NAME:-AKSHAY ASHOK GHADGE

PRN No:-ADT24MGTM0909

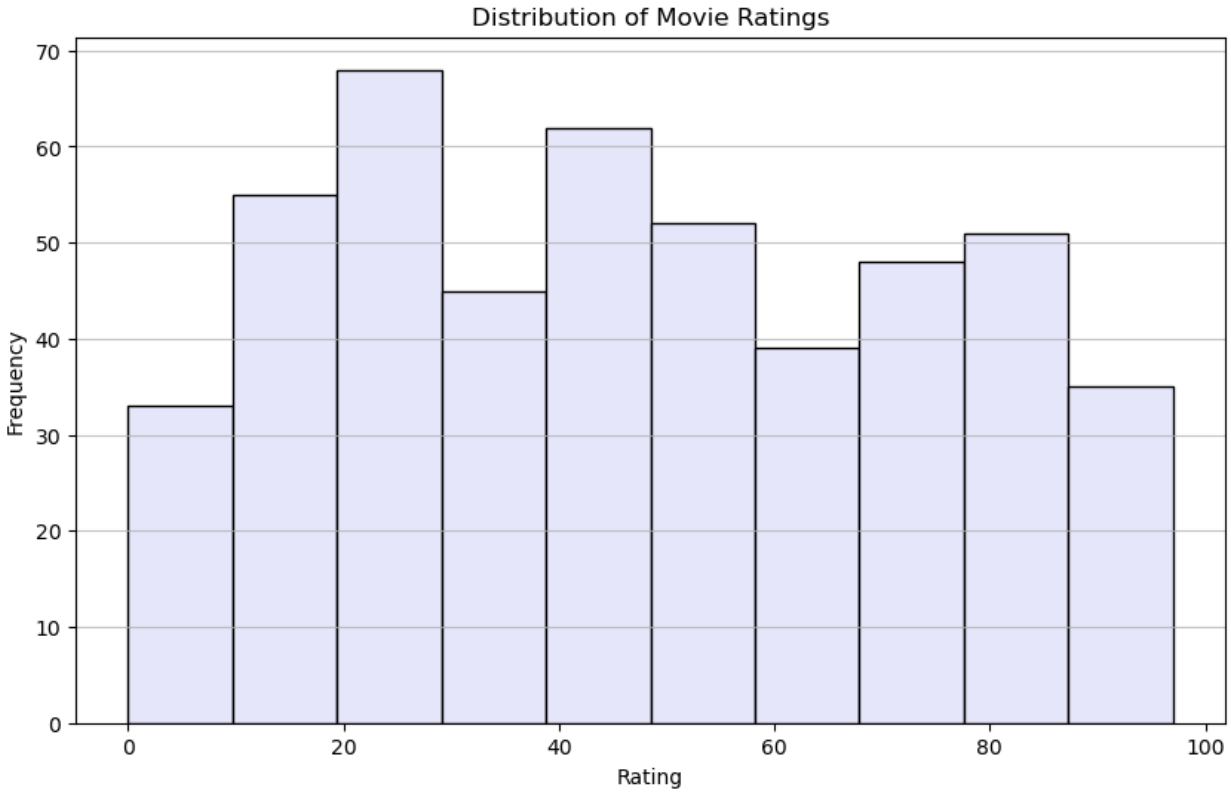
Div:-B Roll no :-0909

Sub:-Introduction To Data Science & Artificial Intelligence

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv('Movie-Ratings.csv')
data.head()
```

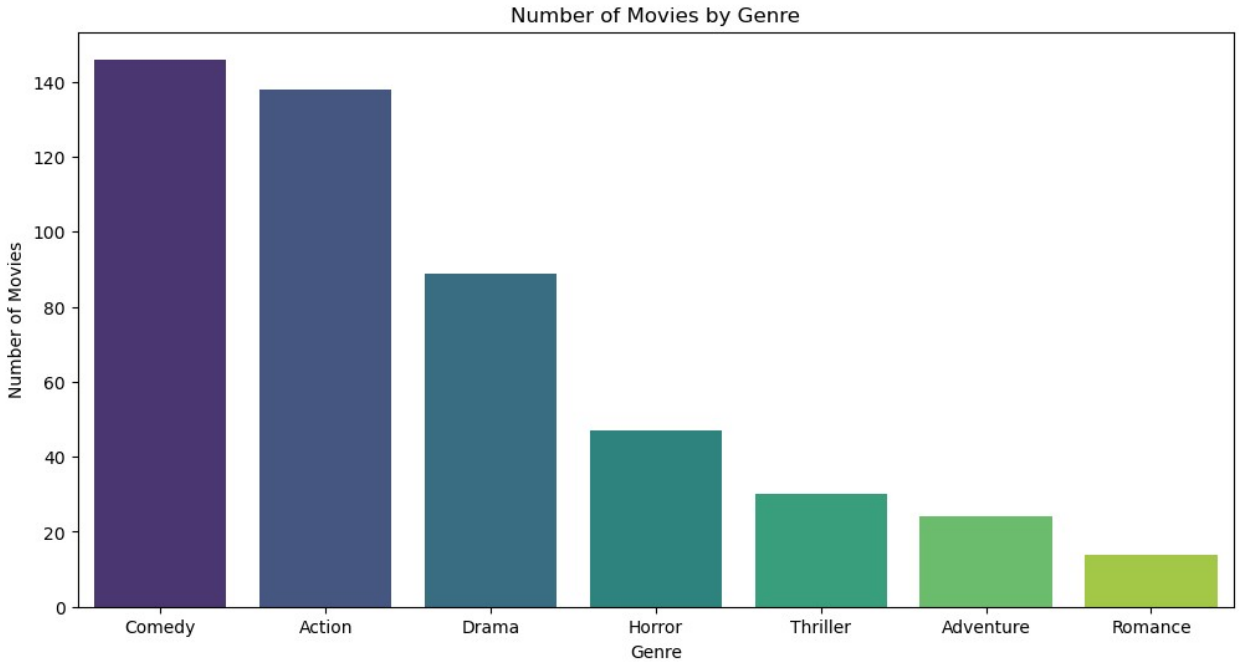
```
                                 Film   Genre  Rotten Tomatoes Ratings
%  \
0                  A Dangerous Method   Drama
79
1          A Nightmare on Elm Street  Horror
13
2                      A Serious Man    Drama
89
3  A Very Harold and Kumar Christmas  Comedy
72
4                          Abduction  Action
4

   Audience Ratings %  Budget (million $)  Year of release
0                  89                  20             2011
1                  40                  35             2010
2                  64                   7             2009
3                  71                  19             2011
4                  46                  35             2011
```

```
#Histogram
plt.figure(figsize=(10, 6))
plt.hist(data['Rotten Tomatoes Ratings %'], bins=10, color='lavender',
edgecolor='black')
plt.title('Distribution of Movie Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.grid(axis='y', alpha=0.75)
plt.show()
```



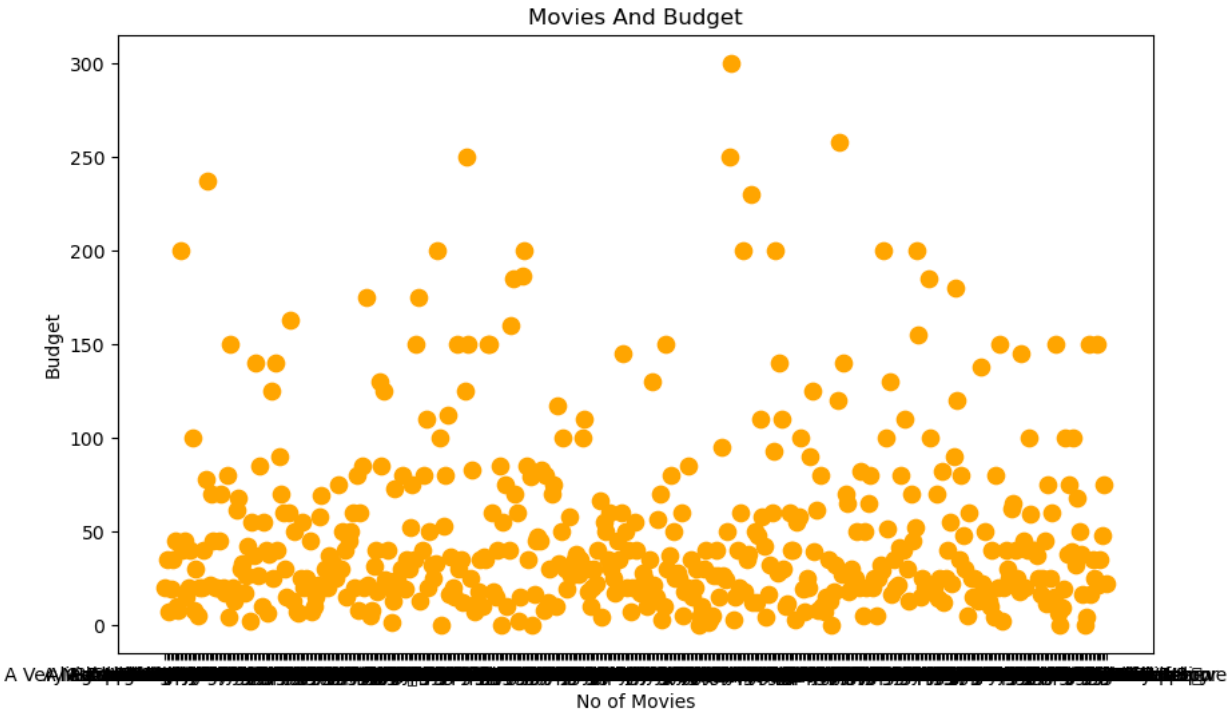

```
genre_counts = data['Genre'].value_counts()
plt.figure(figsize=(12, 6))
sns.barplot(x=genre_counts.index, y=genre_counts.values,
palette='viridis')
plt.title('Number of Movies by Genre')
plt.xlabel('Genre')
plt.ylabel('Number of Movies')
plt.show()
```

```
C:\Users\Akshay Ghadge\AppData\Local\Temp\
ipykernel_2740\2343381231.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=genre_counts.index, y=genre_counts.values,
palette='viridis')
```

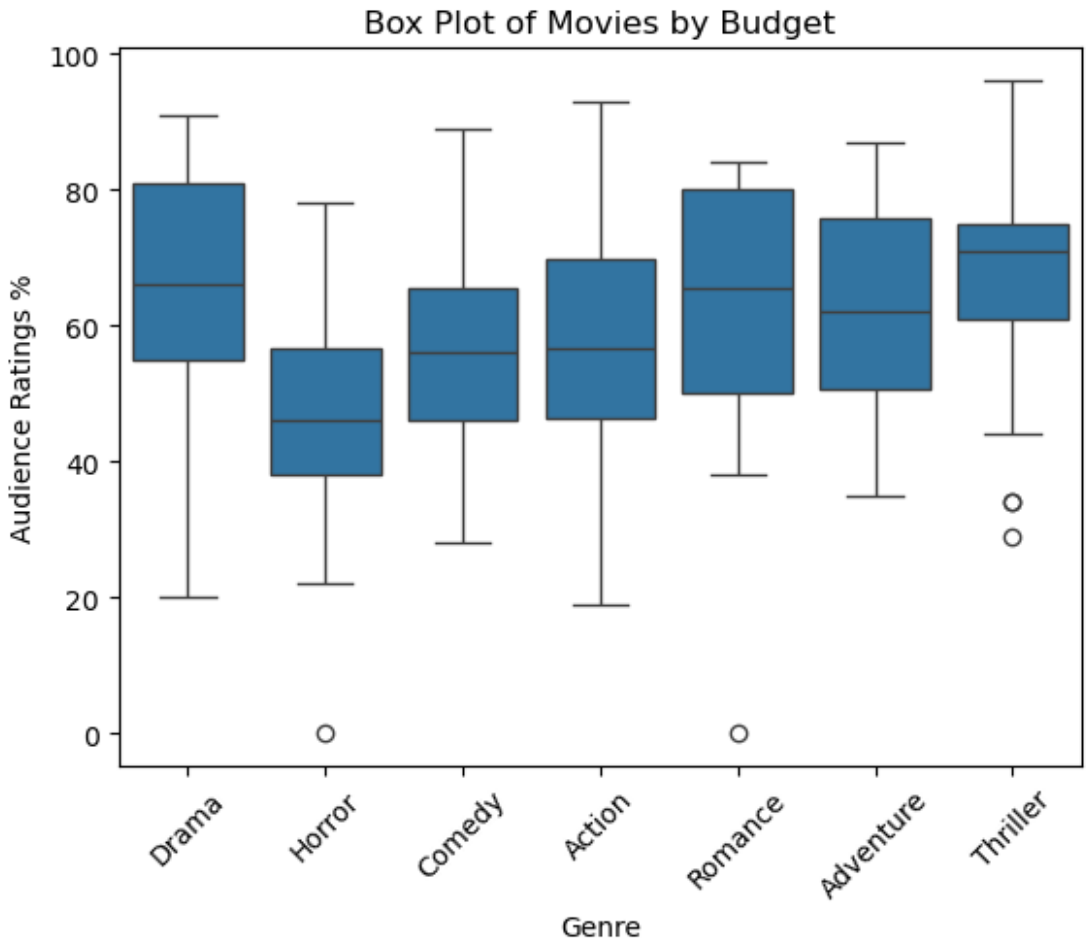



```
#Scatterplot
plt.figure(figsize=(10, 6))
plt.scatter(data['Film'], data['Budget (million $)'], color='orange',
s=78)  # s is the size of the markers
plt.title('Movies And Budget')
plt.xlabel('No of Movies')
plt.ylabel('Budget')
plt.show()
```

```
C:\Users\Akshay Ghadge\anaconda3\Lib\site-packages\IPython\core\
pylabtools.py:170: UserWarning: Glyph 9 (   ) missing from current
font.
  fig.canvas.print_figure(bytes_io, **kw)
```



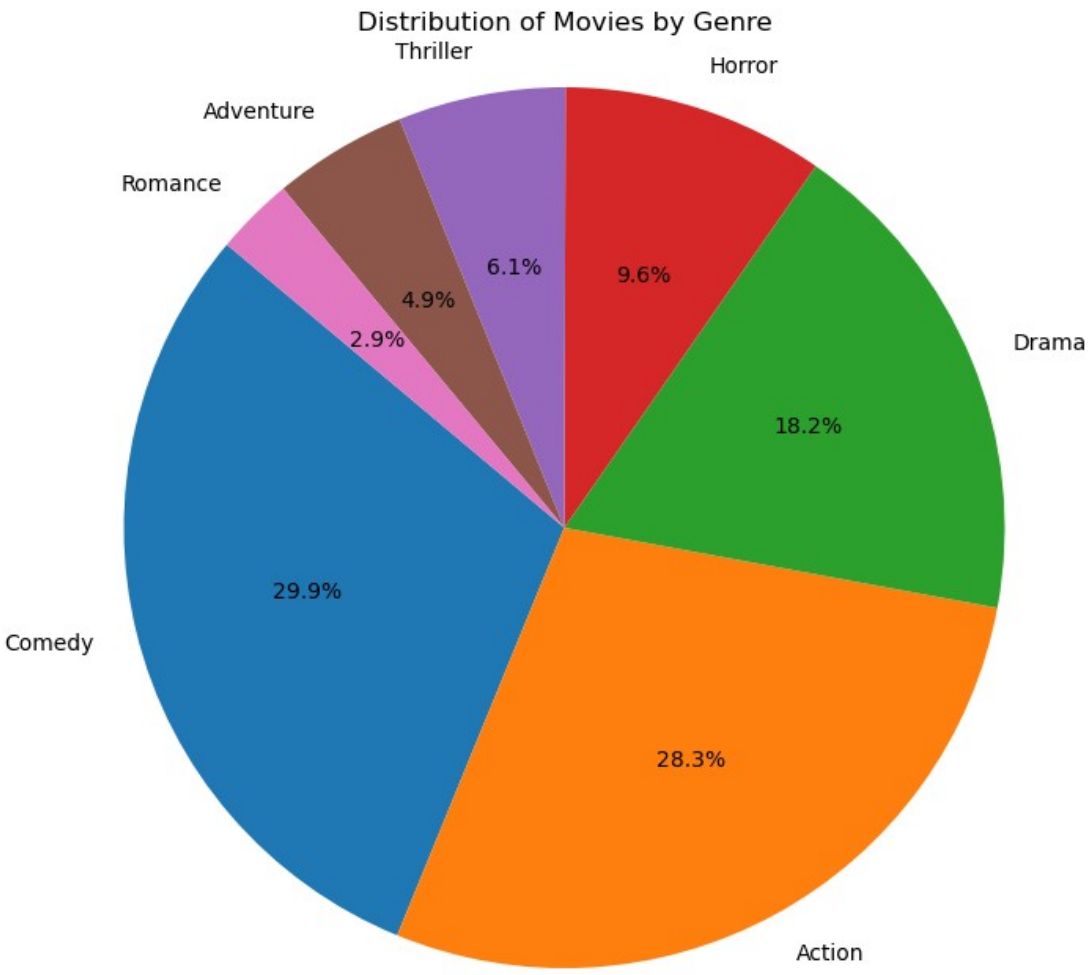

```
#Box Plot
data.head()
sns.boxplot(data=data, x='Genre', y='Audience Ratings %')
plt.title('Box Plot of Movies by Budget')
plt.xlabel('Genre')
plt.ylabel('Audience Ratings %')
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.show()
```



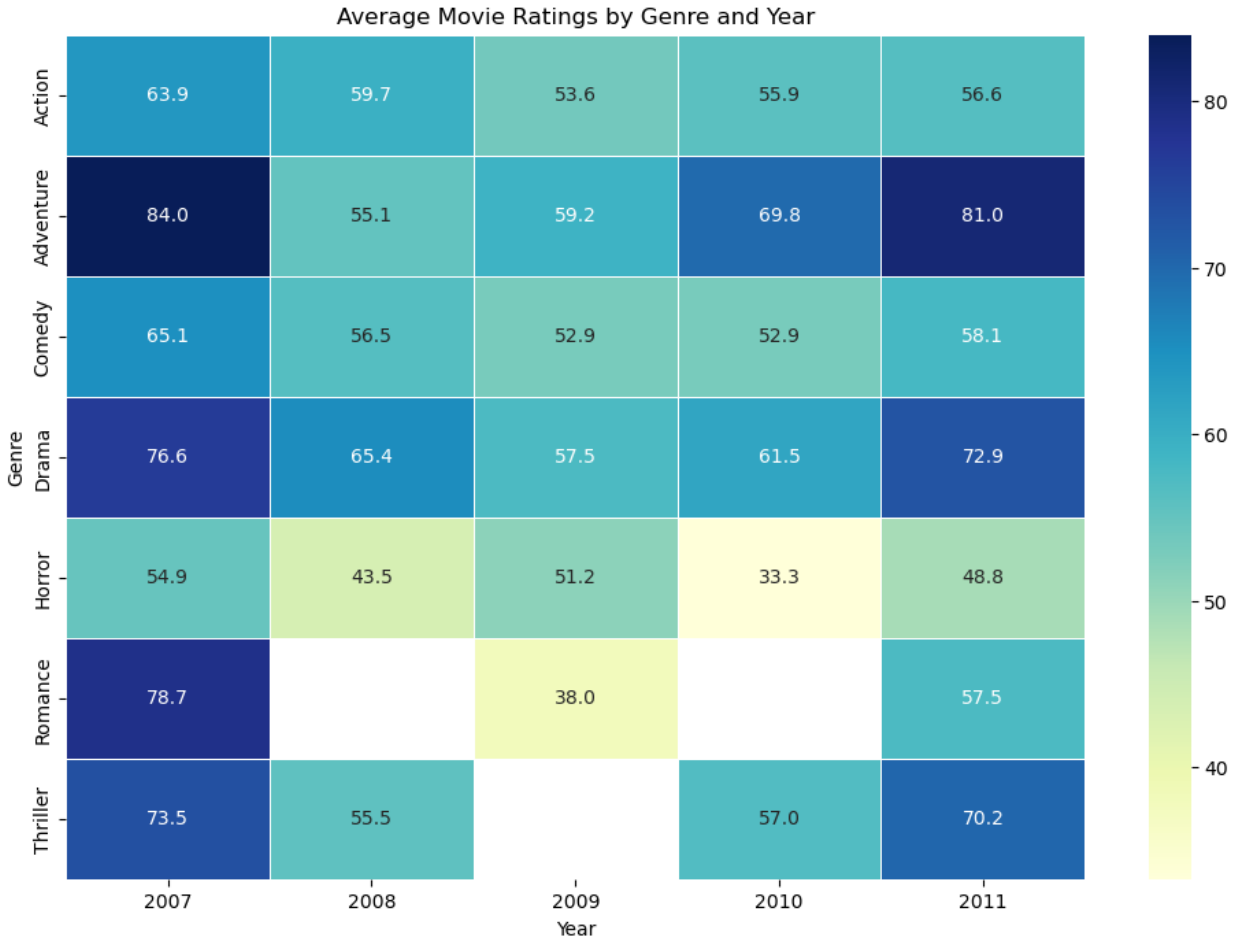

```
#pie chart
print(data.head())

# Count the number of movies in each genre
genre_counts = data['Genre'].value_counts()  # Replace 'Genre' with
the actual column name for genres

# Create a pie chart
plt.figure(figsize=(10, 8))
plt.pie(genre_counts, labels=genre_counts.index, autopct='%1.1f%%',
startangle=140)

# Customize the plot
plt.title('Distribution of Movies by Genre')
plt.axis('equal')  # Ensures the pie chart is a circle

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

```
                                  Film    Genre  Rotten Tomatoes Ratings
%  \
```

```
0                  A Dangerous Method   Drama
79
1           A Nightmare on Elm Street  Horror
13
2                       A Serious Man    Drama
89
3  A Very Harold and Kumar Christmas  Comedy
72
4                           Abduction  Action
4

   Audience Ratings %  Budget (million $)  Year of release
0                  89                  20             2011
1                  40                  35             2010
2                  64                   7             2009
3                  71                  19             2011
4                  46                  35             2011
```

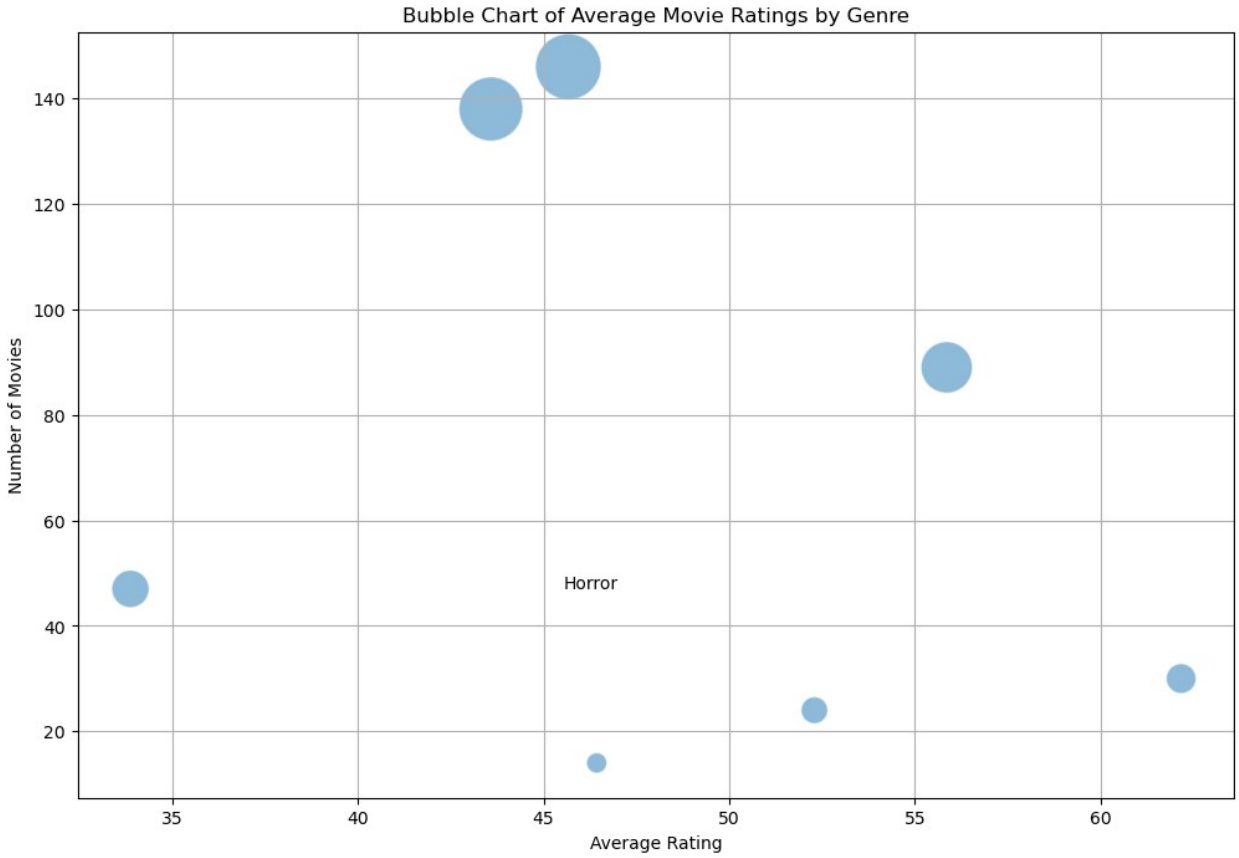


```
#heatmap chart
heatmap_data = data.pivot_table(values='Audience Ratings %',
index='Genre', columns='Year of release', aggfunc='mean')
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, fmt=".1f", cmap='YlGnBu',
linewidths=0.5)
plt.title('Average Movie Ratings by Genre and Year')
plt.xlabel('Year')
plt.ylabel('Genre')
plt.show()
```




```
#Bubble chart
genre_data = data.groupby('Genre').agg({'Rotten Tomatoes Ratings %':
'mean', 'Audience Ratings %': 'count'}).reset_index()  # Replace
'MovieID' with the actual column name for unique movie identifiers
plt.figure(figsize=(12, 8))
plt.scatter(x=genre_data['Rotten Tomatoes Ratings %'],
y=genre_data['Audience Ratings %'], s=genre_data['Audience Ratings %']
```

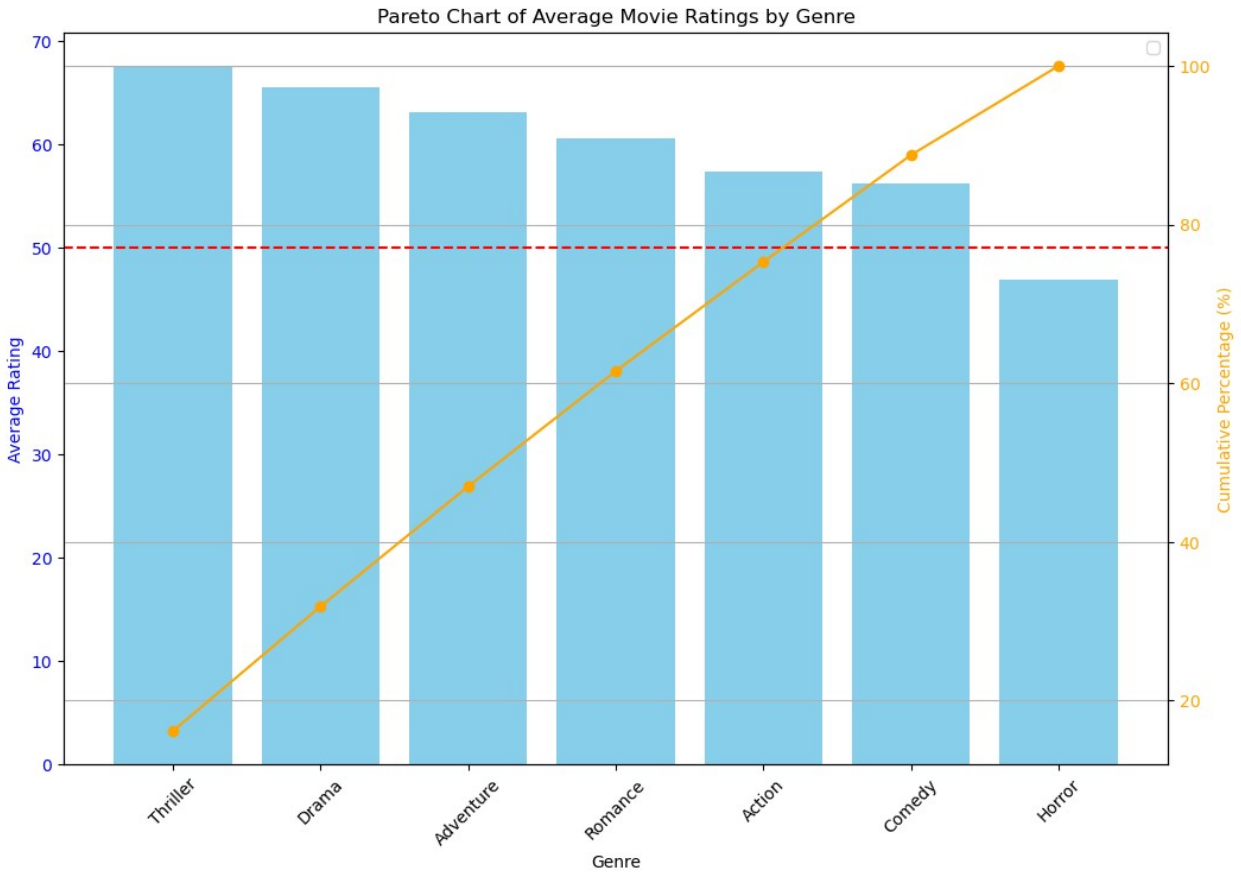
```
* 10, alpha=0.5, edgecolors='w')
plt.title('Bubble Chart of Average Movie Ratings by Genre')
plt.xlabel('Average Rating')
plt.ylabel('Number of Movies')
plt.grid(True)
for i in range(len(genre_data)):
    plt.annotate(genre_data['Genre'][i], (genre_data['Audience Ratings
%'][i], genre_data['Audience Ratings %'][i]), fontsize=10, ha='right')

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




```
#pareto chart
average_ratings = data.groupby('Genre')['Audience Ratings
%'].mean().reset_index()
average_ratings = average_ratings.sort_values(by='Audience Ratings %',
ascending=False)
average_ratings['Cumulative Sum'] = average_ratings['Audience Ratings
%'].cumsum()
average_ratings['Cumulative Percentage'] = 100 *
average_ratings['Cumulative Sum'] / average_ratings['Cumulative
Sum'].max()
#create pareto chart
```

```
fig, ax1 = plt.subplots(figsize=(12, 8))

# Bar chart for average ratings
ax1.bar(average_ratings['Genre'], average_ratings['Audience Ratings
%'], color='skyblue')
ax1.set_xlabel('Genre')
ax1.set_ylabel('Average Rating', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')
plt.xticks(rotation=45)

# Create a second y-axis for the cumulative percentage
ax2 = ax1.twinx()
ax2.plot(average_ratings['Genre'], average_ratings['Cumulative
Percentage'], color='orange', marker='o')
ax2.set_ylabel('Cumulative Percentage (%)', color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

# Customize the plot
plt.title('Pareto Chart of Average Movie Ratings by Genre')
ax1.axhline(50, color='red', linestyle='--', label='80% Threshold')  #
Example threshold
plt.legend()
plt.grid(axis='y')
plt.show()
```

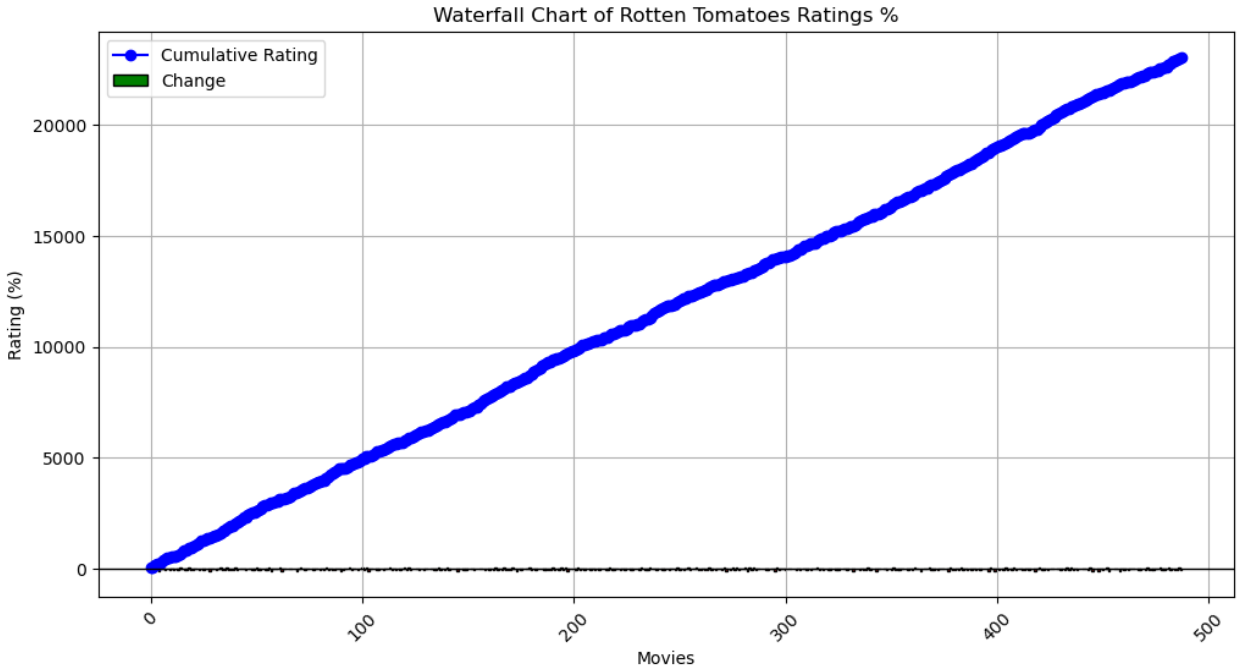
```
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```



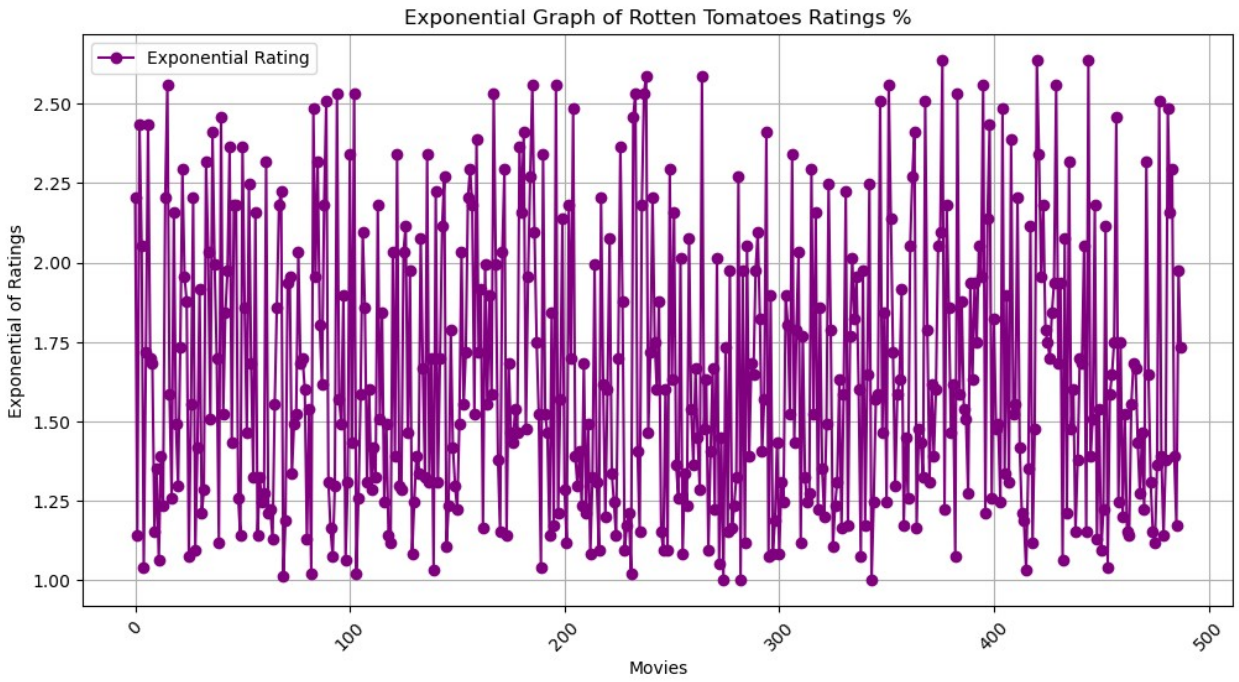

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data (ensure the path to the file is correct)
data = pd.read_csv('Movie-Ratings.csv')

# Check the first few rows to ensure data is loaded properly
print(data.head())  # Optional: Check if the 'Rotten Tomatoes Ratings
%' column exists

# Calculate cumulative sum of Rotten Tomatoes Ratings
data['Cumulative Rating'] = data['Rotten Tomatoes Ratings %'].cumsum()

# Calculate changes (differences between consecutive values)
data['Change'] = data['Rotten Tomatoes Ratings %'].diff().fillna(0)

# Create a color scheme for positive and negative changes
colors = ['green' if x >= 0 else 'red' for x in data['Change']]

# Plot the waterfall chart
plt.figure(figsize=(12, 6))
plt.bar(data.index, data['Change'], color=colors, edgecolor='black',
label='Change')
```

```
plt.plot(data.index, data['Cumulative Rating'], color='blue',
marker='o', label='Cumulative Rating')

# Customize plot
plt.title('Waterfall Chart of Rotten Tomatoes Ratings %')
plt.xlabel('Movies')
plt.ylabel('Rating (%)')
plt.xticks(rotation=45)  # Rotate x-axis labels if needed
plt.axhline(0, color='black', linewidth=1)  # Adding a horizontal line
at y=0 for reference
plt.legend()
plt.grid(True)

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

```
                                  Film   Genre  Rotten Tomatoes Ratings
%  \
0                 A Dangerous Method   Drama
79
1          A Nightmare on Elm Street  Horror
13
2                    A Serious Man    Drama
89
3  A Very Harold and Kumar Christmas  Comedy
72
4                          Abduction  Action
4

   Audience Ratings %  Budget (million $)  Year of release
0                  89                  20             2011
1                  40                  35             2010
2                  64                   7             2009
3                  71                  19             2011
4                  46                  35             2011
```




```
# Check if the Rotten Tomatoes Ratings % column exists
if 'Rotten Tomatoes Ratings %' in data.columns:
    # Calculate the exponential of Rotten Tomatoes Ratings %
    data['Exponential Rating'] = np.exp(data['Rotten Tomatoes Ratings
%'] / 100)  # Dividing by 100 for scaling

    # Plot the Exponential Graph
    plt.figure(figsize=(12, 6))
    plt.plot(data.index, data['Exponential Rating'], color='purple',
marker='o', linestyle='-', label='Exponential Rating')

    # Customize the plot
    plt.title('Exponential Graph of Rotten Tomatoes Ratings %')
    plt.xlabel('Movies')
    plt.ylabel('Exponential of Ratings')
    plt.xticks(rotation=45)  # Rotate x-axis labels if needed
    plt.legend()
    plt.grid(True)

    # Display the plot
    plt.show()
else:
    print("The column 'Rotten Tomatoes Ratings %' does not exist in
the data.")
```




```
# Ensure the column 'Rotten Tomatoes Ratings %' exists and 'Genre' for
grouping
if 'Rotten Tomatoes Ratings %' in data.columns and 'Genre' in
data.columns:
    # Calculate the average rating per genre
    genre_ratings = data.groupby('Genre')['Rotten Tomatoes Ratings
%'].mean()

    # Prepare data for the radar chart
    categories = list(genre_ratings.index)
    values = list(genre_ratings.values)

    # Close the radar chart by appending the first value to the end
    values += values[:1]

    # Radar chart setup
    angles = np.linspace(0, 2 * np.pi, len(values),
endpoint=False).tolist()

    # Plotting
    fig, ax = plt.subplots(figsize=(8, 8),
subplot_kw=dict(polar=True))
    ax.fill(angles, values, color='purple', alpha=0.25)
    ax.plot(angles, values, color='purple', linewidth=2)

    # Customize chart appearance
    ax.set_yticklabels([])
    ax.set_xticks(angles[:-1])
```

```
    ax.set_xticklabels(categories, fontsize=10)
    ax.set_title('Average Rotten Tomatoes Rating by Genre', size=15,
color='purple')

    # Show the plot
    plt.show()
else:
    print("Required columns 'Rotten Tomatoes Ratings %' or 'Genre' are
missing in the data.")
```