In [1]:

```
# (MOVIE RATING ANALYSIS)(ADVANCED VISUALIZATION)

import pandas as pd
import os
```

In [2]:

```
os.getcwd()  # if you want to change the working directory
```

Out[2]:

```
'C:\\Users\\Hanshu\\basics'
```

In [3]:

```
movies = pd.read_csv(r"C:\Users\Hanshu\Desktop\Movie-Rating (2).csv")
```

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.head()
```

Out[6]:

	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 [7]:
```
movies.tail()
```

Out[7]:

	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 [8]:
```
movies.columns
```

Out[8]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

In [9]:
```
movies.columns = ['Film', 'Genre', 'CriticRating', 'AudienceRating','BudgetMilli
movies.columns
```

Out[9]:
```
Index(['Film', 'Genre', 'CriticRating', 'AudienceRating', 'BudgetMillions',
       'Year'],
      dtype='object')
```

In [10]:
```
movies.head()       # removed spaces $ % removed noise characters
```

Out[10]:

	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 [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 val
# we have to change to categroy type
# also from object datatype we will convert to category datatypes
#
```

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

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]:

```
movies.head()
```

Out[16]:

	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 [17]:

```
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 [18]:

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

Out[18]:

```
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 [19]:

```
movies.info()     # here we changed 2 columns object
```

```
<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    int64   
dtypes: category(2), int64(4)
memory usage: 40.1 KB
```

In [20]:

```
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    int64   
dtypes: category(2), int64(4)
memory usage: 40.1 KB
```

In [21]:

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

Out[21]:

```
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 [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    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 [23]:
```
movies
```

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
...	...	...	...	...	...	...
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 [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', 'R
omance', 'Thriller']
```

In [25]:
```
movies.Year   # is it real no. year you can take average,min,max but out come ha
```

```
Out[25]: 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 [26]: movies.Film
```

```
Out[26]: 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 [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.Genre.cat.categories    #categories for unique values here rows from part
```

```
Out[28]: Index(['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance',
                'Thriller'],
               dtype='object')
```

```
In [29]: movies.describe()  #now when you see the describt you will get onl integer value
```

Out[29]:

	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 [30]:

```
# how to working with joint plots

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

JOINT PLOT

In [31]:

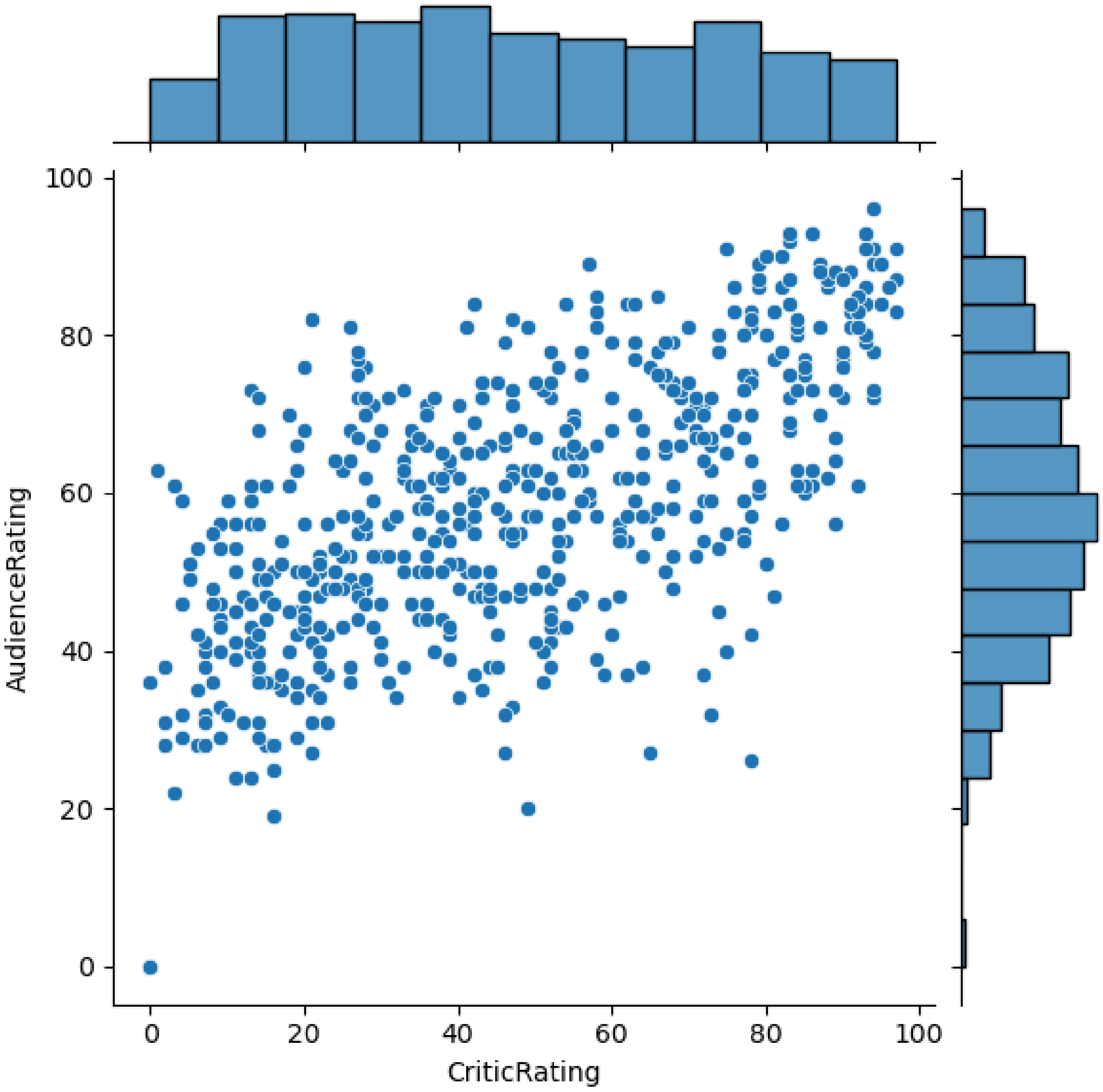
```
#* basically joint plot is a scatter plot & it find the relation b/w audiene & c
#* also if you look up you can find the uniform distribution
#(critics)and normal distriution (audience)
```

In [32]:

```
j = sns.jointplot(data = movies , x='CriticRating' ,y = 'AudienceRating')

# Audience rating is more dominant then critics rating
# Based on this we find out as most people are most liklihood to watch audience
# let me explain the excel - if you filter audience rating & critic rating. crit
```

In [33]:

```
j
plt.show()
```

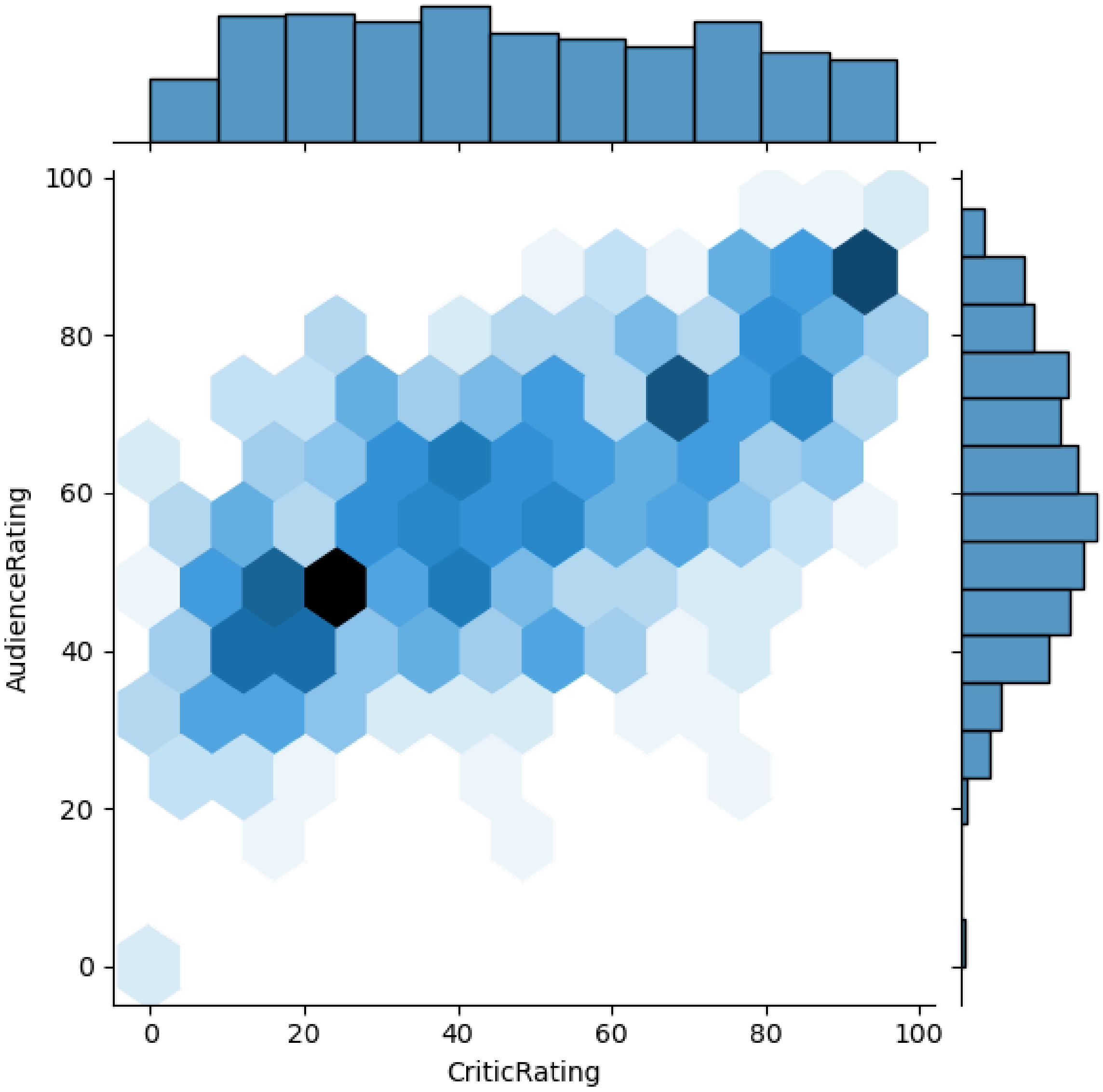


```
In [34]: j = sns.jointplot(data = movies, x = 'CriticRating', y = 'AudienceRating', kind
```

```
In [35]: j
```

```
Out[35]: <seaborn.axisgrid.JointGrid at 0x218e9c3b1a0>
```

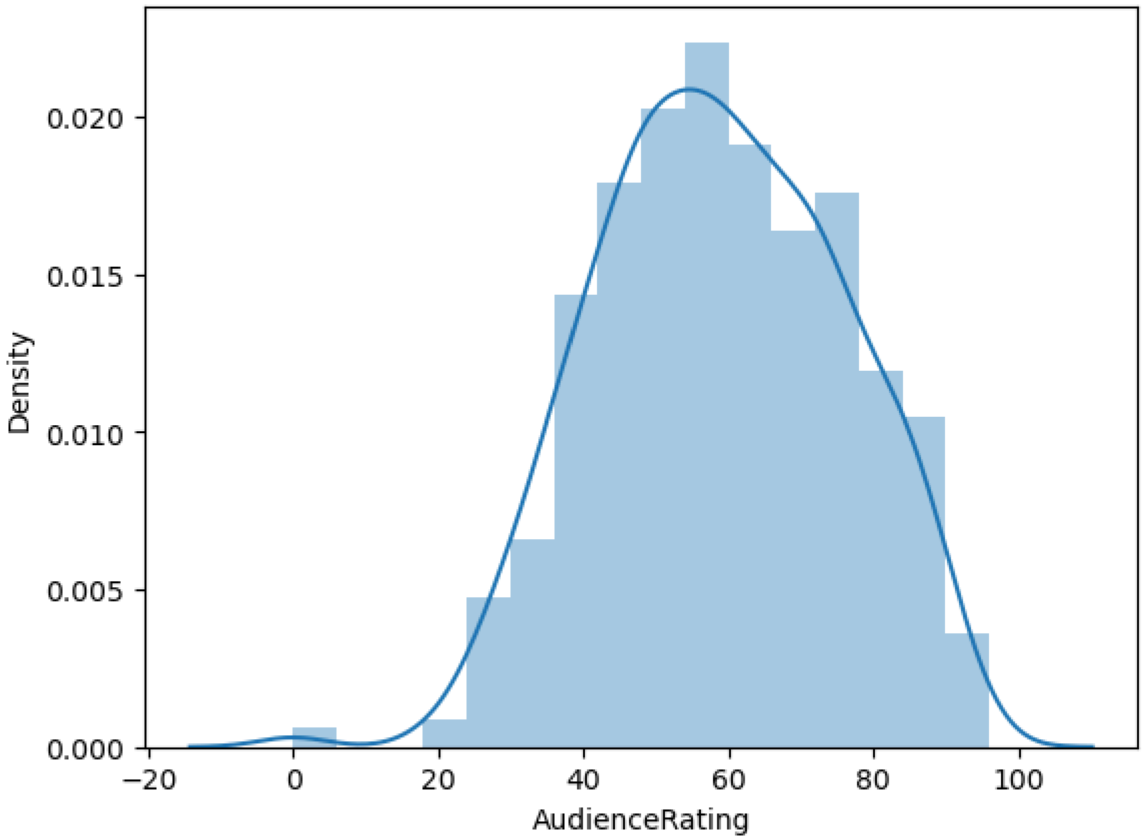
```
In [36]: plt.show()
```



HISTOGRAM

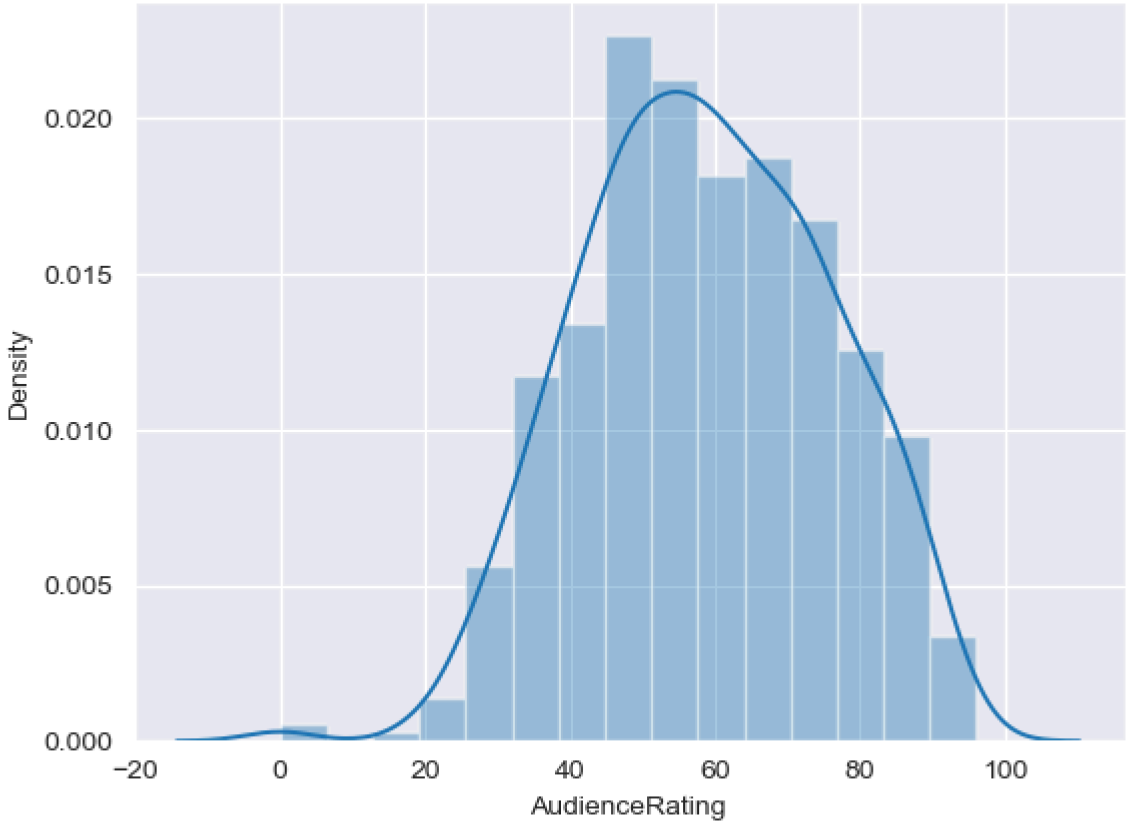
```
In [37]: m1 = sns.distplot(movies.AudienceRating)   #y - axis generated by seaborn automa
```

```
In [38]: m1
         plt.show()
```



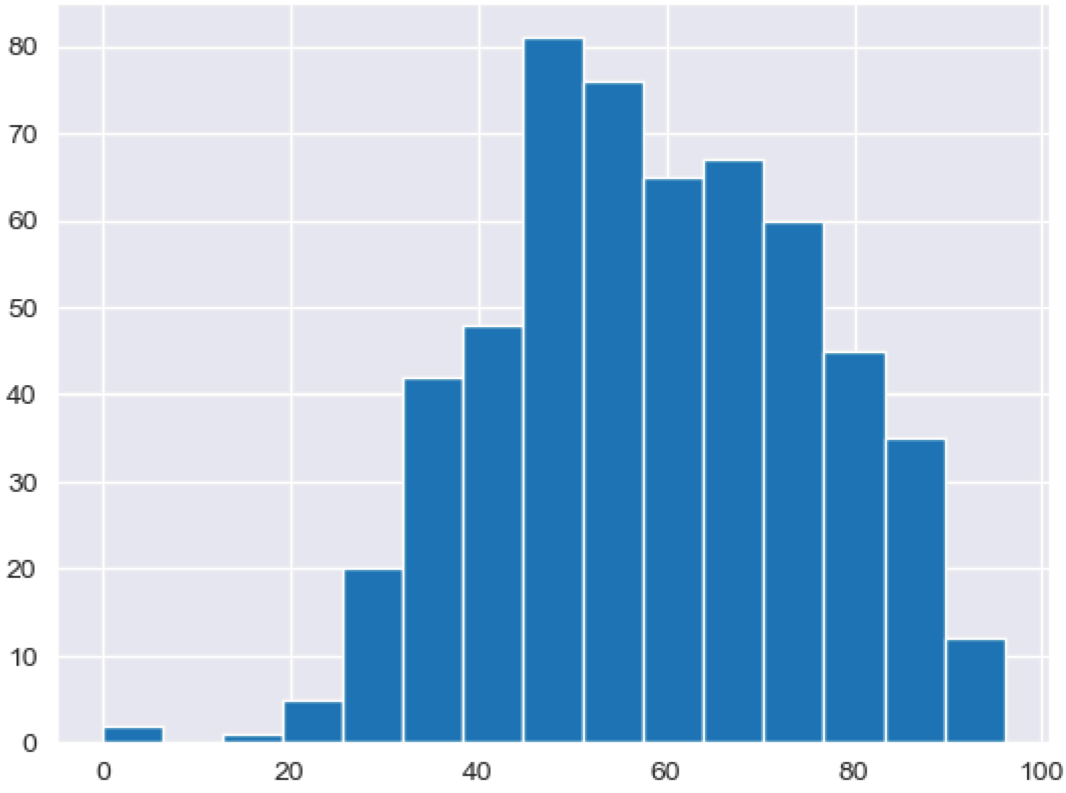
In [39]:
```
sns.set_style('darkgrid')
```

In [40]:
```
m2 = sns.distplot(movies.AudienceRating , bins = 15)
m2
plt.show()
```



In [41]:

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

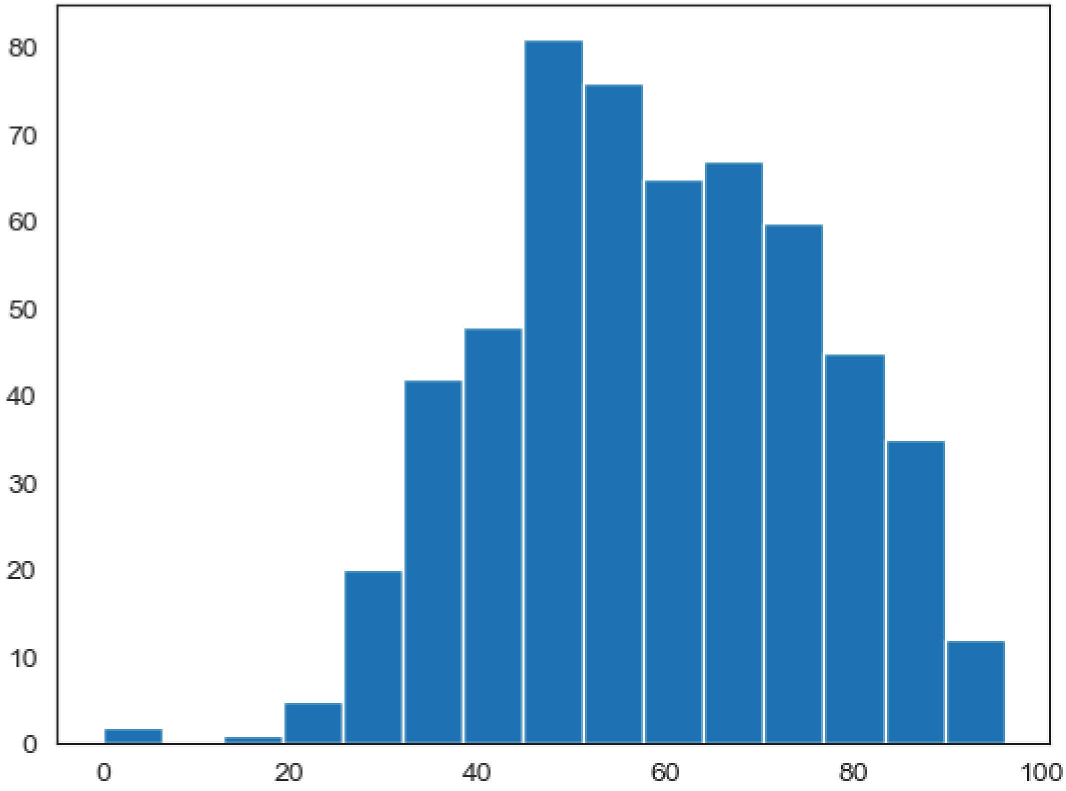


In [42]:

```
sns.set_style('white')     # normal distribution & called as bell curve
n1 = plt.hist(movies.AudienceRating, bins = 15)
```

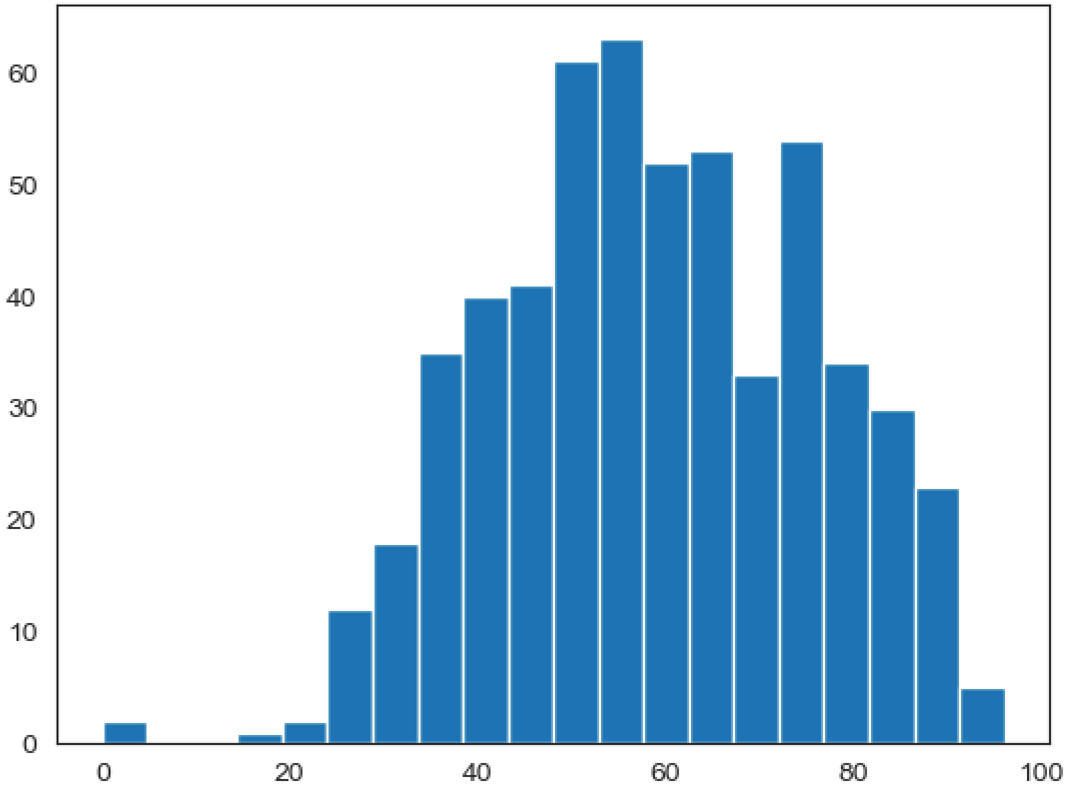
In [43]:

```
plt.show()
```



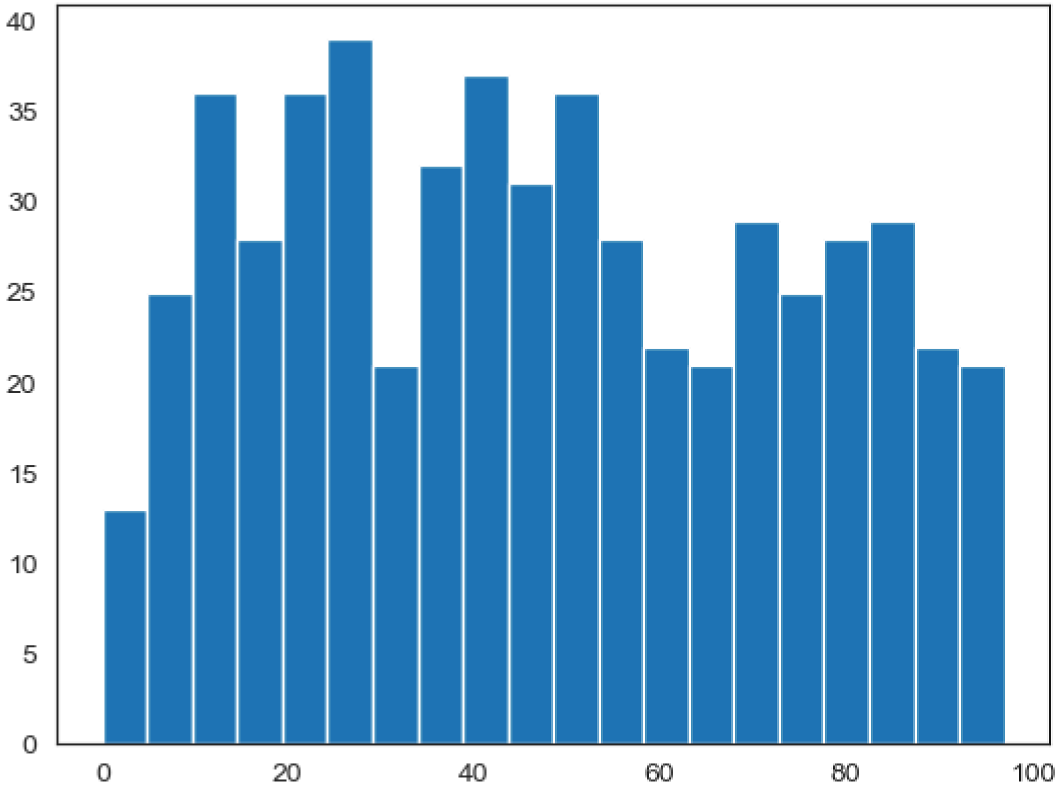
In [44]:
```
sns.set_style('white')      #normal distrubution & called as well curve
n1 = plt.hist(movies.AudienceRating, bins = 20)
```

In [45]:
```
plt.show()
```



In [46]:
```
n1 = plt.hist(movies.CriticRating , bins = 20)    # Uniform distribution
```

In [47]:
```
plt.show()
```

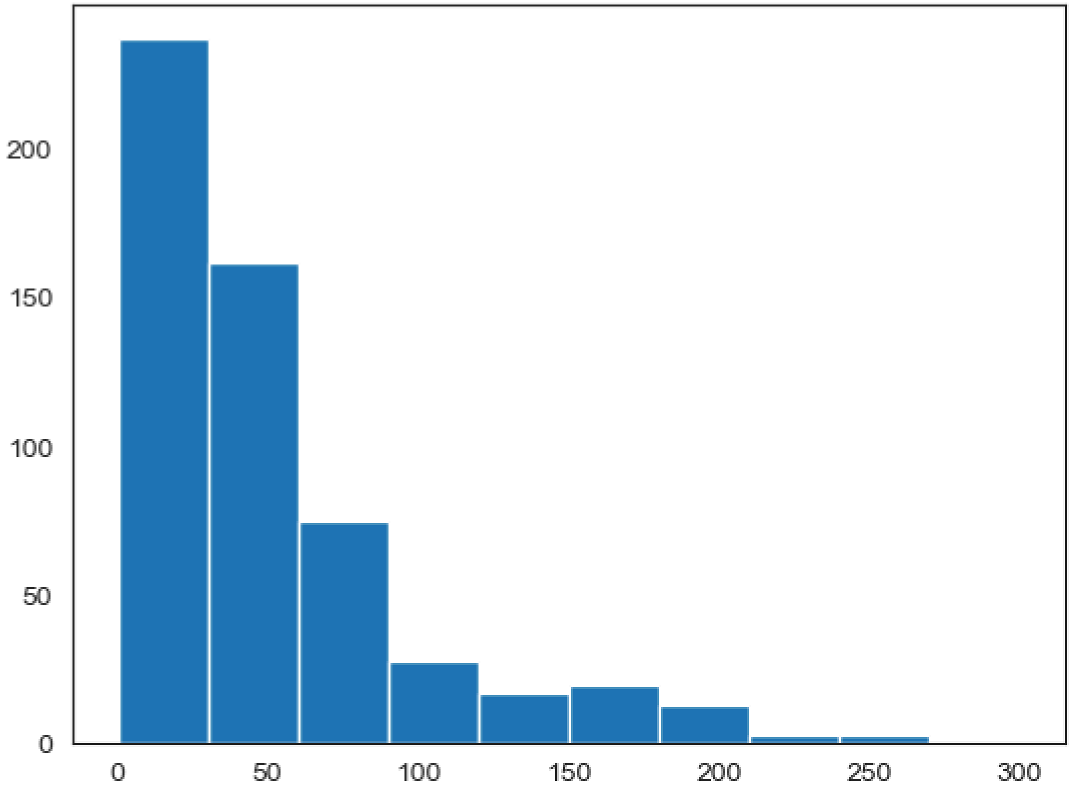


In [48]:

```
# Creating stacked histograms & this is bit tough to understand
```

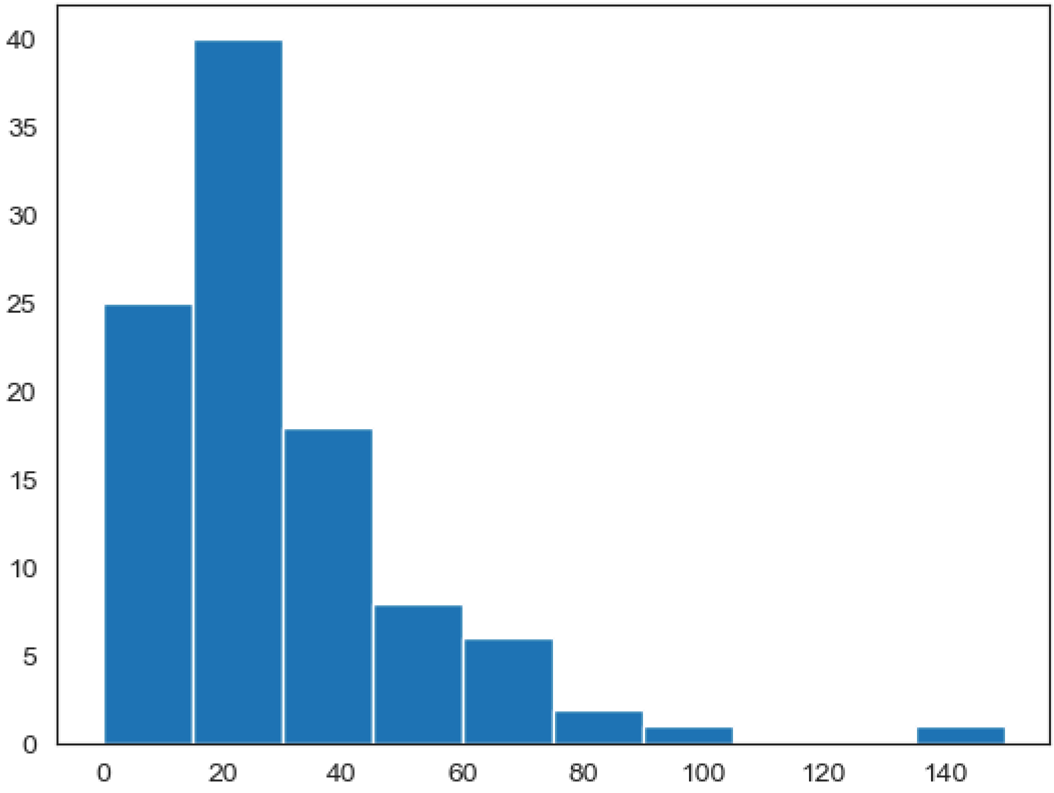
In [49]:

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



In [50]:

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



In [51]:

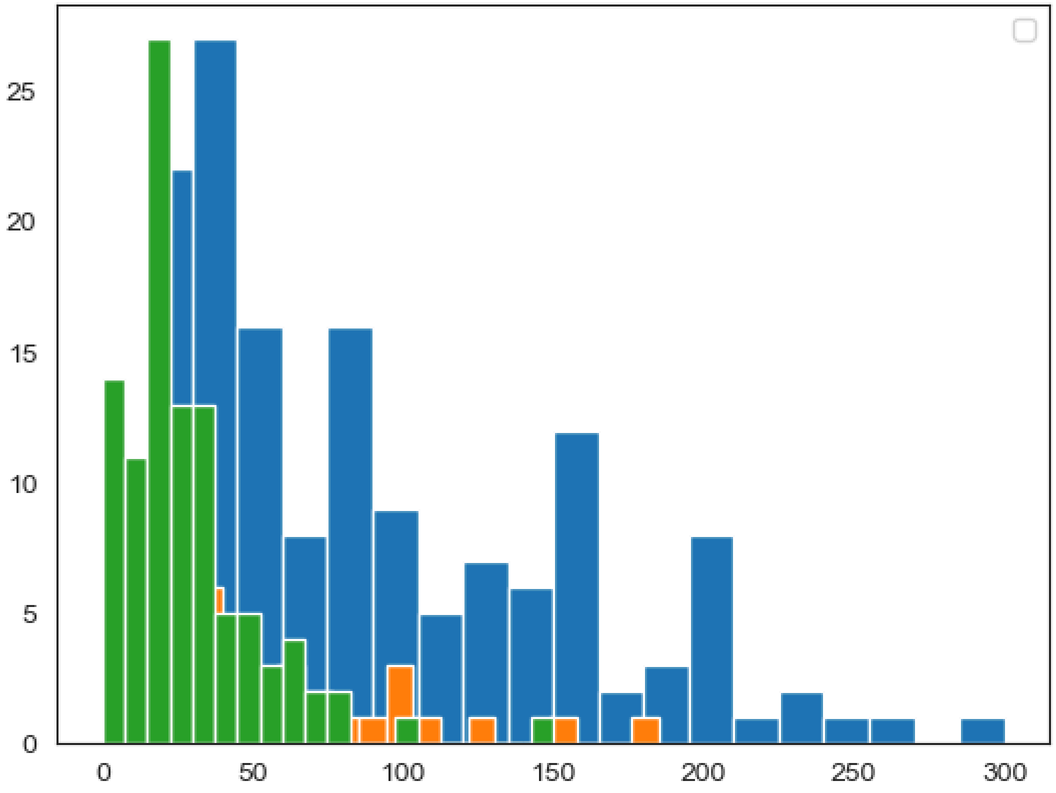
```
movies.head()
```

Out[51]:

	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

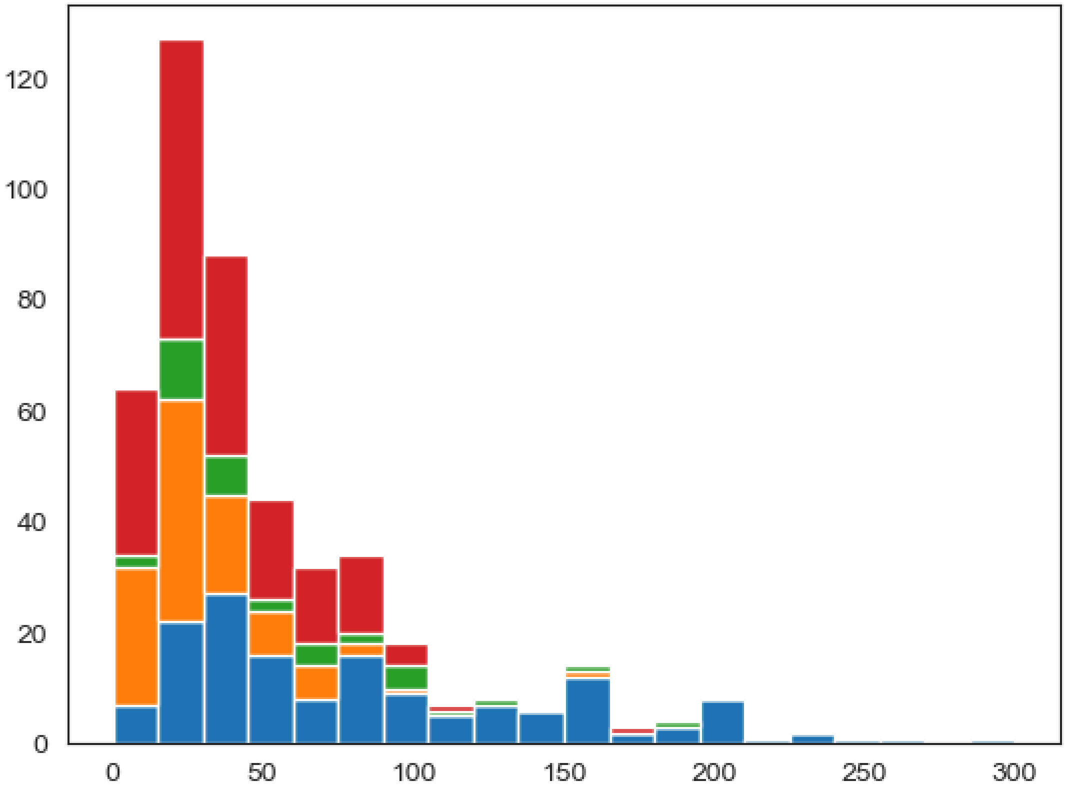
In [52]:

```
# Below plots are stacked histogram becuase overlaped

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()
```



In [53]:

```
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()
```



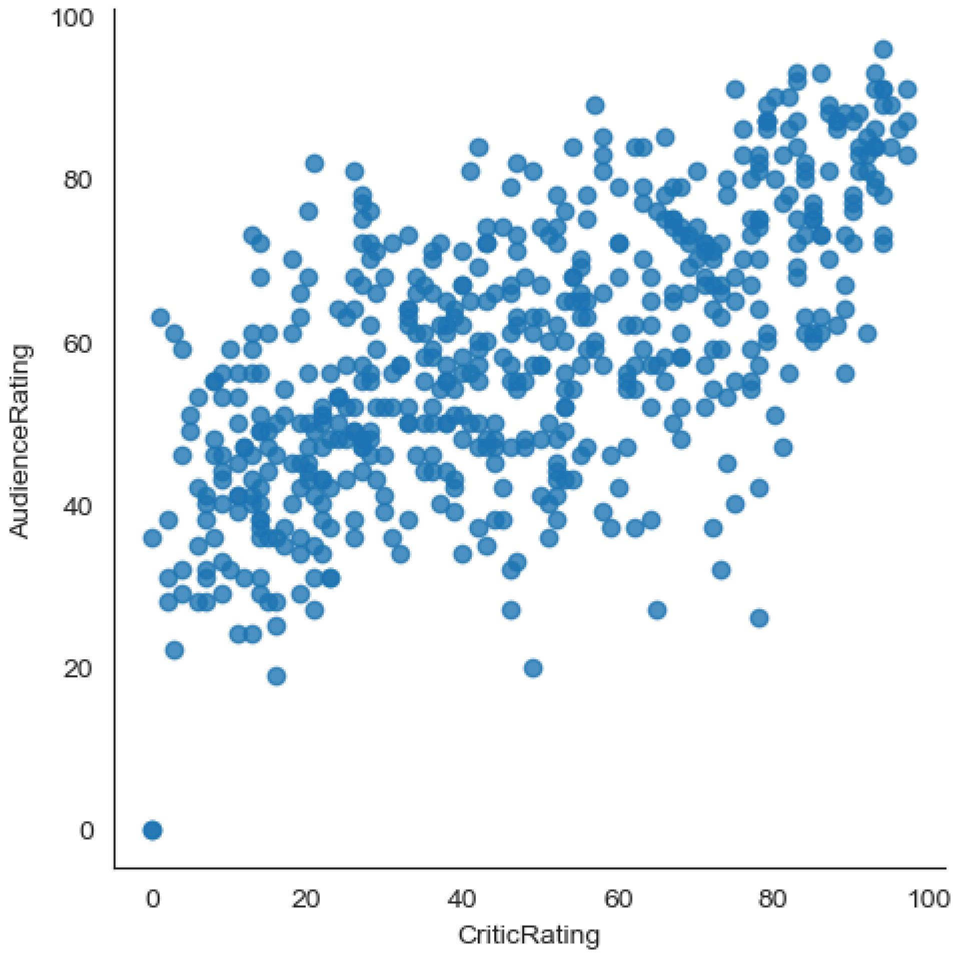
```
In [54]: # if you have 100 categories you cannot copy & paste all the things

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

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

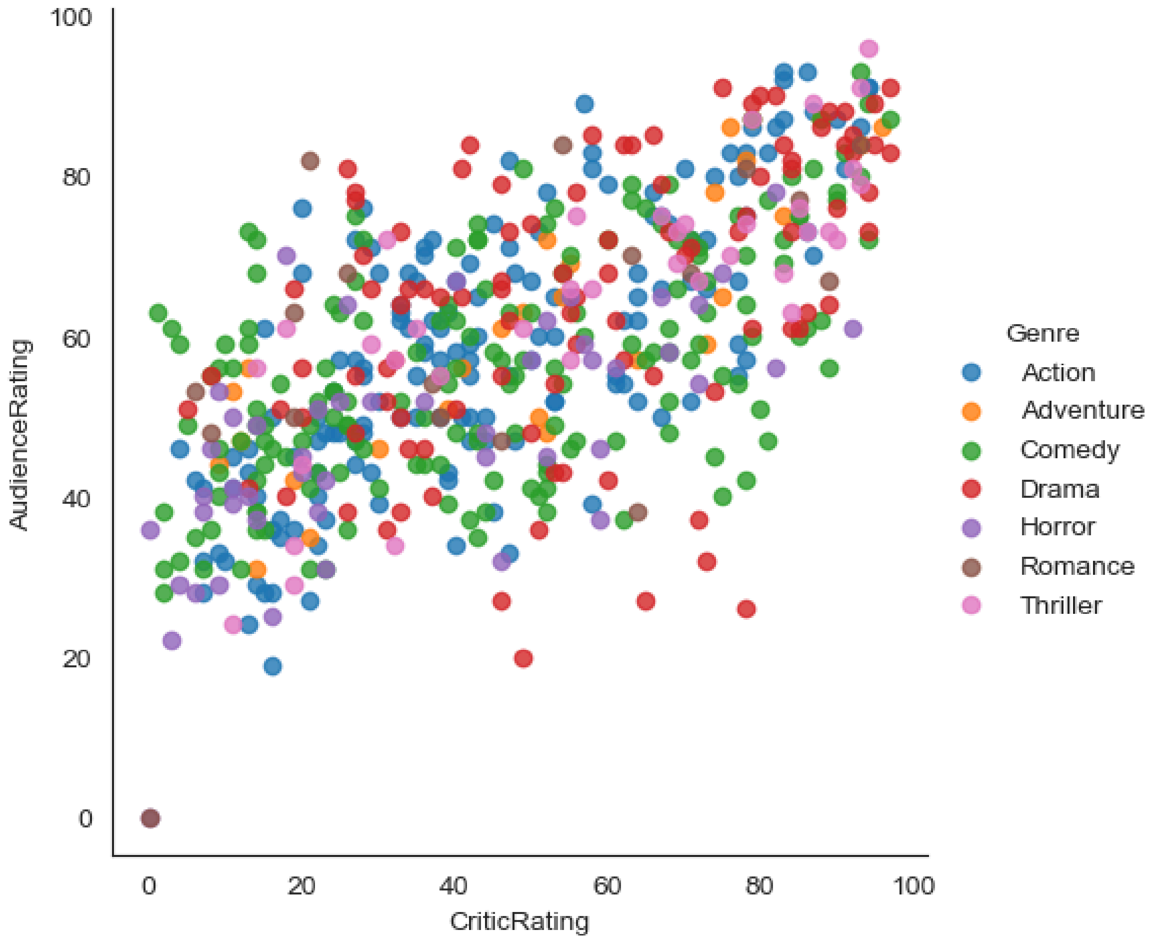
lmplot

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

```
In [56]: plt.show()
```

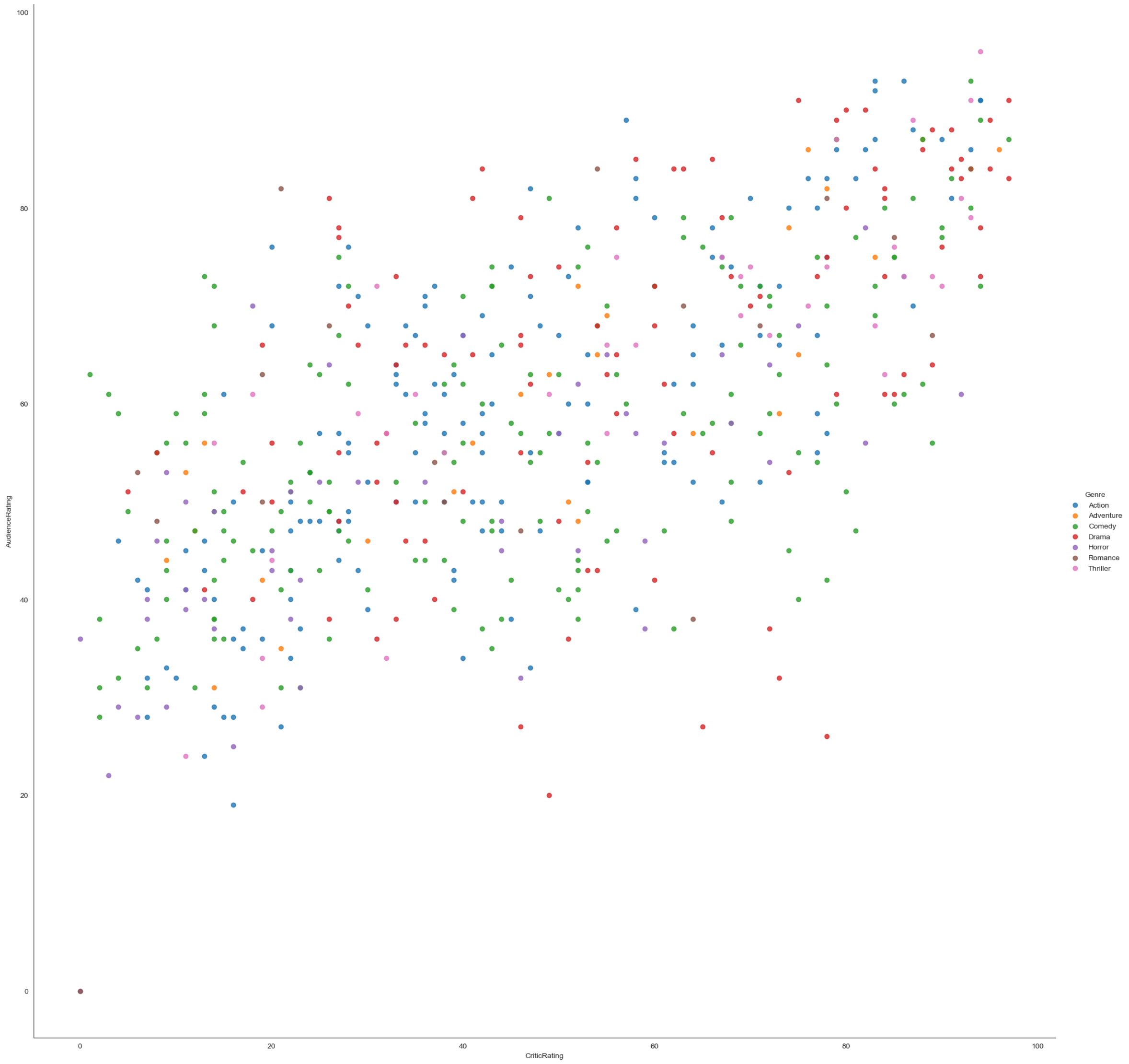


In [57]:
```
vis1 = sns.lmplot(data = movies, x='CriticRating' , y= 'AudienceRating',\
                  fit_reg = False, hue = 'Genre')
plt.show()
```



In [58]:

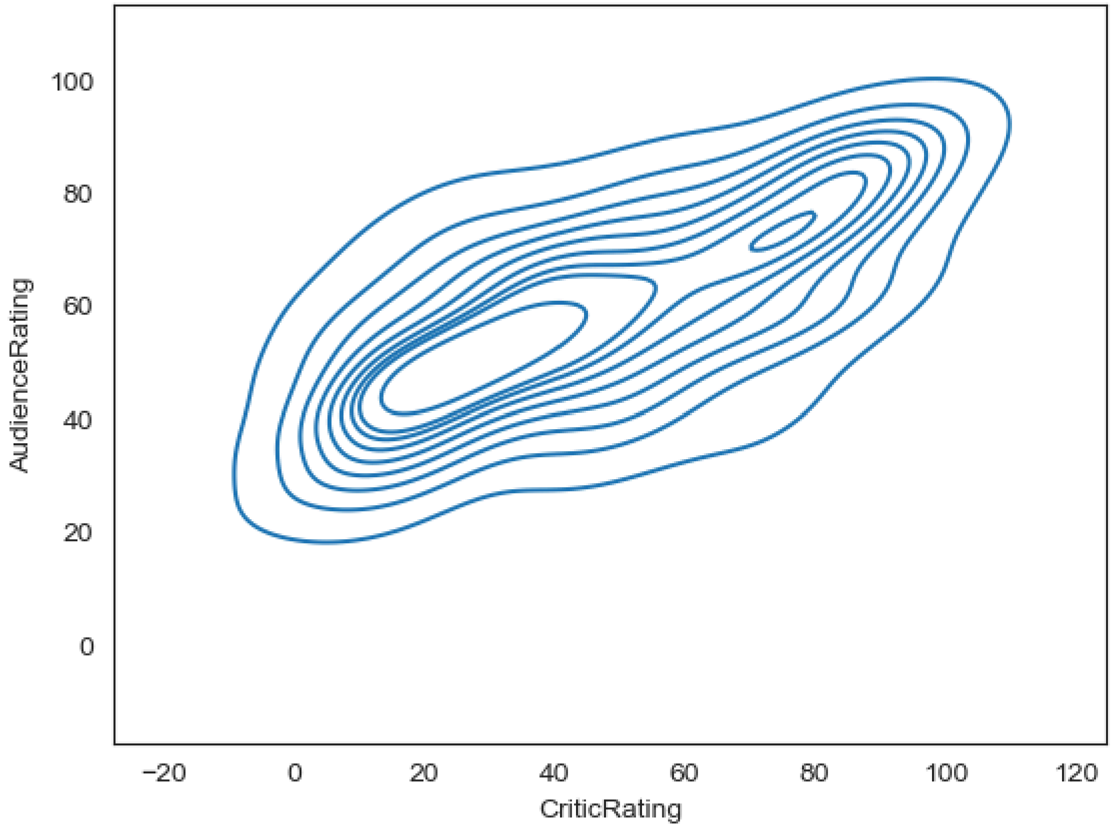
```
vis1 = sns.lmplot(data = movies, x='CriticRating' , y= 'AudienceRating',\
                  fit_reg = False, hue = 'Genre' ,height = 20,aspect =1)
plt.show()
```



KDE Plot

```
In [59]: # Kernal Density Estimate plot(KDE PLOT)
         # how can i visualize audience rating & critics rating using scatter plot
```

```
In [60]: k1 = sns.kdeplot(x= movies.CriticRating, y = movies.AudienceRating)
         plt.show()


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

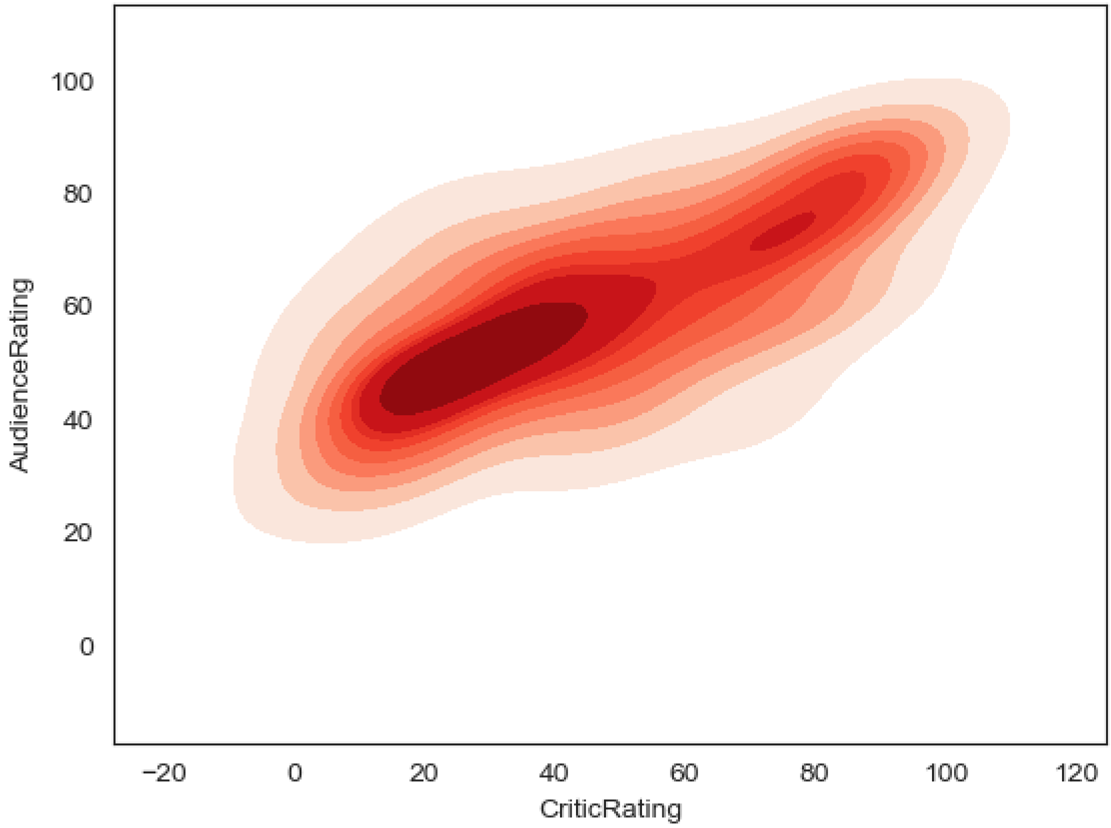


In [61]:

```
k1 = sns.kdeplot(x= movies.CriticRating , y = movies.AudienceRating , shade = Tr
```

In [62]:

```
plt.show()
```

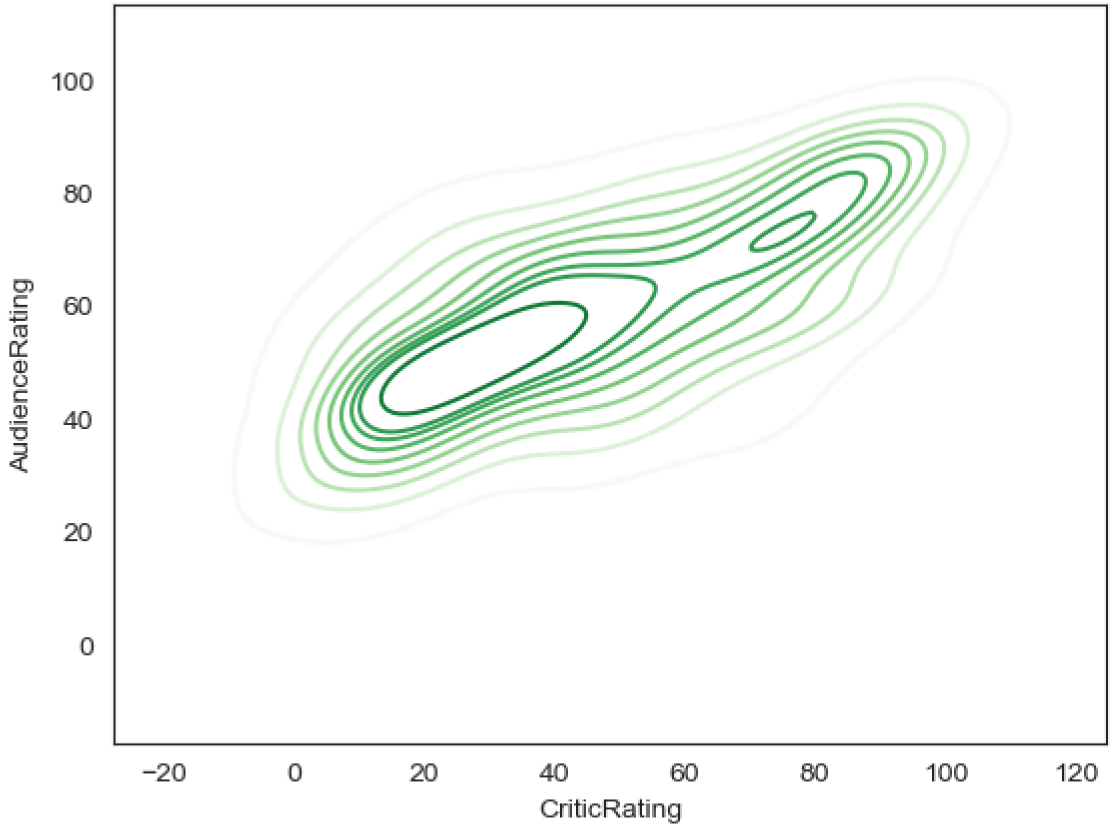


In [63]:

```
k1 = sns.kdeplot(x= movies.CriticRating , y = movies.AudienceRating ,shade_lowes
```

In [64]:

```
plt.show()
```

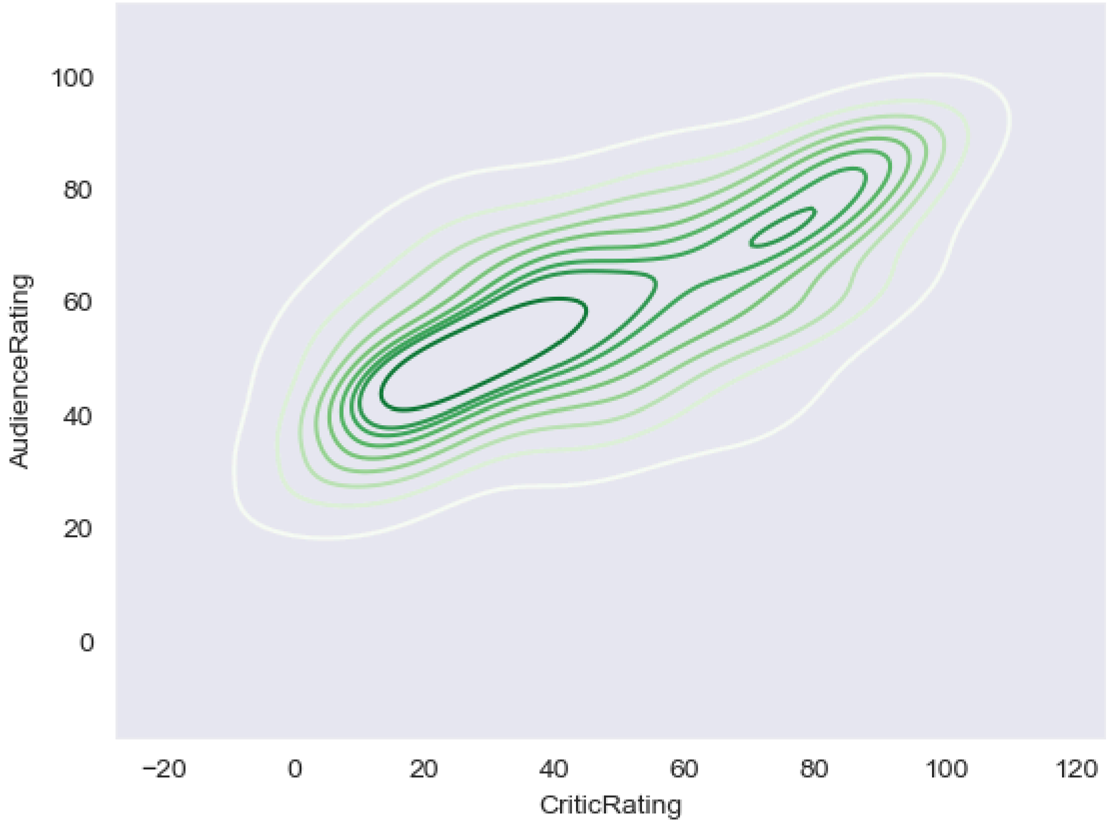


In [65]:

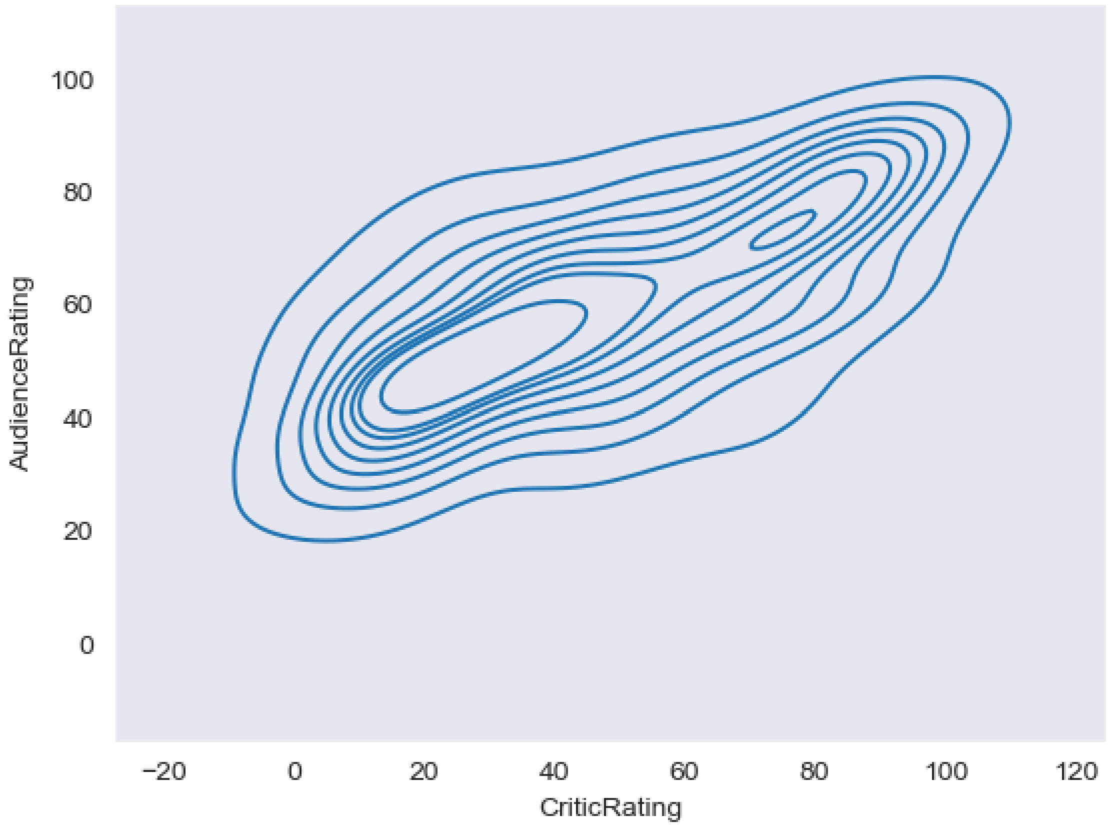
```
sns.set_style('dark')
k1 = sns.kdeplot(x= movies.CriticRating , y = movies.AudienceRating ,shade_lowes
```

In [66]:

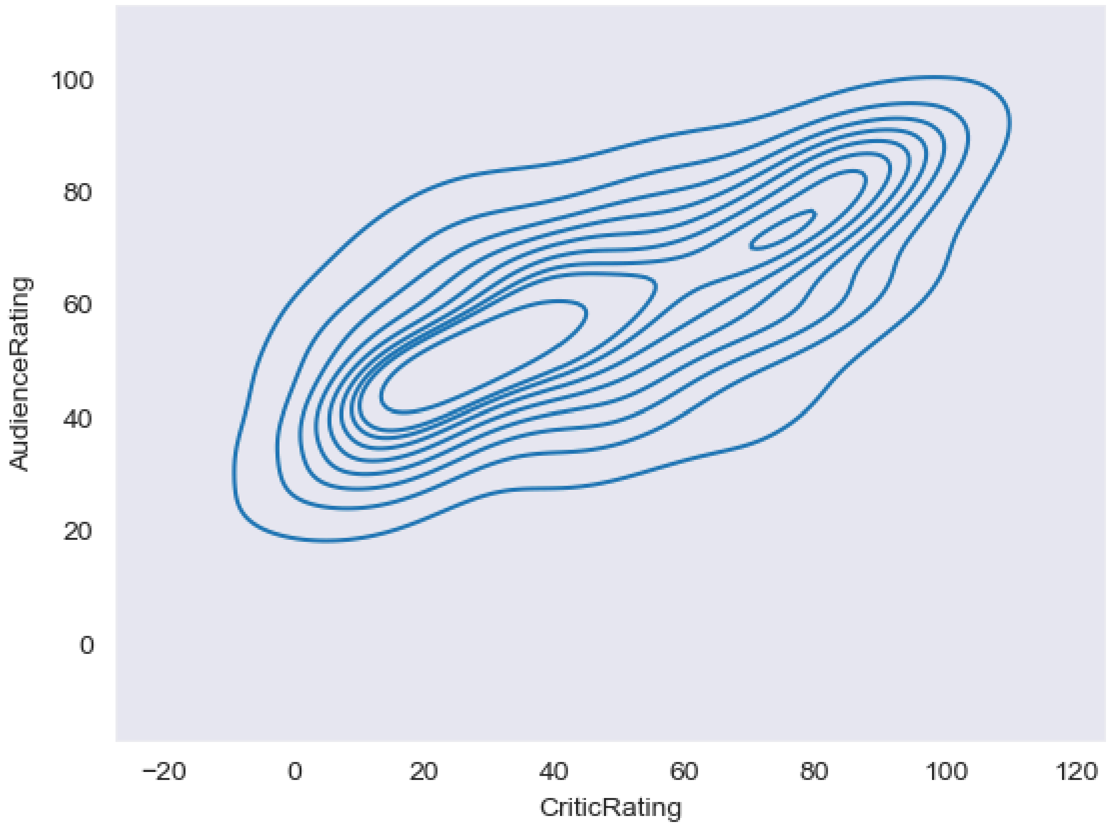
```
plt.show()
```



In [67]:
```
sns.set_style('dark')
k1 = sns.kdeplot(x= movies.CriticRating , y = movies.AudienceRating)
plt.show()
```



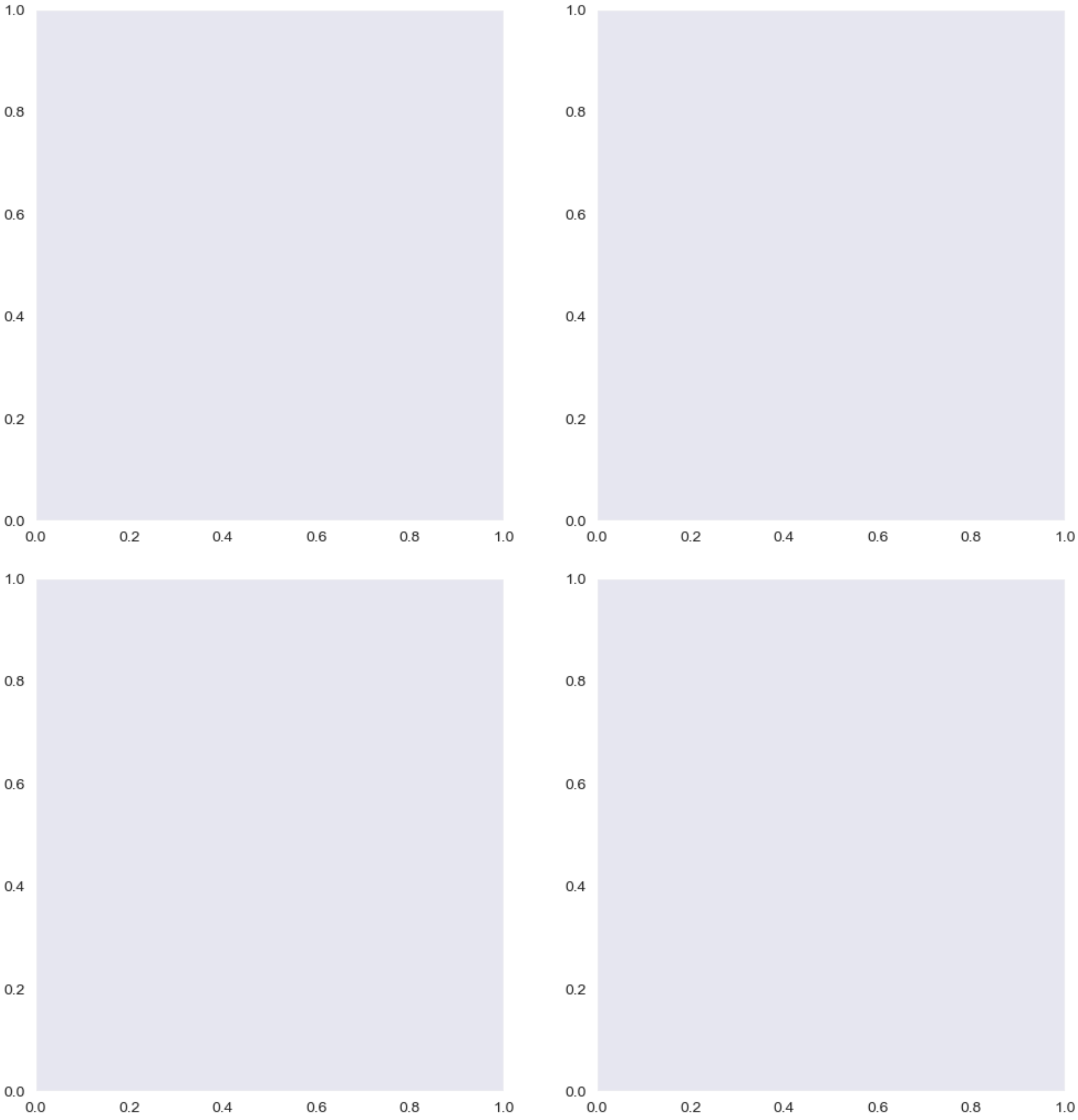
In [70]:
```
k2 = sns.kdeplot(x= movies.CriticRating , y = movies.AudienceRating)
plt.show()
```



Sub plots

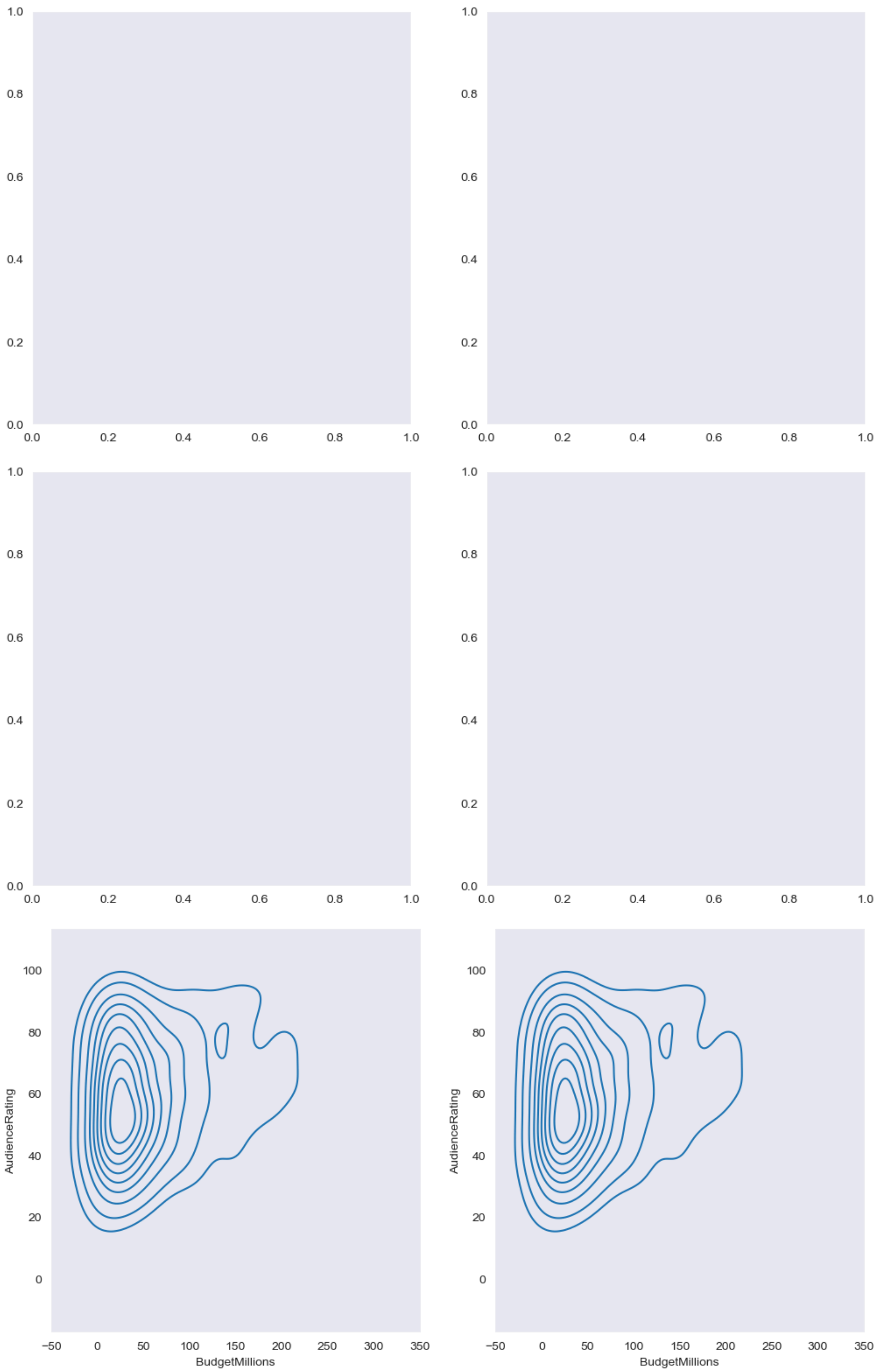
In [73]:

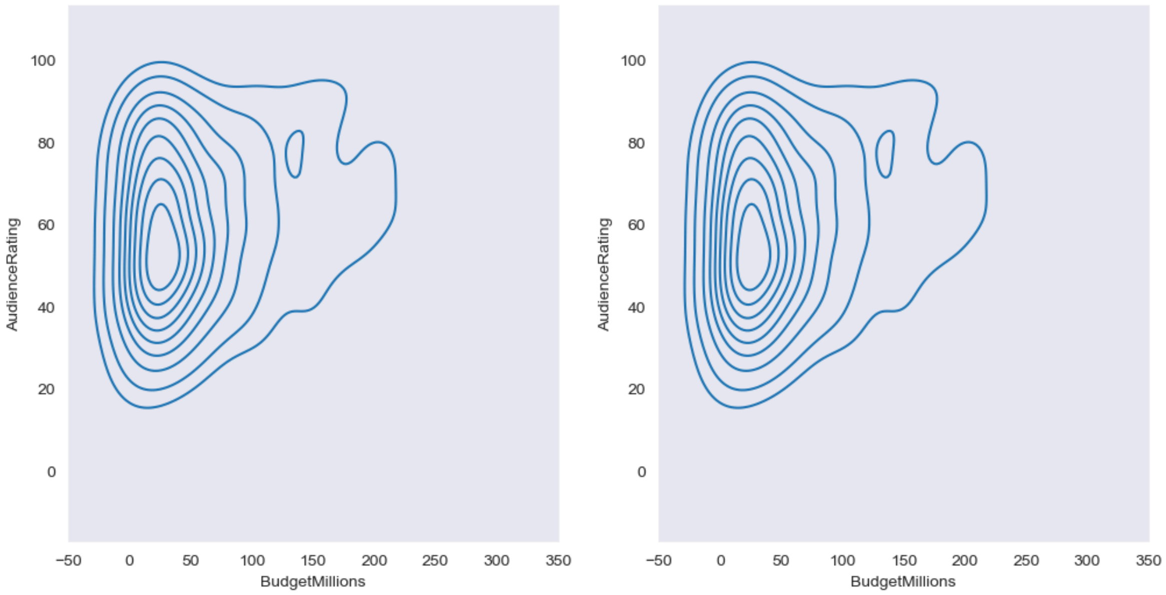
```
# subplots

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



In [77]:

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

k1 = sns.kdeplot(x = movies.BudgetMillions, y=movies.AudienceRating , ax = axes[
k2 = sns.kdeplot(x = movies.BudgetMillions, y=movies.AudienceRating , ax = axes[

plt.show()
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
AudienceRating
100
80
60
40
20
0
−50
0
50
100
150
200
250
300
350
BudgetMillions

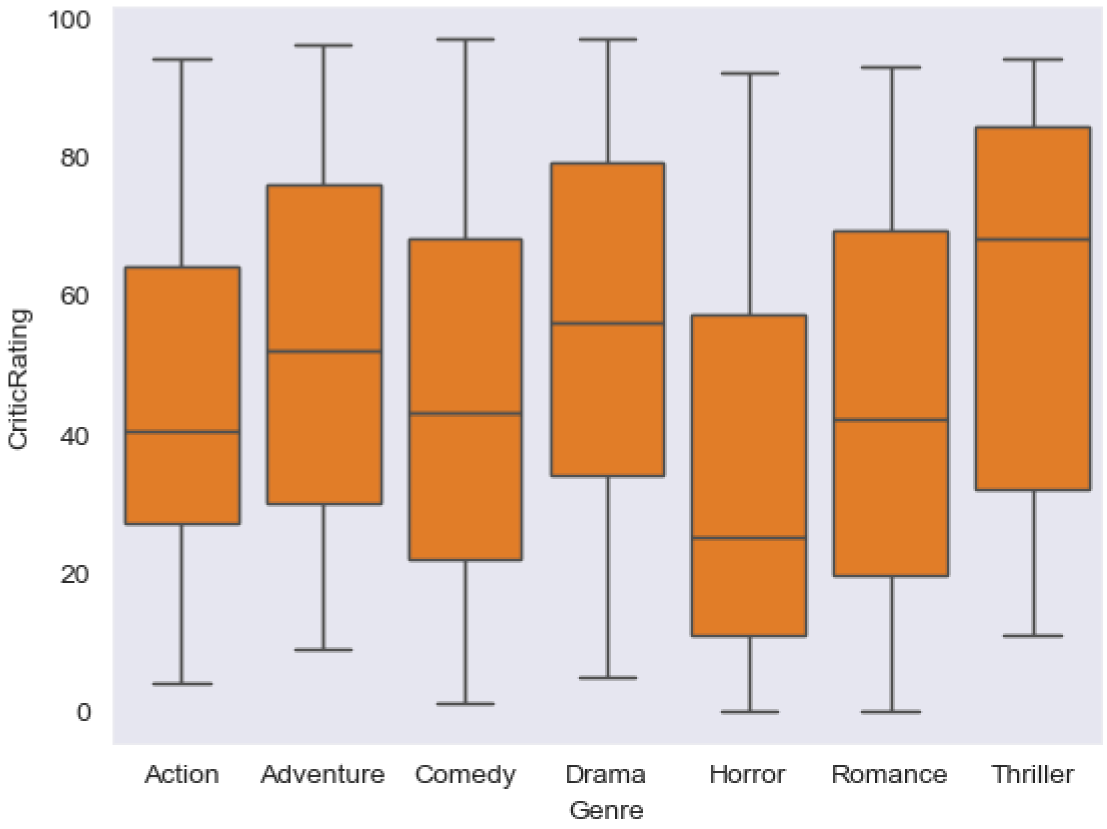


```
In [78]: axes
```

```
Out[78]: array([<Axes: xlabel='BudgetMillions', ylabel='AudienceRating'>,
                <Axes: xlabel='BudgetMillions', ylabel='AudienceRating'>],
               dtype=object)
```

Box plots

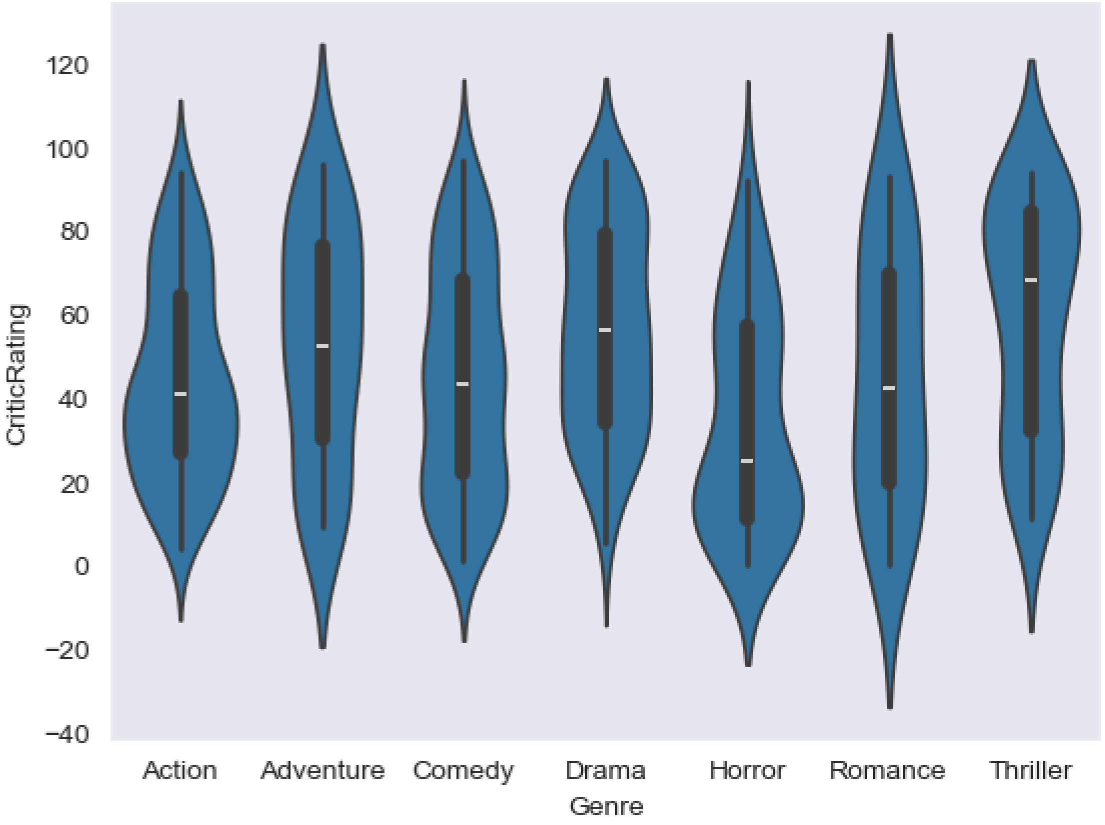
```
In [80]: w = sns.boxplot(data = movies , x = 'Genre' , y = 'CriticRating')
         plt.show()
```



violin plot

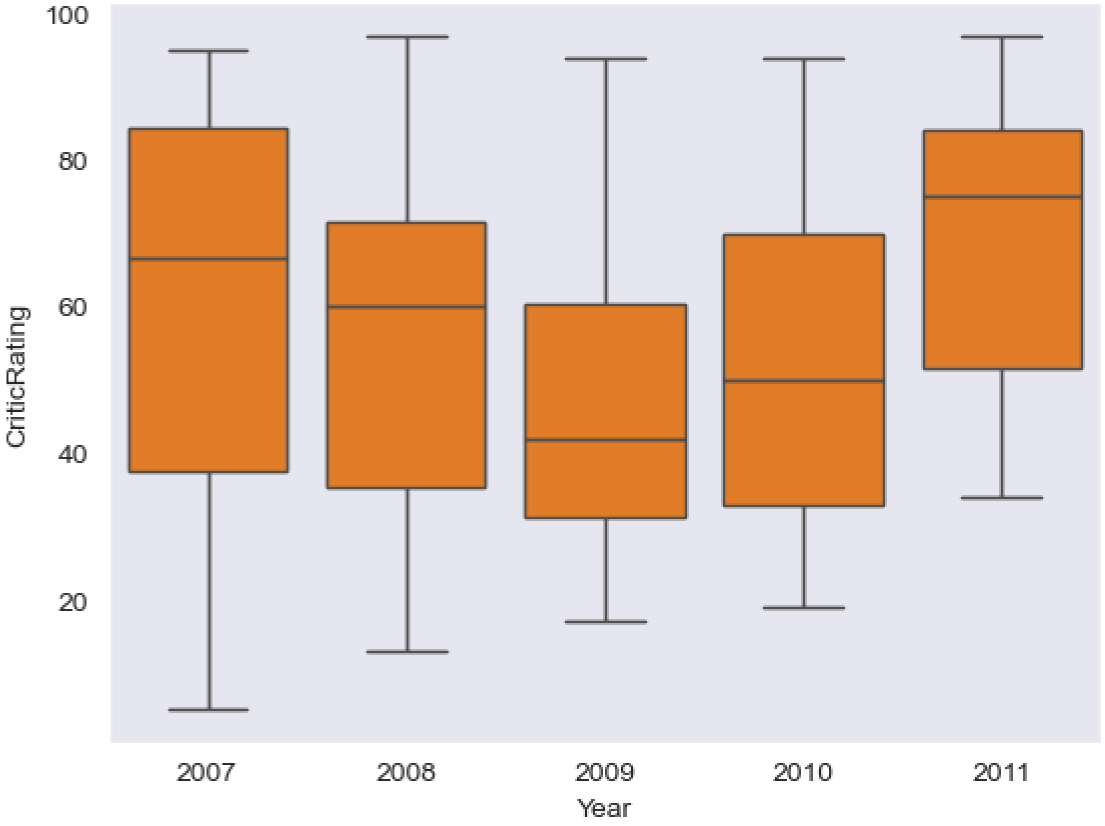
In [81]:

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

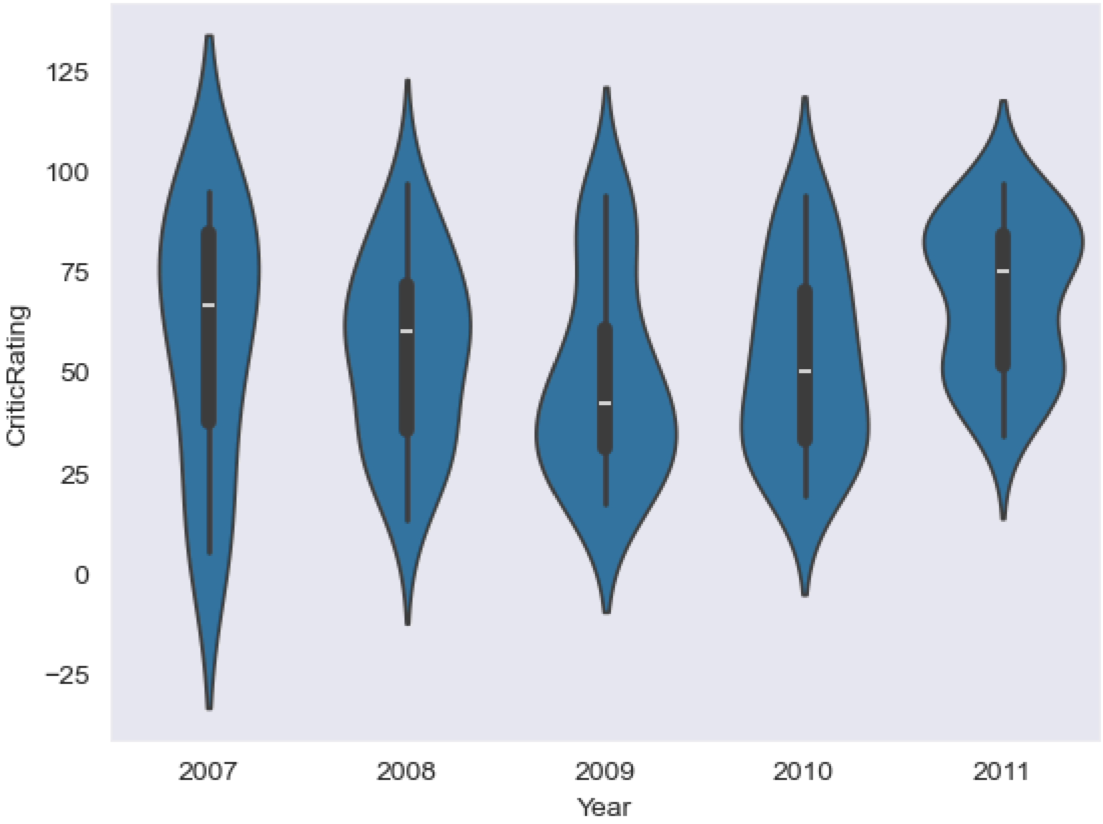


In [84]:

```
w1 = sns.boxplot(data = movies[movies.Genre == 'Drama'], x ='Year' , y='CriticRa
plt.show()
```



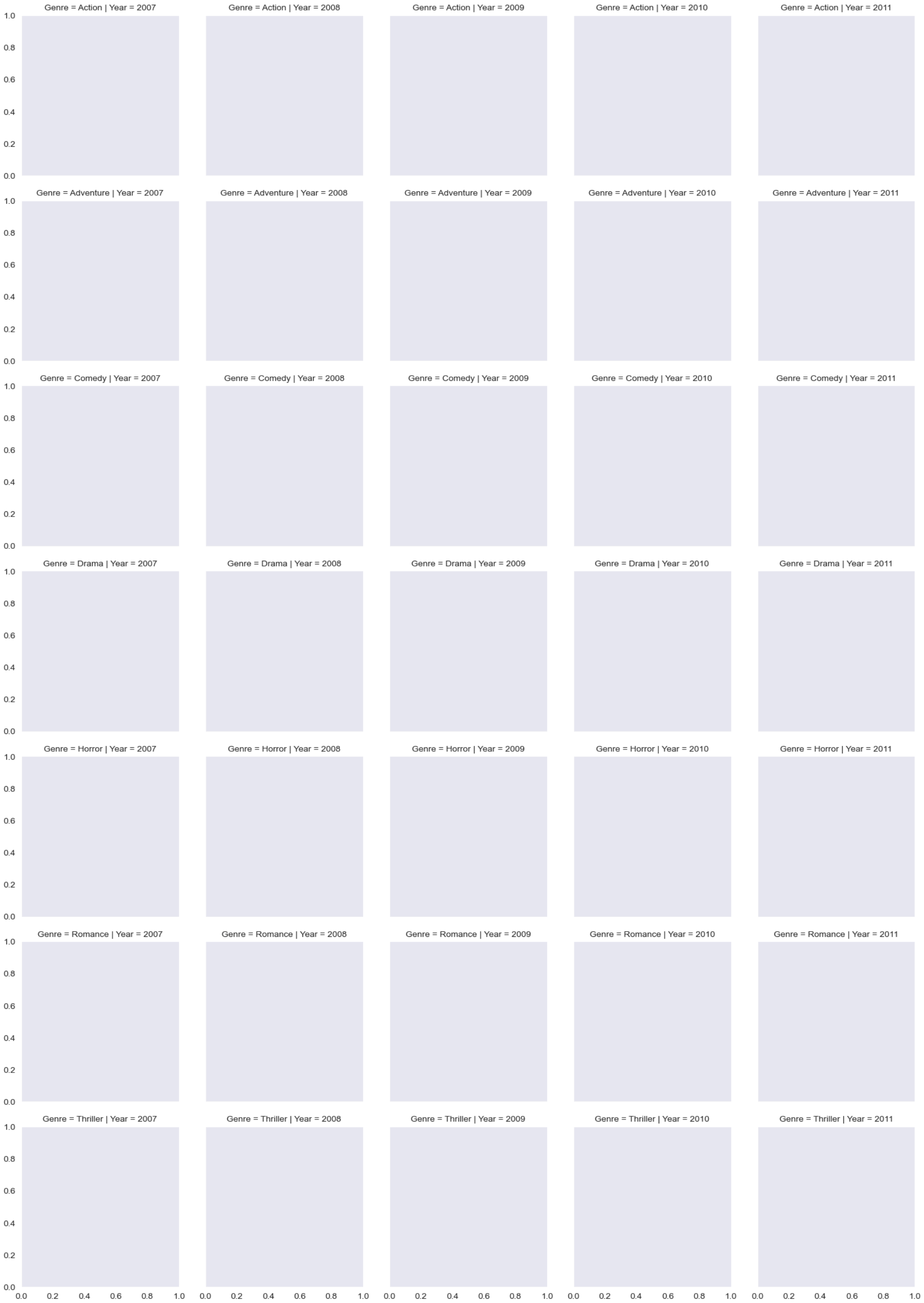
```
In [85]: z = sns.violinplot(data = movies[movies.Genre == 'Drama'], x ='Year' , y='Critic
         plt.show()
```



creating a Facet grid

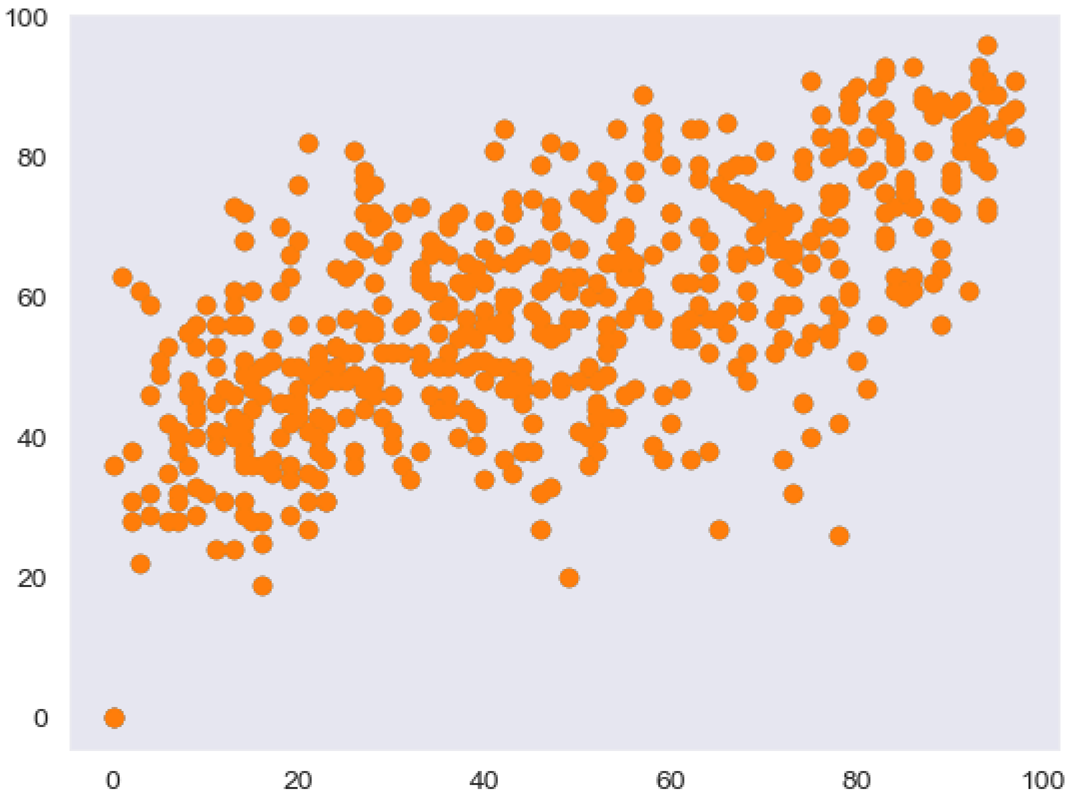
```
In [86]: g = sns.FacetGrid(movies, row = 'Genre', col='Year',hue = 'Genre')   # kind of s
```

```
In [87]: plt.show()
```



In [90]:

```
plt.scatter(movies.CriticRating , movies.AudienceRating)
plt.show()
```

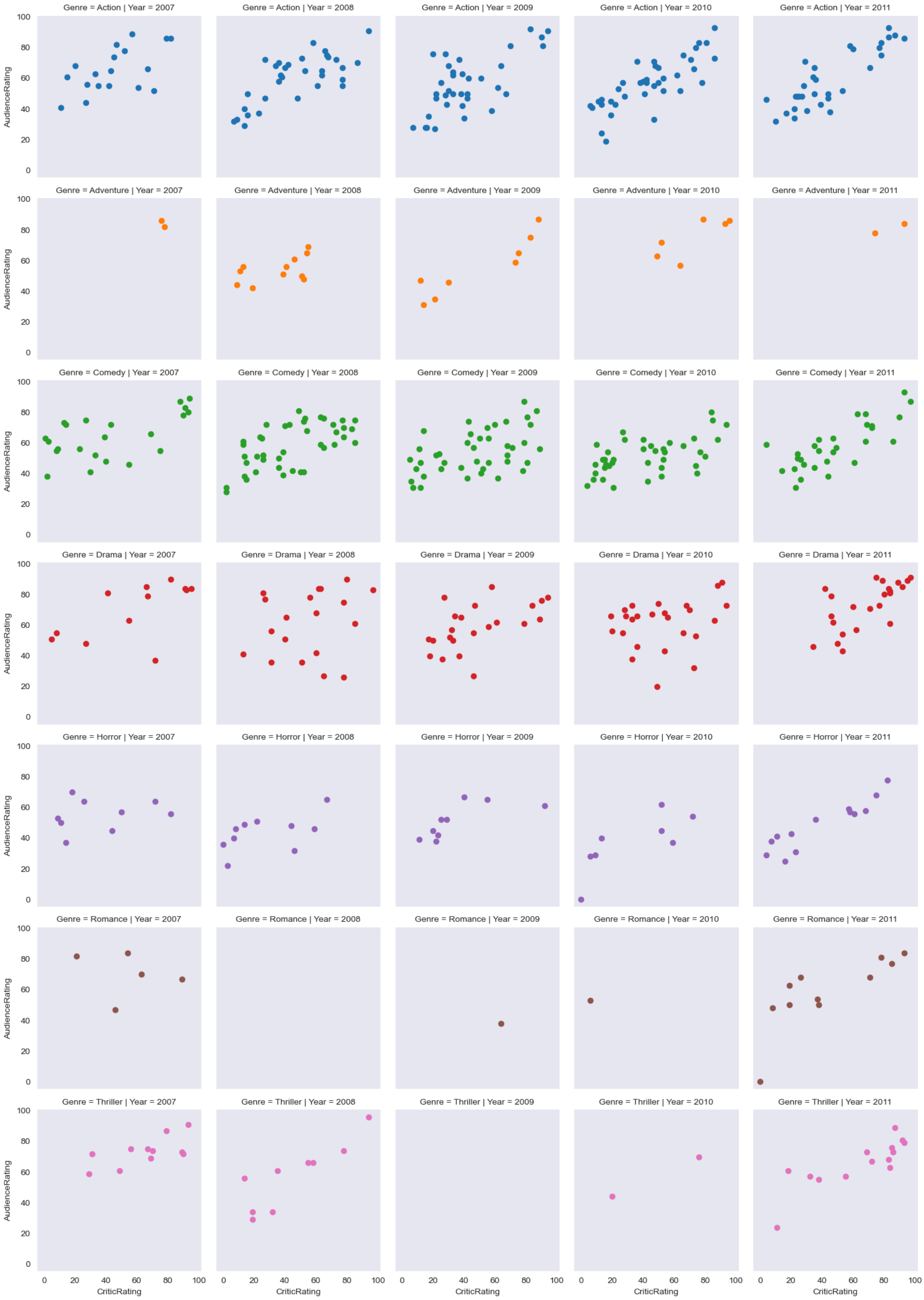


In [92]:

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

In [93]:

```
plt.show()
```

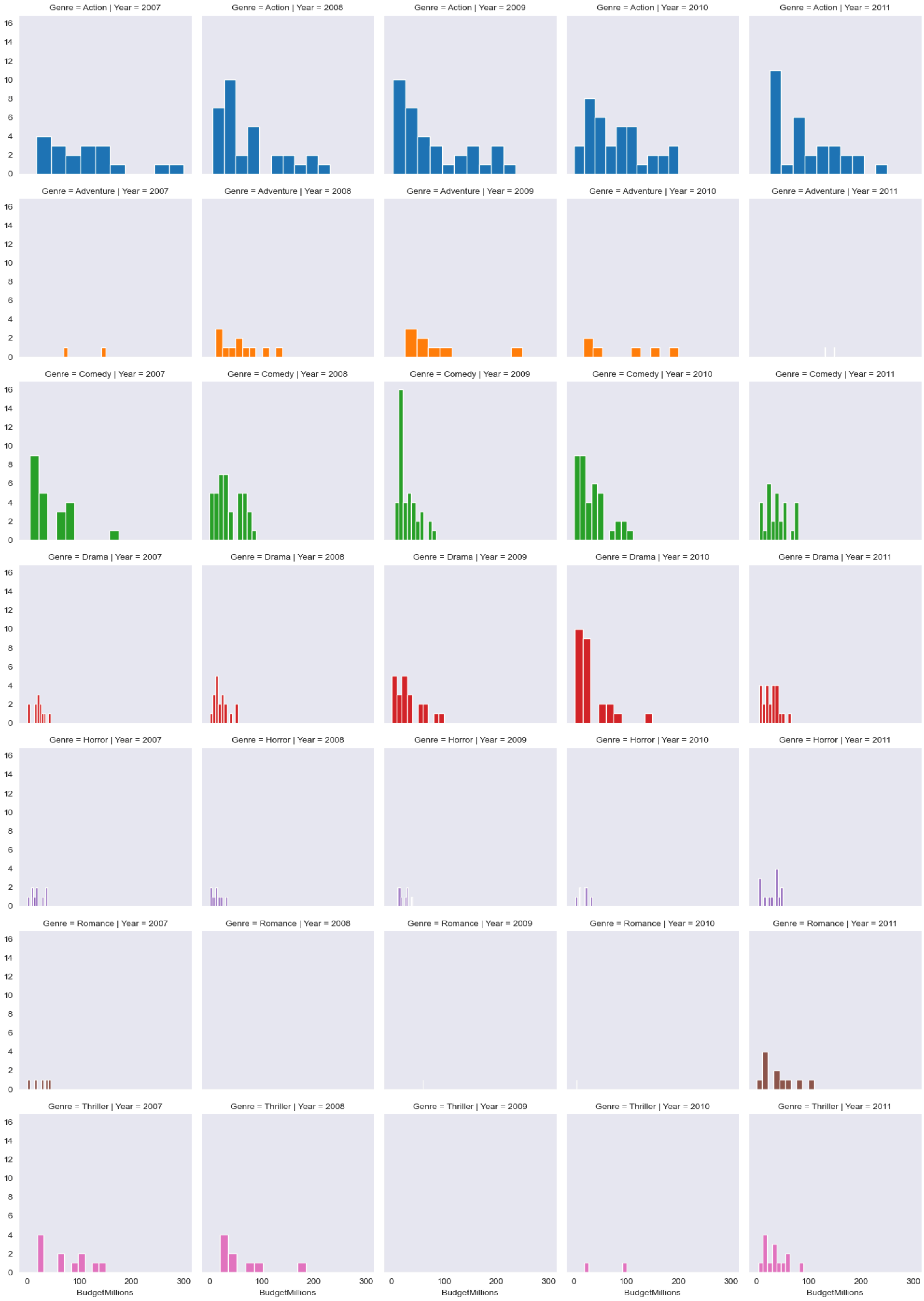


In [94]:

```
# 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
```

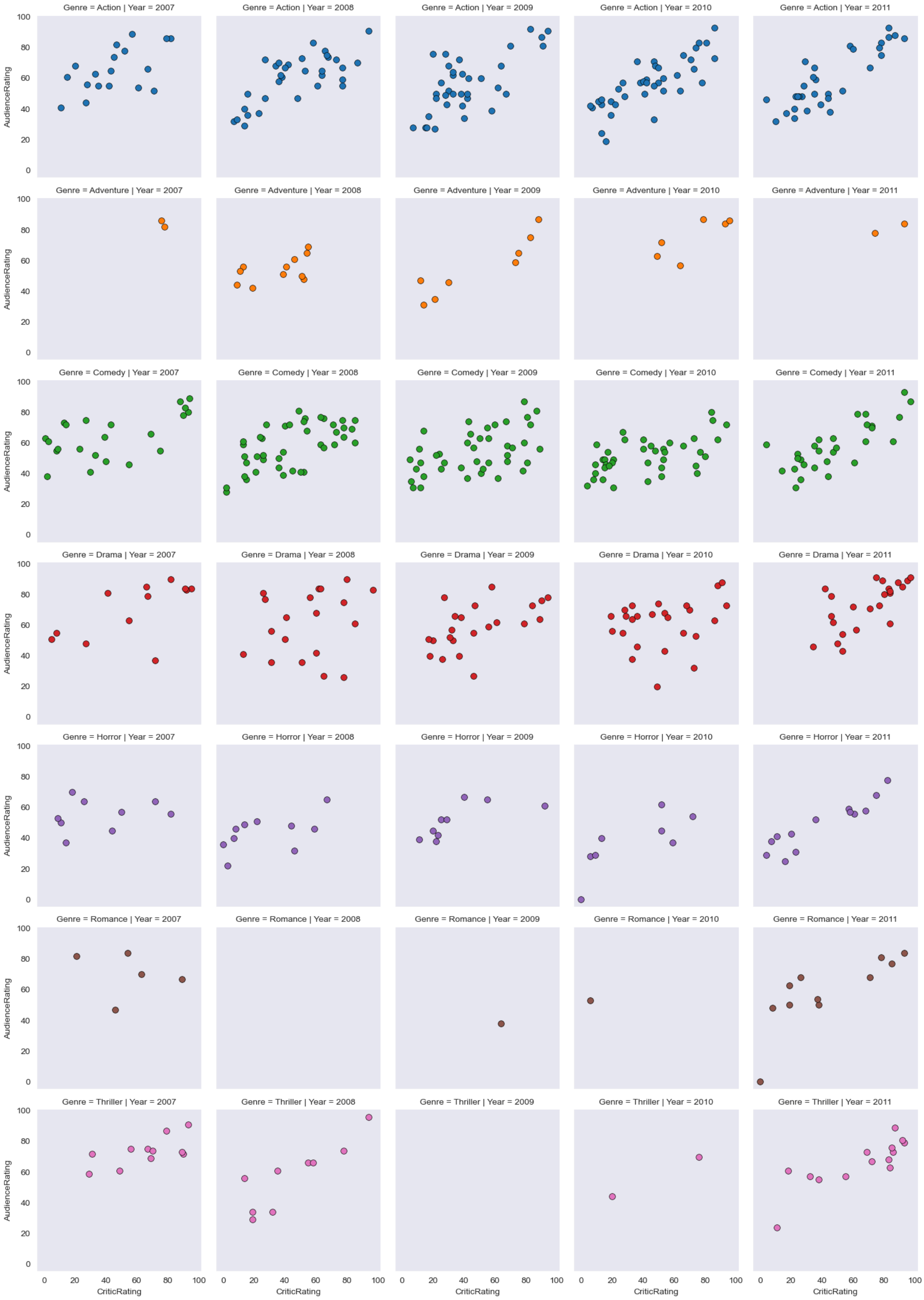
In [95]:

```
plt.show()
```



In [96]:
```python
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
```

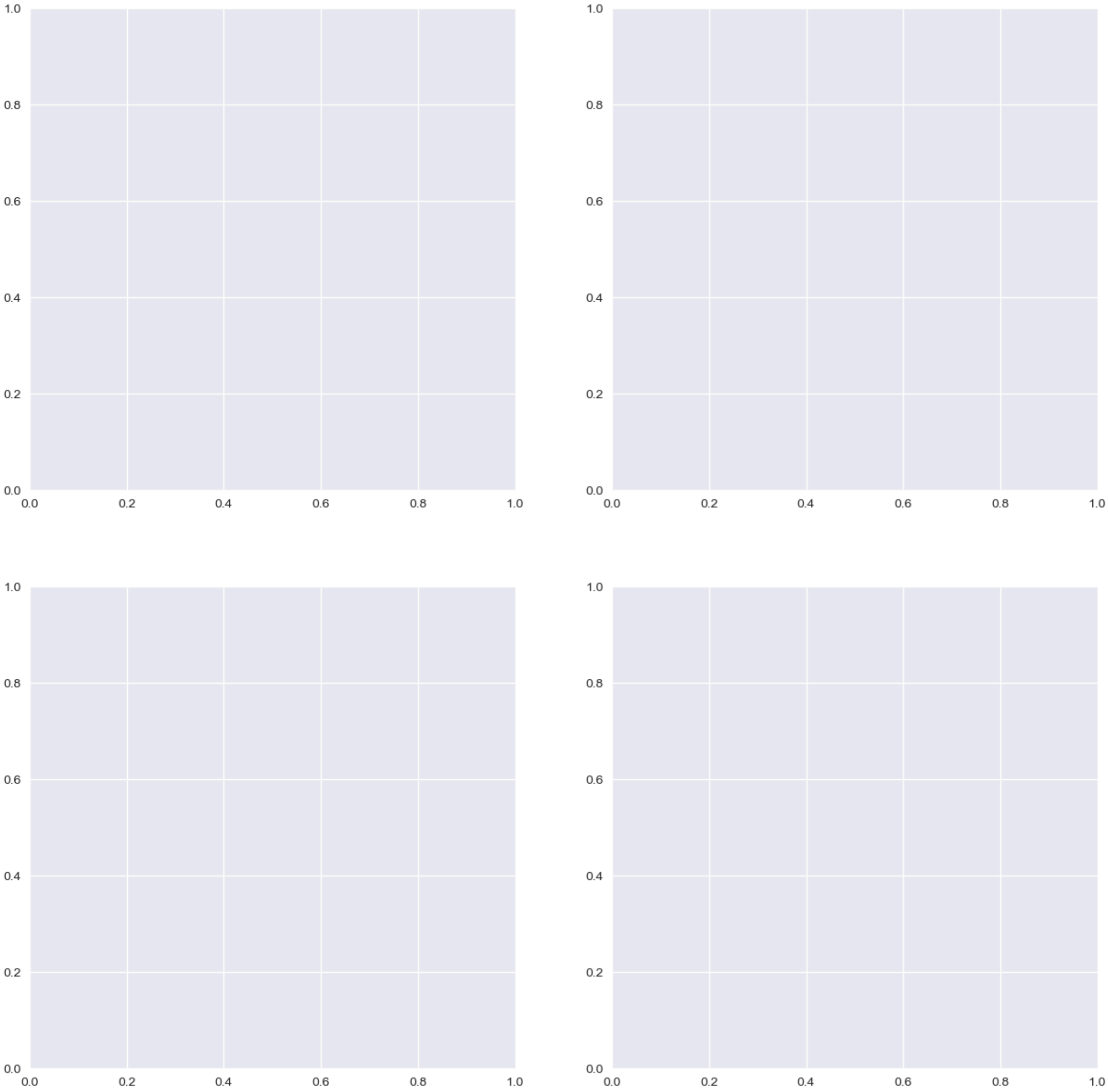
In [97]:
```python
plt.show()
```

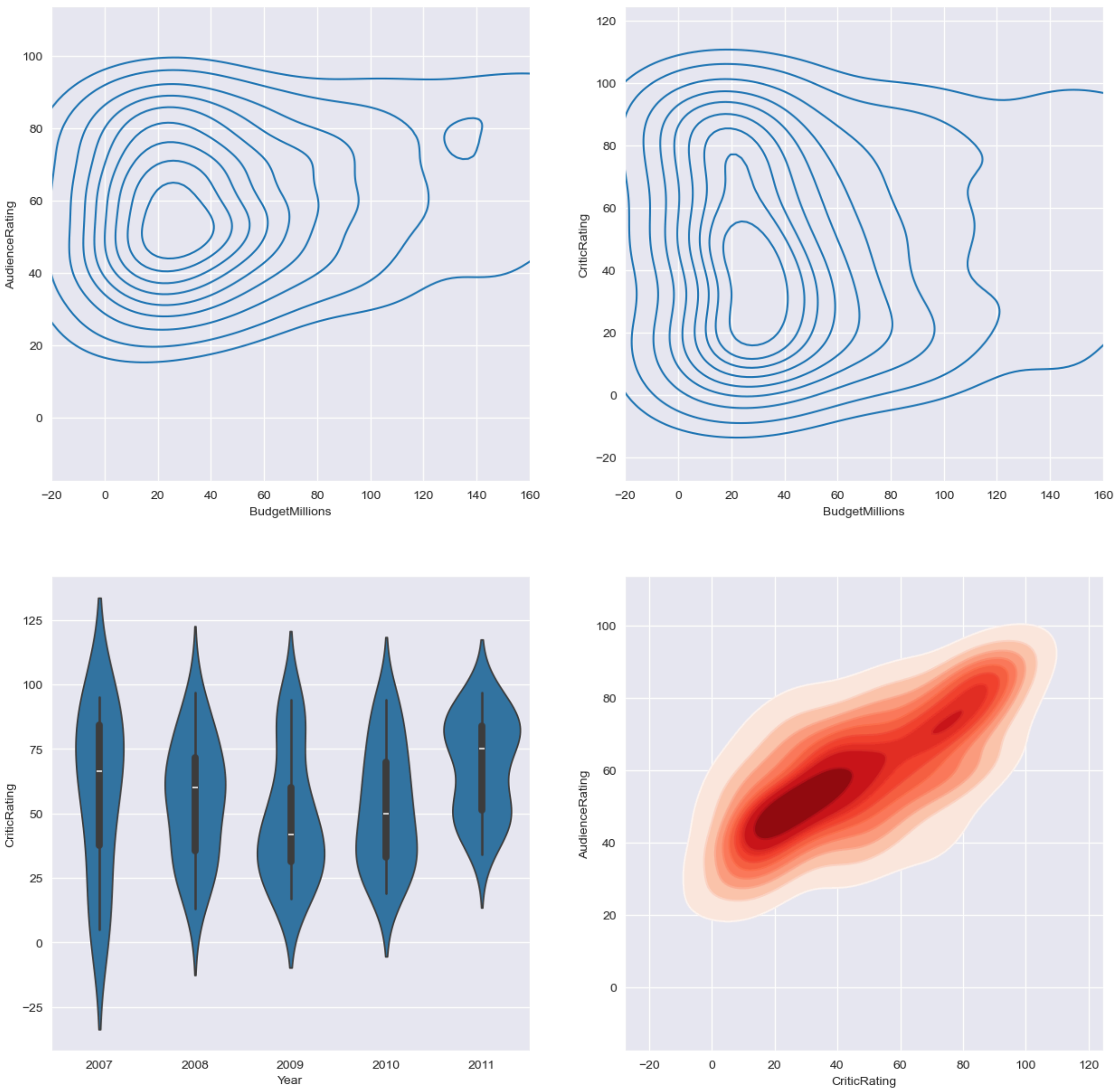


In [106…

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

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))
```

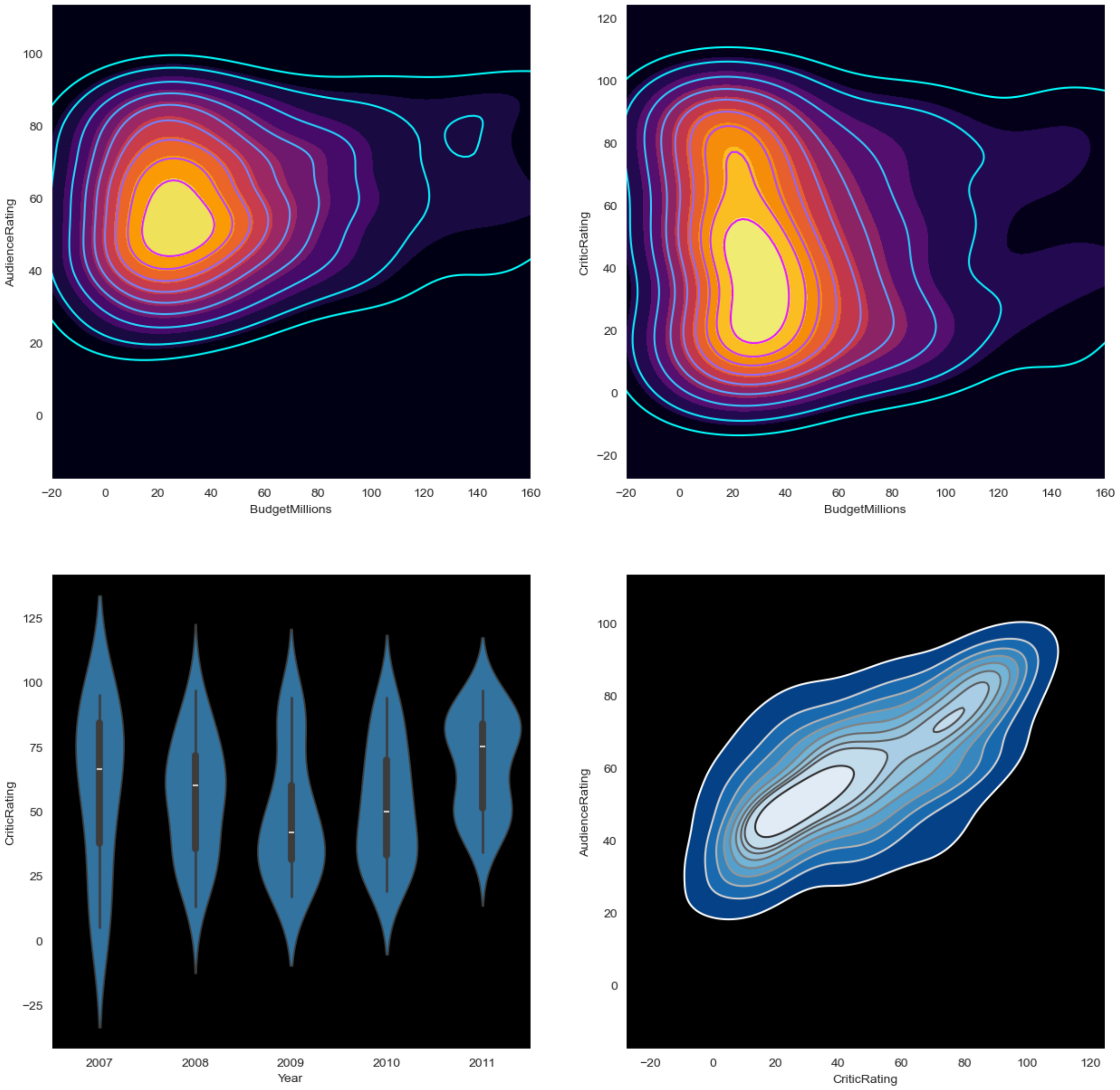
```
k2.set(xlim=(-20,160))

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

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

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

plt.show()
```





In [107…

```
# 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=(15,15))

#plot [0,0]
k1 = sns.kdeplot(x = movies.BudgetMillions , y = movies.AudienceRating, \
                 shade = True, shade_lowest = True, cmap = 'inferno',\
                 fill = 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, cmap='inferno',\
                 fill = 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,cmap='Blues_r', \
                 fill = True , ax=axes[1,1])

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


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

plt.show()
```



In [ ]:

```
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> Factet grid
9> Building dashboards
```

In [ ]:

```
**********************Finally EDA Completed***********
```