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

In [2]:
```
movies=pd.read_csv(r"C:\Users\rahee\Downloads\Movie-Rating.csv")
```

In [3]:
```
movies
```

Out[3]:

	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 [4]:
```
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   Rotten Tomatoes Ratings %  559 non-null    int64 
 3   Audience Ratings %         559 non-null    int64 
 4   Budget (million $)         559 non-null    int64 
 5   Year of release            559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [5]:
```
type(movies)
```

Out[5]:
```
pandas.core.frame.DataFrame
```

In [6]:
```
len(movies)
```

Out[6]:
```
559
```

In [7]:
```
import numpy as np
print(np.__version__)
```

```
1.26.4
```

In [8]:
```
pd.__version__
```

Out[8]:
```
'2.2.2'
```

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

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

In [10]:
```
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   Rotten Tomatoes Ratings %  559 non-null    int64 
 3   Audience Ratings %         559 non-null    int64 
 4   Budget (million $)         559 non-null    int64 
 5   Year of release            559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [11]:
```
movies.shape #no of rows and columns
```

Out[11]:
```
(559, 6)
```

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

Out[12]:

	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 [14]:
```
movies.columns
```

Out[14]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

In [15]:
```
movies.columns=['Film','Genre','CriticRating','AudienceRating','BudgetMillions','Year']
```

In [16]:
```
movies.head(1) #removed space and noise characters
```

Out[16]:

	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009

In [17]:
```
movies.shape
```

Out[17]:
```
(559, 6)
```

In [18]:
```
movies.describe #descriptive stats
```

Out[18]:
```
<bound method NDFrame.describe of                          Film      Genre  CriticRating  Audien
ceRating  \
0    (500) Days of Summer      Comedy            87             81
1              10,000 B.C.  Adventure             9             44
2               12 Rounds      Action            30             52
3               127 Hours  Adventure            93             84
4               17 Again       Comedy            55             70
..                    ...         ...           ...            ...
554          Your Highness     Comedy            26             36
555        Youth in Revolt     Comedy            68             52
556                 Zodiac   Thriller            89             73
557            Zombieland      Action            90             87
558              Zookeeper     Comedy            14             42

     BudgetMillions  Year
0                 8  2009
1               105  2008
2                20  2009
3                18  2010
4                20  2009
..              ...   ...
554              50  2011
555              18  2009
556              65  2007
557              24  2009
558              80  2011

[559 rows x 6 columns]>
```

In [19]:
```
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 [20]:
```
movies.Film=movies.Film.astype('category')
```

In [21]:
```
movies.head()
```

Out[21]:

	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 [22]:
```
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 [23]:
```
movies.Genre=movies.Genre.astype('category')
```

In [24]:
```
movies.Genre
```

Out[24]:
```
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', 'Romance', 'Thri
ller']
```

In [25]:
```
movies.Year=movies.Year.astype('category')
```

In [26]:
```
movies.Year
```

Out[26]:
```
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 [27]:
```
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
```

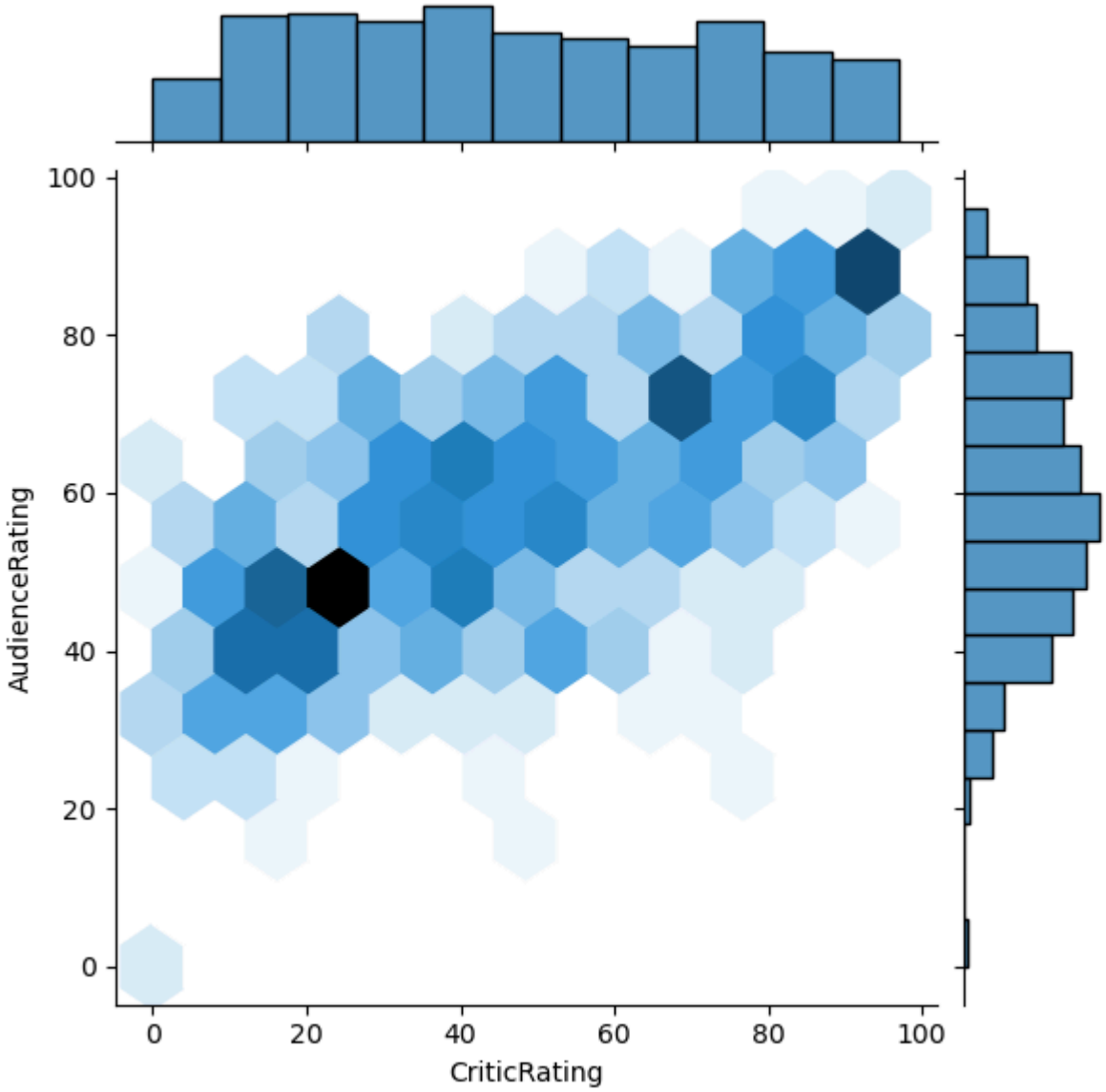
In [28]:
```
movies.describe()
```

Out[28]:

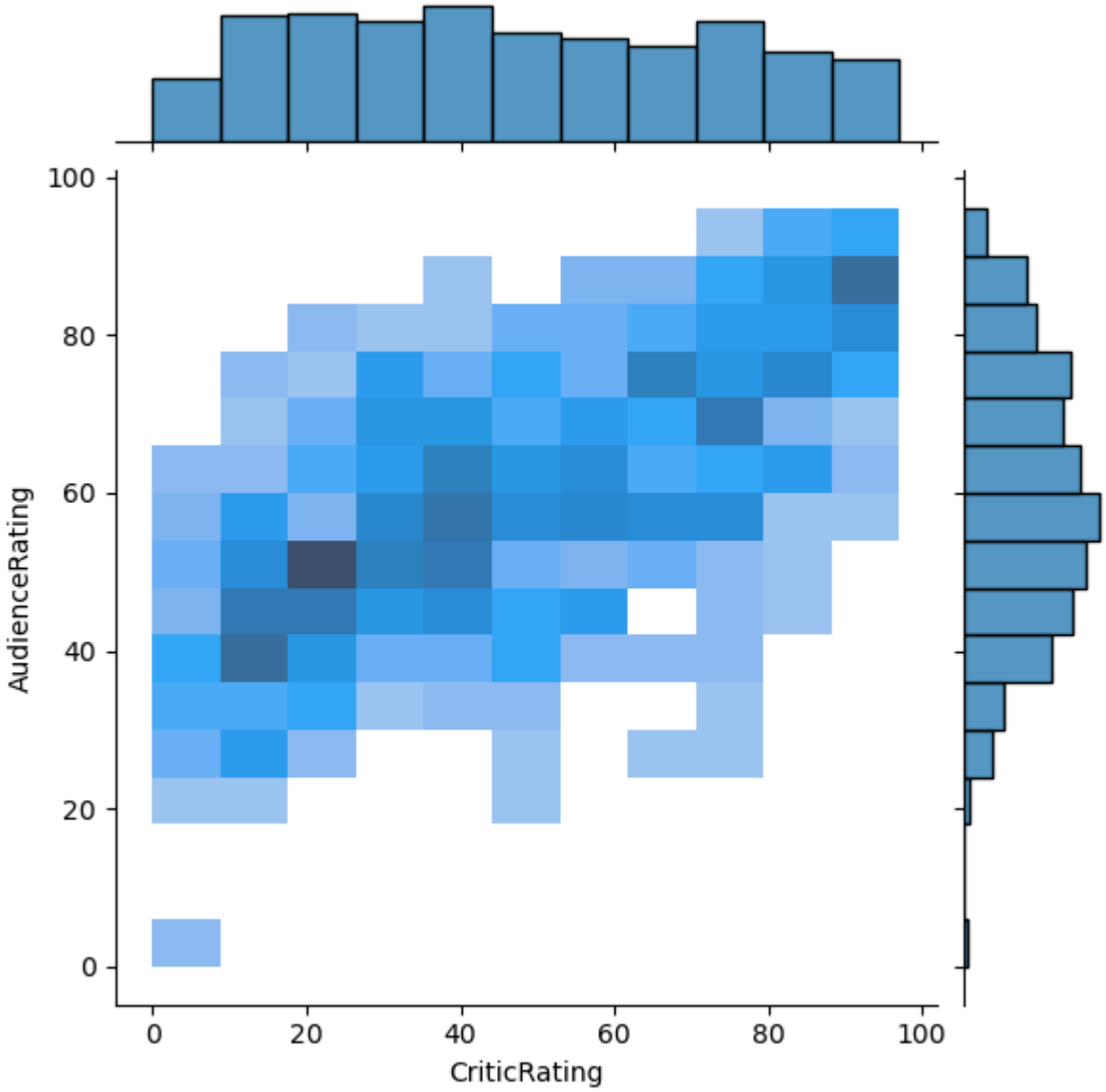
	CriticRating	AudienceRating	BudgetMillions
count	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136
std	26.413091	16.826887	48.731817
min	0.000000	0.000000	0.000000
25%	25.000000	47.000000	20.000000
50%	46.000000	58.000000	35.000000
75%	70.000000	72.000000	65.000000
max	97.000000	96.000000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

In [29]:
```
from matplotlib import pyplot as plt #visualization
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

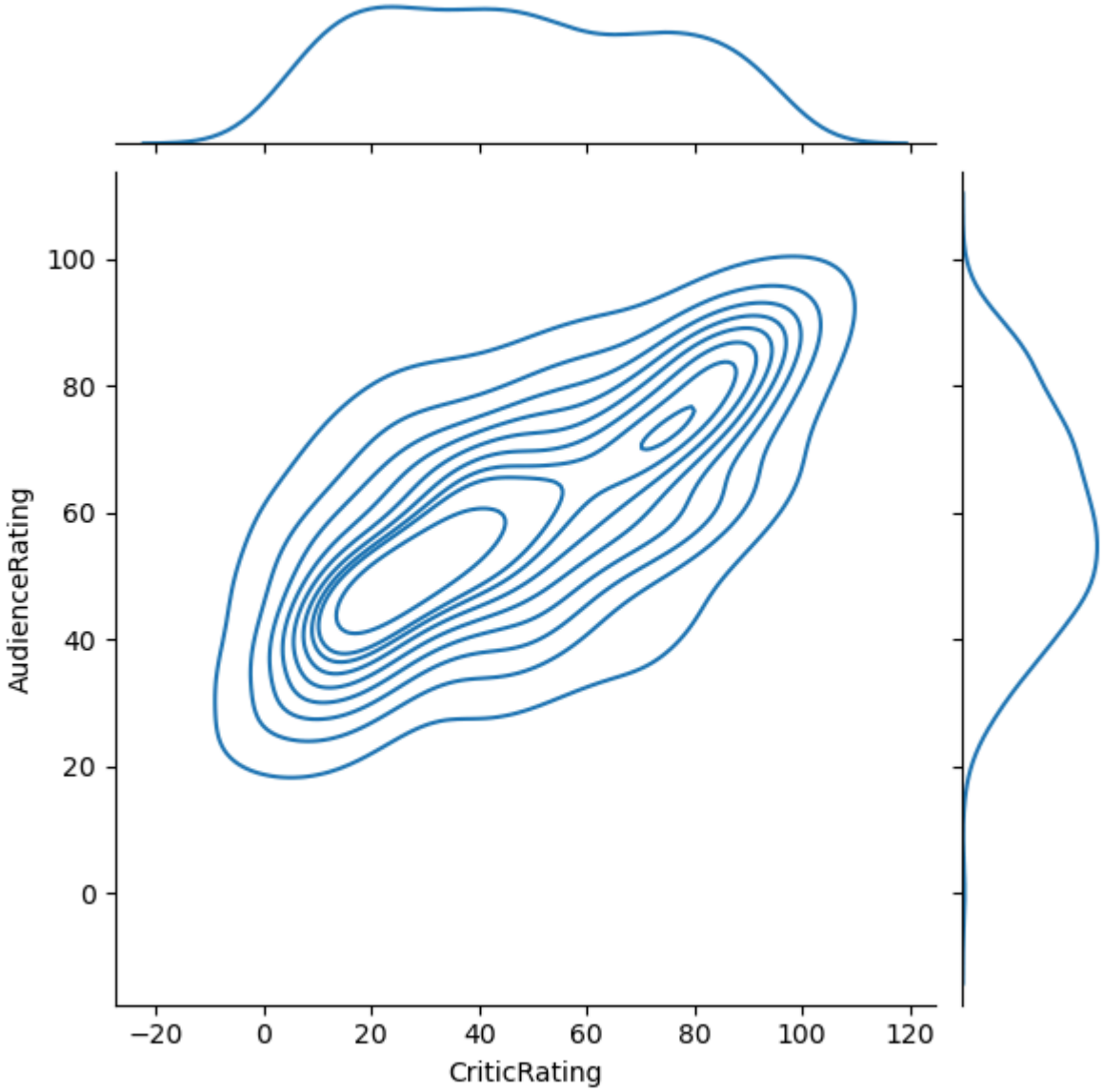
In [30]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='AudienceRating',kind='hex',)
```



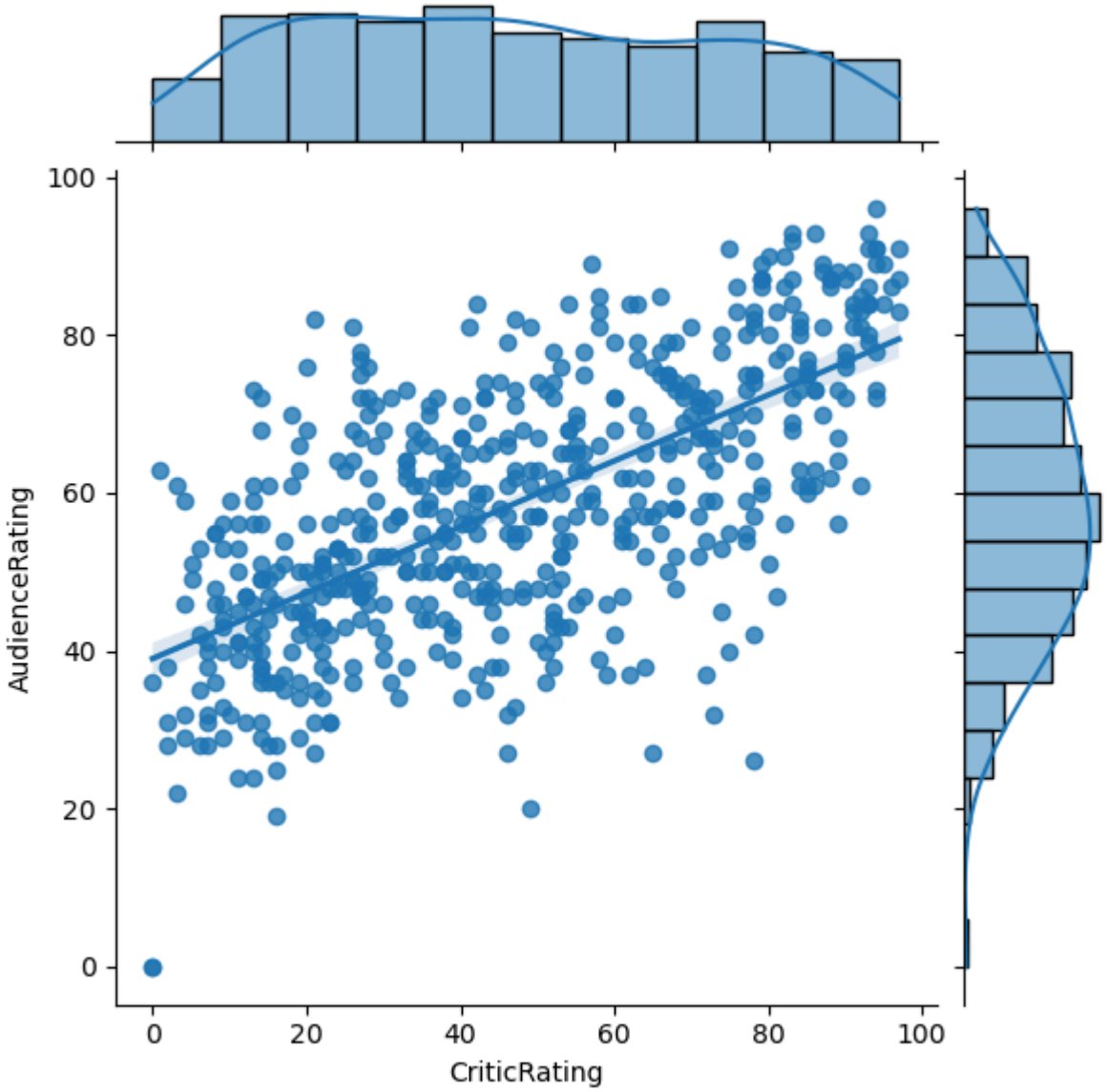
In [31]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='AudienceRating',kind='hist')
```



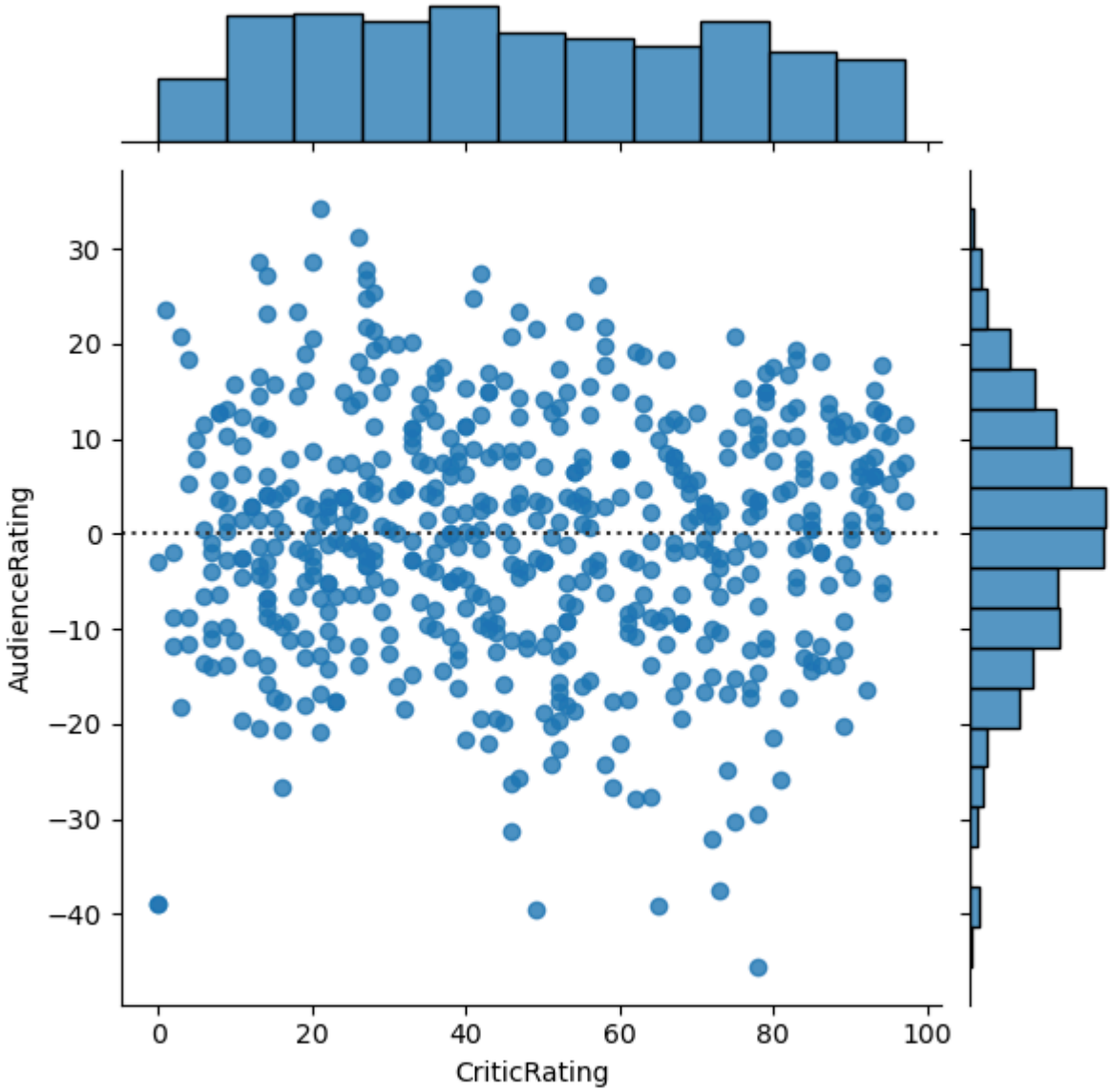
In [32]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='AudienceRating',kind='kde')
```



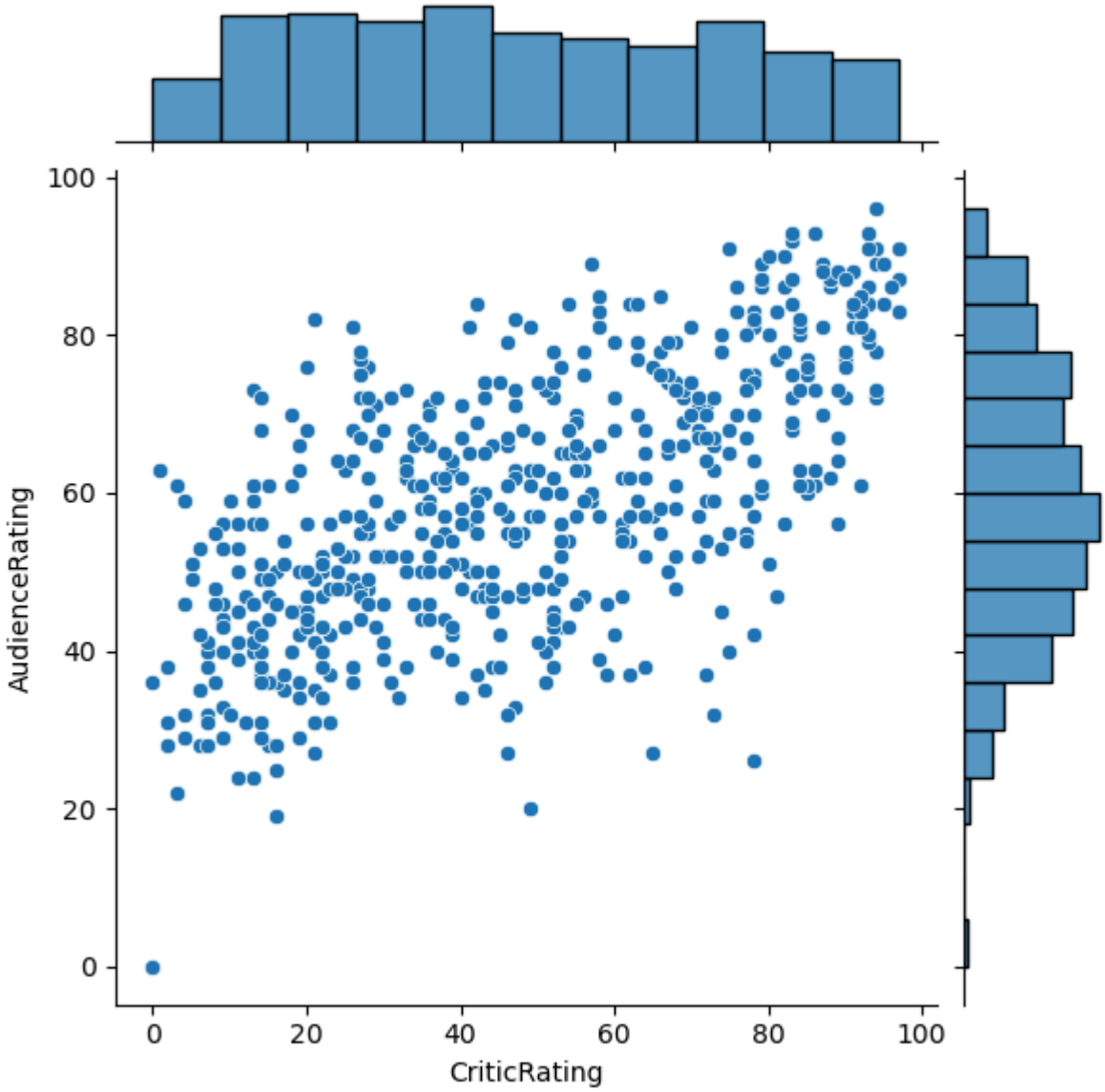
In [33]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='AudienceRating',kind='reg')
```



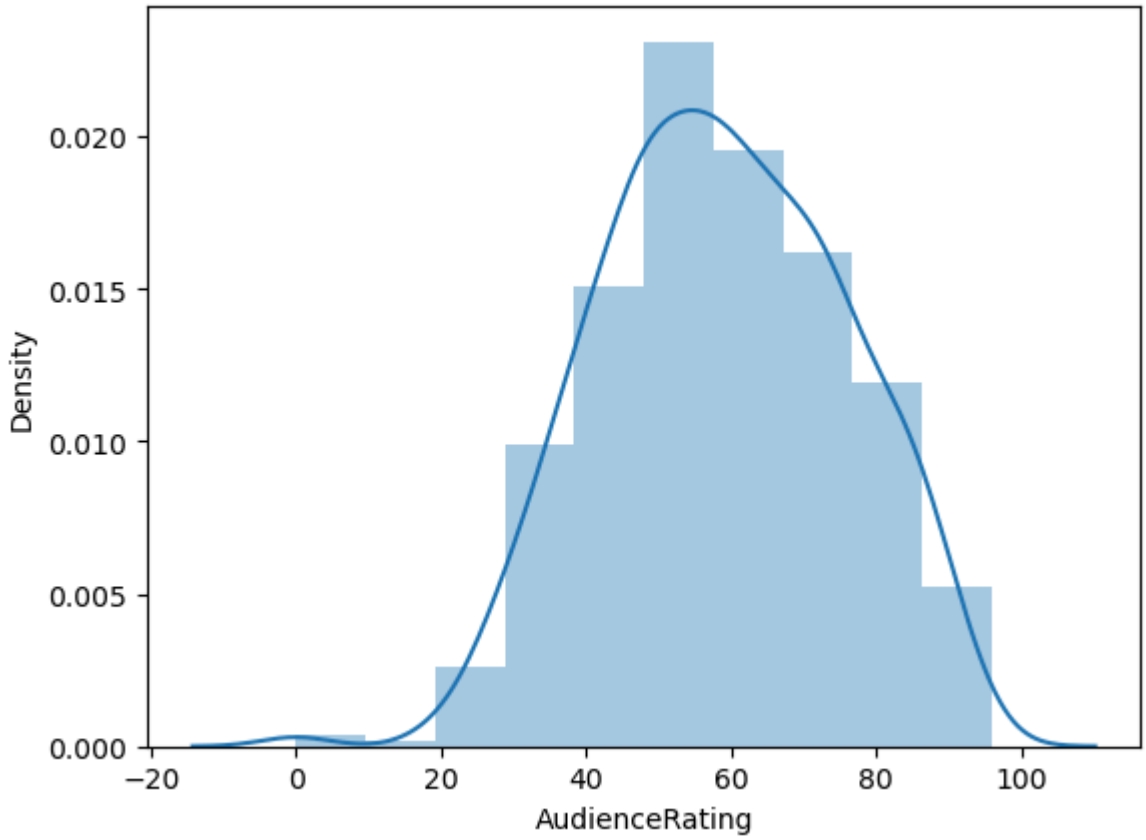
In [34]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='AudienceRating',kind='resid')
```



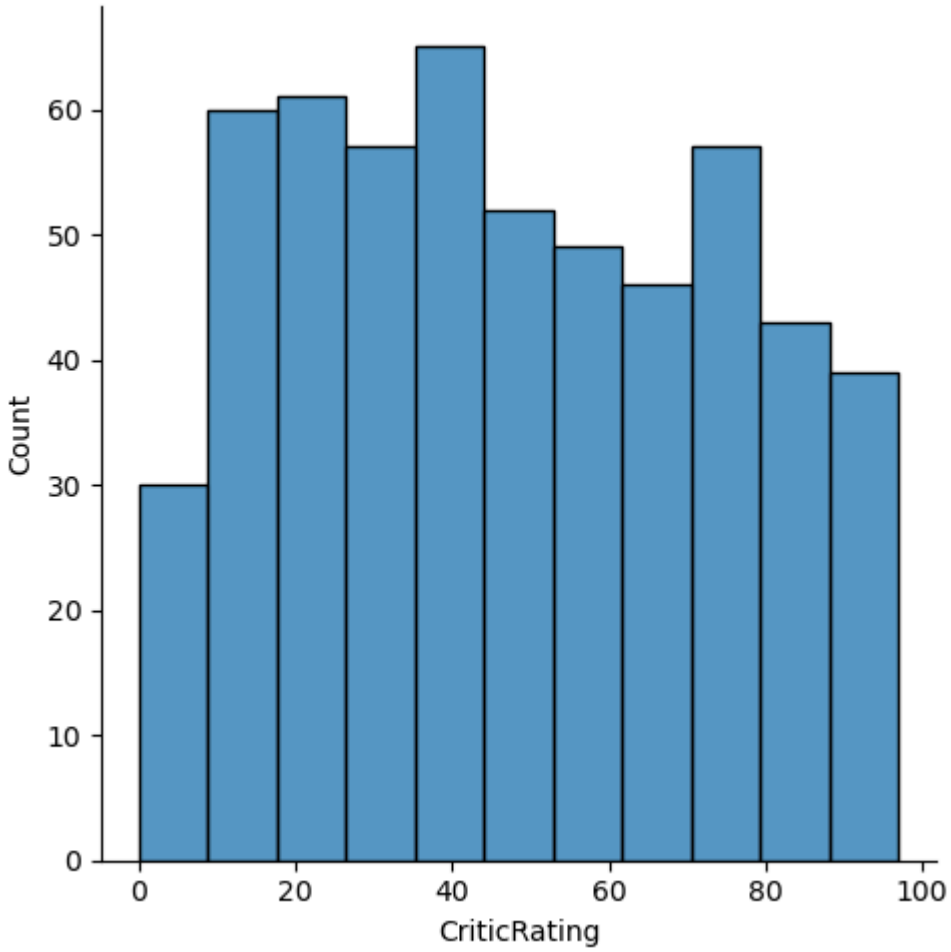
In [35]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='AudienceRating',kind='scatter')
```



In [36]:
```
#Histogram
m1=sns.distplot(movies.AudienceRating,bins=10)
```

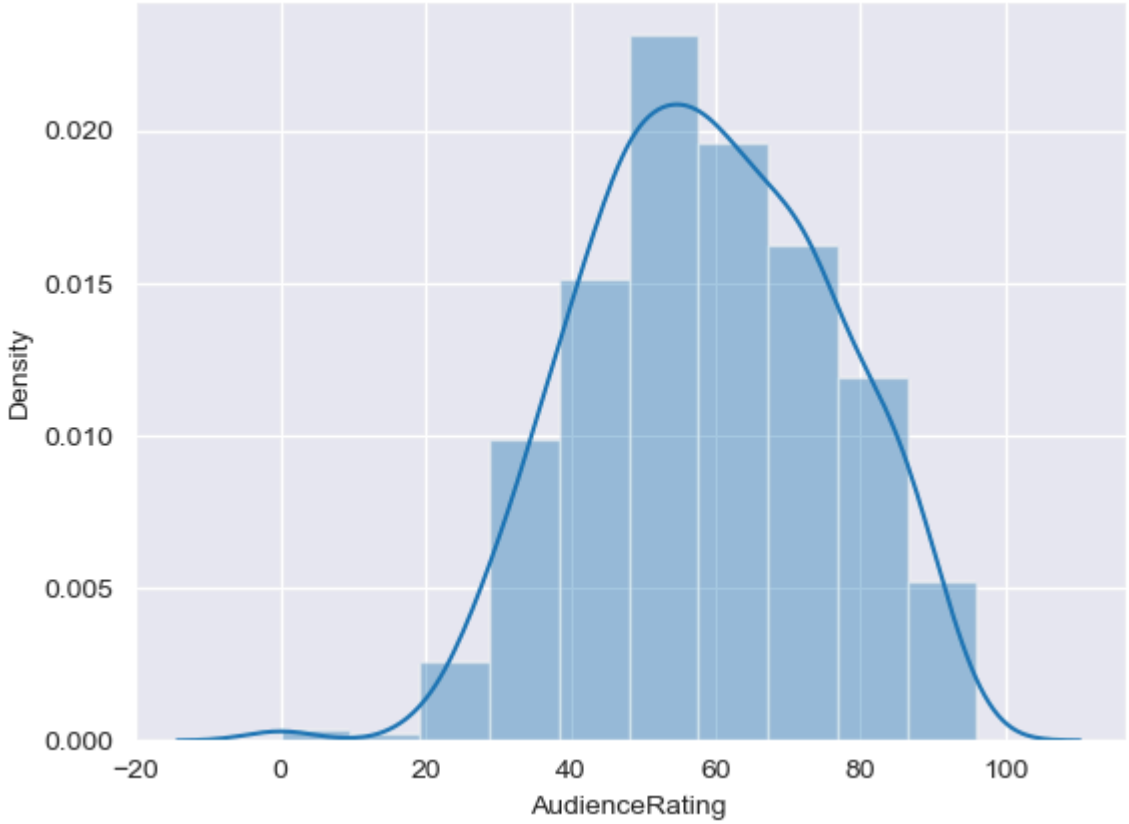


In [37]:
```
m1=sns.displot(movies.CriticRating)
```

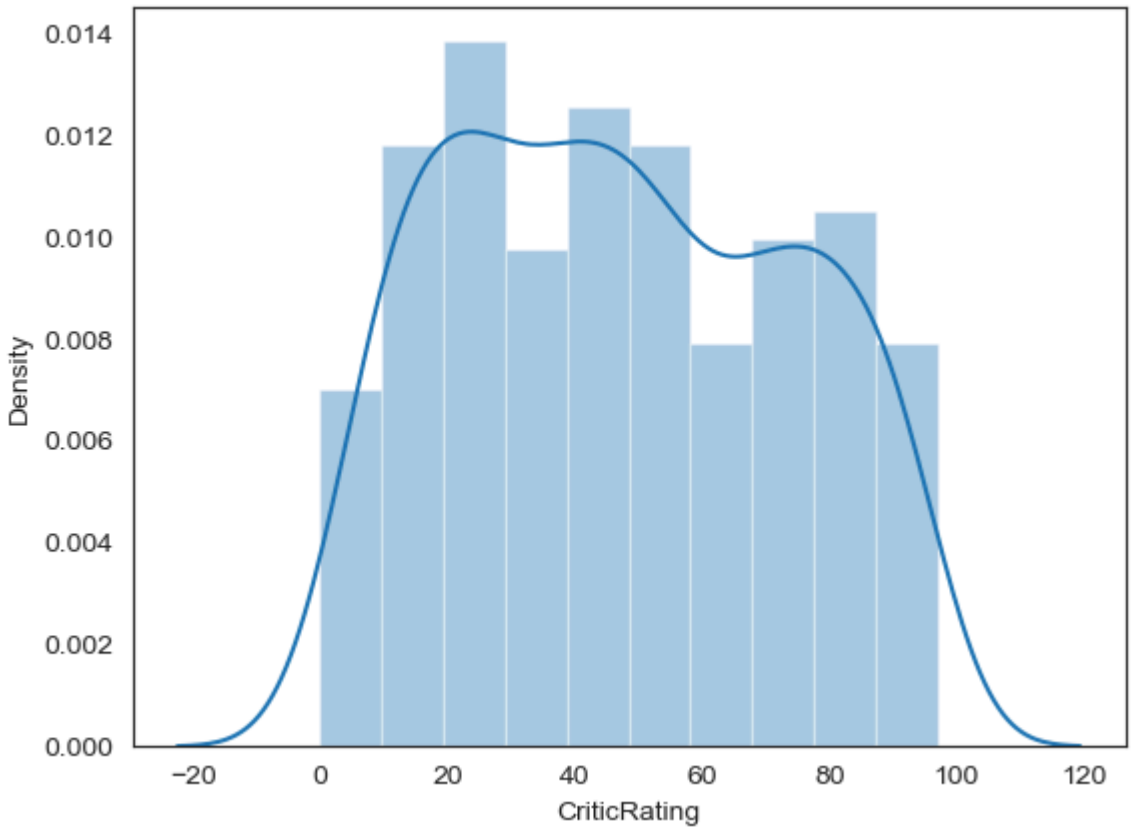


In [38]:
```
sns.set_style('darkgrid')
```

In [39]:
```
m2=sns.distplot(movies.AudienceRating,bins=10)
```

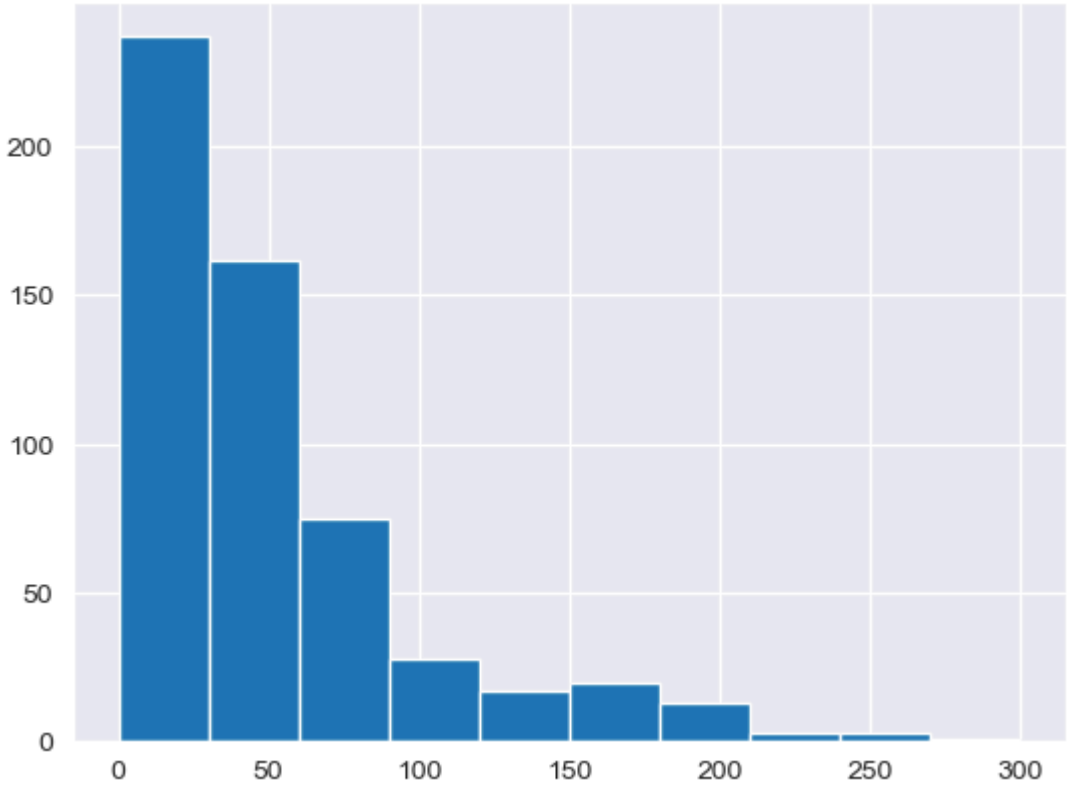


In [40]:
```
sns.set_style('white')
m2=sns.distplot(movies.CriticRating,bins=10)
```

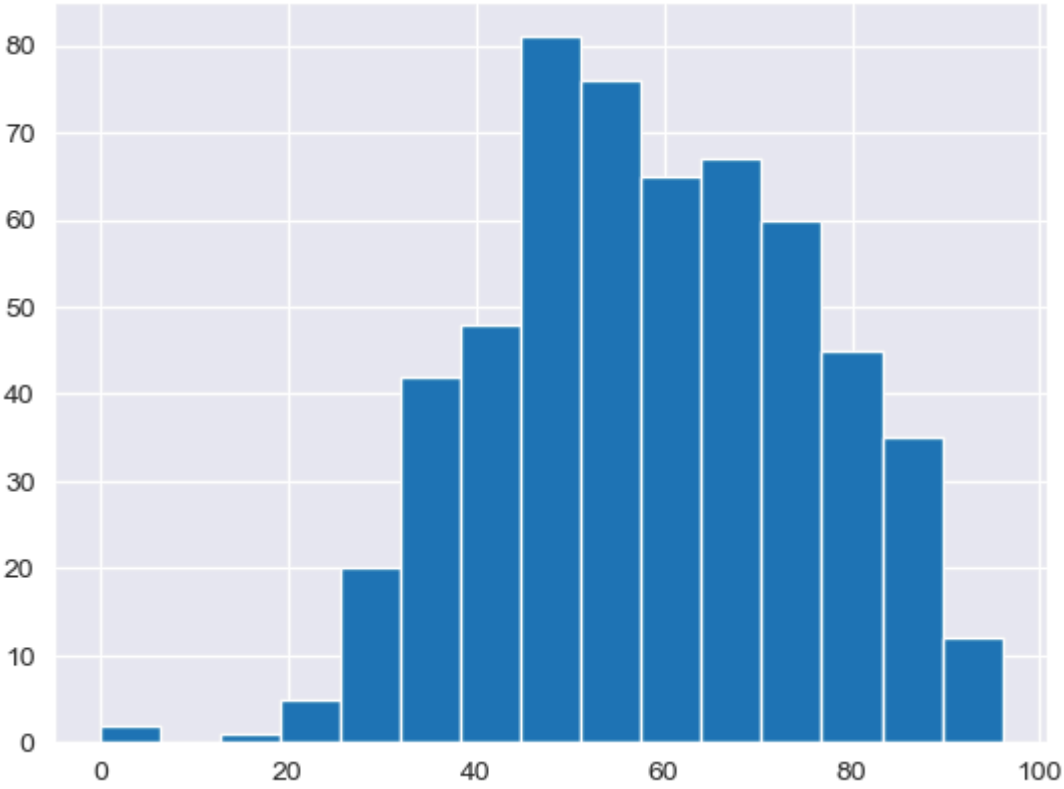


In [41]:
```
sns.set_style('darkgrid')
```

In [42]:
```
plt.hist(movies.BudgetMillions)
plt.show()
```

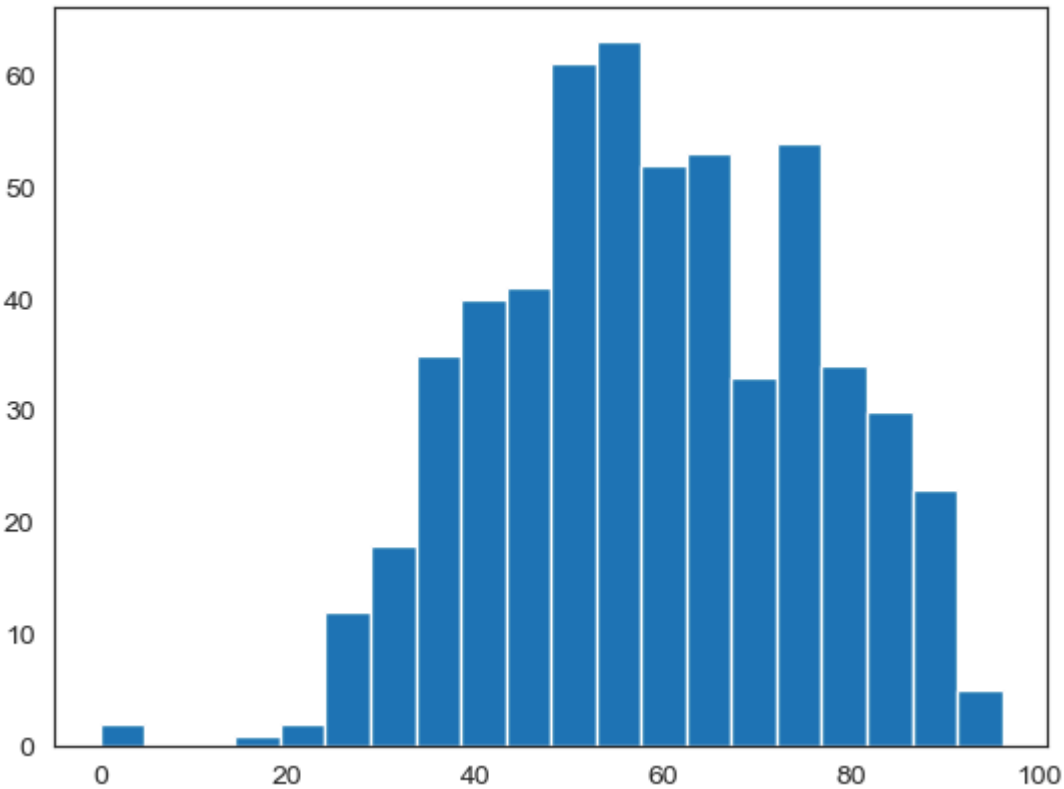


In [106…
```
#sns.set_style('darkgrid')
n1=plt.hist(movies.AudienceRating,bins=15)
plt.show()
```



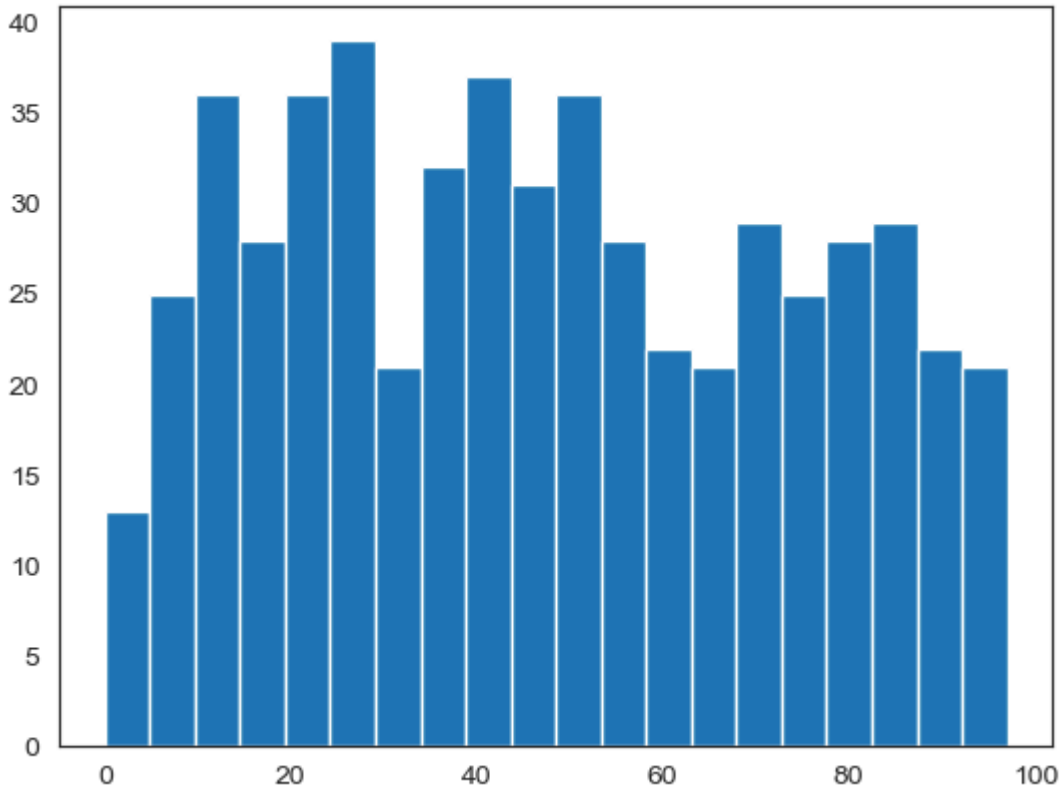
In [108…

```
sns.set_style('white') #normal sitribultion and called a bell curve
n1=plt.hist(movies.AudienceRating,bins=20)
```

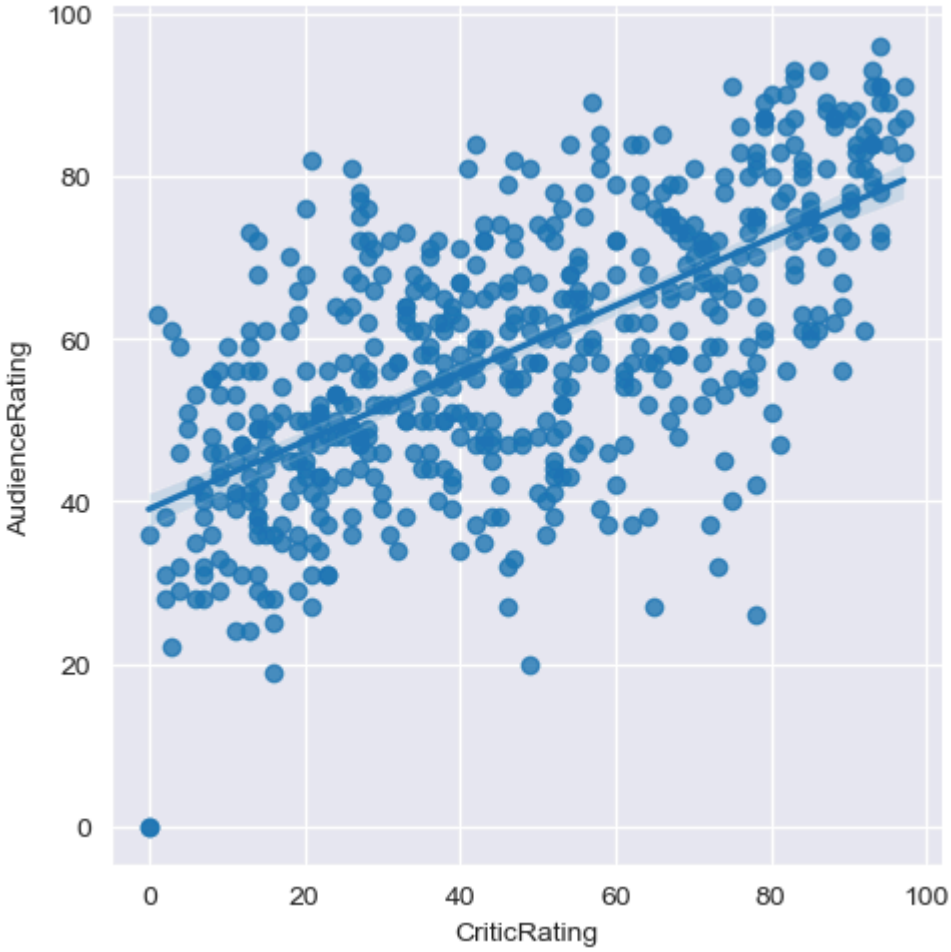


In [113…

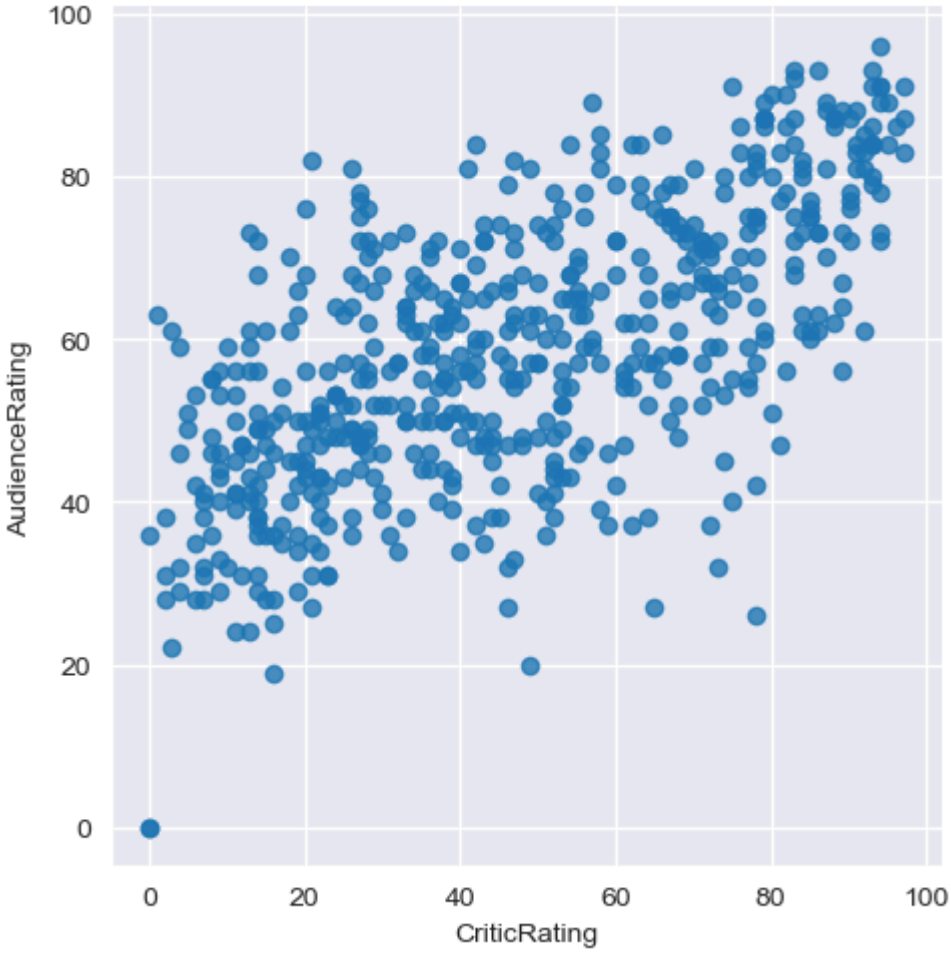
```
n1=plt.hist(movies.CriticRating,bins=20)#Uniform distribution
```



In [43]:
```
vis1=sns.lmplot(data=movies,x='CriticRating',y='AudienceRating',fit_reg=True)
```



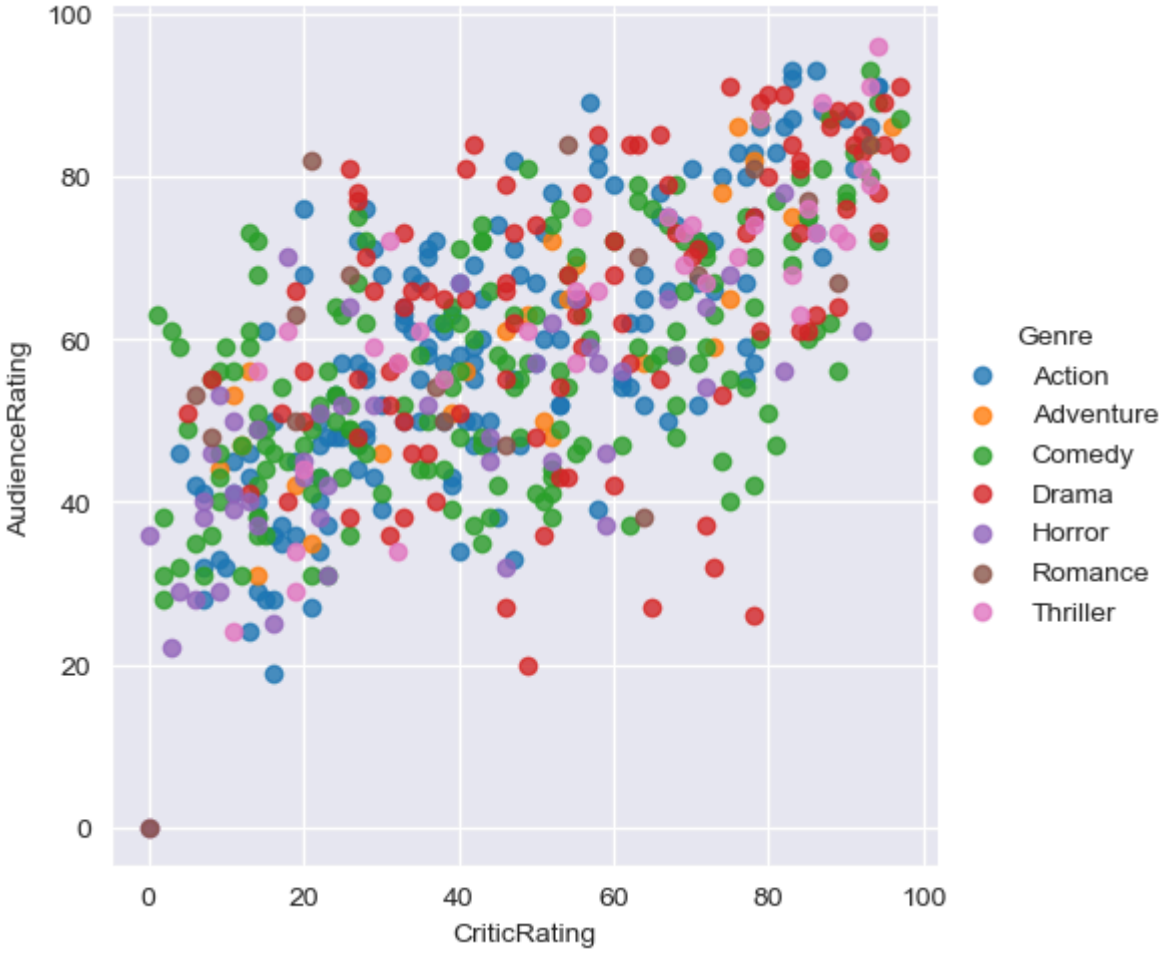
In [44]:
```
vis1=sns.lmplot(data=movies,x='CriticRating',y='AudienceRating',fit_reg=False)
```



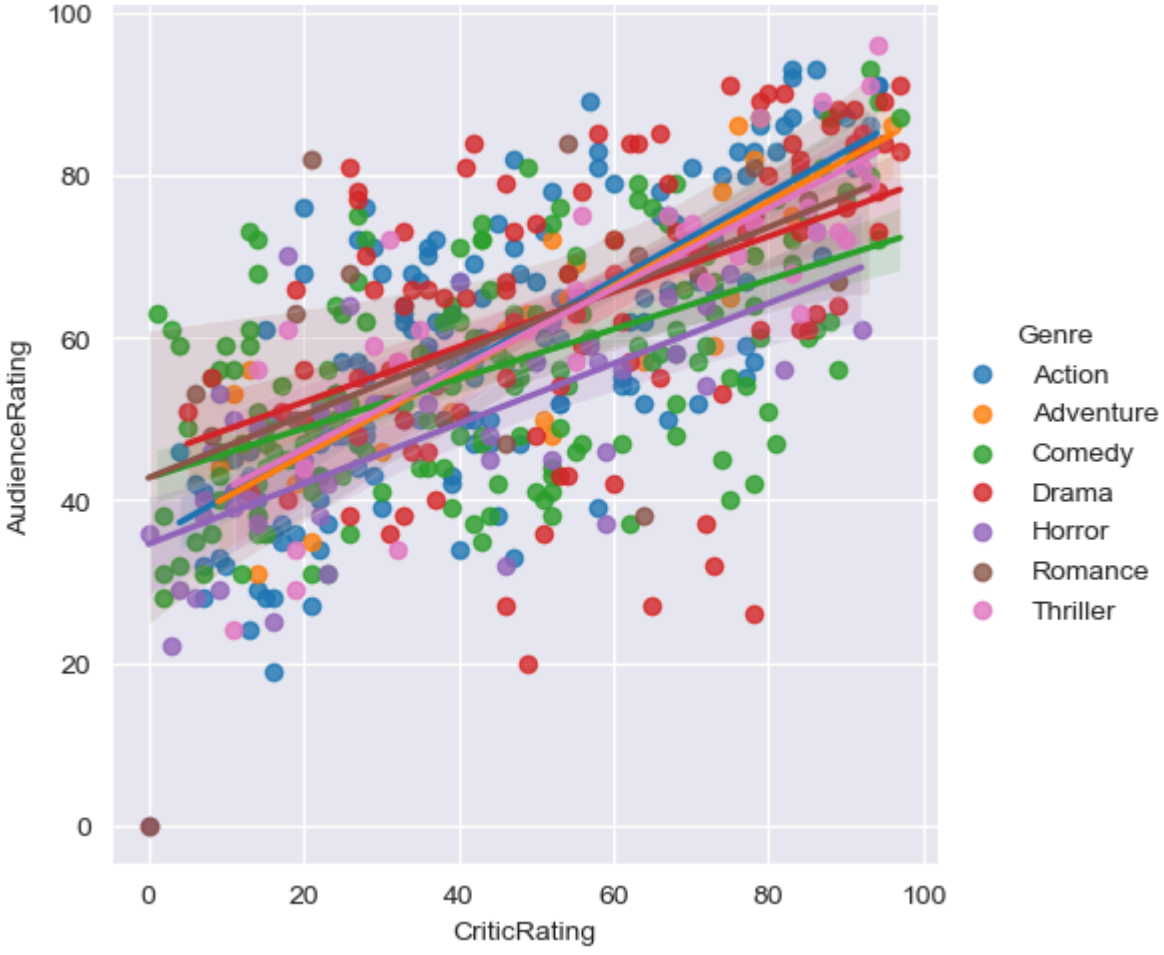
In [141…
```
#if you have 100 categories you cannot copy& paste all the things
for gen in movies.Genre.cat.categories:
    print(gen)
```

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

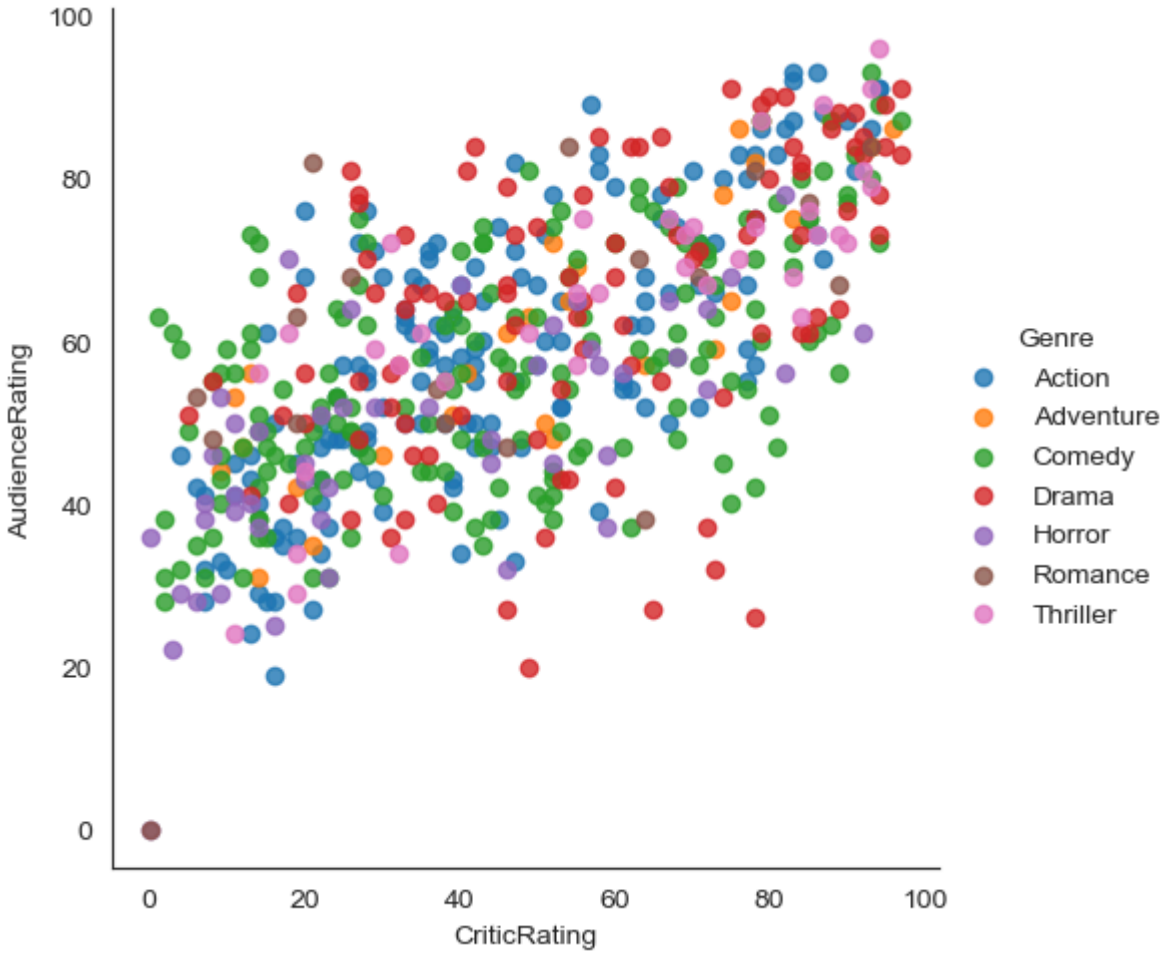
In [45]:
```
vis1=sns.lmplot(data=movies,x='CriticRating',y='AudienceRating',fit_reg=False,hue='Genre')
```



In [46]:
```
vis1=sns.lmplot(data=movies,x='CriticRating',y='AudienceRating',fit_reg=True,hue='Genre')
```

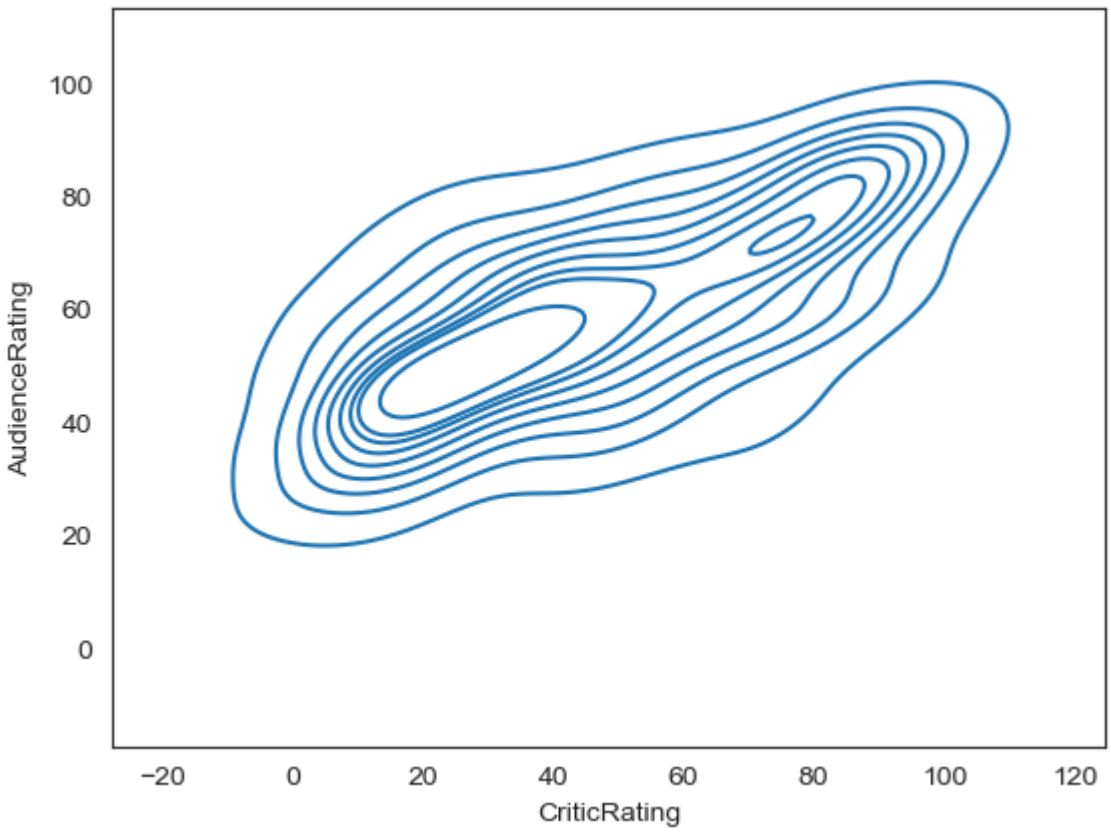


In [151…
```
vis1=sns.lmplot(data=movies,x='CriticRating',y='AudienceRating',\
                fit_reg=False,hue='Genre',aspect=1)
```

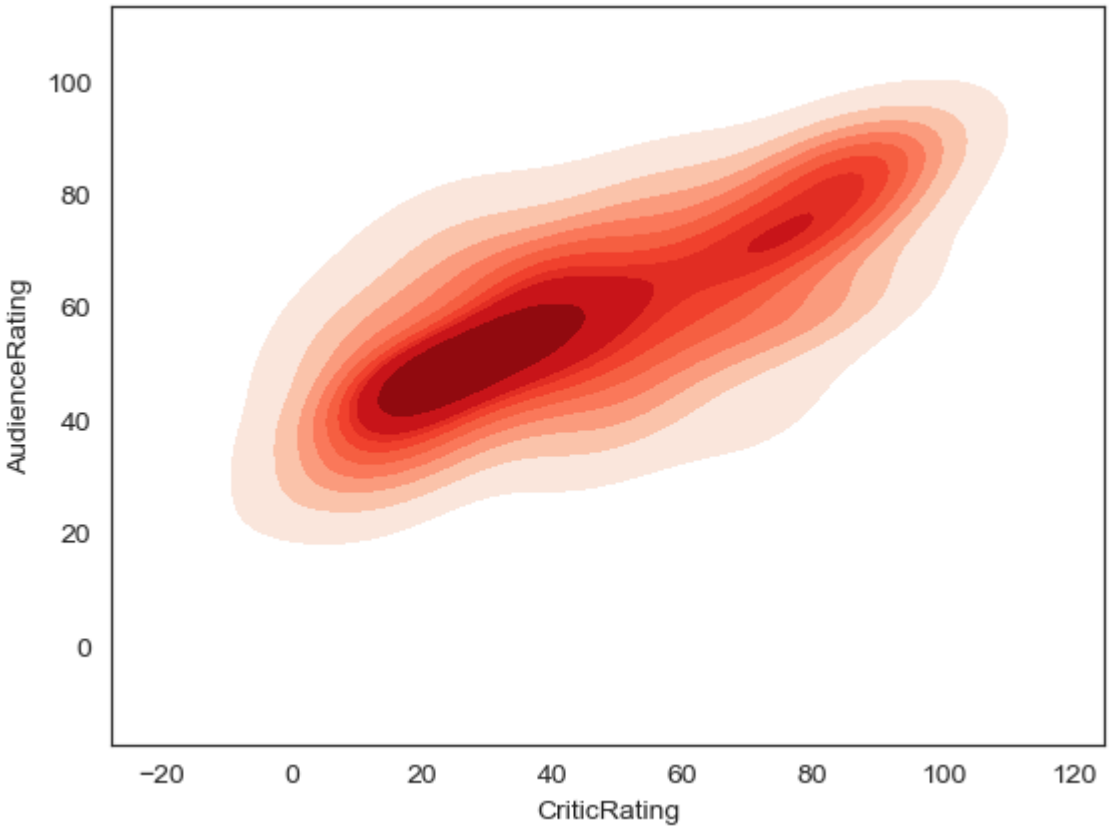


In [ ]:
```
# Kernal Density Estimate plot ( KDE PLOT)
# how can i visulize audience rating & critics rating . using scatterplot
```

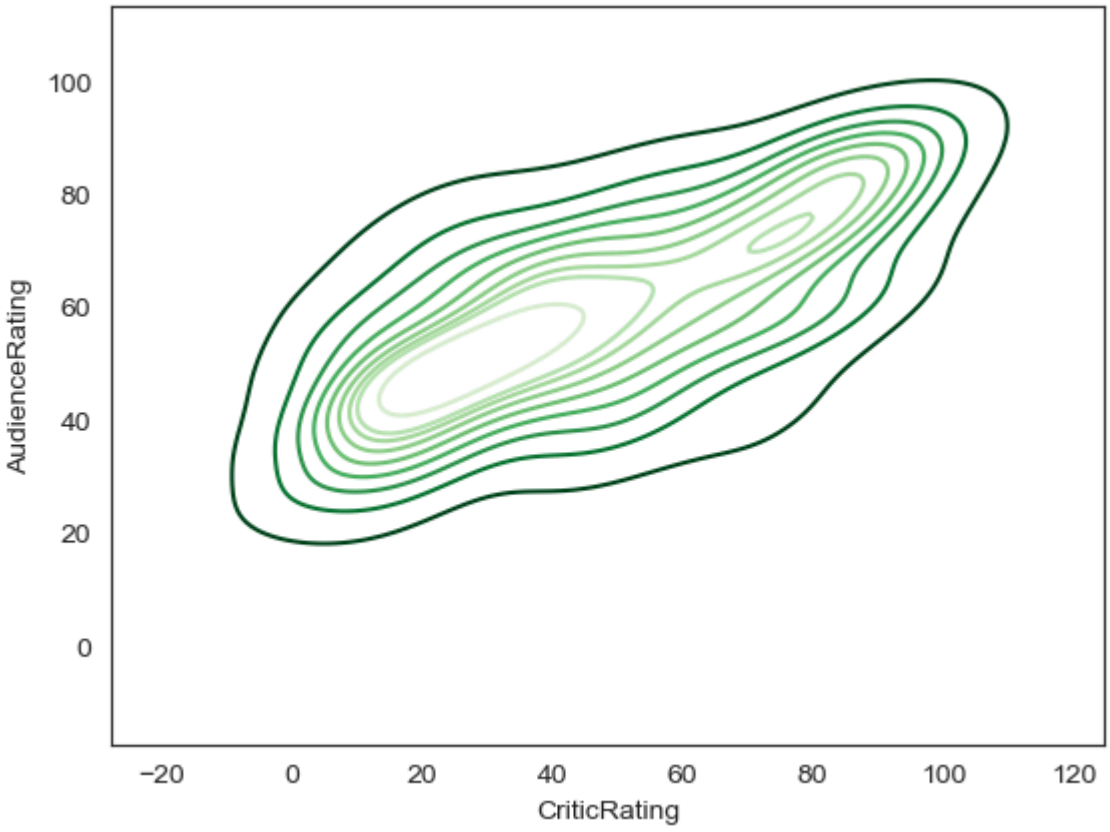
In [157…
```
k1 = sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating')

# where do u find more density and how density is distibuted across from the the chat
# center point is kernal this is calld KDE & insteade of dots it visualize like this
# we can able to clearly see the spread at the audience ratings
```



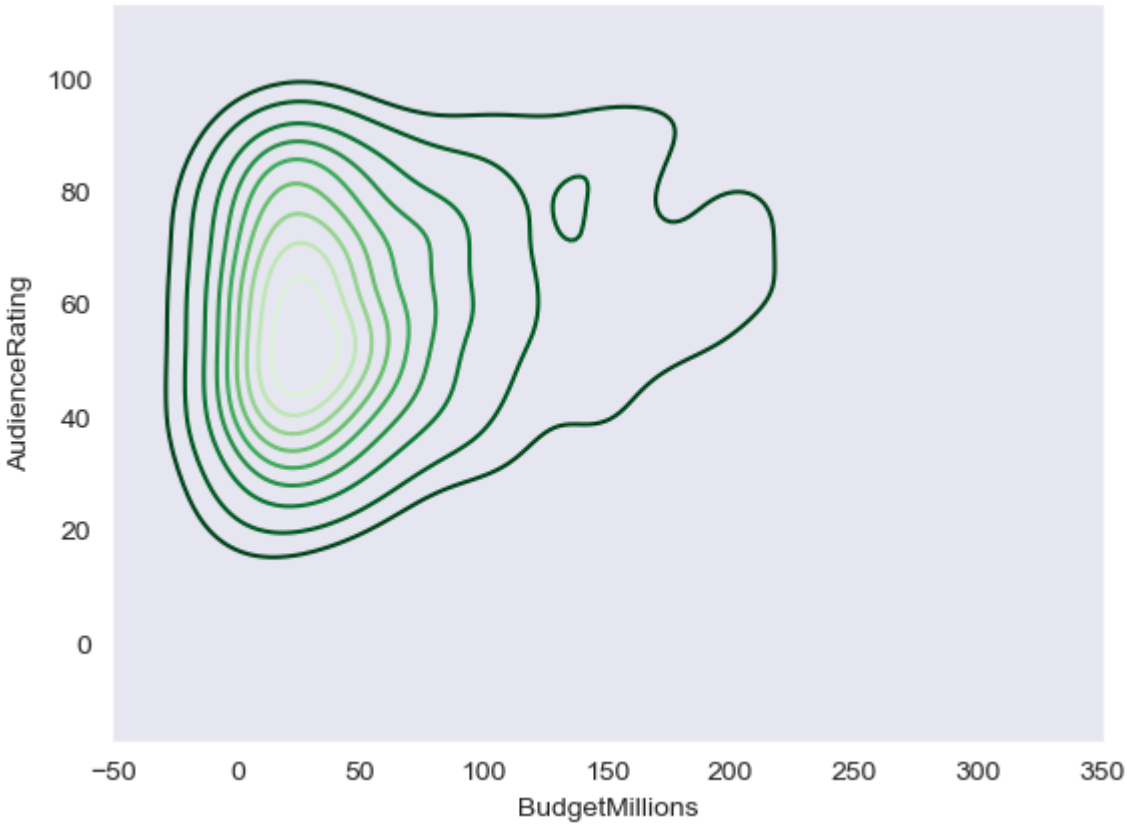
In [159…
```
k1 = sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating',shade=True,shade_lowest=Fals
```



In [161…
```
k1 = sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating',shade_lowest=False,cmap='Gre
```

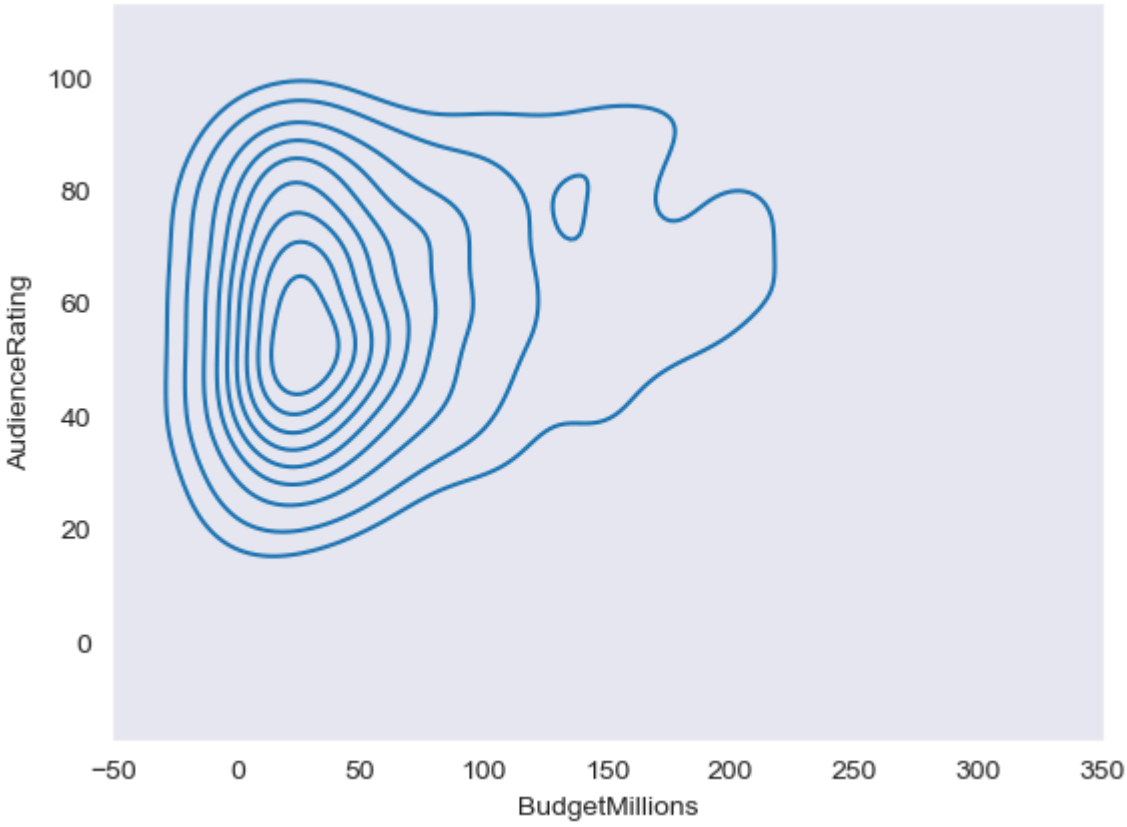


In [169…
```
sns.set_style('dark')
k1 = sns.kdeplot(data=movies,x='BudgetMillions',y='AudienceRating',shade_lowest=False,cmap='G
```



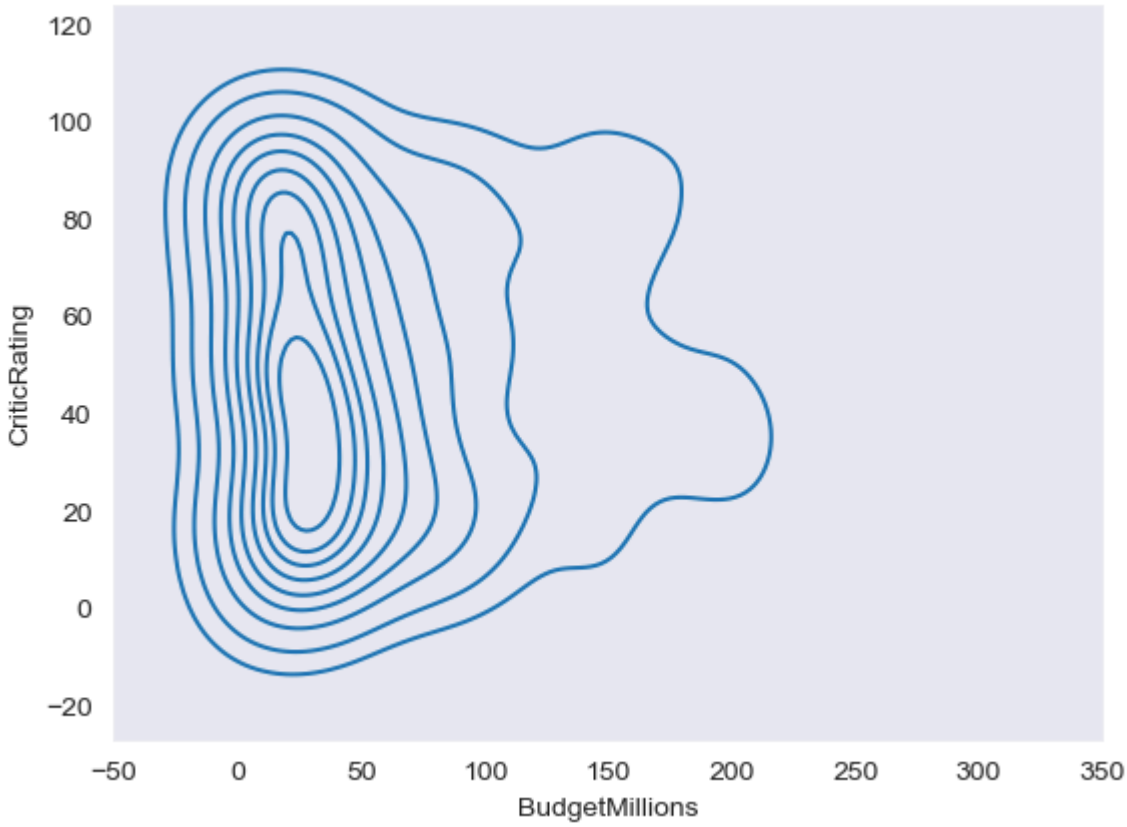
In [171…

```
sns.set_style('dark')
k1 = sns.kdeplot(data=movies,x='BudgetMillions',y='AudienceRating')
```



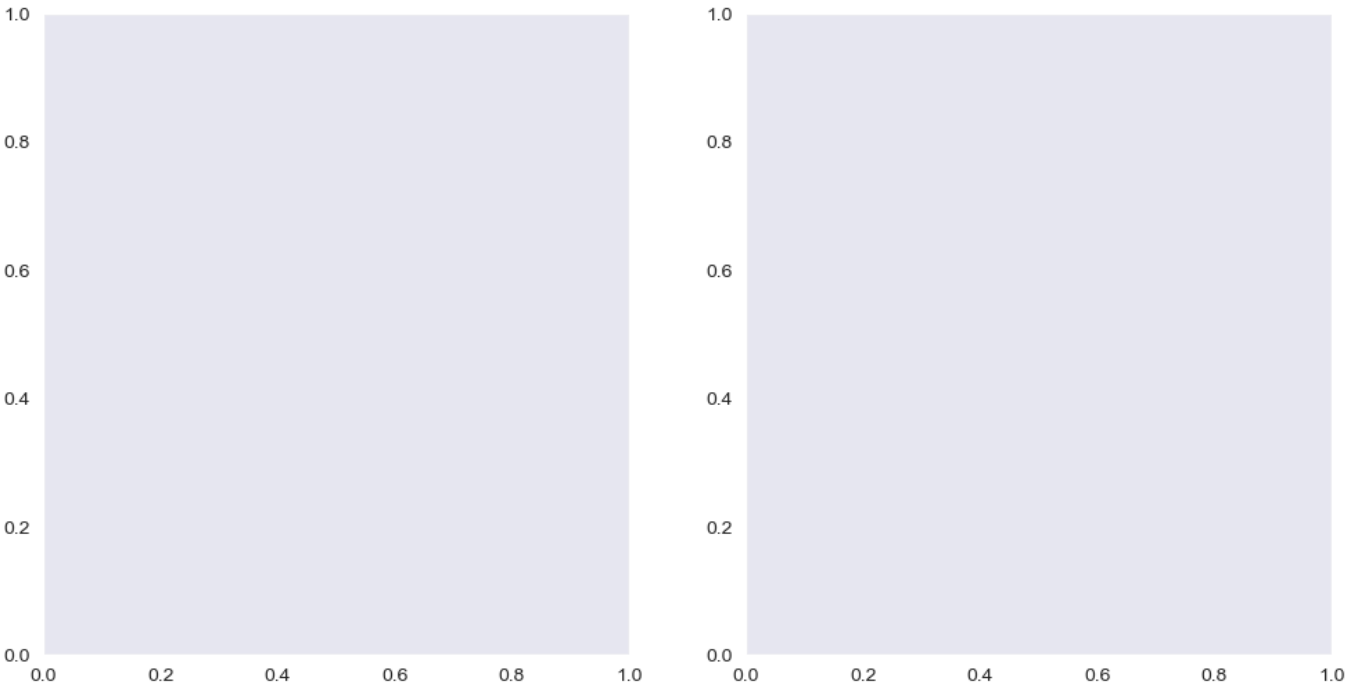
In [175…

```
k2 = sns.kdeplot(data=movies,x='BudgetMillions',y='CriticRating')
```



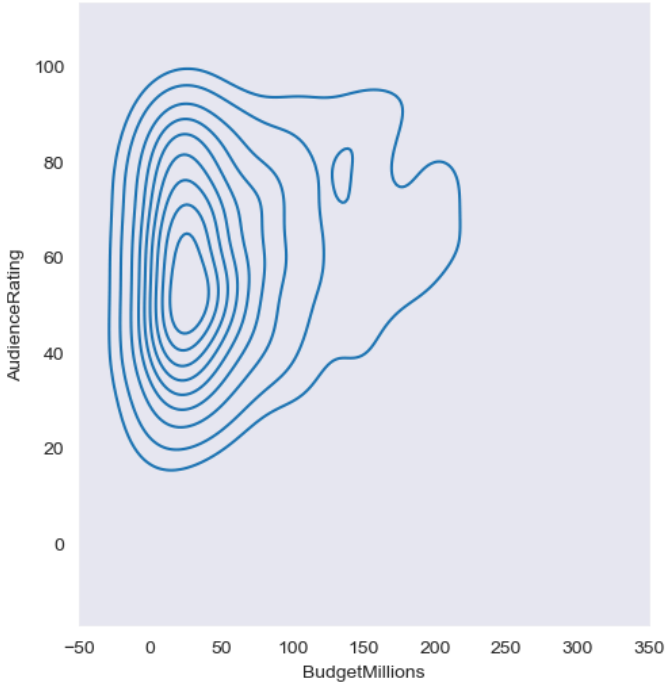
In [189…

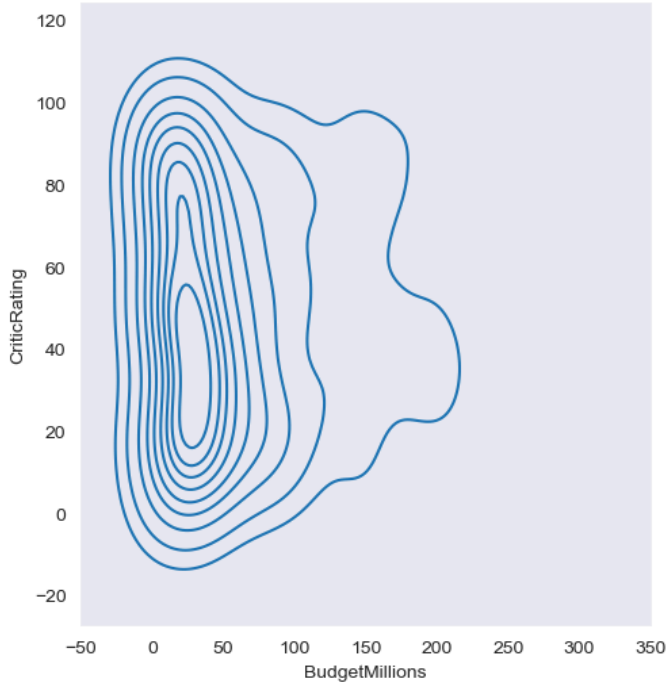
```
#subplots
f,ax=plt.subplots(1,2,figsize=(12,6))
```



In [193…

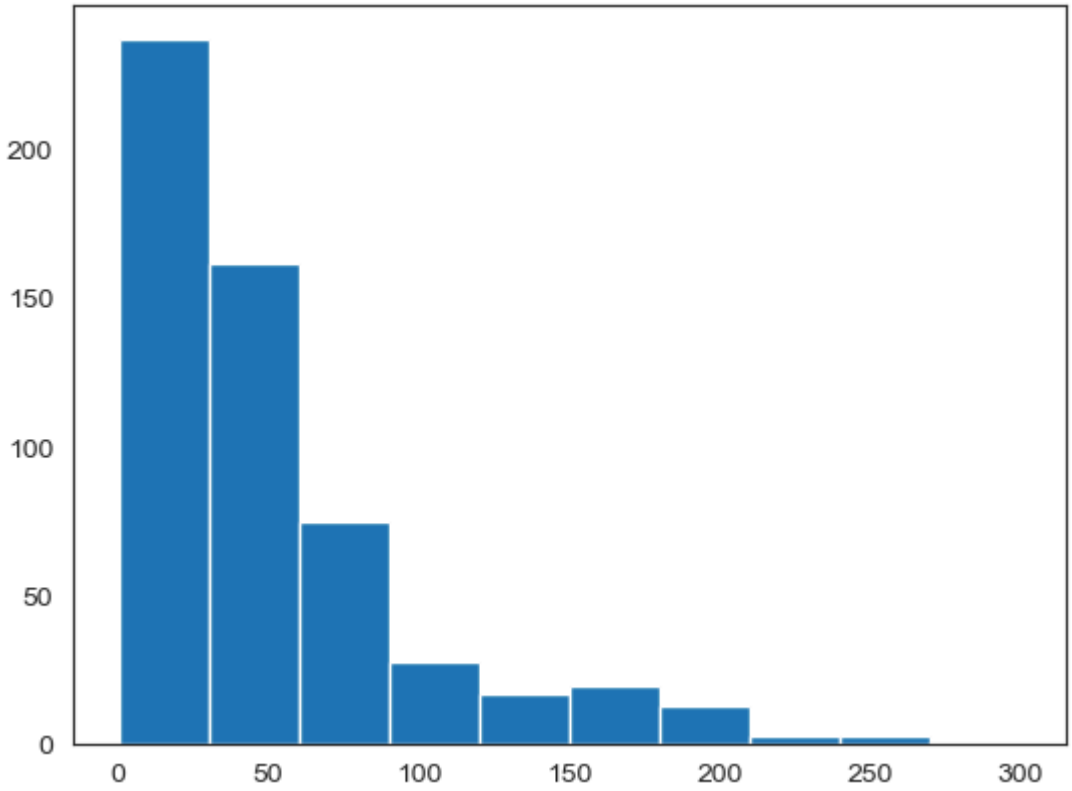
```
f,axes=plt.subplots(1,2,figsize=(12,6))
k1 = sns.kdeplot(data=movies,x='BudgetMillions',y='AudienceRating',ax=axes[0])
k2 = sns.kdeplot(data=movies,x='BudgetMillions',y='CriticRating',ax=axes[1])
```



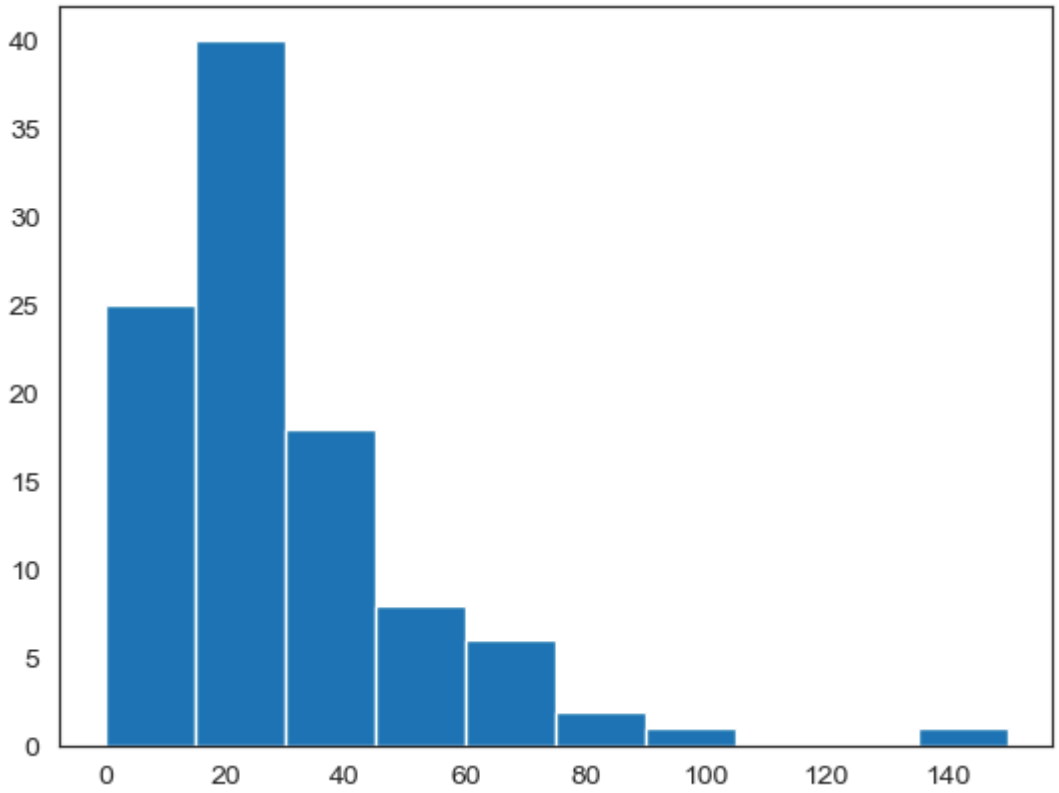


In [ ]:
```
#creating Stacked histogram
```

In [117…
```
#h1=plt.hist(movies.BudgetMillions)
plt.hist(movies.BudgetMillions)
plt.show()
```



In [119…
```
plt.hist(movies[movies.Genre=='Drama'].BudgetMillions)
plt.show()
```



In [121…
```
movies.head()
```

Out[121…

	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

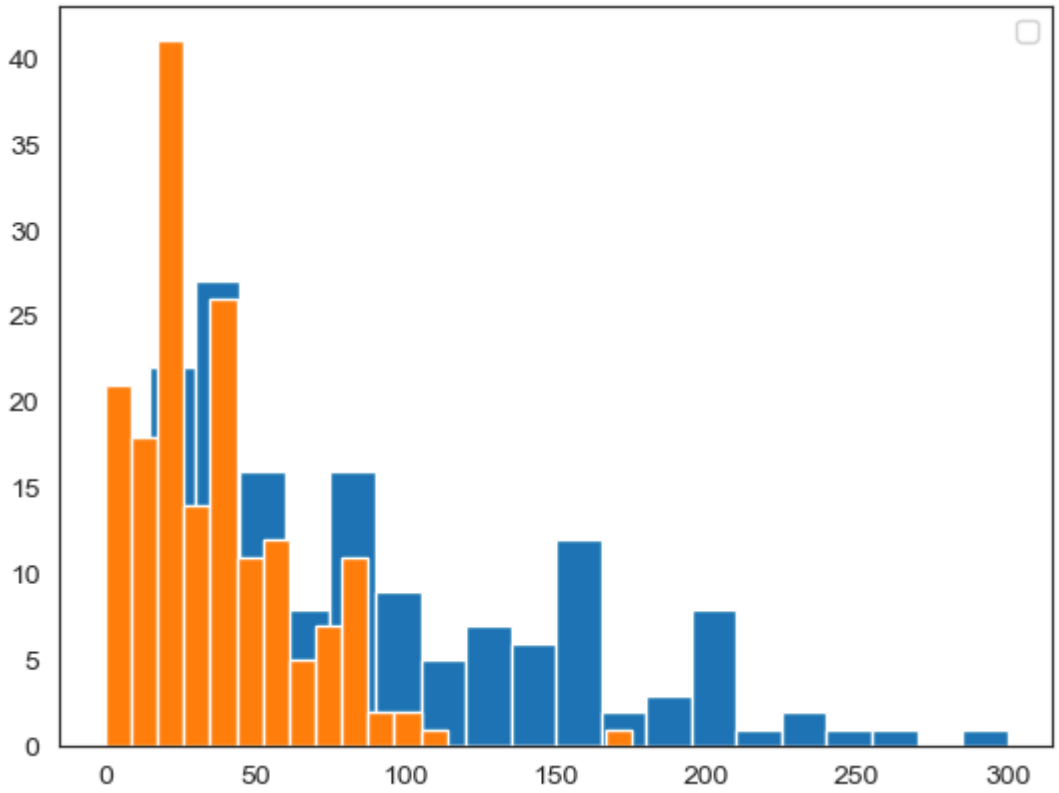
In [125…
```
movies.Genre.unique()
```

Out[125…
```
['Comedy', 'Adventure', 'Action', 'Horror', 'Drama', 'Romance', 'Thriller']
Categories (7, object): ['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance', 'Thri
ller']
```

In [ ]:
```
#below plots are stacked histogram and become ovverlapped
```

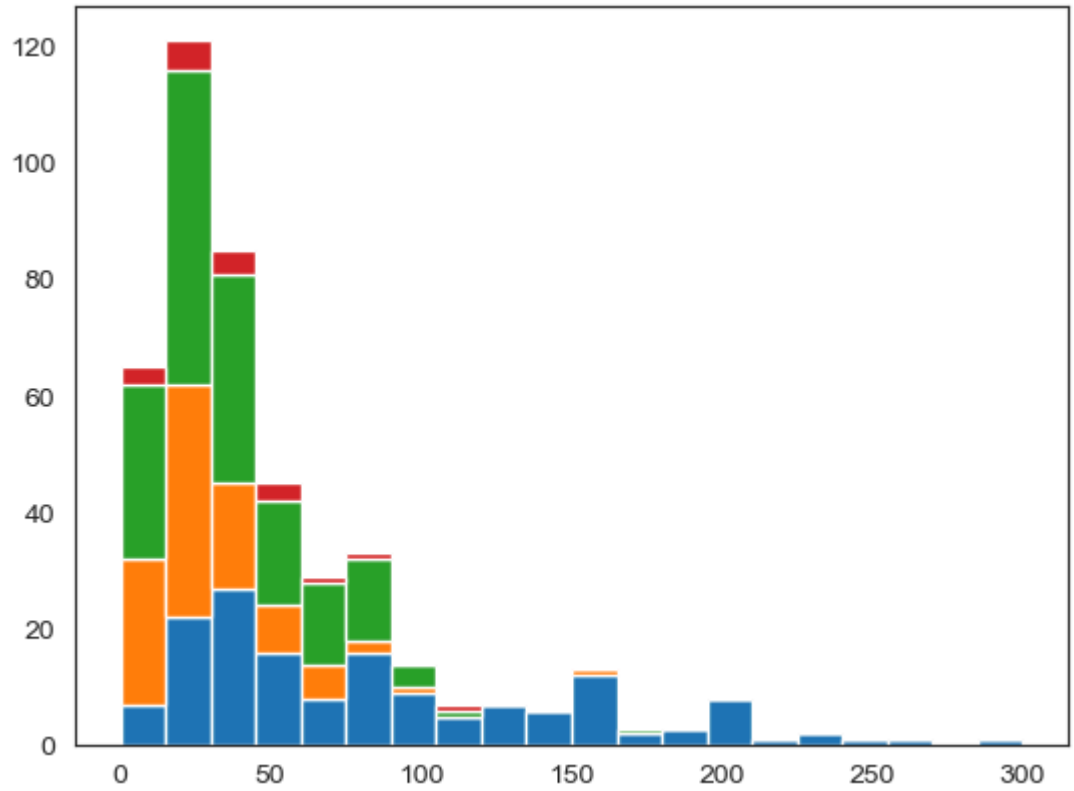
In [131…
```
plt.hist(movies[movies.Genre=='Action'].BudgetMillions,bins=20)
plt.hist(movies[movies.Genre=='Comedy'].BudgetMillions,bins=20)
plt.hist(movies[movies.Genre=='Horor'].BudgetMillions,bins=20)
plt.legend()
plt.show()
```

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



In [139…

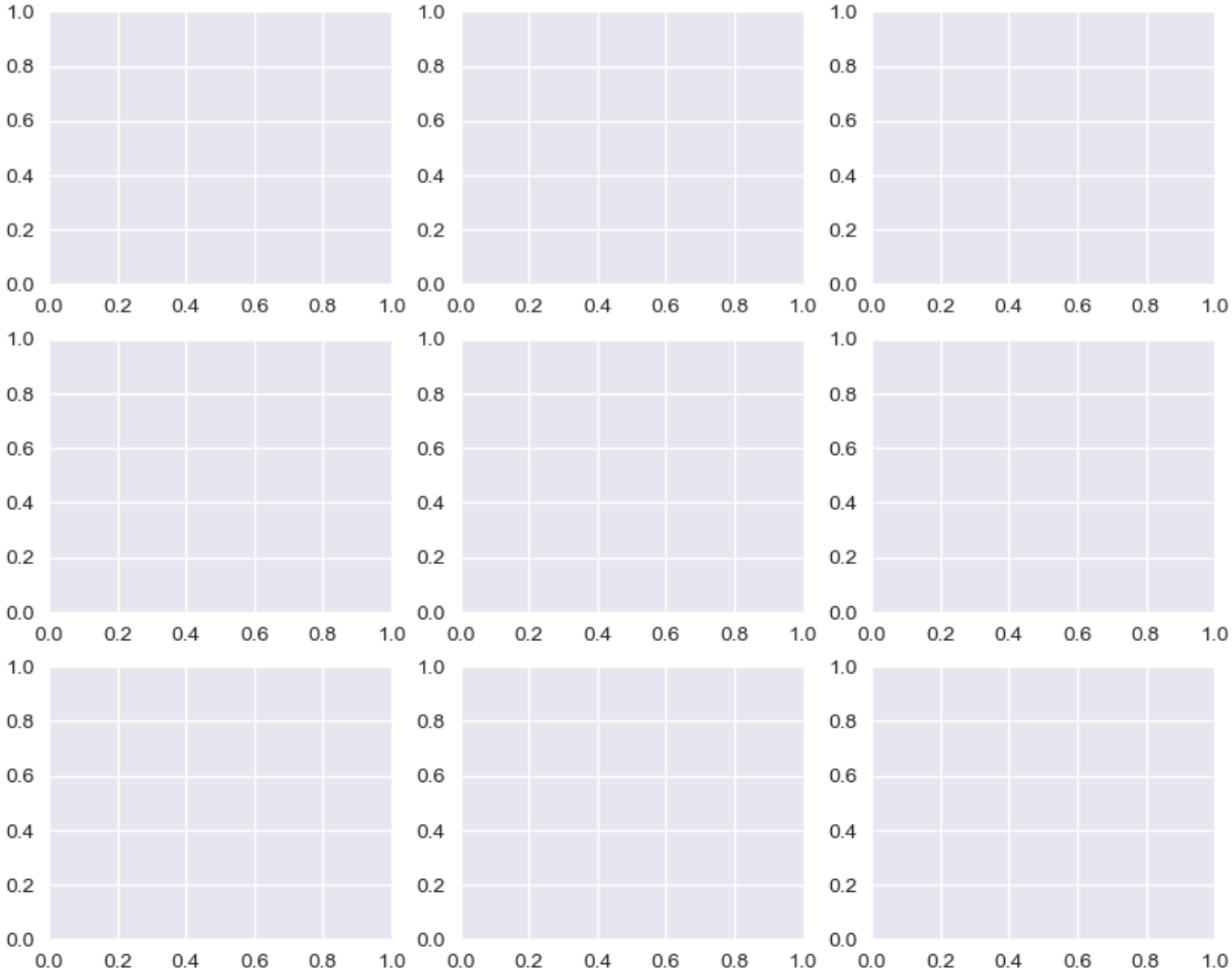
```
plt.hist([movies[movies.Genre=='Action'].BudgetMillions,\
          movies[movies.Genre=='Drama'].BudgetMillions,\
          movies[movies.Genre=='Comedy'].BudgetMillions,\
          movies[movies.Genre=='Romance'].BudgetMillions],
         bins=20,stacked=True)
plt.show()
```



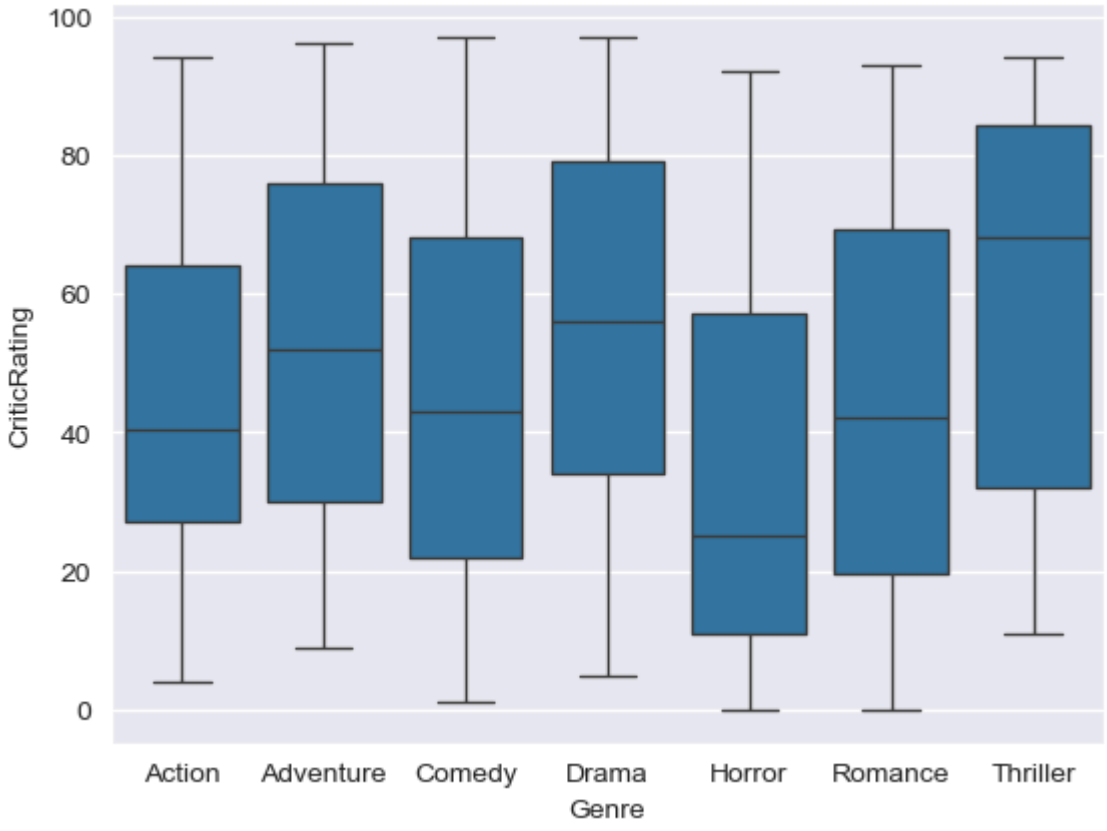
In [47]:

```
#Subplot
fig=plt.figure()
ax=plt.subplots(3,3,figsize=(10,8))
```

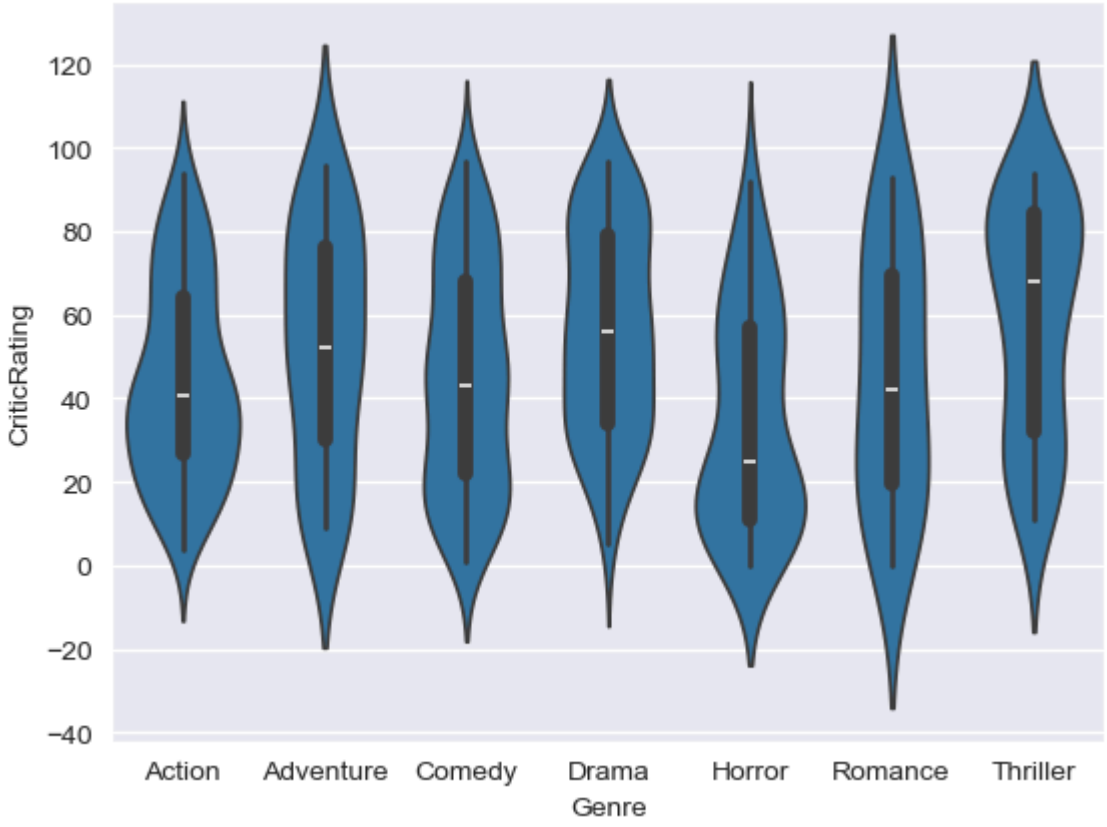
```
<Figure size 640x480 with 0 Axes>
```



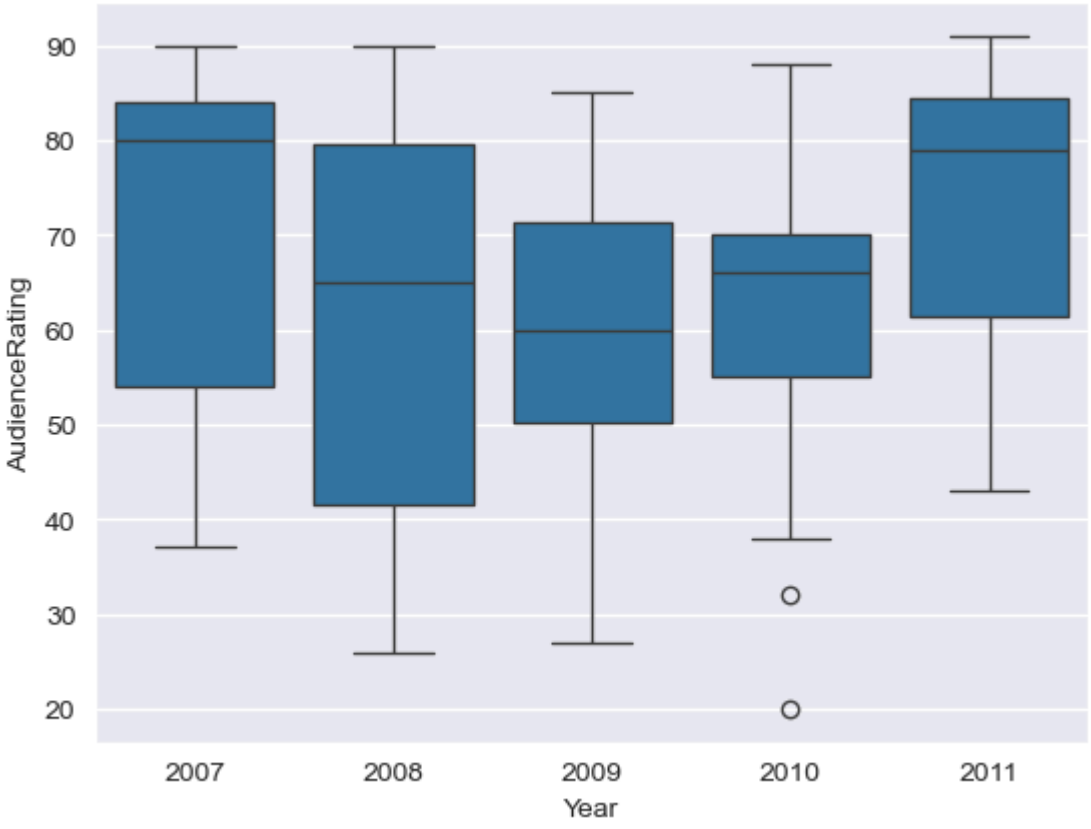
In [48]:
```
w=sns.boxplot(data=movies,x='Genre',y='CriticRating')
```



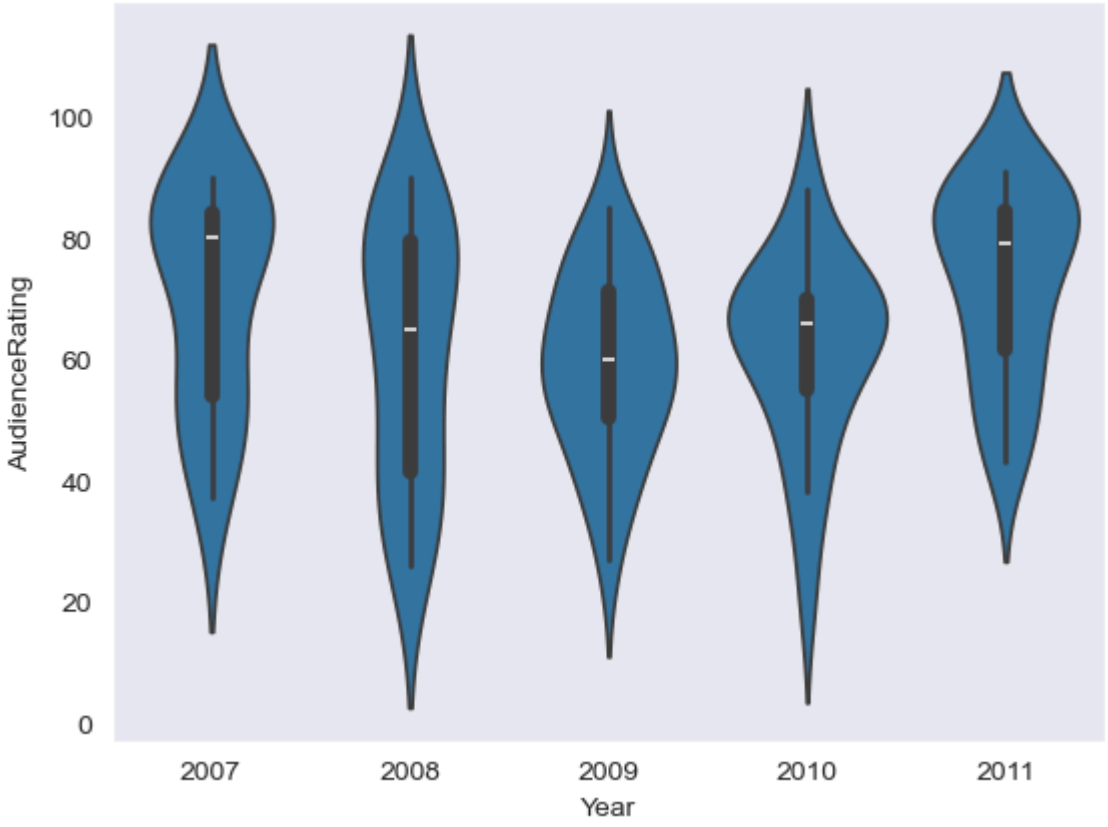
In [49]:
```
v=sns.violinplot(data=movies,x='Genre',y='CriticRating')
```



In [50]:
```
w1=sns.boxplot(data=movies[movies.Genre=='Drama'],x='Year',y='AudienceRating')
```

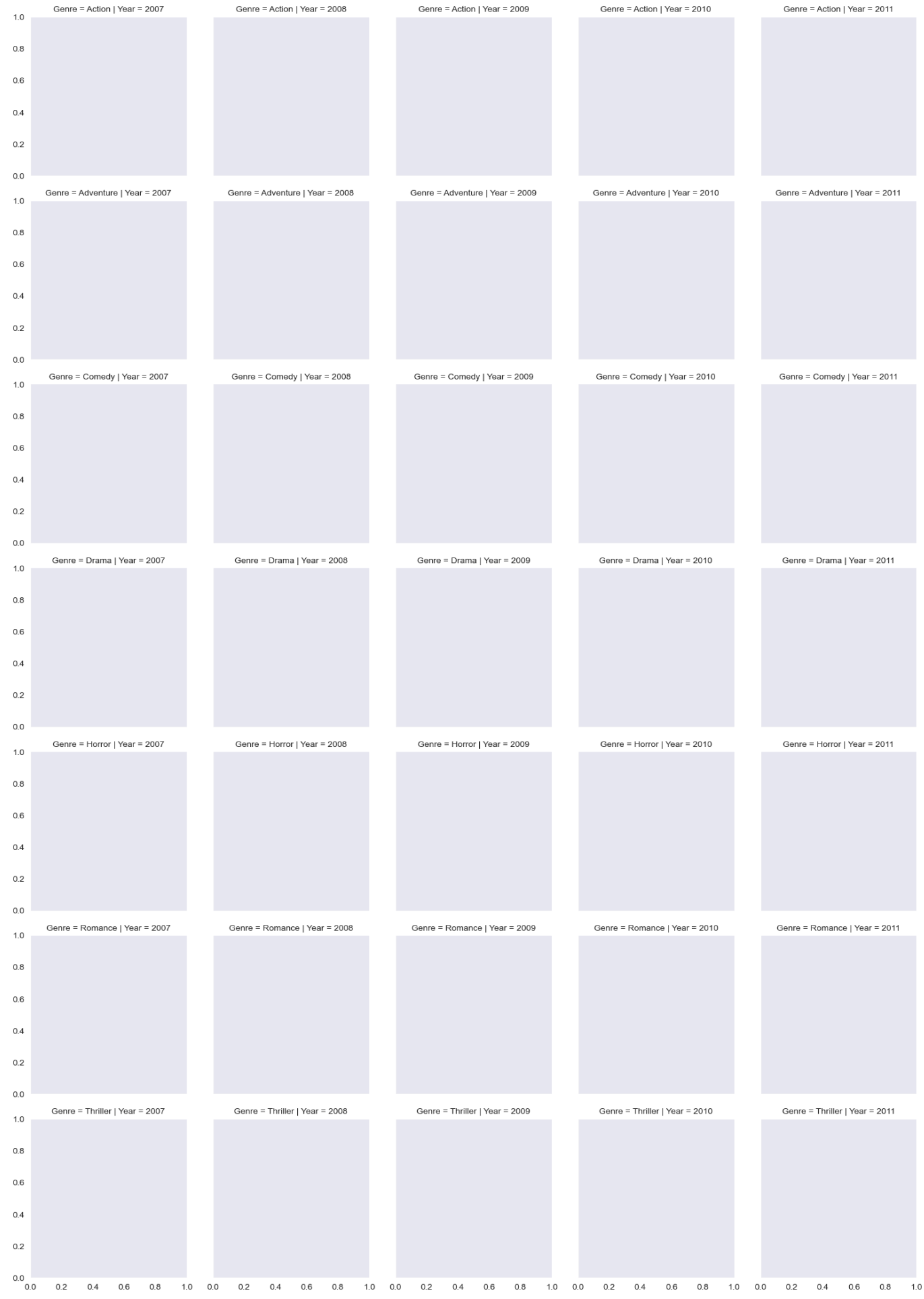


In [195…
```
w1=sns.violinplot(data=movies[movies.Genre=='Drama'],x='Year',y='AudienceRating')
```



In [197…

```
#Creating facet Grid
g=sns.FacetGrid(movies,row='Genre',col='Year',hue='Genre')#Kind of subplots
```

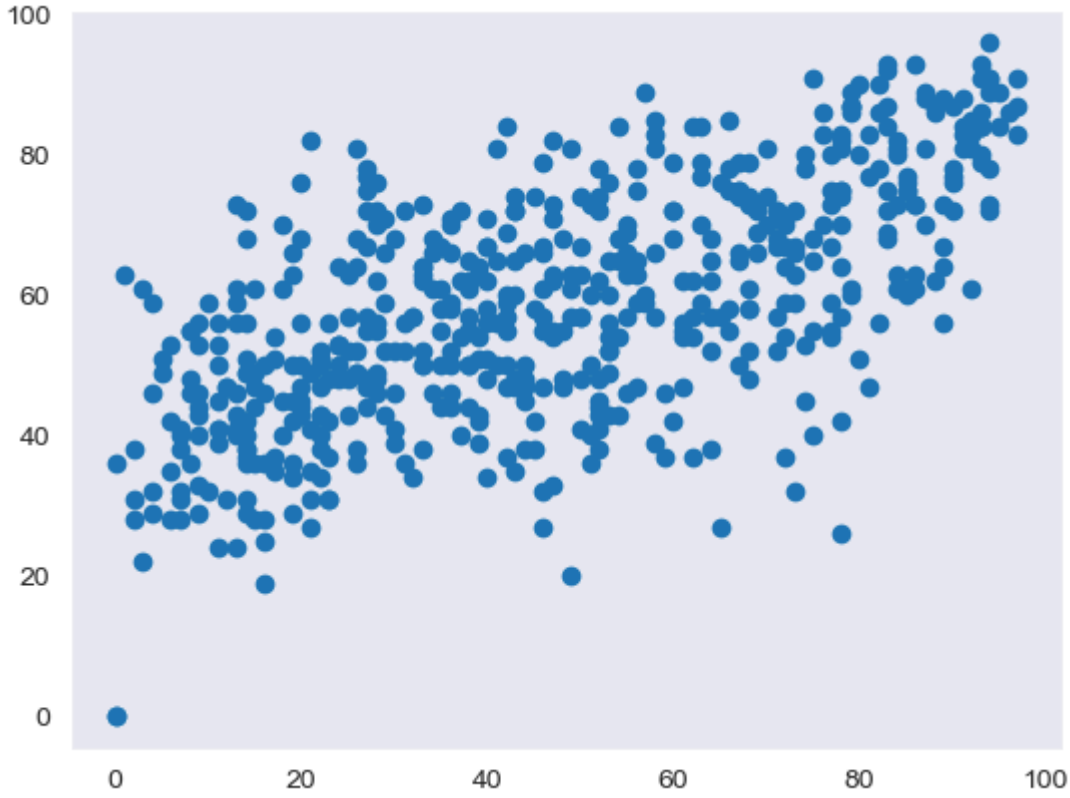


In [199…

```
plt.scatter(data=movies,x='CriticRating',y='AudienceRating')
```

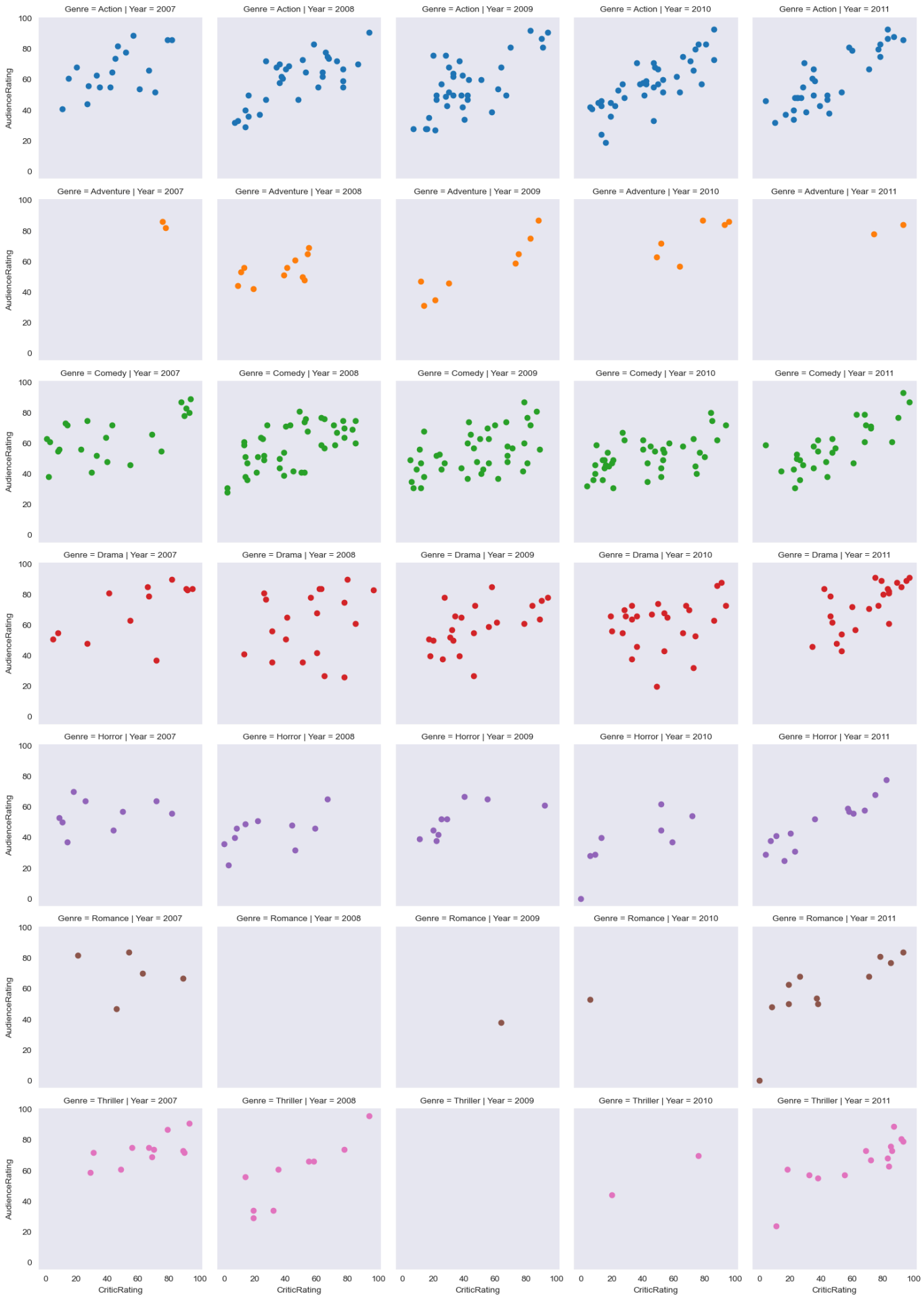
Out[199…

```
<matplotlib.collections.PathCollection at 0x1b4e7ca1af0>
```



In [209…

```
g=sns.FacetGrid(movies,row='Genre',col='Year',hue='Genre')#Kind of subplots
g=g.map(plt.scatter,'CriticRating','AudienceRating')
```

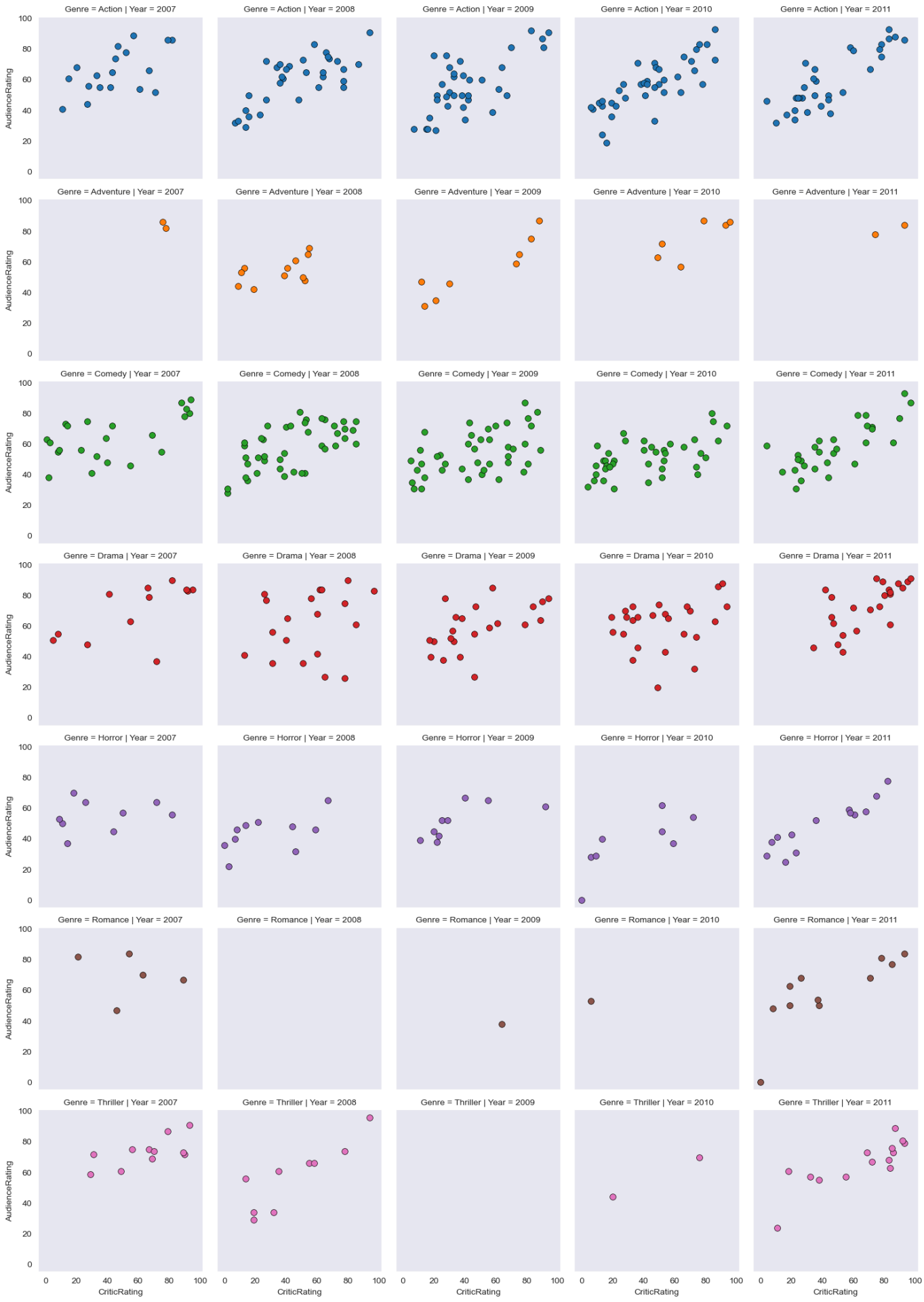


In [211…

```
#You can populate any type of chart
g=sns.FacetGrid(movies,row='Genre',col='Year',hue='Genre')
g=g.map(plt.hist,'BudgetMillions')
```

In [213…

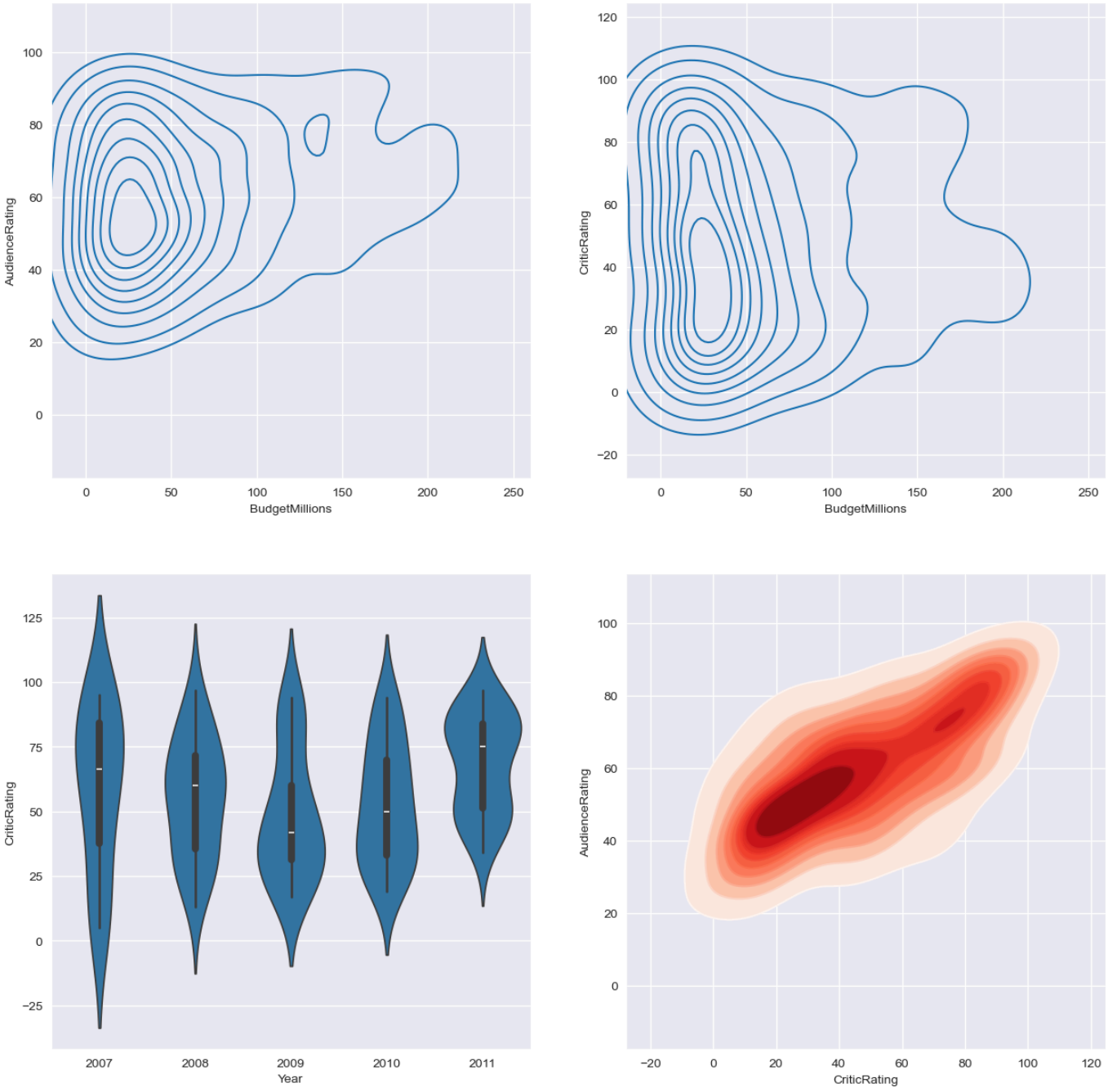
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
kws = dict(s=50, linewidth=0.5,edgecolor='black')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating',**kws )
```



In [257…

```
#Python is not vectorize Programming language
#Building dashboards(dashboards - combination of charts)

sns.set_style('darkgrid')
f,axes=plt.subplots(2,2,figsize=(15,15))

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

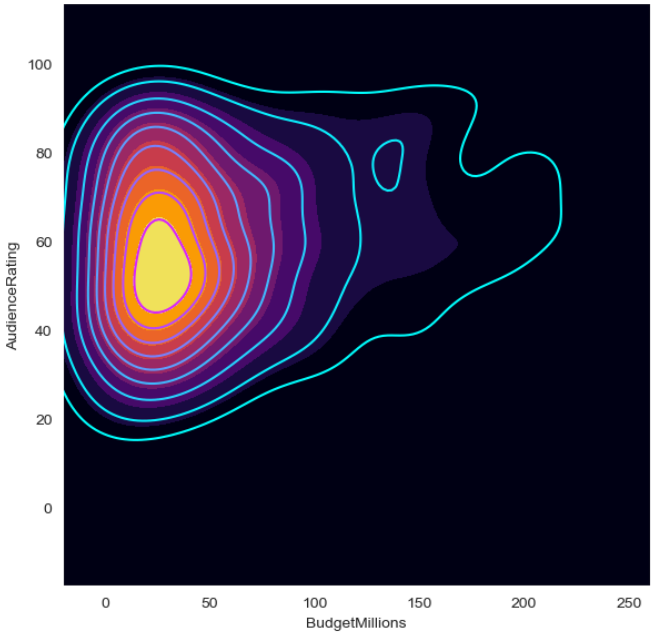
```
k1.set(xlim=(-20,260))
k2.set(xlim=(-20,260))
w1=sns.violinplot(data=movies[movies.Genre=='Drama'],x='Year',y='CriticRating',ax=axes[1,0])
k4=sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating',shade=True,shade_lowest=False,
k4b=sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating',cmap='Reds',ax=axes[1,1])
plt.show()
```

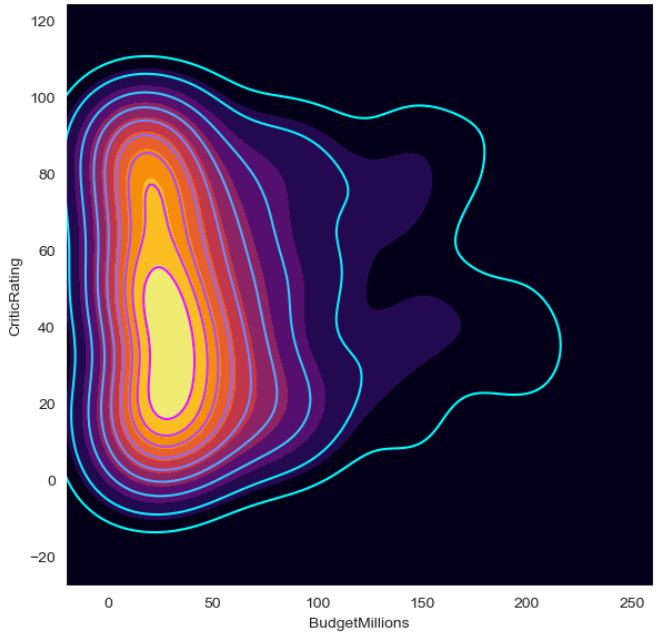


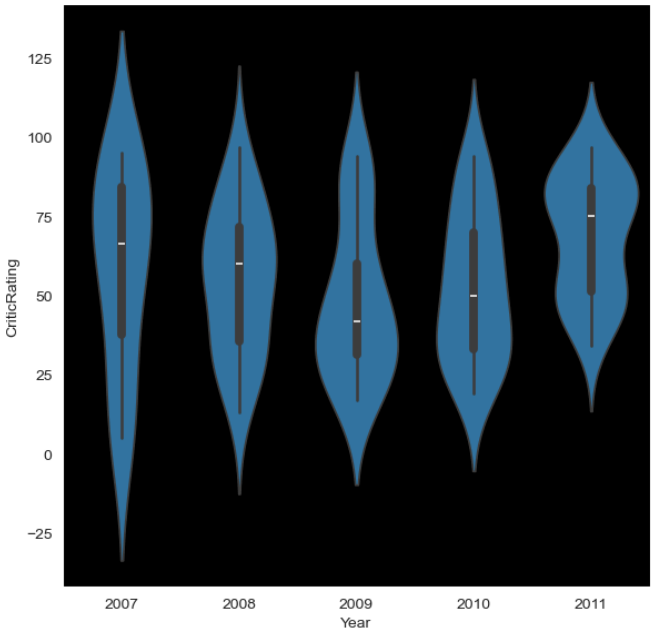
In [293…

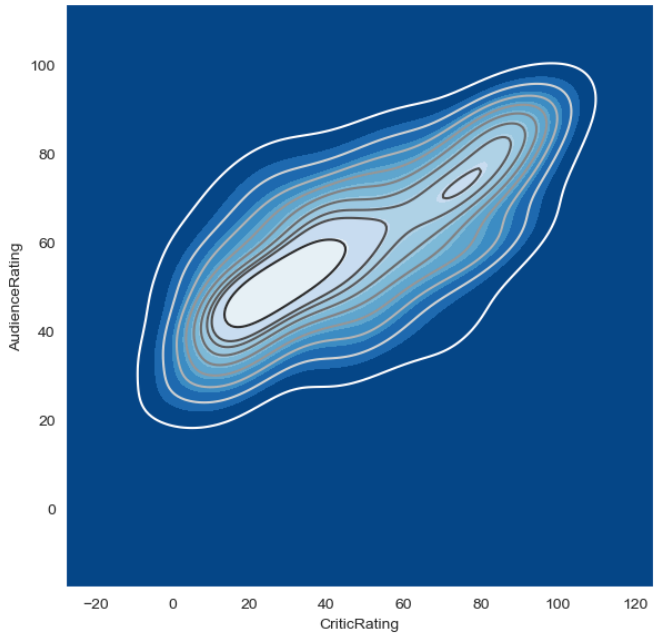
```
#How can you style your dashboard using different color map
#python is not vectorize programming language
#buildin dashboards
sns.set_style('dark',{'axes.facecolor':'black'})
f,axes=plt.subplots(2,2,figsize=(15,15))

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

#plot[0,1]
k2=sns.kdeplot(data=movies,x='BudgetMillions',y='CriticRating',\
               shade=True,shade_lowest=True,cmap='inferno',\
               ax=axes[0,1])
k2b=sns.kdeplot(data=movies,x='BudgetMillions',y='CriticRating',\
```

```
                cmap='cool',ax=axes[0,1])
#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(data=movies,x='CriticRating',y='AudienceRating',\
               shade=True,shade_lowest=True,cmap='Blues_r',\
               ax=axes[1,1])
k4b=sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating',\
                cmap='gist_gray_r',ax=axes[1,1])

k1.set(xlim=(-20,260))
k2.set(xlim=(-20,260))
plt.show()
```









In [ ]: