Movie Rating Analytics (Advance Visualisation)

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

In [2]:
```
data = pd.read_csv(r'C:\Users\hp\OneDrive\Documents\Desktop\Movie-Rating.csv')
```

In [3]:
```
data
```

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]:
```
len(data)
```

Out[4]: 559

In [5]:
```
data.head()
```

Out[5]:

	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 [6]:
```
data.tail()
```

Out[6]:

	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 [7]:
```
data.columns
```

Out[7]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

In [8]:
```
data.columns=['Film', 'Genre', 'criticrating', 'AudienceRating', 'Budgetmillions', 'Year']
```

In [9]:
```
data
```

Out[9]:

	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
...	...	...	...	...	...	...
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]:
```
data.head() # Removed spaces & % removed noise characters
```

Out[10]:

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

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

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

In [12]:
```
data.describe()
#if you look at the year the data type is int but when you look at the mean value it showing 2009 which is meaningless
# we have to change to categroy type
# also from object datatype we will convert to category datatypes
#
```

Out[12]:

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

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

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

In [14]:
```
data.Film
```

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

In [15]:
```
data.Film = data.Film.astype("category")
```

In [16]:
```
data.Film
```

Out[16]:
```
0      (500) Days of Summer
1                10,000 B.C.
2                 12 Rounds
3                   127 Hours
4                   17 Again
                ...
554            Your Highness
555          Youth in Revolt
556                   Zodiac
557              Zombieland
558                Zookeeper
Name: Film, Length: 559, dtype: category
Categories (559, object): ['(500) Days of Summer ', '10,000 B.C.', '12 Rounds ', '127 Hours', ..., 'Youth in Revolt', 'Zodia
c', 'Zombieland ', 'Zookeeper']
```

In [17]:
```
data.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]:
```
data.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 [19]:
```
data.Genre = data.Genre.astype("category")
data.Year=data.Year.astype("category")
```

In [20]:
```
data.Year # is it real no. year you can take average,min,max but out come have no meaning
```

Out[20]:
```
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 [21]:
```
data.Genre.cat.categories
```

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

In [22]:
```
data.describe()
#now when you see the describt you will get only integer value mean, standard deviation which is meaning full
```

Out[22]:

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

In [23]:
```
# How to working with joint plots
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [24]:
```
movies = data.copy()
```

In [26]:
```
movies
```

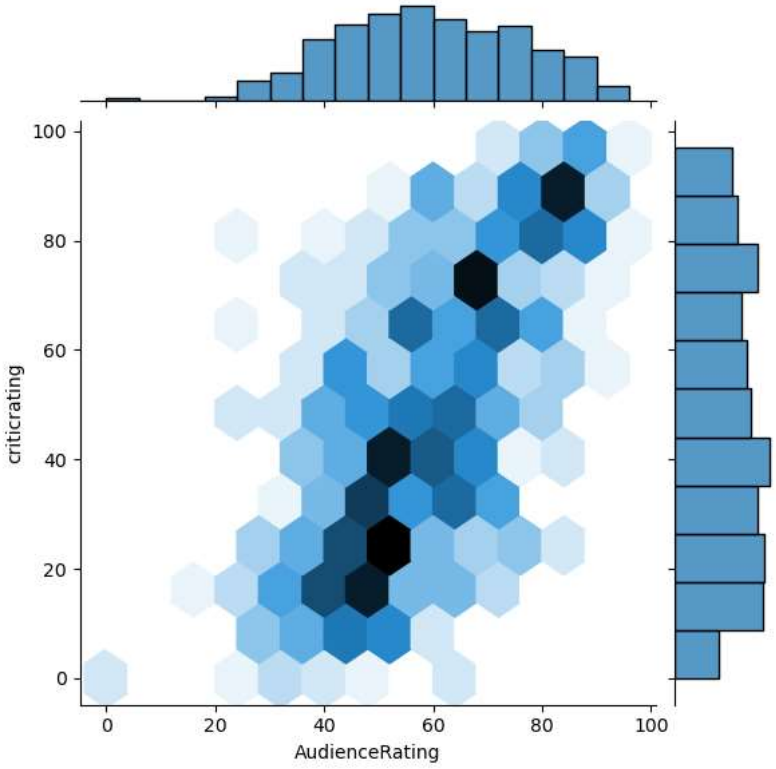
Out[26]:

	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
...	...	...	...	...	...	...
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

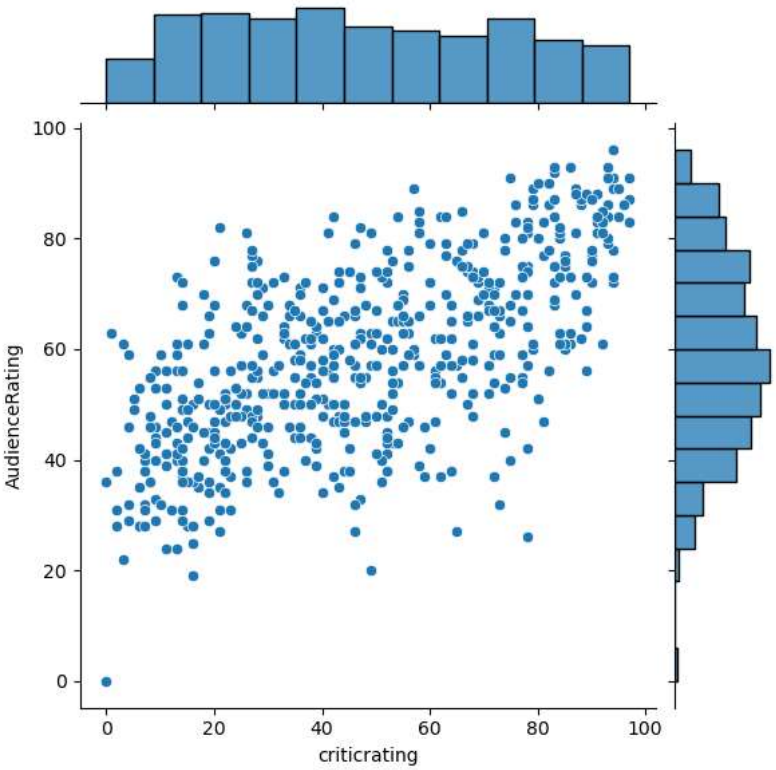
559 rows × 6 columns

- basically joint plot is a scatter plot & it find the relation b/w audiene & critics
- also if you look up you can find the uniform distribution (critics)and normal distriution (audience)

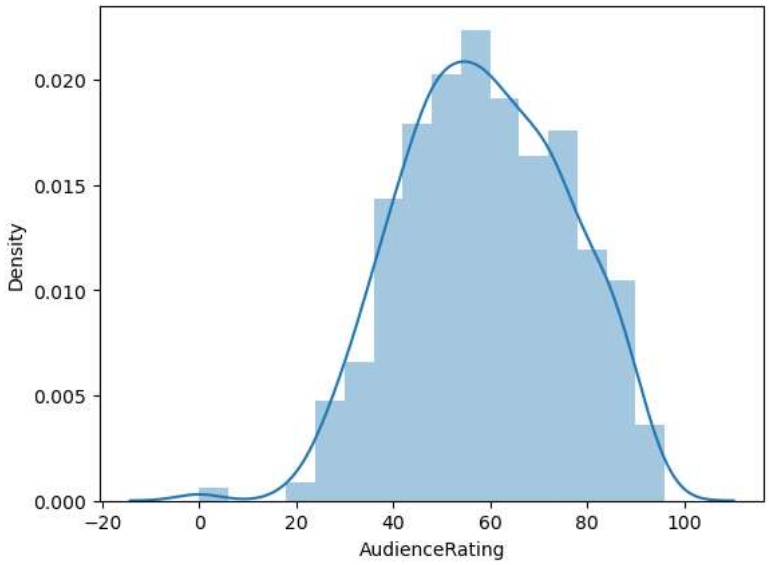
In [28]:
```
a = sns.jointplot( data = movies, x = 'AudienceRating', y ='criticrating',kind='hex')
# Audience rating is more dominant then critics rating
# Based on this we find out as most people are most liklihood to watch audience rating & less likely to wathc critics rating
# let me explain the excel - if you filter audience rating & critic rating. critic rating has very low values compare to audi
```



In [31]:
```
j = sns.jointplot( data = movies, x = 'criticrating', y = 'AudienceRating')
```

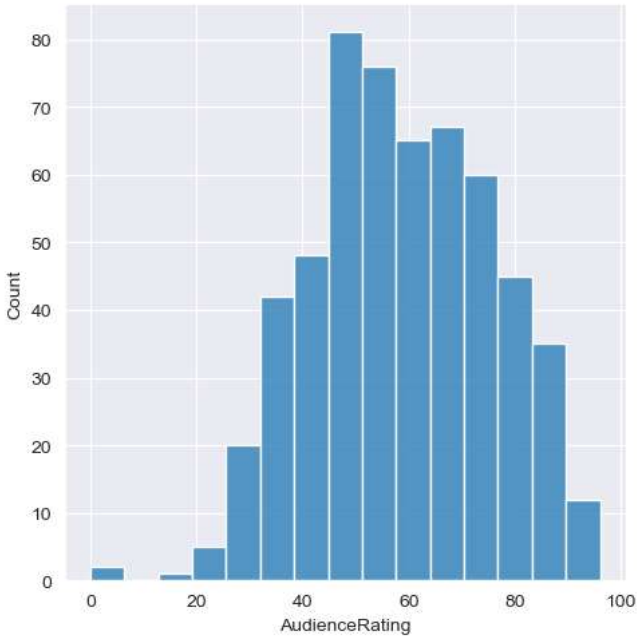


In [33]:
```
# Histogram
#<< chat1
m1 = sns.distplot(movies.AudienceRating)
#y - axis generated by seaborn automatically that is the powefull of seaborn gallery
```

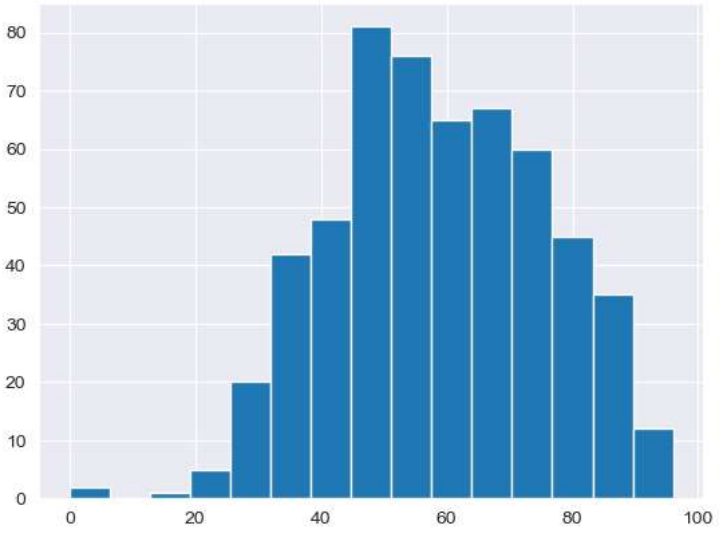


In [34]:
```
1 sns.set_style("darkgrid")
```

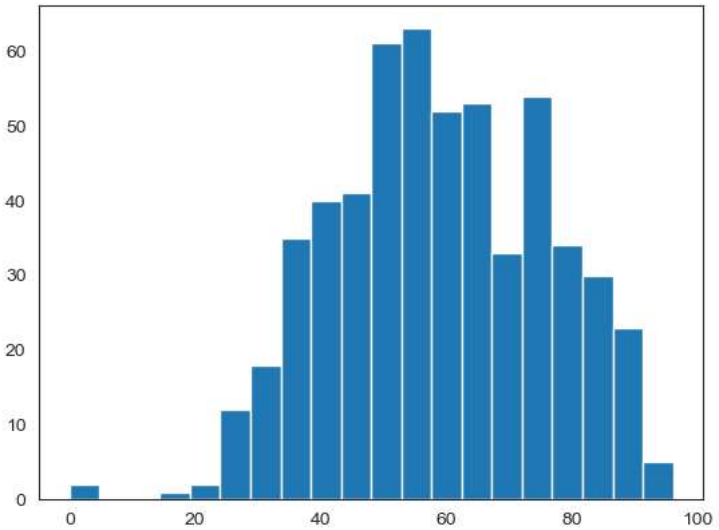
In [36]:
```
m2 = sns.displot(movies.AudienceRating, bins = 15)
```



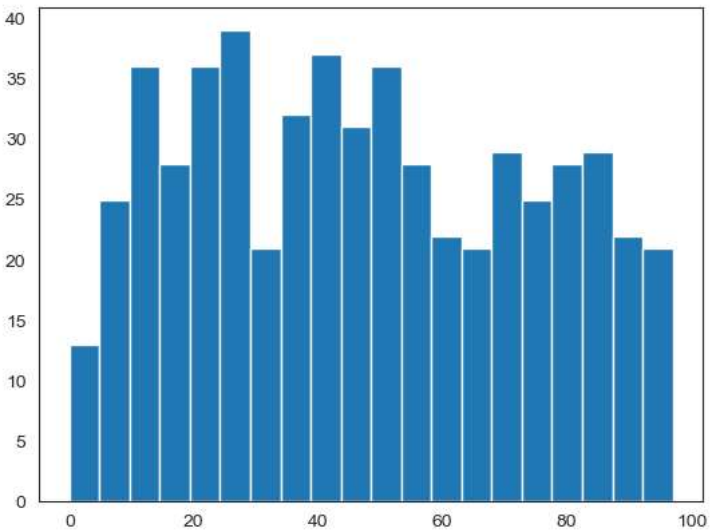
In [37]:
```
#sns.set_style("darkgrid")
n1 = plt.hist(movies.AudienceRating, bins=15)
```



In [40]:
```
sns.set_style("white")# normal distribution & called as bell curve
n1 = plt.hist(movies.AudienceRating, bins=20)
```



In [41]:
```
n1= plt.hist(movies.criticrating,bins=20)# uniform distribution
```



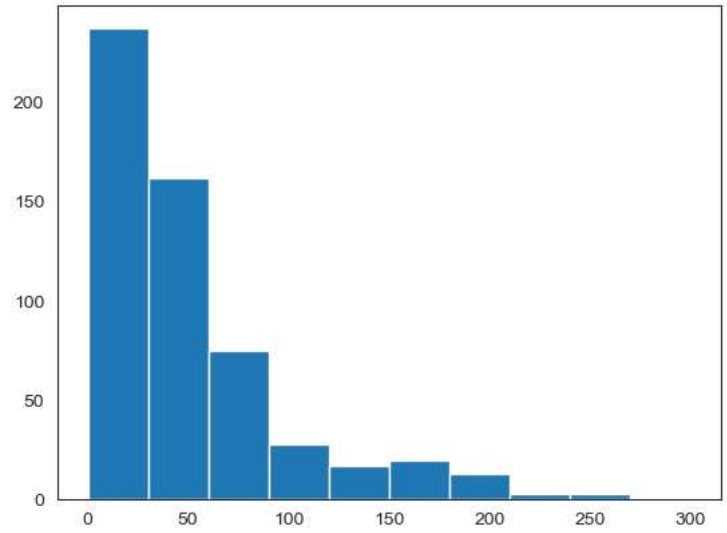
In [42]:
```
# <<< chat - 2
# Creating stacked histograms & this is bit tough to understand
```

In [45]:
```
movies.head(1)
```

Out[45]:

	Film	Genre	criticrating	AudienceRating	Budgetmillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009

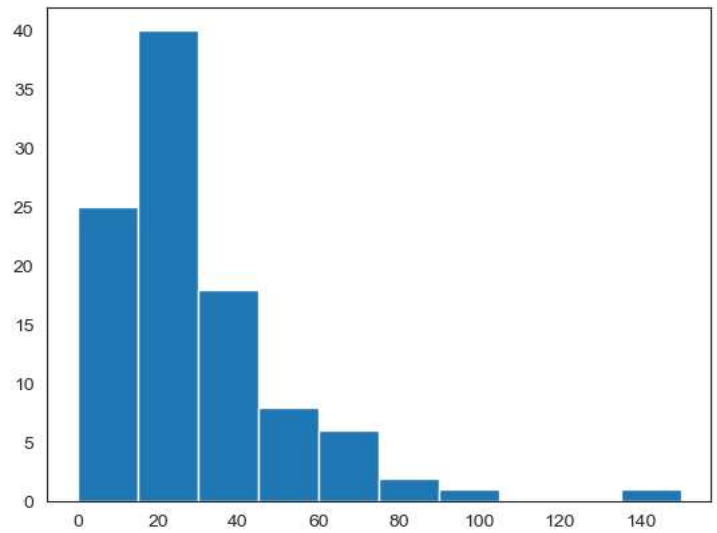
In [46]:
```
#h1 = plt.hist(movies.BudgetMillions)

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



In [49]:
```
movies.Year.cat.categories
```

Out[49]:
```
Int64Index([2007, 2008, 2009, 2010, 2011], dtype='int64')
```

In [50]:
```
plt.hist(movies[movies.Genre == "Drama"].Budgetmillions)
plt.show()
```



In [51]:
```
movies.head()
```

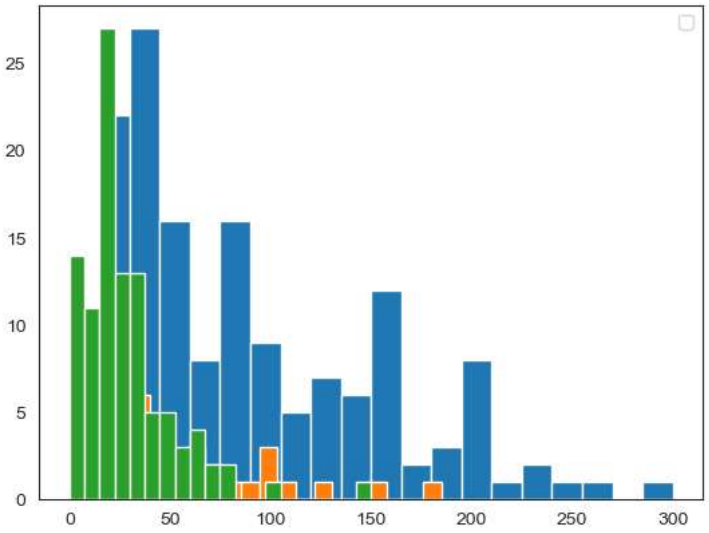
Out[51]:

	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

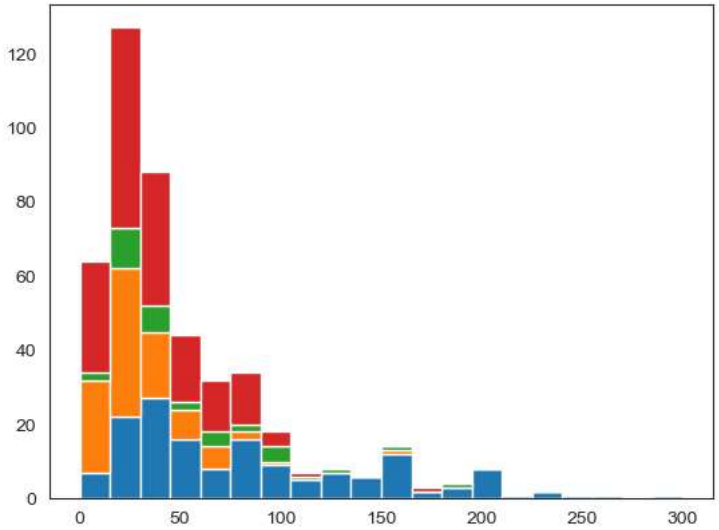
In [52]:
```
#movies.genre.unique()
```

In [56]:
```
# blow plot are stacked histogram because 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()
```

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



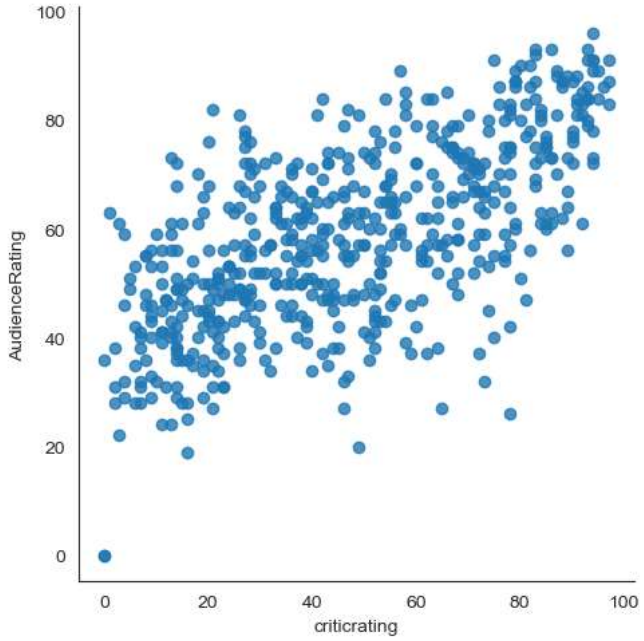
In [59]:
```
plt.hist([movies[movies.Genre == "Action"].Budgetmillions,\
          movies[movies.Genre == "Drama"].Budgetmillions,\
          movies[movies.Genre == "Thriller"].Budgetmillions,\
          movies[movies.Genre == "Comedy"].Budgetmillions],
          bins = 20, stacked = True)
plt.show()
```



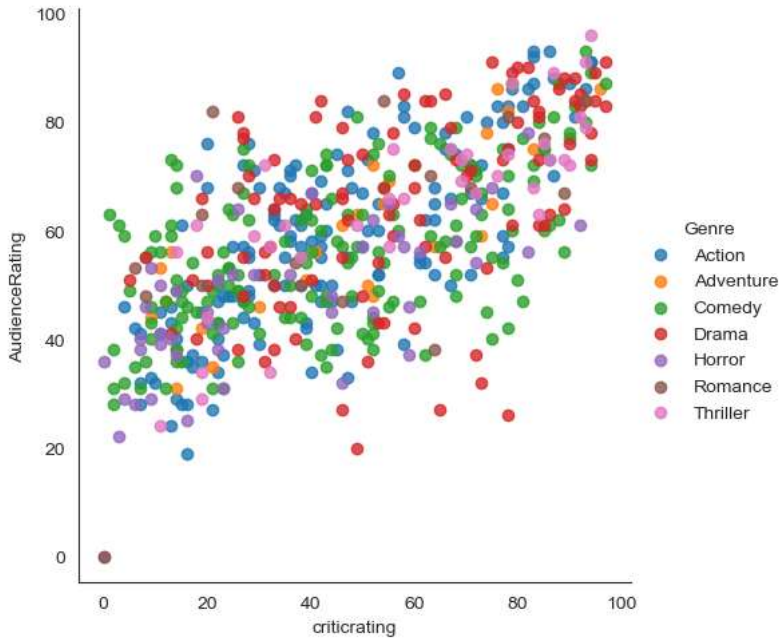
In [61]:
```
# # if you have 100 categories you cannot copy & paste all the things
for gen in movies.Genre.cat.categories:
    print(gen)
```

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

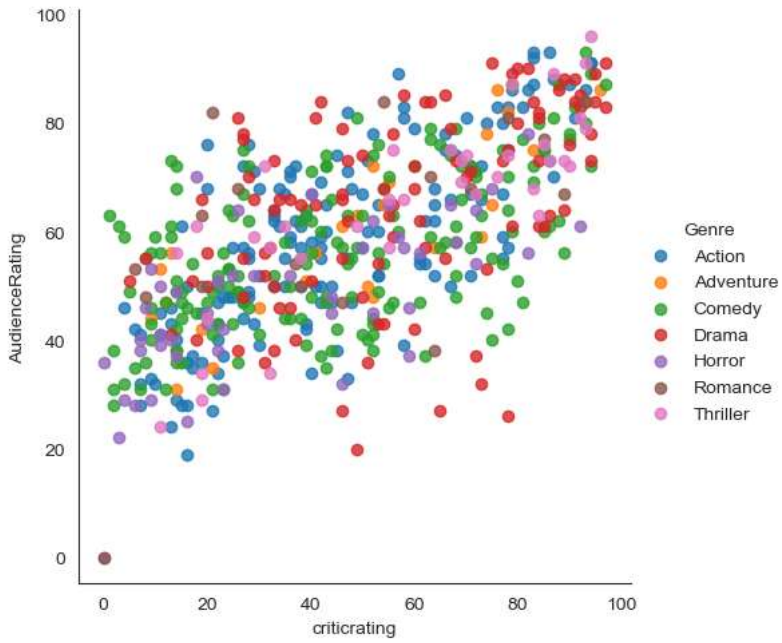
In [63]:
```
vis1 = sns.lmplot(data=movies, x="criticrating", y="AudienceRating",\
                  fit_reg=False)
```



In [64]:
```
vis1 = sns.lmplot(data=movies, x="criticrating", y="AudienceRating",\
                  fit_reg=False, hue = "Genre")
```

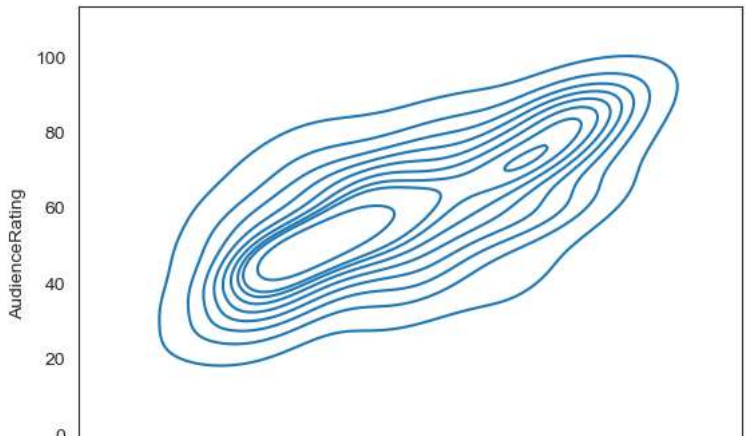


In [71]:
```
vis1 = sns.lmplot(data=movies, x="criticrating", y="AudienceRating",fit_reg=False,hue = "Genre",aspect=1)
```

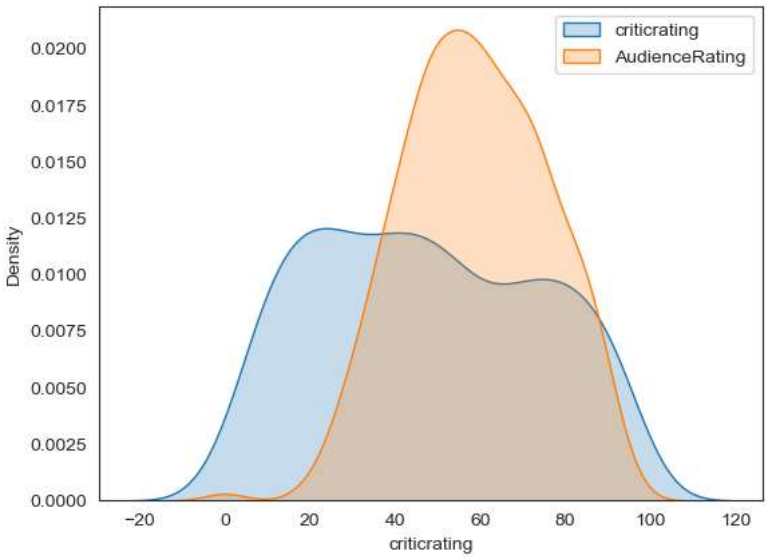


In [72]:
```
# Kernal Density Estimate plot ( KDE PLOT)
# how can i visulize audience rating & critics rating . using scatterplot
```

In [91]:
```
k1 = sns.kdeplot(x= movies.criticrating,y=movies.AudienceRating)
```



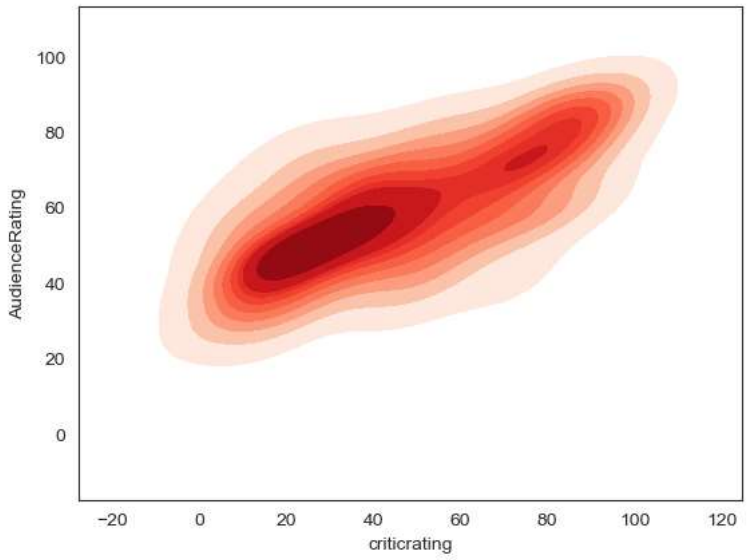
In [86]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'movies' is the DataFrame containing the data

# Plot KDE for 'CriticRating'
sns.kdeplot(data=movies, x='criticrating', fill=True, label='criticrating')

# Plot KDE for 'AudienceRating'
sns.kdeplot(data=movies, x='AudienceRating', fill=True, label='AudienceRating')

# Display the legend
plt.legend()

# Show the plot
plt.show()
```



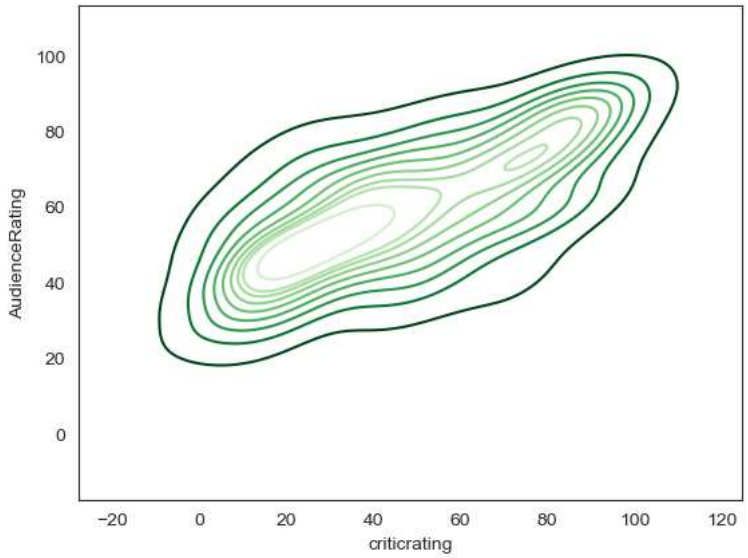
In [87]:
```
pip install seaborn --upgrade
```

```
Requirement already satisfied: seaborn in e:\anaconda3\lib\site-packages (0.12.2)Note: you may need to restart the kernel to
use updated packages.

Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in e:\anaconda3\lib\site-packages (from seaborn) (3.7.0)
Requirement already satisfied: pandas>=0.25 in e:\anaconda3\lib\site-packages (from seaborn) (1.5.3)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in e:\anaconda3\lib\site-packages (from seaborn) (1.23.5)
Requirement already satisfied: python-dateutil>=2.7 in e:\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seabor
n) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in e:\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.
0)
Requirement already satisfied: fonttools>=4.22.0 in e:\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(4.25.0)
Requirement already satisfied: contourpy>=1.0.1 in e:\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(1.0.5)
Requirement already satisfied: kiwisolver>=1.0.1 in e:\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(1.4.4)
Requirement already satisfied: cycler>=0.10 in e:\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.
0)
Requirement already satisfied: pyparsing>=2.3.1 in e:\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(3.0.9)
Requirement already satisfied: packaging>=20.0 in e:\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2
2.0)
Requirement already satisfied: pytz>=2020.1 in e:\anaconda3\lib\site-packages (from pandas>=0.25->seaborn) (2022.7)
Requirement already satisfied: six>=1.5 in e:\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.
1->seaborn) (1.16.0)
```

In [92]:
```
k1 = sns.kdeplot(x=movies.criticrating,y=movies.AudienceRating,shade = True,shade_lowest=False,cmap="Reds")
```



In [97]:
```
k2 = sns.kdeplot(x=movies.criticrating,y=movies.AudienceRating,shade = False,shade_lowest=False,cmap="Greens_r")
```

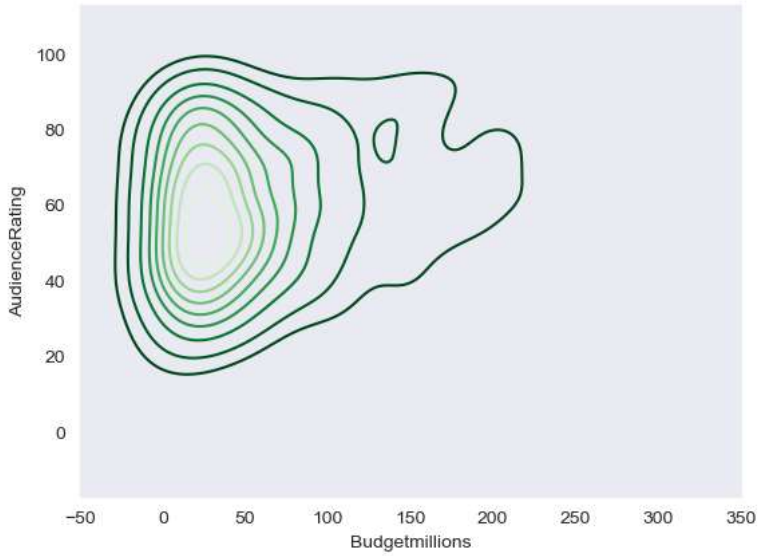


In [99]:
```
movies.head(1)
```

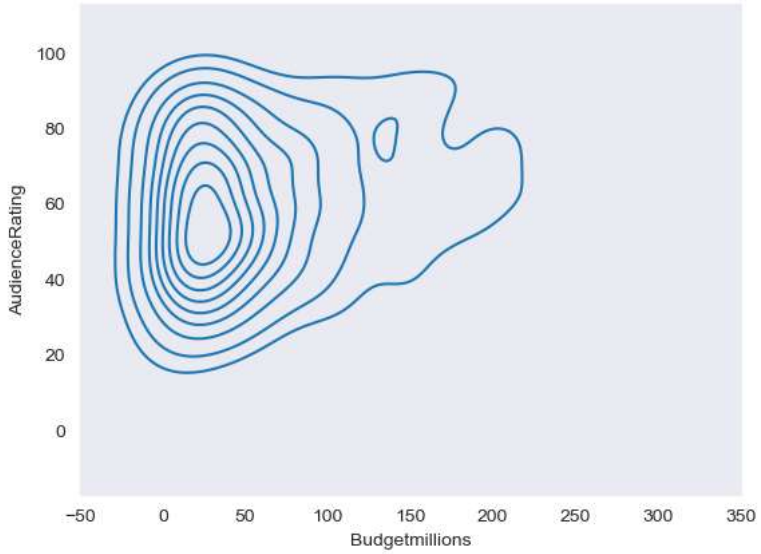
Out[99]:

	Film	Genre	criticrating	AudienceRating	Budgetmillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009

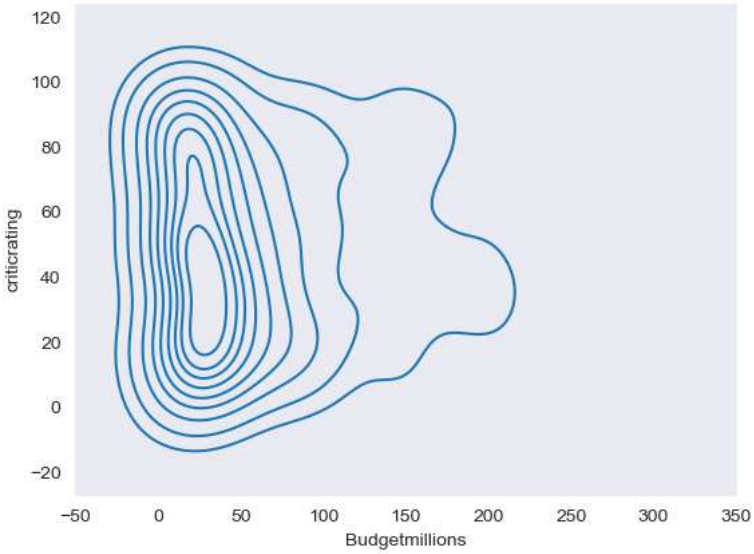
In [100]:
```
sns.set_style("dark")
k2 = sns.kdeplot(x=movies.Budgetmillions,y=movies.AudienceRating,shade = False,shade_lowest=False,cmap="Greens_r")
```



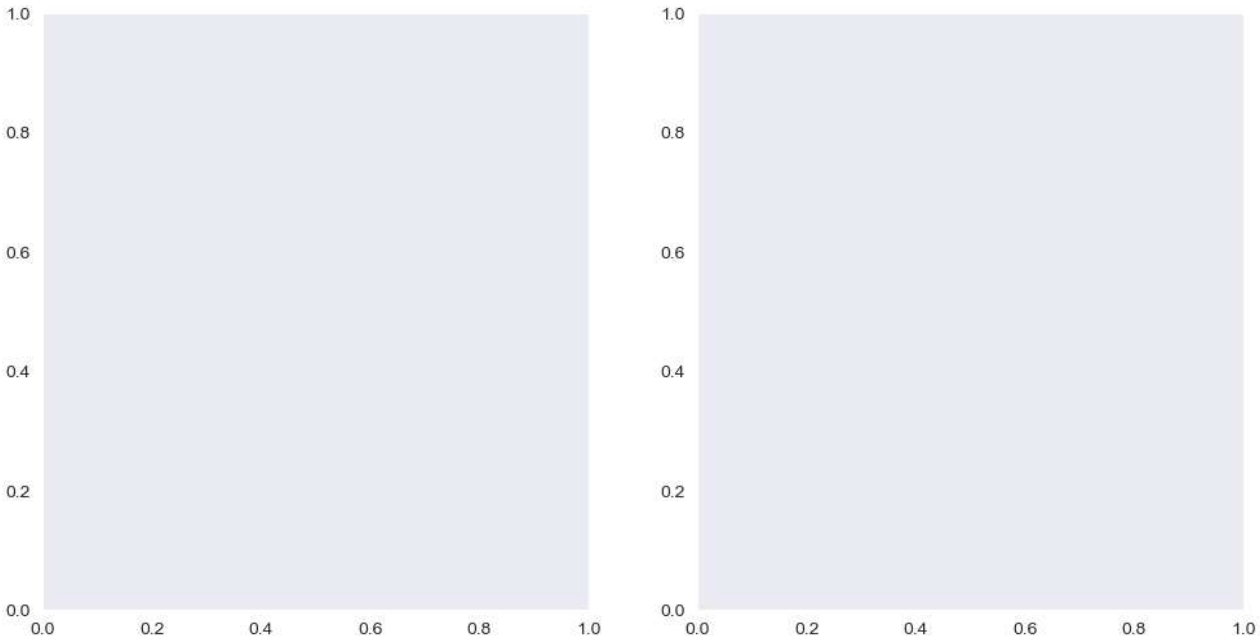
In [101]:
```
sns.set_style("dark")
k2 = sns.kdeplot(x=movies.Budgetmillions,y=movies.AudienceRating)
```



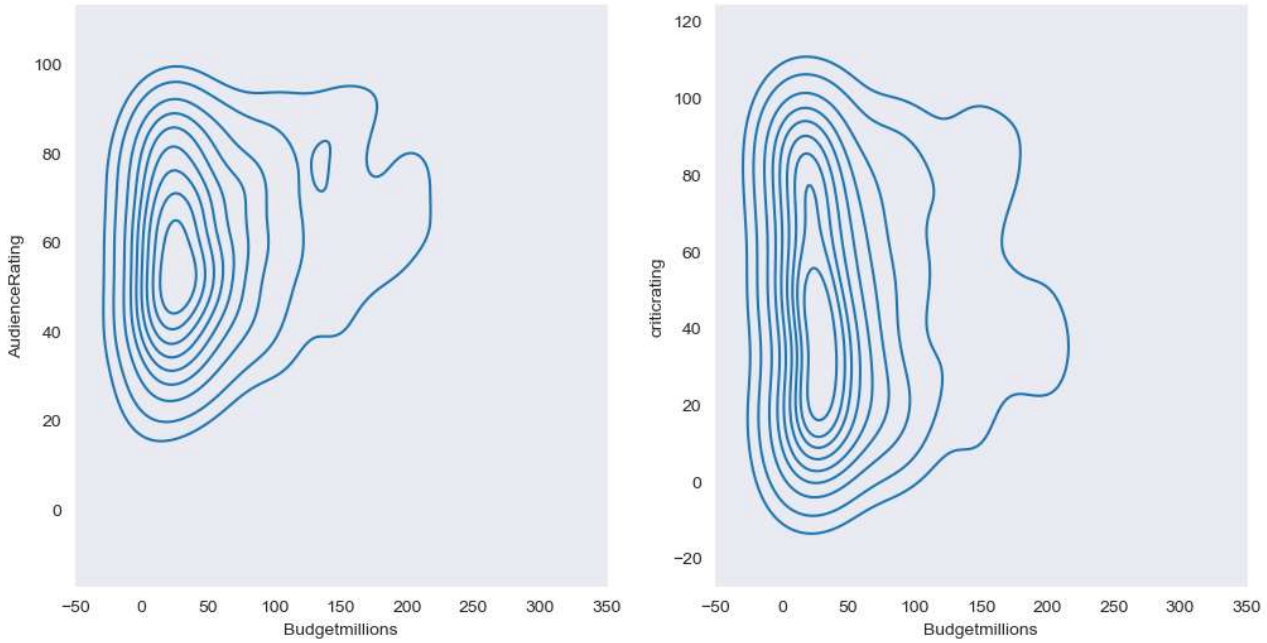
In [102]:
```
k2 = sns.kdeplot(x=movies.Budgetmillions,y=movies.criticrating)
```



In [110]:
```
#subplots
f, ax = plt.subplots(1,2, figsize = (12,6))
#f, ax = plt.subplot(3,3, figsize = (12,6))
#
```



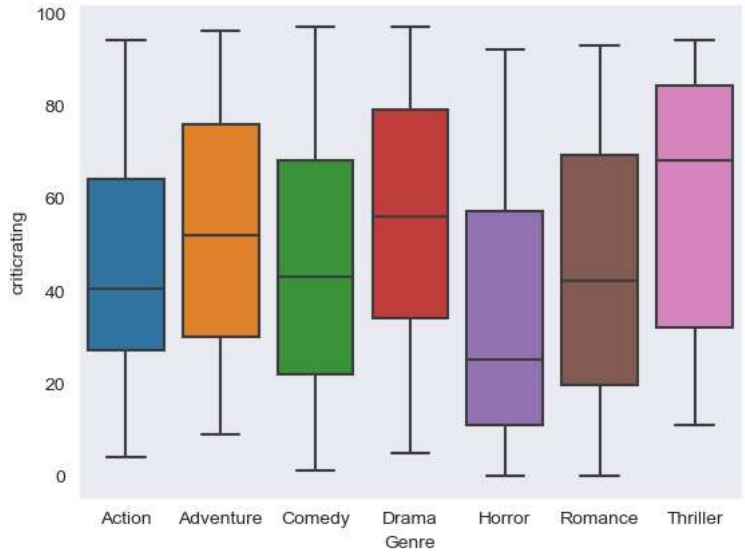
In [111]:
```
f, axes = plt.subplots(1,2, figsize = (12,6))

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



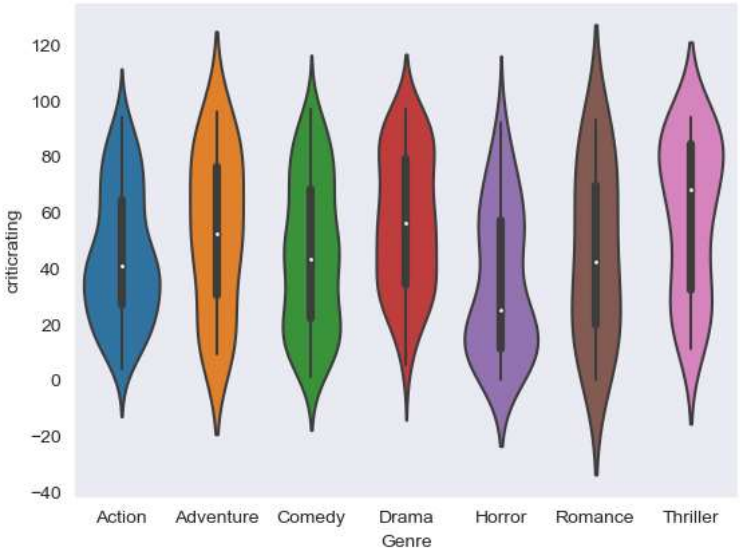
In [112]:
```
axes
```

Out[112]:
```
array([<Axes: xlabel='Budgetmillions', ylabel='AudienceRating'>,
       <Axes: xlabel='Budgetmillions', ylabel='criticrating'>],
      dtype=object)
```

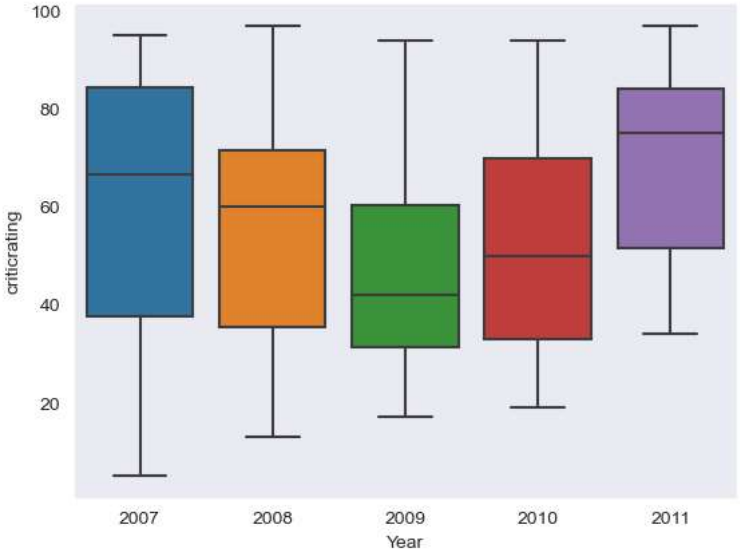
In [114]:
```
#box plot -

w = sns.boxplot(data=movies, x="Genre", y="criticrating")
```



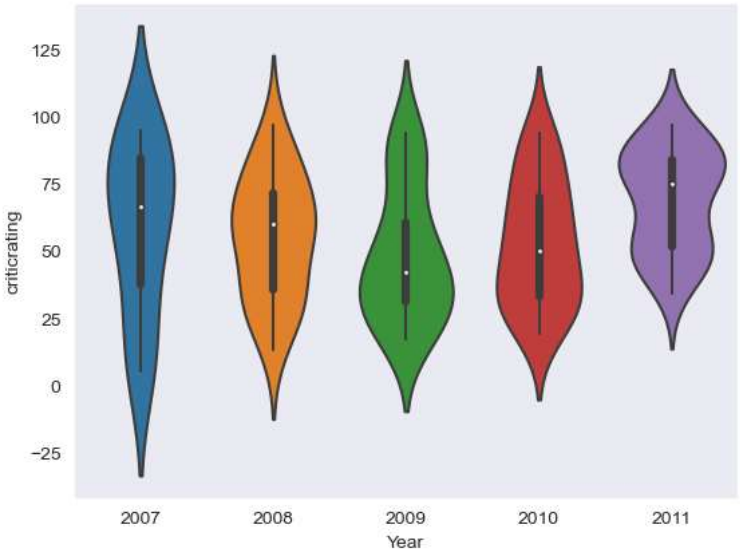
In [115]:
```
#violin plot

z = sns.violinplot(data=movies, x="Genre", y = "criticrating")
```



In [116]:
```
w1 = sns.boxplot(data=movies[movies.Genre == "Drama"], x="Year", y = "criticrating")
```

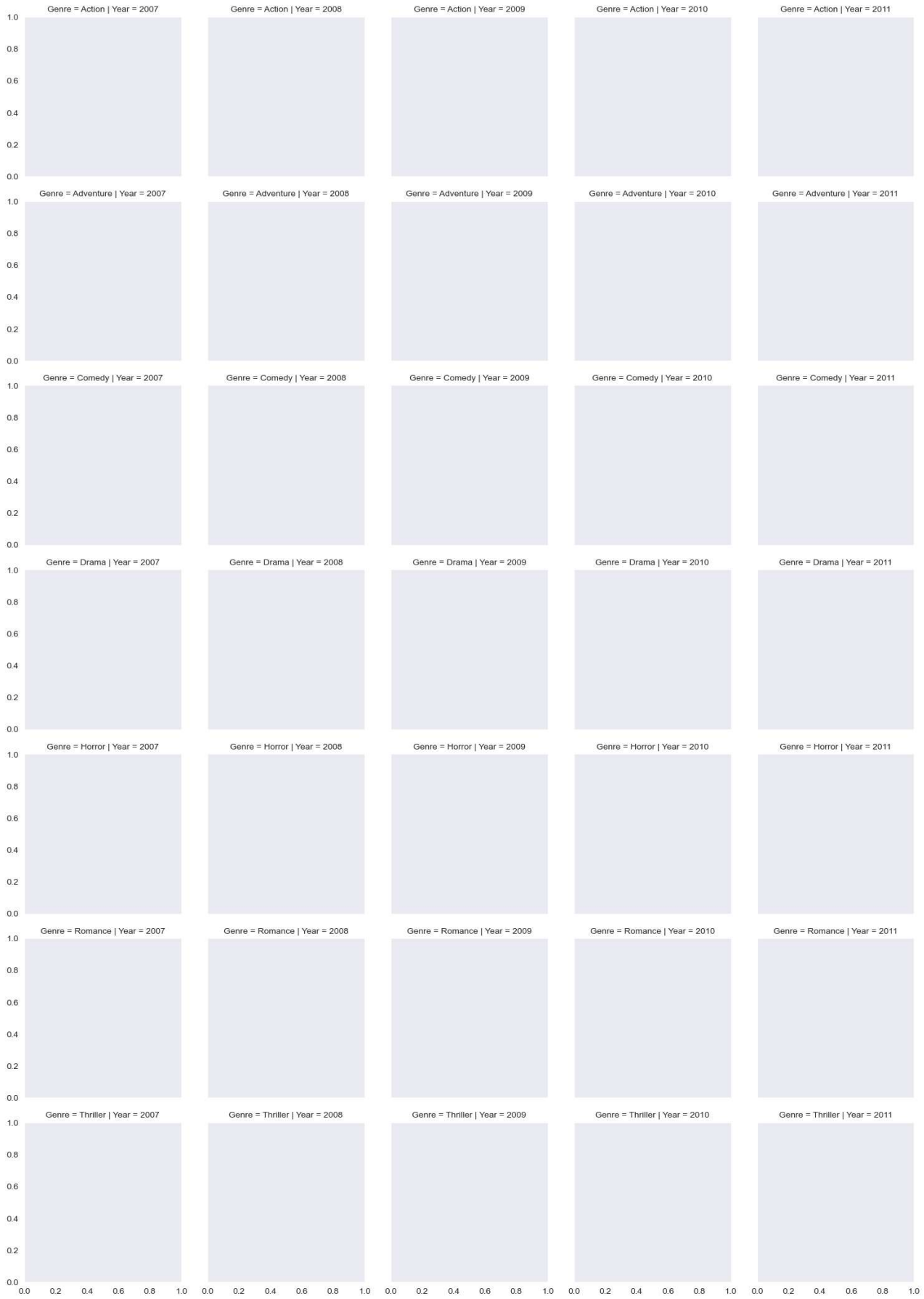


In [117]:
```
z = sns.violinplot(data=movies[movies.Genre == "Drama"], x="Year", y= "criticrating")
```



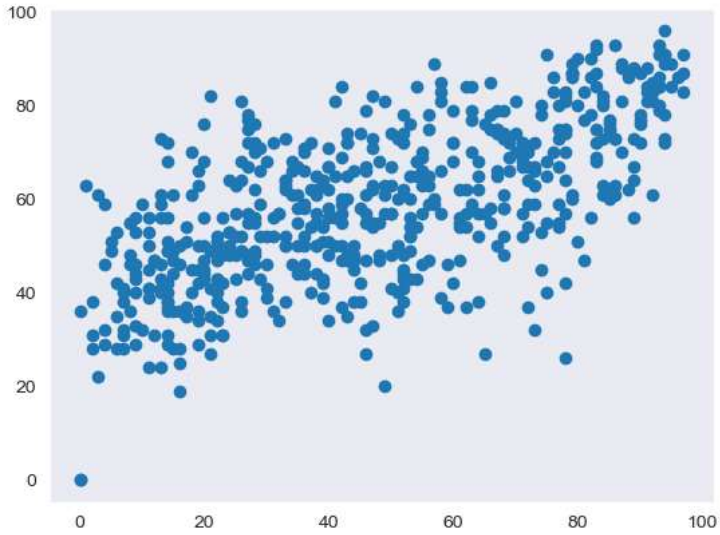
In [118]:
```
# creating a facet grid
```

In [120]:
```
g =sns.FacetGrid (movies,row = "Genre", col = "Year", hue = "Genre") # kind of subject
```

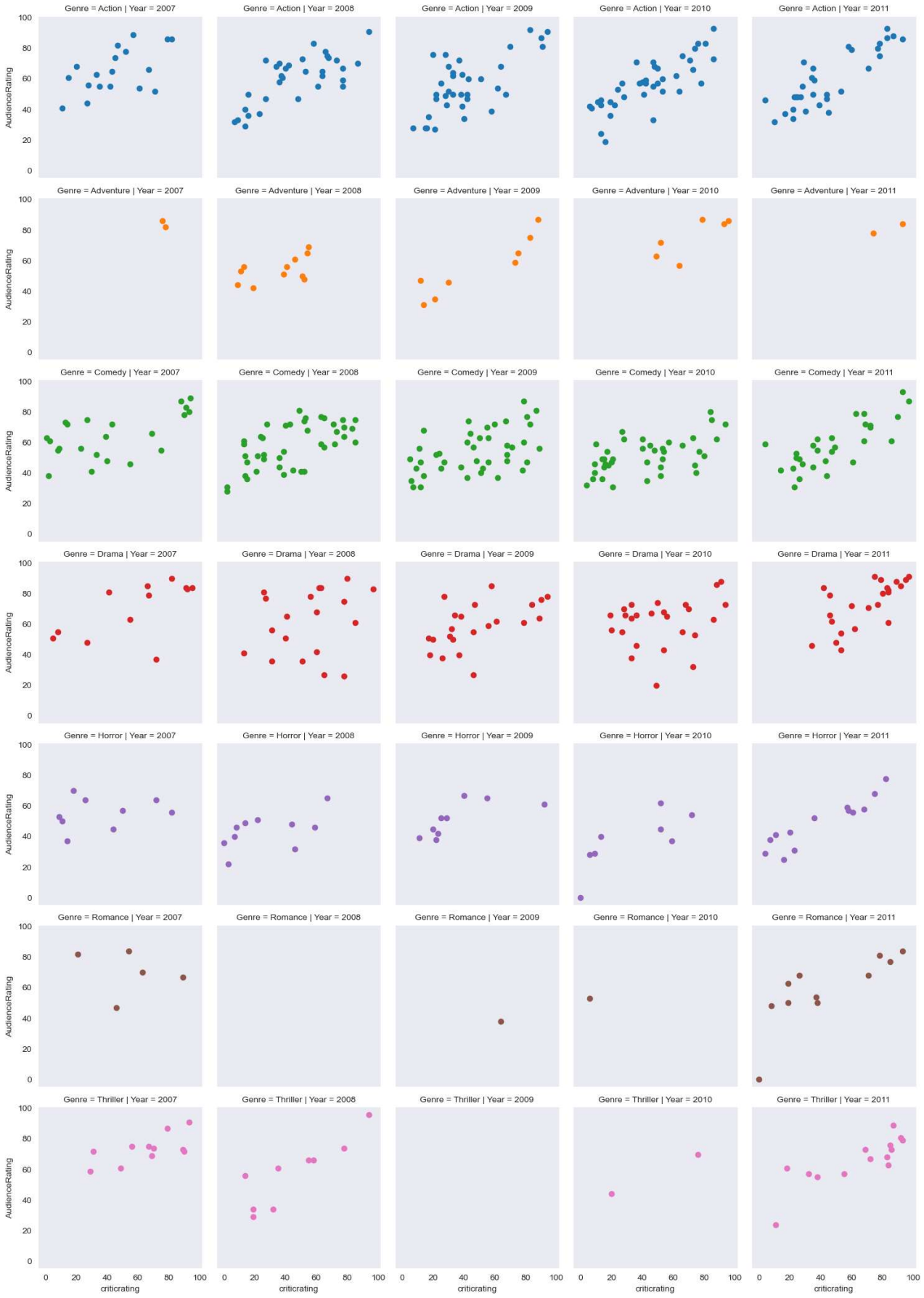


In [121]:
```
plt.scatter(movies.criticrating,movies.AudienceRating)
```

Out[121]:
```
<matplotlib.collections.PathCollection at 0x253d7536a10>
```



In [122]:
```
g = sns.FacetGrid (movies, row = "Genre", col ="Year", hue = "Genre")
g = g.map(plt.scatter, "criticrating", "AudienceRating") #scatterplots are mapped in facetgrid
```



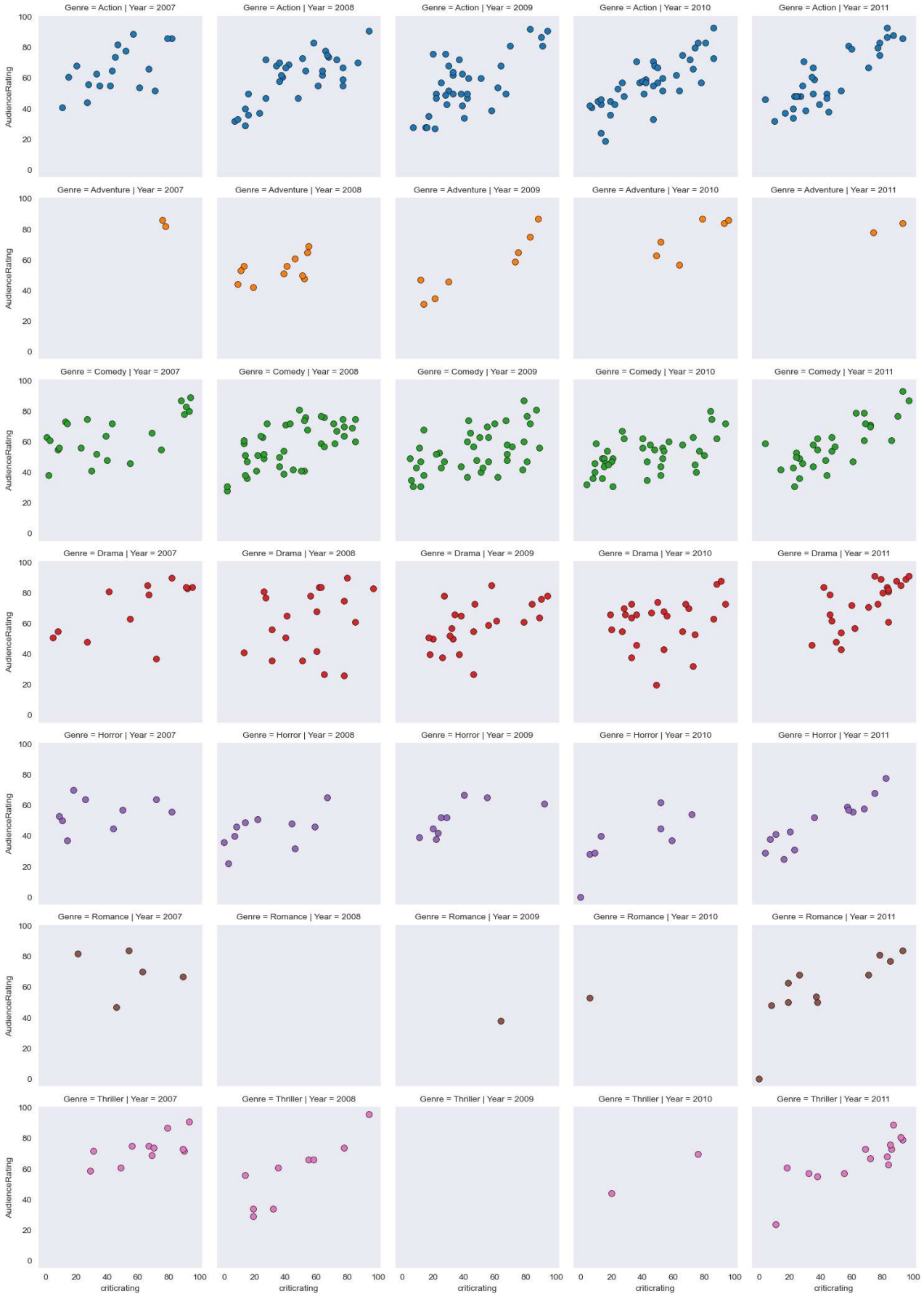
In [123]:
```
movies.head(1)
```

Out[123]:

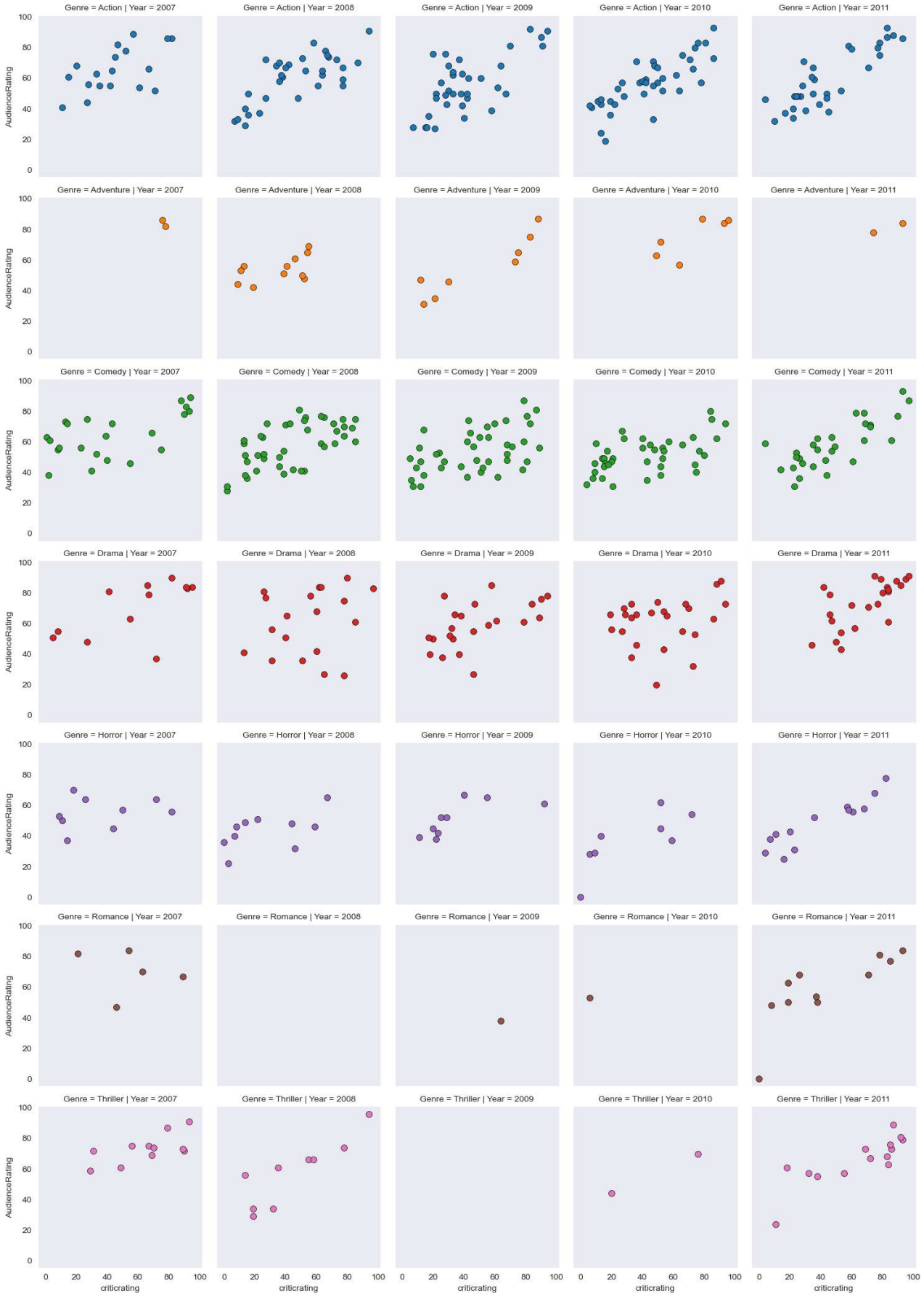
	Film	Genre	criticrating	AudienceRating	Budgetmillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009

In [124]:
```
# 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
```

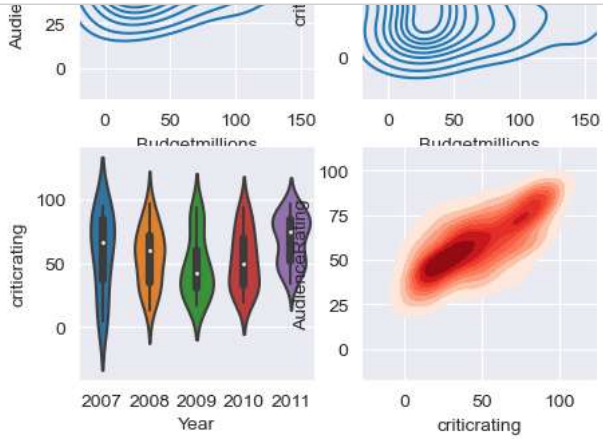
In [135]:
```
g =sns.FacetGrid (movies, row = "Genre", col = "Year", hue = "Genre")
kws = dict(s=50, linewidth=0.5,edgecolor="black")
g = g.map(plt.scatter, 'criticrating', 'AudienceRating',**kws)#scatterplots are mapped in facetgrid
```



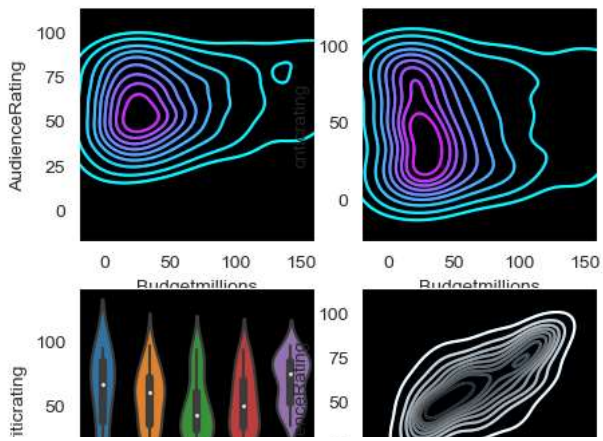
In [134]:
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
kws = dict(s=50, linewidth=0.5,edgecolor='black')
g = g.map(plt.scatter, 'criticrating', 'AudienceRating',**kws )
```



In [156]:
```
 #python is not vectorize programming language
# Building dashboards (dashboard - combination of chats)

sns.set_style("darkgrid")
f,axes = plt.subplots (2,2, figsize = (5,5))

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(movies.criticrating, movies.AudienceRating,cmap="Reds",ax = axes[1,1])
k4 = sns.kdeplot(x=movies.criticrating,y=movies.AudienceRating,shade = True,shade_lowest=False,cmap="Reds")
plt.show()
```



In [158]:
```
sns.set_style('dark',{'axes.facecolor':'black'})
f, axes = plt.subplots (2,2, figsize = (5,5))

#plot [0,0]
k1 = sns.kdeplot(x=movies.Budgetmillions,y=movies.AudienceRating, 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,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,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()
```



In [ ]: