Import library

In [129…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Read csv

In [130…
```
df = pd.read_csv("mymoviedb.csv",lineterminator='\n')
df.head()
```

Out[130…

	Release_Date	Title	Overview	Popularity	Vote_Count	Vote_Average	Origin
0	2021-12-15	Spider-Man: No Way Home	Peter Parker is unmasked and no longer able to...	5083.954	8940	8.3	
1	2022-03-01	The Batman	In his second year of fighting crime, Batman u...	3827.658	1151	8.1	
2	2022-02-25	No Exit	Stranded at a rest stop in the mountains durin...	2618.087	122	6.3	
3	2021-11-24	Encanto	The tale of an extraordinary family, the Madri...	2402.201	5076	7.7	
4	2021-12-22	The King's Man	As a collection of history's worst tyrants and...	1895.511	1793	7.0	

In [131…
```
# dataset info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9827 entries, 0 to 9826
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Release_Date       9827 non-null   object 
 1   Title              9827 non-null   object 
 2   Overview           9827 non-null   object 
 3   Popularity         9827 non-null   float64
 4   Vote_Count         9827 non-null   int64  
 5   Vote_Average       9827 non-null   float64
 6   Original_Language  9827 non-null   object 
 7   Genre              9827 non-null   object 
 8   Poster_Url         9827 non-null   object 
dtypes: float64(2), int64(1), object(6)
memory usage: 691.1+ KB
```

DataFrame Overview

- **Rows:** 9,827 entries
- **Columns:** 9 total
- **Column details:**
  - `Release_Date` → object (string, needs conversion to datetime for analysis)
  - `Title` → object (movie titles)
  - `Overview` → object (text descriptions)
  - `Popularity` → float64 (numeric score)
  - `Vote_Count` → int64 (number of votes)
  - `Vote_Average` → float64 (average rating)
  - `Original_Language` → object (ISO language codes)
  - `Genre` → object (categorical, may need splitting if multiple genres)
  - `Poster_Url` → object (links to images)

Notes

- Majority of columns are **object type** → consider preprocessing (e.g., encoding, parsing dates).
- No missing values reported → dataset is **complete**.
- Numeric columns (`Popularity` , `Vote_Count` , `Vote_Average`) are ready for statistical analysis.

In [132…
```
df.duplicated(subset=["Title", "Release_Date"]).sum()
```

Out[132…
```
np.int64(0)
```

our dataset has no duplicated rows

In [133…
```
# exploring summary statistics
df.describe()
```

Out[133…

	Popularity	Vote_Count	Vote_Average
count	9827.000000	9827.000000	9827.000000
mean	40.326088	1392.805536	6.439534
std	108.873998	2611.206907	1.129759
min	13.354000	0.000000	0.000000
25%	16.128500	146.000000	5.900000
50%	21.199000	444.000000	6.500000
75%	35.191500	1376.000000	7.100000
max	5083.954000	31077.000000	10.000000

In [134…
```
df["Genre"].head()
```

Out[134…
```
0    Action, Adventure, Science Fiction
1               Crime, Mystery, Thriller
2                               Thriller
3    Animation, Comedy, Family, Fantasy
4      Action, Adventure, Thriller, War
Name: Genre, dtype: object
```

genres are seprated by commas followed by whitespaces

- This column represents **multi-label categories** (movies can belong to multiple genres).
- For analysis, consider:
  - Splitting values into lists using `.str.split(", ")`.
  - Creating a **multi-hot encoded matrix** (one column per genre).
  - Normalizing genre names (remove extra spaces, ensure consistent casing).
- Useful for tasks like:
  - Genre distribution analysis
  - Recommendation systems
  - Filtering/searching movies by genre

Exploration Summary

- Rows: 9,827 | Columns: 9
- No missing values
- Release_Date → needs datetime conversion
- Genre → multi-label categorical (comma-separated)
- Numeric columns ready for analysis: Popularity, Vote_Count, Vote_Average

Data Cleaning

In [135…
```
# Release_Date convert to datetime
df['Release_Date'] = pd.to_datetime(df['Release_Date'], errors='coerce')

# confirming changes
print(df['Release_Date'].dtypes)
```

```
datetime64[ns]
```

In [136…
```
# Droping Poster_Url and Overview columns
df = df.drop(columns=['Poster_Url', 'Overview'], errors='ignore')
df.head()
```

Out[136…

	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Original_Language
0	2021-12-15	Spider-Man: No Way Home	5083.954	8940	8.3	en
1	2022-03-01	The Batman	3827.658	1151	8.1	en
2	2022-02-25	No Exit	2618.087	122	6.3	en
3	2021-11-24	Encanto	2402.201	5076	7.7	en
4	2021-12-22	The King's Man	1895.511	1793	7.0	en

In [ ]:
```

```

==========

START OF EXPLORATORY DATA ANALYSIS (EDA)

==========

In [ ]:
```

```

Popularity Analysis

In [137…
```
# Top 10 Most Popular Movies
df.sort_values("Popularity", ascending=False).head(10)
```

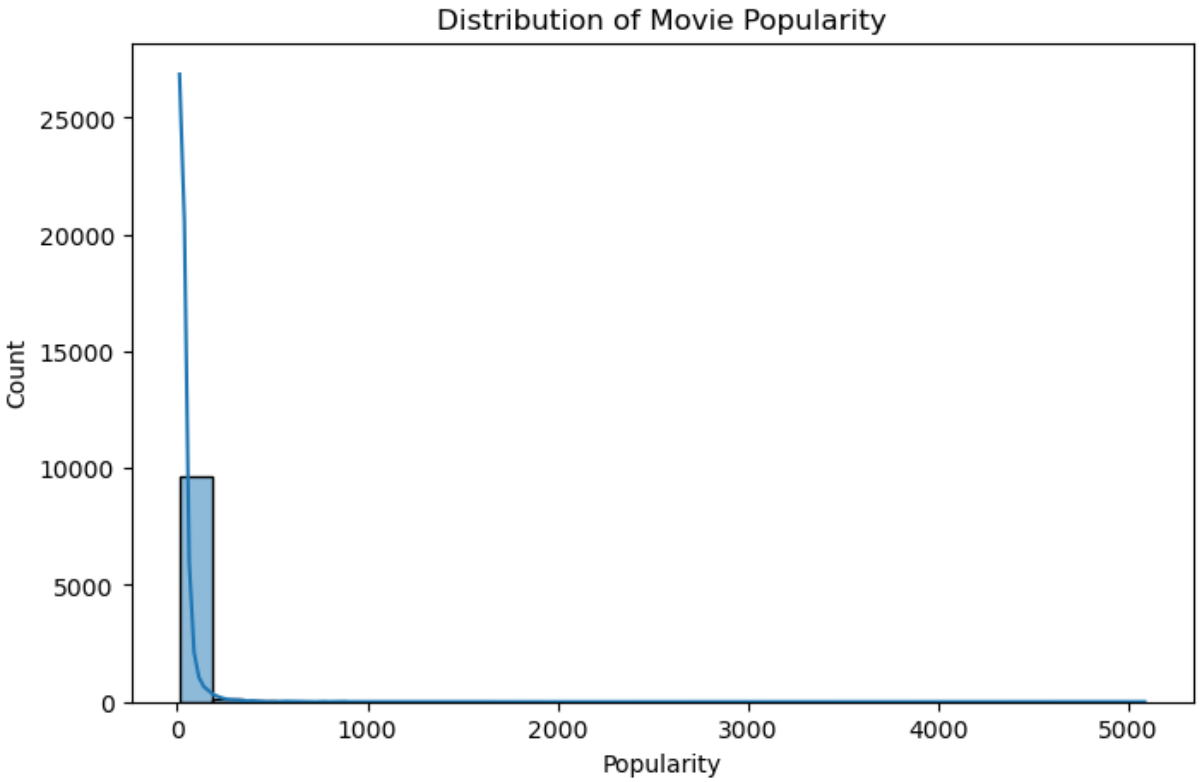
Out[137…

	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Original_Languag
0	2021-12-15	Spider-Man: No Way Home	5083.954	8940	8.3	e
1	2022-03-01	The Batman	3827.658	1151	8.1	e
2	2022-02-25	No Exit	2618.087	122	6.3	e
3	2021-11-24	Encanto	2402.201	5076	7.7	e
4	2021-12-22	The King's Man	1895.511	1793	7.0	e
5	2022-01-07	The Commando	1750.484	33	6.6	e
6	2022-01-12	Scream	1675.161	821	6.8	e
7	2022-02-10	Kimi	1601.782	206	6.3	e
8	2022-02-17	Fistful of Vengeance	1594.013	114	5.3	e
9	2021-11-03	Eternals	1537.406	4726	7.2	e

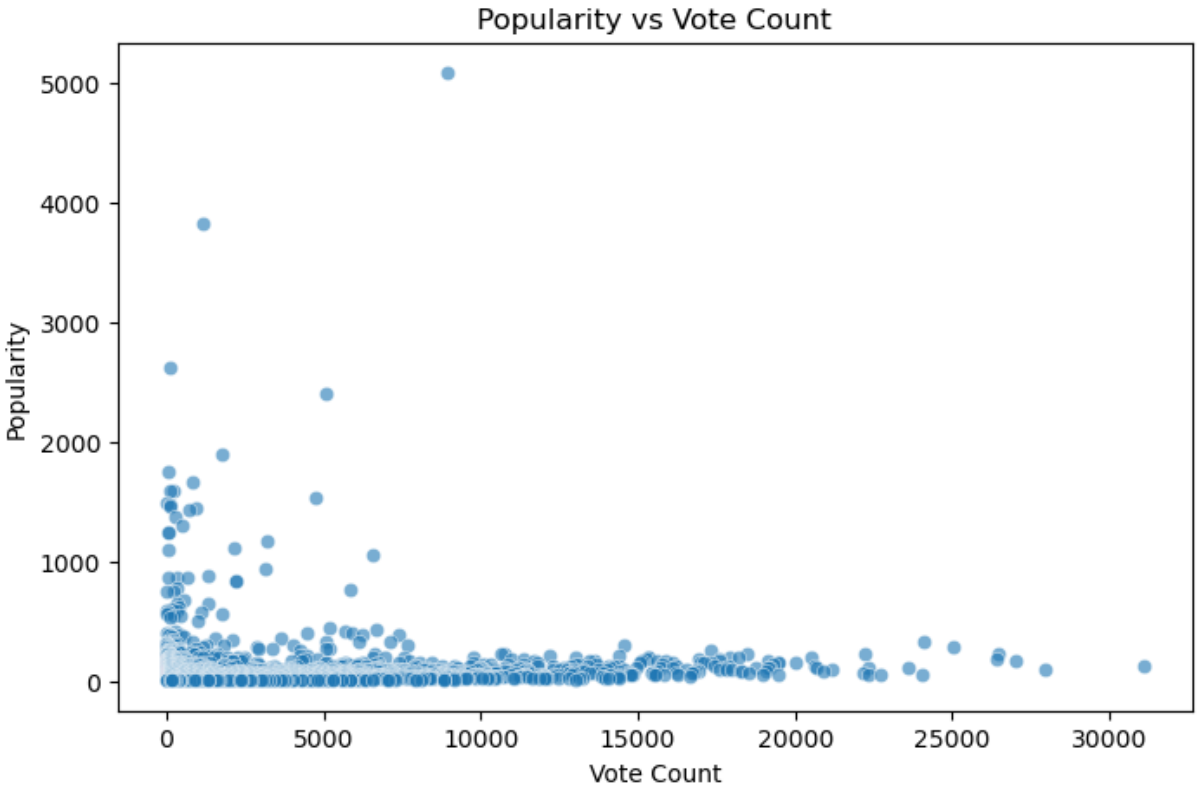
The top 10 most popular movies are dominated by big-budget and franchise films such as Spider-Man: No Way Home and The Batman. These movies have very high popularity scores along with a large number of votes, indicating strong audience engagement and widespread reach.

In [138…
```
# Popularity Distribution
plt.figure(figsize=(8,5))
sns.histplot(df["Popularity"], bins=30, kde=True)
plt.title("Distribution of Movie Popularity")
plt.xlabel("Popularity")
```

```
plt.ylabel("Count")
plt.show()
```

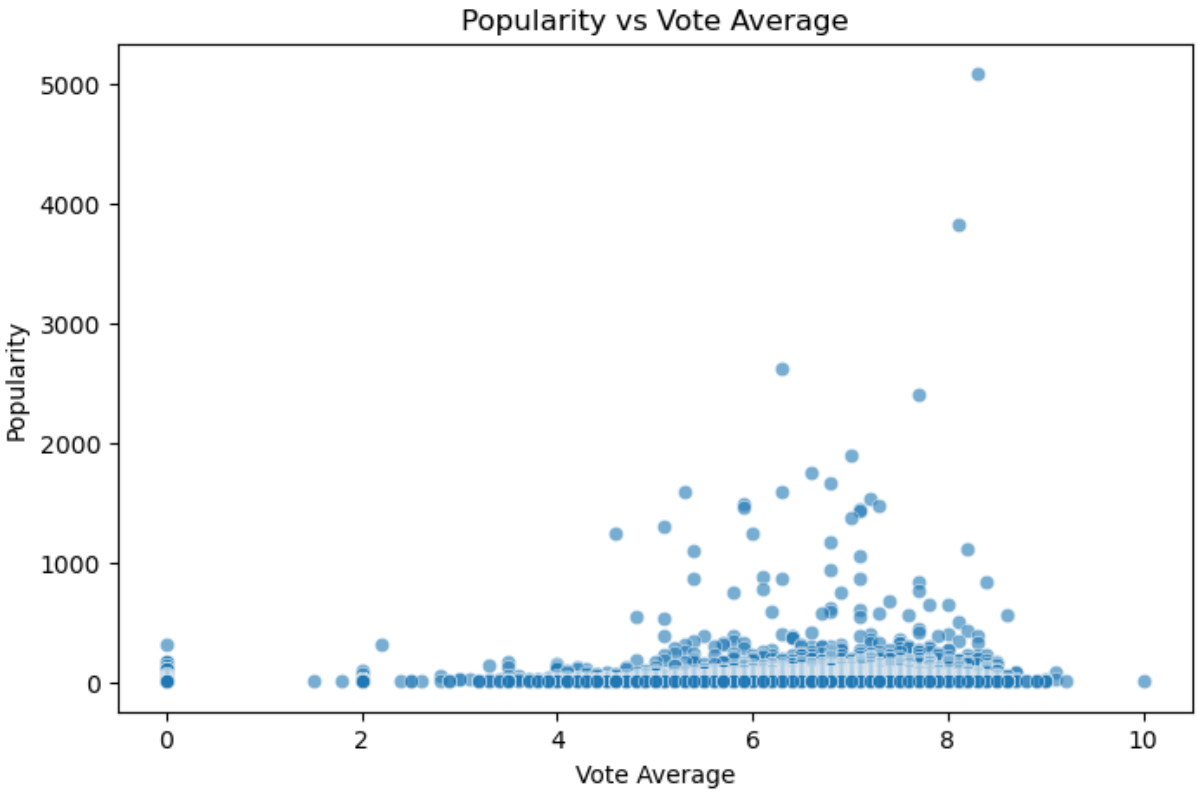


In [139…
```
# Popularity vs Vote Count
plt.figure(figsize=(8,5))
sns.scatterplot(data=df, x="Vote_Count", y="Popularity", alpha=0.6)
plt.title("Popularity vs Vote Count")
plt.xlabel("Vote Count")
plt.ylabel("Popularity")
plt.show()
```



In [140…

```
# Popularity vs Vote Average
plt.figure(figsize=(8,5))
sns.scatterplot(data=df, x="Vote_Average", y="Popularity", alpha=0.6)
plt.title("Popularity vs Vote Average")
plt.xlabel("Vote Average")
plt.ylabel("Popularity")
plt.show()
```



Create Rating Category Column

In [141…
```
def rating_category(rating):
    if rating < 5:
        return "Not Popular"
    elif rating < 6.5:
        return "Below Average"
    elif rating < 7.5:
        return "Average"
    else:
        return "Popular"

df["Rating_Category"] = df["Vote_Average"].apply(rating_category)
```

In [142…
```
df["Rating_Category"].value_counts()
```

Out[142…
```
Rating_Category
Below Average    3945
Average          3847
Popular          1442
Not Popular       593
Name: count, dtype: int64
```

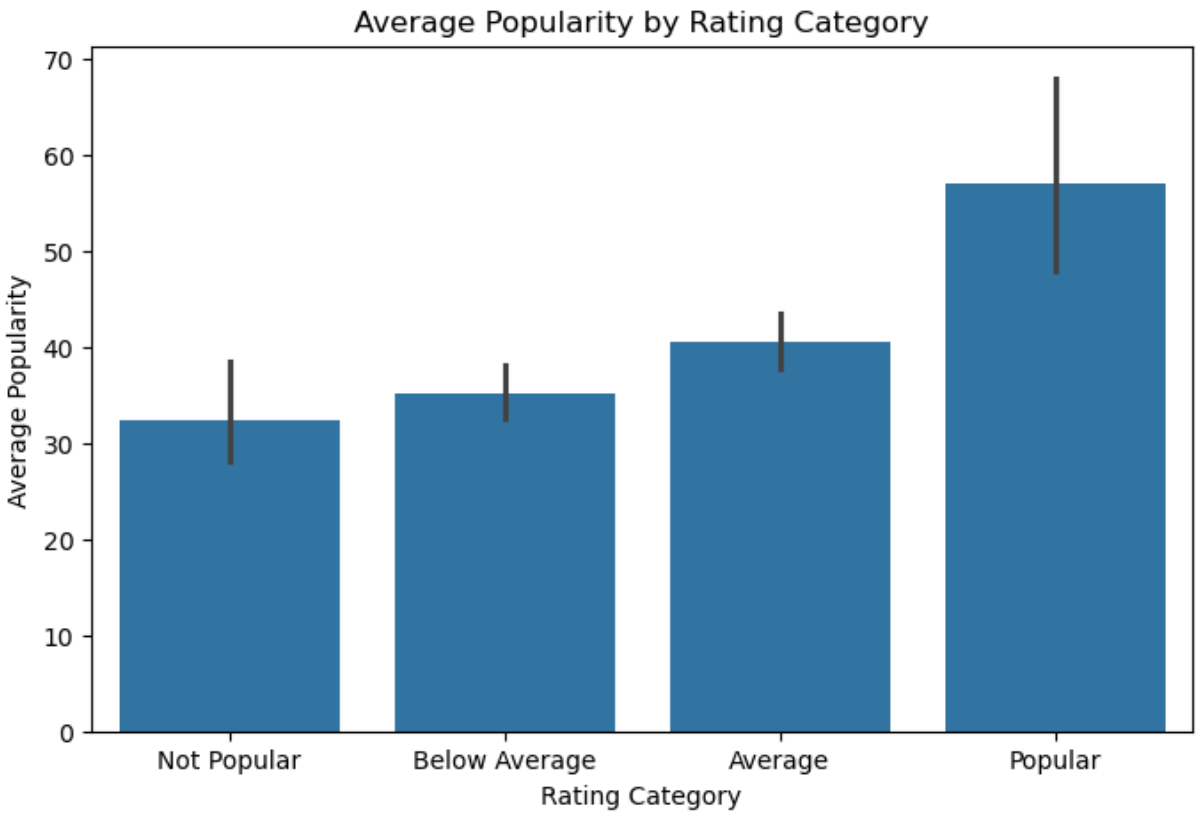
Most movies fall into the Below Average and Average categories, indicating that the majority of films receive moderate audience ratings. Only a smaller proportion of movies are classified as Popular, while Not Popular movies form the smallest group.

In [143…
```
df.groupby("Rating_Category")["Popularity"].mean()
```

Out[143…
```
Rating_Category
Average          40.596661
Below Average    35.191237
Not Popular      32.314492
Popular          56.946736
Name: Popularity, dtype: float64
```

Movies in the Popular category have the highest average popularity, followed by Average and Below Average movies. This shows a clear positive relationship between audience ratings and movie popularity, although a higher rating does not guarantee extreme popularity.

In [144…
```
plt.figure(figsize=(8,5))
sns.barplot(
    data=df,
    x="Rating_Category",
    y="Popularity",
    order=["Not Popular", "Below Average", "Average", "Popular"]
)
plt.title("Average Popularity by Rating Category")
plt.xlabel("Rating Category")
plt.ylabel("Average Popularity")
plt.show()
```



Movies categorized as Popular have the highest average popularity, followed by Average movies. This confirms that higher audience ratings are generally associated with better movie visibility and engagement.

Grouping ratings into categories helps simplify performance comparison and makes the analysis easier to understand for non-technical audiences.

In [ ]:

Md Tanbir Rja

In [ ]:

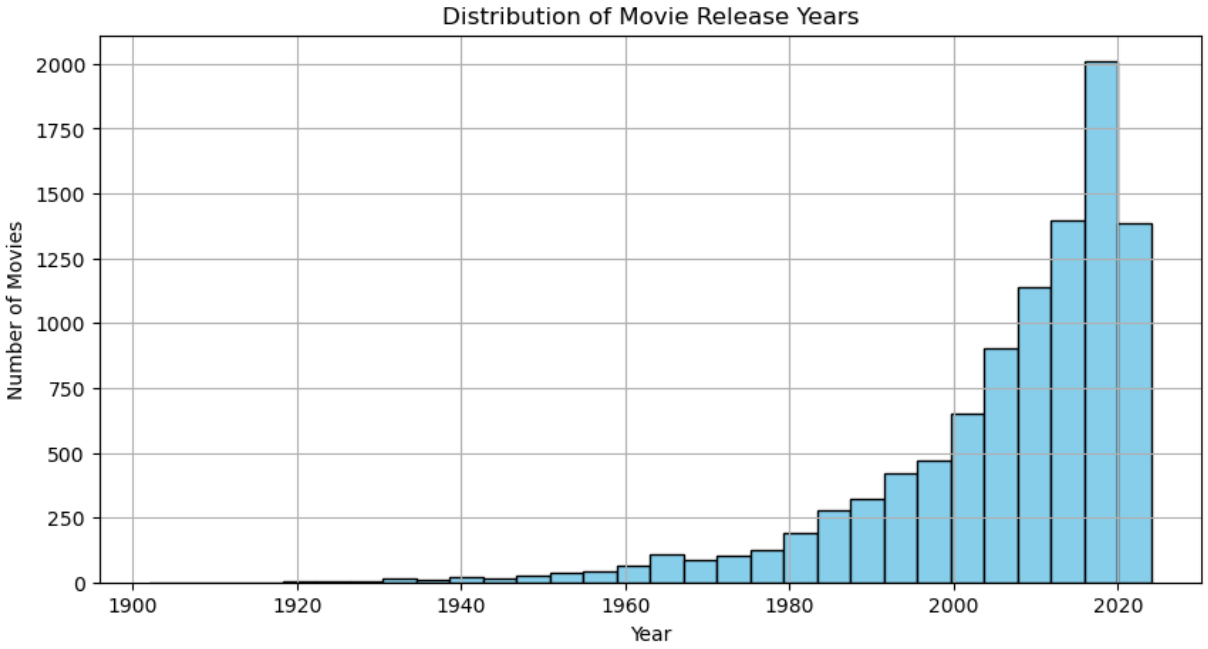
Time-Based Analysis

In [145…
```
# Time-based features
df["Year"] = df["Release_Date"].dt.year
df["Month"] = df["Release_Date"].dt.month
df["DayOfWeek"] = df["Release_Date"].dt.day_name()
df["Quarter"] = df["Release_Date"].dt.quarter
```

These features will help us understand patterns in movie releases and popularity.

In [176…
```
plt.figure(figsize=(10,5))
```

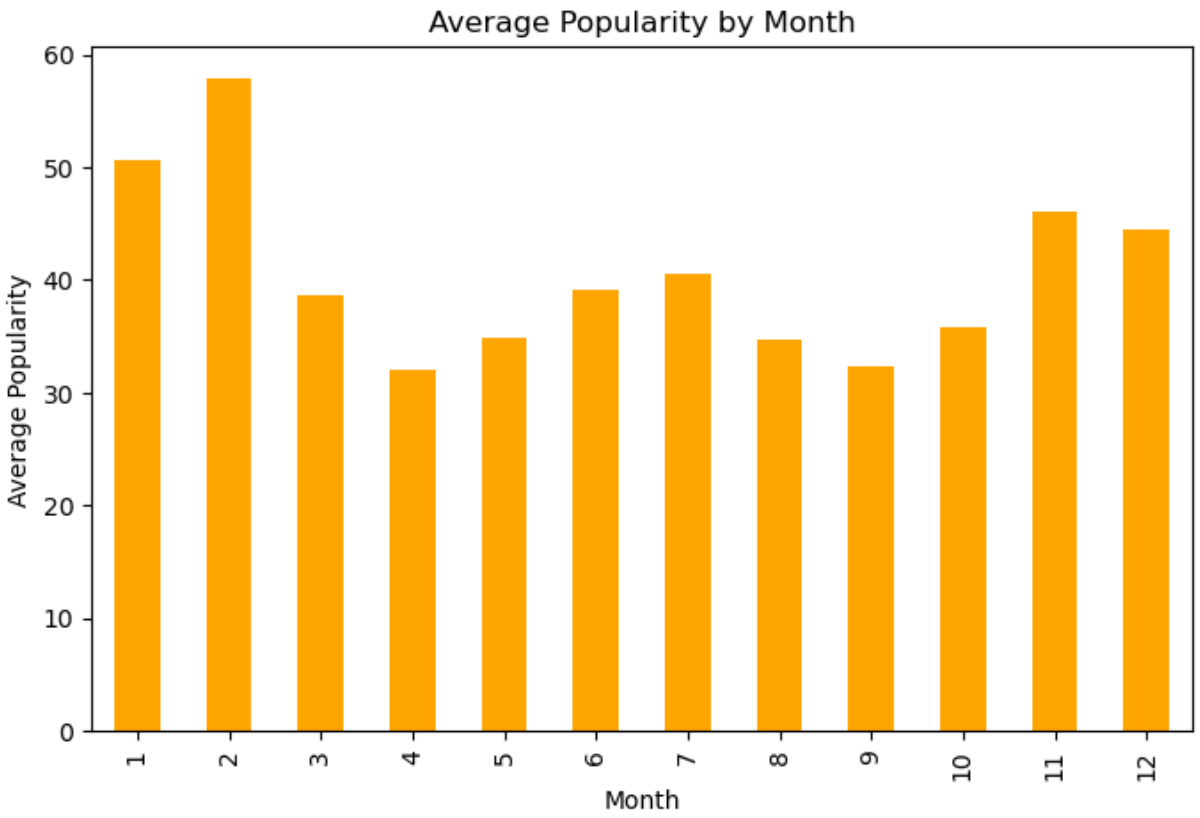
```
df["Year"].hist(bins=30, color="skyblue", edgecolor="black")
plt.title("Distribution of Movie Release Years")
plt.xlabel("Year")
plt.ylabel("Number of Movies")
plt.show()
```



The number of movies released shows a gradual growth over the years, peaking between 2000 and 2020, after which the number slightly declines. This indicates a recent surge in movie production, likely due to increased audience demand and streaming platforms.

In [153…

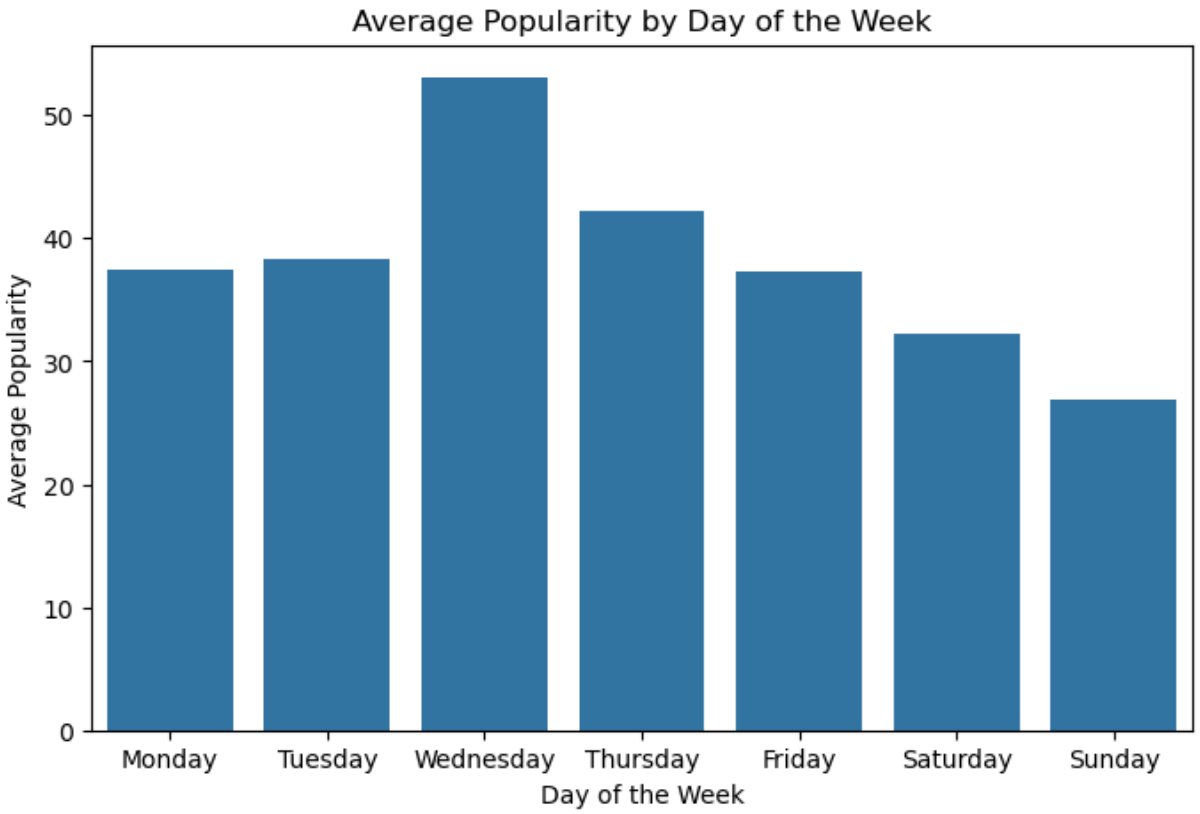
```
month_pop = df.groupby("Month")["Popularity"].mean()

plt.figure(figsize=(8,5))
month_pop.plot(kind="bar", color="orange")
plt.title("Average Popularity by Month")
plt.xlabel("Month")
plt.ylabel("Average Popularity")
plt.show()
```



Movies released in January and February show the highest average popularity, while April has the lowest. This indicates that early-year releases can perform well, possibly due to less competition or strategic blockbuster launches.

In [170…

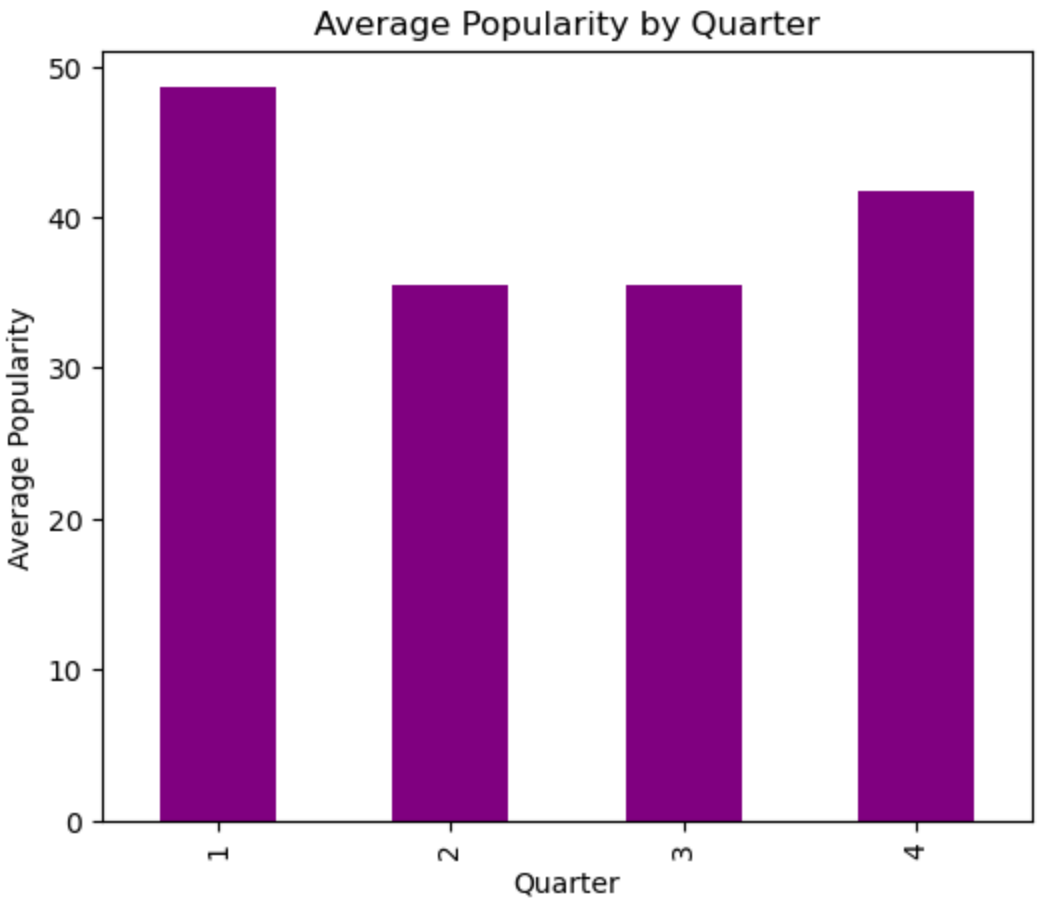
```
dow_order = ["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","

dow_pop = df.groupby("DayOfWeek")["Popularity"].mean().reindex(dow_order)

plt.figure(figsize=(8,5))
sns.barplot(x=dow_pop.index, y=dow_pop.values)
plt.title("Average Popularity by Day of the Week")
plt.xlabel("Day of the Week")
plt.ylabel("Average Popularity")
plt.show()
```



Wednesday releases have the highest average popularity, followed by Thursday. Movies released on Sunday tend to have the lowest popularity. Mid-week release strategy may boost visibility and audience engagement.

In [166…

```
quarter_pop = df.groupby("Quarter")["Popularity"].mean()

plt.figure(figsize=(6,5))
quarter_pop.plot(kind="bar", color="purple")
plt.title("Average Popularity by Quarter")
plt.xlabel("Quarter")
plt.ylabel("Average Popularity")
plt.show()
```



Quarter 1 shows the highest average popularity, followed by Quarter 4, while Quarters 2 and 3 are relatively lower. This suggests that movie studios plan major releases during peak or strategic seasons to maximize audience reach.

Genres Analysis

In [187…

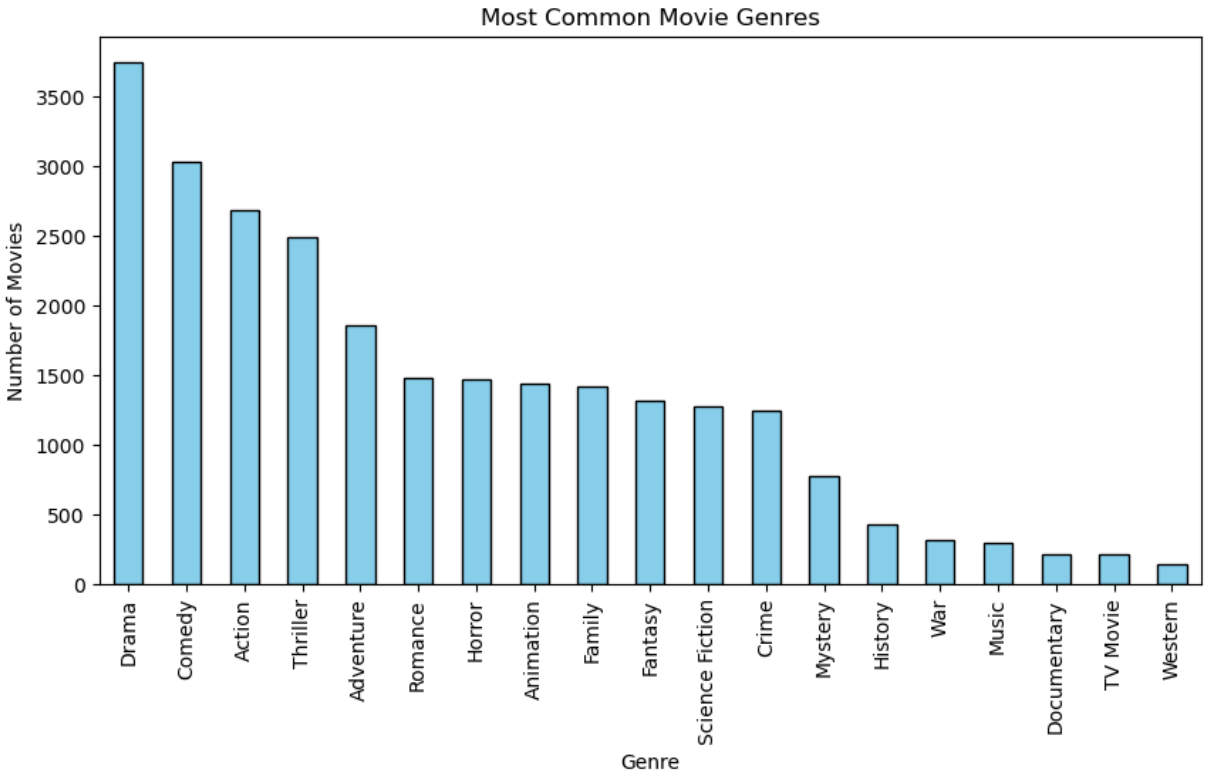
```
# Split the Genre string into a list
df["Genre"] = df["Genre"].str.split(", ")

# Explode the list so each genre gets its own row
df = df.explode("Genre").reset_index(drop=True)

# check
df.head()
```

Out[187…

	index	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Original_Lang
0	0	2021-12-15	Spider-Man: No Way Home	5083.954	8940	8.3	
1	0	2021-12-15	Spider-Man: No Way Home	5083.954	8940	8.3	
2	0	2021-12-15	Spider-Man: No Way Home	5083.954	8940	8.3	
3	1	2022-03-01	The Batman	3827.658	1151	8.1	
4	1	2022-03-01	The Batman	3827.658	1151	8.1	

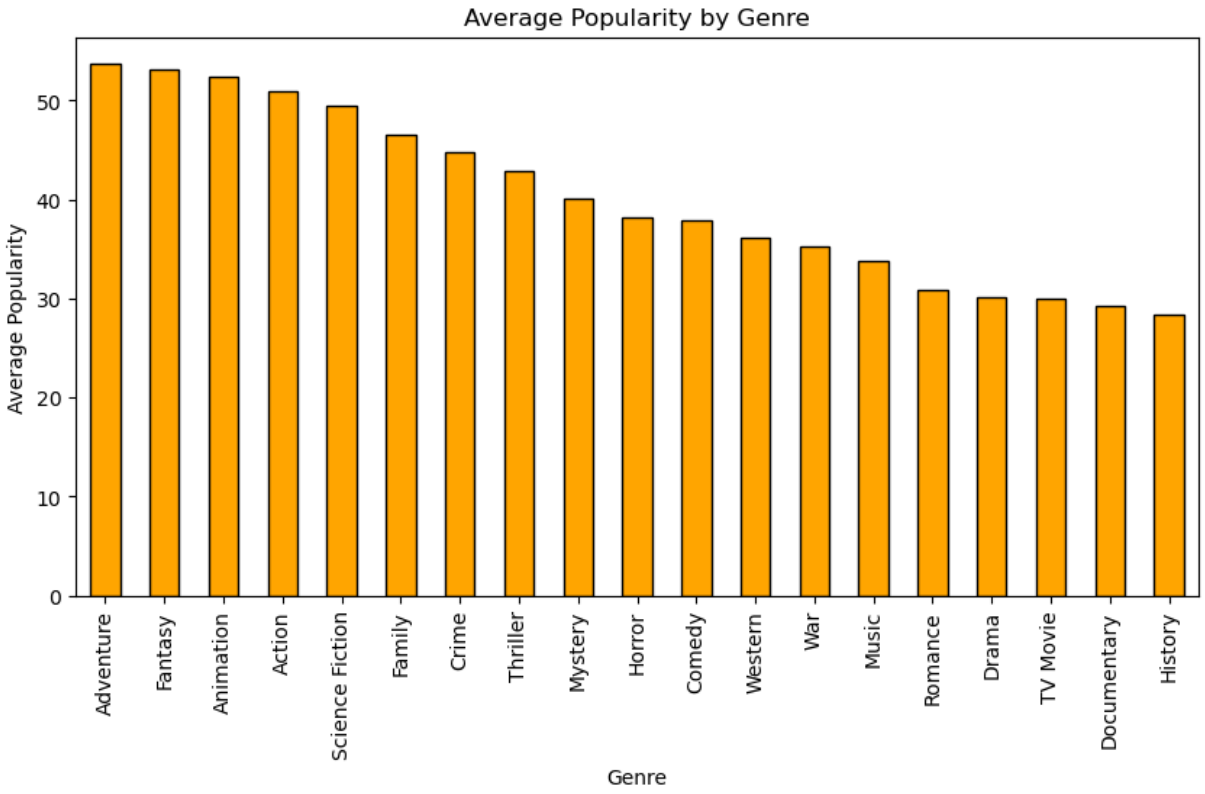
In [188…

```
genre_counts = df["Genre"].value_counts()

plt.figure(figsize=(10,5))
genre_counts.plot(kind="bar", color="skyblue", edgecolor="black")
plt.title("Most Common Movie Genres")
plt.xlabel("Genre")
plt.ylabel("Number of Movies")
plt.show()
```



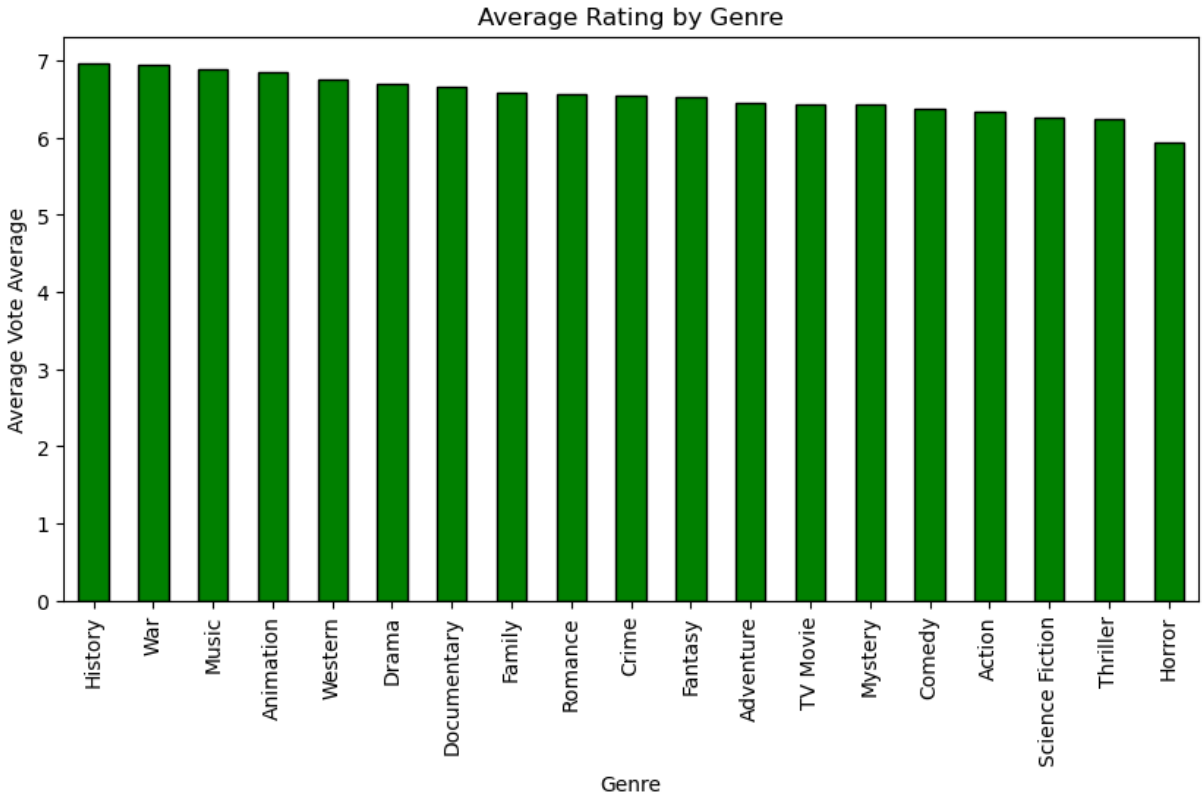
The most common genres are Drama, and Comedy, indicating these dominate movie production. Less frequent genres like Western or Documentary appear rarely.

In [190…
```
genre_popularity = df.groupby("Genre")["Popularity"].mean().sort_values(asce

plt.figure(figsize=(10,5))
genre_popularity.plot(kind="bar", color="orange", edgecolor="black")
plt.title("Average Popularity by Genre")
plt.xlabel("Genre")
plt.ylabel("Average Popularity")
plt.show()
```



Genres like Adventure, and Fantasy have the highest average popularity, showing that audiences are more engaged with high-energy and visually appealing films.

In [191…
```
genre_rating = df.groupby("Genre")["Vote_Average"].mean().sort_values(ascend

plt.figure(figsize=(10,5))
genre_rating.plot(kind="bar", color="green", edgecolor="black")
plt.title("Average Rating by Genre")
plt.xlabel("Genre")
plt.ylabel("Average Vote Average")
plt.show()
```



Genres such as History, War, and Misic have relatively higher average ratings, suggesting that while they may not be the most numerous or popular, they are well-received by audiences.

In [ ]:

Final Insights & Business Conclusions

This project analyzed **9,827 movies** to understand how popularity, ratings, release timing, and genres influence audience engagement.

Key Insights

Popularity & Ratings

- Movies with higher audience ratings generally achieve higher popularity.
- However, high ratings alone do not guarantee extreme popularity.
- Blockbuster films dominate popularity due to large vote counts and visibility.

Rating Categories

- Majority of movies fall under **Below Average** and **Average** categories.

- Only a small fraction of movies are classified as **Popular**.
- Popular movies have the highest average popularity, confirming a positive relationship between ratings and audience reach.

Time-Based Trends

- Movie releases peaked between **2000–2020**, indicating rapid industry growth.
- **Quarter 1 and Quarter 4** releases show higher average popularity.
- Movies released mid-week (especially Wednesday–Thursday) perform better in terms of popularity.

Genre Insights

- **Drama and Comedy** dominate movie production volume.
- **Adventure and Fantasy** genres achieve the highest average popularity.
- **History, War, and Music** genres receive higher average ratings despite lower popularity.

Business Implications

- Studios can maximize visibility by targeting **early-year and Q4 releases**.
- High-engagement genres like **Adventure & Fantasy** are ideal for large-scale releases.
- Critically appreciated genres may benefit from targeted niche marketing.

Conclusion

This analysis demonstrates a complete **end-to-end EDA workflow**: from data cleaning and feature engineering to visualization and insight generation.

The project highlights the ability to:

- Understand raw data structure
- Identify meaningful transformations
- Derive actionable insights using Python and visualization libraries