MICROSOFT MOVIE ANALYSIS

Final Project Submission

Please fill out:

- Student name: Millicent Muthomi
- Student pace: self paced / part time / full time: Part time
- Scheduled project review date/time: 18/02/2024
- Instructor name: Noah Kandie
- Blog post URL:

Overview.

Microsoft have decided to create a new movie studio and require more insight into which types of films are doing best at the box office. This project uses descriptive statistical analysis on data gathered from IMDb website to gain insight into which genres are most popular. Three seperate datasets were used for this analysis to gain insight into the top 5 most and least rated movies while taking note of their genres, genres of movies that topped the domestic gross sales, foreign gross sales and which genres had the top average ratings. The results of the top genres in Domestic Sales, Foreign Sales and number of productions was clearly the Adventure, Drama, Comedy and Action with adventure being present in the majority of the top categories. My recommendation for which type of Movie to produce would be Adventure, Drama, Comedy or Action as this is the most predominant genre in the analysis. I would advise Microsoft to produce a movie that belong to the Adventure, Drama or a combination of both incorporating Comedy and Action into it. The movie should also idearly last for less than 120 minutes.

Business Problem

Microsoft want to produce movies that are going to be successful in order to make profits, they want to know which types of movies are the most successful. To answer that question both Domestic and Foreign Sales data was analysed to see the most financially successful genres, along with the average rating given and number of votes for each type or genre of movie to see how popularity compared with financial success.

Data Understanding

In [1]:
```
# Your code here - remember to use markdown cells for comments as well!
import pandas as pd
import csv
import json
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
f = r"C:\Users\hp\Desktop\Flat iron\DSCPhase1Project\zippedData\bom.movie_gross.csv\bom.movie_gross.csv"
df1 = pd.read_csv(f)
df1
```

Out[2]:

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

3387 rows × 5 columns

In [3]:
```
m = r"C:\Users\hp\Desktop\Flat iron\DSCPhase1Project\zippedData\imdb.title.basics.csv\title.basics.csv"
df2 = pd.read_csv(m)
df2
```

Out[3]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	NaN
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

146144 rows × 6 columns

In [4]:
```
k = r"C:\Users\hp\Desktop\Flat iron\DSCPhase1Project\zippedData\imdb.title.ratings.csv.gz"
df3 = pd.read_csv(k)
df3
```

Out[4]:

	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
...	...	...	...
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 × 3 columns

The data analysed came from IMDb website. IMDb (an acronym for Internet Movie Database) is a popular worldwide online database of infomation relating to all movies, television programs, video games and streaming content online. I used 3 files from IMDb to answer the question of which genres were most successful, mainly focusing on the Domestic and Foreign Gross sales along with average ratings given and number of votes received.

Merging

In [5]:
```
#Lets try to join df2 and df3 together.
#Looking at the columns they have in common, they both have tconst in common, lets use that.
merged_df2_df3 = pd.merge(df2, df3, on='tconst')
merged_df2_df3.head()
```

Out[5]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	numvotes
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	7.0	77
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	7.2	43
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	6.9	4517
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama	6.1	13
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	6.5	119

In [6]:
```
merged_df2_df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 73856 entries, 0 to 73855
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   tconst           73856 non-null  object
 1   primary_title    73856 non-null  object
 2   original_title   73856 non-null  object
 3   start_year       73856 non-null  int64
 4   runtime_minutes  66236 non-null  float64
 5   genres           73052 non-null  object
 6   averagerating    73856 non-null  float64
 7   numvotes         73856 non-null  int64
dtypes: float64(2), int64(2), object(4)
memory usage: 5.1+ MB
```

In [7]:
```
merged_df2_df3.shape
```

Out[7]:
```
(73856, 8)
```

Lets try merge df1 and df2. The common column between the two is the title column but they are just named differently in each data set. Lets try convert the column title in df1 to primary_title.

In [8]:
```
df1 = df1.rename(columns={'title': 'primary_title'})
df1.head()
```

Out[8]:

	primary_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

Now lets merge everything together

In [9]:
```
data_set = pd.merge(merged_df2_df3, df1, on='primary_title', how='inner')
data_set.head(10)
```

Out[9]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	numvotes	studio	domestic_gross
0	tt0315642	Wazir	Wazir	2016	103.0	Action,Crime,Drama	7.1	15378	Relbig.	1100000.C
1	tt0337692	On the Road	On the Road	2012	124.0	Adventure,Drama,Romance	6.1	37886	IFC	744000.C
2	tt4339118	On the Road	On the Road	2014	89.0	Drama	6.0	6	IFC	744000.C
3	tt5647250	On the Road	On the Road	2016	121.0	Drama	5.7	127	IFC	744000.C
4	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Drama	7.3	275300	Fox	58200000.C
5	tt0365907	A Walk Among the Tombstones	A Walk Among the Tombstones	2014	114.0	Action,Crime,Drama	6.5	105116	Uni.	26300000.C
6	tt0369610	Jurassic World	Jurassic World	2015	124.0	Action,Adventure,Sci-Fi	7.0	539338	Uni.	652300000.C
7	tt0372538	Spy	Spy	2011	110.0	Action,Crime,Drama	6.6	78	Fox	110800000.C
8	tt3079380	Spy	Spy	2015	119.0	Action,Comedy,Crime	7.0	213908	Fox	110800000.C
9	tt0376136	The Rum Diary	The Rum Diary	2011	119.0	Comedy,Drama	6.2	94787	FD	13100000.C

In [10]:
```
data_set.tail()
```

Out[10]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	numvotes	studio	domestic_gross
3022	tt8331988	The Chambermaid	La camarista	2018	102.0	Drama	7.1	147	FM	300.0
3023	tt8404272	How Long Will I Love U	Chao shi kong tong ju	2018	101.0	Romance	6.5	607	WGUSA	747000.0
3024	tt8427036	Helicopter Eela	Helicopter Eela	2018	135.0	Drama	5.4	673	Eros	72000.0
3025	tt9078374	Last Letter	Ni hao, Zhihua	2018	114.0	Drama,Romance	6.4	322	CL	181000.0
3026	tt9151704	Burn the Stage: The Movie	Burn the Stage: The Movie	2018	84.0	Documentary,Music	8.8	2067	Trafalgar	4200000.0

In [11]:
```
data_set.sample()
```

Out[11]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	numvotes	studio	domestic_gros
1365	tt1857913	The Sorcerer and the White Snake	Bai she chuan shuo	2011	100.0	Action,Fantasy,Romance	5.9	7691	Magn.	18800.

Now we have loaded the data set needed.

In [12]:
```
data_set.shape
```

Out[12]: (3027, 12)

After checking the information on each table to see column names and null values, I joined the two datasets, df_titles_basic_info and df_ratings together using the 'tconst' column as it was a unique identifier creating a new dataframe called merged_df2_df3. I then renamed the title column from df1 to primary title so that it may relate to the other data set that has already been merged. I then joined the dataset merged_df2_df3 with the df1 using the primary title as the unique identifier, creating a combined new dataset called data set.

DATA CLEANING

In [13]:
```
data_set.copy()
```

Out[13]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	numvotes	studio	domestic_
0	tt0315642	Wazir	Wazir	2016	103.0	Action,Crime,Drama	7.1	15378	Relbig.	110
1	tt0337692	On the Road	On the Road	2012	124.0	Adventure,Drama,Romance	6.1	37886	IFC	74
2	tt4339118	On the Road	On the Road	2014	89.0	Drama	6.0	6	IFC	74
3	tt5647250	On the Road	On the Road	2016	121.0	Drama	5.7	127	IFC	74
4	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Drama	7.3	275300	Fox	5820
...	...	...	...	...	...	...	...	...	...	
3022	tt8331988	The Chambermaid	La camarista	2018	102.0	Drama	7.1	147	FM	
3023	tt8404272	How Long Will I Love U	Chao shi kong tong ju	2018	101.0	Romance	6.5	607	WGUSA	74
3024	tt8427036	Helicopter Eela	Helicopter Eela	2018	135.0	Drama	5.4	673	Eros	7.
3025	tt9078374	Last Letter	Ni hao, Zhihua	2018	114.0	Drama,Romance	6.4	322	CL	18
3026	tt9151704	Burn the Stage: The Movie	Burn the Stage: The Movie	2018	84.0	Documentary,Music	8.8	2067	Trafalgar	420

3027 rows × 12 columns

Primary and Original titles seem to be important, don't drop. Separate the title column average rating and num votes. Capitalize Title columns. Can we assume the column year means production year? The numbers should have a comm

In [14]:
```
#Handling missing values.
#Identify whether the data set has missing values.
data_set.isnull().sum()
```

Out[14]:
```
tconst              0
primary_title       0
original_title      0
start_year          0
runtime_minutes    47
genres              7
averagerating       0
numvotes            0
studio              3
domestic_gross     22
foreign_gross    1195
year                0
dtype: int64
```

In [15]:
```
data_set.isnull().mean()
```

Out[15]:
```
tconst             0.000000
primary_title      0.000000
original_title     0.000000
start_year         0.000000
runtime_minutes    0.015527
genres             0.002313
averagerating      0.000000
numvotes           0.000000
studio             0.000991
domestic_gross     0.007268
foreign_gross      0.394780
year               0.000000
dtype: float64
```

In [16]:
```
#for the null data sets, the highest percentge is foreign gross which is 39%. Because thepercentage isn't big, we ca
#Normaly i would drop them but instead i want to keep all the data as it will be relevant in my data analysis and in
#all the missing values with its mean
data_set = data_set.fillna(data_set.mean())
data_set
```

Out[16]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	numvotes	studio	domestic_
0	tt0315642	Wazir	Wazir	2016	103.0	Action,Crime,Drama	7.1	15378	Relbig.	110
1	tt0337692	On the Road	On the Road	2012	124.0	Adventure,Drama,Romance	6.1	37886	IFC	74
2	tt4339118	On the Road	On the Road	2014	89.0	Drama	6.0	6	IFC	74
3	tt5647250	On the Road	On the Road	2016	121.0	Drama	5.7	127	IFC	74
4	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Drama	7.3	275300	Fox	5820
...	...	...	...	...	...	...	...	...	...	
3022	tt8331988	The Chambermaid	La camarista	2018	102.0	Drama	7.1	147	FM	
3023	tt8404272	How Long Will I Love U	Chao shi kong tong ju	2018	101.0	Romance	6.5	607	WGUSA	74
3024	tt8427036	Helicopter Eela	Helicopter Eela	2018	135.0	Drama	5.4	673	Eros	7
3025	tt9078374	Last Letter	Ni hao, Zhihua	2018	114.0	Drama,Romance	6.4	322	CL	18
3026	tt9151704	Burn the Stage: The Movie	Burn the Stage: The Movie	2018	84.0	Documentary,Music	8.8	2067	Trafalgar	420

3027 rows × 12 columns

In [17]:
```
#Lets confirm if we have filled all the missing values
data_set.isnull().sum()
```

Out[17]:
```
tconst               0
primary_title        0
original_title       0
start_year           0
runtime_minutes      0
genres               7
averagerating        0
numvotes             0
studio               3
domestic_gross       0
foreign_gross     1195
year                 0
dtype: int64
```

In [18]:
```
# The reason that they are still null values is because these items are object data types.
#lets first convert foreighn gross column into a float.
# data_set['foreign_gross'] = data_set['foreign_gross'].astype('int')
# data_set.info

#Lets first try to identify the unique values
print(data_set['foreign_gross'].unique())
```

```
[nan '8000000' '129900000' ... '49400000' '542100000' '82100000']
```

In [19]:
```
#since we have a nan value, we will have to convert it to NaN value then convert them to the mean.
data_set['foreign_gross'] = pd.to_numeric(data_set['foreign_gross'], errors='coerce')
data_set['foreign_gross'].fillna(data_set['foreign_gross'].mean(), inplace=True)
data_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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  3027 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   3027 non-null   float64
 10  foreign_gross    3027 non-null   float64
 11  year             3027 non-null   int64
dtypes: float64(4), int64(3), object(5)
memory usage: 307.4+ KB
```

We were able to convert the data set (foreign gross) into an float and fill the missing values with the mean

In [20]:
```
#Lets fill in the missing values on the studio and genres with the most repeated/common values.
most_common_studio = data_set['studio'].value_counts().idxmax()
count = data_set['studio'].value_counts().max()

print(most_common_studio)
print(count)
```

```
Uni.
156
```

In [21]:
```
#Lets fill in the genres column with the most common genres.
most_common_genres = data_set['genres'].value_counts().idxmax()
count_1 = data_set['genres'].value_counts().max()

print(most_common_genres)
print(count_1)
```

```
Drama
317
```

In [22]:
```
#Lets fill in the missing values with the mode of each column.
data_set['genres'] = data_set['genres'].fillna('Drama')
data_set['studio'] = data_set['studio'].fillna('Uni')

data_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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  3027 non-null   float64
 5   genres           3027 non-null   object 
 6   averagerating    3027 non-null   float64
 7   numvotes         3027 non-null   int64  
 8   studio           3027 non-null   object 
 9   domestic_gross   3027 non-null   float64
 10  foreign_gross    3027 non-null   float64
 11  year             3027 non-null   int64  
dtypes: float64(4), int64(3), object(5)
memory usage: 307.4+ KB
```

There are no missing values

DROPPING IRRELEVANT VALUES

In [23]:
```
#Lets drop the original title and studio columns as they seem to be irrelevant.
data_set.drop(columns=['original_title'], inplace=True)
data_set.head()
```

Out[23]:

	tconst	primary_title	start_year	runtime_minutes	genres	averagerating	numvotes	studio	domestic_gross	foreign_gros
0	tt0315642	Wazir	2016	103.0	Action,Crime,Drama	7.1	15378	Relbig.	1100000.0	7.843093e+(
1	tt0337692	On the Road	2012	124.0	Adventure,Drama,Romance	6.1	37886	IFC	744000.0	8.000000e+(
2	tt4339118	On the Road	2014	89.0	Drama	6.0	6	IFC	744000.0	8.000000e+(
3	tt5647250	On the Road	2016	121.0	Drama	5.7	127	IFC	744000.0	8.000000e+(
4	tt0359950	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Drama	7.3	275300	Fox	58200000.0	1.299000e+(

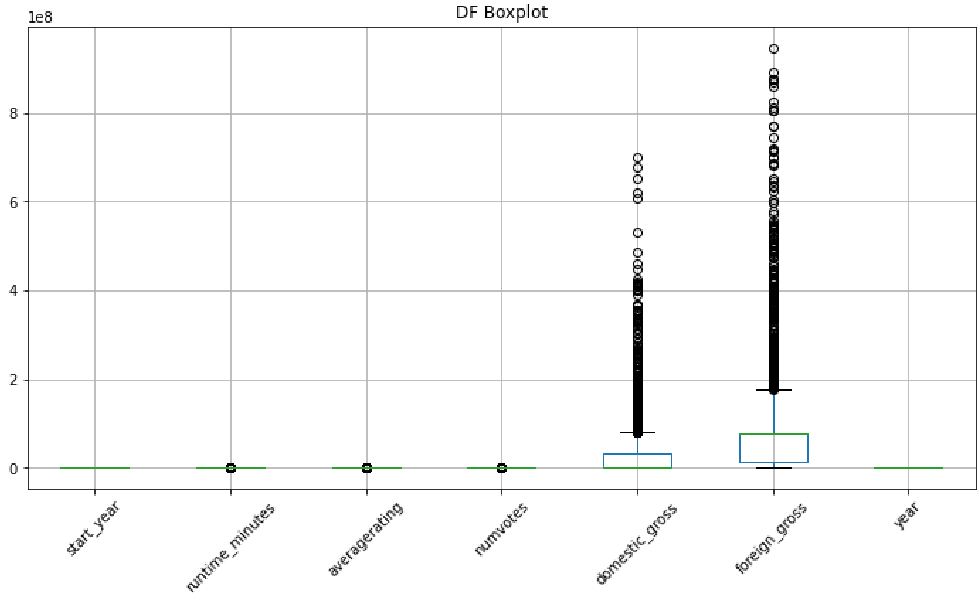
In [24]:
```
#Lets check for duplicates.
data_set.duplicated()
```

Out[24]:
```
0       False
1       False
2       False
3       False
4       False
        ...  
3022    False
3023    False
3024    False
3025    False
3026    False
Length: 3027, dtype: bool
```

There are no duplicated values

Handling outliers

In [25]:
```
#lets first find ou if we have outliers by constructing a boxplot.
# Create a larger figure
plt.figure(figsize=(12, 6))

# Create a boxplot of the DataFrame with rotated x-axis labels
data_set.boxplot()
plt.xticks(rotation=45)

# Add a title to the plot
plt.title('DF Boxplot')

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



From the above analysis, we can see that outliers are on the domestic gross and foreign gross column. There are two ways of getting rid of outliers.

z score - Normal distribution. IQR - skewed distribution.

In [26]:
```
data_set.domestic_gross.skew()
```

Out[26]: 4.167477501645865

In [27]:
```
data_set.foreign_gross.skew()
```

Out[27]: 3.8524090441455057

From the above analysis on skewness, we see that both outliers have a skewed distribution. I will however not drop any outliers as all values have a possibility of happening in the real world.

Feature Engineering.

In [28]:
```
#lets capitalize the Column titles.
data_set.columns = data_set.columns.str.capitalize()
data_set.head()
```

Out[28]:

	Tconst	Primary_title	Start_year	Runtime_minutes	Genres	Averagerating	Numvotes	Studio	Domestic_gross	Foreign_g
0	tt0315642	Wazir	2016	103.0	Action,Crime,Drama	7.1	15378	Relbig.	1100000.0	7.843093e
1	tt0337692	On the Road	2012	124.0	Adventure,Drama,Romance	6.1	37886	IFC	744000.0	8.000000e
2	tt4339118	On the Road	2014	89.0	Drama	6.0	6	IFC	744000.0	8.000000e
3	tt5647250	On the Road	2016	121.0	Drama	5.7	127	IFC	744000.0	8.000000e
4	tt0359950	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Drama	7.3	275300	Fox	58200000.0	1.299000e

In [29]:
```
data_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3027 entries, 0 to 3026
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Tconst           3027 non-null   object
 1   Primary_title    3027 non-null   object
 2   Start_year       3027 non-null   int64
 3   Runtime_minutes  3027 non-null   float64
 4   Genres           3027 non-null   object
 5   Averagerating    3027 non-null   float64
 6   Numvotes         3027 non-null   int64
 7   Studio           3027 non-null   object
 8   Domestic_gross   3027 non-null   float64
 9   Foreign_gross    3027 non-null   float64
 10  Year             3027 non-null   int64
dtypes: float64(4), int64(3), object(4)
memory usage: 283.8+ KB
```

In [30]:
```
#the average rating and Num votes should be two separate word.
data_set.rename(columns={'Averagerating': 'Average_Rating'}, inplace=True)
data_set.rename(columns={'Numvotes': 'Num_votes'}, inplace=True)
data_set.head()
```

Out[30]:

	Tconst	Primary_title	Start_year	Runtime_minutes	Genres	Average_Rating	Num_votes	Studio	Domestic_gross	Foreigr
0	tt0315642	Wazir	2016	103.0	Action,Crime,Drama	7.1	15378	Relbig.	1100000.0	7.8430
1	tt0337692	On the Road	2012	124.0	Adventure,Drama,Romance	6.1	37886	IFC	744000.0	8.0000
2	tt4339118	On the Road	2014	89.0	Drama	6.0	6	IFC	744000.0	8.0000
3	tt5647250	On the Road	2016	121.0	Drama	5.7	127	IFC	744000.0	8.0000
4	tt0359950	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Drama	7.3	275300	Fox	58200000.0	1.2990

In [31]:
```
data_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3027 entries, 0 to 3026
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Tconst           3027 non-null   object
 1   Primary_title    3027 non-null   object
 2   Start_year       3027 non-null   int64
 3   Runtime_minutes  3027 non-null   float64
 4   Genres           3027 non-null   object
 5   Average_Rating   3027 non-null   float64
 6   Num_votes        3027 non-null   int64
 7   Studio           3027 non-null   object
 8   Domestic_gross   3027 non-null   float64
 9   Foreign_gross    3027 non-null   float64
 10  Year             3027 non-null   int64
dtypes: float64(4), int64(3), object(4)
memory usage: 283.8+ KB
```

In [32]:
```
# #LETS PROPERLY ALIGN THE DOMESTIC AND FOREIGN COLUMN INTO THOUSANDS
# data_set['Domestic_gross'] = data_set['Domestic_gross'].apply('{:,.0f}'.format)
# data_set['Foreign_gross'] = data_set['Foreign_gross'].apply('{:,.0f}'.format)
# data_set
```

Checking the information on the new dataframe, I then cleaned up the null values by filling them with the mean, for the non numeric values, filled it with the most common values, tidied up the "Domestic Gross' and 'Foreign Gross' columns and added the commas for easier readability and analysis. The column "original title" was deleted it was required to carry out this analysis.

EXPLORATORY DATA ANALYSIS.

I) What is the most popular movie

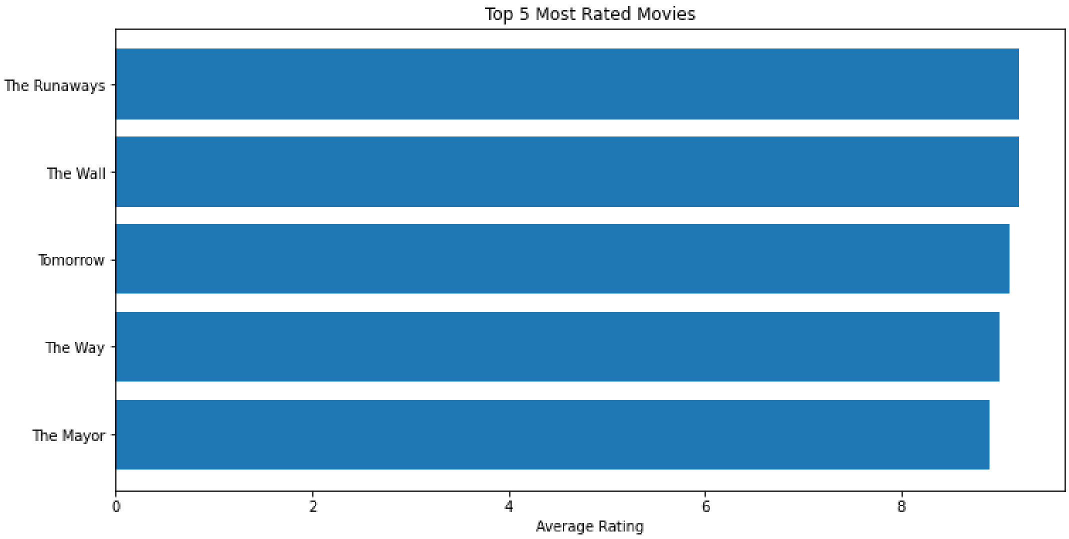
In [33]:
```
most_rated_movie = data_set[data_set['Average_Rating'] == data_set['Average_Rating'].max()]

# Print the most rated movie
print("The most rated movie is:")
print(most_rated_movie[['Primary_title', 'Average_Rating', 'Genres', 'Runtime_minutes']])
```

```
The most rated movie is:
    Primary_title  Average_Rating       Genres  Runtime_minutes
173  The Runaways             9.2    Adventure            108.0
658      The Wall             9.2  Documentary             78.0
```

In [34]:
```
top_5_most_rated_movies = data_set.sort_values(by='Average_Rating', ascending=False).head(5)

# Print the top 5 most rated movies
print("Top 5 most rated movies:")
print(top_5_most_rated_movies[['Primary_title', 'Average_Rating', 'Genres', 'Runtime_minutes']])
```

```
Top 5 most rated movies:
     Primary_title  Average_Rating                    Genres  Runtime_minutes
173   The Runaways             9.2                 Adventure            108.0
658       The Wall             9.2               Documentary             78.0
2039      Tomorrow             9.1                     Drama            115.0
638        The Way             9.0               Documentary             85.0
1186     The Mayor             8.9  Comedy,Documentary,Drama             68.0
```

In [35]:
```
titles = top_5_most_rated_movies['Primary_title']
ratings = top_5_most_rated_movies['Average_Rating']

# Creating a bar chart
plt.figure(figsize=(12, 6))
plt.barh(titles, ratings)
plt.xlabel('Average Rating')
plt.title('Top 5 Most Rated Movies')
plt.gca().invert_yaxis()  # Invert y-axis to display the highest rating at the top
plt.show()
```



- From the analysis above, we can see that the highest rated movie is The Runaways and The Wall.
  - The Runaways is an Adevnture movie while The Wall is a Documentary.
  - Both have a runtime of less than 120 minutes. The top 5 most rated movies also have a runtime of less than 120 minutes. We will explore the least rated movies and look at their runtime.

LEAST RATED MOVIES

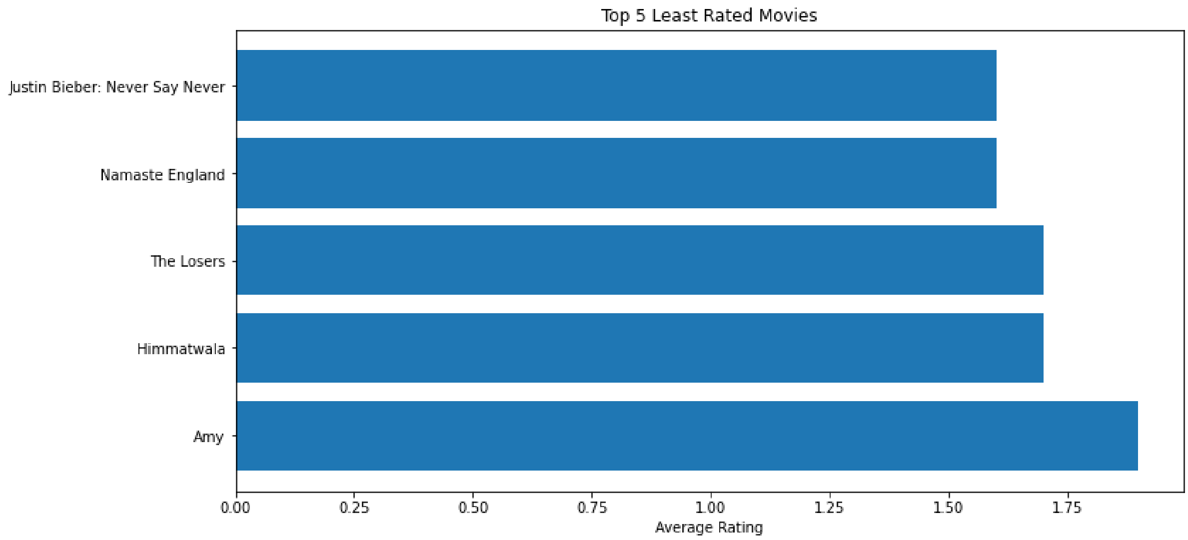
In [36]:
```
least_rated_movie = data_set[data_set['Average_Rating'] == data_set['Average_Rating'].min()]

# Print the least rated movie
print("The least rated movie is:")
print(least_rated_movie[['Primary_title', 'Average_Rating', 'Genres', 'Runtime_minutes']])
```

```
The least rated movie is:
                       Primary_title  Average_Rating               Genres  \
1110  Justin Bieber: Never Say Never             1.6     Documentary,Music
3002                 Namaste England             1.6  Comedy,Drama,Romance

      Runtime_minutes
1110            105.0
3002            141.0
```

In [37]:
```
least_rated_movies = data_set.sort_values(by='Average_Rating', ascending=True).head(5)

# Print the least rated movies
print("Least rated movies:")
print(least_rated_movies[['Primary_title', 'Average_Rating', 'Genres', 'Runtime_minutes']])
```

```
Least rated movies:
                       Primary_title  Average_Rating                Genres  \
1110  Justin Bieber: Never Say Never             1.6      Documentary,Music
3002                 Namaste England             1.6   Comedy,Drama,Romance
60                        The Losers             1.7                  Drama
1843                      Himmatwala             1.7    Action,Comedy,Drama
2119                             Amy             1.9                 Horror

      Runtime_minutes
1110            105.0
3002            141.0
60              112.0
1843            150.0
2119             94.0
```

In [38]:
```
titles = least_rated_movies['Primary_title']
ratings = least_rated_movies['Average_Rating']

# Creating a bar chart
plt.figure(figsize=(12, 6))
plt.barh(titles, ratings)
plt.xlabel('Average Rating')
plt.title('Top 5 Least Rated Movies')
plt.gca().invert_yaxis()  # Invert y-axis to display the highest rating at the top
plt.show()
```



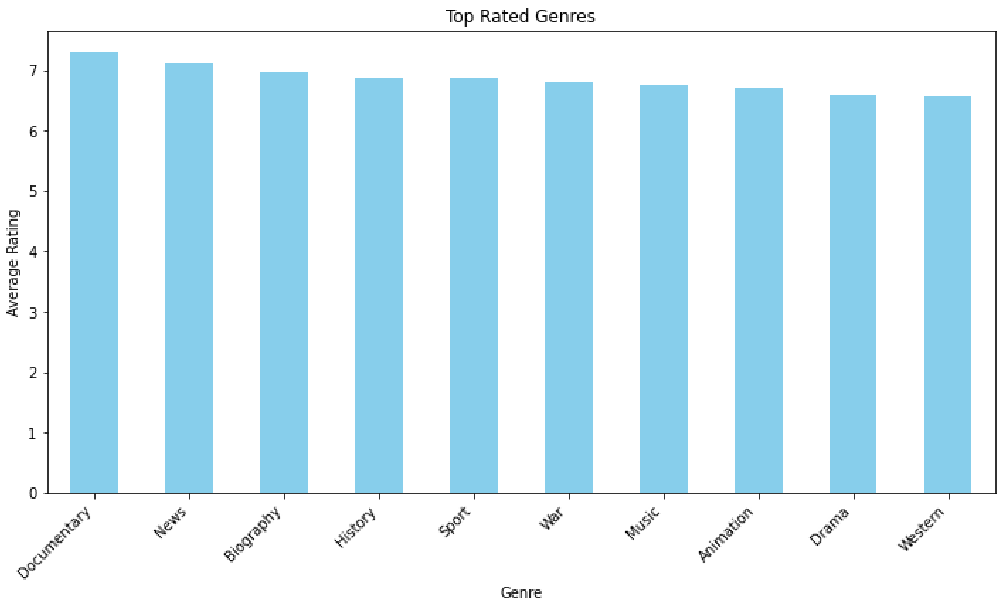
- From the above, we see that the least rated movies are the Justin Bieber: Never Say Never movie and Namaste England movie.
- The belong to the Documentary, Music, Comedy, Drama, Romance Genres.
- We have movies that have a runtime of more than 120 minutes, maybe that could be a factor.

LETS LOOK AT THE MOST RATED GENRES.

In [39]:
```
# Split the 'genres' column and create a new DataFrame with one genre per row
genres_df = data_set['Genres'].str.split(',', expand=True).stack().reset_index(level=1, drop=True).rename('genre')
genres_df = genres_df.str.strip()  # Remove leading and trailing whitespace

# Merge the genres DataFrame with the original DataFrame
data_set_genres = data_set.merge(genres_df, left_index=True, right_index=True)

# Calculate the average rating for each genre
genre_avg_rating = data_set_genres.groupby('genre')['Average_Rating'].mean().sort_values(ascending=False)
genre_avg_rating
```

Out[39]:
```
genre
Documentary    7.292511
News           7.100000
Biography      6.973333
History        6.878676
Sport          6.867925
War            6.801961
Music          6.756522
Animation      6.700000
Drama          6.587181
Western        6.561905
Crime          6.479581
Adventure      6.478360
Sci-Fi         6.451111
Romance        6.335470
Musical        6.316667
Action         6.275232
Mystery        6.274879
Comedy         6.247624
Fantasy        6.242353
Family         6.224786
Thriller       6.172627
Horror         5.684583
Name: Average_Rating, dtype: float64
```

In [40]:
```
# Plotting the top rated genres
plt.figure(figsize=(12, 6))
genre_avg_rating.head(10).plot(kind='bar', color='skyblue')
plt.xlabel('Genre')
plt.ylabel('Average Rating')
plt.title('Top Rated Genres')
plt.xticks(rotation=45, ha='right')
plt.show()
```



From the analysis below, the top 5 genres to get into are

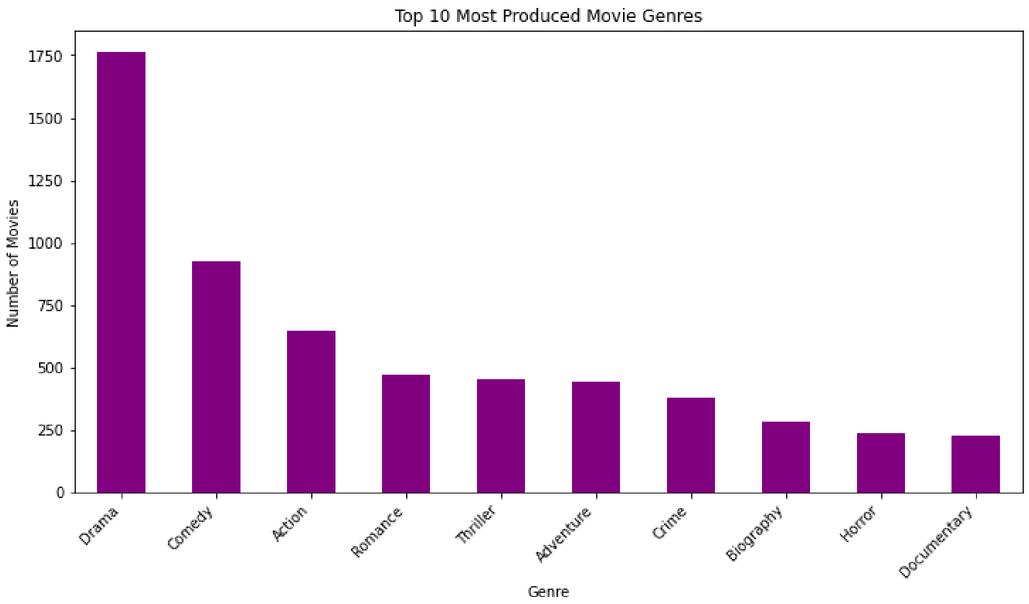
- Documentaries
- News
- Biographies
- History
- Sports

LETS LOOK AT THE MOST PRODUCED GENRES

In [41]:
```
genres_df = data_set['Genres'].str.split(',', expand=True).stack().reset_index(level=1, drop=True).rename('genre')
genres_df = genres_df.str.strip()

# Count the occurrences of each genre
top_genres = genres_df.value_counts().nlargest(10)

print("Top 5 most produced movie genres:")
print(top_genres)
```

```
Top 5 most produced movie genres:
Drama          1763
Comedy          926
Action          646
Romance         468
Thriller        453
Adventure       439
Crime           382
Biography       285
Horror          240
Documentary     227
Name: genre, dtype: int64
```

In [42]:
```
# Plot the top genres
plt.figure(figsize=(12, 6))
top_genres.plot(kind='bar', color='purple')
plt.xlabel('Genre')
plt.ylabel('Number of Movies')
plt.title('Top 10 Most Produced Movie Genres')
plt.xticks(rotation=45, ha='right')
plt.show()
```



From the data above, the following are the most produced movie genres over the years

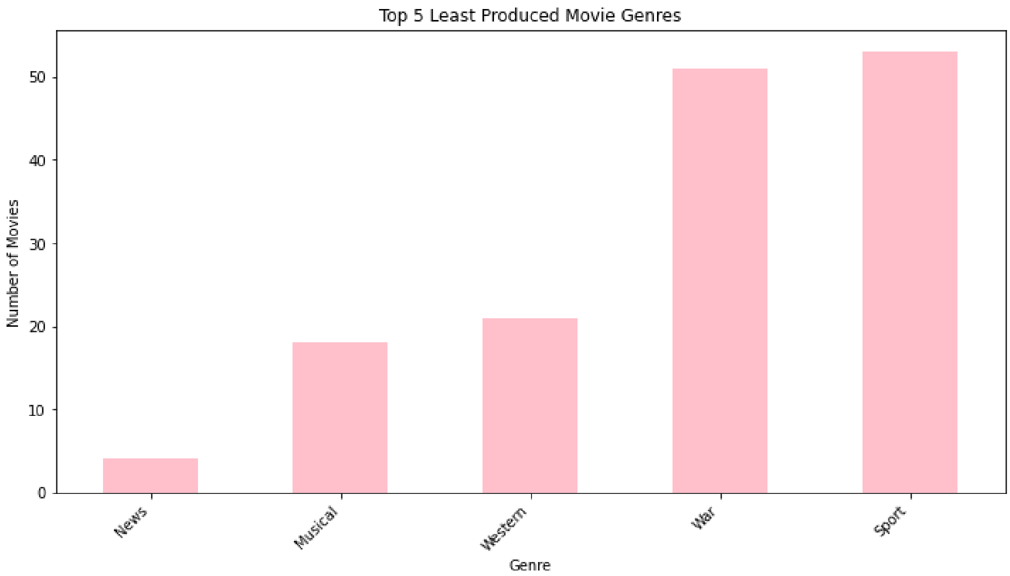
- Drama
- Comedy
- Action
- Romance
- Thriller

LEAST PRODUCED GENRES

In [43]:
```
# Split the 'Genres' column and stack them
genres_df = data_set['Genres'].str.split(',', expand=True).stack().reset_index(level=1, drop=True).rename('genre')
genres_df = genres_df.str.strip()

# Count the occurrences of each genre
least_produced_genres = genres_df.value_counts().nsmallest(5)
least_produced_genres
```

Out[43]:
```
News         4
Musical     18
Western     21
War         51
Sport       53
Name: genre, dtype: int64
```

In [44]:
```
plt.figure(figsize=(12, 6))
least_produced_genres.plot(kind='bar', color='pink')
plt.xlabel('Genre')
plt.ylabel('Number of Movies')
plt.title('Top 5 Least Produced Movie Genres')
plt.xticks(rotation=45, ha='right')
plt.show()
```



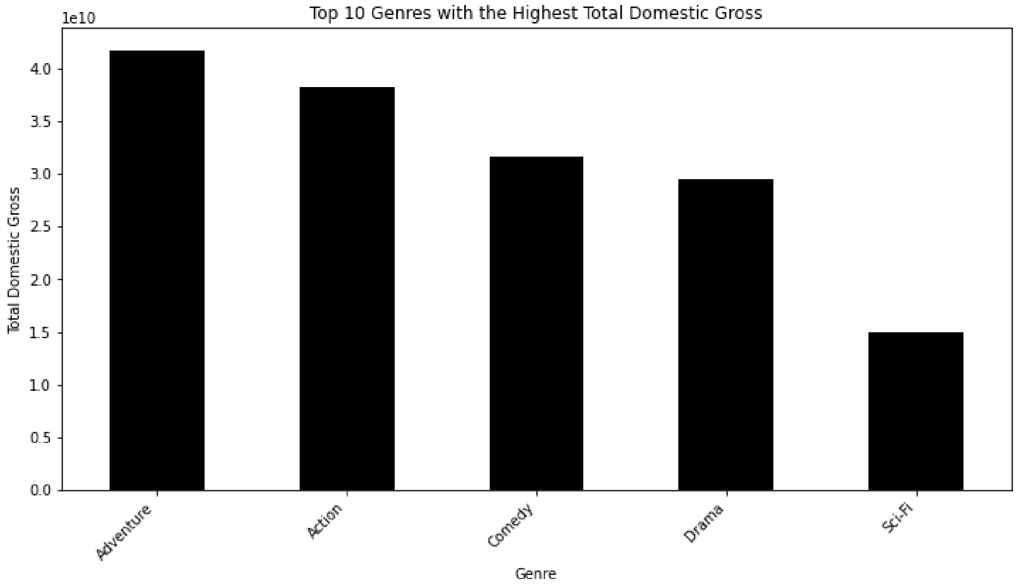
From the data above, although genres such as News, war, sports were among the most rated genres, they are the least produced.

TOP 5 GENRES TO YIELD THE HIGHEST DOMESTIC GROSS

In [57]:
```
# Split the 'Genres' column into individual genres
genres_df = data_set['Genres'].str.split(',', expand=True)

# Stack the genres and reset the index
genres_stacked = genres_df.stack().reset_index(level=1, drop=True).rename('genre')

# Merge the stacked genres back to the original DataFrame
data_set_split = data_set.merge(genres_stacked, left_index=True, right_index=True)

# Group by genre and sum the domestic gross for each genre
genre_domestic_gross = data_set_split.groupby('genre')['Domestic_gross'].sum()

# Select the top 5 genres with the highest total domestic gross
top_5_genres_domestic_gross = genre_domestic_gross.nlargest(5)
top_5_genres_domestic_gross
```

Out[57]:
```
genre
Adventure    4.176354e+10
Action       3.823409e+10
Comedy       3.164528e+10
Drama        2.940409e+10
Sci-Fi       1.498404e+10
Name: Domestic_gross, dtype: float64
```

In [58]:
```
plt.figure(figsize=(12, 6))
top_5_genres_domestic_gross.plot(kind='bar', color='black')
plt.xlabel('Genre')
plt.ylabel('Total Domestic Gross')
plt.title('Top 10 Genres with the Highest Total Domestic Gross')
plt.xticks(rotation=45, ha='right')
plt.show()
```

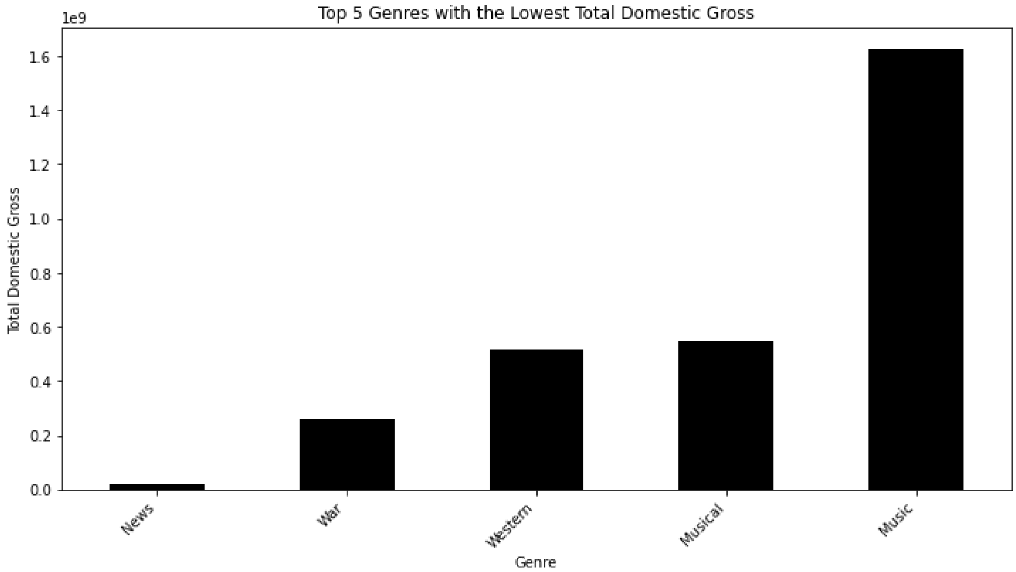


GENRES WITH THE LEAST DOMESTIC GROSS

In [47]:
```
# Split the 'Genres' column into individual genres
genres_df = data_set['Genres'].str.split(',', expand=True)

# Stack the genres and reset the index
genres_stacked = genres_df.stack().reset_index(level=1, drop=True).rename('genre')

# Merge the stacked genres back to the original DataFrame
data_set_split = data_set.merge(genres_stacked, left_index=True, right_index=True)

# Group by genre and sum the domestic gross for each genre
genre_domestic_gross = data_set_split.groupby('genre')['Domestic_gross'].sum()

# Select the top 5 genres with the highest total domestic gross
top_5_genres_domestic_gross = genre_domestic_gross.nsmallest(5)
top_5_genres_domestic_gross
```

Out[47]:
```
genre
News        2.164140e+07
War         2.604493e+08
Western     5.187837e+08
Musical     5.505853e+08
Music       1.625713e+09
Name: Domestic_gross, dtype: float64
```

In [48]:
```
plt.figure(figsize=(12, 6))
top_5_genres_domestic_gross.plot(kind='bar', color='black')
plt.xlabel('Genre')
plt.ylabel('Total Domestic Gross')
plt.title('Top 5 Genres with the Lowest Total Domestic Gross')
plt.xticks(rotation=45, ha='right')
plt.show()
```



From the data above, though News, Musicals and Western type genres are the most rated, they yield the least domestic gross.

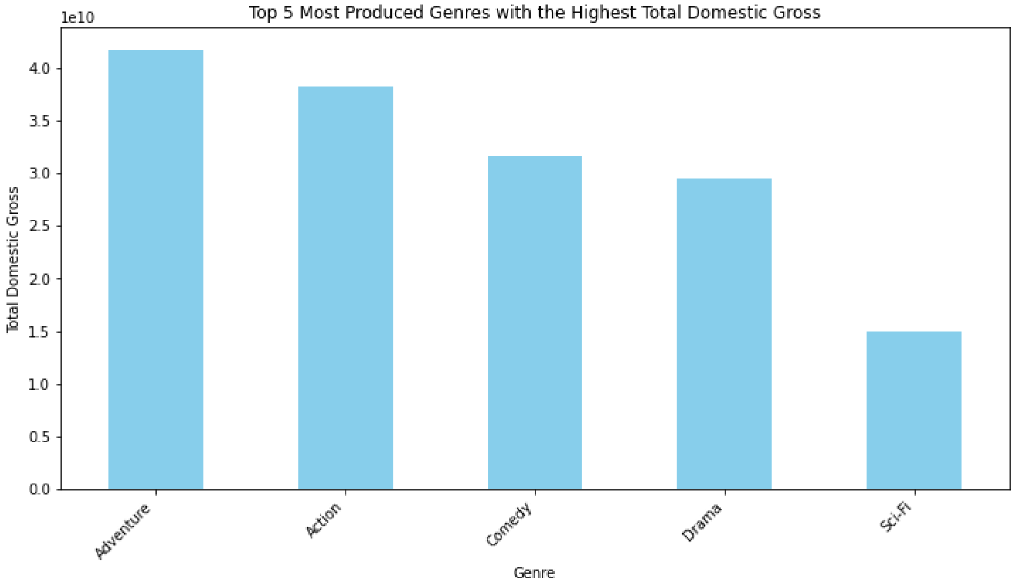
LETS SEE WHICH GENRES ARE THE MOST PRODUCED GENRES WITH THE HIGHEST DOMESTIC GROSS

In [49]:
```
genres_df = data_set['Genres'].str.split(',', expand=True)

# Stack the genres and reset the index
genres_stacked = genres_df.stack().reset_index(level=1, drop=True).rename('genre')

# Merge the stacked genres back to the original DataFrame
data_set_split = data_set.merge(genres_stacked, left_index=True, right_index=True)

# Group by genre and count the number of movies produced and sum the domestic gross for each genre
genre_counts_domestic_gross = data_set_split.groupby('genre').agg({'Start_year': 'count', 'Domestic_gross': 'sum'})

# Rename the columns for clarity
genre_counts_domestic_gross = genre_counts_domestic_gross.rename(columns={'Start_year': 'movie_count'})

# Select the top 5 genres with the highest total domestic gross
top_5_genres_domestic_gross = genre_counts_domestic_gross.nlargest(5, 'Domestic_gross')
top_5_genres_domestic_gross
```

Out[49]:

genre	movie_count	Domestic_gross
Adventure	439	4.176354e+10
Action	646	3.823409e+10
Comedy	926	3.164528e+10
Drama	1763	2.940409e+10
Sci-Fi	135	1.498404e+10

In [50]:
```
plt.figure(figsize=(12, 6))
top_5_genres_domestic_gross['Domestic_gross'].plot(kind='bar', color='skyblue')
plt.xlabel('Genre')
plt.ylabel('Total Domestic Gross')
plt.title('Top 5 Most Produced Genres with the Highest Total Domestic Gross')
plt.xticks(rotation=45, ha='right')
plt.show()
```



From the data, the above genres are the most produced genres in the industry and will yield the most returns. (I have looked at Domestic gross because Microsoft headquarters is based in USA and as a start they would like to earn popularity in North America.)

MOVIE GENRES WITH THE HIGHEST FOREIGN GROSS.

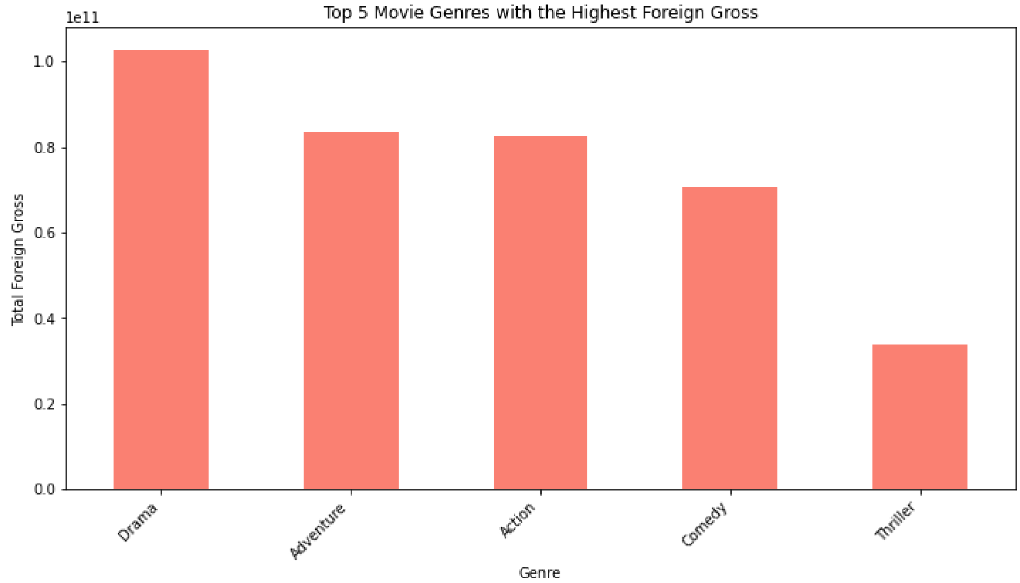
In [51]:
```
# Split the 'Genres' column into individual genres
genres_df = data_set['Genres'].str.split(',', expand=True)

# Stack the genres and merge them back to the original DataFrame
genres_stacked = genres_df.stack().reset_index(level=1, drop=True).rename('genre')
data_set_split = data_set.merge(genres_stacked, left_index=True, right_index=True)

# Group by genre and sum the domestic gross for each genre
genre_domestic_gross = data_set_split.groupby('genre')['Foreign_gross'].sum()

# Select the top 5 genres with the highest total domestic gross
top_5_genres_domestic_gross = genre_domestic_gross.nlargest(5)

# Print the top 5 genres with the highest total domestic gross
print("Top 5 movie genres with the highest domestic gross:")
print(top_5_genres_domestic_gross)
```

```
Top 5 movie genres with the highest domestic gross:
genre
Drama        1.028908e+11
Adventure    8.335772e+10
Action       8.244001e+10
Comedy       7.061427e+10
Thriller     3.392998e+10
Name: Foreign_gross, dtype: float64
```

In [52]:
```
# Group by genre and sum the foreign gross for each genre
genre_foreign_gross = data_set_split.groupby('genre')['Foreign_gross'].sum()

# Select the top 5 genres with the highest total foreign gross
top_5_genres_foreign_gross = genre_foreign_gross.nlargest(5)

# Print the top 5 genres with the highest total foreign gross
print("Top 5 movie genres with the highest foreign gross:")
print(top_5_genres_foreign_gross)
```

```
Top 5 movie genres with the highest foreign gross:
genre
Drama        1.028908e+11
Adventure    8.335772e+10
Action       8.244001e+10
Comedy       7.061427e+10
Thriller     3.392998e+10
Name: Foreign_gross, dtype: float64
```

In [53]:
```
# Plot the top 5 genres with the highest foreign gross
plt.figure(figsize=(12, 6))
top_5_genres_foreign_gross.plot(kind='bar', color='salmon')
plt.xlabel('Genre')
plt.ylabel('Total Foreign Gross')
plt.title('Top 5 Movie Genres with the Highest Foreign Gross')
plt.xticks(rotation=45, ha='right')
plt.show()
```



HOW LONG DO MOST MOVIES LAST

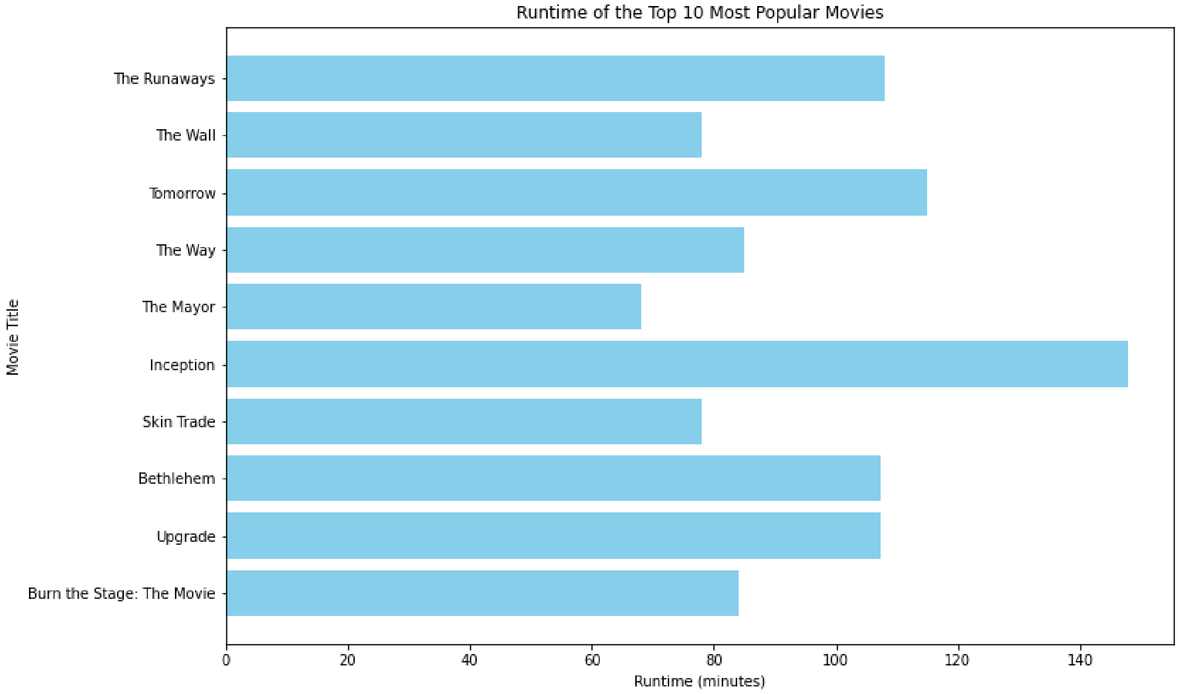
In [54]:
```
# Find the most common runtime minutes for movies
most_common_runtime = data_set['Runtime_minutes'].mode()[0]

print("The most common runtime minutes for movies is:", most_common_runtime, "minutes")
```

```
The most common runtime minutes for movies is: 100.0 minutes
```

In [55]:
```
# Find the top 10 most popular movies based on the highest average rating
top_10_most_popular_movies = data_set.nlargest(10, 'Average_Rating')

# Get the runtime of the top 10 most popular movies
top_10_most_popular_movies_runtime = top_10_most_popular_movies['Runtime_minutes']

print("Runtime of the top 10 most popular movies:")
print(top_10_most_popular_movies_runtime)
#This code will find and print the runtime of the top 10
```

```
Runtime of the top 10 most popular movies:
173     108.000000
658      78.000000
2039    115.000000
638      85.000000
1186     68.000000
514     148.000000
834      78.000000
2150    107.217114
2935    107.217114
3026     84.000000
Name: Runtime_minutes, dtype: float64
```

In [56]:
```
import matplotlib.pyplot as plt

# Find the top 10 most popular movies based on the highest average rating
top_10_most_popular_movies = data_set.nlargest(10, 'Average_Rating')

# Get the movie titles and runtimes
movie_titles = top_10_most_popular_movies['Primary_title']
runtimes = top_10_most_popular_movies['Runtime_minutes']

# Plot the runtime of the top 10 most popular movies
plt.figure(figsize=(12, 8))
plt.barh(movie_titles, runtimes, color='skyblue')
plt.xlabel('Runtime (minutes)')
plt.ylabel('Movie Title')
plt.title('Runtime of the Top 10 Most Popular Movies')
plt.gca().invert_yaxis()  # Invert y-axis to show the highest rating at the top
plt.show()
```



RESULTS

```
 - Top 5 genres with highest foreign gross is Drama, Adventure, Action, Comedy, Thriller
 - Top 5 genres with the highest domestic gross is Adventure, Action, Comedy, Drama, Sci-Fi
 - Top 5 genres that are mostly produced over the years Drama, Comedy, Action, Romance, Thriller.
 - Top 5 genres that are moslty rated Documentary, News, Biography, History, Sports.
 - The Top rated movies had the following genres Adventure, Documentary, Drama and Comedy.
 - Most rated movies do not last more than 120 minutes

Special emphasis on Adevnture and Drama genres that top the highest foreign gross and most produced films.
```

```
CONCLUSION.
From the data collected and analyzed, we can conclude that.
 - The population loves the following genres; Adventure, Drama, Comedy and Action films that appear most in the top lists.
 - The ratings of a movie may not necessarily accurately depict a movies popularity ( Documentaries, News.)
 - Our movie should not last more than 120minutes.

NEXT STEPS.
 - We could consider the age range from which this statistics were collected.
 - We can look at which months was a specific movie genre produced, e.g February, Romace movie because of Valentines.
```

In [ ]:

In [ ]: