```
Start coding or generate with AI.
```

```
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
from sklearn.metrics import mean_squared_error, r2_score
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remoun
```

```
df=pd.read_csv('/content/drive/MyDrive/imdb_movie_dataset.csv')
```

```
df.head(5)
```

	Rank	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Reven (Million
0	1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	8.1	757074	333.
1	2	Prometheus	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.
2	3	Split	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2016	117	7.3	157606	138.
3	4	Sing	Animation,Comedy,Family	In a city of humanoid animals, a	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth	2016	108	7.2	60545	270.

```
df.info()
```

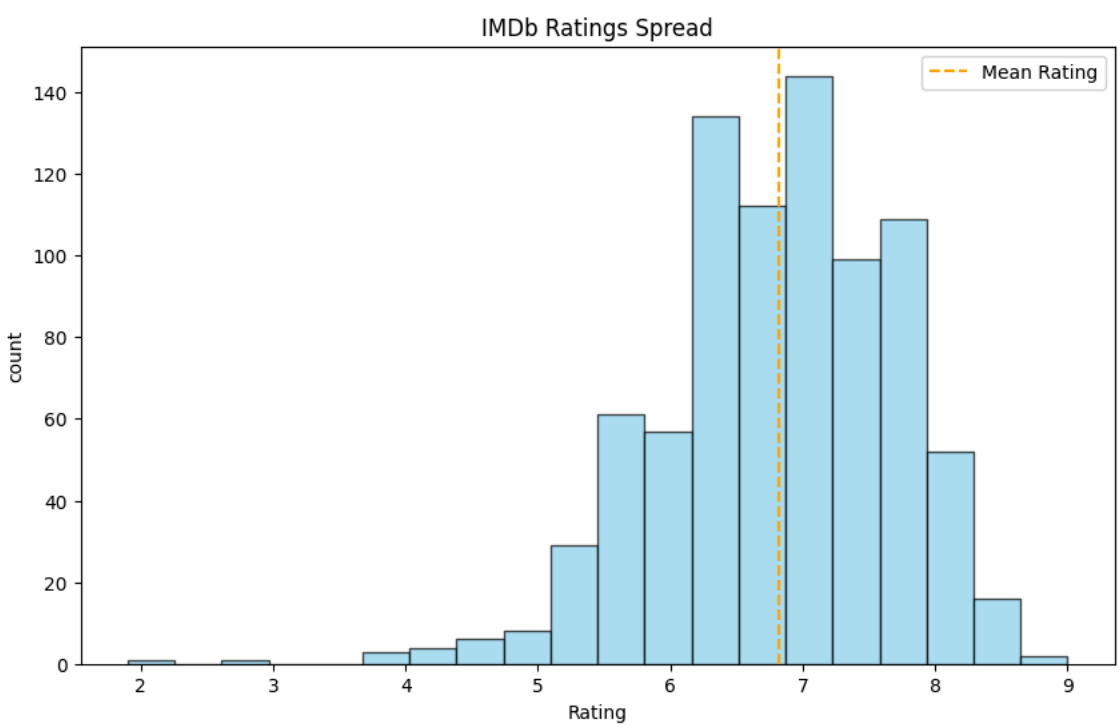
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Rank                1000 non-null   int64
 1   Title               1000 non-null   object
 2   Genre               1000 non-null   object
 3   Description         1000 non-null   object
 4   Director            1000 non-null   object
 5   Actors              1000 non-null   object
 6   Year                1000 non-null   int64
 7   Runtime (Minutes)   1000 non-null   int64
 8   Rating              1000 non-null   float64
 9   Votes               1000 non-null   int64
 10  Revenue (Millions)  872 non-null    float64
 11  Metascore           936 non-null    float64
dtypes: float64(3), int64(4), object(5)
memory usage: 93.9+ KB
```

```
print("Missing values in each column:")
print(df.isnull().sum())
```

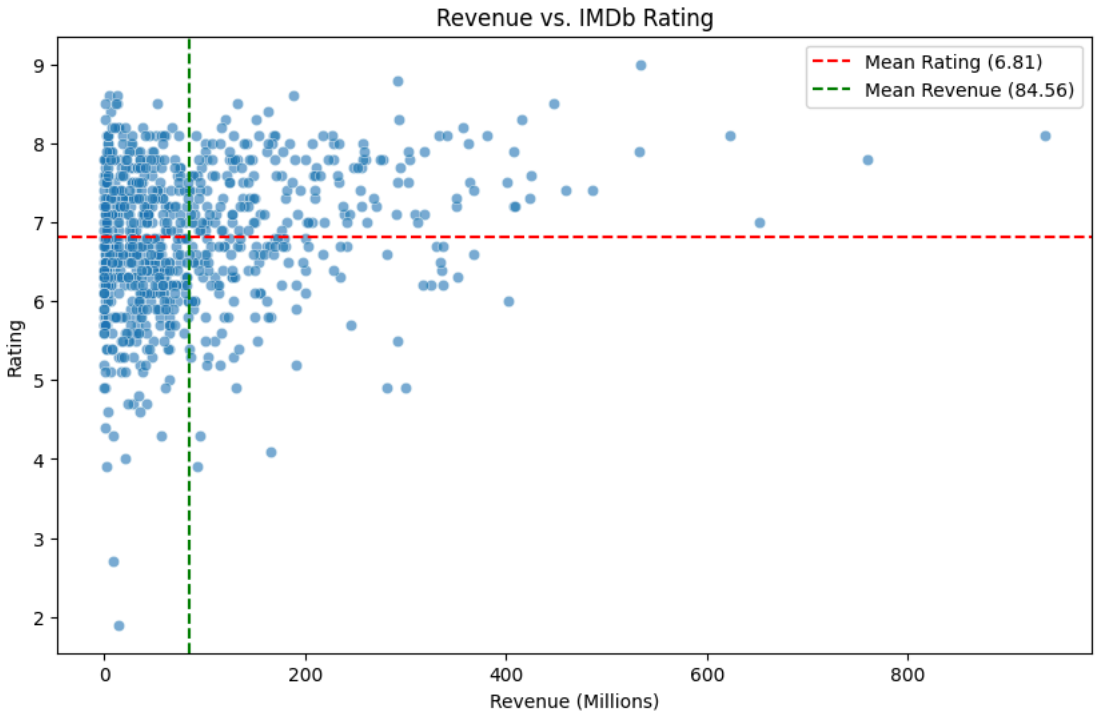
```
Missing values in each column:
Rank                    0
Title                   0
Genre                   0
Description             0
Director                0
Actors                  0
Year                    0
Runtime (Minutes)       0
Rating                  0
Votes                   0
Revenue (Millions)    128
Metascore              64
```

```
dtype: int64
```

```
# converts the values in the Revenue (Millions) column of the DataFrame df to a numeric data type
# non numeric values replaced my Nan

df['Revenue (Millions)'] = pd.to_numeric(df['Revenue (Millions)'], errors='coerce')


# Drop rows with missing values
df.dropna(inplace=True)



plt.figure(figsize=(10, 6))
plt.hist(df['Rating'], bins=20, color='skyblue', edgecolor='black', alpha=0.7)

plt.title('IMDb Ratings Spread')
plt.xlabel('Rating')
plt.ylabel('count')

plt.axvline(df['Rating'].mean(), color='orange', linestyle='--', label='Mean Rating')
plt.legend()

plt.show()
```




```
fig, ax = plt.subplots(figsize=(10, 6))
sns.scatterplot(data=df, x='Revenue (Millions)', y='Rating', alpha=0.6, ax=ax)


ax.set_title('Revenue vs. IMDb Rating')
ax.set_xlabel('Revenue (Millions)')
ax.set_ylabel('Rating')

# Adding the horizontal and vertical lines for the mean values
mean_rating = df['Rating'].mean()
mean_revenue = df['Revenue (Millions)'].mean()

ax.axhline(mean_rating, color='red', linestyle='--', label=f'Mean Rating ({mean_rating:.2f})')
ax.axvline(mean_revenue, color='green', linestyle='--', label=f'Mean Revenue ({mean_revenue:.2f})')

ax.legend()
plt.show()
```




```
X = df[['Revenue (Millions)', 'Runtime (Minutes)', 'Votes', 'Director']]
y = df['Rating']


X = pd.get_dummies(X, columns=['Director'], drop_first=True)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


model = LinearRegression()
model.fit(X_train, y_train)


y_pred = model.predict(X_test)


mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse:.2f}')
print(f'R^2 Score: {r2:.2f}')
```

```
Mean Squared Error: 0.56
R^2 Score: 0.30
```