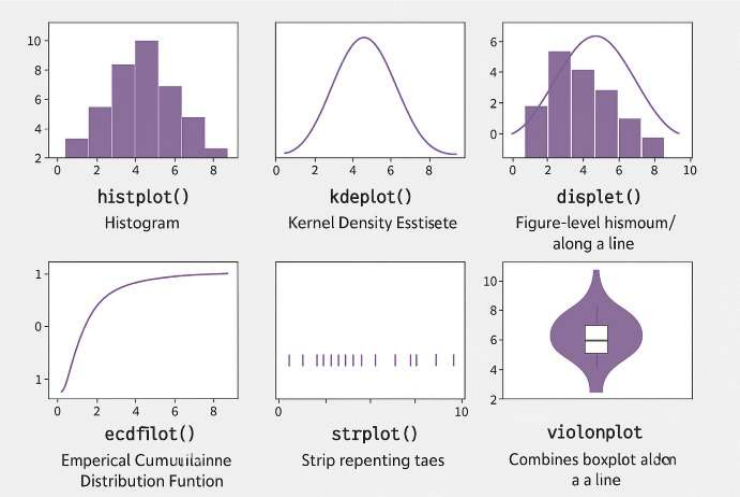
ADVANCE VISUALIZATION (SEABORN) ON MOVIE RATING

Seaborn visualizations

1. Distributions of a Single Variable

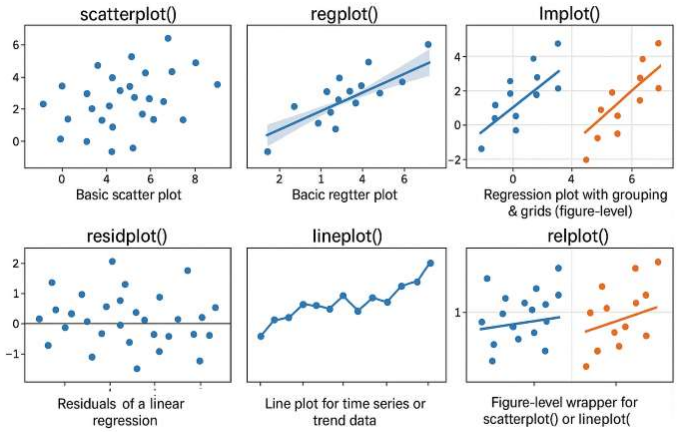
Function	Description
`histplot()`	Histogram of data
`kdeplot()`	Kernel Density Estimate (smooth curve)
`displot()`	Figure-level histogram/KDE/ECDF
`ecdfplot()`	Empirical Cumulative Distribution Function
`rugplot()`	Small vertical lines for each data point
`stripplot()`	Dots representing values along a line
`violinplot()`	Combines boxplot and KDE in one



2. Relationships Between Two Variables

Function	Description
`scatterplot()`	Basic scatter plot

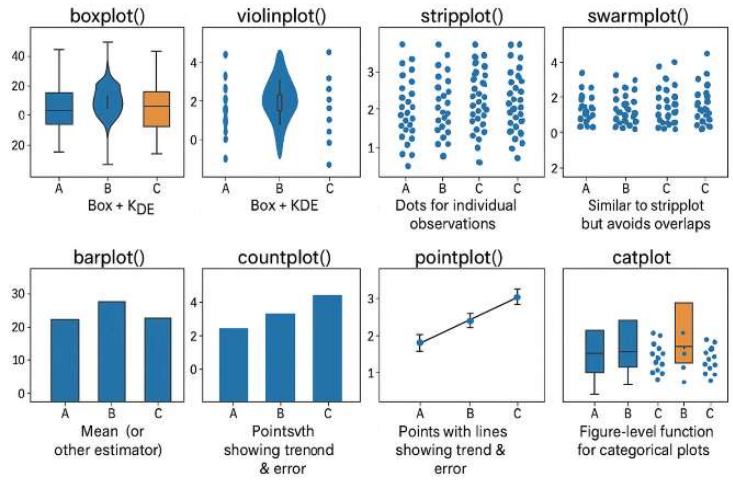
Function	Description
`regplot()`	Scatter plot with regression line (low-level)
`lmplot()`	Regression plot with grouping & grids (figure-level)
`residplot()`	Residuals of a linear regression
`lineplot()`	Line plot for time series or trend data
`relplot()`	Figure-level wrapper for `scatterplot()` or `lineplot()`



3. Categorical Plots

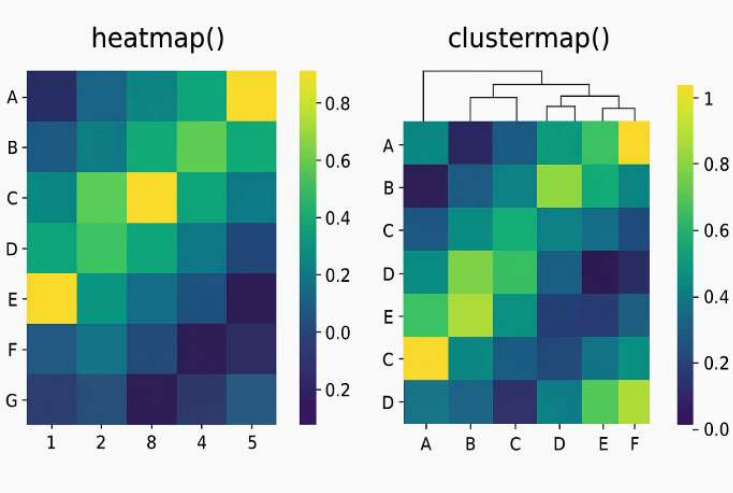
Function	Description
`boxplot()`	Box-and-whisker plot
`violinplot()`	Box + KDE
`stripplot()`	Dots for individual observations
`swarmplot()`	Similar to stripplot but avoids overlaps
`barplot()`	Shows mean (or other estimator) with error bars
`countplot()`	Bar chart showing count of occurrences
`pointplot()`	Points with lines showing trend & error
`catplot()`	Figure-level function for categorical plots

Function	Description



4. Matrix Plots

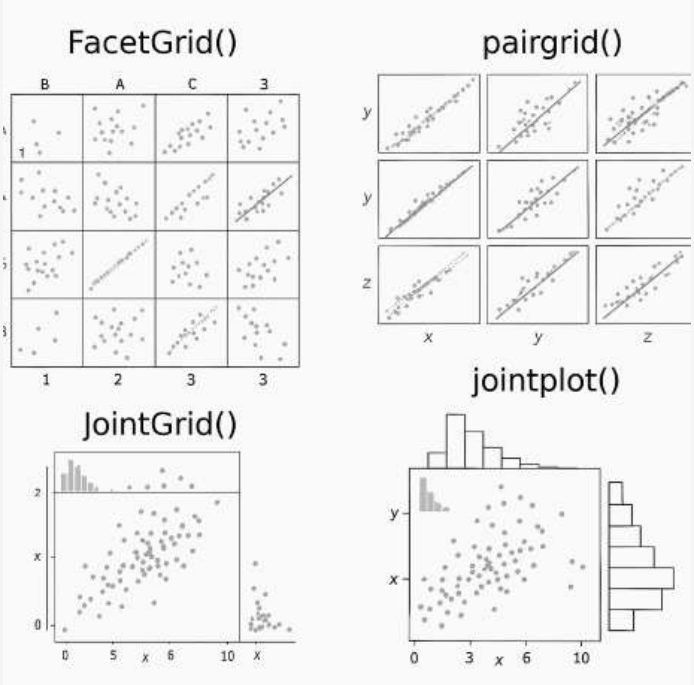
Function	Description
`heatmap()`	Color-coded matrix plot
`clustermap()`	Heatmap with clustering/dendrograms



5. Multi-Plot Grids (Figure-Level)

Function	Description
`FacetGrid()`	Grid of subplots based on row/col variables
`pairplot()`	Grid of scatterplots for pairwise comparisons

Function	Description
`pairgrid()`	Customizable version of `pairplot()`
`JointGrid()`	Grid for joint + marginal plots
`jointplot()`	Easy version of `JointGrid()`



6. Themes and Utilities

Function	Description
`set_theme()`	Set global aesthetic style
`set_style()`	Choose from `'darkgrid'`, `'whitegrid'`, `'dark'`, `'white'`, `'ticks'`
`set_context()`	Control font size and plot scale
`color_palette()`	Set color schemes
`despine()`	Remove top/right axis lines

What Is a Movie Rating?

A movie rating is a score given to a film to reflect how well it was received. A movie rating is a numerical or categorical score given to a film that reflects how well it was received by audiences or critics. It serves as a measure of the film's quality, popularity, or impact, based on various factors like storytelling, acting, direction, visuals, and entertainment value.

Reading the Dataset: Movie-Rating.csv

This dataset contains information about 559 movies, covering their genre, audience and critic ratings, budget, and release year. It can be used for data analysis, machine learning, or visualization projects related to the movie industry.

	Film	Genre	Rotten Tor	Audience F	Budget (mi	Year of release
2	(500) Days	Comedy	87	81	8	2009
3	10,000 B.C	Adventure	9	44	105	2008
4	12 Rounds	Action	30	52	20	2009
5	127 Hours	Adventure	93	84	18	2010
6	17 Again	Comedy	55	70	20	2009
7	2012	Action	39	63	200	2009
8	27 Dresses	Comedy	40	71	30	2008
9	30 Days of	Horror	50	57	32	2007
10	30 Minute:	Comedy	43	48	28	2011
11	50/50	Comedy	93	93	8	2011
12	88 Minute:	Drama	5	51	30	2007
13	A Dangero	Drama	79	89	20	2011
14	A Nightma	Horror	13	40	35	2010
15	A Serious I	Drama	89	64	7	2009
16	A Very Har	Comedy	72	71	19	2011
17	Abduction	Action	4	46	35	2011
18	Across the	Romance	54	84	45	2007
19	Adventure	Comedy	89	56	10	2009

Column Name	Description
`Film`	Title of the movie.
`Genre`	Category of the movie (e.g., Action, Comedy, Adventure).
`Rotten Tomatoes Ratings %`	Rating given by critics on Rotten Tomatoes (0–100%).
`Audience Ratings %`	Rating given by general audiences (0–100%).
`Budget (million $)`	Estimated budget in million USD.
`Year of release`	The year the movie was released.

All Columns in the Movie Ratings Dataset

1. Film :

► A movie title or name that identifies the film.

► It is the official name used for release and marketing.

► Example: “Inception”, “Titanic”, “Avatar”.

► Each row represents a different movie title in this column.

---

2. Genre :

► Describes the type or category of the movie.

► Examples include: Action, Comedy, Drama, Adventure, etc.

► It helps classify the movie based on story, tone, and themes.

► Useful for analyzing which genres perform better in ratings or budget.

---

3. Rotten Tomatoes Rating (%) :

► A movie rating given by professional film critics on Rotten Tomatoes.

► A score out of 100, representing the percentage of positive critic reviews.

► For example, a rating of 87% means 87% of critics liked the movie.

► Helps determine critical acclaim and media reception.

---

4. Audience Rating (%) :

► A movie rating given by the general audience or public viewers.

► Also a score out of 100, reflecting how much the public enjoyed the movie.

► For example, 81% means 81% of viewers gave it a favorable rating.

► Indicates public popularity or entertainment value.

---

5. Budget (million $) :

► The estimated production cost of the movie in millions of US dollars.

► Example: A budget of 100 means the movie cost $100 million to make.

► Includes expenses like actors' salaries, sets, VFX, and marketing.

► Can be used to compare with ratings to analyze budget vs. success.

6.Year of Release :

► The calendar year in which the movie was officially released in theaters.

► Example: 2010, 2015, 2022, etc.

► Useful for time-based analysis of ratings, genre trends, or industry changes.

► Helps track how movies have evolved over the years.

Movie data analysis

In [5]:
```
import pandas as pd
movies=pd.read_csv(r'C:\Users\91630\Desktop\FULL STACK DATASCIENCE AND GENAI\ClASSR
movies
```

Out[5]:

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million $)	Year of release
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009
...	...	...	...	...	...	...
554	Your Highness	Comedy	26	36	50	2011
555	Youth in Revolt	Comedy	68	52	18	2009
556	Zodiac	Thriller	89	73	65	2007
557	Zombieland	Action	90	87	24	2009
558	Zookeeper	Comedy	14	42	80	2011

559 rows × 6 columns

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

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

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

Out[7]:
```
559
```

In [8]:
```
import numpy as np
print(np. __version__)
```

```
1.26.4
```

In [9]:
```
import pandas as pd
print(pd. __version__)
```

```
2.2.2
```

In [10]:
```
movies.columns
```

Out[10]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

In [11]:
```
movies.shape
```

Out[11]:
```
(559, 6)
```

In [12]:
```
movies.head()
```

Out[12]:

	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 [13]:
```
movies.tail()
```

Out[13]:

	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

now we will change the column names

In [15]:
```
movies.columns=['Film','Genre','Critic_Rating','Audience_Ratings','Budget_Millions'
```

In [16]:
```
movies.head(1)
```

Out[16]:

	Film	Genre	Critic_Rating	Audience_Ratings	Budget_Millions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009

In [17]:
```
movies.describe()
```

Out[17]:

	Critic_Rating	Audience_Ratings	Budget_Millions	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 [18]:
```
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   Critic_Rating     559 non-null    int64 
 3   Audience_Ratings  559 non-null    int64 
 4   Budget_Millions   559 non-null    int64 
 5   Year              559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [19]:
```
movies.Film= movies.Film.astype('category')
movies.Genre= movies.Genre.astype('category')
movies.Year= movies.Year.astype('category')
```

The film and genere are categorical so we changed to categorical but year does has meaning in describe so we change year to category

In [21]:
```
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   Critic_Rating     559 non-null    int64
 3   Audience_Ratings  559 non-null    int64
 4   Budget_Millions   559 non-null    int64
 5   Year              559 non-null    category
dtypes: category(3), int64(3)
memory usage: 36.5 KB
```

In [22]:
```
movies.Film
```

Out[22]:
```
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 [23]:
```
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [24]:
```
import warnings
warnings.filterwarnings('ignore')
```

---

jointplot()

Jointplot() is a plotting function that creates a multi-panel figure showing the relationship between two variables as well as their individual distributions.

"Jointplot lets you see how two variables relate to each other and also how each one is distributed—all in one figure."

`It combines:`

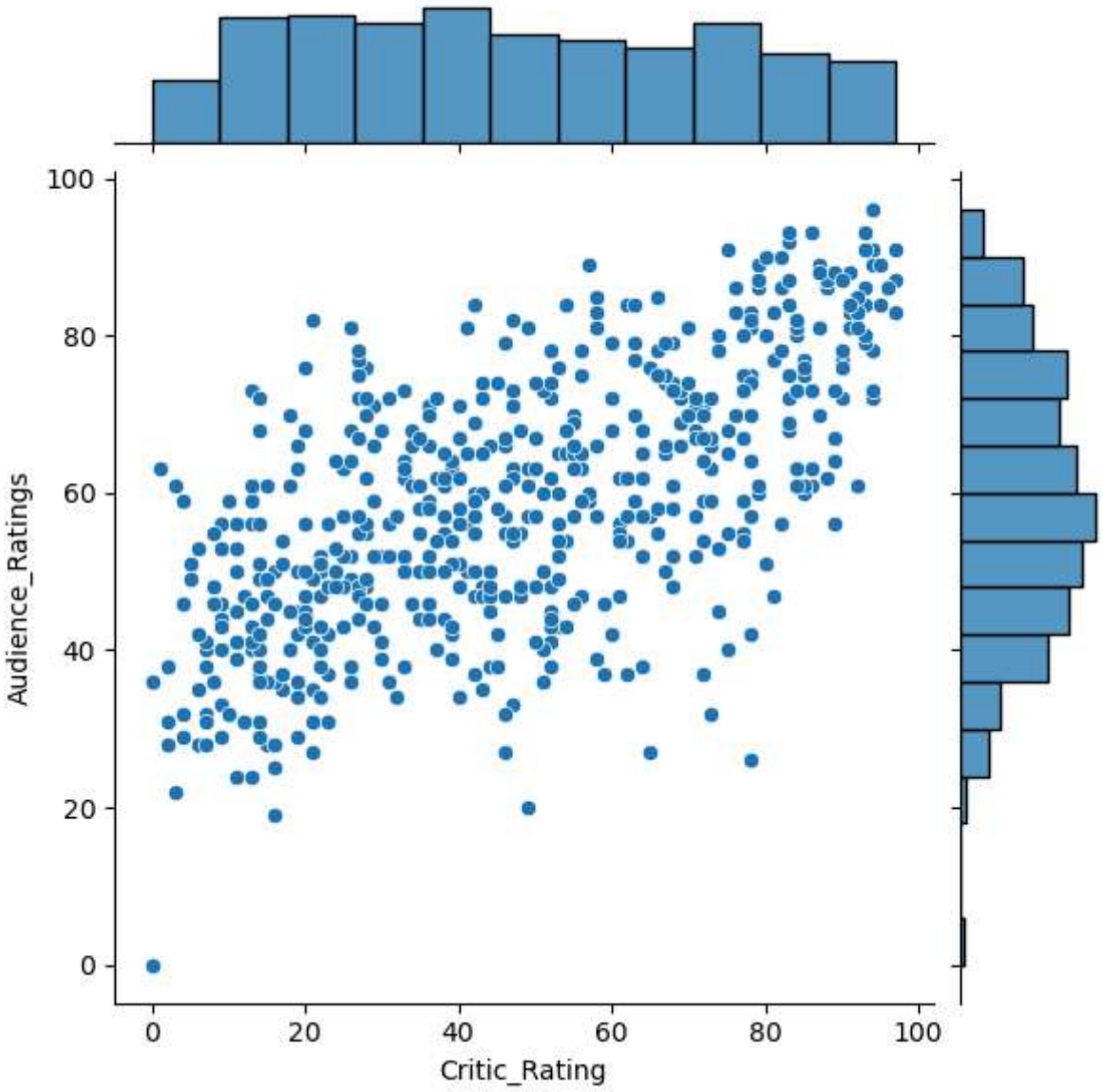
1. A scatter plot (or other bivariate plot) in the center,

2. Histograms or density plots for each variable on the top and right margins.

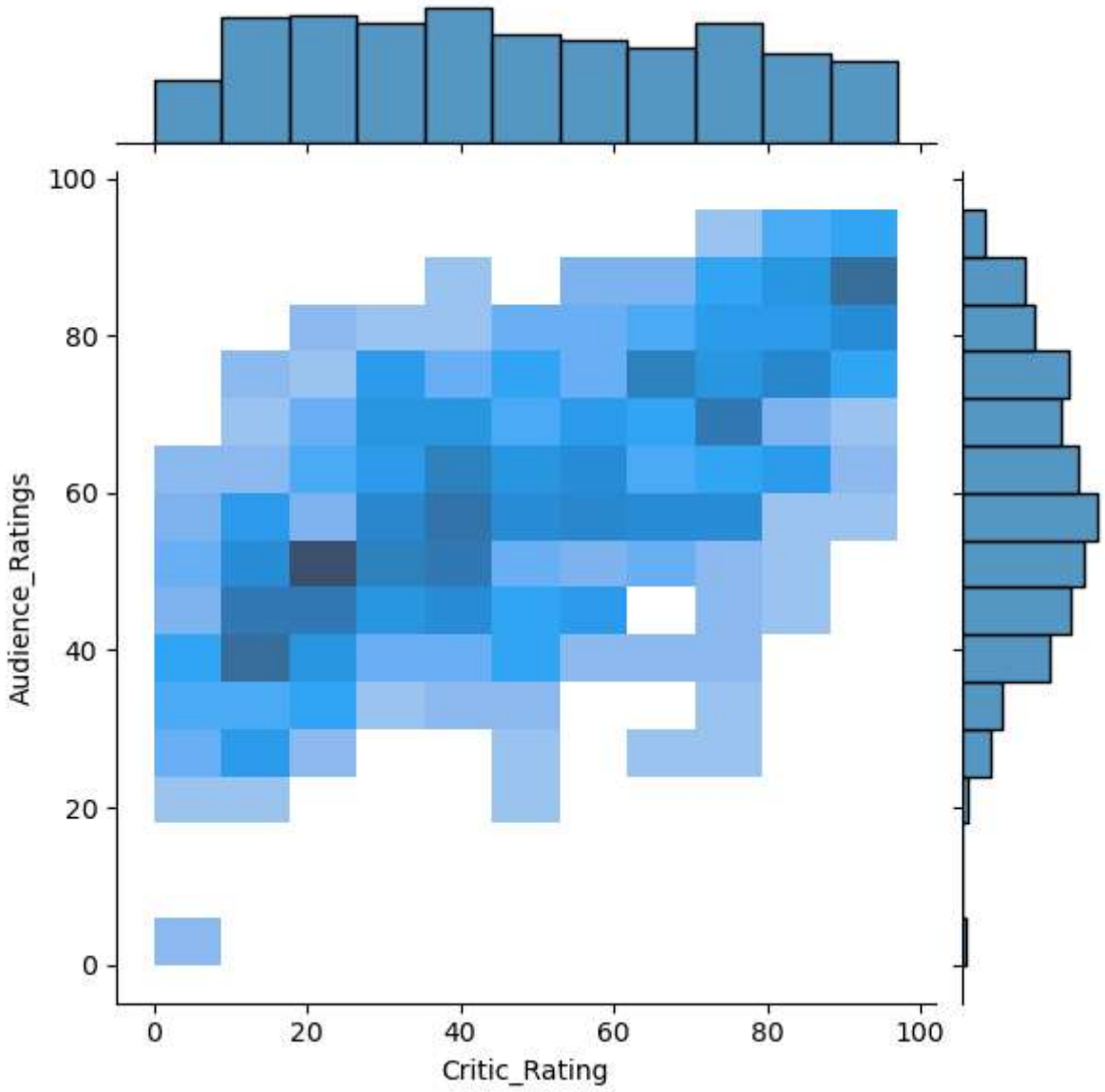
3. kind are kind : { "scatter" | "kde" | "hist" | "hex" | "reg" | "resid" }

Parameter	Description
`x`, `y`	Variables to plot
`data`	DataFrame containing x and y
`kind`	Type of joint plot:
`'scatter'` (default), `'kde'`, `'hist'`, `'hex'`, `'reg'`, `'resid'`	
`hue`	Grouping by category (from Seaborn v0.11+)
`height`	Size of the figure (in inches)
`ratio`	Ratio of joint axes size to marginal axes
`space`	Space between joint and marginal plots
`palette`	Color palette when `hue` is used
`marginal_kws`	Customize marginal plots (e.g., bins, color)
`joint_kws`	Customize joint plot style (e.g., alpha, marker)

In [28]:
```
j=sns.jointplot(data=movies,x='Critic_Rating',y='Audience_Ratings',kind='scatter')
```



In [29]:
```
j1=sns.jointplot(data=movies,x='Critic_Rating',y='Audience_Ratings',kind='hist')
```



Histogram

distplot() or histplot()

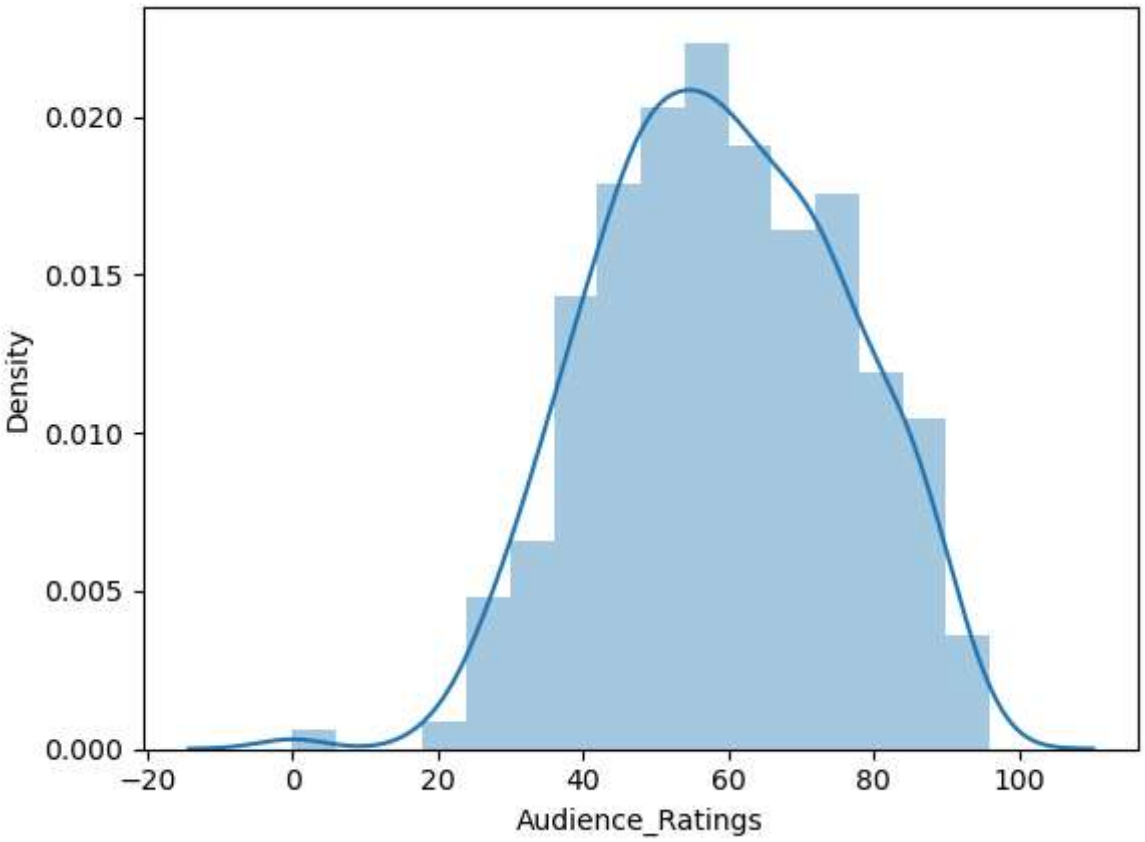
distplot() was a function used to plot univariate distributions—showing a histogram along with an optional Kernel Density Estimate (KDE) curve

"distplot was used to visualize the distribution of a single variable using a histogram and a smooth density curve."

Parameter	Description
a	Data to plot

Parameter	Description
`bins`	Number of bins
`kde`	Show KDE curve
`rug`	Show individual values as dashes
`hist`	Show histogram (True/False)
`color`	Color of plot
`ax`	Matplotlib Axes to draw on

In [34]:
```
m1=sns.distplot(movies.Audience_Ratings)
```



---

displot

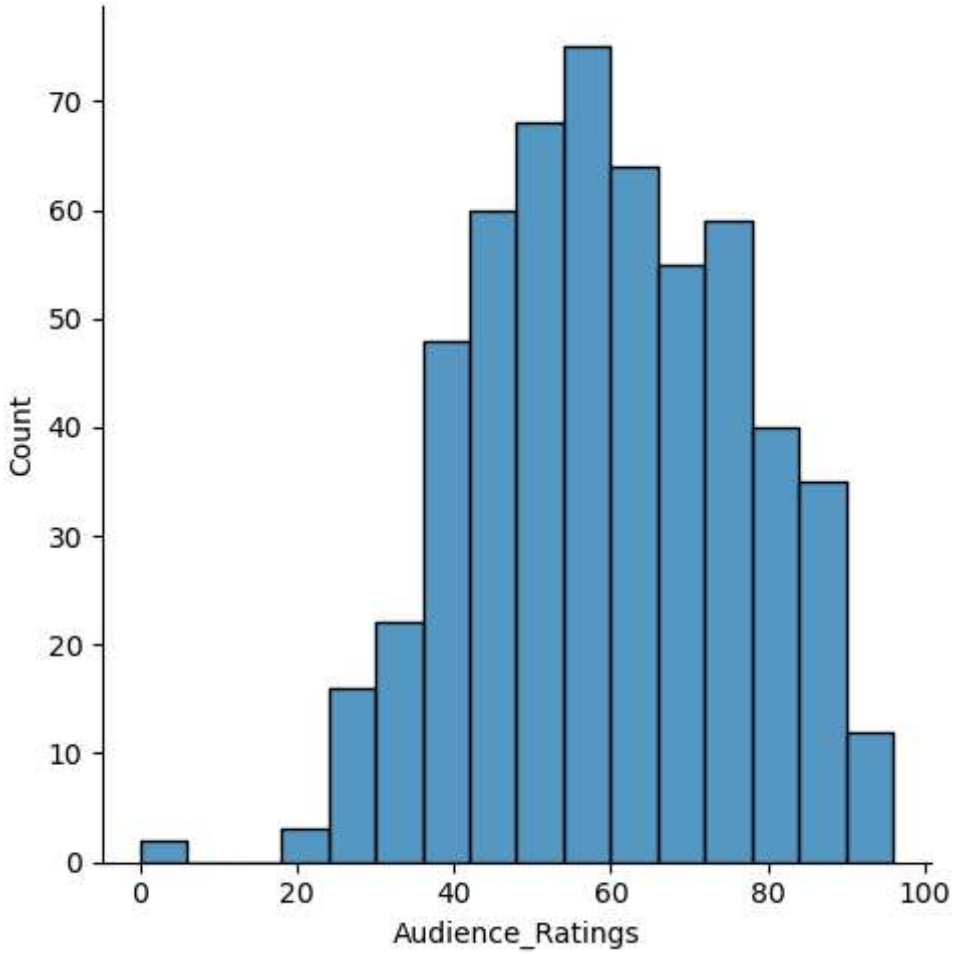
displot() is a flexible function used to visualize the distribution of a dataset. It creates histograms, KDEs (Kernel Density Estimates), or ECDFs (Empirical Cumulative Distribution Functions) using a Figure-level interface.

displot() is the modern replacement for the deprecated distplot().

"displot shows how a single variable is distributed—like how values are spread out—using a histogram or density curve."

Parameter	Description
`data`	DataFrame
`x` , `y`	Variables to plot
`hue`	Color by category
`kind`	`'hist'` , `'kde'` , `'ecdf'`
`bins` , `binwidth` , `discrete`	Same as in `histplot()`
`col` , `row`	Create subplots in grid format
`col_wrap`	Max columns before wrapping rows
`height`	Height (in inches) of each subplot
`aspect`	Width = height × aspect
`kde`	Add KDE (only if `kind='hist'`)
`palette`	Color theme for categories
`facet_kws`	Extra options for grid (like spacing)

In [37]:
```
m2=sns.displot(movies.Audience_Ratings)
```

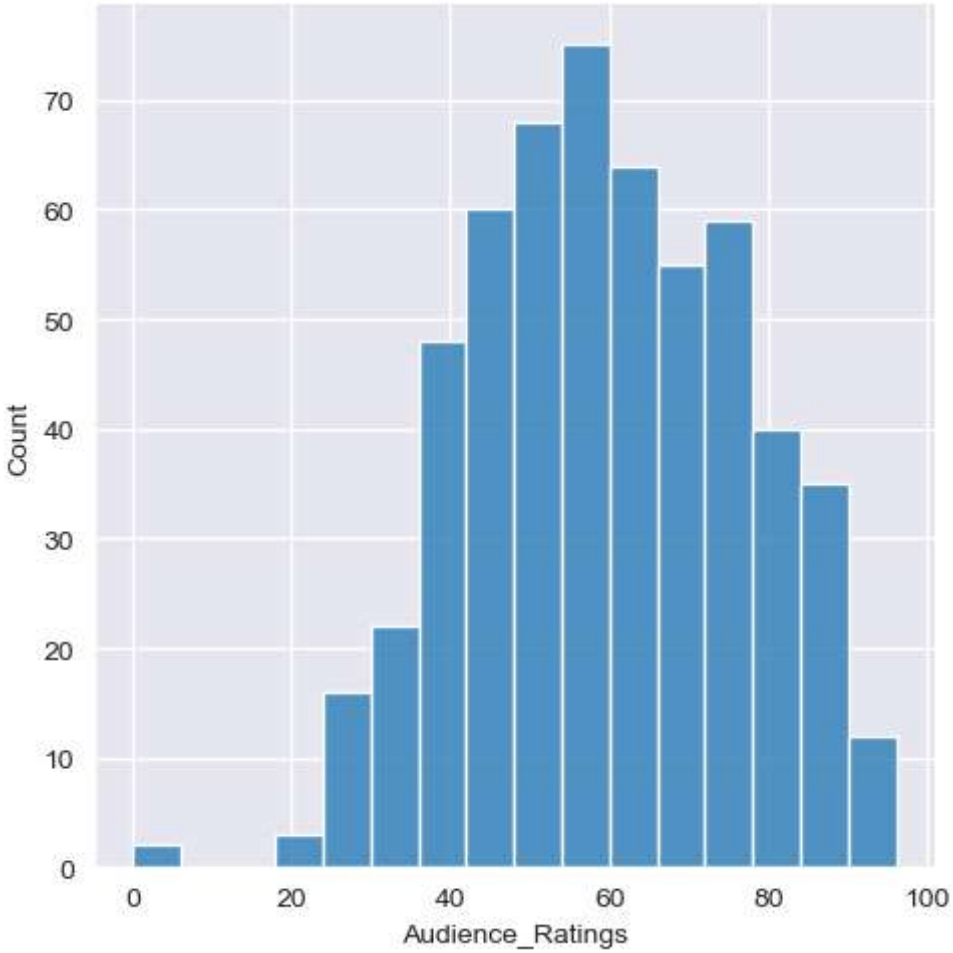


We can change the background of the graph

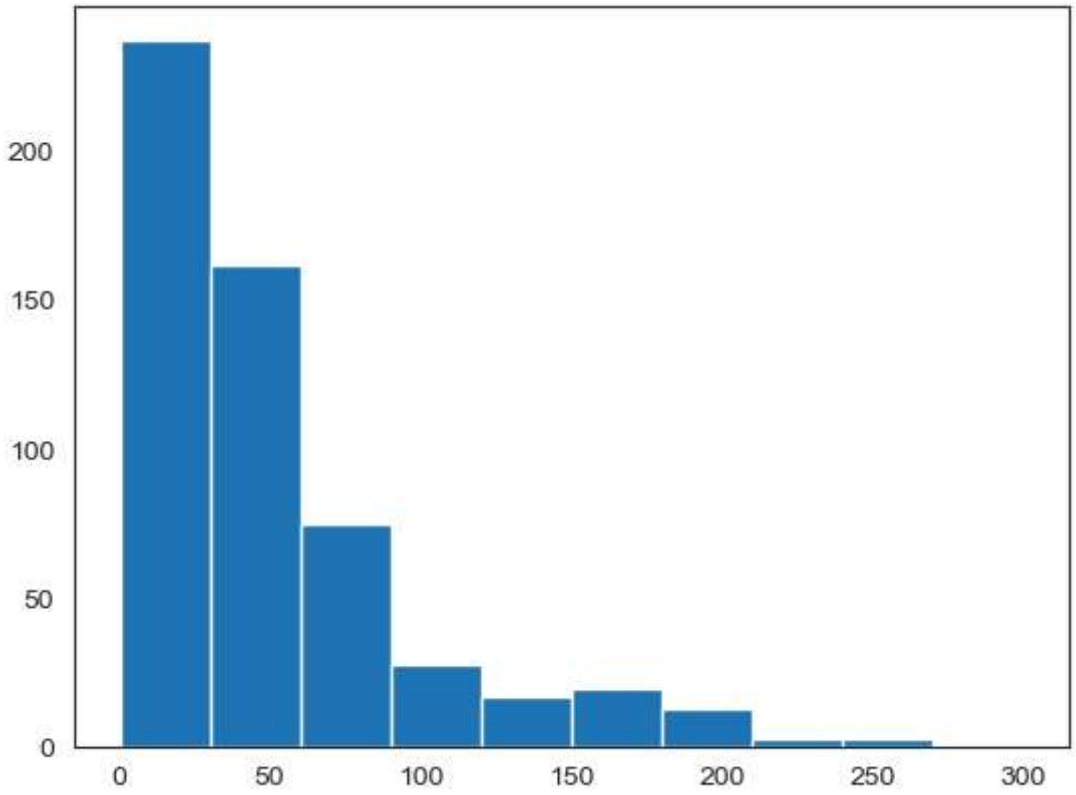
styles are {darkgrid, whitegrid, dark, white, ticks}

In [39]:
```
sns.set_style("darkgrid")
```

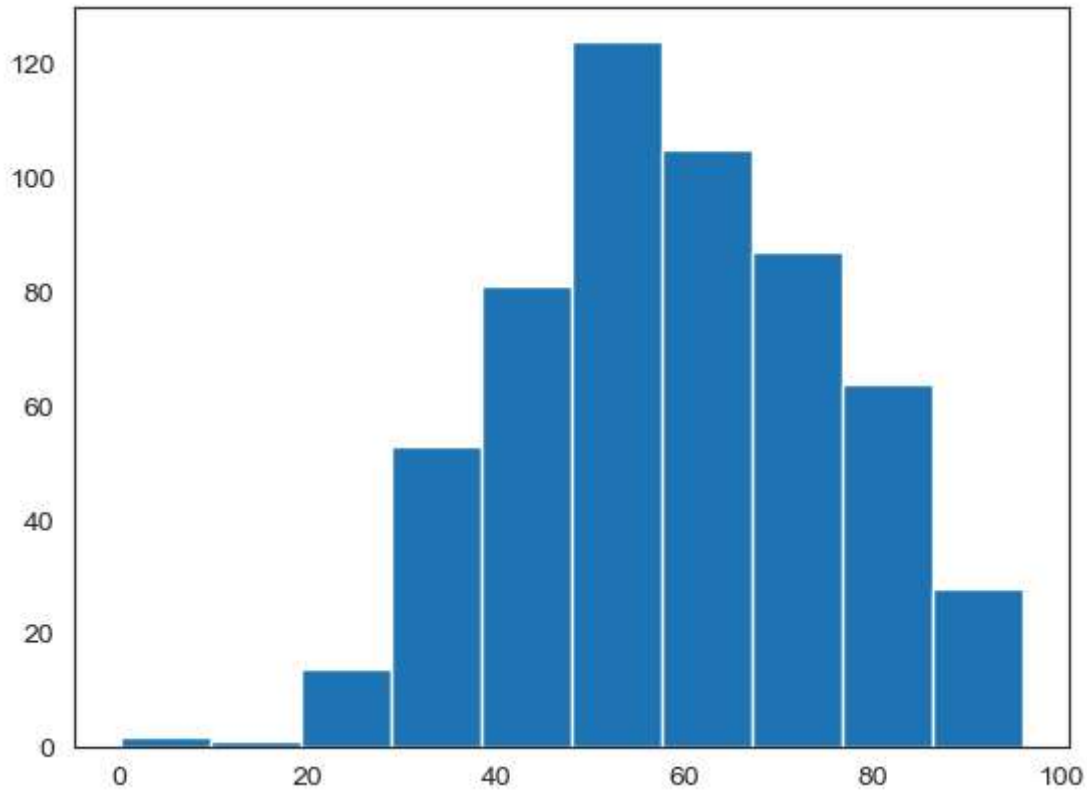
In [40]:
```
m2=sns.displot(movies.Audience_Ratings)
```



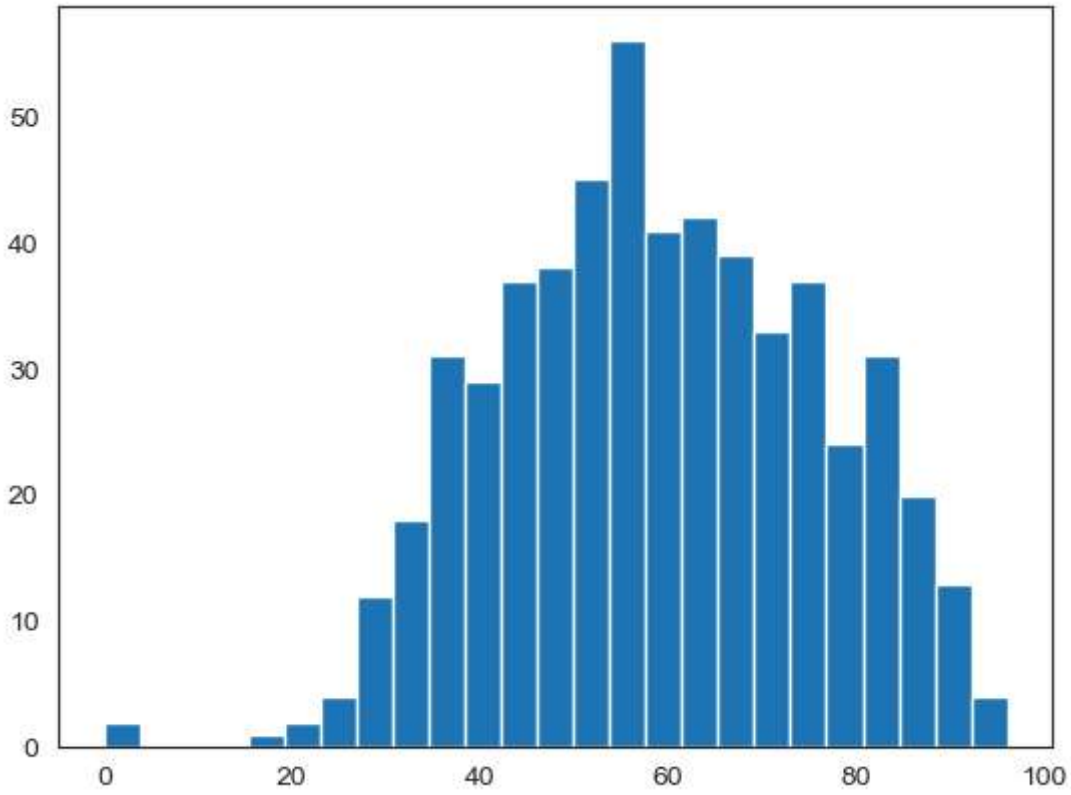
In [41]:
```
sns.set_style("white")
a=plt.hist(movies.Budget_Millions)
a=plt.show()
```



In [42]:
```
n1 = plt.hist(movies.Audience_Ratings)
```



In [43]:
```
n1 = plt.hist(movies.Audience_Ratings,bins=25)
```

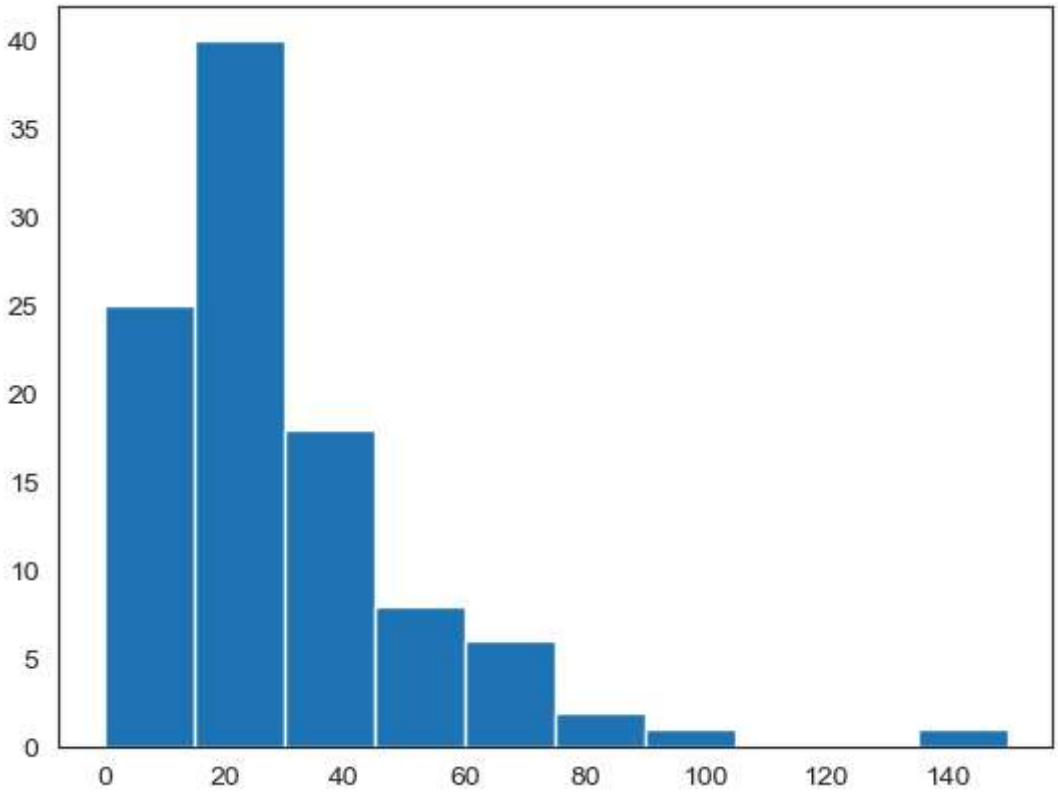


It selects all the Drama movies from the movies dataset.

Then it takes their budgets (in millions).

It makes a histogram — a kind of bar chart that shows how many Drama movies have budgets in different ranges.

In [45]:
```
plt.hist(movies[movies.Genre == 'Drama'].Budget_Millions)
plt.show()
```



First line: Draws a histogram of Action movie budgets using 20 bins (i.e., divides the budget into 20 ranges).

Second line: Adds another histogram for Thriller movie budgets, also with 20 bins — it gets drawn on the same plot.

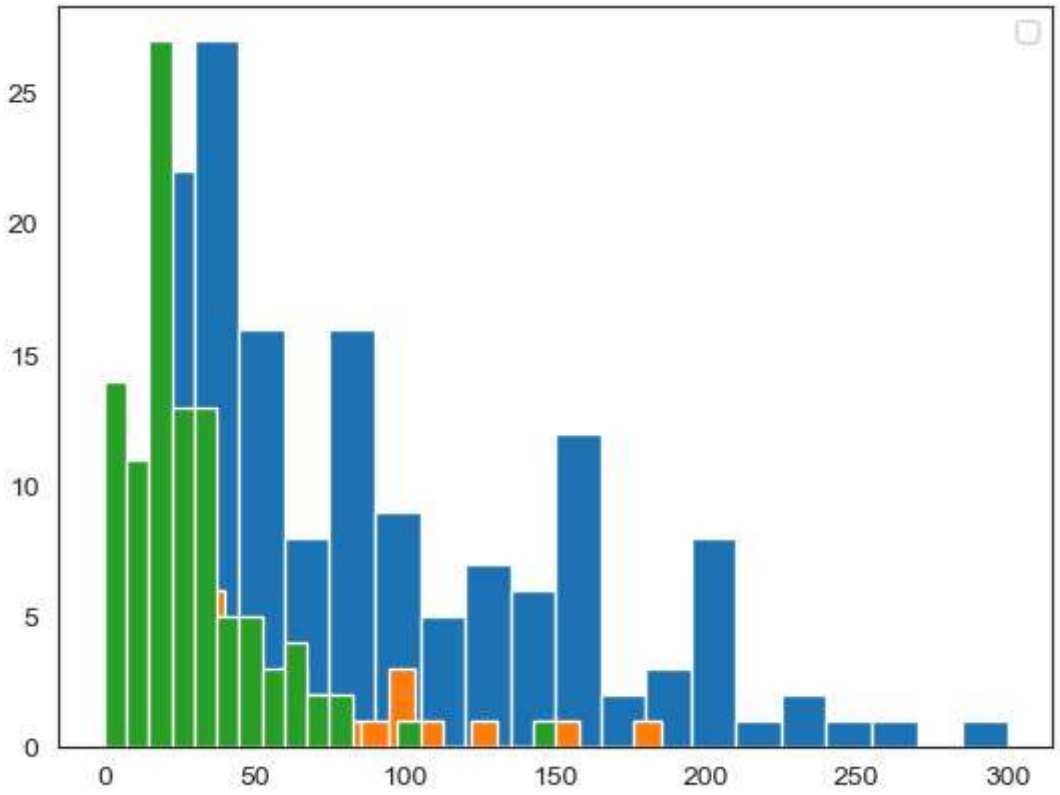
Third line: Adds a Drama movie budget histogram, again with 20 bins, on the same plot.

Displays the final plot with all 3 histograms overlapping on one chart.

This code compares the budget distributions of Action, Thriller, and Drama movies by showing 3 overlapping histograms in one graph.

In [47]:
```
plt.hist(movies[movies.Genre == 'Action'].Budget_Millions, bins = 20)
plt.hist(movies[movies.Genre == 'Thriller'].Budget_Millions, bins = 20)
plt.hist(movies[movies.Genre == 'Drama'].Budget_Millions, bins = 20)
plt.legend()
plt.show()
```

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



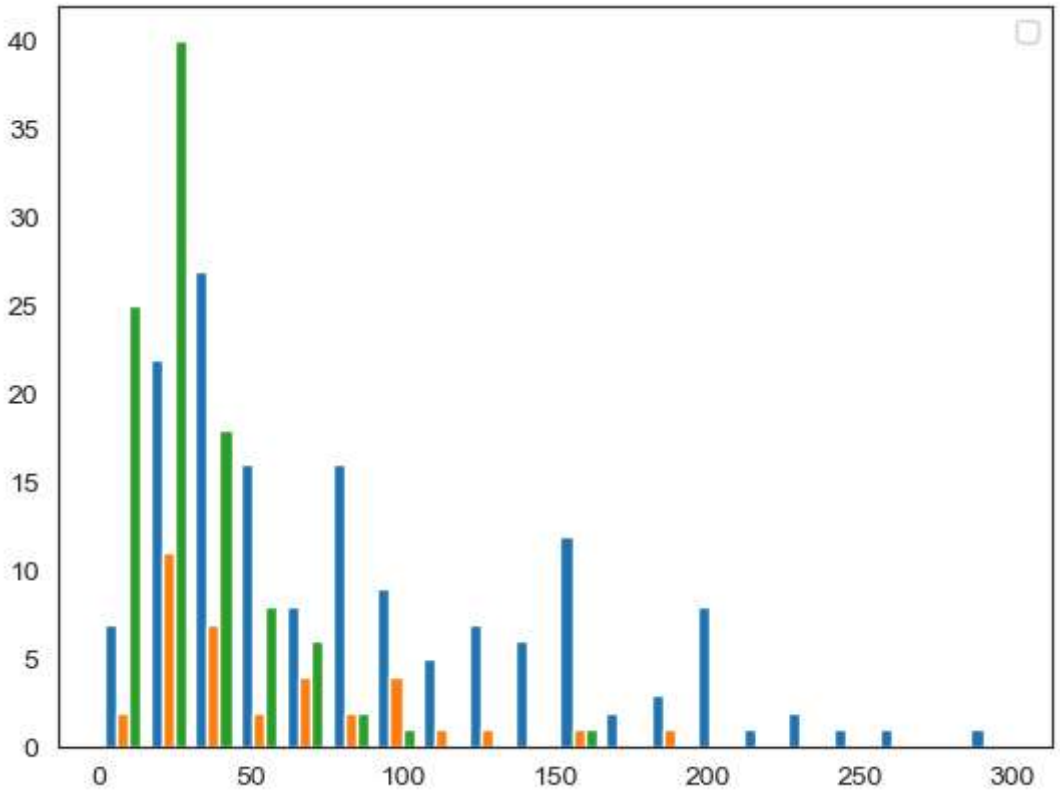
You pass a list of 3 datasets (Action, Thriller, Drama movie budgets).

It draws 3 histograms in one plot, stacked together by default.

Each histogram divides the budget range into 20 buckets.

In [49]:
```
plt.hist([movies[movies.Genre == 'Action'].Budget_Millions,\
        movies[movies.Genre == 'Thriller'].Budget_Millions,\
        movies[movies.Genre == 'Drama'].Budget_Millions], bins = 20)
plt.legend()
plt.show()
```

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



Type	Looks Like	Use When
3 graphs in one	Bars of all genres overlap	You want to **compare shapes**
Stacked	Bars are built on top of each other	You want to **compare totals** and **see each genre's contribution**

---

lmplot()- linear model plot

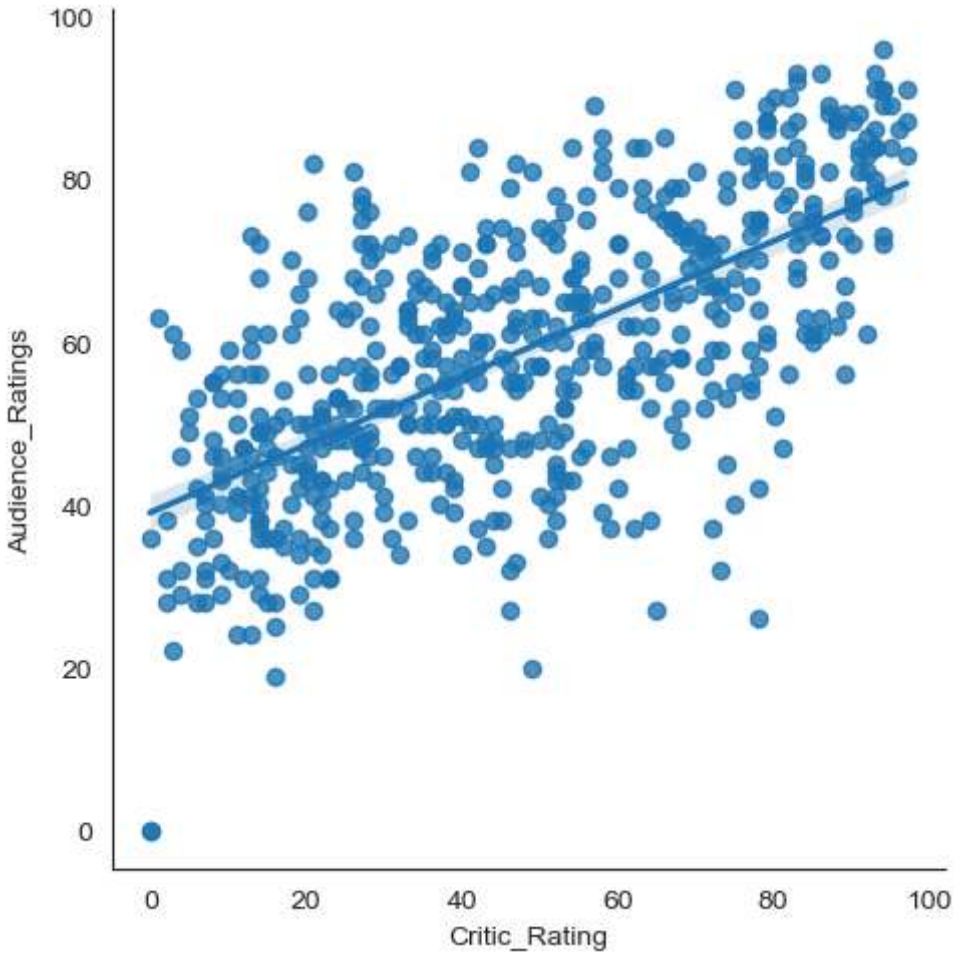
lmplot() is a function used to plot scatterplots with a linear regression line. It shows the relationship between two variables and fits a linear model (a best-fit line) to the data.

"lmplot helps you see how two variables relate to each other and shows a straight line that best fits the data points."

Parameter	Description
`x`, `y`	Names of variables for the x- and y-axis
`data`	DataFrame containing the data
`hue`	Color groups by category (adds different regression lines)
`col`, `row`	Creates subplots by column or row
`markers`	Marker style for scatter points

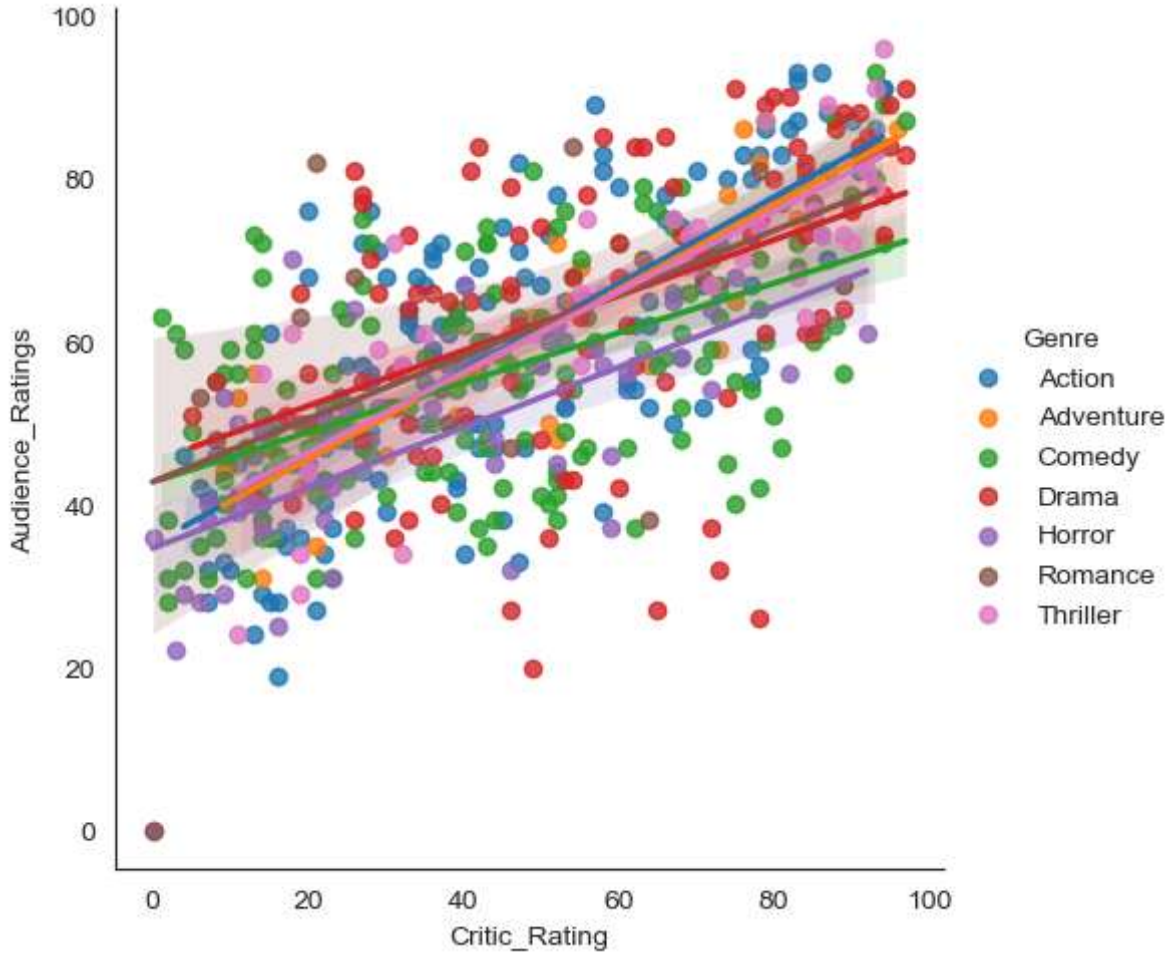
Parameter	Description
`palette`	Color palette
`height`	Height (in inches) of each subplot
`aspect`	Aspect ratio (width = height × aspect)

In [53]:
```
vis=sns.lmplot(data=movies,x='Critic_Rating',y='Audience_Ratings',fit_reg=True)
```

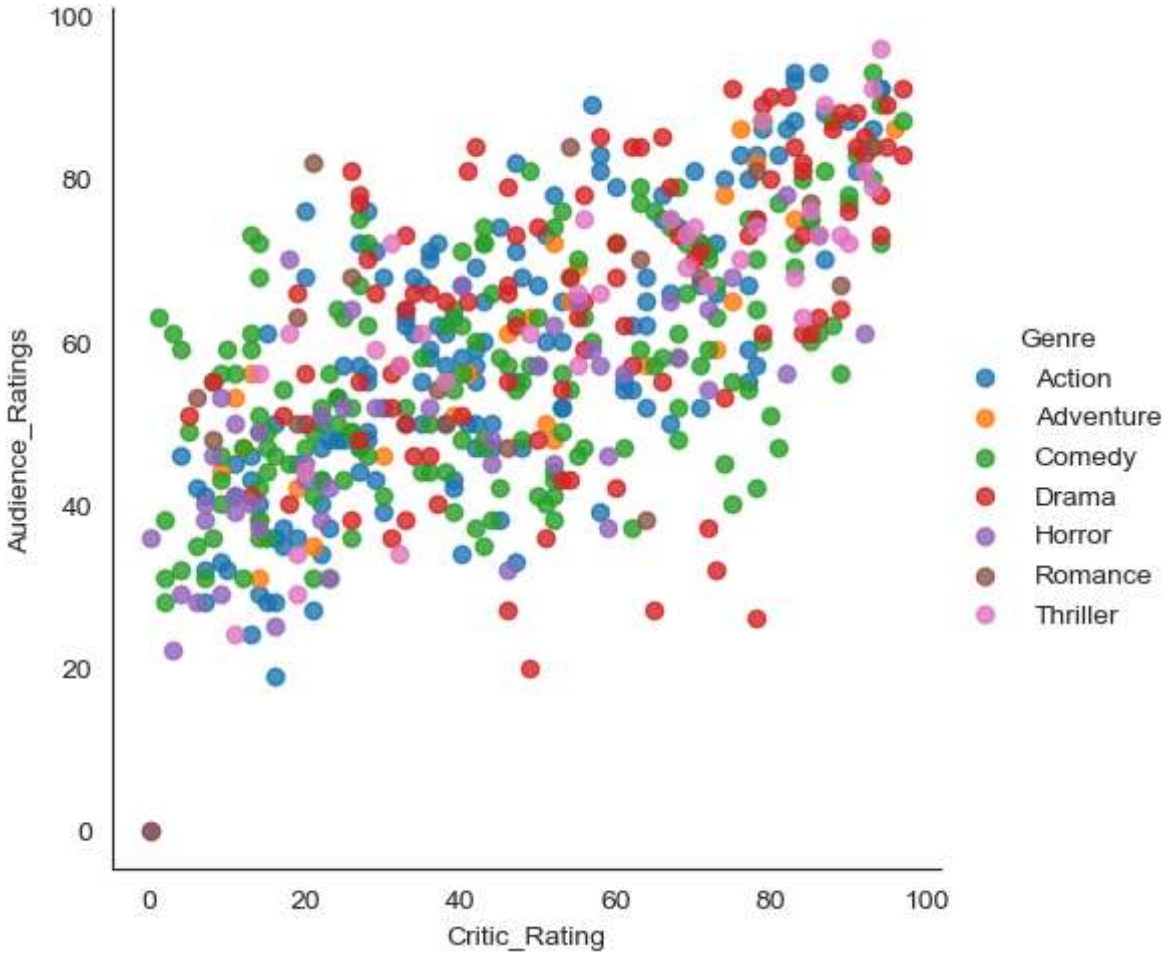


Genre is the categories of movies like action, drama, Comedy.

In [55]:
```
vis1=sns.lmplot(data=movies,x='Critic_Rating',y='Audience_Ratings',fit_reg=True, hu
```



In [56]:
```
vis2=sns.lmplot(data=movies,x='Critic_Rating',y='Audience_Ratings',fit_reg=False, h
```



---

boxplot()

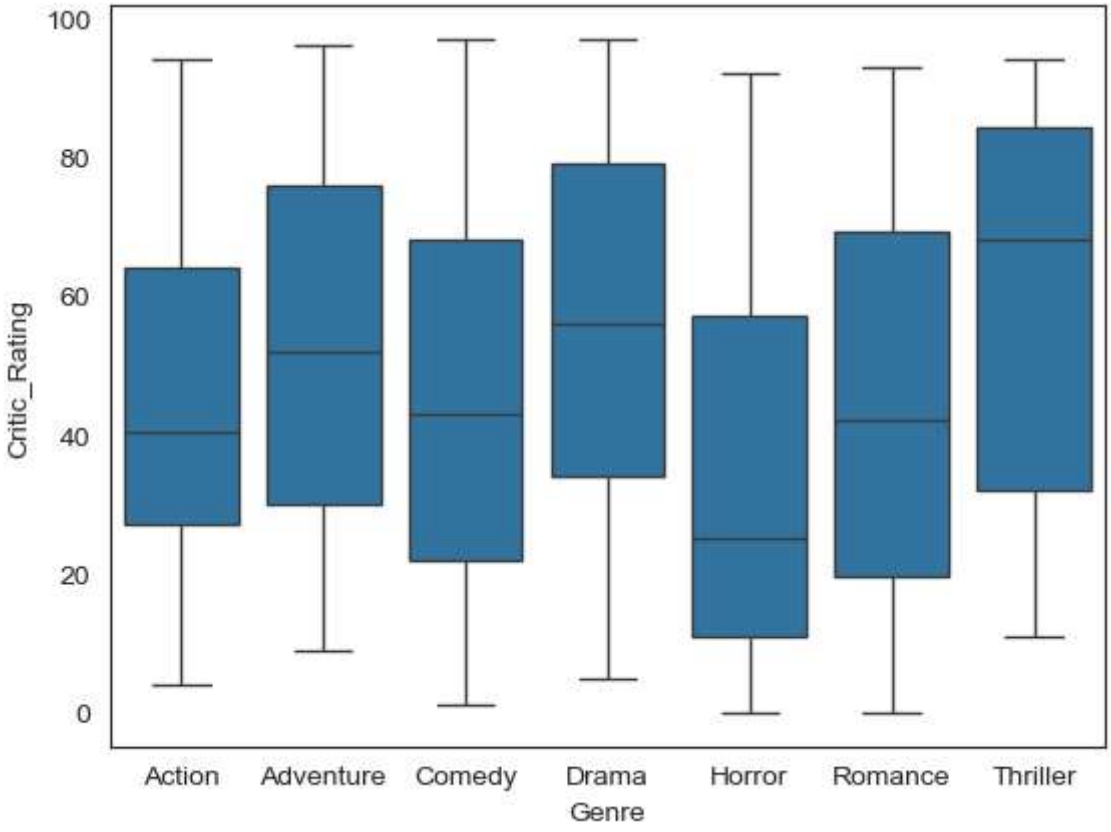
boxplot() is a function used to visualize the distribution of data through their quartiles and to identify outliers.

"boxplot shows the spread of data, highlighting the middle value, the range where most data lie, and any extreme points (outliers)."

Parameter	Description
`x`	Name of the column to use on the x-axis (can be categorical)
`y`	Name of the column to use on the y-axis (numeric)
`data`	DataFrame from which x and y are taken
`hue`	Grouping variable — creates side-by-side boxplots
`order`	Custom order of categories along the axis
`orient`	`'v'` (vertical, default) or `'h'` (horizontal)
`color`	Single color for all boxes

Parameter	Description
`palette`	Color palette for different groups
`width`	Width of each box (default is 0.8)
`linewidth`	Width of the lines (box edges, whiskers)
`fliersize`	Size of outlier markers
`saturation`	Opacity of box colors (0 to 1)
`dodge`	Whether to separate boxes by hue (default `True`)
`ax`	Matplotlib Axes to draw on (optional)

In [59]:
```
w = sns.boxplot(data=movies, x='Genre', y = 'Critic_Rating')
```



---

facetgrid()

FacetGrid is a figure-level object that lets you create a grid of plots based on the values of one or more categorical variables. It's great for visualizing data subsets side-by-side.

"FacetGrid helps you split your data by categories and create multiple smaller plots — arranged in rows and columns — to compare different groups easily."

Parameter	Description
`data`	DataFrame to plot
`row`	Variable to create grid rows
`col`	Variable to create grid columns
`hue`	Variable to color within each plot
`height`	Height (in inches) of each facet
`aspect`	Aspect ratio of each facet (width / height)

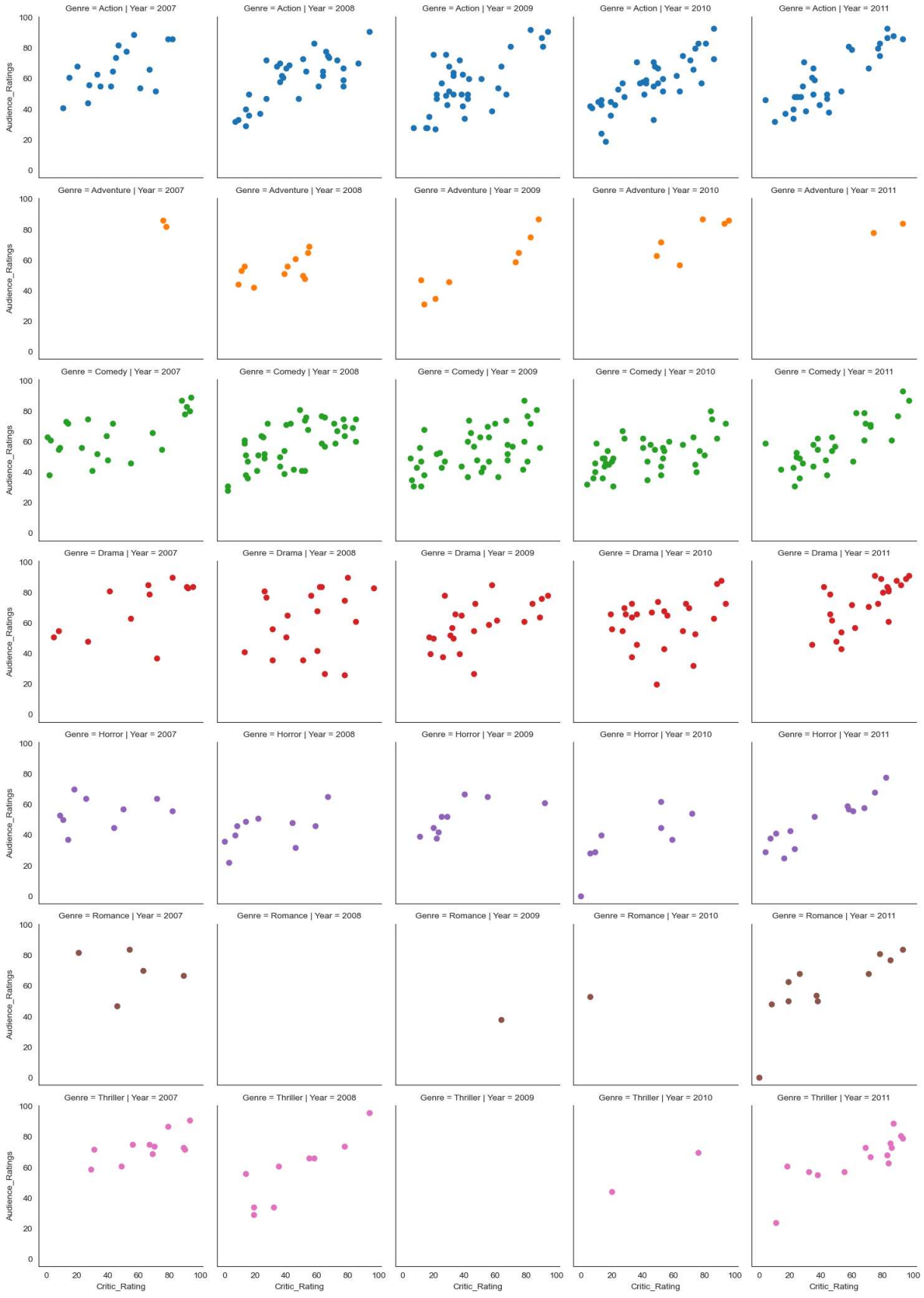
In [128…
```
movies.Genre.unique()
```

Out[128…
```
['Comedy', 'Adventure', 'Action', 'Horror', 'Drama', 'Romance', 'Thriller']
Categories (7, object): ['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Roma
nce', 'Thriller']
```

In [63]:
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre') #kind of subp
```

In [131…

```
g1=sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g1= g1.map(plt.scatter, 'Critic_Rating', 'Audience_Ratings' ) #scatterplots are map
```

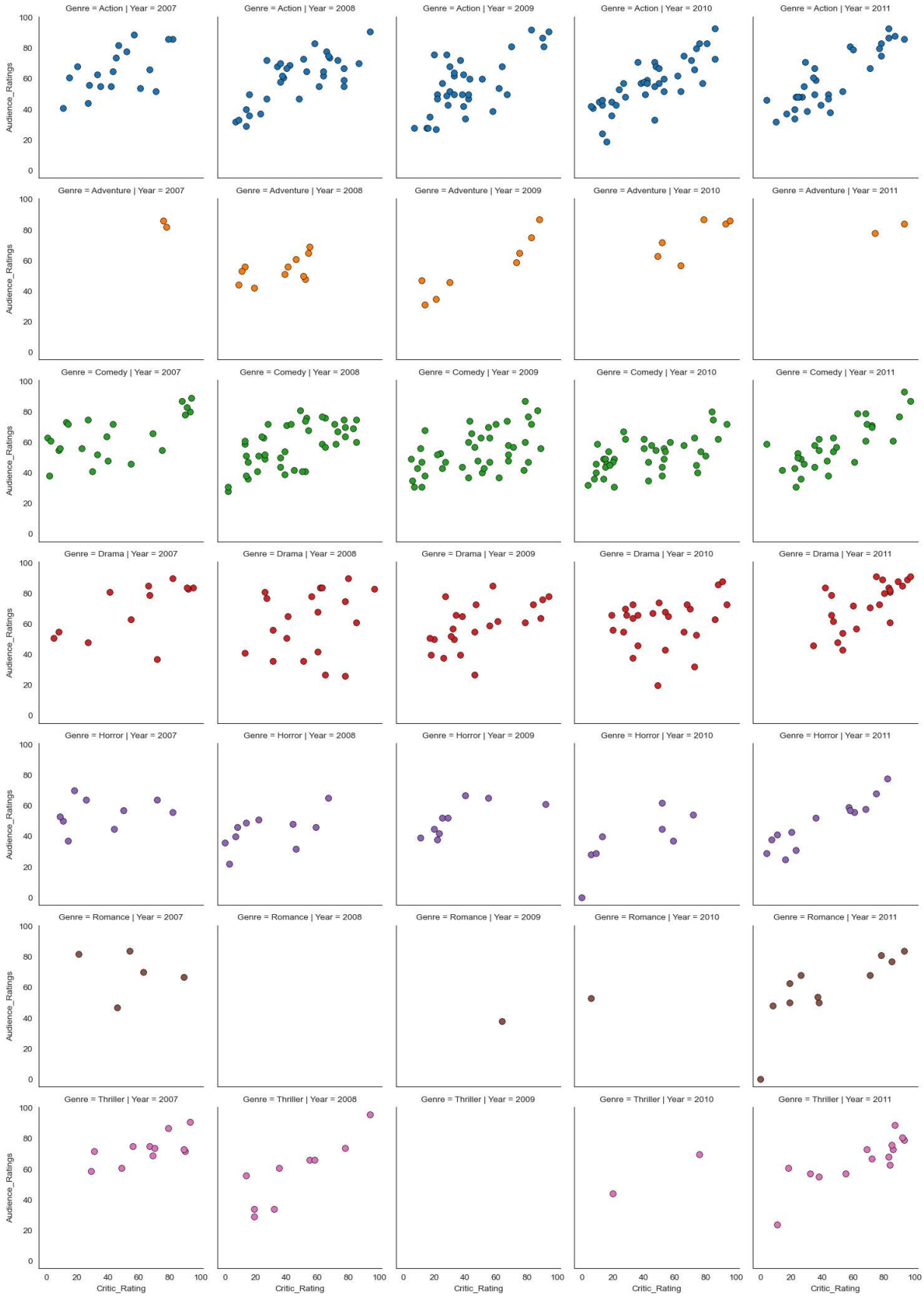


In [135…

```
g2=sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g2= g2.map(plt.hist, 'Budget_Millions') #scatterplots are mapped in facetgrid
```

In [137…

```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
kws = dict(s=50, linewidth=0.5,edgecolor='black')
g = g.map(plt.scatter, 'Critic_Rating', 'Audience_Ratings',**kws ) #scatterplots ar
```


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
Audience_Ratings
Critic_Rating
0
20
40
60
80
100

---

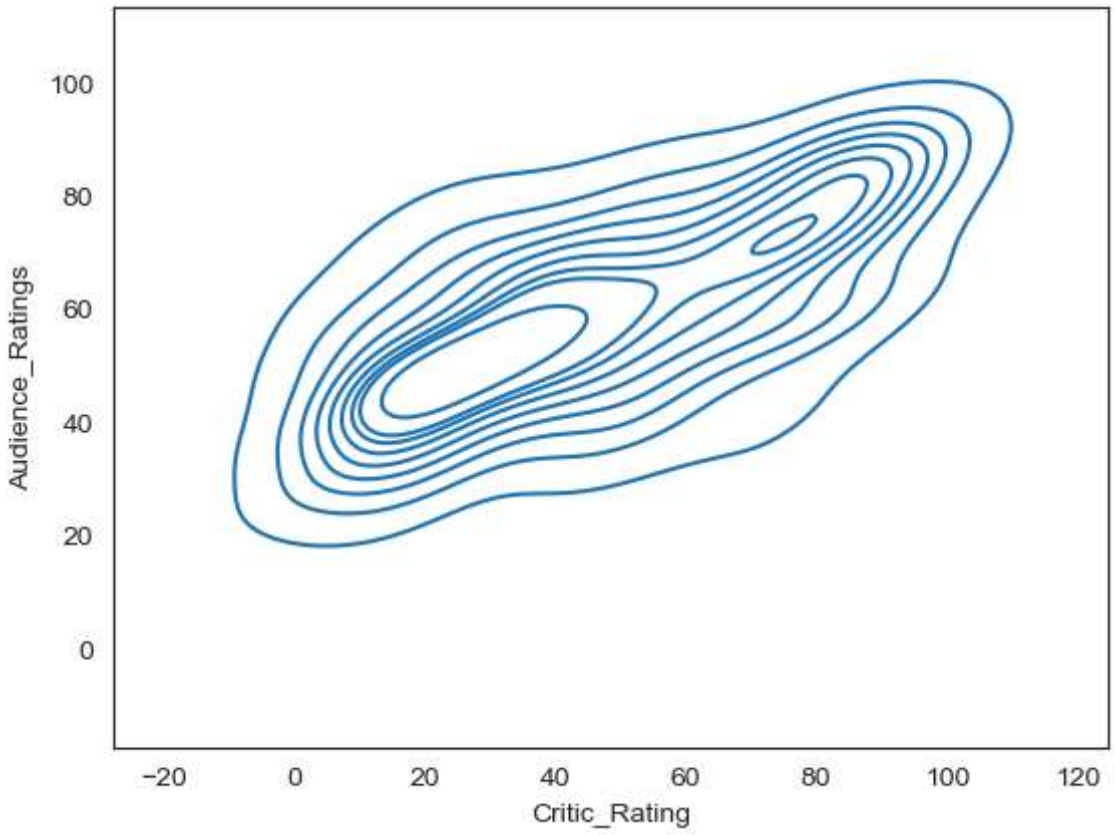
kdeplot()

kdeplot() creates a Kernel Density Estimate (KDE) plot, which is a smooth curve (or surface) that shows the probability density of a continuous variable or variables.

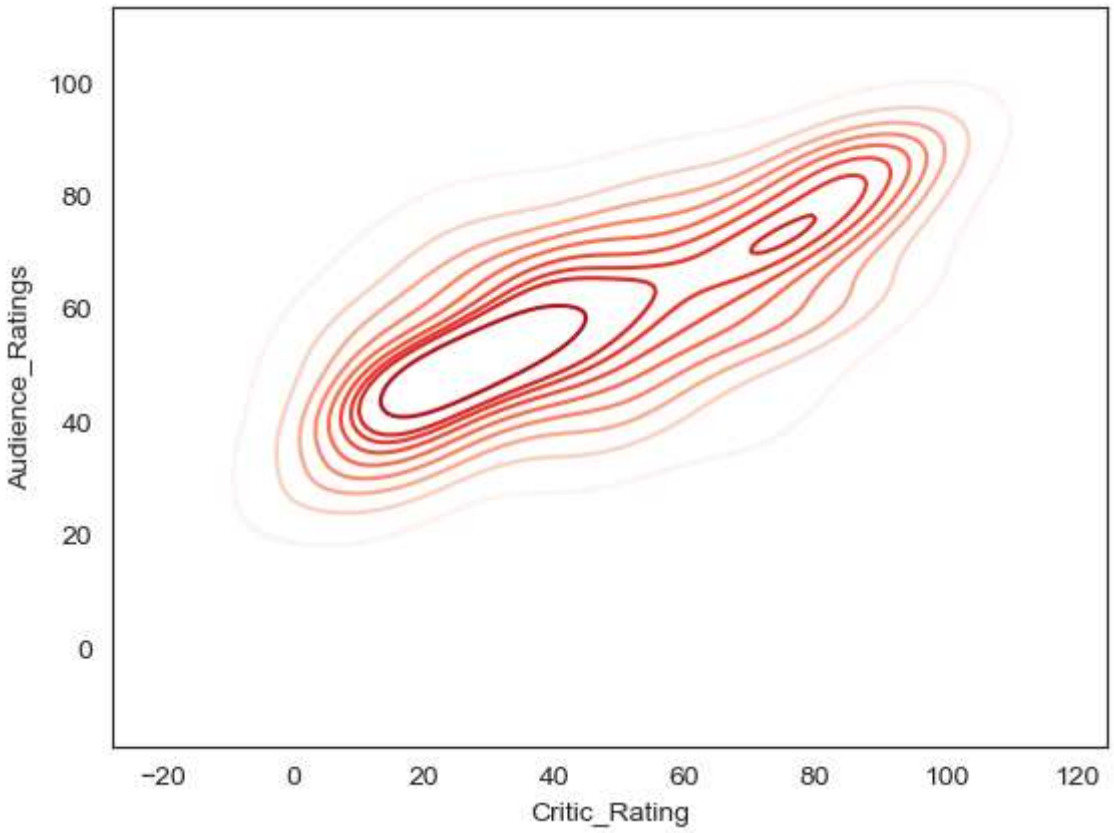
"kdeplot helps you visualize how data points are spread out smoothly, showing where data is more or less concentrated, instead of just raw counts like histograms."

Parameter	Description
`x`, `y`	Variables to plot (1D or 2D)
`shade` or `fill`	Fill the area under the curve
`bw_adjust`	Bandwidth adjustment (controls smoothness)
`cmap`	Colormap for 2D plots
`levels`	Number of contour levels

In [158…
```
k1 = sns.kdeplot(x=movies.Critic_Rating, y=movies.Audience_Ratings)
```

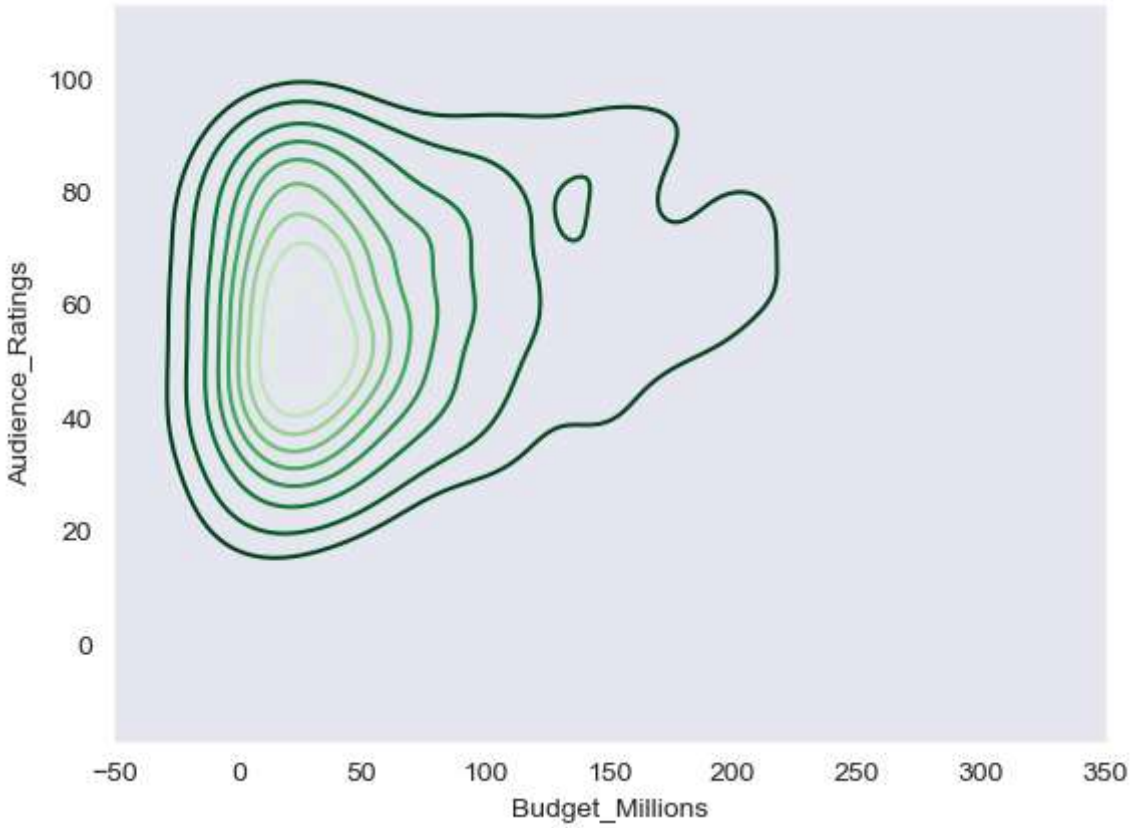


In [169…
```
k2 = sns.kdeplot(x=movies.Critic_Rating, y=movies.Audience_Ratings,cmap='Reds')
```



In [179…

```
sns.set_style('dark')
k3 = sns.kdeplot(x= movies.Budget_Millions, y= movies.Audience_Ratings,cmap='Greens
```

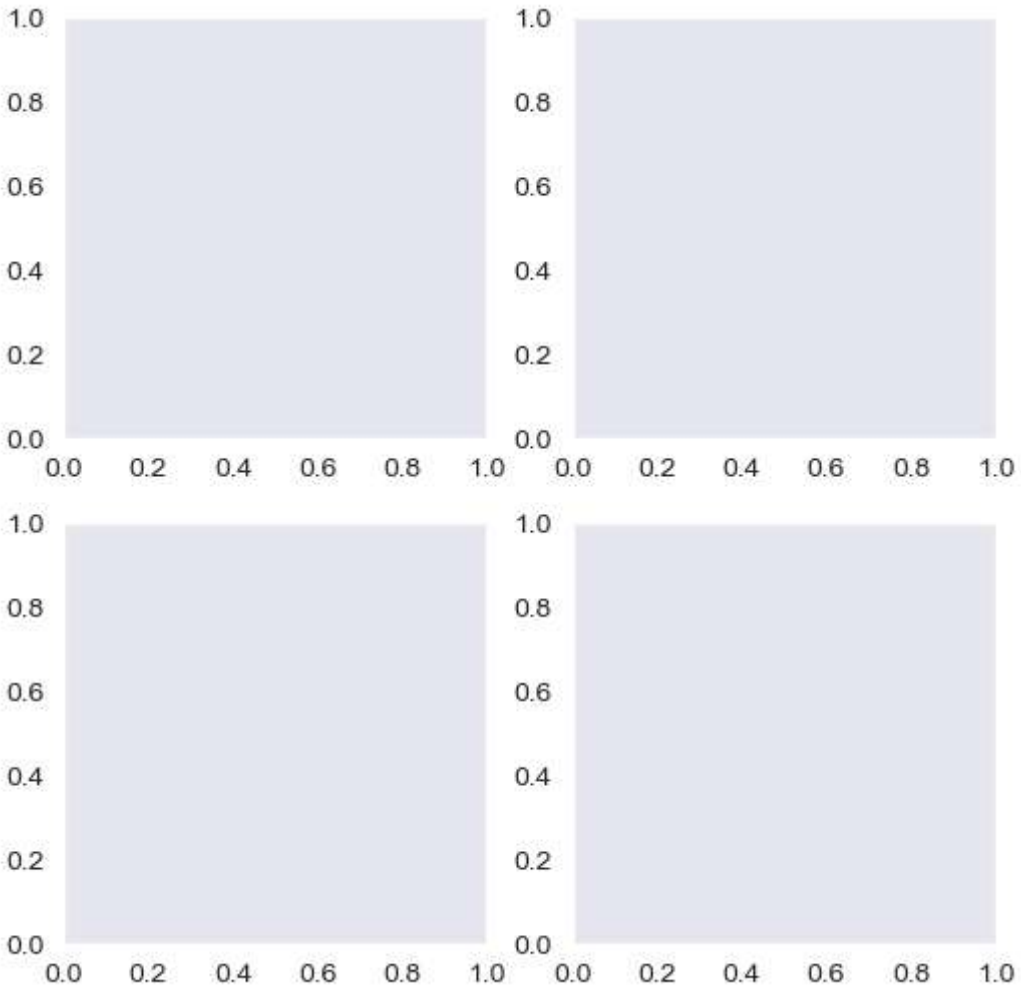


subplots

Subplots allow you to create multiple plots (axes) within a single figure — arranged in a grid of rows and columns. This helps compare multiple visualizations side-by-side in one window.

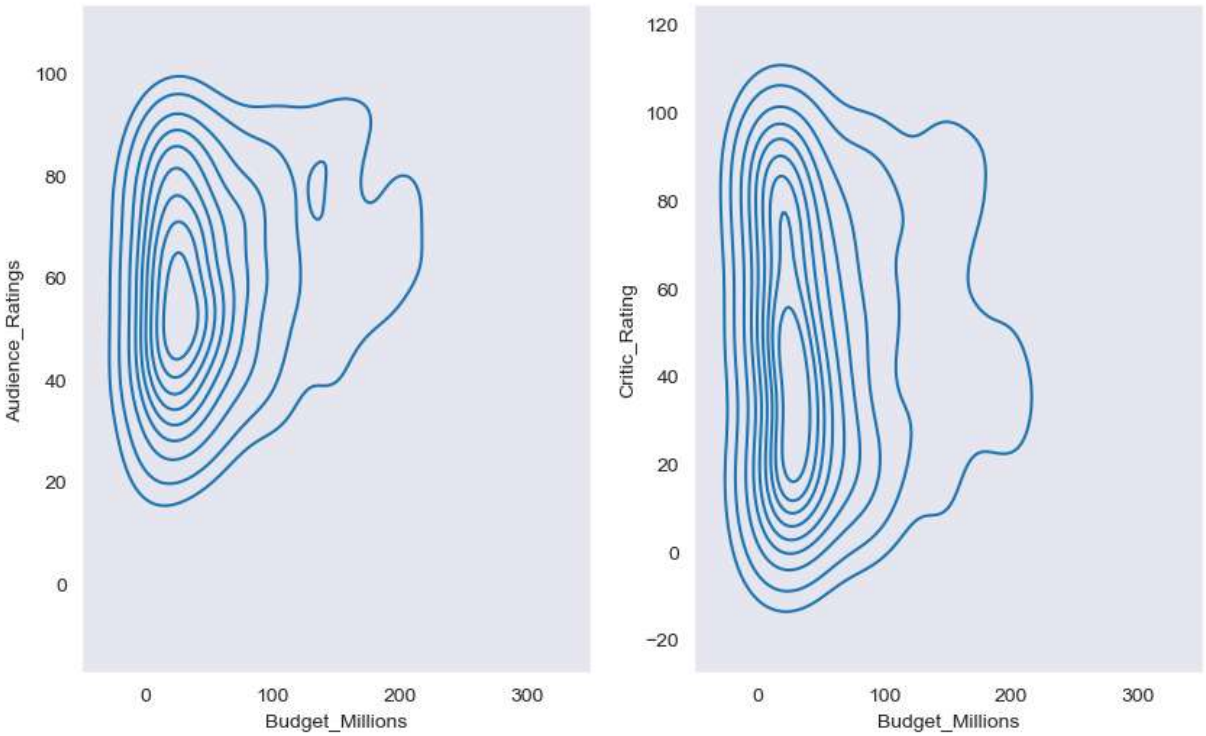
"Subplots are like having multiple smaller plots on one page, so you can view several charts together."

In [187…
```
f,ax = plt.subplots(2,2, figsize =(6,6))
```



In [196…
```
f, axes = plt.subplots(1,2, figsize =(10,6))

k1 = sns.kdeplot(x=movies.Budget_Millions,y=movies.Audience_Ratings,ax=axes[0])
k2 = sns.kdeplot(x=movies.Budget_Millions,y=movies.Critic_Rating,ax = axes[1])
```


Audience_Ratings
100
80
60
40
20
0
0
100
200
300
Budget_Millions
Critic_Rating
120
100
80
60
40
20
0
−20
0
100
200
300
Budget_Millions