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

In [2]:
```
movies=pd.read_csv(r"C:\Users\nandh\Downloads\Movie-Rating.csv")
```

In [3]:
```
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]:
```
type(movies)
```

Out[4]:
```
pandas.core.frame.DataFrame
```

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

Out[5]:
```
559
```

In [6]:
```
print(np.__version__)
```

```
1.26.4
```

In [7]:
```
import pandas
print(pandas.__version__)
```

```
2.2.2
```

In [8]:
```
movies.columns
```

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

In [9]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Film                       559 non-null    object
 1   Genre                      559 non-null    object
 2   Rotten Tomatoes Ratings %  559 non-null    int64 
 3   Audience Ratings %         559 non-null    int64 
 4   Budget (million $)         559 non-null    int64 
 5   Year of release            559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [10]:
```
movies.shape
```

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

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

Out[11]:

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

In [12]:
```
movies.tail()
```

Out[12]:

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

In [13]:
```
movies.columns=['Film', 'Genre', 'CriticRatings', 'AudienceRatings', 'BudgetMillion
```

In [14]:
```
movies.columns
```

Out[14]:
```
Index(['Film', 'Genre', 'CriticRatings', 'AudienceRatings', 'BudgetMillions',
       'Year'],
      dtype='object')
```

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

Out[15]:

	Film	Genre	CriticRatings	AudienceRatings	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 [16]:
```
movies.describe()#stastistical discription
```

Out[16]:

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

In [18]:
```
movies
```

Out[18]:

	Film	Genre	CriticRatings	AudienceRatings	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
...	...	...	...	...	...	...
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 [19]:
```
movies.Genre=movies.Genre.astype('category')
```

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

In [21]:
```
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   CriticRatings    559 non-null    int64   
 3   AudienceRatings  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 [22]:
```
movies.describe()
```

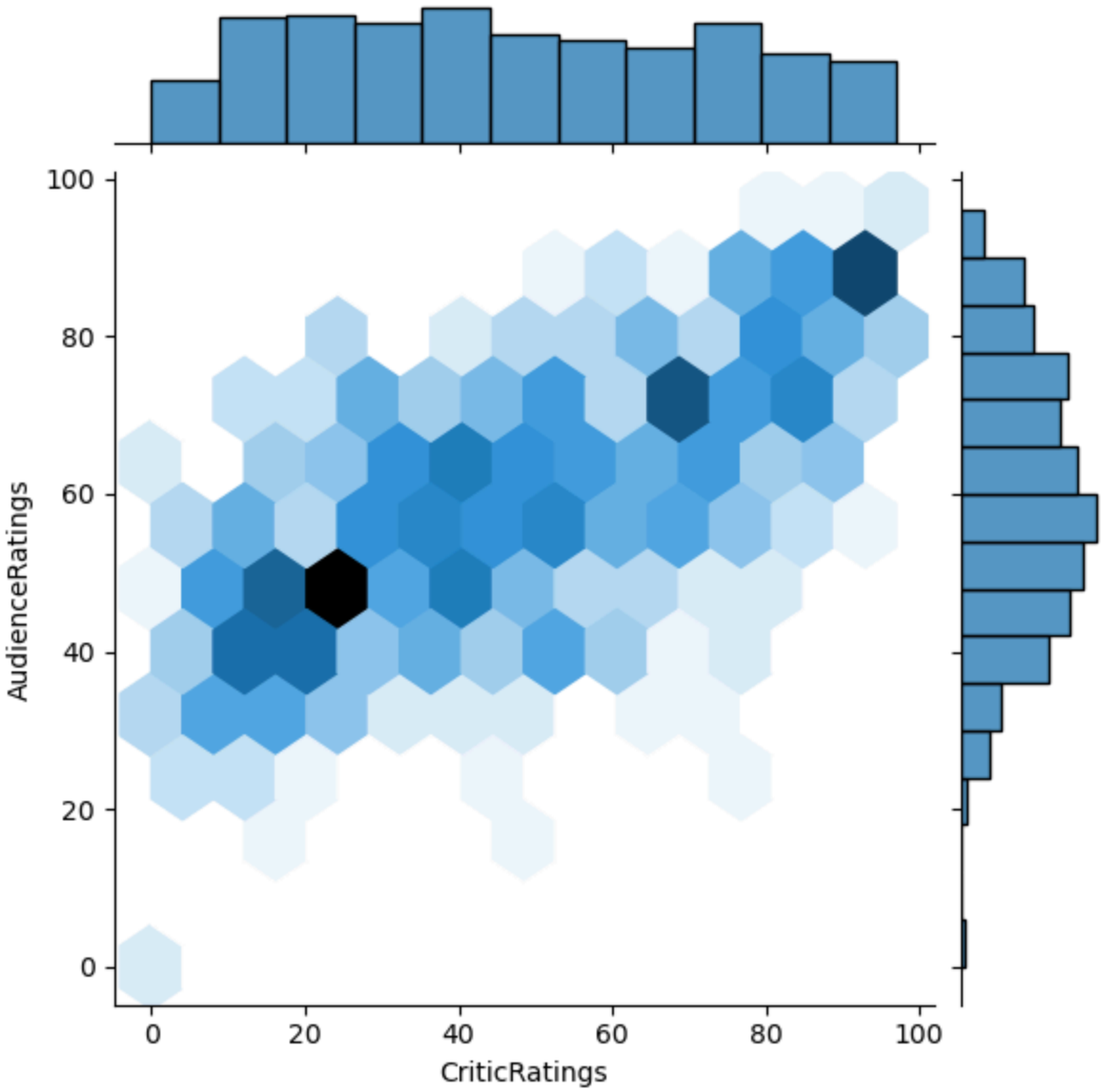
Out[22]:

	CriticRatings	AudienceRatings	BudgetMillions
count	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136
std	26.413091	16.826887	48.731817
min	0.000000	0.000000	0.000000
25%	25.000000	47.000000	20.000000
50%	46.000000	58.000000	35.000000
75%	70.000000	72.000000	65.000000
max	97.000000	96.000000	300.000000

python constructor **init**(or)special method

Jointplot for Bivariant

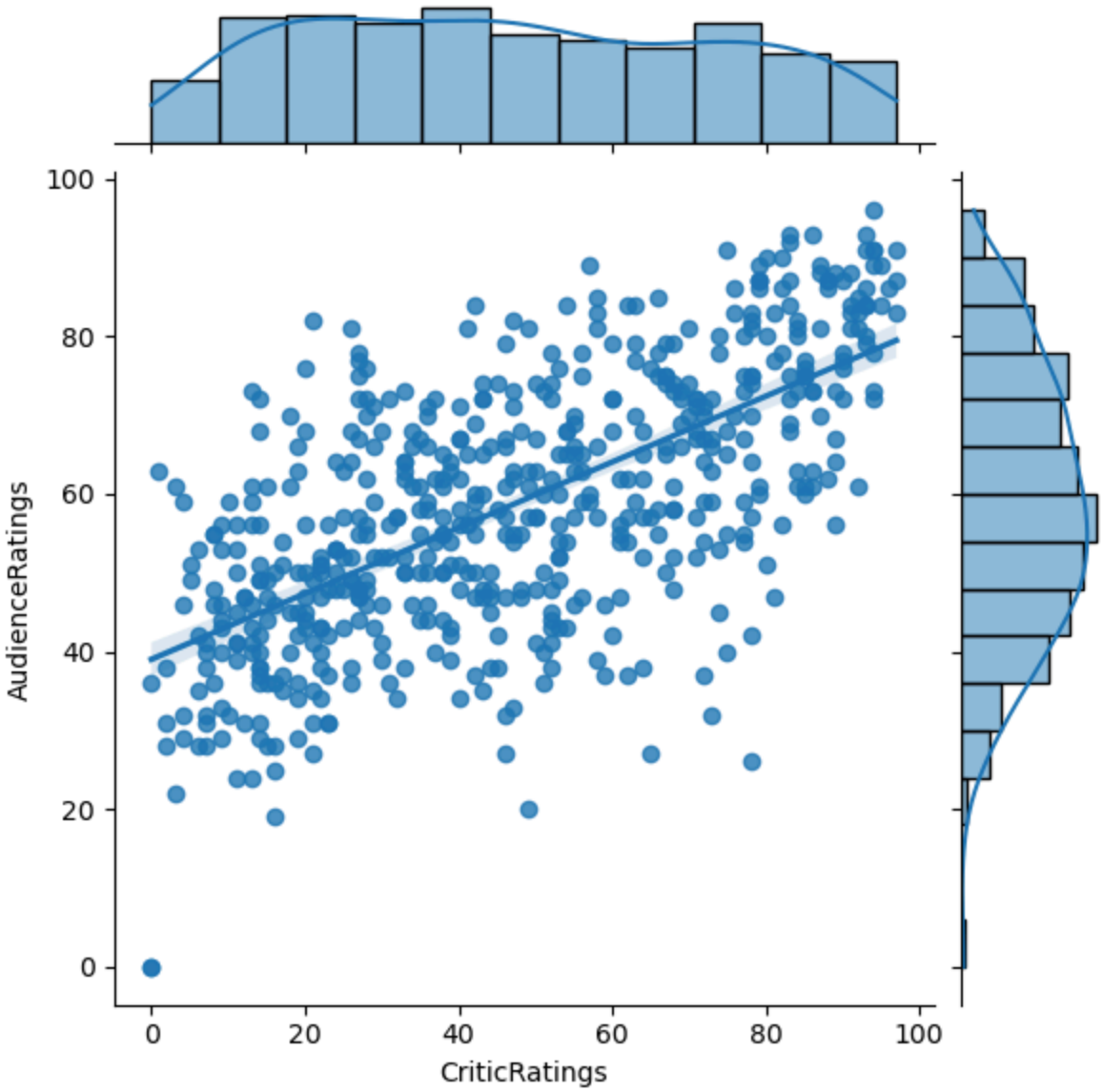
In [23]:
```
j=sns.jointplot(data=movies,x='CriticRatings',y='AudienceRatings',kind='hex')
plt.show()
```



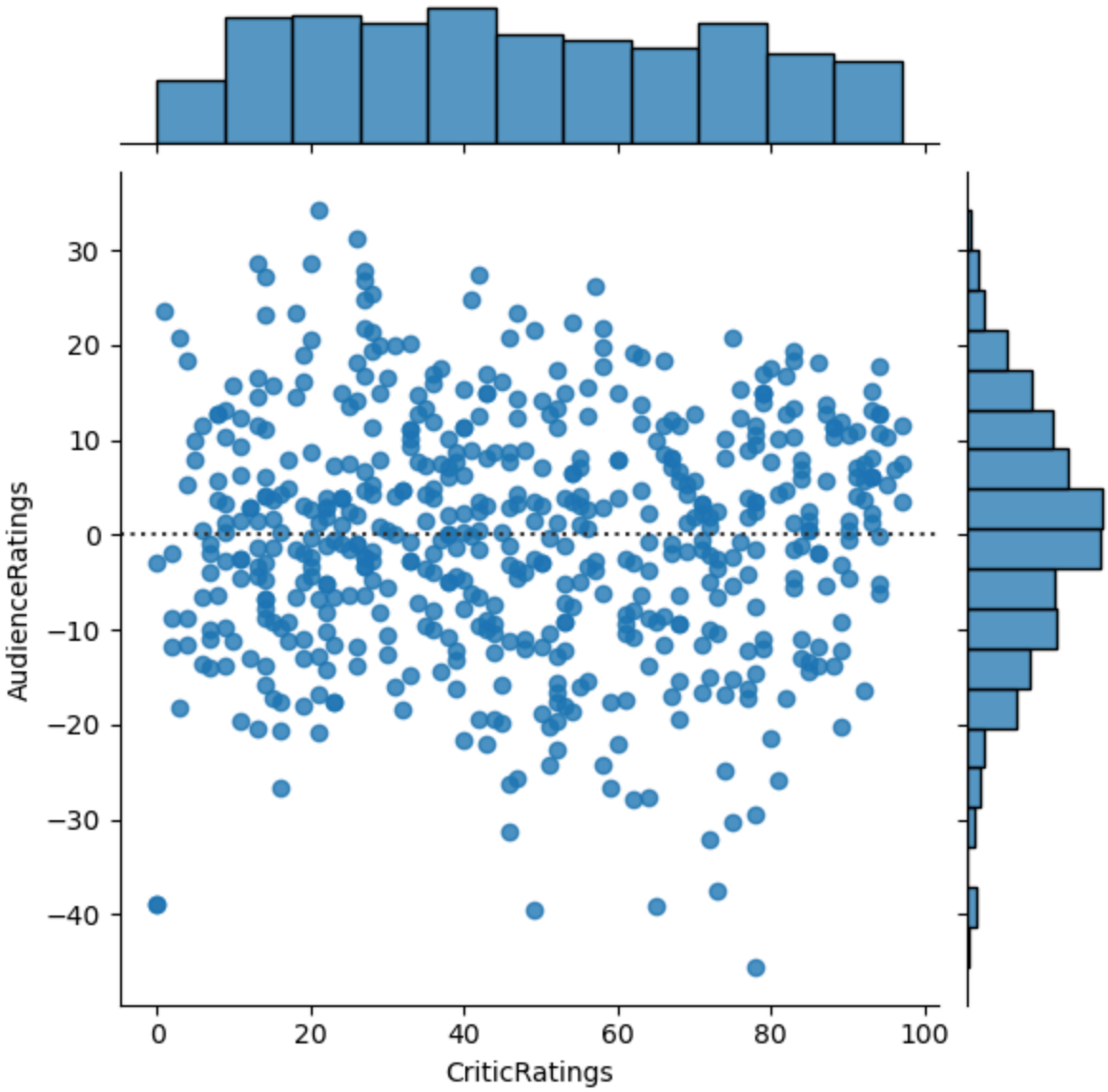
In [24]:
```
movies.columns
```

Out[24]:
```
Index(['Film', 'Genre', 'CriticRatings', 'AudienceRatings', 'BudgetMillions',
       'Year'],
      dtype='object')
```

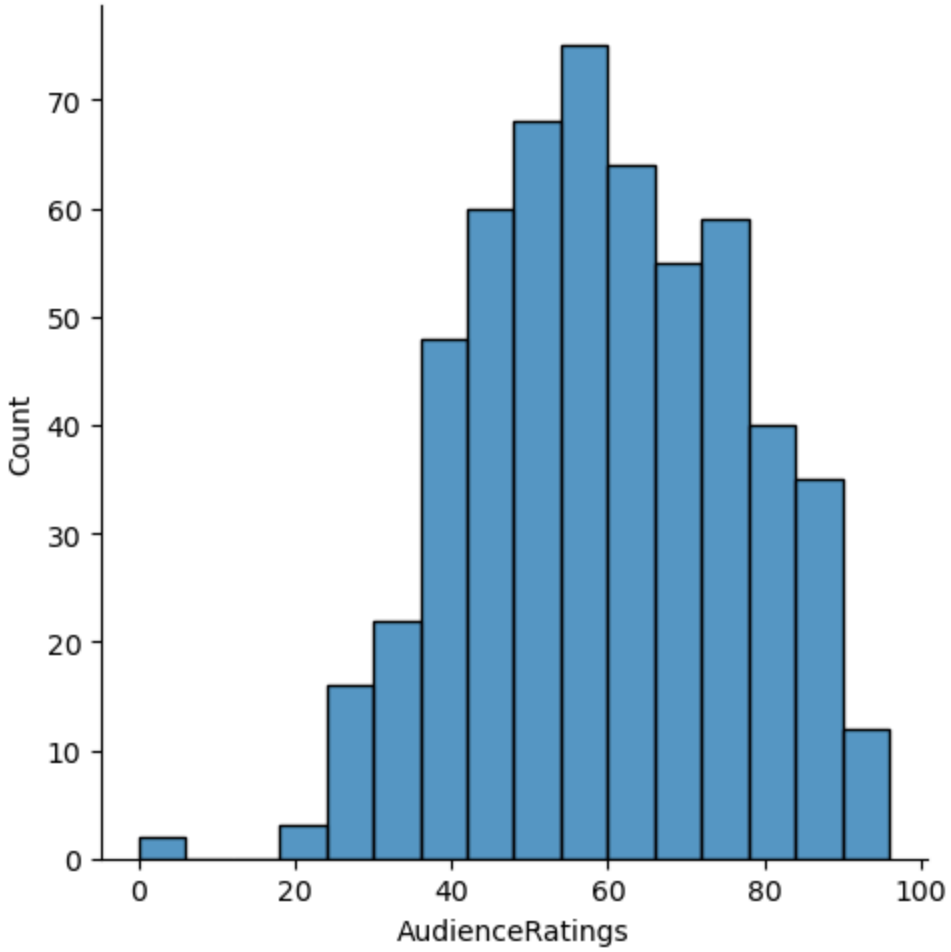
In [25]:
```
j=sns.jointplot(data=movies,x='CriticRatings',y='AudienceRatings',kind='reg')
plt.show()#'scatter', 'hist', 'hex', 'kde', 'reg', 'resid']
```



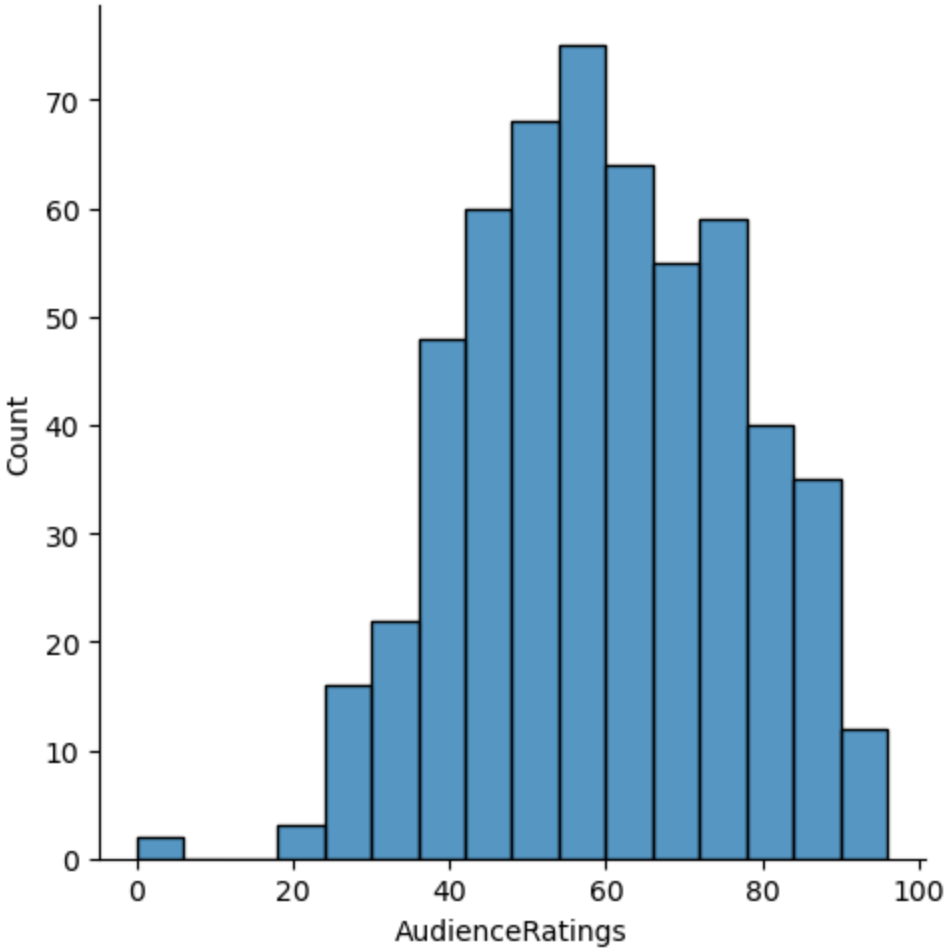
In [26]:
```
j=sns.jointplot(data=movies,x='CriticRatings',y='AudienceRatings',kind='resid')
plt.show()
```



In [27]:
```
m1=sns.displot(movies.AudienceRatings)
plt.show()
```



In [28]:
```
m1=sns.displot(movies.AudienceRatings )
plt.show()
```



In [29]:
```
sns.set_style('ticks')
```

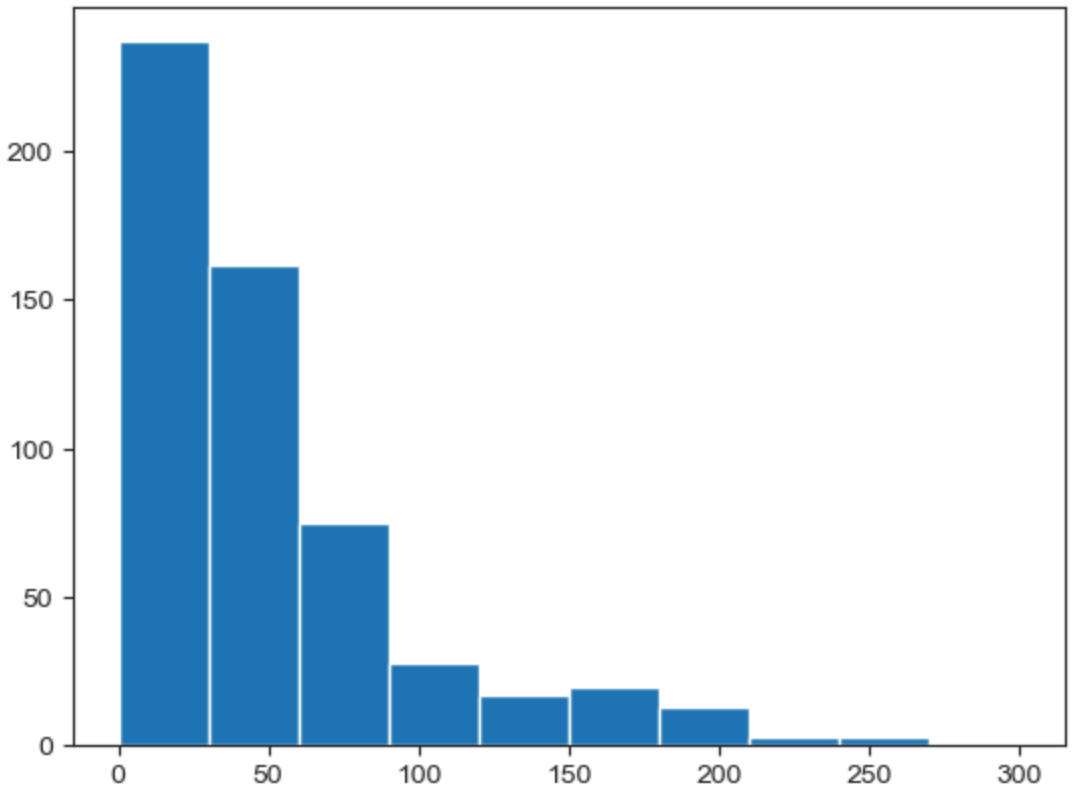
uniform distribution-->

A uniform distribution is a type of probability distribution where all outcomes are equally likely. In simpler terms, every value in a given range has the same chance of occurring

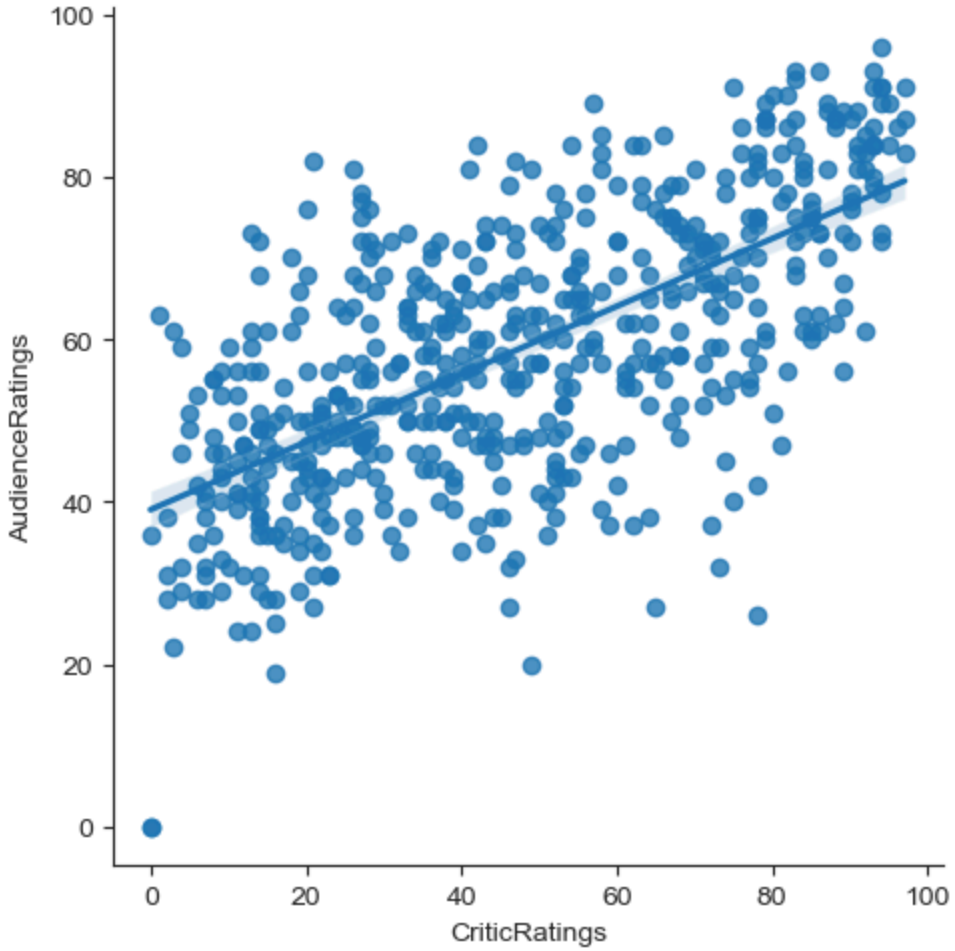
normal distribution--->

normal distribution is a continuous probability distribution that is symmetric around its mean,

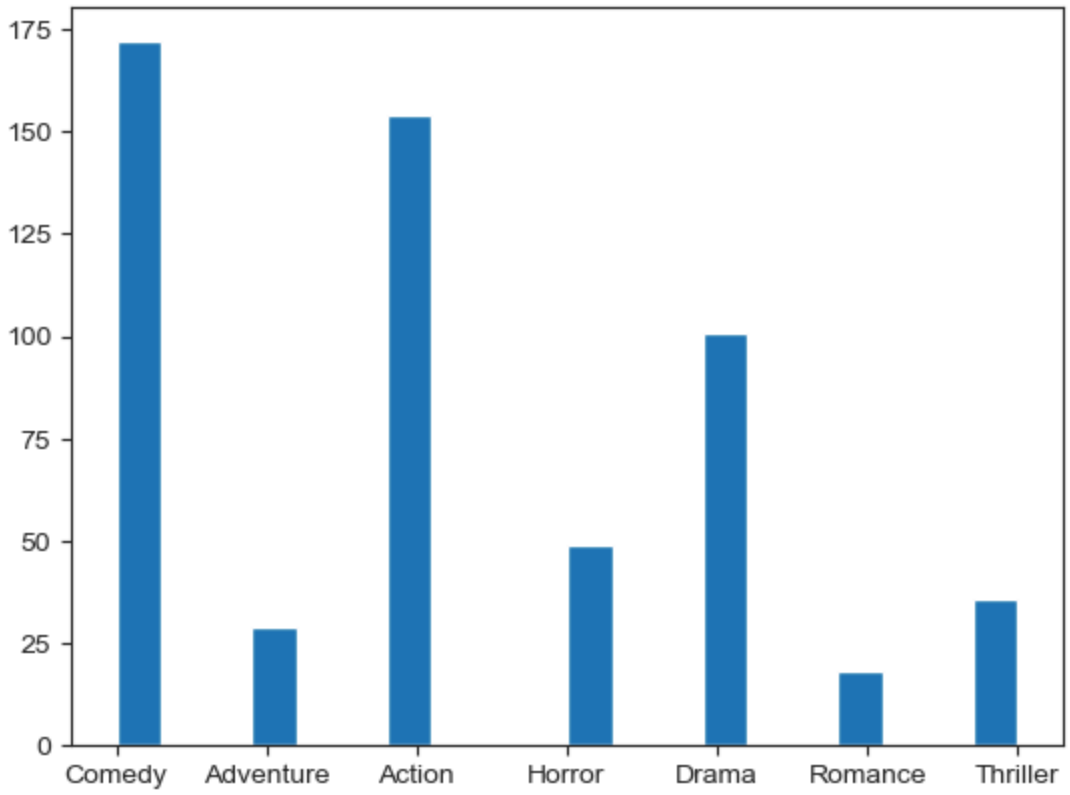
In [30]:
```
 plt.hist(movies.BudgetMillions)
plt.show()
```



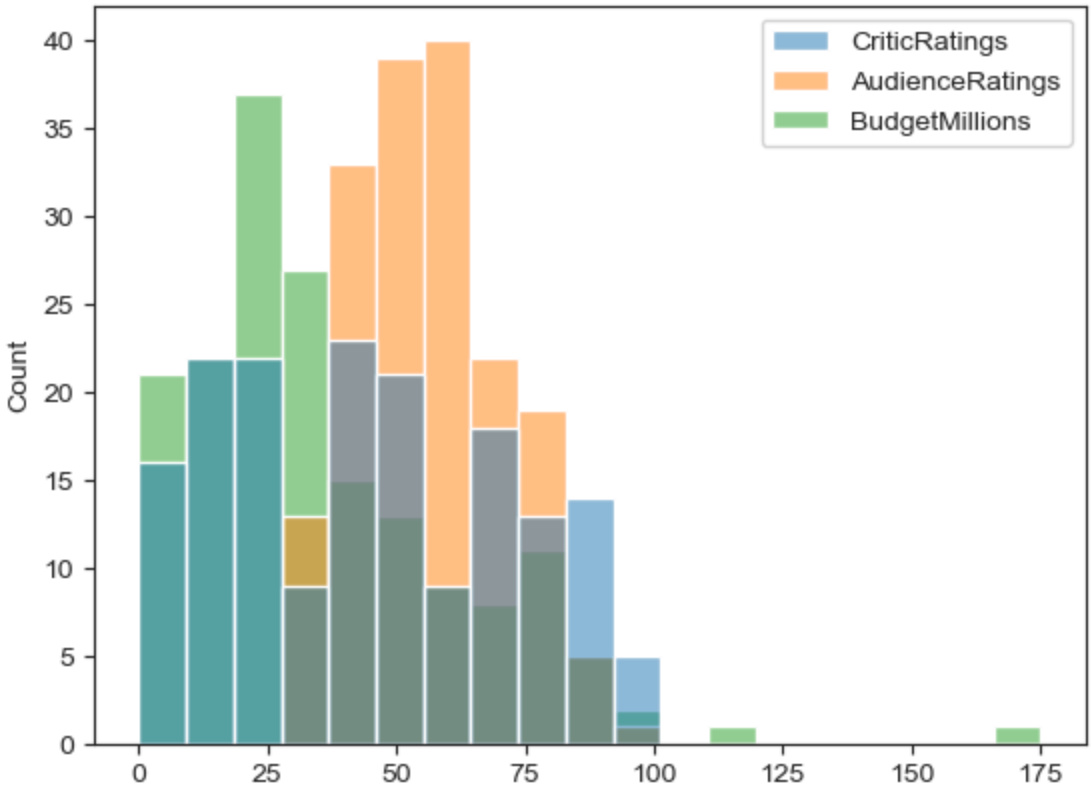
In [31]:
```
vis1=sns.lmplot(data=movies,x='CriticRatings',y='AudienceRatings',fit_reg=True)
vis1
plt.show()
```



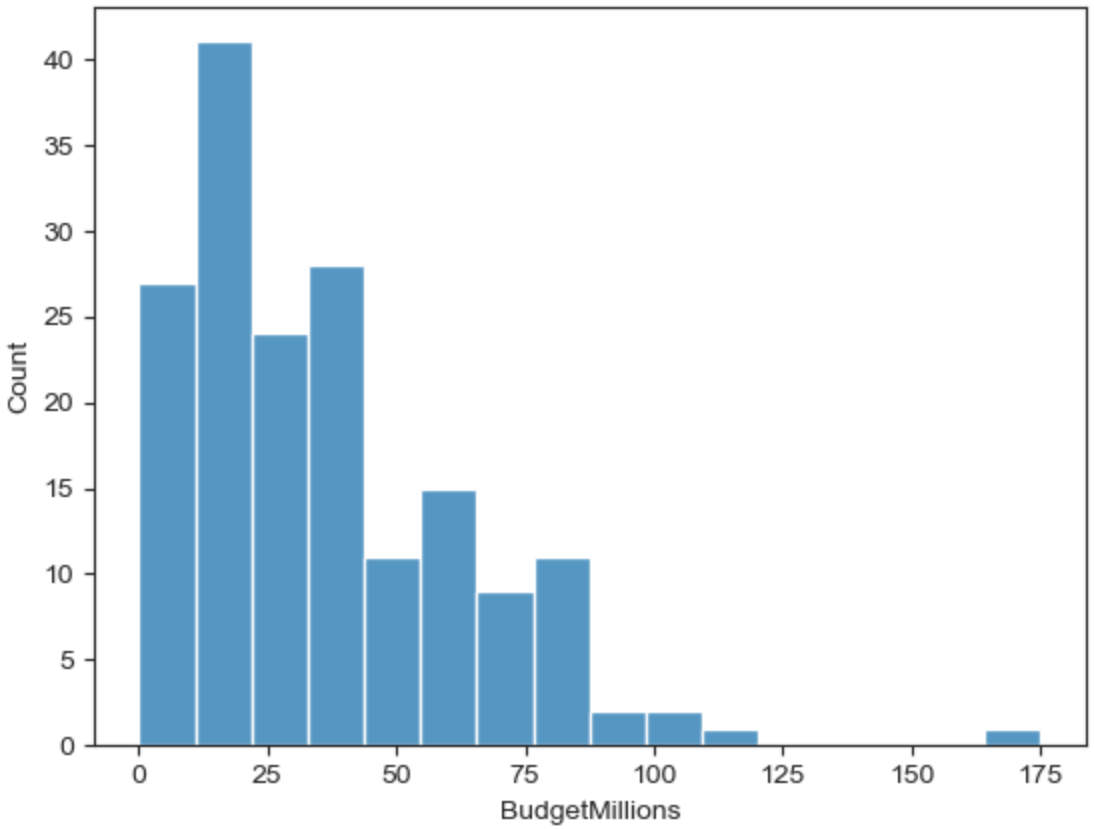
In [32]:
```
plt.hist(movies.Genre,bins=20)
plt.show()
```



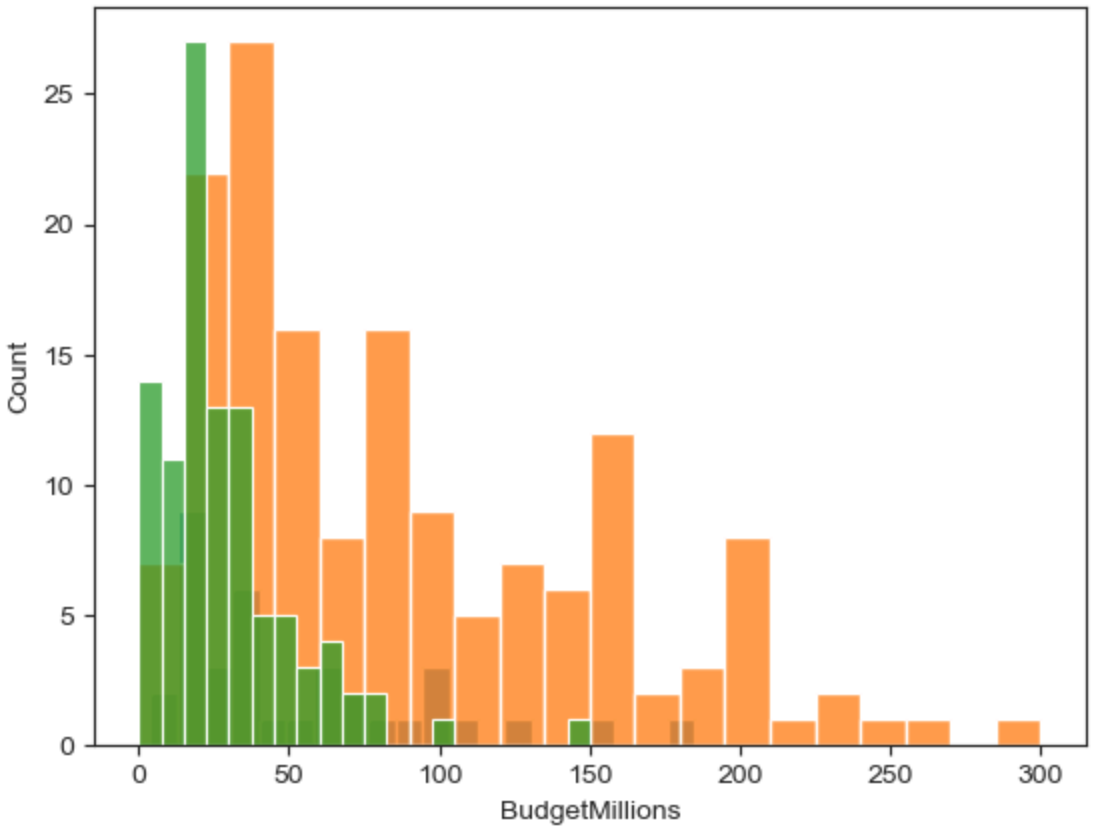
In [33]:
```
sns.histplot(movies[movies.Genre=='Comedy'])
plt.show()
```



In [34]:
```
sns.histplot(movies[movies.Genre=='Comedy'].BudgetMillions)
plt.show()
```

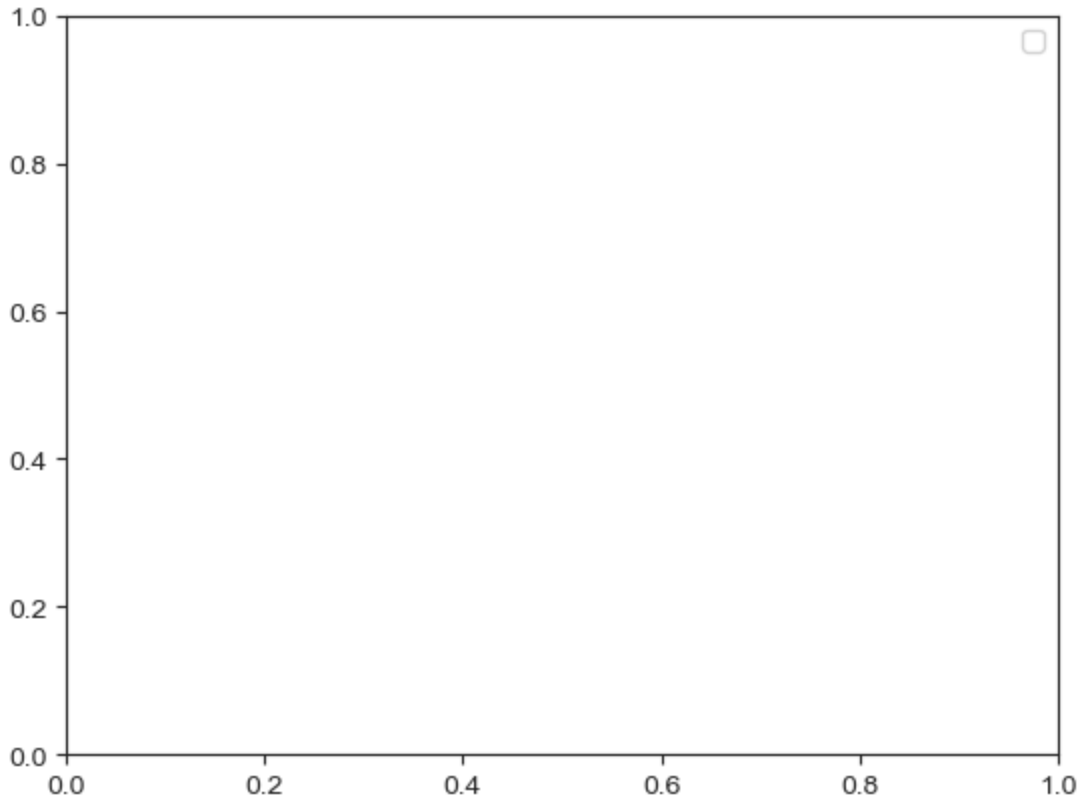


In [35]:
```
sns.histplot(movies[movies.Genre=='Thriller'].BudgetMillions,legend=True,bins=20)
sns.histplot(movies[movies.Genre=='Action'].BudgetMillions,legend=True,bins=20)
sns.histplot(movies[movies.Genre=='Drama'].BudgetMillions,legend=True,bins=20)
plt.show()
plt.legend()
```

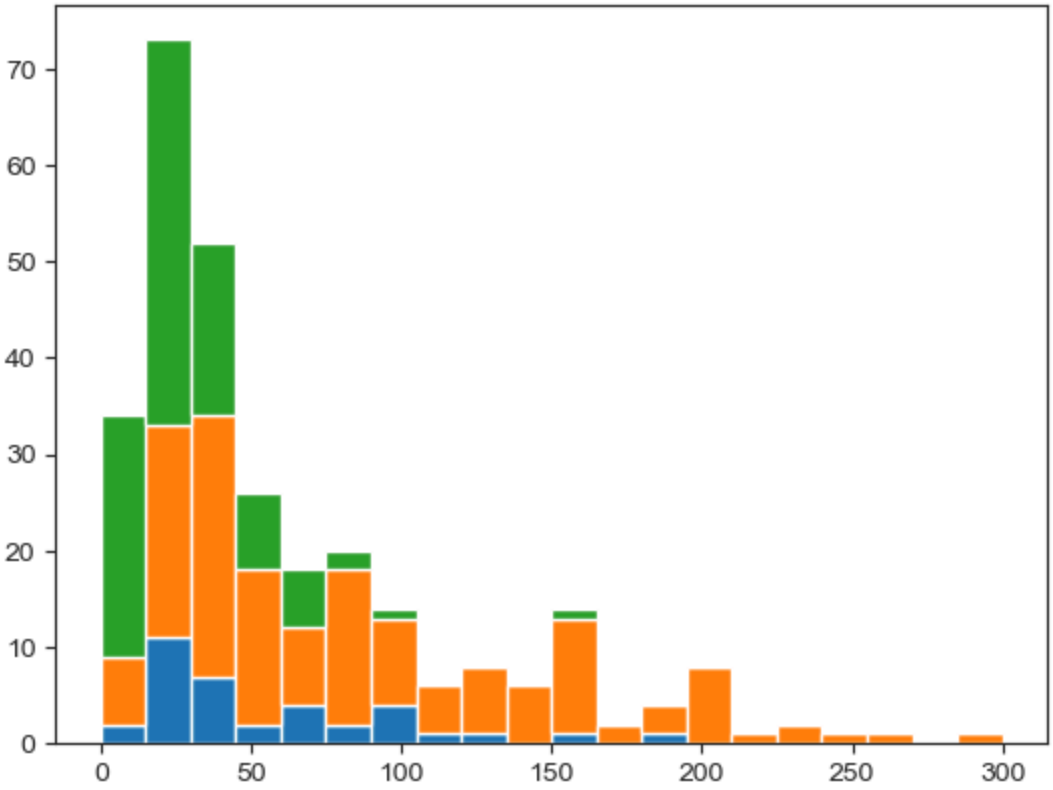



```
C:\Users\nandh\AppData\Local\Temp\ipykernel_14388\710800013.py:5: UserWarning: No ar
tists with labels found to put in legend.  Note that artists whose label start with
an underscore are ignored when legend() is called with no argument.
  plt.legend()
```

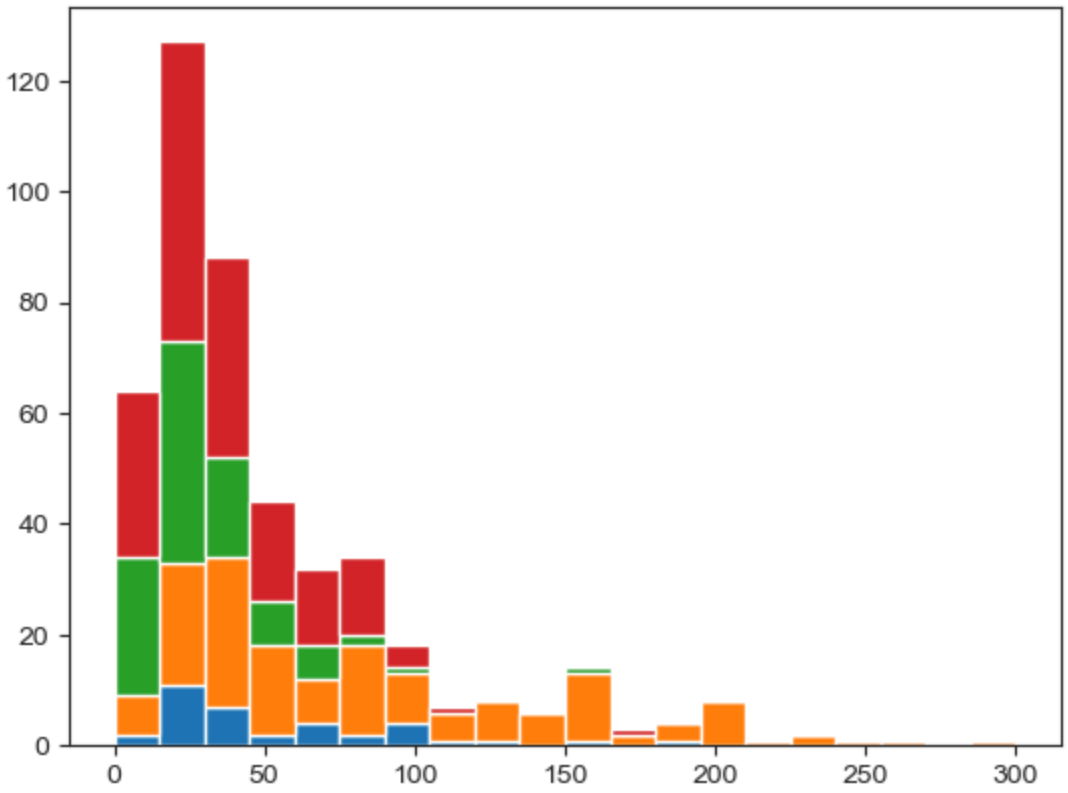
Out[35]: <matplotlib.legend.Legend at 0x17143082e70>



In [36]:
```
plt.hist([movies[movies.Genre=='Thriller'].BudgetMillions,\
          movies[movies.Genre=='Action'].BudgetMillions,\
          movies[movies.Genre=='Drama'].BudgetMillions],
         bins=20,stacked=True)
plt.show()
```



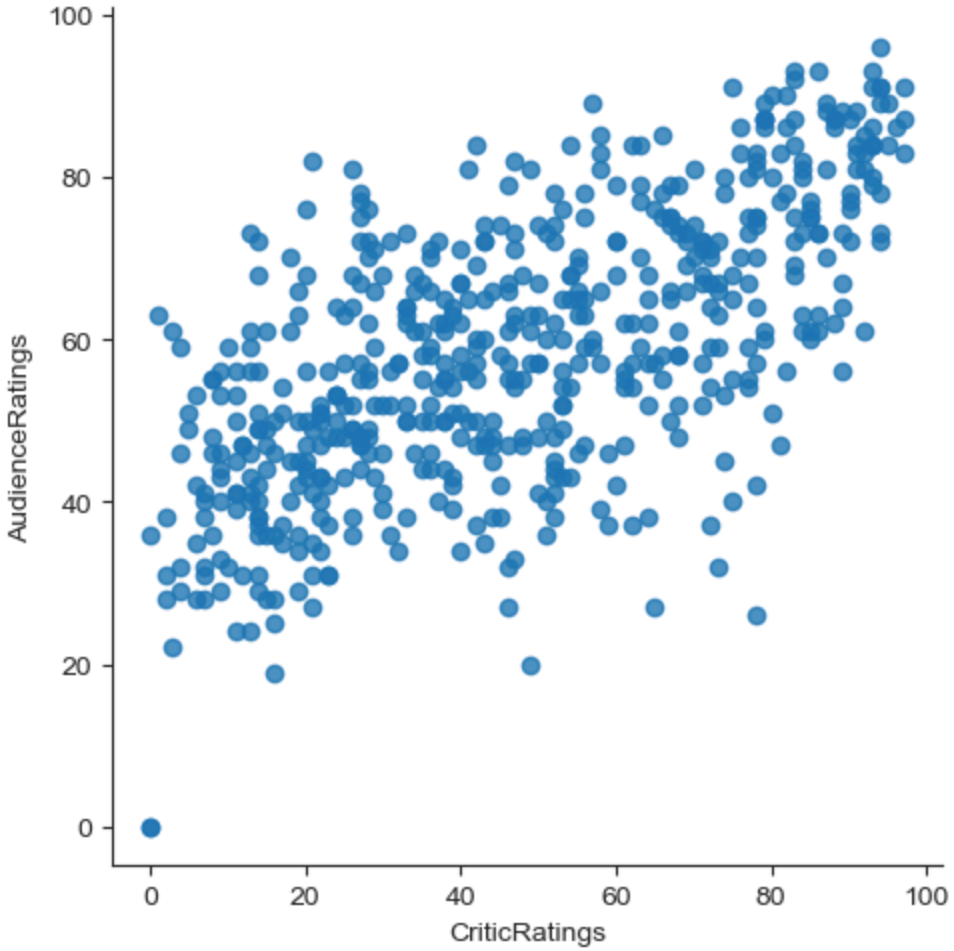
In [37]:
```
plt.hist([movies[movies.Genre=='Thriller'].BudgetMillions,\
          movies[movies.Genre=='Action'].BudgetMillions,\
          movies[movies.Genre=='Drama'].BudgetMillions,\
          movies[movies.Genre=='Comedy'].BudgetMillions],
         bins=20,stacked=True)
plt.show()
```



In [38]:
```
for gen in movies.Genre.cat.categories:
    print(gen)
```

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

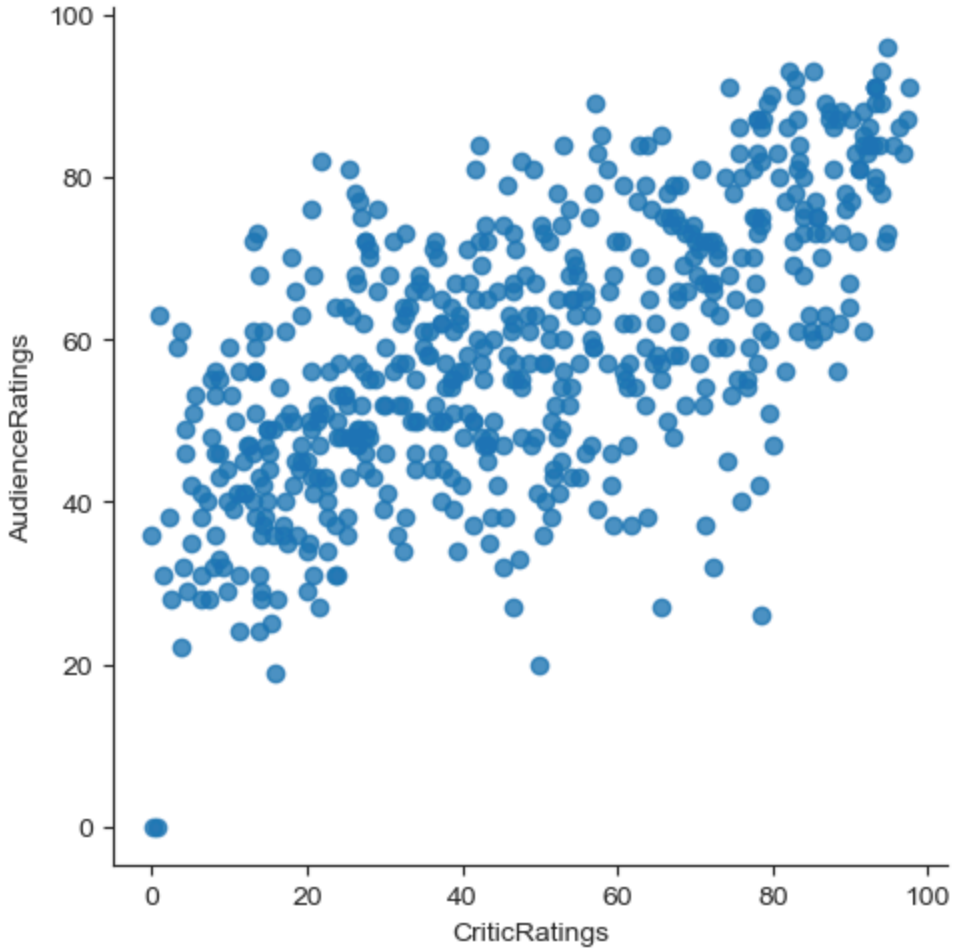
In [39]:
```
vis1 = sns.lmplot(data=movies, x='CriticRatings', y='AudienceRatings', fit_reg=Fals
plt.show()
```



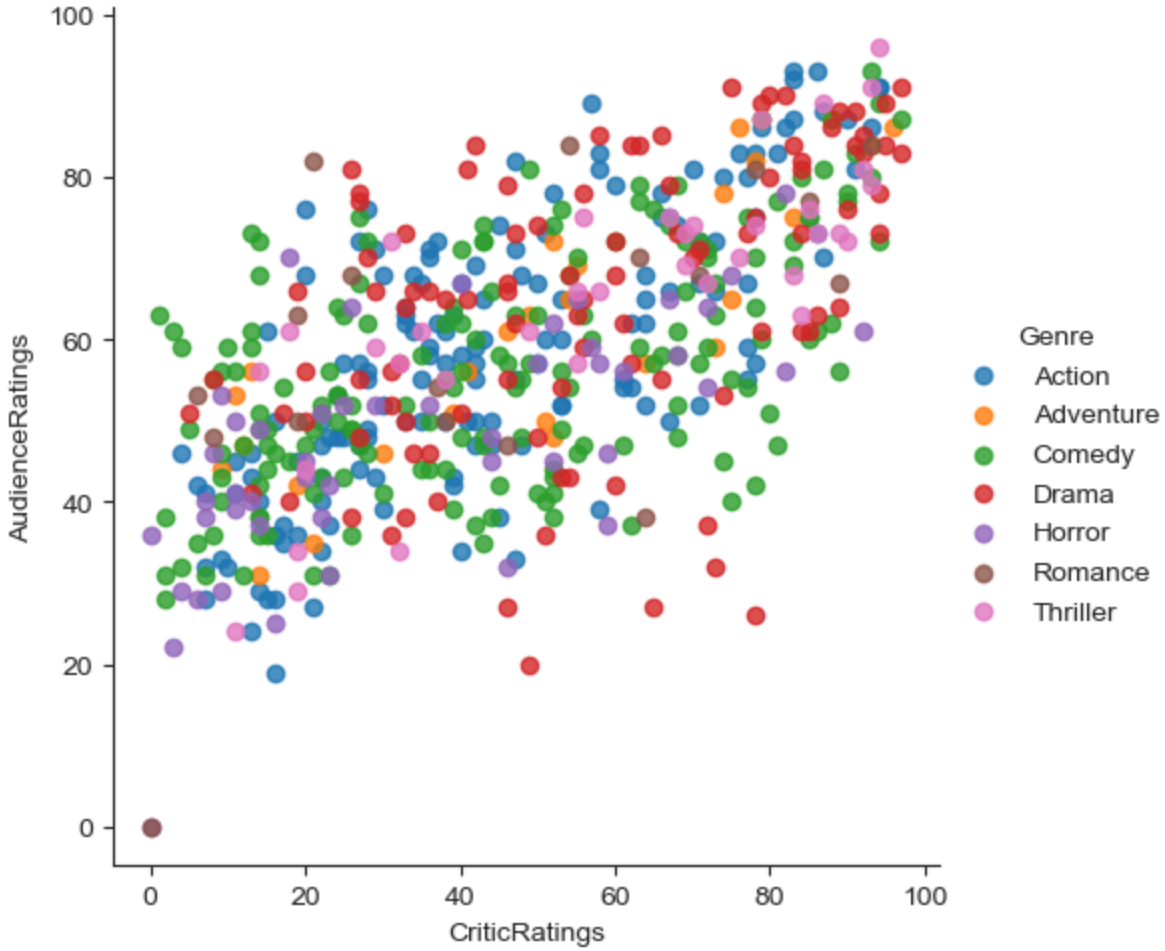
In [40]:
```
movies.columns
```

Out[40]:
```
Index(['Film', 'Genre', 'CriticRatings', 'AudienceRatings', 'BudgetMillions',
       'Year'],
      dtype='object')
```

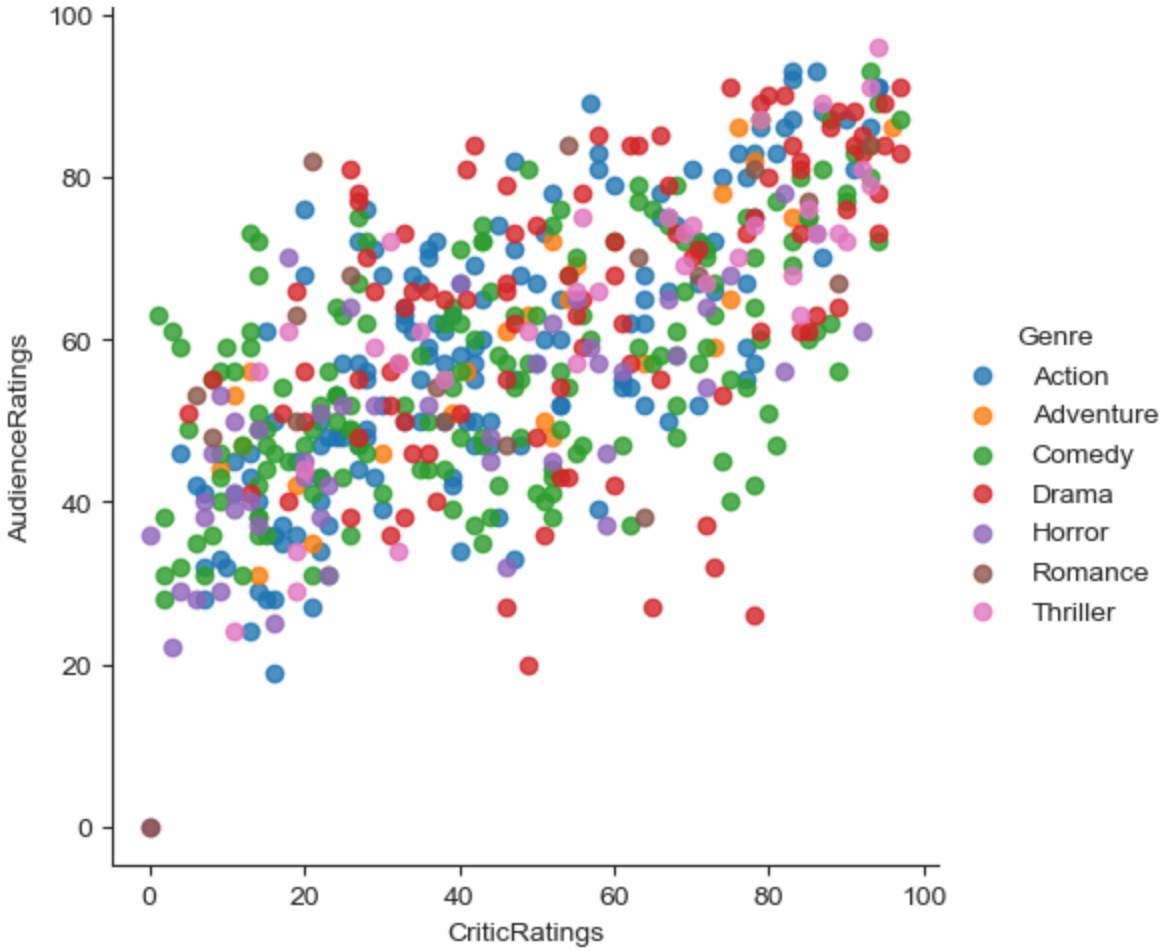
In [41]:
```
vis1 = sns.lmplot(data=movies, x='CriticRatings', y='AudienceRatings', fit_reg=Fals
plt.show()
```



In [42]:
```
vis1 = sns.lmplot(data=movies, x='CriticRatings', y='AudienceRatings', fit_reg=Fals
plt.show()
```



In [43]:
```
vis1 = sns.lmplot(data=movies, x='CriticRatings', y='AudienceRatings', fit_reg=Fals
plt.show()
```



In [44]:
```
sns.set_style('dark')
```

Kernal Density Estimate plot (KDE PLOT)

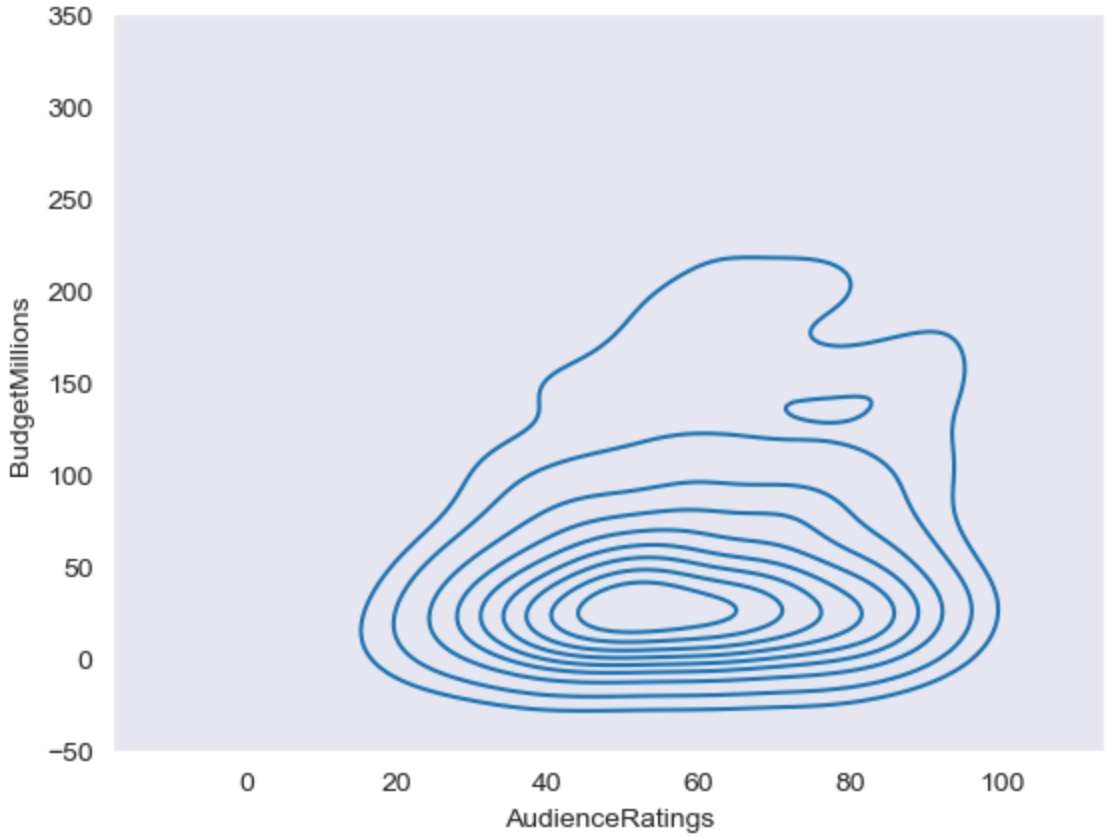
how can i visulize audience rating & critics rating . using scatterplot

In [45]:
```
%matplotlib inline
```

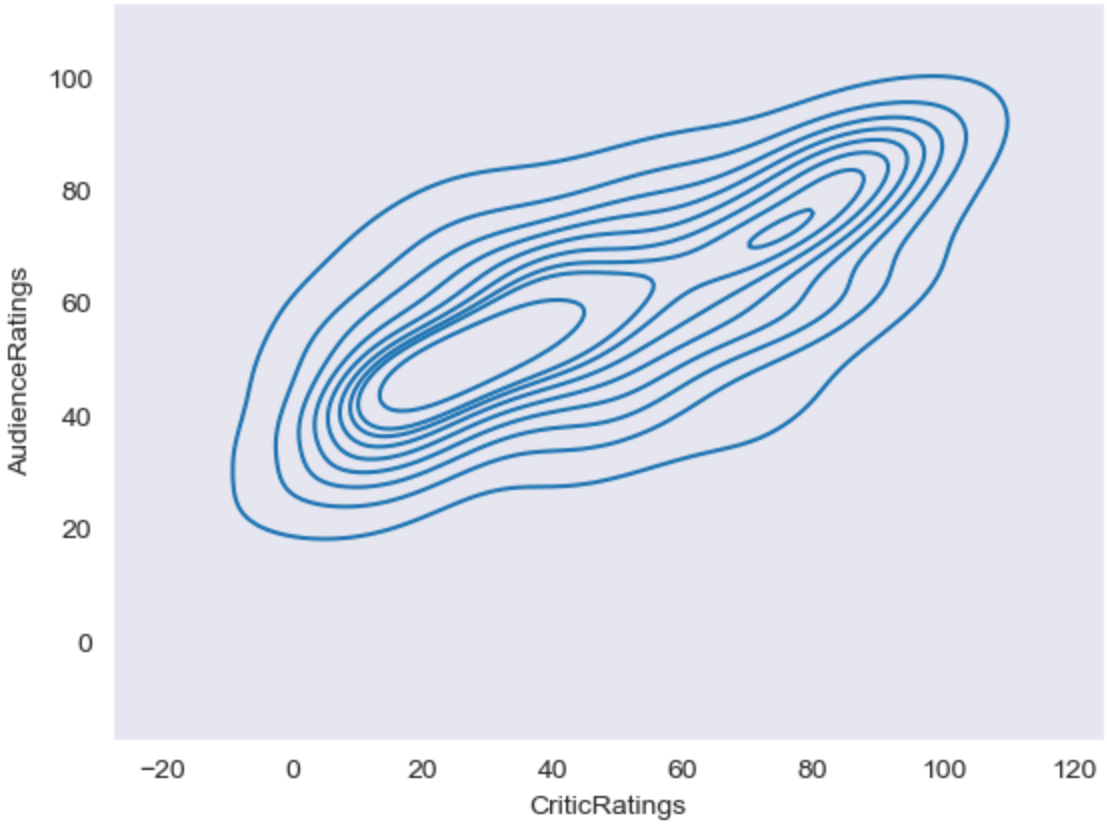
In [46]:
```
movies.columns
```

Out[46]:
```
Index(['Film', 'Genre', 'CriticRatings', 'AudienceRatings', 'BudgetMillions',
       'Year'],
      dtype='object')
```

In [47]:
```
k1=sns.kdeplot(x=movies.AudienceRatings,y=movies.BudgetMillions)
plt.show()
```



In [48]:
```
vis1 = sns.kdeplot(data=movies, x='CriticRatings', y='AudienceRatings')
plt.show()
```



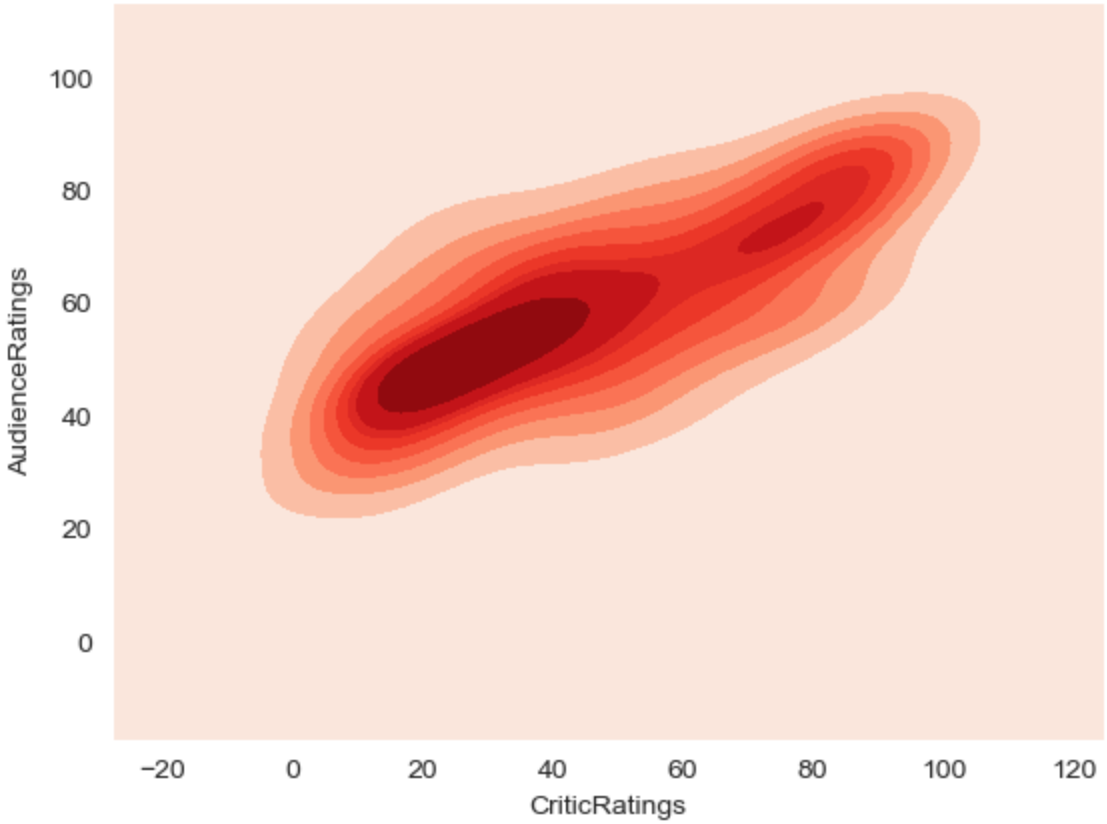
In [49]:
```
movies
```

Out[49]:

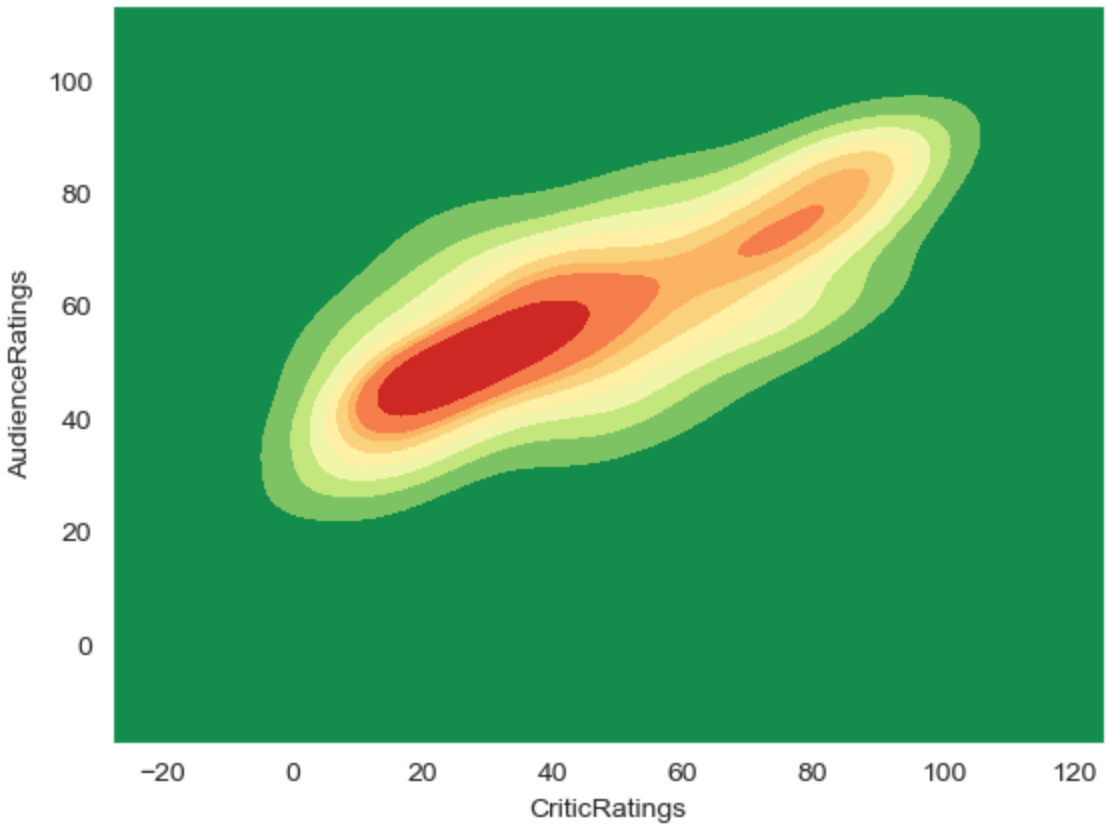
	Film	Genre	CriticRatings	AudienceRatings	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009
...	...	...	...	...	...	...
554	Your Highness	Comedy	26	36	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

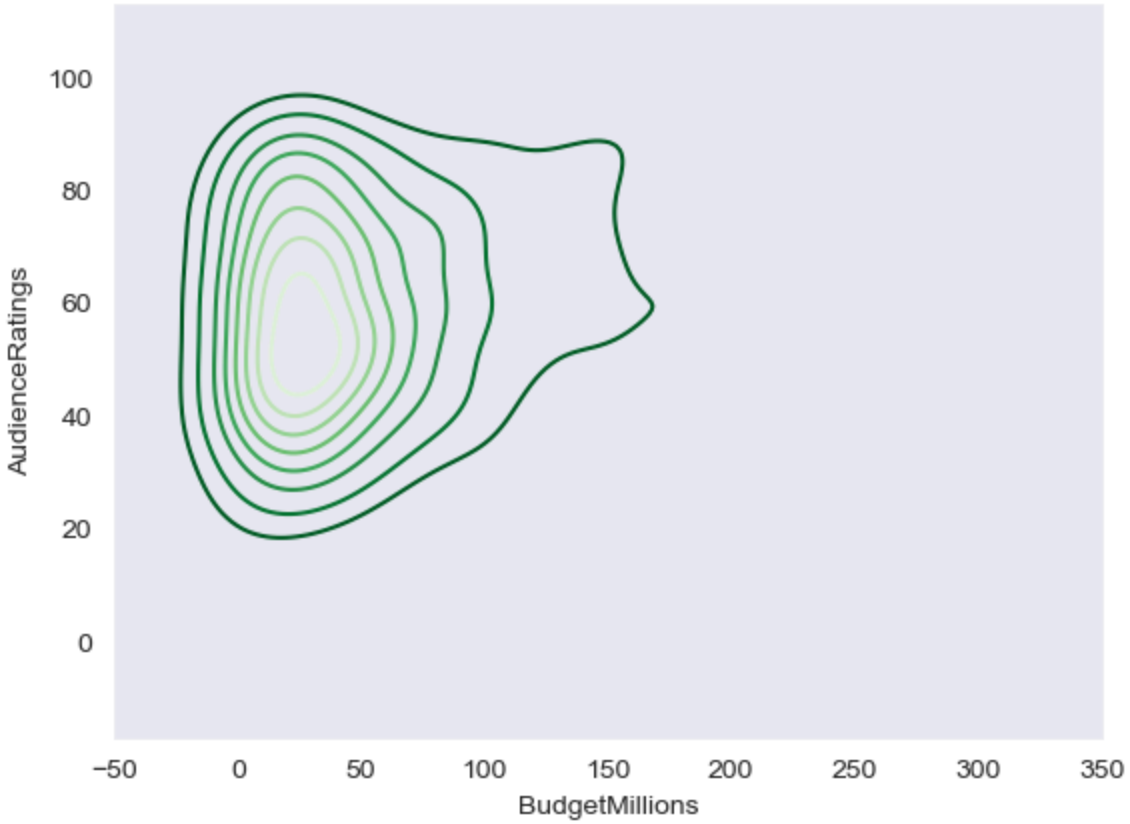
In [50]:
```
k1 = sns.kdeplot(x=movies.CriticRatings,y=movies.AudienceRatings,fill= True,thresh=
plt.show()
```



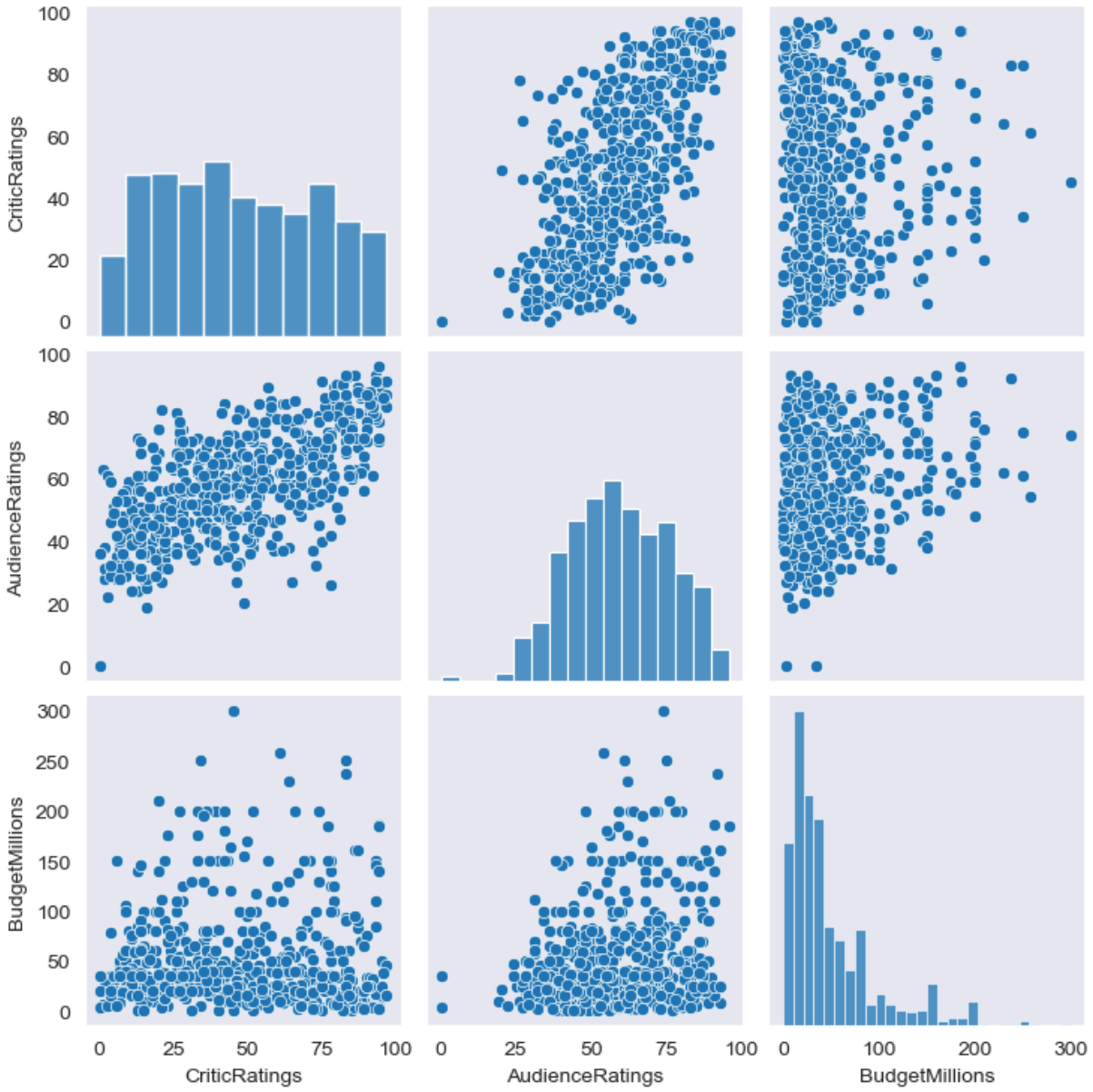
In [51]:
```
k1 = sns.kdeplot(x=movies.CriticRatings,y=movies.AudienceRatings,fill= True,thresh=

#'Accent', 'Accent_r', 'Blues', 'Blues_r', 'BrBG', 'BrBG_r', 'BuGn', 'BuGn_r', 'BuP
#'CMRmap_r',
#'Dark2', 'Dark2_r', 'GnBu', 'GnBu_r', 'Grays', 'Greens',
#'Greens_r', 'Greys', 'Greys_r', 'OrRd', 'OrRd_r', 'Oranges', 'Oranges_r', 'PRGn',
#'Paired_r', 'Pastel1', 'Pastel1_r', 'Pastel2', 'Pastel2_r', 'PiYG', 'PiYG_r', 'PuB
#'PuBu_r', 'PuOr', 'PuOr_r', 'PuRd', 'PuRd_r', 'Purples', 'Purples_r', 'RdBu', 'RdB
#'RdPu', 'RdPu_r', 'RdYlBu', 'RdYlBu_r', 'RdYlGn', 'RdYlGn_r', 'Reds', 'Reds_r', 'S
plt.show()
```



In [52]:
```
k1 = sns.kdeplot(x=movies.BudgetMillions,y=movies.AudienceRatings,thresh=False,cmap
plt.show()
```

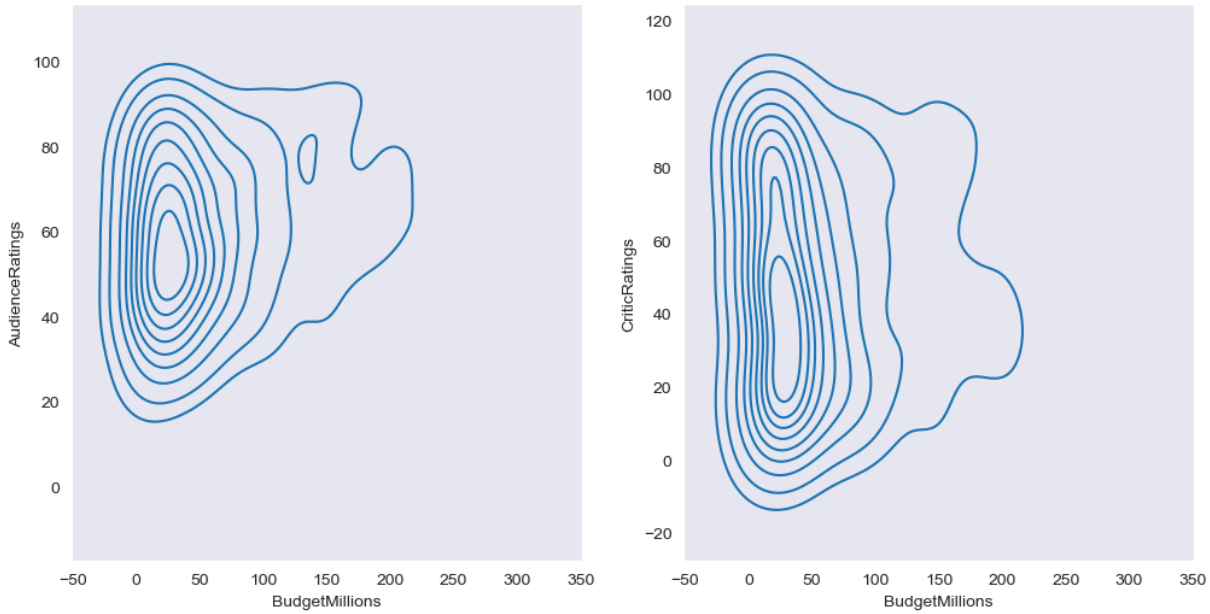


In [53]:
```
sns.pairplot(data=movies,)
plt.show()
```

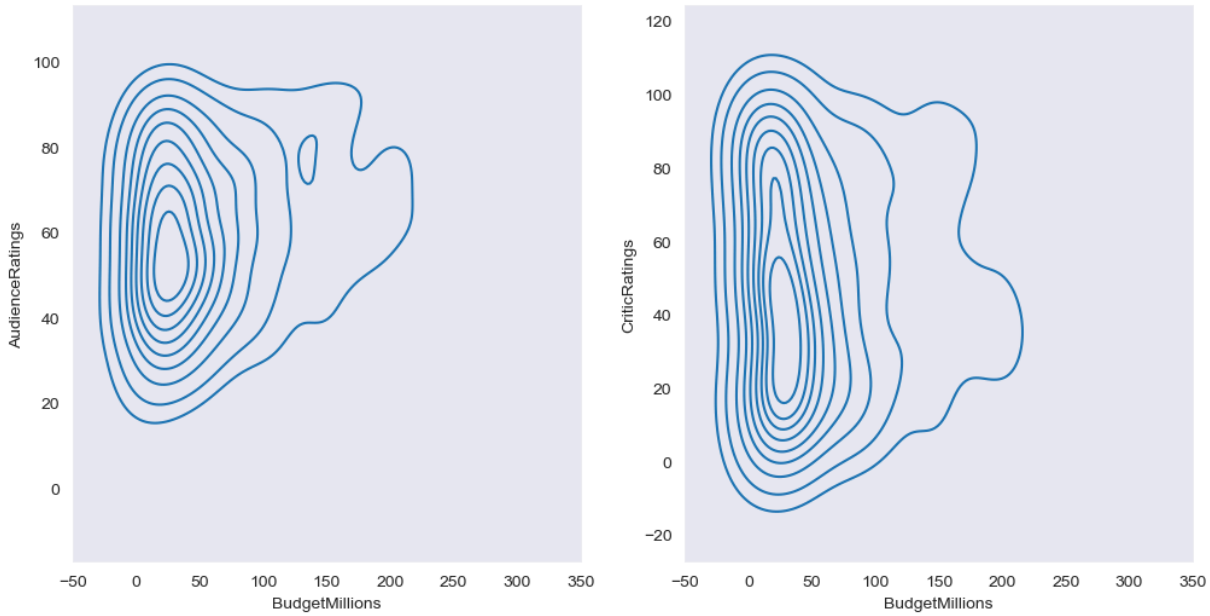


In [100…
```
movies.columns
%matplotlib inline
```

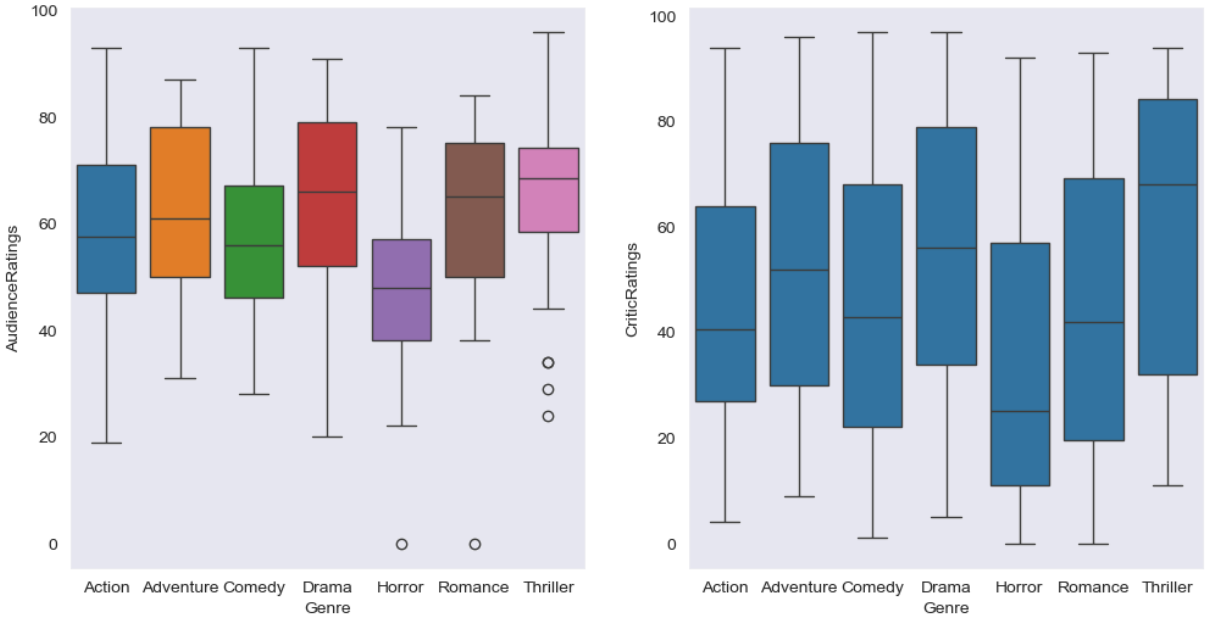
In [134…
```
f, ax = plt.subplots(1,2, figsize =(12,6))
k1=sns.kdeplot(data=movies,x= 'BudgetMillions', y='AudienceRatings',ax=ax[0])
k2=sns.kdeplot(data=movies,x='BudgetMillions',y='CriticRatings',ax=ax[1])
plt.show()
```



In [113…
```
f, axes = plt.subplots(1,2, figsize =(12,6))

k1 = sns.kdeplot(data=movies,x='BudgetMillions',y='AudienceRatings',ax=axes[0])
k2 = sns.kdeplot(data=movies,x='BudgetMillions',y='CriticRatings',ax=axes[1])
plt.show()
```



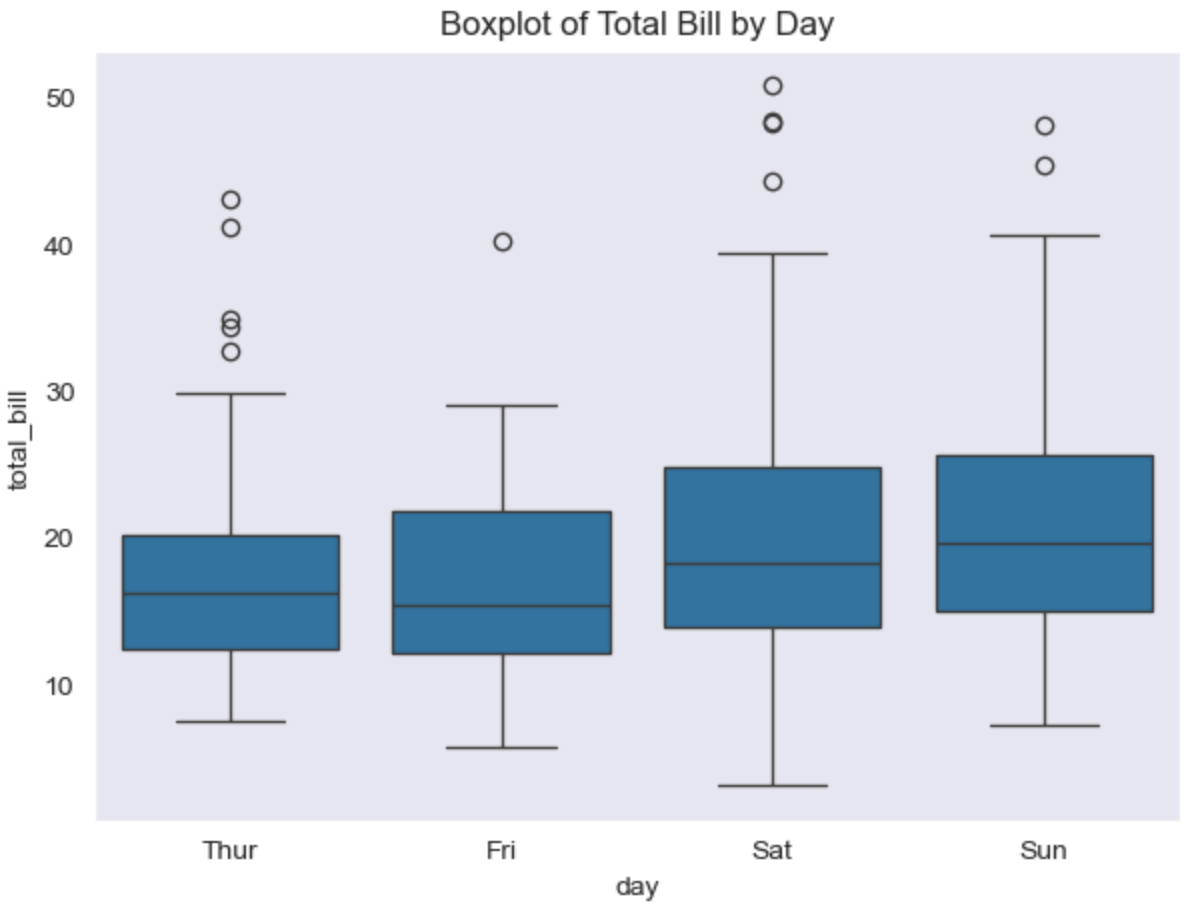
In [130…
```
f, axes = plt.subplots(1,2, figsize =(12,6))
sns.boxplot(data=movies,x='Genre',y='AudienceRatings',ax=axes[0],legend='auto',hue=
sns.boxplot(data=movies,x='Genre',y='CriticRatings',ax=axes[1])
plt.show()
```



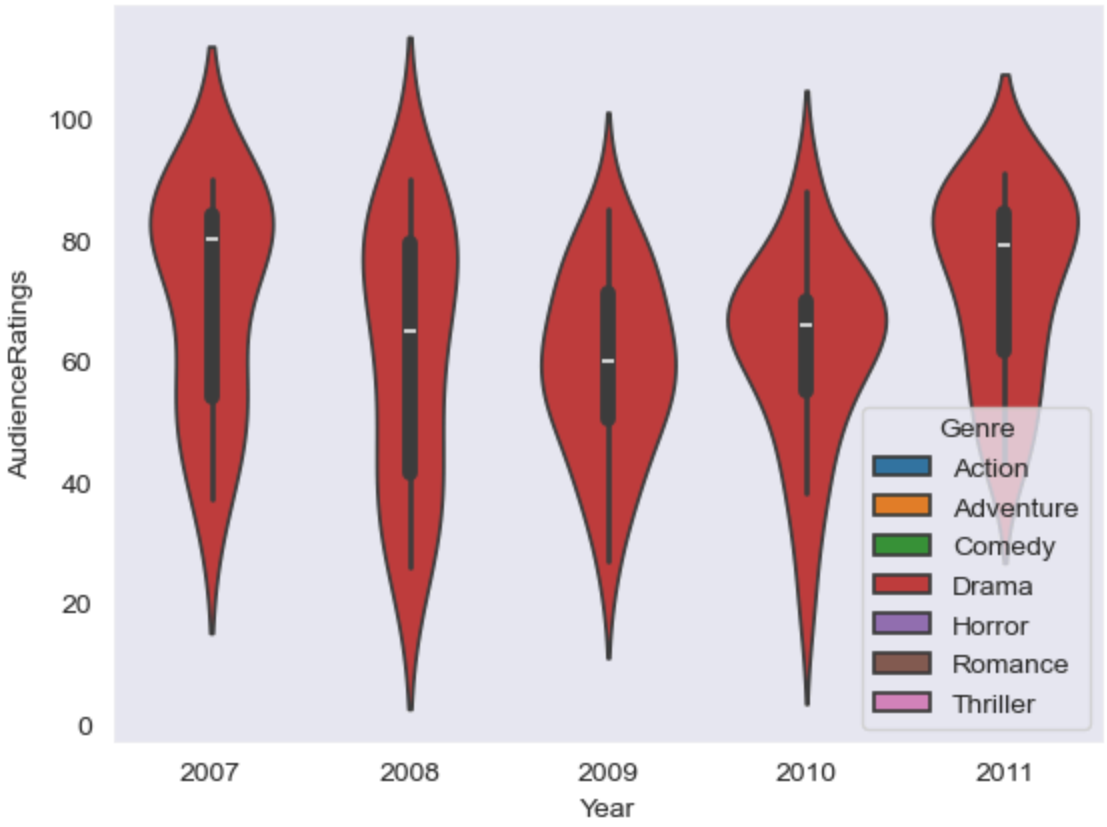
In [145…

```python
# Load sample dataset
tips = sns.load_dataset("tips")

# Create a boxplot
plt.figure(figsize=(7, 5))
ax = sns.boxplot(x="day", y="total_bill", data=tips,hue=None)

# Get label names
x_labels = [label.get_text() for label in ax.get_xticklabels()]
y_labels = [label.get_text() for label in ax.get_yticklabels()]

# Print label names
print("X-axis labels:", x_labels)
print("Y-axis labels:", y_labels)

# Show the plot
plt.title("Boxplot of Total Bill by Day")
plt.show()
```

```
X-axis labels: ['Thur', 'Fri', 'Sat', 'Sun']
Y-axis labels: ['0', '10', '20', '30', '40', '50', '60']
```

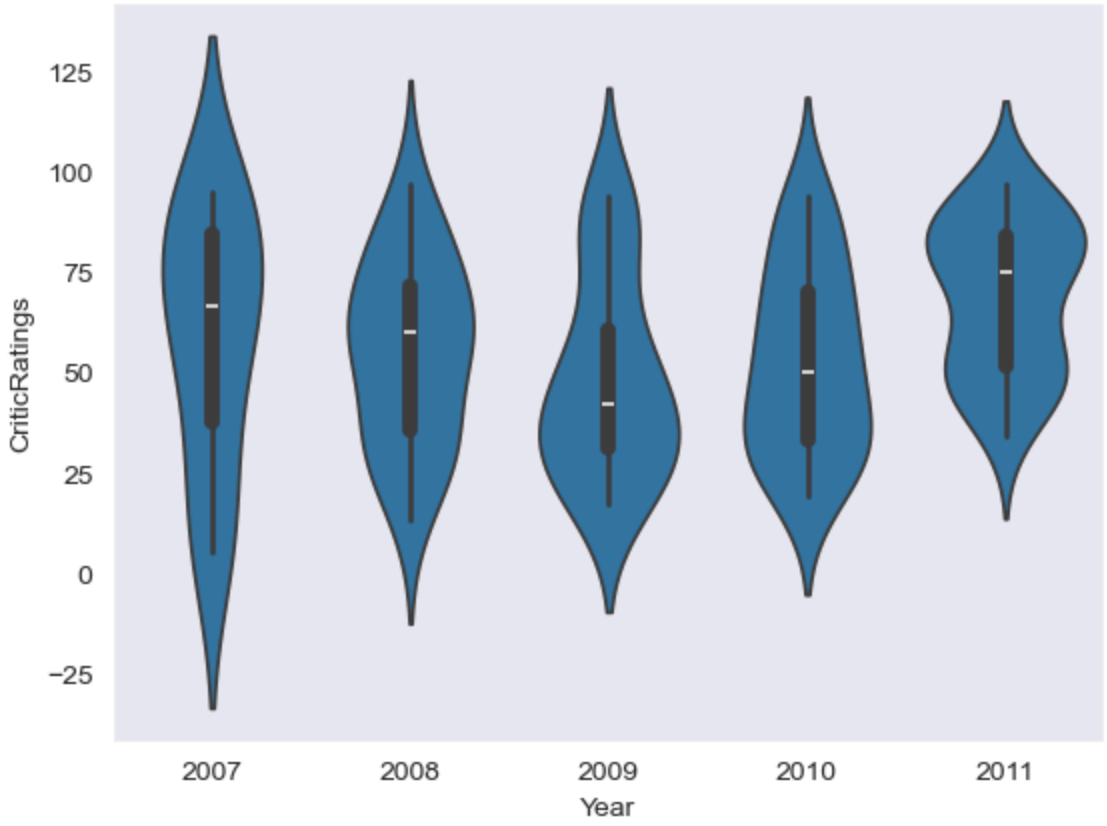


In [161…
```
k1=sns.violinplot(data=movies[movies.Genre == 'Drama'],x='Year',y='AudienceRatings'
plt.show()
```

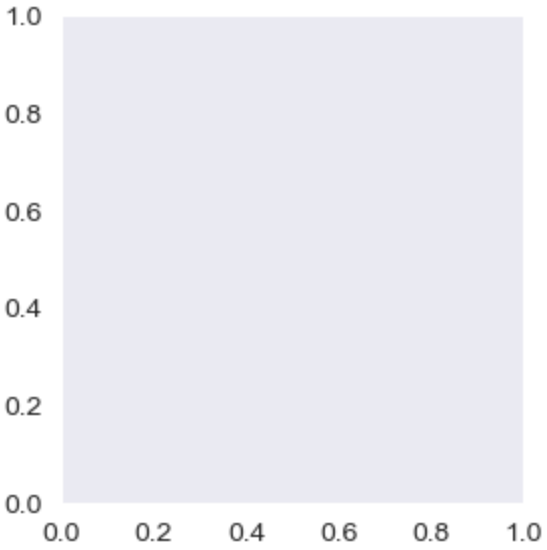


In [158…

```
z = sns.violinplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRatin
plt.show()
```



In [165…
```
sns.FacetGrid(data=movies)
plt.show()
```



In [182…
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map ) #scatterplots are mapped in facetgrid
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[182], line 2
      1 g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
----> 2 g = g.map(plt.scatter (data=movies,x='CriticRating', y='AudienceRating') )

File ~\anaconda3\Lib\site-packages\seaborn\axisgrid.py:758, in FacetGrid.map(self, f
unc, *args, **kwargs)
    755         plot_args = [v.values for v in plot_args]
    757     # Draw the plot
--> 758     self._facet_plot(func, ax, plot_args, kwargs)
    760 # Finalize the annotations and layout
    761 self._finalize_grid(args[:2])

File ~\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854, in FacetGrid._facet_plot
(self, func, ax, plot_args, plot_kwargs)
    852     plot_args = []
    853     plot_kwargs["ax"] = ax
--> 854 func(*plot_args, **plot_kwargs)
    856 # Sort out the supporting information
    857 self._update_legend_data(ax)

TypeError: 'PathCollection' object is not callable
```

In [188…
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.hist, 'BudgetMillions') #scatterplots are mapped in facetgrid
plt.show()
```

In [187…

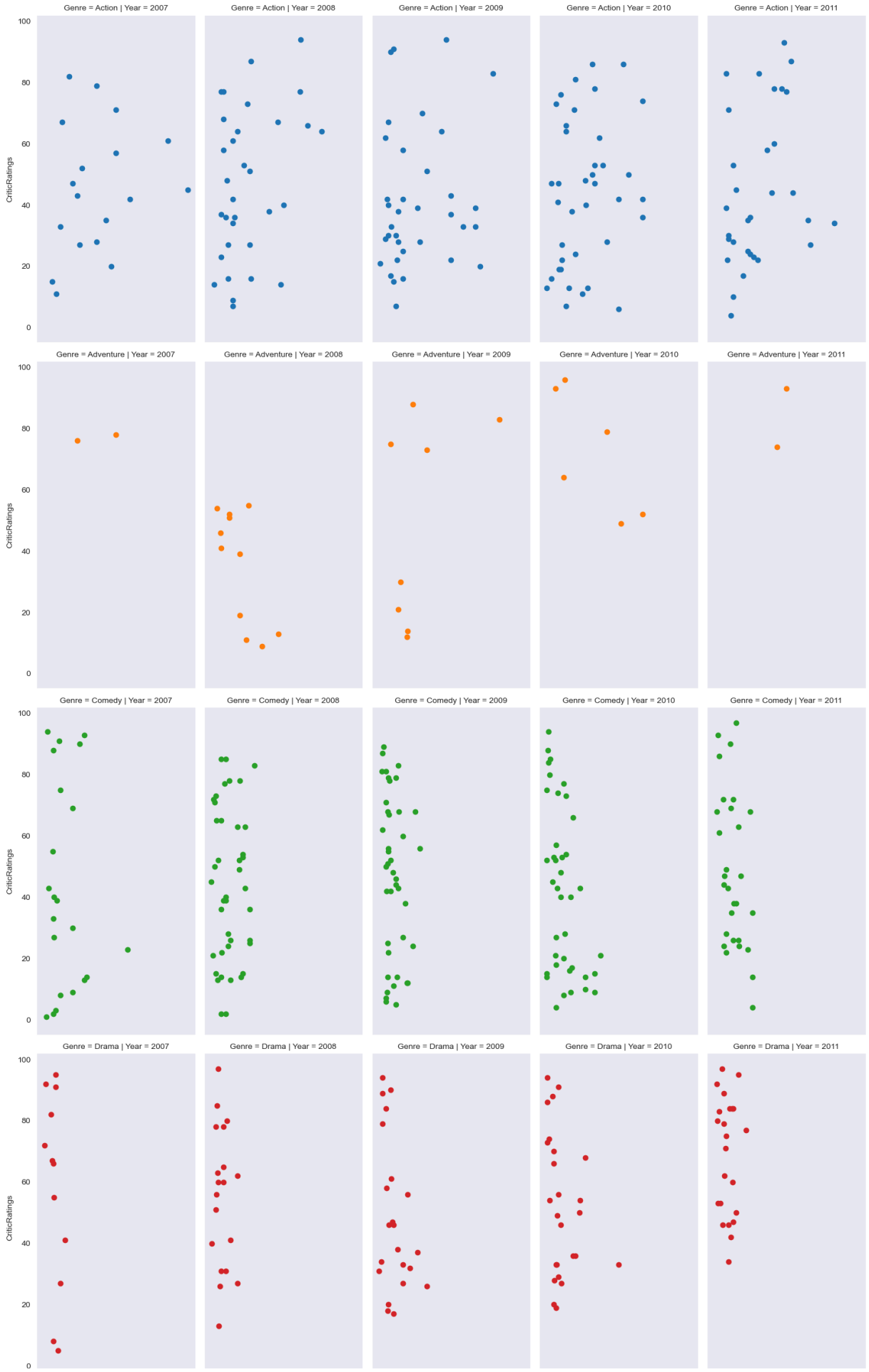
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.hist, 'BudgetMillions') #scatterplots are mapped in facetgrid
plt.show()
```

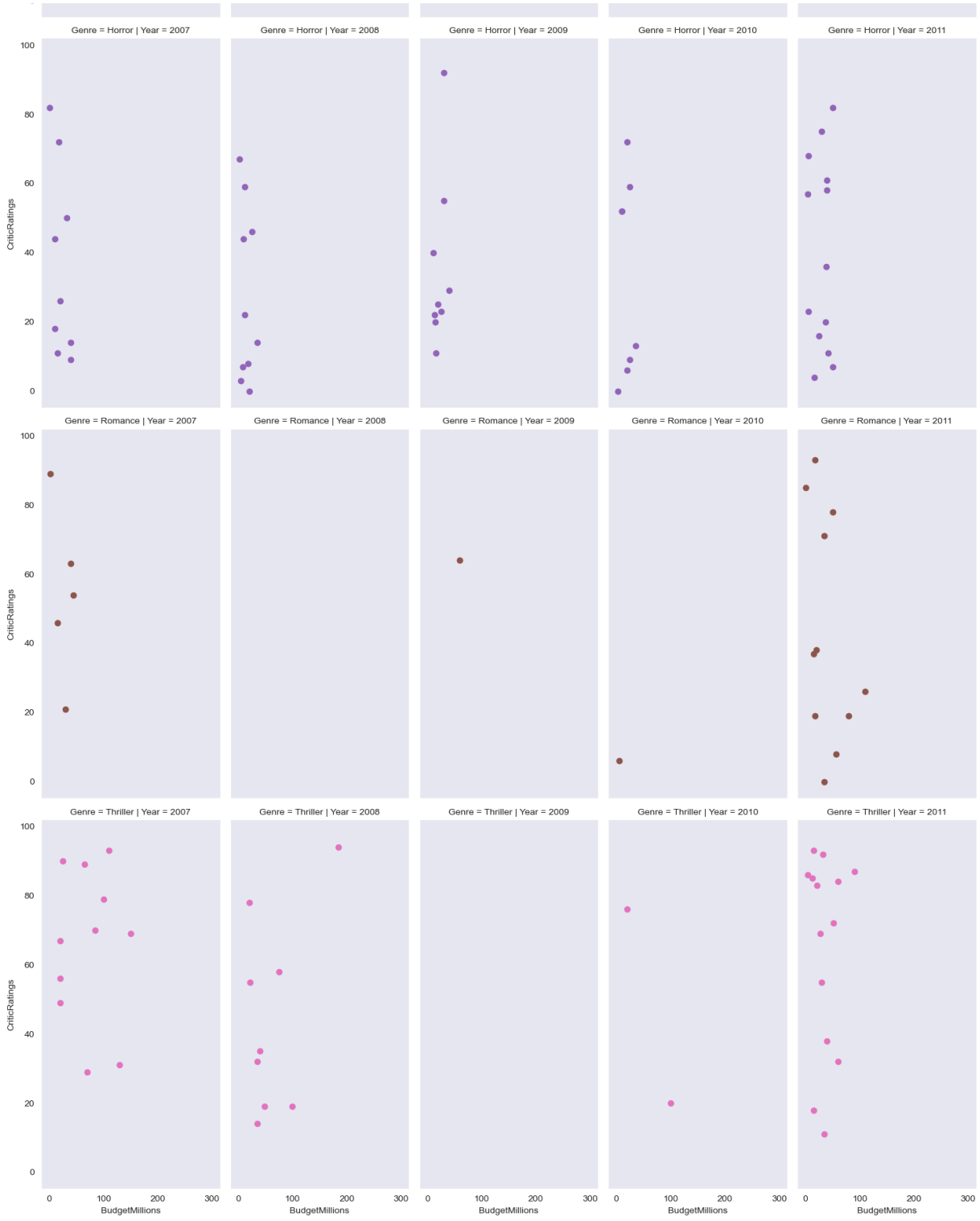

Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011
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 2 4 6 8 10 12 14 16
0 100 200 300
BudgetMillions


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 2 4 6 8 10 12 14 16
0 100 200 300
BudgetMillions

In [201…
```
g=sns.FacetGrid(movies,row='Genre',col='Year',height=6,aspect=0.5,hue='Genre')
```

In [202…
```
g=g.map(plt.scatter,'BudgetMillions','CriticRatings')
plt.show()
```

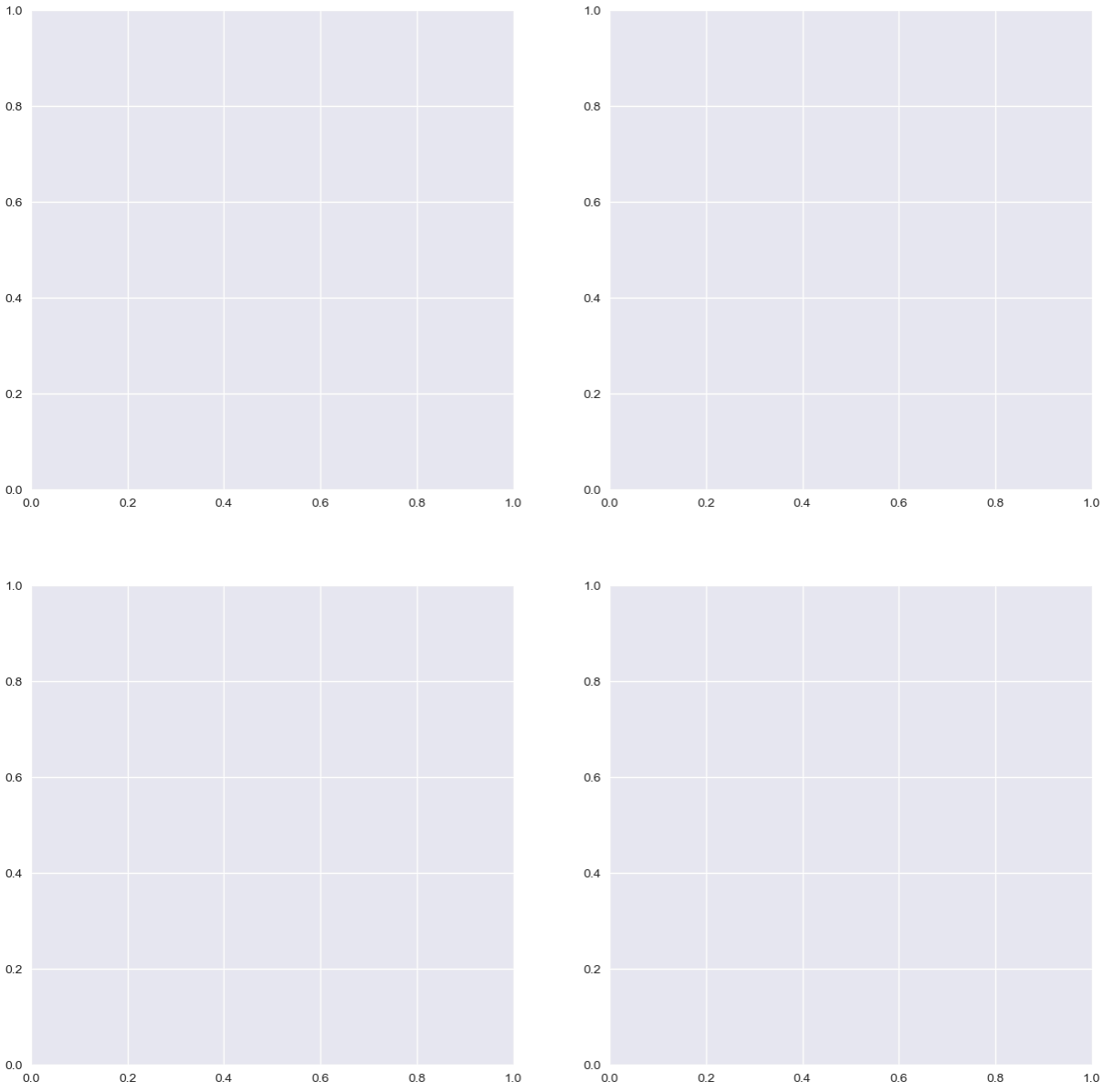

Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011
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
CriticRatings
0
20
40
60
80
100

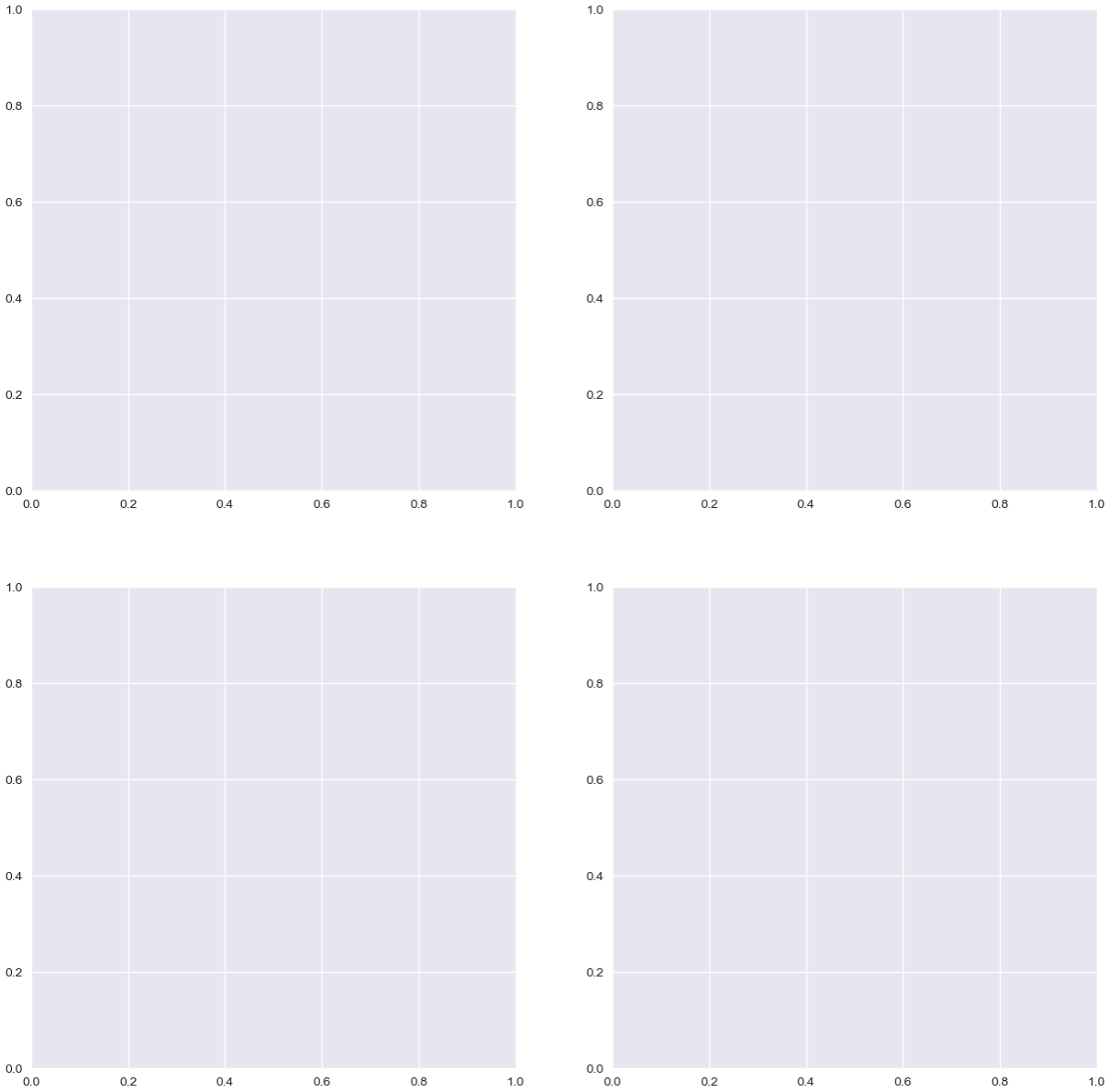


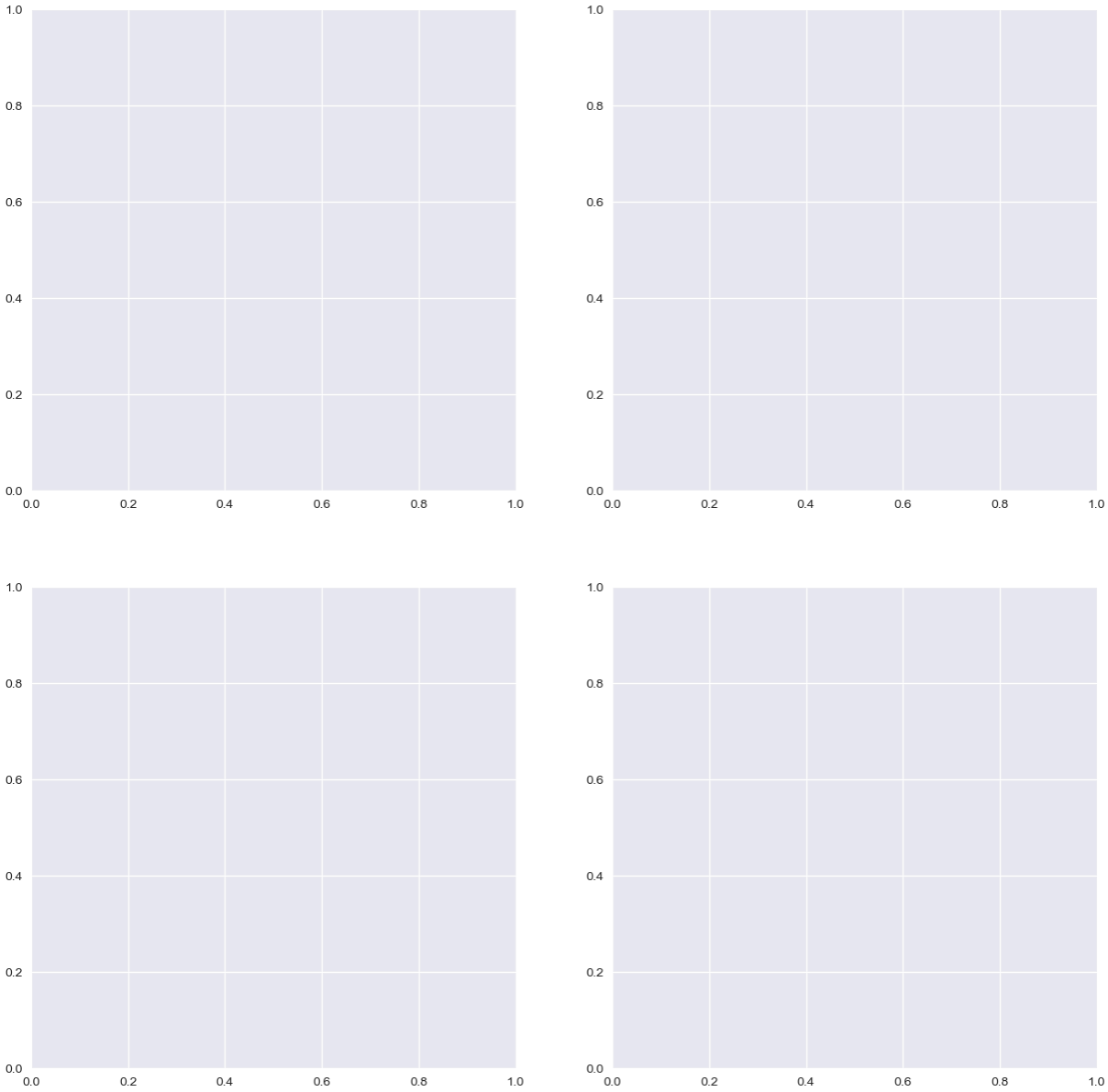
In [215…

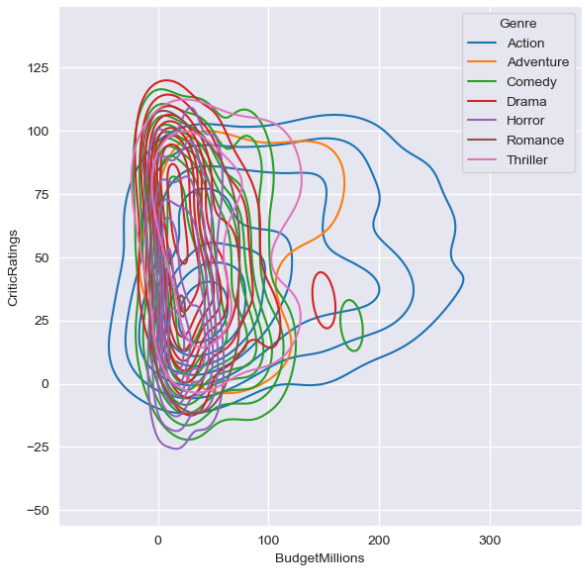
```
sns.set_style('darkgrid')
f, axes = plt.subplots (2,2, figsize = (15,15))
k1=sns.kdeplot(data=movies,x='BudgetMillions',y='CriticRatings',hue='Genre',ax=axes
k2=sns.histplot(data=movies,x='BudgetMillions',y='AudienceRatings',bins=10,ax=axes[
k3=sns.kdeplot(data=movies,x='CriticRatings',y='AudienceRatings',shade=True,cmap='R
k4=sns.violinplot(data=movies[movies.Genre=='Drama'],x='Year',y='CriticRatings',ax=
plt.show()
```

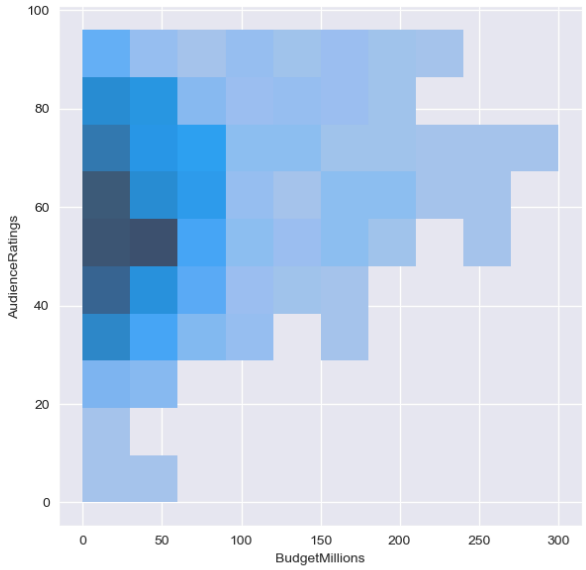
```
C:\Users\nandh\AppData\Local\Temp\ipykernel_14388\3251838508.py:5: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  k3=sns.kdeplot(data=movies,x='CriticRatings',y='AudienceRatings',shade=True,cmap
='Reds',ax=axes[1,0])
```

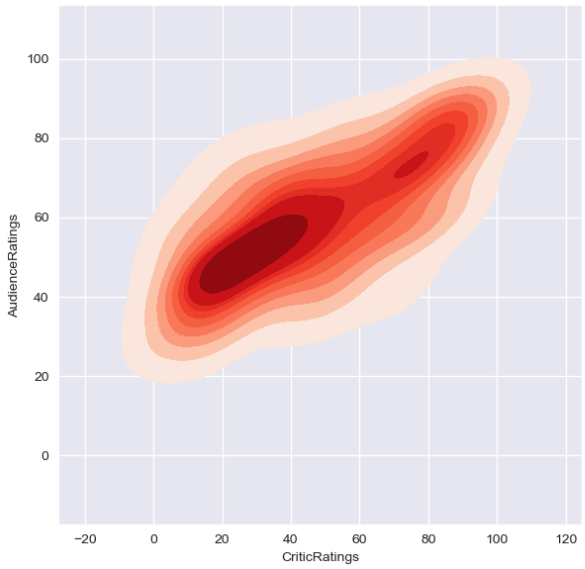


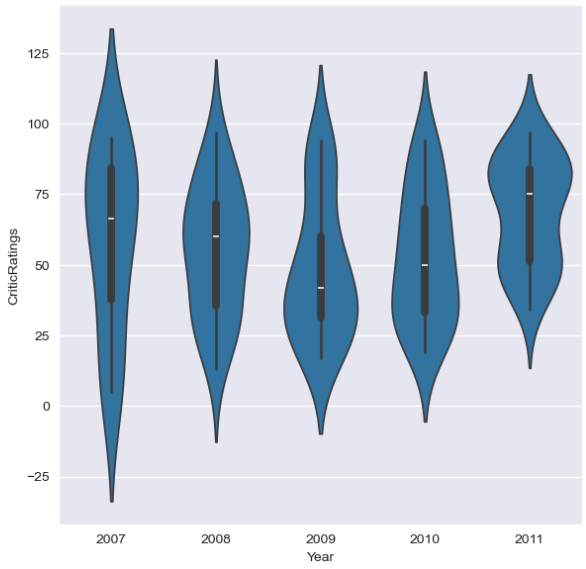

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


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



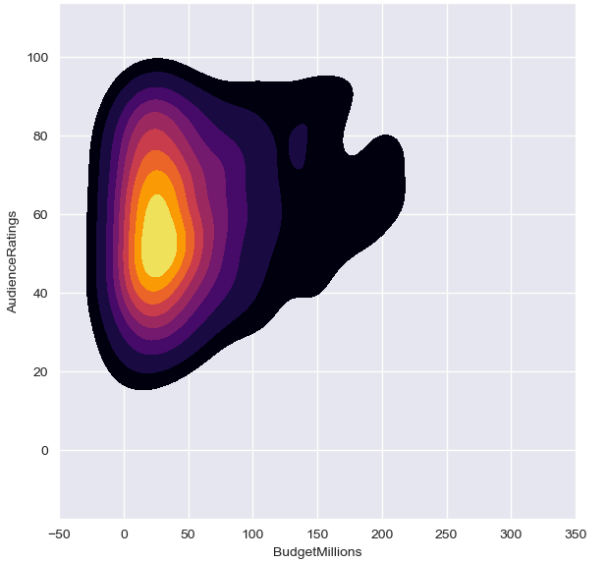


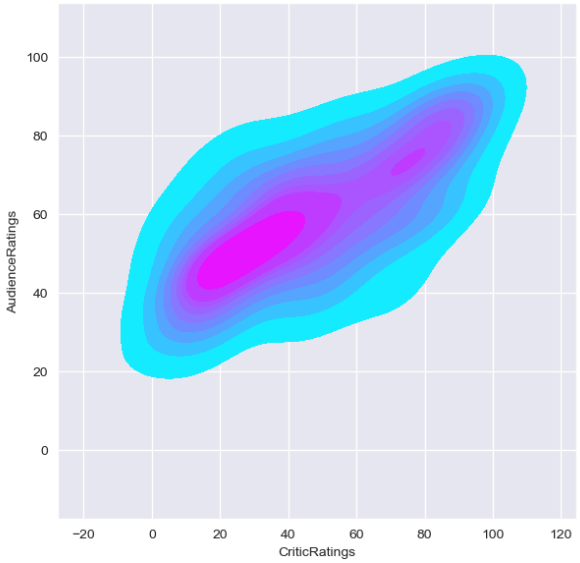


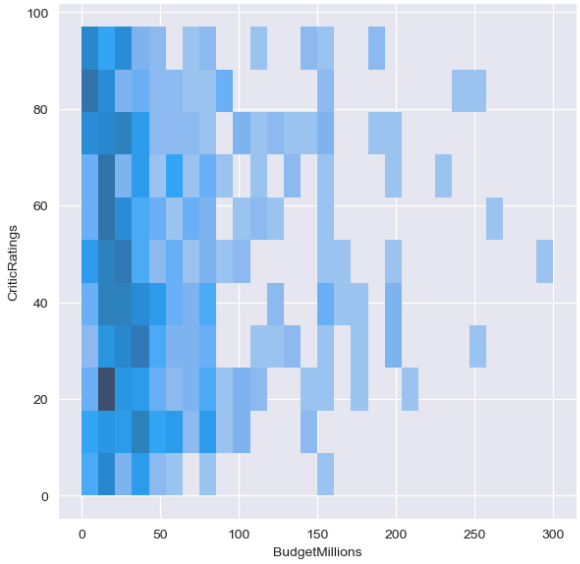


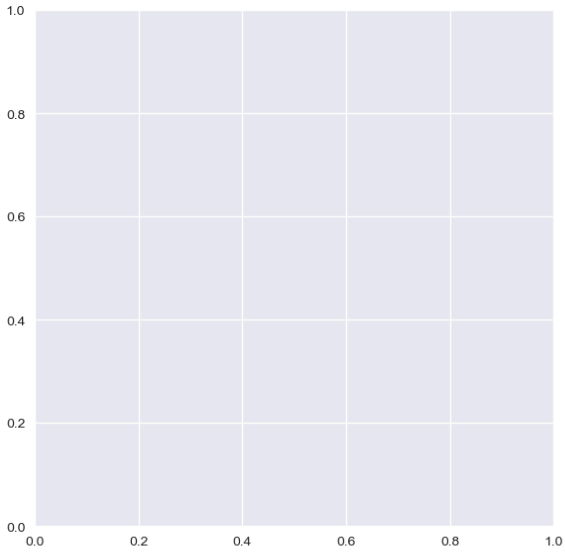
In [269…

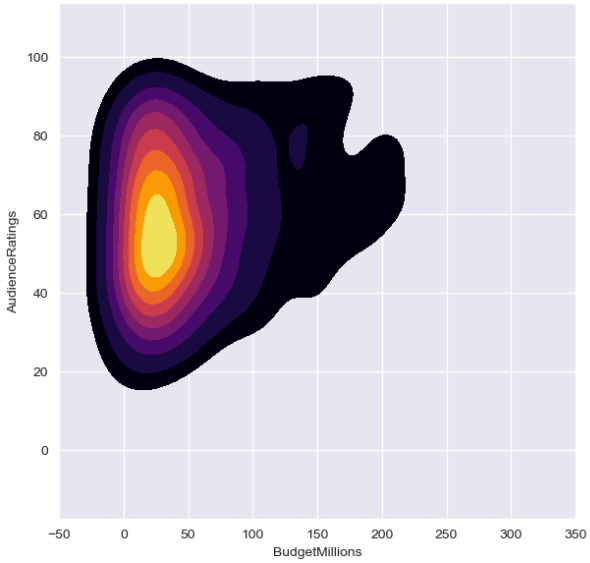
```
sns.set_style('darkgrid')
f, axes = plt.subplots (2,2, figsize = (15,15))
sns.kdeplot(data=movies,x='BudgetMillions',y='AudienceRatings',cmap='inferno',fill=
sns.kdeplot(data=movies,x='CriticRatings',y='AudienceRatings',cmap='cool',fill=True
sns.histplot(data=movies,x='BudgetMillions',y='CriticRatings',bins='auto',shrink=1,
sns.violinplot(data=movies,x='Year',y='CriticRatings',ax=axes[1,1])
plt.show()
```

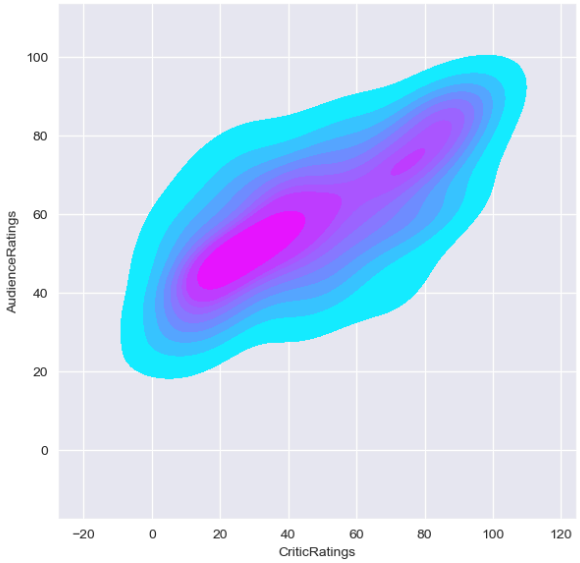

AudienceRatings
BudgetMillions

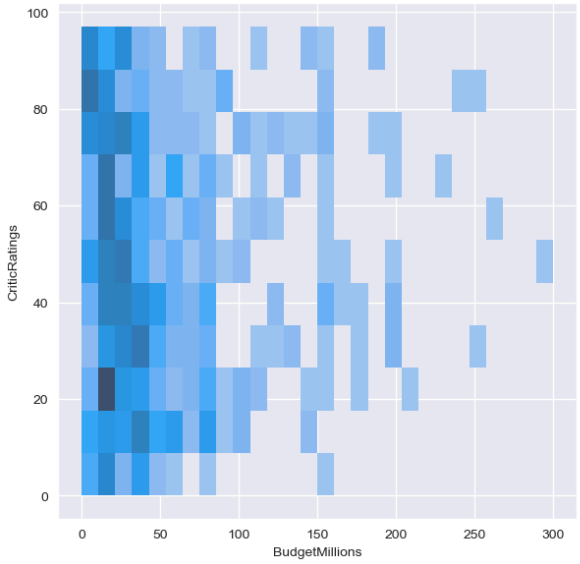

AudienceRatings
CriticRatings

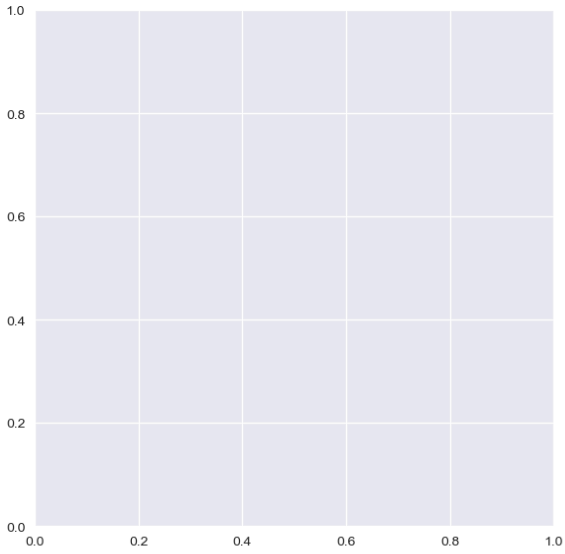

CriticRatings
BudgetMillions

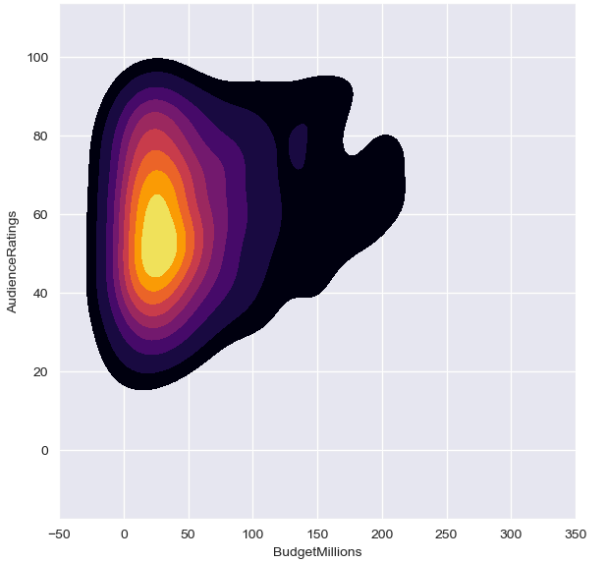


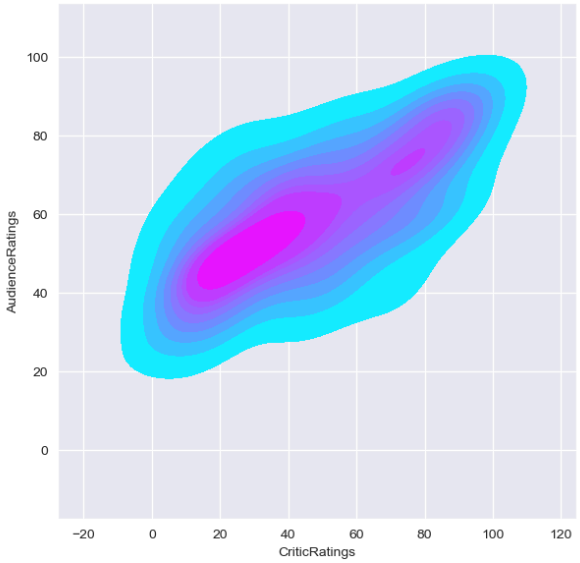

AudienceRatings
BudgetMillions

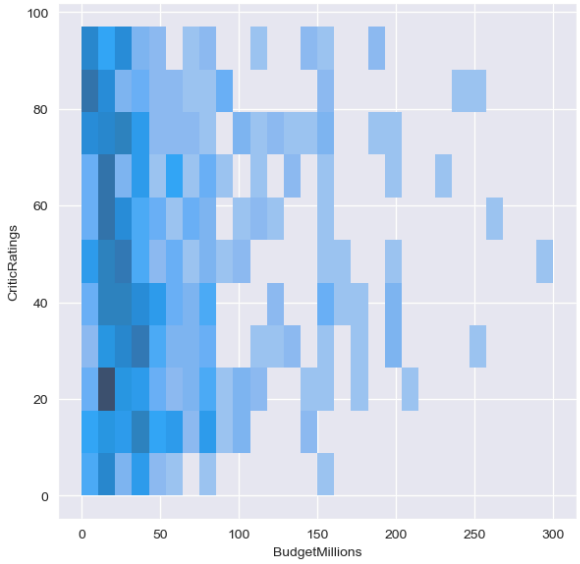

AudienceRatings
CriticRatings

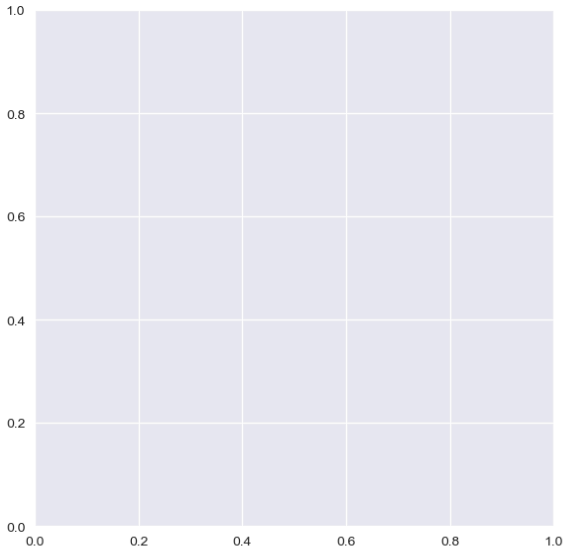

CriticRatings
BudgetMillions

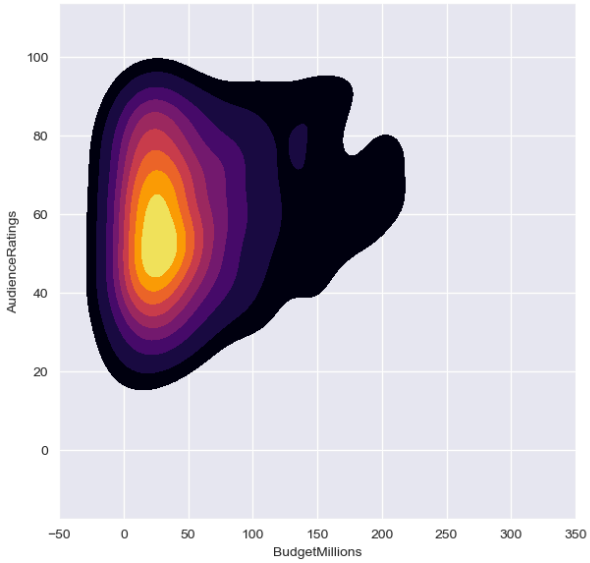


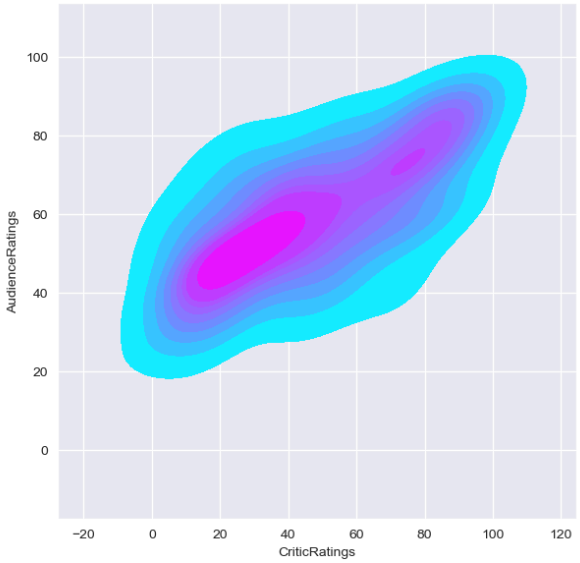

AudienceRatings
BudgetMillions

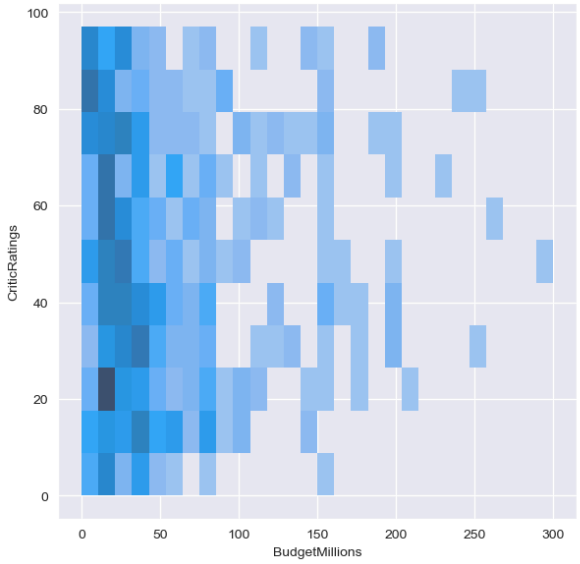

AudienceRatings
CriticRatings

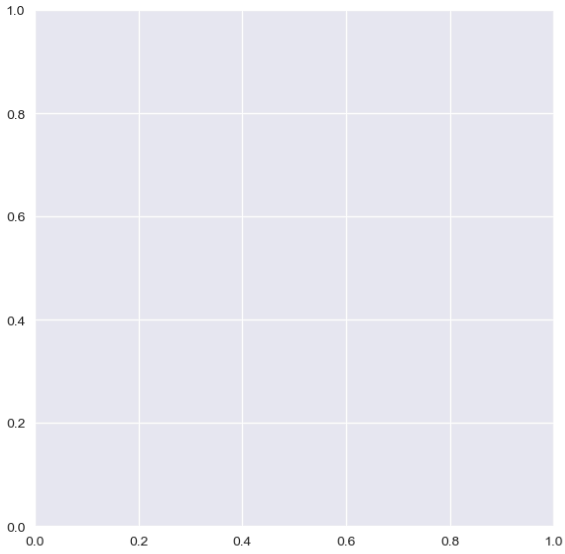

CriticRatings
BudgetMillions

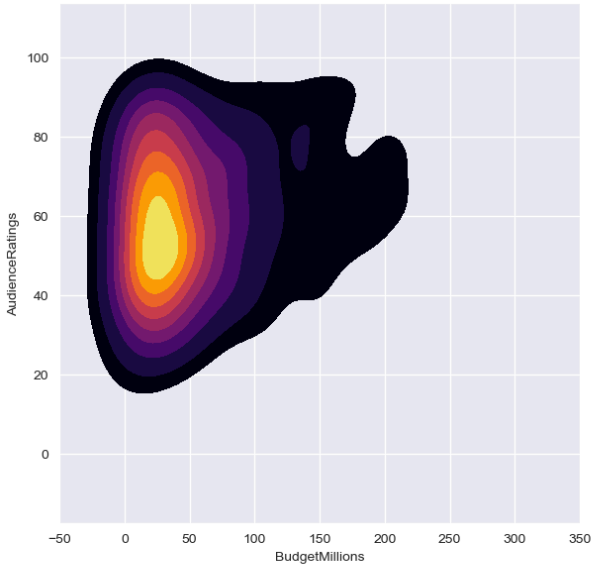


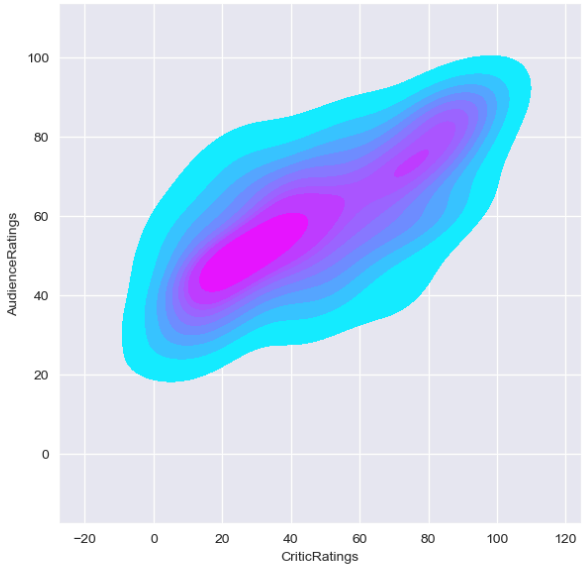

AudienceRatings
BudgetMillions

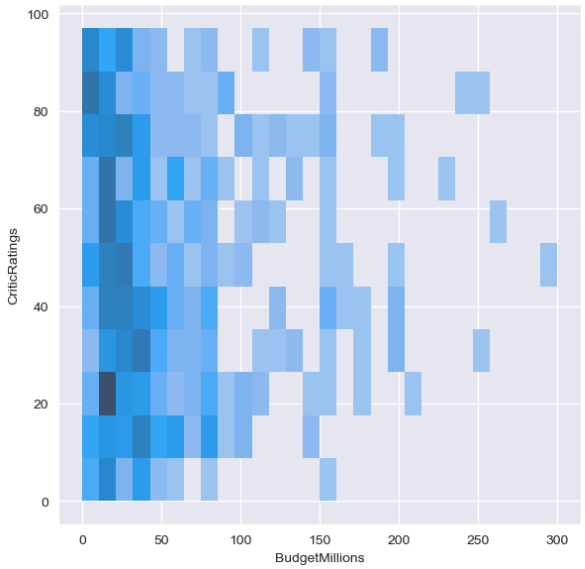

AudienceRatings
CriticRatings

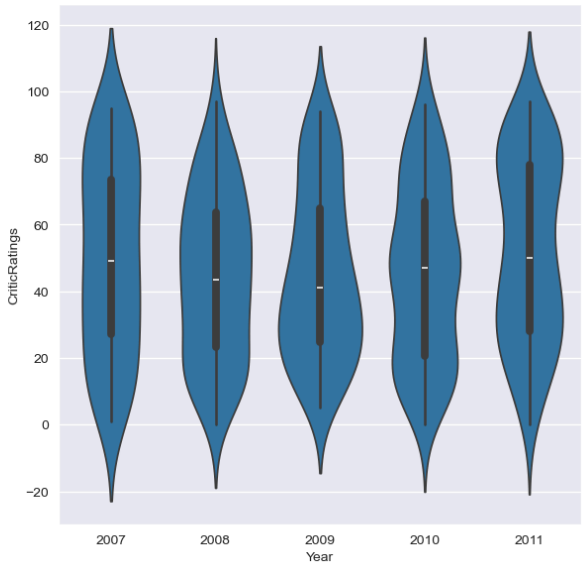

CriticRatings
BudgetMillions











In [259…
```
movies['Genre']
```

Out[259…
```
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', 'Roma
nce', 'Thriller']
```

In [ ]: