TASK 2 MOVIE RATING PREDICTION WITH PYTHON

In [2]:
```
# Step 1: Import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.preprocessing import LabelEncoder
```

In [3]:
```
df = pd.read_csv("IMDb Movies India.csv")
df.head()
```

Out[3]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	
0		NaN	NaN	Drama	NaN	NaN	J.S. Randhawa	Manmauji	
1	#Gadhvi (He thought he was Gandhi)	-2019.0	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Gha
2	#Homecoming	-2021.0	90 min	Drama, Musical	NaN	NaN	Soumyajit Majumdar	Sayani Gupta	Bo
3	#Yaaram	-2019.0	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Is
4	...And Once Again	-2010.0	105 min	Drama	NaN	NaN	Amol Palekar	Rajat Kapoor	Ri Se

In [4]:
```
df = df.drop(index=0).reset_index(drop=True)
```

In [5]:
```
df.head()
```

Out[5]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Act
0	#Gadhvi (He thought he was Gandhi)	-2019.0	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	\ Gham
1	#Homecoming	-2021.0	90 min	Drama, Musical	NaN	NaN	Soumyajit Majumdar	Sayani Gupta	Pl Borth
2	#Yaaram	-2019.0	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishit
3	...And Once Again	-2010.0	105 min	Drama	NaN	NaN	Amol Palekar	Rajat Kapoor	Ritup Seng
4	...Aur Pyaar Ho Gaya	-1997.0	147 min	Comedy, Drama, Musical	4.7	827	Rahul Rawail	Bobby Deol	Aishw Bach

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15508 entries, 0 to 15507
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      15508 non-null  object
 1   Year      14981 non-null  float64
 2   Duration  7240 non-null   object
 3   Genre     13631 non-null  object
 4   Rating    7919 non-null   float64
 5   Votes     7920 non-null   object
 6   Director  14983 non-null  object
 7   Actor 1   13891 non-null  object
 8   Actor 2   13124 non-null  object
 9   Actor 3   12364 non-null  object
dtypes: float64(2), object(8)
memory usage: 1.2+ MB
```

In [6]:
```
df['Duration'] = df['Duration'].str.replace(' min', '', regex=False)
df['Duration'] = pd.to_numeric(df['Duration'], errors='coerce')
```

In [7]:
```
df['Year'] = df['Year'].astype(float) * -1  # To fix -2019 → 2019
```

In [8]:
```
df['Votes'] = pd.to_numeric(df['Votes'], errors='coerce')
df['Rating'] = pd.to_numeric(df['Rating'], errors='coerce')
```

Since we are predicting Rating, it must not be NaN:

In [9]:
```
df = df.dropna(subset=['Rating', 'Genre', 'Director', 'Actor 1'])
df = df.dropna()  # Drop any other remaining NaNs
```

In [10]:
```
df = df[['Genre', 'Director', 'Actor 1', 'Actor 2', 'Actor 3', 'Duration', 'Year
```

In [11]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
for col in ['Genre', 'Director', 'Actor 1', 'Actor 2', 'Actor 3']:
    df[col] = le.fit_transform(df[col])
```

In [12]:
```
X = df.drop('Rating', axis=1)
y = df['Rating']
```

In [13]:
```
df.head()
```

Out[13]:

	Genre	Director	Actor 1	Actor 2	Actor 3	Duration	Year	Votes	Rating
0	202	531	1207	1978	266	109.0	2019.0	8.0	7.0
2	162	1144	1065	624	1825	110.0	2019.0	35.0	4.4
4	140	1312	338	63	1741	147.0	1997.0	827.0	4.7
7	284	107	1725	1028	860	82.0	2012.0	326.0	5.6
8	30	327	270	498	409	116.0	2014.0	11.0	4.0

In [14]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

Initialize and Train Random Forest

In [16]:
```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[16]:
```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

Make Predictions and Evaluate

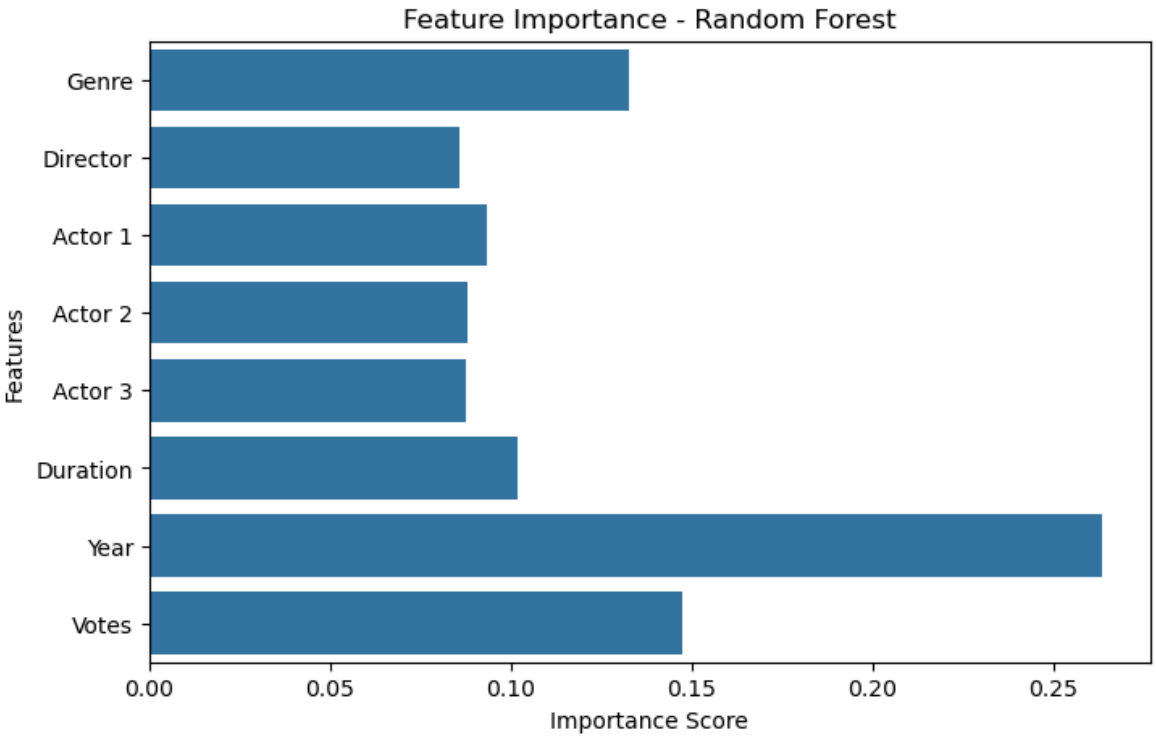
In [17]:
```
y_pred = model.predict(X_test)

print("R² Score:", r2_score(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
```

```
R² Score: 0.34073535280535805
MSE: 1.304958885913853
```

In [18]:
```
# Get feature importances
importances = model.feature_importances_
features = X.columns

# Plot
plt.figure(figsize=(8,5))
sns.barplot(x=importances, y=features)
plt.title("Feature Importance - Random Forest")
plt.xlabel("Importance Score")
plt.ylabel("Features")
plt.show()
```



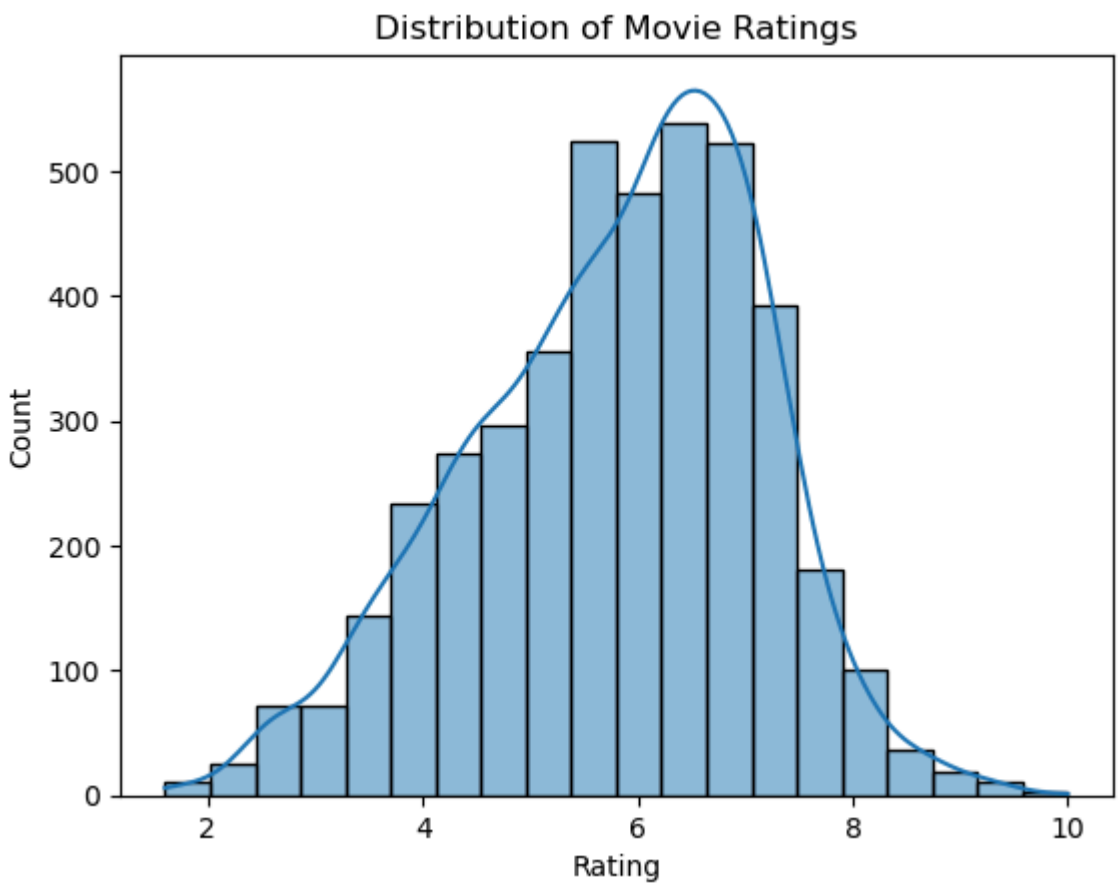
In [19]:
```
df.head()
```

Out[19]:

	Genre	Director	Actor 1	Actor 2	Actor 3	Duration	Year	Votes	Rating
0	202	531	1207	1978	266	109.0	2019.0	8.0	7.0
2	162	1144	1065	624	1825	110.0	2019.0	35.0	4.4
4	140	1312	338	63	1741	147.0	1997.0	827.0	4.7
7	284	107	1725	1028	860	82.0	2012.0	326.0	5.6
8	30	327	270	498	409	116.0	2014.0	11.0	4.0

Distribution of Ratings

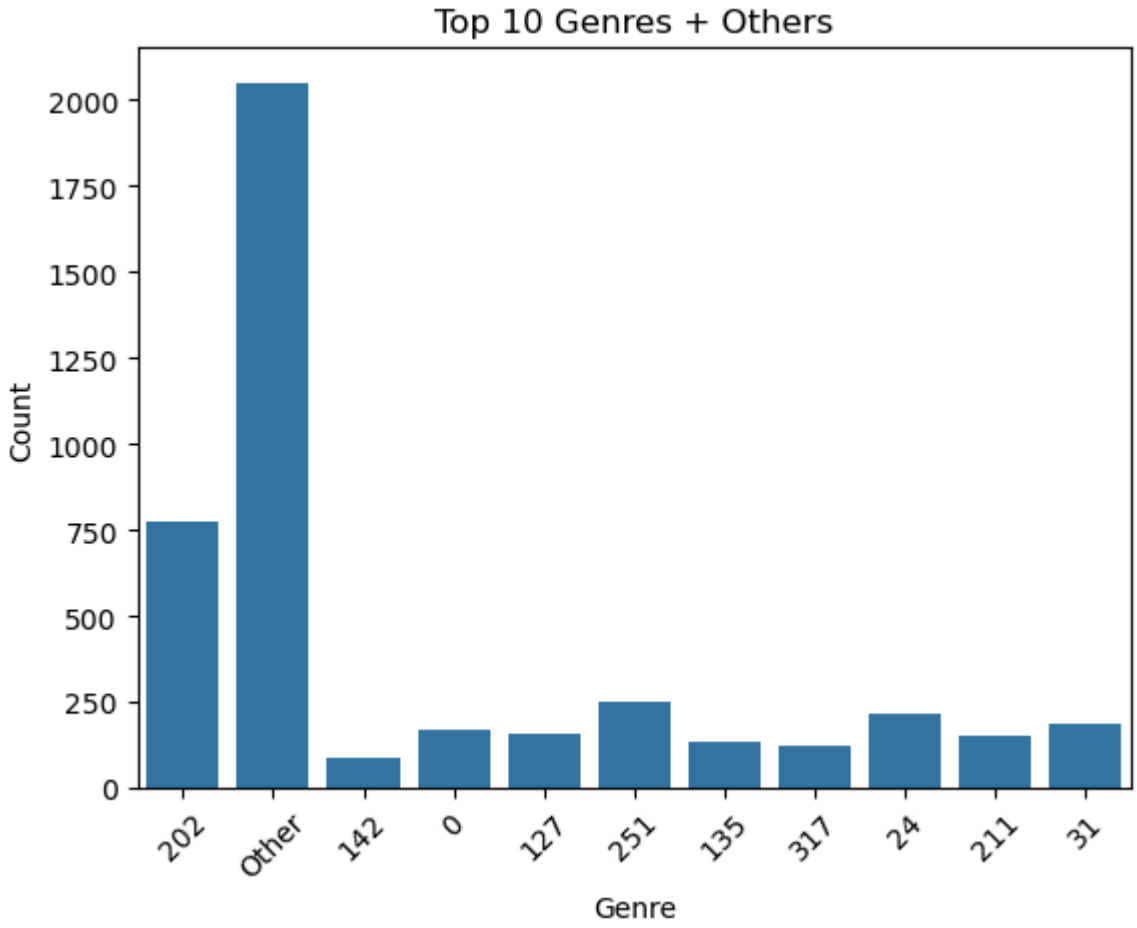
In [20]:
```
sns.histplot(df['Rating'], bins=20, kde=True)
plt.title("Distribution of Movie Ratings")
plt.xlabel("Rating")
plt.ylabel("Count")
plt.show()
```



Count of Movies per Genre

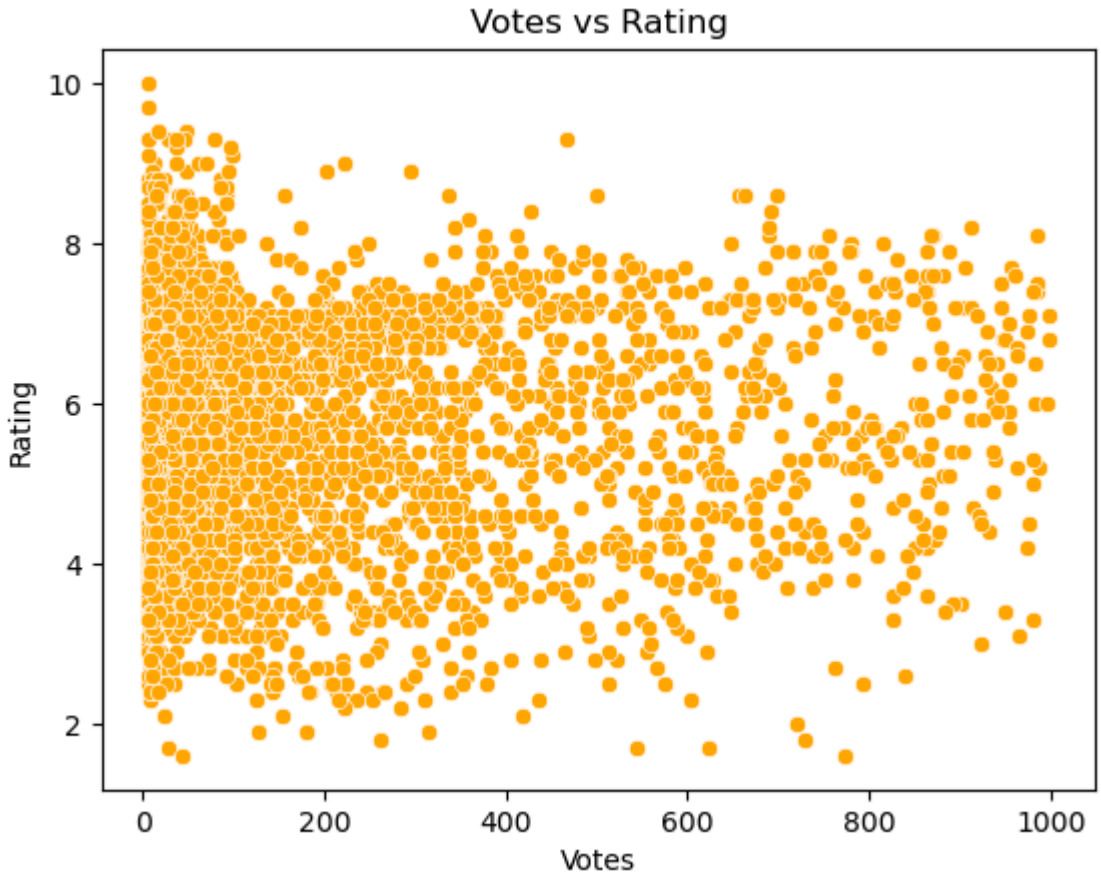
In [21]:
```
top_genres = df['Genre'].value_counts().nlargest(10).index
df['Genre_Grouped'] = df['Genre'].apply(lambda x: x if x in top_genres else 'Oth

sns.countplot(x='Genre_Grouped', data=df)
plt.title("Top 10 Genres + Others")
plt.xlabel("Genre")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



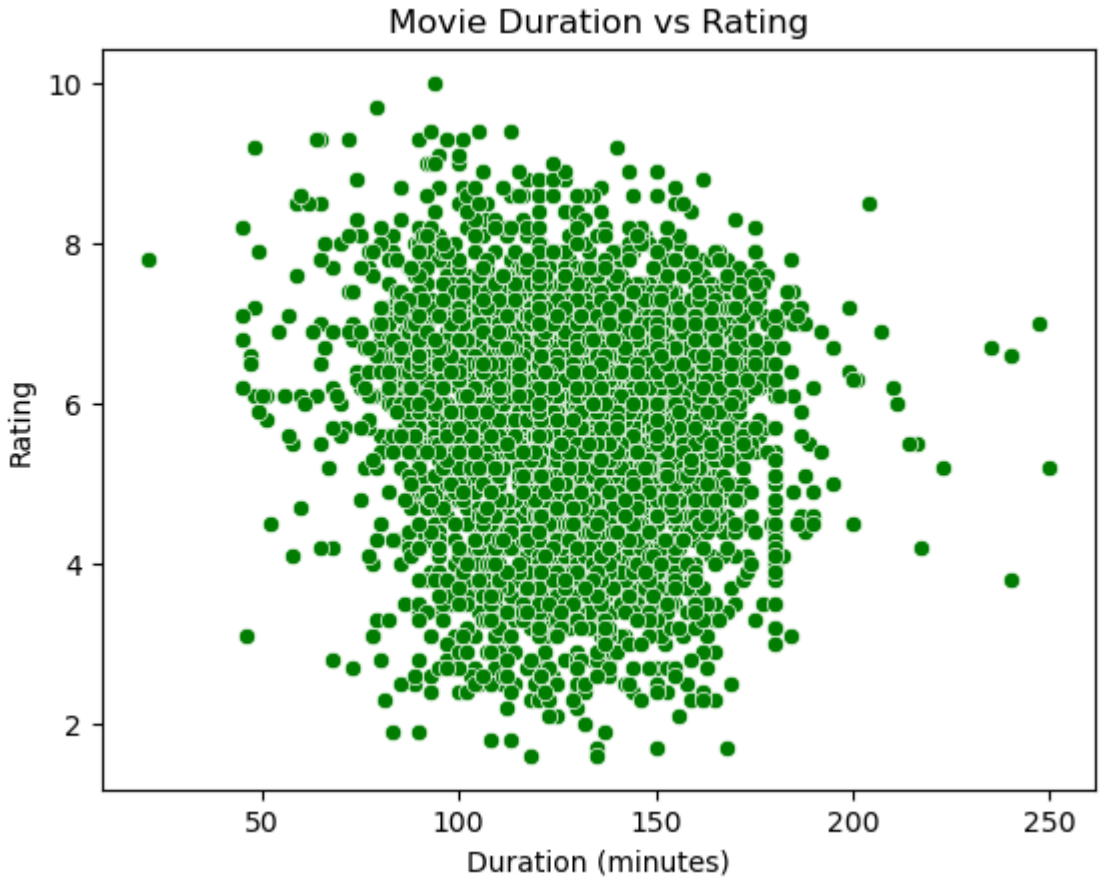
Votes vs Rating

In [22]:
```
sns.scatterplot(x='Votes', y='Rating', data=df, color='orange')
plt.title("Votes vs Rating")
plt.xlabel("Votes")
plt.ylabel("Rating")
plt.show()
```



Duration vs Rating

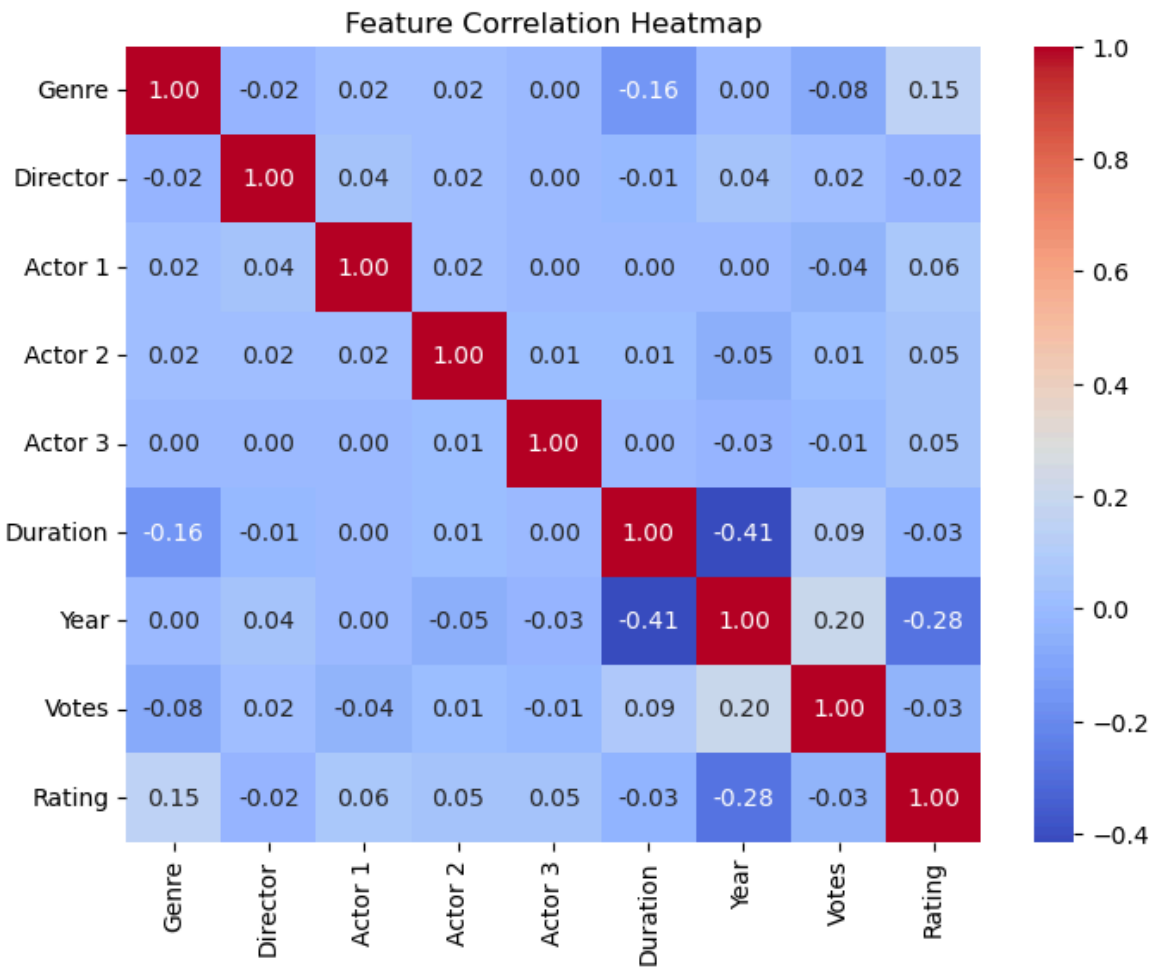
In [23]:
```
sns.scatterplot(x='Duration', y='Rating', data=df, color='green')
plt.title("Movie Duration vs Rating")
plt.xlabel("Duration (minutes)")
plt.ylabel("Rating")
plt.show()
```



Correlation Heatmap

In [24]:
```
# Only numeric columns — safe for correlation
df_numeric = df.select_dtypes(include=['number'])

plt.figure(figsize=(8, 6))
sns.heatmap(df_numeric.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Feature Correlation Heatmap")
plt.show()
```



In [25]:
```
df.to_csv("cleaned_movies.csv", index=False)
```

In [26]:
```
import joblib
joblib.dump(model, "movie_model.pkl")
```

Out[26]:
```
['movie_model.pkl']
```

In [ ]:

In [ ]: