In [134]:

```
import pandas as pd
import os
```

In [135]:

```
os.getcwd()
```

Out[135]:

```
'/Users/myyntiimac'
```

In [136]:

```
df=pd.read_csv("/Users/myyntiimac/Desktop/P4-Movie-Ratings.csv")
df.head()
```

Out[136]:

	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 [145]:

```
len(df)
```

Out[145]:

```
559
```

In [ ]:

```

```

In [146]:

```
df.shape
```

Out[146]:

```
(559, 6)
```

In [147]:

```
df.columns
```

Out[147]:

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

In [148]:

```
df.columns=['Film', 'Genre', 'Criticsratings', 'AudienceRatings',
       'Budgetmillion', 'Year']
```

In [149]:

```
df.head()
```

Out[149]:

	Film	Genre	Criticsratings	AudienceRatings	Budgetmillion	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 [150]:

```
df.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   Criticsratings   559 non-null    int64 
 3   AudienceRatings  559 non-null    int64 
 4   Budgetmillion    559 non-null    int64 
 5   Year             559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [151]:

```
df.isnull().any()
```

Out[151]:

```
Film               False
Genre              False
Criticsratings     False
AudienceRatings    False
Budgetmillion      False
Year               False
dtype: bool
```

In [152]:

```
df.describe().transpose()
```

Out[152]:

	count	mean	std	min	25%	50%	75%	max
Criticsratings	559.0	47.309481	26.413091	0.0	25.0	46.0	70.0	97.0
AudienceRatings	559.0	58.744186	16.826887	0.0	47.0	58.0	72.0	96.0
Budgetmillion	559.0	50.236136	48.731817	0.0	20.0	35.0	65.0	300.0
Year	559.0	2009.152057	1.362632	2007.0	2008.0	2009.0	2010.0	2011.0

In [ ]:

```
#we dont want year treat as number, we can make it catagorical
#in addition , film and genre showing as object , thats also need to converta as

```

In [153]:

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

```

In [154]:

```
df.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   Criticsratings   559 non-null    int64   
 3   AudienceRatings  559 non-null    int64   
 4   Budgetmillion    559 non-null    int64   
 5   Year             559 non-null    int64   
dtypes: category(1), int64(4), object(1)
memory usage: 43.6+ KB
```

In [155]:

```
df['Genre'] = df['Genre'].astype('category')
```

In [156]:

```
df.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   Criticsratings   559 non-null    int64   
 3   AudienceRatings  559 non-null    int64   
 4   Budgetmillion    559 non-null    int64   
 5   Year             559 non-null    int64   
dtypes: category(2), int64(4)
memory usage: 40.1 KB
```

In [21]:

```
df.Year=df.Year.astype("category")
```

In [157]:

```
df.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   Criticsratings   559 non-null    int64   
 3   AudienceRatings  559 non-null    int64   
 4   Budgetmillion    559 non-null    int64   
 5   Year             559 non-null    int64   
dtypes: category(2), int64(4)
memory usage: 40.1 KB
```

In [158]:

```
#to asses unique values in categories
df.Genre.cat.categories
```

Out[158]:

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

In [159]:

```
df.Genre.unique()
```

Out[159]:

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

In [160]:

```
df.describe()
```

Out[160]:

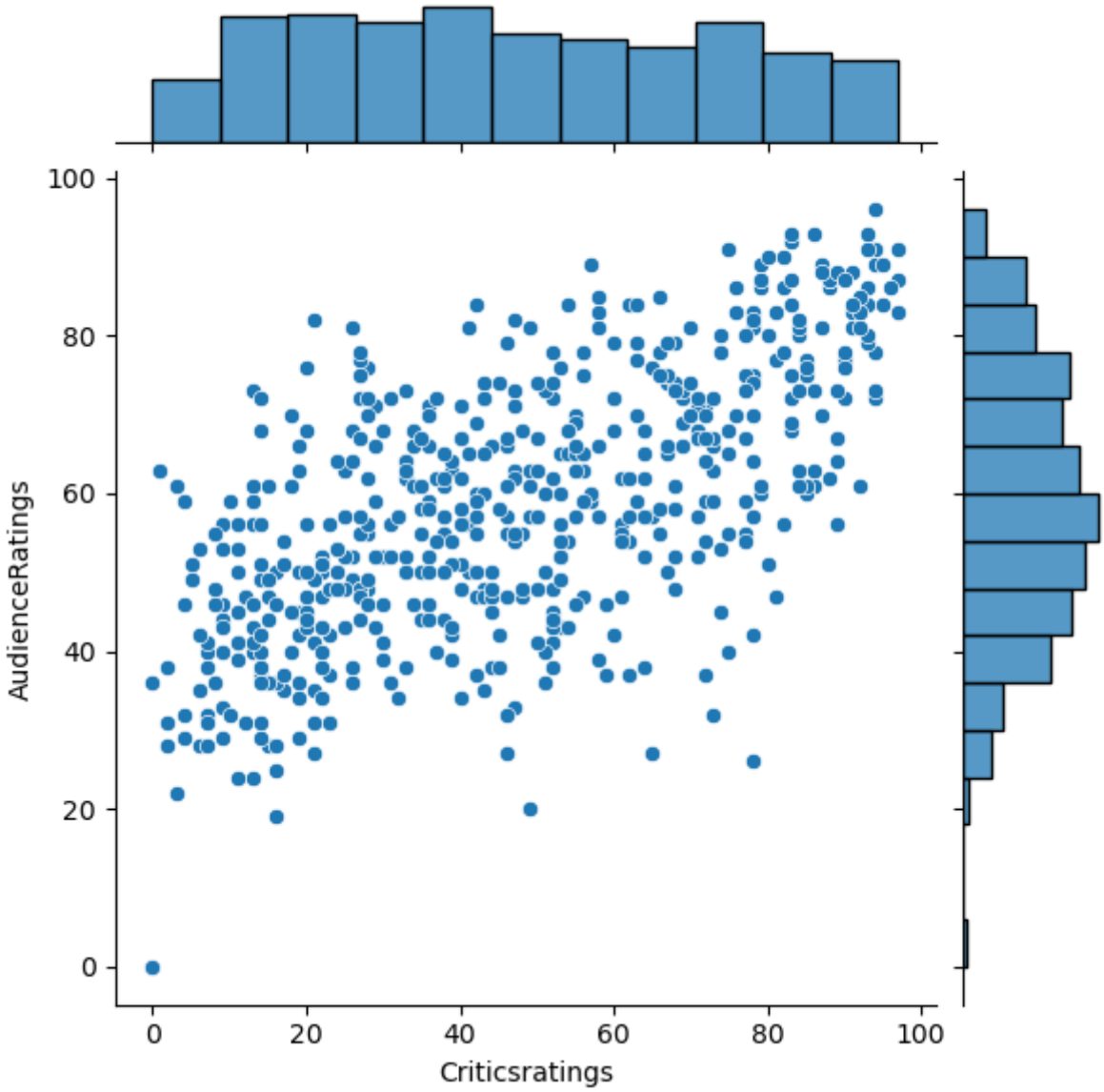
	Criticsratings	AudienceRatings	Budgetmillion	Year
count	559.000000	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136	2009.152057
std	26.413091	16.826887	48.731817	1.362632
min	0.000000	0.000000	0.000000	2007.000000
25%	25.000000	47.000000	20.000000	2008.000000
50%	46.000000	58.000000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

In [106]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

In [33]:

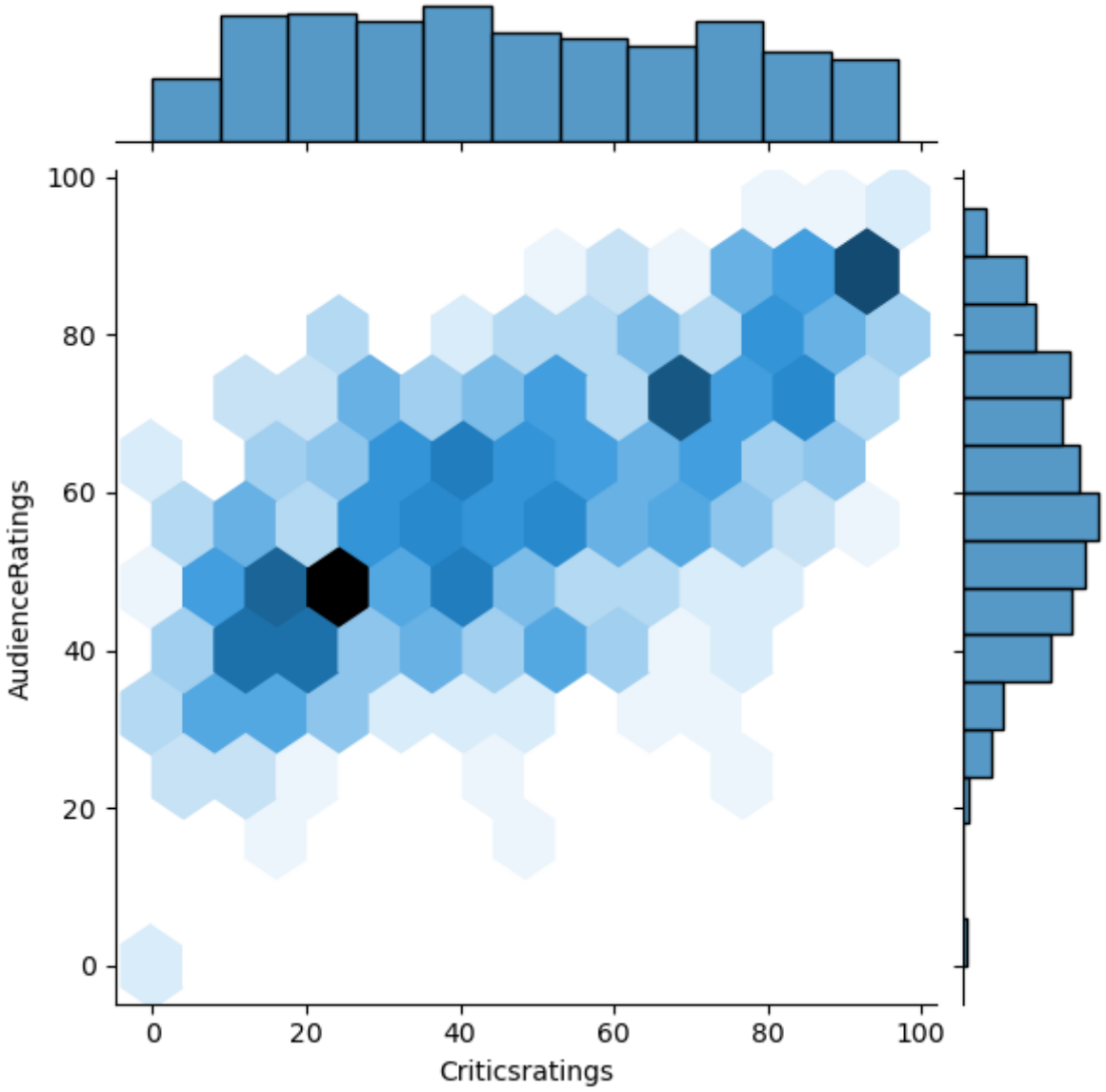
```
#creating jointplot
a=sns.jointplot(data=df, x="Criticsratings",y="AudienceRatings")
```



Insight:injoint plot, you can visualized both univariate and vivariate in scatterplot , you seee the relationship between criticsratings and audience ratings and histplot you see the distribution , in distribution we saw audince is normally disttributed so its a good predictor

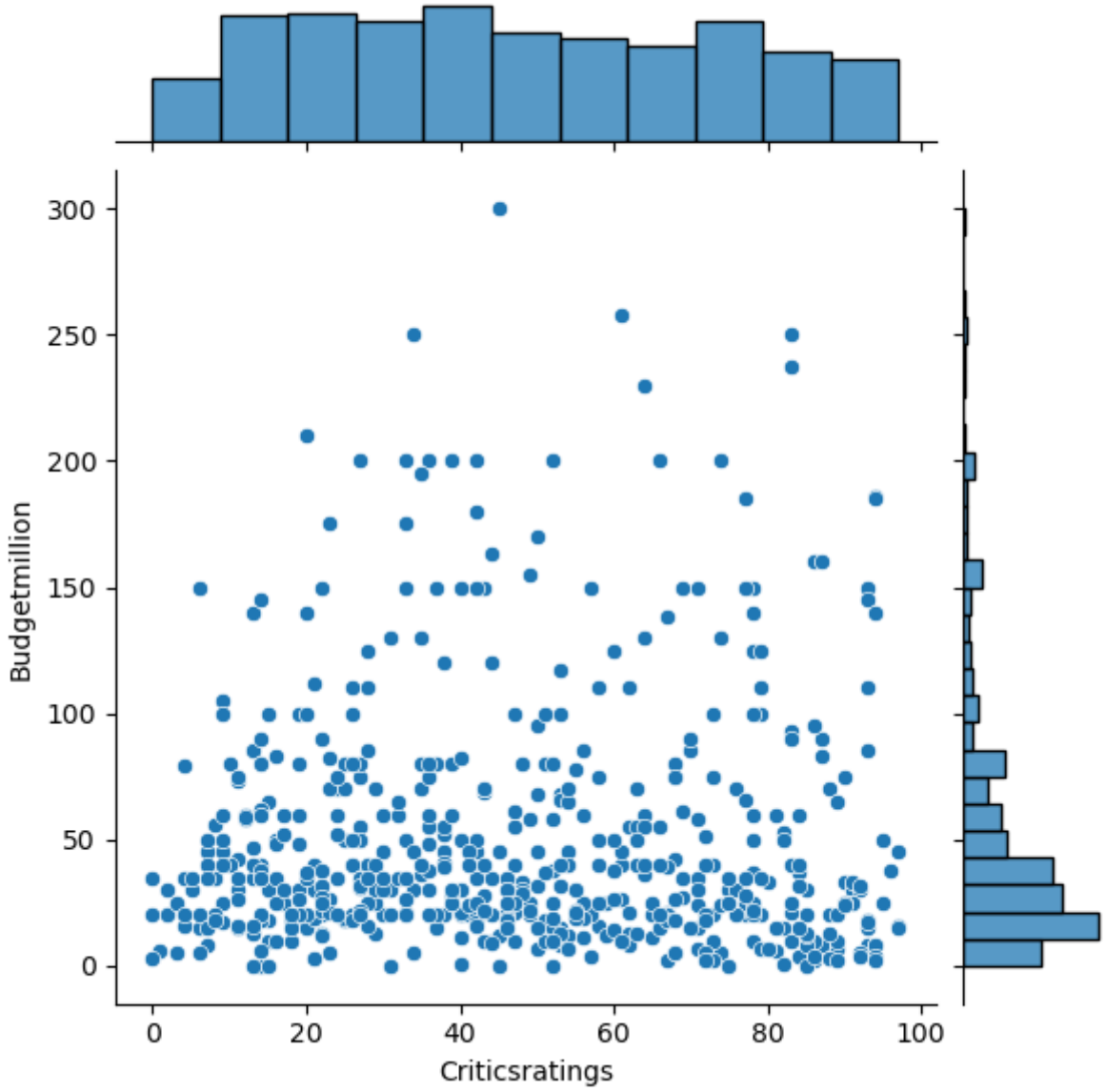
In [34]:

```
1 a=sns.jointplot(data=df, x="Criticsratings",y="AudienceRatings")
```



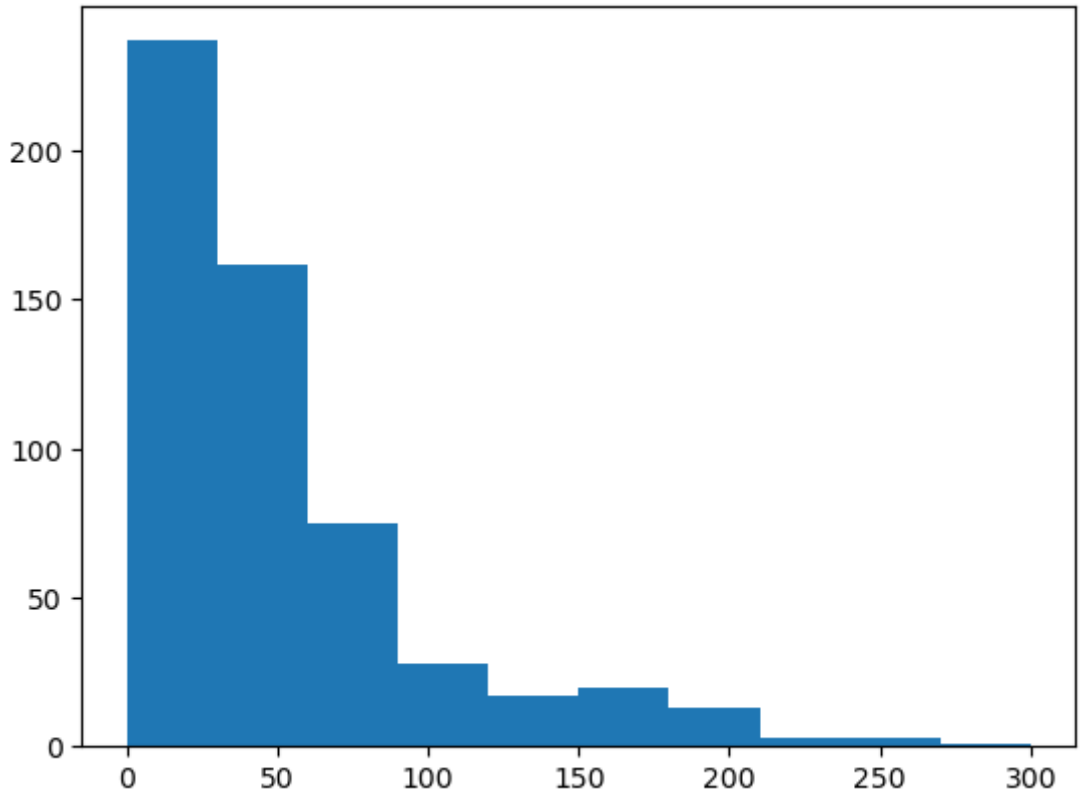
In [35]:

```
1 a=sns.jointplot(data=df, x="Criticsratings",y="Budgetmillion")
```



In [38]:

```
# creating stack distribution plot
plt.hist(df.h=sns.FacetGrid(df,row="Genre",col="Year",hue="Genre")
h=h.map(plt.hist,"Criticsratings"))
plt.show()
```



In [39]:

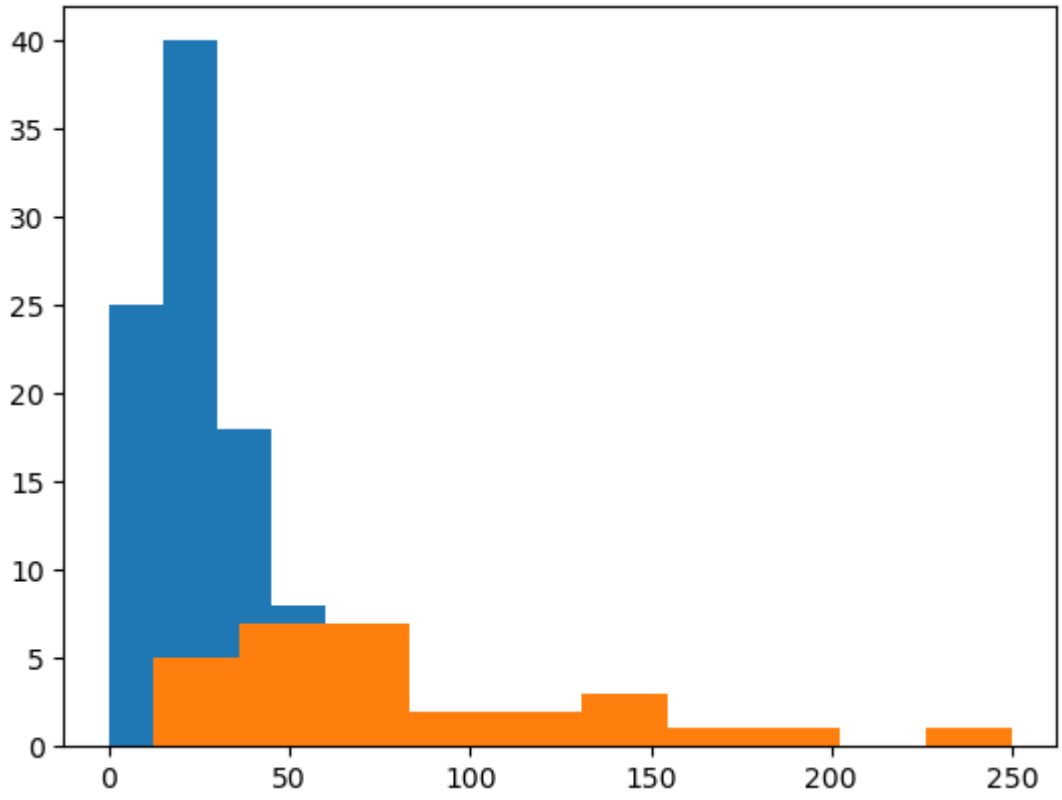
```
#cfreate a new data frame contain only Genre= Dtrama and their budget
df[df. Genre=="Drama"].Budgetmillion
```

Out[39]:

```
10     30
11     20
13      7
18      8
23     20
       ..
529    66
532    38
534    21
541    15
545     2
Name: Budgetmillion, Length: 101, dtype: int64
```

In [42]:

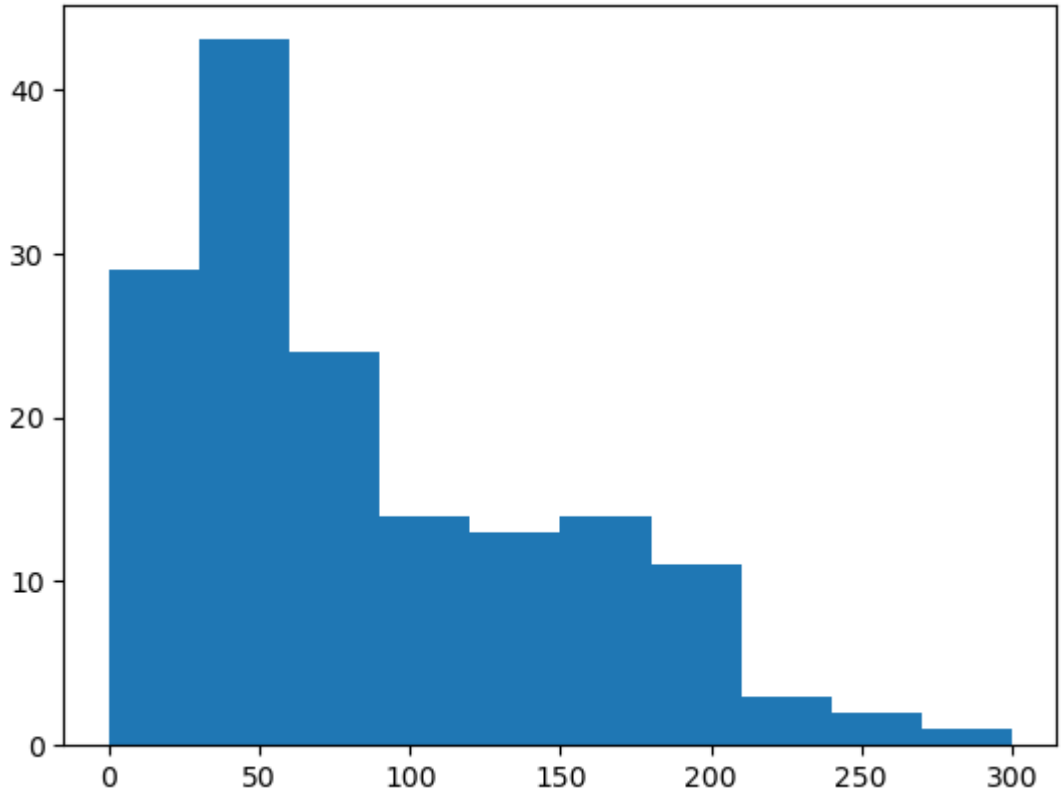
```
plt.hist(df[df. Genre=="Drama"].Budgetmillion)
plt.hist(df[df. Genre=="Adventure"].Budgetmillion)
plt.show()
```



Insight its only for Drama genre and their budget

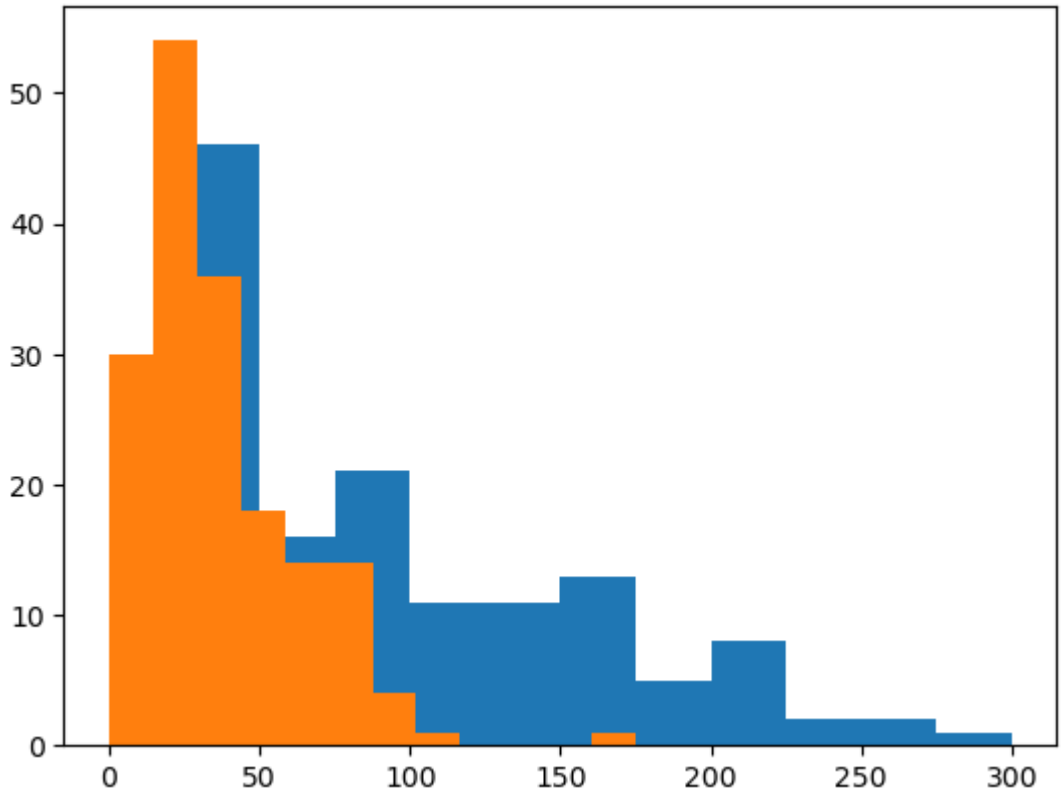
In [41]:

```
plt.hist(df[df. Genre=="Action"].Budgetmillion)
plt.show()
```



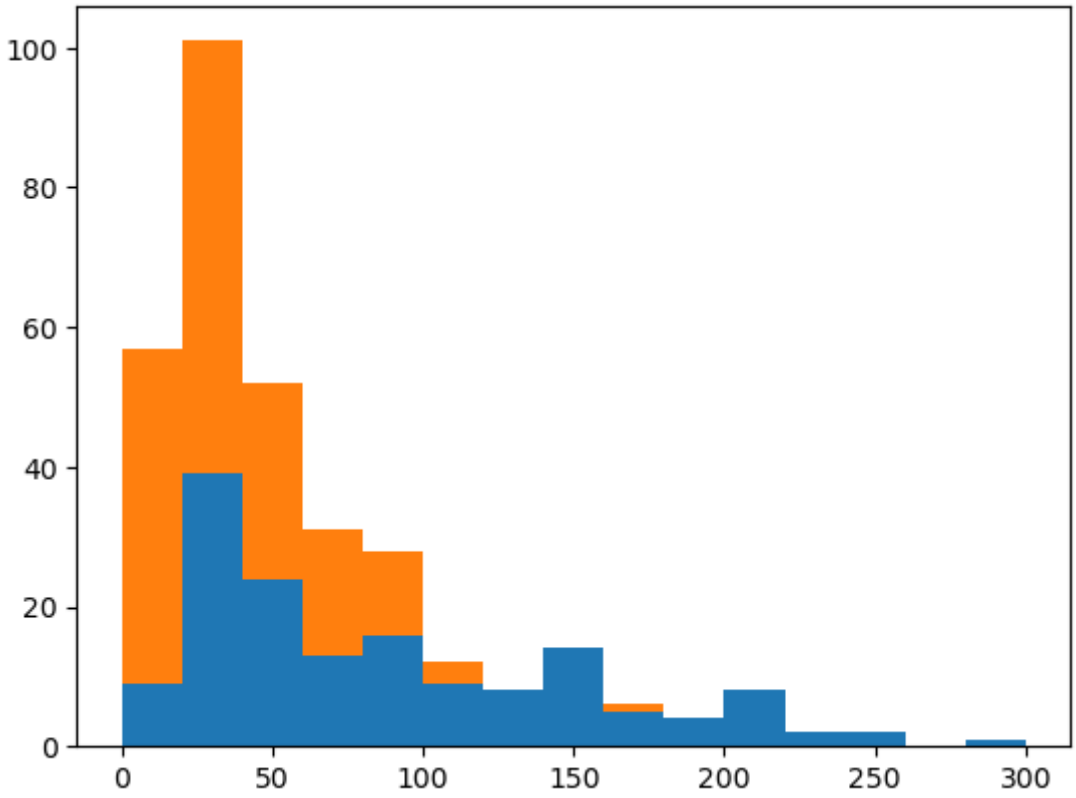
In [44]:

```
plt.hist(df[df. Genre=="Action"].Budgetmillion,bins=12)
plt.hist(df[df. Genre=="Comedy"].Budgetmillion,bins=12)
plt.show()
```



In [50]:

```
#creating stacked column
plt.hist([df[df.Genre=="Action"].Budgetmillion, df[df.Genre=="Comedy"].Budgetmil

plt.show()
```



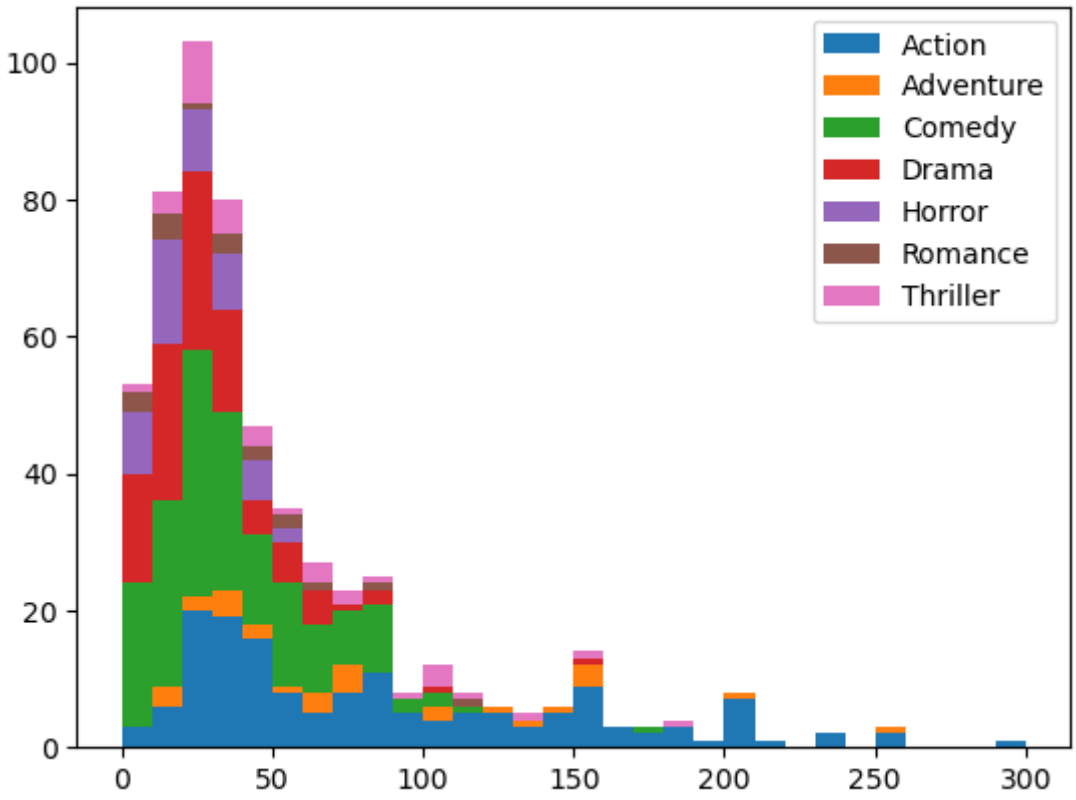
In [51]:

```
# so its repetitive task to see all genre distribution in stack
#we can automize this task by using loop
for gen in df.Genre.cat.categories:
    print(gen)
```

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

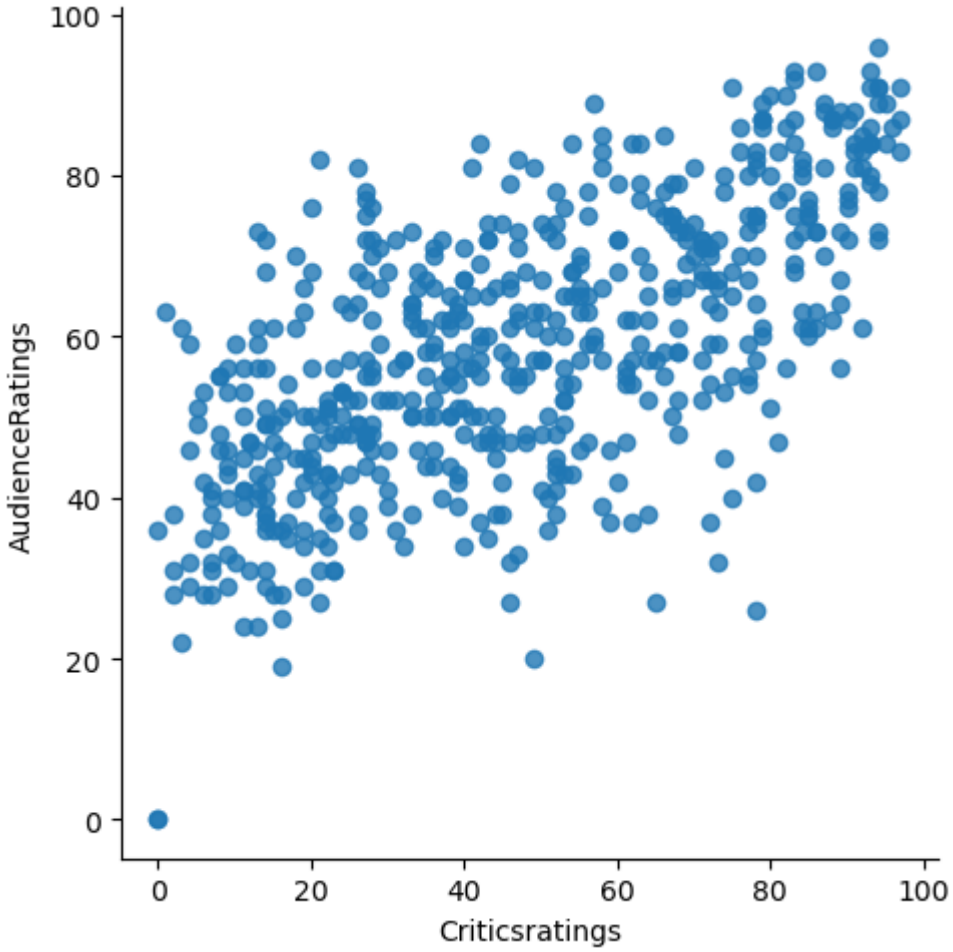
In [65]:

```
list1 = list()
mylabels=list()
for gen in df.Genre.cat.categories:
    list1.append(df[df.Genre == gen].Budgetmillion)
    mylabels.append(gen)
plt.hist(list1,bins=30,stacked=True,rwidth=1,label=mylabels)
plt.legend()
plt.show()
```



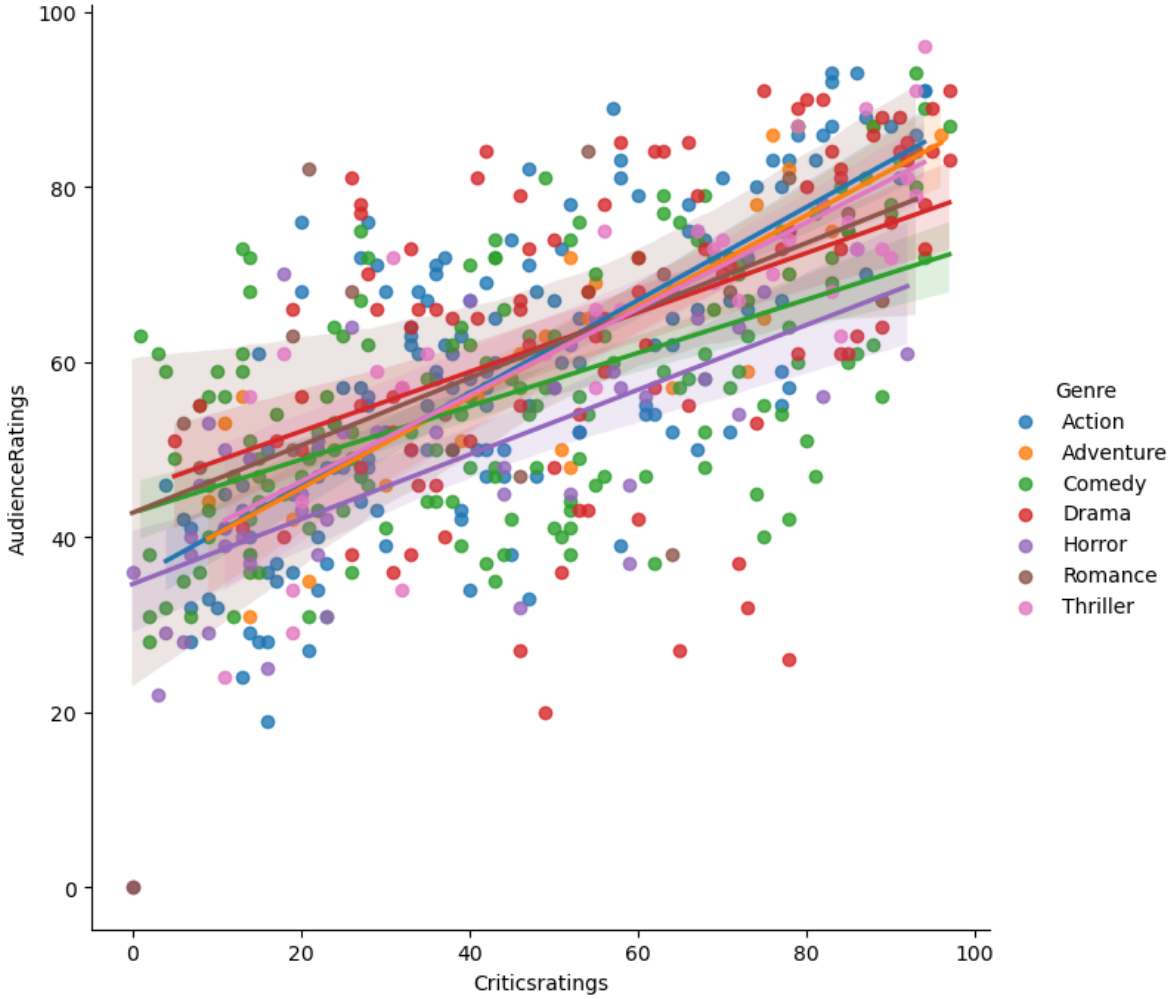
In [66]:

```
#KDE plot,code like histplot
vis1=sns.lmplot(data=df, x="Criticsratings",y="AudienceRatings",fit_reg=False)
```



In [68]:

```
1 vis1=sns.lmplot(data=df, x="Criticsratings",y="AudienceRatings",fit_reg=True,hue
```

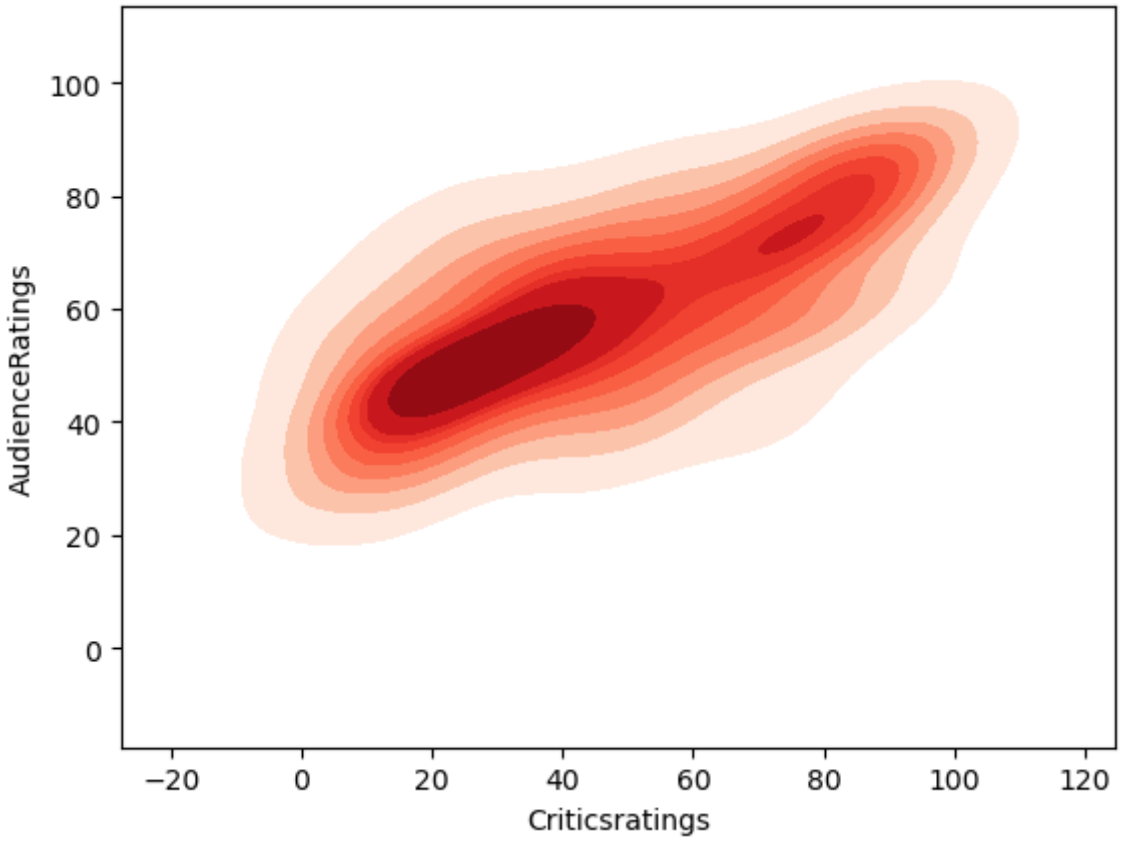


In [76]:

```
#KDE
import seaborn as sns

vis2 = sns.kdeplot(data=df, x="Criticsratings", y="AudienceRatings", shade=True,
vis2
```

Out[76]:

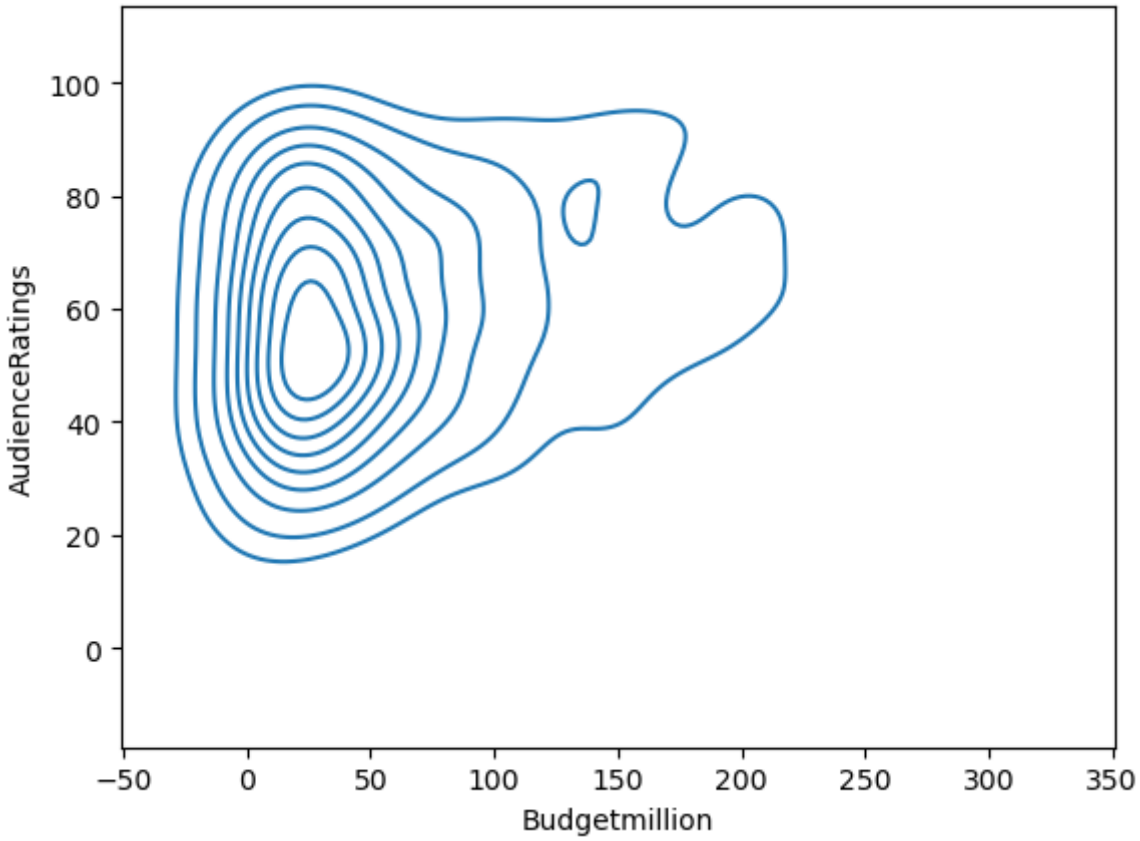
```
<Axes: xlabel='Criticsratings', ylabel='AudienceRatings'>
```



insight:its show where datapoint density is distributed

In [78]:

```
#Subplot in python
vis3 = sns.kdeplot(data=df, x="Budgetmillion", y="AudienceRatings")
```

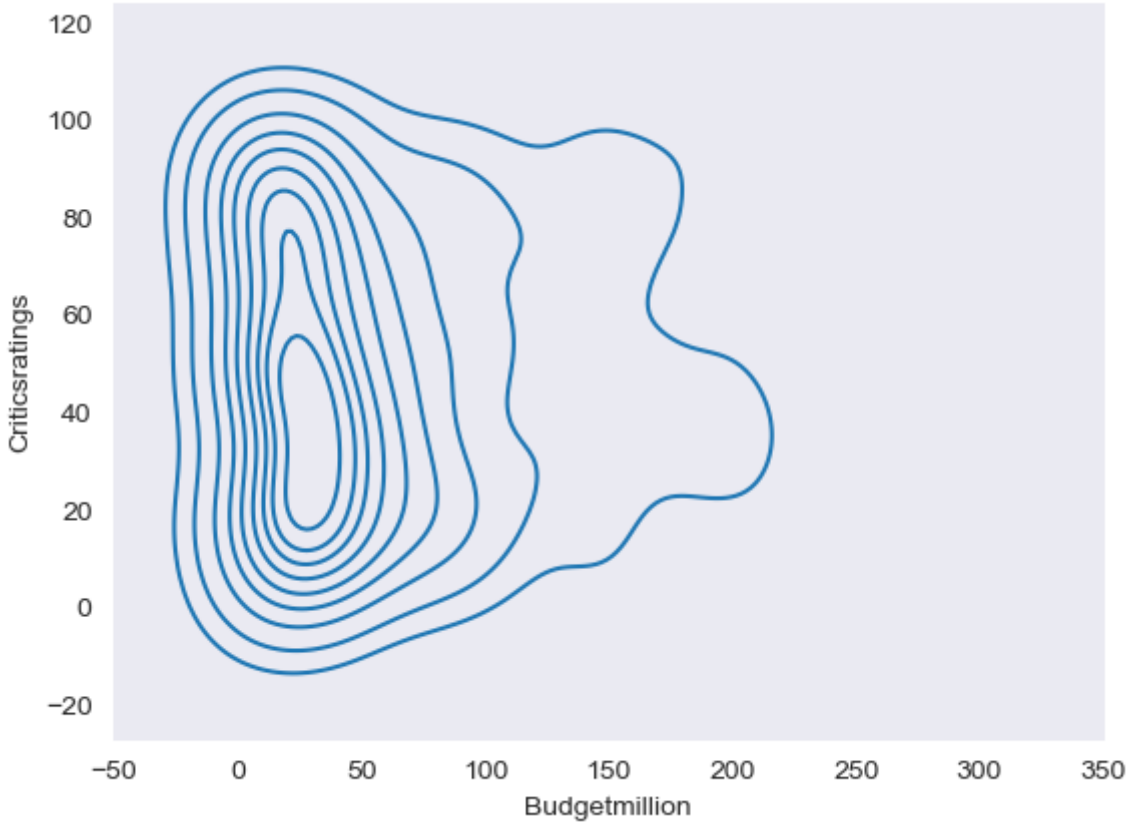


In [ ]:

```
insight:how budget affects the audience ratings
```

In [90]:

```
sns.set_style("dark")
vis4 = sns.kdeplot(data=df, x="Budgetmillion", y="Criticsratings")
```

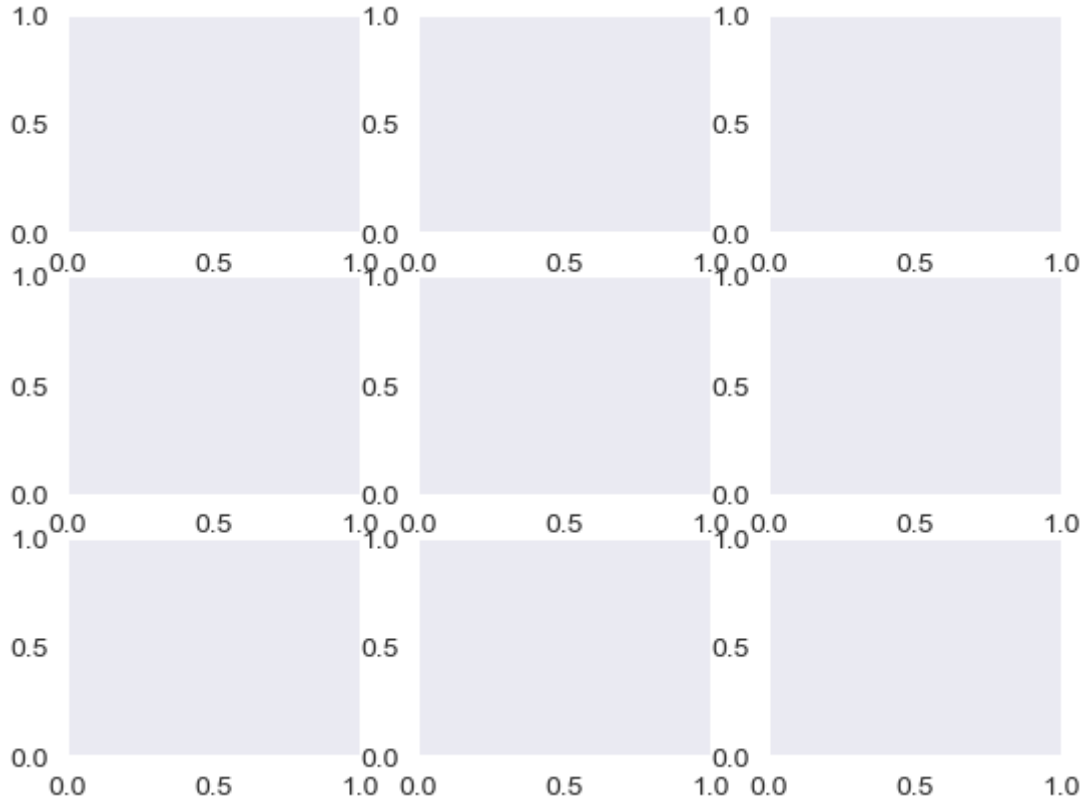


Insight:kind of uniform distribution

Creating subplot

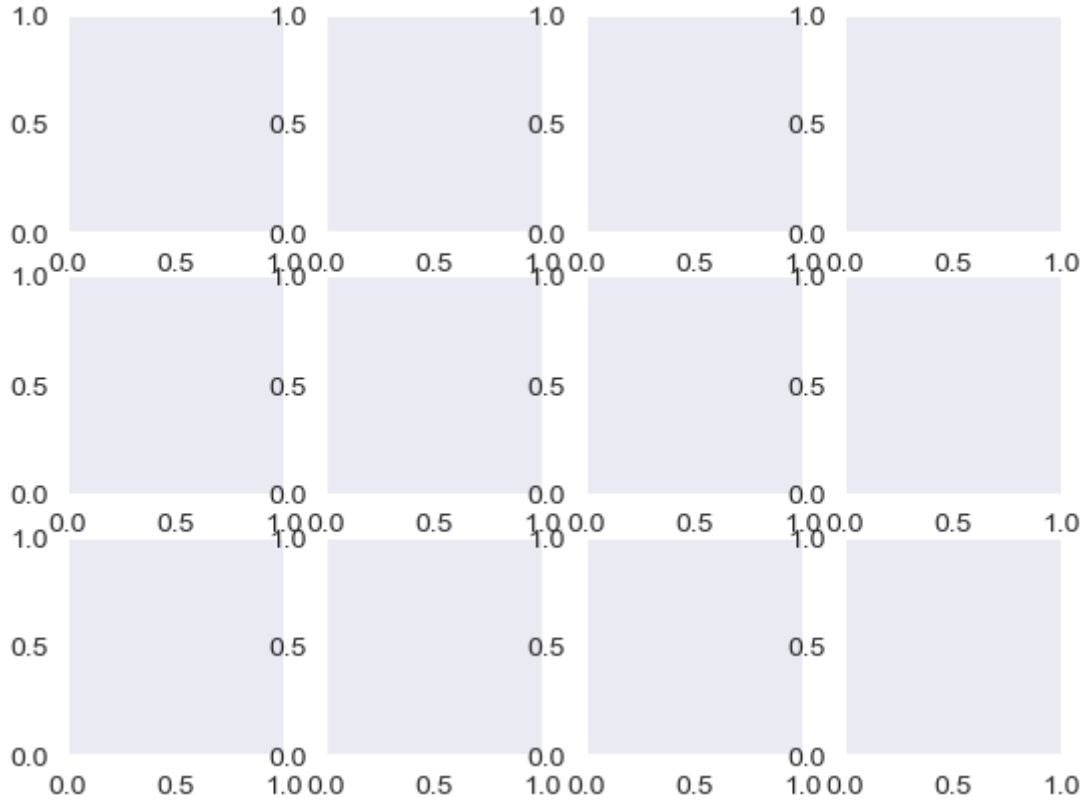
In [85]:

```
1 f,ax=plt.subplots(3,3)
```



In [86]:

```
1 f,ax=plt.subplots(3,4)
```

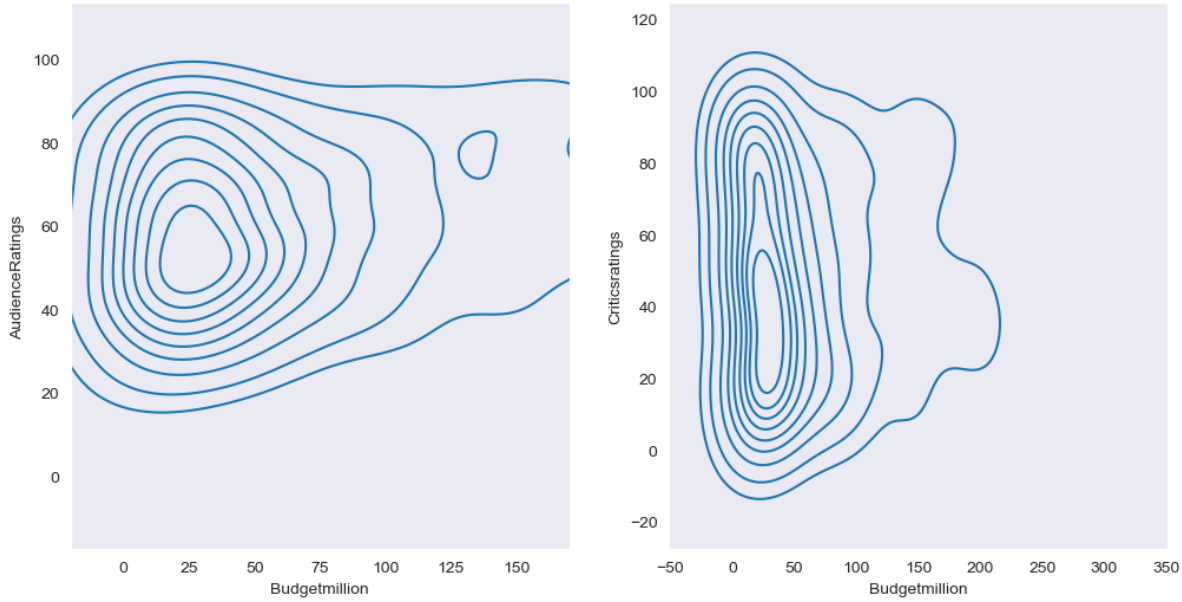


In [94]:

```
f,axes=plt.subplots(1,2,figsize=(12,6))
vis3 = sns.kdeplot(data=df, x="Budgetmillion", y="AudienceRatings",ax=axes[0])
vis4 = sns.kdeplot(data=df, x="Budgetmillion", y="Criticsratings",ax=axes[1])
vis3.set(xlim=(-20,170))
```

Out[94]:

```
[(-20.0, 170.0)]
```

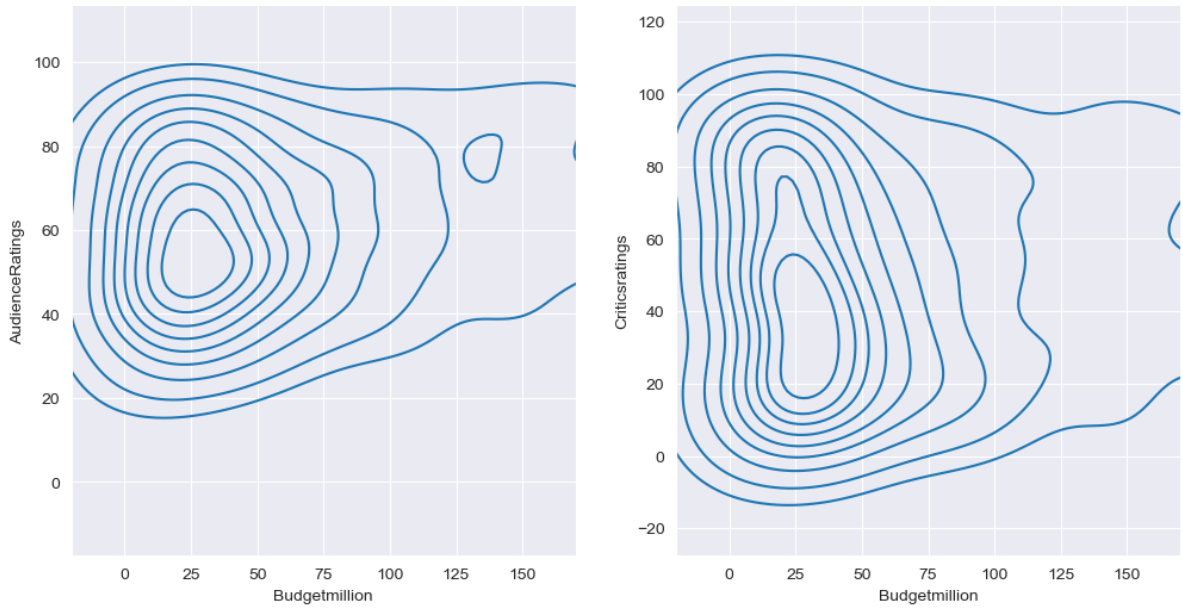


In [161]:

```
f,axes=plt.subplots(1,2,figsize=(12,6),sharex=True)
vis3 = sns.kdeplot(data=df, x="Budgetmillion", y="AudienceRatings",ax=axes[0])
vis4 = sns.kdeplot(data=df, x="Budgetmillion", y="Criticsratings",ax=axes[1])
vis3.set(xlim=(-20,170))
```

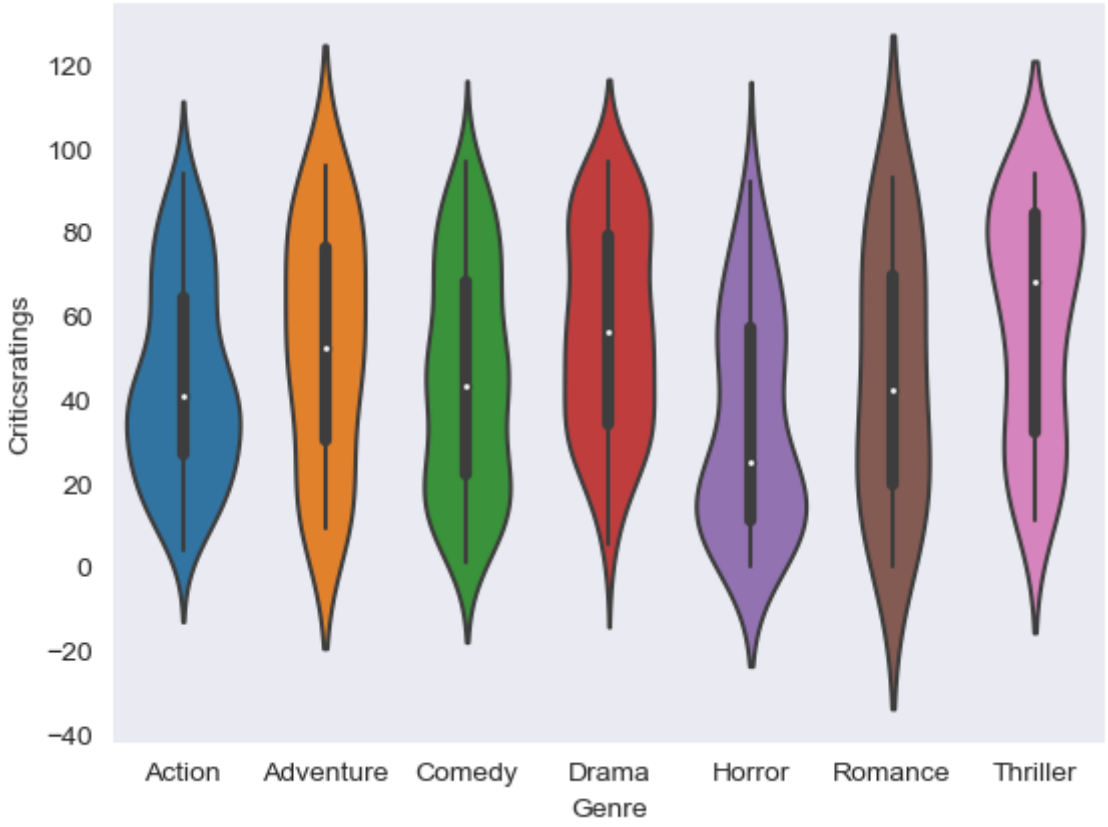
Out[161]:

```
[(-20.0, 170.0)]
```



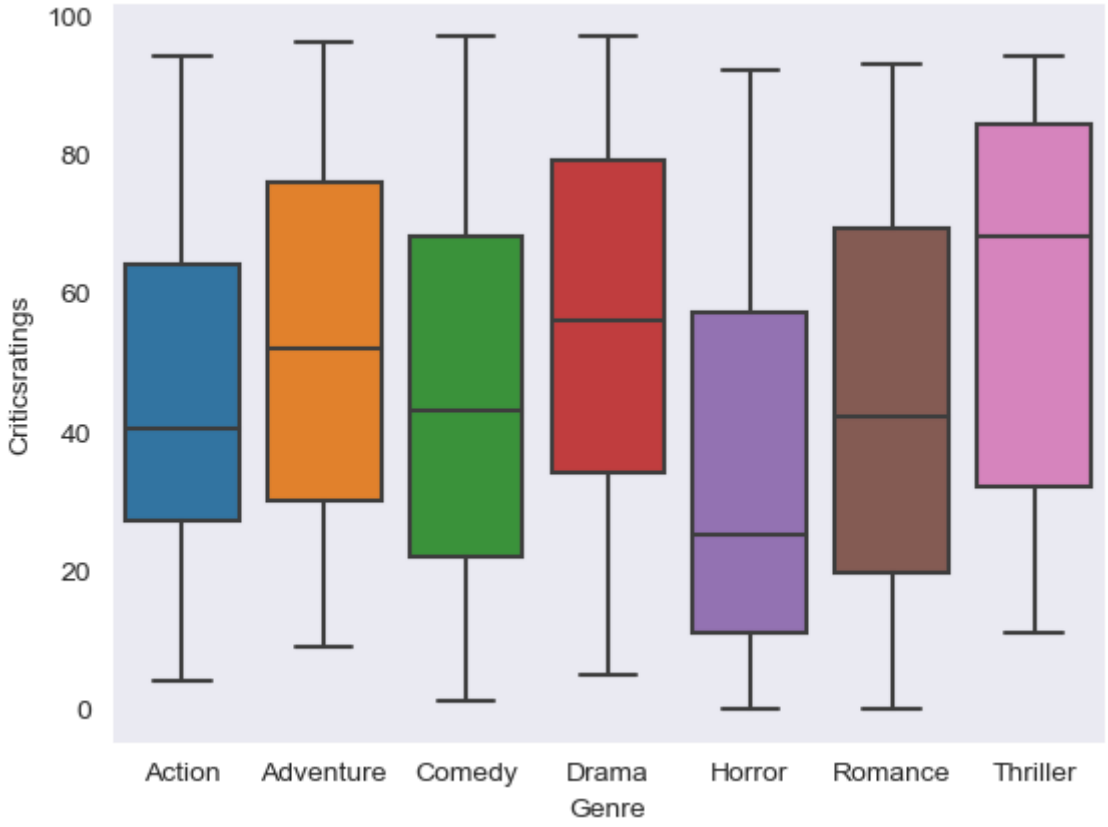
In [96]:

```
# violin and box plot
v=sns.violinplot(data=df,x="Genre",y="Criticsratings")
```



In [97]:

```
b=sns.boxplot(data=df,x="Genre",y="Criticsratings")
```



Insight :its tells hows critic rating distributed across genre, first quartile 2nd and 3rd quartile, , median(chck the highest or lowest median among them) max min and outliar inthis case thriller have high median for critic sratings, so we will build thriller

In [99]:

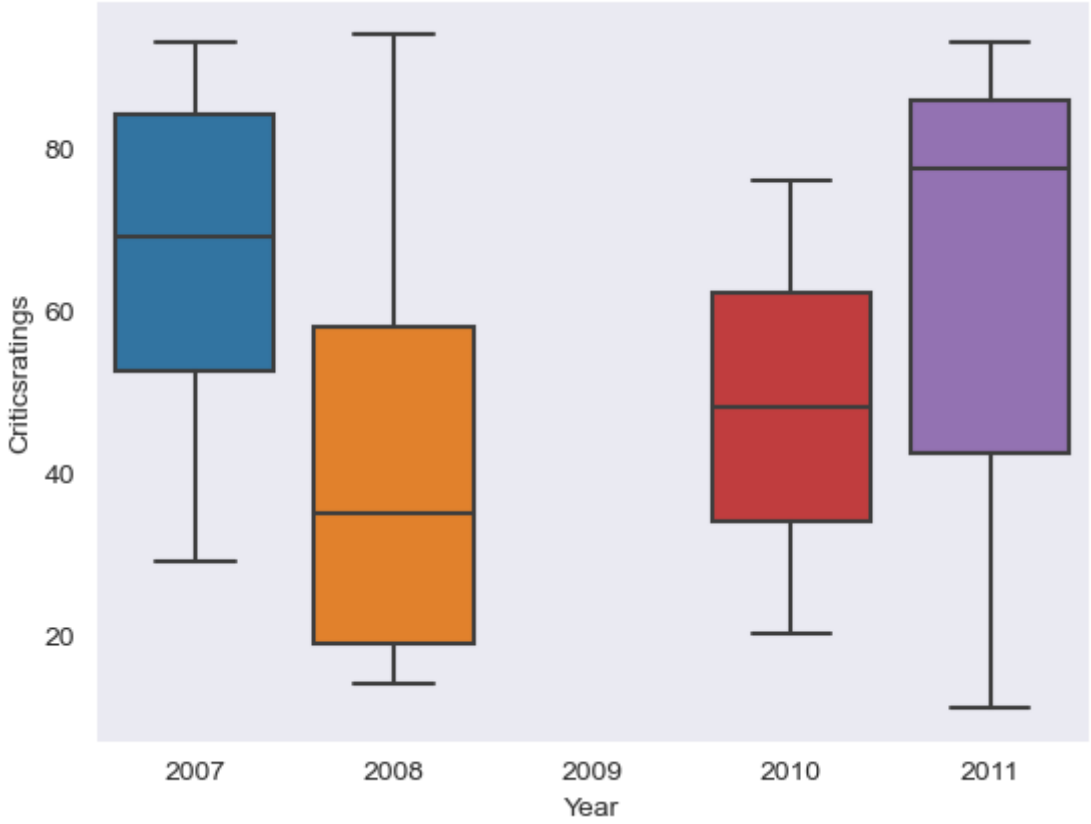
```
#if you want to see only one genre, critics ratings abd with different year
# first crate a new dta frame for specific genre
Thriller=df[df.Genre=="Thriller"]
```

In [100]:

```
b=sns.boxplot(data=Thriller,x="Year",y="Criticsratings")
b
```

Out[100]:

```
<Axes: xlabel='Year', ylabel='Criticsratings'>
```

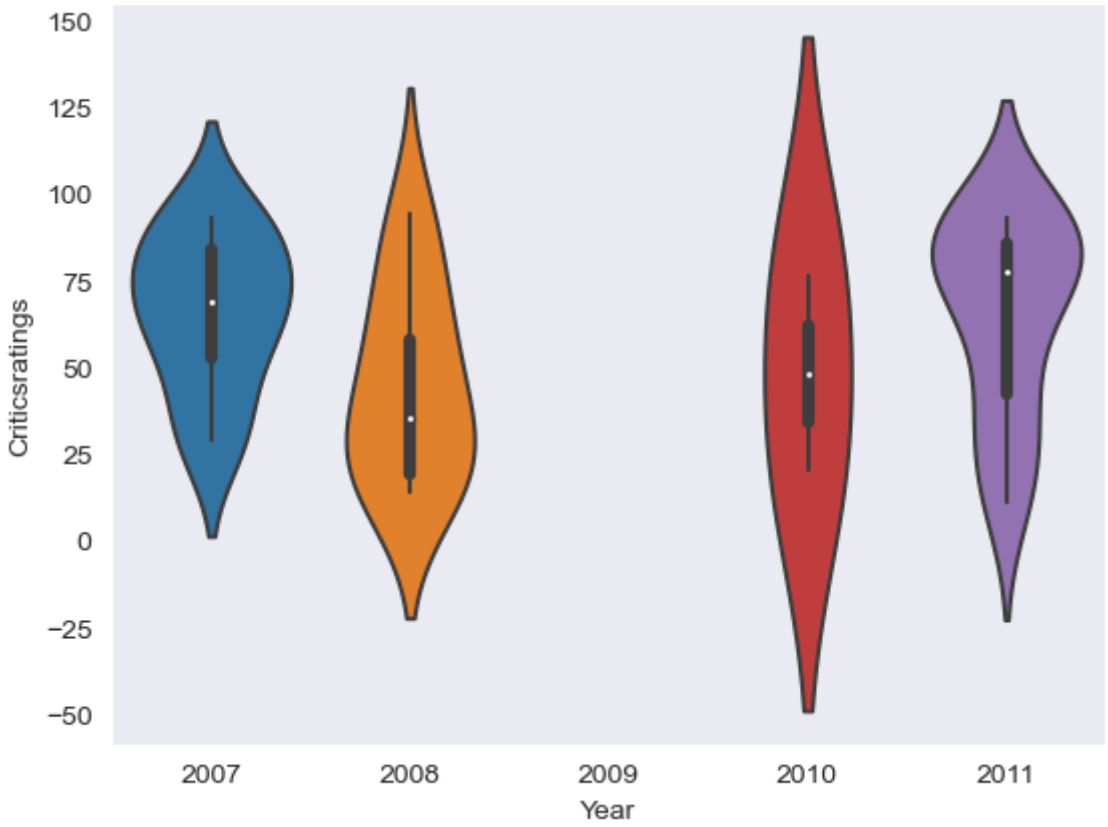


In [101]:

```
v1=sns.violinplot(data=Thriller,x="Year",y="Criticsratings")
v1
```

Out[101]:

```
<Axes: xlabel='Year', ylabel='Criticsratings'>
```



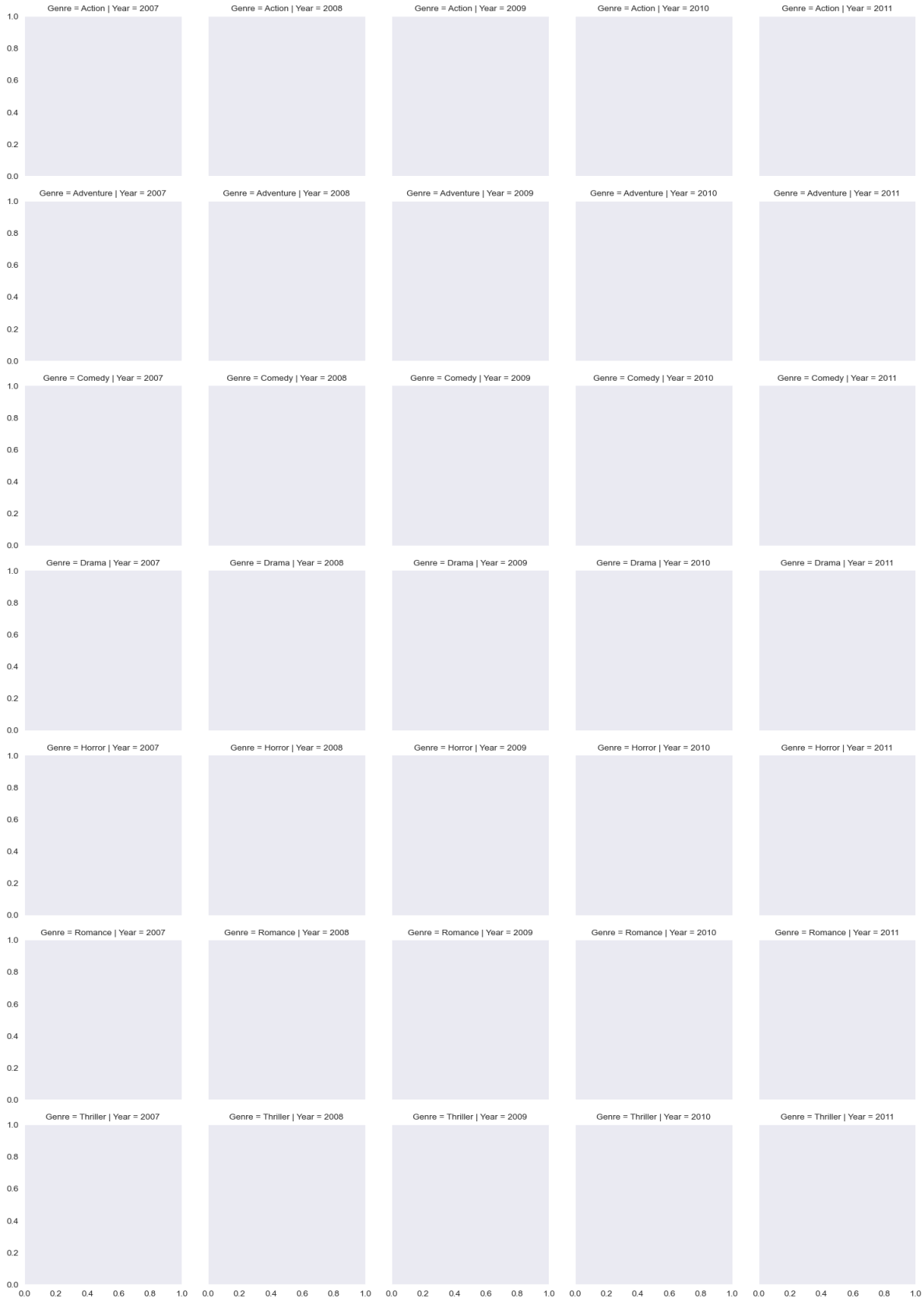
In [ ]:

```
Insight: Violinplot can show you something beterr than b oxpplot by shape of box
```

#creating facedgrid #how to create multiple chart for separate genre, why ?if you see lmplot, we can identify separatly datapoint for individual genre

In [110]:

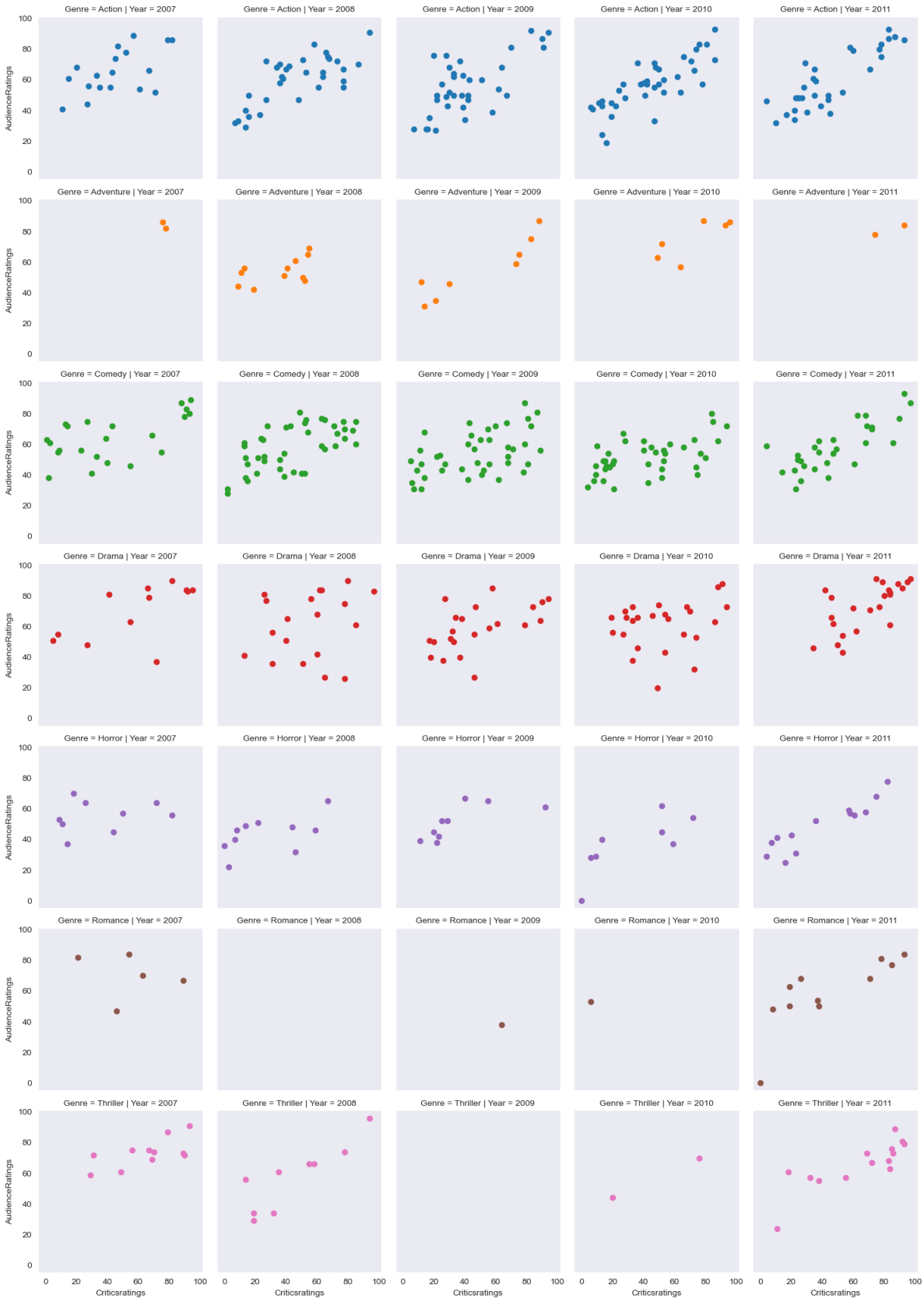
```
1 g=sns.FacetGrid(df,row="Genre",col="Year",hue="Genre")
```



in here, you basically controles the visualizaion using some sort rules

In [115]:

```
#now we are going to insert plot into facedgrid by map function
g=sns.FacetGrid(df,row="Genre",col="Year",hue="Genre")
g=g.map(plt.scatter,"Criticsratings","AudienceRatings")

```

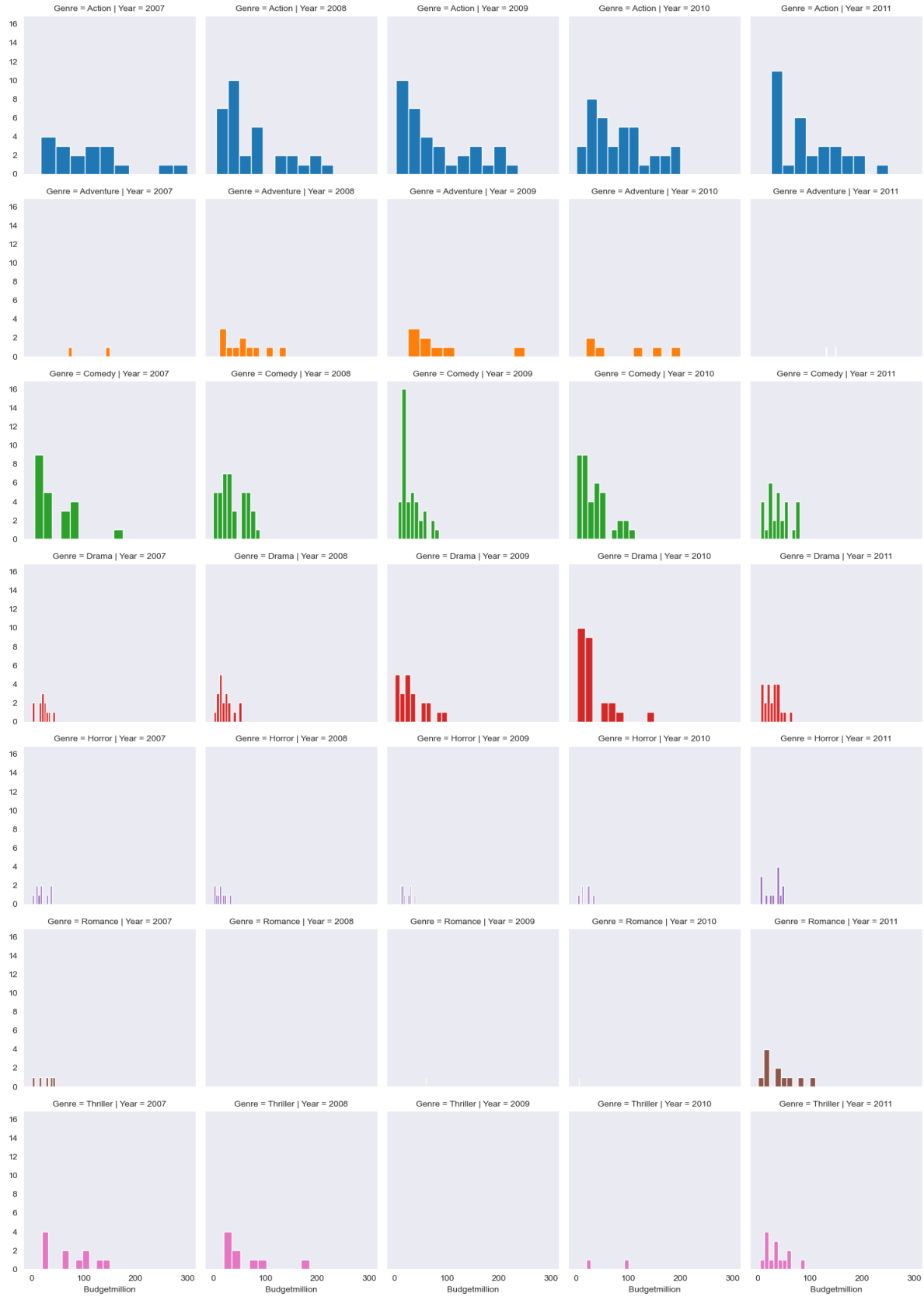


In [118]:

```
# you can do this for other plot also
#do for histplot
h=sns.FacetGrid(df,row="Genre",col="Year",hue="Genre")
h=h.map(plt.hist,"Criticsratings")

```

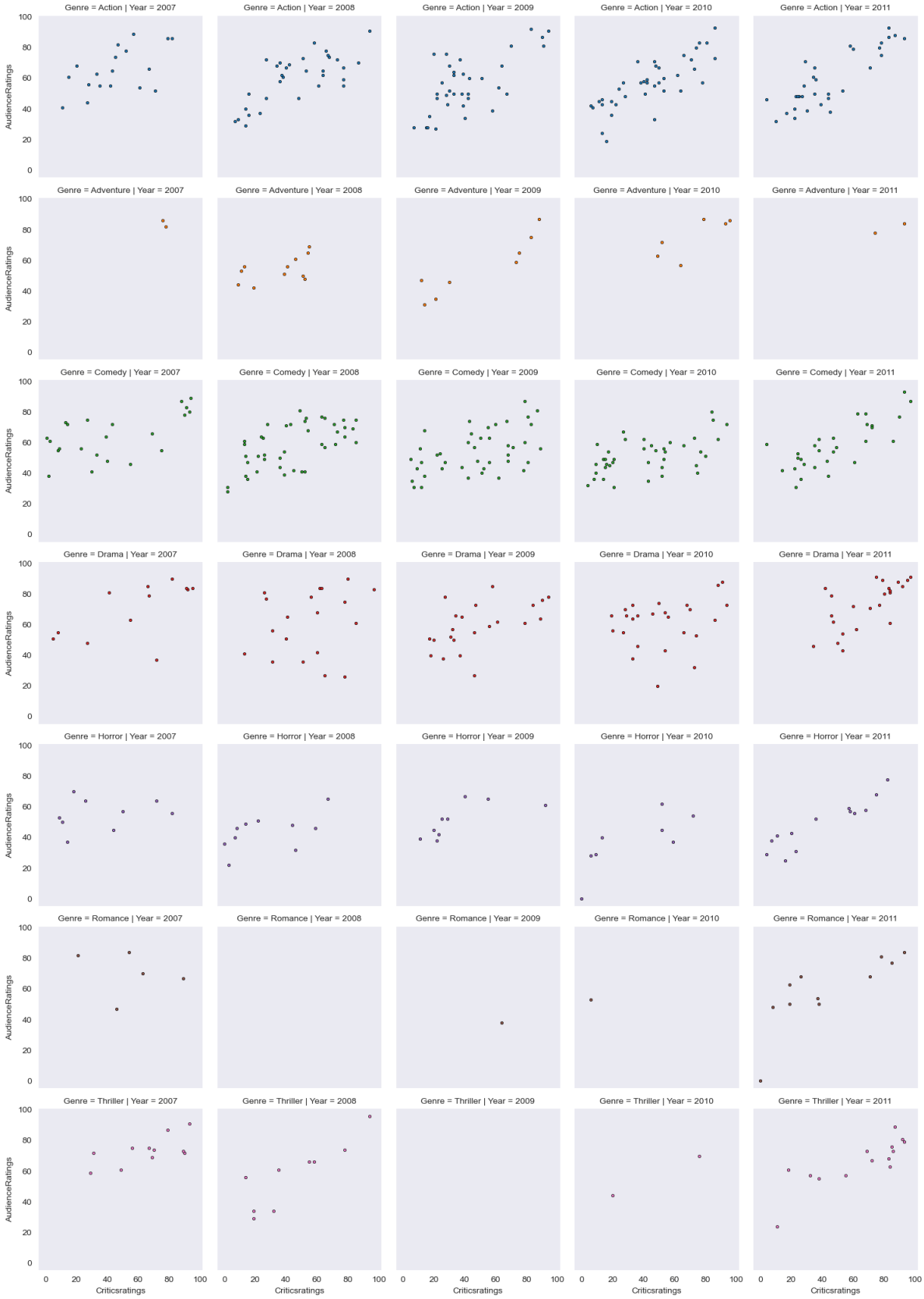
In [120]:

```
h=sns.FacetGrid(df,row="Genre",col="Year",hue="Genre")
h=h.map(plt.hist,"Budgetmillion")
```



In [123]:

```
#inserting third argument , like contoure of data point size

g=sns.FacetGrid(df,row="Genre",col="Year",hue="Genre")
kws=dict(s=10,linewidth=.5,edgecolor="black")
g=g.map(plt.scatter,"Criticsratings","AudienceRatings",**kws)

```



insight: tells how audience and critics rating for each genre throught the year

In [127]:

```
#optimizing cordinal and diagonal to bettter understans, if you see cardinal (x_
#and by diagonal you can distinguished well critics ratings and audience ratings
# for this cardinal, set the limit for x and y axies
g=sns.FacetGrid(df,row="Genre",col="Year",hue="Genre")
kws=dict(s=10,linewidth=.5,edgecolor="black")
g=g.map(plt.scatter,"Criticsratings","AudienceRatings",**kws)
g.set(xlim=(0,90),ylim=(0,90))
```

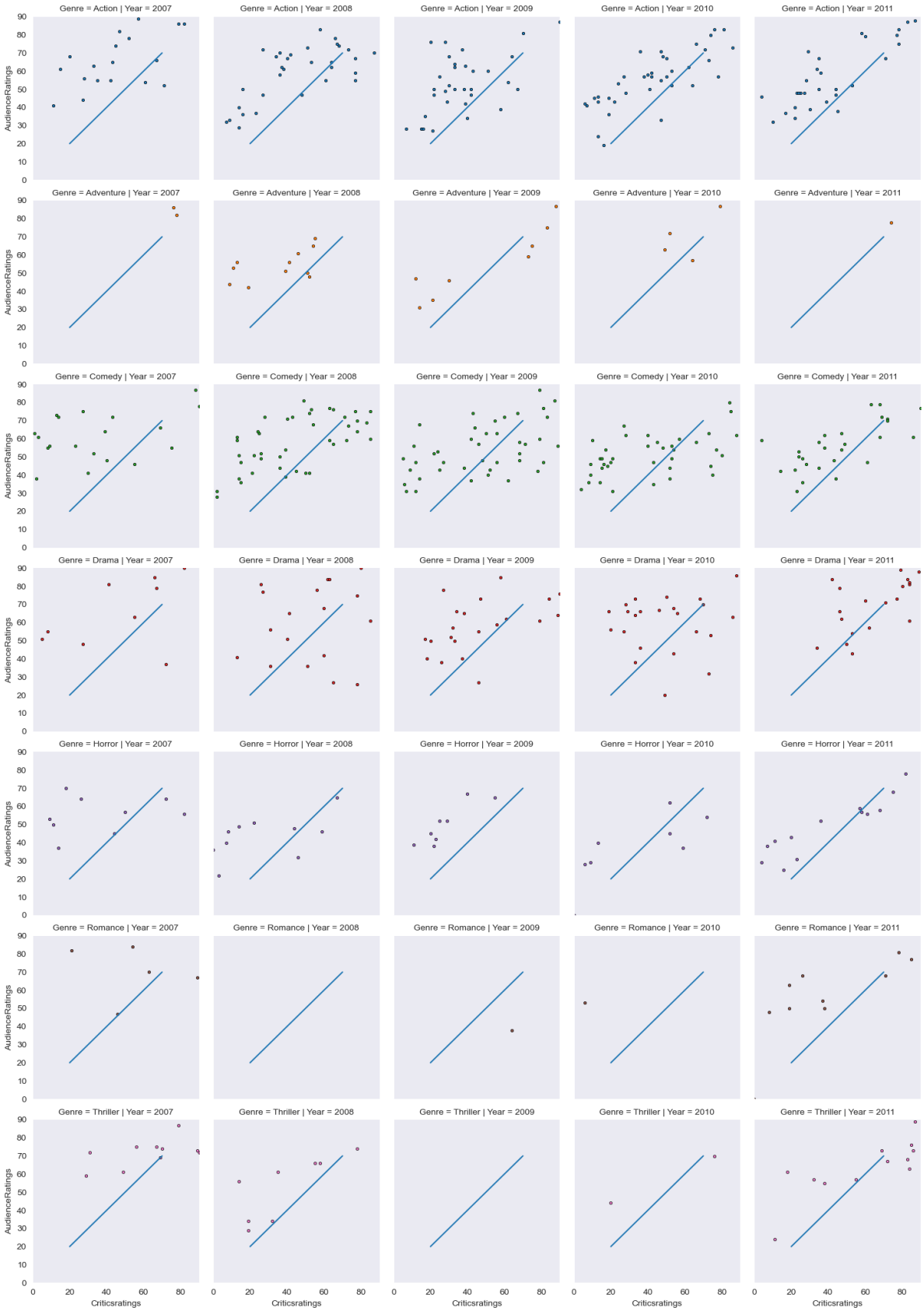
Out[127]:

```
<seaborn.axisgrid.FacetGrid at 0x7faa3a6ef220>
```


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
AudienceRatings
Criticsratings

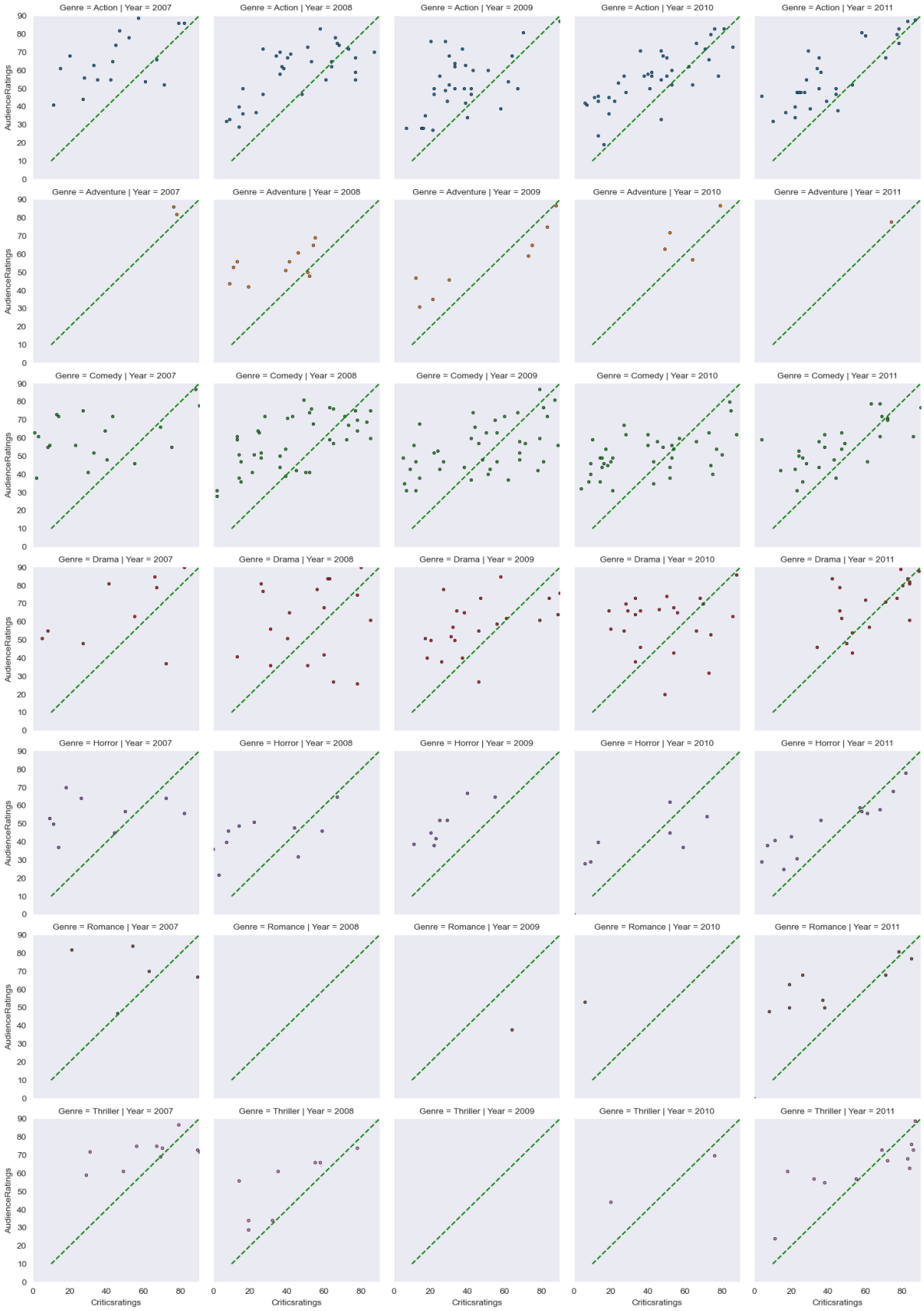
In [129]:

```
#for insert diagonal , you have to loop in ax  andc axes as array
# thhen define the range from what poits draw the line
g=sns.FacetGrid(df,row="Genre",col="Year",hue="Genre")
kws=dict(s=10,linewidth=.5,edgecolor="black")
g=g.map(plt.scatter,"Criticsratings","AudienceRatings",**kws)
g.set(xlim=(0,90),ylim=(0,90))
for ax in g.axes.flat:
    ax.plot((20,70),(20,70))
```



In [131]:

```
g=sns.FacetGrid(df,row="Genre",col="Year",hue="Genre")
kws=dict(s=10,linewidth=.5,edgecolor="black")
g=g.map(plt.scatter,"Criticsratings","AudienceRatings",**kws)
g.set(xlim=(0,90),ylim=(0,90))
for ax in g.axes.flat:
    ax.plot((10,100),(10,100),c="green",ls="--")
```



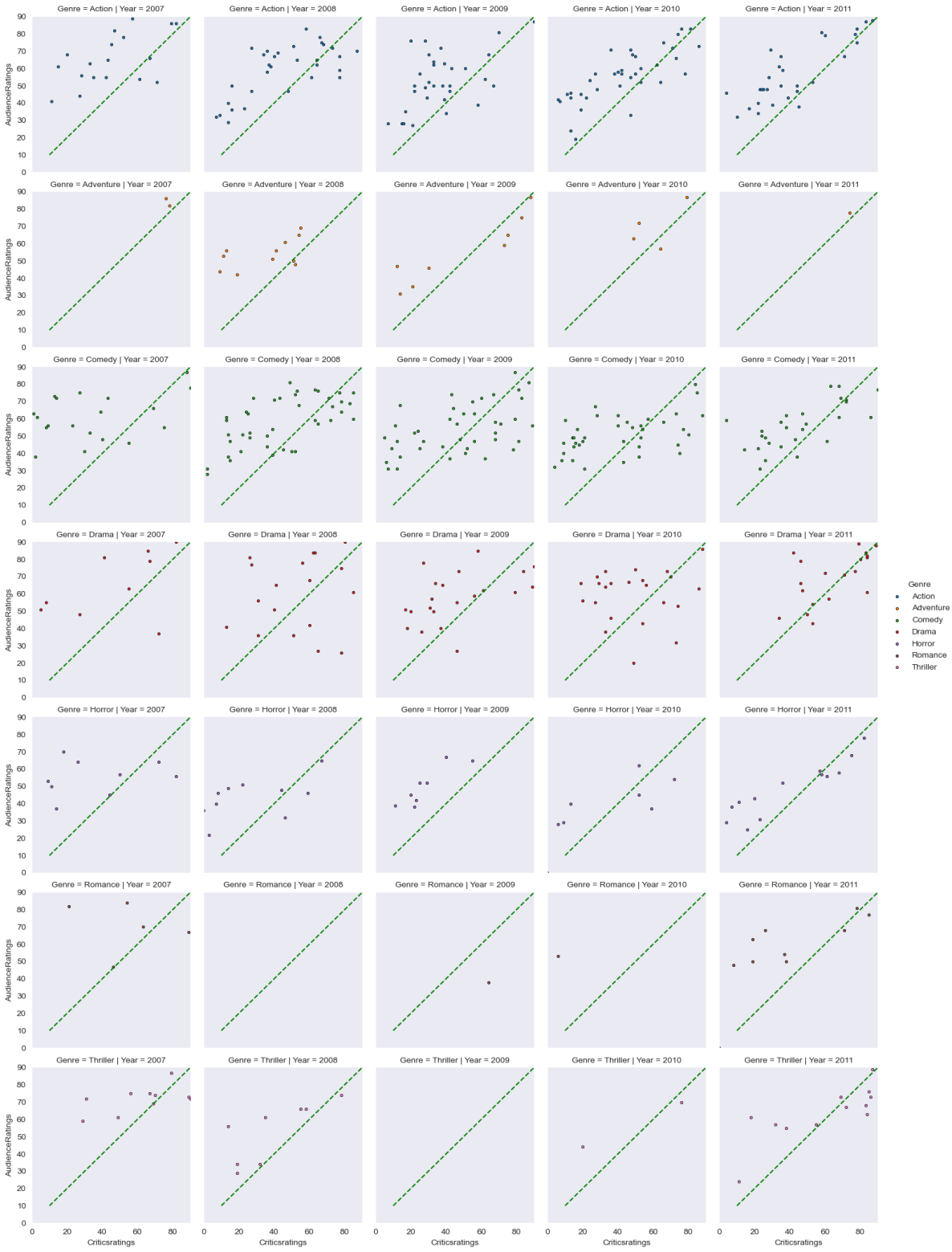
Insight:so this diagonal helps to detect where audince ratimg is more (if upper the diagonal more point) or vice versa

In [133]:

```
#adding ligend
g=sns.FacetGrid(df,row="Genre",col="Year",hue="Genre")
kws=dict(s=10,linewidth=.5,edgecolor="black")
g=g.map(plt.scatter,"Criticsratings","AudienceRatings",**kws)
g.set(xlim=(0,90),ylim=(0,90))
for ax in g.axes.flat:
    ax.plot((10,100),(10,100),c="green",ls="--")
g.add_legend()
```

Out[133]:

```
<seaborn.axisgrid.FacetGrid at 0x7faa3d1818a0>
```

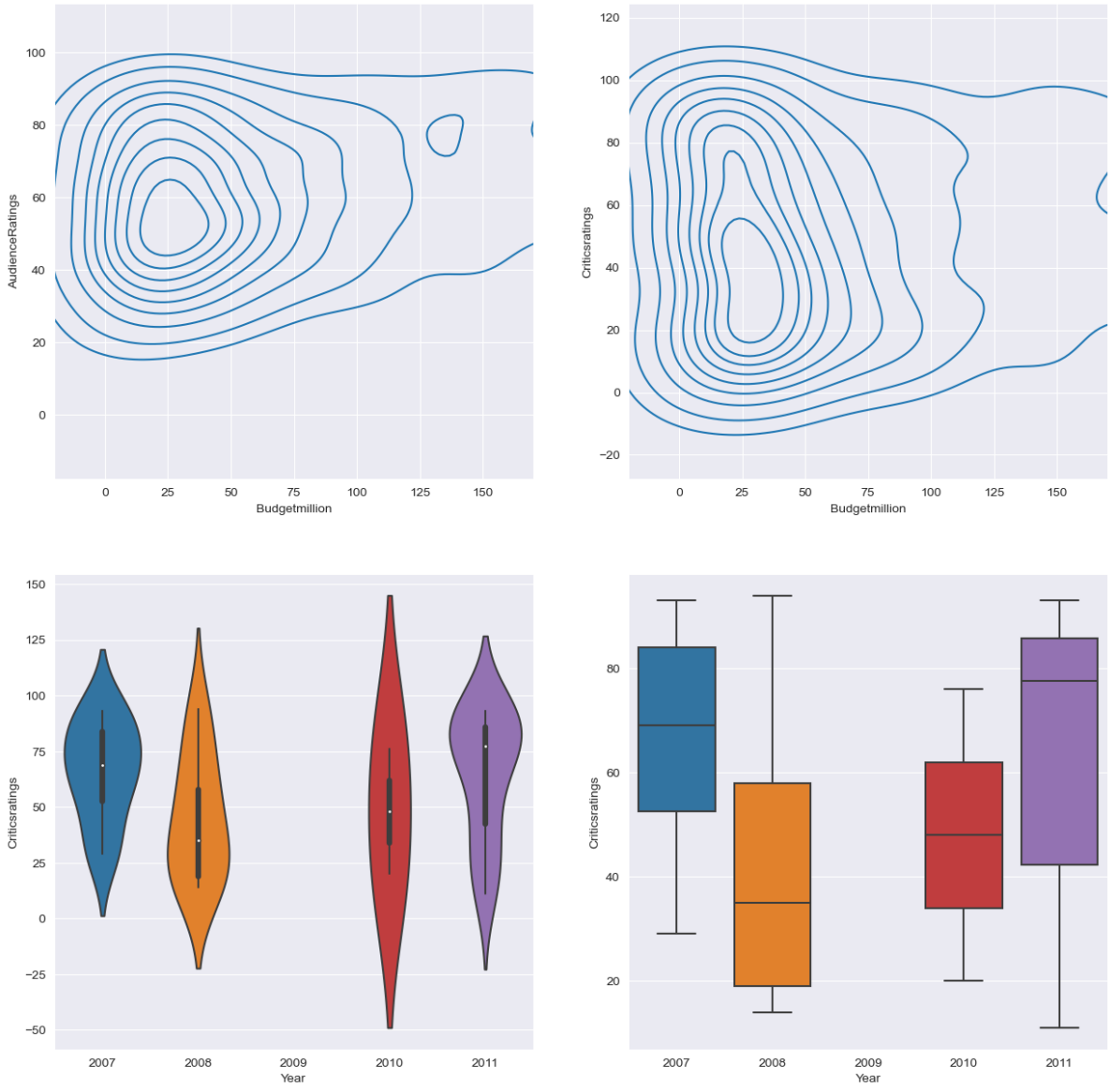


Dashboard

Dashboard is combination of chart, a figure with multiple chart, we wwill use subplot function to create dashboard

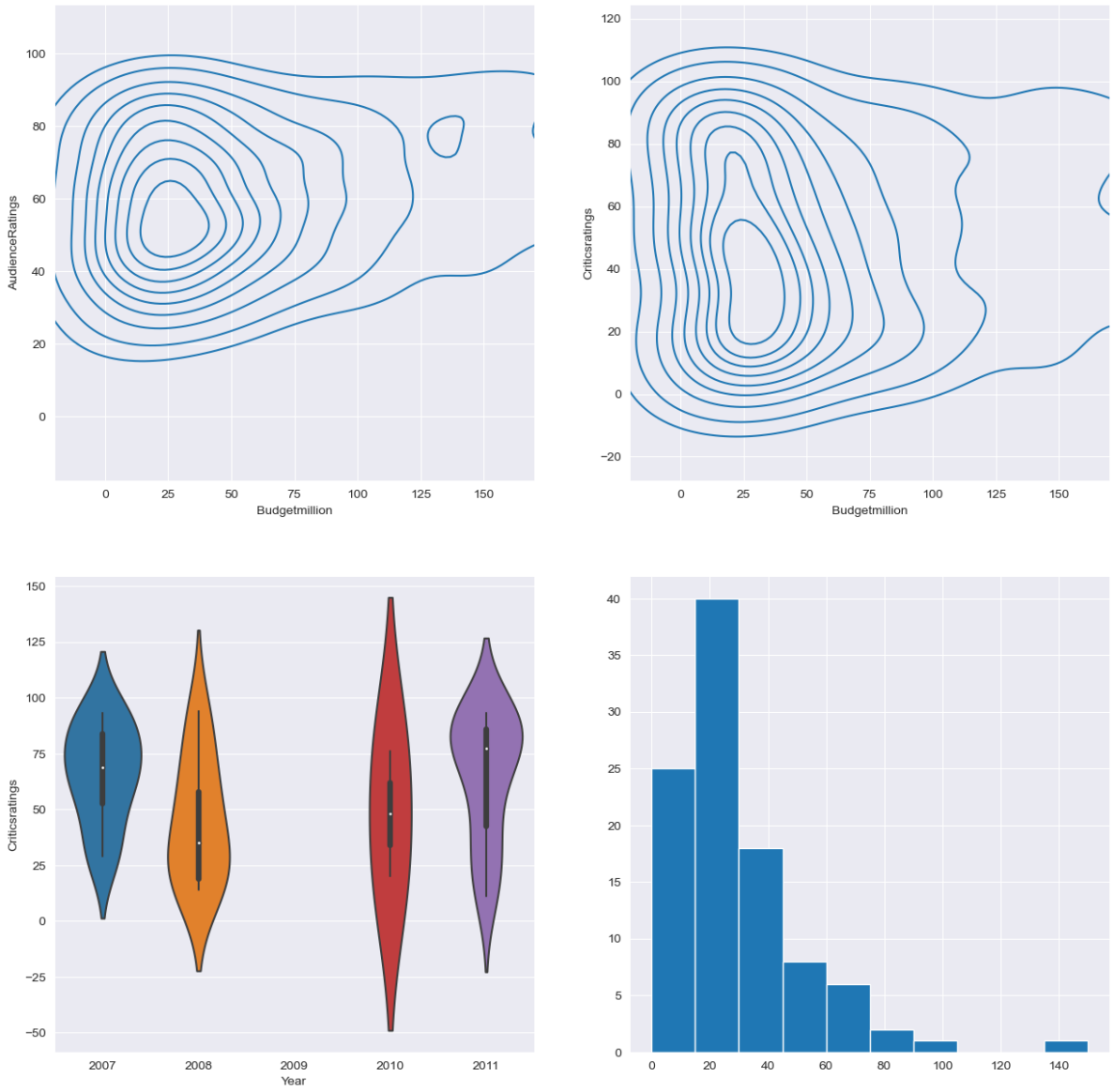
In [166]:

```
sns.set_style("darkgrid")
f,axes=plt.subplots(2,2,figsize=(15,15))
vis3 = sns.kdeplot(data=df, x="Budgetmillion", y="AudienceRatings",ax=axes[0,0])
vis4 = sns.kdeplot(data=df, x="Budgetmillion", y="Criticsratings",ax=axes[0,1])
vis3.set(xlim=(-20,170))
vis4.set(xlim=(-20,170))
v1=sns.violinplot(data=Thriller,x="Year",y="Criticsratings",ax=axes[1,0])
b=sns.boxplot(data=Thriller,x="Year",y="Criticsratings",ax=axes[1,1])
```



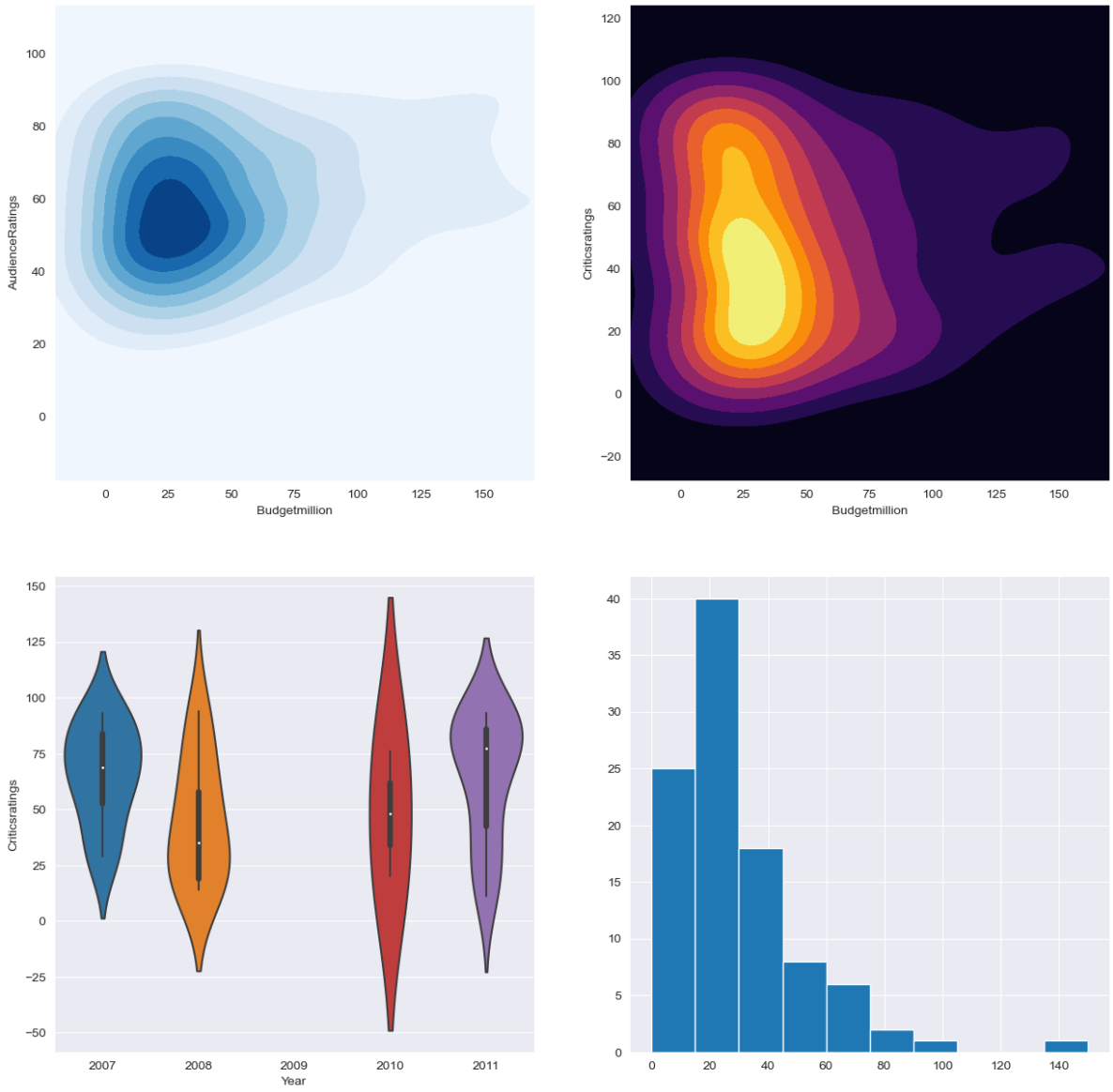
In [169]:

```
#if you want to add std.plot to the seborn plot , you have to  apply a different
sns.set_style("darkgrid")
f,axes=plt.subplots(2,2,figsize=(15,15))
vis3 = sns.kdeplot(data=df, x="Budgetmillion", y="AudienceRatings",ax=axes[0,0])
vis4 = sns.kdeplot(data=df, x="Budgetmillion", y="Criticsratings",ax=axes[0,1])
vis3.set(xlim=(-20,170))
vis4.set(xlim=(-20,170))
v1=sns.violinplot(data=Thriller,x="Year",y="Criticsratings",ax=axes[1,0])
axes[1,1].hist(df[df. Genre=="Drama"].Budgetmillion)
plt.show()
```



In [174]:

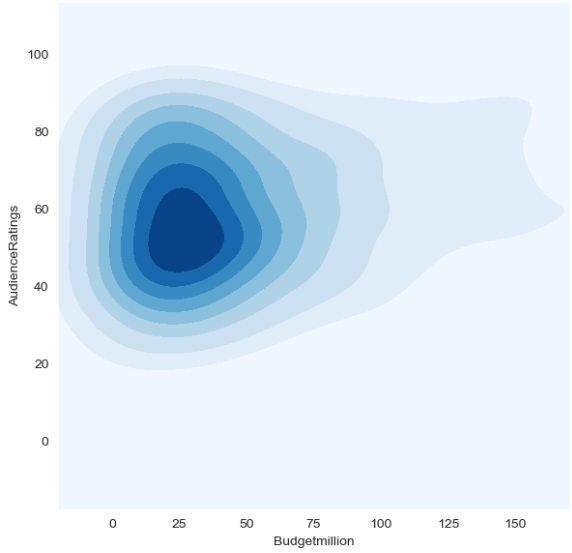
```
##styling Dashboard
sns.set_style("darkgrid")
f,axes=plt.subplots(2,2,figsize=(15,15))
vis3 = sns.kdeplot(data=df, x="Budgetmillion", y="AudienceRatings",shade=True,sh
vis4 = sns.kdeplot(data=df, x="Budgetmillion", y="Criticsratings",shade=True,sha
vis3.set(xlim=(-20,170))
vis4.set(xlim=(-20,170))
v1=sns.violinplot(data=Thriller,x="Year",y="Criticsratings",ax=axes[1,0])
axes[1,1].hist(df[df. Genre=="Drama"].Budgetmillion)
plt.show()
```

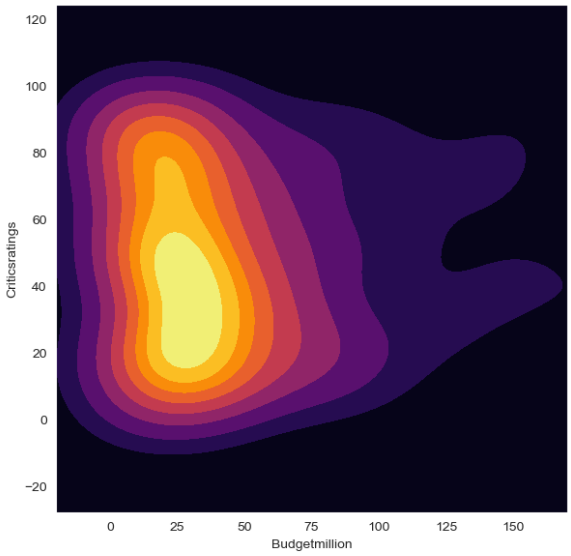


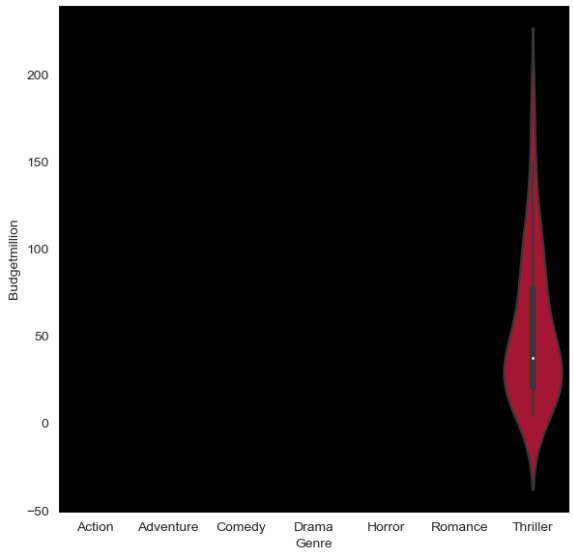
so in here , you can not change the background color of last two, if you do , you can pass the kewrod dictionaries to set style

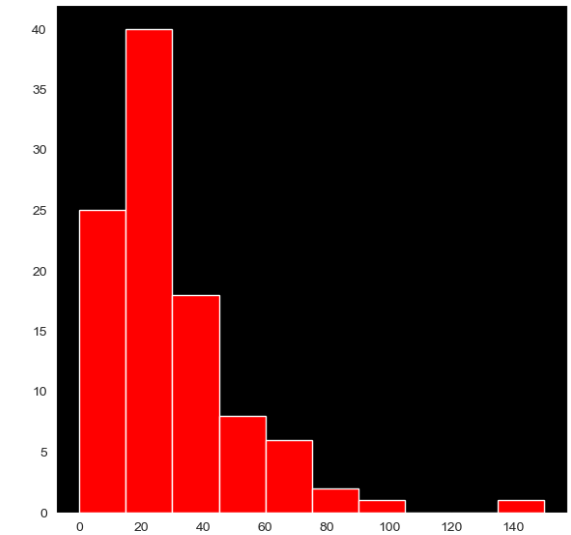
In [187]:

```
sns.set_style("dark", {"axes.facecolor": "black"})
f, axes = plt.subplots(2, 2, figsize=(15, 15))
vis3 = sns.kdeplot(data=df, x="Budgetmillion", y="AudienceRatings", shade=True,
vis4 = sns.kdeplot(data=df, x="Budgetmillion", y="Criticsratings", shade=True, s
vis3.set(xlim=(-20, 170))
vis4.set(xlim=(-20, 170))
v1 = sns.violinplot(data=Thriller, x="Genre", y="Budgetmillion", ax=axes[1, 0],p
axes[1, 1].hist(df[df.Genre == "Drama"].Budgetmillion,color="red")
plt.show()

```



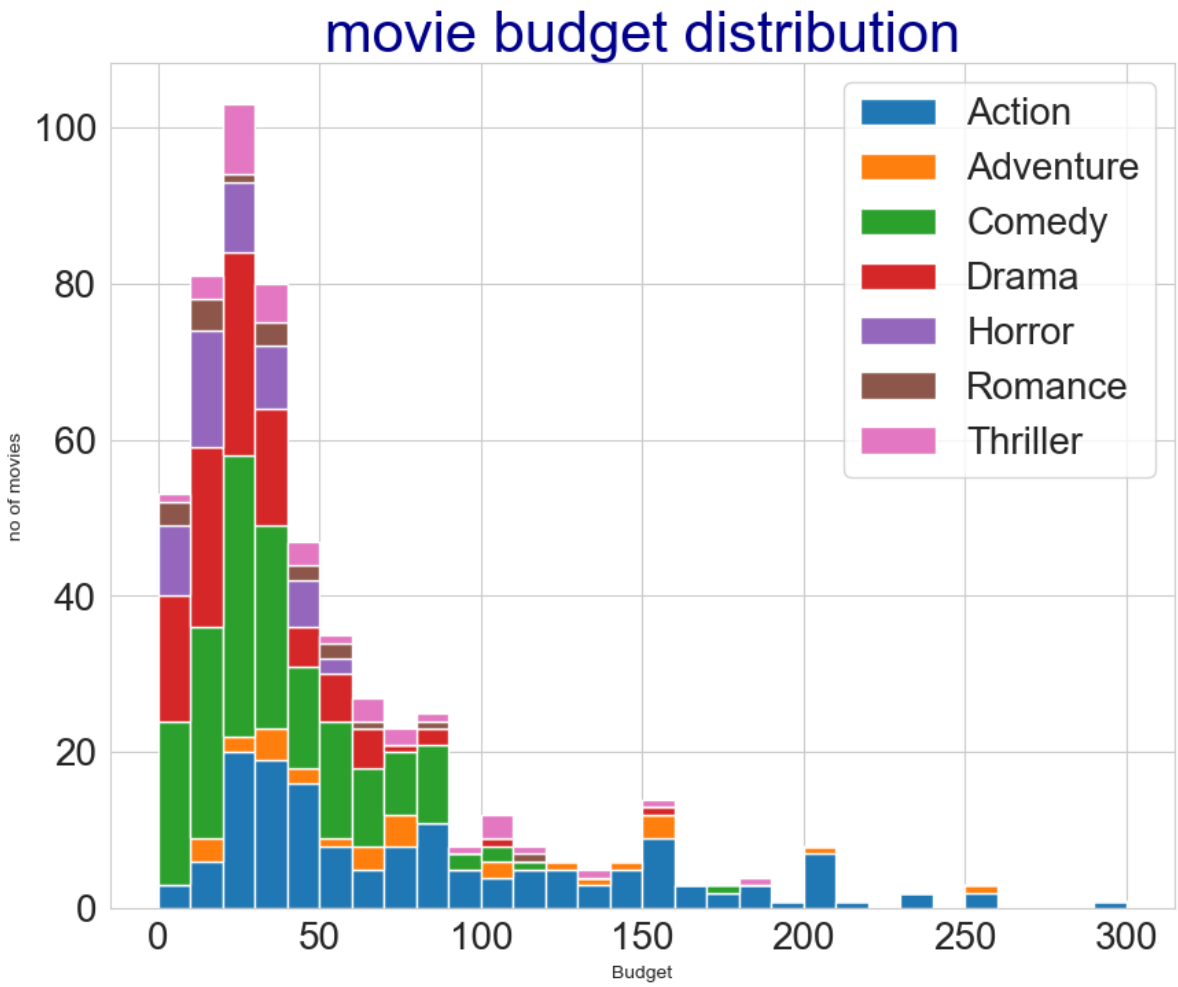






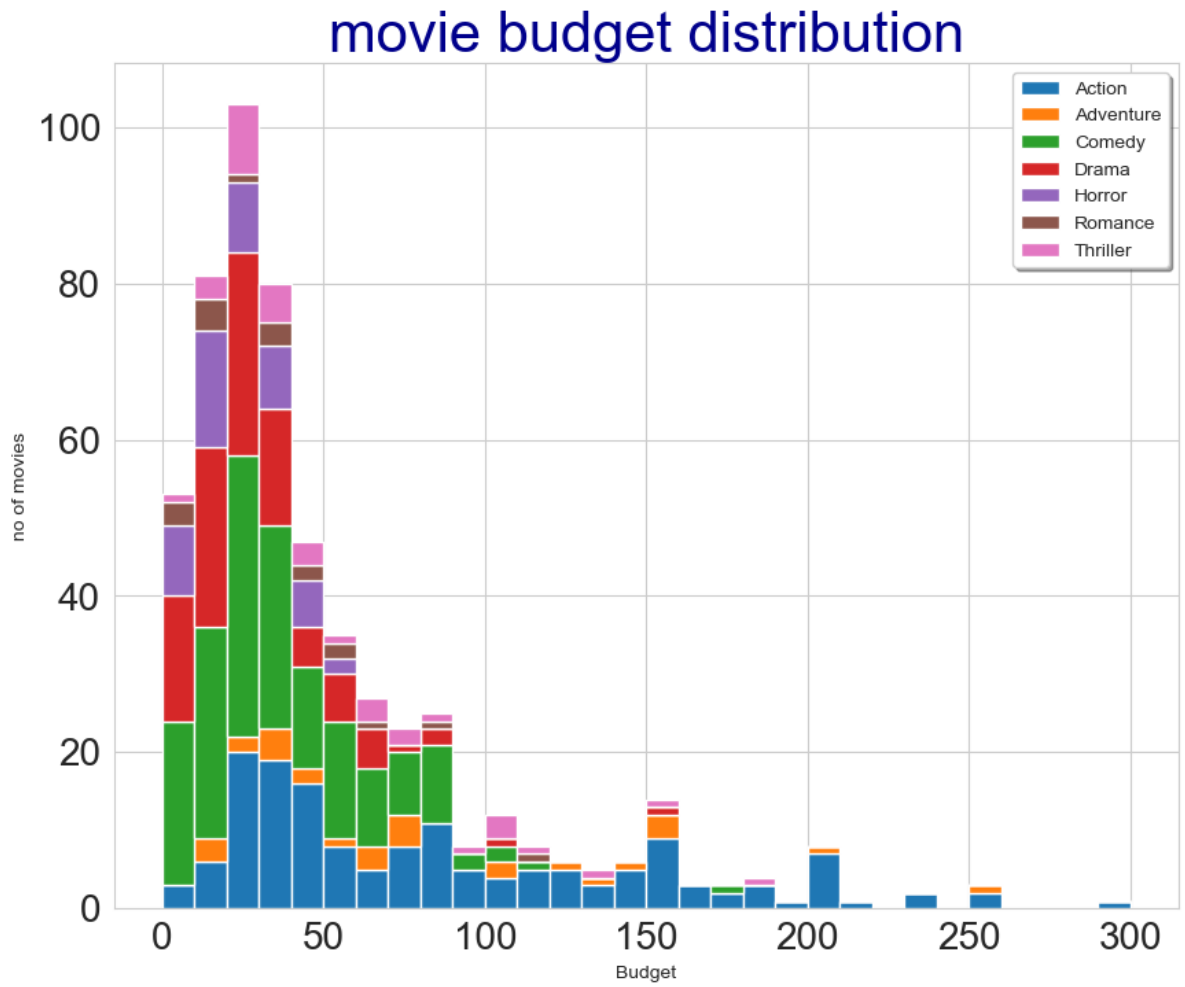
In [207]:

```
#improving the chart
import seaborn as sns
import matplotlib.pyplot as plt
list1 = list()
mylabels=list()

sns.set_style("whitegrid")
f, axes = plt.subplots()
f.set_size_inches(10, 8)
for gen in df.Genre.cat.categories:
    list1.append(df[df.Genre == gen].Budgetmillion)
    mylabels.append(gen)
plt.hist(list1,bins=30,stacked=True,rwidth=1,label=mylabels)
plt.title("movie budget distribution",fontsize=30,color="darkblue")
plt.ylabel("no of movies")
plt.xlabel("Budget")
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.legend(fontsize=20)
plt.show()
```



In [209]:

```
#improving the chart
import seaborn as sns
import matplotlib.pyplot as plt
list1 = list()
mylabels=list()

sns.set_style("whitegrid")
f, axes = plt.subplots()
f.set_size_inches(10, 8)
for gen in df.Genre.cat.categories:
    list1.append(df[df.Genre == gen].Budgetmillion)
    mylabels.append(gen)
plt.hist(list1,bins=30,stacked=True,rwidth=1,label=mylabels)
plt.title("movie budget distribution",fontsize=30,color="darkblue")
plt.ylabel("no of movies")
plt.xlabel("Budget")
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.legend(shadow=True)
plt.show()
```



In [ ]:

```
1
```