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

In [2]:
```
movies = pd.read_csv(r"C:\Users\saniy\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]:
```
movies.columns
```

Out[6]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

In [7]:
```
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 [8]:
```
movies.shape
```

Out[8]:
```
(559, 6)
```

In [9]:
```
movies
```

Out[9]:

	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 [10]:
```
movies.head()
```

Out[10]:

	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 [11]:
```
movies.tail()
```

Out[11]:

	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 [12]:
```
movies.head(1)
```

Out[12]:

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million $)	Year of release
0	(500) Days of Summer	Comedy	87	81	8	2009

In [13]:
```
movies.describe()
```

Out[13]:

	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million $)	Year of release
count	559.000000	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136	2009.152057
std	26.413091	16.826887	48.731817	1.362632
min	0.000000	0.000000	0.000000	2007.000000
25%	25.000000	47.000000	20.000000	2008.000000
50%	46.000000	58.000000	35.000000	2009.000000
75%	70.000000	72.000000	65.000000	2010.000000
max	97.000000	96.000000	300.000000	2011.000000

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

Out[14]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

In [15]:
```
movies.columns=
['Film','Genre','CriticRating','AudienceRating','BudgetMillions','Year']
```

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

Out[16]:

	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 [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()
```

```
<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 [19]:
```
movies.Film = movies.Film.astype('category')
```

In [20]:
```
movies.Film
```

Out[20]:
```
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 [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    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 [22]:
```
movies.Genre = movies.Genre.astype('category')
movies.Genre
```

Out[22]:
```
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 [23]:
```
movies.info()
```

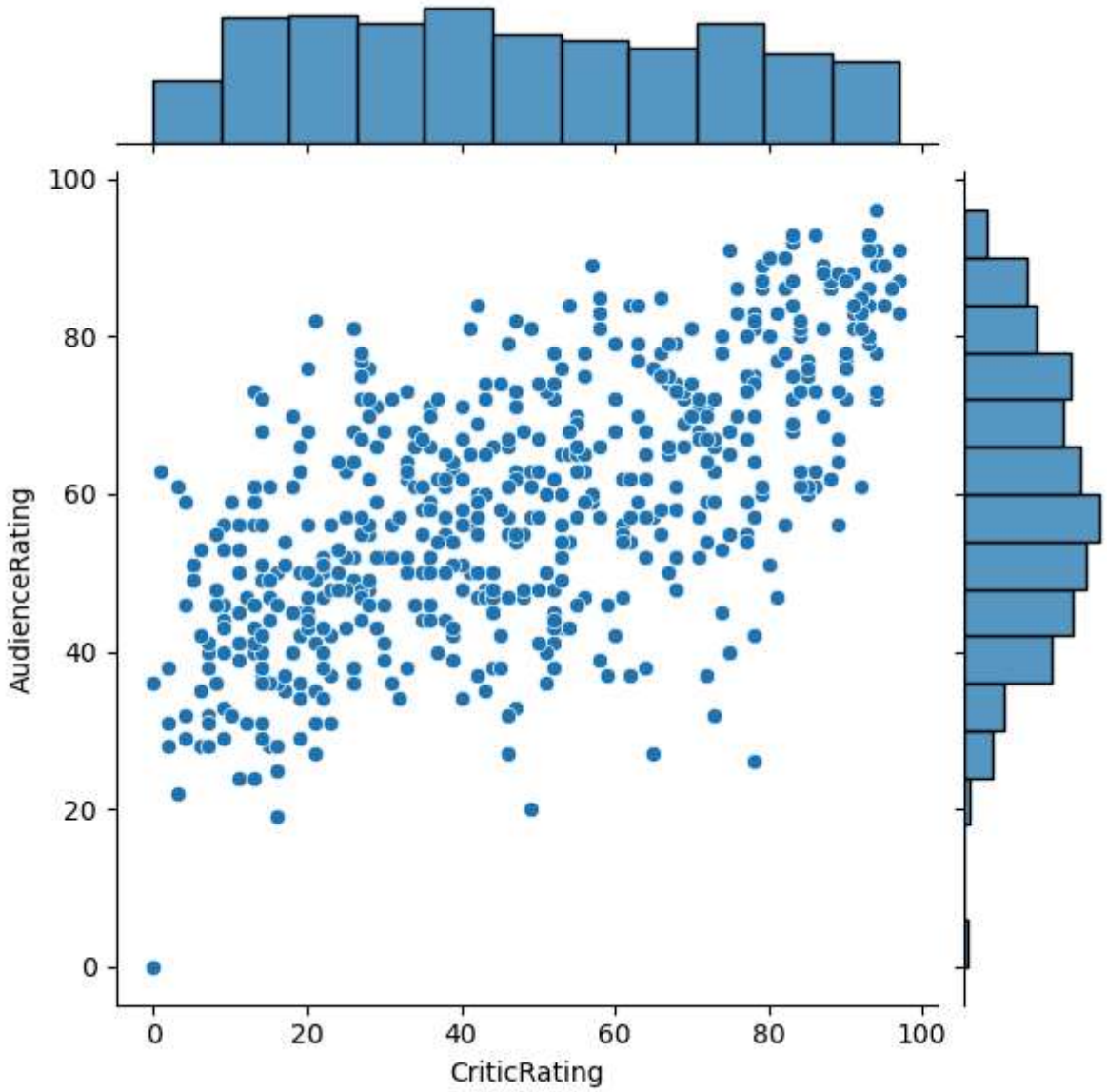
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Film            559 non-null    category
 1   Genre           559 non-null    category
 2   CriticRating    559 non-null    int64
 3   AudienceRating  559 non-null    int64
 4   BudgetMillions  559 non-null    int64
 5   Year            559 non-null    int64
dtypes: category(2), int64(4)
memory usage: 40.1 KB
```

In [24]:
```
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

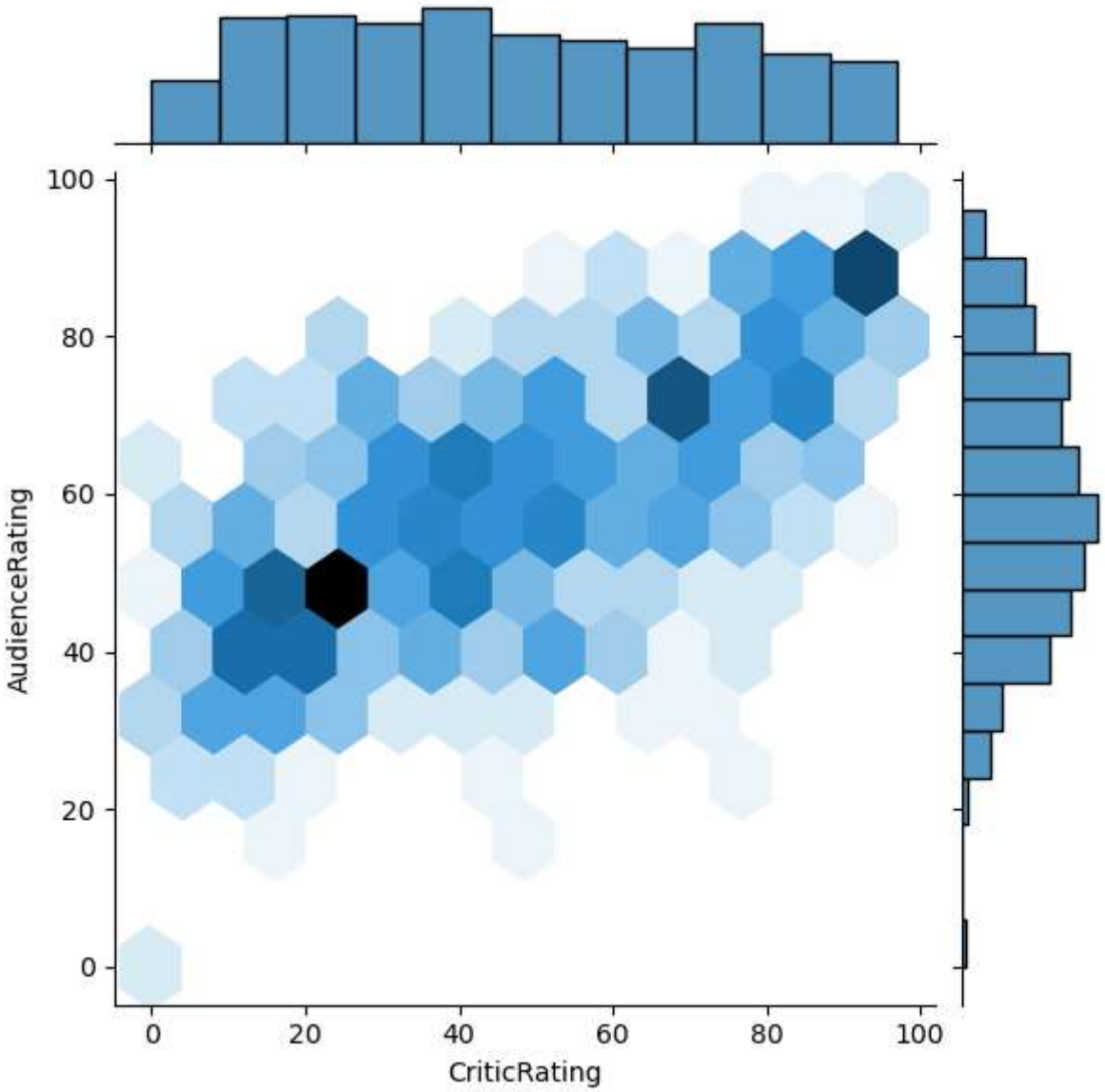
- basically joint plot is a scatter plot & it find the relation b/w audiene & critics

if you look up you can find the uniform distribution (critics)and normal distriution (audience)

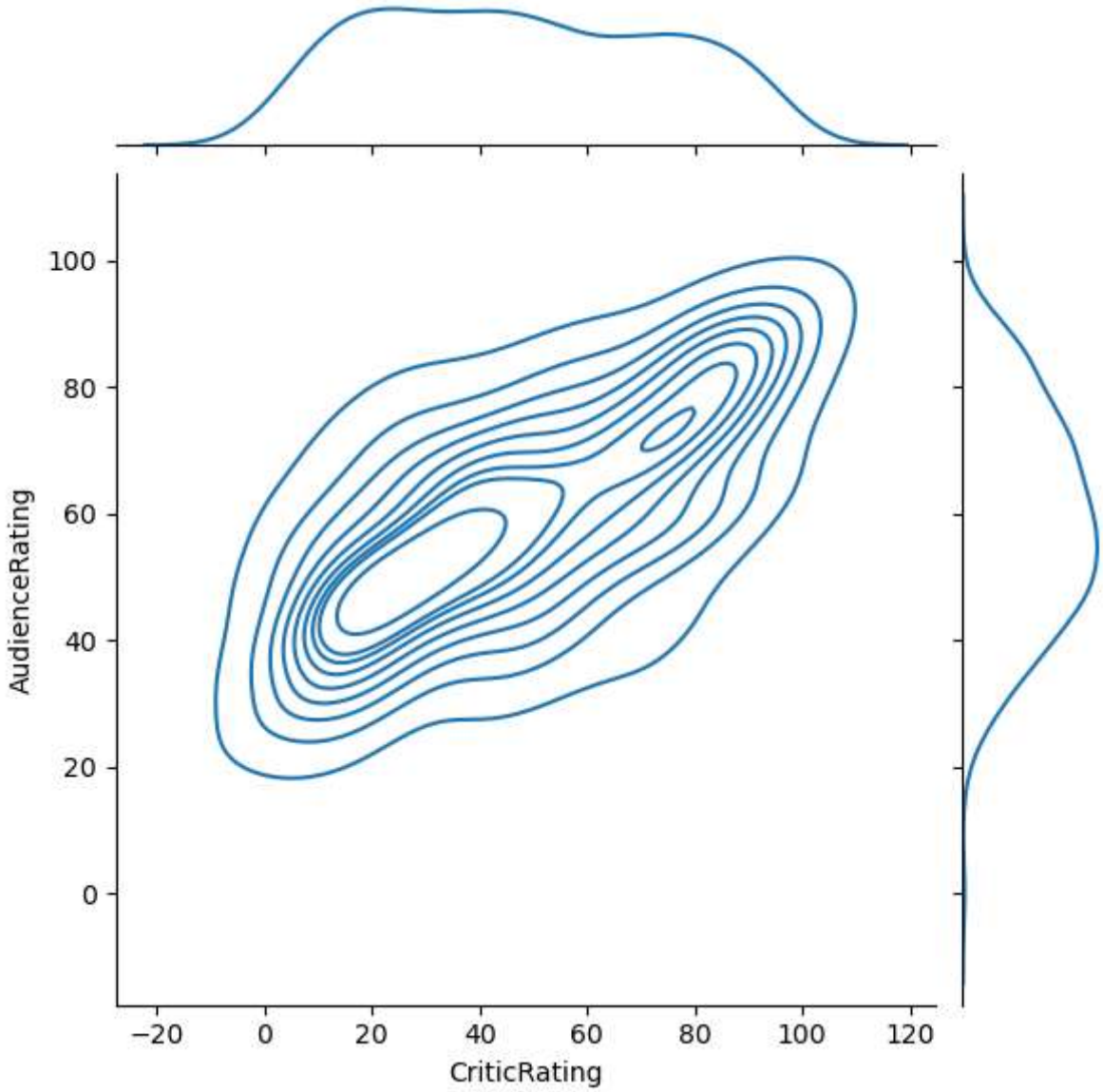
In [25]:
```
j = sns.jointplot(data = movies, x= 'CriticRating', y='AudienceRating')
```



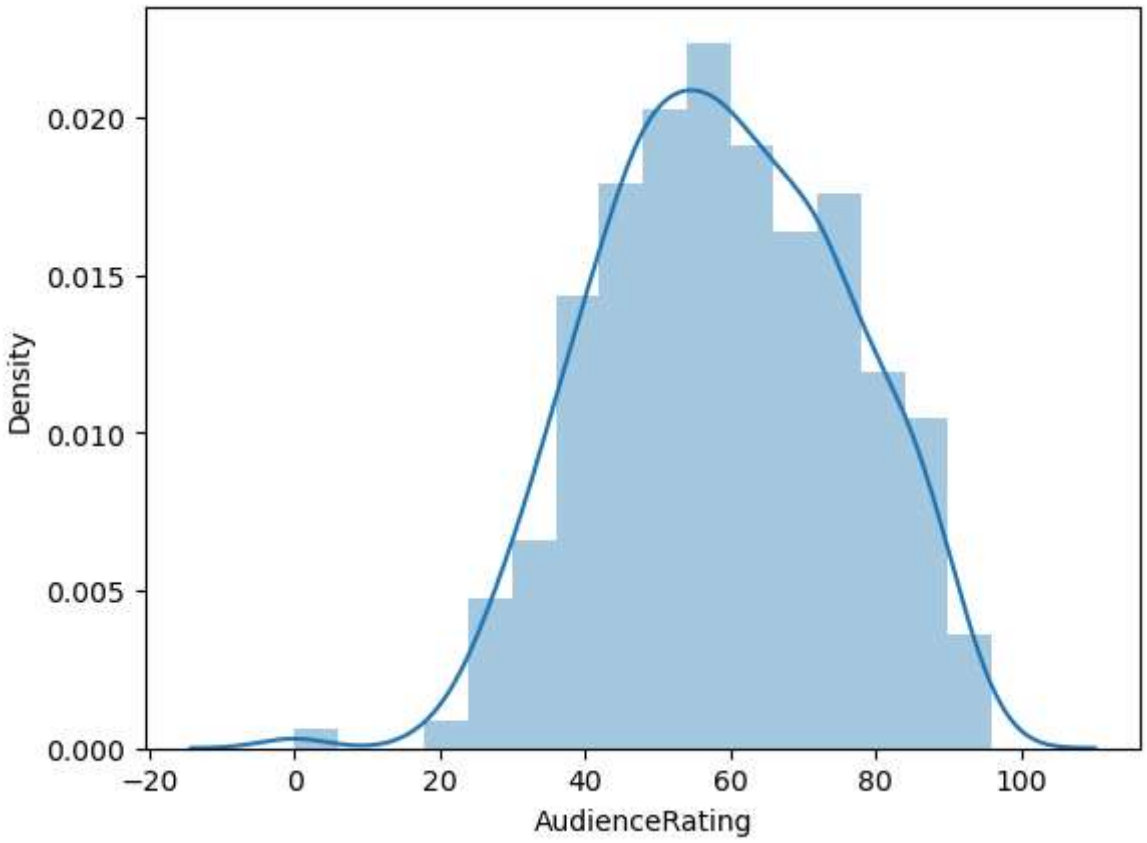
In [26]:
```
j = sns.jointplot(data = movies, x= 'CriticRating',
y='AudienceRating',kind='hex')
```



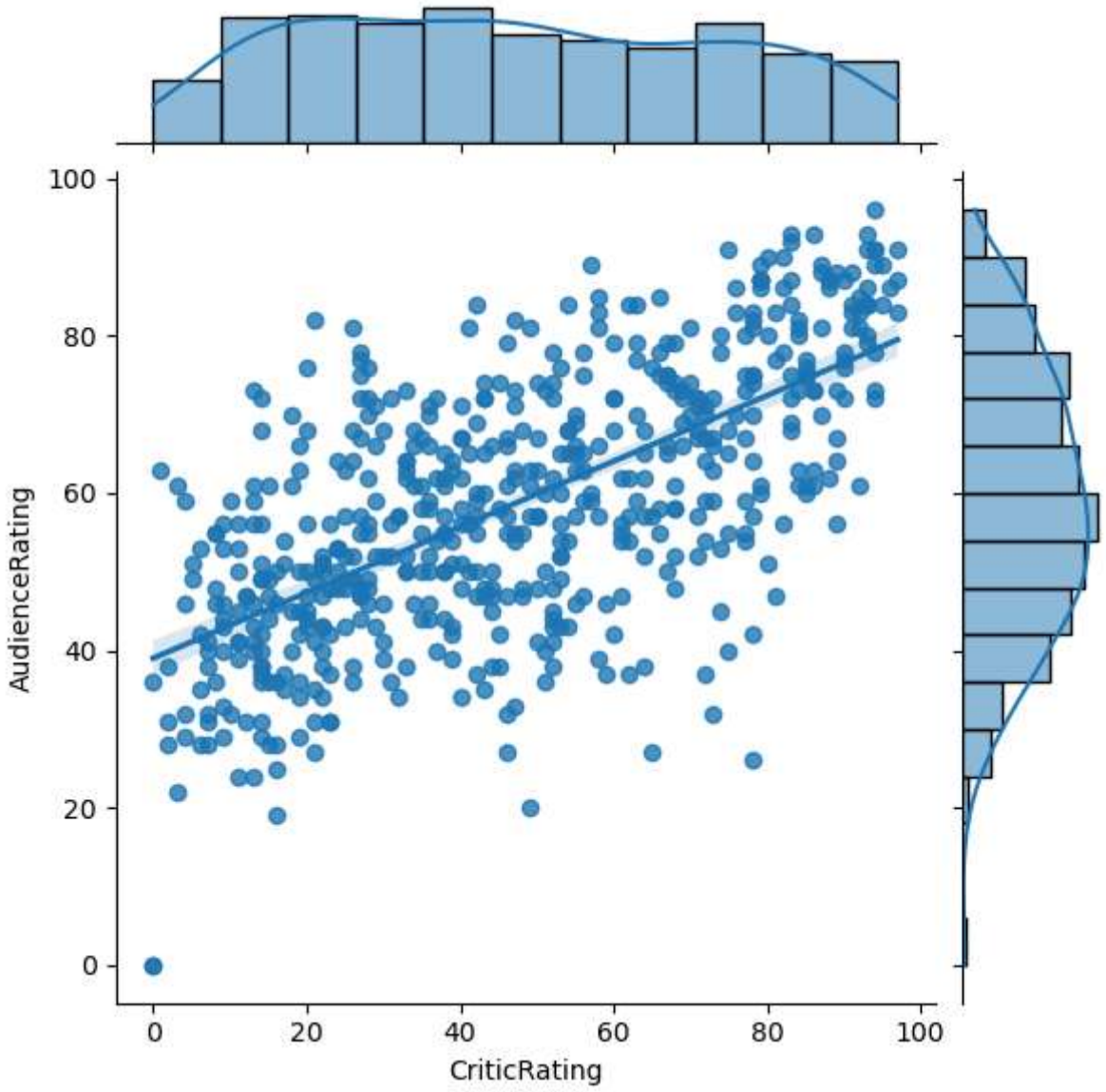
In [27]:
```
j = sns.jointplot(data = movies, x= 'CriticRating',
y='AudienceRating',kind='kde')
```



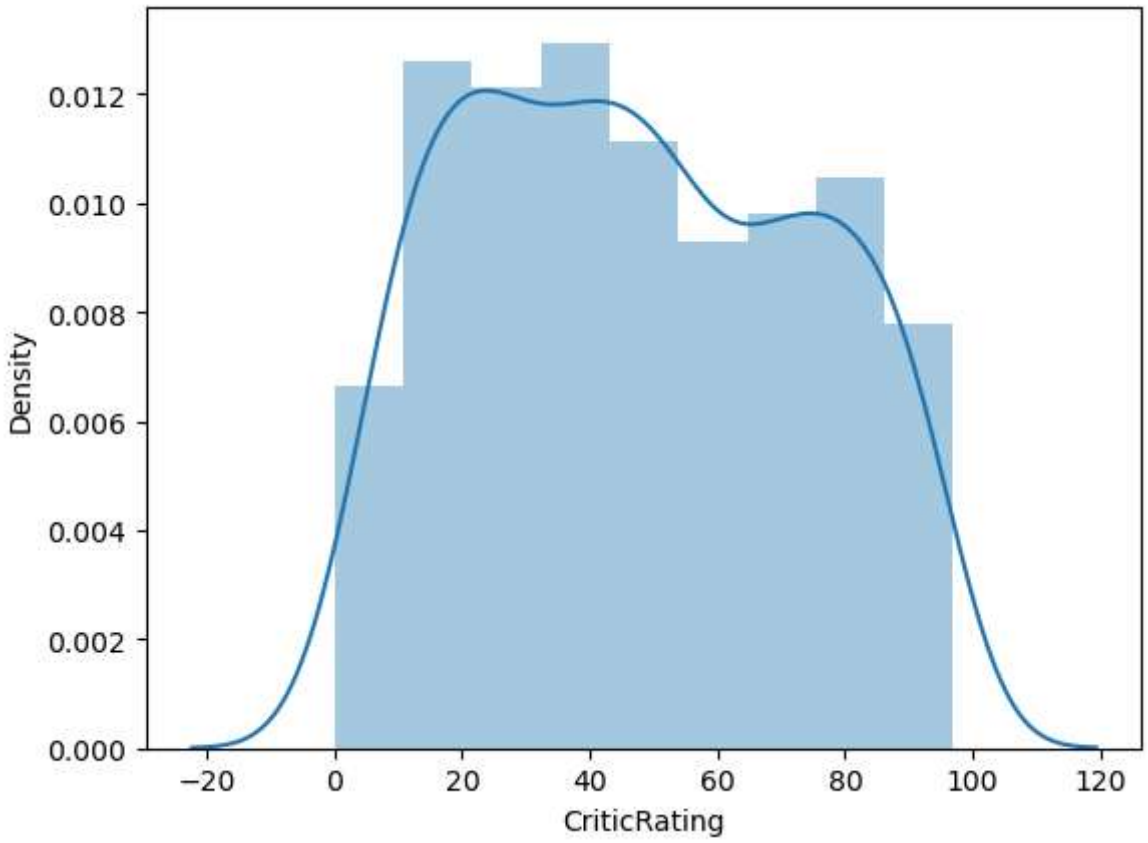
In [28]:
```
m1=sns.distplot(movies.AudienceRating) #histogram distribution plot
```



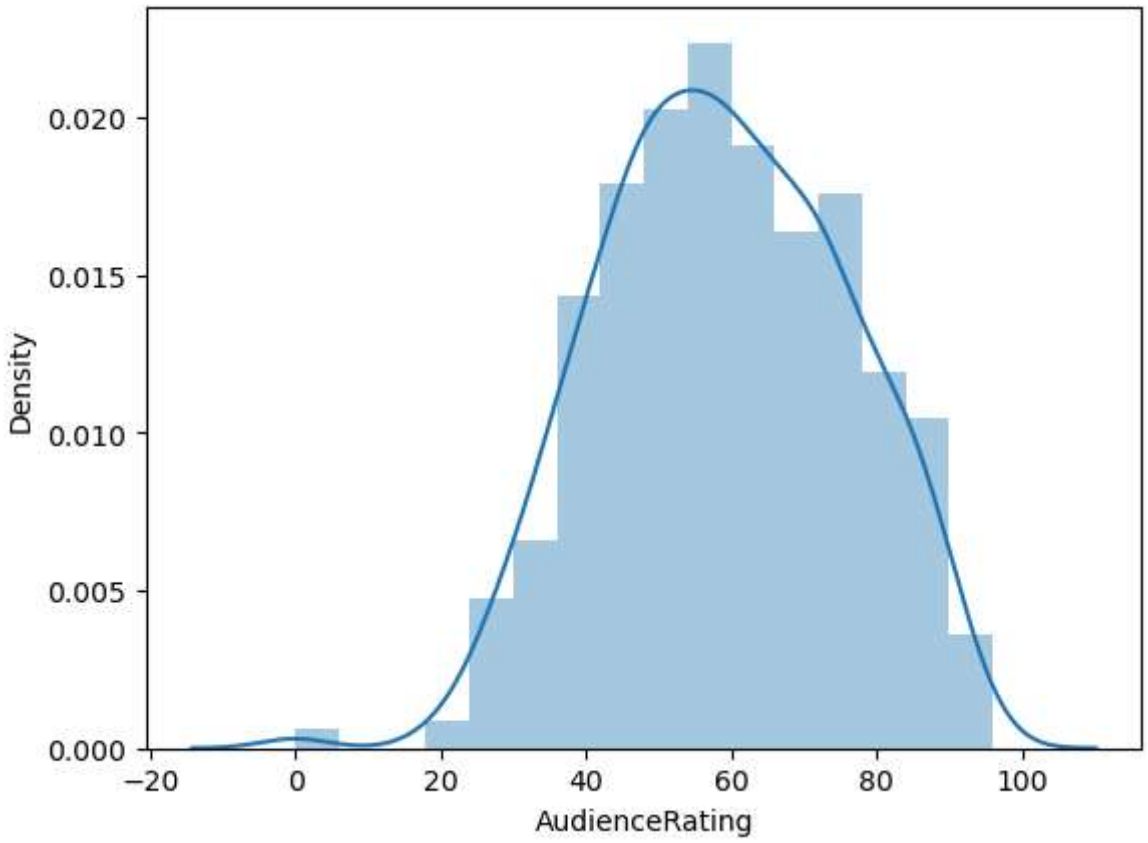
In [29]:
```
j = sns.jointplot( data = movies, x = 'CriticRating', y =
'AudienceRating',kind='reg')
```



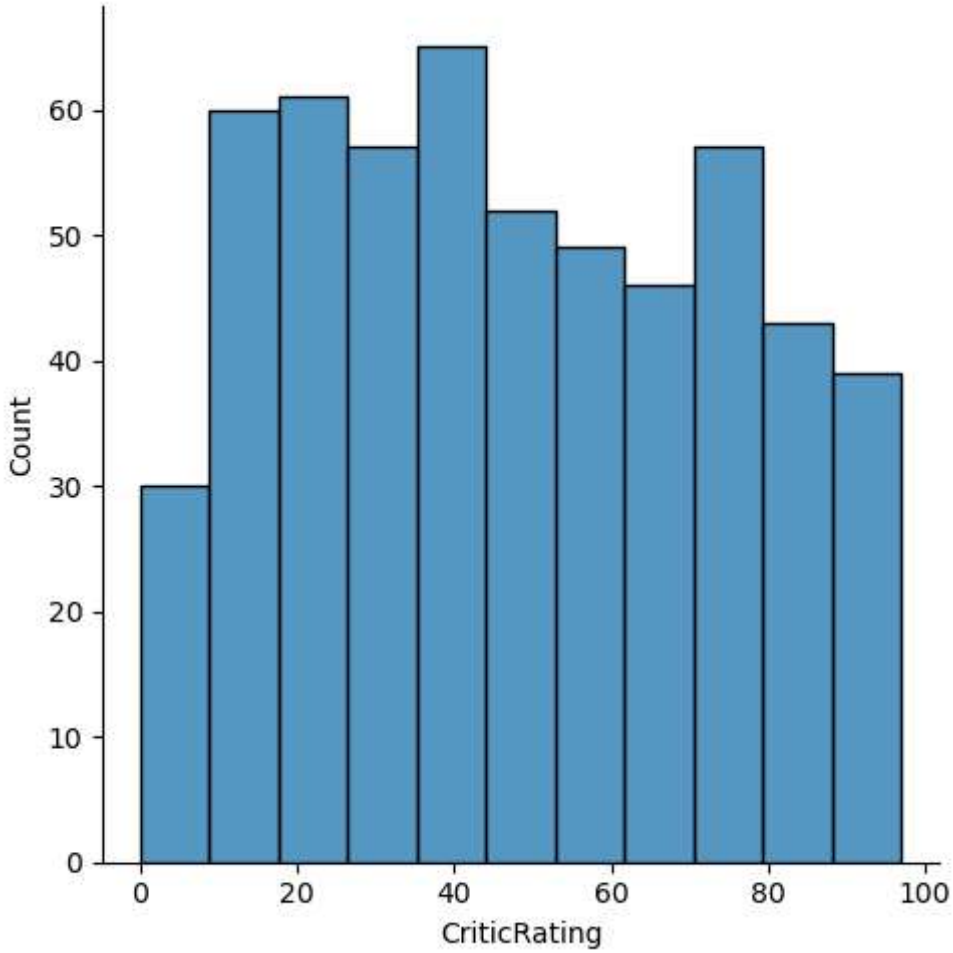
In [30]:
```
#Histograms
m1 = sns.distplot(movies.CriticRating)
```



In [31]:
```
m1 = sns.distplot(movies.AudienceRating)
```

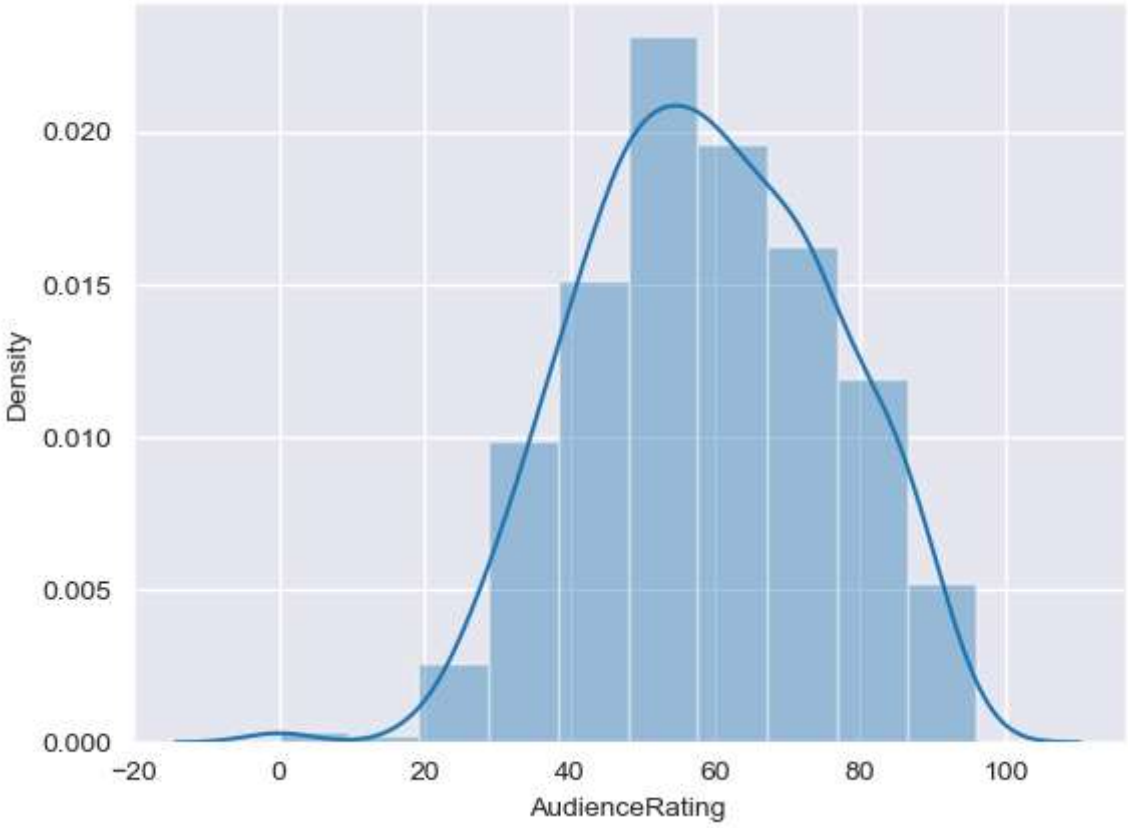


In [32]:
```
m1 = sns.displot(movies.CriticRating)
```

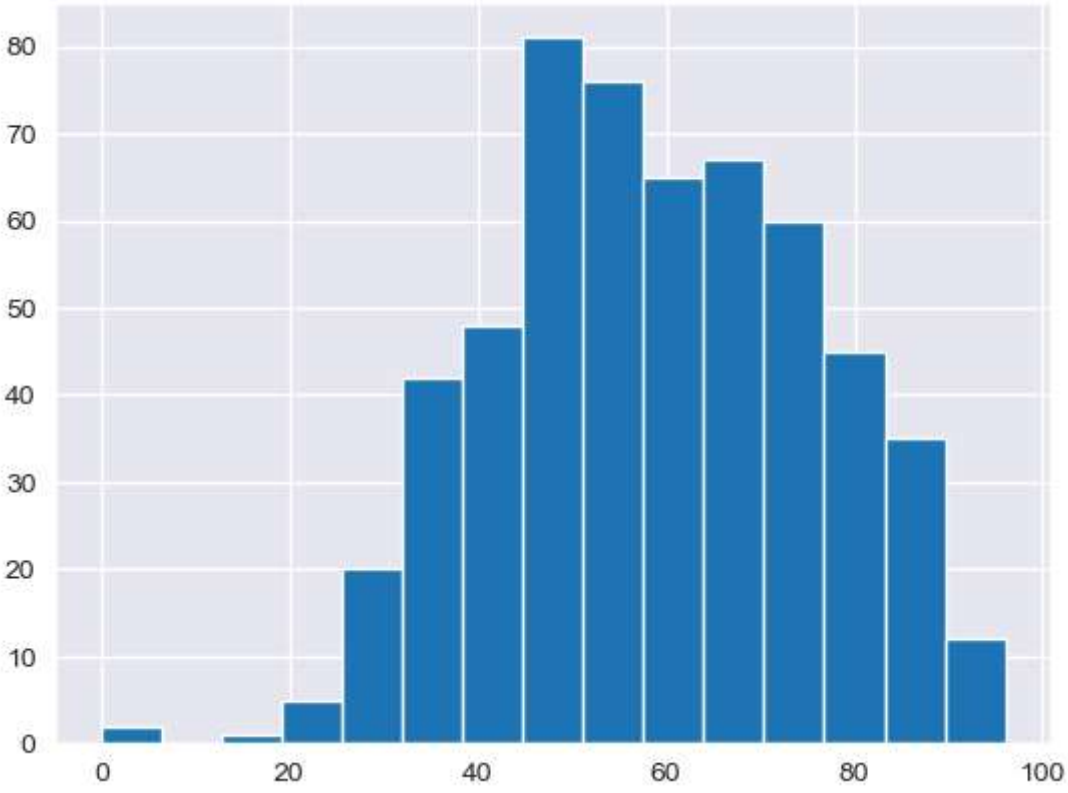


In [33]:
```
sns.set_style('darkgrid')
```

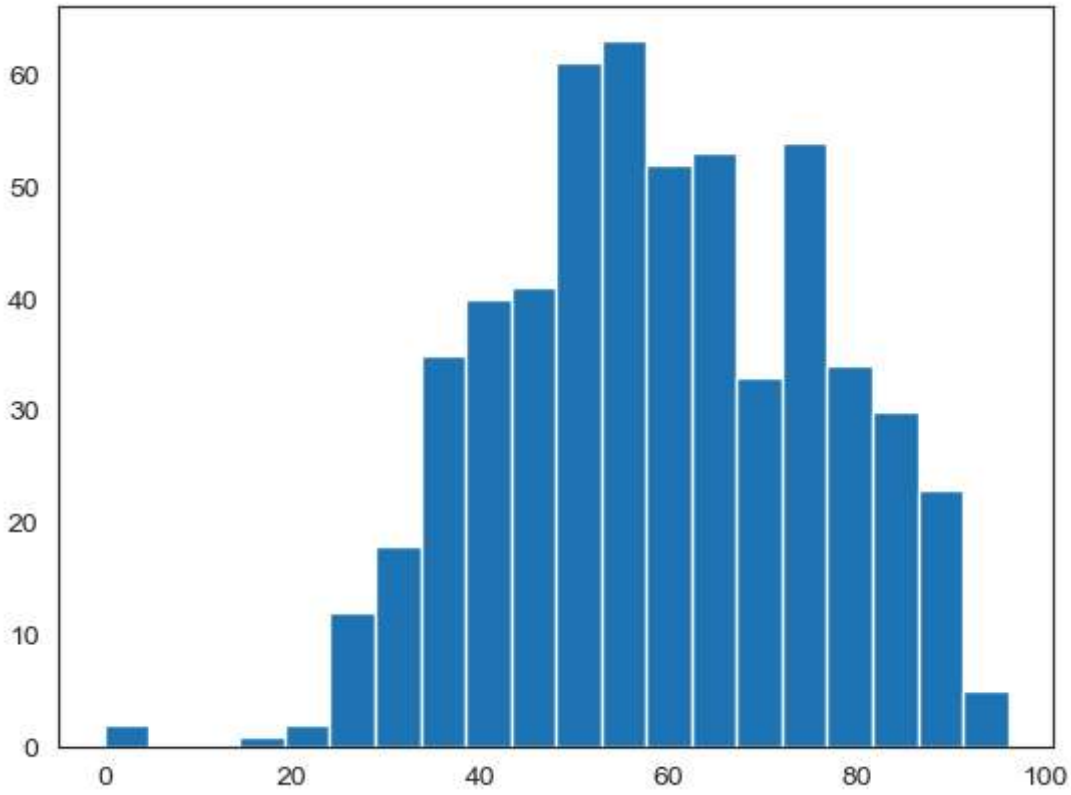
In [34]:
```
m2 = sns.distplot(movies.AudienceRating, bins=10)
```



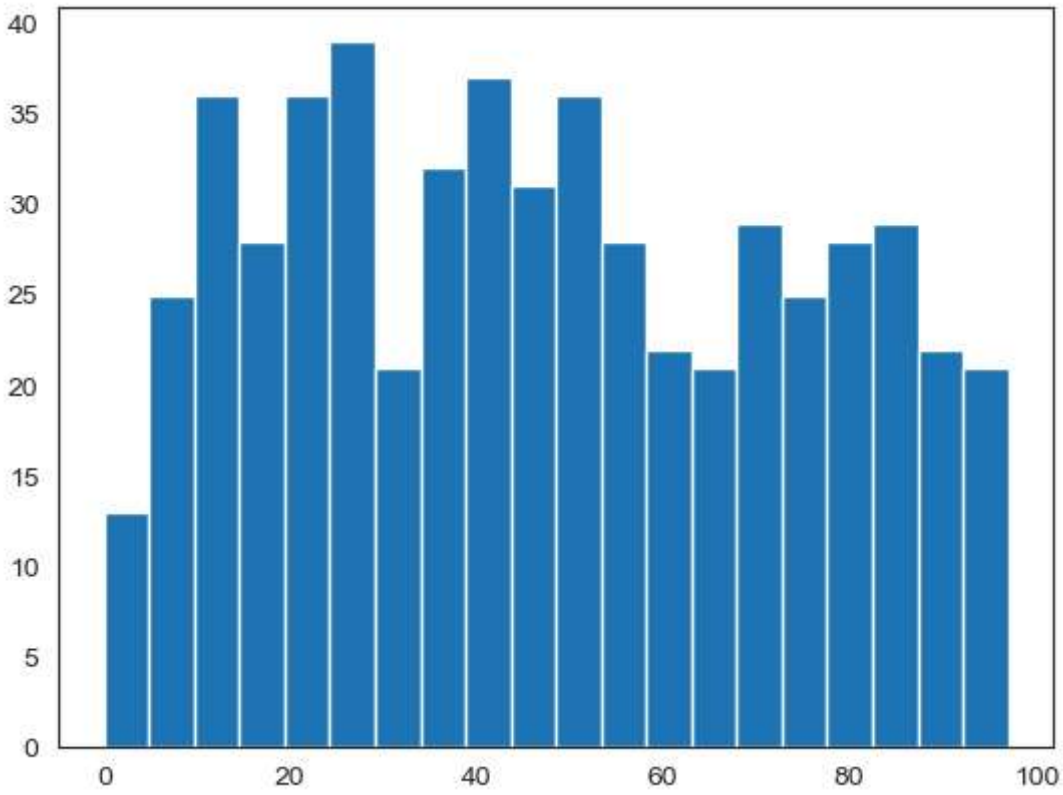
In [35]:
```
n1 = plt.hist(movies.AudienceRating, bins=15)
```



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

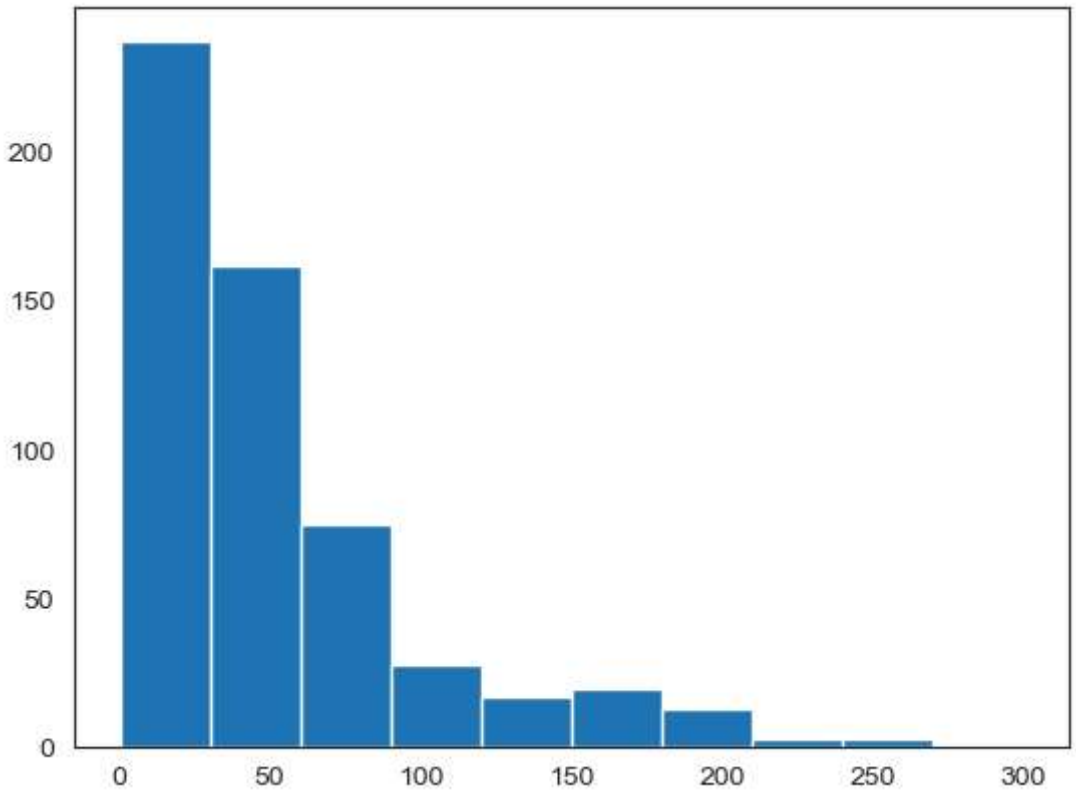


In [37]:
```
n1 = plt.hist(movies.CriticRating, bins=20) #uniform distribution
```

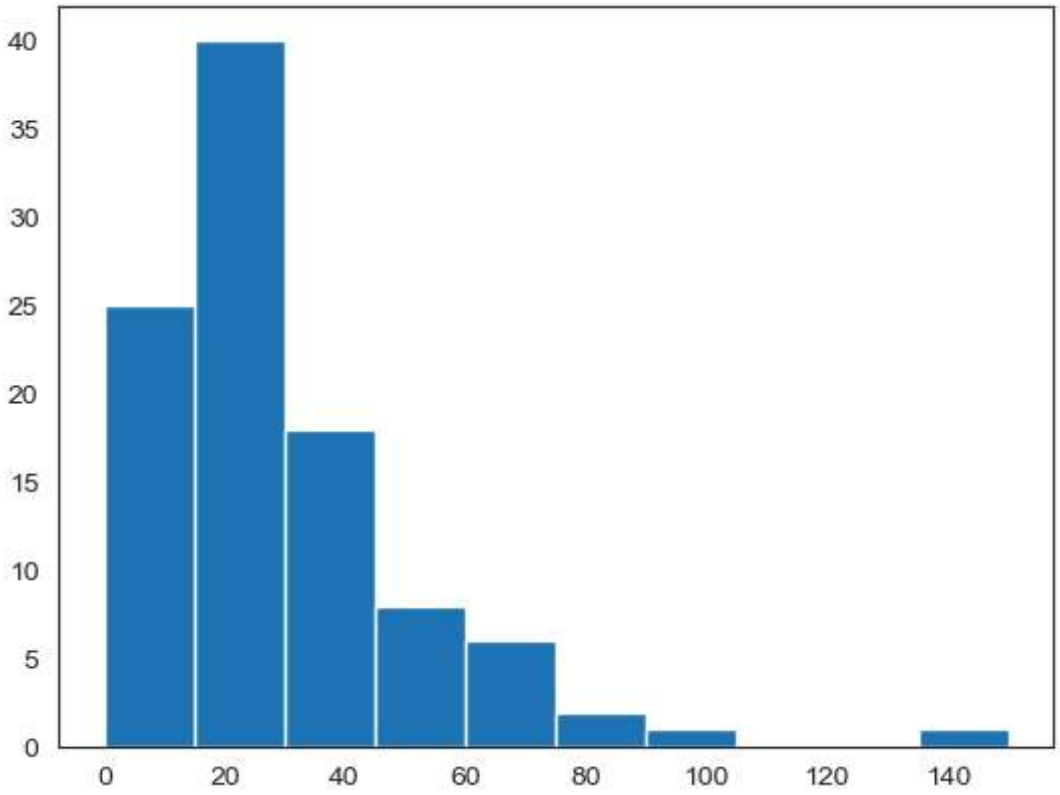


In [38]:
```
# <<< chat - 2

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

In [39]:
```
#h1 = plt.hist(movies.BudgetMillions)

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



In [40]:
```
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions)
plt.show()
```



In [41]:
```
movies.head()
```

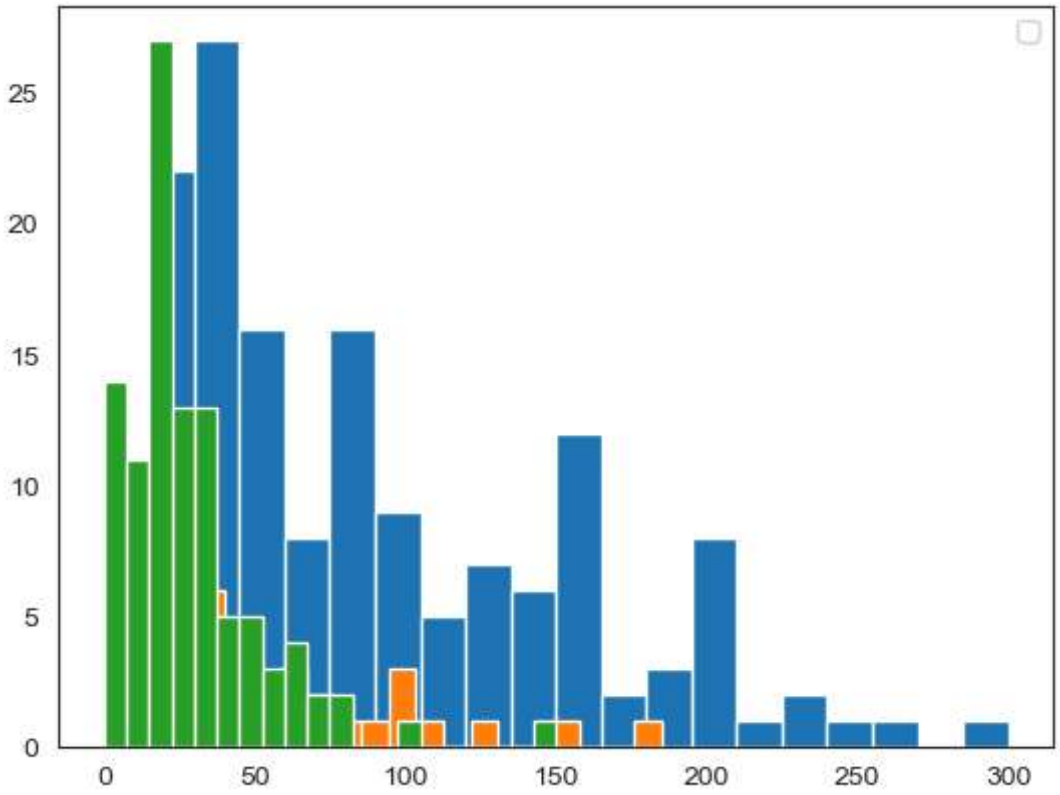
Out[41]:

	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

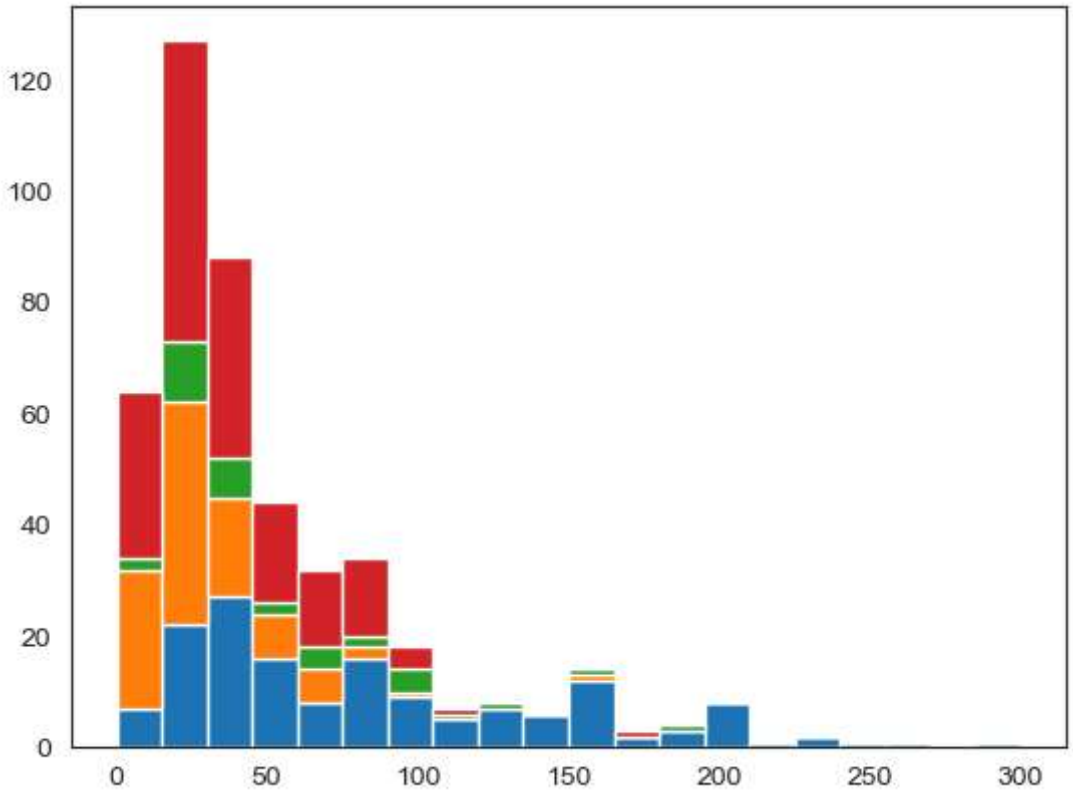
In [42]:
```
movies.Genre.unique()
```

Out[42]:
```
['Comedy', 'Adventure', 'Action', 'Horror', 'Drama', 'Romance', 'Thriller']
Categories (7, object): ['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Roma
nce', 'Thriller']
```

In [43]:
```
# 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()
```



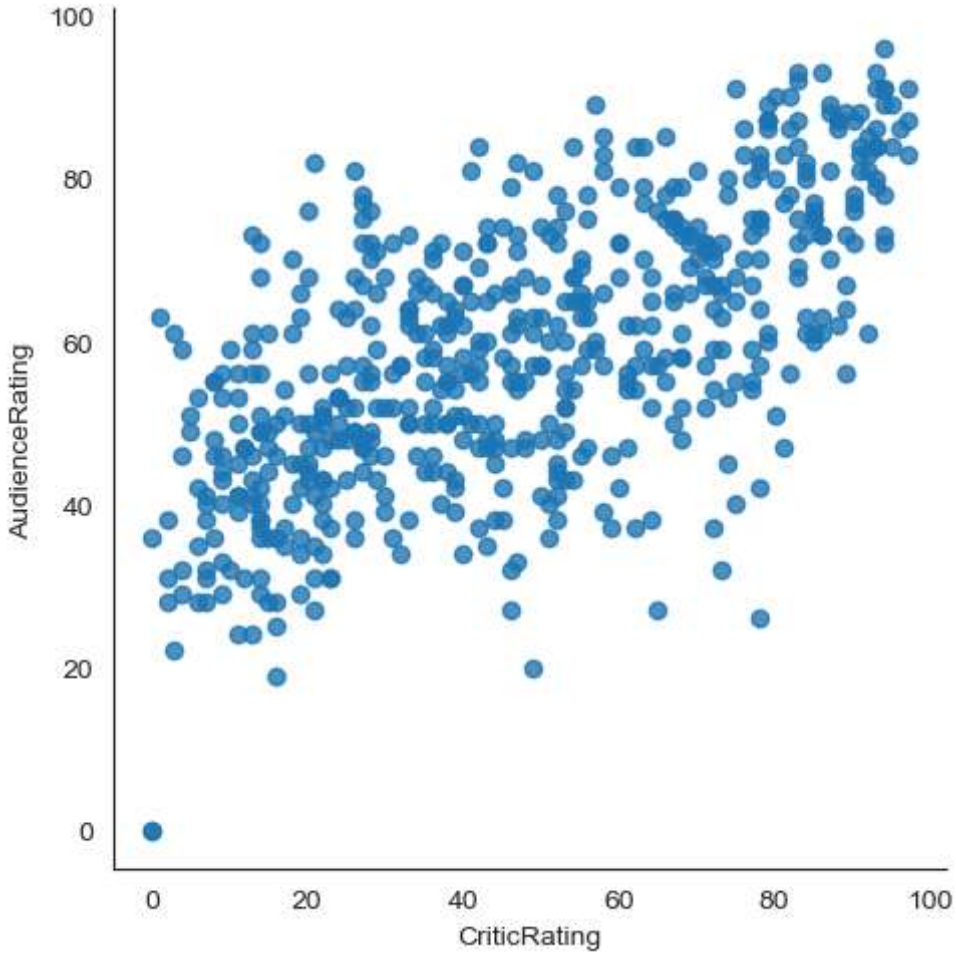
In [44]:
```
plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,\
          movies[movies.Genre == 'Drama'].BudgetMillions, \
          movies[movies.Genre == 'Thriller'].BudgetMillions, \
          movies[movies.Genre == 'Comedy'].BudgetMillions],
         bins = 20, stacked = True)
plt.show()
```



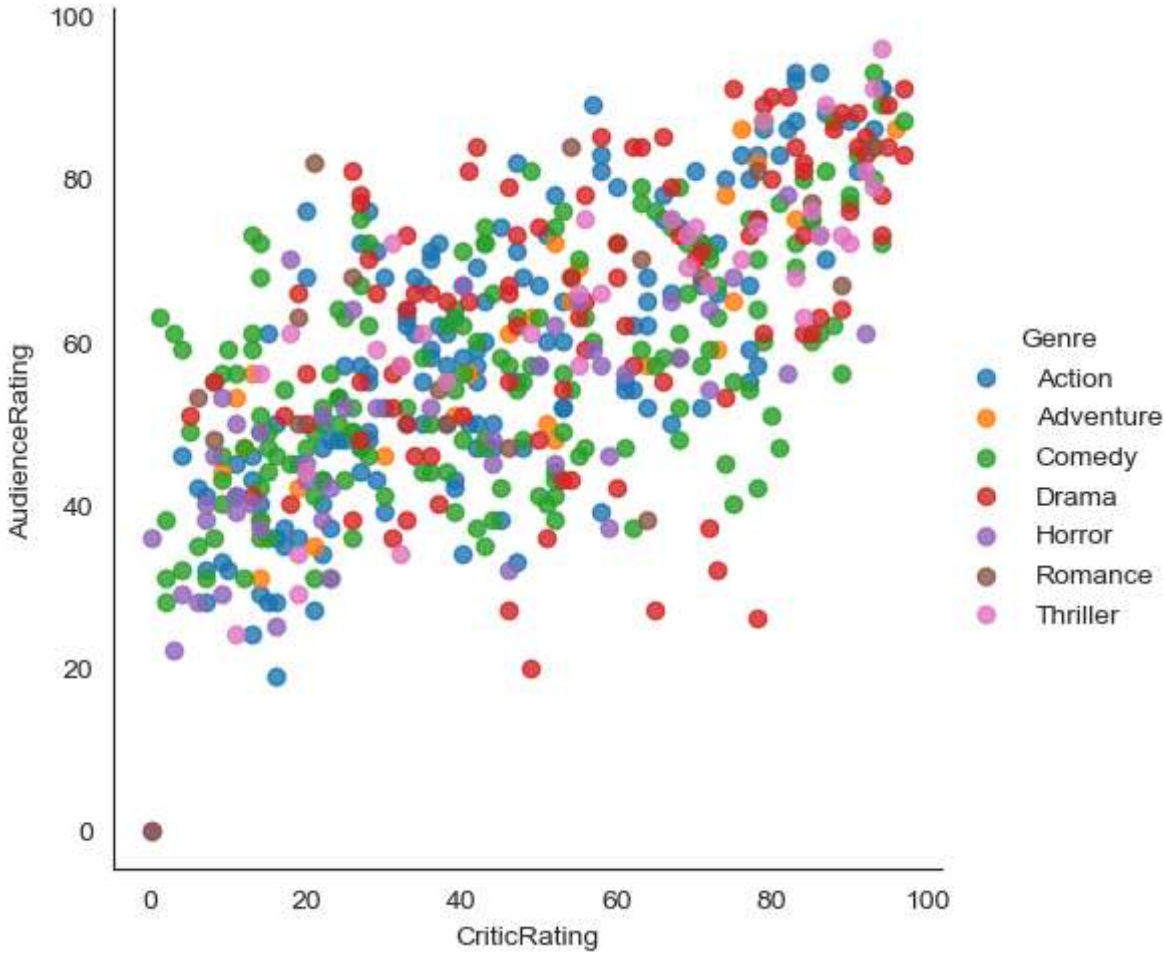
In [45]:
```
# if you have 100 categories you cannot copy & paste all the things

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

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

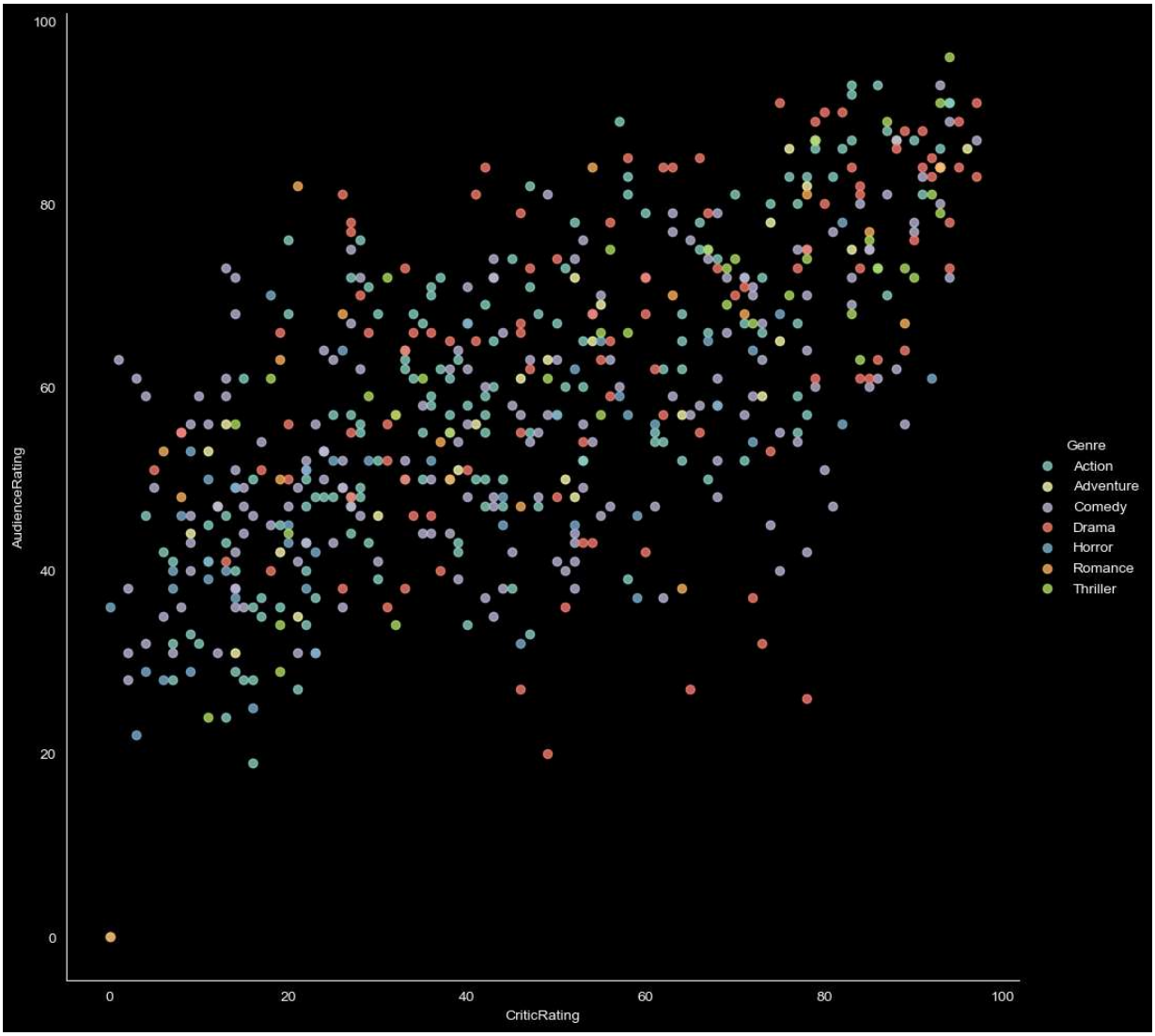
In [46]:
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False)
```



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



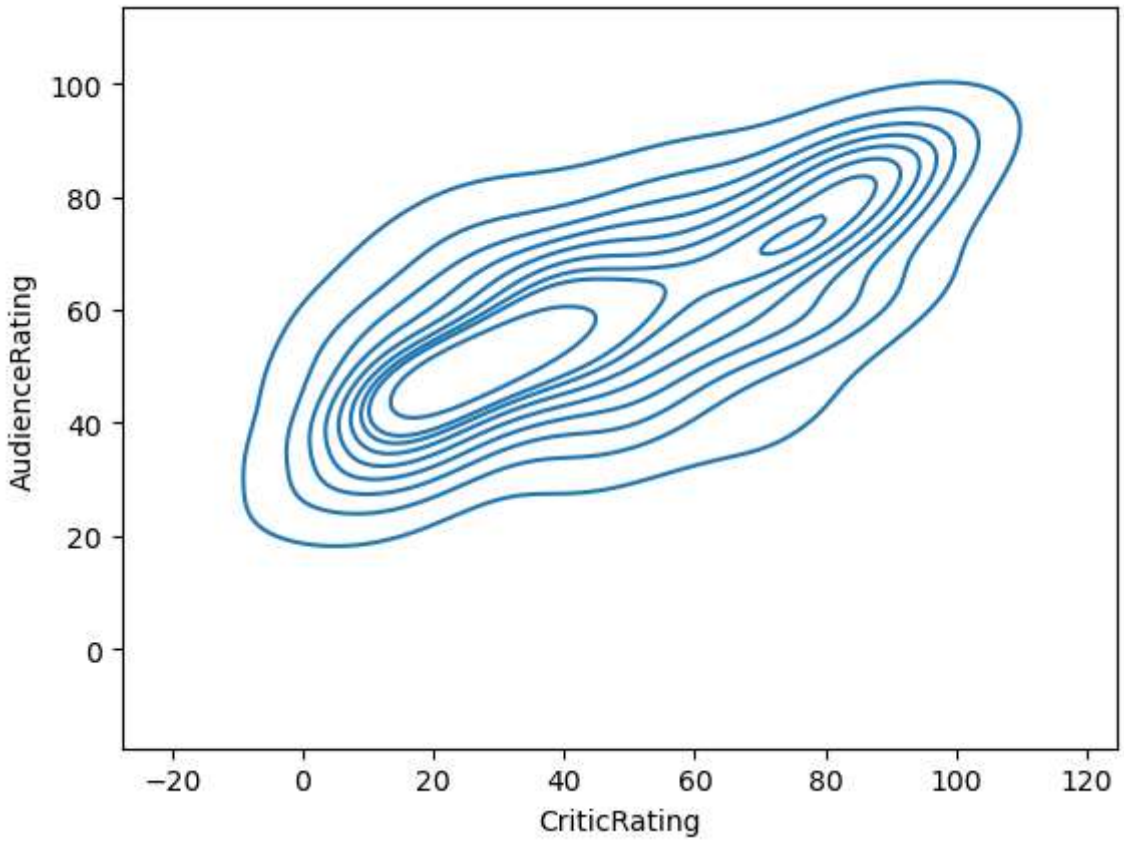
In [48]:
```
plt.style.use('dark_background')
vis1 = sns.lmplot(data=movies, x='CriticRating',
y='AudienceRating',fit_reg=False, hue = 'Genre', height= 10,aspect=1)
```



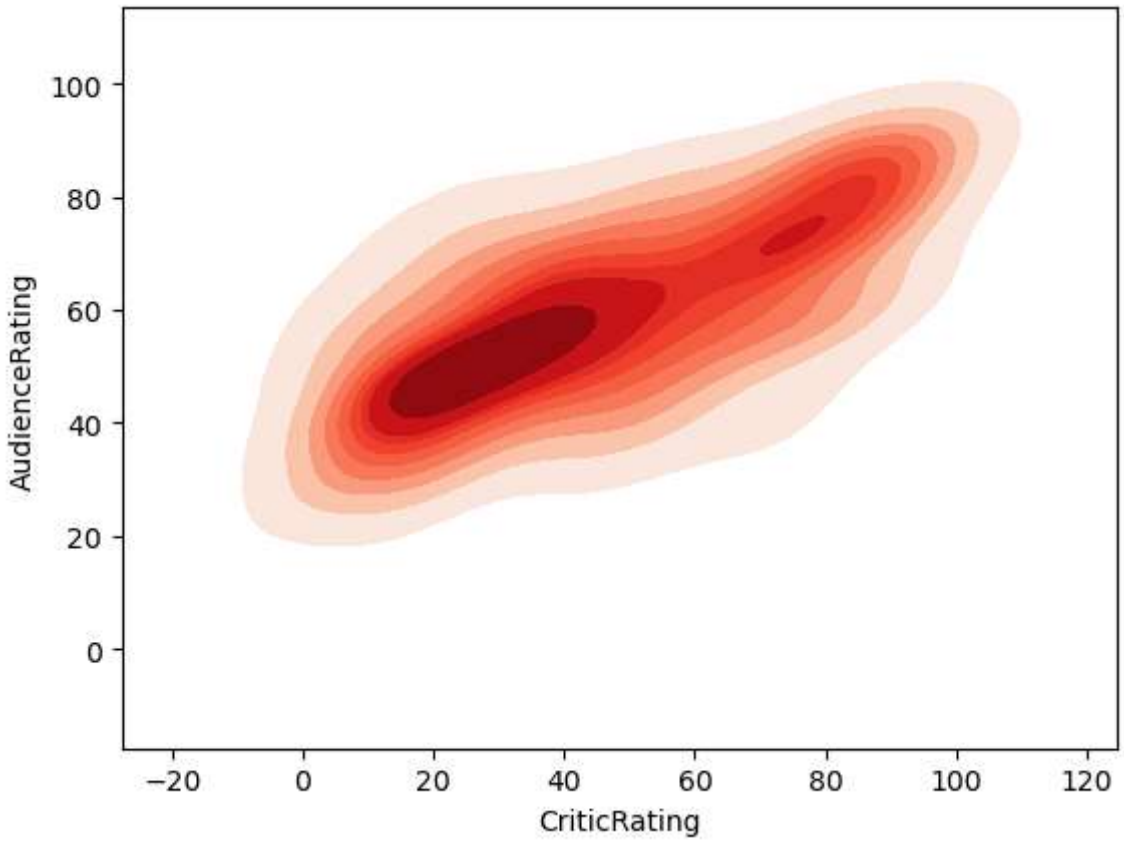
Kernal Density Estimate plot (KDE PLOT) # how can i visulize audience rating & critics rating . using scatterplot

In [49]:
```
plt.style.use('default')
```

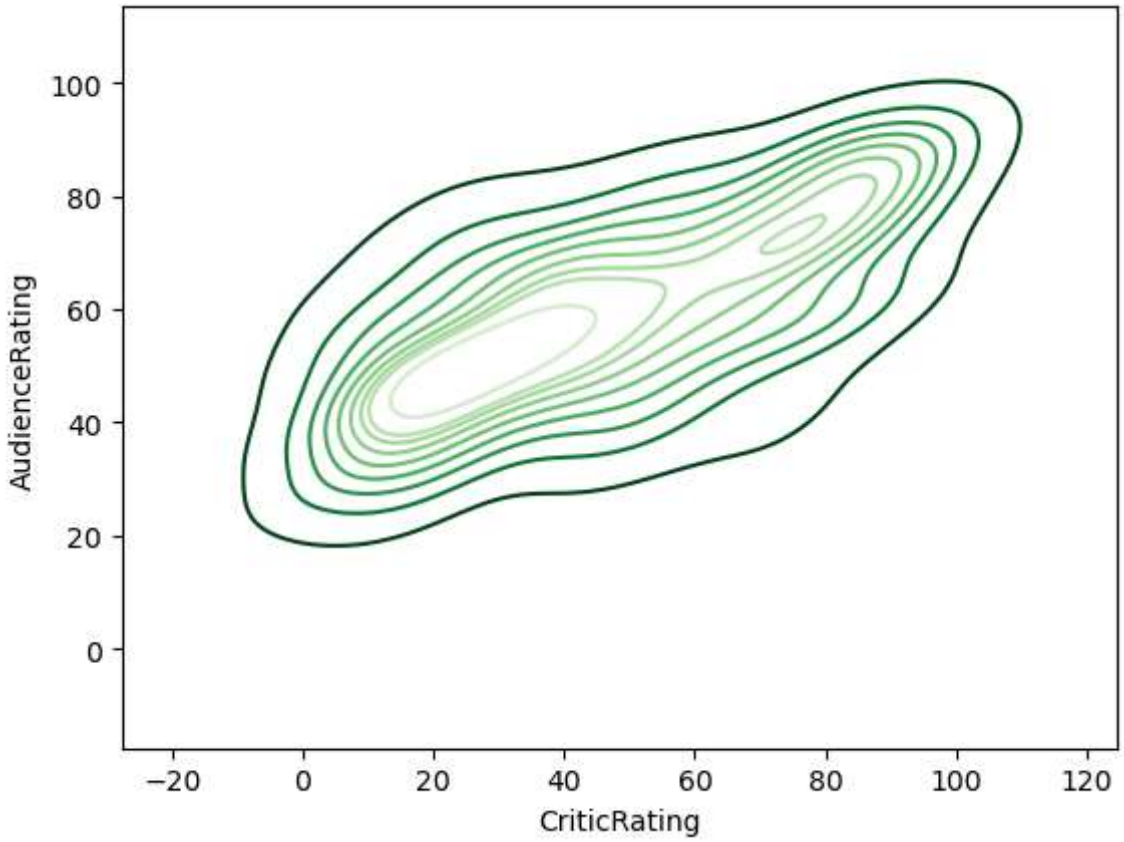
In [50]:
```
k1 = sns.kdeplot(x ='CriticRating',y ='AudienceRating', data = movies )

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



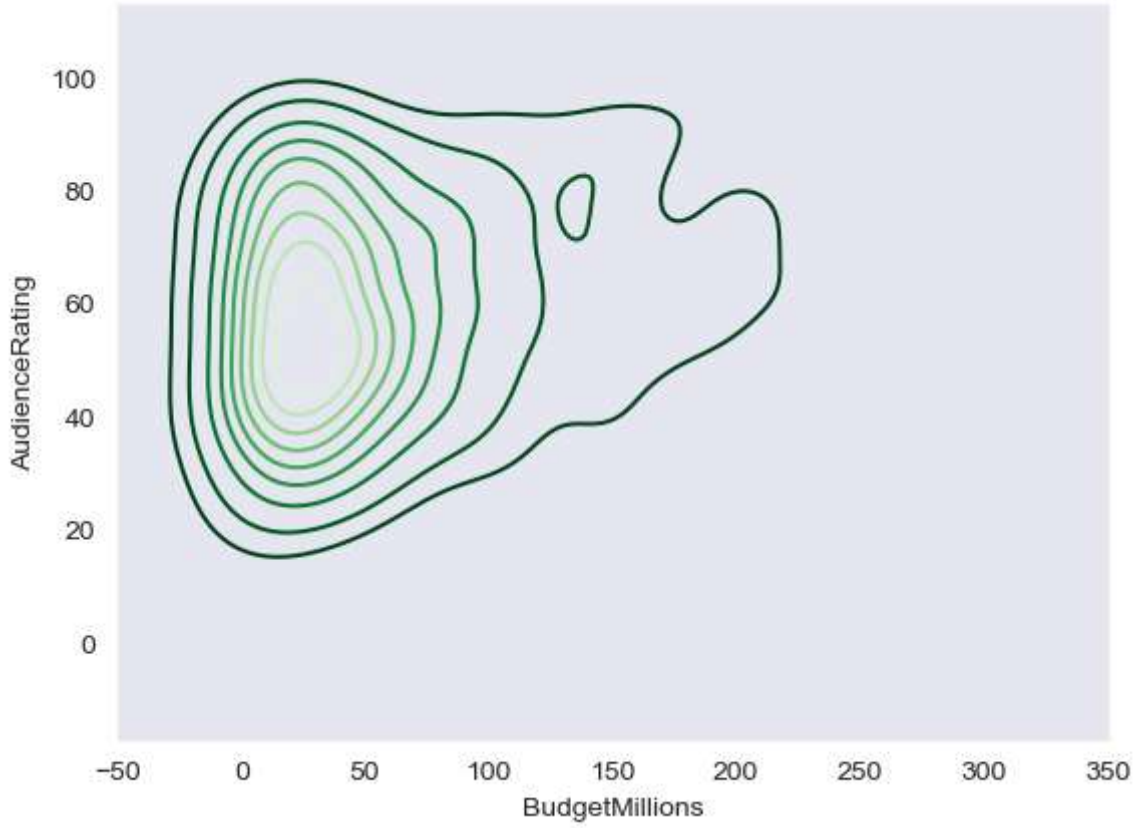
In [51]:
```
k1 = sns.kdeplot(data=movies, x='CriticRating', y='AudienceRating', shade=True,
shade_lowest=False, cmap='Reds')
```



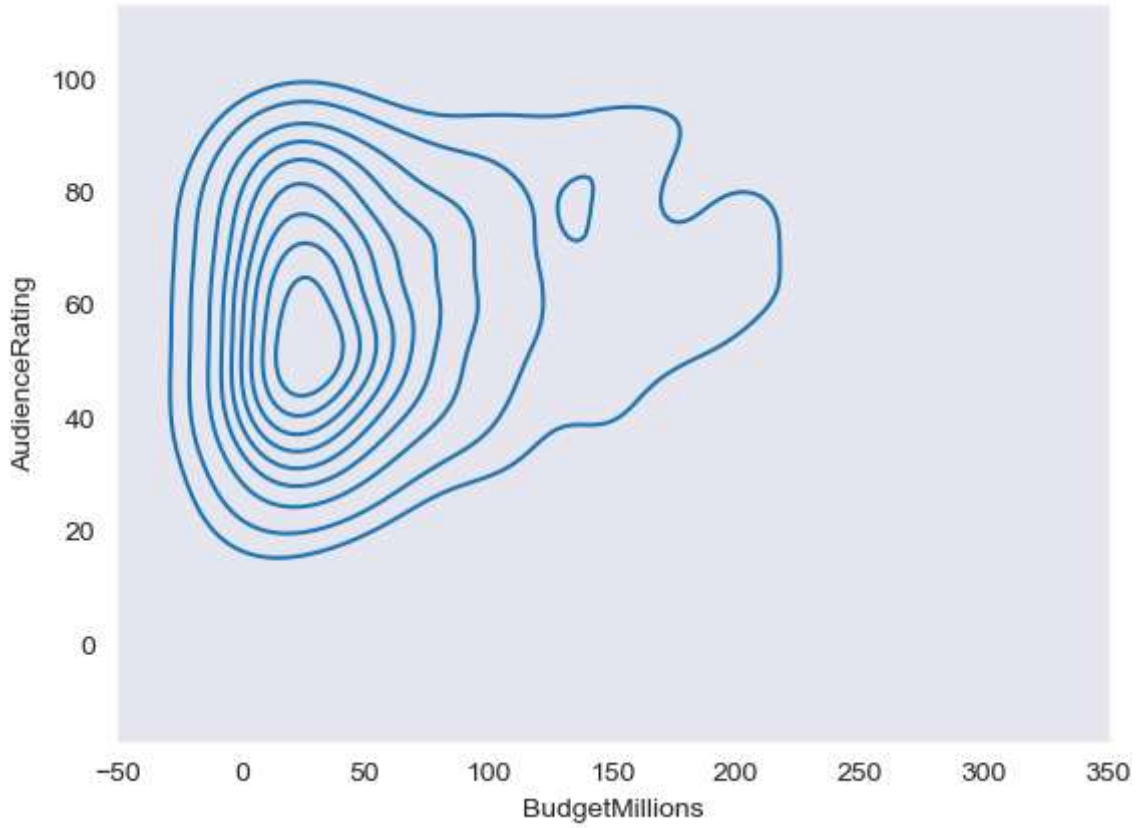
In [52]:
```
k2 = sns.kdeplot(data = movies, x= 'CriticRating', y=
'AudienceRating',shade_lowest=False,cmap='Greens_r')
```



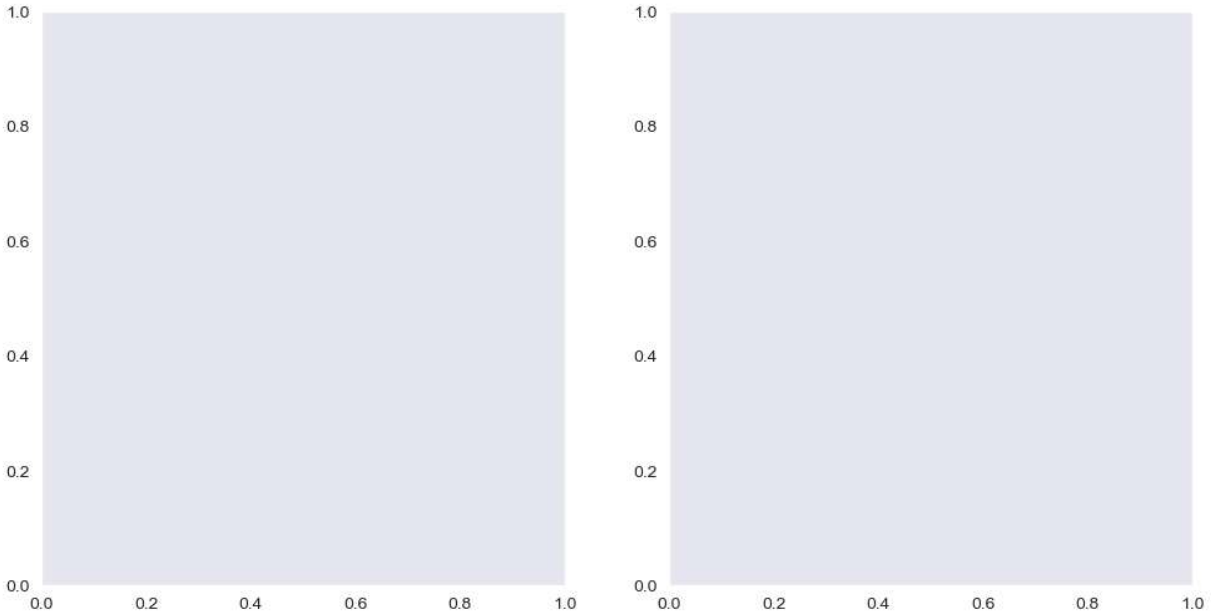
In [53]:
```
sns.set_style('dark')
k1 = sns.kdeplot(data = movies, x = 'BudgetMillions',y
='AudienceRating',shade_lowest=False,cmap='Greens_r')
```



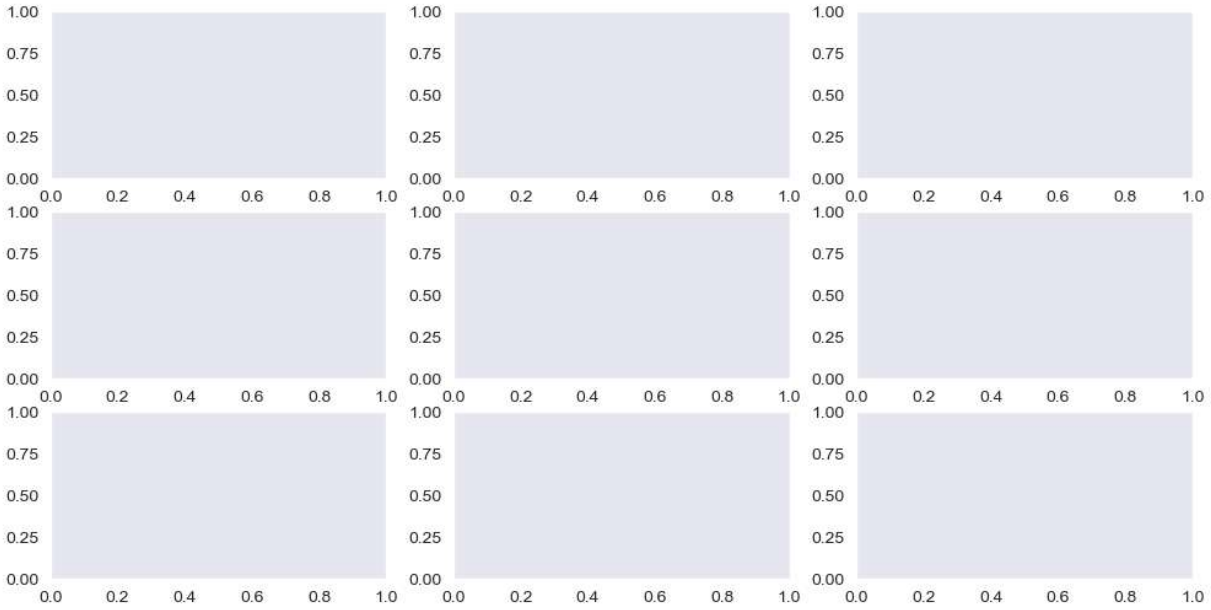
In [54]:
```
sns.set_style('dark')
k1 = sns.kdeplot(data = movies, x = 'BudgetMillions',y ='AudienceRating')
```



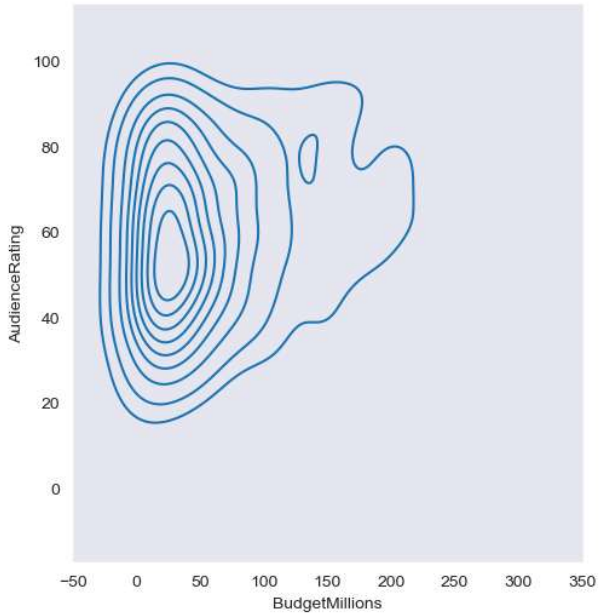
In [55]:
```
#subplots
f, ax = plt.subplots(1,2, figsize =(12,6))
```

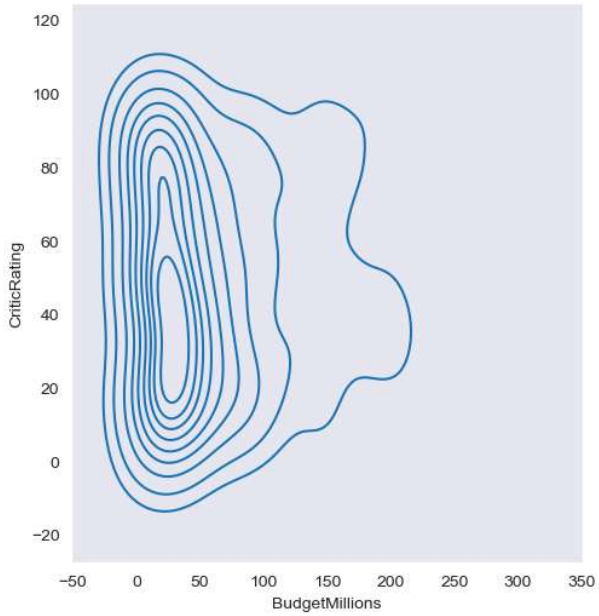


In [56]:
```
f, ax = plt.subplots(3,3, figsize =(12,6))
```



In [57]:
```
f, axes = plt.subplots(1,2, figsize =(12,6))

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

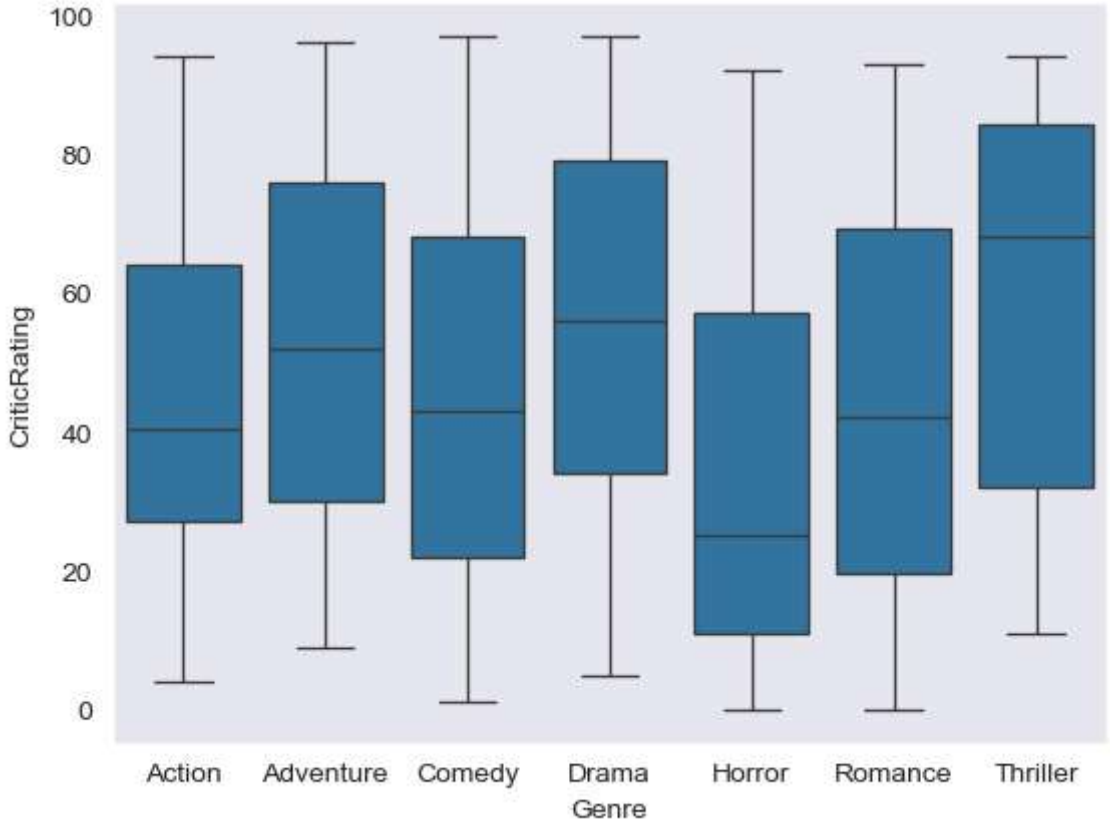




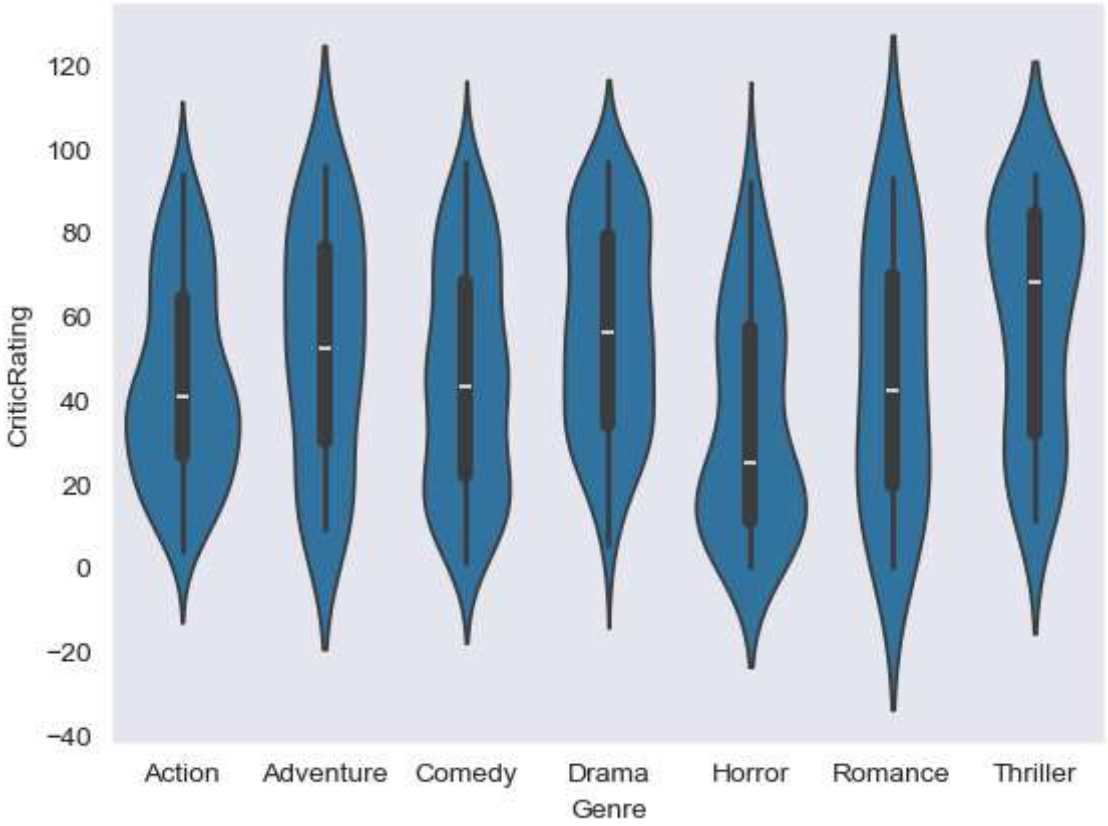
In [58]:
```
axes
```

Out[58]:
```
array([<Axes: xlabel='BudgetMillions', ylabel='AudienceRating'>,
       <Axes: xlabel='BudgetMillions', ylabel='CriticRating'>],
      dtype=object)
```

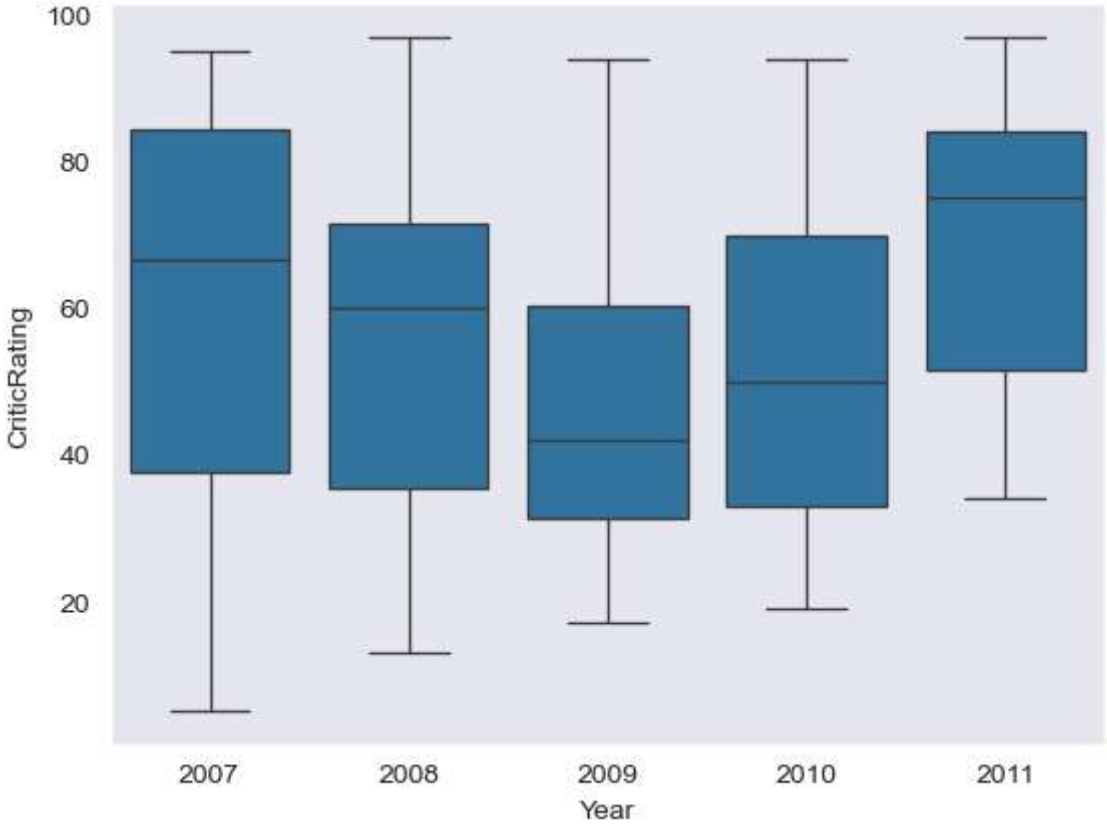
In [59]:
```
#Box plots -

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



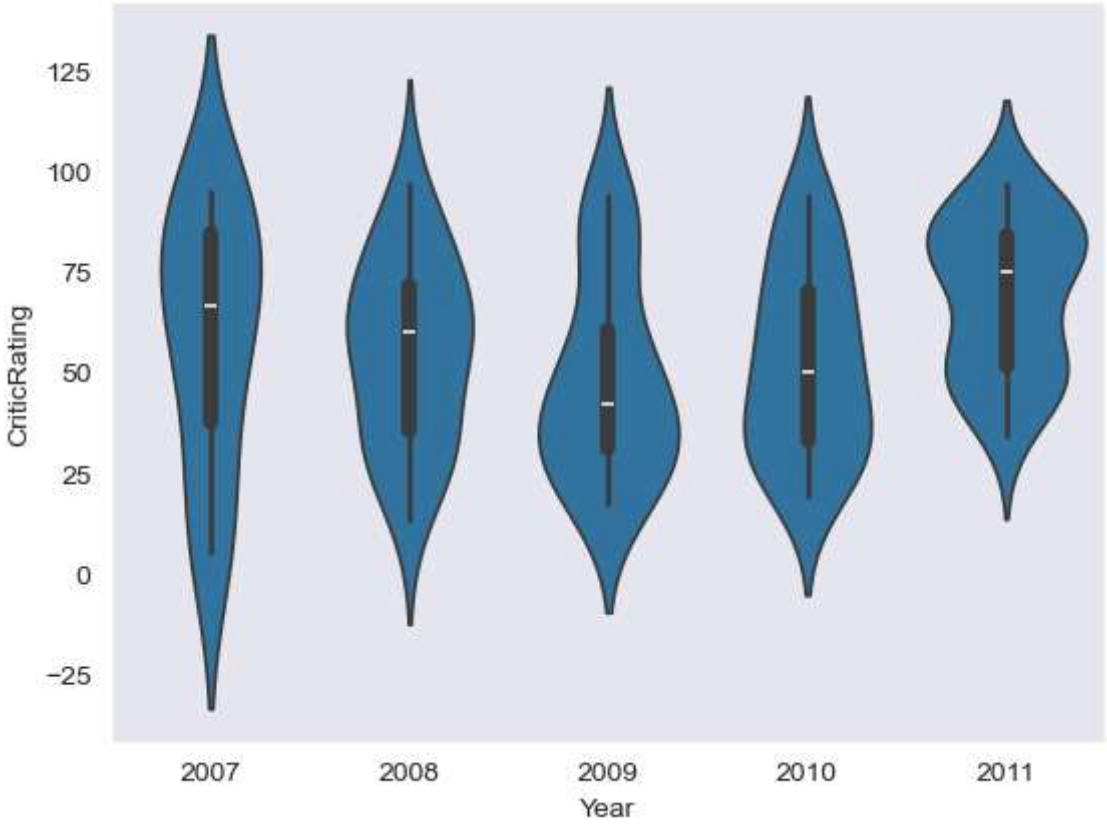
In [60]:
```
#violin plot
z = sns.violinplot(data=movies, x='Genre', y = 'CriticRating')
```



In [61]:
```
w1 = sns.boxplot(data=movies[movies.Genre == 'Drama'], x='Year', y =
'CriticRating')
```



In [62]:
```
z = sns.violinplot(data=movies[movies.Genre == 'Drama'], x='Year', y =
'CriticRating')
```

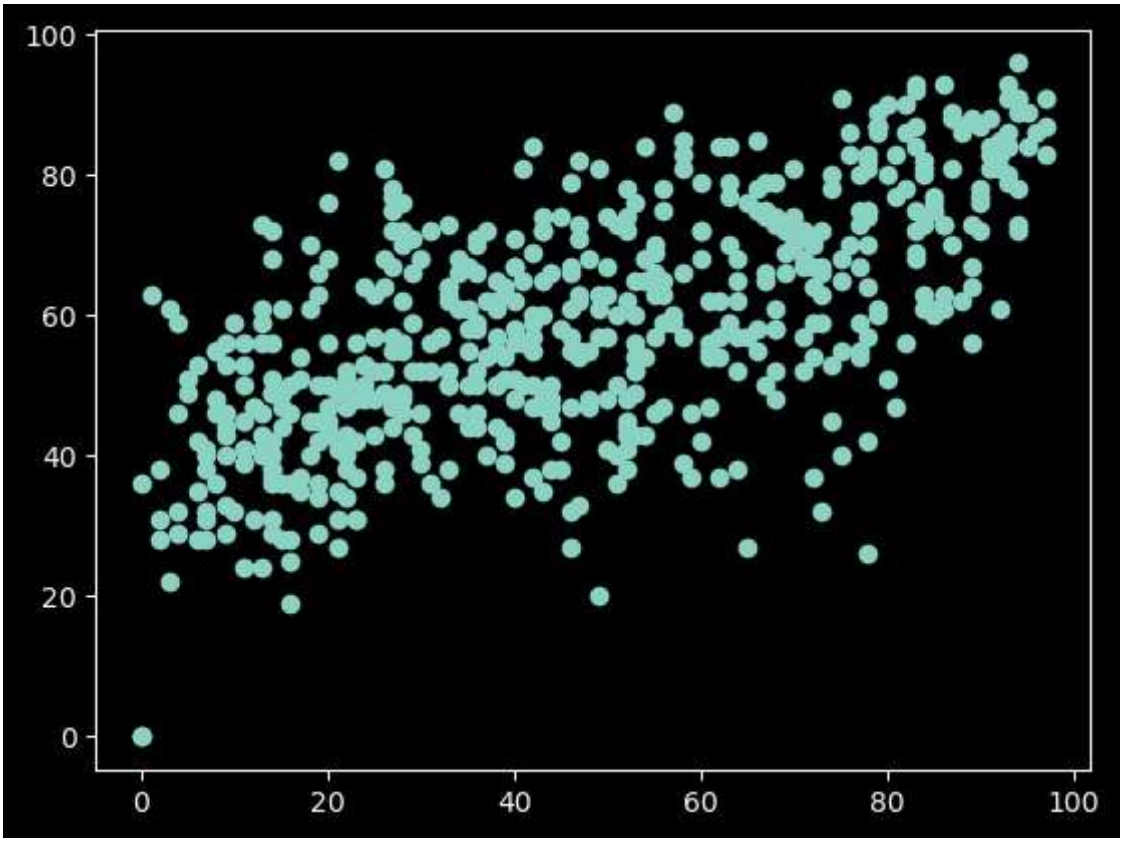


In [63]:
```
# Createing a Facet grid
plt.style.use('default')
```

In [64]:
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre') #kind of
subplots
```

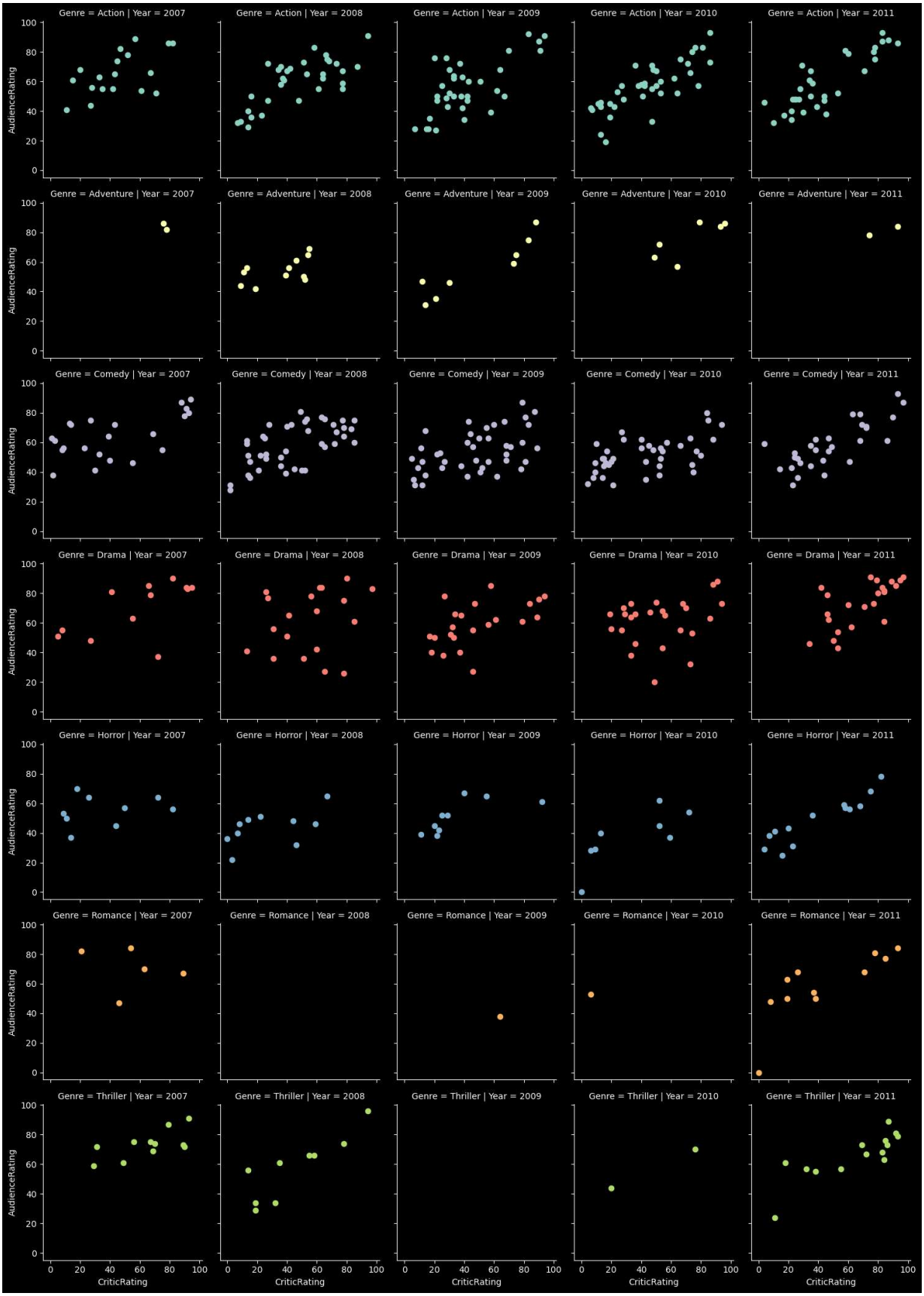
In [65]:
```
plt.style.use('dark_background')
plt.scatter(movies.CriticRating,movies.AudienceRating)
```

Out[65]:
```
<matplotlib.collections.PathCollection at 0x22419642f90>
```



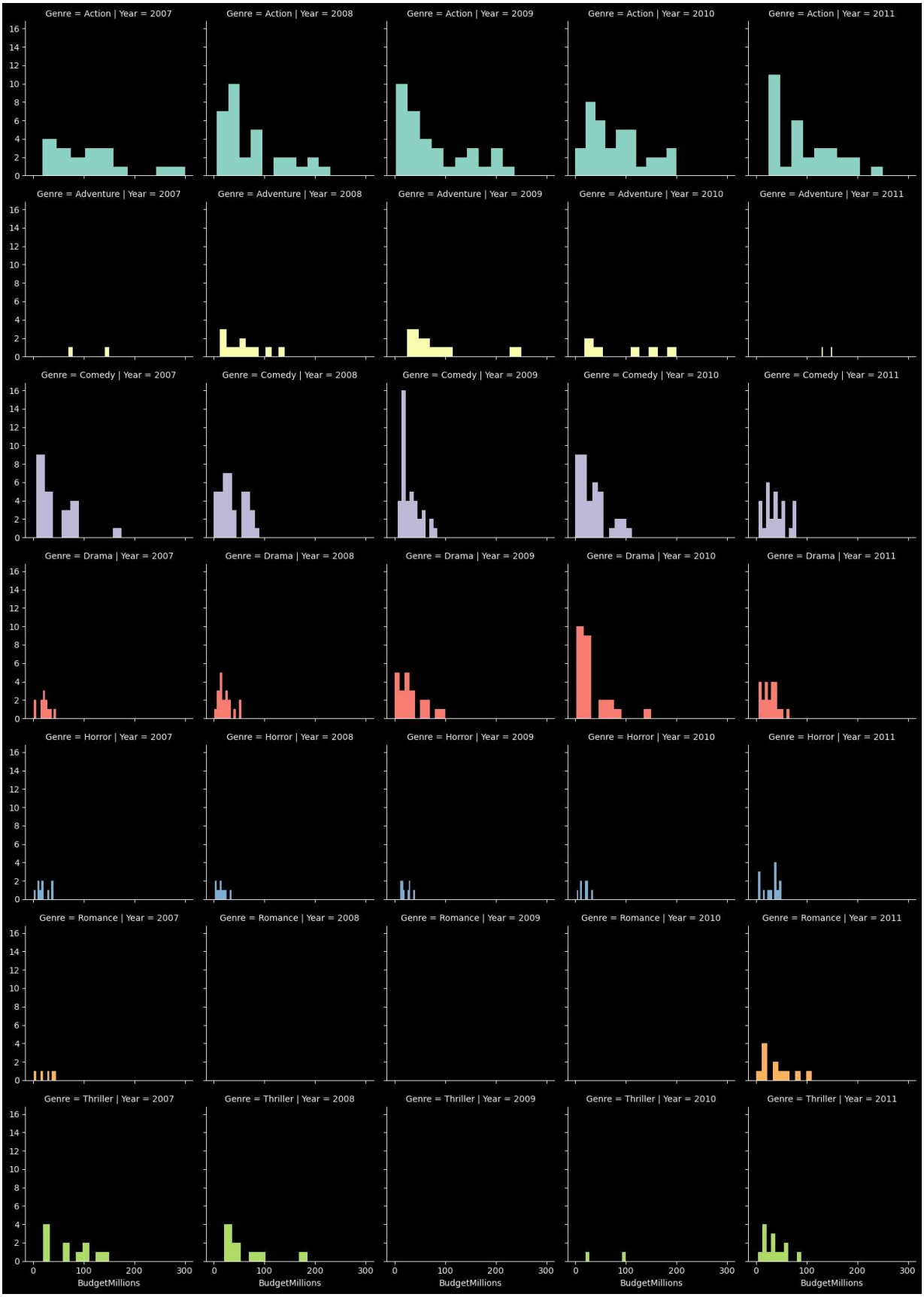
In [67]:

```
plt.style.use('dark_background')
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating' )

# scatterplots are mapped in facetgrid
```



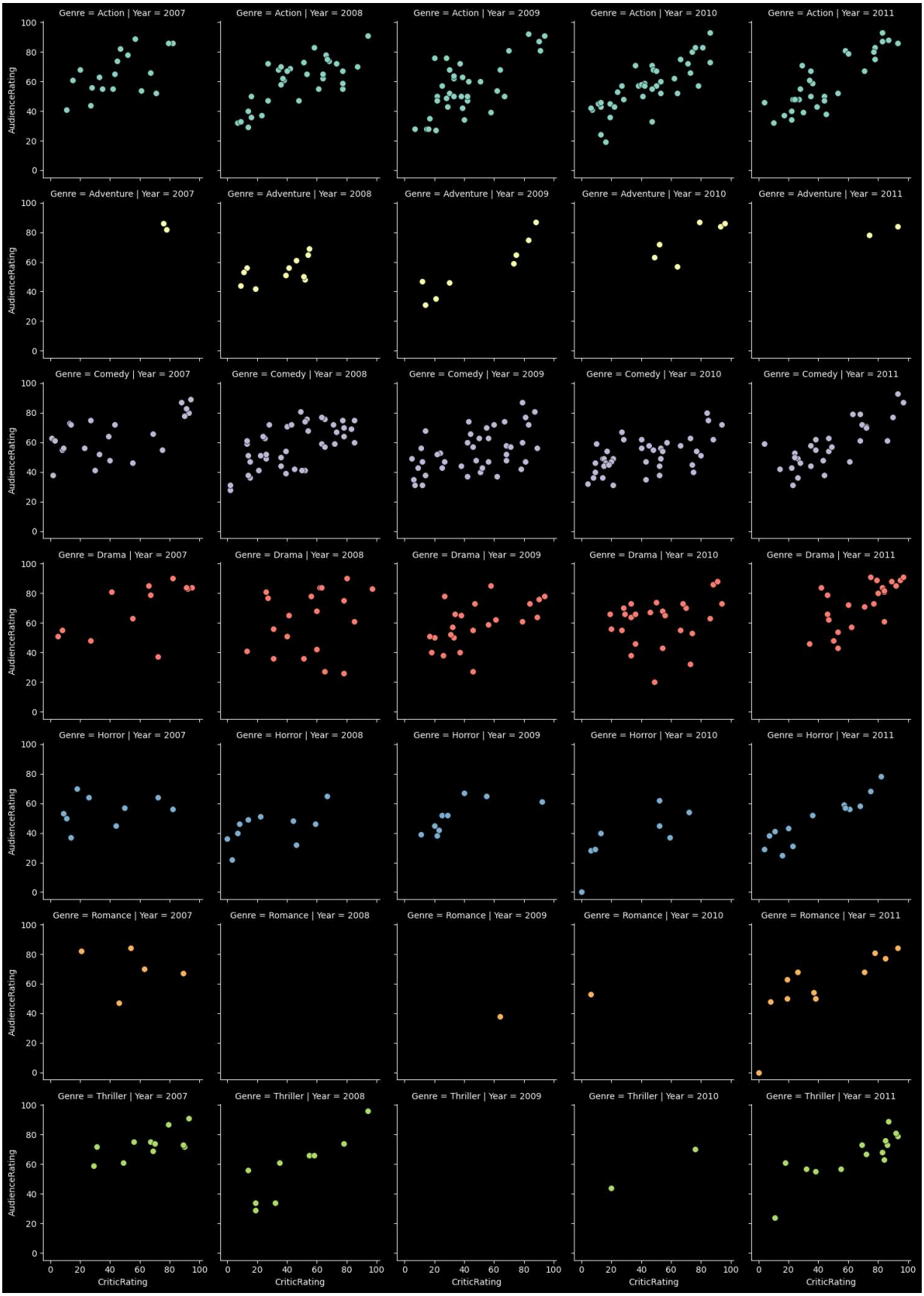
In [69]:

```
# 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
```



In [70]:

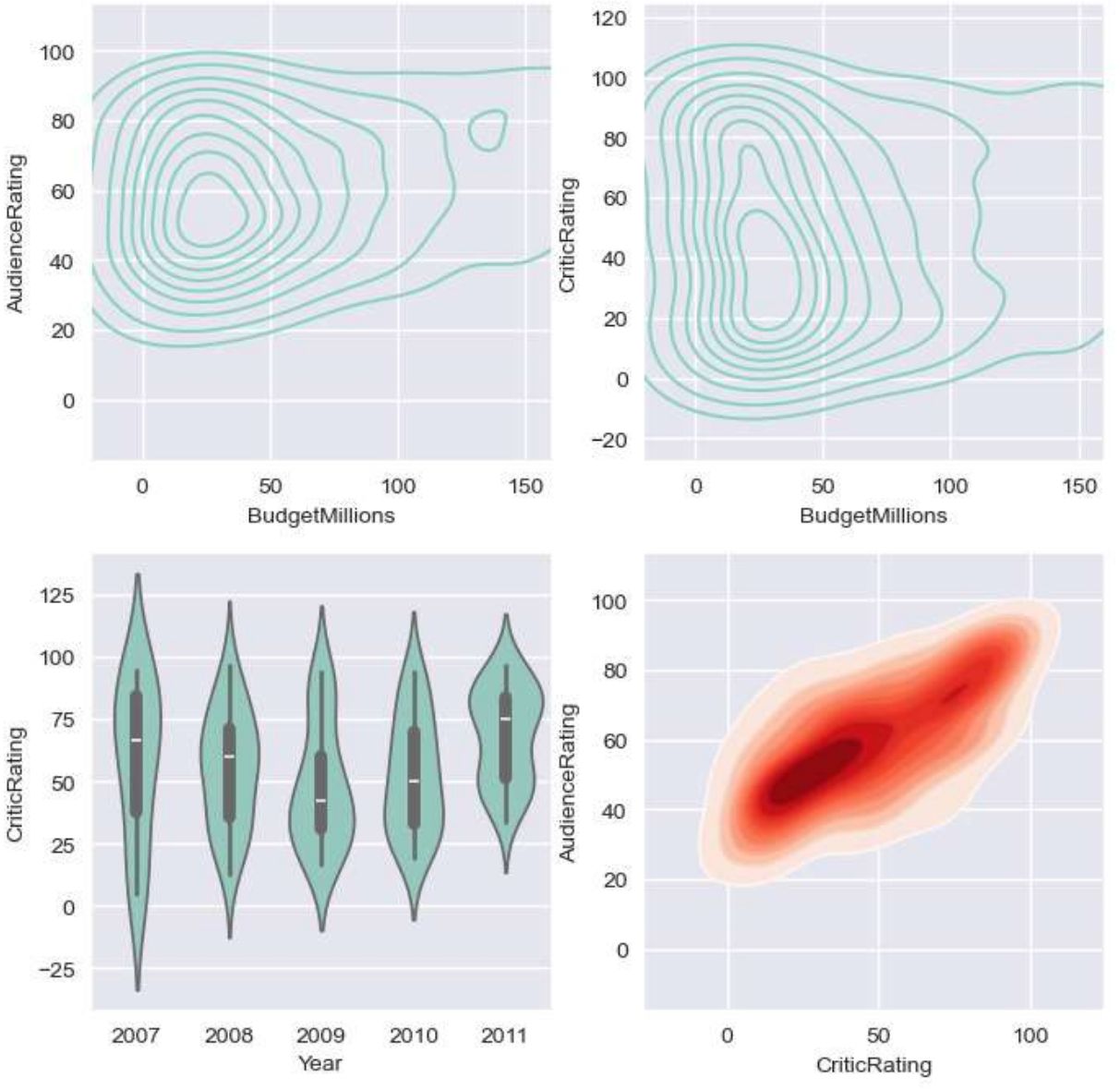
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
kws = dict(s=50, linewidth=0.5,edgecolor='black')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating',**kws ) #scatterplots are
mapped in facetgrid
```



In [72]:

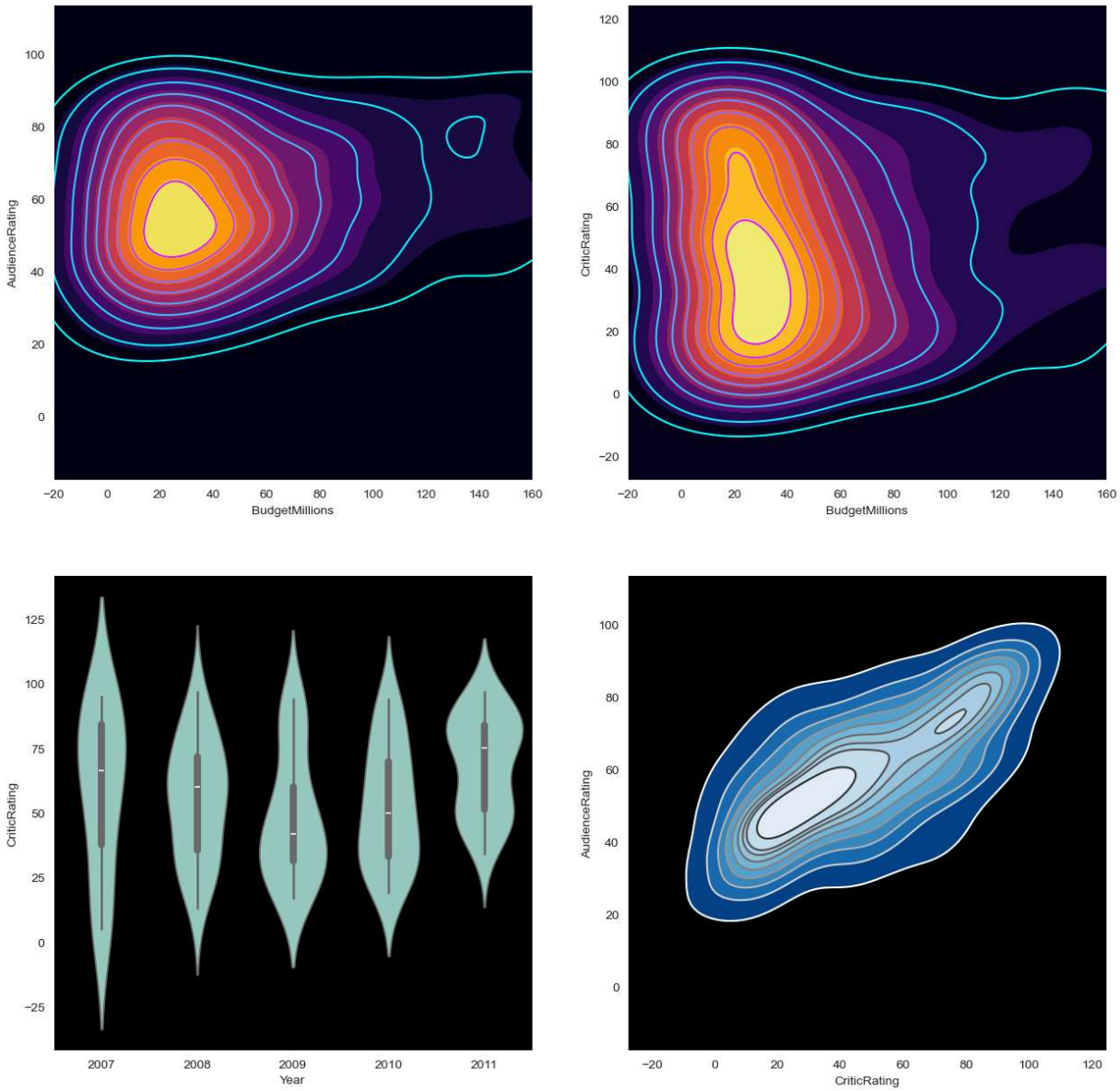
```
# python is not vectorize programming language
# Building dashboards (dashboard - combination of chats)

sns.set_style('darkgrid')
```

```
f, axes = plt.subplots (2,2, figsize = (8,8))

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

k4 = sns.kdeplot(x = movies.CriticRating,y=movies.AudienceRating,shade =
True,shade_lowest=False,cmap='Reds',ax=axes[1,1])

k4b = sns.kdeplot(x = movies.CriticRating,y =
movies.AudienceRating,cmap='Reds',ax = axes[1,1])

plt.show()
```



In [ ]:

In [71]:
```
# How can you style your dashboard  using different color map

# python is not vectorize programming language
# Building dashboards (dashboard - combination of chats)

sns.set_style('dark',{'axes.facecolor':'black'})
f, axes = plt.subplots (2,2, figsize = (15,15))

#plot [0,0]
k1 = sns.kdeplot(x = movies.BudgetMillions,y = movies.AudienceRating, \
                 shade = True, shade_lowest=True,cmap = 'inferno', \
                 ax = axes[0,0])
k1b = sns.kdeplot(x= movies.BudgetMillions,y =  movies.AudienceRating, \
                 cmap = 'cool',ax = axes[0,0])

#plot [0,1]
k2 = sns.kdeplot(x= movies.BudgetMillions,y=movies.CriticRating,\
                 shade=True, shade_lowest=True, cmap='inferno',\
                 ax = axes[0,1])
k2b = sns.kdeplot(x=movies.BudgetMillions,y=movies.CriticRating,\
                  cmap = 'cool', ax = axes[0,1])

#plot[1,0]
z = sns.violinplot(data=movies[movies.Genre=='Drama'], \
                   x='Year', y = 'CriticRating', ax=axes[1,0])

#plot[1,1]
k4 = sns.kdeplot(x = movies.CriticRating,y =movies.AudienceRating, \
                 shade = True,shade_lowest=False,cmap='Blues_r', \
                 ax=axes[1,1])

k4b = sns.kdeplot(x = movies.CriticRating,y = movies.AudienceRating, \
                  cmap='gist_gray_r',ax = axes[1,1])


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

plt.show()
```



Final discussion what we learn so far - 1> category datatype in python 2> jointplots 3> histogram 4> stacked histograms 5> Kde plot 6> subplot 7> violin plots 8> Factet grid 9> Building dashboards

EDA COMPLETED

In [ ]: