Movie Dataset Analysis

This notebook performs an exploratory data analysis (EDA) on the TMDb 5000 movie and credits datasets. The goal is to understand the data, clean it, and extract useful insights about What factors influence movie revenue.

---

0. Installing Required Packages and Creating `requirements.txt`

In [ ]:
```
import subprocess
import sys

# List of required packages
required_packages = ['pandas', 'numpy', 'matplotlib', 'seaborn', 'scikit-learn']

# Install packages
for package in required_packages:
    subprocess.check_call([sys.executable, "-m", "pip", "install", package])

# Create a requirements.txt file with current environment's packages
with open('requirements.txt', 'w') as f:
    subprocess.check_call([sys.executable, "-m", "pip", "freeze"], stdout=f)
```

---

1. Importing Necessary Libraries

We begin by importing the required libraries for data manipulation, visualization, and analysis.

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

# Set visualization style
sns.set(style="darkgrid")
```

---

2. Loading the Data

Next, we load the datasets for movies and credits from CSV files.

In [ ]:
```
# Load CSV files
credits_df = pd.read_csv('data/tmdb_5000_credits.csv')
movies_df = pd.read_csv('data/tmdb_5000_movies.csv')
```

Exploring Datasets

1. Display first few rows of each dataset to get an initial understanding of data.
2. Get a concise summary of datasets, including data types and non-null counts.
3. Check for missing values in each dataset.

In [ ]:
```
# Display first few rows of each dataset
print("First Dataset:")
print(credits_df.head())

print("\nSecond Dataset:")
print(movies_df.head())

# Get summary of data
print(credits_df.info())
print(movies_df.info())

# Check for missing values
print(credits_df.isnull().sum())
print(movies_df.isnull().sum())
```

```
First Dataset:
   movie_id                                     title  \
0     19995                                    Avatar
1       285  Pirates of the Caribbean: At World's End
2    206647                                   Spectre
3     49026                     The Dark Knight Rises
4     49529                               John Carter

                                                cast  \
0  [{"cast_id": 242, "character": "Jake Sully", "...
1  [{"cast_id": 4, "character": "Captain Jack Spa...
2  [{"cast_id": 1, "character": "James Bond", "cr...
3  [{"cast_id": 2, "character": "Bruce Wayne / Ba...
4  [{"cast_id": 5, "character": "John Carter", "c...

                                                crew
0  [{"credit_id": "52fe48009251416c750aca23", "de...
1  [{"credit_id": "52fe4232c3a36847f800b579", "de...
2  [{"credit_id": "54805967c3a36829b5002c41", "de...
3  [{"credit_id": "52fe4781c3a36847f81398c3", "de...
4  [{"credit_id": "52fe479ac3a36847f813eaa3", "de...

Second Dataset:
      budget                                             genres  \
0  237000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...
1  300000000  [{"id": 12, "name": "Adventure"}, {"id": 14, "...
2  245000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...
3  250000000  [{"id": 28, "name": "Action"}, {"id": 80, "nam...
4  260000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...

                                      homepage      id  \
0                  http://www.avatarmovie.com/   19995
1  http://disney.go.com/disneypictures/pirates/     285
2   http://www.sonypictures.com/movies/spectre/  206647
3            http://www.thedarkknightrises.com/   49026
4          http://movies.disney.com/john-carter   49529

                                            keywords original_language  \
0  [{"id": 1463, "name": "culture clash"}, {"id":...                en
1  [{"id": 270, "name": "ocean"}, {"id": 726, "na...                en
2  [{"id": 470, "name": "spy"}, {"id": 818, "name...                en
3  [{"id": 849, "name": "dc comics"}, {"id": 853,...                en
4  [{"id": 818, "name": "based on novel"}, {"id":...                en

                             original_title  \
0                                    Avatar
1  Pirates of the Caribbean: At World's End
2                                   Spectre
3                     The Dark Knight Rises
4                               John Carter

                                            overview  popularity  \
0  In the 22nd century, a paraplegic Marine is di...  150.437577
1  Captain Barbossa, long believed to be dead, ha...  139.082615
2  A cryptic message from Bond’s past sends him o...  107.376788
3  Following the death of District Attorney Harve...  112.312950
4  John Carter is a war-weary, former military ca...   43.926995

                                production_companies  \
0  [{"name": "Ingenious Film Partners", "id": 289...
1  [{"name": "Walt Disney Pictures", "id": 2}, {"...
2  [{"name": "Columbia Pictures", "id": 5}, {"nam...
3  [{"name": "Legendary Pictures", "id": 923}, {"...
4        [{"name": "Walt Disney Pictures", "id": 2}]

                                production_countries release_date     revenue  \
0  [{"iso_3166_1": "US", "name": "United States o...   2009-12-10  2787965087
1  [{"iso_3166_1": "US", "name": "United States o...   2007-05-19   961000000
2  [{"iso_3166_1": "GB", "name": "United Kingdom"...   2015-10-26   880674609
3  [{"iso_3166_1": "US", "name": "United States o...   2012-07-16  1084939099
4  [{"iso_3166_1": "US", "name": "United States o...   2012-03-07   284139100

   runtime                                   spoken_languages    status  \
0    162.0  [{"iso_639_1": "en", "name": "English"}, {"iso...  Released
1    169.0           [{"iso_639_1": "en", "name": "English"}]  Released
2    148.0  [{"iso_639_1": "fr", "name": "Fran\u00e7ais"},...  Released
3    165.0           [{"iso_639_1": "en", "name": "English"}]  Released
4    132.0           [{"iso_639_1": "en", "name": "English"}]  Released

                                          tagline  \
0                     Enter the World of Pandora.
1  At the end of the world, the adventure begins.
2                           A Plan No One Escapes
3                                 The Legend Ends
4           Lost in our world, found in another.

                                      title  vote_average  vote_count
0                                    Avatar           7.2       11800
1  Pirates of the Caribbean: At World's End           6.9        4500
2                                   Spectre           6.3        4466
3                     The Dark Knight Rises           7.6        9106
4                               John Carter           6.1        2124
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   movie_id  4803 non-null   int64 
 1   title     4803 non-null   object
 2   cast      4803 non-null   object
 3   crew      4803 non-null   object
dtypes: int64(1), object(3)
memory usage: 150.2+ KB
None
<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
None
movie_id    0
title       0
cast        0
crew        0
dtype: int64
budget                     0
genres                     0
homepage                3091
id                         0
keywords                   0
original_language          0
original_title             0
overview                   3
popularity                 0
production_companies       0
production_countries       0
release_date               1
revenue                    0
runtime                    2
spoken_languages           0
status                     0
tagline                  844
title                      0
vote_average               0
vote_count                 0
dtype: int64
```

---

4. Merging Datasets

To analyze the data together, we'll merge the movies and credits datasets based on the `movie_id` and `id` columns.

In [ ]:

```
# Merge datasets on 'movie_id' and 'id'
combined_df = pd.merge(movies_df, credits_df, left_on='id', right_on='movie_id', how='inner')

# Display first few rows of combined dataset
print("Combined Dataset - First Few Rows:")
print(combined_df.head())

# Get summary of combined data
print("\nCombined Dataset - Summary:")
print(combined_df.info())
```

```
Combined Dataset - First Few Rows:
      budget                                             genres  \
0  237000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...
1  300000000  [{"id": 12, "name": "Adventure"}, {"id": 14, "...
2  245000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...
3  250000000  [{"id": 28, "name": "Action"}, {"id": 80, "nam...
4  260000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...

                                       homepage      id  \
0                   http://www.avatarmovie.com/   19995
1  http://disney.go.com/disneypictures/pirates/     285
2   http://www.sonypictures.com/movies/spectre/  206647
3            http://www.thedarkknightrises.com/   49026
4          http://movies.disney.com/john-carter   49529

                                            keywords original_language  \
0  [{"id": 1463, "name": "culture clash"}, {"id":...                en
1  [{"id": 270, "name": "ocean"}, {"id": 726, "na...                en
2  [{"id": 470, "name": "spy"}, {"id": 818, "name...                en
3  [{"id": 849, "name": "dc comics"}, {"id": 853,...                en
4  [{"id": 818, "name": "based on novel"}, {"id":...                en

                             original_title  \
0                                    Avatar
1  Pirates of the Caribbean: At World's End
2                                   Spectre
3                     The Dark Knight Rises
4                               John Carter

                                            overview  popularity  \
0  In the 22nd century, a paraplegic Marine is di...  150.437577
1  Captain Barbossa, long believed to be dead, ha...  139.082615
2  A cryptic message from Bond’s past sends him o...  107.376788
3  Following the death of District Attorney Harve...  112.312950
4  John Carter is a war-weary, former military ca...   43.926995

                                production_companies  ...  \
0  [{"name": "Ingenious Film Partners", "id": 289...  ...
1  [{"name": "Walt Disney Pictures", "id": 2}, {"...  ...
2  [{"name": "Columbia Pictures", "id": 5}, {"nam...  ...
3  [{"name": "Legendary Pictures", "id": 923}, {"...  ...
4        [{"name": "Walt Disney Pictures", "id": 2}]  ...

                                    spoken_languages    status  \
0  [{"iso_639_1": "en", "name": "English"}, {"iso...  Released
1           [{"iso_639_1": "en", "name": "English"}]  Released
2  [{"iso_639_1": "fr", "name": "Fran\u00e7ais"},...  Released
3           [{"iso_639_1": "en", "name": "English"}]  Released
4           [{"iso_639_1": "en", "name": "English"}]  Released

                                    tagline  \
0                  Enter the World of Pandora.
1  At the end of the world, the adventure begins.
2                        A Plan No One Escapes
3                              The Legend Ends
4           Lost in our world, found in another.

                                    title_x vote_average vote_count movie_id  \
0                                    Avatar          7.2      11800    19995
1  Pirates of the Caribbean: At World's End          6.9       4500      285
2                                   Spectre          6.3       4466   206647
3                     The Dark Knight Rises          7.6       9106    49026
4                               John Carter          6.1       2124    49529

                                    title_y  \
0                                    Avatar
1  Pirates of the Caribbean: At World's End
2                                   Spectre
3                     The Dark Knight Rises
4                               John Carter

                                                cast  \
0  [{"cast_id": 242, "character": "Jake Sully", "...
1  [{"cast_id": 4, "character": "Captain Jack Spa...
2  [{"cast_id": 1, "character": "James Bond", "cr...
3  [{"cast_id": 2, "character": "Bruce Wayne / Ba...
4  [{"cast_id": 5, "character": "John Carter", "c...

                                                crew
0  [{"credit_id": "52fe48009251416c750aca23", "de...
1  [{"credit_id": "52fe4232c3a36847f800b579", "de...
2  [{"credit_id": "54805967c3a36829b5002c41", "de...
3  [{"credit_id": "52fe4781c3a36847f81398c3", "de...
4  [{"credit_id": "52fe479ac3a36847f813eaa3", "de...

[5 rows x 24 columns]

Combined Dataset - Summary:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
```

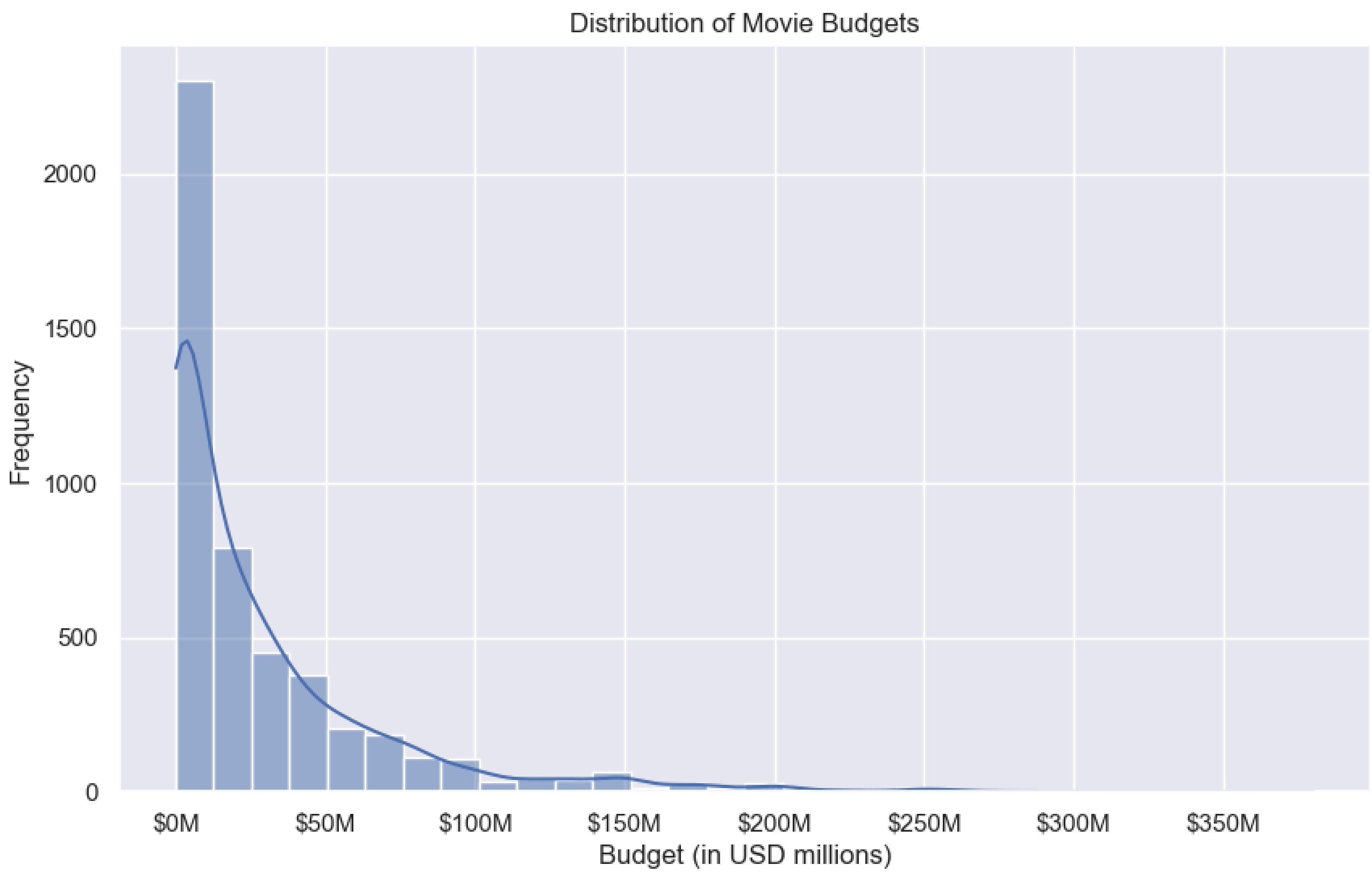
```
 0   budget                4803 non-null   int64
 1   genres                4803 non-null   object
 2   homepage              1712 non-null   object
 3   id                    4803 non-null   int64
 4   keywords              4803 non-null   object
 5   original_language     4803 non-null   object
 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_x               4803 non-null   object
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64
 20  movie_id              4803 non-null   int64
 21  title_y               4803 non-null   object
 22  cast                  4803 non-null   object
 23  crew                  4803 non-null   object
dtypes: float64(3), int64(5), object(16)
memory usage: 900.7+ KB
None
```

---

5. Visualizing Movie Budgets

We will now create a histogram to visualize the distribution of movie budgets.

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.histplot(combined_df['budget'], bins=30, kde=True)

# Customize x-axis labels to show budget in millions
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f'${x/1e6:.0f}M'))

plt.title('Distribution of Movie Budgets')
plt.xlabel('Budget (in USD millions)')
plt.ylabel('Frequency')
plt.show()
```



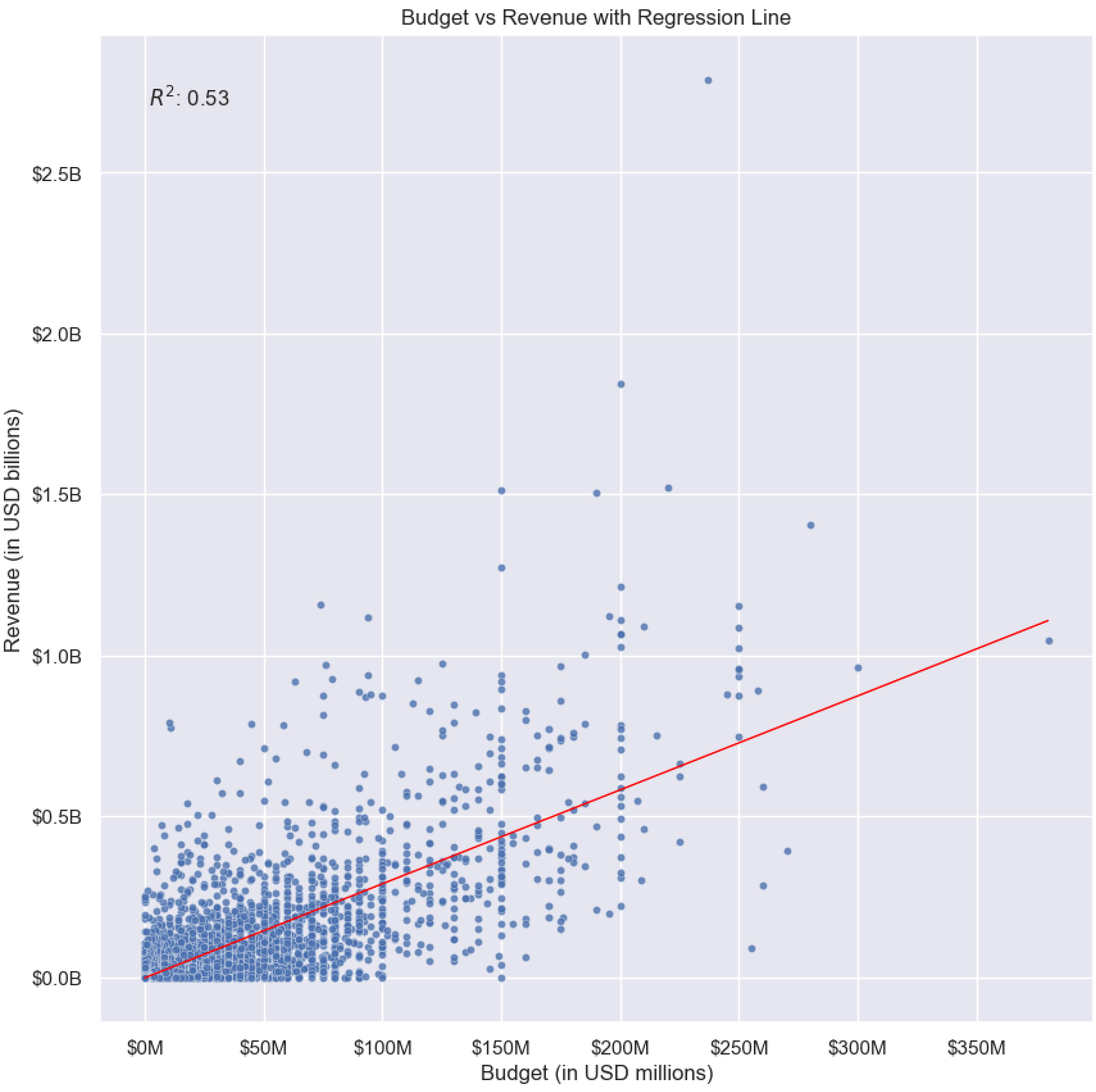
---

6. Budget vs Revenue Scatter Plot

We will plot a scatter plot to visualize the relationship between movie budgets and revenues.

In [ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

plt.figure(figsize=(10, 10))

# Scatter plot with regression line
```

```
sns.regplot(
    data=combined_df,
    x='budget',
    y='revenue',
    scatter_kws={'s': 20, 'edgecolor': 'white', 'linewidths': 0.25},  # Use linewidths instead of linewidth
    line_kws={'color': 'red', 'linewidth': 1},
    ci=None
)

# Customize x-axis labels to show budget in millions
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f'${x/1e6:.0f}M'))

# Customize y-axis labels to show revenue in billions
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda y, _: f'${y/1e9:.1f}B'))

# Calculate R² value
X = combined_df['budget'].values.reshape(-1, 1)
y = combined_df['revenue'].values
model = LinearRegression().fit(X, y)
r2 = model.score(X, y)

# Annotate R² value on plot
plt.text(0.05, 0.95, f'$R^2$: {r2:.2f}', transform=plt.gca().transAxes, fontsize=12, verticalalignment='top')

# Plot title and labels
plt.title('Budget vs Revenue with Regression Line')
plt.xlabel('Budget (in USD millions)')
plt.ylabel('Revenue (in USD billions)')
plt.show()
```



---

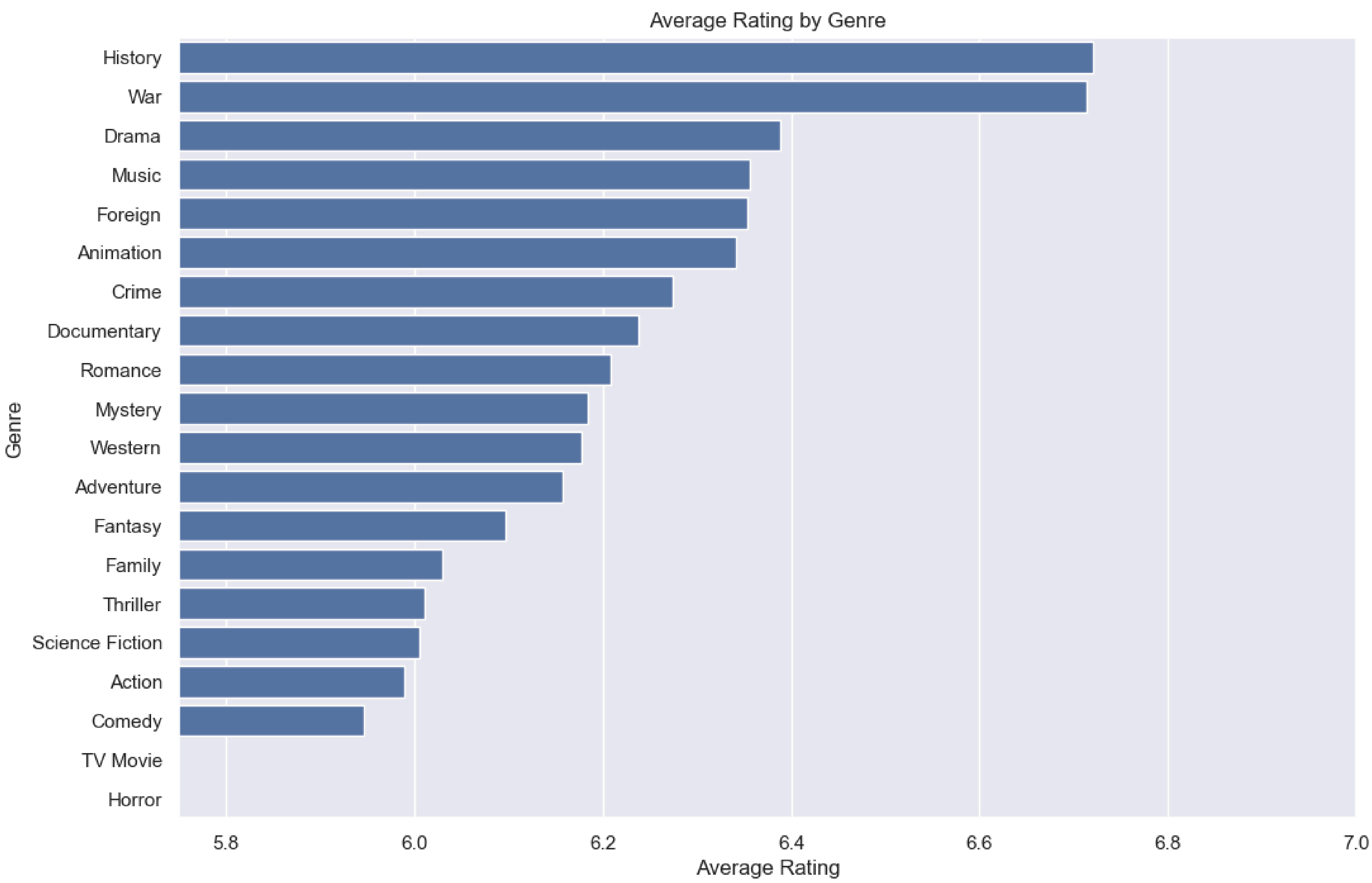
7. Analyzing Genres

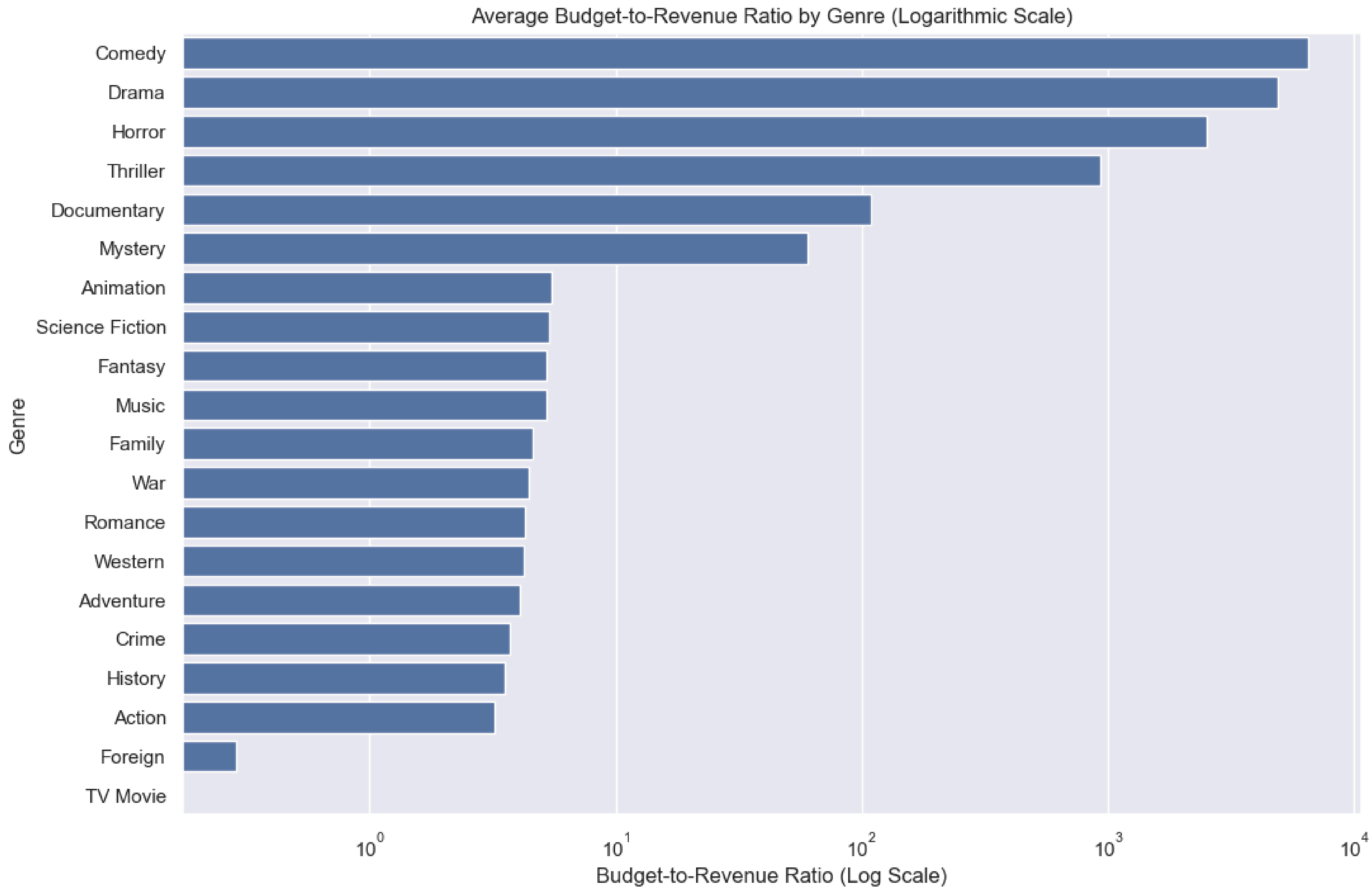
Let's explore the genres in the dataset, compute average ratings by genre as well as correlation between genre and budget-to-revenue ratios, and visualize the results.

In [ ]:
```
import json
from sklearn.linear_model import LinearRegression
```

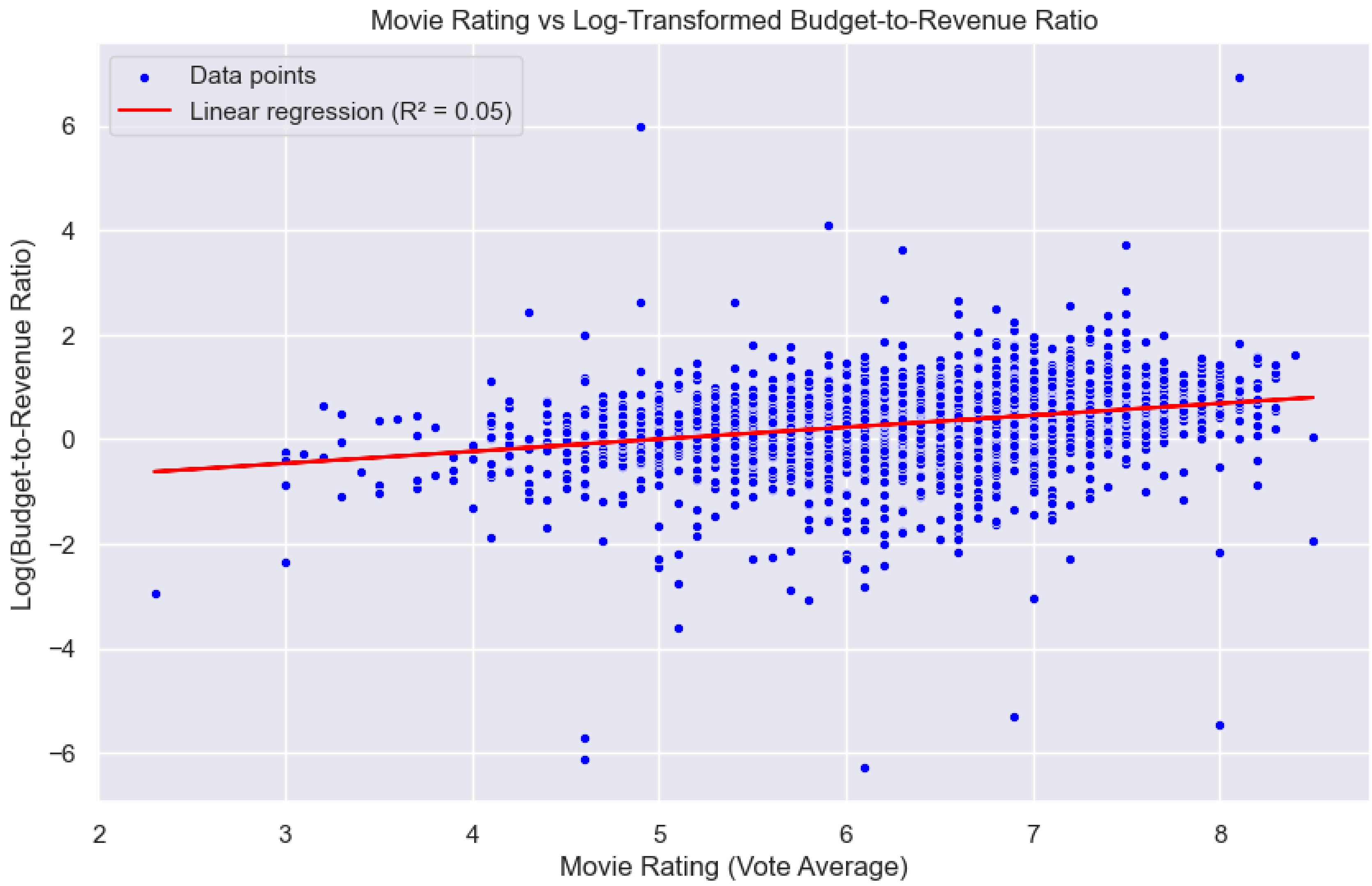
```python
# Parse 'genres' column from JSON strings to Python objects (list of dictionaries)
combined_df['genres_parsed'] = combined_df['genres'].apply(json.loads)

# Extract genre names directly from lists of dictionaries
combined_df['genre_names'] = combined_df['genres_parsed'].apply(lambda x: [i['name'] for i in x])

# Explode genre names into separate rows
genre_df = combined_df.explode('genre_names')

# Calculate average rating per genre
avg_rating_by_genre = genre_df.groupby('genre_names')['vote_average'].mean().sort_values(ascending=False)

# Plot average rating by genre
plt.figure(figsize=(12, 8))
sns.barplot(x=avg_rating_by_genre, y=avg_rating_by_genre.index)
plt.title('Average Rating by Genre')
plt.xlabel('Average Rating')
plt.xlim(5.75, 7)  # Set x-axis limits 5.75-7
plt.ylabel('Genre')
plt.show()

# --- Explore Correlation between Genre and Budget-to-Revenue Ratio ---

# Calculate budget-to-revenue ratio
genre_df['budget_to_revenue_ratio'] = genre_df['revenue'] / genre_df['budget']

# Remove rows with NaN, inf, or very large values in 'budget_to_revenue_ratio'
genre_df = genre_df.replace([np.inf, -np.inf], np.nan)  # Replace inf values with NaN
genre_df = genre_df.dropna(subset=['budget_to_revenue_ratio', 'vote_average'])  # Drop rows with NaN values
genre_df = genre_df[genre_df['budget_to_revenue_ratio'] < 1e10]  # Filter out extremely large values

# Group by genre and calculate average budget-to-revenue ratio
avg_budget_to_revenue_by_genre = genre_df.groupby('genre_names')['budget_to_revenue_ratio'].mean().sort_values(ascending=False)

# Plot average budget-to-revenue ratio by genre with logarithmic x-axis
plt.figure(figsize=(12, 8))
sns.barplot(x=avg_budget_to_revenue_by_genre, y=avg_budget_to_revenue_by_genre.index)
plt.xscale('log')  # Apply log scale to x-axis
plt.title('Average Budget-to-Revenue Ratio by Genre (Logarithmic Scale)')
plt.xlabel('Budget-to-Revenue Ratio (Log Scale)')
plt.ylabel('Genre')
plt.show()

# --- Scatter Plot: Budget-to-Revenue Ratio vs. Movie Rating with Regression Line ---

# Calculate budget-to-revenue ratio for combined dataset
combined_df['budget_to_revenue_ratio'] = combined_df['revenue'] / combined_df['budget']

# Remove rows with NaN, 0, inf, or very large values in 'budget_to_revenue_ratio' or 'vote_average'
combined_df = combined_df.replace([np.inf, -np.inf], np.nan)
combined_df = combined_df.dropna(subset=['budget_to_revenue_ratio', 'vote_average'])
combined_df = combined_df[combined_df['budget_to_revenue_ratio'] < 1e10]
combined_df = combined_df[combined_df['vote_average'] > 0]

# --- Ensure X and y are from same filtered dataset ---

# Add a small constant to avoid log(0) issues
epsilon = 1e-10
combined_df['log_budget_to_revenue_ratio'] = np.log10(combined_df['budget_to_revenue_ratio'] + epsilon)

# Remove any infinite or NaN values that resulted from log transformation
combined_df = combined_df.dropna(subset=['log_budget_to_revenue_ratio', 'vote_average'])

# Prepare data for regression analysis
X = combined_df[['vote_average']].values  # Independent variable: movie ratings
y_log = combined_df['log_budget_to_revenue_ratio'].values  # Dependent variable: log-transformed budget-to-revenue ratio

# Initialize & Fit Model
model = LinearRegression()
model.fit(X, y_log)

# Predict and calculate R-squared for log-transformed data
r2_log = model.score(X, y_log)
print(f'R² value for correlation between movie rating and log-transformed budget-to-revenue ratio: {r2_log:.2f}')

# Predict y values for log-transformed B2R
y_log_pred = model.predict(X)

# Define threshold for filtering out small budget-to-revenue ratios
threshold = 1e-10

# Filter out data points where budget_to_revenue_ratio is too small and create a copy
filtered_df = combined_df[combined_df['budget_to_revenue_ratio'] > threshold].copy()

# Proceed with log transformation
filtered_df['log_budget_to_revenue_ratio'] = np.log10(filtered_df['budget_to_revenue_ratio'] + epsilon)

# Fit model and make predictions
X = filtered_df[['vote_average']].values  # Ensure this is a numpy array
y_log = filtered_df['log_budget_to_revenue_ratio'].values

model = LinearRegression()
model.fit(X, y_log)
```

```
# Predict using model
y_log_pred = model.predict(X)

# Plot data again
plt.figure(figsize=(10, 6))
plt.scatter(filtered_df['vote_average'], filtered_df['log_budget_to_revenue_ratio'], color='blue', edgecolor='white', linewidths=0.5, s=20,
plt.plot(filtered_df['vote_average'], y_log_pred, color='red', linewidth=1.5, label=f'Linear regression (R² = {r2_log:.2f})')
plt.title('Movie Rating vs Log-Transformed Budget-to-Revenue Ratio')
plt.xlabel('Movie Rating (Vote Average)')
plt.ylabel('Log(Budget-to-Revenue Ratio)')
plt.legend()
plt.show()

# Drop rows with 0 for budget and revenue
combined_df = combined_df[(combined_df['budget'] > 0) & (combined_df['revenue'] > 0)]

# Log-transform budget and revenue
combined_df['log_budget'] = np.log10(combined_df['budget'])
combined_df['log_revenue'] = np.log10(combined_df['revenue'])

# --- Scatter Plot: Movie Rating vs. Log-Transformed Budget with Regression Line ---

# Prepare data for regression analysis
X_rating_log_budget = combined_df[['vote_average']].values  # Independent variable: movie ratings
y_log_budget = combined_df['log_budget'].values  # Dependent variable: log-transformed budget

# Initialize & Fit Model
model_rating_log_budget = LinearRegression()
model_rating_log_budget.fit(X_rating_log_budget, y_log_budget)

# Predict and calculate R-squared for rating vs. log-transformed budget
r2_rating_log_budget = model_rating_log_budget.score(X_rating_log_budget, y_log_budget)
print(f'R² value for correlation between movie rating and log-transformed budget: {r2_rating_log_budget:.2f}')

# Predict y values for rating vs. log-transformed budget
y_rating_log_budget_pred = model_rating_log_budget.predict(X_rating_log_budget)

# Scatter plot with regression line for rating vs. log-transformed budget
plt.figure(figsize=(10, 6))
plt.scatter(X_rating_log_budget, y_log_budget, color='green', edgecolor='white', linewidths=0.5, s=20, label='Data points')
plt.plot(X_rating_log_budget, y_rating_log_budget_pred, color='red', linewidth=1.5, label=f'Linear regression (R² = {r2_rating_log_budget:.
plt.title('Movie Rating vs Log-Transformed Budget')
plt.xlabel('Movie Rating (Vote Average)')
plt.ylabel('Log(Budget)')
plt.legend()
plt.show()
```

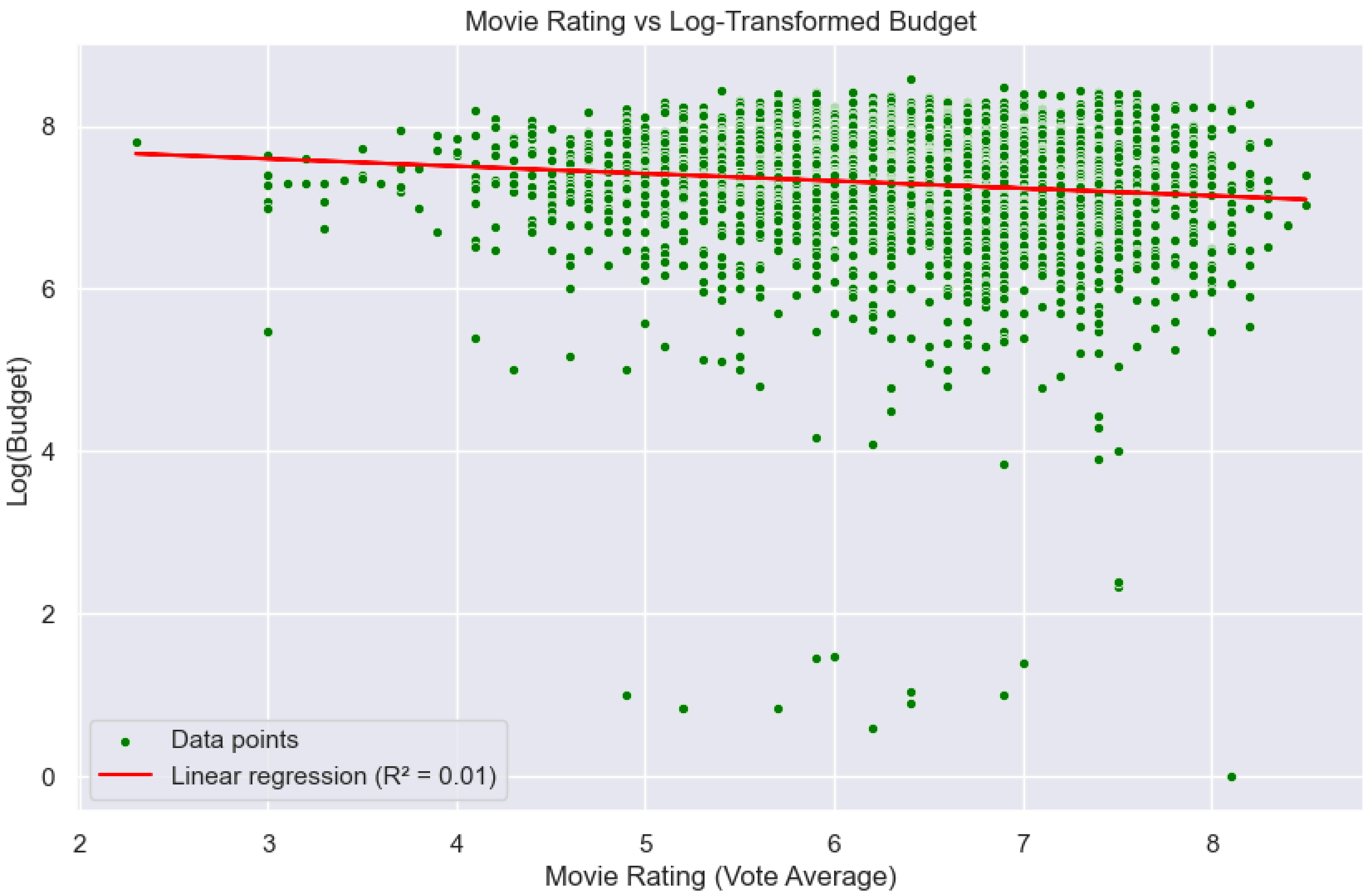





```
R² value for correlation between movie rating and log-transformed budget-to-revenue ratio: 0.05
```




```
R² value for correlation between movie rating and log-transformed budget: 0.01
```



8. Cleaning and Transforming Cast Data

Here, we will parse the `cast` column from the credits dataset, extract relevant information, and clean the data.

In [ ]:
```
import json

# Parse JSON strings
def safe_parse(val):
    if isinstance(val, str):
        try:
            return json.loads(val)
        except (ValueError, json.JSONDecodeError):
            return None  # Handle invalid JSON format
    elif isinstance(val, list):
        return val  # Already a list, no need to parse
    return None  # Handle any other cases

# Apply safe parsing function to 'cast' column
credits_df['cast'] = credits_df['cast'].apply(safe_parse)

# Function to extract desired fields
def extract_cast_info(cast_list):
 return [{'character': cast['character'], 'gender': cast['gender'], 'name': cast['name']} for cast in cast_list]

# Apply function to 'cast' column
credits_df['cast_extracted'] = credits_df['cast'].apply(extract_cast_info)

# Explode 'cast_extracted' column so each row contains a single actor's info
exploded_cast_df = credits_df[['title', 'cast_extracted']].explode('cast_extracted', ignore_index=True)

# Convert list of dictionaries into separate columns
exploded_cast_df = pd.concat([exploded_cast_df.drop('cast_extracted', axis=1), exploded_cast_df['cast_extracted'].apply(pd.Series)], axis=1

# Drop column full of NaN values
exploded_cast_df = exploded_cast_df.drop(columns=[0])

# Change gender column to 'M' and 'F'
exploded_cast_df['gender'] = exploded_cast_df['gender'].replace({2: 'M', 1: 'F'})

# Display final DataFrame
print(exploded_cast_df.head())
```

```
    title            character gender               name
0  Avatar           Jake Sully      M     Sam Worthington
1  Avatar              Neytiri      F         Zoe Saldana
2  Avatar  Dr. Grace Augustine      F    Sigourney Weaver
3  Avatar         Col. Quaritch      M       Stephen Lang
4  Avatar         Trudy Chacon      F  Michelle Rodriguez
```

9. Saving Cleaned Cast Data

We will now save the cleaned cast data to a new CSV file to reduce compute times for future analysis.

In [ ]:
```
# Save new DataFrame to new CSV file
exploded_cast_df.to_csv('data/tmdb_5000_cast_cleaned.csv', index=False)
```

---

10. Merging Cleaned Cast Data with Movies Data

We will merge the cleaned cast data with the movies data to analyze actor performance metrics.

In [ ]:
```
# Load clean cast data
cast_file_path = 'data/tmdb_5000_cast_cleaned.csv'
cast_df = pd.read_csv(cast_file_path)

# Merge datasets on 'title' column
combined_df = pd.merge(cast_df, movies_df[['id', 'title', 'budget', 'revenue']], on='title', how='inner')

# Display first few rows of combined DataFrame to confirm merge
print(combined_df.head())
```

```
    title            character gender               name     id     budget  \
0  Avatar           Jake Sully      M     Sam Worthington  19995  237000000
1  Avatar              Neytiri      F        Zoe Saldana  19995  237000000
2  Avatar  Dr. Grace Augustine      F   Sigourney Weaver  19995  237000000
3  Avatar         Col. Quaritch      M       Stephen Lang  19995  237000000
4  Avatar         Trudy Chacon      F  Michelle Rodriguez  19995  237000000

      revenue
0  2787965087
1  2787965087
2  2787965087
3  2787965087
4  2787965087
```

---

11. Actor Analysis: Budget-to-Revenue Ratio

We will calculate the budget-to-revenue ratio for movies and analyze the average ratios by actor.

In [ ]:
```
# Calculate budget-to-revenue ratio
combined_df['budget_to_revenue_ratio'] = combined_df['revenue'] / combined_df['budget']

# Analyze average budget-to-revenue ratio by actor
actor_analysis = combined_df.groupby('name').agg({
 'budget': 'mean',
 'revenue': 'mean',
 'budget_to_revenue_ratio': 'mean'
}).reset_index()

# Sort results by budget-to-revenue ratio
actor_analysis = actor_analysis.sort_values(by='budget_to_revenue_ratio', ascending=False)

# Display top results
print(actor_analysis.head(10))
```

```
                 name        budget       revenue  budget_to_revenue_ratio
54143        Zoë Hall  0.000000e+00  3.094813e+06                      inf
35      Aakomon Jones  1.533333e+07  1.449227e+08                      inf
37         Aamir Khan  1.100000e+06  6.015562e+06                      inf
40       Aaron Abrams  3.325000e+07  6.390146e+07                      inf
32498       Lydia Fox  0.000000e+00  8.646590e+05                      inf
32483  Lutz Halbhubner  0.000000e+00  3.415310e+07                      inf
54049      Zhang Ziyi  3.031778e+07  1.302567e+08                      inf
54053    Zhao Hongfei  0.000000e+00  9.286394e+07                      inf
54054        Zhao Wei  4.017050e+07  8.529568e+07                      inf
23762       Jerry Hey  0.000000e+00  3.219520e+05                      inf
```

---

12. Filtering Top Actors and Analyzing Movie Performances

We will focus on the top actors based on revenue-to-budget ratio, filter the movies involving them, and analyze their performance.

In [ ]:
```
# Assuming original order of actors in 'cast_extracted' column indicates billing order,
# we'll assign a ranking based on order within each movie title.
cast_df['rank'] = cast_df.groupby('title').cumcount() + 1

# Filter to keep only first 5 actors (based on rank)
filtered_cast_df = cast_df[cast_df['rank'] <= 5]

# Merge filtered cast data with movies data on 'title' column
combined_df = pd.merge(filtered_cast_df, movies_df[['id', 'title', 'budget', 'revenue']], on='title', how='inner')

# Filter out rows where budget or revenue is zero
combined_df = combined_df[(combined_df['budget'] > 0) & (combined_df['revenue'] > 0)]

# Count number of movies per actor
```

```
actor_movie_counts = combined_df['name'].value_counts()

# Filter out actors who have appeared in fewer than 5 movies
actors_with_min_movies = actor_movie_counts[actor_movie_counts >= 5].index
filtered_combined_df = combined_df[combined_df['name'].isin(actors_with_min_movies)].copy()

# Calculate revenue-to-budget ratio for each movie
filtered_combined_df['revenue_to_budget_ratio'] = filtered_combined_df['revenue'] / filtered_combined_df['budget']

# Group by actor and calculate average revenue-to-budget ratio
actor_analysis = filtered_combined_df.groupby('name')['revenue_to_budget_ratio'].mean().reset_index()

# Sort results by revenue-to-budget ratio in descending order
actor_analysis = actor_analysis.sort_values(by='revenue_to_budget_ratio', ascending=False)

# Display top 10 actors by revenue-to-budget ratio
print(actor_analysis.head(10))
```

```
                 name  revenue_to_budget_ratio
465   Kathleen Turner            125002.035637
671  Richard Dreyfuss                52.988923
222   Donald Pleasence               50.122596
350  Jamie Lee Curtis                33.708473
685       Robert Shaw                28.884410
359       Jason Mewes                25.396149
105     Carrie Fisher                20.689612
91      Bruce Campbell               17.489618
697      Roy Scheider                16.997388
645      Peter Coyote                16.792969
```

---

13. Analyzing Kathleen Turner Data

Kathleen Turner's `revenue_to_budget_ratio` was much higher than expected. Let's pull her data to find the cause.

In [ ]:
```
# Filter combined DataFrame for Kathleen Turner's entries
kathleen_turner_df = filtered_combined_df[filtered_combined_df['name'] == 'Kathleen Turner']

# Display movies and their corresponding budgets and revenues
print(kathleen_turner_df[['title', 'budget', 'revenue', 'revenue_to_budget_ratio']])
```

```
                        title    budget    revenue  revenue_to_budget_ratio
3448               Marley & Me  60000000  244082376                 4.068040
5863         Dumb and Dumber To  40000000  169837010                 4.245925
8677             A Simple Wish  28000000    8345056                 0.298038
11668   Peggy Sue Got Married  18000000   41382841                 2.299047
13724           Baby Geniuses  12000000   36450736                 3.037561
13804              Serial Mom  13000000    7820688                 0.601591
15668               Nurse 3-D        10   10000000           1000000.000000
17363     The Virgin Suicides   6000000   10409377                 1.734896
```

---

14. Analyzing Top Actors by Revenue-to-Budget Ratio

We will continue by assigning a ranking to actors based on the order in which they appear in the `cast_extracted` column. Then, we will filter the top 5 actors for each movie and analyze the average revenue-to-budget ratio for these actors. We also need to eliminate placeholder in `budget` to avoid erroneous ratios.

In [ ]:
```
# Assuming original order of actors in 'cast_extracted' column indicates billing order,
# we'll assign a ranking based on order within each movie title.
cast_df['rank'] = cast_df.groupby('title').cumcount() + 1

# Filter to keep only first 5 actors (based on rank)
filtered_cast_df = cast_df[cast_df['rank'] <= 5]

# Merge filtered cast data with movies data on 'title' column
combined_df = pd.merge(filtered_cast_df, movies_df[['id', 'title', 'budget', 'revenue']], on='title', how='inner')

# Filter out rows where budget or revenue is zero, and ensure budget exceeds $100,000
combined_df = combined_df[(combined_df['budget'] > 100000) & (combined_df['revenue'] > 0)]

# Calculate revenue-to-budget ratio for each movie
combined_df['revenue_to_budget_ratio'] = combined_df['revenue'] / combined_df['budget']

# Filter out actors who have appeared in fewer than 5 movies
actor_movie_counts = combined_df['name'].value_counts()
actors_with_min_movies = actor_movie_counts[actor_movie_counts >= 5].index
filtered_combined_df = combined_df[combined_df['name'].isin(actors_with_min_movies)]

# Verify if ratio calculation and filtering are correct
print(filtered_combined_df[['name', 'title', 'revenue_to_budget_ratio']].head())

# Group by actor and calculate average revenue-to-budget ratio
actor_analysis = filtered_combined_df.groupby('name')['revenue_to_budget_ratio'].mean().reset_index()

# Sort results by revenue-to-budget ratio in descending order
actor_analysis = actor_analysis.sort_values(by='revenue_to_budget_ratio', ascending=False)
```

```python
# Display top 10 actors by revenue-to-budget ratio
print(actor_analysis.head(10))
```

```
               name                                      title  \
0     Sam Worthington                                     Avatar   
1         Zoe Saldana                                     Avatar   
2    Sigourney Weaver                                     Avatar   
4   Michelle Rodriguez                                    Avatar   
5         Johnny Depp  Pirates of the Caribbean: At World's End   

   revenue_to_budget_ratio  
0                11.763566  
1                11.763566  
2                11.763566  
4                11.763566  
5                 3.203333  
               name  revenue_to_budget_ratio
671  Richard Dreyfuss                52.988923
222  Donald Pleasence                50.122596
350  Jamie Lee Curtis                33.708473
685       Robert Shaw                28.884410
105     Carrie Fisher                20.689612
91      Bruce Campbell               17.489618
676     Robert Carlyle               17.449983
696      Roy Scheider                16.997388
645      Peter Coyote                16.792969
15      Alec Guinness                16.679081
```

---

16. Further Analysis on Top Actors

We will further refine our analysis by focusing on the top 10 actors with the highest revenue-to-budget ratios. We will extract and display all movies involving these top actors to gain deeper insights.

In [ ]:

```python
# Assuming original order of actors in 'cast_extracted' column indicates billing order,
# we'll assign a ranking based on order within each movie title.
cast_df['rank'] = cast_df.groupby('title').cumcount() + 1

# Filter to keep only first 5 actors (based on rank)
filtered_cast_df = cast_df[cast_df['rank'] <= 5]

# Merge filtered cast data with movies data on 'title' column
combined_df = pd.merge(filtered_cast_df, movies_df[['id', 'title', 'budget', 'revenue']], on='title', how='inner')

# Filter out rows where budget or revenue is zero, and ensure budget exceeds $100,000
combined_df = combined_df[(combined_df['budget'] > 100000) & (combined_df['revenue'] > 0)]

# Calculate revenue-to-budget ratio for each movie
combined_df['revenue_to_budget_ratio'] = combined_df['revenue'] / combined_df['budget']

# Filter out actors who have appeared in fewer than 5 movies
actor_movie_counts = combined_df['name'].value_counts()
actors_with_min_movies = actor_movie_counts[actor_movie_counts >= 5].index
filtered_combined_df = combined_df[combined_df['name'].isin(actors_with_min_movies)]

# Group by actor and calculate average revenue-to-budget ratio
actor_analysis = filtered_combined_df.groupby('name')['revenue_to_budget_ratio'].mean().reset_index()

# Sort results by revenue-to-budget ratio in descending order to get top 10 actors
top_10_actors = actor_analysis.sort_values(by='revenue_to_budget_ratio', ascending=False).head(10)

# Extract names of top 10 actors
top_10_actor_names = top_10_actors['name'].tolist()

# Filter combined DataFrame for movies involving top 10 actors
top_10_movies = filtered_combined_df[filtered_combined_df['name'].isin(top_10_actor_names)]

# Sort by actor and then by movie title for clarity
top_10_movies_sorted = top_10_movies.sort_values(by=['name', 'title'])

# Display all movies and their ratios for top 10 actors
for actor in top_10_actor_names:
 print(f"\nActor: {actor}")
 actor_movies = top_10_movies_sorted[top_10_movies_sorted['name'] == actor]
 for index, row in actor_movies.iterrows():
    print(f" Movie: {row['title']}, Revenue-to-Budget Ratio: {row['revenue_to_budget_ratio']:.2f}")
```

```
Actor: Richard Dreyfuss
 Movie: American Graffiti, Revenue-to-Budget Ratio: 180.18
 Movie: Close Encounters of the Third Kind, Revenue-to-Budget Ratio: 15.19
 Movie: Jaws, Revenue-to-Budget Ratio: 67.24
 Movie: My Life in Ruins, Revenue-to-Budget Ratio: 1.20
 Movie: Poseidon, Revenue-to-Budget Ratio: 1.14

Actor: Donald Pleasence
 Movie: Escape from New York, Revenue-to-Budget Ratio: 8.37
 Movie: Halloween, Revenue-to-Budget Ratio: 233.33
 Movie: Halloween 4: The Return of Michael Myers, Revenue-to-Budget Ratio: 3.55
 Movie: Halloween 5: The Revenge of Michael Myers, Revenue-to-Budget Ratio: 2.33
 Movie: Halloween: The Curse of Michael Myers, Revenue-to-Budget Ratio: 3.02

Actor: Jamie Lee Curtis
 Movie: Drowning Mona, Revenue-to-Budget Ratio: 0.96
 Movie: Freaky Friday, Revenue-to-Budget Ratio: 4.24
 Movie: Halloween, Revenue-to-Budget Ratio: 233.33
 Movie: Halloween: Resurrection, Revenue-to-Budget Ratio: 2.90
 Movie: The Fog, Revenue-to-Budget Ratio: 21.38
 Movie: The Tailor of Panama, Revenue-to-Budget Ratio: 1.33
 Movie: Trading Places, Revenue-to-Budget Ratio: 2.23
 Movie: True Lies, Revenue-to-Budget Ratio: 3.29

Actor: Robert Shaw
 Movie: A Man for All Seasons, Revenue-to-Budget Ratio: 7.27
 Movie: Force 10 from Navarone, Revenue-to-Budget Ratio: 1.45
 Movie: From Russia with Love, Revenue-to-Budget Ratio: 39.45
 Movie: Jaws, Revenue-to-Budget Ratio: 67.24
 Movie: The Sting, Revenue-to-Budget Ratio: 29.02

Actor: Carrie Fisher
 Movie: Return of the Jedi, Revenue-to-Budget Ratio: 17.70
 Movie: Star Wars, Revenue-to-Budget Ratio: 70.49
 Movie: The Empire Strikes Back, Revenue-to-Budget Ratio: 29.91
 Movie: Under the Rainbow, Revenue-to-Budget Ratio: 0.11
 Movie: Undiscovered, Revenue-to-Budget Ratio: 0.12
 Movie: When Harry Met Sally..., Revenue-to-Budget Ratio: 5.80

Actor: Bruce Campbell
 Movie: Evil Dead II, Revenue-to-Budget Ratio: 1.65
 Movie: My Name Is Bruce, Revenue-to-Budget Ratio: 0.12
 Movie: Serving Sara, Revenue-to-Budget Ratio: 0.58
 Movie: The Ant Bully, Revenue-to-Budget Ratio: 1.10
 Movie: The Evil Dead, Revenue-to-Budget Ratio: 84.00

Actor: Robert Carlyle
 Movie: 28 Weeks Later, Revenue-to-Budget Ratio: 4.28
 Movie: Eragon, Revenue-to-Budget Ratio: 2.49
 Movie: The Full Monty, Revenue-to-Budget Ratio: 73.67
 Movie: The World Is Not Enough, Revenue-to-Budget Ratio: 2.68
 Movie: Trainspotting, Revenue-to-Budget Ratio: 4.12

Actor: Roy Scheider
 Movie: Jaws, Revenue-to-Budget Ratio: 67.24
 Movie: Jaws 2, Revenue-to-Budget Ratio: 9.39
 Movie: Romeo Is Bleeding, Revenue-to-Budget Ratio: 0.28
 Movie: Sorcerer, Revenue-to-Budget Ratio: 0.55
 Movie: The French Connection, Revenue-to-Budget Ratio: 22.87
 Movie: The Punisher, Revenue-to-Budget Ratio: 1.66

Actor: Peter Coyote
 Movie: A Walk to Remember, Revenue-to-Budget Ratio: 3.75
 Movie: E.T. the Extra-Terrestrial, Revenue-to-Budget Ratio: 75.52
 Movie: Femme Fatale, Revenue-to-Budget Ratio: 0.48
 Movie: Patch Adams, Revenue-to-Budget Ratio: 4.05
 Movie: Sphere, Revenue-to-Budget Ratio: 0.17

Actor: Alec Guinness
 Movie: A Passage to India, Revenue-to-Budget Ratio: 3.40
 Movie: Doctor Zhivago, Revenue-to-Budget Ratio: 10.17
 Movie: Lawrence of Arabia, Revenue-to-Budget Ratio: 4.67
 Movie: Star Wars, Revenue-to-Budget Ratio: 70.49
 Movie: The Bridge on the River Kwai, Revenue-to-Budget Ratio: 11.10
 Movie: The Fall of the Roman Empire, Revenue-to-Budget Ratio: 0.25
```

---

17. Identifying and Analyzing the Top 10 Actors by Total Revenue-to-Budget Ratio

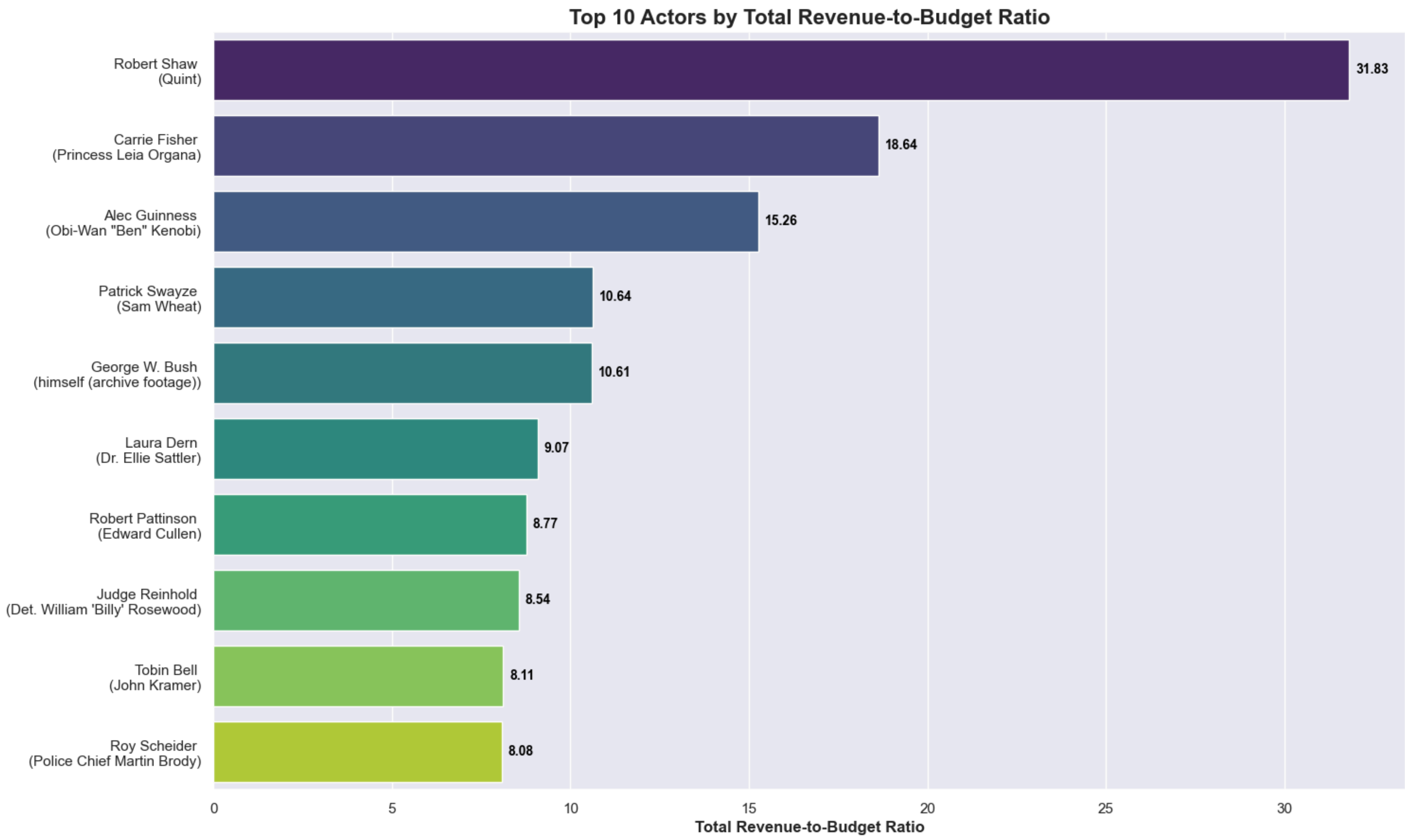
In this section, we aim to identify the top 10 actors based on their total revenue-to-budget ratio across all movies they have appeared in. The steps include:

1. **Assign Actor Ranking**: We assign a rank to actors within each movie based on their billing order (assuming the order in `cast_extracted` indicates billing order).
2. **Filter for Top 5 Actors per Movie**: We keep only the top 5 actors in each movie based on their rank.
3. **Merge with Movie Data**: We merge the filtered cast data with movie data to include information on budgets and revenues.
4. **Data Filtering**: We filter out movies where the budget or revenue is zero and ensure that the budget exceeds $10,000.
5. **Highest Revenue Movie per Actor**: We identify the movie that generated the highest revenue for each actor.
6. **Calculate Total Revenue and Budget**: We calculate the total revenue and total budget for each actor across all movies.

7. **Revenue-to-Budget Ratio**: We compute the total revenue-to-budget ratio for each actor.
8. **Filtering Actors**: We filter out actors who have appeared in fewer than 5 movies to focus on those with substantial filmography.
9. **Merge with Character Information**: We merge the data with the highest revenue movie information to include character names in our analysis. This is not necessary but may aid viewers by associating the actor's character.
10. **Visualization**: We create a bar plot to visualize the top 10 actors by total revenue-to-budget ratio.

This visualization helps us understand which actors have been most profitable in terms of the revenue their movies have generated relative to the budgets.

In [ ]:
```
# Assuming original order of actors in 'cast_extracted' column indicates billing order,
# we'll assign a ranking based on order within each movie title.
cast_df['rank'] = cast_df.groupby('title').cumcount() + 1

# Filter to keep only first 5 actors (based on rank)
filtered_cast_df = cast_df[cast_df['rank'] <= 5]

# Merge filtered cast data with movies data on 'title' column
combined_df = pd.merge(filtered_cast_df, movies_df[['id', 'title', 'budget', 'revenue']], on='title', how='inner')

# Filter out rows where budget or revenue is zero, and ensure budget exceeds $10,000
combined_df = combined_df[(combined_df['budget'] > 10000) & (combined_df['revenue'] > 0)]

# Identify highest revenue movie for each actor
idx = combined_df.groupby('name')['revenue'].idxmax()
highest_revenue_df = combined_df.loc[idx, ['name', 'title', 'revenue', 'budget', 'character']]

# Calculate total revenue and total budget for each actor
actor_totals = combined_df.groupby('name').agg(
   total_revenue=pd.NamedAgg(column='revenue', aggfunc='sum'),
   total_budget=pd.NamedAgg(column='budget', aggfunc='sum')
).reset_index()

# Calculate revenue-to-budget ratio for each actor
actor_totals['total_revenue_to_budget_ratio'] = actor_totals['total_revenue'] / actor_totals['total_budget']

# Filter out actors who have appeared in fewer than 5 movies
actor_movie_counts = combined_df['name'].value_counts()
actors_with_min_movies = actor_movie_counts[actor_movie_counts >= 5].index
actor_totals = actor_totals[actor_totals['name'].isin(actors_with_min_movies)]

# Merge with highest revenue movie info to include character names
actor_totals = pd.merge(actor_totals, highest_revenue_df[['name', 'character', 'title']], on='name')

# Combine actor name and character name for y-axis labels
actor_totals['name_with_character'] = actor_totals['name'] + ' \n(' + actor_totals['character'] + ')'

# Sort results by total revenue-to-budget ratio in descending order
actor_totals_sorted = actor_totals.sort_values(by='total_revenue_to_budget_ratio', ascending=False).head(10)

# Set up plot with a more manageable figure size
plt.figure(figsize=(14, 10), dpi=100)
ax = sns.barplot(x='total_revenue_to_budget_ratio', y='name_with_character', hue='name_with_character', data=actor_totals_sorted, palette='

# Customize plot
plt.title('Top 10 Actors by Total Revenue-to-Budget Ratio', fontsize=16, weight='bold')
plt.xlabel('Total Revenue-to-Budget Ratio', fontsize=12, weight='bold')
ax.set_ylabel('')

# Use ax.bar_label for more reliable text annotation positioning
for container in ax.containers:
   ax.bar_label(container, fmt='%.2f', label_type='edge', fontsize=10, color='black', weight='bold', padding=5)

# Adjust layout manually to avoid clipping
plt.subplots_adjust(left=0.05, right=0.95, top=0.9, bottom=0.1)

# Show plot
plt.show()

# Display top 10 actors by total revenue-to-budget ratio and their highest revenue character
print(actor_totals_sorted[['name', 'total_revenue_to_budget_ratio', 'character', 'title']].head(10))
```




```
               name  total_revenue_to_budget_ratio  \
685       Robert Shaw                      31.826884
105     Carrie Fisher                      18.642832
15      Alec Guinness                      15.261036
631    Patrick Swayze                      10.636271
281    George W. Bush                      10.606001
502        Laura Dern                       9.072935
683  Robert Pattinson                       8.767066
445    Judge Reinhold                       8.542249
772        Tobin Bell                       8.107859
696      Roy Scheider                       8.076311

                      character                                     title
685                       Quint                                      Jaws
105          Princess Leia Organa                                Star Wars
15          Obi-Wan "Ben" Kenobi                                Star Wars
631                   Sam Wheat                                     Ghost
281      himself (archive footage)                           Fahrenheit 9/11
502            Dr. Ellie Sattler                           Jurassic Park
683               Edward Cullen  The Twilight Saga: Breaking Dawn - Part 2
445  Det. William 'Billy' Rosewood                       Beverly Hills Cop
772                 John Kramer                                  Saw III
696      Police Chief Martin Brody                                    Jaws
```

---

18. Analyzing the Top 10 Movies by Revenue-to-Budget Ratio

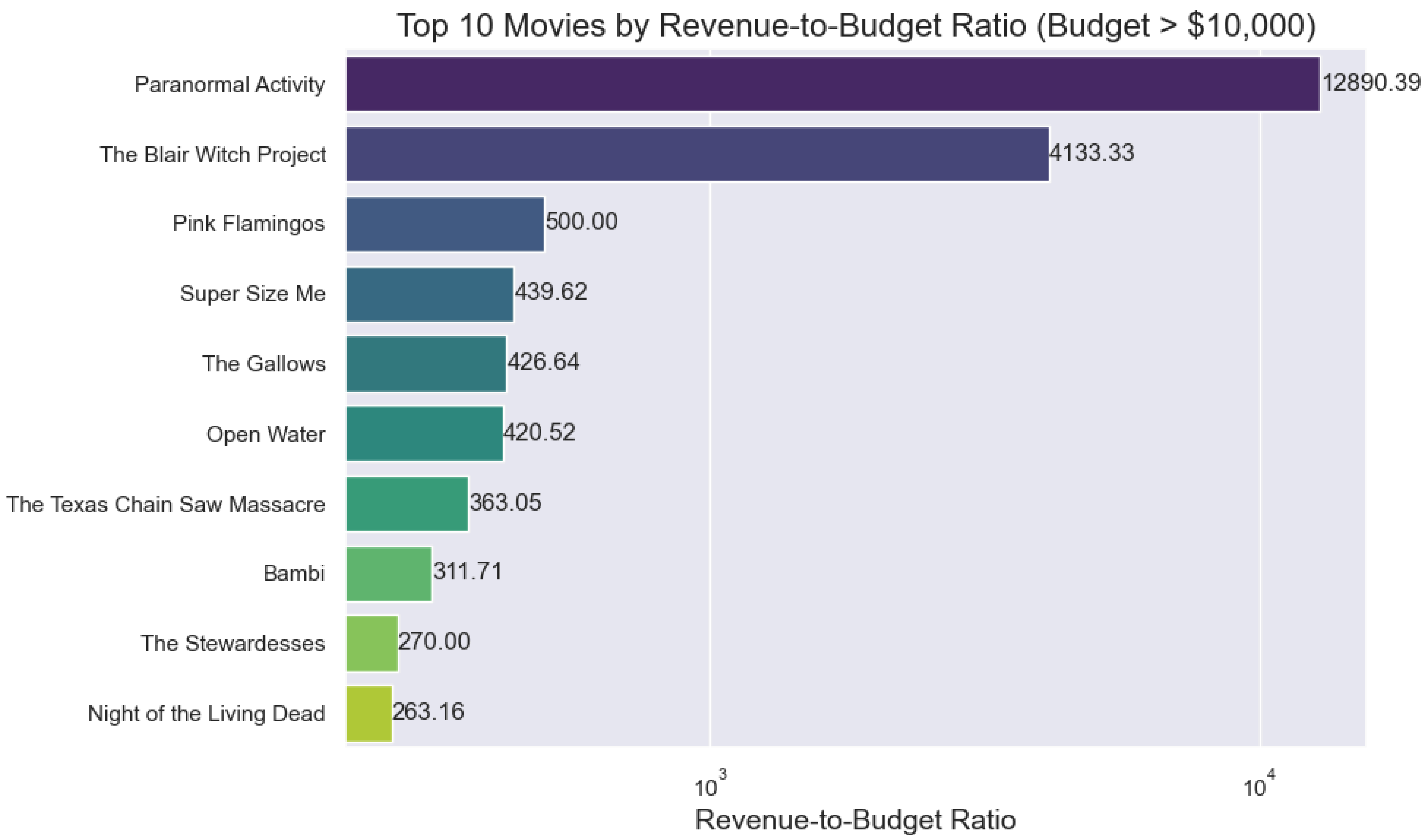
In this section, we shift our focus from actors to movies. We aim to identify the top 10 movies based on their revenue-to-budget ratio. The steps include:

1. **Load and Filter Movie Data**: We load the movie dataset and filter out entries where the budget or revenue is zero, ensuring the budget exceeds $10,000.
2. **Calculate Revenue-to-Budget Ratio**: For each movie, we compute the revenue-to-budget ratio.
3. **Identify Top 10 Movies**: We sort the movies by their revenue-to-budget ratio in descending order and select the top 10 movies.
4. **Visualization**: We create a bar plot to visualize these top 10 movies, allowing for a comparison of how efficiently each movie converted its budget into revenue.

This analysis is crucial to understanding which movies have been the most financially successful relative to their production costs.

In [ ]:
```
# Load movies data
movies_df = pd.read_csv('data/tmdb_5000_movies.csv')

# Filter out rows where budget or revenue is zero, and ensure budget exceeds $10,000
filtered_movies_df = movies_df[(movies_df['budget'] > 10000) & (movies_df['revenue'] > 0)].copy()

# Calculate revenue-to-budget ratio for each movie using .loc to avoid SettingWithCopyWarning
filtered_movies_df.loc[:, 'revenue_to_budget_ratio'] = filtered_movies_df['revenue'] / filtered_movies_df['budget']

# Sort results by revenue-to-budget ratio in descending order and select top 10
top_movies_sorted = filtered_movies_df.sort_values(by='revenue_to_budget_ratio', ascending=False).head(10)

# Set up plot
plt.figure(figsize=(10, 6))
ax = sns.barplot(x='revenue_to_budget_ratio', y='title', hue='title', data=top_movies_sorted[['title', 'revenue_to_budget_ratio']], palette
```

```python
# Customize plot
plt.title('Top 10 Movies by Revenue-to-Budget Ratio (Budget > $10,000)', fontsize=16)
plt.xlabel('Revenue-to-Budget Ratio', fontsize=14)
ax.set_ylabel('')
plt.xscale('log') # Log scale to better visualize differences

# Annotate bars with exact ratio values
for index, value in enumerate(top_movies_sorted['revenue_to_budget_ratio']):
 plt.text(value, index, f'{value:.2f}', va='center')

# Show plot
plt.tight_layout()
plt.show()

# Display top 10 movies by revenue-to-budget ratio
print(top_movies_sorted[['title', 'revenue_to_budget_ratio']].head(10))
```




```
                         title  revenue_to_budget_ratio
4577       Paranormal Activity             12890.386667
4496   The Blair Witch Project              4133.333333
4788            Pink Flamingos               500.000000
4742             Super Size Me               439.616585
4723               The Gallows               426.644100
4514                Open Water               420.522723
3159  The Texas Chain Saw Massacre           363.047059
4441                     Bambi               311.709965
4668          The Stewardesses               270.000000
3737    Night of the Living Dead             263.157895
```

---

19. Identifying and Visualizing the Top 5 Most Profitable Movies

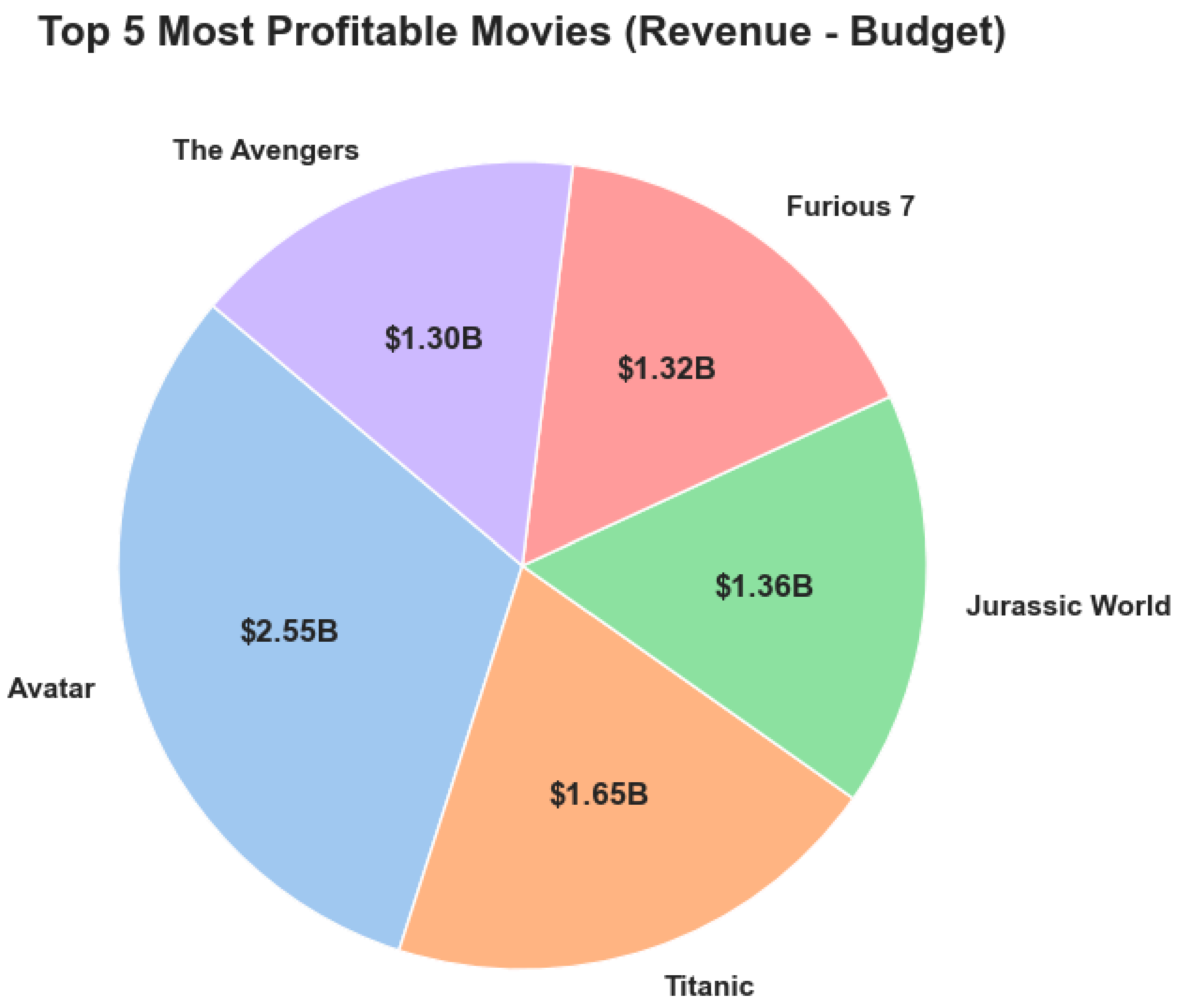
In this section, we focus on identifying the top 5 most profitable movies, where profitability is defined as the difference between the revenue and the budget. The steps involved are:

1. **Prepare Unique Movie Data**:
   - We start by ensuring the `combined_df` is prepared with the required columns: `title`, `budget`, and `revenue`.
   - We drop any duplicate movie titles, keeping only the entry with the highest revenue for each title.
2. **Calculate Profit**:
   - We convert the `revenue` and `budget` columns to `float64` for accurate calculations.
   - We compute the profit for each movie, converting the result into billions of dollars and rounding it to two decimal places.
3. **Identify Top 5 Profitable Movies**:
   - We sort the movies by profit in descending order and select the top 5.
4. **Visualize with a Pie Chart**:
   - We create a pie chart to visualize the distribution of profit among the top 5 movies.
   - The pie chart includes labels for each movie title and displays the profit as a percentage of the total profit, along with the exact profit value in billions of dollars.
5. **Display the Results**:

- We display the names and profits of the top 5 most profitable movies.

This analysis highlights which movies were the most financially successful in terms of absolute profit, providing insights into the most lucrative titles in the dataset.

In [ ]:
```python
# Assuming combined_df is already prepared and contains 'title', 'budget', and 'revenue'
# Drop duplicate titles, keeping entry with highest revenue
unique_movies_df = combined_df.drop_duplicates(subset='title', keep='first').copy()

# Set column type before calculation
unique_movies_df = unique_movies_df.astype({'revenue': 'float64', 'budget': 'float64'})

# Calculate profit for each unique movie, convert to billions, float 2 decimals
unique_movies_df['profit'] = ((unique_movies_df['revenue'] - unique_movies_df['budget']) / 1e9).round(2)

# Sort by profit and select top 5 most profitable movies
top_profitable_movies = unique_movies_df.sort_values(by='profit', ascending=False).head(5)

# Prepare data for pie chart
labels = top_profitable_movies['title']
sizes = top_profitable_movies['profit']

# Create pie chart
plt.figure(figsize=(10, 7))
plt.pie(sizes, labels=labels, autopct=lambda p: f'${p * sum(sizes) / 100:.2f}B', startangle=140,
        colors=sns.color_palette('pastel', len(labels)), textprops={'weight': 'bold'})
plt.title('Top 5 Most Profitable Movies (Revenue - Budget)', fontsize=16, weight='bold')
plt.show()

# Display top 5 most profitable movies with profit values in billions
formatted_output = top_profitable_movies.apply(lambda x: f'{x["title"]}: ${x["profit"]:.2f}B', axis=1)
print(formatted_output)
```




```
0              Avatar: $2.55B
125           Titanic: $1.65B
140    Jurassic World: $1.36B
220         Furious 7: $1.32B
80       The Avengers: $1.30B
dtype: object
```