Data cleaning & Analysis in Pandas

1. Business understanding and Defining Business Question

```
a). Business Understanding:

Introduction:
The real-world problem this project aims to solve is Microsoft's entry into
the movie industry. Microsoft intends to establish a new movie studio but
lacks knowledge and insights into the types of films that perform well in
terms of box office revenue and audience reception. The objective is to
explore the movie data to provide actionable insights that can guide
Microsoft in deciding what type of films to create.

Specifying the Data Analytic Question

- To create insight on the types of films that perform well in terms of
numvotes
- To create insight on audience reception on different types of movies
- To provide any other actionable insights to guide Microsoft on which
films to create

b). Understanding the data

Data Sources: The dataset includes movie data from multiple sources, such
as Box Office Moj and IMDB. These sources provide comprehensive information
about movies, including box office performance, ratings, and other relevant
details. There are other data sources which we haven't used, but we'd like
to include on this list as well. There is data set from Rotten Tomatoes,
TheMovieDB, and The Numbers

Data Suitability:
•   Box Office Mojo provides box office revenue data, which is essential
for understanding a movie's financial success.
•   IMDB and Rotten Tomatoes offer information about user and critic
ratings, helping gauge audience and critical reception.
•   TheMovieDB provides additional movie details, including cast and crew
information.
•   The Numbers offer comprehensive data on budgets, production costs, and
revenue.

Conclusion: The project's implications for the real-world problem are
significant. It will enable Microsoft's new movie studio to make data-
driven decisions, potentially increasing the studio's chances of success.
Additionally, it can lead to the creation of movies that better cater to
audience preferences, enhancing the overall movie-watching experience.

c). Recording the Experimental Design

1. Business understanding and Defining Business Question
2. Reading the data
3. Checking the data
4. Data cleaning
```

```
5. Exploratory data analysis(Univeriate, Bivariate and Multivariate)
6. Conclusion
7. Recommendation
```

2.0 Reading the Data

In [ ]:
```
# Import libraries

import sqlite3
import pandas as pd
```

2.1 Loading the Dataset

In [ ]:
```
# Import dataset 1
# Connect to the SQLite database

conn = sqlite3.connect('im.db')

# Create a cursor object to interact with the database
cursor = conn.cursor()
```

In [3]:
```
cursor.execute("""SELECT * FROM movie_basics;""").fetchall()
```

```
  'Comedy,Drama'),
 ('tt0331314',
  'Bunyan and Babe',
  'Bunyan and Babe',
  2017,
  84.0,
  'Adventure,Animation,Comedy'),
 ('tt0337692',
  'On the Road',
  'On the Road',
  2012,
  124.0,
  'Adventure,Drama,Romance'),
 ('tt0337882', 'Blind Sided', 'Blind Sided', 2010, None, 'Comedy,Crime,Dr
ama'),
 ('tt0337926',
  'Chatô - The King of Brazil',
  'Chatô: O Rei do Brasil',
  2015,
```

In [ ]:
```
# Import movie_basics table using pandas
import pandas as pd

pd.DataFrame(
    data=cursor.execute("""SELECT * FROM movie_basics;""").fetchall(),
    columns=[x[0] for x in cursor.description]
)
```

In [5]:
```
# Import movie_ratings table using pandas

pd.DataFrame(
    data=cursor.execute("""SELECT * FROM movie_ratings;""").fetchall(),
    columns=[x[0] for x in cursor.description]
)
```

Out[5]:

	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 × 3 columns

In [ ]:
```
# Now that we have the connection, we can use the pd.read_sql method instead o
df = pd.read_sql("""SELECT * FROM movie_basics;""", conn)
df
```

In [7]:
```
# Import directors table
pd.DataFrame(
    data=cursor.execute("""SELECT * FROM directors;""").fetchall(),
    columns=[x[0] for x in cursor.description]
)
```

Out[7]:

	movie_id	person_id
0	tt0285252	nm0899854
1	tt0462036	nm1940585
2	tt0835418	nm0151540
3	tt0835418	nm0151540
4	tt0878654	nm0089502
...	...	...
291169	tt8999974	nm10122357
291170	tt9001390	nm6711477
291171	tt9001494	nm10123242
291172	tt9001494	nm10123248
291173	tt9004986	nm4993825

291174 rows × 2 columns

In [38]:
```
# This is the writers table. It's important because we'll dig about writers ar
pd.DataFrame(
    data=cursor.execute("""SELECT * FROM writers;""").fetchall(),
    columns=[x[0] for x in cursor.description]
)
```

Out[38]:

	movie_id	person_id
0	tt0285252	nm0899854
1	tt0438973	nm0175726
2	tt0438973	nm1802864
3	tt0462036	nm1940585
4	tt0835418	nm0310087
...	...	...
255868	tt8999892	nm10122246
255869	tt8999974	nm10122357
255870	tt9001390	nm6711477
255871	tt9004986	nm4993825
255872	tt9010172	nm8352242

255873 rows × 2 columns

In [9]:
```
# This is the persons table. It's important because we'll dig about actors and
pd.DataFrame(
    data=cursor.execute("""SELECT * FROM persons;""").fetchall(),
    columns=[x[0] for x in cursor.description]
)
```

Out[9]:

	person_id	primary_name	birth_year	death_year	primary_
0	nm0061671	Mary Ellen Bauder	NaN	NaN	miscellaneous,production_manag
1	nm0061865	Joseph Bauer	NaN	NaN	composer,music_department,sound_
2	nm0062070	Bruce Baum	NaN	NaN	miscellaneous
3	nm0062195	Axel Baumann	NaN	NaN	camera_department,cinematographer,art_
4	nm0062798	Pete Baxter	NaN	NaN	production_designer,art_department,se
...	...	...	...	...	
606643	nm9990381	Susan Grobes	NaN	NaN	
606644	nm9990690	Joo Yeon So	NaN	NaN	
606645	nm9991320	Madeline Smith	NaN	NaN	
606646	nm9991786	Michelle Modigliani	NaN	NaN	
606647	nm9993380	Pegasus Envoyé	NaN	NaN	director

606648 rows × 5 columns

In [39]:
```
# We're going to join all the data from the chosen tables(movie_basics, movie_
#These are the columns we need; mb.movie_id, mb.primary_title, mb.genres, mb.s

q = """
SELECT mb.movie_id, mb.primary_title, mb.genres, mb.start_year, mb.runtime_mi
FROM movie_basics mb
JOIN movie_ratings mr
    ON mb.movie_id = mr.movie_id
JOIN directors d
    ON mb.movie_id = d.movie_id
JOIN persons p
    ON d.person_id = p.person_id
;
"""
df = pd.read_sql(q, conn)
```

01/04/2023, 08:02pm

In [11]:
```
# Show the first five rows to better understand the data

df.head()
```

Out[11]:

	movie_id	primary_title	genres	start_year	runtime_minutes	averagerating	numv
0	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7.0	
1	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7.0	
2	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7.0	
3	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7.0	
4	tt0066787	One Day Before the Rainy Season	Biography,Drama	2019	114.0	7.2	

3.0 Previewing the Dataset

In [40]:
```
#checking the bottom of the dataset

df.tail()
```

Out[40]:

	movie_id	primary_title	genres	start_year	runtime_minutes	averagerating	numv
181382	tt9914642	Albatross	Documentary	2017	NaN	8.5	
181383	tt9914642	Albatross	Documentary	2017	NaN	8.5	
181384	tt9914942	La vida sense la Sara Amat	None	2019	NaN	6.6	
181385	tt9914942	La vida sense la Sara Amat	None	2019	NaN	6.6	
181386	tt9916160	Drømmeland	Documentary	2019	72.0	6.5	

In [13]:
```
#Show the data types

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181387 entries, 0 to 181386
Data columns (total 10 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   movie_id            181387 non-null  object 
 1   primary_title       181387 non-null  object 
 2   genres              180047 non-null  object 
 3   start_year          181387 non-null  int64  
 4   runtime_minutes     163584 non-null  float64
 5   averagerating       181387 non-null  float64
 6   numvotes            181387 non-null  int64  
 7   primary_name        181387 non-null  object 
 8   primary_profession  181262 non-null  object 
 9   person_id           181387 non-null  object 
dtypes: float64(2), int64(2), object(6)
memory usage: 13.8+ MB
```

In [41]:
```
# Checking the number of entries in each column

df.count()
```

Out[41]:
```
movie_id              181387
primary_title         181387
genres                180047
start_year            181387
runtime_minutes       163584
averagerating         181387
numvotes              181387
primary_name          181387
primary_profession    181262
person_id             181387
dtype: int64
```

In [15]:
```
# Generate summary statistics for numeric columns
df.describe()
```

Out[15]:

	start_year	runtime_minutes	averagerating	numvotes
count	181387.000000	163584.000000	181387.000000	1.813870e+05
mean	2014.309802	97.789484	6.217683	4.955524e+03
std	2.536111	194.434689	1.388026	3.760931e+04
min	2010.000000	3.000000	1.000000	5.000000e+00
25%	2012.000000	84.000000	5.400000	1.900000e+01
50%	2014.000000	94.000000	6.300000	6.600000e+01
75%	2016.000000	107.000000	7.200000	3.110000e+02
max	2019.000000	51420.000000	10.000000	1.841066e+06

4.0 Cleaning the Dataset

4.1 Checking Nulls

In [42]:
```
#Check the dataset for nulls

df.isnull().values.any()
```

Out[42]:
```
True
```

In [17]:
```
# Let's get a summary of the missing data

df.isnull().sum()
```

Out[17]:
```
movie_id                  0
primary_title             0
genres                 1340
start_year                0
runtime_minutes       17803
averagerating             0
numvotes                  0
primary_name              0
primary_profession      125
person_id                 0
dtype: int64
```

In [43]:
```
# Calculating the total number of null values

# Step 1: Calculate the total number of null values

null_count = df.isnull().sum()
null_count
```

Out[43]:
```
movie_id                 0
primary_title            0
genres                1340
start_year               0
runtime_minutes      17803
averagerating            0
numvotes                 0
primary_name             0
primary_profession     125
person_id                0
dtype: int64
```

In [44]:
```
# Step 2: Calculate the total number of values (non-null)
total_count = df.shape[0]
total_count
```

Out[44]:
```
181387
```

In [20]:
```
# Step 3: Calculate the percentage od nulls for each column

null_percentage = (null_count / total_count) * 100

print(null_percentage)
```

```
movie_id              0.000000
primary_title         0.000000
genres                0.738752
start_year            0.000000
runtime_minutes       9.814926
averagerating         0.000000
numvotes              0.000000
primary_name          0.000000
primary_profession    0.068913
person_id             0.000000
dtype: float64
```

The above shows that there is a huge percantage of null values especially in the runtime_minutes column. The other columns have almost negligable nulls so we can do away with those.

In [45]:
```
# Remove null vales from studioand domestic_gross
df.dropna(subset = ['primary_profession'], inplace=True)
df.dropna(subset = ['genres'], inplace=True)
```

In [46]:
```
# Find the mode to relace the null values

runtime_minutes_mode = df['runtime_minutes'].mode()
runtime_minutes_mode
```

Out[46]:
```
0    90.0
Name: runtime_minutes, dtype: float64
```

In [23]:
```
# Let's get a single mode value (the most frequent value)
# by using .iloc[0] to access the first mode in the Series
mode_value = runtime_minutes_mode.iloc[0]
mode_value
```

Out[23]:
```
90.0
```

In [47]:
```
#Replace the null values with mode

df['runtime_minutes'].fillna(mode_value, inplace=True)
df['runtime_minutes']
```

Out[47]:
```
0         175.0
1         175.0
2         175.0
3         175.0
4         114.0
          ...
181380     98.0
181381     98.0
181382     90.0
181383     90.0
181386     72.0
Name: runtime_minutes, Length: 179922, dtype: float64
```

In [48]:
```
# Recheck null values to confirm they're non existant

null_count = df.isnull().sum()
null_count
```

Out[48]:
```
movie_id              0
primary_title         0
genres                0
start_year            0
runtime_minutes       0
averagerating         0
numvotes              0
primary_name          0
primary_profession    0
person_id             0
dtype: int64
```

In [26]:
```
# All the nulls are now gone.
```

4.2 Checking Duplicates

In [49]:
```
#Check for duplicates

df.duplicated().value_counts
```

Out[49]:
```
<bound method IndexOpsMixin.value_counts of 0         False
1          True
2          True
3          True
4         False
          ...
181380    False
181381     True
181382    False
181383     True
181386    False
Length: 179922, dtype: bool>
```

In [28]:
```
# Let's drop the duplicate rows
#We're going to start by checking out the duplicate rows to see what they look

#Identify duplicates
duplicates = df.duplicated(keep=False)  # keep=False marks all duplicates as

# Filter and display the duplicate rows
duplicate_rows = df[duplicates]
duplicate_rows.head(20)
```

Out[28]:

	movie_id	primary_title	genres	start_year	runtime_minutes	averagerati
0	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7
1	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7
2	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7
3	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7
5	tt0069049	The Other Side of the Wind	Drama	2018	122.0	6
6	tt0069049	The Other Side of the Wind	Drama	2018	122.0	6

In [50]:
```
# Remove all duplicates based on the 'movie_id' and 'primary_title' columns

df_no_duplicates = df.drop_duplicates(subset=['movie_id', 'primary_title'], k
df_no_duplicates
```

Out[50]:

	movie_id	primary_title	genres	start_year	runtime_minutes	average
0	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	
4	tt0066787	One Day Before the Rainy Season	Biography,Drama	2019	114.0	
5	tt0069049	The Other Side of the Wind	Drama	2018	122.0	
7	tt0069204	Sabse Bada Sukh	Comedy,Drama	2018	90.0	
8	tt0100275	The Wandering Soap Opera	Comedy,Drama,Fantasy	2017	80.0	

4.3 Checking for data inconsistencies

In [51]:
```
df.describe()
```

Out[51]:

	start_year	runtime_minutes	averagerating	numvotes
count	179922.000000	179922.000000	179922.000000	1.799220e+05
mean	2014.310546	97.096231	6.213478	4.995573e+03
std	2.532863	185.397821	1.386565	3.775948e+04
min	2010.000000	3.000000	1.000000	5.000000e+00
25%	2012.000000	85.000000	5.400000	2.000000e+01
50%	2014.000000	91.000000	6.300000	6.600000e+01
75%	2016.000000	105.000000	7.200000	3.160000e+02
max	2019.000000	51420.000000	10.000000	1.841066e+06

'domestic_gross' Column:

The minimum value is 100 (min: 1.000000e+02), which suggests there are some very low domestic gross values. This is not really an inconsistency.

The 'year' column's minimum value is 2010 (min: 2010.000000), and the maximum value is 2018 (max: 2018.000000). These values fall within a reasonable range for movie release years.

4.4 Checking the data types

In [52]:
```
#df.info() is useful for understanding the structure of your DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 179922 entries, 0 to 181386
Data columns (total 10 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   movie_id            179922 non-null  object 
 1   primary_title       179922 non-null  object 
 2   genres              179922 non-null  object 
 3   start_year          179922 non-null  int64  
 4   runtime_minutes     179922 non-null  float64
 5   averagerating       179922 non-null  float64
 6   numvotes            179922 non-null  int64  
 7   primary_name        179922 non-null  object 
 8   primary_profession  179922 non-null  object 
 9   person_id           179922 non-null  object 
dtypes: float64(2), int64(2), object(6)
memory usage: 15.1+ MB
```

5.0 Exploratory Data Analysis(EDA)

Univariate Analysis

Univariate analysis is a statistical method used to analyze and summarize data involving a single variable. In univariate analysis, you focus on understanding the characteristics, patterns, and distributions of one variable at a time, without considering the relationships or dependencies between variables. It helps you gain insights into the distribution, central tendency, and variability of a single variable.

In [32]:
```
df.head(10)
```

Out[32]:

	movie_id	primary_title	genres	start_year	runtime_minutes	averagerating	nu
0	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7.0	
1	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7.0	
2	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7.0	
3	tt0063540	Sunghursh	Action,Crime,Drama	2013	175.0	7.0	
4	tt0066787	One Day Before the Rainy Season	Biography,Drama	2019	114.0	7.2	
5	tt0069049	The Other Side of the Wind	Drama	2018	122.0	6.9	
6	tt0069049	The Other Side of the Wind	Drama	2018	122.0	6.9	
7	tt0069204	Sabse Bada Sukh	Comedy,Drama	2018	90.0	6.1	
8	tt0100275	The Wandering Soap Opera	Comedy,Drama,Fantasy	2017	80.0	6.5	
9	tt0100275	The Wandering Soap Opera	Comedy,Drama,Fantasy	2017	80.0	6.5	

Objectives 5.1 To identify the most frequently occurring movie ID 5.2 To create insight on the types of films that perform well in terms of numvotes 5.3 To create insight on audience reception on different types of movies 5.4 To create insight on which directors have highest numvotes and average ratings 5.5 To provide any other actionable insights to guide Microsoft on which films to create

In [53]:
```
# Let's start by calculating the number of unique movie IDs to understand the

unique_movie_ids = df['movie_id'].nunique()
unique_movie_ids
```

Out[53]: 72382

5.1 Let's identify the types of films that perform well in terms of numvotes, let's start by analyzing the genres and averagerating columns in the dataset.

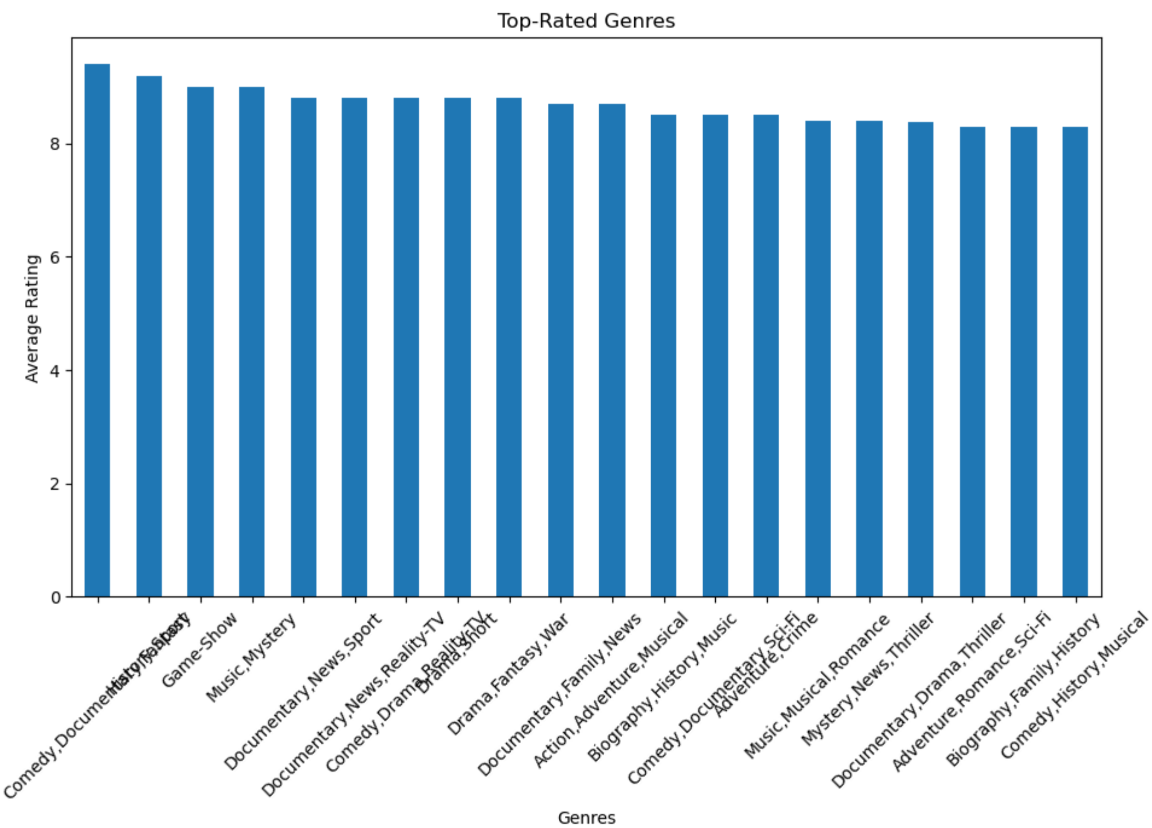
In [54]:
```
genre_ratings = df.groupby('genres')['numvotes'].mean().sort_values(ascending
genre_ratings.head(10)
```

Out[54]:
```
genres
Action,Adventure,Sci-Fi      281551.713615
Adventure,Mystery,Sci-Fi     215778.000000
Action,Adventure,Fantasy     143266.062315
Adventure,Fantasy            120143.217391
Drama,History,Musical        110238.230769
Adventure,Drama,Western      106879.000000
Action,Adventure,Thriller    105231.680000
Family,Fantasy,Musical        99321.083333
Action,Crime,Sci-Fi           94424.000000
Action,Adventure,Mystery      92483.468750
Name: numvotes, dtype: float64
```

In [55]:
```
# Identify the genres with the highest average ratings. These are the genres

top_rated_genres = genre_ratings.head(10)
top_rated_genres
```

Out[55]:
```
genres
Action,Adventure,Sci-Fi      281551.713615
Adventure,Mystery,Sci-Fi     215778.000000
Action,Adventure,Fantasy     143266.062315
Adventure,Fantasy            120143.217391
Drama,History,Musical        110238.230769
Adventure,Drama,Western      106879.000000
Action,Adventure,Thriller    105231.680000
Family,Fantasy,Musical        99321.083333
Action,Crime,Sci-Fi           94424.000000
Action,Adventure,Mystery      92483.468750
Name: numvotes, dtype: float64
```

In [36]:
```python
# Visualize the data

import pandas as pd
import matplotlib.pyplot as plt

# Calculate the average rating for each genre

genre_ratings = df.groupby('genres')['averagerating'].mean().sort_values(asce

# Select the top-rated genres

top_rated_genres = genre_ratings.head(20)

# Create a bar chart to visualize the top-rated genres
plt.figure(figsize=(11, 6))
top_rated_genres.plot(kind='bar')
plt.title('Top-Rated Genres')
plt.xlabel('Genres')
plt.ylabel('Average Rating')
plt.xticks(rotation=45)
plt.show()
```



5.2 Create insight on audience reception on different types of movies

Creating insights into audience reception for different types of movies involves analyzing and visualizing data to understand how different genres or movie attributes relate to audience ratings. Here's a step-by-step approach to gaining insights:

Group by Genres:

Group your dataset by the genres column.

In [56]:
```
genre_group = df.groupby('genres')
genre_group
```

Out[56]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002A32DC4D150>
```

Calculate Summary Statistics:

Calculate key summary statistics for each genre, such as the mean, median, and count of ratings. These statistics will provide an overview of how each genre is rated.

In [57]:
```
genre_stats = genre_group['averagerating'].agg(['mean', 'median', 'count']).s
genre_stats.head(20)
```

Out[57]:

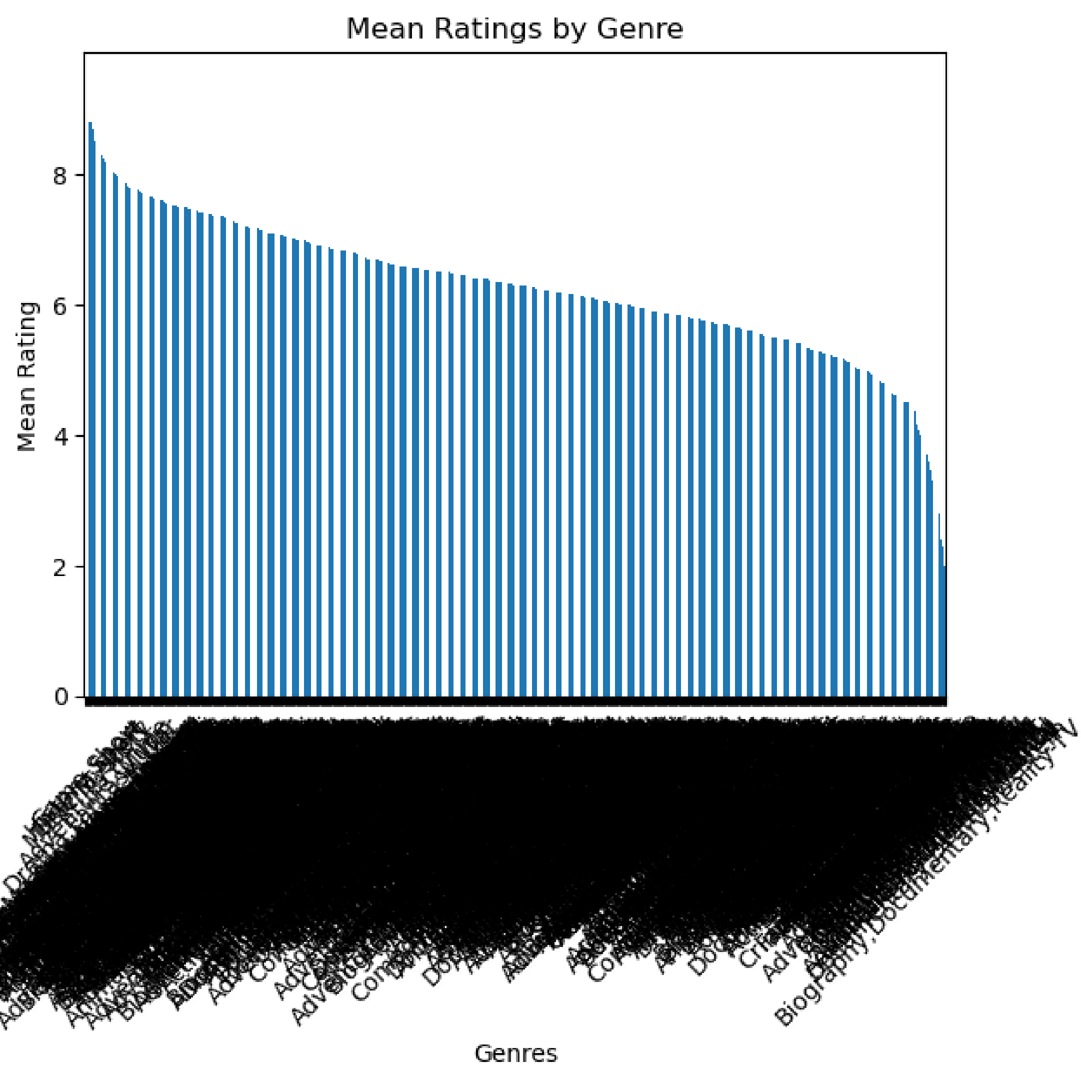
genres	mean	median	count
Comedy,Documentary,Fantasy	9.400000	9.4	1
History,Sport	9.200000	9.2	1
Game-Show	9.000000	9.0	6
Music,Mystery	9.000000	9.0	3
Documentary,News,Sport	8.800000	8.8	2
Documentary,News,Reality-TV	8.800000	8.8	3
Comedy,Drama,Reality-TV	8.800000	8.8	2
Drama,Short	8.800000	8.8	1
Drama,Fantasy,War	8.800000	8.8	2
Documentary,Family,News	8.700000	9.0	106
Action,Adventure,Musical	8.700000	8.7	1
Biography,History,Music	8.500000	8.5	1
Comedy,Documentary,Sci-Fi	8.500000	8.5	1
Adventure,Crime	8.500000	8.5	2
Music,Musical,Romance	8.400000	8.4	2
Mystery,News,Thriller	8.400000	8.4	2
Documentary,Drama,Thriller	8.377778	8.4	18
Adventure,Romance,Sci-Fi	8.300000	8.3	1
Biography,Family,History	8.300000	8.3	1
Comedy,History,Musical	8.300000	8.3	8

Visualize Genre Ratings:

Create visualizations to better understand how genres perform in terms of audience ratings. Here are some visualization ideas: a. Bar Chart for Mean Ratings:

In [58]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
genre_stats[['mean']].plot(kind='bar', legend=False)
plt.title('Mean Ratings by Genre')
plt.xlabel('Genres')
plt.ylabel('Mean Rating')
plt.xticks(rotation=45)
plt.show()
```

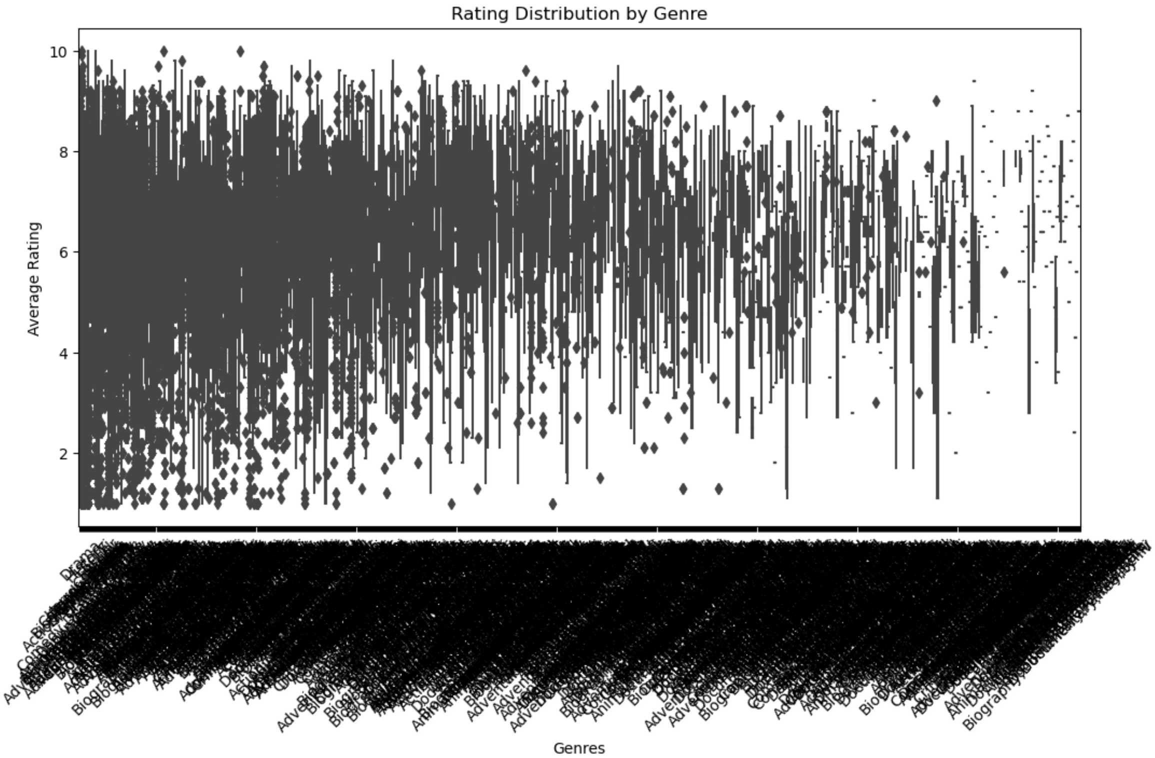
```
<Figure size 1200x600 with 0 Axes>
```



b. Box Plot for Rating Distribution:

Create box plots to visualize the distribution of ratings within each genre. This can help you understand the spread of ratings and identify outliers. python

In [71]:
```
import seaborn as sns

plt.figure(figsize=(12, 6))
sns.boxplot(x='genres', y='averagerating', data=df)
plt.title('Rating Distribution by Genre')
plt.xlabel('Genres')
plt.ylabel('Average Rating')
plt.xticks(rotation=45)
plt.show()
```



In [ ]:
```
5.3 To create insight on which directors have highest numvotes and average ra
```

In [64]:
```
# Group by Director:

# Group your dataset by the primary_name column, which represents the director

director_group = df.groupby('primary_name')
director_group
```

Out[64]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002A32D868A50>

Calculate Summary Statistics:

In [65]:
```
# Calculate key summary statistics for each director, such as the sum of numvo
# These statistics will provide an overview of how each director's movies are
# and how many votes their movies have received.

director_stats = director_group[['numvotes', 'averagerating']].agg({'numvotes
director_stats.head(10)
```

Out[65]:

	primary_name	numvotes	averagerating
0	A Normale Jef	1426	7.2000
1	A'Ali de Sousa	55	4.2000
2	A. Blaine Miller	8	7.0000
3	A. Cengiz Mert	6	3.2000
4	A. Fishman	37	7.8000
5	A. Haluk Unal	5	8.8000
6	A. Jagadesh	291	3.5000
7	A. Joji	6	5.5000
8	A. Karunakaran	4676	5.8125
9	A. Lawrence Dreyfuss	17	7.0000

So these show how each director's movies are rated and how many votes their movies have received.

In [66]:
```
# Sort the resulting DataFrame by the sum of numvotes in descending order
# to find the directors with the highest number of votes.

top_directors_by_votes = director_stats.sort_values(by='numvotes', ascending=
top_directors_by_votes.head(30)
```

Out[66]:

	primary_name	numvotes	averagerating
22159	James Gunn	18640459	6.266667
24831	Joe Russo	18421688	8.180645
4155	Anthony Russo	18421593	8.246667
55757	Zack Snyder	10576977	6.619231
9770	Christopher Nolan	10457390	8.437500
34880	Matthew Vaughn	9962120	7.500000
41392	Peter Jackson	8634677	7.743750
9758	Christopher Miller	6565719	7.421053
41643	Phil Lord	6565719	7.421053
28706	Kenneth Branagh	6454844	6.850000
7574	Bryan Singer	6423171	7.247059
44033	Ridley Scott	6411206	6.595455
47845	Shane Black	6402578	6.714286
43710	Rich Moore	6164592	7.595652
22223	James Mangold	6014842	7.716667
1204	Alan Taylor	5807424	6.746154
26667	Joss Whedon	5726110	7.455556
41584	Peyton Reed	5460683	7.193333
25655	Jon Favreau	5314585	6.606667
21544	J.J. Abrams	5241835	7.755556
52109	Todd Phillips	5076988	6.476471
32729	Marc Webb	5049491	6.807692
17612	Gareth Edwards	4932101	6.938462
34027	Martin Scorsese	4921033	7.642857
50685	Taika Waititi	4878821	7.846667
1398	Alejandro G. Iñárritu	4701810	7.727273
25746	Jon Watts	4365586	6.740000
22352	James Wan	4359407	7.158824
12798	Dean DeBlois	4345742	7.940000
41250	Pete Docter	4289467	7.977778

In [ ]:
```
Sort by Average Rating:
```

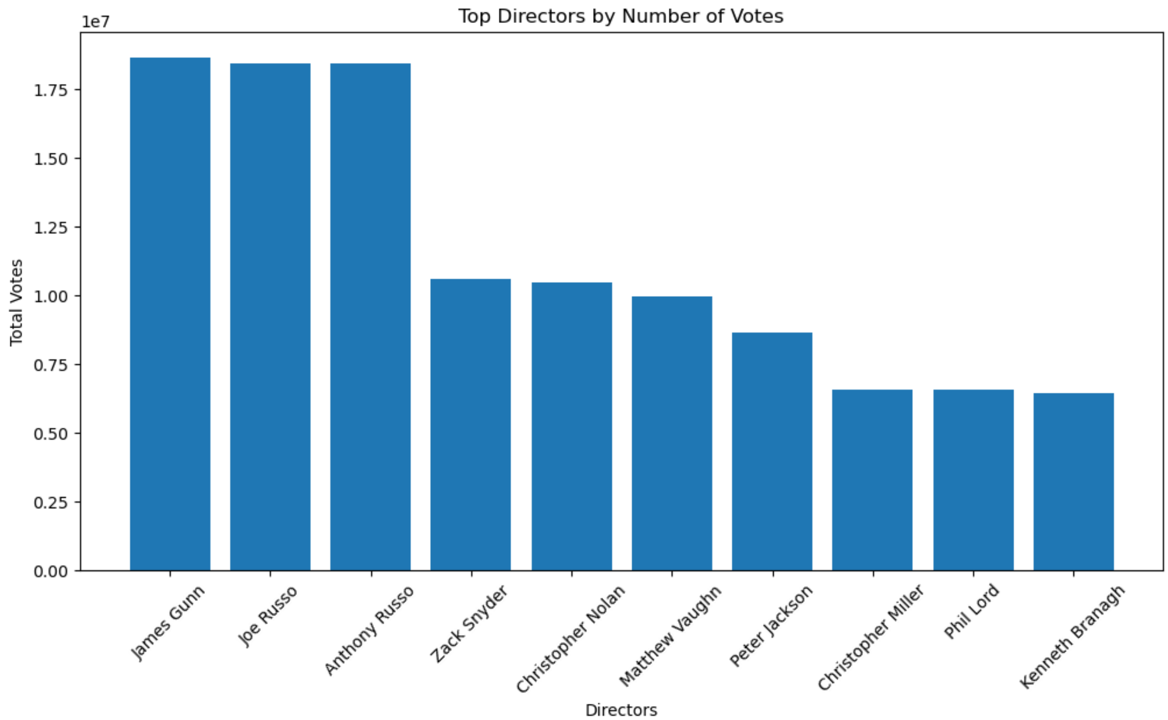
In [67]:
```
# Sort the DataFrame by the mean of averagerating in descending order to find

top_directors_by_rating = director_stats.sort_values(by='averagerating', ascen
top_directors_by_rating.head(30)
```

Out[67]:

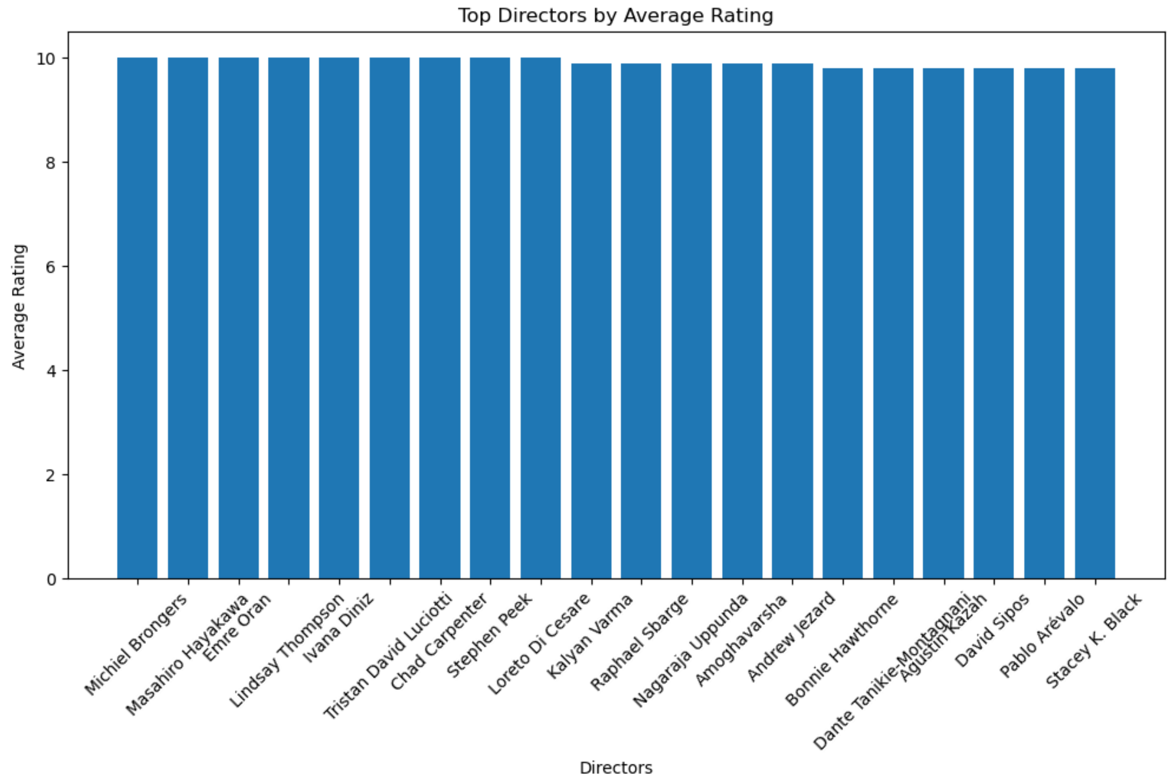
	primary_name	numvotes	averagerating
36390	Michiel Brongers	5	10.0
34224	Masahiro Hayakawa	5	10.0
15248	Emre Oran	6	10.0
30986	Lindsay Thompson	7	10.0
21383	Ivana Diniz	10	10.0
52842	Tristan David Luciotti	6	10.0
8503	Chad Carpenter	5	10.0
49545	Stephen Peek	20	10.0
31282	Loreto Di Cesare	16	10.0
27828	Kalyan Varma	10	9.9
43080	Raphael Sbarge	8	9.9
37785	Nagaraja Uppunda	417	9.9
2627	Amoghavarsha	10	9.9
3184	Andrew Jezard	8	9.9
6678	Bonnie Hawthorne	6	9.8
11717	Dante Tanikie-Montagnani	5	9.8
850	Agustín Kazah	28	9.8
12612	David Sipos	5	9.8
39994	Pablo Arévalo	28	9.8
49150	Stacey K. Black	5	9.8
52086	Todd Howe	45	9.8
23038	Javi Larrauri	5	9.8
33093	Maria Bagnat	5	9.8
49770	Steve Wystrach	96	9.7
54610	Will Watson	60	9.7
52026	Tobias Frindt	7	9.7
21557	J.M. Berrios	6	9.7
52970	Tyler Chandler	216	9.7
51590	Thomas Veit	25	9.7
41033	Pavlina Ivanova	32	9.7

In [ ]:
```
Visualize the Data:
```

In [68]:
```
Create visualizations, such as bar charts, to display the top directors with t

port matplotlib.pyplot as plt

Bar chart for top directors by number of votes
t.figure(figsize=(12, 6))
t.bar(top_directors_by_votes['primary_name'].head(10), top_directors_by_votes[
t.title('Top Directors by Number of Votes')
t.xlabel('Directors')
t.ylabel('Total Votes')
t.xticks(rotation=45)
t.show()
```



In [69]:
```
# Bar chart for top directors by average rating

plt.figure(figsize=(12, 6))
plt.bar(top_directors_by_rating['primary_name'].head(20), top_directors_by_ra
plt.title('Top Directors by Average Rating')
plt.xlabel('Directors')
plt.ylabel('Average Rating')
plt.xticks(rotation=45)
plt.show()
```



These steps will help identify the directors who have received the highest number of votes and have the highest average ratings for their movies.

Summart results.

5.2 Based on insight on audience reception on different types of movies

Here's a summary of the key statistics for the top 10 genres based on mean ratings:

1. Comedy, Documentary, Fantasy: This genre received the highest mean rating of 9.4, indicating exceptionally positive audience feedback. However, please note that there's only one rating available for this genre.
2. History, Sport: With a mean rating of 9.2, this genre also received very high ratings. It is another genre with a limited number of ratings
3. Game-Show: Game-Show genre has a mean rating of 9.0, which suggests strong audience appreciation. There are a total of 6 ratings for this genre.
4. Music, Mystery: This genre also has a mean rating of 9.0, indicating high viewer satisfaction, with 3 ratings recorded.

5. Documentary, News, Sport: With an 8.8 mean rating, this genre is well-received, but it has a relatively small sample size of 2 ratings.
6. Documentary, News, Reality-TV: Similar to the previous genre, it also has an 8.8 mean rating, based on 3 ratings.
7. Comedy, Drama, Reality-TV: This genre achieved an 8.8 mean rating and, like the previous two, has a limited number of ratings
8. Drama, Short: With an 8.8 mean rating, this genre also received positive feedback. However, there's only 1 rating for this genre.
9. Drama, Fantasy, War: This genre received an 8.8 mean rating, based on 2 ratings.
10. Documentary, Family, News: This genre has a mean rating of 8.7, indicating strong viewer satisfaction. Notably, it has a larger sample size of 106 ratings.

5.3 Here's a summary of the key statistics for the top 10 directors based on the sum of numvotes and the mean of averagerating:

1.A Normale Jef: Movies directed by A Normale Jef have received a total of 1426 votes, with an average rating of 7.2.

2. A'Ali de Sousa: This director's movies have received 55 votes on average, with a mean rating of 4.2.
3. A. Blaine Miller: A. Blaine Miller's movies have received 8 votes in total, with an average rating of 7.0.
4. A. Cengiz Mert: The director A. Cengiz Mert's movies have received a total of 6 votes, and they have an average rating of 3.2.
5. A. Fishman: Movies directed by A. Fishman have received 37 votes on average, with an impressive average rating of 7.8.
6. A. Haluk Unal: This director's movies have received an average of 5 votes, and they have an excellent mean rating of 8.8.
7. A. Jagadesh: A. Jagadesh's movies have received a total of 291 votes, with an average rating of 3.5.
8. A. Joji: Movies directed by A. Joji have received 6 votes on average, with a mean rating of 5.5.
9. A. Karunakaran: A. Karunakaran's movies have received a substantial total of 4676 votes, and they have an average rating of 5.8125.
10. A. Lawrence Dreyfuss: Movies directed by A. Lawrence Dreyfuss have received 17 votes on average, with a mean rating of 7.0.

These statistics provide insights into how each director's movies are rated and the level of audience engagement, as measured by the total number of votes received. Directors with higher mean ratings and larger sums of numvotes may be considered more successful or influential in the industry.

5.4 Here’s the overall best directors based on highest average ratings (9 out of 10 directors got a perfect score on the average ratings.)

1. Michiel Brongers: Michiel Brongers has an impressive average rating of 10.0 based on 5 votes.
2. Masahiro Hayakawa: Masahiro Hayakawa also has a perfect average rating of 10.0 with 5 votes.

3. Emre Oran: Emre Oran maintains a perfect 10.0 average rating with 6 votes.
4. Lindsay Thompson: Lindsay Thompson's movies have an average rating of 10.0, based on 7 votes.
5. Ivana Diniz: Ivana Diniz's movies have an excellent average rating of 10.0, backed by 10 votes.
6. Tristan David Luciotti: Tristan David Luciotti's films have a 10.0 average rating with 6 votes.
7. Chad Carpenter: Chad Carpenter has a perfect average rating of 10.0 with 5 votes.
8. Stephen Peek: Stephen Peek's movies have an average rating of 10.0, supported by 20 votes.
9. Loreto Di Cesare: Loreto Di Cesare's films also maintain a perfect 10.0 average rating with 16 votes.
10. Kalyan Varma: Kalyan Varma's movies have a high average rating of 9.9 with 10 votes.

These directors have received exceptionally high average ratings for their work, suggesting that their movies are highly regarded by audiences

Recommendations

Based on the results obtained from the project, which include genre statistics, director statistics, and top directors by average ratings, we can draw several conclusions and make recommendations:

1. Genre Statistics (5.2):

The analysis of genre statistics has revealed that certain genres have exceptionally high mean ratings, indicating strong audience reception. However, it's crucial to consider that genres with extremely high ratings often have limited data, which means these ratings may not be representative of broader trends. Recommendations: For Microsoft's entry into the movie industry, it's advisable to explore genres that consistently perform well and have a substantial sample size for more reliable insights.

2. Director Statistics (5.3):

The director statistics provide valuable insights into how each director's movies are rated and the number of votes they have received. Several directors stand out with exceptionally high average ratings, although some have relatively small sample sizes. Recommendations: Collaborating with directors who consistently produce highly rated films could be a strategic move for Microsoft's movie studio. However, it's essential to consider the director's track record in terms of both ratings and the number of votes to assess their broader appeal.

3. Top Directors by Average Ratings (5.4):

The list of directors with the highest average ratings showcases filmmakers who have achieved perfect ratings (10.0) with a limited number of votes. While these directors demonstrate exceptional quality, the small sample sizes may not represent widespread audience sentiment. Recommendations: Microsoft could explore partnerships with top-rated directors for niche or specialized projects, but it's crucial to balance high ratings with audience reach. Collaborations should be based on a comprehensive evaluation of both factors. Project Limitations:

The project's conclusions are based on available data, and the quality of the insights depends on the dataset's completeness and representativeness. Some genres or directors may not have sufficient data, limiting the depth of analysis. The analysis does not account for temporal trends or changing audience preferences, which could impact future performance. Future Improvement Ideas:

- Continuously update and expand the dataset to include more recent movies and ratings.
- Consider sentiment analysis to understand audience sentiments and preferences in-depth.
- Explore factors beyond genres and directors, such as cast, budget, and marketing, to gain a comprehensive view of movie success.
- Develop predictive models to forecast the potential success of future movies based on various factors.
- Conduct market research to understand the competitive landscape and audience demands in the movie industry.

In summary, while the analysis provides valuable insights, it's important to approach decision-making in the movie industry with a comprehensive understanding of various factors, including audience trends and market dynamics. Collaboration with highly-rated directors and exploration of popular genres should be part of a broader strategy for Microsoft's entry into the