In [109]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
md=pd.read_csv(r'S:\DOCS\7th\MOVIE RATINGS _ ADVANCE VISUALIZATION _ EDA 1\Movie-Rating.csv')
```

In [4]:

```
md
```

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]:

```
md.shape
```

Out[5]:

```
(559, 6)
```

In [6]:

```
md.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 [7]:

```
md.head()
```

Out[7]:

	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 [8]:

```
md.tail()
```

Out[8]:

	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 [11]:

```
md.isnull().any().any()
```

Out[11]:

```
False
```

In [13]:

```
md.duplicated().any()
```

Out[13]:

```
False
```

In [16]:

```
md.columns = ['Film','Genre','CriticRatings','AudienceRating','BudgetMillions','Year']
```

In [17]:

```
md.head()
```

Out[17]:

	Film	Genre	CriticRatings	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]:

```
md.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   CriticRatings   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 [19]:

```
# converting movie and genre to categorical type
```

In [20]:

```
md['Film']=md['Film'].astype('category')
```

In [21]:

```
md.Film
```

Out[21]:

```
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 Revol
t', 'Zodiac', 'Zombieland ', 'Zookeeper']
```

In [22]:

```
md.Genre=md.Genre.astype('category')
```

In [24]:

```
md.Genre
```

Out[24]:

```
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', 'Romance', 'Thriller']
```

In [25]:

```
md.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   CriticRatings   559 non-null    int64   
 3   AudienceRating  559 non-null    int64   
 4   BudgetMillions  559 non-null    int64   
 5   Year            559 non-null    int64   
dtypes: category(2), int64(4)
memory usage: 40.1 KB
```

In [26]:

```
# does year is real number no, beacuse  average,min max have no meaning ,so we convert to category
md['Year']=md['Year'].astype('category')
```

In [27]:

```
md.Year
```

Out[27]:

```
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 [28]:

```
md.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   CriticRatings   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 [29]:

```
md['Genre'].unique()
```

Out[29]:

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

In [30]:

```
md.describe()
```

Out[30]:

	CriticRatings	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

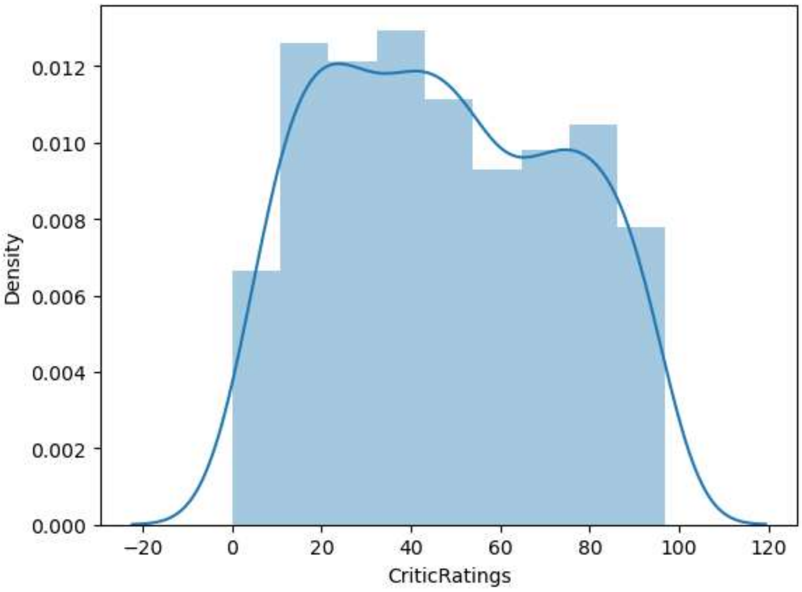
In [31]:

```
# anlysing distribution
```

In [34]:
```
visd=sns.distplot(md['CriticRatings'])
visd
```

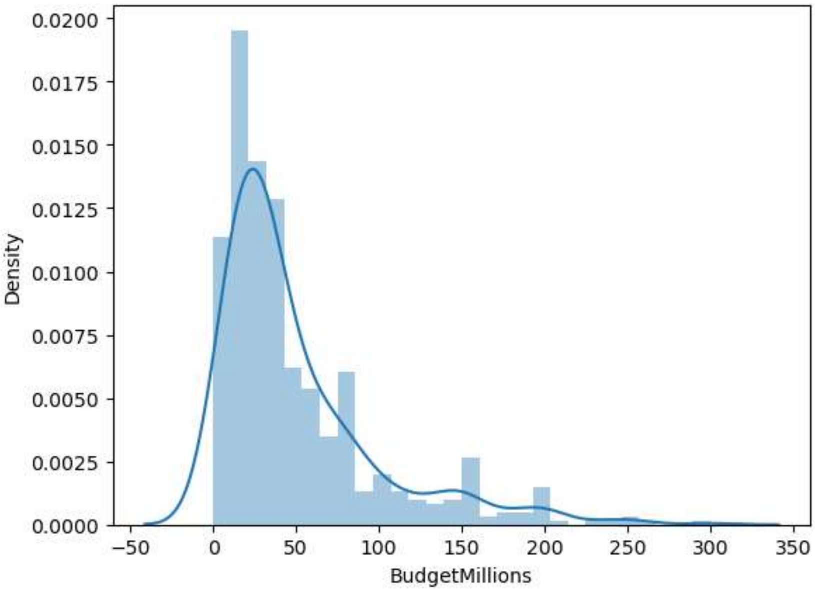
```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_9048\55492828.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457
ad6372750bbe5751)

  visd=sns.distplot(md['CriticRatings'])
```

Out[34]:
```
<Axes: xlabel='CriticRatings', ylabel='Density'>
```



In [36]:
```
visd1=sns.distplot(md['BudgetMillions'])
```

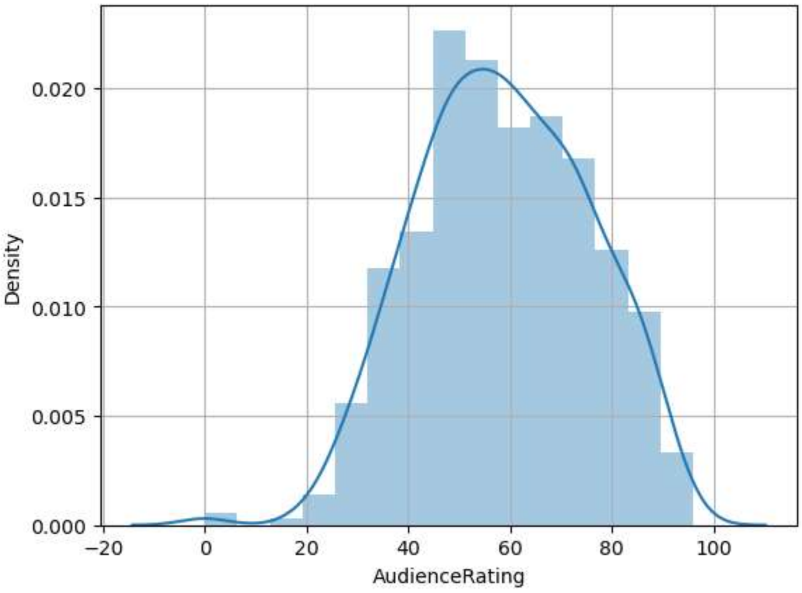
```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_9048\212141428.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457
ad6372750bbe5751)

  visd1=sns.distplot(md['BudgetMillions'])
```



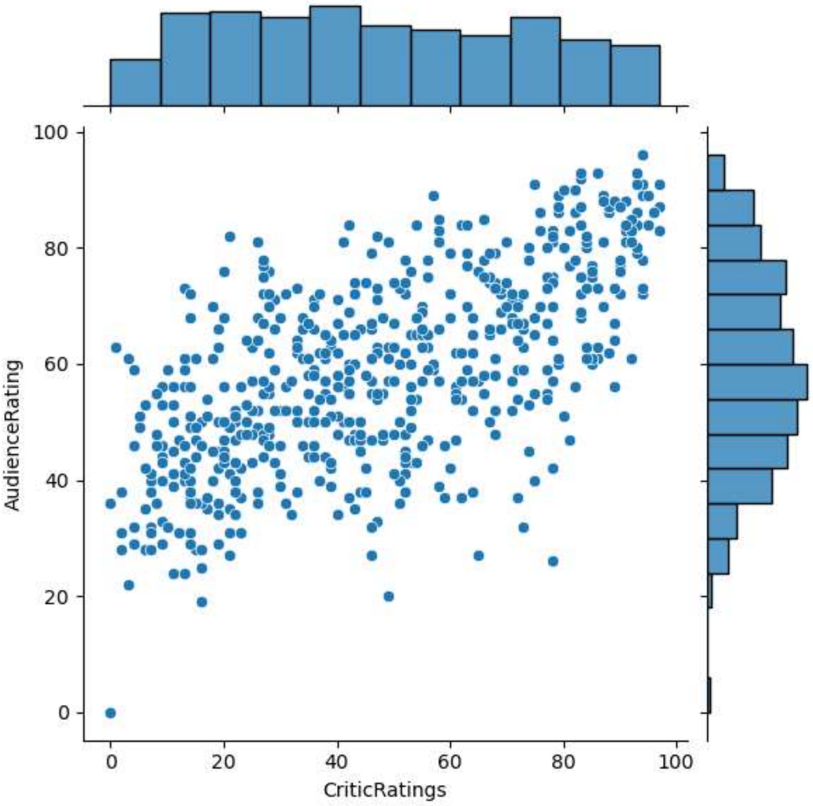
In [37]:
```
# the distribution plot is right skewed
```

In [45]:
```
visd2=sns.distplot(md['AudienceRating'] , bins=15)
plt.grid(True)
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_9048\725103375.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457
ad6372750bbe5751)

  visd2=sns.distplot(md['AudienceRating'] , bins=15)
```

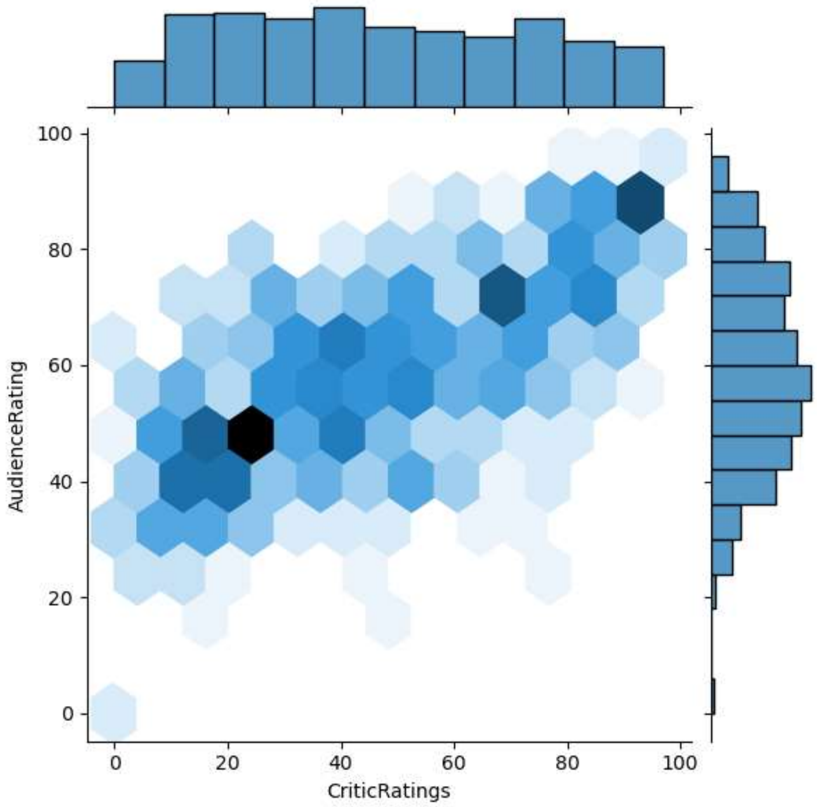


In [46]:
```
vis1 = sns.jointplot(data=md, x='CriticRatings', y='AudienceRating')
```



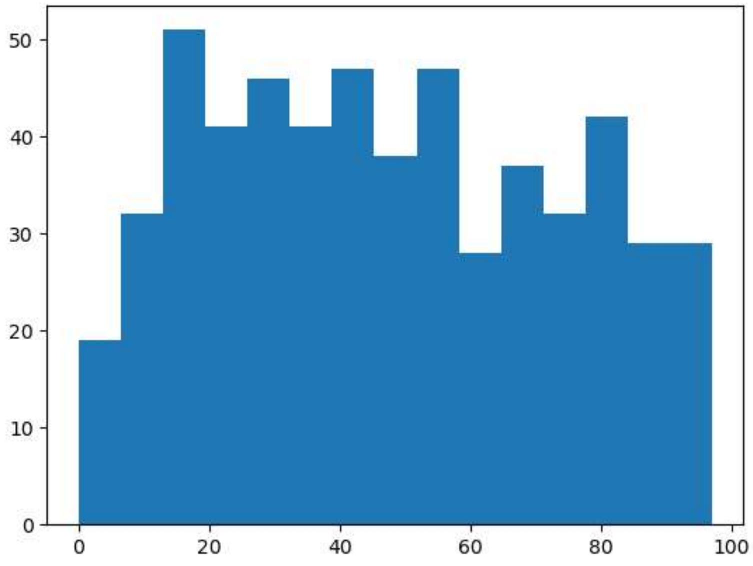
In [48]:

```
vis2 = sns.jointplot(data=md, x='CriticRatings', y='AudienceRating' ,kind='hex')
```

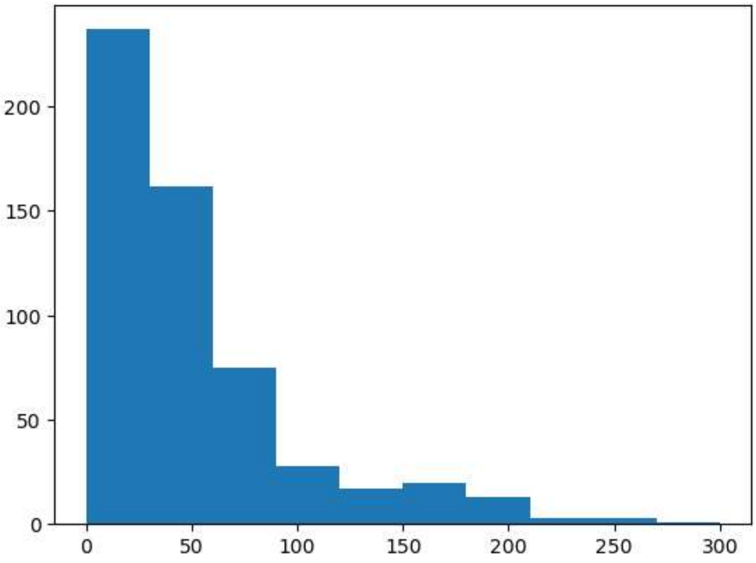


In [51]:

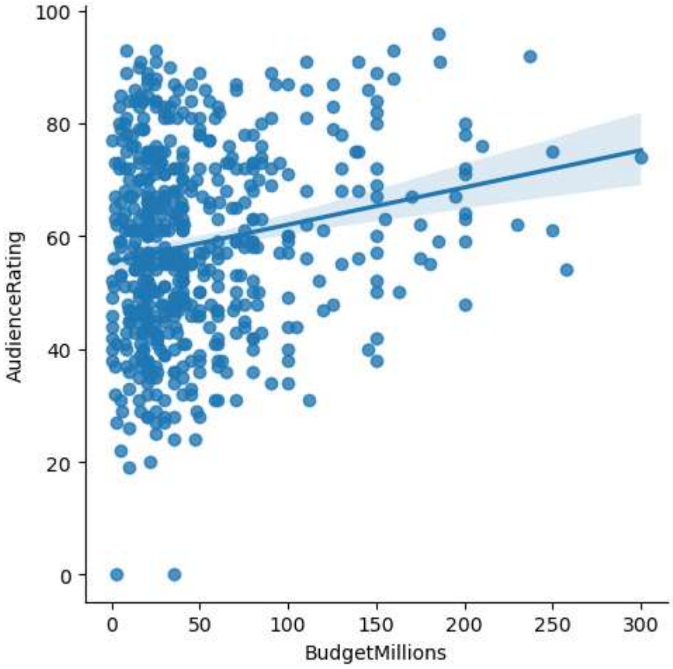
```
n1=plt.hist(md.CriticRatings, bins=15)
```



In [52]:
```
h1=plt.hist(md.BudgetMillions)
```



In [53]:
```
vis3=sns.lmplot(data=md, x='BudgetMillions', y='AudienceRating')
```



In [56]:
```
md.Genre.unique()
```

Out[56]:
```
['Comedy', 'Adventure', 'Action', 'Horror', 'Drama', 'Romance', 'Thriller']
Categories (7, object): ['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance', 'Thriller']
```

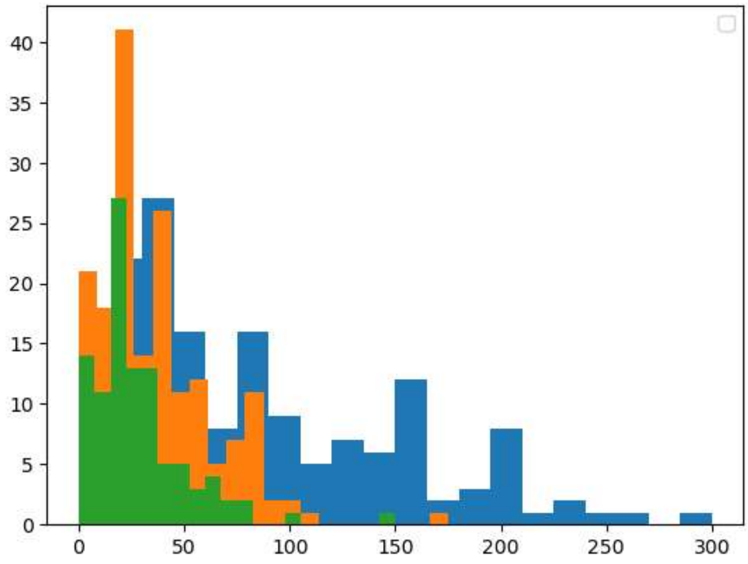
In [75]:

```python
plt.hist(md [md.Genre == 'Action'].BudgetMillions, bins=20)
plt.hist(md [md.Genre == 'Comedy'].BudgetMillions, bins=20)
plt.hist(md [md.Genre == 'Drama'].BudgetMillions, bins=20)
plt.legend()

```

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

Out[75]:

```
<matplotlib.legend.Legend at 0x1ecc935a410>
```

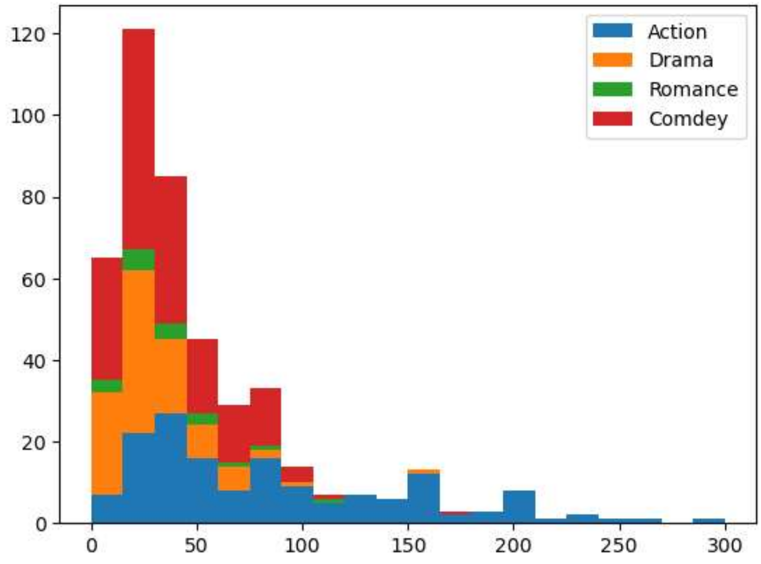


In [88]:

```python
plt.hist ([md[md.Genre=='Action'].BudgetMillions,\
          md[md.Genre == 'Drama'].BudgetMillions,\
          md[md.Genre == 'Romance'].BudgetMillions,\
          md[md.Genre == 'Comedy'].BudgetMillions], bins=20,stacked=True )
plt.legend(['Action','Drama','Romance','Comdey'])
```

Out[88]:

```
<matplotlib.legend.Legend at 0x1eccd400910>
```



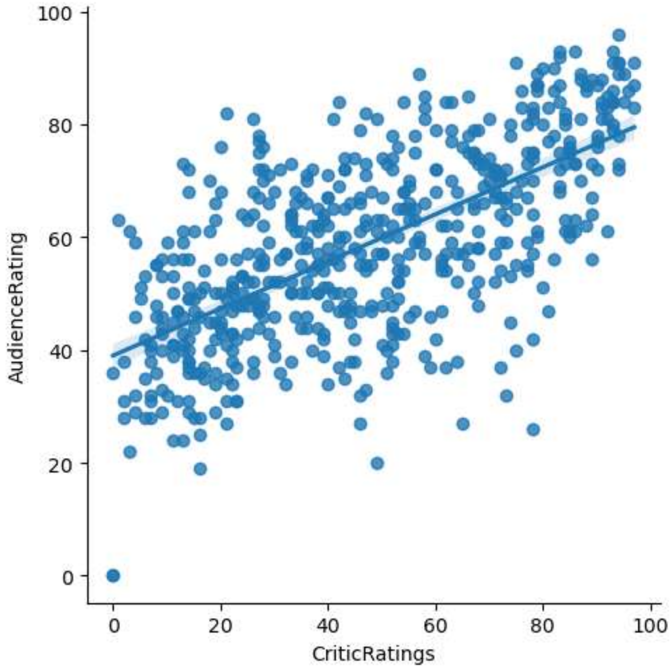
In [93]:

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

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

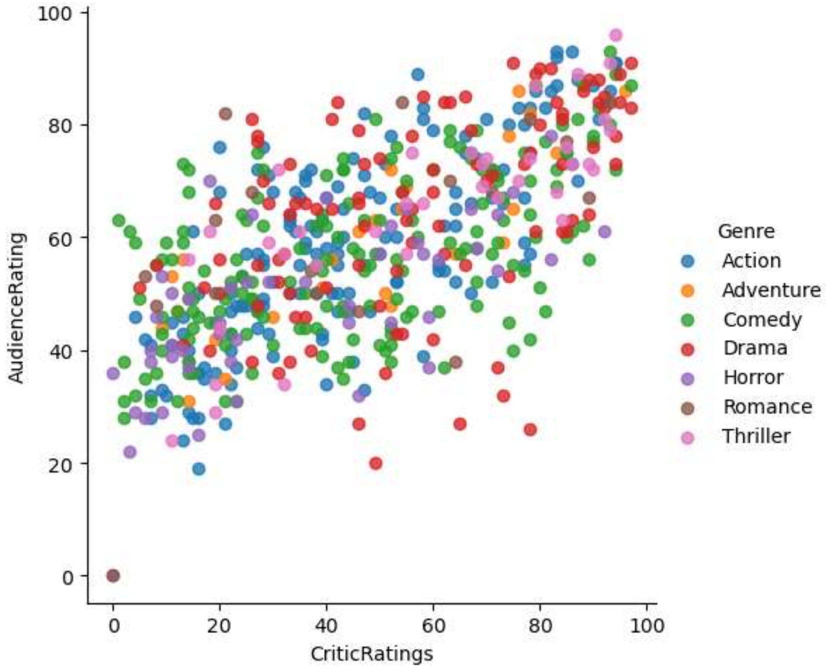
In [94]:
```
sns.lmplot(data=md, x='CriticRatings', y='AudienceRating')
```

Out[94]:
```
<seaborn.axisgrid.FacetGrid at 0x1eccd5d29d0>
```



In [98]:
```
sns.lmplot(data=md, x='CriticRatings', y='AudienceRating' ,hue='Genre', fit_reg=False)
```

Out[98]:
```
<seaborn.axisgrid.FacetGrid at 0x1eccd79e350>
```



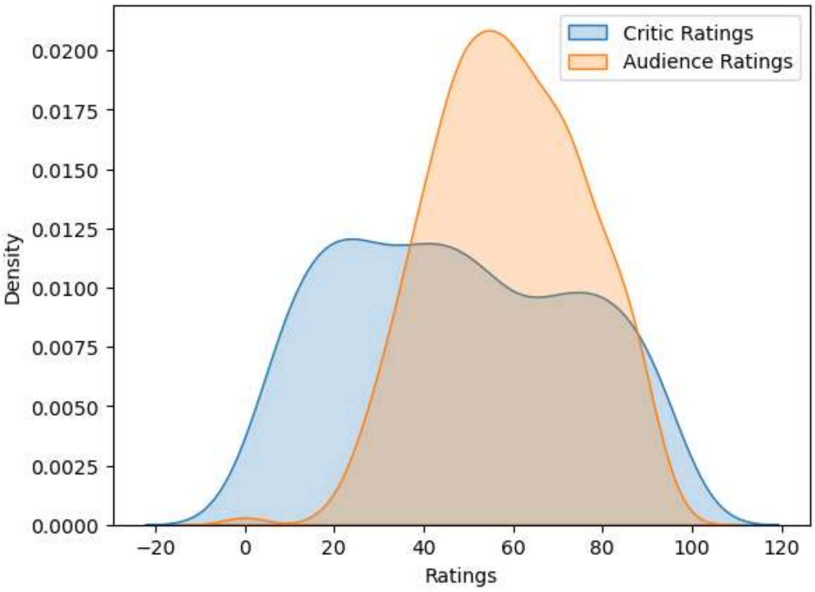
In [99]:
```
#kde plot
```

In [111]:

```
sns.kdeplot(data=md['CriticRatings'], label='Critic Ratings', shade=True)
sns.kdeplot(data=md['AudienceRating'], label='Audience Ratings', shade=True)
plt.xlabel('Ratings')
plt.ylabel('Density')
plt.legend()
```

Out[111]:

```
<matplotlib.legend.Legend at 0x1eccd9f8fd0>
```



In [115]:

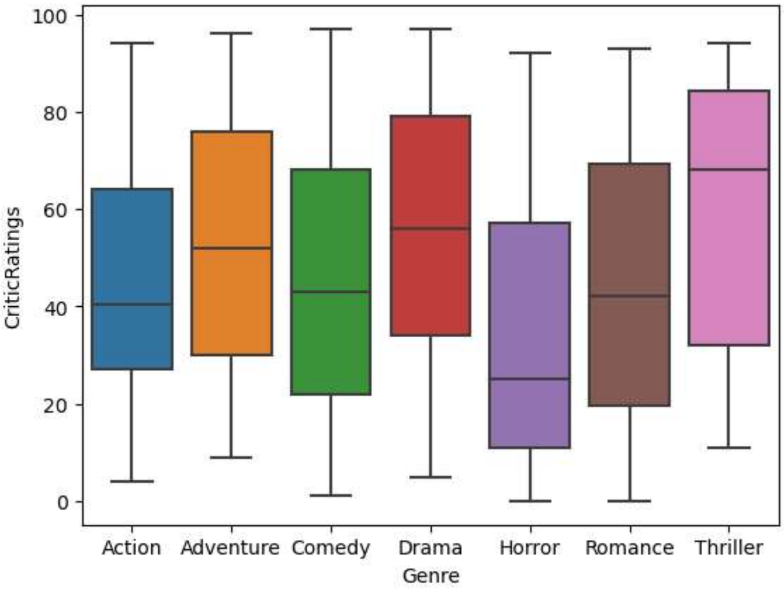
```
# box plot
```

In [116]:

```
sns.boxplot(data=md, x='Genre', y='CriticRatings')
```

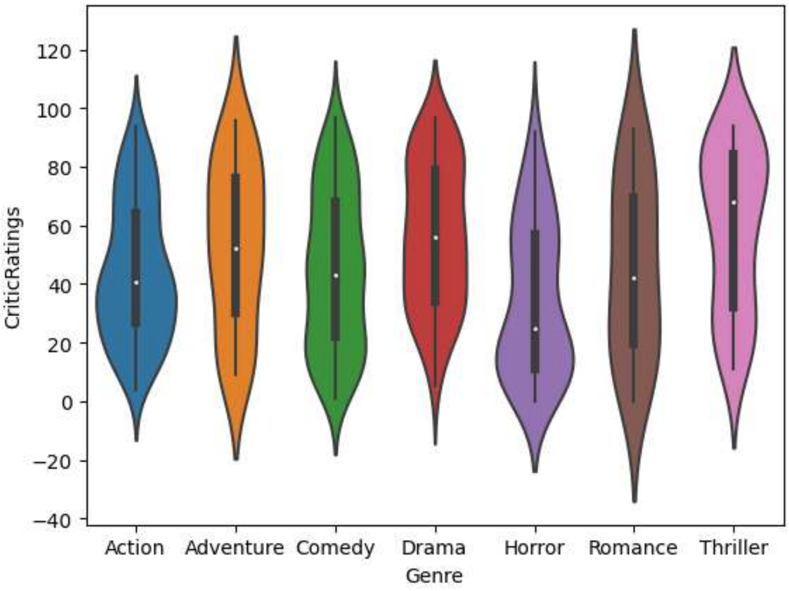
Out[116]:

```
<Axes: xlabel='Genre', ylabel='CriticRatings'>
```

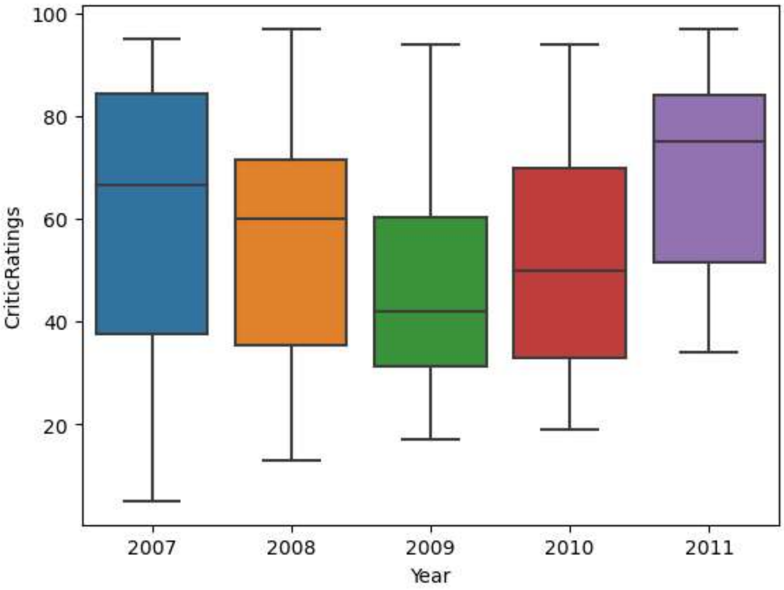


In [118]:
```
sns.violinplot(data=md, x='Genre',y='CriticRatings')
```

Out[118]:
```
<Axes: xlabel='Genre', ylabel='CriticRatings'>
```



In [121]:
```
w1=sns.boxplot(data=md[md.Genre== 'Drama'], x='Year',y='CriticRatings')
```



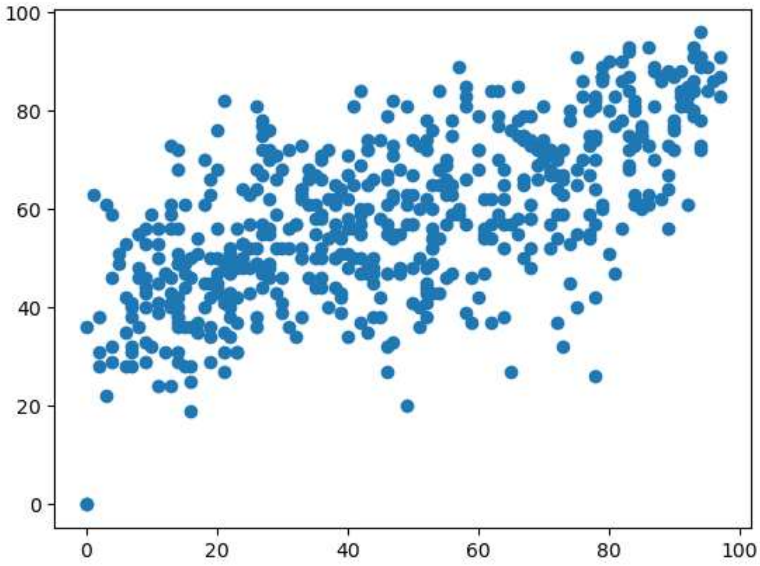
In [126]:

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

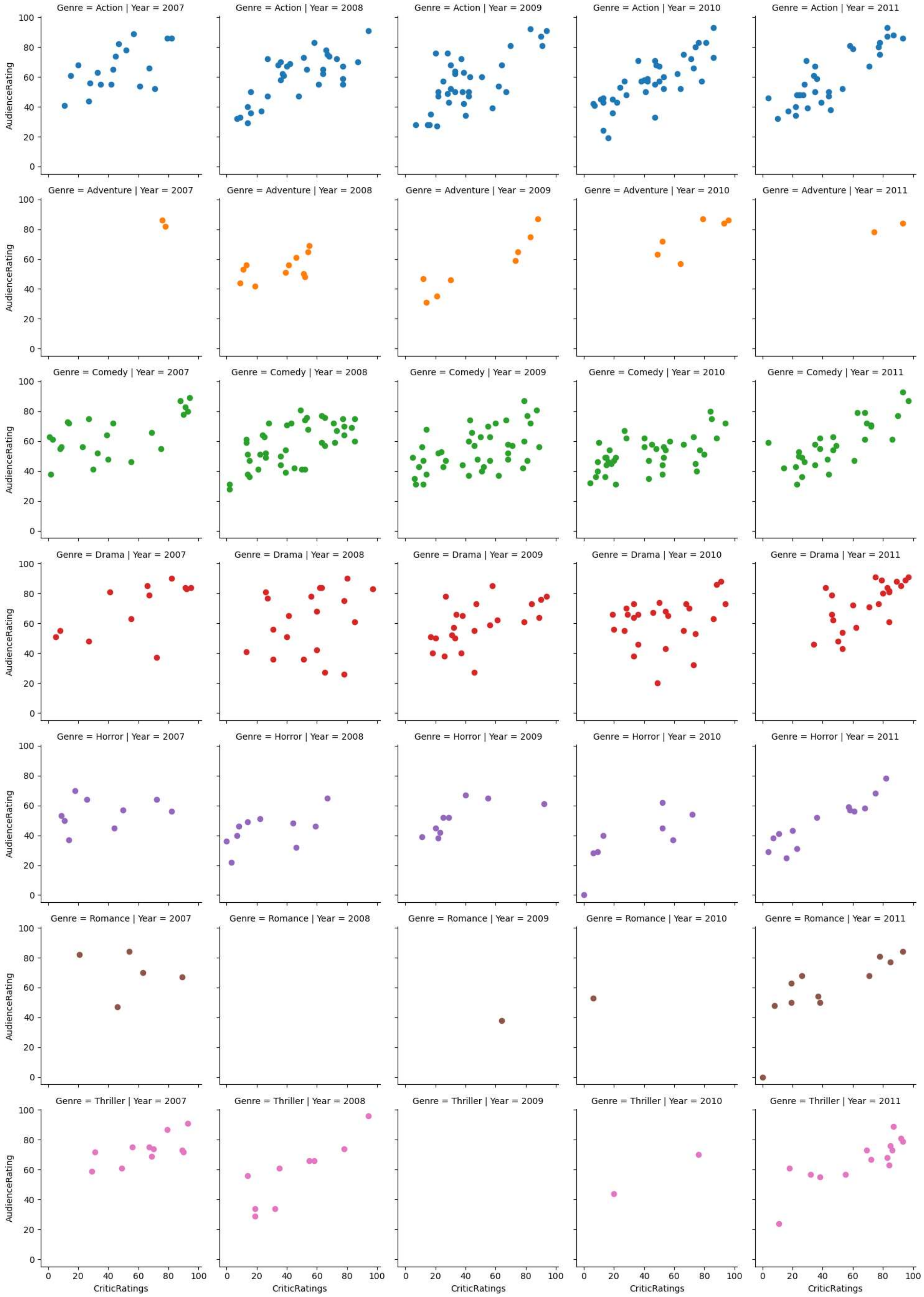
Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011
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
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

In [127]:
```
plt.scatter(md.CriticRatings,md.AudienceRating)
```

Out[127]:
```
<matplotlib.collections.PathCollection at 0x1ecd2450f50>
```



In [130]:
```
g=sns.FacetGrid(md, row='Genre', col='Year' , hue='Genre')
g= g.map(plt.scatter, 'CriticRatings','AudienceRating')
```

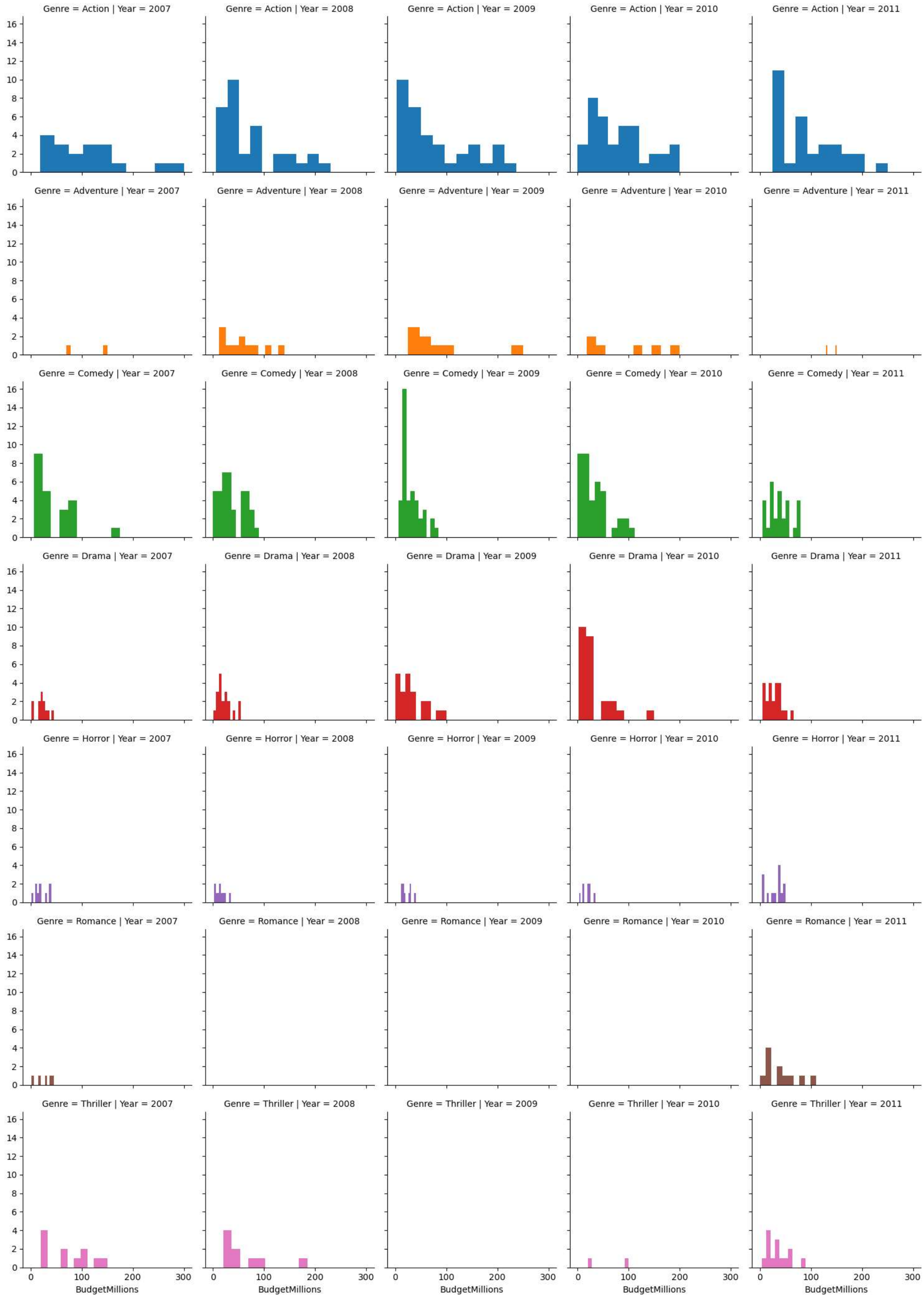


In [135]:

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

Out[135]:

```
<seaborn.axisgrid.FacetGrid at 0x1ecd2151150>
```



In [136]:

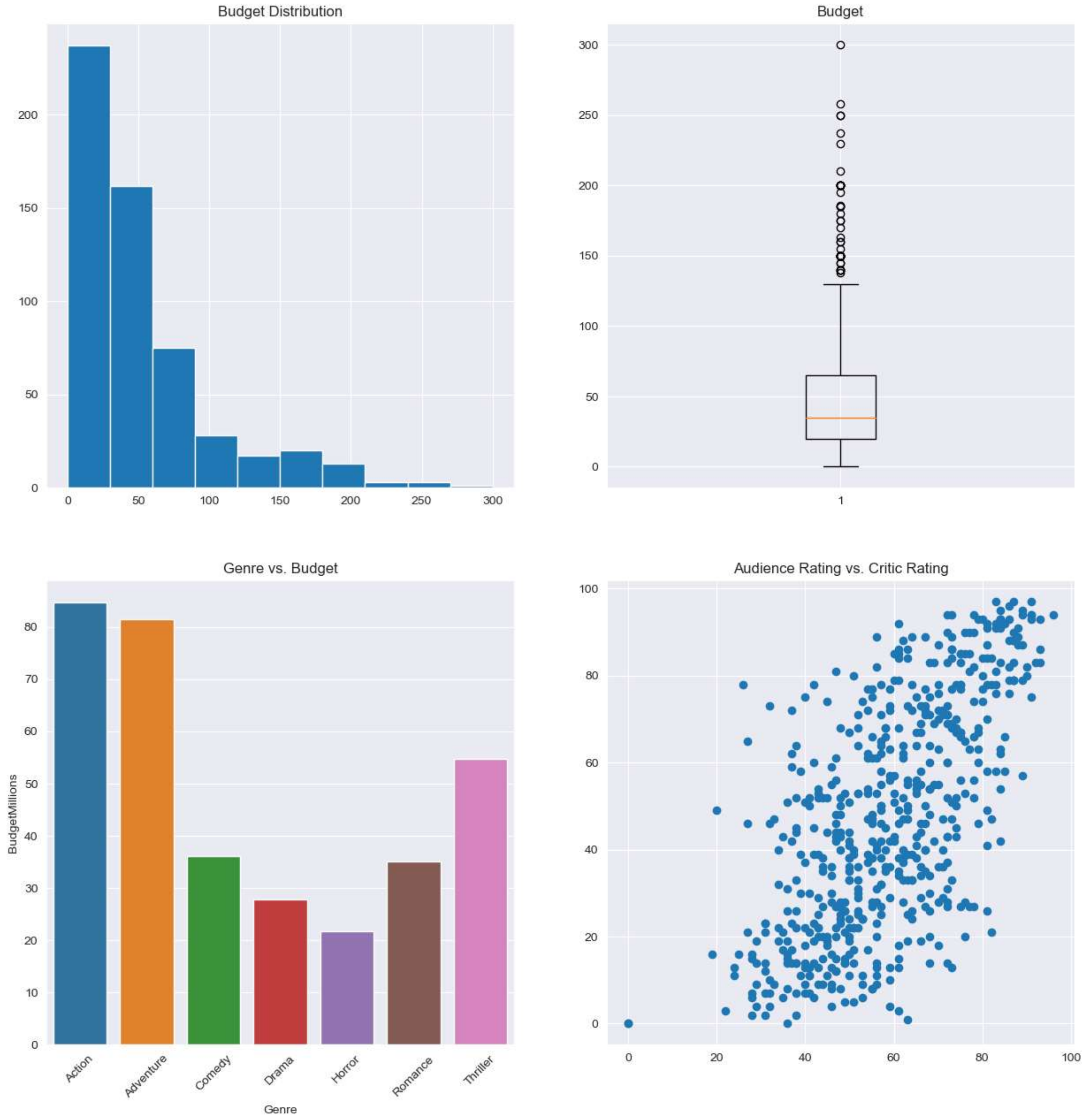
```
# building dashboard
```

In [177]:

```python
sns.set_style('darkgrid')

# Create a 2x2 grid of subplots
f, axes = plt.subplots(2, 2, figsize=(15, 15))

# Plot a histogram in the first subplot (axes[0, 0])
h3 = axes[0, 0].hist(md['BudgetMillions'])
axes[0, 0].set_title('Budget Distribution')

# Plot a scatter plot in the second subplot (axes[0, 1])
h4 = axes[0, 1].boxplot(md['BudgetMillions'])
axes[0, 1].set_title('Budget')

# Plot a bar plot in the third subplot (axes[1, 0])
h5 = sns.barplot(x=md['Genre'], y=md['BudgetMillions'], ax=axes[1, 0], ci=None)
axes[1, 0].set_title('Genre vs. Budget')
axes[1, 0].tick_params(axis='x', rotation=45)  # Rotate x-axis labels for better readability

# Plot a scatter plot in the fourth subplot (axes[1, 1])
h6 = axes[1, 1].scatter(md['AudienceRating'], md['CriticRatings'])
axes[1, 1].set_title('Audience Rating vs. Critic Rating')
```

Out[177]:

```
Text(0.5, 1.0, 'Audience Rating vs. Critic Rating')
```



In [ ]:

```
 
```