004	89	Drama	8.5	78.90	0	86
4	Aliens	2006	101	Crime, Drama	8.4	85.16	121722	72
5	Alien³	2008	116	Drama	6.4	55.47	765172	51
7	Scarface	2005	116	Comedy, Romance	8.3	45.60	429938	62
12	Schindlers List	1986	101	Adventure, Family, Fantasy	9.0	96.90	135164	57
...	...	...	...	...	...	...	...	...
387	Cocoon	1984	116	Adventure, Mystery, Sci-Fi	8.2	40.20	63035	53
391	Resident Evil	2019	122	Action, Adventure, Sci-Fi	7.5	85.71	267261	53
393	Irrational Man	2011	101	Action, Crime, Thriller	8.5	4.11	62984	45
394	The Woman in Black	2000	94	Biography, Crime, Drama	7.7	0.23	182407	52
398	The Drop	2014	114	Adventure, Comedy, Drama	7.2	30.64	151395	0

191 rows × 8 columns

Cleaning Final Dataframe by converting objects into Intiger

In [34]:
```
df["Collection(In Millions)"] = df["Collection(In Millions)"]
```

In [35]:
```
df["MetaScore"] = df["MetaScore"].astype(int)
```

In [9]:
```
df["Ratings"].value_counts()
```

Out[9]:
```
8.2    44
7.6    32
8.1    32
7.8    28
8.6    24
7.5    24
7.7    24
8.4    20
7.2    16
8.0    16
8.5    16
8.3    16
7.3    12
8.7    12
9.0    12
7.4    12
8.8    12
7.1     8
6.8     8
7.9     4
7.0     4
9.2     4
6.2     4
6.4     4
6.3     4
6.7     4
8.9     4
Name: Ratings, dtype: int64
```

In [8]:
```
df["Collection(In Millions)"].value_counts()
```

Out[8]:
```
188.02    6
210.61    6
248.16    3
42.44     3
92.19     3
         ..
100.21    1
26.39     1
31.04     1
39.57     1
176.04    1
Name: Collection(In Millions), Length: 202, dtype: int64
```

Exploratory Data Analysis

In [7]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

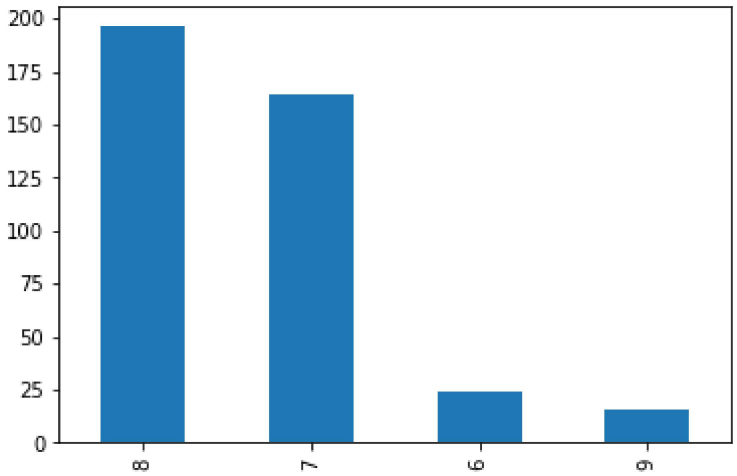
For Rating and Gross

Univariate Analysis

Bar Plot

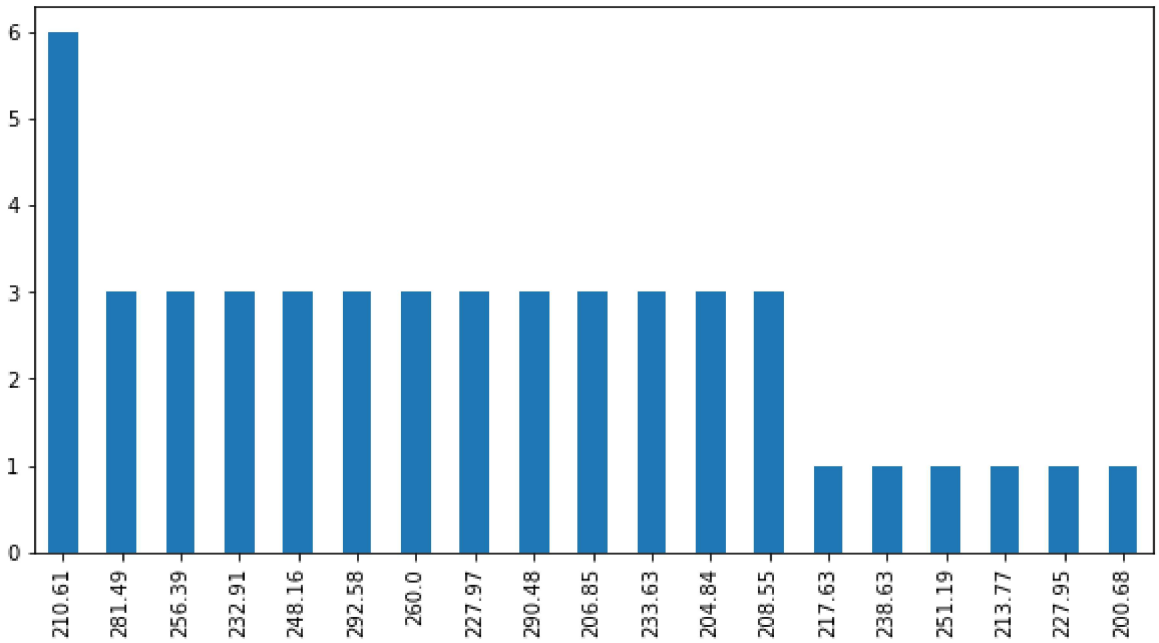
In [41]:
```
df['Ratings'].value_counts().plot.bar()
```

Out[41]: <AxesSubplot:>



In [42]:
```
plt.figure(figsize=(10,5))
plt.xticks(rotation=90)
rdf['Collection(In Millions)'].value_counts().plot.bar()
```

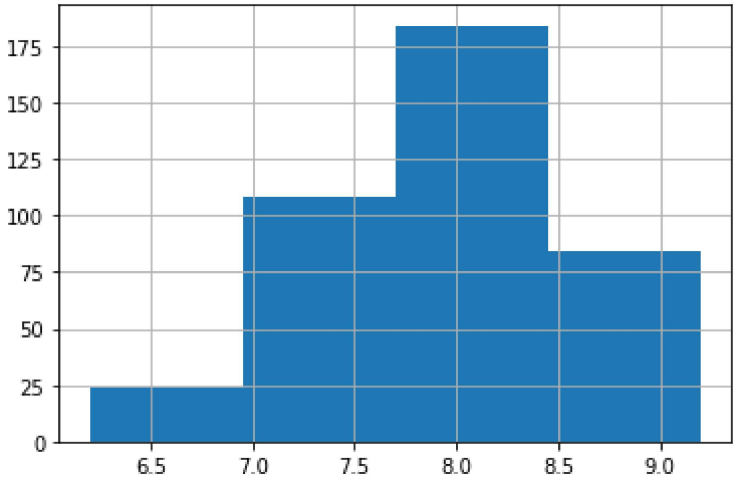
Out[42]: <AxesSubplot:>



Histogram

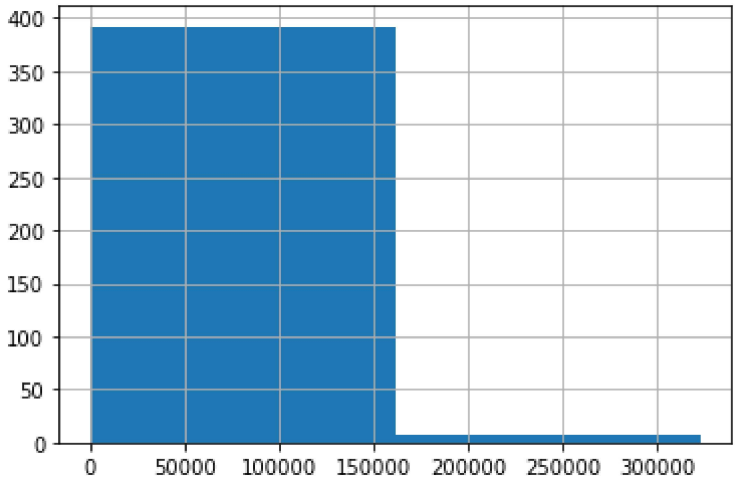
In [27]:
```
df["Ratings"].hist(bins=4)
```

Out[27]:
```
<AxesSubplot:>
```



In [17]:
```
df["Collection(In Millions)"].hist(bins=2)
```

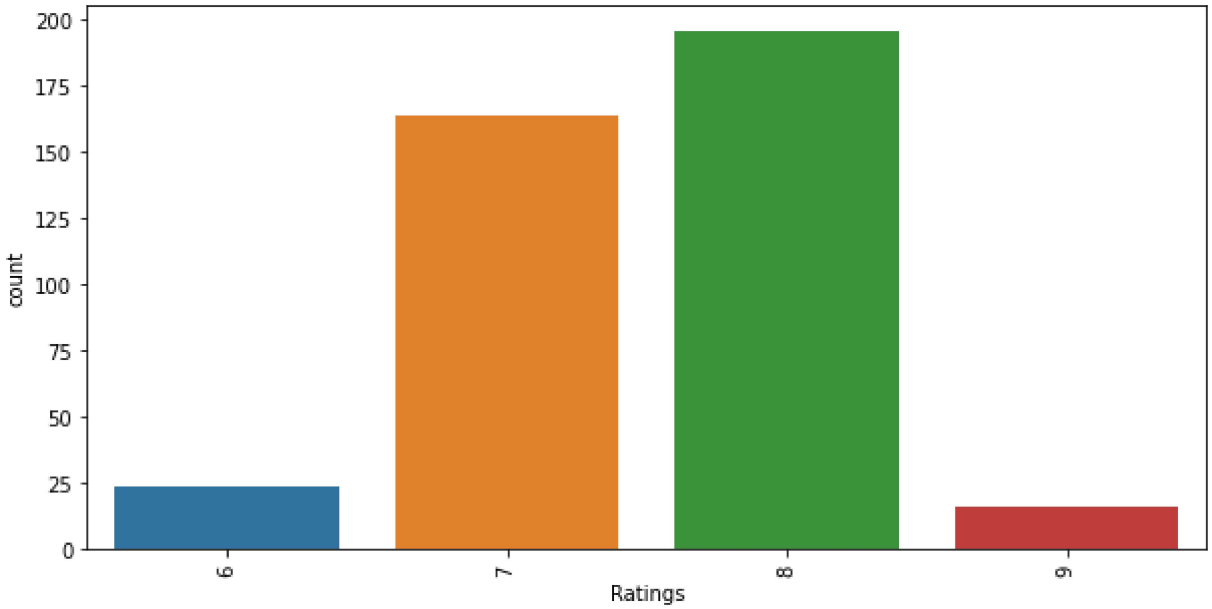
Out[17]:
```
<AxesSubplot:>
```



Count Plot

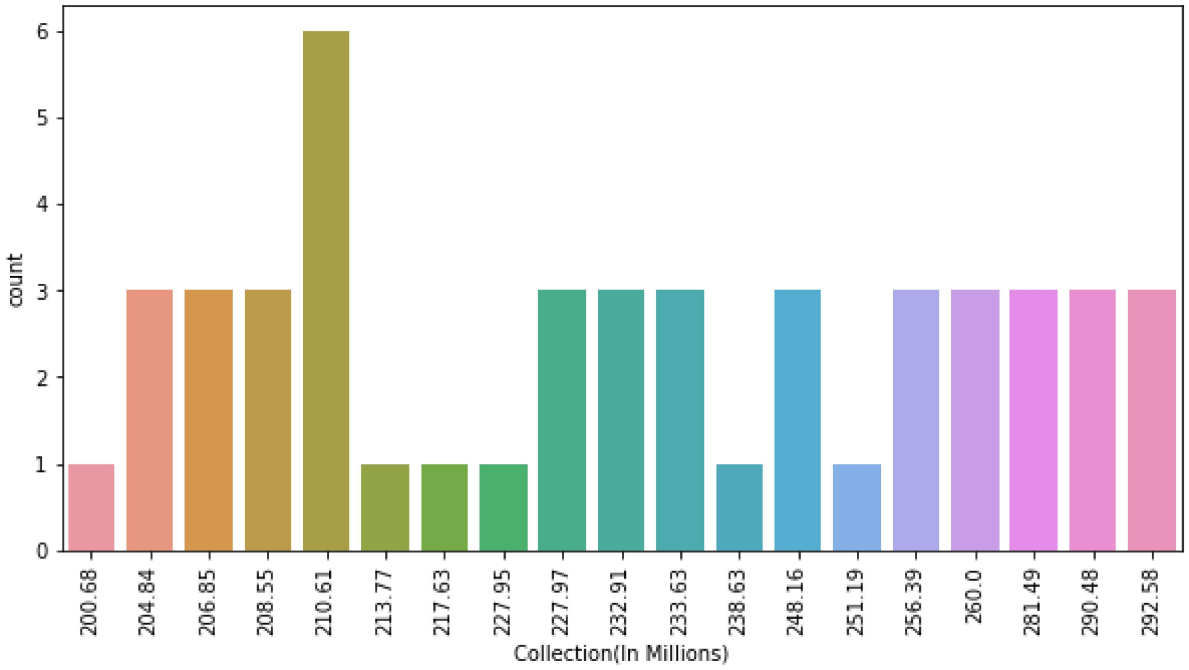
In [43]:
```
plt.figure(figsize=(10,5))
plt.xticks(rotation = 90)
sns.countplot(x='Ratings',data=df)
```

Out[43]:
```
<AxesSubplot:xlabel='Ratings', ylabel='count'>
```



In [44]:
```
plt.figure(figsize=(10,5))
plt.xticks(rotation = 90)
sns.countplot(x='Collection(In Millions)',data=rdf)
```

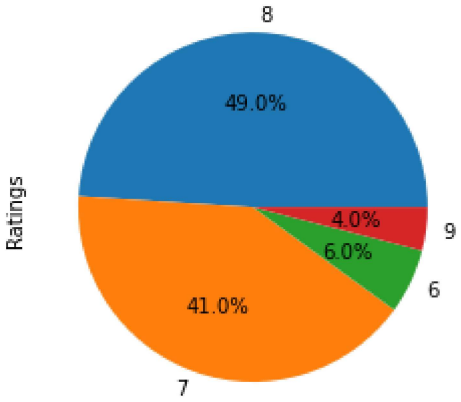
Out[44]: <AxesSubplot:xlabel='Collection(In Millions)', ylabel='count'>



Pie Chart

In [45]:
```
df['Ratings'].value_counts().plot.pie(autopct="%.1f%%")
```

Out[45]: <AxesSubplot:ylabel='Ratings'>

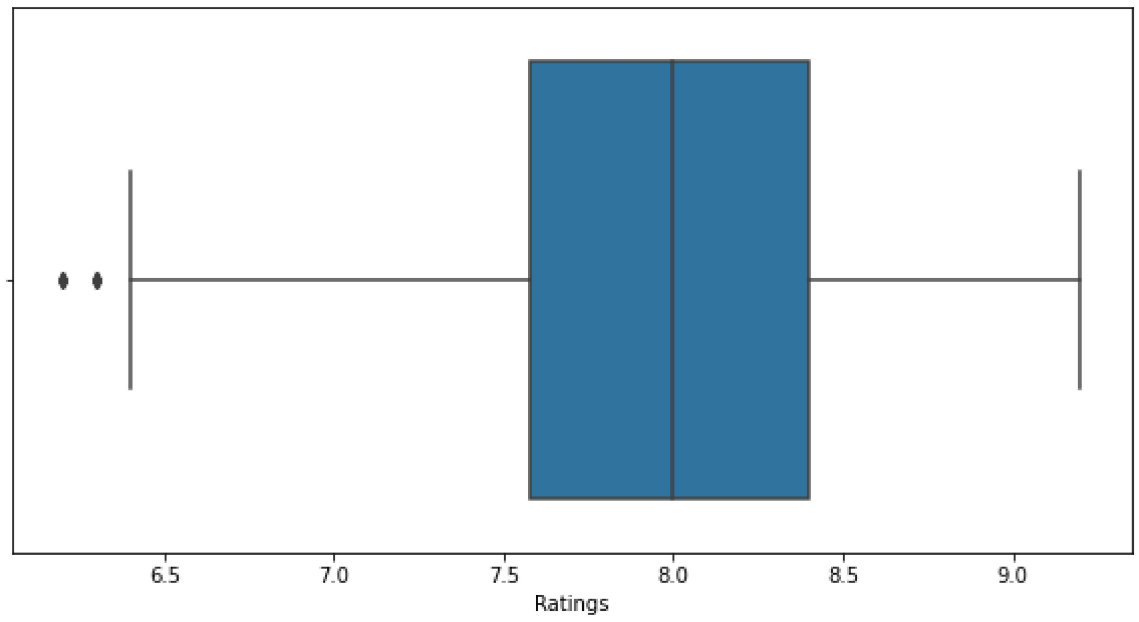


Box Plot

In [20]:
```
plt.figure(figsize=(10,5))
sns.boxplot(df["Ratings"])
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will re
sult in an error or misinterpretation.
  warnings.warn(
```

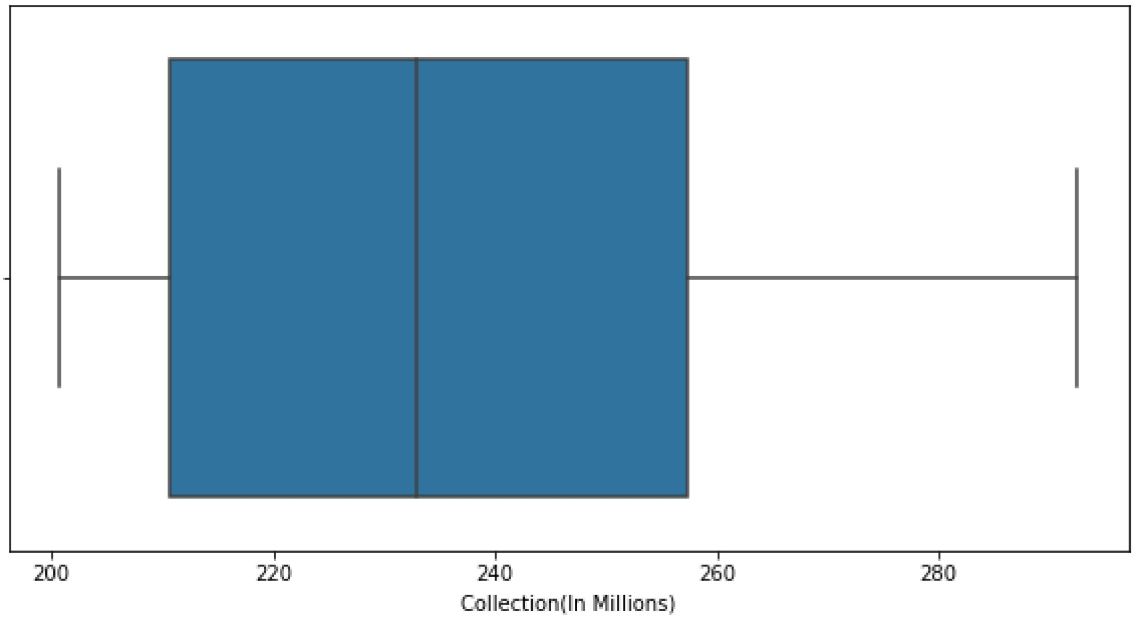
Out[20]:
```
<AxesSubplot:xlabel='Ratings'>
```



In [24]:
```
plt.figure(figsize=(10,5))
sns.boxplot(rdf["Collection(In Millions)"])
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will re
sult in an error or misinterpretation.
  warnings.warn(
```

Out[24]:
```
<AxesSubplot:xlabel='Collection(In Millions)'>
```



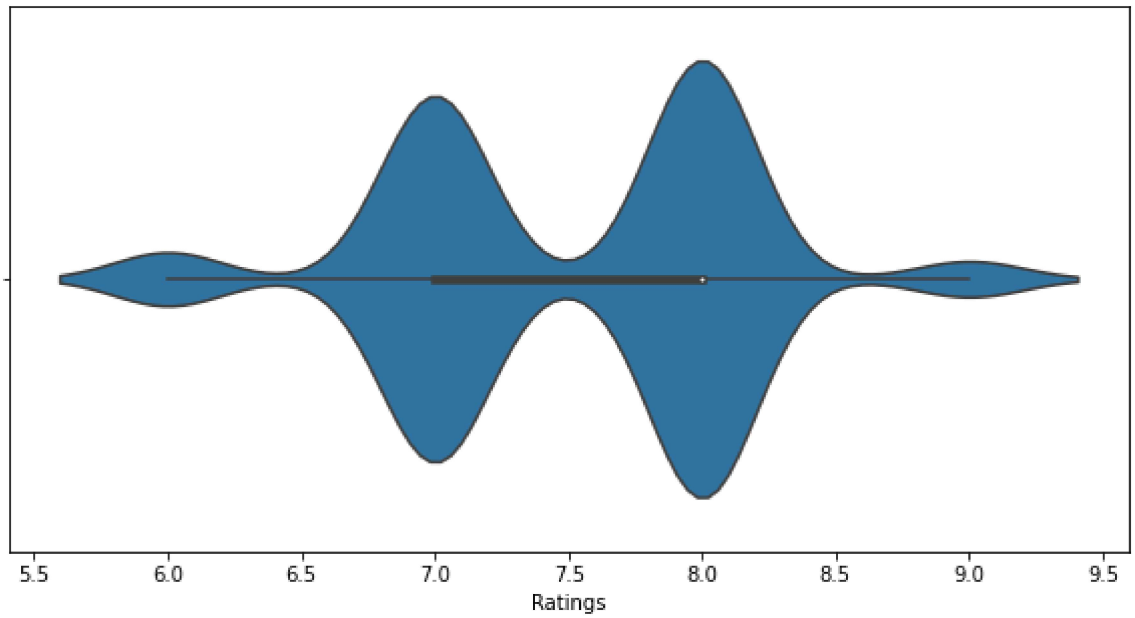
Violin Plot

In [46]:
```
plt.figure(figsize=(10,5))
sns.violinplot(df["Ratings"])
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
```

Out[46]:
```
<AxesSubplot:xlabel='Ratings'>
```

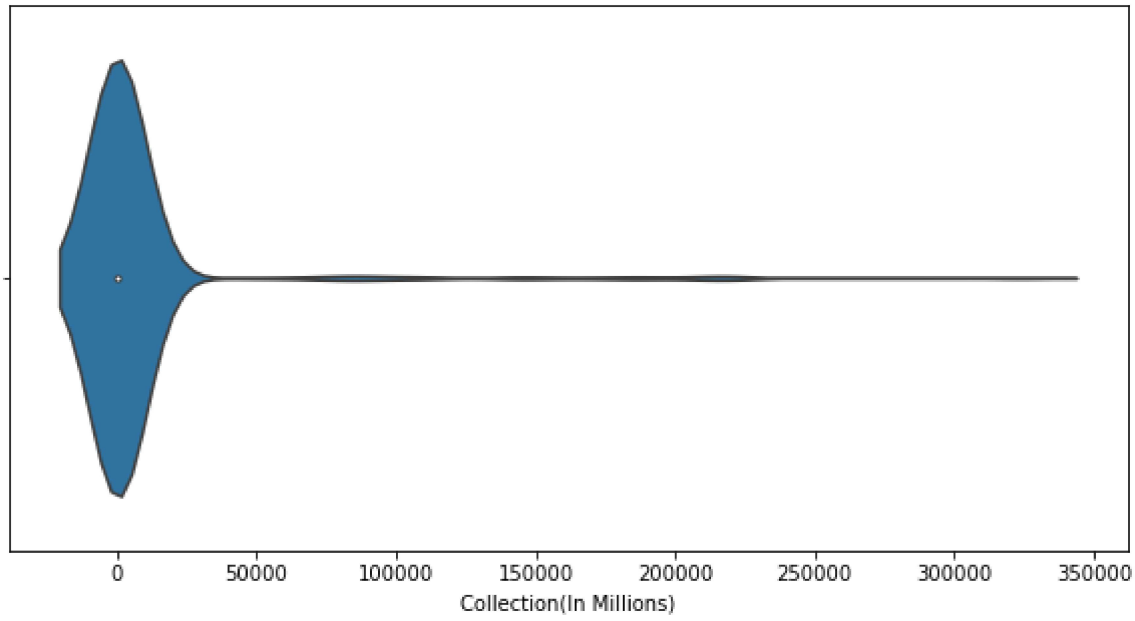


In [47]:
```
plt.figure(figsize=(10,5))
sns.violinplot(df["Collection(In Millions)"])
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
```

Out[47]: <AxesSubplot:xlabel='Collection(In Millions)'>



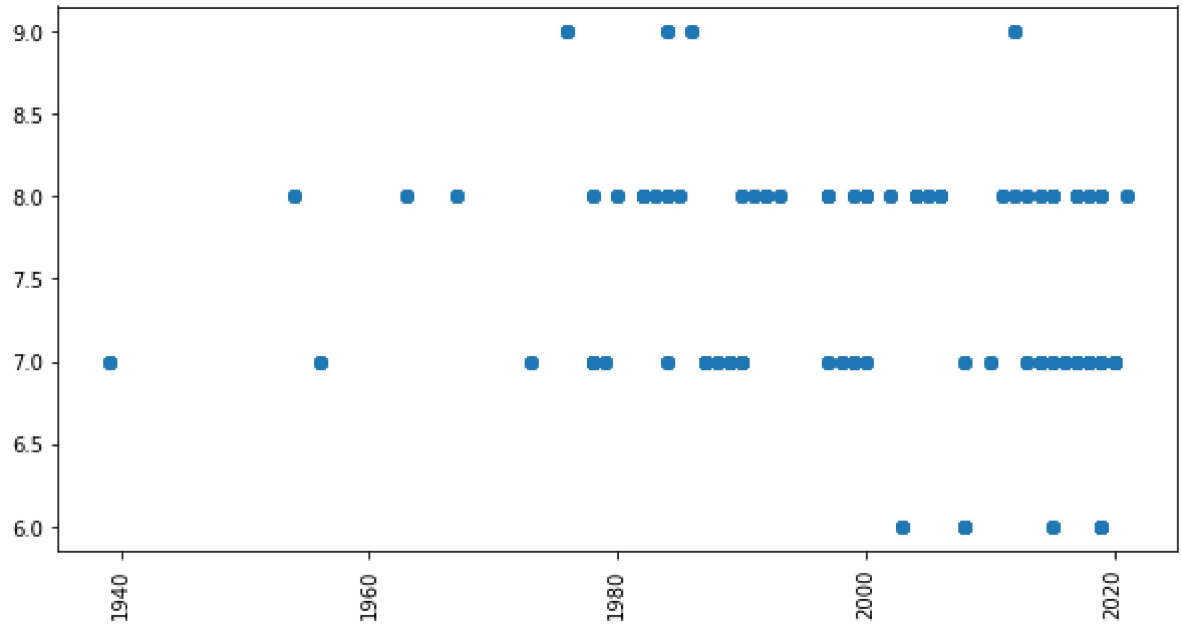
Bi-Variate Analysis

Num vs Num

Scatterplot

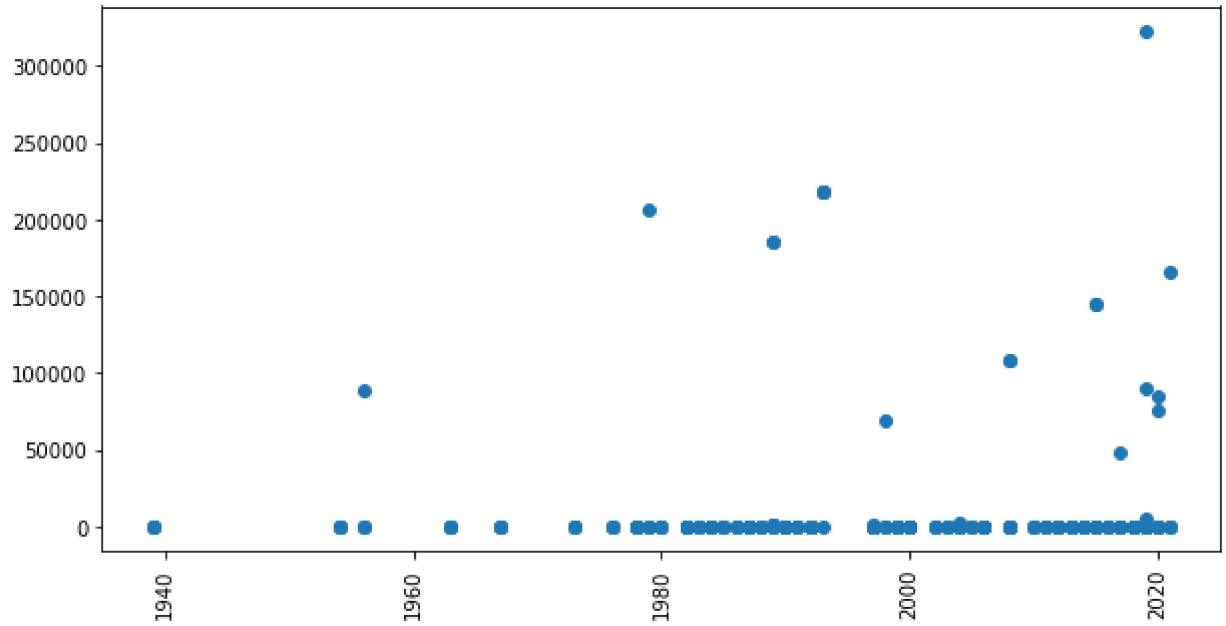
In [51]:
```
plt.figure(figsize=(10,5))
plt.xticks(rotation=90)
plt.scatter(df["Release_Year"],df["Ratings"])
```

Out[51]: <matplotlib.collections.PathCollection at 0x1fe2a780eb0>



In [52]:
```
plt.figure(figsize=(10,5))
plt.xticks(rotation=90)
plt.scatter(df["Release_Year"],df["Collection(In Millions)"])
```

Out[52]:
```
<matplotlib.collections.PathCollection at 0x1fe2a55dd00>
```



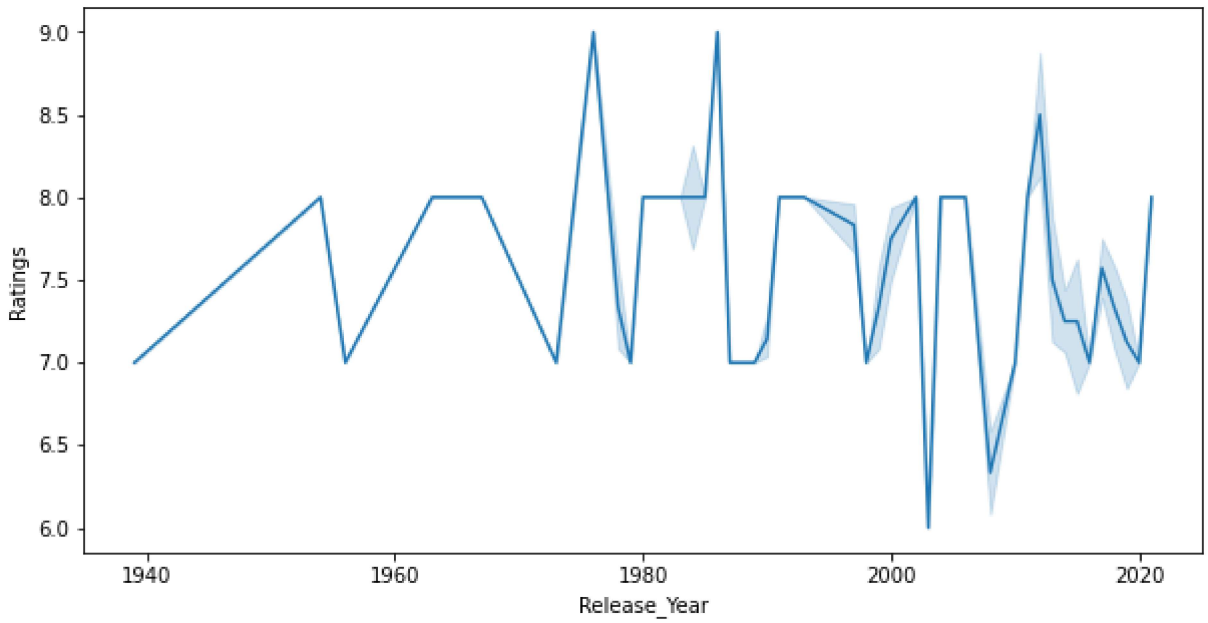
In [33]:
```
df["Release_Year"].corr(df["Ratings"])
```

Out[33]:
```
-0.20216961880118134
```

Line Chart

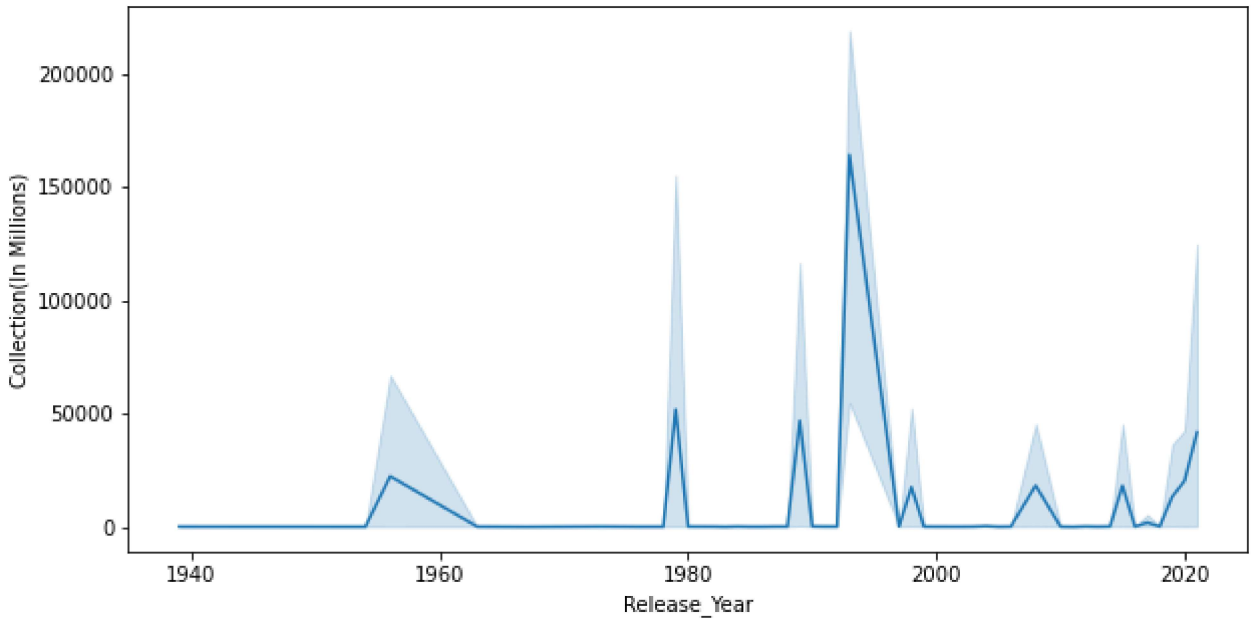
In [53]:
```
plt.figure(figsize=(10,5))
sns.lineplot(x="Release_Year",y="Ratings",data=df)
```

Out[53]: <AxesSubplot:xlabel='Release_Year', ylabel='Ratings'>



In [55]:
```
plt.figure(figsize=(10,5))
sns.lineplot(x="Release_Year",y="Collection(In Millions)",data=df)
```

Out[55]: <AxesSubplot:xlabel='Release_Year', ylabel='Collection(In Millions)'>

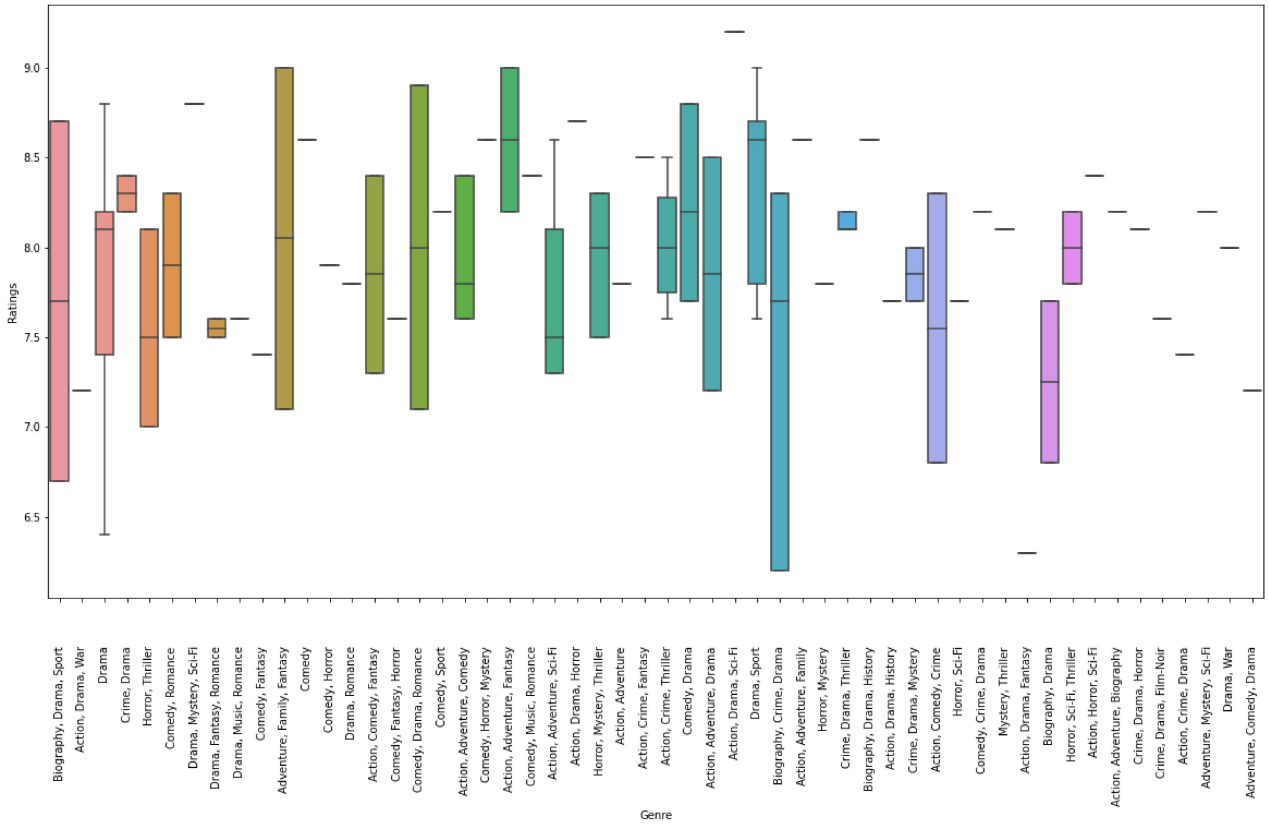


Category Vs Num

Boxplot

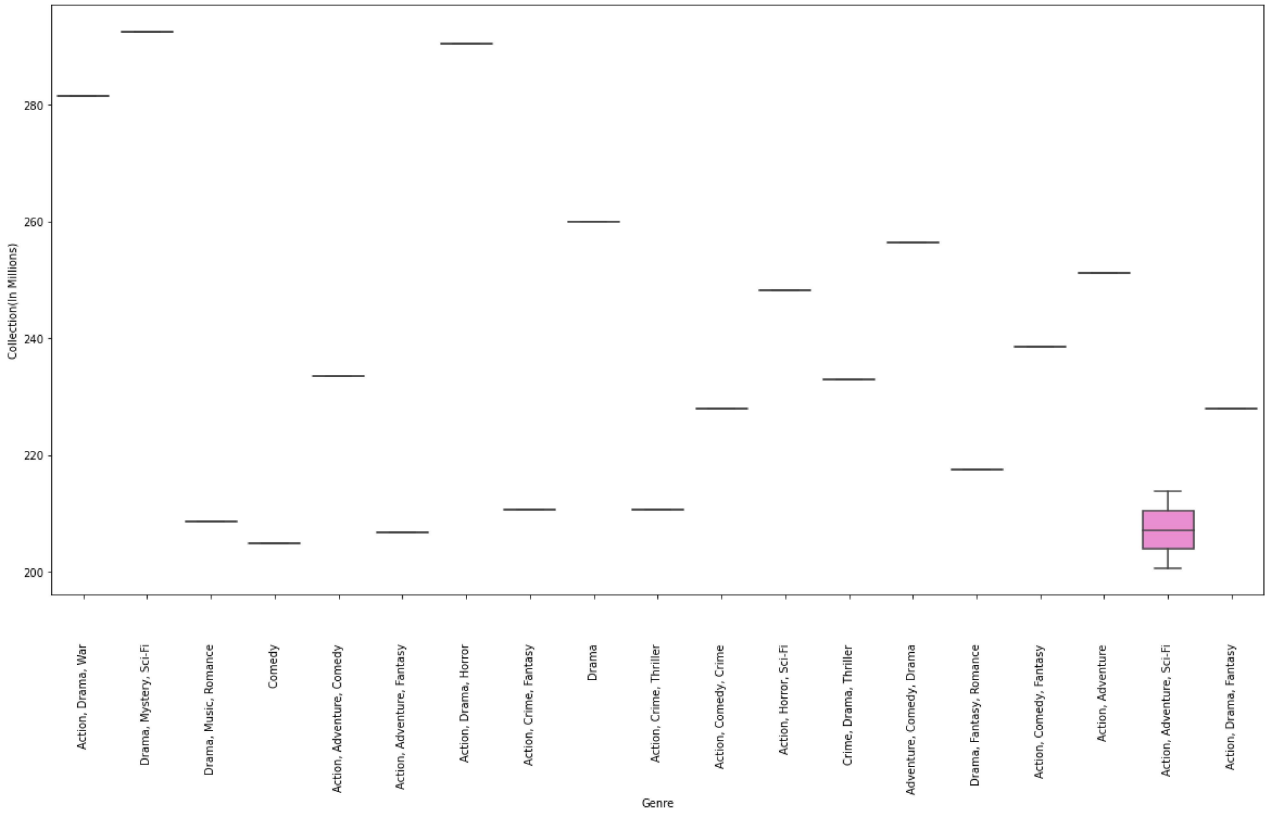
In [35]:
```
plt.figure(figsize=(20,10))
plt.xticks(rotation=90)
sns.boxplot(x="Genre",y="Ratings",data=df)
```

Out[35]: <AxesSubplot:xlabel='Genre', ylabel='Ratings'>



In [10]:
```
plt.figure(figsize=(20,10))
plt.xticks(rotation=90)
sns.boxplot(x="Genre",y="Collection(In Millions)",data=rdf)
```

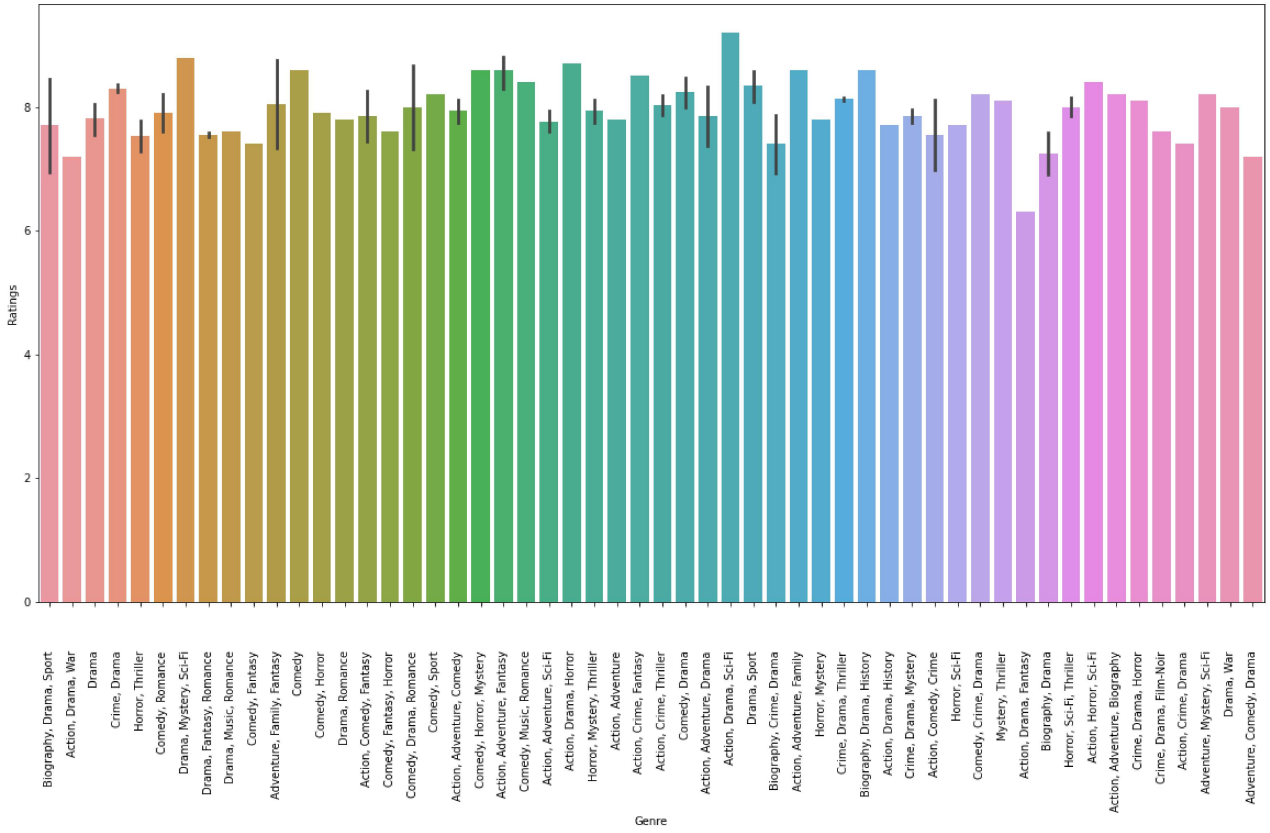
Out[10]: <AxesSubplot:xlabel='Genre', ylabel='Collection(In Millions)'>



BarPlot

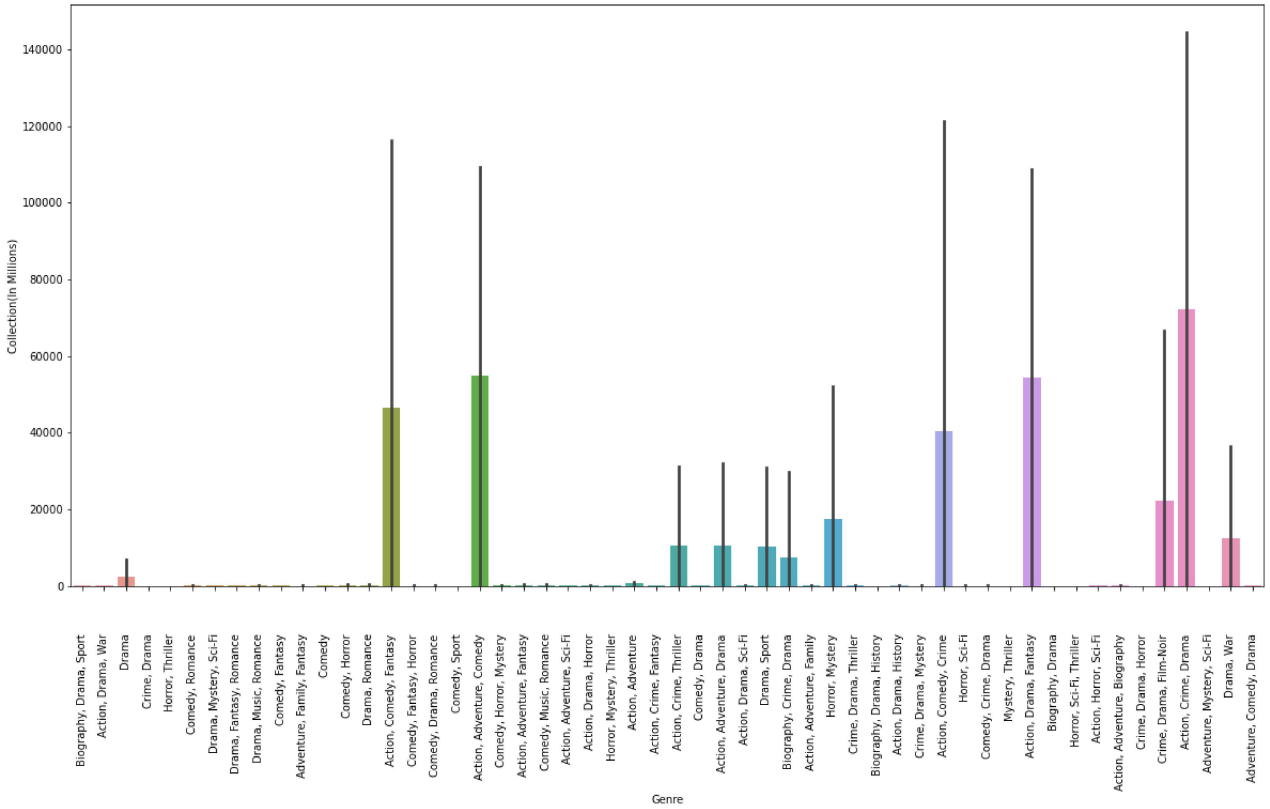
In [36]:
```
plt.figure(figsize=(20,10))
plt.xticks(rotation=90)
sns.barplot(x="Genre",y="Ratings",data=df)
```

Out[36]:
```
<AxesSubplot:xlabel='Genre', ylabel='Ratings'>
```



In [56]:
```
plt.figure(figsize=(20,10))
plt.xticks(rotation=90)
sns.barplot(x="Genre",y="Collection(In Millions)",data=df)
```

Out[56]:
```
<AxesSubplot:xlabel='Genre', ylabel='Collection(In Millions)'>
```



Multi-Variate

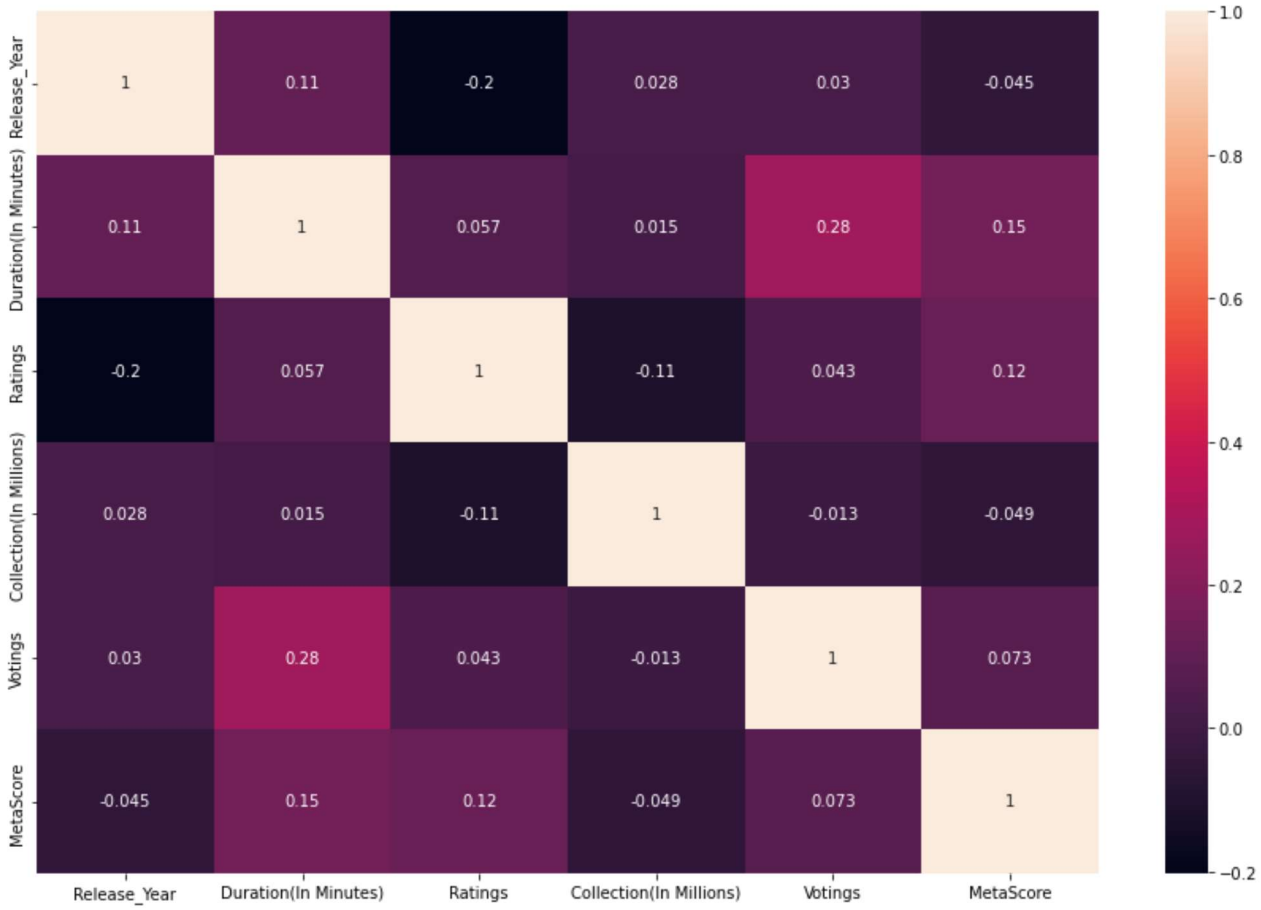
HeatMap

In [27]:
```
num_df=df.select_dtypes(include=['number'])
num_df.corr()
```

Out[27]:

	Release_Year	Duration(In Minutes)	Ratings	Collection(In Millions)	Votings	MetaScore
Release_Year	1.000000	0.111235	-0.202170	0.028140	0.030358	-0.045277
Duration(In Minutes)	0.111235	1.000000	0.056632	0.015389	0.279814	0.150863
Ratings	-0.202170	0.056632	1.000000	-0.112230	0.042995	0.117973
Collection(In Millions)	0.028140	0.015389	-0.112230	1.000000	-0.013475	-0.048949
Votings	0.030358	0.279814	0.042995	-0.013475	1.000000	0.073450
MetaScore	-0.045277	0.150863	0.117973	-0.048949	0.073450	1.000000

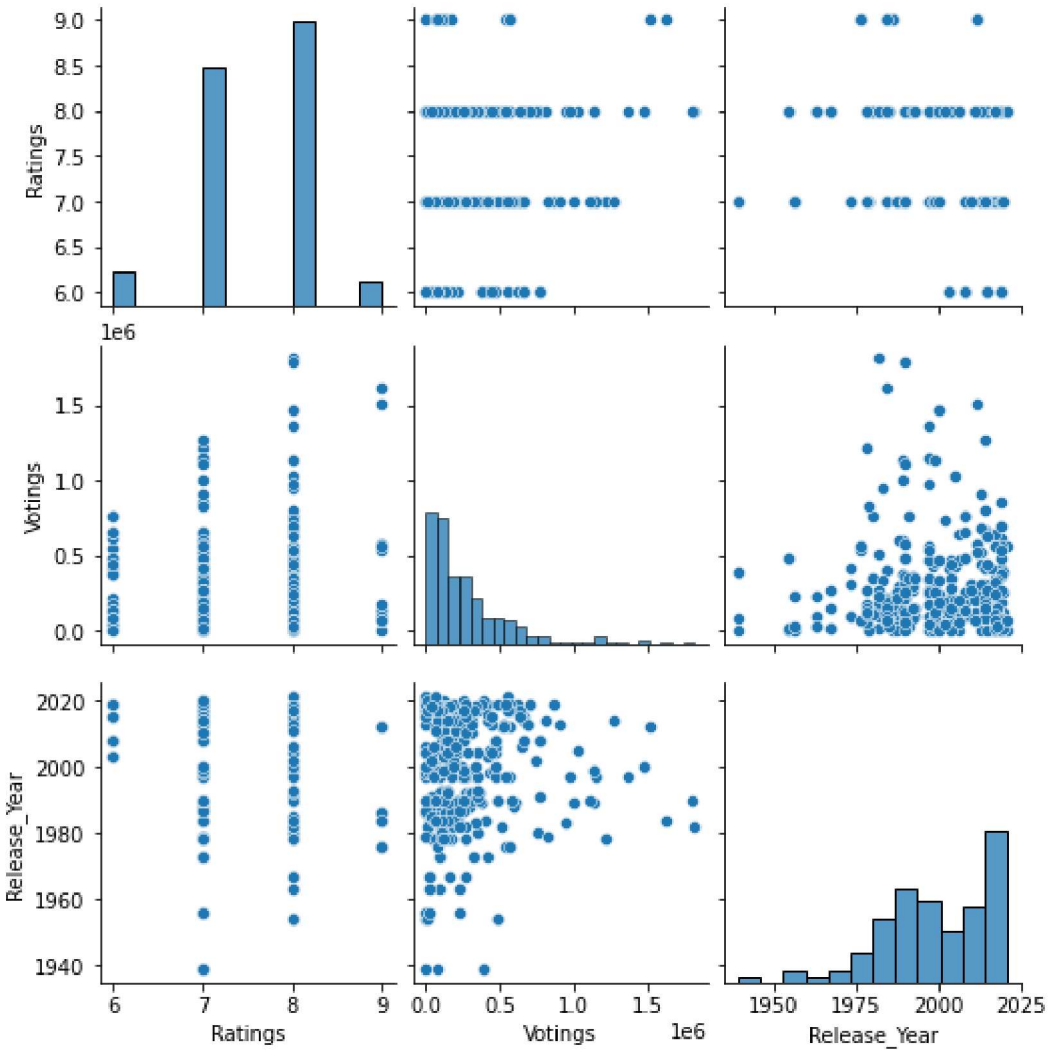
In [28]:
```
plt.figure(figsize=(15,10))
sns.heatmap(num_df.corr(),annot=True)
plt.show()
```



PairPlot

In [57]:

```
sns.pairplot(data = df, vars=['Ratings','Votings','Release_Year'])
plt.show()
```


Ratings
Votings
Release_Year
1e6
1e6
9.0
8.5
8.0
7.5
7.0
6.5
6.0
1.5
1.0
0.5
0.0
2020
2000
1980
1960
1940
6
7
8
9
0.0
0.5
1.0
1.5
1950
1975
2000
2025

In [ ]: