Movie_Rating_Analysis(Advanced Visualization)

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

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

Out[2]:
```
'C:\\Users\\yogay'
```

In [4]:
```
movies=pd.read_csv(r'C:\Users\yogay\OneDrive\Documents\Data Science\13th\MOVIE RATINGS _ ADVANCE VISUALIZATION
```

In [5]:
```
movies
```

Out[5]:

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

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

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

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

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

Out[8]:

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

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

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

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

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

Out[11]:

	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 [13]:
```
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 [14]:
```
movies.describe()
# if you look at the year the data type is int
#but when you look at the mean value it showing 2009 which is meaningless
# we have to change to categroy type
# also from object datatype we will convert to category datatypes
```

Out[14]:

	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 [15]:
```
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: object
```

In [16]:
```
movies.Film
```

Out[16]:
```
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 [17]:
```
movies.Film = movies.Film.astype('category')
```

In [18]:
```
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 [19]:
```
movies.head()
```

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 [20]:
```
movies.Genre = movies.Genre.astype('category')
movies.Year = movies.Year.astype('category')
```

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    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 [22]:
```
movies.Genre
```

Out[22]:
```
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 [23]:
```
movies.Film
```

Out[23]:
```
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 Re
volt', 'Zodiac', 'Zombieland ', 'Zookeeper']
```

In [24]:
```
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 [25]:
```
movies.Genre.cat.categories
```

Out[25]:
```
Index(['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance',
       'Thriller'],
      dtype='object')
```

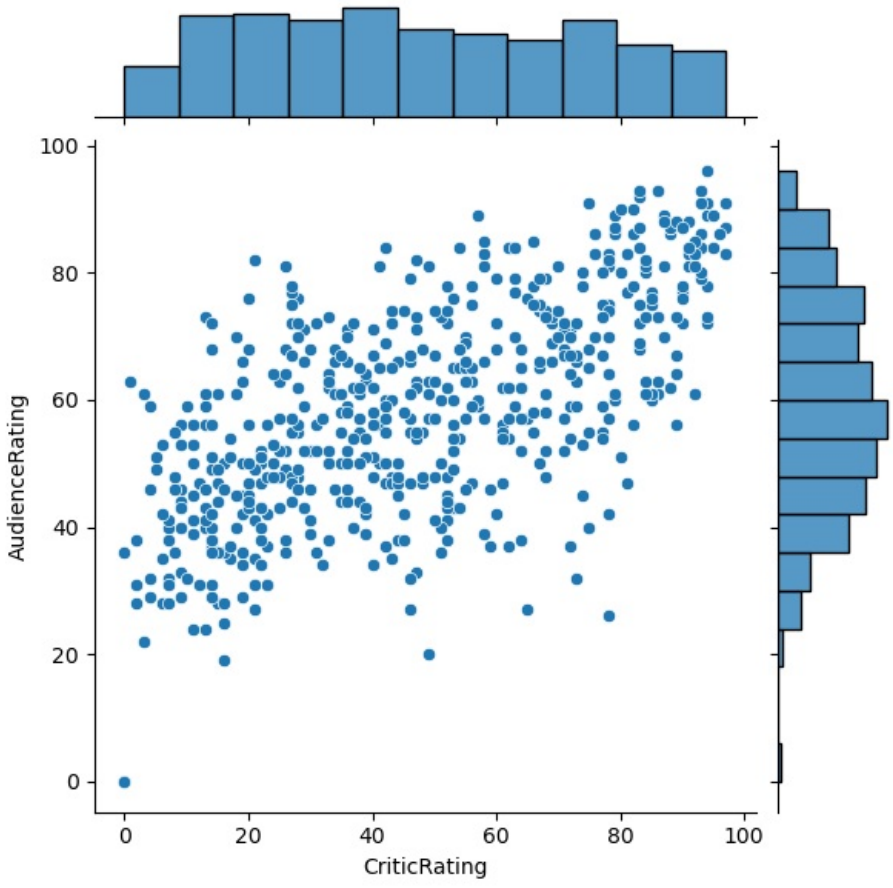
In [26]:
```
movies.describe()
```

Out[26]:

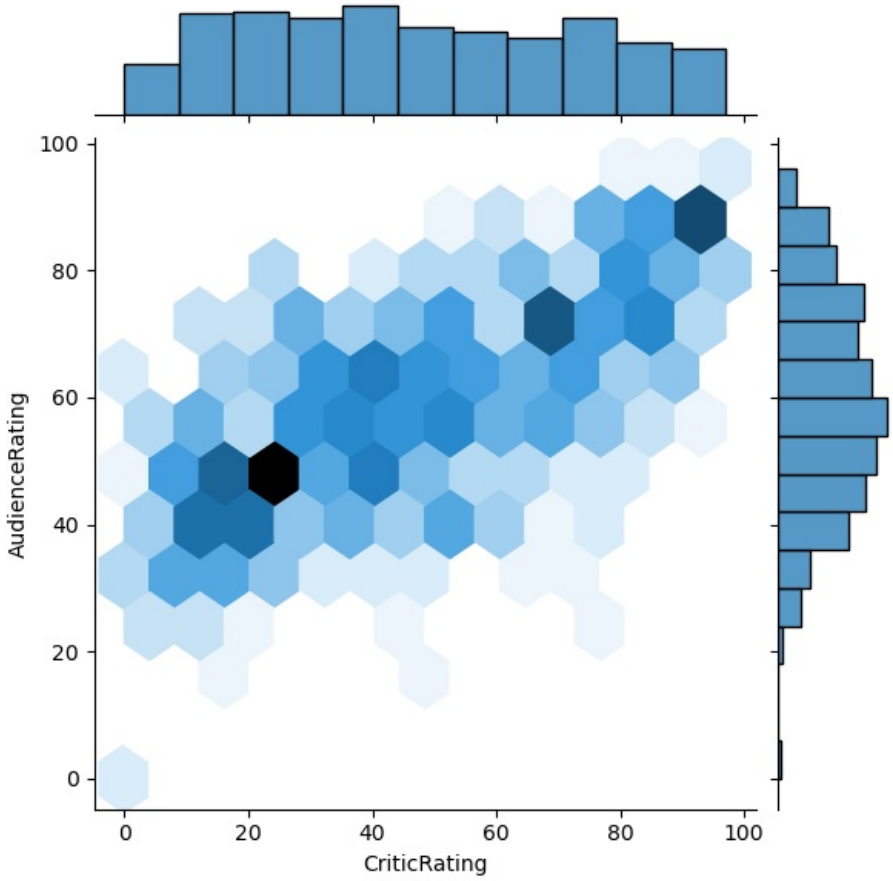
	CriticRating	AudienceRating	BudgetMillions
count	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136
std	26.413091	16.826887	48.731817
min	0.000000	0.000000	0.000000
25%	25.000000	47.000000	20.000000
50%	46.000000	58.000000	35.000000
75%	70.000000	72.000000	65.000000
max	97.000000	96.000000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

In [27]:
```
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [28]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='AudienceRating')
```

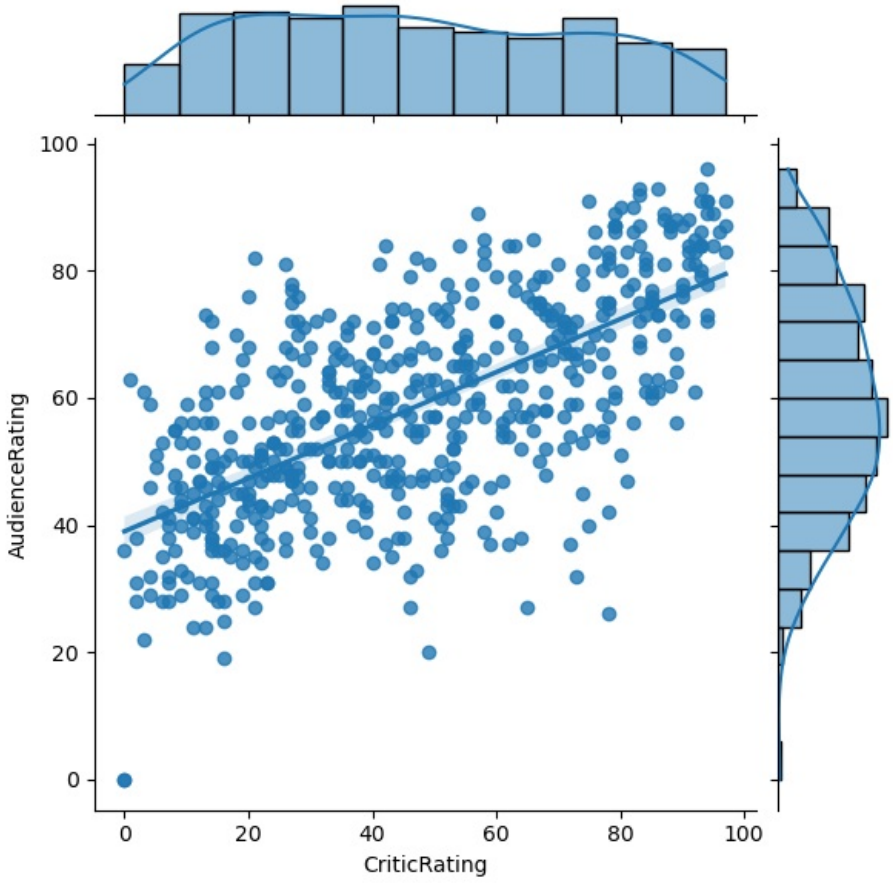


In [29]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='AudienceRating',kind='hex')
```

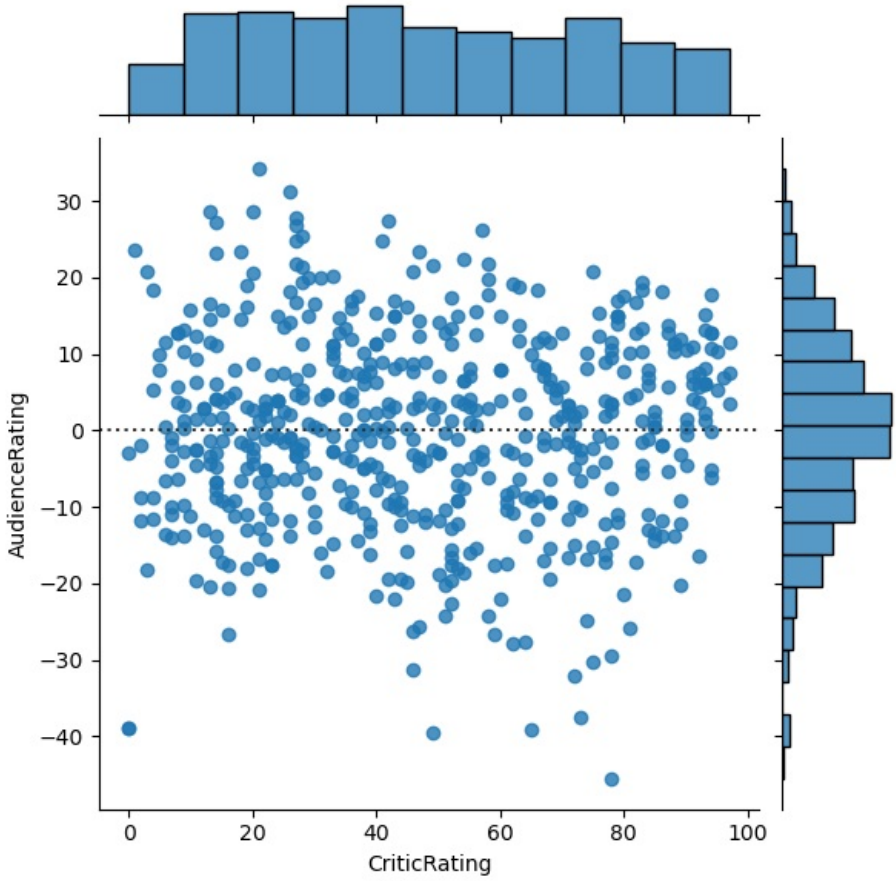


kind : { "scatter" | "kde" | "hist" | "hex" | "reg" | "resid" }

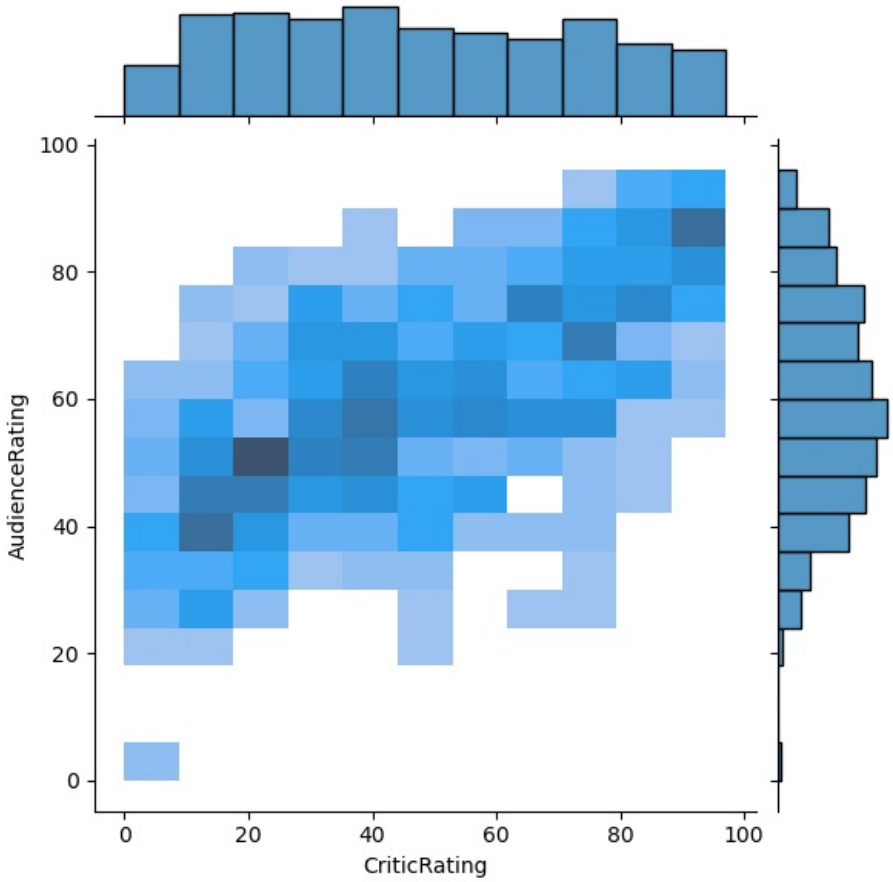
In [30]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='AudienceRating',kind='reg')
```



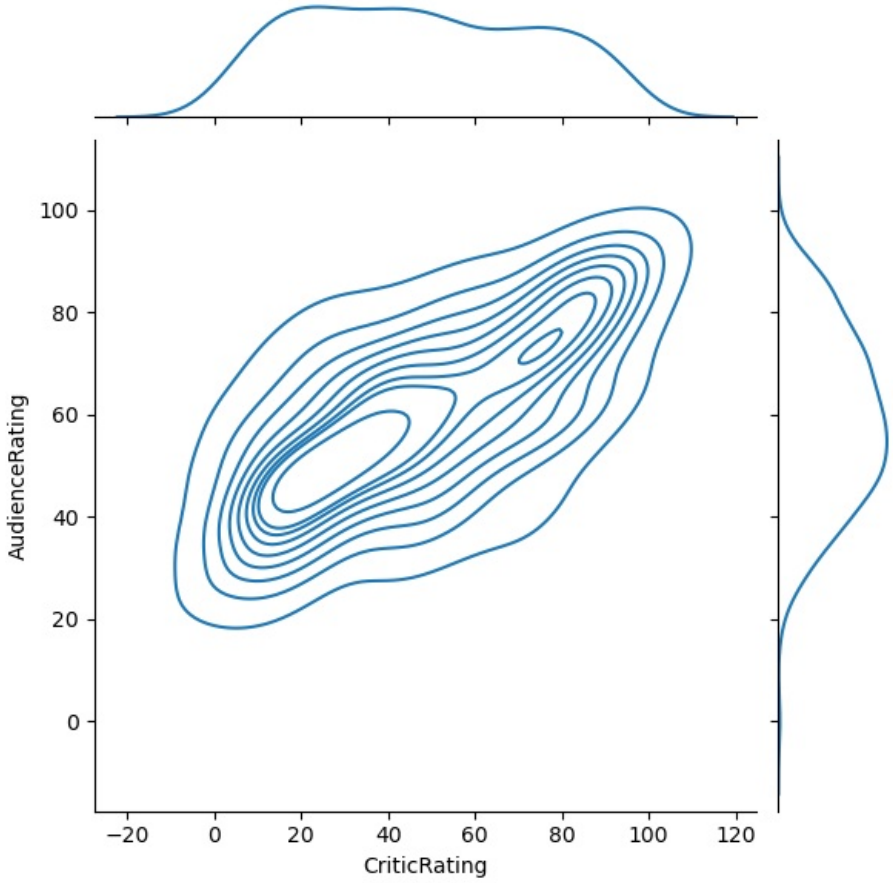
In [31]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='AudienceRating',kind='resid')
```



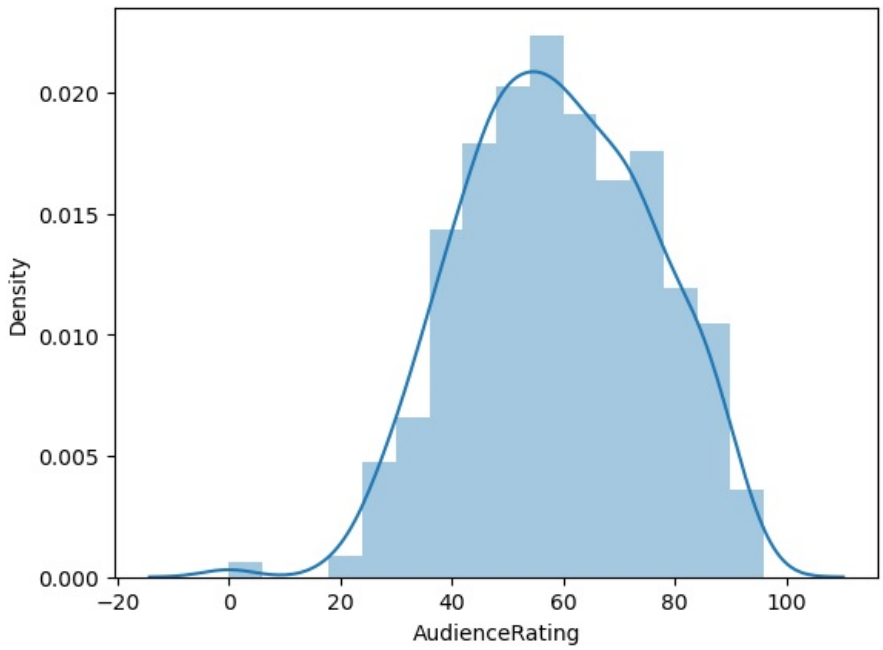
In [32]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='AudienceRating',kind='hist')
```



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

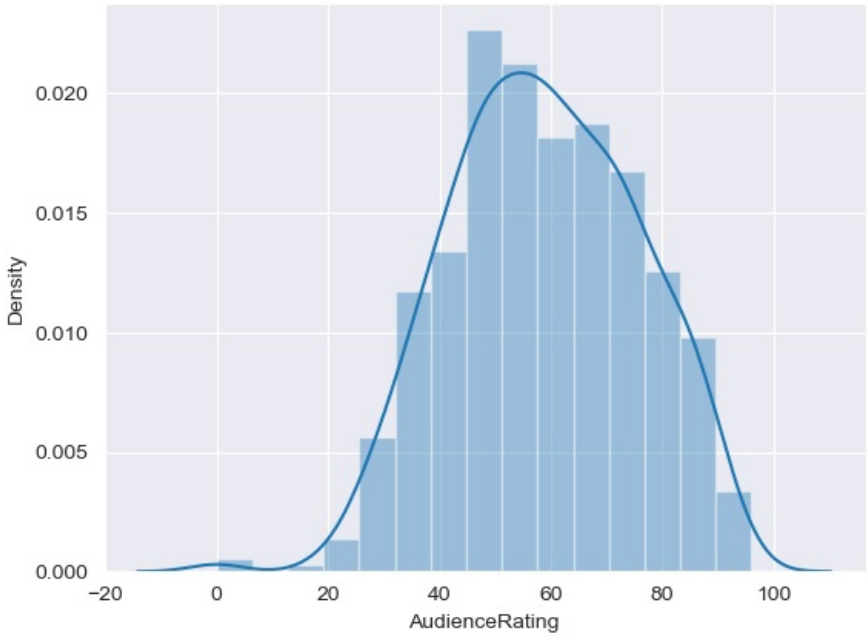


In [34]:
```
m1 = sns.distplot(movies.AudienceRating)
```



In [35]:
```
sns.set_style('darkgrid')  #change background
```

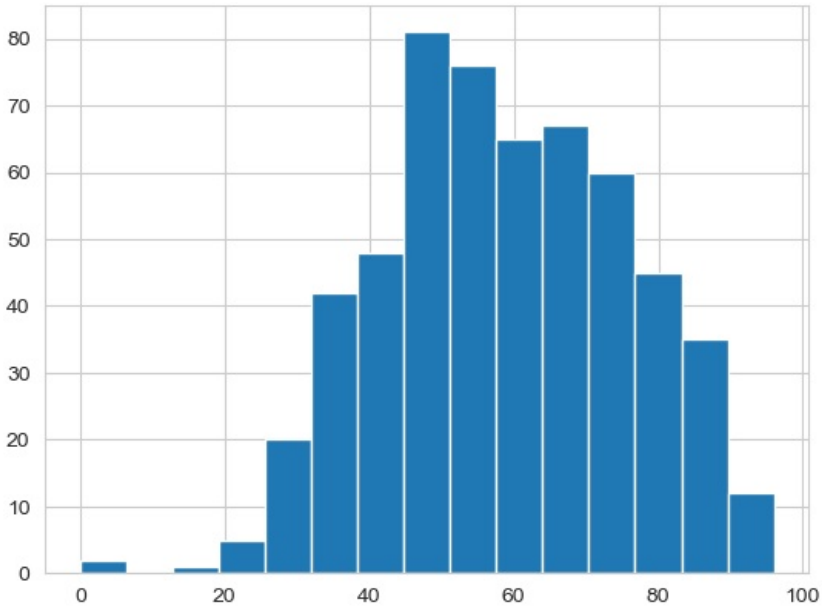
In [36]:
```
m2 = sns.distplot(movies.AudienceRating, bins = 15)
```



style : dict, or one of {darkgrid, whitegrid, dark, white, ticks}

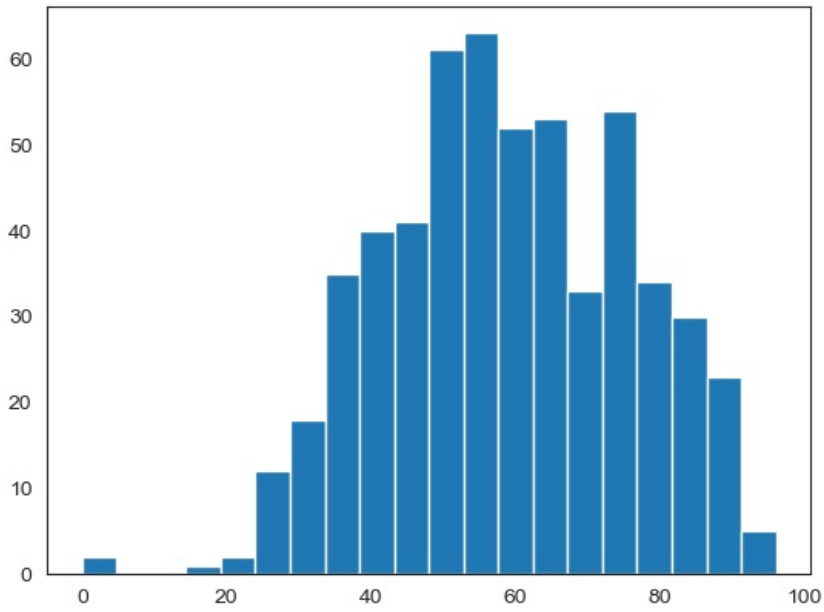
In [39]:
```
sns.set_style('whitegrid')  #change background
```

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



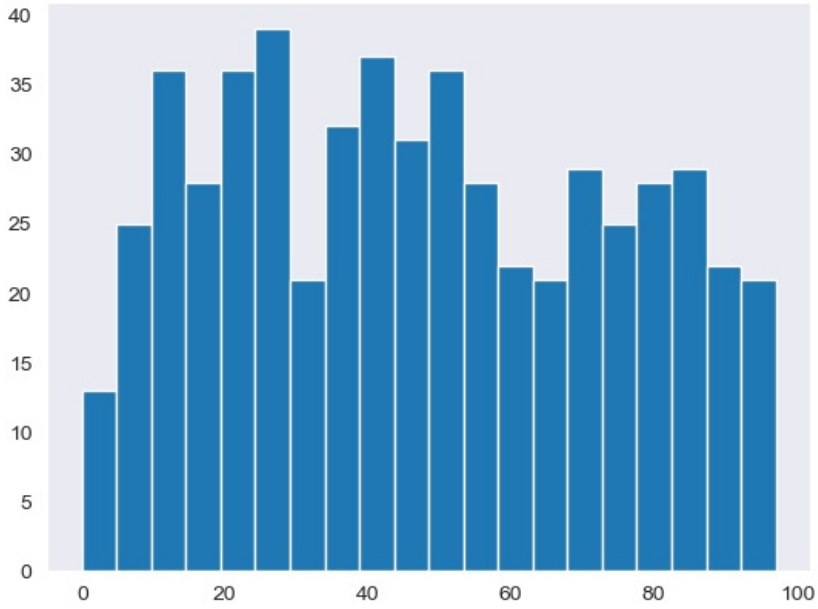
In [41]:
```
sns.set_style('white')  #change background
```

In [42]:
```
n1 = plt.hist(movies.AudienceRating, bins=20)
```



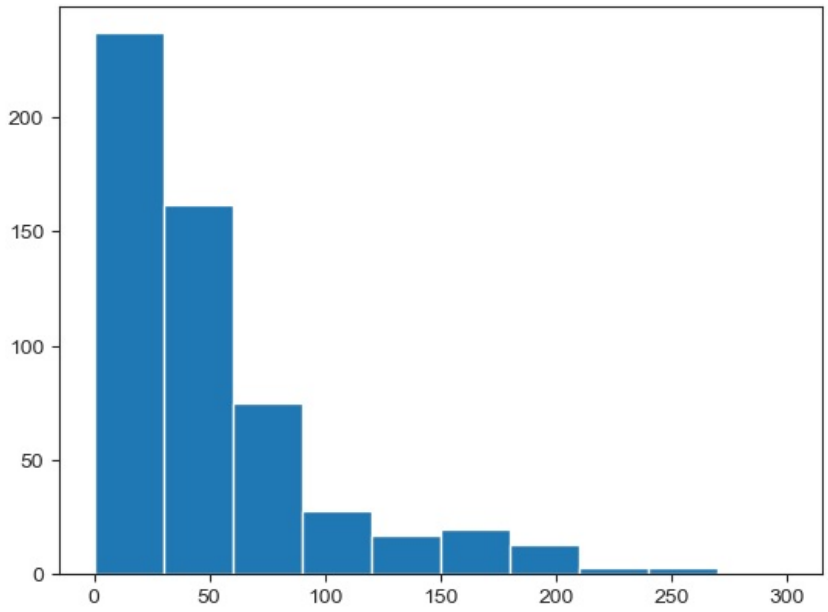
In [43]:
```
sns.set_style('dark')  #change background
```

In [44]:
```
n1 = plt.hist(movies.CriticRating, bins=20) #uniform distribution
```

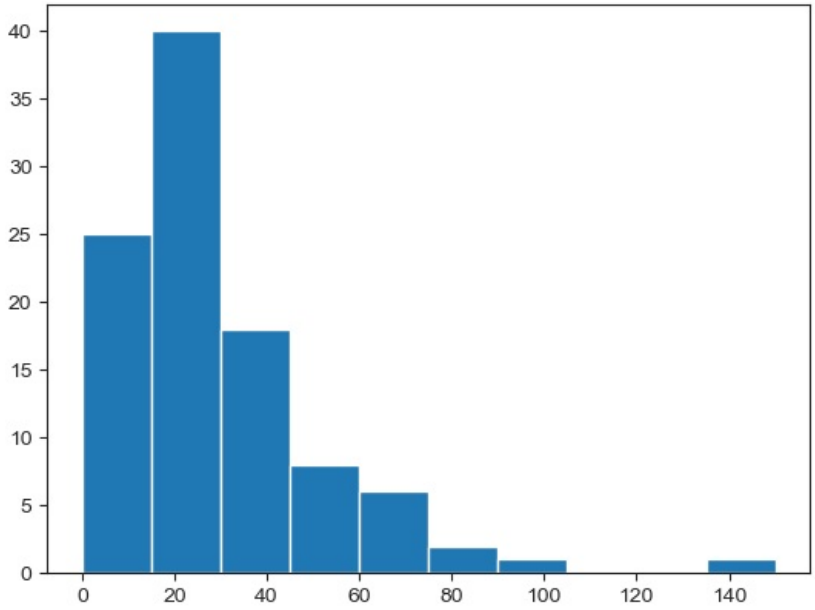


In [45]:
```
sns.set_style('ticks')  #change background
```

In [46]:
```
plt.hist(movies.BudgetMillions)
plt.show()
```



In [47]:
```
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions)
plt.show()
```



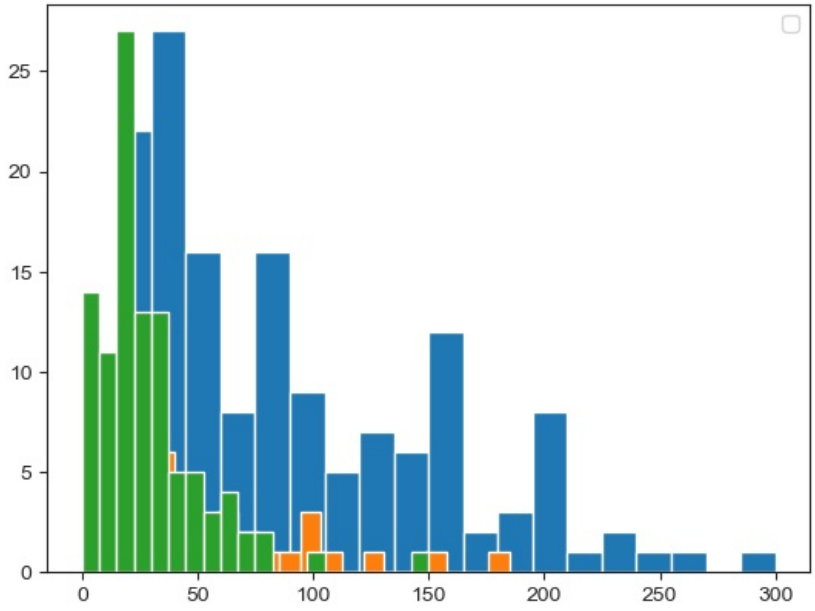
In [48]:
```
movies.head()
```

Out[48]:

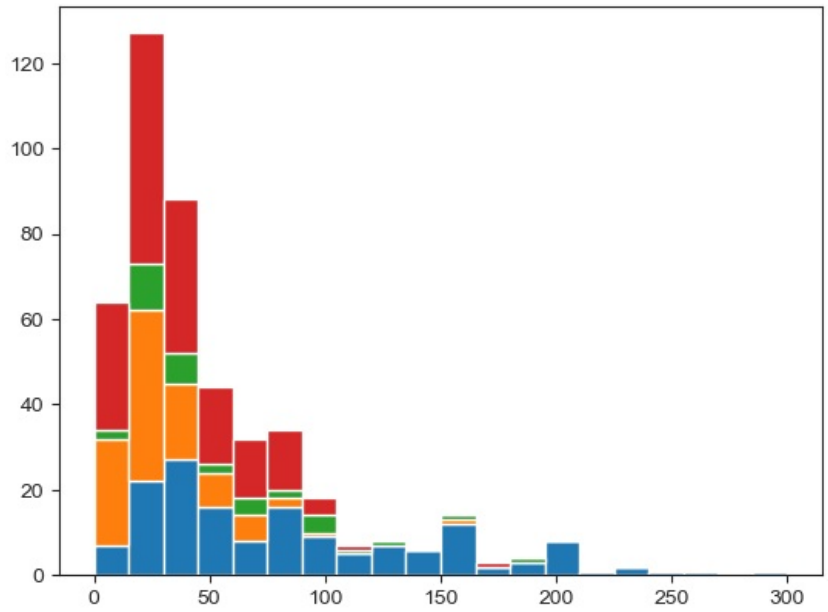
	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009

In [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()
```

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



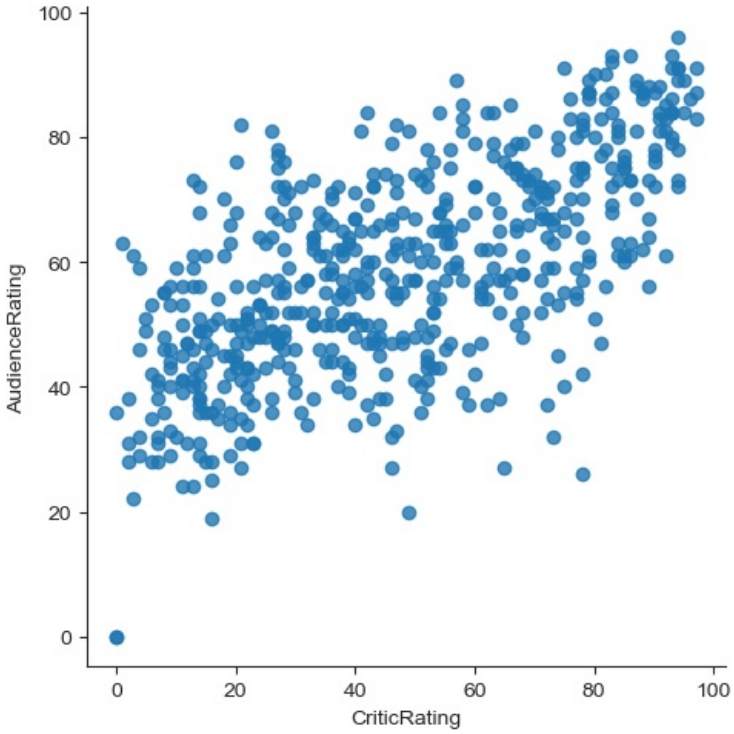
In [50]:
```
plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,\
          movies[movies.Genre == 'Drama'].BudgetMillions, \
          movies[movies.Genre == 'Thriller'].BudgetMillions, \
          movies[movies.Genre == 'Comedy'].BudgetMillions],
         bins = 20, stacked = True)
plt.show()
```



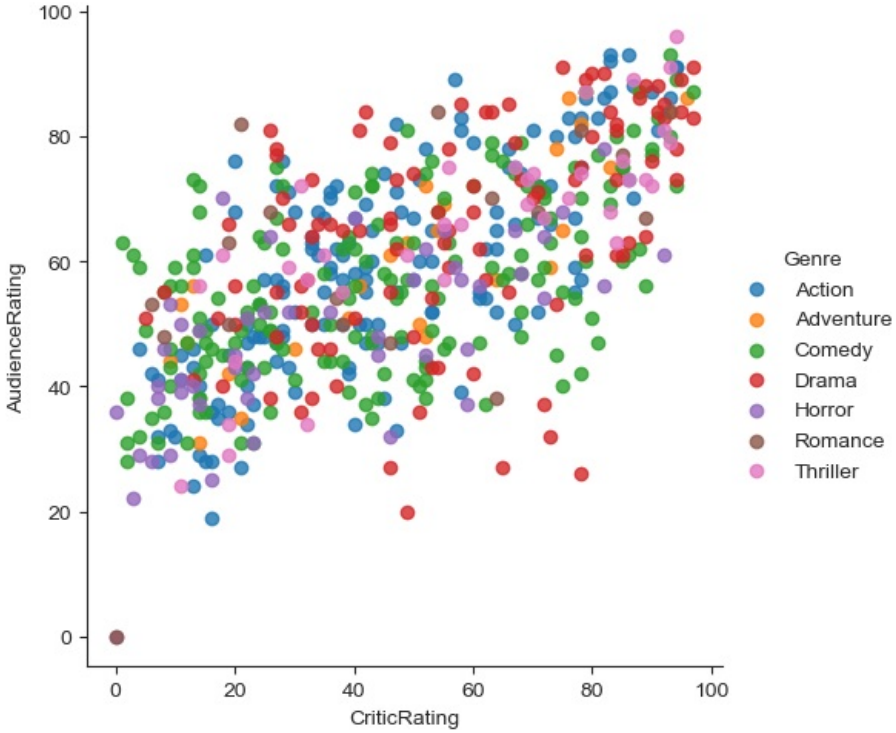
In [51]:
```
# if you have 100 categories you cannot copy & paste all the things

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

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

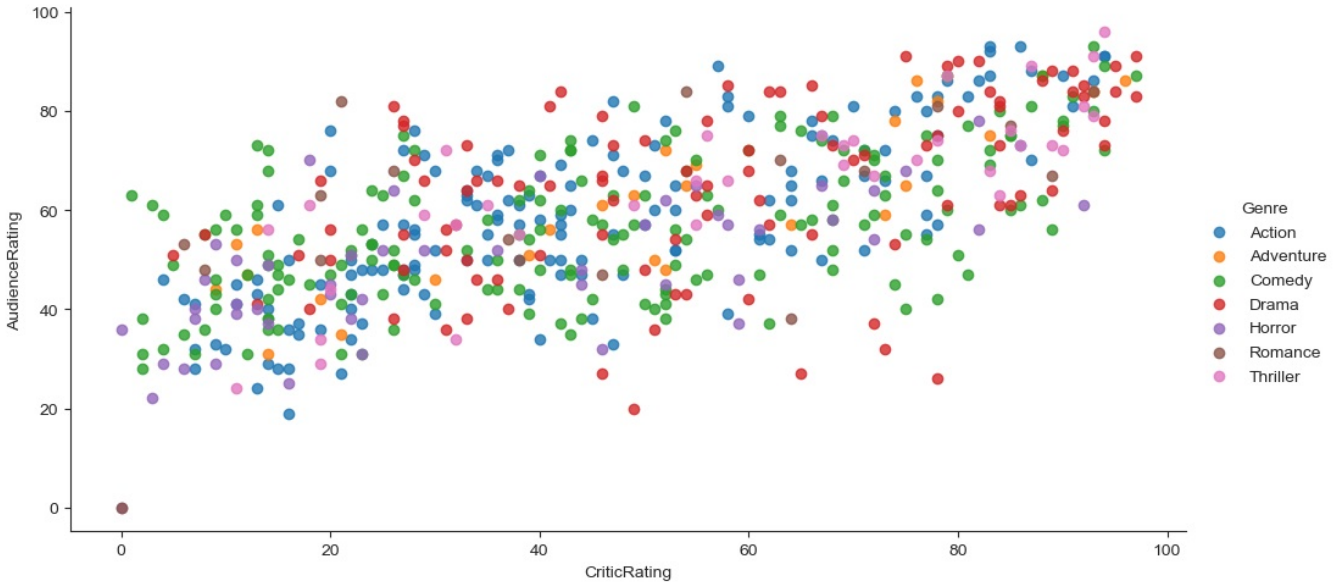
In [52]:
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False)
```



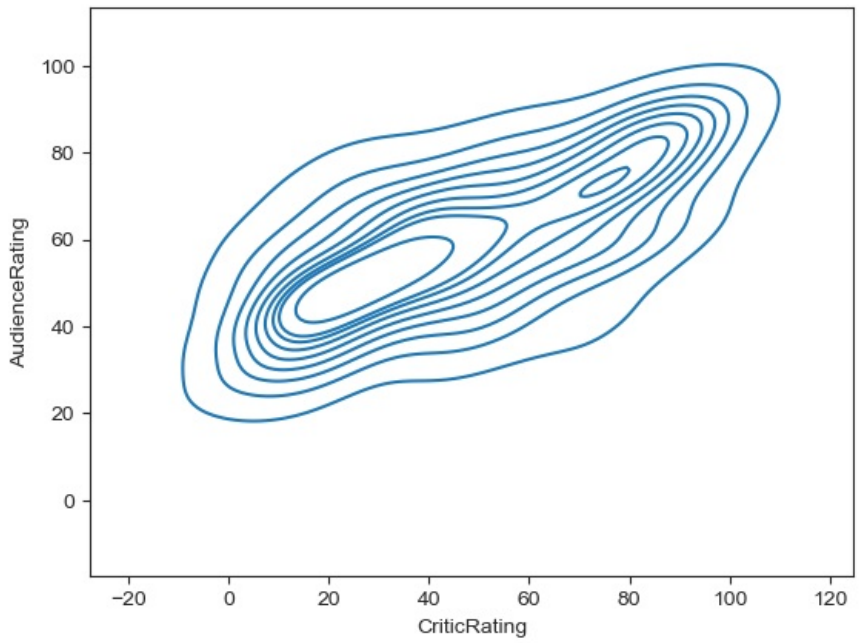
In [53]:
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False, hue = 'Genre')
```



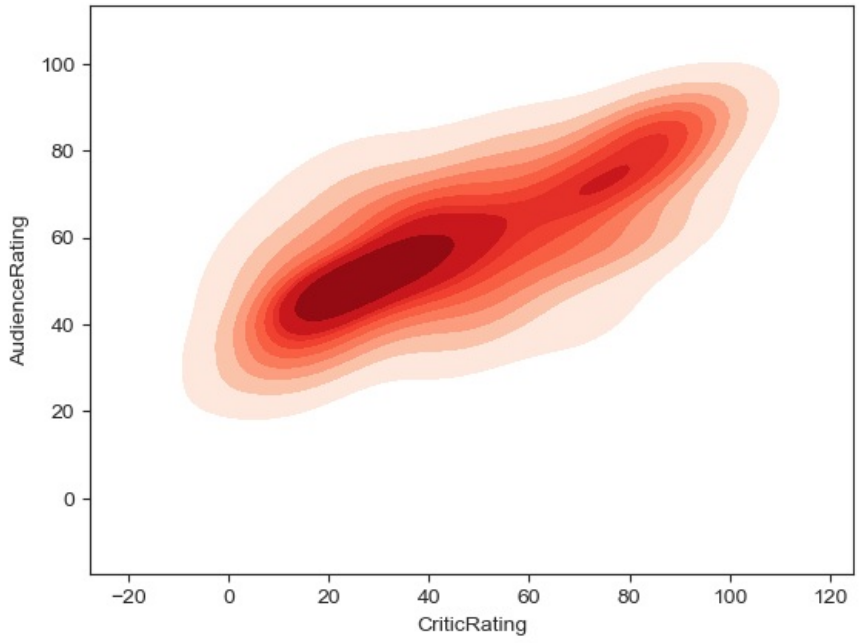
In [58]:
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False, hue = 'Genre',aspect=2)
```



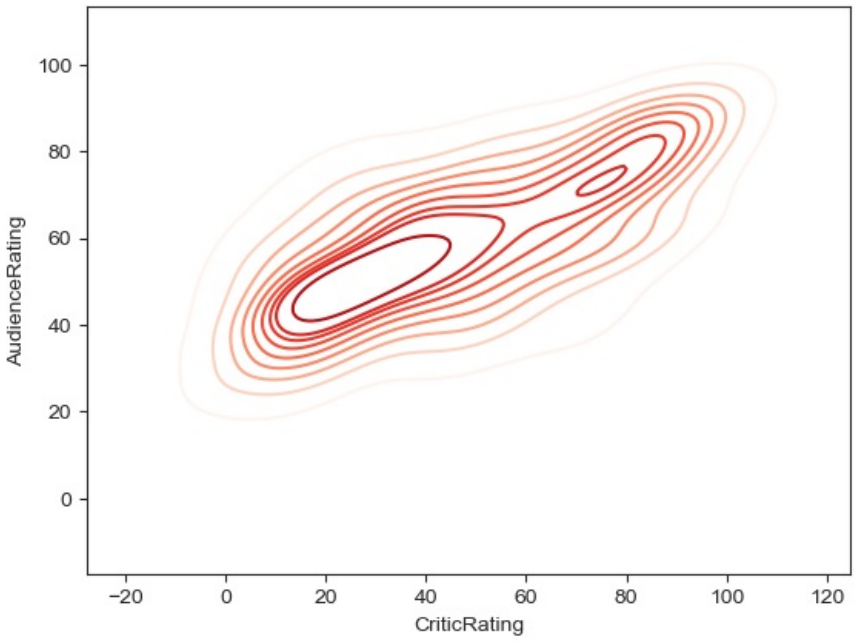
In [60]:
```
k1 = sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating')
```



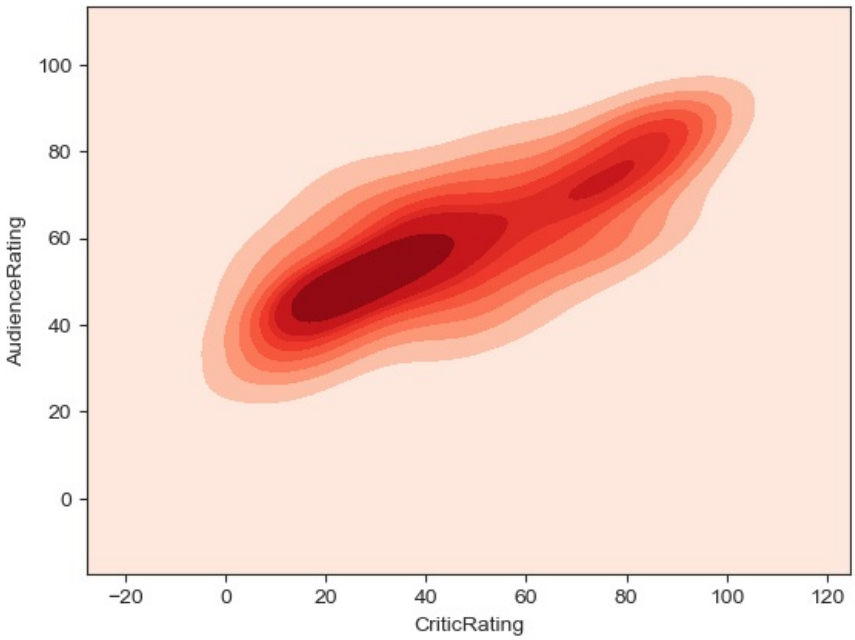
In [62]:
```
k1 = sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating',shade = True,shade_lowest=False,cmap='Reds')
```



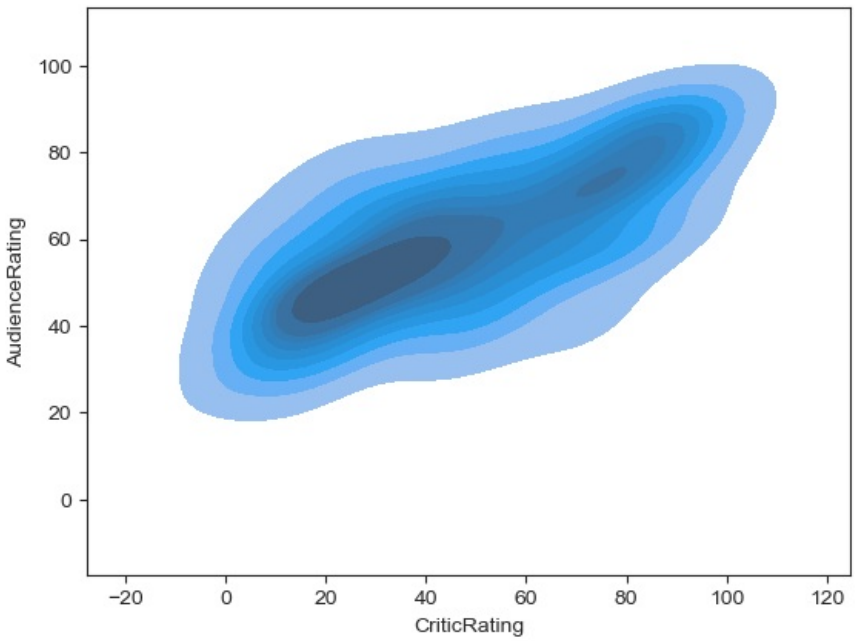
In [63]:
```
k1 = sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating',shade = False,shade_lowest=False,cmap='Reds')
```



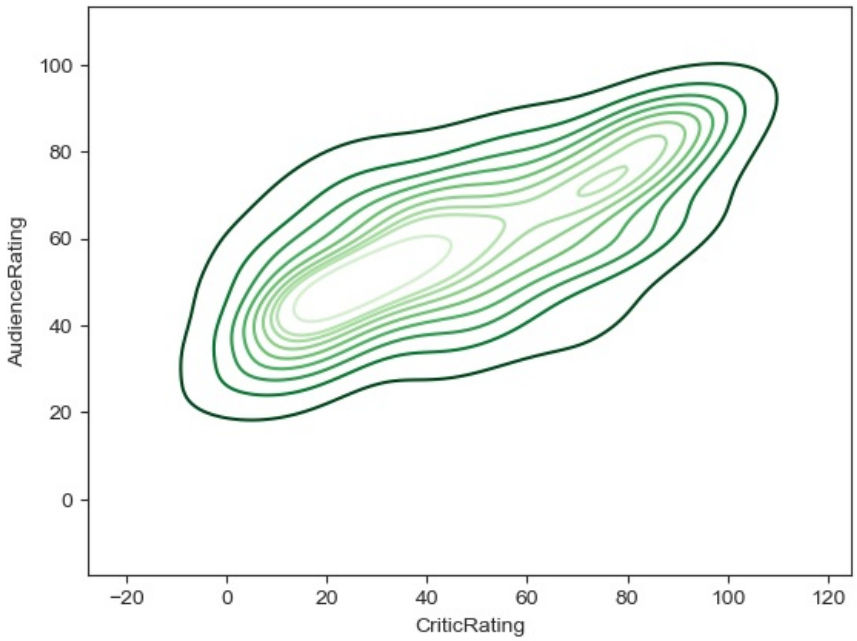
In [64]:
```
k1 = sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating',shade = True,shade_lowest=True,cmap='Reds')
```



In [67]:
```
k1 = sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating',shade = True,shade_lowest=False)
```

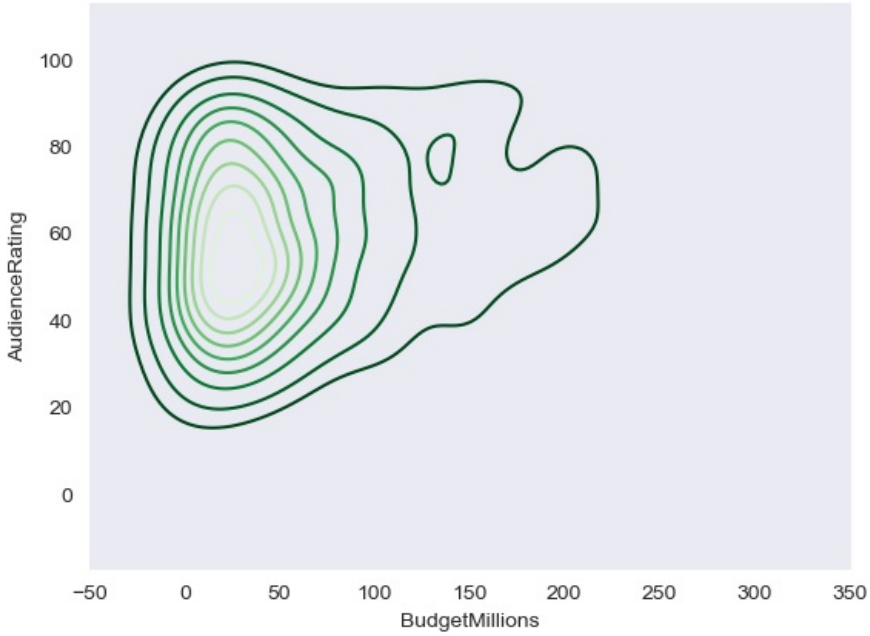


In [68]:
```
k1 = sns.kdeplot(data=movies,x='CriticRating',y='AudienceRating',shade = False,shade_lowest=False,cmap='Greens_
```

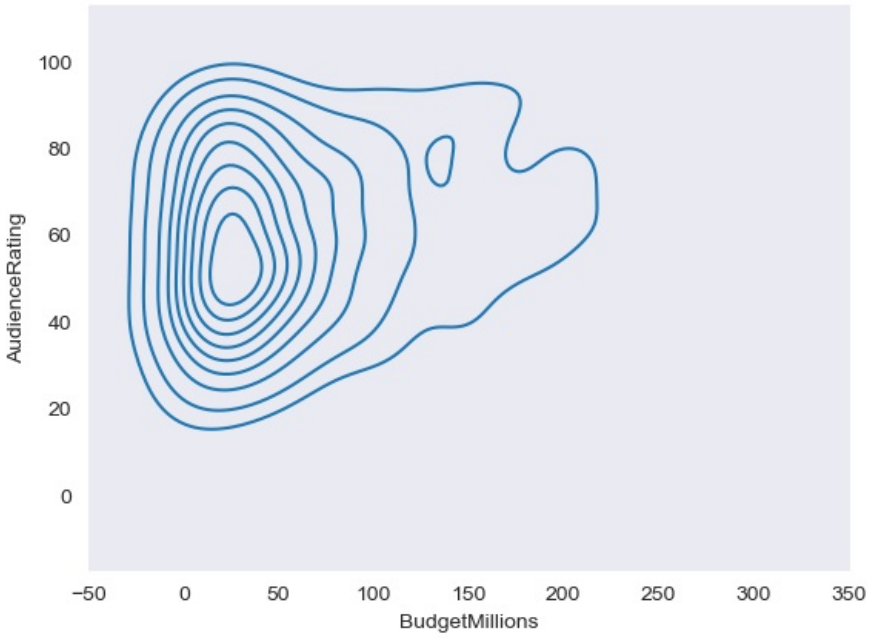


In [69]:
```
sns.set_style('dark')
```

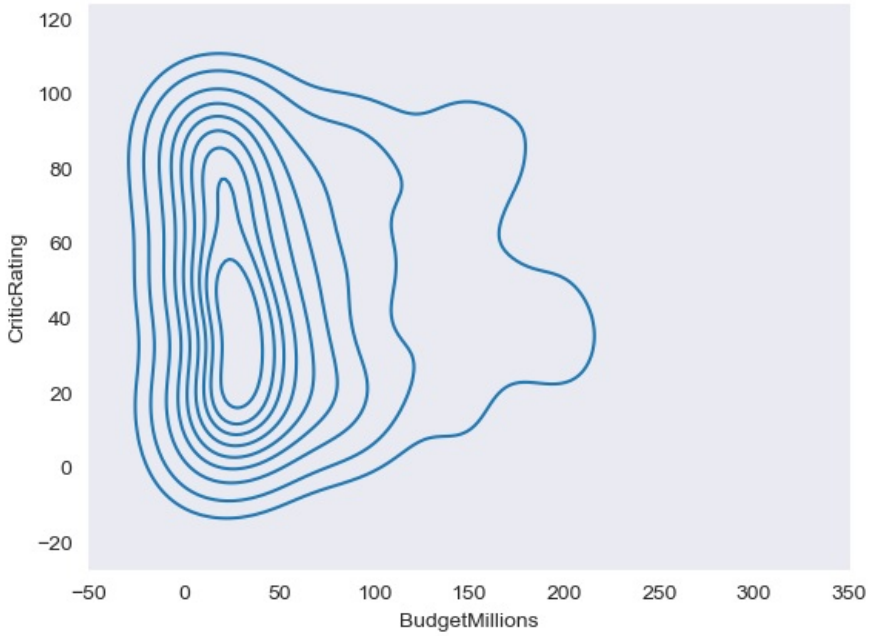
In [70]:
```
k1 = sns.kdeplot(data=movies,x='BudgetMillions',y='AudienceRating',shade_lowest=False,cmap='Greens_r')
```



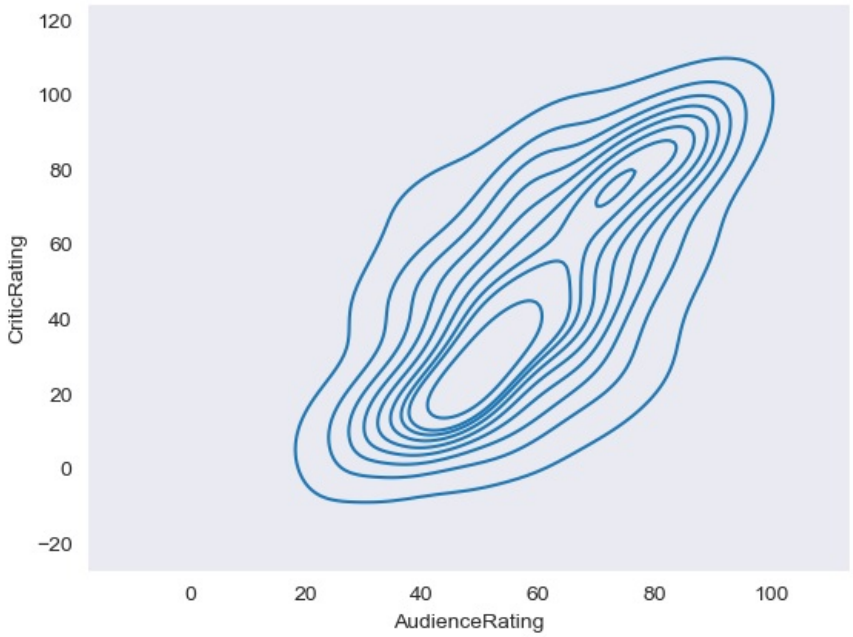
In [71]:
```
sns.set_style('dark')
k1 = sns.kdeplot(data=movies,x='BudgetMillions',y='AudienceRating')
```



In [74]:
```
k2 = sns.kdeplot(data=movies,x='BudgetMillions',y='CriticRating')
```

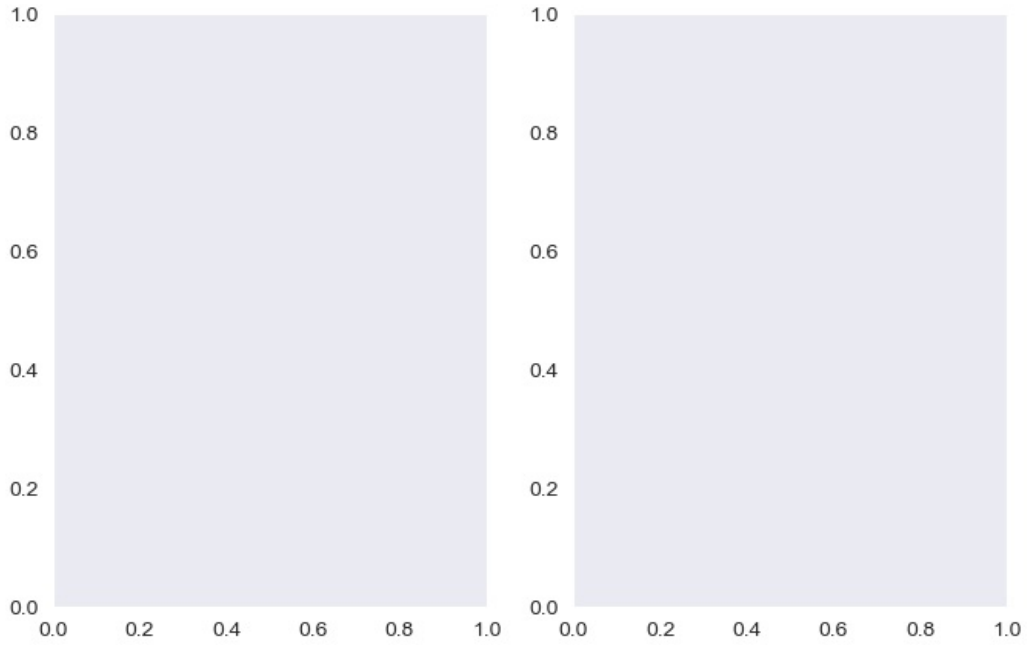


In [75]:
```
k3 = sns.kdeplot(data=movies,x='AudienceRating',y='CriticRating')
```

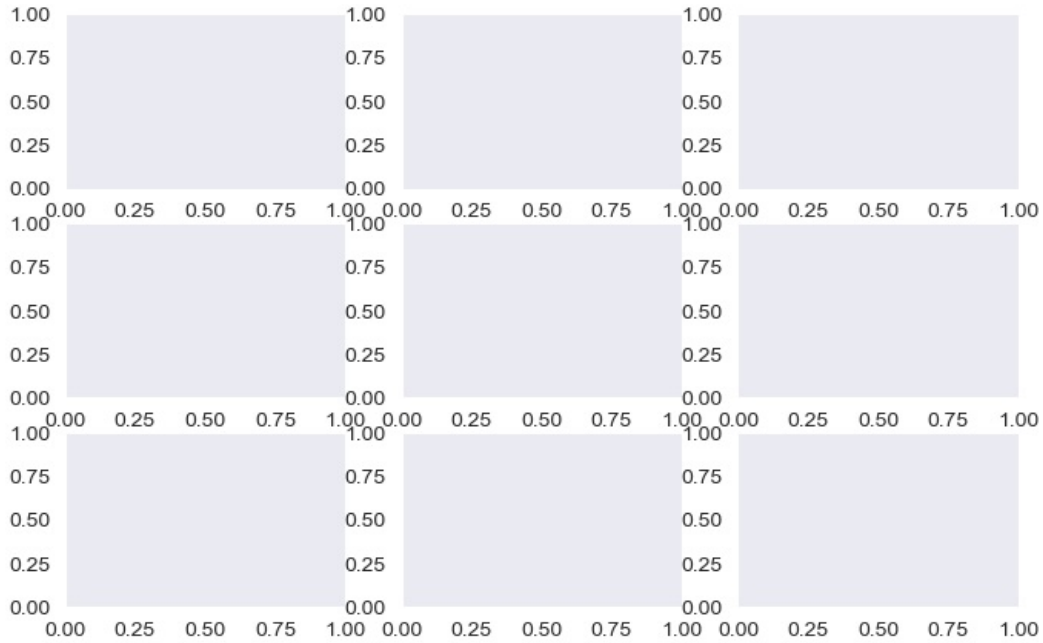


In [76]:
```
#subplots
```

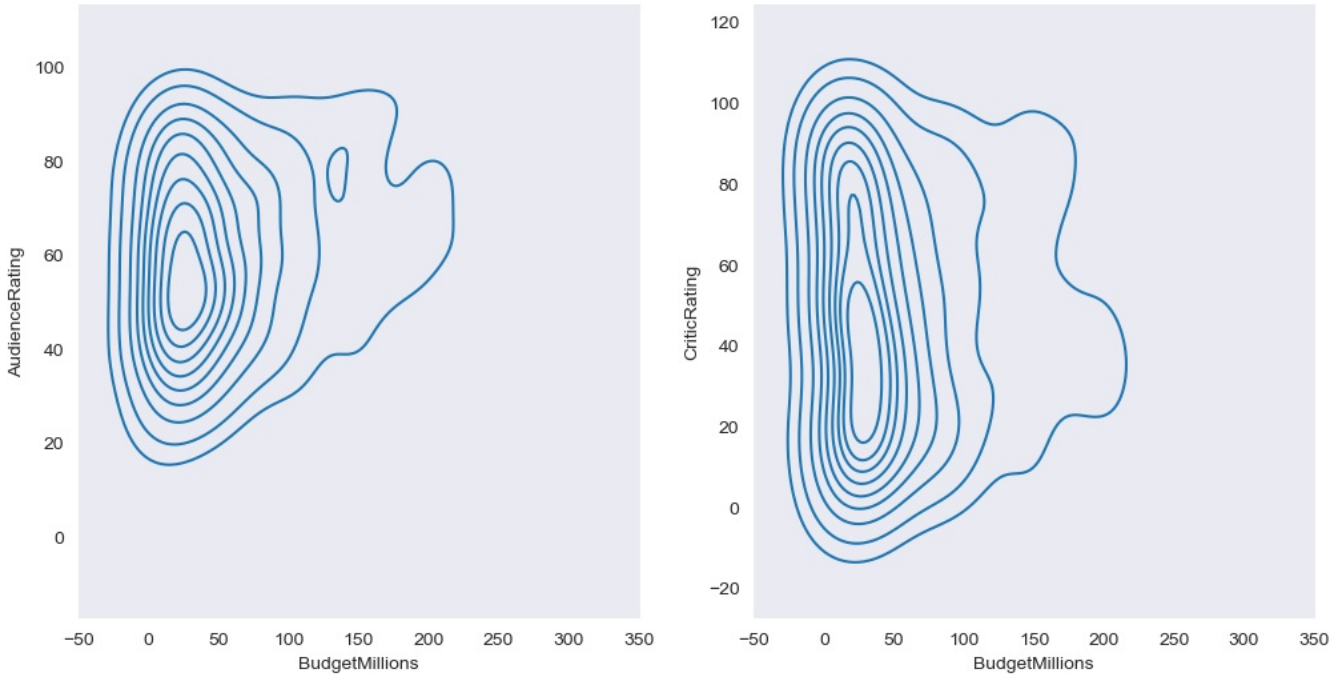
```
f, ax = plt.subplots(1,2, figsize =(8,5))
```



In [77]:
```
f, ax = plt.subplots(3,3, figsize =(8,5))
```



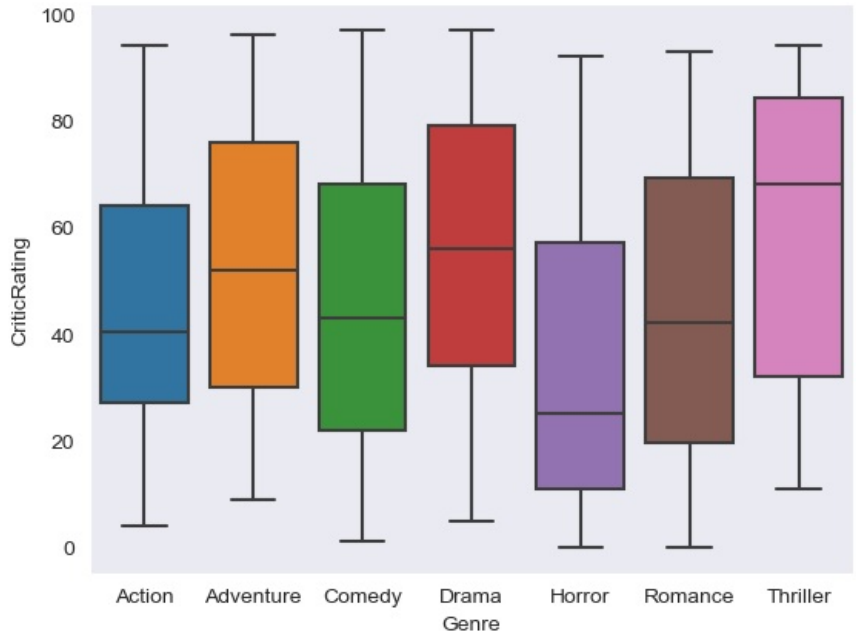
In [81]:
```
f, axes = plt.subplots(1,2, figsize =(12,6))

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



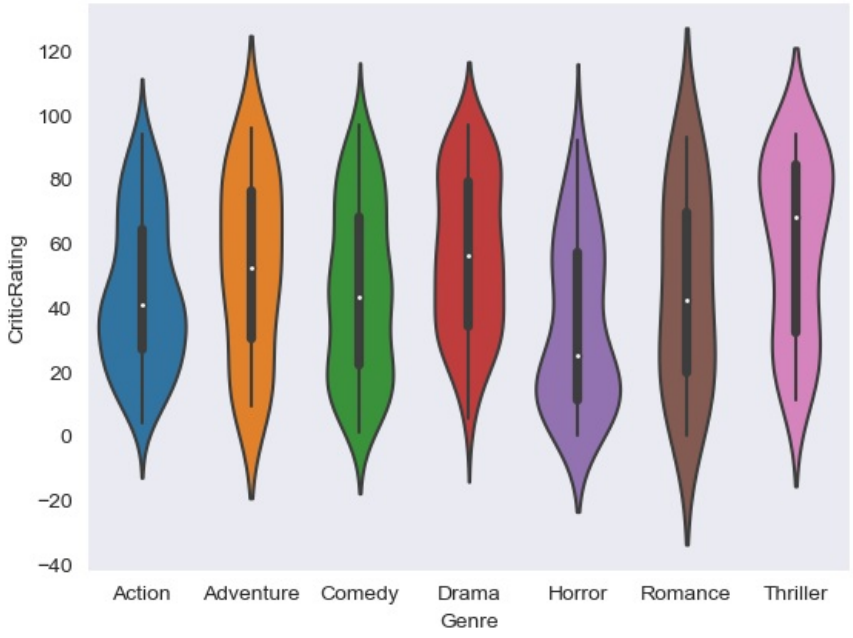
In [82]:
```
axes
```

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

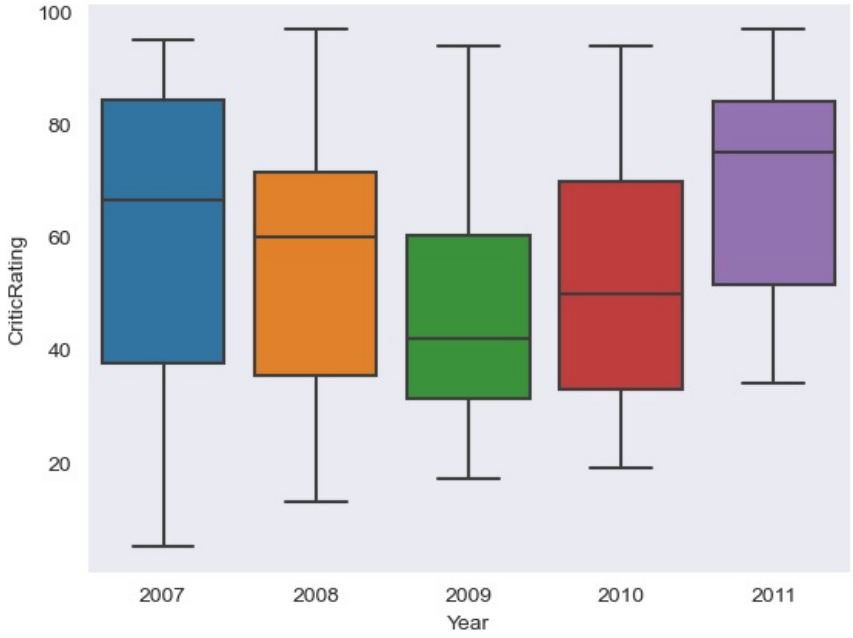
In [83]:
```
#Box plots -

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



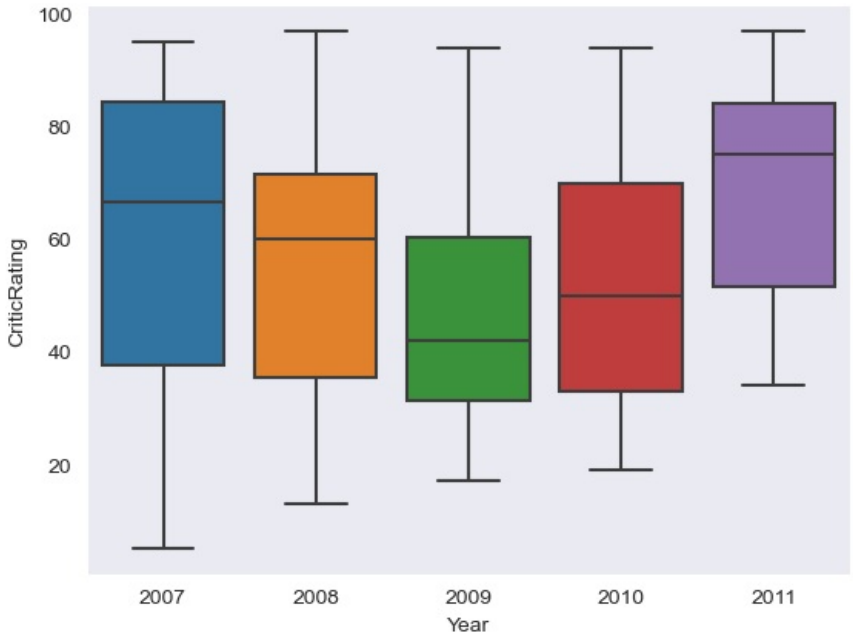
In [84]:
```
#Box plots -

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



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

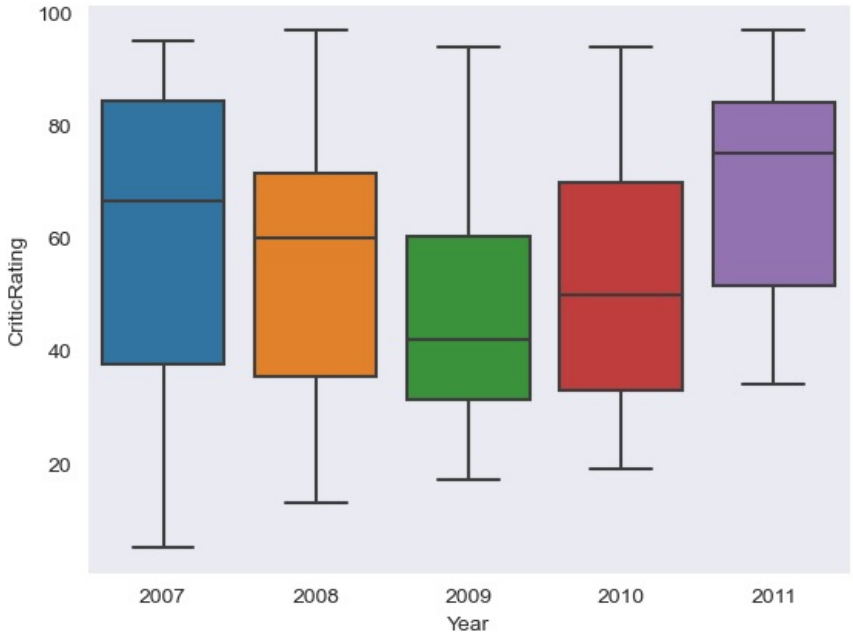


In [86]:
```
z = sns.boxplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRating')
```



In [87]:
```
sns.boxplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRating')
```

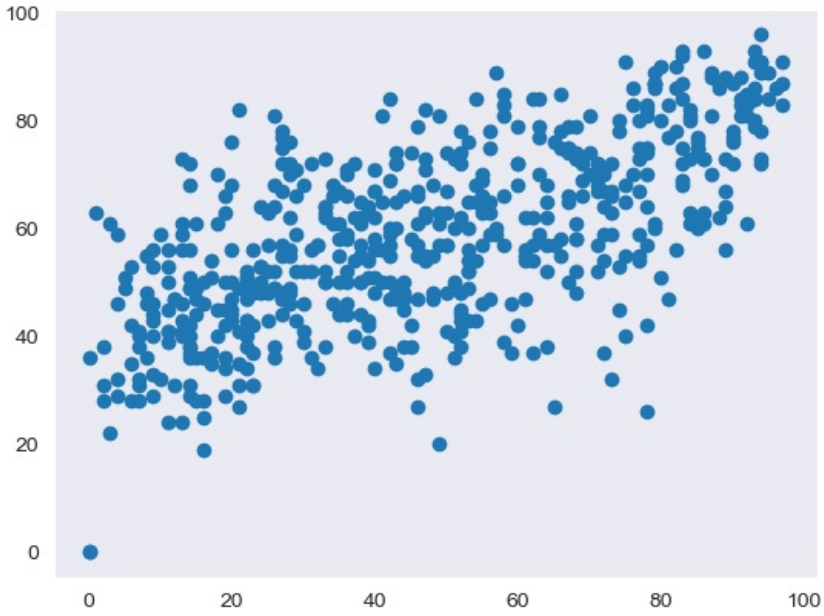
Out[87]: <Axes: xlabel='Year', ylabel='CriticRating'>



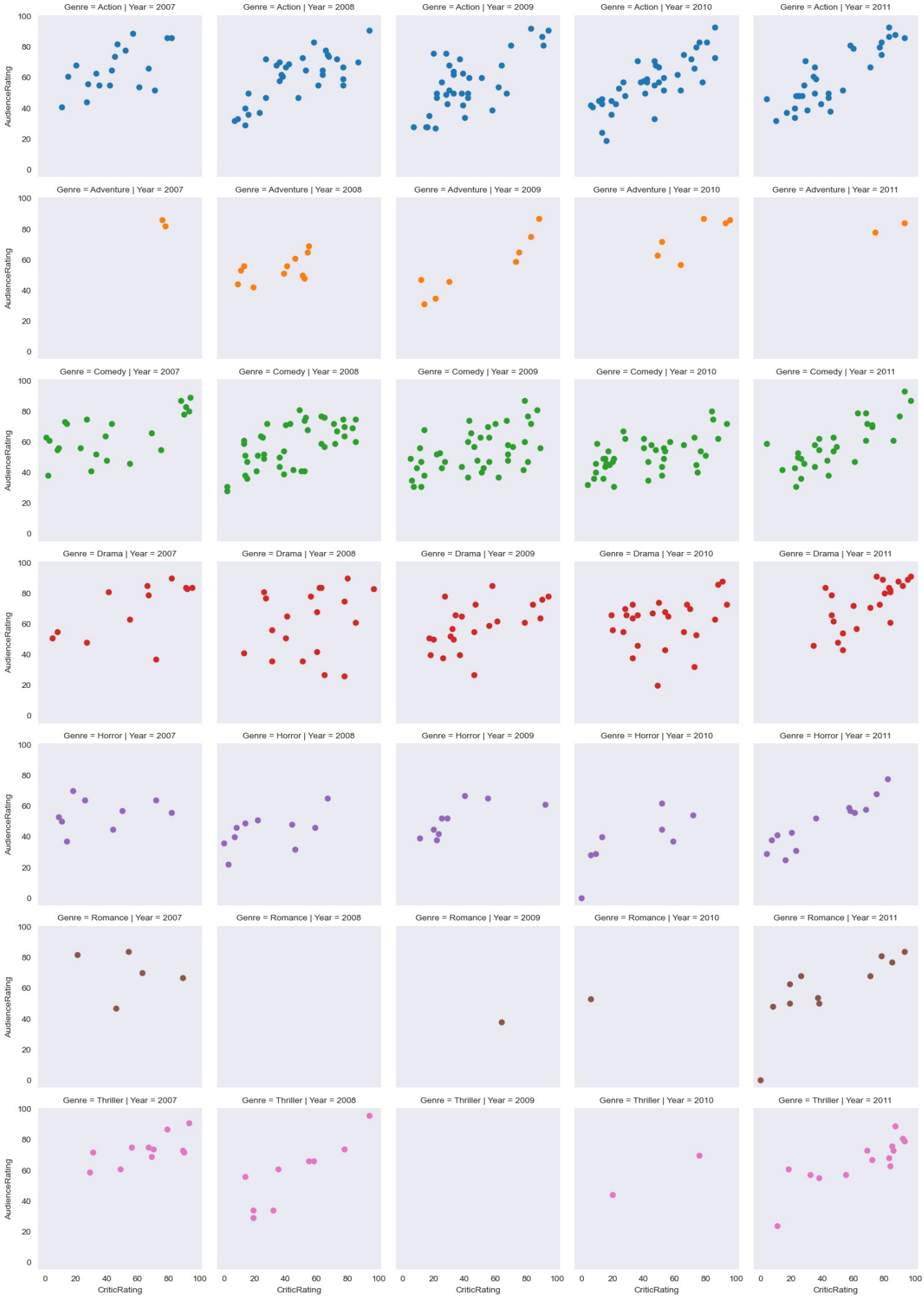
In [88]:
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre') #kind of subplots
```

In [90]:
```
plt.scatter(data=movies,x='CriticRating',y='AudienceRating')
```

Out[90]:
```
<matplotlib.collections.PathCollection at 0x1751cbf5150>
```



In [91]:
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating' ) #scatterplots are mapped in facetgrid
```

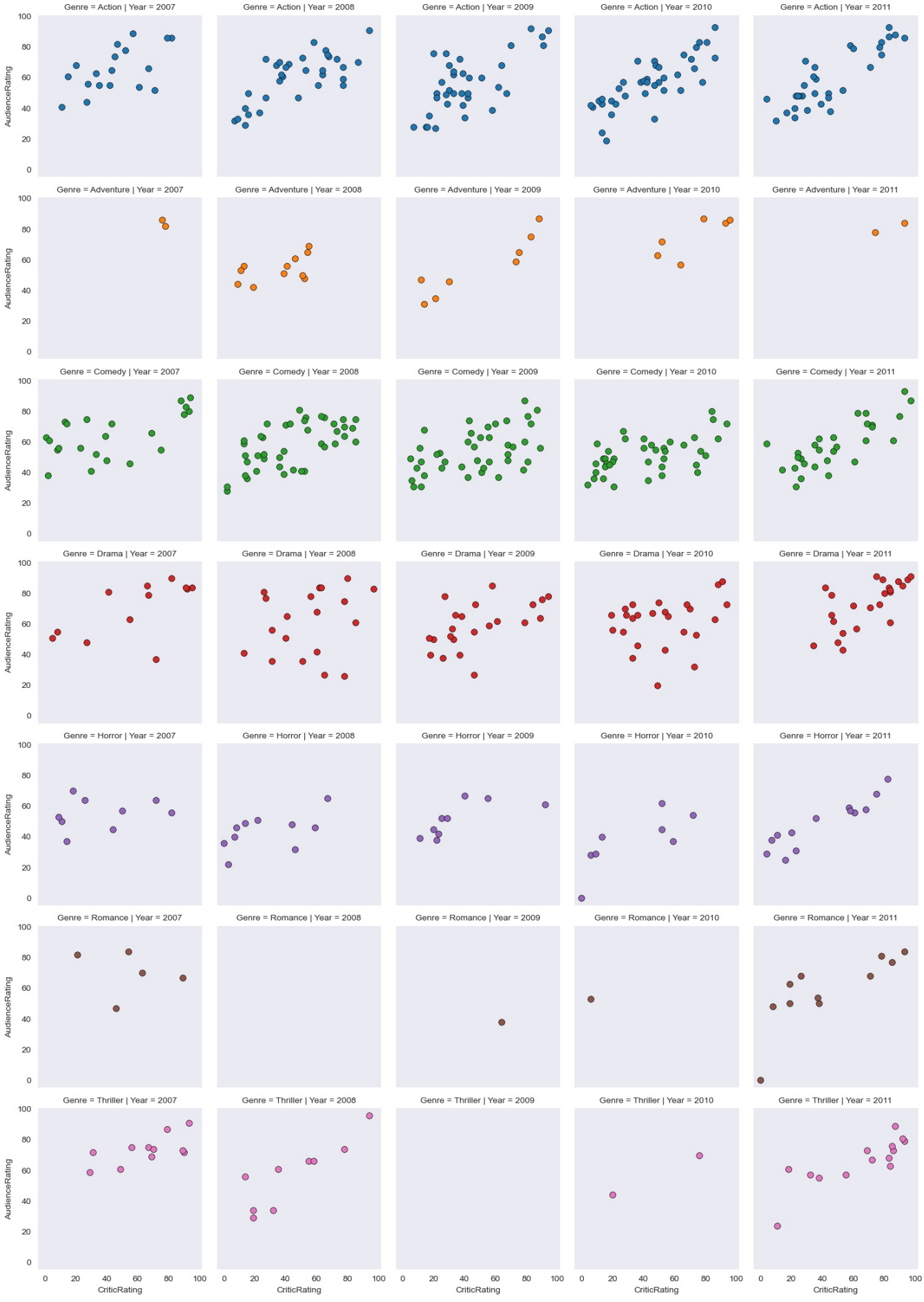


In [92]:

```
# 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 [94]:

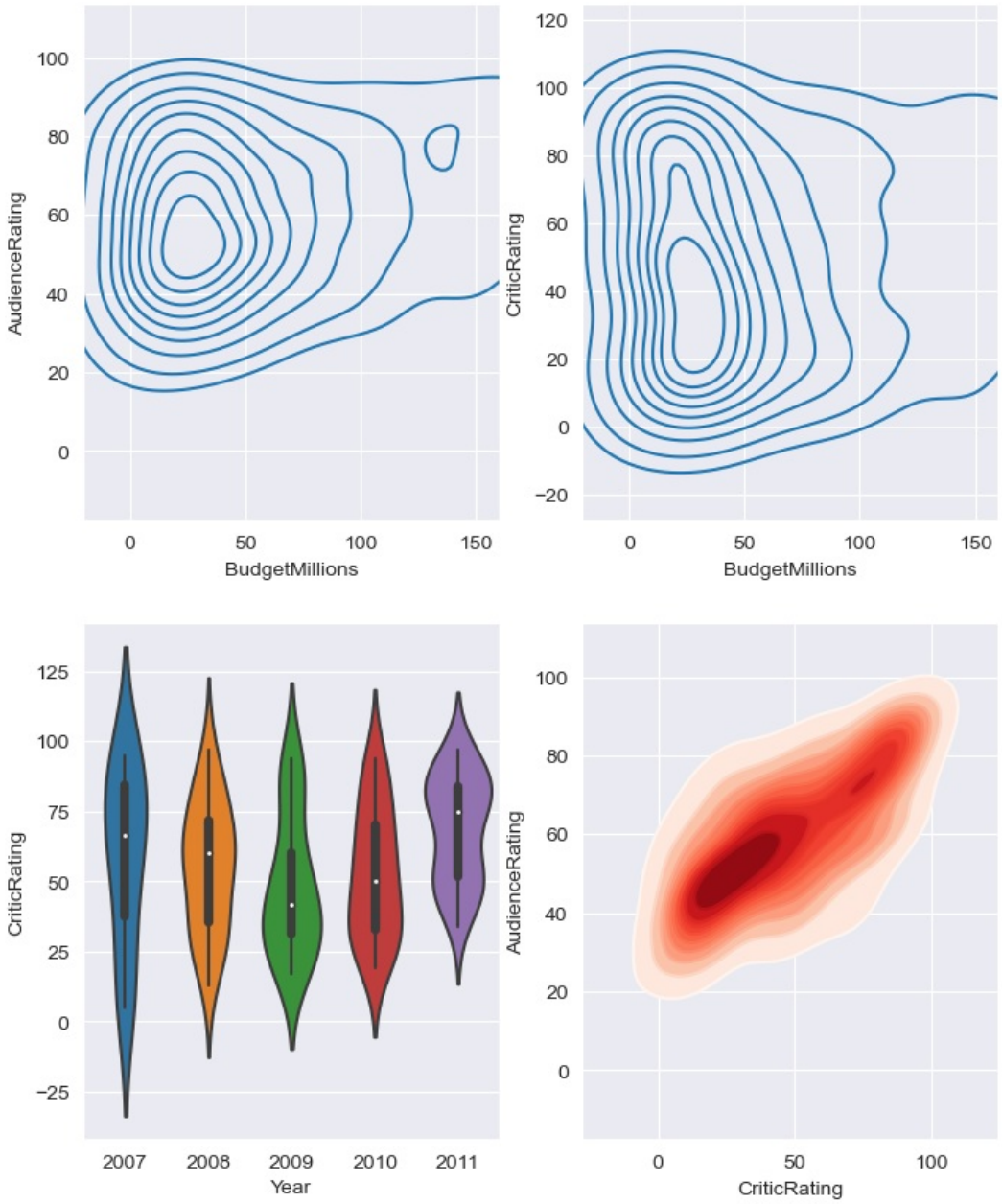
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
kws = dict(s=50, linewidth=0.5,edgecolor='black')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating',**kws ) #scatterplots are mapped in facetgrid
```



In [100…

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

sns.set_style('darkgrid')
f, axes = plt.subplots (2,2, figsize = (8,10))

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

```
z = 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,cmap='Reds',ax

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

plt.show()
```



In [ ]:

In [ ]: