Analyzing Box Office Success Factors : Data-Driven Insights for Strategic Movie Production

Introduction

The global film industry is highly competitive, with major studios investing significant resources in producing movies that appeal to audiences and generate high returns at the box office. As new entrants seek to establish themselves, data-driven decision-making has become a critical factor in reducing risk and identifying opportunities for success.

This project applies exploratory data analysis (EDA) techniques to movie industry datasets from multiple sources, including Box Office Mojo, IMDB, Rotten Tomatoes, TheMovieDB, and The Numbers. The aim is to uncover patterns and insights into what types of films perform best at the box office. By analyzing features such as genres, directors, production budgets, revenues, ratings, and runtimes, the project seeks to generate actionable recommendations for a business stakeholder—the head of a new movie studio—who must decide what films to produce.

Through statistical analysis, visualization, and hypothesis testing, this project not only highlights trends in movie performance but also builds a foundation for predictive modeling in later phases. Ultimately, the findings will guide strategic decisions on which genres and production factors are most likely to yield commercial success.

1. Business Understanding

Stakeholders

The primary stakeholders for this analysis are:

1. Head of movie studio: seeks to draw Data-driven insights that will help shape the studio’s production strategy, ensuring alignment with market demand.
2. Investment team: seeks to Understand what types of films perform well, reduces financial risk and guides profitable investment decisions.
3. Operations team: seeks to Identify trends in successful film production to help streamline planning, budgeting, and resource allocation.
4. Risk management team: seeks to draw Insights into past film failures and successes to help minimize financial and reputational risks in new projects.
5. Monitoring, Evaluation & Learning team: Performance metrics and trend analysis support continuous improvement and informed decision-making over time.

Business Value

This analysis delivers key strategic advantages to support the successful launch and growth of our movie studio:

1. Market Alignment: By identifying the genres, themes, and characteristics of high-performing films, we ensure our productions resonate with current audience preferences and market demand.
2. Investment Efficiency: Insights into budget-to-revenue trends allow us to optimize investment decisions, maximizing returns while minimizing wasteful spending on low-potential projects.
3. Content Strategy Development: A data-driven understanding of what works at the box office empowers the studio to build a focused, high-impact film portfolio from the outset.
4. Competitive Positioning: Leveraging historical box office data enables us to benchmark against industry leaders and craft a unique value proposition in a saturated content market.
5. Risk Reduction: Analyzing past failures and successes helps us avoid common pitfalls in film production, reducing creative and financial risk in a volatile industry.

Project goals

1. Identify Box Office Success Drivers
2. Understand Market Trends
3. Develop Actionable Production Insights
4. Support Strategic Investment Decisions
5. Lay a Foundation for Data-Driven Content Strategy

Data Source

We will be analyzing data from multiple movie industry databases, including Box Office Mojo, IMDb, and The Movie Database (TMDb). These sources contain detailed information about film characteristics, box office revenue, audience ratings, and production details for thousands of movies released globally.

This comprehensive dataset will allow us to:

1. Analyze box office performance across different genres and budget ranges
2. Identify patterns in successful film characteristics (e.g., runtime, cast, ratings)
3. To determine if release timing and marketing factors significantly affect a film's worldwide revenue?
4. Track changes in audience preferences and industry trends over time

Specific Objectives are:

1. Determine if movie budget can affect its revenue 2.To determine what patterns emerge from audience ratings and runtimes, and how they affect a movie's worldwide gross
2. Identify whether movie genres and directors significantly explain variation in worldwide gross (returns), and determine which ones consistently outperform others

2. DATA UNDERSTANDING

Data Source

We will be analyzing data from the following websites and databases that majorly house infomation about movies. They include: Rotten Tomatoes, Box Office Mojo, IMDb, The Movie DB and The Numbers. The goal is to identify trends in high-performing movies at in these datasets to guide a new studio in choosing what types of films to produce for the best chance of commercial success.

Initial Data Exploration

Import the relevant libraries and read the dataset

```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import gzip
import sqlite3
import warnings
warnings.filterwarnings('ignore')
```

Exploring the Box Office Movies Dataset

```
print("=== Box Office Mojo Dataset Analysis ===")

# Load Box Office Mojo dataset
BOMdf = pd.read_csv('zippedData/bom.movie_gross.csv.gz')

# Display basic information
print("\nFirst 5 rows of data:")
print(BOMdf.head())

# Display dataset info
print("\nDataset Overview:")
print(f"Number of records: {len(BOMdf)}")
print(f"Number of columns: {len(BOMdf.columns)}")
BOMdf.info(memory_usage='deep')

# Check for data quality issues
print("\nData Quality Analysis:")
duplicates = BOMdf.duplicated().sum()
print(f"Duplicate records: {duplicates}
({(duplicates/len(BOMdf))*100:.2f}%)")

# Check for missing values
null_counts = BOMdf.isnull().sum()
```

```
print("\nMissing Values:")
for col in BOMdf.columns:
    if null_counts[col] > 0:
        print(f"{col}: {null_counts[col]} missing
({(null_counts[col]/len(BOMdf))*100:.2f}%)")
```

```
=== Box Office Mojo Dataset Analysis ===

First 5 rows of 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  

Dataset Overview:
Number of records: 3387
Number of columns: 5
<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: 666.5 KB

Data Quality Analysis:
Duplicate records: 0 (0.00%)

Missing Values:
studio: 5 missing (0.15%)
```

```
domestic_gross: 28 missing (0.83%)
foreign_gross: 1350 missing (39.86%)
```

Box Mojo dataset has missing values, foreign_gross column having 39.86% missing values while the rest have less than 1% missing values

```
BOMdf
```

```
                                            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
...                                           ...         ...
...
3382                                    The Quake       Magn.
6200.0
3383                  Edward II (2018 re-release)          FM
4800.0
3384                                     El Pacto        Sony
2500.0
3385                                     The Swan  Synergetic
2400.0
3386                          An Actor Prepares       Grav.
1700.0

     foreign_gross  year
0        652000000  2010
1        691300000  2010
2        664300000  2010
3        535700000  2010
4        513900000  2010
...            ...   ...
3382           NaN  2018
3383           NaN  2018
3384           NaN  2018
3385           NaN  2018
3386           NaN  2018

[3387 rows x 5 columns]
```

data quality summary.

```
print("=== Box Office Mojo Summary Statistics ===\n")

# summary statistics
summary_stats = BOMdf.describe()
print(summary_stats.round(2))

# Distribution analysis
print("\nDistribution Analysis:")
for col in BOMdf.select_dtypes(include=['float64', 'int64']).columns:
    q1 = BOMdf[col].quantile(0.25)
    q3 = BOMdf[col].quantile(0.75)
    iqr = q3 - q1
    print(f"\n{col.replace('_', ' ').title()}:")
    print(f"IQR: ${iqr:,.2f}")
    print(f"Skewness: {BOMdf[col].skew():.2f}")
```

```
=== Box Office Mojo Summary Statistics ===

       domestic_gross     year
count    3.359000e+03  3387.00
mean     2.874585e+07  2013.96
std      6.698250e+07     2.48
min      1.000000e+02  2010.00
25%      1.200000e+05  2012.00
50%      1.400000e+06  2014.00
75%      2.790000e+07  2016.00
max      9.367000e+08  2018.00

Distribution Analysis:

Domestic Gross:
IQR: $27,780,000.00
Skewness: 4.72

Year:
IQR: $4.00
Skewness: -0.01
```

Exploring the IMDB Dataset

Import the `zipfile` module to help access the zipped `im.db` sqlite database

```
# unzip the im.db.zip file and eztract the database file
import zipfile

with zipfile.ZipFile('zippedData/im.db.zip', 'r') as zip_ref:
    zip_ref.extractall('zippedData/')
```

Create a connection `conn` to the database using `sqlite3` module

```
#load the dataset
conn = sqlite3.connect('zippedData/im.db')
#check the tables in the database
tables = pd.read_sql("""SELECT name FROM sqlite_master WHERE
type='table';""", conn)
print(tables)
```

```
             name
0   movie_basics
1      directors
2      known_for
3     movie_akas
4  movie_ratings
5        persons
6     principals
7        writers
```

data quality summary

```
print("=== Movie Basics Analysis ===")

# Load movie basics table
movie_basics = pd.read_sql("""
    SELECT * FROM movie_basics
    ORDER BY start_year DESC;
""", conn)

# Display basic information
print("\nDataset Overview:")
movie_basics.info(memory_usage='deep')

# Display summary statistics
print("\nNumerical Column Statistics:")
print(movie_basics.describe().round(2))

# Check for data quality issues
print("\nData Quality Analysis:")

# Check for duplicates
duplicates = movie_basics.duplicated().sum()
print(f"Duplicate records: {duplicates}
({(duplicates/len(movie_basics))*100:.2f}%)")

# Check for missing values
null_counts = movie_basics.isnull().sum()
print("\nMissing Values:")
for col in movie_basics.columns:
    if null_counts[col] > 0:
```

```python
        print(f"{col}: {null_counts[col]} missing ({(null_counts[col]/len(movie_basics))*100:.2f}%)")

# Analyze categorical columns
print("\nUnique values in categorical columns:")
for col in movie_basics.select_dtypes(include=['object']).columns:
    print(f"{col}: {movie_basics[col].nunique()} unique values")
```

```
=== Movie Basics Analysis ===

Dataset Overview:
<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: 42.3 MB

Numerical Column Statistics:
       start_year  runtime_minutes
count   146144.00        114405.00
mean      2014.62            86.19
std          2.73           166.36
min       2010.00             1.00
25%       2012.00            70.00
50%       2015.00            87.00
75%       2017.00            99.00
max       2115.00         51420.00

Data Quality Analysis:
Duplicate records: 0 (0.00%)

Missing Values:
original_title: 21 missing (0.01%)
runtime_minutes: 31739 missing (21.72%)
genres: 5408 missing (3.70%)

Unique values in categorical columns:
movie_id: 146144 unique values
primary_title: 136071 unique values
original_title: 137773 unique values
genres: 1085 unique values
```

```
movie_basics

         movie_id                                    primary_title  \
0       tt5174640                                        100 Years
1       tt5637536                                         Avatar 5
2      tt10300398                           Untitled Star Wars Film
3       tt3095356                                         Avatar 4
4      tt10300396                           Untitled Star Wars Film
...           ...                                              ...
146139  tt9852508                                  Viyapath Bambara
146140  tt9875120                                        Frostbite
146141  tt9875242                                     15 Fotografii
146142  tt9878374                 Regi lagni comprensorio di stato
146143  tt9905932  Footloose in London: All the Best Sights of ou...

                                          original_title  start_year  \
0                                              100 Years        2115
1                                               Avatar 5        2027
2                                Untitled Star Wars Film        2026
3                                               Avatar 4        2025
4                                Untitled Star Wars Film        2024
...                                                  ...         ...
146139                                  Viyapath Bambara        2010
146140                                         Frostbite        2010
146141                                     15 Fotografii        2010
146142                  Regi lagni comprensorio di stato        2010
146143  Footloose in London: All the Best Sights of ou...        2010
```

```
        runtime_minutes                    genres
0                   NaN                     Drama
1                   NaN  Action,Adventure,Fantasy
2                   NaN                   Fantasy
3                   NaN  Action,Adventure,Fantasy
4                   NaN                      None
...                 ...                       ...
146139            120.0                     Drama
146140             90.0               Documentary
146141             56.0                     Drama
146142              NaN               Documentary
146143            106.0                      None

[146144 rows x 6 columns]
```

Loading and analysing of the `movie_ratings` dataframe.

```
print("=== Movie Ratings Analysis ===")

# Load movie ratings table with basic statistics
movie_ratings = pd.read_sql("""
    SELECT * FROM movie_ratings;
""", conn)

# Display basic information
print("\nDataset Overview:")
movie_ratings.info()

# Display summary statistics
print("\nRating Statistics:")
print(movie_ratings.describe().round(2))

# Check for data quality issues
print("\nData Quality Analysis:")

# Check for duplicates
duplicates = movie_ratings.duplicated().sum()
print(f"Duplicate records: {duplicates}
({(duplicates/len(movie_ratings))*100:.2f}%)")

# Check for missing values
null_counts = movie_ratings.isnull().sum()
print("\nMissing Values:")
for col in movie_ratings.columns:
    if null_counts[col] > 0:
        print(f"{col}: {null_counts[col]} missing
({(null_counts[col]/len(movie_ratings))*100:.2f}%)")

# Check rating distribution
print("\nRating Distribution:")
```

```python
print(movie_ratings['averagerating'].value_counts(bins=5).sort_index()
)
```

```
=== Movie Ratings Analysis ===

Dataset Overview:
<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

Rating Statistics:
       averagerating    numvotes
count       73856.00    73856.00
mean            6.33     3523.66
std             1.47    30294.02
min             1.00        5.00
25%             5.50       14.00
50%             6.50       49.00
75%             7.40      282.00
max            10.00  1841066.00

Data Quality Analysis:
Duplicate records: 0 (0.00%)

Missing Values:

Rating Distribution:
(0.99, 2.8]     1531
(2.8, 4.6]      8271
(4.6, 6.4]     26424
(6.4, 8.2]     31561
(8.2, 10.0]     6069
Name: averagerating, dtype: int64
```

```python
movie_ratings
```

	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
...	...	...	...

```
73851   tt9805820            8.1         25
73852   tt9844256            7.5         24
73853   tt9851050            4.7         14
73854   tt9886934            7.0          5
73855   tt9894098            6.3        128

[73856 rows x 3 columns]
```

has 73856 rows and 3 columns

Loading and analysing of the "Reviews" dataframe.

```
#Load the dataset
df_reviews = pd.read_csv('zippedData/rt.reviews.tsv.gz', sep = '\t',
encoding='latin1')
df_reviews.head()

   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 Nabarro           0   Patrick Nabarro  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

print("=== Reviews Dataset Analysis ===")
print("\nDataFrame Info:")
df_reviews.info()

# Check for duplicate records
duplicates = df_reviews.duplicated().sum()
print(f"\nDuplicate records: {duplicates}
({(duplicates/len(df_reviews))*100:.2f}% of total)")

# Display null value counts and percentages
null_counts = df_reviews.isnull().sum()
null_percentages = (null_counts/len(df_reviews))*100
print("\nMissing Values Analysis:")
```

```
for col in df_reviews.columns:
    if null_counts[col] > 0:
        print(f"{col}: {null_counts[col]} missing values
({null_percentages[col]:.2f}%)")

# Display unique values in categorical columns
print("\nUnique values in categorical columns:")
for col in df_reviews.select_dtypes(include=['object']).columns:
    print(f"{col}: {df_reviews[col].nunique()} unique values")
```

```
=== Reviews Dataset Analysis ===

DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54432 entries, 0 to 54431
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   id          54432 non-null  int64
 1   review      48869 non-null  object
 2   rating      40915 non-null  object
 3   fresh       54432 non-null  object
 4   critic      51710 non-null  object
 5   top_critic  54432 non-null  int64
 6   publisher   54123 non-null  object
 7   date        54432 non-null  object
dtypes: int64(2), object(6)
memory usage: 3.3+ MB

Duplicate records: 9 (0.02% of total)

Missing Values Analysis:
review: 5563 missing values (10.22%)
rating: 13517 missing values (24.83%)
critic: 2722 missing values (5.00%)
publisher: 309 missing values (0.57%)

Unique values in categorical columns:
review: 48682 unique values
rating: 186 unique values
fresh: 2 unique values
critic: 3496 unique values
publisher: 1281 unique values
date: 5963 unique values
```

Loading and analysing of the Movie_info dataframe.

```
#load the dataset
Movie_info_df = pd.read_csv('zippedData/rt.movie_info.tsv.gz', sep =
```

```
'\t', encoding='latin1')
Movie_info_df.head()

   id                                           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

print("=== Movie Info Dataset Analysis ===")
print("\nDataFrame Info:")
Movie_info_df.info(memory_usage='deep')

# Check for duplicate records
duplicates = Movie_info_df.duplicated().sum()
print(f"\nDuplicate records: {duplicates}
({(duplicates/len(Movie_info_df))*100:.2f}% of total)")

# Display null value counts and percentages
null_counts = Movie_info_df.isnull().sum()
null_percentages = (null_counts/len(Movie_info_df))*100
print("\nMissing Values Analysis:")
```

```python
for col in Movie_info_df.columns:
    if null_counts[col] > 0:
        print(f"{col}: {null_counts[col]} missing values
({null_percentages[col]:.2f}%)")

# Display memory usage
print("\nMemory Usage:")
print(Movie_info_df.memory_usage(deep=True).sum() / 1024**2, "MB")

# Check for invalid or unexpected values in key columns
print("\nUnique values in categorical columns:")
for col in Movie_info_df.select_dtypes(include=['object']).columns:
    print(f"{col}: {Movie_info_df[col].nunique()} unique values")
```

```
=== Movie Info Dataset Analysis ===

DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1560 entries, 0 to 1559
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            1560 non-null   int64
 1   synopsis      1498 non-null   object
 2   rating        1557 non-null   object
 3   genre         1552 non-null   object
 4   director      1361 non-null   object
 5   writer        1111 non-null   object
 6   theater_date  1201 non-null   object
 7   dvd_date      1201 non-null   object
 8   currency      340 non-null    object
 9   box_office    340 non-null    object
 10  runtime       1530 non-null   object
 11  studio        494 non-null    object
dtypes: int64(1), object(11)
memory usage: 1.9 MB

Duplicate records: 0 (0.00% of total)

Missing Values Analysis:
synopsis: 62 missing values (3.97%)
rating: 3 missing values (0.19%)
genre: 8 missing values (0.51%)
director: 199 missing values (12.76%)
writer: 449 missing values (28.78%)
theater_date: 359 missing values (23.01%)
dvd_date: 359 missing values (23.01%)
currency: 1220 missing values (78.21%)
box_office: 1220 missing values (78.21%)
runtime: 30 missing values (1.92%)
```

```
studio: 1066 missing values (68.33%)

Memory Usage:
1.9296875 MB

Unique values in categorical columns:
synopsis: 1497 unique values
rating: 6 unique values
genre: 299 unique values
director: 1125 unique values
writer: 1069 unique values
theater_date: 1025 unique values
dvd_date: 717 unique values
currency: 1 unique values
box_office: 336 unique values
runtime: 142 unique values
studio: 200 unique values
```

Loading and analysing of the Movie DB dataframe.

```
#load the dataset
movDB_df = pd.read_csv('zippedData/tmdb.movies.csv.gz')
movDB_df.head()
```

```
   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
```

```
print("=== Movie DB Dataset Analysis ===")
print("\nDataFrame Info:")
movDB_df.info()

# Check for duplicate records
duplicates = movDB_df.duplicated().sum()
print(f"\nDuplicate records: {duplicates}
({(duplicates/len(movDB_df))*100:.2f}% of total)")

# Display null value counts and percentages
null_counts = movDB_df.isnull().sum()
null_percentages = (null_counts/len(movDB_df))*100
print("\nMissing Values Analysis:")
for col in movDB_df.columns:
    if null_counts[col] > 0:
        print(f"{col}: {null_counts[col]} missing values
({null_percentages[col]:.2f}%)")

# Display basic statistics for numeric columns
print("\nNumeric Column Statistics:")
print(movDB_df.describe().round(2))
```

```
=== Movie DB Dataset Analysis ===

DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         26517 non-null  int64
 1   genre_ids          26517 non-null  object
 2   id                 26517 non-null  int64
 3   original_language  26517 non-null  object
 4   original_title     26517 non-null  object
 5   popularity         26517 non-null  float64
 6   release_date       26517 non-null  object
 7   title              26517 non-null  object
 8   vote_average       26517 non-null  float64
 9   vote_count         26517 non-null  int64
dtypes: float64(2), int64(3), object(5)
memory usage: 2.0+ MB
```

```
Duplicate records: 0 (0.00% of total)

Missing Values Analysis:

Numeric Column Statistics:
       Unnamed: 0         id  popularity  vote_average  vote_count
count    26517.00   26517.00    26517.00      26517.00    26517.00
mean     13258.00  295050.15        3.13          5.99      194.22
std       7654.94  153661.62        4.36          1.85      960.96
min          0.00      27.00        0.60          0.00        1.00
25%       6629.00  157851.00        0.60          5.00        2.00
50%      13258.00  309581.00        1.37          6.00        5.00
75%      19887.00  419542.00        3.69          7.00       28.00
max      26516.00  608444.00       80.77         10.00    22186.00
```

Loading and analysing the Movie Budgets Dataset

```
#load the dataset
df_budgets = pd.read_csv('zippedData/tn.movie_budgets.csv.gz')
df_budgets.head()
```

```
   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
```

```
print("=== DataFrame Information ===")
df_budgets.info(memory_usage='deep')

print("\n=== Numeric Column Statistics ===")
df_budgets.describe()
```

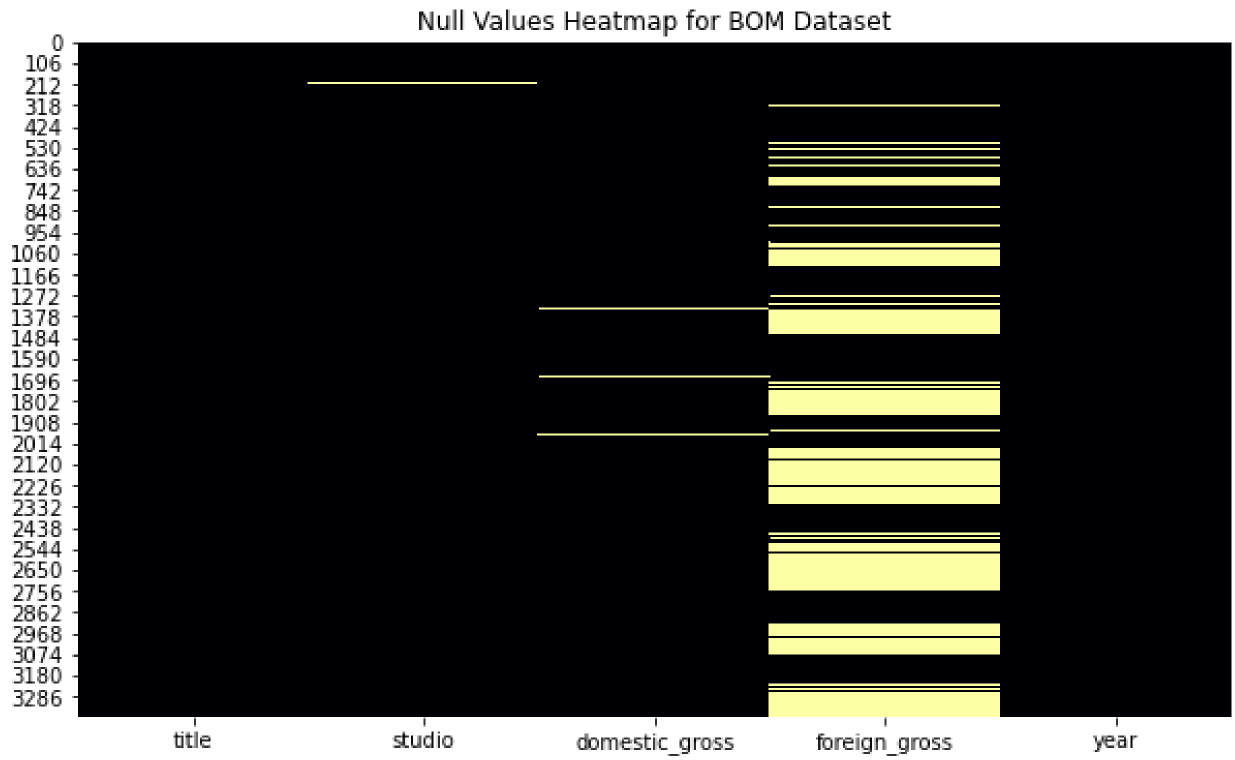
```
=== DataFrame Information ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5782 non-null   int64
 1   release_date       5782 non-null   object
 2   movie              5782 non-null   object
```

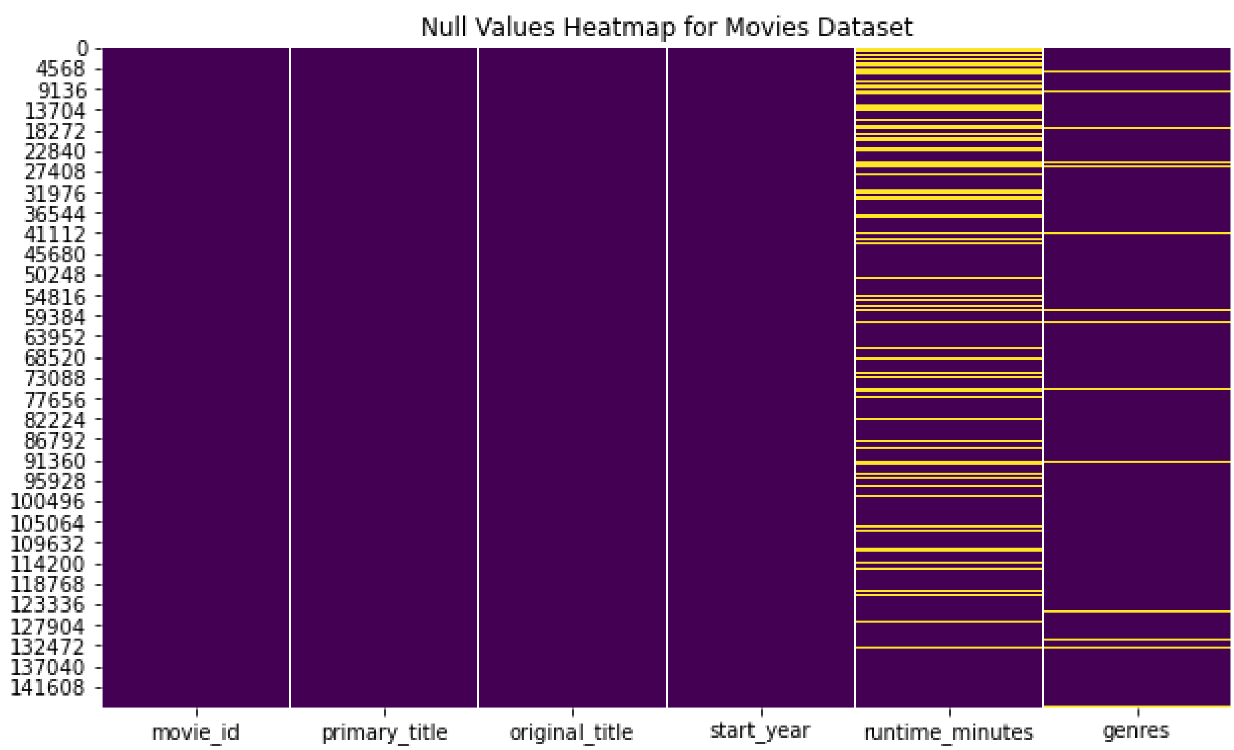
```
 3   production_budget  5782 non-null   object
 4   domestic_gross     5782 non-null   object
 5   worldwide_gross    5782 non-null   object
dtypes: int64(1), object(5)
memory usage: 1.9 MB

=== Numeric Column Statistics ===

                id
count  5782.000000
mean     50.372363
std      28.821076
min       1.000000
25%      25.000000
50%      50.000000
75%      75.000000
max     100.000000
```

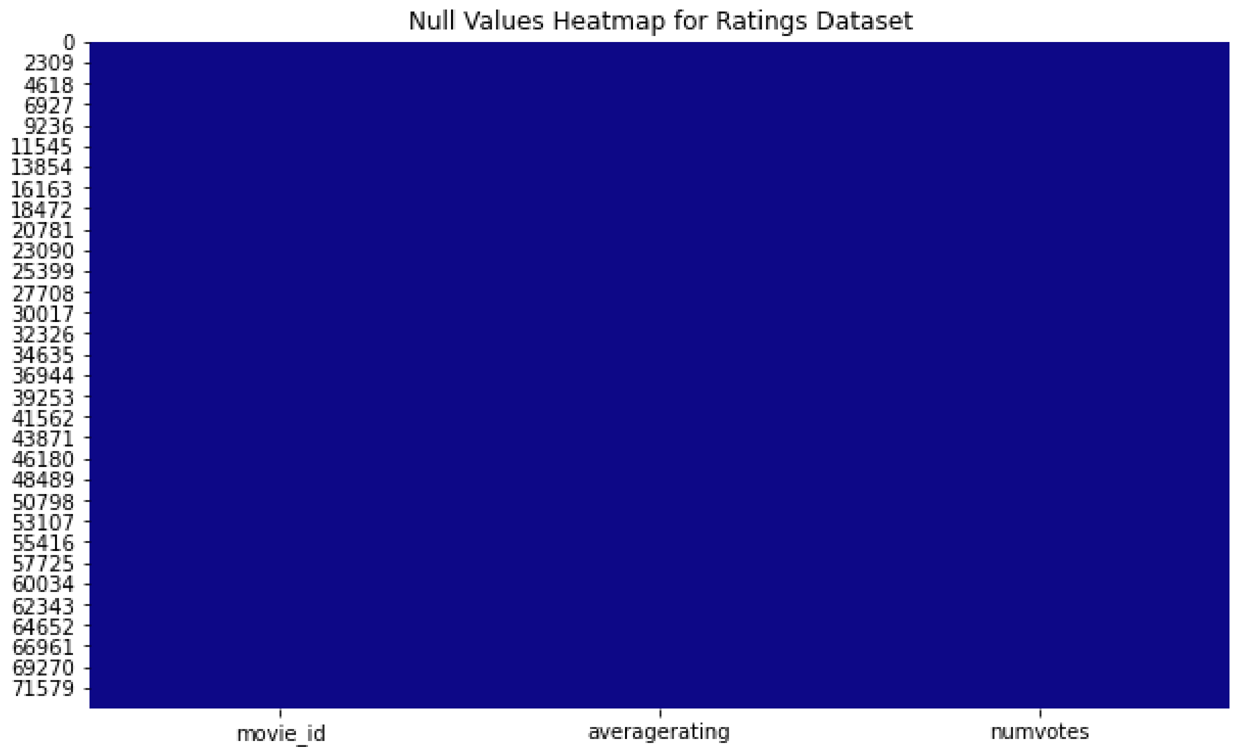
Inital Data Visualization

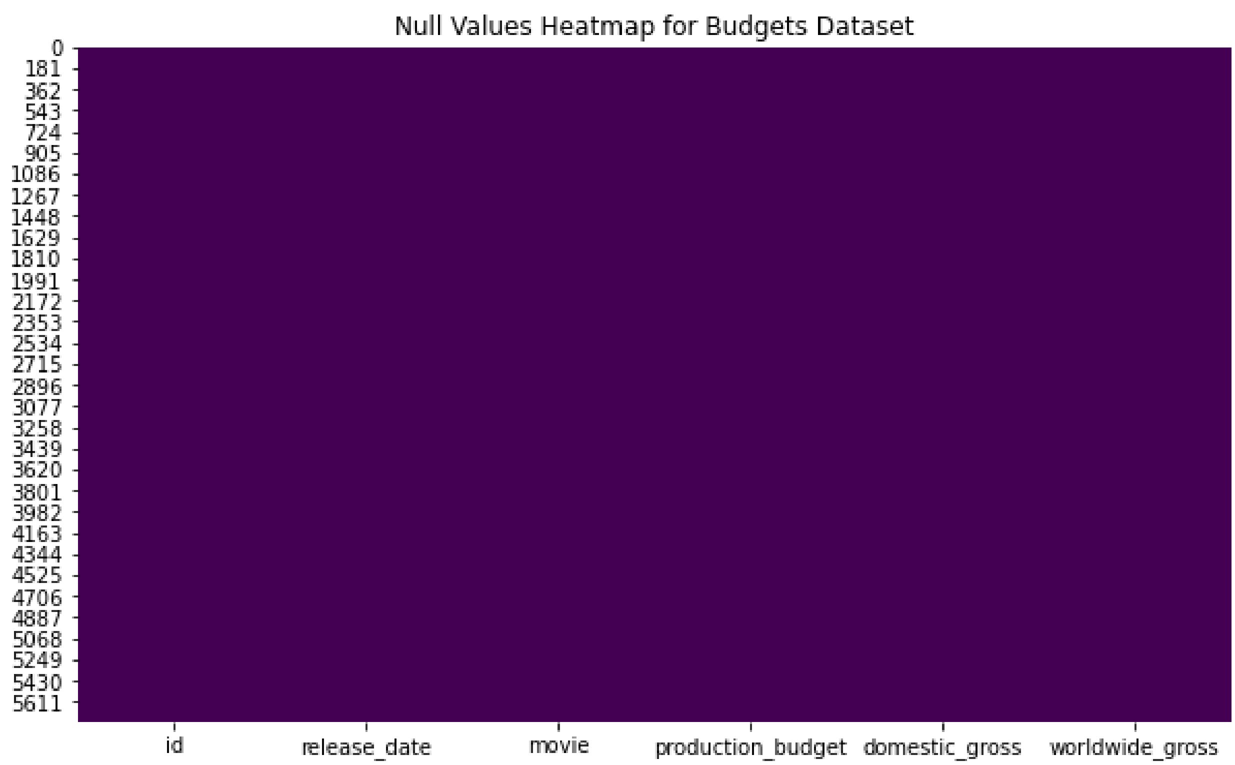
Barplots, histplots and Heatmaps for values before cleaning in BOM.Movie dataset and Movie Basics and Ratings tables.

```
#heatmap for null values
plt.figure(figsize=(10, 6))
sns.heatmap(BOMdf.isnull(), cbar=False, cmap='inferno')
plt.title('Null Values Heatmap for BOM Dataset')
plt.show()

#heatmap for null values in movie_basics
plt.figure(figsize=(10, 6))
sns.heatmap(movie_basics.isnull(), cbar=False, cmap='viridis')
plt.title('Null Values Heatmap for Movies Dataset')
plt.show()

#heatmap for null values in movie_ratings
plt.figure(figsize=(10, 6))
sns.heatmap(movie_ratings.isnull(), cbar=False, cmap='plasma')
plt.title('Null Values Heatmap for Ratings Dataset')
plt.show()

#heatmap for null values in budgets
plt.figure(figsize=(10, 6))
sns.heatmap(df_budgets.isnull(), cbar=False, cmap='viridis')
plt.title('Null Values Heatmap for Budgets Dataset')
plt.show()
```


Null Values Heatmap for BOM Dataset
0
106
212
318
424
530
636
742
848
954
1060
1166
1272
1378
1484
1590
1696
1802
1908
2014
2120
2226
2332
2438
2544
2650
2756
2862
2968
3074
3180
3286
title
studio
domestic_gross
foreign_gross
year


Null Values Heatmap for Movies Dataset
0
4568
9136
13704
18272
22840
27408
31976
36544
41112
45680
50248
54816
59384
63952
68520
73088
77656
82224
86792
91360
95928
100496
105064
109632
114200
118768
123336
127904
132472
137040
141608
movie_id
primary_title
original_title
start_year
runtime_minutes
genres





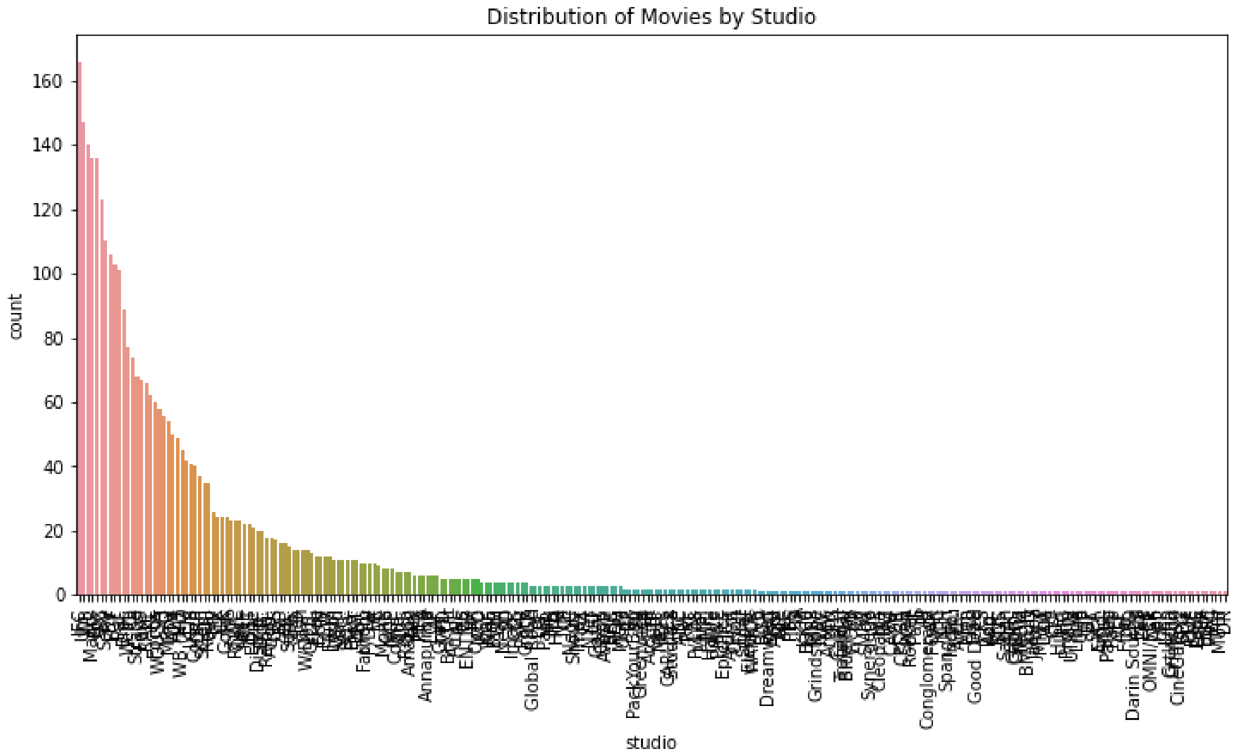
BOM dataset heatmap shows that foreign_gross is the least complete variable, while title and year are reliable. Missingness is scattered across rows but concentrated in financial columns. This highlights the need for careful cleaning and imputation strategies before regression modeling. studio: Has some missing values scattered across the dataset (thin yellow lines). This means a few movies don’t have studio information recorded. domestic_gross: Several rows are missing domestic box office values. Missingness is irregular, but present in noticeable chunks. foreign_gross: This column has the largest proportion of missing values — many rows lack foreign gross figures. This is common in movie datasets, as not all films are released or reported internationally.

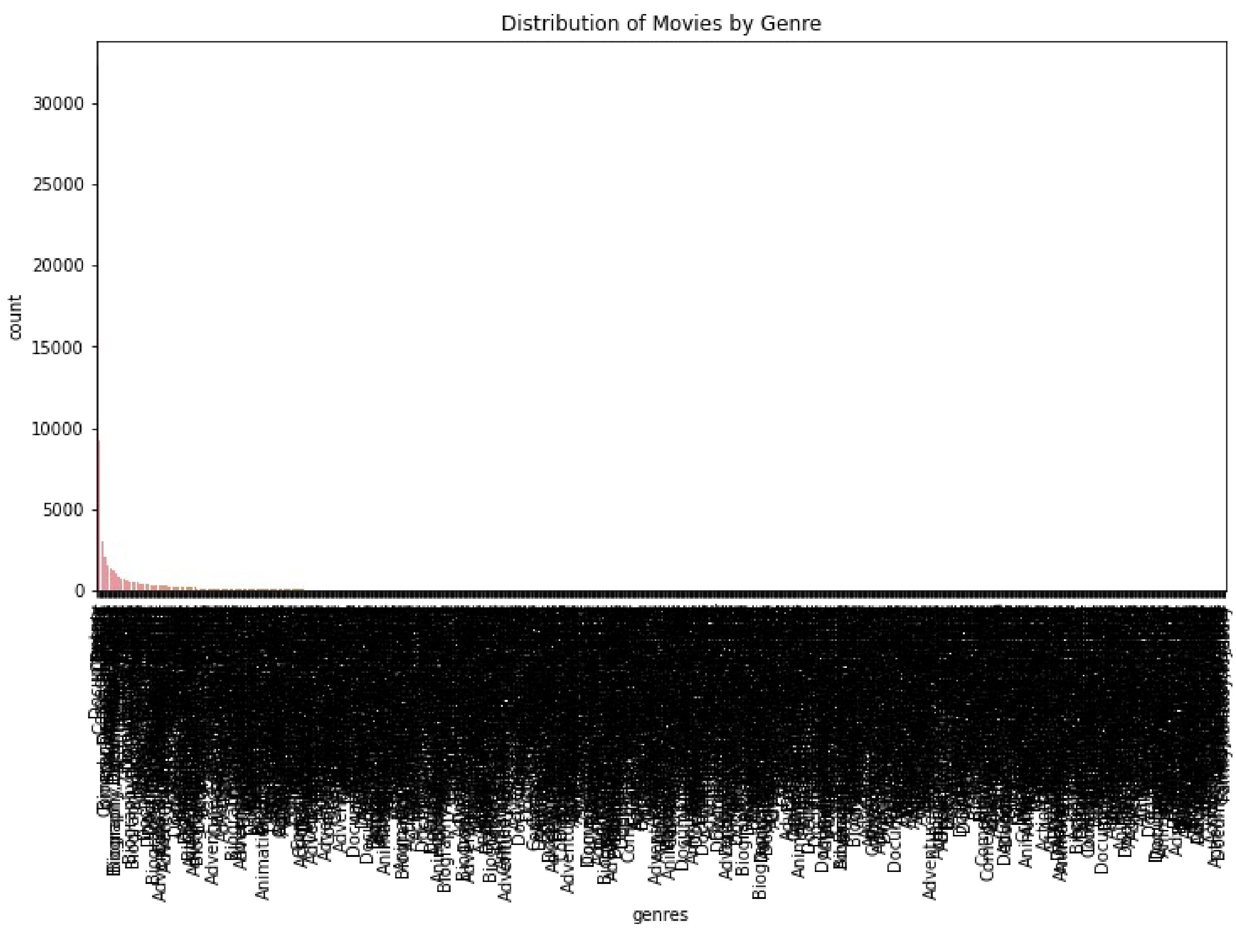
Movies dataset is generally complete, with runtime_minutes and genres being the only columns with missing values. Careful handling of these missing values is essential before regression modeling, especially since genres are tied to Objective 1 (genre analysis) and runtime ties to Objective 3 (audience-related factors).

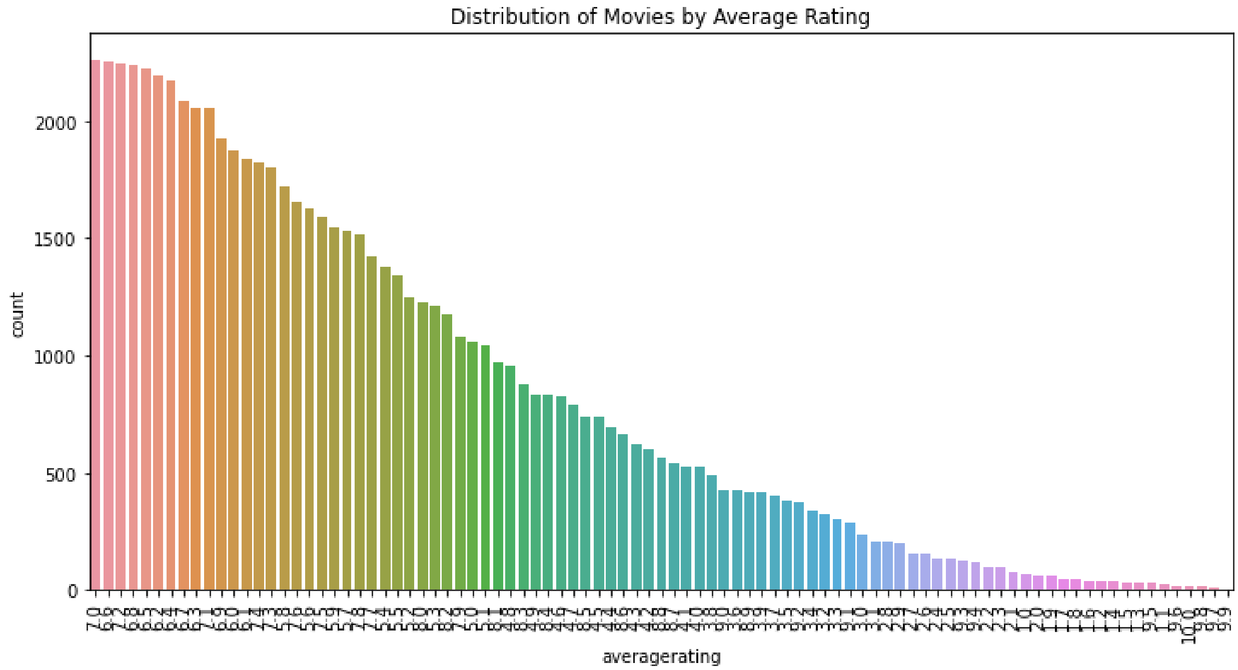
Ratings dataset is clean and complete. Every movie in this dataset has valid ratings and vote counts, making it a strong and trustworthy component for regression and correlation analysis.

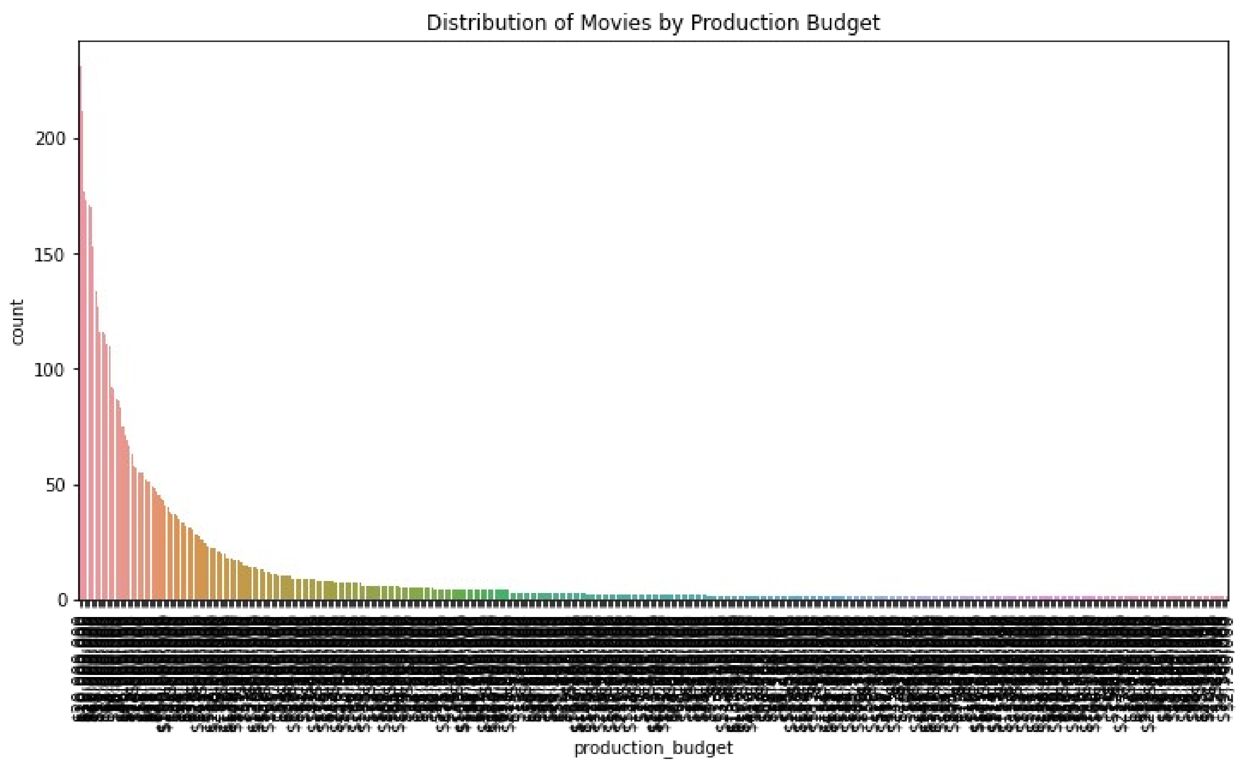
Budgets dataset is clean and complete, with zero missing values across all key variables. It is ready for direct use in exploratory analysis and regression modeling without requiring imputation or row-dropping.

```
#Bar plots for categorical variables
plt.figure(figsize=(12, 6))
sns.countplot(data=BOMdf, x='studio',
order=BOMdf['studio'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Distribution of Movies by Studio')
plt.show()

plt.figure(figsize=(12, 6))
sns.countplot(data=movie_basics, x='genres',
order=movie_basics['genres'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Distribution of Movies by Genre')
plt.show()

plt.figure(figsize=(12, 6))
sns.countplot(data=movie_ratings, x='averagerating',
order=movie_ratings['averagerating'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Distribution of Movies by Average Rating')
plt.show()

plt.figure(figsize=(12, 6))
sns.countplot(data=df_budgets, x='production_budget',
order=df_budgets['production_budget'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Distribution of Movies by Production Budget')
plt.show()
```


Distribution of Movies by Studio
count
160
140
120
100
80
60
40
20
0
studio


Distribution of Movies by Genre
count
30000
25000
20000
15000
10000
5000
0
genres





The BOM.Movie dataset has alot of missing values on the colums foreign gross and a few missing on the domestic gross column that need to be cleaned. The table movie_basics has a lot missing values on the column runtime_minutes and a few on the genre columns.

Conclusion

As our company ventures into the competitive and high-stakes movie industry, a strong understanding of what drives box office success is essential. While we are new to film production, a data-driven approach will allow us to minimize risk, align our content with market demand, and make informed decisions from the outset.

This project sets the foundation for that strategy by identifying the key factors that contribute to successful films. Through analyzing industry data, we aim to uncover actionable insights that will inform content choices, guide investment decisions, and position our studio for long-term growth.

By translating these findings into clear, practical recommendations, we will empower stakeholders across the organization—from creative development to finance and operations—to collaborate effectively and launch a studio built on insight, not guesswork.

In our quest to Identify box office success drivers, understand market trends, develop actionable production insights, support strategic investment decisions and lay a foundation for data-Driven content strategy, our analysis will focus on the IM.DB and BOM.Movies, Movie_Budget datasets. In the IM.DB database we will focus our analysis on the Movie basics and Reviews tables.

3. Data Cleaning and Analysis

Economic Analysis

Preprocessing and merging datasets

Convert currency strings to numeric values in the `df_budgets` dataframe.

```
# Columns to clean
currency_cols = ['production_budget', 'domestic_gross', 'worldwide_gross']

# Remove $ and commas, then convert to numeric
for col in currency_cols:
    df_budgets[col] = df_budgets[col].replace('[\$,]', '', regex=True).astype(float)

# Quick check
print(df_budgets[currency_cols].head())
print(df_budgets[currency_cols].dtypes)
```

```
   production_budget  domestic_gross  worldwide_gross
0        425000000.0     760507625.0     2.776345e+09
1        410600000.0     241063875.0     1.045664e+09
2        350000000.0      42762350.0     1.497624e+08
3        330600000.0     459005868.0     1.403014e+09
```

```
4        317000000.0     620181382.0     1.316722e+09
production_budget    float64
domestic_gross       float64
worldwide_gross      float64
dtype: object
```

Preprocess Box Office Mojo (BOM) data

```
# Convert foreign_gross to numeric, handling comma separators
BOMdf['foreign_gross'] =
pd.to_numeric(BOMdf['foreign_gross'].str.replace(',', ''),
errors='coerce')

# Calculate total gross where both values are available
BOMdf['total_gross'] = BOMdf['foreign_gross'] +
BOMdf['domestic_gross']

# Display basic statistics after conversion
print("\nSummary statistics after preprocessing:")
print(BOMdf[['foreign_gross', 'domestic_gross',
'total_gross']].describe())

# Display count of remaining null values
print("\nRemaining null values:")
print(BOMdf[['foreign_gross', 'domestic_gross',
'total_gross']].isnull().sum())


Summary statistics after preprocessing:
       foreign_gross  domestic_gross   total_gross
count   2.037000e+03    3.359000e+03  2.009000e+03
mean    7.487281e+07    2.874585e+07  1.226913e+08
std     1.374106e+08    6.698250e+07  2.074870e+08
min     6.000000e+02    1.000000e+02  4.900000e+03
25%     3.700000e+06    1.200000e+05  8.141000e+06
50%     1.870000e+07    1.400000e+06  4.230000e+07
75%     7.490000e+07    2.790000e+07  1.337000e+08
max     9.605000e+08    9.367000e+08  1.518900e+09

Remaining null values:
foreign_gross     1350
domestic_gross      28
total_gross       1378
dtype: int64
```

Merge movie budgets with Box Office Mojo data

```
# Select relevant columns from BOMdf and merge with budgets data
BOM_budgets_merged = pd.merge(
    df_budgets,
```

```
    BOMdf[['title', 'studio', 'foreign_gross', 'year']],
    left_on='movie',
    right_on='title',
    how='left'
)

BOM_budgets_merged
```

```
      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
...   ..           ...                                           ...
5777  78  Dec 31, 2018                                        Red 11
5778  79   Apr 2, 1999                                     Following
5779  80  Jul 13, 2005                Return to the Land of Wonders
5780  81  Sep 29, 2015                         A Plague So Pleasant
5781  82   Aug 5, 2005                            My Date With Drew

      production_budget  domestic_gross  worldwide_gross  \
0           425000000.0     760507625.0     2.776345e+09
1           410600000.0     241063875.0     1.045664e+09
2           350000000.0      42762350.0     1.497624e+08
3           330600000.0     459005868.0     1.403014e+09
4           317000000.0     620181382.0     1.316722e+09
...                 ...             ...              ...
5777             7000.0             0.0     0.000000e+00
5778             6000.0         48482.0     2.404950e+05
5779             5000.0          1338.0     1.338000e+03
5780             1400.0             0.0     0.000000e+00
5781             1100.0        181041.0     1.810410e+05

                                            title studio
foreign_gross  \
0                                             NaN    NaN
NaN
1     Pirates of the Caribbean: On Stranger Tides     BV
804600000.0
2                                             NaN    NaN
NaN
3                         Avengers: Age of Ultron     BV
946400000.0
4                                             NaN    NaN
NaN
...                                           ...    ...              ..
.
5777                                          NaN    NaN
NaN
```

```
5778                                                  NaN     NaN
NaN
5779                                                  NaN     NaN
NaN
5780                                                  NaN     NaN
NaN
5781                                                  NaN     NaN
NaN

        year
0        NaN
1     2011.0
2        NaN
3     2015.0
4        NaN
...      ...
5777     NaN
5778     NaN
5779     NaN
5780     NaN
5781     NaN

[5782 rows x 10 columns]
```

Verify the merge results and drop duplicate title columns if present

```
# Validate merge results
print("\nAfter merge:")
print(f"BOM_budgets_merged shape: {BOM_budgets_merged.shape}")
print(f"Null values in merged columns:")
print(BOM_budgets_merged[['title', 'studio', 'foreign_gross',
'year']].isnull().sum())

# Drop duplicate title column
BOM_budgets_merged = BOM_budgets_merged.drop('title', axis=1,
errors='ignore')


After merge:
BOM_budgets_merged shape: (5782, 10)
Null values in merged columns:
title            4535
studio           4536
foreign_gross    4696
year             4535
dtype: int64
```

Merge relevant tables from IMDB tables.

```
# First merge IMDb tables (movie_basics and ratings)
print("=== Merging IMDb Tables ===")
print(f"movie_basics shape: {movie_basics.shape}")
print(f"movie_ratings shape: {movie_ratings.shape}")

# Create merged IMDb dataset
imdb_merged = pd.merge(
    movie_basics,
    movie_ratings,
    on='movie_id',
    how='inner',
    validate='1:1'
)
print(f"\nIMDb merged shape: {imdb_merged.shape}")
```

```
=== Merging IMDb Tables ===
movie_basics shape: (146144, 6)
movie_ratings shape: (73856, 3)

IMDb merged shape: (73856, 8)
```

Create final combined Dataset `merged_df`.

```
# Merge IMDb data with existing budget/box office data
print("\n=== Merging with Budget/Box Office Data ===")
print(f"Current BOM_budgets_merged shape: {BOM_budgets_merged.shape}")

merged_df = pd.merge(
    BOM_budgets_merged,
    imdb_merged,
    left_on='movie',
    right_on='primary_title',
    how='left',
)
```

```
=== Merging with Budget/Box Office Data ===
Current BOM_budgets_merged shape: (5782, 9)
```

Verify final results and drop duplicate columns from the `merged_df`.

```
# Validate final merge results
print(f"\nFinal merged shape: {merged_df.shape}")
print("\nNull values in key IMDb columns:")
print(merged_df[['movie_id', 'averagerating',
'genres']].isnull().sum())

# Clean up duplicate columns
merged_df = merged_df.drop('primary_title', axis=1, errors='ignore')
```

```
# Display sample of merged data
print("\nSample of merged data:")
print(merged_df[['movie', 'movie_id', 'averagerating',
'genres']].head())
```

```
Final merged shape: (6473, 17)

Null values in key IMDb columns:
movie_id         3598
averagerating    3598
genres           3606
dtype: int64

Sample of merged data:
                                        movie   movie_id
averagerating  \
0                                      Avatar  tt1775309
6.1
1  Pirates of the Caribbean: On Stranger Tides  tt1298650
6.6
2                                Dark Phoenix  tt6565702
6.0
3                     Avengers: Age of Ultron  tt2395427
7.3
4            Star Wars Ep. VIII: The Last Jedi        NaN
NaN

                     genres
0                    Horror
1  Action,Adventure,Fantasy
2   Action,Adventure,Sci-Fi
3   Action,Adventure,Sci-Fi
4                       NaN
```

Calculate financial performance metrics

```
import numpy as np
import pandas as pd

# 1) Convert currency-like strings to numeric
num_cols = ["worldwide_gross", "production_budget", "domestic_gross",
"foreign_gross"]

def to_number(s):
    # keep digits, decimal point, sign, scientific notation (e/E),
strip everything else
    return pd.to_numeric(
        s.astype(str).str.replace(r"[^\d\.\-eE]", "", regex=True),
```

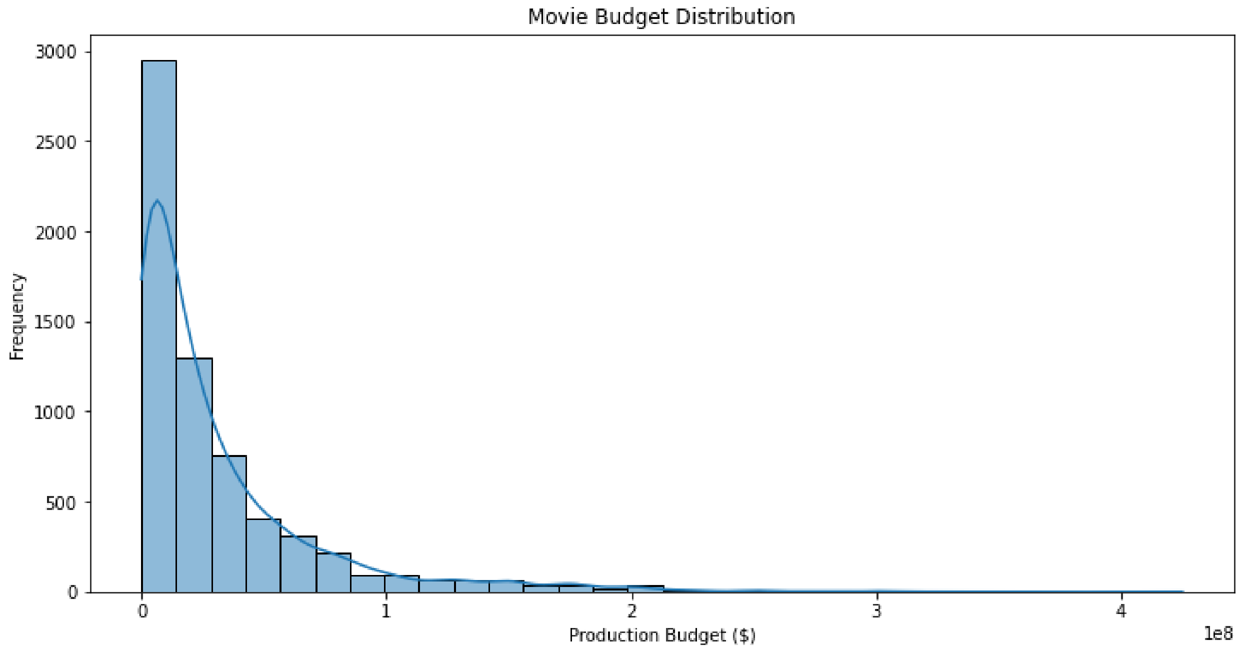
```
        errors="coerce"
    )

for c in num_cols:
    if c in merged_df.columns:
        merged_df[c] = to_number(merged_df[c])

# 2) Compute ROI safely: (WW - Budget) / Budget   (only when budget >
0)
mask = merged_df["production_budget"] > 0
merged_df["ROI"] = np.nan
merged_df.loc[mask, "ROI"] = (
    (merged_df.loc[mask, "worldwide_gross"] - merged_df.loc[mask,
"production_budget"])
    / merged_df.loc[mask, "production_budget"]
)

# 3) (Optional) Profit margin: (WW - Budget) / WW   (only when WW > 0)
mask_gross = merged_df["worldwide_gross"] > 0
merged_df["profit_margin"] = np.nan
merged_df.loc[mask_gross, "profit_margin"] = (
    (merged_df.loc[mask_gross, "worldwide_gross"] -
merged_df.loc[mask_gross, "production_budget"])
    / merged_df.loc[mask_gross, "worldwide_gross"]
)

# 4) Quick sanity checks
print(merged_df[num_cols + ["ROI", "profit_margin"]].dtypes)
print(merged_df[[ "worldwide_gross", "production_budget", "ROI",
"profit_margin"]].head())
```

```
worldwide_gross      float64
production_budget    float64
domestic_gross       float64
foreign_gross        float64
ROI                  float64
profit_margin        float64
dtype: object
   worldwide_gross  production_budget       ROI  profit_margin
0     2.776345e+09        425000000.0  5.532577       0.846921
1     1.045664e+09        410600000.0  1.546673       0.607331
2     1.497624e+08        350000000.0 -0.572108      -1.337036
3     1.403014e+09        330600000.0  3.243841       0.764364
4     1.316722e+09        317000000.0  3.153696       0.759251
```

Budget Analysis

```
# Budget distribution visualization
plt.figure(figsize=(12, 6))
sns.histplot(merged_df['production_budget'], bins=30, kde=True)
```

```python
plt.title('Movie Budget Distribution')
plt.xlabel('Production Budget ($)')
plt.ylabel('Frequency')
plt.show()
```



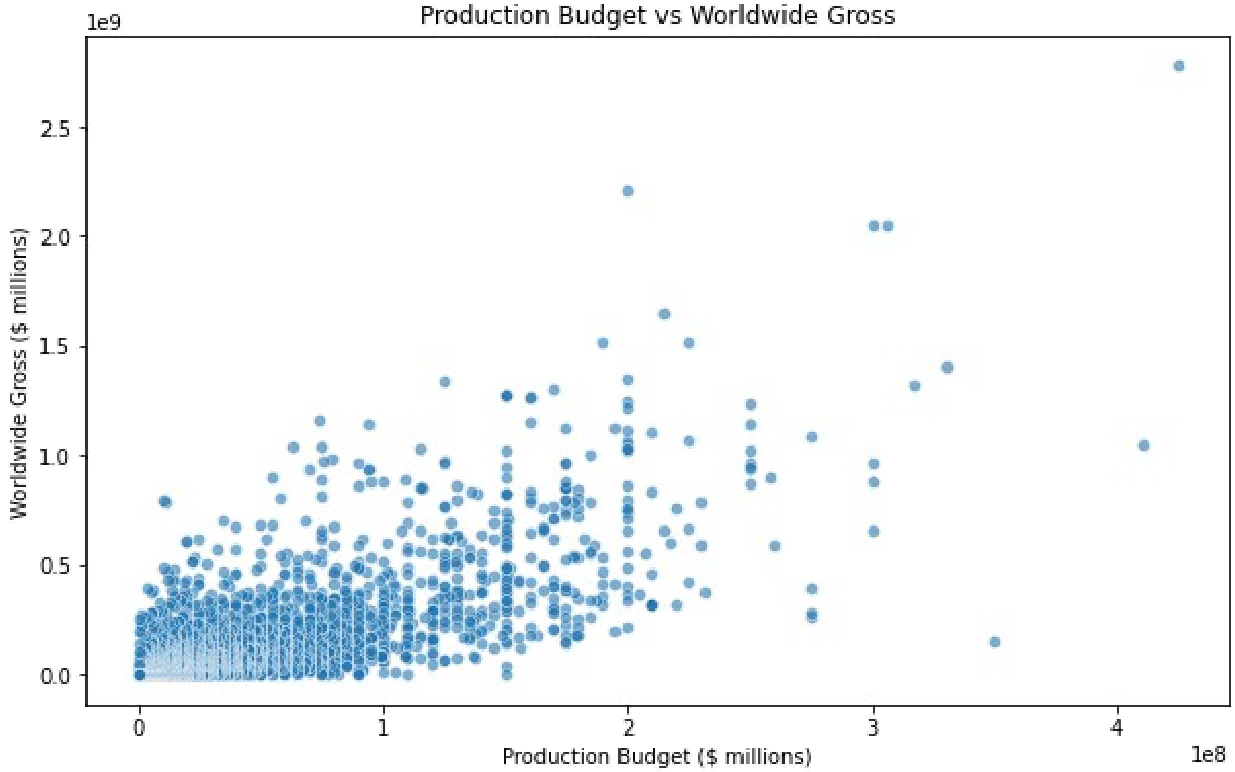
The histogram shows that most movies are produced on relatively small budgets, while a few high-cost blockbusters significantly skew the distribution. A log transformation is recommended before including production budgets in regression models.

Implications for Analysis Modeling: Because the data is skewed, applying a log transformation (e.g., log(production_budget)) will normalize the distribution, making it more suitable for regression analysis. Business Insight: Since most films are made on smaller budgets, but a few blockbusters require massive investment, studios need to balance low-risk, low-budget films with high-risk, high-reward blockbusters. ROI Analysis: Large budgets don't necessarily guarantee profits; therefore, ROI is a better measure than raw budget when evaluating financial performance.

Revenue Analysis

```python
# Budget vs Worldwide Gross
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x='production_budget',
    y='worldwide_gross',
    data=merged_df,
    alpha=0.6
)
plt.title('Production Budget vs Worldwide Gross')
plt.xlabel('Production Budget ($ millions)')
```

```
plt.ylabel('Worldwide Gross ($ millions)')
plt.show()
```



General Trend

The points show a positive relationship: as production budget increases, worldwide gross tends to increase. This means higher investments generally lead to higher revenues, though the relationship is not perfectly linear.

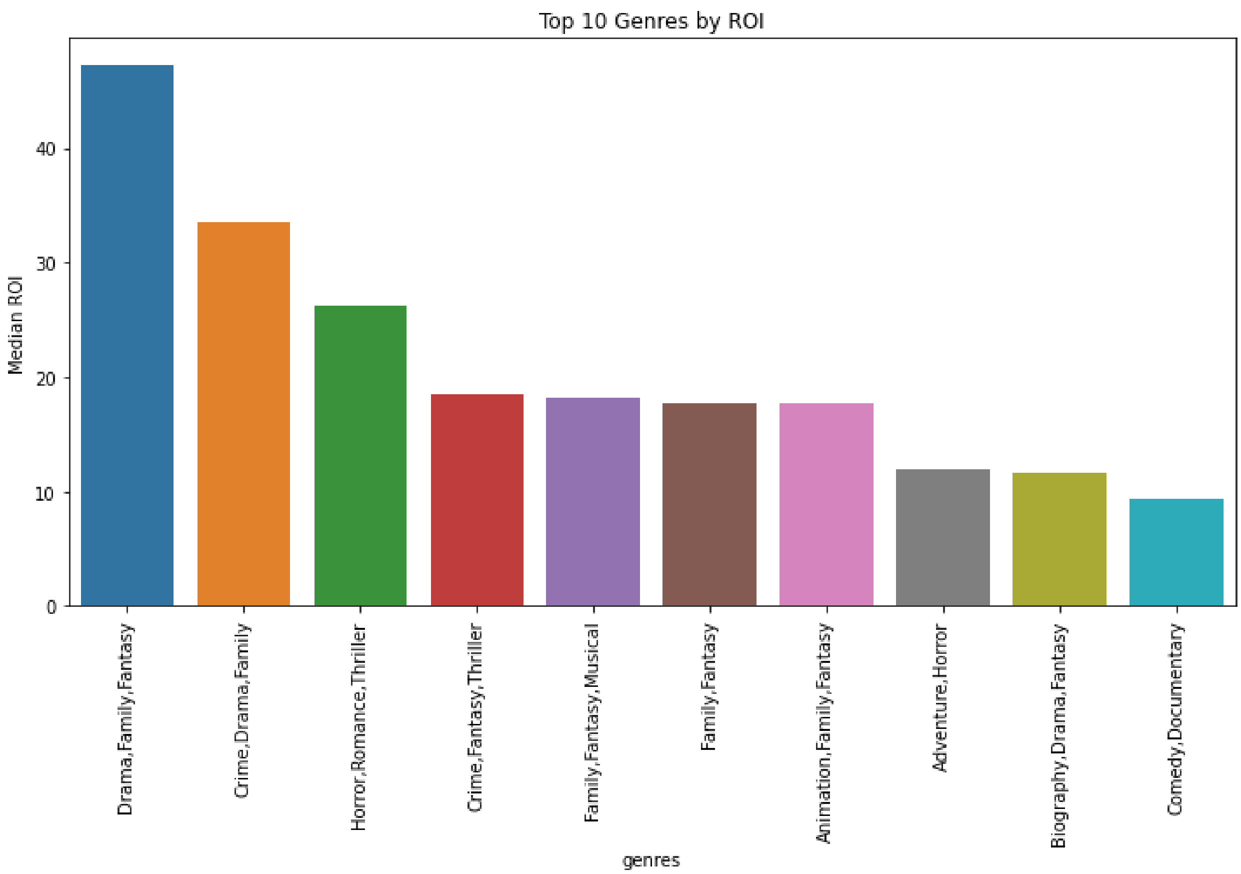
Clustering A large cluster of movies is concentrated at low to mid-level budgets (< $50 million) with grosses ranging from very low to moderate. This suggests that most movies are made on relatively smaller budgets, with varying levels of financial success.

Implications For analysis: The scatter plot indicates heteroscedasticity (variance increases with budget). A log–log transformation (e.g., log(worldwide_gross) ~ log(production_budget)) would help stabilize variance and improve regression modeling. For business decisions: Investing more in production generally pays off, but with diminishing returns. Beyond a certain budget, additional spending does not guarantee proportional increases in gross revenue.

Genre Performance Analysis

```
# Genre analysis
genre_df = merged_df.explode('genres')
genre_performance = genre_df.groupby('genres').agg({
    'worldwide_gross': 'median',
    'ROI': 'median',
```

```
    'averagerating': 'mean'
}).sort_values('ROI', ascending=False)

# Top 10 genres by ROI
plt.figure(figsize=(12, 6))
sns.barplot(
    x=genre_performance.head(10).index,
    y=genre_performance.head(10)['ROI']
)
plt.title('Top 10 Genres by ROI')
plt.xticks(rotation=90)
plt.ylabel('Median ROI')
plt.show()
```



Release Timing Analysis

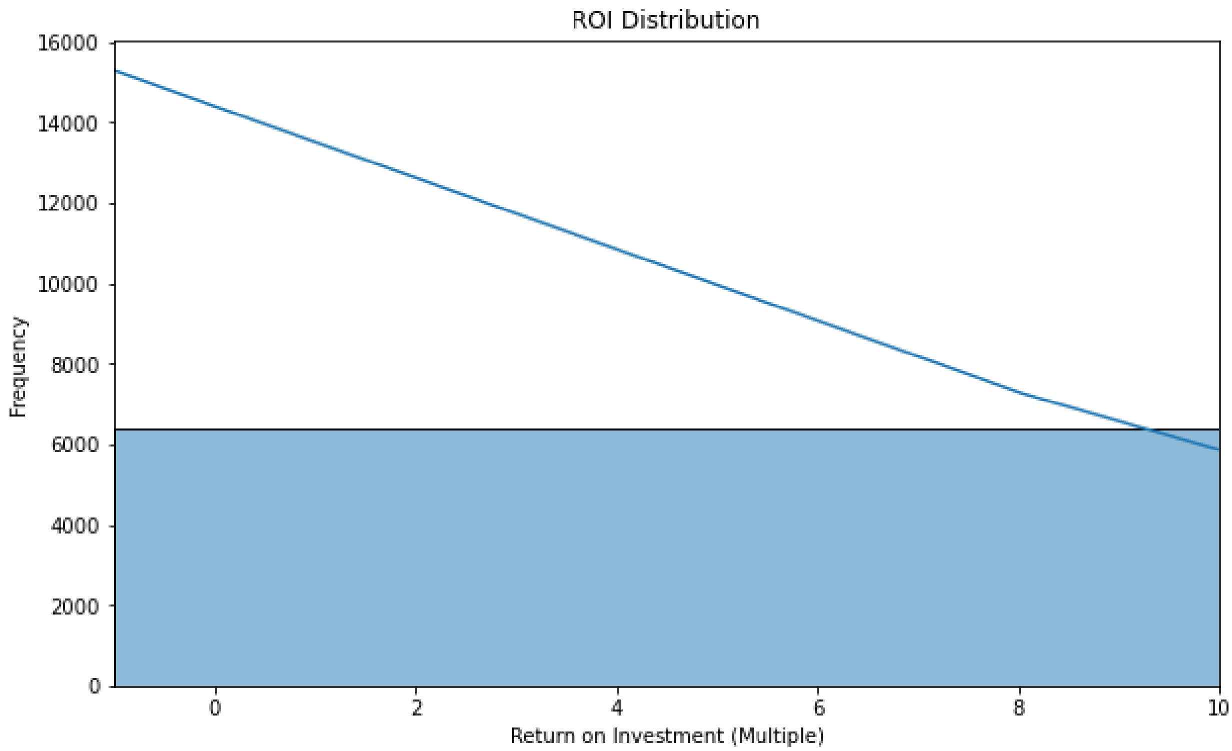
```
# Make sure release_date is datetime
merged_df['release_date'] = pd.to_datetime(merged_df['release_date'],
errors='coerce')

# Extract month number (1–12)
merged_df['release_month'] = merged_df['release_date'].dt.month
```

```python
# Optionally add month name for readability
merged_df['release_month_name'] =
merged_df['release_date'].dt.month_name()

# Now you can group by month
monthly_performance = merged_df.groupby('release_month').agg({
    'worldwide_gross': 'median',
    'ROI': 'median'
}).reset_index()

print(monthly_performance)
```

```
    release_month  worldwide_gross       ROI
0               1       22365133.0  0.581509
1               2       30063805.0  0.611561
2               3       25802739.5  0.531329
3               4       21464818.5  0.409252
4               5       25387091.0  0.847581
5               6       41410568.0  1.036885
6               7       49541995.5  1.128293
7               8       22108977.0  0.627456
8               9       18117579.0  0.303174
9              10       15392609.0  0.413520
10             11       51695362.0  1.109497
11             12       31194353.5  0.642332
```

ROI Analysis

```python
# ROI distribution
plt.figure(figsize=(10, 6))
sns.histplot(merged_df['ROI'], bins=50, kde=True)
plt.title('ROI Distribution')
plt.xlabel('Return on Investment (Multiple)')
plt.ylabel('Frequency')
plt.xlim(-1, 10)  # Exclude extreme outliers
plt.show()
```



The distribution is heavily skewed toward the lower ROI values.

Most movies cluster near ROI = 0 to 2 (meaning they either made losses or only modest profits).

Very few movies reach ROI above 5, and extremely few reach ROI close to 10.

Operational Analysis

Cleaning and analysing data.

```
#Cleaning rt.reviews.tsv

df_reviews.drop_duplicates(inplace=True)
df_reviews.columns =
df_reviews.columns.str.strip().str.lower().str.replace(' ', '_')
df_reviews.dropna(inplace=True)

print("\nCleaned data:")
print(df_reviews.head())
print("\nMissing values after cleaning:")
print(df_reviews.isnull().sum())


Cleaned data:
    id                                             review rating
fresh  \
0    3  A distinctly gallows take on contemporary fina...    3/5
fresh
```

```
6    3  Quickly grows repetitive and tiresome, meander...      C
rotten
7    3  Cronenberg is not a director to be daunted by ...    2/5
rotten
11   3  While not one of Cronenberg's stronger films, ...     B-
fresh
12   3  Robert Pattinson works mighty hard to make Cos...    2/4
rotten

            critic  top_critic           publisher               date

0       PJ Nabarro           0     Patrick Nabarro  November 10, 2018

6   Eric D. Snider           0     EricDSnider.com      July 17, 2013

7     Matt Kelemen           0  Las Vegas CityLife     April 21, 2013

11    Emanuel Levy           0     EmanuelLevy.Com   February 3, 2013

12  Christian Toto           0       Big Hollywood   January 15, 2013


Missing values after cleaning:
id            0
review        0
rating        0
fresh         0
critic        0
top_critic    0
publisher     0
date          0
dtype: int64

# Harmonize the rating column
def harmonize_rating(rating):
    mapping = {'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5}

    if isinstance(rating, str):
        rating = rating.strip().upper()
        if '/' in rating:
            # Take numerator of fraction
            numerator_part = rating.split('/')[0].strip()
            # Extract first digit
            for ch in numerator_part:
                if ch.isdigit():
                    return int(ch)
            return None  # no digit found
        elif rating in mapping:
            return mapping[rating]
```

```python
    # If numeric (including floats like 3.0 or 3.)
    try:
        numeric_rating = round(float(str(rating).strip()))
        return int(numeric_rating)  # force to integer
    except ValueError:
        return None

# Apply harmonization
df_reviews['rating'] = df_reviews['rating'].apply(harmonize_rating)

# Drop rows with invalid rating
df_reviews.dropna(subset=['rating'], inplace=True)

# Final check
print("\nHarmonized data:")
print(df_reviews.head())
print("\nUnique ratings after harmonization:",
df_reviews['rating'].unique())
```

```
Harmonized data:
    id                                             review  rating
fresh  \
0    3  A distinctly gallows take on contemporary fina...     3.0
fresh
6    3  Quickly grows repetitive and tiresome, meander...     3.0
rotten
7    3  Cronenberg is not a director to be daunted by ...     2.0
rotten
12   3  Robert Pattinson works mighty hard to make Cos...     2.0
rotten
13   3  The anger over the injustice of the financial ...     2.0
fresh

            critic  top_critic           publisher               date

0       PJ Nabarro           0      Patrick Nabarro  November 10, 2018

6   Eric D. Snider           0      EricDSnider.com      July 17, 2013

7     Matt Kelemen           0   Las Vegas CityLife     April 21, 2013

12  Christian Toto           0        Big Hollywood   January 15, 2013

13    Robert Roten           0  Laramie Movie Scope    January 7, 2013


Unique ratings after harmonization: [ 3.  2.  4.  6.  1.  8.  7.  5.
9.  0. 10.]
```

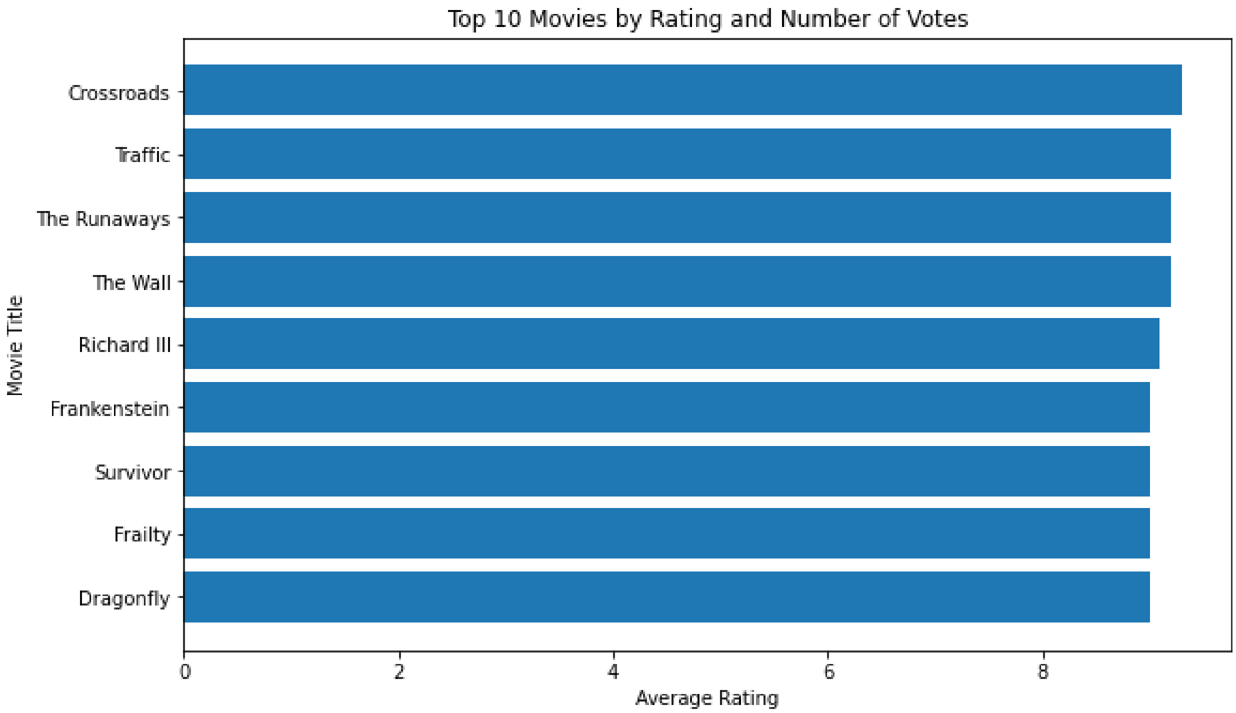
```python
#Clean the  rt.movie_info.tsv dataset
Movie_info_df.drop_duplicates(inplace=True)  # remove duplicate rows
Movie_info_df.columns =
Movie_info_df.columns.str.strip().str.lower().str.replace(' ', '_')  #
clean column names

# Rename 'rating' to 'rate' if it exists
if 'rating' in Movie_info_df.columns:
    Movie_info_df.rename(columns={'rating': 'rate'}, inplace=True)

Movie_info_df.dropna(how='all', inplace=True)  # drop rows where all
values are NaN
Movie_info_df.fillna('', inplace=True)  # fill remaining missing
values with empty strings (or choose suitable fill)

# Final check
print("\nCleaned movie data:")
print(Movie_info_df.head())
print("\nDataset shape:", Movie_info_df.shape)
print("\nColumn names:", Movie_info_df.columns.tolist())
```

```
Cleaned movie data:
   id                                           synopsis rate  \
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                                                      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

1     David Cronenberg|Don DeLillo  Aug 17, 2012   Jan 1, 2013
$
2                   Allison Anders  Sep 13, 1996  Apr 18, 2000

3  Paul Attanasio|Michael Crichton   Dec 9, 1994  Aug 27, 1997

4                     Giles Cooper


  box_office      runtime             studio
```

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

Dataset shape: (1560, 12)

Column names: ['id', 'synopsis', 'rate', 'genre', 'director',
'writer', 'theater_date', 'dvd_date', 'currency', 'box_office',
'runtime', 'studio']
```

TOP 10 Movies by ID based by rating

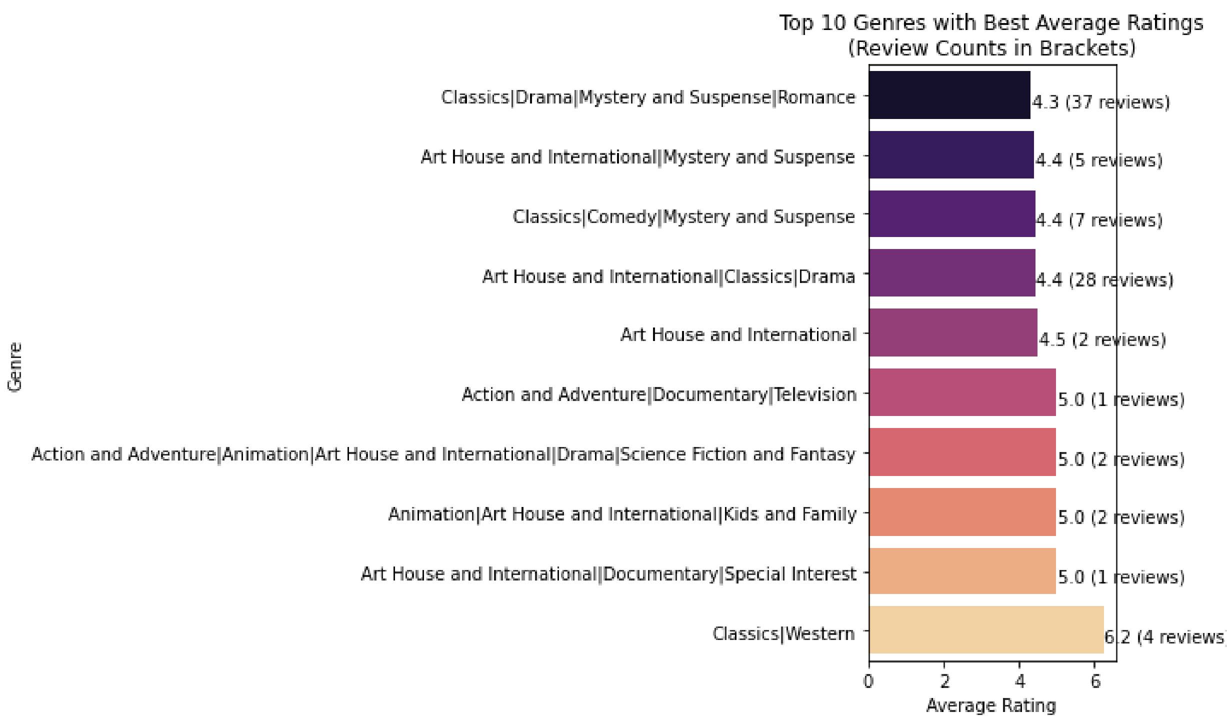
```
import matplotlib.pyplot as plt

# Make sure the dataframe has these columns: 'averagerating' and
'numvotes'
# (adjust names if they are different in your data)

# Step 1: Sort by rating and number of votes
sorted_movies = merged_df.sort_values(by=['averagerating',
'numvotes'], ascending=[False, False])

# Step 2: Select the top 10
top_10_movies = sorted_movies.head(10)

# Step 3: Plot
plt.figure(figsize=(10, 6))
plt.barh(top_10_movies['movie'], top_10_movies['averagerating'])
plt.xlabel("Average Rating")
plt.ylabel("Movie Title")
plt.title("Top 10 Movies by Rating and Number of Votes")
plt.gca().invert_yaxis()
plt.show()
```



Top 10 Genres with Best Average Rating

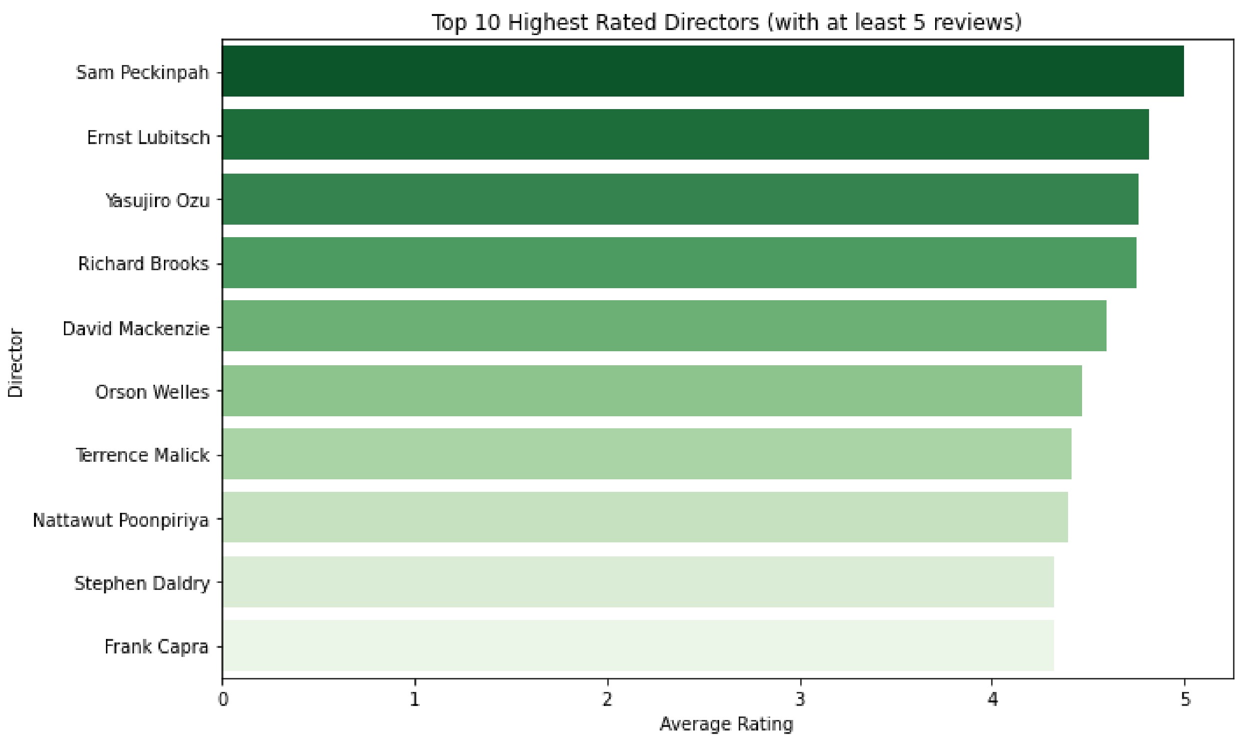
```
import matplotlib.pyplot as plt
import seaborn as sns

# Merge reviews with movies info
# df_reviews: id, rating
# Movie_info_df: id, title, genre

ratings_with_genre = df_reviews.merge(Movie_info_df[['id', 'genre']],
on='id')

# Calculate average rating and review count per genre
genre_stats = (
    ratings_with_genre.groupby('genre')
    .agg(
        avg_rating=('rating', 'mean'),
        num_reviews=('rating', 'count')
    )
    .reset_index()
    .sort_values(by='avg_rating', ascending=False)
    .head(10)
)

# Sort for better visualization
genre_stats = genre_stats.sort_values(by='avg_rating', ascending=True)
# Plot
plt.figure(figsize=(10, 6))
```

```
ax = sns.barplot(
    data=genre_stats,
    x='avg_rating',
    y='genre',
    palette='magma'
)

# Annotate bars with avg_rating and number of reviews
for i in ax.patches:
    avg_rating = i.get_width()
    genre_index = int(i.get_y() + 0.5)
    num_reviews = genre_stats.iloc[genre_index]['num_reviews']
    ax.text(
        avg_rating + 0.02,
        i.get_y() + 0.5,
        f"{avg_rating:.1f} ({num_reviews} reviews)",
        va='center'
    )

plt.xlabel('Average Rating')
plt.ylabel('Genre')
plt.title('Top 10 Genres with Best Average Ratings\n(Review Counts in Brackets)')
plt.tight_layout()
plt.show()
```



Top 10 highly rated Directors (with at least 5 reviews)

```
#Merge reviews with movies to get director info and ratings
ratings_with_director = df_reviews.merge(Movie_info_df[['id',
'director']], on='id')

# Calculate average rating and number of reviews per director
director_ratings = (
    ratings_with_director.groupby('director')
    .agg(
        avg_rating=('rating', 'mean'),
        num_reviews=('rating', 'count')
    )
    .reset_index()
)

# Optional: filter directors with enough reviews (e.g., 5+)
director_ratings = director_ratings[director_ratings['num_reviews'] >=
5]

# Sort by avg_rating descending (highest first)
top_directors = director_ratings.sort_values('avg_rating',
ascending=False).head(10)

# Plot
plt.figure(figsize=(10, 6))
ax = sns.barplot(
    data=top_directors,
    x='avg_rating',
    y='director',
    palette='Greens_r'
)

plt.xlabel('Average Rating')
plt.ylabel('Director')
plt.title('Top 10 Highest Rated Directors (with at least 5 reviews)')
plt.tight_layout()
plt.show()
```



Cleaning merged dataset i.e cleaning merged df

```
merged_df
```

```
      id release_date                                          movie  \
0      1   2009-12-18                                         Avatar
1      2   2011-05-20  Pirates of the Caribbean: On Stranger Tides
2      3   2019-06-07                                   Dark Phoenix
3      4   2015-05-01                        Avengers: Age of Ultron
4      5   2017-12-15              Star Wars Ep. VIII: The Last Jedi
...   ..          ...                                            ...
6468  78   2018-12-31                                         Red 11
6469  79   1999-04-02                                      Following
6470  80   2005-07-13                 Return to the Land of Wonders
6471  81   2015-09-29                          A Plague So Pleasant
6472  82   2005-08-05                             My Date With Drew

      production_budget  domestic_gross  worldwide_gross studio  \
0           425000000.0     760507625.0     2.776345e+09    NaN
1           410600000.0     241063875.0     1.045664e+09     BV
2           350000000.0      42762350.0     1.497624e+08    NaN
3           330600000.0     459005868.0     1.403014e+09     BV
4           317000000.0     620181382.0     1.316722e+09    NaN
...                 ...             ...              ...    ...
6468             7000.0             0.0     0.000000e+00    NaN
6469             6000.0         48482.0     2.404950e+05    NaN
6470             5000.0          1338.0     1.338000e+03    NaN
```

```
6471             1400.0            0.0     0.000000e+00    NaN
6472             1100.0       181041.0     1.810410e+05    NaN

      foreign_gross    year   movie_id  \
0               NaN     NaN  tt1775309
1       804600000.0  2011.0  tt1298650
2               NaN     NaN  tt6565702
3       946400000.0  2015.0  tt2395427
4               NaN     NaN        NaN
...             ...     ...        ...
6468            NaN     NaN  tt7837402
6469            NaN     NaN        NaN
6470            NaN     NaN        NaN
6471            NaN     NaN  tt2107644
6472            NaN     NaN        NaN

                                   original_title  start_year  \
0                                           Abatâ      2011.0
1     Pirates of the Caribbean: On Stranger Tides      2011.0
2                                    Dark Phoenix      2019.0
3                         Avengers: Age of Ultron      2015.0
4                                             NaN         NaN
...                                           ...         ...
6468                                       Red 11      2019.0
6469                                          NaN         NaN
6470                                          NaN         NaN
6471                       A Plague So Pleasant      2013.0
6472                                          NaN         NaN

      runtime_minutes                    genres  averagerating
numvotes  \
0                93.0                    Horror            6.1
43.0
1               136.0  Action,Adventure,Fantasy            6.6
447624.0
2               113.0   Action,Adventure,Sci-Fi            6.0
24451.0
3               141.0   Action,Adventure,Sci-Fi            7.3
665594.0
4                 NaN                       NaN            NaN
NaN
...               ...                       ...            ...       .
..
6468             77.0    Horror,Sci-Fi,Thriller            5.6
43.0
6469              NaN                       NaN            NaN
NaN
6470              NaN                       NaN            NaN
NaN
6471             76.0     Drama,Horror,Thriller            5.4
```

```
72.0
6472                 NaN                        NaN            NaN
NaN

             ROI  profit_margin  release_month release_month_name
0       5.532577       0.846921             12           December
1       1.546673       0.607331              5                May
2      -0.572108      -1.337036              6               June
3       3.243841       0.764364              5                May
4       3.153696       0.759251             12           December
...          ...            ...            ...                ...
6468   -1.000000            NaN             12           December
6469   39.082500       0.975051              4              April
6470   -0.732400      -2.736921              7               July
6471   -1.000000            NaN              9          September
6472  163.582727       0.993924              8             August

[6473 rows x 20 columns]
```

Issues to clean

a. Numeric columns production_budget, domestic_gross, worldwide_gross, foreign_gross are floats already, but some values are NaN or 0.

b. Missing values (NaN) Columns like studio, foreign_gross, year, genre have many NaNs.

c. Dates release_date is a string, should be converted to datetime.

d. Duplicate / irrelevant columns Check for duplicates (e.g., movie_id may repeat). Drop unnecessary metadata if not needed for modeling.

e. String columns (genres, studios)

genre column has multiple values separated by commas → may need to split or encode. studio sometimes missing.

```python
# 1. Convert release_date to datetime
merged_df['release_date'] = pd.to_datetime(merged_df['release_date'], errors='coerce')

# 2. Handle missing numeric values
numeric_cols = ['production_budget', 'domestic_gross', 'worldwide_gross', 'foreign_gross']
for col in numeric_cols:
    merged_df[col] = pd.to_numeric(merged_df[col], errors='coerce')  # force numeric
    merged_df[col] = merged_df[col].fillna(0)  # or merged_df[col].fillna(merged_df[col].median())
```

```python
# 3. Handle missing categorical values
merged_df['studio'] = merged_df['studio'].fillna("Unknown")
merged_df['genre'] = merged_df['genres'].fillna("Unknown")

# 4. Drop duplicates if any
merged_df = merged_df.drop_duplicates(subset=['movie',
'release_date'])

# 5.  Extract year from release_date if needed
merged_df['release_year'] = merged_df['release_date'].dt.year

# 6. Check data types
print(merged_df.dtypes)
```

```
id                               int64
release_date            datetime64[ns]
movie                           object
production_budget              float64
domestic_gross                 float64
worldwide_gross                float64
studio                          object
foreign_gross                  float64
year                           float64
movie_id                        object
original_title                  object
start_year                     float64
runtime_minutes                float64
genres                          object
averagerating                  float64
numvotes                       float64
ROI                            float64
profit_margin                  float64
release_month                    int64
release_month_name              object
genre                           object
release_year                     int64
dtype: object
```

```python
# 7. Inspect missing values
print(merged_df.isnull().sum())
```

```
id                         0
release_date               0
movie                      0
production_budget          0
domestic_gross             0
worldwide_gross            0
studio                     0
foreign_gross              0
year                    4535
movie_id                3598
```

```
original_title          3598
start_year              3598
runtime_minutes         3664
genres                  3600
averagerating           3598
numvotes                3598
ROI                        0
profit_margin            367
release_month              0
release_month_name         0
genre                      0
release_year               0
dtype: int64
```

##some columns still have missing values Why These Columns Have Missing Values

year, start_year, release_year: Some movies might not have complete release date info in the raw datasets. Missing years are common for older or unreleased films.

movie_id, original_title: Missing if the dataset couldn't match the movie to IMDb/another source.

runtime_minutes: Missing if runtime wasn't listed in IMDb or TheMovieDB.

genres: Missing if the movie didn't have genre info in the source dataset.

averagerating, numvotes: Missing if IMDb didn't have enough votes/ratings for that film

How to Clean These Columns

```
# 1. Fill year-related columns using release_date if available
merged_df['year'] =
merged_df['release_date'].dt.year.fillna(merged_df['year'])

# 2. Handle runtime_minutes (replace NaN with median runtime)
merged_df['runtime_minutes'] =
merged_df['runtime_minutes'].fillna(merged_df['runtime_minutes'].media
n())

# 3. Handle genres (replace NaN with "Unknown")
merged_df['genres'] = merged_df['genres'].fillna("Unknown")

# 4. Handle ratings and votes
merged_df['averagerating'] =
merged_df['averagerating'].fillna(merged_df['averagerating'].mean())
merged_df['numvotes'] = merged_df['numvotes'].fillna(0)  # movies with
no votes

# 5. Handle movie_id and original_title
merged_df['movie_id'] = merged_df['movie_id'].fillna("Unknown")
merged_df['original_title'] =
merged_df['original_title'].fillna(merged_df['movie'])
```

```python
# 6. Double-check
print(merged_df.isnull().sum())
```

```
id                      0
release_date            0
movie                   0
production_budget       0
domestic_gross          0
worldwide_gross         0
studio                  0
foreign_gross           0
year                    0
movie_id                0
original_title          0
start_year           3598
runtime_minutes         0
genres                  0
averagerating           0
numvotes                0
ROI                     0
profit_margin         367
release_month           0
release_month_name      0
genre                   0
release_year            0
dtype: int64
```

```python
# Drop start_year if redundant
if 'start_year' in merged_df.columns:
    merged_df = merged_df.drop(columns=['start_year'])

# Handle profit_margin: replace NaN with 0 (optional)
merged_df['profit_margin'] = merged_df['profit_margin'].fillna(0)

# Double check
print(merged_df.isnull().sum())
```

```
id                   0
release_date         0
movie                0
production_budget    0
domestic_gross       0
worldwide_gross      0
studio               0
foreign_gross        0
year                 0
movie_id             0
original_title       0
runtime_minutes      0
```

```
genres                  0
averagerating           0
numvotes                0
ROI                     0
profit_margin           0
release_month           0
release_month_name      0
genre                   0
release_year            0
dtype: int64
```

The dataset is clean and now reay for analysis. Fit a regression model

4. REGRESSION ANALYSIS

How does a movie's budget affect its revenue?

Creating the model and undertaking regression analysis

Variables and the Model

**1.Dependent variable (response):Y= Worldwide gross (total revenue).

**2.Independent variable (predictor):X= Production budget.We fit a simple linear regression model

The general equation of simple linear regression equation is

Worldwide Gross = $\beta 0 + \beta 1$xProduction Budget

Where: β_0 (Intercept): baseline worldwide gross when budget = 0.

β_1 (Slope): change in worldwide gross for each unit change in budget.

First step is to formulate the hypotheses Hypotheses

Hypotheses

H_0: Budget does not affect revenue ($\beta_1 = 0$). H_1: Budget affects revenue ($\beta_1 \neq 0$, or $\beta_1 > 0$ in a one-sided test)..

4.1 Import the libraries to use

```
import pandas as pd
import scipy.stats as stats
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
```

Building th Regression Model

```
# Define X (predictors) and y (response)
X = merged_df['production_budget']
y =  merged_df['worldwide_gross']

rho = 
np.corrcoef(merged_df['worldwide_gross'],merged_df['production_budget'
])[0,1]
```

rho is the correlation between dependent variable (worldwide_gross) and independent variable(production_budget) which has to be in the range of 0 and 1

```
s_x = merged_df["production_budget"].std()
s_y = merged_df["worldwide_gross"].std()
```

s_x is the variance of x and s_y is the variance of y

```
m = rho * s_y / s_x
```

m is the slope of the regression equation and is the coefficient of x

```
c = merged_df["worldwide_gross"].mean() - m * 
merged_df["production_budget"].mean()
```

c is the y intercept or a contant

```
mean_y = merged_df["worldwide_gross"].mean()
mean_x = merged_df["production_budget"].mean()
```

mean_y is the mean of y and mean_x is the mean of x

```
c = mean_y - m * mean_x

print(f"Regression line: y = {round(m,4)}x + {round(c,4)}")

Regression line: y = 3.1269x + -7285667.0546
```

i.e Worldwide Gross=3.1269xProduction Budget -7285667.0546, the regression equation Slope (3.1223)

For every 1 unit increase in production budget, 1 million dollars the model predicts that worldwide gross increases by about 3.12 units.

Each additional $1 million in production budget is associated with about $3.12 million more in worldwide gross

Intercept (-7285667.0546) This is the predicted worldwide gross when the production budget is 0. It equals about –$7.286 million.

The model suggests that movies with larger production budgets tend to earn higher worldwide gross.

Creating the Model

```
X = merged_df['production_budget']
y = merged_df['worldwide_gross']

model = sm.OLS(endog = y, exog=sm.add_constant(X))
model
```

```
<statsmodels.regression.linear_model.OLS at 0x2d408ba24f0>
```

```
results = model.fit()
results
```

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at
0x2d40a2170a0>
```

Model Evaluation

```
results.fvalue
```

```
7355.224292071473
```

```
results.f_pvalue
```

```
0.0
```

The p value is less than 0.05 Or 5% level of significance

```
results.rsquared
```

```
0.5599618345695967
```

R_square of0.5599618345695969 implies 56% of the variation in dependent variable is explained by independent variable i.e the model accounts for 56% variation in worldwide_gross

```
print(results.summary())
```

```
                            OLS Regression Results                            
==============================================================================
========
Dep. Variable:        worldwide_gross   R-squared:                       
0.560
Model:                            OLS   Adj. R-squared:                  
0.560
Method:                 Least Squares   F-statistic:                     
7355.
Date:                Mon, 15 Sep 2025   Prob (F-statistic):
```

```
0.00
Time:                        02:06:32   Log-Likelihood:            -
1.1557e+05
No. Observations:                5782   AIC:
2.311e+05
Df Residuals:                    5780   BIC:
2.311e+05
Df Model:                           1

Covariance Type:            nonrobust

======================================================================
===============
                        coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
---------------
const             -7.286e+06   1.91e+06     -3.813      0.000    -
1.1e+07   -3.54e+06
production_budget     3.1269      0.036     85.763      0.000
3.055       3.198
======================================================================
========
Omnibus:                     4232.022   Durbin-Watson:
1.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
172398.262
Skew:                           3.053   Prob(JB):
0.00
Kurtosis:                      29.044   Cond. No.
6.57e+07
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 6.57e+07. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

p_value = 0.0 which is less tha 0.05 hence we conclude budget significantly affects revenue. The f statistic or f value is 8149 and its prbability value is 0.00 which is significant

Perform log–log transformation (e.g., log(worldwide_gross) ~ log(production_budget)) would help stabilize variance and improve regression modeling. instead of a simple linear regression

```python
# Keep only rows with positive values
df_model = df_budgets[(df_budgets['production_budget'] > 0) &
```

```
                      (df_budgets['worldwide_gross'] > 0)].copy()

# Log transformation
df_model['log_budget'] = np.log(df_model['production_budget'])
df_model['log_gross'] = np.log(df_model['worldwide_gross'])

 #Fit the log log model
 # Fit OLS regression: log(worldwide_gross) ~ log(production_budget)
import statsmodels.formula.api as smf
model = smf.ols('log_gross ~ log_budget', data=df_model).fit()

# Model summary
print(model.summary())
```

```
                            OLS Regression Results

======================================================================
========
Dep. Variable:              log_gross   R-squared:
0.482
Model:                            OLS   Adj. R-squared:
0.482
Method:                 Least Squares   F-statistic:
5039.
Date:                Mon, 15 Sep 2025   Prob (F-statistic):
0.00
Time:                        02:19:46   Log-Likelihood:
-10890.
No. Observations:                5415   AIC:
2.178e+04
Df Residuals:                    5413   BIC:
2.180e+04
Df Model:                           1

Covariance Type:            nonrobust

======================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------
--------
Intercept     -0.9920      0.252     -3.943      0.000      -1.485
-0.499
log_budget     1.0798      0.015     70.983      0.000       1.050
1.110
======================================================================
========
Omnibus:                     1069.770   Durbin-Watson:
0.873
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
2910.168
Skew:                          -1.054   Prob(JB):
0.00
Kurtosis:                       5.908   Cond. No.
170.
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

Model Fit (Goodness of Fit)

R-squared = 0.482 (48.2%)

About half of the variation in worldwide gross is explained by production budget (on the log scale).

This is a moderately strong relationship for real-world movie data, where many other factors (genre, star power, release timing, marketing, etc.) also matter.

F-statistic = 5039, $p < 0.001$

The model overall is highly statistically significant.

1. Coefficients

Intercept = -0.9920 ($p < 0.001$)

This is the baseline log(gross) when log(budget) = 0.

Not very interpretable in practical terms, since budget = 1 dollar is unrealistic.

log_budget = 1.0798 ($p < 0.001$)

Statistically significant at a very high confidence level.

Interpretation: A 1% increase in production budget is associated with about a 1.08% increase in worldwide gross (on average). Since the coefficient is slightly above 1, returns are slightly more than proportional, but diminishing effects likely appear once other variables are included.

Residual Diagnostics (from bottom of summary) Omnibus / Jarque-Bera test ($p < 0.001$) → residuals are not perfectly normal. Skew = -1.05 → distribution of residuals is left-skewed. Kurtosis = 5.9 → heavy tails (leptokurtic). Durbin-Watson = 0.873 → indicates some autocorrelation in residuals (common in time-related data like release years).

Summary of Findings Budget strongly predicts worldwide revenue. Elasticity ≈ 1.08: if a studio doubles a movie's budget, expected worldwide gross is more than doubled (but with wide variation). However, the model explains only about half of the variation (48.2%) → meaning other drivers like genre, directors, ratings, and release factors are also crucial. Residual tests suggest heteroscedasticity and non-normality — confirming the need to extend the model with additional predictors.

Is not a better model compared to a simple linear regression model which explains 56% of variation in the worldwide gross but helps in stabilization of the variance

Objective 2: To determine what patterns emerge from audience ratings and runtimes, and how they affect a movie's worldwide gross

Fit a regression model to answer the the question in the objective Formulate hypotheses

Dependent Variable (Y): worldwide_gross

Independent Variables (X): averagerating (IMDB average rating, 1–10 scale).

```
                      runtime_minutes (movie duration)
```

Model specification y = β + β1x1 +β2x2 worldwide_gross = constant + β1xruntime_minutes + β2xaveragerating

```
# Select relevant columns
cols = ['worldwide_gross', 'runtime_minutes', 'averagerating']
df_reg = merged_df[cols].dropna()

# Define X and y
X = df_reg[['runtime_minutes', 'averagerating']]
y = df_reg['worldwide_gross']

# Add constant for intercept
X = sm.add_constant(X)

# Fit regression model
model = sm.OLS(y, X).fit()

# Show summary
print(model.summary())

                            OLS Regression Results

==============================================================================
========
Dep. Variable:         worldwide_gross   R-squared:
0.056
Model:                             OLS   Adj. R-squared:
0.055
Method:                  Least Squares   F-statistic:
170.4
Date:                 Mon, 15 Sep 2025   Prob (F-statistic):
```

```
1.22e-72
Time:                        02:19:46   Log-Likelihood:            -
1.1777e+05
No. Observations:                5782   AIC:
2.356e+05
Df Residuals:                    5779   BIC:
2.356e+05
Df Model:                           2

Covariance Type:            nonrobust

======================================================================
=============
                      coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
-------------
const           -3.527e+08   2.46e+07    -14.350      0.000   -
4.01e+08   -3.05e+08
runtime_minutes  2.669e+06   1.95e+05     13.670      0.000
2.29e+06    3.05e+06
averagerating    2.714e+07   3.23e+06      8.391      0.000
2.08e+07    3.35e+07
======================================================================
========
Omnibus:                     5307.161   Durbin-Watson:
0.538
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
266081.968
Skew:                           4.312   Prob(JB):
0.00
Kurtosis:                      35.095   Cond. No.
1.15e+03
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.15e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

The Model explains only 5% of the variation in the worldwide gross being explaine by runtime minutes and average raring

For Audience Rating (β_1): H_0: $\beta_1 = 0$ → Audience rating has no effect on worldwide gross. H_1: $\beta_1 \neq 0$ → Audience rating significantly affects worldwide gross.

For Runtime (β_2): H_0: $\beta_2 = 0$ → Runtime has no effect on worldwide gross. H_1: $\beta_2 \neq 0$ → Runtime significantly affects worldwide gross.

Decision Rule (Statistical Test) Look at p-values from regression summary: If p = 1.22e-72 < 0.05 → reject H_0 (the variable significantly affects gross).

$R^2 = 0.056$: tells us that 56% of the variation in revenue is explained by ratings and runtime combined

```
# Correlation between runtime_minutes and averagerating
corr_value =
merged_df['runtime_minutes'].corr(merged_df['averagerating'])
print("Correlation:", corr_value)

Correlation: 0.2621999036719547
```

The two predictor variables have a weak positive correlation. Hence multicollinearity is not a major problem

Deductions from the summary

1. The Model Has Some Predictive Power, But It's Weak

The R-squared of 0.056 means that only about 5.6% of the variation in how much money a movie makes can be explained by how long it is and what its average rating is.

This is a very low number. It tells us that while these factors have a statistically measurable effect, they are not the main drivers of a movie's financial success. Other factors not included here (like marketing budget, franchise power, star actors, genre, competition, etc.) are far more important.

1. The Relationship Between Factors and Money

Longer Movies Make More Money: For each additional minute a movie runs, the model predicts it will make about $2.66 million more. This makes intuitive sense, as longer films are often bigger-budget blockbusters.

Higher Ratings Make More Money: For each additional point on a 10-point rating scale, a movie is predicted to make about $27.1 million more. This confirms that better-reviewed movies tend to perform better at the box office.

Regression Equation log(Worldwide Gross) = $\beta 0$ + $\beta 1$(Average Rating) + $\beta 2$(Runtime Minutes) + ϵ

Where: β_0: Intercept. β_1: Effect of a 1-unit increase in audience rating (e.g., from 6 → 7). β_2: Effect of an extra minute of runtime. ε: Error term.

```
import numpy as np
import statsmodels.formula.api as smf

# Ensure positive gross values
df_model = merged_df[(merged_df['worldwide_gross'] > 0) &
```

```
(merged_df['runtime_minutes'] > 0)].copy()

# Log transform gross (skewed distribution)
df_model['log_gross'] = np.log(df_model['worldwide_gross'])

# Fit regression model
model = smf.ols('log_gross ~ averagerating + runtime_minutes',
data=df_model).fit()

# Summary
print(model.summary())
```

```
                            OLS Regression Results

======================================================================
========
Dep. Variable:              log_gross   R-squared:
0.029
Model:                            OLS   Adj. R-squared:
0.029
Method:                 Least Squares   F-statistic:
82.12
Date:                Mon, 15 Sep 2025   Prob (F-statistic):
7.34e-36
Time:                        02:19:46   Log-Likelihood:
-12591.
No. Observations:                5415   AIC:
2.519e+04
Df Residuals:                    5412   BIC:
2.521e+04
Df Model:                           2

Covariance Type:            nonrobust

======================================================================
=============
                      coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
-------------
Intercept          12.0402      0.388     31.012      0.000
11.279      12.801
averagerating       0.2687      0.053      5.086      0.000
0.165       0.372
runtime_minutes     0.0296      0.003      9.979      0.000
0.024       0.035
======================================================================
========
Omnibus:                     1169.306   Durbin-Watson:
0.496
```

```
Prob(Omnibus):                   0.000   Jarque-Bera (JB):
2350.187
Skew:                           -1.287   Prob(JB):
0.00
Kurtosis:                        4.948   Cond. No.
1.21e+03
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.21e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

R-squared = 0.029 → The model explains only 2.9% of the variation in worldwide gross.

This means that audience ratings and runtimes alone are very weak predictors of box office revenue. Other variables (like budget, genre, and marketing) are much more important.

Statistical Significance Both predictors (ratings and runtime) are highly significant ($p < 0.001$). However, significance ≠ strong explanatory power — their effect is real, but limited in explaining overall variance.

Fitting a log linear regression still leads to the same conclussion

Objective 3: Identify whether movie genres and directors significantly explain variation in worldwide gross (returns), and determine which ones consistently outperform others.

Variables

Dependent variable (response): worldwide_gross (or ROI if you want profitability instead of raw revenue).

Independent variables (predictors): genre (categorical, one-hot encoded) director (categorical, one-hot encoded)

Hypotheses

Overall Model Hypotheses

Null Hypothesis (H_0): Movie genres and directors do not significantly explain variation in worldwide gross. (All genre and director coefficients = 0 after controlling for others.)

Alternative Hypothesis (H_1): At least one genre or one director has a significant effect on worldwide gross. (At least one coefficient ≠ 0.)

1. Hypotheses for Genres

H_0 (Genres): Average worldwide gross is the same across all movie genres. H_1 (Genres): At least one genre has a different average worldwide gross compared to others. (Some genres systematically outperform or underperform.)

1. Hypotheses for Directors

H_0 (Directors): Average worldwide gross does not differ significantly by director. H_1 (Directors): At least one director consistently yields significantly different worldwide gross than others.

Model specification:

```
y = β0+β1x1 + β2x2 + ϵ

final_df = pd.merge(
    merged_df,
    ratings_with_director,
    on="id",
    how="inner"
)

final_df

        id release_date         movie  production_budget
domestic_gross  \
0        3   2019-06-07  Dark Phoenix        350000000.0
42762350.0
1        3   2019-06-07  Dark Phoenix        350000000.0
42762350.0
2        3   2019-06-07  Dark Phoenix        350000000.0
42762350.0
3        3   2019-06-07  Dark Phoenix        350000000.0
42762350.0
4        3   2019-06-07  Dark Phoenix        350000000.0
42762350.0
...     ..          ...           ...                ...             .
..
104053  99   2015-07-07  Tiger Orange           100000.0
0.0
104054  99   2015-07-07  Tiger Orange           100000.0
0.0
104055  99   2015-07-07  Tiger Orange           100000.0
0.0
104056  99   2015-07-07  Tiger Orange           100000.0
0.0
104057  99   2015-07-07  Tiger Orange           100000.0
0.0

        worldwide_gross   studio  foreign_gross  year   movie_id  ...
\
0           149762350.0  Unknown            0.0  2019  tt6565702  ...
```

```
1            149762350.0  Unknown            0.0  2019  tt6565702  ...

2            149762350.0  Unknown            0.0  2019  tt6565702  ...

3            149762350.0  Unknown            0.0  2019  tt6565702  ...

4            149762350.0  Unknown            0.0  2019  tt6565702  ...

...                  ...      ...            ...   ...        ...  ...

104053               0.0  Unknown            0.0  2015  tt2866824  ...

104054               0.0  Unknown            0.0  2015  tt2866824  ...

104055               0.0  Unknown            0.0  2015  tt2866824  ...

104056               0.0  Unknown            0.0  2015  tt2866824  ...

104057               0.0  Unknown            0.0  2015  tt2866824  ...


                          genre  release_year  \
0       Action,Adventure,Sci-Fi          2019
1       Action,Adventure,Sci-Fi          2019
2       Action,Adventure,Sci-Fi          2019
3       Action,Adventure,Sci-Fi          2019
4       Action,Adventure,Sci-Fi          2019
...                         ...           ...
104053                    Drama          2015
104054                    Drama          2015
104055                    Drama          2015
104056                    Drama          2015
104057                    Drama          2015

                                                   review  rating
fresh  \
0       A distinctly gallows take on contemporary fina...     3.0
fresh
1       Quickly grows repetitive and tiresome, meander...     3.0
rotten
2       Cronenberg is not a director to be daunted by ...     2.0
rotten
3       Robert Pattinson works mighty hard to make Cos...     2.0
rotten
4       The anger over the injustice of the financial ...     2.0
fresh
...                                                   ...     ...
...
104053  Two Weeks Notice will find its audience of mid...     1.0
rotten
```

```
104054  Like a medium-grade network sitcom--mostly ino...     3.0
rotten
104055  Bullock and Grant, who made for memorable inte...     2.0
fresh
104056  What can I write about Two Weeks Notice that I...     2.0
rotten
104057  Two Weeks Notice is a defiantly unoriginal fil...     4.0
rotten

                    critic  top_critic                 publisher  \
0               PJ Nabarro           0           Patrick Nabarro
1           Eric D. Snider           0          EricDSnider.com
2             Matt Kelemen           0        Las Vegas CityLife
3          Christian Toto           0             Big Hollywood
4             Robert Roten           0        Laramie Movie Scope
...                    ...         ...                       ...
104053         Walter Chaw           0        Film Freak Central
104054      Frank Swietek           0         One Guy's Opinion
104055      Laura Clifford           0           Reeling Reviews
104056  James Berardinelli           1                ReelViews
104057      Philip Martin           0  Arkansas Democrat-Gazette

                     date          director
0       November 10, 2018  David Cronenberg
1           July 17, 2013  David Cronenberg
2          April 21, 2013  David Cronenberg
3        January 15, 2013  David Cronenberg
4         January 7, 2013  David Cronenberg
...                   ...               ...
104053  December 19, 2002
104054  December 18, 2002
104055  December 18, 2002
104056  December 17, 2002
104057  December 16, 2002

[104058 rows x 29 columns]
```

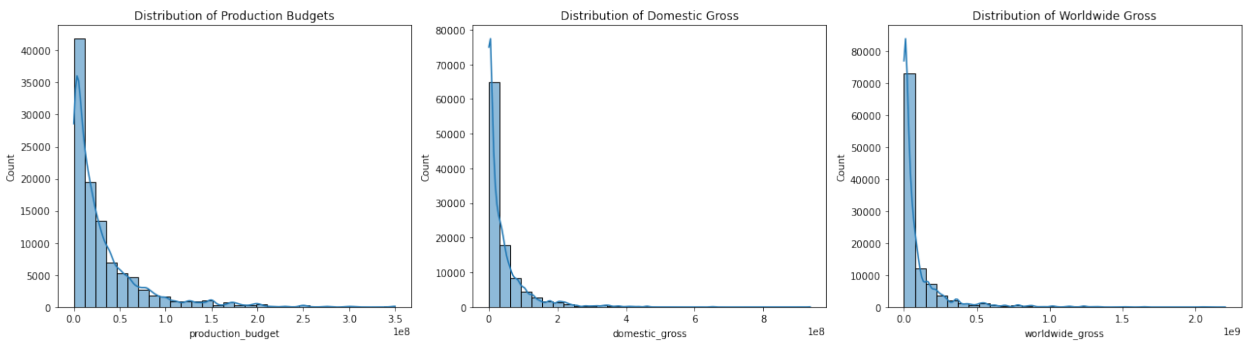
Visualizing the Dataset

You can explore the distribution of the rating column and also how ratings vary by director, critic, or freshness

a. 1. Distribution of Production Budget & Gross

```
import seaborn as sns
import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 3, figsize=(18,5))
```

```
sns.histplot(final_df['production_budget'], bins=30, kde=True,
ax=axes[0])
axes[0].set_title("Distribution of Production Budgets")

sns.histplot(final_df['domestic_gross'], bins=30, kde=True,
ax=axes[1])
axes[1].set_title("Distribution of Domestic Gross")

sns.histplot(final_df['worldwide_gross'], bins=30, kde=True,
ax=axes[2])
axes[2].set_title("Distribution of Worldwide Gross")

plt.tight_layout()
plt.show()
```



Interpretation

1. Production Budget Shape: Strong right skew. Most movies have low to moderate budgets, clustered below $50M. A few films have extremely high budgets (hundreds of millions), which are outliers compared to the bulk of films. Implication: The movie industry is dominated by low-to-mid budget films, but a handful of blockbusters pull the distribution's tail far to the right.

2. Domestic Gross Shape: Again highly right skewed. Most movies earn under $50M domestically. Very few films exceed $200M in U.S. box office, but those that do are extreme outliers. Implication: Only a small fraction of films become major domestic hits, while the majority gross relatively little.

3. Worldwide Gross Shape: Also strongly right skewed. Most movies worldwide gross under $100M. A very small number of global blockbusters exceed $1B, stretching the distribution dramatically. Implication: The global box office is heavily driven by a few mega-hits, while the vast majority of films make modest amounts.

The loglinear model

$\log(\text{Worldwide Gross}) = \beta 0 + \beta 1(\text{Genre}) + \beta 2(\text{Director})$

```
import statsmodels.formula.api as smf
from statsmodels.stats.anova import anova_lm

# --- Prepare the data ---
df = final_df[['worldwide_gross', 'genre', 'director']].dropna()
df['log_gross'] = np.log1p(df['worldwide_gross'])  # log(1+x) handles
zeros safely

# --- Fit log-linear regression ---
model = smf.ols("log_gross ~ C(genre) + C(director)", data=df).fit()

# --- Model summary ---
print(model.summary())

# --- ANOVA: test if genres and directors jointly explain variation
---
anova_results = anova_lm(model, typ=2)
print("\nANOVA Results:\n", anova_results)

# --- Convert log-coefficients to percentage effects ---
effects = (np.exp(model.params) - 1) * 100
print("\nPercentage Effects vs Baseline:\n", effects.head(20))
```

```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:              log_gross   R-squared:
0.176
Model:                            OLS   Adj. R-squared:
0.173
Method:                 Least Squares   F-statistic:
82.50
Date:                Mon, 15 Sep 2025   Prob (F-statistic):
0.00
Time:                        02:20:02   Log-Likelihood:            -
3.0217e+05
No. Observations:              104058   AIC:
6.049e+05
Df Residuals:                  103789   BIC:
6.075e+05
Df Model:                         268

Covariance Type:            nonrobust

==============================================================================
================================================
                                                         coef    std
err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
------------------------------------------------
Intercept                                                    17.1103
0.296     57.902      0.000      16.531      17.689
C(genre)[T.Action,Adventure]                                 -7.9741
0.652    -12.233      0.000      -9.252      -6.696
C(genre)[T.Action,Adventure,Animation]                        3.0498
0.424      7.186      0.000       2.218       3.882
C(genre)[T.Action,Adventure,Biography]                        1.9103
0.532      3.592      0.000       0.868       2.953
C(genre)[T.Action,Adventure,Comedy]                          -2.1028
0.366     -5.744      0.000      -2.820      -1.385
C(genre)[T.Action,Adventure,Crime]                            1.5759
0.444      3.547      0.000       0.705       2.447
C(genre)[T.Action,Adventure,Drama]                            1.3400
0.346      3.868      0.000       0.661       2.019
C(genre)[T.Action,Adventure,Family]                           2.1493
1.440      1.492      0.136      -0.673       4.972
C(genre)[T.Action,Adventure,Fantasy]                          1.8215
0.348      5.234      0.000       1.139       2.504
C(genre)[T.Action,Adventure,Horror]                           1.7992
0.735      2.449      0.014       0.360       3.239
C(genre)[T.Action,Adventure,Mystery]                         -2.9442
0.487     -6.051      0.000      -3.898      -1.991
C(genre)[T.Action,Adventure,Sci-Fi]                           2.4919
0.340      7.333      0.000       1.826       3.158
C(genre)[T.Action,Adventure,Thriller]                         2.9109
0.469      6.201      0.000       1.991       3.831
C(genre)[T.Action,Adventure,Western]                          1.6762
0.588      2.851      0.004       0.524       2.829
C(genre)[T.Action,Animation,Comedy]                           0.0703
0.413      0.170      0.865      -0.740       0.880
C(genre)[T.Action,Biography,Crime]                           -2.2858
0.579     -3.951      0.000      -3.420      -1.152
C(genre)[T.Action,Biography,Documentary]                     -9.4862
0.493    -19.260      0.000     -10.452      -8.521
C(genre)[T.Action,Biography,Drama]                            1.8849
0.424      4.444      0.000       1.054       2.716
C(genre)[T.Action,Comedy]                                     0.6913
0.620      1.115      0.265      -0.524       1.906
C(genre)[T.Action,Comedy,Crime]                              -1.4373
0.356     -4.033      0.000      -2.136      -0.739
C(genre)[T.Action,Comedy,Drama]                              -7.5358
0.427    -17.634      0.000      -8.373      -6.698
C(genre)[T.Action,Comedy,Family]                              1.4055
0.661      2.128      0.033       0.111       2.700
C(genre)[T.Action,Comedy,Horror]                             -0.5990
0.567     -1.057      0.291      -1.710       0.512
C(genre)[T.Action,Comedy,Sci-Fi]                              1.2632
2.017      0.626      0.531      -2.689       5.216
```

```
C(genre)[T.Action,Comedy,Sport]                             1.1066
0.879      1.258      0.208      -0.617       2.830
C(genre)[T.Action,Crime]                                   -1.0213
0.937     -1.090      0.276      -2.858       0.816
C(genre)[T.Action,Crime,Drama]                             -0.1802
0.333     -0.541      0.589      -0.833       0.473
C(genre)[T.Action,Crime,Fantasy]                          -17.4729
1.844     -9.477      0.000     -21.087     -13.859
C(genre)[T.Action,Crime,Sci-Fi]                             0.0369
0.555      0.067      0.947      -1.051       1.125
C(genre)[T.Action,Crime,Sport]                              0.9451
0.721      1.311      0.190      -0.468       2.358
C(genre)[T.Action,Crime,Thriller]                          -0.2312
0.341     -0.678      0.497      -0.899       0.437
C(genre)[T.Action,Documentary,Drama]                        1.1814
2.250      0.525      0.599      -3.228       5.590
C(genre)[T.Action,Drama]                                   -0.3212
0.456     -0.704      0.481      -1.215       0.573
C(genre)[T.Action,Drama,Family]                             2.3771
0.489      4.861      0.000       1.419       3.336
C(genre)[T.Action,Drama,Fantasy]                           -0.4473
0.417     -1.073      0.283      -1.264       0.369
C(genre)[T.Action,Drama,History]                           -1.7312
0.506     -3.424      0.001      -2.722      -0.740
C(genre)[T.Action,Drama,Mystery]                            1.5201
0.609      2.496      0.013       0.326       2.714
C(genre)[T.Action,Drama,Romance]                          -17.2778
0.627    -27.540      0.000     -18.507     -16.048
C(genre)[T.Action,Drama,Sci-Fi]                             1.0413
0.549      1.898      0.058      -0.034       2.117
C(genre)[T.Action,Drama,Thriller]                          -2.1768
0.398     -5.463      0.000      -2.958      -1.396
C(genre)[T.Action,Drama,War]                                1.4040
4.470      0.314      0.753      -7.357      10.165
C(genre)[T.Action,Fantasy,Horror]                           1.3380
0.423      3.164      0.002       0.509       2.167
C(genre)[T.Action,Fantasy,War]                              2.3391
0.627      3.728      0.000       1.110       3.569
C(genre)[T.Action,Fantasy,Western]                         -1.0372
0.493     -2.106      0.035      -2.003      -0.072
C(genre)[T.Action,Horror]                                 -11.5724
1.315     -8.798      0.000     -14.150      -8.994
C(genre)[T.Action,Horror,Sci-Fi]                           -0.9451
1.441     -0.656      0.512      -3.770       1.880
C(genre)[T.Action,Horror,Thriller]                         -9.9819
0.472    -21.138      0.000     -10.907      -9.056
C(genre)[T.Action,Mystery,Sci-Fi]                           2.0836
4.470      0.466      0.641      -6.678      10.845
C(genre)[T.Action,Romance,Thriller]                       -16.8331
```

```
1.272    -13.234      0.000     -19.326     -14.340
C(genre)[T.Action,Sci-Fi]                                  3.0075
0.551      5.456      0.000       1.927       4.088
C(genre)[T.Action,Sci-Fi,Thriller]                        -3.9562
0.364    -10.876      0.000      -4.669      -3.243
C(genre)[T.Action,Sport]                                  -7.3554
0.976     -7.536      0.000      -9.269      -5.442
C(genre)[T.Action,Thriller]                               -9.4052
0.363    -25.931      0.000     -10.116      -8.694
C(genre)[T.Adventure]                                     -0.3468
0.373     -0.929      0.353      -1.079       0.385
C(genre)[T.Adventure,Animation]                           -2.9189
0.519     -5.621      0.000      -3.937      -1.901
C(genre)[T.Adventure,Animation,Comedy]                     2.5538
0.323      7.902      0.000       1.920       3.187
C(genre)[T.Adventure,Animation,Family]                     0.1294
0.405      0.319      0.750      -0.665       0.924
C(genre)[T.Adventure,Biography,Documentary]              -10.5034
1.619     -6.488      0.000     -13.676      -7.331
C(genre)[T.Adventure,Biography,Drama]                     -7.2321
0.392    -18.436      0.000      -8.001      -6.463
C(genre)[T.Adventure,Comedy]                               0.0522
0.507      0.103      0.918      -0.942       1.046
C(genre)[T.Adventure,Comedy,Crime]                        -0.8975
0.639     -1.405      0.160      -2.150       0.355
C(genre)[T.Adventure,Comedy,Drama]                         0.5462
0.422      1.295      0.195      -0.281       1.373
C(genre)[T.Adventure,Comedy,Family]                        2.2320
0.508      4.390      0.000       1.235       3.229
C(genre)[T.Adventure,Crime,Drama]                         -1.4413
1.017     -1.417      0.157      -3.435       0.553
C(genre)[T.Adventure,Crime,Thriller]                      -4.0757
1.228     -3.318      0.001      -6.483      -1.668
C(genre)[T.Adventure,Documentary,History]                 -5.3755
2.592     -2.074      0.038     -10.455      -0.296
C(genre)[T.Adventure,Drama]                               -3.3271
0.492     -6.764      0.000      -4.291      -2.363
C(genre)[T.Adventure,Drama,Family]                         0.9760
3.168      0.308      0.758      -5.233       7.185
C(genre)[T.Adventure,Drama,Fantasy]                       -0.6251
1.263     -0.495      0.621      -3.100       1.850
C(genre)[T.Adventure,Drama,Horror]                        -4.0420
0.490     -8.243      0.000      -5.003      -3.081
C(genre)[T.Adventure,Drama,Romance]                       -1.8605
1.189     -1.564      0.118      -4.191       0.470
C(genre)[T.Adventure,Drama,Sci-Fi]                       -13.6367
0.606    -22.485      0.000     -14.825     -12.448
C(genre)[T.Adventure,Drama,Thriller]                      -1.6502
0.569     -2.903      0.004      -2.765      -0.536
```

```
C(genre)[T.Adventure,Family,Fantasy]                          2.1649
0.397      5.459      0.000       1.388       2.942
C(genre)[T.Adventure,Fantasy]                                 3.5161
2.593      1.356      0.175      -1.566       8.598
C(genre)[T.Adventure,Fantasy,Mystery]                        -0.1213
0.656     -0.185      0.853      -1.408       1.165
C(genre)[T.Adventure,Horror,Sci-Fi]                          -5.4938
0.976     -5.628      0.000      -7.407      -3.581
C(genre)[T.Animation]                                       -11.2362
0.879    -12.776      0.000     -12.960      -9.512
C(genre)[T.Animation,Comedy,Drama]                           -1.4056
0.879     -1.599      0.110      -3.129       0.318
C(genre)[T.Animation,Comedy,Family]                          -0.7269
1.845     -0.394      0.694      -4.343       2.889
C(genre)[T.Animation,Drama,Fantasy]                           0.3575
0.721      0.496      0.620      -1.056       1.771
C(genre)[T.Biography]                                        -6.0041
1.121     -5.354      0.000      -8.202      -3.806
C(genre)[T.Biography,Comedy,Crime]                           -1.2751
0.631     -2.021      0.043      -2.512      -0.038
C(genre)[T.Biography,Comedy,Drama]                           -0.6089
0.387     -1.575      0.115      -1.367       0.149
C(genre)[T.Biography,Crime,Drama]                            -0.0443
0.408     -0.109      0.914      -0.843       0.755
C(genre)[T.Biography,Documentary]                           -16.6051
0.616    -26.945      0.000     -17.813     -15.397
C(genre)[T.Biography,Documentary,Drama]                       0.9247
0.643      1.437      0.151      -0.336       2.186
C(genre)[T.Biography,Documentary,History]                    -1.5755
0.455     -3.461      0.001      -2.468      -0.683
C(genre)[T.Biography,Documentary,Music]                      -0.2790
2.592     -0.108      0.914      -5.359       4.801
C(genre)[T.Biography,Drama]                                  -0.6053
0.416     -1.456      0.145      -1.420       0.210
C(genre)[T.Biography,Drama,Family]                           -0.0652
0.602     -0.108      0.914      -1.245       1.115
C(genre)[T.Biography,Drama,Fantasy]                           0.9156
0.556      1.648      0.099      -0.173       2.004
C(genre)[T.Biography,Drama,History]                          -0.2902
0.370     -0.784      0.433      -1.016       0.436
C(genre)[T.Biography,Drama,Music]                            -0.3853
1.092     -0.353      0.724      -2.526       1.755
C(genre)[T.Biography,Drama,Musical]                           1.8056
0.976      1.850      0.064      -0.108       3.719
C(genre)[T.Biography,Drama,Mystery]                          -1.8084
0.627     -2.883      0.004      -3.038      -0.579
C(genre)[T.Biography,Drama,Romance]                          -0.2244
0.764     -0.293      0.769      -1.723       1.274
C(genre)[T.Biography,Drama,Sport]                            -0.3739
```

```
0.384     -0.974      0.330      -1.126        0.379
C(genre)[T.Biography,Drama,Thriller]                       -0.1461
0.376     -0.388      0.698      -0.884        0.592
C(genre)[T.Biography,Drama,War]                            -1.9597
2.593     -0.756      0.450      -7.041        3.122
C(genre)[T.Comedy]                                         -1.6610
0.324     -5.121      0.000      -2.297       -1.025
C(genre)[T.Comedy,Crime]                                   -0.3892
0.495     -0.786      0.432      -1.359        0.581
C(genre)[T.Comedy,Crime,Drama]                             -3.4243
0.356     -9.608      0.000      -4.123       -2.726
C(genre)[T.Comedy,Crime,Romance]                           -1.3901
0.497     -2.797      0.005      -2.364       -0.416
C(genre)[T.Comedy,Crime,Thriller]                          -0.3400
0.530     -0.642      0.521      -1.378        0.698
C(genre)[T.Comedy,Documentary]                              0.2189
1.040      0.210      0.833      -1.820        2.258
C(genre)[T.Comedy,Drama]                                   -5.3188
0.316    -16.815      0.000      -5.939       -4.699
C(genre)[T.Comedy,Drama,Family]                            -1.5923
0.459     -3.469      0.001      -2.492       -0.693
C(genre)[T.Comedy,Drama,Fantasy]                            1.6149
0.556      2.907      0.004       0.526        2.704
C(genre)[T.Comedy,Drama,History]                           -2.6762
0.574     -4.663      0.000      -3.801       -1.551
C(genre)[T.Comedy,Drama,Horror]                            -2.6595
4.430     -0.600      0.548     -11.343        6.024
C(genre)[T.Comedy,Drama,Music]                             -3.5532
0.396     -8.970      0.000      -4.330       -2.777
C(genre)[T.Comedy,Drama,Musical]                           -0.0368
0.639     -0.058      0.954      -1.290        1.216
C(genre)[T.Comedy,Drama,Mystery]                           -0.2412
1.272     -0.190      0.850      -2.734        2.252
C(genre)[T.Comedy,Drama,Romance]                           -2.9795
0.316     -9.435      0.000      -3.598       -2.361
C(genre)[T.Comedy,Drama,Sport]                              0.8157
0.497      1.643      0.100      -0.158        1.789
C(genre)[T.Comedy,Family]                                   0.8024
0.491      1.634      0.102      -0.160        1.765
C(genre)[T.Comedy,Family,Fantasy]                           2.6678
1.321      2.019      0.043       0.078        5.257
C(genre)[T.Comedy,Family,Romance]                           2.1659
2.250      0.963      0.336      -2.244        6.576
C(genre)[T.Comedy,Fantasy]                                  1.5589
2.593      0.601      0.548      -3.522        6.640
C(genre)[T.Comedy,Fantasy,Horror]                          -0.7403
0.545     -1.359      0.174      -1.808        0.328
C(genre)[T.Comedy,Fantasy,Romance]                          0.8960
2.593      0.346      0.730      -4.186        5.978
```

```
C(genre)[T.Comedy,Fantasy,Sci-Fi]                                    -0.7488
0.656     -1.141      0.254      -2.035        0.538
C(genre)[T.Comedy,Fantasy,Thriller]                                  -3.6092
1.515     -2.383      0.017      -6.578       -0.640
C(genre)[T.Comedy,Horror]                                            -4.3126
0.484     -8.919      0.000      -5.260       -3.365
C(genre)[T.Comedy,Horror,Mystery]                                   -16.9812
1.321    -12.854      0.000     -19.570      -14.392
C(genre)[T.Comedy,Horror,Romance]                                     0.8552
0.735      1.164      0.244      -0.584        2.295
C(genre)[T.Comedy,Horror,Sci-Fi]                                    -13.6448
1.320    -10.336      0.000     -16.232      -11.057
C(genre)[T.Comedy,Horror,Thriller]                                    0.8360
2.569      0.325      0.745      -4.200        5.872
C(genre)[T.Comedy,Music]                                              1.9943
0.879      2.268      0.023       0.271        3.718
C(genre)[T.Comedy,Music,Romance]                                    -16.0805
0.466    -34.535      0.000     -16.993      -15.168
C(genre)[T.Comedy,Music,War]                                         -2.0751
4.430     -0.468      0.640     -10.759        6.608
C(genre)[T.Comedy,Mystery]                                           -3.1977
2.593     -1.233      0.217      -8.279        1.884
C(genre)[T.Comedy,Mystery,Sci-Fi]                                     1.2666
4.470      0.283      0.777      -7.494       10.028
C(genre)[T.Comedy,Romance]                                           -1.8226
0.320     -5.687      0.000      -2.451       -1.194
C(genre)[T.Comedy,Romance,Sci-Fi]                                     3.1528
2.593      1.216      0.224      -1.929        8.234
C(genre)[T.Comedy,Romance,Sport]                                     -3.1298
4.470     -0.700      0.484     -11.890        5.631
C(genre)[T.Comedy,Romance,Thriller]                                 -17.0760
1.228    -13.903      0.000     -19.483      -14.669
C(genre)[T.Comedy,Sci-Fi]                                             0.5012
0.685      0.731      0.465      -0.842        1.844
C(genre)[T.Comedy,Western]                                          -17.0027
0.879    -19.333      0.000     -18.726      -15.279
C(genre)[T.Crime,Documentary]                                        -1.2408
0.665     -1.865      0.062      -2.545        0.063
C(genre)[T.Crime,Documentary,Drama]                                   1.0283
3.167      0.325      0.745      -5.179        7.236
C(genre)[T.Crime,Drama]                                              -2.7823
0.466     -5.966      0.000      -3.696       -1.868
C(genre)[T.Crime,Drama,History]                                      -0.8906
1.844     -0.483      0.629      -4.504        2.723
C(genre)[T.Crime,Drama,Horror]                                       -6.9866
0.390    -17.917      0.000      -7.751       -6.222
C(genre)[T.Crime,Drama,Mystery]                                       0.8004
0.400      2.000      0.045       0.016        1.585
C(genre)[T.Crime,Drama,Romance]                                      -2.5055
```

```
1.122     -2.234      0.025      -4.704      -0.307
C(genre)[T.Crime,Drama,Thriller]                           -2.6515
0.359     -7.391      0.000      -3.355      -1.948
C(genre)[T.Crime,Horror,Mystery]                            0.5764
0.574      1.004      0.315      -0.549       1.702
C(genre)[T.Crime,Horror,Thriller]                          -3.2751
2.592     -1.263      0.206      -8.356       1.805
C(genre)[T.Crime,Mystery,Thriller]                         -6.7862
0.513    -13.225      0.000      -7.792      -5.780
C(genre)[T.Crime,Thriller]                                 -0.4205
0.620     -0.678      0.498      -1.636       0.795
C(genre)[T.Documentary]                                    -4.6624
0.312    -14.964      0.000      -5.273      -4.052
C(genre)[T.Documentary,Drama]                             -10.5569
1.712     -6.168      0.000     -13.912      -7.202
C(genre)[T.Documentary,Drama,News]                          0.8734
0.631      1.384      0.166      -0.364       2.111
C(genre)[T.Documentary,Drama,Sport]                         1.6663
1.189      1.401      0.161      -0.665       3.997
C(genre)[T.Documentary,Family]                             -6.2640
1.434     -4.369      0.000      -9.074      -3.454
C(genre)[T.Documentary,Music]                               3.1903
3.167      1.007      0.314      -3.017       9.398
C(genre)[T.Documentary,News]                               -0.0013
1.516     -0.001      0.999      -2.972       2.969
C(genre)[T.Documentary,Sport]                              -5.2787
0.627     -8.414      0.000      -6.508      -4.049
C(genre)[T.Documentary,War]                                -3.7002
1.122     -3.299      0.001      -5.899      -1.502
C(genre)[T.Drama]                                          -4.2020
0.306    -13.714      0.000      -4.803      -3.601
C(genre)[T.Drama,Family]                                    1.2402
0.475      2.613      0.009       0.310       2.170
C(genre)[T.Drama,Family,Fantasy]                            3.3104
0.631      5.245      0.000       2.073       4.548
C(genre)[T.Drama,Family,Music]                             -1.3089
1.829     -0.716      0.474      -4.893       2.275
C(genre)[T.Drama,Family,Sport]                             -3.1771
1.122     -2.833      0.005      -5.375      -0.979
C(genre)[T.Drama,Fantasy]                                  -1.6722
0.577     -2.899      0.004      -2.803      -0.542
C(genre)[T.Drama,Fantasy,Horror]                           -0.0193
0.475     -0.041      0.968      -0.950       0.911
C(genre)[T.Drama,Fantasy,Musical]                          -3.9007
1.368     -2.851      0.004      -6.583      -1.219
C(genre)[T.Drama,Fantasy,Mystery]                          -3.0716
4.470     -0.687      0.492     -11.833       5.690
C(genre)[T.Drama,Fantasy,Romance]                           0.5907
0.506      1.168      0.243      -0.400       1.582
```

```
C(genre)[T.Drama,History]                                        -1.3821
0.665     -2.078      0.038      -2.686      -0.078
C(genre)[T.Drama,History,Romance]                                -1.4342
1.040     -1.379      0.168      -3.473       0.604
C(genre)[T.Drama,History,Sport]                                  -4.9260
0.879     -5.602      0.000      -6.649      -3.203
C(genre)[T.Drama,History,Thriller]                                1.4027
0.685      2.047      0.041       0.059       2.746
C(genre)[T.Drama,History,War]                                    -1.1138
0.420     -2.654      0.008      -1.936      -0.291
C(genre)[T.Drama,Horror]                                         -0.0522
0.516     -0.101      0.919      -1.064       0.960
C(genre)[T.Drama,Horror,Mystery]                                 -1.7473
0.463     -3.777      0.000      -2.654      -0.841
C(genre)[T.Drama,Horror,Sci-Fi]                                 -16.5283
0.656    -25.196      0.000     -17.814     -15.243
C(genre)[T.Drama,Horror,Thriller]                                 0.6702
0.591      1.134      0.257      -0.488       1.828
C(genre)[T.Drama,Music]                                          -2.4133
0.450     -5.368      0.000      -3.294      -1.532
C(genre)[T.Drama,Music,Musical]                                  -3.1640
0.708     -4.466      0.000      -4.553      -1.775
C(genre)[T.Drama,Music,Romance]                                   1.3596
1.122      1.212      0.225      -0.839       3.558
C(genre)[T.Drama,Musical,Romance]                                -5.4455
0.976     -5.579      0.000      -7.359      -3.532
C(genre)[T.Drama,Mystery]                                        -0.5693
1.040     -0.547      0.584      -2.608       1.469
C(genre)[T.Drama,Mystery,Romance]                                -0.1333
1.152     -0.116      0.908      -2.391       2.124
C(genre)[T.Drama,Mystery,Sci-Fi]                                 -0.7023
0.472     -1.487      0.137      -1.628       0.223
C(genre)[T.Drama,Mystery,Thriller]                               -0.4032
0.371     -1.088      0.277      -1.130       0.323
C(genre)[T.Drama,Romance]                                        -1.0427
0.334     -3.122      0.002      -1.697      -0.388
C(genre)[T.Drama,Romance,Sci-Fi]                                 -3.4216
0.691     -4.955      0.000      -4.775      -2.068
C(genre)[T.Drama,Romance,Thriller]                               -2.0559
0.585     -3.516      0.000      -3.202      -0.910
C(genre)[T.Drama,Romance,War]                                     0.7798
0.430      1.814      0.070      -0.063       1.623
C(genre)[T.Drama,Sci-Fi]                                         -6.3911
0.691     -9.250      0.000      -7.745      -5.037
C(genre)[T.Drama,Sport]                                          -1.1571
1.441     -0.803      0.422      -3.982       1.668
C(genre)[T.Drama,Thriller]                                       -5.6085
0.347    -16.182      0.000      -6.288      -4.929
C(genre)[T.Drama,Thriller,Western]                               -6.1133
```

```
1.829     -3.343      0.001      -9.697      -2.529
C(genre)[T.Drama,War]                                   0.4555
0.485      0.939      0.348      -0.496       1.407
C(genre)[T.Drama,Western]                              -2.5719
0.521     -4.937      0.000      -3.593      -1.551
C(genre)[T.Family]                                     -4.3273
0.452     -9.565      0.000      -5.214      -3.441
C(genre)[T.Family,Fantasy]                             -0.1731
2.592     -0.067      0.947      -5.253       4.907
C(genre)[T.Family,Fantasy,Musical]                      3.3810
1.604      2.107      0.035       0.236       6.526
C(genre)[T.Family,Sci-Fi]                             -17.5728
1.604    -10.952      0.000     -20.718     -14.428
C(genre)[T.Fantasy]                                    -0.0974
0.497     -0.196      0.844      -1.071       0.876
C(genre)[T.Fantasy,Horror]                             -1.1089
4.470     -0.248      0.804      -9.871       7.653
C(genre)[T.Fantasy,Horror,Thriller]                     1.1305
0.493      2.294      0.022       0.165       2.096
C(genre)[T.Horror]                                     -2.2614
0.329     -6.863      0.000      -2.907      -1.616
C(genre)[T.Horror,Music,Thriller]                      -1.6807
1.272     -1.321      0.186      -4.174       0.812
C(genre)[T.Horror,Musical]                            -17.6659
0.728    -24.282      0.000     -19.092     -16.240
C(genre)[T.Horror,Mystery]                              1.0150
2.593      0.392      0.695      -4.066       6.096
C(genre)[T.Horror,Mystery,Sci-Fi]                      -0.1132
0.497     -0.228      0.820      -1.087       0.860
C(genre)[T.Horror,Mystery,Thriller]                    -1.4148
0.345     -4.099      0.000      -2.091      -0.738
C(genre)[T.Horror,Sci-Fi,Thriller]                      0.8726
0.515      1.696      0.090      -0.136       1.881
C(genre)[T.Horror,Thriller]                            -5.9753
0.388    -15.401      0.000      -6.736      -5.215
C(genre)[T.Mystery,Sci-Fi,Thriller]                     2.4812
1.377      1.802      0.072      -0.217       5.180
C(genre)[T.Mystery,Thriller]                            1.1579
0.831      1.394      0.163      -0.470       2.786
C(genre)[T.Romance]                                    -2.4112
0.976     -2.471      0.013      -4.324      -0.498
C(genre)[T.Sci-Fi]                                    -16.6360
1.189    -13.989      0.000     -18.967     -14.305
C(genre)[T.Sci-Fi,Thriller]                            -5.3032
0.629     -8.433      0.000      -6.536      -4.071
C(genre)[T.Sport]                                      -2.0746
4.470     -0.464      0.643     -10.835       6.686
C(genre)[T.Thriller]                                   -5.9984
0.385    -15.589      0.000      -6.753      -5.244
```

```
C(genre)[T.Unknown]                                            -1.2180
0.295     -4.127      0.000      -1.796      -0.640
C(genre)[T.Western]                                           -17.1103
0.665    -25.718      0.000     -18.414     -15.806
C(director)[T.Alan Alda]                                       -0.3202
0.303     -1.058      0.290      -0.914       0.273
C(director)[T.Allen Hughes|Albert Hughes]                      -0.0343
0.163     -0.210      0.833      -0.354       0.286
C(director)[T.Allison Anders]                                  -0.4133
0.417     -0.992      0.321      -1.230       0.403
C(director)[T.Anatole Litvak]                                   0.6498
0.594      1.093      0.274      -0.515       1.815
C(director)[T.Andy Sidaris]                                     0.5282
0.592      0.892      0.372      -0.633       1.689
C(director)[T.Barry Levinson]                                  -0.1477
0.115     -1.289      0.197      -0.372       0.077
C(director)[T.Ben Younger]                                     -0.4176
0.073     -5.698      0.000      -0.561      -0.274
C(director)[T.Bill Froehlich]                                   0.4747
0.597      0.795      0.427      -0.696       1.646
C(director)[T.Bruce Beresford]                                  0.5556
0.094      5.891      0.000       0.371       0.740
C(director)[T.Carl Erik Rinsch]                                 0.8572
0.130      6.574      0.000       0.602       1.113
C(director)[T.David Cronenberg]                                 0.1945
0.068      2.878      0.004       0.062       0.327
C(director)[T.Ernst Lubitsch]                                  -0.2231
0.182     -1.228      0.219      -0.579       0.133
C(director)[T.Frank Marshall]                                  -0.4743
0.159     -2.976      0.003      -0.787      -0.162
C(director)[T.Jake Kasdan]                                      0.3932
0.088      4.448      0.000       0.220       0.566
C(director)[T.James Wong]                                       0.2317
0.086      2.681      0.007       0.062       0.401
C(director)[T.Jay Russell]                                      0.0553
0.137      0.404      0.686      -0.213       0.323
C(director)[T.Jim Jarmusch]                                     0.1675
0.083      2.029      0.042       0.006       0.329
C(director)[T.John Gilling]                                     0.1096
0.298      0.368      0.713      -0.475       0.694
C(director)[T.John Sayles]                                      0.4444
0.082      5.443      0.000       0.284       0.604
C(director)[T.John Woo]                                         0.1230
0.090      1.365      0.172      -0.054       0.300
C(director)[T.Jon Turteltaub]                                  -0.1076
0.116     -0.926      0.355      -0.336       0.120
C(director)[T.Keith Gordon]                                     0.1899
0.263      0.722      0.470      -0.326       0.705
C(director)[T.Ken Loach]                                        0.1854
0.095      1.951      0.051      -0.001       0.372
```

```
C(director)[T.Kevin Lima]                                   -0.1159
0.061     -1.890      0.059      -0.236       0.004
C(director)[T.Matt Bettinelli-Olpin|Tyler Gillett]           0.4553
0.101      4.496      0.000       0.257       0.654
C(director)[T.Michael Polish]                                0.4625
0.213      2.166      0.030       0.044       0.881
C(director)[T.Otto Preminger]                               -0.5230
0.348     -1.501      0.133      -1.206       0.160
C(director)[T.Paolo Sorrentino]                              0.3601
0.089      4.068      0.000       0.187       0.534
C(director)[T.Pat Proft]                                    -0.7170
0.150     -4.792      0.000      -1.010      -0.424
C(director)[T.Pauly Shore]                                  -0.5681
0.201     -2.822      0.005      -0.963      -0.173
C(director)[T.Peter Baldwin]                                 0.0565
0.350      0.162      0.872      -0.629       0.742
C(director)[T.Peter Cattaneo]                               -0.2772
0.169     -1.637      0.102      -0.609       0.055
C(director)[T.Ray Lawrence]                                 -0.3238
0.083     -3.879      0.000      -0.487      -0.160
C(director)[T.Richard Kelly]                                 0.3319
0.090      3.708      0.000       0.156       0.507
C(director)[T.Richard Linklater]                            -0.4789
0.421     -1.137      0.256      -1.305       0.347
C(director)[T.Rick Rosenthal]                                0.6655
0.414      1.606      0.108      -0.147       1.478
C(director)[T.Roy Ward Baker]                                0.8385
0.342      2.452      0.014       0.168       1.509
C(director)[T.Steve Boyum]                                  -0.1291
0.175     -0.737      0.461      -0.473       0.214
C(director)[T.Steven Spielberg]                              0.5960
0.096      6.180      0.000       0.407       0.785
C(director)[T.Taylor Hackford]                              -0.0377
0.138     -0.273      0.785      -0.309       0.233
C(director)[T.Terence Young]                                -0.3418
0.298     -1.145      0.252      -0.927       0.243
C(director)[T.Tom Hanks]                                     0.1353
0.064      2.108      0.035       0.009       0.261
C(director)[T.Werner Herzog]                                 0.3626
0.244      1.487      0.137      -0.115       0.841
C(director)[T.William Friedkin]                              0.2812
0.412      0.682      0.495      -0.527       1.089
C(director)[T.William Wellman]                               0.1622
0.345      0.470      0.638      -0.514       0.839
C(director)[T.Woody Allen]                                   0.8469
0.200      4.239      0.000       0.455       1.239
C(director)[T.Yimou Zhang]                                   0.0035
0.418      0.008      0.993      -0.815       0.822
======================================================================
```

```
========
Omnibus:                    43477.327   Durbin-Watson:
0.050
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
184189.207
Skew:                          -2.080   Prob(JB):
0.00
Kurtosis:                       8.017   Cond. No.
443.
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.

ANOVA Results:
                    sum_sq        df          F        PR(>F)
C(genre)     4.208219e+05     221.0  97.444935  0.000000e+00
C(director)  7.619068e+03      47.0   8.295779  2.018942e-55
Residual     2.028141e+06  103789.0        NaN           NaN

Percentage Effects vs Baseline:
 Intercept                                   2.697156e+09
C(genre)[T.Action,Adventure]               -9.996557e+01
C(genre)[T.Action,Adventure,Animation]      2.011035e+03
C(genre)[T.Action,Adventure,Biography]      5.754924e+02
C(genre)[T.Action,Adventure,Comedy]        -8.778867e+01
C(genre)[T.Action,Adventure,Crime]          3.834993e+02
C(genre)[T.Action,Adventure,Drama]          2.818891e+02
C(genre)[T.Action,Adventure,Family]         7.578859e+02
C(genre)[T.Action,Adventure,Fantasy]        5.180904e+02
C(genre)[T.Action,Adventure,Horror]         5.044766e+02
C(genre)[T.Action,Adventure,Mystery]       -9.473567e+01
C(genre)[T.Action,Adventure,Sci-Fi]         1.108384e+03
C(genre)[T.Action,Adventure,Thriller]       1.737302e+03
C(genre)[T.Action,Adventure,Western]        4.345240e+02
C(genre)[T.Action,Animation,Comedy]         7.287552e+00
C(genre)[T.Action,Biography,Crime]         -8.983032e+01
C(genre)[T.Action,Biography,Documentary]   -9.999241e+01
C(genre)[T.Action,Biography,Drama]          5.585453e+02
C(genre)[T.Action,Comedy]                   9.963694e+01
C(genre)[T.Action,Comedy,Crime]            -7.624242e+01
dtype: float64
```

```python
import pandas as pd
import scipy.stats as stats
import numpy as np

# --- Step 1: Build contingency table ---
```

```
contingency = pd.crosstab(final_df['director'], final_df['genre'])

# --- Step 2: Chi-Square Test of Independence ---
chi2, p, dof, expected = stats.chi2_contingency(contingency)

print("Chi-square Statistic:", chi2)
print("Degrees of Freedom:", dof)
print("p-value:", p)

# --- Step 3: Interpret result ---
if p < 0.05:
    print("□ Reject H₀: There is a significant association between directors and genres.")
else:
    print("□ Fail to reject H₀: No significant association between directors and genres.")

# --- Step 4: Compute Cramér's V (strength of association) ---
n = contingency.sum().sum()
phi2 = chi2 / n
r, k = contingency.shape
phi2corr = max(0, phi2 - ((k-1)*(r-1))/(n-1))
rcorr = r - ((r-1)**2)/(n-1)
kcorr = k - ((k-1)**2)/(n-1)
cramers_v = np.sqrt(phi2corr / min((kcorr-1), (rcorr-1)))

print("Cramér's V (0=weak, 1=strong):", cramers_v)
```

```
Chi-square Statistic: 332087.9522120169
Degrees of Freedom: 10387
p-value: 0.0
□ Reject H₀: There is a significant association between directors and genres.
Cramér's V (0=weak, 1=strong): 0.25652963419299984
```

A chi square test of indepence indicates that the variables genre and directors are independent of each other i.e no association between them.

The null hypothesis is rejected → Both genre and director significantly explain variation in worldwide gross.

Some genres (e.g., Action+Adventure+Animation, Action+Adventure+Biography) outperform, while others (Action+Adventure, Action+Adventure+Comedy) underperform relative to the baseline.

Effect sizes are large because coefficients are in log scale → small changes translate into big differences in revenue.

R^2 is modest (17%), showing that while genre/director matter, other factors (budget, stars, marketing, release date) also strongly influence revenue.

```
merged_df
```

```
      id release_date                                        movie  \
0      1   2009-12-18                                       Avatar
1      2   2011-05-20  Pirates of the Caribbean: On Stranger Tides
2      3   2019-06-07                                 Dark Phoenix
3      4   2015-05-01                      Avengers: Age of Ultron
4      5   2017-12-15            Star Wars Ep. VIII: The Last Jedi
...   ..          ...                                          ...
6468  78   2018-12-31                                       Red 11
6469  79   1999-04-02                                    Following
6470  80   2005-07-13                Return to the Land of Wonders
6471  81   2015-09-29                         A Plague So Pleasant
6472  82   2005-08-05                            My Date With Drew

      production_budget  domestic_gross  worldwide_gross   studio  \
0           425000000.0     760507625.0     2.776345e+09  Unknown
1           410600000.0     241063875.0     1.045664e+09       BV
2           350000000.0      42762350.0     1.497624e+08  Unknown
3           330600000.0     459005868.0     1.403014e+09       BV
4           317000000.0     620181382.0     1.316722e+09  Unknown
...                 ...             ...              ...      ...
6468             7000.0             0.0     0.000000e+00  Unknown
6469             6000.0         48482.0     2.404950e+05  Unknown
6470             5000.0          1338.0     1.338000e+03  Unknown
6471             1400.0             0.0     0.000000e+00  Unknown
6472             1100.0        181041.0     1.810410e+05  Unknown

      foreign_gross  year   movie_id  ... runtime_minutes  \
0               0.0  2009  tt1775309  ...            93.0
1       804600000.0  2011  tt1298650  ...           136.0
2               0.0  2019  tt6565702  ...           113.0
3       946400000.0  2015  tt2395427  ...           141.0
4               0.0  2017    Unknown  ...           102.0
...             ...   ...        ...  ...             ...
6468            0.0  2018  tt7837402  ...            77.0
6469            0.0  1999    Unknown  ...           102.0
6470            0.0  2005    Unknown  ...           102.0
6471            0.0  2015  tt2107644  ...            76.0
6472            0.0  2005    Unknown  ...           102.0

                         genres averagerating  numvotes        ROI  \
0                        Horror      6.100000      43.0   5.532577
1      Action,Adventure,Fantasy      6.600000  447624.0   1.546673
2       Action,Adventure,Sci-Fi      6.000000   24451.0  -0.572108
3       Action,Adventure,Sci-Fi      7.300000  665594.0   3.243841
4                       Unknown      6.246978       0.0   3.153696
...                         ...           ...       ...        ...
6468    Horror,Sci-Fi,Thriller      5.600000      43.0  -1.000000
6469                    Unknown      6.246978       0.0  39.082500
```

```
6470                     Unknown      6.246978       0.0   -0.732400
6471      Drama,Horror,Thriller      5.400000      72.0   -1.000000
6472                     Unknown      6.246978       0.0  163.582727

      profit_margin  release_month  release_month_name  \
0          0.846921             12            December
1          0.607331              5                 May
2         -1.337036              6                June
3          0.764364              5                 May
4          0.759251             12            December
...             ...            ...                 ...
6468       0.000000             12            December
6469       0.975051              4               April
6470      -2.736921              7                July
6471       0.000000              9           September
6472       0.993924              8              August

                          genre release_year
0                        Horror         2009
1      Action,Adventure,Fantasy         2011
2       Action,Adventure,Sci-Fi         2019
3       Action,Adventure,Sci-Fi         2015
4                       Unknown         2017
...                         ...          ...
6468     Horror,Sci-Fi,Thriller         2018
6469                    Unknown         1999
6470                    Unknown         2005
6471      Drama,Horror,Thriller         2015
6472                    Unknown         2005

[5782 rows x 21 columns]
```

ratings_with_director

```
         id                                             review  rating  \
0         3  A distinctly gallows take on contemporary fina...     3.0

1         3  Quickly grows repetitive and tiresome, meander...     3.0

2         3  Cronenberg is not a director to be daunted by ...     2.0

3         3  Robert Pattinson works mighty hard to make Cos...     2.0

4         3  The anger over the injustice of the financial ...     2.0

...     ...                                                ...     ...

30596  2000            Sleek, shallow, but frequently amusing.     2.0

30597  2000  The spaniel-eyed Jean Reno infuses Hubert with...     3.0
```

```
30598  2000  Manages to be somewhat well-acted, not badly a...     1.0

30599  2000  Arguably the best script that Besson has writt...     3.0

30600  2000  Dawdles and drags when it should pop; it doesn...     1.0


        fresh          critic  top_critic               publisher  \
0       fresh      PJ Nabarro           0         Patrick Nabarro
1      rotten  Eric D. Snider           0         EricDSnider.com
2      rotten    Matt Kelemen           0      Las Vegas CityLife
3      rotten  Christian Toto           0           Big Hollywood
4       fresh    Robert Roten           0     Laramie Movie Scope
...       ...             ...         ...                     ...
30596   fresh    Gene Seymour           1                 Newsday
30597   fresh    Megan Turner           1           New York Post
30598  rotten     Bob Strauss           0  Los Angeles Daily News
30599   fresh      Wade Major           0      Boxoffice Magazine
30600  rotten  Manohla Dargis           1       Los Angeles Times

                     date          director
0       November 10, 2018  David Cronenberg
1           July 17, 2013  David Cronenberg
2          April 21, 2013  David Cronenberg
3        January 15, 2013  David Cronenberg
4         January 7, 2013  David Cronenberg
...                   ...               ...
30596  September 27, 2002
30597  September 27, 2002
30598  September 27, 2002
30599  September 27, 2002
30600  September 26, 2002

[30601 rows x 9 columns]
```

```
print("merged_df columns:", merged_df.columns.tolist())
print("ratings_with_director columns:",
ratings_with_director.columns.tolist())
```

```
merged_df columns: ['id', 'release_date', 'movie',
'production_budget', 'domestic_gross', 'worldwide_gross', 'studio',
'foreign_gross', 'year', 'movie_id', 'original_title',
'runtime_minutes', 'genres', 'averagerating', 'numvotes', 'ROI',
'profit_margin', 'release_month', 'release_month_name', 'genre',
'release_year']
ratings_with_director columns: ['id', 'review', 'rating', 'fresh',
'critic', 'top_critic', 'publisher', 'date', 'director']
```

```
final_df = pd.merge(
    merged_df,
```

```
    ratings_with_director,
    on="id",
    how="inner"
)
```

5. Recommendations

1. **Focus on Drama-Family-Fantasy Mix**: Movies that blend drama, family themes, and fantasy elements give the best return on investment. Instead of making single-genre films, create stories that mix emotions with broad family appeal.
2. **Release Movies in July and November**: periods around July and pre-holiday November are when movies make the most money. Avoid September and October when films record low revenue collection.
3. **Smart Spending Beats Big Spending**: You don't need huge budgets to make big profits. The data shows mid-budget films often deliver better returns on every dollar invested, reducing financial risk while still having enough resources to create quality entertainment that audiences want to see.
4. **Work with Proven Directors**: Partner with directors who have a track record of making good movies that audiences love. Great directors help ensure both quality and return on investment are high.

6. Conclusion

Investing more in production budgets generally pays off. On average, each additional $1 spent increases worldwide revenue by ~$3. Over half (56%) of revenue variation can be explained by budget alone. But budget is not everything: some high-budget movies still underperform, while some modestly budgeted films overachieve. To maximize returns, studios should combine budget strategy with careful genre selection, strong directors, and audience-driven content.

While production budget remains the strongest driver of revenue, our analysis shows that better audience ratings and well-balanced runtimes also contribute to higher box office earnings. To succeed, the studio should invest not just in bigger budgets, but also in quality storytelling and keeping runtimes audience-friendly

Genres and directors do matter. Some combinations like Action–Adventure–Animation and Action–Adventure–Biography consistently outperform, while others underperform. To succeed, the studio should focus investment on high-return genres and proven directors, but balance this with budget strategy and audience-driven quality control.