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

```
rawdata = pd.read_csv('movie.csv')
```

```
#Copy of raw data
df = rawdata.copy()
```

Exploration

```
df.shape
```

```
(4916, 27)
```

```
df.head()
```

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	ac
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	

5 rows × 28 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4916 entries, 0 to 4915
Data columns (total 28 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   color                      4897 non-null   object
 1   director_name              4814 non-null   object
 2   num_critic_for_reviews     4867 non-null   float64
 3   duration                   4901 non-null   float64
 4   director_facebook_likes    4814 non-null   float64
 5   actor_3_facebook_likes     4893 non-null   float64
 6   actor_2_name               4903 non-null   object
 7   actor_1_facebook_likes     4909 non-null   float64
 8   gross                      4054 non-null   float64
 9   genres                     4916 non-null   object
 10  actor_1_name               4909 non-null   object
 11  movie_title                4916 non-null   object
 12  num_voted_users            4916 non-null   int64
 13  cast_total_facebook_likes  4916 non-null   int64
 14  actor_3_name               4893 non-null   object
 15  facenumber_in_poster       4903 non-null   float64
 16  plot_keywords              4764 non-null   object
 17  movie_imdb_link            4916 non-null   object
 18  num_user_for_reviews       4895 non-null   float64
 19  language                   4902 non-null   object
 20  country                    4911 non-null   object
 21  content_rating             4616 non-null   object
 22  budget                     4432 non-null   float64
```

```
 23  title_year                 4810 non-null   float64
 24  actor_2_facebook_likes     4903 non-null   float64
 25  imdb_score                 4916 non-null   float64
 26  aspect_ratio               4590 non-null   float64
 27  movie_facebook_likes       4916 non-null   int64
dtypes: float64(13), int64(3), object(12)
memory usage: 1.1+ MB
```

```
df.describe(include='all')
```

```
#print the numer of nan values  for each column there are two ways
#first using for loop
for col in list(df.columns):
    print(f"The number of missing values in column {col}= {df[col].isnull().sum()}")
#usnig builtin function
#print(f"The number of missing values in  \n {df.isnull().sum()}=")
```

```
The number of missing values in column color= 19
The number of missing values in column director_name= 102
The number of missing values in column num_critic_for_reviews= 49
The number of missing values in column duration= 15
The number of missing values in column director_facebook_likes= 102
The number of missing values in column actor_3_facebook_likes= 23
The number of missing values in column actor_2_name= 13
The number of missing values in column actor_1_facebook_likes= 7
The number of missing values in column gross= 862
The number of missing values in column genres= 0
The number of missing values in column actor_1_name= 7
The number of missing values in column movie_title= 0
The number of missing values in column num_voted_users= 0
The number of missing values in column cast_total_facebook_likes= 0
The number of missing values in column actor_3_name= 23
The number of missing values in column facenumber_in_poster= 13
The number of missing values in column plot_keywords= 152
The number of missing values in column movie_imdb_link= 0
The number of missing values in column num_user_for_reviews= 21
The number of missing values in column language= 14
The number of missing values in column country= 5
The number of missing values in column content_rating= 300
The number of missing values in column budget= 484
The number of missing values in column title_year= 106
The number of missing values in column actor_2_facebook_likes= 13
The number of missing values in column imdb_score= 0
The number of missing values in column aspect_ratio= 326
The number of missing values in column movie_facebook_likes= 0
```

Processing

```
#Drop unnecessary Columns
#Drop the 'color column from the DataFrame
df.drop('color', axis=1, inplace=True)
df.drop('aspect_ratio', axis=1, inplace=True)
df.drop('plot_keywords', axis=1, inplace=True)
df.drop('facenumber_in_poster', axis=1, inplace=True)
df.drop('actor_3_facebook_likes', axis=1, inplace=True)
df.drop('actor_2_facebook_likes', axis=1, inplace=True)
df.drop('actor_3_name', axis=1, inplace=True)
df.drop_duplicates(subset='movie_title',inplace=True)
```

```
#fill in missing values
df['gross'].fillna(df['gross'].mean(),inplace = True)
df['budget'].fillna(df['gross'].mean(),inplace = True)
#remove empty rows
df.dropna(inplace=True)
#shape after removing empty rows and Drop unnecessary Columns
df.shape
```

```
(4296, 22)
```

```
#converting data type
df['budget'] = df['budget'].astype('int64')
df['gross'] = df['gross'].astype('int64')
df['title_year'] = pd.to_datetime(df['title_year'], format='%Y')
df['title_year'] = df['title_year'].dt.strftime('%Y')
```

```
# Calculate profit by subtracting the budget from the gross amount
df['profit'] = df['gross'] - df['budget']
```

```
# Split the 'genres' column by '|' and select the first genre
df['genres'] = df['genres'].str.split('|').str[0]
```

```
#searching for empty values
print(df.isnull().sum())
```

Show hidden output

```
# Display rows where 'country' doesn't start with a letter (a-z) or end with a letter (a-z)
df[~df["country"].str.match("^[a-zA-Z].*[a-zA-Z]$")]
```

director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	gen

0 rows × 23 columns

```
#download preprocessed data
preprocessed_data = df.to_csv('df.csv', index=False)
```

Analytics

```
#Exploration step
print('the average facebooklikes= ',df["movie_facebook_likes"].mean().round(2))
print('Max rate : ',df["imdb_score"].max())
print('Minimun rate : ',df["imdb_score"].min())
print('Number of voted people : ',df["num_voted_users"].sum())
print('The Average Budget : ',df["budget"].mean())
```

```
print('The most frequent country : ',df["country"].mode())
```

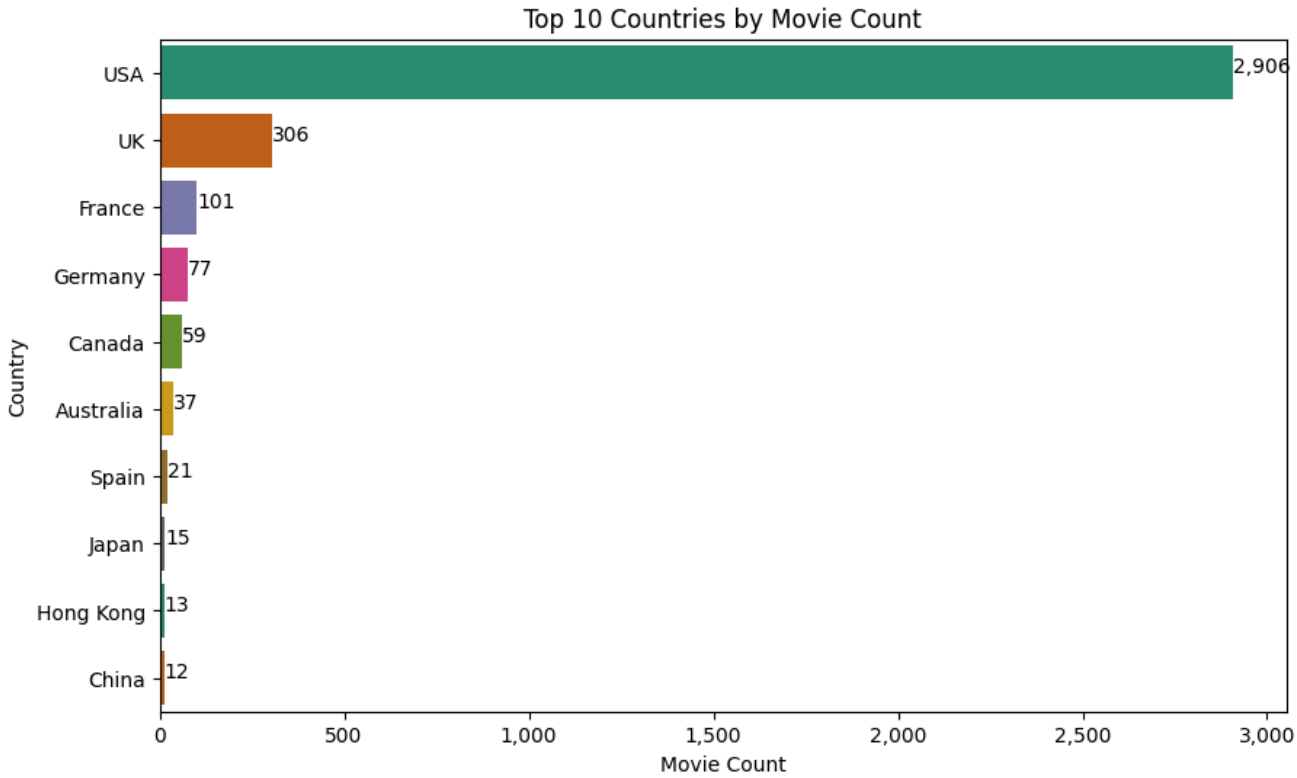
Show hidden output

```
max_rate = df['imdb_score'].max()
rate = df[df['imdb_score'] == max_rate][['movie_title','genres','imdb_score']]
rate.columns = ['title','genres','rate']
min_rate =df['imdb_score'].min()
rate2 = df[df['imdb_score'] == min_rate][['movie_title','genres','imdb_score']]
rate2.columns = ['title','genres','rate']
rate = pd.concat([rate,rate2])
rate.reset_index(drop=True,inplace=True)
rate
```

Show hidden output

```
# Group by 'country' and count the number of 'movie_title', then sort by 'movie_count' in descending order
grouped = df.groupby('country')['movie_title'].count().reset_index(name='movie_count').sort_values(by='movie_count', ascending=F

# Select the top 10 rows
top_10 = grouped.head(10)

# Plot the top 10 values using seaborn with horizontal bars
plt.figure(figsize=(10, 6))
sns.barplot(x='movie_count', y='country', data=top_10, palette='Dark2')  # Horizontal bar plot

# Add values to each bar
for index, value in enumerate(top_10['movie_count']):
    # Format the value with commas and no decimal places
    formatted_value = f"{value:,.00f}"
    plt.text(value, index, str(formatted_value))
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f"{x:,.0f}"))
plt.title('Top 10 Countries by Movie Count')
plt.xlabel('Movie Count')
plt.ylabel('Country')
plt.show()
```

```
<ipython-input-11-b7dd16064ee9>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and

  sns.barplot(x='movie_count', y='country', data=top_10, palette='Dark2')  # Horizontal bar plot
```



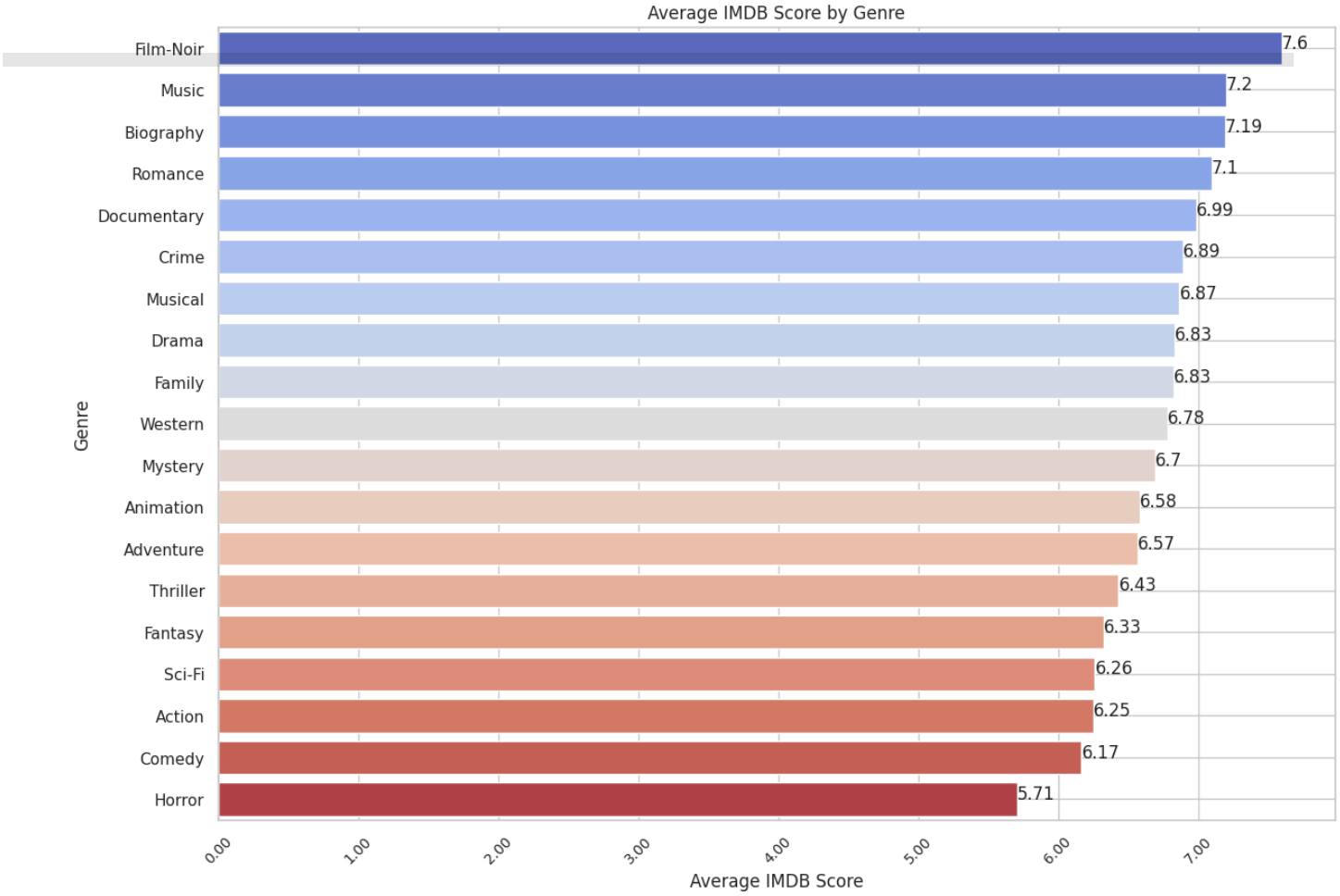

```
genre =df['genres'].str.split('|', expand=True).stack().value_counts()
print(genre)
```

```
Drama          2203
Comedy         1660
Thriller       1228
Action         1022
Romance         982
Adventure       825
Crime           785
Sci-Fi          546
Fantasy         537
Family          474
Horror          471
Mystery         430
Biography       270
Animation       212
War             185
Music           183
History         179
Sport           167
Musical         116
Western          83
Documentary      64
Film-Noir         5
News              1
Name: count, dtype: int64
```

```
# Set the style and figure size
sns.set(style="whitegrid")
plt.figure(figsize=(14, 10))

groupedrate = df.groupby('genres')['imdb_score'].mean().sort_values(ascending=False).reset_index() # Convert the Series to DataF
sns.barplot(x='imdb_score', y='genres', data=groupedrate, palette='coolwarm',legend=False)
```

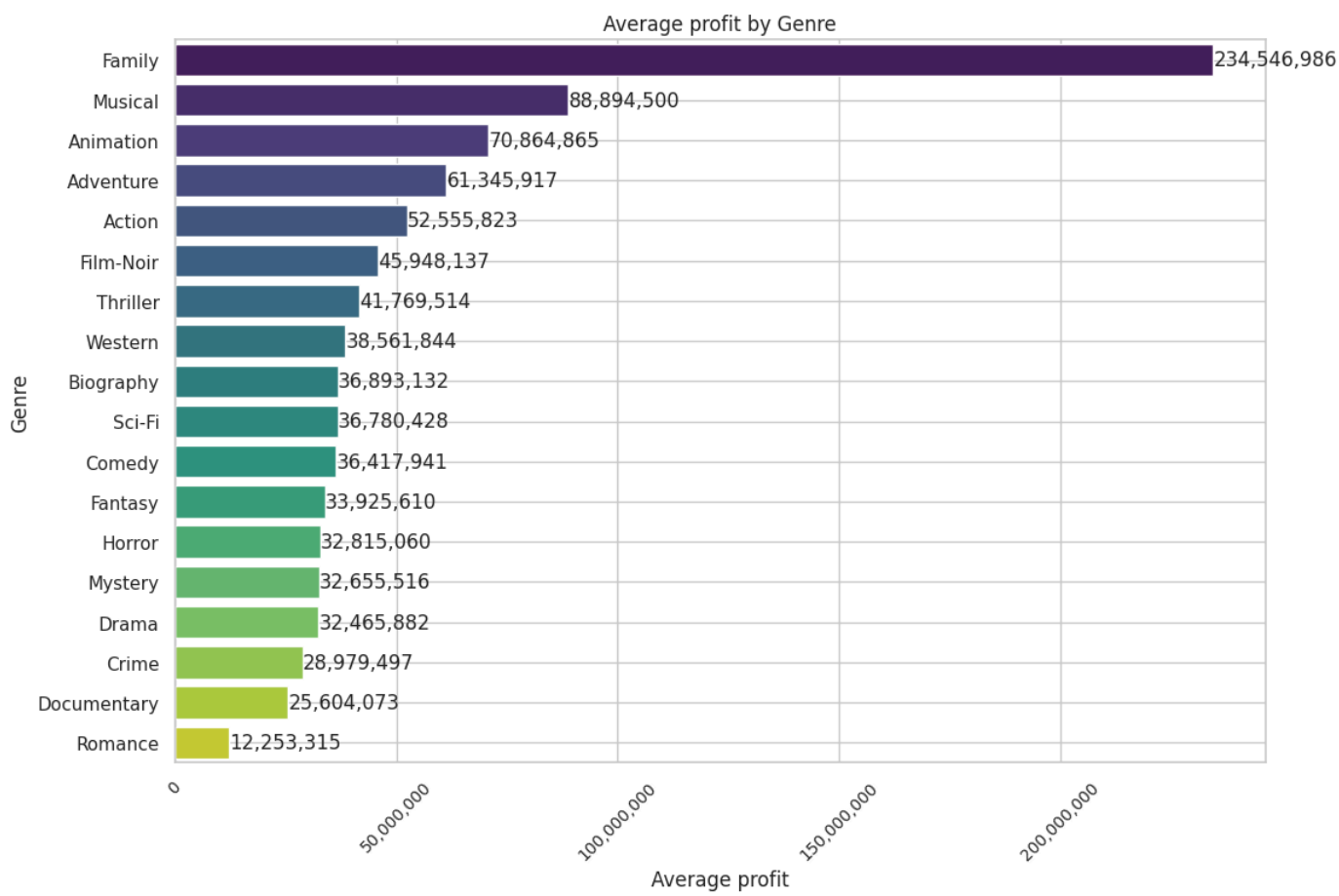
```
# Add values to each bar
for index,value in enumerate(groupedrate['imdb_score']):

    plt.text(value,index,str(round(value,2)))

# Customize the x-axis ticks for better readability
#This line of code customizes the formatting of the major tick labels on the x-axis.
plt.xticks(rotation=45, fontsize=10)  # Rotate x-axis labels and set font size

plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f"{x:,.2f}"))
plt.grid(True)

plt.title('Average IMDB Score by Genre')
plt.xlabel('Average IMDB Score')
plt.ylabel('Genre')
plt.show()
```

```
<ipython-input-64-84f318965660>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and

  sns.barplot(x='imdb_score', y='genres', data=groupedrate, palette='coolwarm',legend=False)
```



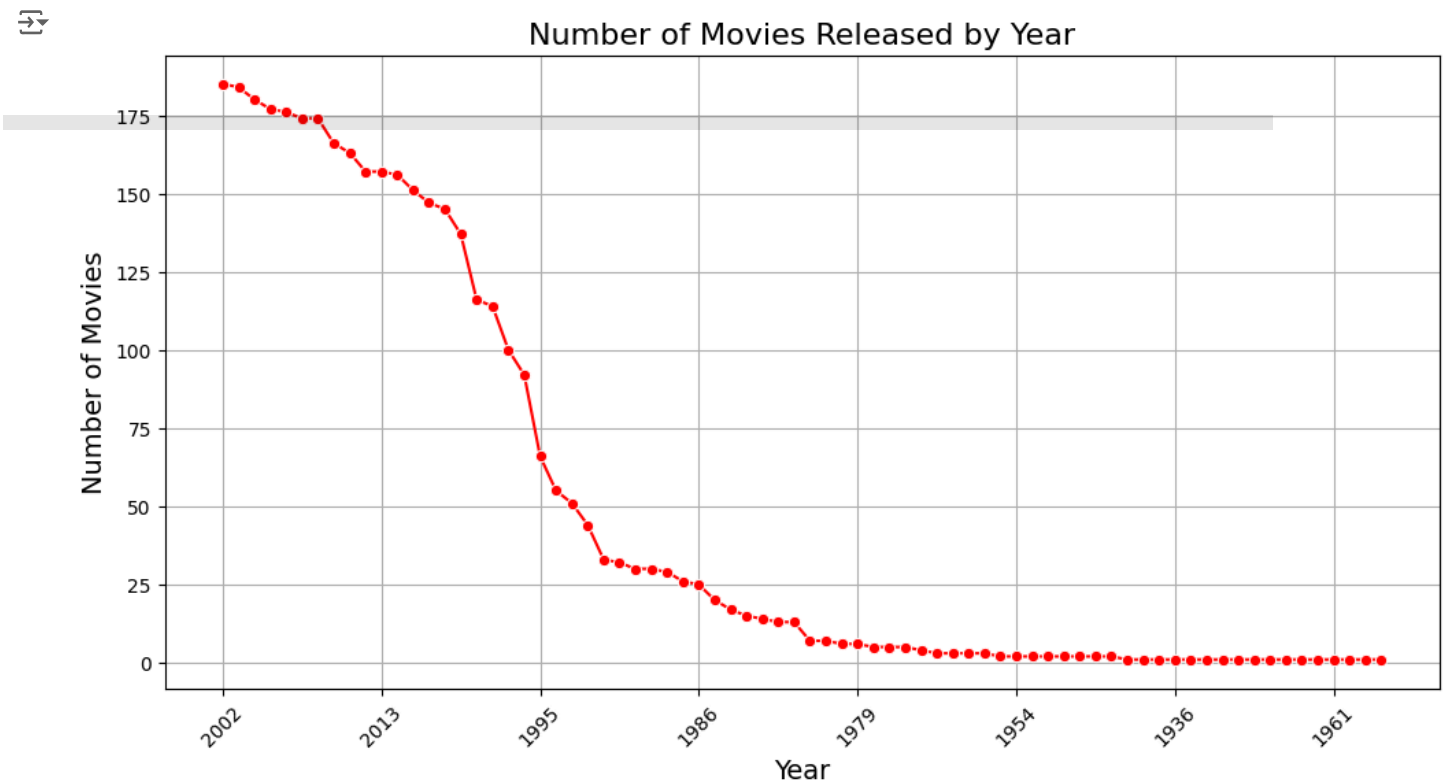

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set the style and figure size
sns.set(style="whitegrid")
plt.figure(figsize=(12, 8))
```

```
# Filter the DataFrame to include only positive profit values
positive_profit_df = df[df['profit'] > 0]

# Group by 'genres' and calculate the mean profit, then sort and reset the index
groupedrate = positive_profit_df.groupby('genres')['profit'].mean().sort_values(ascending=False).reset_index()

# Create a bar plot with 'hue' set to 'genres' and legend disabled
sns.barplot(x='profit', y='genres', hue='genres', data=groupedrate, palette='viridis', dodge=False, legend=False)

# Add values to each bar
for index, value in enumerate(groupedrate['profit']):
    # Format the value with commas and no decimal places
    formatted_value = f"{value:,.0f}"
    plt.text(value, index, formatted_value, va='center')  # Adjust vertical alignment

# Customize the x-axis ticks for better readability
plt.xticks(rotation=45, fontsize=10)  # Rotate x-axis labels and set font size

# Customize the formatting of the major tick labels on the x-axis
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f"{x:,.0f}"))

plt.grid(True)

plt.title('Average profit by Genre')
plt.xlabel('Average profit')
plt.ylabel('Genre')
plt.show()
```



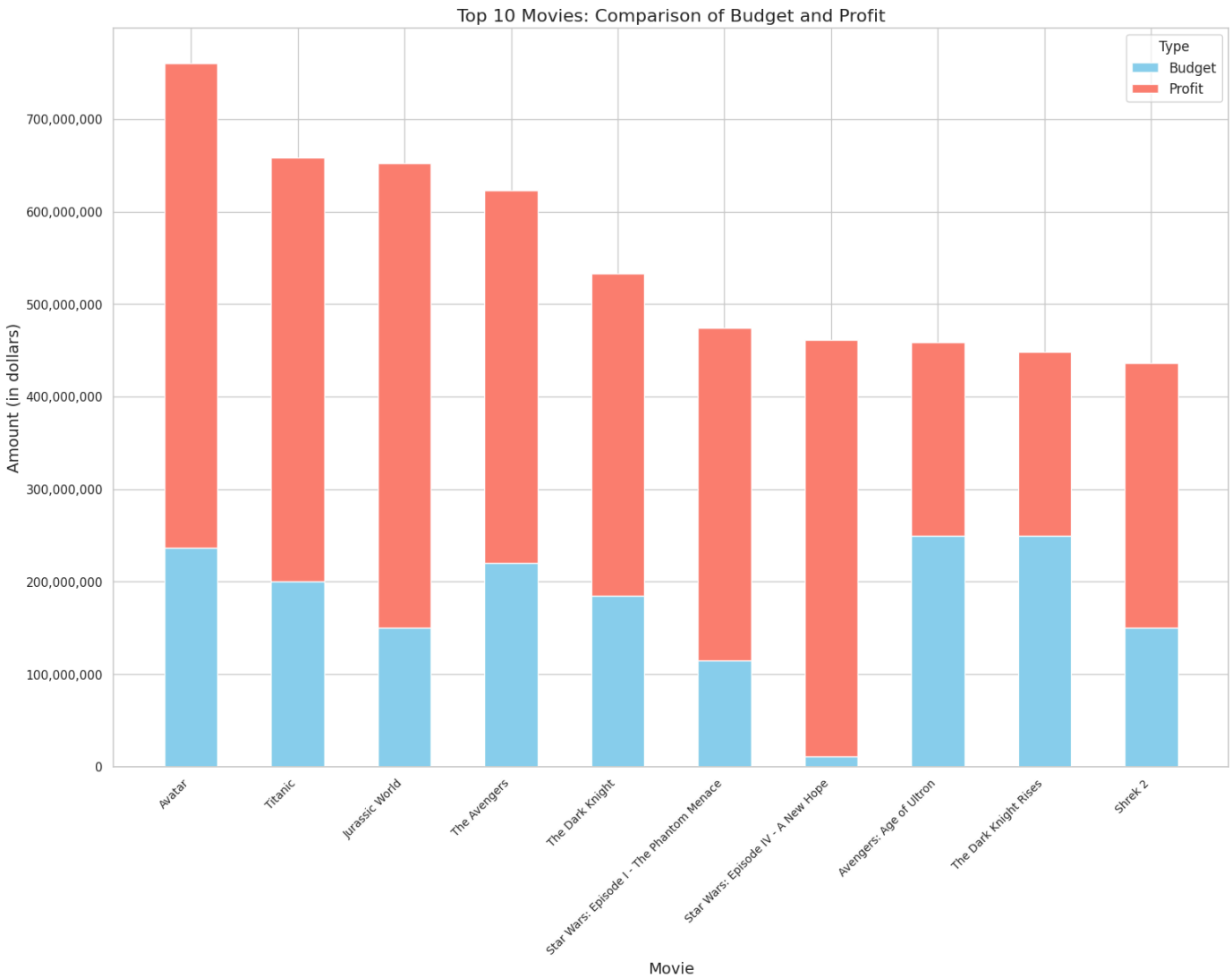

```
import seaborn as sns
import matplotlib.pyplot as plt

# Group by 'title_year' and count the number of 'movie_title', then sort by 'movie_count' in descending order
```

```
releasedate = df.groupby('title_year')['movie_title'].count().sort_values(ascending=False).reset_index(name='movie_count')

# Plot the number of movies released by year using a line plot
plt.figure(figsize=(12, 6))
sns.lineplot(x='title_year', y='movie_count', data=releasedate, marker='o', linestyle='-', color='r')  # Add markers and set lin

# Add titles and labels
plt.title('Number of Movies Released by Year', fontsize=16)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Number of Movies', fontsize=14)

# Customize the x-axis ticks for better readability
plt.xticks(rotation=45, fontsize=10)  # Rotate x-axis labels and set font size

# Set the x-axis tick frequency
plt.gca().xaxis.set_major_locator(plt.MaxNLocator(integer=True))  # Ensure ticks are integers

# Add grid for better readability
plt.grid(True)

# Show the plot
plt.show()
```




```
# Calculate total amount (budget + profit) and sort by it
df['total'] = df['budget'] + df['profit']
df_sorted = df.sort_values(by='total', ascending=False).head(10)

# Set the figure size
plt.figure(figsize=(18, 12))

# Create a stacked bar plot
bar_width = 0.5
movies = df_sorted['movie_title']
budget = df_sorted['budget']
profit = df_sorted['profit']

# Plot budget
plt.bar(movies, budget, bar_width, label='Budget', color='skyblue')
```

```
# Plot profit on top of budget
plt.bar(movies, profit, bar_width, bottom=budget, label='Profit', color='salmon')

# Customize the plot
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f"{x:,.0f}"))


plt.title('Top 10 Movies: Comparison of Budget and Profit', fontsize=16)
plt.xlabel('Movie', fontsize=14)
plt.ylabel('Amount (in dollars)', fontsize=14)
plt.xticks(rotation=45, fontsize=10, ha='right')
plt.legend(title='Type', fontsize=12)
plt.grid(True)

# Show the plot
plt.show()
```




```
# Group by director and get the highest IMDb score along with the movie title
director_movies = df.loc[df.groupby('director_name')['imdb_score'].idxmax()][['director_name', 'movie_title', 'imdb_score']]
director_movies = director_movies.sort_values(by='imdb_score', ascending=False)[:10]

# Plot with kind 'barh'
plt.figure(figsize=(25, 15))
sns.barplot(x='imdb_score', y='director_name', data=director_movies, palette='magma', width=0.6)
```

```
# Plot labels
plt.title("Top 10 Directors by Highest IMDb Score", fontsize=24)
plt.xlabel('IMDb Score', fontsize=20)
plt.ylabel("Director", fontsize=20)

# Annotate with movie titles
for index, value in enumerate(director_movies['imdb_score']):
    plt.text(value, index, f" {director_movies['movie_title'].iloc[index]}", fontsize=16, va='center', color='black')

# Add gridlines for better readability
plt.grid(axis='x', linestyle='--', alpha=0.7)

sns.set_style("whitegrid")
plt.show()
```

```
<ipython-input-103-a6af04621b28>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and

  sns.barplot(x='imdb_score', y='director_name', data=director_movies, palette='magma', width=0.6)
```