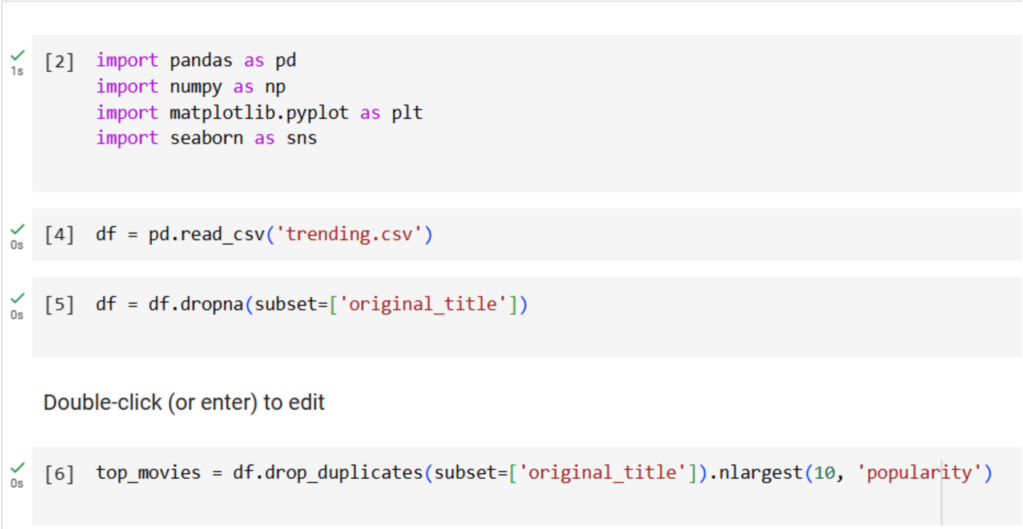
Name : Ishan Kiran Joshi **Roll No : 21** **Class : D15C**

DS Lab Experiment-2

Q. Perform following data visualization and exploration on your selected dataset.
1. Create bar graph, contingency table using any 2 features.
2. Plot Scatter plot, box plot, Heatmap using seaborn.
3. Create histogram and normalized Histogram.
4. Describe what this graph and table indicates.
5. Handle outlier using box plot and Inter quartile range.

- **Data Loading & Preprocessing**



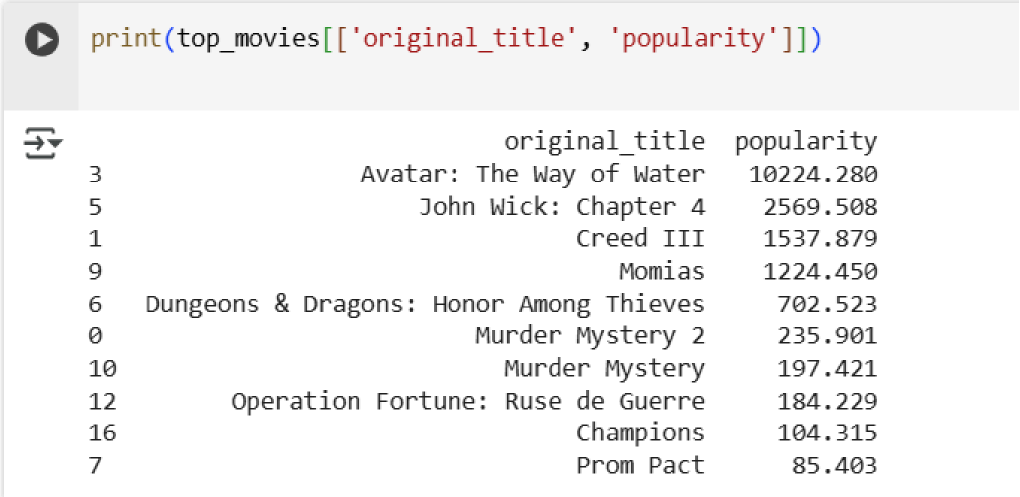
```
[2] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
```

```
[4] df = pd.read_csv('trending.csv')
```

```
[5] df = df.dropna(subset=['original_title'])
```

Double-click (or enter) to edit

```
[6] top_movies = df.drop_duplicates(subset=['original_title']).nlargest(10, 'popularity')
```



```
print(top_movies[['original_title', 'popularity']])
```

```
                          original_title  popularity
3                Avatar: The Way of Water   10224.280
5                      John Wick: Chapter 4    2569.508
1                               Creed III    1537.879
9                                  Momias    1224.450
6   Dungeons & Dragons: Honor Among Thieves     702.523
0                       Murder Mystery 2     235.901
10                        Murder Mystery     197.421
12        Operation Fortune: Ruse de Guerre     184.229
16                              Champions     104.315
7                               Prom Pact      85.403
```

Loading the Dataset (pd.read_csv)

- Reads the CSV file (trending.csv) and loads it into a Pandas DataFrame.

Handling Missing Values (dropna)

- Removes rows where the column 'original_title' has missing (NaN) values.

Removing Duplicates (drop_duplicates)

- Ensures each movie title appears only once.

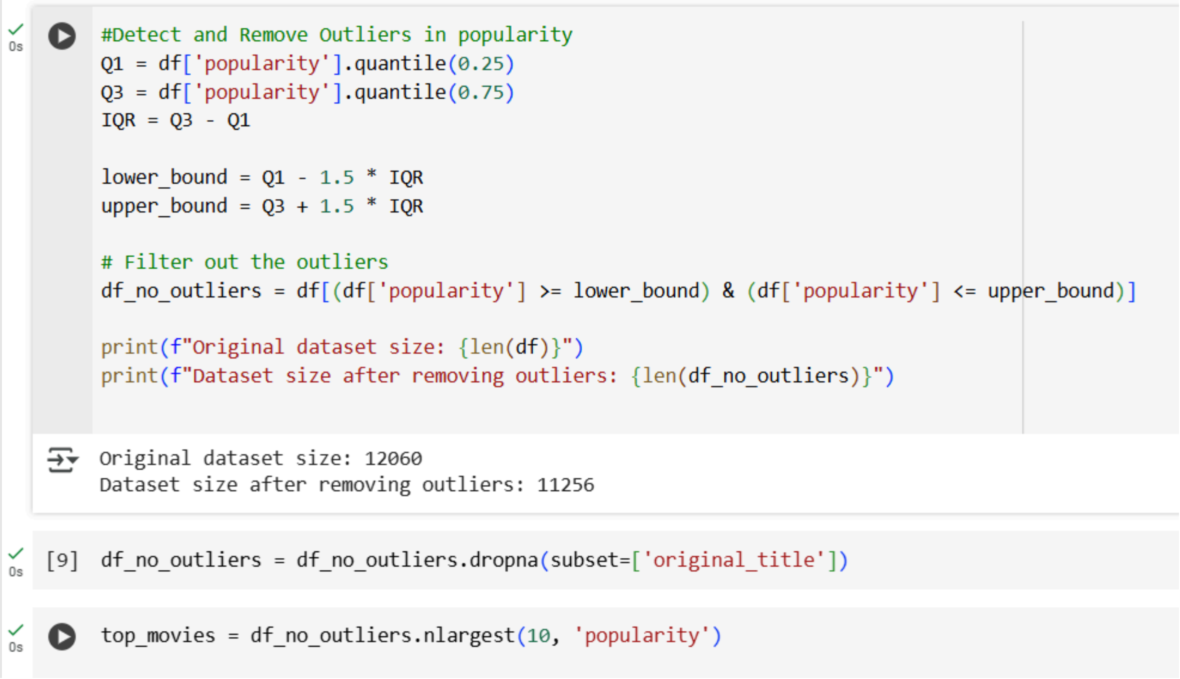
Selecting Top 10 Movies (nlargest)

- Sorts the movies by popularity and selects the top 10.

Displaying Data (print)

- Prints a table with movie titles and their popularity scores.

- **Outlier Detection & Handling (IQR Method)**



```
#Detect and Remove Outliers in popularity
Q1 = df['popularity'].quantile(0.25)
Q3 = df['popularity'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter out the outliers
df_no_outliers = df[(df['popularity'] >= lower_bound) & (df['popularity'] <= upper_bound)]

print(f"Original dataset size: {len(df)}")
print(f"Dataset size after removing outliers: {len(df_no_outliers)}")
```

```
Original dataset size: 12060
Dataset size after removing outliers: 11256
```

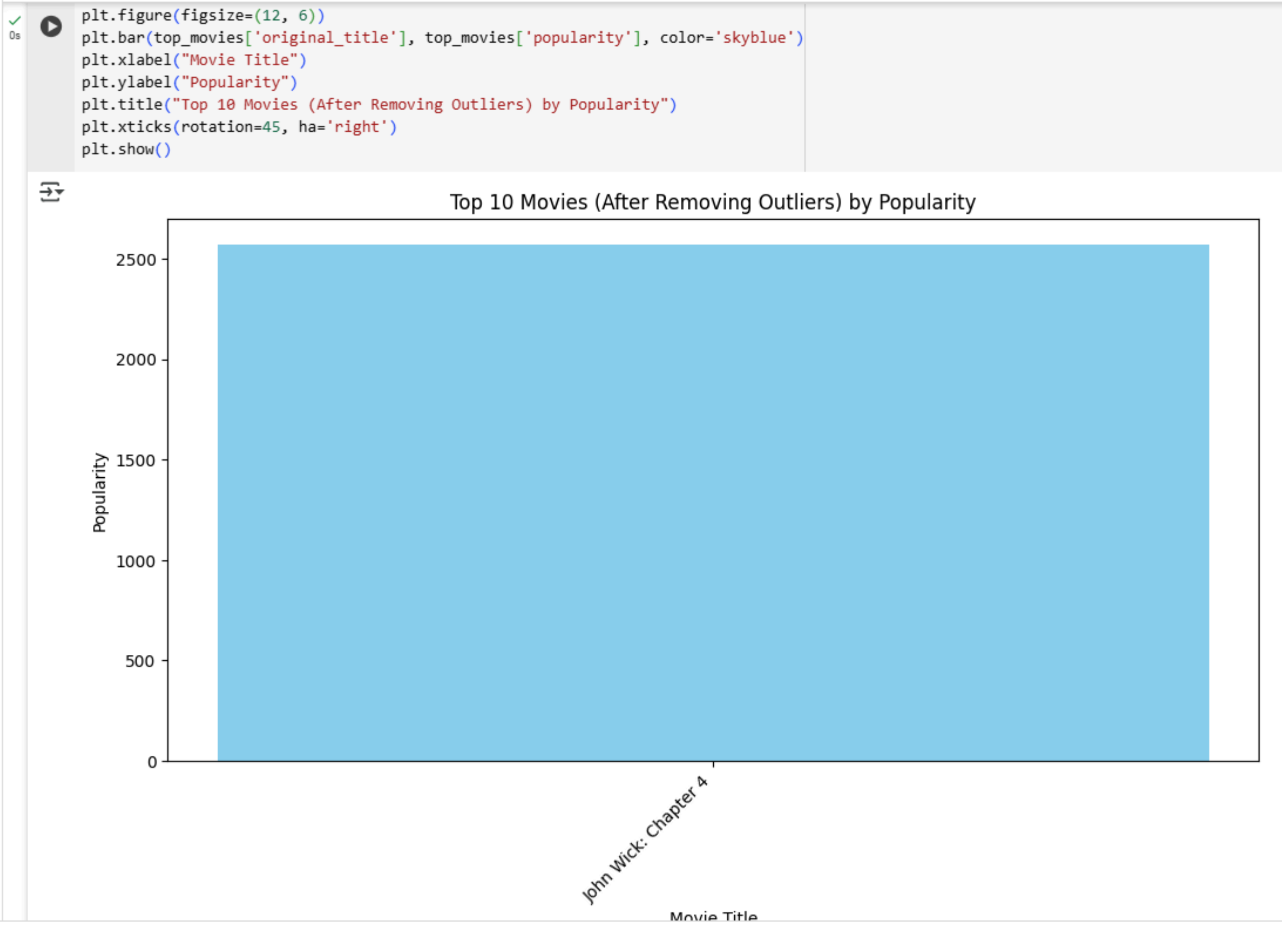
```
[9] df_no_outliers = df_no_outliers.dropna(subset=['original_title'])
```

```
top_movies = df_no_outliers.nlargest(10, 'popularity')
```

This step detects and removes outliers from the 'popularity' column using the Interquartile Range (IQR) method. The first quartile (Q1) and third quartile (Q3) define the middle 50% of the data, and the IQR (Q3 - Q1) is used to calculate outlier boundaries. Any value beyond 1.5 times the IQR from Q1 or Q3 is considered an outlier and removed.

After removing outliers, the dataset size reduces from 12,060 to 11,256, ensuring cleaner data. The dataset is further refined by dropping missing movie titles and selecting the top 10 most popular movies. This step prevents extreme values from skewing the analysis.

- **Bar Graph**



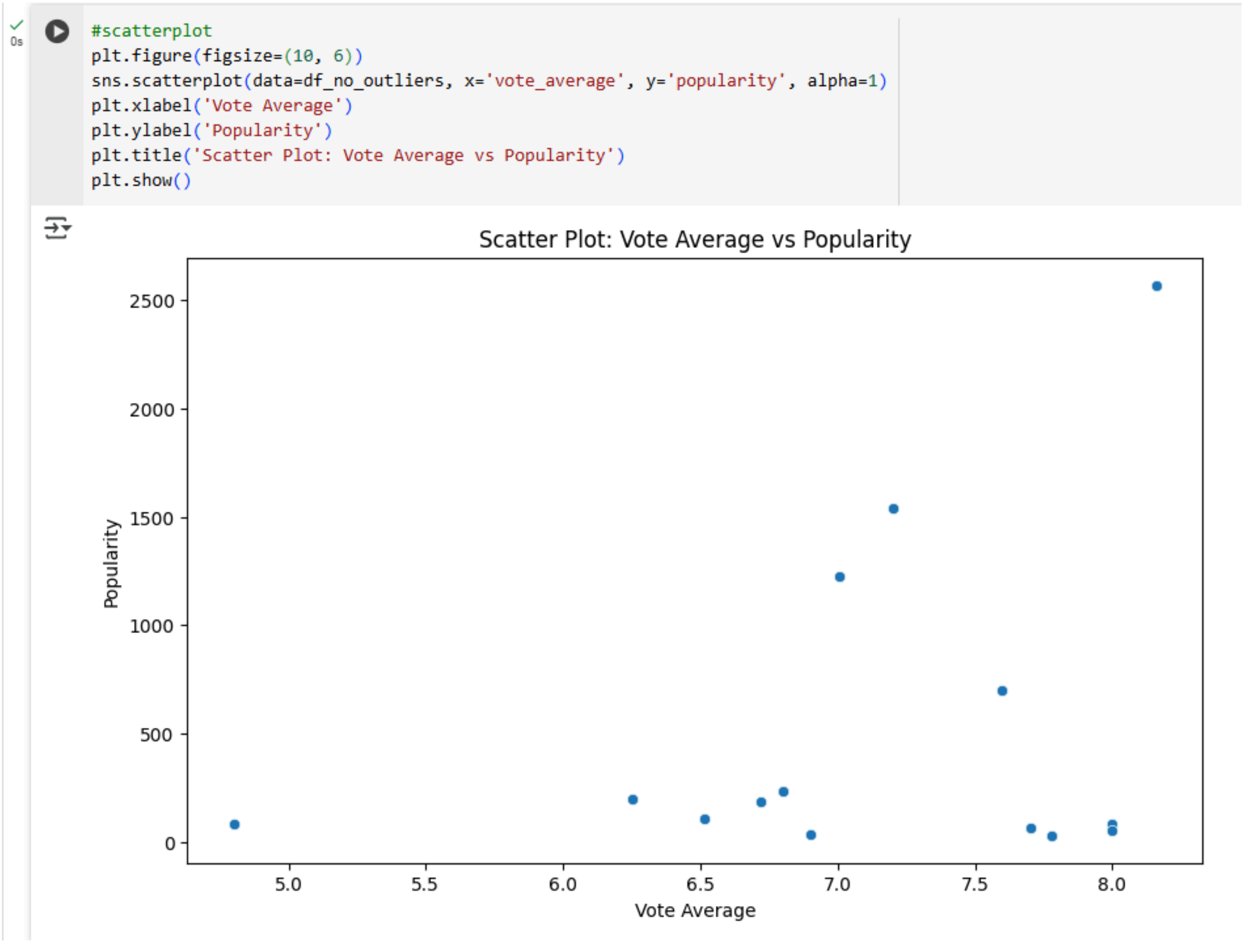
```
plt.figure(figsize=(12, 6))
plt.bar(top_movies['original_title'], top_movies['popularity'], color='skyblue')
plt.xlabel("Movie Title")
plt.ylabel("Popularity")
plt.title("Top 10 Movies (After Removing Outliers) by Popularity")
plt.xticks(rotation=45, ha='right')
plt.show()
```

This bar graph visualizes the top 10 most popular movies after removing outliers. The x-axis represents movie titles, while the y-axis shows their popularity scores. The bars are colored sky blue for clarity, and the movie titles are rotated for better readability.

Observations:

- The most popular movie has an overwhelmingly high score compared to the others.
- The popularity distribution seems highly skewed, possibly dominated by one or two major titles.

- **Scatter Plot**



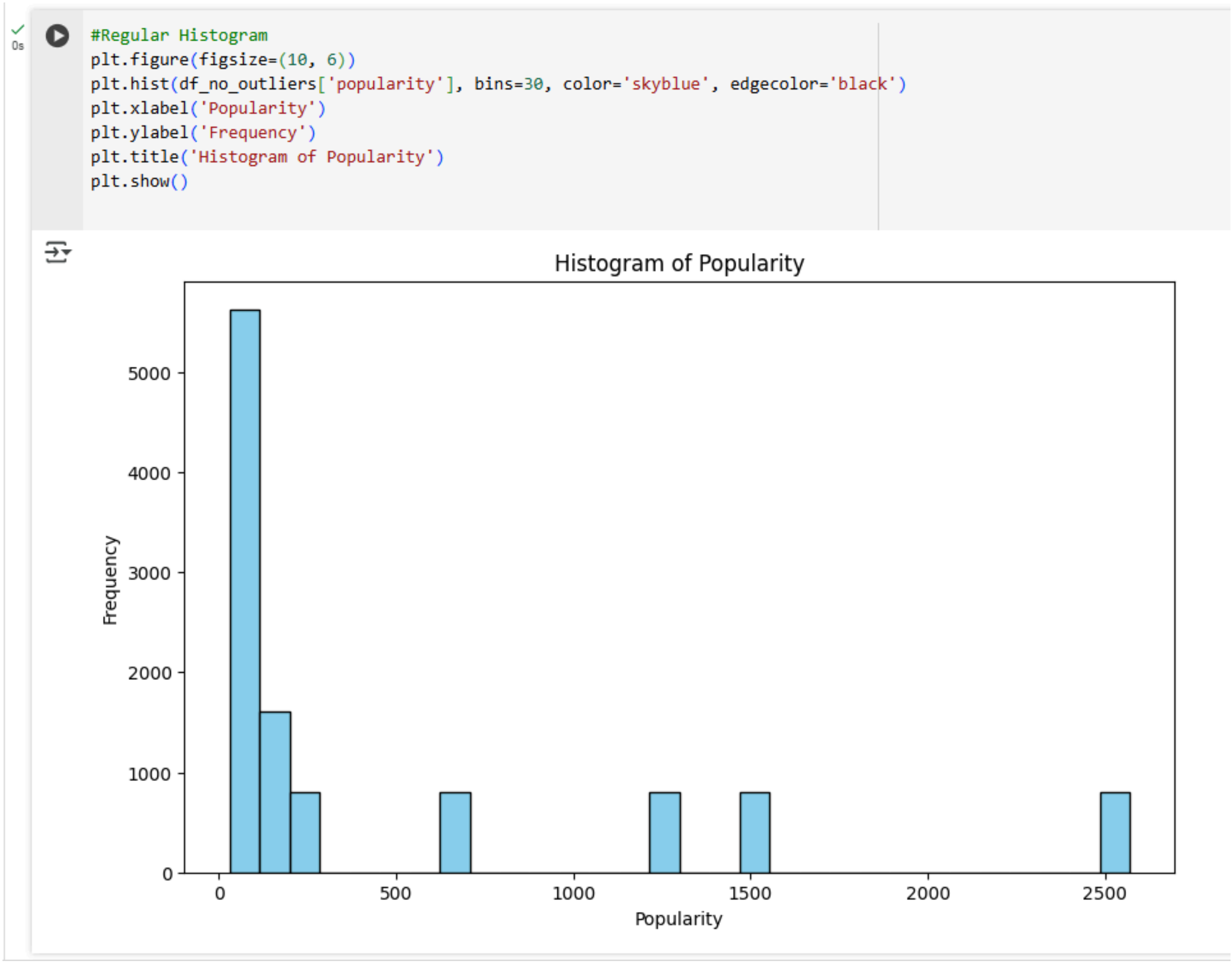
```
#scatterplot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_no_outliers, x='vote_average', y='popularity', alpha=1)
plt.xlabel('Vote Average')
plt.ylabel('Popularity')
plt.title('Scatter Plot: Vote Average vs Popularity')
plt.show()
```

This scatter plot visualizes the relationship between vote average and popularity for movies. The x-axis represents the vote average (ratings), while the y-axis shows popularity scores. Each dot corresponds to a movie, indicating how its rating correlates with popularity.

Observations:

- Most movies have a moderate vote average (6-8) but vary significantly in popularity.
- A few movies with high vote averages (above 8) show extreme popularity, suggesting they are widely recognized hits.
- There's no strong linear correlation, as movies with similar ratings have vastly different popularity scores.

- **Histogram (Regular and Normalised)**



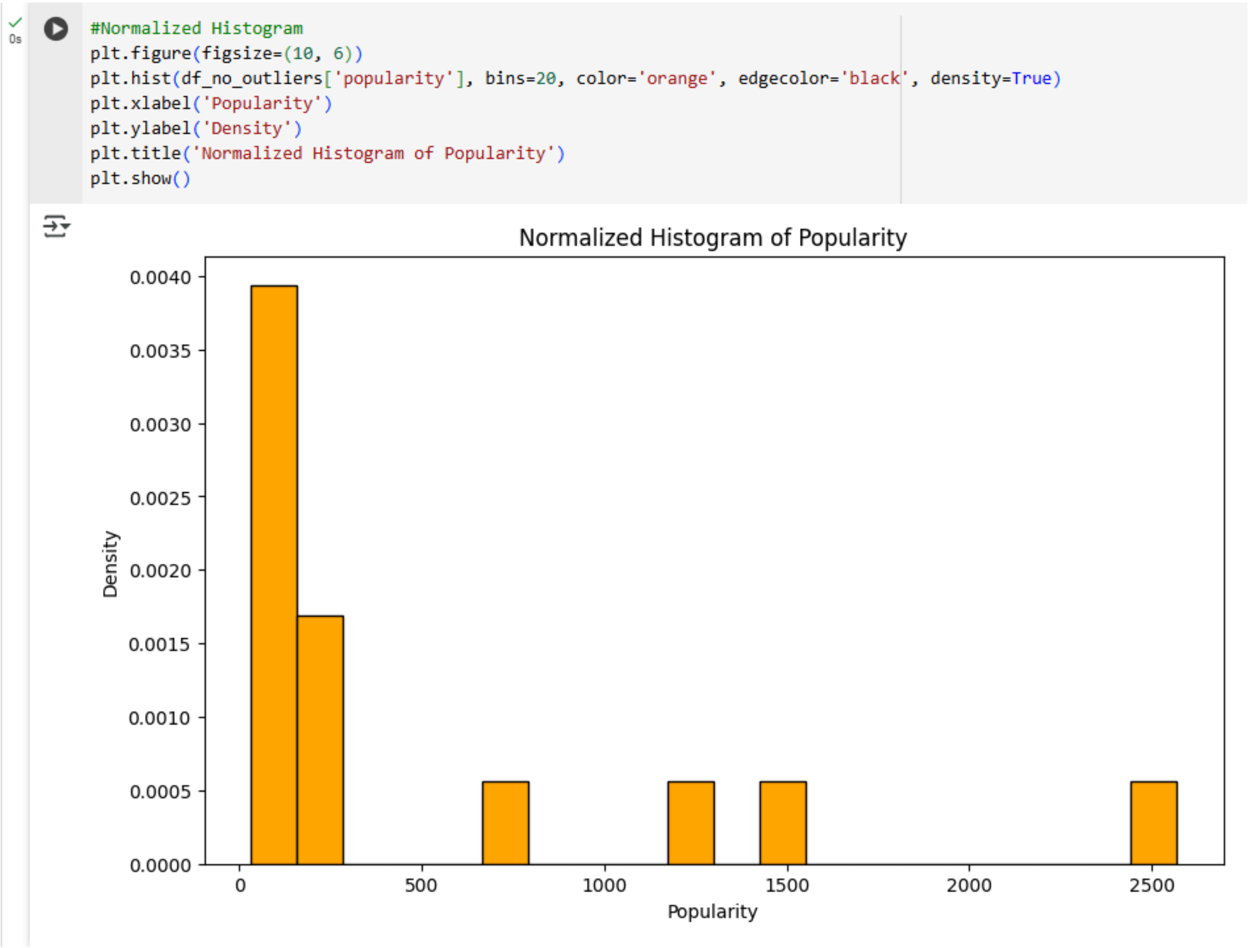
```
#Regular Histogram
plt.figure(figsize=(10, 6))
plt.hist(df_no_outliers['popularity'], bins=30, color='skyblue', edgecolor='black')
plt.xlabel('Popularity')
plt.ylabel('Frequency')
plt.title('Histogram of Popularity')
plt.show()
```

This histogram represents the distribution of movie popularity scores. The x-axis shows popularity values, while the y-axis indicates how frequently those values occur.

Observations:

- The majority of movies have low popularity scores, with a sharp drop-off as popularity increases.
- A few movies have very high popularity, appearing as isolated bars on the right side of the graph.
- The distribution is highly skewed, suggesting that only a handful of movies achieve extreme popularity, while most remain relatively unknown.

Normalised Histogram



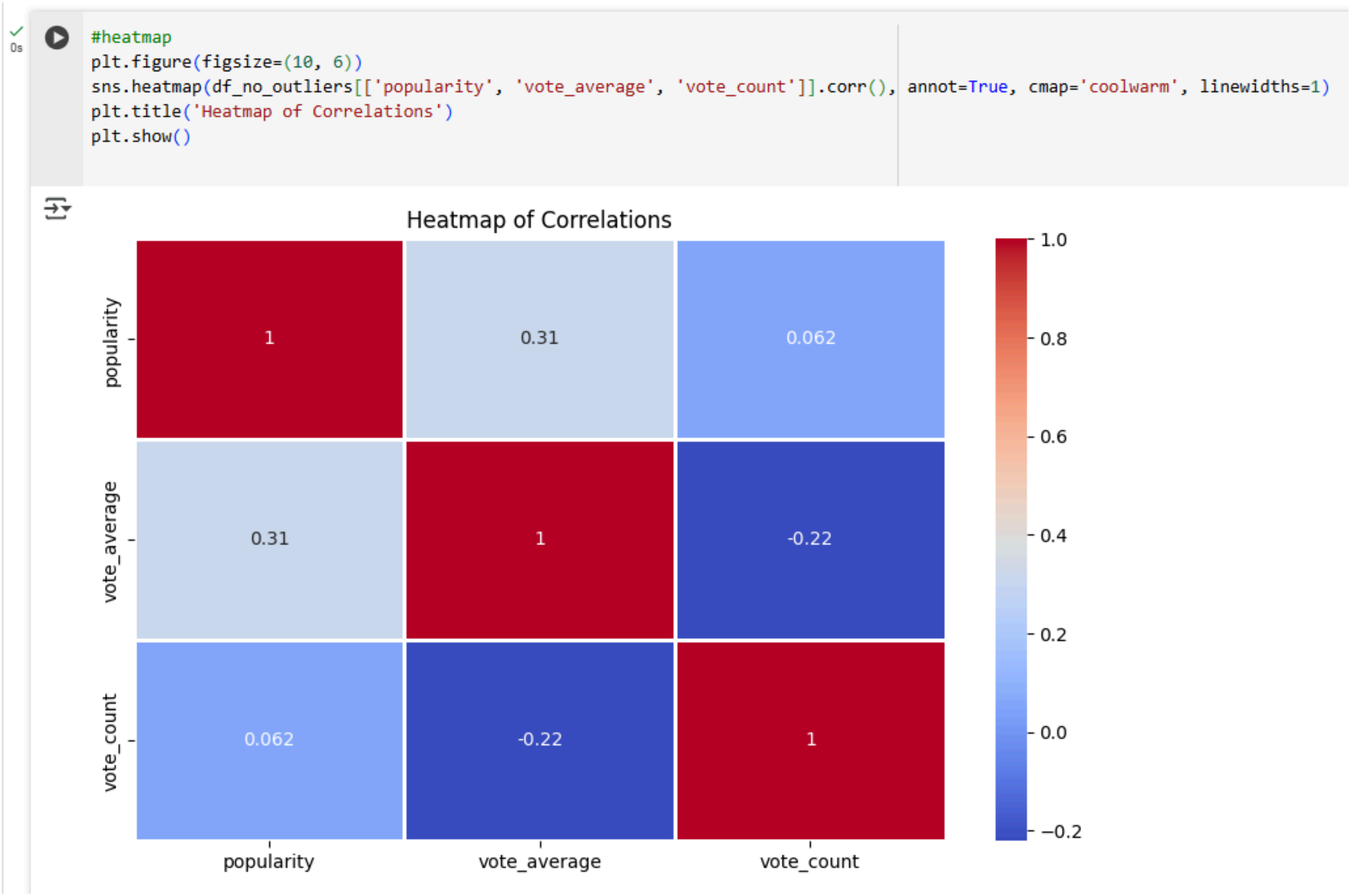
```
#Normalized Histogram
plt.figure(figsize=(10, 6))
plt.hist(df_no_outliers['popularity'], bins=20, color='orange', edgecolor='black', density=True)
plt.xlabel('Popularity')
plt.ylabel('Density')
plt.title('Normalized Histogram of Popularity')
plt.show()
```

This histogram represents the normalized distribution of movie popularity scores.

Observations:

- Density-based scaling: The y-axis now represents probability density instead of raw frequency, making it easier to compare different datasets.
- Highly skewed distribution: Most movies have low popularity, while a few movies are outliers with extremely high popularity.
- Smooth probability distribution: Normalizing helps in understanding relative likelihood rather than absolute counts.

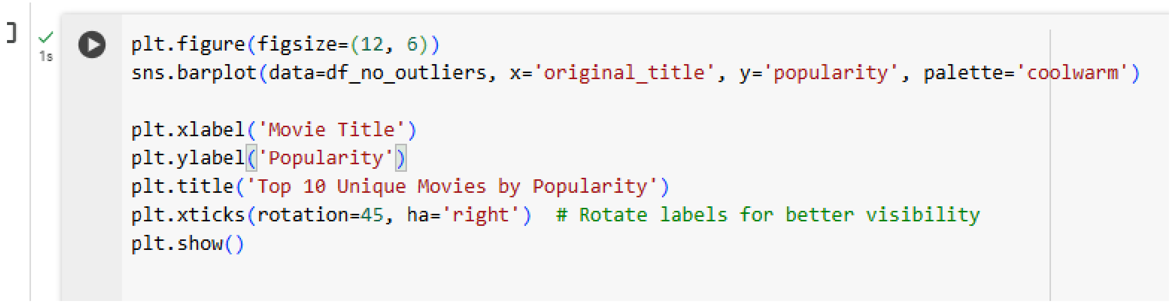
- **HeatMap**



```
#heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df_no_outliers[['popularity', 'vote_average', 'vote_count']].corr(), annot=True, cmap='coolwarm', linewidths=1)
plt.title('Heatmap of Correlations')
plt.show()
```

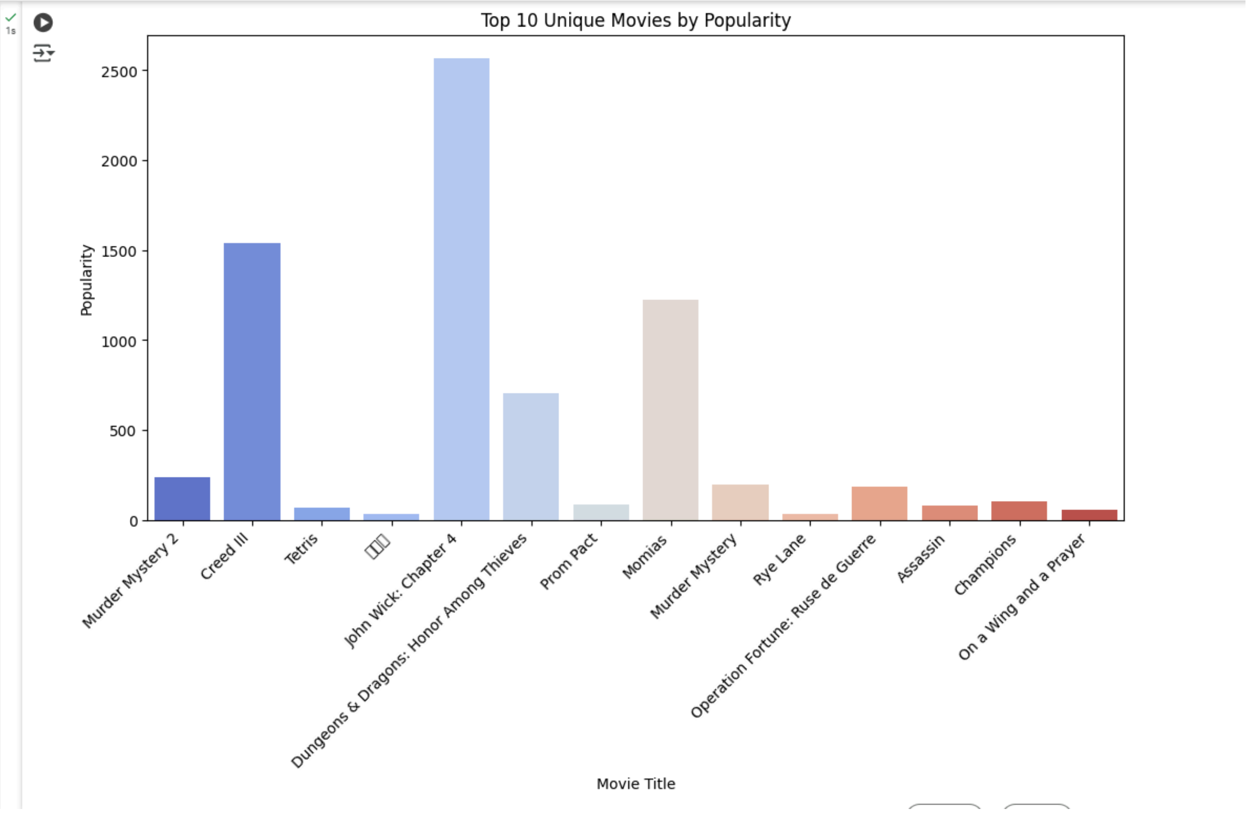
- Popularity vs Vote Average: Weak positive correlation (0.31) → More popular movies tend to have slightly higher ratings.
- Popularity vs Vote Count: Almost no correlation (0.062) → A movie's popularity does not strongly relate to how many people voted.
- Vote Average vs Vote Count: Slight negative correlation (-0.22) → More votes might slightly lower the average rating, possibly due to mixed opinions.

- **Bar Graph**



```
plt.figure(figsize=(12, 6))
sns.barplot(data=df_no_outliers, x='original_title', y='popularity', palette='coolwarm')

plt.xlabel('Movie Title')
plt.ylabel('Popularity')
plt.title('Top 10 Unique Movies by Popularity')
plt.xticks(rotation=45, ha='right')  # Rotate labels for better visibility
plt.show()
```



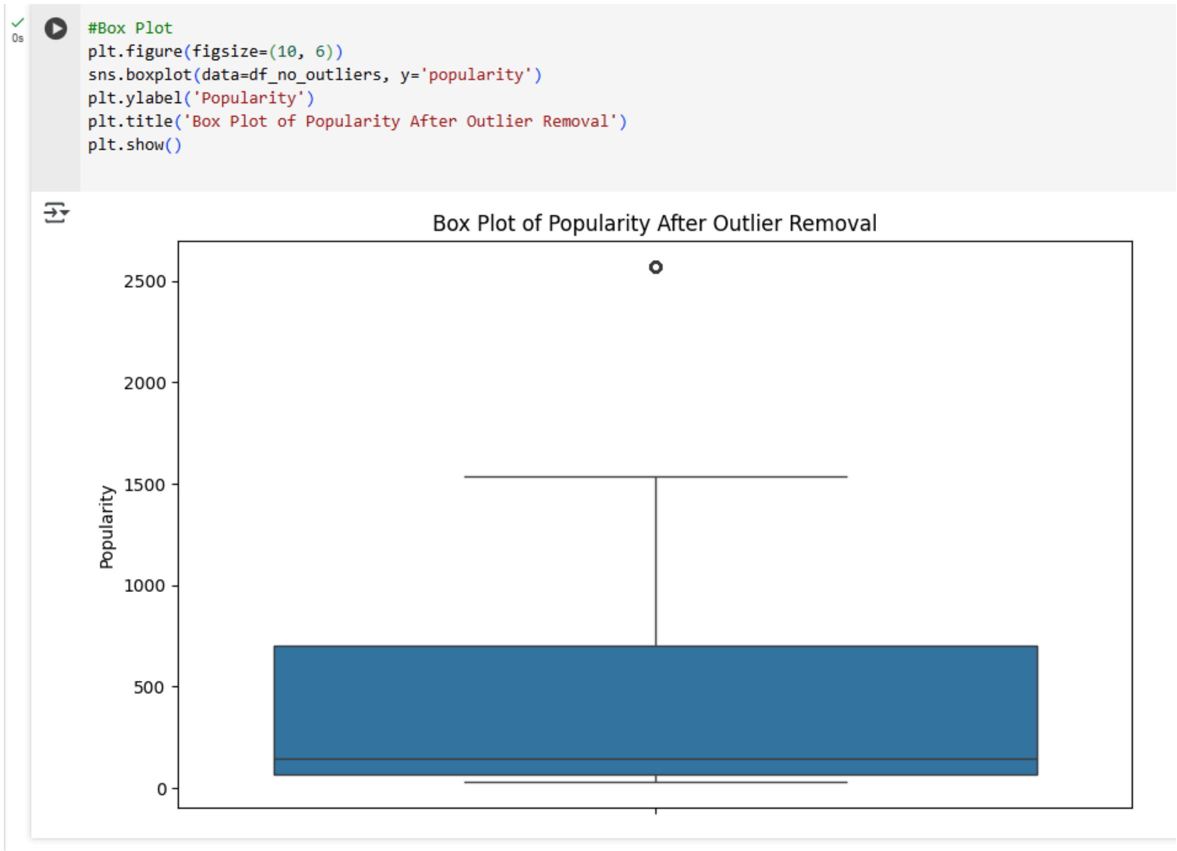
The bar graph displays the popularity of different movies based on a dataset. The x-axis represents the movie titles, while the y-axis represents their popularity scores. The color scheme (coolwarm palette) visually distinguishes between high and low popularity values.

Key Observations:

1. John Wick: Chapter 4 has the highest popularity score, significantly surpassing other movies in the dataset.
2. Creed III and Momias also have relatively high popularity scores, making them notable competitors in terms of audience engagement.
3. There is a steep decline in popularity after the top three movies, with several movies showing much lower scores.

4. Some movies, such as *Rye Lane, Assassin, and On a Wing and a Prayer*, have considerably lower popularity, indicating less audience engagement.
5. The variation in popularity suggests that a few movies dominate public interest, while others have minimal reach.

- **Box Plot**



```
#Box Plot
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_no_outliers, y='popularity')
plt.ylabel('Popularity')
plt.title('Box Plot of Popularity After Outlier Removal')
plt.show()
```

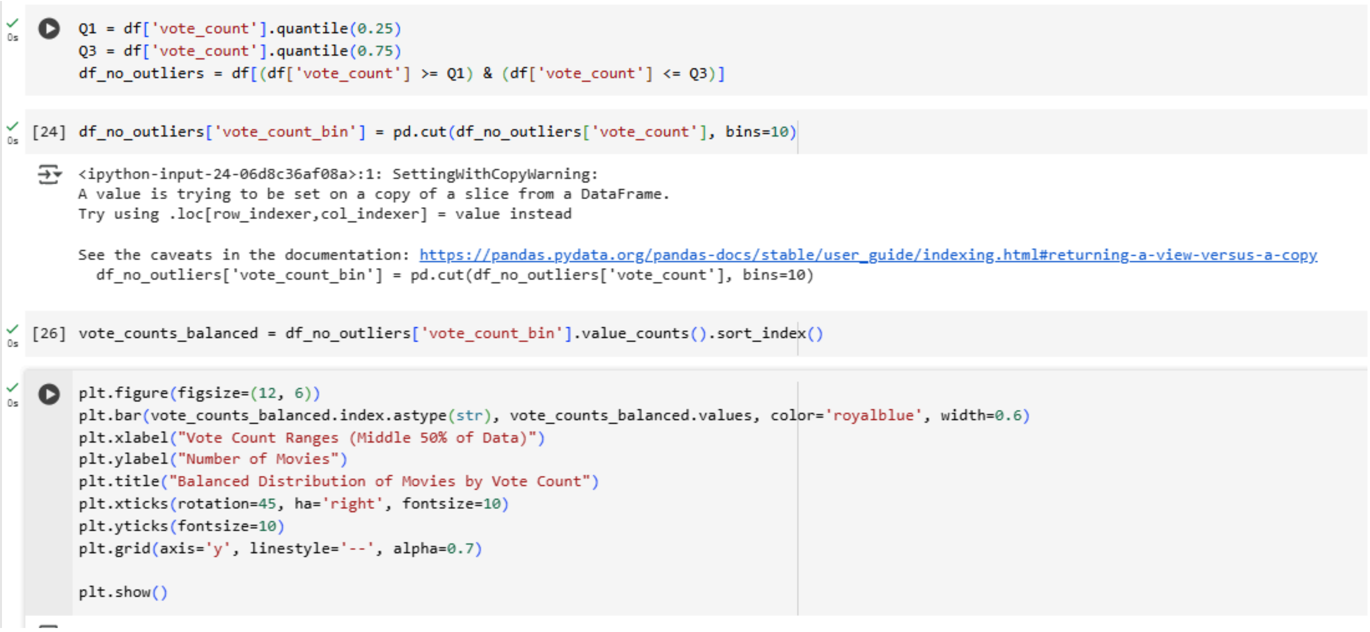
The box plot visualizes the distribution of movie popularity after removing outliers. The y-axis represents the popularity scores, while the box plot provides insights into the spread and central tendency of the data.

Observations :

- The median popularity is relatively low, indicating most movies have moderate popularity.
- Whiskers show the spread, with a few outliers exceeding 2500 in popularity.
- The distribution is right-skewed, meaning some movies are significantly more popular.
- Despite outlier removal, some movies still dominate in popularity.

Name : Ishan Kiran Joshi **Roll No : 21** **Class : D15C**

- **Bar Chart**



```
Q1 = df['vote_count'].quantile(0.25)
Q3 = df['vote_count'].quantile(0.75)
df_no_outliers = df[(df['vote_count'] >= Q1) & (df['vote_count'] <= Q3)]
```

```
[24] df_no_outliers['vote_count_bin'] = pd.cut(df_no_outliers['vote_count'], bins=10)
```

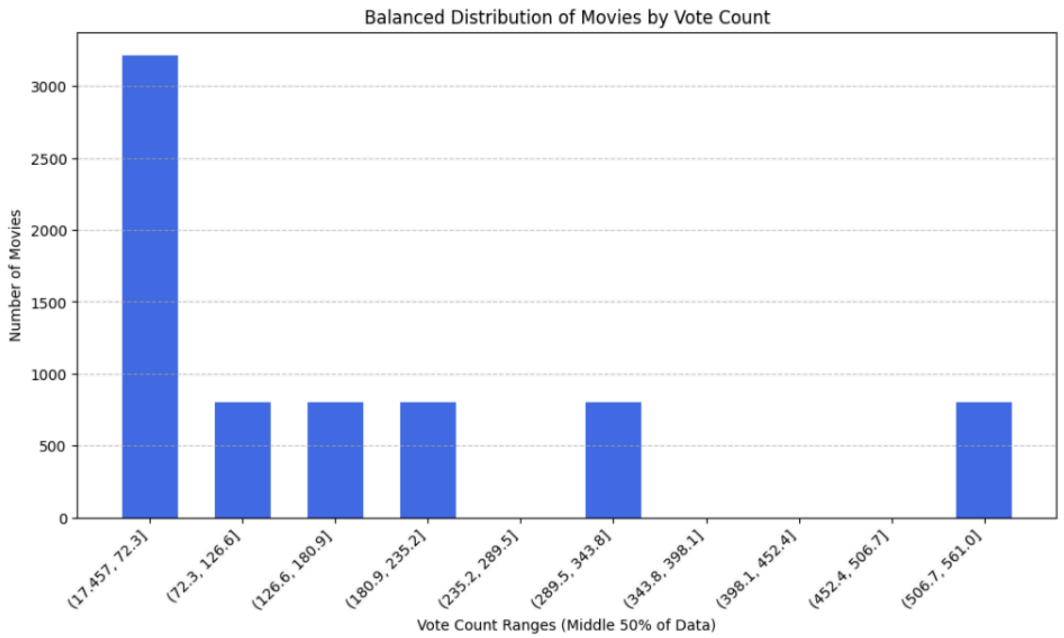
```
<ipython-input-24-06d8c36af08a>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_no_outliers['vote_count_bin'] = pd.cut(df_no_outliers['vote_count'], bins=10)
```

```
[26] vote_counts_balanced = df_no_outliers['vote_count_bin'].value_counts().sort_index()
```

```
plt.figure(figsize=(12, 6))
plt.bar(vote_counts_balanced.index.astype(str), vote_counts_balanced.values, color='royalblue', width=0.6)
plt.xlabel("Vote Count Ranges (Middle 50% of Data)")
plt.ylabel("Number of Movies")
plt.title("Balanced Distribution of Movies by Vote Count")
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```



- Most movies have low vote counts (17.45 - 72.3 range has the highest number).
- Fewer movies have high votes, showing a right-skewed distribution.
- Gradual decline in movies as votes increase, but a slight rise in the last bin suggests a few highly popular movies.

- Outliers removed for balance, showing the middle 50% of data.
- Insight: Most movies don’t get many votes, but a few dominate. Useful for recommendations or marketing focus.

Conclusion

The analysis of movie vote counts revealed a highly skewed distribution, where most movies receive low vote counts, while a few receive significantly more. By removing outliers and binning the data, we observed that the middle 50% of movies are distributed unevenly, with the majority having low votes and only a few receiving higher engagement.

This insight is valuable for understanding audience engagement—most movies struggle to gain widespread attention, while a small fraction dominates. Such data can help in recommendation systems, marketing strategies, or content curation, focusing efforts on movies with higher engagement potential.