IMDb

Exploratory Data Analysis

MOVIES

INTRODUCTION

IMDb (an acronym for Internet Movie Database)[2] is an online database of information related to films, television series, podcasts, home videos, video games, and streaming content online – including cast, production crew and personal biographies, plot summaries, trivia, ratings, and fan and critical reviews. IMDb began as a fan-operated movie database on the Usenet group "rec.arts.movies" in 1990, and moved to the Web in 1993. Since 1998, it has been owned and operated by IMDb.com, Inc., a subsidiary of Amazon.

The site's message boards were disabled in February 2017. As of 2019, IMDb was the 52nd most visited website on the Internet, as ranked by Alexa.[3] As of March 2022, the database contained some 10.1 million titles (including television episodes), 11.5 million person records, and 83 million registered users.[4]

DESCRIPTION

- I have conducted my work using Google Colab Notebook.
- The dataset has been imported from Google Drive.
- As we begin our Exploratory Data Analysis (EDA), I've named the dataset "imdb".
- The dataset comprises of 3000 rows and 23 columns.
- For data cleaning, I have utilized libraries like Numpy , Pandas , Matplotlib , Plotly and Seaborn .
- Any duplicate entries that were found have also been removed.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

```
[5] imdb = pd.read_csv('/content/drive/MyDrive/NOTES/data/Copy of imdb_data.csv')
```

```
[ ] imdb.drop_duplicates()
```

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

```
imdb.shape

(3000, 23)
```

DESCRIPTION

- **id :** It contain the serial number.
- **belongs_to_collection :** It contains the belongings of the movie \ series
- **Budget :** It contains how much money is available for the movie \ series
- **Genres :** The type of the movie \ series like – comedy , horror , thriller..etc
- **Homepage :** Contains the link of about the movie \ series.
- **imdb_id :** Unique Id of the movie \ series
- **original_language :** language in which a film or a performance work was originally created.
- **original_title :** Contains the original title of the movie \ series.
- **Overview :** A general summary .
- **Popularity:** The rating of the movie \ series.
- **poster_path :** The link of the poster of the movie \ poster.
- **production_companies :** Companies that produce the movie \ series
- **production_countries :** Countries that produce the movie \ series
- **release_date :** Release date of the movie \ series.

DESCRIPTION

- **Runtime :** The duration of the movie \ series.
- **spoken_languages :** The other languages spoken in the movie \ series.
- **Status :** The status of the movie \ series (released or not).
- **Tagline:** Contains the tag lines
- **Title :** Contains the title of the movie \ series.
- **Keywords :** Contains the main points about the movie \ series.
- **Cast :** The actor \ actress works in the movie \ series.
- **Crew :** The backend team of the movie \ series.
- **Revenue :** The collection of the movie \ series.

DESCRIPTION

```
imdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     3000 non-null   int64
 1   belongs_to_collection  604 non-null    object
 2   budget                 3000 non-null   int64
 3   genres                 2993 non-null   object
 4   homepage               946 non-null    object
 5   imdb_id                3000 non-null   object
 6   original_language      3000 non-null   object
 7   original_title         3000 non-null   object
 8   overview               2992 non-null   object
 9   popularity             3000 non-null   float64
 10  poster_path            2999 non-null   object
 11  production_companies   2844 non-null   object
 12  production_countries   2945 non-null   object
 13  release_date           3000 non-null   object
 14  runtime                2998 non-null   float64
 15  spoken_languages       2980 non-null   object
 16  status                 3000 non-null   object
 17  tagline                2403 non-null   object
 18  title                  3000 non-null   object
 19  Keywords               2724 non-null   object
 20  cast                   2987 non-null   object
 21  crew                   2984 non-null   object
 22  revenue                3000 non-null   int64
dtypes: float64(2), int64(3), object(18)
memory usage: 539.2+ KB
```

```
imdb.describe()
```

	id	budget	popularity	runtime	revenue
count	3000.000000	3.000000e+03	3000.000000	2998.000000	3.000000e+03
mean	1500.500000	2.253133e+07	8.463274	107.856571	6.672585e+07
std	866.169729	3.702609e+07	12.104000	22.086434	1.375323e+08
min	1.000000	0.000000e+00	0.000001	0.000000	1.000000e+00
25%	750.750000	0.000000e+00	4.018053	94.000000	2.379808e+06
50%	1500.500000	8.000000e+06	7.374861	104.000000	1.680707e+07
75%	2250.250000	2.900000e+07	10.890983	118.000000	6.891920e+07
max	3000.000000	3.800000e+08	294.337037	338.000000	1.519558e+09

Data Cleaning & Pre-Processing

❑ **The initial step in data cleaning involves removing the columns from the dataset that are not needed. The columns we remove are listed below :-**

- id
- homepage
- overview
- poster_path
- title
- spoken_languages
- tagline
- Keywords
- crew

```
imdb.drop(['id','homepage','overview','poster_path','title','spoken_languages','tagline','Keywords','crew'],axis=1,inplace=True)
```

Data Cleaning & Pre-Processing

- ❑ **Setting the new index = imdb_id .**

```
imdb = imdb.set_index('imdb_id')
```

- ❑ **Before moving to handeling null values part let me show you the new info and describe outputs.**

```
imdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   belongs_to_collection  604 non-null    object 
 1   budget                 3000 non-null   int64  
 2   genres                 2993 non-null   object 
 3   imdb_id                3000 non-null   object 
 4   original_language      3000 non-null   object 
 5   original_title         3000 non-null   object 
 6   popularity             3000 non-null   float64
 7   production_companies   2844 non-null   object 
 8   production_countries   2945 non-null   object 
 9   release_date           3000 non-null   object 
 10  runtime                2998 non-null   float64
 11  status                 3000 non-null   object 
 12  cast                   2987 non-null   object 
 13  revenue                3000 non-null   int64  
dtypes: float64(2), int64(2), object(10)
memory usage: 328.2+ KB
```

```
imdb.describe()
```

	budget	popularity	runtime	revenue
count	3.000000e+03	3000.000000	2998.000000	3.000000e+03
mean	2.253133e+07	8.463274	107.856571	6.672585e+07
std	3.702609e+07	12.104000	22.086434	1.375323e+08
min	0.000000e+00	0.000001	0.000000	1.000000e+00
25%	0.000000e+00	4.018053	94.000000	2.379808e+06
50%	8.000000e+06	7.374861	104.000000	1.680707e+07
75%	2.900000e+07	10.890983	118.000000	6.891920e+07
max	3.800000e+08	294.337037	338.000000	1.519558e+09

Data Cleaning & Pre-Processing

❑ Our dataset has a total of **2629** null values. Of these, **2627 are found in categorical features**, while **2 are in numerical features**.

```
imdb.isnull().sum()
```

	0
belongs_to_collection	2396
budget	0
genres	7
original_language	0
original_title	0
popularity	0
production_companies	156
production_countries	55
release_date	0
runtime	2
status	0
cast	13
revenue	0

dtype: int64

```
imdb.isnull().sum().sum()
```

2629

- The **‘belong-to-collection’** attribute, which has 2396 null values the missing entries with the **‘Not Available’** will help ensure data completeness

```
imdb['belongs_to_collection'].fillna('Not Available', inplace=True)
```

Data Cleaning & Pre-Processing

- The data completeness can be ensured by searching for missing entries with the **original values on the internet** for the **'genres'** attribute that have **7 null values**. So we create a function to get the original title and release date by imdb id.

```python
imdb[imdb['genres'].isna()].index
```

```
Index(['tt0349159', 'tt0261755', 'tt0110289', 'tt0352622', 'tt0984177',
       'tt0833448', 'tt1766044'],
      dtype='object', name='imdb_id')
```

```python
def get_details_by_index(imdb_id):
  return imdb.loc[imdb_id, ['release_date', 'original_title']]
```

```python
get_details_by_index('tt1766044')
```

	tt1766044
release_date	11/1/12
original_title	Poslednyaya skazka Rity

dtype: object

```python
imdb.loc['tt0349159', 'genres'] = 'Action,Adventure,Drama'
imdb.loc['tt0261755', 'genres'] = 'Comedy,Drama,Dance and Music'
imdb.loc['tt0110289', 'genres'] = 'Comedy/Drama'
imdb.loc['tt0352622', 'genres'] = 'Romance/Melodrama'
imdb.loc['tt0984177', 'genres'] = 'Action/Romance'
imdb.loc['tt0833448', 'genres'] = 'Thriller/Mystery'
imdb.loc['tt1766044', 'genres'] = 'Drama,Fantasy,Mystery-Suspense'
```

Data Cleaning & Pre-Processing

- The data completeness can be ensured by searching for missing entries with the **original values on the internet** for the **'production companies'** attribute that have **156 null values**. So we create a function to get the original title and release date by imdb id.

```
imdb[imdb['production_companies'].isna()].index.tolist()
```

Show hidden output

```
def get_details_by_index(imdb_id):
  return imdb.loc[imdb_id, ['release_date', 'original_title']]

get_details_by_index('tt1194664')
```

	tt1194664
release_date	11/3/07
original_title	恋空

dtype: object

```
imdb.loc['tt1821480', 'production_companies'] = 'Pen Movies, Boundscript, Viacom 18 Motion Pictures'
imdb.loc['tt1380152', 'production_companies'] = 'Realies Pictures'
imdb.loc['tt0093743', 'production_companies'] = 'Filmation, Filmation Studios'
imdb.loc['tt0391024', 'production_companies'] = 'Noujaim Films'
imdb.loc['tt2710368', 'production_companies'] = 'No Specific Company'
imdb.loc['tt0948530', 'production_companies'] = 'A Private View'
imdb.loc['tt0402906', 'production_companies'] = 'Hypermarket Film'
imdb.loc['tt0242572', 'production_companies'] = 'V Creations, Sri Surya Films'
imdb.loc['tt1582271', 'production_companies'] = 'Shore Z Productions, 3AD, and Entermedia'
imdb.loc['tt1894561', 'production_companies'] = 'The Walt Disney Company and Sense and Sensibility Ventures'
imdb.loc['tt0098061', 'production_companies'] = 'Working Title Films'
```

Data Cleaning & Pre-Processing

- The data completeness can be ensured by searching for missing entries with the original values on the internet for the 'production countries' attribute that have 55 null values. So we create a function to get the original title and release date by imdb id.

```
imdb[imdb['production_countries'].isna()].index.tolist()
```

Show hidden output

```
def get_details_of_index(imdb_id):
  return imdb.loc[imdb_id, ['release_date', 'original_title']]

get_details_by_index('tt0118663')
```

	tt0118663
release_date	3/28/97
original_title	B.A.P.S.

dtype: object

```
imdb.loc['tt0093743', 'production_countries'] = 'United States'
imdb.loc['tt0391024', 'production_countries'] = 'United States'
imdb.loc['tt2710368', 'production_countries'] = 'United States'
imdb.loc['tt0169302', 'production_countries'] = 'India'
imdb.loc['tt1092004', 'production_countries'] = 'United States'
imdb.loc['tt0096223', 'production_countries'] = 'United States'
imdb.loc['tt0105508', 'production_countries'] = 'United States'
imdb.loc['tt0349159', 'production_countries'] = 'United States'
imdb.loc['tt0841119', 'production_countries'] = 'United States'
imdb.loc['tt0120254', 'production_countries'] = 'United States'
imdb.loc['tt0123324', 'production_countries'] = 'United States'
imdb.loc['tt0083170', 'production_countries'] = 'United States'
imdb.loc['tt0119842', 'production_countries'] = 'United States'
imdb.loc['tt0110168', 'production_countries'] = 'United States'
```

Data Cleaning & Pre-Processing

- The data completeness can be ensured by searching for missing entries with the **original values on the internet** for the **'runtime'** attribute that have **2 null values**. So we create a function to get the original title and release date by imdb id.

```
imdb[imdb['runtime'].isna()].index.tolist()
```

```
['tt1107828', 'tt0116485']
```

```
def get_details_of_index(imdb_id):
  return imdb.loc[imdb_id, ['release_date', 'original_title']]

get_details_by_index('tt0116485')
```

	tt0116485
release_date	3/14/96
original_title	Happy Weekend

dtype: object

```
imdb.loc['tt1107828', 'runtime'] = '130.0'
imdb.loc['tt0116485', 'runtime'] = '81.0'
```

Data Cleaning & Pre-Processing

- **The data completeness can be ensured by searching for missing entries with the original values on the internet for the 'cast' attribute that have 13 null values. So we create a function to get the original title and release date by imdb id.**

```
imdb[imdb['cast'].isna()].index.tolist()
```

```
['tt0451279',
 'tt0319262',
 'tt1345836',
 'tt4425200',
 'tt0031679',
 'tt0364961',
 'tt3315342',
 'tt0993846',
 'tt0179626',
 'tt0960144',
 'tt2239822',
 'tt0443701',
 'tt1707386']
```

```
def get_details_of_index(imdb_id):
  return imdb.loc[imdb_id, ['release_date', 'original_title']]

get_details_by_index('tt1707386')
```

	tt1707386
release_date	12/18/12
original_title	Les Misérables

dtype: object

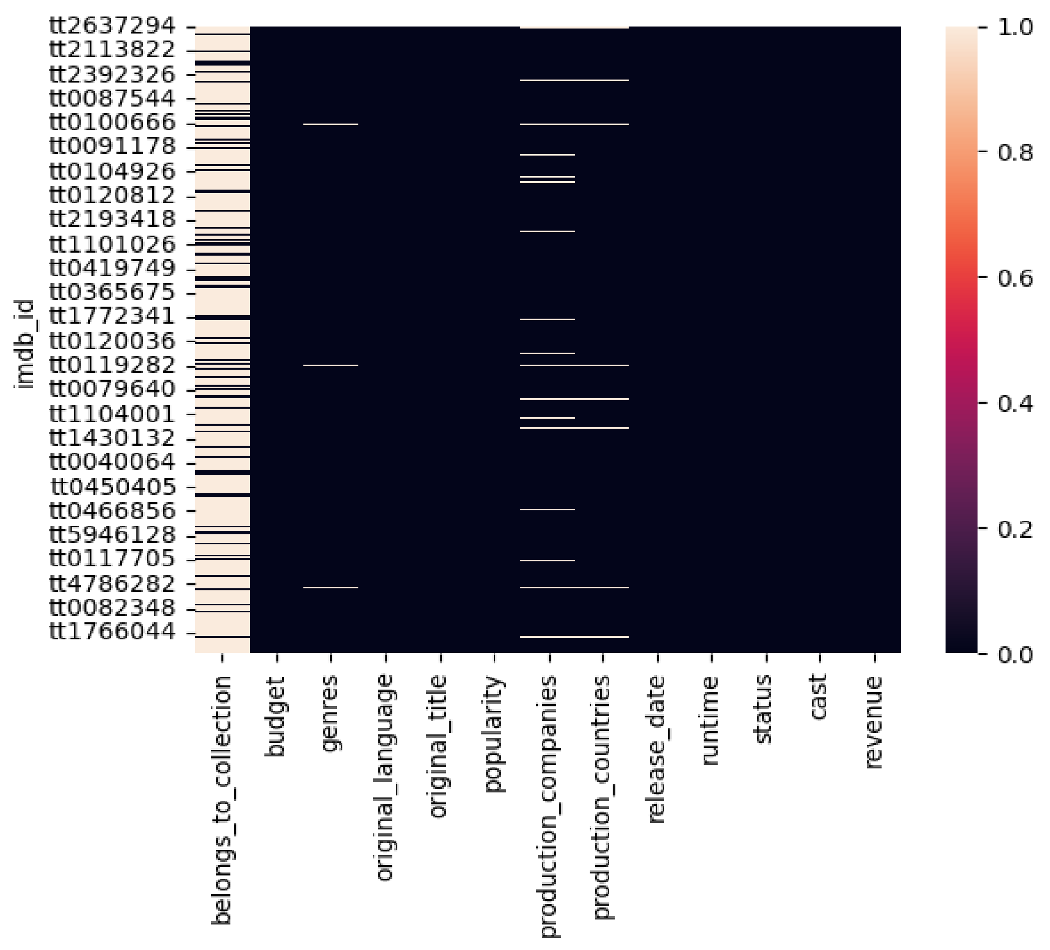
```
imdb.loc['tt0451279', 'cast'] = 'Gal Gadot · Diana ; Chris Pine · Steve Trevor ; Robin Wright · Antiope ; Lucy Davis · Etta ; Connie Nielsen · Hippolyta.'
imdb.loc['tt0319262', 'cast'] =  'Dennis Quaid, Jake Gyllenhaal, Emmy Rossum, Dash Mihok. Jack Hall'
imdb.loc['tt1345836', 'cast'] = 'Christian Bale · Gary Oldman · Tom Hardy · Joseph Gordon-Levitt · Anne Hathaway · Marion Cotillard · Morgan Freeman · Michael Caine'
imdb.loc['tt4425200', 'cast'] = ' Keanu Reeves · Riccardo Scamarcio · Ian McShane · Ruby Rose · Common · Claudia Gerini · Lance Reddick · Laurence Fishburne'
imdb.loc['tt0031679', 'cast'] = ' Jean Arthur · James Stewart · Claude Rains · Edward Arnold · Guy Kibbee · Thomas Mitchell · Eugene Pallette · Beulah Bondi'
imdb.loc['tt0364961', 'cast'] = 'Sean Penn, Naomi Watts, Jack Thompson, Don Cheadle, Michael Wincott'
imdb.loc['tt3315342', 'cast'] = 'Hugh Jackman · Patrick Stewart · Dafne Keen · Boyd Holbrook · Stephen Merchant · Elizabeth Rodriguez · Richard E. Grant · Eriq La Salle'
imdb.loc['tt0993846', 'cast'] = ' Leonardo DiCaprio · Jonah Hill · Margot Robbie · Matthew McConaughey · Kyle Chandler · Rob Reiner · Jon Bernthal · Jon Favreau.'
imdb.loc['tt0179626', 'cast'] = ' Robert De Niro · Edward Burns · Kelsey Grammer · Avery Brooks · Melina Kanakaredes · Karel Roden · Oleg Taktarov · Vera Farmiga.'
imdb.loc['tt0960144', 'cast'] = 'Adam Sandler · John Turturro · Emmanuelle Chriqui · Nick Swardson · Lainie Kazan · Ido Mosseri · Rob Schneider · Dave Matthews.'
imdb.loc['tt2239822', 'cast'] = ' Dane DeHaan · Cara Delevingne · Clive Owen · Rihanna · Ethan Hawke · Herbie Hancock · Kris Wu · Sam Spruell.'
imdb.loc['tt0443701', 'cast'] = 'David Duchovny, Gillian Anderson, Amanda Peet, Billy Connolly. Mulder and Scully'
imdb.loc['tt1707386', 'cast'] = ' Tom Hooper · Russell Crowe · Helena Bonham Carter · Anne Hathaway · Sacha Baron Cohen · Amanda Seyfried · Eddie'
```

HEATMAPS

Before Cleaning

```
sns.heatmap(imdb.isnull())
```

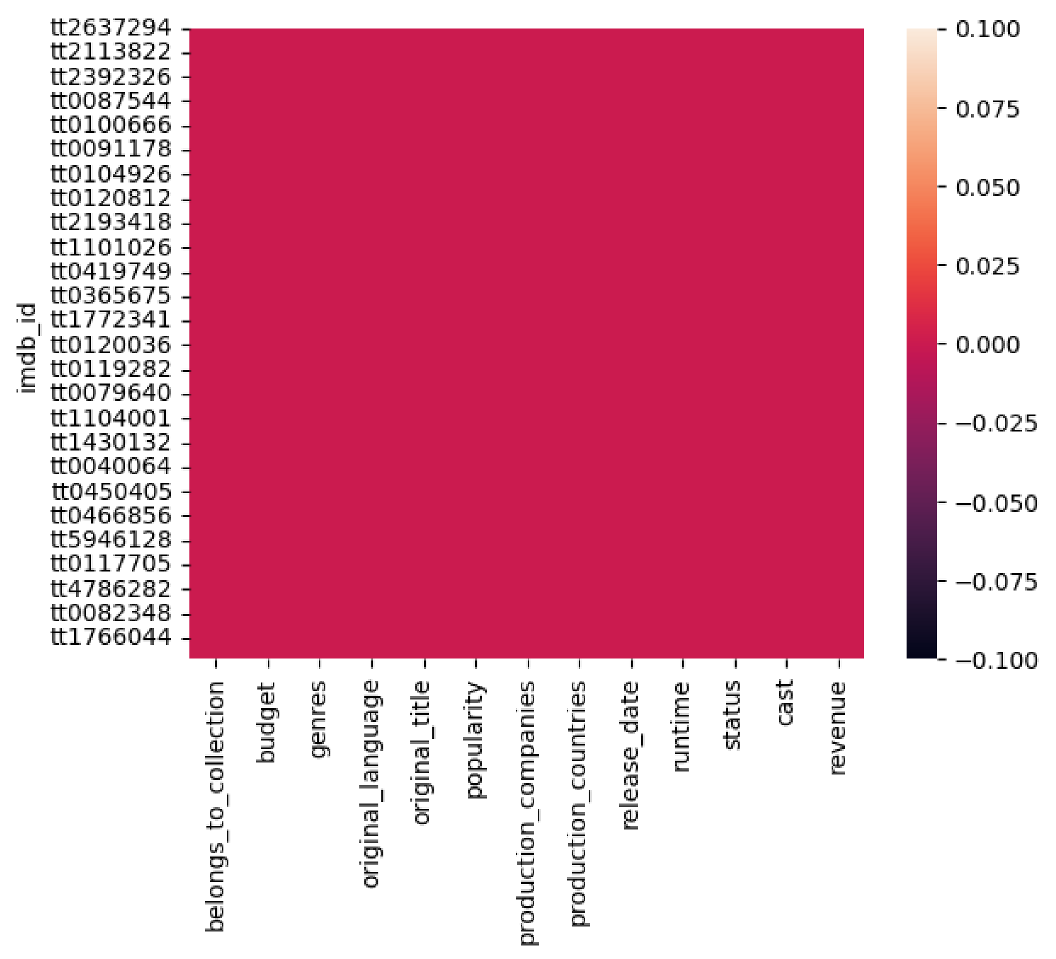
<Axes: ylabel='imdb_id'>



After Cleaning

```
sns.heatmap(imdb.isnull())
```

<Axes: ylabel='imdb_id'>



Data Cleaning & Pre-Processing

❑ **Adding 2 new columns in the dataset…**

➢ **The first one is Profit**

```
imdb['Profit'] = imdb['revenue'] - imdb['budget']
```

➢ **The second one is profit margin**

```
imdb['profit_margin'] = imdb['budget'] * 2
```

➢ **The third one is hit\flop**

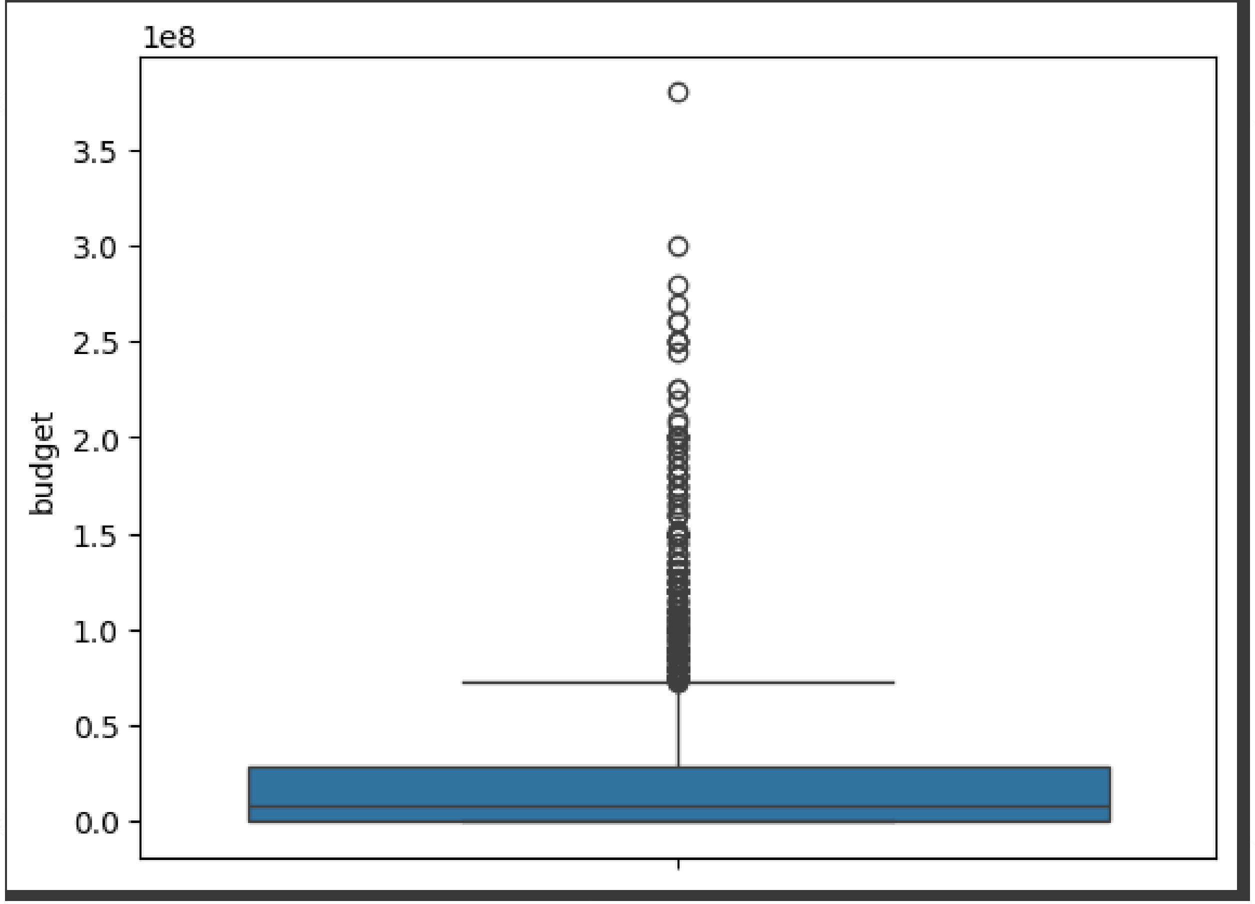
```
imdb['Hit or Flop'] = imdb['budget'] * 2

imdb['hit/flop'] = imdb.apply(lambda row: 'hit' if row['Profit'] >= row['Hit or Flop'] else 'flop', axis=1)
imdb.head()
```

Removing Outliers

After generating a box plot for the 'budget', we identified the presence of outliers in this column.

```
sns.boxplot(imdb['budget'])
```

```
<Axes: ylabel='budget'>
```



Removing Outliers

To address these outliers, we will apply the IQR method .

```
Q1 = imdb['budget'].quantile(0.25)
print(f"Q1 is {Q1}")

Q3 = imdb['budget'].quantile(0.75)
print(f"Q3 is {Q3}")
```

```
Q1 is 9000000.0
Q3 is 110000000.0
```

```
IQR = Q3 - Q1
print(f"IQR is {IQR}")
```

```
IQR is 101000000.0
```

```
lower_bound = Q1 - 1.5 * IQR
print(lower_bound)

upper_bound = Q3 + 1.5 * IQR
print(upper_bound)
```

```
-142500000.0
261500000.0
```

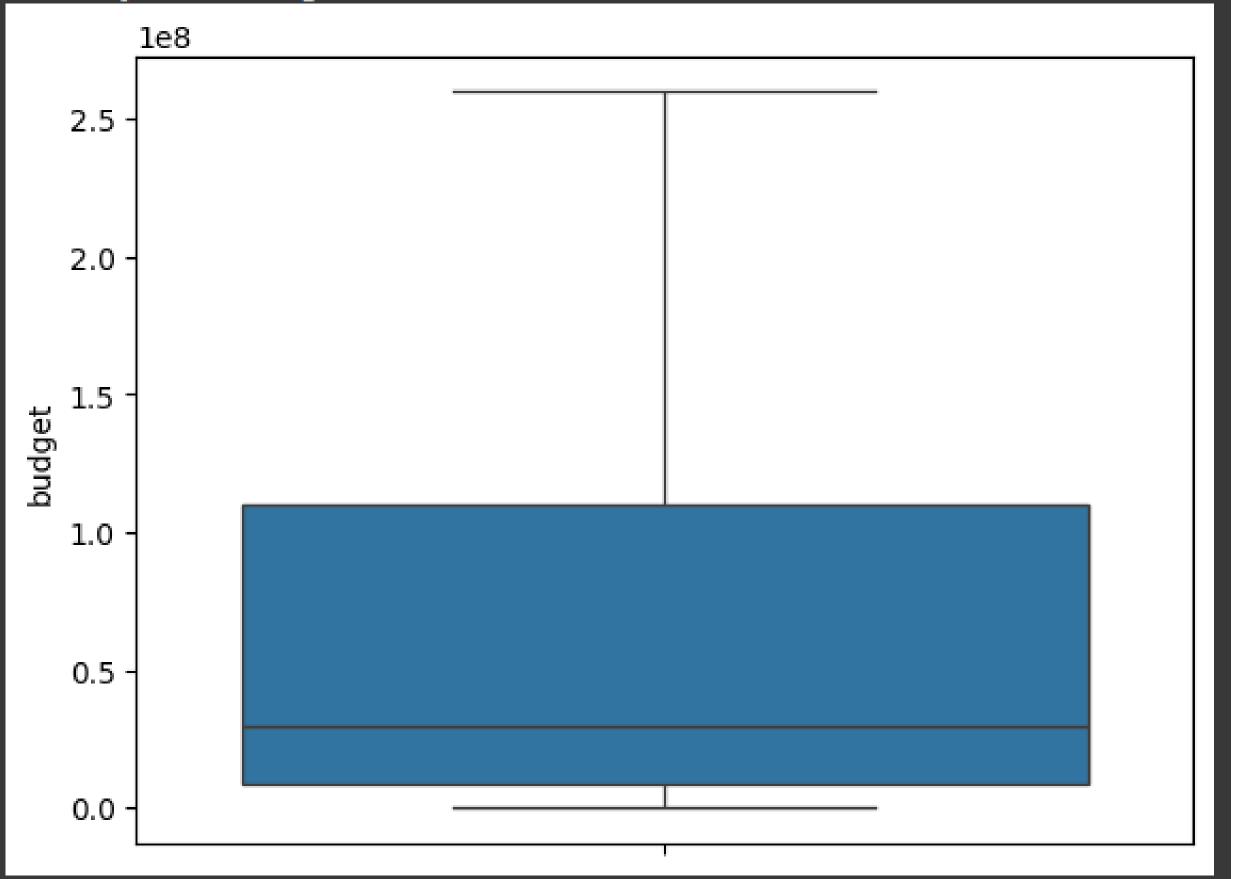
```
lower_bound = Q1 - 1.5 * IQR
print(lower_bound)

upper_bound = Q3 + 1.5 * IQR
print(upper_bound)
```

```
-142500000.0
261500000.0
```

```
outliers = imdb[(imdb['budget'] < lower_bound) | (imdb['budget'] > upper_bound)]
outliers
```

```
sns.boxplot(imdb['budget'])
```

```
<Axes: ylabel='budget'>
```




```
median_popularity = imdb['budget'].median()
```

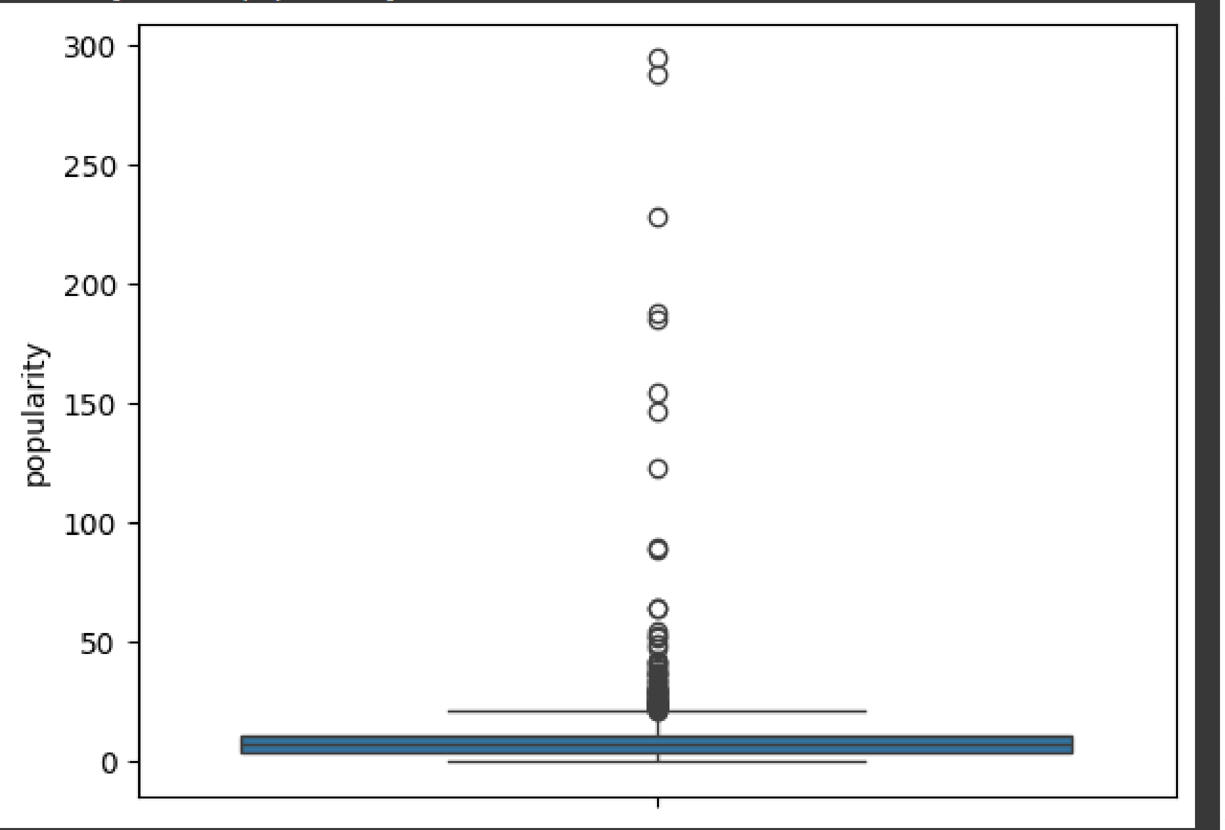
```
imdb['budget'] = np.where((imdb['budget'] < lower_bound) | (imdb['budget'] > upper_bound), median_popularity, imdb['budget'])
print(imdb['budget'])
```

Removing Outliers

After generating a box plot for the 'popularity', we identified the presence of outliers in this column.

```
sns.boxplot(imdb['popularity'])
```

<Axes: ylabel='popularity'>



Removing Outliers

To address these outliers, we will apply the **IQR method .**

```
Q1 = imdb['popularity'].quantile(0.25)
print(f"Q1 is {Q1}")

Q3 = imdb['popularity'].quantile(0.75)
print(f"Q3 is {Q3}")
```

```
Q1 is 4.018052750000001
Q3 is 10.449357249999998
```

```
IQR = Q3 - Q1
print(f"IQR is {IQR}")
```

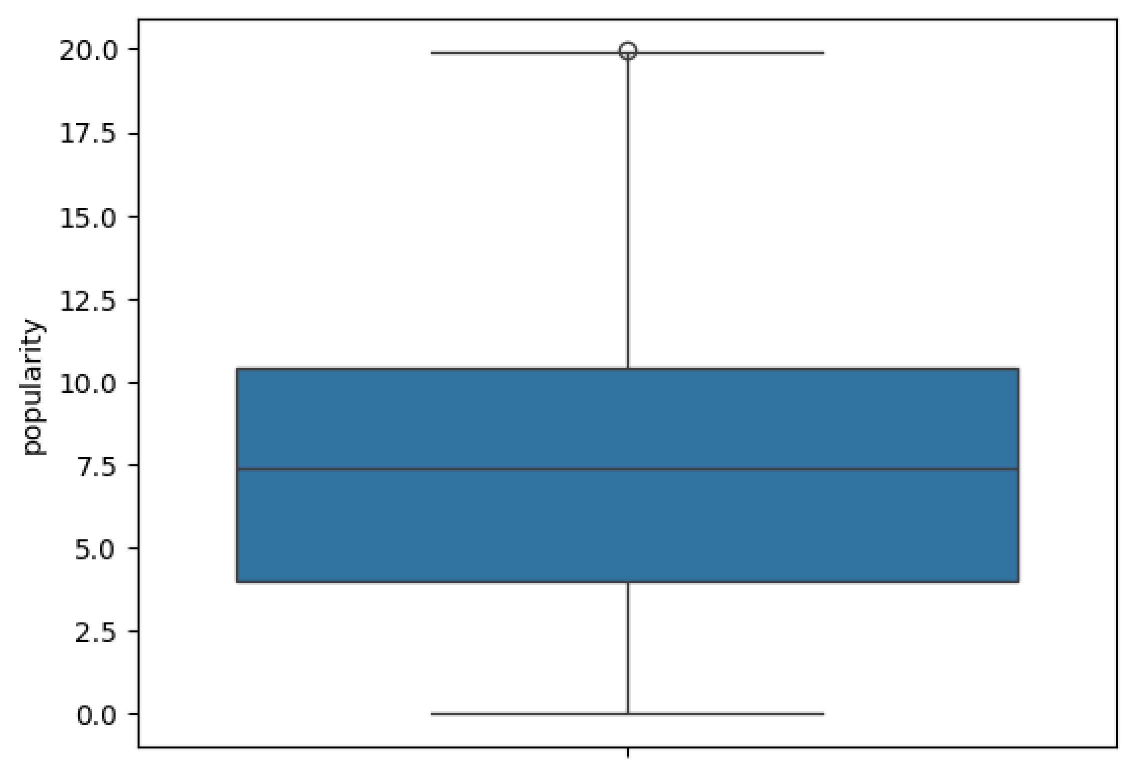
```
IQR is 6.431304499999998
```

```
lower_bound = Q1 - 1.5 * IQR
print(lower_bound)

upper_bound = Q3 + 1.5 * IQR
print(upper_bound)
```

```
-5.628903999999997
20.096313999999996
```

```
sns.boxplot(imdb['popularity'])
```

```
<Axes: ylabel='popularity'>
```



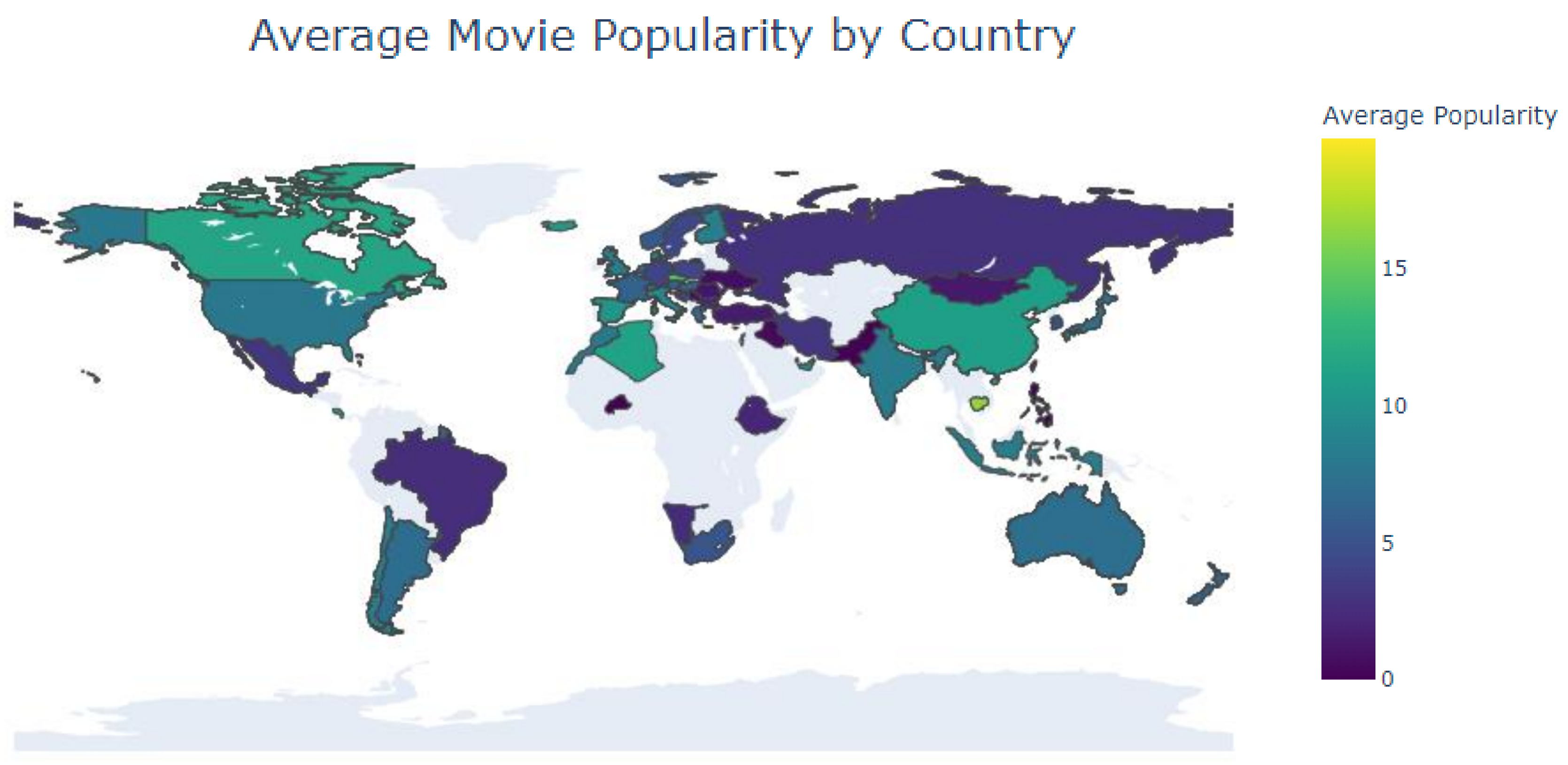

```
outliers = imdb[(imdb['popularity'] < lower_bound) | (imdb['popularity'] > upper_bound)]
outliers
```

Show hidden output

```
imdb['popularity'] = np.where((imdb['popularity'] < lower_bound) | (imdb['popularity'] > upper_bound), median_popularity, imdb['popularity'])
print(imdb['popularity'])
```

Data Visualization and Insights

- The popularity of movies vary across different countries



Key Findings :

- **Highest Popularity**: The countries with the highest average movie popularity are located in Central Asia, particularly Mongolia and Kazakhstan.

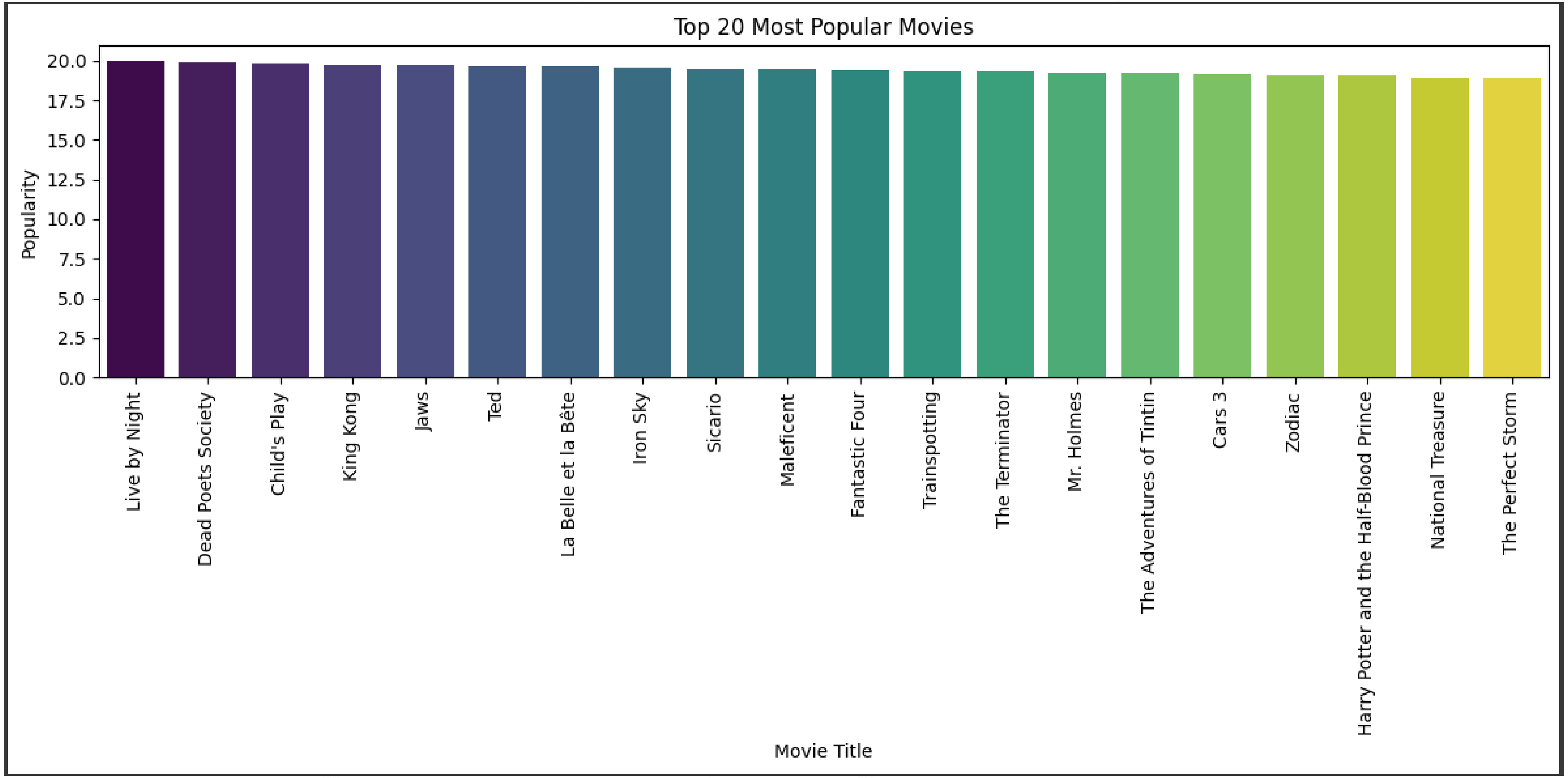
- **Moderate Popularity**: Europe, North America, and Australia show moderate levels of movie popularity.

- **Lower Popularity**: Africa and South America generally have lower average movie popularity.

- **Data Gaps**: Some countries have no data available, indicated by the grey color. This could be due to lack of data collection or reporting.

Data Visualization and Insights

- Top 20 most popular movies.



Key Findings :

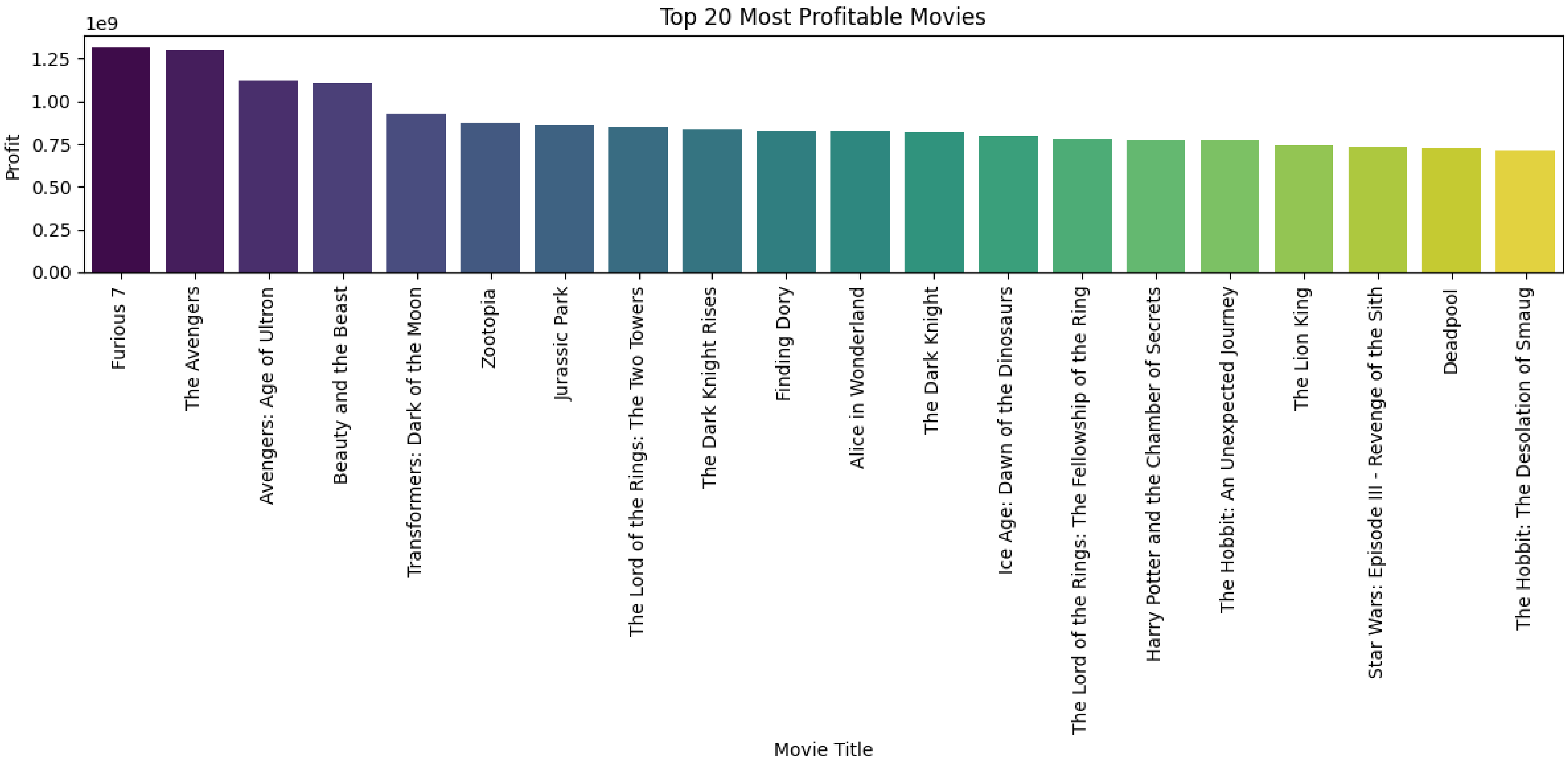
- The graph shows the top 20 most popular movies, ranked by their popularity. The most popular movie is "Live by Night," followed by "Dead Poets Society" and "Child's Play".

- The x-axis represents the movie titles, while the y-axis represents the popularity score. The color gradient from dark blue to bright yellow indicates the ranking, with darker colors representing higher popularity.

- The graph provides a quick visual overview of the popularity of these movies, allowing for easy comparisons between them.

Data Visualization and Insights

- Top 20 most profitable movies.



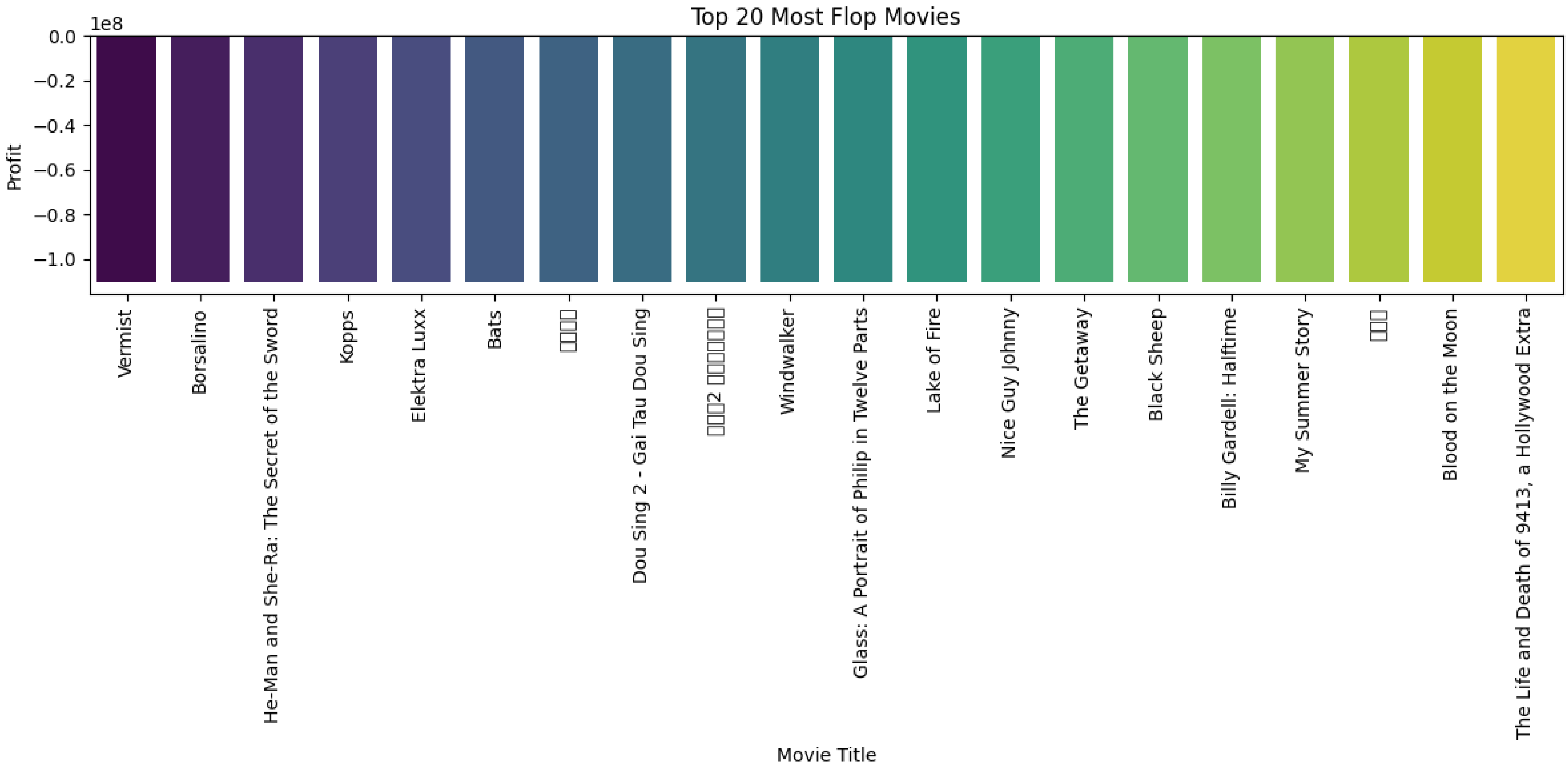
Key Findings :

- **Highest profit**: Furious 7 is the most profitable movie, with a profit of over $1.5 billion.
- **Marvel dominance**: The Avengers and Avengers: Age of Ultron are in the top 3, indicating the strong profitability of Marvel movies.
- **Successful franchises** : Several franchises appear multiple times on the list, including The Lord of the Rings, The Hobbit, and Jurassic Park. This highlights the profitability of sequels and established fanbases.
- **Animated films**: Finding Dory and Zootopia are among the top 20, demonstrating the potential profitability of animated films.
- **Variety of genres**: The list includes a mix of genres, including action, adventure, fantasy, and animation.

Overall, the graph highlights the key factors contributing to a movie's profitability, such as strong franchises, popular genres, and successful marketing campaigns.

Data Visualization and Insights

- Top 20 flop movies.



Key Findings :

- The movie with the biggest loss is "The Life and Death" of 9413, a Hollywood Extra.

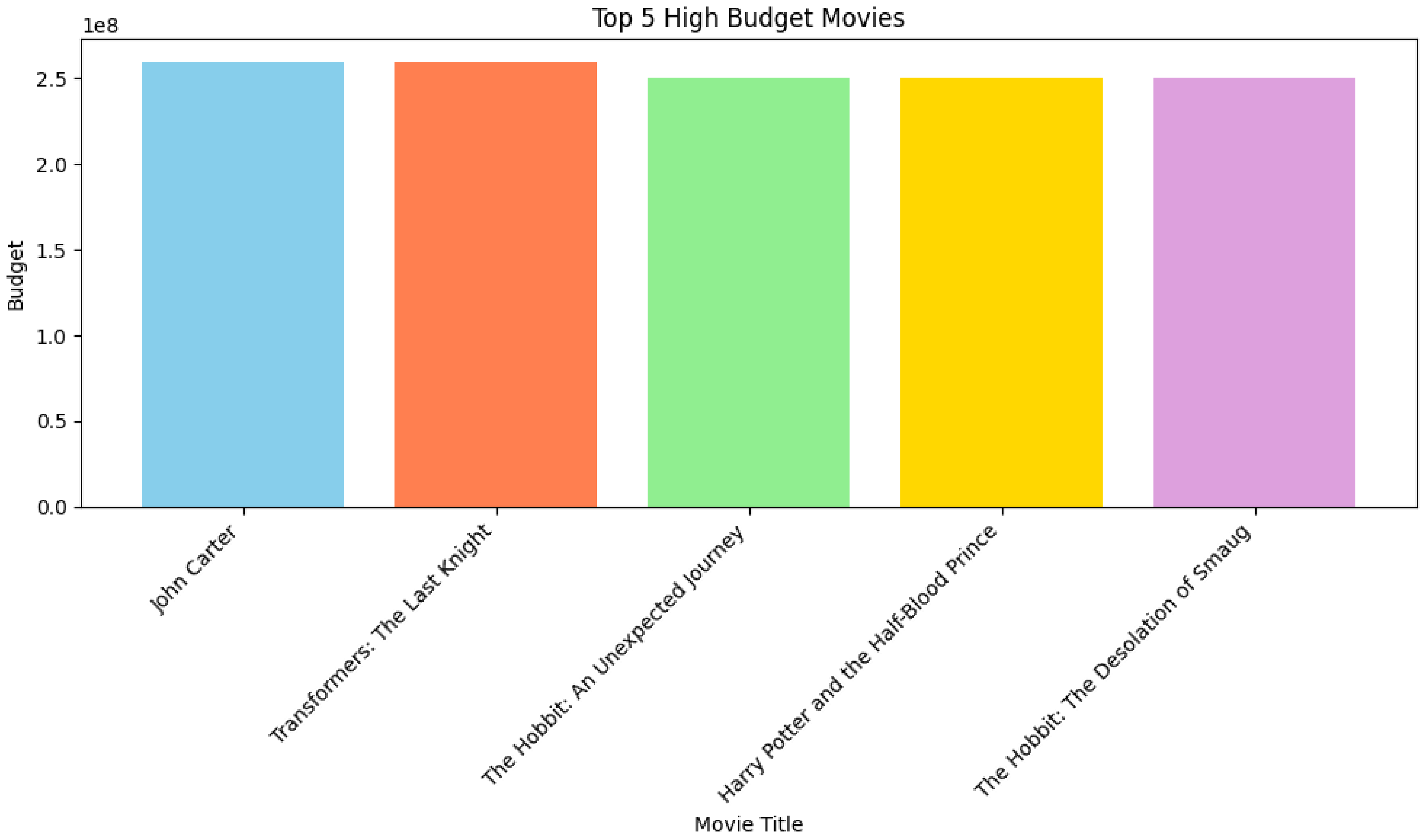
- All the movies on the list lost money, with profits ranging from 0 to -1.0.

- The x-axis shows the profit in negative values, indicating the extent of financial loss.

- The graph reveals that "Vermist" is the biggest flop, followed by "Borsalino" and "He-Man and She-Ra: The Secret of the Sword". It also shows that some movies, like "The Getaway" and "Black Sheep," lost less money than others.

Data Visualization and Insights

- Top 5 high budget movies.

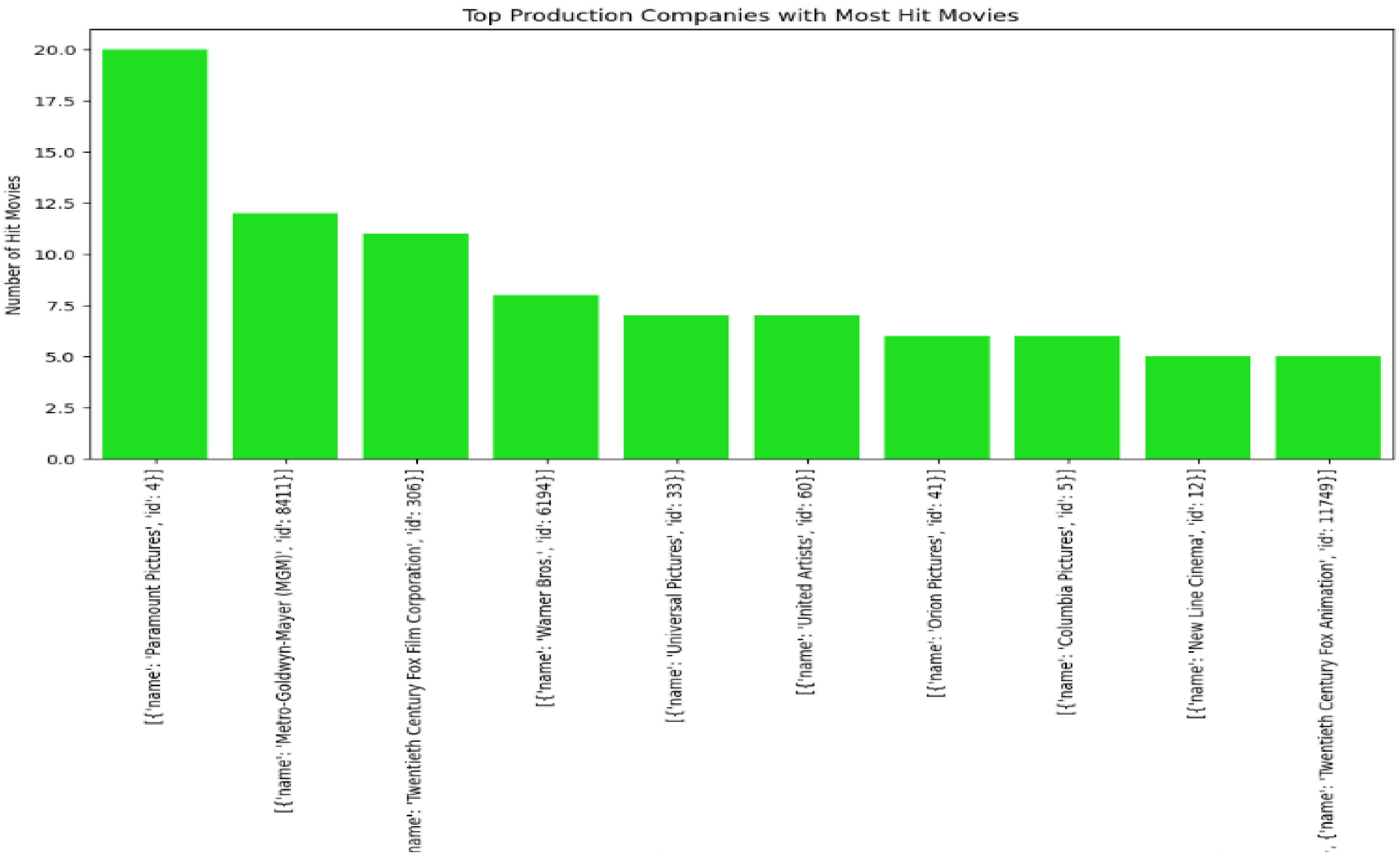


Key Findings :

- The movie with the highest budget is **"John Carter"** with a budget of slightly over **2.5 million dollars**
- The other four movies on the list have **budgets** between **2.25 and 2.5 million dollars.**
- All five movies are big-budget productions, likely involving extensive special effects and large casts.
- The budgets of these movies are significantly higher than the average movie budget, indicating that they are major investments for the studios involved.

Data Visualization and Insights

- Top production companies with hit movies.



Key Findings :

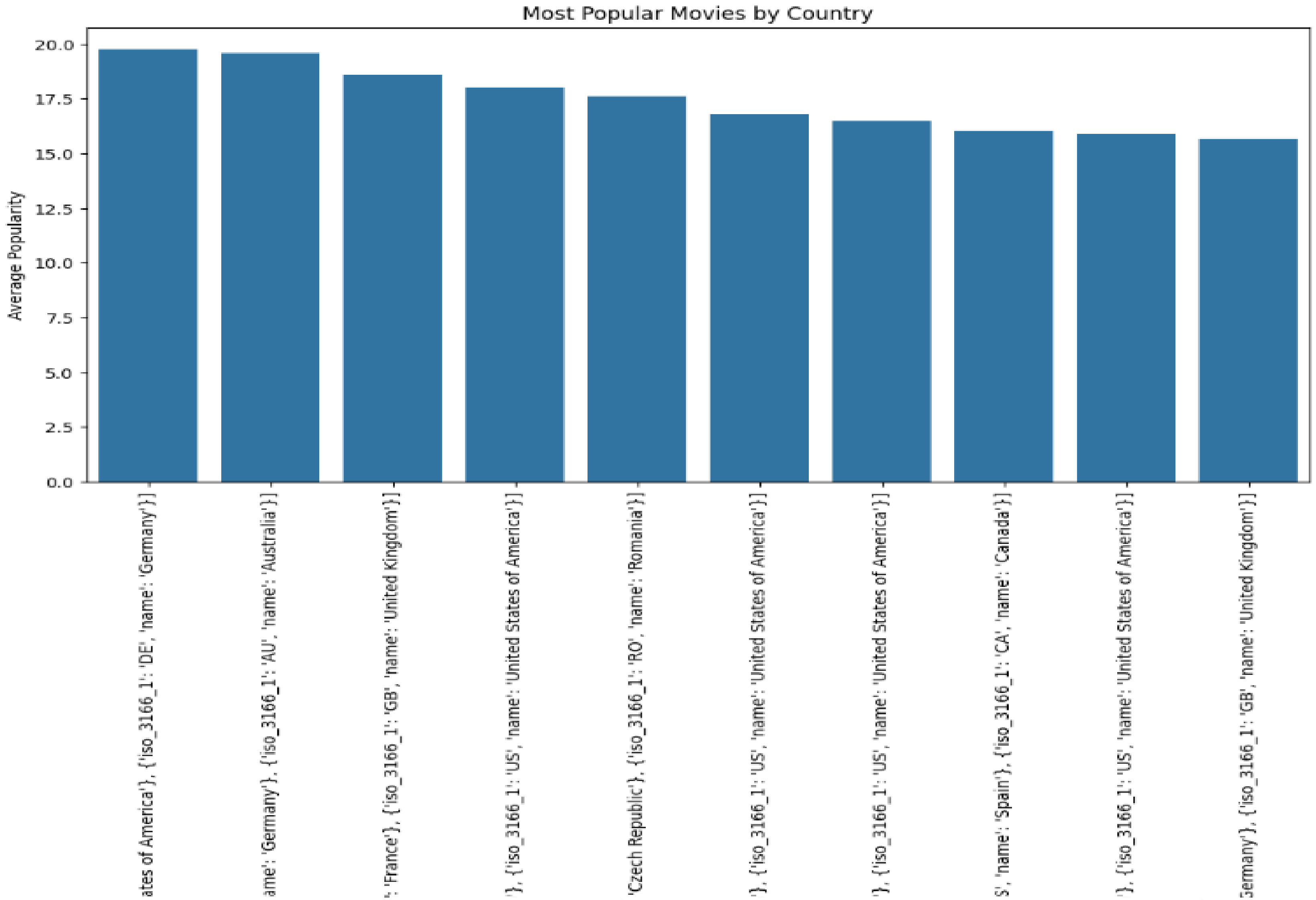
- Paramount Pictures leads with the most hit movies, followed closely by Metro-Goldwyn-Mayer (MGM) and Twentieth Century Fox

- The top 3 companies have a significant lead over the rest.

- There's a steep drop-off in the number of hit movies after the top 5 companies

Data Visualization and Insights

- Most popular by the Country.



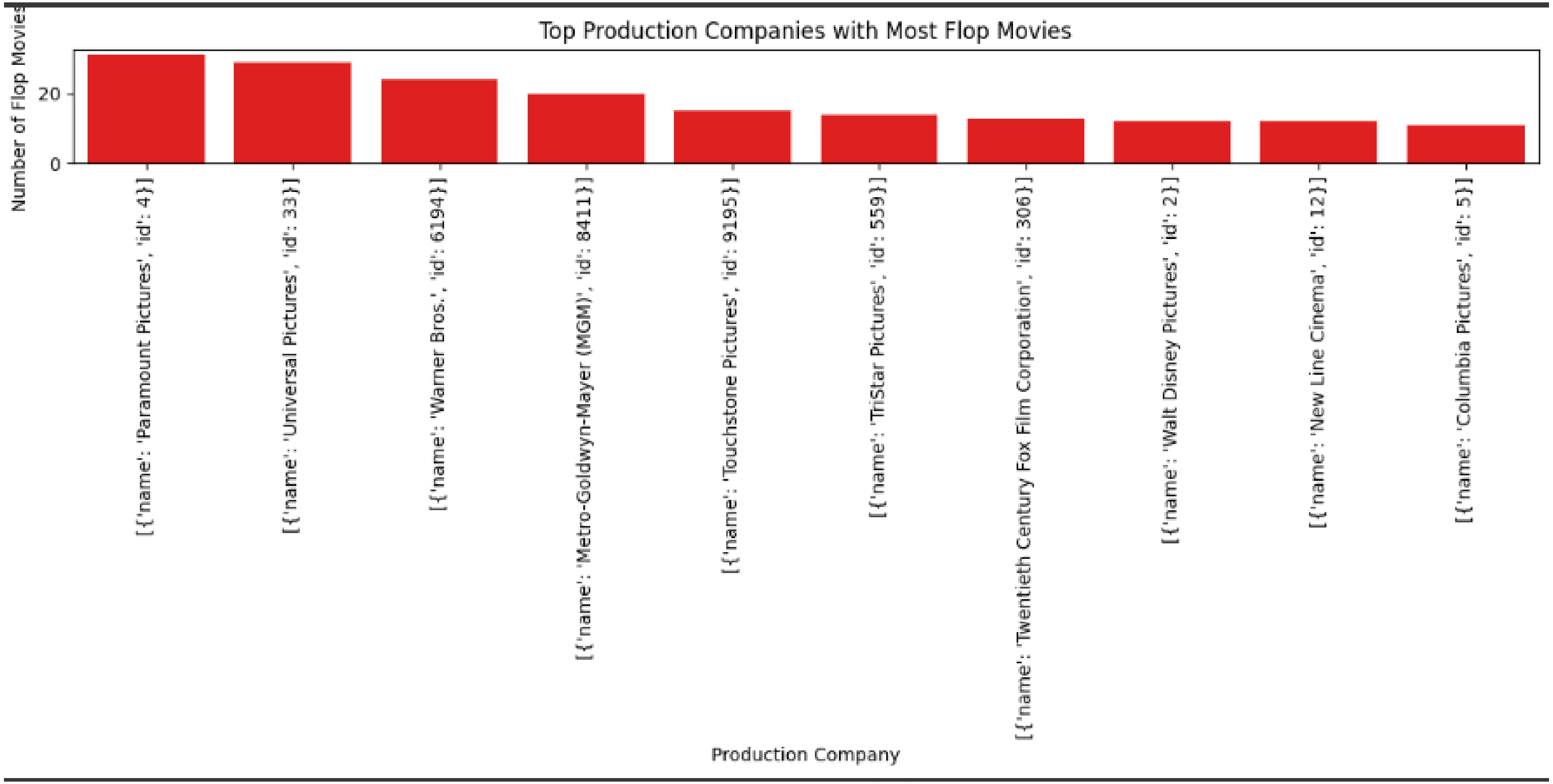
Key Findings :

- This graph shows the average popularity of the most popular movies by country. The countries listed are **Germany, Australia, The United Kingdom, The United States, Romania, Canada, and Spain.**

- **The United States** has the highest average popularity, followed by **Germany and Australia**. This means that movies from the **United States** are generally more popular than movies from other countries

Data Visualization and Insights

- Top production companies with most flop movies.

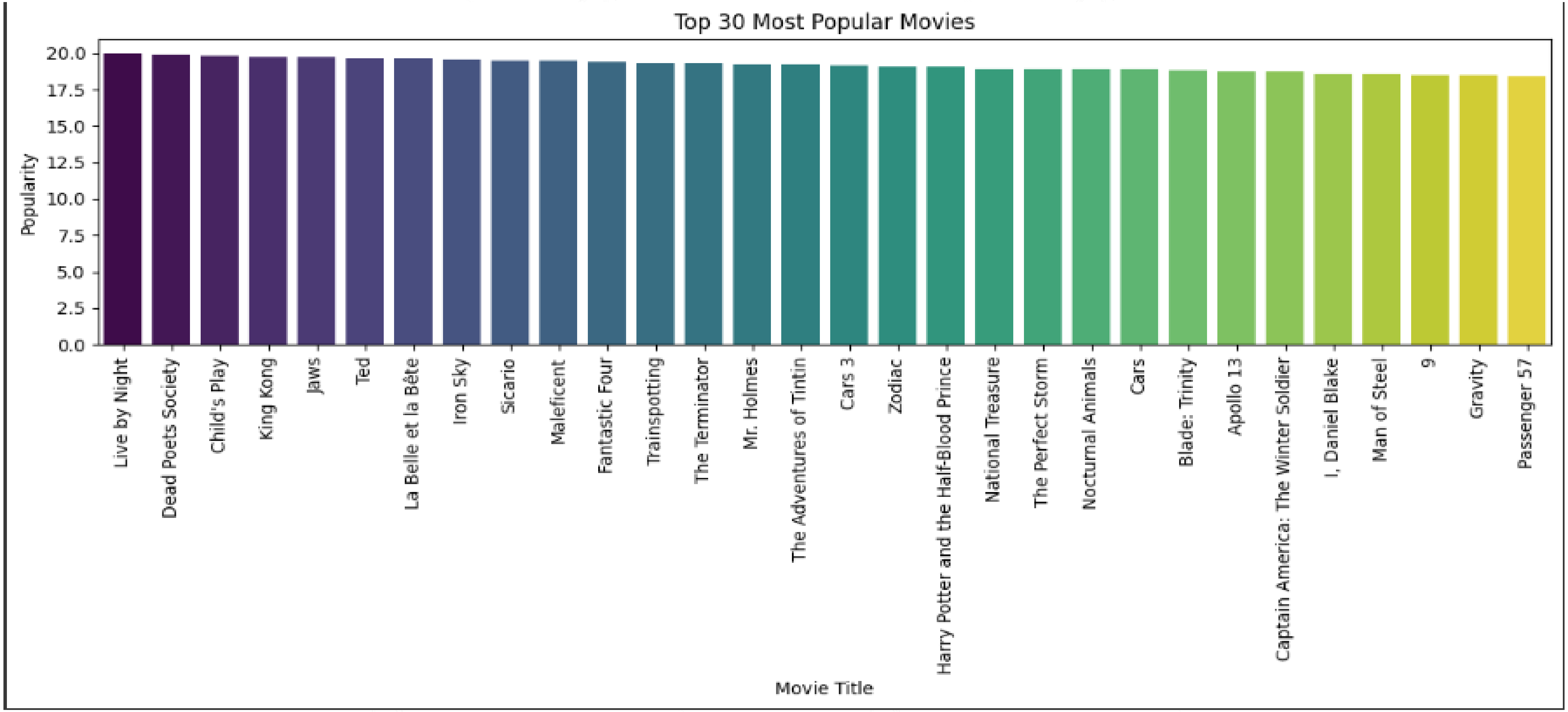


Key Findings :

- Paramount Pictures and Universal Pictures are tied for the most flop movies.
- Columbia Pictures has the least flop movies among the top 10
- The number of flop movies for each company ranges from approximately 15 to 25.

Data Visualization and Insights

- Top 30 most popular movies.



Key Findings :

- ”Live by Night”: is the most popular movie, followed by ”Dead Poets Society” and ”Child’s Play“ .

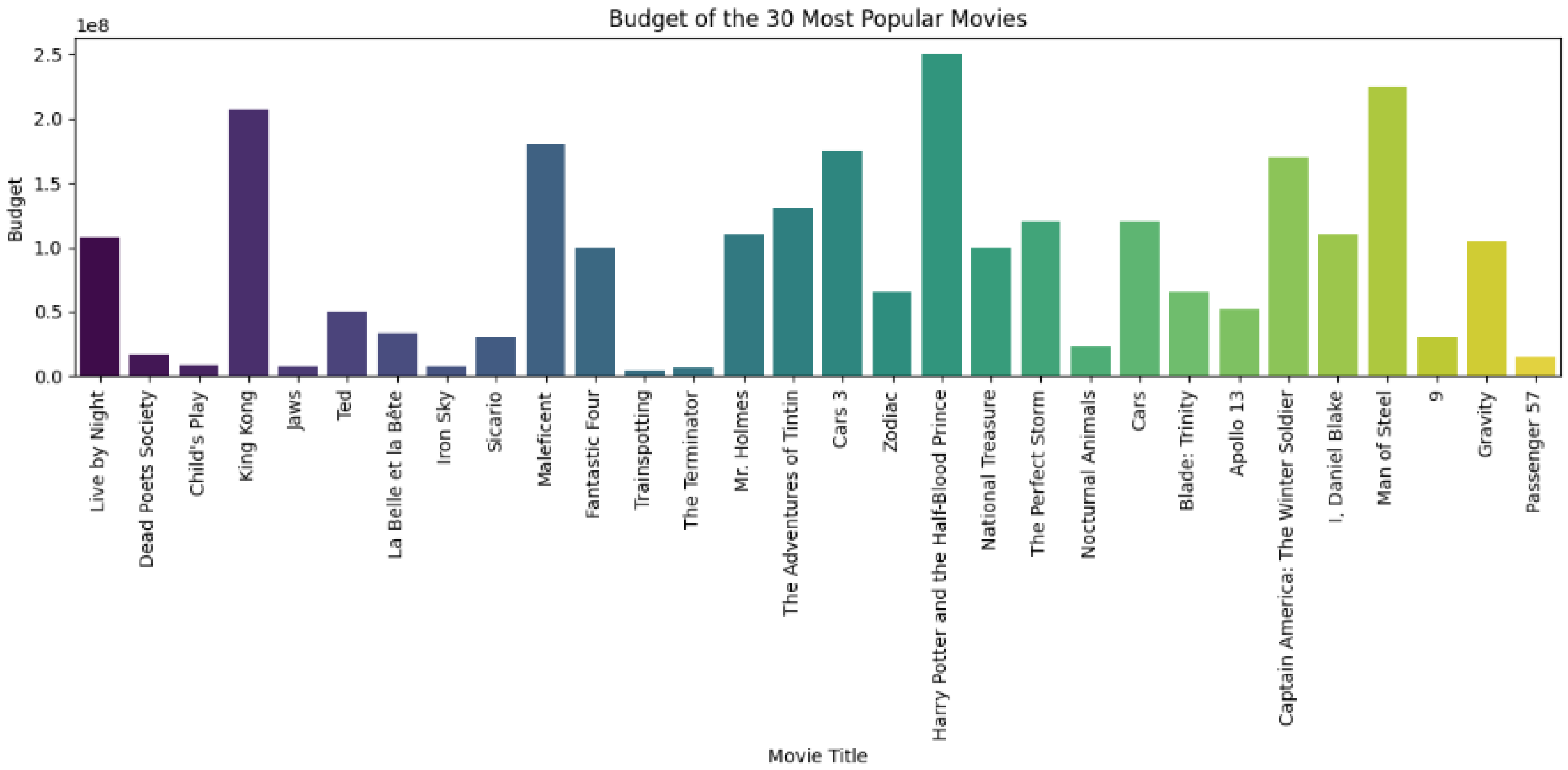
- The least popular movie on the list is ”Passenger 57“.

- There is a wide range of popularity among the movies on the list, with some being much more popular than others.

- The most popular movies tend to be newer releases, while the less popular movies are often older films.

Data Visualization and Insights

- **Budget of 30 most popular movies.**



Key Findings :

- Highest Budget : The movie with the highest budget is "Live by Night", exceeding $250 million.

- Lowest Budget : The movie with the lowest budget is "Passenger 57", costing less than $25 million

- Wide Range : The budgets vary significantly, with some movies costing over ten times more than others.

- Mid-Range : Most movies fall in the $100-200 million budget range.

Final Report :

- The dataset contains information about **3000 movies**.

- The dataset contained information on various movies, including **budget, revenue, popularity, genres, popularity, and cast and more.**

- There are **more flops than hits** in the dataset.

- Hits generally have a higher **average budget** and revenue compared to flops.

- There is a **positive correlation** between **budget** and **revenue**, suggesting that movies with higher budgets tend to generate more revenue.

- The most **common genres** in the dataset are **Drama, Comedy, and Thriller**.

- There is a **positive correlation** between **popularity and revenue**, indicating that more popular movies tend to generate more revenue.

Final Report :

- The dataset contained information on various movies, including budget, revenue, popularity, and more.

- Data cleaning involved handling missing values in 'production_companies', 'production_countries', 'runtime', and 'cast' columns

- Outliers in 'budget' and 'popularity' were identified using the IQR method and replaced with the median value.

- A 'Profit' column was calculated and a 'hit/flop' classification was added based on profit margins.

- Further analysis can be performed to explore correlations between variables and gain deeper insights into movie success factors.

THANKS FOR READING--->>>

IMDb

FOR CODING PART-->>>

https://github.com/sahilyadav7i/IMDB-EDA