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

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

In [3]:
```
movies.columns
```

Out[3]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

In [4]:
```
movies.head()
```

Out[4]:

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

In [5]:
```
movies.shape
```

Out[5]:
```
(559, 6)
```

In [6]:
```
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 [7]:
```
movies.columns=['Film','Genre','CriticRating','audienceRating','BudgetMillion','
```

In [8]:
```
movies.head()
```

Out[8]:

	Film	Genre	CriticRating	audienceRating	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 [9]:
```
movies.describe()
```

Out[9]:

	CriticRating	audienceRating	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 [10]:
```
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   BudgetMillion   559 non-null    int64 
 5   Year            559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [11]:
```
movies.Film=movies.Film.astype('category')
```

In [12]:
```
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   BudgetMillion   559 non-null    int64   
 5   Year            559 non-null    int64   
dtypes: category(1), int64(4), object(1)
memory usage: 43.6+ KB
```

In [13]:
```
movies.Genre=movies.Genre.astype('category')
movies.Year=movies.Year.astype('category')
```

In [14]:
```
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   BudgetMillion   559 non-null    int64   
 5   Year            559 non-null    category
dtypes: category(3), int64(3)
memory usage: 36.5 KB
```

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

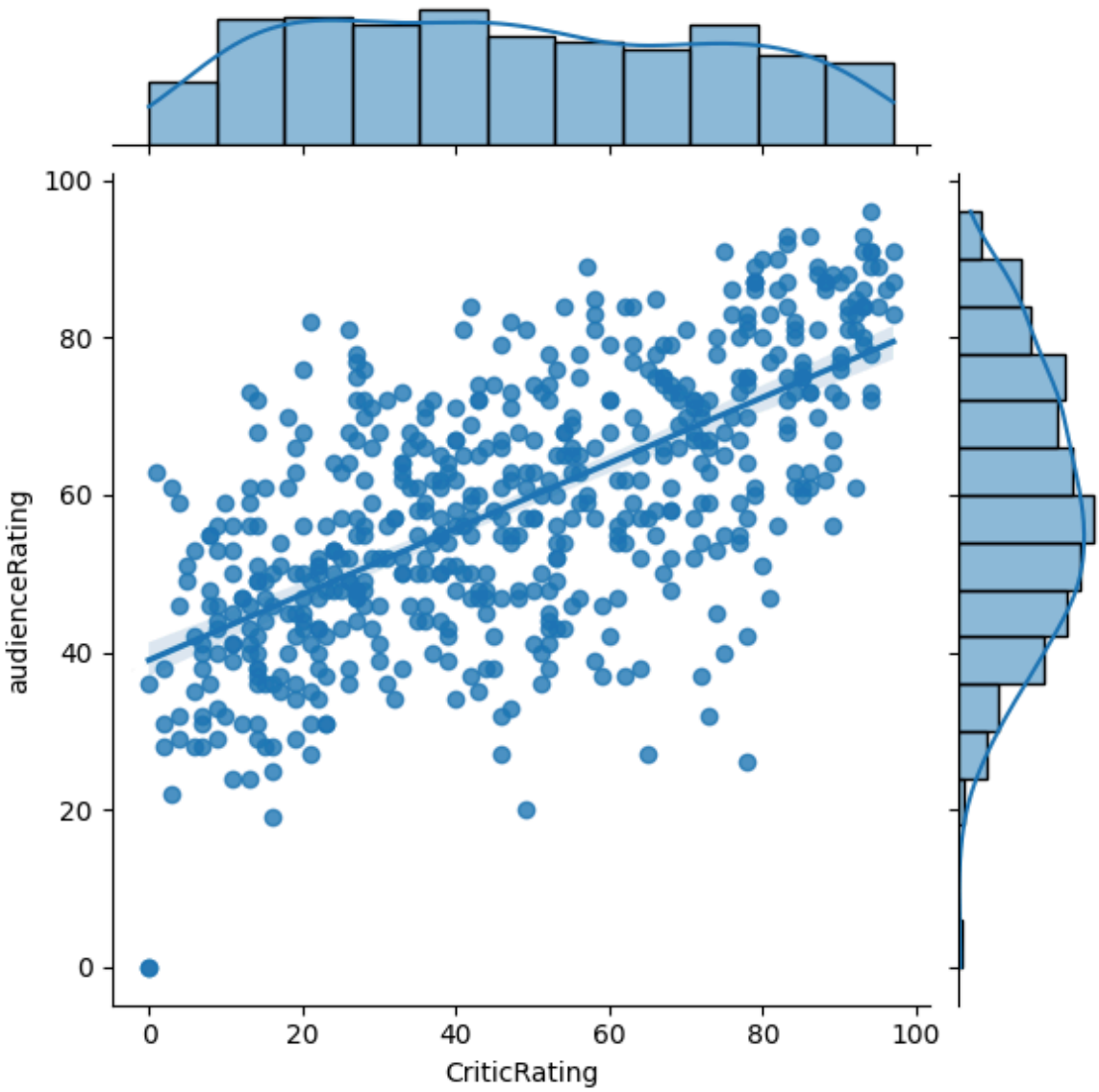
Out[15]:

	CriticRating	audienceRating	BudgetMillion
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

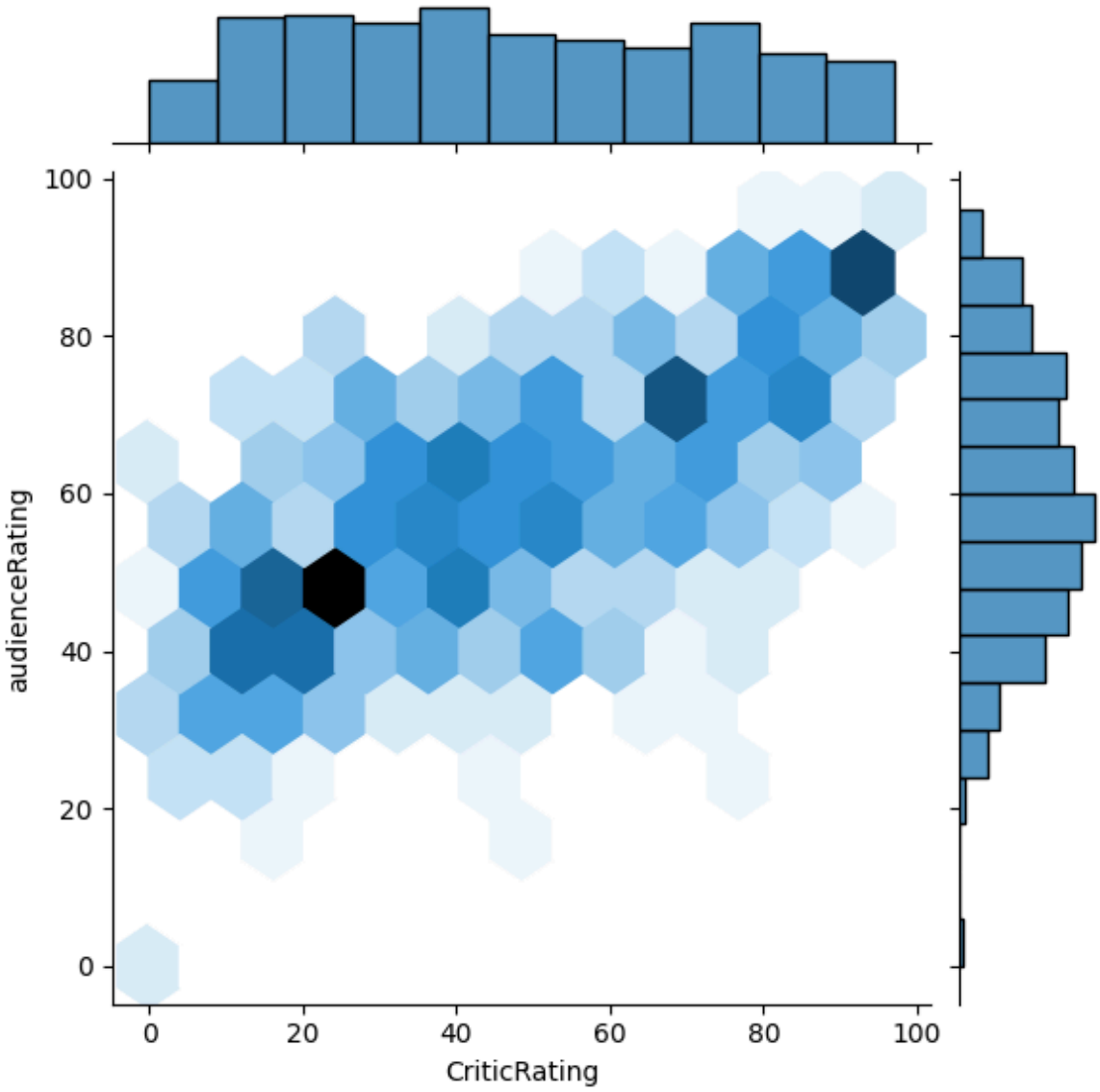
In [16]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [17]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='audienceRating',kind='reg') #thi
#audience rating are most watched
```

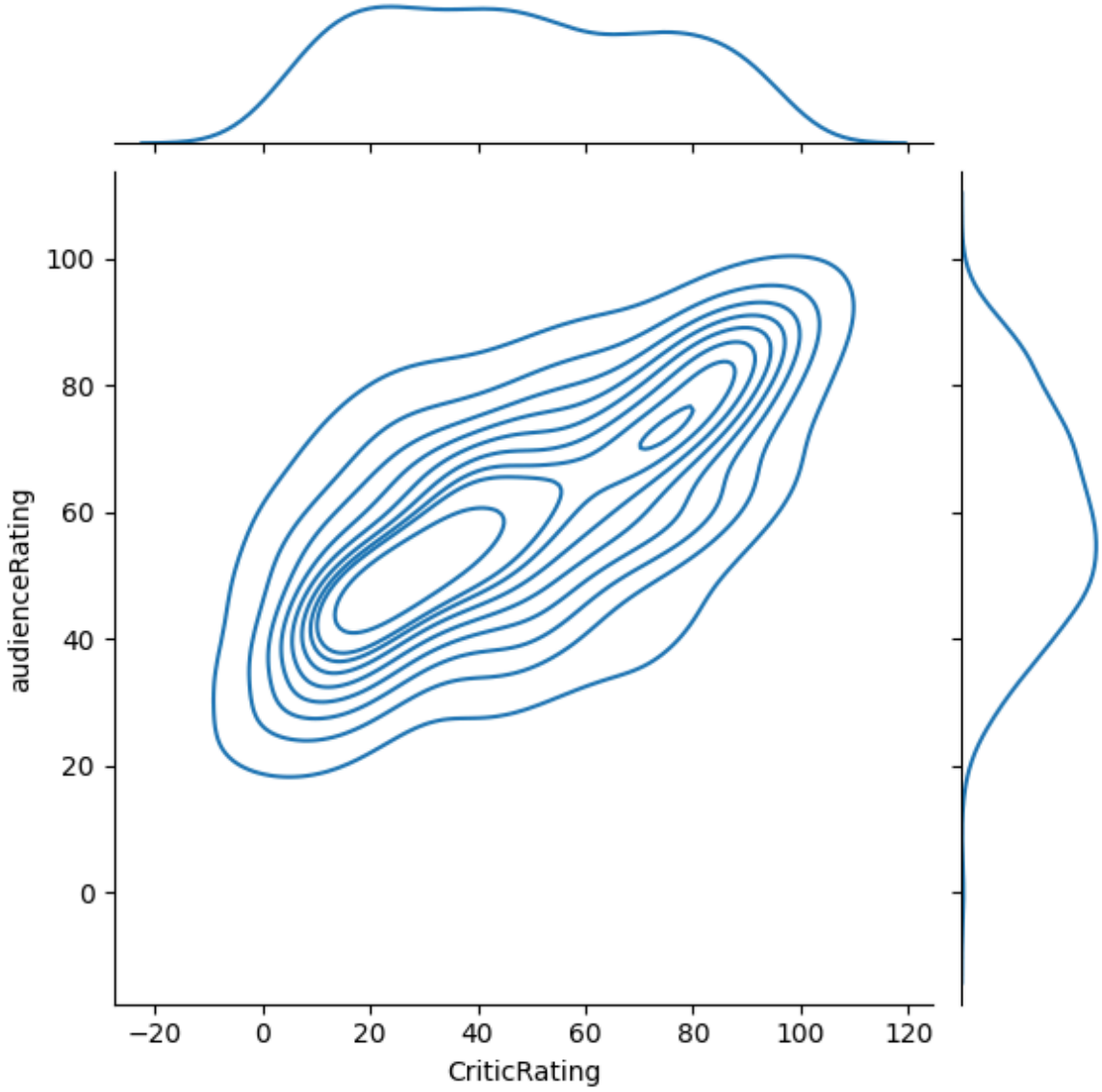
```
#joint plots are used for bi variante analysis
```



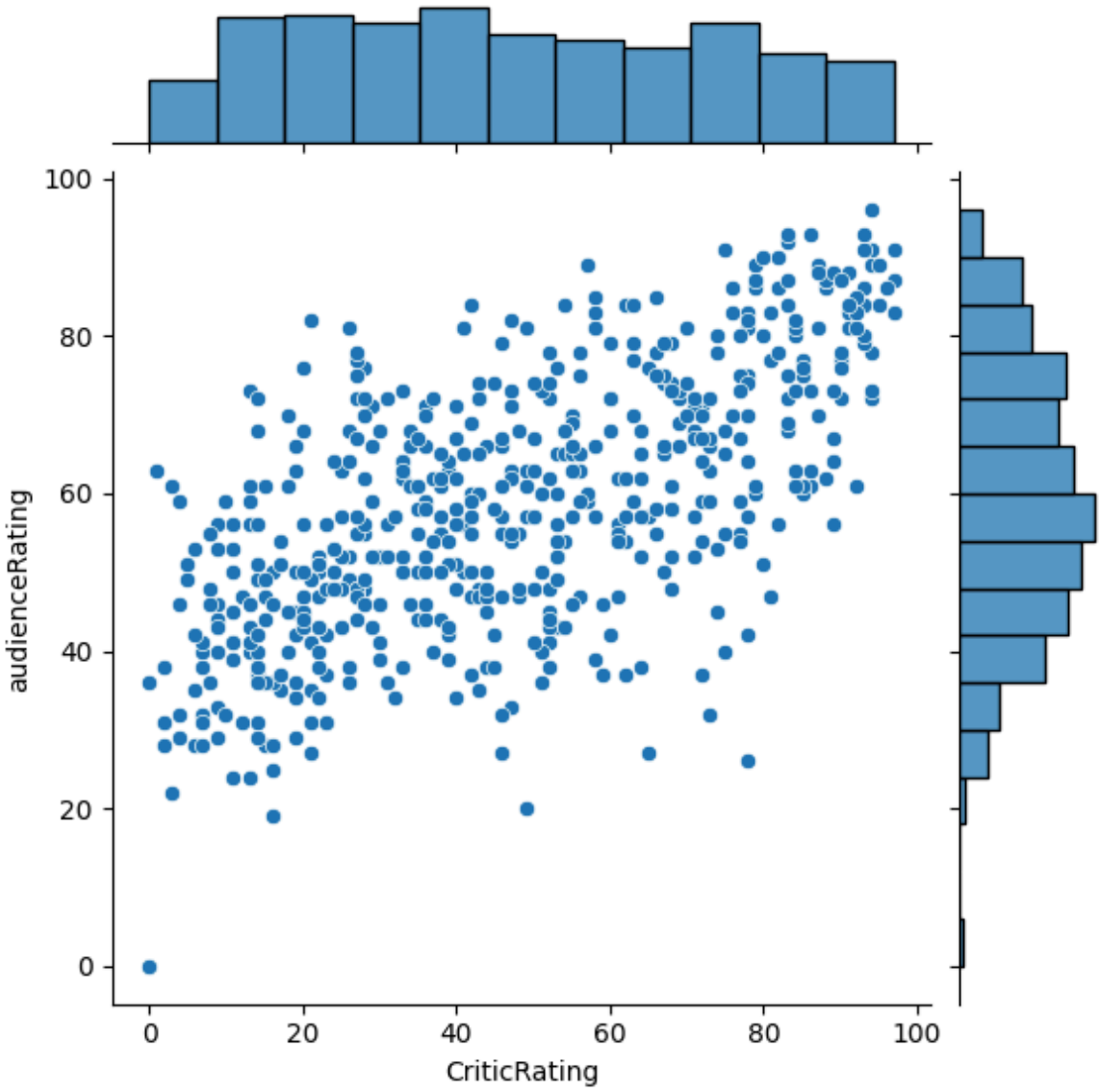
In [18]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='audienceRating',kind='hex')
```



In [19]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='audienceRating',kind='kde')
```

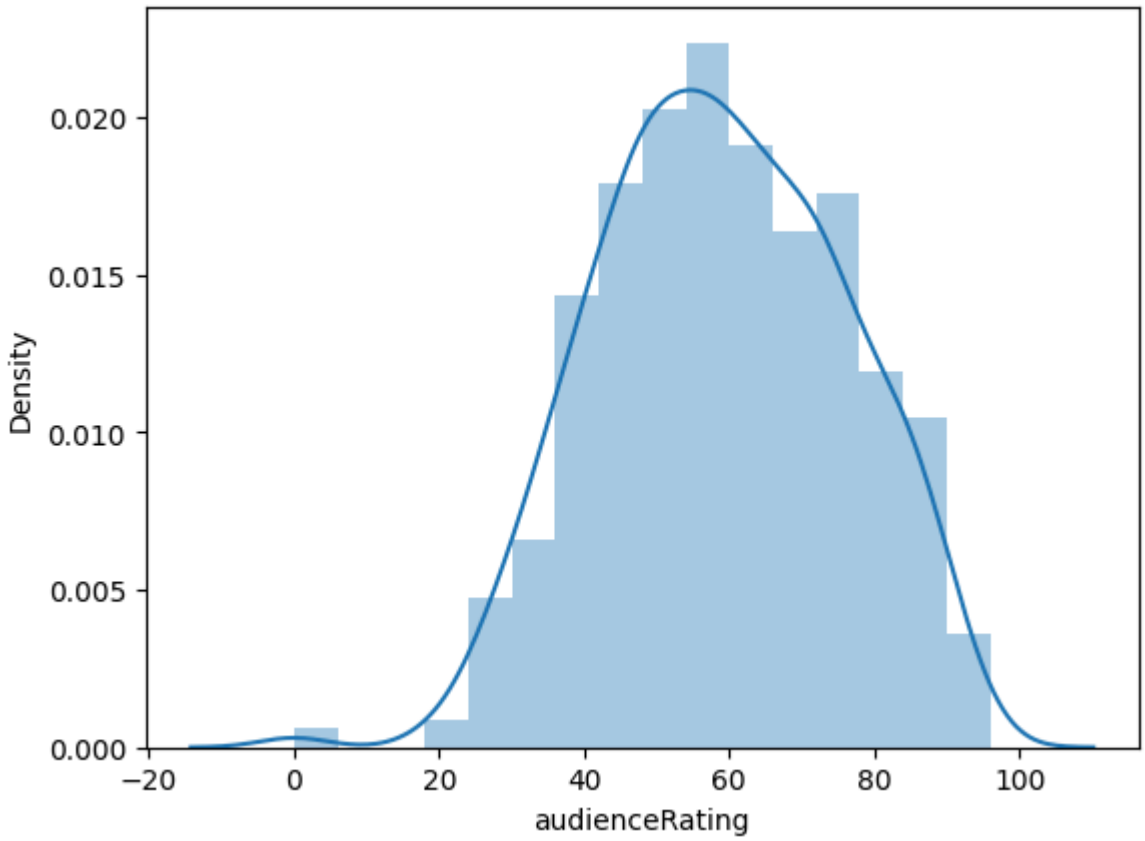


In [20]:
```
j=sns.jointplot(data=movies,x='CriticRating',y='audienceRating',kind='scatter')
```

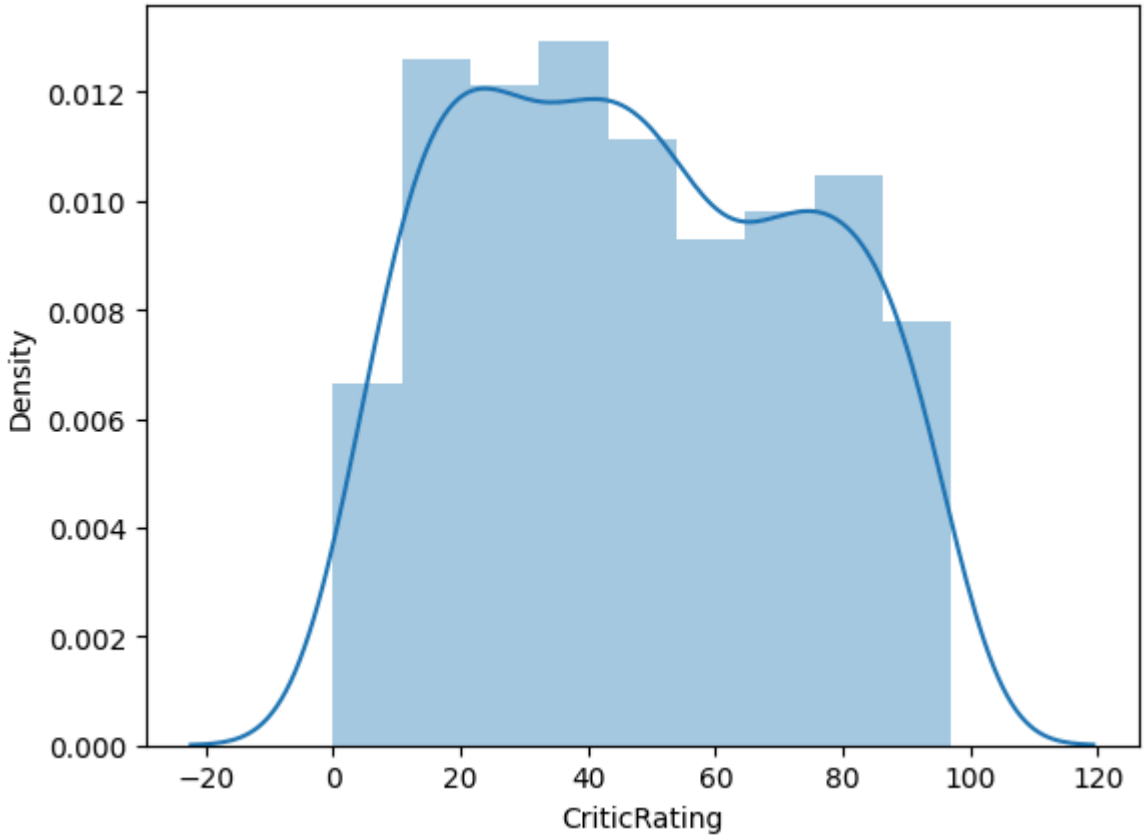


uniform,normal,probability are three types of distributions

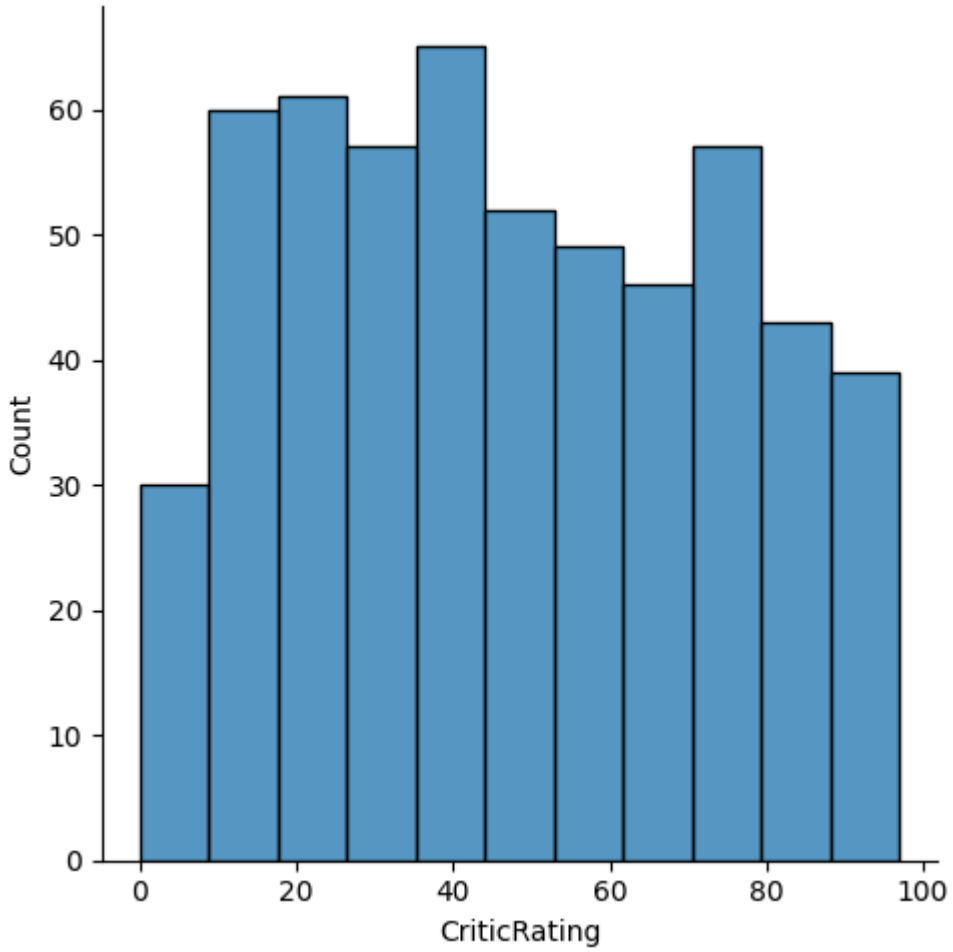
In [21]:
```
m=sns.distplot(movies.audienceRating)#normal distribution
```



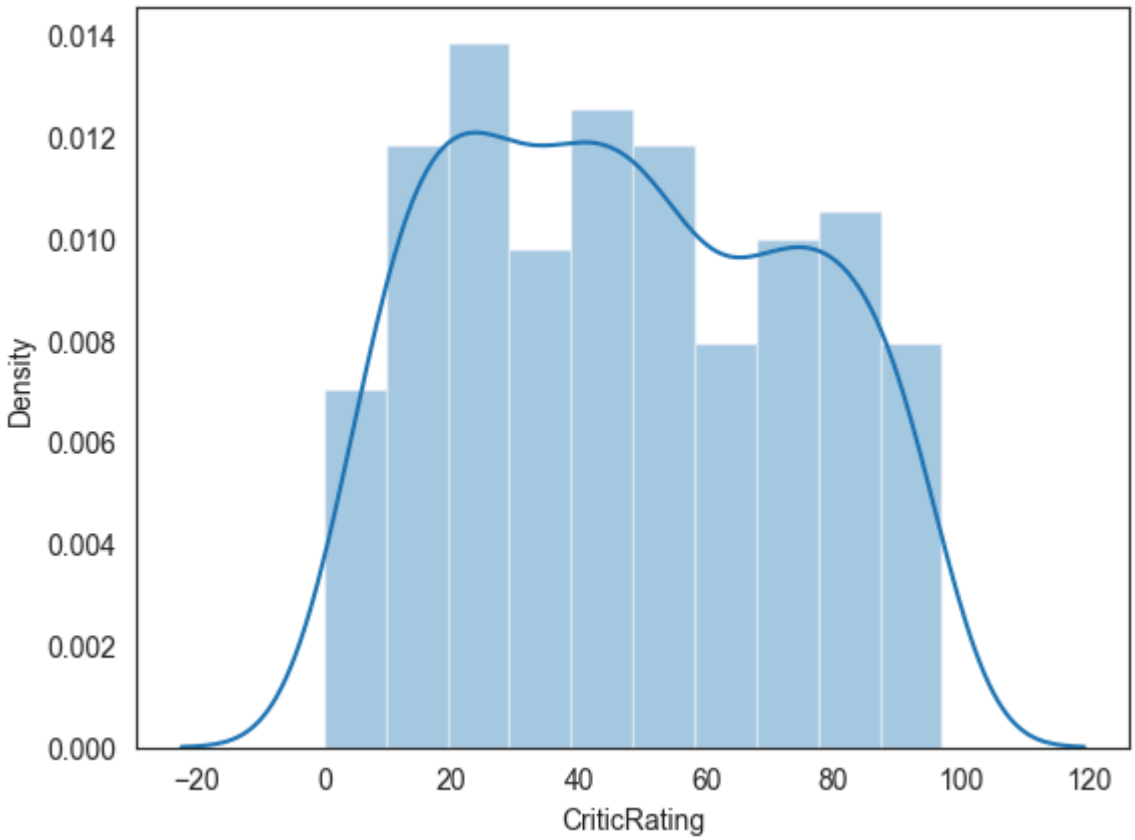
In [22]:
```
m1=sns.distplot(movies.CriticRating) #uniform distribution
```



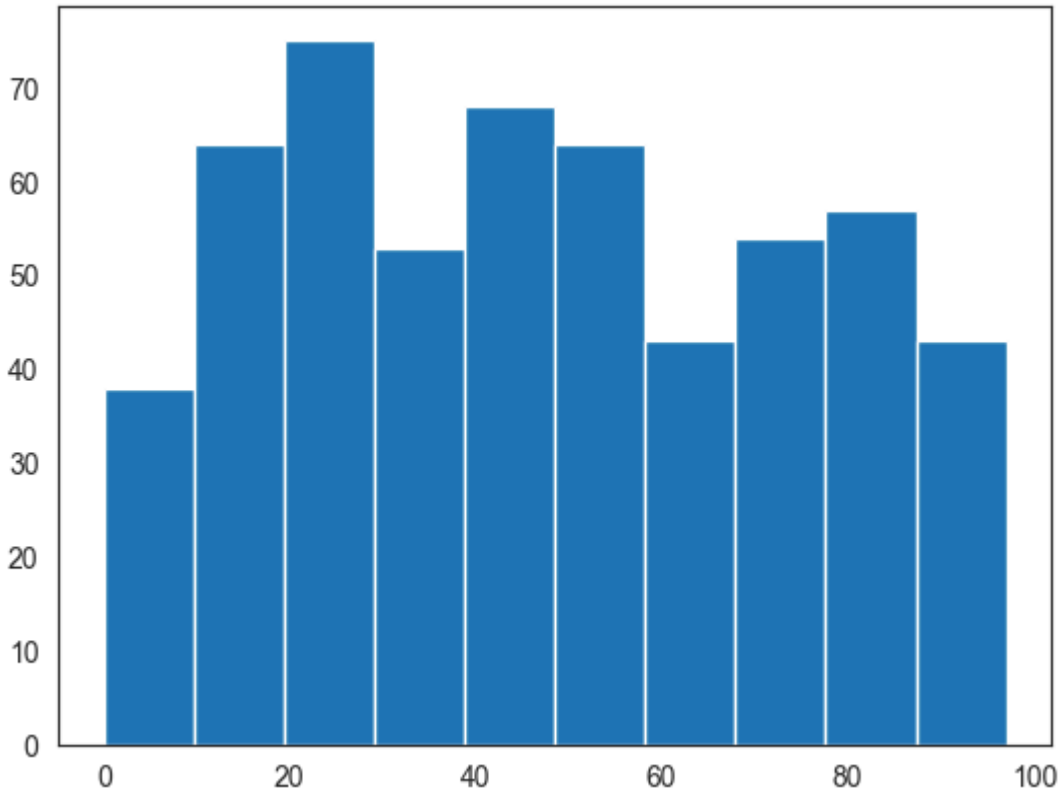
In [23]:
```
m2=sns.displot(movies.CriticRating)
```



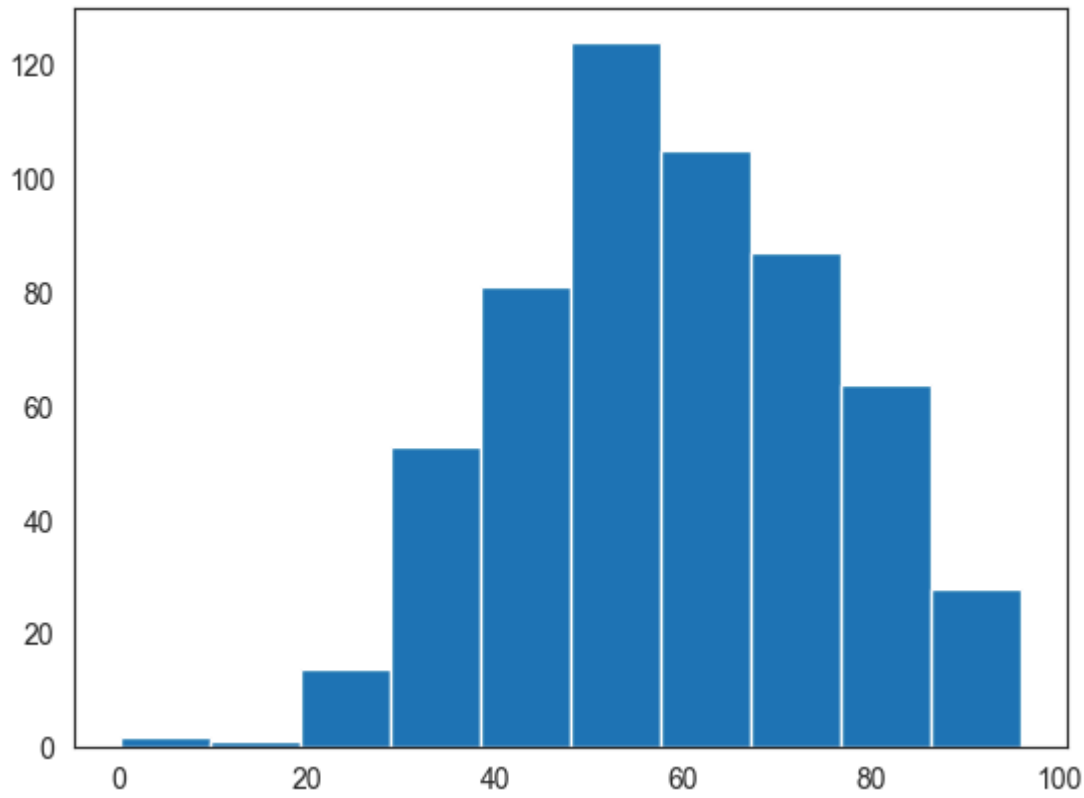
In [24]:
```
sns.set_style('white')
m3=sns.distplot(movies.CriticRating,bins=10)
```



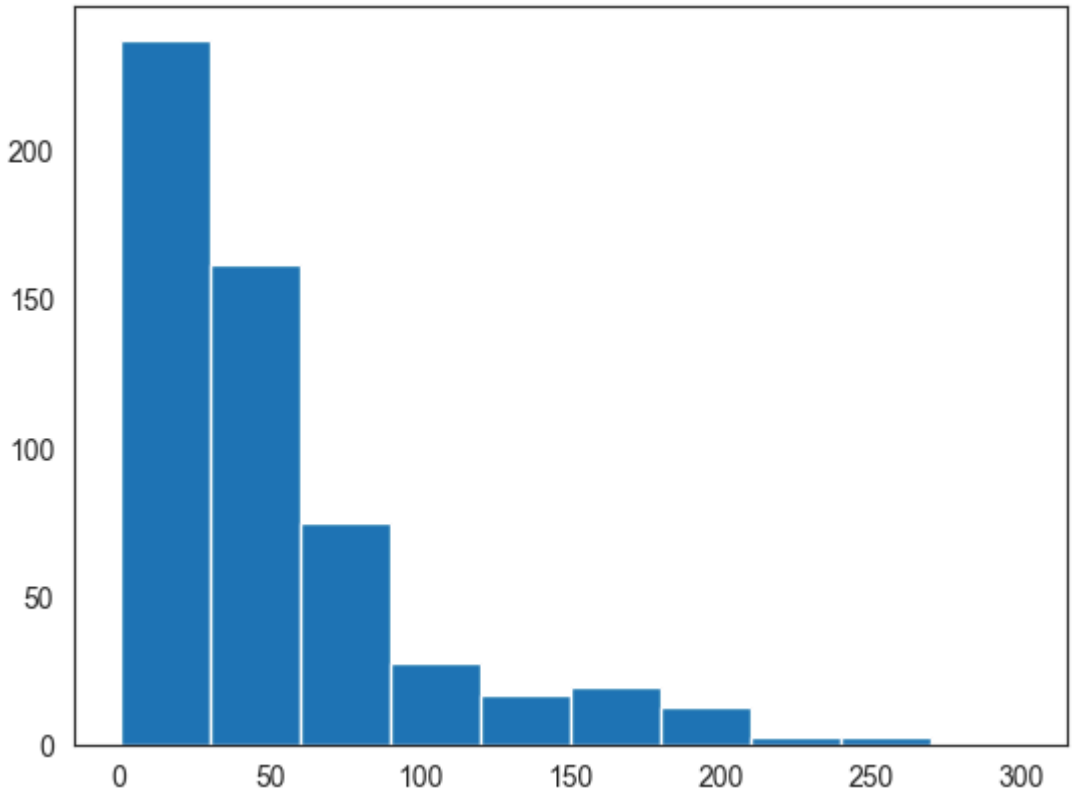
In [25]:
```
plt.hist(movies.CriticRating)
plt.show()
```



In [26]:
```
plt.hist(movies.audienceRating)
plt.show()
```



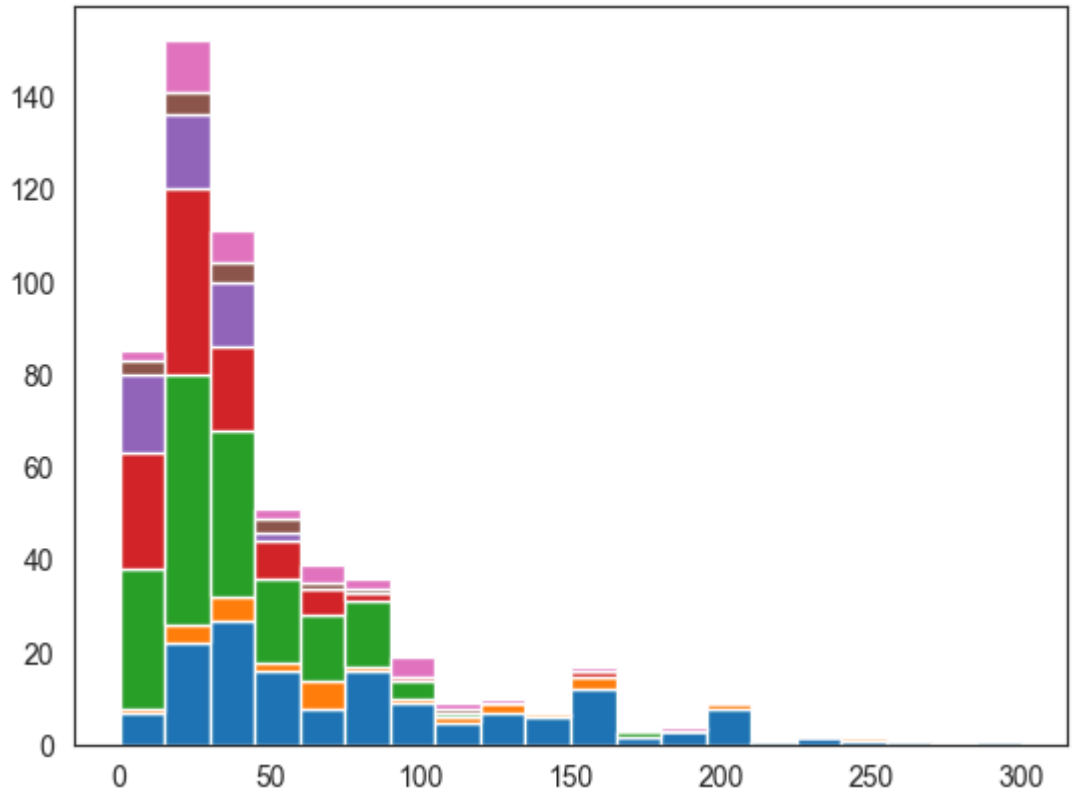
In [27]:
```
plt.hist(movies.BudgetMillion)
plt.show()
```



In [28]:
```
movies. Genre.unique()
```

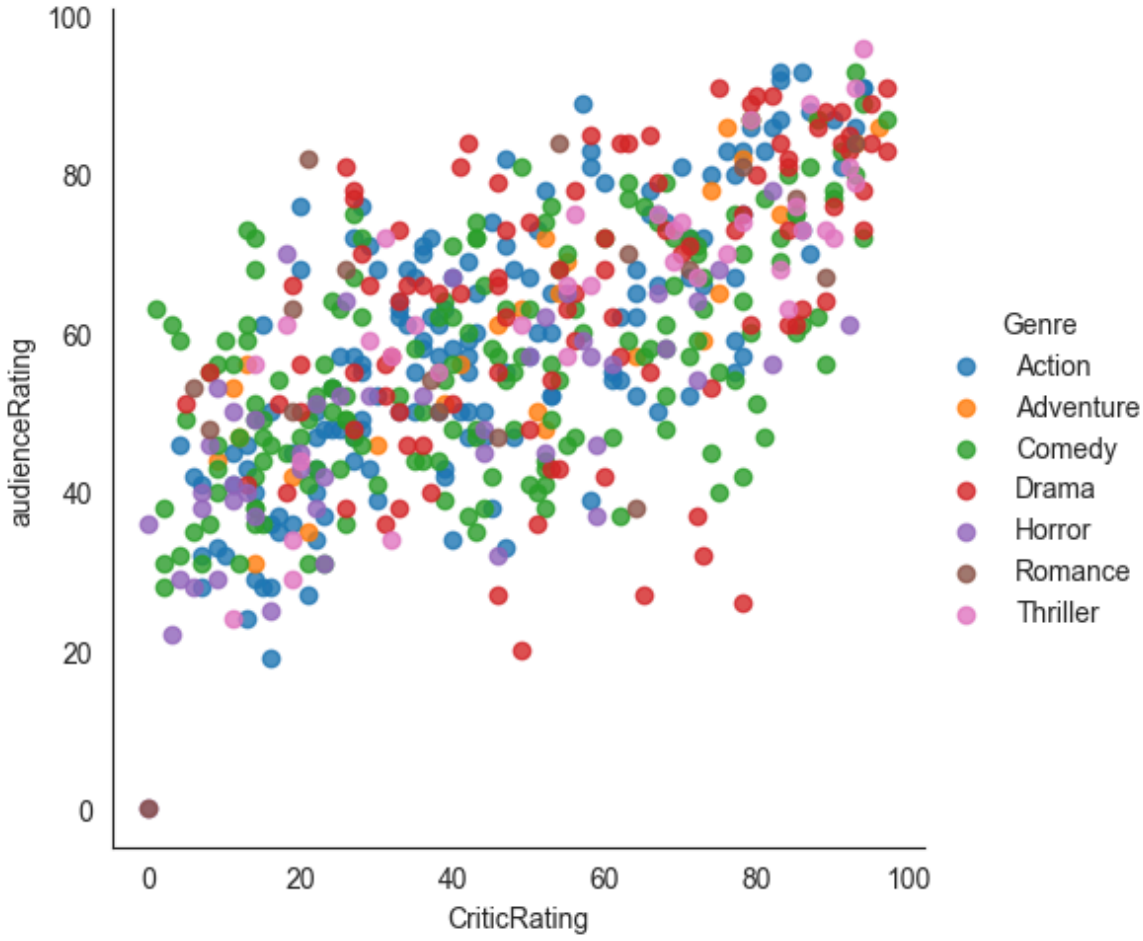
Out[28]:
```
['Comedy', 'Adventure', 'Action', 'Horror', 'Drama', 'Romance', 'Thriller']
Categories (7, object): ['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'R
omance', 'Thriller']
```

In [29]:
```
plt.hist([movies[movies.Genre=='Action'].BudgetMillion,
          movies[movies.Genre=='Adventure'].BudgetMillion,
          movies[movies.Genre=='Comedy'].BudgetMillion,
          movies[movies.Genre=='Drama'].BudgetMillion,
          movies[movies.Genre=='Horror'].BudgetMillion,
          movies[movies.Genre=='Romance'].BudgetMillion,
          movies[movies.Genre=='Thriller'].BudgetMillion],
          bins=20,stacked=True)
plt.show()
```



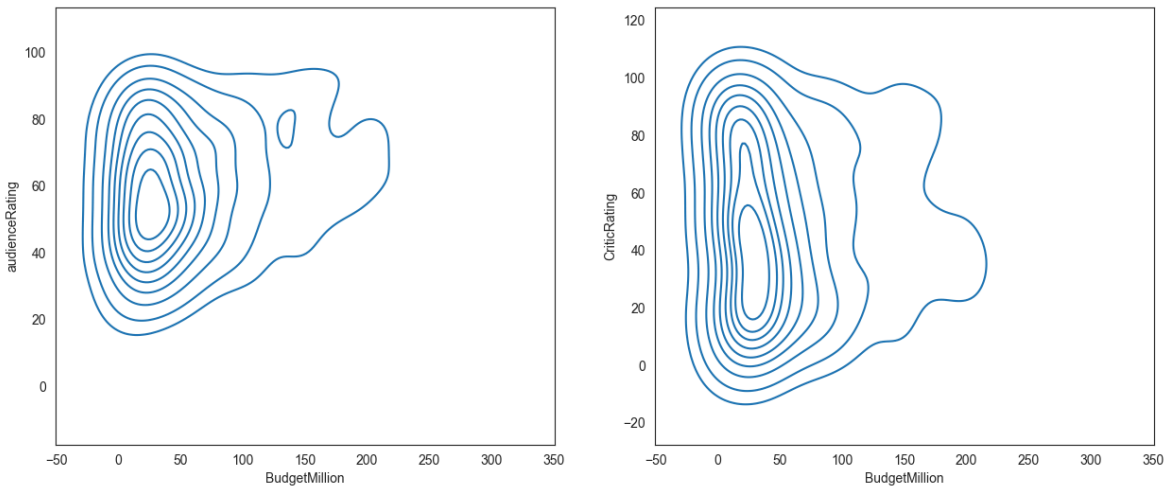
In [30]:
```
sns.lmplot(data=movies,x='CriticRating',y='audienceRating',hue='Genre',fit_reg=F
```

Out[30]:
```
<seaborn.axisgrid.FacetGrid at 0x17f0a9876e0>
```

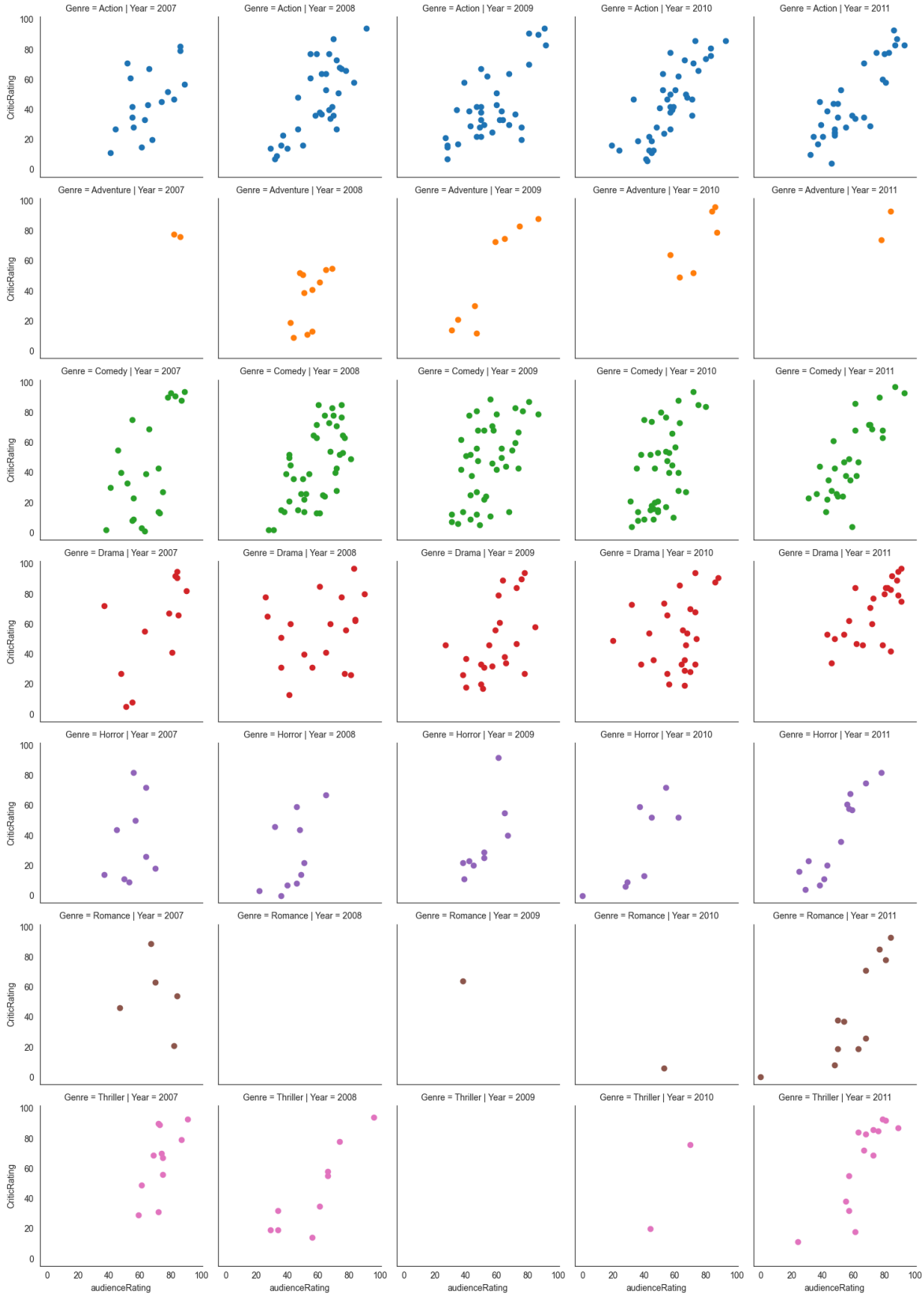


In [31]:
```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,6))
k1=sns.kdeplot(data=movies,x='BudgetMillion',y='audienceRating',ax=ax1)
```

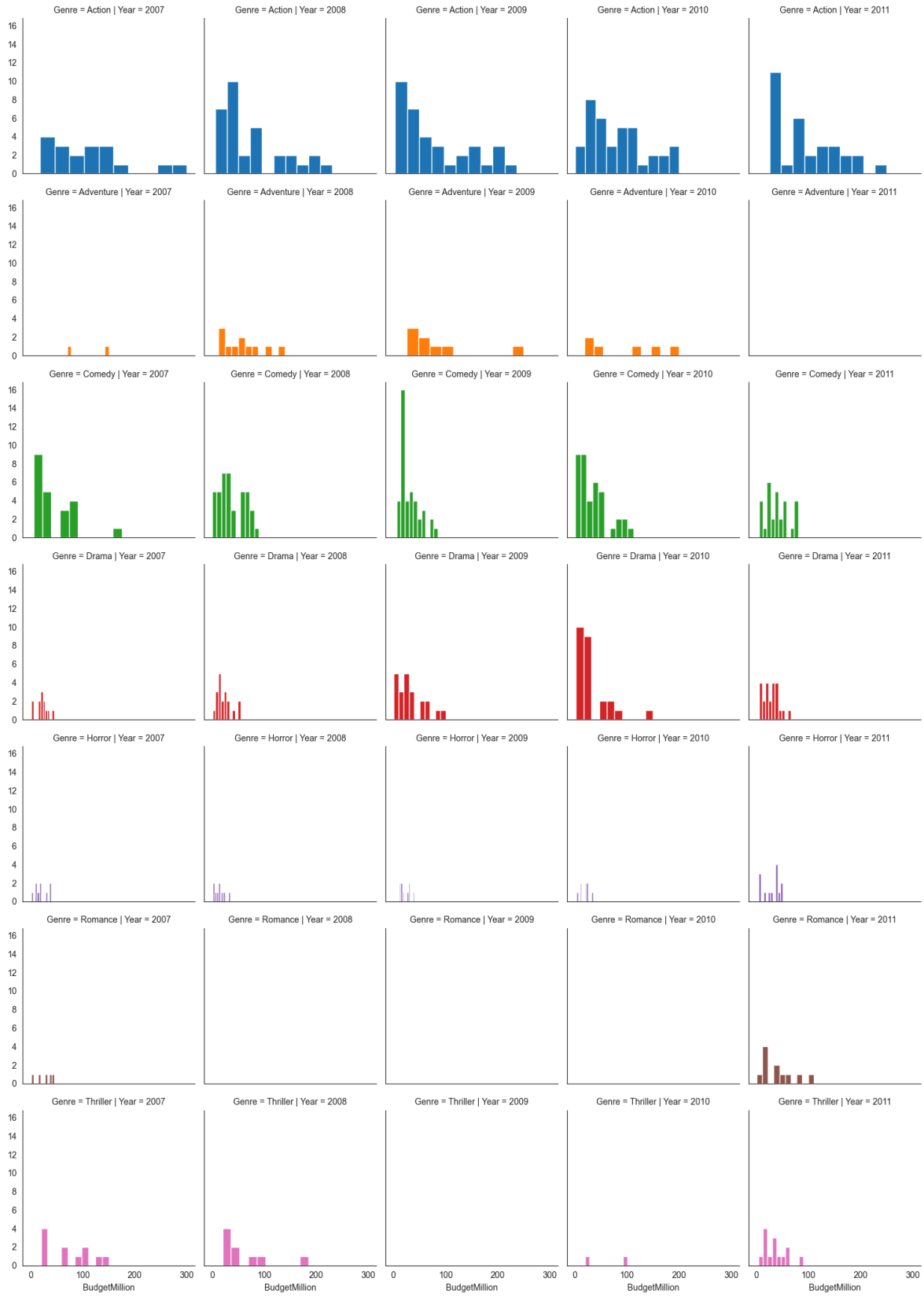
```
k2=sns.kdeplot(data=movies,x='BudgetMillion',y='CriticRating',ax=ax2)
```



In [32]:
```
g=sns.FacetGrid(movies,row='Genre',col='Year',hue='Genre')
g=g.map(plt.scatter,'audienceRating','CriticRating')
```



In [33]:
```
g=sns.FacetGrid(movies,row='Genre',col='Year',hue='Genre')
g=g.map(plt.hist,'BudgetMillion')
```


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
BudgetMillion

In [ ]:

In [ ]:

In [ ]:

In [ ]: