

EDA_MOVIE_RATINGS_ADVANCE_VISUALIZA



Final discussion what we learn so far

- 1> category datatype in python
- 2> jointplots
- 3> histogram
- 4> stacked histograms
- 5> Kde plot
- 6> subplot
- 7> violin plots
- 8> Facet grid
- 9> Building dashboards

```
In [1]: import pandas as pd  
import os
```

```
In [2]: os.getcwd() # if you want to change the working directory
```

```
Out[2]: 'C:\\\\Users\\\\Hp\\\\Desktop\\\\7-30 BATCH\\\\DATA SCIENCE\\\\TASKS_PROJECTS'
```

```
In [3]: movies = pd.read_csv(r"C:\\Users\\Hp\\Desktop\\7-30 BATCH\\DATA SCIENCE\\CSV_FILES\\Movie-
```

```
In [4]: movies
```

Out[4]:

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million \$)	Year of release
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009
...
554	Your Highness	Comedy	26	36	50	2011
555	Youth in Revolt	Comedy	68	52	18	2009
556	Zodiac	Thriller	89	73	65	2007
557	Zombieland	Action	90	87	24	2009
558	Zookeeper	Comedy	14	42	80	2011

559 rows × 6 columns

In [5]: `len(movies)`

Out[5]: 559

In [6]: `movies.shape`

Out[6]: (559, 6)

In [7]: `movies.head()`

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million \$)	Year of release
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009

In [8]: `movies.tail()`

```
Out[8]:
```

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million \$)	Year of release
554	Your Highness	Comedy	26	36	50	2011
555	Youth in Revolt	Comedy	68	52	18	2009
556	Zodiac	Thriller	89	73	65	2007
557	Zombieland	Action	90	87	24	2009
558	Zookeeper	Comedy	14	42	80	2011

```
In [9]: movies.columns
```

```
Out[9]: Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',  
               'Budget (million $)', 'Year of release'],  
              dtype='object')
```

```
In [10]: movies.columns = ['Film', 'Genre', 'CriticRating', 'AudienceRating', 'BudgetMillions']
```

```
In [11]: movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 559 entries, 0 to 558  
Data columns (total 6 columns):  
 #   Column           Non-Null Count  Dtype     
---  --    
 0   Film             559 non-null    object    
 1   Genre            559 non-null    object    
 2   CriticRating     559 non-null    int64    
 3   AudienceRating   559 non-null    int64    
 4   BudgetMillions  559 non-null    int64    
 5   Year             559 non-null    int64    
dtypes: int64(4), object(2)  
memory usage: 26.3+ KB
```

```
In [12]: movies.describe()
```

```
# if you look at the year the data type is int but when you look at the mean value
```

Out[12]:

	CriticRating	AudienceRating	BudgetMillions	Year
count	559.000000	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136	2009.152057
std	26.413091	16.826887	48.731817	1.362632
min	0.000000	0.000000	0.000000	2007.000000
25%	25.000000	47.000000	20.000000	2008.000000
50%	46.000000	58.000000	35.000000	2009.000000
75%	70.000000	72.000000	65.000000	2010.000000
max	97.000000	96.000000	300.000000	2011.000000

we have to change to category type from object datatype we will convert to category datatypes

In [13]:

```
movies['Film']
# movies['Audience Ratings %']
```

Out[13]:

```
0      (500) Days of Summer
1                  10,000 B.C.
2                  12 Rounds
3                  127 Hours
4                  17 Again
...
554             Your Highness
555        Youth in Revolt
556            Zodiac
557       Zombieland
558        Zookeeper
Name: Film, Length: 559, dtype: object
```

In [14]:

```
movies.Film
```

Out[14]:

```
0      (500) Days of Summer
1                  10,000 B.C.
2                  12 Rounds
3                  127 Hours
4                  17 Again
...
554             Your Highness
555        Youth in Revolt
556            Zodiac
557       Zombieland
558        Zookeeper
Name: Film, Length: 559, dtype: object
```

In [15]:

```
movies.Film = movies.Film.astype('category')
movies.Film
```

```
Out[15]: 0      (500) Days of Summer
         1          10,000 B.C.
         2          12 Rounds
         3          127 Hours
         4          17 Again
         ...
        554      Your Highness
        555      Youth in Revolt
        556          Zodiac
        557      Zombieland
        558      Zookeeper
Name: Film, Length: 559, dtype: category
Categories (559, object): ['(500) Days of Summer ', '10,000 B.C.', '12 Rounds ',
 '127 Hours', ..., 'Youth in Revolt', 'Zodiac', 'Zombieland ', 'Zookeeper']
```

```
In [16]: # now the same thing we will change genra to category & year to category
```

```
In [17]: movies.Genre = movies.Genre.astype('category')
movies.Genre
```

```
Out[17]: 0      Comedy
         1      Adventure
         2      Action
         3      Adventure
         4      Comedy
         ...
        554      Comedy
        555      Comedy
        556      Thriller
        557      Action
        558      Comedy
Name: Genre, Length: 559, dtype: category
Categories (7, object): ['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Roma
nce', 'Thriller']
```

```
In [18]: movies.Year = movies.Year.astype('category')
movies.Year # is it real no. year you can take average,min,max but out come have n
```

```
Out[18]: 0      2009
         1      2008
         2      2009
         3      2010
         4      2009
         ...
        554      2011
        555      2009
        556      2007
        557      2009
        558      2011
Name: Year, Length: 559, dtype: category
Categories (5, int64): [2007, 2008, 2009, 2010, 2011]
```

```
In [19]: movies.info() # now Film, Genre, and Year has converted to category
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Film              559 non-null    category
 1   Genre             559 non-null    category
 2   CriticRating     559 non-null    int64  
 3   AudienceRating   559 non-null    int64  
 4   BudgetMillions  559 non-null    int64  
 5   Year              559 non-null    category
dtypes: category(3), int64(3)
memory usage: 36.5 KB
```

```
In [20]: movies.describe()
#now when you see the describt you will get only integer value mean, standard devia
```

```
Out[20]:
```

	CriticRating	AudienceRating	BudgetMillions
count	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136
std	26.413091	16.826887	48.731817
min	0.000000	0.000000	0.000000
25%	25.000000	47.000000	20.000000
50%	46.000000	58.000000	35.000000
75%	70.000000	72.000000	65.000000
max	97.000000	96.000000	300.000000

```
In [21]: movies.Genre.cat.categories # .cat.categories will give you the unique categories
```

```
Out[21]: Index(['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance',
               'Thriller'],
               dtype='object')
```

```
In [22]: # How to working with joint plots
from matplotlib import pyplot as plt
import seaborn as sns
```

```
In [23]: %matplotlib inline
```

```
In [24]: import warnings
warnings.filterwarnings('ignore')
```

```
In [25]: movies
```

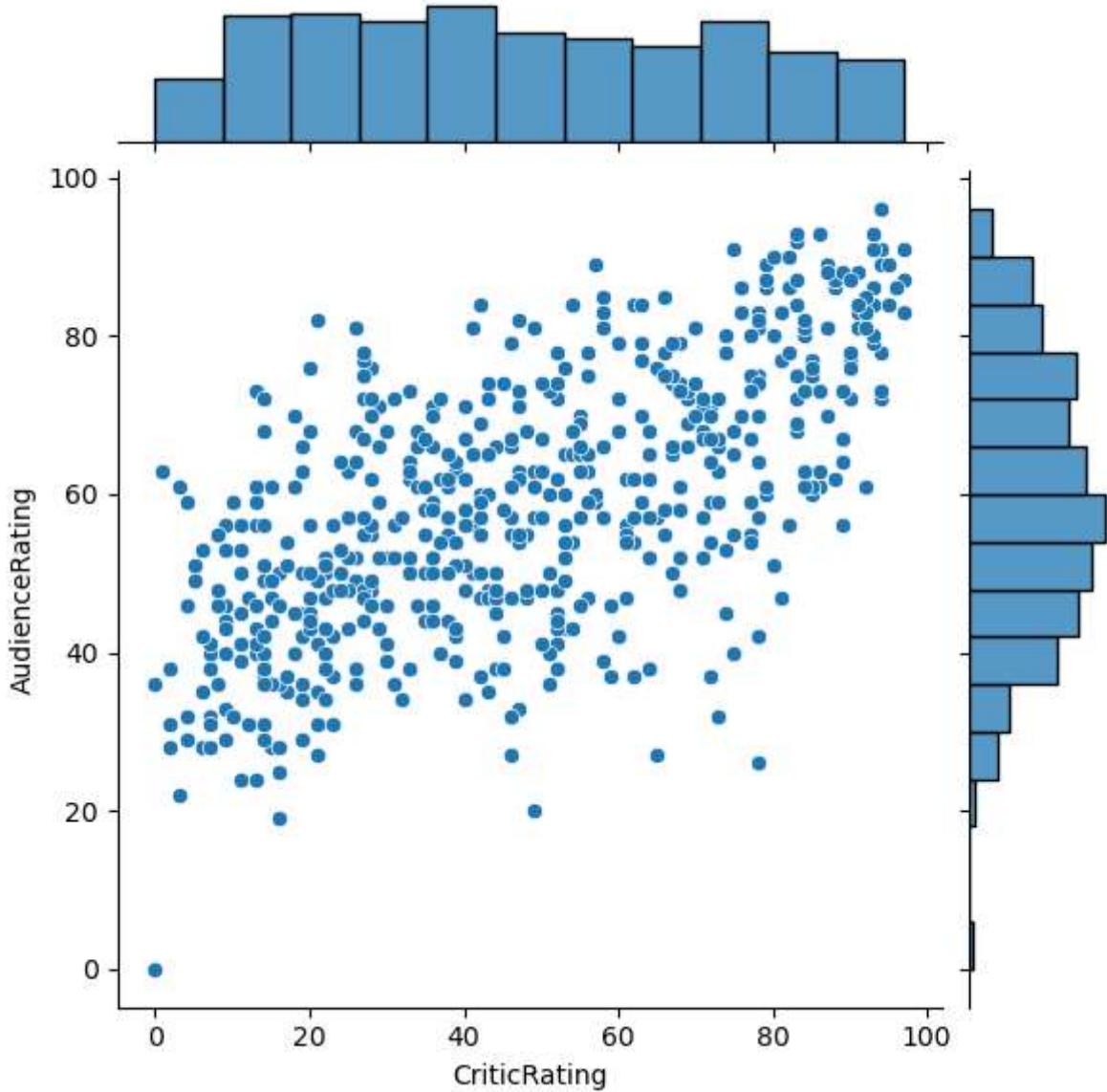
Out[25]:

	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009
...
554	Your Highness	Comedy	26	36	50	2011
555	Youth in Revolt	Comedy	68	52	18	2009
556	Zodiac	Thriller	89	73	65	2007
557	Zombieland	Action	90	87	24	2009
558	Zookeeper	Comedy	14	42	80	2011

559 rows × 6 columns

- basically joint plot is a scatter plot & it find the relation b/w audiene & critics
- also if you look up you can find the uniform distribution (critics)and normal distriution (audience)

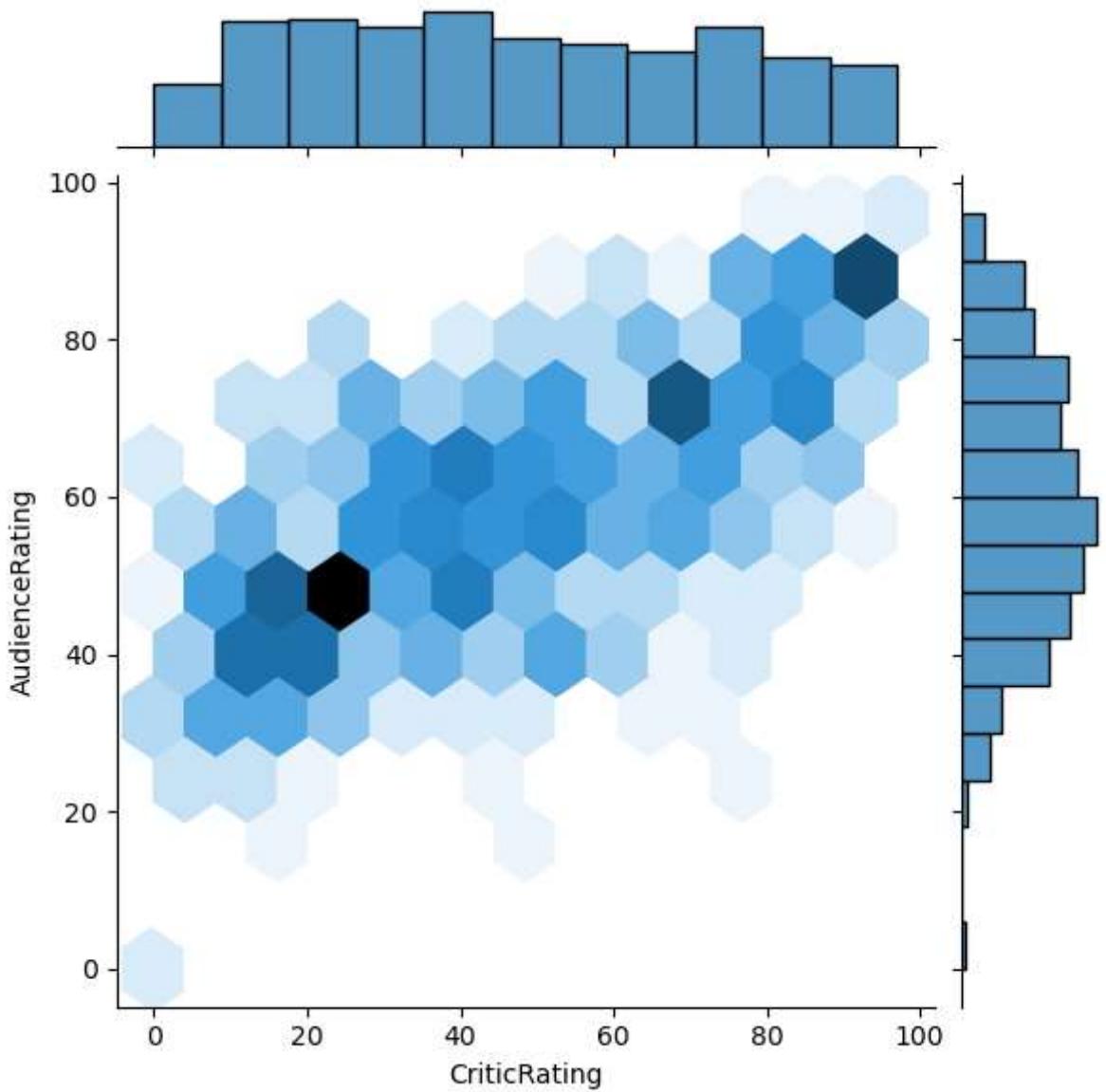
In [26]: `j = sns.jointplot(data = movies, x = 'CriticRating', y = 'AudienceRating')`



Interpretation

- Moderate Positive Correlation
- Audience rating is more dominant than critics rating
- Based on this we find out as most people are most likely to watch audience rating & less likely to watch critics rating
- Let me explain the excel - if you filter audience rating & critic rating. critic rating has very low values compare to audience rating

```
In [27]: j = sns.jointplot(data= movies, x = 'CriticRating', y = 'AudienceRating', kind='hex')
```



In [28]: movies

Out[28]:

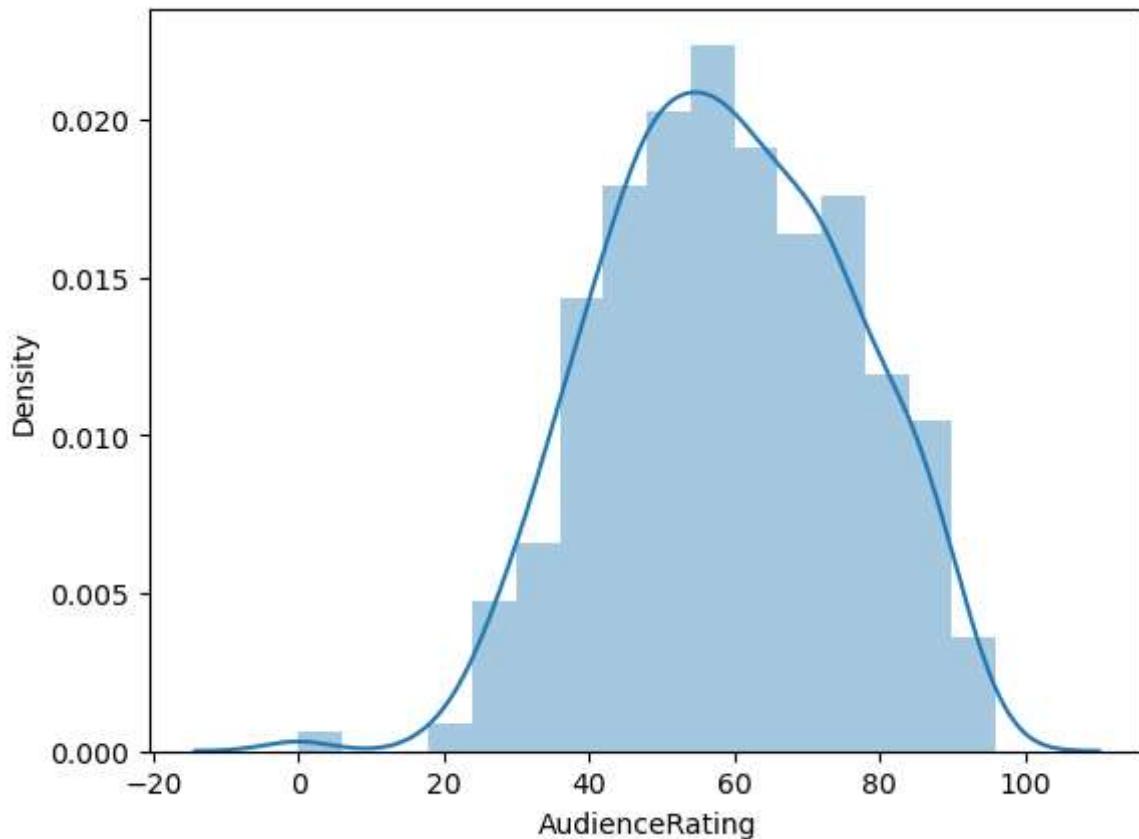
	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009
...
554	Your Highness	Comedy	26	36	50	2011
555	Youth in Revolt	Comedy	68	52	18	2009
556	Zodiac	Thriller	89	73	65	2007
557	Zombieland	Action	90	87	24	2009
558	Zookeeper	Comedy	14	42	80	2011

559 rows × 6 columns

Plot distplot in Seaborn

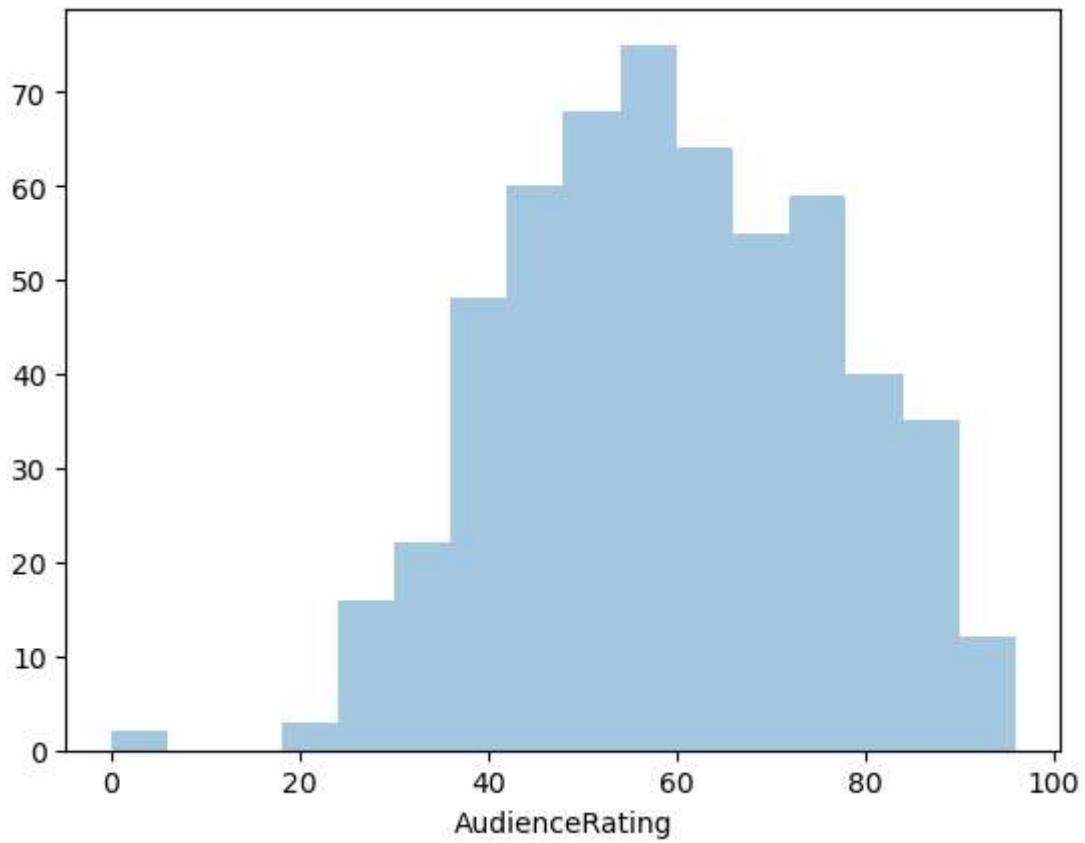
In [29]:

```
# Histograms
m1 = sns.distplot(movies.AudienceRating)
#y - axis generated by seaborn automatically that is the powefull of seaborn galler
```

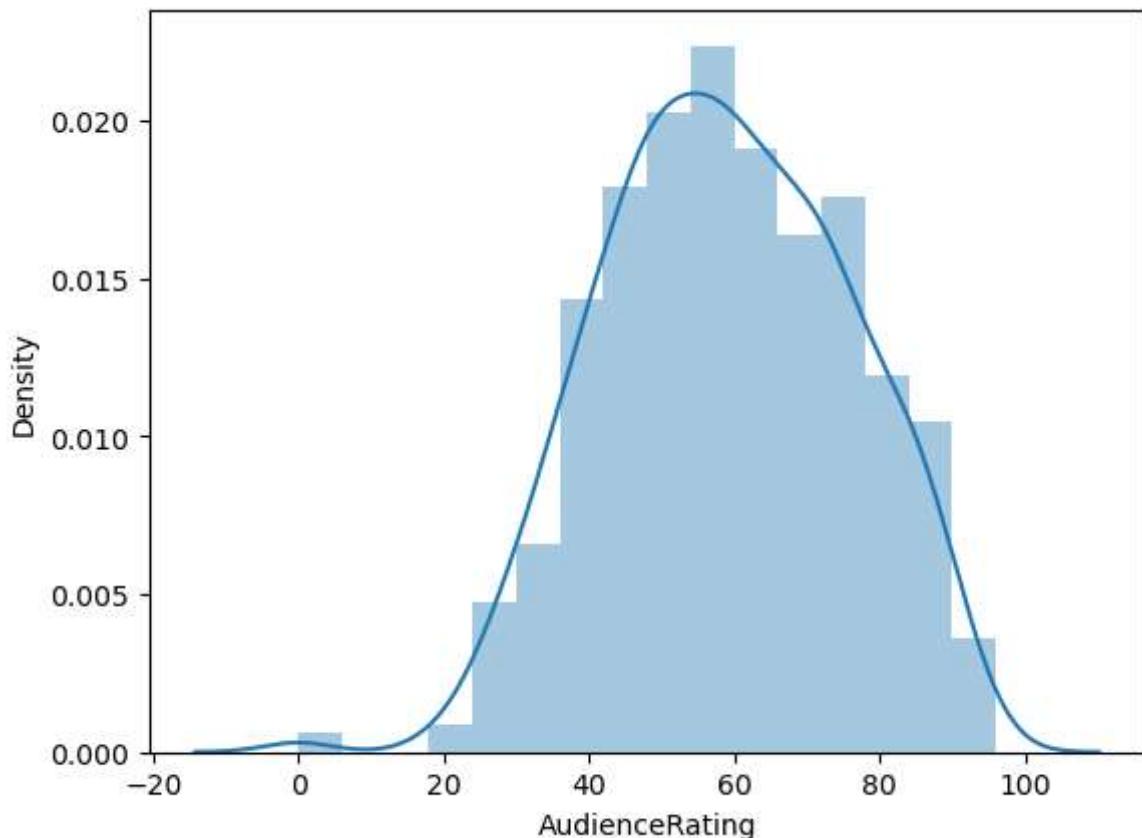


.distplot --> When you run this code, it will produce a distribution plot showing the distribution of the data, including both the histogram and the kernel density estimate.

```
In [30]: m1 = sns.distplot(movies.AudienceRating, kde=False)
```



```
In [31]: m1 = sns.distplot(movies.AudienceRating, kde=True)
```



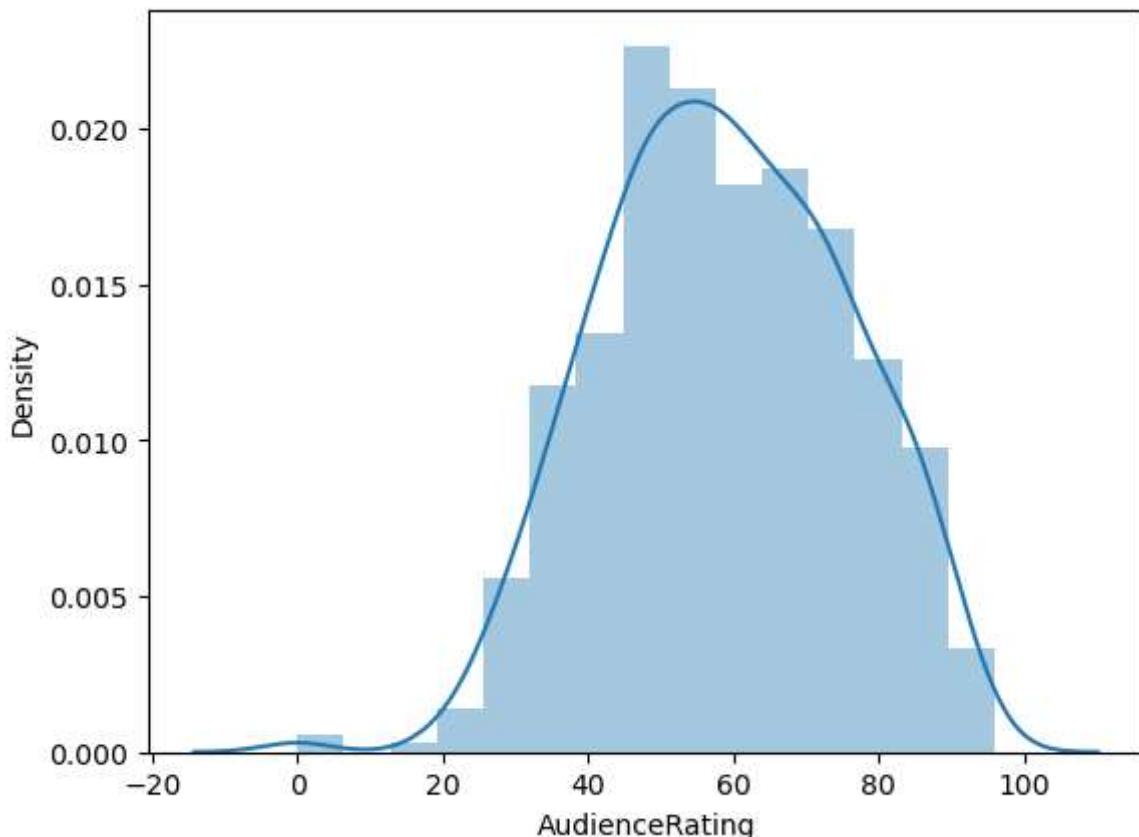
```
In [32]: movies.AudienceRating.count
```

```
Out[32]: <bound method Series.count of 0>      81
1      44
2      52
3      84
4      70
..
554    36
555    52
556    73
557    87
558    42
Name: AudienceRating, Length: 559, dtype: int64
```

```
In [33]: movies.AudienceRating.unique()
```

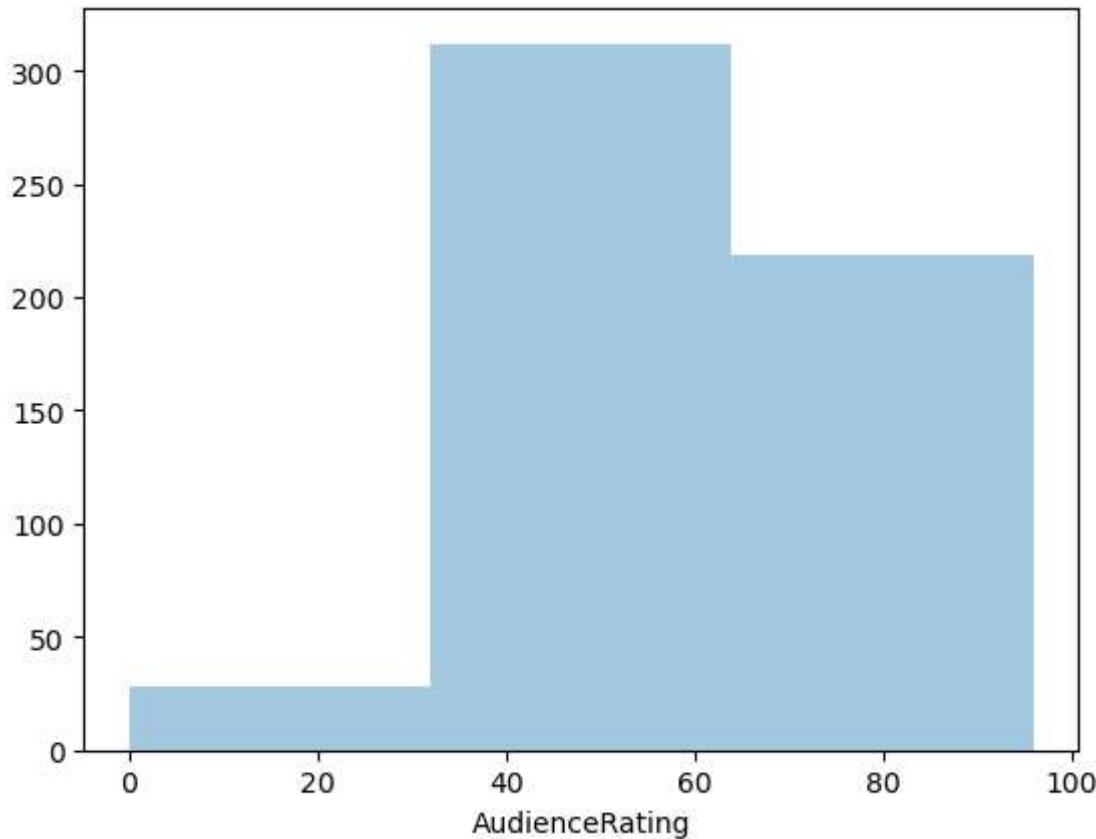
```
Out[33]: array([81, 44, 52, 84, 70, 63, 71, 57, 48, 93, 51, 89, 40, 64, 46, 56, 43,
 72, 37, 35, 87, 78, 66, 31, 55, 34, 49, 69, 92, 74, 59, 32, 38, 60,
 33, 50, 80, 86, 65, 77, 47, 62, 75, 42, 61, 67, 53, 45, 54, 58, 28,
 79, 76, 83, 73, 68, 36, 90, 85, 82, 91, 24, 26, 41, 29, 27, 0, 39,
 25, 19, 20, 96, 88, 22], dtype=int64)
```

```
In [34]: m1 = sns.distplot(movies.AudienceRating, kde=True, bins=15)
```

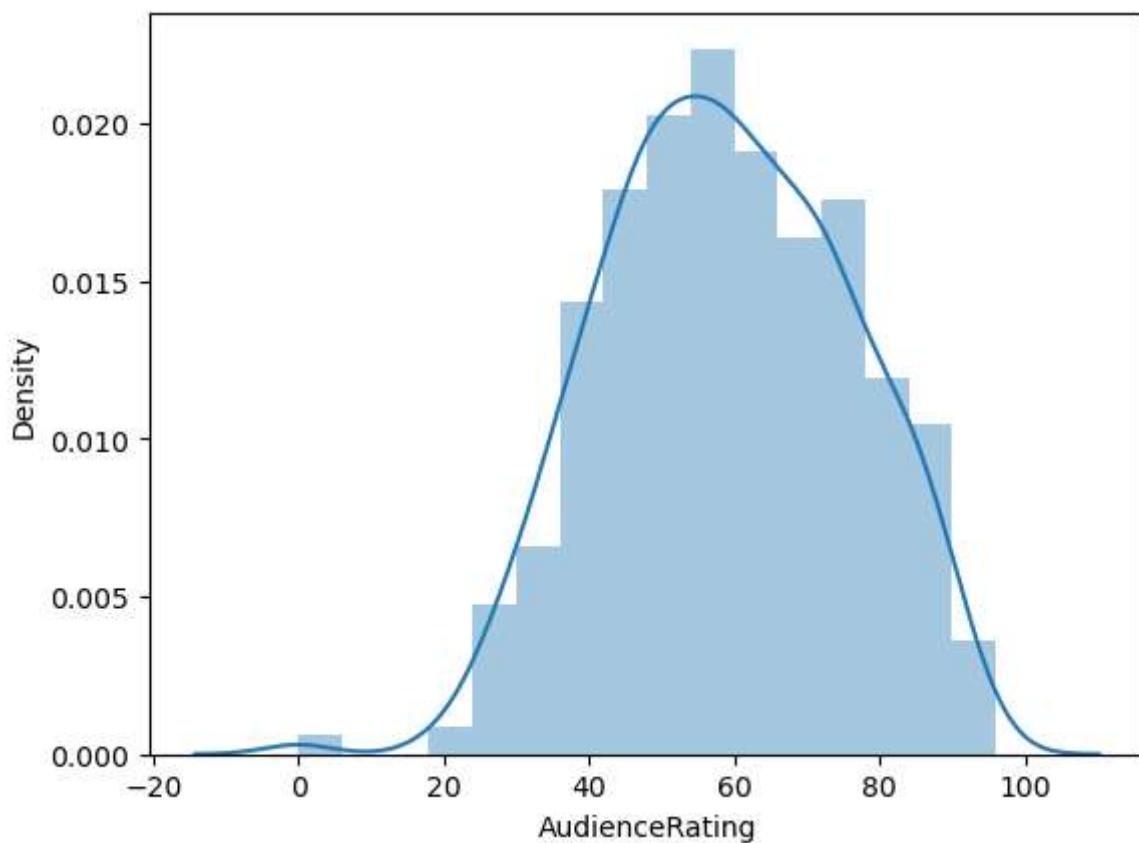


The towers or bars of a histogram are called bins. The height of each bin shows how many values from that data fall into that range. Width of each bin is = $(\text{max value of data} - \text{min value of data}) / \text{total number of bins}$. The default value of the number of bins to be created in a histogram is 10.

```
In [35]: m1 = sns.distplot(movies.AudienceRating, kde=False, bins=3)
```

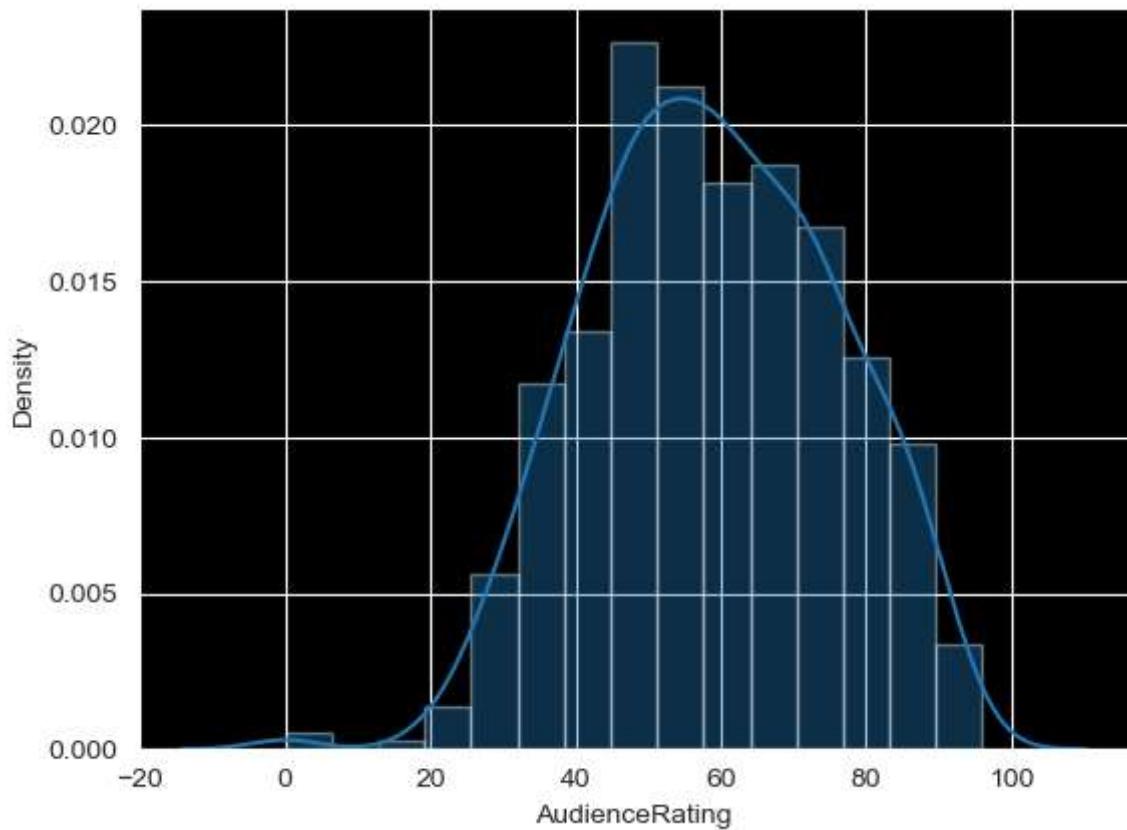


```
In [36]: m1 = sns.distplot(movies.AudienceRating, kde=True, norm_hist=True)
```

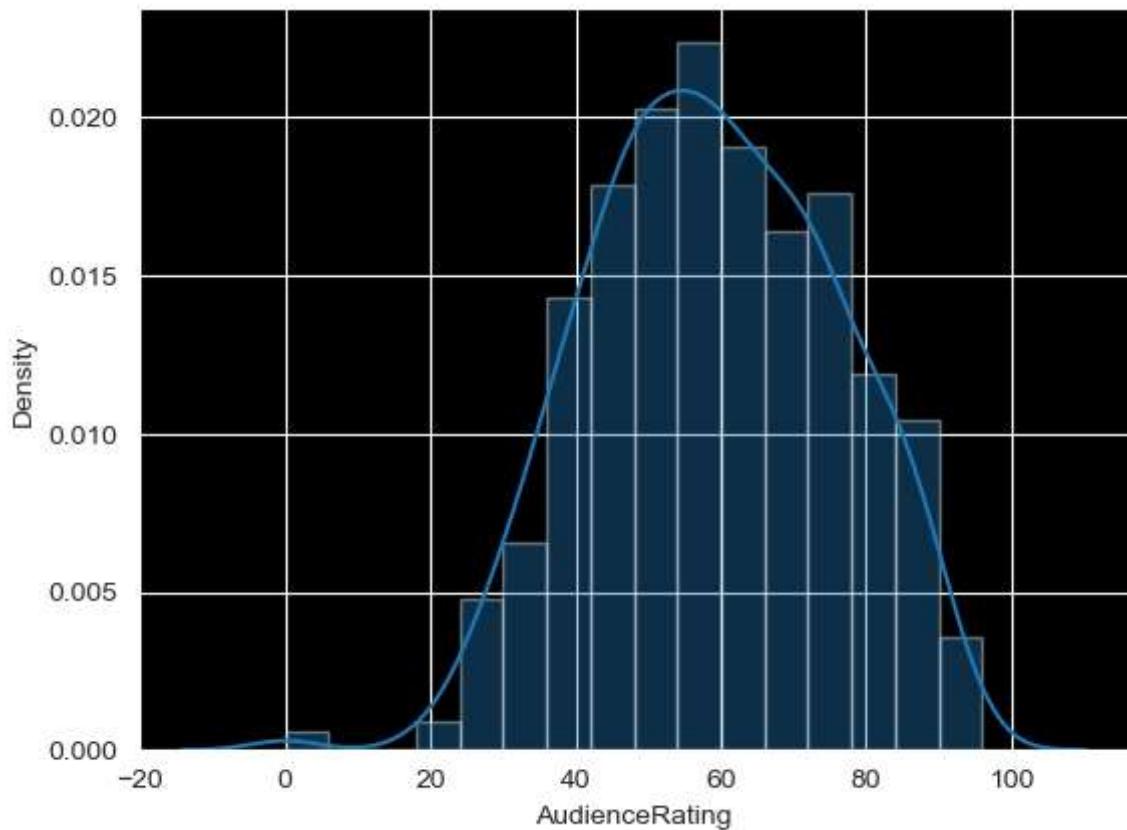


```
In [37]: sns.set_style('darkgrid', {'axes.facecolor' : 'black'})
```

```
In [38]: m2 = sns.distplot(movies.AudienceRating, bins=15)
```

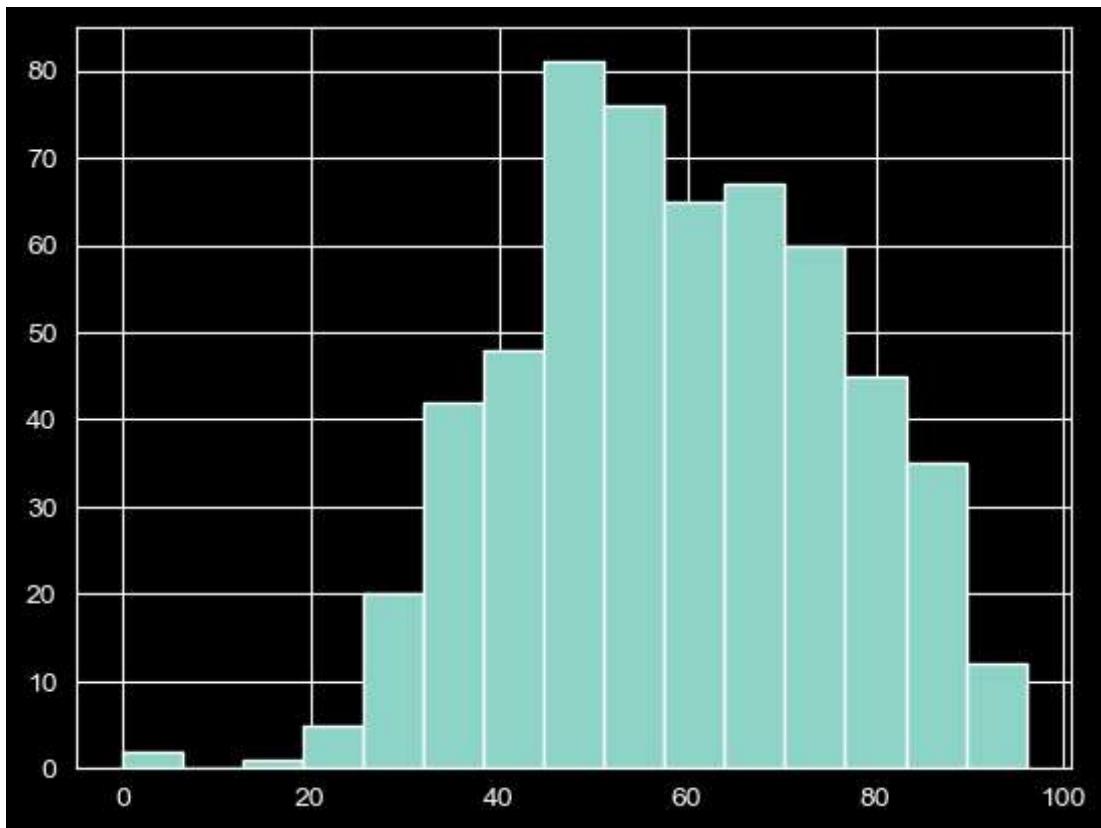


```
In [39]: m2 = sns.distplot(movies.AudienceRating)
```

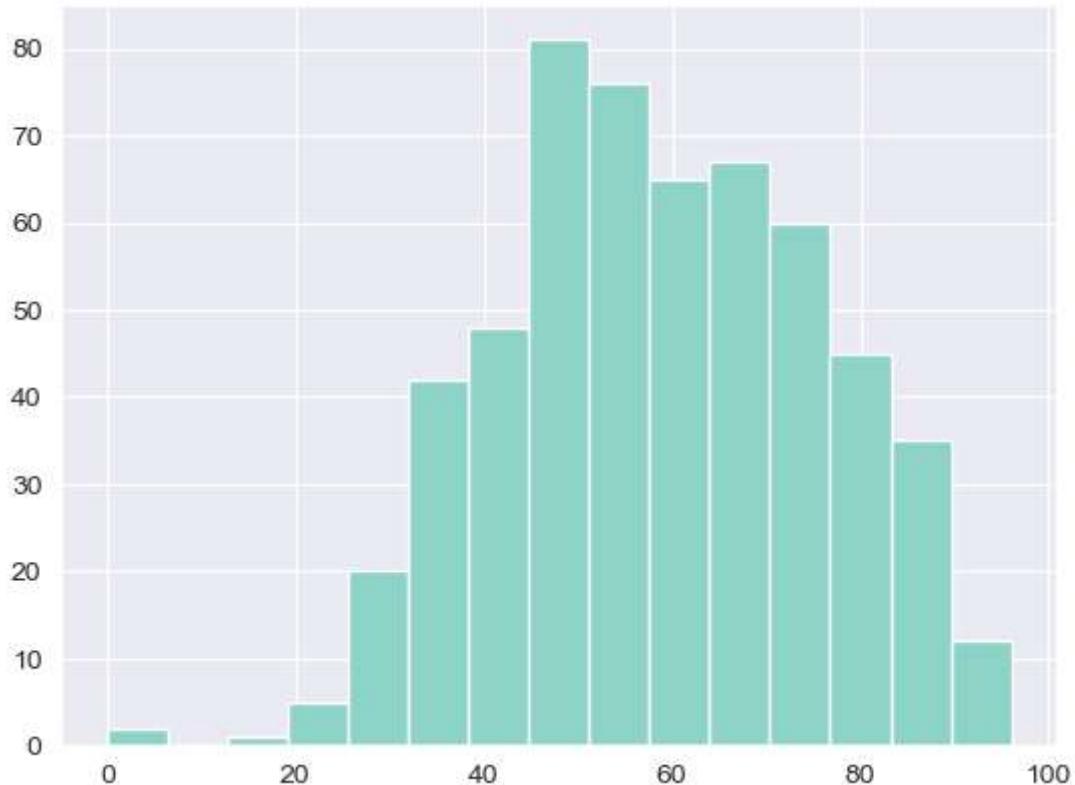


Plot histogram in Matplotlib + Seaborn

```
In [40]: plt.style.use("dark_background")
n1 = plt.hist(movies.AudienceRating, bins=15)
```

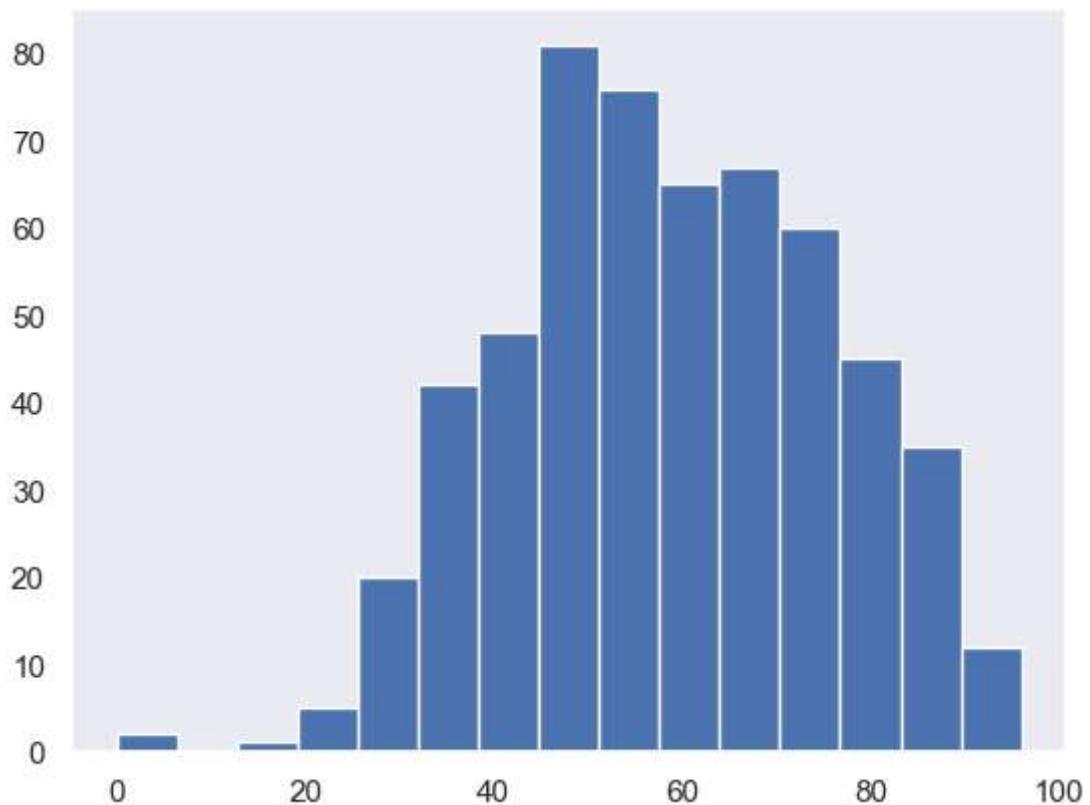


```
In [41]: # darkgrid, whitegrid, dark, white, ticks  
sns.set_style('darkgrid')  
n1 = plt.hist(movies.AudienceRating, bins=15)
```

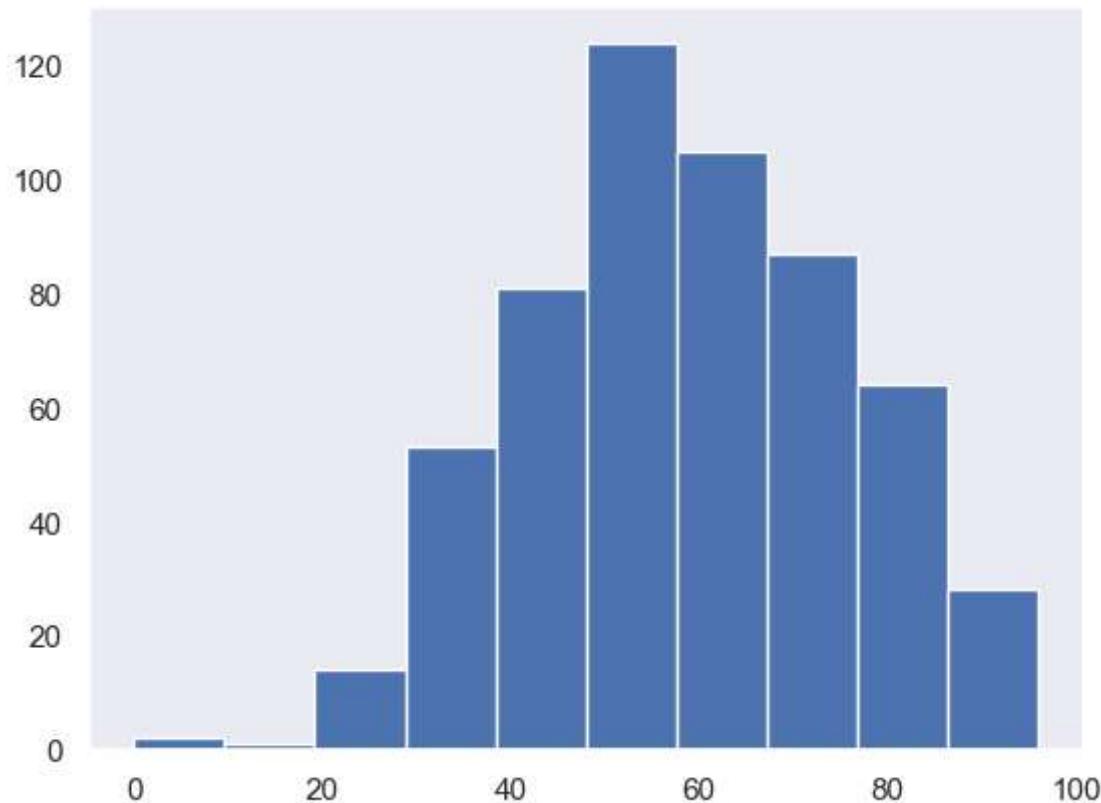


```
In [42]: sns.set_theme('notebook', style='dark')
```

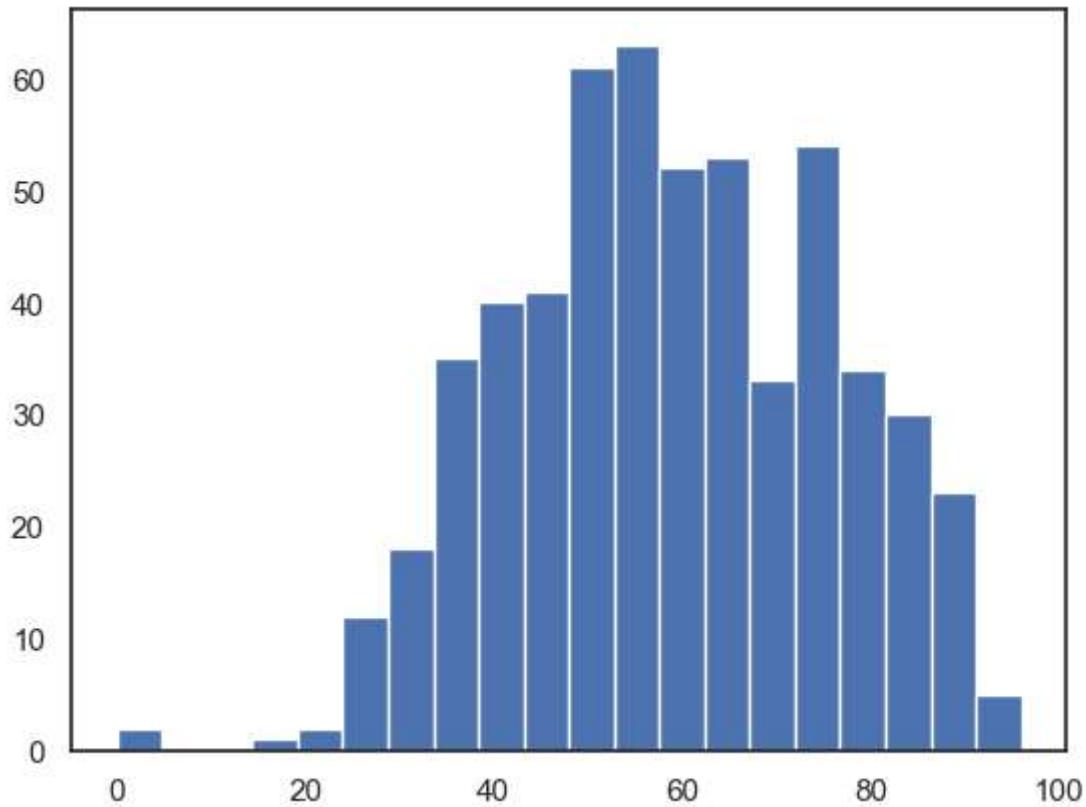
```
n1 = plt.hist(movies.AudienceRating, bins=15)
```



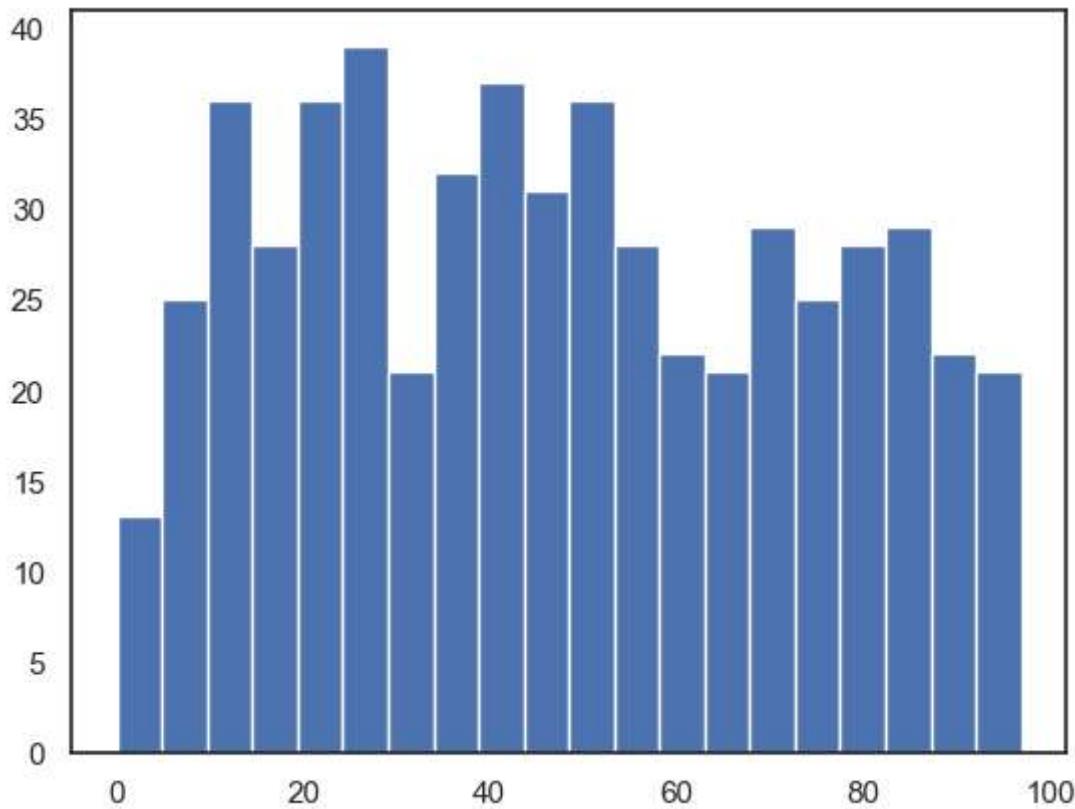
```
In [43]: n1 = plt.hist(movies.AudienceRating)
```



```
In [44]: sns.set_style('white') #normal distribution & called as bell curve  
n1 = plt.hist(movies.AudienceRating, bins=20)
```



```
In [45]: n1 = plt.hist(movies.CriticRating, bins= 20) #uniform distribution
```



```
In [46]: # Creating stacked histograms
```

```
In [47]: movies
```

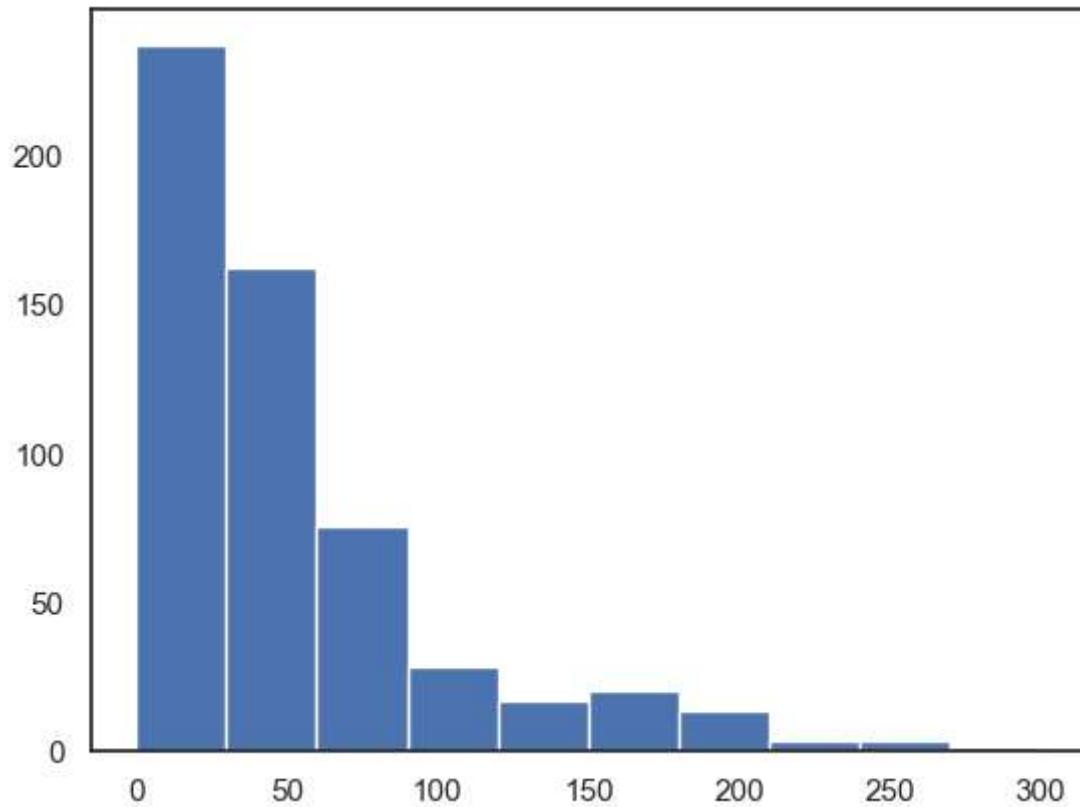
```
Out[47]:
```

	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009
...
554	Your Highness	Comedy	26	36	50	2011
555	Youth in Revolt	Comedy	68	52	18	2009
556	Zodiac	Thriller	89	73	65	2007
557	Zombieland	Action	90	87	24	2009
558	Zookeeper	Comedy	14	42	80	2011

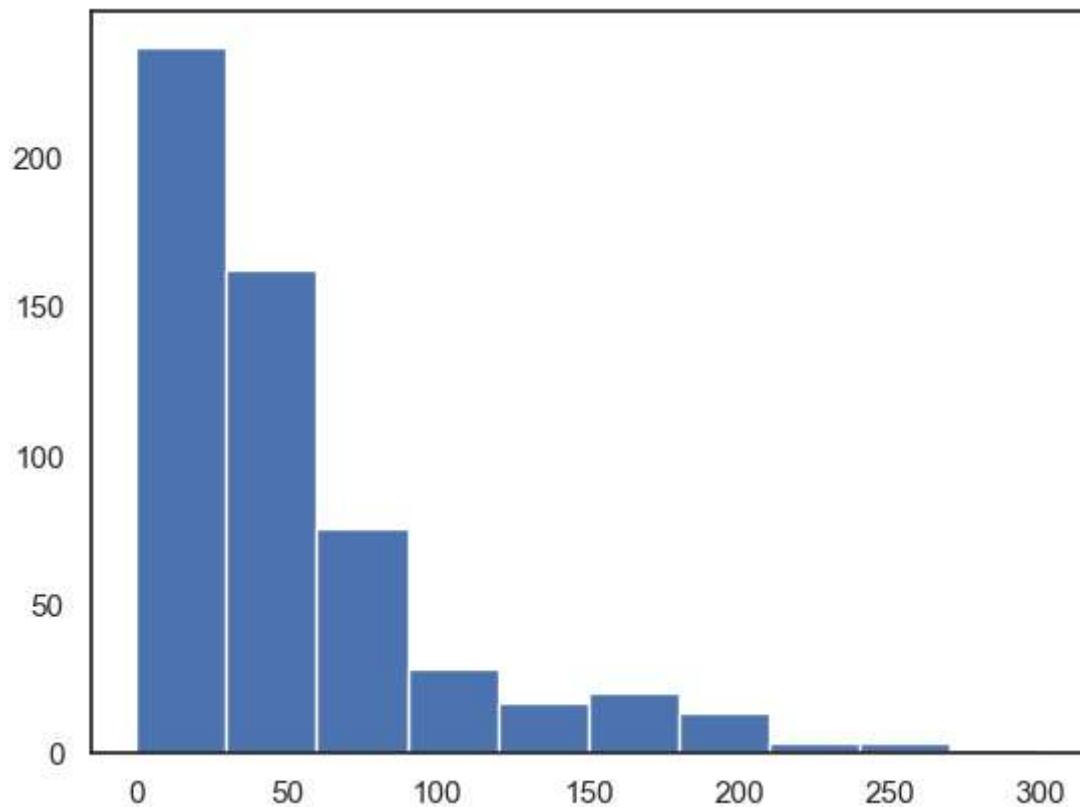
559 rows × 6 columns

```
In [48]: plt.hist(movies.BudgetMillions)
```

```
Out[48]: (array([237., 162., 75., 28., 17., 20., 13., 3., 3., 1.]),  
 array([ 0., 30., 60., 90., 120., 150., 180., 210., 240., 270., 300.]),  
 <BarContainer object of 10 artists>)
```



```
In [49]: plt.hist(movies.BudgetMillions)  
plt.show()
```



```
In [50]: movies
```

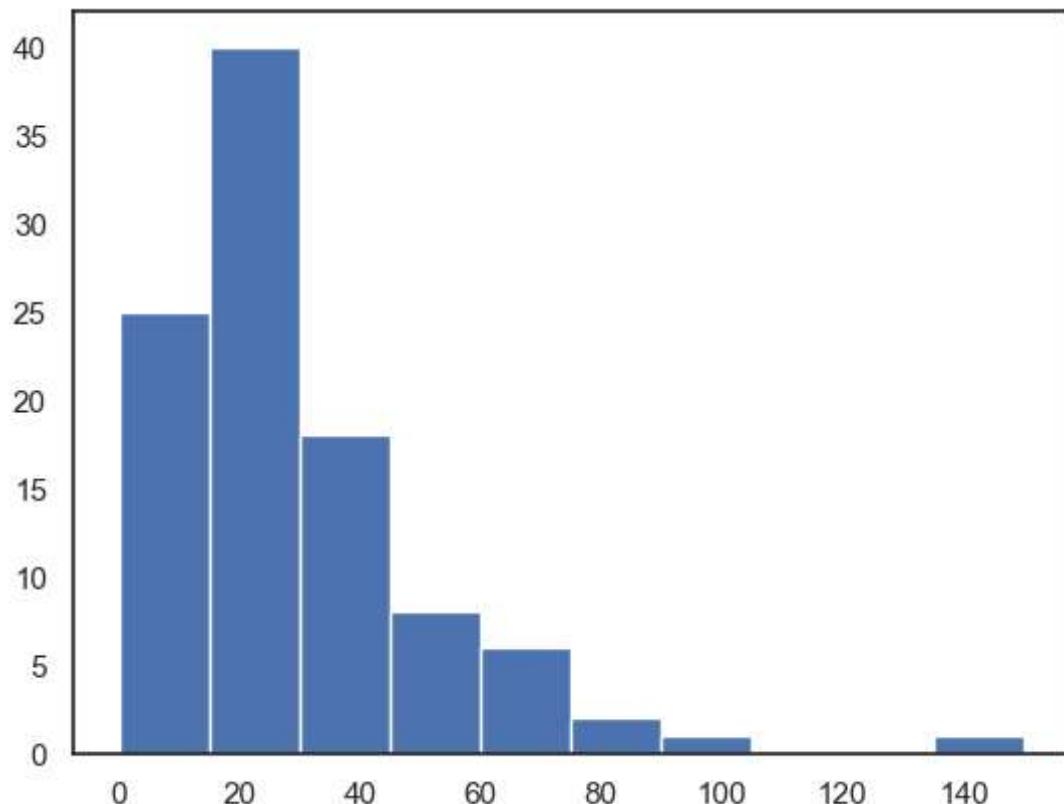
Out[50]:

	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009
...
554	Your Highness	Comedy	26	36	50	2011
555	Youth in Revolt	Comedy	68	52	18	2009
556	Zodiac	Thriller	89	73	65	2007
557	Zombieland	Action	90	87	24	2009
558	Zookeeper	Comedy	14	42	80	2011

559 rows × 6 columns

In [51]:

```
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions)
plt.show()
```



In [52]:

```
movies.head()
```

```
Out[52]:
```

	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009

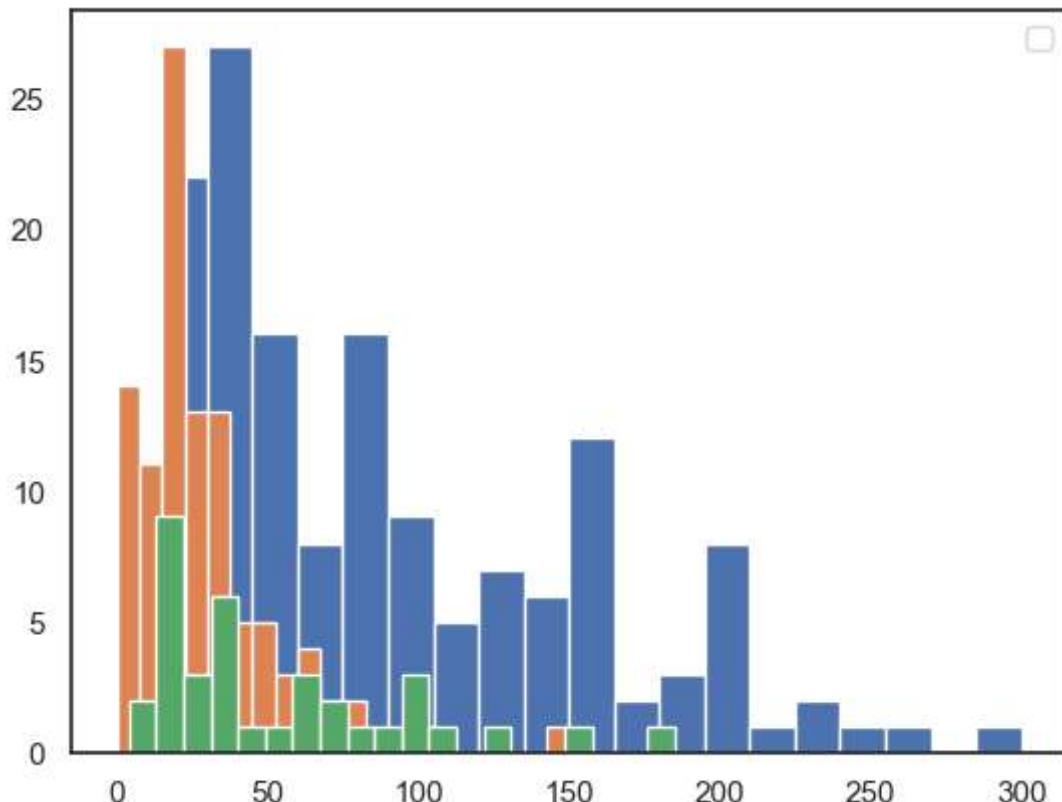
```
In [53]: movies.Genre.unique()
```

```
Out[53]: ['Comedy', 'Adventure', 'Action', 'Horror', 'Drama', 'Romance', 'Thriller']
Categories (7, object): ['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance', 'Thriller']
```

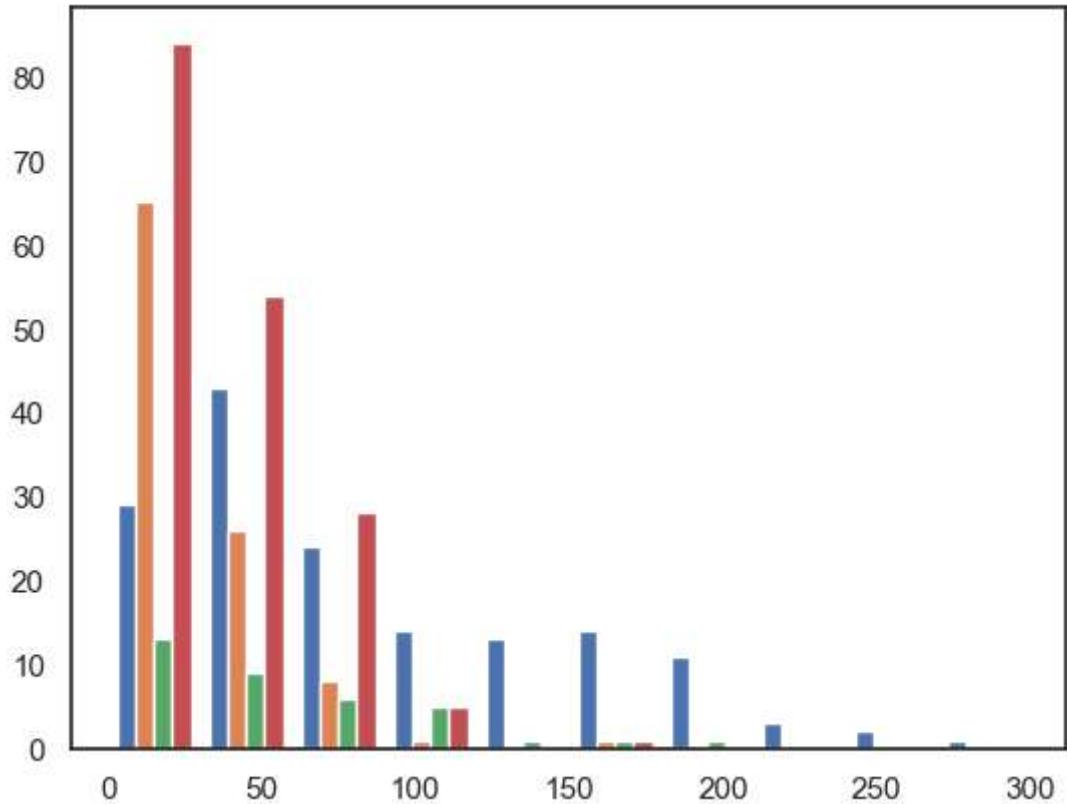
Below plots are stacked histogram because overlaped

```
In [54]: plt.hist(movies[movies.Genre == 'Action'].BudgetMillions, bins = 20)
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions, bins = 20)
plt.hist(movies[movies.Genre == 'Thriller'].BudgetMillions, bins = 20)
plt.legend()
plt.show()
```

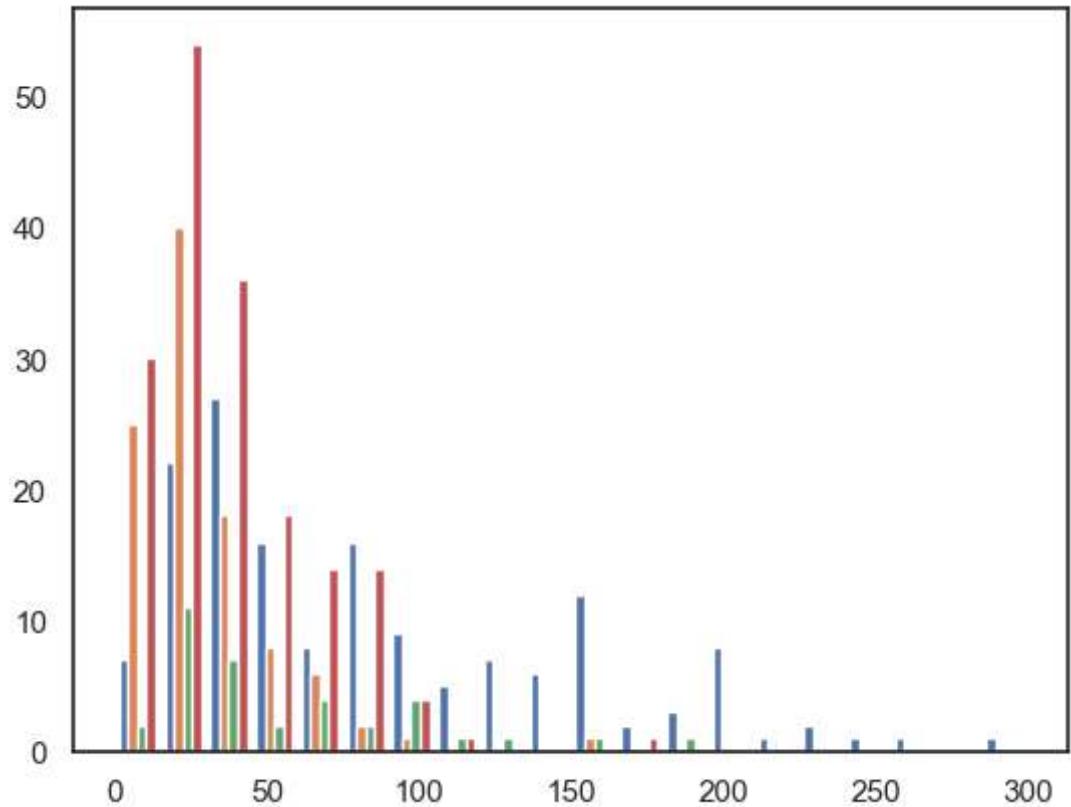
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



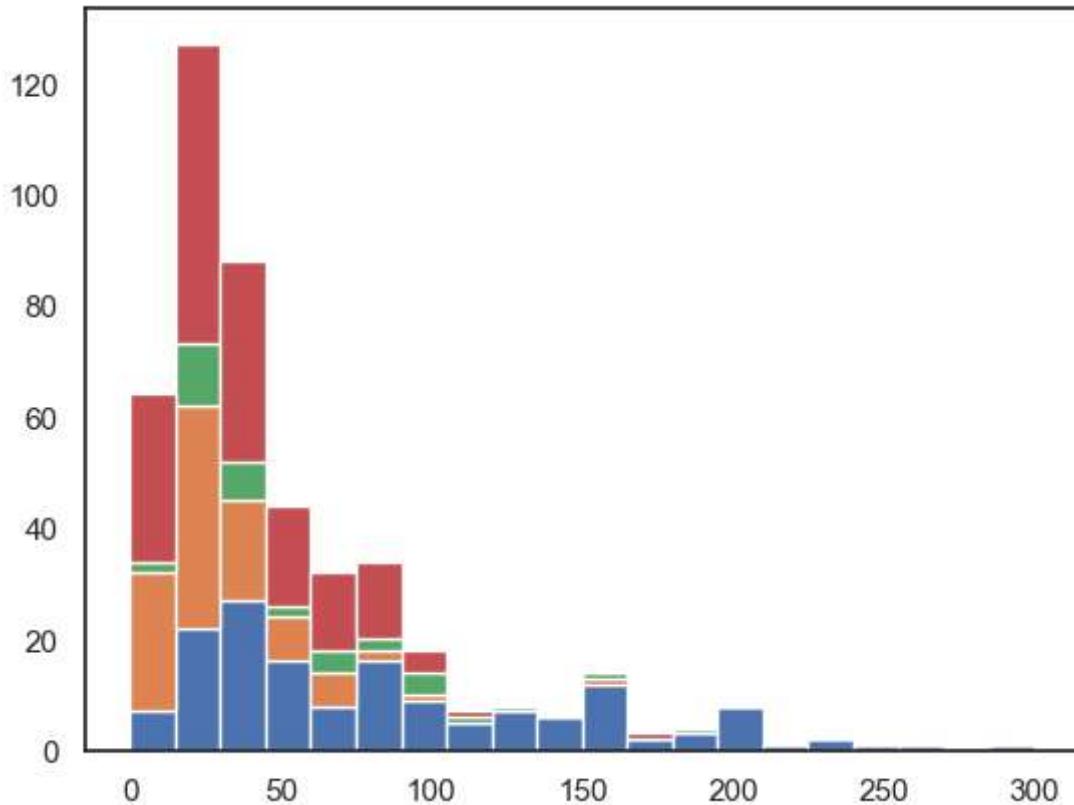
```
In [55]: plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,\n               movies[movies.Genre == 'Drama'].BudgetMillions,\n               movies[movies.Genre == 'Thriller'].BudgetMillions,\n               movies[movies.Genre == 'Comedy'].BudgetMillions])\nplt.show()
```



```
In [56]: plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,\n               movies[movies.Genre == 'Drama'].BudgetMillions,\n               movies[movies.Genre == 'Thriller'].BudgetMillions,\n               movies[movies.Genre == 'Comedy'].BudgetMillions],\n               bins = 20) # stacked is False by default\nplt.show()
```



```
In [57]: plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,\n              movies[movies.Genre == 'Drama'].BudgetMillions,\n              movies[movies.Genre == 'Thriller'].BudgetMillions,\n              movies[movies.Genre == 'Comedy'].BudgetMillions],\n              bins = 20, stacked = True)\nplt.show()
```

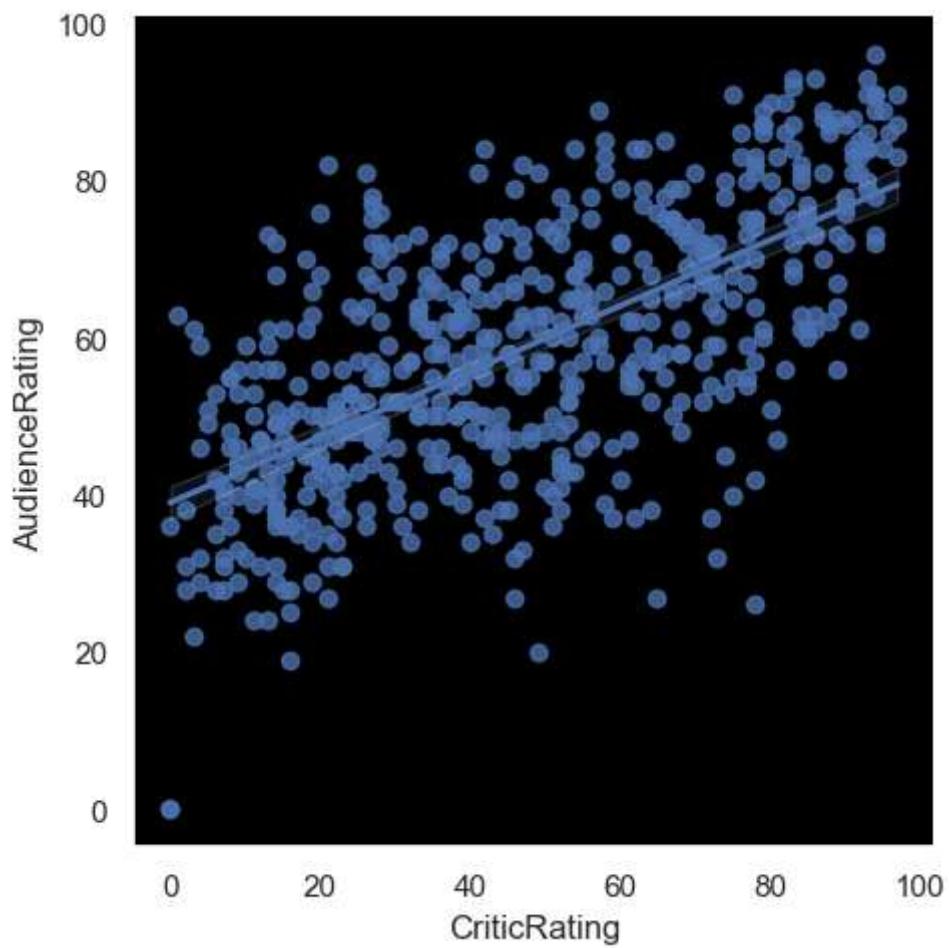


```
In [58]: # for gen in movies.Genre:  
#     print(gen)  
""" if you will use this code you will get all movies name """  
for gen in movies.Genre.cat.categories:  
    print(gen)
```

Action
Adventure
Comedy
Drama
Horror
Romance
Thriller

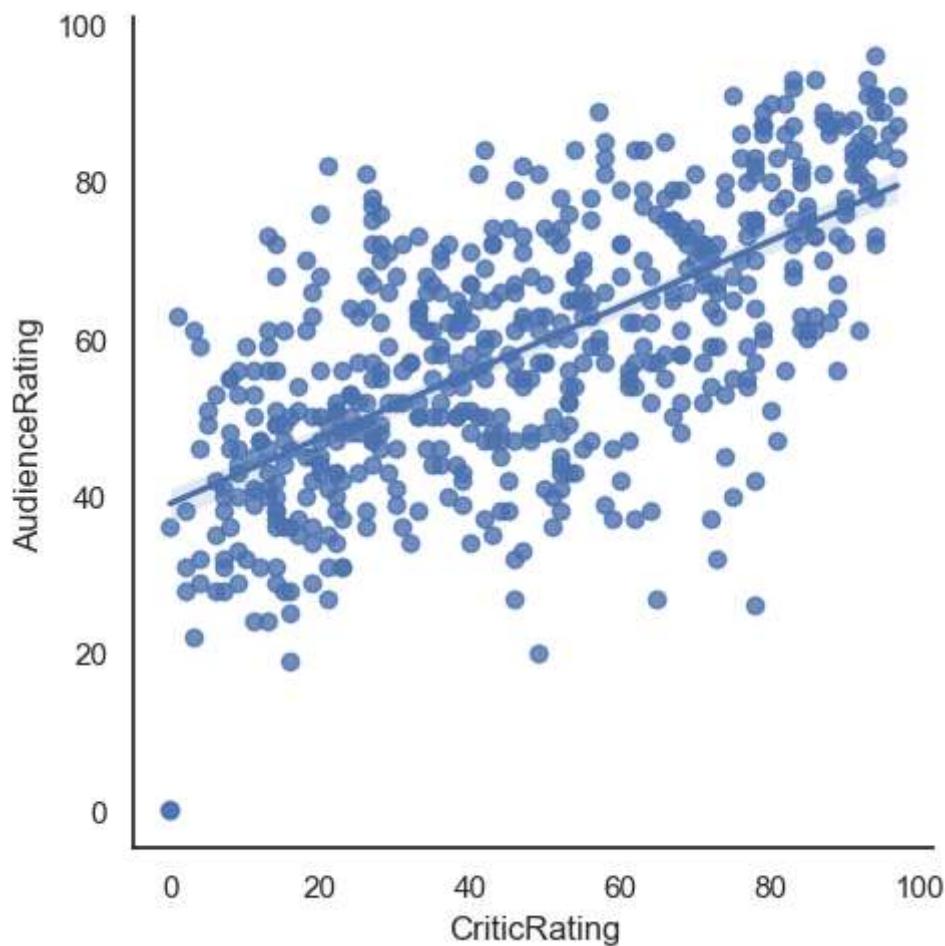
```
In [91]: x = movies['CriticRating']  
y = movies['AudienceRating']  
# Create a regression plot using Seaborn  
sns.lmplot(data=movies, x='CriticRating', y='AudienceRating')
```

Out[91]: <seaborn.axisgrid.FacetGrid at 0x1f49d859060>

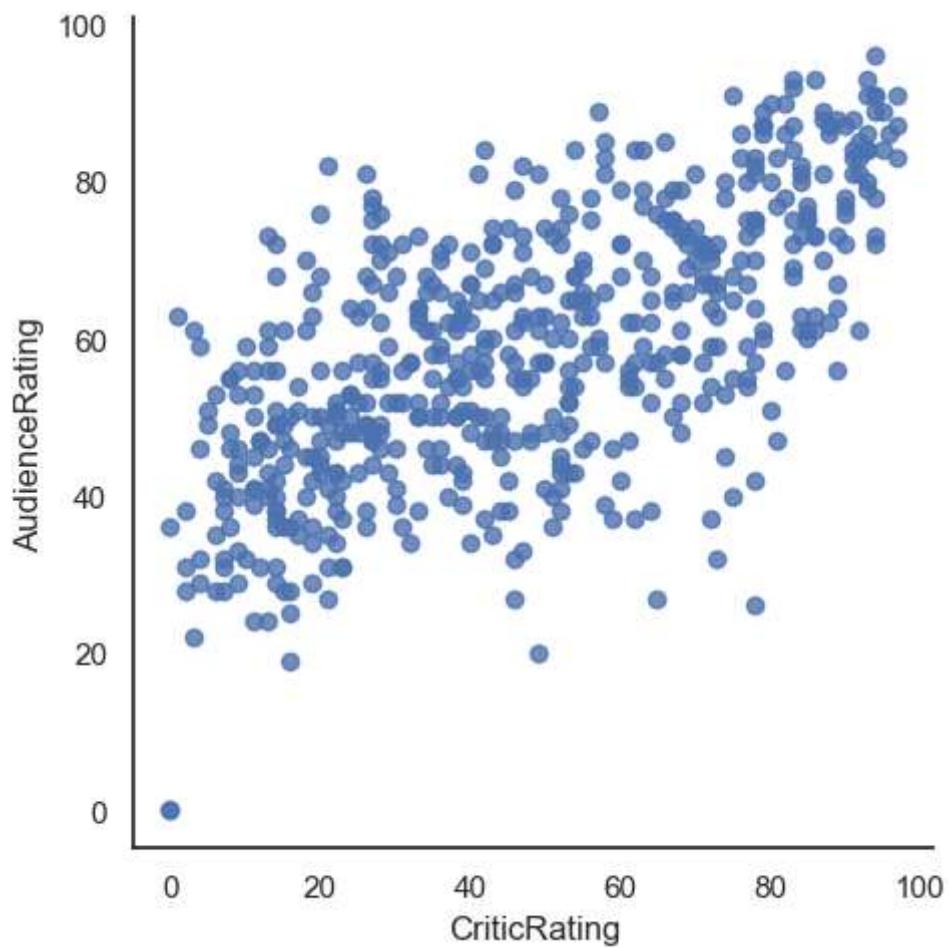


```
In [60]: vis1 = sns.lmplot(data = movies, x = 'CriticRating', y = 'AudienceRating') # fit_r  
vis1
```

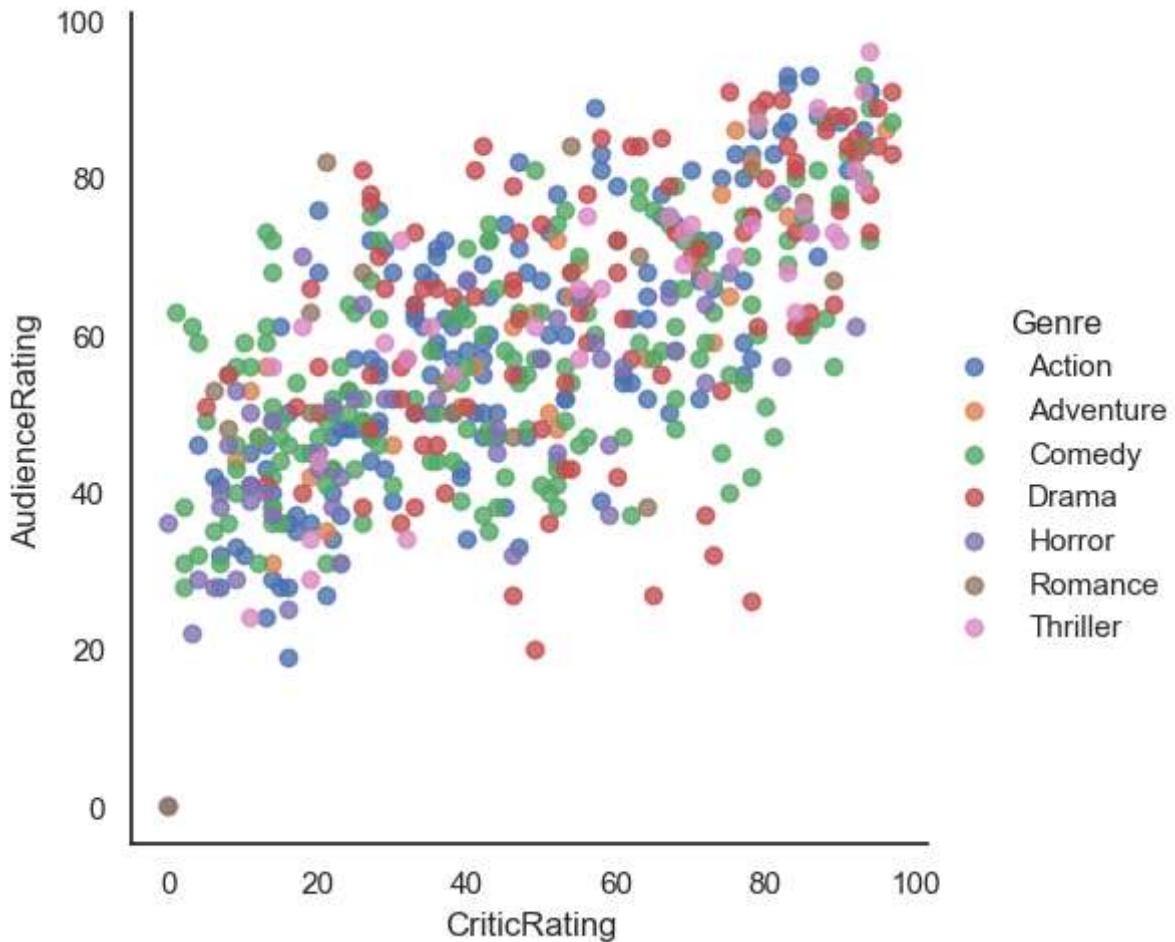
```
Out[60]: <seaborn.axisgrid.FacetGrid at 0x1f4987dbca0>
```



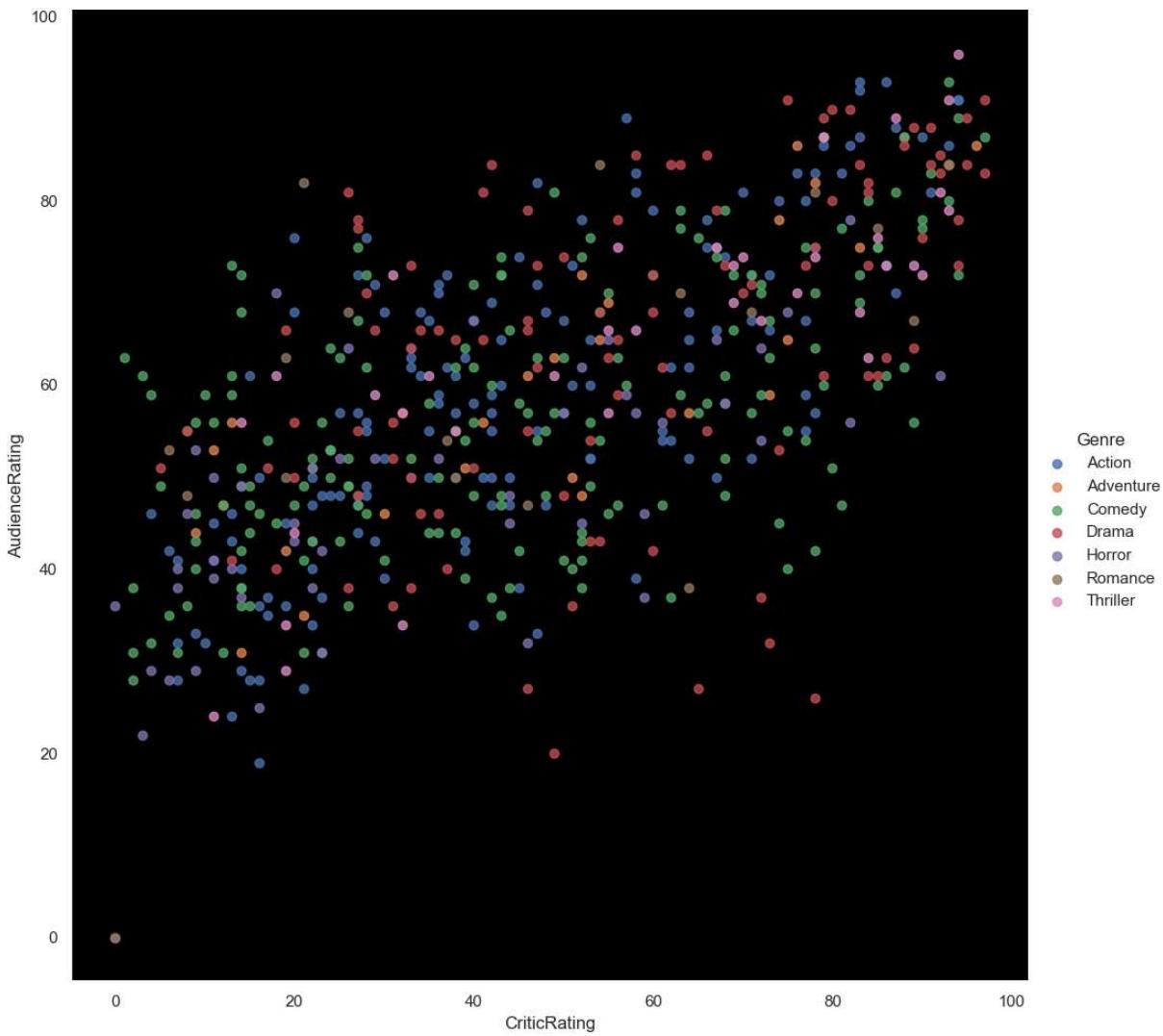
```
In [61]: vis1 = sns.lmplot(data = movies, x = 'CriticRating', y = 'AudienceRating',\n                      fit_reg = False)\n# Line will get removed when we will use 'False'
```



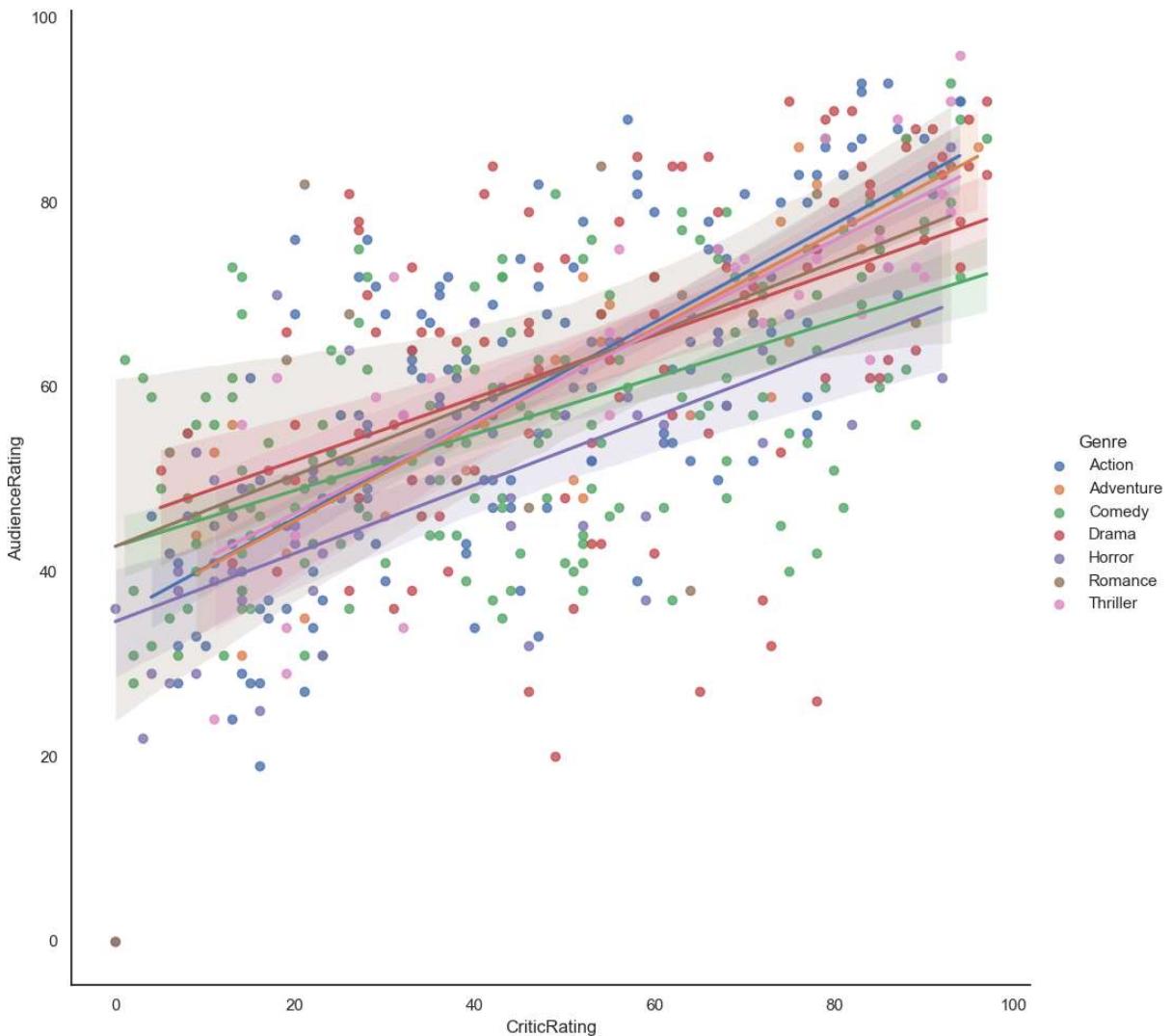
```
In [62]: vis1 = sns.lmplot(data = movies, x = 'CriticRating', y = 'AudienceRating',\n                      fit_reg = False, hue = 'Genre')
```



```
In [92]: vis1 = sns.lmplot(data = movies, x = 'CriticRating', y = 'AudienceRating',\n                      fit_reg = False, hue = 'Genre', height = 10, aspect = 1)
```

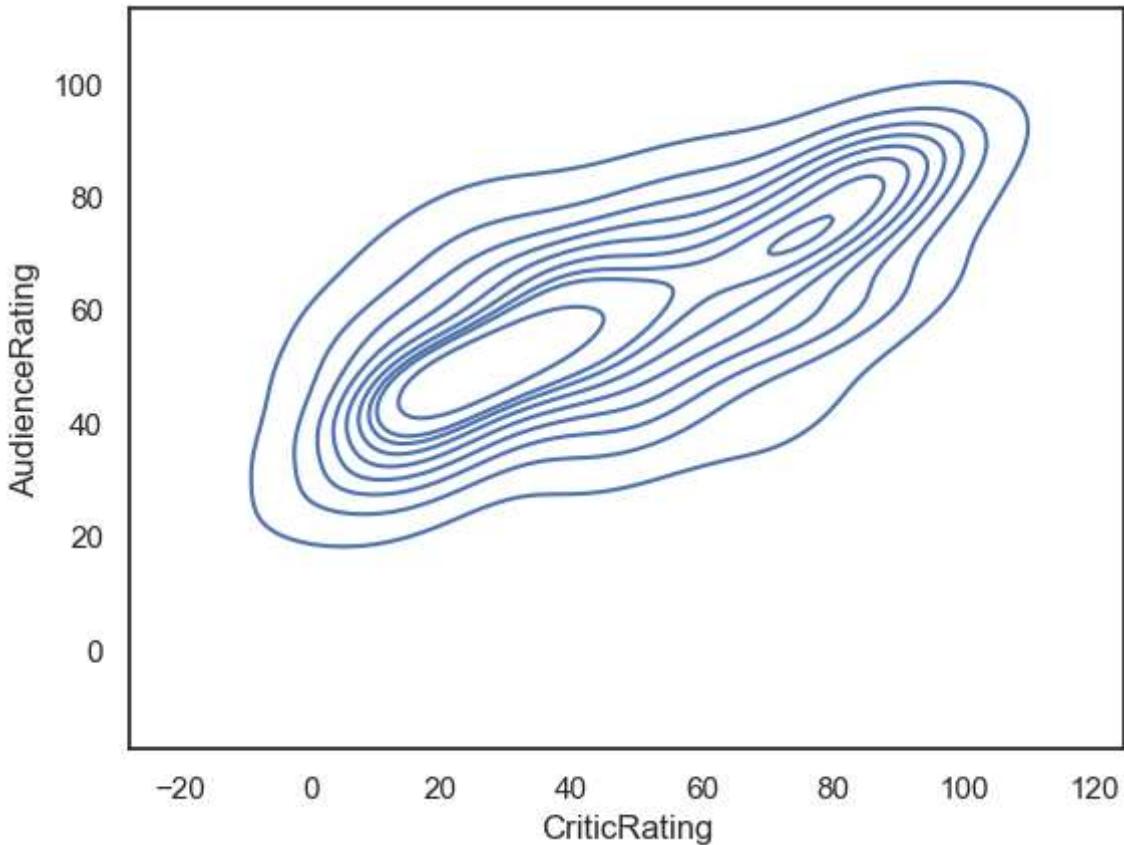


```
In [64]: vis1 = sns.lmplot(data = movies, x = 'CriticRating', y = 'AudienceRating',\n                      fit_reg = True, hue = 'Genre', height = 10, aspect = 1)
```



```
In [65]: # Kernel Density Estimate Plot (KDE plot)
# how can i visualize audience rating & critics rating . using scatterplot
```

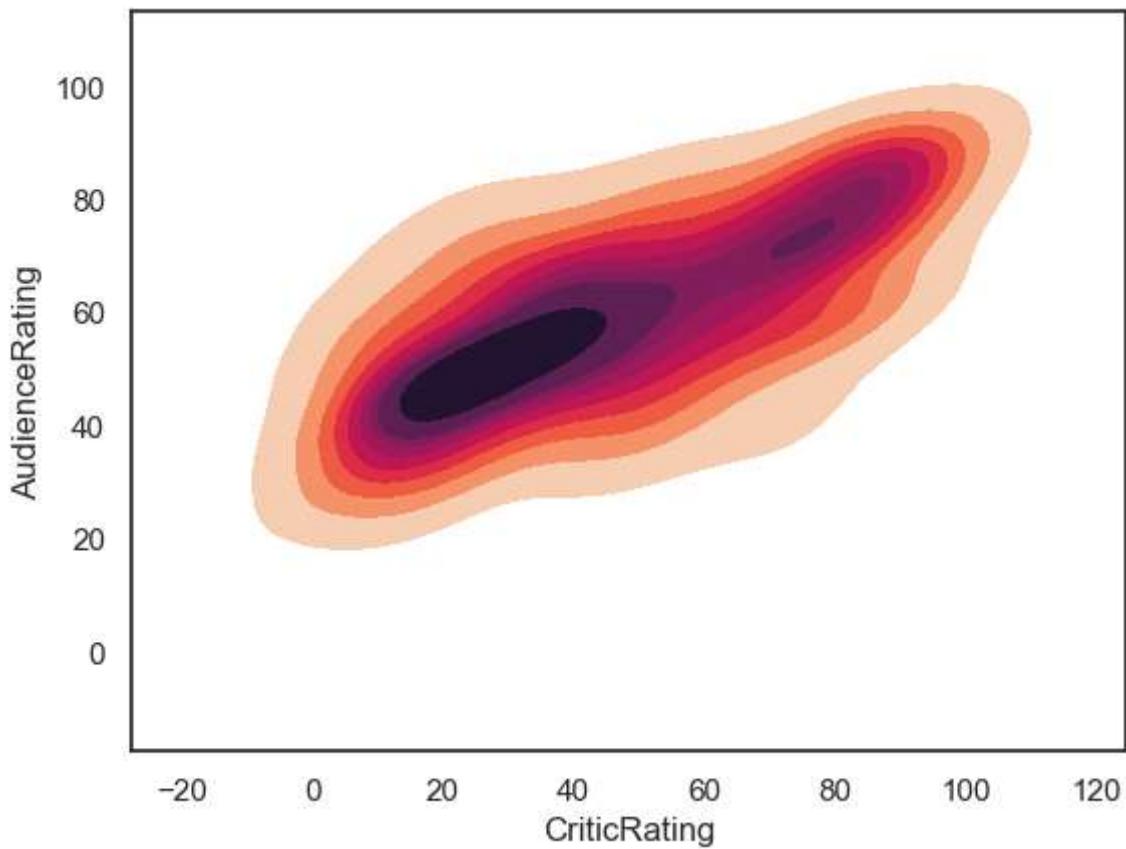
```
In [66]: k1 = sns.kdeplot(x= movies.CriticRating, y=movies.AudienceRating) # use X and Y va
# where do u find more density and how density is distributed across from the the ch
# center point is kernel this is calld KDE & insteade of dots it visualize like thi
# we can able to clearly see the spread at the audience ratings
```



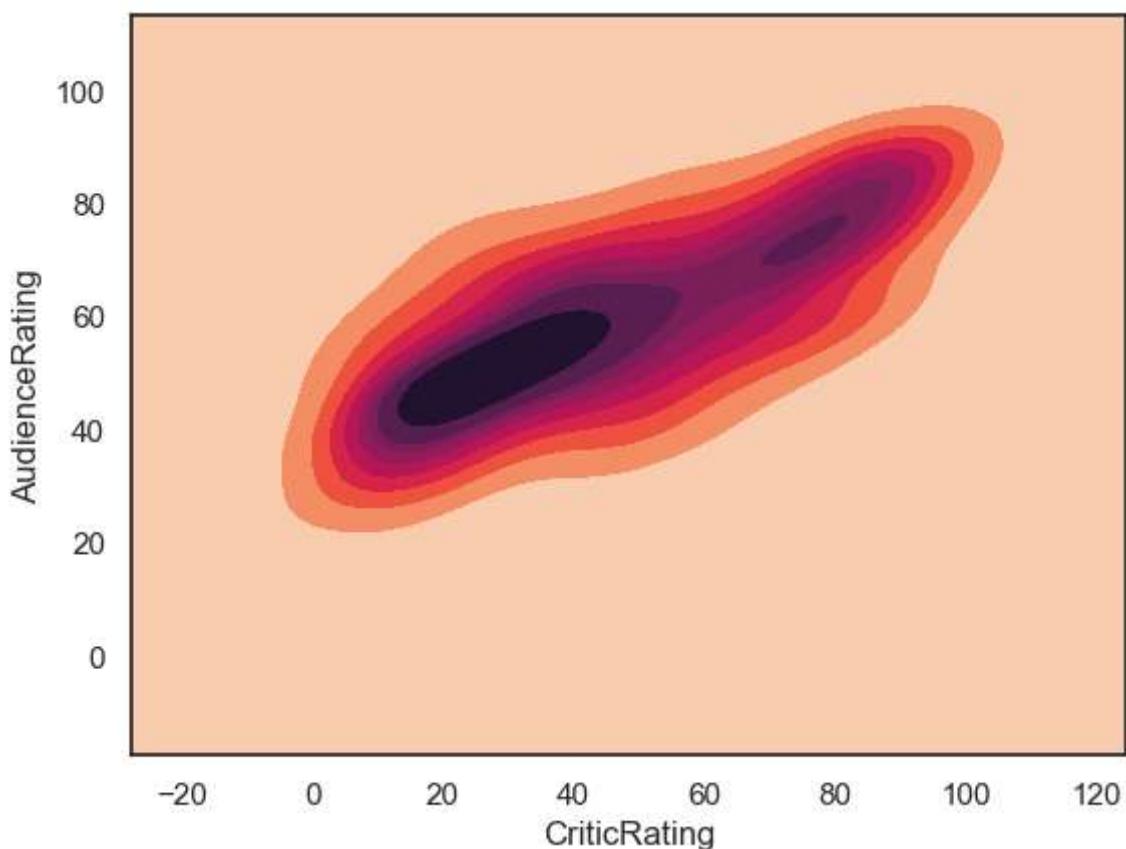
As with the convention in matplotlib, every continuous colormap has a reversed version, which has the suffix "`_r`":

- When `shade` is set to `True`, Seaborn will fill the area under the curve or within the confidence interval with a color (usually a translucent shade) to visually emphasize the region of interest.
- When `shade` is set to `False`, Seaborn will not fill the area under the curve or within the confidence interval with any color. This can be useful when you want to emphasize the line or points in the plot without the distraction of shading.

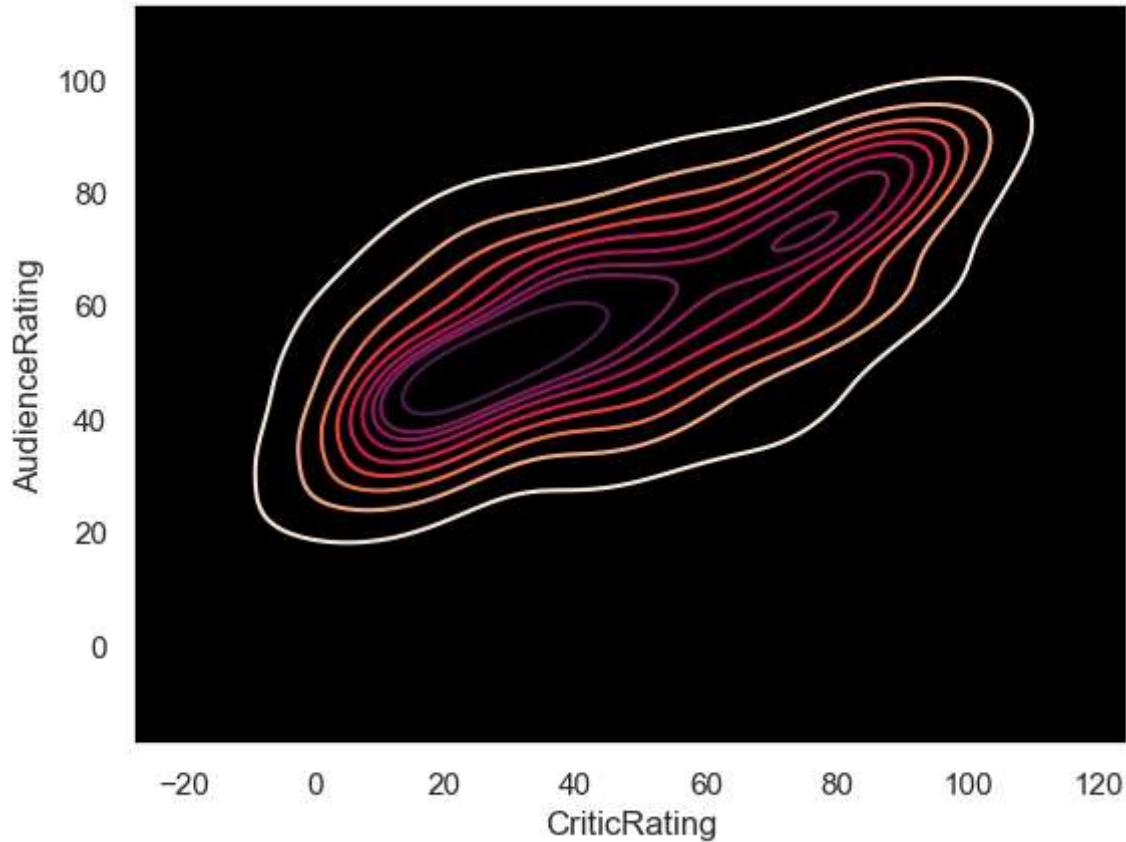
```
In [67]: k1 = sns.kdeplot(x = movies.CriticRating, y = movies.AudienceRating, shade = True, s
```



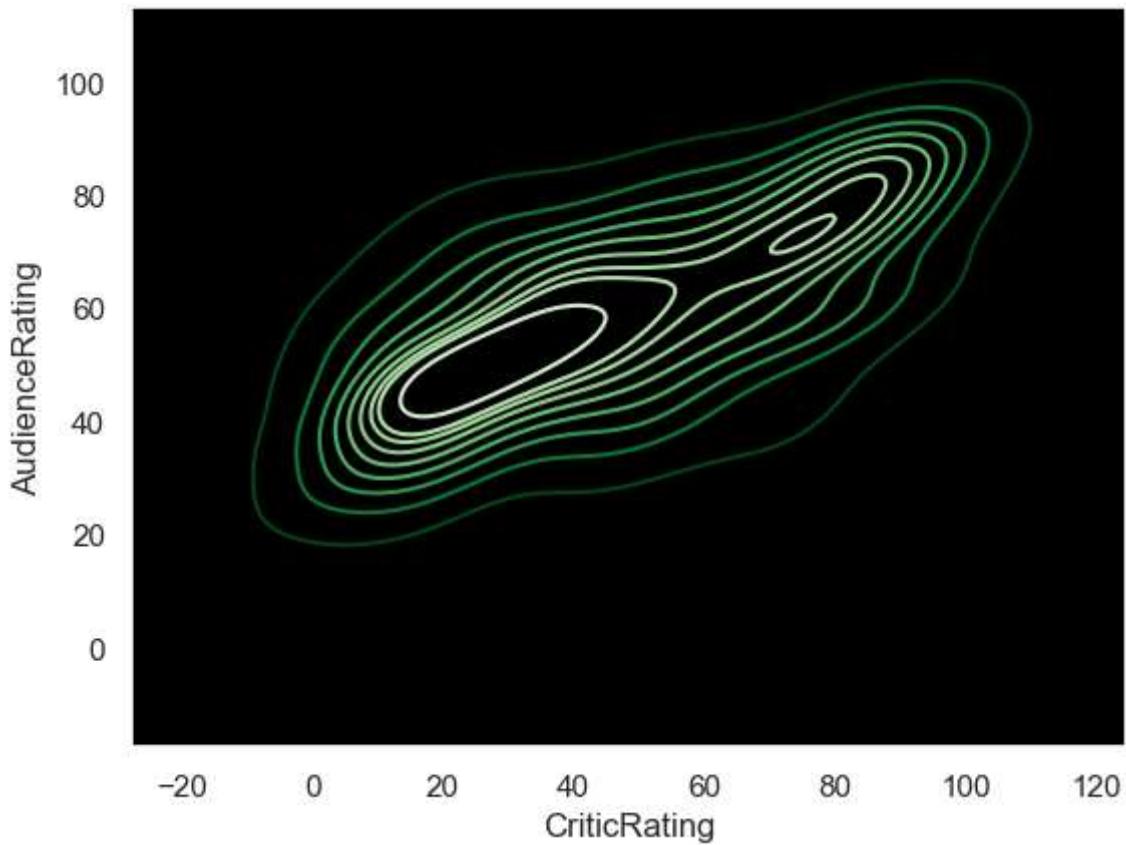
```
In [68]: k1 = sns.kdeplot(x = movies.CriticRating, y = movies.AudienceRating, shade = True,s
```



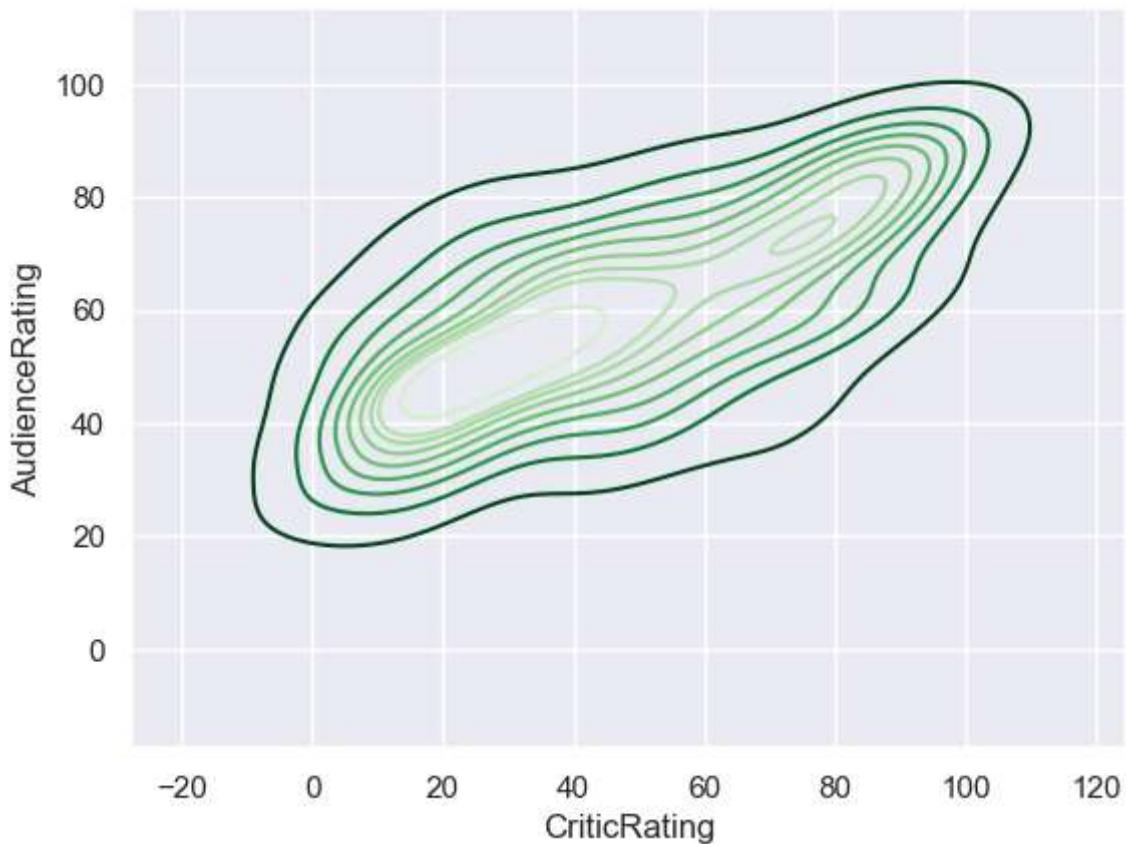
```
In [93]: k1 = sns.kdeplot(x = movies.CriticRating, y = movies.AudienceRating, shade = False,
```



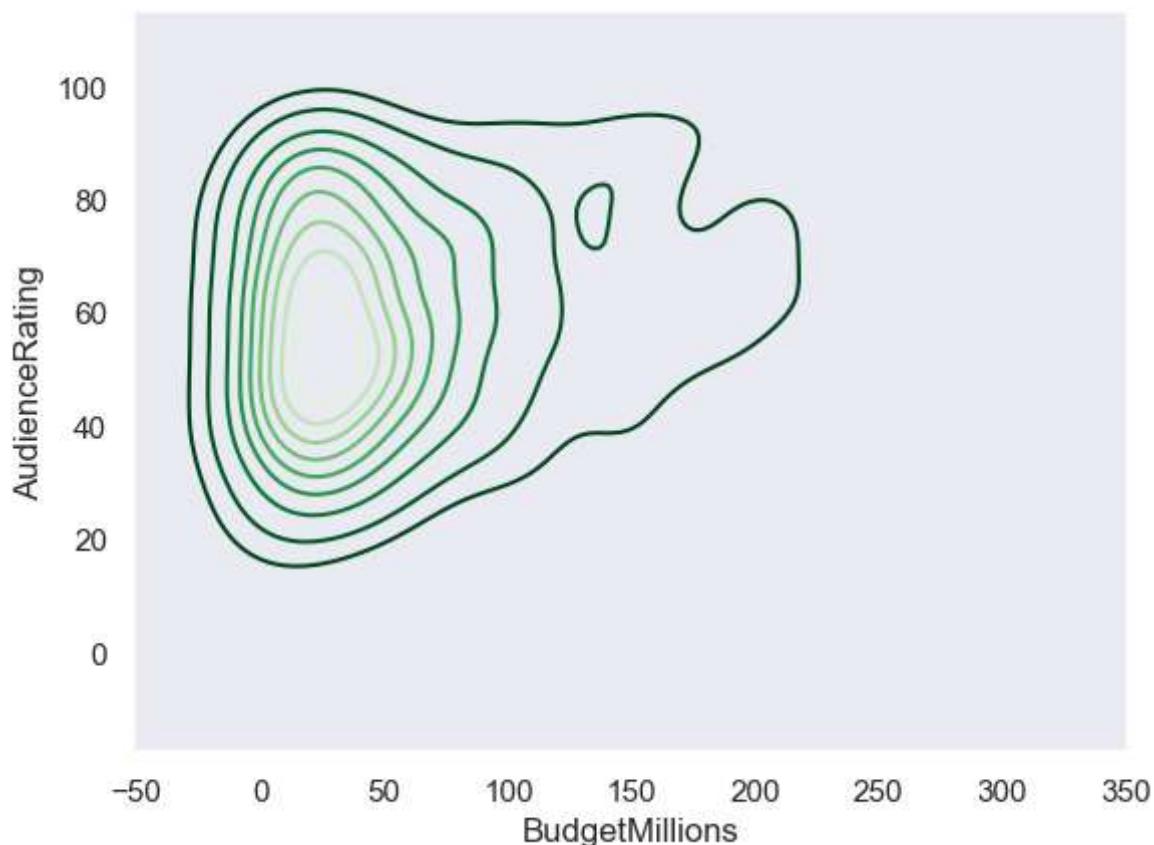
```
In [94]: k1 = sns.kdeplot(x = movies.CriticRating, y = movies.AudienceRating, shade_lowest =
```



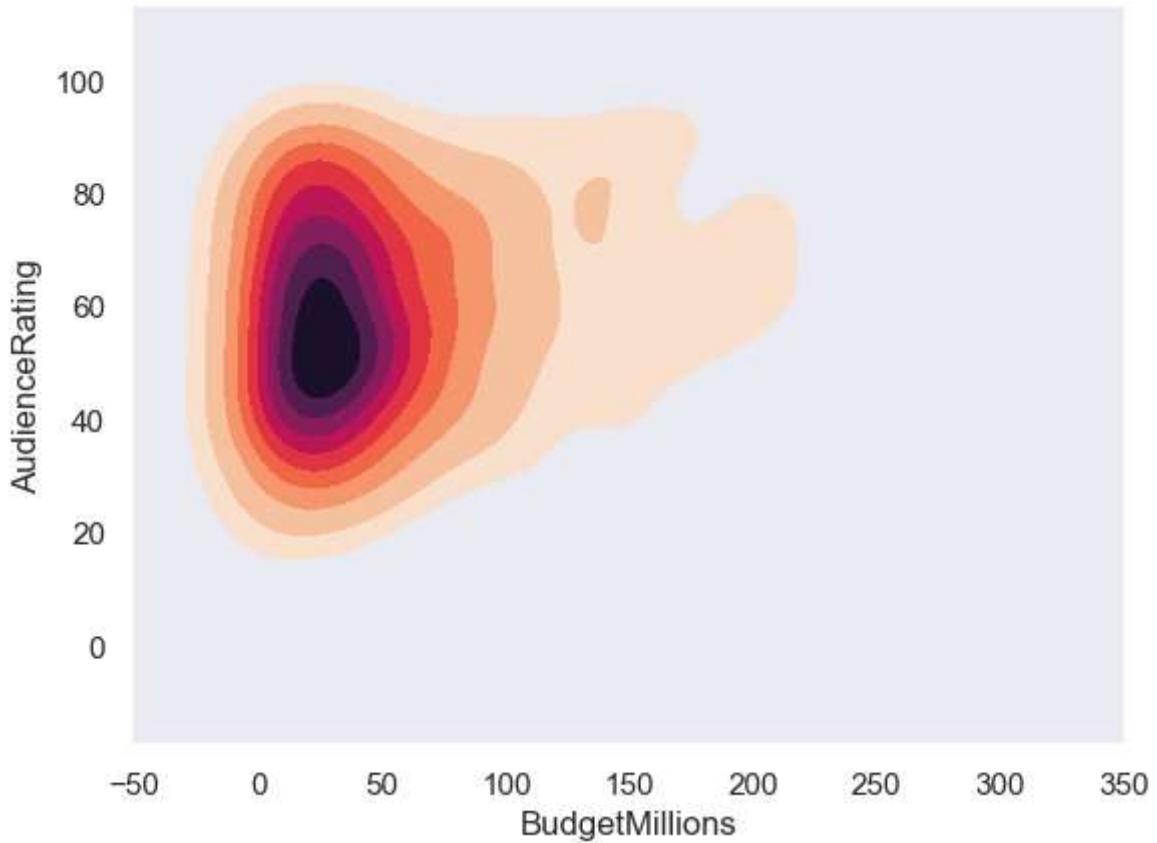
```
In [71]: sns.set_style('darkgrid') # the background we have is in the form of dark mode with  
k1 = sns.kdeplot(x = movies.CriticRating, y = movies.AudienceRating, shade_lowest
```



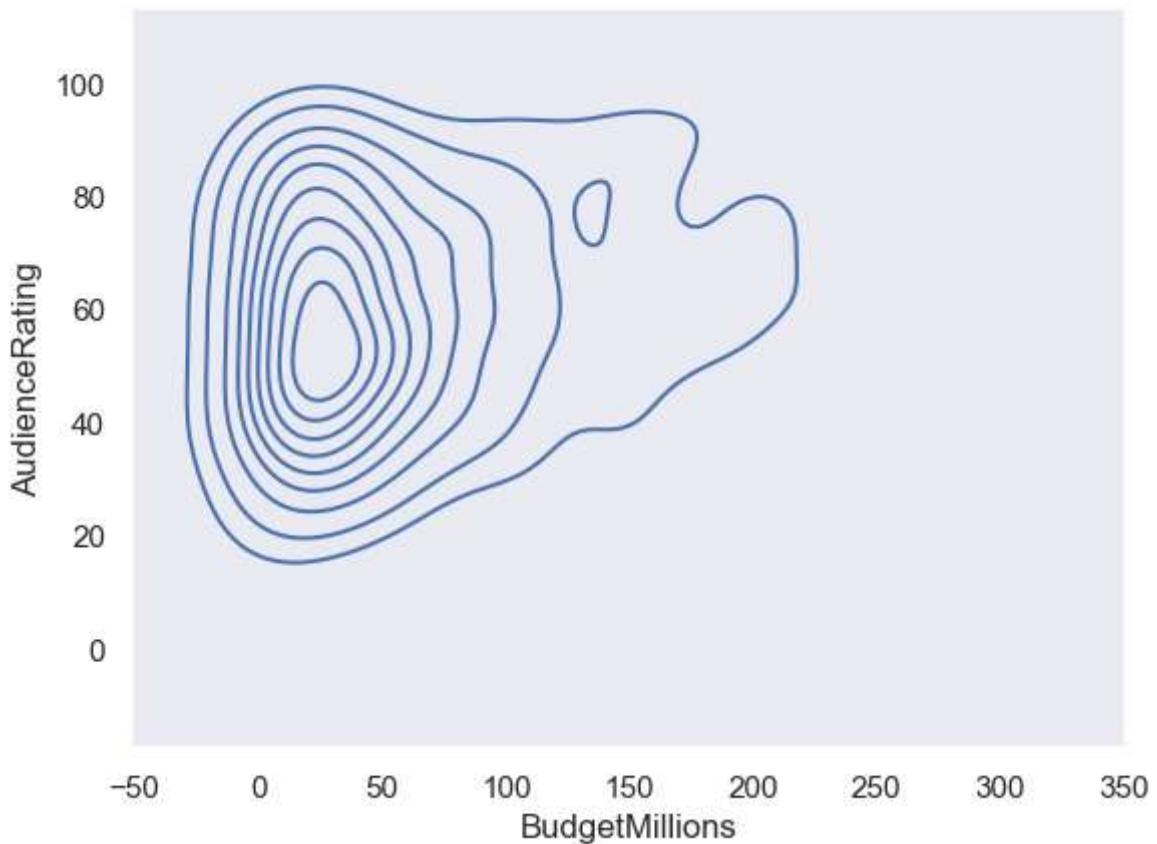
```
In [72]: sns.set_style('dark')
k1 = sns.kdeplot(x = movies.BudgetMillions, y = movies.AudienceRating, shade_lowes
```



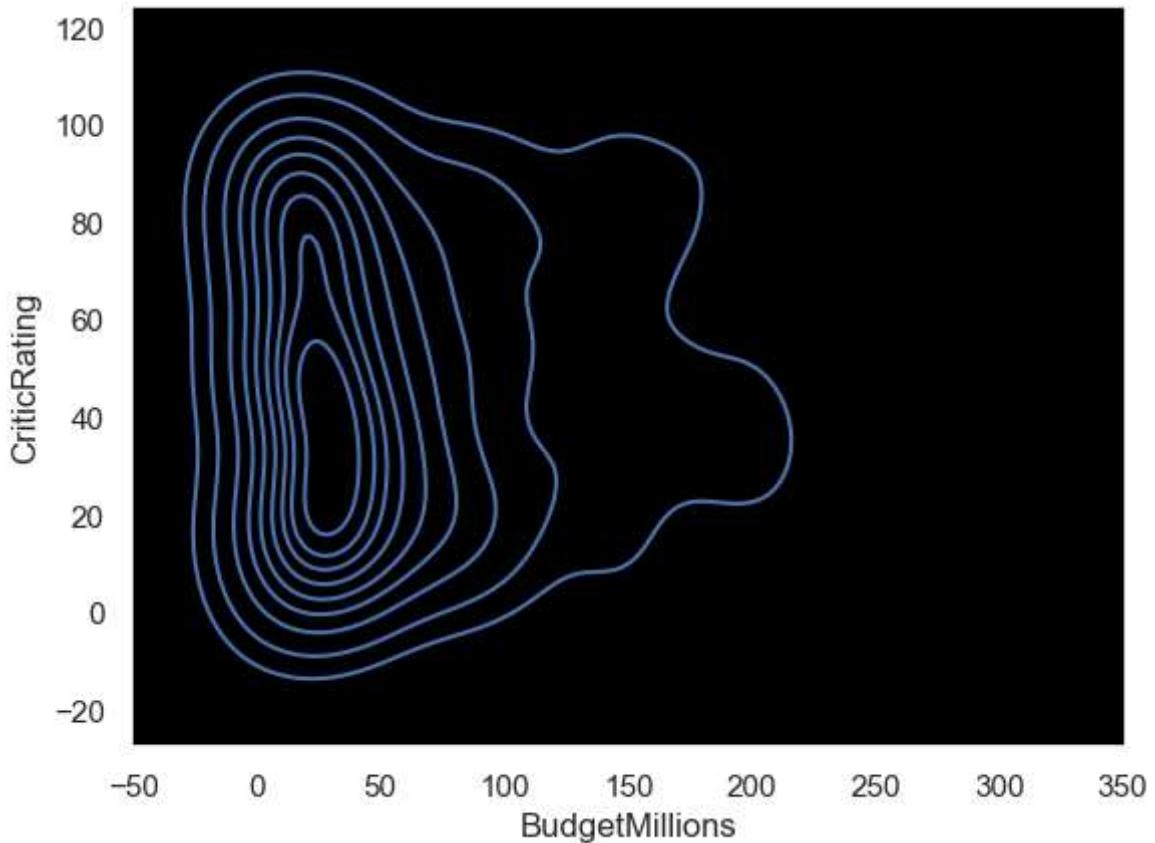
```
In [73]: k1 = sns.kdeplot(x = movies.BudgetMillions, y = movies.AudienceRating, shade = Tru
```



```
In [74]: sns.set_style('dark')
k1 = sns.kdeplot(x = movies.BudgetMillions, y = movies.AudienceRating)
```

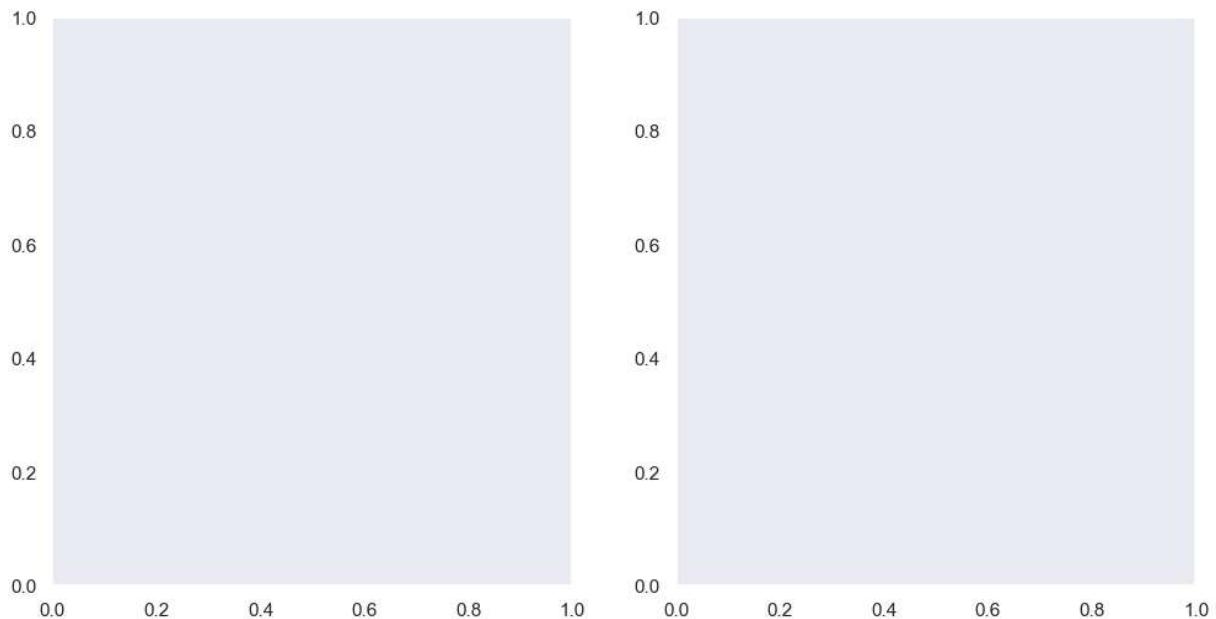


```
In [96]: k2 = sns.kdeplot(x = movies.BudgetMillions, y = movies.CriticRating)
```



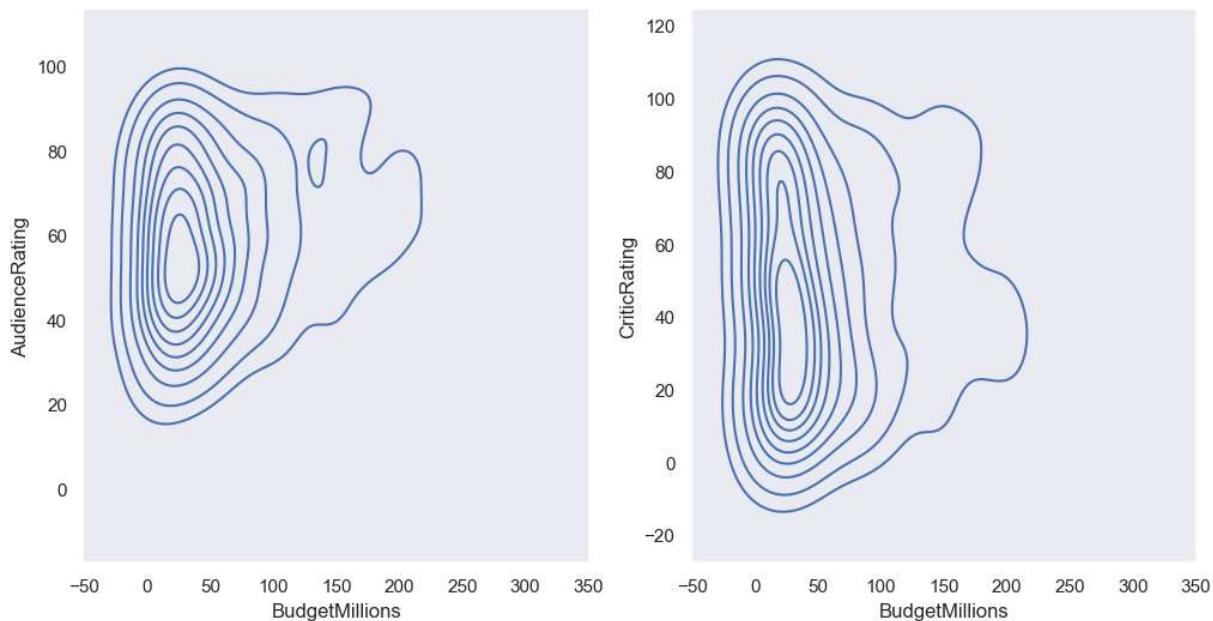
subplots

```
In [76]: f, ax = plt.subplots(1, 2, figsize = (12, 6))
```



```
In [77]: # In new version use arguments with keywords because "positional argument follows k"
```

```
In [78]: f, axes = plt.subplots(1, 2, figsize = (12, 6))
k1 = sns.kdeplot(x = movies.BudgetMillions, y = movies.AudienceRating, ax = axes[0])
k2 = sns.kdeplot(x = movies.BudgetMillions, y = movies.CriticRating, ax = axes[1])
```

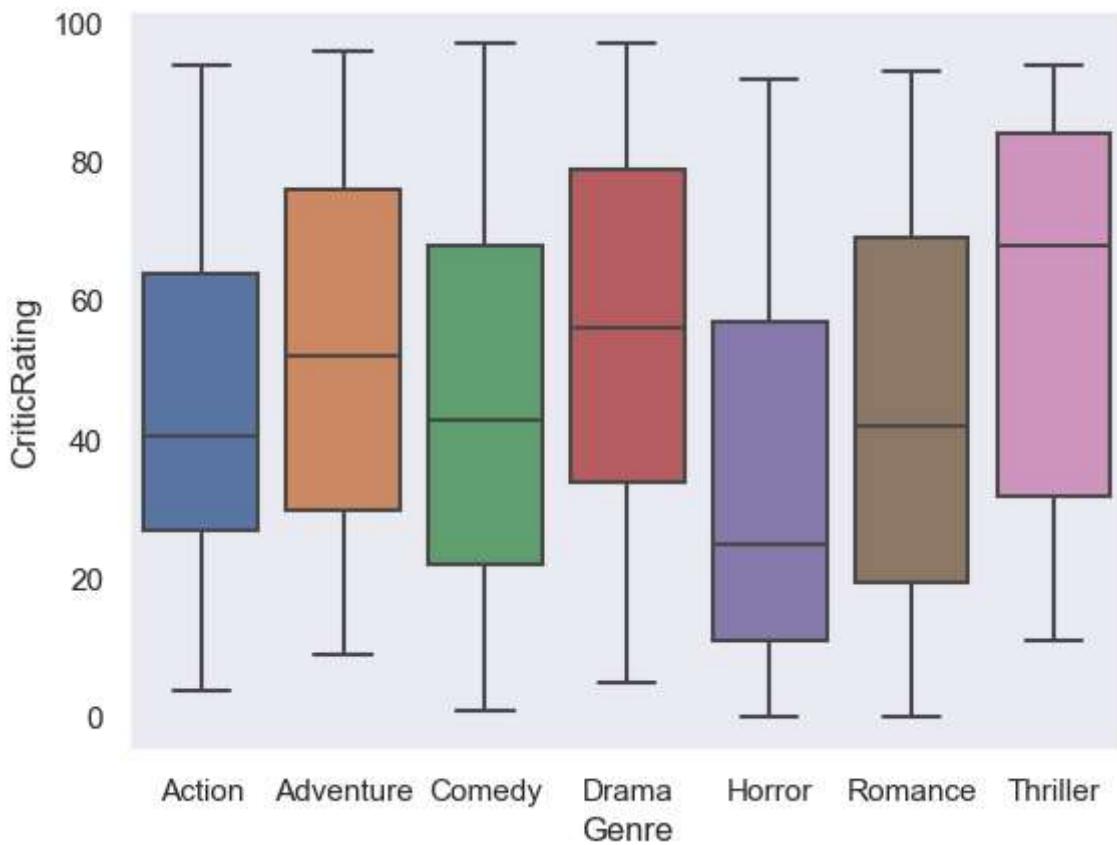


```
In [79]: axes
```

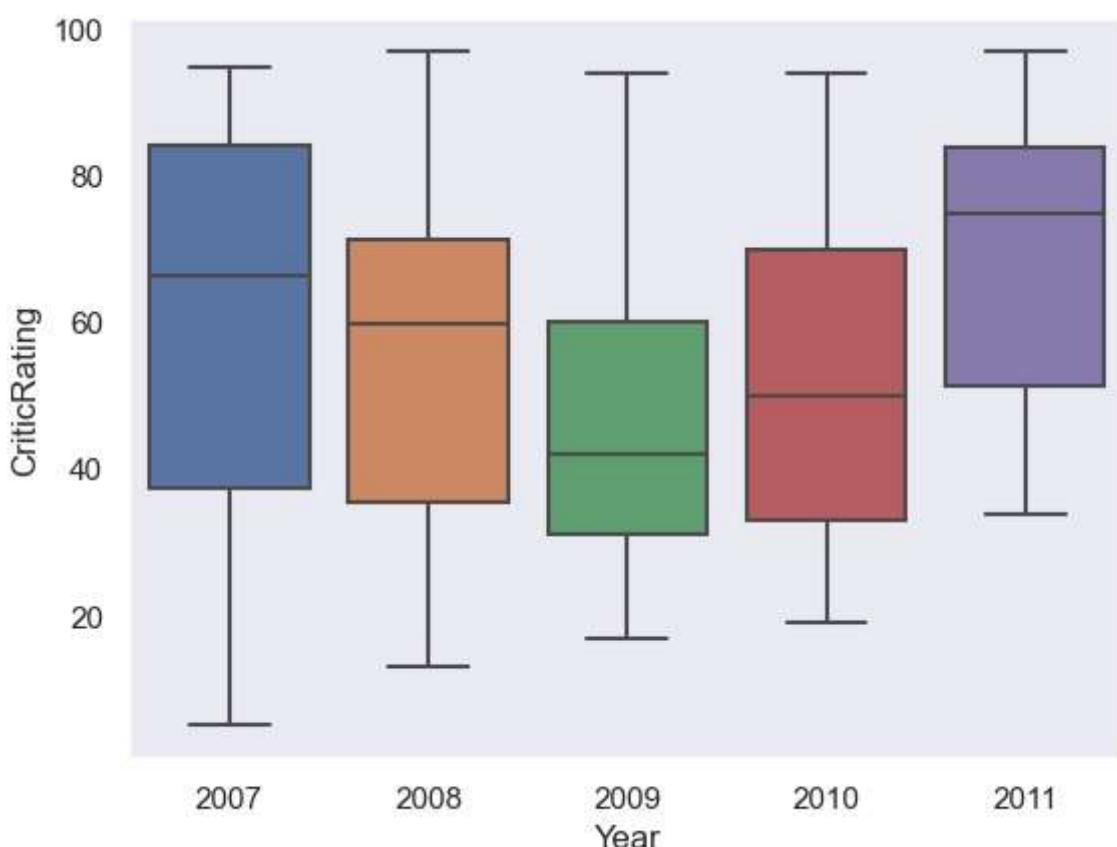
```
Out[79]: array([<Axes: xlabel='BudgetMillions', ylabel='AudienceRating'>,
   <Axes: xlabel='BudgetMillions', ylabel='CriticRating'>],
  dtype=object)
```

box plots

```
In [80]: w = sns.boxplot(data=movies, x = 'Genre', y = 'CriticRating')
# here we are finding critical rating in 'all movies' category
```

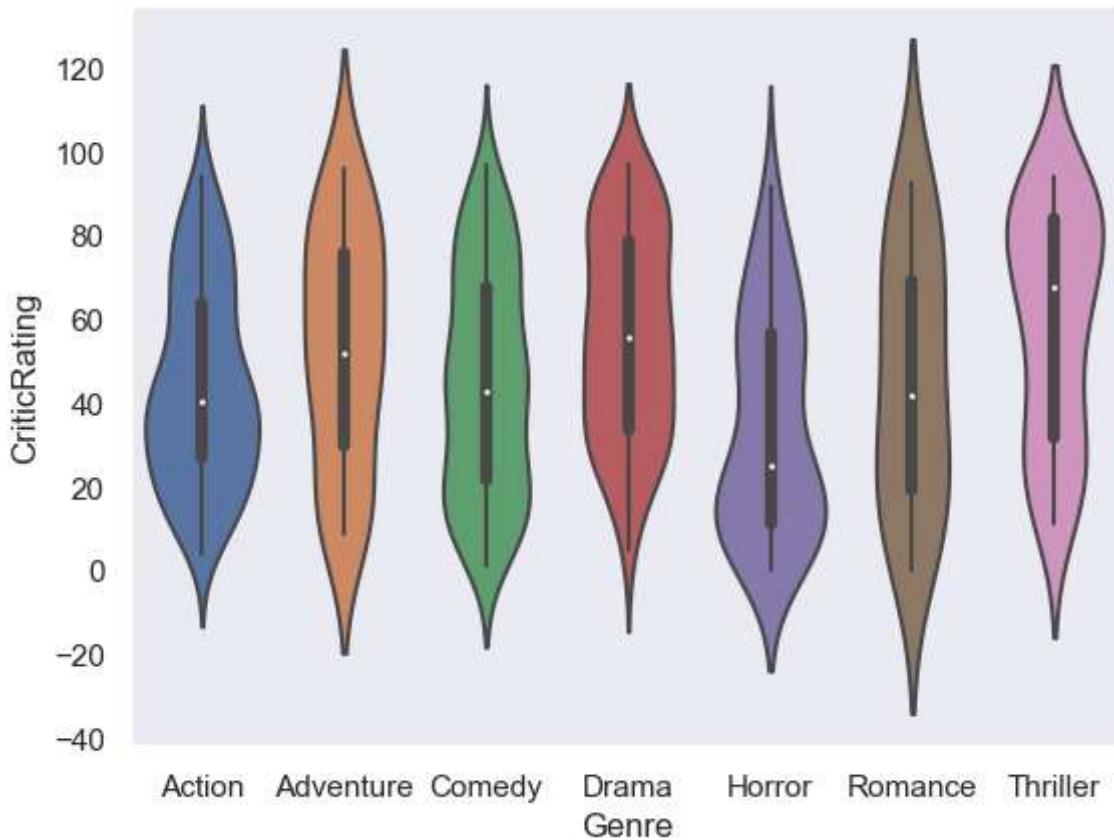


```
In [81]: w1 = sns.boxplot(data=movies[movies.Genre == 'Drama'], x = 'Year', y = 'CriticRating'  
# here we are finding critical rating in 'movies of 5 years'
```

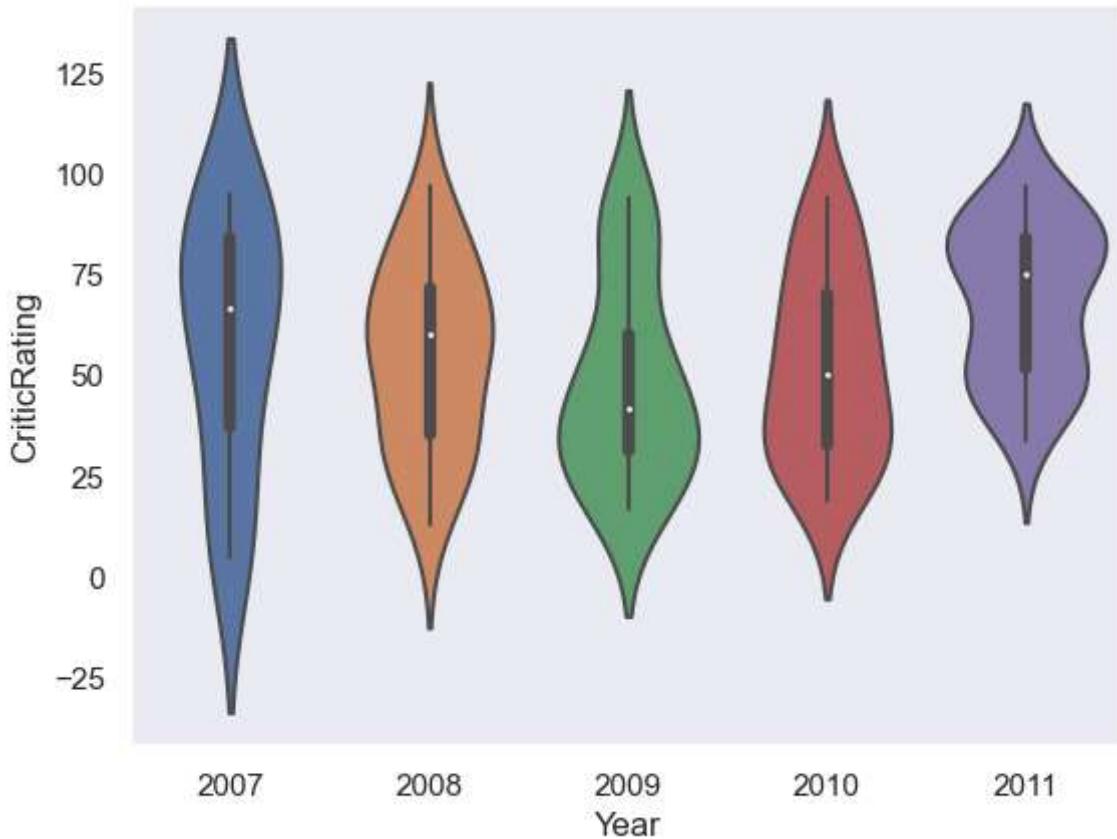


violin plot

```
In [82]: z = sns.violinplot(data = movies, x = 'Genre', y = 'CriticRating')
```

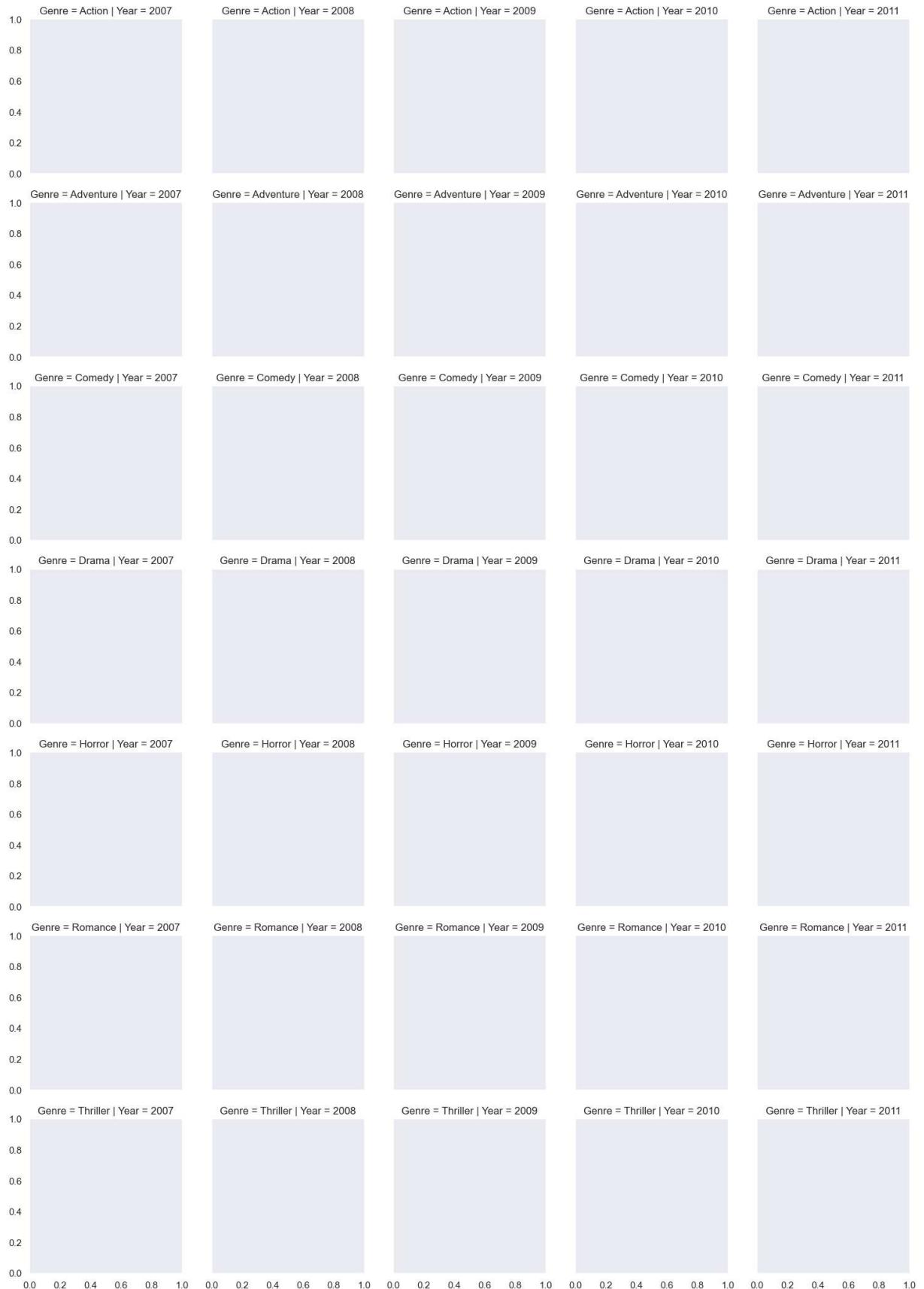


```
In [83]: z1 = sns.violinplot(data = movies[movies.Genre == 'Drama'], x = 'Year', y = 'CriticRating')
```



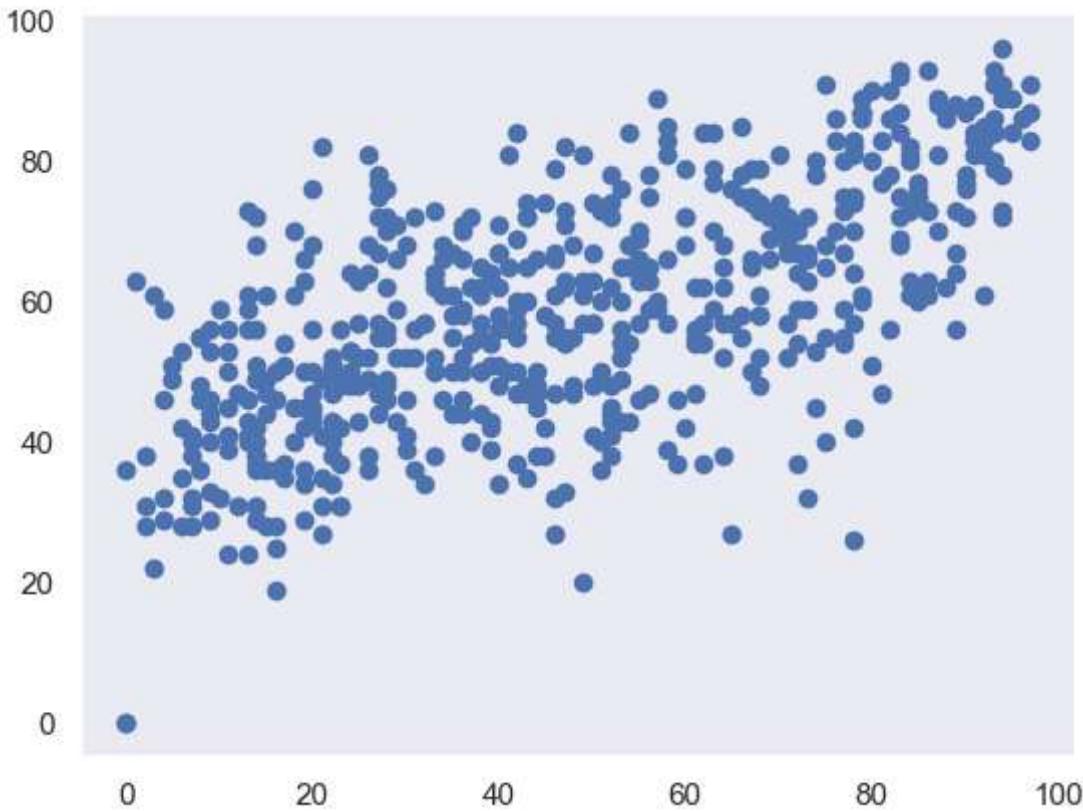
Facet Grid

```
In [84]: g = sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre') #kind of su
```

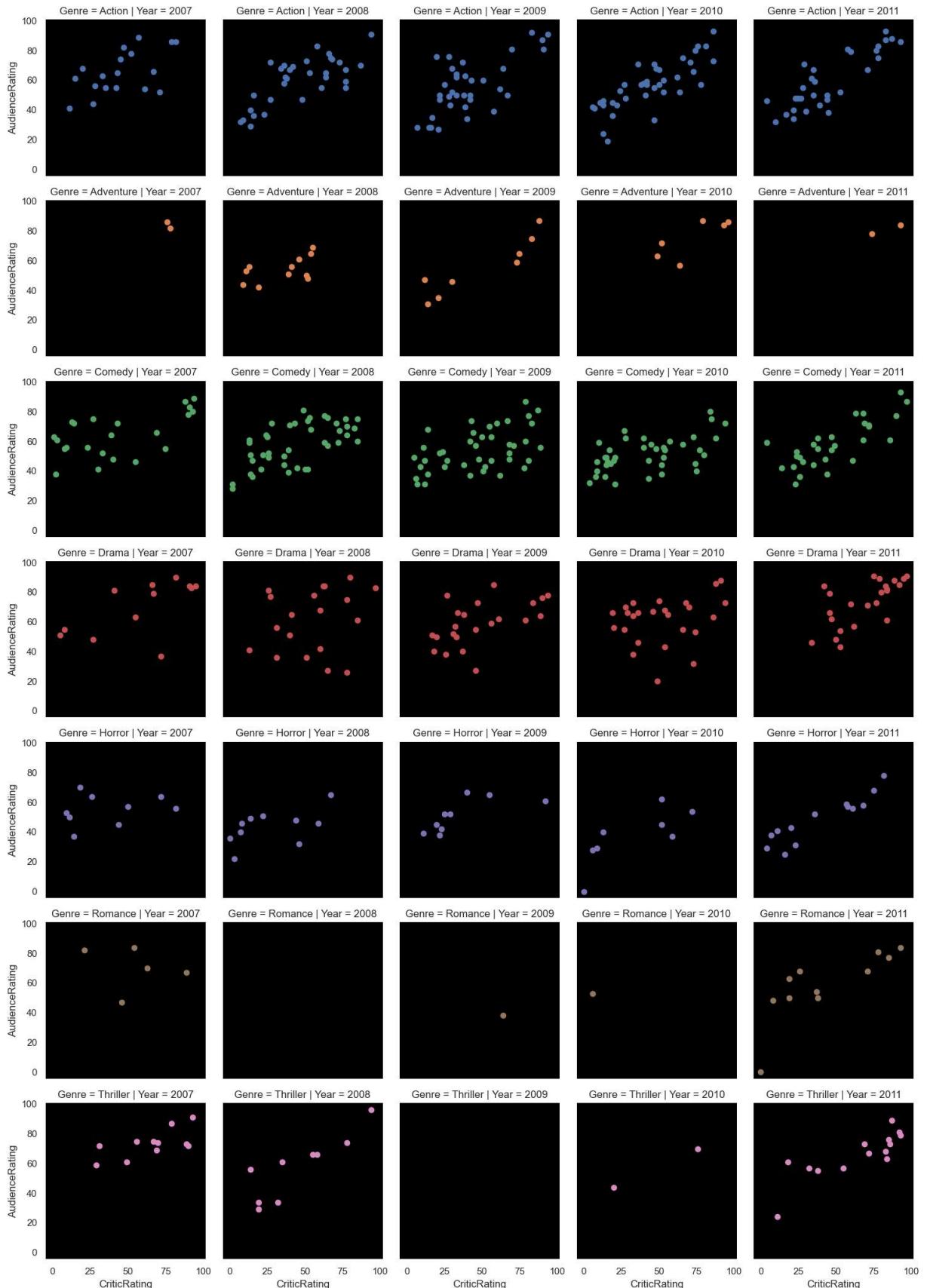


```
In [85]: plt.scatter(x = movies.CriticRating, y = movies.AudienceRating)
```

```
Out[85]: <matplotlib.collections.PathCollection at 0x1f49dd2baf0>
```



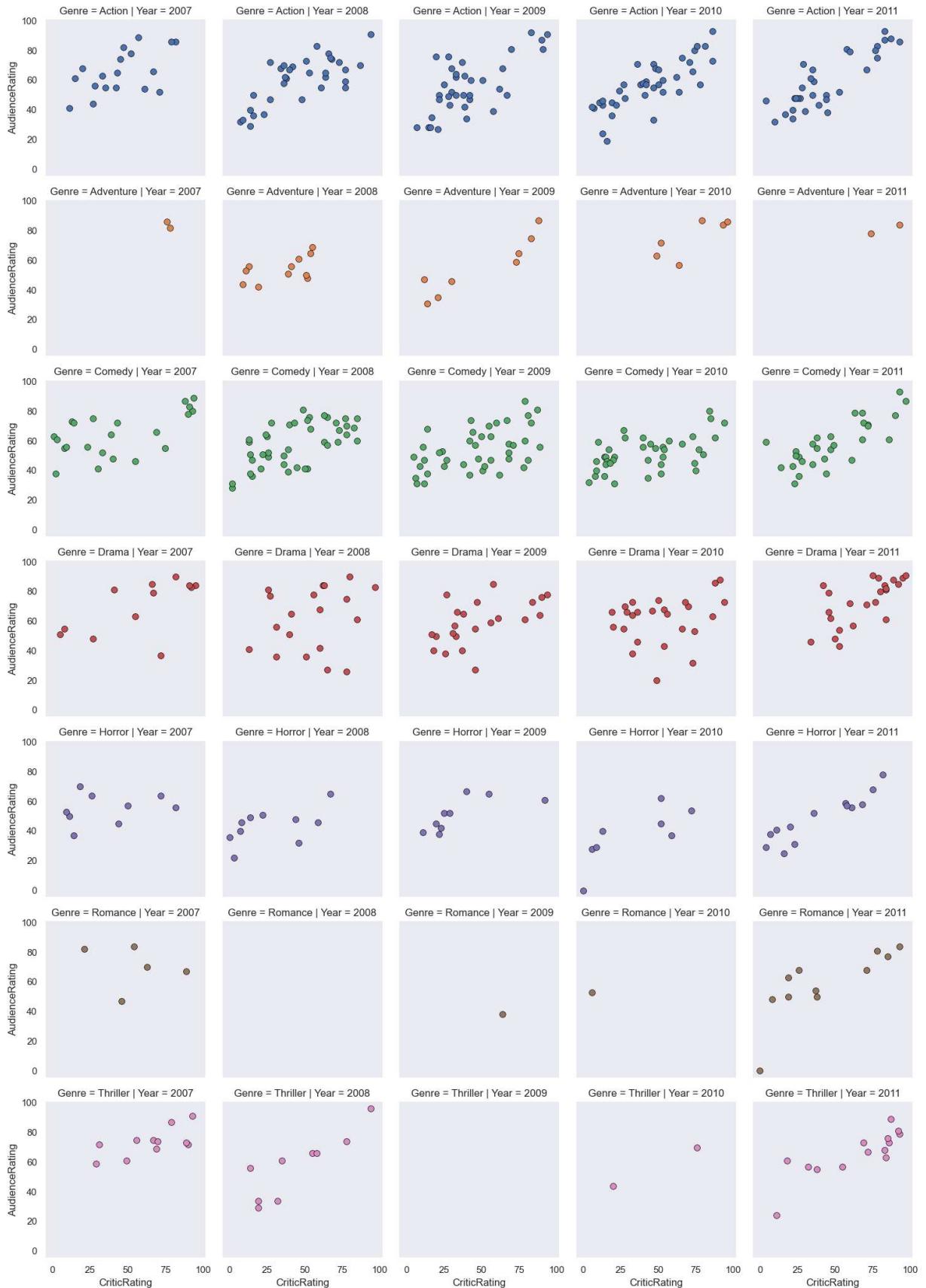
```
In [95]: g = sns.FacetGrid(movies, row = 'Genre', col = 'Year', hue = 'Genre')
g.map(plt.scatter, 'CriticRating', 'AudienceRating')
# g = g.map(plt.scatter, x = movies.CriticRating, y = movies.AudienceRating) # do
# as you can see scatterplots are mapped in facetgrid
```



```
In [87]: # you can populated any type of chart
g = sns.FacetGrid(movies, row='Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.hist, 'BudgetMillions')
# as you can see scatterplots are mapped in facetgrid
```



```
In [88]: g = sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
kws = dict(s=50, linewidth=0.5, edgecolor='black')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating', **kws )
# scatterplots are mapped in facetgrid
```



```
In [89]: sns.set_style('darkgrid')
f, axes = plt.subplots(2, 2, figsize = (15, 15))
k1 = sns.kdeplot(x = movies.BudgetMillions, y = movies.AudienceRating,ax=axes[0,0])
k2 = sns.kdeplot(x = movies.BudgetMillions, y = movies.CriticRating,ax = axes[0,1])
```

```

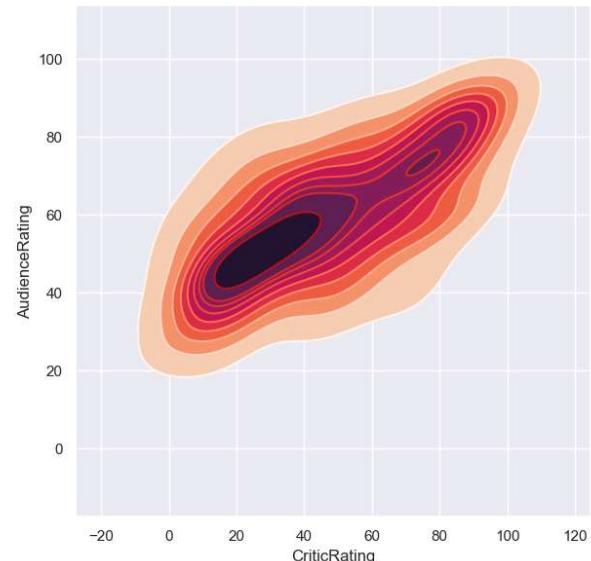
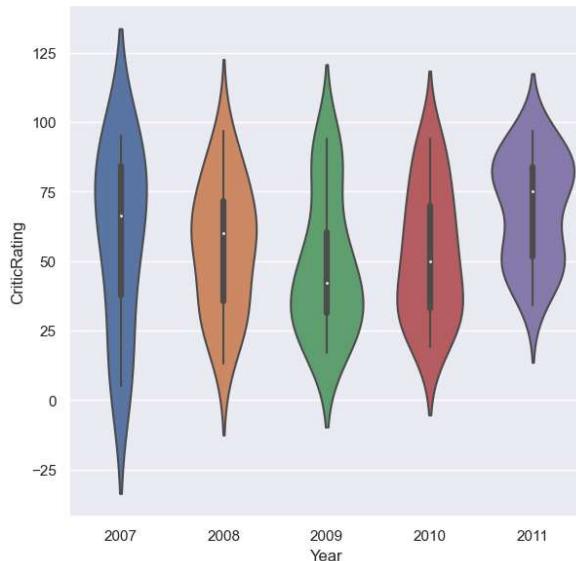
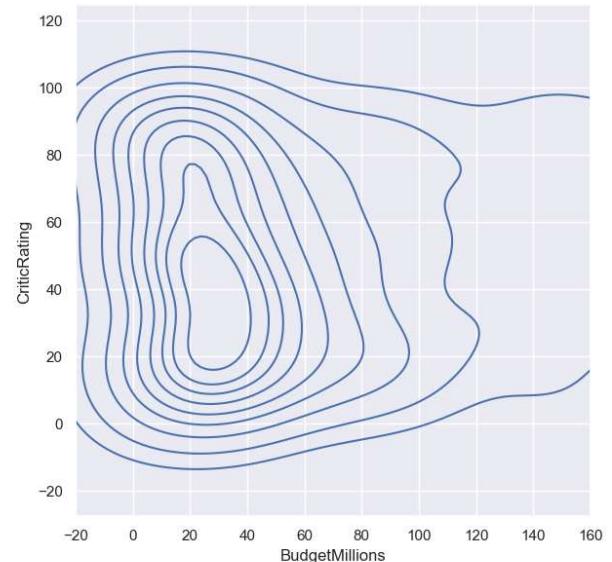
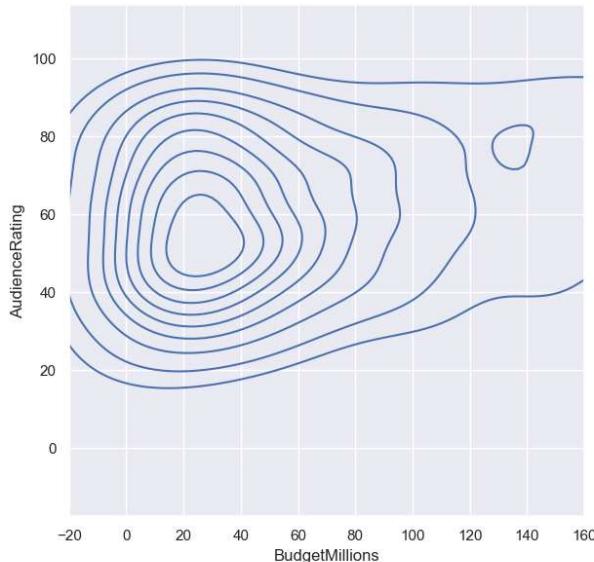
k1.set(xlim = (-20, 160))
k2.set(xlim = (-20, 160))

z1 = sns.violinplot(data = movies[movies.Genre == 'Drama'], x = 'Year', y = 'CriticRating', inner='quartile', palette='magma')
z2 = sns.violinplot(data = movies[movies.Genre == 'Drama'], x = 'Year', y = 'AudienceRating', inner='quartile', palette='viridis')

k4 = sns.kdeplot(x = movies.CriticRating, y = movies.AudienceRating, shade = True, shade_lowest=True, ax=axes[1,0])
k4b = sns.kdeplot(x = movies.CriticRating, y = movies.AudienceRating, cmap='Reds', ax=axes[1,1])

plt.show()

```



```

In [90]: # How can you style your dashboard using different color map
sns.set_style('dark', {'axes.facecolor': 'black'})
f, axes = plt.subplots(2, 2, figsize = (15, 15))

# Plot[0, 0]
k1 = sns.kdeplot(x = movies.BudgetMillions, y = movies.AudienceRating, \
                  shade = True, shade_lowest=True,\ 
                  ax=axes[0,0])
k1b = sns.kdeplot(x = movies.BudgetMillions, y = movies.AudienceRating, cmap='cool', 
                  ax=axes[0,1])

```

```
k1.set(xlim=(-20,160))

# Plot[0, 1]
k2 = sns.kdeplot(x = movies.BudgetMillions, y = movies.CriticRating, \
                  shade = True, shade_lowest=True, cmap='inferno', \
                  ax = axes[0,1])
k2b = sns.kdeplot(x = movies.BudgetMillions,y = movies.CriticRating, \
                   cmap = 'cool', ax = axes[0,1])
k2.set(xlim=(-20,160))

# Plot[1, 0]
z = sns.violinplot(data=movies[movies.Genre=='Drama'], \
                     x='Year', y = 'CriticRating', ax=axes[1,0])

# Plot[1, 1]
k4 = sns.kdeplot(x = movies.CriticRating, y = movies.AudienceRating, \
                  shade = True,shade_lowest=False,cmap='Blues_r', \
                  ax=axes[1,1])
k4b = sns.kdeplot(x = movies.CriticRating, y = movies.AudienceRating, \
                   cmap='gist_gray_r',ax = axes[1,1])

plt.show()
```

