Creating main dataset to standardize for the Final Jupyter Notebook

Needs to have ROI, Budgets, and Runntime

In [41]:

```
### Import the nessesary libraries
import pandas as pd
import sqlite3
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
import seaborn as sns
import scipy.stats as stats
```

In [42]:

```
df_num = pd.read_csv('data/tn.movie_budgets.csv'
```

In [43]:

```
conn = sqlite3.connect('data/im.db')
q = """
SELECT
    primary_title AS p_title,
    runtime_minutes AS time_min,
    averagerating AS avg_rating,
    genres
FROM movie_basics
JOIN movie_ratings
    USING(movie_id)
;
"""
df_IMDB = pd.read_sql(q, conn)
```

Combine the two into a single dataframe

In [44]:

```
### Merge on Primary Title
df = pd.merge(df_IMDB,
            df_num,
            how='inner',
            left_on = 'p_title',
            right_on = 'movie')
#df.info()
```

Cleaning the dataset

In [45]:

```
#Domestic
df['domestic_millions'] = df.domestic_gross.str.r
df['domestic_millions'] = df.domestic_millions.st
df['domestic_millions'] = pd.to_numeric(df.domest

#Worldwide
df['worldwide_millions'] = df.worldwide_gross.str
df['worldwide_millions'] = df.worldwide_millions
df['worldwide_millions'] = pd.to_numeric(df.worl

# Get rid of non-numerics in production_budget
df['budget_millions'] = df.production_budget.str
df['budget_millions'] = df.budget_millions.str.re
```

```
df['budget_millions'] = pd.to_numeric(df.budget_m


#df.info()
```

In [46]:
```
#Drop duplicates
df.drop_duplicates(subset=['movie'], inplace=Tru
# df.head(10)
```

In [47]:
```
#df.info()
```

Adding in the correct roi

In [48]:
```
df['roi'] = (df.worldwide_millions - df.budget_m
```

Add in the binning for the production budget

In [49]:
```
def binning(x):
    if x > 216:
        return 216
    elif x > 108:
        return 108
    elif x > 64:
        return 64
    elif x > 32:
        return 32
    elif x > 16:
        return 16
    elif x > 8:
        return 8
    elif x > 4:
        return 4
    elif x > 2:
        return 2
    else:
        return 1
```

In [50]:
```
x = [binning(x) for x in df['budget_millions']]
```

In [51]:
```
pd.options.mode.chained_assignment = None
df['budget_binning'] = x
```

In [52]:
```
df.head(10)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2126 entries, 0 to 2874
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   p_title             2126 non-null   object
 1   time_min            2072 non-null   float64
 2   avg_rating          2126 non-null   float64
 3   genres              2124 non-null   object
 4   id                  2126 non-null   int64
 5   release_date        2126 non-null   object
 6   movie               2126 non-null   object
```

```
 6   movie               2126 non-null   object
 7   production_budget   2126 non-null   object
 8   domestic_gross      2126 non-null   object
 9   worldwide_gross     2126 non-null   object
 10  domestic_millions   2126 non-null   float64
 11  worldwide_millions  2126 non-null   float64
 12  budget_millions     2126 non-null   float64
 13  roi                 2126 non-null   float64
 14  budget_binning      2126 non-null   int64
dtypes: float64(6), int64(2), object(7)
memory usage: 265.8+ KB
```

Genre Z-Test Analysis

In [53]:

```
df = df.dropna()
myList = df['genres'].tolist()
genre_list=[]
for i in range(len(myList)):
    genre_list.append(myList[i].split(','))
#genre_list
```

In [54]:

```
from itertools import chain
genres=list(set(chain(*genre_list)))
#print(genres)
#print(len(genres))
#print(type(genres))
```

In [55]:

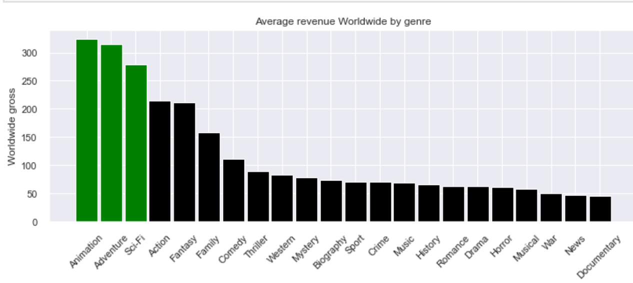
```
#Worldwide gross Average profits.

wwg_avg=[]
for i in range(len(genres)):
    wwg = df['genres'].astype(str).str.contains(
    wwg_avg.append(df['worldwide_millions'][wwg]

wwg_avg, genres = zip(*sorted(zip(wwg_avg, genre
c = range(len(wwg_avg))

#creating the bar plot
clrs = ['black' if (x < wwg_avg[2]) else 'green'
fig = plt.figure(figsize = (12, 4))
ax = fig.add_subplot(111)

ax.bar(c, wwg_avg, color=clrs ,width = 0.9)
plt.xticks(c, genres, rotation=45)
plt.xlabel("Genres")
plt.ylabel("Worldwide gross")
plt.title("Average revenue Worldwide by genre")
plt.show()
```



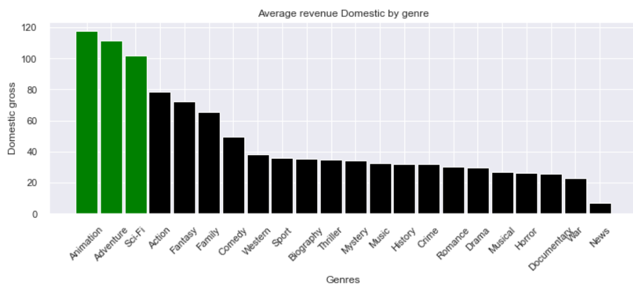
In [56]:

```
#Domestic gross Average profits.
```

```
dg_avg=[]
for i in range(len(genres)):
    dg = df['genres'].astype(str).str.contains(g
    dg_avg.append(df['domestic_millions'][dg].mea
#dg_avg
dg_avg, genres = zip(*sorted(zip(dg_avg, genres)
c = range(len(dg_avg))

#creating the bar plot
clrs = ['black' if (x < dg_avg[2]) else 'green' 
fig = plt.figure(figsize = (12, 4))
ax = fig.add_subplot(111)

ax.bar(c, dg_avg, color=clrs ,width = 0.9)
plt.xticks(c, genres, rotation=45)
plt.xlabel("Genres")
plt.ylabel("Domestic gross")
plt.title("Average revenue Domestic by genre")
plt.show()
```



In [57]:

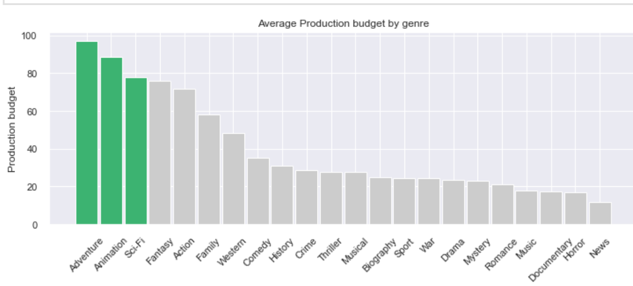
```
#production_budget Average profits.

pb_avg=[]
for i in range(len(genres)):
    pb = df['genres'].astype(str).str.contains(g
    pb_avg.append(df['budget_millions'][pb].mean
#print(pb_avg)

pb_avg, genres = zip(*sorted(zip(pb_avg, genres)
c = range(len(pb_avg))

#creating the bar plot
clrs = ['0.8' if (x < pb_avg[2]) else 'mediumsea
fig = plt.figure(figsize = (12, 4))
ax = fig.add_subplot(111)

ax.bar(c, pb_avg, color=clrs ,width = 0.9)
plt.xticks(c, genres, rotation=45)
plt.xlabel("Genres")
plt.ylabel("Production budget")
plt.title("Average Production budget by genre")
plt.show()
```



It seems that the film genres that generate the most money worldwide and by country of origin are Animation, Adventure, Science Fiction, Fantasy and Action, respectively in that order, remaining as the top 5.

On the other hand we have that the most expensive films to produce are precisely Adventure, Animation, Fantasy, Sci-Fi and Action in that order, given the equipment and technology that is needed to create these films such as visual effects, computers or tools and labor in order to Make the movie.

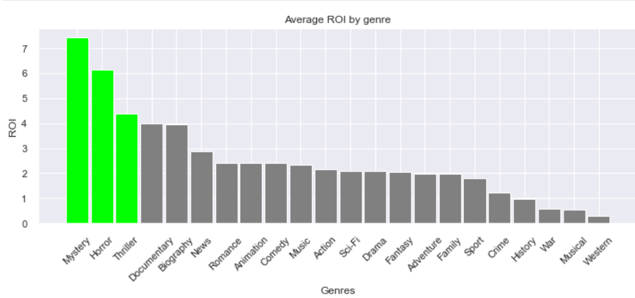
In [58]:

```
#Roi Average profits.

roi_avg=[]
for i in range(len(genres)):
    roi = df['genres'].astype(str).str.contains(
    roi_avg.append(df['roi'][roi].mean())
#print(roi_avg)
roi_avg, genres = zip(*sorted(zip(roi_avg, genre
c = range(len(roi_avg))

#creating the bar plot
clrs = ['grey' if (x < wwg_avg[2]) else 'lime' f
fig = plt.figure(figsize = (12, 4))
ax = fig.add_subplot(111)

ax.bar(c, roi_avg, color=clrs ,width = 0.9)
plt.xticks(c, genres, rotation=45)
plt.xlabel("Genres")
plt.ylabel("ROI")
plt.title("Average ROI by genre")
plt.show()
```



Regarding the ROI we can clearly see that the genres that stands out the most are Horror and Mystery especially, then Thriller, Documentary and Biography in that order.

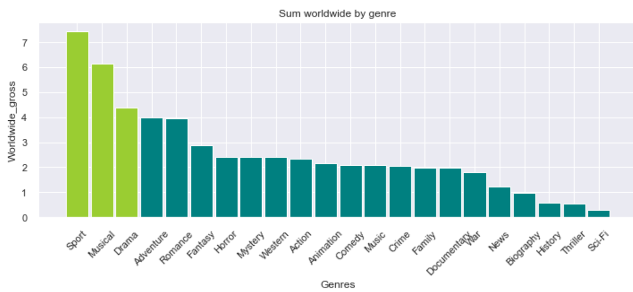
In [59]:

```
#Worldwide gross sum profits.

wwg_sum=[]
for i in range(len(genres)):
    Gsum = df['genres'].astype(str).str.contains
    wwg_sum.append(df['worldwide_gross'][Gsum].s
#wwg_sum
wwg_sum, genres = zip(*sorted(zip(wwg_sum, genre
c = range(len(roi_avg))
```

```
#creating the bar plot
clrs = ['#008080' if (x < wwg_sum[2]) else '#9ACI
fig = plt.figure(figsize = (12, 4))
ax = fig.add_subplot(111)

ax.bar(c, roi_avg, color=clrs ,width = 0.9)
plt.xticks(c, genres, rotation=45)
plt.xlabel("Genres")
plt.ylabel("Worldwide_gross")
plt.title("Sum worldwide by genre")
plt.show()
```



Hypothesis Testing

Our team wanted to see if Mystery, Horror and Thriller movies generete a larger ROI than other genres.

State our Hypotheses:

H-Alt -> The average ROI for Mystery, Horror and Thriller is higher than other movies

$$H_a\colon \mu < \bar{x}$$

H-Null -> There is no difference in the ROI for Mystery, , Horror and Thriller vs. other movies

$$H_o\colon \mu \geq \bar{x}$$

Alpha: 0.05

In [60]:
```
df.describe()
```

Out[60]:

	time_min	avg_rating	id	domestic_mi
count	2070.000000	2070.000000	2070.000000	2070.0
mean	103.908213	6.228068	50.533333	46.7
std	18.747006	1.130268	28.547208	78.0
min	5.000000	1.600000	1.000000	0.0
25%	91.000000	5.600000	26.000000	0.6
50%	102.000000	6.300000	50.000000	18.9
75%	114.000000	7.000000	75.000000	56.4
max	180.000000	9.200000	100.000000	760.5

In [61]:

```
#population mean and std

roi_mean = df['roi'].mean()
roi_std = df['roi'].std()

stats.norm(roi_mean, roi_std)
plt.style.use('seaborn')
```

In [62]:

```
#Mystery mean
df_mystery = df[df['genres'].str.contains('Myste
mystery_mean = df_mystery['roi'].mean()
#df_mystery
#print(mystery_mean)

#Horror mean
df_horror = df[df['genres'].str.contains('Horror
horror_mean = df_horror['roi'].mean()
#df_mystery
#print(mystery_mean)

#Thriller mean
df_thriller = df[df['genres'].str.contains('Thri
thriller_mean = df_thriller['roi'].mean()
#df_mystery
#print(mystery_mean)

#Documentary mean
df_Documentary = df[df['genres'].str.contains('D
Documentary_mean = df_Documentary['roi'].mean()
#df_mystery
#print(mystery_mean)
```

In [63]:

```
# Z score Function
from math import sqrt
def Zscore(Dataframe, mean):
    x_bar= mean
    n =  len(Dataframe.id)
    sigma = roi_std
    mu = roi_mean
    z = (x_bar - mu)/(sigma/sqrt(n))

    pval= 1-stats.norm.cdf(z)
    print(f'P-value: {pval}')
    #print(pval<0.05)
    print(f'Percent area under the curve from Zs


Zscore(df_mystery, mystery_mean)
Zscore(df_horror, horror_mean)
Zscore(df_thriller, thriller_mean)
Zscore(df_Documentary, Documentary_mean)
```

```
P-value: 4.619940964678548e-06
Percent area under the curve from Zscore of 4.434
237277030223 is 99.99953800590353%

P-value: 2.0679048268856803e-05
Percent area under the curve from Zscore of 4.099
758791095601 is 99.99793209517311%

P-value: 0.008758839449245048
```

```
Percent area under the curve from Zscore of 2.375
6582508012034 is 99.12411605507549%

P-value: 0.1475443516480548
Percent area under the curve from Zscore of 1.047
023579203531 is 85.24556483519451%
```

Due to our p- value < 0.05 we know that there is enough evidence to reject the null hypothesis with the given sample in other words ROI average for Mystery, Horror and Thriller films are significantly higher than the population.

Law of Diminishing Returns Analysis

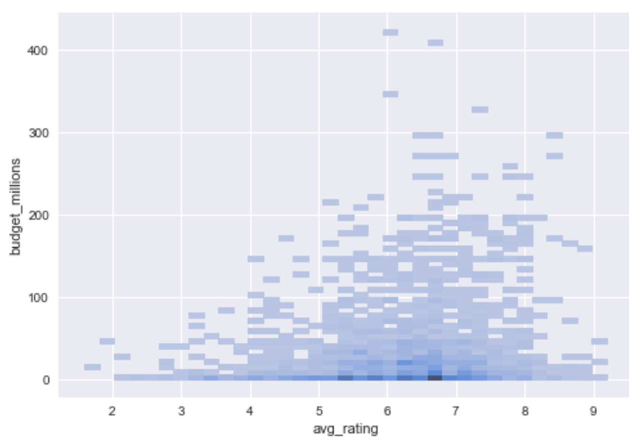
The objective with this analysis is to visualize the law of dimishing returns comparing production budget (our group will assume that production budget is cost) to the overall rating and ROI of the film.

Starting with Production budget vs. Overall Rating

In [64]:

```
#First a simple viuslization
sns.histplot(x=df.avg_rating,
             y=df.budget_millions)
```

Out[64]: <AxesSubplot:xlabel='avg_rating', ylabel='budget_millions'>



From this visual we can see that the average ratings are concentrated around the mean, as well as there are many outliers in budget upward of 200 Million. Using the binning already added into the dataframe, we can bin the production budgets for a visualization showing the increase in budget vs the average rating. This will demonstrate the rate and variance of ratings as production budget (Movie Costs) increase.

In [65]:

```
# Use matplotlib to set the figure size and shape
```

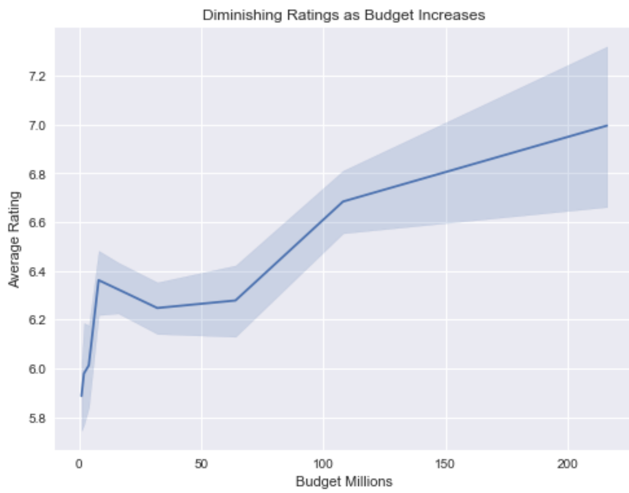
```
fig, ax = plt.subplots(figsize=(8,6))

#Create a lineplot of both using seaborn
ax = sns.lineplot(df.budget_binning, df.avg_ratii
ax.set(xlabel='Budget Millions', ylabel='Average
# ax.axhline(df.avg_rating.mean(), color='red')
```

```
C:\Users\psoloriocabrera\Anaconda3\envs\learn-env
\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variables as keywo
rd args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing o
ther arguments without an explicit keyword will r
esult in an error or misinterpretation.
  warnings.warn(
```

Out[65]:

```
[Text(0.5, 0, 'Budget Millions'),
 Text(0, 0.5, 'Average Rating'),
 Text(0.5, 1.0, 'Diminishing Ratings as Budget In
creases')]
```



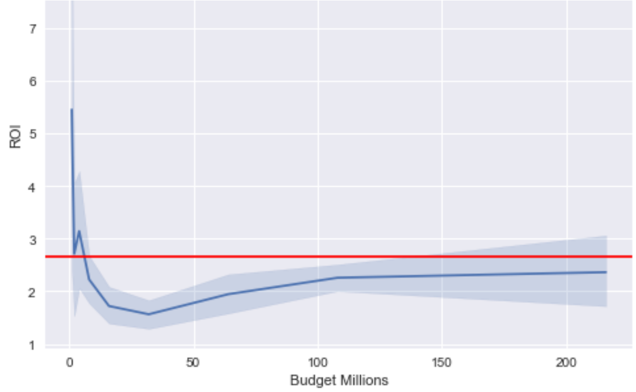
Based on the plot above, we can see that after budgets start to excede 100 million, returns start to diminish as well as become more volitile. It shows that ratings will tend to show less improvement in ratings as the production budget increases. Now let's try the same thing but for ROI.

In [66]:

```
fig, ax = plt.subplots(figsize=(8,6))
ax = sns.lineplot(df.budget_binning, df.roi)
ax.set(xlabel='Budget Millions', ylabel='ROI', t:
ax.axhline(df.roi.mean(), color='red')
```

```
C:\Users\psoloriocabrera\Anaconda3\envs\learn-env
\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variables as keywo
rd args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing o
ther arguments without an explicit keyword will r
esult in an error or misinterpretation.
  warnings.warn(
```

Out[66]: <matplotlib.lines.Line2D at 0x1fd49868d00>



Based on the graph above, there seem to be a large amount of outliers on the upper end of ROI skewing our data. To get rid of that we're going to make a normal distribution using the ROI data points, and keep the lower 99% of those values as to not deal with large and unlikely ROI's.

In [67]:

```
dist = stats.norm(df.roi.mean(),df.roi.std())
dist.ppf(.99)
#This now identifies the new cutoff for ROI value
#We can use this to drop rows with ROI's greater
```

Out[67]: 34.73519995681795

In [68]:

```
#Drop ROI's where the roi is > 3
df_dropped34 = df.drop(df[df['roi'] > 34].index,
df_dropped34.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2053 entries, 0 to 2873
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   p_title             2053 non-null   object 
 1   time_min            2053 non-null   float64
 2   avg_rating          2053 non-null   float64
 3   genres              2053 non-null   object 
 4   id                  2053 non-null   int64  
 5   release_date        2053 non-null   object 
 6   movie               2053 non-null   object 
 7   production_budget   2053 non-null   object 
 8   domestic_gross      2053 non-null   object 
 9   worldwide_gross     2053 non-null   object 
 10  domestic_millions   2053 non-null   float64
 11  worldwide_millions  2053 non-null   float64
 12  budget_millions     2053 non-null   float64
 13  roi                 2053 non-null   float64
 14  budget_binning      2053 non-null   int64  
dtypes: float64(6), int64(2), object(7)
memory usage: 256.6+ KB
```

Let's run the graph again, this time without the large outliers on ROI.

In [69]:

```
fig, ax = plt.subplots(figsize=(8,6))
ax = sns.lineplot(df_dropped34.budget_binning, d
ax.set(xlabel='Budget Millions', ylabel='ROI', t
```

```
ax.axhline(df_dropped34.roi.mean(), color='red')
```

```
C:\Users\psoloriocabrera\Anaconda3\envs\learn-env
\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variables as keywo
rd args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing o
ther arguments without an explicit keyword will r
esult in an error or misinterpretation.
  warnings.warn(
```

Out[69]: `<matplotlib.lines.Line2D at 0x1fd49bb8610>`

From the graph above, we can now see that returns on investment virtually flatline after 100 million in production budget, with an increase in volatility. From both average ROI and average rating, we can say that returns diminish substantially after an investment of 100 million.

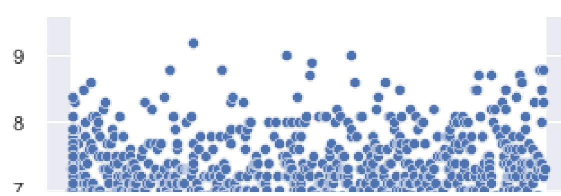
Rating to Worldwide Revenue Assessment

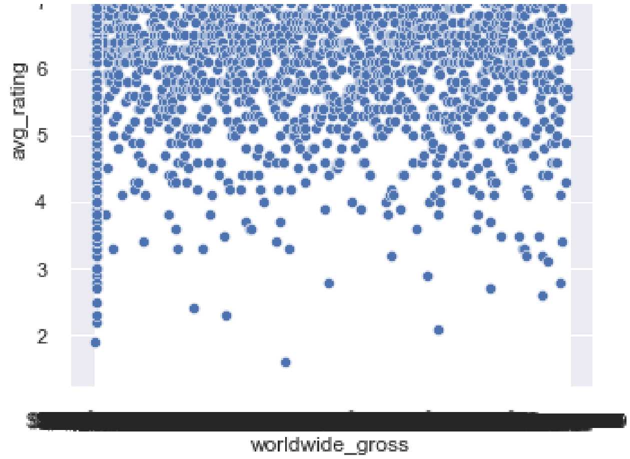
This is the first draft of a scatter plot comparing the average ratings to the worldwide gross margin of each movie

In [31]:

```
## First Draft
sns.set_theme()
sns.relplot(
            y=df.avg_rating,
            x=df.worldwide_gross
            )
```

Out[31]: `<seaborn.axisgrid.FacetGrid at 0x1fd42f4ad00>`



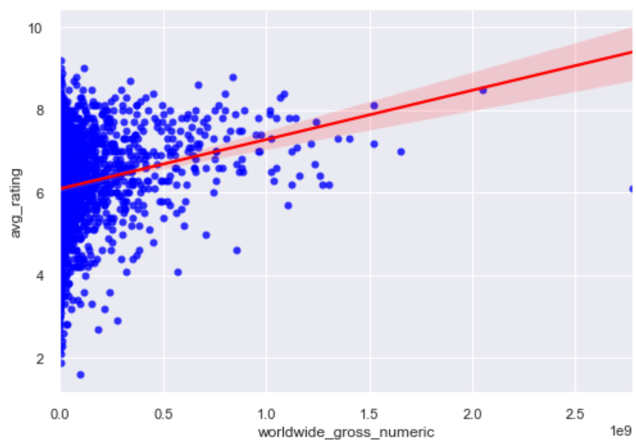


We needed to determine the line of best fit for the data and plot it accordingly

In [32]:

```
## Regression Data
df['worldwide_gross_numeric'] = pd.to_numeric(df
sns.regplot(
    y=df['avg_rating'],
    x=df['worldwide_gross_numeric'],
    scatter_kws={"color": "blue"},
    line_kws={"color": "red"}
)
```

Out[32]: <AxesSubplot:xlabel='worldwide_gross_numeric', ylabel='avg_rating'>

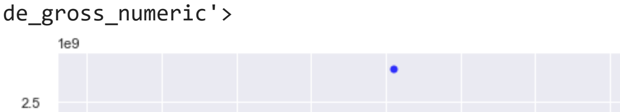


In order to remain consistent we need to flip the axes

In [33]:

```
## Regression Data
df['worldwide_gross_numeric'] = pd.to_numeric(df
sns.regplot(
    y=df['worldwide_gross_numeric'],
    x=df['avg_rating'],
    scatter_kws={"color": "blue"},
    line_kws={"color": "red"},
)
```

Out[33]: <AxesSubplot:xlabel='avg_rating', ylabel='worldwide_gross_numeric'>



In conclusion, Ratings are valuable and Computing Vision should target above a 6 rating to ensure the best possible results. We can also clearly see there are diminishing returns after about a 7.75-8 rating where there are less positive outliers. This shows that it is extremely valuable to reach a certain movie standard but not worth crazy investments of time and money to get above an 8.

Ideal Runtime

In [34]:

```
df_rating = df.drop(df[df['avg_rating'] < 7.1].i
df_rating.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 495 entries, 6 to 2872
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dty
pe
---  ------              --------------  ---
--
 0   p_title             495 non-null    obj
ect
 1   time_min            495 non-null    flo
at64
 2   avg_rating          495 non-null    flo
at64
 3   genres              495 non-null    obj
ect
 4   id                  495 non-null    int
64
 5   release_date        495 non-null    obj
ect
 6   movie               495 non-null    obj
ect
 7   production_budget   495 non-null    obj
ect
 8   domestic_gross      495 non-null    obj
ect
 9   worldwide_gross     495 non-null    obj
ect
 10  domestic_millions   495 non-null    flo
at64
 11  worldwide_millions  495 non-null    flo
at64
 12  budget_millions     495 non-null    flo
at64
 13  roi                 495 non-null    flo
```

```
 13  roi                     495 non-null    flo
at64
 14  budget_binning          495 non-null    int
64
 15  worldwide_gross_numeric  495 non-null    int
64
dtypes: float64(6), int64(3), object(7)
memory usage: 65.7+ KB
```

Created a new dataframe that drops all the movies with a rating less than 7.1.

In [35]:

```
df_rating.describe()
```

Out[35]:

	time_min	avg_rating	id	domestic_millio
count	495.000000	495.000000	495.000000	495.0000
mean	110.561616	7.555354	48.286869	74.7739
std	23.784281	0.419715	28.145345	108.9161
min	5.000000	7.100000	1.000000	0.0000
25%	96.000000	7.200000	25.000000	4.5007
50%	109.000000	7.400000	46.000000	33.3954
75%	126.500000	7.800000	72.000000	99.7106
max	180.000000	9.200000	100.000000	700.0595

Describing the new data frame created.

In [36]:

```
fig, ax = plt.subplots(figsize=(8,6))
ax = sns.histplot(df_rating.time_min, kde = True
ax.set(xlabel='Runtime in Minutes', ylabel=' Cou
```

Out[36]:

```
[Text(0.5, 0, 'Runtime in Minutes'),
 Text(0, 0.5, ' Count of Movies Above Average Rat
ing'),
 Text(0.5, 1.0, 'Movie Rating to Movie Runtime')]
```