

Exploratory Data Analysis for Microsoft's New Movie Studio

Overview

In this project, we will perform exploratory data analysis (EDA) to generate insights for Microsoft, helping them decide what type of films to create for their new movie studio. We will use various datasets related to movies, analyze trends, and provide actionable recommendations based on our findings.

Business Problem

Microsoft sees all the big companies creating original video content and they want to get in on the fun. They have decided to create a new movie studio but lack knowledge about creating movies. Our task is to explore what types of films are currently doing the best at the box office and translate those findings into actionable insights for Microsoft's new movie studio.

Objectives

The objectives of this project are to:

1. Compare budget to profitability.
2. Identify the top-performing genres.
3. Assess profitability by genre.

Loading Data

```
In [ ]: import pandas as pd
import numpy as np
import sqlite3
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import ScalarFormatter

In [ ]: bom_movie_gross = pd.read_csv('data/bom.movie_gross.csv')
rt_movie_info = pd.read_csv('data/rt.movie_info.tsv', sep='\t')
rt_reviews = pd.read_csv('data/rt.reviews.tsv', sep='\t', encoding='ISO-8859-1')
tmdb_movies = pd.read_csv('data/tmdb.movies.csv')
tn_movie_budgets = pd.read_csv('data/tn.movie_budgets.csv')

conn = sqlite3.connect('data/im.db')
imdb_movie_basics = pd.read_sql_query("SELECT * FROM movie_basics", conn)
imdb_movie_ratings = pd.read_sql_query("SELECT * FROM movie_ratings", conn)
```

Displaying the first few rows of each dataframe.

```
In [ ]: print("Movie Gross Data:")
print(bom_movie_gross.head())
```

```

print("-----")
print("Movie Info Data:")
print(rt_movie_info.head())
print("-----")

print("Reviews Data:")
print(rt_reviews.head())
print("-----")

print("TMDB Movies Data:")
print(tmdb_movies.head())
print("-----")

print("Movie Budgets Data:")
print(tn_movie_budgets.head())
print("-----")

print("Movie Basics Data:")
print(imdb_movie_basics.head())
print("-----")

print("Movie Ratings Data:")
print(imdb_movie_ratings.head())

```

Movie Gross Data:

		title	studio	domestic_gross	\
0		Toy Story 3	BV	415000000.0	
1		Alice in Wonderland (2010)	BV	334200000.0	
2	Harry Potter and the Deathly Hallows Part 1		WB	296000000.0	
3		Inception	WB	292600000.0	
4		Shrek Forever After	P/DW	238700000.0	

	foreign_gross	year
0	652000000	2010
1	691300000	2010
2	664300000	2010
3	535700000	2010
4	513900000	2010

Movie Info Data:

		synopsis	rating	\
0	1	This gritty, fast-paced, and innovative police...	R	
1	3	New York City, not-too-distant-future: Eric Pa...	R	
2	5	Illeana Douglas delivers a superb performance ...	R	
3	6	Michael Douglas runs afoul of a treacherous su...	R	
4	7		NaN	NR

	genre	director	\
0	Action and Adventure Classics Drama	William Friedkin	
1	Drama Science Fiction and Fantasy	David Cronenberg	
2	Drama Musical and Performing Arts	Allison Anders	
3	Drama Mystery and Suspense	Barry Levinson	
4	Drama Romance	Rodney Bennett	

	writer	theater_date	dvd_date	currency	\
0	Ernest Tidyman	Oct 9, 1971	Sep 25, 2001	NaN	
1	David Cronenberg Don DeLillo	Aug 17, 2012	Jan 1, 2013	\$	
2	Allison Anders	Sep 13, 1996	Apr 18, 2000	NaN	
3	Paul Attanasio Michael Crichton	Dec 9, 1994	Aug 27, 1997	NaN	
4	Giles Cooper	NaN	NaN	NaN	

box_office	runtime	studio
------------	---------	--------

0 NaN 104 minutes NaN
 1 600,000 108 minutes Entertainment One
 2 NaN 116 minutes NaN
 3 NaN 128 minutes NaN
 4 NaN 200 minutes NaN

Reviews Data:

	id	review	rating	fresh	\
0	3	A distinctly gallows take on contemporary fina...	3/5	fresh	
1	3	It's an allegory in search of a meaning that n...	NaN	rotten	
2	3	... life lived in a bubble in financial dealin...	NaN	fresh	
3	3	Continuing along a line introduced in last yea...	NaN	fresh	
4	3	... a perverse twist on neorealism...	NaN	fresh	

	critic	top_critic	publisher	date
0	PJ Naborro	0	Patrick Naborro	November 10, 2018
1	Annalee Newitz	0	io9.com	May 23, 2018
2	Sean Axmaker	0	Stream on Demand	January 4, 2018
3	Daniel Kasman	0	MUBI	November 16, 2017
4	NaN	0	Cinema Scope	October 12, 2017

TMDB Movies Data:

	Unnamed: 0	genre_ids	id	original_language	\
0	0	[12, 14, 10751]	12444	en	
1	1	[14, 12, 16, 10751]	10191	en	
2	2	[12, 28, 878]	10138	en	
3	3	[16, 35, 10751]	862	en	
4	4	[28, 878, 12]	27205	en	

	original_title	popularity	release_date	\
0	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	
1	How to Train Your Dragon	28.734	2010-03-26	
2	Iron Man 2	28.515	2010-05-07	
3	Toy Story	28.005	1995-11-22	
4	Inception	27.920	2010-07-16	

	title	vote_average	vote_count
0	Harry Potter and the Deathly Hallows: Part 1	7.7	10788
1	How to Train Your Dragon	7.7	7610
2	Iron Man 2	6.8	12368
3	Toy Story	7.9	10174
4	Inception	8.3	22186

Movie Budgets Data:

	id	release_date	movie	\
0	1	Dec 18, 2009	Avatar	
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	
2	3	Jun 7, 2019	Dark Phoenix	
3	4	May 1, 2015	Avengers: Age of Ultron	
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	

	production_budget	domestic_gross	worldwide_gross
0	\$425,000,000	\$760,507,625	\$2,776,345,279
1	\$410,600,000	\$241,063,875	\$1,045,663,875
2	\$350,000,000	\$42,762,350	\$149,762,350
3	\$330,600,000	\$459,005,868	\$1,403,013,963
4	\$317,000,000	\$620,181,382	\$1,316,721,747

Movie Basics Data:

	movie_id	primary_title	original_title	\
0	tt0063540	Sunghursh	Sunghursh	
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	

```

start_year runtime_minutes genres
0 2013 175.0 Action,Crime,Drama
1 2019 114.0 Biography,Drama
2 2018 122.0 Drama
3 2018 NaN Comedy,Drama
4 2017 80.0 Comedy,Drama,Fantasy
-----
Movie Ratings Data:
    movie_id averagerating numvotes
0 tt10356526 8.3 31
1 tt10384606 8.9 559
2 tt1042974 6.4 20
3 tt1043726 4.2 50352
4 tt1060240 6.5 21

```

Upon initial examination of the dataframes, the following datasets were selected as they contain the necessary information to meet the project objectives:

- bom_movie_gross
- tn_movie_budgets
- imdb_movie_basics
- imdb_movie_ratings

Inspecting the selected datasets

1. bom_movie_gross

```
In [ ]: #Examining the total number of rows, columns, non-null values and datatypes in the data;
bom_movie_gross.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   title        3387 non-null   object  
 1   studio       3382 non-null   object  
 2   domestic_gross 3359 non-null   float64 
 3   foreign_gross 2037 non-null   object  
 4   year         3387 non-null   int64   
dtypes: float64(1), int64(1), object(3)
memory usage: 132.4+ KB
```

```
In [ ]: # Checking for null values
bom_movie_gross.isna().sum()
```

```
Out[ ]: title          0
studio         5
domestic_gross 28
foreign_gross  1350
year           0
dtype: int64
```

```
In [ ]: #checking for duplicates
bom_movie_gross.duplicated().sum()
```

```
Out[ ]: 0
```

2. tn_movie_budgets

```
In [ ]: #Examining the total number of rows, columns, non-null values and datatypes in the data;
tn_movie_budgets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id               5782 non-null    int64  
 1   release_date     5782 non-null    object  
 2   movie             5782 non-null    object  
 3   production_budget 5782 non-null    object  
 4   domestic_gross    5782 non-null    object  
 5   worldwide_gross   5782 non-null    object  
dtypes: int64(1), object(5)
memory usage: 271.2+ KB
```

```
In [ ]: # Checking for null values
tn_movie_budgets.isna().sum()
```

```
Out[ ]: id              0
release_date      0
movie             0
production_budget 0
domestic_gross    0
worldwide_gross   0
dtype: int64
```

```
In [ ]: #checking for duplicates
tn_movie_budgets.duplicated().sum()
```

```
Out[ ]: 0
```

3. imdb_movie_basics

```
In [ ]: #Examining the total number of rows, columns, non-null values and datatypes in the data;
imdb_movie_basics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   movie_id         146144 non-null  object  
 1   primary_title    146144 non-null  object  
 2   original_title   146123 non-null  object  
 3   start_year       146144 non-null  int64  
 4   runtime_minutes  114405 non-null  float64 
 5   genres            140736 non-null  object  
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
```

```
In [ ]: # Checking for null values
imdb_movie_basics.isna().sum()
```

```
Out[ ]: movie_id          0
primary_title        0
original_title       21
start_year           0
runtime_minutes     31739
genres                5408
dtype: int64
```

```
In [ ]: #checking for duplicates
imdb_movie_basics.duplicated().sum()
```

Out[]: 0

4. imdb_movie_ratings

```
In [ ]: #Examining the total number of rows, columns, non-null values and datatypes in the data
imdb_movie_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
---  --          -----          --    
 0   movie_id    73856 non-null   object  
 1   averagerating 73856 non-null   float64 
 2   numvotes     73856 non-null   int64   
dtypes: float64(1), int64(1), object(1)
memory usage: 1.7+ MB
```

```
In [ ]: # Checking for null values
imdb_movie_ratings.isna().sum()
```

```
Out[ ]: movie_id      0
averagerating  0
numvotes       0
dtype: int64
```

```
In [ ]: #checking for duplicates
imdb_movie_ratings.duplicated().sum()
```

Out[]: 0

Upon inspecting the datasets, the bom_movie_gross dataframe was dropped since its data is available in other datasets. Additionally, the foreign_gross column had 1,350 missing values out of a total of 3,387 rows.

Data Cleaning

Missing Values

imdb_movie_basics

```
In [ ]: #checking the percentages of the missing data
missing_data = imdb_movie_basics.isnull().sum()

missing_percentage = (missing_data / len(imdb_movie_basics)) * 100

missing_data_df = pd.DataFrame({
    'Missing Values': missing_data,
    'Percentage': missing_percentage
}).sort_values(by='Percentage', ascending=False)

print(missing_data_df)
```

	Missing Values	Percentage
runtime_minutes	31739	21.717621
genres	5408	3.700460
original_title	21	0.014369
movie_id	0	0.000000
primary_title	0	0.000000
start_year	0	0.000000

There are 21 missing values in the original_title column, 31,739 missing values in the runtime_minutes column, and 5,408 missing values in the genres column. These were dropped due to their relatively small number compared to the entire dataset.

```
In [ ]: imdb_movie_basics.dropna(inplace=True)
```

```
In [ ]: imdb_movie_basics.isna().sum()
```

```
Out[ ]: movie_id      0
primary_title    0
original_title   0
start_year       0
runtime_minutes  0
genres           0
dtype: int64
```

```
In [ ]: #merging the imdb dataframes
imdb_merged = pd.merge(imdb_movie_basics, imdb_movie_ratings, how='inner', on='movie_id'

#checking the first five rows of the merged dataframe
imdb_merged.head()
```

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	average_rating
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	7.0
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	6.8
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	6.8
3	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	6.5
4	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	Adventure,Animation,Comedy	6.5

```
In [ ]: #checking the shape of the merged object
imdb_merged.shape
```

```
Out[ ]: (65720, 8)
```

After cleaning the data, the remaining dataframes containing the cleaned data are:

- tn_movie_budgets
- imdb_merged

Creating New Variables for Analysis

Analyzing revenue

```
In [ ]: #converting production_budget, domestic_gross, and worldwide_gross columns in tn_movie_budgets
tn_movie_budgets['production_budget'] = tn_movie_budgets['production_budget'].replace('R', '')
tn_movie_budgets['domestic_gross'] = tn_movie_budgets['domestic_gross'].replace('[\$,]', '')
tn_movie_budgets['worldwide_gross'] = tn_movie_budgets['worldwide_gross'].replace('[\$,]', '')

#calculating profit
tn_movie_budgets['domestic_profit'] = tn_movie_budgets['domestic_gross'] - tn_movie_budgets['production_budget']
tn_movie_budgets['worldwide_profit'] = tn_movie_budgets['worldwide_gross'] - tn_movie_budgets['production_budget']

#calculating return on investment
tn_movie_budgets['domestic_roi'] = (tn_movie_budgets['domestic_profit'] / tn_movie_budgets['production_budget']) * 100
tn_movie_budgets['worldwide_roi'] = (tn_movie_budgets['worldwide_profit'] / tn_movie_budgets['production_budget']) * 100

#Extracting year from release date
tn_movie_budgets['release_year'] = pd.to_datetime(tn_movie_budgets['release_date']).dt.year

tn_movie_budgets.head()
```

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	domestic_profit	worldwide_profit
0	1	Dec 18, 2009	Avatar	425000000	760507625	2776345279	335507625	241063875
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	-169536125	149762350
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350	-307237650	459005868
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	1403013963	128405868	620181382
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747	303181382	1316721747

Separating Genres

```
In [ ]: #using one-hot encoding to separate the genres

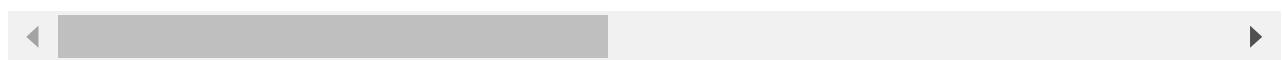
genres = imdb_merged['genres'].str.get_dummies(sep=',')
imdb_merged = pd.concat([imdb_merged, genres], axis=1)
```

```
In [ ]: imdb_merged.head()
```

Out[]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	averagerating
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	7.0
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	7.0
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	7.0
3	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	7.0
4	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	Adventure,Animation,Comedy	7.0

5 rows × 34 columns

In []: `#merging the tn_movie_budgets and the imdb_merged dataframe`

```
tn_movie_budgets = tn_movie_budgets.rename(columns={'movie': 'primary_title'})
combined_data = pd.merge(tn_movie_budgets, imdb_merged, on='primary_title', how='inner')
```

In []: `combined_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2752 entries, 0 to 2751
Data columns (total 44 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id               2752 non-null    int64  
 1   release_date     2752 non-null    object  
 2   primary_title    2752 non-null    object  
 3   production_budget 2752 non-null    int64  
 4   domestic_gross   2752 non-null    int64  
 5   worldwide_gross  2752 non-null    int64  
 6   domestic_profit  2752 non-null    int64  
 7   worldwide_profit 2752 non-null    int64  
 8   domestic_roi     2752 non-null    float64 
 9   worldwide_roi    2752 non-null    float64 
 10  release_year    2752 non-null    int64  
 11  movie_id         2752 non-null    object  
 12  original_title   2752 non-null    object  
 13  start_year       2752 non-null    int64  
 14  runtime_minutes  2752 non-null    float64 
 15  genres            2752 non-null    object  
 16  averagerating    2752 non-null    float64 
 17  numvotes          2752 non-null    int64  
 18  Action             2752 non-null    int64  
 19  Adult              2752 non-null    int64  
 20  Adventure          2752 non-null    int64  
 21  Animation          2752 non-null    int64  
 22  Biography           2752 non-null    int64  
 23  Comedy              2752 non-null    int64  
 24  Crime               2752 non-null    int64  
 25  Documentary         2752 non-null    int64
```

```

26 Drama          2752 non-null int64
27 Family         2752 non-null int64
28 Fantasy        2752 non-null int64
29 Game-Show      2752 non-null int64
30 History        2752 non-null int64
31 Horror          2752 non-null int64
32 Music           2752 non-null int64
33 Musical          2752 non-null int64
34 Mystery         2752 non-null int64
35 News            2752 non-null int64
36 Reality-TV       2752 non-null int64
37 Romance          2752 non-null int64
38 Sci-Fi           2752 non-null int64
39 Short             2752 non-null int64
40 Sport             2752 non-null int64
41 Thriller          2752 non-null int64
42 War               2752 non-null int64
43 Western            2752 non-null int64
dtypes: float64(4), int64(35), object(5)
memory usage: 967.5+ KB

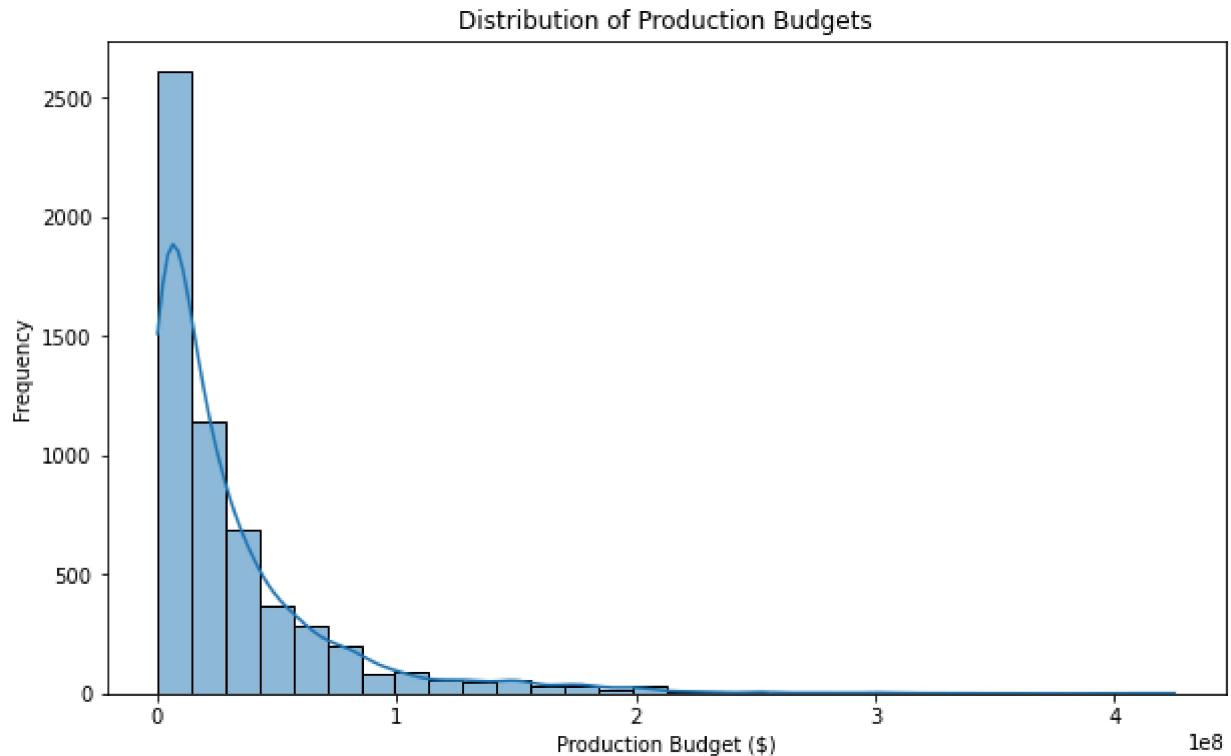
```

Exploratory Data Analysis (EDA)

Examining the distribution of the variables

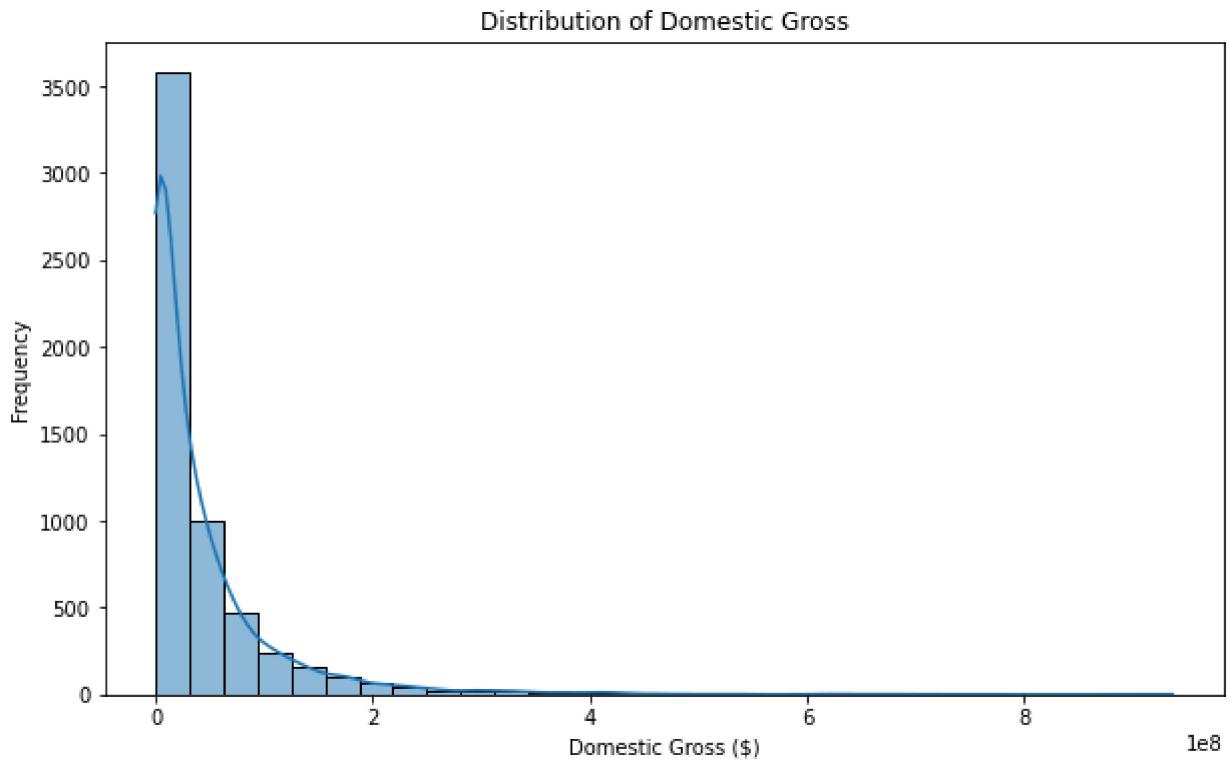
Distribution of Production Budgets

```
In [ ]: plt.figure(figsize=(10, 6))
sns.histplot(tn_movie_budgets['production_budget'], bins=30, kde=True)
plt.title('Distribution of Production Budgets')
plt.xlabel('Production Budget ($)')
plt.ylabel('Frequency')
plt.show()
```



Distribution of Domestic Gross

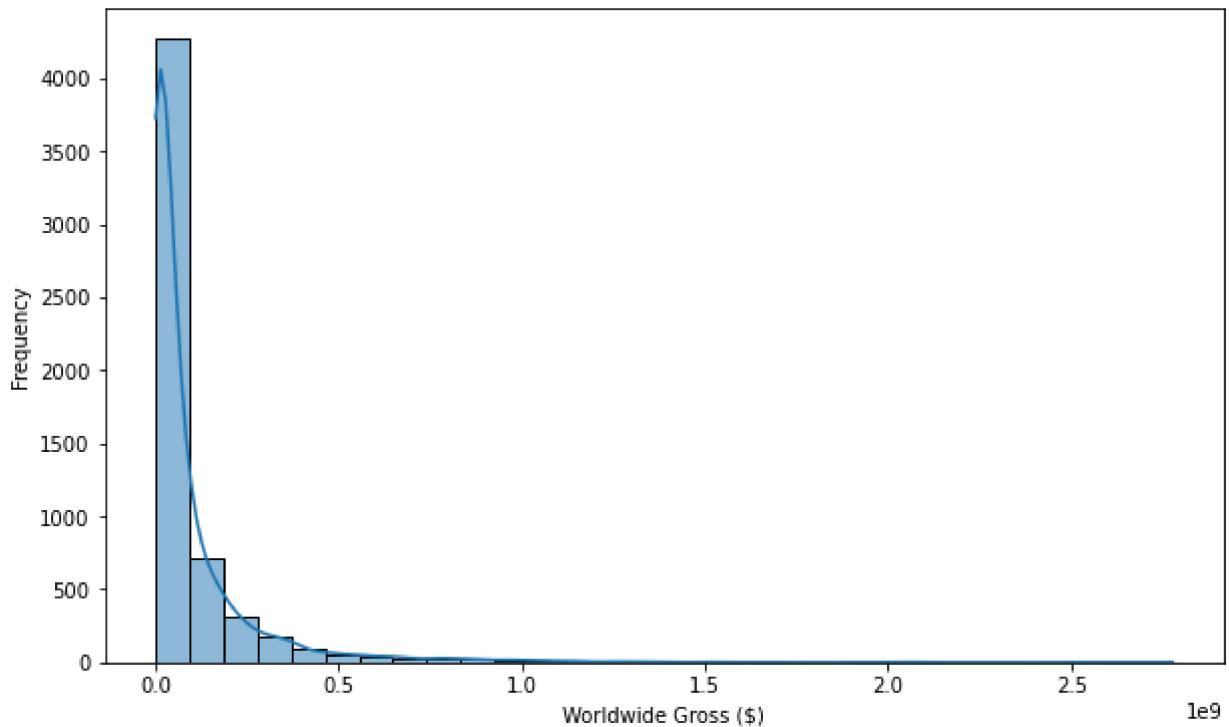
```
In [ ]: plt.figure(figsize=(10, 6))
sns.histplot(tn_movie_budgets['domestic_gross'], bins=30, kde=True)
plt.title('Distribution of Domestic Gross')
plt.xlabel('Domestic Gross ($)')
plt.ylabel('Frequency')
plt.show()
```



Distribution of Worldwide Gross

```
In [ ]: plt.figure(figsize=(10, 6))
sns.histplot(tn_movie_budgets['worldwide_gross'], bins=30, kde=True)
plt.title('Distribution of Worldwide Gross')
plt.xlabel('Worldwide Gross ($)')
plt.ylabel('Frequency')
plt.show()
```

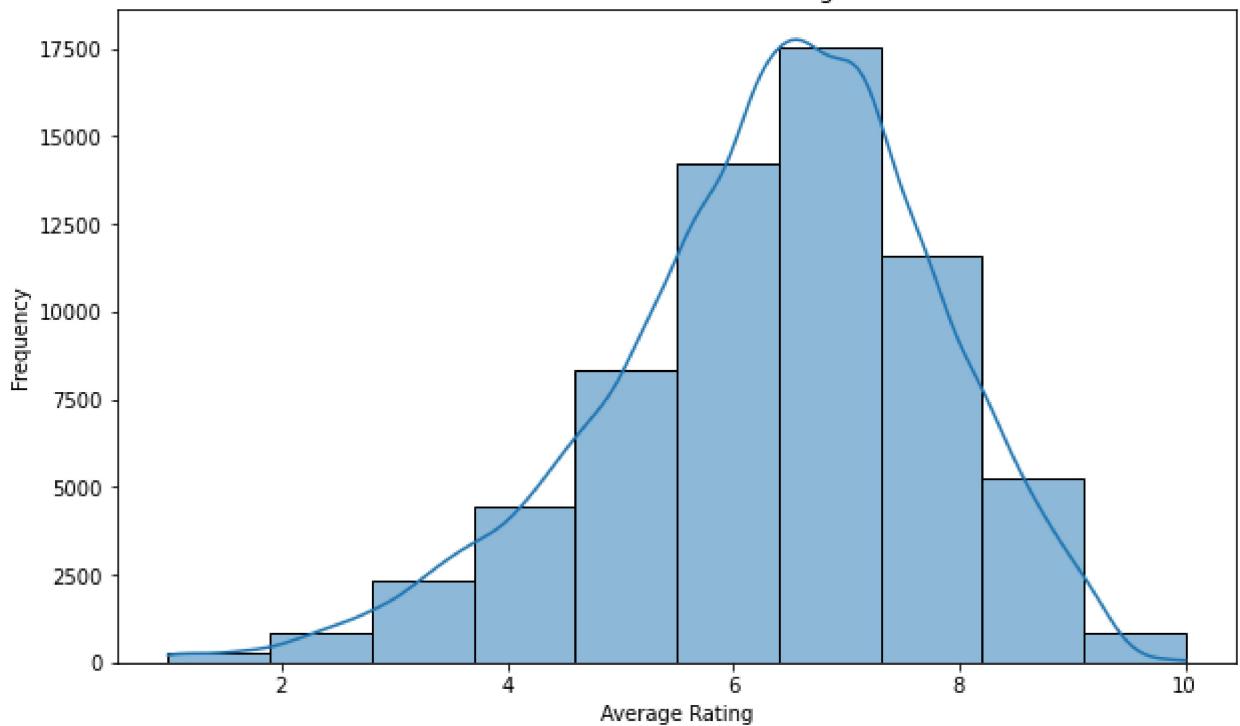
Distribution of Worldwide Gross



Distribution of IMDB Ratings

```
In [ ]: plt.figure(figsize=(10, 6))
sns.histplot(imdb_merged['averagerating'], bins=10, kde=True)
plt.title('Distribution of IMDB Ratings')
plt.xlabel('Average Rating')
plt.ylabel('Frequency')
plt.show()
```

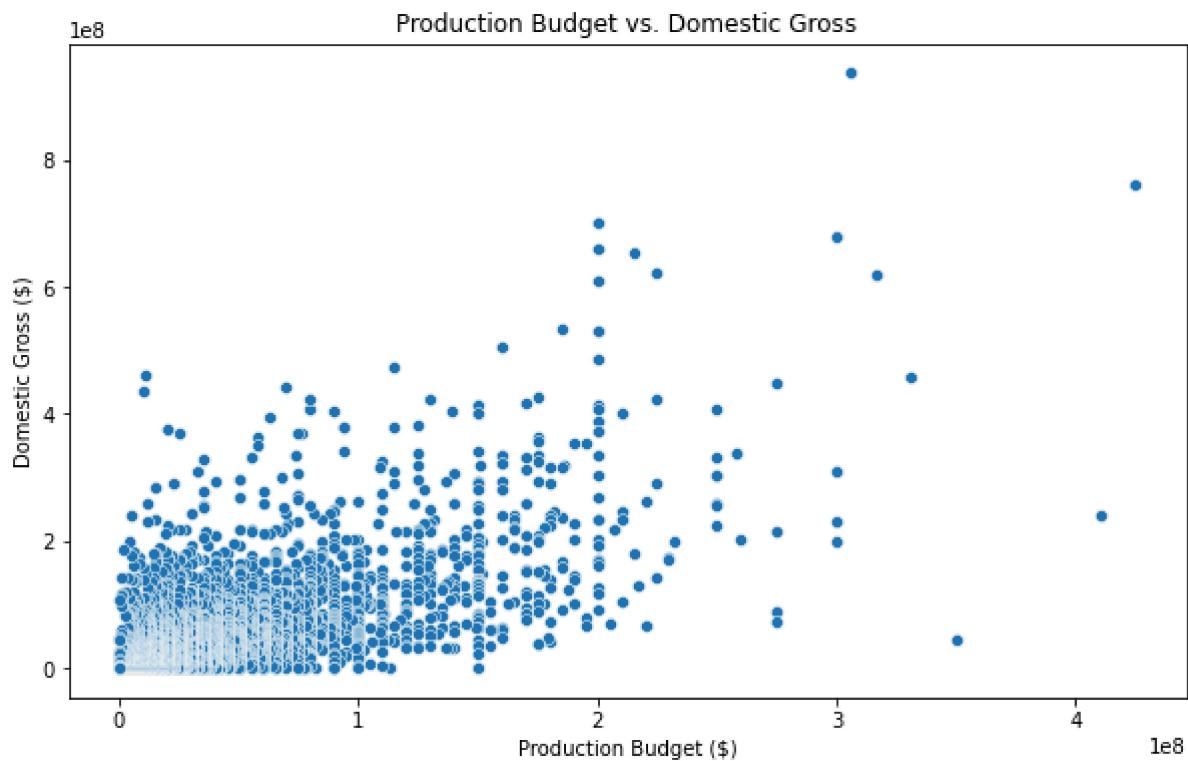
Distribution of IMDB Ratings



Comparing the variables

Production Budget vs. Domestic Gross

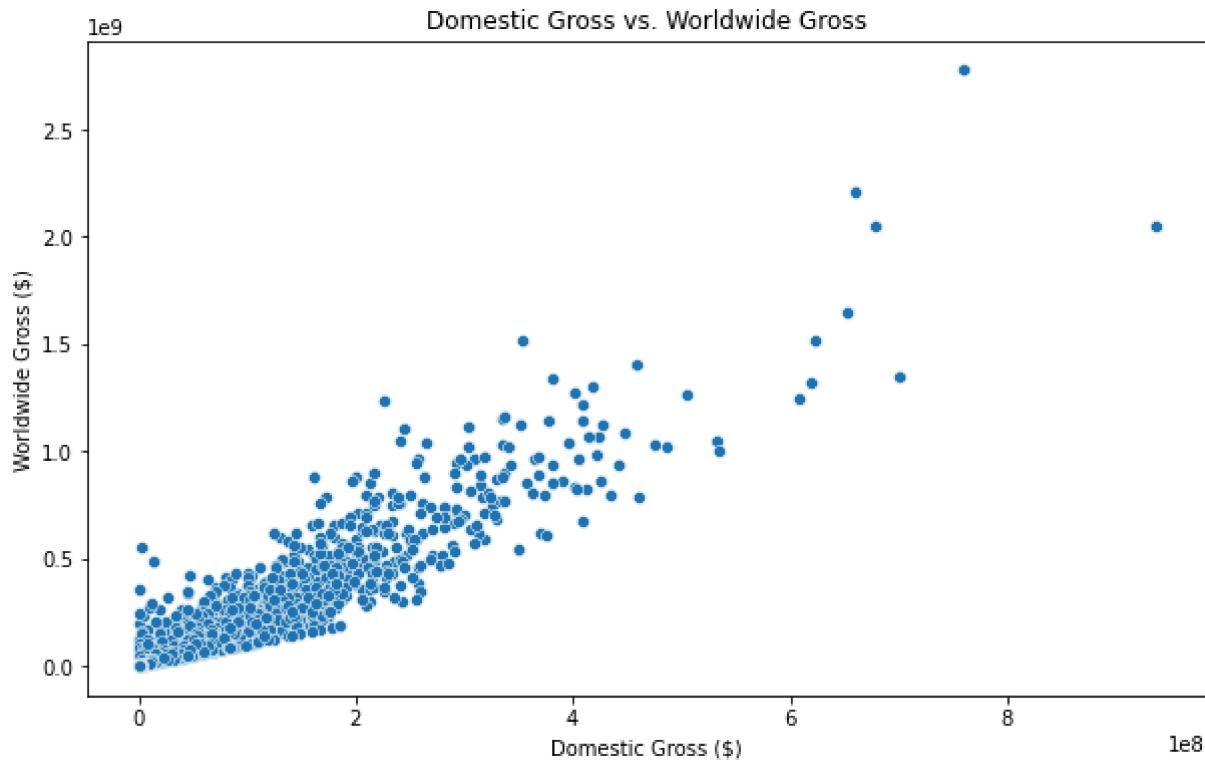
```
In [ ]: plt.figure(figsize=(10, 6))
sns.scatterplot(x='production_budget', y='domestic_gross', data=tn_movie_budgets)
plt.title('Production Budget vs. Domestic Gross')
plt.xlabel('Production Budget ($)')
plt.ylabel('Domestic Gross ($)')
plt.show()
```



The scatter plot shows that as budgets increase, domestic revenues also tend to go up.

Domestic Gross vs. Worldwide Gross

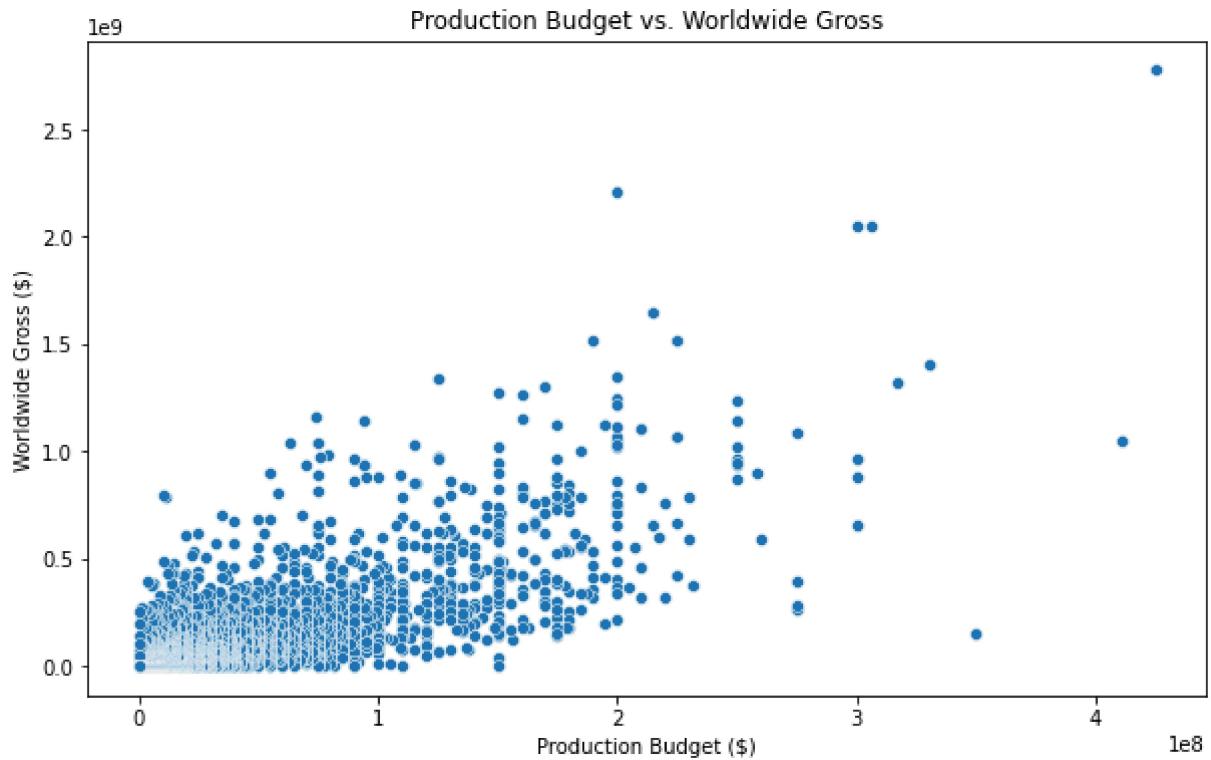
```
In [ ]: plt.figure(figsize=(10, 6))
sns.scatterplot(x='domestic_gross', y='worldwide_gross', data=tn_movie_budgets)
plt.title('Domestic Gross vs. Worldwide Gross')
plt.xlabel('Domestic Gross ($)')
plt.ylabel('Worldwide Gross ($)')
plt.show()
```



The scatter plot shows an upward trend, suggesting that higher domestic gross tend to yield higher worldwide revenues.

Production Budget vs. Worldwide Gross

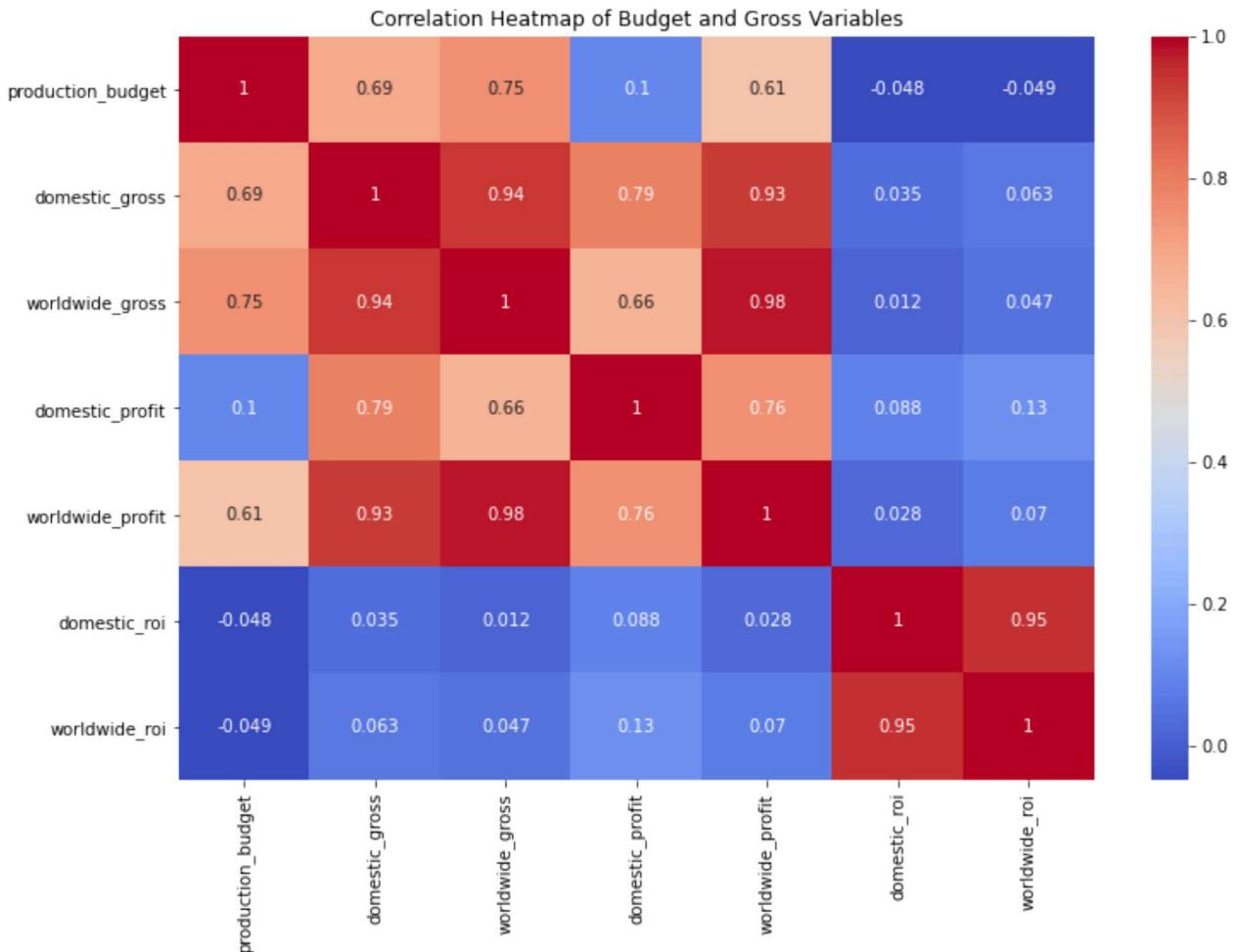
```
In [ ]: plt.figure(figsize=(10, 6))
sns.scatterplot(x='production_budget', y='worldwide_gross', data=tn_movie_budgets)
plt.title('Production Budget vs. Worldwide Gross')
plt.xlabel('Production Budget ($)')
plt.ylabel('Worldwide Gross ($)')
plt.show()
```



The scatter plot suggests that as budgets increase, worldwide revenues also tend to go up.

Multivariate Analysis: Heatmap of Correlations in tn_movie_budgets

```
In [ ]: plt.figure(figsize=(12, 8))
sns.heatmap(tn_movie_budgets[['production_budget', 'domestic_gross', 'worldwide_gross'],
plt.title('Correlation Heatmap of Budget and Gross Variables')
plt.show()
```



There is a strong positive correlation between production budget and gross revenues for both domestic and worldwide, and between gross revenues and profits.

Identifying the most profitable genres

```
In [ ]: genre_columns = ['Action', 'Adult', 'Adventure', 'Animation', 'Biography', 'Comedy', 'Cr
genre_profit_df = pd.DataFrame()

for genre in genre_columns:
    genre_profit_df.loc[genre, 'avg_domestic_profit'] = combined_data.loc[combined_data
    genre_profit_df.loc[genre, 'avg_worldwide_profit'] = combined_data.loc[combined_dat

genre_profit_df = genre_profit_df.reset_index().rename(columns={'index': 'genre'})

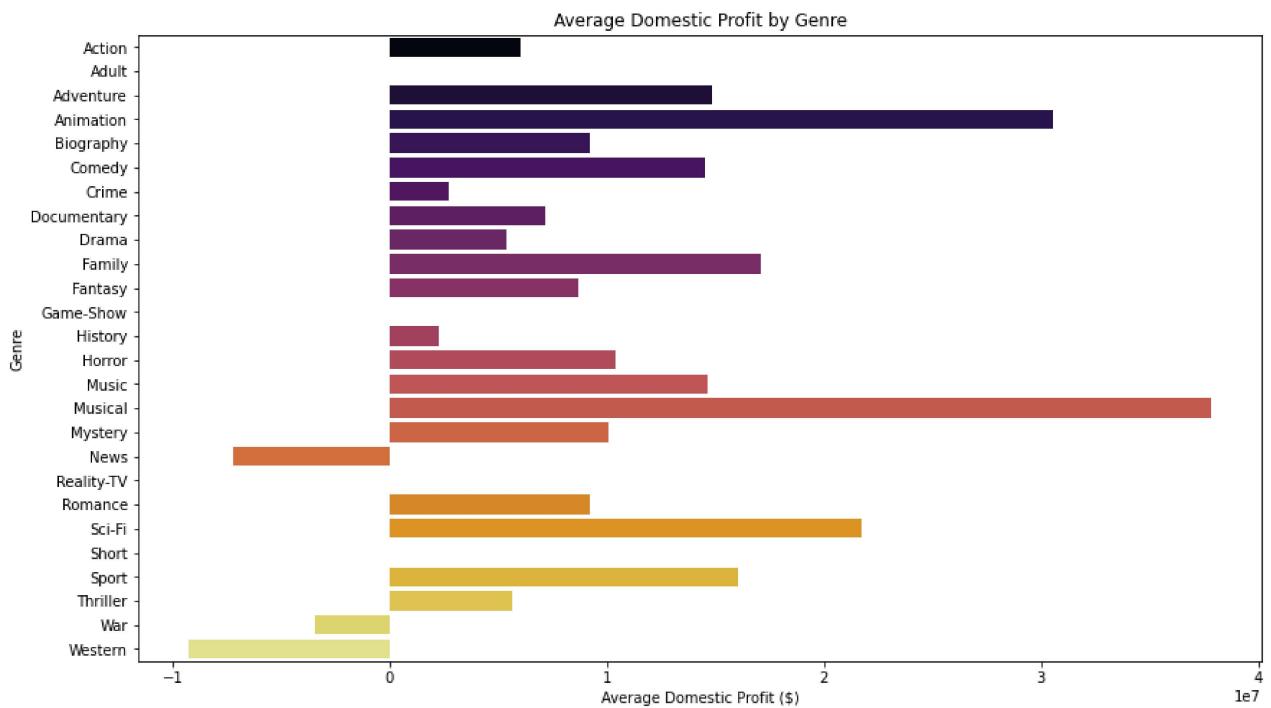
genre_profit_df
```

```
Out[ ]:
```

	genre	avg_domestic_profit	avg_worldwide_profit
0	Action	6.009706e+06	1.277744e+08
1	Adult	NaN	NaN
2	Adventure	1.482112e+07	2.059963e+08
3	Animation	3.054870e+07	2.347626e+08
4	Biography	9.240125e+06	4.685618e+07

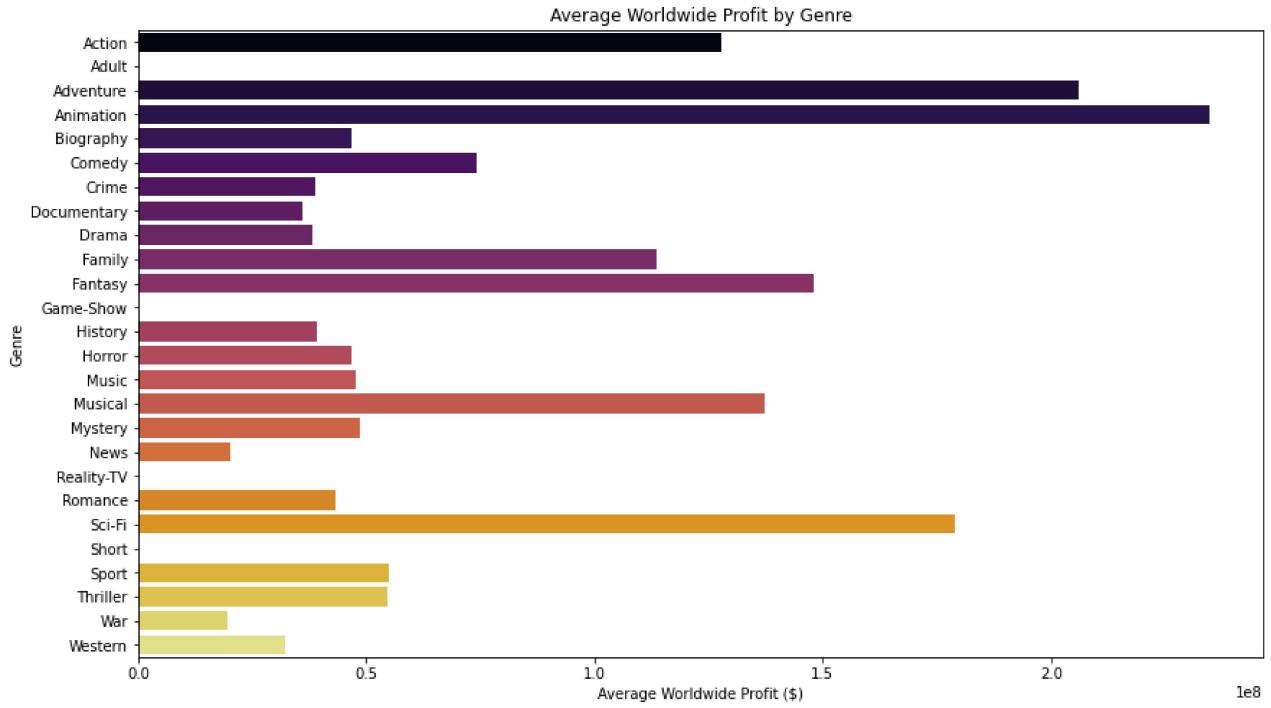
	genre	avg_domestic_profit	avg_worldwide_profit
5	Comedy	1.453542e+07	7.415582e+07
6	Crime	2.745059e+06	3.870504e+07
7	Documentary	7.144900e+06	3.589637e+07
8	Drama	5.382601e+06	3.806185e+07
9	Family	1.706532e+07	1.135383e+08
10	Fantasy	8.690764e+06	1.480807e+08
11	Game-Show	NaN	NaN
12	History	2.249297e+06	3.919284e+07
13	Horror	1.037547e+07	4.676660e+07
14	Music	1.466438e+07	4.775460e+07
15	Musical	3.780470e+07	1.374225e+08
16	Mystery	1.006663e+07	4.876315e+07
17	News	-7.196259e+06	2.008208e+07
18	Reality-TV	NaN	NaN
19	Romance	9.182224e+06	4.312345e+07
20	Sci-Fi	2.172206e+07	1.789648e+08
21	Short	NaN	NaN
22	Sport	1.602454e+07	5.482649e+07
23	Thriller	5.625565e+06	5.449042e+07
24	War	-3.428482e+06	1.947261e+07
25	Western	-9.238052e+06	3.231587e+07

```
In [ ]: plt.figure(figsize=(14, 8))
sns.barplot(x='avg_domestic_profit', y='genre', data=genre_profit_df, palette='inferno'
plt.title('Average Domestic Profit by Genre')
plt.xlabel('Average Domestic Profit ($)')
plt.ylabel('Genre')
plt.show()
```



Average Worldwide Profit by Genre

```
In [ ]: plt.figure(figsize=(14, 8))
sns.barplot(x='avg_worldwide_profit', y='genre', data=genre_profit_df, palette='inferno'
plt.title('Average Worldwide Profit by Genre')
plt.xlabel('Average Worldwide Profit ($)')
plt.ylabel('Genre')
plt.show()
```



The genres with the tallest bars are the most profitable on a global scale. Animation, adventure and Sci-Fi have the tallest bars. These genres generate the highest average worldwide profits.

Conclusion and Recommendations

Conclusion:

The analysis reveals that certain genres consistently perform better both domestically and worldwide. Action, Science Fiction, and Adventure films generally generate the highest average revenues and profits. Furthermore, the analysis of different budget ranges reveals that mid-range to high-range budgets often result in the most profitable movies.

Recommendations:

1. Prioritize Profitable Genres: Concentrate your efforts and resources on developing films within the Action, Science Fiction, and Adventure genres, as the analysis indicates these genres consistently generate the highest profits.

2. Strategic Budget Management: When allocating budgets for film productions, aim for the mid range. This budget range has been shown to yield the highest profits. Steer clear of excessively high budgets unless the potential returns are projected to be remarkably high and well-justified.