Movie Rating Analytics (Advanced Visualization)

In [2]:
```
import pandas as pd
```

In [3]:
```
movies = pd.read_csv(r'D:\Full Stack Data Scientist and AI\March 28 - Seaborn Pr
```

In [4]:
```
movies
```

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

Out of 6 attributes, 2 are category rest are numerical

In [6]:
```
type(movies)
```

Out[6]:
```
pandas.core.frame.DataFrame
```

In [7]:
```
id(movies)
```

Out[7]:
```
1176748758928
```

In [8]:
```
len(movies)   #There is no missing values
```

Out[8]:
```
559
```

In [9]:
```
import numpy
```

In [10]:
```
print(numpy.__version__)
```

```
1.26.4
```

In [11]:
```
import pandas
```

In [12]:
```
print(pandas.__version__)
```

```
2.2.2
```

In [13]:
```
movies.columns
```

Out[13]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

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    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 [15]:
```
movies.shape     #Gives number of rows and columns
```

Out[15]:
```
(559, 6)
```

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

Out[16]:

	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 [17]:
```
movies.tail()
```

Out[17]:

	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 [18]:
```
movies.isnull().sum()
```

Out[18]:
```
Film                        0
Genre                       0
Rotten Tomatoes Ratings %   0
Audience Ratings %          0
Budget (million $)          0
Year of release             0
dtype: int64
```

In [19]:
```
movies.columns
```

Out[19]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

To Clean the Attribute

- List is mutable/changeable

In [21]:
```
movies.columns = ['Film', 'Genre', 'CriticRating', 'AudienceRating', 'BudgetMill

# Rotten Tomatoes Ratings % changed to CriticRating
# Audience Ratings % changed to AudienceRating
# Budget (million $) changed to BudgetMillions
# Year of release changed to year
```

In [22]:
```
movies.head()   #Displays new cleansed attribute, without noisy characters - %,
```

Out[22]:

	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 [23]:
```
movies.head(1)
```

Out[23]:

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

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

Out[24]: (559, 6)

In [25]:
```
movies.describe()        #Descriptive Statistics

# min is 0 of CriticRating, AudienceRating and BudgetMillions as there are 559 r
# If we look at the year, the data type is int. When we look at the mean it show
# take average of years. So, we have to convert Year also to Category.
# which we have to change to category type.
# Also from Object datatype, we will convert to category datatypes
```

Out[25]:

	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 [26]:
```
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
```

We have to convert Film and Genre to category

In [28]:
```
movies.Film = movies.Film.astype('category')
```

In [29]:
```
movies.Genre = movies.Genre.astype('category')
```

In [30]:
```
movies.Film
```

Out[30]:
```
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 [31]:
```
movies.Genre
```

Out[31]:
```
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', 'R
omance', 'Thriller']
```

In [32]:
```
movies.Year = movies.Year.astype('category')
```

In [33]:
```
movies.Year
```

Out[33]:
```
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 [34]:
```
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    category
dtypes: category(3), int64(3)
memory usage: 36.5 KB
```

In [35]:
```
movies.describe()    #Year won't display now as it is not numeric now
```

Out[35]:

	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

Working with Joint Plots

In [37]:
```
from matplotlib import pyplot as plt   #Visualisation
import seaborn as sns                  #Advanced Visualisation

%matplotlib inline

#All the plot should be inside the line

import warnings
warnings.filterwarnings('ignore')
```

- Joint Plot is a plot of two variables with bivariate and univariate graphs.
- Joint plot is a scattered plot. It is used to find the relation between the critics and audience
- If we look up, we can find uniform distribution (critics) and normal distribution (audience)

In [39]:
```
j = plt.jointplot( data = movies, x = 'CriticRating', y = 'AudienceRating')
```

```
# Name of the variables -> x = 'CriticRating', y = 'AudienceRating'
# x = 'CriticRating', y = 'AudienceRating' -> These are called attributes
# We get error -> module 'matplotlib.pyplot' has no attribute 'jointplot'

# Matplotlib is a visualization library, but we want advanced. Thus, we will cha
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[39], line 1
----> 1 j = plt.jointplot( data = movies, x = 'CriticRating', y = 'AudienceRatin
g')

AttributeError: module 'matplotlib.pyplot' has no attribute 'jointplot'
```

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

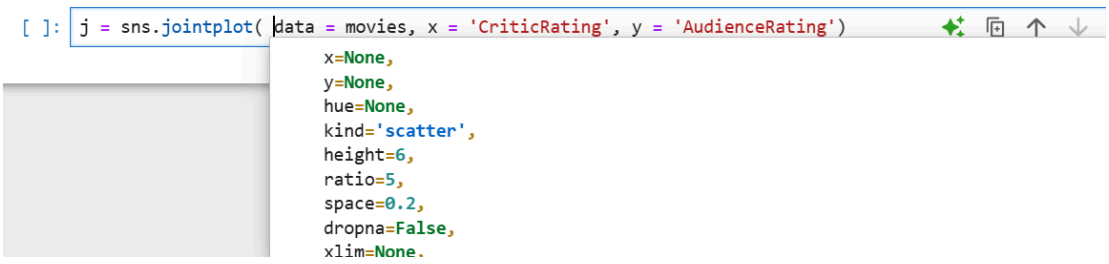
In [ ]:
```
j
plt.show()
```

This is a positive co-relation graph

- There is 1 outlier -> Audience rating is 0 and Critics rating is also 0
- **Interview Ques.** - > How to identify the Outlier?
- **Ans.** Based on graph and visualization
- Ques. How these datapoints are created?
- Ans. It is Bi-variate Analysis. X-axis is Critic rating, Y-axis is audience rating. Datapoints are created from the excel sheet.

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

# Press shift + tab on data and we can see kind = scatter

# kind : kind : { "scatter" | "kde" | "hist" | "hex" | "reg" | "resid" }

#   Kind of plot to draw. See the examples for references to the underlying func
```


- **Ans.** Based on graph and visualization
- Ques. How these datapoints are created?
- Ans. It is Bi-variate Analysis. X-axis is Critic rating, Y-axis is audience rating. Datapoints are created from the exce



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

In [ ]:
```
j
```

```
plt.show()
```

kind = 'hex'presents datapoints in hexagon shape. This is an advanced graph. It is not possible to create this graph in excel and plt(matplotlib), but possible in sns(seaborn).This is the difference between visualization and statistical visualization.

Insights - Explaintation of above graph -

- Based on this we find out that most people are likely to watch movies as per Audience rating rather than critic rating
- Audience rating is more dominant than critics rating
- There is a positive correlation between the 2 attributes

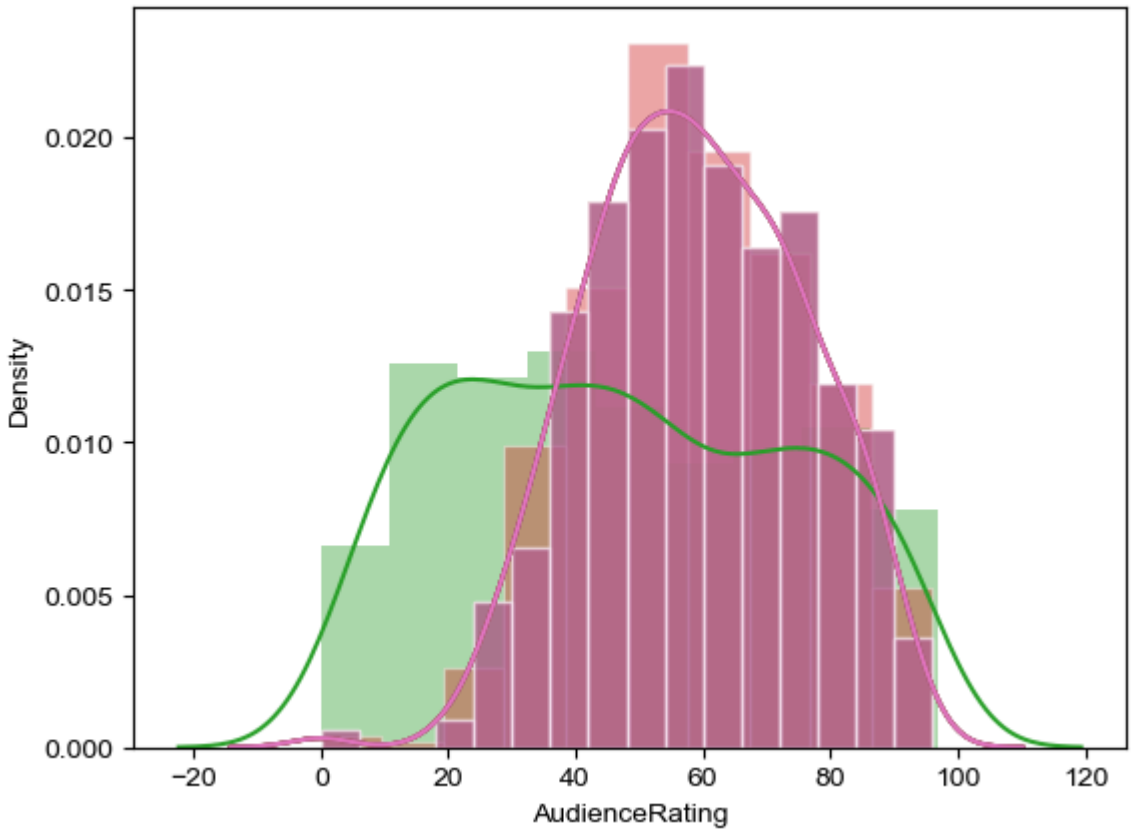
In [ ]:
```
j = sns.jointplot( data = movies, x = 'CriticRating', y = 'AudienceRating', kind
j
plt.show()
```

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

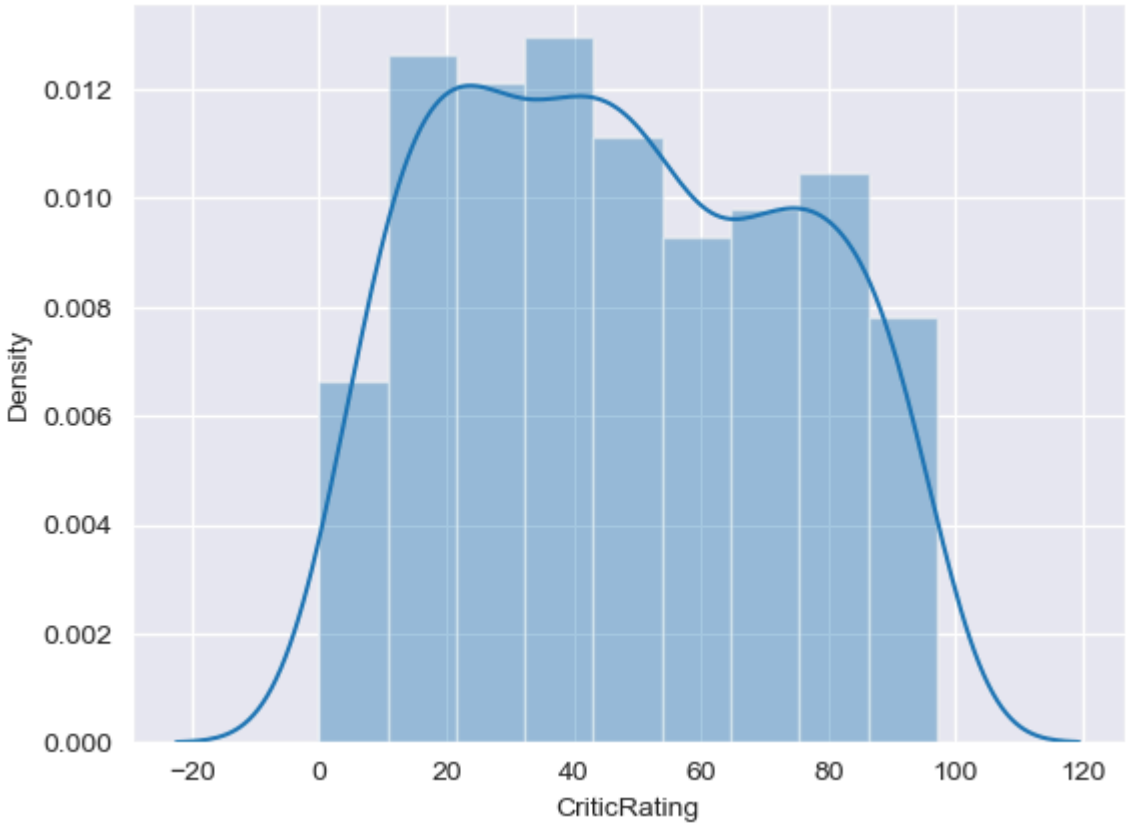
In [ ]:
```
j = sns.jointplot( data = movies, x = 'CriticRating', y = 'AudienceRating', kind
j
plt.show()

# kde is Kernel Density Plot
# No more dots as all dots are connected
```

If someone asks, now we can explain -> This is called Indepth analysis

In [56]:
```
m1 = sns.distplot(movies.AudienceRating) #displot -> distribution plot
plt.show()

# sns -> SeaBorn
# displot -> distribution plot
# Above statement displays Normal Distribution Graph
```

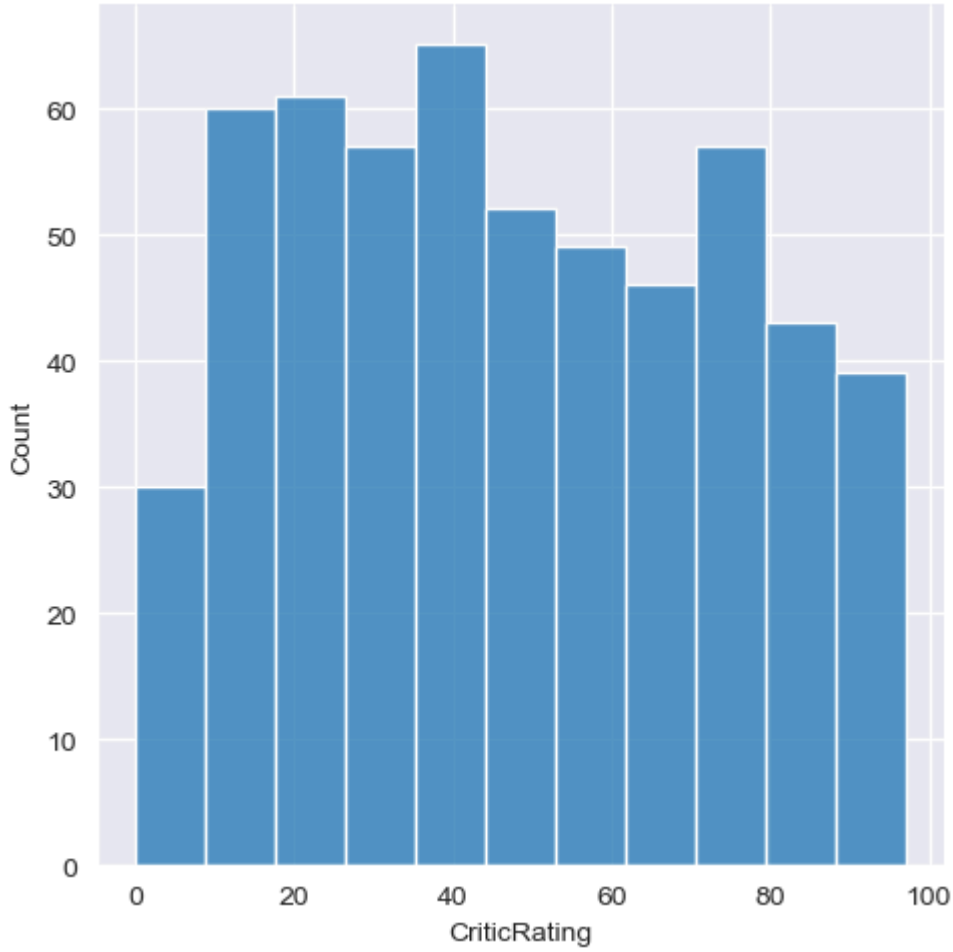


In [72]:
```
m1 = sns.distplot(movies.CriticRating)
plt.show()
```



In distplot, distribtion line is absent

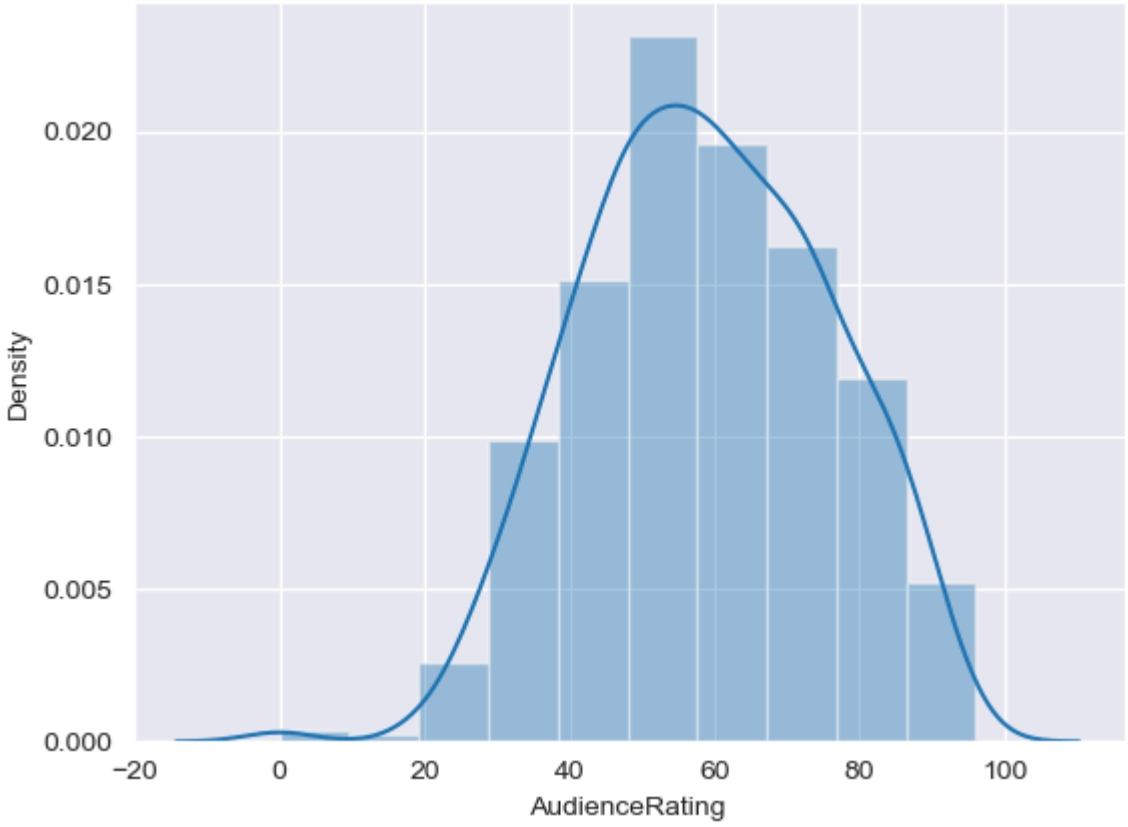
In [60]:
```
m1 = sns.displot(movies.CriticRating)
plt.show()
```



In displot, distribtion line is present

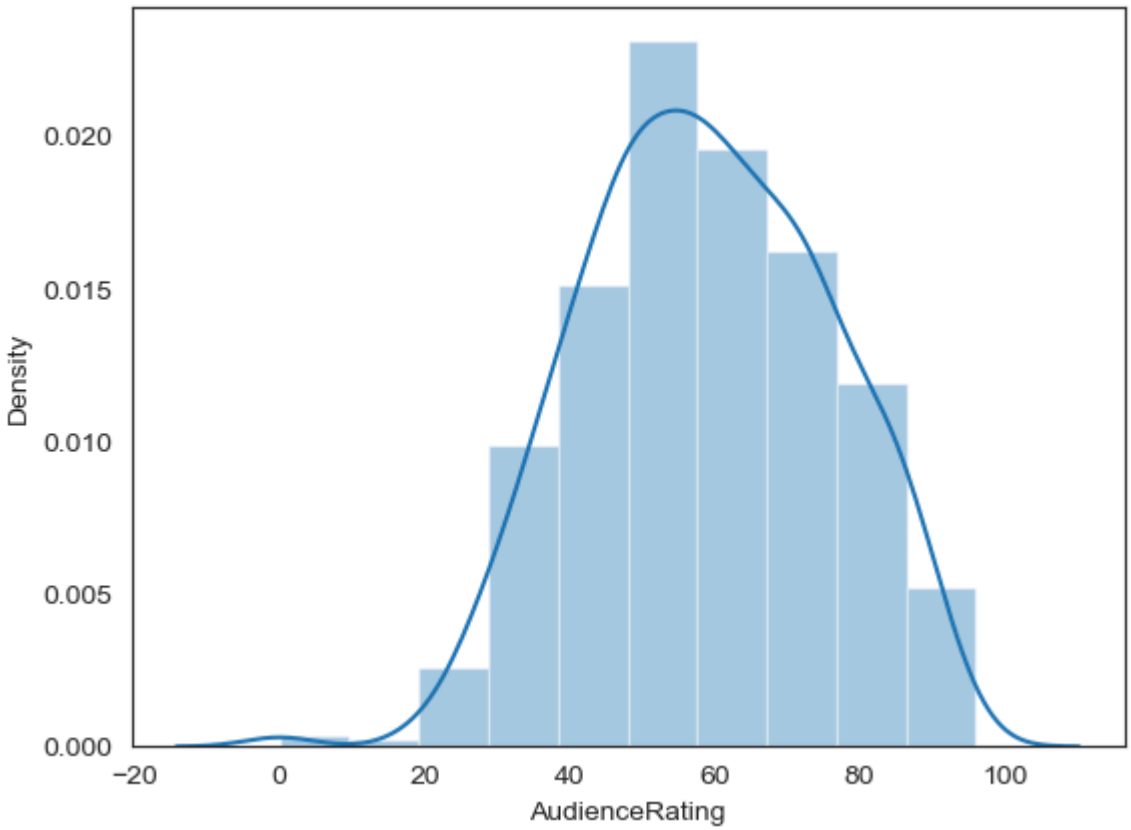
In [66]:
```
sns.set_style('darkgrid')

# style : dict, or one of {darkgrid, whitegrid, dark, white, ticks}
```

In [77]:
```
m2 = sns.distplot(movies.AudienceRating, bins = 10)
plt.show()

# 10 bins means 10 bars
# Background grid is grey
```

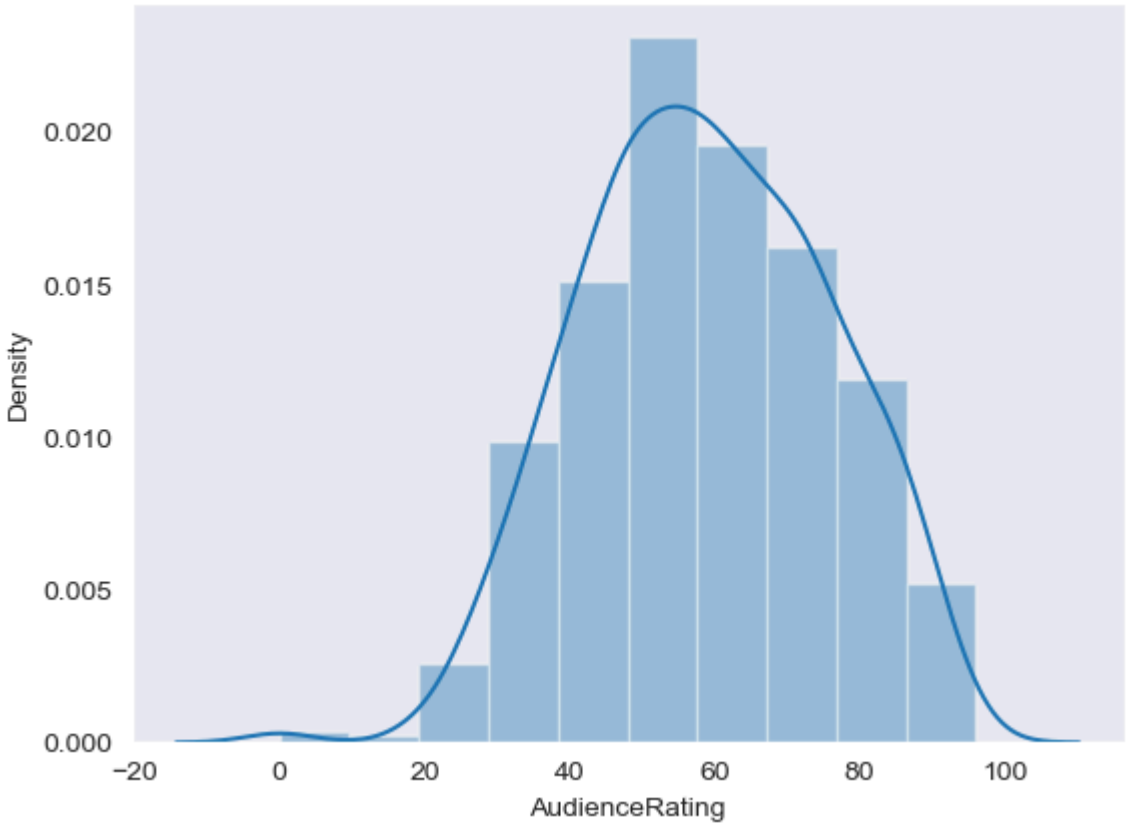


Now the background is dark

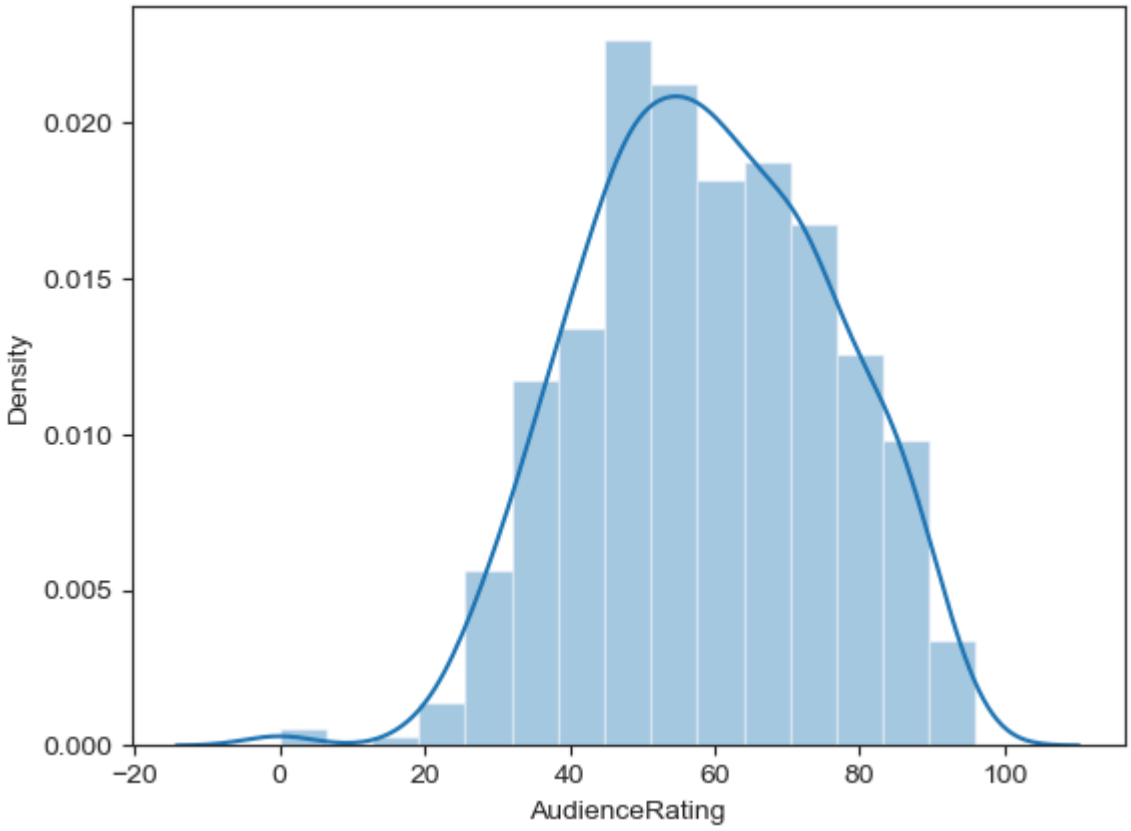
In [81]:
```
sns.set_style('white')
m2 = sns.distplot(movies.AudienceRating, bins = 10)
plt.show()

#Again the background will change to white
```



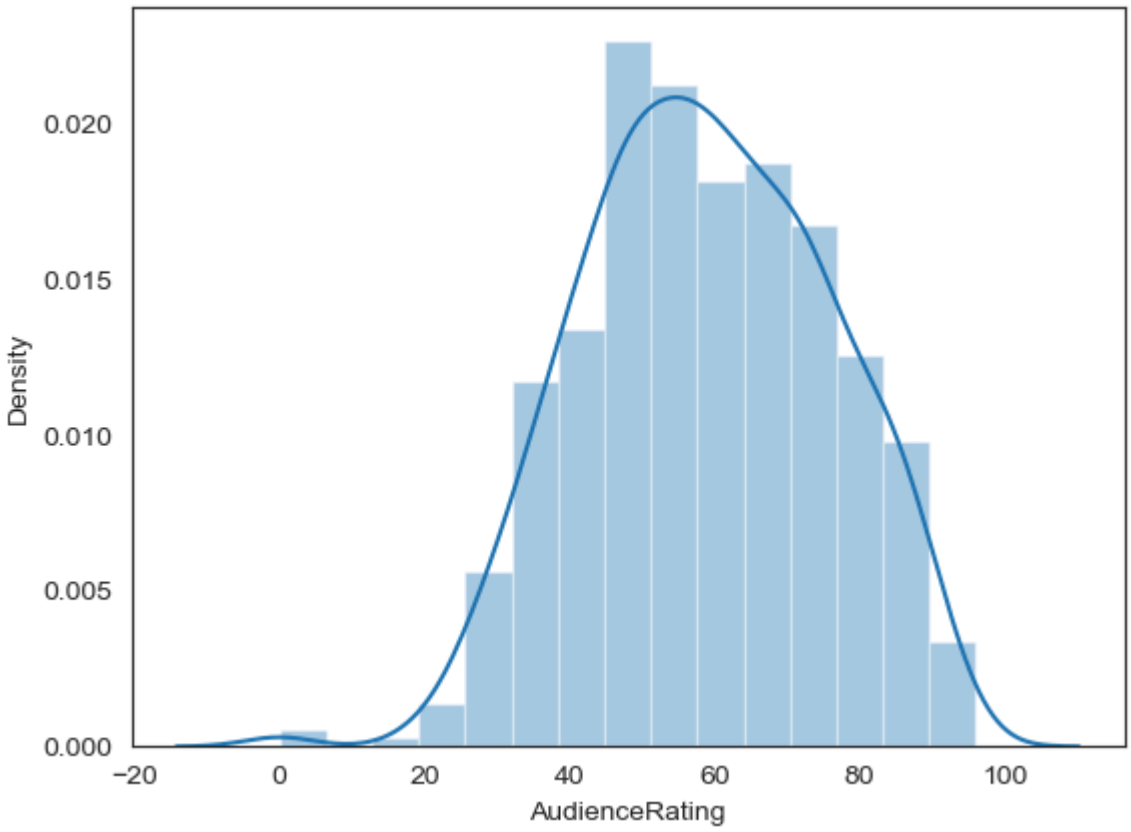
In [85]:
```
sns.set_style('dark')
m2 = sns.distplot(movies.AudienceRating, bins = 10)
plt.show()

#Now the background will change to complete dark
#we can't use 'black' as .set_style understands only 4 parameters - {darkgrid, w
```



In [89]:
```
sns.set_style('ticks')
m2 = sns.distplot(movies.AudienceRating, bins = 15)
plt.show()

#It is same as white only
```

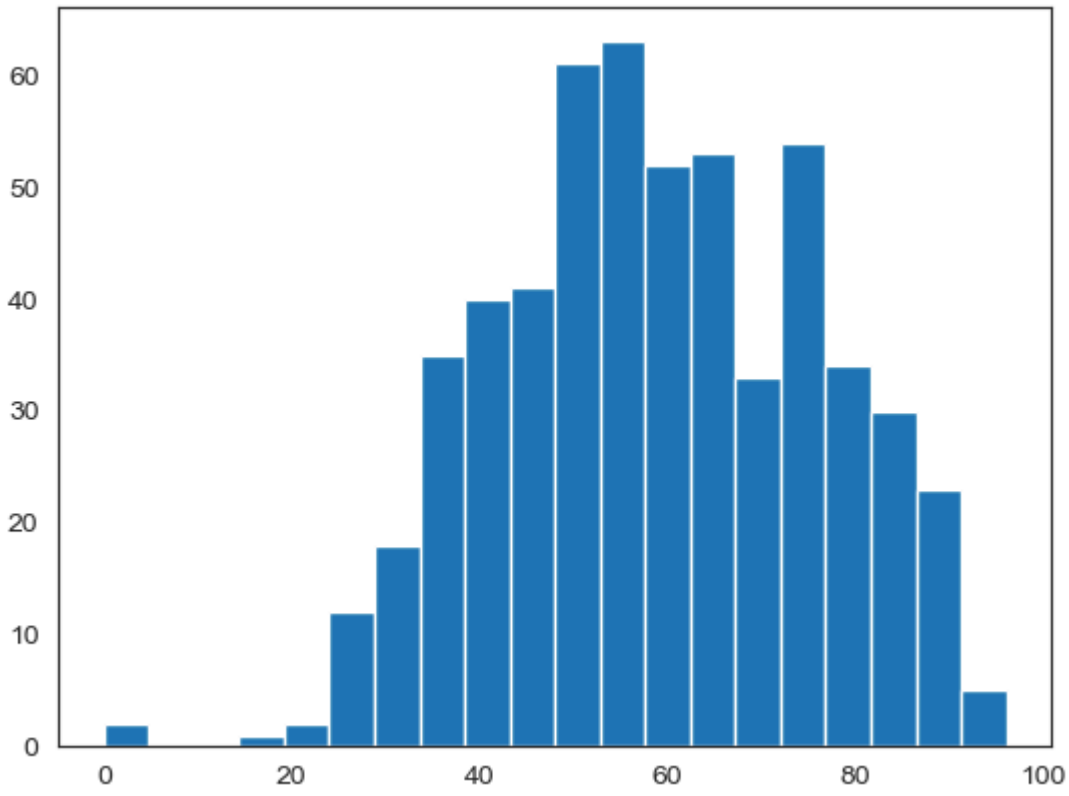


In [91]:
```
sns.set_style('white')
m2 = sns.distplot(movies.AudienceRating, bins = 15)
plt.show()
```



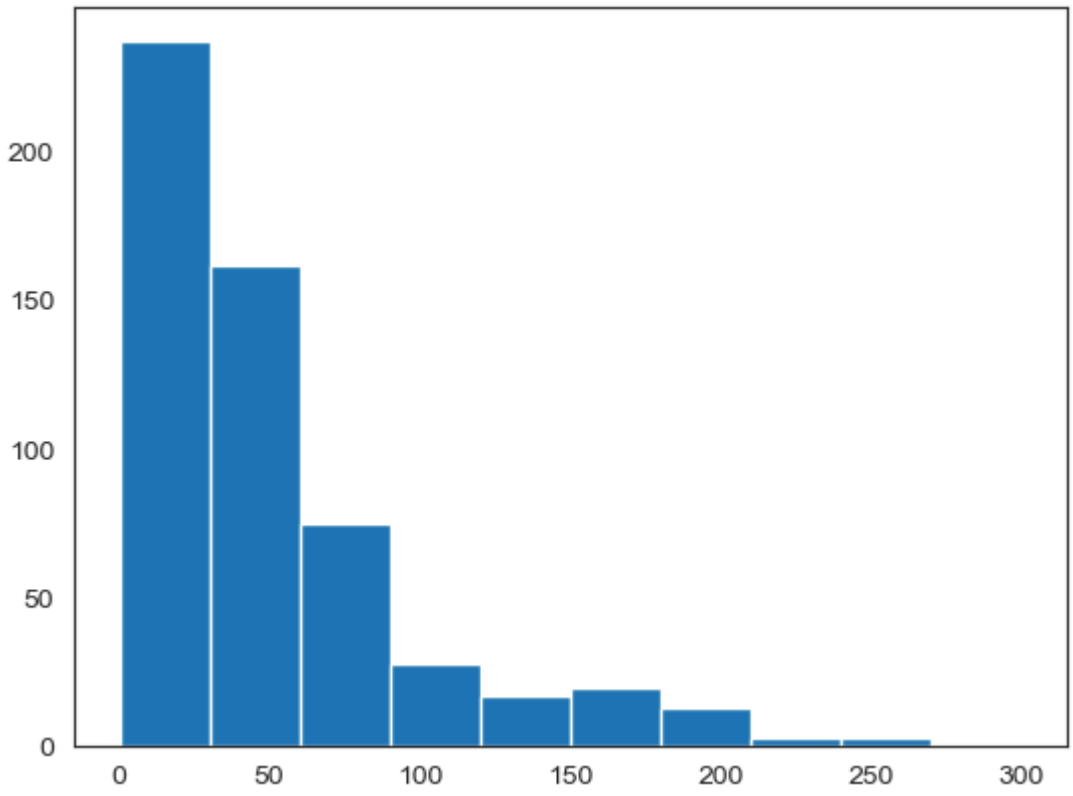
In [99]:
```
n1 = plt.hist(movies.AudienceRating, bins = 20)
plt.show()
```

```
#This creates 20 bins
```



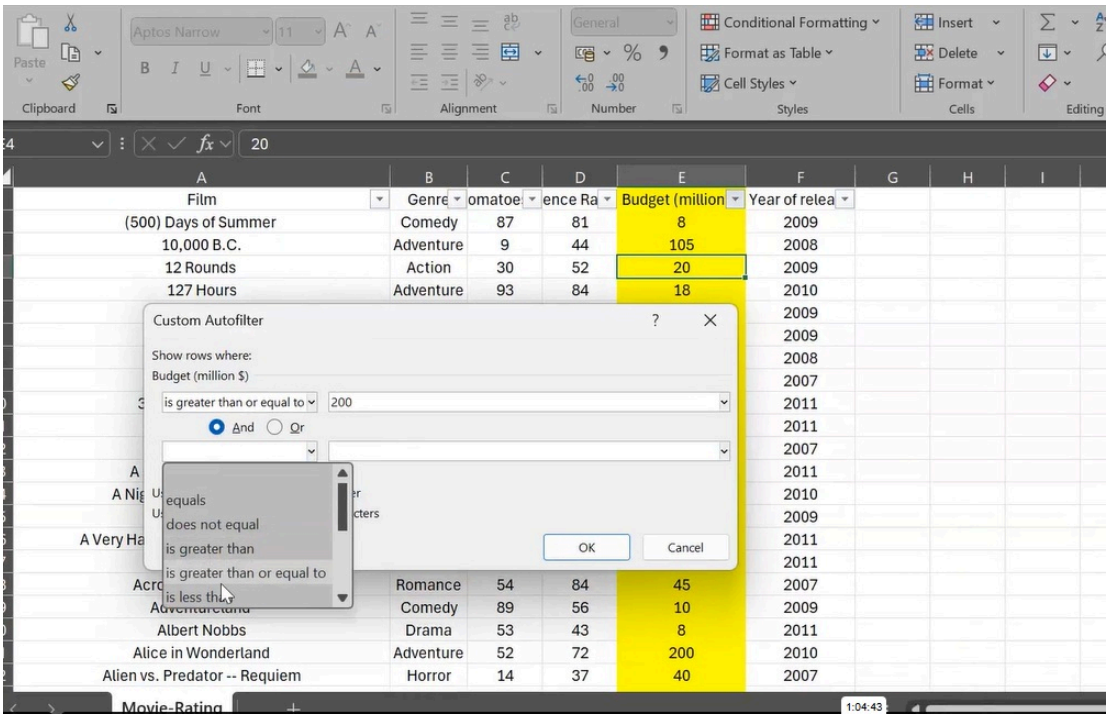
In [101…

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

#plt.hist means histogram
```



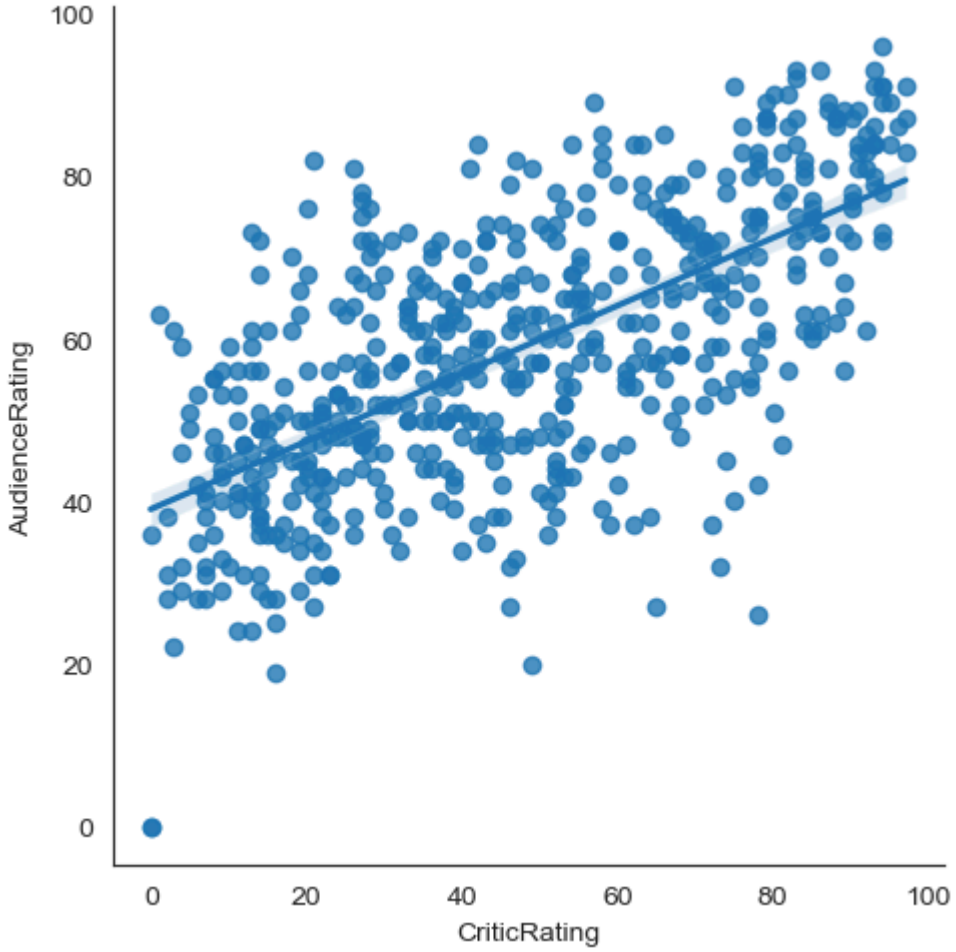
- Lowest budget movies is between 200-300, we know this from Graph.

- The excel sheet is visualised here.
- As we have less movies from 200-300 thus bar size is also less



In [133…

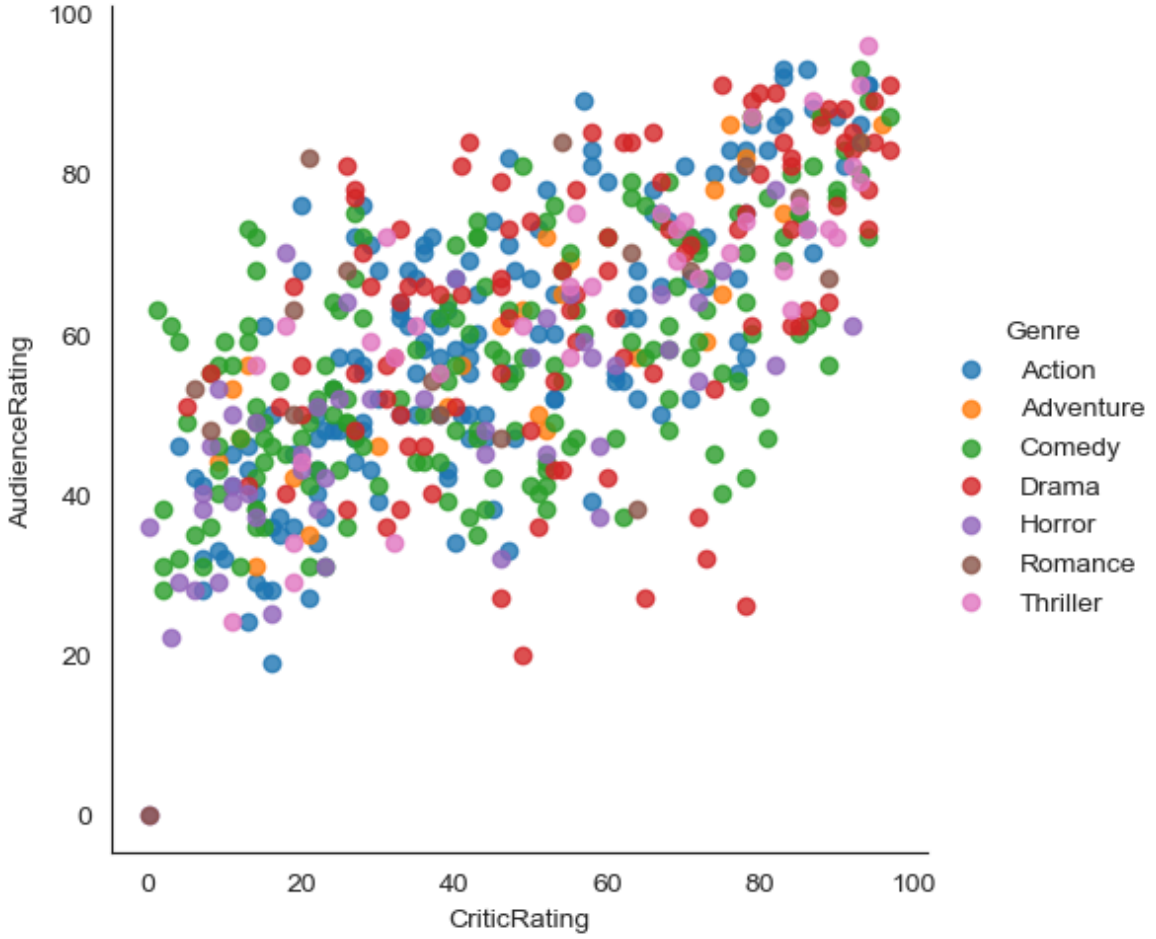
```
vis1 = sns.lmplot( data = movies, x = 'CriticRating', y = 'AudienceRating', fit_
plt.show()

#.lmplot -> Linear model plot
# fit_reg = 'True' means we will get the centre line.
# fit_reg = 'False' means we will NOT get the centre line.
```



As all datapoints are of same color thus unable to answer, which genre movies are outlier

In [131…

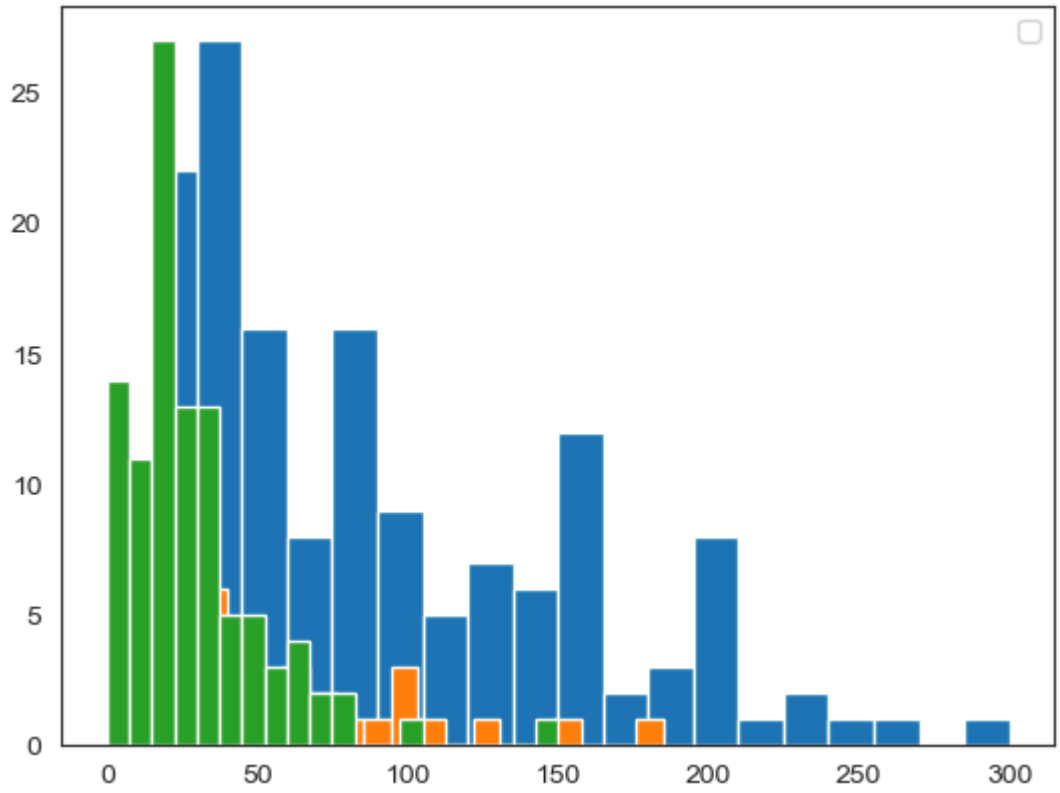
```
vis1 = sns.lmplot( data = movies, x = 'CriticRating', y = 'AudienceRating', fit_
plt.show()

#hue = 'Genre' -> displays colorful datapoints
#hue is pandas is same as legend in Numpy
```



As all datapoints are of different color thus we can say that romance genre has an outlier as audience rating is 0 and Critic Rating is also 0

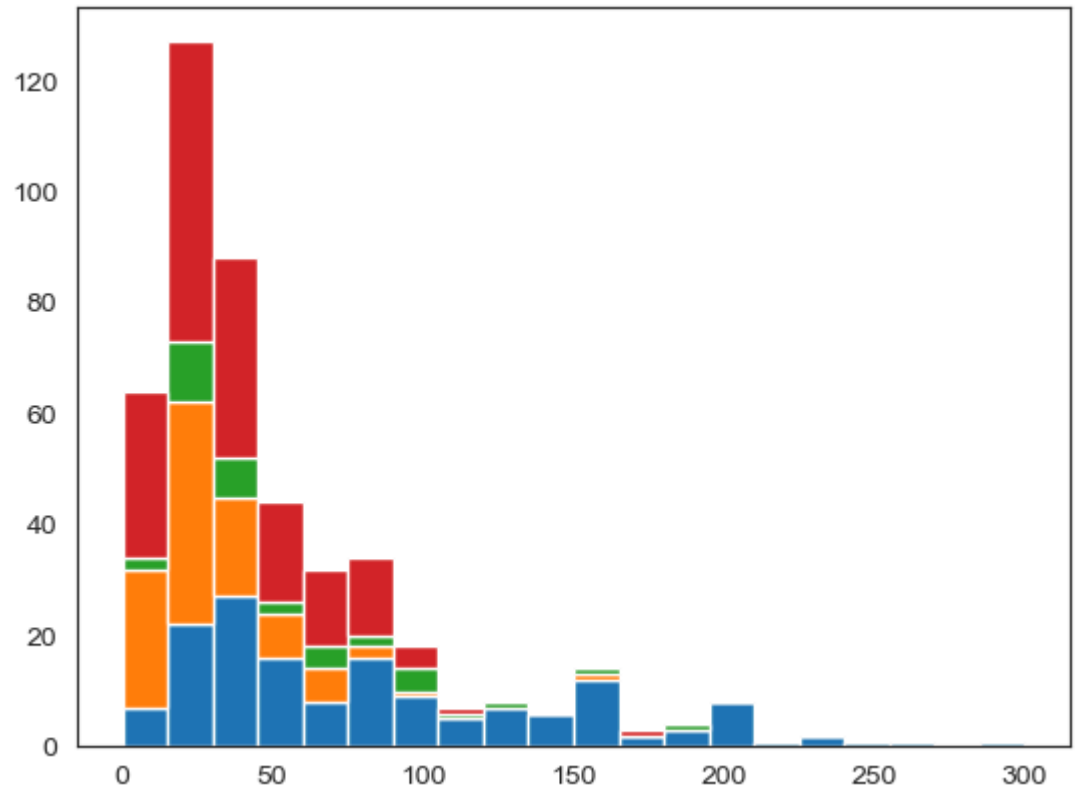
In [119…

```
# Below plots are stacked histogram because they are 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 handles with labels found to put in legend.
```



In [123…

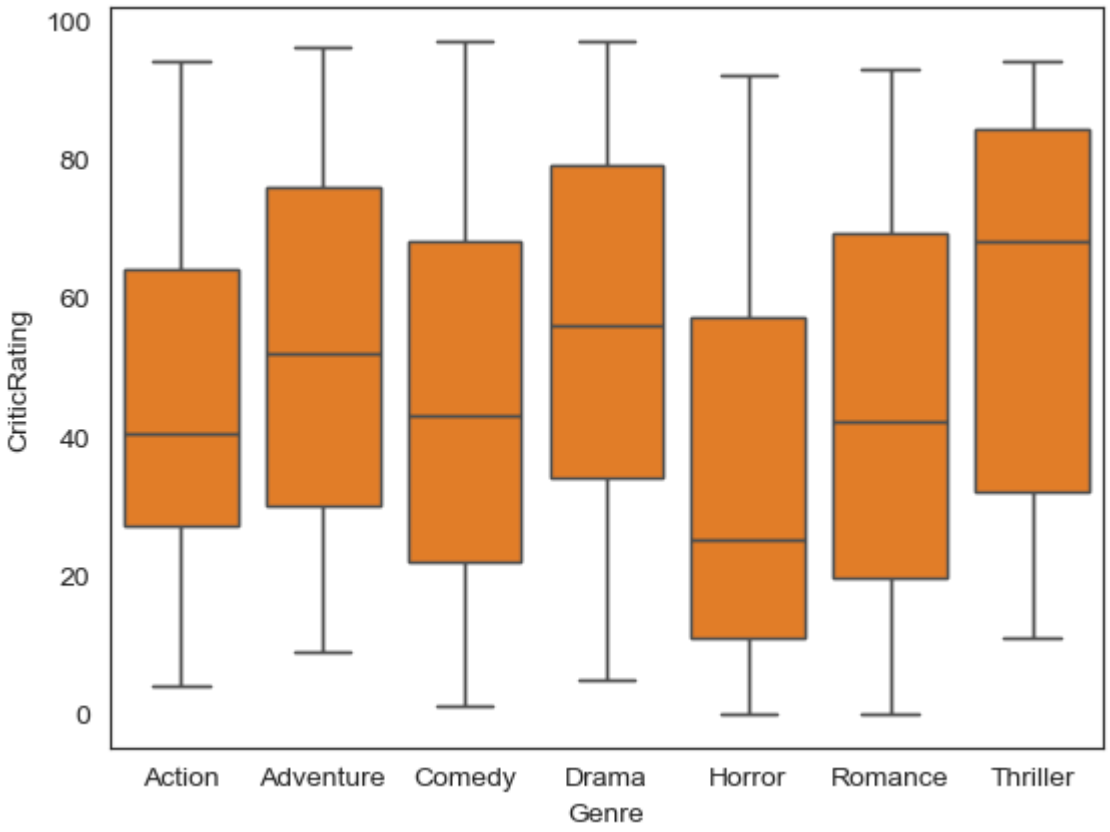
```
plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,
          movies[movies.Genre == 'Drama'].BudgetMillions,
          movies[movies.Genre == 'Thriller'].BudgetMillions,
          movies[movies.Genre == 'Comedy'].BudgetMillions],
         bins = 20, stacked = True)
plt.show()
```



In [143…

```
#Box plots -
```

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



Ques. Which Genre has highest average critic rating?

Ans. Thriller

Ques. Which Genre has lowest average critic rating?

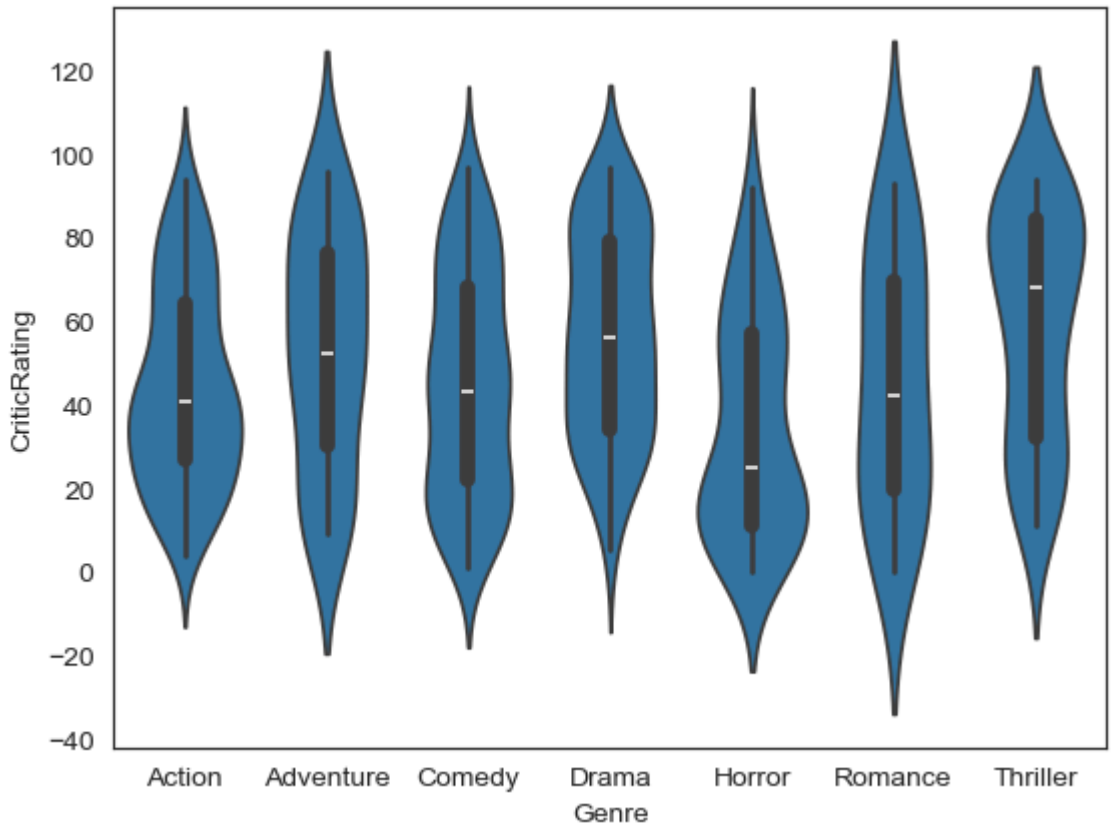
Ans. Horrer

Ques. Which Genre has lowest critic rating?

Ans. Horrer and Romance

In [147…

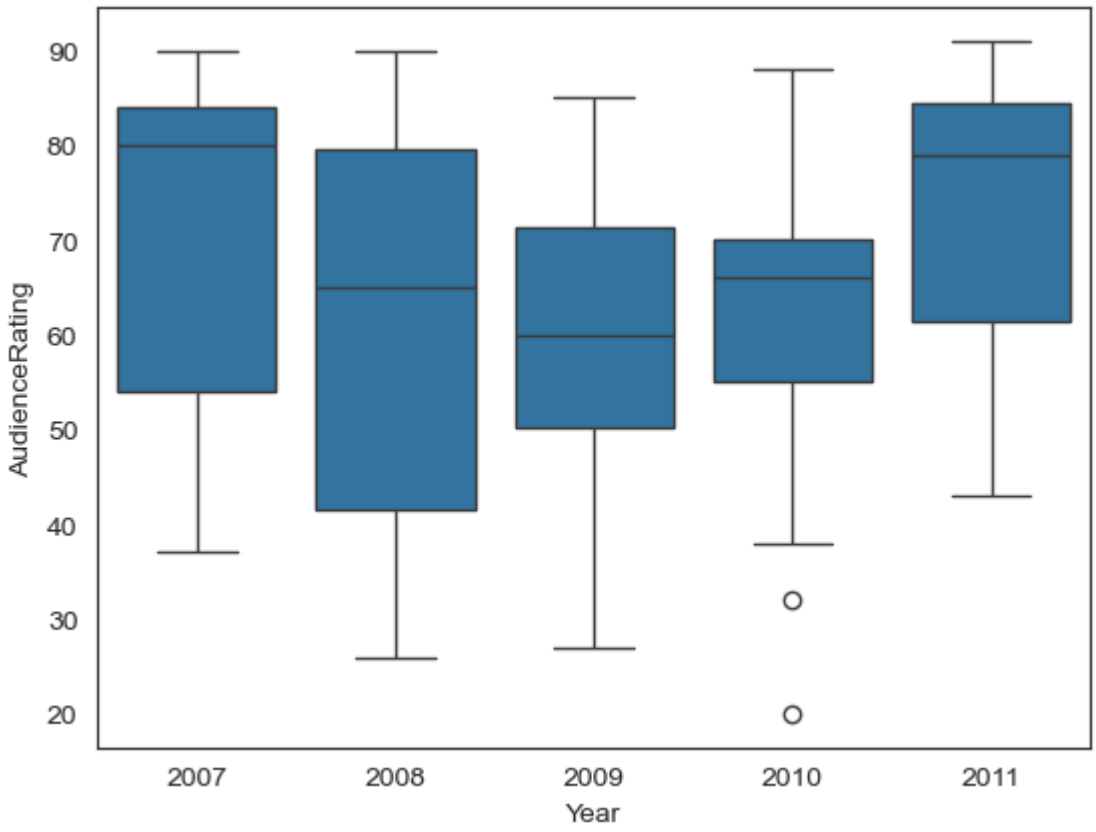
```
#Violin plots -

w = sns.violinplot(data=movies, x='Genre', y = 'CriticRating')
plt.show()

#Violin plot works in Seaborn only, not in Plt
```



- In the violin graph, we have to focus on average points more i.e white dots
- In the box plot, focus is on min and max

In [156…

```
w1 = sns.boxplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'AudienceRa
plt.show()

#To get information of only Drama Category movie
```



- We get the historical data from 2007 to 2011
- In 2007, drama movies were watched very much
- In 2008, people reduced watching it
- In 2009, it reduced further
- Thus, there is a downfall.
- After 2009, client changed something in coming movies.
- If explained this to client, he will be very happy as this is the right time to improve the business.

Facet Grid

- Plotting multiple graphs in 1 sheet is called facet grid

We have 7 category of movies and 5 years

In [167…
```
g = sns.FacetGrid(movies, row = 'Genre', col = 'Year', hue = 'Genre')
```

In [169…
```
plt.show()
```

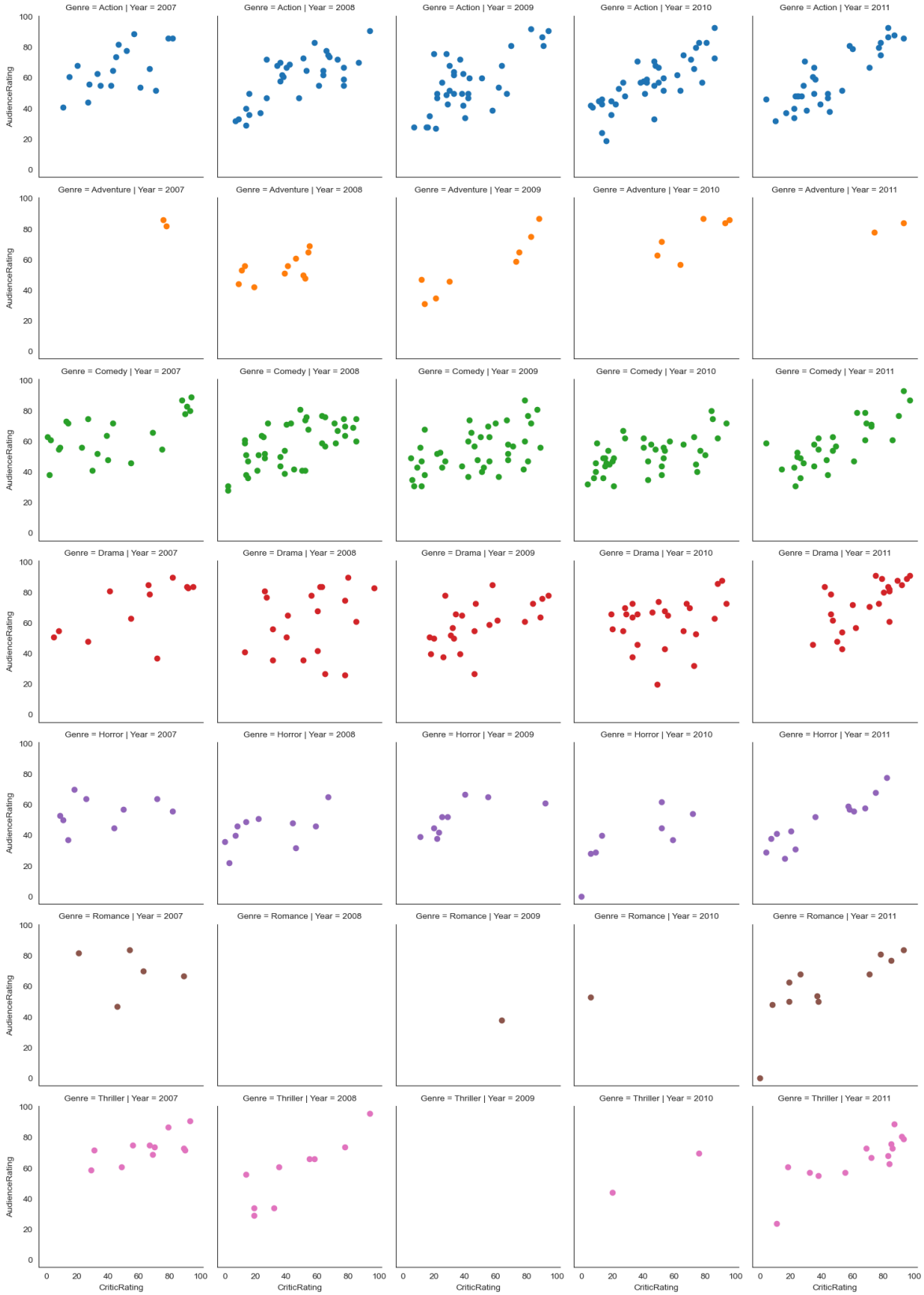
The above graphs are empty, we need to fill the data.

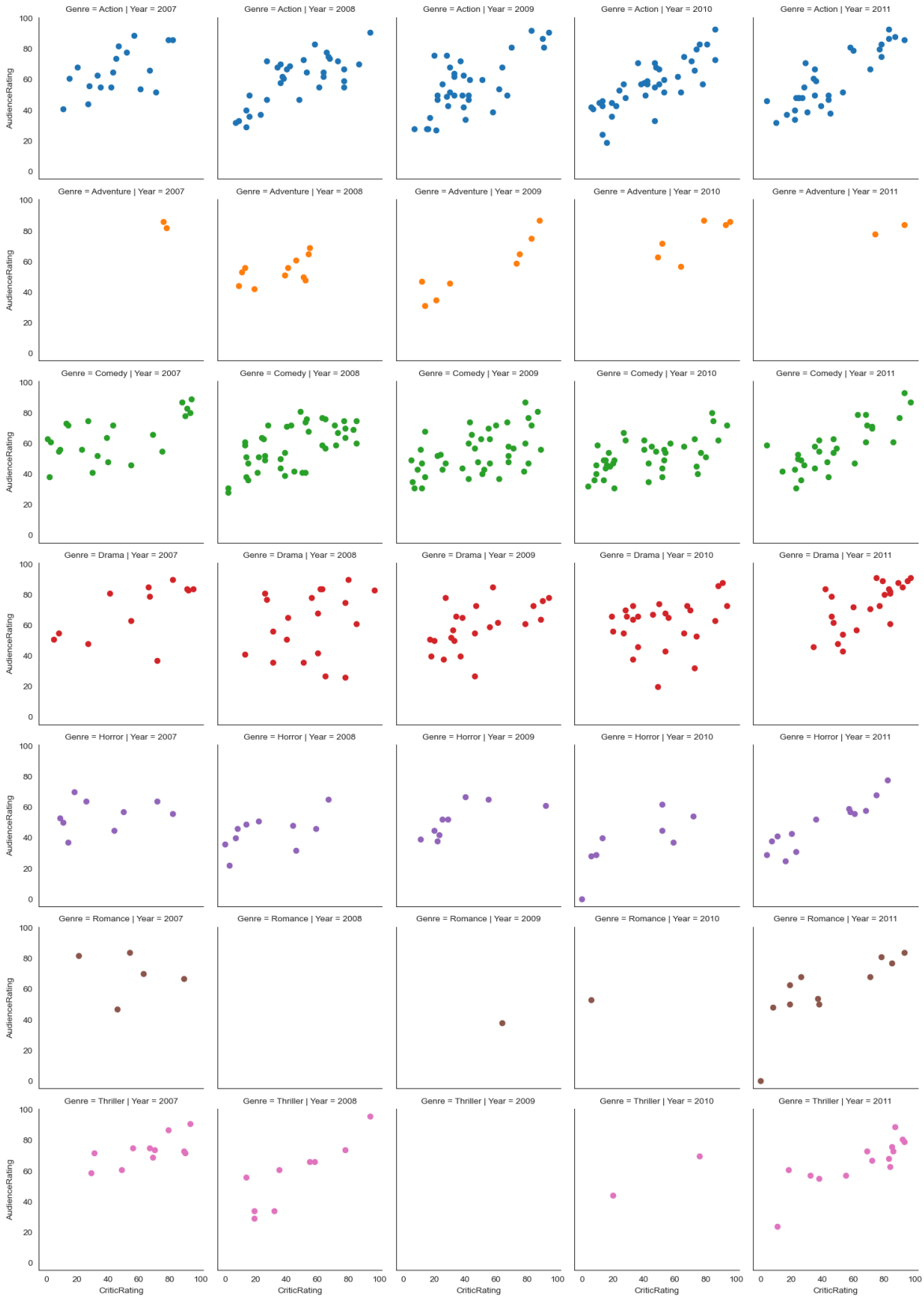
- To fill the line no - 133 graph in these boxes.
- This is not possible in Java and .Net
- SeaBorn has Inbuild datastructure and thus, it is very powerful.
- So, there is a concept called Map

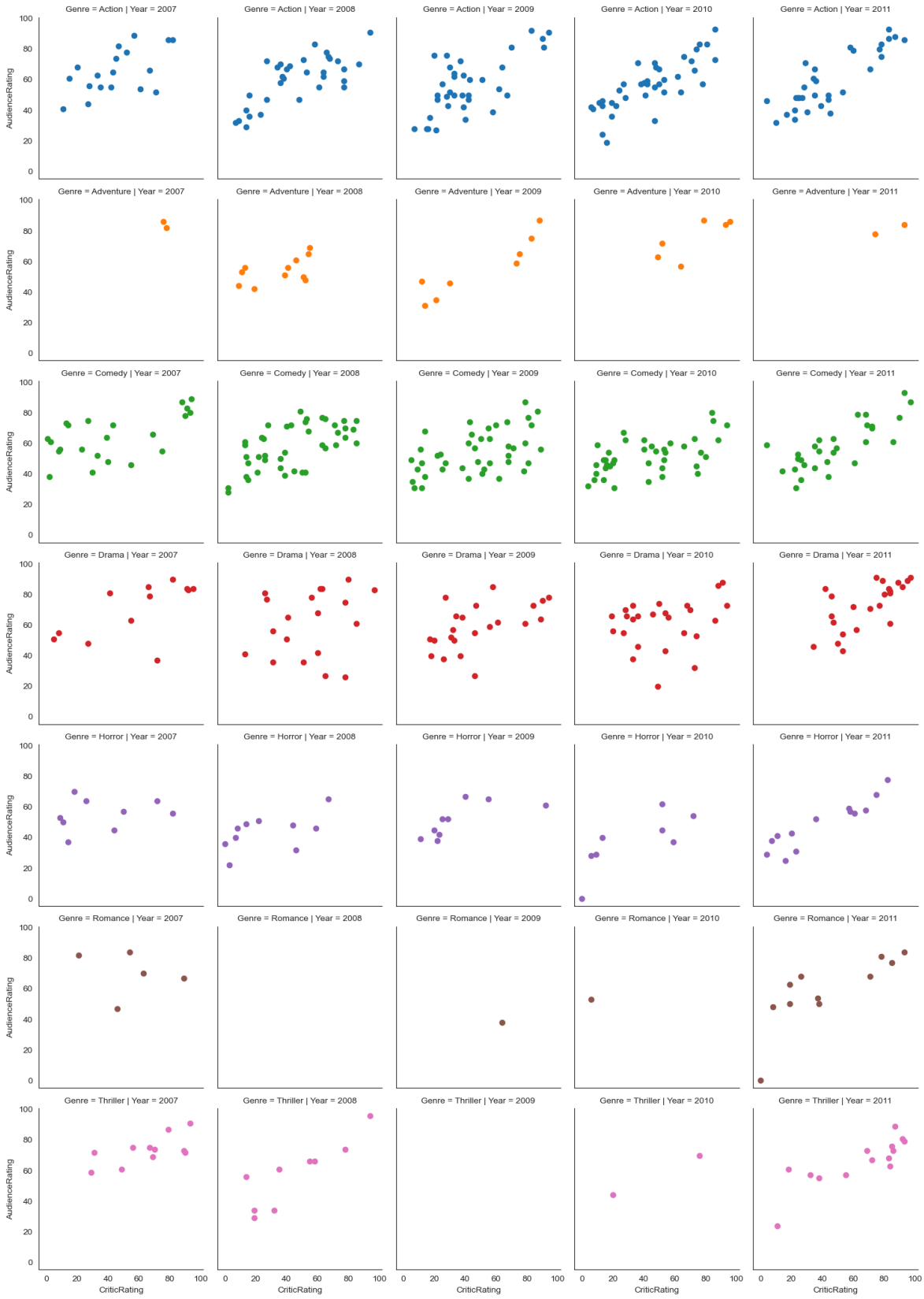
In [175…

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

```
plt.show()

#Scatterplots are mapped in Facedgrid
```




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
AudienceRating
CriticRating
0
20
40
60
80
100



Insights -

- In 2007 most audience watched Action category movies
- Adventure movie is watched by very less people, as it is sparsely polulated(less dense). Thus, I won't recommend producing Adventure movie
- Comedy movie - Historicaly clients have watched it, thus we can suggest
- Drama movie - Historicaly clients have watched it, thus we can suggest
- Horror movie - Historicaly clients have watched it, thus we can suggest

- Romance movie - Historicaly very less people have watched it, thus we won't recommend
- Thriller movie - Historicaly very less people have watched it, thus we won't recommend

Wherever there are highest datapoints, we will suggest that movie to the client

In [ ]: