

```
In [16]: import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt
```

```
In [17]: data=pd.read_csv('IMDB-Movie-Data.csv')
```

```
In [18]: #First 10 rows of Dataset.
data.head(10)
```

Out[18]:

	Rank	Title	Genre	Description	Director	Actors
0	1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...
1	2	Prometheus	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...
2	3	Split	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...
3	4	Sing	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...
4	5	Suicide Squad	Action,Adventure,Fantasy	A secret government agency recruits some of th...	David Ayer	Will Smith, Jared Leto, Margot Robbie, Viola D...
5	6	The Great Wall	Action,Adventure,Fantasy	European mercenaries searching for black powde...	Yimou Zhang	Matt Damon, Tian Jing, Willem Dafoe, Andy Lau
6	7	La La Land	Comedy,Drama,Music	A jazz pianist falls for an aspiring actress i...	Damien Chazelle	Ryan Gosling, Emma Stone, Rosemarie DeWitt, J....
7	8	Mindhorn	Comedy	A has-been actor best known for playing the ti...	Sean Foley	Essie Davis, Andrea Riseborough, Julian Barrat...
8	9	The Lost City of Z	Action,Adventure,Biography	A true-life drama, centering on British explor...	James Gray	Charlie Hunnam, Robert Pattinson, Sienna Mille...
9	10	Passengers	Adventure,Drama,Romance	A spacecraft traveling to a distant colony pla...	Morten Tyldum	Jennifer Lawrence, Chris Pratt, Michael Sheen,...

```
In [19]: data.tail(10)
# Last 10 Rows of The Dataset
```

Out[19]:

	Rank	Title	Genre	Description	Director	Actors	Ye
990	991	Underworld: Rise of the Lycans	Action,Adventure,Fantasy	An origins story centered on the centuries-old...	Patrick Tatopoulos	Rhona Mitra, Michael Sheen, Bill Nighy, Steven...	201
991	992	Taare Zameen Par	Drama,Family,Music	An eight-year-old boy is thought to be a lazy ...	Aamir Khan	Darsheel Safary, Aamir Khan, Tanay Chheda, Sac...	201
992	993	Take Me Home Tonight	Comedy,Drama,Romance	Four years after graduation, an awkward high s...	Michael Dowse	Topher Grace, Anna Faris, Dan Fogler, Teresa P...	20
993	994	Resident Evil: Afterlife	Action,Adventure,Horror	While still out to destroy the evil Umbrella C...	Paul W.S. Anderson	Milla Jovovich, Ali Larter, Wentworth Miller,K...	20
994	995	Project X	Comedy	3 high school seniors throw a birthday party t...	Nima Nourizadeh	Thomas Mann, Oliver Cooper, Jonathan Daniel Br...	20
995	996	Secret in Their Eyes	Crime,Drama,Mystery	A tight-knit team of rising investigators, alo...	Billy Ray	Chiwetel Ejiofor, Nicole Kidman, Julia Roberts...	20
996	997	Hostel: Part II	Horror	Three American college students studying abroa...	Eli Roth	Lauren German, Heather Matarazzo, Bijou Philli...	201
997	998	Step Up 2: The Streets	Drama,Music,Romance	Romantic sparks occur between two dance studen...	Jon M. Chu	Robert Hoffman, Briana Evigan, Cassie Ventura,...	201
998	999	Search Party	Adventure,Comedy	A pair of friends embark on a mission to reuni...	Scot Armstrong	Adam Pally, T.J. Miller, Thomas Middleditch,Sh...	20
999	1000	Nine Lives	Comedy,Family,Fantasy	A stuffy businessman finds himself trapped ins...	Barry Sonnenfeld	Kevin Spacey, Jennifer Garner, Robbie Amell,Ch...	20

```
In [20]: data.shape  
#Number of Rows And Number of Columns.
```

```
Out[20]: (1000, 12)
```

```
In [21]: print("The number of rows",data.shape[0])  
print("The number of columns",data.shape[1])
```

```
The number of rows 1000  
The number of columns 12
```

```
In [22]: data.info()  
#Total Number Rows, Total Number of Columns  
#Datatypes of Each Column And Memory Requirement
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1000 entries, 0 to 999  
Data columns (total 12 columns):  
#   Column                Non-Null Count  Dtype  
---  -  
0   Rank                  1000 non-null  int64  
1   Title                 1000 non-null  object  
2   Genre                 1000 non-null  object  
3   Description            1000 non-null  object  
4   Director              1000 non-null  object  
5   Actors                1000 non-null  object  
6   Year                  1000 non-null  int64  
7   Runtime (Minutes)     1000 non-null  int64  
8   Rating                1000 non-null  float64  
9   Votes                 1000 non-null  int64  
10  Revenue (Millions)    872 non-null   float64  
11  Metascore             936 non-null   float64  
dtypes: float64(3), int64(4), object(5)  
memory usage: 93.9+ KB
```

```
In [23]: print("Any missing value?",data.isnull().values.any())
```

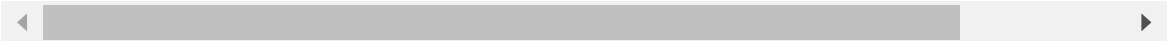
```
Any missing value? True
```

```
In [24]: #Check Null Values In The Dataset.
data.isnull()
```

Out[24]:

	Rank	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Rev (Mill)
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	
...	...	...	...	...	...	...	...	...	...	...	...
995	False	False	False	False	False	False	False	False	False	False	
996	False	False	False	False	False	False	False	False	False	False	
997	False	False	False	False	False	False	False	False	False	False	
998	False	False	False	False	False	False	False	False	False	False	
999	False	False	False	False	False	False	False	False	False	False	

1000 rows × 12 columns



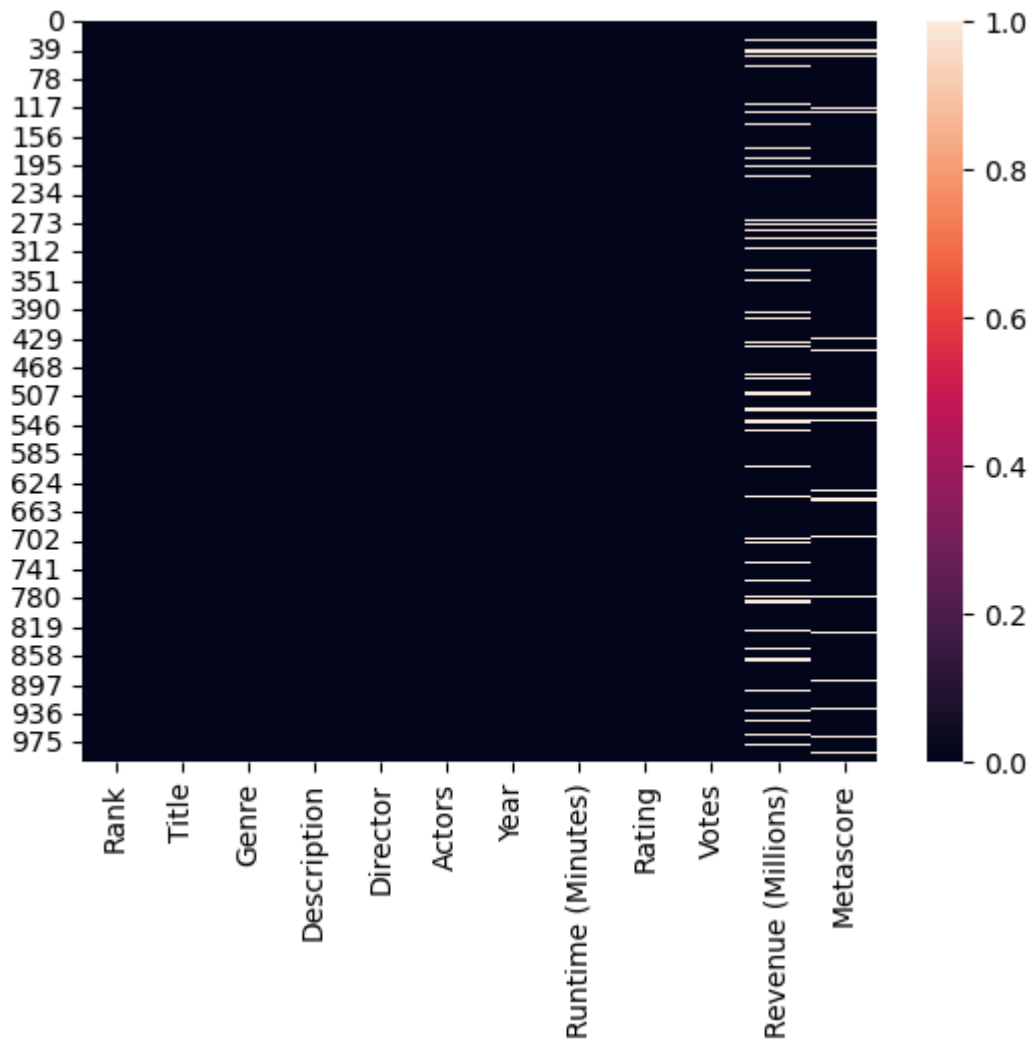
```
In [25]: data.isnull().sum()
```

Out[25]:

Rank	0
Title	0
Genre	0
Description	0
Director	0
Actors	0
Year	0
Runtime (Minutes)	0
Rating	0
Votes	0
Revenue (Millions)	128
Metascore	64
dtype:	int64

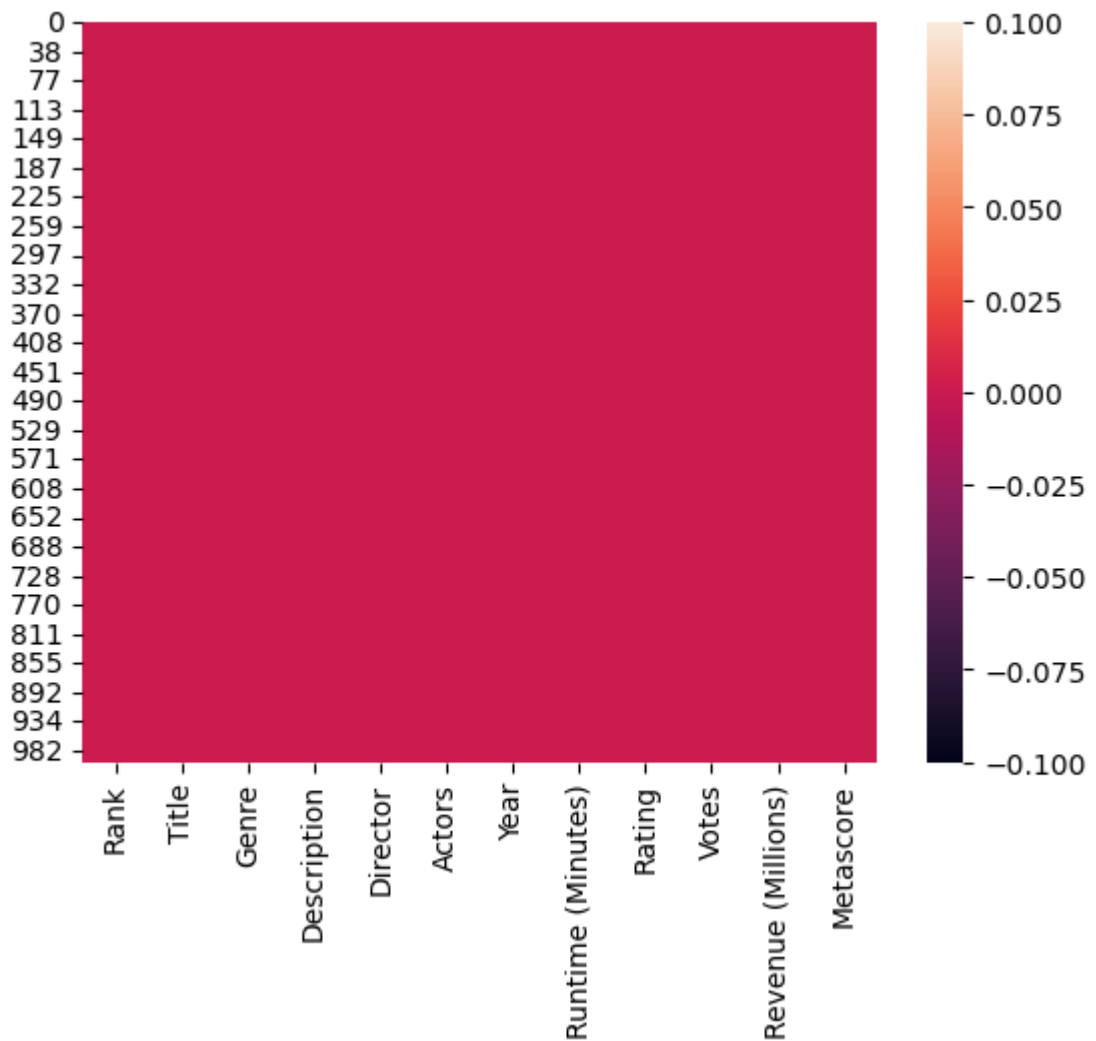
In [26]:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.heatmap(data.isnull())
plt.show()
```



```
In [27]: #Drop All The Missing Values.
data = data.dropna(axis=0)
```

```
In [28]: sns.heatmap(data.isnull())  
plt.show()
```



```
In [29]: #Check For Duplicate Data  
dup_data=data.duplicated().any()  
print("Are there any duplicated values in data?",dup_data)
```

Are there any duplicated values in data? False

In [30]: *#Get Overall Statistics About The DataFrame*  
`data.describe()`

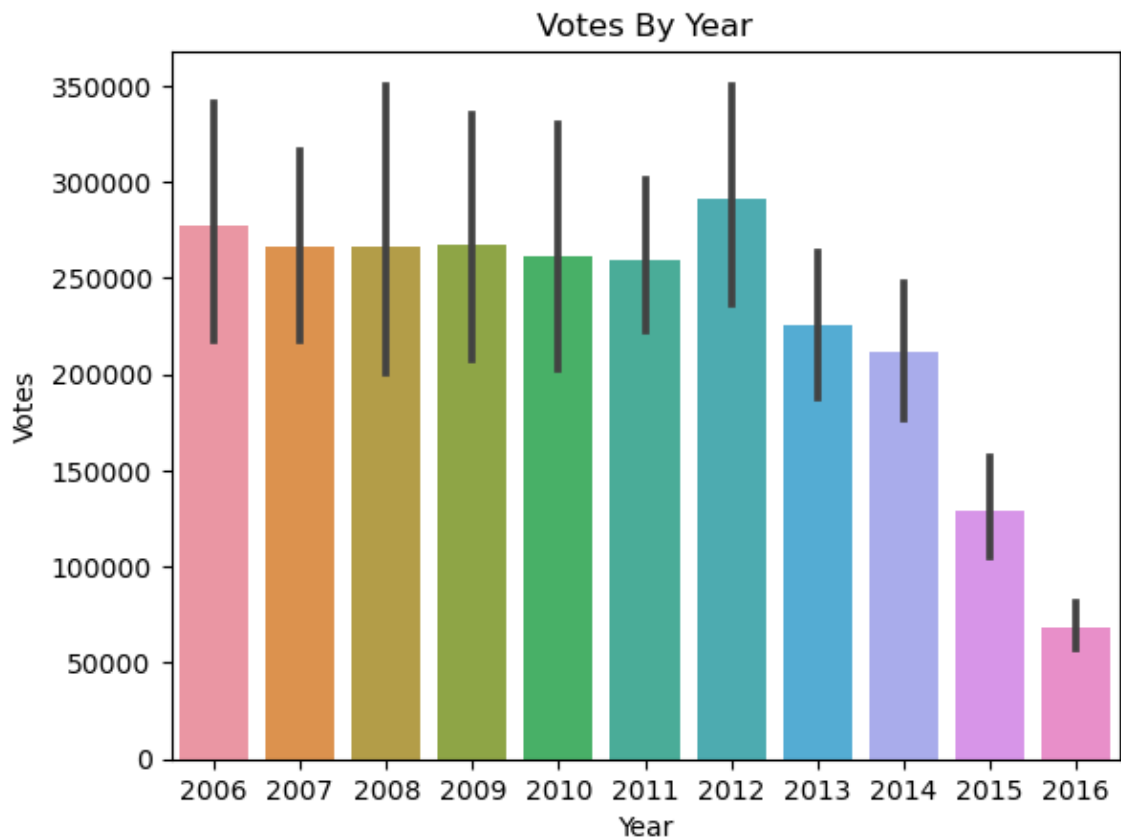
Out[30]:

	Rank	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metascore
count	838.000000	838.00000	838.000000	838.000000	8.380000e+02	838.000000	838.000000
mean	485.247017	2012.50716	114.638425	6.814320	1.932303e+05	84.564558	59.57517
std	286.572065	3.17236	18.470922	0.877754	1.930990e+05	104.520227	16.95241
min	1.000000	2006.00000	66.000000	1.900000	1.780000e+02	0.000000	11.00000
25%	238.250000	2010.00000	101.000000	6.300000	6.127650e+04	13.967500	47.00000
50%	475.500000	2013.00000	112.000000	6.900000	1.368795e+05	48.150000	60.00000
75%	729.750000	2015.00000	124.000000	7.500000	2.710830e+05	116.800000	72.00000
max	1000.000000	2016.00000	187.000000	9.000000	1.791916e+06	936.630000	100.00000

In [31]: *#Display Title of The Movie Having Runtime >= 180 Minutes*  
`data[data['Runtime (Minutes)']>=180]['Title']`

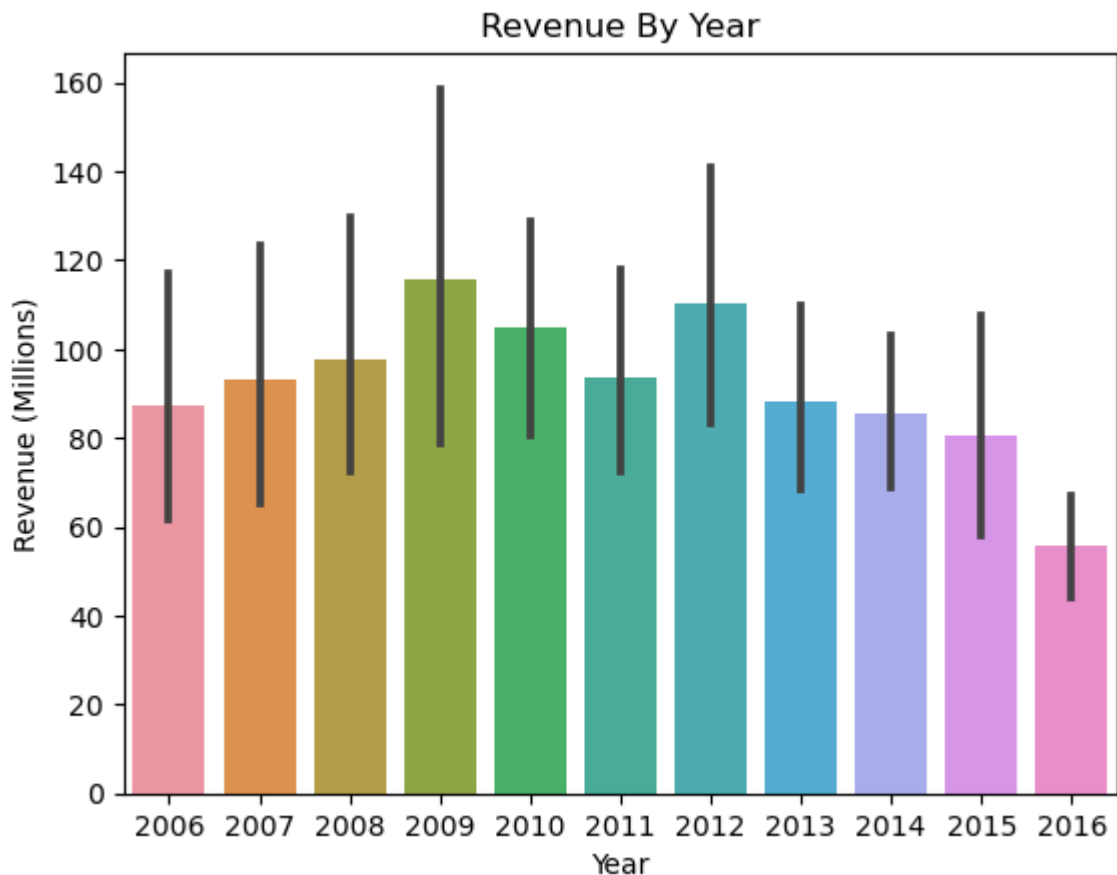
Out[31]: 82 The Wolf of Wall Street  
88 The Hateful Eight  
311 La vie d'Adèle  
Name: Title, dtype: object

In [32]: *# In Which Year There Was The Highest Voting?*  
`sns.barplot(x='Year',y='Votes',data=data)`  
`plt.title("Votes By Year")`  
`plt.show()`





```
In [33]: #In Which Year There Was The Highest Revenue?
sns.barplot(x='Year',y='Revenue (Millions)',data=data)
plt.title("Revenue By Year")
plt.show()
```



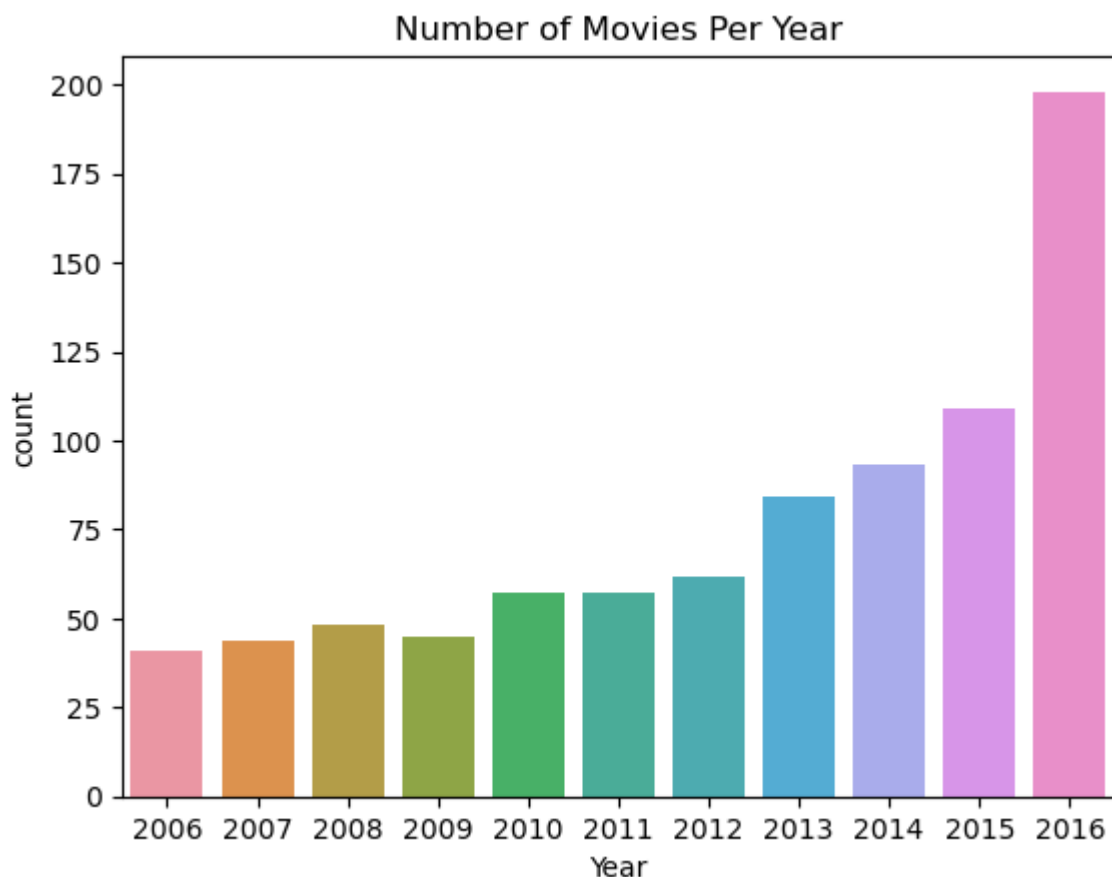
```
In [34]: #The Average Rating For Each Director.
data.groupby('Director')['Rating'].mean().sort_values(ascending=False)
```

```
Out[34]: Director
Christopher Nolan      8.68
Olivier Nakache        8.60
Makoto Shinkai         8.60
Florian Henckel von Donnersmarck  8.50
Aamir Khan             8.50
...
Sam Taylor-Johnson     4.10
Joey Curtis            4.00
George Nolfi           3.90
James Wong             2.70
Jason Friedberg        1.90
Name: Rating, Length: 524, dtype: float64
```

```
In [35]: #No.of movies per year.
le =data.nlargest(10,'Runtime (Minutes)')[['Title','Runtime (Minutes)']]. \
set_index('Title')
```

```
In [65]: sns.countplot(x='Year',data=data)
plt.title("Number of Movies Per Year")
```

```
Out[65]: Text(0.5, 1.0, 'Number of Movies Per Year')
```



```
In [39]: data.columns
```

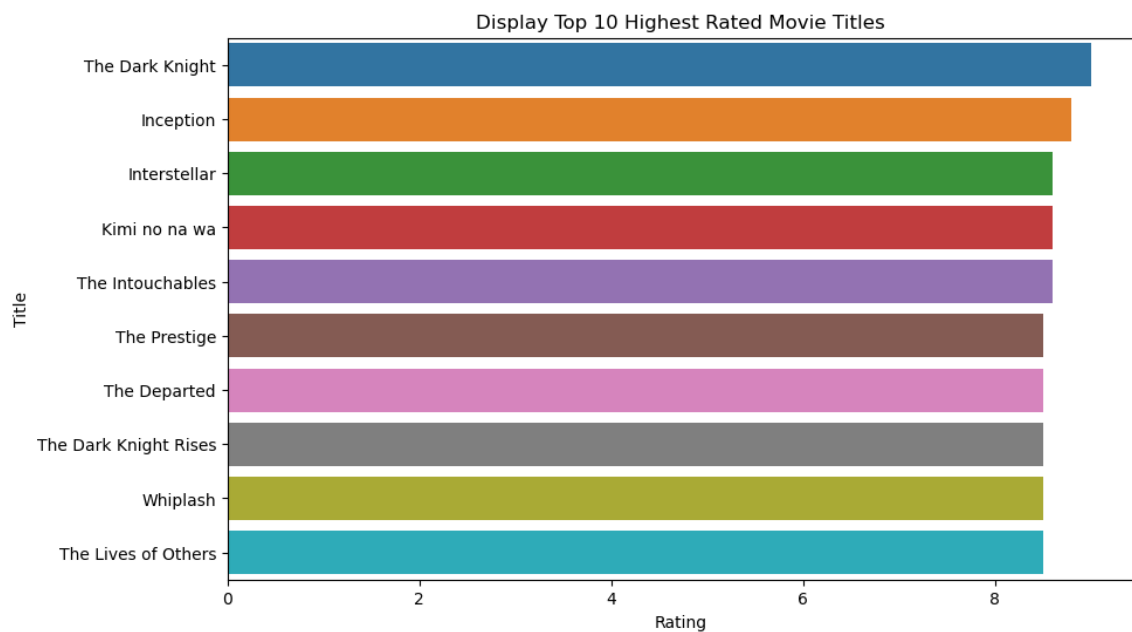
```
Out[39]: Index(['Rank', 'Title', 'Genre', 'Description', 'Director', 'Actors', 'Year',
               'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',
               'Metascore'],
              dtype='object')
```

```
In [40]: data[data['Revenue (Millions)'].max() == data['Revenue (Millions)']]['Title']
```

```
Out[40]: 50    Star Wars: Episode VII - The Force Awakens
Name: Title, dtype: object
```

```
In [41]: #Displaying top 10 highest rated movie titles.
top_10=data.nlargest(10,'Rating')[['Title','Rating','Director']].set_index(
```

```
In [63]: top_10 = data.nlargest(10, 'Rating')[['Title', 'Director', 'Rating']].set_index('Rating')
plt.figure(figsize=(10, 6))
sns.barplot(x='Rating', y=top_10.index, data=top_10)
plt.title("Display Top 10 Highest Rated Movie Titles")
plt.show()
```



```
In [45]: data.columns
```

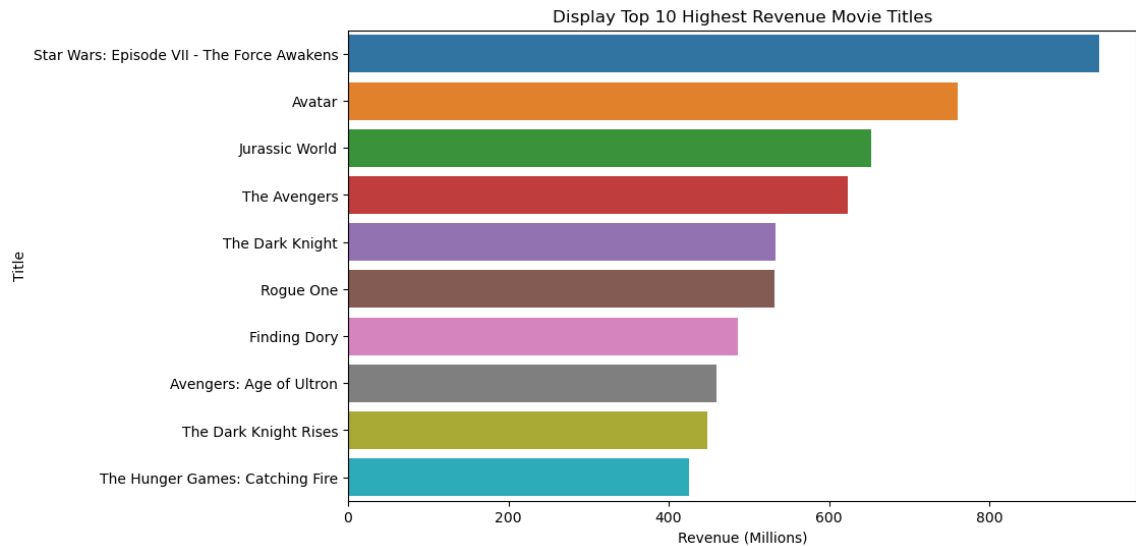
```
Out[45]: Index(['Rank', 'Title', 'Genre', 'Description', 'Director', 'Actors', 'Year',
               'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',
               'Metascore'],
              dtype='object')
```

```
In [46]: data.sort_values(by='Revenue (Millions)',ascending=False).head(10)
```

```
Out[46]:
```

	Rank	Title	Genre	Description	Director	Actors	Ye
50	51	Star Wars: Episode VII - The Force Awakens	Action,Adventure,Fantasy	Three decades after the defeat of the Galactic...	J.J. Abrams	Daisy Ridley, John Boyega, Oscar Isaac, Domhna...	20
87	88	Avatar	Action,Adventure,Fantasy	A paraplegic marine dispatched to the moon Pan...	James Cameron	Sam Worthington, Zoe Saldana, Sigourney Weaver...	20
85	86	Jurassic World	Action,Adventure,Sci-Fi	A new theme park, built on the original site o...	Colin Trevorrow	Chris Pratt, Bryce Dallas Howard, Ty Simpkins,...	20
76	77	The Avengers	Action,Sci-Fi	Earth's mightiest heroes must come together an...	Joss Whedon	Robert Downey Jr., Chris Evans, Scarlett Johan...	20
54	55	The Dark Knight	Action,Crime,Drama	When the menace known as the Joker wreaks havo...	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart,Mi...	20
12	13	Rogue One	Action,Adventure,Sci-Fi	The Rebel Alliance makes a risky move to steal...	Gareth Edwards	Felicity Jones, Diego Luna, Alan Tudyk, Donnie...	20
119	120	Finding Dory	Animation,Adventure,Comedy	The friendly but forgetful blue tang fish, Dor...	Andrew Stanton	Ellen DeGeneres, Albert Brooks,Ed O'Neill, Kai...	20
94	95	Avengers: Age of Ultron	Action,Adventure,Sci-Fi	When Tony Stark and Bruce Banner try to jump-s...	Joss Whedon	Robert Downey Jr., Chris Evans, Mark Ruffalo, ...	20
124	125	The Dark Knight Rises	Action,Thriller	Eight years after the Joker's reign of anarchy...	Christopher Nolan	Christian Bale, Tom Hardy, Anne Hathaway,Gary ...	20
578	579	The Hunger Games: Catching Fire	Action,Adventure,Mystery	Katniss Everdeen and Peeta Mellark become targ...	Francis Lawrence	Jennifer Lawrence, Josh Hutcherson, Liam Hemsw...	20

```
In [60]: #Displaying top 10 highest revenue movie titles.
top_10 = data.nlargest(10, 'Revenue (Millions)')[['Title', 'Director', 'Revenue (Millions)']]
plt.figure(figsize=(10, 6))
sns.barplot(x='Revenue (Millions)', y=top_10.index, data=top_10)
plt.title("Display Top 10 Highest Revenue Movie Titles")
plt.show()
```



```
In [50]: data.columns
```

```
Out[50]: Index(['Rank', 'Title', 'Genre', 'Description', 'Director', 'Actors', 'Year',
               'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',
               'Metascore'],
              dtype='object')
```

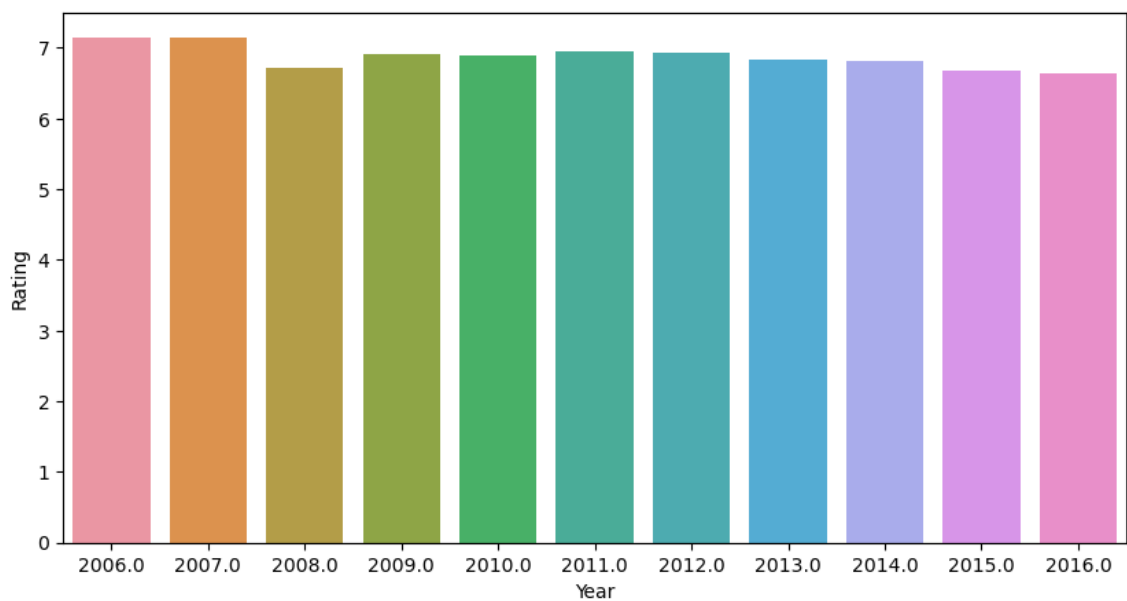
```
In [55]: data1 = data.groupby('Year')[['Year', 'Rating']].mean()\
          .sort_values(by='Rating', ascending=False)\
          .set_index('Year')
```

```
In [56]: data1
```

```
Out[56]:
```

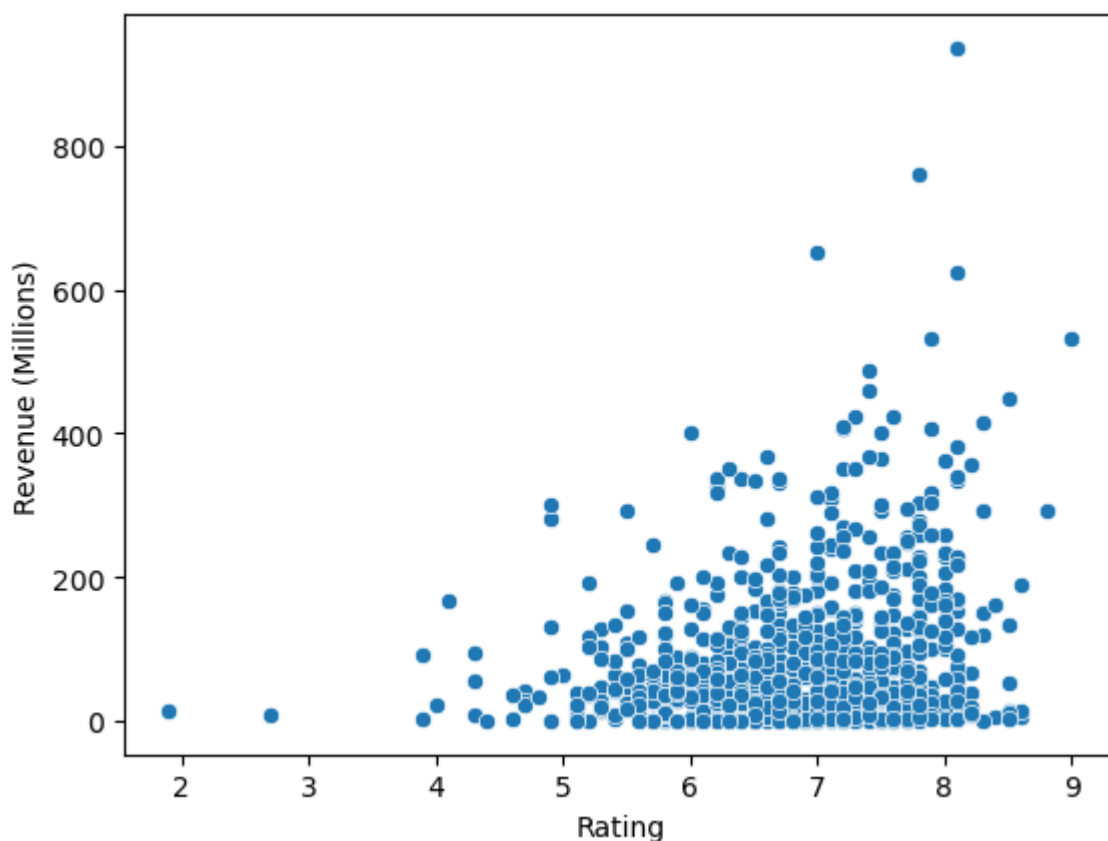
	Rating
Year	
2006.0	7.143902
2007.0	7.140909
2011.0	6.945614
2012.0	6.933871
2009.0	6.911111
2010.0	6.894737
2013.0	6.832143
2014.0	6.822581
2008.0	6.708333
2015.0	6.674312
2016.0	6.644444

```
In [58]: #Average rating of movie per year.  
plt.figure(figsize=(10, 5))  
sns.barplot(x=data1.index, y=data1['Rating'])  
plt.show()
```



```
In [66]: sns.scatterplot(x='Rating',y='Revenue (Millions)',data=data)
```

```
Out[66]: <Axes: xlabel='Rating', ylabel='Revenue (Millions)'>
```



```
In [67]: data.columns
```

```
Out[67]: Index(['Rank', 'Title', 'Genre', 'Description', 'Director', 'Actors', 'Year',  
              'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',  
              'Metascore'],  
             dtype='object')
```

```
In [68]: def rating(rating):  
         if rating >= 7.0:  
             return 'Excellent'  
         elif rating >= 6.0:  
             return 'Good'  
         else:  
             return 'Average'
```

```
In [69]: data['rating_cat'] = data['Rating'].apply(rating)
```

In [76]: `data.head()`

Out[76]:

	Rank	Title	Genre	Description	Director	Actors	Y
0	1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2
1	2	Prometheus	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2
2	3	Split	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2
3	4	Sing	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	2
4	5	Suicide Squad	Action,Adventure,Fantasy	A secret government agency recruits some of th...	David Ayer	Will Smith, Jared Leto, Margot Robbie, Viola D...	2



In [77]: `data.columns`

Out[77]: Index(['Rank', 'Title', 'Genre', 'Description', 'Director', 'Actors', 'Year',  
'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',  
'Metascore', 'rating\_cat', 'temp'],  
dtype='object')

In [78]: `data['Genre'].dtype`

Out[78]: dtype('O')

In [80]: `len(data[data['Genre'].str.contains('Action',case=False)])`

Out[80]: 277

In [81]: `data.columns`

Out[81]: Index(['Rank', 'Title', 'Genre', 'Description', 'Director', 'Actors', 'Year',  
'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',  
'Metascore', 'rating\_cat', 'temp'],  
dtype='object')



```
In [82]: data['Genre']
```

```
Out[82]: 0      Action,Adventure,Sci-Fi
1      Adventure,Mystery,Sci-Fi
2              Horror,Thriller
3      Animation,Comedy,Family
4      Action,Adventure,Fantasy
...
993     Action,Adventure,Horror
994              Comedy
996              Horror
997      Drama,Music,Romance
999     Comedy,Family,Fantasy
Name: Genre, Length: 838, dtype: object
```

```
In [83]: list=[]
for value in data['Genre']:
    list1.append(value.split(','))
```

```
In [84]: list1
```

```
['Action', 'Adventure', 'Sci-Fi'],
['Action', 'Adventure', 'Fantasy'],
['Action', 'Adventure', 'Fantasy'],
['Comedy', 'Drama'],
['Action', 'Crime', 'Thriller'],
['Action', 'Crime', 'Drama'],
['Adventure', 'Drama', 'History'],
['Crime', 'Horror', 'Thriller'],
['Drama', 'Romance'],
['Comedy', 'Drama', 'Romance'],
['Biography', 'Drama'],
['Action', 'Adventure', 'Sci-Fi'],
['Crime', 'Drama', 'Mystery'],
['Drama', 'Romance', 'Thriller'],
['Drama', 'Mystery', 'Sci-Fi'],
['Action', 'Adventure', 'Comedy'],
['Drama', 'History', 'Thriller'],
['Action', 'Adventure', 'Sci-Fi'],
['Drama'],
['Action', 'Drama', 'Thriller'],
...
```

```
In [85]: one_d=[]
for item in list1:
    for item1 in item:
        one_d.append(item1)
```

```
In [86]: one_d
        history,
        'Action',
        'Thriller',
        'Biography',
        'Drama',
        'Drama',
        'Mystery',
        'Sci-Fi',
        'Adventure',
        'Drama',
        'Thriller',
        'Drama',
        'Animation',
        'Adventure',
        'Comedy',
        'Action',
        'Adventure',
        'Sci-Fi',
        'Comedy',
        'Action',
        'Action',
```

```
In [89]: uni_list=[]
        for item in one_d:
            if item not in uni_list:
                uni_list.append(item)
```

```
In [92]: uni_list
```

```
Out[92]: ['Action',
          'Adventure',
          'Sci-Fi',
          'Mystery',
          'Horror',
          'Thriller',
          'Animation',
          'Comedy',
          'Family',
          'Fantasy',
          'Drama',
          'Music',
          'Biography',
          'Romance',
          'History',
          'Western',
          'Crime',
          'War',
          'Musical',
          'Sport']
```

```
In [93]: one_d=[]
        for item in list1:
            for item1 in item:
                one_d.append(item1)
```

```
In [94]: one_d
         Romance',
         'Adventure',
         'Family',
         'Fantasy',
         'Biography',
         'Drama',
         'History',
         'Action',
         'Adventure',
         'Sci-Fi',
         'Animation',
         'Adventure',
         'Comedy',
         'Action',
         'Comedy',
         'Drama',
         'Animation',
         'Adventure',
         'Comedy',
         'Biography',
         'Romance',
```

```
In [96]: from collections import Counter
```

```
In [97]: Counter(one_d)
```

```
Out[97]: Counter({'Drama': 838,
                  'Action': 554,
                  'Comedy': 500,
                  'Adventure': 488,
                  'Thriller': 296,
                  'Crime': 252,
                  'Romance': 240,
                  'Sci-Fi': 214,
                  'Fantasy': 184,
                  'Horror': 174,
                  'Mystery': 172,
                  'Biography': 134,
                  'Family': 96,
                  'Animation': 90,
                  'History': 50,
                  'Music': 30,
                  'Sport': 30,
                  'War': 20,
                  'Musical': 10,
                  'Western': 8})
```

```
In [ ]:
```