# IMDb Movie Rating Analysis

# Author: Meet Vaghamshi

# Objective: Explore and analyze movie ratings from IMDb dataset

In [5]:
```
import os
os.listdir()
```

Out[5]:
```
['.ipynb_checkpoints', 'tmdb_5000_movies.csv', 'Untitled.ipynb']
```

In [9]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Optional: To see plots in the notebook
%matplotlib inline

sns.set(style="whitegrid")
```

In [10]:
```
df = pd.read_csv("tmdb_5000_movies.csv")  # Replace with your filename
df.head()
```

Out[10]:

| | budget | genres | homepage | id | keyword |
|---|---|---|---|---|---|
| **0** | 237000000 | [{"id": 28, "name": "Action"}, {"id": 12, "nam... | http://www.avatarmovie.com/ | 19995 | [{"id 146: "name "cultur clash" {"id": |
| **1** | 300000000 | [{"id": 12, "name": "Adventure"}, {"id": 14, "... | http://disney.go.com/disneypictures/pirates/ | 285 | [{"id 27( "name "ocean" {"id": 72( "na |
| **2** | 245000000 | [{"id": 28, "name": "Action"}, {"id": 12, "nam... | http://www.sonypictures.com/movies/spectre/ | 206647 | [{"id 47( "name "spy" {"id": 81( "name |
| **3** | 250000000 | [{"id": 28, "name": "Action"}, {"id": 80, "nam... | http://www.thedarkknightrises.com/ | 49026 | [{"id 84( "name "d comics" {"id 853, |
| **4** | 260000000 | [{"id": 28, "name": "Action"}, {"id": 12, "nam... | http://movies.disney.com/john-carter | 49529 | [{"id 81( "name "based o novel" {"id": |

In [11]:

```
df.info()
df.describe()
df.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   budget                4803 non-null   int64  
 1   genres                4803 non-null   object 
 2   homepage              1712 non-null   object 
 3   id                    4803 non-null   int64  
 4   keywords              4803 non-null   object 
 5   original_language     4803 non-null   object 
 6   original_title        4803 non-null   object 
 7   overview              4800 non-null   object 
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object 
 10  production_countries  4803 non-null   object 
 11  release_date          4802 non-null   object 
 12  revenue               4803 non-null   int64  
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object 
 15  status                4803 non-null   object 
 16  tagline               3959 non-null   object 
 17  title                 4803 non-null   object 
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64  
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

Out[11]:

```
Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'original_language',
       'original_title', 'overview', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'vote_average',
       'vote_count'],
      dtype='object')
```

In [13]:

```python
# Check for missing values
df.isnull().sum()

# Drop rows with missing release_date or vote_average
df.dropna(subset=['release_date', 'vote_average'], inplace=True)

# Convert release_date to datetime
df['release_date'] = pd.to_datetime(df['release_date'], errors='coerce')

# Create a release_year column
df['release_year'] = df['release_date'].dt.year
```

In [14]:

```python
# Top 10 movies by rating
df[['title', 'vote_average']].sort_values(by='vote_average', ascending=False).he

# Average rating
df['vote_average'].mean()
```

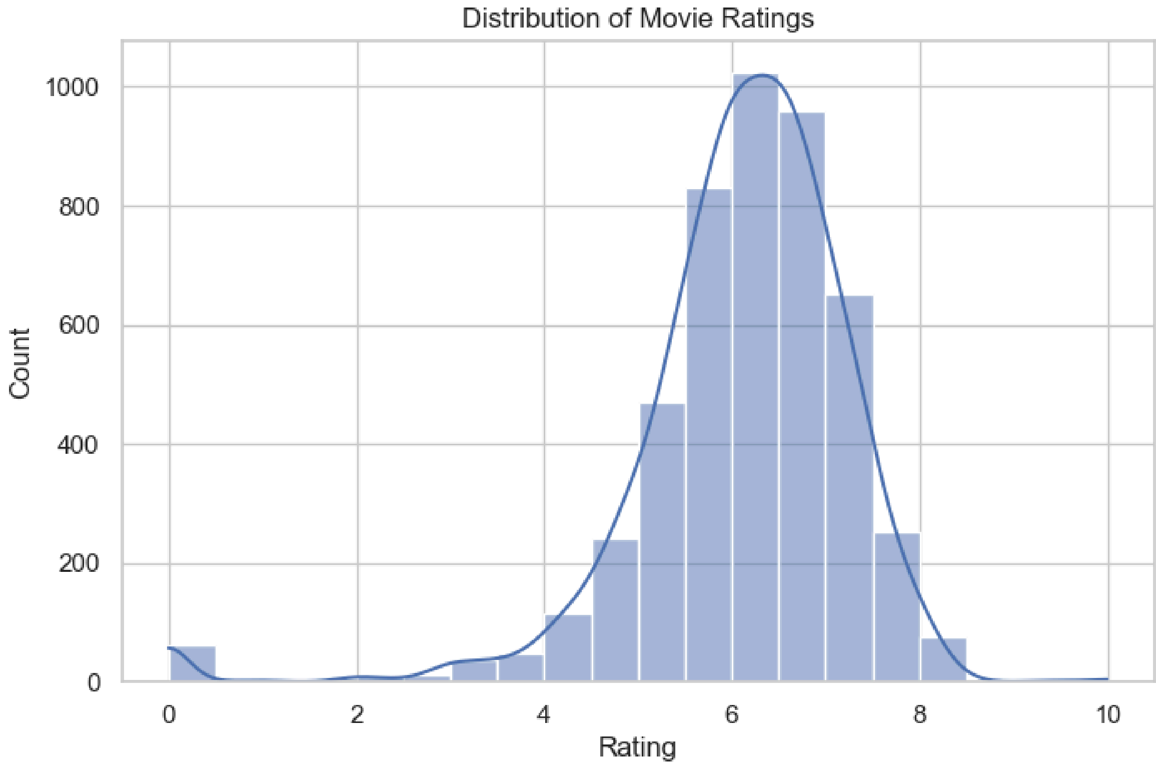
Out[14]:

```
6.0934402332361515
```

In [15]:

```python
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
sns.histplot(df['vote_average'], bins=20, kde=True)
```

```
plt.title('Distribution of Movie Ratings')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```



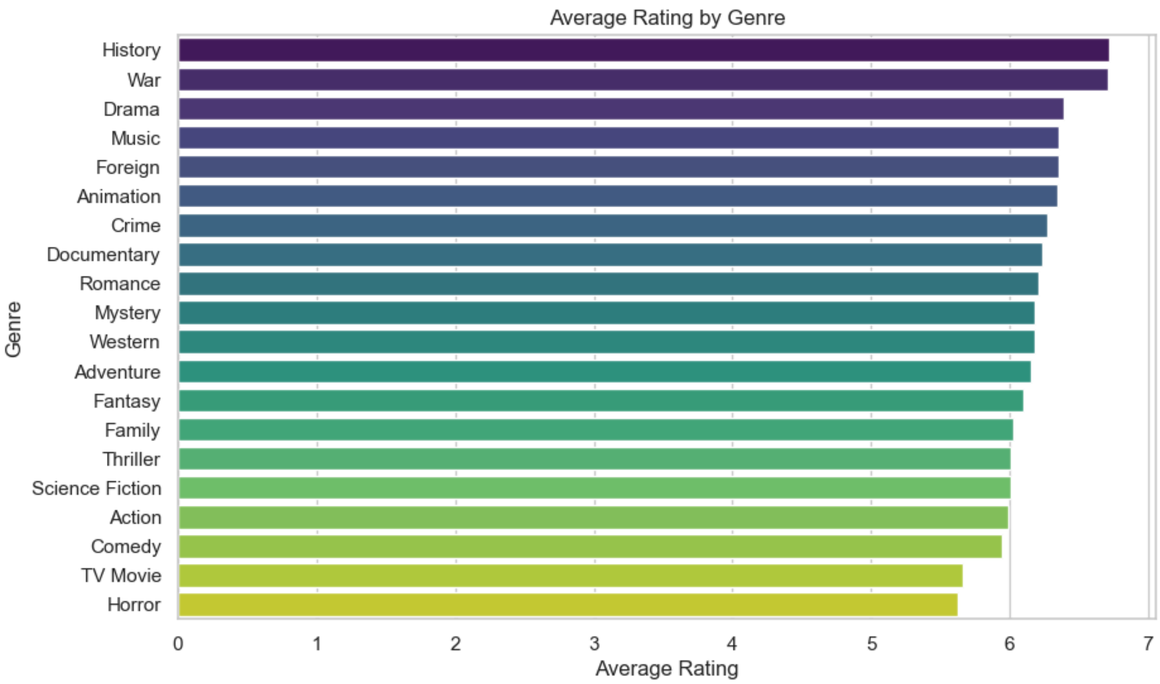

In [16]:
```
import ast

# Convert stringified list to list of dicts
df['genres'] = df['genres'].apply(lambda x: [i['name'] for i in ast.literal_eval

# Create a copy with one genre per row
genre_df = df.explode('genres')

# Average rating per genre
genre_rating = genre_df.groupby('genres')['vote_average'].mean().sort_values(asc
genre_rating
```

Out[16]:

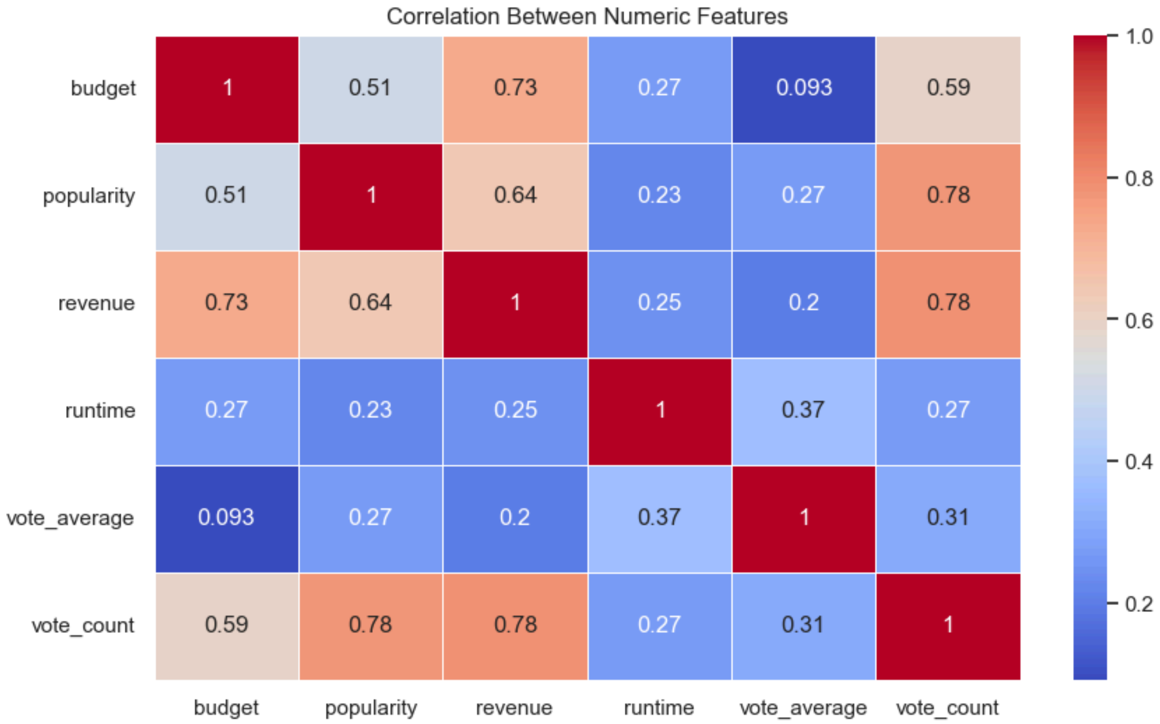
```
genres
History            6.719797
War                6.713889
Drama              6.388594
Music              6.355676
Foreign            6.352941
Animation          6.341453
Crime              6.274138
Documentary        6.238182
Romance            6.207718
Mystery            6.183908
Western            6.178049
Adventure          6.156962
Fantasy            6.096698
Family             6.029630
Thriller           6.010989
Science Fiction    6.005607
Action             5.989515
Comedy             5.945587
TV Movie           5.662500
Horror             5.626590
Name: vote_average, dtype: float64
```

In [20]:

```python
plt.figure(figsize=(10, 6))
sns.barplot(
    x=genre_rating.values,
    y=genre_rating.index,
    hue=genre_rating.index,  # Add hue to apply palette correctly
    palette='viridis',
    dodge=False,             # Ensure bars don't separate
    legend=False             # Hide legend since hue is just the index
)
plt.title('Average Rating by Genre')
plt.xlabel('Average Rating')
plt.ylabel('Genre')
plt.show()
```



In [18]:
```
numeric_cols = ['budget', 'popularity', 'revenue', 'runtime', 'vote_average', 'v
plt.figure(figsize=(10, 6))
sns.heatmap(df[numeric_cols].corr(), annot=True, cmap='coolwarm', linewidths=0.5
plt.title('Correlation Between Numeric Features')
plt.show()
```




## Key Insights:

- Genres like **Documentary** and **History** have higher average ratings.
- Most movies have a rating between 5 and 7.
- Budget and revenue have weak correlation with rating.
- Popularity doesn't directly translate to higher ratings.

In [ ]: