PHASE 2 PROJECT

Film Making Entry Analysis

- **Business Understanding**

1. The Problem

The company is establishing a new movie studio to expand into original content production. An analysis of current box office trends is underway to identify high-performing film genres and translate these findings into strategic creative direction.

2. Primary Goal

Determine what types of films are currently performing best at the box office to inform strategic decisions for our new movie studio.

Key Questions:

- What budget ranges are most profitable?
- What are the characteristics of successful films in recent years?
- **Data Understanding**

1. **Datasets Chosen**

- *tmdb.movies.csv.gz*
- *tn.movie_budgets.csv.gz*

1. **Data Description**

TMDB Dataset

- Records: ~26,000+ movies
- Key Variables: title, release_date, popularity, vote_average
- Time Period: 1960-2019

TN Dataset

- Records: ~5,000+ movies
- Key Variables: movie, production_budget, domestic_gross, worldwide_gross
- Time Period: 2010-2018
- **Data Preparation**

1. Importing Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import gzip
```

2. Loading the Data

2.1 Decompressing the tmdb file

```
gzfile_path = 'tmdb.movies.csv.gz'
output_name = gzfile_path.replace('.gz', '')

with gzip.open(gzfile_path, 'rt') as gz_file:
    with open(output_name, 'w') as csv_file:
        csv_file.write(gz_file.read())
```

2.2 Loading the decompressed TMDB file

```
# Veiwing the data
tmdb_data = pd.read_csv("tmdb.movies.csv")
tmdb_data.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          34   2010-11-
19
1                    How to Train Your Dragon          29   2010-03-
26
2                                  Iron Man 2          29   2010-05-
07
3                                   Toy Story          28   1995-11-
22
4                                   Inception          28   2010-07-
16

                                        title  vote_average
vote_count
0  Harry Potter and the Deathly Hallows: Part 1             8
10788
1                    How to Train Your Dragon             8
7610
2                                  Iron Man 2             7
12368
3                                   Toy Story             8
10174
4                                   Inception             8
22186
```

2.3 Decompressing the tn file

```
gzfile_path = 'tn.movie_budgets.csv.gz'
output_name = gzfile_path.replace('.gz', '')

with gzip.open(gzfile_path, 'rt') as gz_file:
    with open(output_name, 'w') as csv_file:
        csv_file.write(gz_file.read())
```

2.4 Loading the decompressed tn file

```
tn_data= pd.read_csv("tn.movie_budgets.csv")
tn_data.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
```

2.5 Checking for issues on both tables

```
# Check data types
tmdb_data.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
```

```
tn_data.info()
```

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

```
# Basic info
tmdb_data.describe()
```

	Unnamed: 0	id	popularity	vote_average	vote_count
count	26,517	26,517	26,517	26,517	26,517
mean	13,258	295,050	3	6	194
std	7,655	153,662	4	2	961
min	0	27	1	0	1
25%	6,629	157,851	1	5	2
50%	13,258	309,581	1	6	5
75%	19,887	419,542	4	7	28
max	26,516	608,444	81	10	22,186

```
tn_data.describe()
```

	id
count	5,782
mean	50
std	29
min	1
25%	25
50%	50
75%	75
max	100

```
# Check for missing values first
tmdb_data.isnull().sum()
```

```
Unnamed: 0           0
genre_ids            0
id                   0
original_language    0
original_title       0
popularity           0
release_date         0
```

```
title                0
vote_average         0
vote_count           0
dtype: int64
```

```
tn_data.isnull().sum()
```

```
id                   0
release_date         0
movie                0
production_budget    0
domestic_gross       0
worldwide_gross      0
dtype: int64
```

- **Data Analysis**

1. Cleaning tmdb

```
# First check of data shape before Cleaning
tmdb_data.shape
```

```
(26517, 10)
```

```
# Dropping Columns that are not needed
tmdb_data = tmdb_data.drop(columns=['Unnamed: 0', 'genre_ids',
'release_date', 'title', 'original_title'])
tmdb_data
```

```
           id original_language  popularity  vote_average  vote_count
0       12444                en          34             8       10788
1       10191                en          29             8        7610
2       10138                en          29             7       12368
3         862                en          28             8       10174
4       27205                en          28             8       22186
...       ...               ...         ...           ...         ...
26512  488143                en           1             0           1
26513  485975                en           1             0           1
26514  381231                en           1             0           1
26515  366854                en           1             0           1
26516  309885                en           1             0           1

[26517 rows x 5 columns]
```

```
# Convert to whole number
tmdb_data['vote_count'] = tmdb_data['vote_count'].astype(int)

# Convert to decimal
tmdb_data['vote_average'] = tmdb_data['vote_average'].round(1)

# Convert popularity to whole numbers
```

```
tmdb_data['popularity'] = tmdb_data['popularity'].astype(int)

print(tmdb_data['vote_count'])
print(tmdb_data['vote_average'])
print(tmdb_data['popularity'])
```

```
0        10788
1         7610
2        12368
3        10174
4        22186
         ...
26512        1
26513        1
26514        1
26515        1
26516        1
Name: vote_count, Length: 26517, dtype: int32
0       8
1       8
2       7
3       8
4       8
        ..
26512   0
26513   0
26514   0
26515   0
26516   0
Name: vote_average, Length: 26517, dtype: float64
0        33
1        28
2        28
3        28
4        27
         ..
26512     0
26513     0
26514     0
26515     0
26516     0
Name: popularity, Length: 26517, dtype: int32
```

```
# Remove duplicates
tmdb_data = tmdb_data.drop_duplicates(subset=['id'])

# Remove rows with missing critical information
tmdb_data = tmdb_data.dropna(subset=['id', 'original_language',
'popularity', 'vote_average', 'vote_count'])
tmdb_data
```

```
           id original_language  popularity  vote_average  vote_count
0       12444                en          33             8       10788
1       10191                en          28             8        7610
2       10138                en          28             7       12368
3         862                en          28             8       10174
4       27205                en          27             8       22186
...       ...               ...         ...           ...         ...
26512  488143                en           0             0           1
26513  485975                en           0             0           1
26514  381231                en           0             0           1
26515  366854                en           0             0           1
26516  309885                en           0             0           1

[25497 rows x 5 columns]
```

```
# Remove movies with 0 vote_count or vote_average
tmdb_data = tmdb_data[tmdb_data['vote_count'] > 0]
tmdb_data = tmdb_data[tmdb_data['vote_average'] > 0]
tmdb_data
```

```
           id original_language  popularity  vote_average  vote_count
0       12444                en          33             8       10788
1       10191                en          28             8        7610
2       10138                en          28             7       12368
3         862                en          28             8       10174
4       27205                en          27             8       22186
...       ...               ...         ...           ...         ...
26499  589607                en           0             1           1
26500  561932                en           0             1           1
26501  558305                en           0             1           1
26502  551756                en           0             1           1
26503  543481                de           0             1           1

[25367 rows x 5 columns]
```

```
# Sort by release date
tmdb_data = tmdb_data.sort_values('popularity', ascending=False)
tmdb_data
```

```
           id original_language  popularity  vote_average  vote_count
23811  299536                en          80             8       13948
11019  245891                en          78             7       10081
23812  324857                en          60             8        4048
11020  122917                en          53             7        8392
5179    24428                en          50             8       19673
...       ...               ...         ...           ...         ...
12950  300695                en           0             4           6
12951  343801                en           0             7           4
12952  306220                en           0             6           5
12953  336893                en           0             8           6
```

```
26503  543481                 de           0             1           1

[25367 rows x 5 columns]
```

```
# Final check of data shape
tmdb_data.shape
```

```
(25367, 5)
```

1. Cleaning tmdb

```
# First check of data shape before Cleaning
tn_data.shape
```

```
(5782, 6)
```

```
# Renaming the movie column
tn_data = tn_data.rename(columns={'movie': 'title'})
tn_data.head()
```

```
   id  release_date                                        title  \
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
```

```
# Drop rows where missing (these are likely invalid entries)
tn_data = tn_data.dropna(subset=['title', 'id', 'release_date',
'production_budget', 'domestic_gross', 'worldwide_gross'])
tn_data.head()
```

```
   id  release_date                                        title  \
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
```

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

# Change to desired format 'YYYY-MM-DD'
tn_data['release_date'] = tn_data['release_date'].dt.strftime('%Y-%m-
%d')
tn_data['release_date']
```

```
0       2009-12-18
1       2011-05-20
2       2019-06-07
3       2015-05-01
4       2017-12-15
           ...
5777    2018-12-31
5778    1999-04-02
5779    2005-07-13
5780    2015-09-29
5781    2005-08-05
Name: release_date, Length: 5782, dtype: object
```

```
# Convert each money column to float
tn_data['production_budget'] =
tn_data['production_budget'].str.replace('$', '').str.replace(',',
'').astype(float)
tn_data['domestic_gross'] = tn_data['domestic_gross'].str.replace('$',
'').str.replace(',', '').astype(float)
tn_data['worldwide_gross'] =
tn_data['worldwide_gross'].str.replace('$', '').str.replace(',',
'').astype(float)

# Display with commas as thousands separators
pd.set_option('display.float_format', '{:,.0f}'.format)
tn_data.head()
```

```
   id release_date                                        title  \
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

   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
```

```
# Reset index after cleaning
tn_data = tn_data.reset_index(drop=True)
tn_data
```

```
      id release_date                                          title  \
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
...   ..          ...                                          ...
5777  78   2018-12-31                                     Red 11
5778  79   1999-04-02                                  Following
5779  80   2005-07-13              Return to the Land of Wonders
5780  81   2015-09-29                       A Plague So Pleasant
5781  82   2005-08-05                          My Date With Drew

      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
...                 ...             ...              ...
5777              7,000               0                0
5778              6,000          48,482          240,495
5779              5,000           1,338            1,338
5780              1,400               0                0
5781              1,100         181,041          181,041

[5782 rows x 6 columns]
```

```
# Final check of data shape
tn_data.shape
```

```
(5782, 6)
```

3. Merging The Data

```
# Merge the two datasets using the 'id' column
merged_data = tn_data.merge(tmdb_data, on='id', how='inner')
merged_data.head(10)
```

```
   id release_date                                    title
production_budget  \
0  27   2012-05-04                          The Avengers
225,000,000
```

```
1  27   2010-05-21                    Shrek Forever After
165,000,000
2  27   2015-12-25                         The Revenant
135,000,000
3  27   2004-12-17                          The Aviator
110,000,000
4  27   2002-12-18  The Lord of the Rings: The Two Towers
94,000,000
5  27   2000-07-28         Nutty Professor II: The Klumps
84,000,000
6  27   1996-07-02                       Independence Day
75,000,000
7  27   1998-11-06                             The Siege
70,000,000
8  27   2011-04-01                                   Hop
63,000,000
9  27   2000-11-03             The Legend of Bagger Vance
60,000,000
```

	domestic_gross	worldwide_gross	original_language	popularity	\
0	623,279,547	1,517,935,897	en	10	
1	238,736,787	756,244,673	en	10	
2	183,637,894	532,938,302	en	10	
3	102,608,827	208,370,892	en	10	
4	342,548,984	934,699,645	en	10	
5	123,307,945	166,307,945	en	10	
6	306,169,255	817,400,878	en	10	
7	40,934,175	116,625,798	en	10	
8	108,085,305	188,657,593	en	10	
9	30,695,227	39,235,486	en	10	

	vote_average	vote_count
0	5	170
1	5	170
2	5	170
3	5	170
4	5	170
5	5	170
6	5	170
7	5	170
8	5	170
9	5	170

4.Cleaning Merged Data

```
# Before cleaning
merged_data.shape
```

```
(344, 10)
```

```
# Check for and handle missing values/remove rows with NaN values
merged_data.isnull().sum()
merged_data = merged_data.dropna()
merged_data
```

```
     id release_date                                    title  \
0    27   2012-05-04                            The Avengers
1    27   2010-05-21                     Shrek Forever After
2    27   2015-12-25                            The Revenant
3    27   2004-12-17                             The Aviator
4    27   2002-12-18  The Lord of the Rings: The Two Towers
..   ..          ...                                     ...
339  97   2015-05-15                               Childless
340  97   1933-04-07                               King Kong
341  97   2006-08-11         Conversations with Other Women
342  97   2014-04-11                            Jesus People
343  97   2015-03-25                             Open Secret

     production_budget  domestic_gross  worldwide_gross
original_language  \
0          225,000,000     623,279,547    1,517,935,897
en
1          165,000,000     238,736,787      756,244,673
en
2          135,000,000     183,637,894      532,938,302
en
3          110,000,000     102,608,827      208,370,892
en
4           94,000,000     342,548,984      934,699,645
en
..                 ...             ...              ...
...
339         1,000,000           1,036            1,036
en
340           672,000      10,000,000       10,000,650
en
341           450,000         379,418        1,297,745
en
342           250,000               0                0
en
343           100,000               0                0
en

     popularity  vote_average  vote_count
0            10             5         170
1            10             5         170
2            10             5         170
3            10             5         170
4            10             5         170
..          ...           ...         ...
```

```
339          12              7        1100
340          12              7        1100
341          12              7        1100
342          12              7        1100
343          12              7        1100

[344 rows x 10 columns]
```

```
# Remove title duplicates
merged_data.drop_duplicates(subset=['title'], inplace=True)

# Handle text data
merged_data['title'] = merged_data['title'].str.strip()
merged_data['original_language'] =
merged_data['original_language'].str.lower()
merged_data['title'], merged_data['original_language']
```

```
(0                                The Avengers
 1                      Shrek Forever After
 2                                The Revenant
 3                                 The Aviator
 4      The Lord of the Rings: The Two Towers
                        ...
 339                                 Childless
 340                                 King Kong
 341           Conversations with Other Women
 342                               Jesus People
 343                                Open Secret
 Name: title, Length: 343, dtype: object,
 0      en
 1      en
 2      en
 3      en
 4      en
        ..
 339    en
 340    en
 341    en
 342    en
 343    en
 Name: original_language, Length: 343, dtype: object)
```

```
# Convert release_date to datetime
merged_data['release_date'] =
pd.to_datetime(merged_data['release_date'])
merged_data['release_date']
```

```
0     2012-05-04
1     2010-05-21
2     2015-12-25
```

```
3     2004-12-17
4     2002-12-18
         ...
339   2015-05-15
340   1933-04-07
341   2006-08-11
342   2014-04-11
343   2015-03-25
Name: release_date, Length: 343, dtype: datetime64[ns]
```

```
# Create calculated fields for better analysis

# The profit
merged_data['profit'] = merged_data['worldwide_gross'] -
merged_data['production_budget']

# The return on investment
merged_data['roi'] = merged_data['profit'] /
merged_data['production_budget']

merged_data.head(10)
```

```
   id release_date                                    title
production_budget  \
0  27   2012-05-04                          The Avengers
225,000,000
1  27   2010-05-21                   Shrek Forever After
165,000,000
2  27   2015-12-25                          The Revenant
135,000,000
3  27   2004-12-17                           The Aviator
110,000,000
4  27   2002-12-18  The Lord of the Rings: The Two Towers
94,000,000
5  27   2000-07-28         Nutty Professor II: The Klumps
84,000,000
6  27   1996-07-02                       Independence Day
75,000,000
7  27   1998-11-06                             The Siege
70,000,000
8  27   2011-04-01                                   Hop
63,000,000
9  27   2000-11-03             The Legend of Bagger Vance
60,000,000

   domestic_gross  worldwide_gross original_language  popularity  \
0     623,279,547    1,517,935,897                en          10
1     238,736,787      756,244,673                en          10
2     183,637,894      532,938,302                en          10
3     102,608,827      208,370,892                en          10
```

```
4     342,548,984      934,699,645                en           10
5     123,307,945      166,307,945                en           10
6     306,169,255      817,400,878                en           10
7      40,934,175      116,625,798                en           10
8     108,085,305      188,657,593                en           10
9      30,695,227       39,235,486                en           10
```

```
   vote_average  vote_count        profit  roi
0             5         170 1,292,935,897    6
1             5         170   591,244,673    4
2             5         170   397,938,302    3
3             5         170    98,370,892    1
4             5         170   840,699,645    9
5             5         170    82,307,945    1
6             5         170   742,400,878   10
7             5         170    46,625,798    1
8             5         170   125,657,593    2
9             5         170   -20,764,514   -0
```

```
# After Cleaning
merged_data.shape
```

```
(343, 12)
```

5. Saving Merged Data

```
merged_data.to_csv('Merged.csv', index=False)
print("Cleaned data saved!")
```

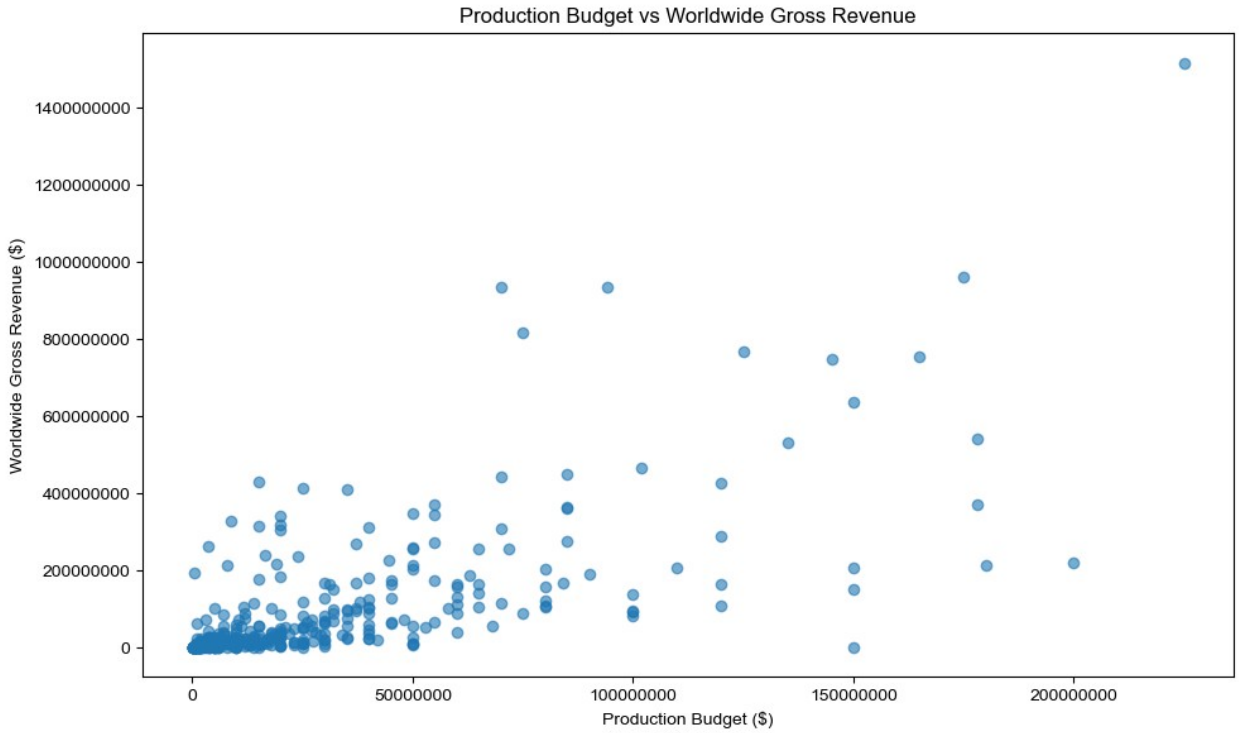
```
Cleaned data saved!
```

- **Modeling**

Visualizations

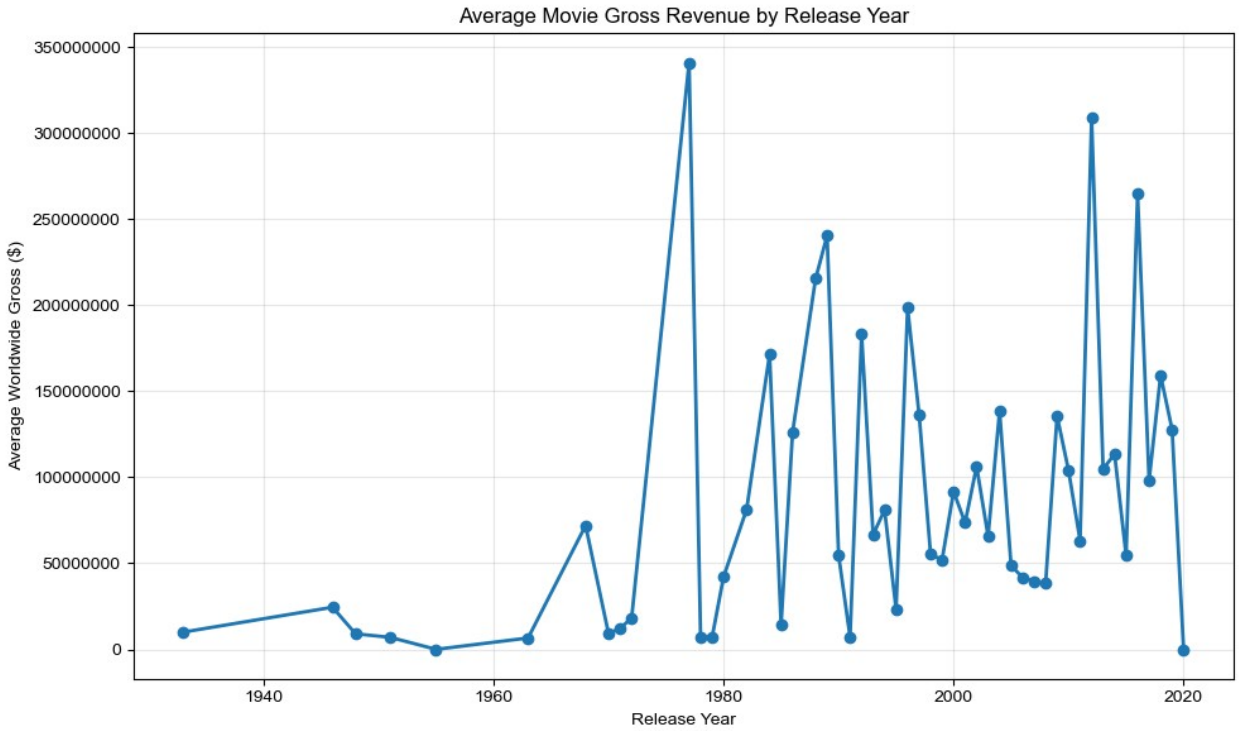
1. Production Budget vs Worldwide Gross Revenue

```
plt.figure(figsize=(10, 6))
plt.scatter(merged_data['production_budget'],
merged_data['worldwide_gross'], alpha=0.6)
plt.xlabel('Production Budget ($)')
plt.ylabel('Worldwide Gross Revenue ($)')
plt.title('Production Budget vs Worldwide Gross Revenue')
plt.ticklabel_format(style='plain')
plt.tight_layout()
plt.show()
```



2. Average Movie Gross Revenue by Release Year

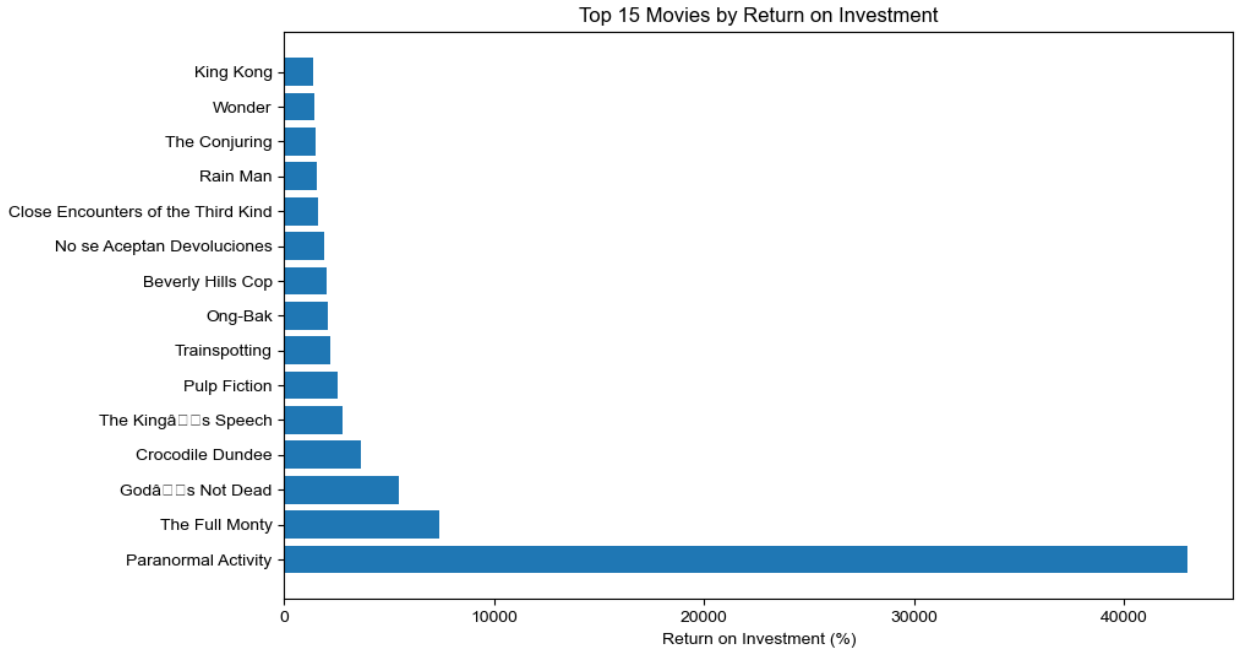
```
plt.figure(figsize=(10, 6))
merged_data['release_year'] =
pd.to_datetime(merged_data['release_date']).dt.year
yearly_avg = merged_data.groupby('release_year')
['worldwide_gross'].mean()
plt.plot(yearly_avg.index, yearly_avg.values, marker='o', linewidth=2)
plt.xlabel('Release Year')
plt.ylabel('Average Worldwide Gross ($)')
plt.title('Average Movie Gross Revenue by Release Year')
plt.grid(True, alpha=0.3)
plt.ticklabel_format(style='plain', axis='y')
plt.tight_layout()
plt.show()
```



3. Top 15 Movies by Return on Investment

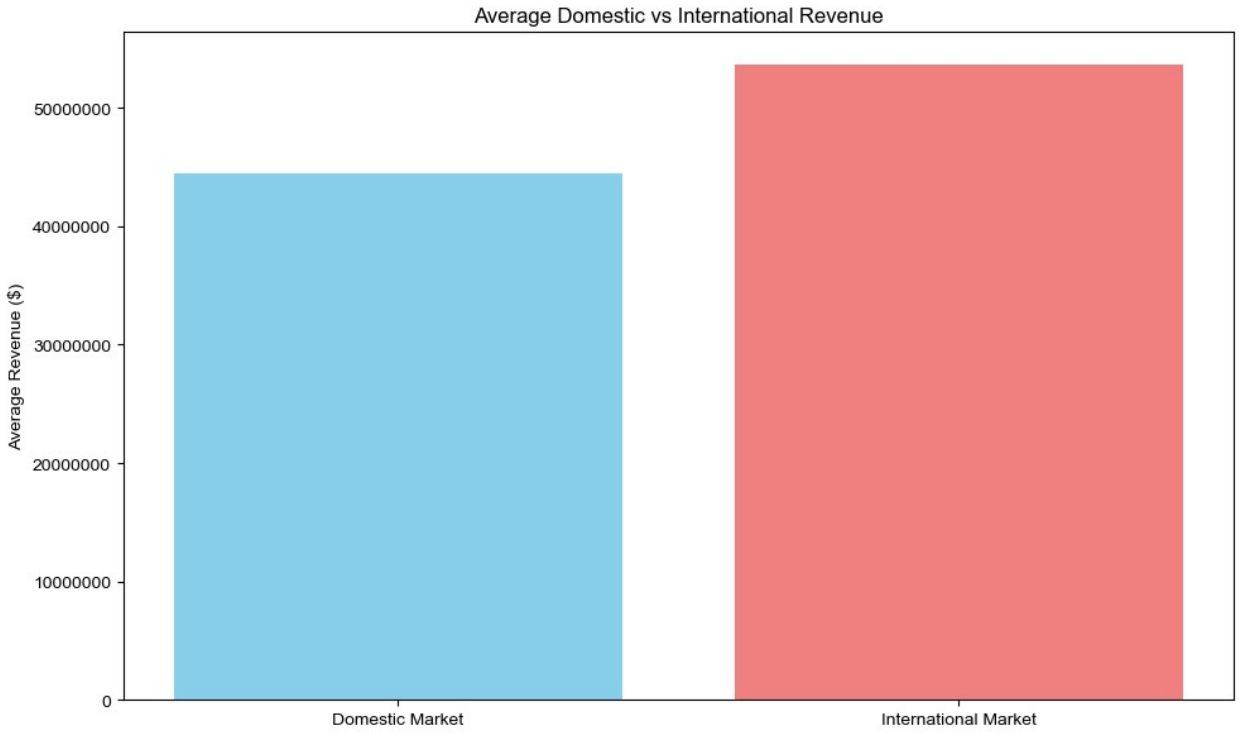
```
plt.figure(figsize=(10, 6))
merged_data['roi'] = (merged_data['worldwide_gross'] -
merged_data['production_budget']) / merged_data['production_budget'] *
100
top_roi = merged_data.nlargest(15, 'roi')

plt.barh(range(len(top_roi)), top_roi['roi'])
plt.yticks(range(len(top_roi)), top_roi['title'])
plt.xlabel('Return on Investment (%)')
plt.title('Top 15 Movies by Return on Investment')
plt.show()
```



4. Average Domestic vs International Revenue

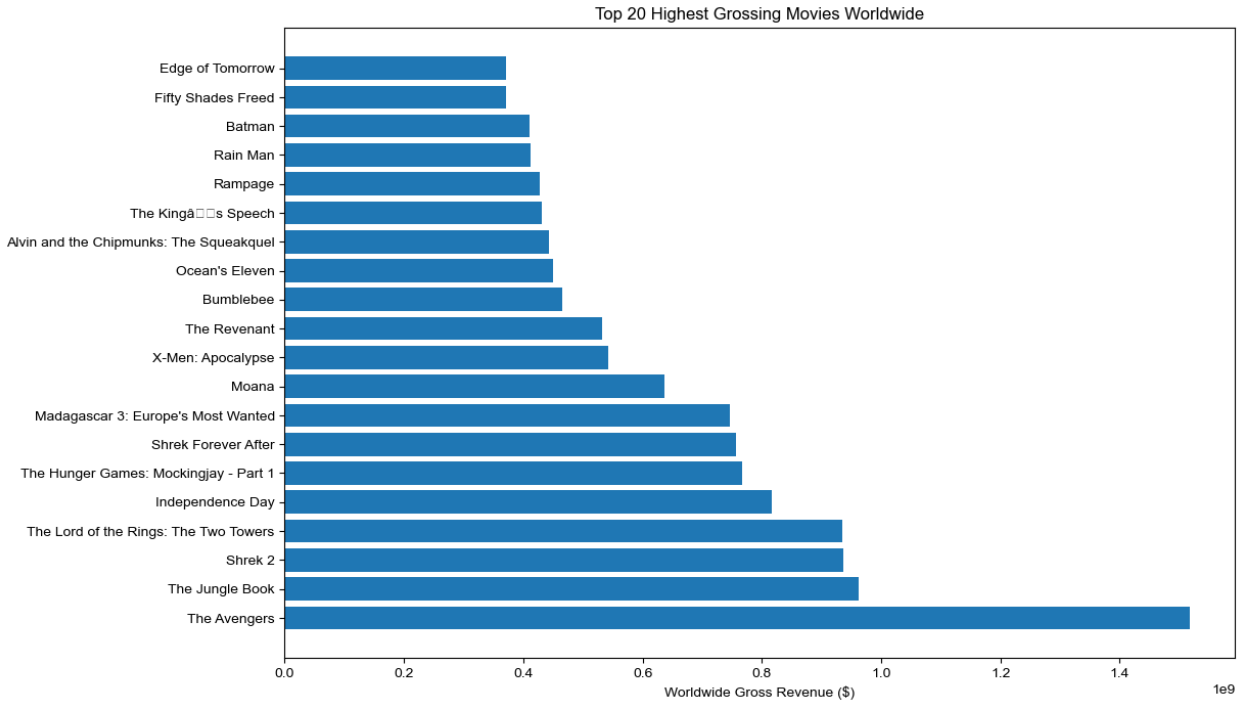
```
plt.figure(figsize=(10, 6))
merged_data['international_gross'] = merged_data['worldwide_gross'] -
merged_data['domestic_gross']
avg_domestic = merged_data['domestic_gross'].mean()
avg_international = merged_data['international_gross'].mean()
categories = ['Domestic Market', 'International Market']
revenues = [avg_domestic, avg_international]
plt.bar(categories, revenues, color=['skyblue', 'lightcoral'])
plt.ylabel('Average Revenue ($)')
plt.title('Average Domestic vs International Revenue')
plt.ticklabel_format(style='plain', axis='y')
plt.tight_layout()
plt.show()
```



5. Top 20 Highest Grossing Movies Worldwide

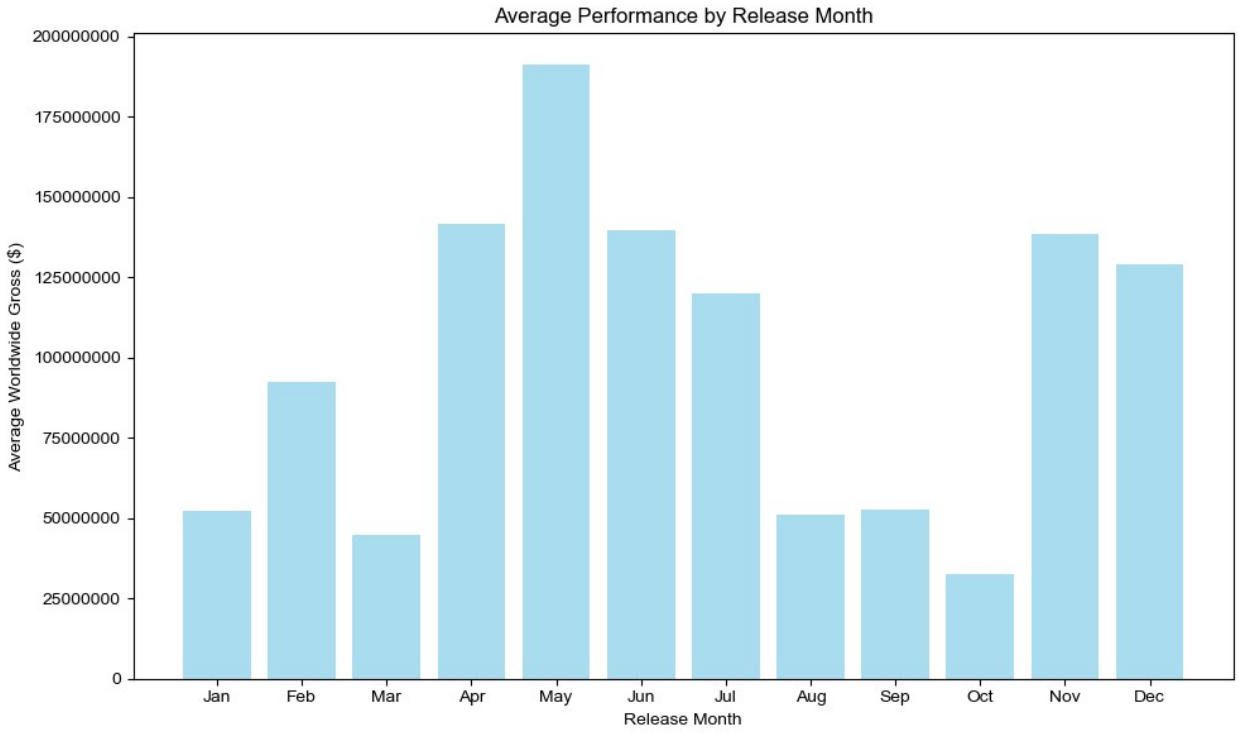
```
plt.figure(figsize=(12, 8))
top_20_movies = merged_data.nlargest(20, 'worldwide_gross')
plt.barh(range(len(top_20_movies)), top_20_movies['worldwide_gross'])
plt.yticks(range(len(top_20_movies)), top_20_movies['title'])
plt.xlabel('Worldwide Gross Revenue ($)')
plt.title('Top 20 Highest Grossing Movies Worldwide')

plt.show()
```



6. Seasonal Release Patterns

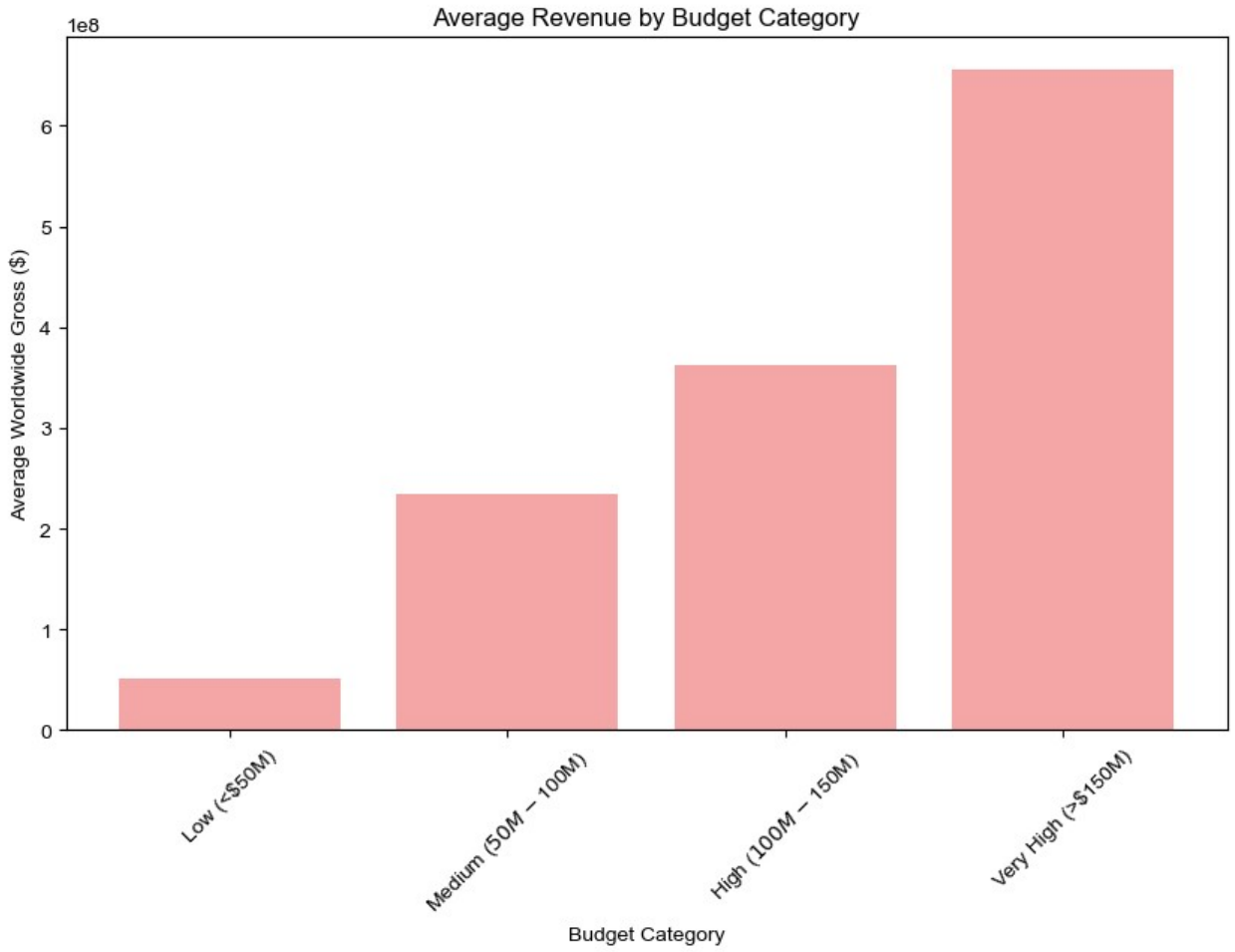
```
plt.figure(figsize=(10, 6))
merged_data['release_month'] =
pd.to_datetime(merged_data['release_date']).dt.month
monthly_avg = merged_data.groupby('release_month')
['worldwide_gross'].mean()
month_names = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
'Sep', 'Oct', 'Nov', 'Dec']
plt.bar(range(1, 13), [monthly_avg.get(i, 0) for i in range(1, 13)],
color='skyblue', alpha=0.7)
plt.xlabel('Release Month')
plt.ylabel('Average Worldwide Gross ($)')
plt.title('Average Performance by Release Month')
plt.xticks(range(1, 13), month_names)
plt.ticklabel_format(style='plain', axis='y')
plt.tight_layout()
plt.show()
```



7. Budget Ranges and Success Rates

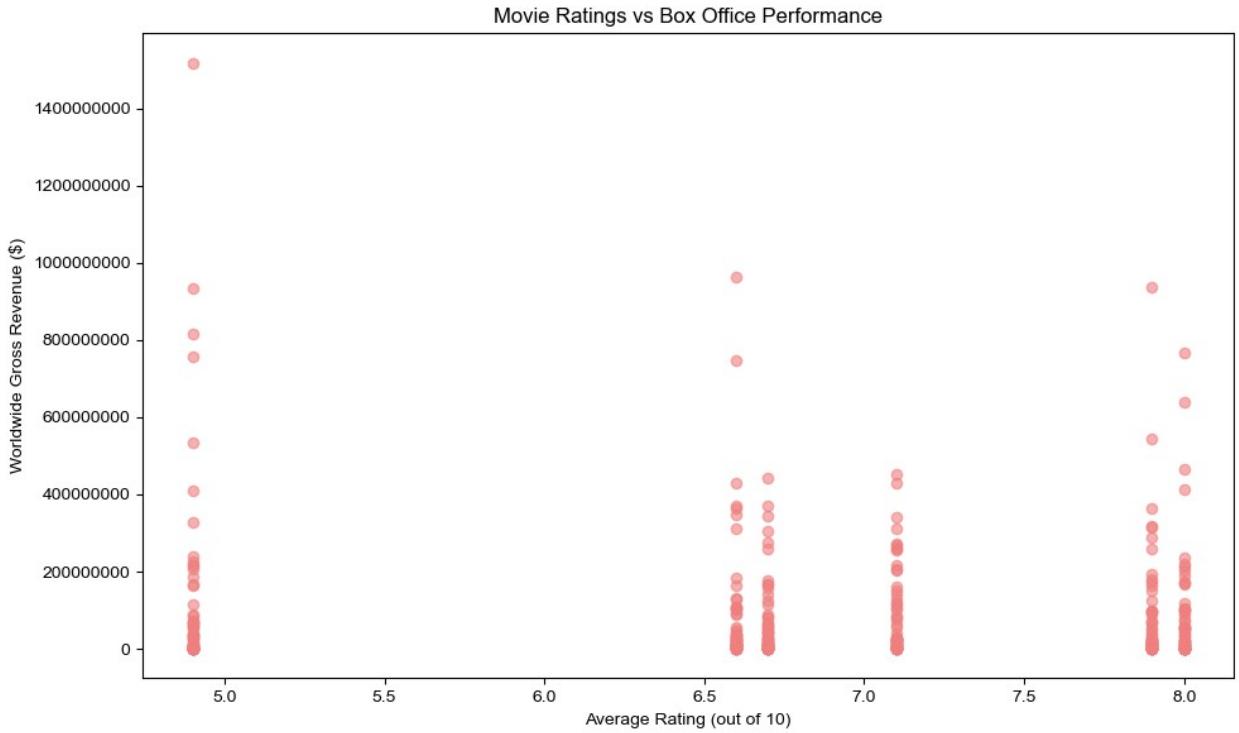
```
plt.figure(figsize=(10, 6))
budget_bins = [0, 50000000, 100000000, 150000000, float('inf')]
budget_labels = ['Low (<$50M)', 'Medium ($50M-$100M)', 'High ($100M-
$150M)', 'Very High (>$150M)']
merged_data['budget_category'] =
pd.cut(merged_data['production_budget'], bins=budget_bins,
labels=budget_labels)
budget_avg_gross = merged_data.groupby('budget_category')
['worldwide_gross'].mean()
plt.bar(budget_avg_gross.index, budget_avg_gross.values,
color='lightcoral', alpha=0.7)
plt.xlabel('Budget Category')
plt.ylabel('Average Worldwide Gross ($)')
plt.title('Average Revenue by Budget Category')
plt.xticks(rotation=45)

plt.show()
```



8. Vote Average vs Box Office Performance

```
plt.figure(figsize=(10, 6))
plt.scatter(merged_data['vote_average'],
merged_data['worldwide_gross'], alpha=0.6, color='lightcoral')
plt.xlabel('Average Rating (out of 10)')
plt.ylabel('Worldwide Gross Revenue ($)')
plt.title('Movie Ratings vs Box Office Performance')
plt.ticklabel_format(style='plain', axis='y')
plt.tight_layout()
plt.show()
```



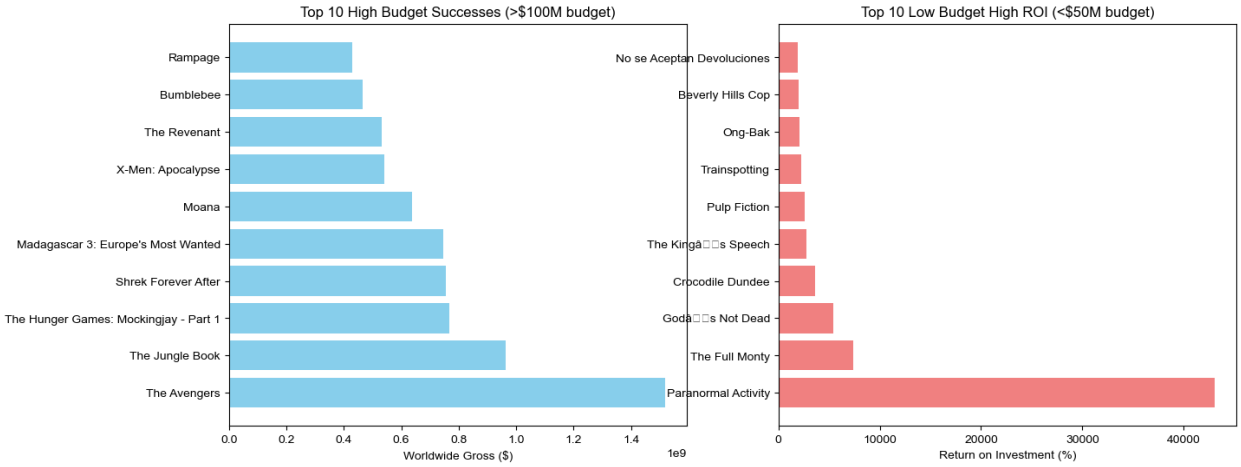
9. Top Performances: High Budget vs Low Budget Success Charts

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

# High budget successes
high_budget = merged_data[merged_data['production_budget'] >
100000000].nlargest(10, 'worldwide_gross')
ax1.barh(range(len(high_budget)), high_budget['worldwide_gross'],
color='skyblue')
ax1.set_yticks(range(len(high_budget)))
ax1.set_yticklabels(high_budget['title'])
ax1.set_xlabel('Worldwide Gross ($)')
ax1.set_title('Top 10 High Budget Successes (>$100M budget)')

# Low budget high ROI
low_budget = merged_data[merged_data['production_budget'] <
50000000].nlargest(10, 'roi')
ax2.barh(range(len(low_budget)), low_budget['roi'],
color='lightcoral')
ax2.set_yticks(range(len(low_budget)))
ax2.set_yticklabels(low_budget['title'])
ax2.set_xlabel('Return on Investment (%)')
ax2.set_title('Top 10 Low Budget High ROI (<$50M budget)')

plt.show()
```


Top 10 High Budget Successes (>$100M budget)
Rampage
Bumblebee
The Revenant
X-Men: Apocalypse
Moana
Madagascar 3: Europe's Most Wanted
Shrek Forever After
The Hunger Games: Mockingjay - Part 1
The Jungle Book
The Avengers
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e9
Worldwide Gross ($)
Top 10 Low Budget High ROI (<$50M budget)
No se Aceptan Devoluciones
Beverly Hills Cop
Ong-Bak
Trainspotting
Pulp Fiction
The King□□s Speech
Crocodile Dundee
God□□s Not Dead
The Full Monty
Paranormal Activity
0
10000
20000
30000
40000
Return on Investment (%)