FLIM Rating full EDA with SEABORN

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
Film = pd.read_csv(r"C:\Users\ratho\.ipynb_checkpoints\DATA\FilmRating.csv")
Film
```

Out[2]:

	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 [3]:
```
len(Film)
```

Out[3]:
```
559
```

In [4]:
```
Film.shape
```

Out[4]:
```
(559, 6)
```

In [5]:
```
type(Film)
```

Out[5]:
```
pandas.core.frame.DataFrame
```

In [6]:
```
Film.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]:
```
Film.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]:
```
Film.dtypes
```

Out[8]:
```
Film                        object
Genre                       object
Rotten Tomatoes Ratings %    int64
Audience Ratings %           int64
Budget (million $)           int64
Year of release              int64
dtype: object
```

In [9]:
```
Film.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]:
```
Film.describe()
# if you look at the year the data type is int but when you look at the mean value it
# we have to change to categroy type
# also from object datatype we will convert to category datatypes
```

Out[10]:

	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million $)	Year of release
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 [11]:
```
Film.columns
```

Out[11]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

In [12]:
```
Film.isnull()
```

Out[12]:

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million $)	Year of release
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
554	False	False	False	False	False	False
555	False	False	False	False	False	False
556	False	False	False	False	False	False
557	False	False	False	False	False	False
558	False	False	False	False	False	False

559 rows × 6 columns

In [13]:
```
Film.isnull().sum()
```

Out[13]:
```
Film                        0
Genre                       0
Rotten Tomatoes Ratings %   0
Audience Ratings %          0
Budget (million $)          0
Year of release             0
dtype: int64
```

In [14]:
```
Film.columns = ['Film','Genre', 'CriticRating', 'AudienceRatings',
       'BudgetMillion', 'Year']
Film
```

Out[14]:

	Film	Genre	CriticRating	AudienceRatings	BudgetMillion	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 [15]:
```
Film.head(1)
```

Out[15]:

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

In [16]:
```
# Changing the objects into category datatype

Film.Year = Film.Year.astype('category')
Film.Film = Film.Film.astype('category')
Film.Genre = Film.Genre.astype('category')
```

In [17]:
```
Film.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   AudienceRatings  559 non-null    int64
 4   BudgetMillion    559 non-null    int64
 5   Year             559 non-null    category
dtypes: category(3), int64(3)
memory usage: 36.5 KB
```

In [18]:
```
Film['Year']
```

```
# is it real no. year you can take average,min,max but out come have no meaning
```

Out[18]:
```
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]
```

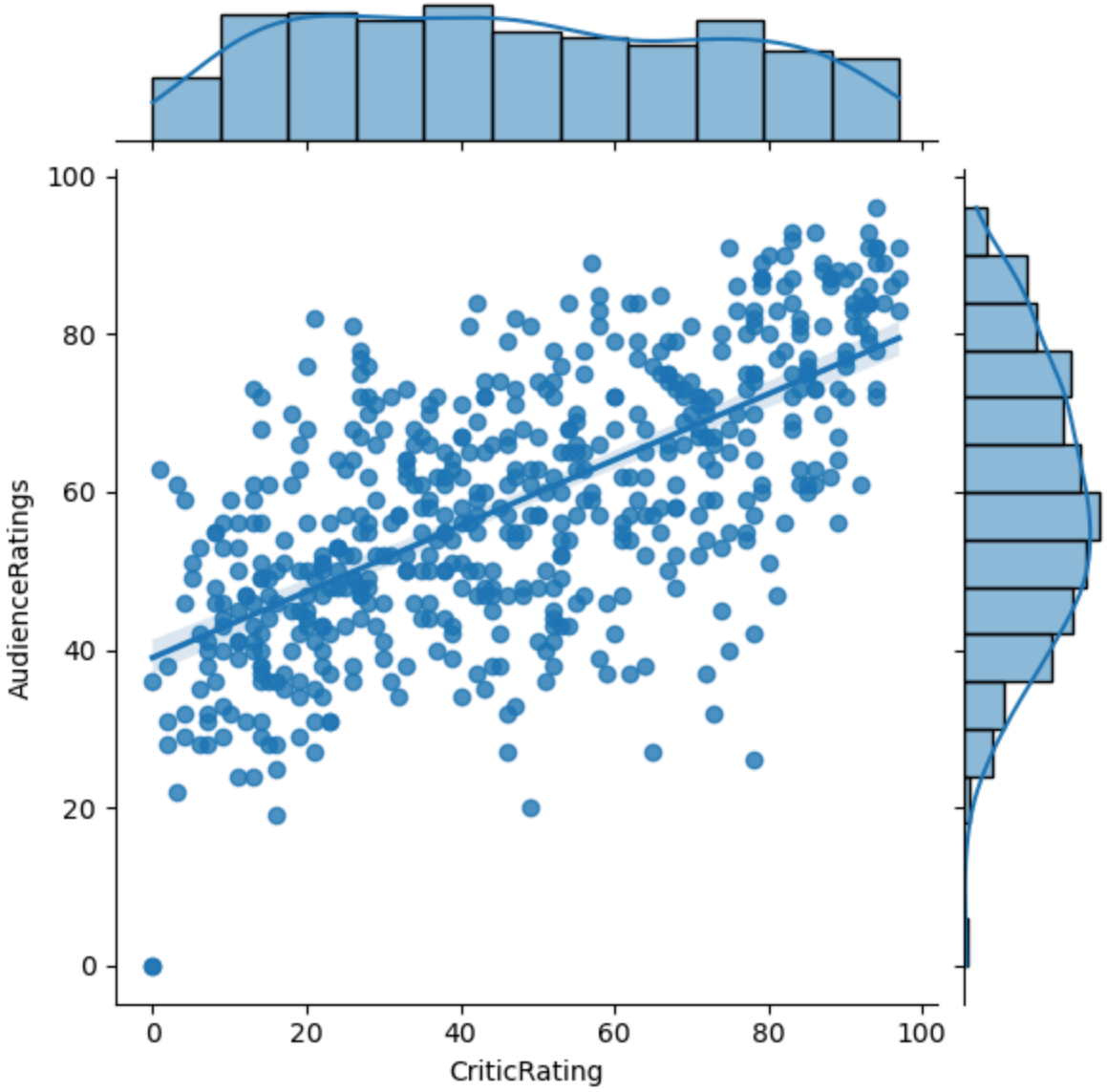
Seperating categorical and numerical data

In [19]:
```
Cat = Film.select_dtypes(include = 'object').columns

num =  Film.select_dtypes(exclude ='object').columns

Cat,num
```

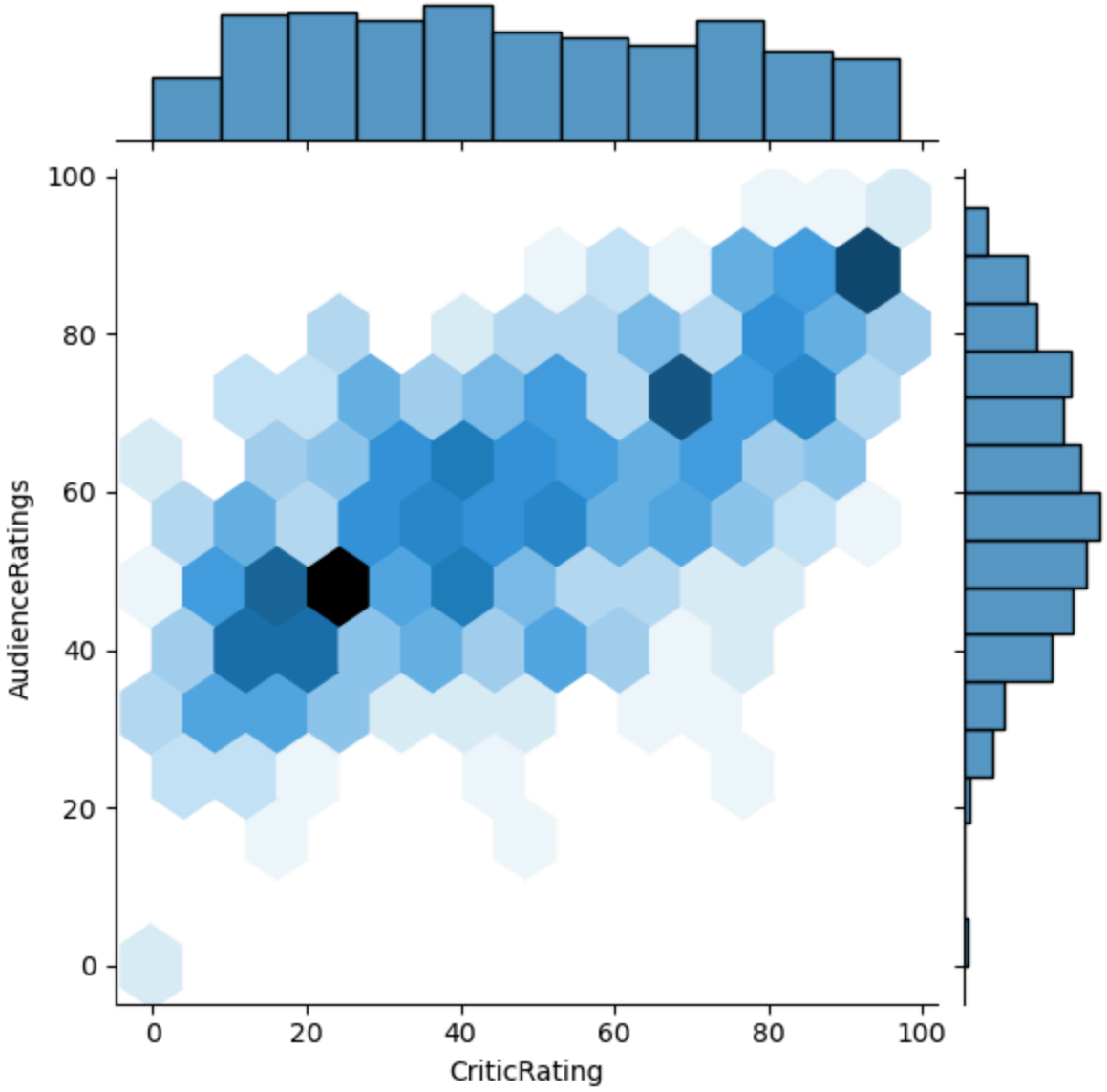
Out[19]:
```
(Index([], dtype='object'),
 Index(['Film', 'Genre', 'CriticRating', 'AudienceRatings', 'BudgetMillion',
        'Year'],
       dtype='object'))
```

In [20]:
```
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

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

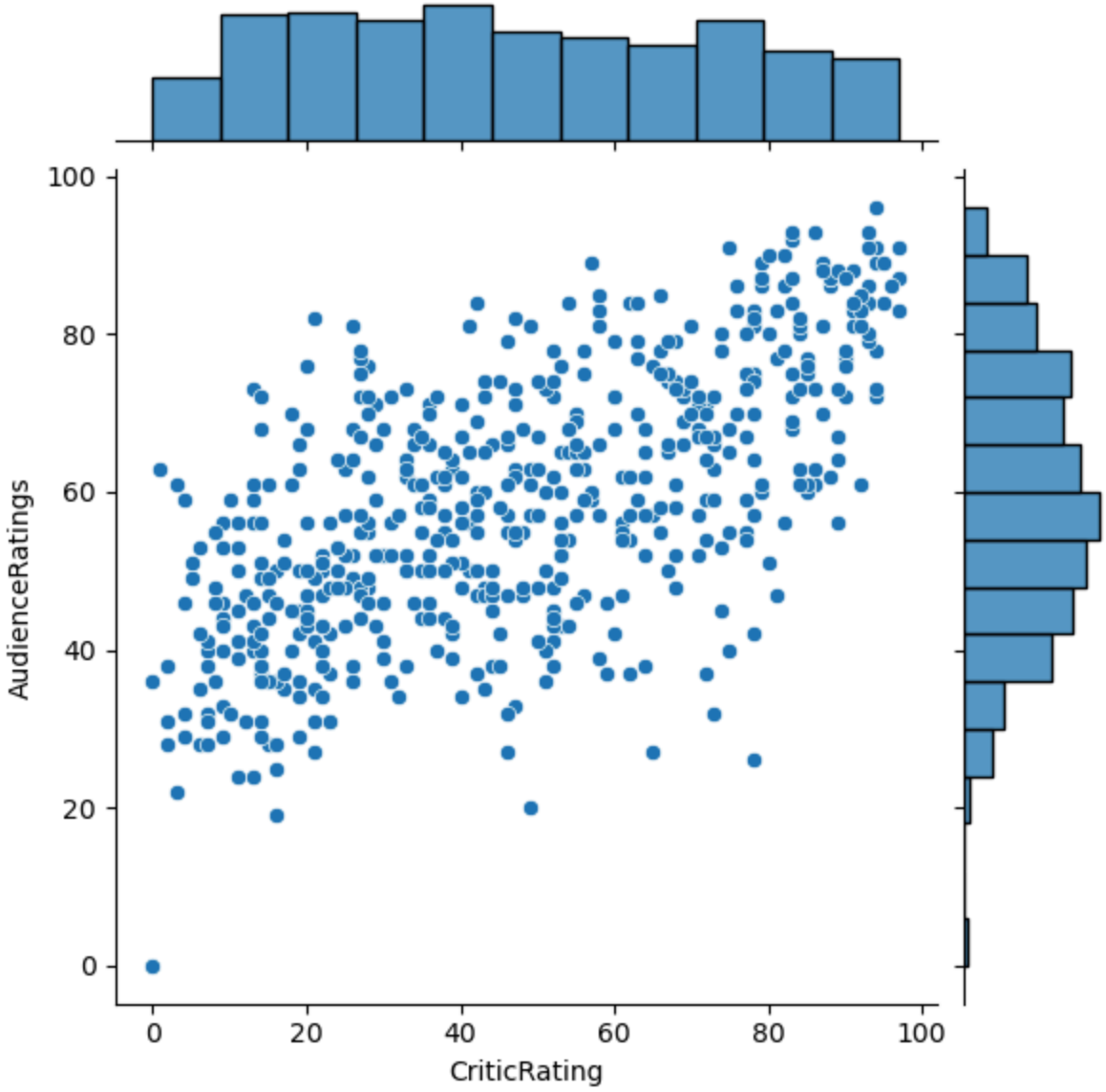
In [21]:
```
viz1 = sns.jointplot(data= Film, x = 'CriticRating', y ='AudienceRatings', kind = 'reg

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



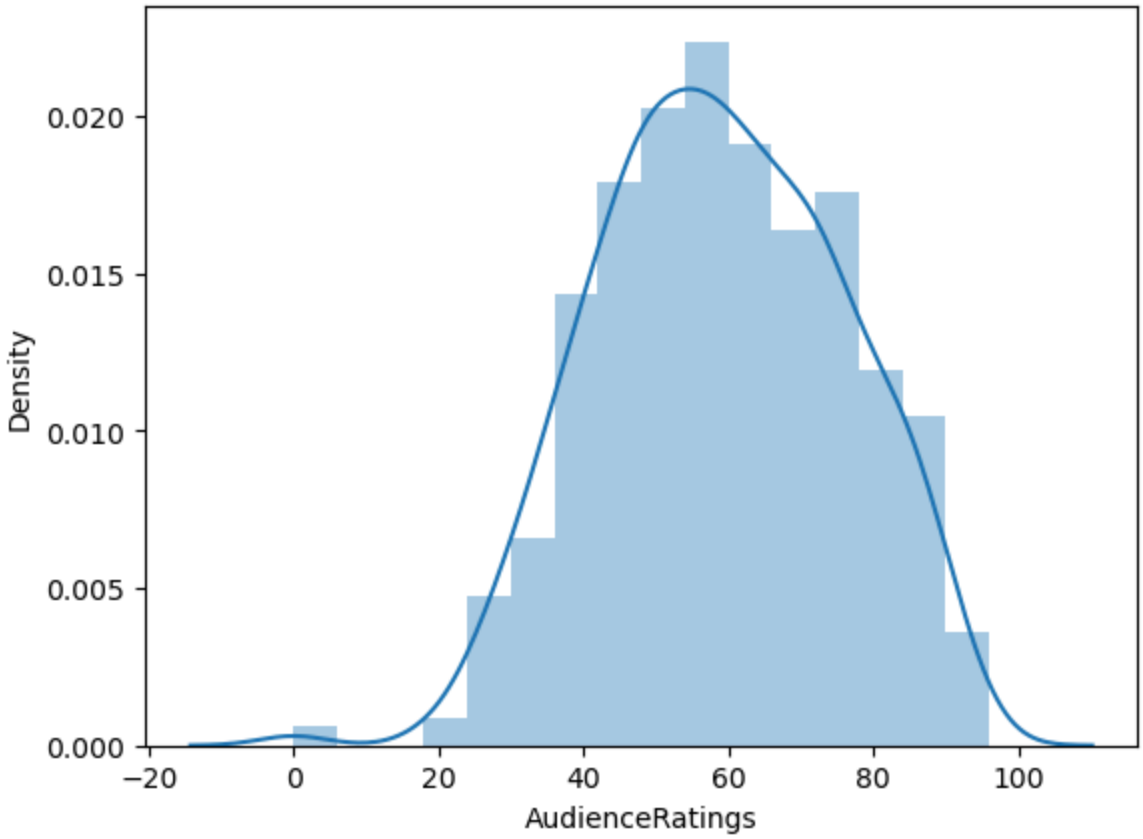
In [22]:
```
viz1 = sns.jointplot(data= Film, x = 'CriticRating', y ='AudienceRatings', kind = 'hex
```



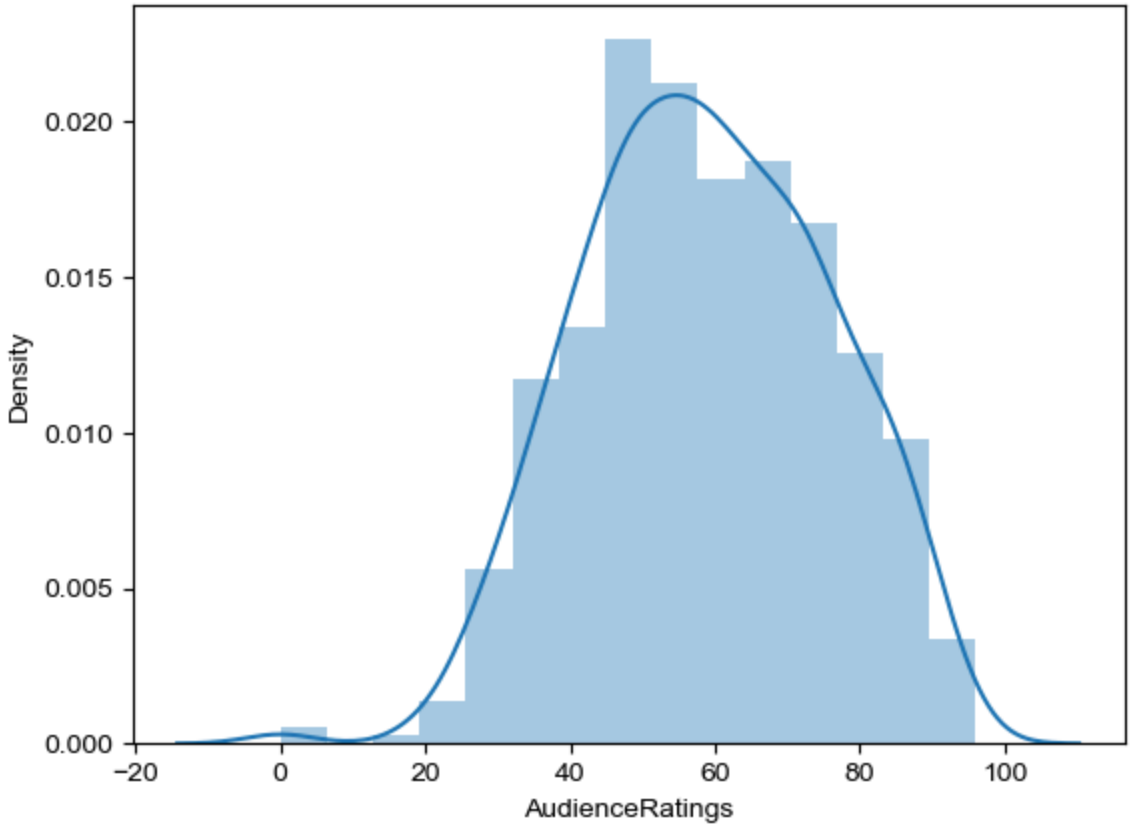
In [36]:
```
viz1 = sns.jointplot(data= Film, x = 'CriticRating', y ='AudienceRatings', kind = 'sca
```



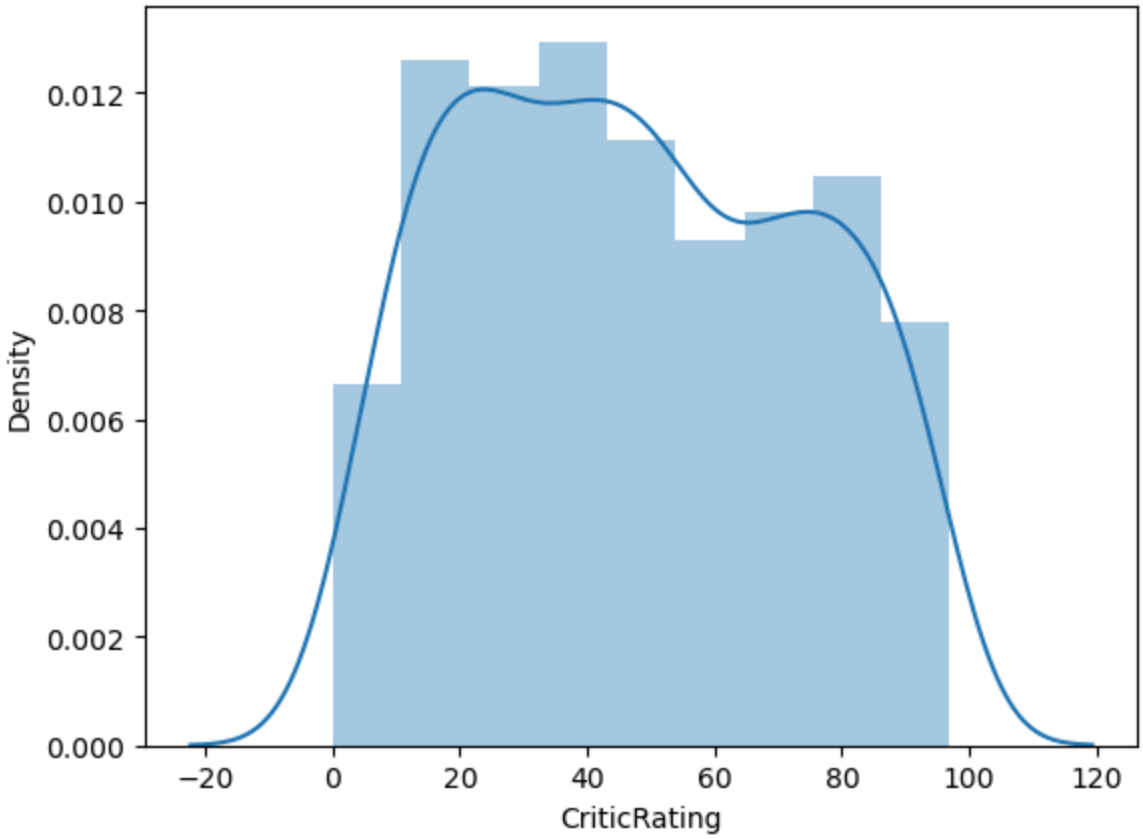
In [38]:
```
viz2 = sns.distplot(Film.AudienceRatings)
```



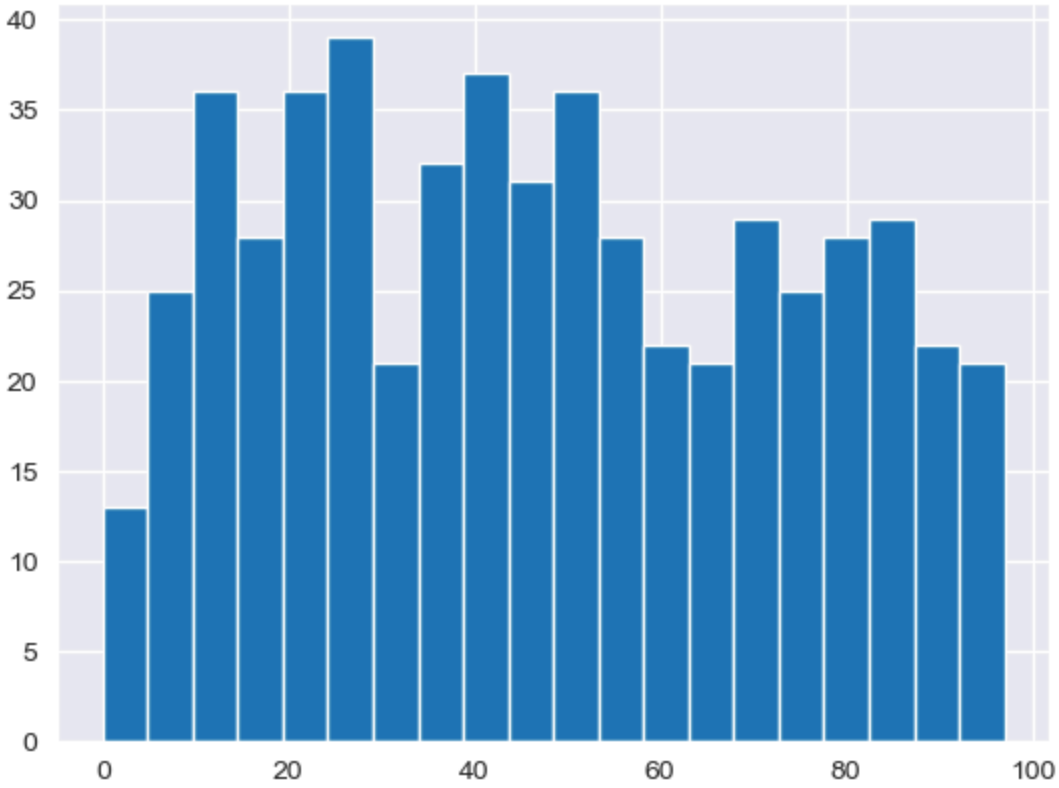
In [30]:
```
sns.distplot(Film.AudienceRatings, bins = 15)
sns.set_style('darkgrid')
```



In [41]:
```
viz3 = sns.distplot(Film.CriticRating)
```

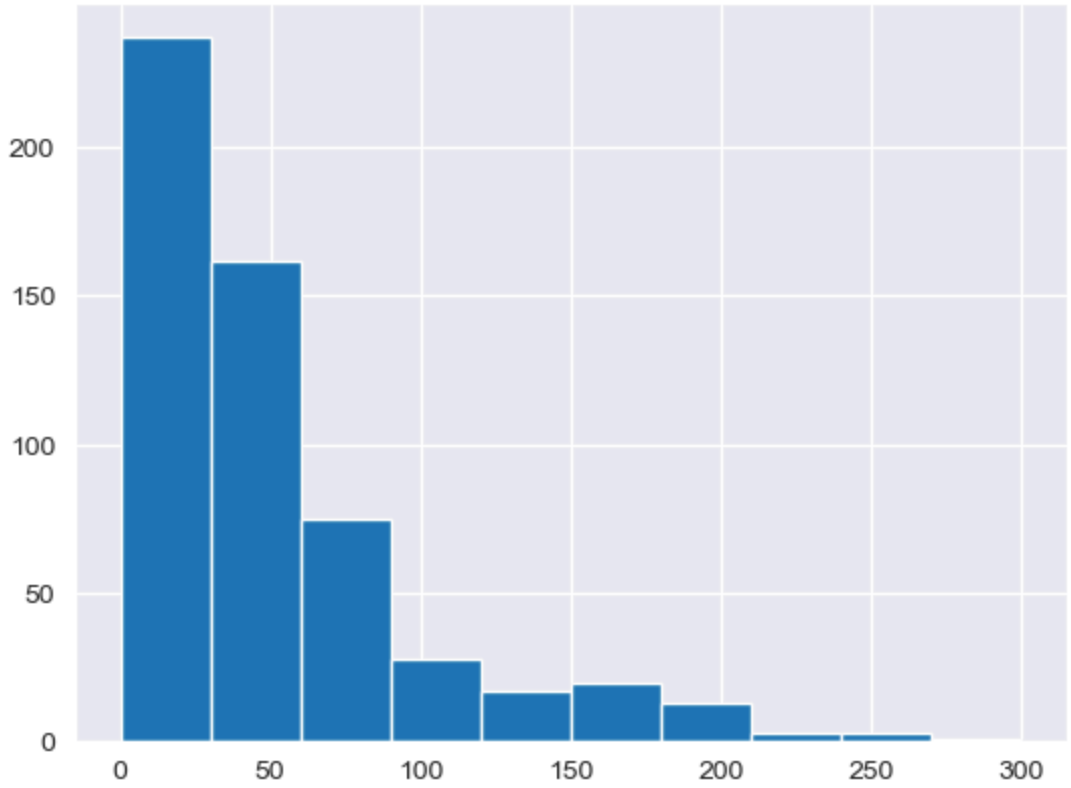


In [32]:
```
n1 = plt.hist(Film.CriticRating, bins=20)
#uniform distribution
```

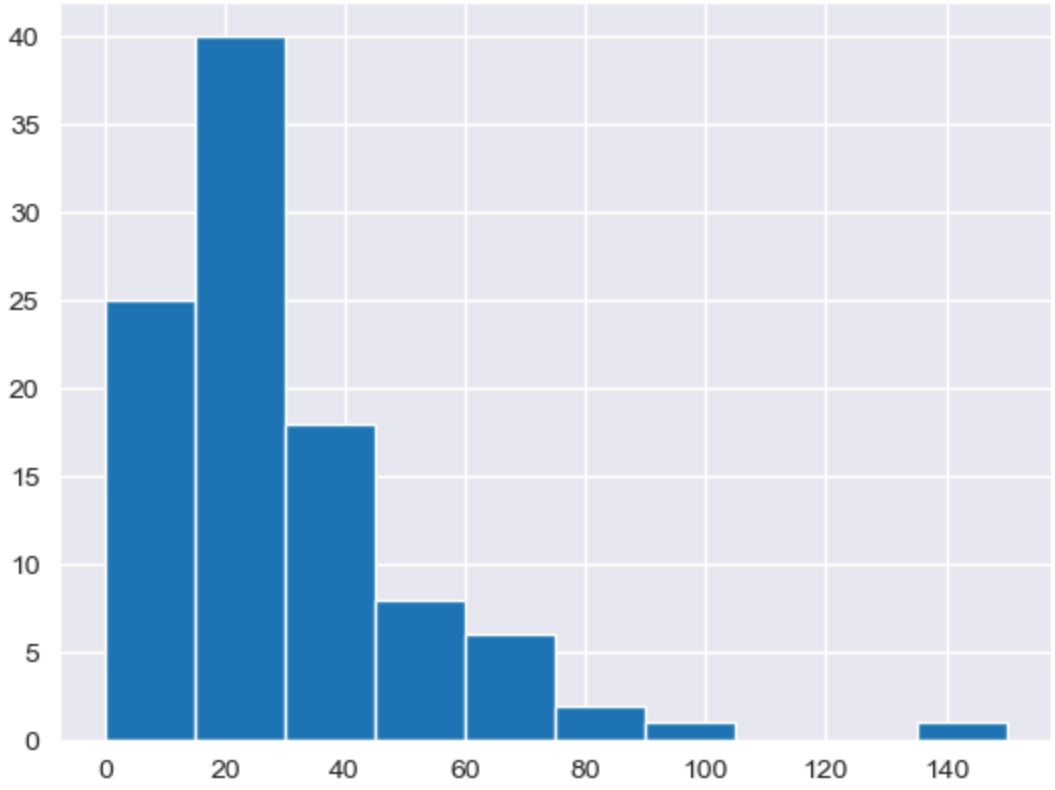


In [34]:
```
#h1 = plt.hist(movies.BudgetMillions)
```

```
plt.hist(Film.BudgetMillion)
plt.show()
```



In [36]:
```
plt.hist(Film[Film.Genre == 'Drama'].BudgetMillion)
plt.show()
```

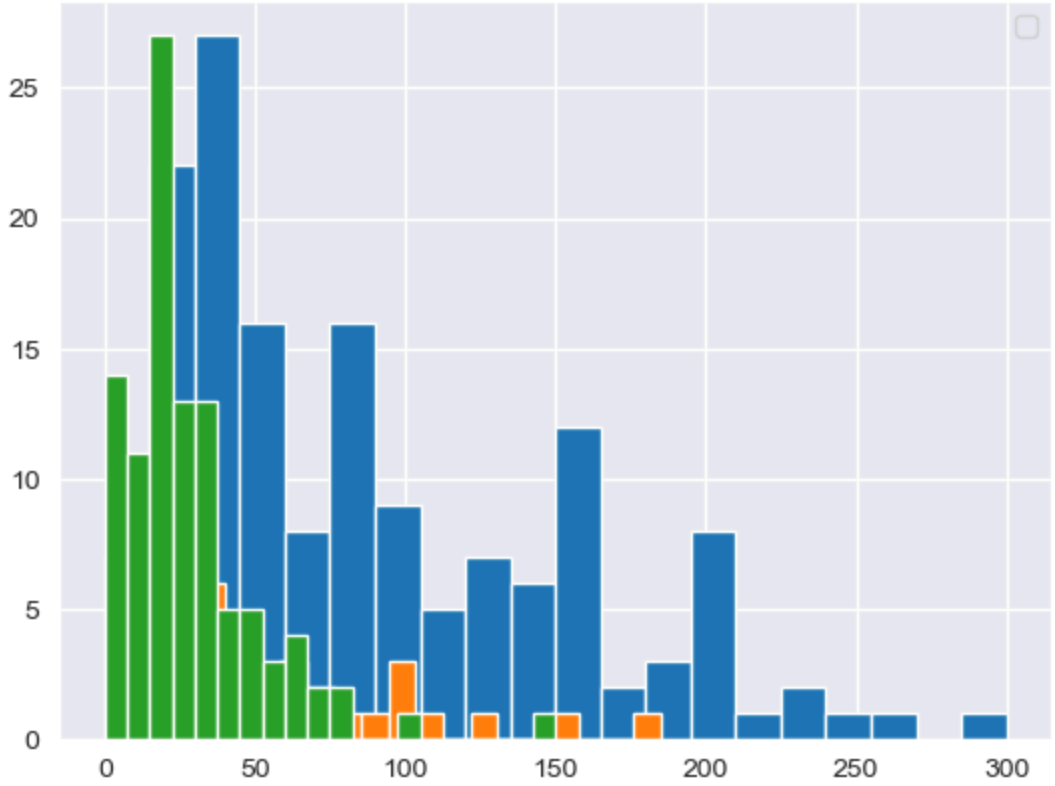


In [37]:
```
Film.Genre.unique()
```

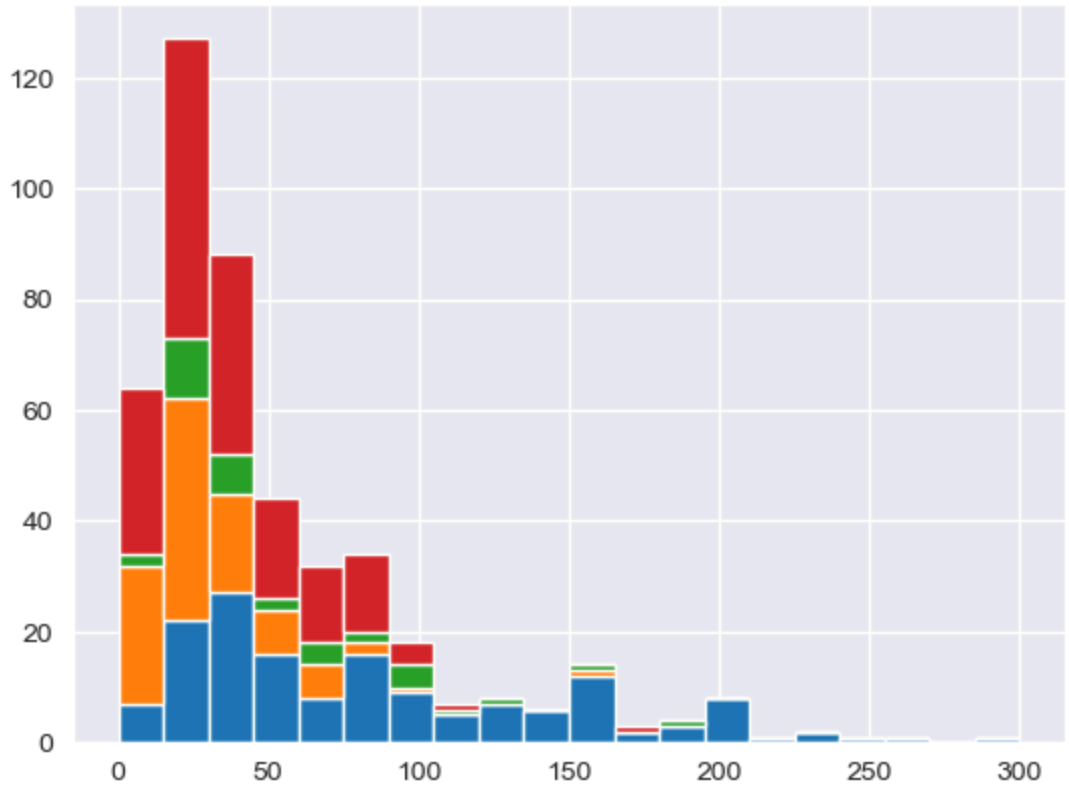
Out[37]:
```
['Comedy', 'Adventure', 'Action', 'Horror', 'Drama', 'Romance', 'Thriller']
Categories (7, object): ['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romanc
e', 'Thriller']
```

In [38]:
```
# Below plots are stacked histogram becuase overlaped

plt.hist(Film[Film.Genre == 'Action'].BudgetMillion, bins = 20)
plt.hist(Film[Film.Genre == 'Thriller'].BudgetMillion, bins = 20)
plt.hist(Film[Film.Genre == 'Drama'].BudgetMillion, bins = 20)
plt.legend()
plt.show()
```

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



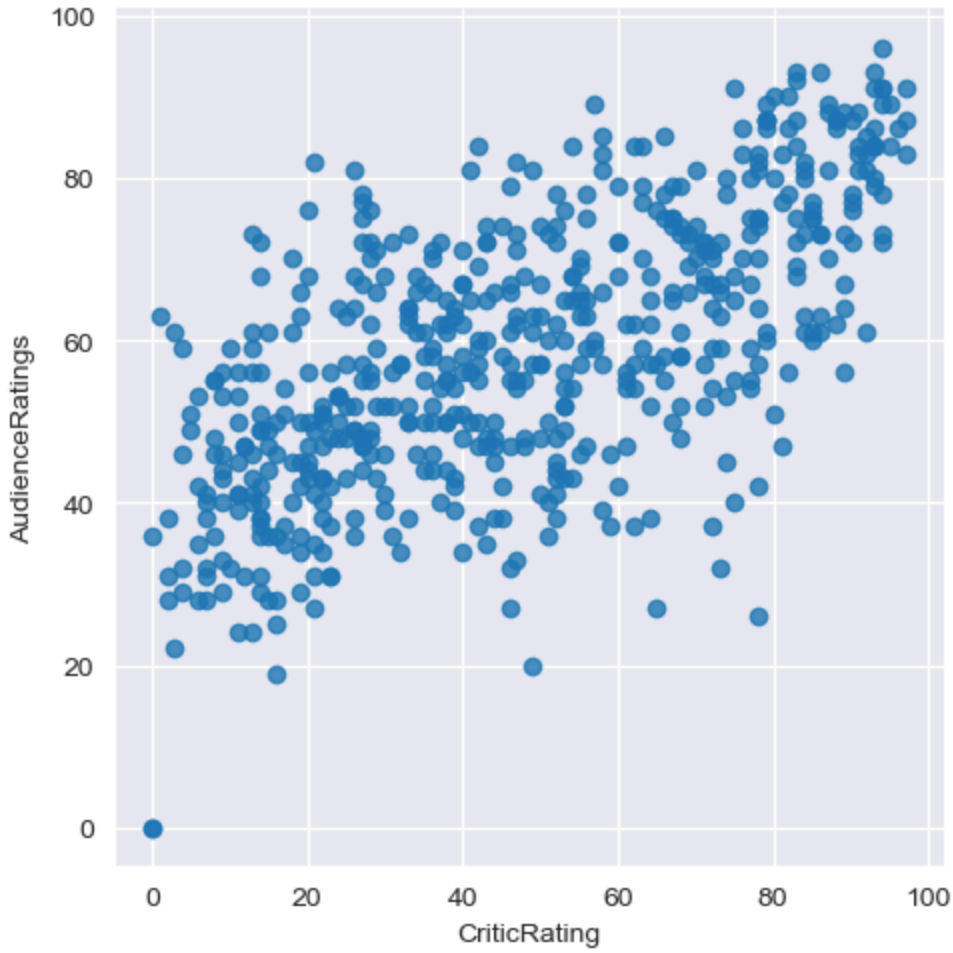
In [41]:
```
plt.hist([Film[Film.Genre == 'Action'].BudgetMillion,\
          Film[Film.Genre == 'Drama'].BudgetMillion, \
          Film[Film.Genre == 'Thriller'].BudgetMillion, \
          Film[Film.Genre == 'Comedy'].BudgetMillion],
         bins = 20, stacked = True)
plt.show()
```



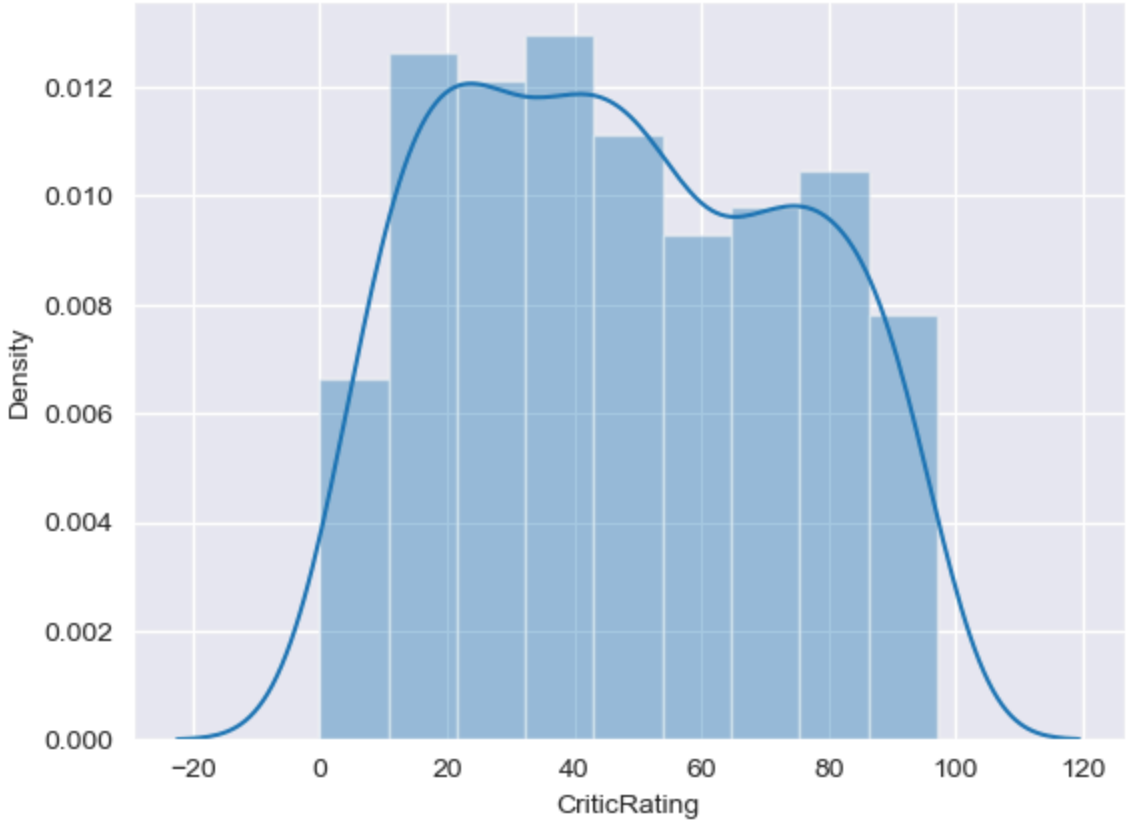
In [43]:
```
# if you have 100 categories you cannot copy & paste all the things

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

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

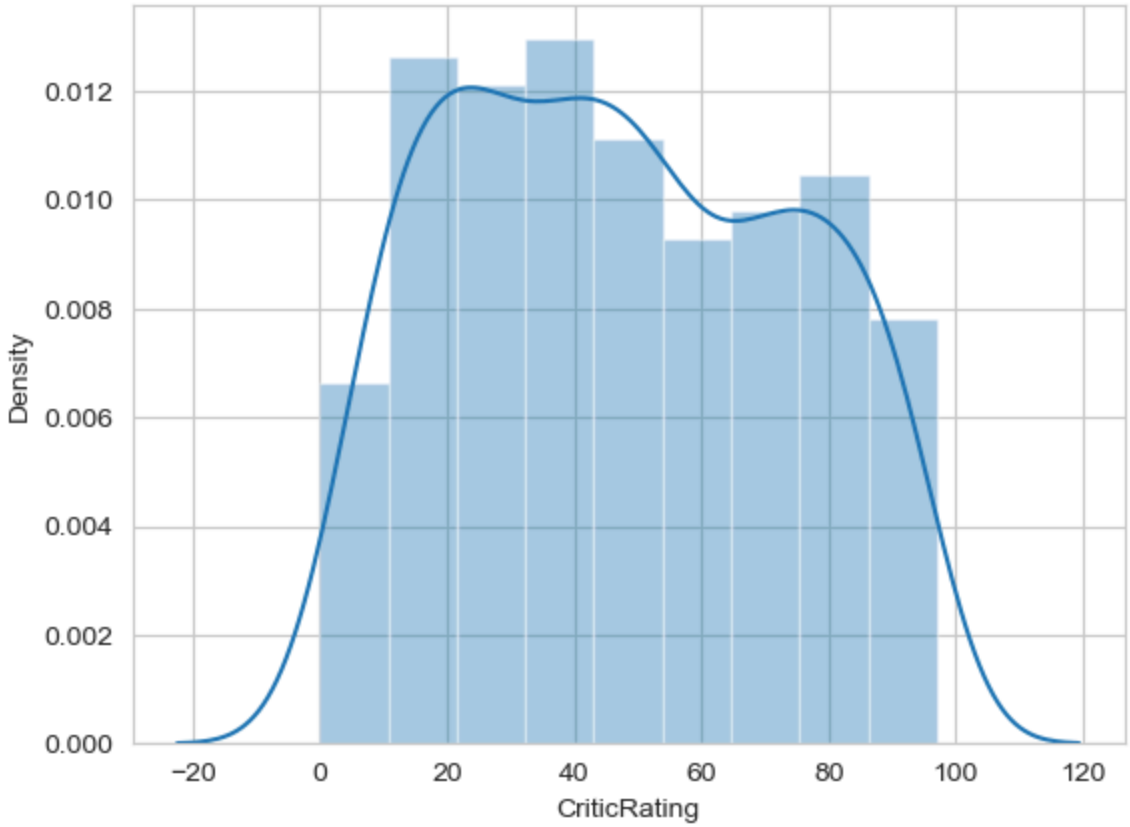
In [44]:
```
vis1 = sns.lmplot(data=Film, x='CriticRating', y='AudienceRatings',\
                  fit_reg=False)
```



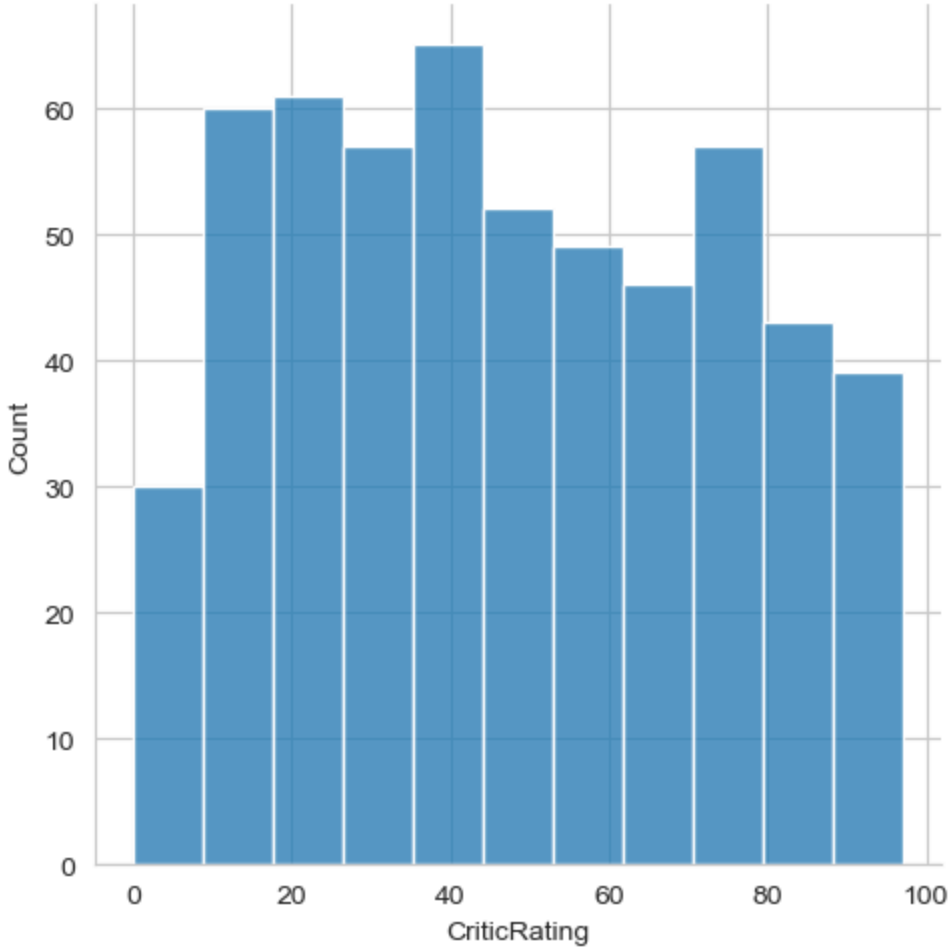
In [43]:
```
sns.set_style('darkgrid')
viz3 = sns.distplot(Film.CriticRating)
```



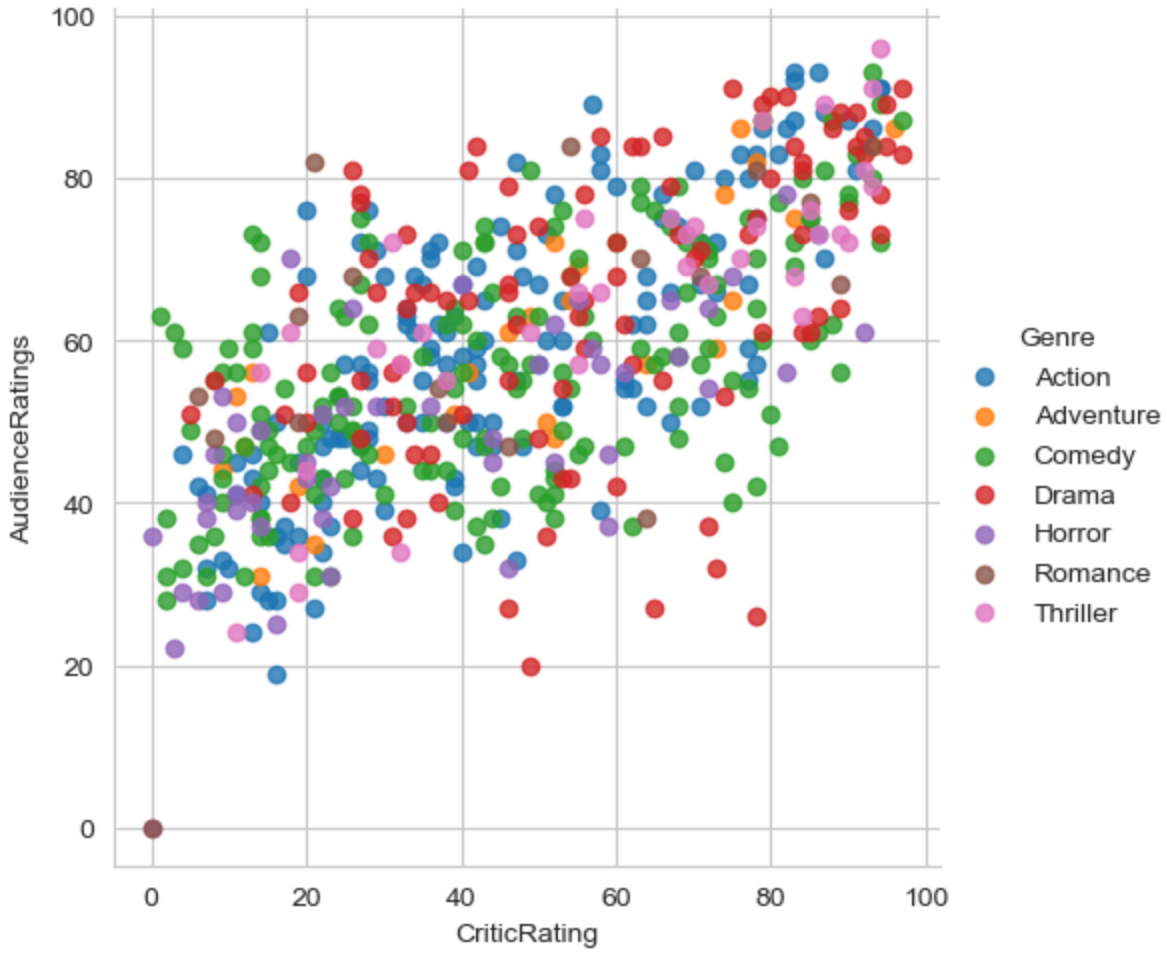
In [45]:
```
sns.set_style('whitegrid')
viz3 = sns.distplot(Film.CriticRating)
```



In [46]:
```
viz3 = sns.displot(Film.CriticRating)
sns.set_style('darkgrid')
```

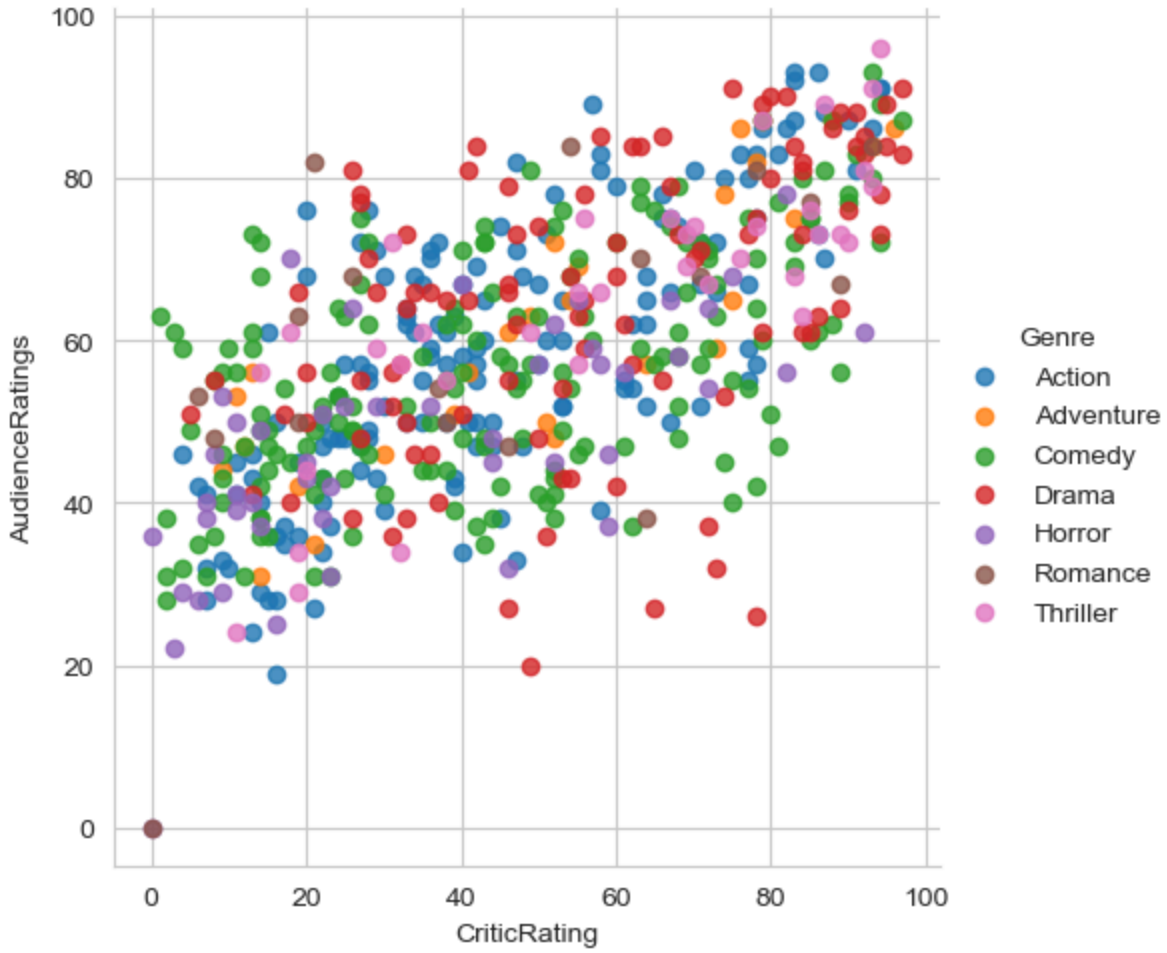


In [47]:
```
vis1 = sns.lmplot(data=Film, x='CriticRating', y='AudienceRatings',\
                  fit_reg=False, hue = 'Genre')
```



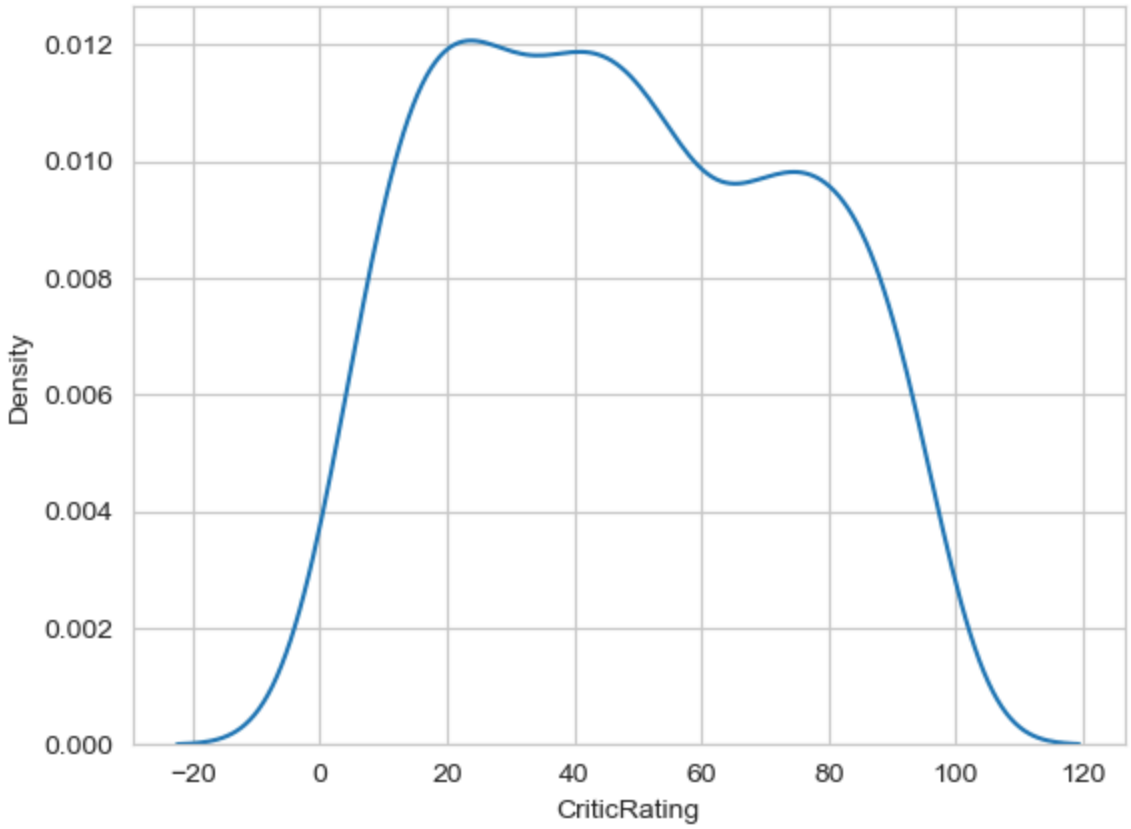
In [55]:
```
sns.lmplot(data=Film, x='CriticRating', y='AudienceRatings',\
                  fit_reg=False, hue = 'Genre',aspect=1)
```

Out[55]:
```
<seaborn.axisgrid.FacetGrid at 0x28157485ed0>
```

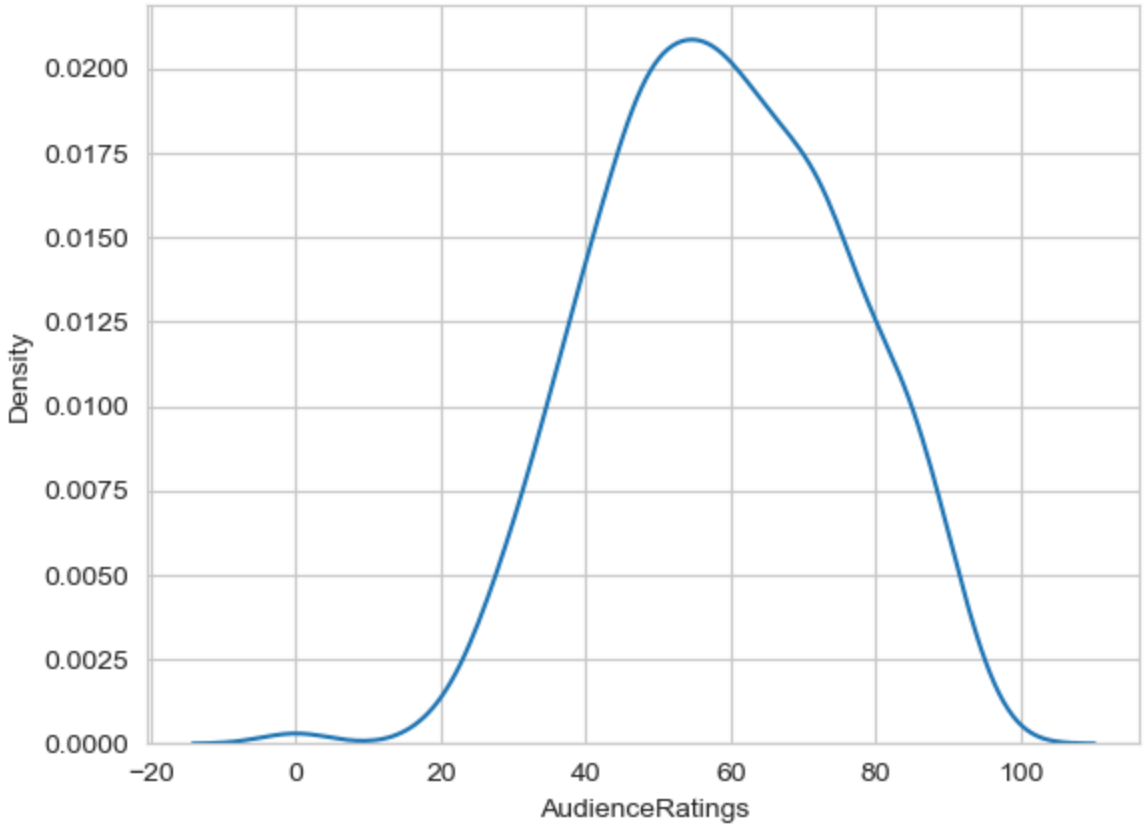


In [56]:
```
# Kernal Density Estimate plot ( KDE PLOT)
# how can i visulize audience rating & critics rating . using scatterplot
```

In [63]:
```
k1 = sns.kdeplot(Film.CriticRating)
# where do u find more density and how density is distibuted across from the the chat
# center point is kernal this is calld KDE & insteade of dots it visualize like this
# we can able to clearly see the spread at the audience ratings
```

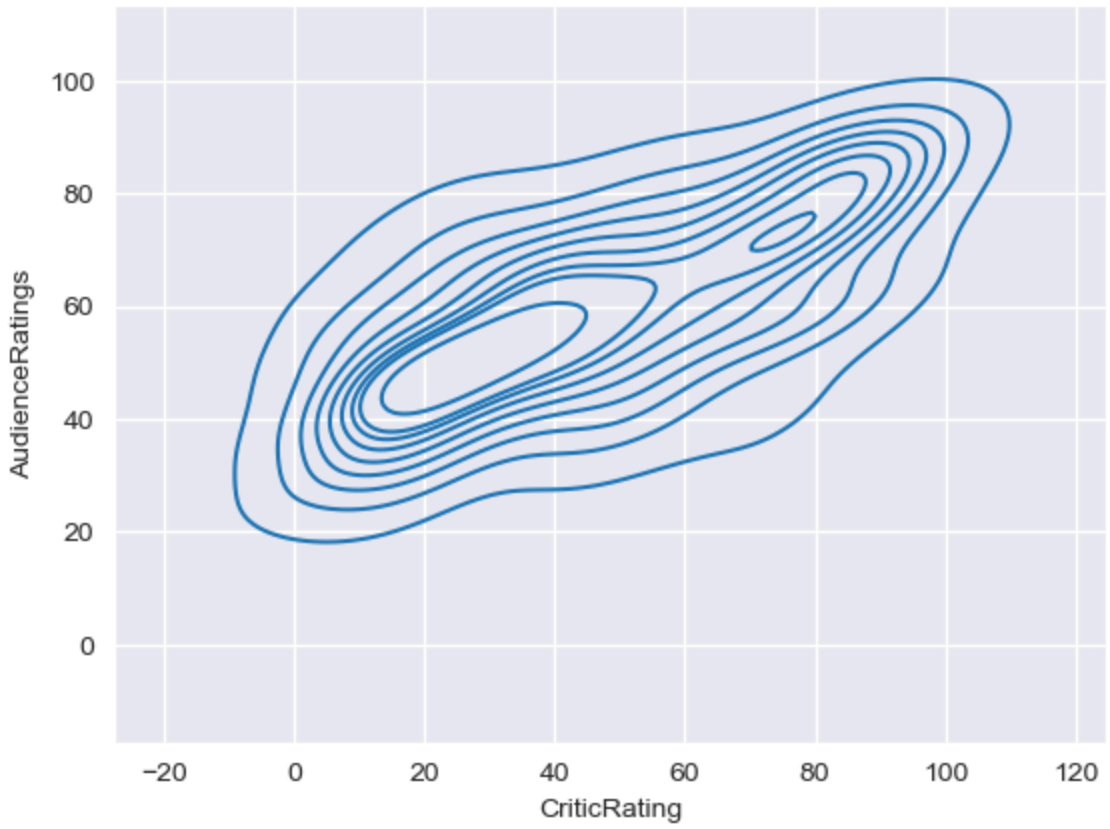


In [64]:
```
K2 = sns.kdeplot(Film.AudienceRatings)
```



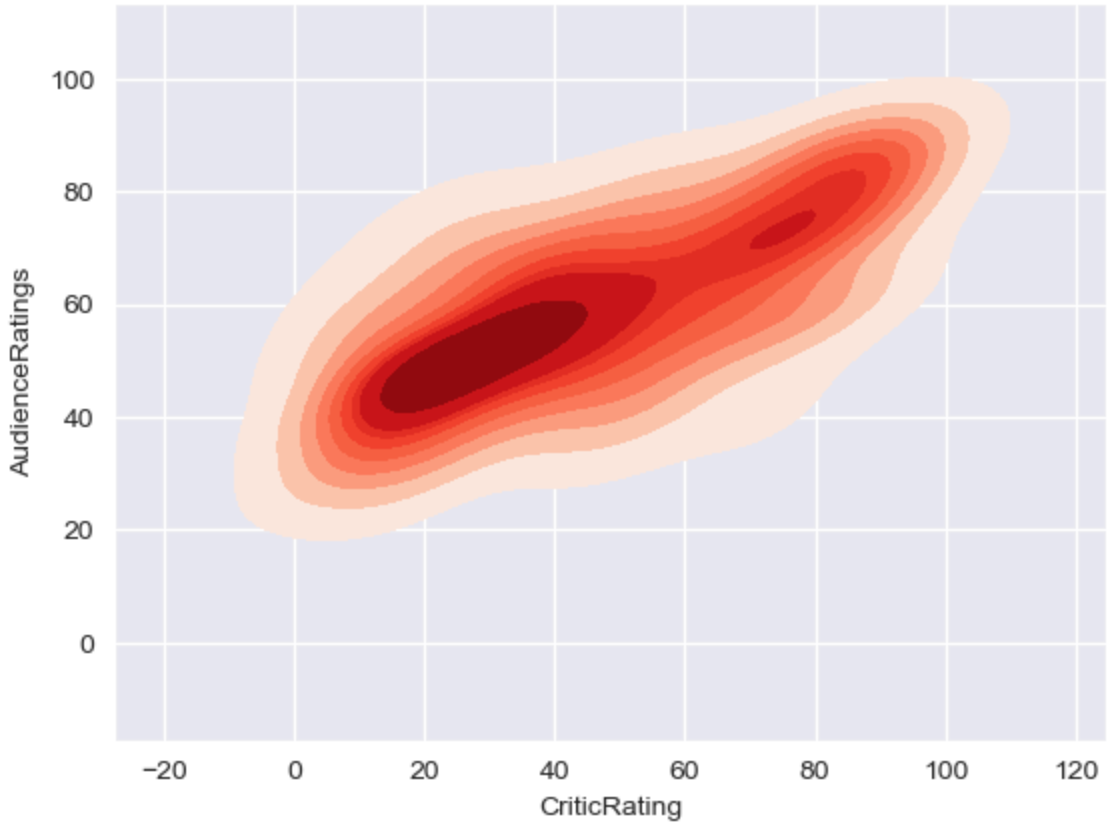
In [83]:
```
sns.kdeplot(x = Film.CriticRating , y = Film.AudienceRatings)
```

Out[83]: <Axes: xlabel='CriticRating', ylabel='AudienceRatings'>



In [84]:
```
sns.kdeplot(x = Film.CriticRating,y = Film.AudienceRatings,shade = True,shade_lowest=F
```

Out[84]: <Axes: xlabel='CriticRating', ylabel='AudienceRatings'>

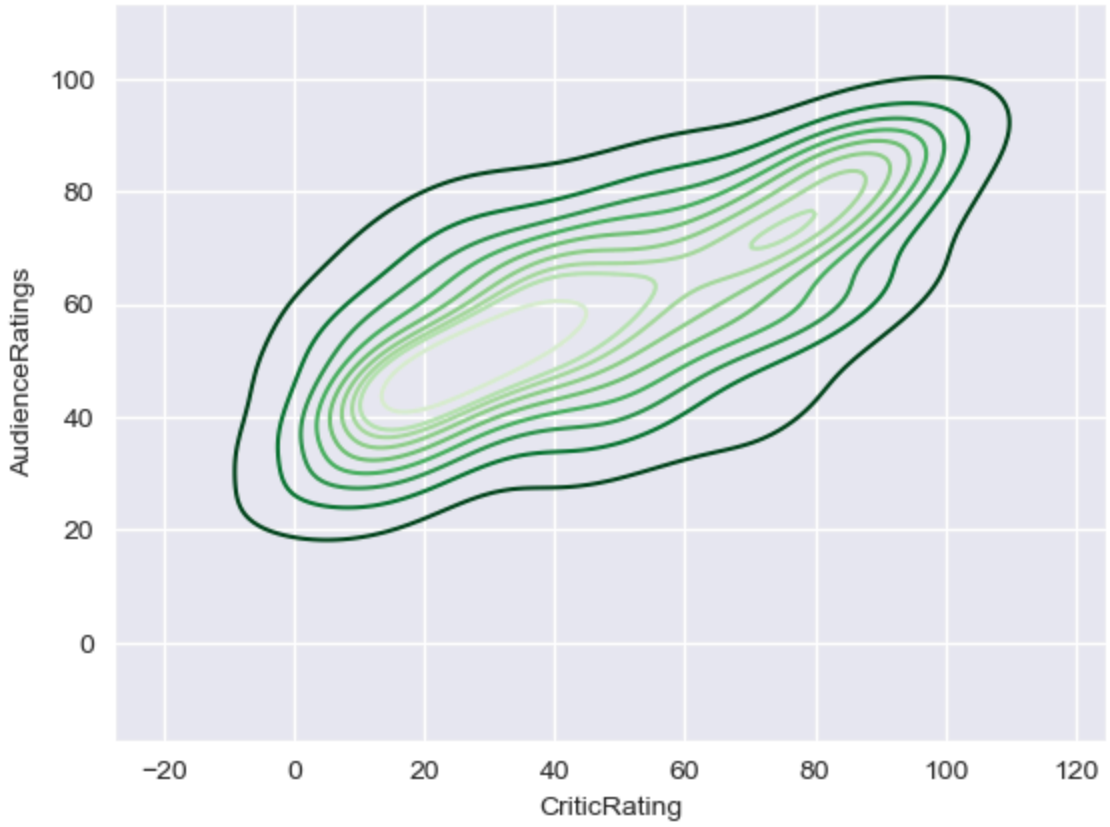


In [70]:
```
axes
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[70], line 1
----> 1 axes

NameError: name 'axes' is not defined
```

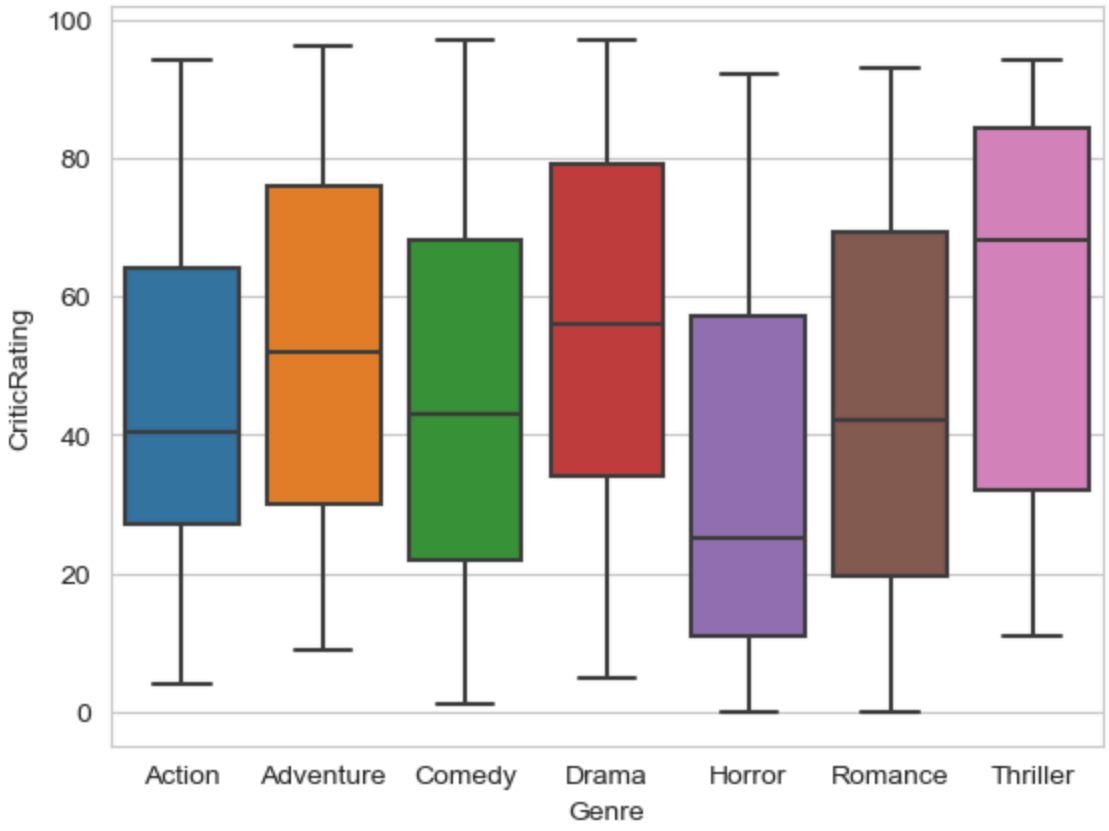
In [94]:
```
sns.kdeplot(x = Film.CriticRating, y = Film.AudienceRatings, shade_lowest = False, cma
```

Out[94]:
```
<Axes: xlabel='CriticRating', ylabel='AudienceRatings'>
```



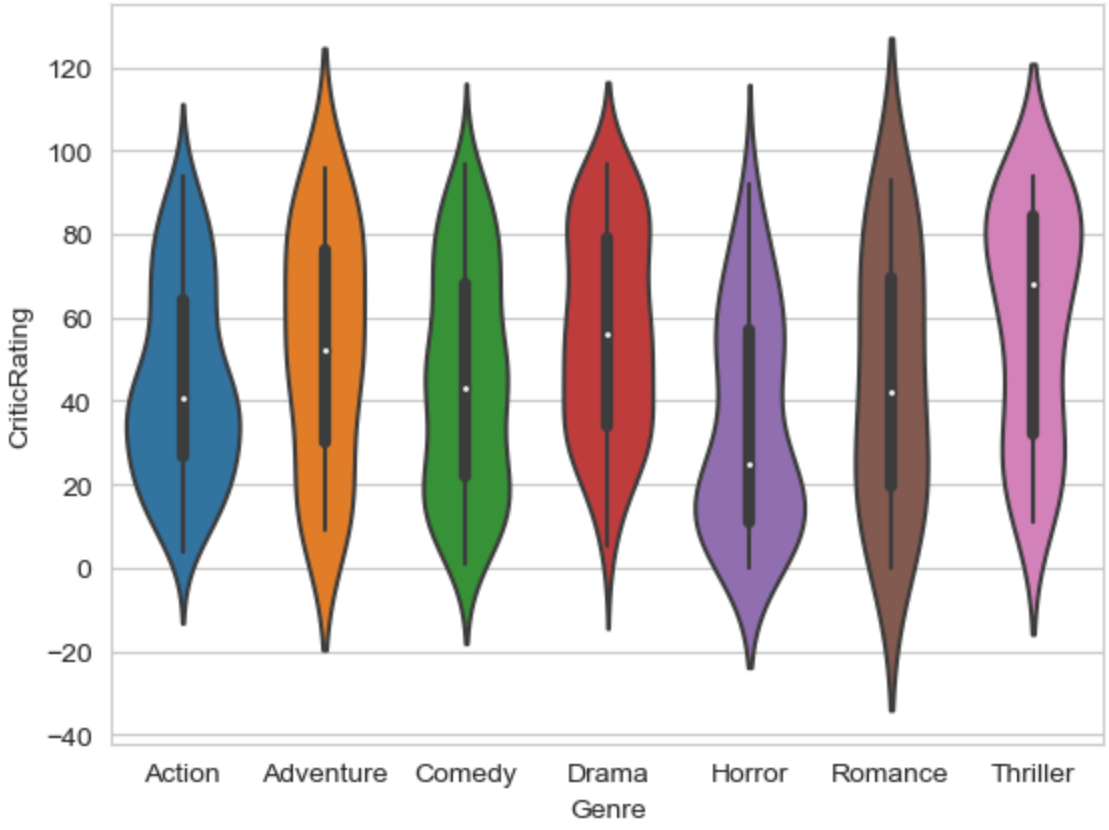
In [ ]:
```
```

In [71]:
```
# Boxplots
w = sns.boxplot(data=Film, x='Genre', y = 'CriticRating')
```



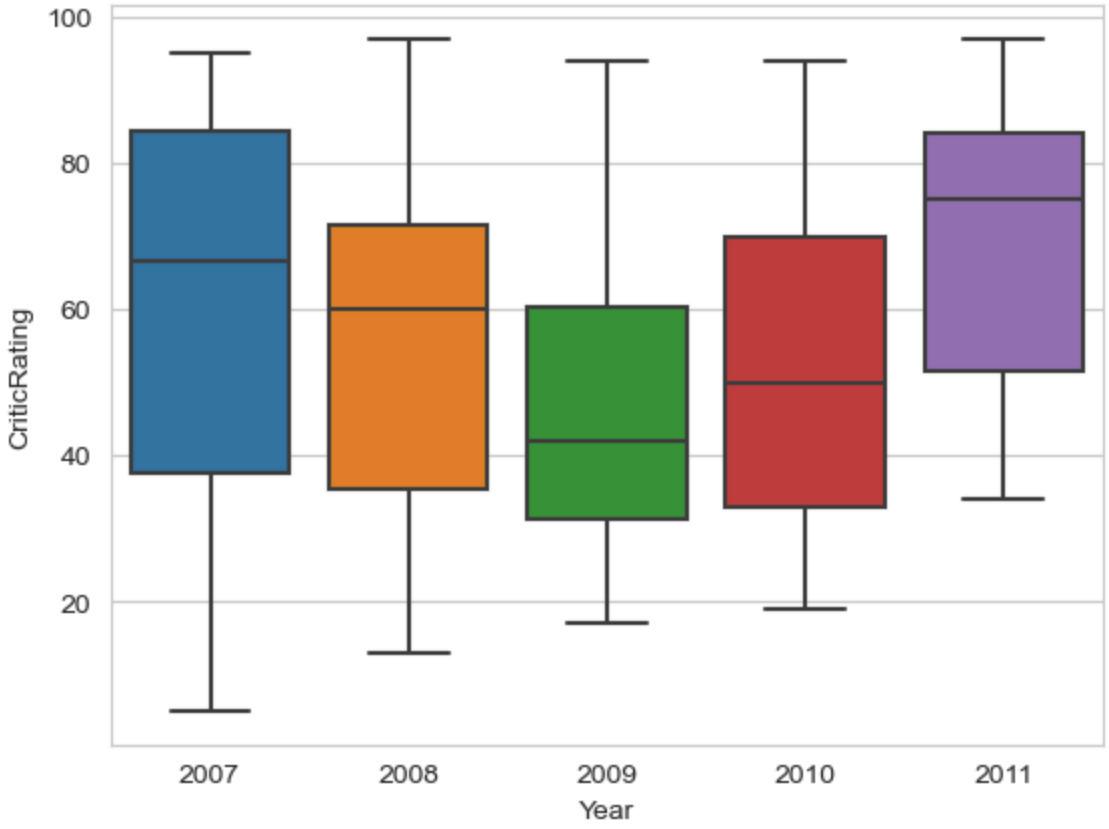
In [72]:
```
# Voilin Plots
sns.violinplot(data=Film, x='Genre', y = 'CriticRating')
```

Out[72]:
```
<Axes: xlabel='Genre', ylabel='CriticRating'>
```



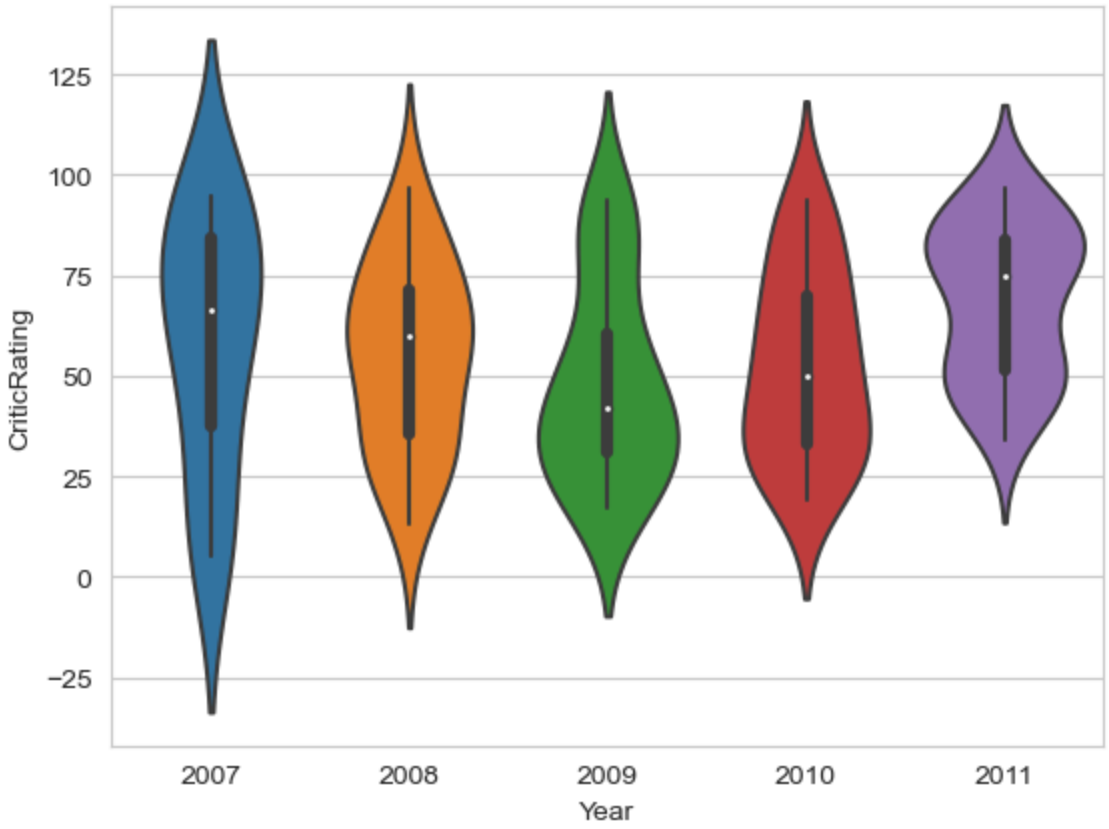
In [73]:
```
sns.boxplot(data=Film[Film.Genre == 'Drama'], x='Year', y = 'CriticRating')
```

Out[73]:
```
<Axes: xlabel='Year', ylabel='CriticRating'>
```



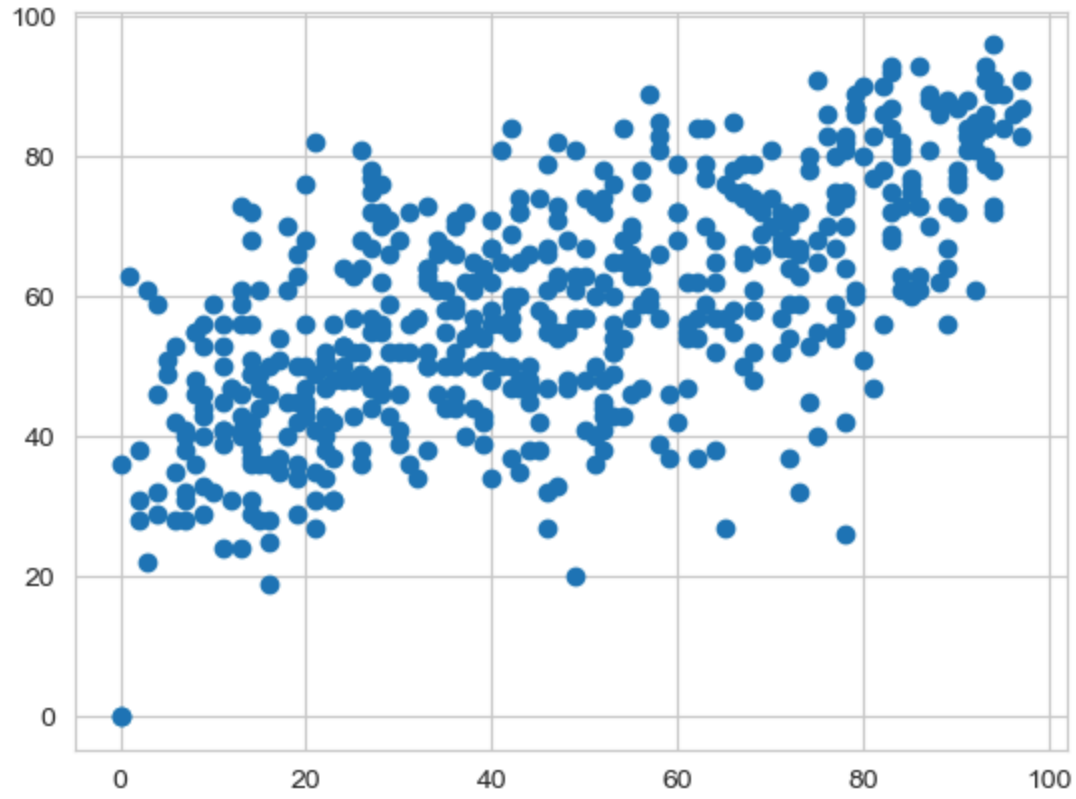
In [74]:
```
sns.violinplot(data=Film[Film.Genre == 'Drama'], x='Year', y = 'CriticRating')
```

Out[74]:
```
<Axes: xlabel='Year', ylabel='CriticRating'>
```



In [76]:
```
plt.scatter(Film.CriticRating,Film.AudienceRatings)
```

Out[76]:
```
<matplotlib.collections.PathCollection at 0x28156186290>
```

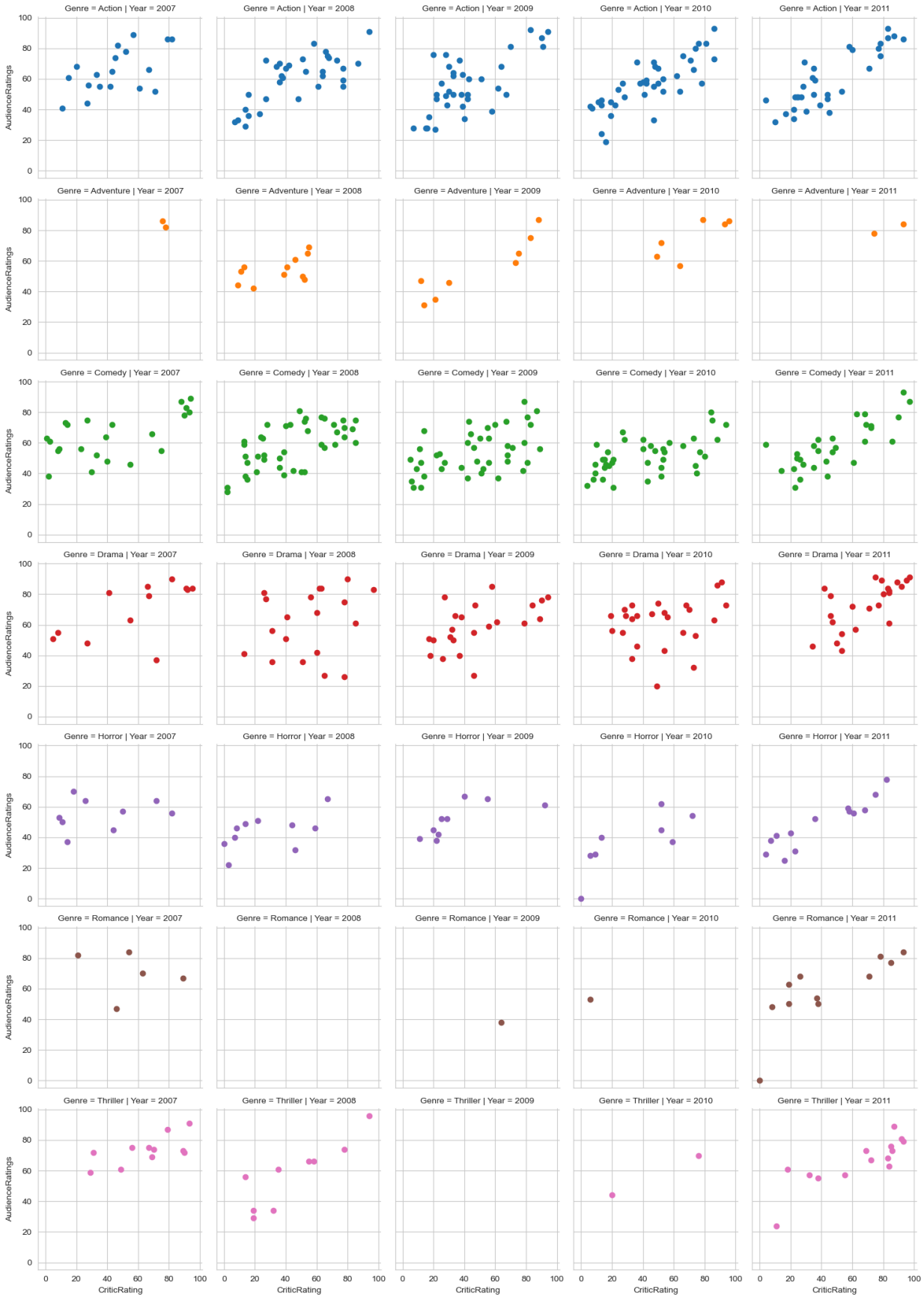


In [77]:
```
# Creating a face grid
```

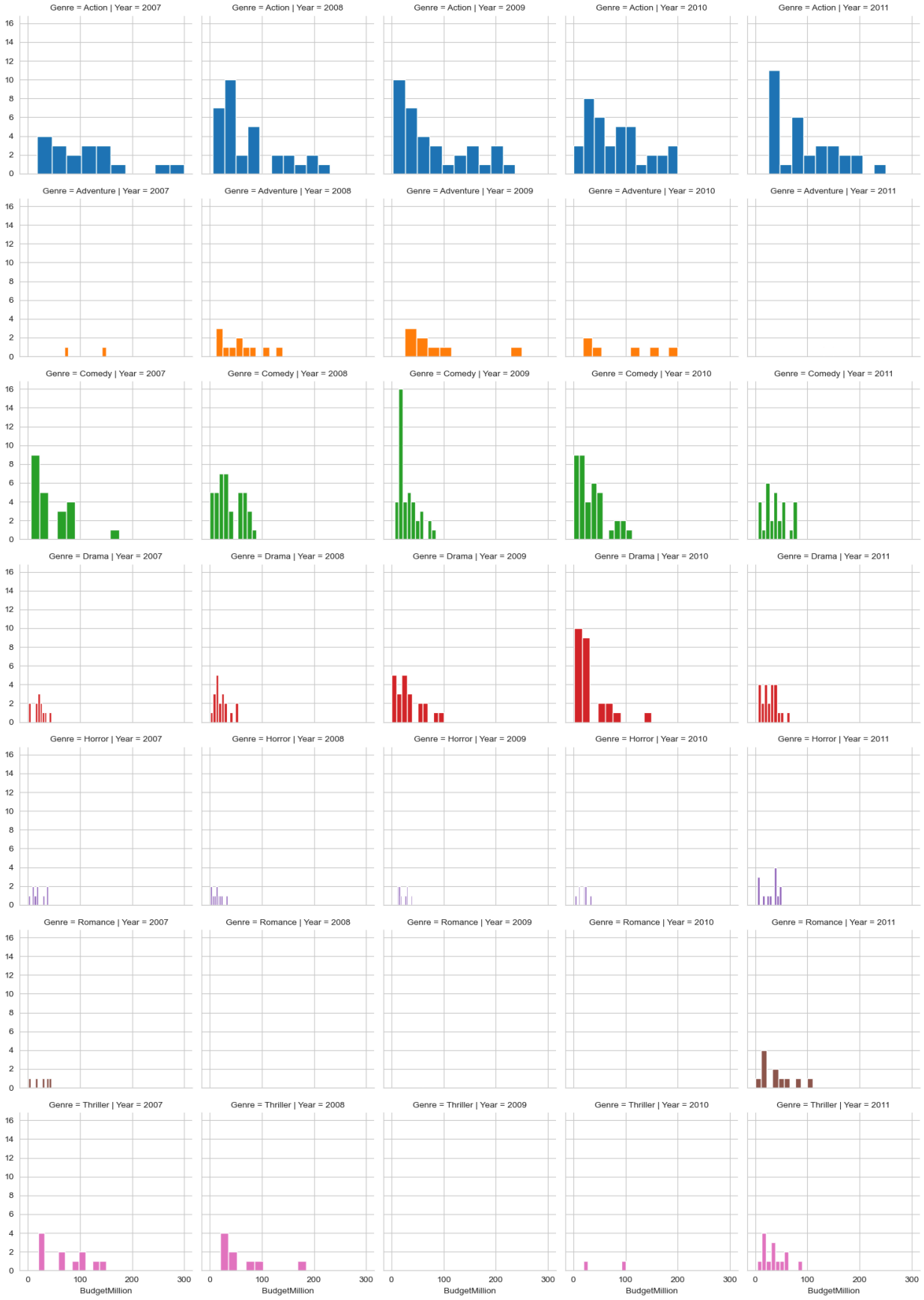
```
sns.FacetGrid (Film, row = 'Genre', col = 'Year', hue = 'Genre')

#kind of subplots
```

Out[77]: <seaborn.axisgrid.FacetGrid at 0x28156114a00>

In [78]:
```
g =sns.FacetGrid (Film, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRatings' )

#scatterplots are mapped in facetgrid
```



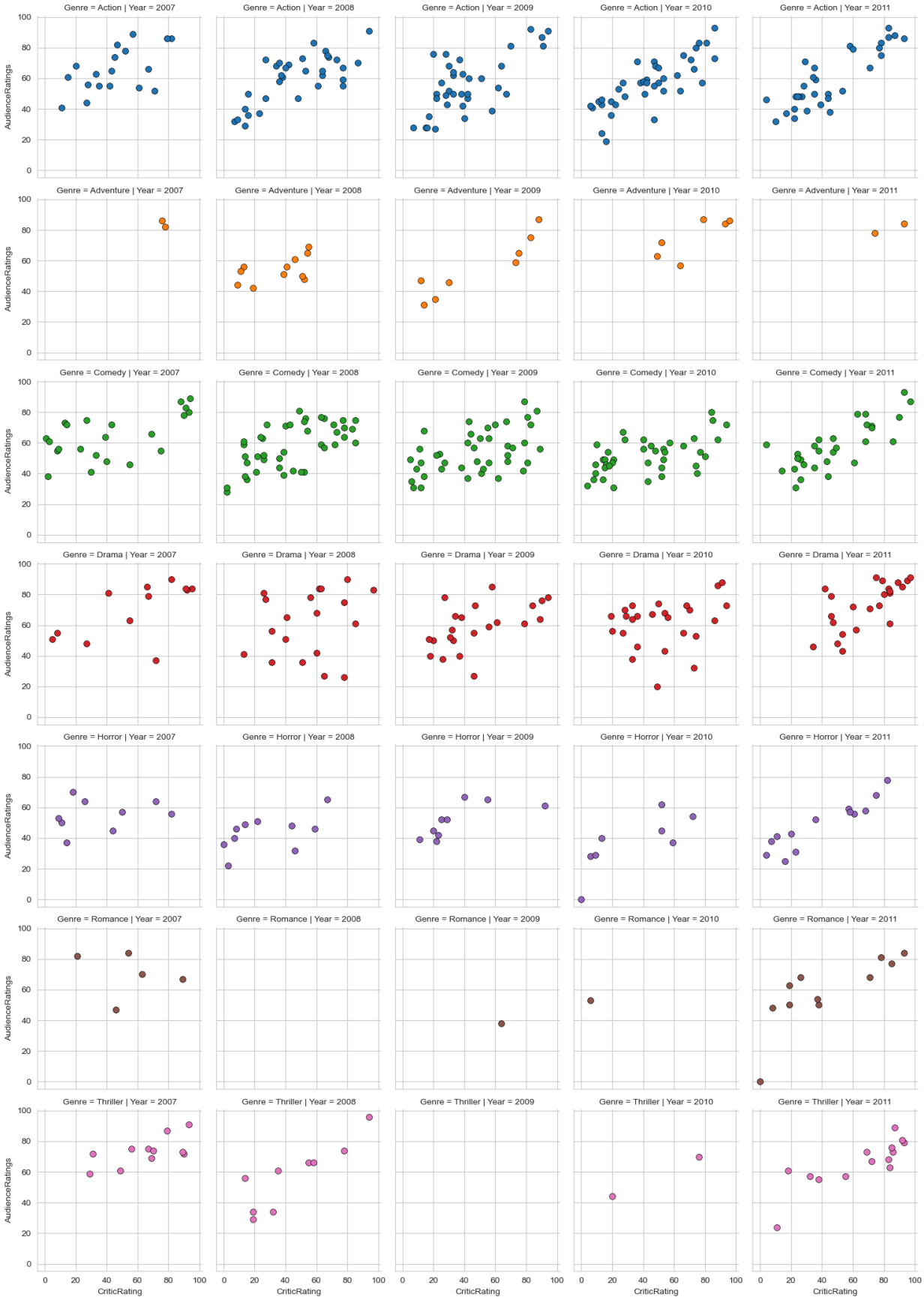
In [80]:
```
# you can populated any type of chat.

g =sns.FacetGrid (Film, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.hist, 'BudgetMillion')

#scatterplots are mapped in facetgrid
```



In [81]:

```
#
g =sns.FacetGrid (Film, row = 'Genre', col = 'Year', hue = 'Genre')
kws = dict(s=50, linewidth=0.5,edgecolor='black')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRatings',**kws )
```

```
#scatterplots are mapped in facetgrid
```



In [89]:
```
# 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 = Film.BudgetMillion,y = Film.AudienceRatings,ax=axes[0,0])
k2 = sns.kdeplot(x = Film.BudgetMillion,y = Film.CriticRating,ax = axes[0,1])

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

z = sns.violinplot(data=Film[Film.Genre=='Drama'], x='Year', y = 'CriticRating', ax=ax

k4 = sns.kdeplot(x = Film.CriticRating,y=Film.AudienceRatings,shade = True,shade_lowes

k4b = sns.kdeplot(x = Film.CriticRating,y = Film.AudienceRatings,cmap='Reds',ax = axes

plt.show()
```