Rilsoft Movie Data Analysis and Market Insights for a New Studio

Authors: John, Wilber, Maureen,Stella, Isaiah

Overview

Rilsoft aims to capitalize on the rising trend of movie creation by establishing a new studio. As newcomers to the industry, the team seeks to leverage data from leading movie review platforms to generate actionable insights that will guide their entry into the competitive movie-making landscape.

Business Problem

Rilsoft wants to venture in the movie industry to compete with other big companies that create original video content. They have the challenge of determining optimal approach to the market. The problem is how to balance financial investment, creative vision and market demand.

Objectives:

- To understand high-performing movie genres and provide recommendations based on film genres with the highest ratings
- To understand revenue projections and ROI based on the various movie genres.
- To help in analyzing the various roles for creative movie production

---

Business Questions:

- Which movie genres consistently achieve the highest ratings and high ROI?
- What are the projected revenue and return on investment (ROI) across different movie genres for strategic decision making?
- What roles contribute to the success of high performing movie and movie genres?
- What insights can be derived from top-performing movie genres to inform Rilsoft movie studio production?

Data Understanding

Data Sources and Relevance

The data comes from several reputable sources in the movie industry, including Box Office Mojo, IMDB, Rotten Tomatoes, TheMovieDB, and The Numbers. These datasets provide insights into various aspects of movie performance e.g BOM Detailed information on movie performances, release years, and industry statistics.These datasets collectively address questions about box office trends, movie profitability, audience preferences, and the relationship between budgets and revenue.Hence,the choice of datasets that would help us achieve the objectives are; the imdb dataset,bom.movie gross dataset and TN.movie budgets dataset

In [179…
```
# Importing python libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sqlite3
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
import scipy.stats as stats
%matplotlib inline
```

In [180…
```
# Exploring IM.db table names
conn=sqlite3.connect('Datasets/im.db')

df_imdb = pd.read_sql("""SELECT name
                      AS 'Table Names'
                      FROM sqlite_master
                      WHERE type='table';""", conn)
cursor = conn.cursor()
df_imdb
```

Out[180…

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

In [181…
```
#Viewing columns and data from the movie basic table
first_query = """SELECT * FROM movie_basics;"""
pd.read_sql(first_query, conn).head()
```

Out[181…

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy

In [182…

```
#Viewing columns and data from the person table
Querry = """SELECT * FROM persons;"""
pd.read_sql(Querry, conn).head()
```

Out[182…

	person_id	primary_name	birth_year	death_year	primary_profession
0	nm0061671	Mary Ellen Bauder	NaN	NaN	miscellaneous,production_manager,producer
1	nm0061865	Joseph Bauer	NaN	NaN	composer,music_department,sound_department
2	nm0062070	Bruce Baum	NaN	NaN	miscellaneous,actor,writer
3	nm0062195	Axel Baumann	NaN	NaN	camera_department,cinematographer,art_department
4	nm0062798	Pete Baxter	NaN	NaN	production_designer,art_department,set_decorator

In [183…

```
#Viewing movie gross data and columns
df_gross=pd.read_csv('Datasets/bom.movie_gross.csv')
df_gross
```

Out[183…

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

3387 rows × 5 columns

In [184…

```
# Checking for null values in the movie gross dataset
df_gross.isnull().sum()
```

Out[184...
```
title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64
```

In [185...
```python
# Checking the column data types
df_gross.dtypes
```

Out[185...
```
title              object
studio             object
domestic_gross    float64
foreign_gross      object
year                int64
dtype: object
```

In [186...
```python
#Importing tmdb.movies dataset
df2=pd.read_csv('Datasets/tmdb.movies.csv')
df2
```

Out[186...

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Ha and th Hallc
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	Ho You
2	2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Ir
3	3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	
4	4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	
...	...	...	...	...	...	...	...	
26512	26512	[27, 18]	488143	en	Laboratory Conditions	0.600	2018-10-13	L C
26513	26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.600	2018-05-01	_EXHIE
26514	26514	[14, 28, 12]	381231	en	The Last One	0.600	2018-10-01	The
26515	26515	[10751, 12, 28]	366854	en	Trailer Made	0.600	2018-06-22	Tra
26516	26516	[53, 27]	309885	en	The Church	0.600	2018-10-05	Tl

26517 rows × 10 columns

In [187...
```python
#Importing tn.movies dataset
df_budgets=pd.read_csv('Datasets/tn.movie_budgets.csv')
df_budgets
```

Out[187…

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	$7,000	$0	$0
5778	79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
5779	80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
5780	81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
5781	82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

5782 rows × 6 columns

In [188…

```
#Importing rt.movie_info dataset
df4=pd.read_csv('Datasets/rt.movie_info.tsv', sep='\t')
df4
```

Out[188…

	id	synopsis	rating	genre	director	writer	theater_dat
0	1	This gritty, fast-paced, and innovative police...	R	Action and Adventure|Classics|Drama	William Friedkin	Ernest Tidyman	Oct 9, 197
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 201
2	5	Illeana Douglas delivers a superb performance ...	R	Drama|Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 199
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama|Mystery and Suspense	Barry Levinson	Paul Attanasio|Michael Crichton	Dec 9, 199

	id	synopsis	rating	genre	director	writer	theater_dat
4	7	NaN	NR	Drama|Romance	Rodney Bennett	Giles Cooper	NaN
...	...	...	...	...	...	...	.
1555	1996	Forget terrorists or hijackers -- there's a ha...	R	Action and Adventure|Horror|Mystery and Suspense	NaN	NaN	Aug 18, 200
1556	1997	The popular Saturday Night Live sketch was exp...	PG	Comedy|Science Fiction and Fantasy	Steve Barron	Terry Turner|Tom Davis|Dan Aykroyd|Bonnie Turner	Jul 23, 199
1557	1998	Based on a novel by Richard Powell, when the l...	G	Classics|Comedy|Drama|Musical and Performing Arts	Gordon Douglas	NaN	Jan 1, 196
1558	1999	The Sandlot is a coming-of-age story about a g...	PG	Comedy|Drama|Kids and Family|Sports and Fitness	David Mickey Evans	David Mickey Evans|Robert Gunter	Apr 1, 199
1559	2000	Suspended from the force, Paris cop Hubert is ...	R	Action and Adventure|Art House and Internation...	NaN	Luc Besson	Sep 27, 200

1560 rows × 12 columns

In [189…
```
# Checking for null values
df4.isnull().sum()
```

Out[189…
```
id                 0
synopsis          62
rating             3
genre              8
director         199
writer           449
theater_date     359
dvd_date         359
currency        1220
box_office      1220
runtime           30
studio          1066
dtype: int64
```

In [190…
```
#Importing rt.reviews dataset
df5=pd.read_csv('Datasets/rt.reviews.tsv', sep='\t', encoding='latin-1')
df5
```

Out[190…

	id	review	rating	fresh	critic	top_critic	publisher	date
0	3	A distinctly gallows take on contemporary fina...	3/5	fresh	PJ Nabarro	0	Patrick Nabarro	November 10, 2018
1	3	It's an allegory in search of a meaning that n...	NaN	rotten	Annalee Newitz	0	io9.com	May 23, 2018
2	3	... life lived in a bubble in financial dealin...	NaN	fresh	Sean Axmaker	0	Stream on Demand	January 4, 2018
3	3	Continuing along a line introduced in last yea...	NaN	fresh	Daniel Kasman	0	MUBI	November 16, 2017
4	3	... a perverse twist on neorealism...	NaN	fresh	NaN	0	Cinema Scope	October 12, 2017
...	...	...	...	...	...	...	...	...
54427	2000	The real charm of this trifle is the deadpan c...	NaN	fresh	Laura Sinagra	1	Village Voice	September 24, 2002
54428	2000	NaN	1/5	rotten	Michael Szymanski	0	Zap2it.com	September 21, 2005
54429	2000	NaN	2/5	rotten	Emanuel Levy	0	EmanuelLevy.Com	July 17, 2005
54430	2000	NaN	2.5/5	rotten	Christopher Null	0	Filmcritic.com	September 7, 2003
54431	2000	NaN	3/5	fresh	Nicolas Lacroix	0	Showbizz.net	November 12, 2002

54432 rows × 8 columns

Summary of the Various Variables in the Movie Datasets

The Target variables include but not limited to; Revenue data (domestic_gross, foreign_gross, worldwide_gross). Budget data (production_budget). Popularity metrics (popularity, rating, vote_count). Categorical information (studio, original_language,ages, genre_ids)

The dataset variables to be utilized here are both quantitative and qualitative(categorical)

Data Preparation/Cleaning

In [191…

```
#Select movies with their ratings
movie_ratings = ("""SELECT
    mb.movie_id,
    mb.original_title,
    mb.primary_title,
    mb.start_year,
    mr.averagerating,
```

```
    mr.numvotes,
    mb.genres
FROM movie_basics mb
JOIN movie_ratings mr
ON mb.movie_id = mr.movie_id
""")
df_movie_ratings = pd.read_sql(movie_ratings,conn)
df_movie_ratings
```

Out[191…

	movie_id	original_title	primary_title	start_year	averagerating	numvotes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	7.0	77	Action,Crime,Drama
1	tt0066787	Ashad Ka Ek Din	One Day Before the Rainy Season	2019	7.2	43	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	6.9	4517	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	6.1	13	Comedy,Drama
4	tt0100275	La Telenovela Errante	The Wandering Soap Opera	2017	6.5	119	Comedy,Drama,Fantasy
...	...	...	...	...	...	...	...
73851	tt9913084	Diabolik sono io	Diabolik sono io	2019	6.2	6	Documentary
73852	tt9914286	Sokagin Çocuklari	Sokagin Çocuklari	2019	8.7	136	Drama,Family
73853	tt9914642	Albatross	Albatross	2017	8.5	8	Documentary
73854	tt9914942	La vida sense la Sara Amat	La vida sense la Sara Amat	2019	6.6	5	None
73855	tt9916160	Drømmeland	Drømmeland	2019	6.5	11	Documentary

73856 rows × 7 columns

In [192…

```
#Select relevant persons,professions involved in the movies
movie_ratings_1 = ("""
SELECT
    mb.movie_id,
    mb.original_title,
    mb.primary_title,
    mb.start_year,
    mb.genres,
    mr.averagerating,
    mr.numvotes,
    dr.person_id,
    pr.primary_name,
    pr.primary_profession
FROM movie_basics mb
JOIN movie_ratings mr
```

```
ON mb.movie_id = mr.movie_id
JOIN known_for dr
ON mb.movie_id = dr.movie_id
JOIN persons pr
ON pr.person_id = dr.person_id
""")
df_movie_ratings_known_for = pd.read_sql(movie_ratings_1,conn)
df_movie_ratings_known_for
```

Out[192…

	movie_id	original_title	primary_title	start_year	genres	averagerating	numvotes
0	tt0063540	Sunghursh	Sunghursh	2013	Action,Crime,Drama	7.0	77
1	tt0063540	Sunghursh	Sunghursh	2013	Action,Crime,Drama	7.0	77
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	Drama	6.9	4517
3	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	Drama	6.9	4517
4	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	Drama	6.9	4517
...	...	...	...	...	...	...	...
526723	tt9914286	Sokagin Çocuklari	Sokagin Çocuklari	2019	Drama,Family	8.7	136
526724	tt9914642	Albatross	Albatross	2017	Documentary	8.5	8
526725	tt9914942	La vida sense la Sara Amat	La vida sense la Sara Amat	2019	None	6.6	5
526726	tt9914942	La vida sense la Sara Amat	La vida sense la Sara Amat	2019	None	6.6	5
526727	tt9914942	La vida sense la Sara Amat	La vida sense la Sara Amat	2019	None	6.6	5

526728 rows × 10 columns

In [193…

```
#checking for null values
df_movie_ratings.isna().sum()
```

Out[193…

```
movie_id          0
original_title    0
primary_title     0
start_year        0
averagerating     0
numvotes          0
genres          804
dtype: int64
```

No null values were found in the dataset

In [194…

```
#Checking for duplicates
df_movie_ratings.duplicated().value_counts()
```

Out[194…

```
False    73856
dtype: int64
```

In [195…

```
#summary statistics
df_movie_ratings.describe()
```

Out[195…

	start_year	averagerating	numvotes
count	73856.000000	73856.000000	7.385600e+04
mean	2014.276132	6.332729	3.523662e+03
std	2.614807	1.474978	3.029402e+04
min	2010.000000	1.000000	5.000000e+00
25%	2012.000000	5.500000	1.400000e+01
50%	2014.000000	6.500000	4.900000e+01
75%	2016.000000	7.400000	2.820000e+02
max	2019.000000	10.000000	1.841066e+06

In [196…

```
#data types
df_movie_ratings.dtypes
```

Out[196…

```
movie_id           object
original_title     object
primary_title      object
start_year          int64
averagerating     float64
numvotes            int64
genres             object
dtype: object
```

There are no duplicates in the dataset

In [197…

```
#Get most popular genres based on number of votes:
pop_genres = ("""SELECT
    mb.genres,
    COUNT(mr.numvotes) AS total_votes,
    AVG(mr.averagerating) AS avg_rating
FROM movie_basics mb
JOIN movie_ratings mr
ON mb.movie_id = mr.movie_id
GROUP BY mb.genres
ORDER BY total_votes DESC;""")
pop_genres_df = pd.read_sql(pop_genres,conn)
```

In [198…

```
#checking for null values
pop_genres_df.isna().sum()
```

Out[198…

```
genres         1
total_votes    0
avg_rating     0
dtype: int64
```

In [199…
```
#Replacing the null value with mode
pop_genres_df['genres'].fillna(pop_genres_df['genres'].mode()[0], inplace=True)
```

In [200…
```
#The null value has been replace by mode
pop_genres_df.isna().sum()
```

Out[200…
```
genres         0
total_votes    0
avg_rating     0
dtype: int64
```

In [201…
```
#checking for duplicates
pop_genres_df.duplicated().value_counts()
```

Out[201…
```
False    924
dtype: int64
```

There are no duplicates on the dataset

In [202…
```
#Summary Statistics
pop_genres_df.describe()
```

Out[202…

	total_votes	avg_rating
count	924.000000	924.000000
mean	79.930736	6.280216
std	569.601986	1.053560
min	1.000000	1.400000
25%	2.000000	5.683482
50%	5.000000	6.300000
75%	29.000000	6.973125
max	11612.000000	9.400000

In [203…
```
# Converting the foreign gross column data to float, removing  string values
df_gross['foreign_gross']= df_gross['foreign_gross'].replace('[,\'NaN'']', '0', regex=T
print(df_gross.dtypes)
df_gross['foreign_gross'].fillna(df_gross['foreign_gross'].std(), inplace=True)
df_gross
```

```
title              object
studio             object
domestic_gross    float64
foreign_gross     float64
year                int64
dtype: object
```

Out[203…

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	6.520000e+08	2010
1	Alice in Wonderland (2010)	BV	334200000.0	6.913000e+08	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	6.643000e+08	2010
3	Inception	WB	292600000.0	5.357000e+08	2010

	title	studio	domestic_gross	foreign_gross	year
4	Shrek Forever After	P/DW	238700000.0	5.139000e+08	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	1.374106e+08	2018
3383	Edward II (2018 re-release)	FM	4800.0	1.374106e+08	2018
3384	El Pacto	Sony	2500.0	1.374106e+08	2018
3385	The Swan	Synergetic	2400.0	1.374106e+08	2018
3386	An Actor Prepares	Grav.	1700.0	1.374106e+08	2018

3387 rows × 5 columns

In [204…

```
#Renaming the column movie to movie_title under the budgets data set
df_budgets.rename(columns={'movie': 'movie_title'}, inplace=True)

#Merging the movie ratings and budgets datasets
matched_df = df_movie_ratings.merge(df_budgets, left_on='original_title', right_on='mov
matched_df
```

Out[204…

	movie_id	original_title	primary_title	start_year	averagerating	numvotes	genr
0	tt0249516	Foodfight!	Foodfight!	2012	1.9	8248	Action,Animation,Come
1	tt0326592	The Overnight	The Overnight	2010	7.5	24	No
2	tt3844362	The Overnight	The Overnight	2015	6.1	14828	Comedy,Myste
3	tt0337692	On the Road	On the Road	2012	6.1	37886	Adventure,Drama,Roman
4	tt4339118	On the Road	On the Road	2014	6.0	6	Drar
...	...	...	...	...	...	...	
2633	tt8662424	Never Again	Never Again	2017	5.7	67	Drar
2634	tt8680254	Richard III	Richard III	2016	9.1	28	Drar
2635	tt8824064	Heroes	Heroes	2019	7.3	7	Documenta
2636	tt8976772	Push	Push	2019	7.3	33	Documenta
2637	tt9024106	Unplanned	Unplanned	2019	6.3	5945	Biography,Drar

2638 rows × 13 columns

In [205…

```
#Checking merged data from column titles
matched_df.columns
```

Out[205…

```
Index(['movie_id', 'original_title', 'primary_title', 'start_year',
       'averagerating', 'numvotes', 'genres', 'id', 'release_date',
```

```
       'movie_title', 'production_budget', 'domestic_gross',
       'worldwide_gross'],
      dtype='object')
```

In [206…

```
#Checking for null values
matched_df.isnull().sum()
```

Out[206…

```
movie_id             0
original_title       0
primary_title        0
start_year           0
averagerating        0
numvotes             0
genres               5
id                   0
release_date         0
movie_title          0
production_budget    0
domestic_gross       0
worldwide_gross      0
dtype: int64
```

No null values found

In [207…

```
#Checking for duplicates
matched_df.duplicated().value_counts()
```

Out[207…

```
False    2638
dtype: int64
```

No duplicates found

In [208…

```
#Summary statistics
matched_df.describe()
```

Out[208…

	start_year	averagerating	numvotes	id
count	2638.000000	2638.000000	2.638000e+03	2638.000000
mean	2013.891205	6.241205	7.162586e+04	50.963230
std	2.554063	1.188941	1.375680e+05	28.458683
min	2010.000000	1.600000	5.000000e+00	1.000000
25%	2012.000000	5.600000	1.922500e+02	27.000000
50%	2014.000000	6.400000	1.335150e+04	51.000000
75%	2016.000000	7.100000	8.458400e+04	76.000000
max	2019.000000	9.300000	1.841066e+06	100.000000

In [209…

```
#Removing string values from the budget and gross columns and converting to float data 
columns_to_clean = ['production_budget', 'domestic_gross', 'worldwide_gross']
for column in columns_to_clean:
    matched_df[column] = matched_df[column].replace({'\\$': '', ',': ''}, regex=True).a
```

In [210…

```
matched_df.dtypes
```

Out[210…

```
movie_id              object
original_title        object
primary_title         object
```

```
start_year              int64
averagerating         float64
numvotes                int64
genres                 object
id                      int64
release_date           object
movie_title            object
production_budget     float64
domestic_gross        float64
worldwide_gross       float64
dtype: object
```

In [211…

```
#viewing the dataset
matched_df
```

Out[211…

	movie_id	original_title	primary_title	start_year	averagerating	numvotes	genr
0	tt0249516	Foodfight!	Foodfight!	2012	1.9	8248	Action,Animation,Come
1	tt0326592	The Overnight	The Overnight	2010	7.5	24	No
2	tt3844362	The Overnight	The Overnight	2015	6.1	14828	Comedy,Myste
3	tt0337692	On the Road	On the Road	2012	6.1	37886	Adventure,Drama,Roman
4	tt4339118	On the Road	On the Road	2014	6.0	6	Drar
...	...	...	...	...	...	...	
2633	tt8662424	Never Again	Never Again	2017	5.7	67	Drar
2634	tt8680254	Richard III	Richard III	2016	9.1	28	Drar
2635	tt8824064	Heroes	Heroes	2019	7.3	7	Documenta
2636	tt8976772	Push	Push	2019	7.3	33	Documenta
2637	tt9024106	Unplanned	Unplanned	2019	6.3	5945	Biography,Drar

2638 rows × 13 columns

In [212…

```
#merging the known for dataset with the budgets dataset
matched_df_known_for = df_movie_ratings_known_for.merge(df_budgets, left_on='original_t
matched_df_known_for
```

Out[212…

	movie_id	original_title	primary_title	start_year	genres	averagerating	numvot
0	tt0249516	Foodfight!	Foodfight!	2012	Action,Animation,Comedy	1.9	82
1	tt0249516	Foodfight!	Foodfight!	2012	Action,Animation,Comedy	1.9	82
2	tt0249516	Foodfight!	Foodfight!	2012	Action,Animation,Comedy	1.9	82
3	tt0326592	The Overnight	The Overnight	2010	None	7.5	

	movie_id	original_title	primary_title	start_year	genres	averagerating	numvot
4	tt0326592	The Overnight	The Overnight	2010	None	7.5	
...	...	...	...	...	...	...	
53591	tt9024106	Unplanned	Unplanned	2019	Biography,Drama	6.3	59
53592	tt9024106	Unplanned	Unplanned	2019	Biography,Drama	6.3	59
53593	tt9024106	Unplanned	Unplanned	2019	Biography,Drama	6.3	59
53594	tt9024106	Unplanned	Unplanned	2019	Biography,Drama	6.3	59
53595	tt9024106	Unplanned	Unplanned	2019	Biography,Drama	6.3	59

53596 rows × 16 columns

In [213...

```
#Checking for duplicates
matched_df_known_for.duplicated().value_counts()
```

Out[213...

```
False    53596
dtype: int64
```

In [214...

```
#Checking for null values
matched_df_known_for.isnull().sum()
```

Out[214...

```
movie_id               0
original_title         0
primary_title          0
start_year             0
genres                21
averagerating          0
numvotes               0
person_id              0
primary_name           0
primary_profession   285
id                     0
release_date           0
movie_title            0
production_budget      0
domestic_gross         0
worldwide_gross        0
dtype: int64
```

In [215...

```
# Dropping the null and missing values, there are number of professions data
matched_df_known_for = matched_df_known_for.dropna().drop_duplicates()
print(matched_df_known_for.isnull().sum())
print(matched_df_known_for.duplicated().value_counts())
matched_df_known_for.columns
```

```
movie_id             0
original_title       0
primary_title        0
start_year           0
genres               0
averagerating        0
numvotes             0
person_id            0
```

```
primary_name          0
primary_profession    0
id                    0
release_date          0
movie_title           0
production_budget     0
domestic_gross        0
worldwide_gross       0
dtype: int64
False    53290
dtype: int64
```

Out[215…

```
Index(['movie_id', 'original_title', 'primary_title', 'start_year', 'genres',
       'averagerating', 'numvotes', 'person_id', 'primary_name',
       'primary_profession', 'id', 'release_date', 'movie_title',
       'production_budget', 'domestic_gross', 'worldwide_gross'],
      dtype='object')
```

In [216…

```
#Removing string values known for dataset and converting to float data type
for column in columns_to_clean:
    matched_df_known_for[column] = matched_df_known_for[column].replace({'\\$': '', ','
```

Feature Engineering

In [217…

```
#Calculating gross earnings
matched_df['gross_earnings'] = matched_df['worldwide_gross']-matched_df['production_bud
matched_df.columns
```

Out[217…

```
Index(['movie_id', 'original_title', 'primary_title', 'start_year',
       'averagerating', 'numvotes', 'genres', 'id', 'release_date',
       'movie_title', 'production_budget', 'domestic_gross', 'worldwide_gross',
       'gross_earnings'],
      dtype='object')
```

In [218…

```
# Saving the cleaned data to the folder
output_path = "Movies_data.csv"
matched_df.to_csv(output_path, index=False)
```

Visualization

Data Modeling

In [219…

```
#Identifying correlation between variables with int or float data types
corr_df = matched_df.select_dtypes(include=['number'])
corr_df
```

Out[219…

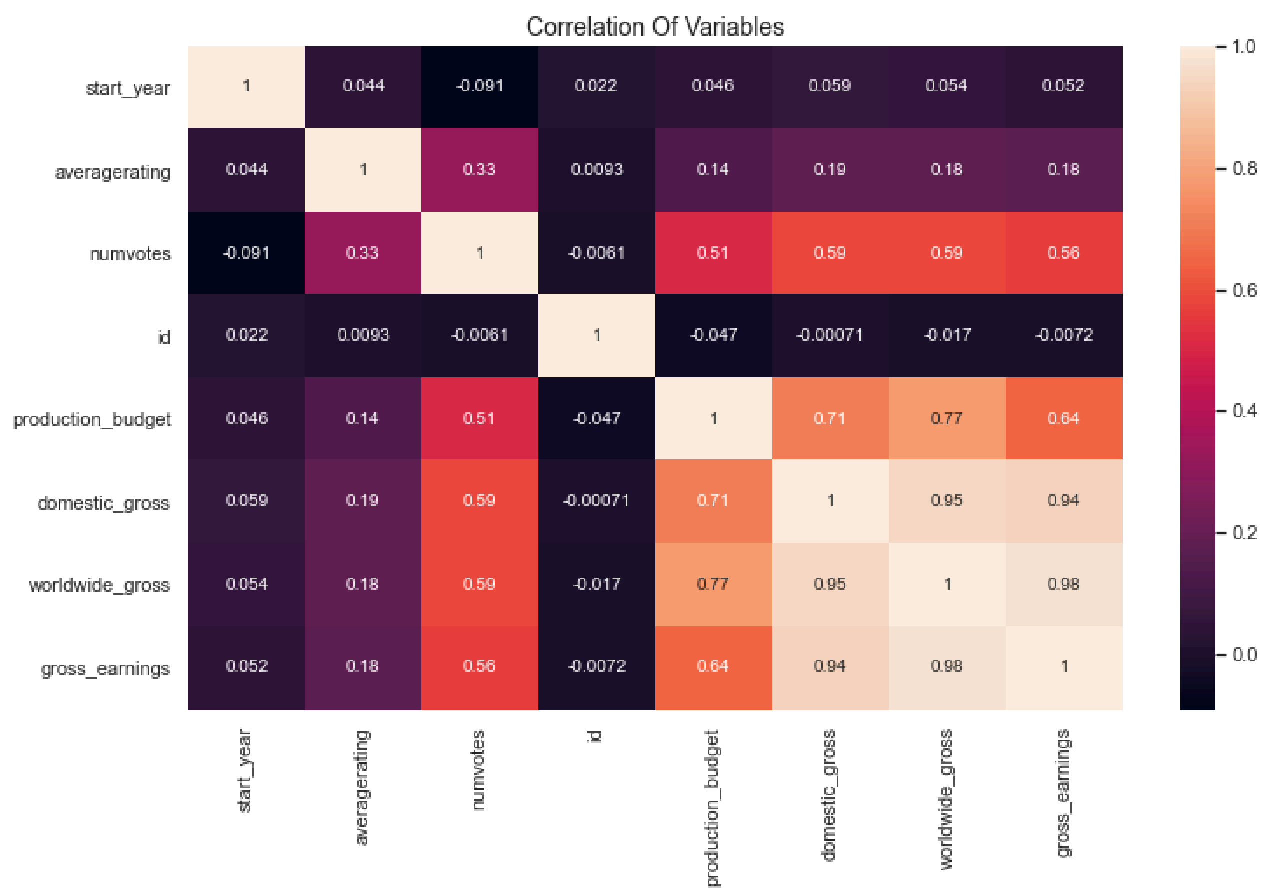
	start_year	averagerating	numvotes	id	production_budget	domestic_gross	worldwide_gross	gr
0	2012	1.9	8248	26	45000000.0	0.0	73706.0	
1	2010	7.5	24	21	200000.0	1109808.0	1165996.0	
2	2015	6.1	14828	21	200000.0	1109808.0	1165996.0	
3	2012	6.1	37886	17	25000000.0	720828.0	9313302.0	
4	2014	6.0	6	17	25000000.0	720828.0	9313302.0	
...	...	...	...	...	...	...	...	
2633	2017	5.7	67	47	500000.0	307631.0	308793.0	

	start_year	averagerating	numvotes	id	production_budget	domestic_gross	worldwide_gross	gr
2634	2016	9.1	28	65	9200000.0	2684904.0	4199334.0	
2635	2019	7.3	7	12	400000.0	655538.0	655538.0	
2636	2019	7.3	33	70	38000000.0	31811527.0	49678401.0	
2637	2019	6.3	5945	33	6000000.0	18107621.0	18107621.0	

2638 rows × 8 columns

In [220…

```
#Correlation plot
fig, ax = plt.subplots(figsize = (12, 7))
ax.set_title('Correlation Of Variables', fontsize=14)

corr = corr_df.corr()

sns.heatmap(corr, annot = True)

plt.savefig('Correlation using Heatmap');

plt.show()
```



Variables with strong positive correlation are: production budget with both worldwide gross (0.77) and domestic gross(0.71) Variables with weak positive correlation are: average rating with both both worldwide gross (0.18) and domestic gross(0.19)

In [221…

```
# Get specific movie genres for accurate analysis(Some movies have more than one genre)
df_budget_genres = matched_df[['genres', 'production_budget', 'worldwide_gross', 'numvo
print(df_budget_genres.duplicated().value_counts())
```

```
False    2636
True        2
dtype: int64
```

In [222…

```
# Use mean to get average between the duplicated rows
df_budget_genres = df_budget_genres.groupby('genres', as_index=False).mean()
print(df_budget_genres.duplicated().value_counts())
df_budget_genres
```

```
False    302
dtype: int64
```

Out[222…

	genres	production_budget	worldwide_gross	numvotes	gross_earnings
0	Action	2.812391e+07	7.058096e+07	2627.173913	4.245705e+07
1	Action,Adventure	4.500000e+06	1.177400e+04	6955.000000	-4.488226e+06
2	Action,Adventure,Animation	1.218125e+08	4.570707e+08	190419.375000	3.352582e+08
3	Action,Adventure,Biography	8.375000e+07	2.254565e+08	236267.250000	1.417065e+08
4	Action,Adventure,Comedy	8.234483e+07	3.379218e+08	208682.344828	2.555770e+08
...	...	...	...	...	...
297	Sci-Fi,Thriller	1.434000e+07	4.364433e+06	10279.666667	-9.975567e+06
298	Sport	1.900000e+07	5.745503e+06	77.000000	-1.325450e+07
299	Thriller	2.718006e+07	6.491041e+07	227.738095	3.773034e+07
300	War	4.000000e+07	3.019910e+07	9.000000	-9.800895e+06
301	Western	2.450000e+06	2.209775e+05	55.500000	-2.229022e+06

302 rows × 5 columns

In [223…

```
#Getting individual genres
df_exploded = df_budget_genres.assign(genres=df_budget_genres['genres'].str.split(','))
df_exploded
```

Out[223…

	genres	production_budget	worldwide_gross	numvotes	gross_earnings
0	Action	2.812391e+07	7.058096e+07	2627.173913	4.245705e+07
1	Action	4.500000e+06	1.177400e+04	6955.000000	-4.488226e+06
1	Adventure	4.500000e+06	1.177400e+04	6955.000000	-4.488226e+06
2	Action	1.218125e+08	4.570707e+08	190419.375000	3.352582e+08
2	Adventure	1.218125e+08	4.570707e+08	190419.375000	3.352582e+08
...	...	...	...	...	...
297	Thriller	1.434000e+07	4.364433e+06	10279.666667	-9.975567e+06
298	Sport	1.900000e+07	5.745503e+06	77.000000	-1.325450e+07

	genres	production_budget	worldwide_gross	numvotes	gross_earnings
299	Thriller	2.718006e+07	6.491041e+07	227.738095	3.773034e+07
300	War	4.000000e+07	3.019910e+07	9.000000	-9.800895e+06
301	Western	2.450000e+06	2.209775e+05	55.500000	-2.229022e+06

795 rows × 5 columns

In [224…

```
#resetting index
df_exploded = df_exploded.reset_index(drop=True)
df_exploded
```

Out[224…

	genres	production_budget	worldwide_gross	numvotes	gross_earnings
0	Action	2.812391e+07	7.058096e+07	2627.173913	4.245705e+07
1	Action	4.500000e+06	1.177400e+04	6955.000000	-4.488226e+06
2	Adventure	4.500000e+06	1.177400e+04	6955.000000	-4.488226e+06
3	Action	1.218125e+08	4.570707e+08	190419.375000	3.352582e+08
4	Adventure	1.218125e+08	4.570707e+08	190419.375000	3.352582e+08
...	...	...	...	...	...
790	Thriller	1.434000e+07	4.364433e+06	10279.666667	-9.975567e+06
791	Sport	1.900000e+07	5.745503e+06	77.000000	-1.325450e+07
792	Thriller	2.718006e+07	6.491041e+07	227.738095	3.773034e+07
793	War	4.000000e+07	3.019910e+07	9.000000	-9.800895e+06
794	Western	2.450000e+06	2.209775e+05	55.500000	-2.229022e+06

795 rows × 5 columns

In [225…

```
#Summary statistics for budget by genre:
budget_stats = df_exploded.groupby('genres')['production_budget'].describe()
budget_stats
```

Out[225…

	count	mean	std	min	25%	50%	75%
genres							
Action	70.0	4.804199e+07	3.577022e+07	500000.0	2.037500e+07	4.200000e+07	6.237500e+07
Adventure	57.0	5.627934e+07	5.217009e+07	500000.0	1.536862e+07	4.000000e+07	8.375000e+07
Animation	14.0	5.522589e+07	4.382572e+07	5000000.0	2.203616e+07	4.303125e+07	7.843750e+07
Biography	32.0	2.743001e+07	2.127658e+07	500000.0	1.485536e+07	2.366474e+07	3.144318e+07
Comedy	75.0	2.765695e+07	2.076534e+07	900000.0	1.220000e+07	2.300000e+07	3.913167e+07
Crime	39.0	2.600301e+07	2.075530e+07	500000.0	9.450714e+06	2.446667e+07	3.950667e+07
Documentary	32.0	1.997831e+07	1.875209e+07	362500.0	5.604167e+06	1.520931e+07	2.719688e+07
Drama	124.0	2.659960e+07	2.556004e+07	500000.0	1.137304e+07	2.139098e+07	3.104167e+07

genres	count	mean	std	min	25%	50%	75%
Family	32.0	3.862525e+07	3.548126e+07	350000.0	8.750000e+06	2.523750e+07	5.831250e+07
Fantasy	42.0	4.351616e+07	4.613736e+07	900000.0	1.223750e+07	2.869375e+07	5.236250e+07
History	17.0	3.143423e+07	2.137567e+07	1500000.0	1.320000e+07	3.000000e+07	4.000000e+07
Horror	43.0	1.616780e+07	1.665795e+07	500000.0	4.875000e+06	1.110000e+07	2.166667e+07
Music	16.0	1.584737e+07	9.179269e+06	2000000.0	1.003750e+07	1.364167e+07	1.866667e+07
Musical	12.0	3.454028e+07	3.800219e+07	500000.0	3.162500e+06	1.808333e+07	6.315000e+07
Mystery	33.0	2.648097e+07	2.472130e+07	500000.0	1.000000e+07	1.753565e+07	3.500000e+07
News	2.0	1.890000e+07	2.248600e+07	3000000.0	1.095000e+07	1.890000e+07	2.685000e+07
Romance	33.0	2.014722e+07	1.555774e+07	25000.0	1.215000e+07	1.701071e+07	2.446667e+07
Sci-Fi	32.0	3.893618e+07	3.701497e+07	350000.0	1.587500e+07	3.007011e+07	4.705000e+07
Sport	19.0	2.856333e+07	3.386070e+07	362500.0	1.001750e+07	1.900000e+07	3.050000e+07
Thriller	48.0	2.303656e+07	2.137556e+07	25000.0	8.075000e+06	1.847545e+07	3.619783e+07
War	13.0	2.487967e+07	2.768036e+07	1500000.0	1.148571e+07	1.750000e+07	2.800000e+07
Western	10.0	3.714500e+07	3.614087e+07	2300000.0	1.115000e+07	3.400000e+07	4.133333e+07

In [226…

```
#Average budget per genre:
avg_budget_per_genre = df_exploded.groupby('genres')['production_budget'].mean().sort_v
avg_budget_per_genre
```

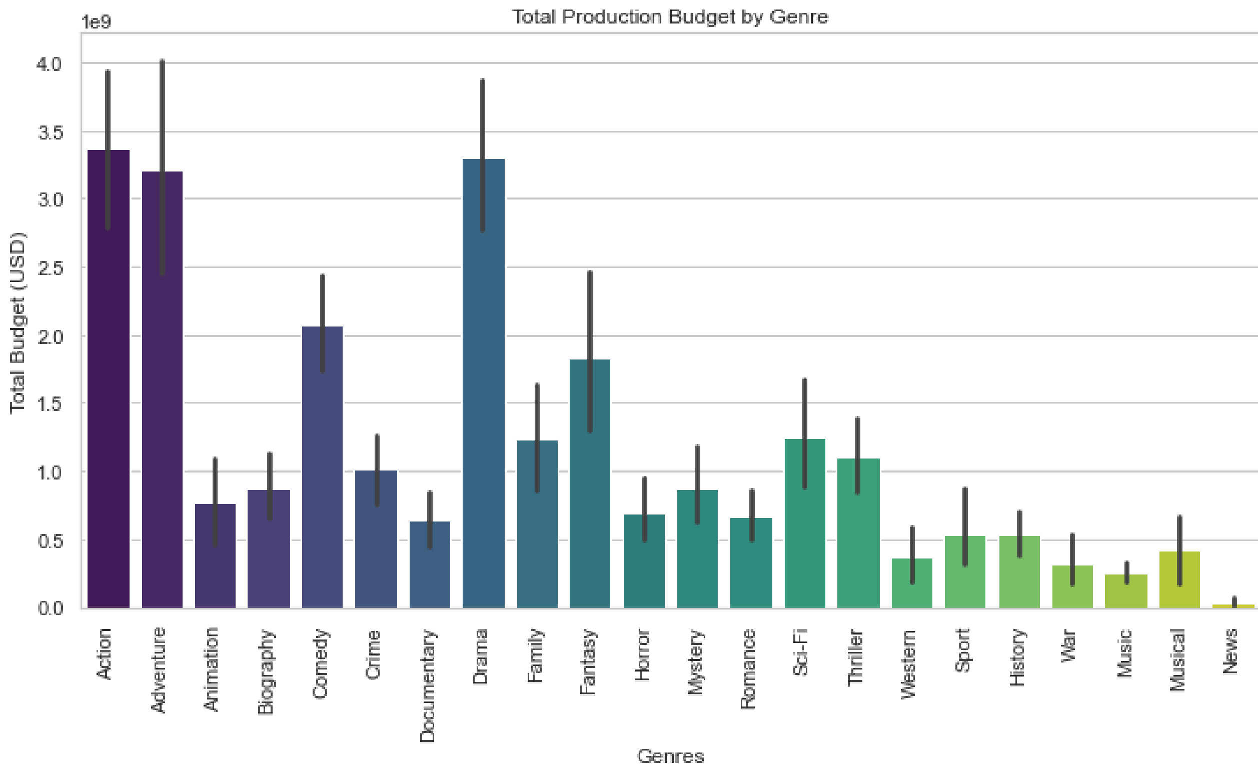
Out[226…

```
genres
Adventure      5.627934e+07
Animation      5.522589e+07
Action         4.804199e+07
Fantasy        4.351616e+07
Sci-Fi         3.893618e+07
Family         3.862525e+07
Western        3.714500e+07
Musical        3.454028e+07
History        3.143423e+07
Sport          2.856333e+07
Comedy         2.765695e+07
Biography      2.743001e+07
Drama          2.659960e+07
Mystery        2.648097e+07
Crime          2.600301e+07
War            2.487967e+07
Thriller       2.303656e+07
Romance        2.014722e+07
Documentary    1.997831e+07
News           1.890000e+07
Horror         1.616780e+07
Music          1.584737e+07
Name: production_budget, dtype: float64
```

In [227…

```
# Calculating the total production_budget per genre and sorting in descending order
df_genre_gross = df_exploded.groupby('genres', as_index=False)['production_budget'].sum
```

```
df_genre_gross = df_genre_gross.sort_values(by='production_budget', ascending=True)

plt.figure(figsize=(12, 6))
sns.barplot(data=df_exploded, x='genres', y='production_budget', estimator=sum, palette
plt.xticks(rotation=90)
plt.title('Total Production Budget by Genre')
plt.ylabel('Total Budget (USD)')
plt.xlabel('Genres')
plt.show()
```



Action, Adventure and Drama have the highest budget

In [228…

```
#Average revenue per genre:
avg_worldwide_gross_per_genre = df_exploded.groupby('genres')['worldwide_gross'].mean()
avg_worldwide_gross_per_genre
```

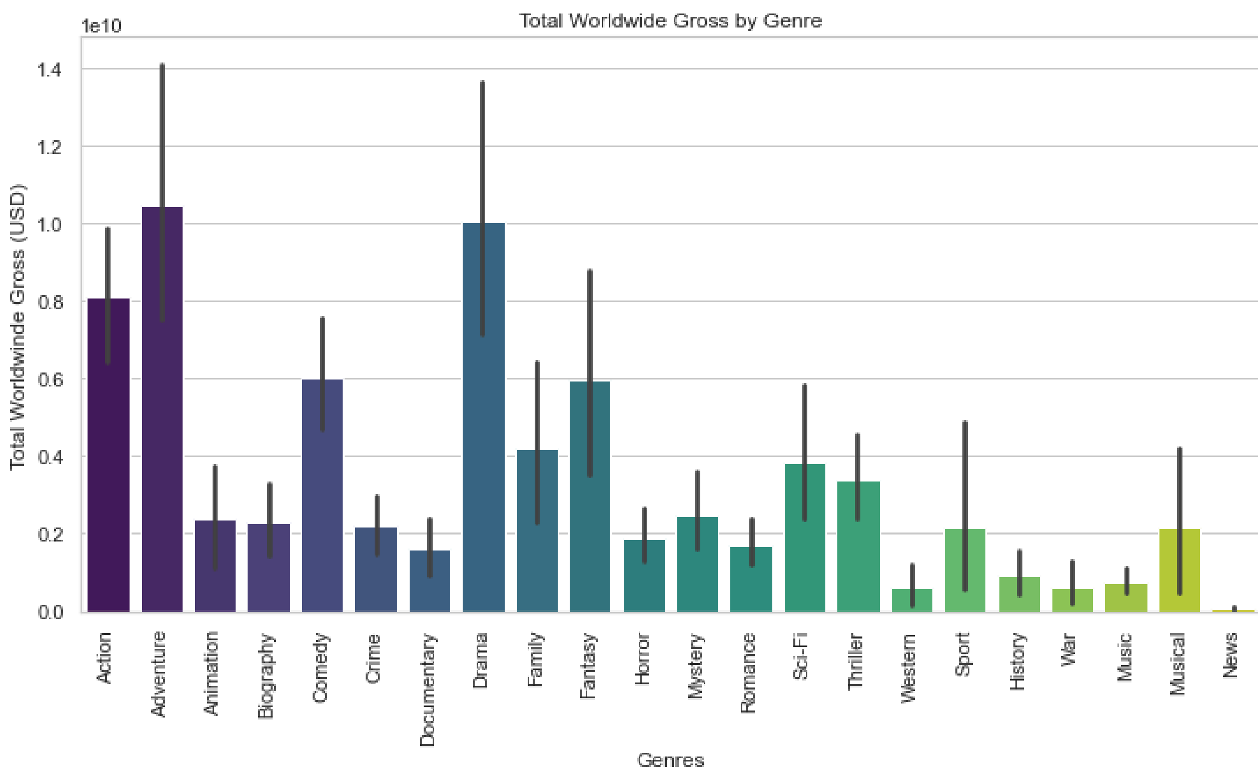
Out[228…

```
genres
Adventure      1.832333e+08
Musical        1.796094e+08
Animation      1.701160e+08
Fantasy        1.420349e+08
Family         1.307622e+08
Sci-Fi         1.199147e+08
Action         1.160846e+08
Sport          1.131311e+08
Drama          8.100754e+07
Comedy         8.047069e+07
Mystery        7.543937e+07
Biography      7.141653e+07
Thriller       7.066527e+07
Western        6.325434e+07
Crime          5.647420e+07
History        5.492947e+07
Romance        5.222557e+07
Documentary    5.105460e+07
Music          4.789611e+07
War            4.685774e+07
```

```
Horror          4.385025e+07
News            3.165783e+07
Name: worldwide_gross, dtype: float64
```

In [229…

```python
# Distribution of revenue across genres:
plt.figure(figsize=(12, 6))
sns.barplot(data=df_exploded, x='genres', y='worldwide_gross', estimator=sum, palette="
plt.xticks(rotation=90)
plt.title('Total Worldwide Gross by Genre')
plt.ylabel('Total Worldwinde Gross (USD)')
plt.xlabel('Genres')
plt.show()
```



Adventure followed by Drama then action have the highest worldwide gross

In [230…

```python
#Retrieving individual genres on the pop_genres dataset
pop_genres_df_exploded = pop_genres_df.assign(genres=pop_genres_df['genres'].str.split(
pop_genres_df_exploded = pop_genres_df_exploded.reset_index(drop=True)
pop_genres_df_exploded
```

Out[230…

	genres	total_votes	avg_rating
0	Drama	11612	6.494265
1	Documentary	10313	7.293794
2	Comedy	5613	5.777998
3	Horror	2692	4.835475
4	Comedy	2617	6.364119
...	...	...	...
2532	Adventure	1	7.600000
2533	Sport	1	7.600000

	genres	total_votes	avg_rating
2534	Action	1	8.700000
2535	Adventure	1	8.700000
2536	Musical	1	8.700000

2537 rows × 3 columns

In [231…

```
#grouping the genres by total votes and average rating
pop_genres_df_exploded = pop_genres_df_exploded.groupby('genres')[['total_votes', 'avg_
pop_genres_df_exploded['total_votes'] = pop_genres_df_exploded['total_votes'].round(0)
pop_genres_df_exploded
```

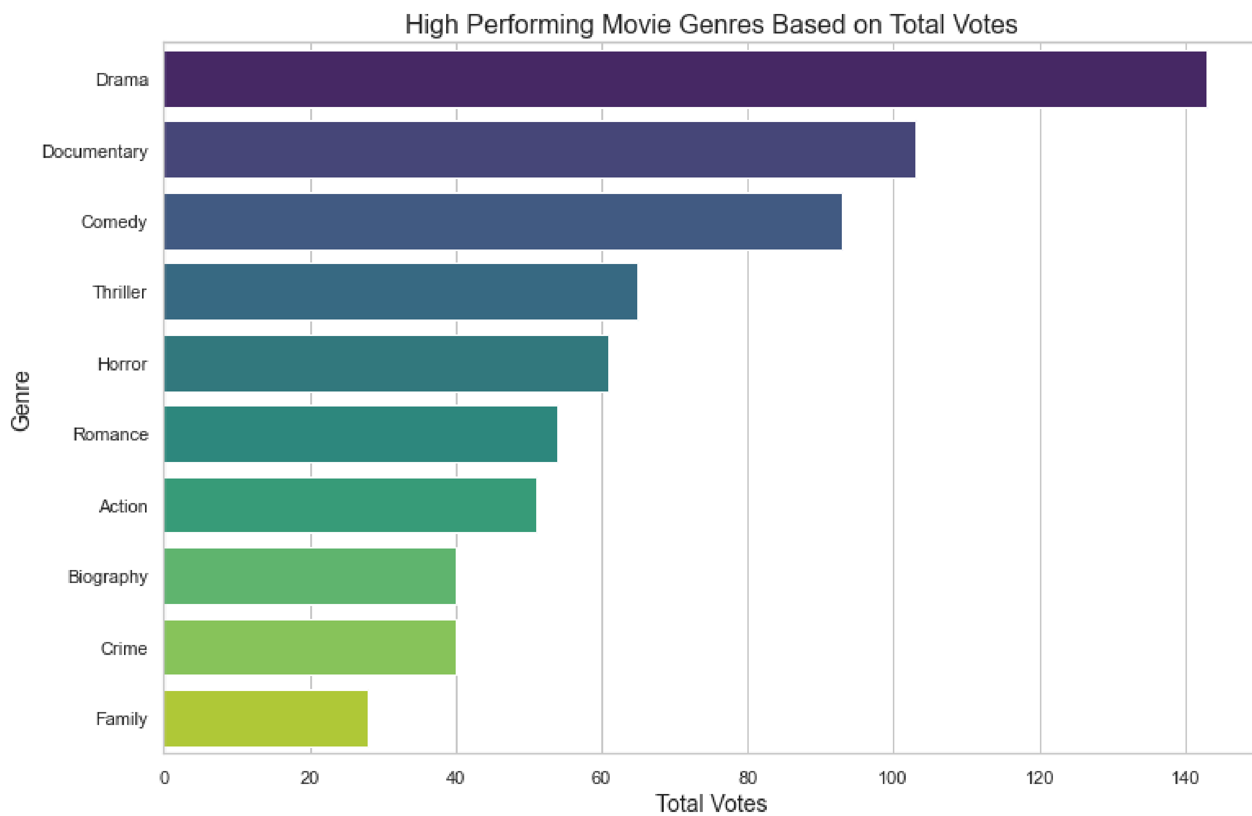
Out[231…

genres	total_votes	avg_rating
Drama	143.0	6.396930
Documentary	103.0	7.279466
Comedy	93.0	6.211247
Thriller	65.0	6.063236
Horror	61.0	5.373191
Romance	54.0	6.189206
Action	51.0	6.019026
Biography	40.0	6.672649
Crime	40.0	6.089916
Family	28.0	6.267387
Mystery	28.0	6.202709
Adventure	27.0	6.174485
History	25.0	6.499445
Music	22.0	6.562830
Sci-Fi	21.0	5.918842
Sport	19.0	6.468112
Fantasy	18.0	6.151740
News	17.0	6.839206
Animation	15.0	6.421499
War	13.0	6.451347
Musical	9.0	6.319958
Western	4.0	6.016386
Reality-TV	2.0	6.768000
Adult	2.0	3.325000

	total_votes	avg_rating
genres		
Game-Show	1.0	7.300000
Short	1.0	8.800000

In [232…

```
#Bar Graph showing high-performing movie genres
plt.figure(figsize=(12, 8))
sns.barplot(x='total_votes', y='genres', data=pop_genres_df_exploded.reset_index().head

plt.title('High Performing Movie Genres Based on Total Votes', fontsize=16)
plt.xlabel('Total Votes', fontsize=14)
plt.ylabel('Genre', fontsize=14)
plt.show()
```

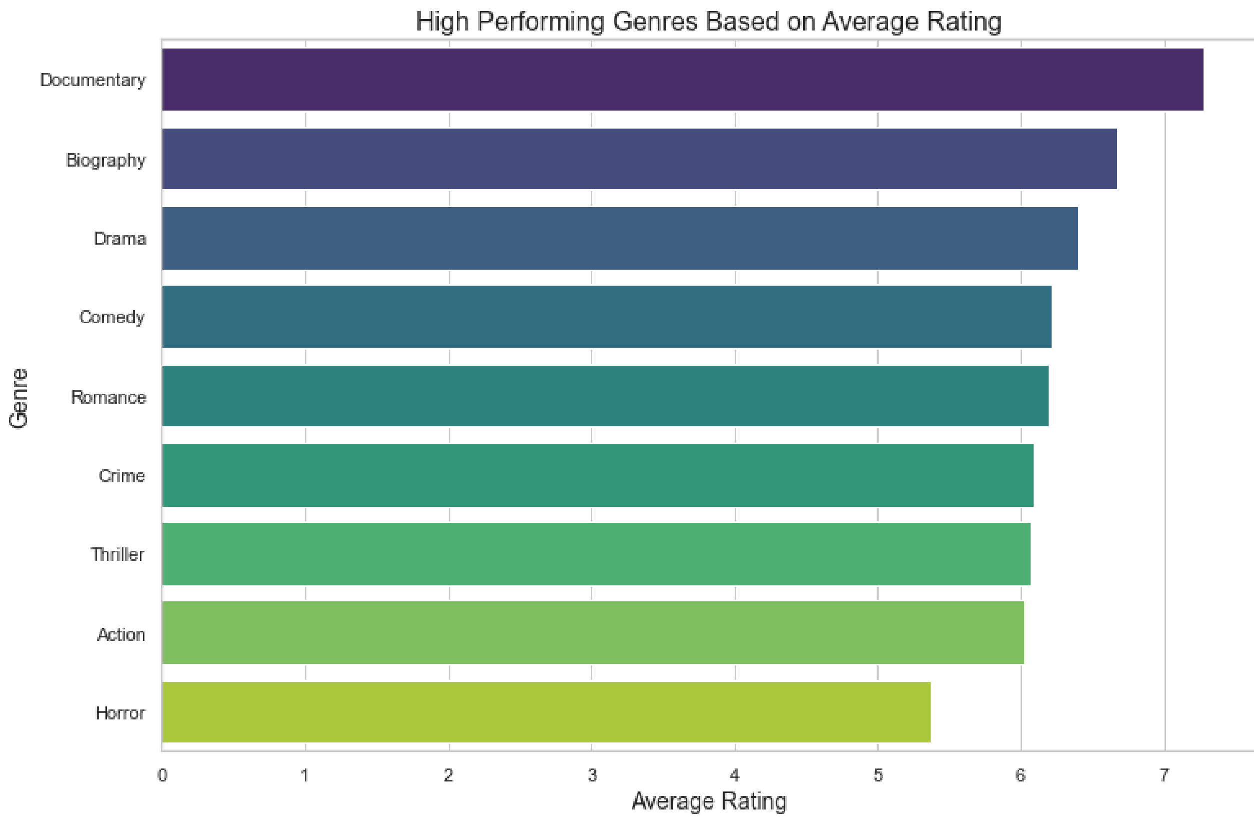


Drama, Documentary and Comedy received the highest number of votes The only correlation between number of votes and revenue is in the Drama category Hence lower correlation between number of votes and revenue

In [233…

```
#Bar Graph showing high-performing movie genres

# Filter the rows where the number of votes is less than the average to avoid cases whe
# had 1 number of vote with a high rating
average_votes = pop_genres_df_exploded['total_votes'].mean()
filtered_df = pop_genres_df_exploded[pop_genres_df_exploded['total_votes'] >= average_v

plt.figure(figsize=(12, 8))
sns.barplot(x='avg_rating', y='genres', data=filtered_df.reset_index().head(10).sort_va


plt.title('High Performing Genres Based on Average Rating', fontsize=16)
```

```
plt.xlabel('Average Rating', fontsize=14)
plt.ylabel('Genre', fontsize=14)
plt.show()
```

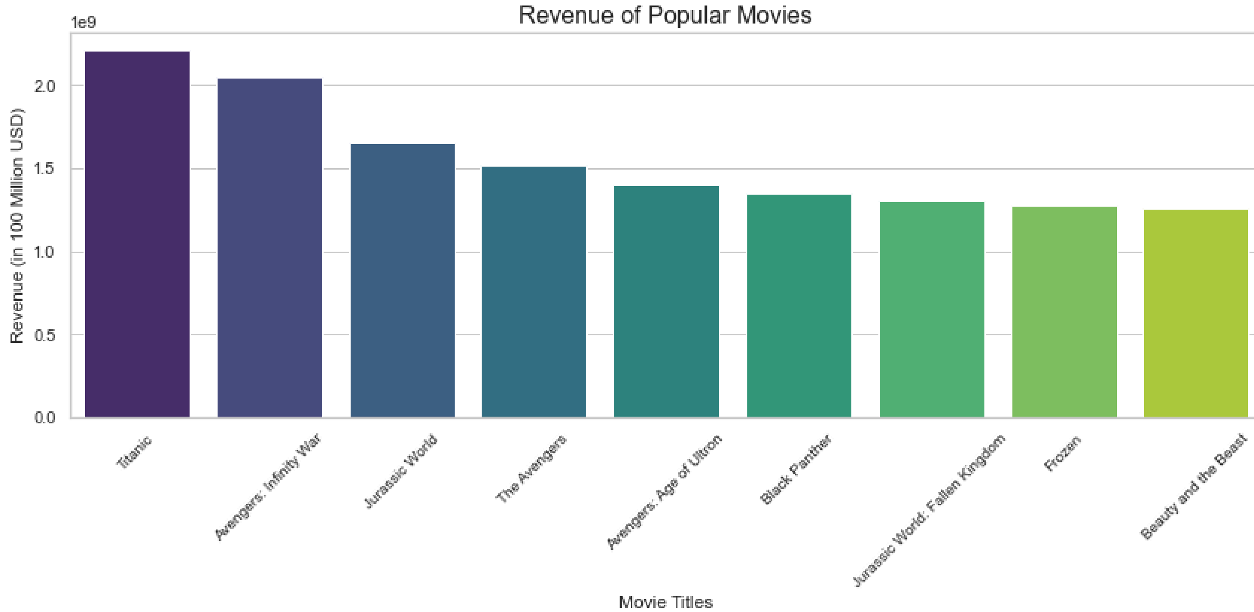


Documentaries, Biographies and Drama have the highest rating based on the total number of votes The Drama category stands out in all 4 variables i.e production budget, revenue, total votes and average rating

Documentary was the highest performing genre followed closely by biography and drama

In [234…

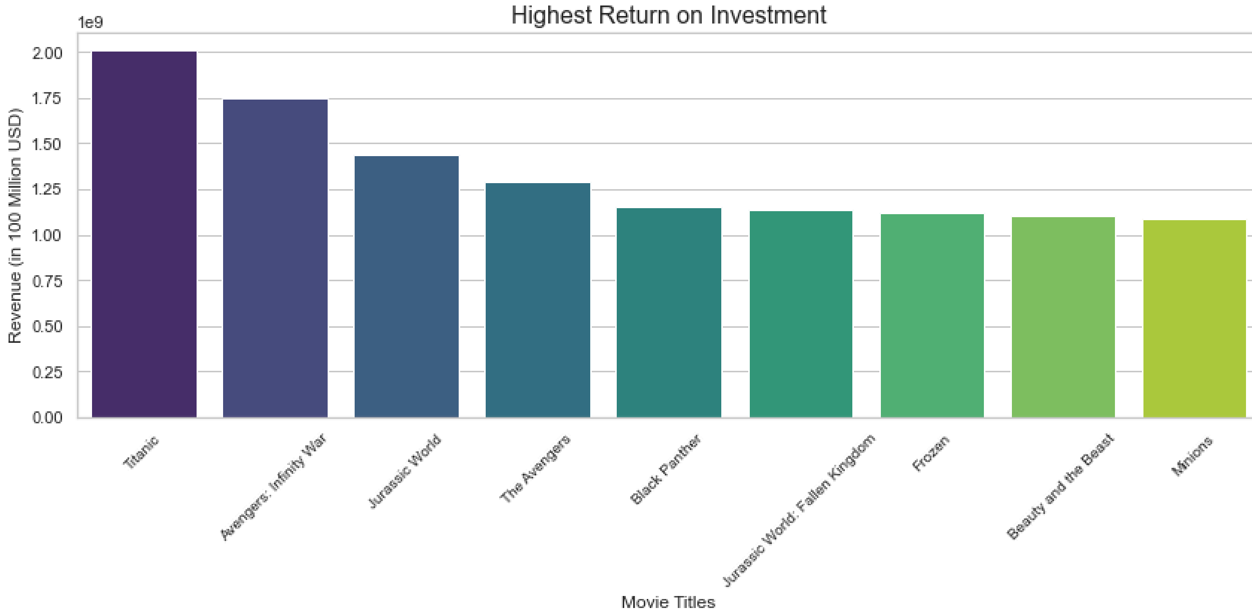
```
#Histogram to determine highest revenues of movie titles
sns.set_theme(style="whitegrid")

top_10_revenues = matched_df.nlargest(10,"worldwide_gross")

top_10_revenues = top_10_revenues.sort_values("worldwide_gross", ascending=False)

plt.figure(figsize=(12, 6))
sns.barplot(x="original_title", y="worldwide_gross", data=top_10_revenues, palette="vir


plt.title("Revenue of Popular Movies", fontsize=16)
plt.xlabel("Movie Titles", fontsize=12)
plt.ylabel("Revenue (in 100 Million USD)", fontsize=12)


plt.xticks(rotation=45, fontsize=10)
plt.tight_layout()
plt.show()
```



There were several titles that generated high revenues with Titanic being the highest

In [235…

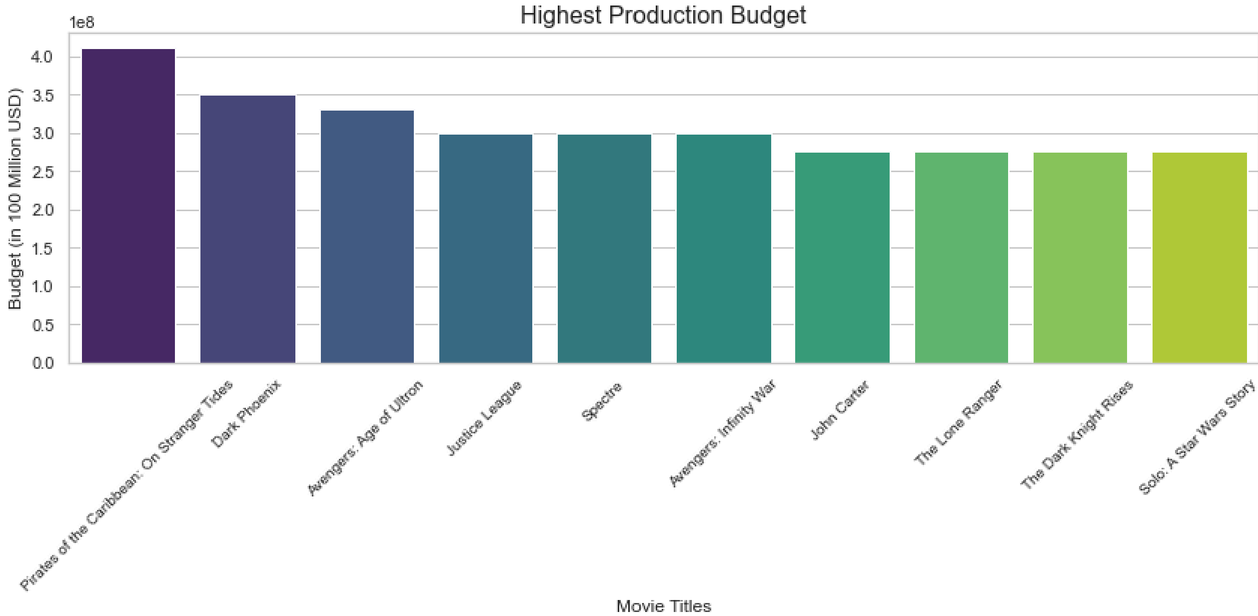
```
#Histogram to determine highest return on investment of movie titles
sns.set_theme(style="whitegrid")

top_10_revenues = matched_df.nlargest(10,"gross_earnings")

top_10_revenues = top_10_revenues.sort_values("gross_earnings", ascending=False)

plt.figure(figsize=(12, 6))
sns.barplot(x="original_title", y="gross_earnings", data=top_10_revenues, palette="viri


plt.title("Highest Return on Investment", fontsize=16)
plt.xlabel("Movie Titles", fontsize=12)
plt.ylabel("Revenue (in 100 Million USD)", fontsize=12)


plt.xticks(rotation=45, fontsize=10)
plt.tight_layout()
plt.show()
```



The highest returns were still the movies with the highest revenue

In [236…

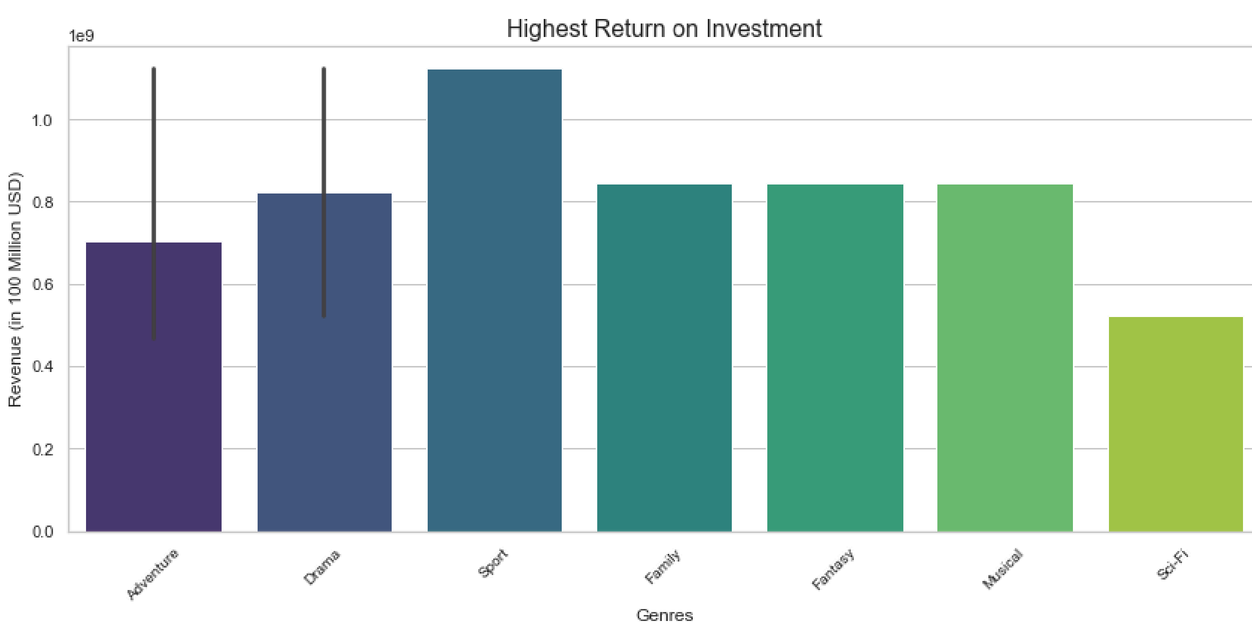
```
#Histogram to determine production budget of movie titles
sns.set_theme(style="whitegrid")

top_10_revenues = matched_df.nlargest(10,"production_budget")

top_10_revenues = top_10_revenues.sort_values("production_budget", ascending=False)

plt.figure(figsize=(12, 6))
sns.barplot(x="original_title", y="production_budget", data=top_10_revenues, palette="v


plt.title("Highest Production Budget", fontsize=16)
plt.xlabel("Movie Titles", fontsize=12)
plt.ylabel("Budget (in 100 Million USD)", fontsize=12)


plt.xticks(rotation=45, fontsize=10)
plt.tight_layout()
plt.show()
```



High production budget do not necessarily equate revenue generation since very few titles appearing on the gross earnings are replicated here

In [237…

```
#Histogram to determine highest return on investment on genres
sns.set_theme(style="whitegrid")
top_10_revenues = df_exploded.nlargest(10,"gross_earnings")
top_10_revenues = top_10_revenues.sort_values("gross_earnings", ascending=False)

plt.figure(figsize=(12, 6))
sns.barplot(x="genres", y="gross_earnings", data=top_10_revenues, palette="viridis")

plt.title("Highest Return on Investment", fontsize=16)
plt.xlabel("Genres", fontsize=12)
plt.ylabel("Revenue (in 100 Million USD)", fontsize=12)

plt.xticks(rotation=45, fontsize=10)
plt.tight_layout()
plt.show()
```



Adventure, Drama and Sport brought the highest return on investment Drama is among the top 3 on return on investment Adventure had high budget, high revenue and also high ROI Sport had low budget and high ROI

In [238…

```
#Finding the mean of the budget and revenue variables
prod_wwwgross_df_exploded = df_exploded.groupby('genres')[['production_budget', 'worldw
prod_wwwgross_df_exploded = prod_wwwgross_df_exploded.reset_index()
prod_wwwgross_df_exploded
```

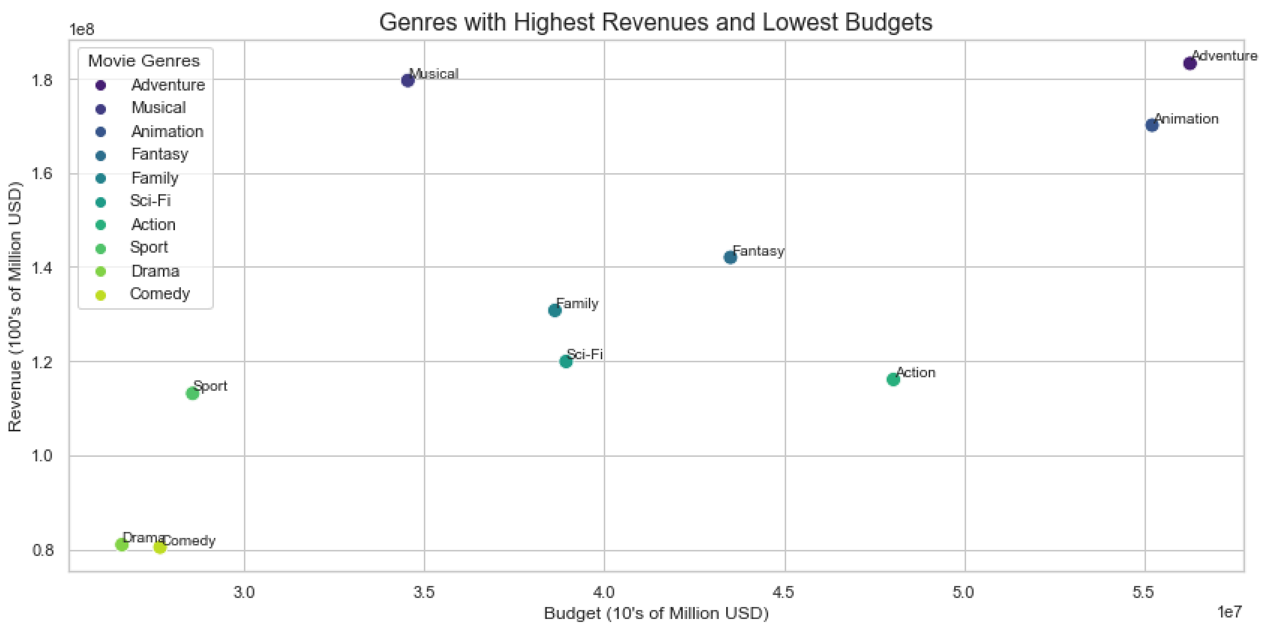
Out[238…

	genres	production_budget	worldwide_gross
0	Action	4.804199e+07	1.160846e+08
1	Adventure	5.627934e+07	1.832333e+08
2	Animation	5.522589e+07	1.701160e+08
3	Biography	2.743001e+07	7.141653e+07
4	Comedy	2.765695e+07	8.047069e+07
5	Crime	2.600301e+07	5.647420e+07
6	Documentary	1.997831e+07	5.105460e+07
7	Drama	2.659960e+07	8.100754e+07
8	Family	3.862525e+07	1.307622e+08
9	Fantasy	4.351616e+07	1.420349e+08
10	History	3.143423e+07	5.492947e+07
11	Horror	1.616780e+07	4.385025e+07
12	Music	1.584737e+07	4.789611e+07
13	Musical	3.454028e+07	1.796094e+08
14	Mystery	2.648097e+07	7.543937e+07
15	News	1.890000e+07	3.165783e+07
16	Romance	2.014722e+07	5.222557e+07
17	Sci-Fi	3.893618e+07	1.199147e+08
18	Sport	2.856333e+07	1.131311e+08
19	Thriller	2.303656e+07	7.066527e+07
20	War	2.487967e+07	4.685774e+07
21	Western	3.714500e+07	6.325434e+07

In [239…

```
# Sorting by 'worldwide_gross' and 'production_budget' for visualization
df_sorted = prod_wwwgross_df_exploded.sort_values(by=["worldwide_gross", "production_bu


plt.figure(figsize=(12, 6))
sns.scatterplot(
    x="production_budget", y="worldwide_gross", hue="genres", data=df_sorted, palette="
)
```

```
for i in range(len(df_sorted)):
    plt.text(
        x=df_sorted["production_budget"].iloc[i] + 0.5,
        y=df_sorted["worldwide_gross"].iloc[i],
        s=df_sorted["genres"].iloc[i],
        fontsize=10,
        ha='left',
        va='bottom'
    )


plt.title("Genres with Highest Revenues and Lowest Budgets", fontsize=16)
plt.xlabel("Budget (10's of Million USD)", fontsize=12)
plt.ylabel("Revenue (100's of Million USD)", fontsize=12)
plt.legend(title="Movie Genres")
plt.tight_layout()

plt.show()
```



Musical has the highest revenue and the lowest budget cost

In [240…

```
#Retrieving individual genres on the known for dataset
df_exploded_proffession = matched_df_known_for.assign(primary_profession=matched_df_kno
df_exploded_proffession = df_exploded_proffession.reset_index(drop=True)
df_exploded_proffession
```

Out[240…

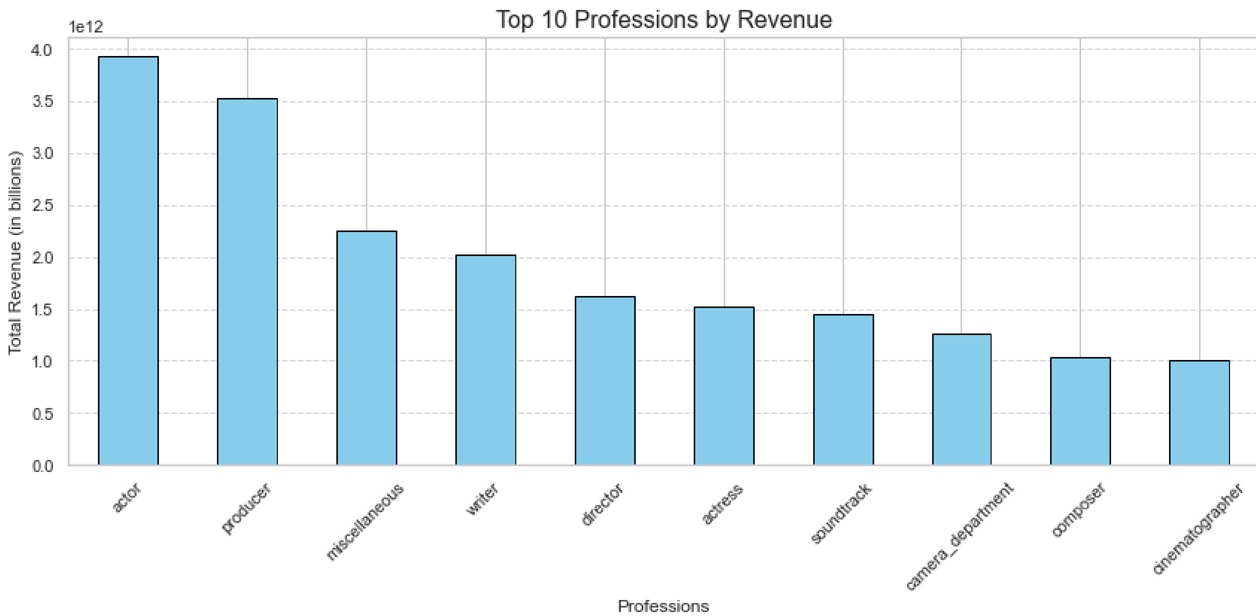
	movie_id	original_title	primary_title	start_year	genres	averagerating	numv
0	tt0249516	Foodfight!	Foodfight!	2012	Action,Animation,Comedy	1.9	8
1	tt0249516	Foodfight!	Foodfight!	2012	Action,Animation,Comedy	1.9	8
2	tt0249516	Foodfight!	Foodfight!	2012	Action,Animation,Comedy	1.9	8
3	tt0249516	Foodfight!	Foodfight!	2012	Action,Animation,Comedy	1.9	8
4	tt0249516	Foodfight!	Foodfight!	2012	Action,Animation,Comedy	1.9	8

	movie_id	original_title	primary_title	start_year	genres	averagerating	numv
...	...	...	...	...	...	...	
132226	tt9024106	Unplanned	Unplanned	2019	Biography,Drama	6.3	5
132227	tt9024106	Unplanned	Unplanned	2019	Biography,Drama	6.3	5
132228	tt9024106	Unplanned	Unplanned	2019	Biography,Drama	6.3	5
132229	tt9024106	Unplanned	Unplanned	2019	Biography,Drama	6.3	5
132230	tt9024106	Unplanned	Unplanned	2019	Biography,Drama	6.3	5

132231 rows × 16 columns

In [241…

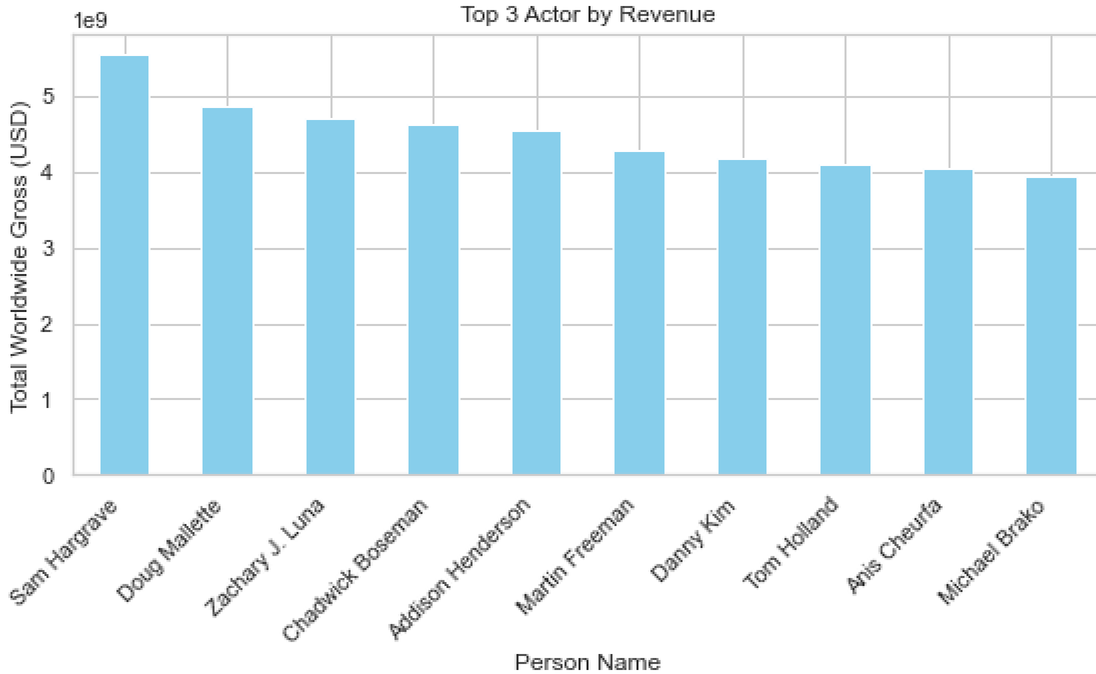
```
#Plotting the top 10 professions by revenue
director_revenue = df_exploded_proffession.groupby('primary_profession')['worldwide_gro


top_10_directors = director_revenue.head(10)


plt.figure(figsize=(12, 6))
top_10_directors.plot(kind='bar', color='skyblue', edgecolor='black')


plt.title('Top 10 Professions by Revenue', fontsize=16)
plt.xlabel('Professions', fontsize=12)
plt.ylabel('Total Revenue (in billions)', fontsize=12)
plt.xticks(rotation=45)
plt.grid(axis='y',
linestyle='--', alpha=0.7)


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

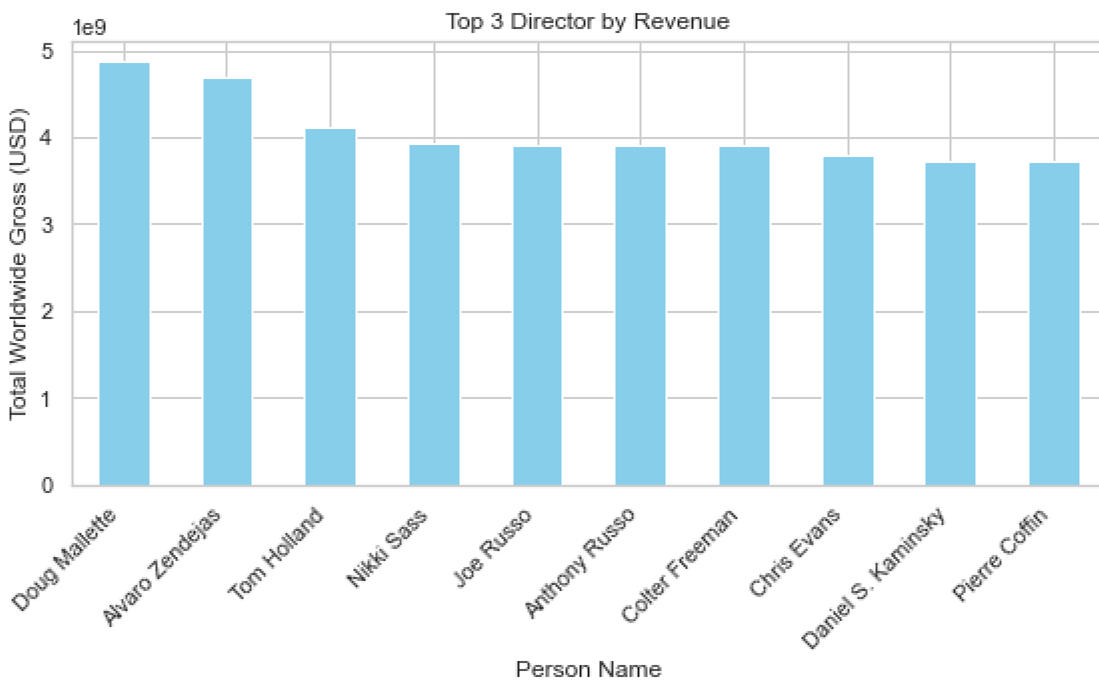


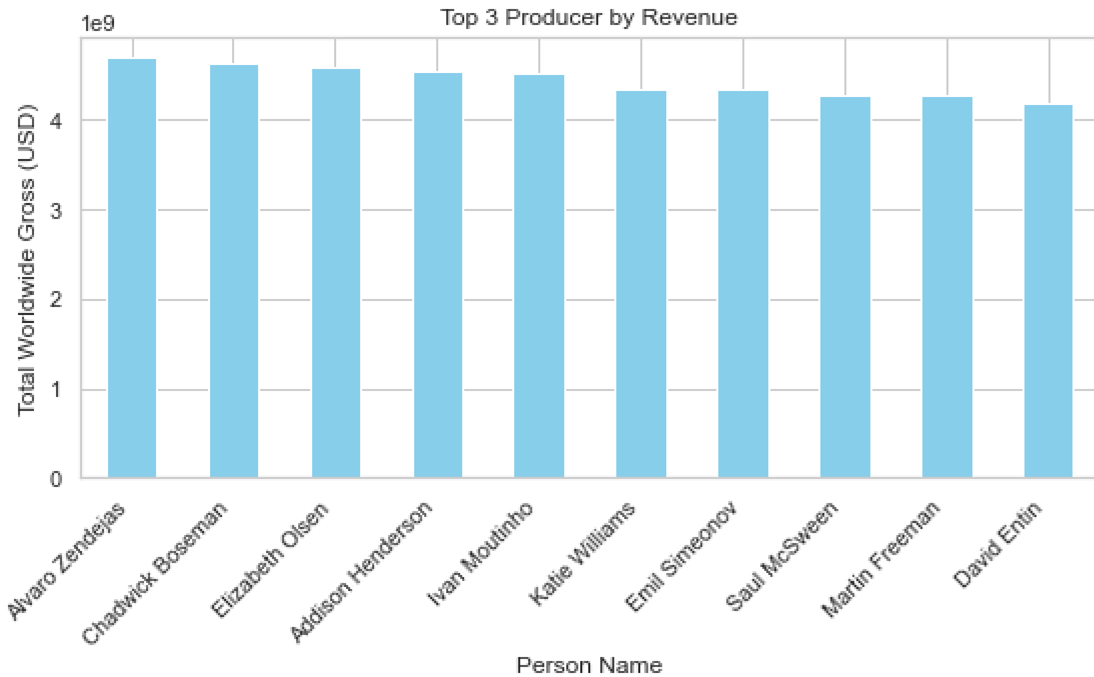
The top performing professions are actor producer and writer

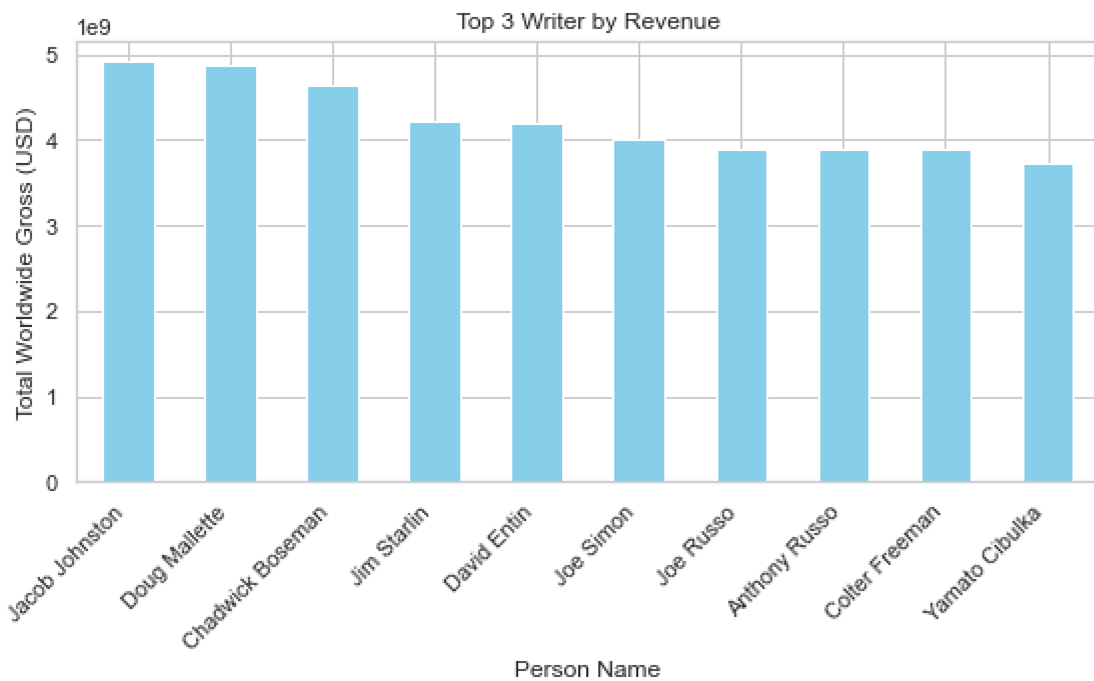
In [242…

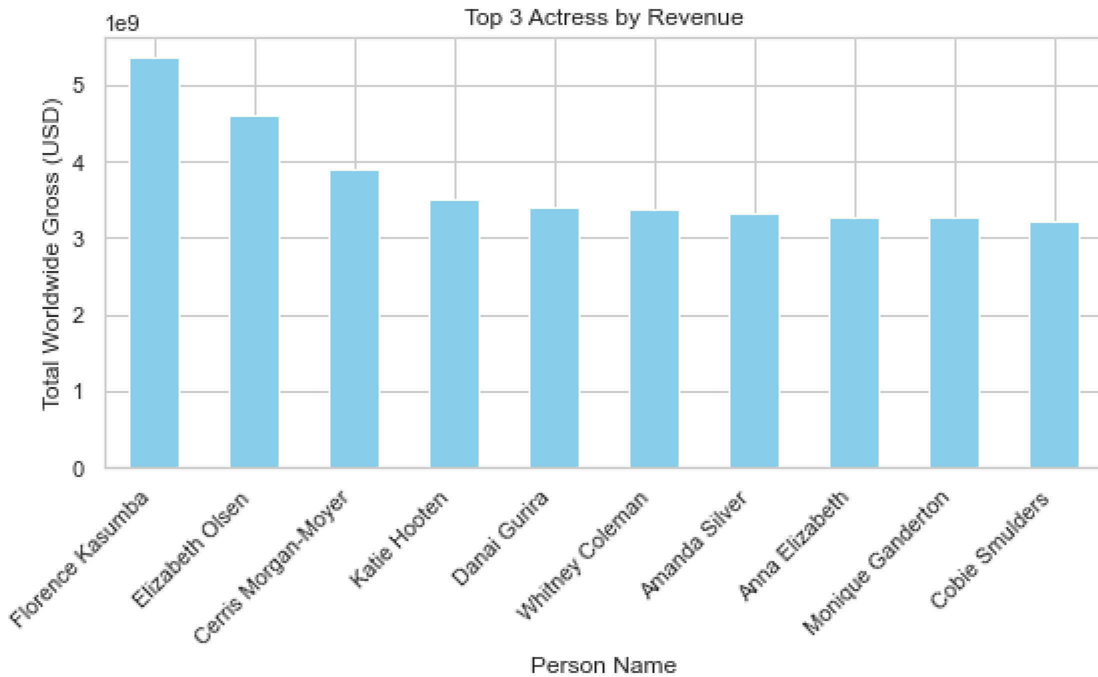
```
#Plotting the top 10 names in the highest performing professions by revenue
categories = ['actor', 'director', 'producer', 'writer', 'actress', 'cinematographer']


for category in categories:
    top_3 = (
        df_exploded_proffession[df_exploded_proffession['primary_profession'] == catego
        .groupby('primary_name')['worldwide_gross']
        .sum()
        .sort_values(ascending=False)
        .head(10)
    )


    plt.figure(figsize=(8, 5))
    top_3.plot(kind='bar', color='skyblue')
    plt.title(f"Top 3 {category.capitalize()} by Revenue")
    plt.ylabel("Total Worldwide Gross (USD)")
    plt.xlabel("Person Name")
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()
```

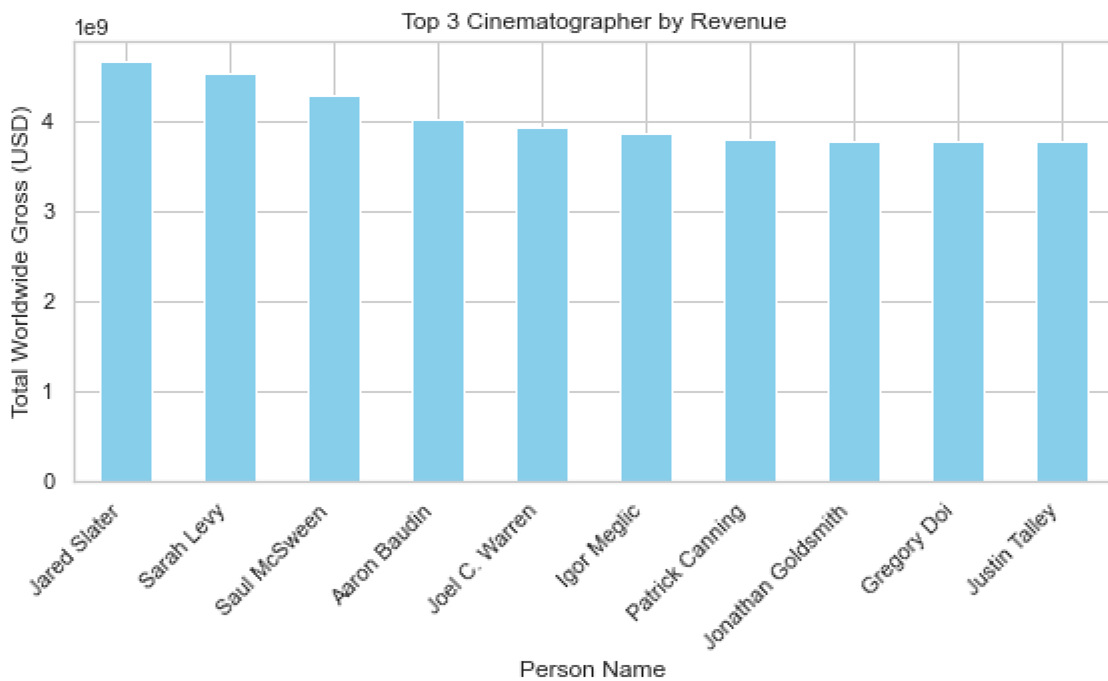

Top 3 Actor by Revenue
1e9
0
1
2
3
4
5
Total Worldwide Gross (USD)
Sam Hargrave
Doug Mallette
Zachary J. Luna
Chadwick Boseman
Addison Henderson
Martin Freeman
Danny Kim
Tom Holland
Anis Cheurfa
Michael Brako
Person Name


Top 3 Director by Revenue
1e9
0
1
2
3
4
5
Total Worldwide Gross (USD)
Doug Mallette
Alvaro Zendejas
Tom Holland
Nikki Sass
Joe Russo
Anthony Russo
Colter Freeman
Chris Evans
Daniel S. Kaminsky
Pierre Coffin
Person Name


Top 3 Producer by Revenue
1e9
Total Worldwide Gross (USD)
0
1
2
3
4
Alvaro Zendejas
Chadwick Boseman
Elizabeth Olsen
Addison Henderson
Ivan Moutinho
Katie Williams
Emil Simeonov
Saul McSween
Martin Freeman
David Entin
Person Name


Top 3 Writer by Revenue
1e9
Total Worldwide Gross (USD)
0
1
2
3
4
5
Jacob Johnston
Doug Mallette
Chadwick Boseman
Jim Starlin
David Entin
Joe Simon
Joe Russo
Anthony Russo
Colter Freeman
Yamato Cibulka
Person Name





The categories selected were based on the top 10 professions. The top 10 names gives a variety of selection from the talent pool

In [243…

```
#Viewing the unique professions
unique_professions = df_exploded_proffession['primary_profession'].value_counts()
unique_professions
```

Out[243…

```
producer                19384
actor                   19198
writer                  11023
miscellaneous           10175
actress                  8824
director                 8585
soundtrack               6916
camera_department        5803
composer                 5222
```

```
cinematographer              4853
music_department             4140
editor                       3711
art_department               2895
editorial_department         2824
visual_effects               2440
assistant_director           2302
stunts                       2011
production_manager           1964
production_designer          1605
sound_department             1557
executive                    1260
art_director                 1110
location_management           757
casting_department            639
animation_department          552
special_effects               427
casting_director              358
set_decorator                 353
make_up_department            335
costume_department            327
transportation_department     199
costume_designer              197
manager                       180
talent_agent                   52
legal                          39
publicist                       9
assistant                       5
Name: primary_profession, dtype: int64
```

Statistical Analysis

Analyzing the Distribution of Movie budgets Across Genres

In [244…

```
#Resetting the index
df_exploded = df_exploded.reset_index(drop=True)
```

In [245…

```
#Summary statistics for budget by genre:
budget_stats = df_exploded.groupby('genres')['production_budget'].describe()
budget_stats
```

Out[245…

genres	count	mean	std	min	25%	50%	75%
Action	70.0	4.804199e+07	3.577022e+07	500000.0	2.037500e+07	4.200000e+07	6.237500e+07
Adventure	57.0	5.627934e+07	5.217009e+07	500000.0	1.536862e+07	4.000000e+07	8.375000e+07
Animation	14.0	5.522589e+07	4.382572e+07	5000000.0	2.203616e+07	4.303125e+07	7.843750e+07
Biography	32.0	2.743001e+07	2.127658e+07	500000.0	1.485536e+07	2.366474e+07	3.144318e+07
Comedy	75.0	2.765695e+07	2.076534e+07	900000.0	1.220000e+07	2.300000e+07	3.913167e+07
Crime	39.0	2.600301e+07	2.075530e+07	500000.0	9.450714e+06	2.446667e+07	3.950667e+07
Documentary	32.0	1.997831e+07	1.875209e+07	362500.0	5.604167e+06	1.520931e+07	2.719688e+07
Drama	124.0	2.659960e+07	2.556004e+07	500000.0	1.137304e+07	2.139098e+07	3.104167e+07
Family	32.0	3.862525e+07	3.548126e+07	350000.0	8.750000e+06	2.523750e+07	5.831250e+07
Fantasy	42.0	4.351616e+07	4.613736e+07	900000.0	1.223750e+07	2.869375e+07	5.236250e+07

	count	mean	std	min	25%	50%	75%
genres							
History	17.0	3.143423e+07	2.137567e+07	1500000.0	1.320000e+07	3.000000e+07	4.000000e+07
Horror	43.0	1.616780e+07	1.665795e+07	500000.0	4.875000e+06	1.110000e+07	2.166667e+07
Music	16.0	1.584737e+07	9.179269e+06	2000000.0	1.003750e+07	1.364167e+07	1.866667e+07
Musical	12.0	3.454028e+07	3.800219e+07	500000.0	3.162500e+06	1.808333e+07	6.315000e+07
Mystery	33.0	2.648097e+07	2.472130e+07	500000.0	1.000000e+07	1.753565e+07	3.500000e+07
News	2.0	1.890000e+07	2.248600e+07	3000000.0	1.095000e+07	1.890000e+07	2.685000e+07
Romance	33.0	2.014722e+07	1.555774e+07	25000.0	1.215000e+07	1.701071e+07	2.446667e+07
Sci-Fi	32.0	3.893618e+07	3.701497e+07	350000.0	1.587500e+07	3.007011e+07	4.705000e+07
Sport	19.0	2.856333e+07	3.386070e+07	362500.0	1.001750e+07	1.900000e+07	3.050000e+07
Thriller	48.0	2.303656e+07	2.137556e+07	25000.0	8.075000e+06	1.847545e+07	3.619783e+07
War	13.0	2.487967e+07	2.768036e+07	1500000.0	1.148571e+07	1.750000e+07	2.800000e+07
Western	10.0	3.714500e+07	3.614087e+07	2300000.0	1.115000e+07	3.400000e+07	4.133333e+07

Linear Regression

In [246…

```
# Defining the variables
X=matched_df[['production_budget','numvotes']]
y=matched_df['worldwide_gross']
```

In [247…

```
# Adding constant
model = sm.OLS(y, sm.add_constant(X))
model
```

Out[247…

```
<statsmodels.regression.linear_model.OLS at 0x1d48c58c6d0>
```

In [248…

```
#Fitting the model
results=model.fit()
results
```

Out[248…

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1d44f74e760>
```

In [249…

```
#Evaluation
results.fvalue, results.f_pvalue
```

Out[249…

```
(2456.429178831795, 0.0)
```

In [250…

```
#Checking the goodness of the fit
results.rsquared
```

Out[250…

```
0.6508943497429841
```

The R-squared value of 0.6508 means the model explains about 65% of the variation in revenue using production budget and number of votes, other factors or adjustments might improve the fit.

In [251…

```
#Constants
results.params
```

Out[251…

```
const              -1.498054e+07
production_budget   2.643436e+00
numvotes            3.744110e+02
dtype: float64
```

Intercept (const): This represents the predicted revenue when both the production budget and number of votes are zero.

Production Budget Coefficient: For every additional dollar increase in the production budget, the revenue increases by $2.64 on average, holding other factors constant.This shows a strong positive relationship between production budgets and revenue.

Number of Votes Coefficient: For every 1-unit increase in the number of votes, the predicted revenue increases by $374.4 , holding other factors constant.This indicates that movies with more votes tend to generate significantly higher revenue.

In [252…

```
#Confidence interval
print(results.conf_int())
```

```
                              0             1
const             -2.053522e+07 -9.425863e+06
production_budget  2.535836e+00  2.751035e+00
numvotes           3.368893e+02  4.119327e+02
```

In [253…

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

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:        worldwide_gross   R-squared:                       0.651
Model:                            OLS   Adj. R-squared:                  0.651
Method:                 Least Squares   F-statistic:                     2456.
Date:                Sun, 26 Jan 2025   Prob (F-statistic):               0.00
Time:                        14:51:18   Log-Likelihood:                -52741.
No. Observations:                2638   AIC:                         1.055e+05
Df Residuals:                    2635   BIC:                         1.055e+05
Df Model:                           2                                         
Covariance Type:            nonrobust                                         
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const             -1.498e+07   2.83e+06     -5.288      0.000   -2.05e+07   -9.43e+06
production_budget     2.6434      0.055     48.173      0.000       2.536       2.751
numvotes            374.4110     19.135     19.567      0.000     336.889     411.933
==============================================================================
Omnibus:                     2050.898   Durbin-Watson:                   1.749
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           110923.794
Skew:                           3.200   Prob(JB):                         0.00
Kurtosis:                      34.116   Cond. No.                     7.39e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The condition number is large, 7.39e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Summary

Model fit based on data

From the linear regression, the model explains about 65% of the variation in revenue using production budget and number of votes, other factors or adjustments might improve the fit.

Significance: Since none of the confidence intervals include 0, all three coefficients (intercept, production_budget, and numvotes) are statistically significant at the 95% confidence level.

Practical Implications: The production budget has a clear and precise positive effect on revenue. For every dollar spent, revenue increases by a consistent multiplier.

The number of votes is also a significant factor, with a large impact per additional vote, which might reflect audience engagement or popularity translating to revenue.

if the dataset is limited (e.g., only includes movies from a specific region or time period), generalization may be limited.

Model Summary

We are confident with our model if subjected to new data, at a level of 65%.

Evaluation

Movie genres consistently achieve the highest ratings and high ROI

The genres that consistently achieve the highest ratings and high ROI; Drama cut across both highest rating and high return on investment

Projected revenue and return on investment (ROI) across different Movie genres for strategic decision making

Adventure, Drama and Sport gave the highets return on investment. Sport should be considered due to its low budget, Drama cuts across audiences as well as return, while Adventure will deliver the highest return but also with a high budget. Musicals were also noted to have the lowest budget with the highest revenue

Roles contributing to the success of high performing movies and movie genres

The major roles were actors, producers,writers, directors and actresses

Insights derived from top-performing movie genres to inform Rilsoft movie studio production

1. The production budget, has a significant effect on revenue. Therefore for higher returns high budget is required.
2. The Drama category cuts across both the audiences as well as high return
3. The talents selected play a key role in success of the movies

4. Not all high budgets result in high revenues as seen in the case of Pirates of the Caribbean movie
5. The total number of votes per genre has a direct correlation with the revenue. More votes, more revenue.

Conclusions

Recommendations for the business

The following genres are recommended;

Adventure, Sports and Drama.

Sport should be considered due to its low budget, Drama cuts across audiences as well as return, while Adventure will deliver the highest return but also with a high budget.

The following professions are recommended;

actors, producers,writers, directors and actresses with the top names being;

Actors; Sam Hargrave, Doug Mallette, Zachary J. Luna, Chadwick Boseman

Producers; Alvaro Zendejas, Chadwick Boseman, Elizabeth Olsen

Writers; Jacob Johnston, Doug Mallette, Chadwick Boseman

Directors; Doug Mallette, Alvaro Zendejas, Tom Holland

Actresses; Florence Kasumba, Elizabeth Olsen, Cerris Morgan-Moyer

Reasons why the analysis might not fully solve the business problem

There were some columns in datasets that had multiple null values hence could not be used to retrieve insights

The 65% fit of the model may affect accuracy

Variance in the data sets that were merged resulted in some loss of data

Future input to improve the project

To provide marketing insights, we would need to have domestic gross per country.

Continued analysis and adaptation to market trends to stay ahead of industry shifts

Geting insights based on demographics e.g. age, gender etc.

Include comments from critics