index

May 31, 2024

1 MICROSOFT MOVIE STUDIO PROJECT ANALYSIS

Student Name: Amos Kipngetich Rotich

Student Pace: Part Time

Scheduled Project Review Date/time: June 3, 2024

Instructor Name: Winnie Anyoso

Blog Post URL: None

1.1 Introduction

1.1.1 1. Project Overview

Microsoft is has decided to venture into movie industry and wants to launch its own movie studio. To successfully do this, we need to look at the trends and factors contributing to the success of other movie studios.

1.1.2 2. Goals

1. Analyze movie data to uncover insights for decision making
2. Leverage data from several box offices
3. Identify patterns and attributes that correlate with box office performance

1.1.3 3. Data Sources

For purpose of this project we used data from: 1. Box Office Mojo - Provided information on box office revenue. 2. IMDB - Detailed information on movie basics and ratings.

1.1.4 4. Methodology

1. Data Collection - data from Box Office Mojo and IMDB used each providing different insights into movie performance.
2. Data Cleaning, Standardization and Integration - we cleaned the datasets, standardized and intergrated them ensuring that our analysis maintains the consistency and reliability required.
3. Exploratory Data Analysis (EDA) - we performed extensive EDA to uncover correlations, trends, and actionable insights.

1.1.5 5. Expected Results

By the end of this analysis, we will address: 1. Information on most profitable genres 2. Insights on genres, budgeting strategies and audince preferences 3. Conclusions to guide Microsoft's new venture into movie industry based on data analysis and clear visualizations.

1.2 Data Preparation

1.2.1 1. Extracting data and perform initial exploration of datatsets

[1]:
```
# Import relevant modules
import csv
import pandas as pd
import sqlite3
import os
```

[2]:
```
#Loading CSV Files
box_office_data = pd.read_csv("bom.movie_gross.csv.gz")
```

[3]:
```
#Loading SQLite Database
conn = sqlite3.connect("im.db")
```

[4]:
```
box_office_data.head()
```

[4]:
```
                                         title studio  domestic_gross  \
0                                  Toy Story 3     BV     415000000.0
1                   Alice in Wonderland (2010)     BV     334200000.0
2  Harry Potter and the Deathly Hallows Part 1     WB     296000000.0
3                                    Inception     WB     292600000.0
4                          Shrek Forever After   P/DW     238700000.0

  foreign_gross  year
0     652000000  2010
1     691300000  2010
2     664300000  2010
3     535700000  2010
4     513900000  2010
```

[5]:
```
table_list_query = "SELECT name FROM sqlite_master WHERE type='table';"
tables = pd.read_sql_query(table_list_query, conn)

print("Available tables:", tables)
```

```
Available tables:              name
0   movie_basics
1      directors
2      known_for
3     movie_akas
4  movie_ratings
```

```
5        persons
6     principals
7        writers
```

[6]:
```
movie_ratings = pd.read_sql("""SELECT * FROM movie_ratings;""", conn)
movie_basics = pd.read_sql("""SELECT * FROM movie_basics;""", conn)

conn.close()

print(movie_ratings.head())
print(movie_basics.head())
```

```
    movie_id  averagerating  numvotes
0  tt10356526            8.3        31
1  tt10384606            8.9       559
2   tt1042974            6.4        20
3   tt1043726            4.2     50352
4   tt1060240            6.5        21
    movie_id                    primary_title              original_title  \
0  tt0063540                        Sunghursh                   Sunghursh
1  tt0066787  One Day Before the Rainy Season             Ashad Ka Ek Din
2  tt0069049       The Other Side of the Wind  The Other Side of the Wind
3  tt0069204                  Sabse Bada Sukh             Sabse Bada Sukh
4  tt0100275         The Wandering Soap Opera       La Telenovela Errante

   start_year  runtime_minutes                genres
0        2013            175.0    Action,Crime,Drama
1        2019            114.0       Biography,Drama
2        2018            122.0                 Drama
3        2018              NaN          Comedy,Drama
4        2017             80.0  Comedy,Drama,Fantasy
```

1.2.2 2. Data Aggregation and Cleaning

1. Combine movie_basics and movie_ratings, then merge output with with the box_office_data
2. Handle missing values - drop unnecessary columns and fill missing values in key columns
3. Standardize genre labels

[7]:
```
# merge IMDB data
imdb_data = pd.merge(movie_basics, movie_ratings, on='movie_id', how='left')

print(imdb_data.head())
```

```
    movie_id                    primary_title              original_title  \
0  tt0063540                        Sunghursh                   Sunghursh
1  tt0066787  One Day Before the Rainy Season             Ashad Ka Ek Din
2  tt0069049       The Other Side of the Wind  The Other Side of the Wind
3  tt0069204                  Sabse Bada Sukh             Sabse Bada Sukh
4  tt0100275         The Wandering Soap Opera       La Telenovela Errante
```

```
   start_year  runtime_minutes                genres  averagerating  numvotes
0        2013            175.0    Action,Crime,Drama            7.0      77.0
1        2019            114.0       Biography,Drama            7.2      43.0
2        2018            122.0                 Drama            6.9    4517.0
3        2018              NaN          Comedy,Drama            6.1      13.0
4        2017             80.0  Comedy,Drama,Fantasy            6.5     119.0
```

[8]:
```
# Merge imdb merging output to the box_office_data
merged_data = pd.merge(box_office_data, imdb_data, left_on='title',␣
↪right_on='primary_title', how='left')
print(merged_data)
```

```
                                             title      studio  domestic_gross  \
0                                      Toy Story 3          BV     415000000.0
1                       Alice in Wonderland (2010)          BV     334200000.0
2     Harry Potter and the Deathly Hallows Part 1          WB     296000000.0
3                                        Inception          WB     292600000.0
4                              Shrek Forever After        P/DW     238700000.0
…                                                …           …               …
4142                                     The Quake       Magn.          6200.0
4143                 Edward II (2018 re-release)          FM          4800.0
4144                                      El Pacto        Sony          2500.0
4145                                      The Swan  Synergetic          2400.0
4146                              An Actor Prepares       Grav.          1700.0

     foreign_gross  year   movie_id        primary_title       original_title  \
0        652000000  2010  tt0435761          Toy Story 3          Toy Story 3
1        691300000  2010        NaN                  NaN                  NaN
2        664300000  2010        NaN                  NaN                  NaN
3        535700000  2010  tt1375666            Inception            Inception
4        513900000  2010  tt0892791  Shrek Forever After  Shrek Forever After
…              …    …          …                    …                    …
4142           NaN  2018  tt6523720            The Quake             Skjelvet
4143           NaN  2018        NaN                  NaN                  NaN
4144           NaN  2018        NaN                  NaN                  NaN
4145           NaN  2018        NaN                  NaN                  NaN
4146           NaN  2018  tt5718046    An Actor Prepares    An Actor Prepares

      start_year  runtime_minutes                      genres  averagerating  \
0         2010.0            103.0  Adventure,Animation,Comedy            8.3
1            NaN              NaN                         NaN            NaN
2            NaN              NaN                         NaN            NaN
3         2010.0            148.0     Action,Adventure,Sci-Fi            8.8
4         2010.0             93.0  Adventure,Animation,Comedy            6.3
…            …                …                           …              …
4142      2018.0            106.0       Action,Drama,Thriller            6.2
4143         NaN              NaN                         NaN            NaN
```

```
4144          NaN              NaN                          NaN              NaN
4145          NaN              NaN                          NaN              NaN
4146       2018.0             97.0                       Comedy              5.0

       numvotes
0      682218.0
1           NaN
2           NaN
3     1841066.0
4      167532.0
…          …
4142     5270.0
4143        NaN
4144        NaN
4145        NaN
4146      388.0

[4147 rows x 13 columns]
```

[9]:
```
# Check missing values in merged data
print(merged_data.isna().sum())
```

```
title                 0
studio                5
domestic_gross       35
foreign_gross      1631
year                  0
movie_id            781
primary_title       781
original_title      781
start_year          781
runtime_minutes     949
genres              821
averagerating      1120
numvotes           1120
dtype: int64
```

[10]:
```
# Drop columns not necessary to our analysis
merged_data.drop(columns=['original_title', 'start_year', 'numvotes'],␣
↪inplace=True)
```

[11]:
```
#Handle Missing values in key columns

# filling missing studio values with 'Unknown'
merged_data['studio'].fillna('Unknown', inplace=True)

# filling missing domestic gross and foreign gross with '0'
merged_data['domestic_gross'].fillna(0, inplace=True)
```

```
merged_data['foreign_gross'].fillna(0, inplace=True)

# finding median runtime and replacing missing runtimes
median_runtime = merged_data['runtime_minutes'].median()
merged_data['runtime_minutes'].fillna(median_runtime, inplace=True)

# filling missing genres with 'Unknown'
merged_data['genres'].fillna('Unknown', inplace=True)

# finding mean rating and replace to missing average ratings
mean_rating = merged_data['averagerating'].mean()
merged_data['averagerating'].fillna(mean_rating, inplace=True)

# dropping rows that do not have movie id
merged_data.dropna(subset=['movie_id'], inplace=True)

# check if there are further missing values
print(merged_data.isna().sum())
```

```
title              0
studio             0
domestic_gross     0
foreign_gross      0
year               0
movie_id           0
primary_title      0
runtime_minutes    0
genres             0
averagerating      0
dtype: int64
```

[12]:
```
#Check if data is clean before proceeding to analysis
print(merged_data.head())
```

```
                        title studio  domestic_gross foreign_gross  year  \
0                 Toy Story 3     BV     415000000.0     652000000  2010
3                   Inception     WB     292600000.0     535700000  2010
4         Shrek Forever After   P/DW     238700000.0     513900000  2010
5  The Twilight Saga: Eclipse   Sum.     300500000.0     398000000  2010
6                  Iron Man 2   Par.     312400000.0     311500000  2010

    movie_id               primary_title  runtime_minutes  \
0  tt0435761                 Toy Story 3            103.0
3  tt1375666                   Inception            148.0
4  tt0892791         Shrek Forever After             93.0
5  tt1325004  The Twilight Saga: Eclipse            124.0
6  tt1228705                  Iron Man 2            124.0
```

```
                     genres  averagerating
0  Adventure,Animation,Comedy            8.3
3     Action,Adventure,Sci-Fi            8.8
4  Adventure,Animation,Comedy            6.3
5     Adventure,Drama,Fantasy            5.0
6     Action,Adventure,Sci-Fi            7.0
```

[13]:
```
# Standardize genre column
def clean_and_explode(df, genre_column):
    df[genre_column] = df[genre_column].str.split(',')
    df = df.explode(genre_column).reset_index(drop=True)
    return df

merged_data = clean_and_explode(merged_data, 'genres')

merged_data['averagerating'] = merged_data['averagerating'].
 ↪fillna(merged_data['averagerating'].mean())

print(merged_data.head())
```

```
        title studio  domestic_gross foreign_gross  year   movie_id  \
0  Toy Story 3     BV     415000000.0     652000000  2010  tt0435761
1  Toy Story 3     BV     415000000.0     652000000  2010  tt0435761
2  Toy Story 3     BV     415000000.0     652000000  2010  tt0435761
3    Inception     WB     292600000.0     535700000  2010  tt1375666
4    Inception     WB     292600000.0     535700000  2010  tt1375666

  primary_title  runtime_minutes     genres  averagerating
0   Toy Story 3            103.0  Adventure            8.3
1   Toy Story 3            103.0  Animation            8.3
2   Toy Story 3            103.0     Comedy            8.3
3     Inception            148.0     Action            8.8
4     Inception            148.0  Adventure            8.8
```

1.3 Data Analysis

We have our data and we need to perform analysis to identifying the best-performing genre. Ensure that financial data(domestic and foreign gross) are numeric values for easier analysis.

[14]:
```
# Ensuring that revenue data is captured as int or float for analysis
merged_data['domestic_gross'] = pd.to_numeric(merged_data['domestic_gross'],␣
 ↪errors='coerce').fillna(0).astype(int)
merged_data['foreign_gross'] = pd.to_numeric(merged_data['foreign_gross'],␣
 ↪errors='coerce').fillna(0).astype(int)
```

[15]:
```
# Aggregated Domestic and Foreign Revenue by Genre
genre_revenue = merged_data.groupby('genres')[['domestic_gross',␣
 ↪'foreign_gross', 'averagerating']].agg({
```

```python
    'domestic_gross': ['sum', 'mean'],
    'foreign_gross': ['sum', 'mean'],
    'averagerating': 'mean'
}).reset_index()

# rename main columns for easier identity
genre_revenue.columns = ['genre', 'total_domestic_gross',␣
↪'average_domestic_gross', 'total_foreign_gross', 'average_foreign_gross',␣
↪'average_rating']

# Sort by total domestic gross revenue
genre_revenue = genre_revenue.sort_values(by='total_domestic_gross',␣
↪ascending=False)

print(genre_revenue)
```

```
          genre  total_domestic_gross  average_domestic_gross  \
1     Adventure          4.191778e+10            9.398605e+07
0        Action          3.843915e+10            5.789028e+07
4        Comedy          3.249809e+10            3.367678e+07
7         Drama          3.105158e+10            1.655201e+07
17       Sci-Fi          1.495762e+10            1.076088e+08
19     Thriller          1.367092e+10            2.854054e+07
2     Animation          1.362289e+10            8.676997e+07
5         Crime          9.352542e+09            2.398088e+07
9       Fantasy          9.288773e+09            5.247895e+07
16      Romance          7.331809e+09            1.517973e+07
11       Horror          7.088680e+09            2.715969e+07
3     Biography          6.420383e+09            2.098164e+07
8        Family          5.597358e+09            4.339037e+07
6   Documentary          5.443313e+09            1.629734e+07
14      Mystery          4.974365e+09            2.250844e+07
10      History          2.943172e+09            1.975284e+07
18        Sport          2.122595e+09            3.723851e+07
20      Unknown          1.726364e+09            4.315910e+07
12        Music          1.697182e+09            1.731819e+07
13      Musical          5.508563e+08            2.899244e+07
22      Western          5.294837e+08            2.406744e+07
21          War          2.814003e+08            5.309440e+06
15         News          2.184540e+07            3.640900e+06

    total_foreign_gross  average_foreign_gross  average_rating
1          7.804942e+10           1.749987e+08        6.478034
0          6.900419e+10           1.039220e+08        6.280175
4          4.639226e+10           4.807488e+07        6.256110
7          4.262900e+10           2.272335e+07        6.580922
17         2.368079e+10           1.703654e+08        6.451297
```

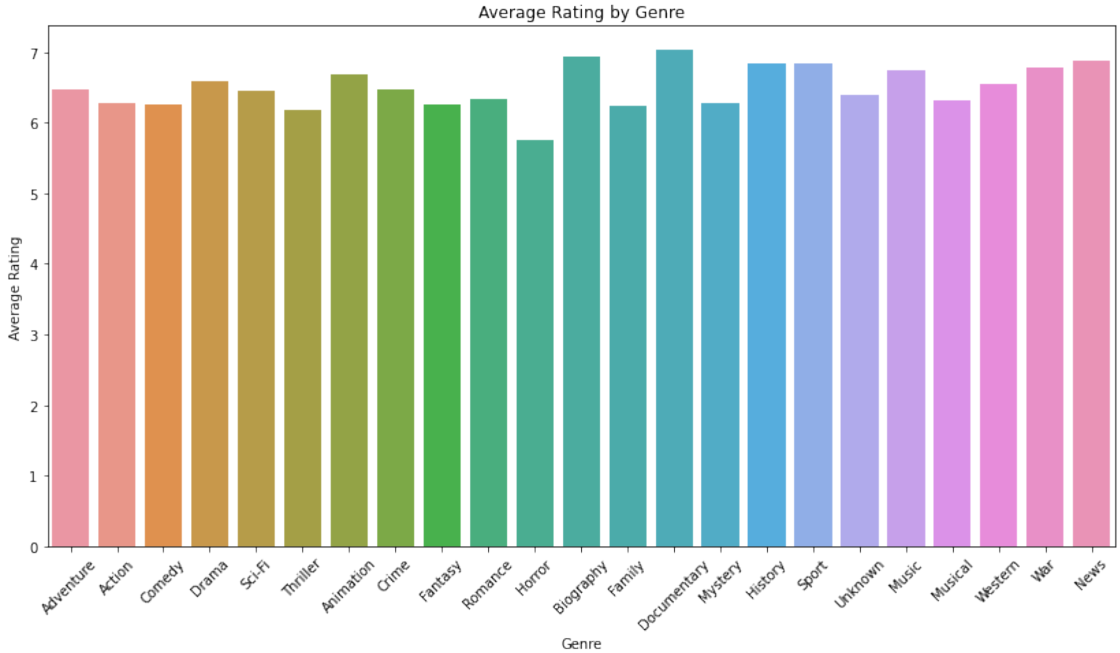
19	2.182380e+10	4.556117e+07	6.188094
2	2.587052e+10	1.647804e+08	6.692280
5	1.012734e+10	2.596755e+07	6.479130
9	1.861195e+10	1.051523e+08	6.250865
16	9.261870e+09	1.917571e+07	6.339262
11	9.251392e+09	3.544594e+07	5.746779
3	7.633332e+09	2.494553e+07	6.937939
8	8.359439e+09	6.480185e+07	6.246442
6	6.099421e+09	1.826174e+07	7.025034
14	7.247272e+09	3.279308e+07	6.286453
10	3.742451e+09	2.511712e+07	6.841937
18	2.318602e+09	4.067724e+07	6.839129
20	2.688589e+09	6.721472e+07	6.385005
12	2.191535e+09	2.236260e+07	6.738219
13	7.411853e+08	3.900975e+07	6.324083
22	7.018230e+08	3.190105e+07	6.557163
21	5.830160e+08	1.100030e+07	6.788965
15	4.800000e+07	8.000000e+06	6.885861

1.4 Data Visualization

[16]:
```python
# Import plotting modules
import seaborn as sns
import matplotlib.pyplot as plt
```

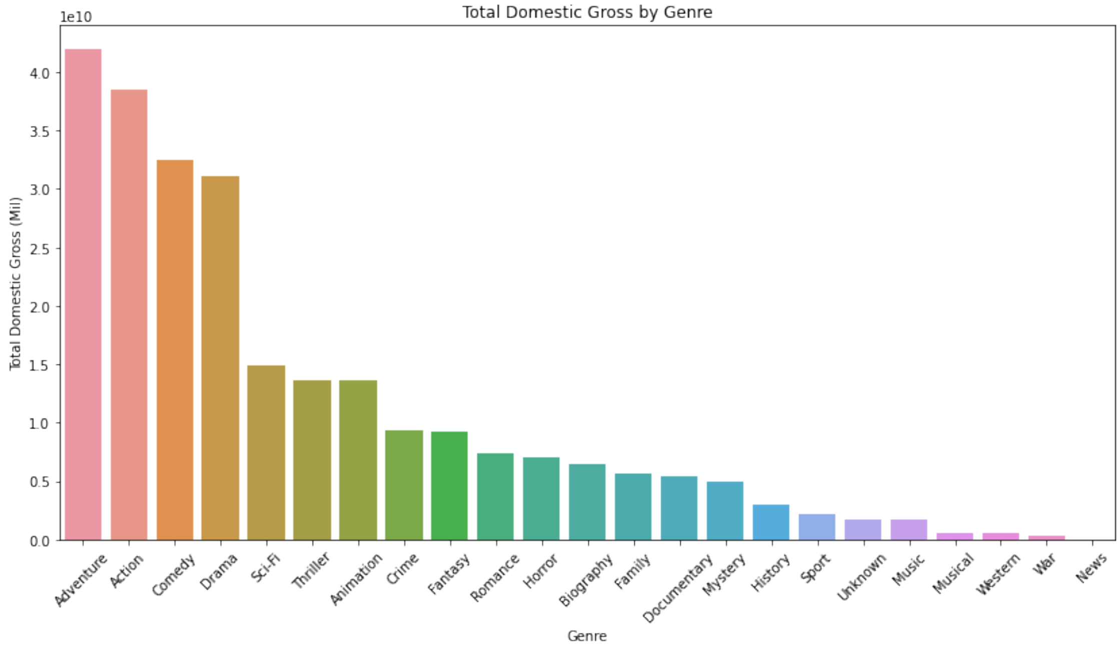
1.4.1 1. Best-rated genre

[17]:
```python
plt.figure(figsize=(14, 7))
sns.barplot(x='genre', y='average_rating', data=genre_revenue)
plt.title('Average Rating by Genre')
plt.xlabel('Genre')
plt.ylabel('Average Rating')
plt.xticks(rotation=45)
plt.show()
```



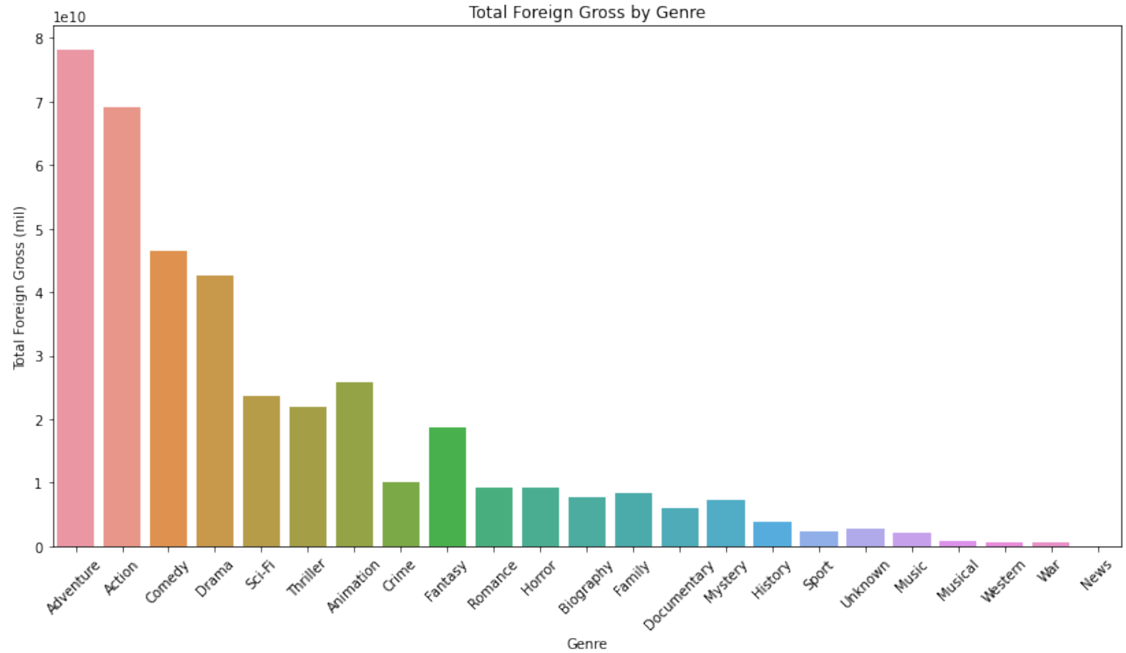
1.4.2 2. Genre income from the domestic market

[18]:
```
plt.figure(figsize=(14, 7))
sns.barplot(x='genre', y='total_domestic_gross', data=genre_revenue)
plt.title('Total Domestic Gross by Genre')
plt.xlabel('Genre')
plt.ylabel('Total Domestic Gross (Mil)')
plt.xticks(rotation=45)
plt.show()
```



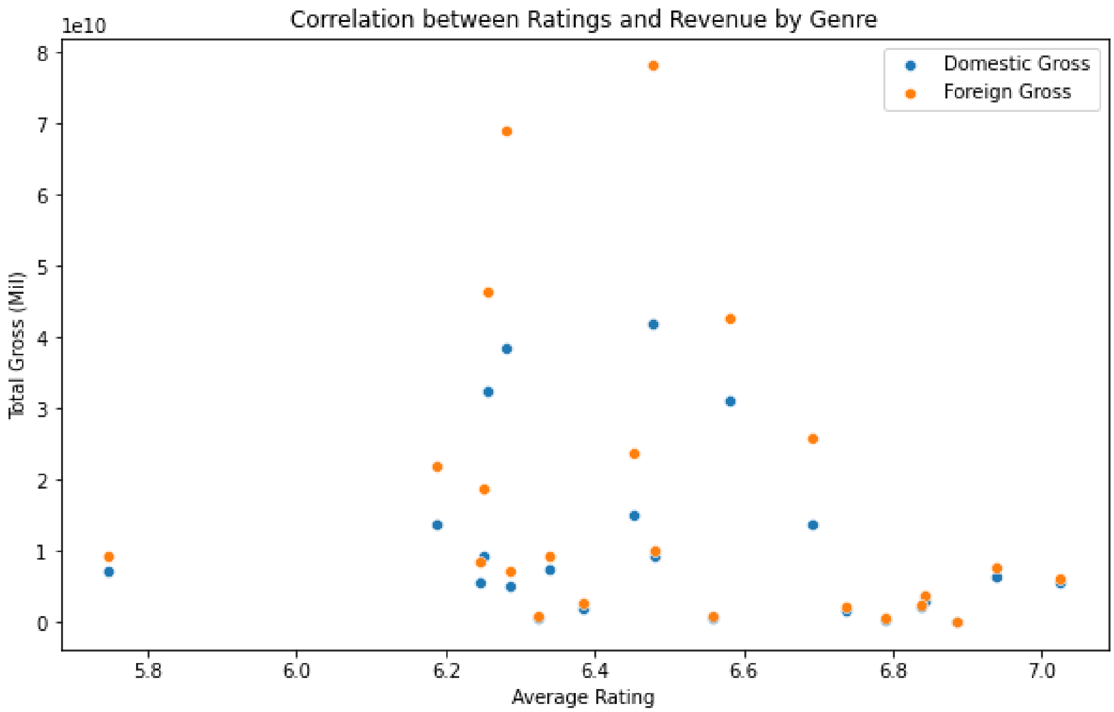
1.4.3 3. Genre income from the international market

[19]:
```
plt.figure(figsize=(14, 7))
sns.barplot(x='genre', y='total_foreign_gross', data=genre_revenue)
plt.title('Total Foreign Gross by Genre')
plt.xlabel('Genre')
plt.ylabel('Total Foreign Gross (mil)')
plt.xticks(rotation=45)
plt.show()
```



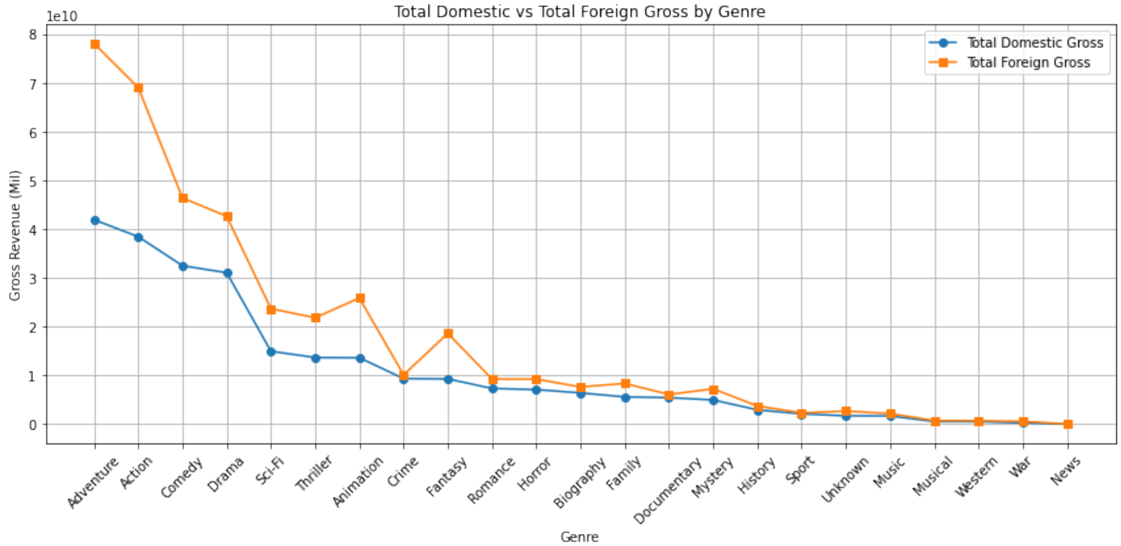
1.4.4 4. Correlation between ratings and revenue by genre

[20]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='average_rating', y='total_domestic_gross',␣
 ↪data=genre_revenue, label='Domestic Gross')
sns.scatterplot(x='average_rating', y='total_foreign_gross',␣
 ↪data=genre_revenue, label='Foreign Gross')
plt.title('Correlation between Ratings and Revenue by Genre')
plt.xlabel('Average Rating')
plt.ylabel('Total Gross (Mil)')
plt.legend()
plt.show()
```



1.4.5 5. Comparison of revenue generated in the domestic and foreign markets

[21]:
```
plt.figure(figsize=(12, 6))
plt.plot(genre_revenue['genre'], genre_revenue['total_domestic_gross'],
 ↪marker='o', label='Total Domestic Gross')
plt.plot(genre_revenue['genre'], genre_revenue['total_foreign_gross'],
 ↪marker='s', label='Total Foreign Gross')
plt.title('Total Domestic vs Total Foreign Gross by Genre')
plt.xlabel('Genre')
plt.ylabel('Gross Revenue (Mil)')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



1.5 Insights and Business Recommendations

From the analysis, this is what we recommend to Microsoft:

1. Focus on top-performing genres Action, Adventure and Animation genres have shown strong performance on the locan and international markets.

2. Invest in the highly rated genres Microsoft should build on genres where higher ratings strongly correlate with it success. Animation, Biography and Drama genres indicates that positive reviews significantly boosts its revenue.

3. Target global market Microsoft should consider investing in the global audience. Our analysis indicates that almost every genre does well in the foreign market compared to the domestic market.