In [28]:
```
# Movie Success Prediction + Sentiment Study

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, a

import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\tedla\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Out[28]: True

In [29]:
```
# Step 1: Load dataset
df = pd.read_csv(r"C:\Users\tedla\Downloads\movie_success_rate.csv")

# Step 2: Clean data
df.fillna(0, inplace=True)
```

In [30]:
```
# Step 3: Sentiment Analysis on Descriptions
sid = SentimentIntensityAnalyzer()
df['Sentiment_Score'] = df['Description'].astype(str).apply(lambda x: sid.polari

X_rev = df[['Runtime (Minutes)', 'Rating', 'Votes', 'Metascore', 'Sentiment_Scor
y_rev = df['Revenue']

X_train, X_test, y_train, y_test = train_test_split(X_rev, y_rev, test_size=0.2,

rf_reg = RandomForestRegressor(n_estimators=200, random_state=42)
rf_reg.fit(X_train, y_train)
y_pred_rf = rf_reg.predict(X_test)

print("📌 Revenue Prediction Results (Random Forest):")
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))
print("R² Score:", r2_score(y_test, y_pred_rf))
```

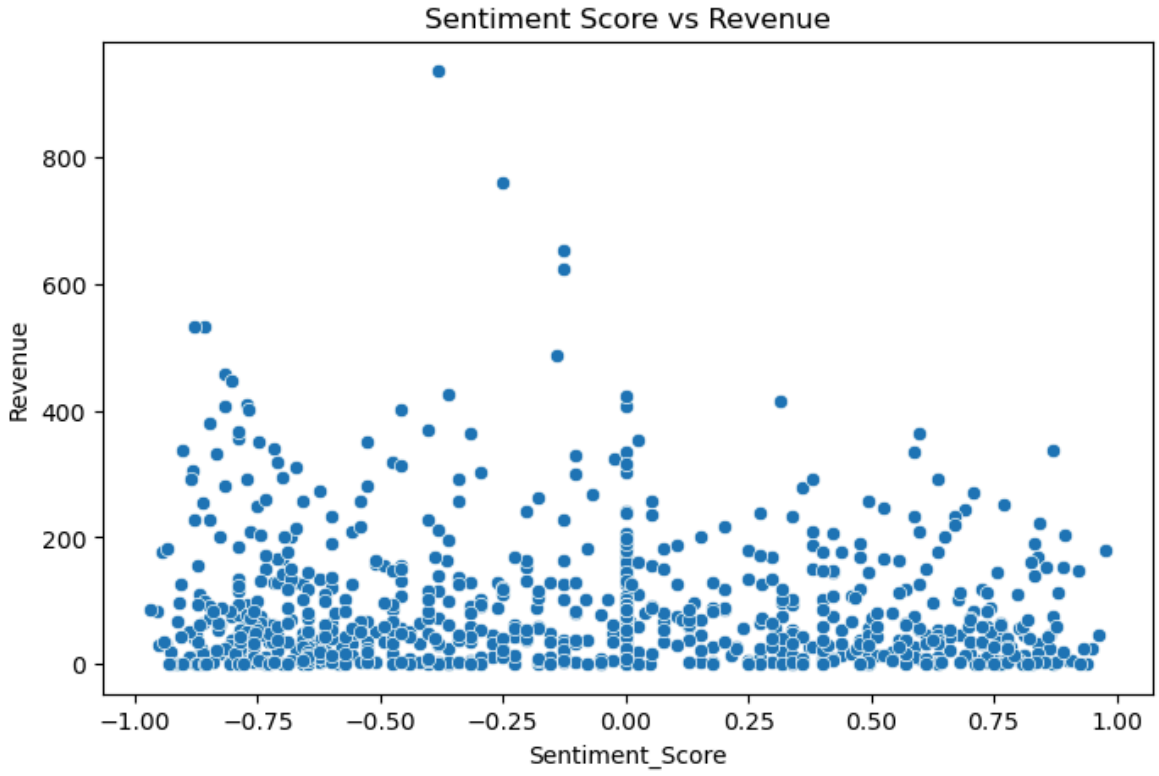
```
📌 Revenue Prediction Results (Random Forest):
MAE: 63.35047738521338
RMSE: 98.92096959971319
R² Score: 0.23740183491786437
```

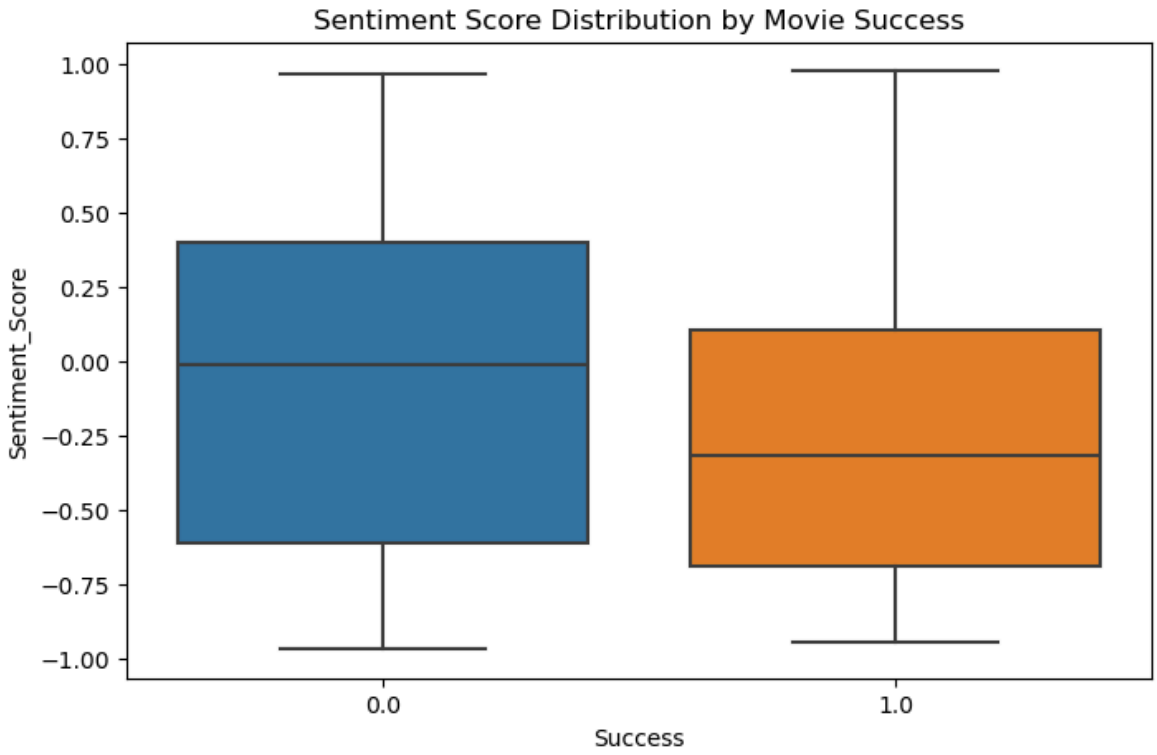
In [31]:
```
X_cls = df[['Runtime (Minutes)', 'Rating', 'Votes', 'Metascore', 'Sentiment_Scor
y_cls = df['Success']

X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X_cls, y_cls, test_s

rf_clf = RandomForestClassifier(n_estimators=200, random_state=42)
rf_clf.fit(X_train_c, y_train_c)
```

```
y_pred_c = rf_clf.predict(X_test_c)

print("\n📌 Success Classification Results (Random Forest):")
print("Accuracy:", accuracy_score(y_test_c, y_pred_c))
print(confusion_matrix(y_test_c, y_pred_c))
print(classification_report(y_test_c, y_pred_c))
```

```
📌 Success Classification Results (Random Forest):
Accuracy: 0.9285714285714286
[[132   6]
 [  6  24]]
              precision    recall  f1-score   support

         0.0       0.96      0.96      0.96       138
         1.0       0.80      0.80      0.80        30

    accuracy                           0.93       168
   macro avg       0.88      0.88      0.88       168
weighted avg       0.93      0.93      0.93       168
```

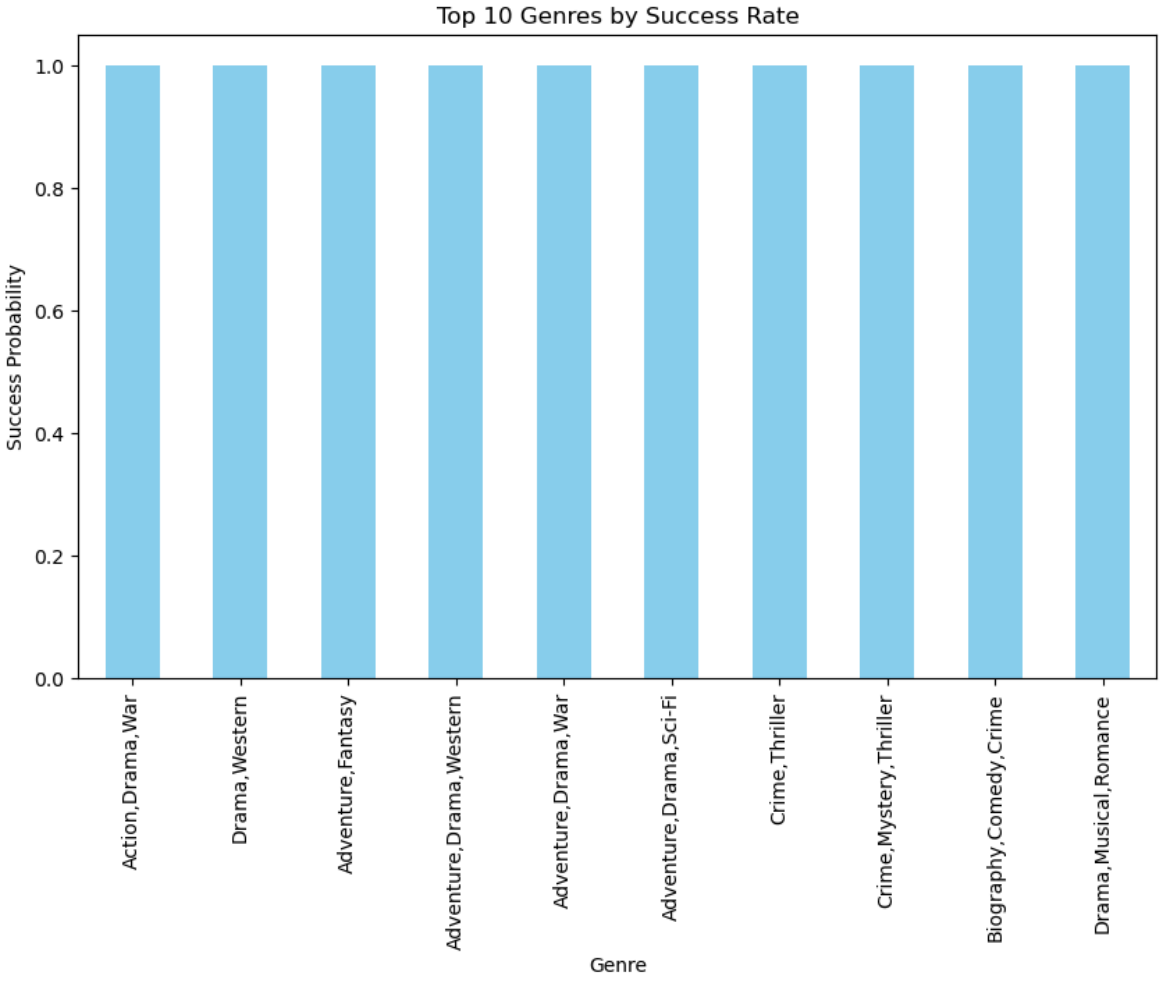
In [32]:
```
# Step 6: Visualization
plt.figure(figsize=(8,5))
sns.scatterplot(x=df['Sentiment_Score'], y=df['Revenue'])
plt.title("Sentiment Score vs Revenue")
plt.show()

plt.figure(figsize=(8,5))
sns.boxplot(x=df['Success'], y=df['Sentiment_Score'])
plt.title("Sentiment Score Distribution by Movie Success")
plt.show()
```



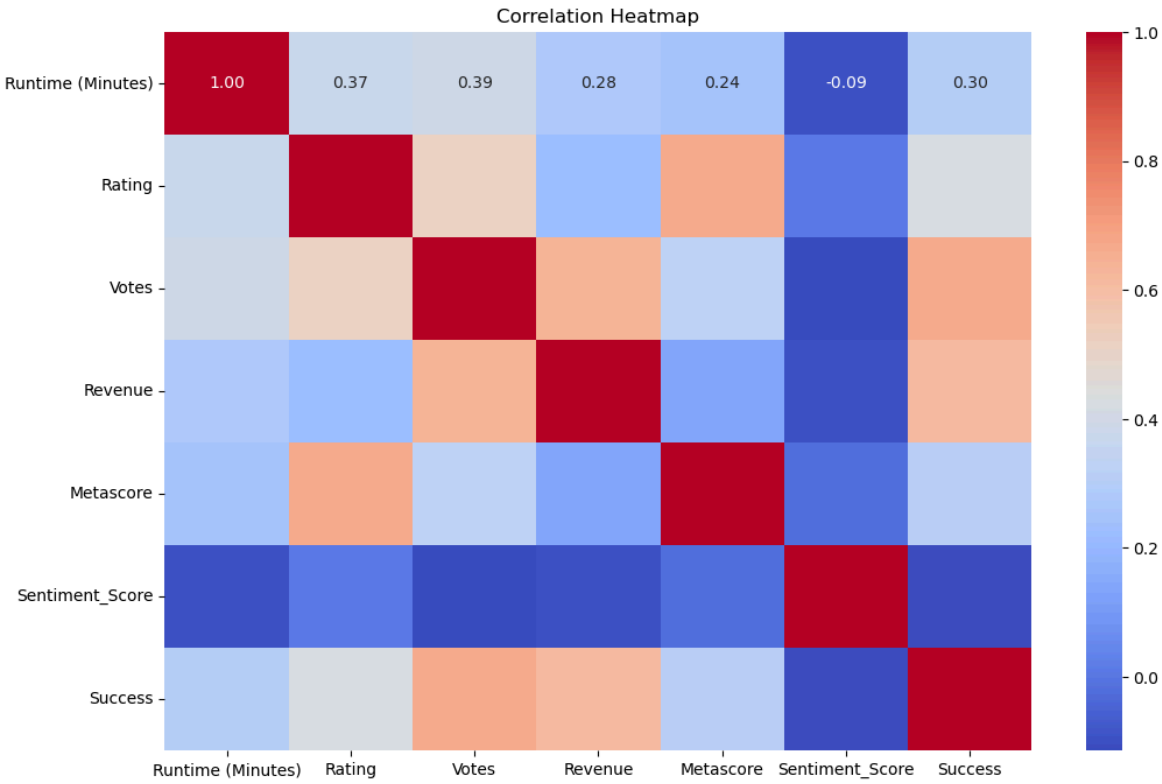


In [33]:
```
genre_success = df.groupby('Genre')['Success'].mean().sort_values(ascending=Fals
print(genre_success.head(10))

plt.figure(figsize=(10,6))
genre_success.head(10).plot(kind='bar', color='skyblue')
plt.title("Top 10 Genres by Success Rate")
plt.ylabel("Success Probability")
plt.show()
```

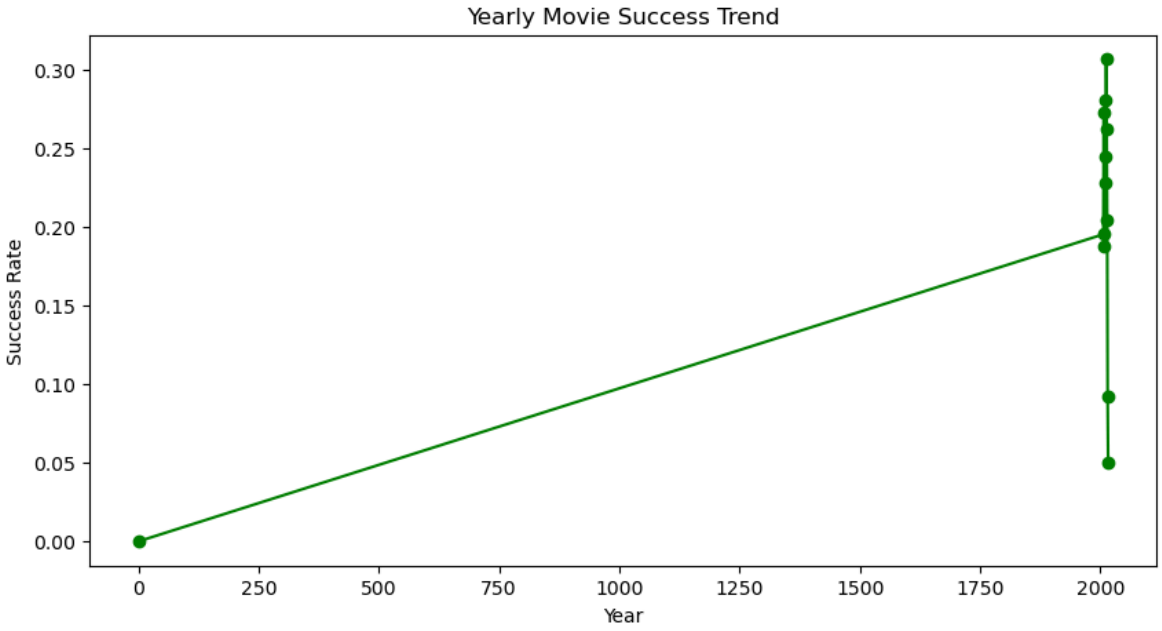
```
Genre
Action,Drama,War           1.0
Drama,Western              1.0
Adventure,Fantasy          1.0
Adventure,Drama,Western    1.0
Adventure,Drama,War        1.0
Adventure,Drama,Sci-Fi     1.0
Crime,Thriller             1.0
Crime,Mystery,Thriller     1.0
Biography,Comedy,Crime     1.0
Drama,Musical,Romance      1.0
Name: Success, dtype: float64
```



In [34]:
```
plt.figure(figsize=(12,8))
sns.heatmap(df[['Runtime (Minutes)','Rating','Votes','Revenue','Metascore','Sent
            annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

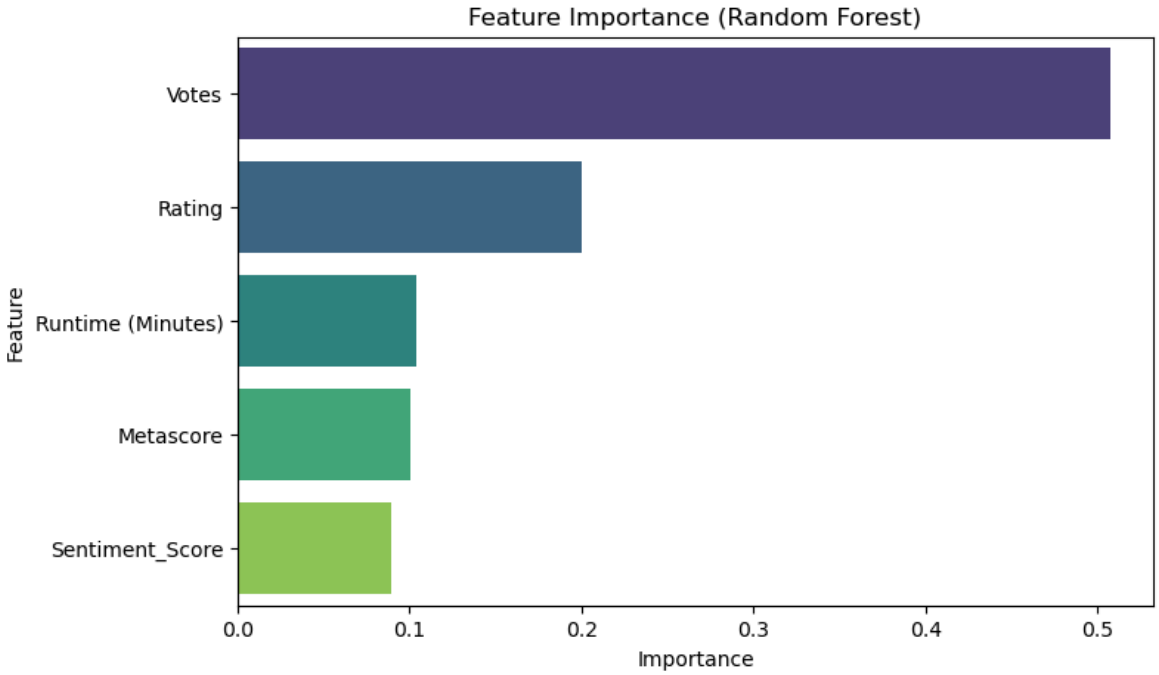


In [35]:
```
yearly = df.groupby('Year')['Success'].mean()

plt.figure(figsize=(10,5))
yearly.plot(marker='o', color='green')
plt.title("Yearly Movie Success Trend")
plt.ylabel("Success Rate")
plt.show()
```

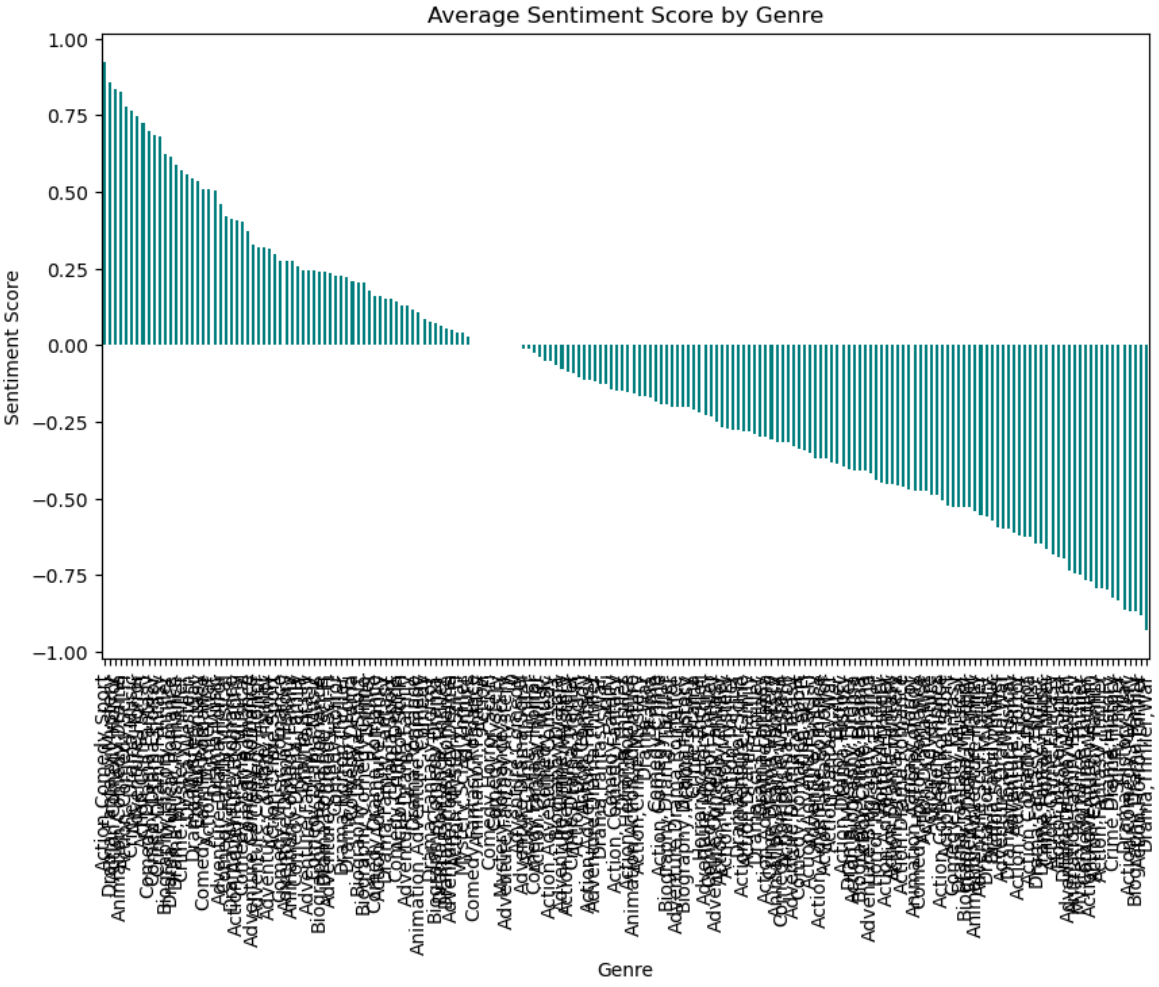


In [36]:
```
importances = rf_clf.feature_importances_
features = X_cls.columns

importance_df = pd.DataFrame({"Feature": features, "Importance": importances})
importance_df = importance_df.sort_values(by="Importance", ascending=False)

print(importance_df)

# Plot
plt.figure(figsize=(8,5))
sns.barplot(x="Importance", y="Feature", data=importance_df, palette="viridis")
plt.title("Feature Importance (Random Forest)")
plt.show()
```

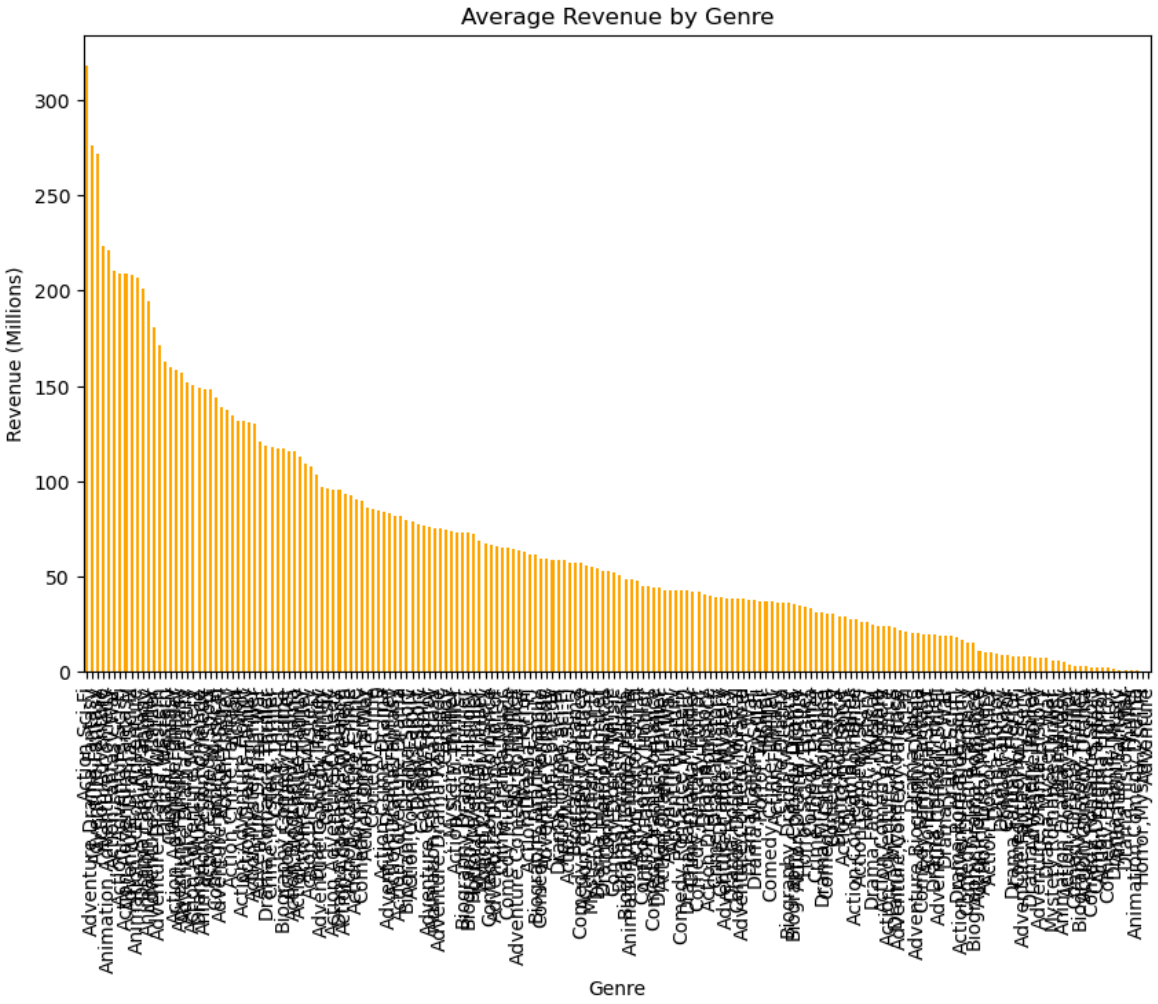
```
            Feature  Importance
2             Votes    0.507050
1            Rating    0.199716
0  Runtime (Minutes)    0.103483
3         Metascore    0.100579
4   Sentiment_Score    0.089173
```



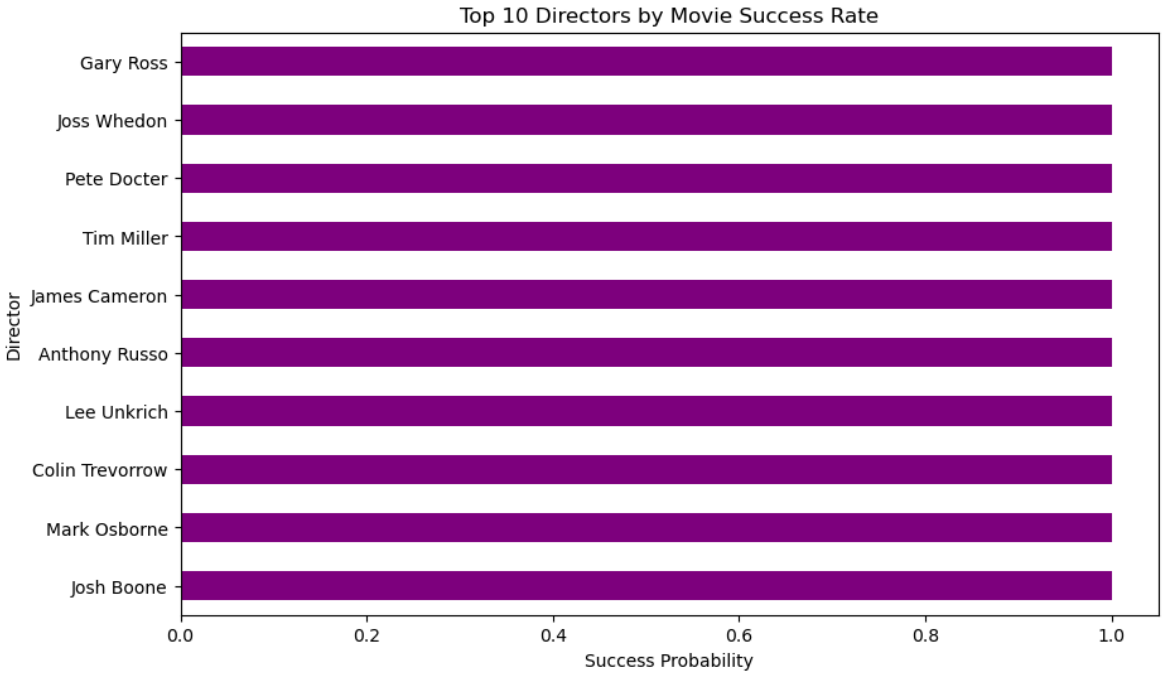
In [37]:
```
genre_sentiment = df.groupby('Genre')['Sentiment_Score'].mean().sort_values(asce

plt.figure(figsize=(10,6))
genre_sentiment.plot(kind='bar', color='teal')
plt.title("Average Sentiment Score by Genre")
plt.ylabel("Sentiment Score")
plt.show()
```



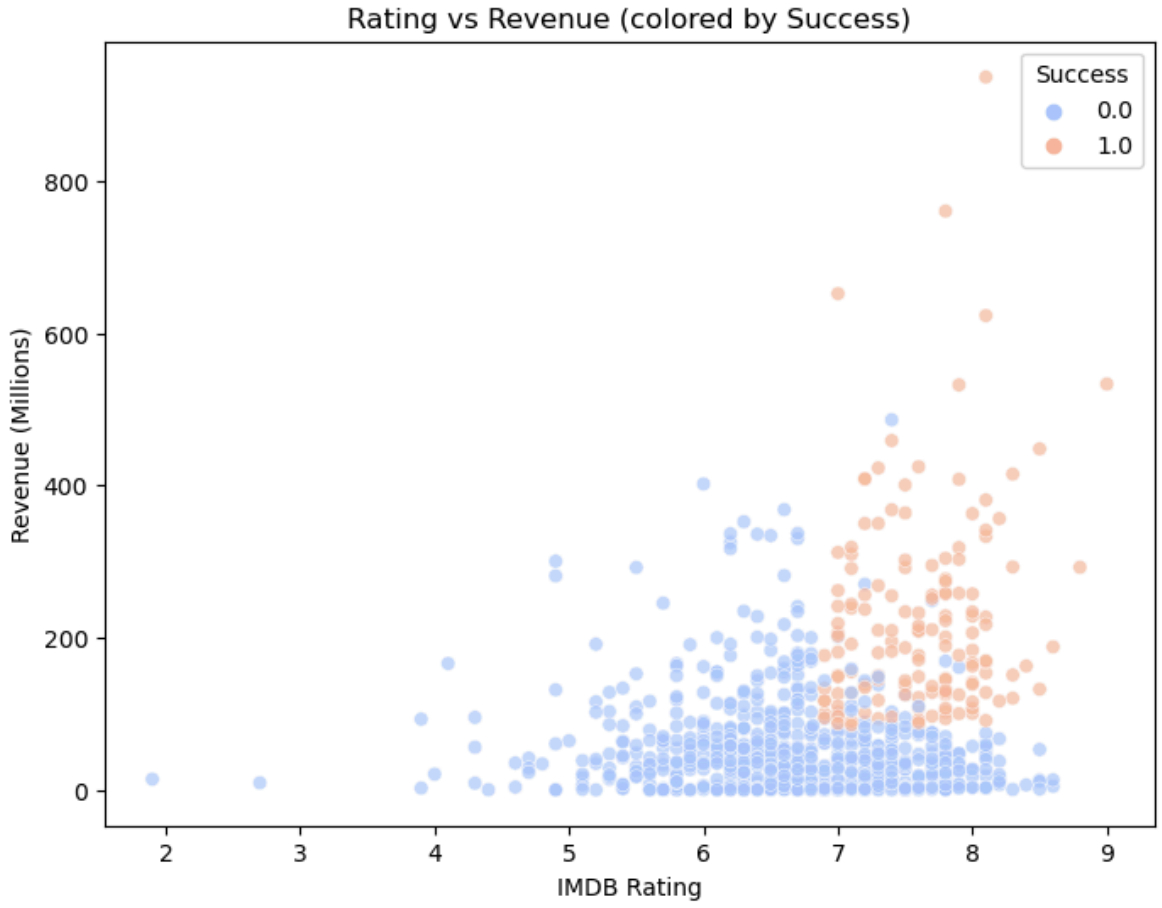
In [38]:
```
genre_revenue = df.groupby('Genre')['Revenue'].mean().sort_values(ascending=Fals

plt.figure(figsize=(10,6))
genre_revenue.plot(kind='bar', color='orange')
plt.title("Average Revenue by Genre")
plt.ylabel("Revenue (Millions)")
plt.show()
```



In [39]:
```
director_success = df.groupby('Director')['Success'].mean().sort_values(ascendin

plt.figure(figsize=(10,6))
director_success.plot(kind='barh', color='purple')
plt.title("Top 10 Directors by Movie Success Rate")
plt.xlabel("Success Probability")
plt.show()
```

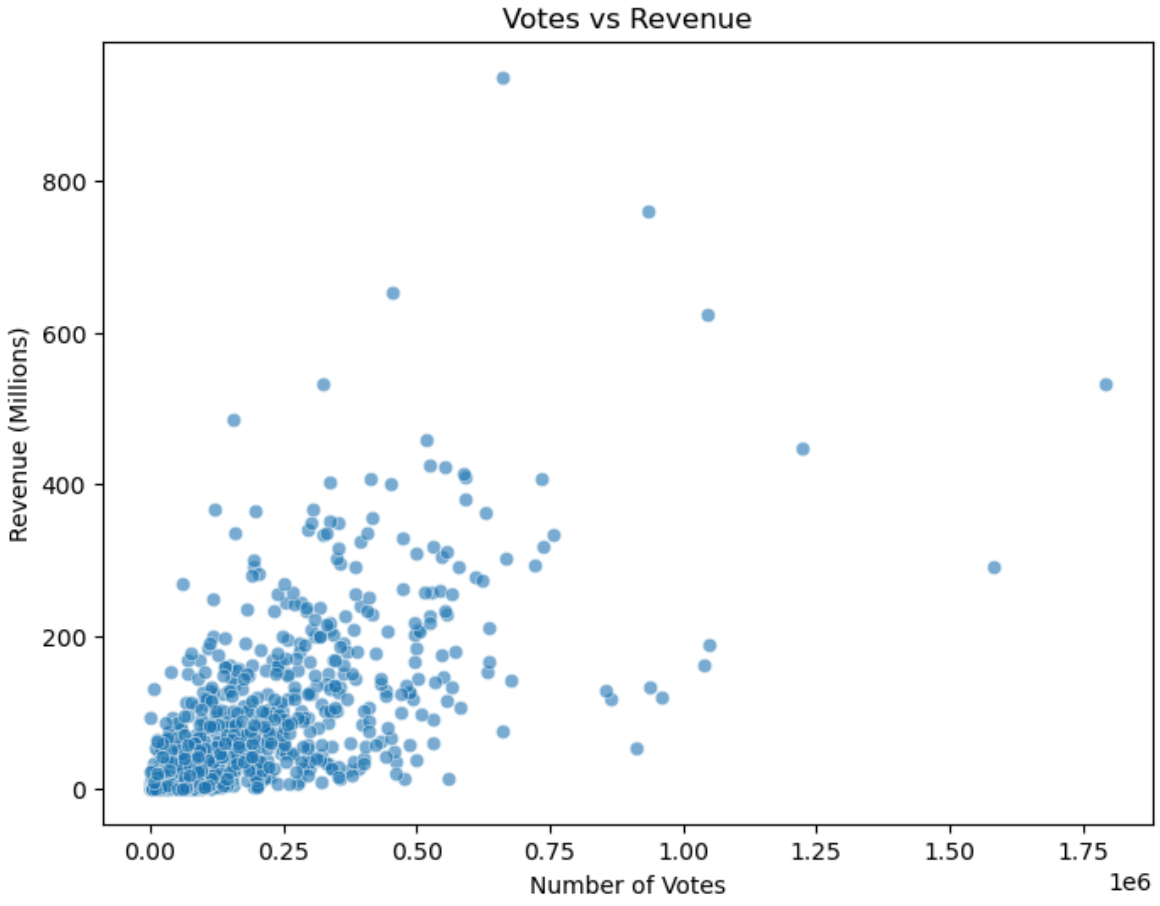


In [40]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(x=df['Rating'], y=df['Revenue'], hue=df['Success'], palette="coc
plt.title("Rating vs Revenue (colored by Success)")
plt.xlabel("IMDB Rating")
plt.ylabel("Revenue (Millions)")
plt.show()
```

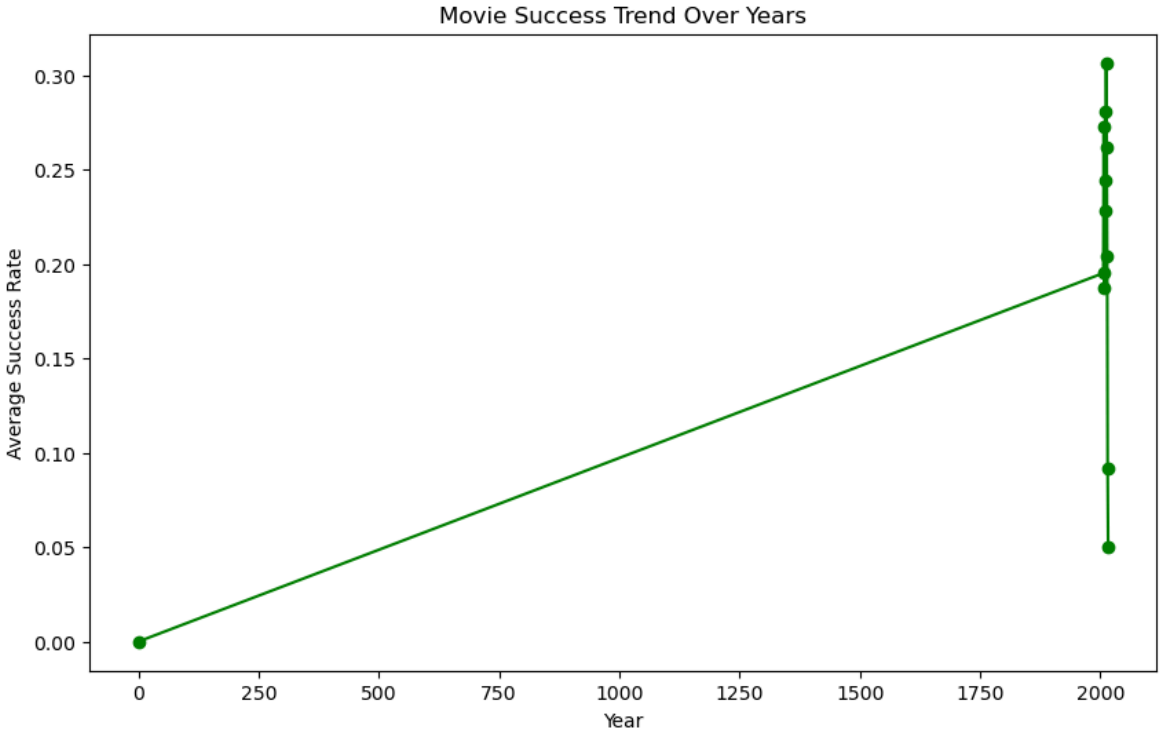


In [41]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(x=df['Votes'], y=df['Revenue'], alpha=0.6)
plt.title("Votes vs Revenue")
plt.xlabel("Number of Votes")
```

```
plt.ylabel("Revenue (Millions)")
plt.show()
```



In [42]:
```
year_success = df.groupby('Year')['Success'].mean()

plt.figure(figsize=(10,6))
year_success.plot(marker='o', color='green')
plt.title("Movie Success Trend Over Years")
plt.ylabel("Average Success Rate")
plt.show()
```


Movie Success Trend Over Years
Average Success Rate
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0
250
500
750
1000
1250
1500
1750
2000
Year

In [ ]:

In [ ]: