# Project-1

December 10, 2023

## 1 MOVIE INDUSTRY EXPLORATION

Movie production can be a lucrative and glamorous business, and yet highly risky. It is therefore important to choose moves into the movie industry wisely and with precaution. The movie genres range from drama, comedy, action, adventure, horror, sciFi, and many more. Sometimes there is a thin line between movies that we are unable to distinguish one genre from the other. In such instances, we find that we can have romance/comedy or comedy/drama or even action/comedy. The combinations are limitless.

### 1.1 Business Understanding

The goal of this project is to provide microsoft with information about the best movies to invest in.

### 1.2 Objectives

1. Determine which movie genres are doing well based on popularity(frequency)
2. Find out which movies genres have the highest rating
3. Find out which movies generate most revenue based on their box office (domestic_gross + foreign_gross)

### 1.3 Data understanding

The data used in this project comes from movie sites: 1. Box Office 2. IMDBLinks 3. Rotten Tomatoes 4. TheMovieDB 5. The Numbers

The data this project focuses on, is contained in the following tables: 1. bom.movie_gross.csv 2. movie_basics 3. movie_ratings 4. tmdb.movies.csv

The project will focus on data about movie ratings, box office and movie genres.

### 1.4 Data Analysis

The process of data analysis includes data cleaning and analysis, and finally visualization in form of graphs or bar charts.

[1]:
```
# Importing relevant modules to assist in data cleaning, analysis and␣
 ↪visualization.
import csv
import pandas as pd
import numpy as np
import sqlite3
conn = sqlite3.connect('im.db')
cursor = conn.cursor()
import matplotlib.pyplot as plt
%matplotlib inline
```

### 1.4.1 Table 1: bom.movie_gross

[2]:
```
df1 = pd.read_csv('bom.movie_gross.csv')
df1
```

[2]:
```
                                            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
…                                               …           …               …
3382                                    The Quake       Magn.          6200.0
3383                  Edward II (2018 re-release)          FM          4800.0
3384                                     El Pacto        Sony          2500.0
3385                                     The Swan   Synergetic          2400.0
3386                              An Actor Prepares       Grav.          1700.0

     foreign_gross  year
0        652000000  2010
1        691300000  2010
2        664300000  2010
3        535700000  2010
4        513900000  2010
…              …     …
3382           NaN  2018
3383           NaN  2018
3384           NaN  2018
3385           NaN  2018
3386           NaN  2018

[3387 rows x 5 columns]
```

[3]:
```
# checking for duplicates
df1.duplicated().value_counts()
```

[3]:
```
False    3387
Name: count, dtype: int64
```

```
[4]: # checking for missing data and resolving
     df1.isna().sum()
```

```
[4]: title                 0
     studio                5
     domestic_gross       28
     foreign_gross      1350
     year                  0
     dtype: int64
```

```
[5]: df1['domestic_gross'].describe()
```

```
[5]: count    3.359000e+03
     mean     2.874585e+07
     std      6.698250e+07
     min      1.000000e+02
     25%      1.200000e+05
     50%      1.400000e+06
     75%      2.790000e+07
     max      9.367000e+08
     Name: domestic_gross, dtype: float64
```

```
[6]: # Converting string to float so as to perform statistics on the column values
     df1['domestic_gross'] = df1['domestic_gross'].replace(',', '', regex=True).
      ↪astype(float)
     # Calculating the median
     mode_value = df1['foreign_gross'].mode()
     # Replacing missing values with median
     df1['domestic_gross'].fillna('mode_value'[0], inplace=True)
     # Rechecking for missing values
     df1['domestic_gross'].isna().sum()
```

```
[6]: 0
```

```
[7]: df1['foreign_gross'].mode()
```

```
[7]: 0    1200000
     Name: foreign_gross, dtype: object
```

```
[8]: df1['foreign_gross'].describe()
```

```
[8]: count        2037
     unique       1204
     top       1200000
     freq           23
     Name: foreign_gross, dtype: object
```

[9]:
```
df1['foreign_gross'] = df1['foreign_gross'].replace(',', '', regex=True).
  ↪astype(float)
# Calculating the mode
median_value = df1['foreign_gross'].median()
# Filling missing values with the mode value
df1['foreign_gross'].fillna(median_value, inplace=True)
# Checkings the count of missing values after replacement
df1['foreign_gross'].isna().sum()
```

[9]:
```
0
```

[10]:
```
#Confirming NaN values have been replaced
df1.isna().sum()
```

[10]:
```
title             0
studio            5
domestic_gross    0
foreign_gross     0
year              0
dtype: int64
```

[11]:
```
df1['domestic_gross'] = pd.to_numeric(df1['domestic_gross'], errors='coerce')
df1['foreign_gross'] = pd.to_numeric(df1['foreign_gross'], errors='coerce')

# Summing 'domestic_gross' and 'foreign_gross' to create a new column␣
  ↪'box_office'
df1['box_office'] = df1['domestic_gross'] + df1['foreign_gross']

# Display the updated DataFrame
print(df1)
```

```
                                             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
…                                                …           …               …
3382                                     The Quake       Magn.          6200.0
3383                   Edward II (2018 re-release)          FM          4800.0
3384                                      El Pacto        Sony          2500.0
3385                                      The Swan  Synergetic          2400.0
3386                           An Actor Prepares       Grav.          1700.0

      foreign_gross  year    box_office
0       652000000.0  2010  1.067000e+09
1       691300000.0  2010  1.025500e+09
2       664300000.0  2010  9.603000e+08
```

```
3       535700000.0  2010  8.283000e+08
4       513900000.0  2010  7.526000e+08
…               …    …             …
3382     18700000.0  2018  1.870620e+07
3383     18700000.0  2018  1.870480e+07
3384     18700000.0  2018  1.870250e+07
3385     18700000.0  2018  1.870240e+07
3386     18700000.0  2018  1.870170e+07

[3387 rows x 6 columns]
```

[12]:
```
# Combining bom_movie with tmdb.movies to access genre column from tmdb
df2 = pd.read_csv('tmdb.movies.csv')
merged_df = pd.merge(df1, df2, on='title', how='inner')
merged_df.head(10)
```

[12]:
```
                                               title studio  domestic_gross  \
0                                        Toy Story 3     BV     415000000.0
1                                          Inception     WB     292600000.0
2                                Shrek Forever After   P/DW     238700000.0
3                         The Twilight Saga: Eclipse   Sum.     300500000.0
4                                         Iron Man 2   Par.     312400000.0
5                                            Tangled     BV     200800000.0
6                                       Despicable Me   Uni.     251500000.0
7                             How to Train Your Dragon   P/DW     217600000.0
8  The Chronicles of Narnia: The Voyage of the Da…    Fox     104400000.0
9                                   The King's Speech  Wein.     135500000.0

   foreign_gross  year   box_office  Unnamed: 0                genre_ids  \
0    652000000.0  2010  1.067000e+09           7         [16, 10751, 35]
1    535700000.0  2010  8.283000e+08           4           [28, 878, 12]
2    513900000.0  2010  7.526000e+08          38  [35, 12, 14, 16, 10751]
3    398000000.0  2010  6.985000e+08          15      [12, 14, 18, 10749]
4    311500000.0  2010  6.239000e+08           2           [12, 28, 878]
5    391000000.0  2010  5.918000e+08          13             [16, 10751]
6    291600000.0  2010  5.431000e+08           8         [16, 10751, 35]
7    277300000.0  2010  4.949000e+08           1      [14, 12, 16, 10751]
8    311300000.0  2010  4.157000e+08          22         [12, 10751, 14]
9    275400000.0  2010  4.109000e+08          25                [18, 36]

      id original_language                                   original_title  \
0  10193                en                                      Toy Story 3
1  27205                en                                        Inception
2  10192                en                              Shrek Forever After
3  24021                en                       The Twilight Saga: Eclipse
4  10138                en                                       Iron Man 2
5  38757                en                                          Tangled
```

```
6   20352             en                                          Despicable Me
7   10191             en                                 How to Train Your Dragon
8   10140             en    The Chronicles of Narnia: The Voyage of the Da…
9   45269             en                                        The King's Speech

   popularity release_date  vote_average  vote_count
0      24.445   2010-06-17           7.7        8340
1      27.920   2010-07-16           8.3       22186
2      15.041   2010-05-16           6.1        3843
3      20.340   2010-06-23           6.0        4909
4      28.515   2010-05-07           6.8       12368
5      21.511   2010-11-24           7.5        6407
6      23.673   2010-07-09           7.2       10057
7      28.734   2010-03-26           7.7        7610
8      17.382   2010-12-10           6.3        3196
9      16.798   2010-09-06           7.7        5013
```

[13]:

```
# Replacing genre_ids with genre name for easier interpretaion
# Values to replace genre_ids with name are derived from 'https://www.
↪themoviedb.org/talk/5daf6eb0ae36680011d7e6ee'
replacement_dict = {
    '28': 'action',
    '12': 'adventure',
    '878': 'scifi',
    '14': 'fantasy',
    '35': 'comedy',
    '18': 'drama',
    '16': 'animation',
    '80': 'crime',
    '99': 'documentary',
    '10751': 'family',
    '36': 'history',
    '27': 'horror',
    '10749': 'music',
    '9648': 'mystery',
    '10770': 'TV movie',
    '53': 'thriller',
    '10752': 'war',
    '37': 'western'
}
for old_value, new_value in replacement_dict.items():
    merged_df['genre_ids'] = merged_df['genre_ids'].replace(old_value,␣
↪new_value, regex=True)

merged_df['box_office'] = pd.to_numeric(merged_df['box_office'],␣
↪errors='coerce')
merged_df = merged_df.dropna(subset=['box_office'])
```

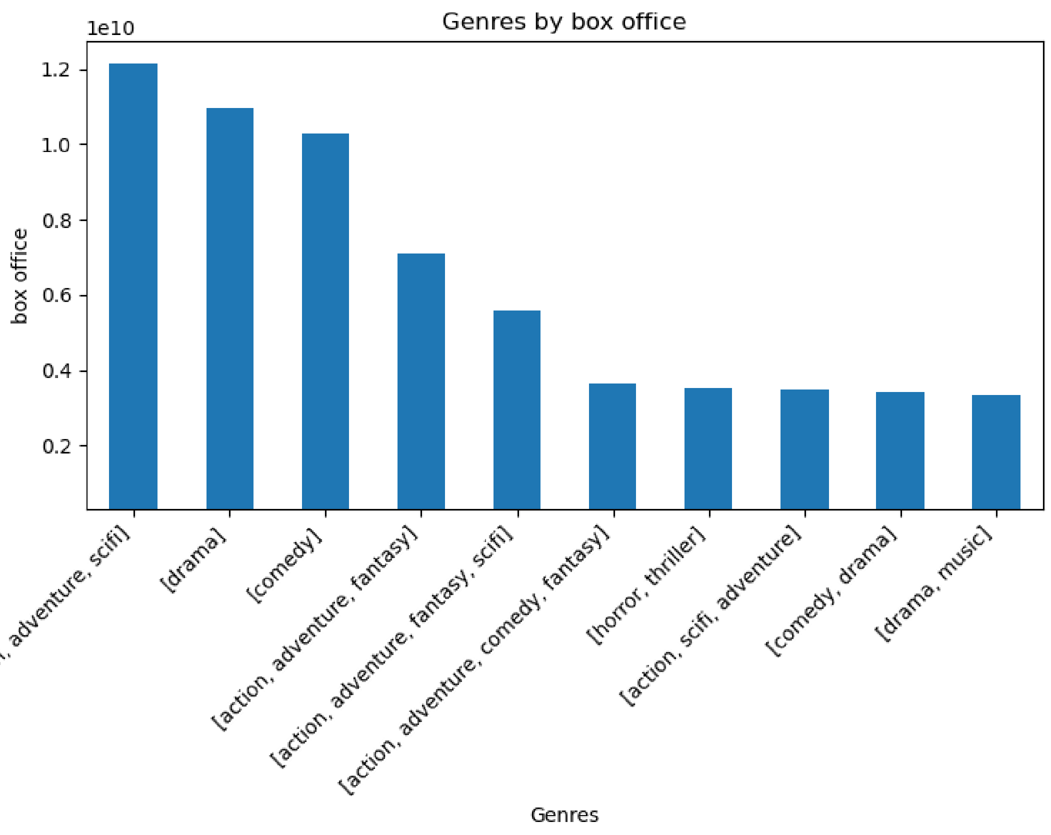
```
genre_gross_sum = merged_df.groupby('genre_ids')['box_office'].sum()

sorted_genres = genre_gross_sum.sort_values(ascending=False)
top_genres = sorted_genres.head(10)
top_genres

merged_df[['genre_ids', 'box_office']].head()
```

[13]:

```
                                          genre_ids    box_office
0                       [animation, family, comedy]  1.067000e+09
1                        [action, scifi, adventure]  8.283000e+08
2  [comedy, adventure, fantasy, animation, family]  7.526000e+08
3                [adventure, fantasy, drama, music]  6.985000e+08
4                        [adventure, action, scifi]  6.239000e+08
```

[14]:

```
plt.figure(figsize=(8, 6))
top_genres.plot(kind='bar')
plt.xlabel('Genres')
plt.ylabel('box office')
plt.title('Genres by box office')
plt.xticks(rotation=45 ,ha='right')
plt.ylim(bottom=300000000)
plt.tight_layout()
plt.show()
```



### 1.5 Conclusions from genres by box office

The top 5 genre grouping in box_office are: 1. Action, adventure, scifi
2. comedy 3. drama
4. action, adventure, fantasy 5. action, adventure, fantasy, scifi.

## 2 Table 2: movie_basics

[15]:
```
#Previewing available tables from the database
table_name_query = """SELECT name
                      AS 'Table Names'
                      FROM sqlite_master
                      WHERE type='table';"""

pd.read_sql(table_name_query, conn)
```

[15]:
```
     Table Names
0   movie_basics
1      directors
2      known_for
```

```
3       movie_akas
4     movie_ratings
5          persons
6       principals
7          writers
8     joined_mb_mr
9   new_table_name
10      table_name
11       new_table
```

[16]:
```
#Previewing data from movie_basics
first_query = """
SELECT *
FROM movie_basics
;
"""
pd.read_sql(first_query, conn)
```

[16]:
```
         movie_id                                primary_title  \
0       tt0063540                                    Sunghursh
1       tt0066787              One Day Before the Rainy Season
2       tt0069049                   The Other Side of the Wind
3       tt0069204                              Sabse Bada Sukh
4       tt0100275                     The Wandering Soap Opera
…             …                                            …
146139  tt9916538                          Kuambil Lagi Hatiku
146140  tt9916622  Rodolpho Teóphilo - O Legado de um Pioneiro
146141  tt9916706                              Dankyavar Danka
146142  tt9916730                                       6 Gunn
146143  tt9916754               Chico Albuquerque - Revelações

                                     original_title  start_year  \
0                                         Sunghursh        2013
1                                   Ashad Ka Ek Din        2019
2                        The Other Side of the Wind        2018
3                                   Sabse Bada Sukh        2018
4                             La Telenovela Errante        2017
…                                                 …         …
146139                          Kuambil Lagi Hatiku        2019
146140  Rodolpho Teóphilo - O Legado de um Pioneiro        2015
146141                              Dankyavar Danka        2013
146142                                       6 Gunn        2017
146143               Chico Albuquerque - Revelações        2013

        runtime_minutes                genres
0                 175.0    Action,Crime,Drama
1                 114.0       Biography,Drama
```

```
2                  122.0                  Drama
3                    NaN           Comedy,Drama
4                   80.0   Comedy,Drama,Fantasy
…                    …                    …
146139             123.0                  Drama
146140               NaN            Documentary
146141               NaN                 Comedy
146142             116.0            Documentary
146143               NaN            Documentary

[146144 rows x 6 columns]
```

[17]:
```
#Checking for null values
second_query = """
SELECT genres, COUNT(*)
FROM movie_basics
WHERE genres IS NULL
;
"""
pd.read_sql(second_query, conn)
# Missing values were 5408
```

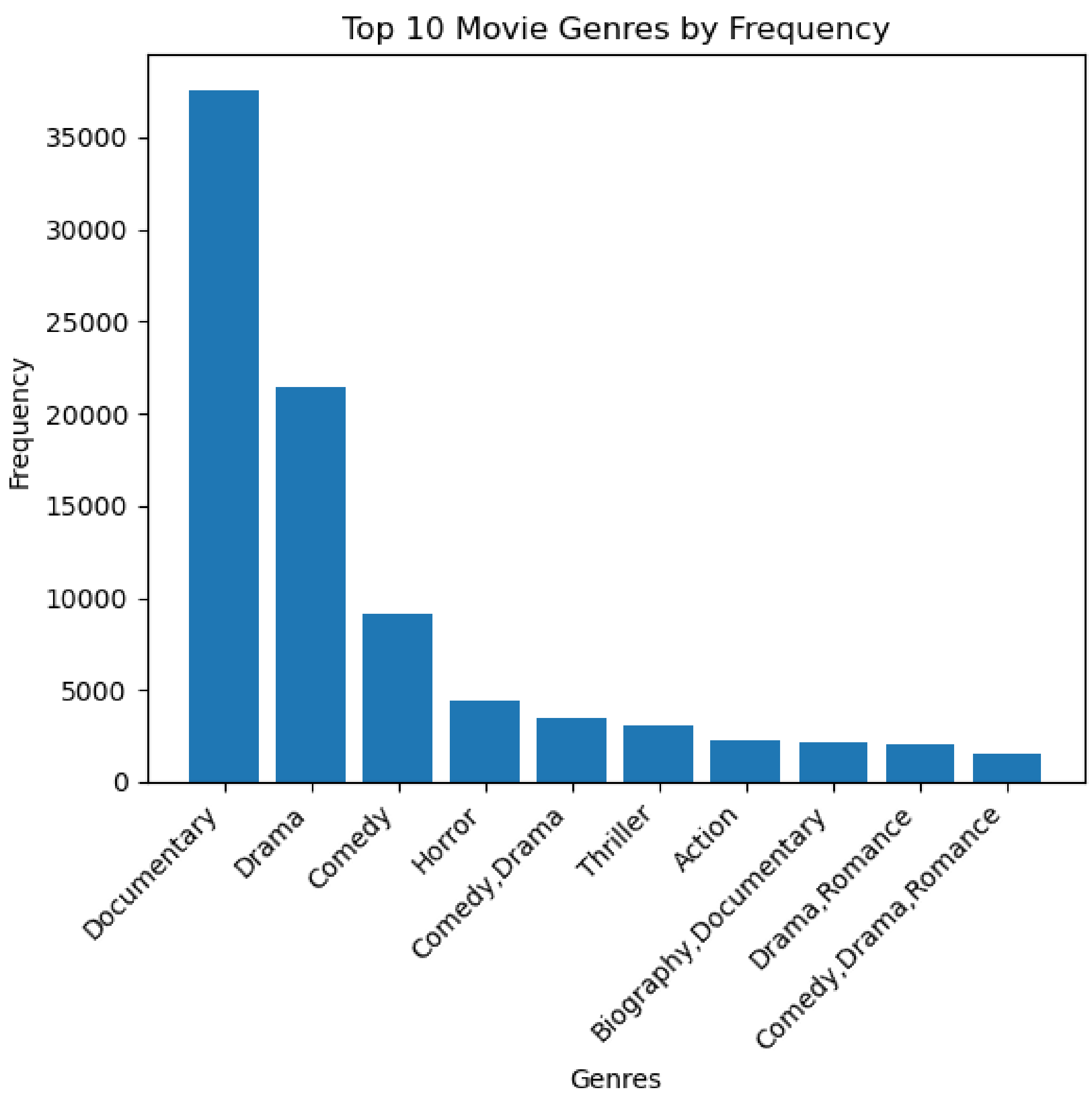
[17]:
```
  genres  COUNT(*)
0   None         0
```

[18]:
```
# replacing null values with the mode
third_query = '''
    UPDATE movie_basics
    SET genres = (
        SELECT genres
        FROM (
            SELECT genres, COUNT(*) AS count
            FROM movie_basics
            WHERE genres IS NOT NULL
            GROUP BY genres
            ORDER BY count DESC
            LIMIT 1
        ) AS mode_value
    )
    WHERE genres IS NULL
;
'''
cursor.execute(third_query)
conn.commit()

cursor.close()
conn.close()
```

[19]:
```
conn = sqlite3.connect('im.db')
cursor = conn.cursor()
#Getting top movies in terms of genre
fourth_query = """
SELECT genres, count(*) AS frequency
FROM movie_basics
GROUP by genres
ORDER BY frequency DESC
LIMIT 10
;
"""
df = pd.read_sql(fourth_query, conn)

df = df.dropna(subset=['genres', 'frequency'])
df
```

[19]:
```
                    genres  frequency
0              Documentary      37593
1                    Drama      21486
2                   Comedy       9177
3                   Horror       4372
4             Comedy,Drama       3519
5                 Thriller       3046
6                   Action       2219
7   Biography,Documentary       2115
8            Drama,Romance       2079
9    Comedy,Drama,Romance       1558
```

[20]:
```
plt.figure(figsize=(6, 6))

plt.bar(df['genres'], df['frequency'])
plt.xlabel('Genres')
plt.ylabel('Frequency')
plt.title('Top 10 Movie Genres by Frequency')
plt.xticks(rotation=45, ha='right')

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



### 2.1 Conclusions from genres by frequency

The leading genres in terms of frequency are: 1. documentary 2. drama 3. comedy 4. horror 5. comedy, drama

## 3 Table 3(movie_ratings)

[21]:
```
#Previewing data from movie_ratings
fourth_query = """
SELECT *
FROM movie_ratings
;
```

```
"""
pd.read_sql(fourth_query, conn)
```

[21]:
```
         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
…             …              …       …
73851   tt9805820            8.1        25
73852   tt9844256            7.5        24
73853   tt9851050            4.7        14
73854   tt9886934            7.0         5
73855   tt9894098            6.3       128

[73856 rows x 3 columns]
```

[22]:
```
# Joining movie_basics with movie_ratings using common column 'movie_id'
# Showing genres rating (as weighted_average_rating) from highest to lowest
new_table = """
SELECT genres, SUM(averagerating * movie_count) / SUM(movie_count) AS␣
 ↪weighted_average_rating
FROM (
    SELECT genres, averagerating, COUNT(*) AS movie_count
    FROM movie_basics
    JOIN movie_ratings USING (movie_id)
    GROUP BY genres, averagerating
) AS subquery
GROUP BY genres
ORDER BY weighted_average_rating DESC
;
"""
# The inner subquery calculates the count of movies for each genre and their␣
 ↪respective averagerating.
# The outer query then computes the weighted average rating for each genre by␣
 ↪summing the product of
# averagerating and movie_count, divided by the total movie_count for that␣
 ↪genre.
df = pd.read_sql(new_table, conn)

df.to_sql('new_table', conn, index=False, if_exists='replace')
df
```

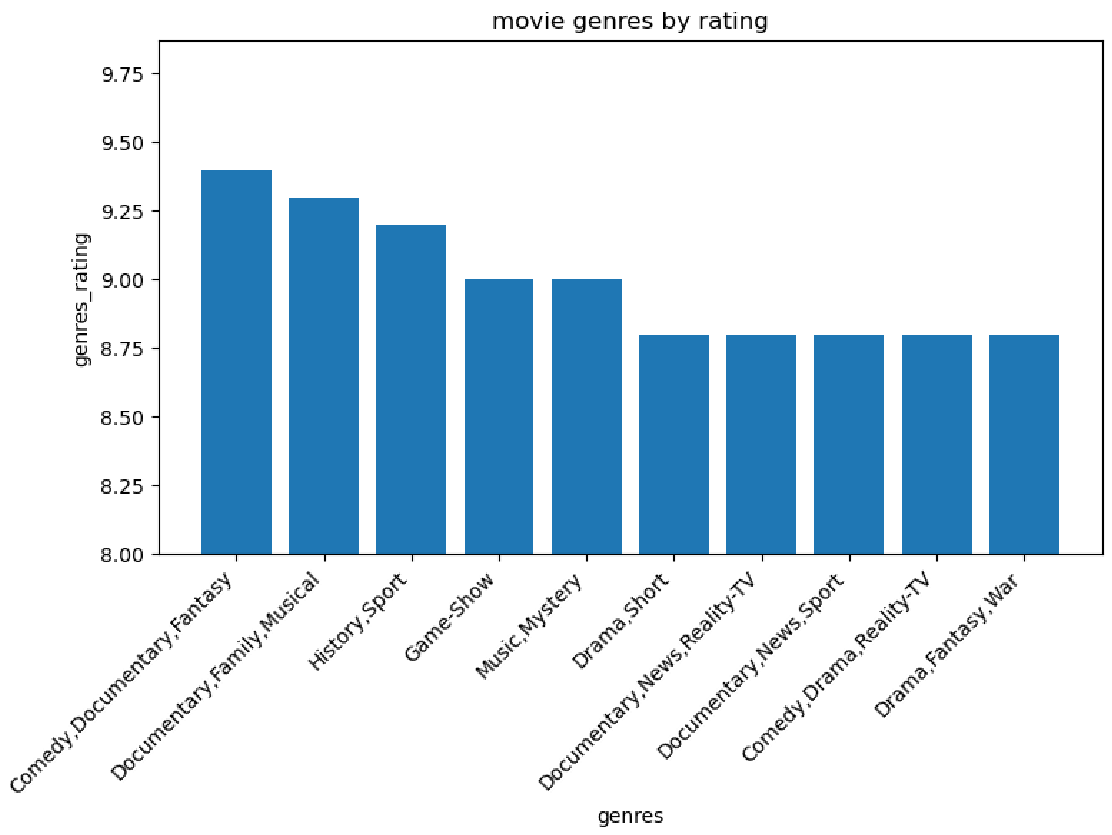
[22]:
```
                          genres  weighted_average_rating
0     Comedy,Documentary,Fantasy                      9.4
1     Documentary,Family,Musical                      9.3
```

```
2                   History,Sport                       9.2
3                   Music,Mystery                       9.0
4                       Game-Show                       9.0
..                            …                       …
918                   Crime,Music                       2.4
919      History,Sci-Fi,Thriller                       2.3
920      Adventure,Crime,Romance                       2.3
921                  Adult,Horror                       2.0
922         Comedy,Musical,Sport                       1.4

[923 rows x 2 columns]
```

[23]:
```
# Getting the mean of weighted_average_rating to avoid overcrowding of lables␣
↪when plotting
genre_ratings = df.groupby('genres')['weighted_average_rating'].mean().
↪reset_index()

df = genre_ratings.sort_values(by='weighted_average_rating', ascending=False).
↪head(10)
df
```

[23]:
```
                          genres  weighted_average_rating
449   Comedy,Documentary,Fantasy                      9.4
633   Documentary,Family,Musical                      9.3
851                History,Sport                      9.2
837                    Game-Show                      9.0
882                Music,Mystery                      9.0
775                  Drama,Short                      8.8
679  Documentary,News,Reality-TV                      8.8
681       Documentary,News,Sport                      8.8
471       Comedy,Drama,Reality-TV                      8.8
717            Drama,Fantasy,War                      8.8
```

[24]:
```
# Plotting movie genres against average rating
plt.figure(figsize=(8, 6))
plt.bar(df['genres'], df['weighted_average_rating'])
plt.xlabel('genres')
plt.ylabel('genres_rating')
plt.title('movie genres by rating')
plt.xticks(rotation=45, ha='right')
plt.ylim(bottom=8)
plt.tight_layout()
plt.show()
```



## 3.1 Conclusions from movie genres by rating

Leading genres by rating are: 1. Comedy, documentary, fantasy 2. documentary, family, musical 3. history, sport 4. game-show 5. music, mystery