In [2]:
```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
df=pd.read_csv('movie_info.csv')

print(df) #by this we have read the csv file
```

```
                                   movie_title       release_date  season  \
0                                     Iron Man        May 2, 2008  Spring
1                          The Incredible Hulk      June 13, 2008  Spring
2                                   Iron Man 2        May 7, 2010  Spring
3                                         Thor        May 6, 2011  Spring
4           Captain America: The First Avenger      July 22, 2011  Summer
5                                 The Avengers        May 4, 2012  Spring
6                                   Iron Man 3        May 3, 2013  Spring
7                         Thor: The Dark World   November 8, 2013    Fall
8          Captain America: The Winter Soldier      April 4, 2014  Spring
9                      Guardians of the Galaxy     August 1, 2014  Summer
10                     Avengers: Age of Ultron        May 1, 2015  Spring
11                                    Ant-Man      July 17, 2015  Summer
12                  Captain America: Civil War        May 6, 2016  Spring
13                             Doctor Strange   November 4, 2016    Fall
14              Guardians of the Galaxy Vol. 2        May 5, 2017  Spring
15                      Spider-Man: Homecoming       July 7, 2017  Summer
16                             Thor: Ragnarok   November 3, 2017    Fall
17                               Black Panther  February 16, 2018  Winter
18                      Avengers: Infinity War     April 27, 2018  Spring
19                        Ant-Man and the Wasp       July 6, 2018  Summer
20                              Captain Marvel      March 8, 2019  Winter
21                           Avengers: Endgame     April 26, 2019  Spring
22                     Spider-Man: Far From Home       July 2, 2019  Summer
23                                 Black Widow       July 9, 2021  Summer
24  Shang-Chi and the Legend of the Ten Rings  September 3, 2021  Summer
25                                    Eternals   November 5, 2021    Fall
26                      Spider-Man: No Way Home  December 17, 2021    Fall

    phase  production_budget_in_million_(USD)  \
0       1                                 140
1       1                                 150
2       1                                 200
3       1                                 150
4       1                                 140
5       1                                 220
6       2                                 200
7       2                                 170
8       2                                 170
9       2                                 170
10      2                                 250
11      2                                 130
12      3                                 250
13      3                                 165
14      3                                 200
15      3                                 175
16      3                                 180
17      3                                 200
18      3                                 325
19      3                                 162
20      3                                 160
21      3                                 356
22      3                                 160
23      4                                 200
24      4                                 150
25      4                                 200
26      4                                 200

    worldwide_collection_in_million_(USD)  tomatometer  tomato_audience_score  \
0                                  585.37          9.4                    9.1
1                                  264.77          6.7                    7.0
2                                  623.93          7.2                    7.1
3                                  449.33          7.7                    7.6
4                                  370.57          7.9                    7.5
5                                 1518.82          9.1                    9.1
6                                 1214.81          7.9                    7.8
7                                  644.78          6.6                    7.5
8                                  714.42          9.0                    9.2
9                                  773.35          9.2                    9.2
10                                1402.81          7.6                    8.3
11                                 519.31          8.3                    8.5
12                                1153.34          9.0                    8.9
13                                 677.80          8.9                    8.5
14                                 863.76          8.5                    8.7
15                                 880.17          9.2                    8.7
16                                 853.98          9.3                    8.7
17                                1347.60          9.6                    7.9
18                                2048.36          8.5                    9.1
19                                 622.67          8.7                    8.1
20                                1128.46          7.9                    4.5
21                                2797.50          9.4                    9.0
22                                1131.93          9.0                    9.5
23                                 379.75          7.9                    9.1
24                                 432.24          9.1                    9.8
25                                 402.06          4.7                    7.8
26                                1892.62          9.3                    9.8

    imdb  metascore  meta_user_score
0    7.9        7.9              8.6
1    6.6        6.1              7.0
2    6.9        5.7              6.4
3    7.0        5.7              7.1
4    6.9        6.6              6.8
5    8.0        6.9              8.0
6    7.1        6.2              6.7
7    6.8        5.4              7.1
8    7.8        7.0              8.3
9    8.0        7.6              8.2
10   7.3        6.6              7.1
11   7.3        6.4              7.5
12   7.8        7.5              8.0
13   7.5        7.2              8.0
14   7.6        6.7              7.8
15   7.4        7.3              7.6
16   7.9        7.4              7.8
17   7.3        8.8              6.3
18   8.4        6.8              8.6
19   7.0        7.0              7.1
20   6.8        6.4              3.0
21   8.4        7.8              7.9
22   7.4        6.9              7.6
23   6.7        6.7              6.1
24   7.4        7.1              7.1
25   6.4        5.2              6.3
26   8.4        7.1              8.6
```

In [3]:
```python
df.head() #this means first five rows will be display
```

Out[3]:

	movie_title	release_date	season	phase	production_budget_in_million_(USD)	worldwide_collection_in_million_(USD)	tomatometer	tomato_audience_score	imdb	metascore
0	Iron Man	May 2, 2008	Spring	1	140	585.37	9.4	9.1	7.9	7.9
1	The Incredible Hulk	June 13, 2008	Spring	1	150	264.77	6.7	7.0	6.6	6.1
2	Iron Man 2	May 7, 2010	Spring	1	200	623.93	7.2	7.1	6.9	5.7
3	Thor	May 6, 2011	Spring	1	150	449.33	7.7	7.6	7.0	5.7
4	Captain America: The First Avenger	July 22, 2011	Summer	1	140	370.57	7.9	7.5	6.9	6.6

In [4]:
```python
df.tail()  #this means last five rows will be display
```

Out[4]:

	movie_title	release_date	season	phase	production_budget_in_million_(USD)	worldwide_collection_in_million_(USD)	tomatometer	tomato_audience_score	imdb	metascore
22	Spider-Man: Far From Home	July 2, 2019	Summer	3	160	1131.93	9.0	9.5	7.4	6.9
23	Black Widow	July 9, 2021	Summer	4	200	379.75	7.9	9.1	6.7	6.7
24	Shang-Chi and the Legend of the Ten Rings	September 3, 2021	Summer	4	150	432.24	9.1	9.8	7.4	7.1
25	Eternals	November 5, 2021	Fall	4	200	402.06	4.7	7.8	6.4	5.2
26	Spider-Man: No Way Home	December 17, 2021	Fall	4	200	1892.62	9.3	9.8	8.4	7.1

In [5]:
```python
df.info() #this  prints information about the DataFrame.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27 entries, 0 to 26
Data columns (total 11 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   movie_title                            27 non-null     object
 1   release_date                           27 non-null     object
 2   season                                 27 non-null     object
 3   phase                                  27 non-null     int64
 4   production_budget_in_million_(USD)     27 non-null     int64
 5   worldwide_collection_in_million_(USD)  27 non-null     float64
 6   tomatometer                            27 non-null     float64
 7   tomato_audience_score                  27 non-null     float64
 8   imdb                                   27 non-null     float64
 9   metascore                              27 non-null     float64
 10  meta_user_score                        27 non-null     float64
dtypes: float64(6), int64(2), object(3)
memory usage: 2.4+ KB
```

In [6]:
```python
df.columns    # returns the list of columns
```

Out[6]:
```
Index(['movie_title', 'release_date', 'season', 'phase',
       'production_budget_in_million_(USD)',
       'worldwide_collection_in_million_(USD)', 'tomatometer',
       'tomato_audience_score', 'imdb', 'metascore', 'meta_user_score'],
      dtype='object')
```

In [7]:
```python
df.dtypes  #returns datatypes of columns
```

Out[7]:
```
movie_title                               object
release_date                              object
season                                    object
phase                                      int64
production_budget_in_million_(USD)         int64
worldwide_collection_in_million_(USD)    float64
tomatometer                              float64
tomato_audience_score                    float64
imdb                                     float64
metascore                                float64
meta_user_score                          float64
dtype: object
```

In [8]:
```python
df.describe()    #f.describe() analyzes numeric as well as to object series or series of a DataFrame.
```

Out[8]:

	phase	production_budget_in_million_(USD)	worldwide_collection_in_million_(USD)	tomatometer	tomato_audience_score	imdb	metascore	meta_user_score
count	27.000000	27.000000	27.000000	27.000000	27.000000	27.000000	27.000000	27.000000
mean	2.481481	191.592593	951.648519	8.355556	8.370370	7.407407	6.814815	7.281481
std	1.014145	52.859217	592.519375	1.111248	1.099586	0.567671	0.798895	1.127638
min	1.000000	130.000000	264.770000	4.700000	4.500000	6.400000	5.200000	3.000000
25%	2.000000	160.000000	552.340000	7.900000	7.800000	6.950000	6.400000	6.900000
50%	3.000000	175.000000	773.350000	8.700000	8.700000	7.400000	6.900000	7.500000
75%	3.000000	200.000000	1184.075000	9.150000	9.100000	7.850000	7.250000	8.000000
max	4.000000	356.000000	2797.500000	9.600000	9.800000	8.400000	8.800000	8.600000

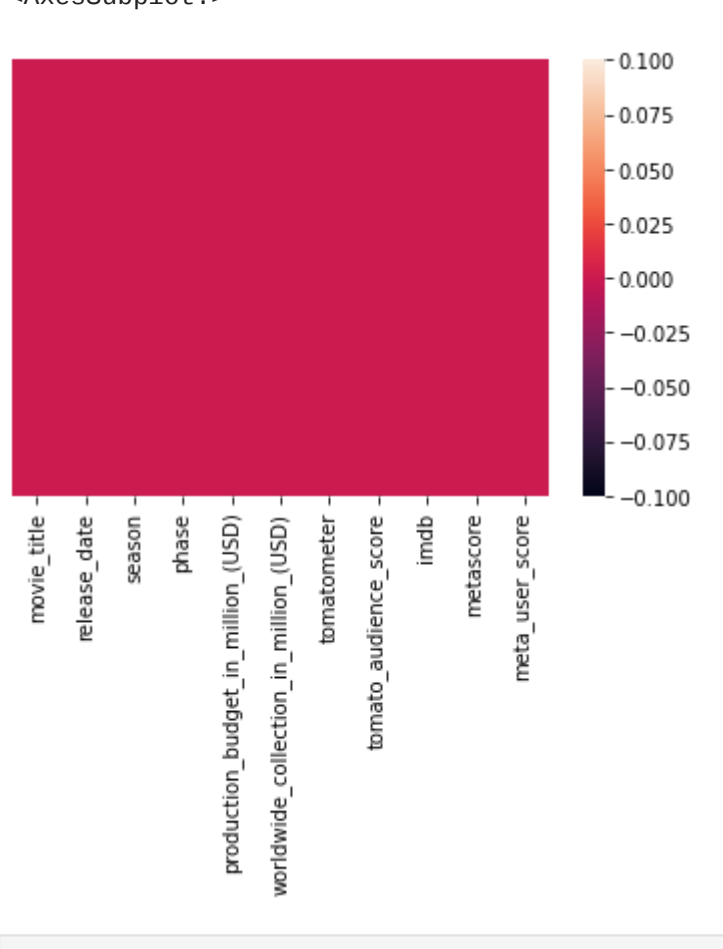
In [9]:
```python
df.isnull() # IF some colmn consist true then there is null value present there , in this case all columns are false so no null values
```

Out[9]:

	movie_title	release_date	season	phase	production_budget_in_million_(USD)	worldwide_collection_in_million_(USD)	tomatometer	tomato_audience_score	imdb	metascore
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False	False	False	False
11	False	False	False	False	False	False	False	False	False	False
12	False	False	False	False	False	False	False	False	False	False
13	False	False	False	False	False	False	False	False	False	False
14	False	False	False	False	False	False	False	False	False	False
15	False	False	False	False	False	False	False	False	False	False
16	False	False	False	False	False	False	False	False	False	False
17	False	False	False	False	False	False	False	False	False	False
18	False	False	False	False	False	False	False	False	False	False
19	False	False	False	False	False	False	False	False	False	False
20	False	False	False	False	False	False	False	False	False	False
21	False	False	False	False	False	False	False	False	False	False
22	False	False	False	False	False	False	False	False	False	False
23	False	False	False	False	False	False	False	False	False	False
24	False	False	False	False	False	False	False	False	False	False
25	False	False	False	False	False	False	False	False	False	False
26	False	False	False	False	False	False	False	False	False	False

In [10]:
```python
sns.heatmap(df.isnull(),yticklabels=False)   #this is a heat map to see if our data
#set is null or not , here the map is clear so there is no null values
```

Out[10]: <AxesSubplot:>



In [11]:
```python
df.max()  #The max () method returns a Series with the maximum value of each column.
```

Out[11]:
```
movie_title                              Thor: The Dark World
release_date                                September 3, 2021
season                                                 Winter
phase                                                       4
production_budget_in_million_(USD)                        356
worldwide_collection_in_million_(USD)                  2797.5
tomatometer                                               9.6
tomato_audience_score                                     9.8
imdb                                                      8.4
metascore                                                 8.8
meta_user_score                                           8.6
dtype: object
```

In [12]:
```python
df.min()   #The max () method returns a Series with the minimum value of each column.
```

Out[12]:
```
movie_title                                    Ant-Man
release_date                            April 26, 2019
season                                            Fall
phase                                                1
production_budget_in_million_(USD)                 130
worldwide_collection_in_million_(USD)           264.77
tomatometer                                        4.7
tomato_audience_score                              4.5
imdb                                               6.4
metascore                                          5.2
meta_user_score                                    3.0
dtype: object
```

In [13]:
```python
df[df['tomato_audience_score'] ==max(df['tomato_audience_score'])].head() # this returns maximimum tomato_audience_score
```

Out[13]:

	movie_title	release_date	season	phase	production_budget_in_million_(USD)	worldwide_collection_in_million_(USD)	tomatometer	tomato_audience_score	imdb	metascore
24	Shang-Chi and the Legend of the Ten Rings	September 3, 2021	Summer	4	150	432.24	9.1	9.8	7.4	7.1
26	Spider-Man: No Way Home	December 17, 2021	Fall	4	200	1892.62	9.3	9.8	8.4	7.1

In [14]:
```python
df.mean()
```

```
C:\Users\Zidane Khan\AppData\Local\Temp\ipykernel_14724\3698961737.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df.mean()
```

Out[14]:
```
phase                                      2.481481
production_budget_in_million_(USD)       191.592593
worldwide_collection_in_million_(USD)    951.648519
tomatometer                                8.355556
tomato_audience_score                      8.370370
imdb                                       7.407407
metascore                                  6.814815
meta_user_score                            7.281481
dtype: float64
```

In [15]:
```python
df.skew()
```

```
C:\Users\Zidane Khan\AppData\Local\Temp\ipykernel_14724\1665899112.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df.skew()
```

Out[15]:
```
phase                                   -0.184755
production_budget_in_million_(USD)       1.793097
worldwide_collection_in_million_(USD)    1.478494
tomatometer                             -1.589216
tomato_audience_score                   -1.699066
imdb                                     0.212946
metascore                                0.058526
meta_user_score                         -2.104872
dtype: float64
```

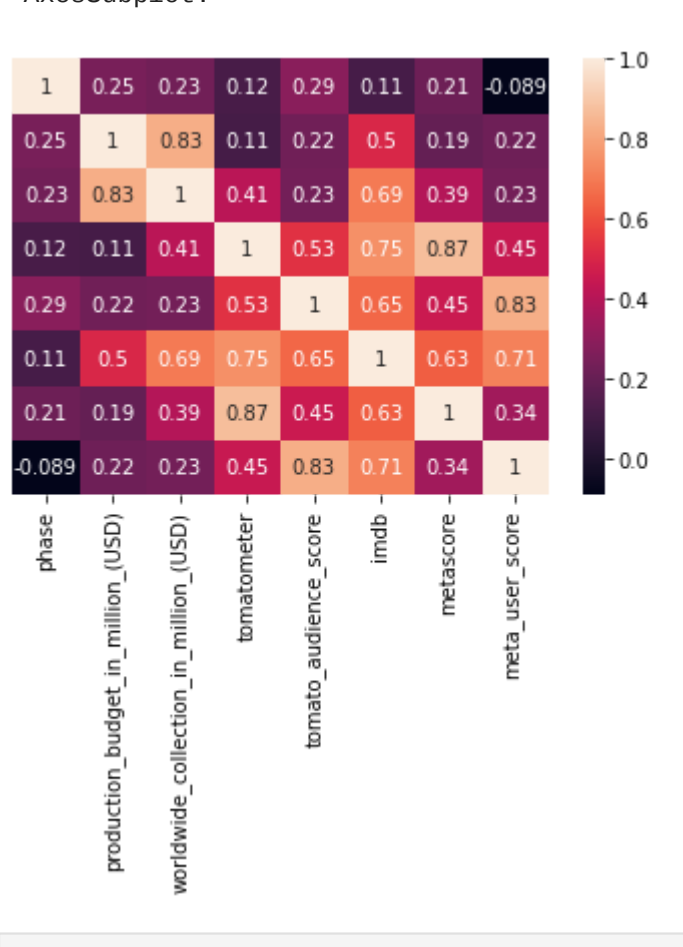
In [16]:
```python
df.corr()
```

Out[16]:

	phase	production_budget_in_million_(USD)	worldwide_collection_in_million_(USD)	tomatometer	tomato_audience_score	imdb	metascore
phase	1.000000	0.245589	0.226243	0.115278	0.289208	0.113821	0.213975
production_budget_in_million_(USD)	0.245589	1.000000	0.828636	0.108373	0.224175	0.500378	0.194237
worldwide_collection_in_million_(USD)	0.226243	0.828636	1.000000	0.411035	0.233255	0.694050	0.394607
tomatometer	0.115278	0.108373	0.411035	1.000000	0.528945	0.751697	0.869411
tomato_audience_score	0.289208	0.224175	0.233255	0.528945	1.000000	0.651657	0.453238
imdb	0.113821	0.500378	0.694050	0.751697	0.651657	1.000000	0.629877
metascore	0.213975	0.194237	0.394607	0.869411	0.453238	0.629877	1.000000
meta_user_score	-0.089437	0.218419	0.232394	0.445294	0.828989	0.711018	0.344857

In [17]:
```python
sns.heatmap(df.corr(),yticklabels=False,  annot=True)
```

Out[17]: <AxesSubplot:>



In [18]:
```python
df.median()
```

```
C:\Users\Zidane Khan\AppData\Local\Temp\ipykernel_14724\530051474.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df.median()
```

Out[18]:
```
phase                                      3.00
production_budget_in_million_(USD)       175.00
worldwide_collection_in_million_(USD)    773.35
tomatometer                                8.70
tomato_audience_score                      8.70
imdb                                       7.40
metascore                                  6.90
meta_user_score                            7.50
dtype: float64
```

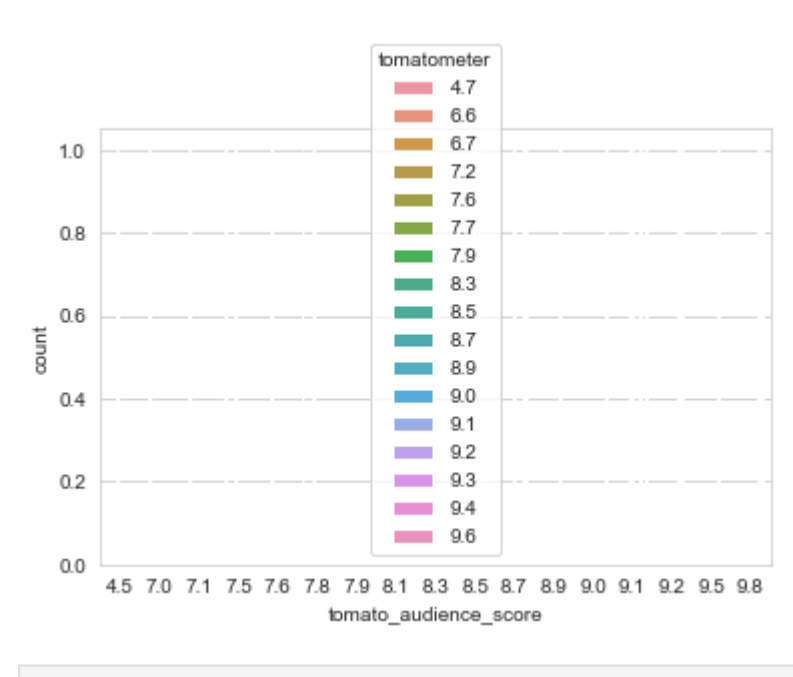
In [19]:
```python
df.std()
```

```
C:\Users\Zidane Khan\AppData\Local\Temp\ipykernel_14724\3390915376.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df.std()
```

Out[19]:
```
phase                                      1.014145
production_budget_in_million_(USD)        52.859217
worldwide_collection_in_million_(USD)    592.519375
tomatometer                                1.111248
tomato_audience_score                      1.099586
imdb                                       0.567671
metascore                                  0.798895
meta_user_score                            1.127638
dtype: float64
```

In [31]:
```python
sns.set_style('whitegrid')
sns.countplot(x='tomato_audience_score',hue='tomatometer',data=df) #this is graph of tomatometer scores and
#tomatometer audience scores ranging between 4 to 10
```

Out[31]: <AxesSubplot:xlabel='tomato_audience_score', ylabel='count'>

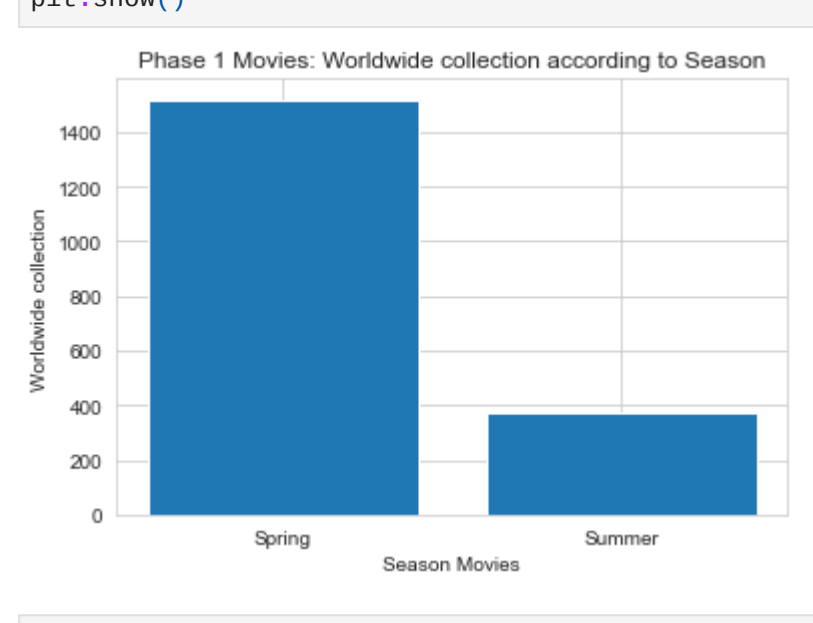


In [21]:
```python
df.var()
```

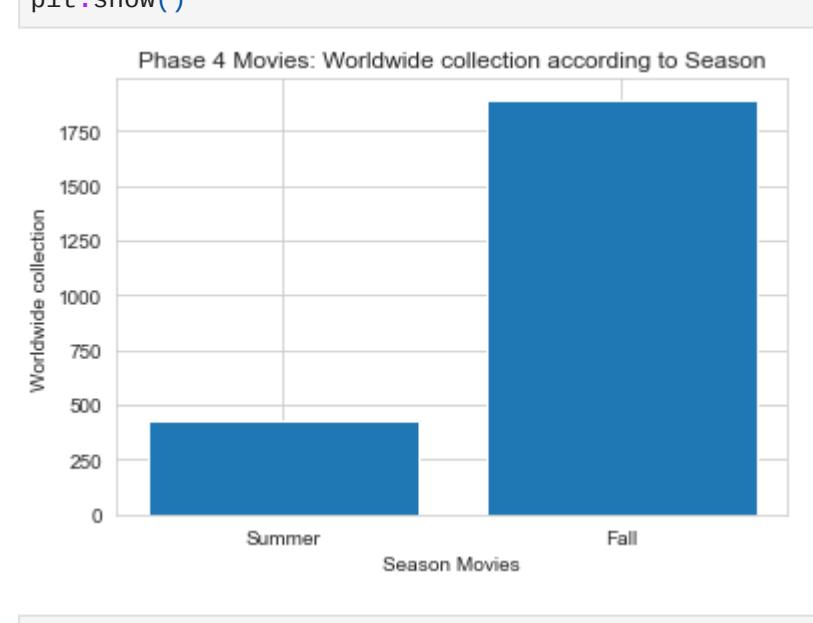
```
C:\Users\Zidane Khan\AppData\Local\Temp\ipykernel_14724\1568254755.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df.var()
```

Out[21]:
```
phase                                         1.028490
production_budget_in_million_(USD)         2794.096866
worldwide_collection_in_million_(USD)    351079.209867
tomatometer                                   1.234872
tomato_audience_score                         1.209088
imdb                                          0.322251
metascore                                     0.638234
meta_user_score                               1.271567
dtype: float64
```

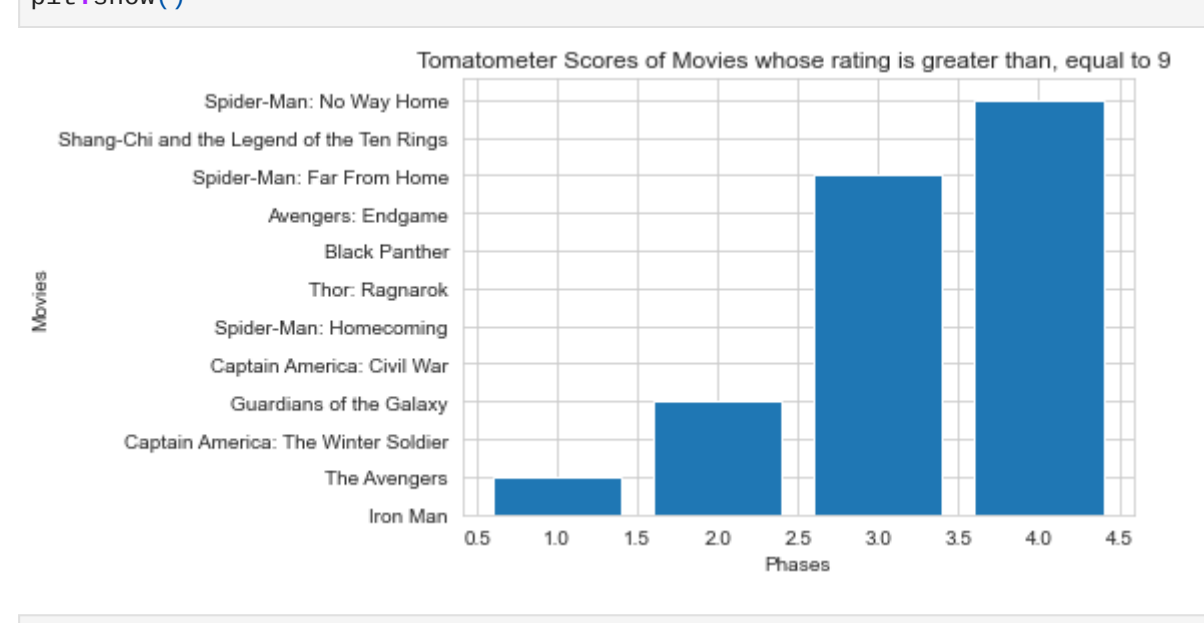
In [32]:
```python
x = df.loc[(df.phase == 1), 'worldwide_collection_in_million_(USD)']
y = df.loc[(df.phase == 1), 'season']
plt.bar(y, x)

plt.xlabel('Season Movies')
plt.ylabel('Worldwide collection')
plt.title('Phase 1 Movies: Worldwide collection according to Season')
plt.show()
```



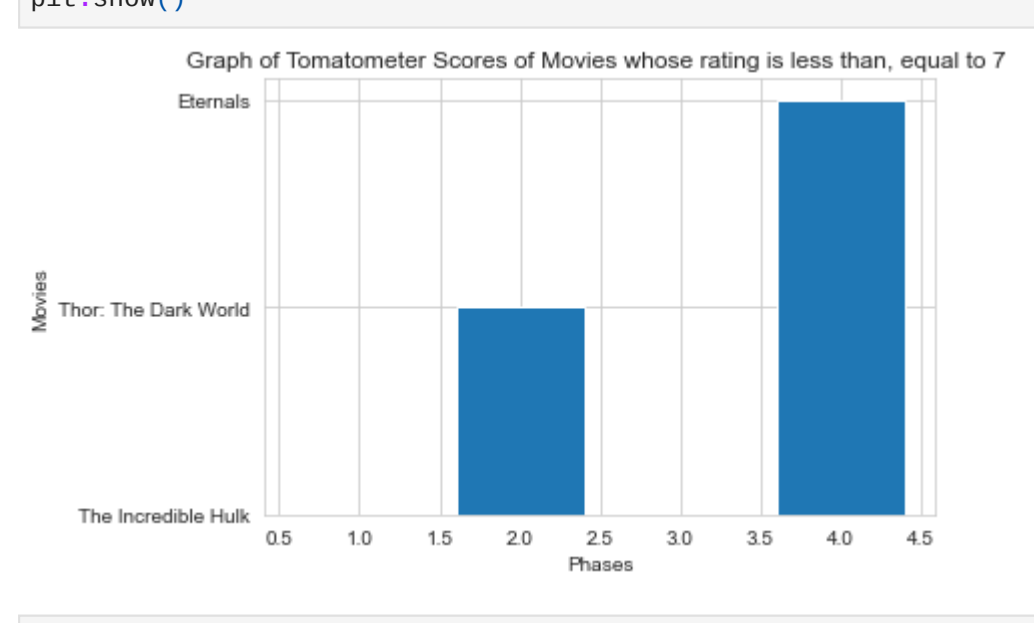
In [33]:
```python
x = df.loc[(df.phase == 4), 'worldwide_collection_in_million_(USD)']
y = df.loc[(df.phase == 4), 'season']
plt.bar(y, x)

plt.xlabel('Season Movies')
plt.ylabel('Worldwide collection')
plt.title('Phase 4 Movies: Worldwide collection according to Season')
plt.show()
```



In [24]:
```python
x = df.loc[(df.tomatometer>= 9), 'movie_title']
y = df.loc[(df.tomatometer>= 9), 'phase']
plt.bar( y,x)

plt.xlabel('Phases')
plt.ylabel('Movies')
plt.title('Tomatometer Scores of Movies whose rating is greater than, equal to 9 ')
plt.show()
```

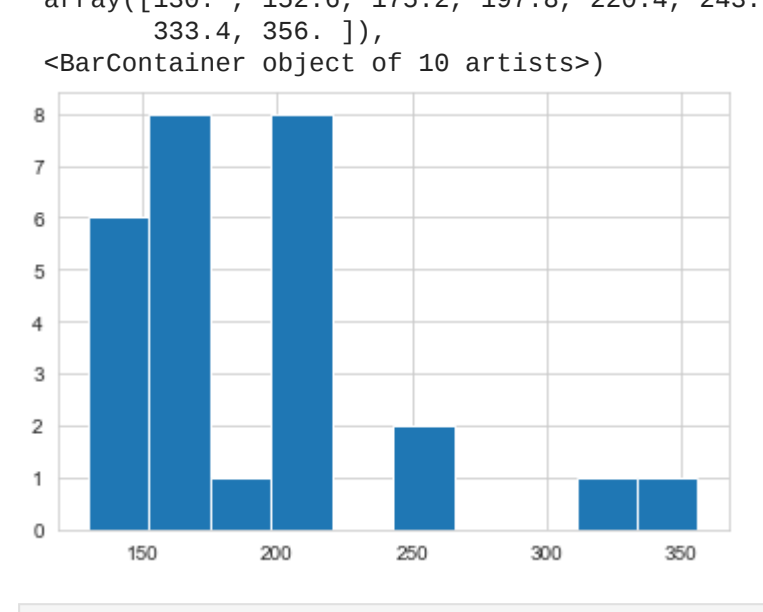


In [34]:
```python
x = df.loc[(df.tomatometer<= 7), 'movie_title']
y = df.loc[(df.tomatometer<= 7), 'phase']
plt.bar( y,x)

plt.xlabel('Phases')
plt.ylabel('Movies')
plt.title('Graph of Tomatometer Scores of Movies whose rating is less than, equal to 7 ')
plt.show()
```



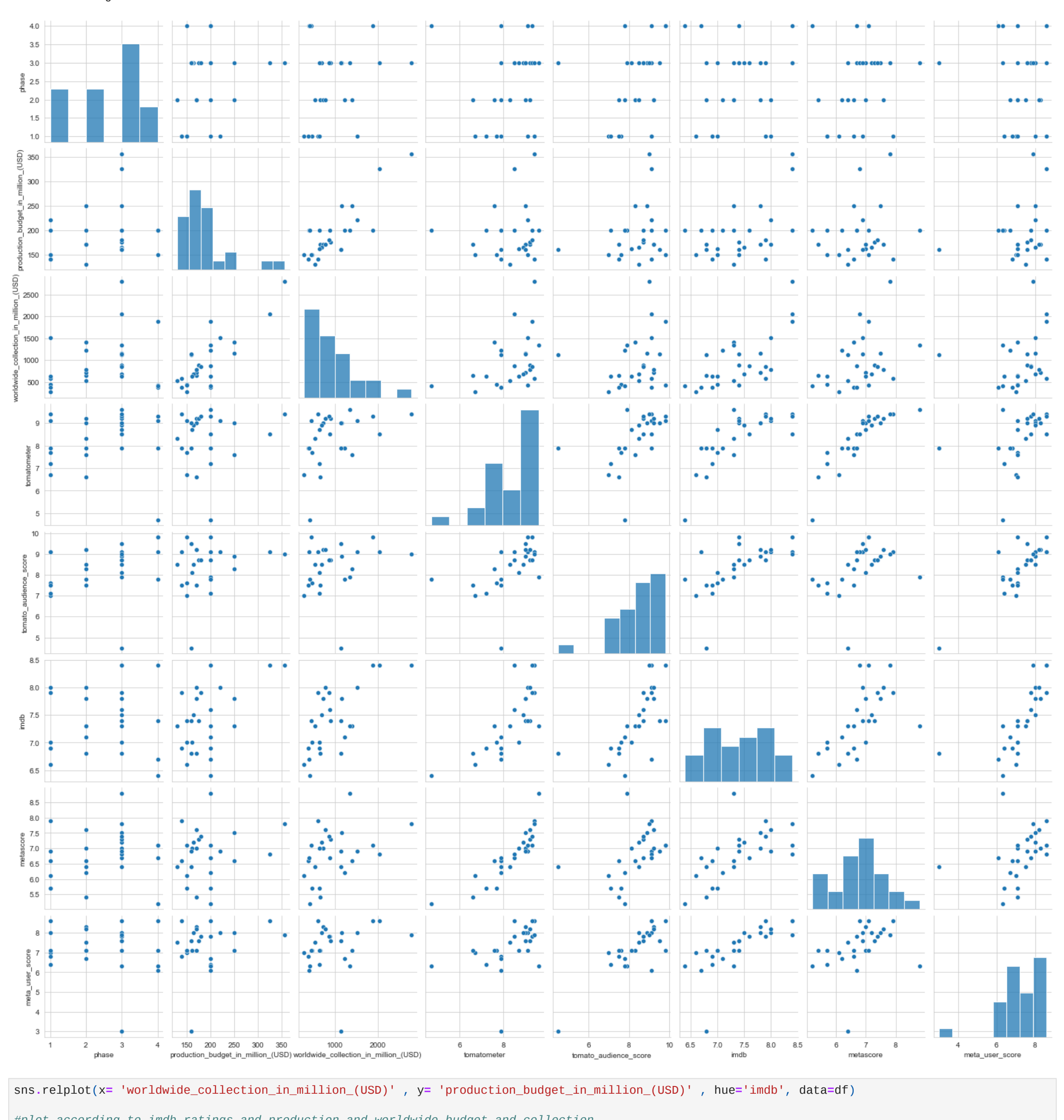
In [26]:
```python
plt.hist(df['production_budget_in_million_(USD)'])

#histogram of production budget of movies
```

Out[26]:
```
(array([6., 8., 1., 8., 0., 2., 0., 0., 1., 1.]),
 array([130. , 152.6, 175.2, 197.8, 220.4, 243. , 265.6, 288.2, 310.8,
        333.4, 356. ]),
 <BarContainer object of 10 artists>)
```



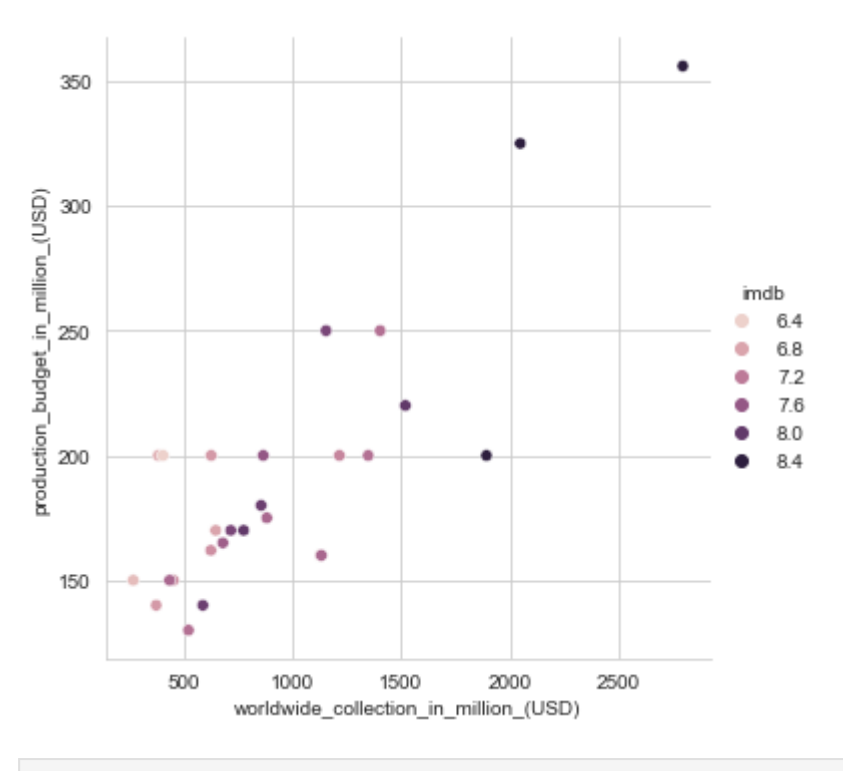
In [27]:
```python
sns.pairplot(df)
```

Out[27]: <seaborn.axisgrid.PairGrid at 0x23803231ed0>



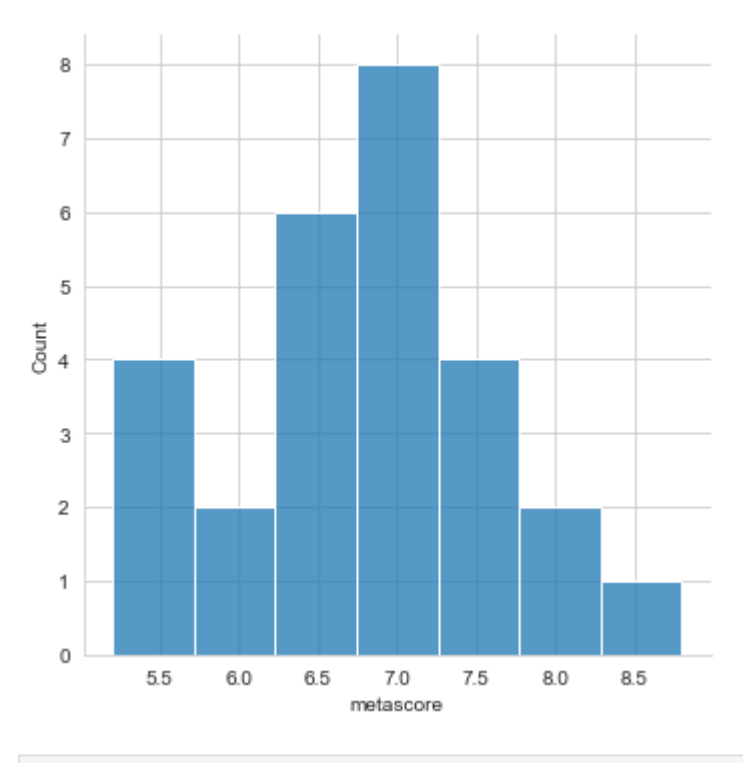
In [28]:
```python
sns.relplot(x= 'worldwide_collection_in_million_(USD)' , y= 'production_budget_in_million_(USD)' , hue='imdb', data=df)

#plot according to imdb ratings and production and worldwide budget and collection
```

Out[28]: <seaborn.axisgrid.FacetGrid at 0x238032ee710>



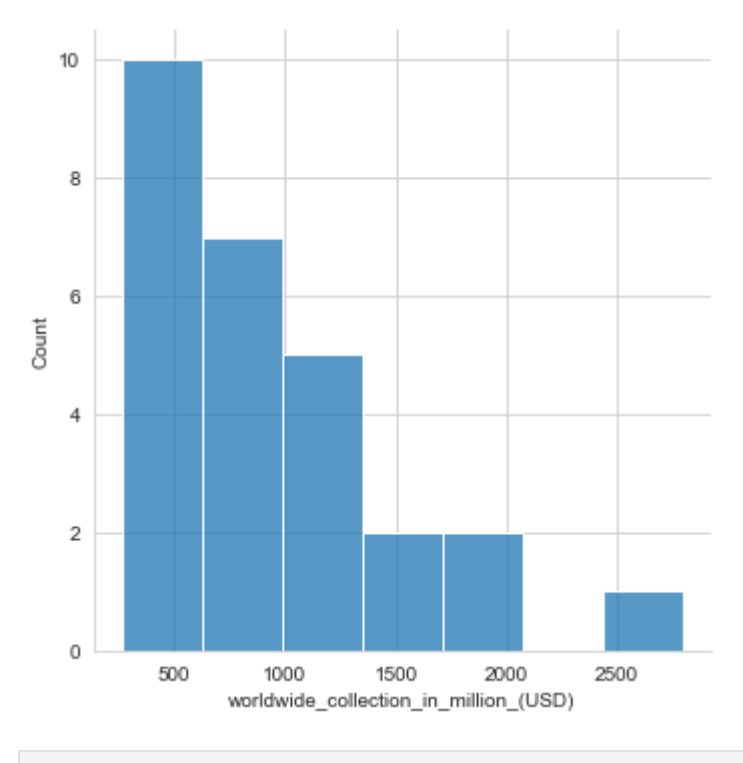
In [29]:
```python
sns.displot(df['metascore'])
```

Out[29]: <seaborn.axisgrid.FacetGrid at 0x238032ee230>



In [30]:
```python
sns.displot(df['worldwide_collection_in_million_(USD)'])
```

Out[30]: <seaborn.axisgrid.FacetGrid at 0x23803231ff0>



In [ ]:

In [ ]: