Final Project Submission

* Student name: BRIAN CHABARI NYAGAH.

* Student pace: full time

* Scheduled project review date/time:

* Instructor name: William Okomba.

* Blog post URL:

Project Overview

Following the creation of movie studio, we have been tasked by Microsoft, who have no idea about making films, to identify what makes a film perform well at the box office. After identifying return on investment (RoI) as the primary metric of success, we narrowed down the datasets provided to the top 200 most grossing movies worldwide then calculated the RoI for each. After plotting several scatter and bar plots comparing runtime, production budget, gross revenue, release date, genre, directors,

Business Problem

Microsoft sees all the big companies creating original video content and they want to get in on the fun. They have decided to create a new movie studio, but they don't know anything about creating movies. You are charged with exploring what types of films are currently doing the best at the box office. You must then translate those findings into actionable

In [72]:
```
#importing pandas for data analysis.
import pandas as pd
#importing numpy for numerical analysis.
import numpy as np
#importing seaborn and matplotlib for data visualization.
import seaborn as sns
import matplotlib.pyplot as plt
#importing sqlite3 for basement management.
import sqlite3
```

THE IMDB DATA SET.

DATA UNDERSTANDING.

In [73]:
```
imdb = pd.read_sql("""SELECT * FROM movie_basics; """, conn)
imdb.head(10)
```

Out[73]:

	movie_id	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
5	tt0111414	A Thin Life	A Thin Life	2018	75.0	Comedy
6	tt0112502	Bigfoot	Bigfoot	2017	NaN	Horror,Thriller
7	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	Adventure,Animation,Comedy
8	tt0139613	O Silêncio	O Silêncio	2012	NaN	Documentary,History
9	tt0144449	Nema aviona za Zagreb	Nema aviona za Zagreb	2012	82.0	Biography

In [74]:
```
#Importing the data set and previewing.
imdb = pd.read_sql("""SELECT * FROM movie_ratings; """, conn)
imdb.head(10)
```

Out[74]:

	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
5	tt1069246	6.2	326
6	tt1094666	7.0	1613
7	tt1130982	6.4	571
8	tt1156528	7.2	265
9	tt1161457	4.2	148

In [75]:
```
imdb = pd.read_sql(""" SELECT *
FROM movie_basics
JOIN movie_ratings
USING(movie_id);
""",conn).head(10)
```

In [76]:
```
#getting data summary
imdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   movie_id         10 non-null     object 
 1   primary_title    10 non-null     object 
 2   original_title   10 non-null     object 
 3   start_year       10 non-null     int64  
 4   runtime_minutes  8 non-null      float64
 5   genres           10 non-null     object 
 6   averagerating    10 non-null     float64
 7   numvotes         10 non-null     int64  
dtypes: float64(2), int64(2), object(4)
memory usage: 768.0+ bytes
```

This is a pandas DataFrame object that has 73856 rows and 3 columns. There are no missing values in any of the columns, as indicated by the non-null count. The memory usage of this DataFrame is 1.7+ MB.

In [77]:
```
#getting statistical summary
imdb.describe()
```

Out[77]:

	start_year	runtime_minutes	averagerating	numvotes
count	10.000000	8.000000	10.000000	10.000000
mean	2015.200000	123.750000	6.490000	563.200000
std	3.392803	38.130415	1.260026	1395.785466
min	2010.000000	80.000000	4.100000	13.000000
25%	2013.000000	95.750000	6.200000	45.500000
50%	2017.000000	118.000000	6.850000	70.500000
75%	2017.750000	145.750000	7.150000	227.000000
max	2019.000000	180.000000	8.100000	4517.000000

The summary statistics provide information about the distribution of four columns - start_year, runtime_minutes, averagerating, and numvotes in a DataFrame.

For the start_year column, we can see that there are 10 entries and the earliest start year is 2010, while the latest is 2019. This tells us the range of years when the movies were released.

The runtime_minutes column has only 8 entries, indicating that two movies have missing values. The average runtime across all movies is 123.75 minutes, with a standard deviation of 38.13 minutes. The minimum and maximum values of the runtime column show the shortest and longest movies in the dataset.

For the averagerating column, we can see that there are 10 entries and the average rating across all movies is 6.49, with a standard deviation of 1.26. The minimum and maximum values of the column show the lowest and highest ratings in the dataset.

For the numvotes column, we can see that there are 10 entries, and the average number of votes across all movies is 563.2, with a standard deviation of 1395.79. The minimum and maximum values of the column show the least and most voted movies in the dataset.

DATA CLEANING

In [78]:
```
#checking for missing values
imdb.isnull().sum()
```

Out[78]:
```
movie_id           0
primary_title      0
original_title     0
start_year         0
runtime_minutes    2
genres             0
averagerating      0
numvotes           0
dtype: int64
```

The runtime_minutes column contains two null values. All the other columns do not have any null values.

In [79]:
```
#finding dublicates
imdb.duplicated().sum()
```

Out[79]:
```
0
```

This data does not contain any dupkicated values.

In [80]:
```
#replacing genres missing values with the mode since this is a categorical dat
imdb['genres'].mode()[0]
imdb['genres'].value_counts()
```

Out[80]:
```
Drama                          2
Action,Crime,Drama             1
Biography,Drama                1
Comedy,Drama                   1
Comedy,Drama,Fantasy           1
Horror,Thriller                1
Adventure,Animation,Comedy     1
History                        1
Documentary                    1
Name: genres, dtype: int64
```

In [81]:
```
#filling runtime_minutes null values with mode

imdb_mode = imdb['runtime_minutes'].mode()[0]
imdb['runtime_minutes'].fillna('imdb_mode', inplace = True)

imdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   movie_id         10 non-null     object 
 1   primary_title    10 non-null     object 
 2   original_title   10 non-null     object 
 3   start_year       10 non-null     int64  
 4   runtime_minutes  10 non-null     object 
 5   genres           10 non-null     object 
 6   averagerating    10 non-null     float64
 7   numvotes         10 non-null     int64  
dtypes: float64(1), int64(2), object(5)
memory usage: 768.0+ bytes
```

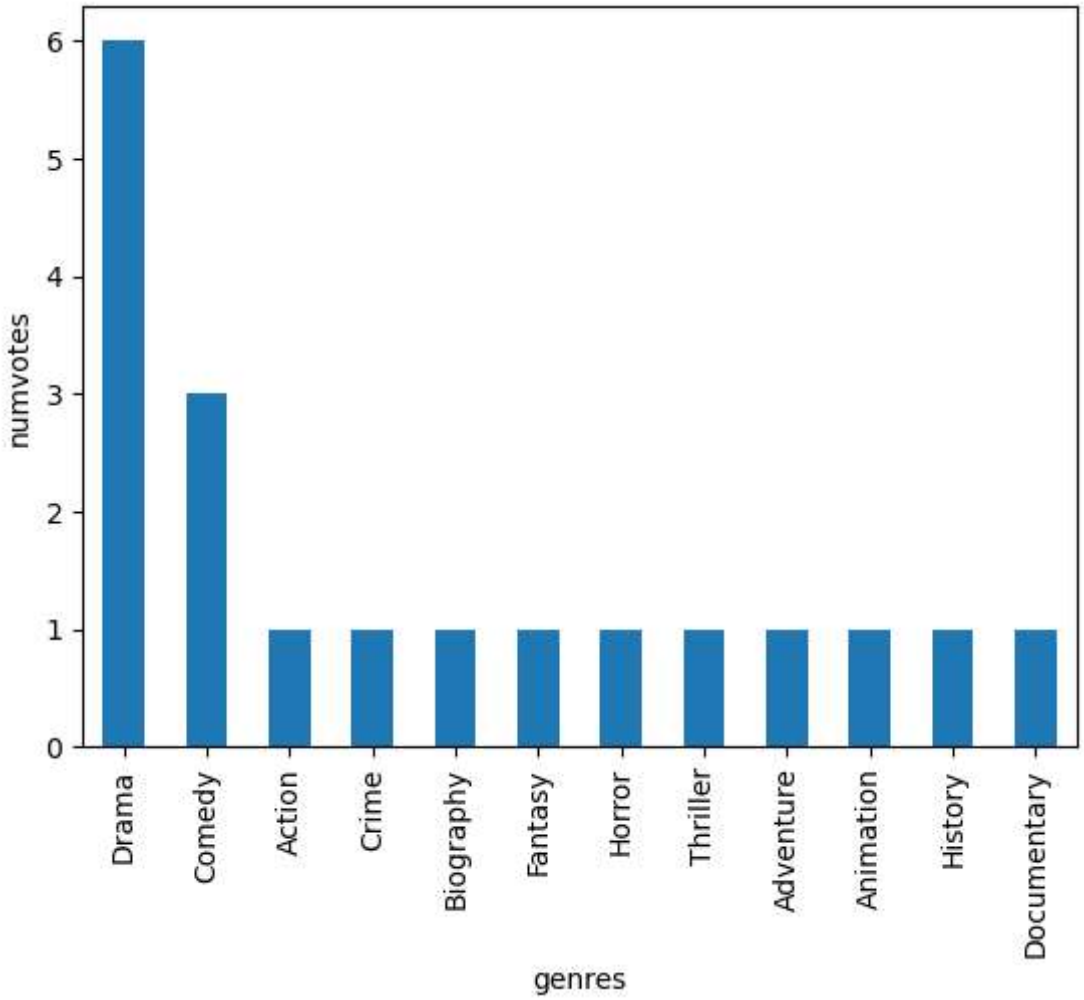
In [82]:
```
#cheking now if the data is clean

imdb.isnull().sum()
```

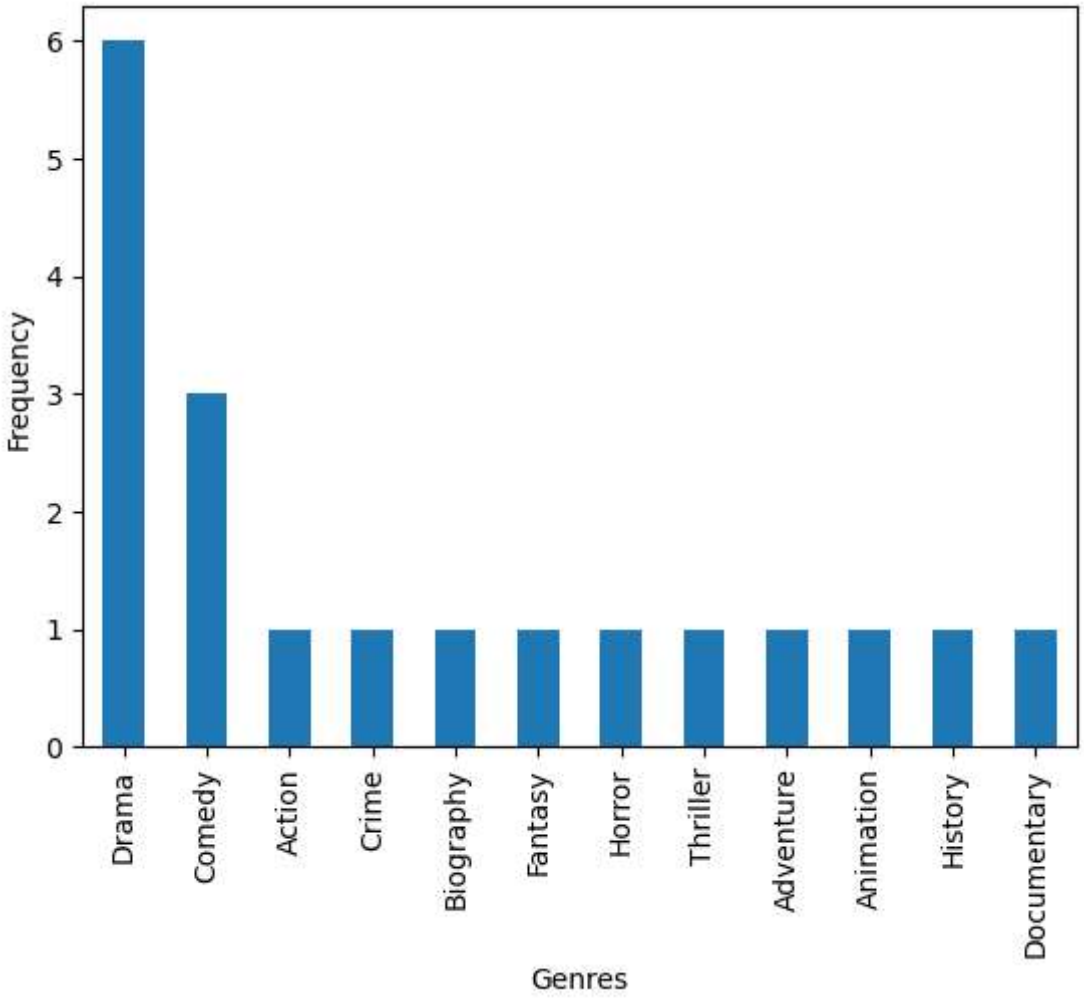
Out[82]:
```
movie_id           0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
genres             0
averagerating      0
numvotes           0
dtype: int64
```

In [89]:
```
#Creating a histogram showing the frequency of each genre in the genres column

genres = imdb.genres.str.split(',', expand=True).stack().reset_index(drop=True
genres.value_counts().plot(kind='bar')
plt.ylabel('numvotes')
plt.xlabel('genres')
plt.show()
```

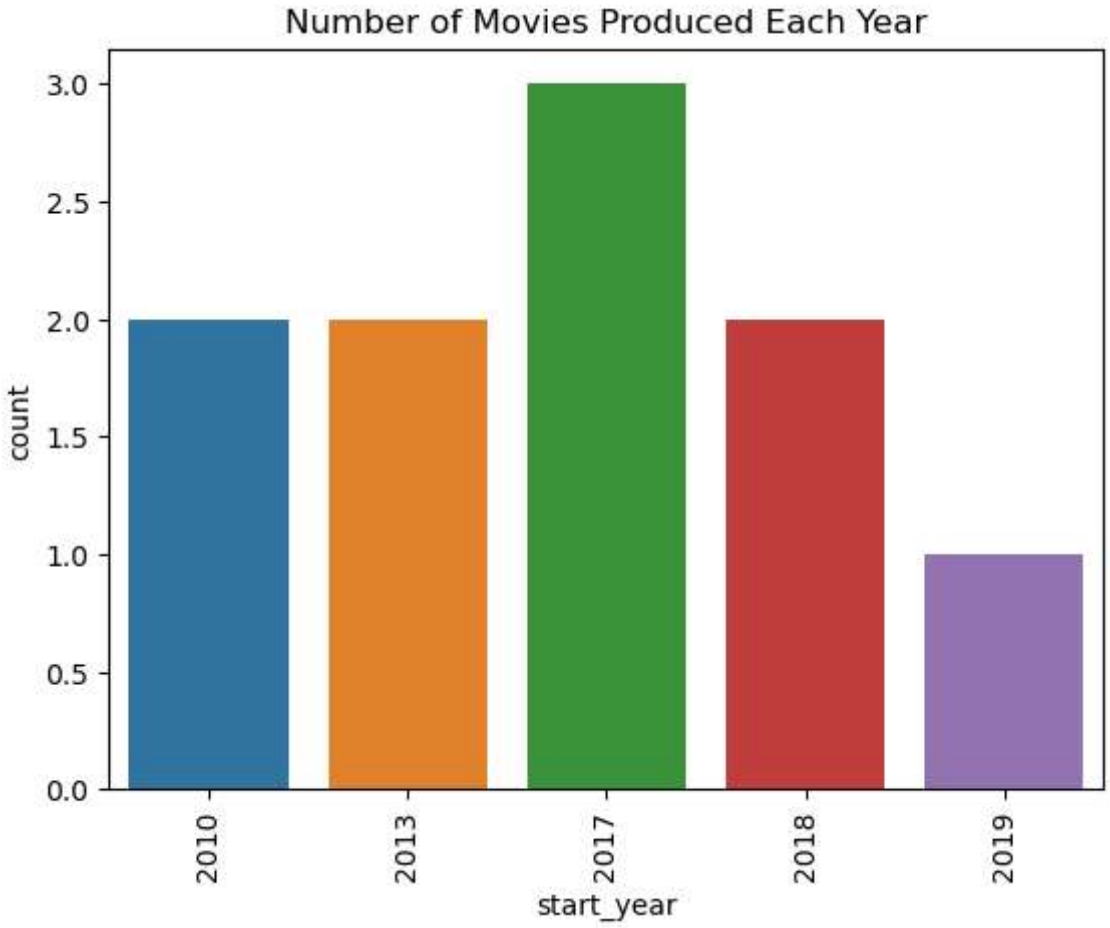


In [87]:
```
# Create a histogram showing the frequency of each genre in the genres column
genres = imdb.genres.str.split(',', expand=True).stack().reset_index(drop=True
genres.value_counts().plot(kind='bar')
plt.xlabel('Genres')
plt.ylabel('Frequency')
plt.show()
```



Drama movies had the highest frequency followed by comedy and action at third.

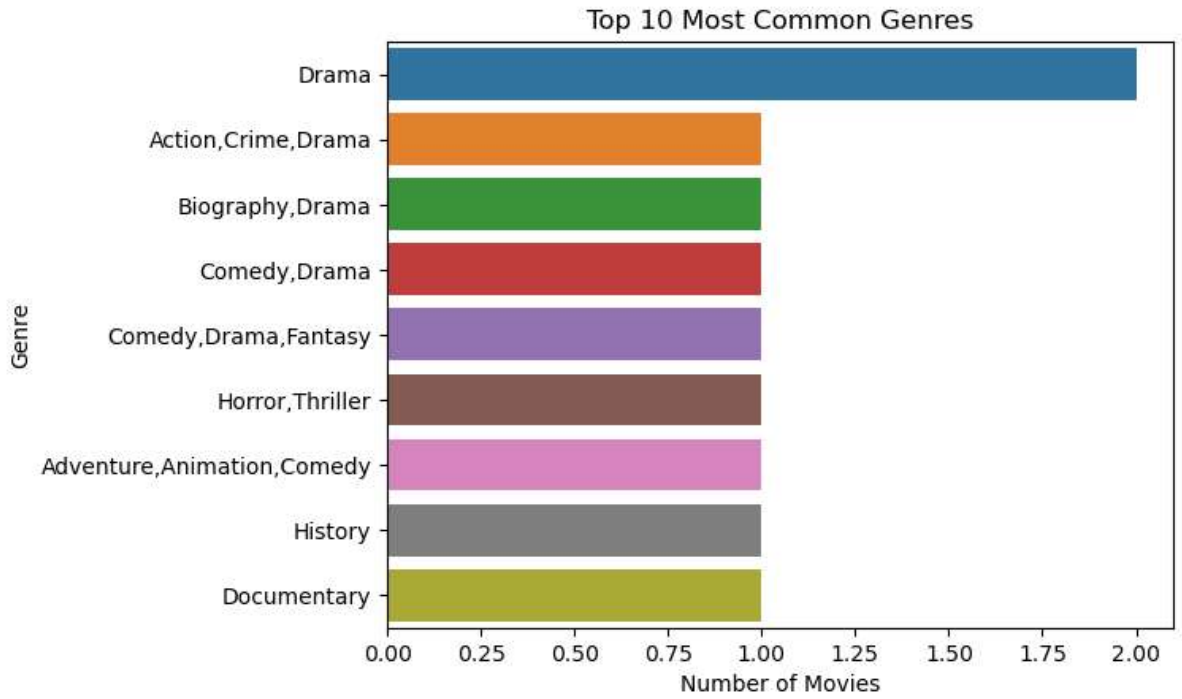
In [86]:
```
#Plotting number of movies produced per year.
sns.countplot(data=imdb, x='start_year')
plt.xticks(rotation=90)
plt.title('Number of Movies Produced Each Year')
plt.show()
```



2017 had the highest number of movie productions.

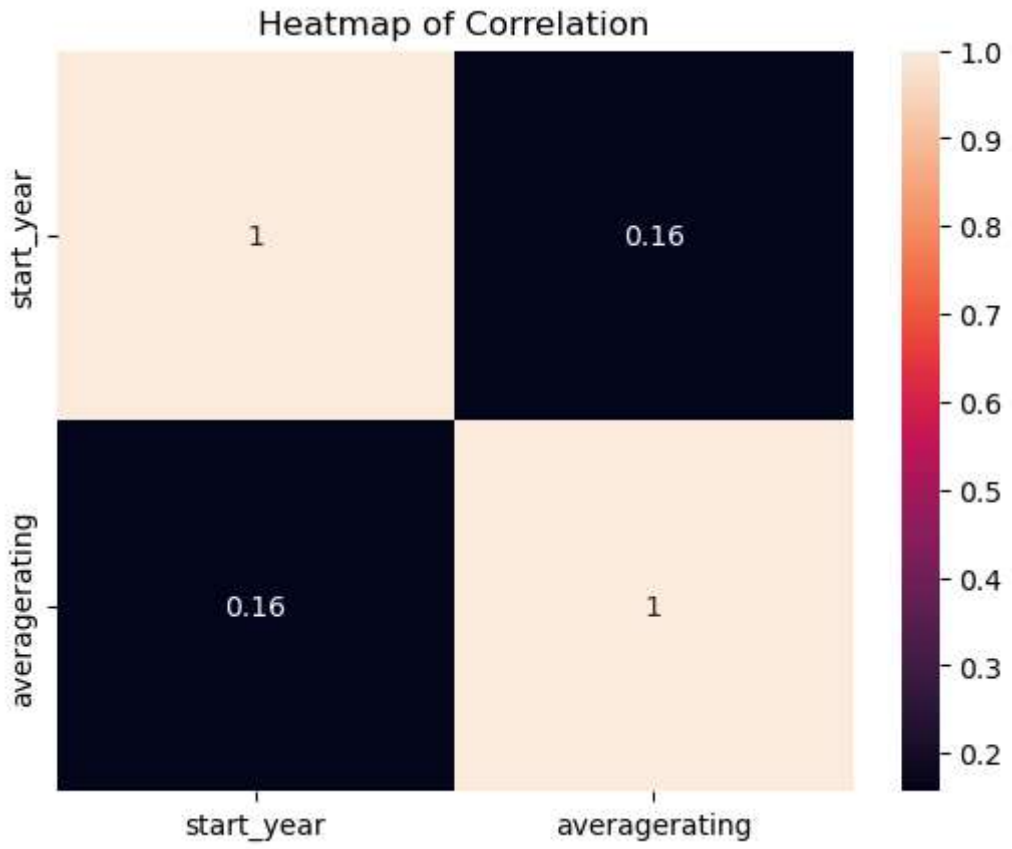
In [85]:
```
#Plotting top ten most common genres.
top_genres = imdb['genres'].value_counts().head(10)

sns.barplot(x=top_genres.values, y=top_genres.index)
plt.xlabel('Number of Movies')
plt.ylabel('Genre')
plt.title('Top 10 Most Common Genres')
plt.show()
```



The above graph ndicates that drama was th most common genre.

In [84]:
```
#Plotting a heat map of correlation
corr_matrix = imdb[['start_year', 'runtime_minutes', 'averagerating']].corr()
sns.heatmap(corr_matrix, annot=True)
plt.title('Heatmap of Correlation')
plt.show()
```



FINDINGS.

Drama got the highest number of voters followed by Documentary, Comedy, Thriller, Horror, Action, Romance, Crime, Adventure, Biography, Family, Mystery, History, Sci-Fi, Fantacy, Music, Animation, Sport, War, Musical, News, Western, Reality-TV, Adult, Game-Show and Short in that order.

RECOMMENDATIONS.

Highly recommend Microsoft to produce a lot of Drama genres, Documentary and Comedy since these three, in the order, recored highest number of voters.

In [ ]: