ating-analysis-using-seaborn-rohan

August 22, 2024

#**Project 09: Movie Rating Analysis (Seaborn)** Author: Rohan Waghmare || Date: 22/08/2024

ADVANCED VISUALIZATION MOVIE RATINGS

###Importing essential libraries

[1]:
```
import pandas as pd  # Import the pandas library and alias it as 'pd' for easy␣
↪reference

import os  # Import the os module, which provides functions for interacting␣
↪with the operating system
```

[2]:
```
os.getcwd()  # Get the current working directory and return it as a string
```

[2]:
```
'/content'
```

###Connect Google Colab to import CSV file

[3]:
```
from google.colab import files
uploaded = files.upload()  # Opens a dialog to upload files
```

```
<IPython.core.display.HTML object>
```

```
Saving Movie-Rating.csv to Movie-Rating.csv
```

[14]:
```
import pandas as pd
rating = pd.read_csv("Movie-Rating.csv")
rating
```

[14]:

	Film	Genre	Rotten Tomatoes Ratings %	\
0	(500) Days of Summer	Comedy	87	
1	10,000 B.C.	Adventure	9	
2	12 Rounds	Action	30	
3	127 Hours	Adventure	93	
4	17 Again	Comedy	55	
..	…	…	…	
554	Your Highness	Comedy	26	
555	Youth in Revolt	Comedy	68	
556	Zodiac	Thriller	89	

```
557            Zombieland      Action                          90
558              Zookeeper     Comedy                          14

     Audience Ratings %  Budget (million $)  Year of release
0                    81                   8             2009
1                    44                 105             2008
2                    52                  20             2009
3                    84                  18             2010
4                    70                  20             2009
..                  …                   …               …
554                  36                  50             2011
555                  52                  18             2009
556                  73                  65             2007
557                  87                  24             2009
558                  42                  80             2011

[559 rows x 6 columns]
```

[15]:
```
len(rating)
```

[15]:
```
559
```

[16]:
```
rating.head()
```

[16]:
```
                   Film      Genre  Rotten Tomatoes Ratings %  \
0  (500) Days of Summer      Comedy                         87
1            10,000 B.C.  Adventure                          9
2             12 Rounds      Action                         30
3             127 Hours  Adventure                         93
4             17 Again       Comedy                         55

   Audience Ratings %  Budget (million $)  Year of release
0                  81                   8             2009
1                  44                 105             2008
2                  52                  20             2009
3                  84                  18             2010
4                  70                  20             2009
```

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

[17]:
```
                Film     Genre  Rotten Tomatoes Ratings %  Audience Ratings %  \
554    Your Highness    Comedy                         26                  36
555  Youth in Revolt    Comedy                         68                  52
556           Zodiac  Thriller                         89                  73
557      Zombieland     Action                         90                  87
558        Zookeeper    Comedy                         14                  42
```

```
     Budget (million $)  Year of release
554                  50              2011
555                  18              2009
556                  65              2007
557                  24              2009
558                  80              2011
```

[19]:
```
type(rating)
```

[19]:
```
pandas.core.frame.DataFrame
```

###Checking the versions of numpy & pandas

[21]:
```
import numpy
import pandas
print(numpy.__version__)
print(pandas.__version__)
```

```
1.26.4
2.1.4
```

###Renaming Attributes & removing noise characters

[18]:
```
rating.columns
```

[18]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

[67]:
```
rating.columns = ['Film', 'Genre', 'CriticRating',␣
 ↪'AudienceRating','BudgetMillions','Year']
```

[68]:
```
rating.head() # removed spaces & % removed noise characters
```

[68]:
```
                    Film      Genre  CriticRating  AudienceRating  \
0  (500) Days of Summer      Comedy            87              81
1            10,000 B.C.  Adventure             9              44
2             12 Rounds      Action            30              52
3             127 Hours  Adventure            93              84
4             17 Again      Comedy            55              70

   BudgetMillions  Year
0               8  2009
1             105  2008
2              20  2009
3              18  2010
4              20  2009
```

0.0.1 After describe() function print columns which contain numerical values

- 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

[69]:
```
rating.describe()
# Provides summary statistics of the 'rating' column in a DataFrame
```

[69]:

	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

[70]:
```
rating.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
```

###Now I'm going to change the Dtype of the Attributes into the 'category' of: - "Film", - "Genre", - "Year"

[71]:
```
rating.Film = rating.Film.astype('category')
```

[35]:
```
rating.Film
```

[35]:
```
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']
```

[72]:
```
rating.Genre = rating.Genre.astype('category')
```

[36]:
```
rating.Genre
```

[36]:
```
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',
'Romance', 'Thriller']
```

[73]:
```
rating.Year = rating.Year.astype('category')
```

[37]:
```
rating.Year
```

[37]:
```
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]
```

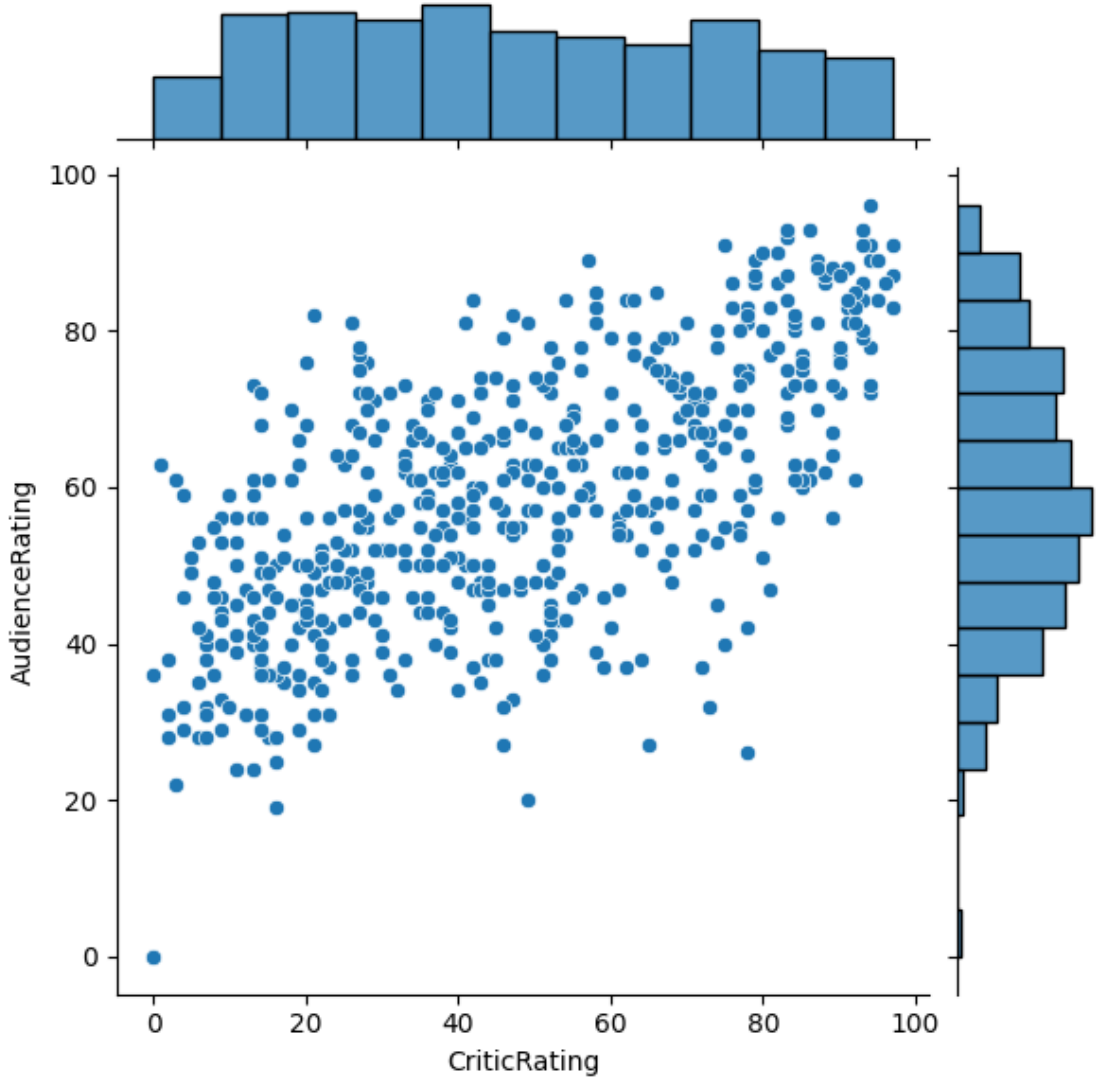
[74]:
```
rating.info() # successfully converted into category
```

```
<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
```

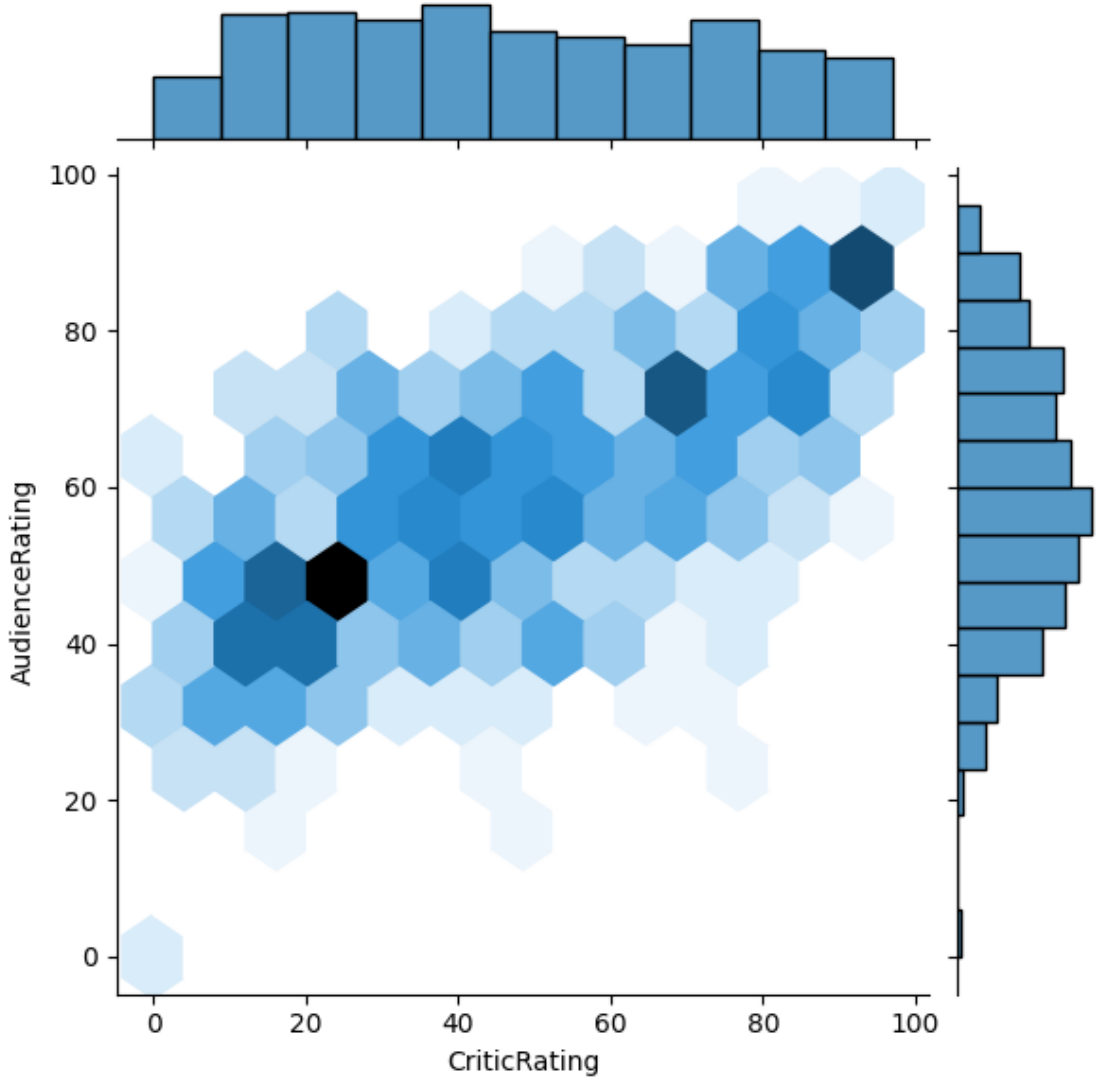
[75]:
```
from matplotlib import pyplot as plt  # Import the pyplot module from␣
 ↪Matplotlib for creating visualizations
import seaborn as sns  # Import the Seaborn library for enhanced data␣
 ↪visualization based on Matplotlib
#%matplotlib inline  # Display plots inline in Jupyter Notebook (only needed if␣
 ↪using Jupyter)
import warnings  # Import the warnings module to manage warning messages
warnings.filterwarnings('ignore')  # Suppress warnings to avoid cluttering the␣
 ↪output
```

- 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)

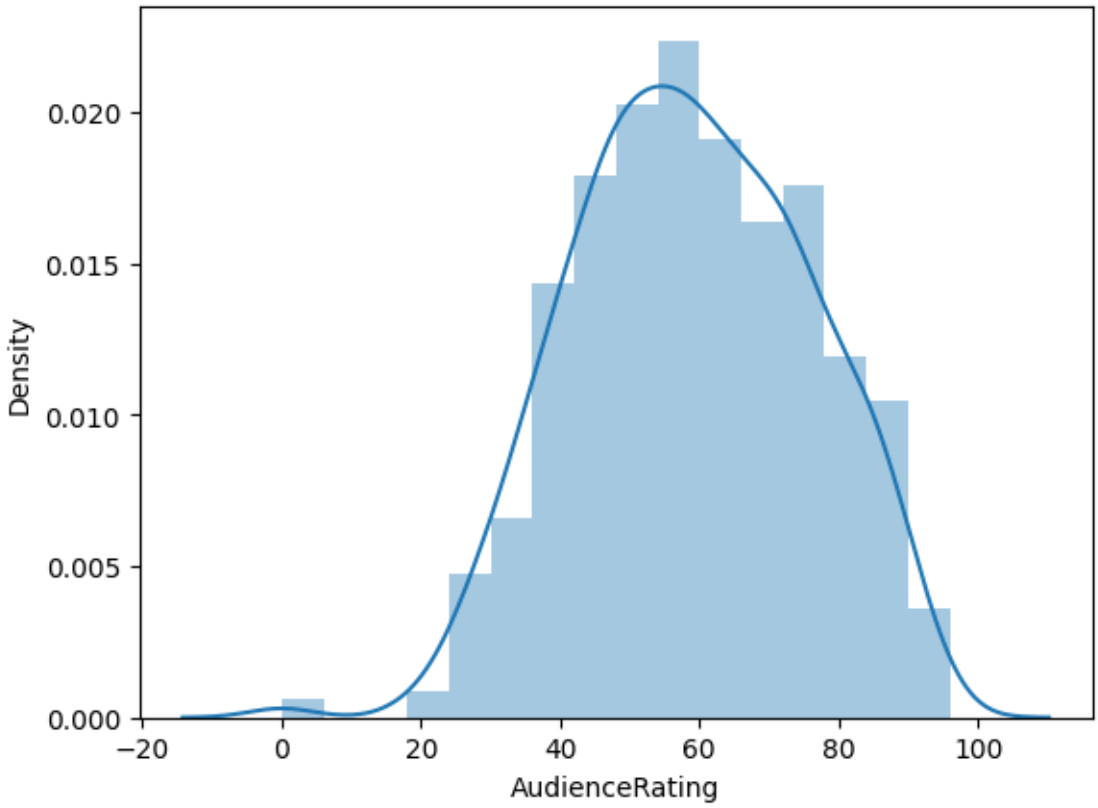
[77]:
```
# Create a joint plot to visualize the relationship between Critic Rating and␣
 ↪Audience Rating from the movies dataset
j = sns.jointplot(data=rating, x='CriticRating', y='AudienceRating')
```



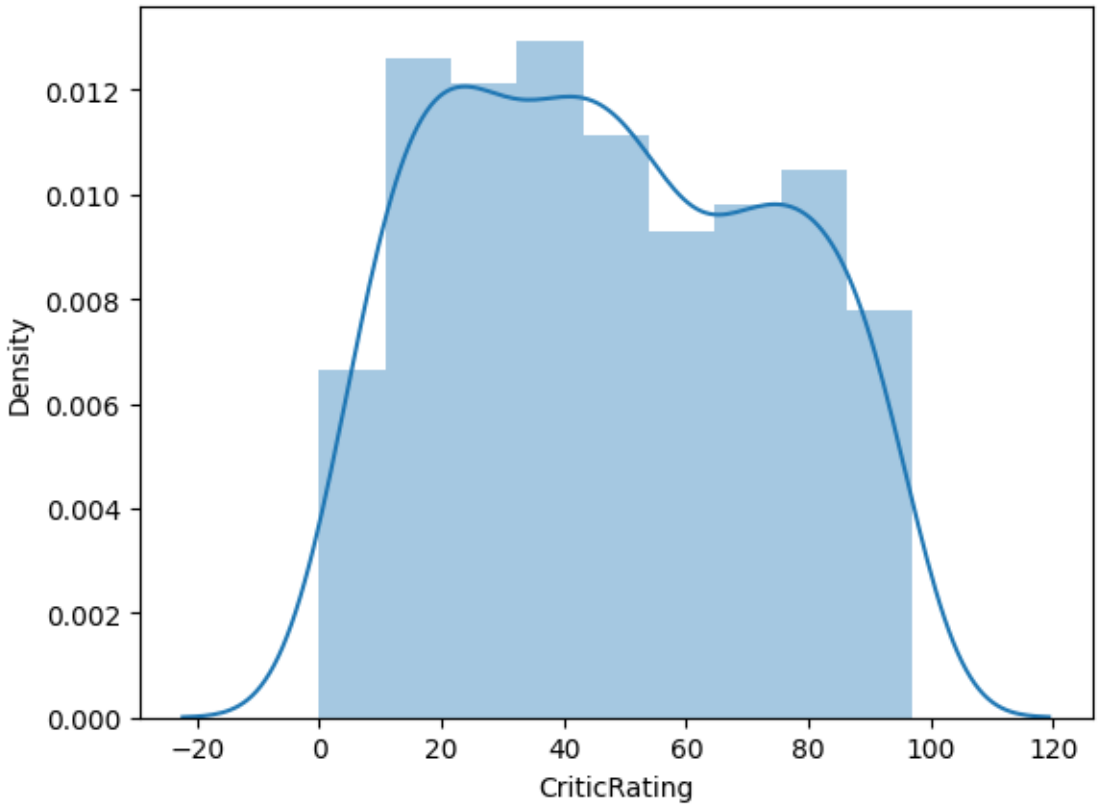
[78]:
```
# Create a hexbin joint plot to visualize the relationship between Critic␣
↪Rating and Audience Rating from the movie dataset
j = sns.jointplot(data=rating, x='CriticRating', y='AudienceRating', kind='hex')
```



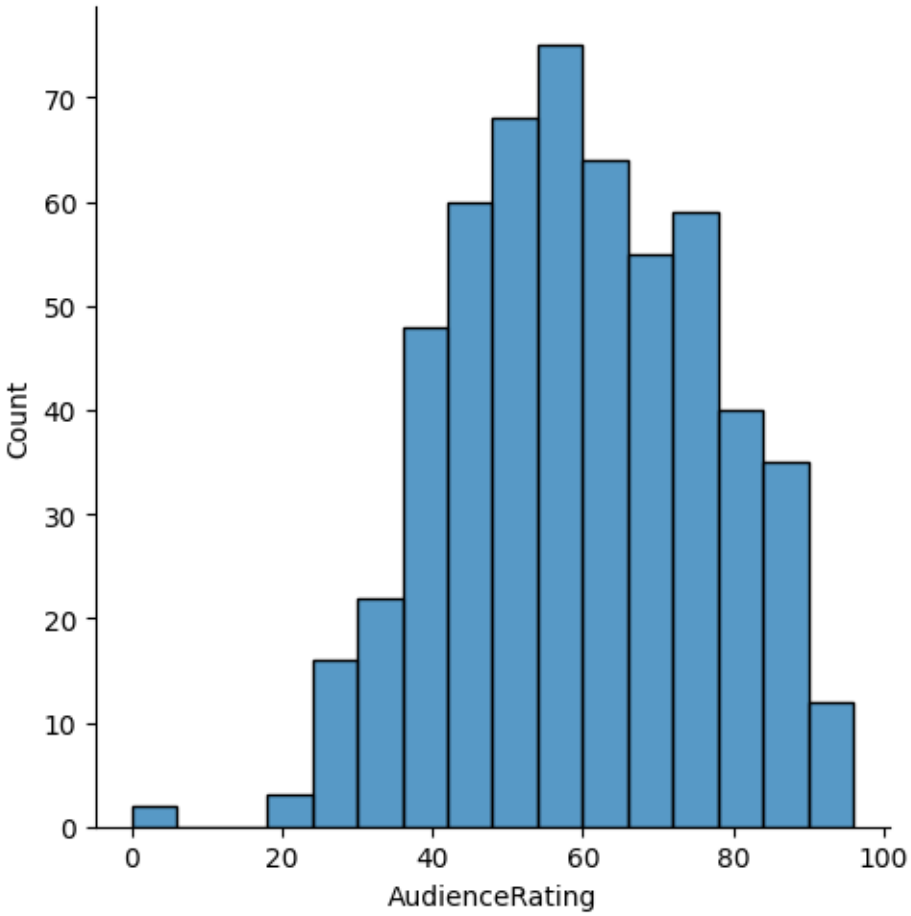
[79]:
```
# Create a distribution plot to visualize the distribution of AudienceRating␣
↪from the movies dataset
audi = sns.distplot(rating.AudienceRating)
```



[80]:
```
# Create a distribution plot to visualize the distribution of Critic Rating␣
↪from the movies dataset
audi = sns.distplot(rating.CriticRating)
```

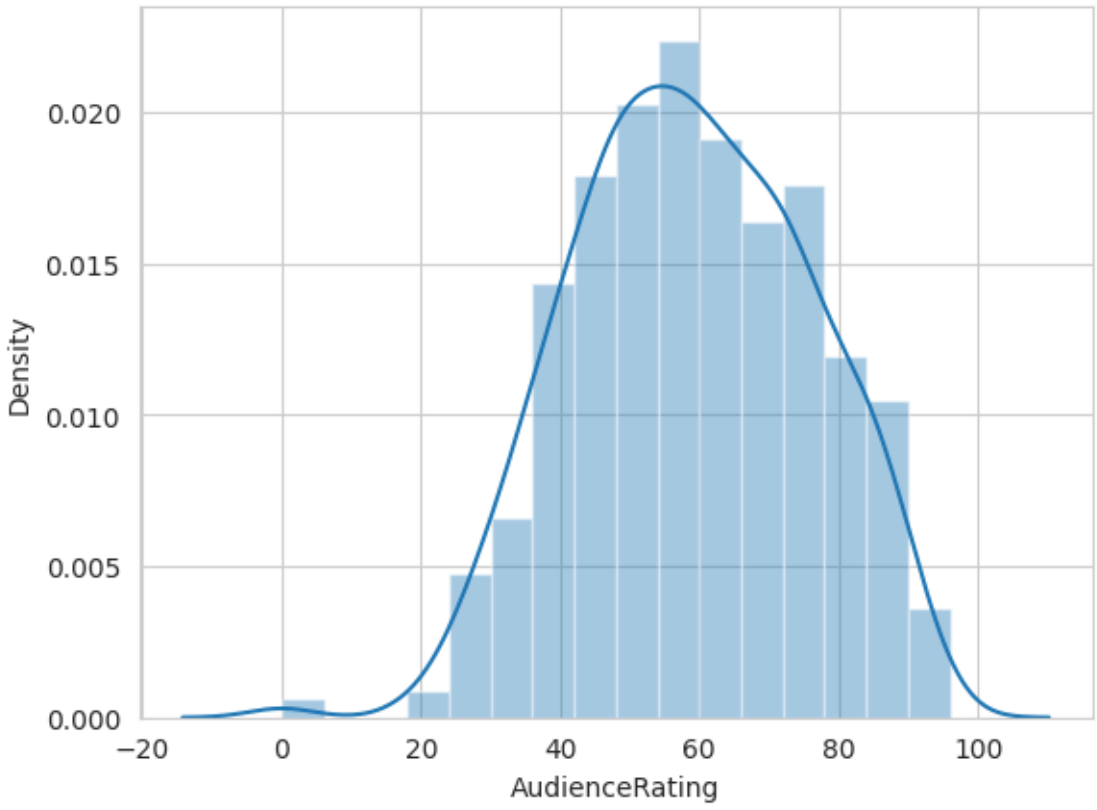


[81]:
```
# For removing frequency line I have to add displot() instead of distplot()
# I used here displot()
audi = sns.displot(rating.AudienceRating)
```

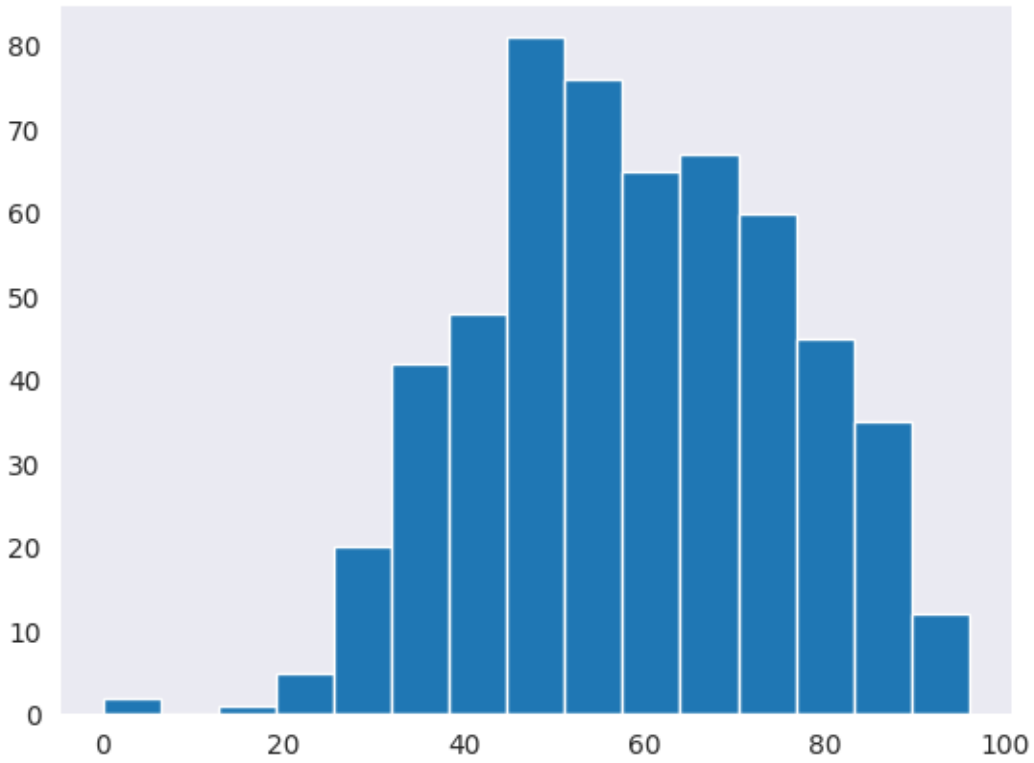


0.0.2 Here I added set_style() funtion

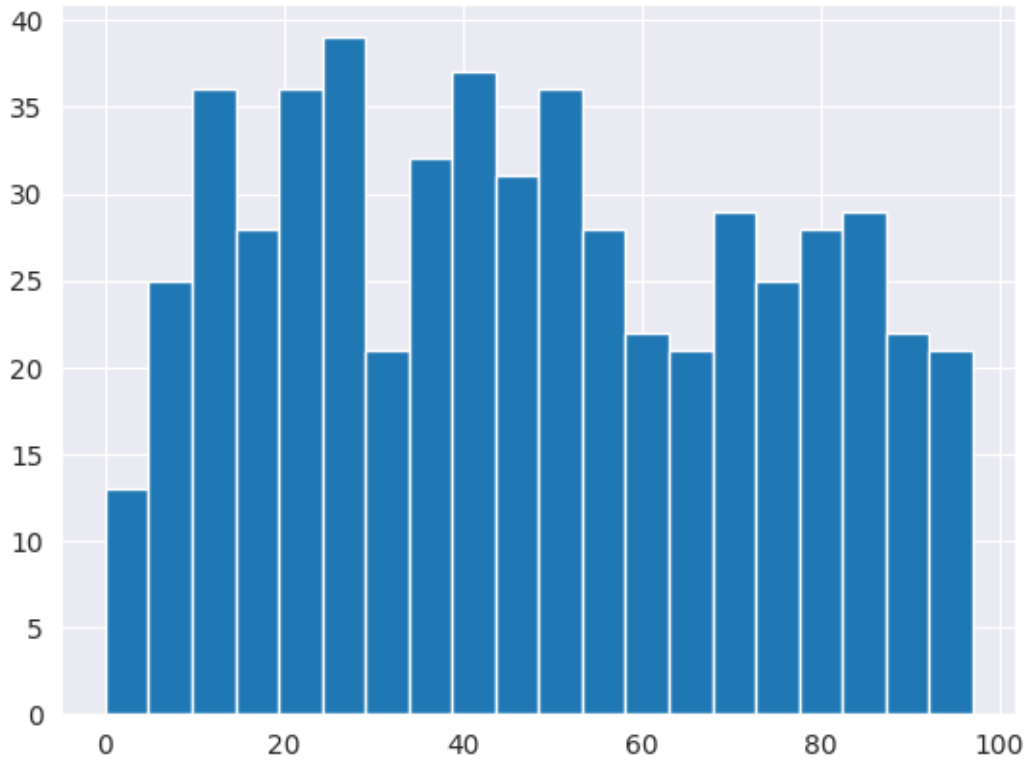
[84]:
```
# The sns.set_style() function in Seaborn is used to set the style of the plots
sns.set_style('whitegrid')
audi = sns.distplot(rating.AudienceRating, bins = 16)
```



[88]:
```
sns.set_style('dark')
audi = plt.hist(rating.AudienceRating, bins=15)
# Normal Distribution
```

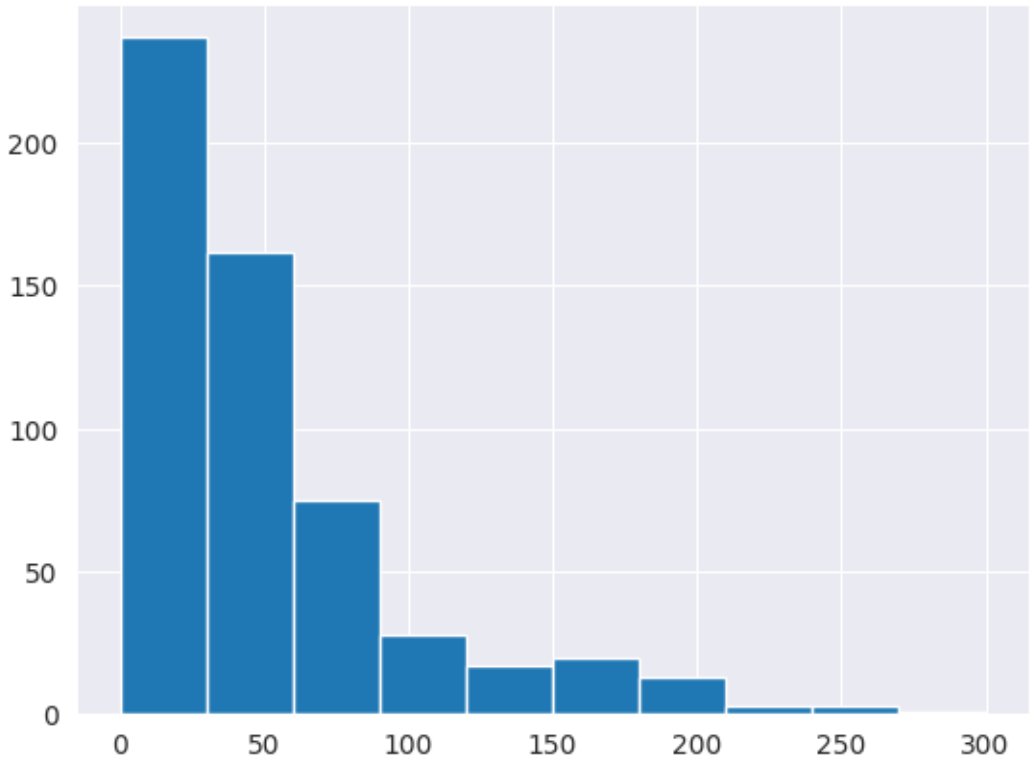


[95]:
```
sns.set_style('darkgrid')
audi = plt.hist(rating.CriticRating, bins=20)
# Uniform Distribution
```

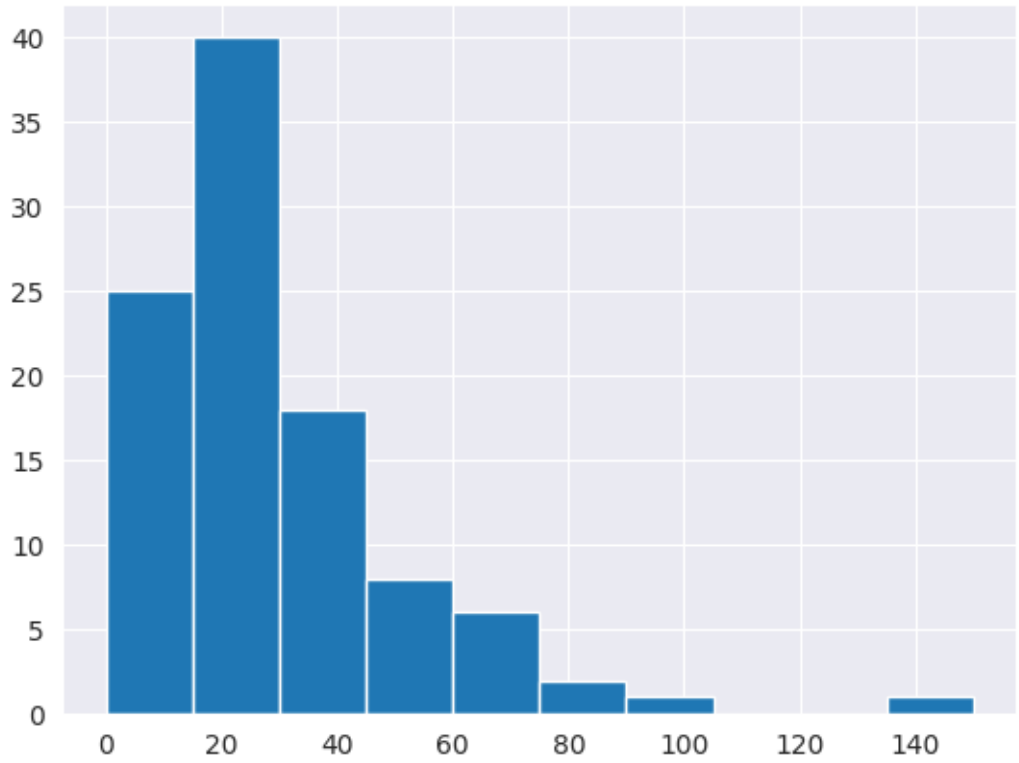


0.0.3 Here I'm creating stacked Histograms:

[97]:
```
plt.hist(rating.BudgetMillions)  # Create a histogram to visualize the␣
↪distribution of BudgetMillions from the movies dataset
plt.show()  # Display the histogram plot
```

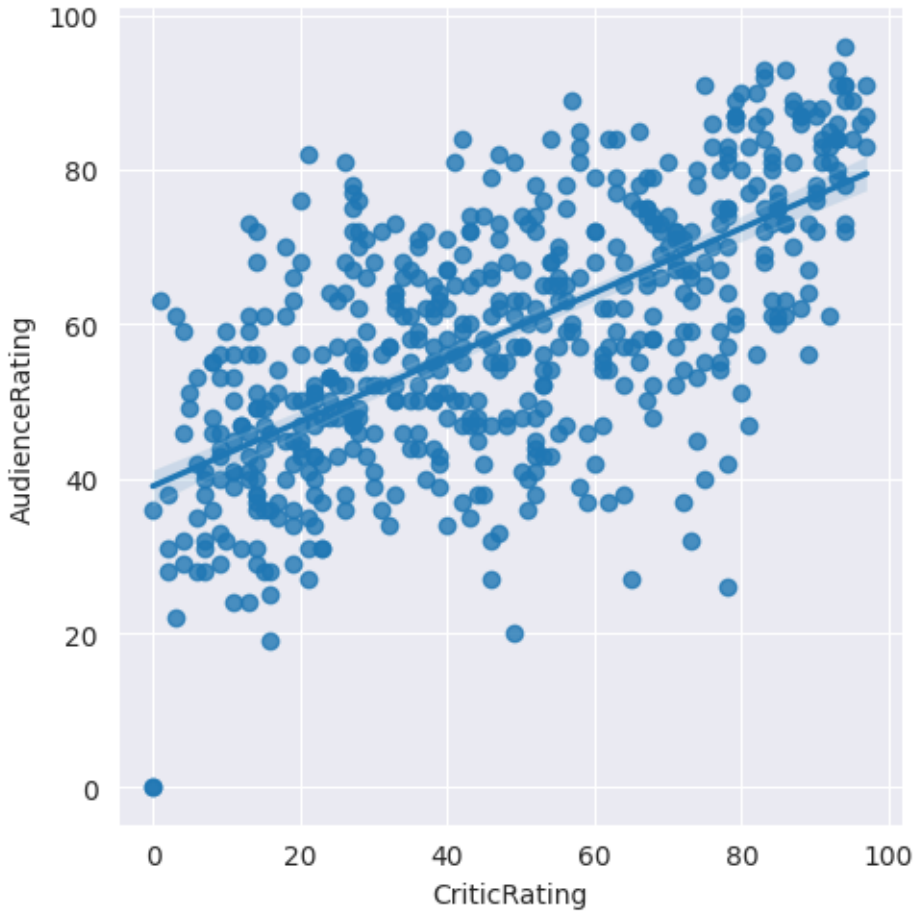


[98]:
```
# It prints the budget of the Drama movies from dataset

plt.hist(rating[rating.Genre == 'Drama'].BudgetMillions)
plt.show()
```

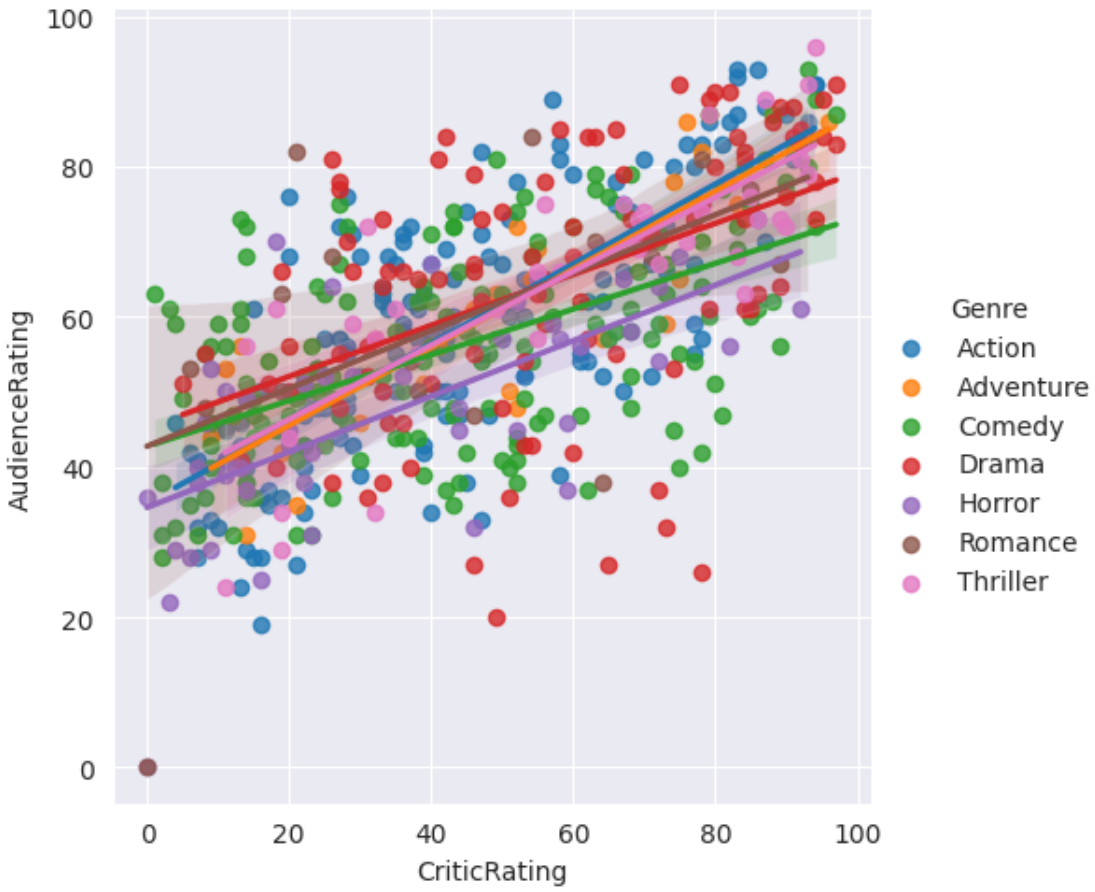


###Here I gonna to show Scatter Plot

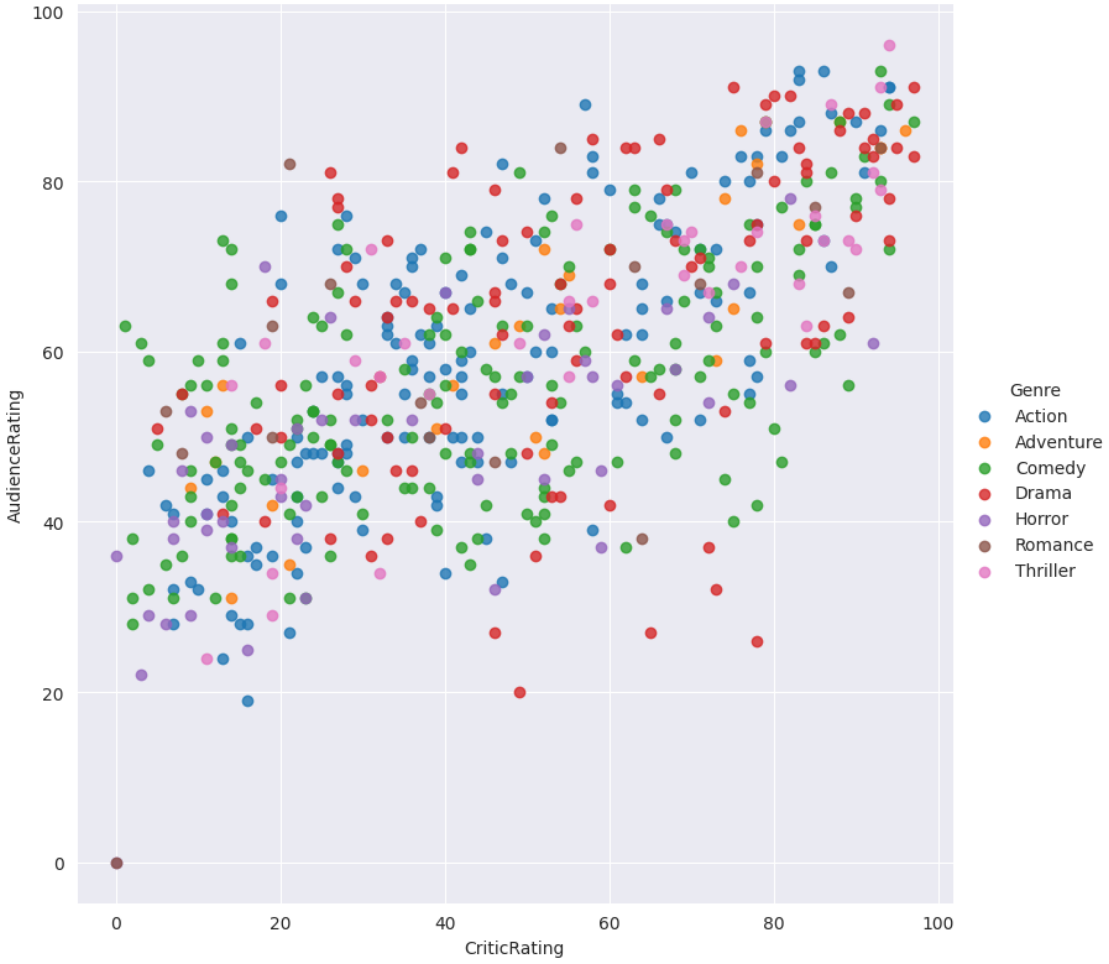
[104]:
```
visual = sns.lmplot(data=rating, x='CriticRating', y='AudienceRating',␣
↪fit_reg=True)
```



[110]:
```
visual = sns.lmplot(data=rating, x='CriticRating', y='AudienceRating',␣
↪fit_reg=True, hue = 'Genre')
```



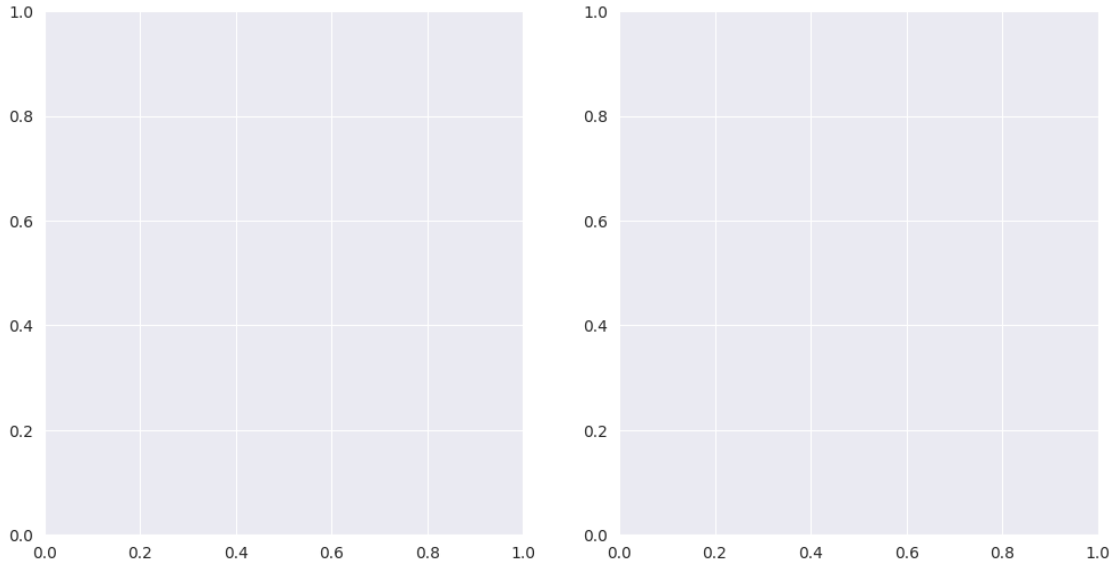
[121]:
```
visual = sns.lmplot(data=rating, x='CriticRating', y='AudienceRating',␣
↪fit_reg=False, hue='Genre', height=8, aspect=1)
```



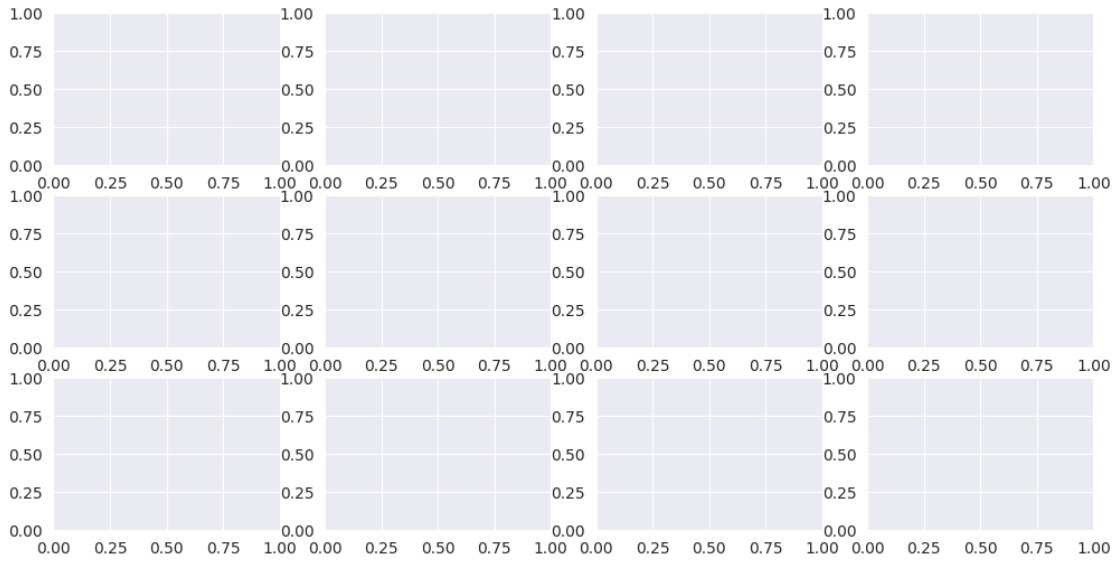
0.0.4 Here I'm going to use plt.subplot()

- Used to create multiple subplots within a single fig/graph
- Used in Grid Layout
- Used to compare in matrix format

[124]:
```
sub = plt.subplots(1,2, figsize =(12,6))
```



[128]:
```
sub = plt.subplots(3,4, figsize =(12,6))
```



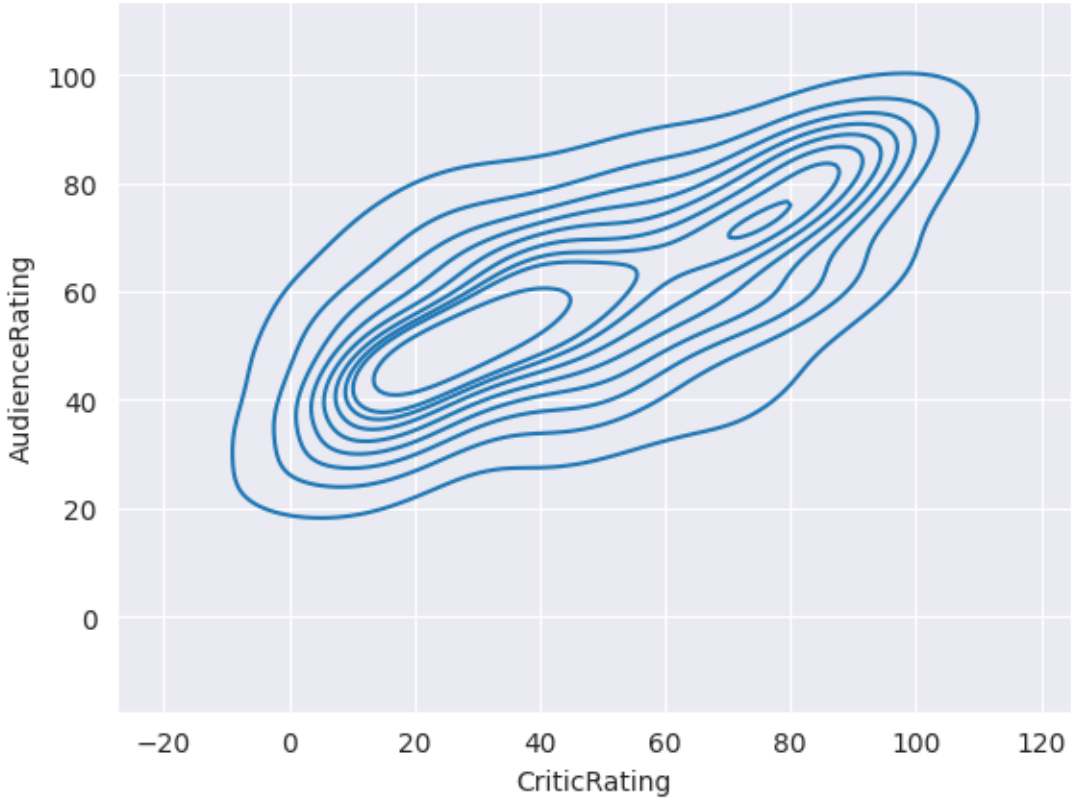
0.0.5 Kernal Density Estimate Plot (KDE PLOT)

- Where do u find more density and how density is distibuted across from the the chat
- Center point is kernal this is calld KDE & insteade of dots it visualize like this
- We can able to clearly see the spread at the audience ratings

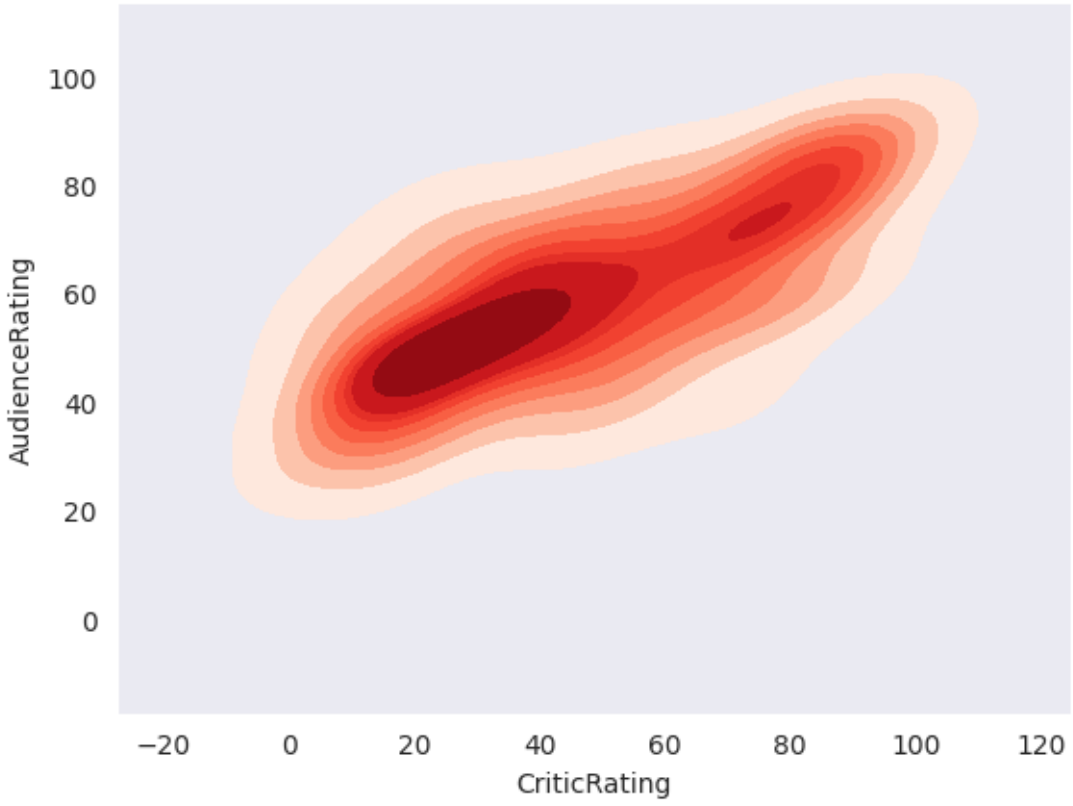
[139]:
```
rating.info()
```

```
<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
```

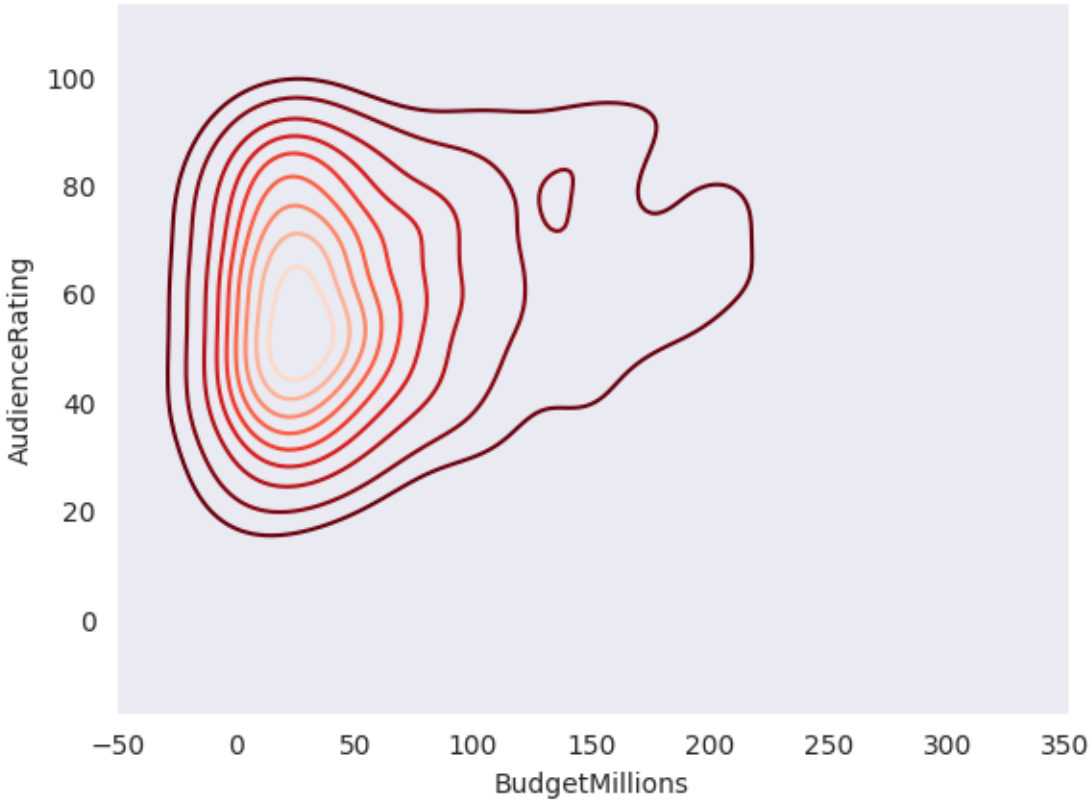
[135]:
```
kernel = sns.kdeplot(x=rating.CriticRating, y=rating.AudienceRating)
```



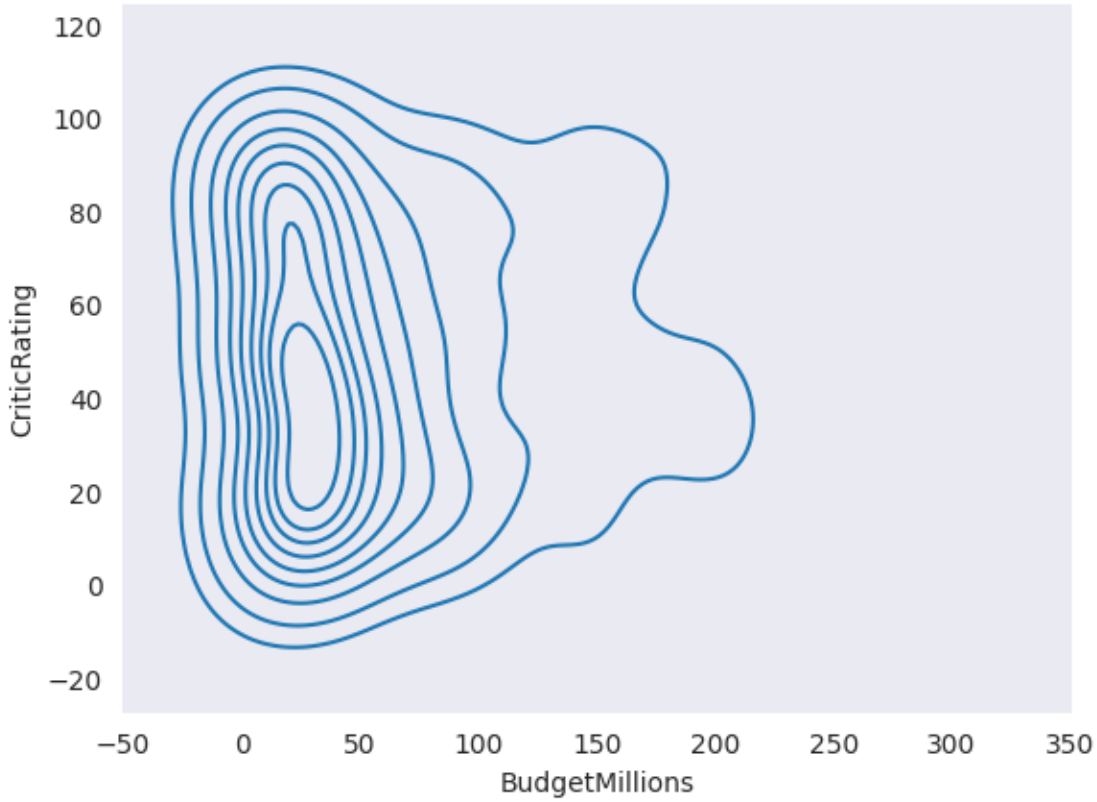
[146]:
```
kernel = sns.kdeplot(x=rating.CriticRating, y=rating.AudienceRating,shade =␣
↪True,shade_lowest=False,cmap='Reds')
```



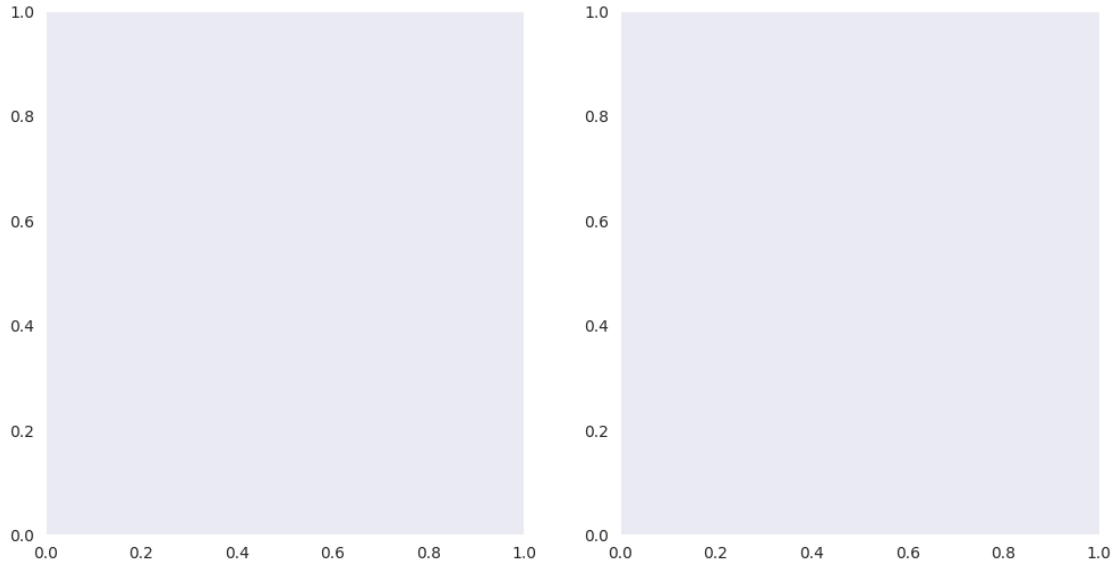
[147]:
```
sns.set_style('dark')
kernel = sns.kdeplot(x = rating.BudgetMillions, y = rating.
↪AudienceRating,shade_lowest=False,cmap='Reds_r')
```



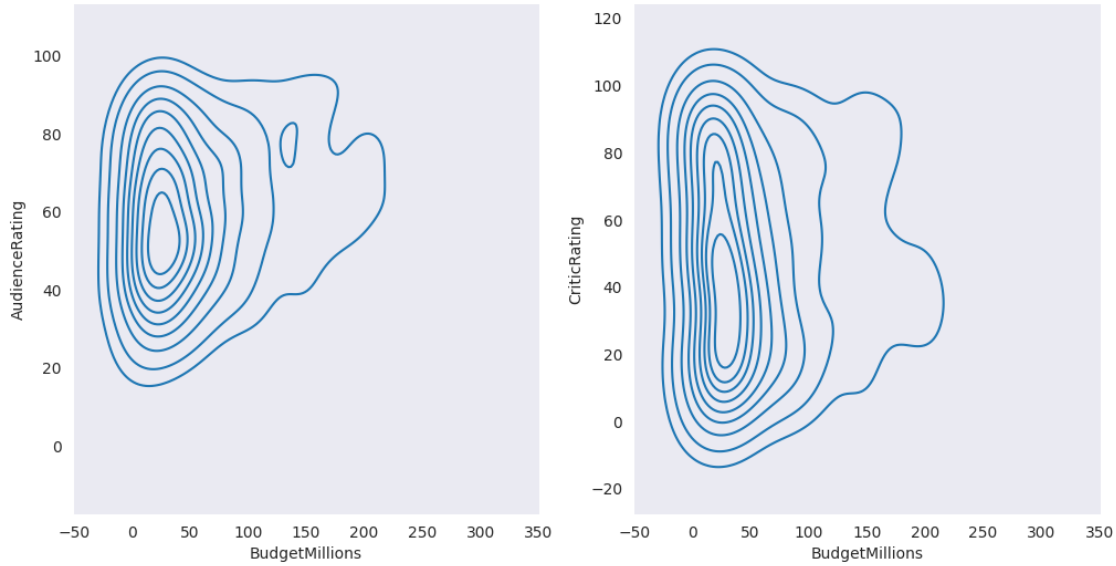
[162]:
```
kernel = sns.kdeplot(x = rating.BudgetMillions, y = rating.CriticRating)
```



[182]:
```
f, ax = plt.subplots(1,2, figsize =(12,6))
```



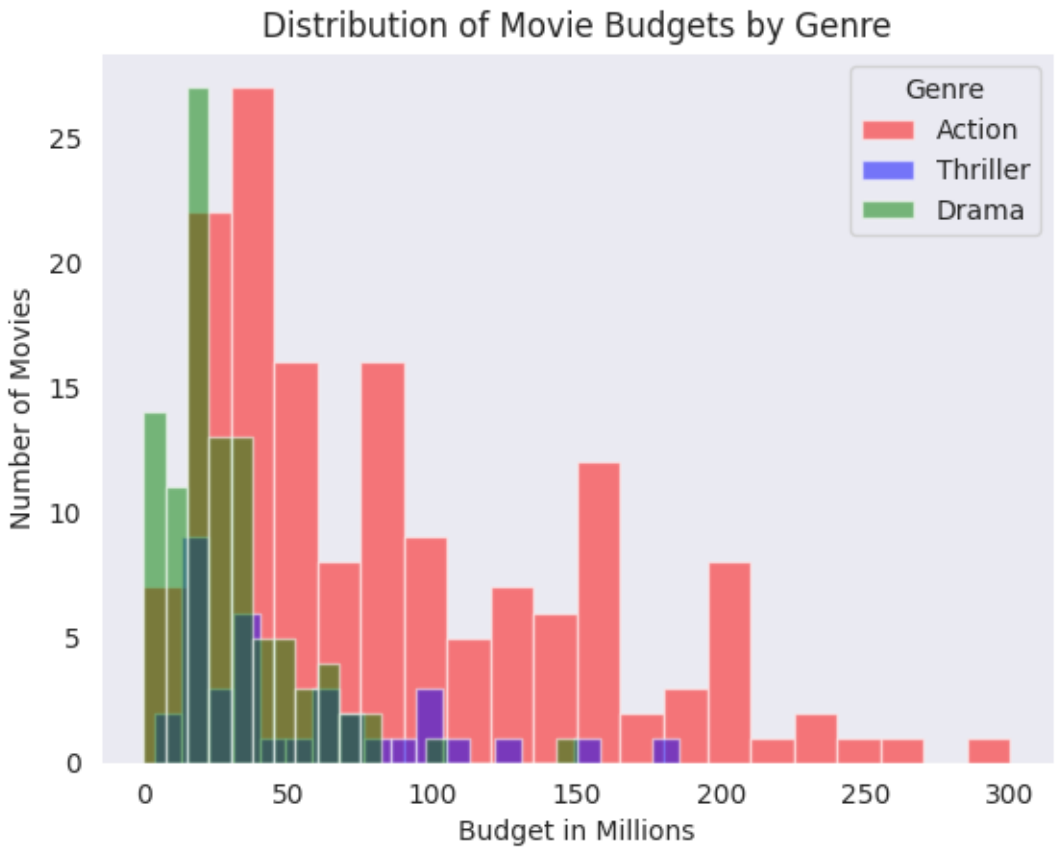
[183]:
```
f, sub = plt.subplots(1,2, figsize =(12,6))

kernel = sns.kdeplot(x = rating.BudgetMillions, y = rating.AudienceRating, ax =␣
↪sub[0])

kernel = sns.kdeplot(x = rating.BudgetMillions, y = rating.CriticRating, ax =␣
↪sub[1])
```



0.0.6 Plots are Stacked Histogram

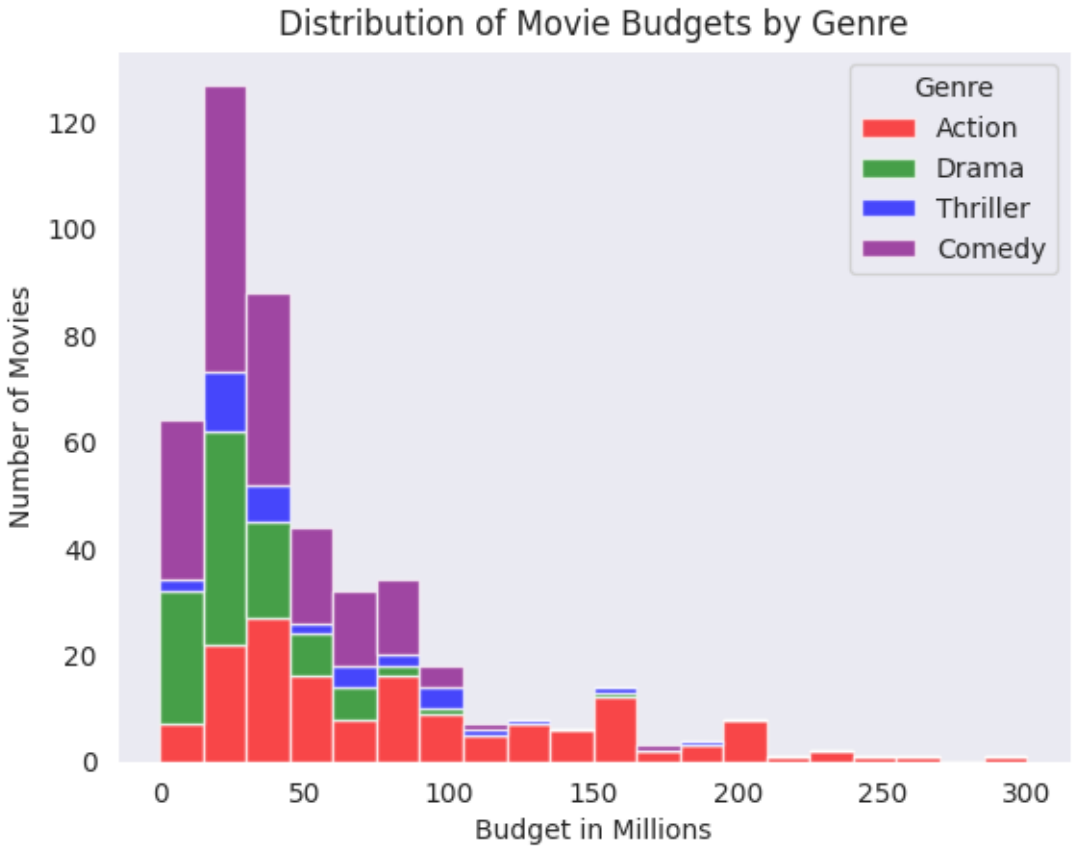
[185]:
```
# Define colors for each genre
colors = {'Action': 'red', 'Thriller': 'blue', 'Drama': 'green'}

# Plot histograms for each genre
plt.hist(rating[rating.Genre == 'Action'].BudgetMillions, bins=20, alpha=0.5,␣
↪color=colors['Action'], label='Action')
plt.hist(rating[rating.Genre == 'Thriller'].BudgetMillions, bins=20, alpha=0.5,␣
↪color=colors['Thriller'], label='Thriller')
plt.hist(rating[rating.Genre == 'Drama'].BudgetMillions, bins=20, alpha=0.5,␣
↪color=colors['Drama'], label='Drama')

# Add labels and title
plt.xlabel('Budget in Millions')
plt.ylabel('Number of Movies')
plt.title('Distribution of Movie Budgets by Genre')

plt.legend(title='Genre')
```

```
plt.show()
```



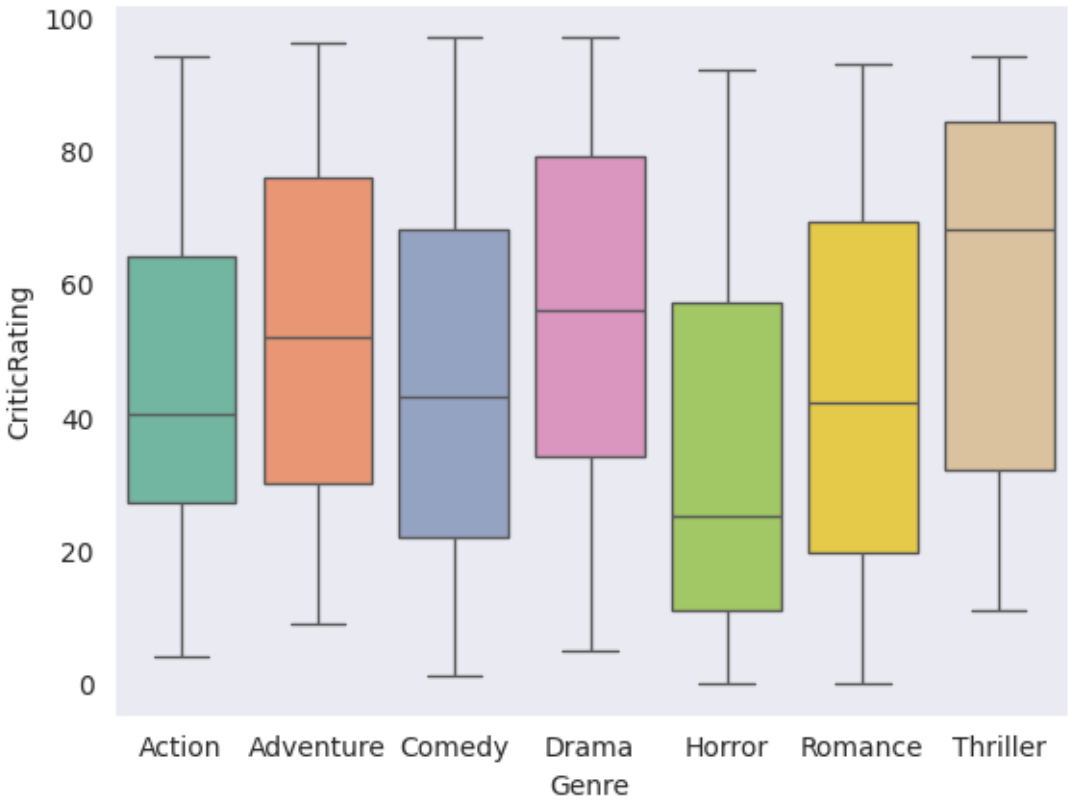
[186]:
```
# Define colors for each genre
colors = ['red', 'green', 'blue', 'purple']
labels = ['Action', 'Drama', 'Thriller', 'Comedy']

# Data for each genre
data = [
    rating[rating.Genre == 'Action'].BudgetMillions,
    rating[rating.Genre == 'Drama'].BudgetMillions,
    rating[rating.Genre == 'Thriller'].BudgetMillions,
    rating[rating.Genre == 'Comedy'].BudgetMillions
]

# Plot stacked histogram
plt.hist(data, bins=20, stacked=True, color=colors, label=labels, alpha=0.7)

# Add labels and title
plt.xlabel('Budget in Millions')
plt.ylabel('Number of Movies')
```

```
plt.title('Distribution of Movie Budgets by Genre')

plt.legend(title='Genre')
plt.show()
```

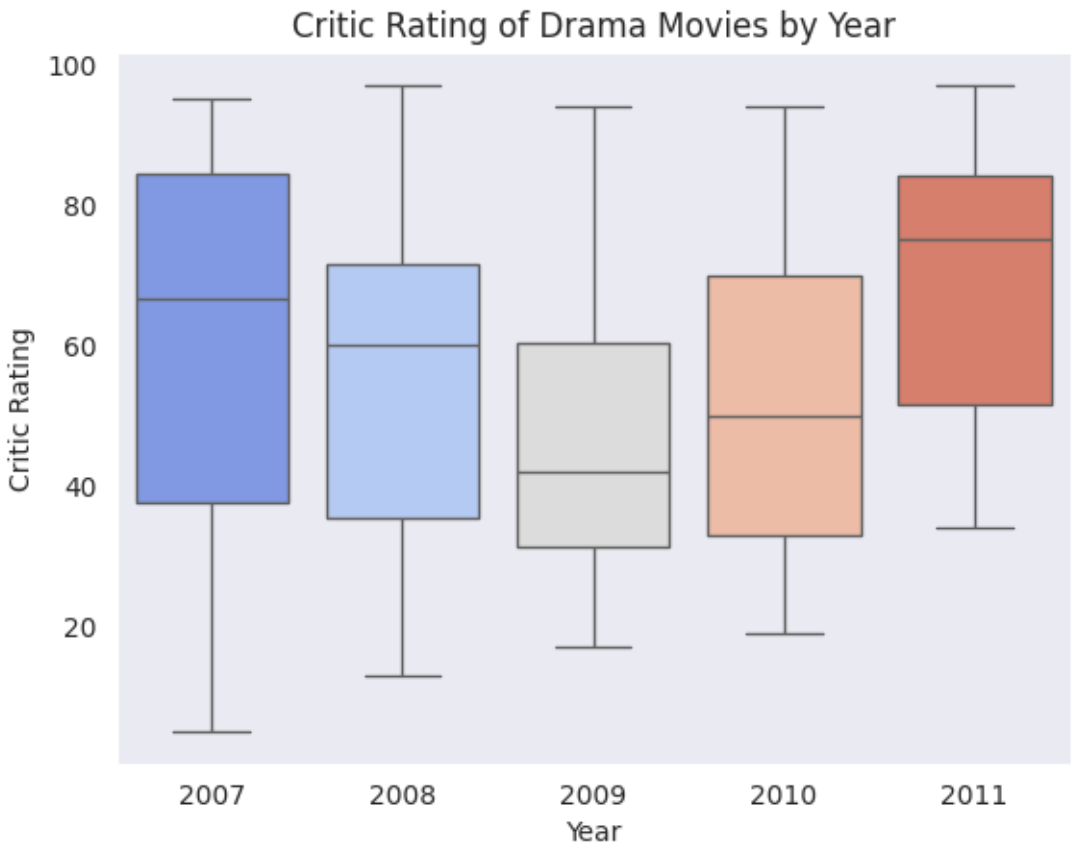


0.0.7 Creating Box Plot, Violin Plot, and Facet Grid with Seaborn

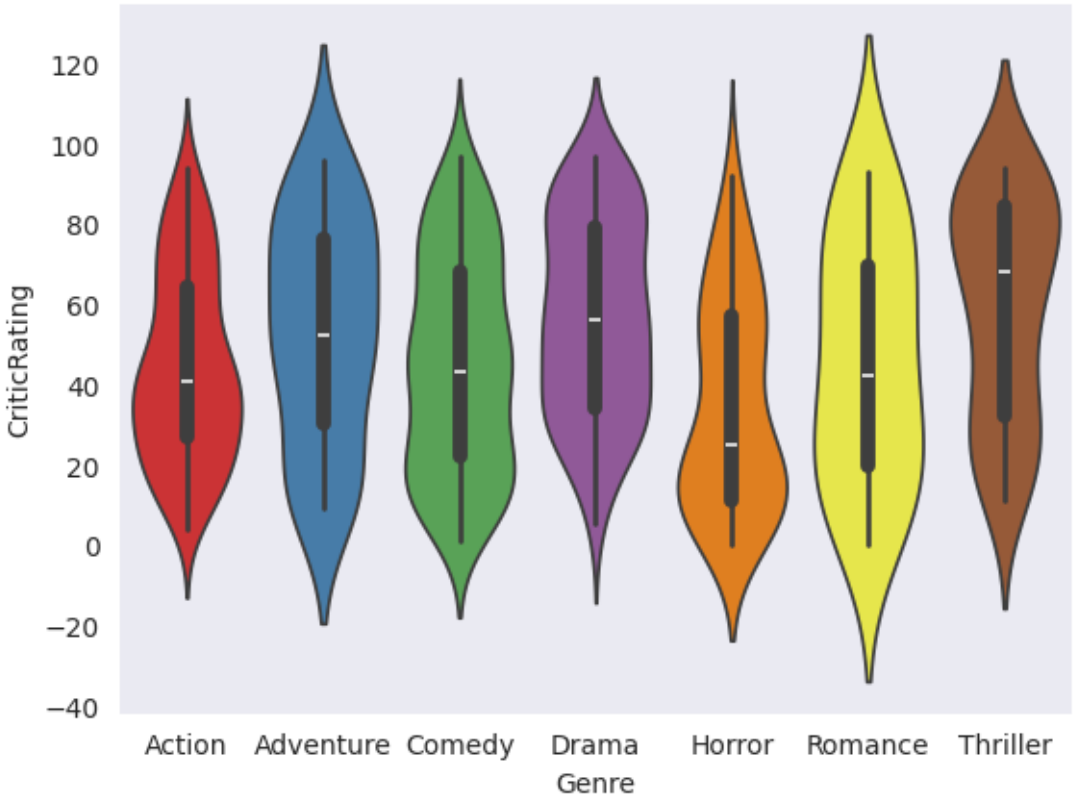
[190]:
```
# Box Plot -
box = sns.boxplot(data=rating, x='Genre', y='CriticRating', palette='Set2')

plt.show()
```



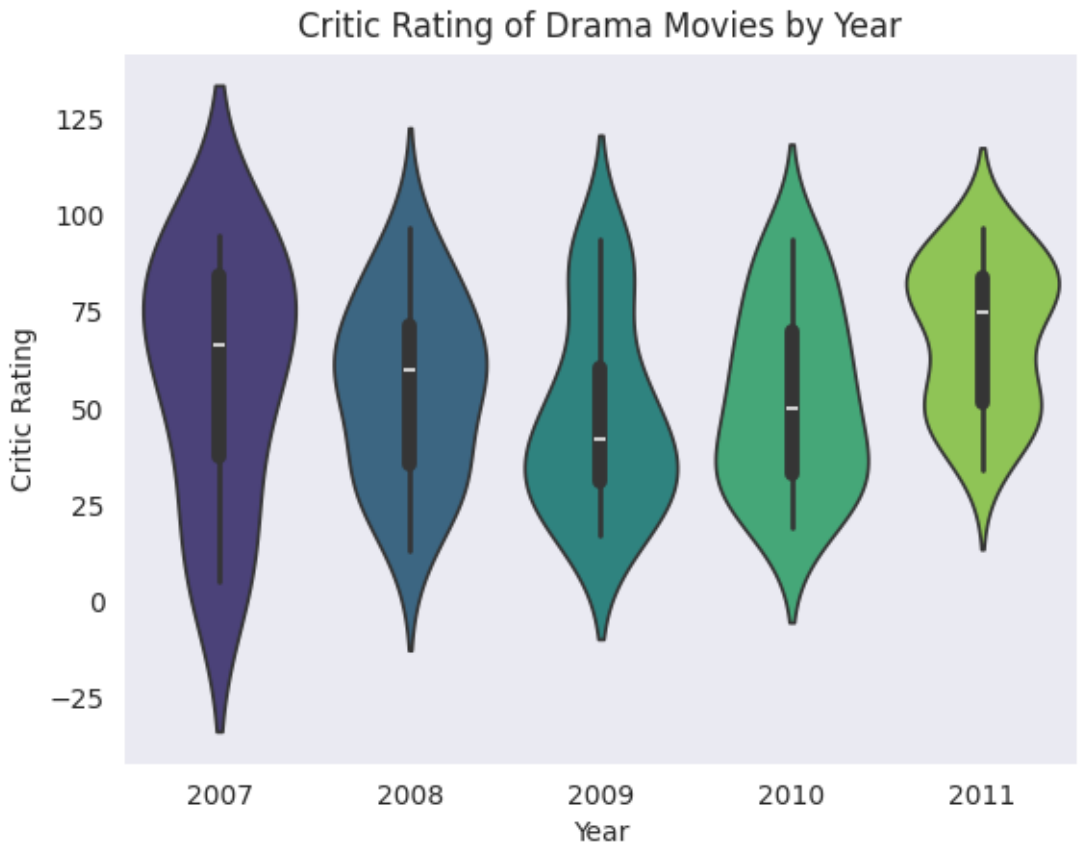
[194]:
```
box = sns.boxplot(data=rating[rating.Genre == 'Drama'], x='Year',␣
 ↪y='CriticRating', palette='coolwarm')

plt.xlabel('Year')
plt.ylabel('Critic Rating')
plt.title('Critic Rating of Drama Movies by Year')

plt.show()
```



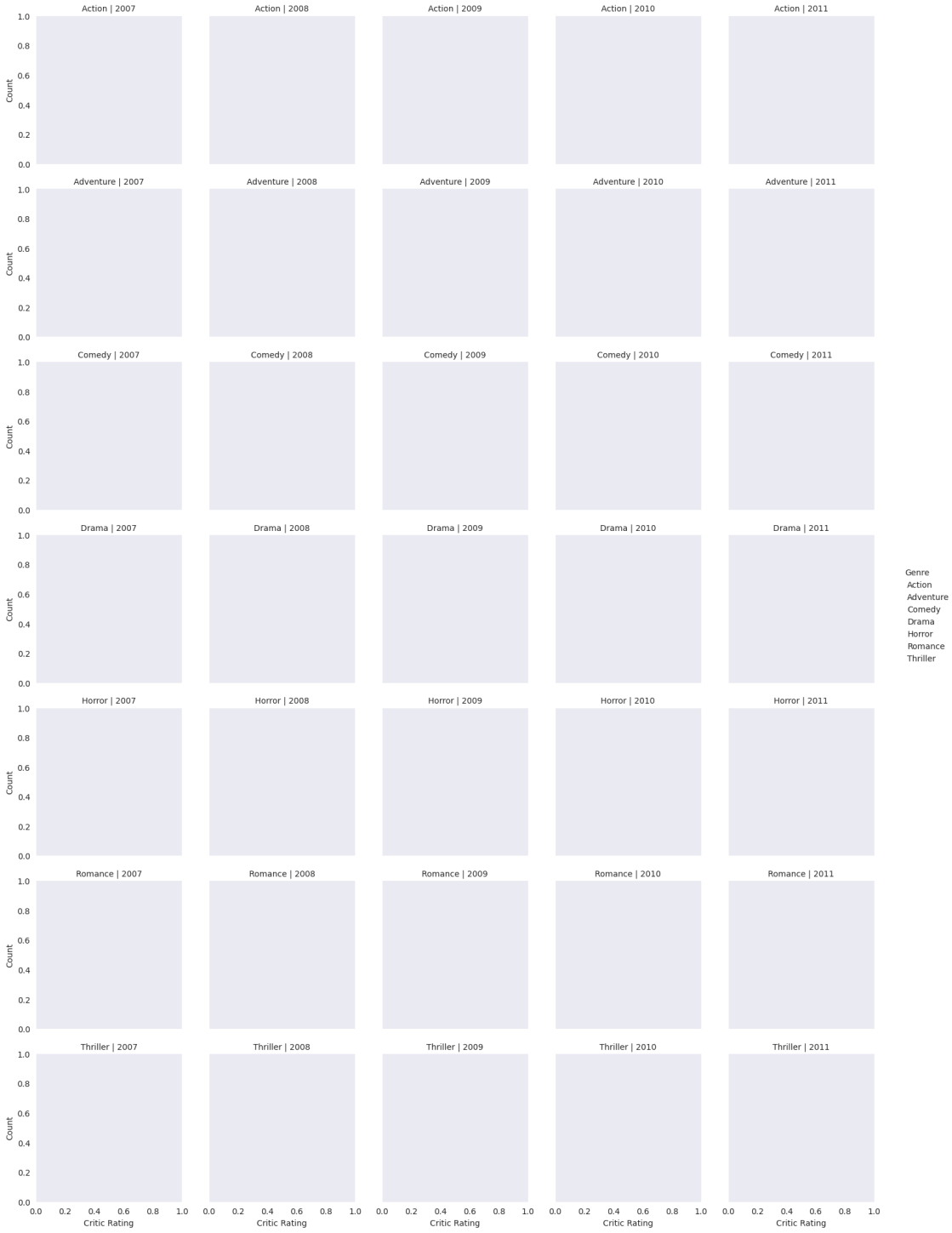
[193]:
```
# Violin Plot
voilin = sns.violinplot(data=rating, x='Genre', y='CriticRating',␣
 ↪palette='Set1')

plt.show()
```



[195]:
```
voilin = sns.violinplot(data=rating[rating.Genre == 'Drama'], x='Year',␣
 ↪y='CriticRating', palette='viridis')

plt.xlabel('Year')
plt.ylabel('Critic Rating')
plt.title('Critic Rating of Drama Movies by Year')

plt.show()
```



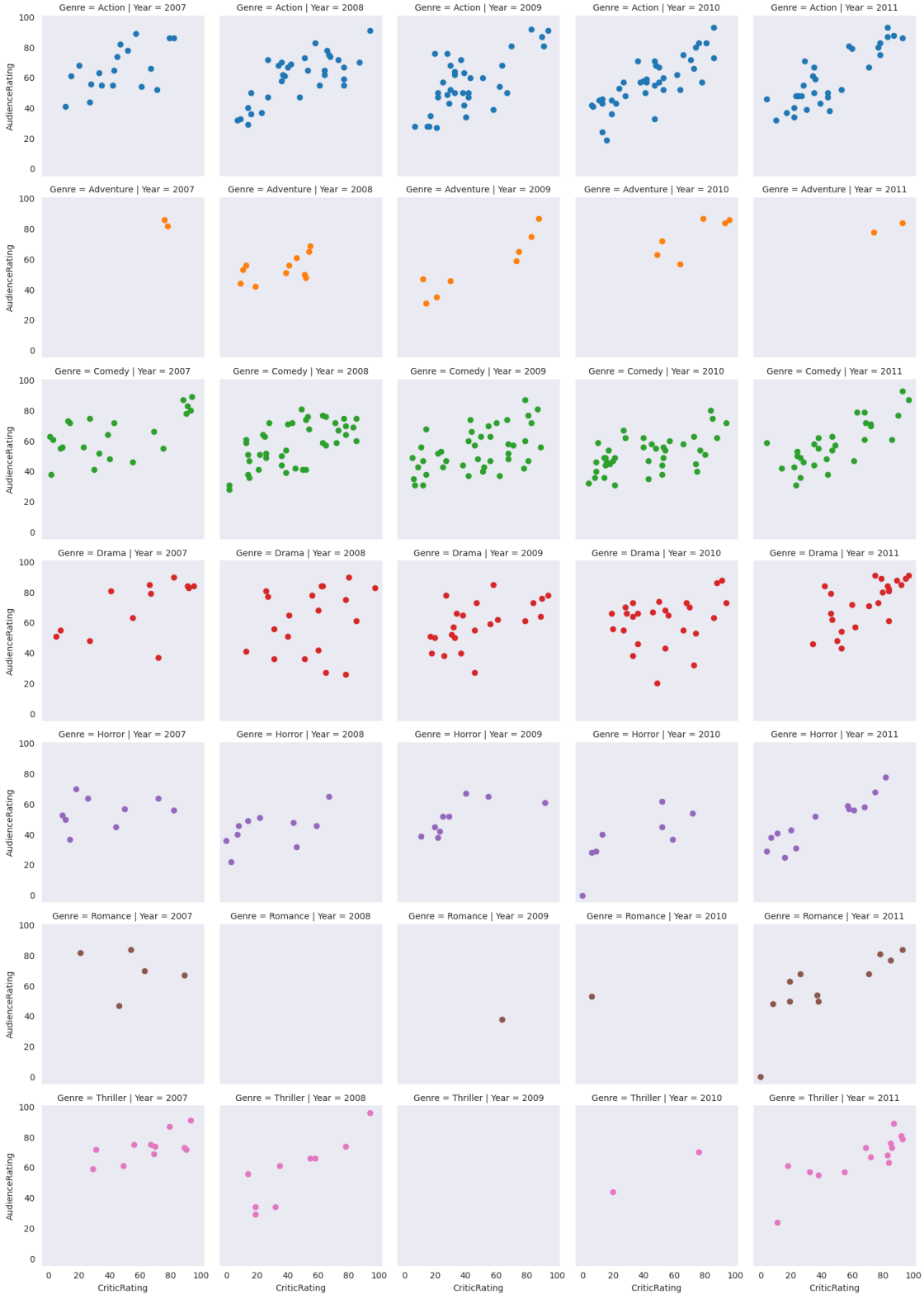
[197]:
```
# Creating a Facet grid, kind of subplots in Seaborn

facet = sns.FacetGrid(rating, row='Genre', col='Year', hue='Genre')

# Add legend and adjust layout
facet.add_legend()
facet.set_axis_labels('Critic Rating', 'Count')
facet.set_titles(col_template='{col_name}', row_template='{row_name}')

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



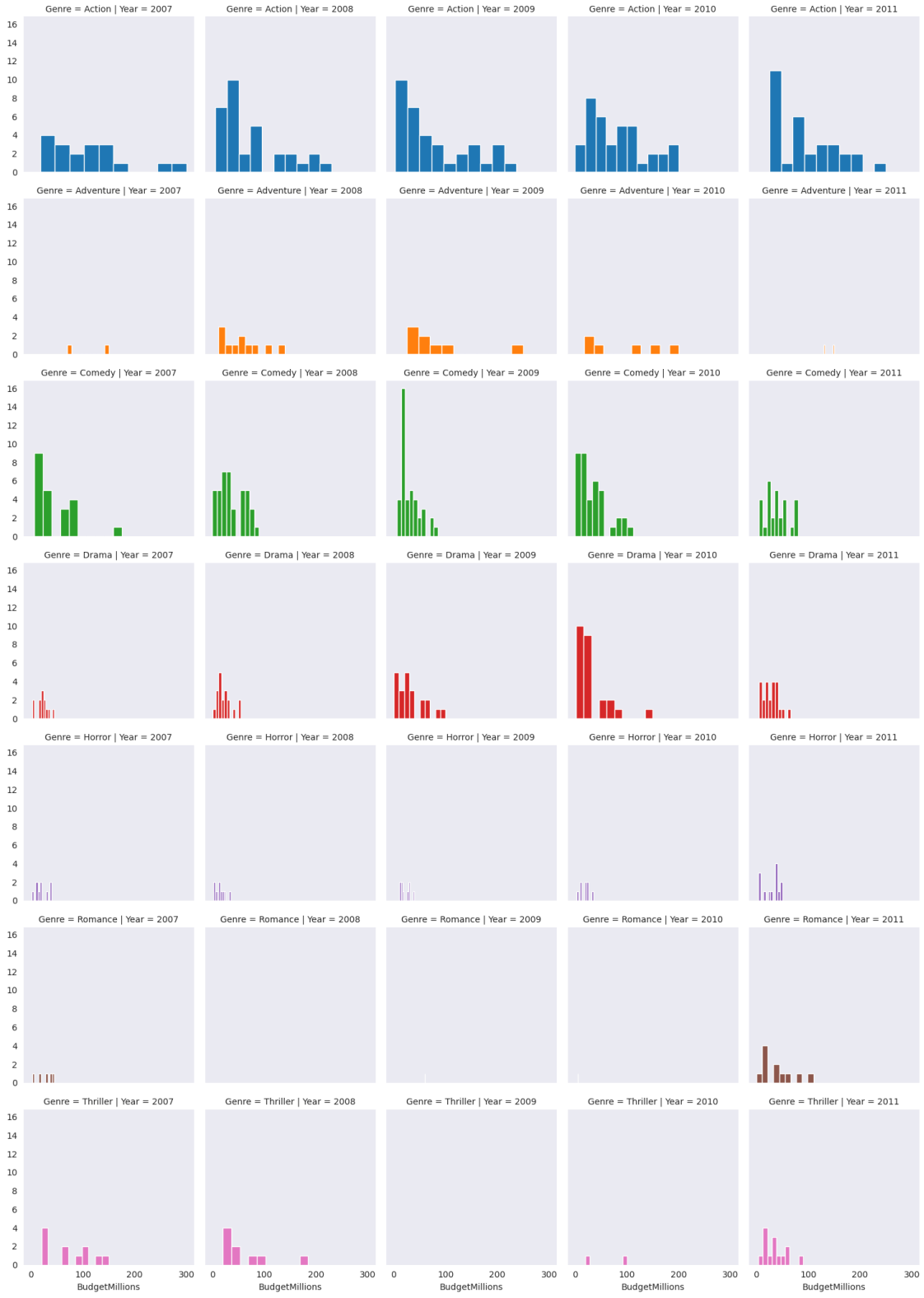
###Lets now mapped Scatter Plots, Line Plots in the Facet Grid

[199]:
```
# Scatterplots are mapped in facetgrid
scat =sns.FacetGrid (rating, row = 'Genre', col = 'Year', hue = 'Genre')
```

```
scat = scat.map(plt.scatter, 'CriticRating', 'AudienceRating')
```

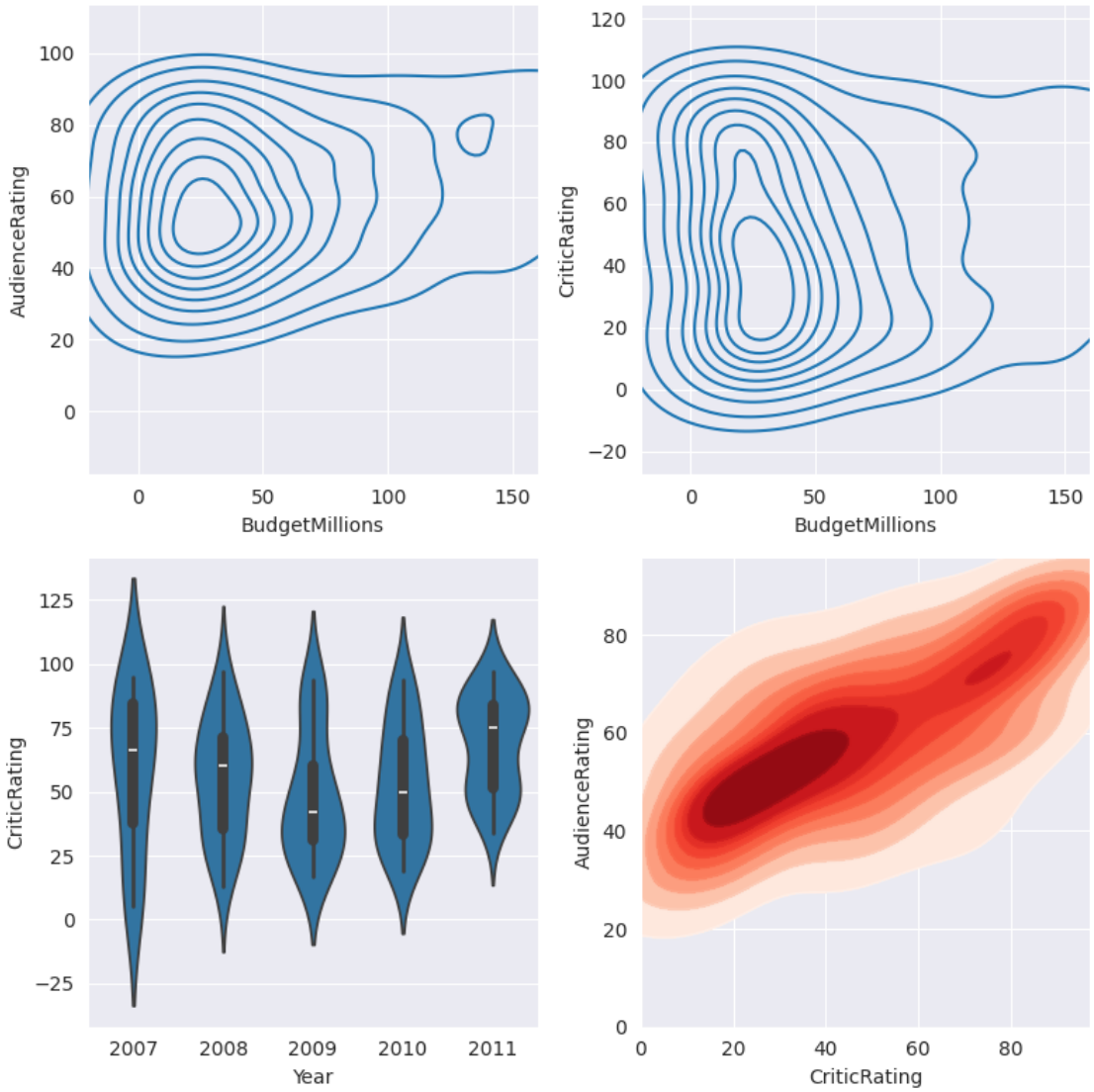


[200]:
```
# Line Plot are mapped in facet grid
line = sns.FacetGrid (rating, row = 'Genre', col = 'Year', hue = 'Genre')
line = line.map(plt.hist, 'BudgetMillions')
```


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
BudgetMillions

0.0.8 Building Dashboards (Dashboard - Combination of chats)

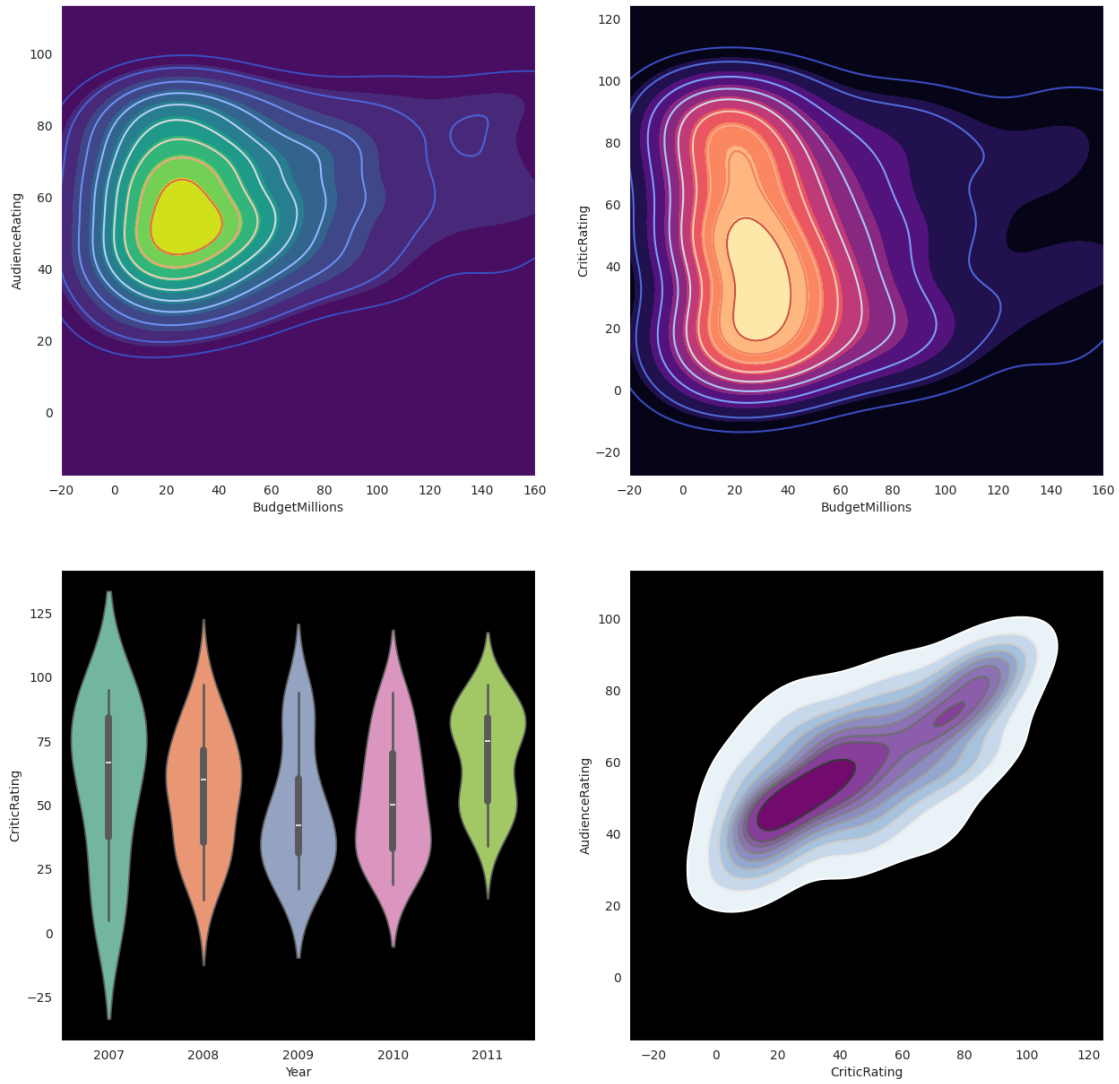
[207]:
```
sns.set_style('darkgrid')
f, sub = plt.subplots(2, 2, figsize=(8, 8))

# KDE Plots
kernel1 = sns.kdeplot(x=rating.BudgetMillions, y=rating.AudienceRating,␣
 ↪ax=sub[0, 0])
kernel1.set(xlim=(-20, 160))

kernel2 = sns.kdeplot(x=rating.BudgetMillions, y=rating.CriticRating, ax=sub[0,␣
 ↪1])
kernel2.set(xlim=(-20, 160))

# Violin Plot
violin = sns.violinplot(data=rating[rating.Genre == 'Drama'], x='Year',␣
 ↪y='CriticRating', ax=sub[1, 0])

# KDE Plots with shading
kernel3 = sns.kdeplot(x=rating.CriticRating, y=rating.AudienceRating,␣
 ↪shade=True, shade_lowest=False, cmap='Reds', ax=sub[1, 1])
kernel3.set(xlim=(rating.CriticRating.min(), rating.CriticRating.max()),␣
 ↪ylim=(rating.AudienceRating.min(), rating.AudienceRating.max()))

# KDE Plot without shading
kernel4 = sns.kdeplot(x=rating.CriticRating, y=rating.AudienceRating,␣
 ↪cmap='Reds', ax=sub[1, 1], alpha=0.5)

# Adjust subplot layout
plt.tight_layout()

plt.show()
```



0.0.9 Here you can style your dashboard using different color map

[212]:
```
# Set the style and background color
sns.set_style('dark', {'axes.facecolor':'black'})

# Create subplots
f, axes = plt.subplots(2, 2, figsize=(15, 15))

# Plot [0,0]
sns.kdeplot(x=rating.BudgetMillions, y=rating.AudienceRating, shade=True,␣
 ↪shade_lowest=True, ax=axes[0, 0], cmap='viridis')
```

```
sns.kdeplot(x=rating.BudgetMillions, y=rating.AudienceRating, cmap='coolwarm',␣
 ↪ax=axes[0, 0])

# Plot [0,1]
sns.kdeplot(x=rating.BudgetMillions, y=rating.CriticRating, shade=True,␣
 ↪shade_lowest=True, cmap='magma', ax=axes[0, 1])
sns.kdeplot(x=rating.BudgetMillions, y=rating.CriticRating, cmap='coolwarm',␣
 ↪ax=axes[0, 1])

# Plot [1,0]
sns.violinplot(data=rating[rating.Genre == 'Drama'], x='Year',␣
 ↪y='CriticRating', ax=axes[1, 0], palette='Set2')

# Plot [1,1]
sns.kdeplot(x=rating.CriticRating, y=rating.AudienceRating, shade=True,␣
 ↪shade_lowest=False, cmap='BuPu', ax=axes[1, 1])
sns.kdeplot(x=rating.CriticRating, y=rating.AudienceRating, cmap='Greys',␣
 ↪ax=axes[1, 1])

# Set axis limits
axes[0, 0].set_xlim(-20, 160)
axes[0, 1].set_xlim(-20, 160)

plt.show()
```



0.1 Completion of Movies Rating Analysis Project

###Key Learnings: 1. Category Data Type: Efficiently manage categorical data. 2. Joint Plots: Visualize relationships and distributions of two variables. 3. Histograms: Display frequency distribution of movie ratings. 4. Stacked Histograms: Compare rating distributions across categories. 5. KDE Plots: Estimate and visualize density of rating distributions. 6. Subplots: Compare multiple visualizations in one figure. 7. Violin Plots: Show distribution and spread of ratings across categories. 8. Facet Grid: Create subplots for detailed comparison based on categories. 9. Dashboards: Integrate visualizations for interactive exploration of data.

[214]:
```
#thank you for visiting here!
#rohanwaghmare
```