In [1]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
```

In [2]:

```
df= pd.read_csv("IMDB-Movie-Data.csv")
```

Collecting information about data

In [3]:

```
# to display top 3 rows

df.head(3)
```

Out[3]:

	Rank	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)
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...	2014	121
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...	2012	124
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...	2016	117

In [4]:

```
# to display last rows of dataset leaving top3 rows from starting

df.tail(-3)
```

Out[4]:

	Rank	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)
3	4	Sing	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	2016	108
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...	2016	123
5	6	The Great Wall	Action,Adventure,Fantasy	European mercenaries searching for black powde...	Yimou Zhang	Matt Damon, Tian Jing, Willem Dafoe, Andy Lau	2016	103

In [5]:

```
df.shape
```

Out[5]:

```
(1000, 12)
```

In [6]:

```
df.columns
```

Out[6]:

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

In [7]:

```
# to see statistics of all the columns in dataset

df.describe()
```

Out[7]:

	Rank	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metas
count	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	872.000000	936.00
mean	500.500000	2012.783000	113.172000	6.723200	1.698083e+05	82.956376	58.98
std	288.819436	3.205962	18.810908	0.945429	1.887626e+05	103.253540	17.19
min	1.000000	2006.000000	66.000000	1.900000	6.100000e+01	0.000000	11.00
25%	250.750000	2010.000000	100.000000	6.200000	3.630900e+04	13.270000	47.00
50%	500.500000	2014.000000	111.000000	6.800000	1.107990e+05	47.985000	59.50
75%	750.250000	2016.000000	123.000000	7.400000	2.399098e+05	113.715000	72.00
max	1000.000000	2016.000000	191.000000	9.000000	1.791916e+06	936.630000	100.00

In [8]:

```
# to see all the informations of columns in dataset

df.info()
```

```
<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
```

DATA CLEANING

In [9]:

```
# Checking missing values

df.isnull().sum()
```

Out[9]:

```
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
```

The above code is showing 128 null values are present in revenue column and 64 null values in metascore column

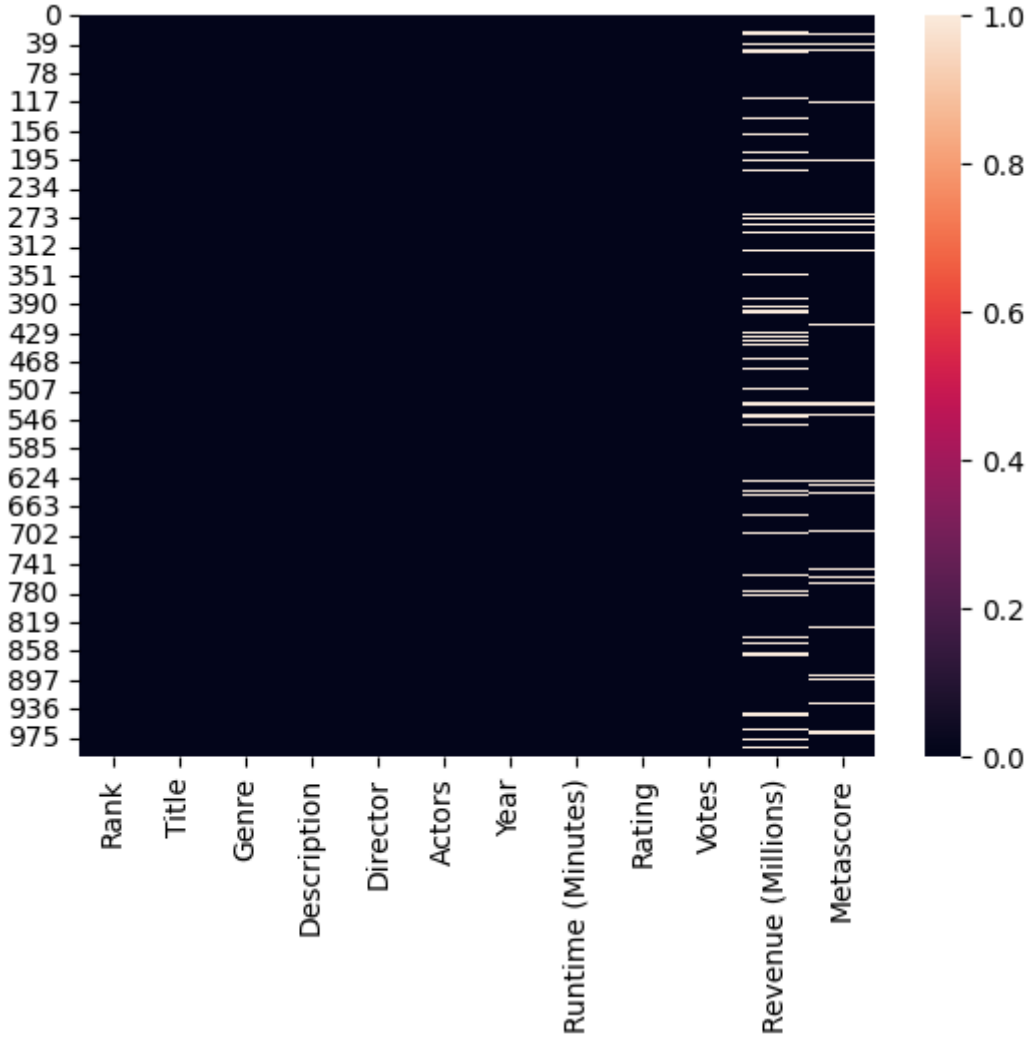
In [10]:

```
# visualising missing values

sns.heatmap(df.isnull())
```

Out[10]:

```
<Axes: >
```



Below code is demonstrating 12.8% missing values in revenue column and 6.4% missing values in metascore coumn

In [11]:

```
# checking how much % of missing values present in dataset


df.isnull().sum() * 100 / len(df)
```

Out[11]:

```
Rank                   0.0
Title                  0.0
Genre                  0.0
Description            0.0
Director               0.0
Actors                 0.0
Year                   0.0
Runtime (Minutes)      0.0
Rating                 0.0
Votes                  0.0
Revenue (Millions)    12.8
Metascore              6.4
dtype: float64
```

In [12]:

```
# In Revenue(Millions) column, 12.8% values are missing so we are filling those values wi


df['Revenue (Millions)'].fillna(df['Revenue (Millions)'].mean(),inplace=True)
```

Filling 12.8% missing values of revenue column with the mean of same column

In [13]:

```
# In Meta-score, there are 6.4% missing values let's treat them

df['Metascore'].fillna(df['Metascore'].mean(),inplace=True)
```

Filled 6.4% missing values of Metascore column with its mean

In [14]:

```
df.info()
```

```
<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)  1000 non-null   float64
 11  Metascore           1000 non-null   float64
dtypes: float64(3), int64(4), object(5)
memory usage: 93.9+ KB
```

Checking Duplicate data

In [15]:

```
# checking if any duplicate data present in dataset
# approach 1

df[df.duplicated()]
```

Out[15]:

Rank	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)

In [16]:

```
# checking duplicate data
# approach 2

df.duplicated().any()
```

Out[16]:

```
False
```

overall statistics of all columns including object data type

In [17]:

```
df.describe(include='all')
```

Out[17]:

	Rank	Title	Genre	Description	Director	Actors	Yea
count	1000.000000	1000	1000	1000	1000	1000	1000.00000
unique	NaN	999	207	1000	644	996	Na
top	NaN	The Host	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	Ridley Scott	Jennifer Lawrence, Josh Hutcherson, Liam Hemsw...	Na
freq	NaN	2	50	1	8	2	Na
mean	500.500000	NaN	NaN	NaN	NaN	NaN	2012.78300
std	288.819436	NaN	NaN	NaN	NaN	NaN	3.20596
min	1.000000	NaN	NaN	NaN	NaN	NaN	2006.00000
25%	250.750000	NaN	NaN	NaN	NaN	NaN	2010.00000
50%	500.500000	NaN	NaN	NaN	NaN	NaN	2014.00000
75%	750.250000	NaN	NaN	NaN	NaN	NaN	2016.00000
max	1000.000000	NaN	NaN	NaN	NaN	NaN	2016.00000

DATA ANALYSIS

In [18]:

```
# Showing title of movies which are having runtime of more than or equal to 180 minutes

df[df['Runtime (Minutes)'] >= 180]['Title']
```

Out[18]:

```
82     The Wolf of Wall Street
88           The Hateful Eight
311             La vie d'Adèle
828                 Grindhouse
965              Inland Empire
Name: Title, dtype: object
```

In [19]:

```
# Showing in which year there was highest average voting
# approach 1

df.groupby('Year')['Votes'].mean().sort_values( ascending = False)
```

Out[19]:

```
Year
2012    285226.093750
2008    275505.384615
2006    269289.954545
2009    255780.647059
2010    252782.316667
2007    244331.037736
2011    240790.301587
2013    219049.648352
2014    203930.224490
2015    115726.220472
2016     48591.754209
Name: Votes, dtype: float64
```

In [20]:

```
# Showing the highest average voting


df.groupby('Year')['Votes'].mean().max()
```

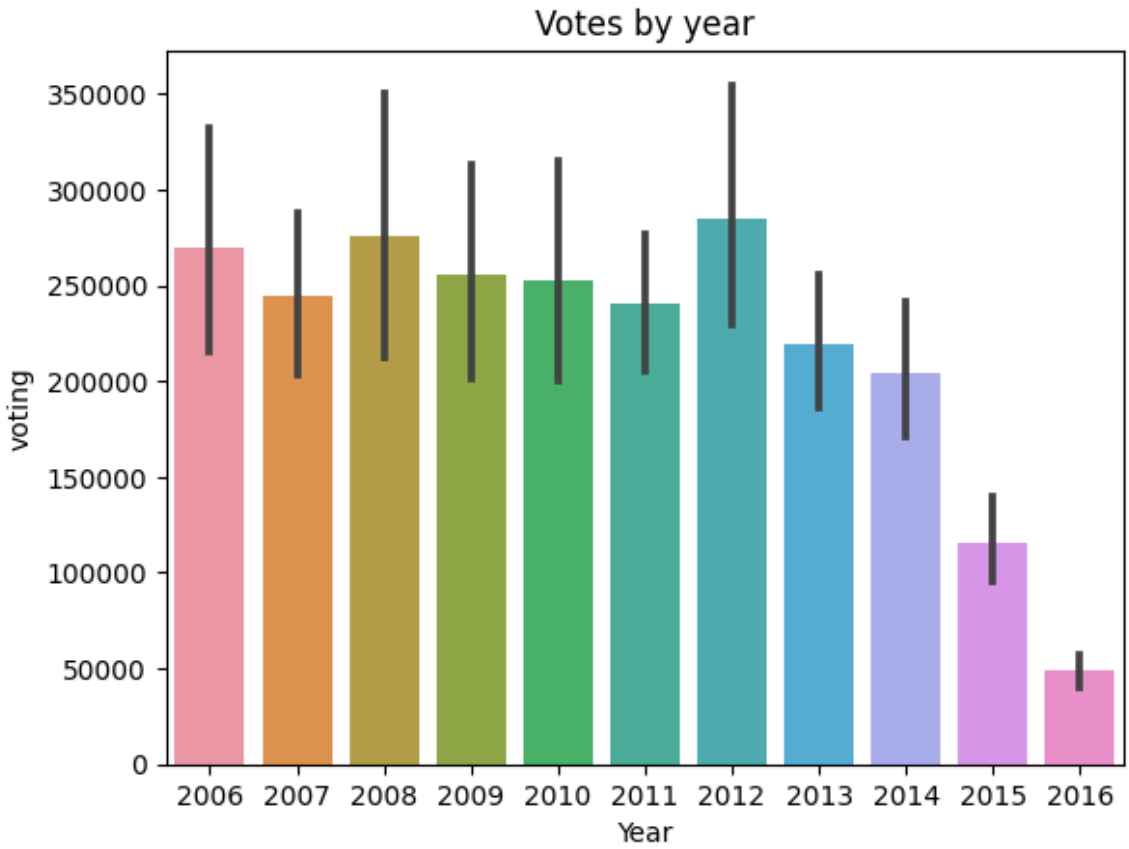
Out[20]:

```
285226.09375
```

Visualising highest average voting

In [21]:

```
sns.barplot(x='Year' , y='Votes' , data=df)
plt.title('Votes by year')
plt.ylabel('voting')
plt.show()
```



In the above bar plot, it is visible that highest voting has been performed in the year 2012

In [22]:

```
# Year having highest average revenue

df.groupby('Year')['Revenue (Millions)'].mean().sort_values(ascending=False)
```

Out[22]:

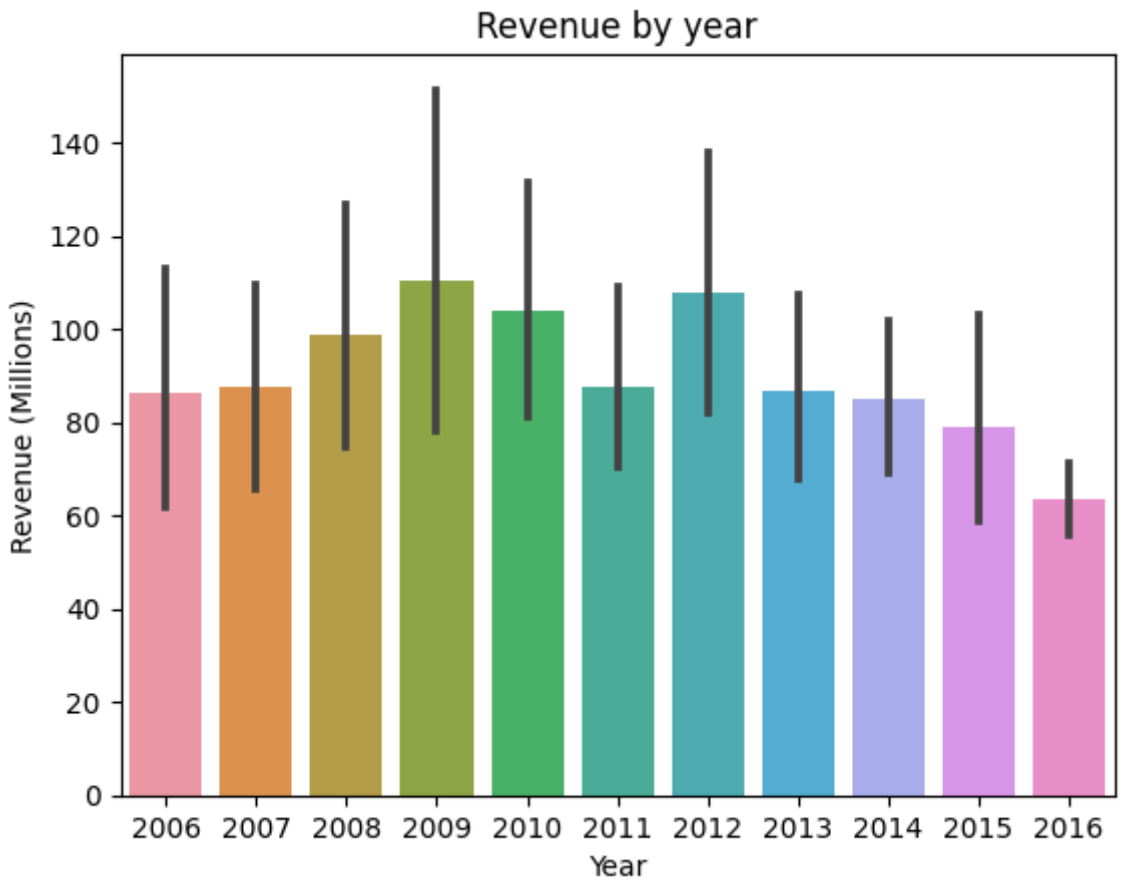
```
Year
2009    110.276186
2012    107.973281
2010    103.975319
2008     98.772623
2011     87.538355
2007     87.510481
2013     86.984496
2006     86.144835
2014     84.992097
2015     78.862278
2016     63.446588
Name: Revenue (Millions), dtype: float64
```

In [23]:

```
sns.barplot(x='Year', y='Revenue (Millions)', data =df)
plt.title('Revenue by year')
```

Out[23]:

```
Text(0.5, 1.0, 'Revenue by year')
```



2009 is having highest revenue

In [24]:

```
# average rating for each director

df.groupby('Director')['Rating'].mean().sort_values(ascending=False)
```

Out[24]:

```
Director
Nitesh Tiwari         8.80
Christopher Nolan     8.68
Olivier Nakache       8.60
Makoto Shinkai        8.60
Aamir Khan            8.50
                      ...
Micheal Bafaro        3.50
Jonathan Holbrook     3.20
Shawn Burkett         2.70
James Wong            2.70
Jason Friedberg       1.90
Name: Rating, Length: 644, dtype: float64
```

In [25]:

```
df['Director'].value_counts()
```

Out[25]:

```
Director
Ridley Scott          8
David Yates           6
M. Night Shyamalan    6
Paul W.S. Anderson    6
Michael Bay           6
                     ..
Lee Toland Krieger    1
Gillies MacKinnon     1
Peter Atencio         1
James Mangold         1
Scot Armstrong        1
Name: count, Length: 644, dtype: int64
```

TOP 10 LENGTHY MOVIES

In [26]:

```
top10_movies=df.sort_values(by='Runtime (Minutes)',ascending=False) [['Title' , 'Runtime
top10_movies
```

Out[26]:

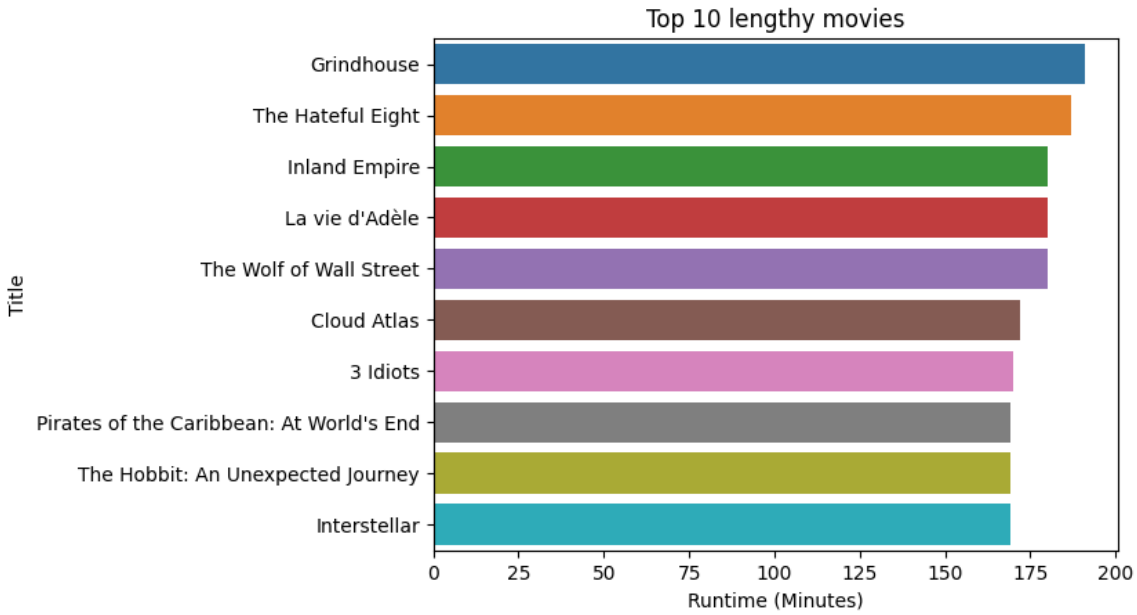
	Title	Runtime (Minutes)
828	Grindhouse	191
88	The Hateful Eight	187
965	Inland Empire	180
311	La vie d'Adèle	180
82	The Wolf of Wall Street	180
267	Cloud Atlas	172
430	3 Idiots	170
75	Pirates of the Caribbean: At World's End	169
271	The Hobbit: An Unexpected Journey	169
36	Interstellar	169

In [27]:

```
sns.barplot(y='Title', x='Runtime (Minutes)', data= top10_movies)
plt.title('Top 10 lengthy movies')
```

Out[27]:

```
Text(0.5, 1.0, 'Top 10 lengthy movies')
```



The Hateful Eight is the most lengthy movie with runtime of 187 minutes

In [28]:

```
# top 10 lengthy movies but without pre built index and treating Title as index

top10=df.sort_values(by='Runtime (Minutes)',ascending=False) [['Title' , 'Runtime (Minute
top10
```

Out[28]:

Title	Runtime (Minutes)
Grindhouse	191
The Hateful Eight	187
Inland Empire	180
La vie d'Adèle	180
The Wolf of Wall Street	180
Cloud Atlas	172
3 Idiots	170
Pirates of the Caribbean: At World's End	169
The Hobbit: An Unexpected Journey	169
Interstellar	169

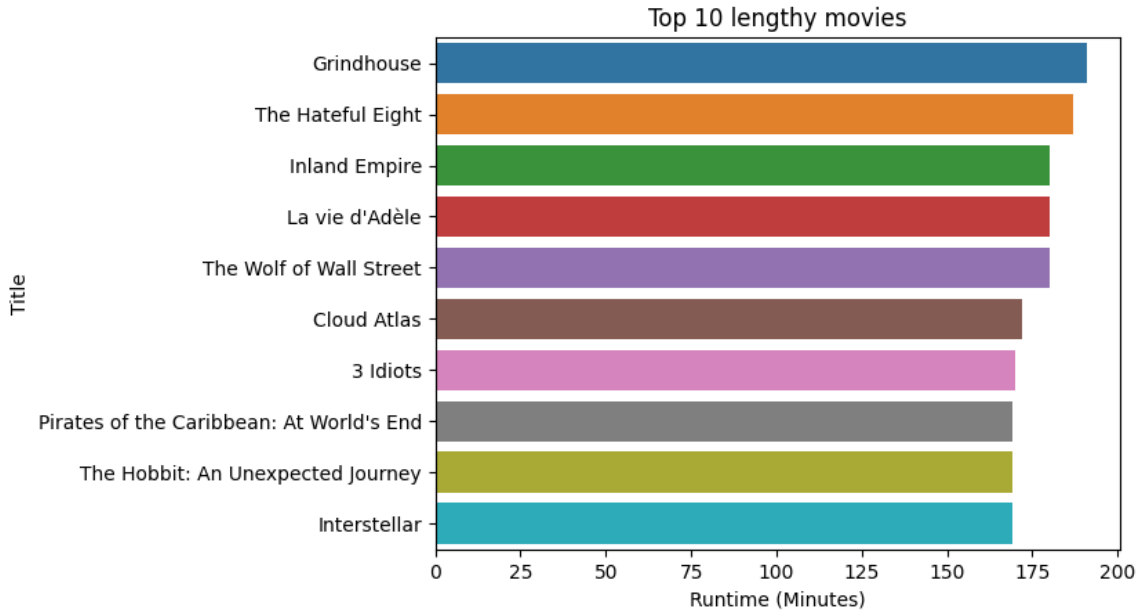
In [29]:

```
# using index function as title column becoame index in ths scenario

sns.barplot(x='Runtime (Minutes)', y=top10.index , data= top10)
plt.title('Top 10 lengthy movies')
```

Out[29]:

```
Text(0.5, 1.0, 'Top 10 lengthy movies')
```



Barplot is using .index() method for accessing the values of x-axis which has become index now for this graph

The Hateful Eight is the most lengthy movie with runtime of 187 minutes

In [30]:

```
# No. of movies each year
# Approach 1


df.groupby('Year')['Rank'].count().sort_values(ascending=False)
```

Out[30]:

```
Year
2016    297
2015    127
2014     98
2013     91
2012     64
2011     63
2010     60
2007     53
2008     52
2009     51
2006     44
Name: Rank, dtype: int64
```

In [31]:

```
# Approach 2


df['Year'].value_counts()
```

Out[31]:

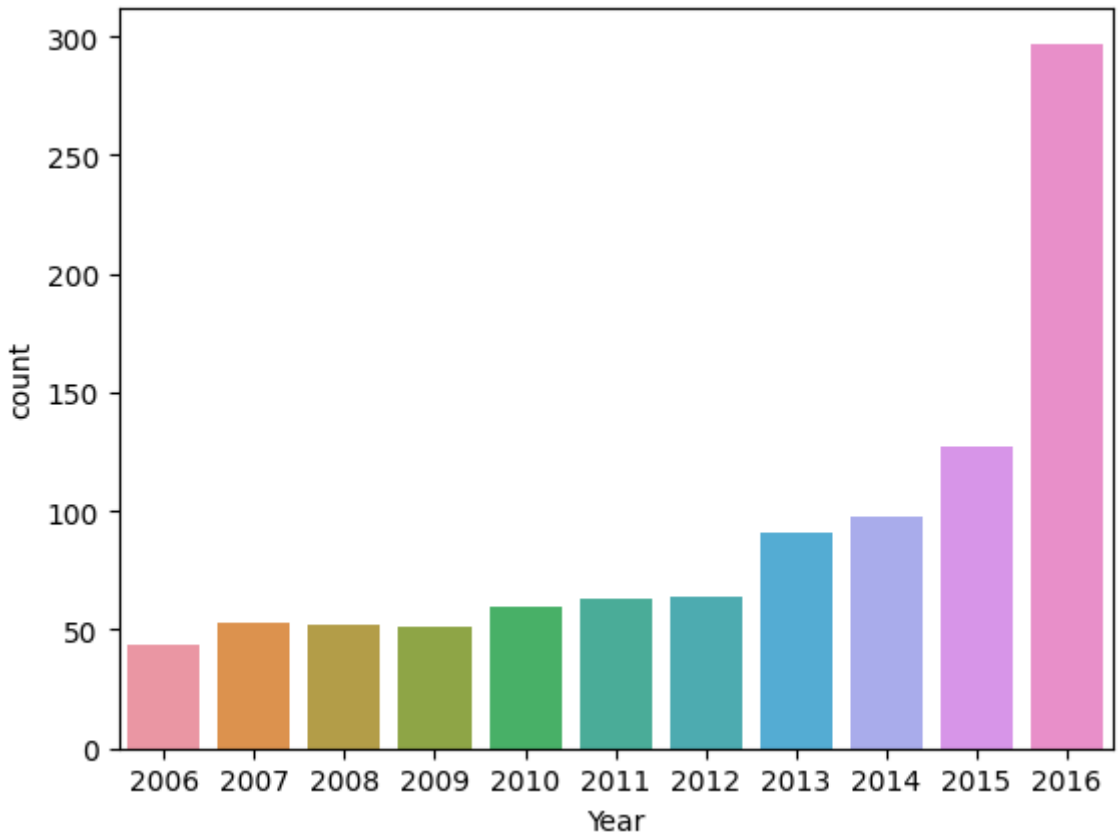
```
Year
2016    297
2015    127
2014     98
2013     91
2012     64
2011     63
2010     60
2007     53
2008     52
2009     51
2006     44
Name: count, dtype: int64
```

In [32]:

```
sns.countplot(x='Year',data=df)
```

Out[32]:

```
<Axes: xlabel='Year', ylabel='count'>
```



Above graph is representing 2016 is the year when highest movies have been released 👆

In [33]:

```
# Most popular movie in terms of revenue it has generated
# Approach 1


df[df['Revenue (Millions)'].max()== df['Revenue (Millions)']]['Title']
```

Out[33]:

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

In [34]:

```
# Most popular movie in terms of revenue it has generated
# Approach 2


df.sort_values(by='Revenue (Millions)' , ascending=False).head(1)['Title']
```

Out[34]:

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

Star Wars has generated the most revenue

In [77]:

```
# top 10 directors and movies with highest ratings

top10 = df.sort_values(by='Rating' , ascending=False).head(10)[['Title' , 'Director','Rat
top10
```

Out[77]:

	Title	Director	Rating
54	The Dark Knight	Christopher Nolan	9.0
80	Inception	Christopher Nolan	8.8
117	Dangal	Nitesh Tiwari	8.8
36	Interstellar	Christopher Nolan	8.6
96	Kimi no na wa	Makoto Shinkai	8.6
249	The Intouchables	Olivier Nakache	8.6
133	Whiplash	Damien Chazelle	8.5
64	The Prestige	Christopher Nolan	8.5
99	The Departed	Martin Scorsese	8.5
991	Taare Zameen Par	Aamir Khan	8.5

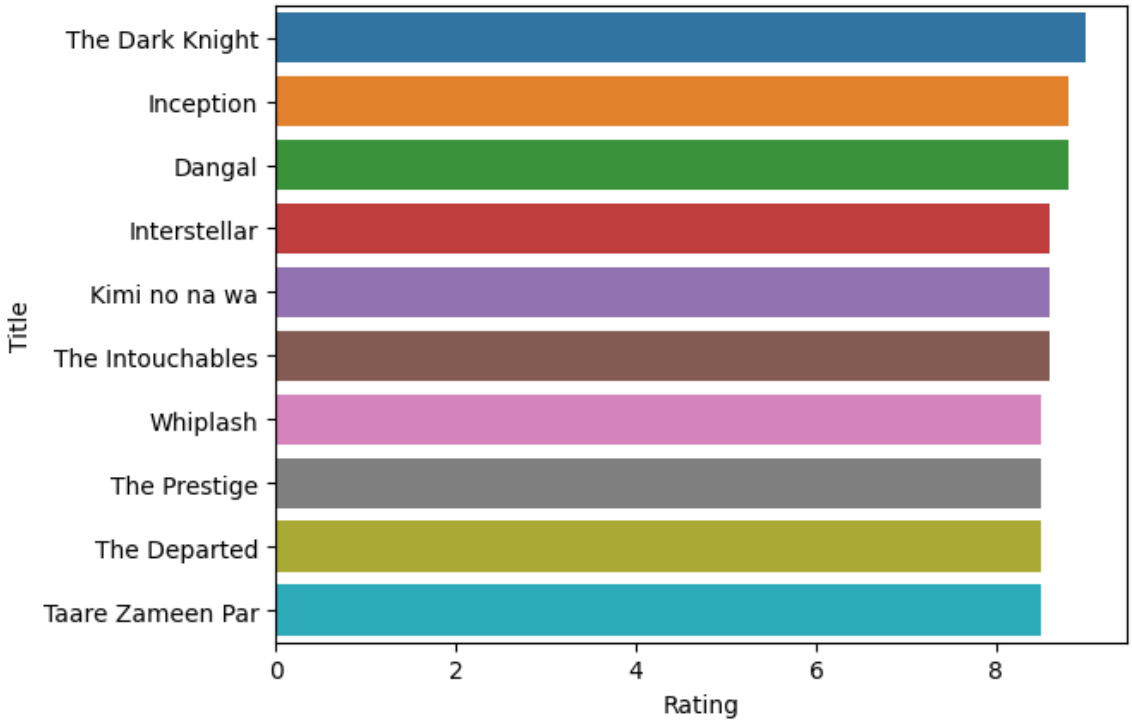
Above table is representing top-10 highest rated movies and its associated director

In [36]:

```
sns.barplot(y='Title', x='Rating', data=top10)
```

Out[36]:

```
<Axes: xlabel='Rating', ylabel='Title'>
```



Above graph is depicting top 10 highest rated movies where Dark Knight is most highest rated with 9.0 rating 👆

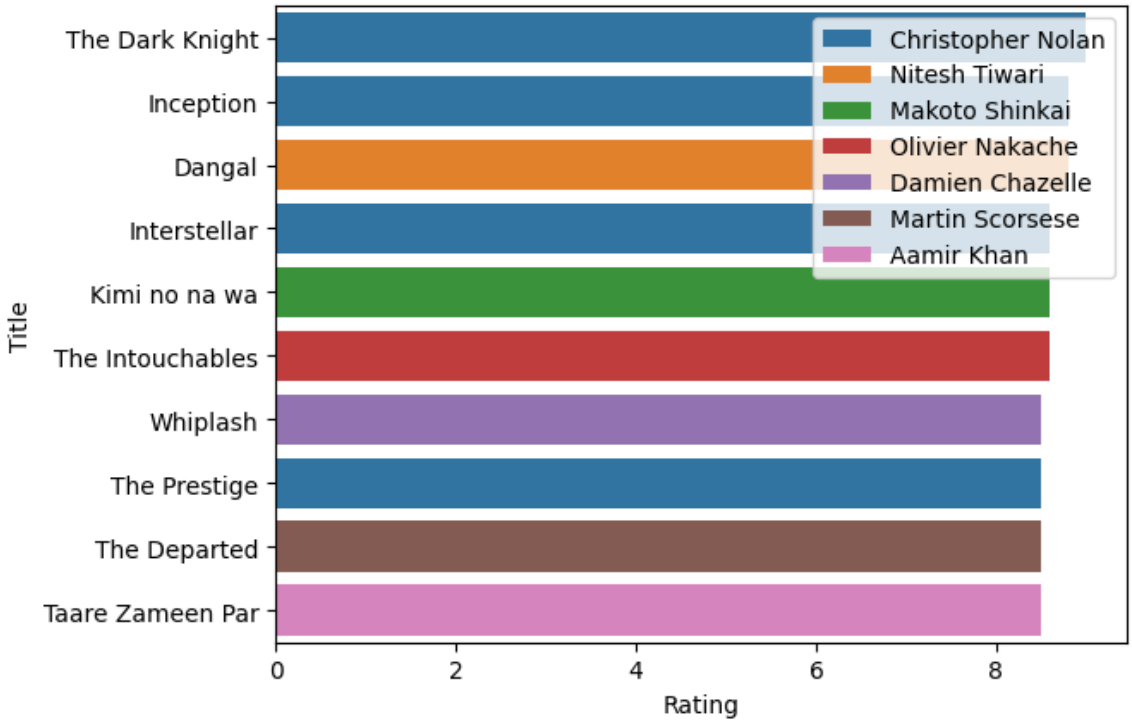
In [41]:

```
sns.barplot(y='Title', x='Rating', hue='Director', data=top10, dodge=False)

plt.legend(loc= 'upper right')
```

Out[41]:

```
<matplotlib.legend.Legend at 0x17764792770>
```



Above graph is depicting top 10 highest rated movies with its director showing 3 movies of Christopher Nolan are in top-5 of this list👆

In [78]:

```
# Top 10 highest revenue movie titles


top10 = df.sort_values(by='Revenue (Millions)', ascending=False)[['Revenue (Millions)', '
top10
```

Out[78]:

Title	Revenue (Millions)
Star Wars: Episode VII - The Force Awakens	936.63
Avatar	760.51
Jurassic World	652.18
The Avengers	623.28
The Dark Knight	533.32
Rogue One	532.17
Finding Dory	486.29
Avengers: Age of Ultron	458.99
The Dark Knight Rises	448.13
The Hunger Games: Catching Fire	424.65

Above Table is representing top-10 highest revenue generating movie and Star Wars has topped the list with 936.63 millions

In [79]:

```
# Top 10 highest revenue movie titles
# Approach2


df.nlargest(10,'Revenue (Millions)')['Title']
```

Out[79]:

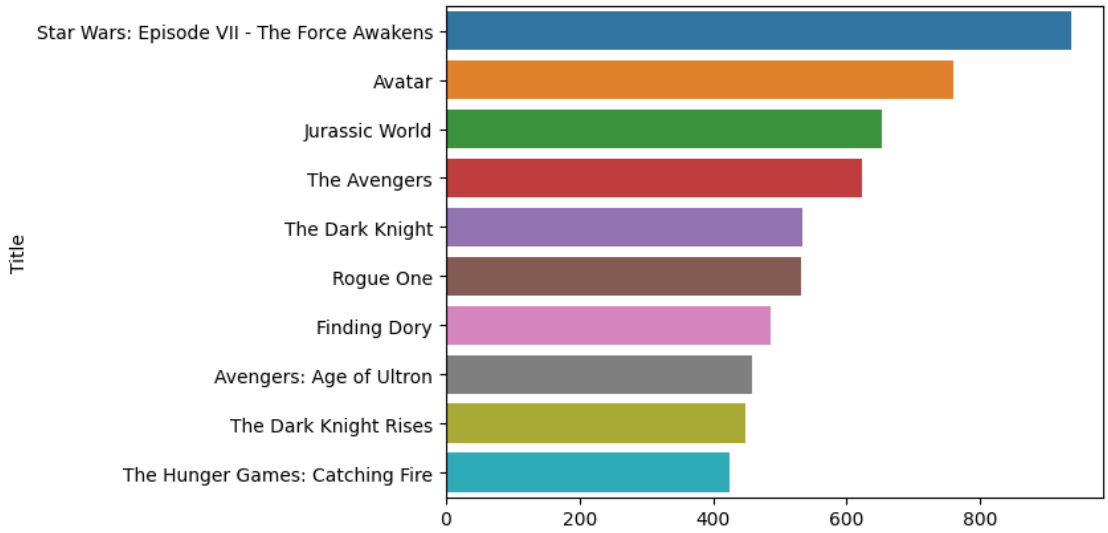
```
50     Star Wars: Episode VII - The Force Awakens
87                                         Avatar
85                                 Jurassic World
76                                   The Avengers
54                                 The Dark Knight
12                                       Rogue One
119                                  Finding Dory
94                        Avengers: Age of Ultron
124                         The Dark Knight Rises
578                The Hunger Games: Catching Fire
Name: Title, dtype: object
```

In [48]:

```
sns.barplot(x='Revenue (Millions)' , y=top10.index, data= top10)
```

Out[48]:

```
<Axes: xlabel='Revenue (Millions)', ylabel='Title'>
```



Above Bar Graph is visual representation of top-10 highest revenue generating movies and Star Wars has topped the list with 936.63 millions

In [80]:

```
# Average rating of movies year-wise


df.groupby('Year')['Rating'].mean().sort_values(ascending=False)
```

Out[80]:

```
Year
2007    7.133962
2006    7.125000
2009    6.960784
2012    6.925000
2011    6.838095
2014    6.837755
2010    6.826667
2013    6.812088
2008    6.784615
2015    6.602362
2016    6.436700
Name: Rating, dtype: float64
```

Above table is representing average rating of movies in each year and 2007 is the year with highet average rating of around 7.13

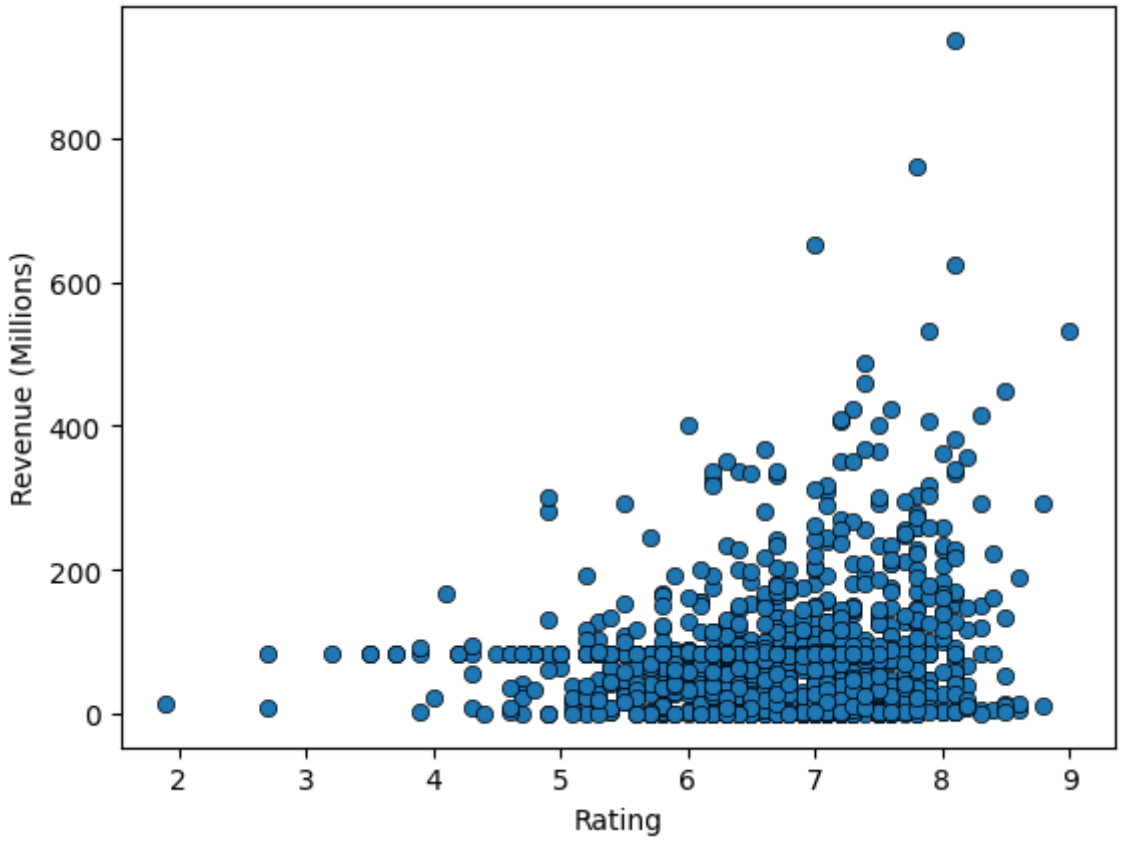
In [64]:

```
# Does rating affect revenue


sns.scatterplot(x='Rating', y='Revenue (Millions)', data=df, edgecolor='black')
```

Out[64]:

```
<Axes: xlabel='Rating', ylabel='Revenue (Millions)'>
```



Above plot is depicting the relationship between Rating & Revenue and it's visible that rating is severly affecting revenue

Categorise Ratings in Excellent, Good and Average

In [81]:

```
# Creating a function to categorise rating

def rating(rating) :
    if rating >= 7.0:
        return "Excellent"
    elif rating >= 6.0:
        return "Good"
    else:
        return "Average"
```

In [82]:

```
# Creating a new column containing the above categorisation function

df['Rating_ cat'] = df['Rating'].apply(rating)
```

In [83]:

```
# Representing the newly created rating categorisation column

df.head()
```

Out[83]:

	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(

Created a new column to categorise ratings into Excellent, Good, Average

In [72]:

```
# Count number of action movies
# Approach 1

df[df['Genre'].str.contains('Action',case=False)].count()
```

Out[72]:

```
Rank                  303
Title                 303
Genre                 303
Description           303
Director              303
Actors                303
Year                  303
Runtime (Minutes)     303
Rating                303
Votes                 303
Revenue (Millions)    303
Metascore             303
Rating_ cat           303
dtype: int64
```

Above analysis is representing no. of action movies being released and they are 303 in no.

In [73]:

```
# Count number of action movies
# Approach 2

len(df[df['Genre'].str.contains('Action',case=False)])
```

Out[73]:

```
303
```