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

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

In [123…
```
movies
```

Out[123…

	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 [4]:
```
type(movies)
```

Out[4]:
```
pandas.core.frame.DataFrame
```

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

Out[5]:
```
559
```

In [6]:
```
import numpy
print(numpy.__version__)
```

```
2.1.3
```

In [7]:
```
import pandas
print(pandas.__version__)
```

```
2.2.3
```

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

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

In [9]:
```
movies.info() # info - information of dataframe
```

```
<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 [10]:
```
movies.shape # no of row & no of columns
```

Out[10]:
```
(559, 6)
```

In [11]:
```
movies.head() # top five rows
```

Out[11]:

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million $)	Year of release
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009

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

Out[12]:

	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 [13]:
```
movies.columns
```

Out[13]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

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

In [15]:
```
movies.head(1)
```

Out[15]:

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

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

Out[16]:
```
(559, 6)
```

In [17]:
```
movies.describe() # describe statistics
```

Out[17]:

	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 [18]:
```
movies.describe().transpose()
```

Out[18]:

	count	mean	std	min	25%	50%	75%	max
CriticRating	559.0	47.309481	26.413091	0.0	25.0	46.0	70.0	97.0
AudienceRating	559.0	58.744186	16.826887	0.0	47.0	58.0	72.0	96.0
BudgetMillions	559.0	50.236136	48.731817	0.0	20.0	35.0	65.0	300.0
Year	559.0	2009.152057	1.362632	2007.0	2008.0	2009.0	2010.0	2011.0

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

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

Out[23]:

	CriticRating	AudienceRating	BudgetMillions	Year
count	559.000000	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136	2009.152057
std	26.413091	16.826887	48.731817	1.362632
min	0.000000	0.000000	0.000000	2007.000000
25%	25.000000	47.000000	20.000000	2008.000000
50%	46.000000	58.000000	35.000000	2009.000000
75%	70.000000	72.000000	65.000000	2010.000000
max	97.000000	96.000000	300.000000	2011.000000

In [24]:
```
movies['Genre'] = movies['Genre'].astype('category')
movies['Year'] = movies['Year'].astype('category')
```

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

Out[41]:
```
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 [42]:
```
movies.info
```

Out[42]:

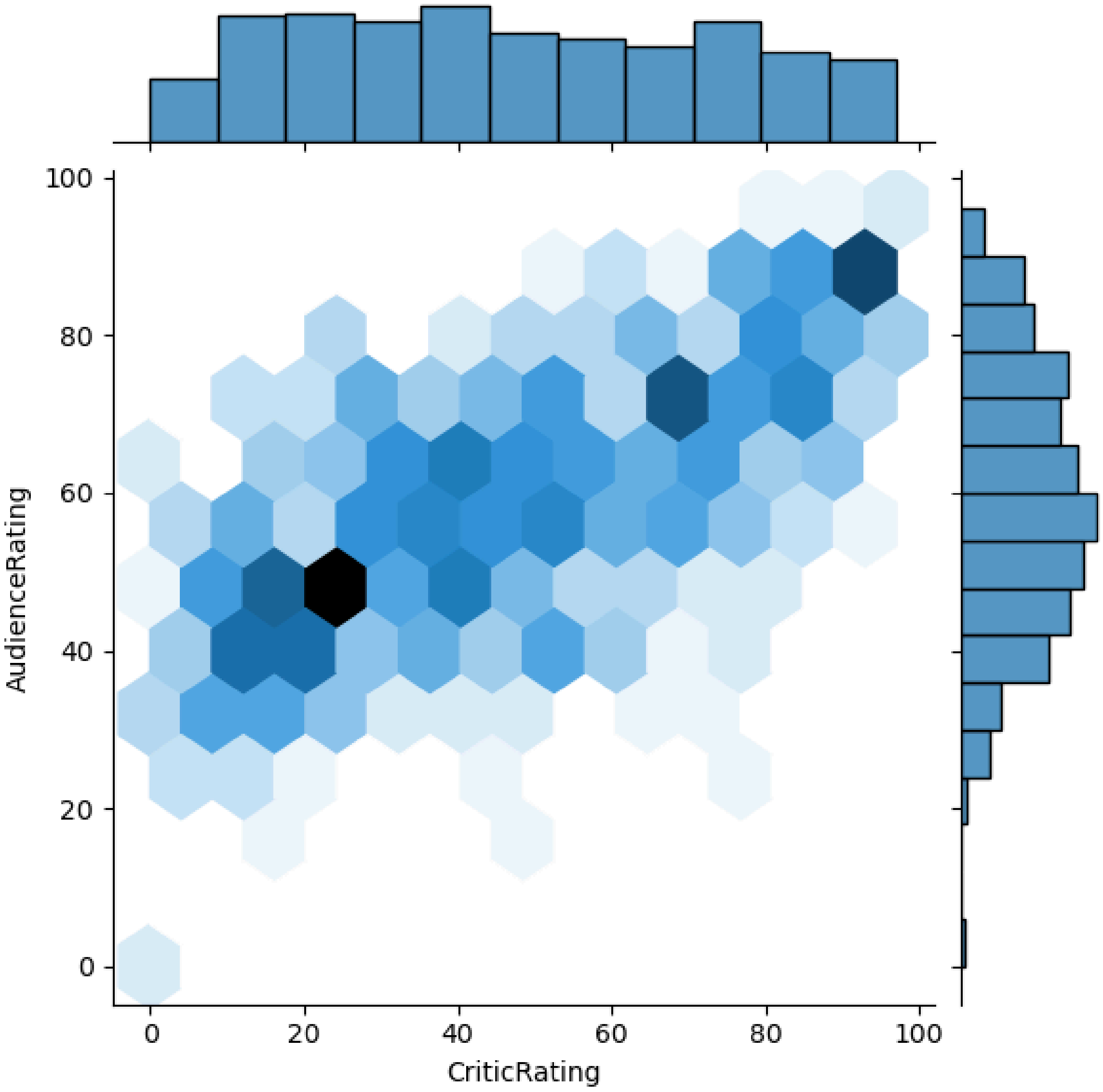
```
<bound method DataFrame.info of                       Film      Genre  Rotten T
omatoes Ratings %  \
0    (500) Days of Summer      Comedy                         87
1              10,000 B.C.  Adventure                          9
2               12 Rounds      Action                         30
3                127 Hours  Adventure                         93
4                17 Again      Comedy                         55
..                     ...        ...                        ...
554          Your Highness     Comedy                         26
555        Youth in Revolt     Comedy                         68
556                 Zodiac   Thriller                         89
557            Zombieland      Action                         90
558              Zookeeper     Comedy                         14

     Audience Ratings %  Budget (million $)  Year of release
0                    81                   8             2009
1                    44                 105             2008
2                    52                  20             2009
3                    84                  18             2010
4                    70                  20             2009
..                  ...                 ...              ...
554                  36                  50             2011
555                  52                  18             2009
556                  73                  65             2007
557                  87                  24             2009
558                  42                  80             2011

[559 rows x 6 columns]>
```

In [26]:

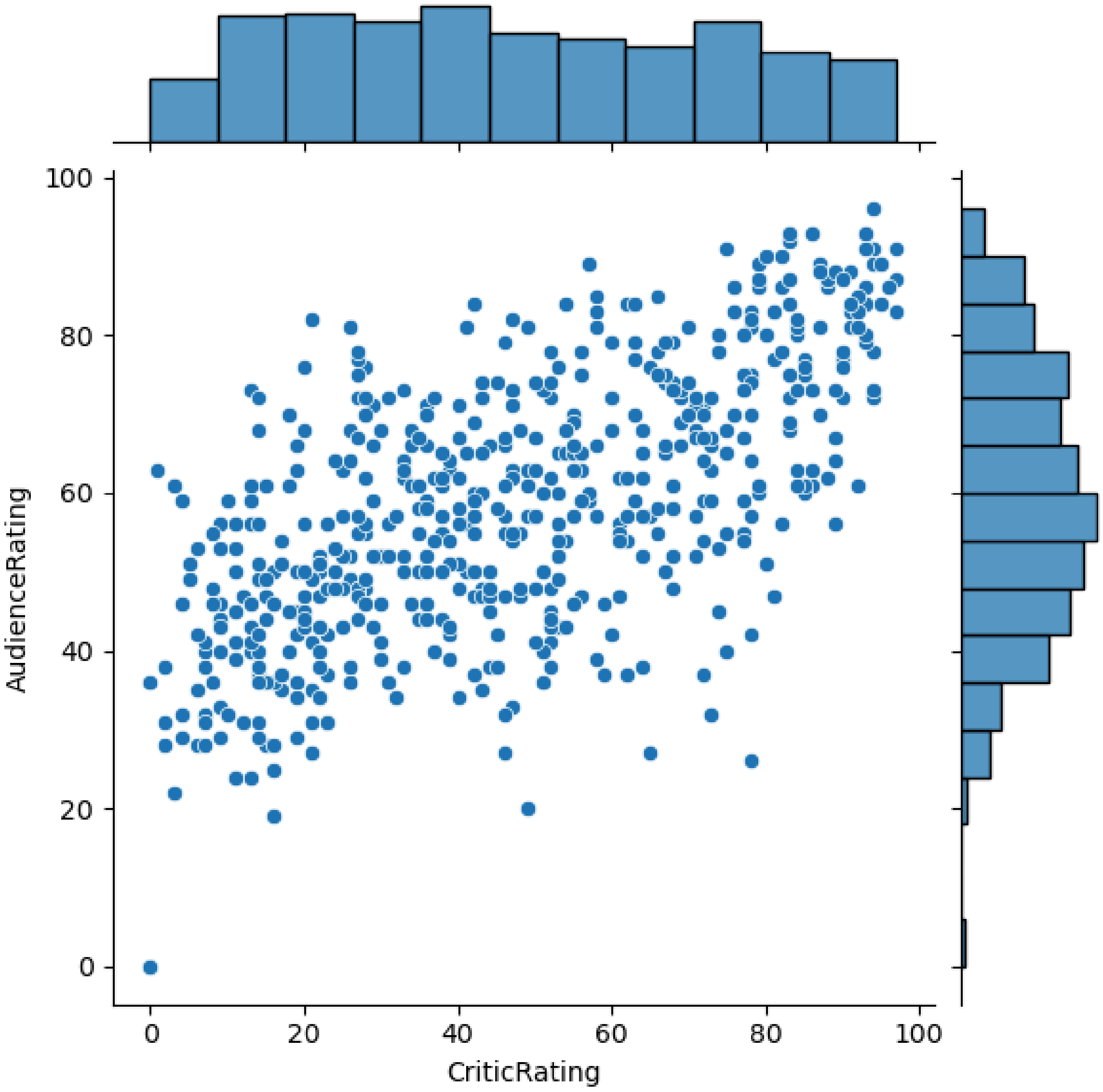
```python
from matplotlib import pyplot as plt
import seaborn as sns


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

In [27]:

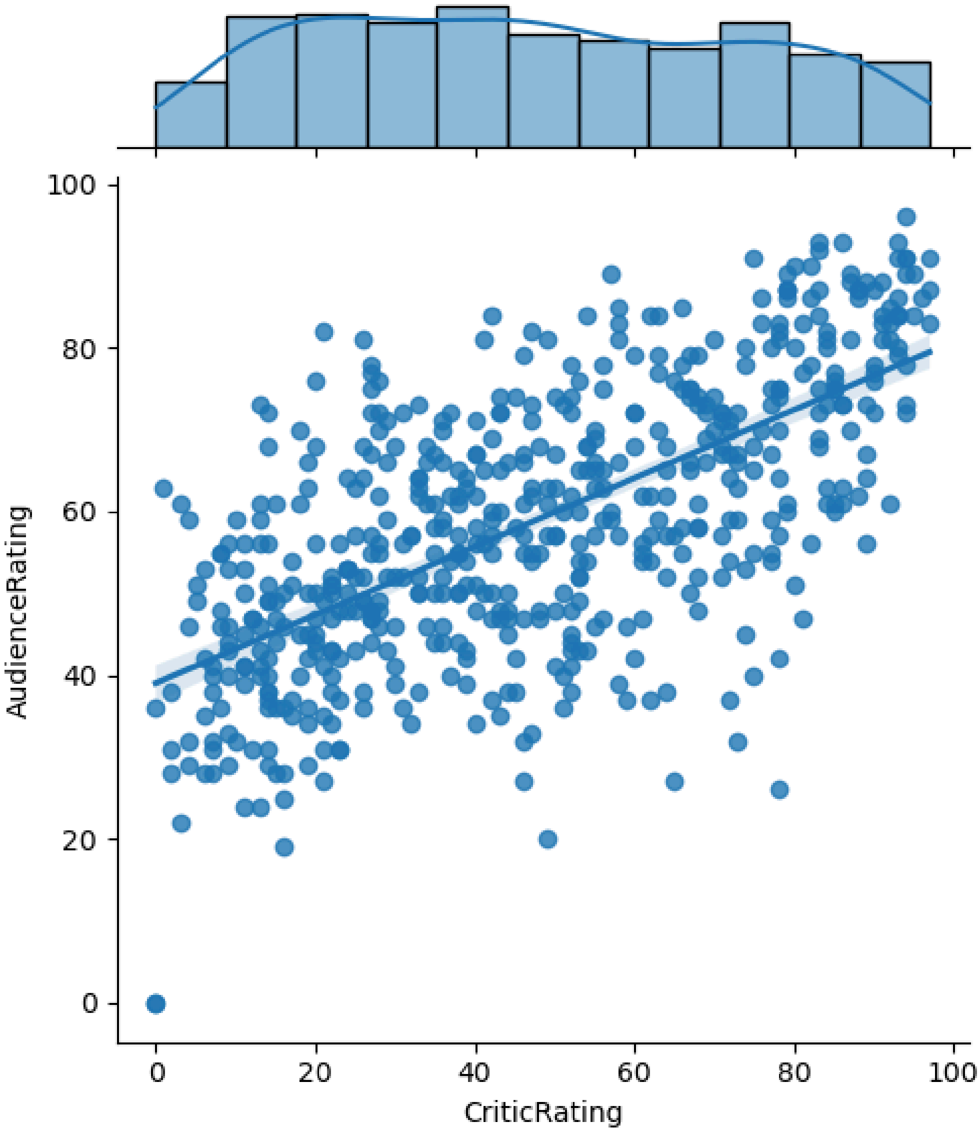
```python
j = sns.jointplot(data = movies, x ='CriticRating', y = 'AudienceRating', kind='
```



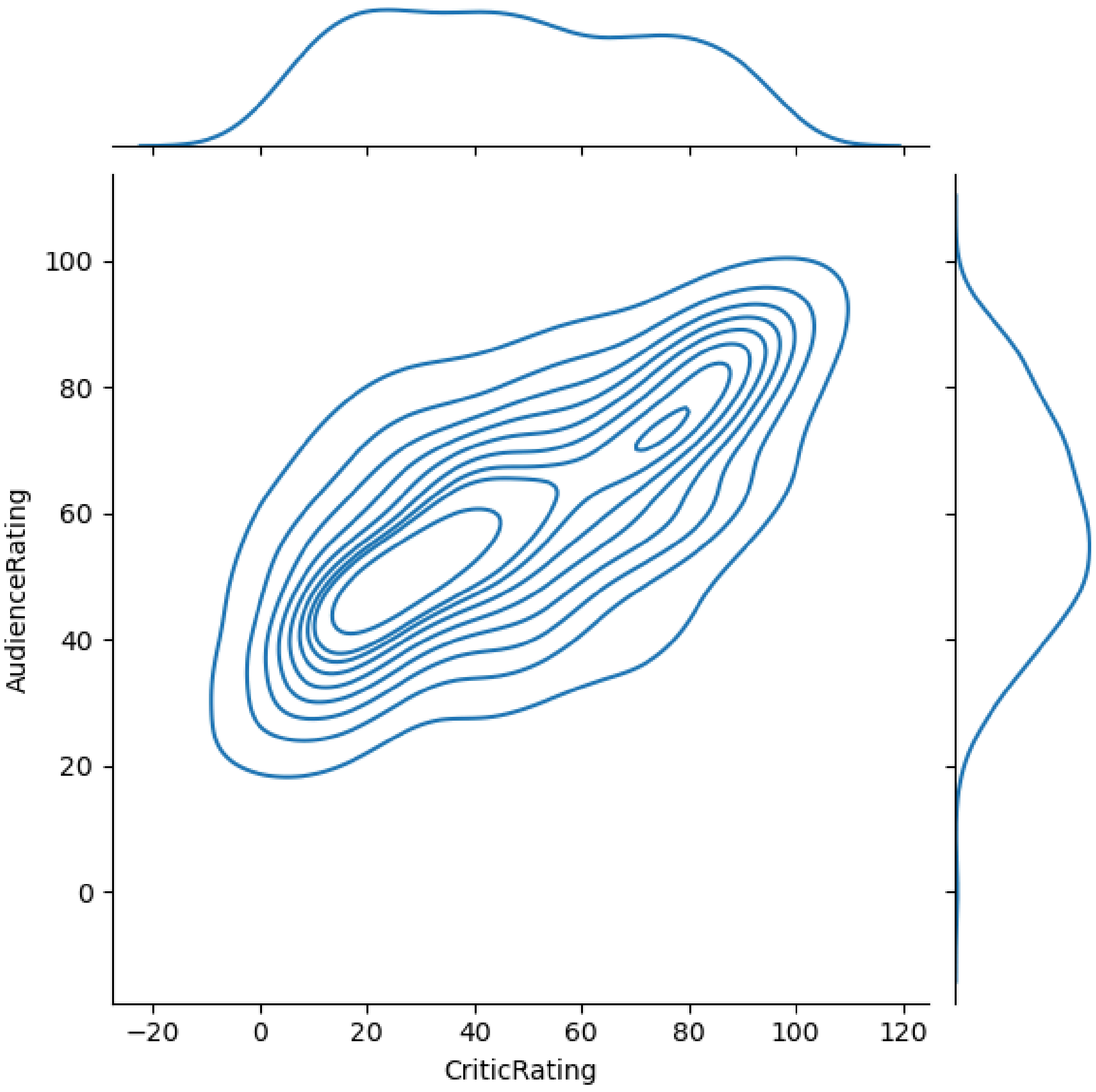
```
In [28]: j = sns.jointplot(data = movies, x ='CriticRating', y = 'AudienceRating', kind='
```



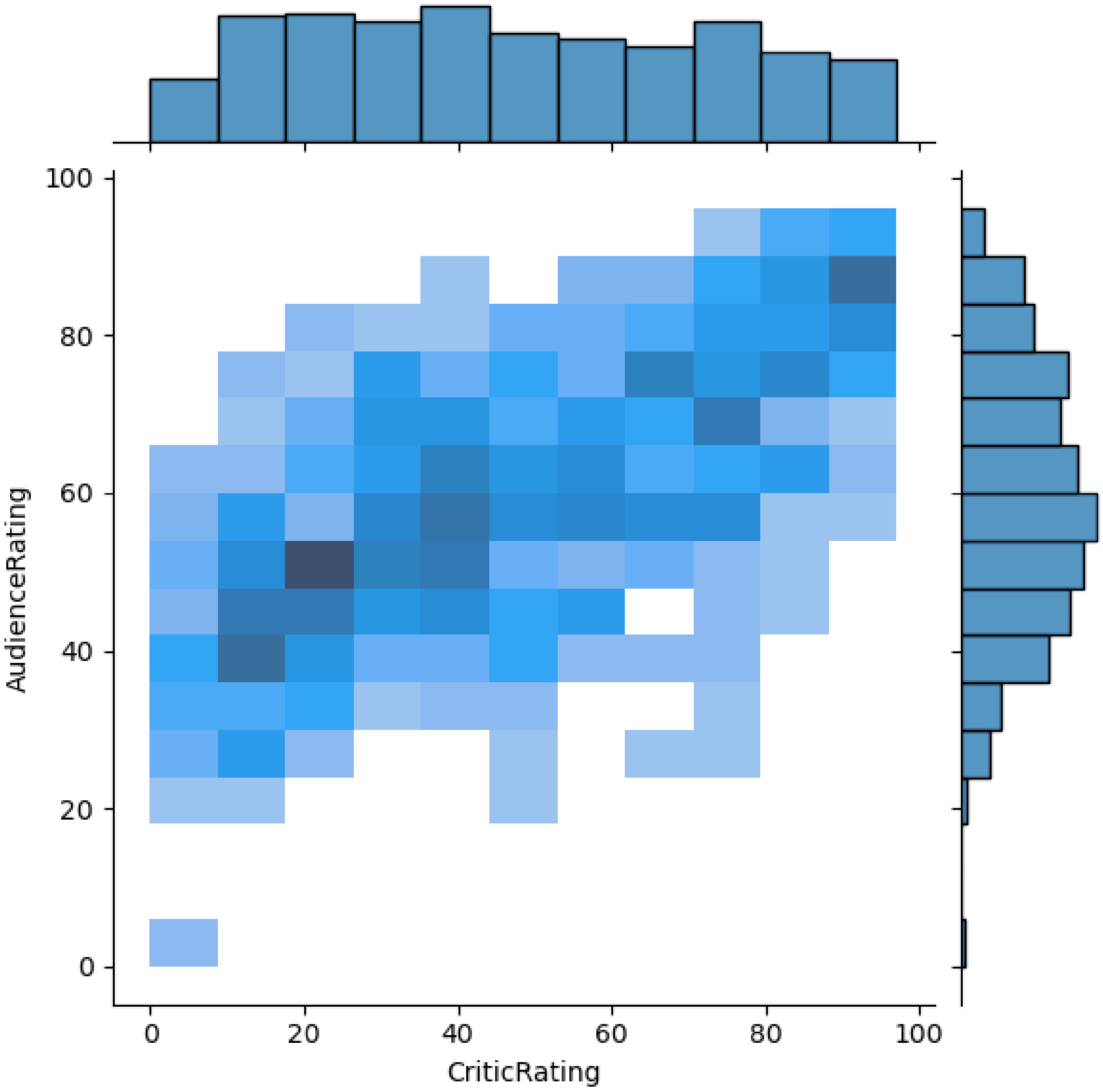
```
In [29]: j = sns.jointplot(data = movies, x ='CriticRating', y = 'AudienceRating', kind='
```



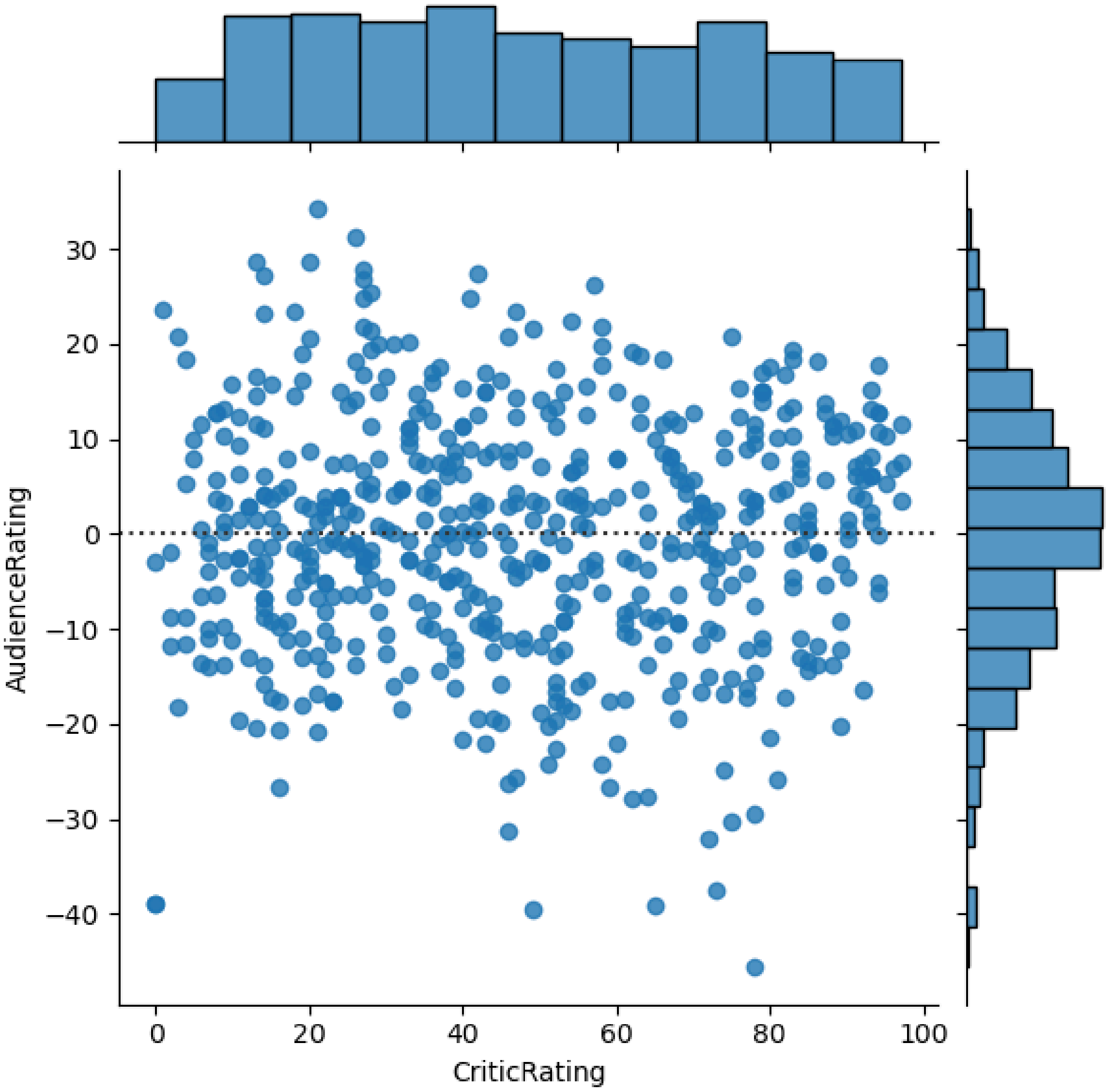
In [30]:
```
j = sns.jointplot(data = movies, x ='CriticRating', y = 'AudienceRating', kind='
```



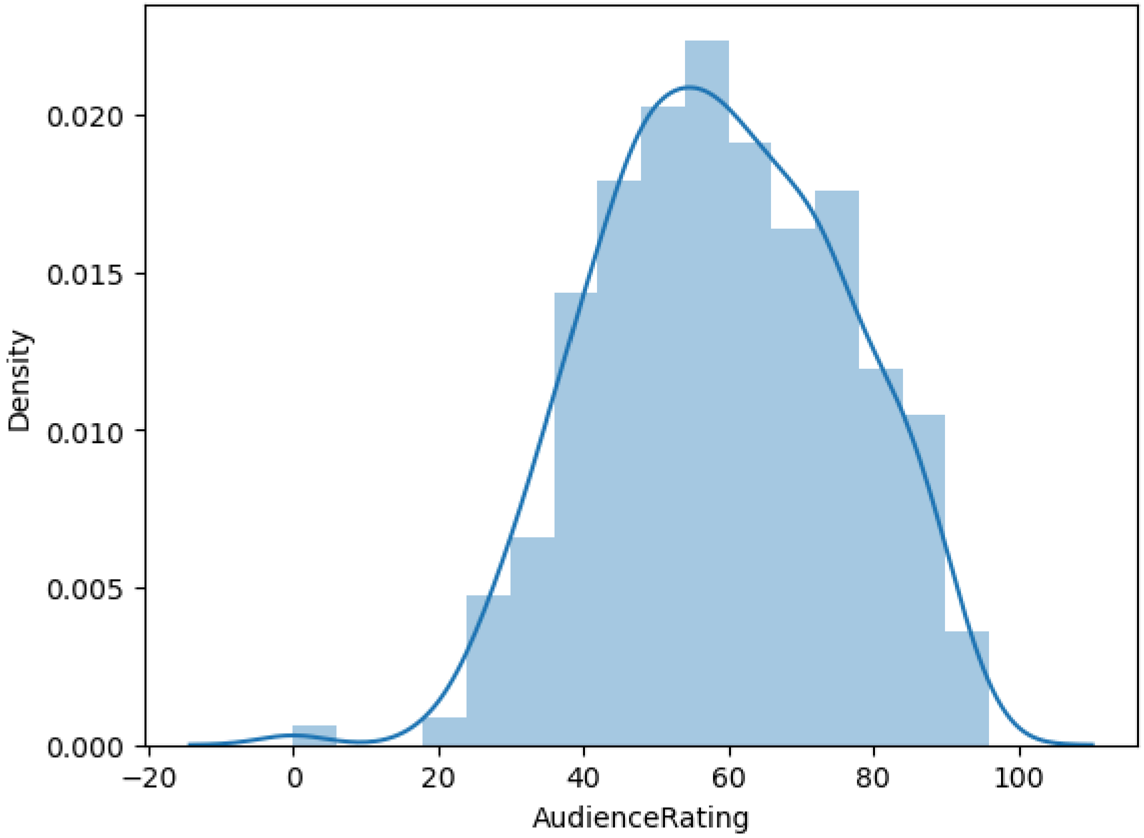
In [31]:
```
j = sns.jointplot(data = movies, x ='CriticRating', y = 'AudienceRating', kind='
```



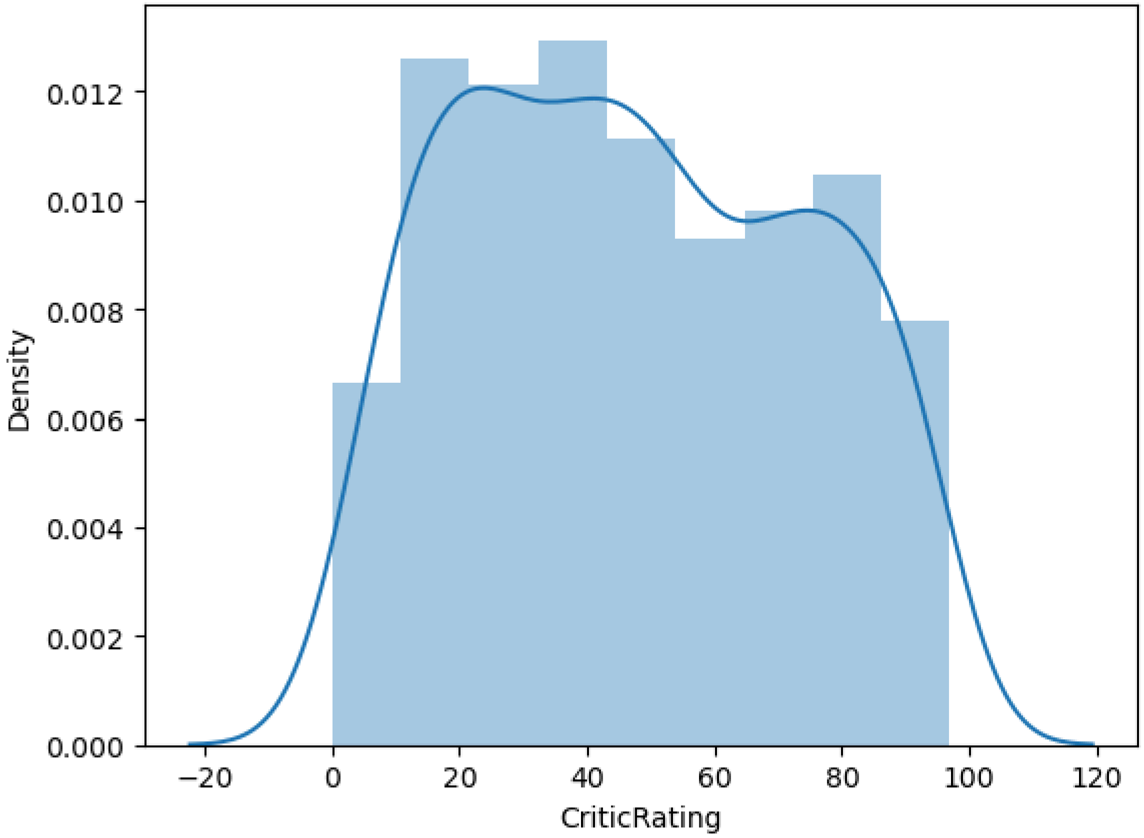
```
In [32]: j = sns.jointplot(data = movies, x ='CriticRating', y = 'AudienceRating', kind='
```



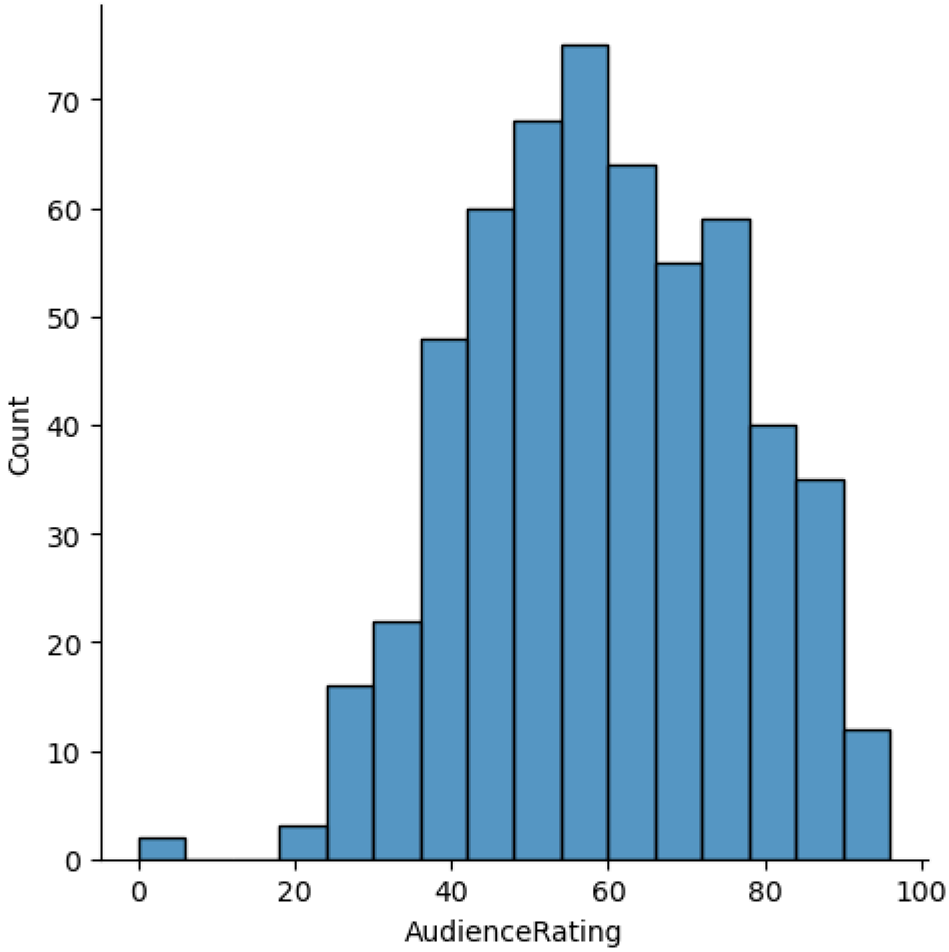
```
In [33]: m1 = sns.distplot(movies.AudienceRating)
```



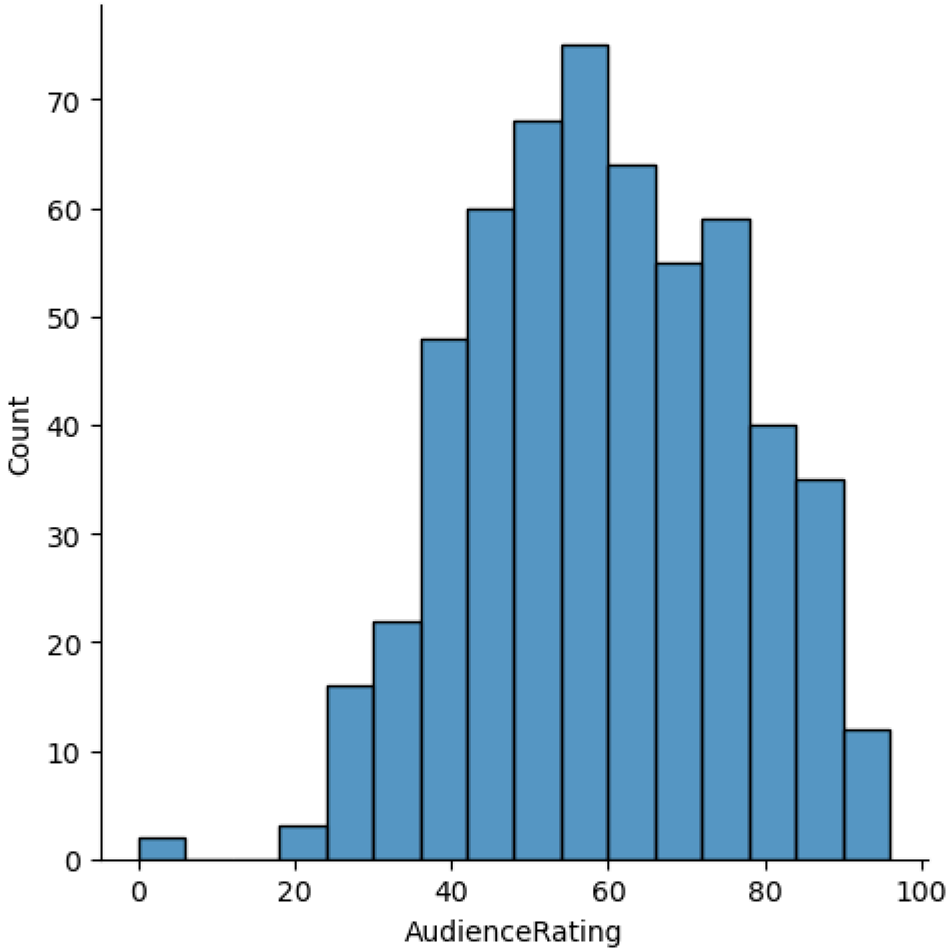
```
In [34]: m1 = sns.distplot(movies.CriticRating)
```



```
In [48]: m1 = sns.displot(movies.AudienceRating)
```

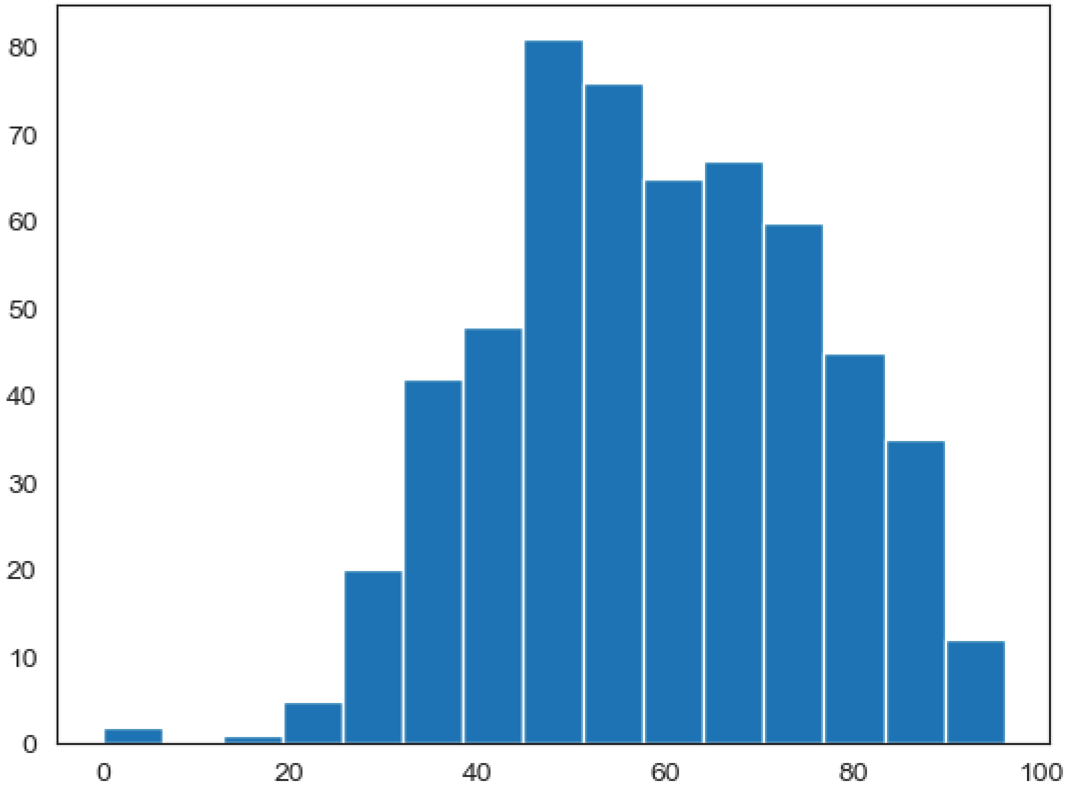


In [50]:
```
m1 = sns.displot(movies['AudienceRating'])
```

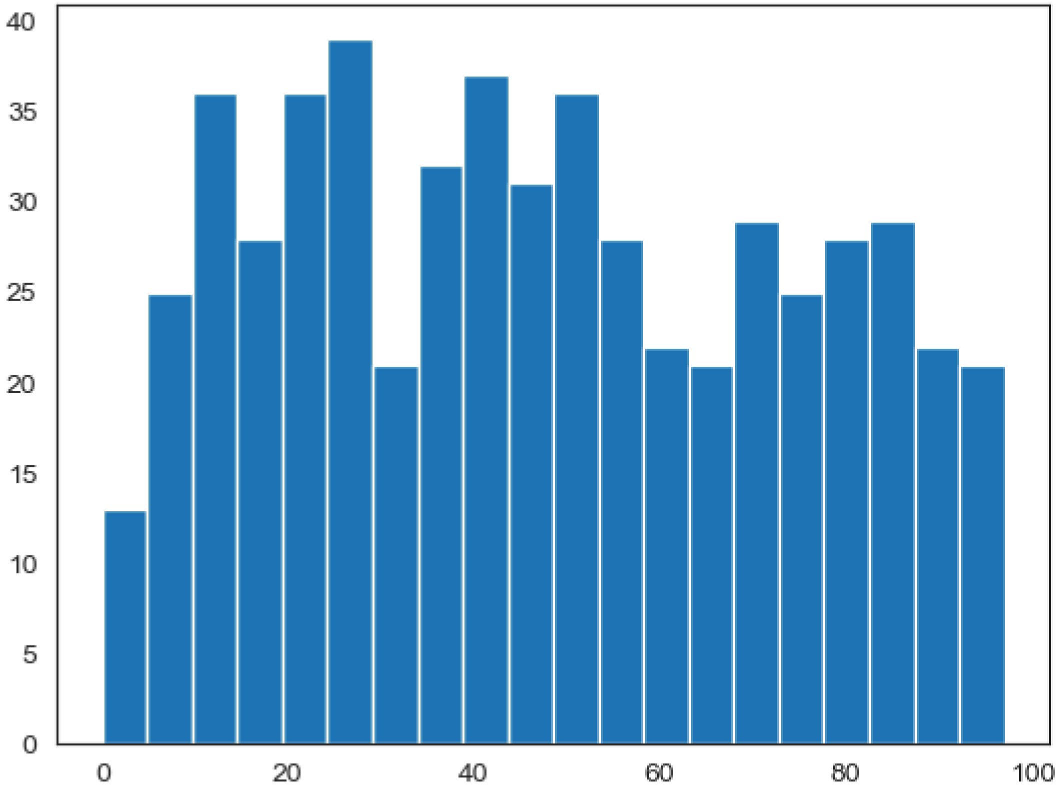


```
In [51]: sns.set_style('white')
```

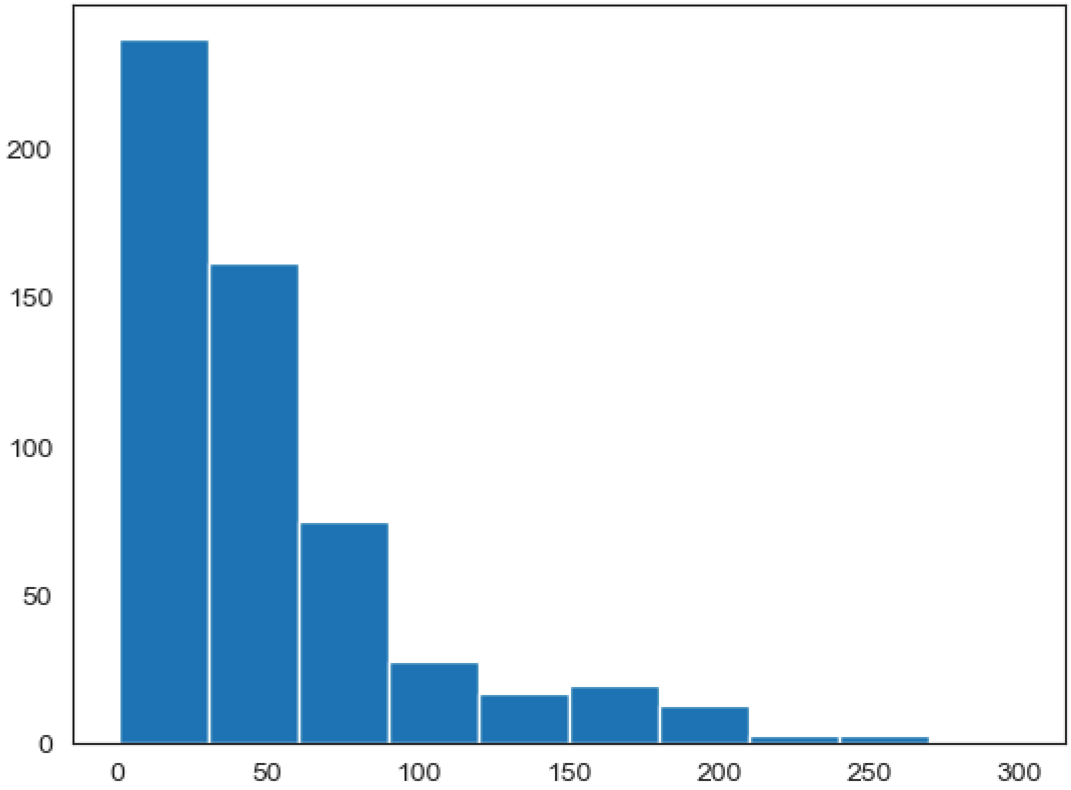
```
In [56]: m1 = plt.hist(movies.AudienceRating, bins = 15)
```



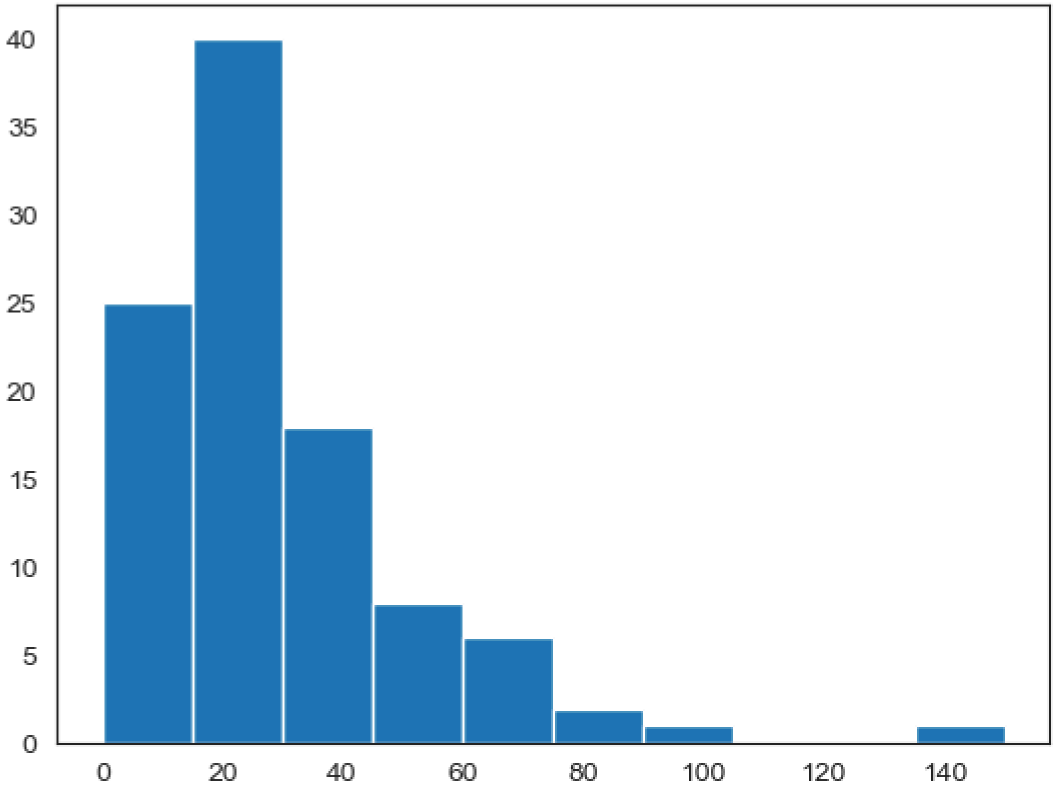
In [57]:
```
m1 = plt.hist(movies.CriticRating, bins=20)
```



In [58]:
```
plt.hist(movies.BudgetMillions)
plt.show()
```



In [59]:
```
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions)
plt.show()
```



In [61]:

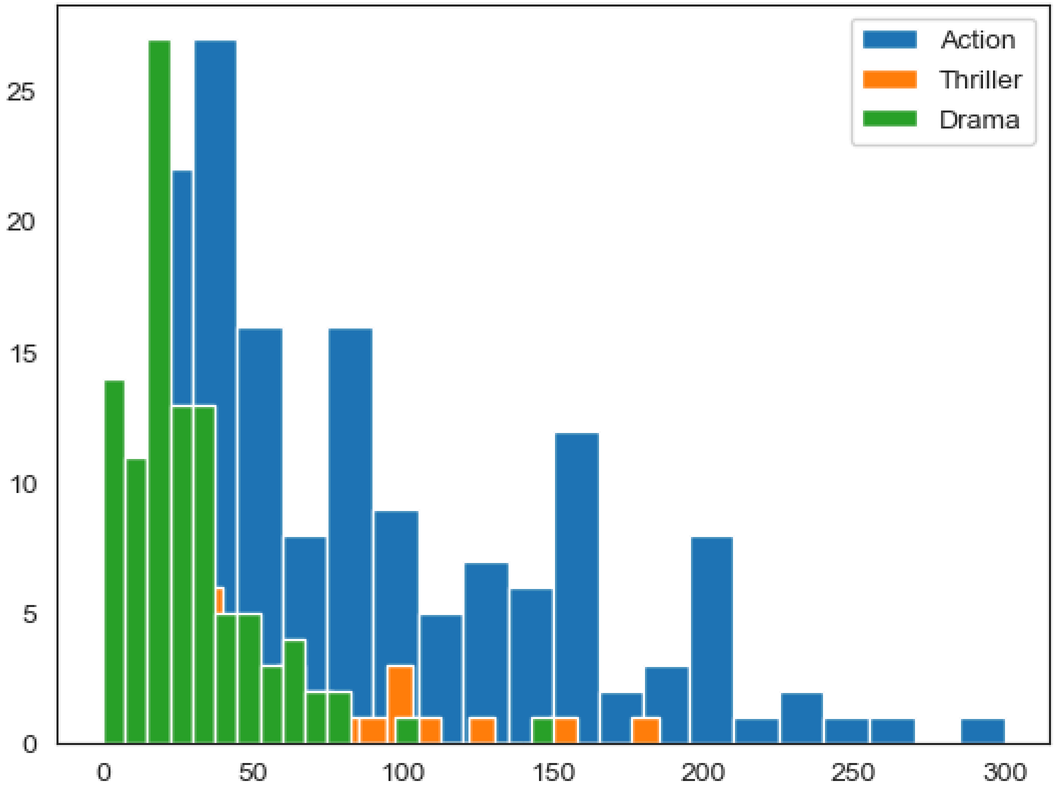
```
movies.head()
```

Out[61]:

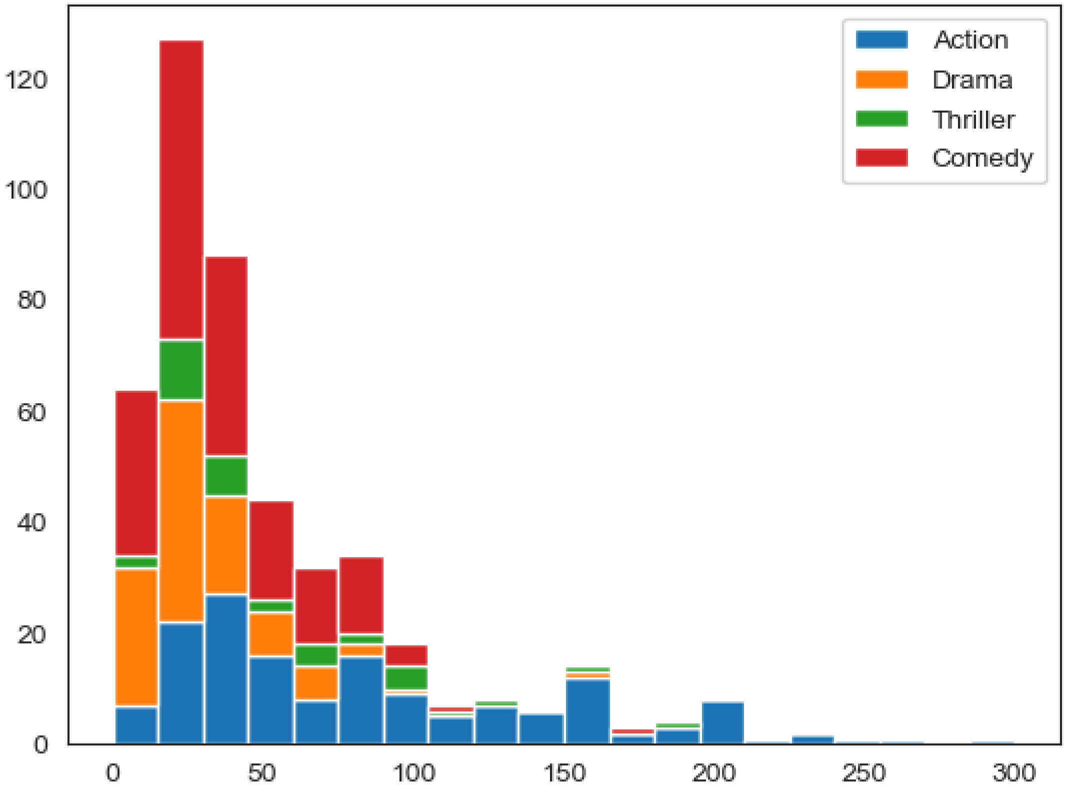
	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009

In [62]:

```
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(['Action', 'Thriller', 'Drama'])
plt.show()
```



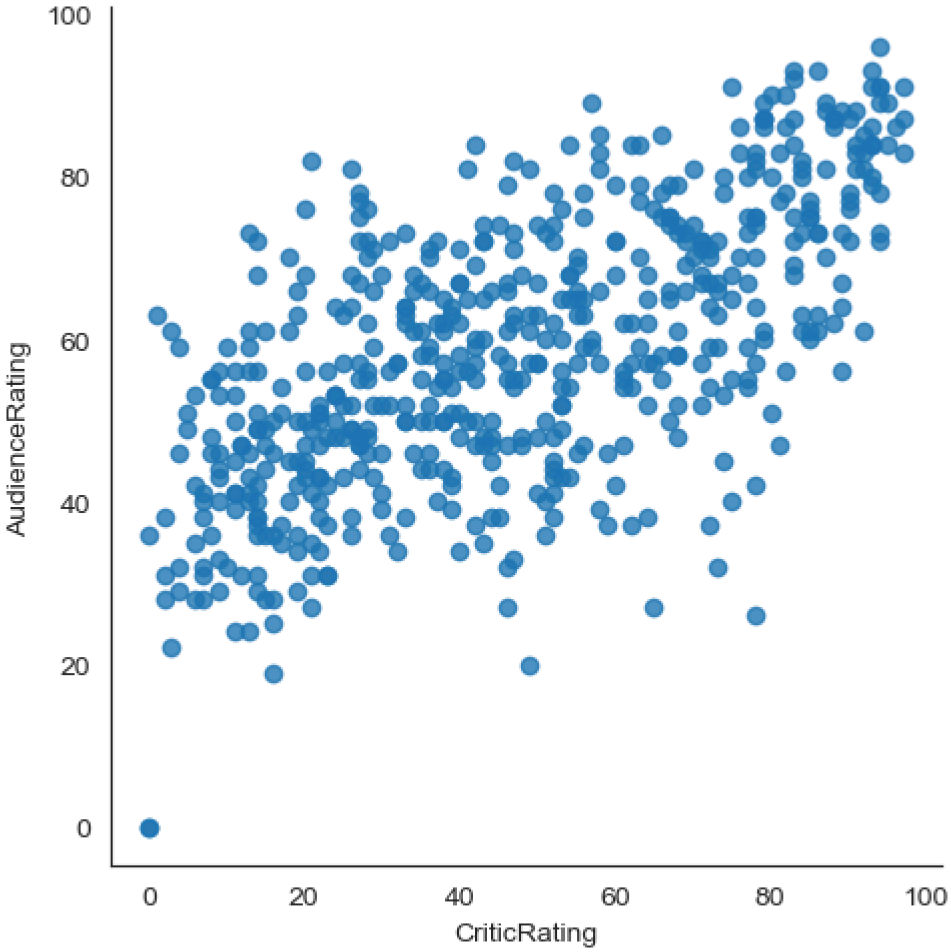
In [64]:
```
plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,\
          movies[movies.Genre == 'Drama'].BudgetMillions,\
            movies[movies.Genre == 'Thriller'].BudgetMillions,\
           movies[movies.Genre == 'Comedy'].BudgetMillions],
           bins=20, stacked=True)

plt.legend(['Action', 'Drama', 'Thriller', 'Comedy'])
plt.show()
```



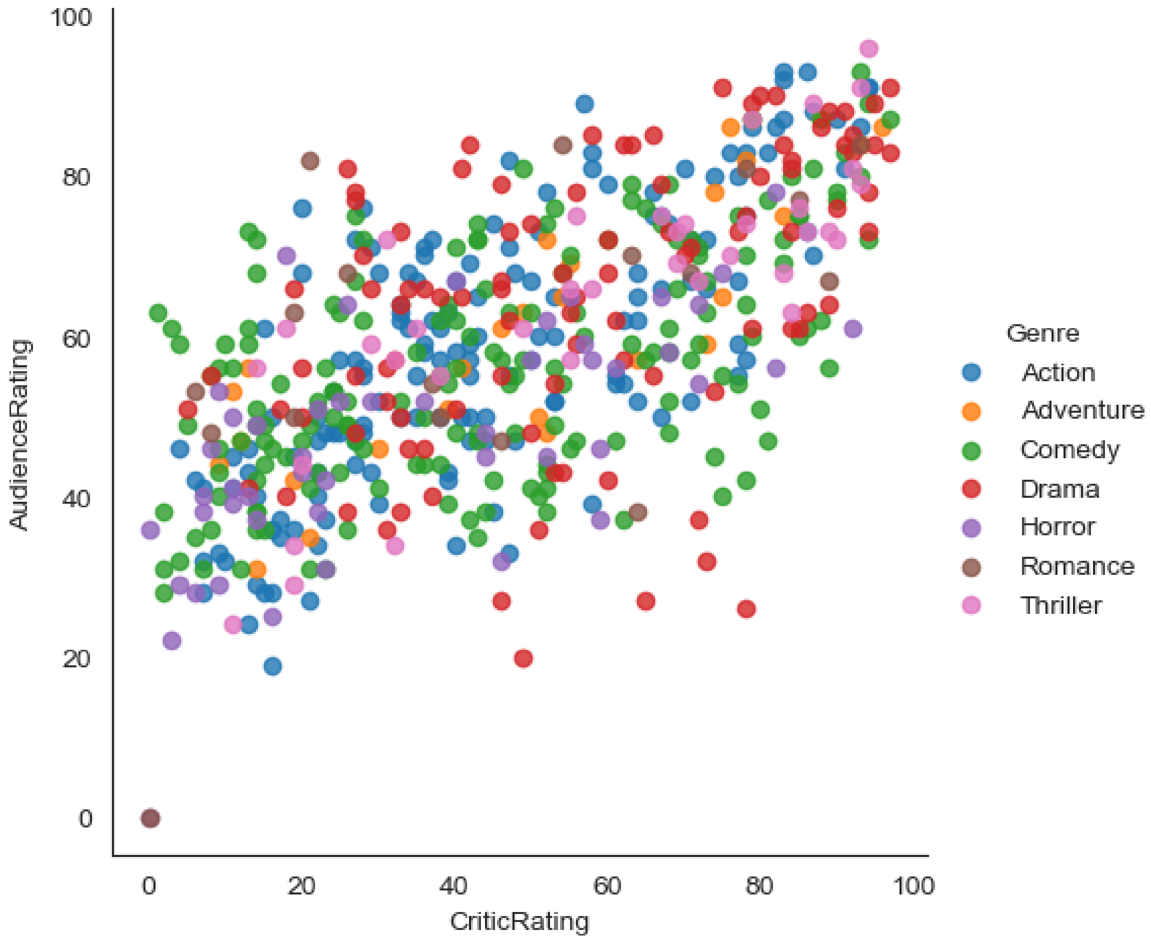
```
In [65]: for gen in movies.Genre.cat.categories:
             print(gen)
```

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

```
In [66]: vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating', fit_reg=Fal
```

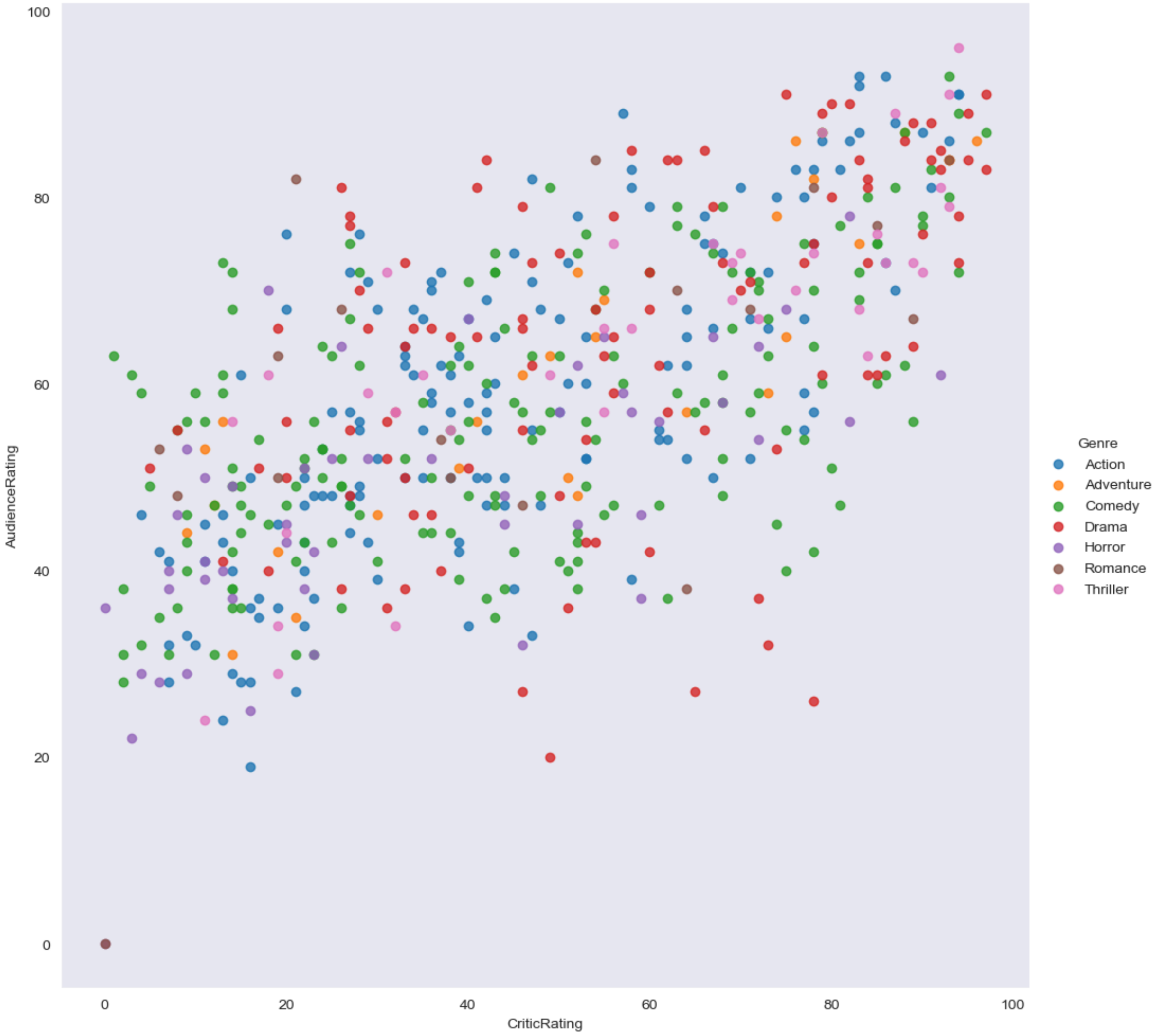


```
In [67]: vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating', fit_reg=Fal
```



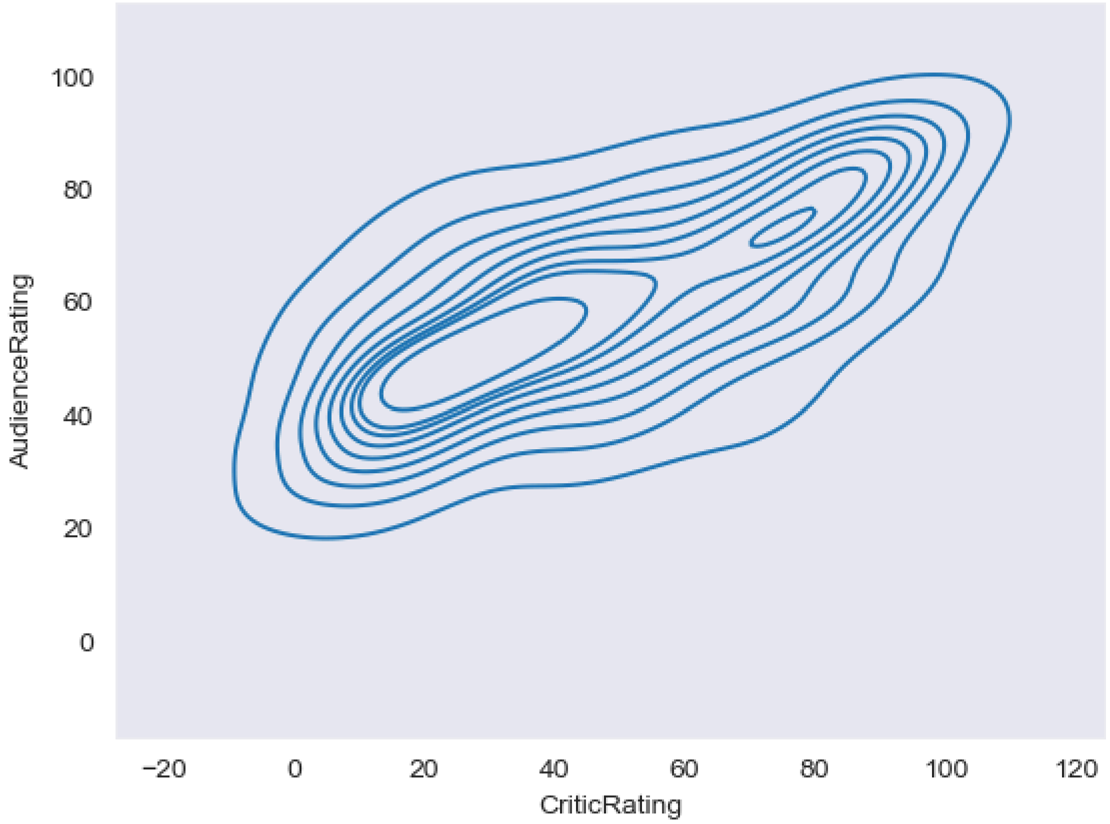
```
In [68]: sns.set_style('dark')
```

```
In [69]: vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating', fit_reg=Fal
```

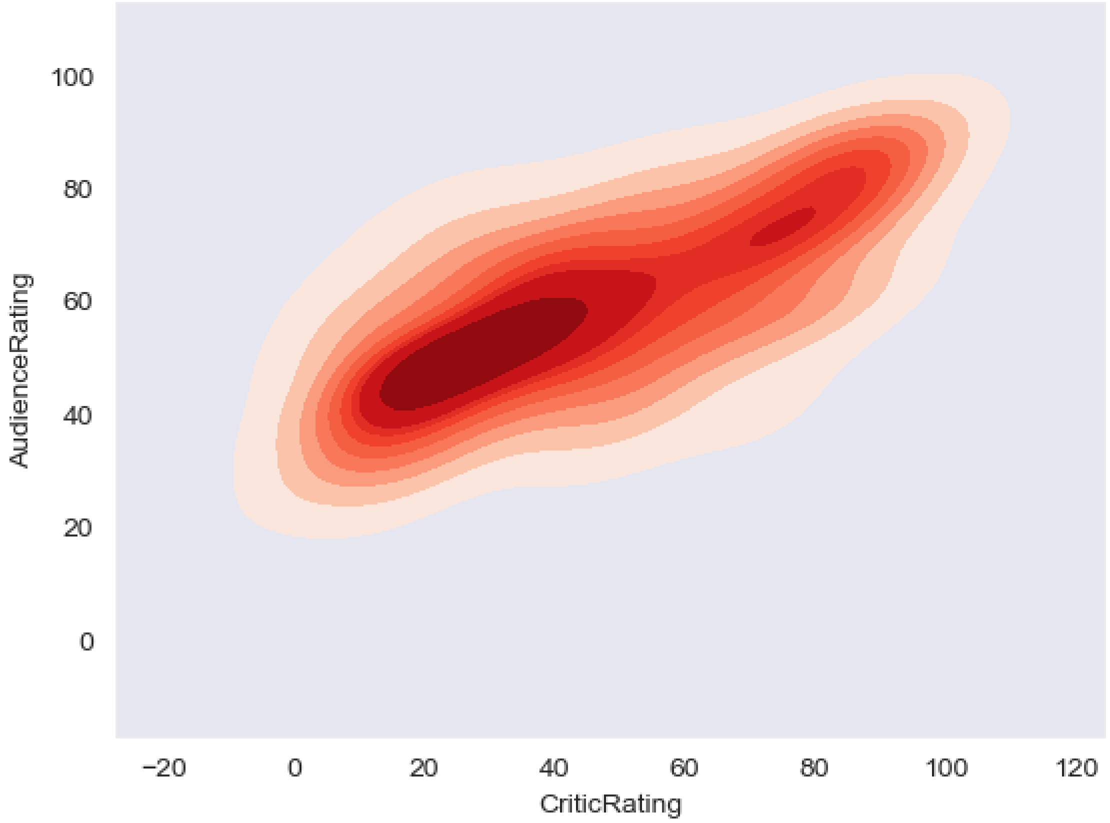


Kernal Density Estimate Plot (KDE Plot)

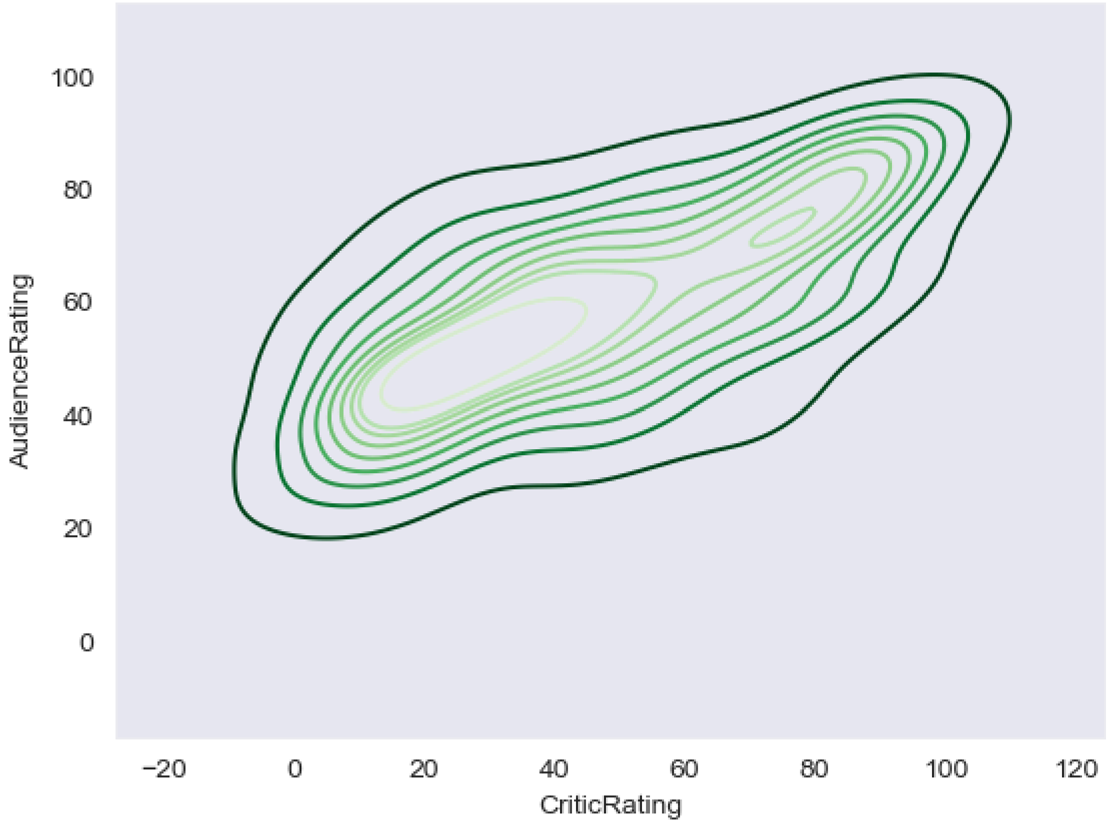
In [70]:
```
k1 = sns.kdeplot(data=movies, x=movies.CriticRating, y=movies.AudienceRating)
```



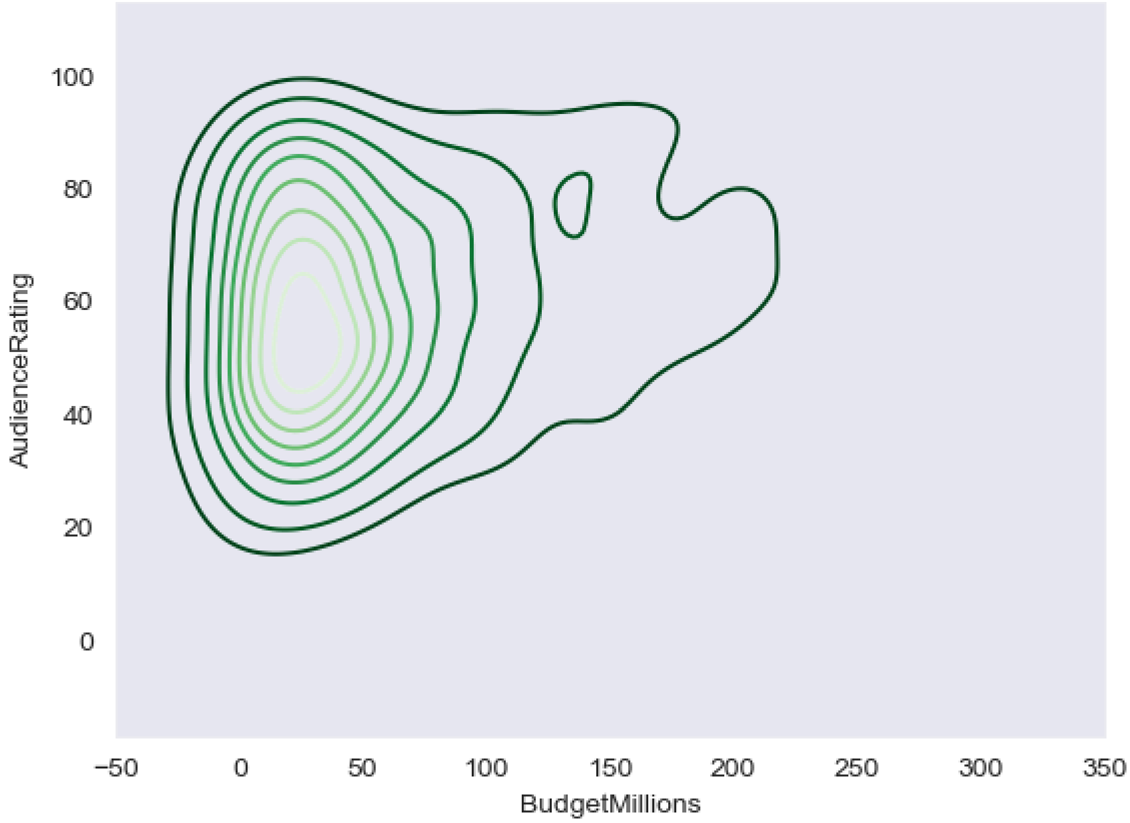
```
In [73]: k1 = sns.kdeplot(data=movies, x='CriticRating', y='AudienceRating', shade=True,s
```



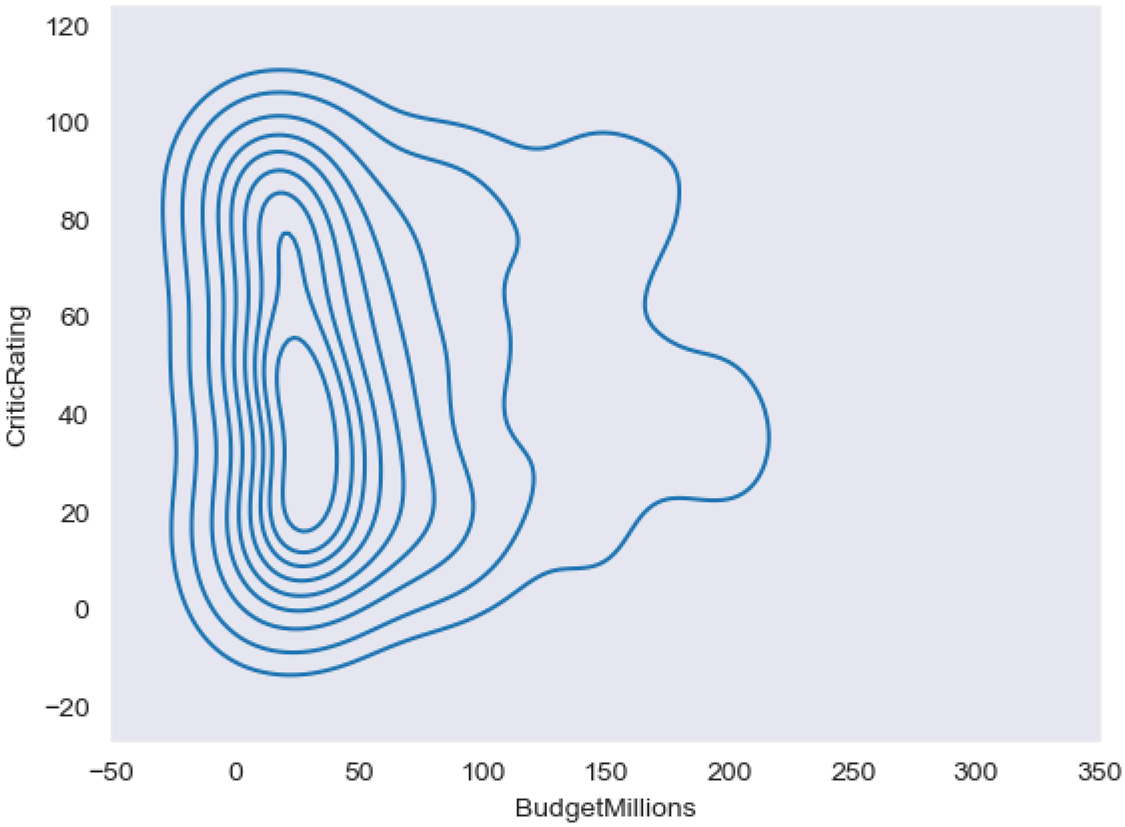
```
In [92]: k1= sns.kdeplot(data=movies, x='CriticRating', y='AudienceRating', shade_lowest=
```



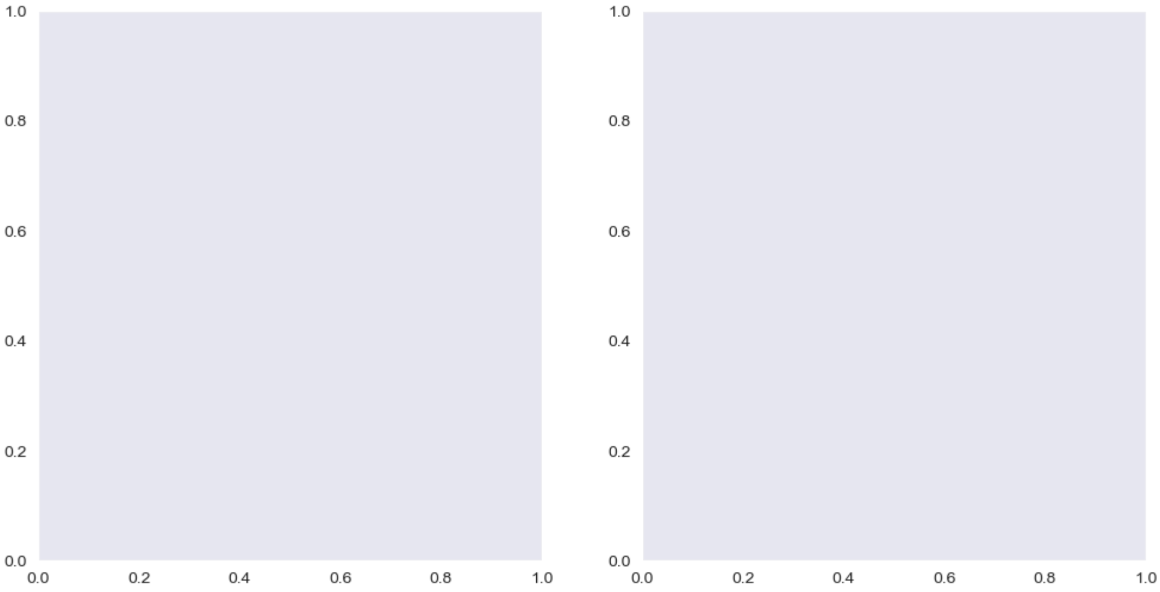
In [97]:
```
k1 = sns.kdeplot(data=movies, x='BudgetMillions', y='AudienceRating', shade_lowe
```



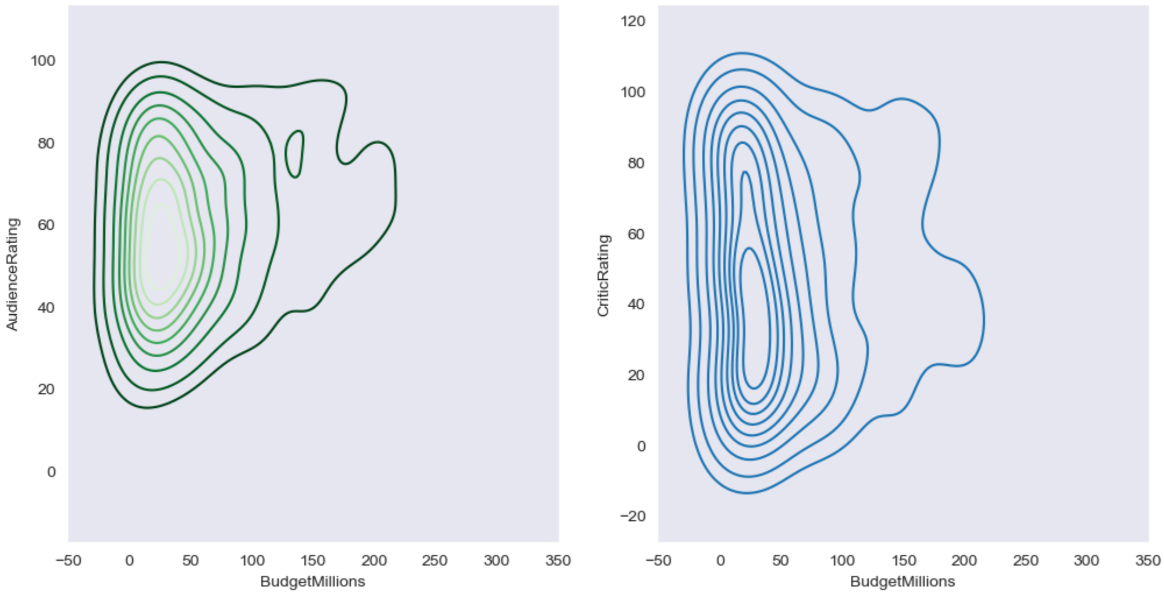
In [98]:
```
k2 =sns.kdeplot(data=movies, x='BudgetMillions', y='CriticRating')
```



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



In [104…
```
f, axes = plt.subplots(1, 2, figsize=(12, 6))
k1 = sns.kdeplot(data=movies, x='BudgetMillions', y='AudienceRating', cmap='Gree
k2 = sns.kdeplot(data=movies, x='BudgetMillions', y='CriticRating', ax=axes[1])
```



In [105…

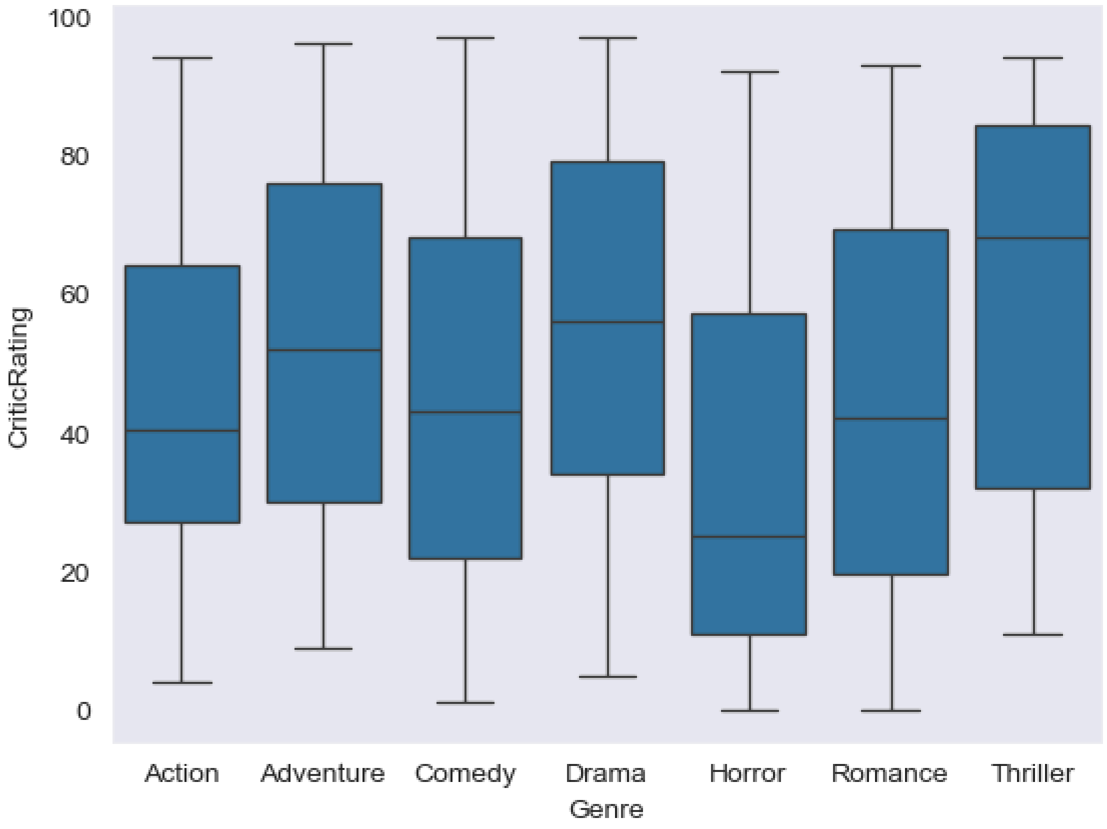
```
axes
```

Out[105…

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

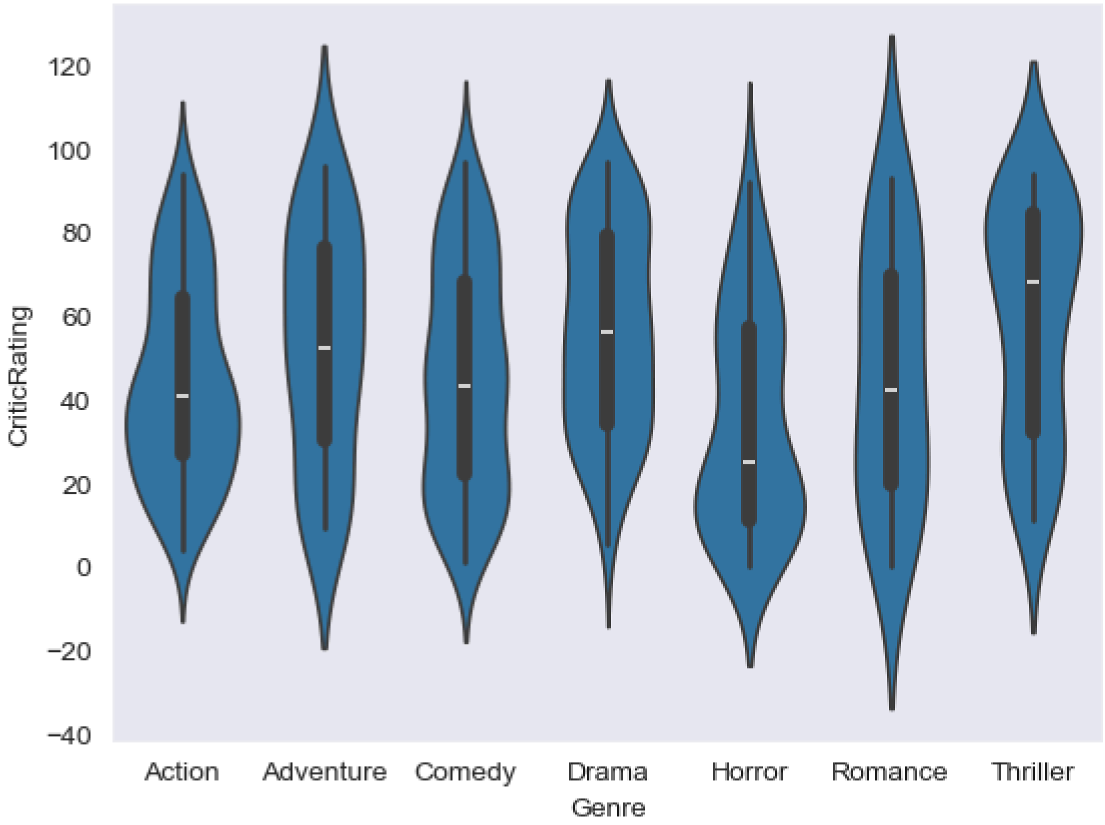
In [106…

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



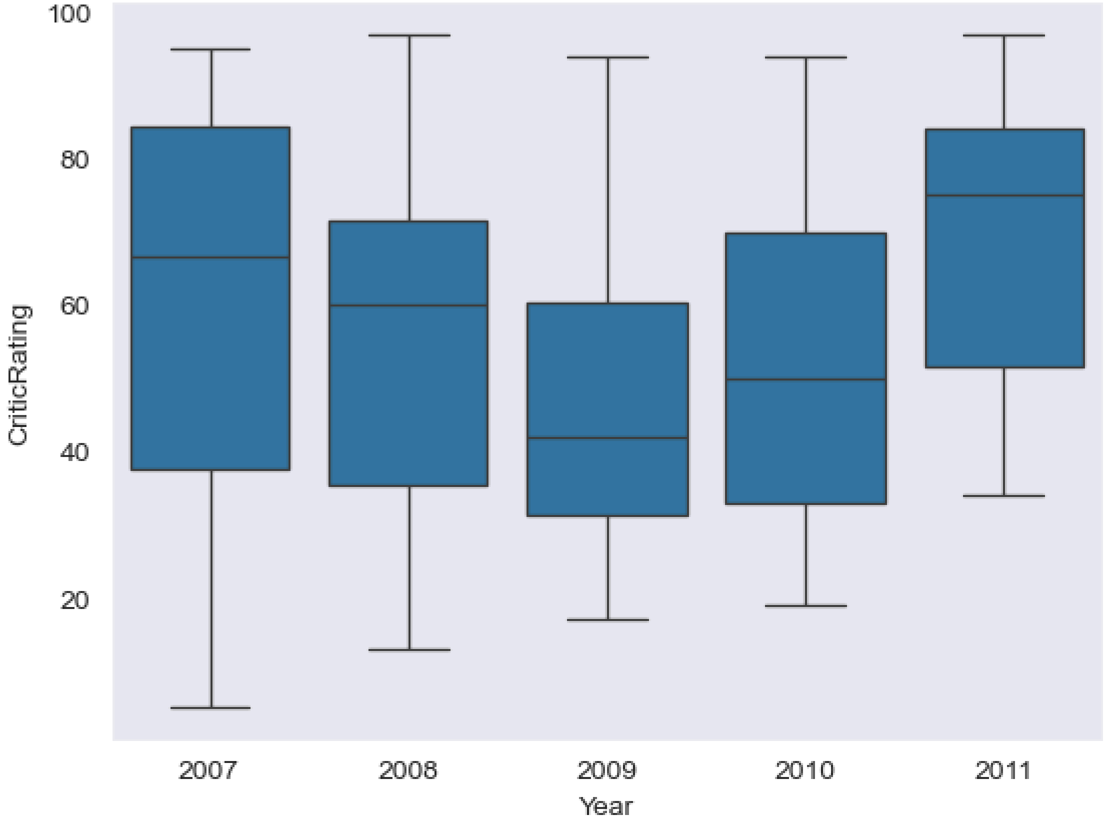
In [107…

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



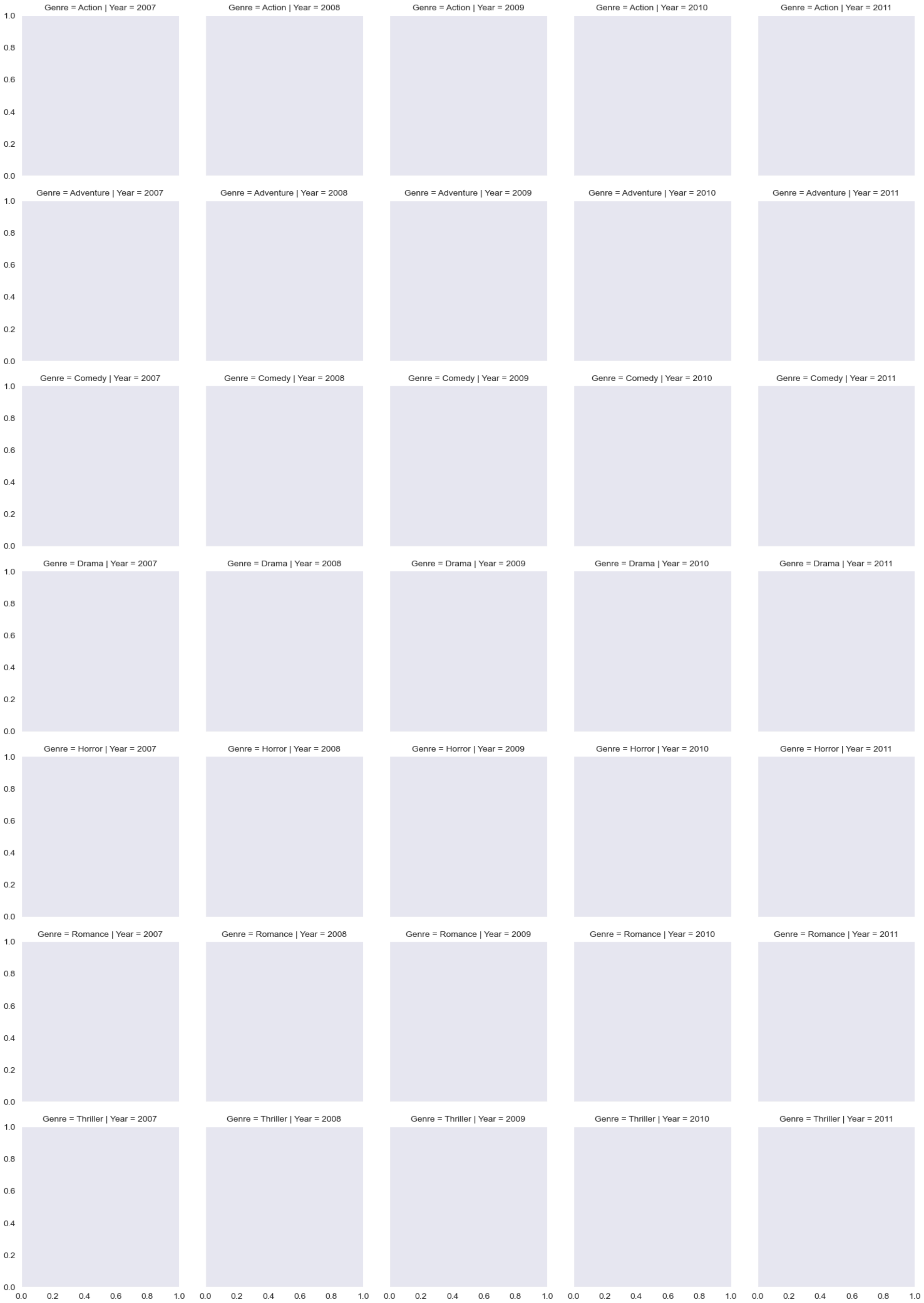
In [108…

```
w1 = sns.boxplot(data=movies[movies.Genre == 'Drama'], x='Year', y= 'CriticRatin
```



In [110…

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

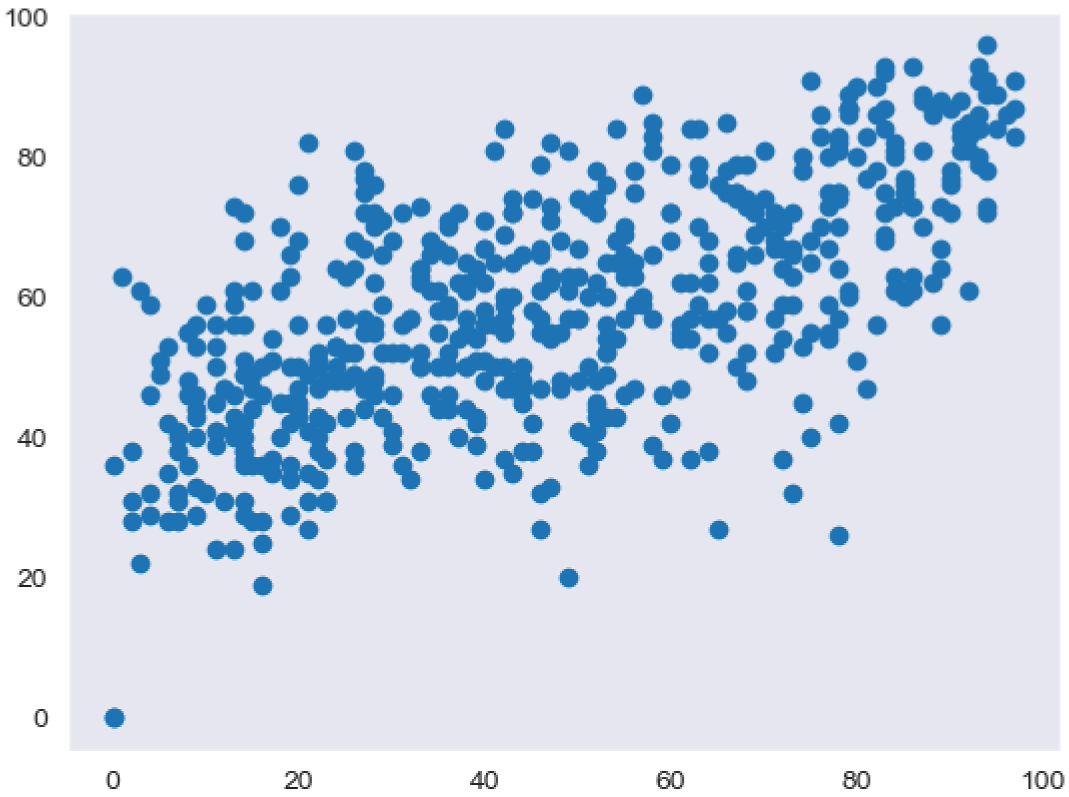


In [112…

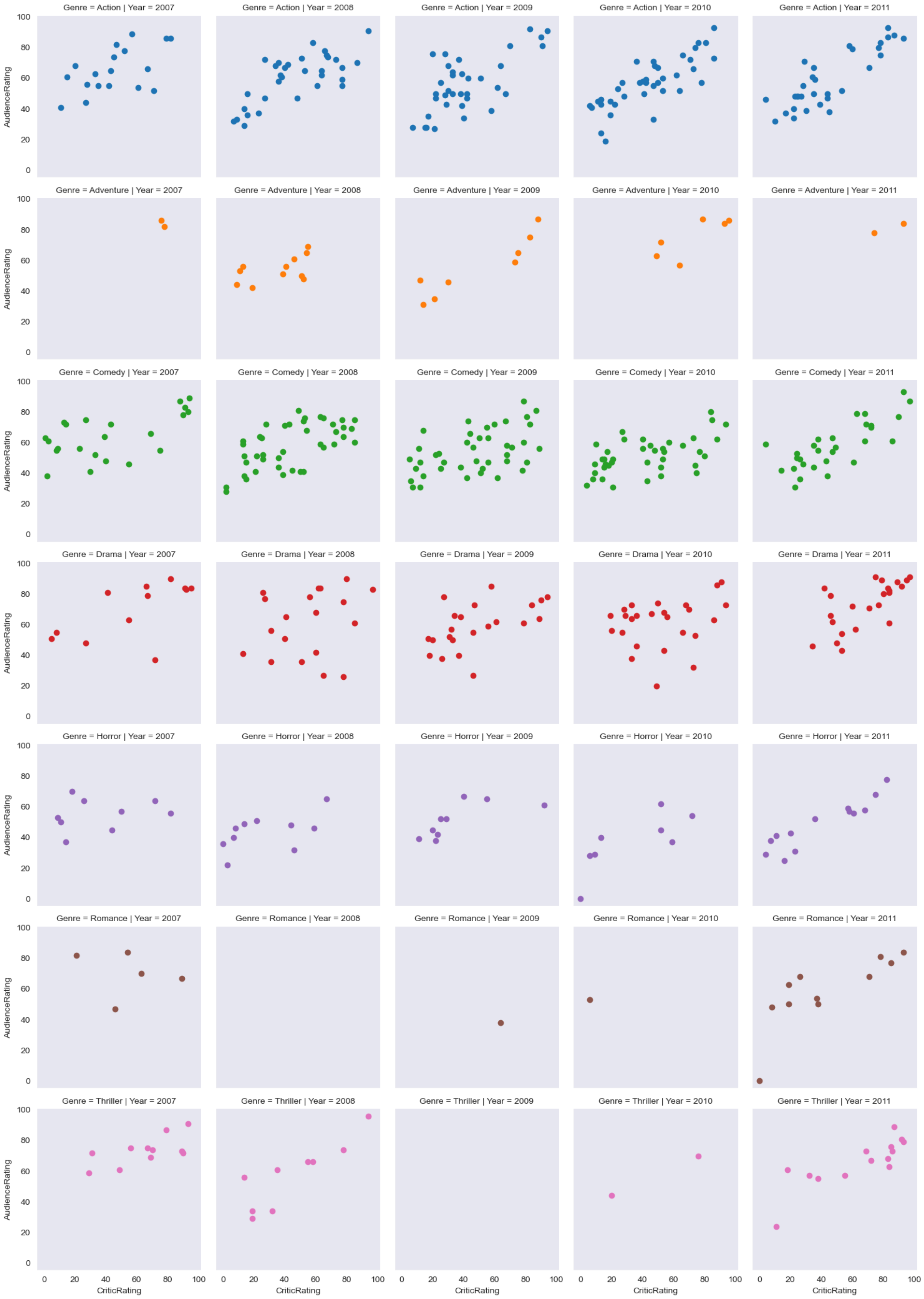
```
plt.scatter(movies.CriticRating,movies.AudienceRating)
```

Out[112…

```
<matplotlib.collections.PathCollection at 0x1788452ac10>
```

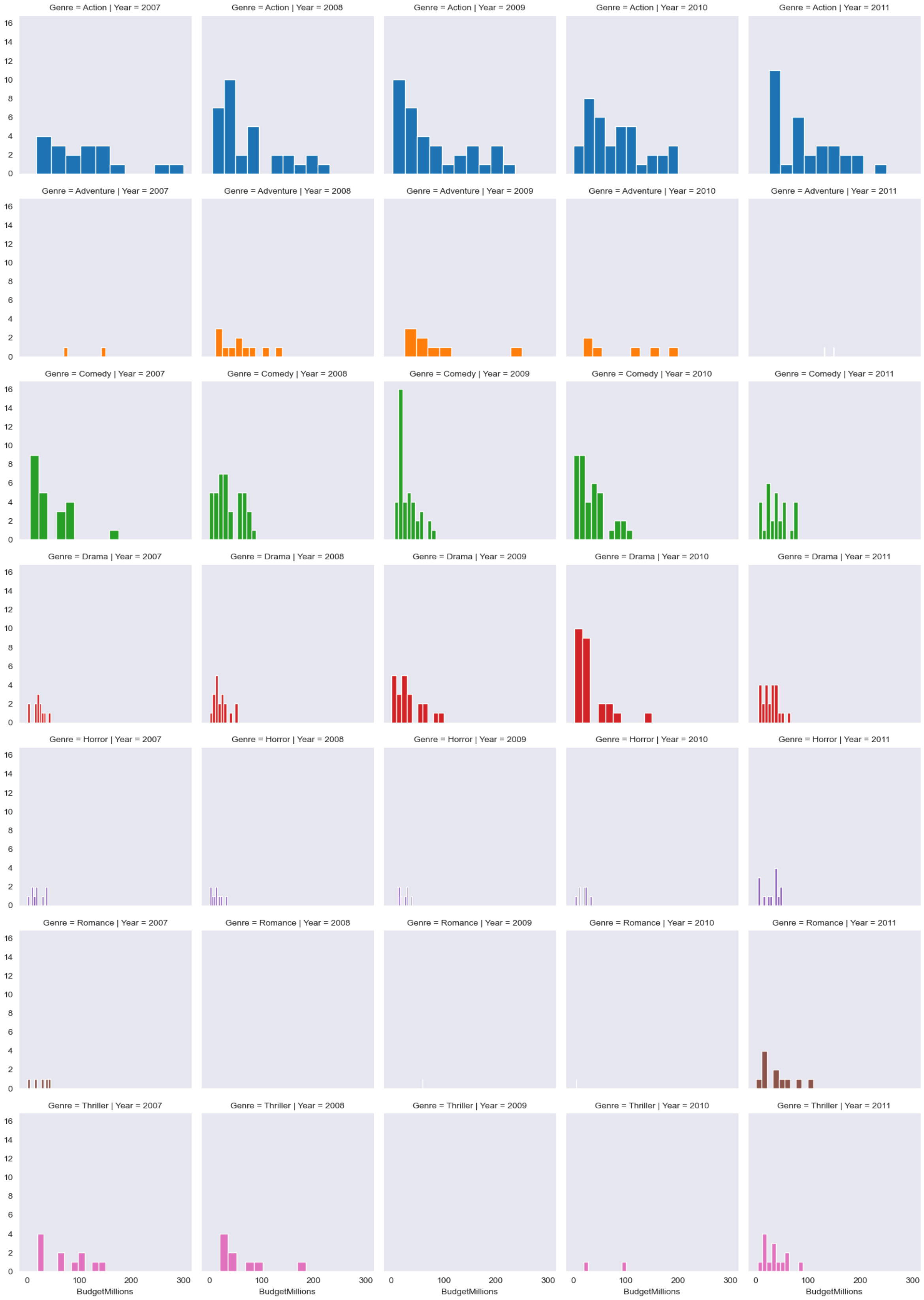


```
In [116…  g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue ='Genre')
          g= g.map(plt.scatter, 'CriticRating', 'AudienceRating' )
```



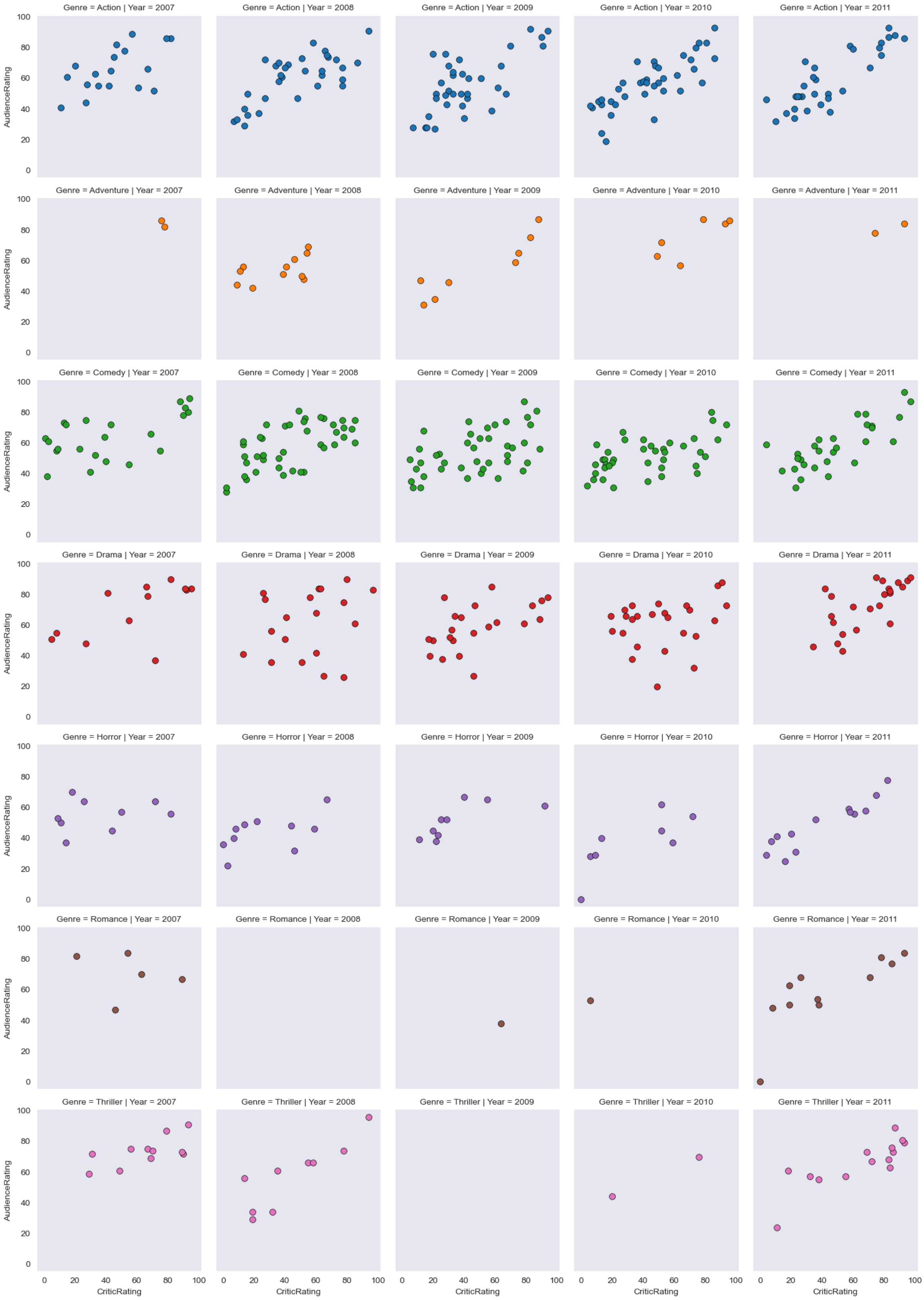
In [117…

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



In [119…

```
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)
```



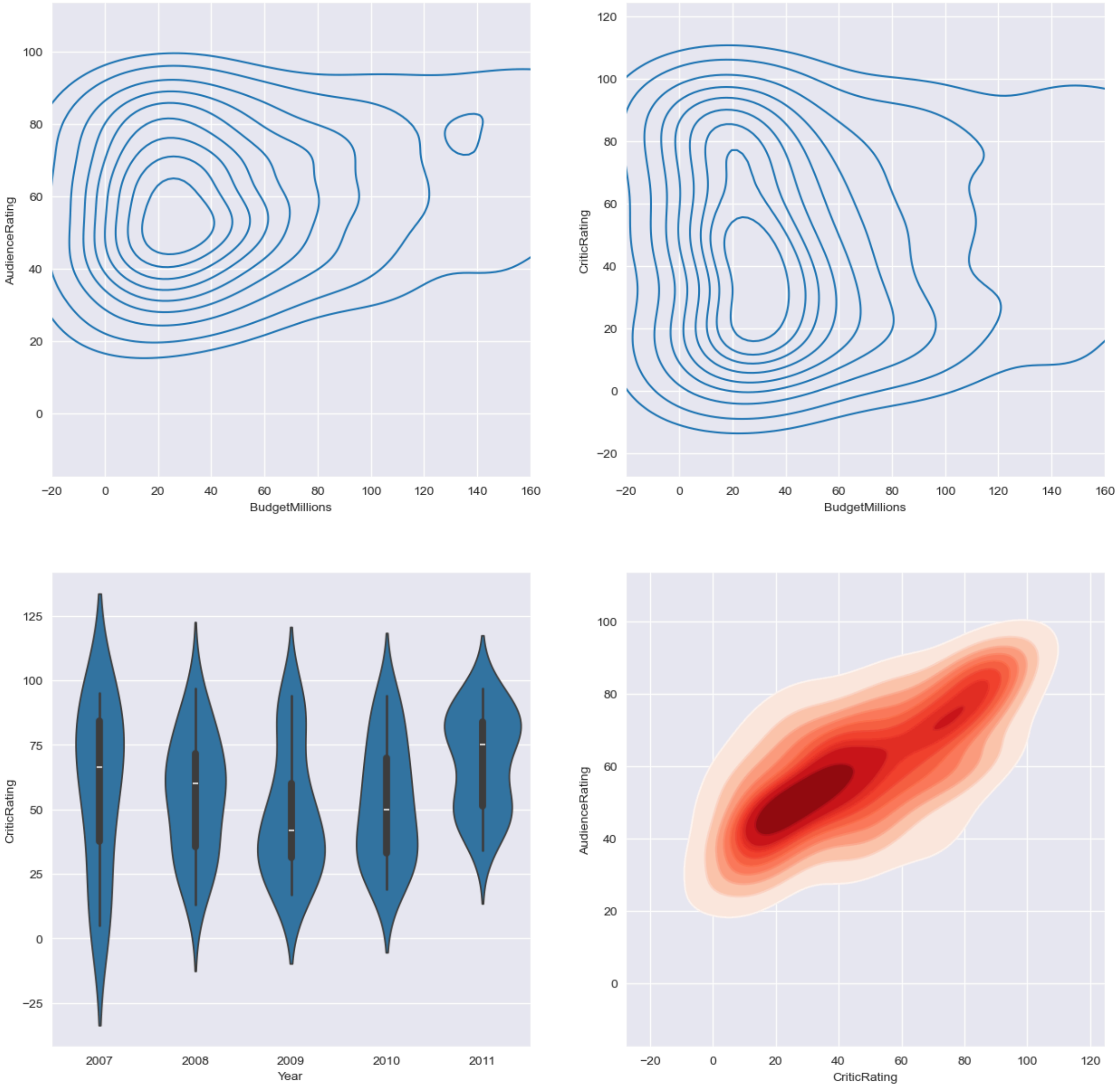
In [120…

```
pip install seaborn==0.11.2
```

```
Collecting seaborn==0.11.2
  Downloading seaborn-0.11.2-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: numpy>=1.15 in c:\users\hp\anaconda3\lib\site-pack
ages (from seaborn==0.11.2) (2.1.3)
Requirement already satisfied: scipy>=1.0 in c:\users\hp\anaconda3\lib\site-packa
ges (from seaborn==0.11.2) (1.15.3)
Requirement already satisfied: pandas>=0.23 in c:\users\hp\anaconda3\lib\site-pac
kages (from seaborn==0.11.2) (2.2.3)
Requirement already satisfied: matplotlib>=2.2 in c:\users\hp\anaconda3\lib\site-
packages (from seaborn==0.11.2) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\hp\anaconda3\lib\site
-packages (from matplotlib>=2.2->seaborn==0.11.2) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\hp\anaconda3\lib\site-pac
kages (from matplotlib>=2.2->seaborn==0.11.2) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\hp\anaconda3\lib\sit
e-packages (from matplotlib>=2.2->seaborn==0.11.2) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\hp\anaconda3\lib\sit
e-packages (from matplotlib>=2.2->seaborn==0.11.2) (1.4.8)
Requirement already satisfied: packaging>=20.0 in c:\users\hp\anaconda3\lib\site-
packages (from matplotlib>=2.2->seaborn==0.11.2) (24.2)
Requirement already satisfied: pillow>=8 in c:\users\hp\anaconda3\lib\site-packag
es (from matplotlib>=2.2->seaborn==0.11.2) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\hp\anaconda3\lib\site
-packages (from matplotlib>=2.2->seaborn==0.11.2) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\hp\anaconda3\lib
\site-packages (from matplotlib>=2.2->seaborn==0.11.2) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\hp\anaconda3\lib\site-pac
kages (from pandas>=0.23->seaborn==0.11.2) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\hp\anaconda3\lib\site-p
ackages (from pandas>=0.23->seaborn==0.11.2) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\lib\site-package
s (from python-dateutil>=2.7->matplotlib>=2.2->seaborn==0.11.2) (1.17.0)
Downloading seaborn-0.11.2-py3-none-any.whl (292 kB)
Installing collected packages: seaborn
  Attempting uninstall: seaborn
    Found existing installation: seaborn 0.13.2
    Uninstalling seaborn-0.13.2:
      Successfully uninstalled seaborn-0.13.2
Successfully installed seaborn-0.11.2
Note: you may need to restart the kernel to use updated packages.
```

In [122…

```python
sns.set_style('darkgrid')
f, axes = plt.subplots(2, 2, figsize=(15, 15))
# KDE plots
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))
# Violin plot
z = sns.violinplot(
data=movies[movies.Genre == 'Drama'],
x='Year',
y='CriticRating',
ax=axes[1,0]
)
# Scatter style KDE
k4 = sns.kdeplot(
x=movies.CriticRating,
y=movies.AudienceRating,
fill=True,
ax=axes[1,1],
cmap='Reds'
)
k4b = sns.kdeplot(
x=movies.CriticRating,
y=movies.AudienceRating,
ax=axes[1,1],
cmap='Reds'
)
plt.show()
```


AudienceRating
BudgetMillions
CriticRating
BudgetMillions
CriticRating
Year
AudienceRating
CriticRating