İsmail Sarp Erdem 32105

DSA210 Spring2025 Project Second Draft

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import ast
from scipy import stats
from itertools import combinations

def load_and_merge(movies_path: str, credits_path: str) -> pd.DataFrame:
    """Merging two dataset"""
    movies = pd.read_csv(movies_path)
    credits = pd.read_csv(credits_path)
    df = pd.merge(movies, credits, left_on='id', right_on='movie_id', how='inner')
    return df
```

```python
def preprocess(df: pd.DataFrame) -> pd.DataFrame:
    """Data Process"""
    df['release_date'] = pd.to_datetime(df['release_date'], errors='coerce')
    df['year'] = df['release_date'].dt.year

    df['genre_list'] = df['genres'].apply(lambda x: [d['name'] for d in ast.literal_eval(x)])
    df['primary_genre'] = df['genre_list'].apply(lambda lst: lst[0] if lst else None)
    df['genre_count'] = df['genre_list'].apply(len)

    df['runtime_bucket'] = pd.cut(
        df['runtime'],
        bins=[0, 90, 120, 150, np.inf],
        labels=['<=90', '91-120', '121-150', '150+']
    )
    return df
```

```python
def print_all_pearson(df: pd.DataFrame, cols: list):
    """Print Pearson r and p-value for every unique pair in cols."""
    print("\n=== Pairwise Pearson Correlations with p-values ===")
    for a, b in combinations(cols, 2):
        x, y = df[a], df[b]
        mask = x.notna() & y.notna()
        if mask.sum() < 2: continue
        r, p = stats.pearsonr(x[mask], y[mask])
        print(f"{a:12s} <-> {b:12s} : r = {r:6.3f}, p = {p:.2e}")
```

```
=== Pairwise Pearson Correlations with p-values ===
budget       ↔ revenue      : r =  0.731, p = 0.00e+00
budget       ↔ popularity   : r =  0.505, p = 7.05e-310
budget       ↔ vote_count   : r =  0.593, p = 0.00e+00
budget       ↔ vote_average : r =  0.093, p = 9.95e-11
budget       ↔ runtime      : r =  0.270, p = 6.91e-81
budget       ↔ genre_count  : r =  0.269, p = 1.66e-80
revenue      ↔ popularity   : r =  0.645, p = 0.00e+00
revenue      ↔ vote_count   : r =  0.781, p = 0.00e+00
revenue      ↔ vote_average : r =  0.197, p = 2.72e-43
```

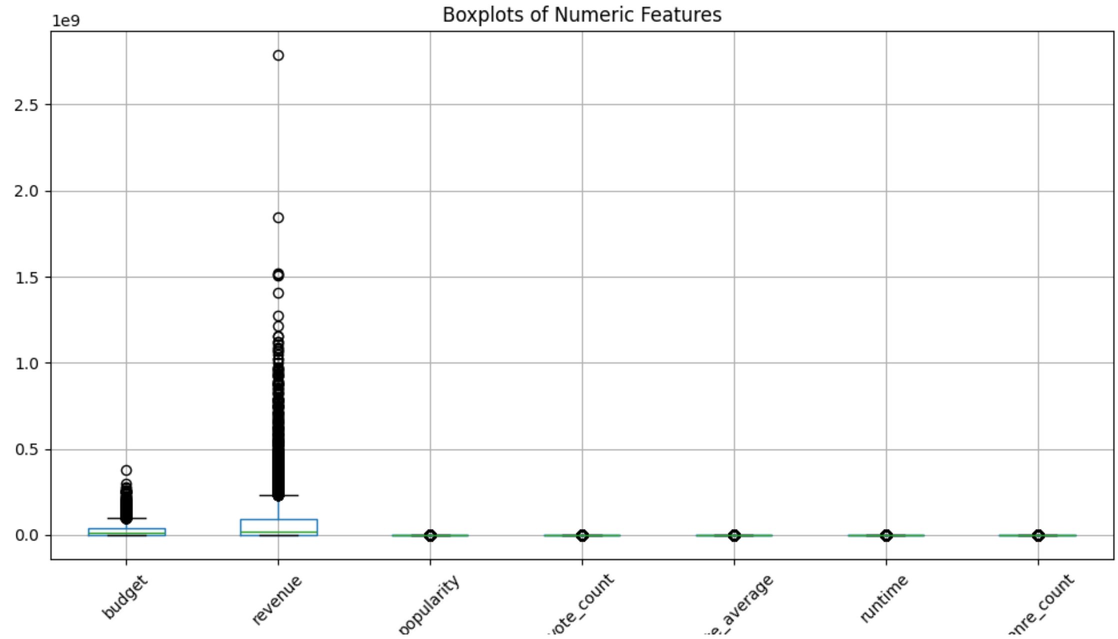
```
revenue      ↔ runtime      : r =  0.251, p = 6.25e-70
revenue      ↔ genre_count  : r =  0.182, p = 4.05e-37
popularity   ↔ vote_count   : r =  0.778, p = 0.00e+00
popularity   ↔ vote_average : r =  0.274, p = 1.95e-83
popularity   ↔ runtime      : r =  0.226, p = 2.09e-56
popularity   ↔ genre_count  : r =  0.155, p = 3.48e-27
vote_count   ↔ vote_average : r =  0.313, p = 1.19e-109
vote_count   ↔ runtime      : r =  0.272, p = 3.64e-82
vote_count   ↔ genre_count  : r =  0.154, p = 7.04e-27
vote_average ↔ runtime      : r =  0.375, p = 3.29e-160
```

```
vote_average ↔ genre_count  : r =  0.086, p = 2.85e-09
runtime      ↔ genre_count  : r =  0.098, p = 8.78e-12
```

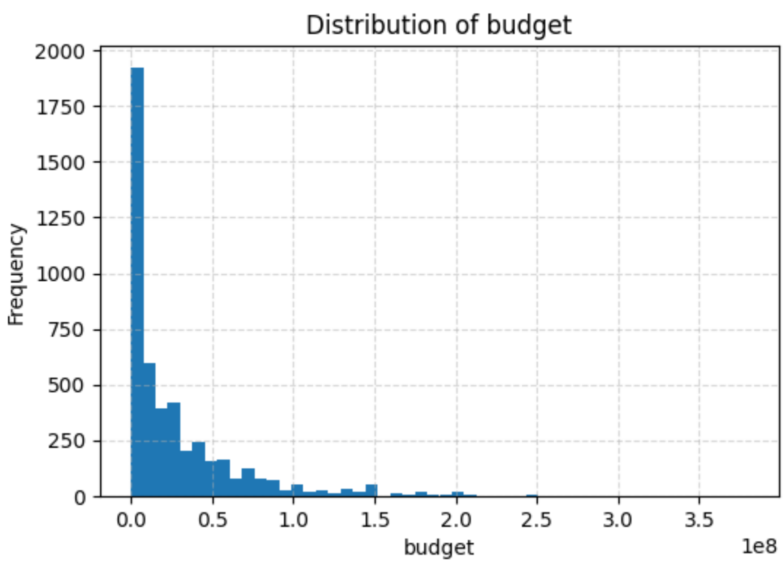
```python
def eda(df: pd.DataFrame):  1 usage
    # 2. Descriptive stats
    numeric = [
        'budget', 'revenue', 'popularity',
        'vote_count', 'vote_average', 'runtime', 'genre_count'
    ]
    print("\n=== Descriptive Statistics ===")
    print(df[numeric].describe())

```

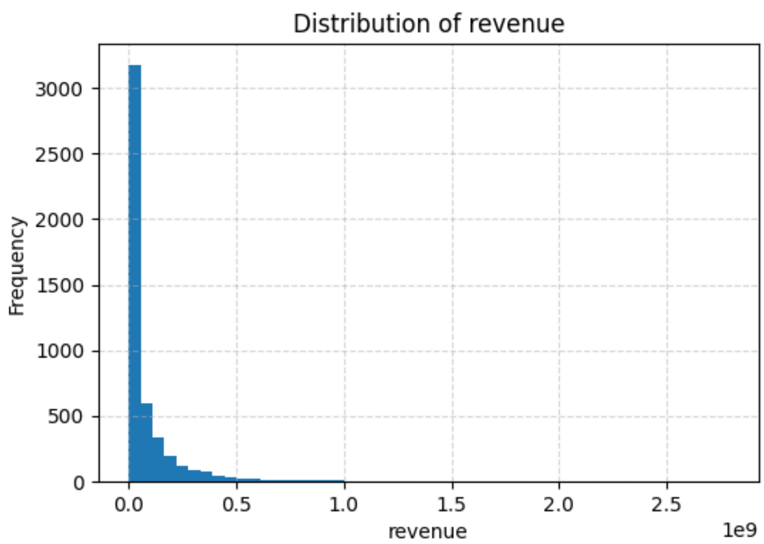
```
=== Descriptive Statistics ===
             budget       revenue  ...      runtime  genre_count
count  4.803000e+03  4.803000e+03  ...  4801.000000  4803.000000
mean   2.904504e+07  8.226064e+07  ...   106.875859     2.531751
std    4.072239e+07  1.628571e+08  ...    22.611935     1.120955
min    0.000000e+00  0.000000e+00  ...     0.000000     0.000000
25%    7.900000e+05  0.000000e+00  ...    94.000000     2.000000
50%    1.500000e+07  1.917000e+07  ...   103.000000     2.000000
75%    4.000000e+07  9.291719e+07  ...   118.000000     3.000000
max    3.800000e+08  2.787965e+09  ...   338.000000     7.000000
```

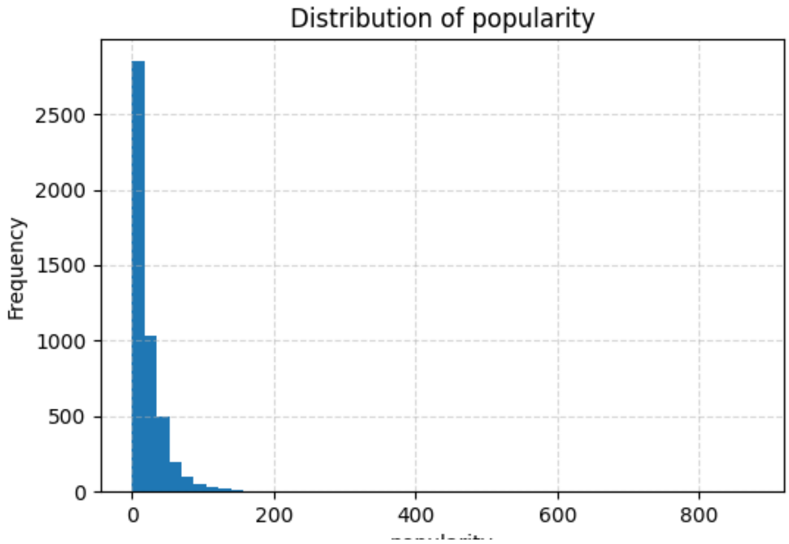
```python
def eda(df: pd.DataFrame):  1 usage
    # 3. Boxplots
    plt.figure(figsize=(12, 6))
    df[numeric].boxplot()
    plt.title("Boxplots of Numeric Features")
    plt.xticks(rotation=45)
    plt.show()
```

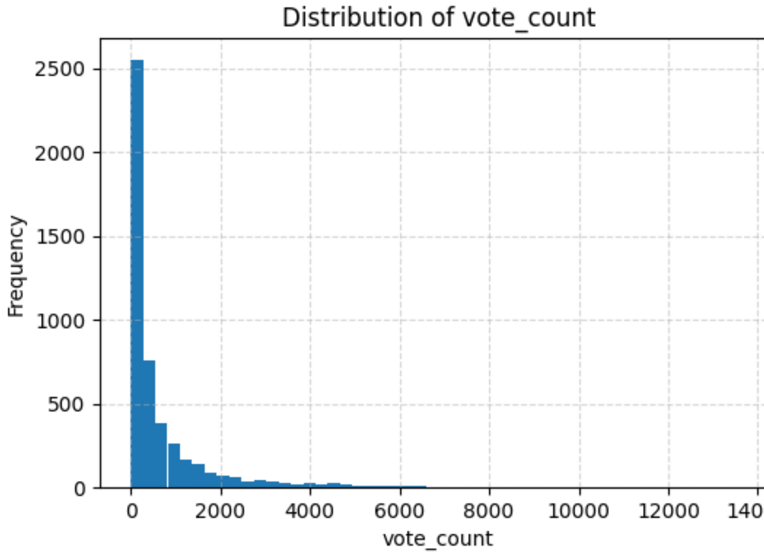


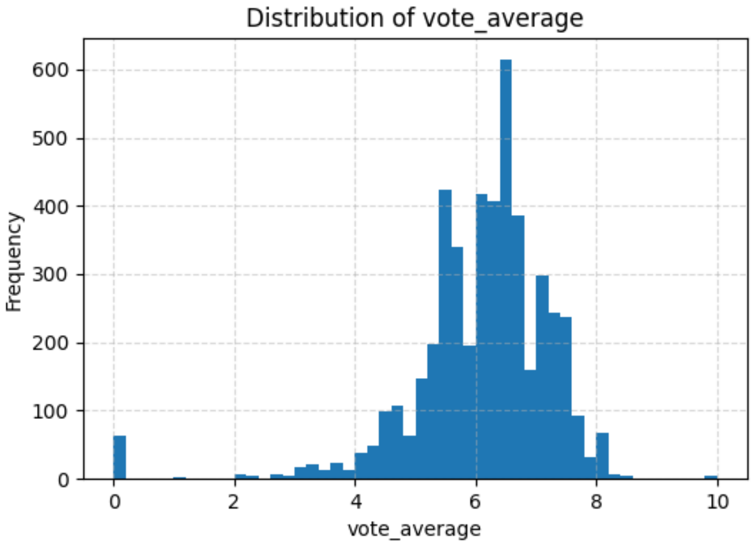

```python
def eda(df: pd.DataFrame):  1 usage
    # 4. Histograms
    for col in numeric:
        plt.figure(figsize=(6, 4))
        df[col].dropna().hist(bins=50)
        plt.title(f'Distribution of {col}')
        plt.xlabel(col)
        plt.ylabel('Frequency')
        plt.grid( visible: True, linestyle='--', alpha=0.5)
        plt.show()
```

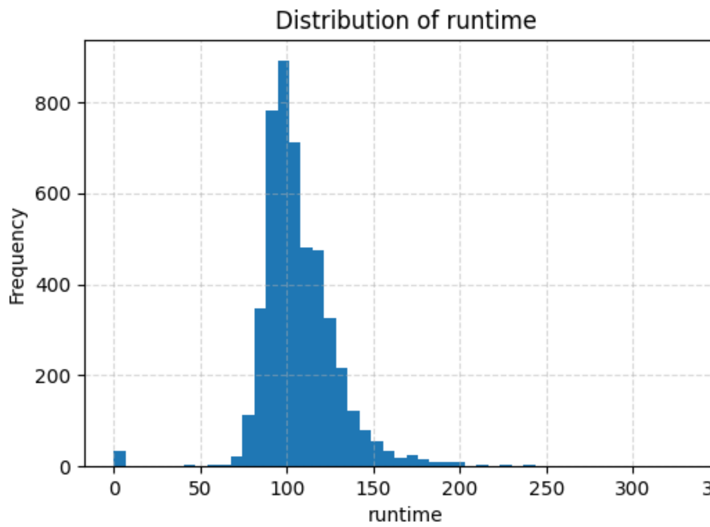


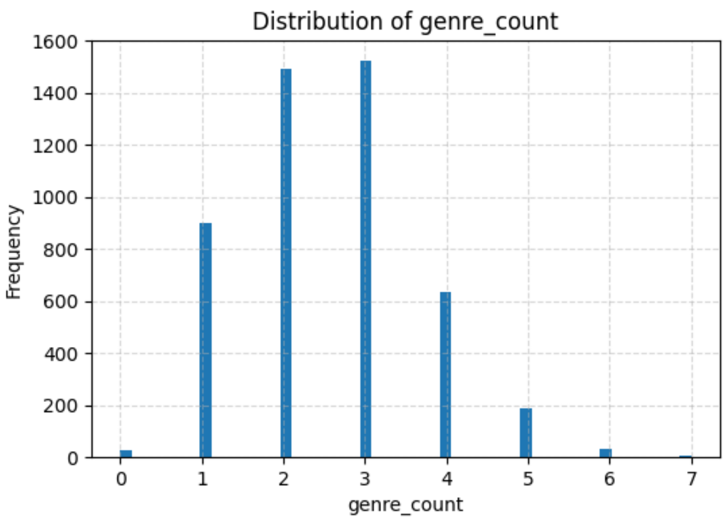


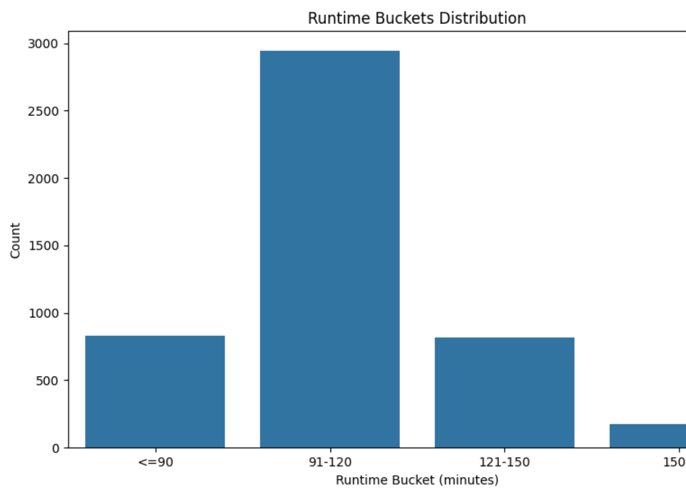




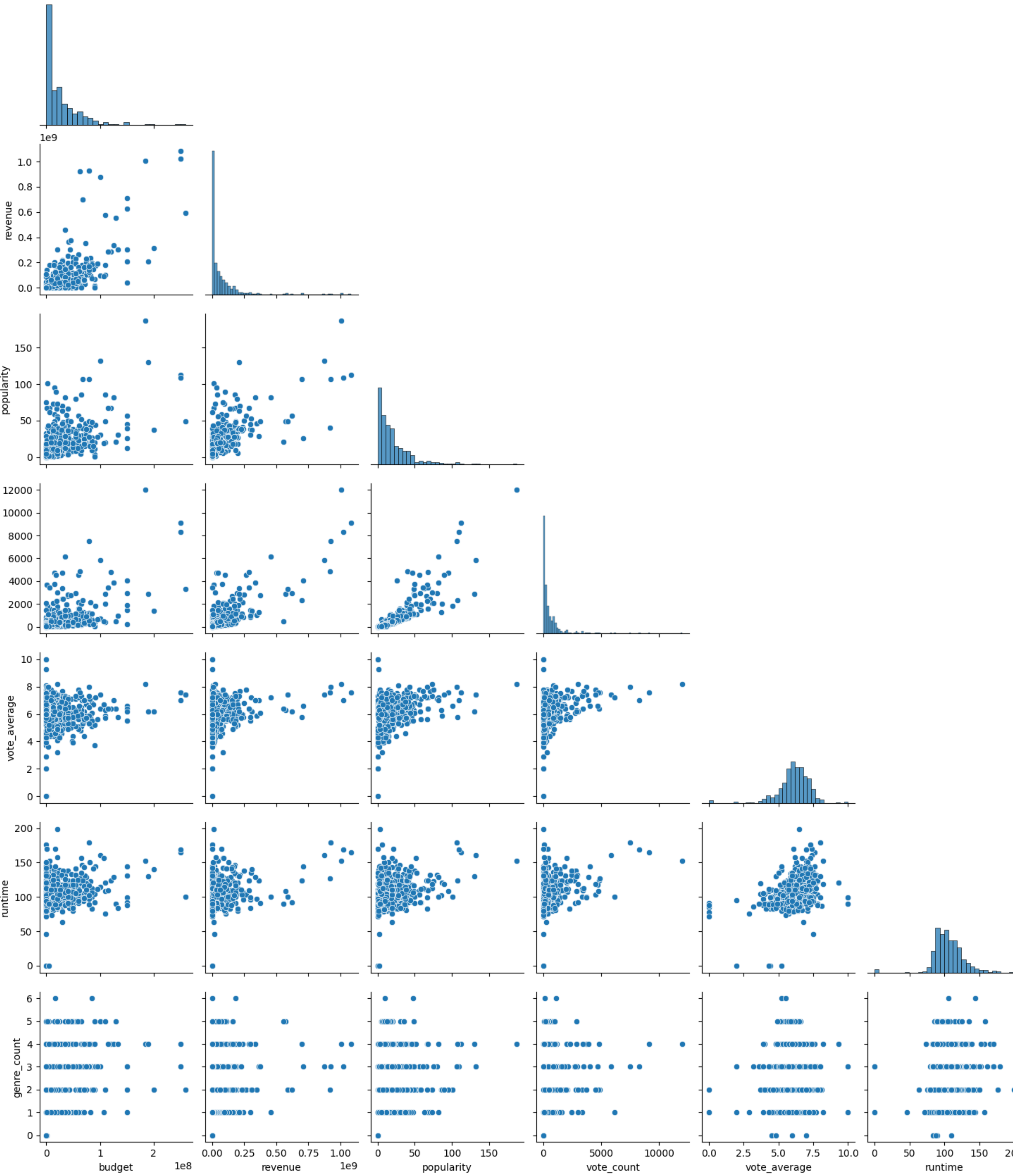








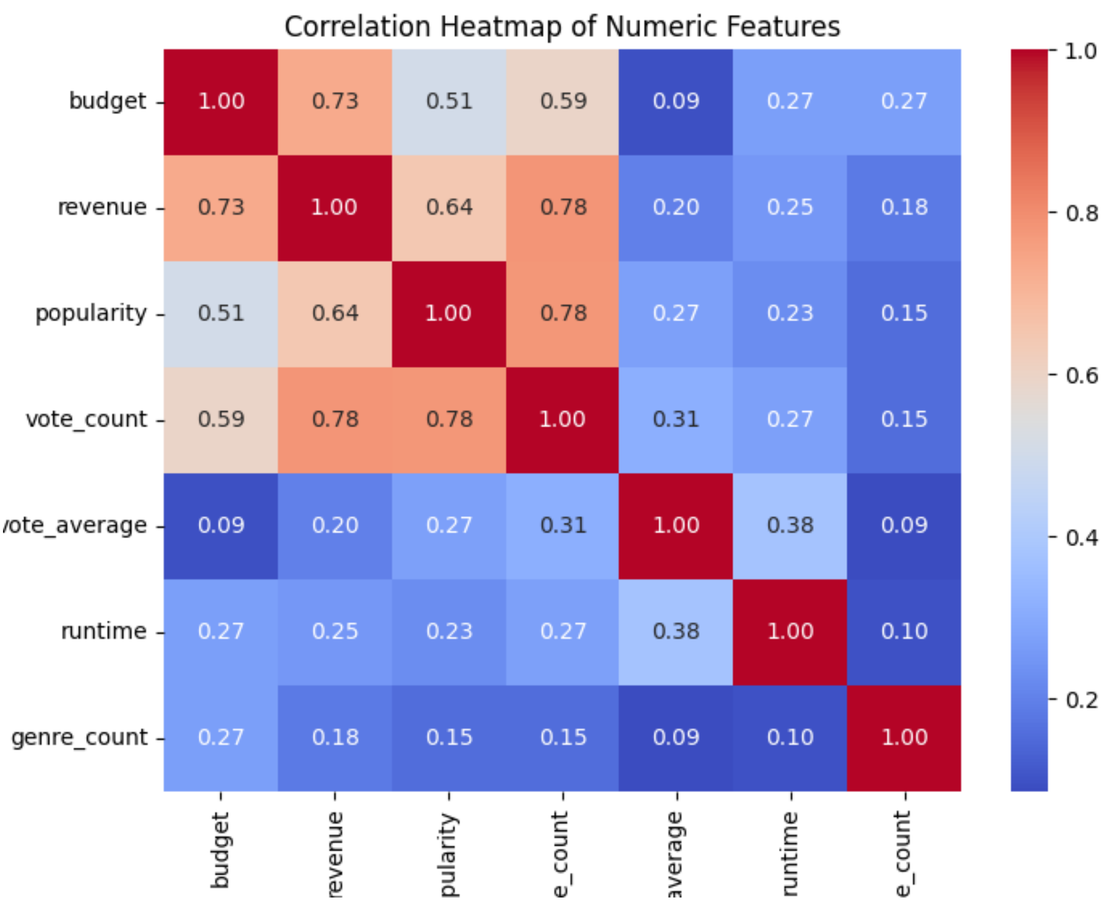

```
# 5. Pairplot (scatter-matrix) of numeric features (sampled)
sample_df = df[numeric].dropna().sample(min(len(df), 500))
g = sns.pairplot(sample_df, corner=True)
g.fig.suptitle( t: "Pairwise Scatterplots (sampled)", y=1.02)
plt.show()
```



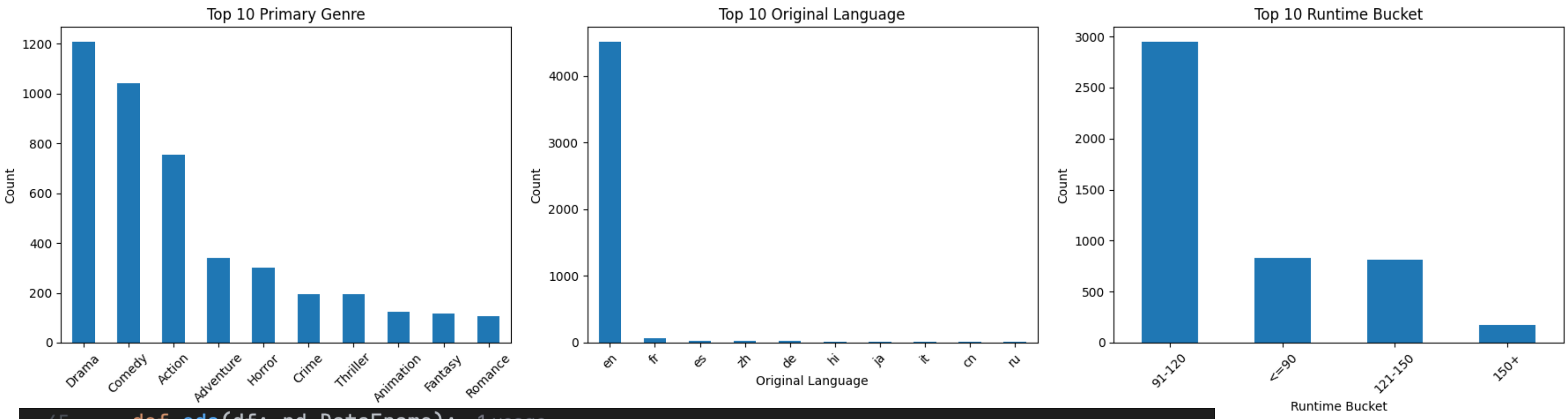
1e9
revenue
1.0 0.8 0.6 0.4 0.2 0.0
popularity
150 100 50 0
12000 10000 8000 6000 4000 2000 0
vote_average
10 8 6 4 2 0
runtime
200 150 100 50 0
genre_count
6 5 4 3 2 1 0
0 1 2
budget
1e8
0.00 0.25 0.50 0.75 1.00
revenue
1e9
0 50 100 150
popularity
0 5000 10000
vote_count
0.0 2.5 5.0 7.5 10.0
vote_average
0 50 100 150
runtime

```python
# 6. Correlation matrix & heatmap
corr = df[numeric].corr()
print("\n=== Correlation Matrix ===")
print(corr)
plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap of Numeric Features")
plt.show()
```

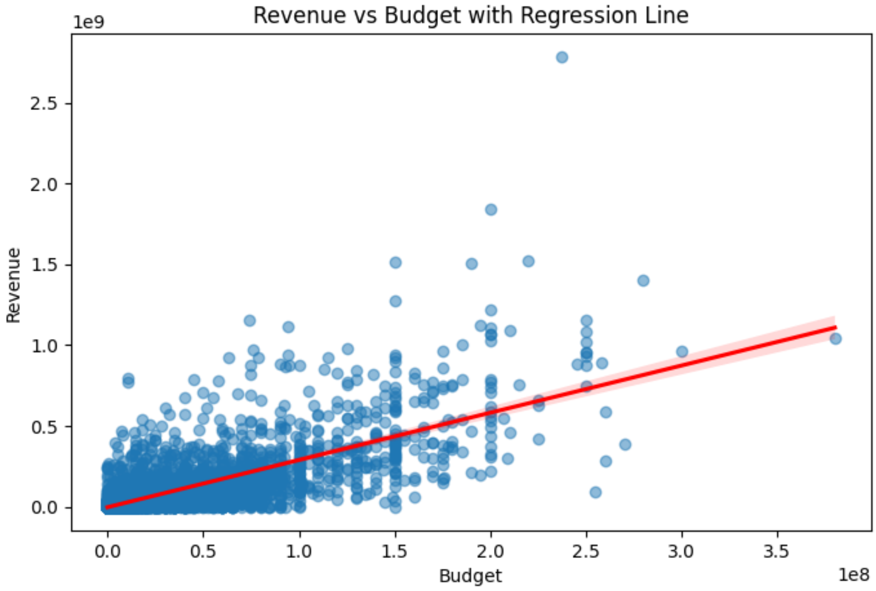
```
=== Correlation Matrix ===
                budget   revenue  ...   runtime  genre_count
budget        1.000000  0.730823  ...  0.269851     0.269170
revenue       0.730823  1.000000  ...  0.251093     0.182185
popularity    0.505414  0.644724  ...  0.225502     0.154918
vote_count    0.593180  0.781487  ...  0.271944     0.154000
vote_average  0.093146  0.197150  ...  0.375046     0.085577
runtime       0.269851  0.251093  ...  1.000000     0.098290
genre_count   0.269170  0.182185  ...  0.098290     1.000000
```

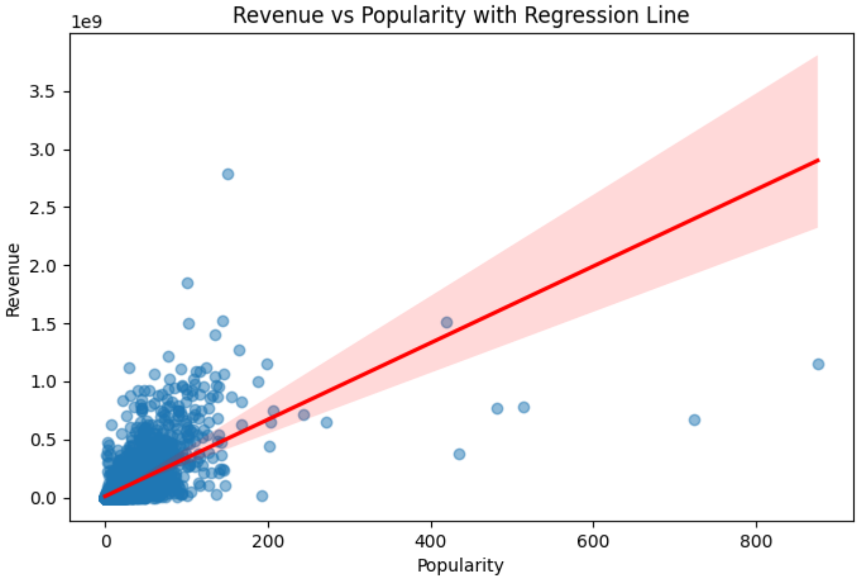


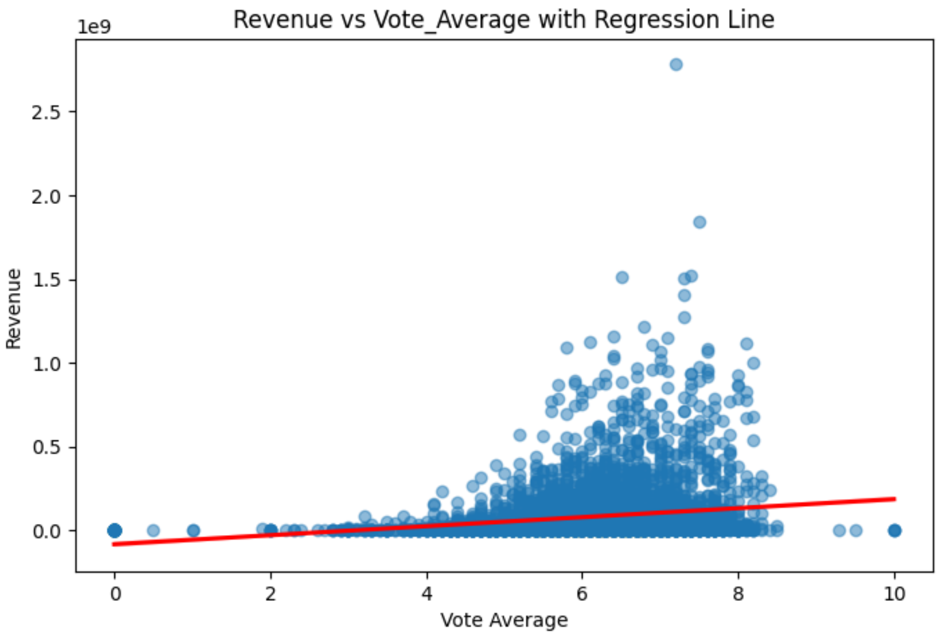
```python
    # 7. Categorical distributions
    cats = ['primary_genre', 'original_language', 'runtime_bucket']
    fig, axes = plt.subplots(1, len(cats), figsize=(18, 5))
    for ax, var in zip(axes, cats):
        counts = df[var].value_counts().nlargest(10)
        counts.plot(kind='bar', ax=ax)
        ax.set_title(f'Top 10 {var.replace("_", " ").title()}')
        ax.set_xlabel(var.replace("_", " ").title())
        ax.set_ylabel("Count")
        ax.tick_params(axis='x', rotation=45)
    plt.tight_layout()
    plt.show()
```




```python
def eda(df: pd.DataFrame):
    # 8. Scatter plots with regression lines
    pairs = [
        ('budget', 'revenue'),
        ('popularity', 'revenue'),
        ('vote_average', 'revenue')
    ]
    for x, y in pairs:
        plt.figure(figsize=(8, 5))
        sns.regplot(
            x=x, y=y, data=df,
            scatter_kws={'alpha':0.5},
            line_kws={'color':'red'}
        )
        plt.title(f"{y.title()} vs {x.title()} with Regression Line")
        plt.xlabel(x.replace('_', ' ').title())
        plt.ylabel(y.replace('_', ' ').title())
        plt.show()
```


Revenue vs Budget with Regression Line
1e9
2.5
2.0
1.5
1.0
0.5
0.0
Revenue
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Budget
1e8


Revenue vs Popularity with Regression Line
1e9
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Revenue
0
200
400
600
800
Popularity


Revenue vs Vote_Average with Regression Line
1e9
2.5
2.0
1.5
1.0
0.5
0.0
Revenue
0
2
4
6
8
10
Vote Average

```python
def hypothesis_tests(df: pd.DataFrame):  1 usage
    """Run hypothesis tests with clear null/alternative statements."""
    clean = df[['budget', 'revenue']].dropna()
    r, p = stats.pearsonr(clean['budget'], clean['revenue'])
    print("\n--- Hypothesis Test 1: Budget vs Revenue ---")
    print("H0: ρ = 0 (no correlation)\nHa: ρ ≠ 0 (non-zero correlation)")
    print(f"Pearson r = {r:.3f}, p-value = {p:.3e}")
    print("→", "Reject H0" if p < 0.05 else "Fail to reject H0")

    median_rating = df['vote_average'].median()
    grp_high = df[df['vote_average'] > median_rating]['revenue'].dropna()
    grp_low  = df[df['vote_average'] <= median_rating]['revenue'].dropna()
    t_stat, p_val = stats.ttest_ind(grp_high, grp_low, equal_var=False)
    print("\n--- Hypothesis Test 2: High vs Low Rating Revenue ---")
    print("H0: μ_high = μ_low\nHa: μ_high ≠ μ_low")
    print(f"t-statistic = {t_stat:.3f}, p-value = {p_val:.3e}")
    print("→", "Reject H0" if p_val < 0.05 else "Fail to reject H0")
```

```
--- Hypothesis Test 1: Budget vs Revenue ---
H0: ρ = 0 (no correlation)
Ha: ρ ≠ 0 (non-zero correlation)
Pearson r = 0.731, p-value = 0.000e+00
→ Reject H0

--- Hypothesis Test 2: High vs Low Rating Revenue ---
H0: μ_high = μ_low
Ha: μ_high ≠ μ_low
t-statistic = 10.824, p-value = 6.837e-27
→ Reject H0
```

Results from the Exploratory Data Analysis and Tests

Predictor | r with Revenue | Interpretations

1) Vote Count | ≈ 0.78 **|** Very strong positive association. Films that more people vote on tend to earn much more—both a signal and an outcome of popularity.

2) Budget | ≈ 0.73 **|** Strong positive link. Bigger investment generally yields bigger gross.

3) Popularity | ≈ 0.64 **|** Moderate-strong. TMDB's composite popularity score tracks revenue well.

4) Vote Average | ≈ 0.20 **|** Weak positive. Higher average ratings alone aren't a great predictor of gross.

5) Runtime | ≈ 0.10–0.20 (approximately) **|** Slight positive; standard feature lengths (90–150 min) do tend to earn more.

6) Genre Count | ≈ 0.05 **|** Virtually no linear trend between "how many genres" a movie is tagged with and its revenue.

Hypothesis Tests

1. **Budget ↔ Revenue**
    - **H_0:** No correlation ($\rho = 0$)
    - **Result:** $r \approx 0.73$, $p \ll 0.001$ → **Reject H_0**
    - **Conclusion:** There is a highly significant, strong linear relationship between budget and revenue.
2. **High vs. Low Rating Revenue**
    - Splitting our sample at the median for IMDb-style rating.
    - **H_0:** Mean revenue of “high-rating” equals “low-rating” group
    - **Result:** $t \approx 10.8$, $p \approx 6.8 \times 10^{-27}$ → **Reject H_0**
    - **Conclusion:** Even though the r is small (≈0.20), movies with above-median ratings earn significantly more on average than lower-rated ones.

- Also further hypothesis tests can be concluded for seeing relationships between other entities too.

<u>Overall Conclusions From The Analysis</u>

Vote_count (how many users rate the film) and **popularity** are the strongest correlates of box-office revenue. They combine both pre- and post-release results .

Budget remains a powerful predictor: larger production and marketing budgets yield higher revenues.

Ratings (vote_average) play a mediocre role. While they don’t linearly track revenue as tightly, the t-test shows that better-rated movies still deliver a clear boost in mean earnings and revenue.

Runtime and **genre count** have minimal linear effects.