Data mining - mini project 2 (Harsh Siroya)

The questions i am addressing are:

1. what is the most common genre of movies? and what is the total and mean box office revenue for each genre?
2. Is there a correlation between the actor age and the box office revenue? and what is the most common age of actors in the dataset? This will help us to know if the age of the actor has an impact on the box office revenue. Also the total box office revenue for each movie a specific actor has acted in. this will help us identify the most successful actors in the dataset. This we can also correlate with the number of movies an actor has acted in.
3. What is the total revenue per year? This will help us to know the trend of the box office revenue over the years, and if the revenue is increasing or decreasing.
4. Which movie is similar to the other movies in the dataset? This will help people watch movies that are similar to the movies they like. Just like the recommendation system in Netflix.

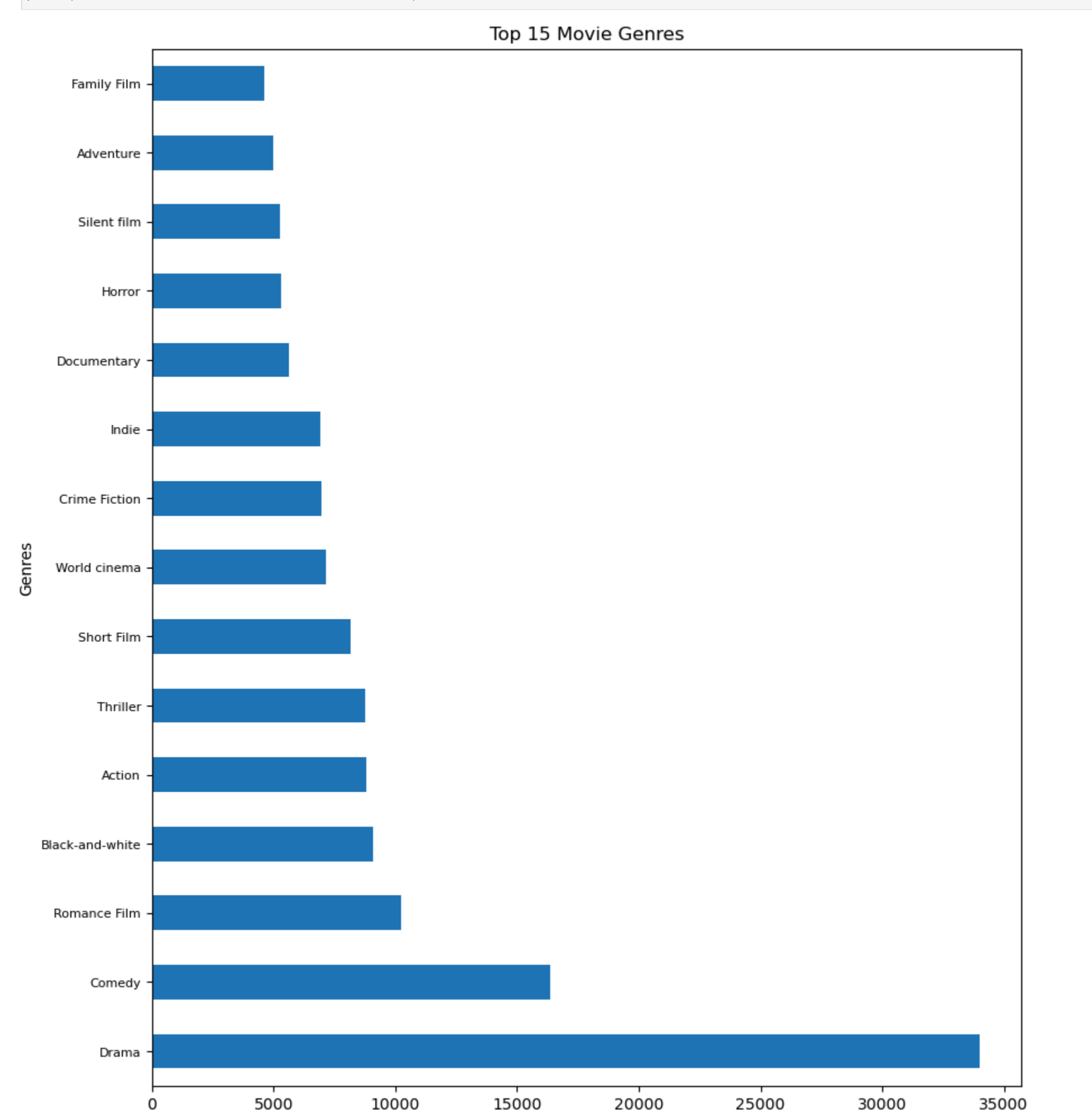
In [ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import ast
```

In [ ]:
```python
character_metadata_cols = [
    "Wikipedia Movie ID", "Freebase Movie ID","Movie Release Date", "Character Name", "Actor DOB",
    "Actor Gender", "Actor Height", "Actor Ethnicity", "Actor Name",
    "Actor Age at Movie Release", "Freebase character map id", "Freebase character id", "Freebase actor id"
]
character_metadata_df = pd.read_csv("MovieSummaries/character.metadata.tsv", sep="\t", header=None, names=character_metadata_cols)

movie_metadata_cols = [
    "Wikipedia Movie ID", "Freebase Movie ID", "Movie Name", "Movie Release Date",
    "Movie Box Office Revenue", "Movie Runtime", "Movie Languages", "Movie Countries",
    "Movie Genres"
]
movie_metadata_df = pd.read_csv("MovieSummaries/movie.metadata.tsv", sep="\t", header=None, names=movie_metadata_cols)

plot_summaries_df = pd.read_csv("MovieSummaries/plot_summaries.txt", sep="\t", header=None, names=["Wikipedia Movie ID", "Plot Summary"])

name_clusters_df = pd.read_csv("MovieSummaries/name.clusters.txt", sep="\t", header=None, names=["Name", "Cluster ID"])

tvtropes_clusters_df = pd.read_csv("MovieSummaries/tvtropes.clusters.txt", sep="\t", header=None, names=["TV Trope", "Details"])
```

In [ ]:
```python
#Exploratory Data Anakysis
print("\nMovie Metadata:")
print(movie_metadata_df.head())
print("\nSummary Information of Movie Metadata:")
print(movie_metadata_df.info())
print("\nMissing Values in Movie Metadata:")
print(movie_metadata_df.isnull().sum())
```

```
Movie Metadata:
   Wikipedia Movie ID Freebase Movie ID  \
0              975900         /m/03vyhn
1             3196793         /m/08yl5d
2            28463795        /m/0crgdbh
3             9363483        /m/0285_cd
4              261236         /m/01mrr1

                                         Movie Name Movie Release Date  \
0                                    Ghosts of Mars         2001-08-24
1  Getting Away with Murder: The JonBenét Ramsey ...         2000-02-16
2                                       Brun bitter               1988
3                                   White Of The Eye               1987
4                                  A Woman in Flames               1983

   Movie Box Office Revenue  Movie Runtime  \
0                14010832.0           98.0
1                       NaN           95.0
2                       NaN           83.0
3                       NaN          110.0
4                       NaN          106.0

                       Movie Languages  \
0  {"/m/02h40lc": "English Language"}
1  {"/m/02h40lc": "English Language"}
2  {"/m/05f_3": "Norwegian Language"}
3  {"/m/02h40lc": "English Language"}
4   {"/m/04306rv": "German Language"}

                              Movie Countries  \
0  {"/m/09c7w0": "United States of America"}
1  {"/m/09c7w0": "United States of America"}
2                      {"/m/05b4w": "Norway"}
3             {"/m/07ssc": "United Kingdom"}
4                     {"/m/0345h": "Germany"}

                                        Movie Genres
0  {"/m/01jfsb": "Thriller", "/m/06n90": "Science...
1  {"/m/02n4kr": "Mystery", "/m/03bxz7": "Biograp...
2  {"/m/0lsxr": "Crime Fiction", "/m/07s9rl0": "D...
3  {"/m/01jfsb": "Thriller", "/m/0glj9q": "Erotic...
4                             {"/m/07s9rl0": "Drama"}

Summary Information of Movie Metadata:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81741 entries, 0 to 81740
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Wikipedia Movie ID        81741 non-null  int64
 1   Freebase Movie ID         81741 non-null  object
 2   Movie Name                81741 non-null  object
 3   Movie Release Date        74839 non-null  object
 4   Movie Box Office Revenue  8401 non-null   float64
 5   Movie Runtime             61291 non-null  float64
 6   Movie Languages           81741 non-null  object
 7   Movie Countries           81741 non-null  object
 8   Movie Genres              81741 non-null  object
dtypes: float64(2), int64(1), object(6)
memory usage: 5.6+ MB
None

Missing Values in Movie Metadata:
Wikipedia Movie ID              0
Freebase Movie ID               0
Movie Name                      0
Movie Release Date           6902
Movie Box Office Revenue    73340
Movie Runtime               20450
Movie Languages                 0
Movie Countries                 0
Movie Genres                    0
dtype: int64
```

In [ ]:
```python
#Function to extract the genres from the "Movie Genres" column
def extract_genres(genres_str):
    genres_dict = ast.literal_eval(genres_str)
    genres = list(genres_dict.values())
    return genres

# Applying the function to each row of the DataFrame and concatenate the results
all_genres = movie_metadata_df["Movie Genres"].apply(extract_genres).explode()
genre_counts = all_genres.value_counts()
top_15_genres = genre_counts.head(15)

# Visualize data using horizontal bar plots for the top 100 genres
plt.figure(figsize=(10, 12))
top_15_genres.plot(kind='barh')
plt.title("Top 15 Movie Genres")
plt.xlabel("Count")
plt.ylabel("Genres")
plt.tick_params(axis='y', labelsize=8)
plt.show()

# Calculate the total number of genres
total_genres = len(all_genres.unique())
print('---------------------------------------')
print("Total number of genres:", total_genres)
print('---------------------------------------')
```



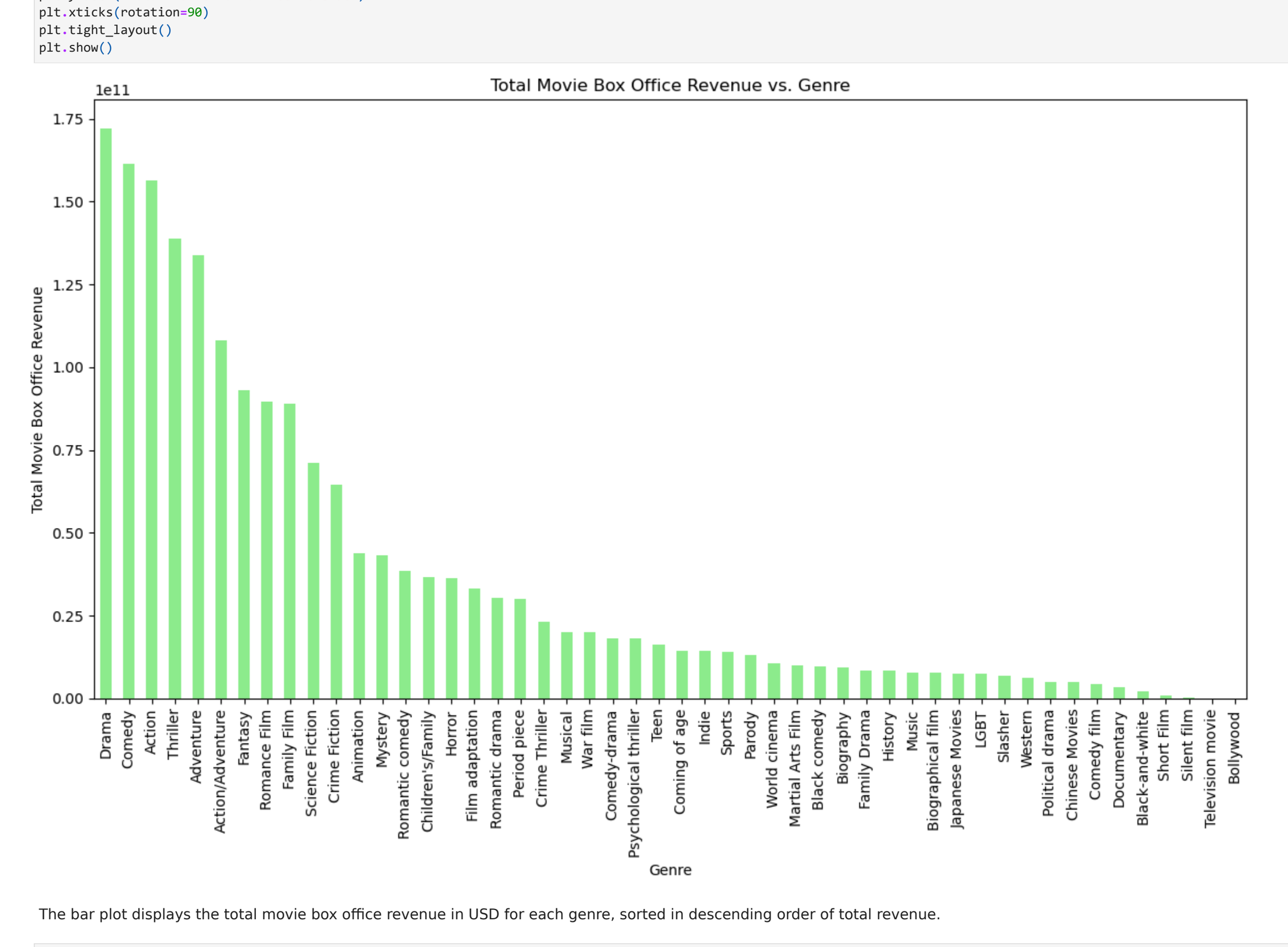

```
---------------------------------------
Total number of genres: 364
---------------------------------------
```

This bar plot show the genre which occurs the most in the dataset. this shows us which is the most popular genre in the dataset. Directors can use this information to make movies that are more likely to be successful. along with more valuable insights.

In [ ]:
```python
# To understand the character metadata
print(character_metadata_df.head())
print(character_metadata_df.info())
print(character_metadata_df.isnull().sum())
```

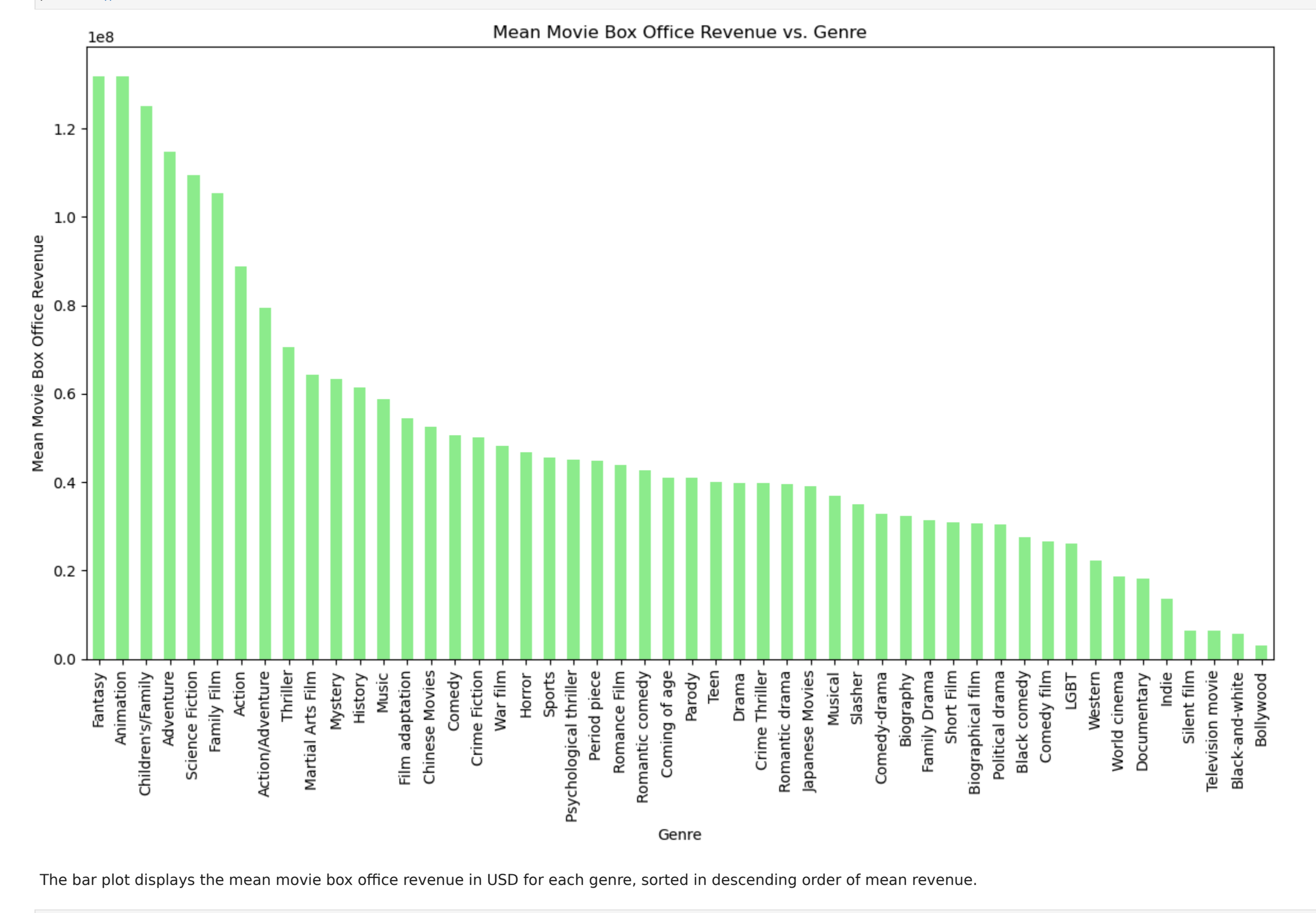
```
   Wikipedia Movie ID Freebase Movie ID Movie Release Date  \
0              975900         /m/03vyhn         2001-08-24
1              975900         /m/03vyhn         2001-08-24
2              975900         /m/03vyhn         2001-08-24
3              975900         /m/03vyhn         2001-08-24
4              975900         /m/03vyhn         2001-08-24

               Character Name   Actor DOB Actor Gender  Actor Height  \
0                    Akooshay  1958-08-26            F         1.620
1  Lieutenant Melanie Ballard  1974-08-15            F         1.780
2          Desolation Williams  1969-06-15            M         1.727
3           Sgt Jericho Butler  1967-09-12            M         1.750
4             Bashira Kincaid  1977-09-25            F         1.650

  Actor Ethnicity          Actor Name  Actor Age at Movie Release  \
0             NaN      Wanda De Jesus                        42.0
1      /m/044038p  Natasha Henstridge                        27.0
2         /m/0x67            Ice Cube                        32.0
3             NaN       Jason Statham                        33.0
4             NaN         Clea DuVall                        23.0

  Freebase character map id Freebase character id Freebase actor id
0                /m/0bgchxw            /m/0bgcj3x        /m/03wcfv7
1                 /m/0jys3m            /m/0bgchn4         /m/0346l4
2                 /m/0jys3g            /m/0bgchn_        /m/01vw26l
3                /m/02vchl6            /m/0bgchnq         /m/034hyc
4                /m/02vbb3r            /m/0bgchp9         /m/01y9xg
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 450669 entries, 0 to 450668
Data columns (total 13 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Wikipedia Movie ID          450669 non-null  int64
 1   Freebase Movie ID           450669 non-null  object
 2   Movie Release Date          440674 non-null  object
 3   Character Name              192794 non-null  object
 4   Actor DOB                   344524 non-null  object
 5   Actor Gender                405060 non-null  object
 6   Actor Height                154824 non-null  float64
 7   Actor Ethnicity             106058 non-null  object
 8   Actor Name                  449441 non-null  object
 9   Actor Age at Movie Release  292556 non-null  float64
 10  Freebase character map id   450669 non-null  object
 11  Freebase character id       192804 non-null  object
 12  Freebase actor id           449854 non-null  object
dtypes: float64(2), int64(1), object(10)
memory usage: 44.7+ MB
None
Wikipedia Movie ID                 0
Freebase Movie ID                  0
Movie Release Date              9995
Character Name                257875
Actor DOB                     106145
Actor Gender                   45609
Actor Height                  295845
Actor Ethnicity               344611
Actor Name                      1228
Actor Age at Movie Release    158113
Freebase character map id          0
Freebase character id         257865
Freebase actor id                815
dtype: int64
```

In [ ]:
```python
movie_metadata_df["Genres"] = movie_metadata_df["Movie Genres"].apply(extract_genres)
movie_genres_df = movie_metadata_df.explode("Genres")
movie_genres_df = movie_genres_df.dropna(subset=["Movie Box Office Revenue"])

top_50_genres = genre_counts.head(50).index
movie_genres_df = movie_genres_df[movie_genres_df["Genres"].isin(top_50_genres)]

movie_genres_df["Movie Box Office Revenue"] = pd.to_numeric(movie_genres_df["Movie Box Office Revenue"])

# Calculate the total revenue for each genre
total_revenue_by_genre = movie_genres_df.groupby("Genres")["Movie Box Office Revenue"].sum().sort_values(ascending=False)

# Plot the bar plot
plt.figure(figsize=(12, 8))
total_revenue_by_genre.plot(kind='bar', color='lightgreen')
plt.title("Total Movie Box Office Revenue vs. Genre")
plt.xlabel("Genre")
plt.ylabel("Total Movie Box Office Revenue")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



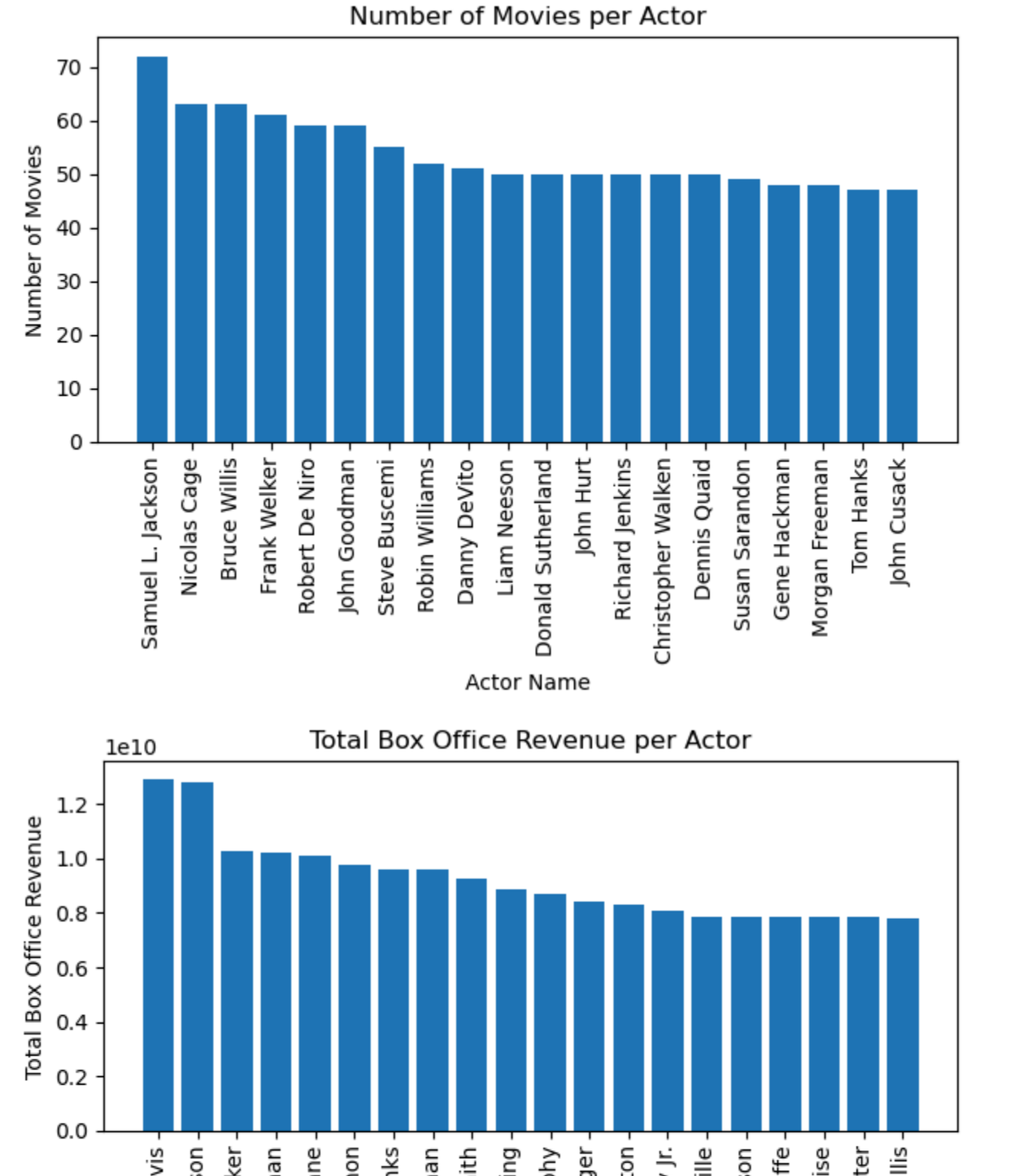
The bar plot displays the total movie box office revenue in USD for each genre, sorted in descending order of total revenue.

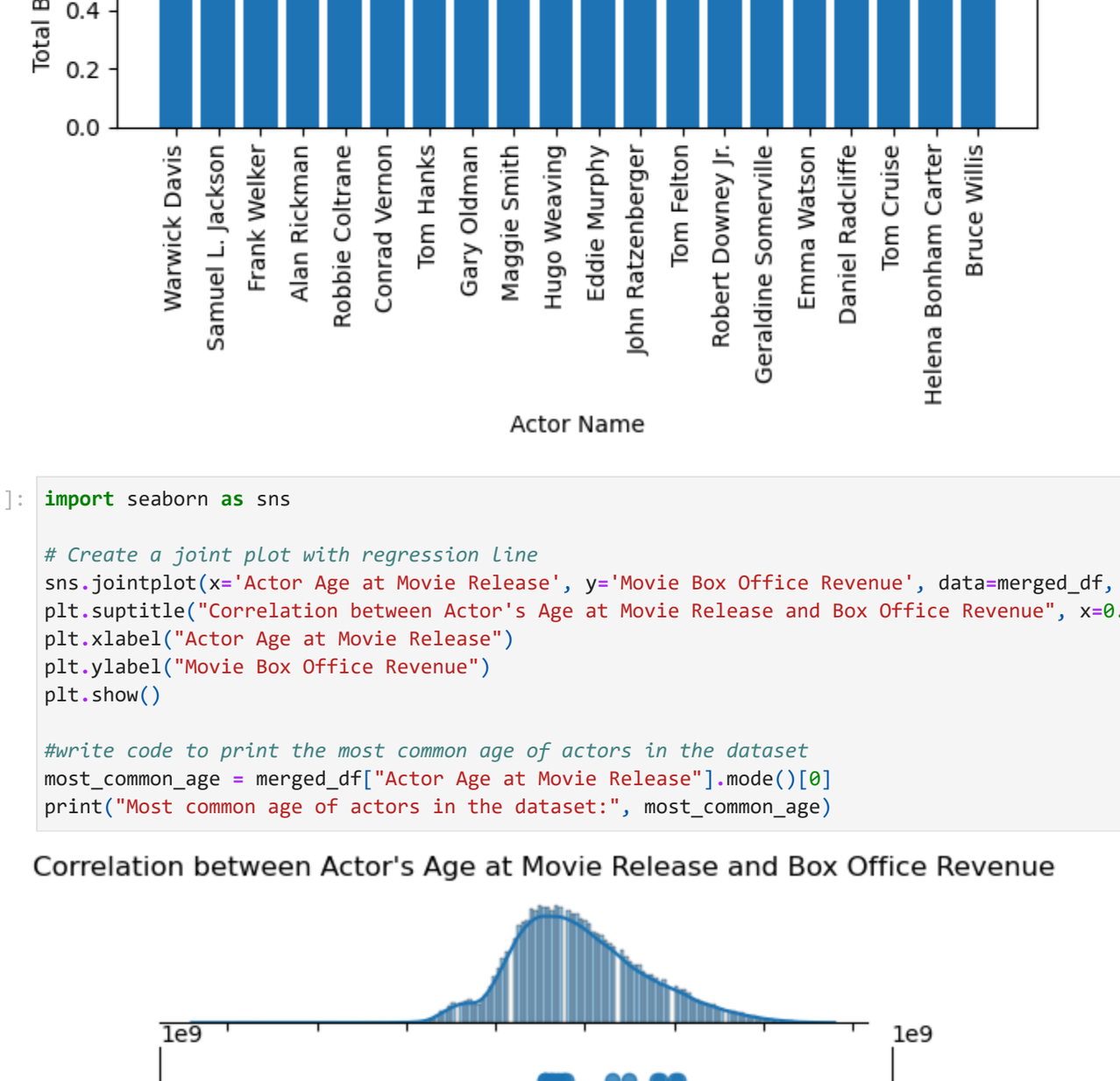
In [ ]:
```python
# Calculate the total revenue for each genre
mean_revenue_by_genre = movie_genres_df.groupby("Genres")["Movie Box Office Revenue"].mean().sort_values(ascending=False)

# Plot the bar plot
plt.figure(figsize=(12, 8))
mean_revenue_by_genre.plot(kind='bar', color='lightgreen')
plt.title("Mean Movie Box Office Revenue vs. Genre")
plt.xlabel("Genre")
plt.ylabel("Mean Movie Box Office Revenue")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



The bar plot displays the mean movie box office revenue in USD for each genre, sorted in descending order of mean revenue.

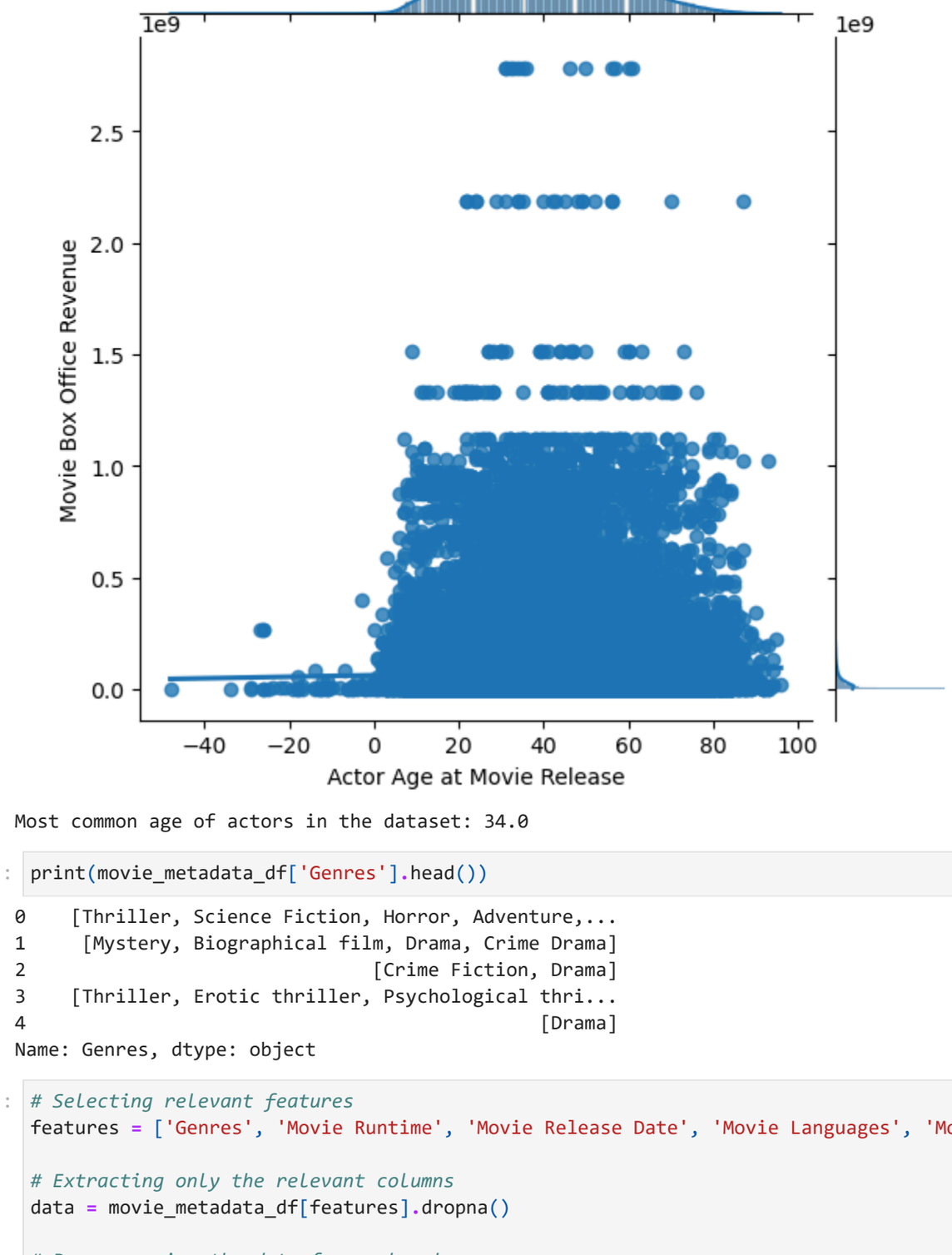
In [ ]:
```python
merged_df = pd.merge(character_metadata_df, movie_metadata_df, on="Wikipedia Movie ID", how="inner")
merged_df.dropna(subset=["Actor Name", "Movie Name", "Movie Box Office Revenue"], inplace=True)

# Exploratory Data Analysis
# Analyze the distribution of the number of movies each actor has appeared in
movies_per_actor = (merged_df.groupby("Actor Name")["Movie Name"].count()).sort_values(ascending=False)
plt.bar(movies_per_actor.index[:20], movies_per_actor.values[:20])
plt.title("Number of Movies per Actor")
plt.xlabel("Actor Name")
plt.ylabel("Number of Movies")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()


# # Analyze box office success of actors
box_office_revenue_per_actor = (merged_df.groupby("Actor Name")["Movie Box Office Revenue"].sum()).sort_values(ascending=False)
plt.bar(box_office_revenue_per_actor.index[:20], box_office_revenue_per_actor.values[:20])
plt.title("Total Box Office Revenue per Actor")
plt.xlabel("Actor Name")
plt.ylabel("Total Box Office Revenue")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```





In [ ]:
```python
import seaborn as sns

# Create a joint plot with regression line
sns.jointplot(x='Actor Age at Movie Release', y='Movie Box Office Revenue', data=merged_df, kind='reg')
plt.suptitle("Correlation between Actor's Age at Movie Release and Box Office Revenue", x=0.5, y=1.02)
plt.xlabel("Actor Age at Movie Release")
plt.ylabel("Movie Box Office Revenue")
plt.show()

#write code to print the most common age of actors in the dataset
most_common_age = merged_df["Actor Age at Movie Release"].mode()[0]
print("Most common age of actors in the dataset:", most_common_age)
```

Correlation between Actor's Age at Movie Release and Box Office Revenue

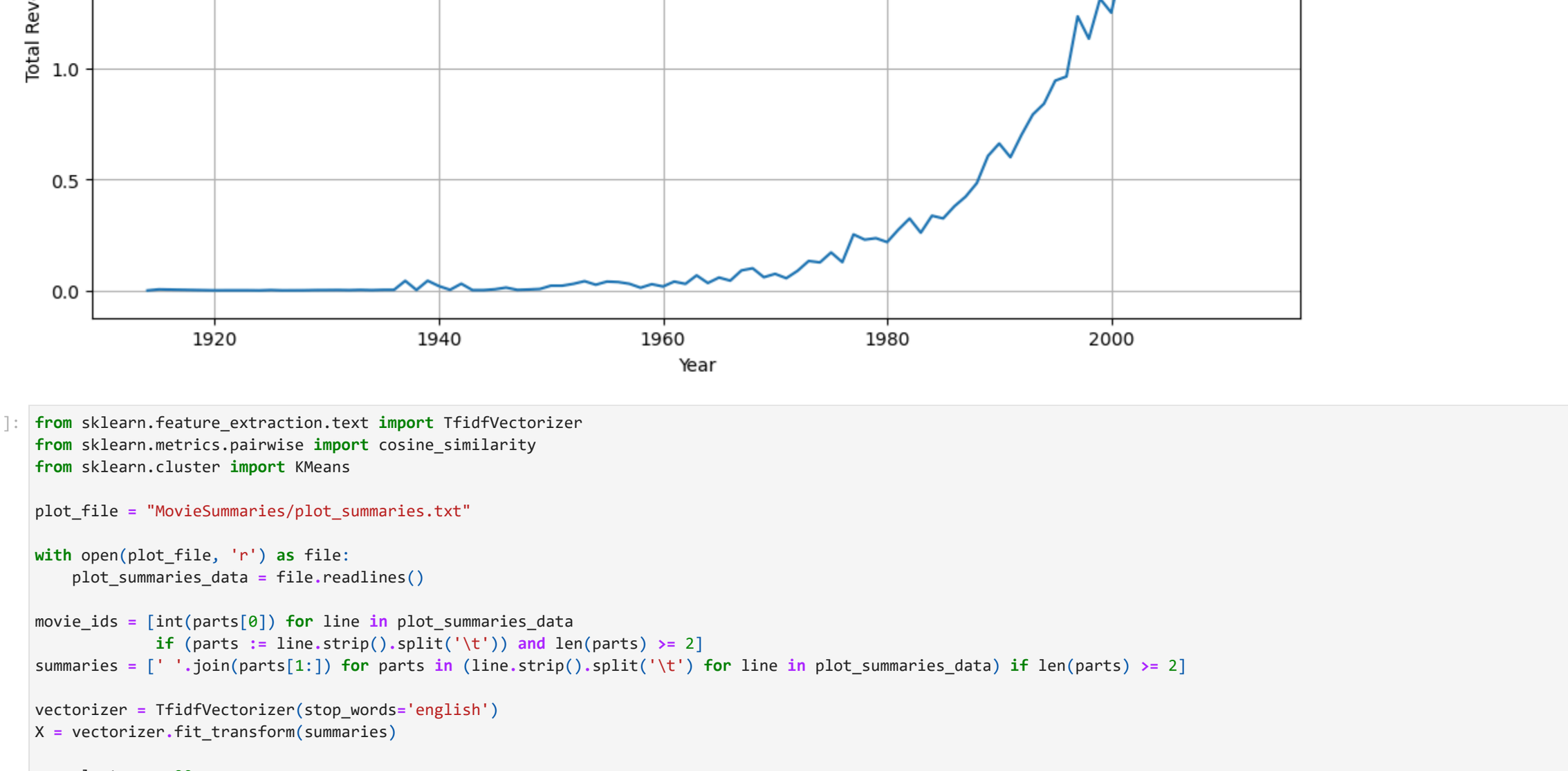



```
Most common age of actors in the dataset: 34.0
```

In [ ]:
```python
print(movie_metadata_df['Genres'].head())
```

```
0    [Thriller, Science Fiction, Horror, Adventure,...
1     [Mystery, Biographical film, Drama, Crime Drama]
2                              [Crime Fiction, Drama]
3    [Thriller, Erotic thriller, Psychological thri...
4                                            [Drama]
Name: Genres, dtype: object
```

In [ ]:
```python
# Selecting relevant features
features = ['Genres', 'Movie Runtime', 'Movie Release Date', 'Movie Languages', 'Movie Countries', 'Movie Box Office Revenue']

# Extracting only the relevant columns
data = movie_metadata_df[features].dropna()

# Preprocessing the data for each column
data['Movie Release Date'] = data['Movie Release Date'].str.extract(r'(\d{4})')
data['Movie Release Date'] = pd.to_numeric(data['Movie Release Date'])


data['Movie Release Date'] = data['Movie Release Date'].drop(data[data['Movie Release Date'] < 1900].index)

# for each release year, calculate the total revenue in that year
revenue_per_year = data.groupby('Movie Release Date')['Movie Box Office Revenue'].sum()

# Plot the total revenue per year
plt.figure(figsize=(12, 6))
revenue_per_year.plot()
plt.title("Total Revenue per Year")
plt.xlabel("Year")
plt.ylabel("Total Revenue")
plt.grid()
plt.show()
```



In [ ]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import KMeans

plot_file = "MovieSummaries/plot_summaries.txt"

with open(plot_file, 'r') as file:
    plot_summaries_data = file.readlines()

movie_ids = [int(parts[0]) for line in plot_summaries_data
             if (parts := line.strip().split('\t')) and len(parts) >= 2]
summaries = [' '.join(parts[1:]) for parts in (line.strip().split('\t') for line in plot_summaries_data) if len(parts) >= 2]

vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(summaries)

num_clusters = 20
kmeans = KMeans(n_clusters=num_clusters)
kmeans.fit(X)

label_clusters = kmeans.labels_

movie_names = pd.DataFrame({"Wikipedia Movie ID": movie_ids})
movie_names = movie_names.merge(movie_metadata_df[["Wikipedia Movie ID", "Movie Name"]],on="Wikipedia Movie ID", how="left")

movies_clusters = {}
for i, cluster_label in enumerate(label_clusters):
    if cluster_label not in movies_clusters:
        movies_clusters[cluster_label] = []
    movies_clusters[cluster_label].append(movie_names.iloc[i]["Movie Name"])

for cluster_label, movies in movies_clusters.items():
    print(f"Cluster {cluster_label + 1}:")
    movie_names = movies[:10]
    print(", ".join(movie_names))
    print("Total Movies:", len(movies))
    print()
```

```
/opt/miniconda3/lib/python3.12/site-packages/sklearn/cluster/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
Cluster 4:
Taxi Blues, The Hunger Games, Destination Meatball, Up and Down, Charlie Chan's Secret, Ashes to Ashes, Red's Dream, Nee Sneham, Bhagwan Dada, Kehtaa Hai Dil Baar Baar
Total Movies: 14922

Cluster 10:
Narasimham, Dark Water, Husband for Hire, Come Back, Africa, Mr. & Mrs. '55, Eastern Promises, You, Kelviyum Naane Pathilum Naane, Geometria, Wait Until Spring, Bandini
Total Movies: 5763

Cluster 13:
The Lemon Drop Kid, A Cry in the Dark, End Game, House Party 2, Pieces, Killjoy, Jaws: The Revenge, Hush… Hush, Sweet Charlotte, Treed Murray, Expired
Total Movies: 6392

Cluster 9:
Sing, The Biggest Fan, A la salida nos vemos, Girl, Positive, The Incredibly True Adventure of Two Girls in Love, Classmates, Friendship, The Brotherhood III: Young Demons, Fish Hooky, American Pie Presents: The Book of Love
Total Movies: 1460

Cluster 17:
Meet John Doe, Darkness, A Modern Hero, Saturday the 14th, The Tie That Binds, All Quiet on the Western Front, Let Sleeping Corpses Lie, Heat Lightning, Food of Love, Princess of Thieves
Total Movies: 698

Cluster 12:
Ghost In The Noonday Sun, Class of '61, Against Her Will: An Incident in Baltimore, La Cité de la peur, A Slumdog Millionaire Goes Dancing, Twist, Mysterious Mose, In the Land of the Deaf, Changes, The Emperor Jones
Total Movies: 3442

Cluster 6:
Exodus, Green Dragon, The Rats of Tobruk, A Merry Mixup, The Good, the Bad, the Weird, Rendezvous, The Bloody Fists, The Assassin, Jacob, the Liar, Ghetto Stories: The Movie
Total Movies: 1904

Cluster 5:
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[39], line 35
     33 print(f"Cluster {cluster_label + 1}:")
```