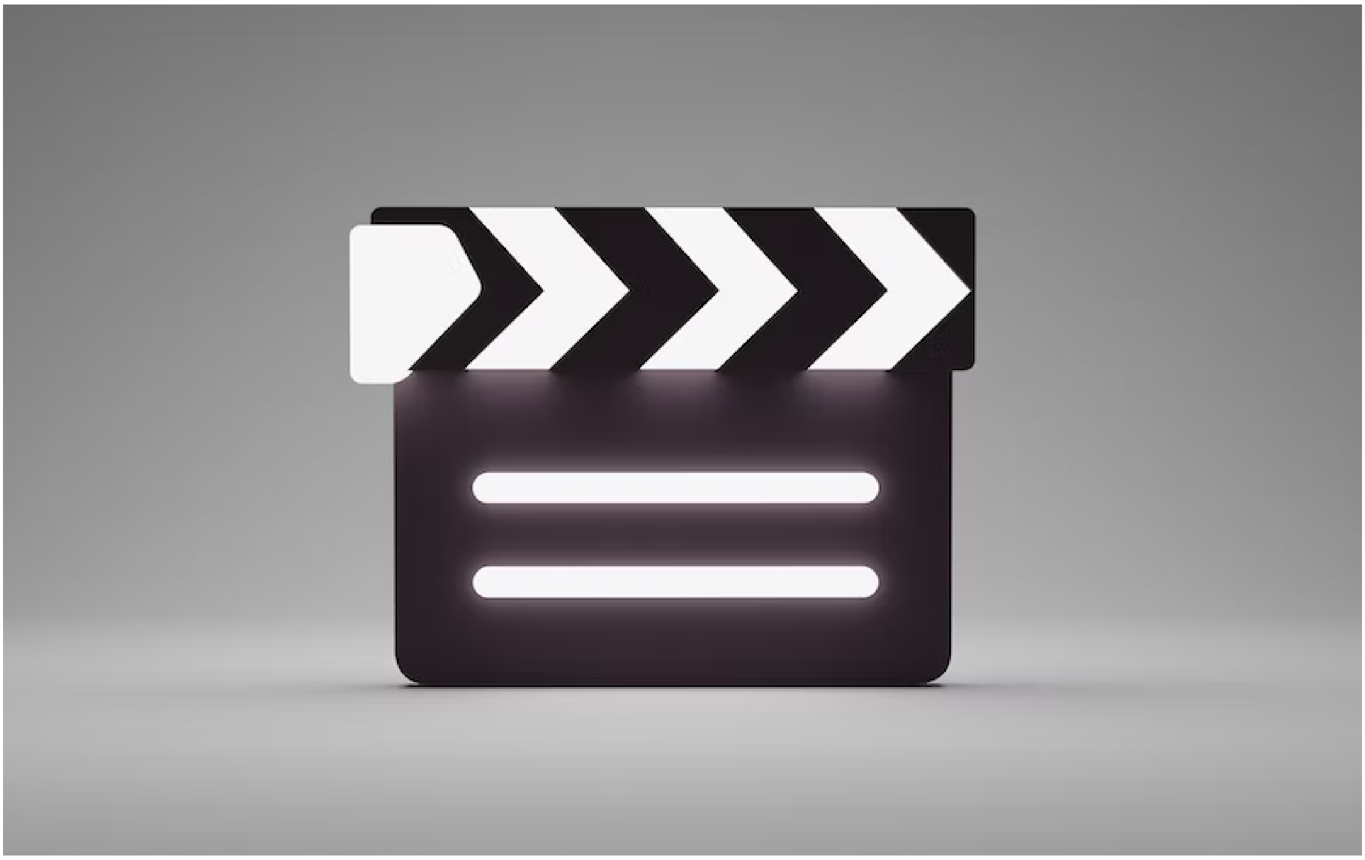


Microsoft Movie Studio Recommendations

Author: Stella Kitur

Student pace: Full time

Overview

This project aims to primarily assist Microsoft in creating a new movie studio by analyzing various data to identify the most profitable movies, top-rated directors, and genres among others.

The data analysis will involve statistical methods and data visualization to determine currently popular movie genres. The resulting insights will provide recommendations to the head of Microsoft's new movie studio on what types of films to produce, as well as an overview of potential competitors.

These recommendations will be based on empirical evidence and industry trends, enabling Microsoft to make informed decisions that increase the likelihood of their films being commercially successful.

Business Problem

As Microsoft has taken note on other big companies enjoying success by creating original video content, they also want a slice of the pie.

However, the big question is how?

As a result, I have been tasked to study what films are doing the best in the box office as well as to provide other valuable insights before a decision is made.

The questions that were asked in analysing this data are:

1. What were the top 5 rated studios?
2. What are the top rated genres?
3. What were the most profitable movies?
4. Who were the top rated directors?

These are relevant in answering the business problem at hand because as much as the profit made would be one of the key determinants of the success of the new movie studio, ensuring that the needs of the audience are met will ensure that Microsoft Movie Studio has an "edge" over the other studios through gaining a higher rating. This is made possible by identifying the top genres as well as directors.

Data Understanding

The data we're working with in this project is based on publicly available movie datasets. Our aim is to answer questions related to movie ratings, box office performance, and profitability. We're looking at movies that were released between 2010 and 2018 and gathering information such as ratings, budgets, and domestic and worldwide gross profits. Our target variable is the profitability of each movie. Some of the properties of the variables we're working with include both categorical and numerical data types, and there are a few missing values that we'll need to address.

In [172]:

```
# Import standard packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sqlite3

%matplotlib inline
```

Import Datasets

The datasets that will be used are loaded.

The datasets that are used in this project include:

```
1. Box Office Mojo Gross dataset
2. The Numbers Dataset
3. The IMDB dataset
```

In [173]:

```
#Load the datafiles in csv format first.
bom_movies_gross = pd.read_csv('./MS-Project/Data/bom.movie_gross.csv')
numbers_data = pd.read_csv('./MS-Project/Data/tn.movie_budgets.csv')
```

In [174]:

```
bom_movies_gross
```

Out[174]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.00	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.00	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.00	664300000	2010
3	Inception	WB	292600000.00	535700000	2010
4	Shrek Forever After	P/DW	238700000.00	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.00	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.00	NaN	2018
3384	El Pacto	Sony	2500.00	NaN	2018
3385	The Swan	Synergetic	2400.00	NaN	2018
3386	An Actor Prepares	Grav.	1700.00	NaN	2018

3387 rows × 5 columns

In [175]:

```
numbers_data
```

Out[175]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	$7,000	$0	$0
5778	79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
5779	80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
5780	81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
5781	82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

5782 rows × 6 columns

In [176]:

```
conn = sqlite3.connect('im.db')
imdb_df = pd.read_sql("""SELECT name FROM sqlite_master WHERE type = 'tabl
e';""", conn)
```

In [177]:

```
imdb_df
```

Out[177]:

	name
0	movie_basics
1	directors
2	known_for
3	movie_akas
4	movie_ratings
5	persons
6	principals
7	writers

In [178]:

```
imdb = pd.read_sql("""
    SELECT mb.primary_title, mb.original_title, mb.start_year, mb.runtime_minute
s,
             mb.genres, mr.averagerating, mr.numvotes, d.person_id, p.primary_name
    FROM movie_basics mb
    JOIN movie_ratings mr ON mr.movie_id = mb.movie_id
    JOIN directors d ON d.movie_id = mb.movie_id
    JOIN persons p ON p.person_id = d.person_id
""", conn)

imdb.head()
```

Out[178]:

	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	n
0	Sunghursh	Sunghursh	2013	175.00	Action,Crime,Drama	7.00	
1	Sunghursh	Sunghursh	2013	175.00	Action,Crime,Drama	7.00	
2	Sunghursh	Sunghursh	2013	175.00	Action,Crime,Drama	7.00	
3	Sunghursh	Sunghursh	2013	175.00	Action,Crime,Drama	7.00	
4	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.00	Biography,Drama	7.20	

In [179]:

```
imdb.shape
print(f" This tells us that there are {imdb.shape} rows and columns in this data set")
```

```
 This tells us that there are (181387, 9) rows and columns in this d
ataset
```

In [180]:

```
imdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181387 entries, 0 to 181386
Data columns (total 9 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   primary_title    181387 non-null  object 
 1   original_title   181387 non-null  object 
 2   start_year       181387 non-null  int64  
 3   runtime_minutes  163584 non-null  float64
 4   genres           180047 non-null  object 
 5   averagerating    181387 non-null  float64
 6   numvotes         181387 non-null  int64  
 7   person_id        181387 non-null  object 
 8   primary_name     181387 non-null  object 
dtypes: float64(2), int64(2), object(5)
memory usage: 12.5+ MB
```

In [181]:

```
# Summary statistics of this dataframe
pd.options.display.float_format = '{:.2f}'.format
imdb.describe()
```

Out[181]:

	start_year	runtime_minutes	averagerating	numvotes
count	181387.00	163584.00	181387.00	181387.00
mean	2014.31	97.79	6.22	4955.52
std	2.54	194.43	1.39	37609.31
min	2010.00	3.00	1.00	5.00
25%	2012.00	84.00	5.40	19.00
50%	2014.00	94.00	6.30	66.00
75%	2016.00	107.00	7.20	311.00
max	2019.00	51420.00	10.00	1841066.00

Data Preparation

To get the data ready for analysis, I started by dropping columns that weren't relevant to my research questions and merging data from different sources. Next, I looked for missing values and outliers in the data. For missing values, I used imputation techniques to fill in the gaps, while for outliers, I used both visualizations and statistical methods to identify and address them. I believe these choices were appropriate given the nature of the data and my research questions, as they allowed me to conduct a thorough and accurate analysis while keeping the most relevant variables.

In [182]:

```
bom_movies_gross.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3387 non-null   object 
 1   studio          3382 non-null   object 
 2   domestic_gross  3359 non-null   float64
 3   foreign_gross   2037 non-null   object 
 4   year            3387 non-null   int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 132.4+ KB
```

It is noted that in the *foreign_gross* column, it is in the datatype object.

Therefore, we will define a function that will change it to a float instead.

In [183]:

```
# create a function that will change foreign_gross into a float

def to_float(df,column_name):
    df[column_name] = pd.to_numeric(df[column_name], errors='coerce')
    return df
to_float(bom_movies_gross, 'foreign_gross')
bom_movies_gross
```

Out[183]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.00	652000000.00	2010
1	Alice in Wonderland (2010)	BV	334200000.00	691300000.00	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.00	664300000.00	2010
3	Inception	WB	292600000.00	535700000.00	2010
4	Shrek Forever After	P/DW	238700000.00	513900000.00	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.00	nan	2018
3383	Edward II (2018 re-release)	FM	4800.00	nan	2018
3384	El Pacto	Sony	2500.00	nan	2018
3385	The Swan	Synergetic	2400.00	nan	2018
3386	An Actor Prepares	Grav.	1700.00	nan	2018

3387 rows × 5 columns

As there are missing values in the *domestic_gross and foreign_gross* columns, we will fill the values with 0.

In [184]:

```
# Next we will fill in the missing values in domestic_gross and foreign_gross
bom_movies_gross['domestic_gross'].fillna(0, inplace = True)
bom_movies_gross['foreign_gross'].fillna(0, inplace= True)
bom_movies_gross.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3387 non-null   object 
 1   studio          3382 non-null   object 
 2   domestic_gross  3387 non-null   float64
 3   foreign_gross   3387 non-null   float64
 4   year            3387 non-null   int64  
dtypes: float64(2), int64(1), object(2)
memory usage: 132.4+ KB
```

For the missing data in the *studio* column will be dropped.

In [185]:

```
bom_movies_gross.dropna(subset=['studio'],inplace=True)
bom_movies_gross.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3382 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   title           3382 non-null   object
 1   studio          3382 non-null   object
 2   domestic_gross  3382 non-null   float64
 3   foreign_gross   3382 non-null   float64
 4   year            3382 non-null   int64
dtypes: float64(2), int64(1), object(2)
memory usage: 158.5+ KB
```

2. The Numbers Dataset

In [186]:

```
numbers_data
```

Out[186]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	$7,000	$0	$0
5778	79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
5779	80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
5780	81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
5781	82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

5782 rows × 6 columns

In [187]:

```
"""
In the columns production_budget, domestic_gross and worldwide_gross, there are
commas and $
which will interfere with the analysis process.
So we will run a code that will drop the commas and the $.
"""
```

Out[187]:

```
'\nIn the columns production_budget, domestic_gross and worldwide_gr
oss, there are commas and $\nwhich will interfere with the analysis
process.\nSo we will run a code that will drop the commas and the
$.\n'
```

In [188]:

```
numbers_data['production_budget']= pd.to_numeric(numbers_data.production_budget.
str.replace('[^\d.]',''))
```

In [189]:

```
numbers_data['domestic_gross']= pd.to_numeric(numbers_data.domestic_gross.str.re
place('[^\d.]',''))
```

In [190]:

```
numbers_data['worldwide_gross'] = pd.to_numeric(numbers_data.worldwide_gross.st
r.replace('[^\d.]',''))
```

In [191]:

```
numbers_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   id                 5782 non-null   int64 
 1   release_date       5782 non-null   object
 2   movie              5782 non-null   object
 3   production_budget  5782 non-null   int64 
 4   domestic_gross     5782 non-null   int64 
 5   worldwide_gross    5782 non-null   int64 
dtypes: int64(4), object(2)
memory usage: 271.2+ KB
```

In [192]:

```
numbers_data
```

Out[192]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	425000000	760507625	2776345279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	1403013963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	7000	0	0
5778	79	Apr 2, 1999	Following	6000	48482	240495
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000	1338	1338
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	0
5781	82	Aug 5, 2005	My Date With Drew	1100	181041	181041

5782 rows × 6 columns

In [193]:

```
# The datatype has been successfully changed to what was desired.
```

In [194]:

```
# Creating a new column called domestic profits
numbers_data['domestic_profits']= numbers_data['domestic_gross'] - numbers_data
['production_budget']
numbers_data
```

Out[194]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	dom
0	1	Dec 18, 2009	Avatar	425000000	760507625	2776345279	
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350	
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	1403013963	
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747	
...	...	...	...	...	...	...	
5777	78	Dec 31, 2018	Red 11	7000	0	0	
5778	79	Apr 2, 1999	Following	6000	48482	240495	
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000	1338	1338	
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	0	
5781	82	Aug 5, 2005	My Date With Drew	1100	181041	181041	

5782 rows × 7 columns

In [195]:

```
# Creating a new column called worldwide profits

numbers_data['worldwide_profits']= numbers_data['worldwide_gross'] - numbers_dat
a['production_budget']
numbers_data
```

Out[195]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	dom
0	1	Dec 18, 2009	Avatar	425000000	760507625	2776345279	
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350	
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	1403013963	
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747	
...	...	...	...	...	...	...	
5777	78	Dec 31, 2018	Red 11	7000	0	0	
5778	79	Apr 2, 1999	Following	6000	48482	240495	
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000	1338	1338	
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	0	
5781	82	Aug 5, 2005	My Date With Drew	1100	181041	181041	

5782 rows × 8 columns

In [196]:

```
numbers_data.sort_values(by='domestic_profits', ascending=False, inplace=True)
numbers_data
```

Out[196]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	dom
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2053311220	
41	42	Feb 16, 2018	Black Panther	200000000	700059566	1348258224	
42	43	Dec 19, 1997	Titanic	200000000	659363944	2208208395	
3464	65	May 25, 1977	Star Wars Ep. IV: A New Hope	11000000	460998007	786598007	
33	34	Jun 12, 2015	Jurassic World	215000000	652270625	1648854864	
...	...	...	...	...	...	...	
31	32	May 18, 2012	Battleship	220000000	65233400	313477717	
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	
12	13	Jul 2, 2013	The Lone Ranger	275000000	89302115	260002115	
13	14	Mar 9, 2012	John Carter	275000000	73058679	282778100	
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350	

5782 rows × 8 columns

This will enable us to create a barchart in the next section **Data Modeling**

3. The Movie_Basics and Movie_Ratings tables

Here we extract data from the *movie_basics and movie_ratings* table from the IMDB database.

In [197]:

```
#Load movie basics table
movie_basics_df = pd.read_sql("""
    SELECT *
        FROM movie_basics AS mb
        """,conn)
movie_basics_df.head()
```

Out[197]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.00	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.00	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.00	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	nan	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.00	Comedy,Drama,Fantasy

In [198]:

```
#Inspect the data in the data frame to identify necessary steps needed to be tak
en.
movie_basics_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   movie_id         146144 non-null  object 
 1   primary_title    146144 non-null  object 
 2   original_title   146123 non-null  object 
 3   start_year       146144 non-null  int64  
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object 
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
```

In [199]:

```
missing_values = movie_basics_df[['runtime_minutes', 'genres']].isnull().sum()

print(missing_values)
print("""
This shows the total of missing values in these two columns.
""")
```

```
runtime_minutes    31739
genres              5408
dtype: int64

This shows the total of missing values in these two columns.
```

Due to the large amount of missing values in the runtime_minutes we will drop it

And for the genres we will just replace the missing values so that it is it's own category

In [200]:

```
movie_basics_df.dropna(subset=['runtime_minutes'], inplace=True)
movie_basics_df
```

Out[200]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genre
0	tt0063540	Sunghursh	Sunghursh	2013	175.00	Action,Crime,Dram
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.00	Biography,Dram
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.00	Dram
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.00	Comedy,Drama,Fanta
5	tt0111414	A Thin Life	A Thin Life	2018	75.00	Comec
...	...	...	...	...	...	
146135	tt9916170	The Rehearsal	O Ensaio	2019	51.00	Dram
146136	tt9916186	Illenau - die Geschichte einer ehemaligen Heil...	Illenau - die Geschichte einer ehemaligen Heil...	2017	84.00	Documenta
146137	tt9916190	Safeguard	Safeguard	2019	90.00	Drama,Thrill
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.00	Dram
146142	tt9916730	6 Gunn	6 Gunn	2017	116.00	Nor

114405 rows × 6 columns

In [201]:

```
movie_basics_df['genres'] = movie_basics_df['genres'].fillna('Unknown')
movie_basics_df
```

Out[201]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genre
0	tt0063540	Sunghursh	Sunghursh	2013	175.00	Action,Crime,Dram
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.00	Biography,Dram
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.00	Dram
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.00	Comedy,Drama,Fanta
5	tt0111414	A Thin Life	A Thin Life	2018	75.00	Comec
...	...	...	...	...	...	
146135	tt9916170	The Rehearsal	O Ensaio	2019	51.00	Dram
146136	tt9916186	Illenau - die Geschichte einer ehemaligen Heil...	Illenau - die Geschichte einer ehemaligen Heil...	2017	84.00	Documenta
146137	tt9916190	Safeguard	Safeguard	2019	90.00	Drama,Thrill
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.00	Dram
146142	tt9916730	6 Gunn	6 Gunn	2017	116.00	Unknow

114405 rows × 6 columns

In [202]:

```
# Drop duplicates from movie_id
movie_basics_df.drop_duplicates(subset=['movie_id'], inplace= True)
```

In [203]:

```
movie_basics_df.isnull().sum()
```

Out[203]:

```
movie_id           0
primary_title      0
original_title     4
start_year         0
runtime_minutes    0
genres             0
dtype: int64
```

In [204]:

```
#Identify the movies that are duplicates in original_title column

missing_original_title = movie_basics_df[movie_basics_df['original_title'].isnul
l()]
print(missing_original_title)
```

```
        movie_id                            primary_title original_t
itle  \
39095   tt2397619               Woody Allen: A Documentary
None
58624   tt3414266                          The Outer Loop
None
115983  tt6882442  Hirugao: Love Affairs in the Afternoon
None
116350  tt6911842                              Senioritus
None

        start_year  runtime_minutes   genres
39095         2012           195.00  Unknown
58624         2013            78.00  Unknown
115983        2017           125.00  Romance
116350        2017            75.00  Unknown
```

In [205]:

```
# Replaces "None" in the original_title column with the value in the primary_tit
le
mask = movie_basics_df['original_title'].isna()
movie_basics_df.loc[mask, 'original_title'] = movie_basics_df.loc[mask, 'primary
_title']
print(movie_basics_df)
```

```
        movie_id                                primary_title  \
0       tt0063540                                    Sunghursh
1       tt0066787              One Day Before the Rainy Season
2       tt0069049                   The Other Side of the Wind
4       tt0100275                    The Wandering Soap Opera
5       tt0111414                                  A Thin Life
...           ...                                          ...
146135  tt9916170                                The Rehearsal
146136  tt9916186  Illenau - die Geschichte einer ehemaligen Heil...
146137  tt9916190                                    Safeguard
146139  tt9916538                          Kuambil Lagi Hatiku
146142  tt9916730                                       6 Gunn

                                         original_title  start_yea
r  \
0                                             Sunghursh        201
3
1                                        Ashad Ka Ek Din        201
9
2                            The Other Side of the Wind        201
8
4                                 La Telenovela Errante        201
7
5                                           A Thin Life        201
8
...                                                 ...
...
146135                                         O Ensaio        201
9
146136  Illenau - die Geschichte einer ehemaligen Heil...        201
7
146137                                        Safeguard        201
9
146139                              Kuambil Lagi Hatiku        201
9
146142                                           6 Gunn        201
7

        runtime_minutes                genres
0                175.00    Action,Crime,Drama
1                114.00       Biography,Drama
2                122.00                 Drama
4                 80.00  Comedy,Drama,Fantasy
5                 75.00                Comedy
...                 ...                   ...
146135            51.00                 Drama
146136            84.00           Documentary
146137            90.00        Drama,Thriller
146139           123.00                 Drama
146142           116.00               Unknown

[114405 rows x 6 columns]
```

In [206]:

```
# check if the value in the original_title column was successfully replaced for
a specific primary_title
primary_title = 'Woody Allen: A Documentary'
original_title = movie_basics_df.loc[movie_basics_df['primary_title'] == primary
_title, 'original_title'].values[0]
print(original_title)
```

```
Woody Allen: A Documentary
```

In [207]:

```
movie_basics_df.drop_duplicates()
```

Out[207]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genre
0	tt0063540	Sunghursh	Sunghursh	2013	175.00	Action,Crime,Dram
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.00	Biography,Dram
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.00	Dram
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.00	Comedy,Drama,Fantas
5	tt0111414	A Thin Life	A Thin Life	2018	75.00	Comec
...	...	...	...	...	...	
146135	tt9916170	The Rehearsal	O Ensaio	2019	51.00	Dram
146136	tt9916186	Illenau - die Geschichte einer ehemaligen Heil...	Illenau - die Geschichte einer ehemaligen Heil...	2017	84.00	Documenta
146137	tt9916190	Safeguard	Safeguard	2019	90.00	Drama,Thrill
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.00	Dram
146142	tt9916730	6 Gunn	6 Gunn	2017	116.00	Unknow

114405 rows × 6 columns

In [208]:

```
# Load movie_ratings table
movie_ratings_df = pd.read_sql("""
    SELECT *
    FROM movie_ratings AS mr
    """, conn)

movie_ratings_df.head()
```

Out[208]:

	movie_id	averagerating	numvotes
0	tt10356526	8.30	31
1	tt10384606	8.90	559
2	tt1042974	6.40	20
3	tt1043726	4.20	50352
4	tt1060240	6.50	21

In [209]:

```
#Inspect the data in the data frame to identify necessary steps needed to be tak
en.
movie_ratings_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   movie_id       73856 non-null  object 
 1   averagerating  73856 non-null  float64
 2   numvotes       73856 non-null  int64  
dtypes: float64(1), int64(1), object(1)
memory usage: 1.7+ MB
```

In [210]:

```
movie_ratings_df
```

Out[210]:

	movie_id	averagerating	numvotes
0	tt10356526	8.30	31
1	tt10384606	8.90	559
2	tt1042974	6.40	20
3	tt1043726	4.20	50352
4	tt1060240	6.50	21
...	...	...	...
73851	tt9805820	8.10	25
73852	tt9844256	7.50	24
73853	tt9851050	4.70	14
73854	tt9886934	7.00	5
73855	tt9894098	6.30	128

73856 rows × 3 columns

No further actions were taken in this dataframe.

Data Visualization and Analysis

The questions asked were:

1. **What were the top 5 rated studios?**
2. **What are the top rated genres?**
3. **What were the most profitable movies?**
4. **Who were the top rated directors? **

Top Studios

In order to determine the top 5 rated studios in the industry, the following code was run in order to group together the studio as well as the gross income received.

In [211]:

```
# Group by studio and sum the values to get the total gross income so far per st
udio
studio_list = bom_movies_gross.loc[:,
          ["studio","domestic_gross","foreign_gross"]].groupby("studio").sum()
```

In [212]:

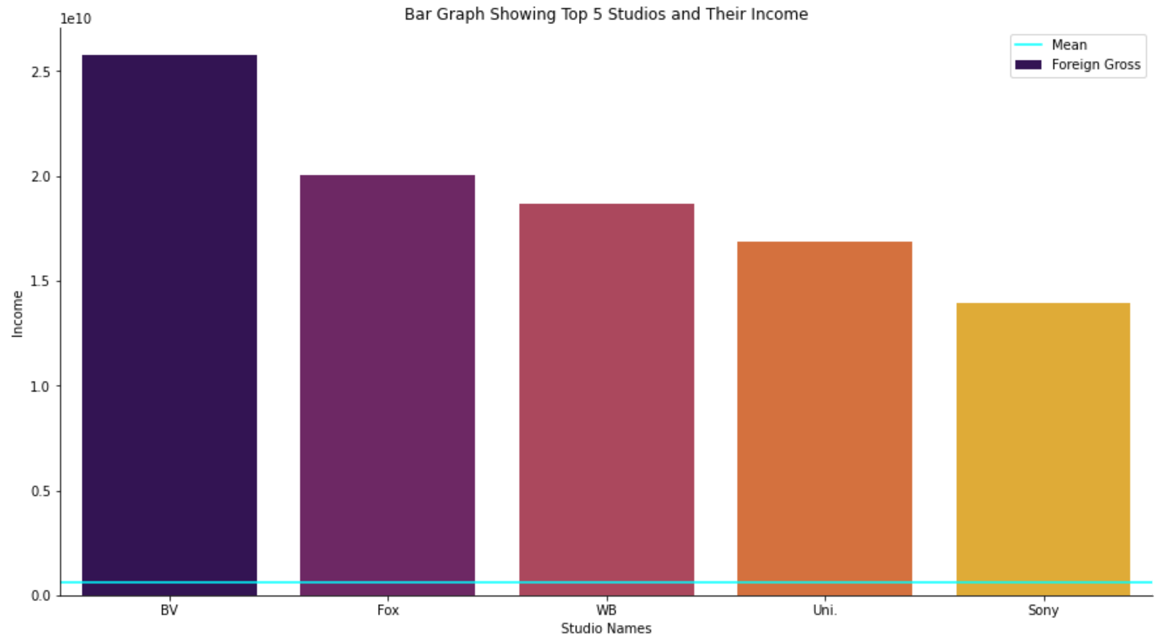
```
# sort the top studios by foreign gross and display only the top 5 studios
studio_list.sort_values(by="foreign_gross",ascending=False, inplace=True)
top_studios = studio_list.iloc[:5]
top_studios.reset_index(inplace = True)
top_studios
```

Out[212]:

	studio	domestic_gross	foreign_gross
0	BV	18419029199.00	25793852199.00
1	Fox	10949499997.00	20055866599.00
2	WB	12168046000.00	18667902998.00
3	Uni.	12902393000.00	16854767999.00
4	Sony	8459683098.00	13945354998.00

In [213]:

```
plt.figure(figsize=(15,8))
ax = sns.barplot(data=top_studios, x='studio', y='foreign_gross', palette='infer
no', label='Foreign Gross')
ax.axhline(y=studio_list['foreign_gross'].mean(), color='cyan', label='Mean')

# set the x and y labels and title
plt.xlabel('Studio Names')
plt.ylabel('Income')
plt.title('Bar Graph Showing Top 5 Studios and Their Income')

# show the legend
ax.legend(loc="upper right")

# remove the top and right spines
sns.despine()
plt.show()
```



As illustrated in the barchart above, the top 5 best rated studios include:

1. **BV (Buena Vista)**
2. **Fox**
3. **WB (Warner Brothers)**
4. **Universal Studios and**
5. **Sony**

This is necessary in regards to the business problem as it highlights the competition that Microsoft Studios would be up against.

Top rated Genres

In order to determine the top rated genres in the industry, the genres were grouped together based on their average rating.

In [214]:

```
genre_ratings = imdb.groupby('genres')['averagerating'].mean().reset_index()
genre_ratings
```

Out[214]:

	genres	averagerating
0	Action	5.75
1	Action,Adult,Comedy	4.65
2	Action,Adventure	5.31
3	Action,Adventure,Animation	6.32
4	Action,Adventure,Biography	7.16
...	...	...
916	Thriller	5.64
917	Thriller,War	5.43
918	Thriller,Western	6.75
919	War	6.24
920	Western	5.02

921 rows × 2 columns

In [215]:

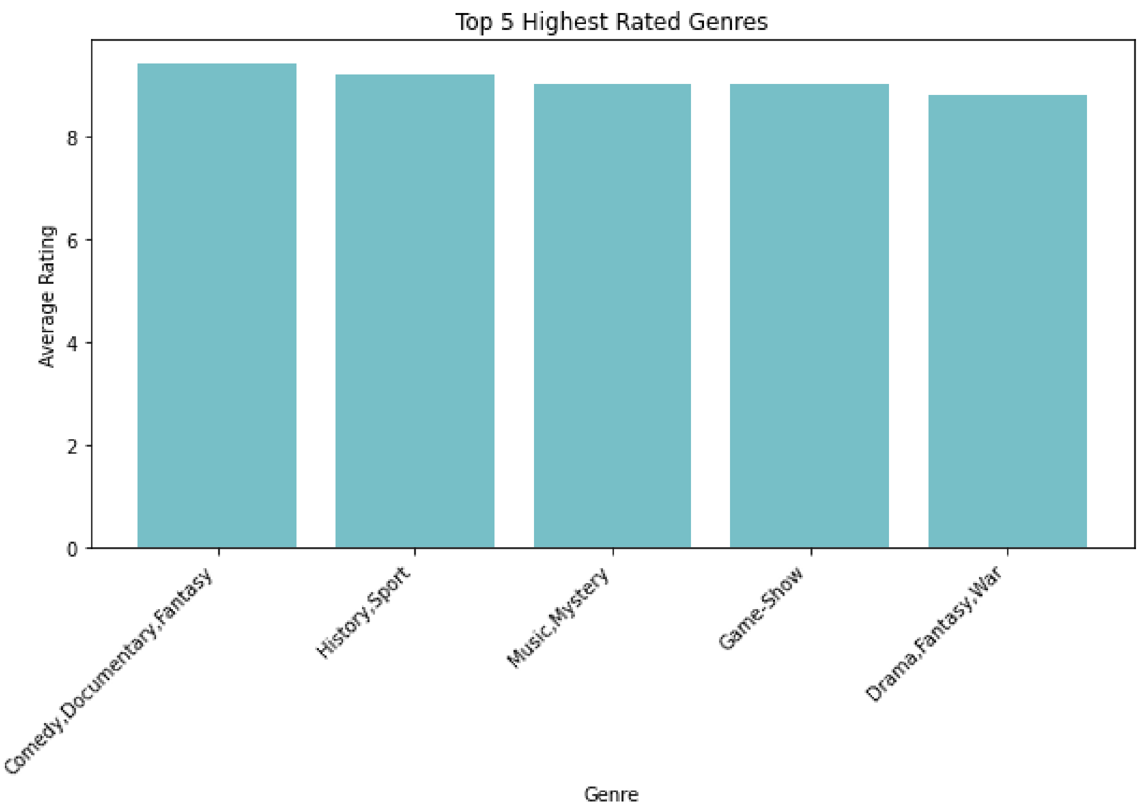
```
top_genres = genre_ratings.sort_values('averagerating', ascending=False).head(5)
top_genres
```

Out[215]:

	genres	averagerating
449	Comedy,Documentary,Fantasy	9.40
849	History,Sport	9.20
880	Music,Mystery	9.00
835	Game-Show	9.00
715	Drama,Fantasy,War	8.80

In [216]:

```
fig, ax = plt.subplots(figsize=(10, 5))
ax.bar(top_genres['genres'], top_genres['averagerating'], color='#77BFC7')

# set x and y axis labels and title
ax.set_xlabel('Genre')
ax.set_ylabel('Average Rating')
ax.set_title('Top 5 Highest Rated Genres')

# rotate x-axis labels to avoid overlapping
plt.xticks(rotation=45, ha='right')

# show plot
plt.show()
```



Based on the barchart illustrated, the top rated genres include :

1. **Comedy, Documentary, Fantasy**
2. **History, Sport**
3. **Music, Mystery**
4. **Game shows**
5. **Drama, Fantasy, War**

This gives us an overview of what types of films are well rated in the box office.

Furthermore, using this data we wanted to explore what are the 5 lowest rated genres, this was in order to be able to determine what the audience also does not enjoy.

In [217]:

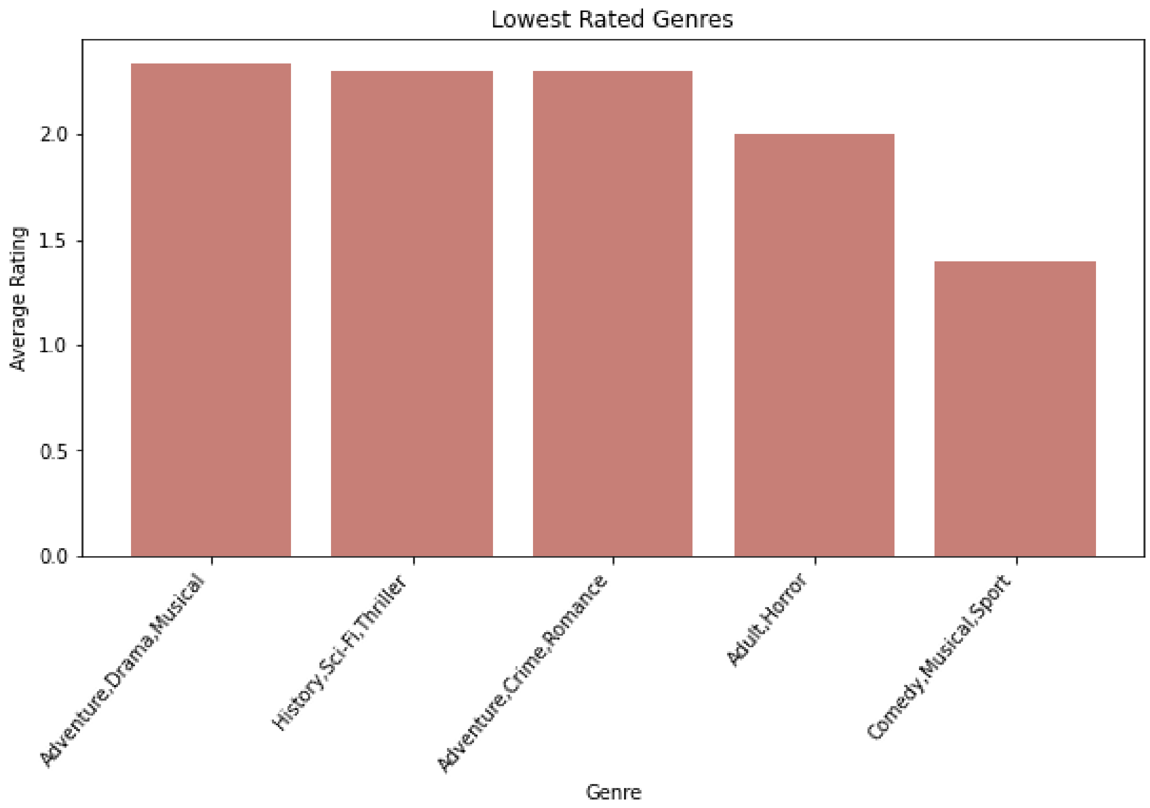
```
# Find the lowest rated genres
lowest_genres = genre_ratings.sort_values('averagerating', ascending=False).tail
(5)
lowest_genres
```

Out[217]:

	genres	averagerating
219	Adventure,Drama,Musical	2.33
848	History,Sci-Fi,Thriller	2.30
195	Adventure,Crime,Romance	2.30
153	Adult,Horror	2.00
527	Comedy,Musical,Sport	1.40

In [218]:

```
#Create the barplot

fig, ax = plt.subplots(figsize=(10, 5))
ax.bar(lowest_genres['genres'], lowest_genres['averagerating'], color='#C77F77')

# set x and y axis labels and title
ax.set_xlabel('Genre')
ax.set_ylabel('Average Rating')
ax.set_title('''
Lowest Rated Genres''')

# rotate x-axis labels to avoid overlapping
plt.xticks(rotation=50, ha='right')

# show plot
plt.show()
```



The lowest rated genres are displayed in the bar chart above, this highlights that the audience are likely to give a lower rating on films that belong to the following genres:

1. **Adventure, Drama, Musical**
2. **History, Sci-Fi, Thriller**
3. **Adventure, Crime, Romance**
4. **Adult, Horror**
5. **Comedy, Musical, Sport**

Profitable Movies

Next, the dataset The Numbers were used in order to explore the top 5 most profitable movies.

In [219]:

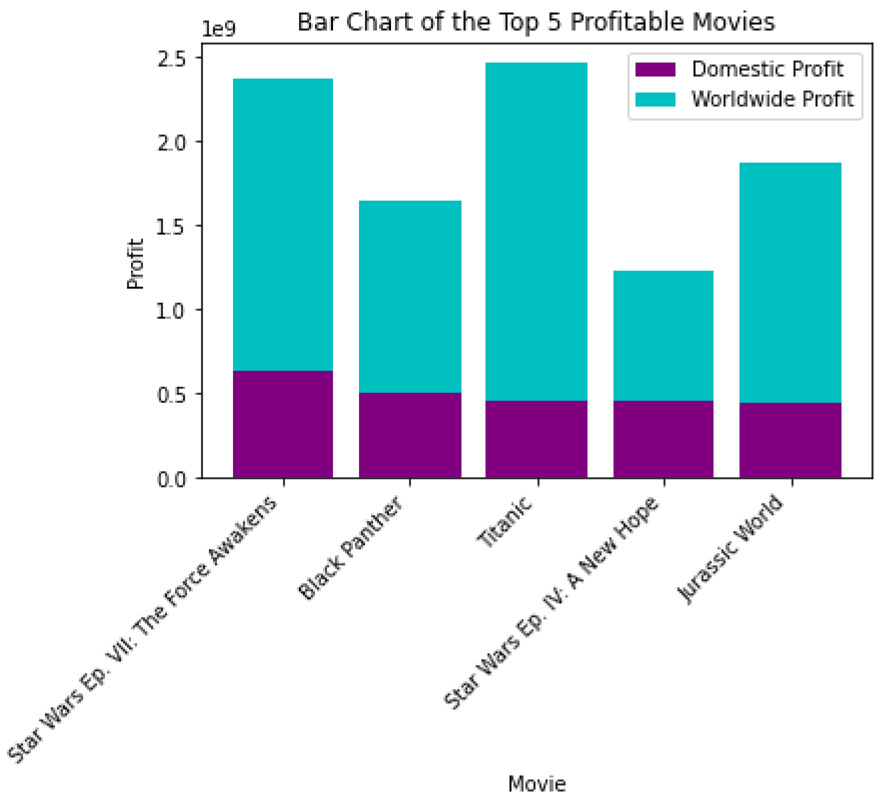
```
top_5_movies = numbers_data.head(5)
top_5_movies
```

Out[219]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	dome:
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2053311220	
41	42	Feb 16, 2018	Black Panther	200000000	700059566	1348258224	
42	43	Dec 19, 1997	Titanic	200000000	659363944	2208208395	
3464	65	May 25, 1977	Star Wars Ep. IV: A New Hope	11000000	460998007	786598007	
33	34	Jun 12, 2015	Jurassic World	215000000	652270625	1648854864	

In [220]:

```
plt.bar(top_5_movies['movie'], top_5_movies['domestic_profits'], color ='purpl
e')
plt.bar(top_5_movies['movie'], top_5_movies['worldwide_profits'], bottom=top_5_m
ovies['domestic_profits'], color ='c')

# Add labels and title to the plot
plt.xlabel('Movie')
plt.ylabel('Profit')
plt.title('Bar Chart of the Top 5 Profitable Movies')
plt.legend(labels=['Domestic Profit', 'Worldwide Profit'])
plt.xticks(rotation=45, ha='right')
# Display the plot
plt.show()
```



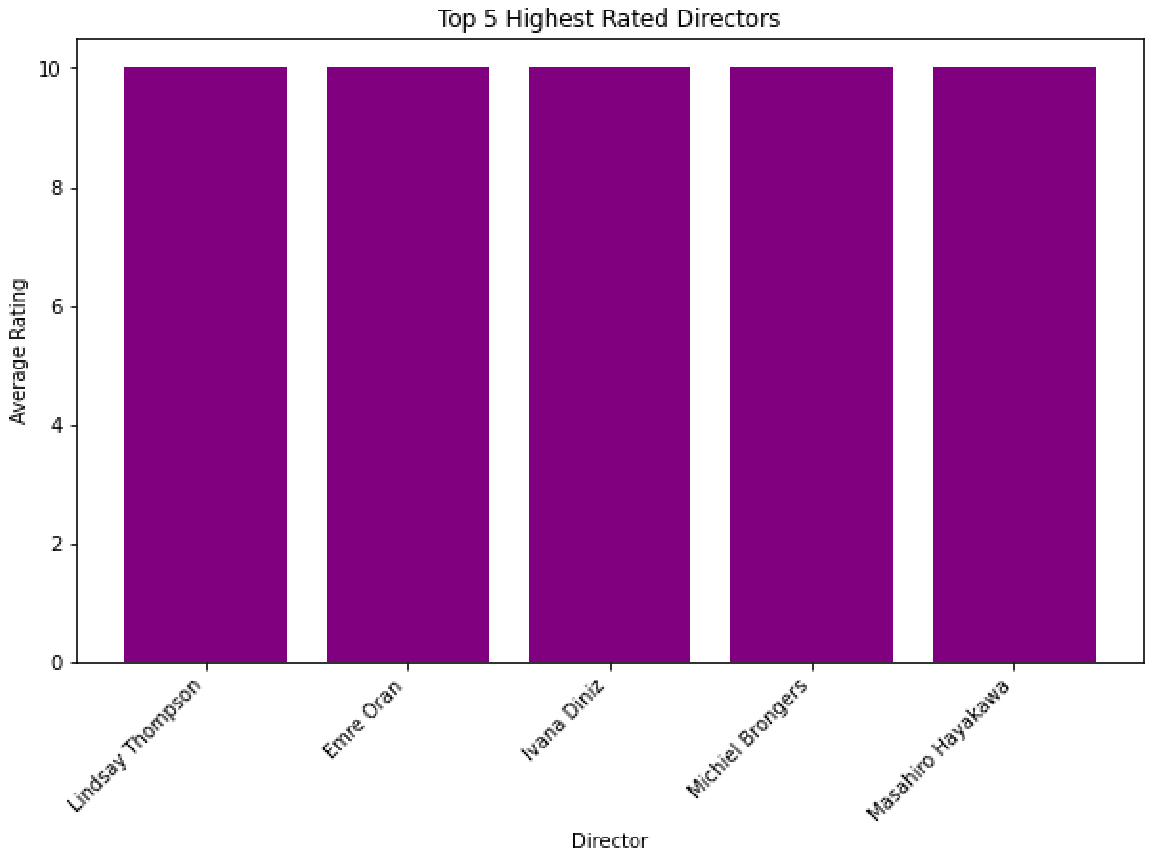
As shown in the above barchart, it is worth mentioning that in comparison to *domestic_profits, worldwide_profits* were significantly higher.

The Top Rated Directors

Finally, in order to find out who the top directors are, the data for the directors was grouped together with the average rating. Then, the new dataframe was cleaned to drop any duplicate values and then sorted through to retrieve the top 5 directors.

In [221]:

```
# calculate average rating by director
director_ratings = imdb.groupby('primary_name')['averagerating'].mean().reset_in
dex()

# drop duplicates
director_ratings.drop_duplicates(subset='primary_name', keep='first', inplace=Tr
ue)

# sort by rating in descending order and extract the top 10 highest rated direct
ors
top_directors = director_ratings.sort_values('averagerating', ascending=False).h
ead(5)

# create bar plot
fig, ax = plt.subplots(figsize=(10, 6))
ax.bar(top_directors['primary_name'], top_directors['averagerating'], color='pur
ple')

# set x and y axis labels and title
ax.set_xlabel('Director')
ax.set_ylabel('Average Rating')
ax.set_title('Top 5 Highest Rated Directors')

# rotate x-axis labels to avoid overlapping
plt.xticks(rotation=45, ha='right')

# show plot
plt.show()
```



In [222]:

```
top_directors
```

Out[222]:

	primary_name	averagerating
31252	Lindsay Thompson	10.00
15370	Emre Oran	10.00
21566	Ivana Diniz	10.00
36721	Michiel Brongers	10.00
34538	Masahiro Hayakawa	10.00

As noted by the barchart and displayed in the table above, the following directors are the most top rated directors as they all have an average rating of 10.00

Recommendations

Recommendation 1: Focus on the Top Rated Genres

Due to there being higher ratings on those types of films, Microsoft should consider focusing on those genres to ensure customer satisfaction, if that is one of their goals.

Recommendation 2: Focus on the Foreign/Worldwide audience

According to the analysis done regarding the profits, the highest profits generated was due to worldwide profits.

Therefore, where the focus is on increasing profits more emphasis should be placed on a global audience rather than only reaching the domestic audience.

Recommendation 3: Try and work with the top rated directors

Thirdly, based on the analysis, I would recommend that Microsoft considers directing movies with those that have a higher rating due to the quality of work that is produced by them.

Recommendation 4: Study the competition further

While we have identified the other top studios based on income, it would be necessary to further explore why they are at the top.

Therefore, I suggest further analysis is done as based on the SWOT method, it can highlight potential threats to Microsoft's latest venture or expose opportunities that they can use for their success.

Conclusions

In conclusion, through the use of the datasets The Numbers, Box office Mojo and the IMDB database, an exploratory data analysis has allowed to a certain extent answer the business problem.

However, there are limitations to this analysis, which include:

1. **Causation does not mean correlarion**

As the saying goes, "Causation is not correlation ", so while there may be higher ratings for certain variable such as genres, it may not only be the only factor leading to this result and therefore further analysis would be needed to provide a more accurate result.

1. **Accuracy**

The lack of using all the datasets provided as well as how to use them effectively is a limitation to the results provided which may therefore influence the accuracy of the results returned.

Future Improvements:

One area of improvement would be to complete a SWOT Analysis based on the findings as this would be beneficial to Microsoft and help in the decision making process.

Additionally, I would improve on the way data is explored, for example to find out what the Return of Investment (RoI) would be for this scenario.