ADVANCED VISUALIZATION MOVIE RATINGS

Importing Required Libraries

In [106…
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os

%matplotlib inline

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

In [10]:
```
# if you want to change the working directory
os.getcwd()
```

Out[10]:
```
'C:\\Users\\kusha'
```

Reading Dataset

In [11]:
```
movies = pd.read_csv(r"C:\Users\kusha\OneDrive\Desktop\CSV Files\Movie-Rating.csv")
movies
```

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
...	...	...	...	...	...	...
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
```

Dataset Columns

In [12]:
```
movies.columns
```

Out[12]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

Removeing spaces & % removed noise characters

In [14]:
```
movies.columns = ['Film', 'Genre', 'CriticRating', 'AudienceRating','BudgetMillions','Ye
```

In [15]:
```
movies.head()
```

Out[15]:

	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

Information about dataset

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
```

Statistical Analysis

In [18]:
```
# if you look at the year the data type is int but when you look at the mean value it sh
# we have to change to categroy type
# also from object datatype we will convert to category datatypes

movies.describe()
```

Out[18]:

	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

Change Object data type to category

In [19]:
```
#movies['Audience Ratings %']
movies['Film']
```

Out[19]:
```
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 [20]:
```
movies.Film
```

Out[20]:
```
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 [21]:
```
movies.Film = movies.Film.astype('category')
```

In [22]:
```
movies.Film
```

Out[22]:
```
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 Ho
urs', ..., 'Youth in Revolt', 'Zodiac', 'Zombieland ', 'Zookeeper']
```

In [23]:
```
movies.head()
```

Out[23]:

	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 [24]:
```
# now the same thing we will change genra to category & year to category
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 [25]:
```
movies.Genre = movies.Genre.astype('category')
movies.Year = movies.Year.astype('category')
```

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

Out[26]:
```
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',
'Thriller']
```

In [27]:
```
movies.Year # is it real no. year you can take average,min,max but out come have no mean
```

Out[27]:
```
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 [28]:
```
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 [29]:
```
movies.Genre.cat.categories
```

Out[29]:
```
Index(['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance',
       'Thriller'],
      dtype='object')
```

In [31]:
```
# Now when you see the describt you will get only integer value mean, standard deviation
movies.describe()
```

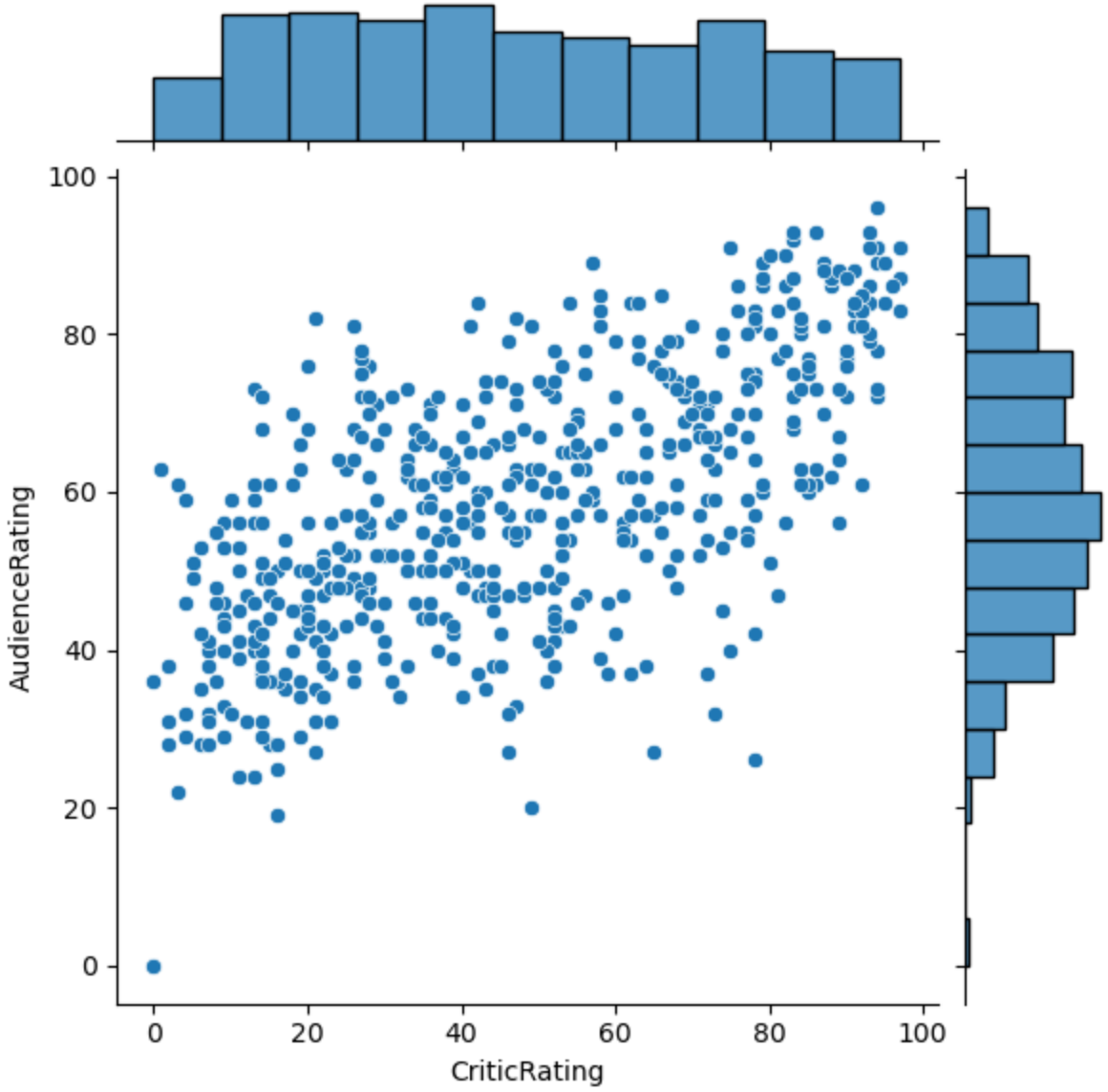
Out[31]:

	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

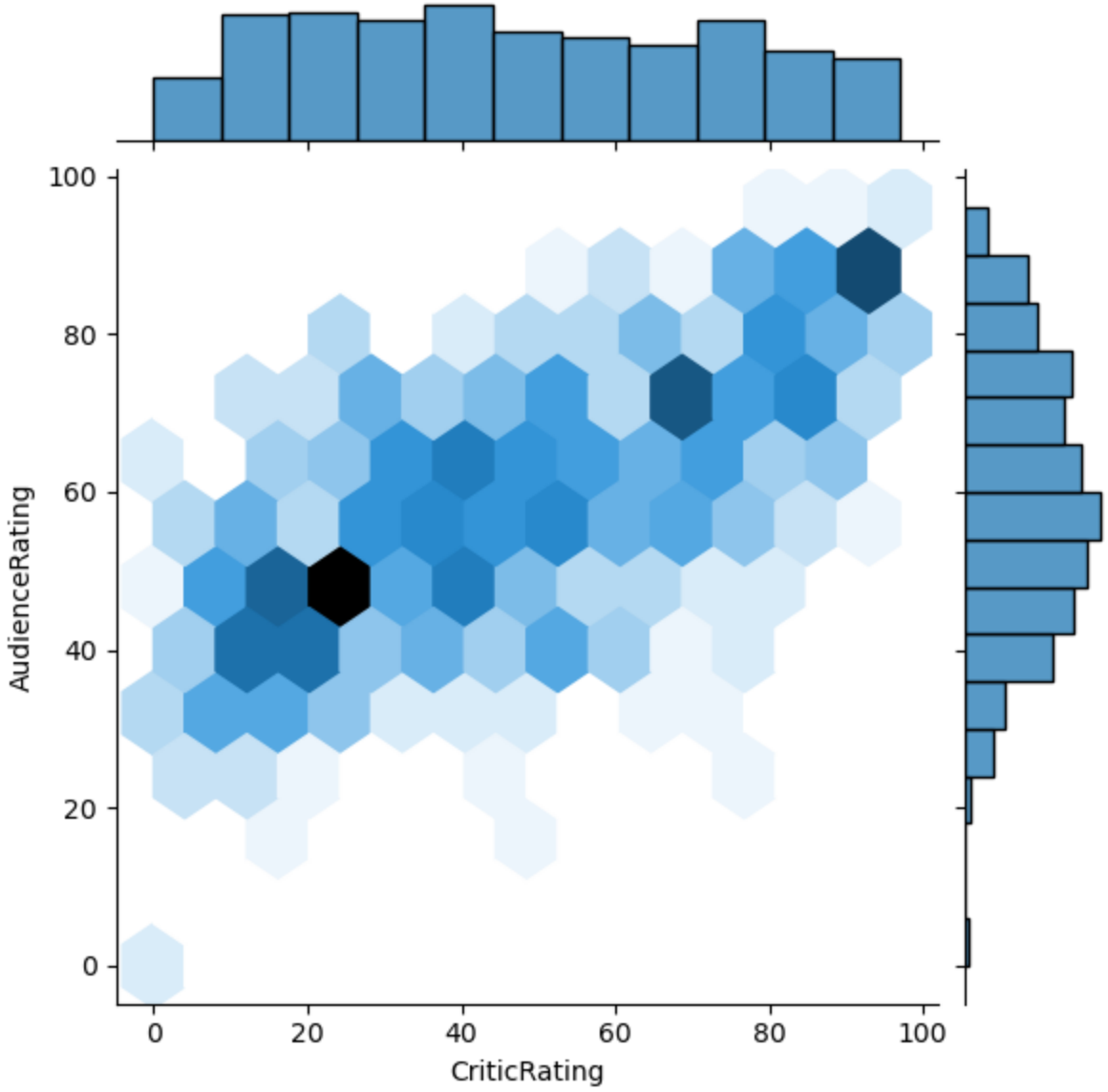
Joint plots

- 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)

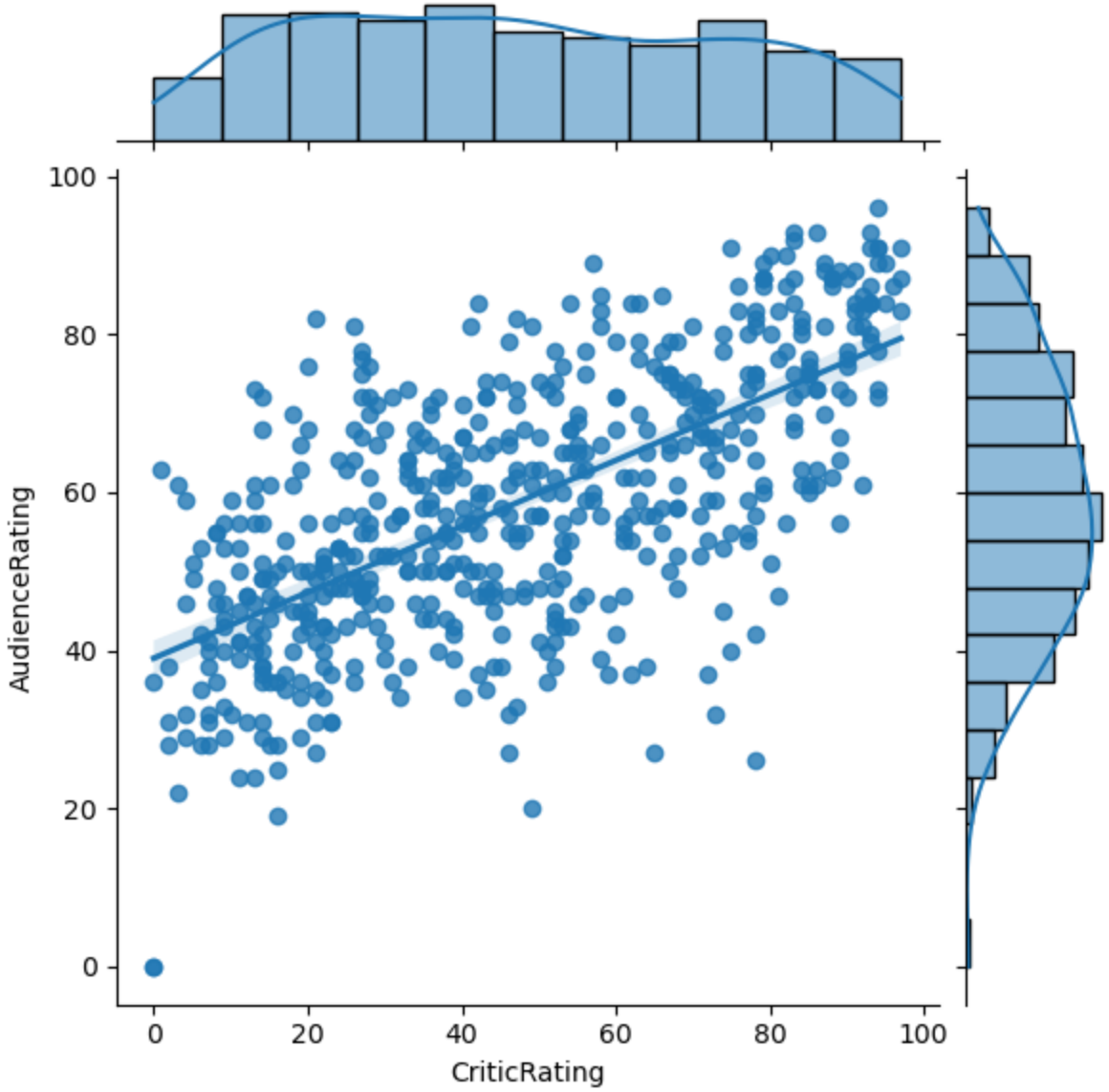
In [32]:
```
# Audience rating is more dominant then critics rating
# Based on this we find out as most people are most liklihood to watch audience rating &
# let me explain the excel - if you filter audience rating & critic rating. critic ratin
j = sns.jointplot( data = movies, x = 'CriticRating', y = 'AudienceRating')
```



In [36]:
```
# Hexagonal plot
j = sns.jointplot( data = movies, x = 'CriticRating', y = 'AudienceRating', kind='hex')
```

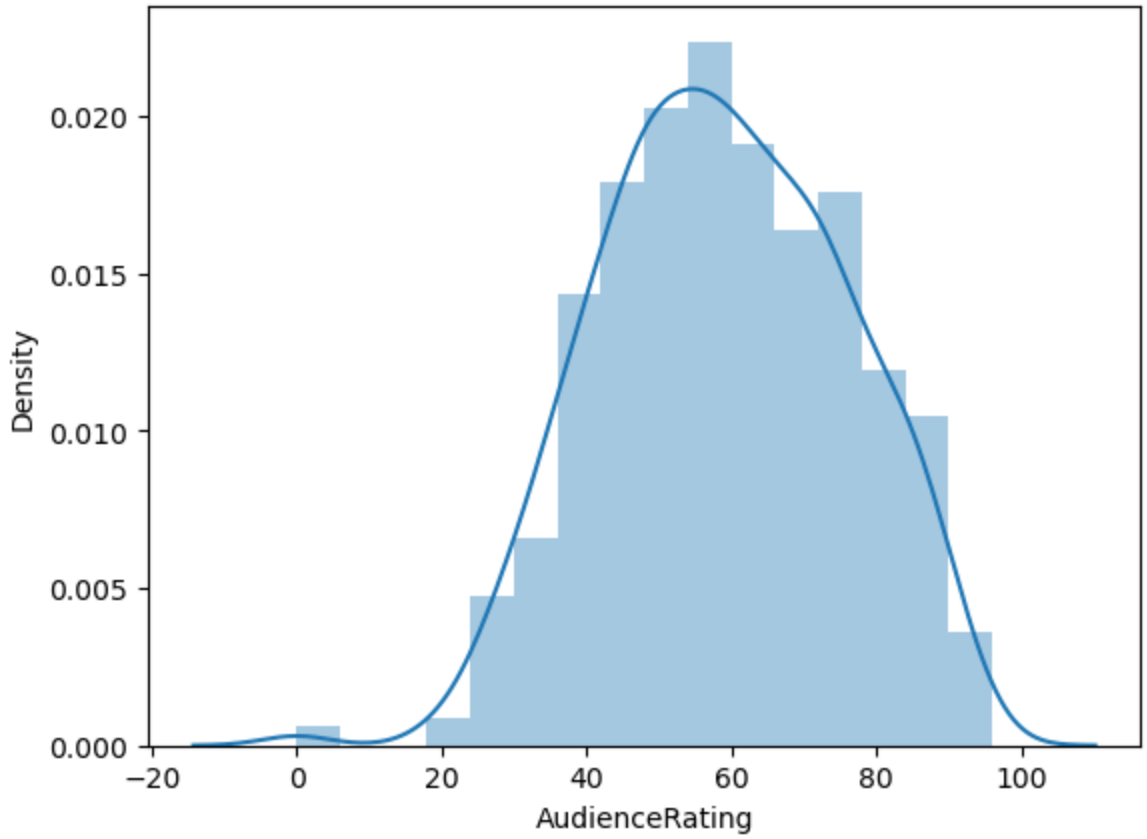


In [37]:
```
# Regression plot
j = sns.jointplot( data = movies, x = 'CriticRating', y = 'AudienceRating', kind='reg')
```

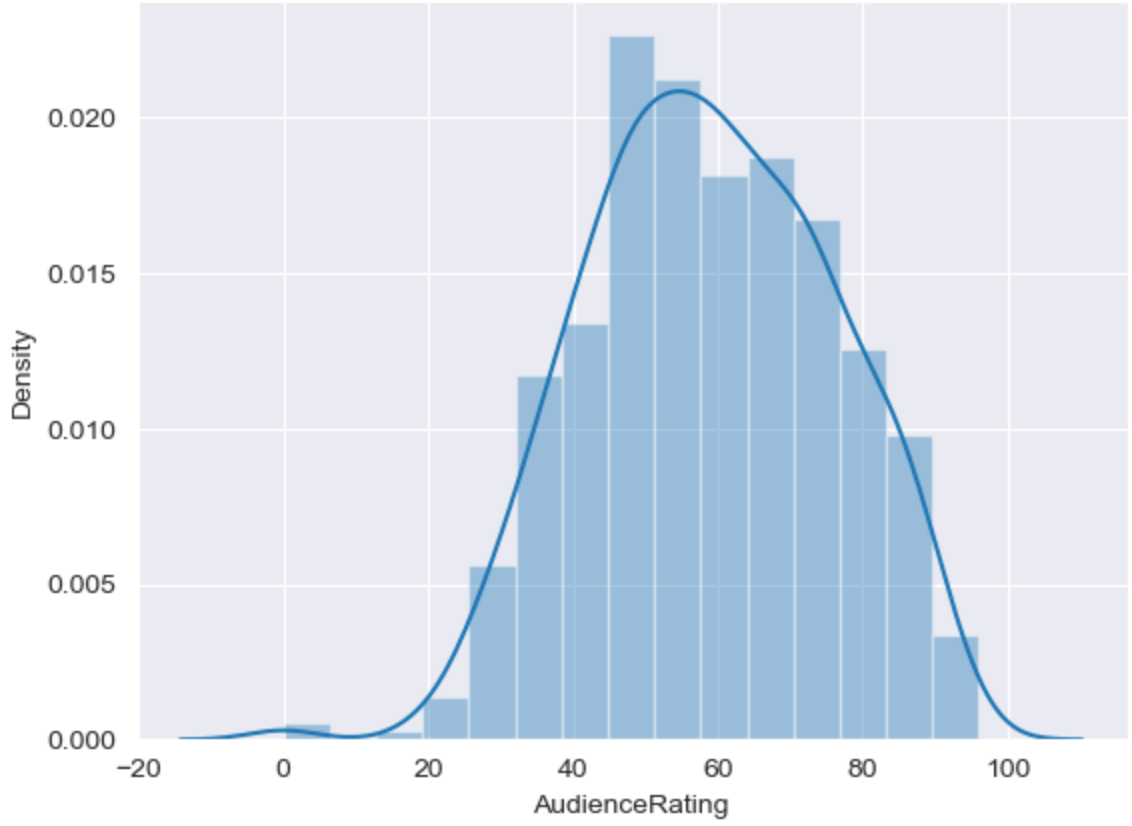


Histograms

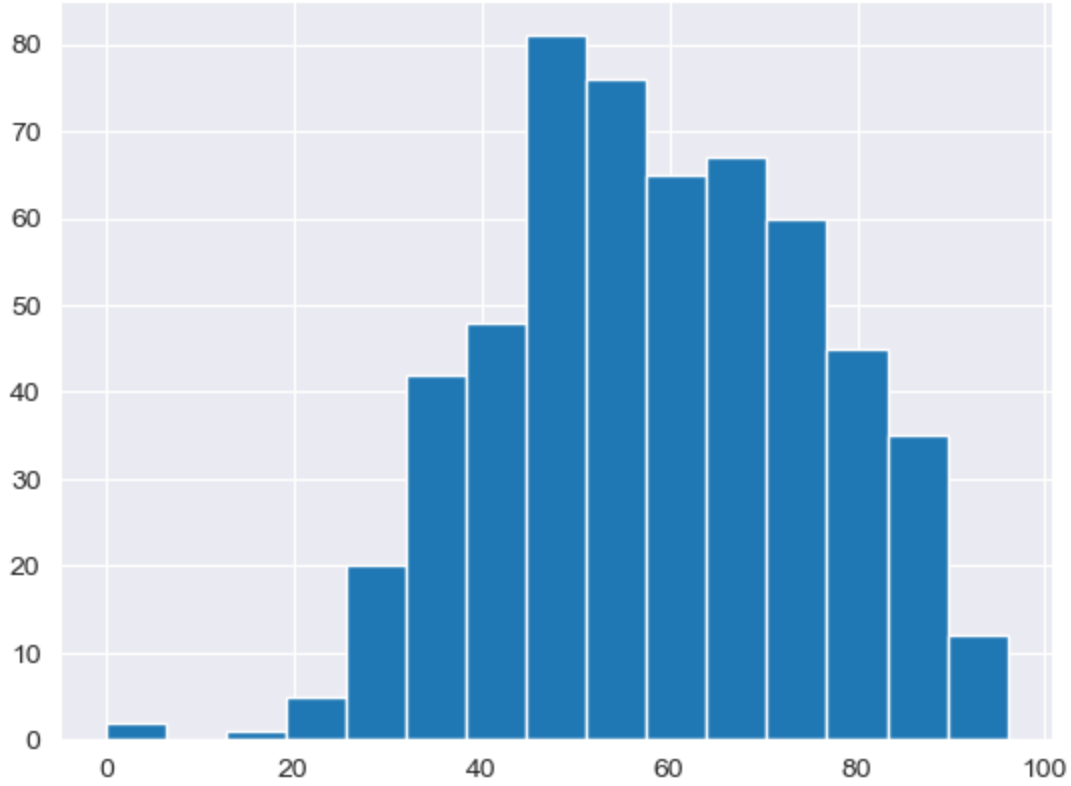
In [38]:
```
# Chat 1
# y-axis generated by seaborn automatically that is the powefull of seaborn gallery
m1 = sns.distplot(movies.AudienceRating)
```



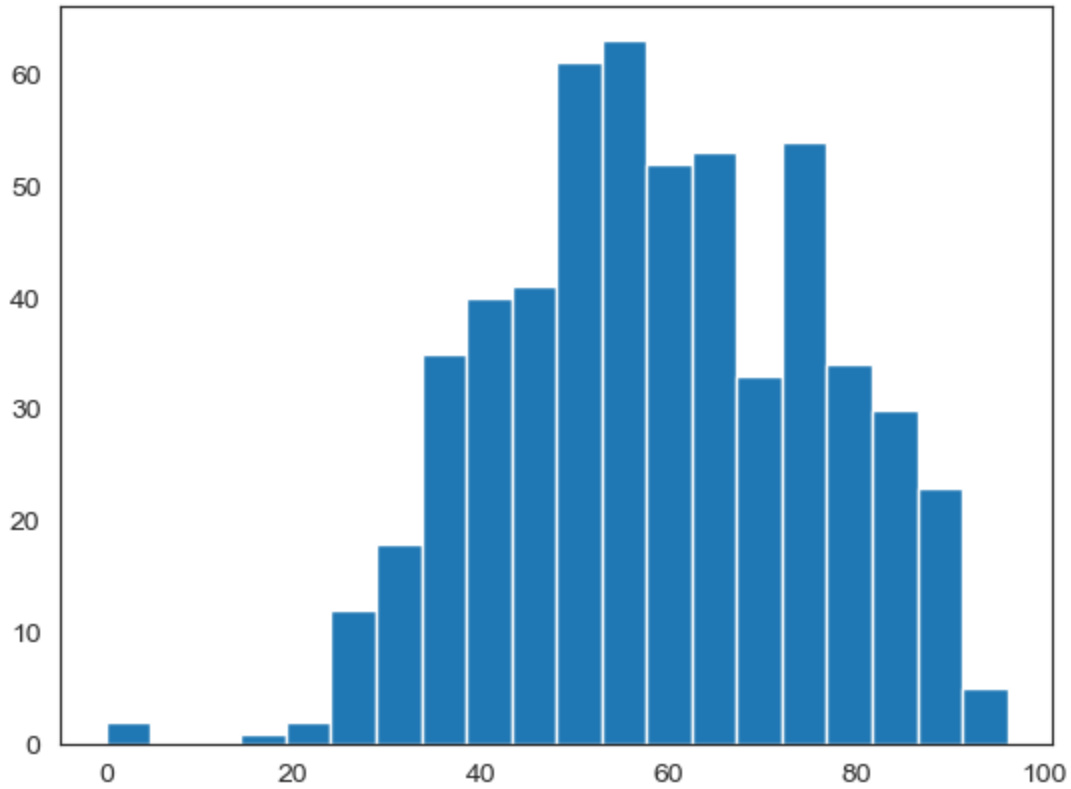
In [39]:
```
# Set the grid
sns.set_style('darkgrid')
m2 = sns.distplot(movies.AudienceRating, bins = 15)
```



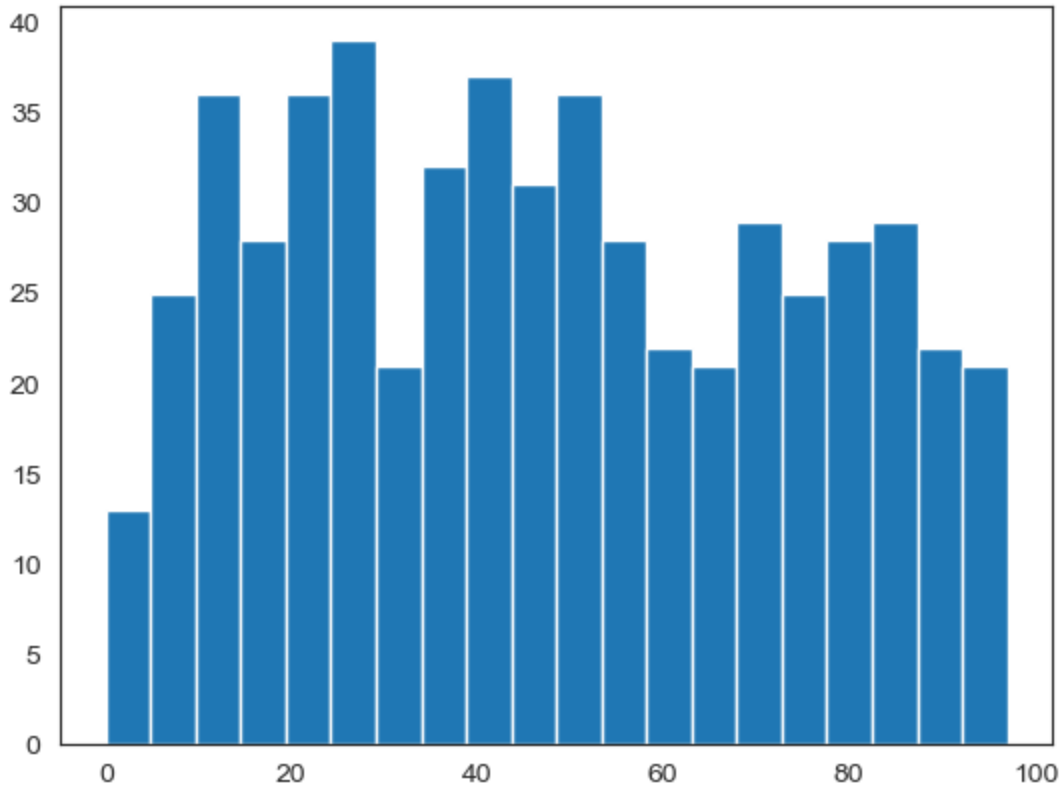
In [40]:
```
sns.set_style('darkgrid')
n1 = plt.hist(movies.AudienceRating, bins=15)
```



In [41]:
```
#normal distribution & called as bell curve
sns.set_style('white')
n1 = plt.hist(movies.AudienceRating, bins=20)
```



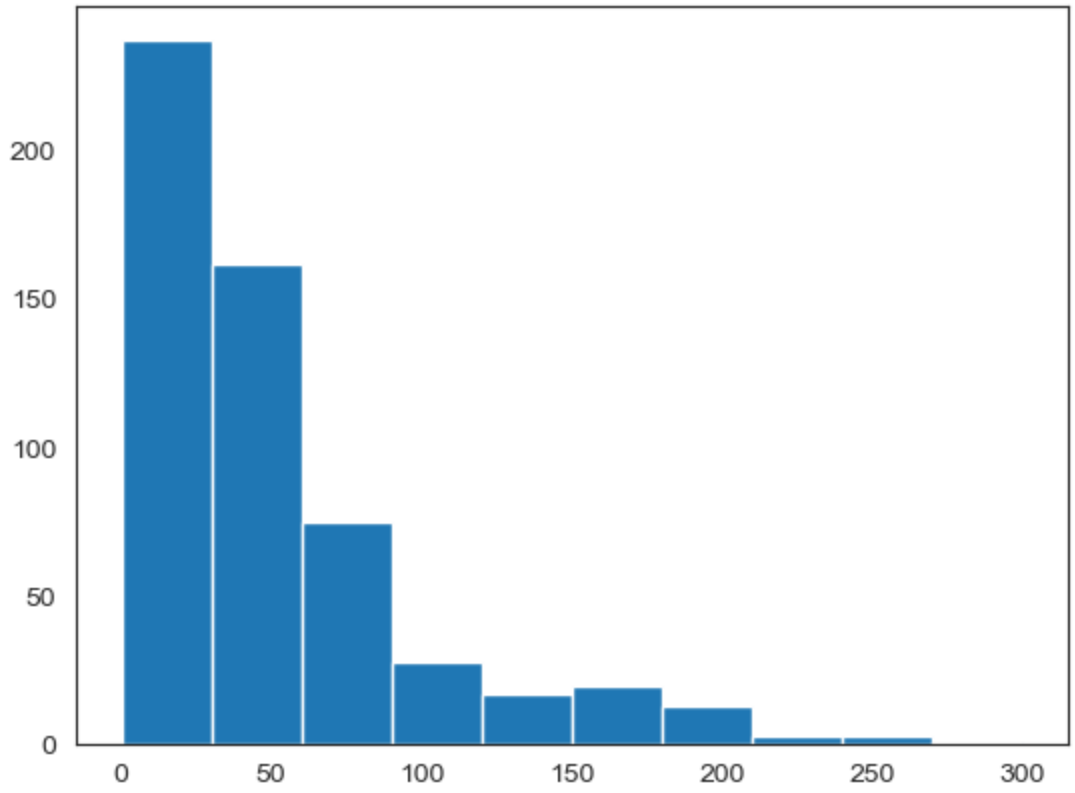
In [42]:
```
n1 = plt.hist(movies.CriticRating, bins=20) # uniform distribution
```



Chat - 2

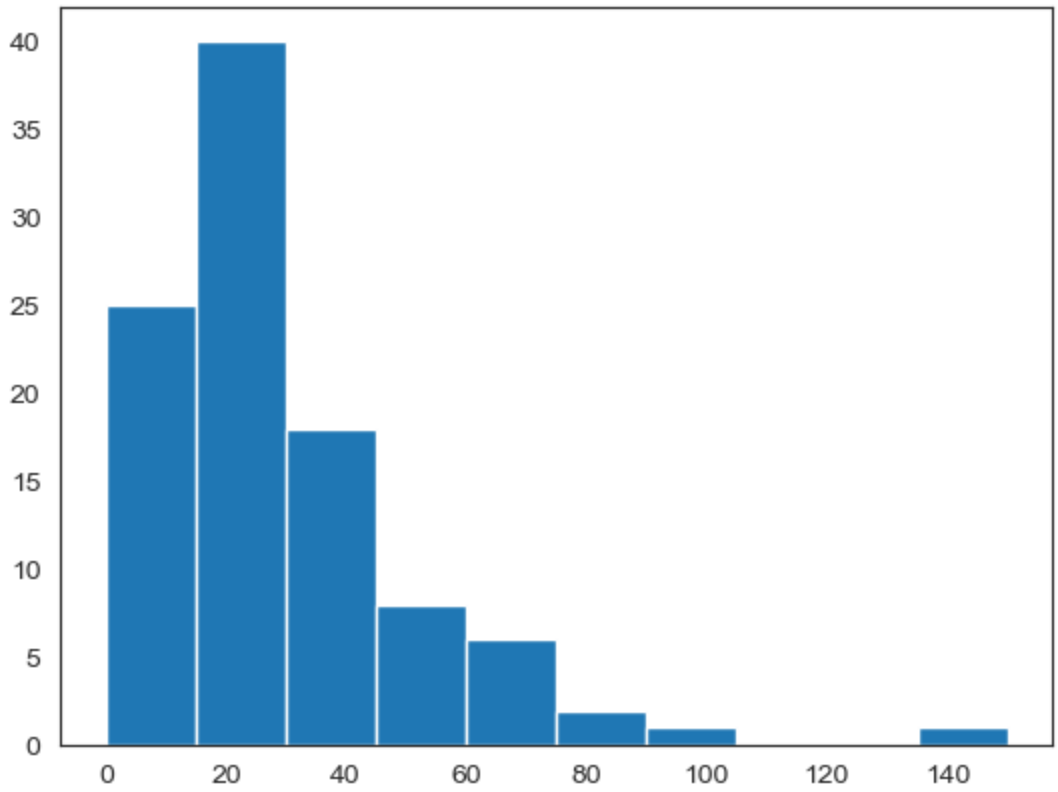
Creating stacked histograms & this is bit tough to understand

In [43]:
```
#h1 = plt.hist(movies.BudgetMillions)

plt.hist(movies.BudgetMillions)
plt.show()
```



In [44]:
```
# Plot Budget movie genrewise
```

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



In [45]:
```
movies.head()
```

Out[45]:

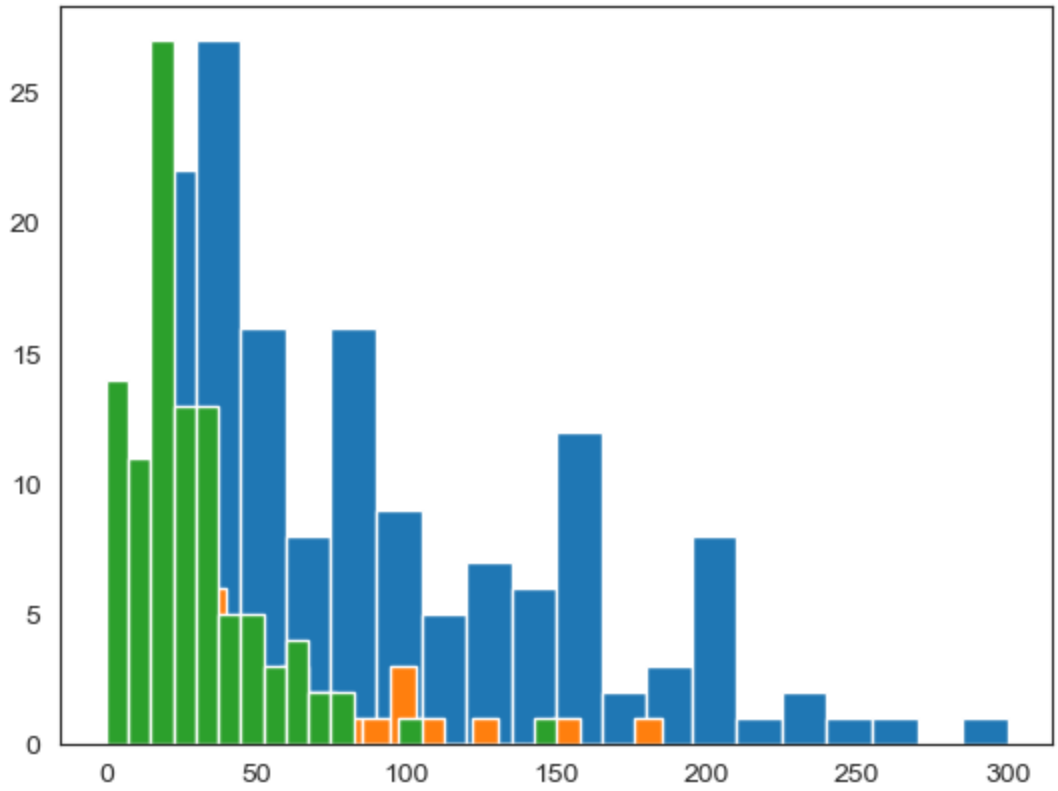
	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009

In [46]:
```
movies.Genre.unique()
```

Out[46]:
```
['Comedy', 'Adventure', 'Action', 'Horror', 'Drama', 'Romance', 'Thriller']
Categories (7, object): ['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance',
'Thriller']
```

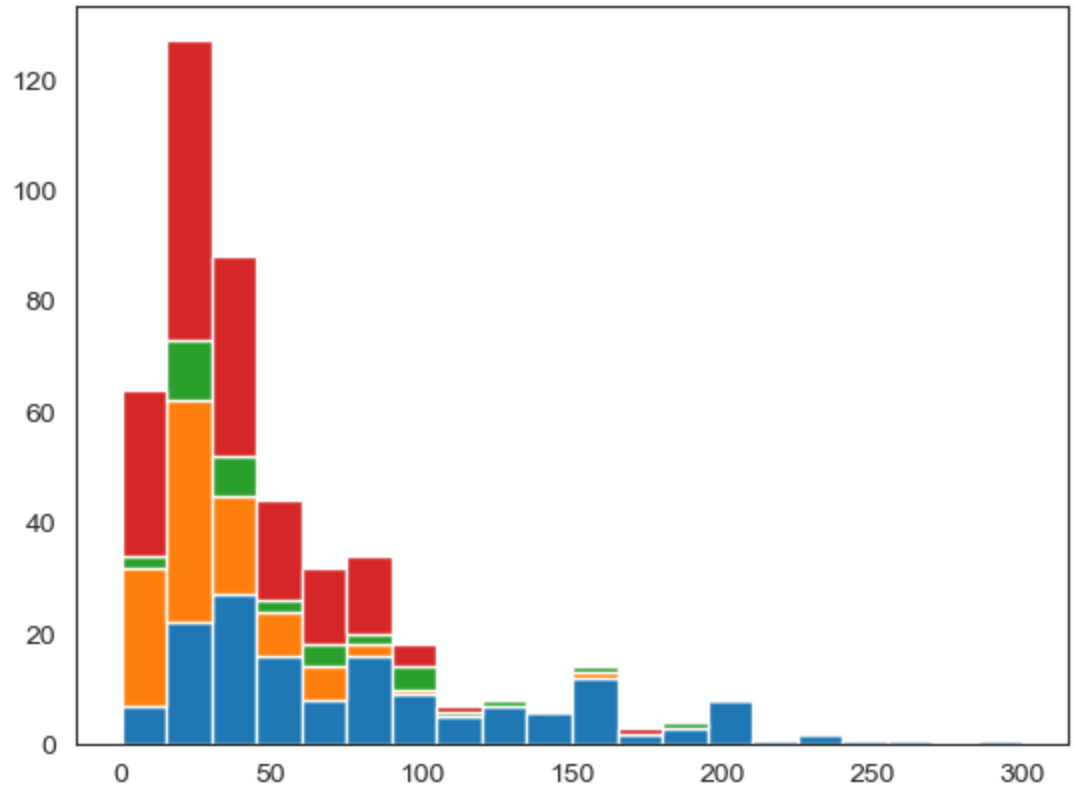
Below plots are stacked histogram becuase overlaped

In [49]:
```
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()
```



That's why we put stacked = True

In [48]:
```
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()
```



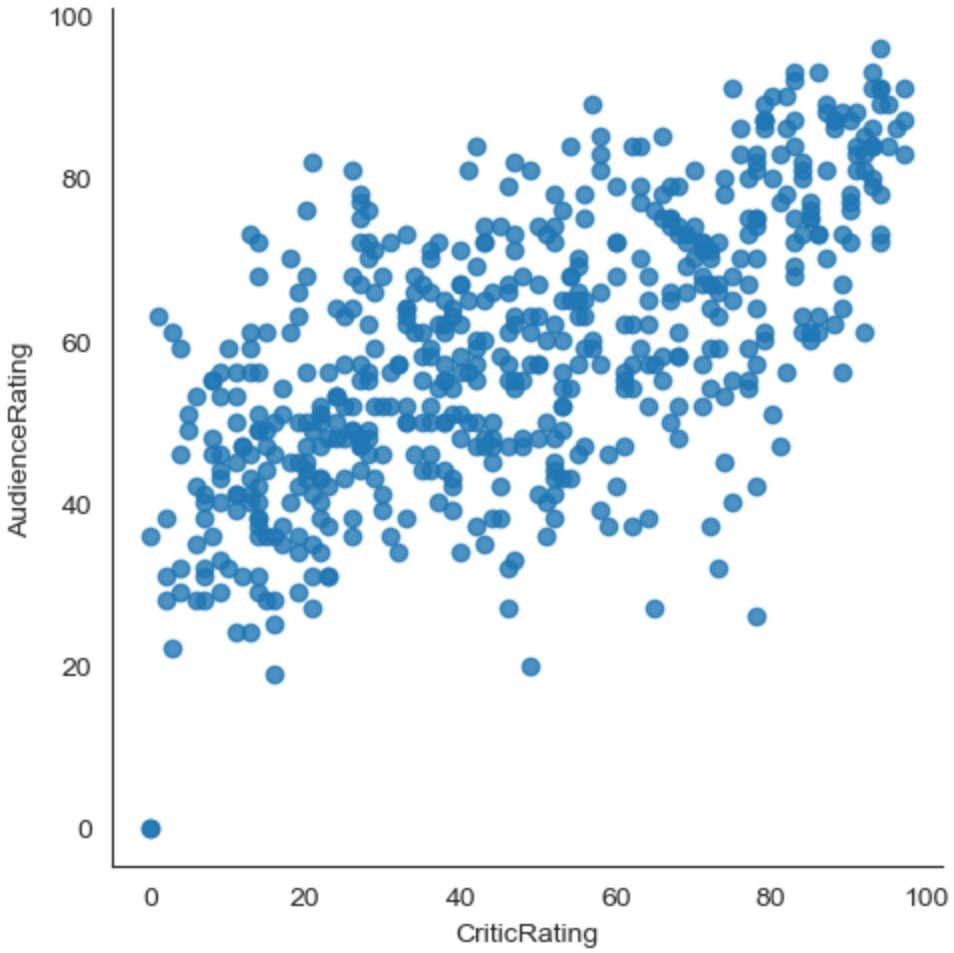
In [52]:
```
# if you have 100 categories you cannot copy & paste all the things
```

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

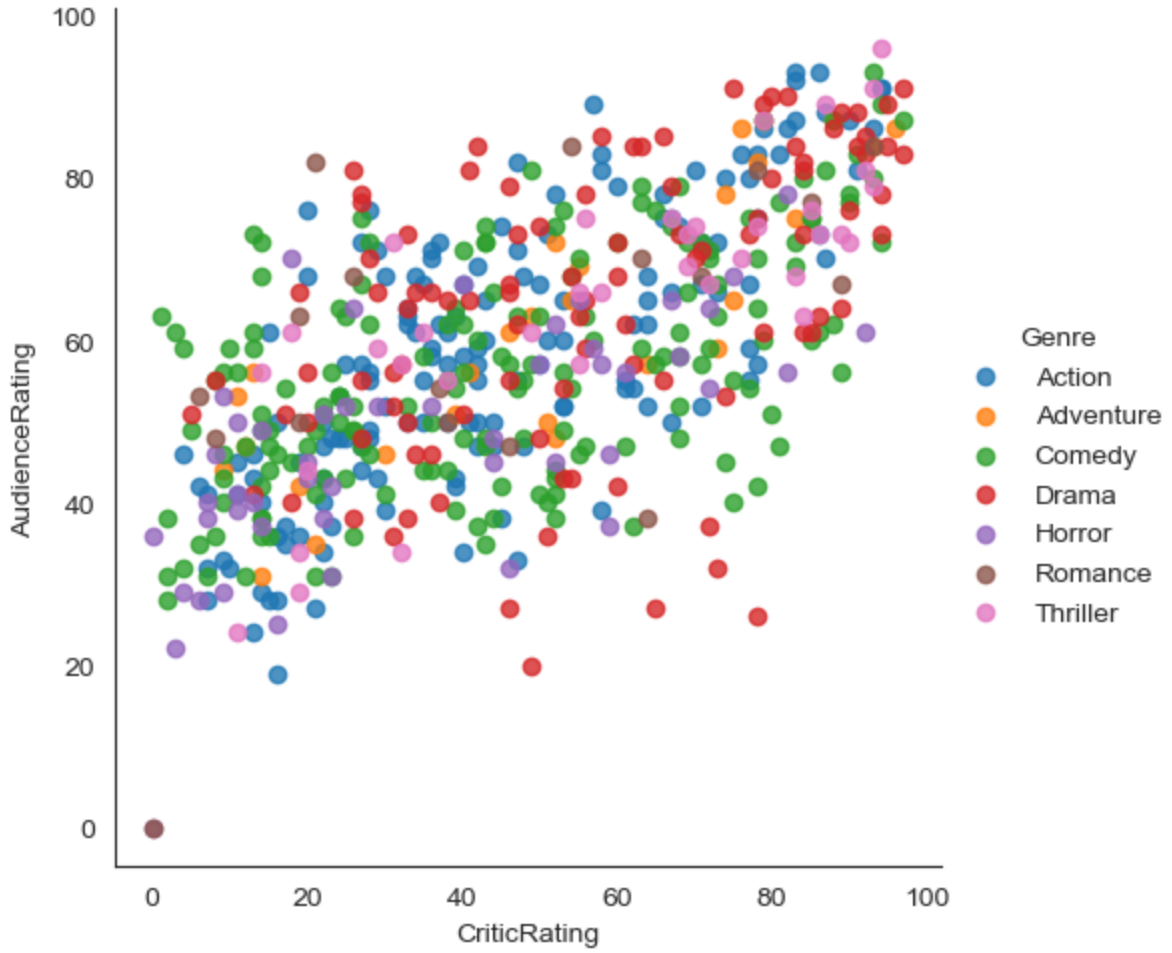
```
Action
Adventure
Comedy
Drama
Horror
Romance
Thriller
```

Scatter plot

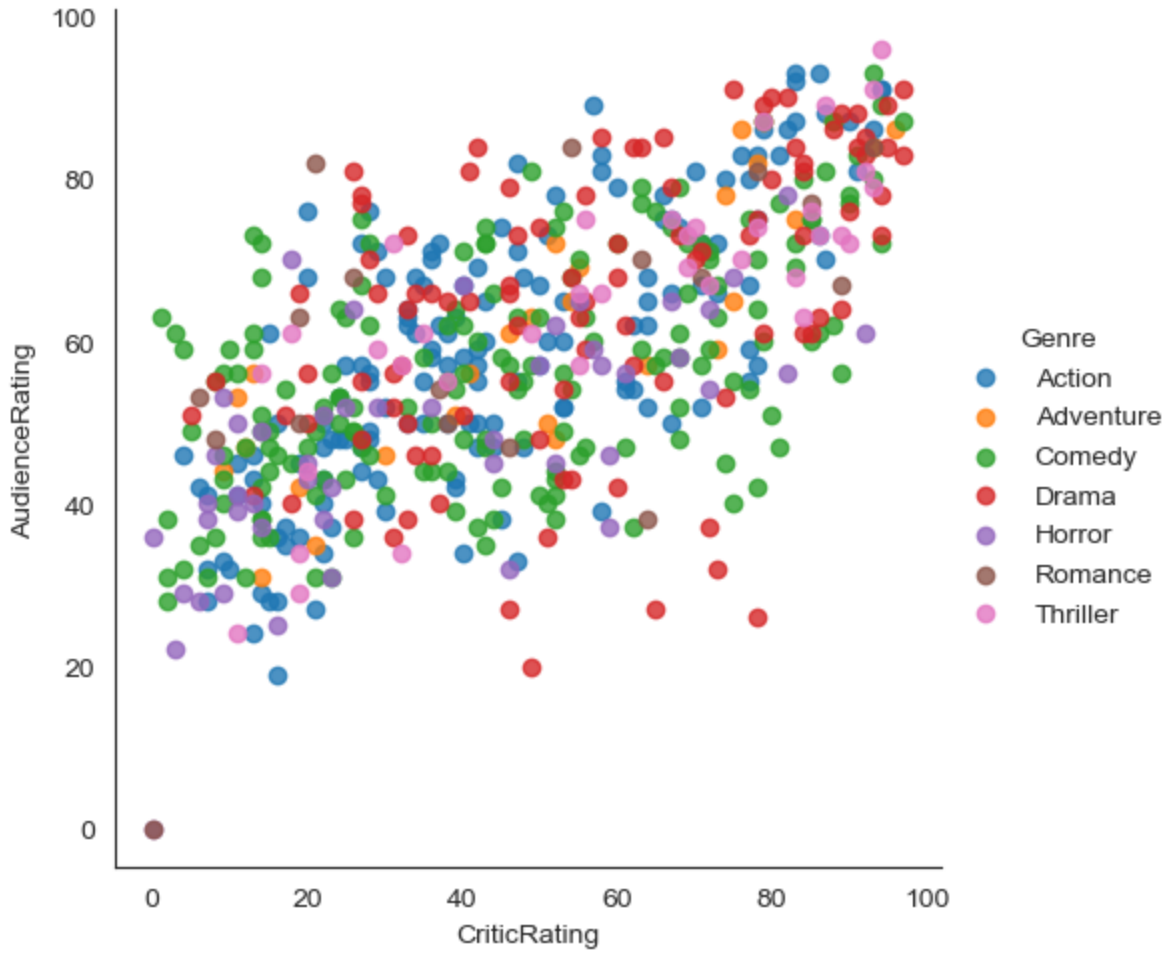
In [53]:
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False)
```



In [55]:
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False, hue = 'Genre')
```



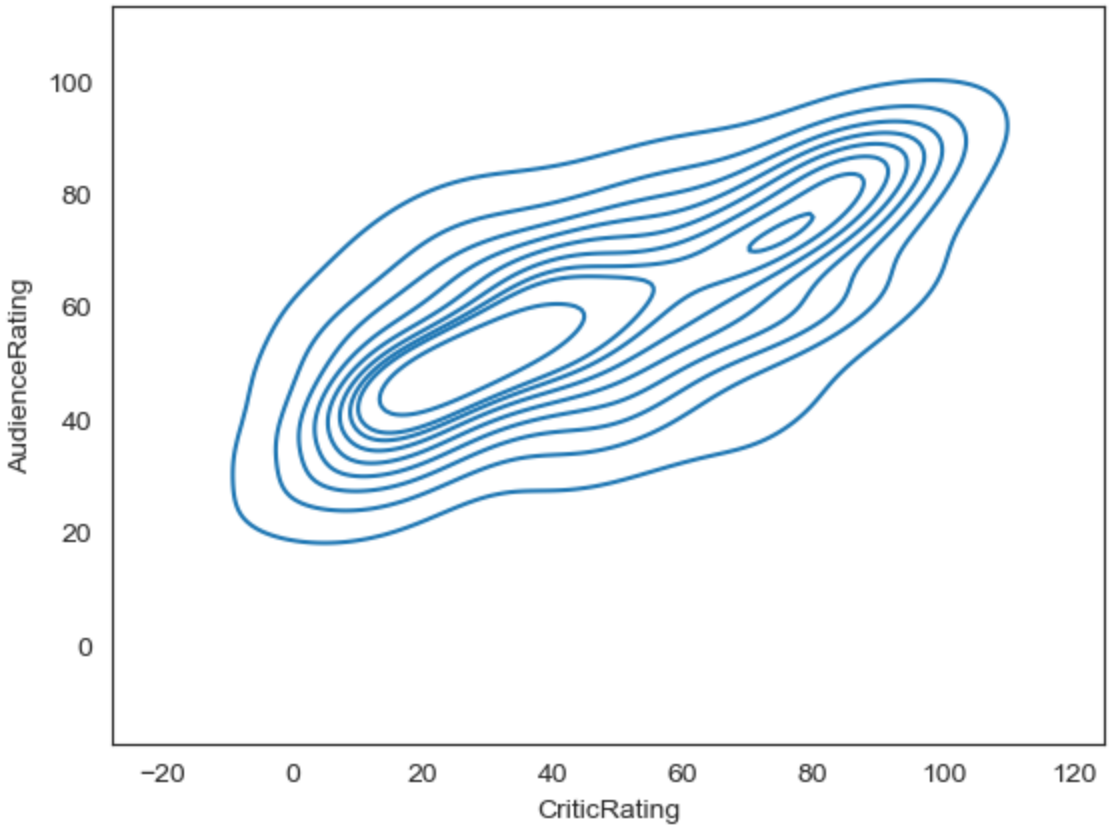
In [58]:
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False, hue = 'Genre', aspect=1)
```



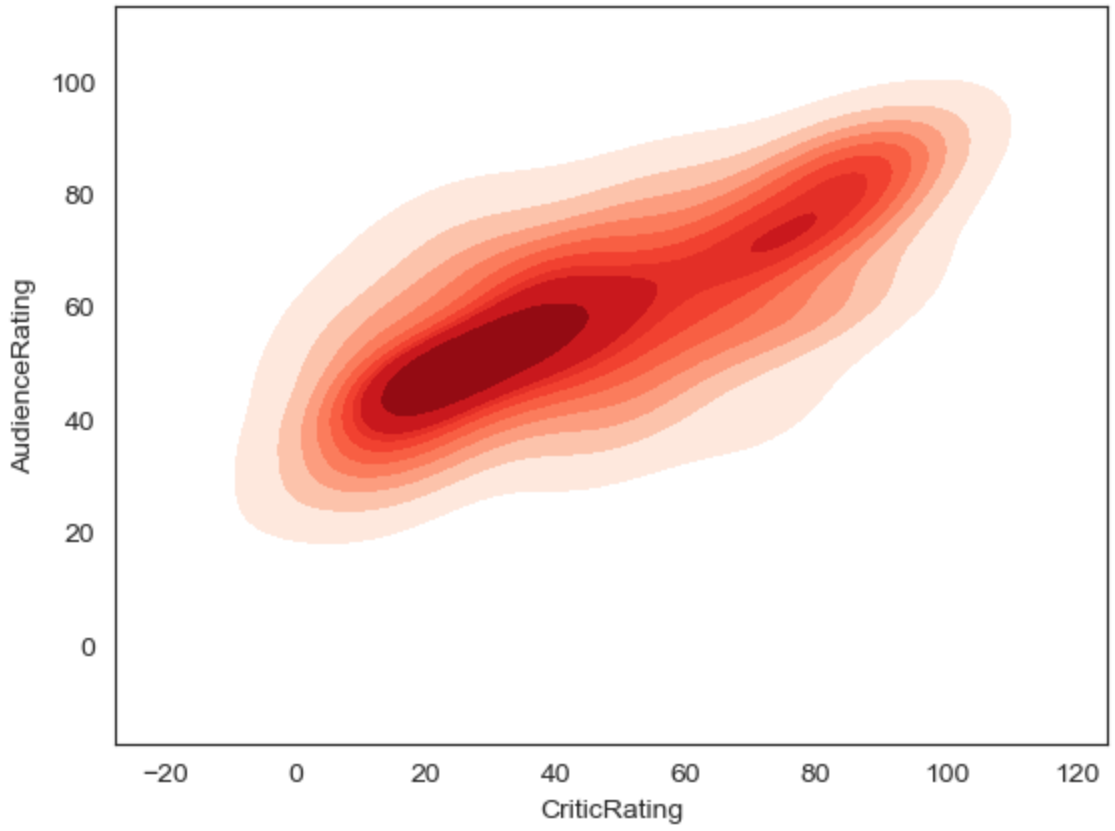
Kernal Density Estimate plot (KDE PLOT)

- How can we visulize audience rating & critics rating using scatterplot

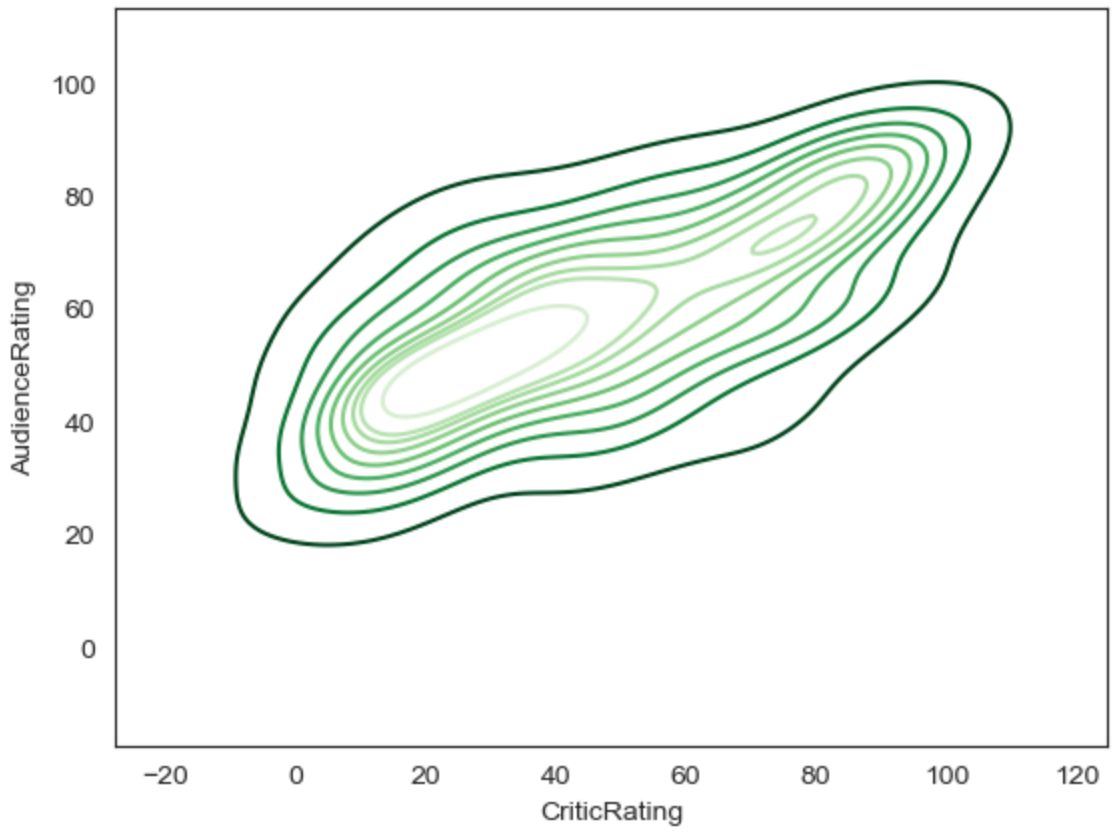
In [66]:
```
# where do u find more density and how density is distibuted across from the the chat
# center point is kernal this is calld KDE & insteade of dots it visualize like this
# we can able to clearly see the spread at the audience ratings

k1 = sns.kdeplot(x = movies['CriticRating'],y = movies['AudienceRating'])
plt.show()
```



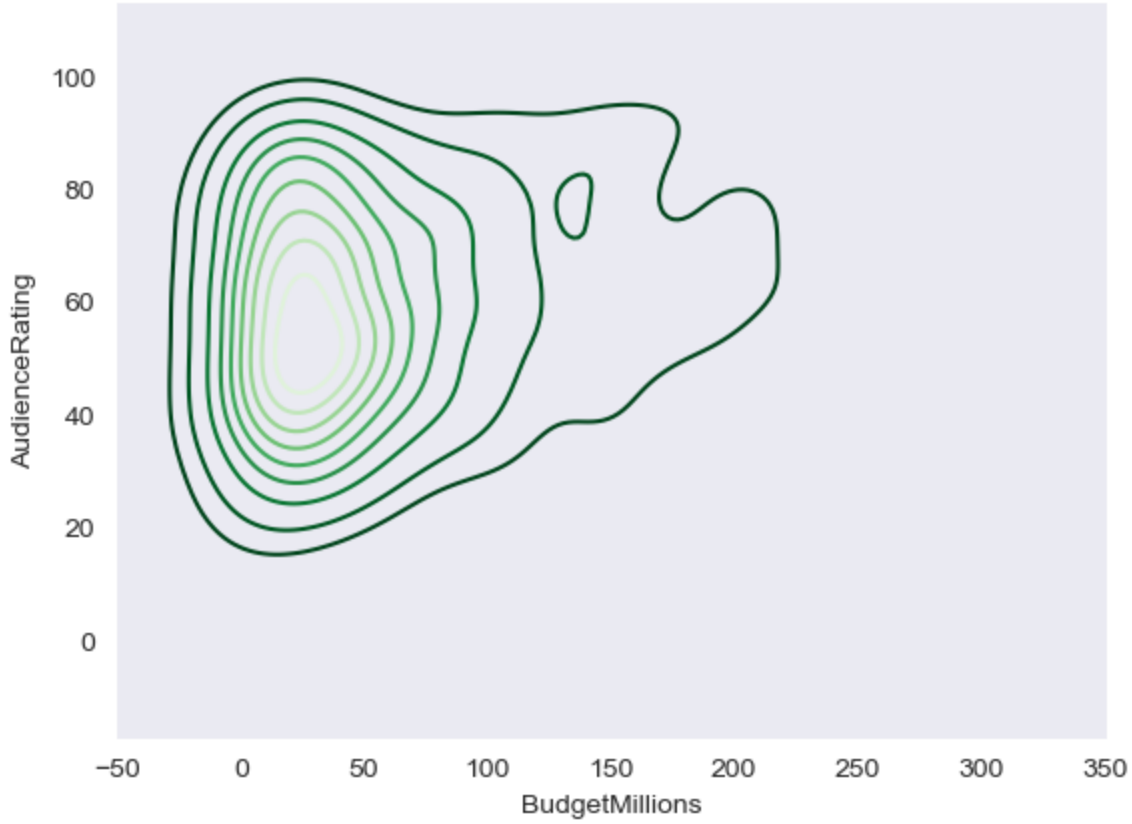
In [67]:
```
k1 = sns.kdeplot(x = movies.CriticRating,y = movies.AudienceRating,shade = True,shade_lo
plt.show()
```



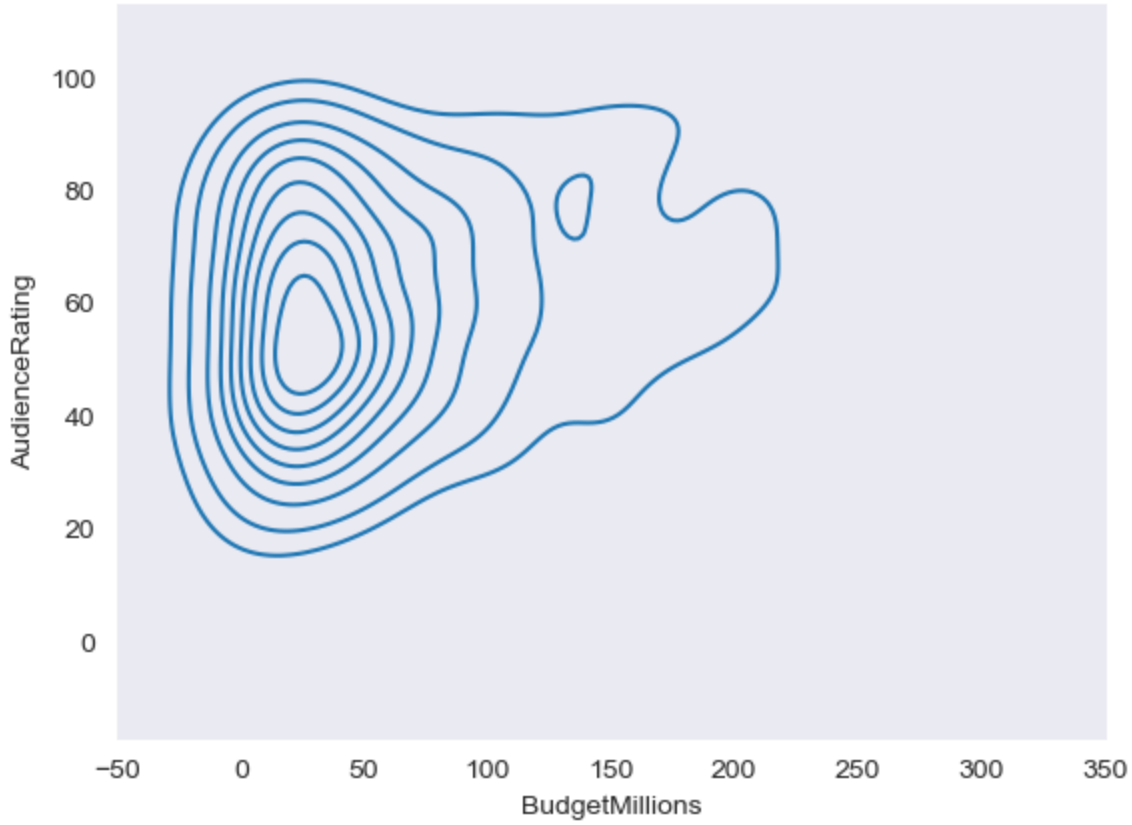
In [69]:
```
k2 = sns.kdeplot(x = movies.CriticRating, y = movies.AudienceRating, shade_lowest=False,
plt.show()
```



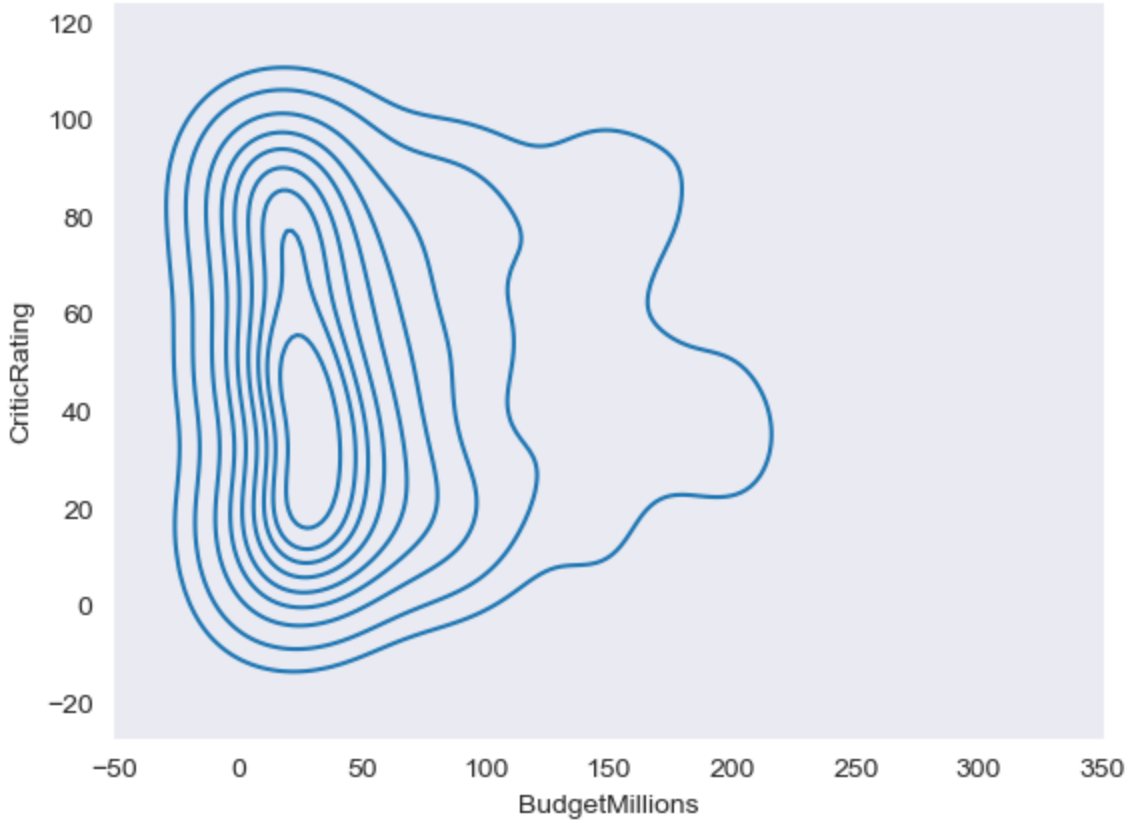
In [70]:
```
sns.set_style('dark')
k1 = sns.kdeplot(x = movies.BudgetMillions, y = movies.AudienceRating, shade_lowest=Fals
plt.show()
```



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

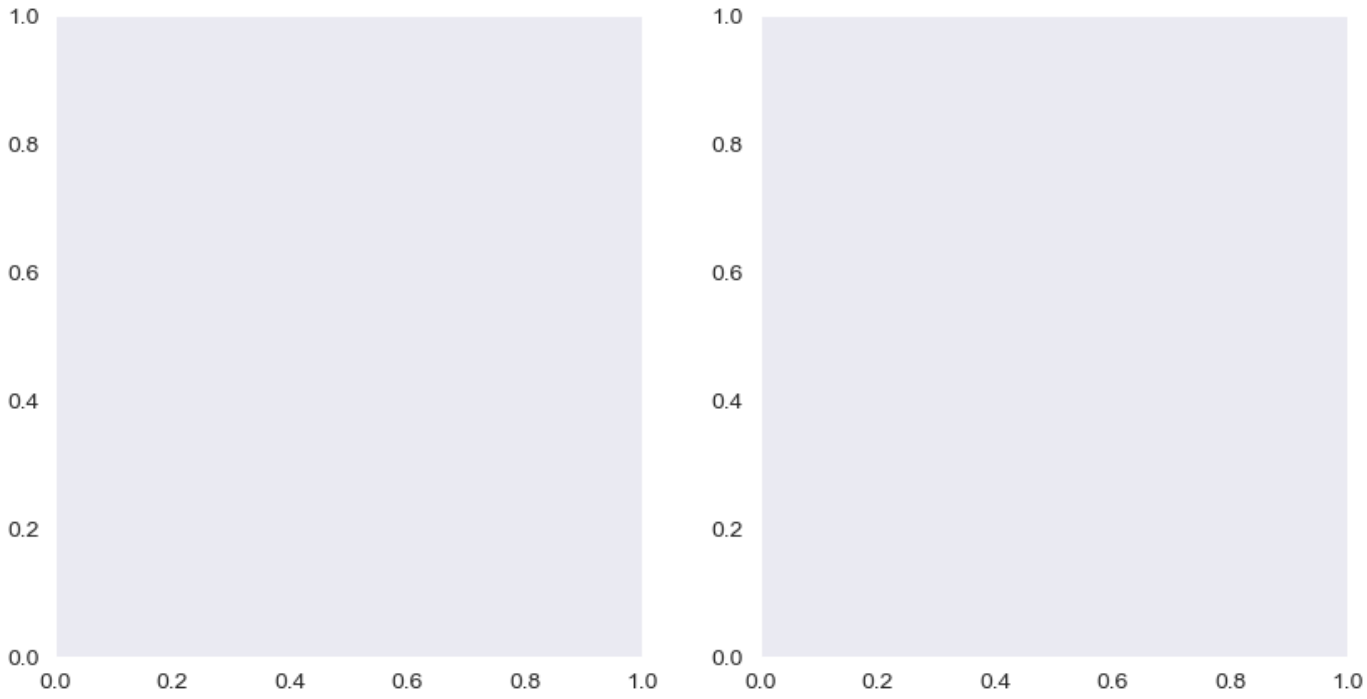


In [73]:
```
k2 = sns.kdeplot(x = movies.BudgetMillions, y = movies.CriticRating)
plt.show()
```

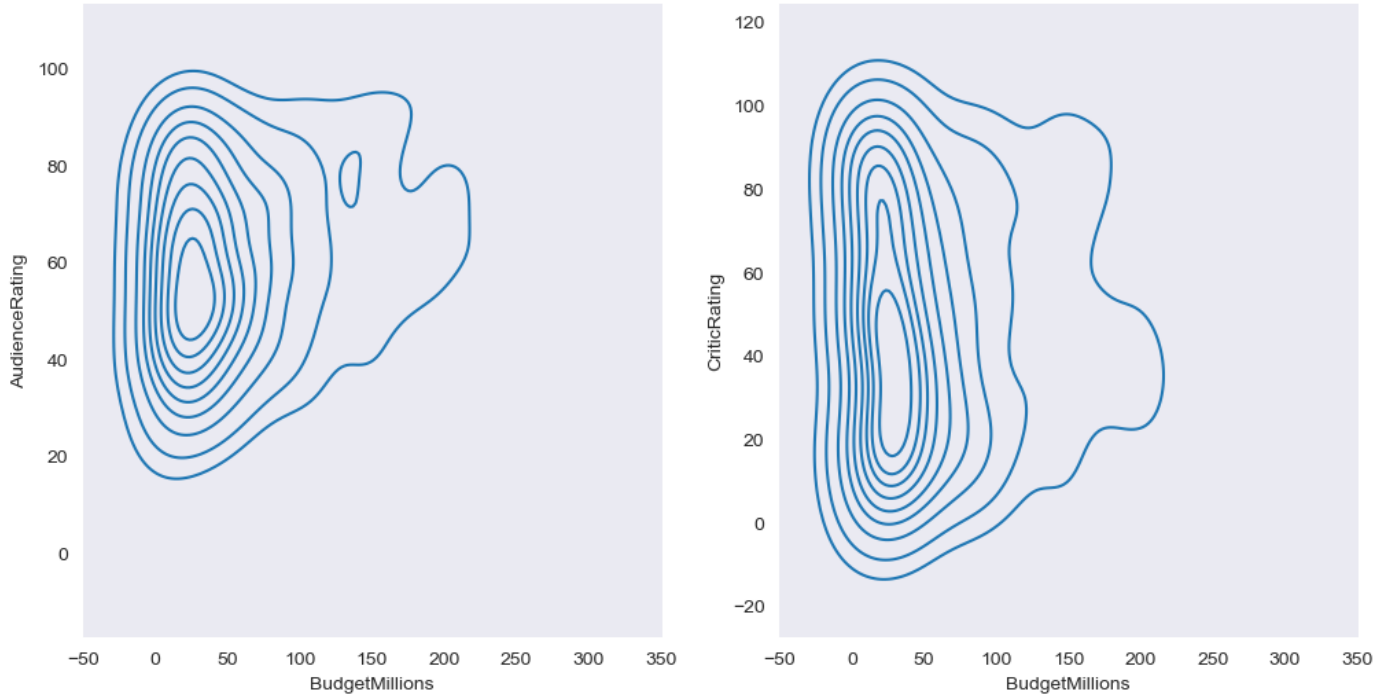


Ploting subplots

In [79]:
```
f, ax = plt.subplots(1,2, figsize = (10,5))
#f, ax = plt.subplots(3,3, figsize = (12,6))
```



In [81]:
```
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])
```

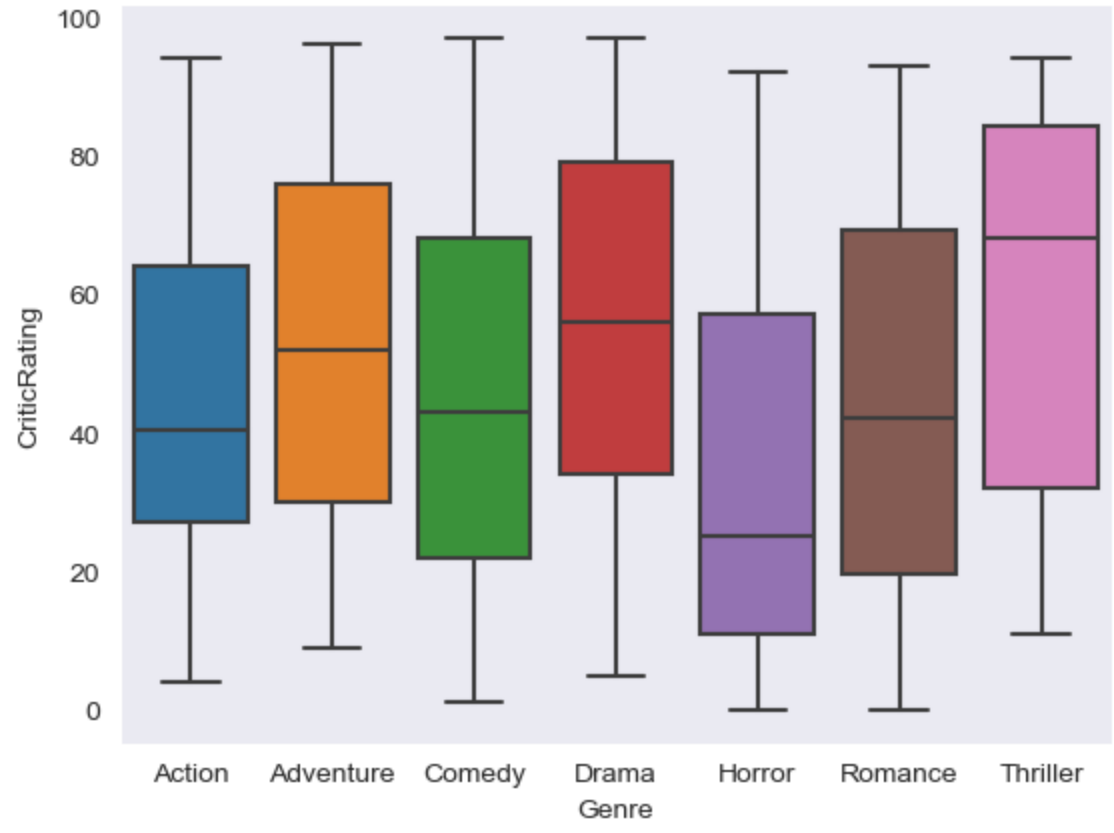


In [82]:
```
axes
```

Out[82]:
```
array([<Axes: xlabel='BudgetMillions', ylabel='AudienceRating'>,
       <Axes: xlabel='BudgetMillions', ylabel='CriticRating'>],
      dtype=object)
```

Box plot

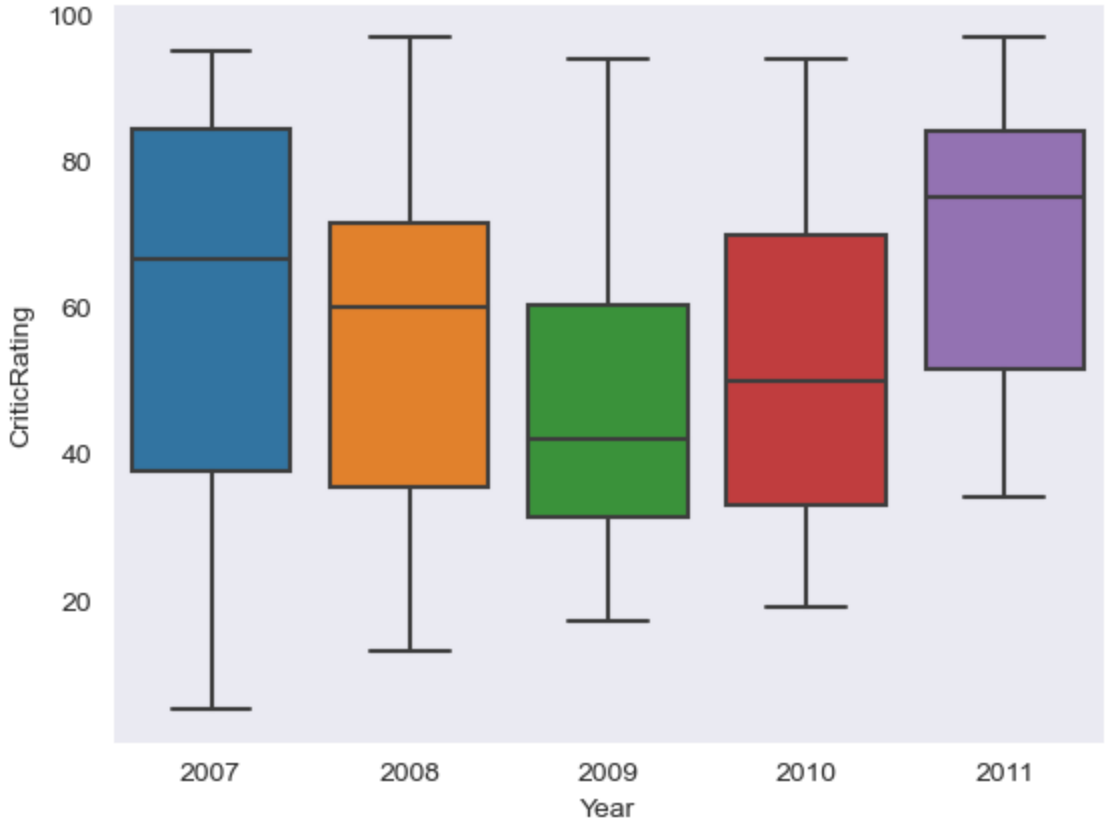
In [83]:
```
w = sns.boxplot(data=movies, x='Genre', y = 'CriticRating')
```



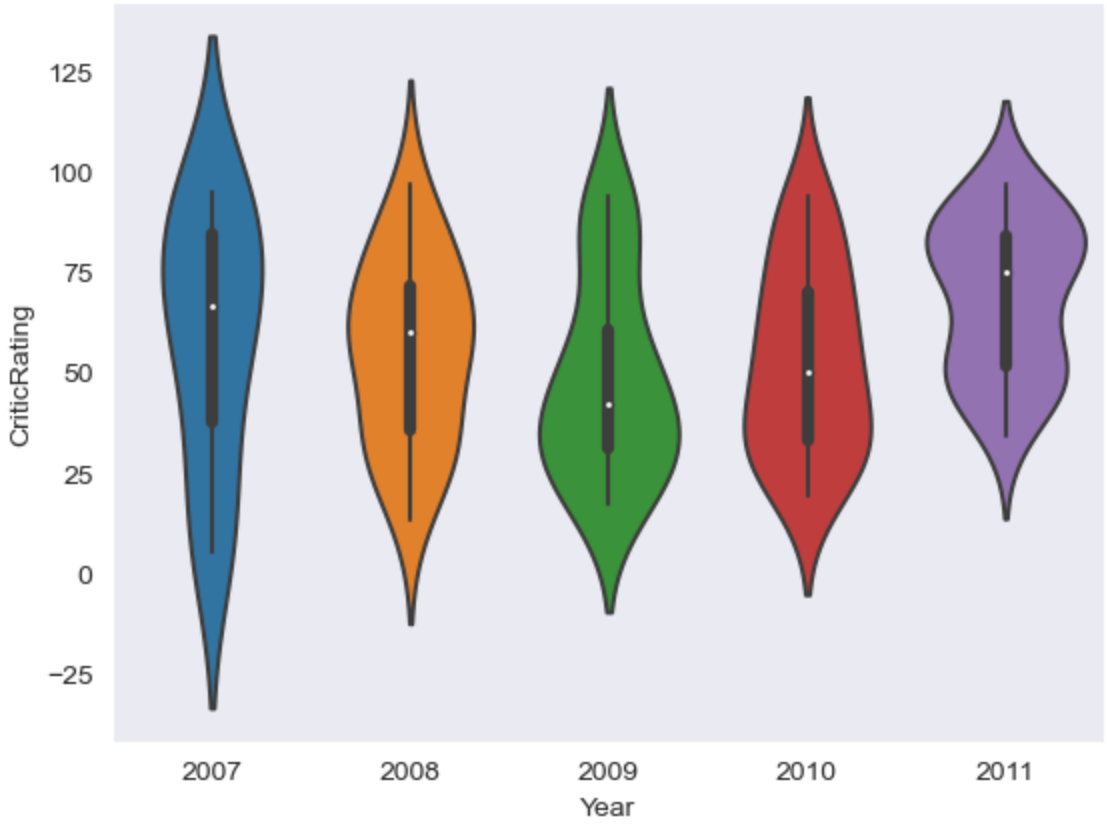
Violin plot

In [84]:
```
z = sns.violinplot(data=movies, x='Genre', y = 'CriticRating')
```

In [85]:
```
# Box plot movies genre drama
w1 = sns.boxplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRating')
```

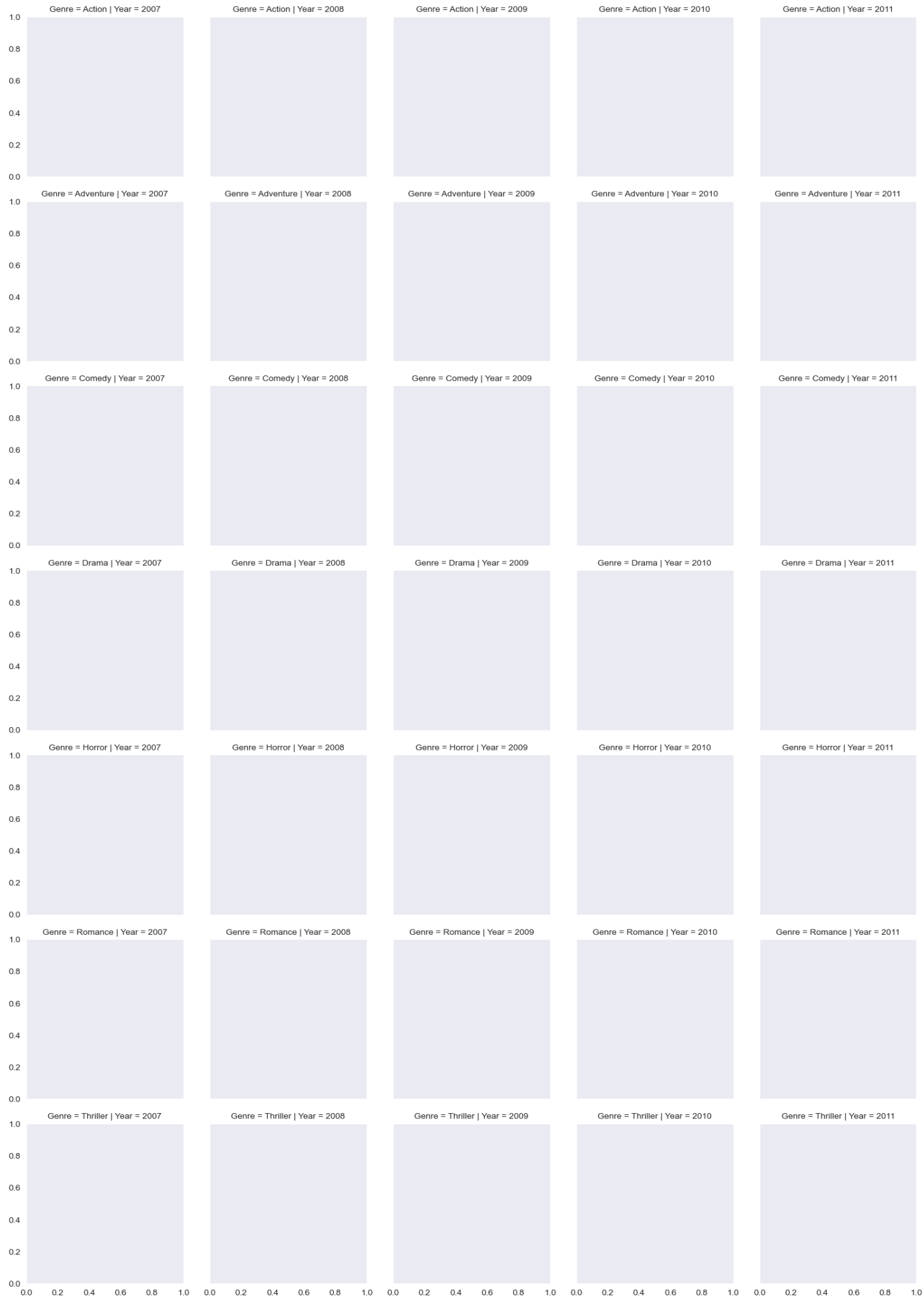


In [86]:
```
# Violn plot movies genre drama
z = sns.violinplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRating')
```

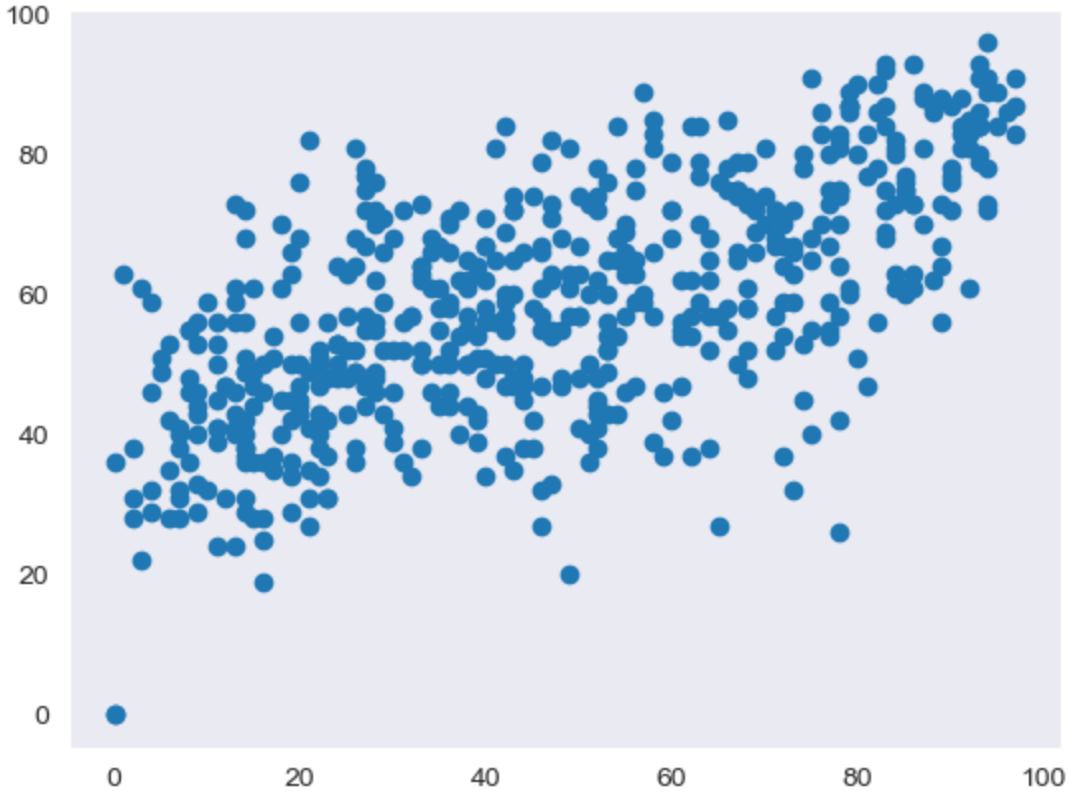


Creating a Facet grid

In [87]:
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre') # kind of subplots
```

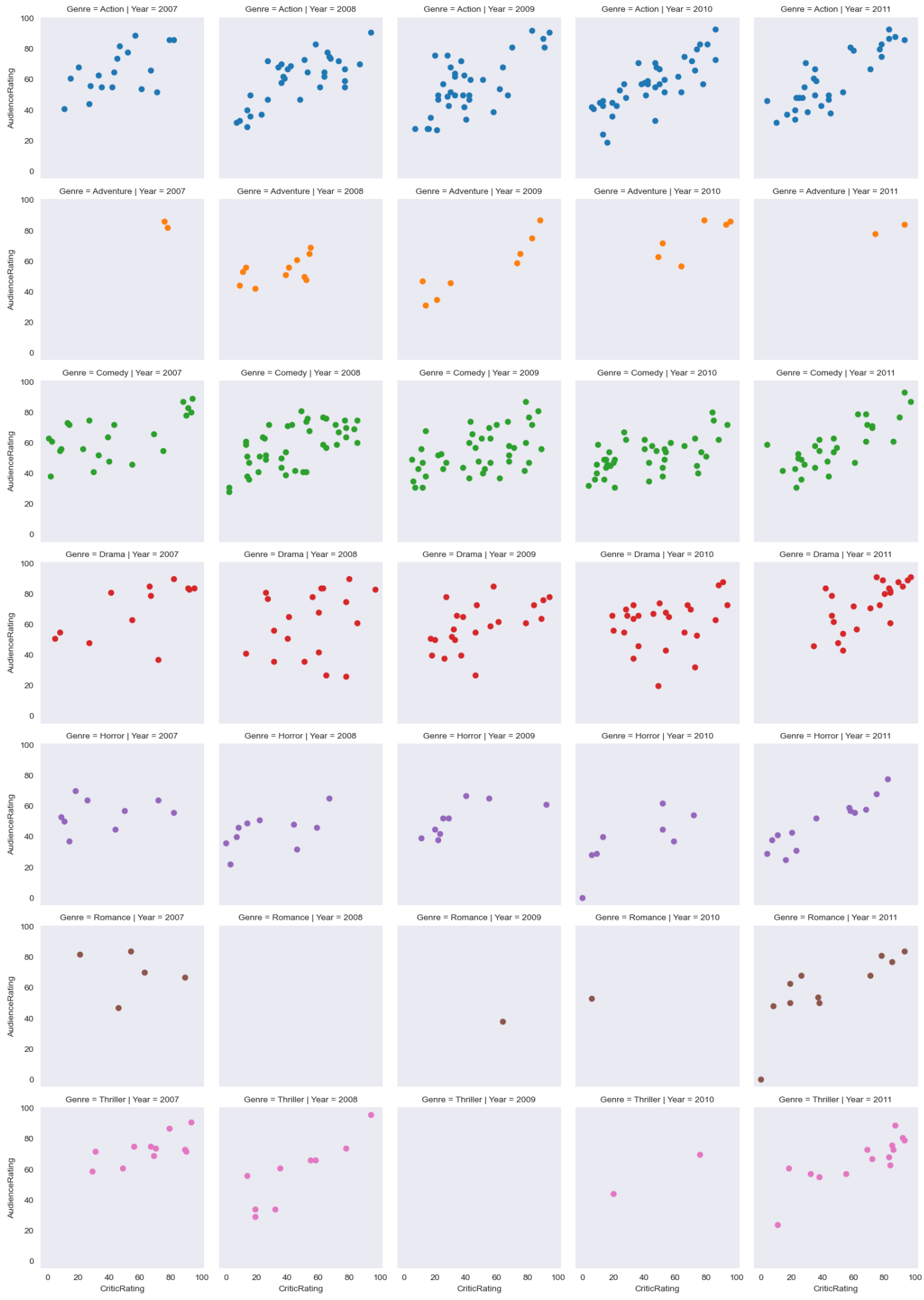


In [89]:
```
plt.scatter(x = movies.CriticRating,y = movies.AudienceRating)
plt.show()
```



Scatterplots are mapped in facetgrid

In [90]:
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating' )
```

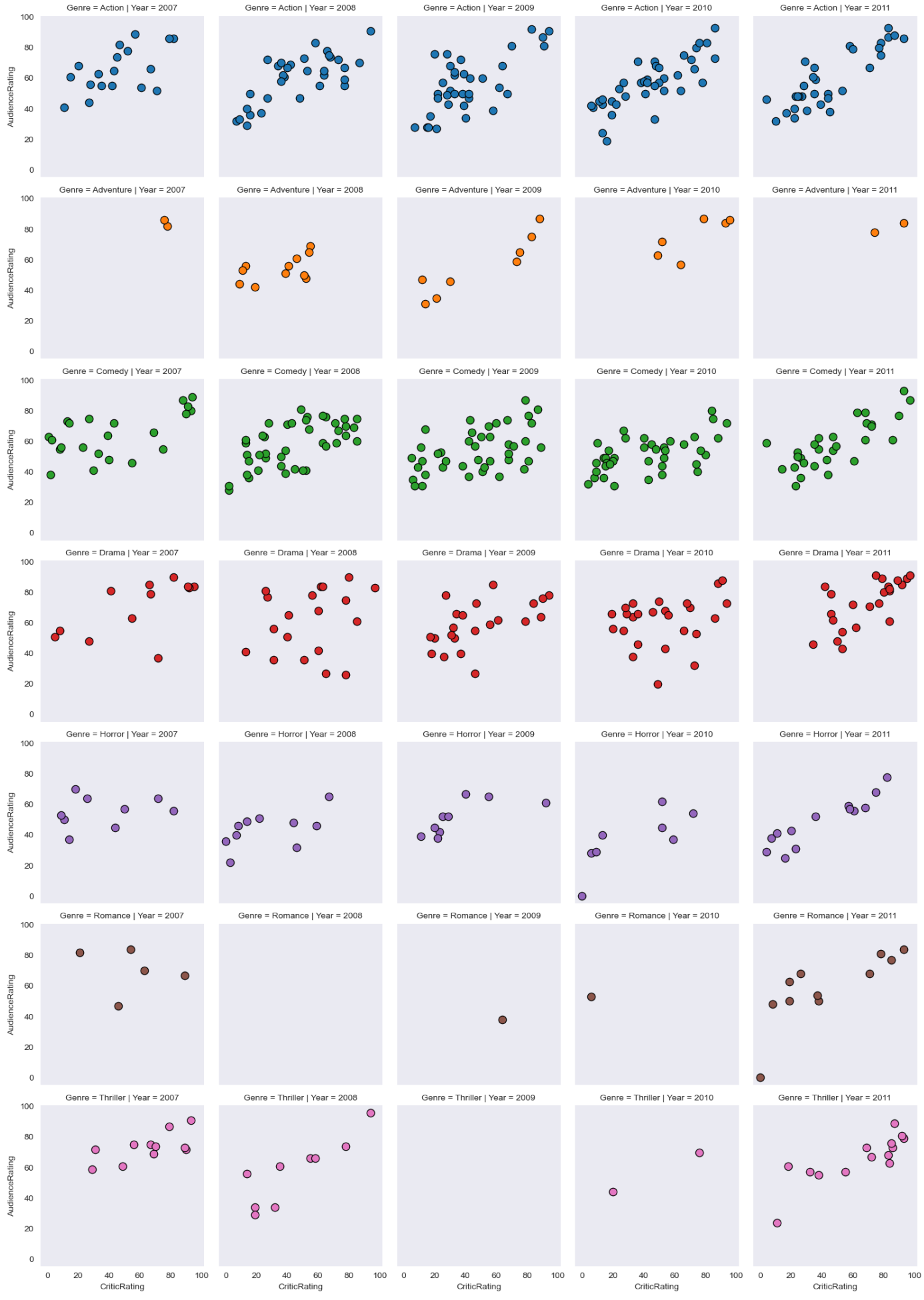


In [91]:
```
# You can populated any type of chat.
# Histogram plots are mapped in facetgrid
```

```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.hist, 'BudgetMillions')
```

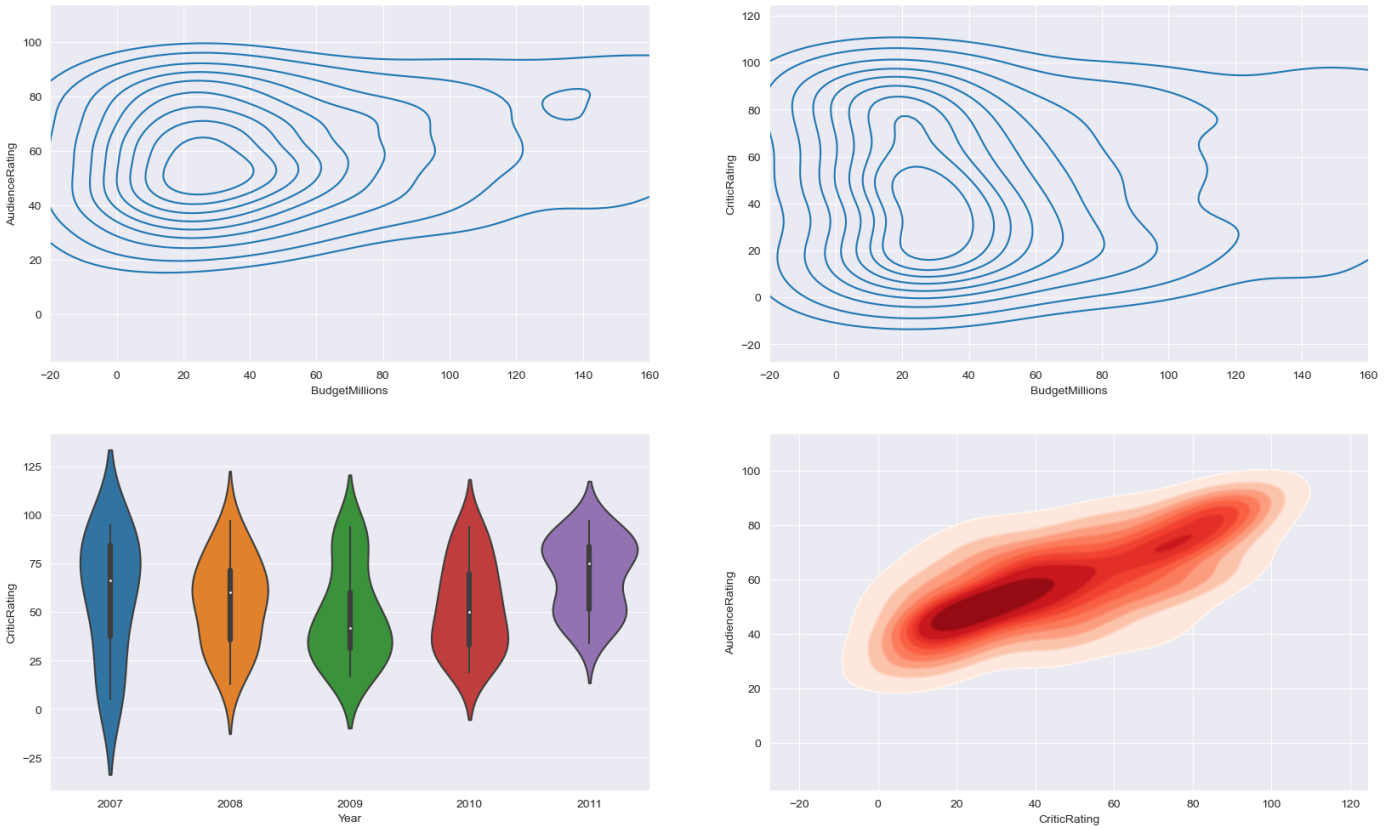
In [94]:
```
# Change size and linewidth
```

```
g = sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
kws = dict(s = 80, linewidth=0.9,edgecolor='black')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating',**kws )
```



In [105…

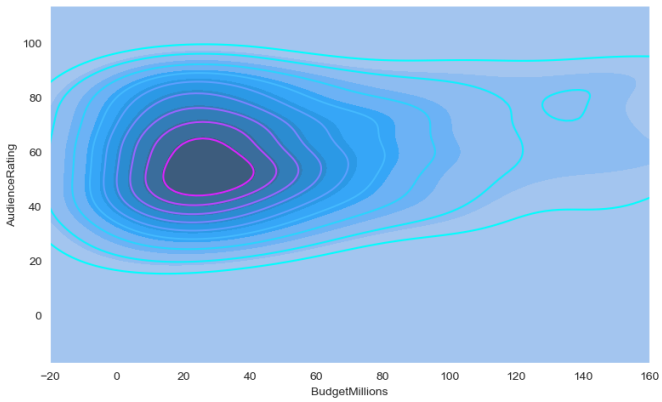
```
# python is not vectorize programming language
# Building dashboards (dashboard - combination of chats)

sns.set_style('darkgrid')
f, axes = plt.subplots (2,2, figsize = (20,12))

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(data=movies[movies.Genre=='Drama'], x='Year', y = 'CriticRating', ax=

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

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

plt.show()
```

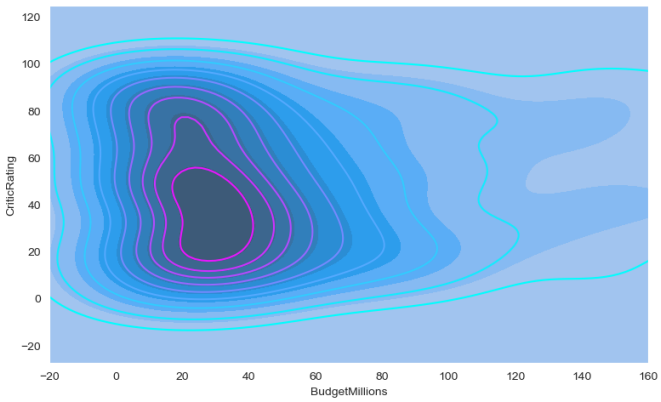


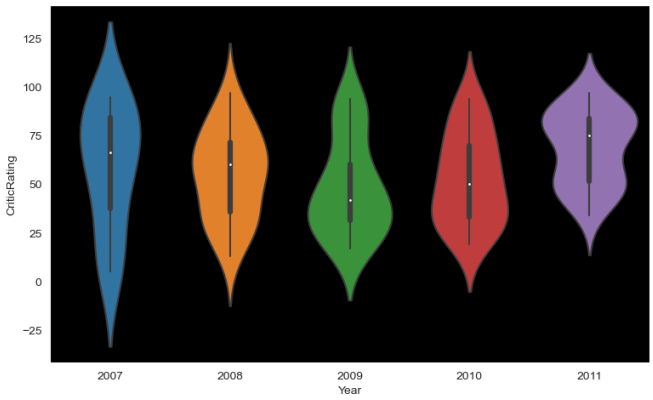
In [104…

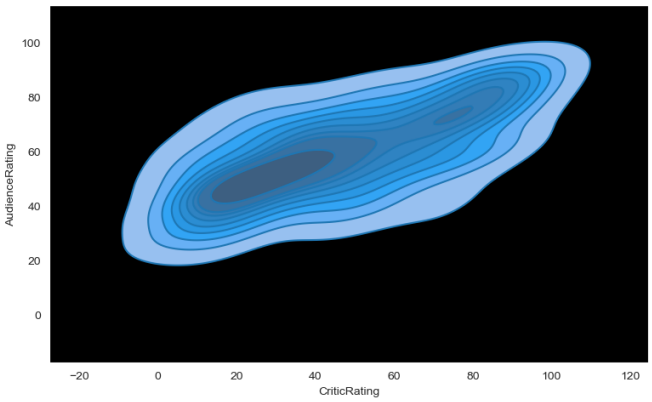
```
# How can you style your dashboard  using different color map

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

sns.set_style('dark',{'axes.facecolor':'black'})
f, axes = plt.subplots (2,2, figsize = (20,12))

#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[0,0])

#plot [0,1]
k2 = sns.kdeplot(x = movies.BudgetMillions, y = movies.CriticRating,\
                 shade=True, shade_lowest=True,\
```

```
                    ax = axes[0,1])
k2b = sns.kdeplot(x = movies.BudgetMillions, y = movies.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(x = movies.CriticRating,y = movies.AudienceRating, \
                    shade = True,shade_lowest=False, \
                    ax=axes[1,1])

k4b = sns.kdeplot(x = movies.CriticRating, y = movies.AudienceRating, \
                     ax = axes[1,1])


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

plt.show()
```









Final discussion what we learn so far -

- Category datatype in python
- Jointplots
- Histogram
- Stacked histograms
- Kde plot
- Subplot
- Violin plots
- Factet grid
- Building dashboards

EDA is explan by Mr. Prakash Senapati.