In [1]:
```
# MOVIE RATING ANALYTICS (ADVANCED VISULIZATION)

import pandas as pd
import os
```

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

Out[2]: 'C:\\Users\\chitt'

In [5]:
```
movies = pd.read_csv(r"D:\NIT Daily Task\4th\MOVIE RATINGS _ ADVANCE VISUALIZATI
```

In [7]:
```
movies
```

Out[7]:

	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 [9]:
```
len(movies)
```

Out[9]: 559

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

Out[11]:

	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 [13]:
```
movies.tail()
```

Out[13]:

	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 [15]:
```
movies.columns
```

Out[15]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

In [17]:
```
movies.columns = ['Film', 'Genre', 'CriticRating', 'AudienceRating','BudgetMilli
```

In [19]:
```
movies.head() # Removed spaces & % removed noise characters
```

Out[19]:

	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 [21]:
```
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 [23]:
```
movies.describe()
```

Out[23]:

	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

In [25]:
```
movies['Film']
#movies['Audience Ratings %']
```

Out[25]:
```
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 [27]:
```
movies.Film
```

Out[27]:
```
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 [29]:
```
movies.Film = movies.Film.astype('category')
```

In [31]:
```
movies.Film
```

Out[31]:
```
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 [33]:
```
movies.head()
```

Out[33]:

	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 [35]:
```
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    category
 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: category(1), int64(4), object(1)
memory usage: 43.6+ KB
```

In [37]:
```
movies.Genre = movies.Genre.astype('category')
movies.Year = movies.Year.astype('category')
```

In [39]:
```
movies.Genre
```

Out[39]:
```
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', 'R
omance', 'Thriller']
```

In [41]:
```
movies.Year
```

Out[41]:
```
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 [43]:
```
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    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 [45]:
```
movies.Genre.cat.categories
```

Out[45]:
```
Index(['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance',
       'Thriller'],
      dtype='object')
```

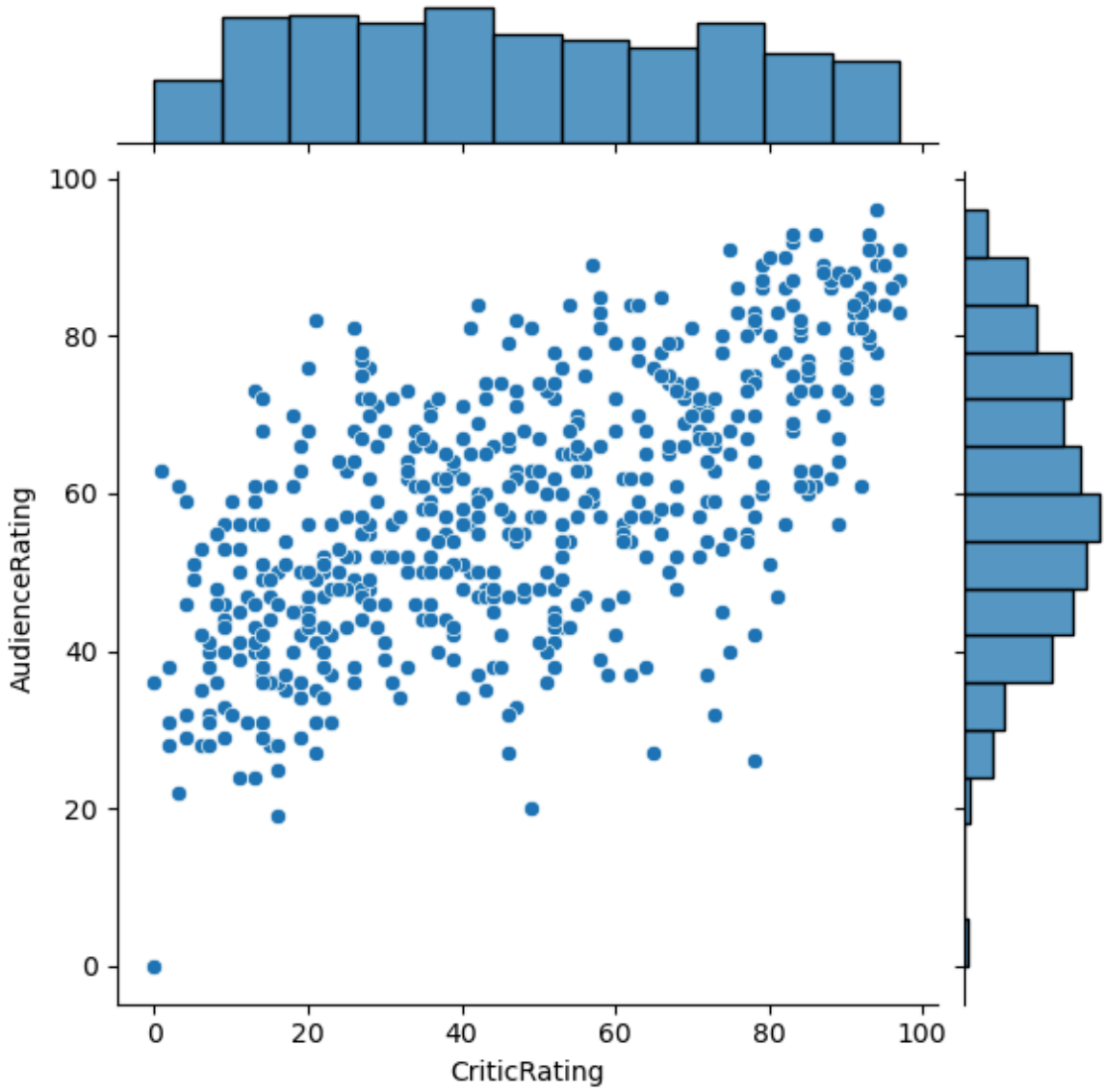
In [47]:
```
movies.describe()
```

Out[47]:

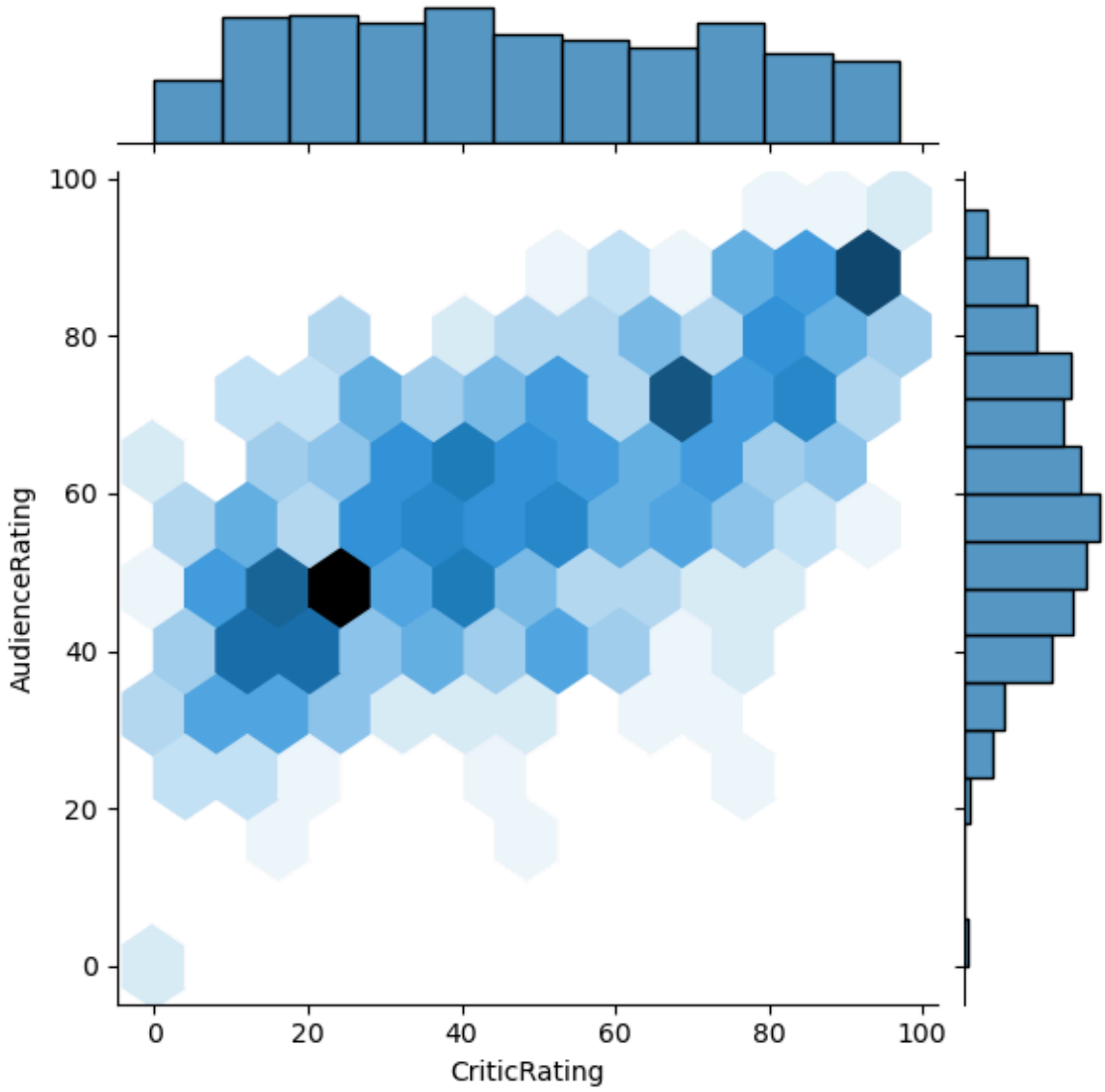
	CriticRating	AudienceRating	BudgetMillions
count	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136
std	26.413091	16.826887	48.731817
min	0.000000	0.000000	0.000000
25%	25.000000	47.000000	20.000000
50%	46.000000	58.000000	35.000000
75%	70.000000	72.000000	65.000000
max	97.000000	96.000000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

In [49]:
```
# How to working with joint plots

from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

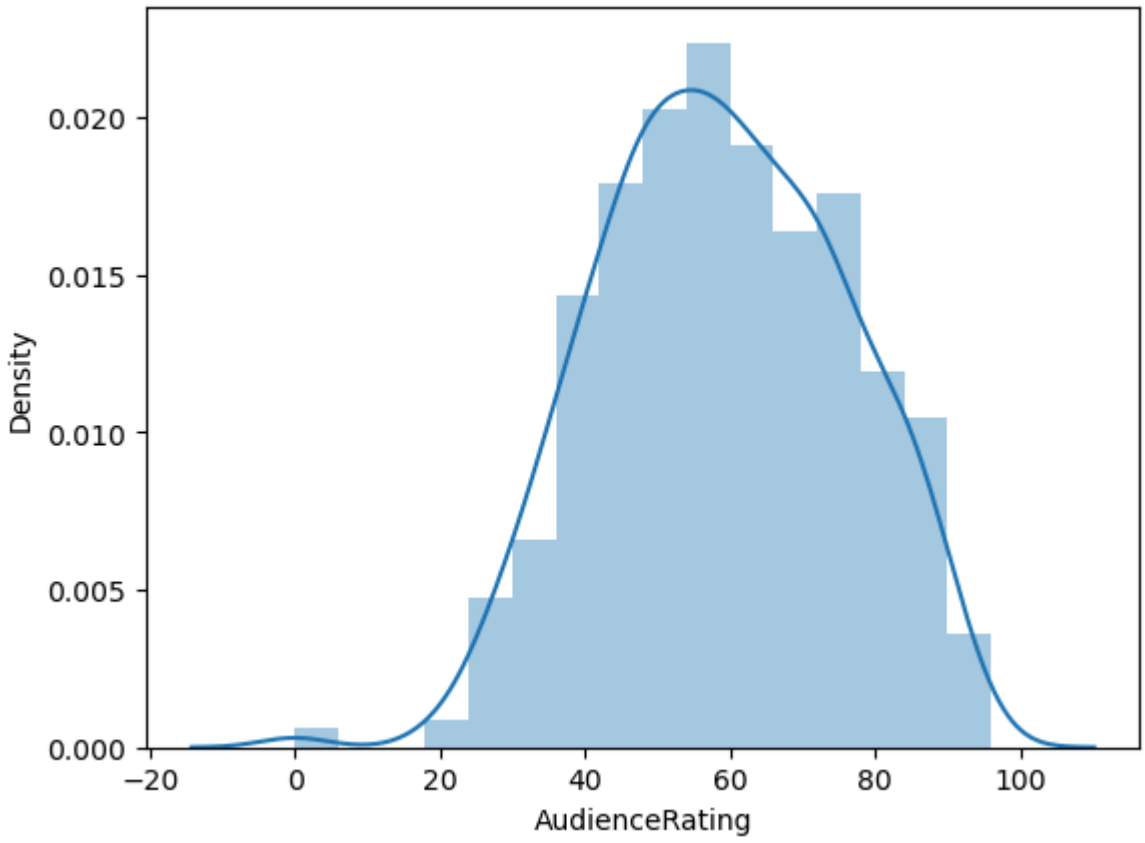
In [50]:
```
j = sns.jointplot( data = movies, x = 'CriticRating', y = 'AudienceRating')
```



In [51]:
```
j = sns.jointplot( data = movies, x = 'CriticRating', y = 'AudienceRating', kind
```

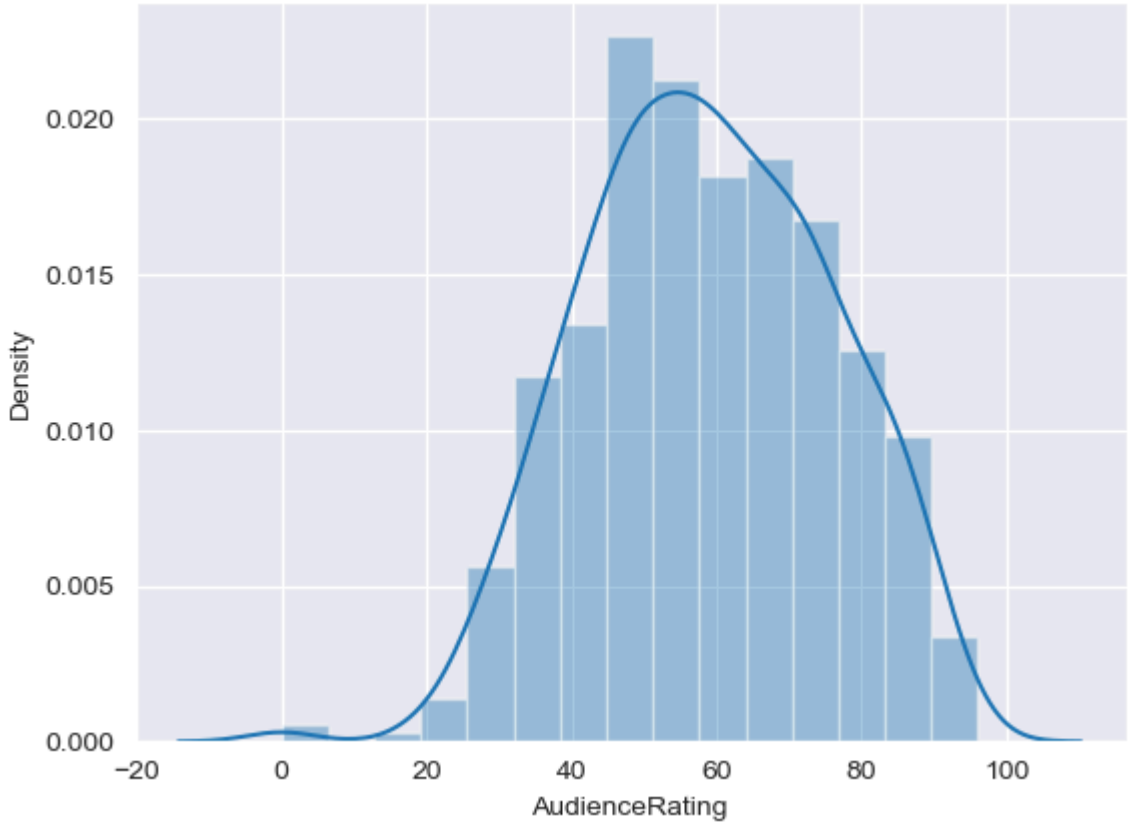


In [52]:
```
m1 = sns.distplot(movies.AudienceRating)
```

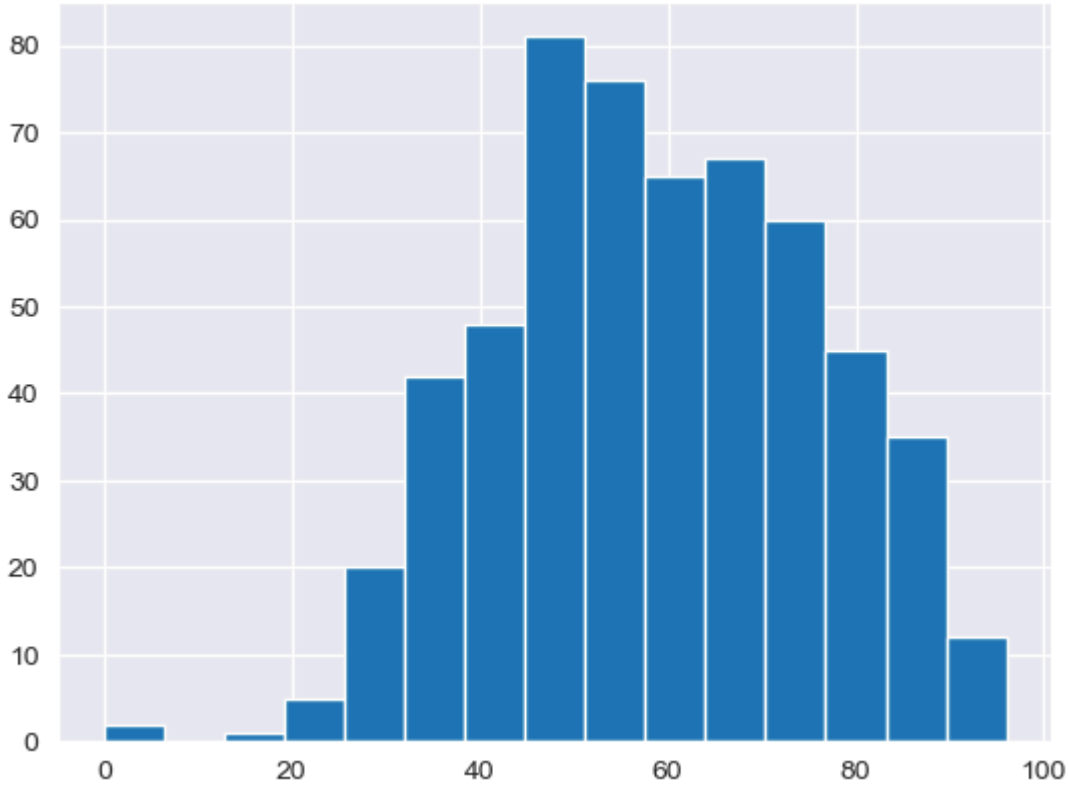


In [57]:
```
sns.set_style('darkgrid')
```

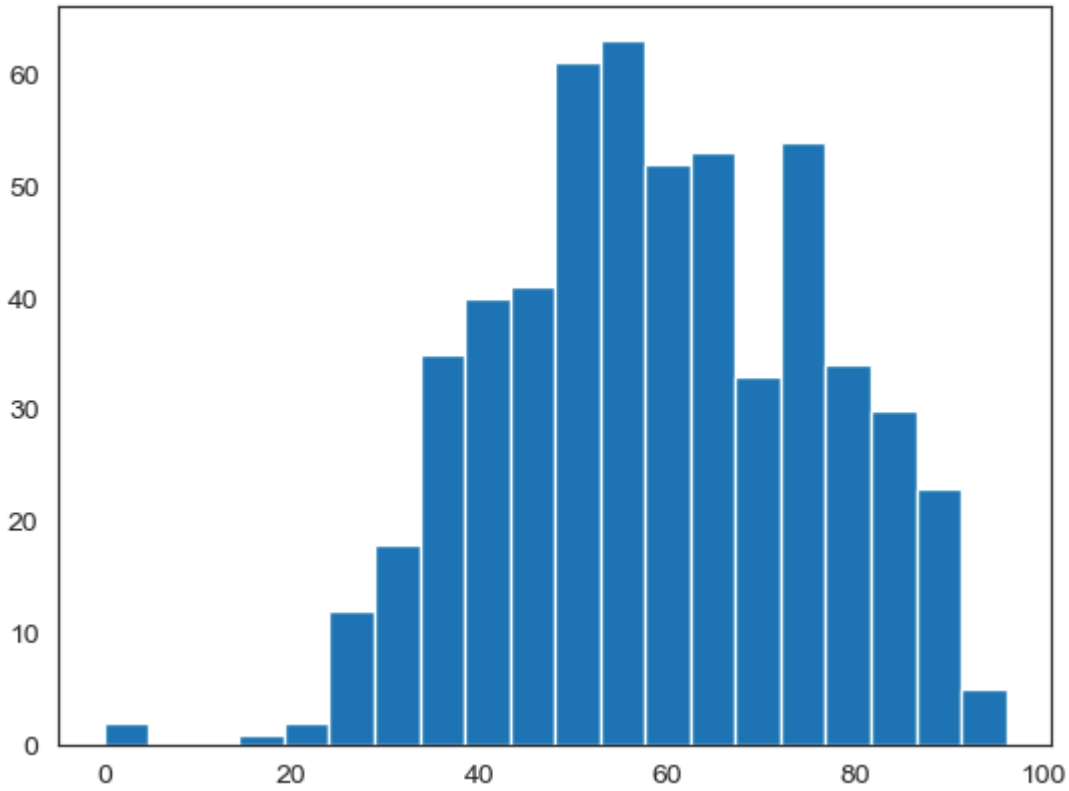
In [59]:
```
m2 = sns.distplot(movies.AudienceRating, bins = 15)
```



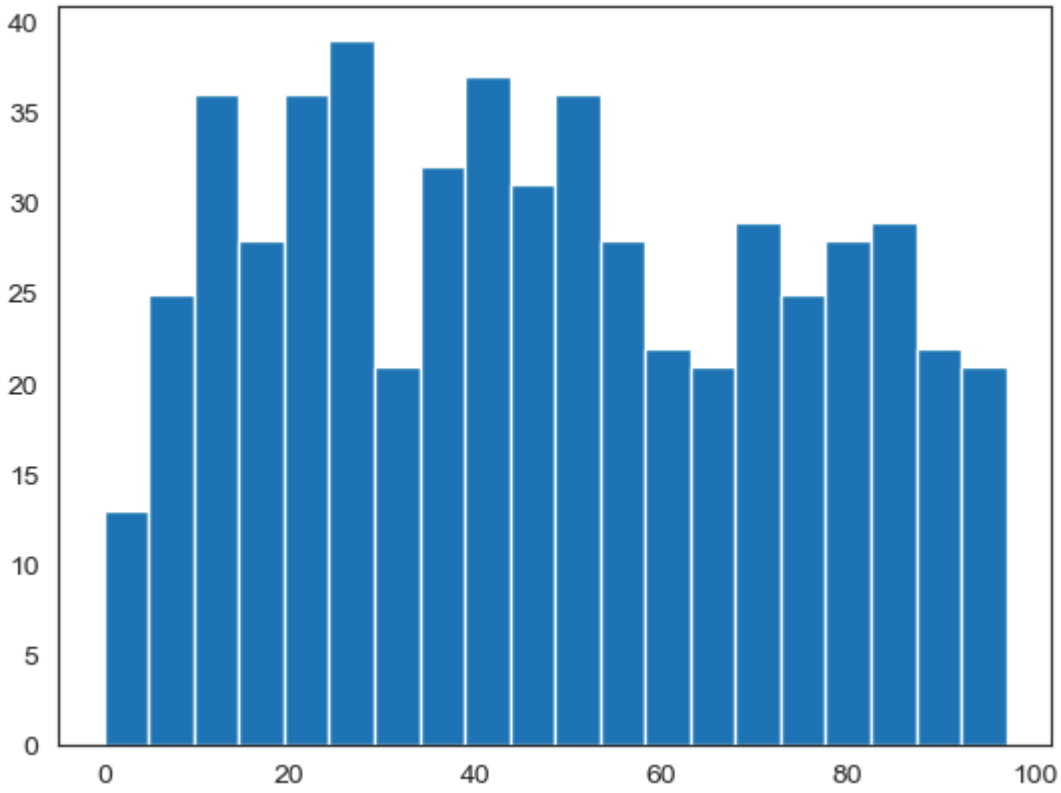
In [61]:
```
#sns.set_style('darkgrid')
n1 = plt.hist(movies.AudienceRating, bins=15)
```



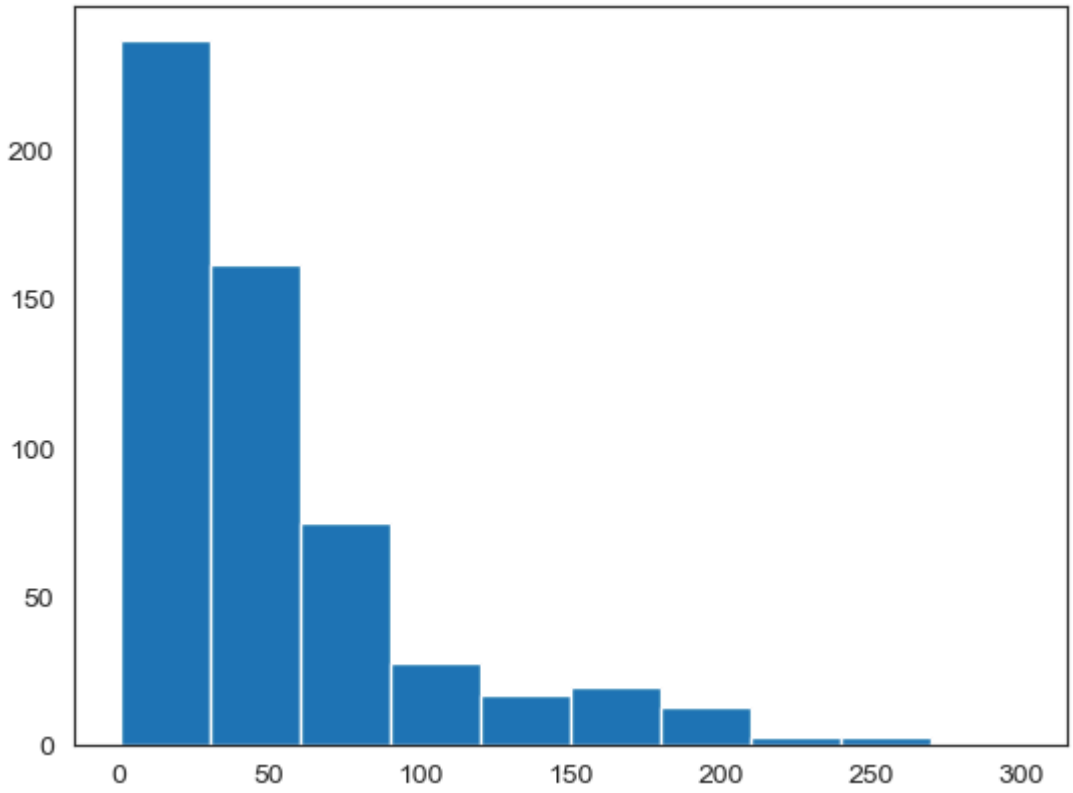
In [63]:
```
sns.set_style('white') #normal distribution & called as bell curve
n1 = plt.hist(movies.AudienceRating, bins=20)
```



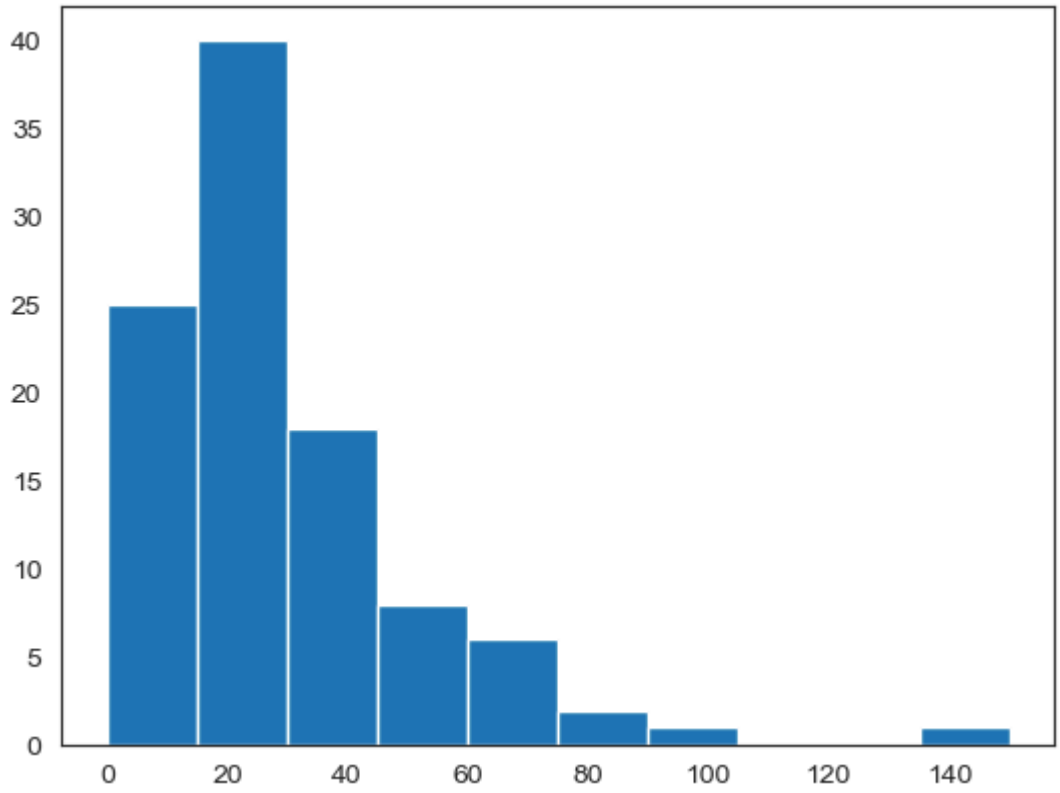
In [65]:
```
n1 = plt.hist(movies.CriticRating, bins=20) #uniform distribution
```



In [67]:
```
plt.hist(movies.BudgetMillions)
plt.show()
```



In [69]:
```
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions)
plt.show()
```



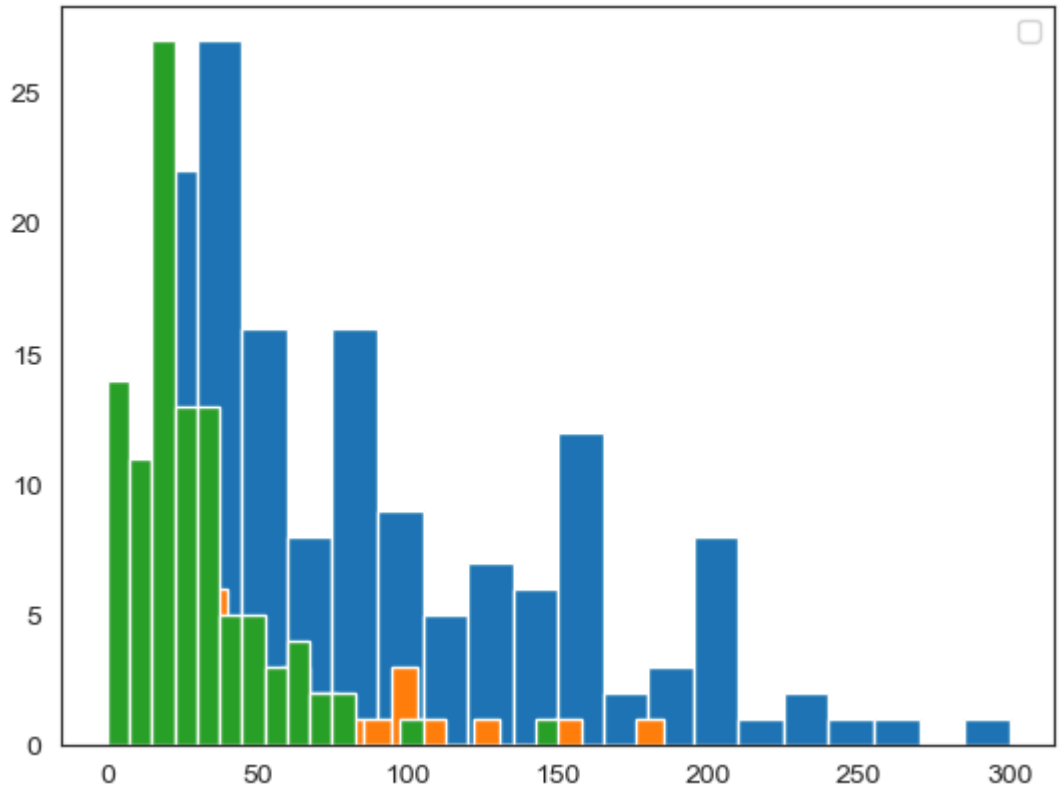
In [71]:
```
movies.head()
```

Out[71]:

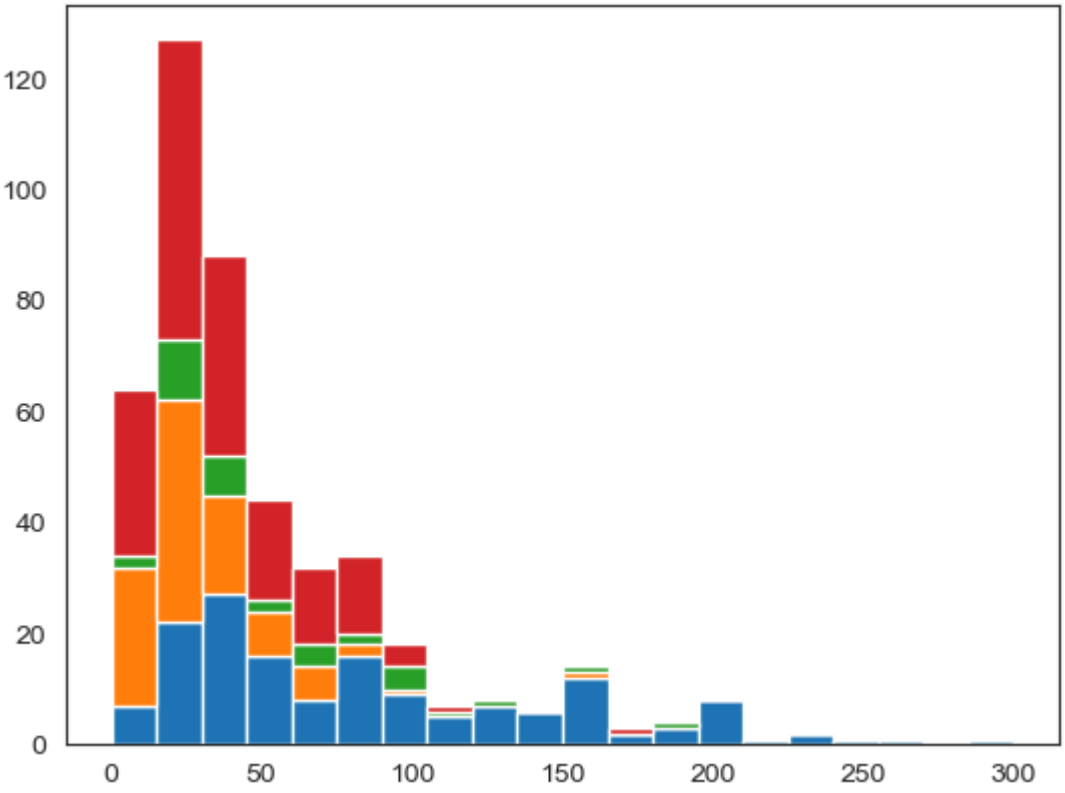
	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009

In [73]:
```
plt.hist(movies[movies.Genre == 'Action'].BudgetMillions, bins = 20)
plt.hist(movies[movies.Genre == 'Thriller'].BudgetMillions, bins = 20)
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions, bins = 20)
plt.legend()
plt.show()
```

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



In [75]:
```
plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,\
          movies[movies.Genre == 'Drama'].BudgetMillions, \
          movies[movies.Genre == 'Thriller'].BudgetMillions, \
          movies[movies.Genre == 'Comedy'].BudgetMillions],
         bins = 20, stacked = True)
plt.show()
```

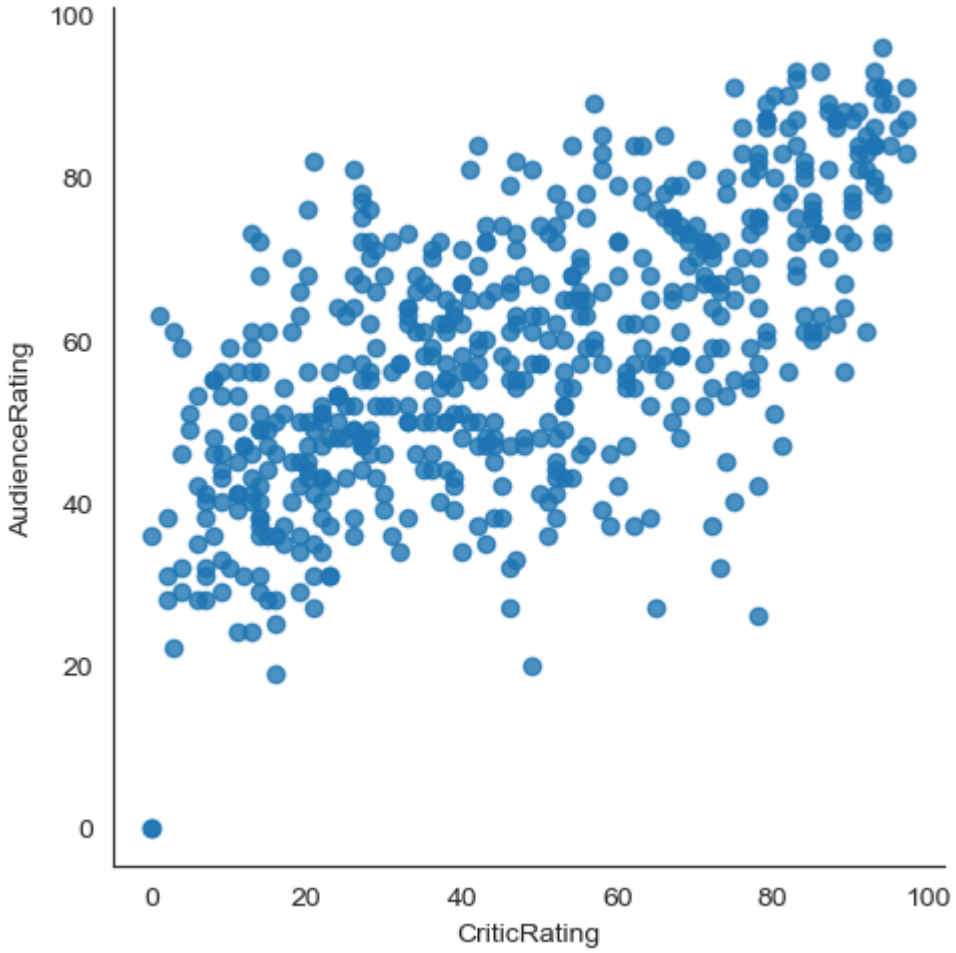


In [77]:
```
# if you have 100 categories you cannot copy & paste all the things
```

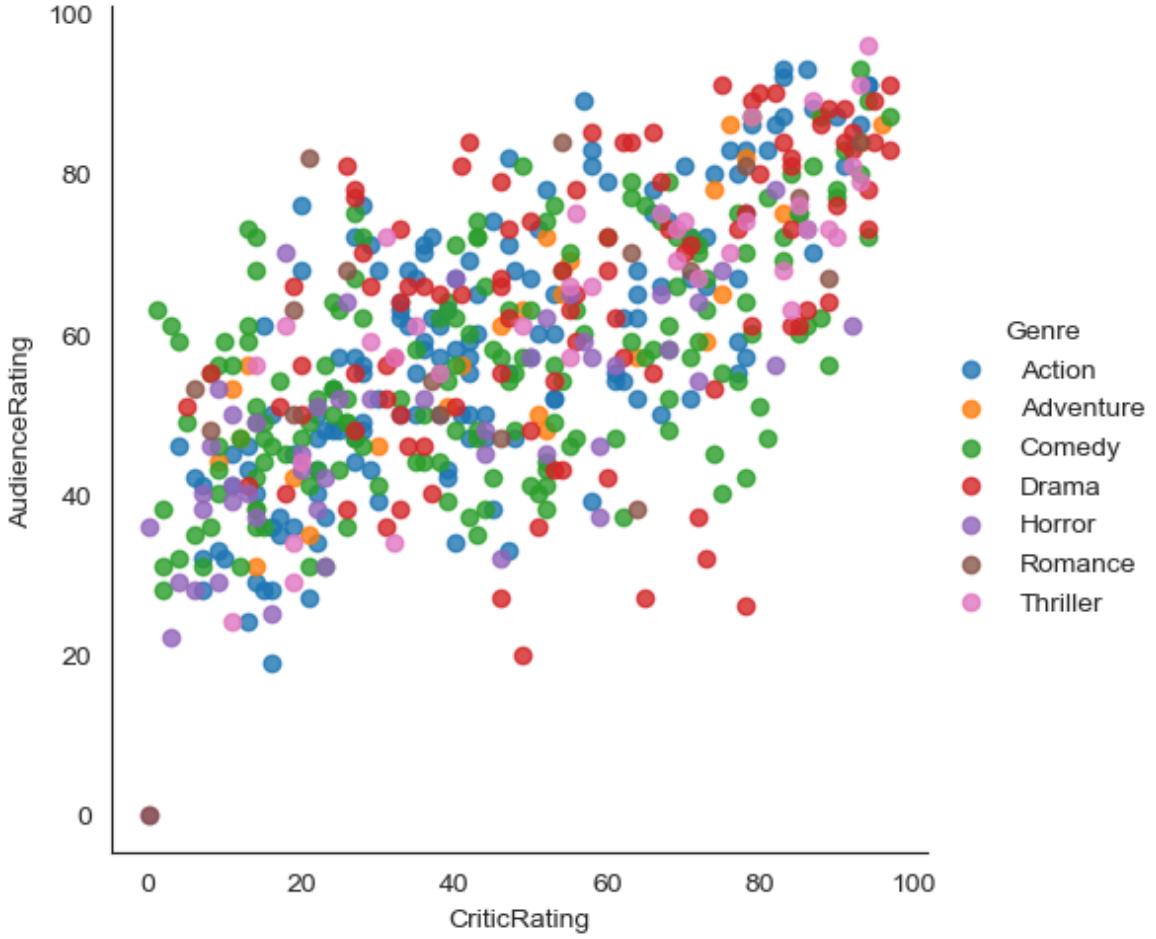
```
for gen in movies.Genre.cat.categories:
    print(gen)
```

```
Action
Adventure
Comedy
Drama
Horror
Romance
Thriller
```

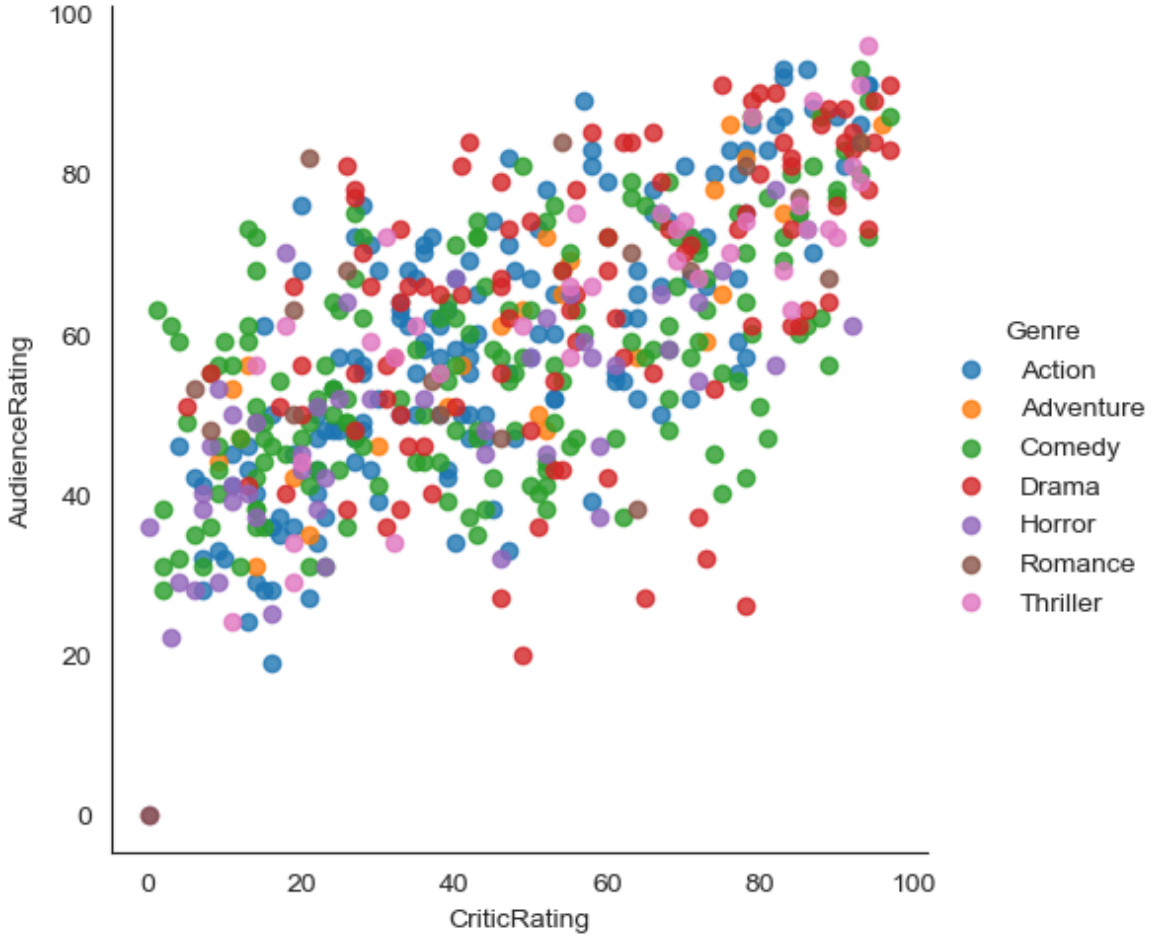
In [79]:
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False)
```



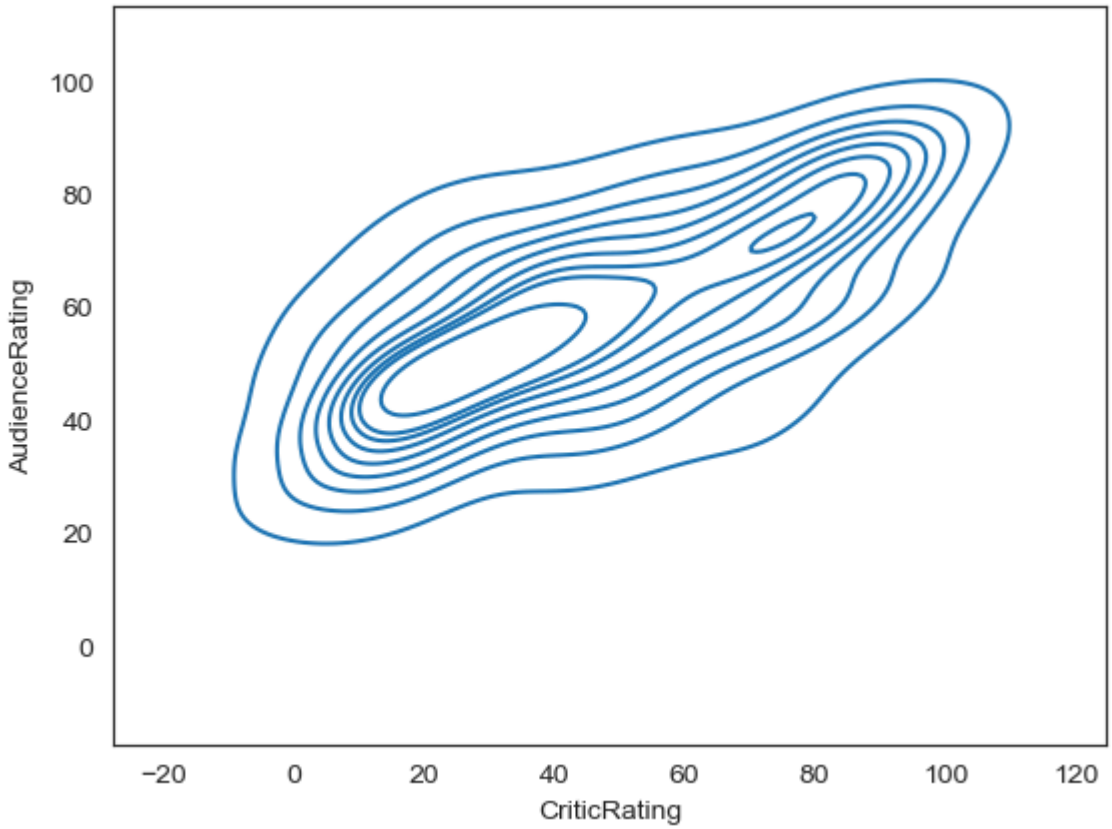
In [81]:
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False, hue = 'Genre')
```



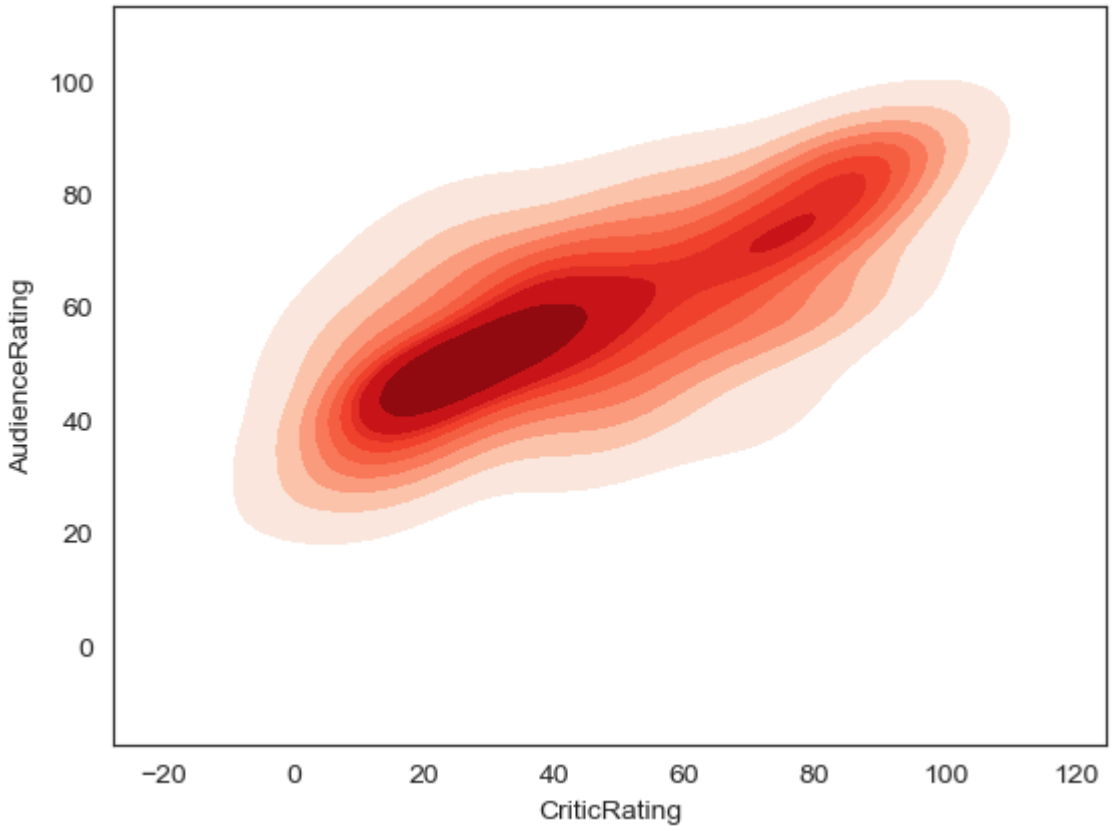
In [83]:
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False, hue = 'Genre',aspect=1)
```



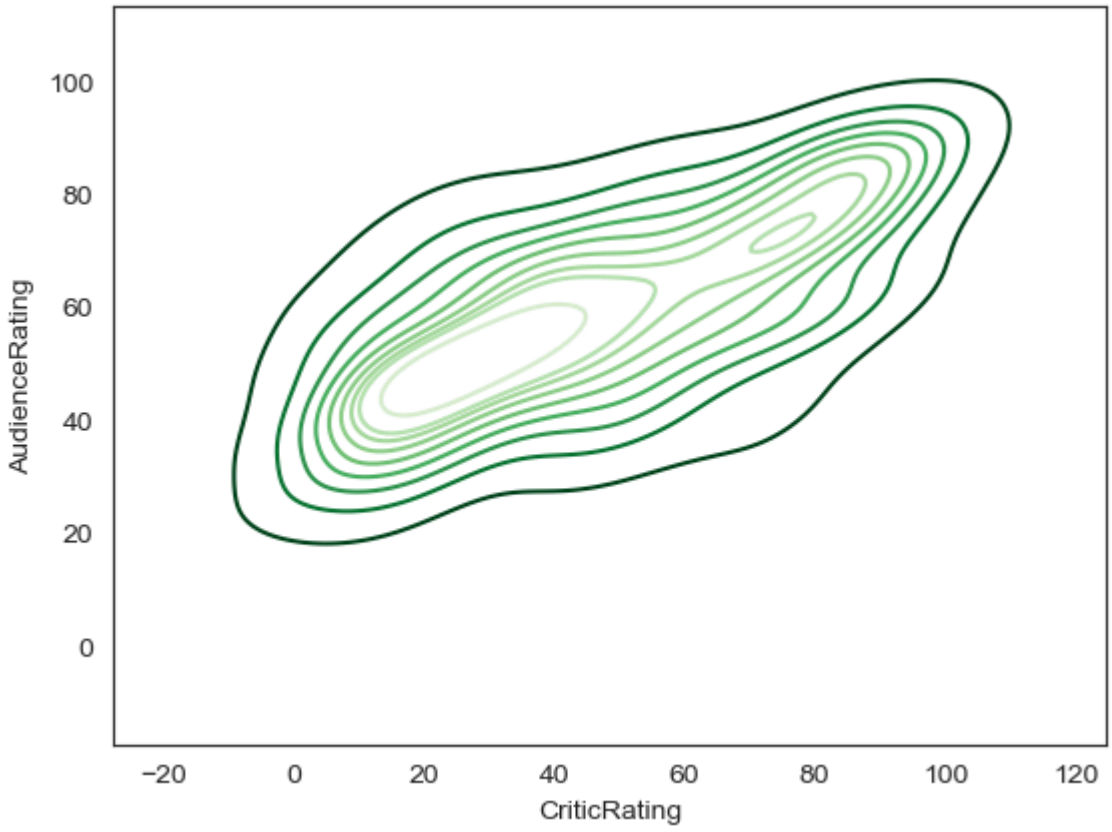
In [85]:
```
k1 = sns.kdeplot(data = movies,x=movies.CriticRating,y=movies.AudienceRating)
```



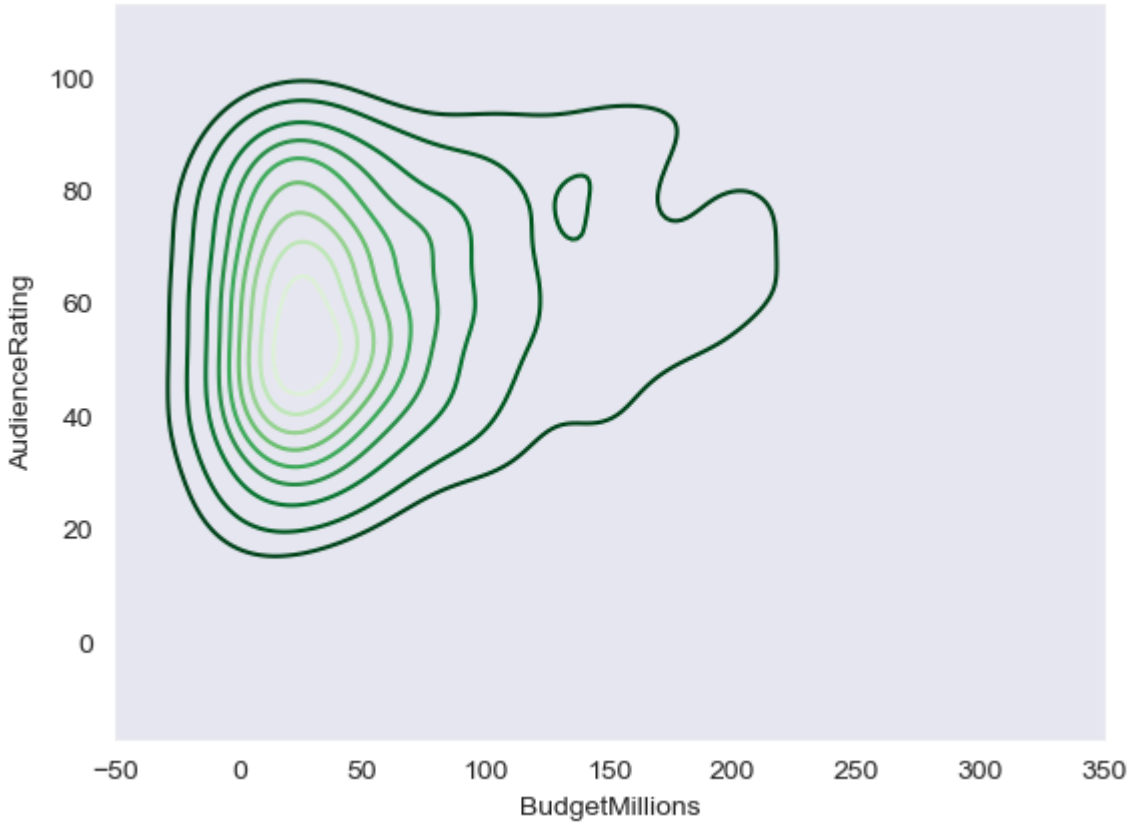
In [87]:
```
k1 = sns.kdeplot(x=movies.CriticRating,y=movies.AudienceRating,shade = True,shad
```



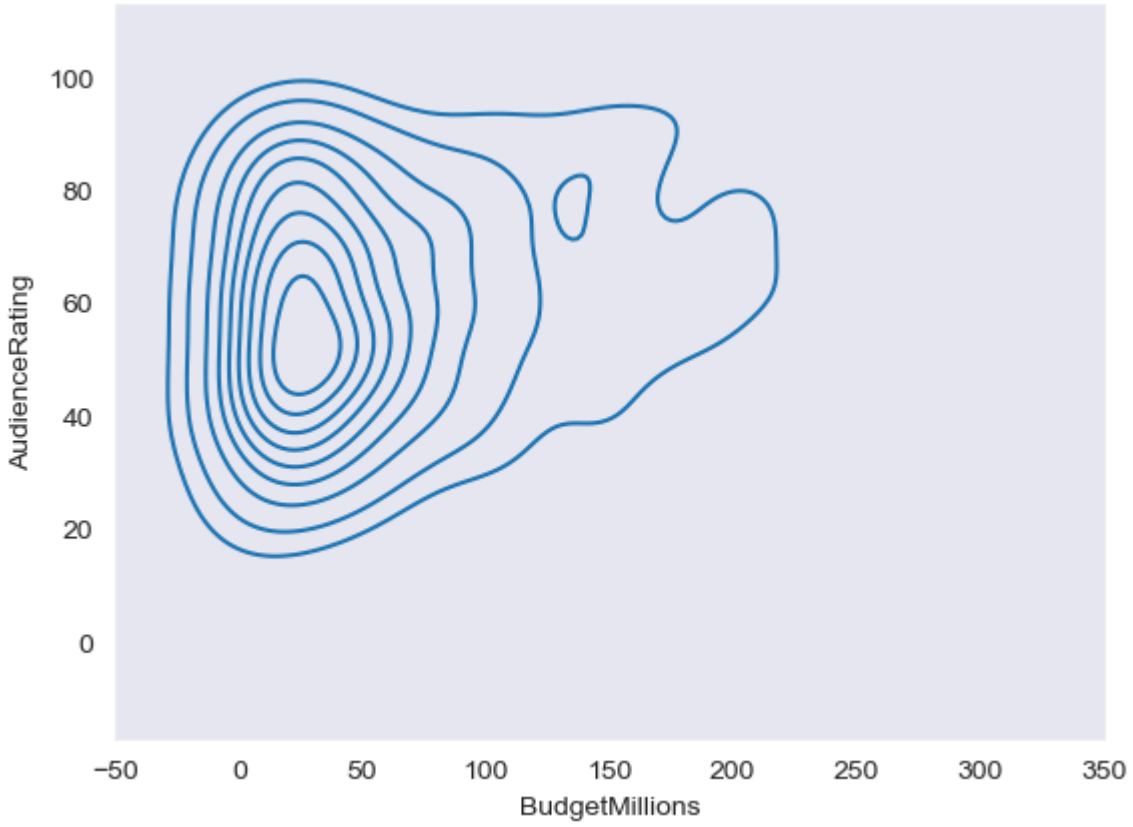
In [89]:
```
k2 = sns.kdeplot(x=movies.CriticRating,y=movies.AudienceRating,shade_lowest=Fals
```



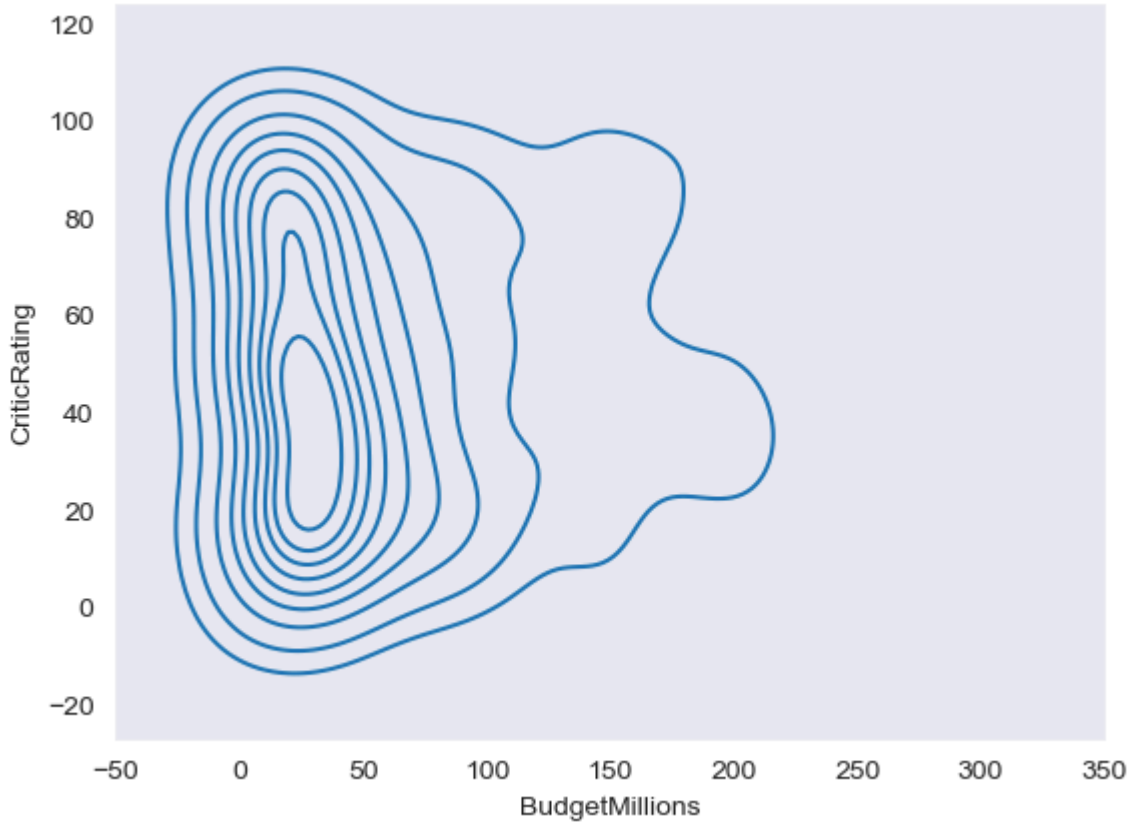
In [91]:
```
sns.set_style('dark')
k1 = sns.kdeplot(x=movies.BudgetMillions,y=movies.AudienceRating,shade_lowest=Fa
```



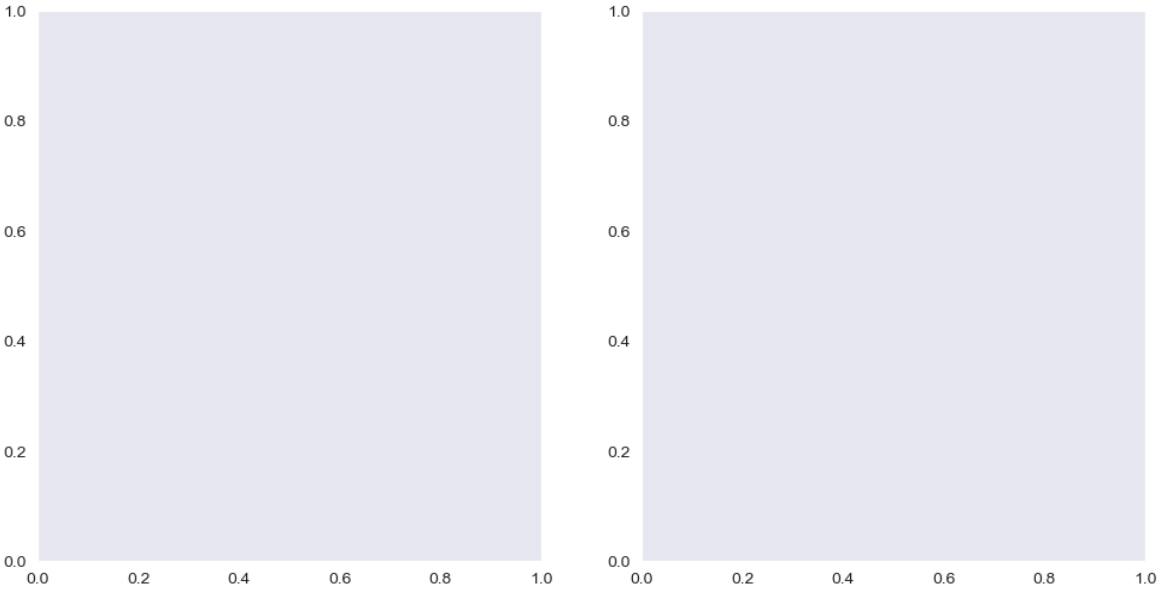
In [93]:
```
sns.set_style('dark')
k1 = sns.kdeplot(x=movies.BudgetMillions,y=movies.AudienceRating)
```



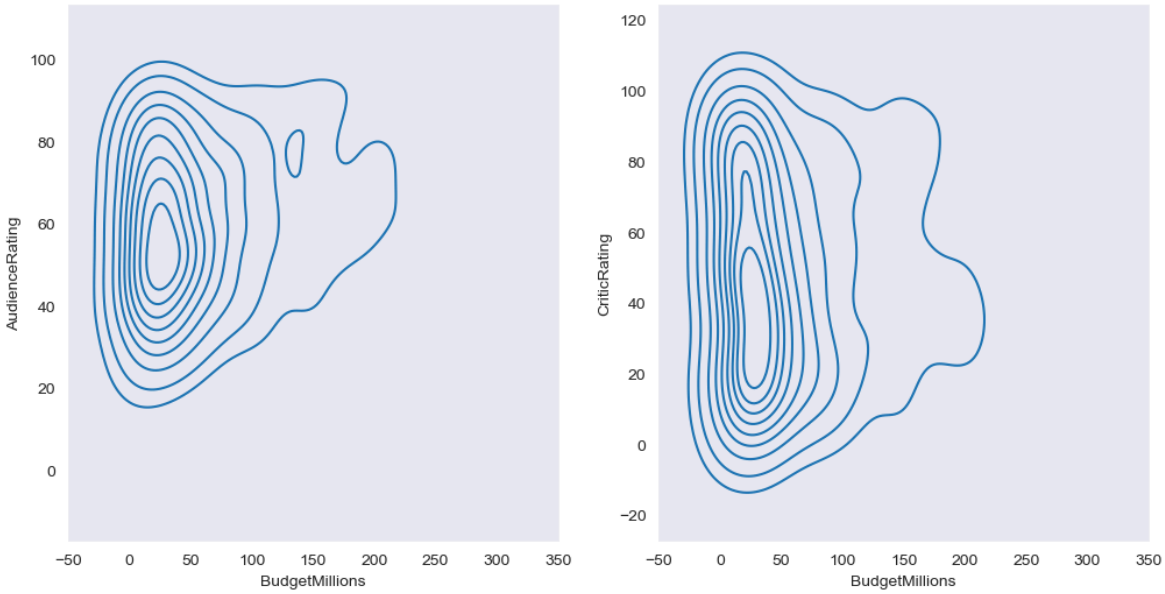
In [95]:
```
k2 = sns.kdeplot(x=movies.BudgetMillions,y=movies.CriticRating)
```



In [97]:
```
#subplots

f, ax = plt.subplots(1,2, figsize =(12,6))
#f, ax = plt.subplots(3,3, figsize =(12,6))
```



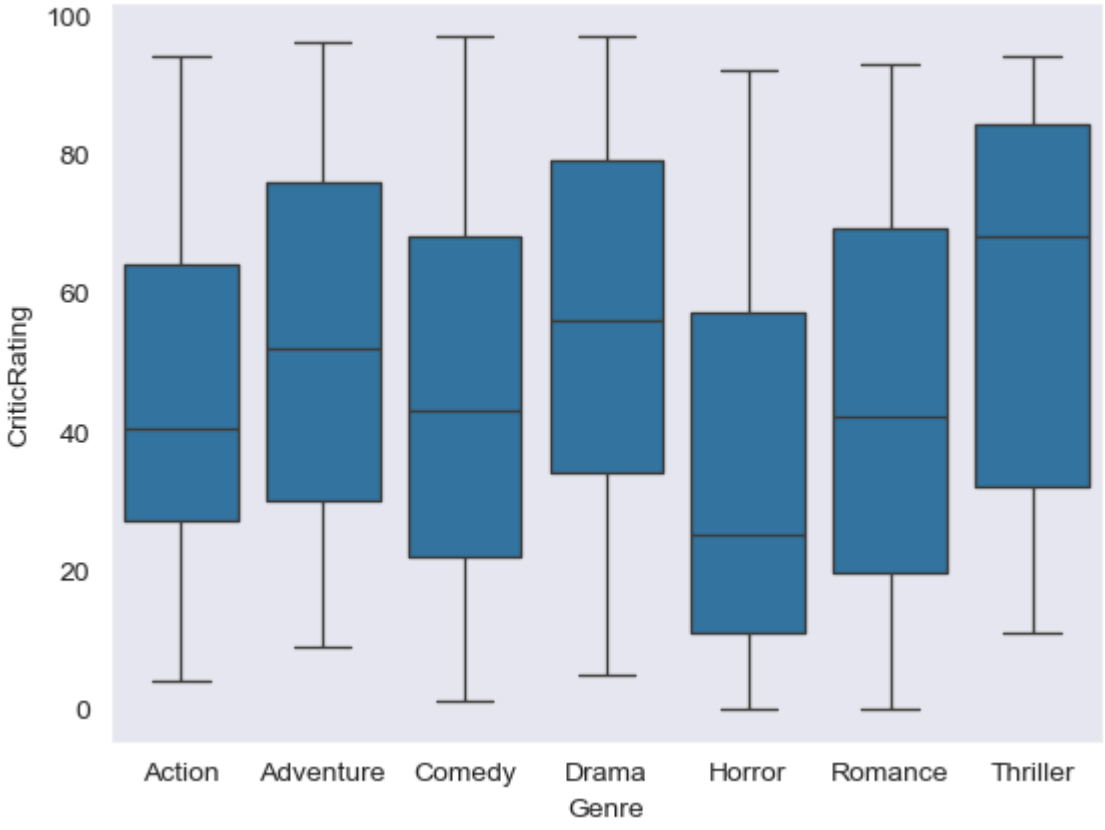
In [99]:
```
f, axes = plt.subplots(1,2, figsize =(12,6))

k1 = sns.kdeplot(x=movies.BudgetMillions,y=movies.AudienceRating,ax=axes[0])
k2 = sns.kdeplot(x=movies.BudgetMillions,y=movies.CriticRating,ax = axes[1])
```



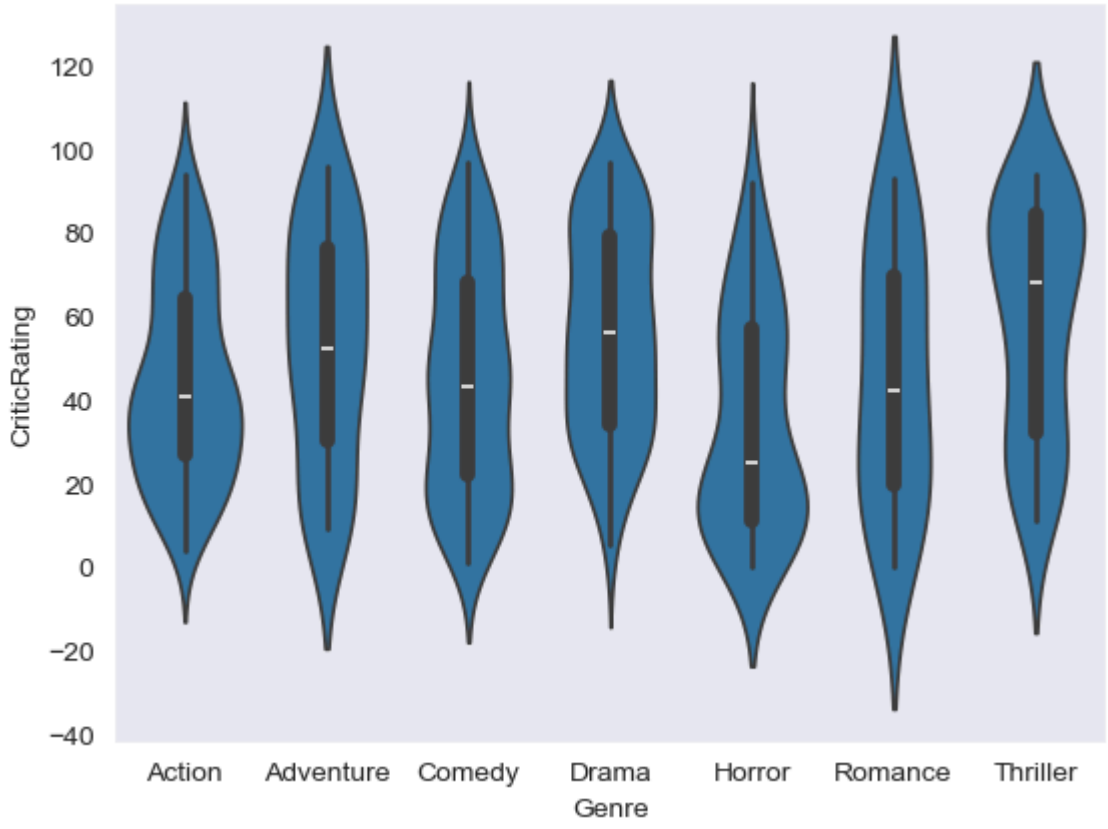
In [100…
```
axes
```

Out[100…
```
array([<Axes: xlabel='BudgetMillions', ylabel='AudienceRating'>,
       <Axes: xlabel='BudgetMillions', ylabel='CriticRating'>],
      dtype=object)
```

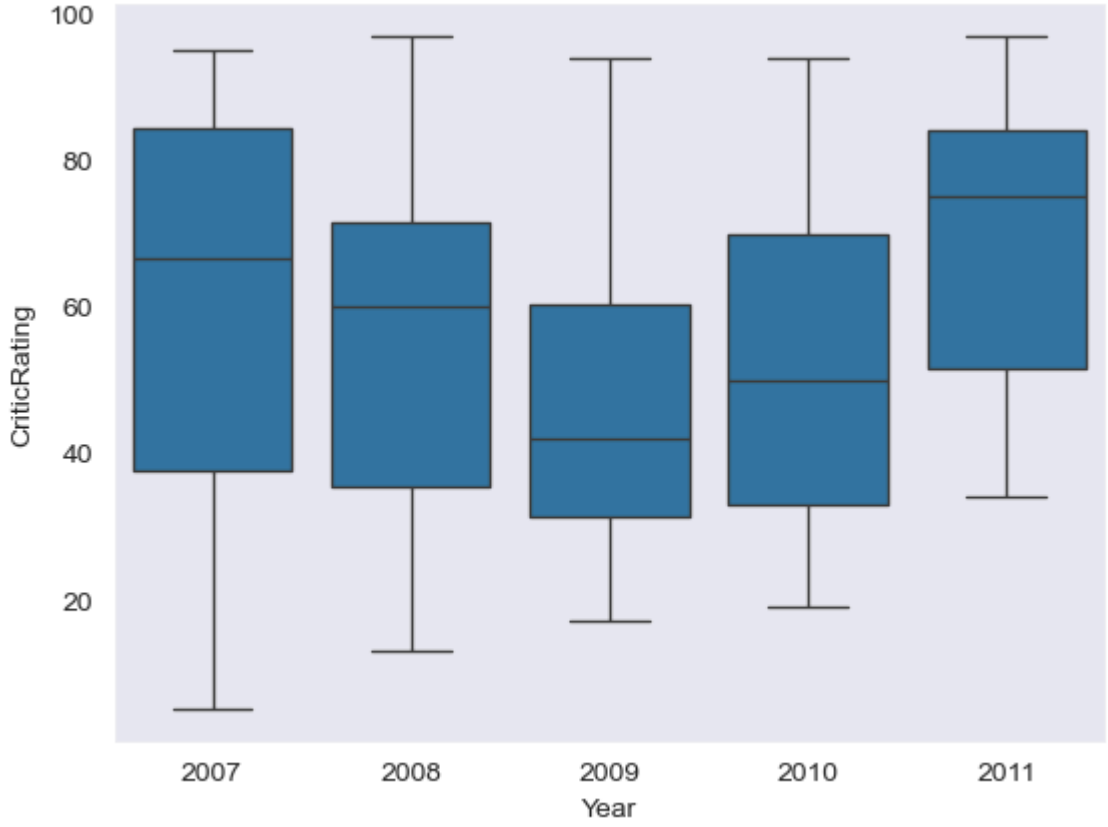
In [103…
```
#Box plots -

w = sns.boxplot(data=movies, x='Genre', y = 'CriticRating')
```



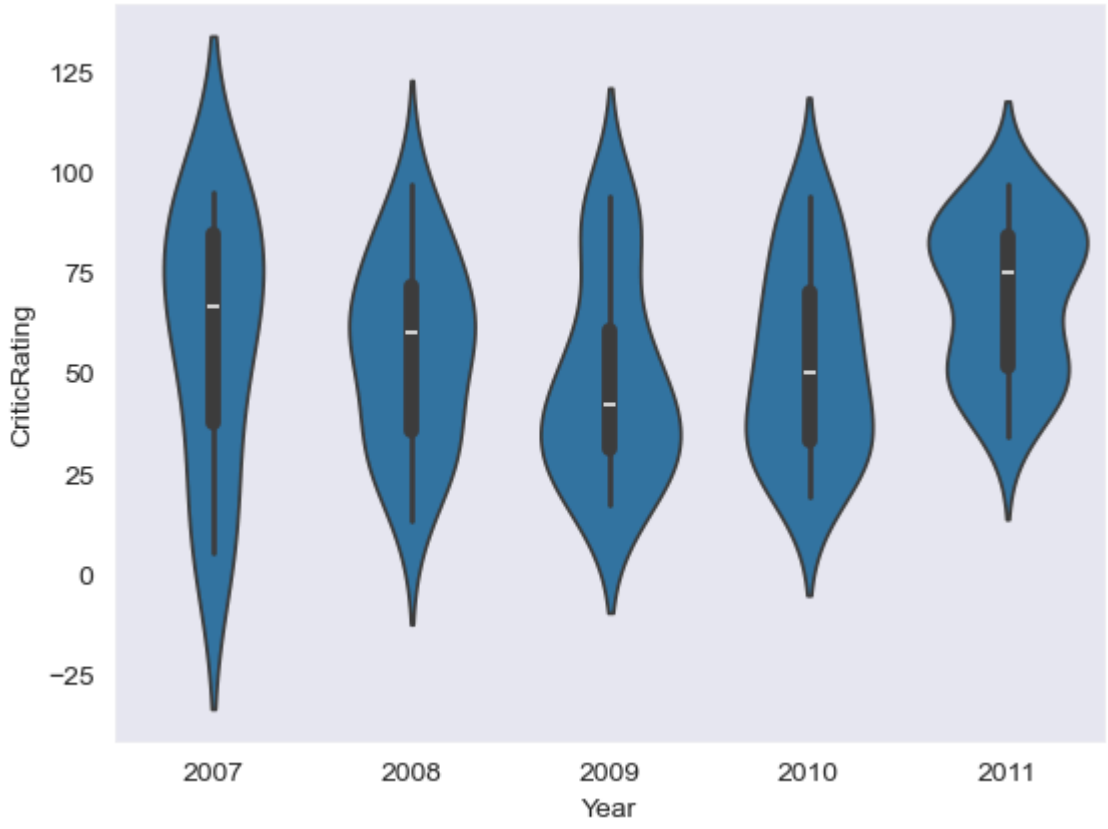
In [105…
```
#violin plot

z = sns.violinplot(data=movies, x='Genre', y = 'CriticRating')
```



In [107…
```
w1 = sns.boxplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRati
```



In [109…
```
z = sns.violinplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRa
```

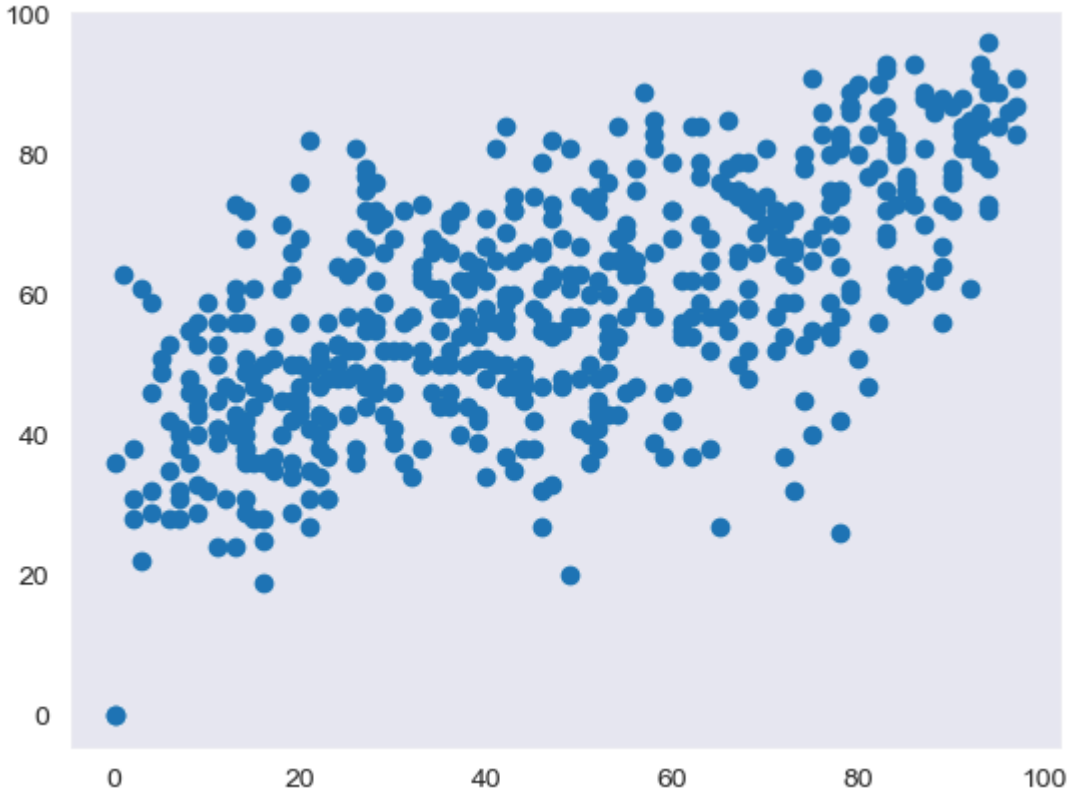


In [111…
```
# Createing a Facet grid
```

In [113…
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre') #kind of s
```

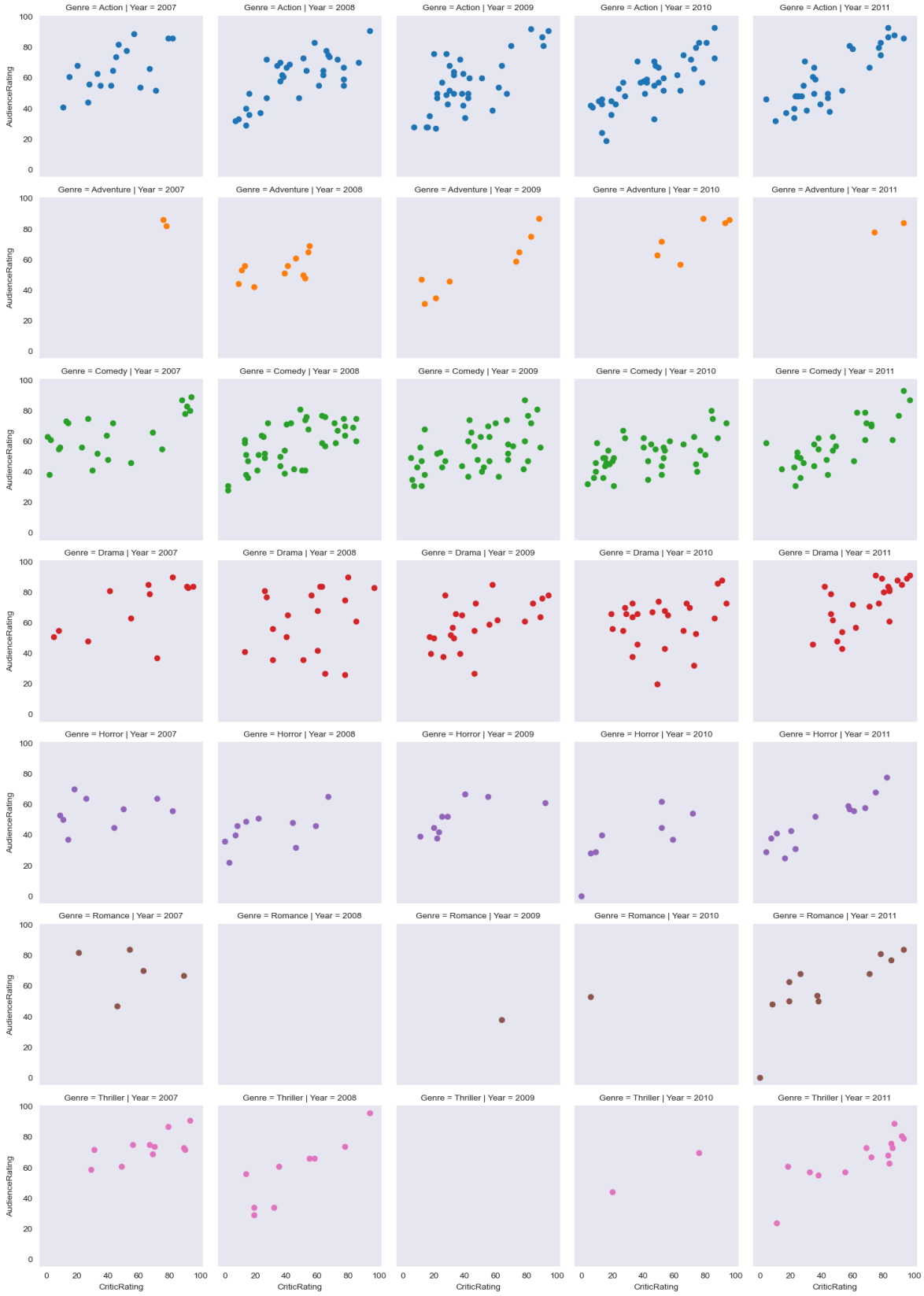
In [115…
```
plt.scatter(movies.CriticRating,movies.AudienceRating)
```

Out[115…
```
<matplotlib.collections.PathCollection at 0x26952a5af90>
```



In [117…

```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating' ) #scatterplots are mapp
```

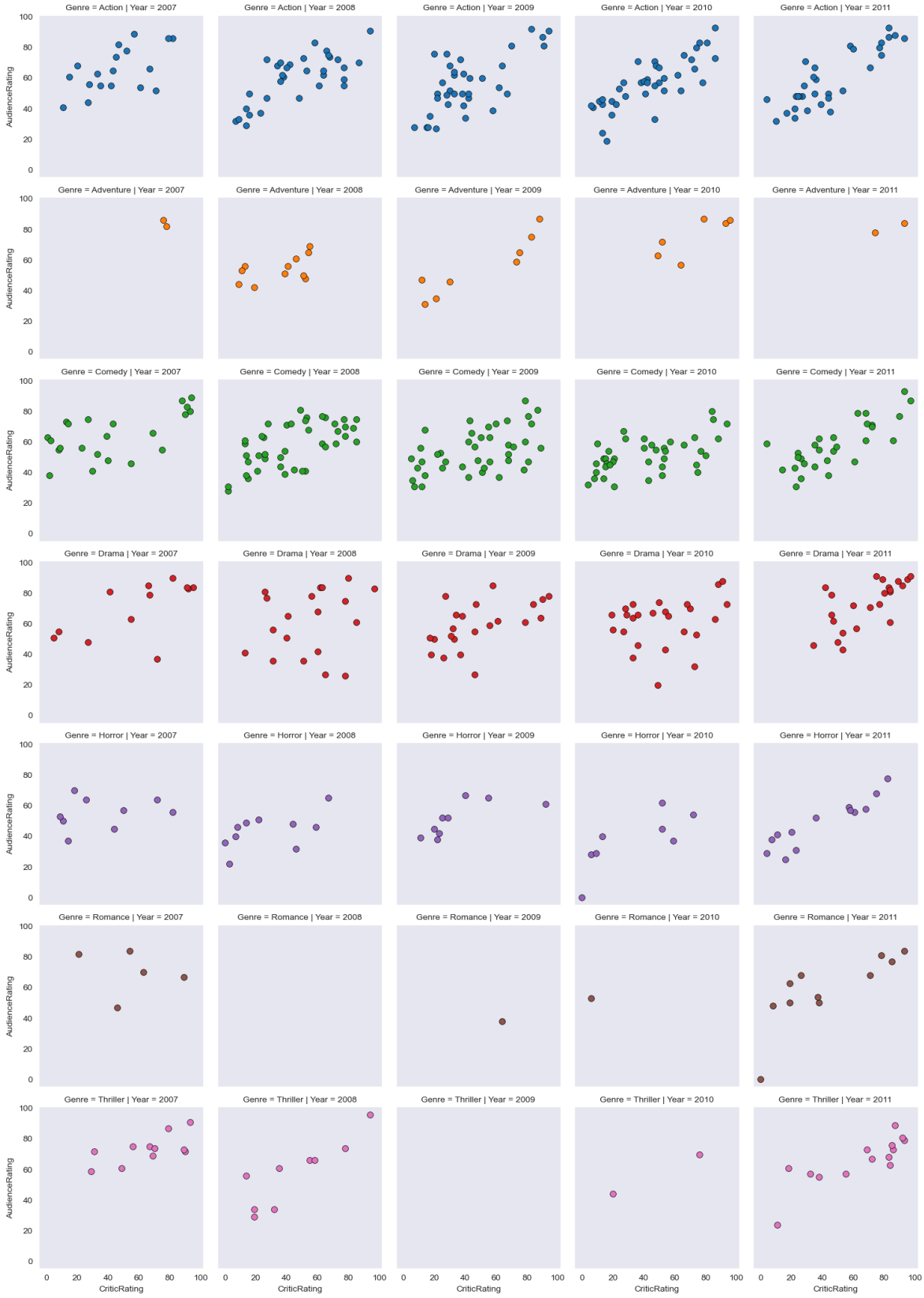


In [119…

```
# you can populated any type of chat.

g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.hist, 'BudgetMillions') #scatterplots are mapped in facetgrid
```

In [121…

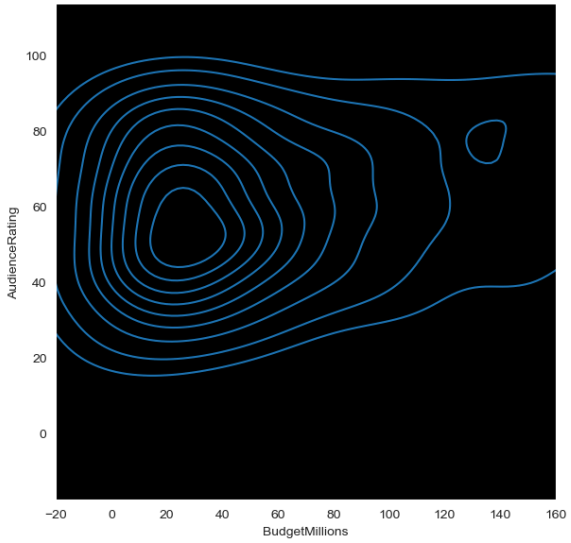
```
#
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
kws = dict(s=50, linewidth=0.5,edgecolor='black')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating',**kws ) #scatterplots ar
```

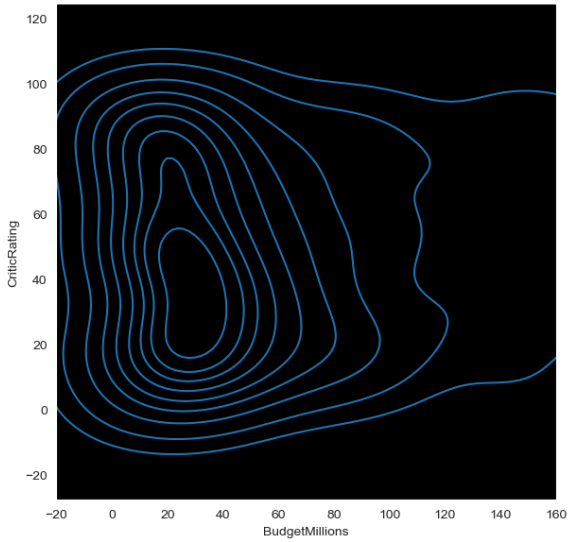


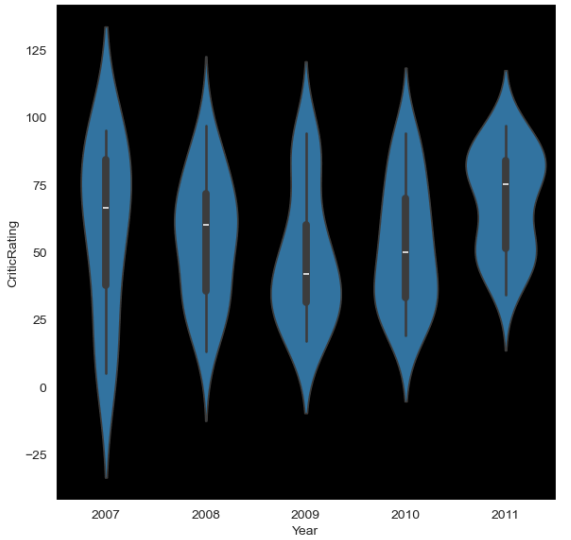
In [135…

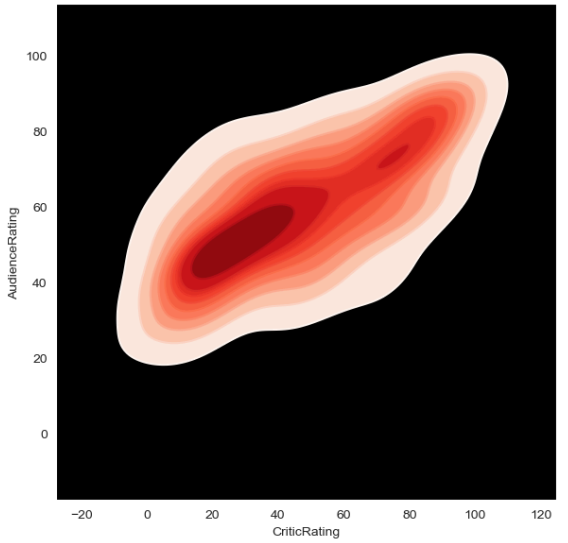
```
# python is not vectorize programming language
# Building dashboards (dashboard - combination of chats)

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))
```

```
z = sns.violinplot(movies[movies.Genre=='Drama'], x='Year', y = 'CriticRating',

k4 = sns.kdeplot(x=movies.CriticRating,y=movies.AudienceRating,shade = True,shad

k4b = sns.kdeplot(x=movies.CriticRating,y=movies.AudienceRating,cmap='Reds',ax =

plt.show()
```





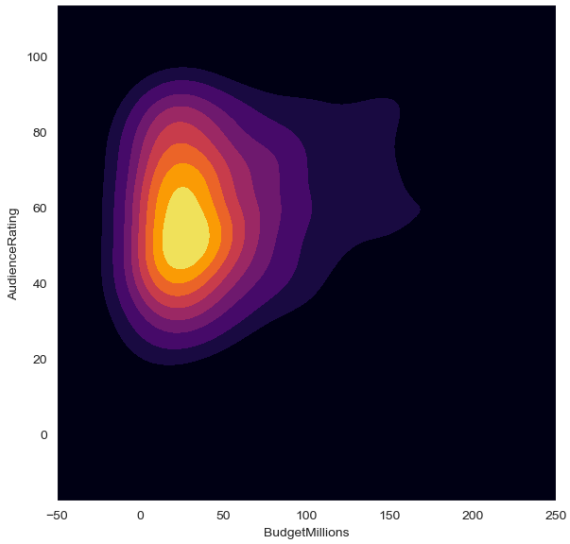


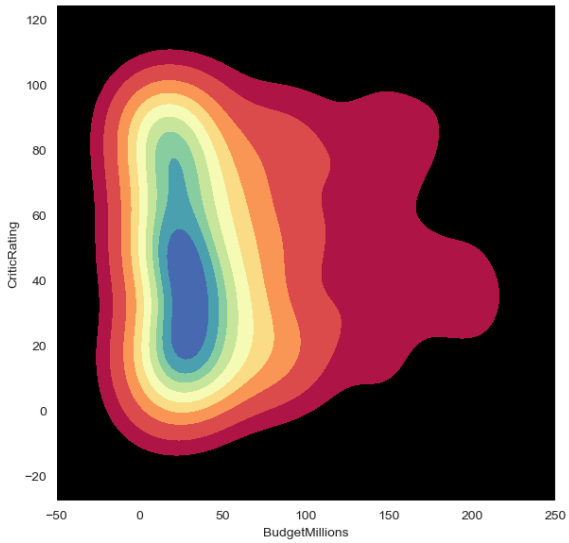


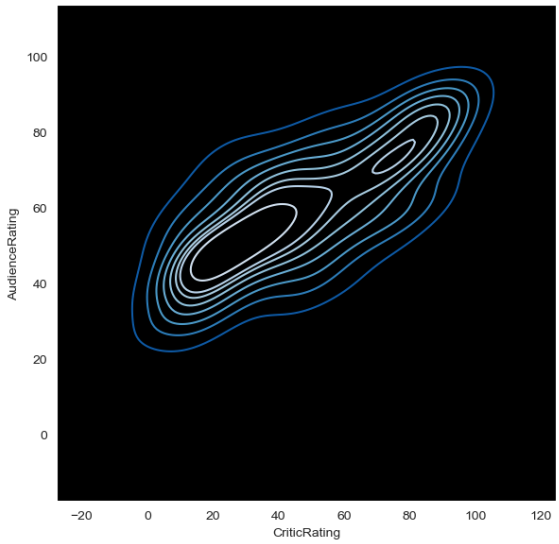
In [137…

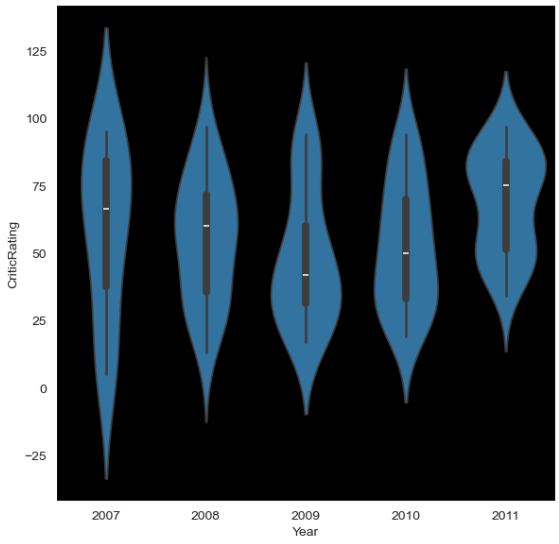
```
sns.set_style('dark',{'axes.facecolor':'black'})
f , axes = plt.subplots(2,2,figsize=(15,15))
#plot [0,0]
k1 = sns.kdeplot(x=movies.BudgetMillions,y=movies.AudienceRating, \
shade = True, shade_lowest=True,cmap = 'inferno', \
ax = axes[0,0])
#plot [0,1]
k2 = sns.kdeplot(x=movies.BudgetMillions, y=movies.CriticRating,\
shade=True, shade_lowest=False, cmap='Spectral',\
ax = axes[0,1])
#plot[1,0]
k3 = sns.kdeplot(x=movies.CriticRating, y=movies.AudienceRating, \
shade = False,shade_lowest=True,cmap='Blues_r', \
ax=axes[1,0])
#plot[1,1]
vi = sns.violinplot(data=movies[movies.Genre=='Drama'], \
```

```
x='Year', y = 'CriticRating', ax=axes[1,1])
k1.set(xlim=(-50,250))
k2.set(xlim=(-50,250))
plt.show()
```









Completed

In [ ]: