example

Analysis for the launch of Microsoft Movie Studio

Authors: Nishmitha Naik

Overview

Microsoft sees all the big companies creating original video content and they want to get in on the fun. They have decided to create a new movie studio, but they don’t know anything about creating movies. By analyzing top genres, box office performance, and user ratings for different genres, we can leverage data from IMDb's title.basics, title.ratings and Box Office Mojo's movie_gross datasets. This analysis will provide actionable insights for Microsoft to consider when developing new movies.

Business Problem

Microsoft lacks experience in the film industry and needs guidance in developing content for their new movie studio. They need to identify movie genres with high box office potential to maximize their return on investment. We can analyze the below mentioned points to answer the business perspective questions.

1. Genre Performance: Which movie genres historically generate the highest average box office gross?

2. Genre Profitability: Considering foreign gross and domestic gross, which genres offer the highest potential for profit margins?

3. Genre Trends: Are there any emerging genres or sub-genres experiencing significant growth in popularity and box office success?

Data Understanding

The data files provide the foundation for analyzing movie performance and generating actionable insights for Microsoft.

1. IMDb: Data from IMDb "title.basics" and "title.ratings" datasets can provide information on:
   A. Movie titles and release years.
   B. Genre classifications for each movie.
   C. User ratings, offering insights into audience reception.

2. Box Office Mojo: Data from Box Office Mojo "movie_gross" dataset can offer:
   A. Domestic and international box office gross for each movie.

In [2]:
```
# Import standard packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [3]:
```
#loading data
df_title = pd.read_csv("./dsc-project-template-template-mvp/zippedData/im
df_ratings = pd.read_csv("./dsc-project-template-template-mvp/zippedData/
df_bom = pd.read_csv("./dsc-project-template-template-mvp/zippedData/bom.
```

In [4]:
```
df_title.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   tconst           146144 non-null  object 
 1   primary_title    146143 non-null  object 
 2   original_title   146122 non-null  object 
 3   start_year       146144 non-null  int64  
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object 
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
```

In [5]:
```
df_ratings.info()
```

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

In [6]:
```
df_bom.info()
```

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

In [7]:
```
df_title.head()
```

Out[7]:

	tconst	primary_title	original_title	start_year	runtime_minutes	
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Cri
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biograp
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Come
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Dram

In [8]:
```
df_ratings.head()
```

Out[8]:

	tconst	averagerating	numvotes
0	tt10356526	8.3	31
1	tt10384606	8.9	559
2	tt1042974	6.4	20
3	tt1043726	4.2	50352
4	tt1060240	6.5	21

In [9]:
```
df_bom.head()
```

Out[9]:

	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

Data Preparation

Step 1 : Merging the datasets

1. To integrate the datasets, we'll merge them based on common identifiers.

2. In this case, we'll use the "tconst" from the second dataset as the common identifier to merge with the first and third datasets.
3. Once merged, we can proceed with to clean the datas to adrres missing values.

Step 2 : Cleaning the datasets

1. Dropping rows with missing tconst, primary_title, and original_title
2. Filling missing values for start_year, runtime_minutes, averagerating, numvotes with median.
3. Dropping rows with missing genres as imputing might not be accurate
4. Filling missing studio, domestic_gross, foreign_gross, year with appropriate values
5. Verify if there are any remaining missing values

In [10]:
```
# Merge datasets
merged_df = pd.merge(df_title, df_ratings, on='tconst', how='inner')
merged_df = pd.merge(merged_df, df_bom, left_on='primary_title', right_on

# Drop unnecessary columns
merged_df.drop(columns=['title'], inplace=True)
```

In [11]:
```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3027 entries, 0 to 3026
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   tconst           3027 non-null   object 
 1   primary_title    3027 non-null   object 
 2   original_title   3027 non-null   object 
 3   start_year       3027 non-null   int64  
 4   runtime_minutes  2980 non-null   float64
 5   genres           3020 non-null   object 
 6   averagerating    3027 non-null   float64
 7   numvotes         3027 non-null   int64  
 8   studio           3024 non-null   object 
 9   domestic_gross   3005 non-null   float64
 10  foreign_gross    1832 non-null   object 
 11  year             3027 non-null   int64  
dtypes: float64(3), int64(3), object(6)
memory usage: 283.9+ KB
```

In [12]:
```
merged_df.head()
```

Out[12]:

	tconst	primary_title	original_title	start_year	runtime_minutes	
0	tt0315642	Wazir	Wazir	2016	103.0	Action
1	tt0337692	On the Road	On the Road	2012	124.0	Adventure,Dra
2	tt4339118	On the Road	On the Road	2014	89.0	
3	tt5647250	On the Road	On the Road	2016	121.0	
4	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adventure,C

In [ ]:

```
#Clean the data set
```

In [13]:

```
# Dropping rows with missing tconst, primary_title, and original_title
merged_df.dropna(subset=['tconst', 'primary_title', 'original_title'], in

# Filling missing values for start_year, runtime_minutes, averagerating,
merged_df['start_year'].fillna(merged_df['start_year'].median(), inplace=
merged_df['runtime_minutes'].fillna(merged_df['runtime_minutes'].median()
merged_df['averagerating'].fillna(merged_df['averagerating'].median(), in
merged_df['numvotes'].fillna(merged_df['numvotes'].median(), inplace=True

# Dropping rows with missing genres as imputing might not be accurate
merged_df.dropna(subset=['genres'], inplace=True)

# Filling missing studio, domestic_gross, foreign_gross, year with approp
merged_df['studio'].fillna('Unknown', inplace=True)
merged_df['domestic_gross'].fillna(0, inplace=True)
merged_df['foreign_gross'].fillna(0, inplace=True)
merged_df['year'].fillna(merged_df['year'].median(), inplace=True)

# Verify if there are any remaining missing values
print(merged_df.isnull().sum())
```

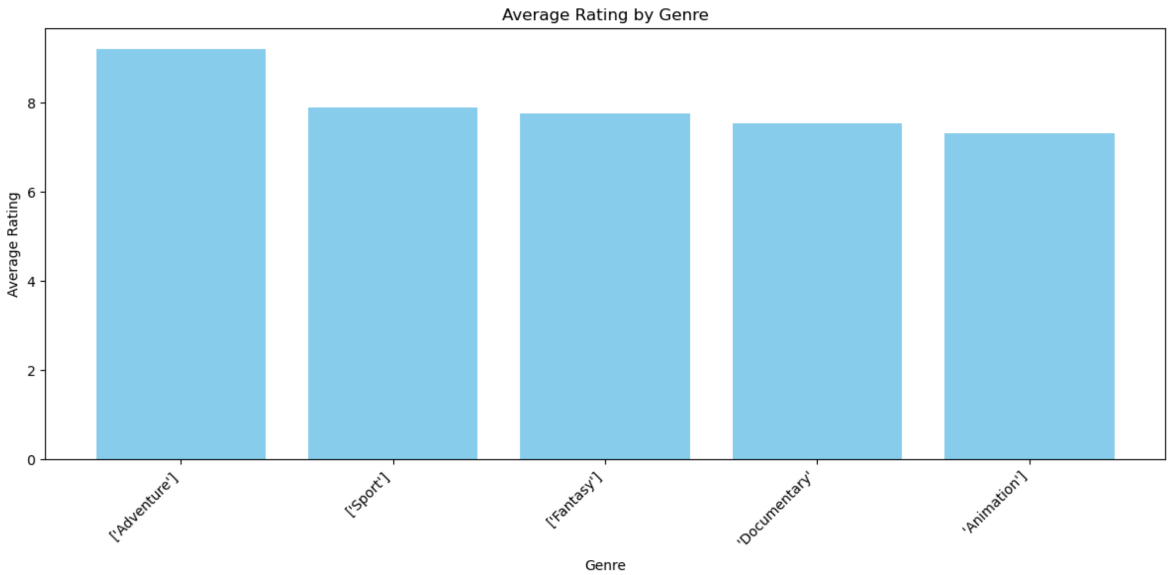
```
tconst             0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
genres             0
averagerating      0
numvotes           0
studio             0
domestic_gross     0
foreign_gross      0
year               0
dtype: int64
```

Data Modeling

Let's conduct exploratory analysis to uncover patterns, trends, and relationships in the data. As genre plays an important role in deciding what kind of movies would interest the target audience, I will start my analysis based on user ratings and box office performance, which will be depicted in the form of bar graphs and line graphs for clearer understanding of the distribution and associations between variables.

Rating Analysis for Genres

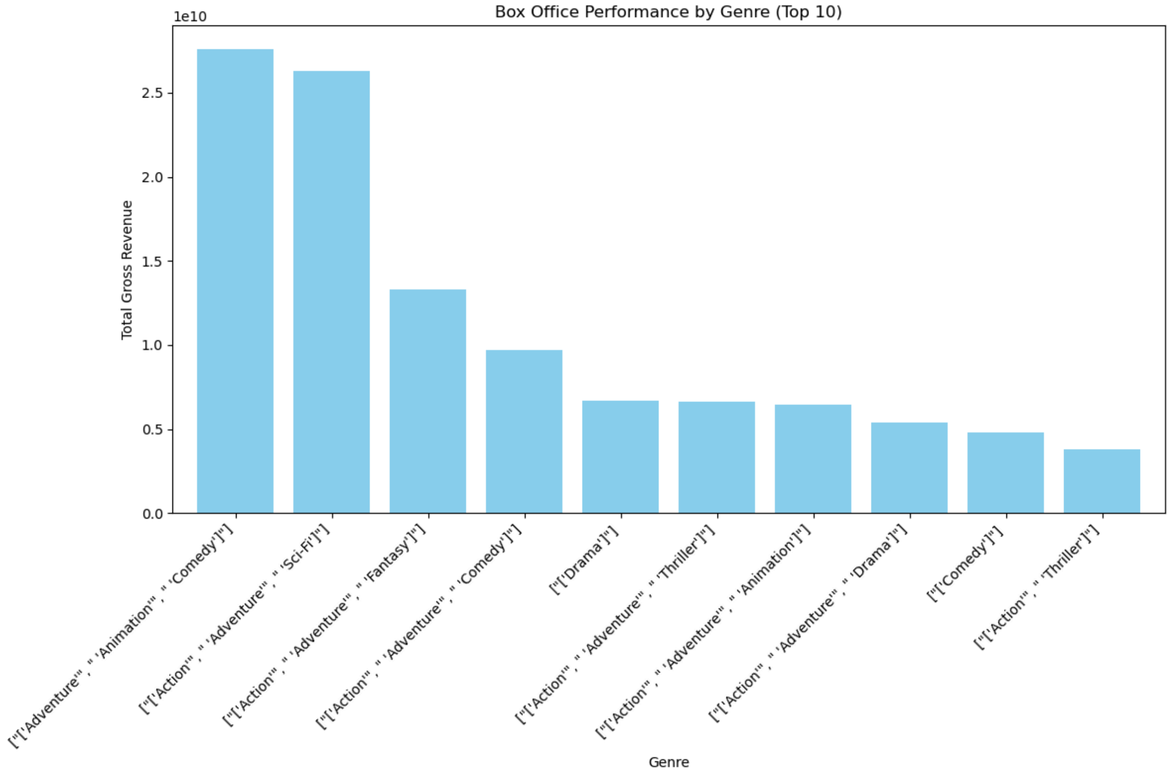
In [15]:
```
# Ensure all values in the 'genres' column are treated as strings
merged_df['genres'] = merged_df['genres'].astype(str)

# Split the 'genres' column into individual genres
merged_df['genres'] = merged_df['genres'].str.split(',')

# Explode the 'genres' column to create multiple rows for each movie-genr
genres_df = merged_df.explode('genres')

# Group the data by each individual genre and calculate the average ratin
genre_avg_rating = genres_df.groupby('genres')['averagerating'].mean().so

# Select the top 5 genres with the highest ratings
top_5_genres = genre_avg_rating.head(5)

# Remove duplicates from the index
top_5_genres = top_5_genres[~top_5_genres.index.duplicated(keep='first')]

# Plot the graph to visualize each genre separately on the x-axis
plt.figure(figsize=(12, 6))
plt.bar(top_5_genres.index, top_5_genres.values, color='skyblue')
plt.xlabel('Genre')
plt.ylabel('Average Rating')
plt.title('Average Rating by Genre')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
```



Our analysis of average user ratings suggests that Animation, Documentary, Fantasy, and Sport genres tend to have higher ratings. However, these ratings alone don't guarantee financial success.
Let's now delve into box office performance to gain a more comprehensive understanding of genre viability.

Box office analysis by checking proitability of each Genre

In [16]:
```
# Convert 'domestic_gross' and 'foreign_gross' columns to numeric
merged_df['domestic_gross'] = pd.to_numeric(merged_df['domestic_gross'],
merged_df['foreign_gross'] = pd.to_numeric(merged_df['foreign_gross'], er

# Calculate total gross revenue (domestic + foreign) by year
merged_df['total_gross'] = merged_df['domestic_gross'] + merged_df['forei

# Convert lists in the 'genres' column to strings
merged_df['genres'] = merged_df['genres'].astype(str)
# Explode the 'genres' column to create multiple rows for each movie-genr
genres_df = merged_df.explode('genres')

#Group the data by genre and sum the total gross revenue for each genre
genre_total_gross = merged_df.groupby('genres')['total_gross'].sum()
#Sort the genres based on total gross revenue
genre_total_gross_sorted = genre_total_gross.sort_values(ascending=False)

#Select the top N genres by total gross revenue
top_genres = genre_total_gross_sorted.head(10)

#Plot the graph with top genres
plt.figure(figsize=(12, 8))
top_genres.plot(kind='bar', color='skyblue', width=0.8)  # Adjust bar wid
plt.xlabel('Genre')
plt.ylabel('Total Gross Revenue')
plt.title('Box Office Performance by Genre (Top 10)')
plt.xticks(rotation=45, ha='right')  # Rotate genre labels
plt.tight_layout()
plt.show()
```



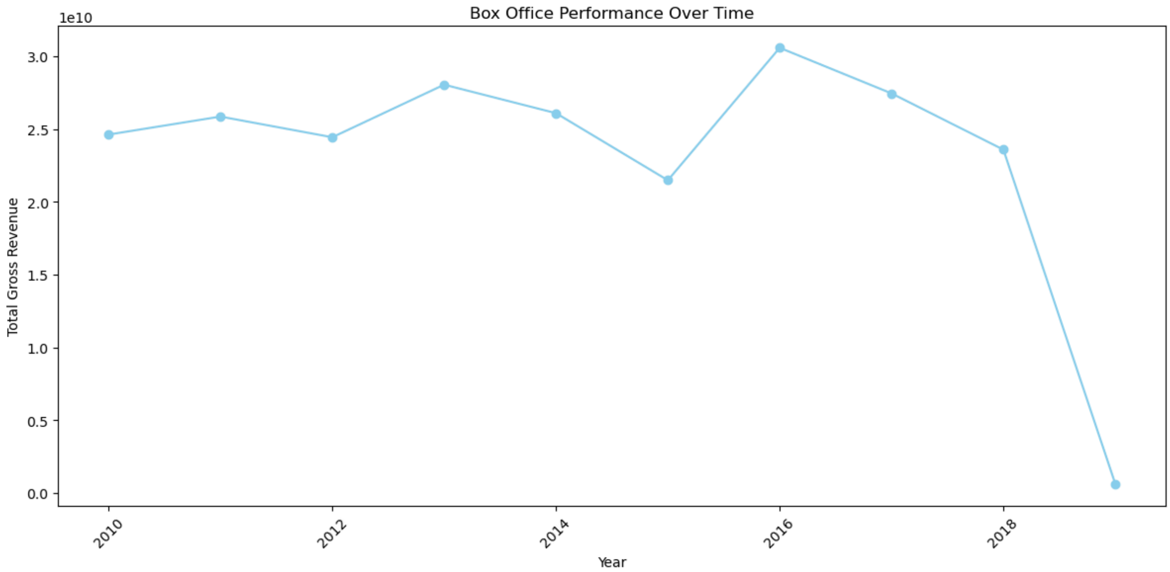
The genres grouped together under "Total Gross Revenue" exhibit the highest average box office performance, notably Action, Adventure, Comedy, and Animation. Conversely, Drama and Action Thriller genres demonstrate significantly lower average box office performance compared to the top genres.

Overall, this graph assists Microsoft in identifying genres typically associated with higher box office performance. This information serves as a valuable starting point for determining where to allocate resources, such as budget and talent, when producing movies. With this insight, Microsoft's movie studio can consider investing more resources in producing films within these high-grossing genre

Box office performance Over time

Next lets visualize the box office performance trends over time, we can create line plots to represent the total gross revenue for each year. We can also analyze the distribution of genres over time to identify any shifts or trends in audience preferences.

In [17]:
```
yearly_revenue = merged_df.groupby('start_year')['total_gross'].sum()

# Visualize trends in box office performance over time using a line plot
plt.figure(figsize=(12, 6))
yearly_revenue.plot(kind='line', marker='o', color='skyblue')
plt.xlabel('Year')
plt.ylabel('Total Gross Revenue')
plt.title('Box Office Performance Over Time')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



From the above trend, we can observe that the Box Office Performance was highest in the year 2016 compared to any other year. This indicates that the movies released in that year performed exceptionally well at the box office. This information could be crucial for Microsoft's movie studio when planning to invest in particular genres. Now, let's dig deeper and find out which genre of movies was released in 2016.

In [18]:
```
# Ensure all values in the 'genres' column are treated as strings
merged_df['genres'] = merged_df['genres'].astype(str)

# Explode the 'genres' column to create multiple rows for each movie-genr
exploded_df = merged_df.explode('genres')

# Filter the dataset to include only movies released in the year 2016
movies_2016 = exploded_df[exploded_df['year'] == 2016]

# Group the filtered dataset by genre and calculate the total revenue for
genre_revenue_2016 = movies_2016.groupby('genres')[['domestic_gross', 'fo

# Calculate the total revenue (sum of domestic and foreign gross) for eac
genre_revenue_2016['total_revenue'] = genre_revenue_2016['domestic_gross'

# Convert nested lists to strings
genre_revenue_2016.index = genre_revenue_2016.index.map(lambda x: ', '.jo

# Identify the genre with the highest total revenue in 2016
highest_revenue_genre_2016 = genre_revenue_2016['total_revenue'].idxmax()
highest_revenue = genre_revenue_2016['total_revenue'].max()

# Print the results
print("Genre with the highest total revenue in 2016:", highest_revenue_ge
print("Total revenue in 2016 for the highest revenue genre:", highest_rev
```

```
Genre with the highest total revenue in 2016: ['Adventure',  'Animation',
'Comedy']
Total revenue in 2016 for the highest revenue genre: 4668294999.0
```

From the above anlaysis we find the following observations:

Target Audience: Understanding which genres generated high revenue can help Microsoft in defining their target audience. They might consider producing content that appeals to similar demographics who enjoyed movies from the Adventure, Animation, Comedy genre in 2017.

Content Strategy: This insight could guide Microsoft in developing their content strategy. They might prioritize investing in movies belonging to the Adventure, Animation, Comedy genre or consider incorporating elements of these genres into their productions to increase their chances of success.

Evaluation

The analysis provides valuable insights into the relationship between movie genres, user ratings, and box office performance.

---

1. The analysis identifies action, adventure, comedy, and animation genres as the top performers in terms of box office revenue and audience ratings that can inform Microsoft's movie studio strategy.
2. The insights derived from the analysis can serve as a starting point for strategic decision-making based on the data available.

3. However, it's important to remember that the movie industry is always changing, so these recommendations might not always be accurate. It's essential to keep an eye on trends and adapt strategies accordingly.
4. Overall, by using this data-driven approach, Microsoft's movie studio can make smarter decisions about which movies to produce, potentially leading to greater success and profitability.

Conclusions

1. Recommendations for the Business:Focus resources on producing movies in genres like action, adventure, comedy, and animation, which have shown high box office performance and positive ratings.

Use a data-driven approach to guide decisions on resource allocation, talent acquisition, and content strategy.

2. Limitations of the Analysis:Movie success depends on factors beyond genre, like marketing, cast selection, and economic conditions, not accounted for in the analysis.

External factors such as competition and unforeseen events could influence movie performance.

3. Future Improvements: Incorporate additional data sources like release dates, casting informations and audience demographics.

Use predictive modeling to forecast box office performance for upcoming movies. Conduct sentiment analysis on audience reviews for deeper insights. Regularly update and refine the analysis to adapt to changing market conditions and preferences.

In [ ]: