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

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

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

```
In [3]: movies = pd.read_excel(r"C:\Users\Vansh\OneDrive\Documents\movies.xlsx")
        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]: 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 = ['Film','Genre','Critic Ratings','Audience Rating','Budget','Ye
```

In [9]:
```
movies.head()# Removed spaces & % removed nice characters
```

Out[9]:

	Film	Genre	Critic Ratings	Audience Rating	Budget	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 [10]:
```
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   Critic Ratings   559 non-null    int64 
 3   Audience Rating  559 non-null    int64 
 4   Budget           559 non-null    int64 
 5   Year             559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [11]:
```
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[11]:

	Critic Ratings	Audience Rating	Budget	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 [12]:
```
movies['Film']
# movies ['Audience Ratings %']
```

Out[12]:
```
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 [13]:
```
movies.Film
```

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

In [15]:
```
movies.Film
```

```
Out[15]: 0      (500) Days of Summer
         1                10,000 B.C.
         2                 12 Rounds
         3                  127 Hours
         4                  17 Again
                         ...
         554              Your Highness
         555            Youth in Revolt
         556                     Zodiac
         557                Zombieland
         558                  Zookeeper
         Name: Film, Length: 559, dtype: category
         Categories (559, object): [2012, '(500) Days of Summer ', '10,000 B.C.', '12 Ro
         unds ', ..., 'Youth in Revolt', 'Zodiac', 'Zombieland ', 'Zookeeper']
```

```
In [16]: movies.head()
```

Out[16]:

	Film	Genre	Critic Ratings	Audience Rating	Budget	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 [17]: movies.info()
         # now the samw 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   Critic Ratings   559 non-null    int64   
 3   Audience Rating  559 non-null    int64   
 4   Budget           559 non-null    int64   
 5   Year             559 non-null    int64   
dtypes: category(1), int64(4), object(1)
memory usage: 43.6+ KB
```

```
In [18]: movies.Genre = movies.Genre.astype('category')
```

```
In [19]: movies.Genre
```

Out[19]:
```
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 [20]:
```
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   Critic Ratings   559 non-null    int64
 3   Audience Rating  559 non-null    int64
 4   Budget           559 non-null    int64
 5   Year             559 non-null    int64
dtypes: category(2), int64(4)
memory usage: 40.1 KB
```

In [21]:
```
movies.Genre.cat.categories
```

Out[21]:
```
Index(['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance',
       'Thriller'],
      dtype='object')
```

In [22]:
```
movies.describe()
```

Out[22]:

	Critic Ratings	Audience Rating	Budget	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 [23]:
```
#now when you see the describe you will get only integer value mean, standard de
```

HOW TO WORK WITH JOINT PLOTS

In [25]:
```
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

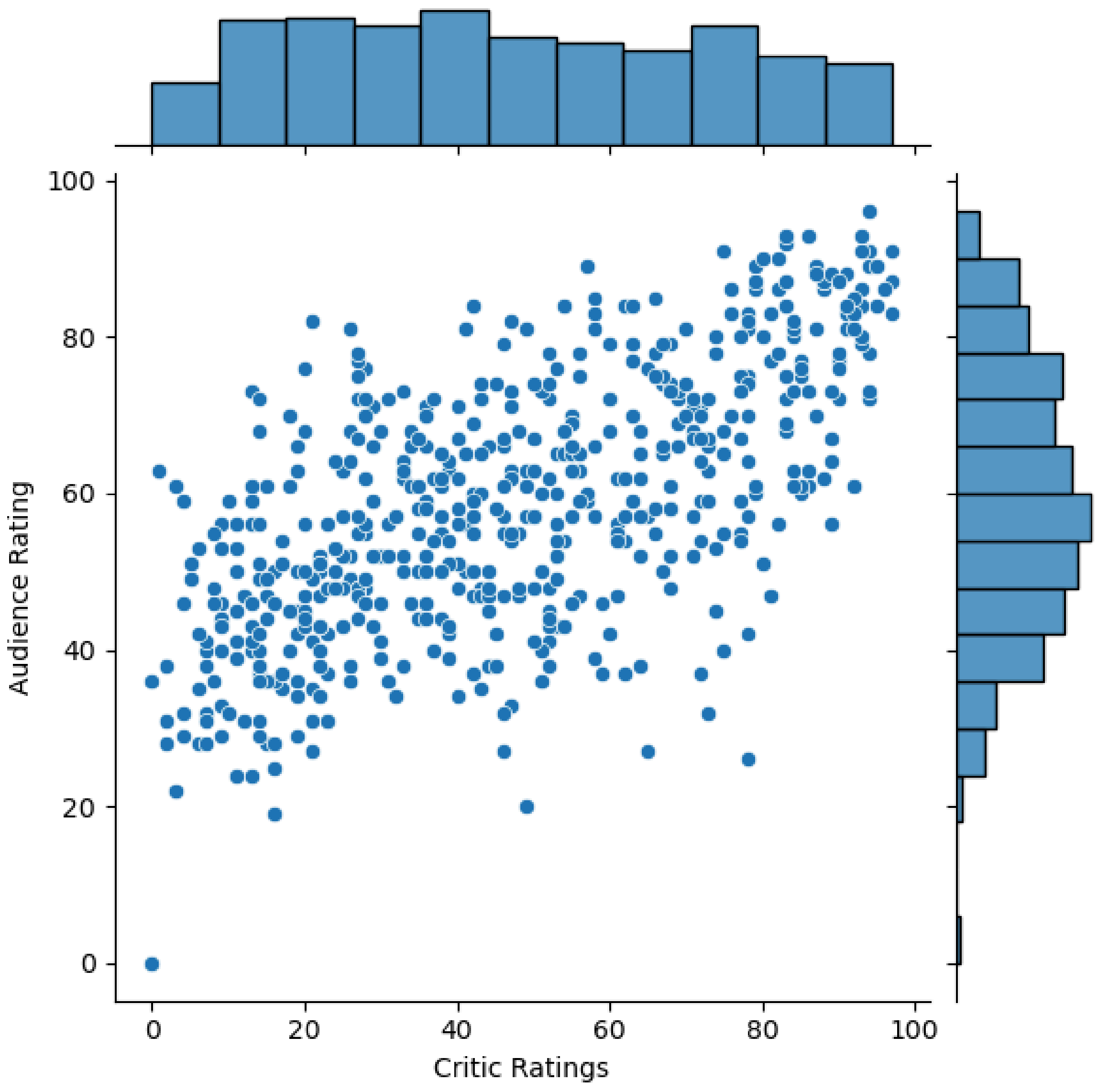
Basically Joint plot is a scatter plot & it find the relation b\w audience and critics also if you look up you can find the uniform disttribution(critics) and normal distribution(audience)

In [27]:
```
print(movies.columns)
```

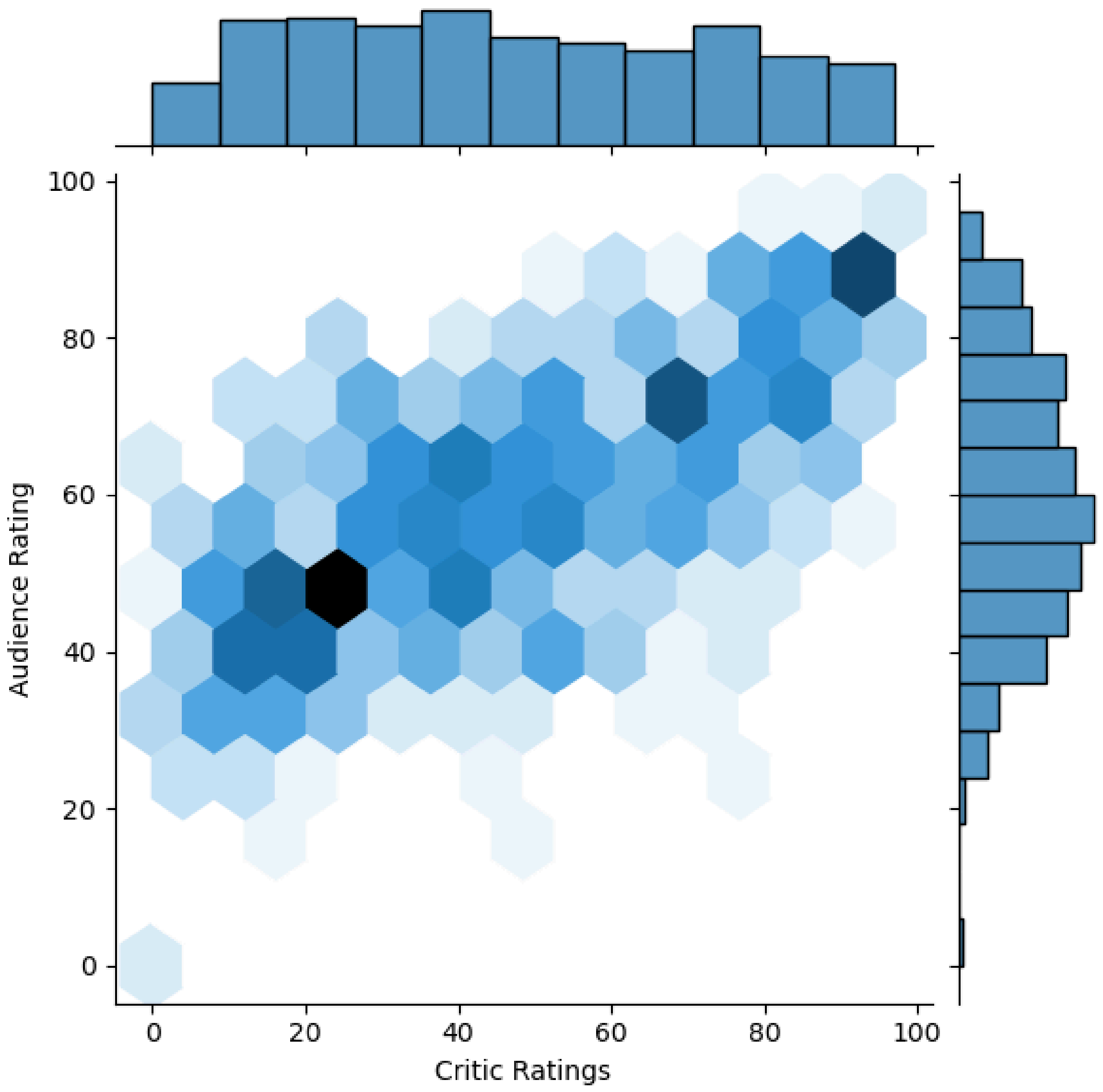
```
Index(['Film', 'Genre', 'Critic Ratings', 'Audience Rating', 'Budget', 'Year'], d
type='object')
```

In [28]:
```
import seaborn as sns
```

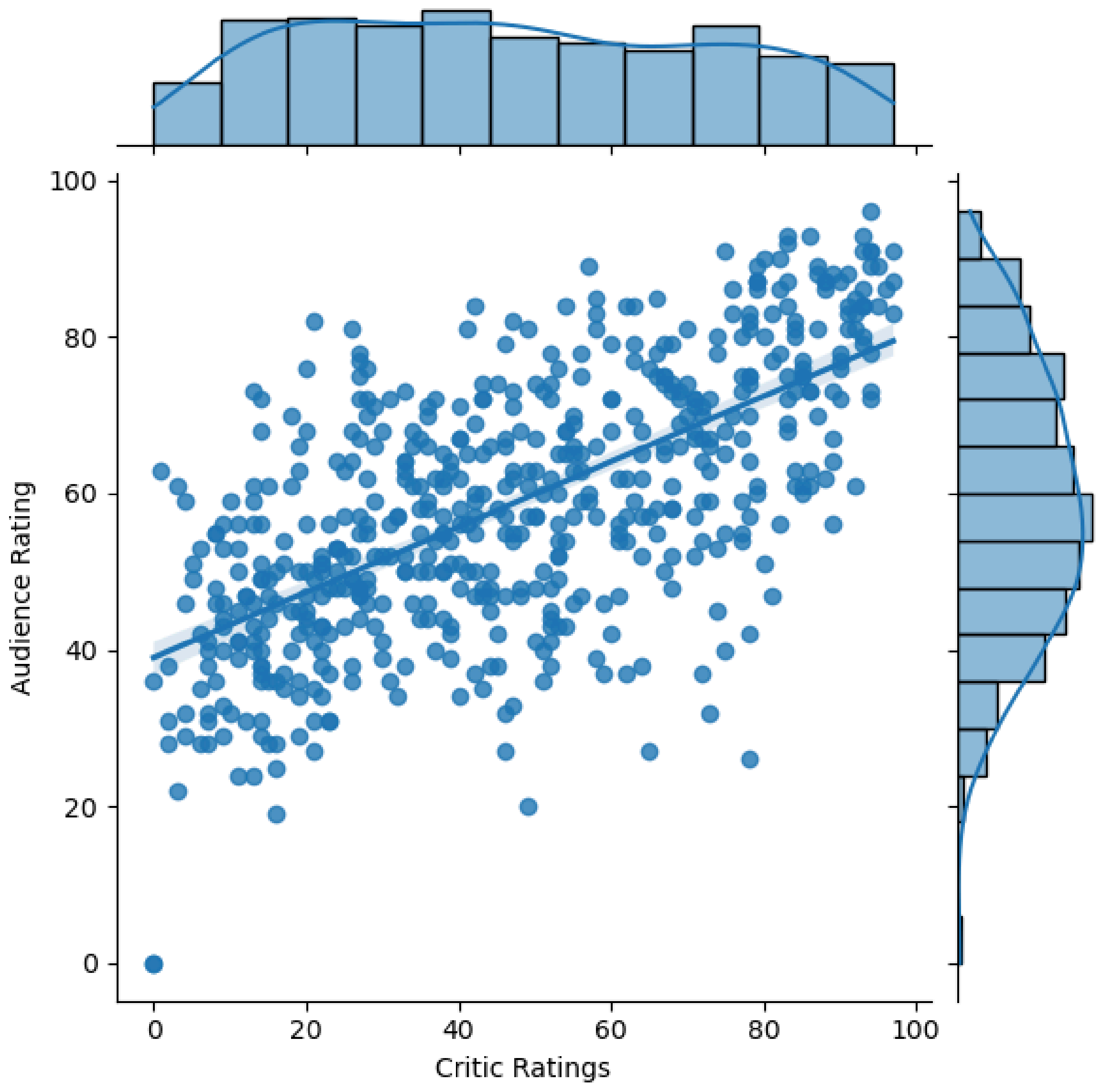
In [29]:
```
sns.jointplot (data = movies, x = 'Critic Ratings', y = 'Audience Rating')
plt.show()
```



```
In [30]: j=sns.jointplot( data = movies , x = 'Critic Ratings', y = 'Audience Rating',kin
         j=sns.jointplot( data = movies , x = 'Critic Ratings', y = 'Audience Rating',kin
         plt.show()
```



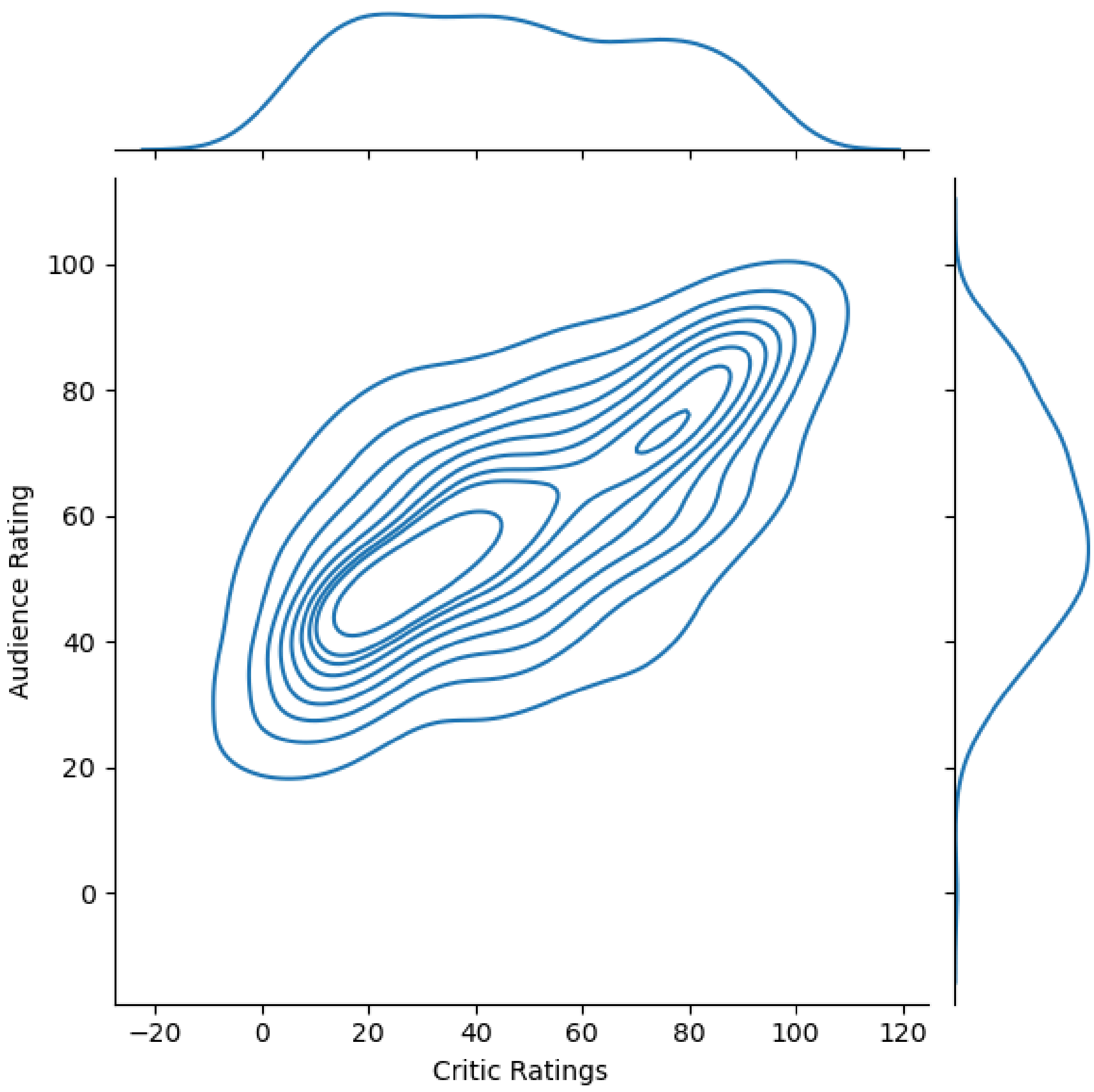
100
80
60
40
20
0
Audience Rating
0
20
40
60
80
100
Critic Ratings



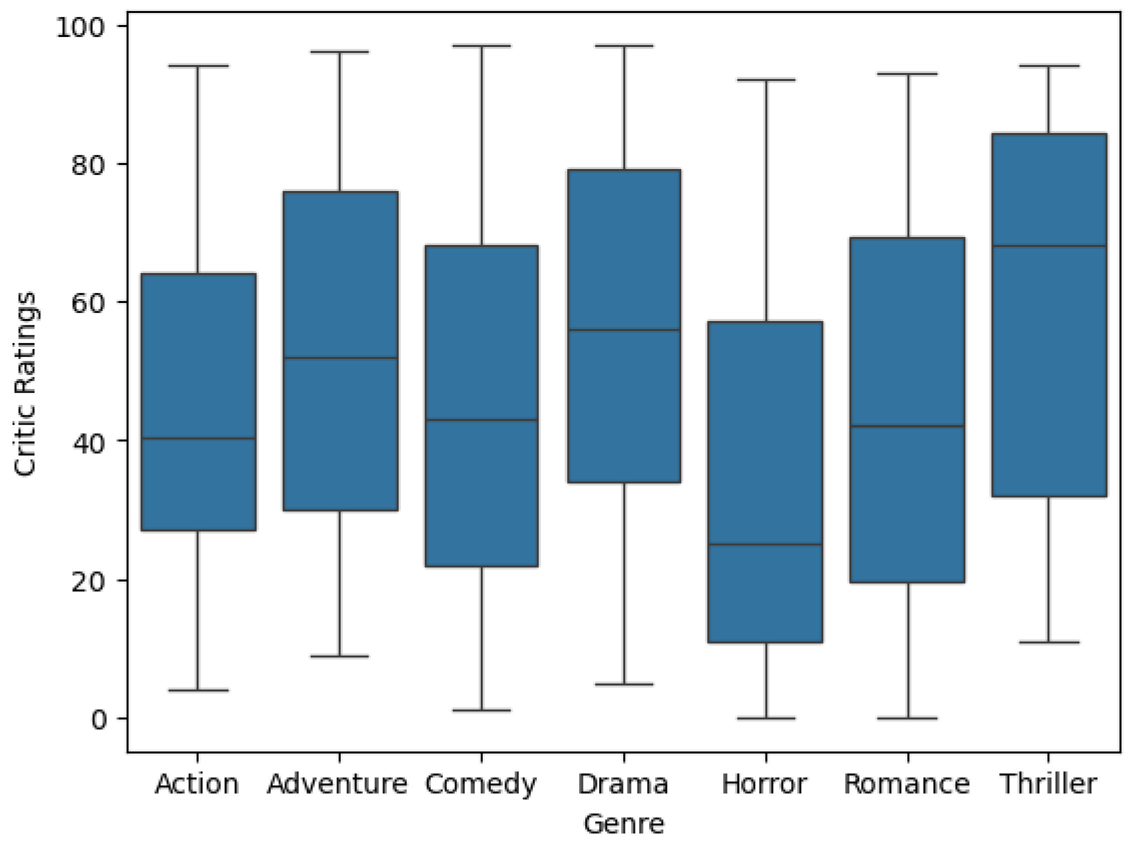
```
In [31]: print(movies.columns)
```

```
Index(['Film', 'Genre', 'Critic Ratings', 'Audience Rating', 'Budget', 'Year'], d
type='object')
```

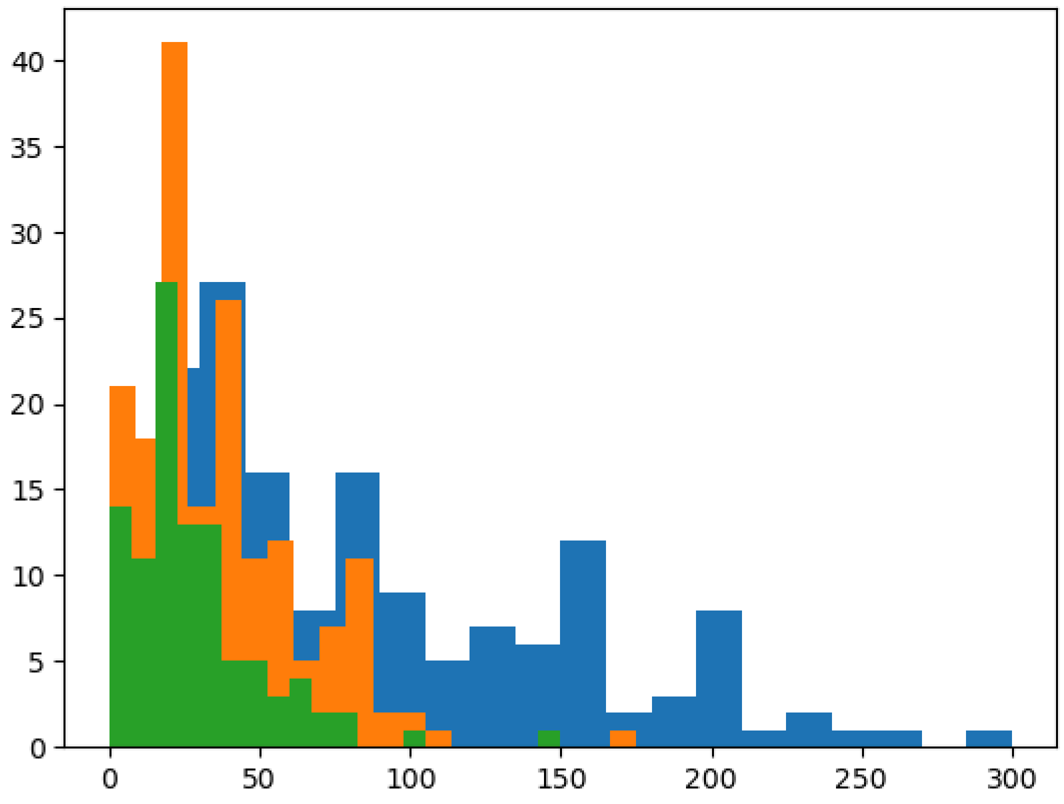
```
In [32]: j=sns.jointplot( data = movies , x = 'Critic Ratings', y = 'Audience Rating',kin
         plt.show()
```



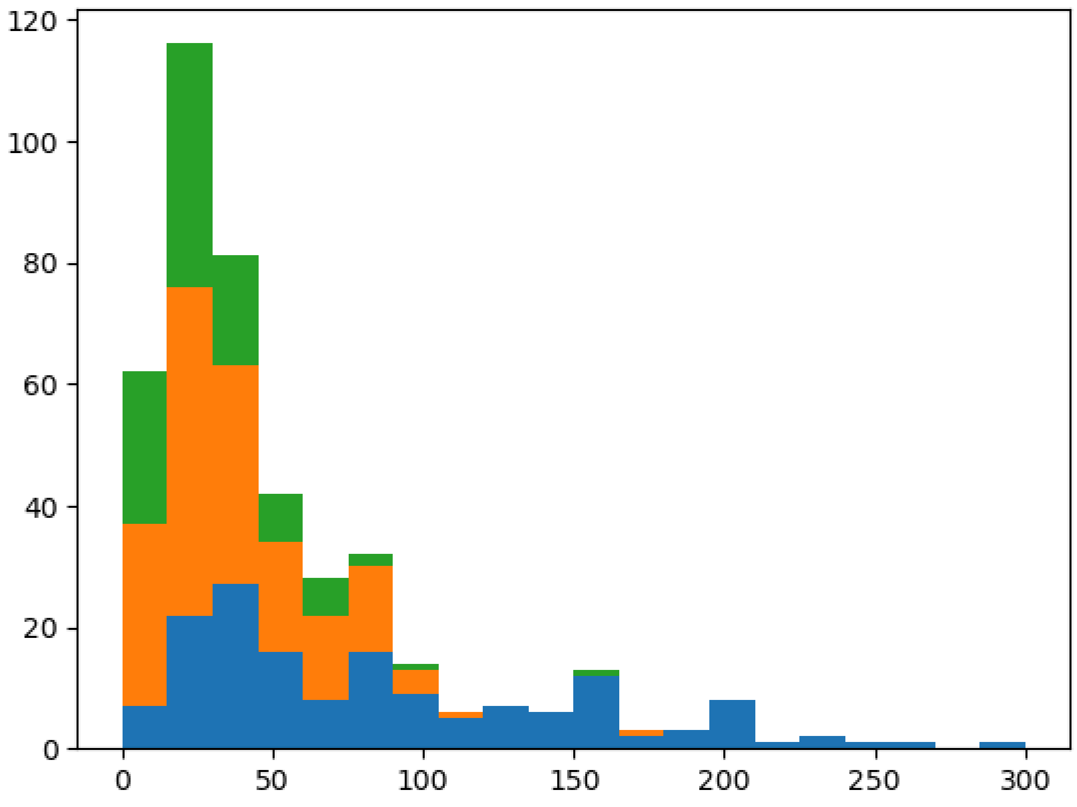
```
In [33]: j=sns.boxplot( data = movies , x = 'Genre', y = 'Critic Ratings')
         plt.show()
```



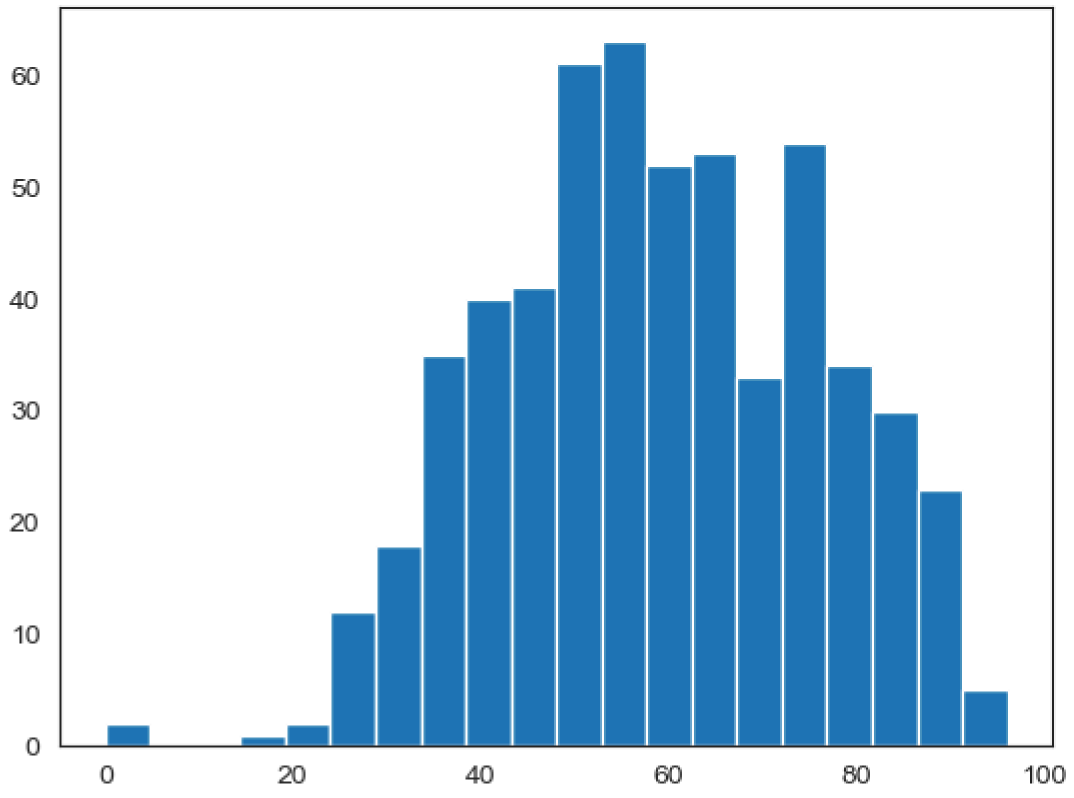
In [34]:
```
#Below plots are stacked histogram beacuse overlapped
#Filters Budget for Action, comedy, drama

plt.hist(movies[movies.Genre == 'Action'].Budget, bins=20)
plt.hist(movies[movies.Genre == 'Comedy'].Budget, bins=20)
plt.hist(movies[movies.Genre == 'Drama'].Budget, bins=20)
plt.show()
```



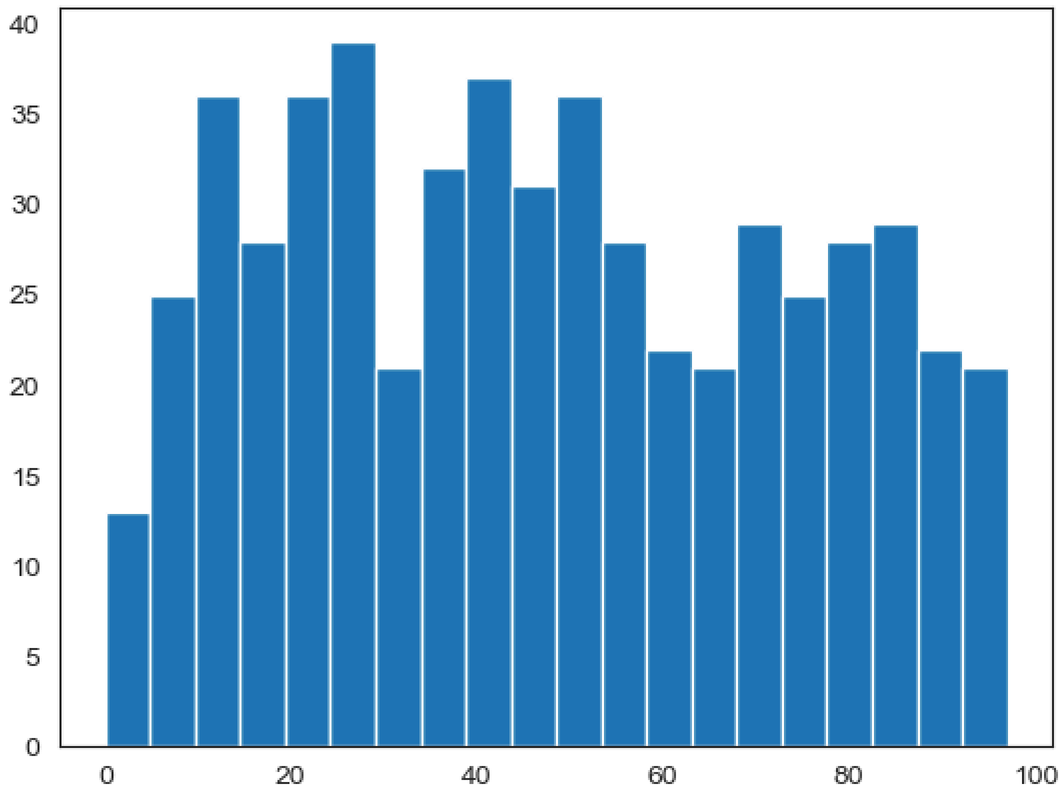
```
In [35]: plt.hist([movies[movies.Genre == 'Action'].Budget,
                   movies[movies.Genre == 'Comedy'].Budget,
                   movies[movies.Genre == 'Drama'].Budget],
                   bins=20, stacked=True)
         plt.show()
```



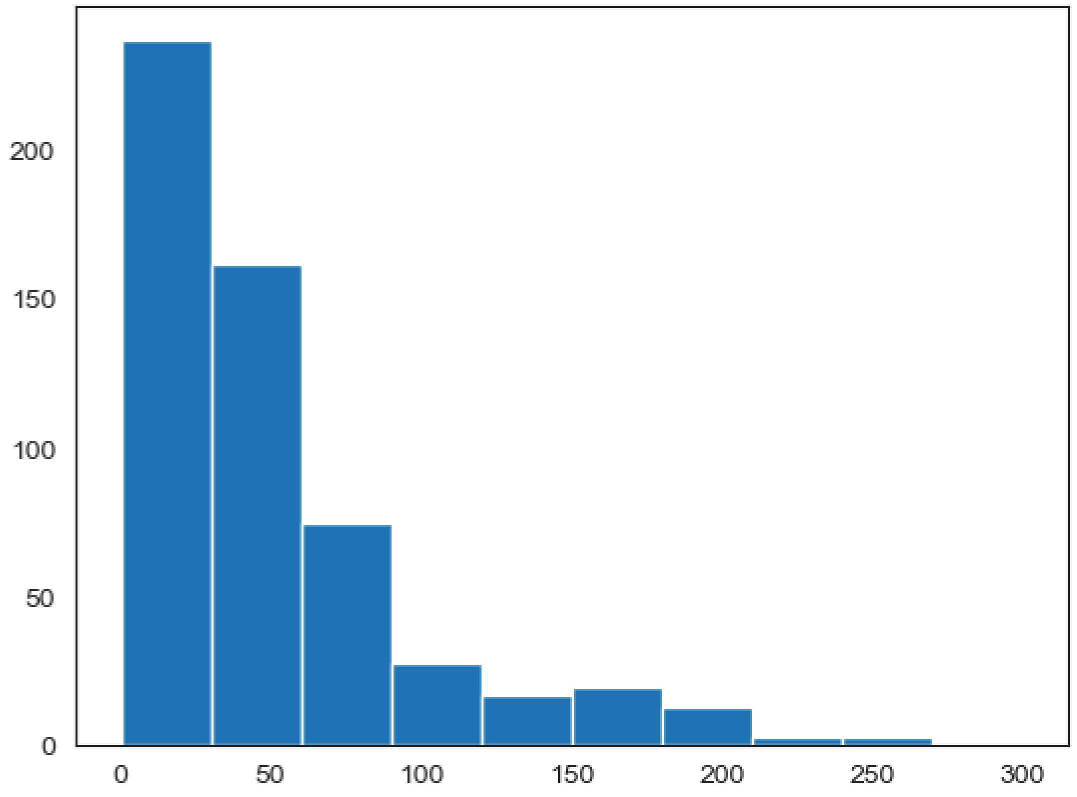
```
In [36]: sns.set_style('white') #normal distribution & called as bell curve
         n1 = plt.hist(movies['Audience Rating'], bins=20)
         plt.show()
```



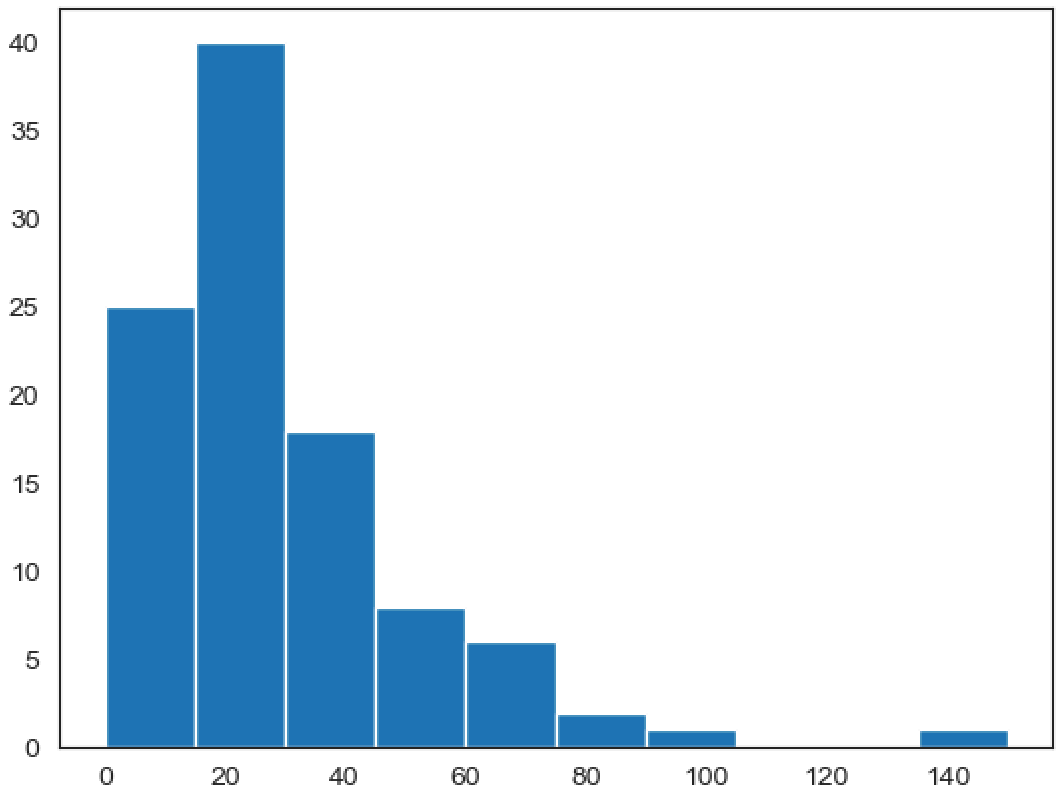
In [37]:
```python
n=plt.hist(movies["Critic Ratings"], bins=20) #uniform distribution
plt.show()
```



In [38]:
```python
plt.hist(movies.Budget)
plt.show()
```

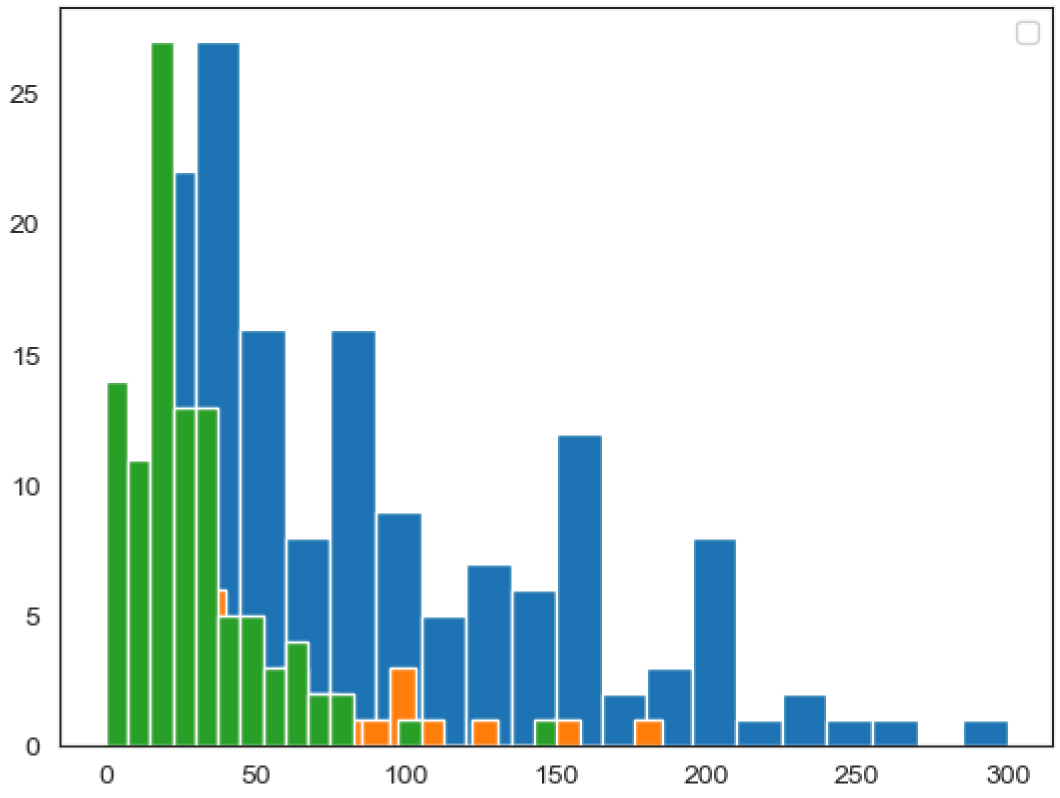


In [39]:
```python
plt.hist(movies[movies.Genre == 'Drama'].Budget)
plt.show()
```

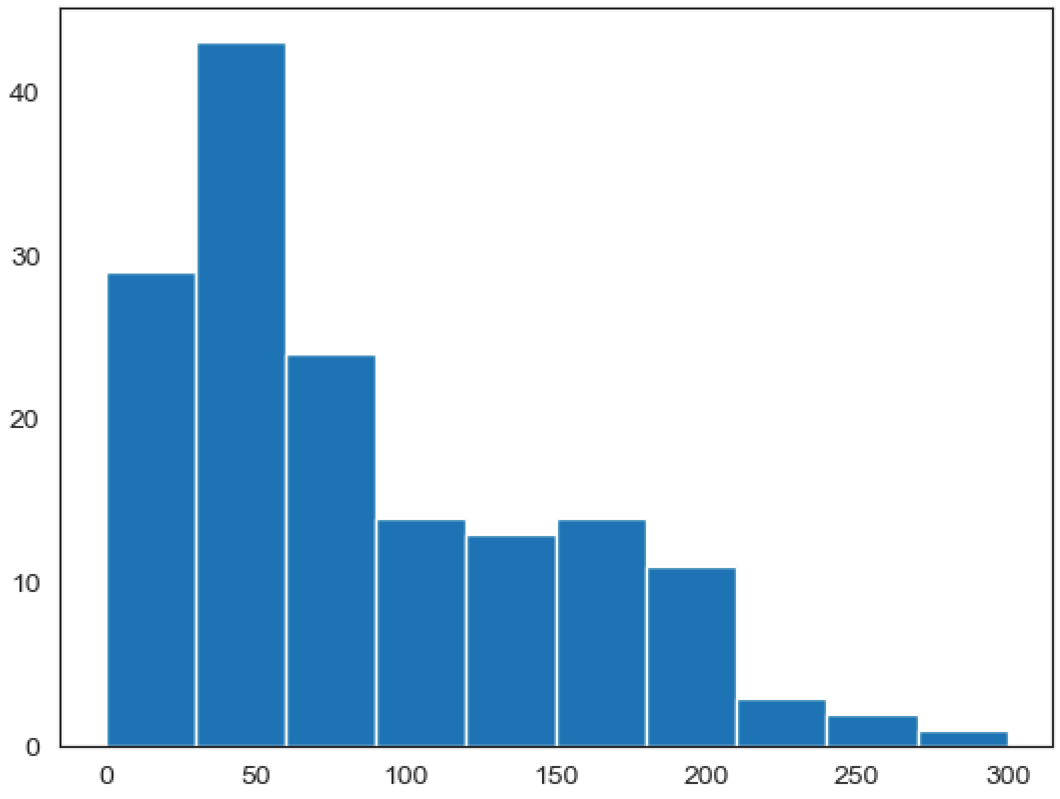


In [40]:
```
#movies.Genre.unique()
```

In [41]:
```
# Below plots are stacked histogram becuase overlaped

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



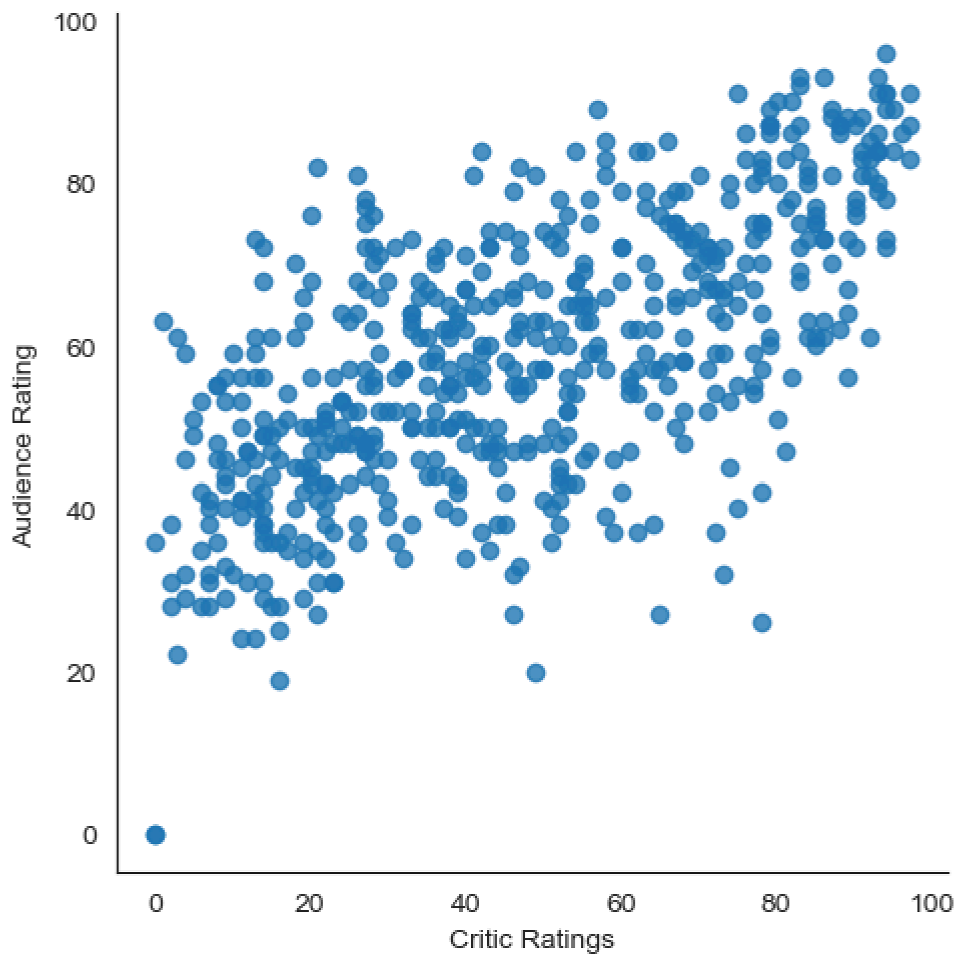
```
In [42]: plt.hist([movies[movies.Genre == 'Action'].Budget])
         plt.show()
```



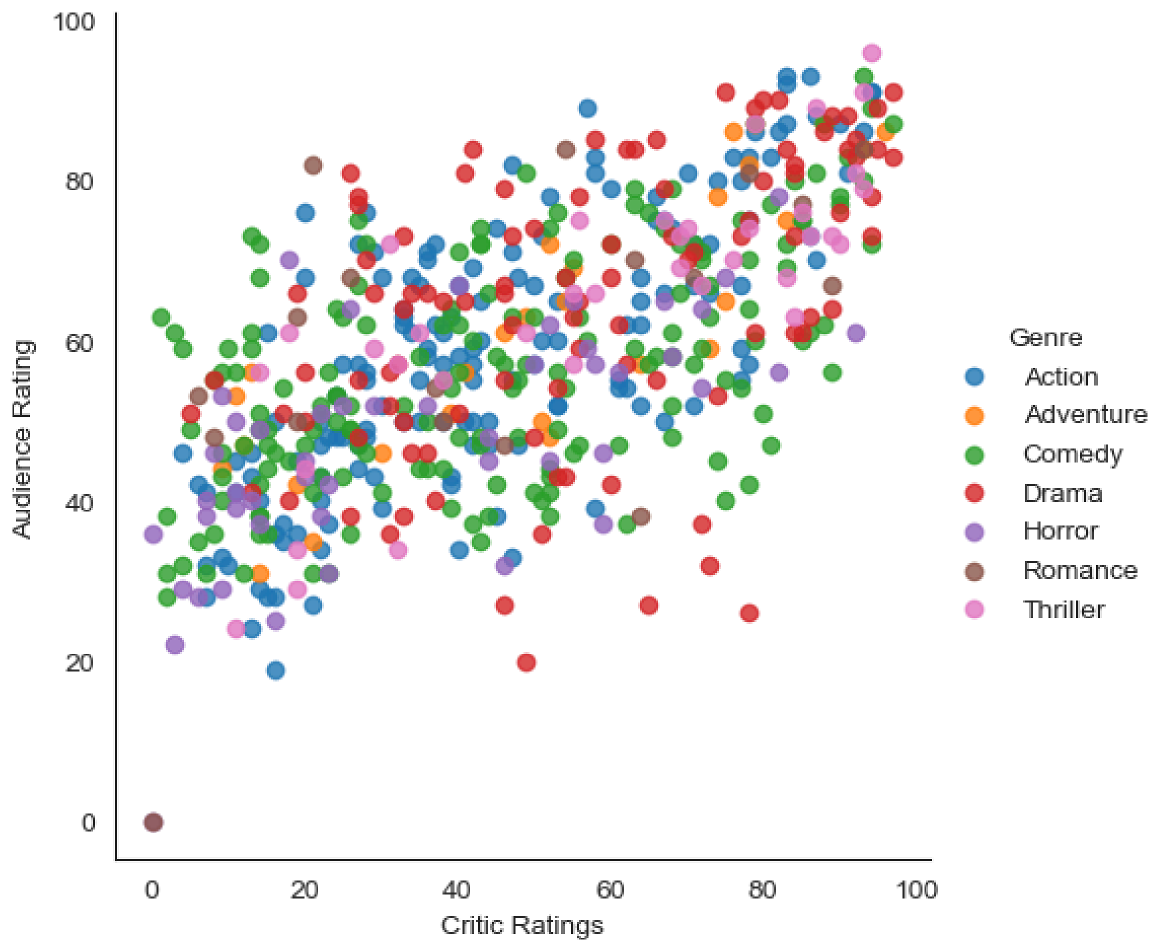
```
In [43]: # if you have 100 categories you cannot copy & paste all the things

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

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

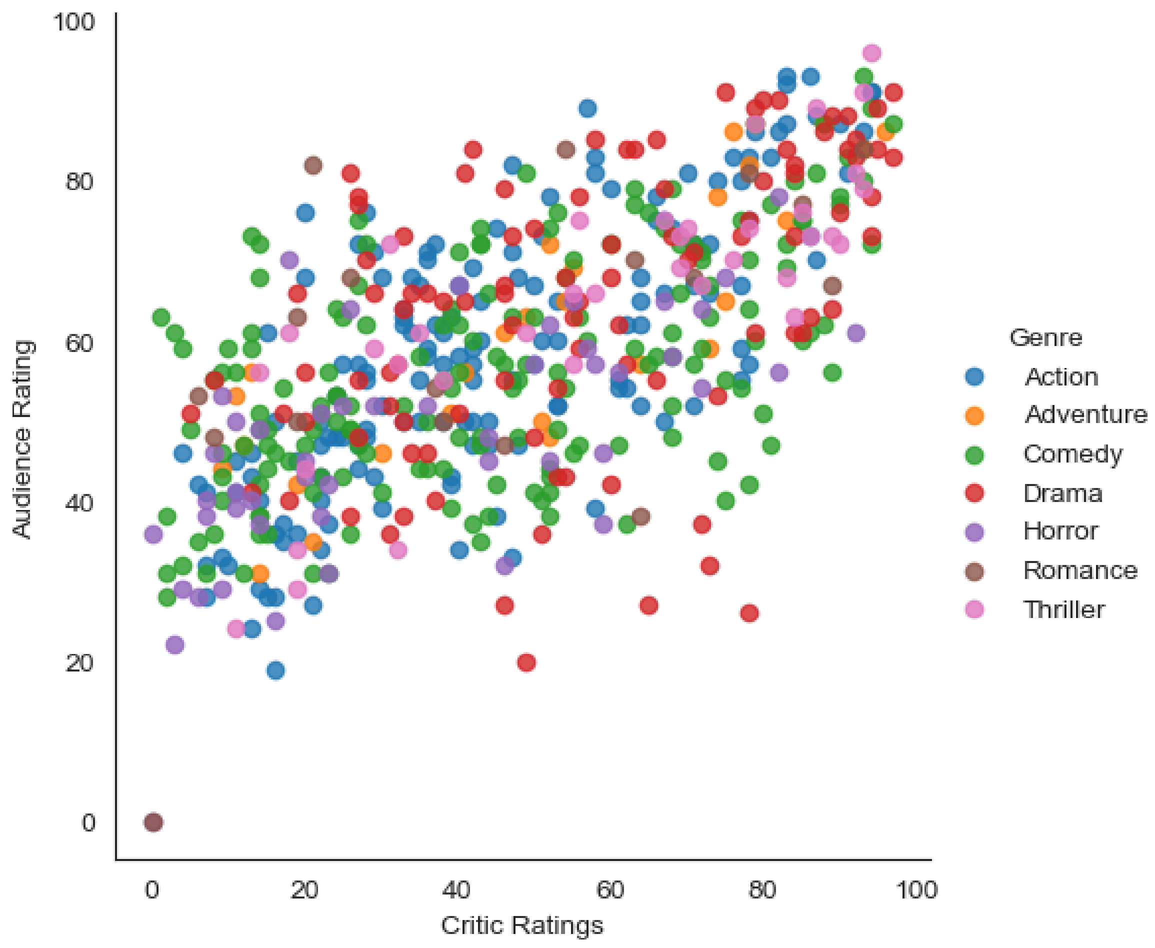
```
In [44]: vis1 = sns.lmplot(data=movies, x='Critic Ratings', y='Audience Rating',fit_reg=F
         plt.show()
```



```
In [45]:  vis1 = sns.lmplot(data=movies, x='Critic Ratings', y='Audience Rating',fit_reg=F
           plt.show()
```

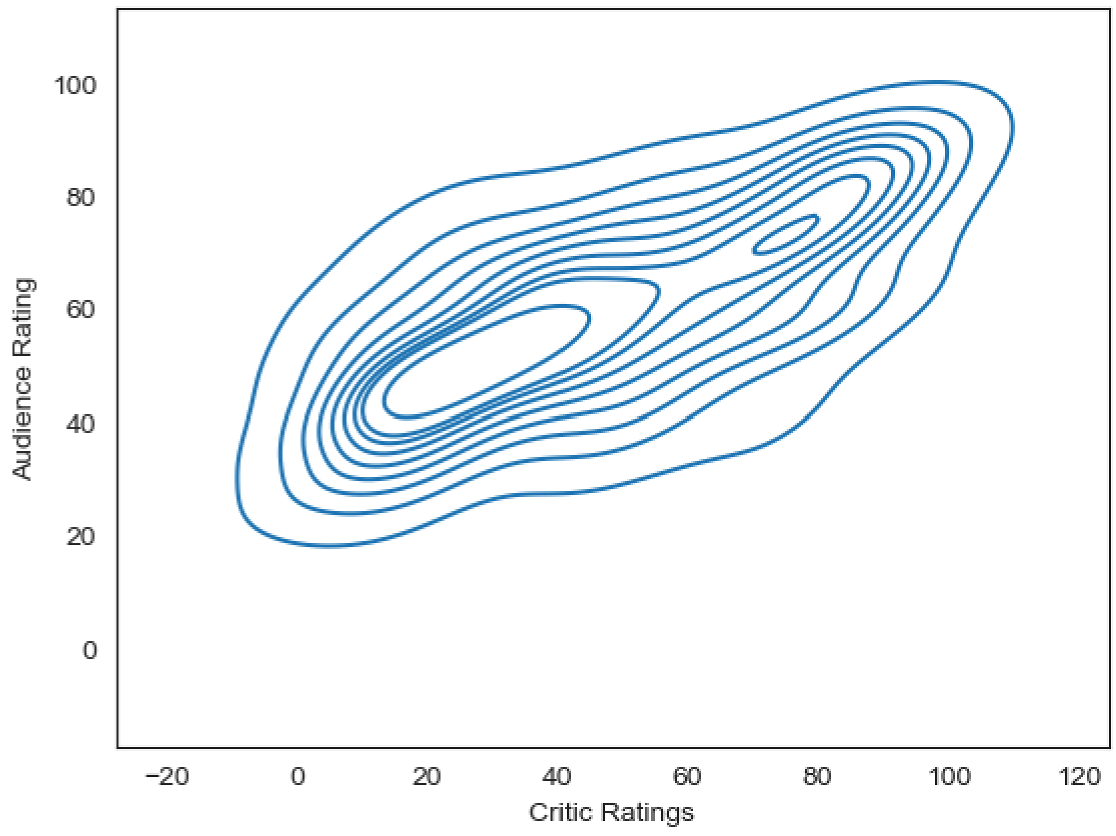


```
In [46]: vis1 = sns.lmplot(data=movies, x='Critic Ratings', y='Audience Rating', fit_reg=
         plt.show()
```

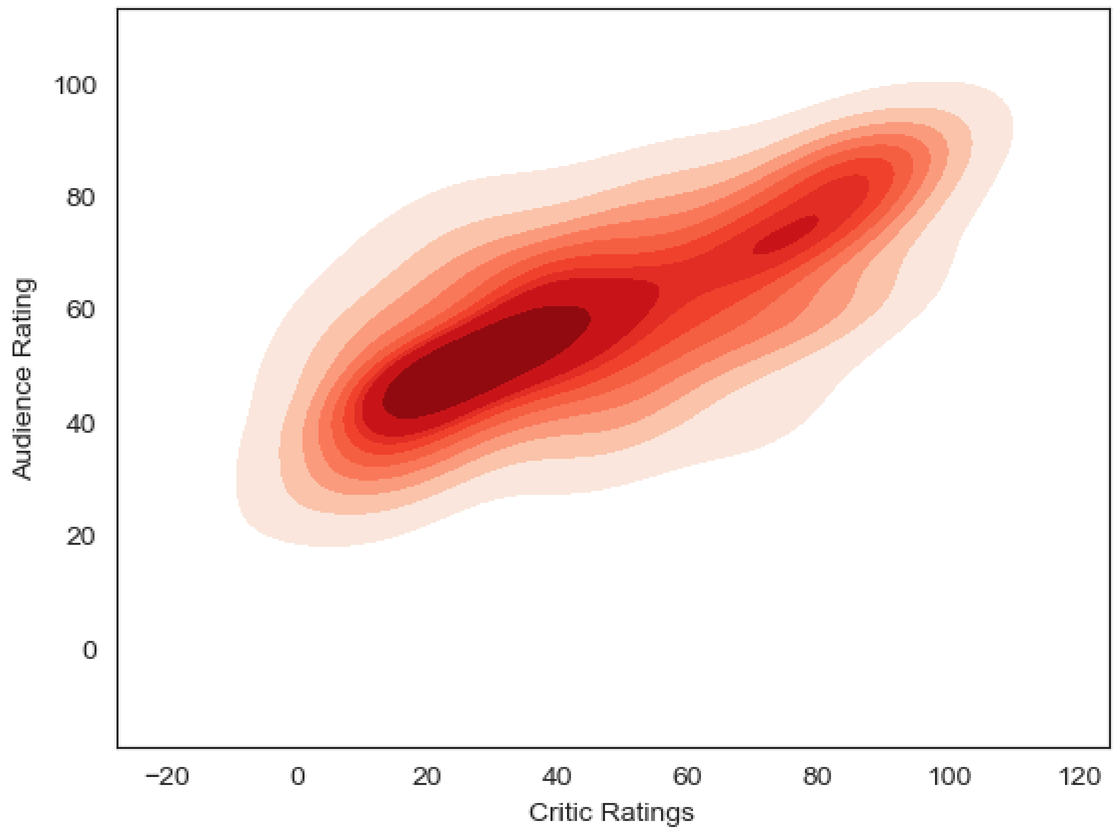


In [47]:
```
# Kernal Density Estimate plot ( KDE PLOT)
# how can i visulize audience rating & critics rating . using scatterplot
```

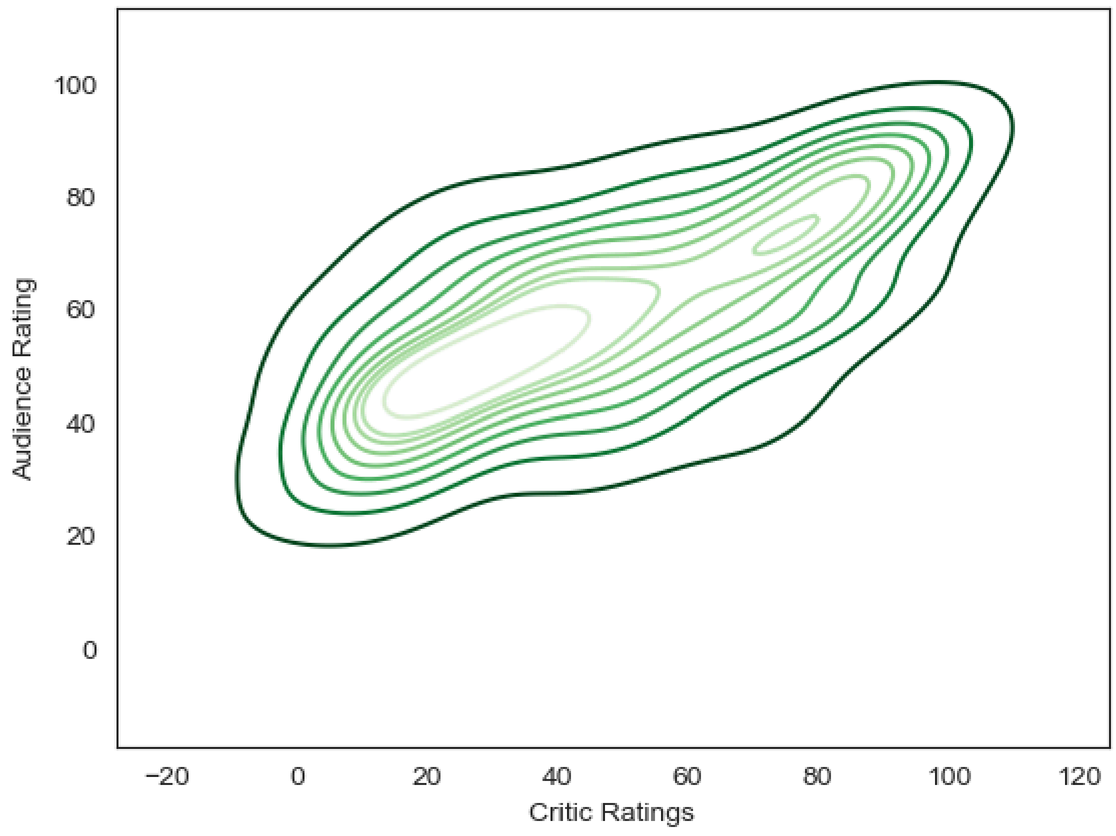
In [48]:
```
sns.kdeplot(x=movies['Critic Ratings'], y=movies['Audience Rating'])
plt.show()
# where do u find more density and how density is distibuted across from the the
# center point is kernal this is calld KDE & insteade of dots it visualize like
# we can able to clearly see the spread at the audience ratings
```



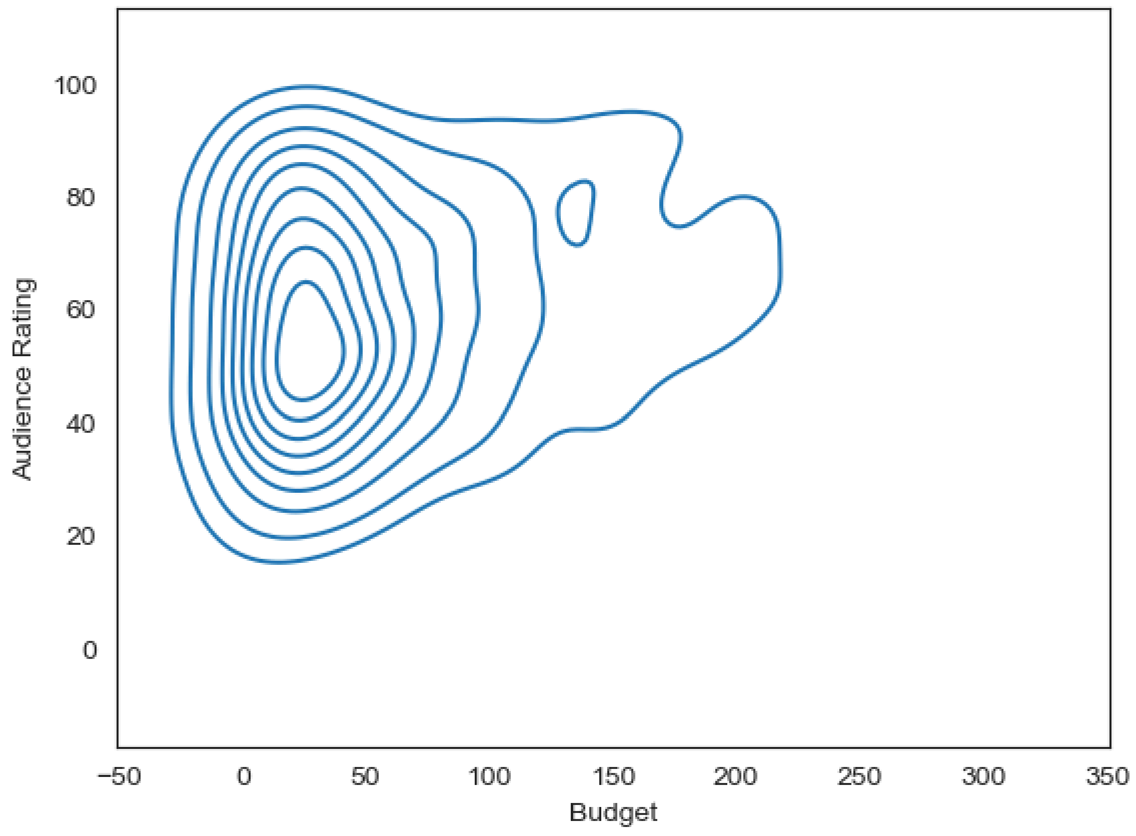
In [49]:
```
# Assuming 'movies' is your DataFrame with columns 'CriticRating' and 'AudienceR
k1 = sns.kdeplot(x=movies['Critic Ratings'], y=movies['Audience Rating'], shade=
plt.show()
```



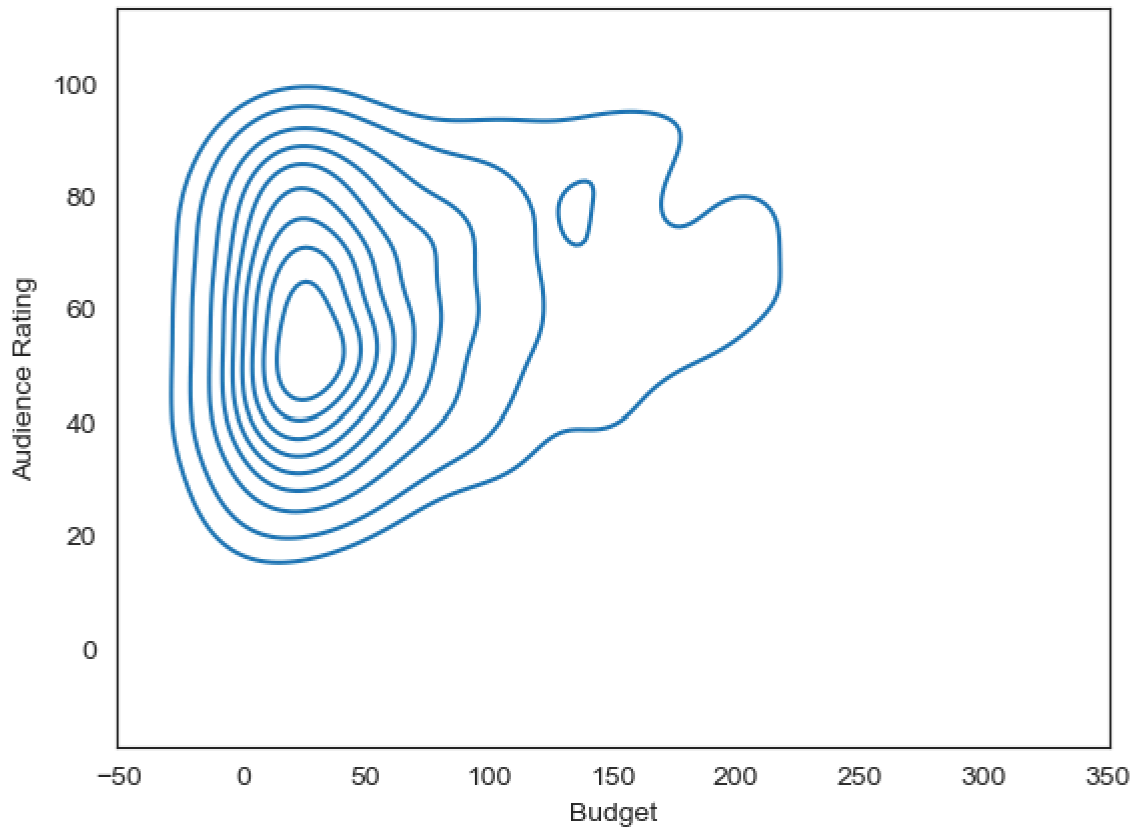
In [50]:
```
k2 = sns.kdeplot(x=movies['Critic Ratings'], y=movies['Audience Rating'], shade_
plt.show()
```



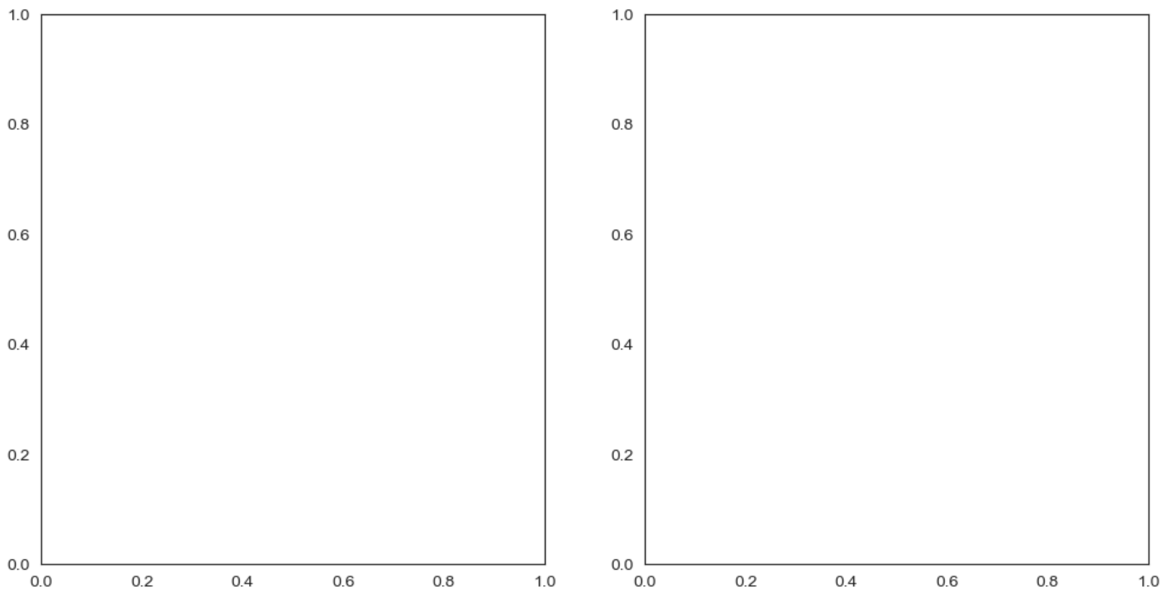
```
In [51]: #sns.set_style('dark')
         k1 = sns.kdeplot(x= movies['Budget'], y= movies["Audience Rating"])
         plt.show()
```



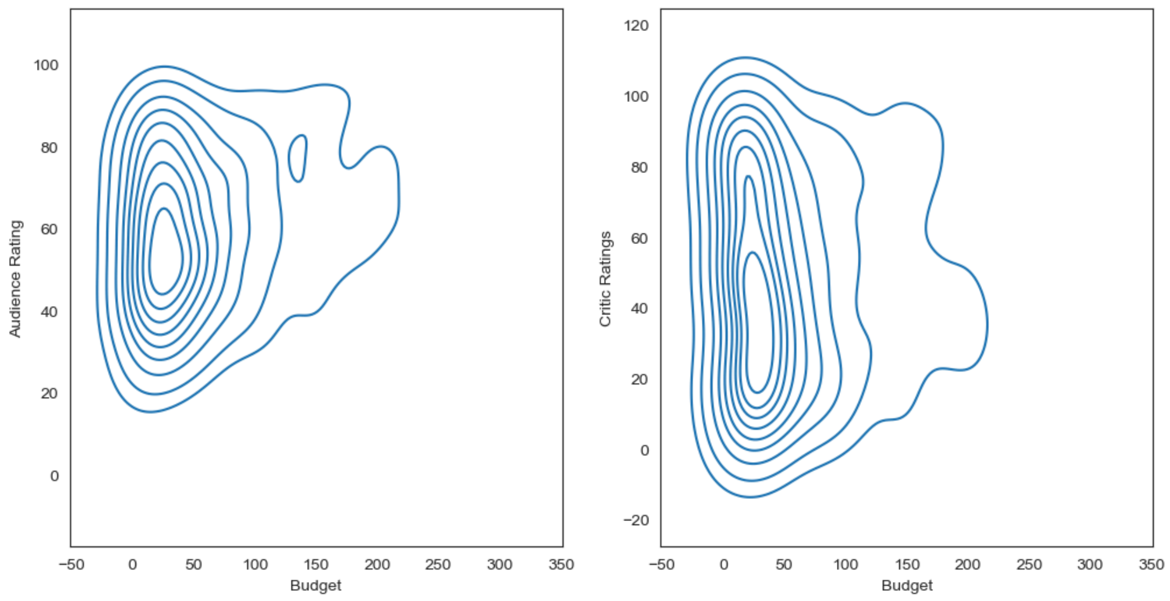
```
In [52]: k2 = sns.kdeplot(x= movies['Budget'], y=movies['Audience Rating'])
         plt.show()
```



In [53]:
```
#subplots

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



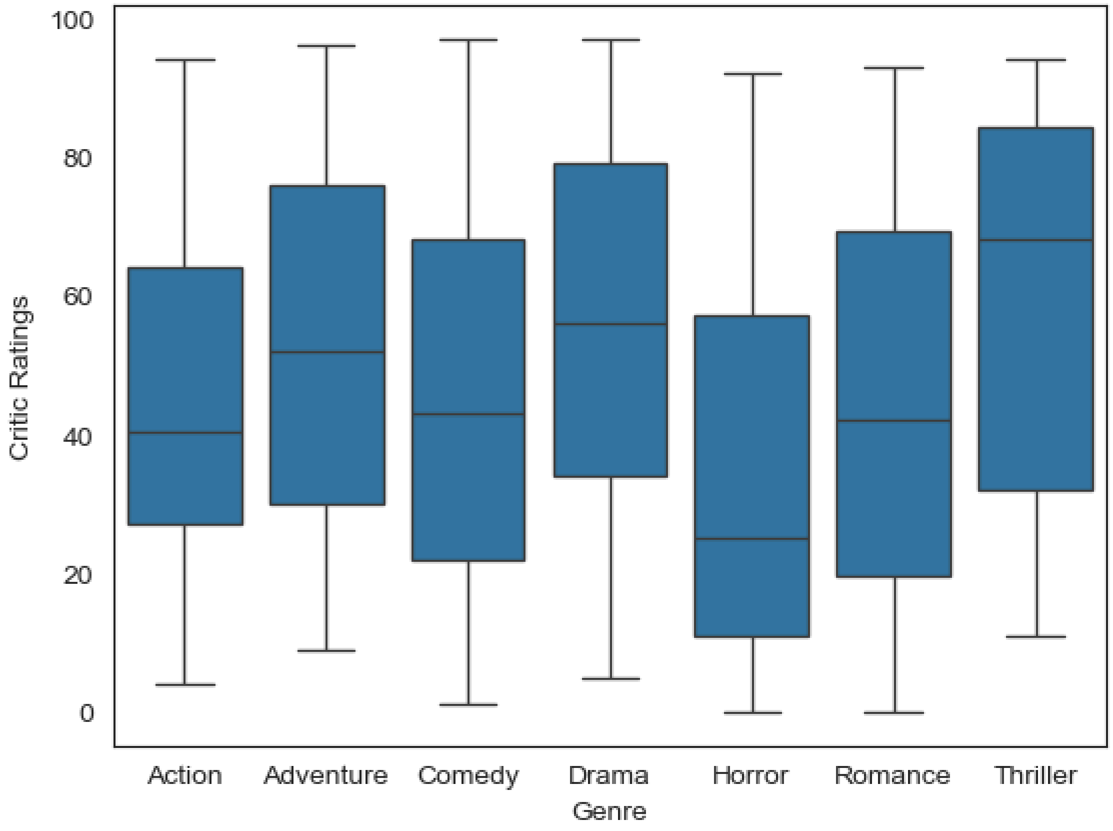
In [54]:
```
f , axes = plt.subplots(1,2, figsize =(12,6))

k1 = sns.kdeplot(movies,x = movies['Budget'],y=movies['Audience Rating'],ax=axes
k2 = sns.kdeplot(movies,x = movies['Budget'],y=movies['Critic Ratings'],ax = axe
plt.show()
```



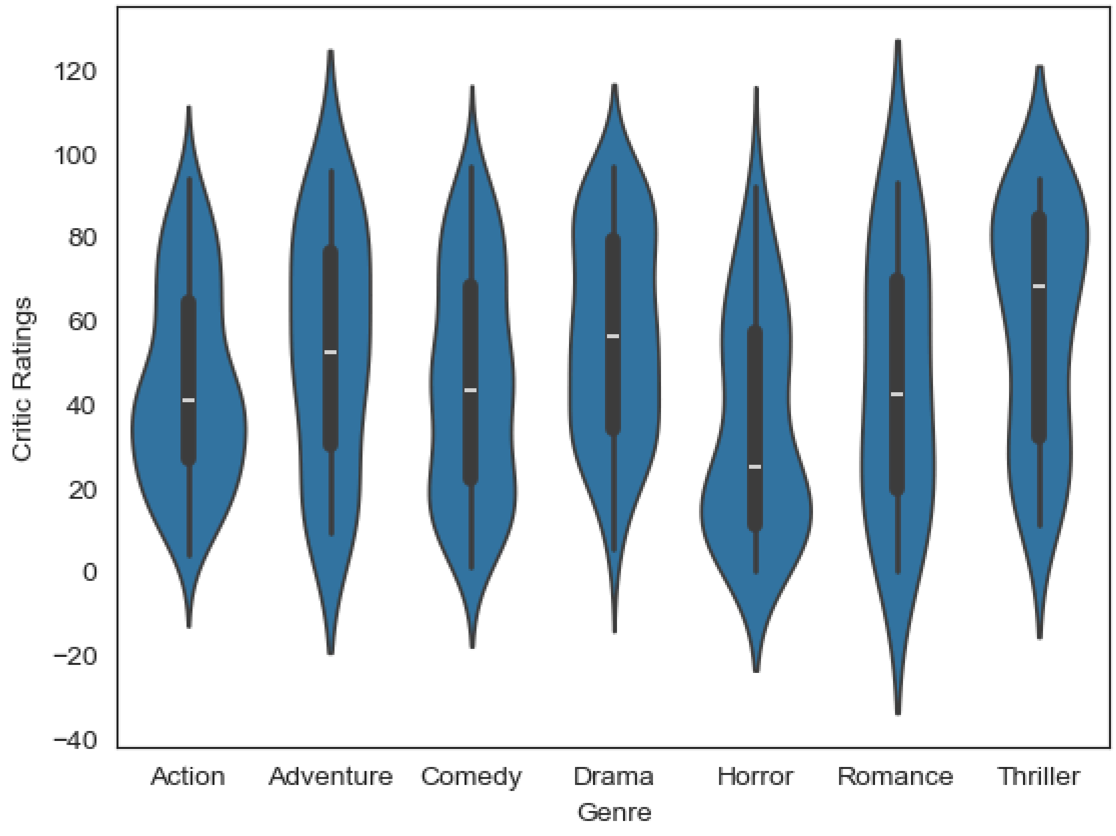
```
In [55]: axes
```

```
Out[55]: array([<Axes: xlabel='Budget', ylabel='Audience Rating'>,
                <Axes: xlabel='Budget', ylabel='Critic Ratings'>], dtype=object)
```

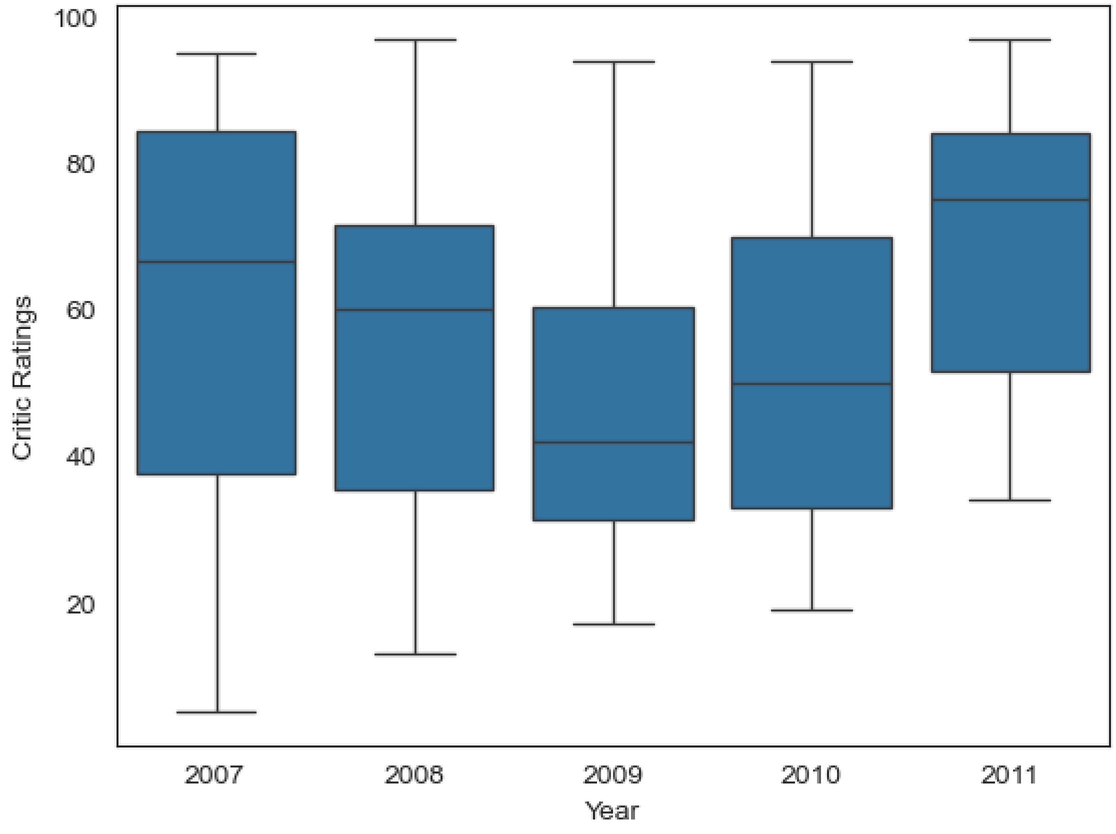
```
In [56]: #Boxplots
         w = sns.boxplot(data=movies, x='Genre', y = 'Critic Ratings')
         plt.show()
```



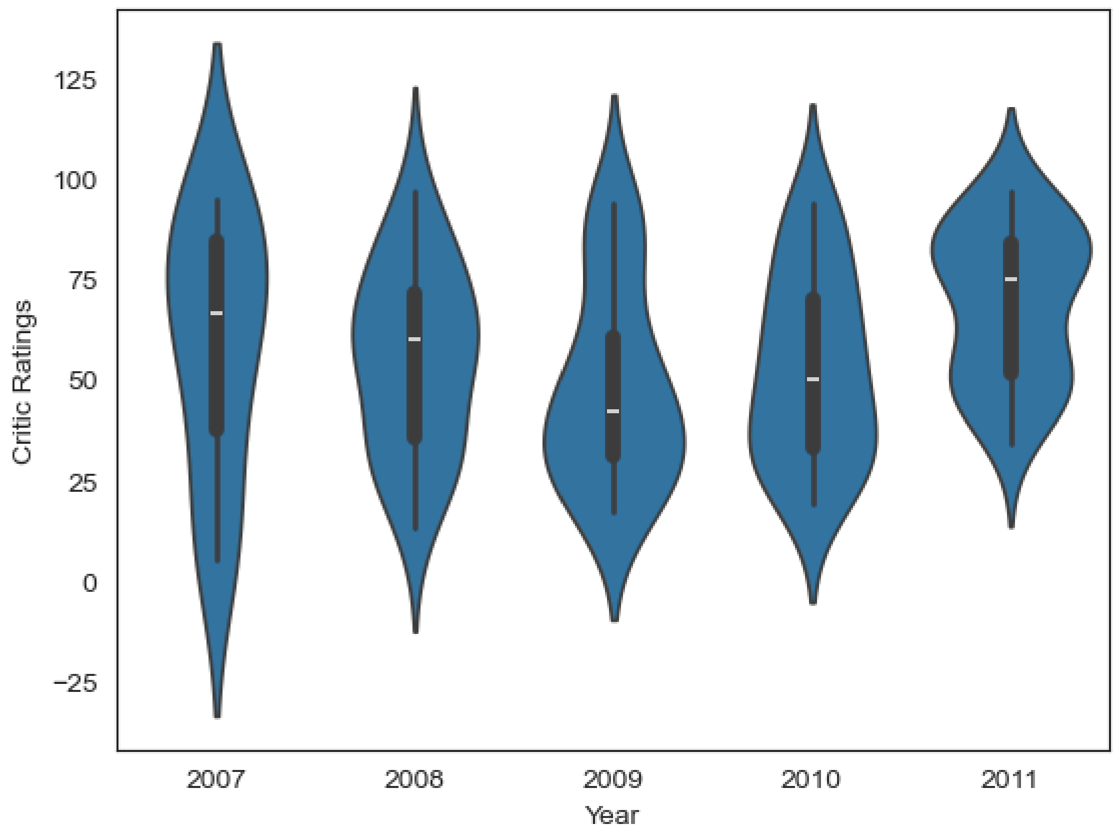
```
In [57]: # Violin plot
         z = sns.violinplot(data=movies, x='Genre', y = 'Critic Ratings')
         plt.show()
```



```
In [58]: w1 = sns.boxplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'Critic Rat
         plt.show()
```



```
In [59]: z = sns.violinplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'Critic R
         plt.show()
```

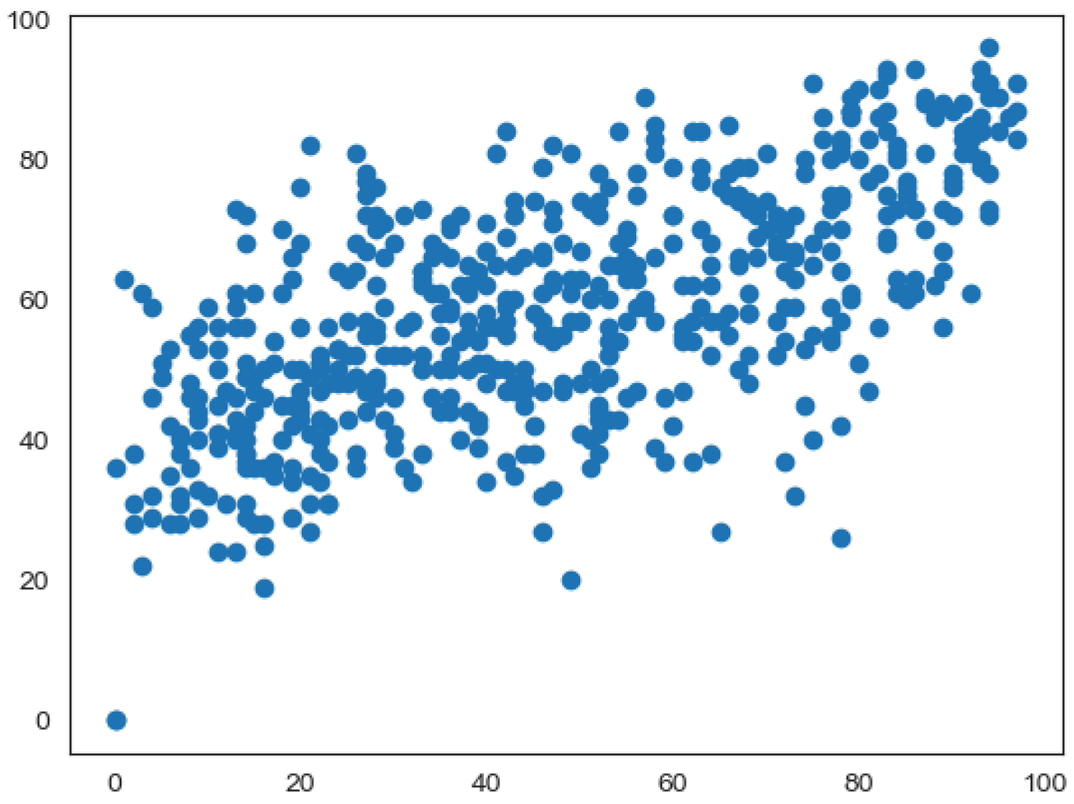


```
In [60]: # creating a facet grid
```

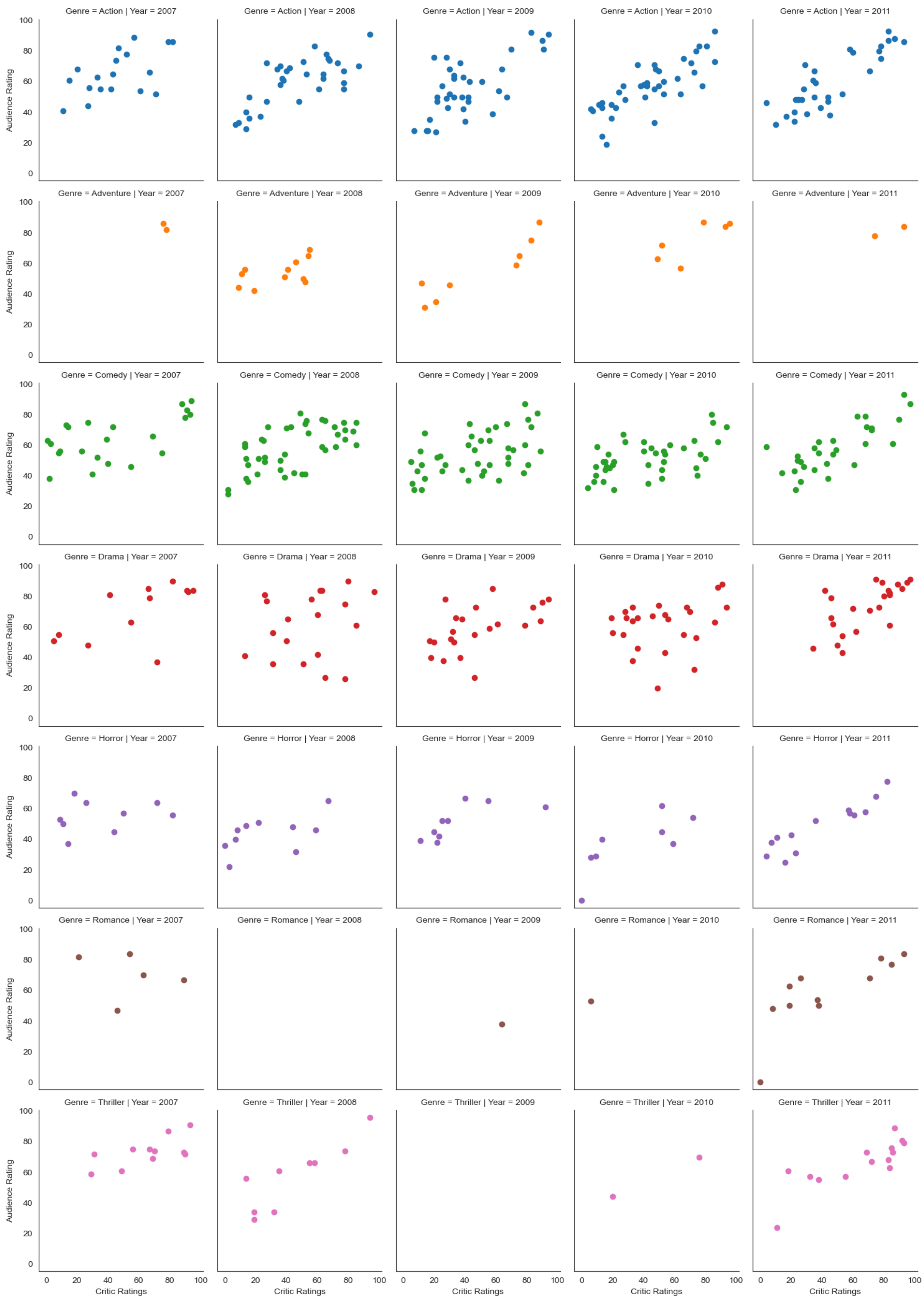
```
In [61]: g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre') #kind of s
         plt.show()
```

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
0.0
0.2
0.4
0.6
0.8
1.0

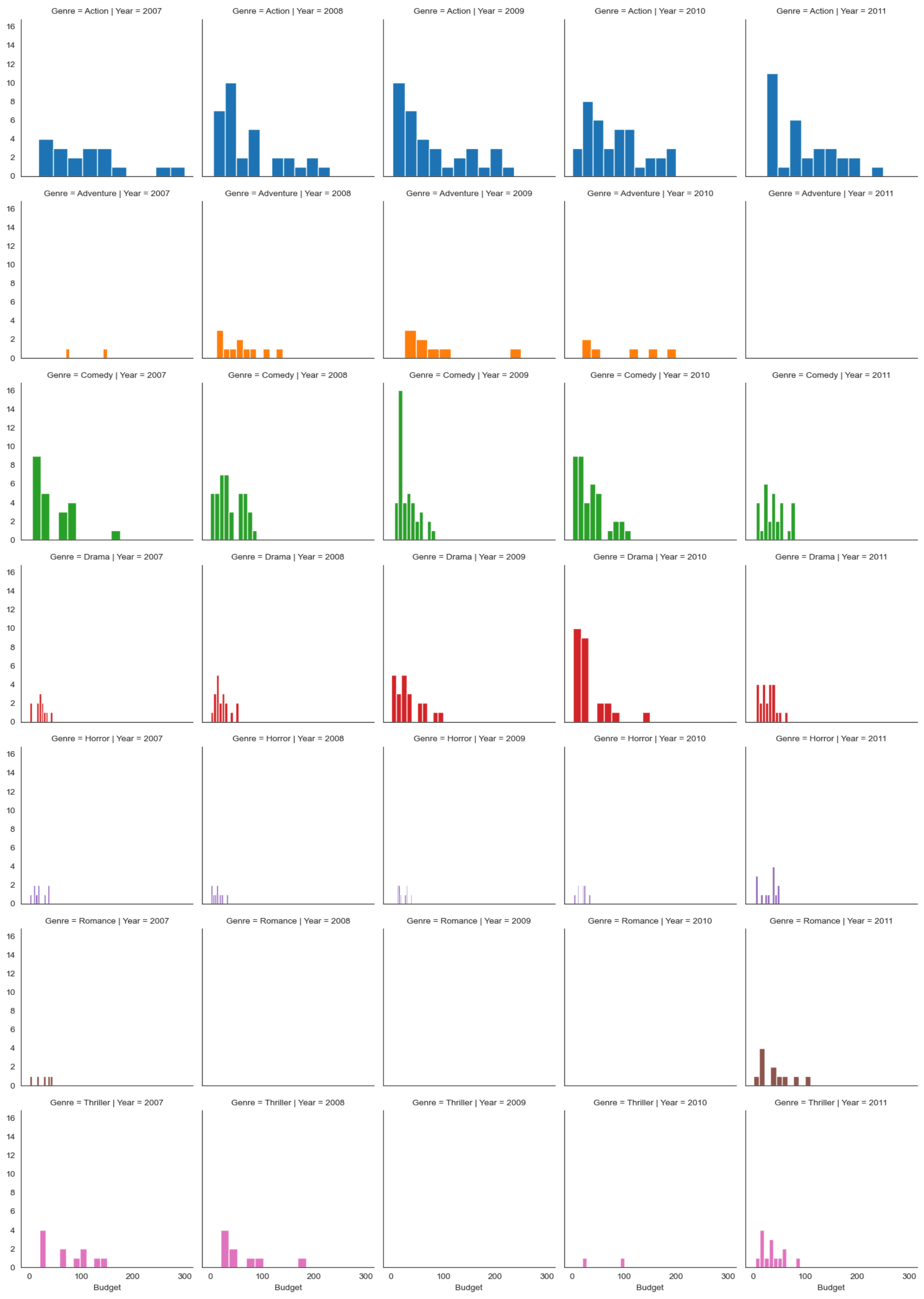
```
In [62]: plt.scatter(movies['Critic Ratings'],movies['Audience Rating'])
         plt.show()
```



In [63]:
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.scatter, 'Critic Ratings', 'Audience Rating' ) #scatterplots are m
plt.show()
```

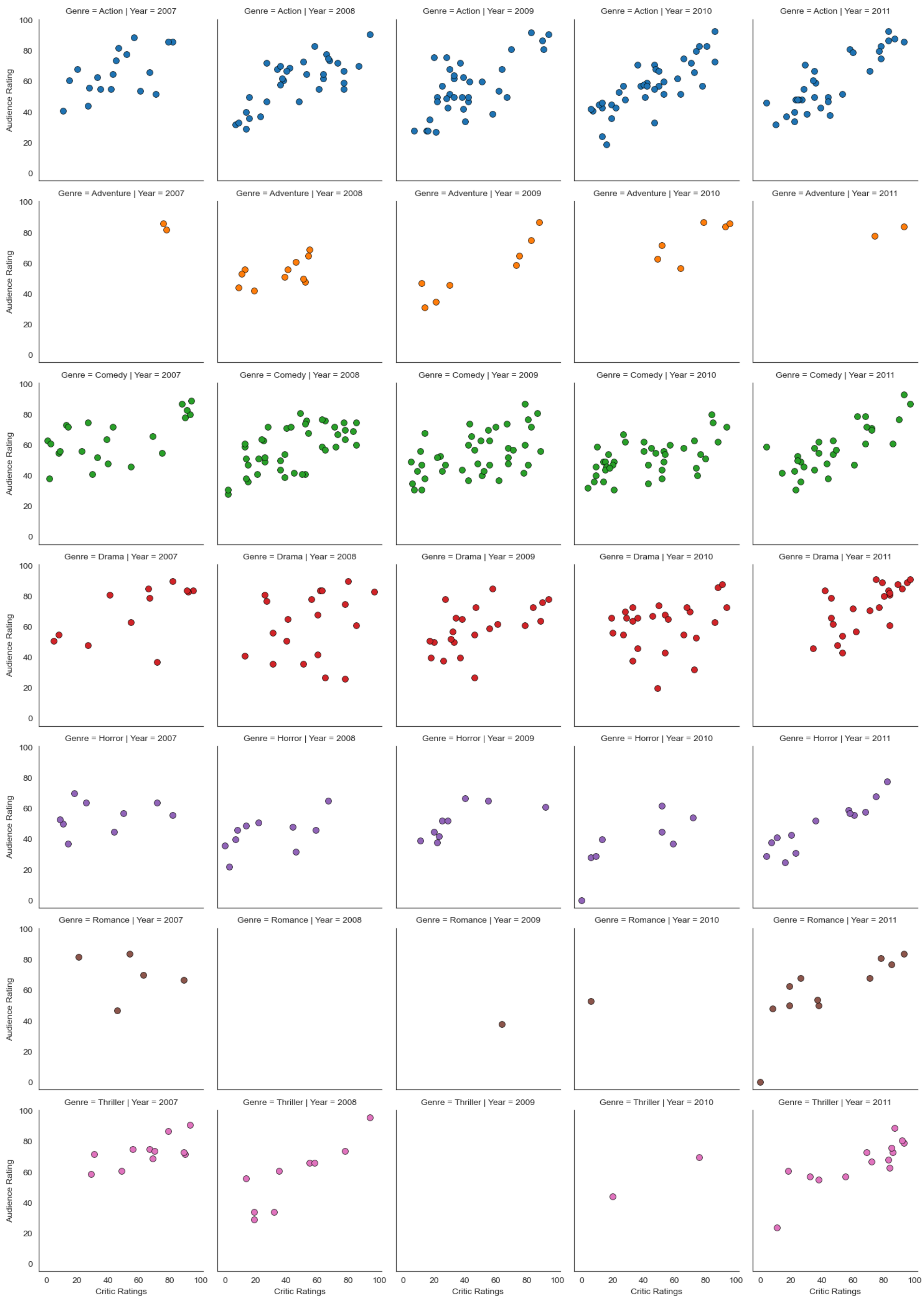


In [64]:
```
# you can populated any type of chat.

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



In [65]:

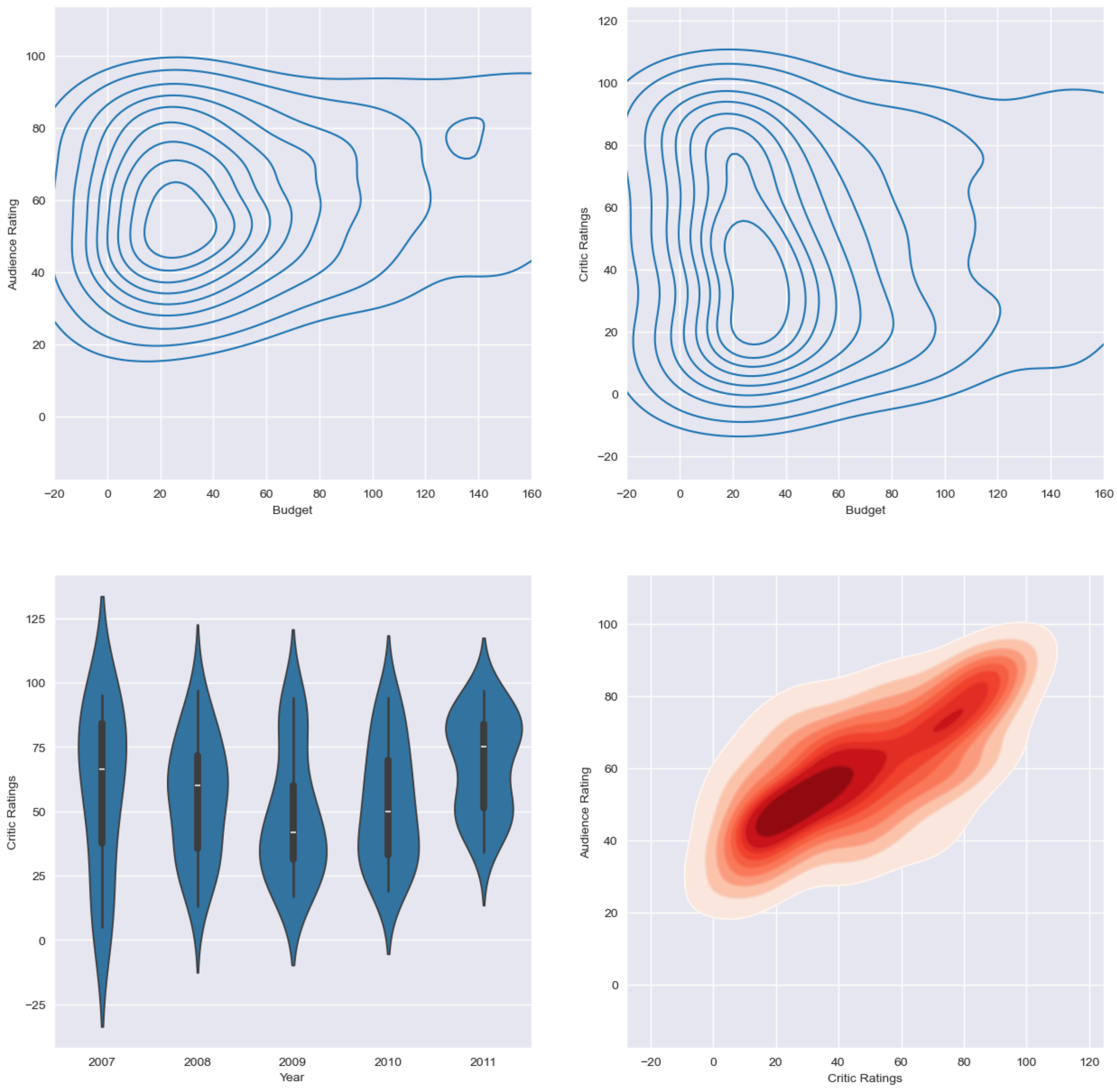
```
#
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
kws = dict(s=50, linewidth=0.5,edgecolor='black')
g = g.map(plt.scatter, 'Critic Ratings', 'Audience Rating',**kws ) #scatterplots
plt.show()
```



In [166…

```
# 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=movies.Budget,y=movies['Audience Rating'],ax=axes[0,0])
k2 = sns.kdeplot(x=movies.Budget,y=movies['Critic Ratings'],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 = 'Critic Rat

k4 = sns.kdeplot(x=movies['Critic Ratings'],y=movies['Audience Rating'],shade =

k4b = sns.kdeplot(x=movies['Critic Ratings'], y=movies['Audience Rating'],cmap='

plt.show()
```

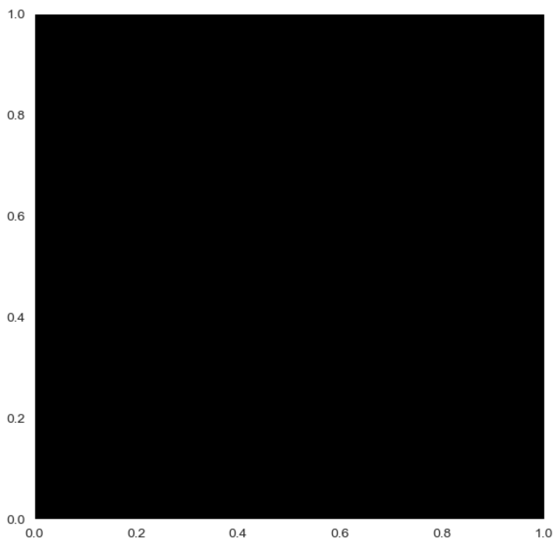


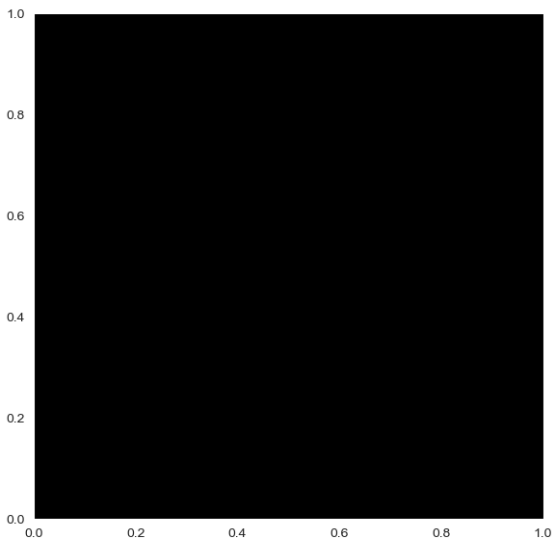
In [183…

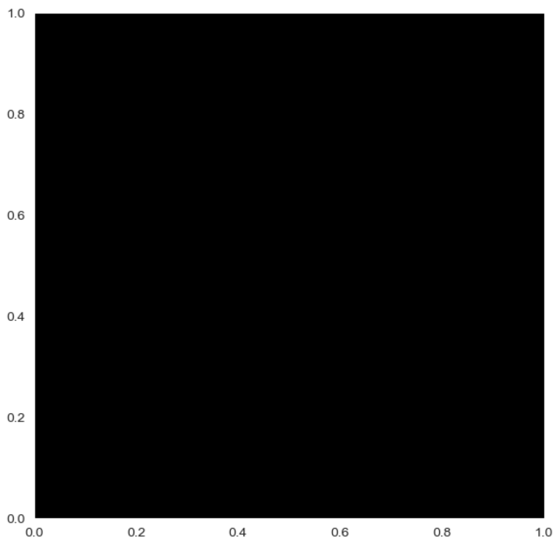
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Convert columns to numeric
movies['Budget'] = pd.to_numeric(movies['Budget'], errors='coerce')
movies['Audience Rating'] = pd.to_numeric(movies['Audience Rating'], errors='coe
movies['Critic Ratings'] = pd.to_numeric(movies['Critic Ratings'], errors='coerc

# Print column names and data types for debugging
print(movies.columns)
print(movies.dtypes)

# Set dark theme with black background
sns.set_style('dark', {'axes.facecolor': 'black'})
```

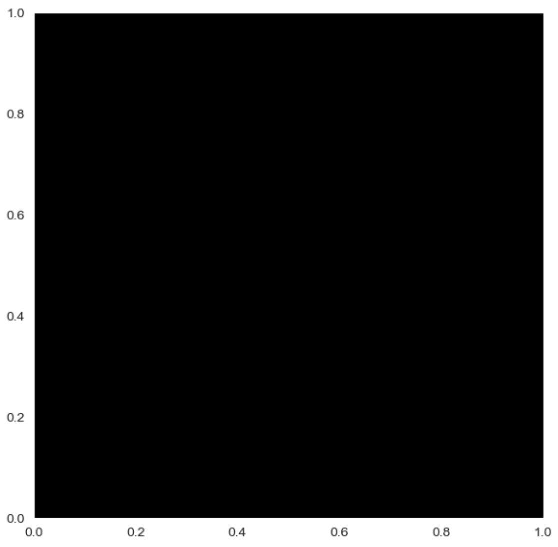
```python
# Create a figure with subplots
f, axes = plt.subplots(2, 2, figsize=(15, 15))

# KDE Plot [0,0]
sns.kdeplot(x=movies['Budget'], y=movies['Audience Rating'],
            fill=True, cmap='inferno', ax=axes[0, 0])
sns.kdeplot(x=movies['Budget'], y=movies['Audience Rating'],
            cmap='coolwarm', ax=axes[0, 0])

# KDE Plot [0,1]
sns.kdeplot(x=movies['Budget'], y=movies['Critic Ratings'],
            fill=True, cmap='inferno', ax=axes[0, 1])
sns.kdeplot(x=movies['Budget'], y=movies['Critic Ratings'],
            cmap='coolwarm', ax=axes[0, 1])

# Violin Plot [1,0] (Ensure 'Drama' exists)
if 'Genre' in movies.columns and 'Drama' in movies['Genre'].unique():
    sns.violinplot(data=movies[movies['Genre'] == 'Drama'],
                   x='Year', y='Critic Ratings', ax=axes[1, 0])
else:
    print("Genre column missing or 'Drama' not found in dataset")

# KDE Plot [1,1]
sns.kdeplot(x=movies['Critic Ratings'], y=movies['Audience Rating'],
            fill=True, cmap='Blues_r', ax=axes[1, 1])
sns.kdeplot(x=movies['Critic Ratings'], y=movies['Audience Rating'],
            cmap='coolwarm', ax=axes[1, 1])

# Set x-axis limits
axes[0, 0].set(xlim=(-20, 160))
axes[0, 1].set(xlim=(-20, 160))

plt.show()
```

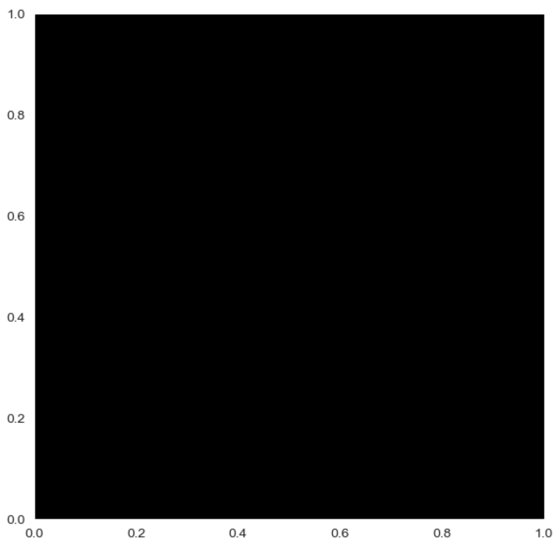
```
Index(['Film', 'Genre', 'Critic Ratings', 'Audience Rating', 'Budget', 'Year'], d
type='object')
Film                   category
Genre                  category
Critic Ratings            int64
Audience Rating           int64
Budget                    int64
Year                      int64
dtype: object
```

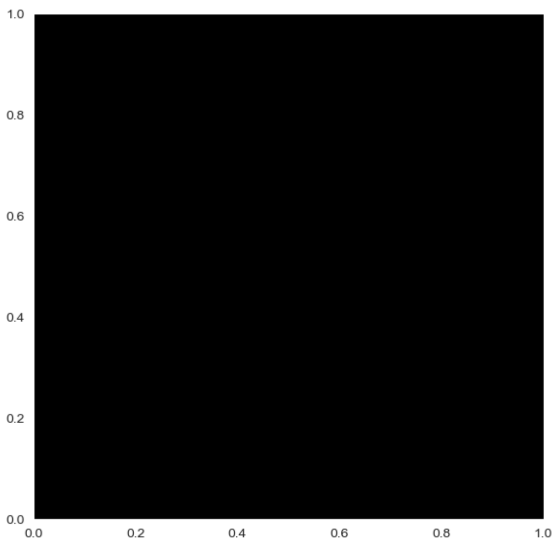

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

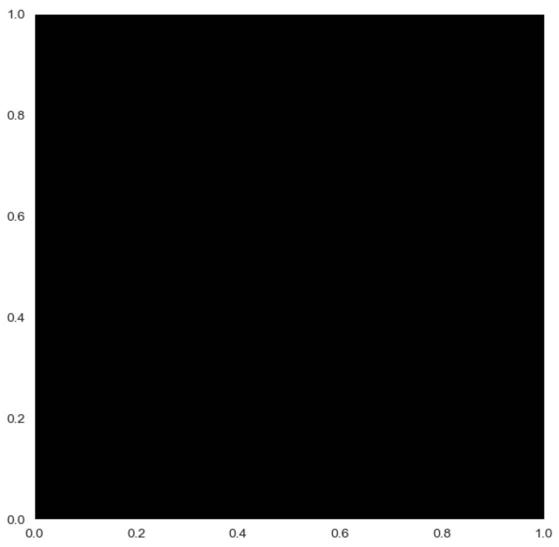

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

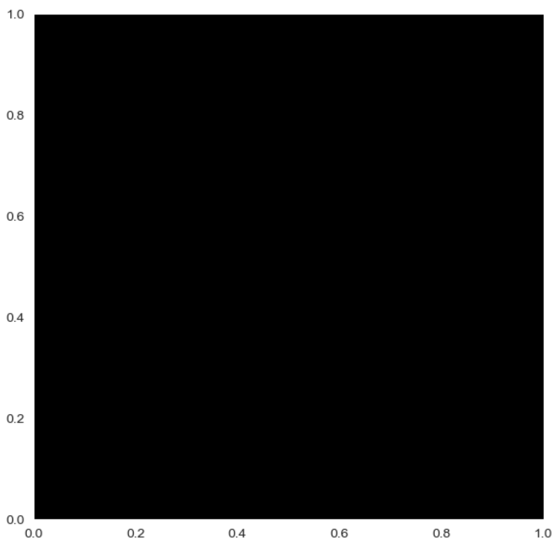

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

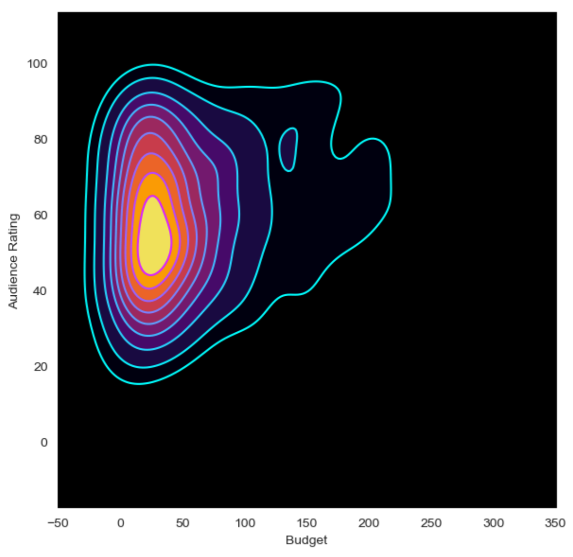

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

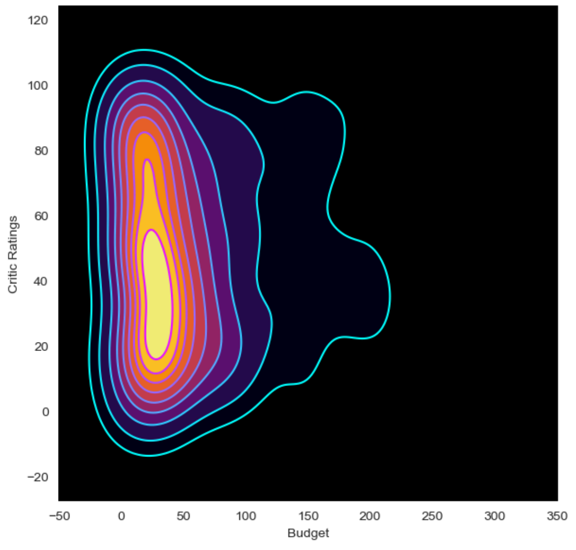

0.0
0.2
0.4
0.6
0.8
1.0

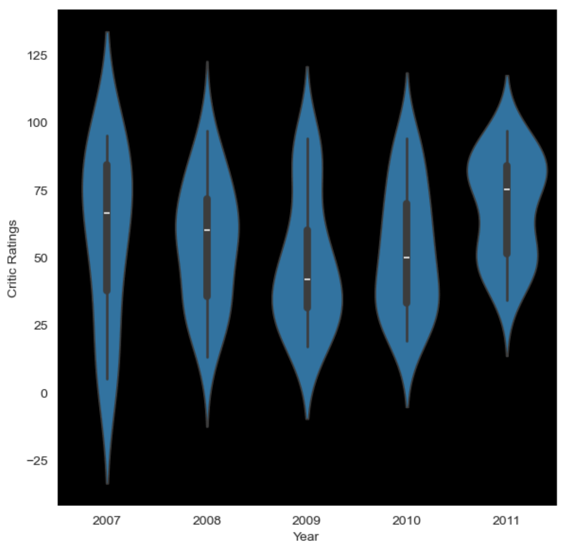

0.0
0.2
0.4
0.6
0.8
1.0

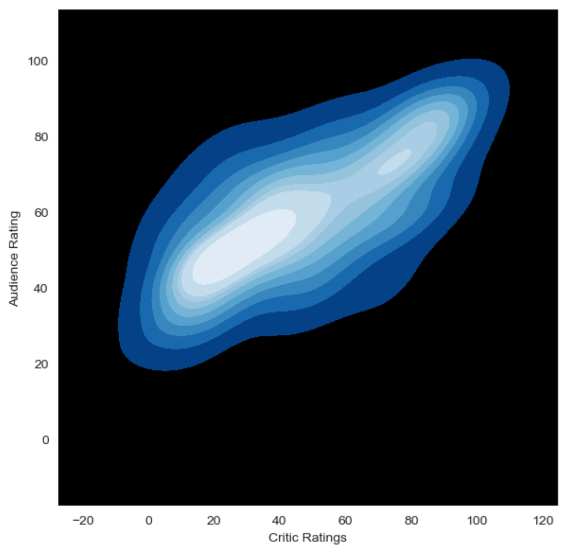

0.0
0.2
0.4
0.6
0.8
1.0

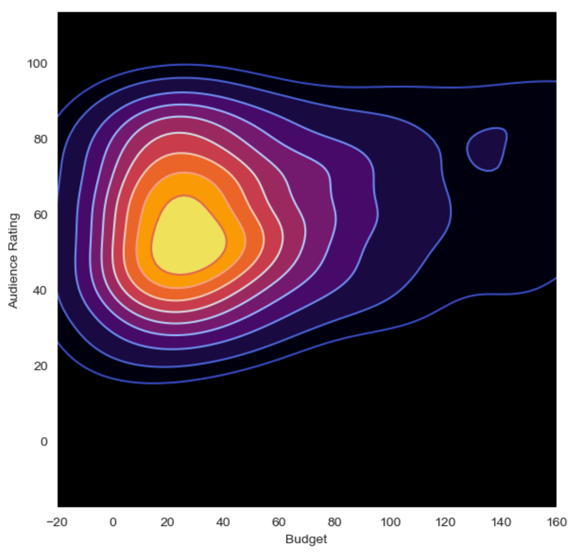

0.0
0.2
0.4
0.6
0.8
1.0

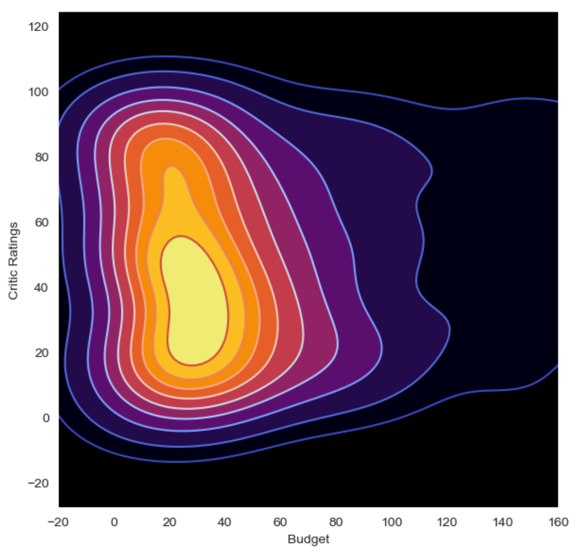

Audience Rating
100
80
60
40
20
0
−50
0
50
100
150
200
250
300
350
Budget

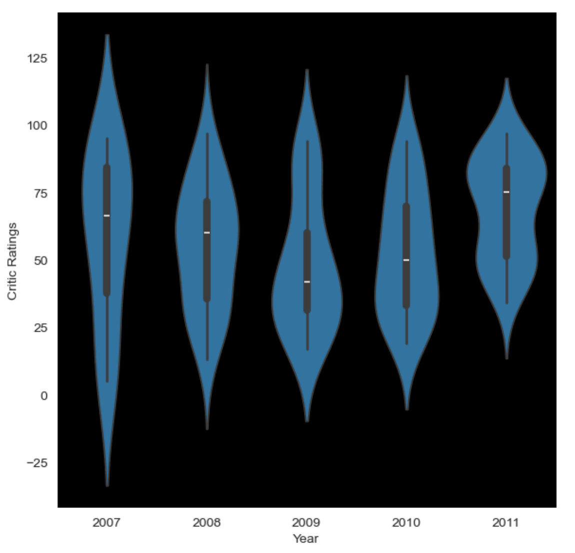

Critic Ratings
120
100
80
60
40
20
0
−20
−50
0
50
100
150
200
250
300
350
Budget

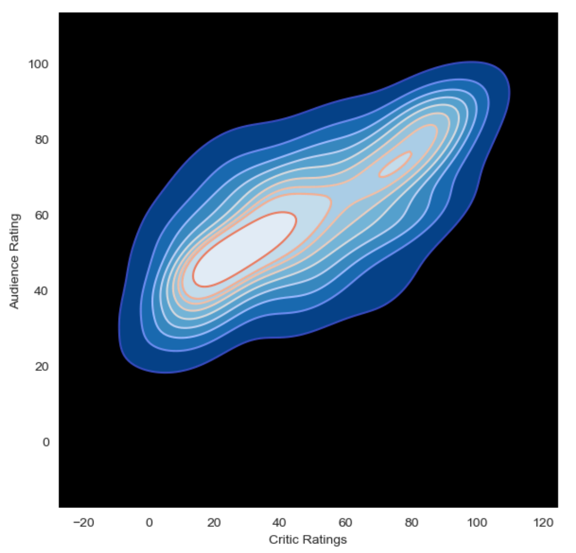

Critic Ratings
125
100
75
50
25
0
−25
2007
2008
2009
2010
2011
Year


Audience Rating
100
80
60
40
20
0
−20
0
20
40
60
80
100
120
Critic Ratings









```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```