```
In [1]:    # MOVIE RATING ANALYTICS (ADVANCED VISULIZATION)

           import pandas as pd
           import os
```

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

Out[2]:   'C:\\Users\\shaik'

```
In [3]:    movies = pd.read_csv(r"C:\Users\shaik\Downloads\Movie-Rating.csv")
```

```
In [4]:    movies
```

Out[4]:

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

559 rows × 6 columns

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

Out[5]:   559

```
In [6]:    movies.head()
```

Out[6]:

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

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

Out[7]:

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

In [8]: `movies.columns`

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

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

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

Out[10]:

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

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Film           559 non-null    object
 1   Genre          559 non-null    object
 2   CriticRating   559 non-null    int64
 3   AudienceRating 559 non-null    int64
 4   BudgetMillions 559 non-null    int64
 5   Year           559 non-null    int64
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [12]:
```python
movies.describe()
# if you look at the year the data type is int but when you look at the mean val
# we have to change to categroy type
# also from object datatype we will convert to category datatypes
#
```

Out[12]:

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

In [13]:
```python
movies['Film']
#movies['Audience Ratings %']
```

Out[13]:
```
0        (500) Days of Summer
1              10,000 B.C.
2              12 Rounds
3              127 Hours
4              17 Again
              ...
554          Your Highness
555        Youth in Revolt
556              Zodiac
557            Zombieland
558            Zookeeper
Name: Film, Length: 559, dtype: object
```

In [14]:
```python
movies.Film
```

```
Out[14]: 0        (500) Days of Summer
         1               10,000 B.C.
         2                 12 Rounds
         3                 127 Hours
         4                   17 Again
                          ...
         554            Your Highness
         555          Youth in Revolt
         556                    Zodiac
         557                Zombieland
         558                 Zookeeper
         Name: Film, Length: 559, dtype: object
```

In [15]: `movies.Film = movies.Film.astype('category')`

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: category
         Categories (559, object): ['(500) Days of Summer ', '10,000 B.C.', '12 Rounds
         ', '127 Hours', ..., 'Youth in Revolt', 'Zodiac', 'Zombieland ', 'Zookeeper']
```

In [17]: `movies.head()`

Out[17]:

	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 [18]: ```
movies.info()

now the same thing we will change genra to category & year to category
```

```
<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]:
```python
movies.Genre = movies.Genre.astype('category')
movies.Year = movies.Year.astype('category')
```

In [20]:
```python
movies.Genre
```

Out[20]:
```
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 [21]:
```python
movies.Year # is it real no. year you can take average,min,max but out come have
```

Out[21]:
```
0      2009
1      2008
2      2009
3      2010
4      2009
       ...
554    2011
555    2009
556    2007
557    2009
558    2011
Name: Year, Length: 559, dtype: category
Categories (5, int64): [2007, 2008, 2009, 2010, 2011]
```

In [22]:
```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    category
 2   CriticRating   559 non-null    int64
 3   AudienceRating 559 non-null    int64
 4   BudgetMillions 559 non-null    int64
 5   Year           559 non-null    category
dtypes: category(3), int64(3)
memory usage: 36.5 KB
```

In [23]:
```python
movies.Genre.cat.categories
```

Out[23]:
```
Index(['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance',
       'Thriller'],
      dtype='object')
```

In [24]:
```python
movies.describe()
#now when you see the describt you will get only integer value mean, standard de
```

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

In [25]:
```python
#How to working with joint plots
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [26]:
```python
j = sns.jointplot( data = movies, x = 'CriticRating', y = 'AudienceRating')
plt.show()
# Audience rating is more dominant then critics rating
# Based on this we find out as most people are most liklihood to watch audience
# let me explain the excel - if you filter audience rating & critic rating. crit
```
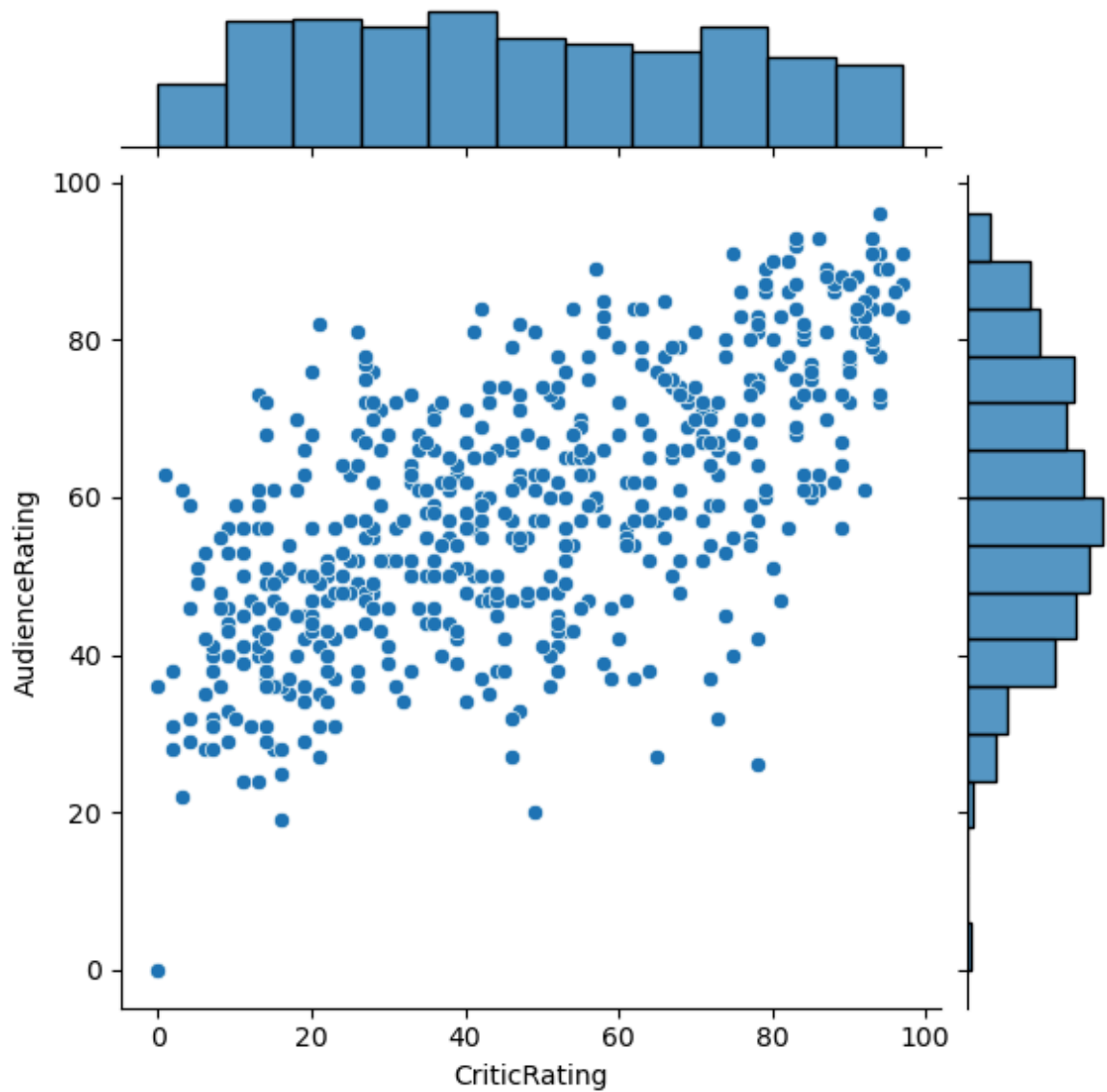
```
In [27]:  j = sns.jointplot( data = movies, x = 'CriticRating', y = 'AudienceRating', kind
          plt.show()

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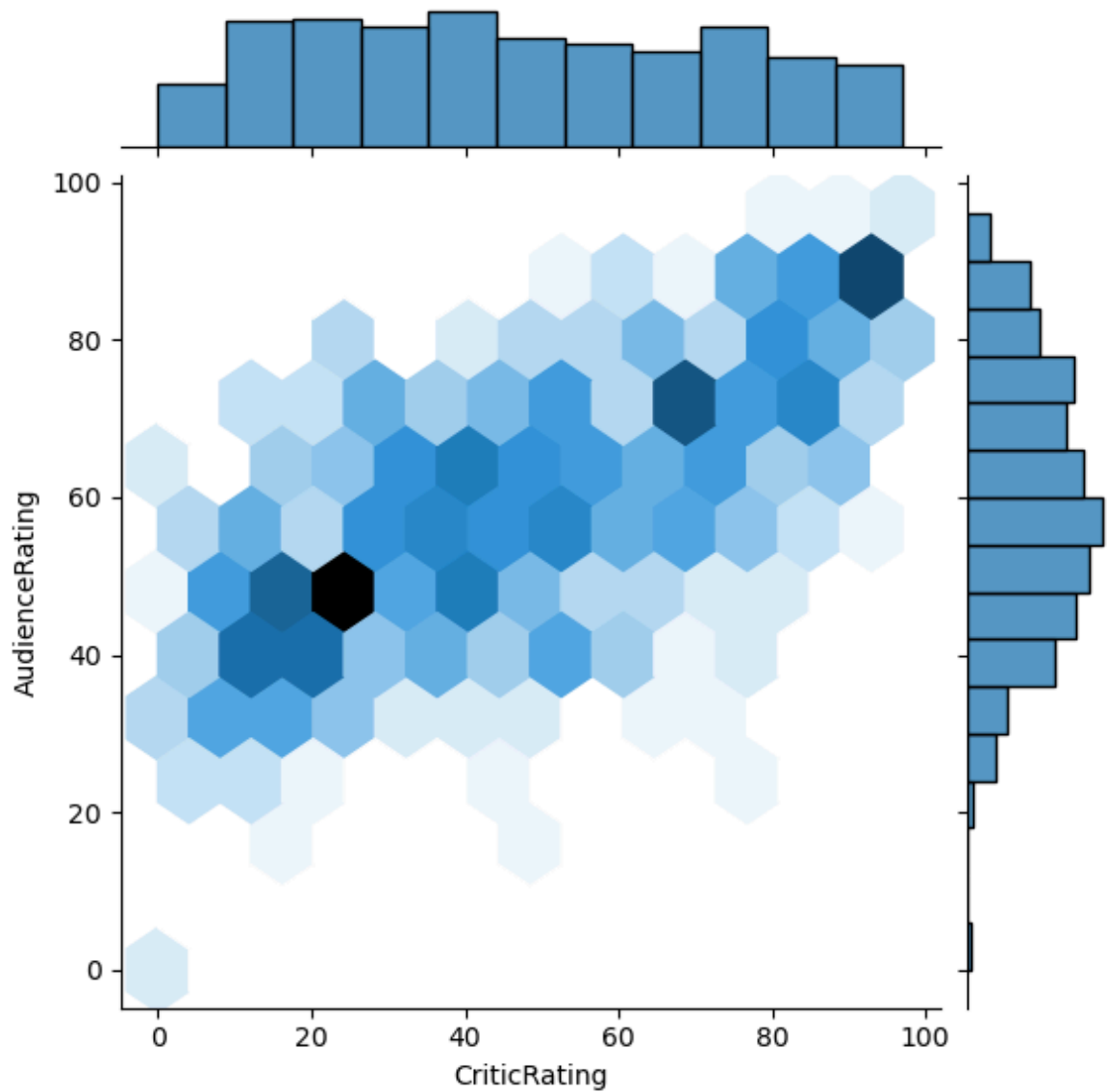
In [28]:
```python
#Histograms

# <<< chat1

m1 = sns.distplot(movies.AudienceRating)
plt.show()
#y - axis generated by seaborn automatically that is the powefull of seaborn gal
```
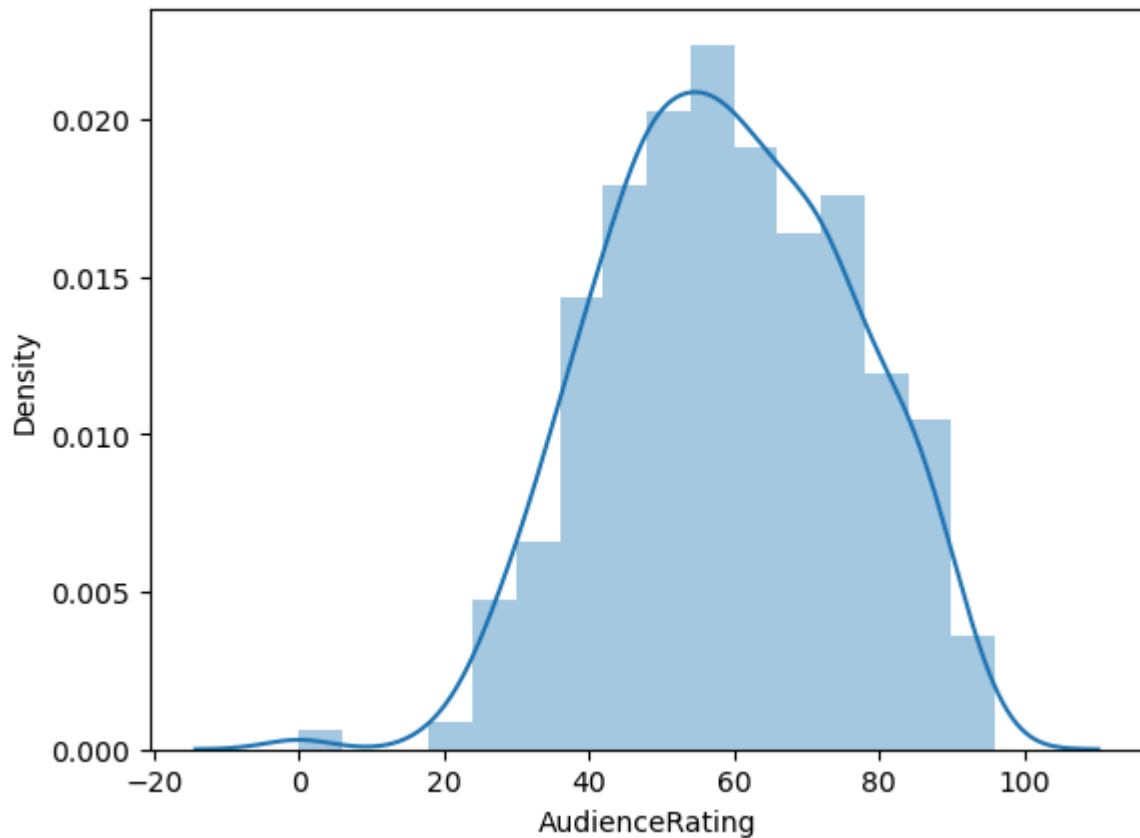
```
In [29]:  sns.set_style('darkgrid')
```

```
In [30]:  m2 = sns.distplot(movies.AudienceRating, bins = 15)
          plt.show()
```
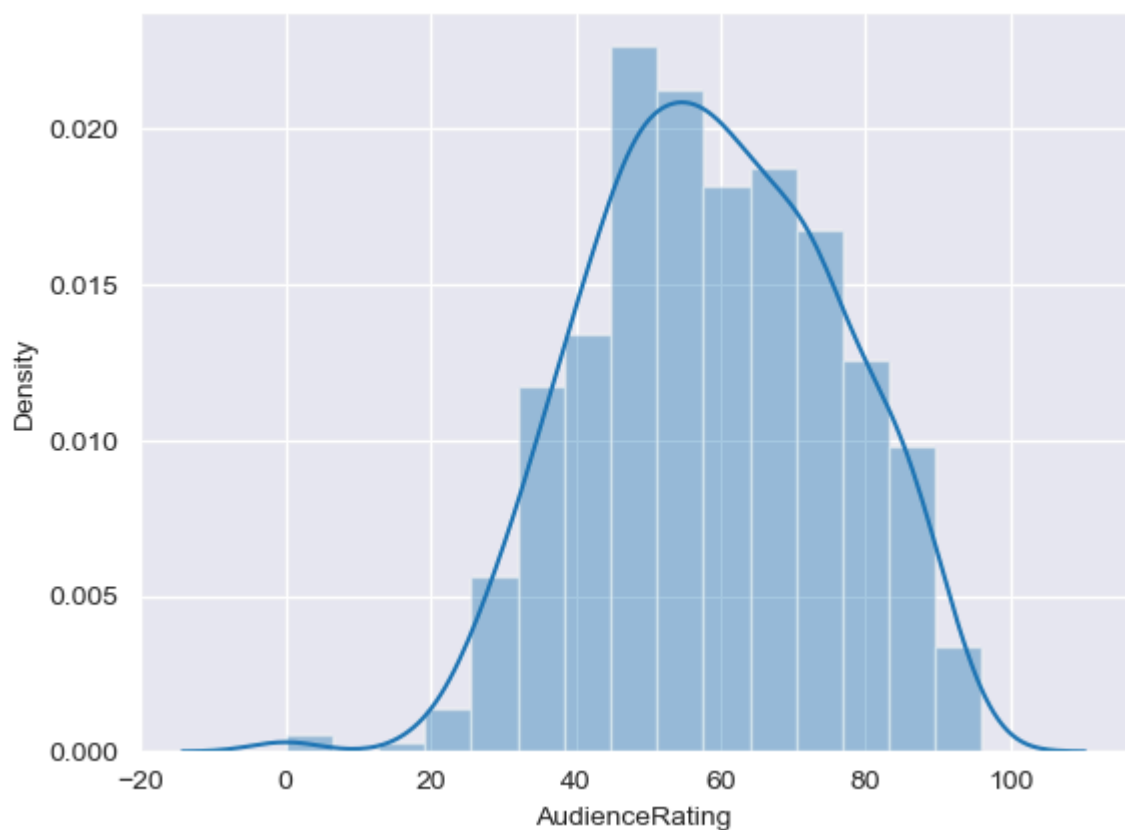


```
In [31]:  #sns.set_style('darkgrid')
          n1 = plt.hist(movies.AudienceRating, bins=15)
```

```python
plt.show()
```
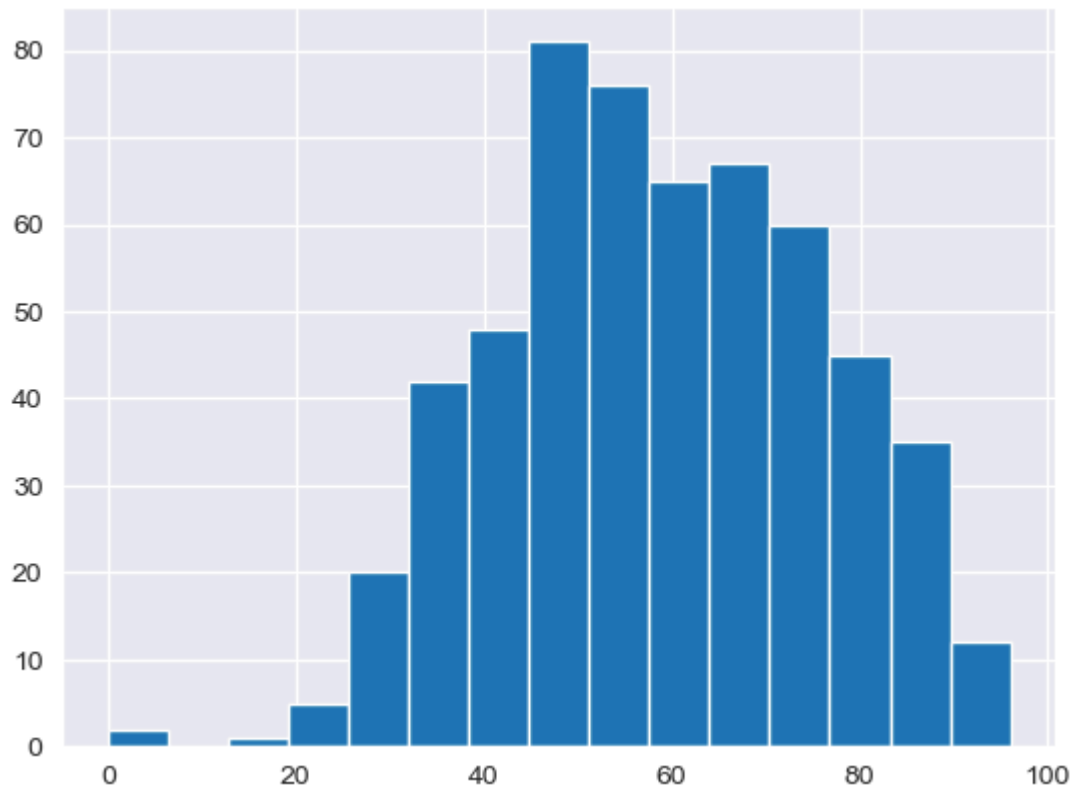


```python
In [32]: sns.set_style('white') #normal distribution & called as bell curve
         n1 = plt.hist(movies.AudienceRating, bins=20)
         plt.show()
```
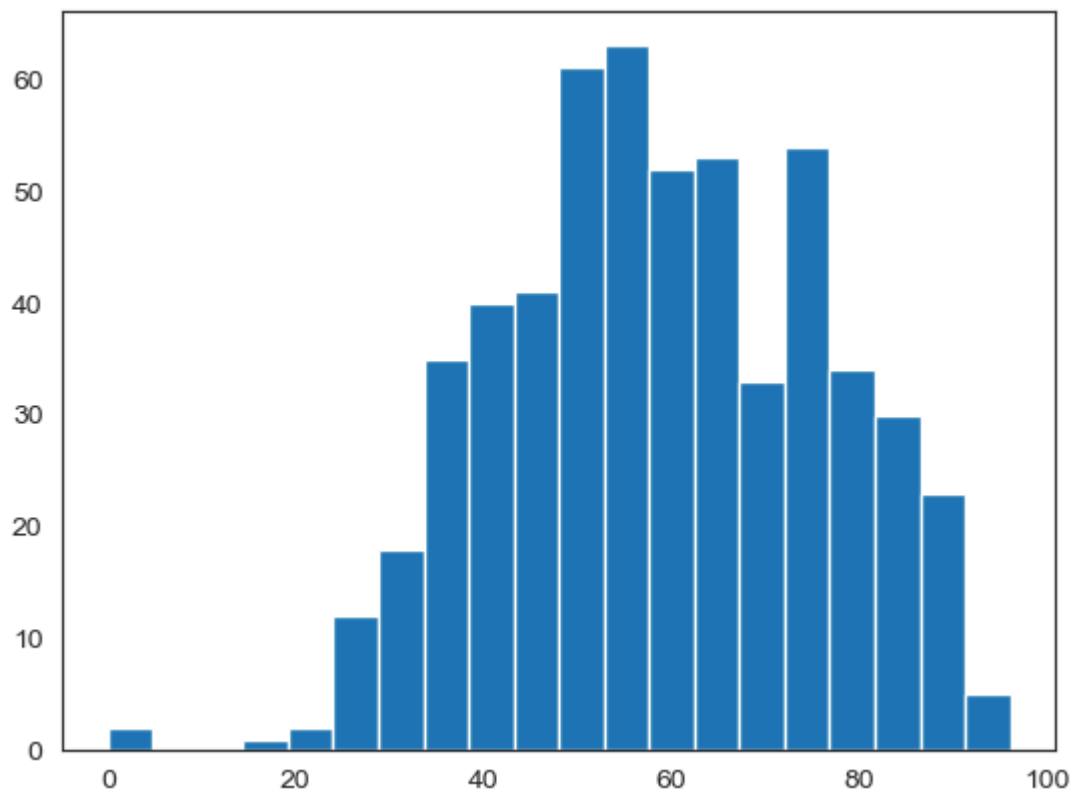


```python
In [33]: n1 = plt.hist(movies.CriticRating, bins=20) #uniform distribution
```

```python
In [34]: # <<< chat - 2
```

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

In [35]:
```python
#h1 = plt.hist(movies.BudgetMillions)

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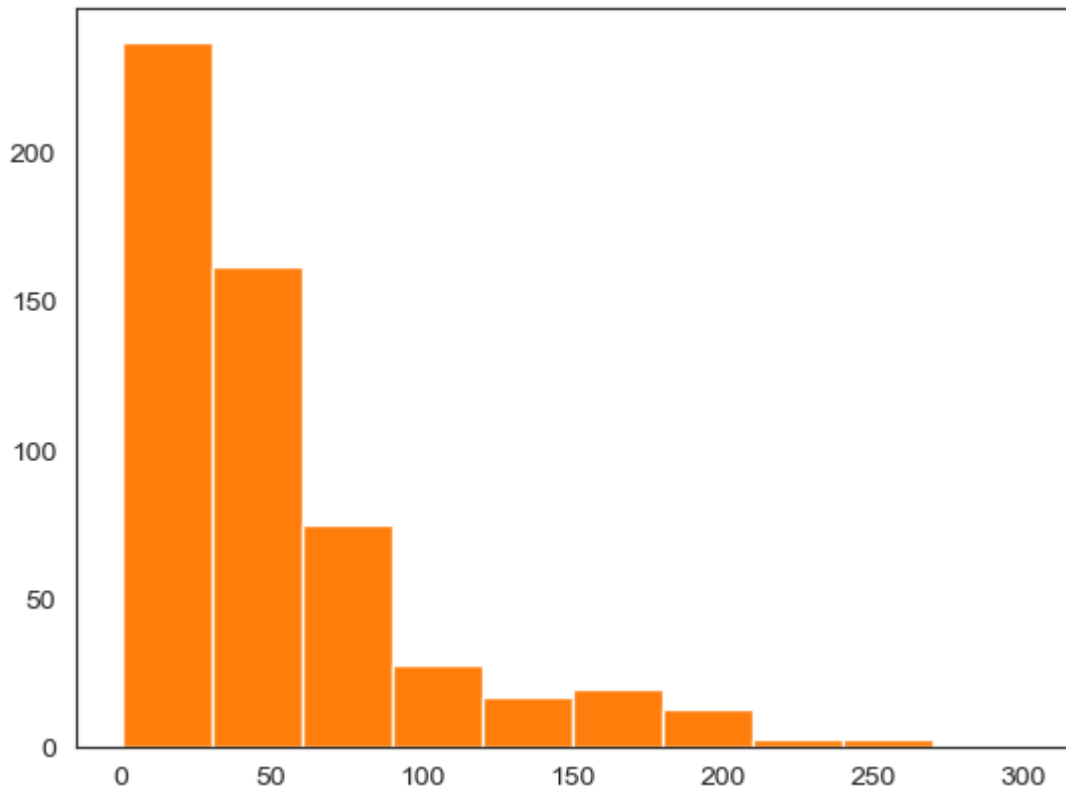

In [36]:
```python
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions)
plt.show()
```
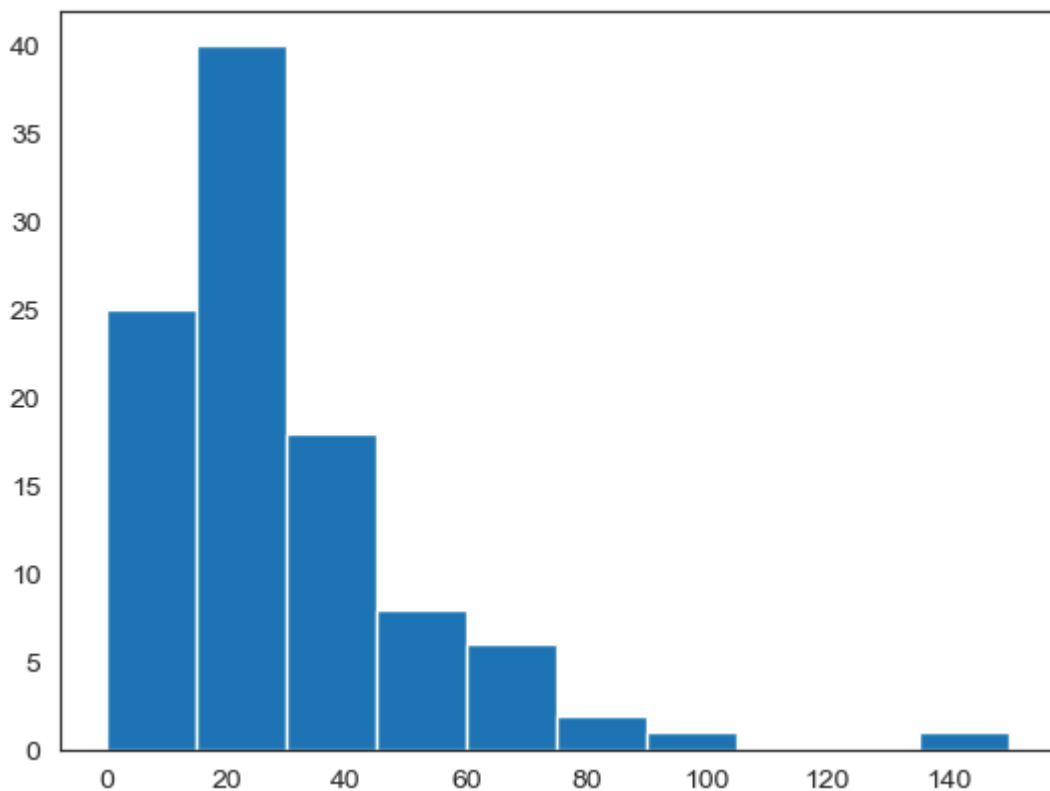
In [37]:
```python
movies.head()
```

Out[37]:

	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 [38]:
```python
#movies.Genre.unique()
```

In [39]:
```python
# Below plots are stacked histogram becuase overlaped

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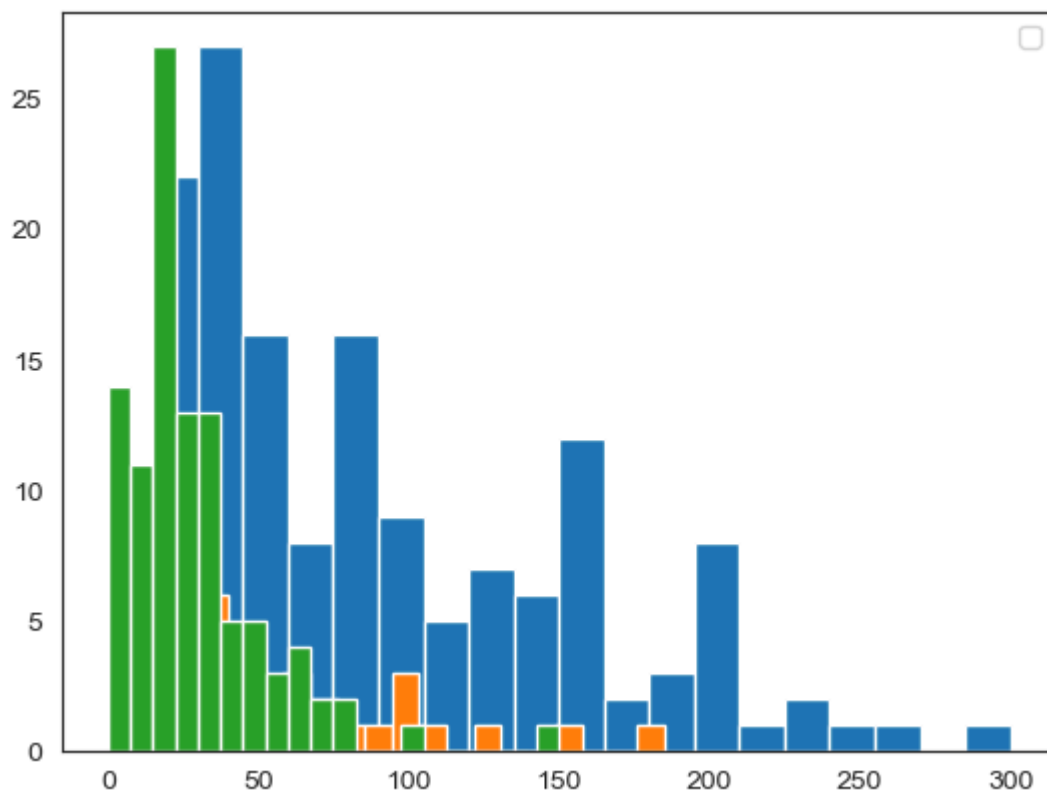

In [40]:
```python
plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,\
          movies[movies.Genre == 'Drama'].BudgetMillions, \
          movies[movies.Genre == 'Thriller'].BudgetMillions, \
          movies[movies.Genre == 'Comedy'].BudgetMillions],
       bins = 20, stacked = True)
plt.show()
```
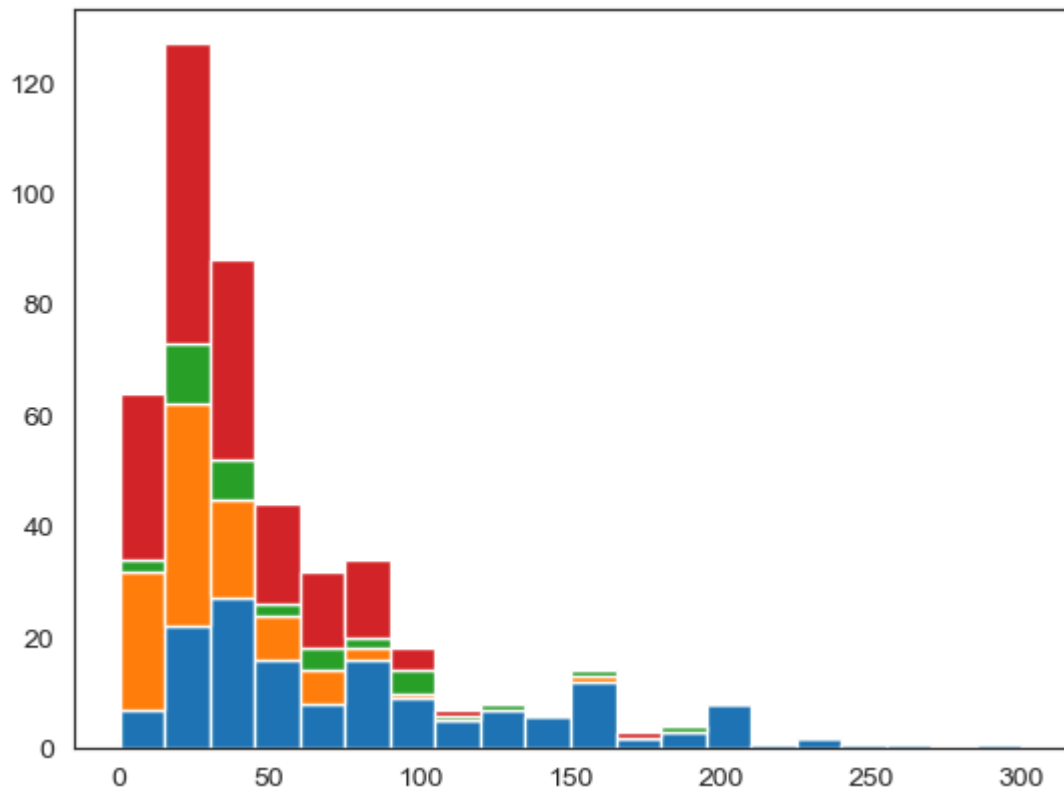
In [41]:
```python
# if you have 100 categories you cannot copy & paste all the things

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

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

In [42]:
```python
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False)
```

In [43]:
```python
plt.show()
```
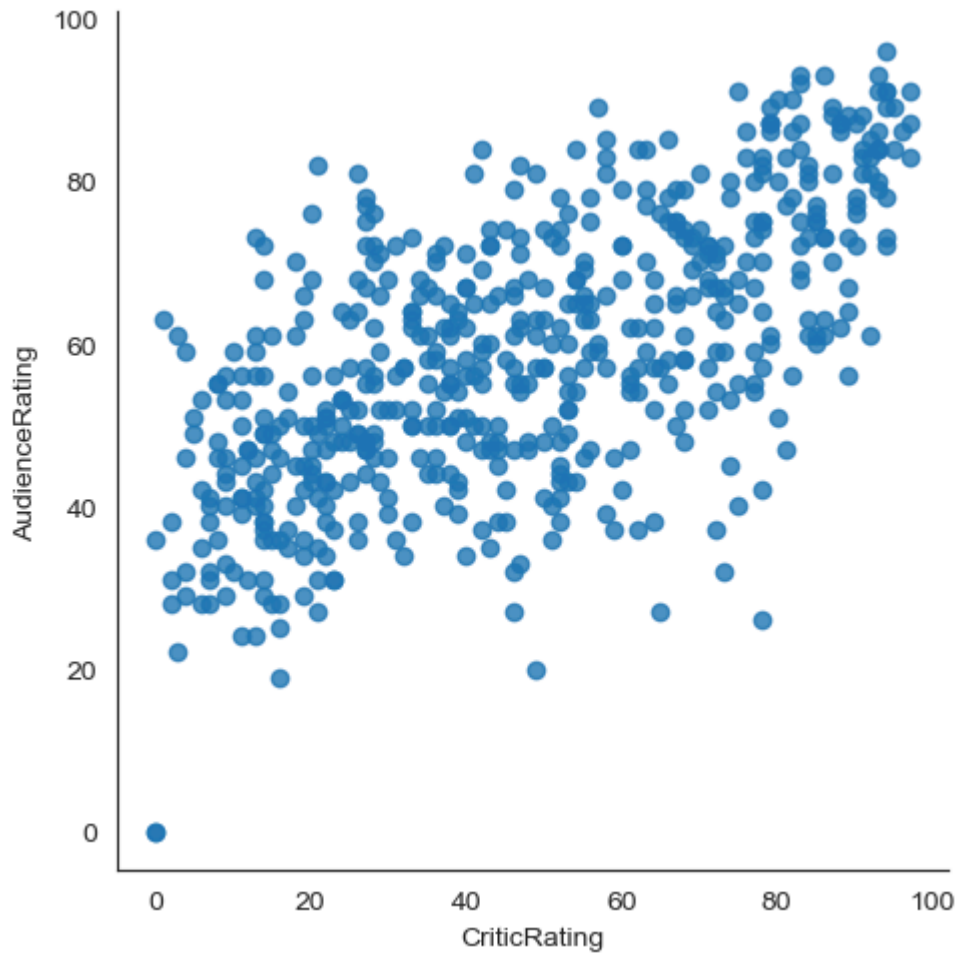
```
In [44]: vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                        fit_reg=False, hue = 'Genre')
```

```
In [45]: plt.show()
```
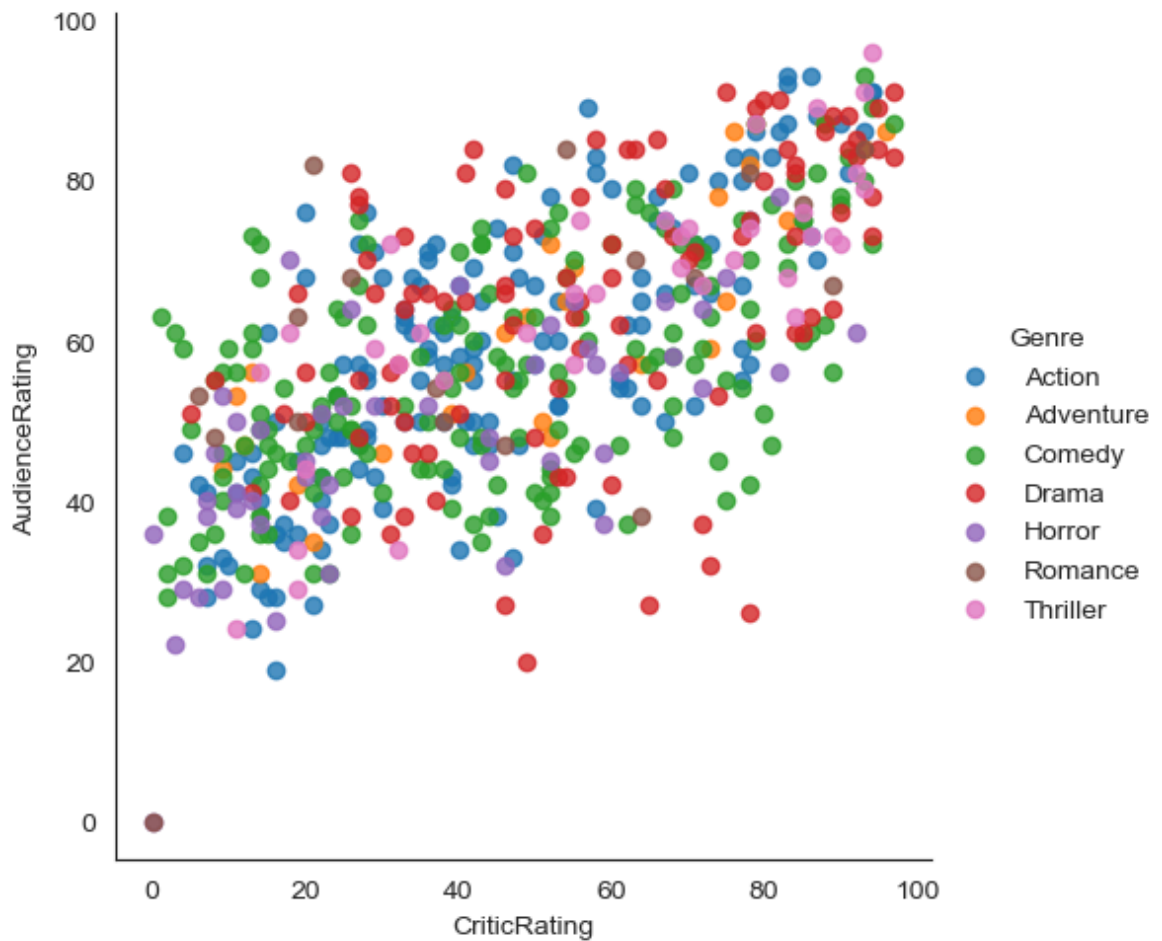
```
In [46]:  vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',
                            fit_reg=False, hue='Genre', height=10, aspect=1)
```

```
In [47]:  plt.show()
```
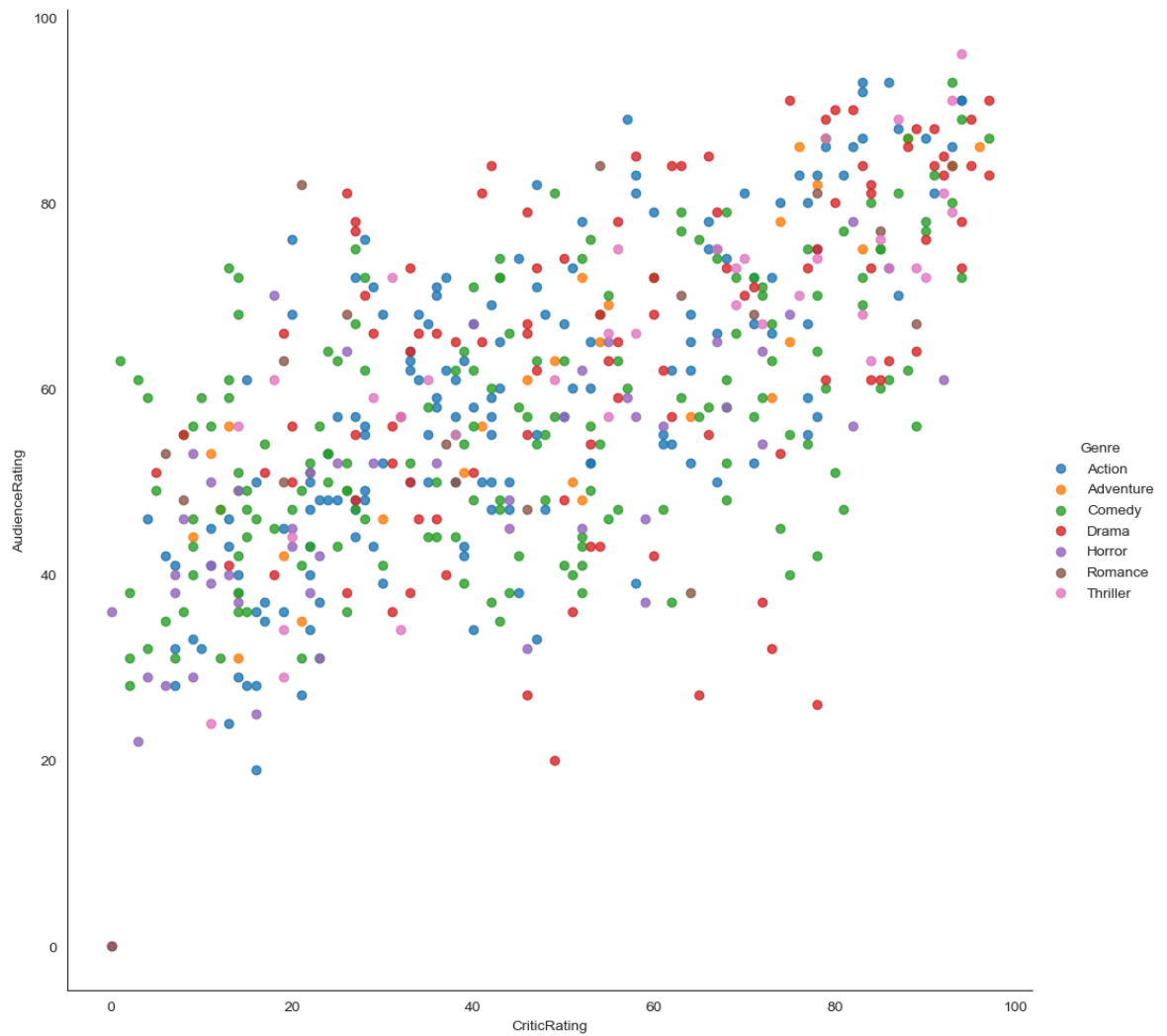
In [48]:
```
# Kernal Density Estimate plot ( KDE PLOT)
# how can i visulize audience rating & critics rating . using scatterplot
```

In [49]:
```
k1 = sns.kdeplot(x=movies.CriticRating, y=movies.AudienceRating)


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

In [50]:
```
plt.show()
```
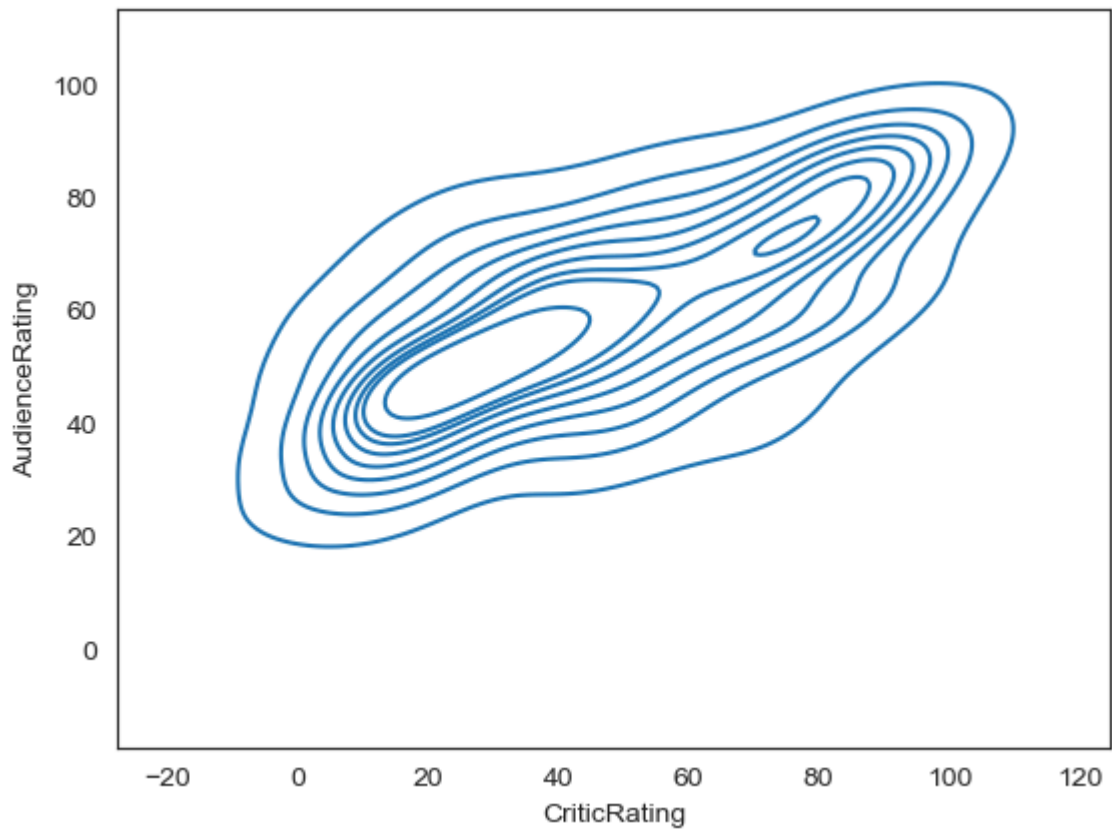
```
In [53]:  k1 = sns.kdeplot(x=movies.CriticRating, y=movies.AudienceRating, fill=True, cmap
```

```
In [54]:  plt.show()
```
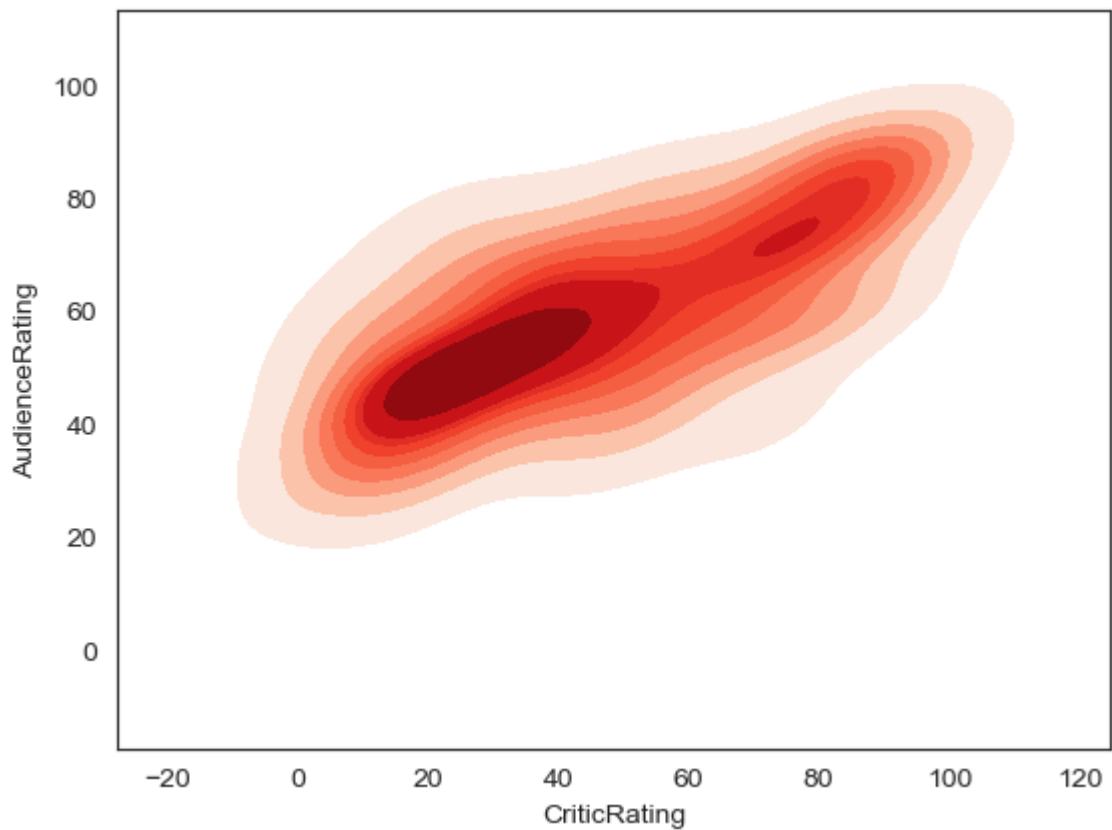


```
In [56]:  k2 = sns.kdeplot(x=movies.CriticRating, y=movies.AudienceRating, fill=True, cmap
```

```
In [57]:  plt.show()
```
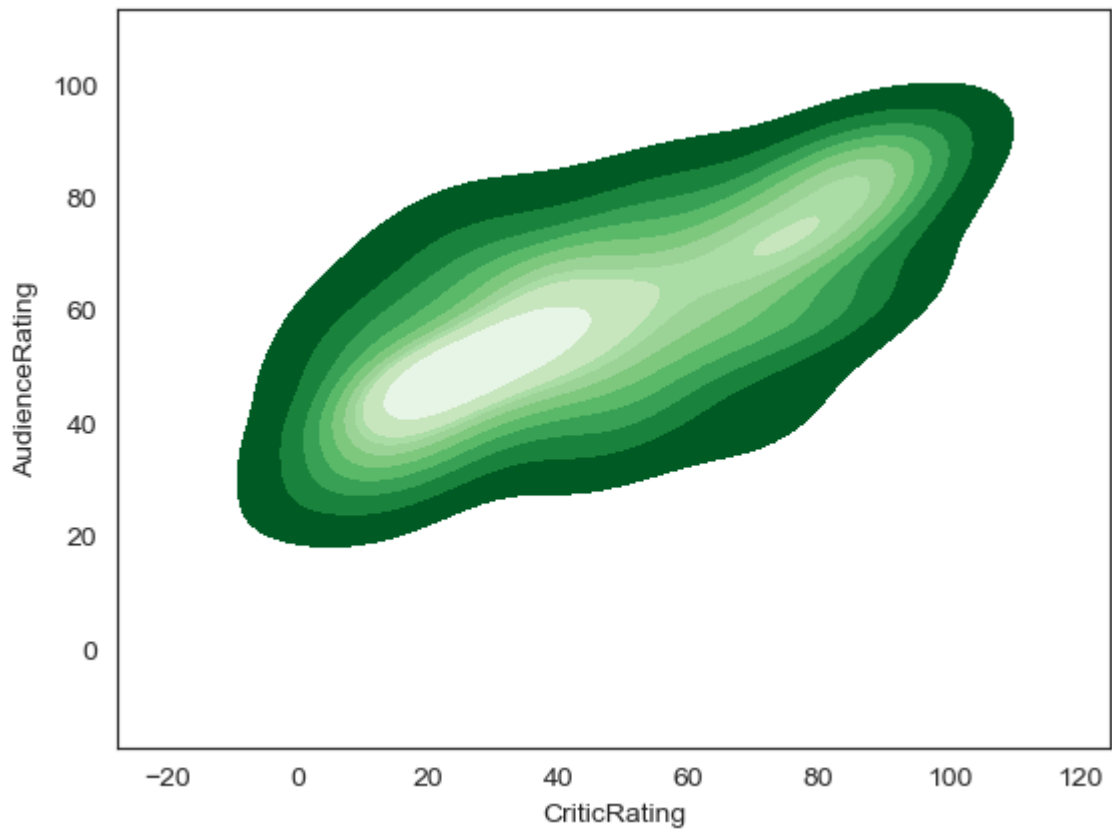
```
In [59]:  sns.set_style('dark')
          k1 = sns.kdeplot(x=movies.BudgetMillions,y=movies.AudienceRating,shade_lowest=Fa
```

```
In [60]:  plt.show()
```
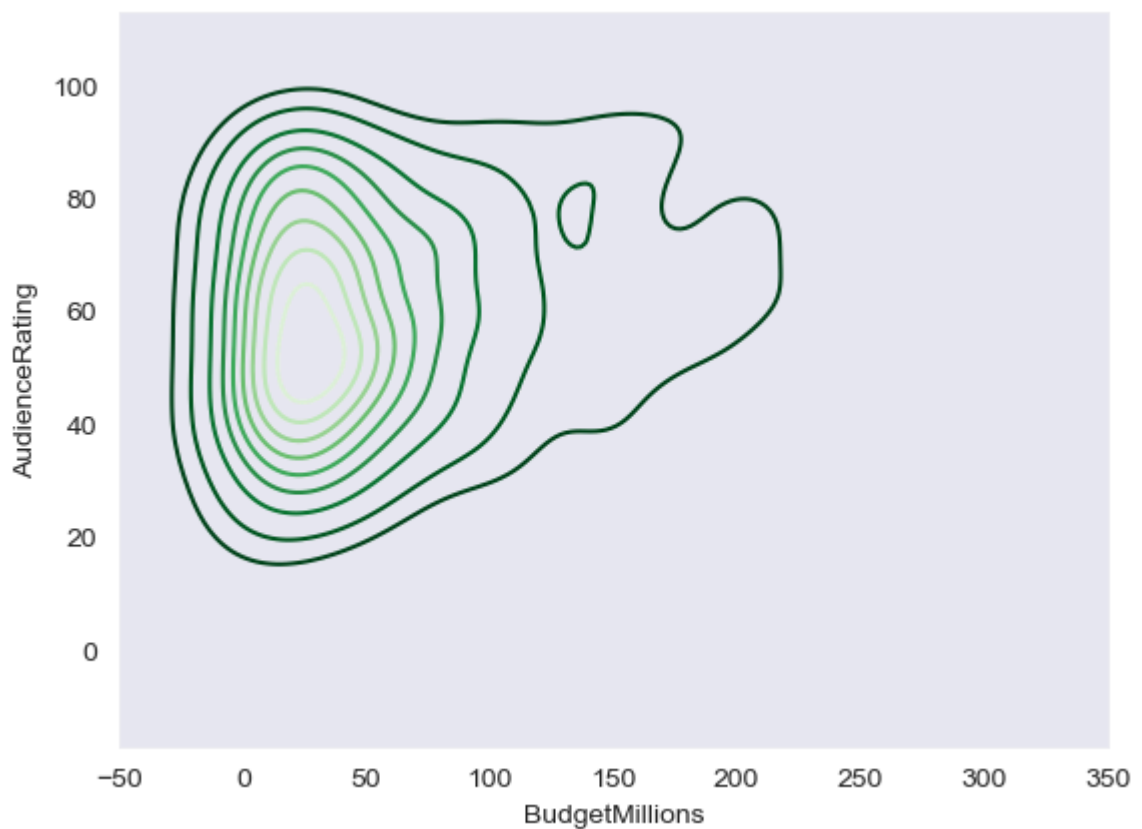


```
In [61]:  sns.set_style('dark')
          k1 = sns.kdeplot(x=movies.BudgetMillions,y=movies.AudienceRating)
```

In [62]: `plt.show()`
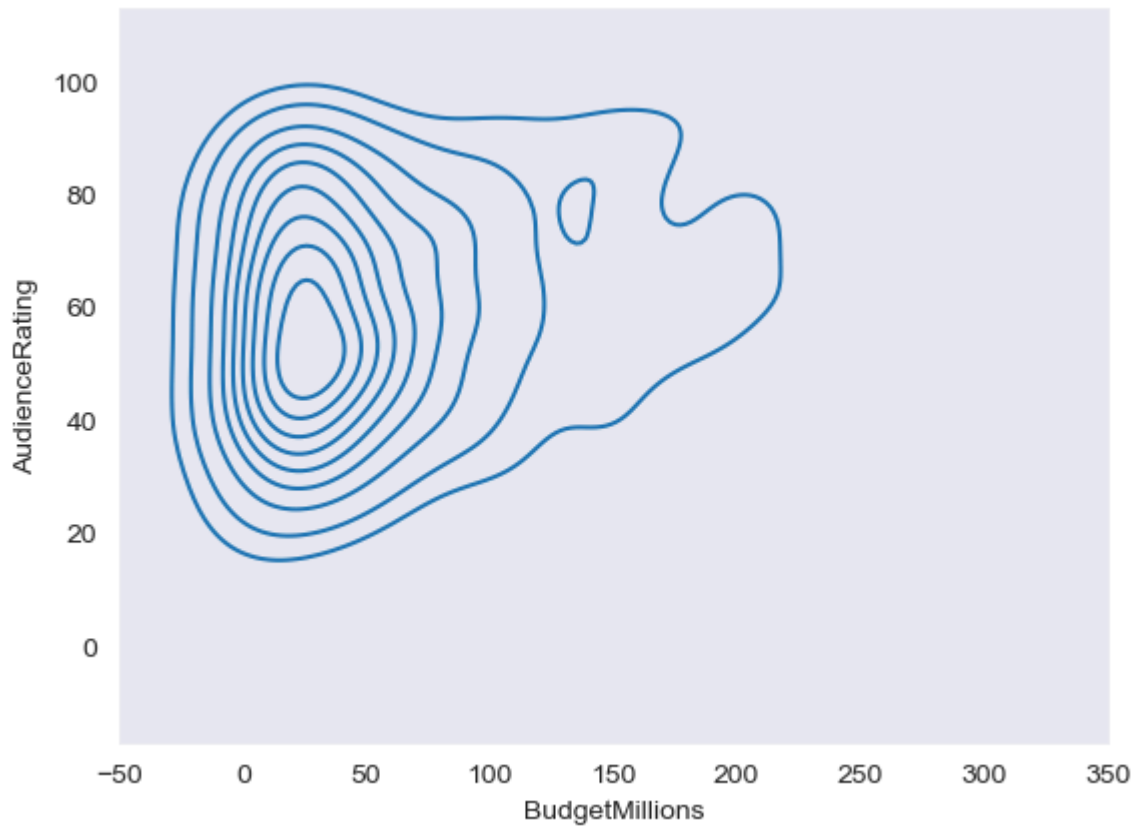
In [64]: `k2 = sns.kdeplot(x=movies.BudgetMillions,y=movies.CriticRating)`

In [65]: `plt.show()`
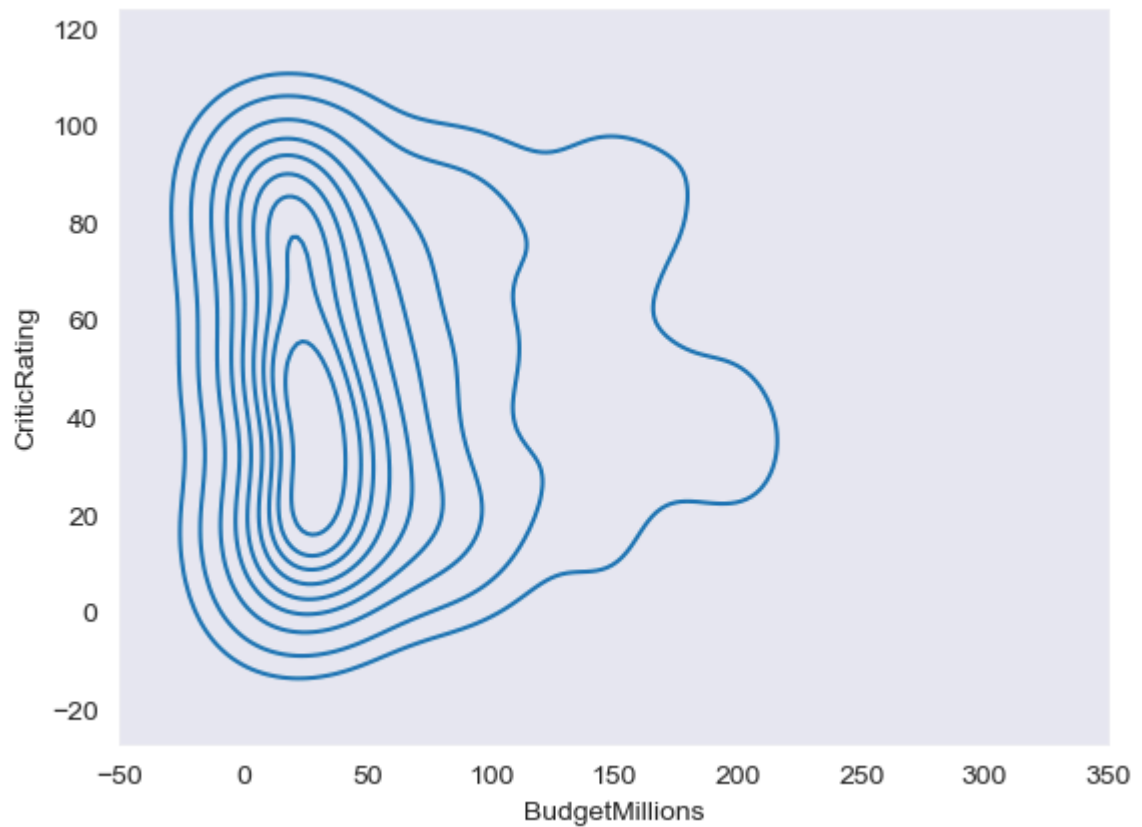
In [66]: `#subplots`

```python
f, ax = plt.subplots(1,2, figsize =(12,6))
#f, ax = plt.subplots(3,3, figsize =(12,6))
```

In [67]: `plt.show()`
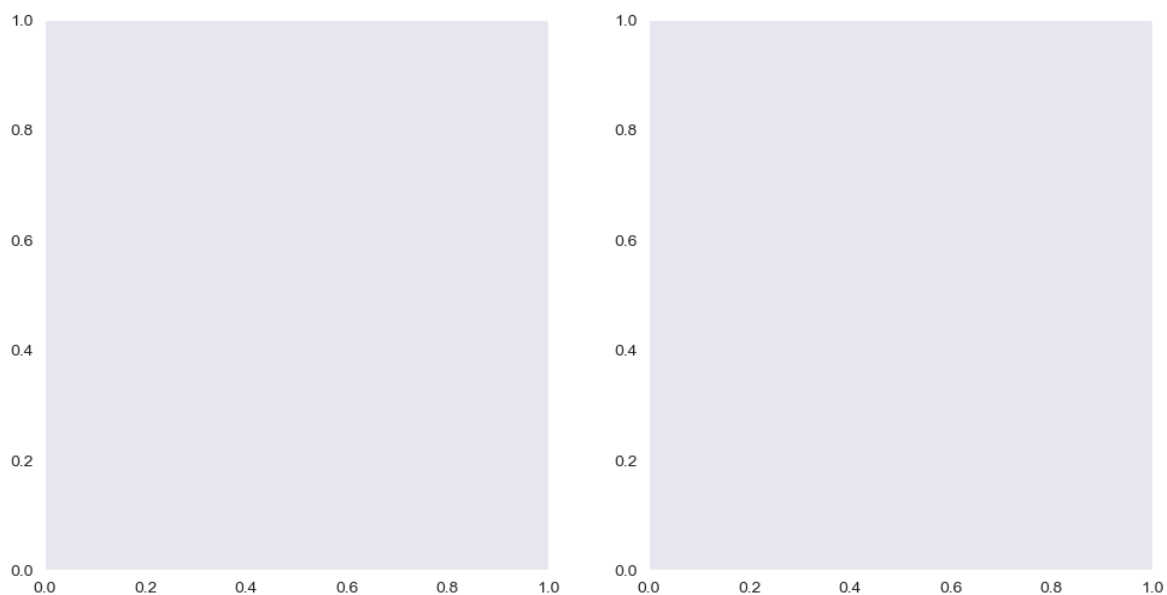
In [68]:
```python
f, axes = plt.subplots(1,2, figsize =(12,6))

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

In [69]: `plt.show()`
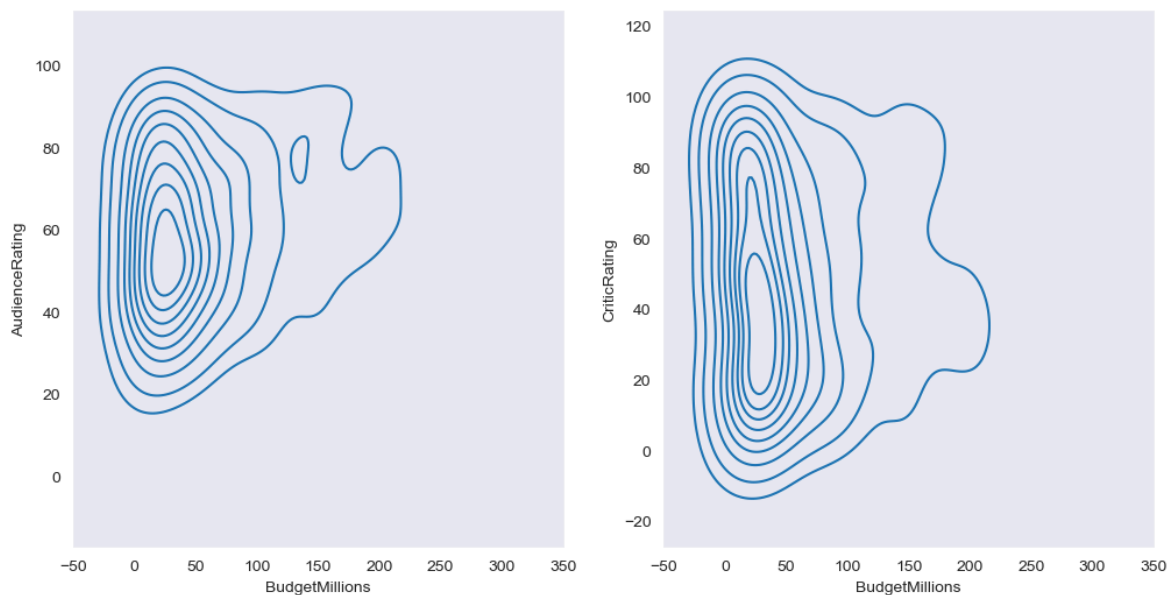
In [70]: `axes`

Out[70]:
```
array([<Axes: xlabel='BudgetMillions', ylabel='AudienceRating'>,
       <Axes: xlabel='BudgetMillions', ylabel='CriticRating'>],
      dtype=object)
```

In [77]:
```python
#Box plots -

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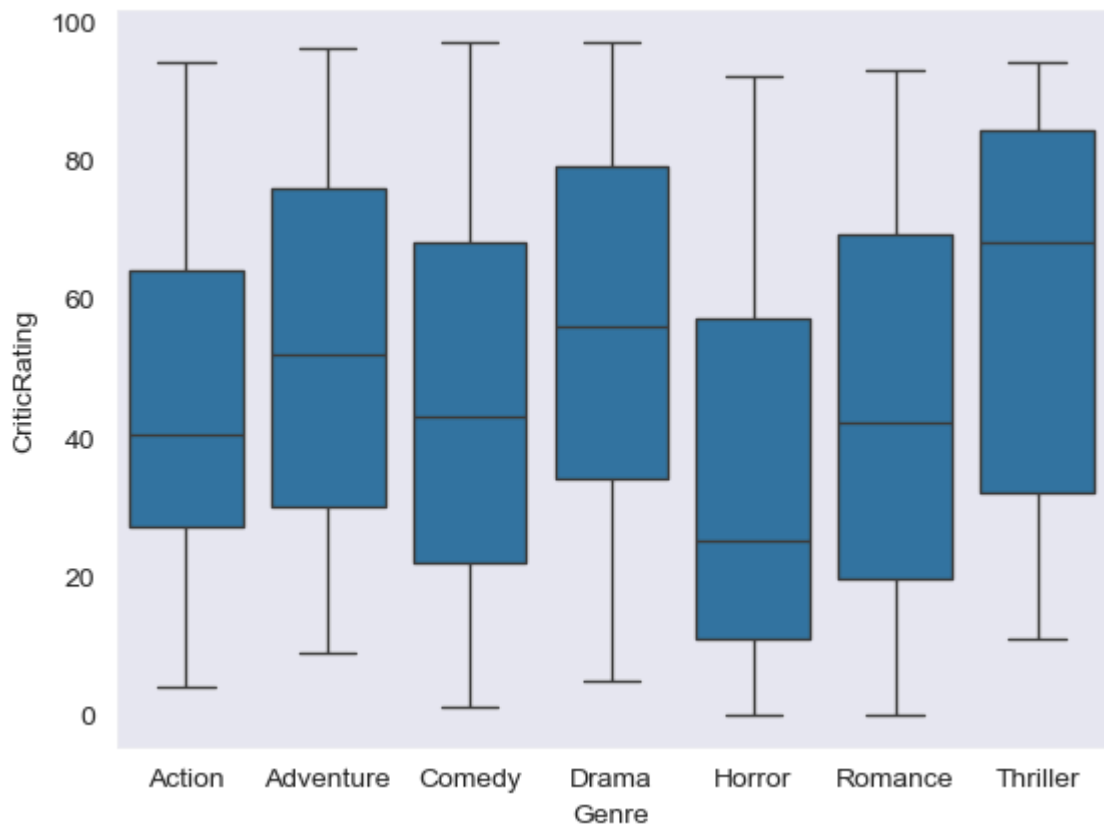

In [78]:
```python
#violin plot

z = sns.violinplot(data=movies, x='Genre', y = 'CriticRating')
plt.show()
```
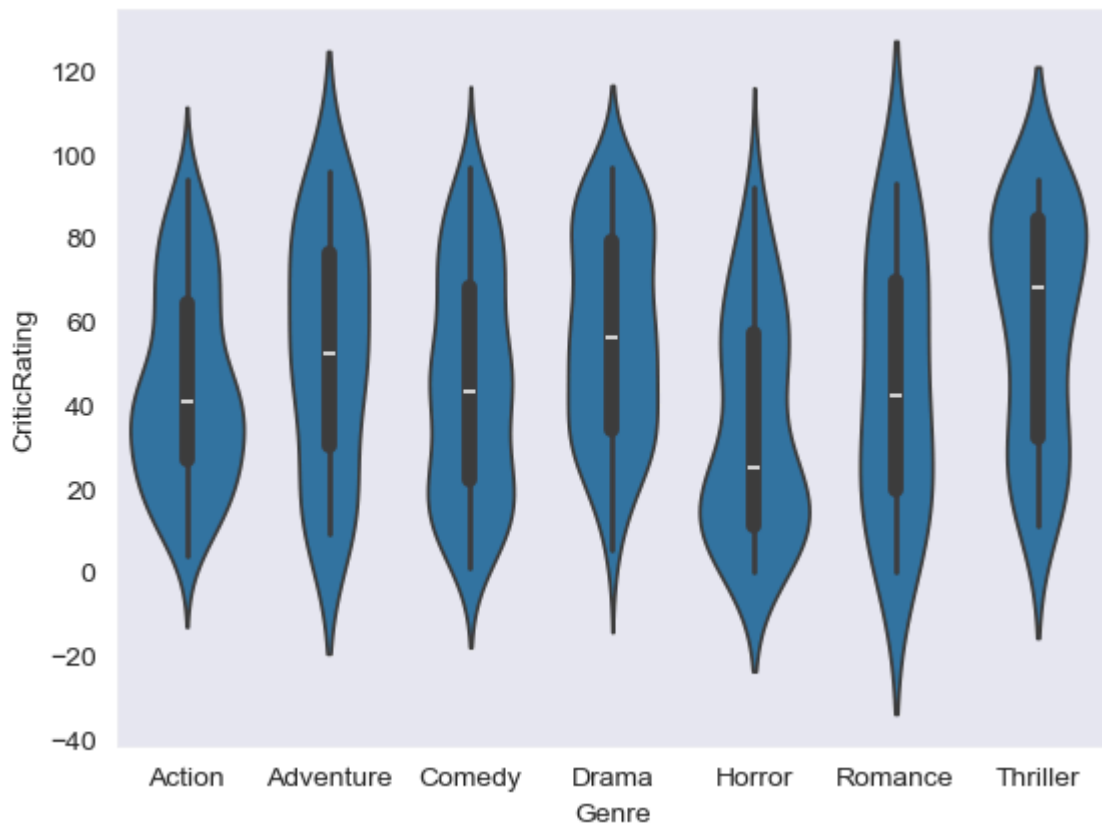
```
In [79]: w1 = sns.boxplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRati
         plt.show()
```
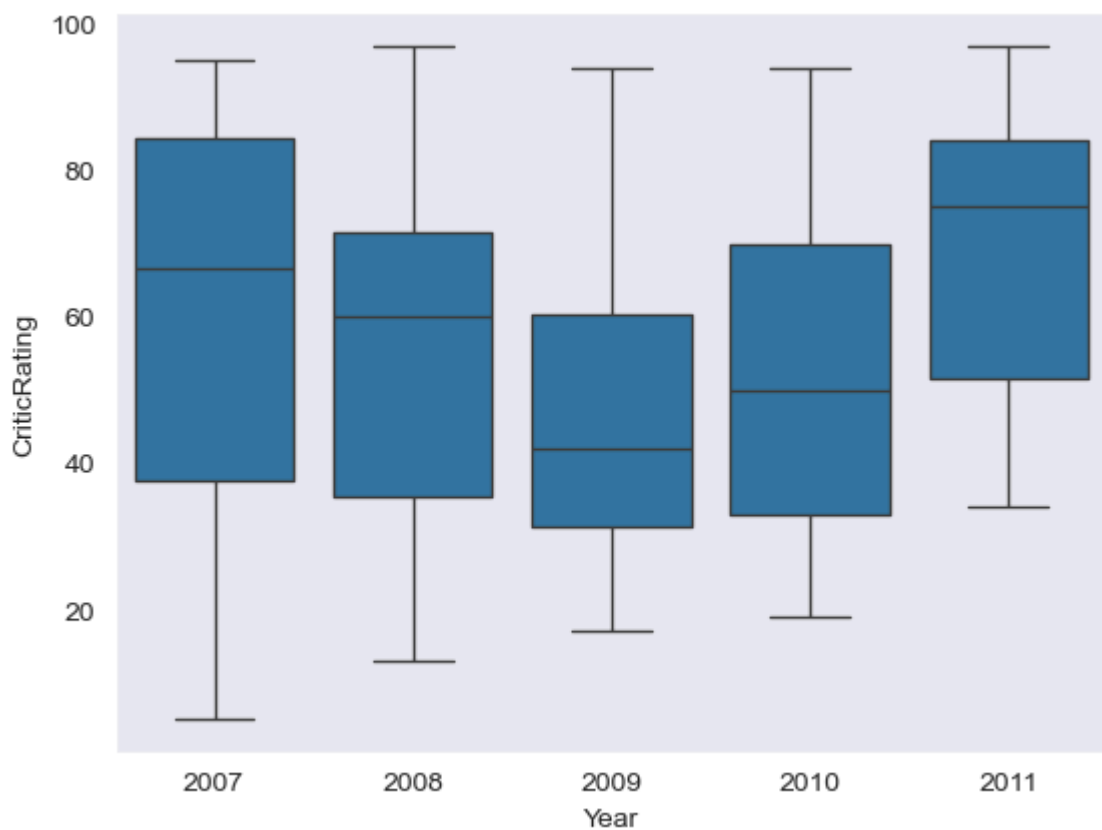


```
In [80]: z = sns.violinplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRa
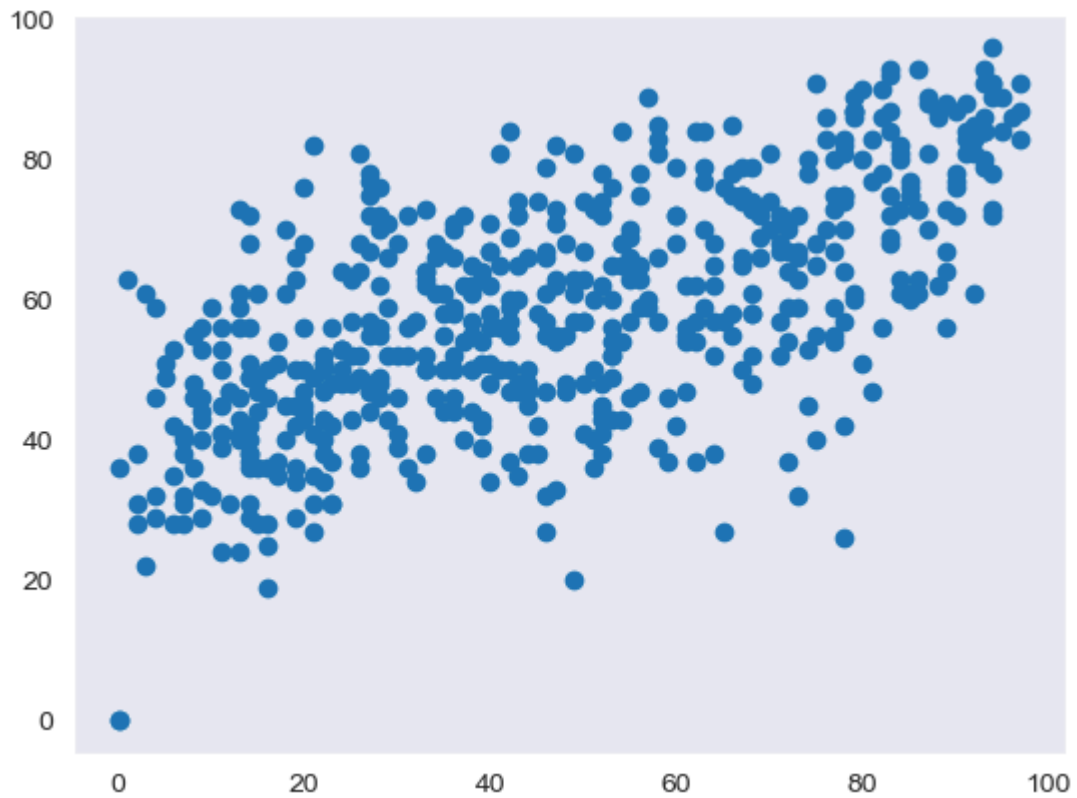         plt.show()
```
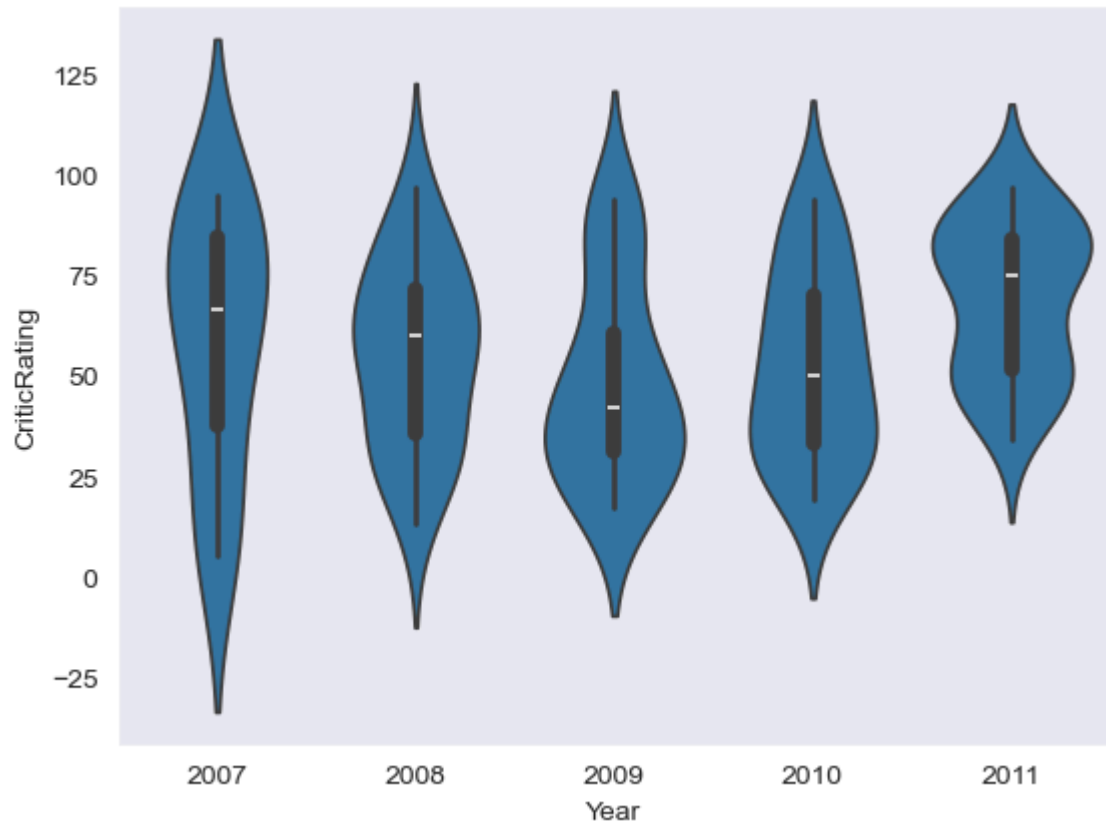
In [81]: *# Createing a Facet grid*

In [82]:
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre') #kind of s
plt.show()
```
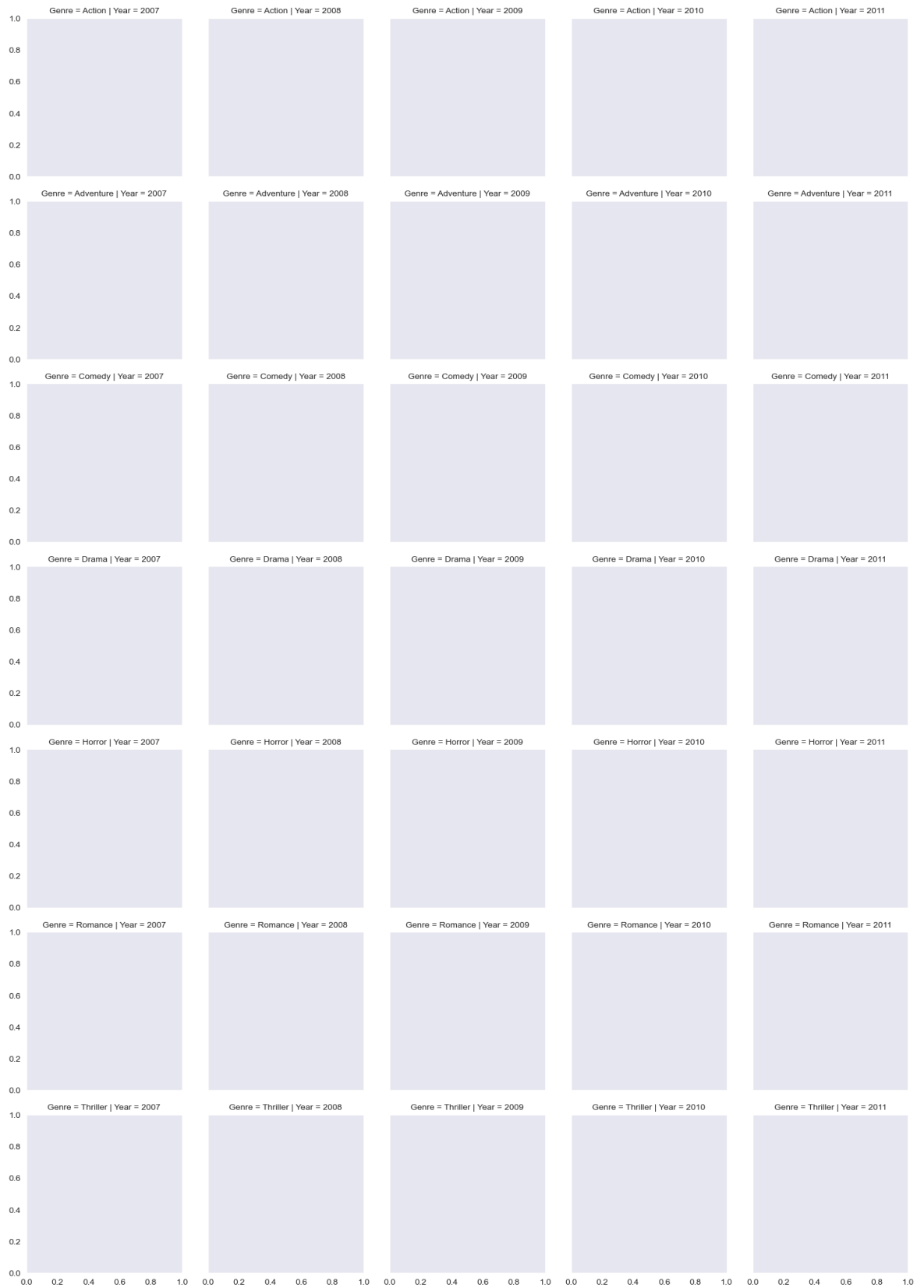
```
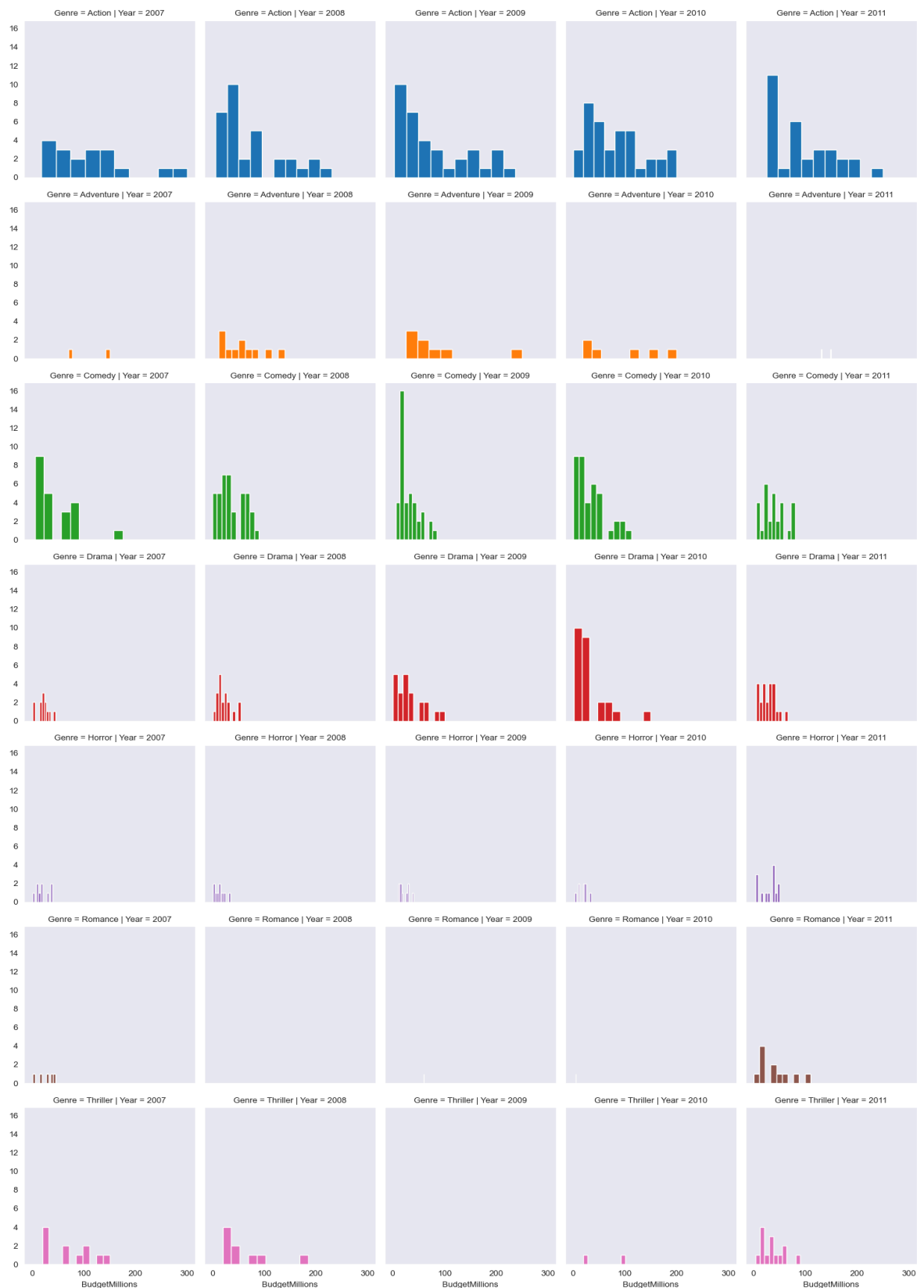In [83]: plt.scatter(movies.CriticRating,movies.AudienceRating)
         plt.show()
```

```
In [84]:  g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
          g = g.map(plt.scatter, 'CriticRating', 'AudienceRating' ) #scatterplots are mapp
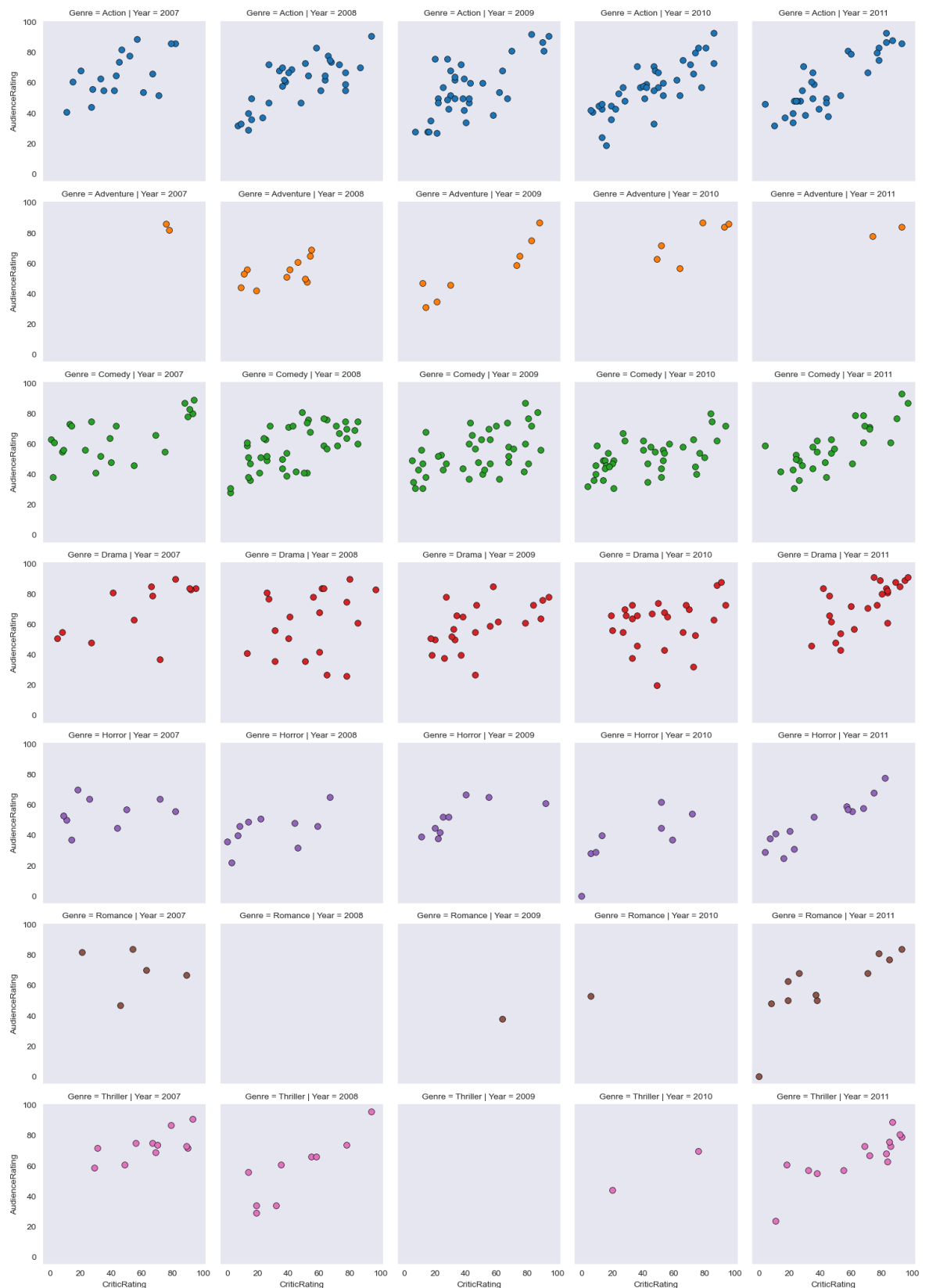          plt.show()
```
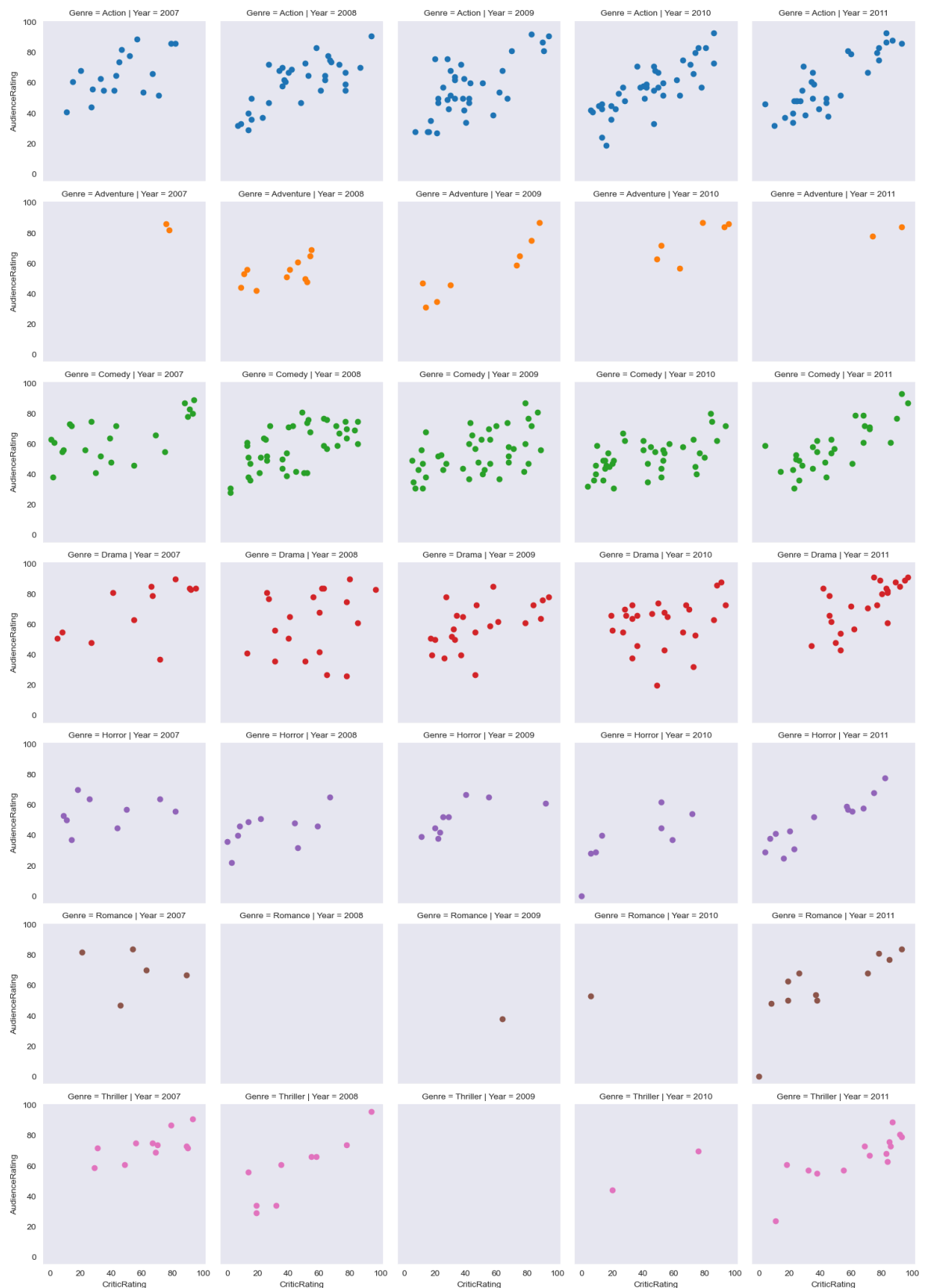
```
In [85]:   # you can populated any type of chat.

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

```
In [86]: #
         g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
         kws = dict(s=50, linewidth=0.5,edgecolor='black')
         g = g.map(plt.scatter, 'CriticRating', 'AudienceRating',**kws ) #scatterplots ar
         plt.show()
```

```python
In [88]: import seaborn as sns
         import matplotlib.pyplot as plt

         sns.set_style('darkgrid')
         f, axes = plt.subplots(2, 2, figsize=(15, 15))

         # KDE Plot 1
         k1 = sns.kdeplot(
             x=movies.BudgetMillions,
             y=movies.AudienceRating,
```

```python
        ax=axes[0, 0]
)

# KDE Plot 2
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 for Drama genre
z = sns.violinplot(
    data=movies[movies.Genre == 'Drama'],
    x='Year',
    y='CriticRating',
    ax=axes[1, 0]
)

# KDE Plot 3 with fill
k4 = sns.kdeplot(
    x=movies.CriticRating,
    y=movies.AudienceRating,
    fill=True,
    cmap='Reds',
    thresh=0.05,
    ax=axes[1, 1]
)

# KDE contour lines (optional overlay)
k4b = sns.kdeplot(
    x=movies.CriticRating,
    y=movies.AudienceRating,
    cmap='Reds',
    ax=axes[1, 1]
)

plt.tight_layout()
plt.show()
```
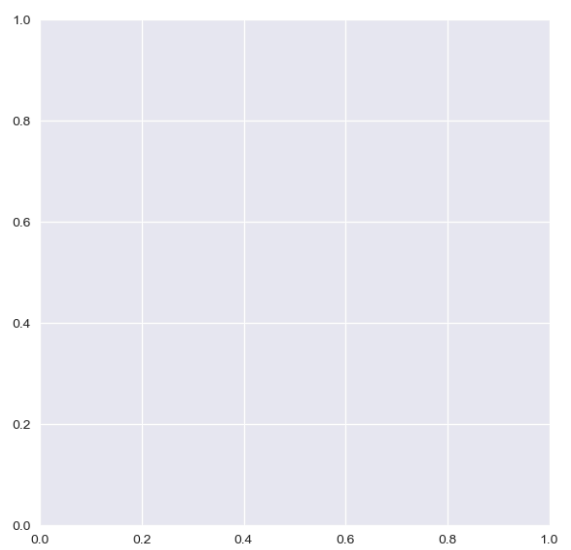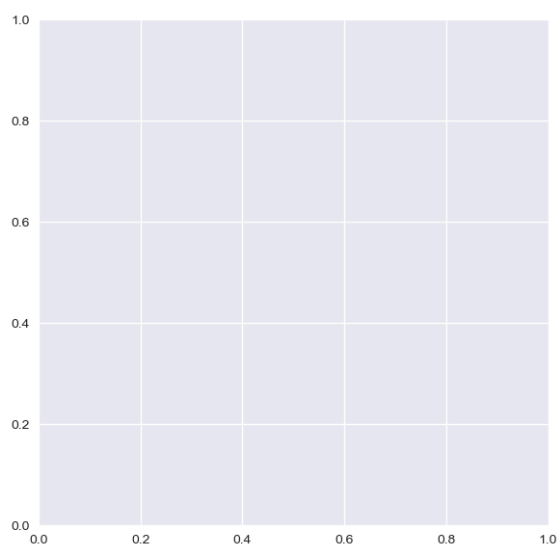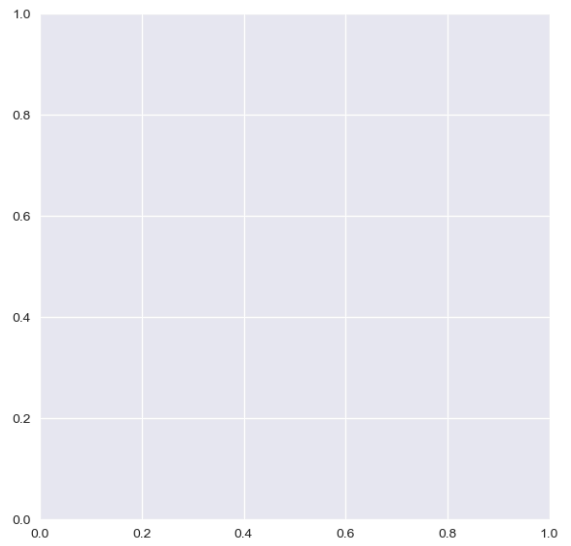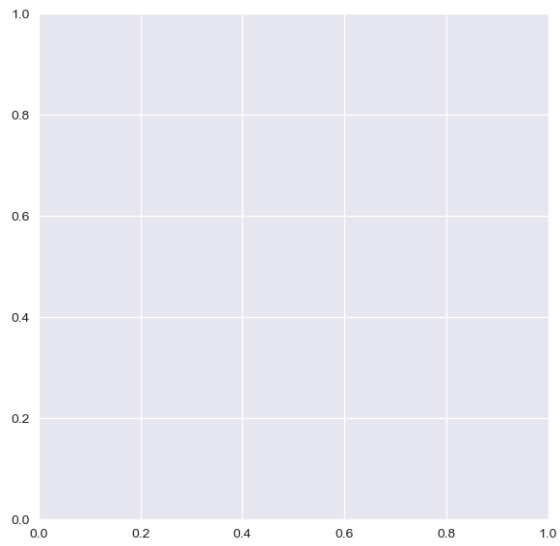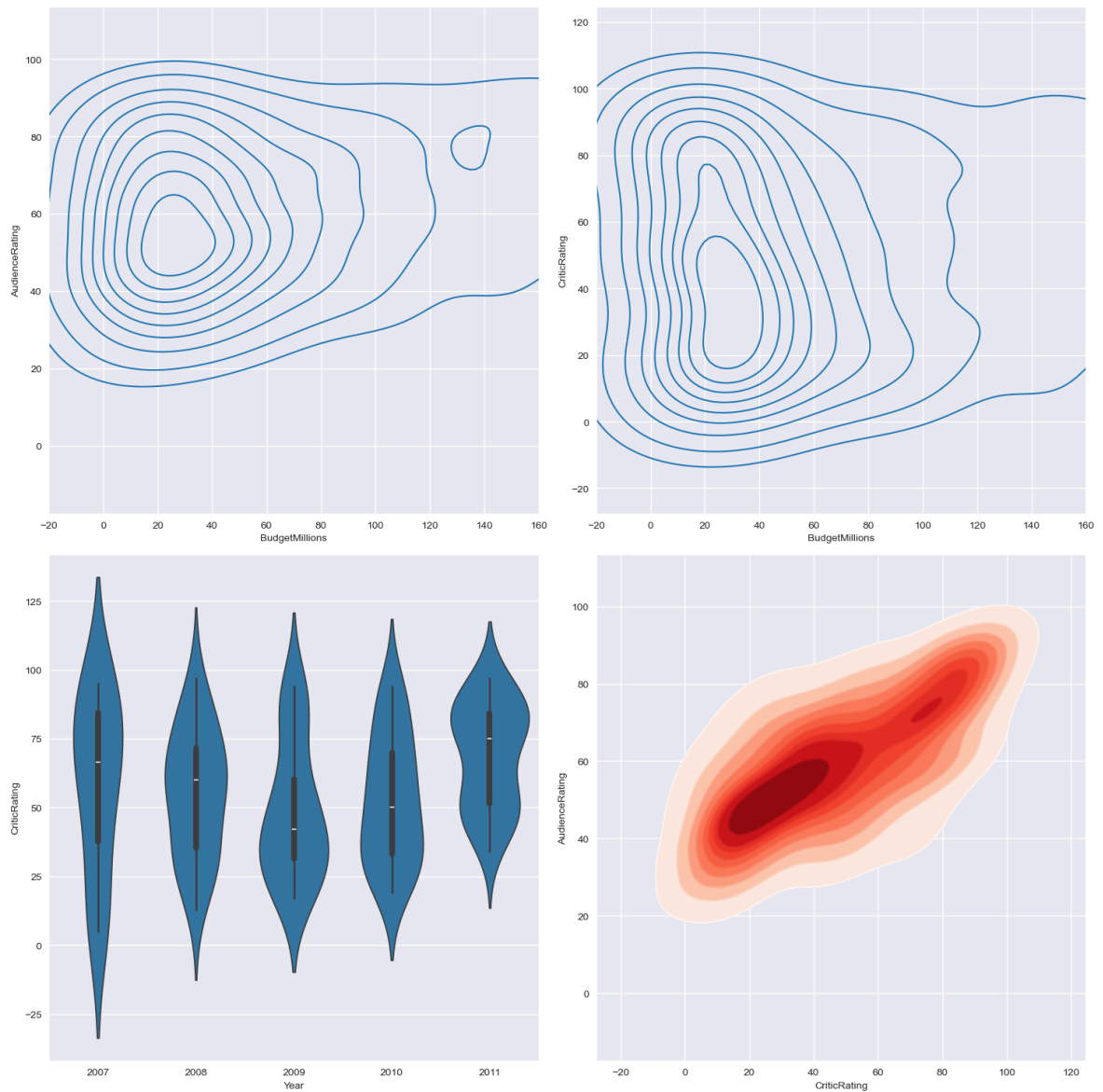
```
In [90]:   import seaborn as sns
           import matplotlib.pyplot as plt

           # Set dark background and style
           sns.set_style('dark', {'axes.facecolor': 'black'})

           # Create subplot grid
           f, axes = plt.subplots(2, 2, figsize=(15, 15))

           # Plot [0,0] - Budget vs AudienceRating
           k1 = sns.kdeplot(
               x=movies.BudgetMillions,
               y=movies.AudienceRating,
               fill=True,
               cmap='inferno',
               thresh=0.05,
               ax=axes[0, 0]
           )
           k1b = sns.kdeplot(
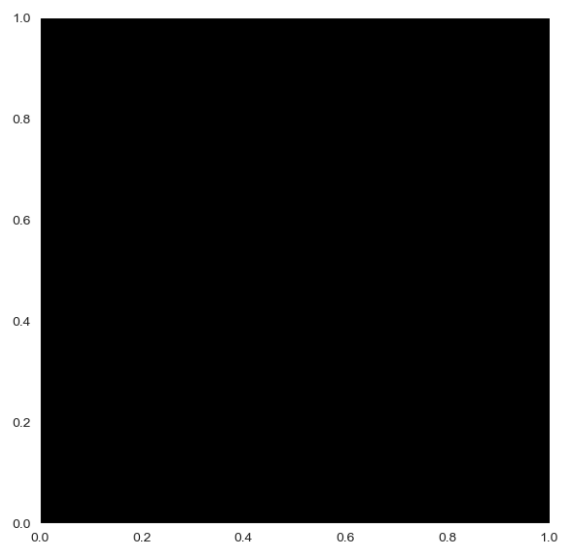               x=movies.BudgetMillions,
               y=movies.AudienceRating,
               cmap='cool',
               ax=axes[0, 0]
           )
```

```python
# Plot [0,1] - Budget vs CriticRating
k2 = sns.kdeplot(
    x=movies.BudgetMillions,
    y=movies.CriticRating,
    fill=True,
    cmap='inferno',
    thresh=0.05,
    ax=axes[0, 1]
)
k2b = sns.kdeplot(
    x=movies.BudgetMillions,
    y=movies.CriticRating,
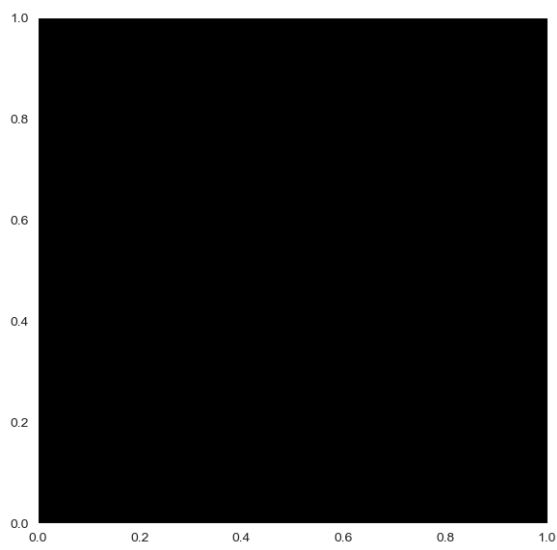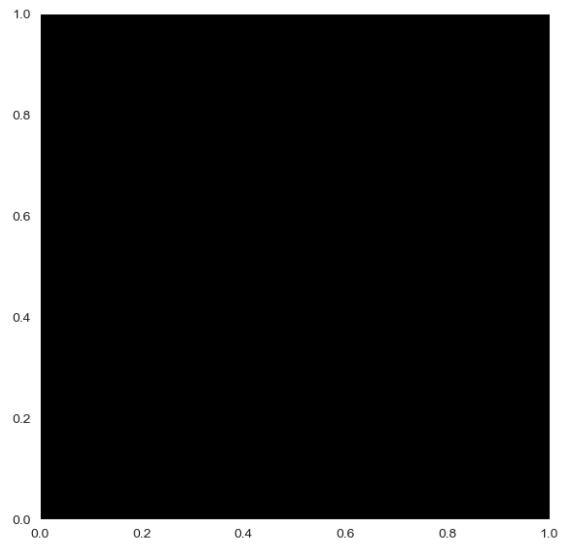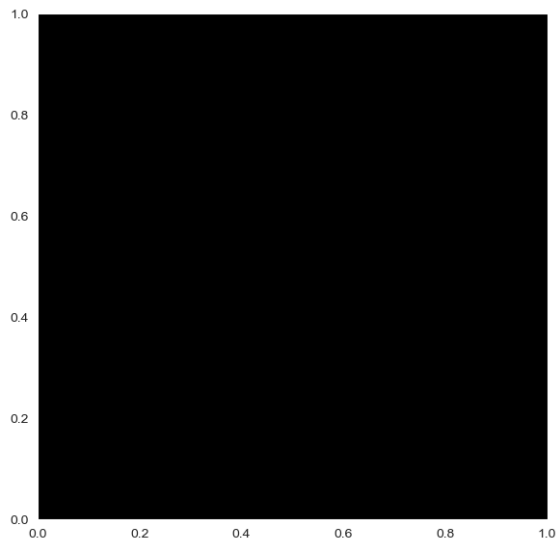    cmap='cool',
    ax=axes[0, 1]
)

# Set x-limits for both top plots
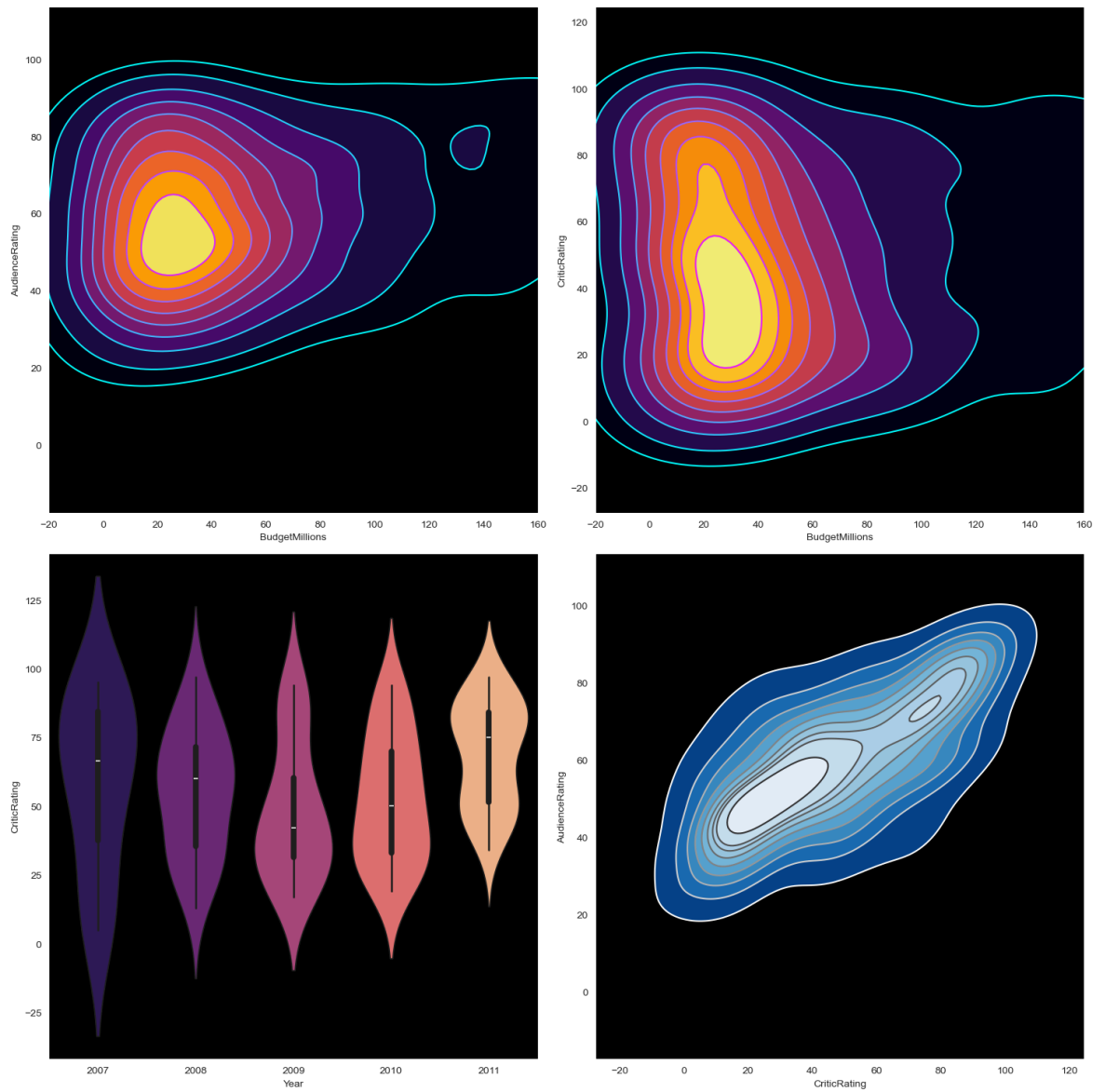k1.set(xlim=(-20, 160))
k2.set(xlim=(-20, 160))

# Plot [1,0] - Violin plot for Drama
z = sns.violinplot(
    data=movies[movies.Genre == 'Drama'],
    x='Year',
    y='CriticRating',
    palette='magma',
    ax=axes[1, 0]
)

# Plot [1,1] - Critic vs Audience rating
k4 = sns.kdeplot(
    x=movies.CriticRating,
    y=movies.AudienceRating,
    fill=True,
    cmap='Blues_r',
    thresh=0.05,
    ax=axes[1, 1]
)
k4b = sns.kdeplot(
    x=movies.CriticRating,
    y=movies.AudienceRating,
    cmap='gist_gray_r',
    ax=axes[1, 1]
)

# Final layout
plt.tight_layout()
plt.show()
```