Advance Visualization

```
In [2]: import pandas as pd
        import os
```

```
In [3]: movies = pd.read_csv(r"/content/Movie-Rating.csv")
        movies
```

Out[3]:

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million $)	Year of release
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009
...	...	...	...	...	...	...
554	Your Highness	Comedy	26	36	50	2011
555	Youth in Revolt	Comedy	68	52	18	2009
556	Zodiac	Thriller	89	73	65	2007
557	Zombieland	Action	90	87	24	2009
558	Zookeeper	Comedy	14	42	80	2011

559 rows × 6 columns

```
In [4]: print(id(movies))
        print(type(movies))
```

```
139371249182496
<class 'pandas.core.frame.DataFrame'>
```

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

	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]: import numpy
```

```
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()
```

```
<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`

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

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

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 = ['Film', 'Genre','CriticRating','AudienceRating','BudgetMilli`

```
In [14]: movies.head(1)
```

Out[14]:

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

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

Out[15]:

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

```
In [18]: movies.Film
```

	Film
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

559 rows × 1 columns

dtype: category

In [19]:
```python
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 [20]:
```python
movies.Genre = movies.Genre.astype('category')
movies.Year = movies.Year.astype('category')
```

In [21]:
```python
movies.Genre
```

	Genre
0	Comedy
1	Adventure
2	Action
3	Adventure
4	Comedy
...	...
554	Comedy
555	Comedy
556	Thriller
557	Action
558	Comedy

559 rows × 1 columns

dtype: category

`movies.Year`

	Year
0	2009
1	2008
2	2009
3	2010
4	2009
...	...
554	2011
555	2009
556	2007
557	2009
558	2011

559 rows × 1 columns

dtype: category

```
In [23]: 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 [24]: movies.describe()
```

Out[24]:

	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

```python
In [36]: from matplotlib import pyplot as plt
         import seaborn as sns

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

```python
In [26]: j = sns.jointplot(data = movies, x = 'CriticRating', y = 'AudienceRating', kir
         #Audience rating is more dominant then critic rating
         #Possitive corelation
```
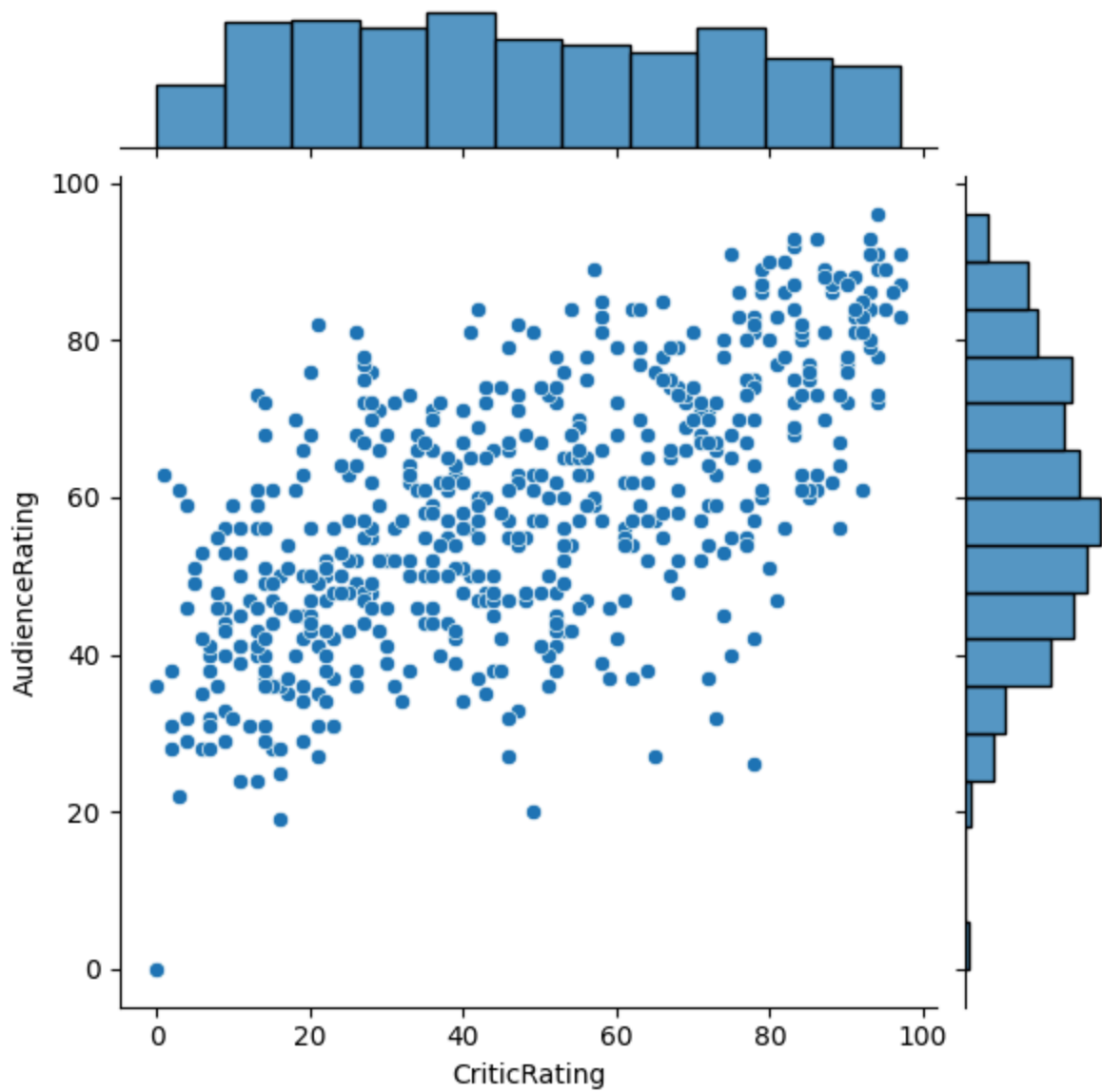
```
In [27]: j = sns.jointplot(data = movies, x = 'CriticRating', y = 'AudienceRating', kir
         )
         #uniform gra
```
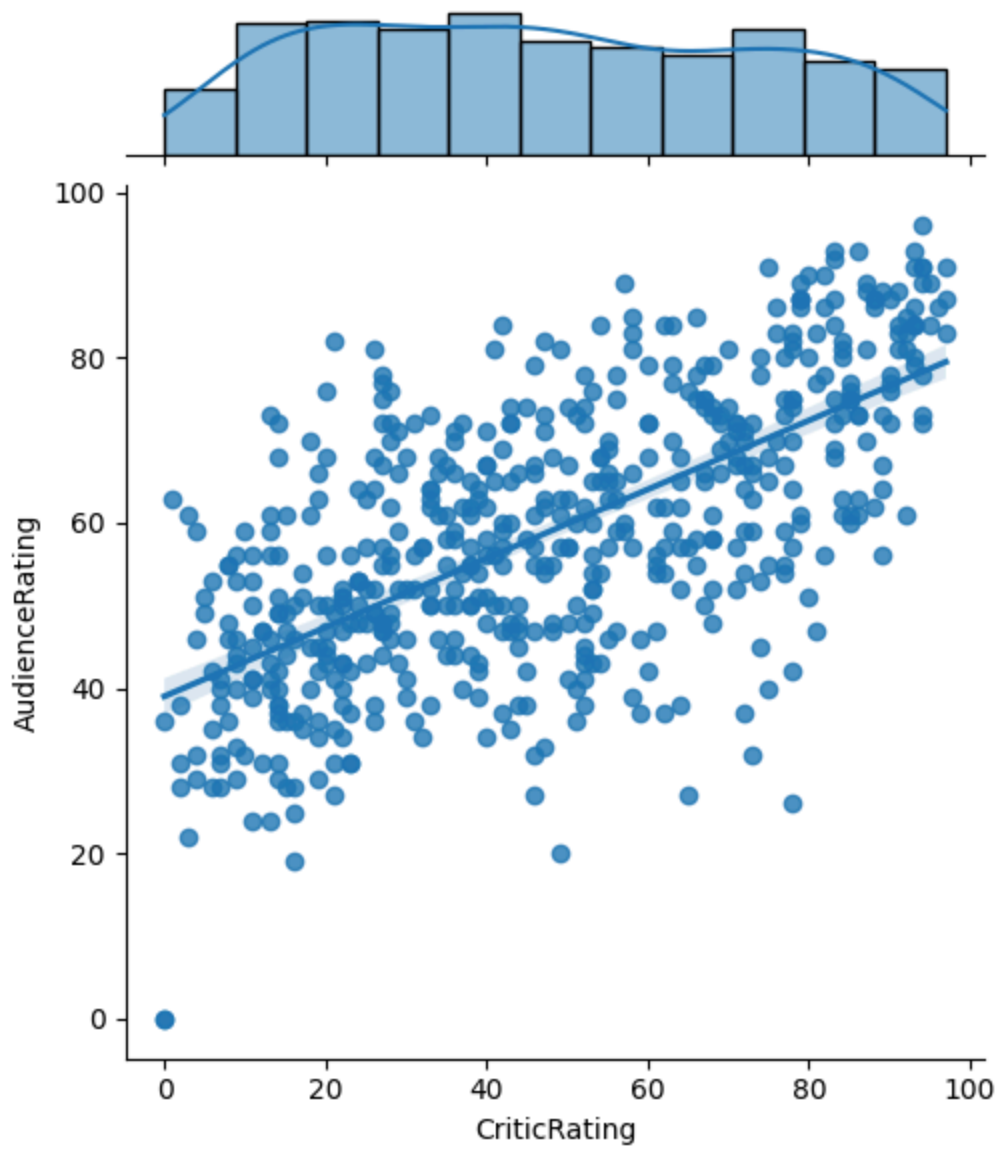
```
In [28]: j = sns.jointplot(data = movies, x = 'CriticRating', y = 'AudienceRating', kir
```
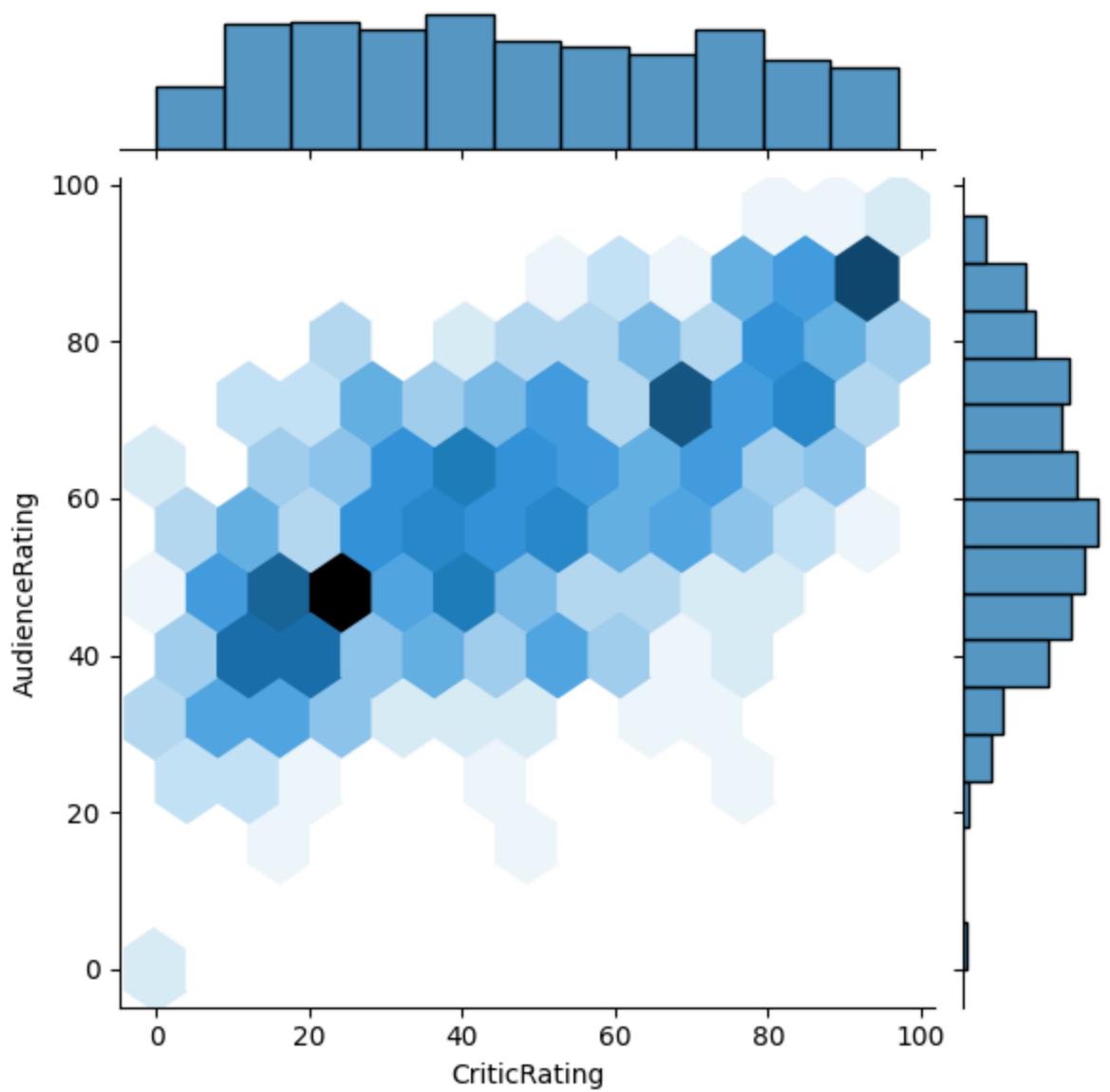
`m1 = sns.distplot(movies.AudienceRating) #univariated Analyasis`
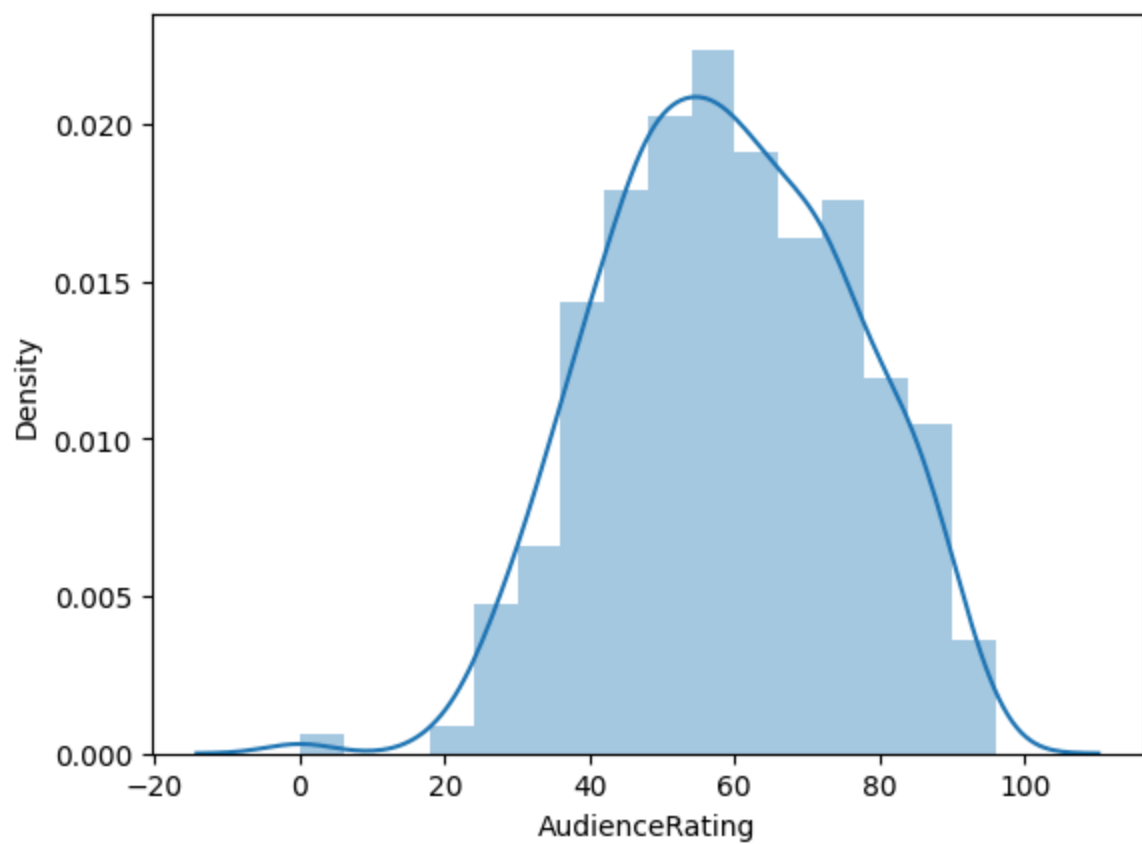
In [30]: `m1 = sns.distplot(movies.CriticRating)`
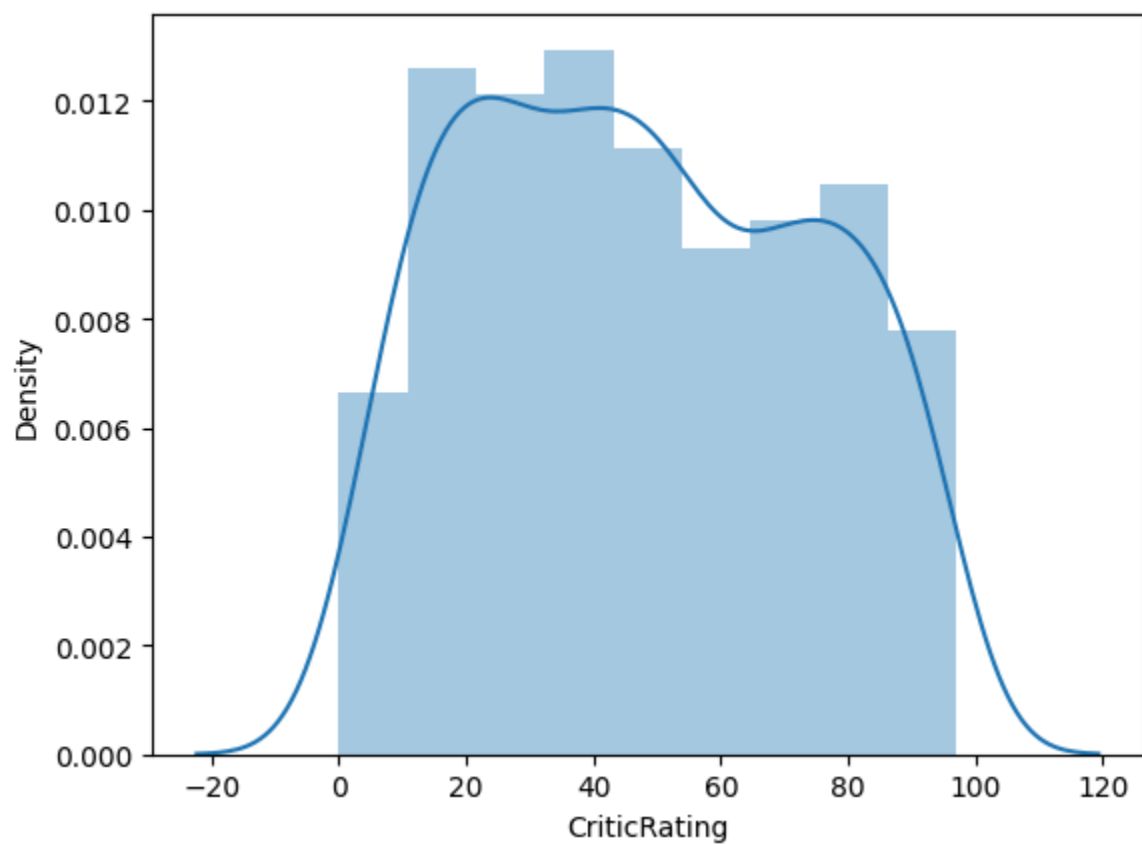
In [31]: `m1 = sns.displot(movies.CriticRating) #histogram`
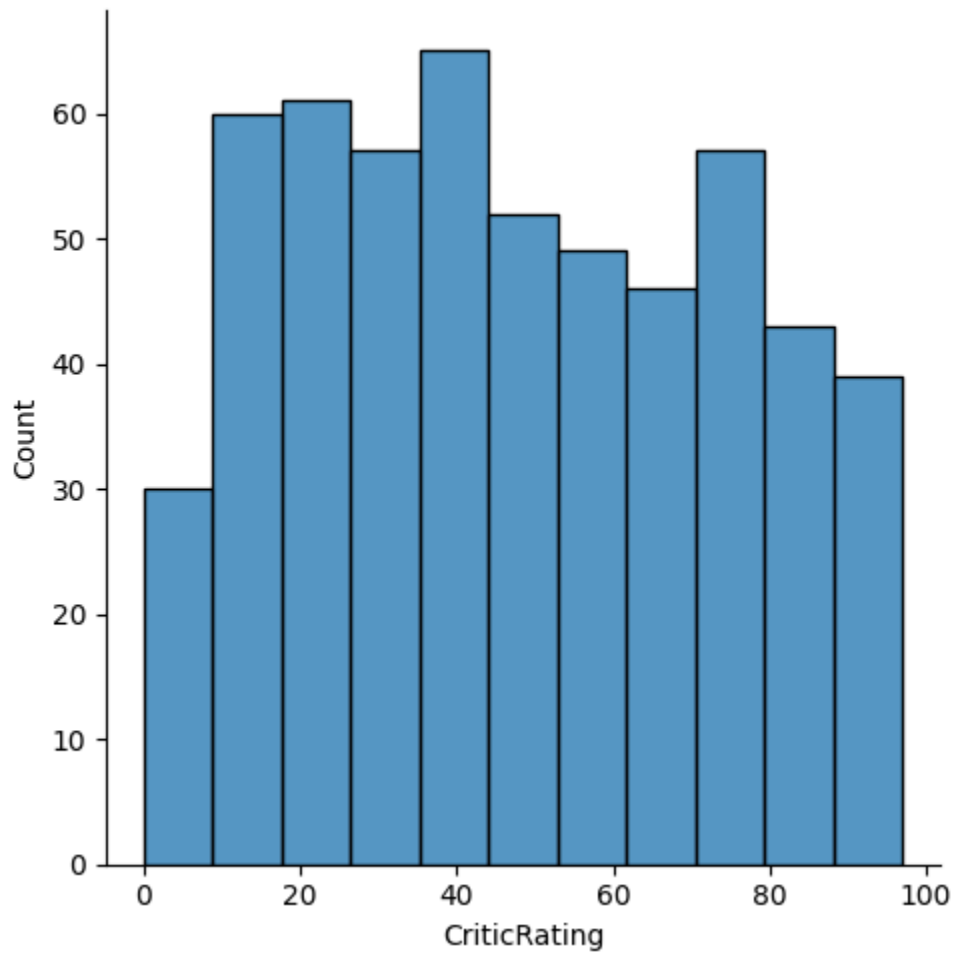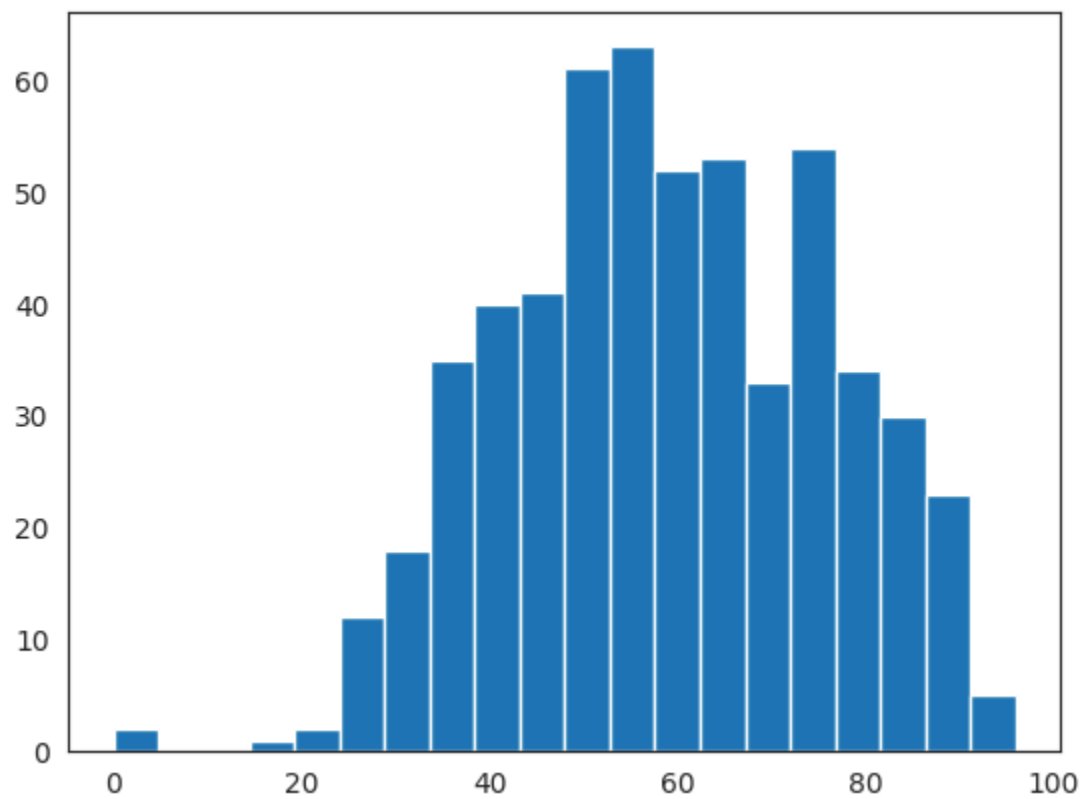

In [32]: `sns.set_style('darkgrid')`

In [33]: `m2 = sns.distplot(movies.AudienceRating, bins = 15)`
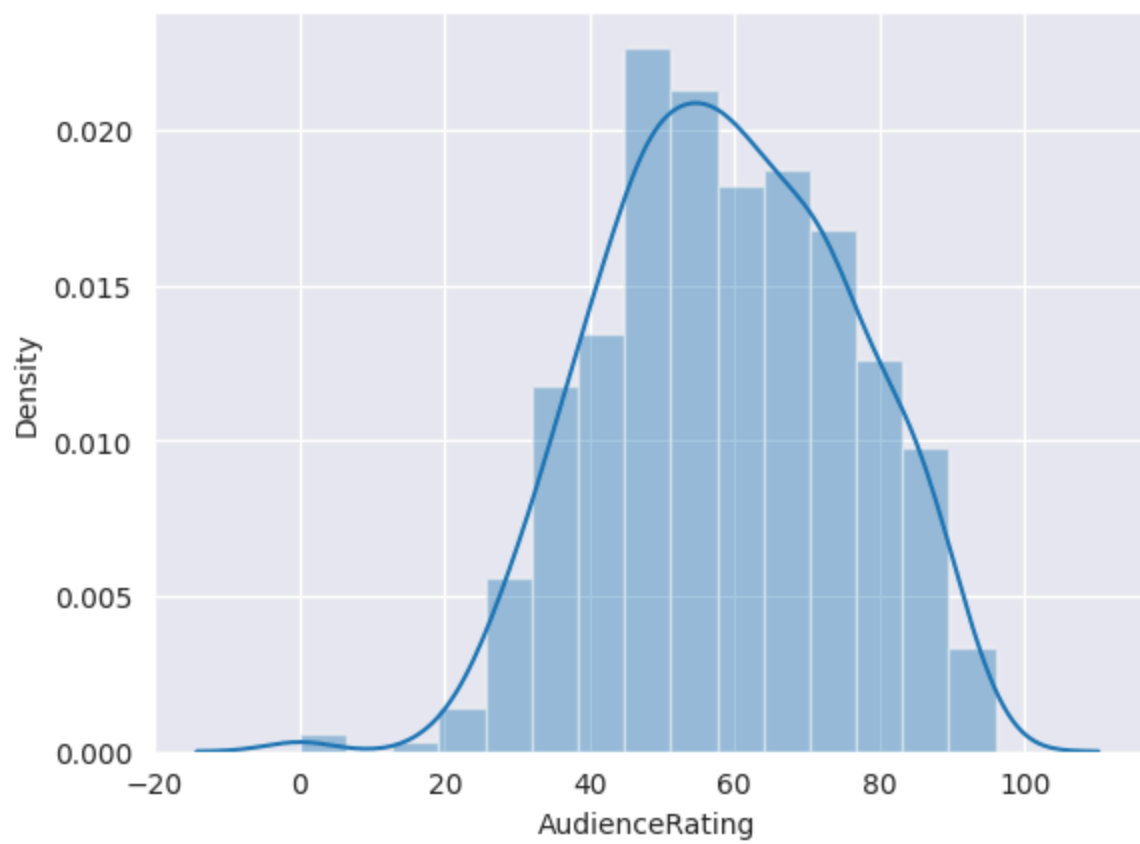
```
In [34]: sns.set_style('white')
         m3 = sns.distplot(movies.AudienceRating, bins = 15)
```
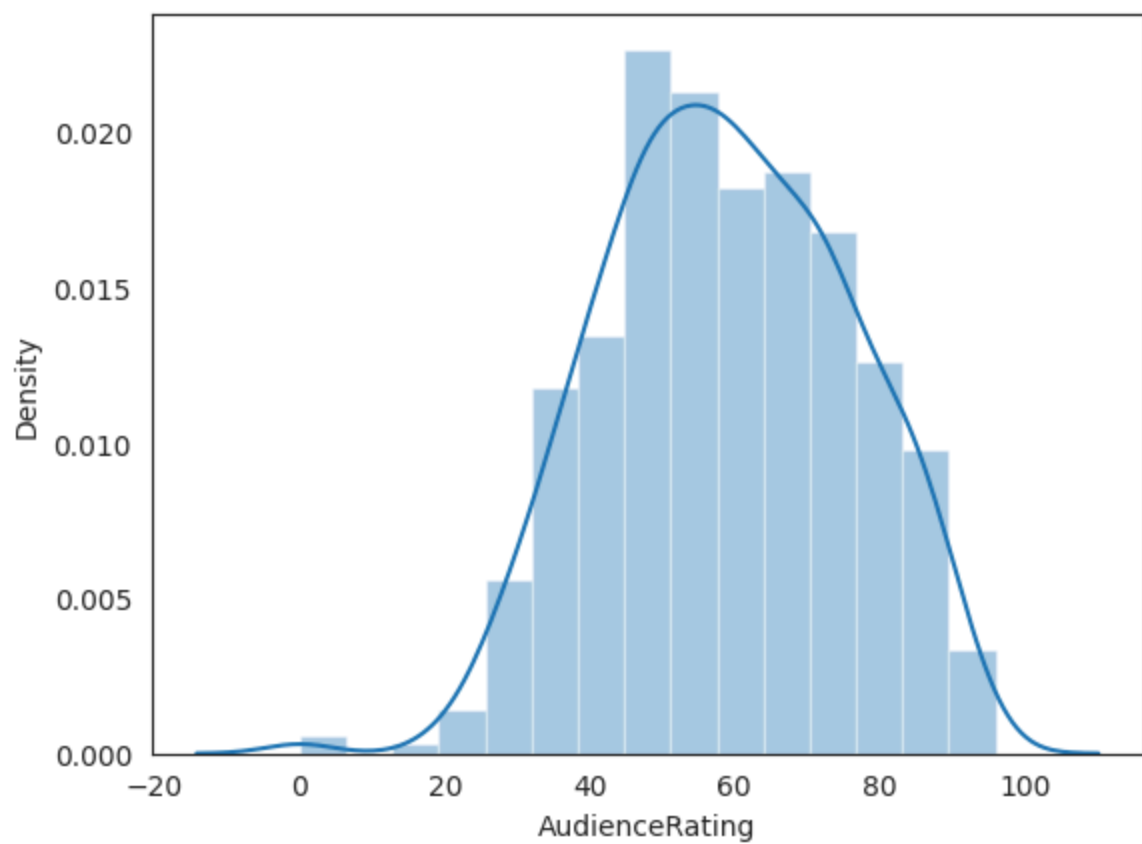
In [37]: `n1 = plt.hist(movies.AudienceRating, bins = 20)`

In [42]: `n1 = plt.hist(movies.CriticRating, bins=20) #uniform distribution`
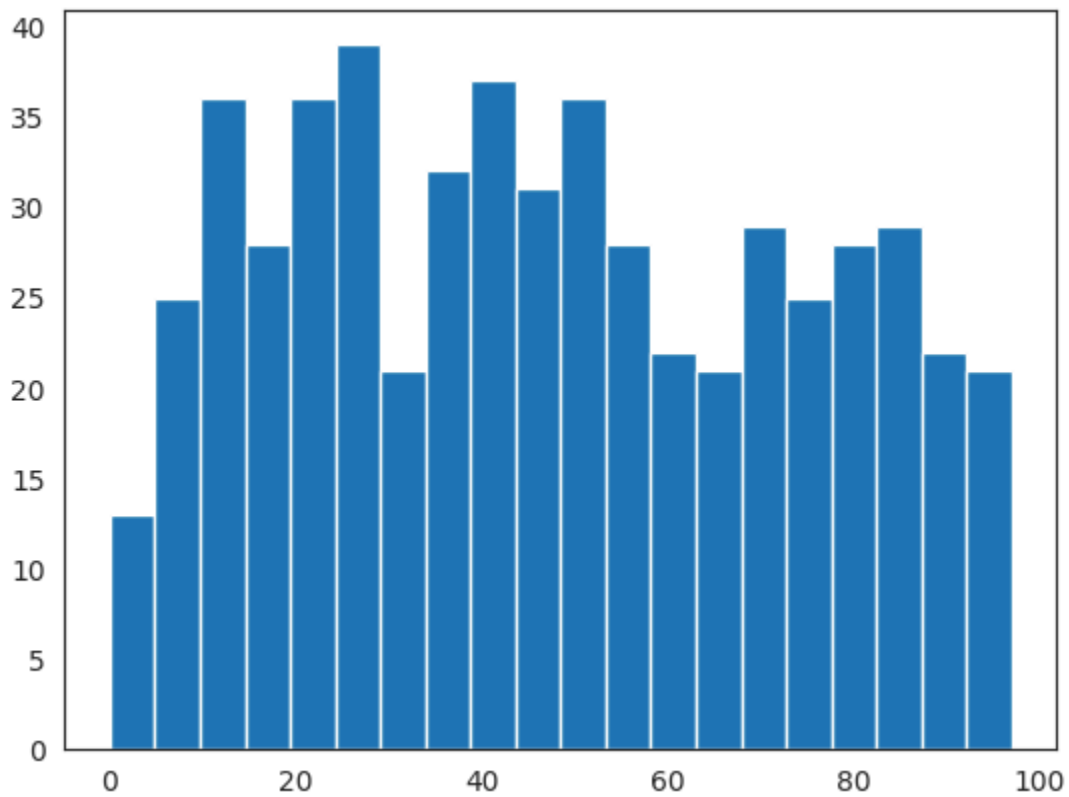
In [38]:
```
plt.hist(movies.BudgetMillions)
plt.show() #budgets is increse and movies are less
```
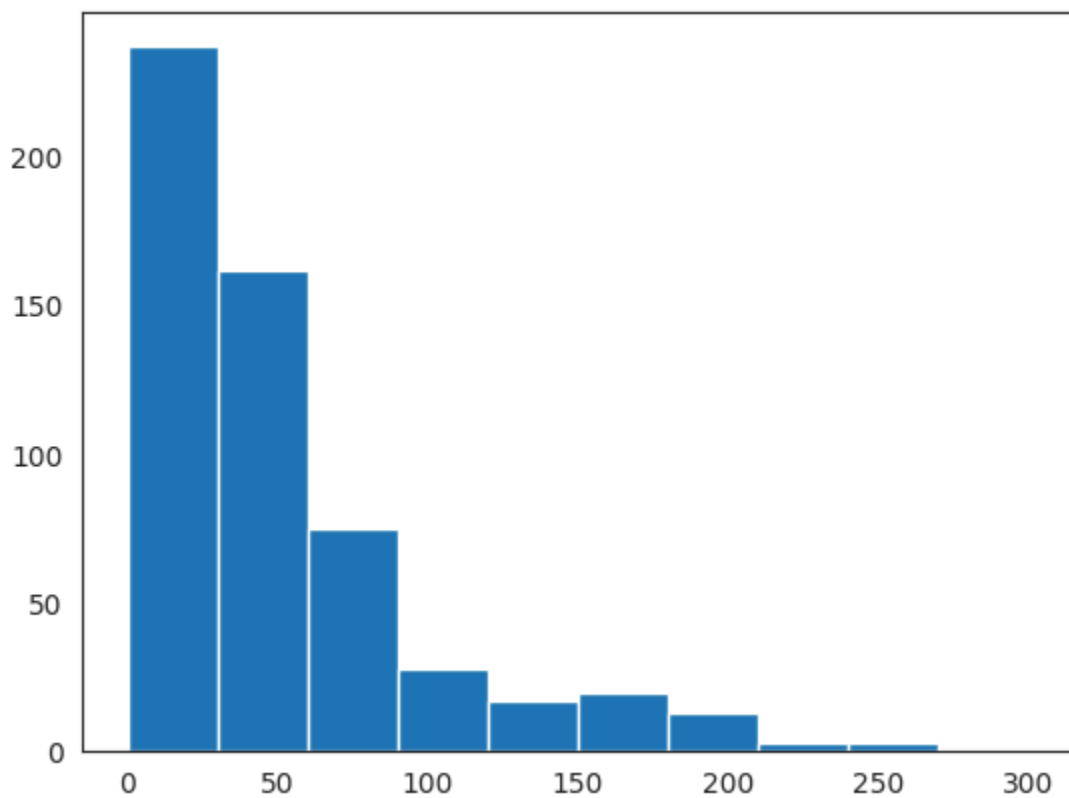
In [43]:
```python
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions)
plt.show()
```
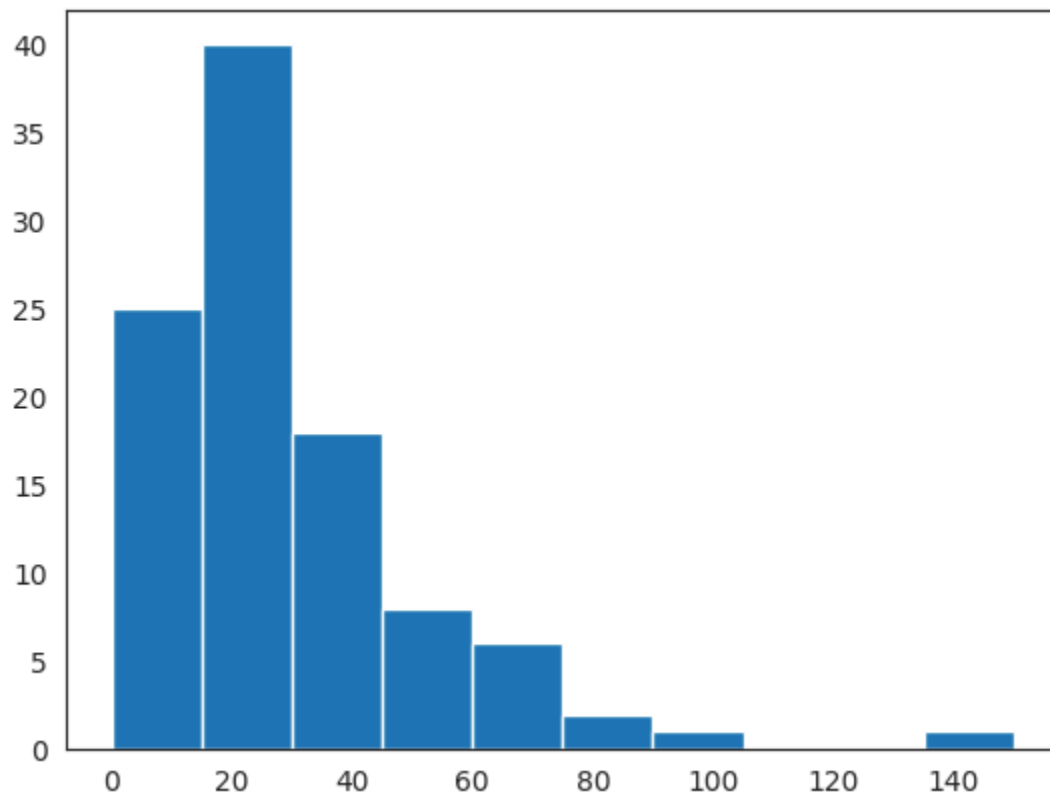

In [39]:
```python
vis1 = sns.lmplot(data = movies, x ='CriticRating', y = 'AudienceRating', fit_
```
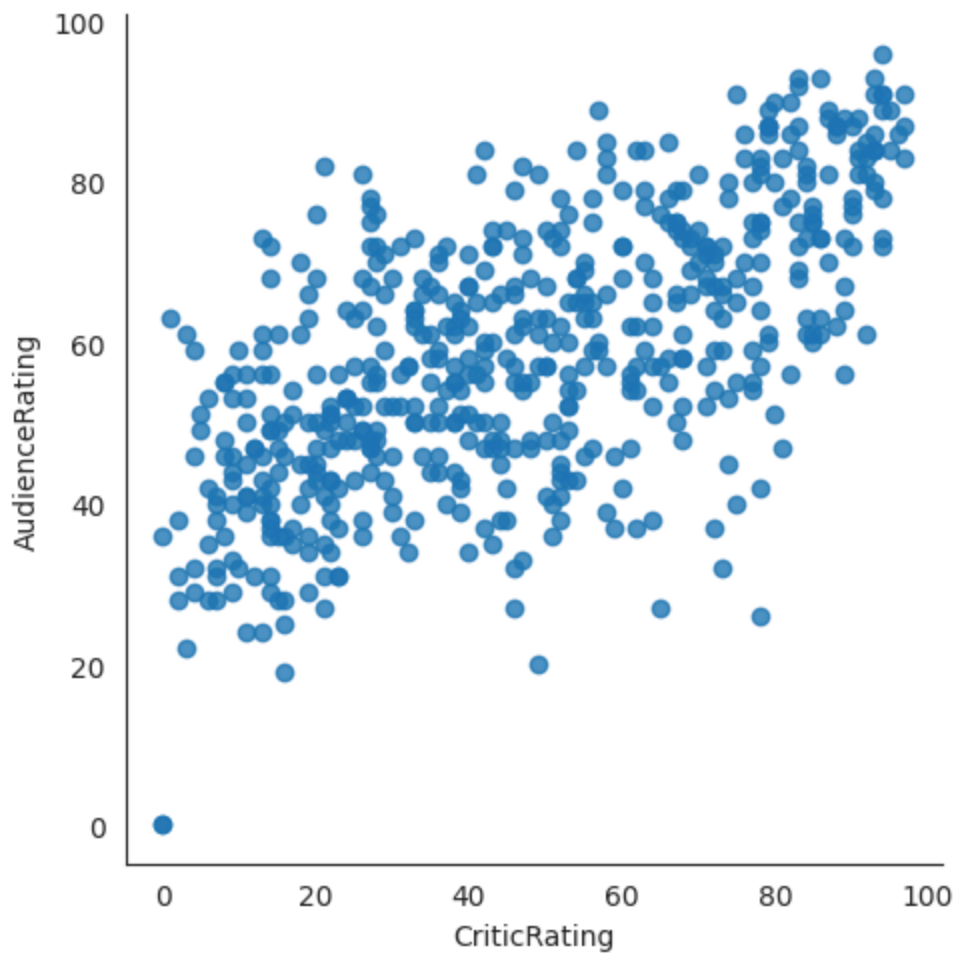
In [44]: # Below plots are stacked histogram becuase overlaped

```python
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()
```
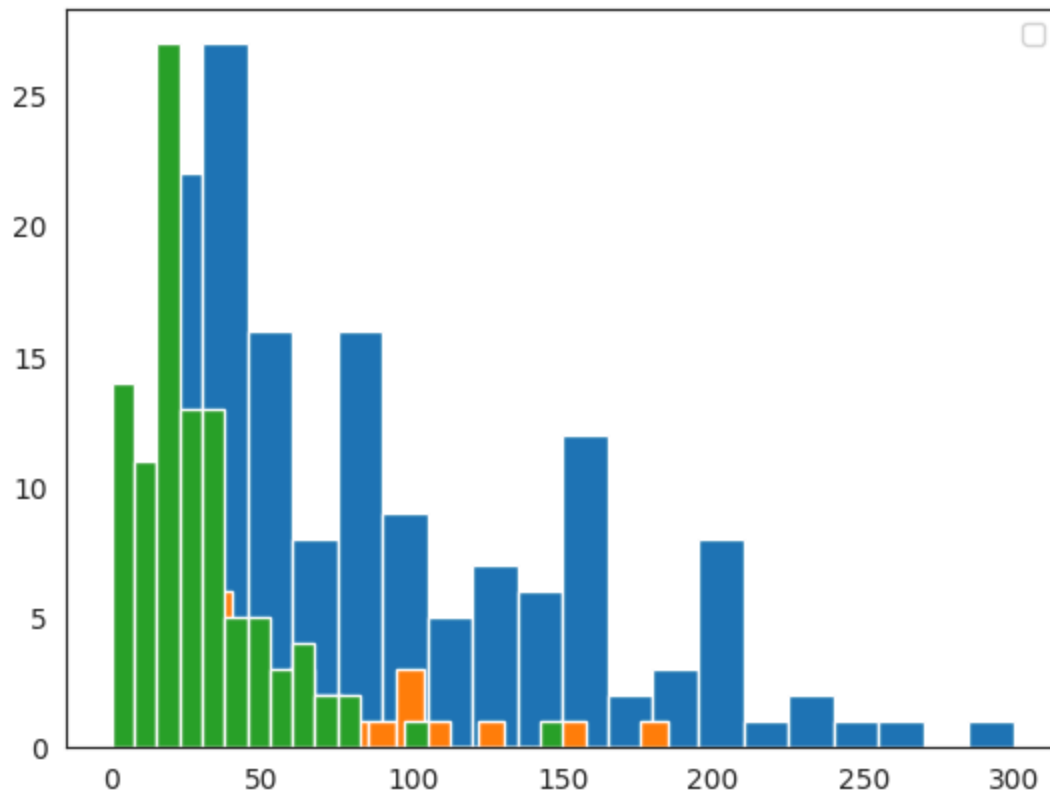
```
In [45]: plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,\
                    movies[movies.Genre == 'Drama'].BudgetMillions, \
                    movies[movies.Genre == 'Thriller'].BudgetMillions, \
                    movies[movies.Genre == 'Comedy'].BudgetMillions],
                   bins = 20, stacked = True)
         plt.show()
```
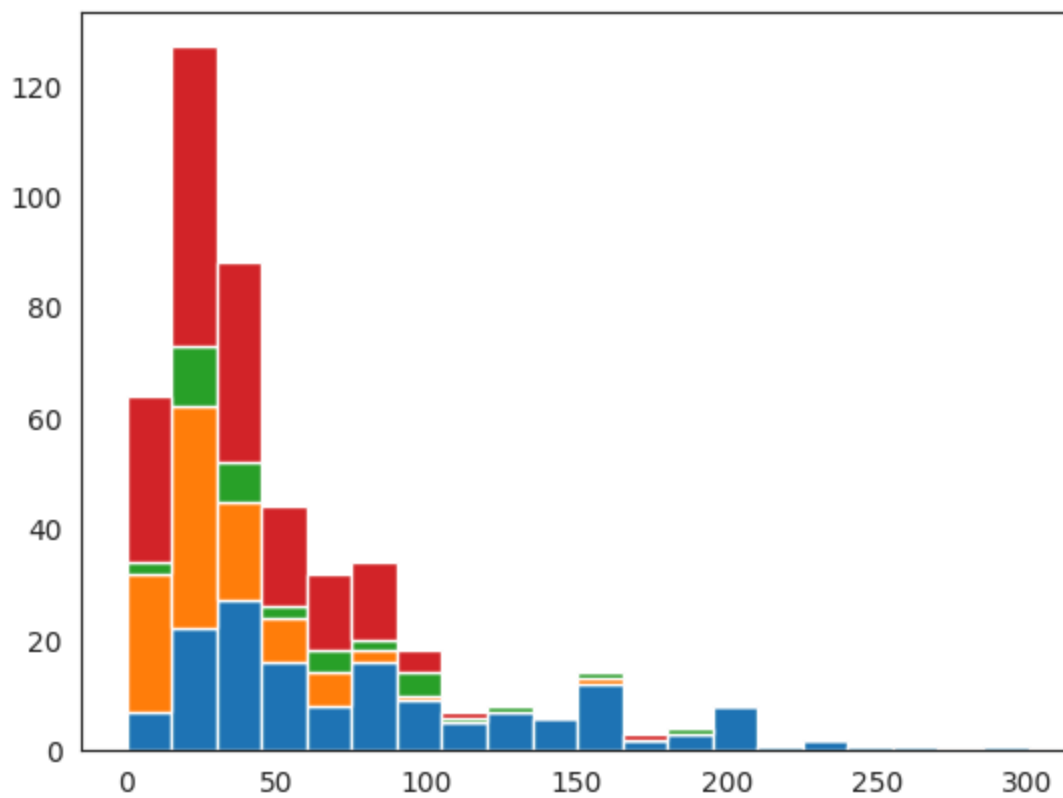
```
for gen in movies.Genre.cat.categories:
    print(gen)
```

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

```
vis2 = sns.lmplot(data=movies, x ='CriticRating', y='AudienceRating',fit_reg=T
```
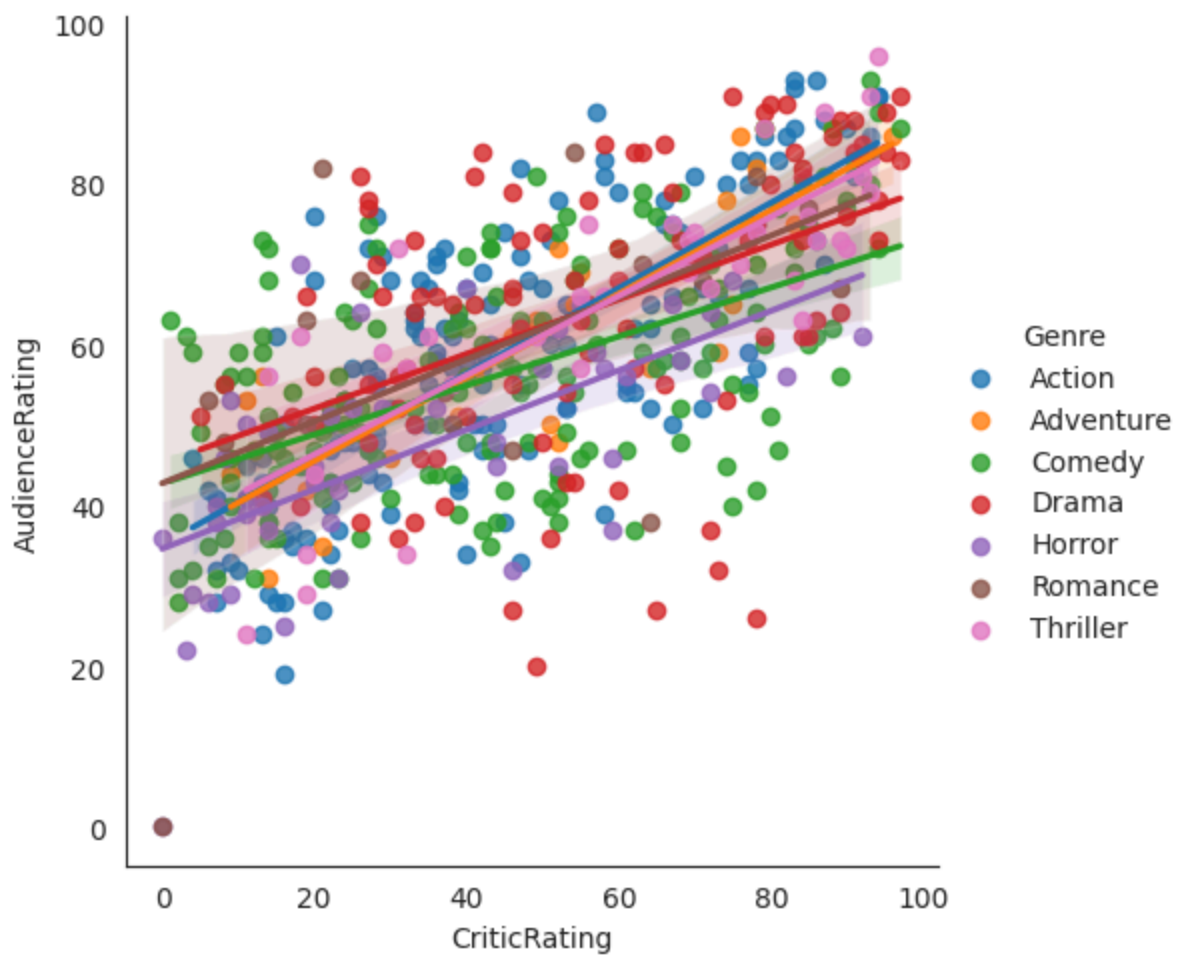
```
In [41]: vis2 = sns.lmplot(data=movies, x ='CriticRating', y='AudienceRating',fit_reg=F
```
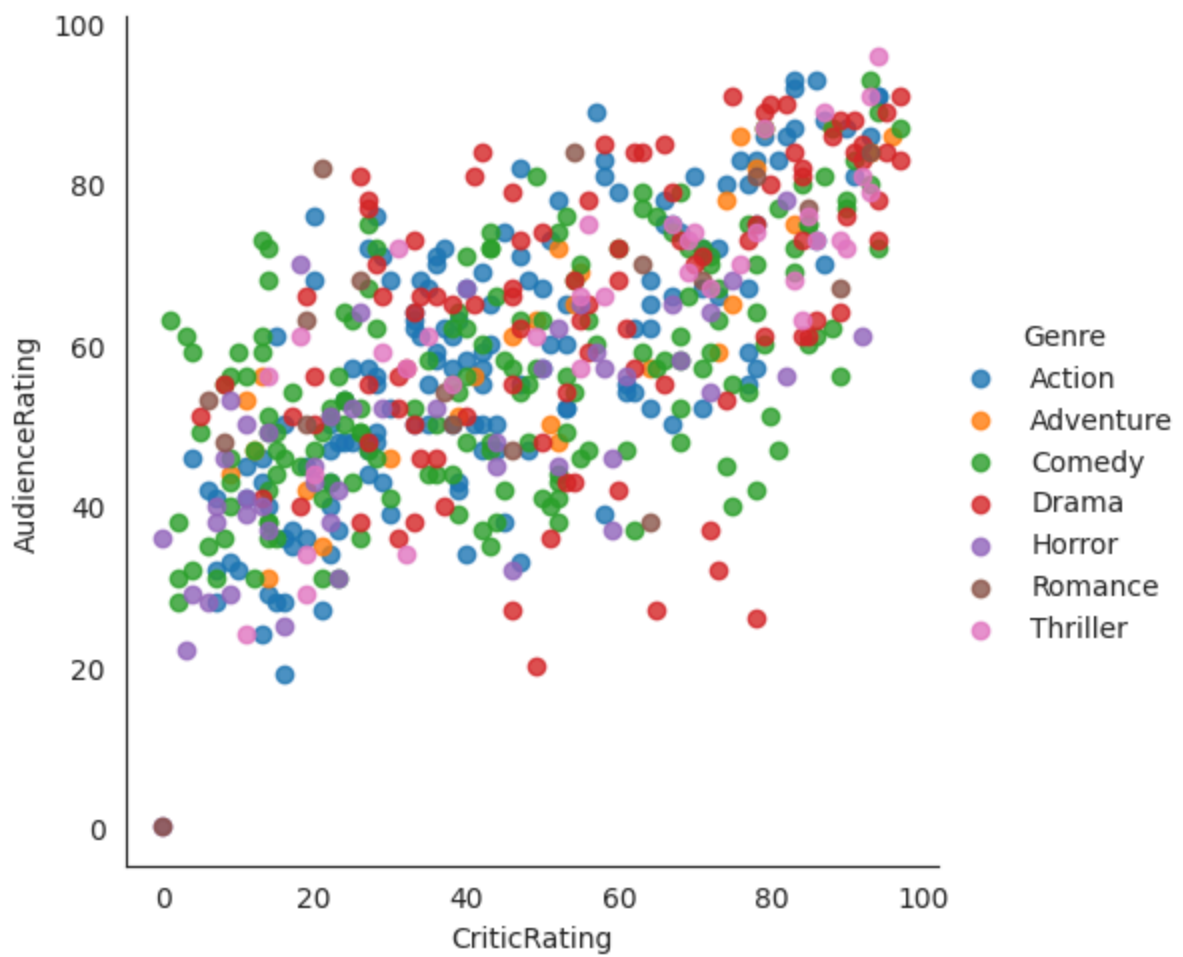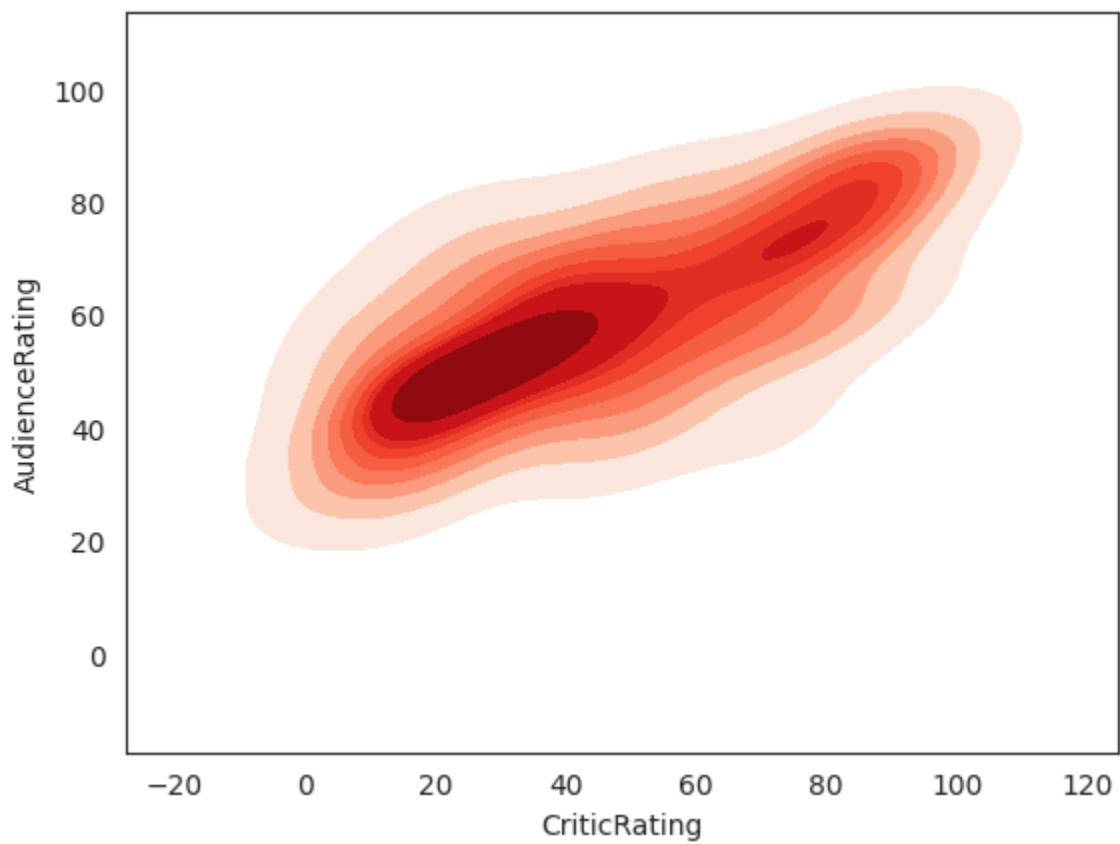
```
In [53]: vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating', fit_reg=F
```
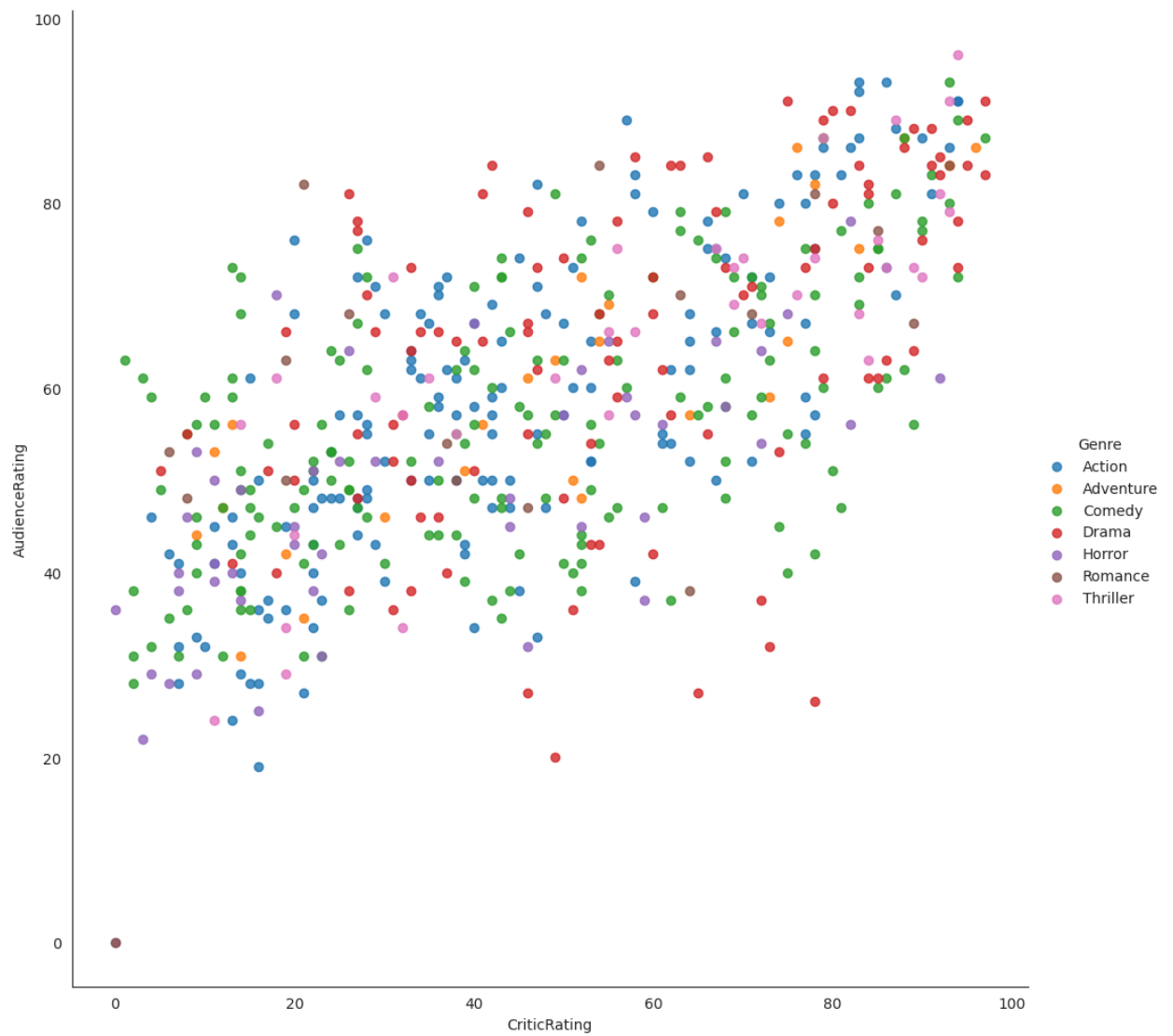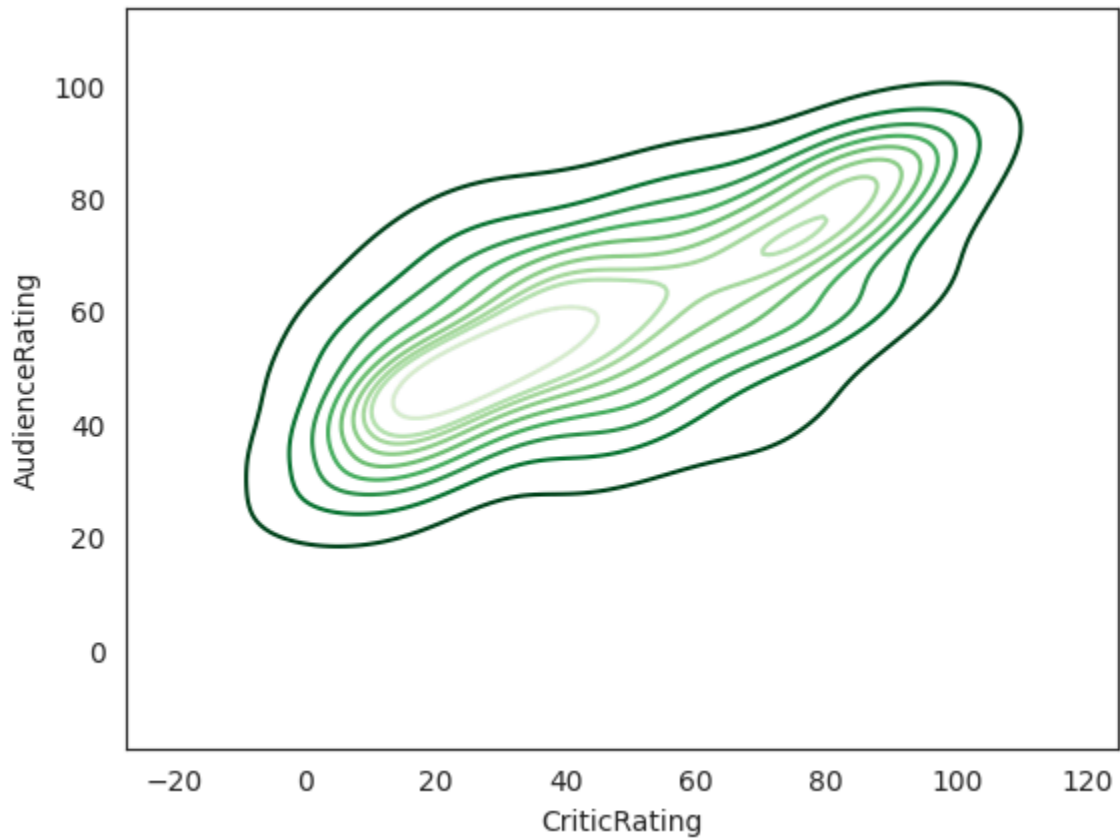
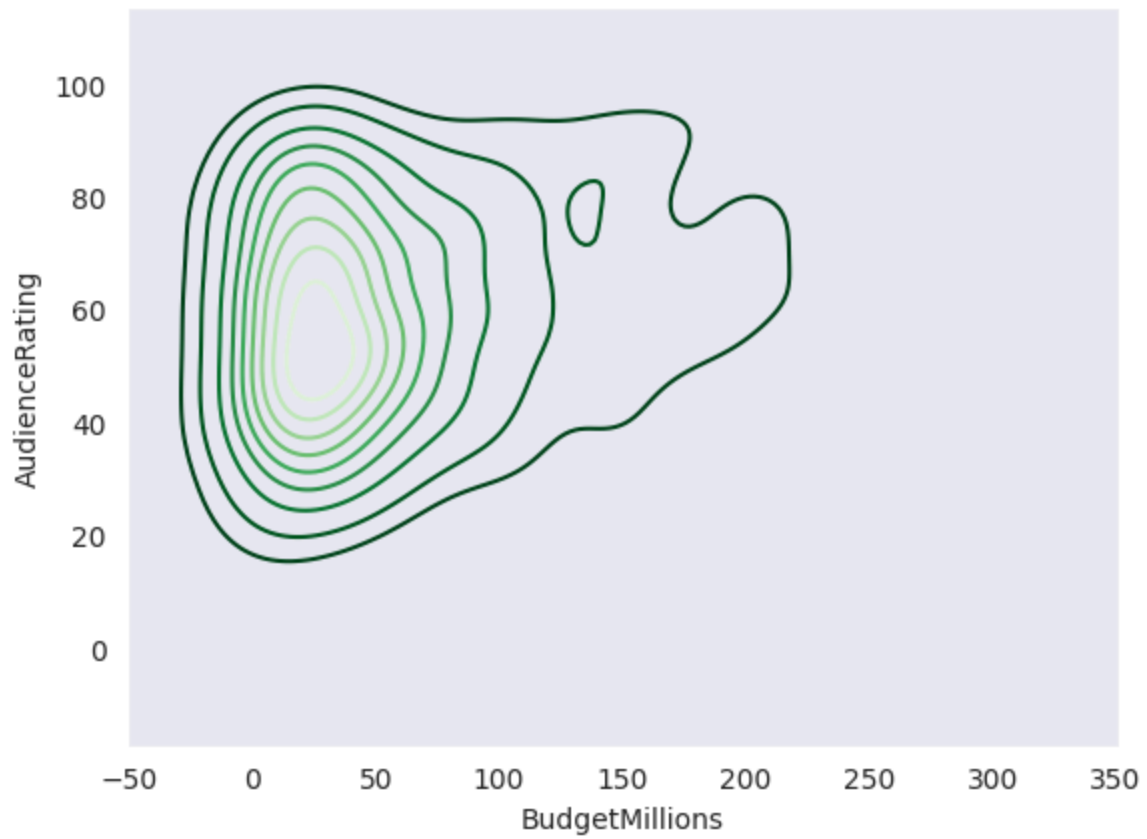In [55]: `k1 = sns.kdeplot(x=movies.CriticRating, y=movies.AudienceRating)`
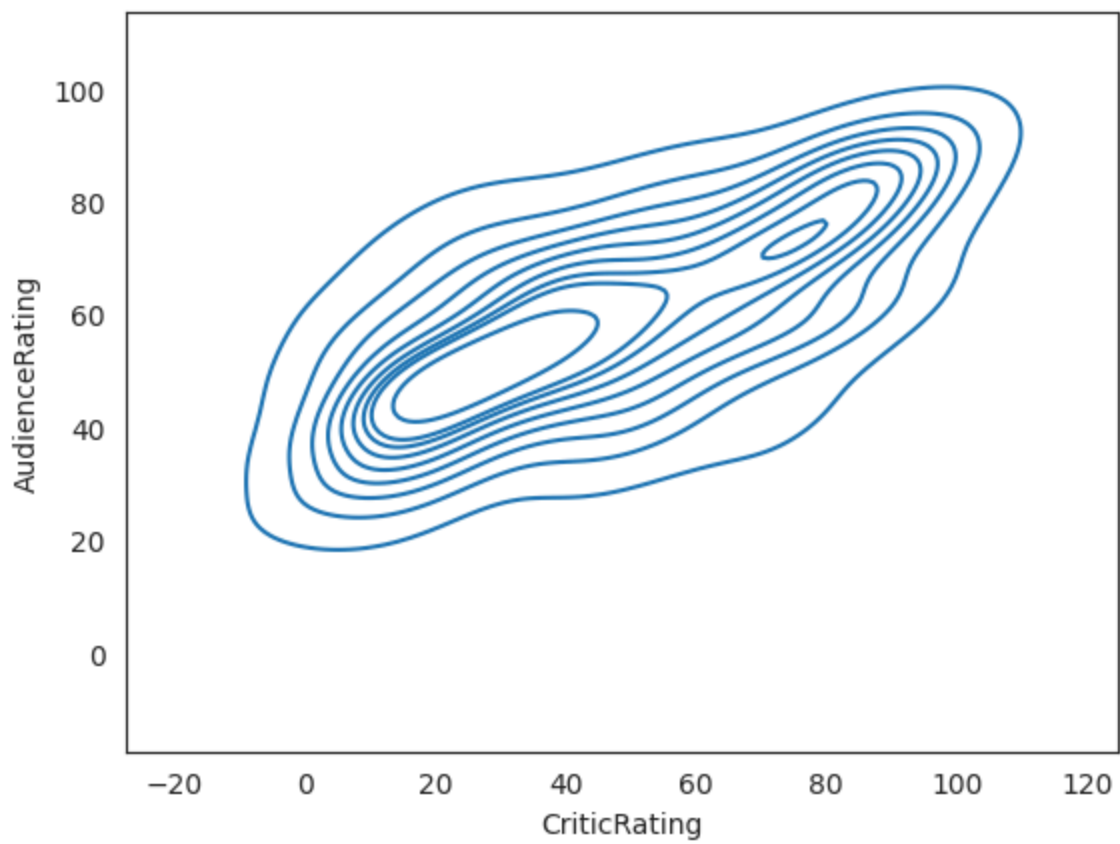
```
In [56]: k1 = sns.kdeplot(x=movies.CriticRating,y=movies.AudienceRating,shade = True,sh
```
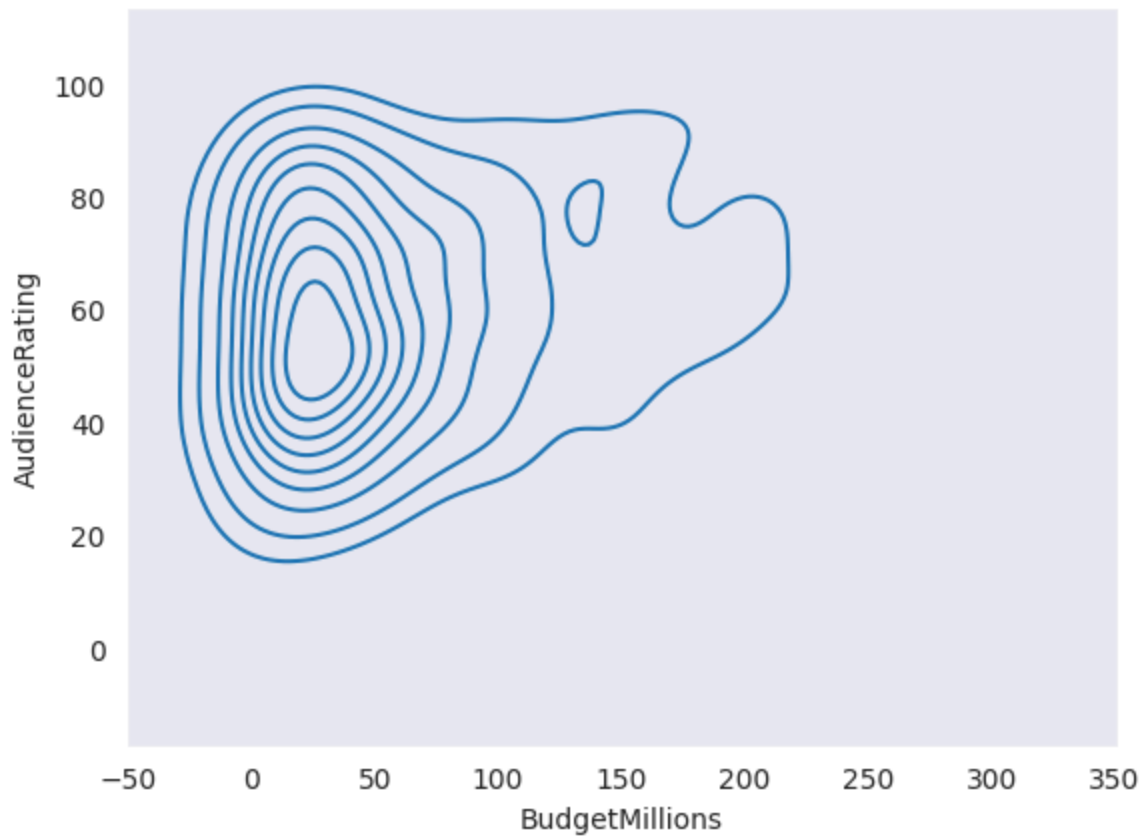
```
In [57]: k2 = sns.kdeplot(x=movies.CriticRating,y=movies.AudienceRating,shade_lowest=Fa
```
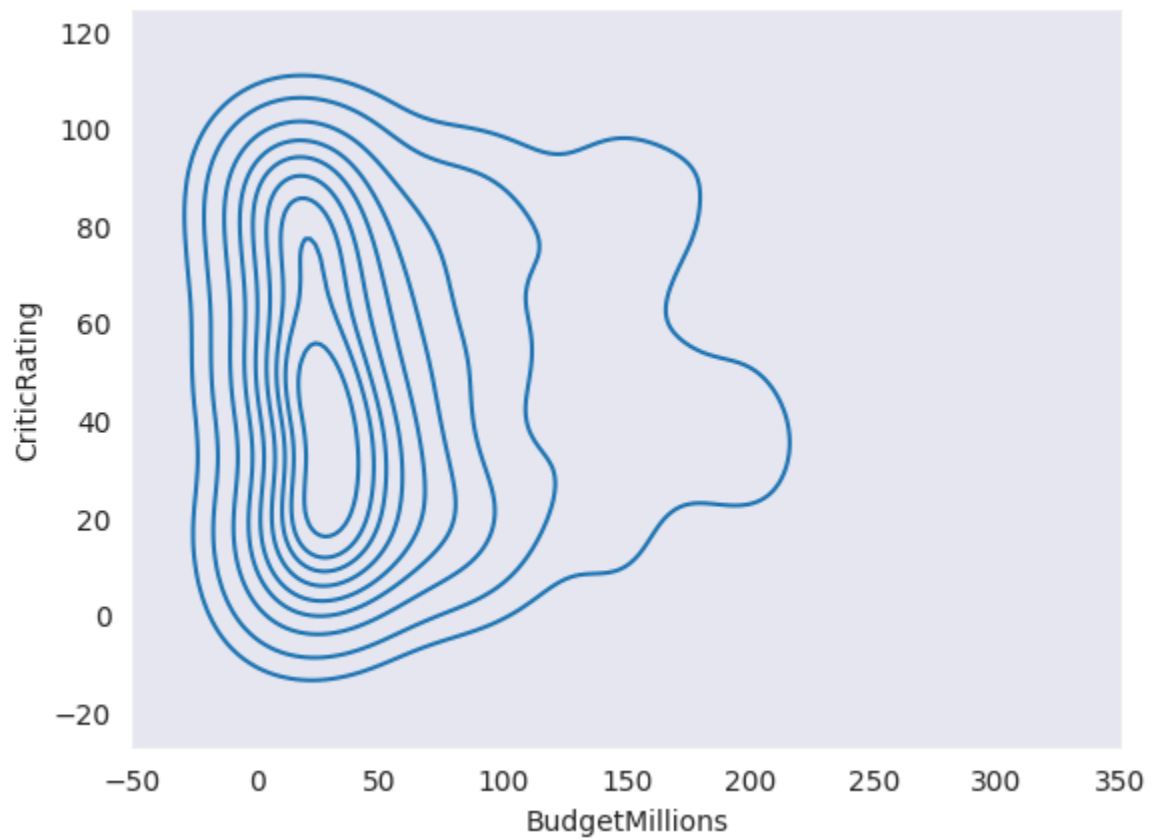


```
In [58]: sns.set_style('dark')
         k1 = sns.kdeplot(x=movies.BudgetMillions,y=movies.AudienceRating,shade_lowest=
```

```
In [59]: sns.set_style('dark')
         k1 = sns.kdeplot(x=movies.BudgetMillions,y=movies.AudienceRating)
```
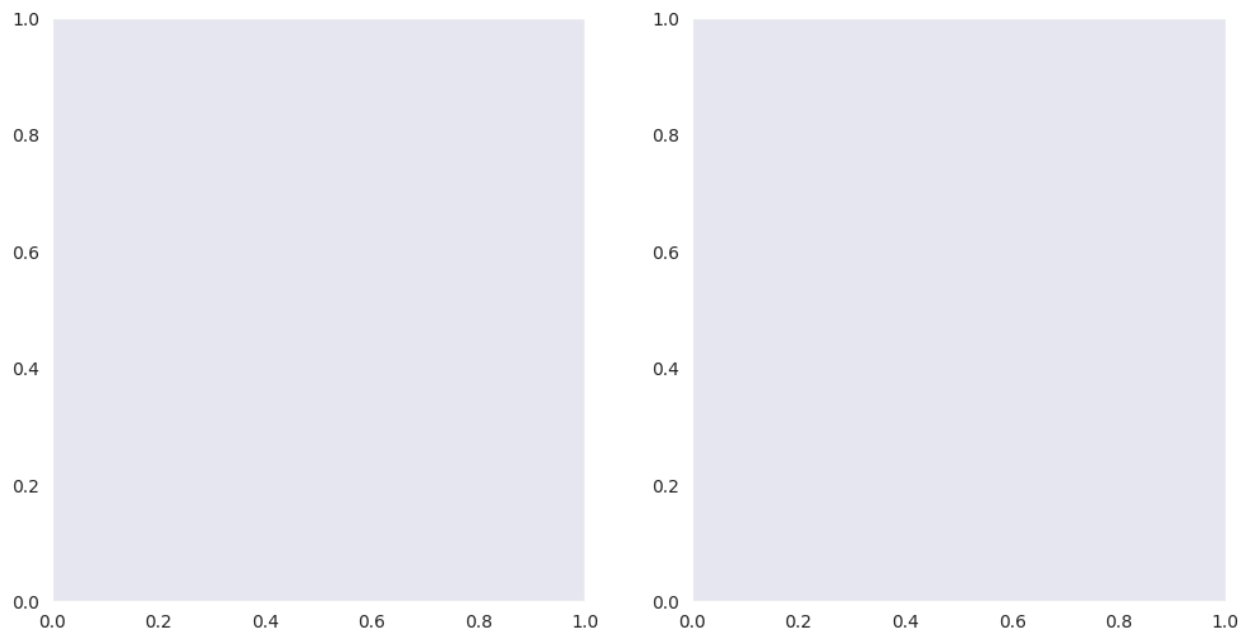
```
k2 = sns.kdeplot(x=movies.BudgetMillions,y=movies.CriticRating)
```
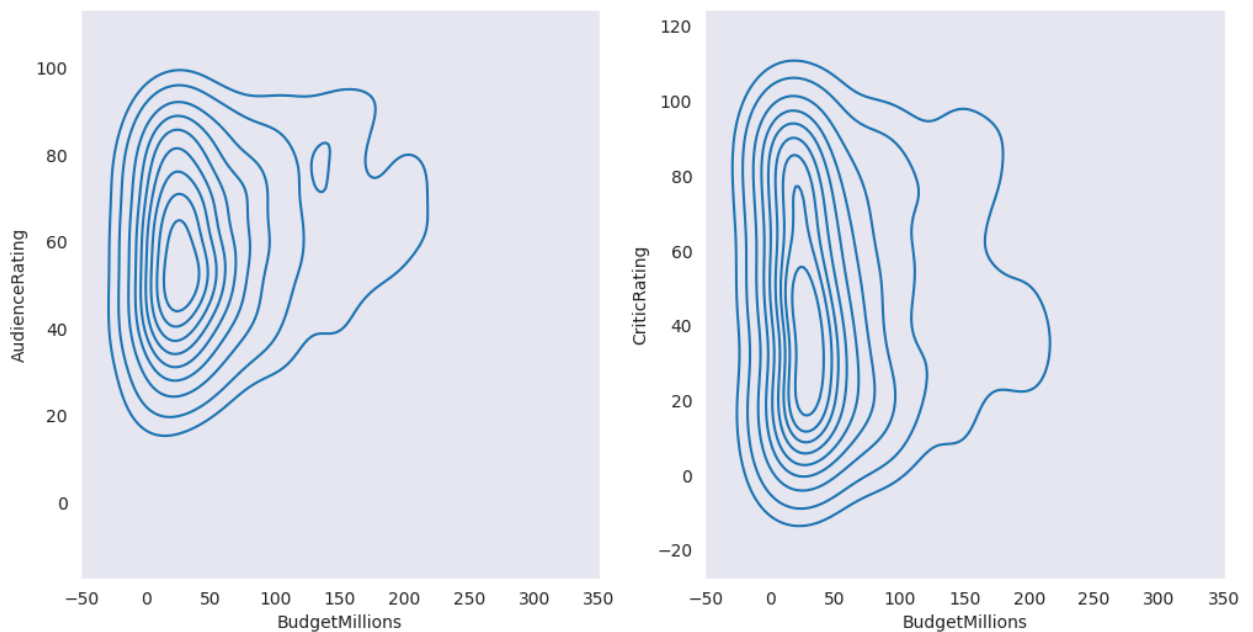
`#subplots`

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

```python
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])
```
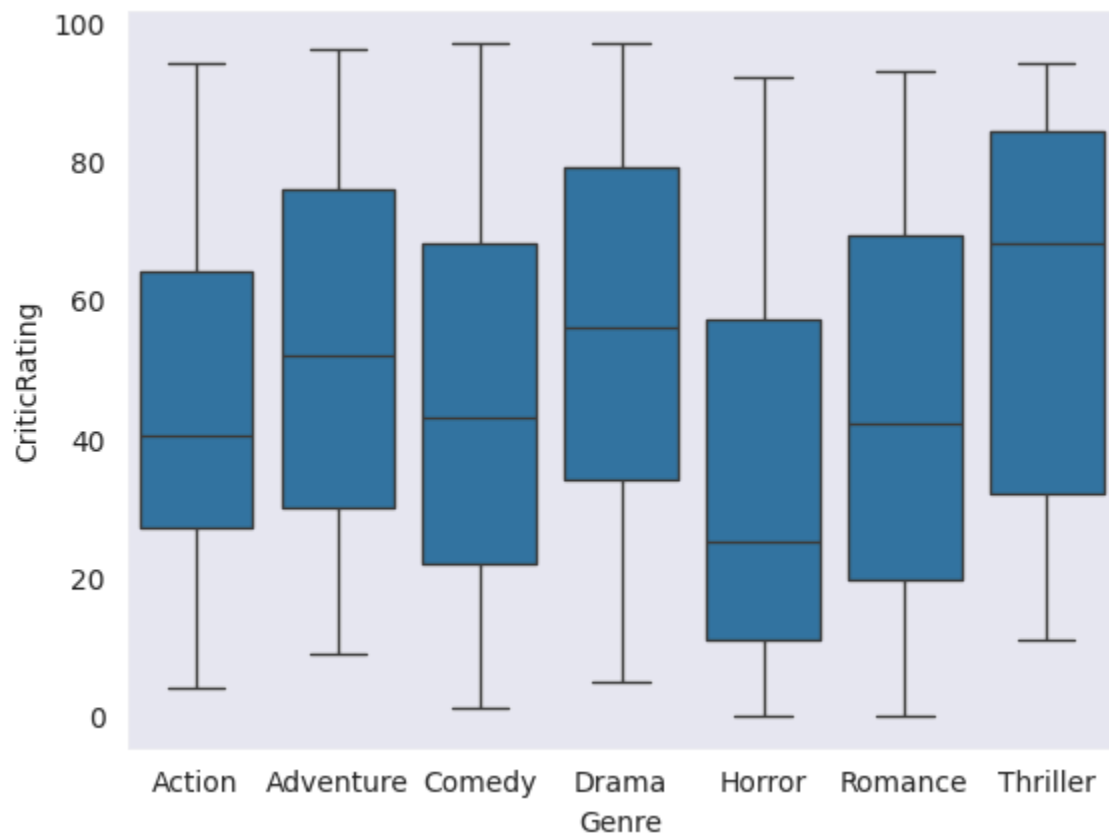

`axes`

```
array([<Axes: xlabel='BudgetMillions', ylabel='AudienceRating'>,
       <Axes: xlabel='BudgetMillions', ylabel='CriticRating'>],
      dtype=object)
```
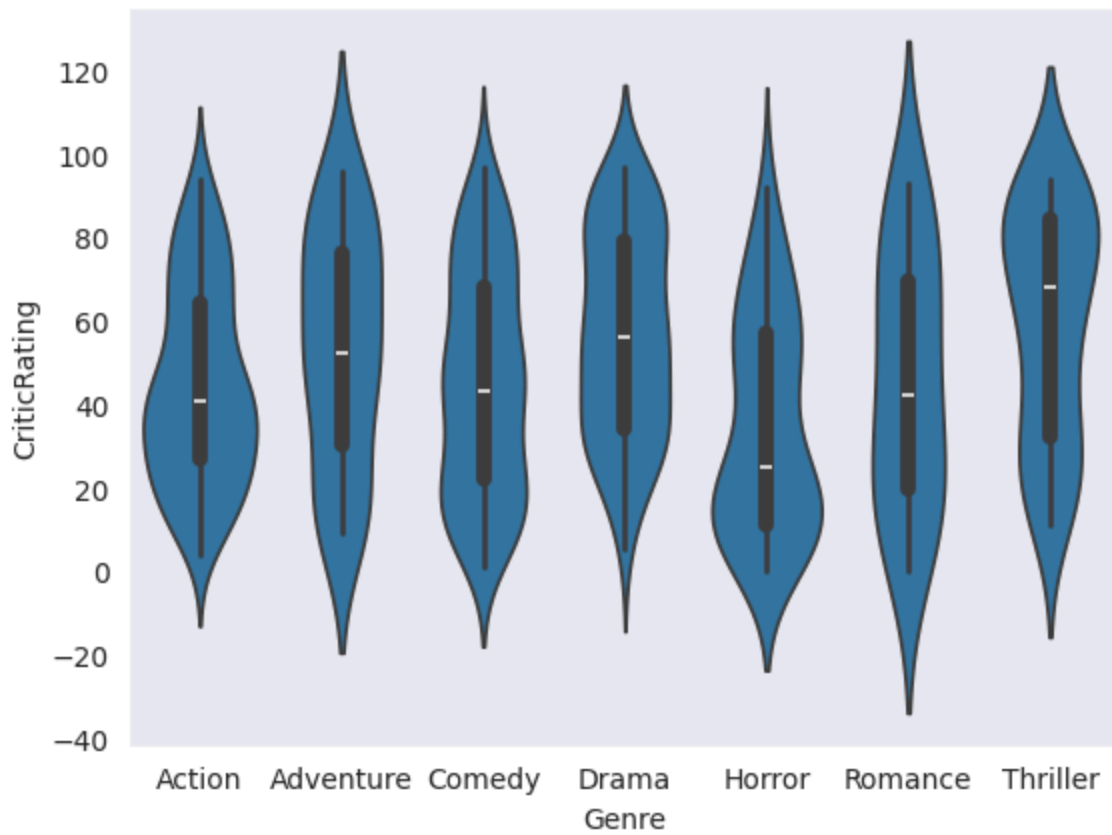
In [64]: *#Box plots -*

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


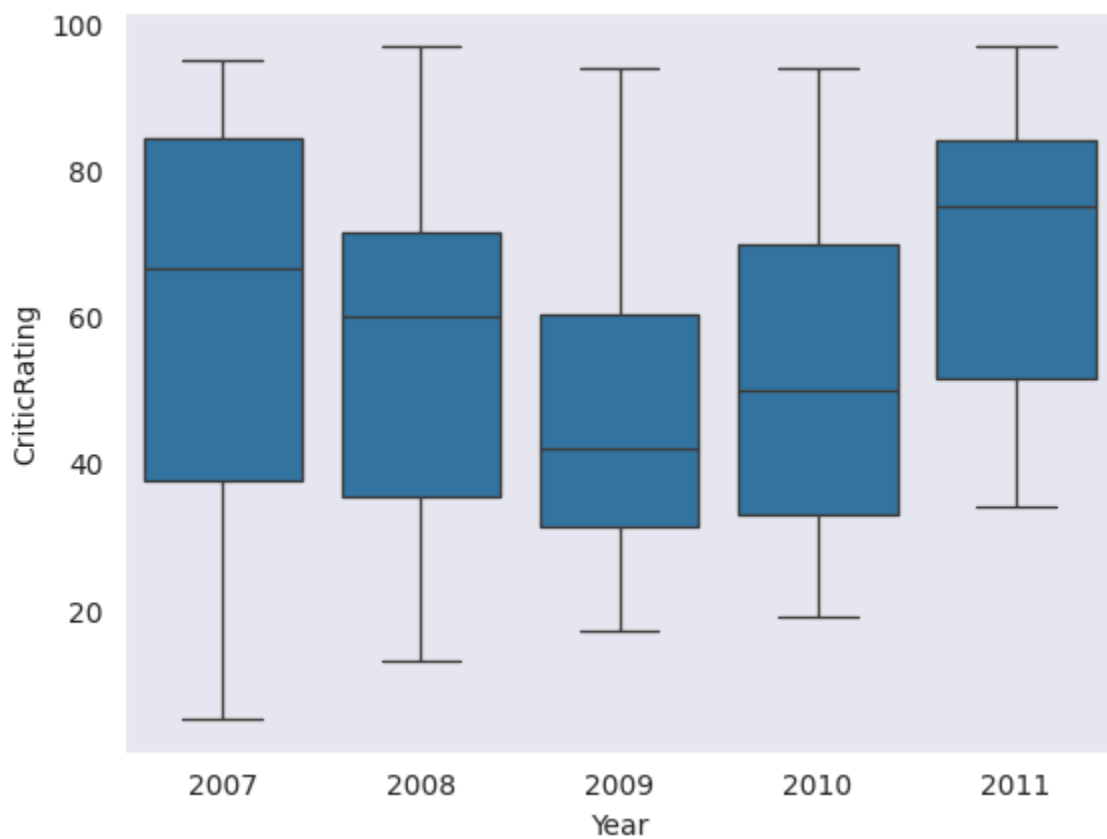In [65]: *#violin plot*

```python
z = sns.violinplot(data=movies, x='Genre', y = 'CriticRating')
```
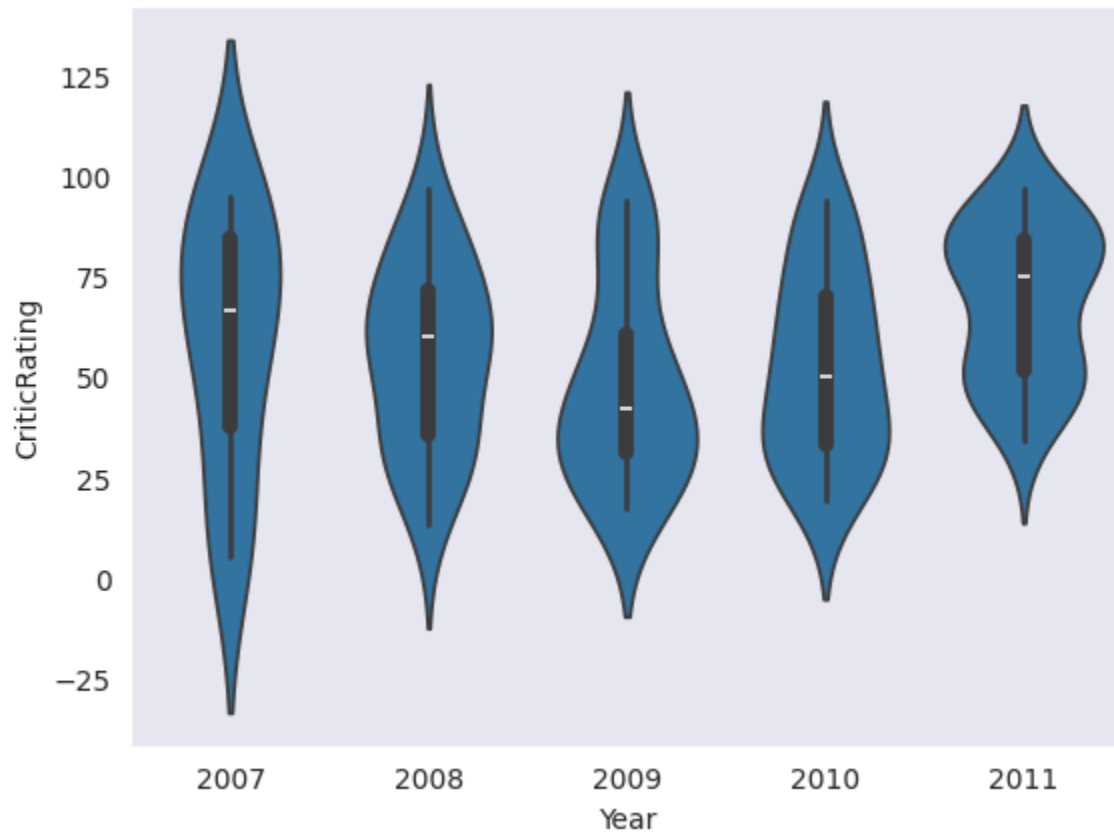
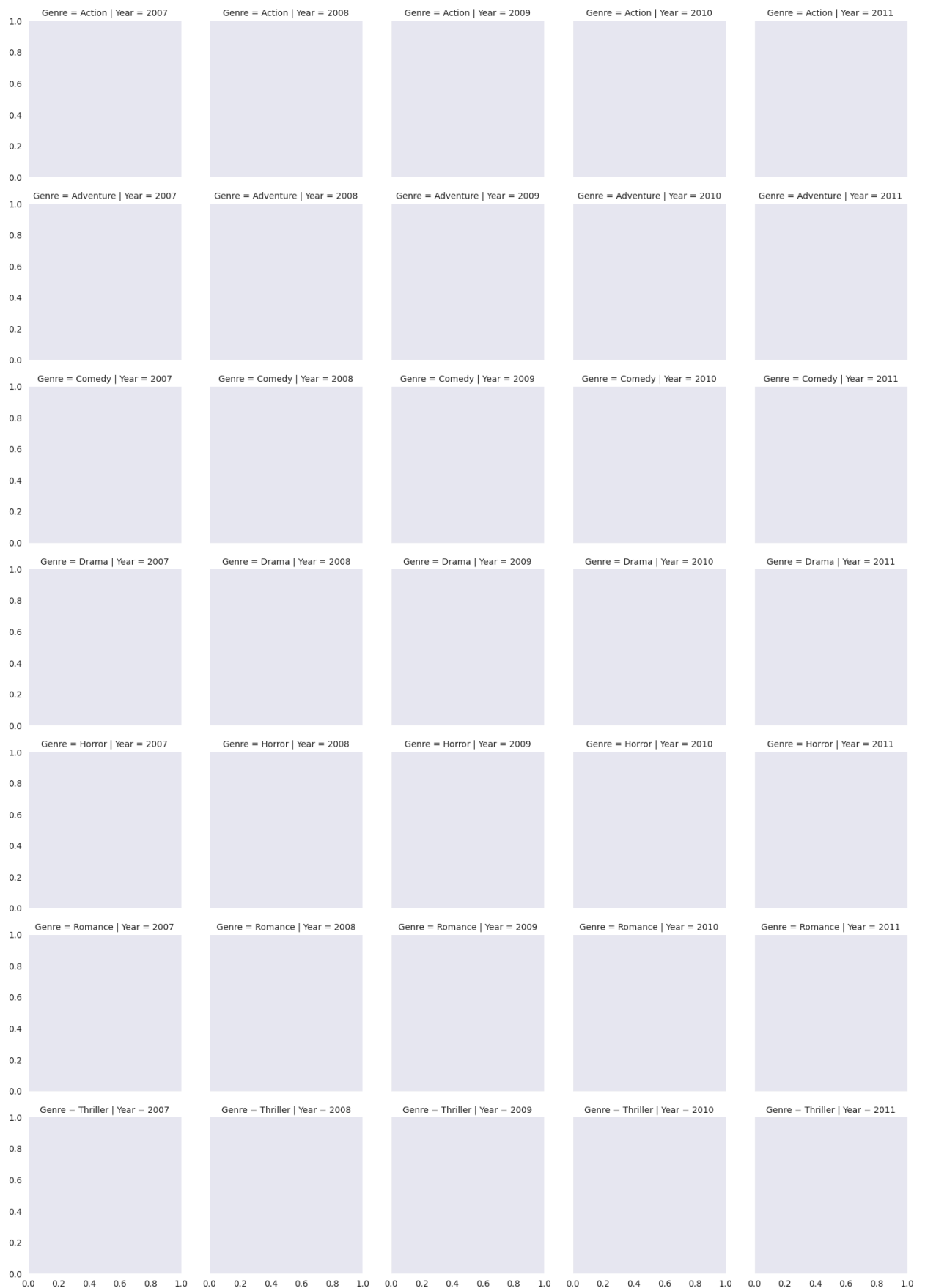In [66]: `w1 = sns.boxplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRa`

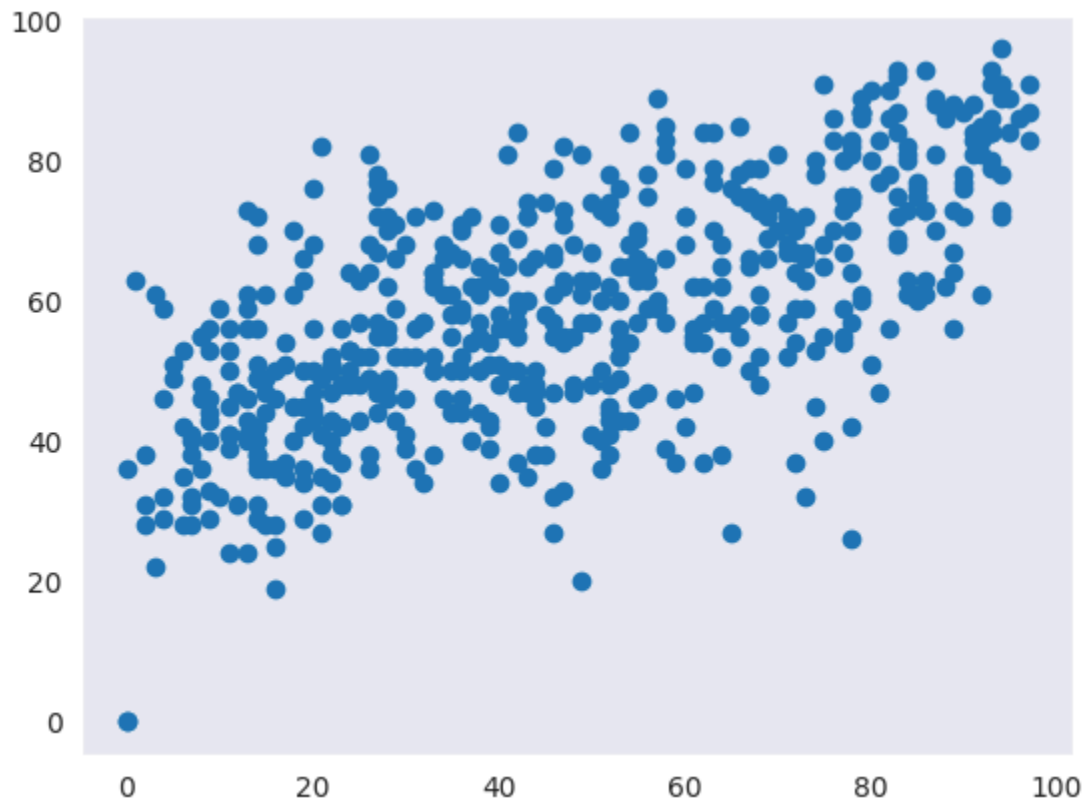In [67]: `z = sns.violinplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'Critic`

Createing a Facet grid

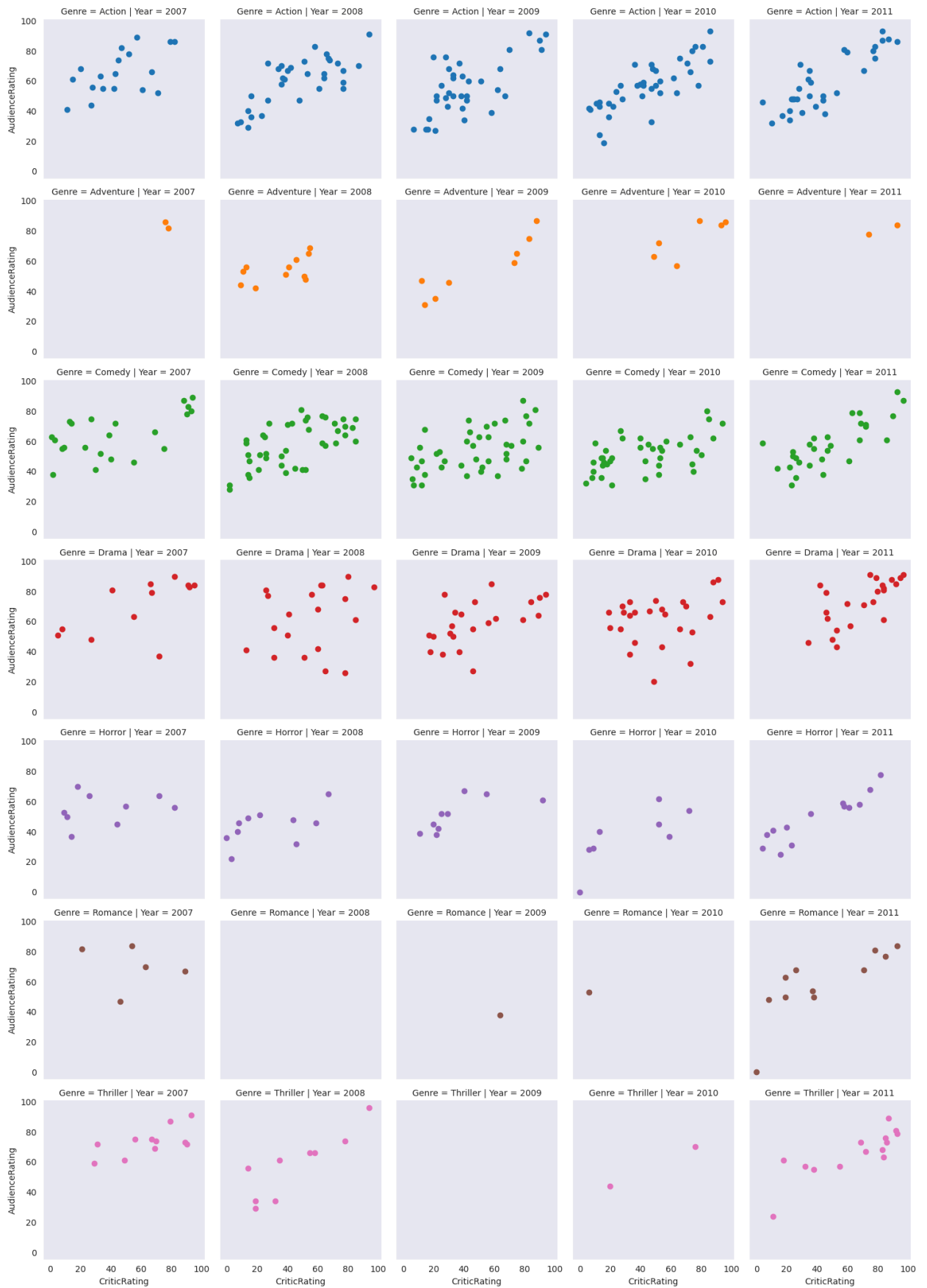In [68]: `g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre') #kind of`

Genre = Action | Year = 2007	Genre = Action | Year = 2008	Genre = Action | Year = 2009	Genre = Action | Year = 2010	Genre = Action | Year = 2011
Genre = Adventure | Year = 2007	Genre = Adventure | Year = 2008	Genre = Adventure | Year = 2009	Genre = Adventure | Year = 2010	Genre = Adventure | Year = 2011
Genre = Comedy | Year = 2007	Genre = Comedy | Year = 2008	Genre = Comedy | Year = 2009	Genre = Comedy | Year = 2010	Genre = Comedy | Year = 2011
Genre = Drama | Year = 2007	Genre = Drama | Year = 2008	Genre = Drama | Year = 2009	Genre = Drama | Year = 2010	Genre = Drama | Year = 2011
Genre = Horror | Year = 2007	Genre = Horror | Year = 2008	Genre = Horror | Year = 2009	Genre = Horror | Year = 2010	Genre = Horror | Year = 2011
Genre = Romance | Year = 2007	Genre = Romance | Year = 2008	Genre = Romance | Year = 2009	Genre = Romance | Year = 2010	Genre = Romance | Year = 2011
Genre = Thriller | Year = 2007	Genre = Thriller | Year = 2008	Genre = Thriller | Year = 2009	Genre = Thriller | Year = 2010	Genre = Thriller | Year = 2011

In [69]:
```python
plt.scatter(movies.CriticRating,movies.AudienceRating)
```

```
In [70]: g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
         g = g.map(plt.scatter, 'CriticRating', 'AudienceRating' ) #scatterplots are ma
```
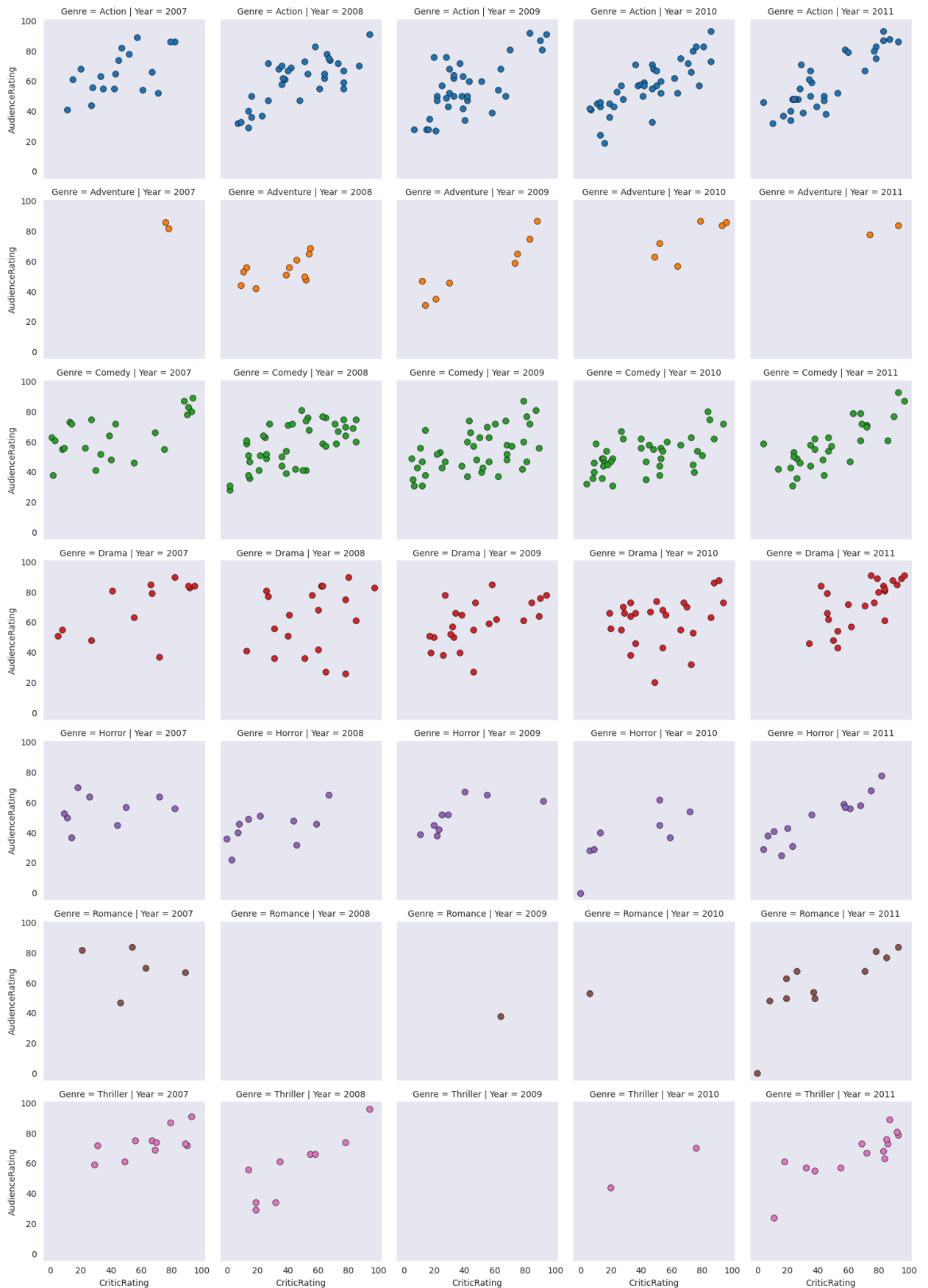
In [71]: # you can populated any type of chat.

```python
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.hist, 'BudgetMillions') #scatterplots are mapped in facetgrid
```

```
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
```

In [75]: # python is not vectorize programming language

```python
# 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 = 'CriticRa

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

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

plt.show()
```