Final Project Submission

Please fill out:

- Student name: Geoffrey Rotich
- Student pace: full time
- Scheduled project review date/time: 15/09/2023
- Instructor name:
- Blog post URL:

Overview

This project mainly deals with the ways in which microsoft can implement in order to catch up with its competitors like Netflix in the film industry.Since its a new movie studio Microsoft needs a lot of research done in order to catch up qickly with the trends. The movie industry is a crucial domain in modern world entertainment hence it is set to gain a lot of profit on the current market

Background of the Business

The movie industry started to take shape in the late 19th century and the early 20th century. It started really flourishing in the early 20's with motion pictures being one of the early industries to promote the industry.While microsoft which was started in 1975 really missed a lot of the trends and innovations made early on in the filmmaking industry.The movie industry mainly works on creating or recreating extraordinary scenarios in real world or fantasy.

Domain of the business

The industry really tries to make the viewers have amaizing experiences while watching the shows. The viewers take a step from reality to virtual scenarios happening somewhere else. The films require a lot of staff cooperation including actors managers and etc.

Business Case

This project is necessary for the business because; -Gives wide research from various other industries in the trend. -Easy reading of the visualisations. -Easy understanding of the data given. -Good examples of modern trends.

The project aims to take opportunity of the current methods of making profits from the industry.

Business Understanding

The filmmaking industry venturing into sets to revolutionalize entertainment to a whole new level.The innovations in things like camera quality's and graphics really have a huge impact in the profits being made which is even improving customer satisfaction that much.

Data Understanding

This will focus on getting the detailed information about the business venture in filmmaking and understanding the data.

In [127]:
```
# Import the necessary libraries for the data

import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
import sqlite3
%matplotlib inline
```

In [ ]:
```

```

In [128]:
```
# Creating a function that returns the summary of the data.
def summary(data):
    shape = data.shape
    columns = data.columns
    info = data.info()

    return shape, columns, info
```

We can set the display format of our values so that we do not use the scientific notations.

In [129]:
```
# Setting the display format to not use scientific notation.
pd.set_option('display.float_format', lambda x: '%.0f' % x)
```

We can now read the `.csv` files used for the research.This dataframe shows how much revenue movies made in the USA(domestic_gross)and globally(foreign_gross).The data was obtained from `bom.movies_gross.csv.gz` which was obtained from `https://www.boxofficemojo.com/`

In [130]:
```
# To read and check the data.
df1 = pd.read_csv("bom.movie_gross.csv")
df1.head()
```

Out[130]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000	664300000	2010
3	Inception	WB	292600000	535700000	2010
4	Shrek Forever After	P/DW	238700000	513900000	2010

In [131]:
```
#check summary
summary(df1)
```

```
<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
```

Out[131]:

```
((3387, 5),
 Index(['title', 'studio', 'domestic_gross', 'foreign_gross', 'year'], dtype
='object'),
 None)
```

In [132]:
```
#reading the second dataframe
df2 = pd.read_csv("tmdb.movies.csv")
df2.head()
```

Out[132]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	tit
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	34	2010-11-19	Har Pott and t Death Hallow Part
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	29	2010-03-26	How Tra Yo Drag
2	2	[12, 28, 878]	10138	en	Iron Man 2	29	2010-05-07	Iron M
3	3	[16, 35, 10751]	862	en	Toy Story	28	1995-11-22	T Sto
4	4	[28, 878, 12]	27205	en	Inception	28	2010-07-16	Incepti

In [133]:
```
#summary of df2
summary(df2)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         26517 non-null  int64  
 1   genre_ids          26517 non-null  object 
 2   id                 26517 non-null  int64  
 3   original_language  26517 non-null  object 
 4   original_title     26517 non-null  object 
 5   popularity         26517 non-null  float64
 6   release_date       26517 non-null  object 
 7   title              26517 non-null  object 
 8   vote_average       26517 non-null  float64
 9   vote_count         26517 non-null  int64  
dtypes: float64(2), int64(3), object(5)
memory usage: 2.0+ MB
```

Out[133]:

```
((26517, 10),
 Index(['Unnamed: 0', 'genre_ids', 'id', 'original_language', 'original_titl
e',
        'popularity', 'release_date', 'title', 'vote_average', 'vote_count'],
       dtype='object'),
 None)
```

In [134]:
```
#Final data frame
df3 = pd.read_csv("rt.movie_info.tsv", sep='\t', encoding='latin1')
df3.head()
```

Out[134]:

	id	synopsis	rating	genre	director	writer	theater_date	d
0	1	This gritty, fast-paced, and innovative police...	R	Action and Adventure|Classics|Drama	William Friedkin	Ernest Tidyman	Oct 9, 1971	
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 2012	
2	5	Illeana Douglas delivers a superb performance ...	R	Drama|Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 1996	
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama|Mystery and Suspense	Barry Levinson	Paul Attanasio|Michael Crichton	Dec 9, 1994	
4	7	NaN	NR	Drama|Romance	Rodney Bennett	Giles Cooper	NaN	

In [135]:
```
#summary for df3
summary(df3)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1560 entries, 0 to 1559
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   id            1560 non-null   int64 
 1   synopsis      1498 non-null   object
 2   rating        1557 non-null   object
 3   genre         1552 non-null   object
 4   director      1361 non-null   object
 5   writer        1111 non-null   object
 6   theater_date  1201 non-null   object
 7   dvd_date      1201 non-null   object
 8   currency      340 non-null    object
 9   box_office    340 non-null    object
 10  runtime       1530 non-null   object
 11  studio        494 non-null    object
dtypes: int64(1), object(11)
memory usage: 146.4+ KB
```

Out[135]:
```
((1560, 12),
 Index(['id', 'synopsis', 'rating', 'genre', 'director', 'writer',
        'theater_date', 'dvd_date', 'currency', 'box_office', 'runtime',
        'studio'],
       dtype='object'),
 None)
```

Next we check our database where we will focus only on `movie_basics` and `movie_ratings` tables.

In [136]:
```
#Connecting to the database
conn = sqlite3.connect("im.db")

#Create a cursor
cur = conn.cursor()

#Checking the table names
cur.execute("""SELECT name FROM sqlite_master WHERE type = "table" """)
table_names = cur.fetchall()
table_names
```

Out[136]:
```
[('movie_basics',),
 ('directors',),
 ('known_for',),
 ('movie_akas',),
 ('movie_ratings',),
 ('persons',),
 ('principals',),
 ('writers',)]
```

Data Cleaning

Here we will outline the steps taken to clean the data, including handling missing values and duplicates seen in the dataframes above.

In [137]:
```
# Creating a function that returns the number of duplicate values
def sum_duplicates(data):
    return data.duplicated().sum()
```

In [138]:
```
# Create a function that returns the percentage of null values in each column.
def null_percentage(data):
    null_count = data.isnull().sum()
    total_count = data.shape[0]
    percentage = (null_count / total_count) * 100
    return percentage
```

In [139]:
```
# Find percentage of null values in df1
null_percentage(df1)
```

Out[139]:
```
title             0
studio            0
domestic_gross    1
foreign_gross    40
year              0
dtype: float64
```

In [140]:
```
# finding the number of duplicates in df1.
sum_duplicates(df1)
```

Out[140]:
```
0
```

We need to remove the null values to make it easier to analyse the data from our previous study

In [141]:
```
# Dropping the rows with null values
df1 = df1.dropna()
```

In [142]:
```
#Summary of df1
summary(df1)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2007 entries, 0 to 3353
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           2007 non-null   object 
 1   studio          2007 non-null   object 
 2   domestic_gross  2007 non-null   float64
 3   foreign_gross   2007 non-null   object 
 4   year            2007 non-null   int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 94.1+ KB
```

Out[142]:
```
((2007, 5),
 Index(['title', 'studio', 'domestic_gross', 'foreign_gross', 'year'], dtype
='object'),
 None)
```

We can see the number of rows reduced to 2007 due to our function we created above for dropping the rows

We still need to fix the `foreign_gross` and `year` datatypes and change them to `float` and `object` respectively

In [143]:
```
#Changing the datatype
df1['year'] = df1['year'].astype(object)
df1['foreign_gross'] = pd.to_numeric(df1['foreign_gross'], errors='coerce')

#Checking the summary
summary(df1)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2007 entries, 0 to 3353
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           2007 non-null   object 
 1   studio          2007 non-null   object 
 2   domestic_gross  2007 non-null   float64
 3   foreign_gross   2002 non-null   float64
 4   year            2007 non-null   object 
dtypes: float64(2), object(3)
memory usage: 94.1+ KB
```

Out[143]:
```
((2007, 5),
 Index(['title', 'studio', 'domestic_gross', 'foreign_gross', 'year'], dtype
='object'),
 None)
```

In [144]:
```
#Now going to df2
df2.head()
```

Out[144]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	ti
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	34	2010-11-19	Ha Pot and t Death Hallow Par
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	29	2010-03-26	How Tra Yo Drag
2	2	[12, 28, 878]	10138	en	Iron Man 2	29	2010-05-07	Iron M
3	3	[16, 35, 10751]	862	en	Toy Story	28	1995-11-22	T Sto
4	4	[28, 878, 12]	27205	en	Inception	28	2010-07-16	Incepti

In [145]:
```
# summary  of df2.
summary(df2)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         26517 non-null  int64  
 1   genre_ids          26517 non-null  object 
 2   id                 26517 non-null  int64  
 3   original_language  26517 non-null  object 
 4   original_title     26517 non-null  object 
 5   popularity         26517 non-null  float64
 6   release_date       26517 non-null  object 
 7   title              26517 non-null  object 
 8   vote_average       26517 non-null  float64
 9   vote_count         26517 non-null  int64  
dtypes: float64(2), int64(3), object(5)
memory usage: 2.0+ MB
```

Out[145]:

```
((26517, 10),
 Index(['Unnamed: 0', 'genre_ids', 'id', 'original_language', 'original_titl
e',
        'popularity', 'release_date', 'title', 'vote_average', 'vote_count'],
       dtype='object'),
 None)
```

In [146]:
```
# Checking percentage of null values in df2.
null_percentage(df2)
```

Out[146]:
```
Unnamed: 0          0
genre_ids           0
id                  0
original_language   0
original_title      0
popularity          0
release_date        0
title               0
vote_average        0
vote_count          0
dtype: float64
```

In [147]:
```
# Checking for duplicate values in df2
sum_duplicates(df2)
```

Out[147]:
```
0
```

We need to change the datatype of the `id` column

In [148]:
```
#changing the id column
df2['id'] = df2['id'].astype(object)

# Checking the summary
summary(df2)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         26517 non-null  int64  
 1   genre_ids          26517 non-null  object 
 2   id                 26517 non-null  object 
 3   original_language  26517 non-null  object 
 4   original_title     26517 non-null  object 
 5   popularity         26517 non-null  float64
 6   release_date       26517 non-null  object 
 7   title              26517 non-null  object 
 8   vote_average       26517 non-null  float64
 9   vote_count         26517 non-null  int64  
dtypes: float64(2), int64(2), object(6)
memory usage: 2.0+ MB
```

Out[148]:
```
((26517, 10),
 Index(['Unnamed: 0', 'genre_ids', 'id', 'original_language', 'original_titl
e',
        'popularity', 'release_date', 'title', 'vote_average', 'vote_count'],
       dtype='object'),
 None)
```

In [149]:
```
#Now on df3
df3.head()
```

Out[149]:

	id	synopsis	rating	genre	director	writer	theater_date	d
0	1	This gritty, fast-paced, and innovative police...	R	Action and Adventure|Classics|Drama	William Friedkin	Ernest Tidyman	Oct 9, 1971	
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 2012	
2	5	Illeana Douglas delivers a superb performance ...	R	Drama|Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 1996	
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama|Mystery and Suspense	Barry Levinson	Paul Attanasio|Michael Crichton	Dec 9, 1994	
4	7	NaN	NR	Drama|Romance	Rodney Bennett	Giles Cooper	NaN	

In [150]:
```
#Summary of df3
summary(df3)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1560 entries, 0 to 1559
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   id            1560 non-null   int64 
 1   synopsis      1498 non-null   object
 2   rating        1557 non-null   object
 3   genre         1552 non-null   object
 4   director      1361 non-null   object
 5   writer        1111 non-null   object
 6   theater_date  1201 non-null   object
 7   dvd_date      1201 non-null   object
 8   currency      340 non-null    object
 9   box_office    340 non-null    object
 10  runtime       1530 non-null   object
 11  studio        494 non-null    object
dtypes: int64(1), object(11)
memory usage: 146.4+ KB
```

Out[150]:
```
((1560, 12),
 Index(['id', 'synopsis', 'rating', 'genre', 'director', 'writer',
        'theater_date', 'dvd_date', 'currency', 'box_office', 'runtime',
        'studio'],
       dtype='object'),
 None)
```

In [151]:
```
# Checking for missing values
null_percentage(df3)
```

Out[151]:
```
id              0
synopsis        4
rating          0
genre           1
director       13
writer         29
theater_date   23
dvd_date       23
currency       78
box_office     78
runtime         2
studio         68
dtype: float64
```

In [152]:
```
#Dropping the null values .
df3 = df3.dropna()
```

In [153]:
```
# Checking for duplicate values.
sum_duplicates(df3)
```

Out[153]:
```
0
```

In [154]:
```
#summary of df3
summary(df3)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 235 entries, 1 to 1545
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   id            235 non-null    int64 
 1   synopsis      235 non-null    object
 2   rating        235 non-null    object
 3   genre         235 non-null    object
 4   director      235 non-null    object
 5   writer        235 non-null    object
 6   theater_date  235 non-null    object
 7   dvd_date      235 non-null    object
 8   currency      235 non-null    object
 9   box_office    235 non-null    object
 10  runtime       235 non-null    object
 11  studio        235 non-null    object
dtypes: int64(1), object(11)
memory usage: 23.9+ KB
```

Out[154]:
```
((235, 12),
 Index(['id', 'synopsis', 'rating', 'genre', 'director', 'writer',
        'theater_date', 'dvd_date', 'currency', 'box_office', 'runtime',
        'studio'],
       dtype='object'),
 None)
```

the datatype of `id` needs to be `object` ,it is an anomaly and the data type of `box_office` should be changed too since its values are numbers

In [155]:
```
# Ensure 'box_office' is treated as a string
df3['box_office'] = df3['box_office'].astype(str)
# Removing the commas in the values for easy conversion.
df3['box_office'] = df3['box_office'].str.replace(',', '')

# Changing the datatypes
df3['id'] = df3['id'].astype(object)
df3['box_office'] = df3['box_office'].astype(int)
summary(df3)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 235 entries, 1 to 1545
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   id            235 non-null    object
 1   synopsis      235 non-null    object
 2   rating        235 non-null    object
 3   genre         235 non-null    object
 4   director      235 non-null    object
 5   writer        235 non-null    object
 6   theater_date  235 non-null    object
 7   dvd_date      235 non-null    object
 8   currency      235 non-null    object
 9   box_office    235 non-null    int32 
 10  runtime       235 non-null    object
 11  studio        235 non-null    object
dtypes: int32(1), object(11)
memory usage: 22.9+ KB
```

Out[155]:
```
((235, 12),
 Index(['id', 'synopsis', 'rating', 'genre', 'director', 'writer',
        'theater_date', 'dvd_date', 'currency', 'box_office', 'runtime',
        'studio'],
       dtype='object'),
 None)
```

Exploratory Data Analysis

Univariate Analysis

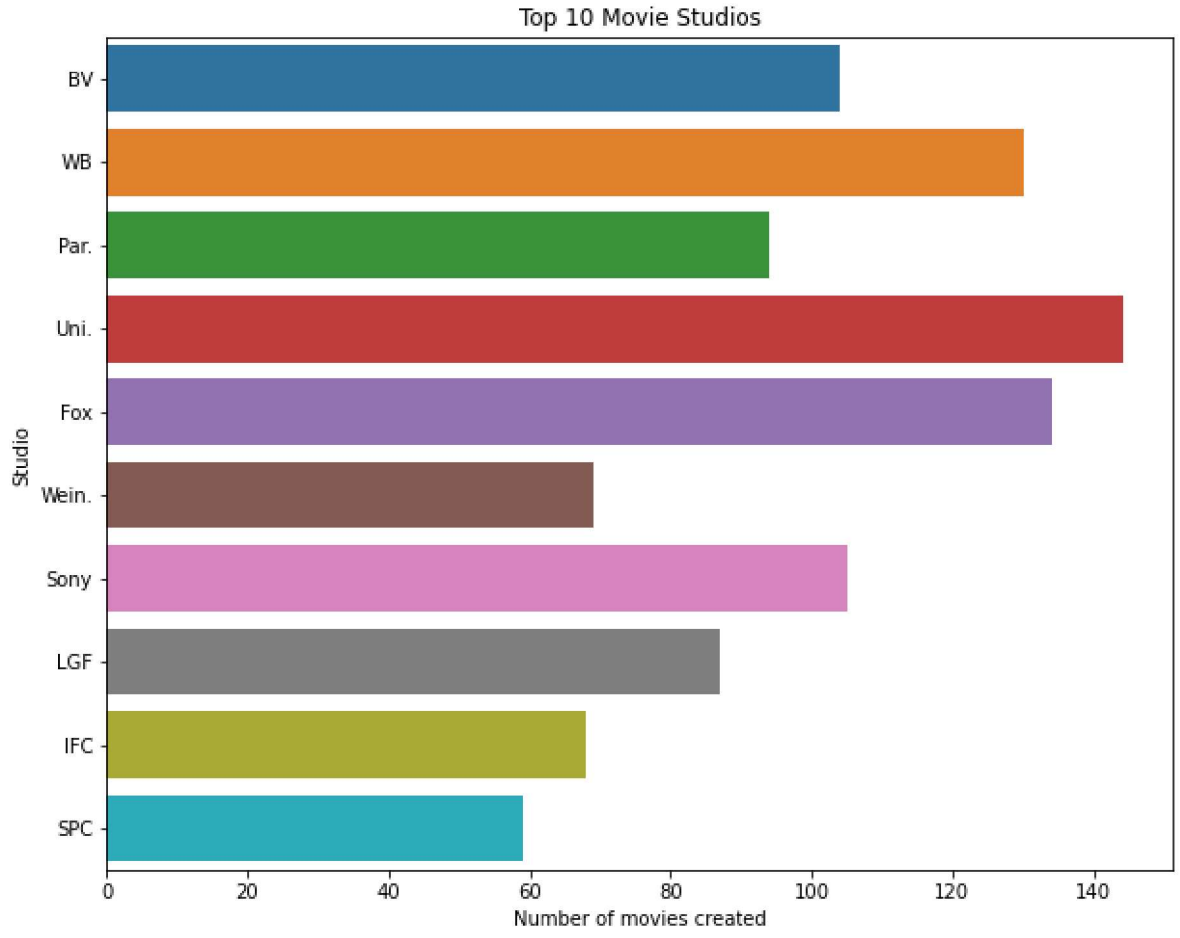
We will start on the first dataframe focusing only on `studio` column.

In [156]:
```
# Analysing the  studio  column.
df1['studio'].value_counts()
```

Out[156]:
```
Uni.          144
Fox           134
WB            130
Sony          105
BV            104
              ...
SV              1
FOAK            1
LGP             1
Greenwich       1
BGP             1
Name: studio, Length: 172, dtype: int64
```

In [157]:
```
# Plotting horizontal bar graph of studio

# Setting up the number of items to be shown in the studio graph
num_items = 10

# Subsetting it
subset = df1['studio'].value_counts().nlargest(num_items)

# Filtering df1
subset_data = df1[df1['studio'].isin(subset.index)]

# Setting up the figure and axes
fig, (ax1) = plt.subplots( figsize = (10, 8))

# Plotting the studio horizontal bar graph
sns.countplot(y='studio', data = subset_data, ax=ax1, orient = 'h')
ax1.set_xlabel('Number of movies created')
ax1.set_ylabel('Studio')
ax1.set_title('Top 10 Movie Studios')
```

Out[157]:
```
Text(0.5, 1.0, 'Top 10 Movie Studios')
```



We can see the top 3 studios being `Uni` , `Fox` , `WB` respectively

Bivariate analysis

Still on df1: Here we will focuse on `foreign_gross` and `domestic_gross` looking at central tendency and measures of dispersion

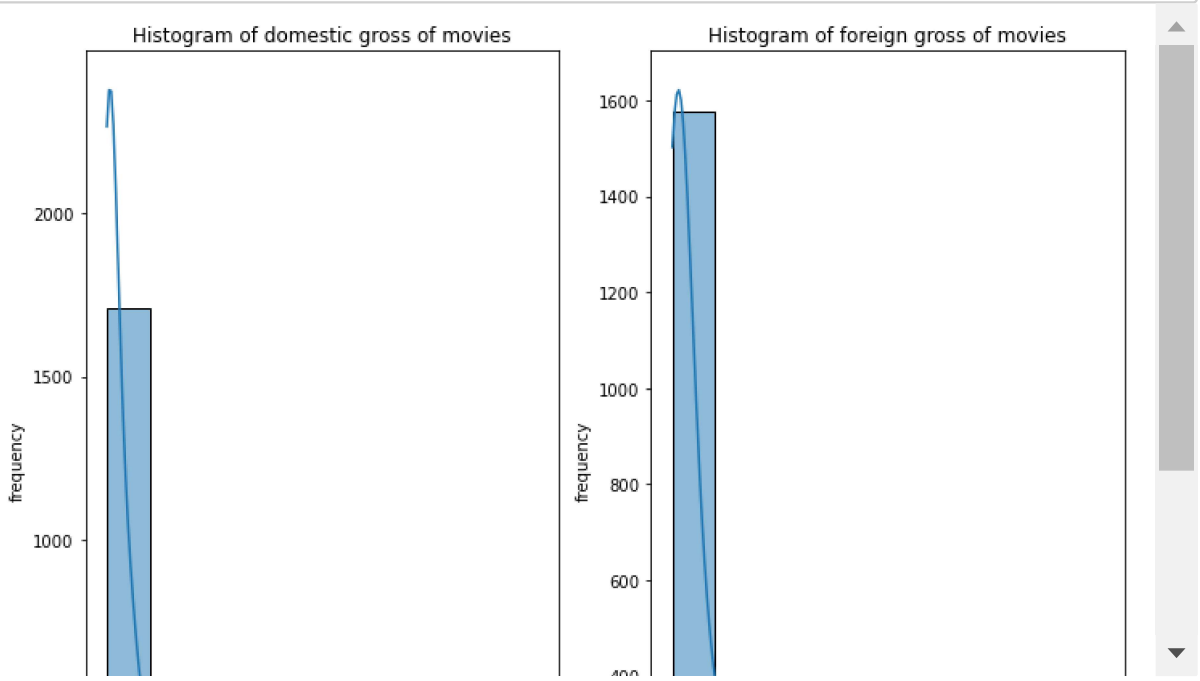
In [158]:
```
#Check on their statistics first
df1.describe()
```

Out[158]:

	domestic_gross	foreign_gross
count	2007	2002
mean	47019840	75979669
std	81626889	138300073
min	400	600
25%	670000	4000000
50%	16700000	19600000
75%	56050000	76450000
max	936700000	960500000

We can also use a histogram for our data for easier visualisation and understanding.

In [159]:
```
# Plotting the histograms

# Setting up the figure and axes
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 8))

# Plotting the domestic gross histogram.
sns.histplot(data = df1, x = 'domestic_gross', bins =  10, kde = True, ax=ax1)
ax1.set_xlabel('domestic gross')
ax1.set_ylabel('frequency')
ax1.set_title('Histogram of domestic gross of movies')

# Plotting the foreign gross hitogram.
sns.histplot(data = df1, x = 'foreign_gross', bins =  10, kde = True, ax=ax2)
ax2.set_xlabel('foreign gross')
ax2.set_ylabel('frequency')
ax2.set_title('Histogram of foreign gross of movies')

# Adjusting the layout to prevent overlap
plt.tight_layout()

# Showing the plot
plt. show();
```



From the histogram we can tell that the foreign_gross has more freqency hence providing a good chance for Microsoft to invest now in movie industry to gain profits since the trend is set to continue on for long.

In [160]:
```
#Cleaning df2
df2.head()
```

Out[160]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	tit
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	34	2010-11-19	Har Pott and t Death Hallow Part
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	29	2010-03-26	How Tra Yo Drag
2	2	[12, 28, 878]	10138	en	Iron Man 2	29	2010-05-07	Iron M
3	3	[16, 35, 10751]	862	en	Toy Story	28	1995-11-22	T Sto
4	4	[28, 878, 12]	27205	en	Inception	28	2010-07-16	Incepti

In [161]:
```
#Summary of df2
summary(df2)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         26517 non-null  int64  
 1   genre_ids          26517 non-null  object 
 2   id                 26517 non-null  object 
 3   original_language  26517 non-null  object 
 4   original_title     26517 non-null  object 
 5   popularity         26517 non-null  float64
 6   release_date       26517 non-null  object 
 7   title              26517 non-null  object 
 8   vote_average       26517 non-null  float64
 9   vote_count         26517 non-null  int64  
dtypes: float64(2), int64(2), object(6)
memory usage: 2.0+ MB
```

Out[161]:
```
((26517, 10),
 Index(['Unnamed: 0', 'genre_ids', 'id', 'original_language', 'original_titl
e',
        'popularity', 'release_date', 'title', 'vote_average', 'vote_count'],
       dtype='object'),
 None)
```

We need to remove the `unnamed` column since itsnot in same format as the other data types

In [162]:
```
# Dropping the unnamed column
df2 = df2.drop(columns=['Unnamed: 0'])
```

In [163]:
```
summary(df2)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   genre_ids          26517 non-null  object 
 1   id                 26517 non-null  object 
 2   original_language  26517 non-null  object 
 3   original_title     26517 non-null  object 
 4   popularity         26517 non-null  float64
 5   release_date       26517 non-null  object 
 6   title              26517 non-null  object 
 7   vote_average       26517 non-null  float64
 8   vote_count         26517 non-null  int64  
dtypes: float64(2), int64(1), object(6)
memory usage: 1.8+ MB
```

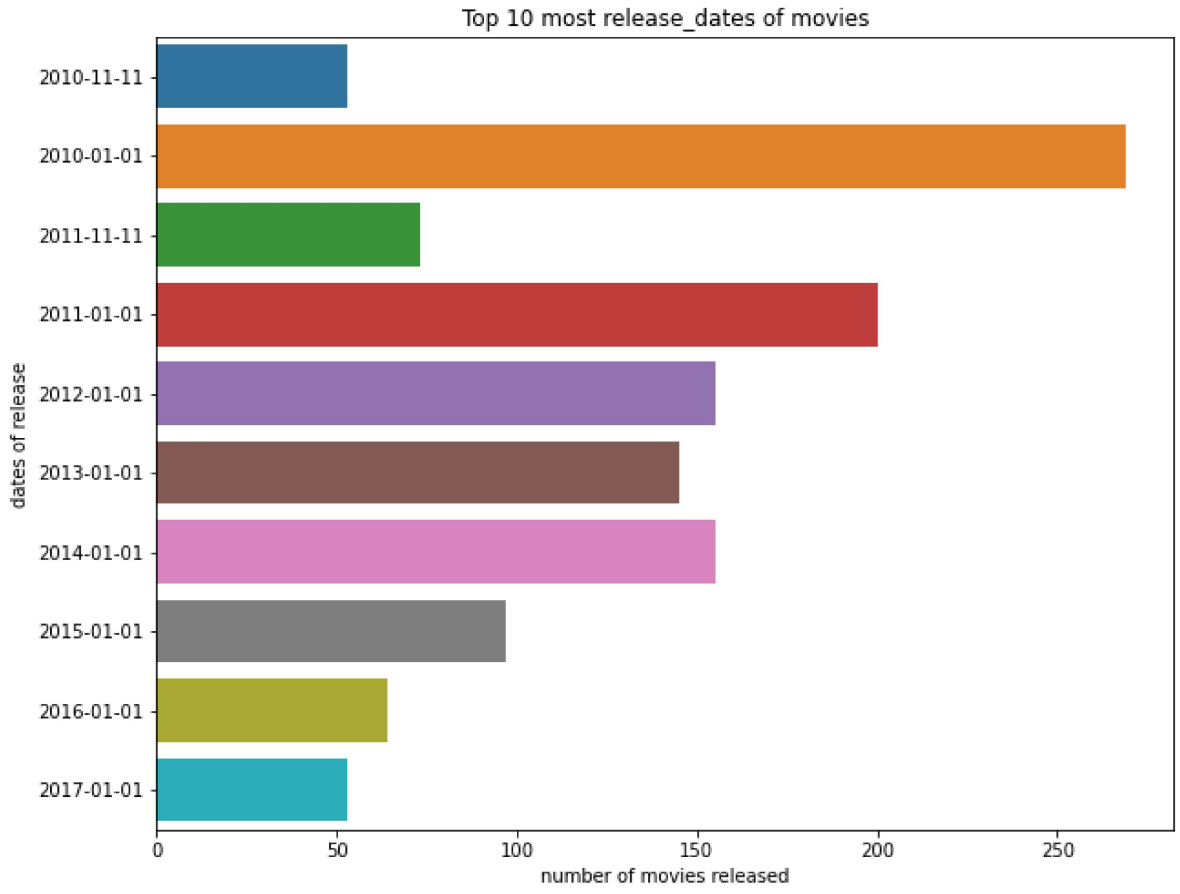
Out[163]:
```
((26517, 9),
 Index(['genre_ids', 'id', 'original_language', 'original_title', 'popularit
y',
        'release_date', 'title', 'vote_average', 'vote_count'],
       dtype='object'),
 None)
```

Our main focus in df2 is `release_date` .(univariate)

In [164]:
```
# Getting the number of unique values in the column
df2['release_date'].value_counts()
```

Out[164]:
```
2010-01-01    269
2011-01-01    200
2014-01-01    155
2012-01-01    155
2013-01-01    145
             ...
2000-10-06      1
2015-06-25      1
2017-08-23      1
2010-03-25      1
2012-06-28      1
Name: release_date, Length: 3433, dtype: int64
```

In [165]:
```
# Plotting a bar graph of relese_date

# Setting up the number of items to be shown in the graph
num_items2 = 10

# Subsetting it
subset2 = df2['release_date'].value_counts().nlargest(num_items2)

# Filtering df1
subset_data2 = df2[df2['release_date'].isin(subset2.index)]

# Setting up the figure and axes
fig, ax = plt.subplots(figsize = (10, 8))

# Plotting the horizontal bar graph
sns.countplot(y='release_date', data = subset_data2)
ax.set_xlabel('number of movies released')
ax.set_ylabel('dates of release')
ax.set_title('Top 10 most release_dates of movies')

# Showing the plot
plt. show();
```



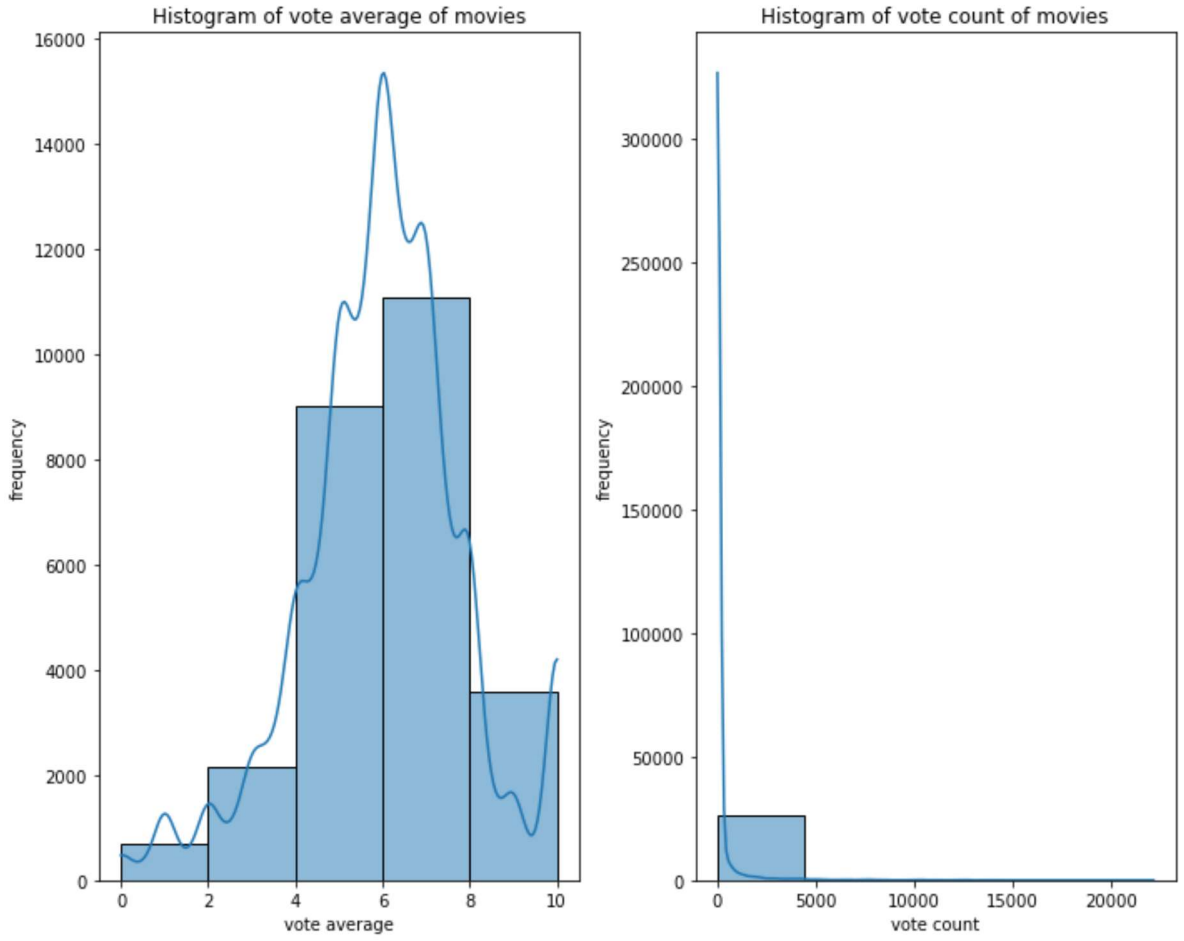
We still don't have enough statistics for our data,we can use the measures of central tendency and measures of dispersion to help us mainly focusing on `vote_average` and `vote_count`

In [166]:
```
# Statistics
df2.describe()
```

Out[166]:

	popularity	vote_average	vote_count
count	26517	26517	26517
mean	3	6	194
std	4	2	961
min	1	0	1
25%	1	5	2
50%	1	6	5
75%	4	7	28
max	81	10	22186

In [167]:
```
# Plotting histograms
# Setting up the figure and axes
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 8))



# Plotting the vote average hitogram.
sns.histplot(data = df2, x = 'vote_average', bins =  5, kde = True, ax=ax1)
ax1.set_xlabel('vote average')
ax1.set_ylabel('frequency')
ax1.set_title('Histogram of vote average of movies')

# Plotting the vote count hitogram.
sns.histplot(data = df2, x = 'vote_count', bins =  5, kde = True, ax=ax2)
ax2.set_xlabel('vote count')
ax2.set_ylabel('frequency')
ax2.set_title('Histogram of vote count of movies')

# Adjusting the layout to prevent overlap
plt.tight_layout()

# Showing the plot
plt. show();
```



We can see that the vote average is evenly spaced on both sides and the vote count is more inclined to the left side like the previous analysis taken in df1.This means that although the release of movies in 2012 was that hig,not many people were watching them as it should.Hence showing the steady decline in release of movies the following years since the rate of production and consumption was imbalanced

Now the third dataframe;

In [168]:
```
#Data of df3
df3.head()
```

Out[168]:

	id	synopsis	rating	genre	director	writer	theater_date	d
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 2012	
6	10	Some cast and crew from NBC's highly acclaimed...	PG-13	Comedy	Jake Kasdan	Mike White	Jan 11, 2002	
7	13	Stewart Kane, an Irishman living in the Austra...	R	Drama	Ray Lawrence	Raymond Carver|Beatrix Christian	Apr 27, 2006	
15	22	Two-time Academy Award Winner Kevin Spacey giv...	R	Comedy|Drama|Mystery and Suspense	George Hickenlooper	Norman Snider	Dec 17, 2010	
18	25	From ancient Japan's most enduring tale, the e...	PG-13	Action and Adventure|Drama|Science Fiction and...	Carl Erik Rinsch	Chris Morgan|Hossein Amini	Dec 25, 2013	

In [169]:
```
#Summary of df3
summary(df3)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 235 entries, 1 to 1545
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   id            235 non-null    object
 1   synopsis      235 non-null    object
 2   rating        235 non-null    object
 3   genre         235 non-null    object
 4   director      235 non-null    object
 5   writer        235 non-null    object
 6   theater_date  235 non-null    object
 7   dvd_date      235 non-null    object
 8   currency      235 non-null    object
 9   box_office    235 non-null    int32 
 10  runtime       235 non-null    object
 11  studio        235 non-null    object
dtypes: int32(1), object(11)
memory usage: 22.9+ KB
```

Out[169]:
```
((235, 12),
 Index(['id', 'synopsis', 'rating', 'genre', 'director', 'writer',
        'theater_date', 'dvd_date', 'currency', 'box_office', 'runtime',
        'studio'],
       dtype='object'),
 None)
```

We will work on `rating` , `genre` and `runtime` to help with the analysis and research

In [170]:
```
# To see the value counts of the genre column
df3['genre'].value_counts()
```

Out[170]:
```
Drama                                                  33
Comedy                                                 32
Comedy|Drama                                           22
Drama|Mystery and Suspense                             11
Comedy|Drama|Romance                                    9
                                                       ..
Action and Adventure|Drama|Horror|Mystery and Suspense  1
Action and Adventure|Art House and International        1
Documentary|Musical and Performing Arts                 1
Action and Adventure|Comedy                             1
Art House and International|Horror|Mystery and Suspense 1
Name: genre, Length: 76, dtype: int64
```

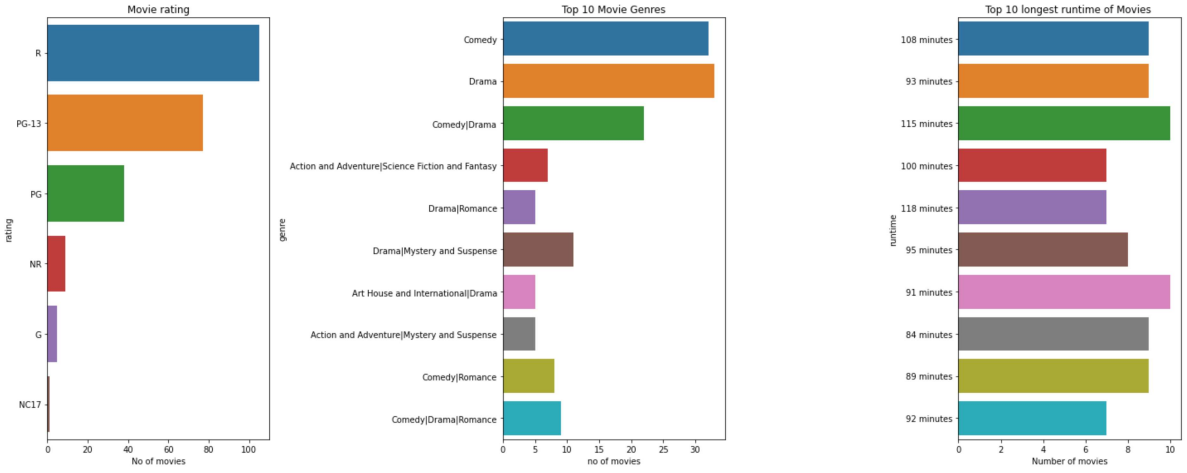
In [171]:
```
# To see the value counts of the rating column
df3['rating'].value_counts()
```

Out[171]:
```
R         105
PG-13      77
PG         38
NR          9
G           5
NC17        1
Name: rating, dtype: int64
```

In [172]:
```
# To see the value counts of the runtime column
df3['runtime'].value_counts()
```

Out[172]:
```
91 minutes     10
115 minutes    10
93 minutes      9
84 minutes      9
89 minutes      9
               ..
113 minutes     1
67 minutes      1
81 minutes      1
145 minutes     1
80 minutes      1
Name: runtime, Length: 69, dtype: int64
```

In [200]:
```
# Plotting horizontal bar graphs of genre and runtime

# Setting up the number of items to be shown in the genre and runtime graph
num_items3 = 10

# Subsetting it
subset3 = df3['genre'].value_counts().nlargest(num_items3)
subset4 = df3['runtime'].value_counts().nlargest(num_items3)

# Filtering df1
subset_data3 = df3[df3['genre'].isin(subset3.index)]
subset_data4 = df3[df3['runtime'].isin(subset4.index)]

# Setting up the figure and axes
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize = (20, 8))

# Plotting the horizontal bar graphs
# Plotting the rating horizontal bar graph
sns.barplot(x=df3['rating'].value_counts(), y=df3['rating'].unique(), ax=ax1)
ax1.set_xlabel('No of movies')
ax1.set_ylabel('rating')
ax1.set_title('Movie rating')

# Plotting the rating horizontal bar graph
sns.countplot(y='genre', data = subset_data3, ax=ax2)
ax2.set_xlabel('no of movies')
ax2.set_ylabel('genre')
ax2.set_title('Top 10 Movie Genres')

# Plotting the rating horizontal bar graph
sns.countplot(y='runtime', data = subset_data4, ax=ax3)
ax3.set_xlabel('Number of movies')
ax3.set_ylabel('runtime')
ax3.set_title('Top 10 longest runtime of Movies ')

# Adjusting the layout to prevent overlap
plt.tight_layout()

# Showing the plot
plt.show();
```



We can see that R rated movies were the most watched then following the order as they appear. Drama was mainly watched and closely followed by comedy as the most watched genres. Comedy has the longest runtime yet it still is one of the best genres meaning it must be very entertaining to many people.It would be advisable if Microsoft started first on making comedy/drama genre movies to get a lot of viewers easily.

We can use the numeric column `box_office` for more analysis of our data

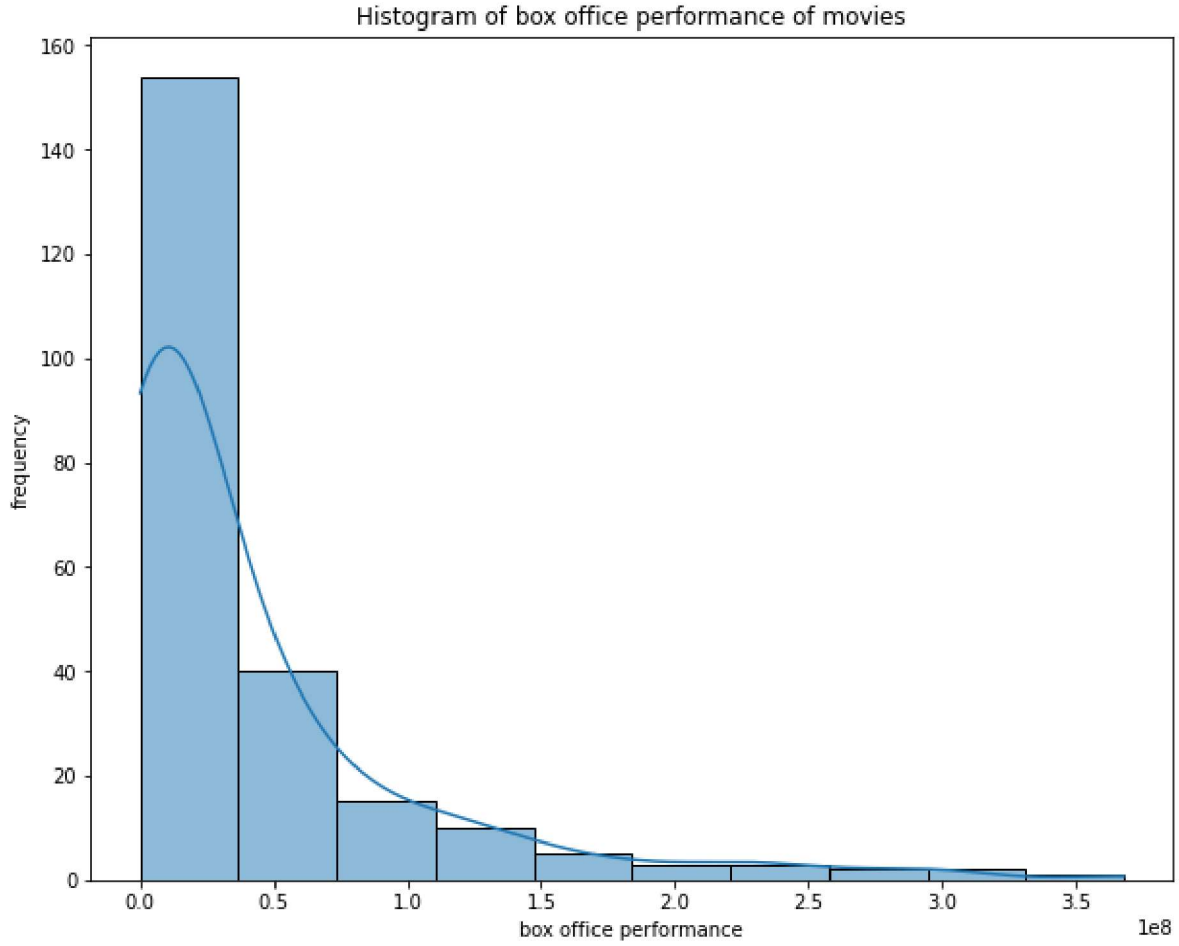
In [174]:
```
# checking summary statistics
df3.describe()
```

Out[174]:

	box_office
count	235
mean	41958400
std	62630156
min	363
25%	2302444
50%	15536310
75%	52649522
max	368000000

Since we have the mean and midian and the percentiles,we can make a histogram to help with our research.

In [175]:
```
# Plotting the histogram
# Setting up the figure and axes
fig, ax = plt.subplots(figsize = (10, 8))

# Plotting the box office performance histogram.
sns.histplot(data = df3, x = 'box_office', bins =  10, kde = True)
ax.set_xlabel('box office performance')
ax.set_ylabel('frequency')
ax.set_title('Histogram of box office performance of movies')

# Showing the plot
plt. show();
```



Since the median is lower than the mean,it means that majority of the data is in the left side meaning box_office does not have a lot of viewers generally even though some of the movies there are really good.

Bivariate Analysis

Now we will be using relationship between two columns in our dataframes and the two tables

In [176]:
```
#checking df1
df1.head()
```

Out[176]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000	664300000	2010
3	Inception	WB	292600000	535700000	2010
4	Shrek Forever After	P/DW	238700000	513900000	2010

In [177]:
```
#Summary of df1
summary(df1)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2007 entries, 0 to 3353
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           2007 non-null   object 
 1   studio          2007 non-null   object 
 2   domestic_gross  2007 non-null   float64
 3   foreign_gross   2002 non-null   float64
 4   year            2007 non-null   object 
dtypes: float64(2), object(3)
memory usage: 94.1+ KB
```

Out[177]:

```
((2007, 5),
 Index(['title', 'studio', 'domestic_gross', 'foreign_gross', 'year'], dtype
='object'),
 None)
```

we will be comparing `studio` to (`domestic_gross` and `foreign_gross`)

In [178]:
```
# Checking the data that compares studio and domestic gross for the top 20 stu
st1 = df1.groupby('studio')['domestic_gross'].sum().nlargest(20).reset_index()
st1
```

Out[178]:

	studio	domestic_gross
0	BV	18396529199
1	Uni.	12892038000
2	WB	12123600000
3	Fox	10924499997
4	Sony	8459479098
5	Par.	7580812699
6	LGF	3991851400
7	WB (NL)	3975099999
8	LG/S	1965199998
9	P/DW	1682900000
10	SGem	1516100000
11	Wein.	1471816698
12	Focus	1126800000
13	FoxS	982100000
14	Sum.	930071000
15	Rela.	892539000
16	TriS	828100000
17	STX	728000000
18	ORF	681899999
19	FD	353700000

In [179]:
```
# Checking the data that compares studio and foreign gross for the top 20 stua
st2 = df1.groupby('studio')['foreign_gross'].sum().nlargest(20).reset_index()
st2
```

Out[179]:

	studio	foreign_gross
0	BV	25793852199
1	Fox	20055866599
2	WB	18667902998
3	Uni.	16854767999
4	Sony	13945235998
5	Par.	11863384998
6	WB (NL)	6339000000
7	LGF	4475619300
8	P/DW	3393600000
9	LG/S	3353724000
10	WGUSA	2761447000
11	Wein.	2624085999
12	CL	1891000000
13	SGem	1624062000
14	FoxS	1492587700
15	Focus	1369968999
16	Sum.	1354900000
17	SPC	956805998
18	TriS	884955000
19	HC	867600000

In [180]:
```
#Statistics domestic_gross
st1.describe()
```

Out[180]:

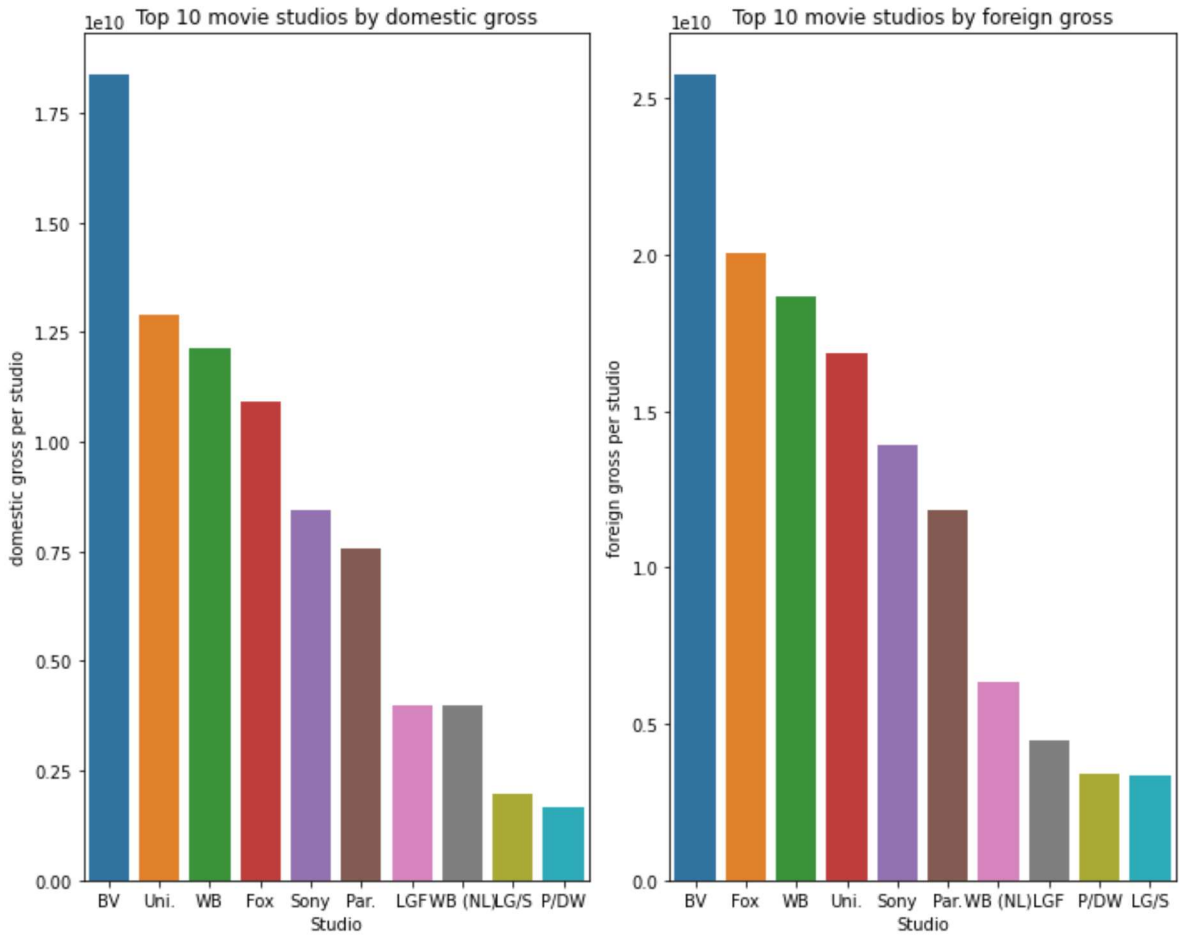
	domestic_gross
count	20
mean	4575156854
std	5281300461
min	353700000
25%	920688000
50%	1599500000
75%	7800479299
max	18396529199

In [181]:
```
#Statistics foreign_gross
st2.describe()
```

Out[181]:

	foreign_gross
count	20
mean	7028518339
std	7811722644
min	867600000
25%	1461933025
50%	3057585500
75%	12383847748
max	25793852199

In [182]:
```
# Plotting the bar graphs

# Subsetting the data
data1 = df1.groupby('studio')['domestic_gross'].sum().nlargest(10).reset_index
data2 = df1.groupby('studio')['foreign_gross'].sum().nlargest(10).reset_index(

# Setting up the figure and axes
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 8))


# Plotting the first graph
sns.barplot(x='studio', y='domestic_gross', data = data1, ax=ax1)
ax1.set_xlabel('Studio')
ax1.set_ylabel('domestic gross per studio')
ax1.set_title('Top 10 movie studios by domestic gross')

# Plotting the second graph
sns.barplot(x='studio', y='foreign_gross', data = data2, ax=ax2)
ax2.set_xlabel('Studio')
ax2.set_ylabel('foreign gross per studio')
ax2.set_title('Top 10 movie studios by foreign gross')

# Adjusting the layout to prevent overlap
plt.tight_layout()

# Showing the plot
plt.show();
```



From the analysis above,its clear that the movie industry generates a lot of revenue highest being at 25793852199 by foreign_gross only.This is a good sign for companies interested in filmmaking like microsoft since it should be sure of making profits if the correct modern trends set by the famous studios now are followed.

Now the second df:

In [183]:
```
#Checking data
df2.head()
```

Out[183]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_av
0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	34	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	29	2010-03-26	How to Train Your Dragon	
2	[12, 28, 878]	10138	en	Iron Man 2	29	2010-05-07	Iron Man 2	
3	[16, 35, 10751]	862	en	Toy Story	28	1995-11-22	Toy Story	
4	[28, 878, 12]	27205	en	Inception	28	2010-07-16	Inception	

In [184]:
```
#Summary of df2
summary (df2)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   genre_ids          26517 non-null  object 
 1   id                 26517 non-null  object 
 2   original_language  26517 non-null  object 
 3   original_title     26517 non-null  object 
 4   popularity         26517 non-null  float64
 5   release_date       26517 non-null  object 
 6   title              26517 non-null  object 
 7   vote_average       26517 non-null  float64
 8   vote_count         26517 non-null  int64  
dtypes: float64(2), int64(1), object(6)
memory usage: 1.8+ MB
```

Out[184]:
```
((26517, 9),
 Index(['genre_ids', 'id', 'original_language', 'original_title', 'popularit
y',
        'release_date', 'title', 'vote_average', 'vote_count'],
       dtype='object'),
 None)
```

In [185]:
```
#Checking the data that compares original_language and vote average for the to
ot1 = df2.groupby('original_language')['vote_average'].sum().nlargest(10).rese
ot1
```

Out[185]:

	original_language	vote_average
0	en	138662
1	fr	3131
2	es	2874
3	ja	1809
4	ru	1579
5	de	1457
6	zh	1113
7	hi	1039
8	it	769
9	ko	636

In [186]:
```
#Checking the data that compares original_language and vote average for the to
ot2 = df2.groupby('original_language')['vote_count'].sum().nlargest(10).reset_
ot2
```

Out[186]:

	original_language	vote_count
0	en	4874990
1	fr	75337
2	ja	54774
3	es	29396
4	it	17233
5	de	13900
6	ko	11017
7	sv	9726
8	da	8453
9	hi	7870

In [187]:
```
#Summary statistics of vote_average
ot1.describe()
```

Out[187]:

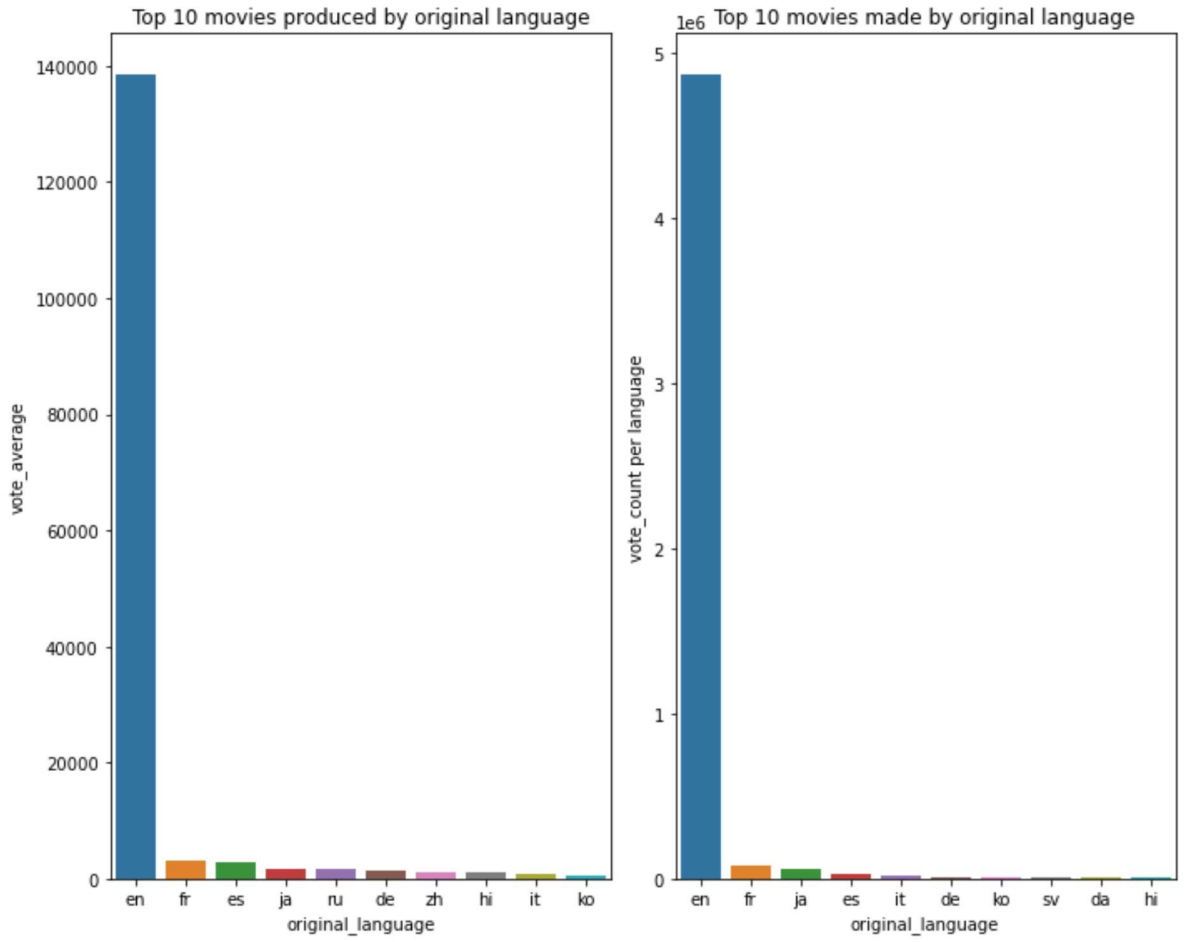
	vote_average
count	10
mean	15307
std	43351
min	636
25%	1057
50%	1518
75%	2608
max	138662

In [188]:
```
#Summary statistics of vote_count
ot2.describe()
```

Out[188]:

	vote_count
count	10
mean	510270
std	1533774
min	7870
25%	10049
50%	15566
75%	48430
max	4874990

In [189]:
```
# Plotting the bar graphs

# Subsetting the data
ot1 = df2.groupby('original_language')['vote_average'].sum().nlargest(10).rese
ot2 = df2.groupby('original_language')['vote_count'].sum().nlargest(10).reset_

# Setting up the figure and axes
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 8))


# Plotting the first graph
sns.barplot(x='original_language', y='vote_average', data = ot1, ax=ax1)
ax1.set_xlabel('original_language')
ax1.set_ylabel('vote_average')
ax1.set_title('Top 10 movies produced by original language')

# Plotting the second graph
sns.barplot(x='original_language', y='vote_count', data = ot2, ax=ax2)
ax2.set_xlabel('original_language')
ax2.set_ylabel('vote_count per language')
ax2.set_title('Top 10 movies made by original language')

# Adjusting the layout to prevent overlap
plt.tight_layout()

# Showing the plot
plt.show();
```



This clearly shows that movies made in English are generally more voted for by many people.Microsoft should ensure to make their first movies in English to have an easy start in this movie industry.

Now the third df:

In [190]:
```
#Check data
df3.head()
```

Out[190]:

	id	synopsis	rating	genre	director	writer	theater_date	c
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 2012	
6	10	Some cast and crew from NBC's highly acclaimed...	PG-13	Comedy	Jake Kasdan	Mike White	Jan 11, 2002	
7	13	Stewart Kane, an Irishman living in the Austra...	R	Drama	Ray Lawrence	Raymond Carver|Beatrix Christian	Apr 27, 2006	
15	22	Two-time Academy Award Winner Kevin Spacey giv...	R	Comedy|Drama|Mystery and Suspense	George Hickenlooper	Norman Snider	Dec 17, 2010	
18	25	From ancient Japan's most enduring tale, the e...	PG-13	Action and Adventure|Drama|Science Fiction and...	Carl Erik Rinsch	Chris Morgan|Hossein Amini	Dec 25, 2013	

In [191]:
```
#Summary of df3
summary(df3)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 235 entries, 1 to 1545
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   id            235 non-null    object
 1   synopsis      235 non-null    object
 2   rating        235 non-null    object
 3   genre         235 non-null    object
 4   director      235 non-null    object
 5   writer        235 non-null    object
 6   theater_date  235 non-null    object
 7   dvd_date      235 non-null    object
 8   currency      235 non-null    object
 9   box_office    235 non-null    int32 
 10  runtime       235 non-null    object
 11  studio        235 non-null    object
dtypes: int32(1), object(11)
memory usage: 22.9+ KB
```

Out[191]:
```
((235, 12),
 Index(['id', 'synopsis', 'rating', 'genre', 'director', 'writer',
        'theater_date', 'dvd_date', 'currency', 'box_office', 'runtime',
        'studio'],
       dtype='object'),
 None)
```

We will take `rating` to(box_office)

In [192]:
```
# Grouping the data
rt3 = df3.groupby('rating')['box_office'].sum().nlargest().reset_index()
rt3
```

Out[192]:

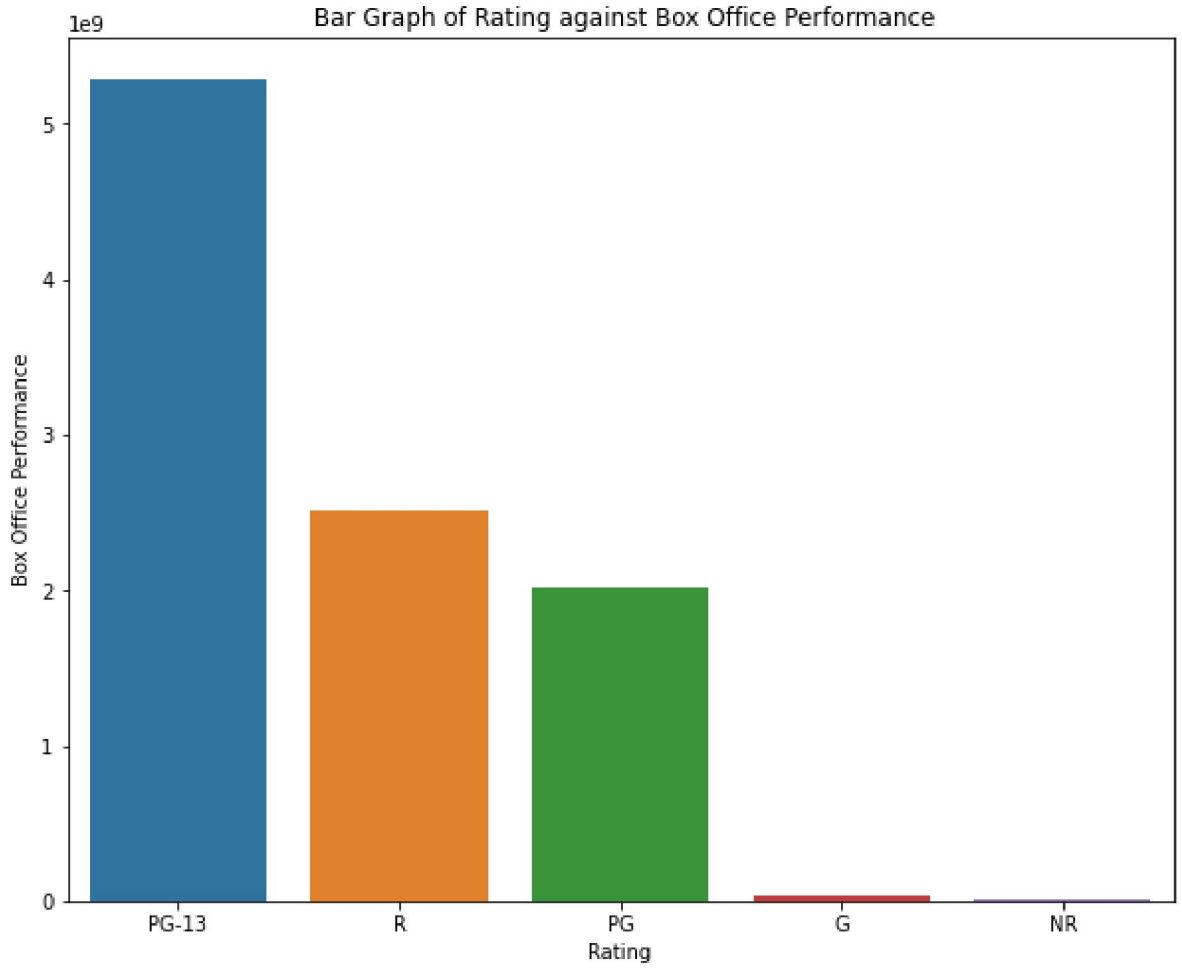
	rating	box_office
0	PG-13	5291716108
1	R	2514567999
2	PG	2009926506
3	G	37013942
4	NR	5739231

In [193]:
```
#Statistics of df3
rt3.describe()
```

Out[193]:

	box_office
count	5
mean	1971792757
std	2175240546
min	5739231
25%	37013942
50%	2009926506
75%	2514567999
max	5291716108

In [194]:
```
# Plotting the bar graph
# Setting up the figure and axes
fig, ax = plt.subplots(figsize = (10, 8))

# Plotting the graph
sns.barplot(x='rating', y='box_office', data = rt3)
ax.set_xlabel('Rating')
ax.set_ylabel('Box Office Performance')
ax.set_title(' Bar Graph of Rating against Box Office Performance')

# Showing the plot
plt.show();
```



We now see that the PG-13,R rated and PG movies had the best performances.Microsoft should make sure to consider the ratings of their films in order to find the best environment to produce the best movies.The bivariate analysis has given us good factors to consider while making a movie:
1)Studio types
2)Movie Ratings
3)Type of language of the movies
4)Target viewers (aim for foreigners)
This will help in making perfect films.

Multivariate analysis

Analysis of multiple columns

First data frame:A heat map is generally needed to create the correlation matrix between the different columns

In [195]:
```
#Check data
df1.head()
```

Out[195]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000	664300000	2010
3	Inception	WB	292600000	535700000	2010
4	Shrek Forever After	P/DW	238700000	513900000	2010

In [196]:
```
#summary of the data
summary(df1)
```

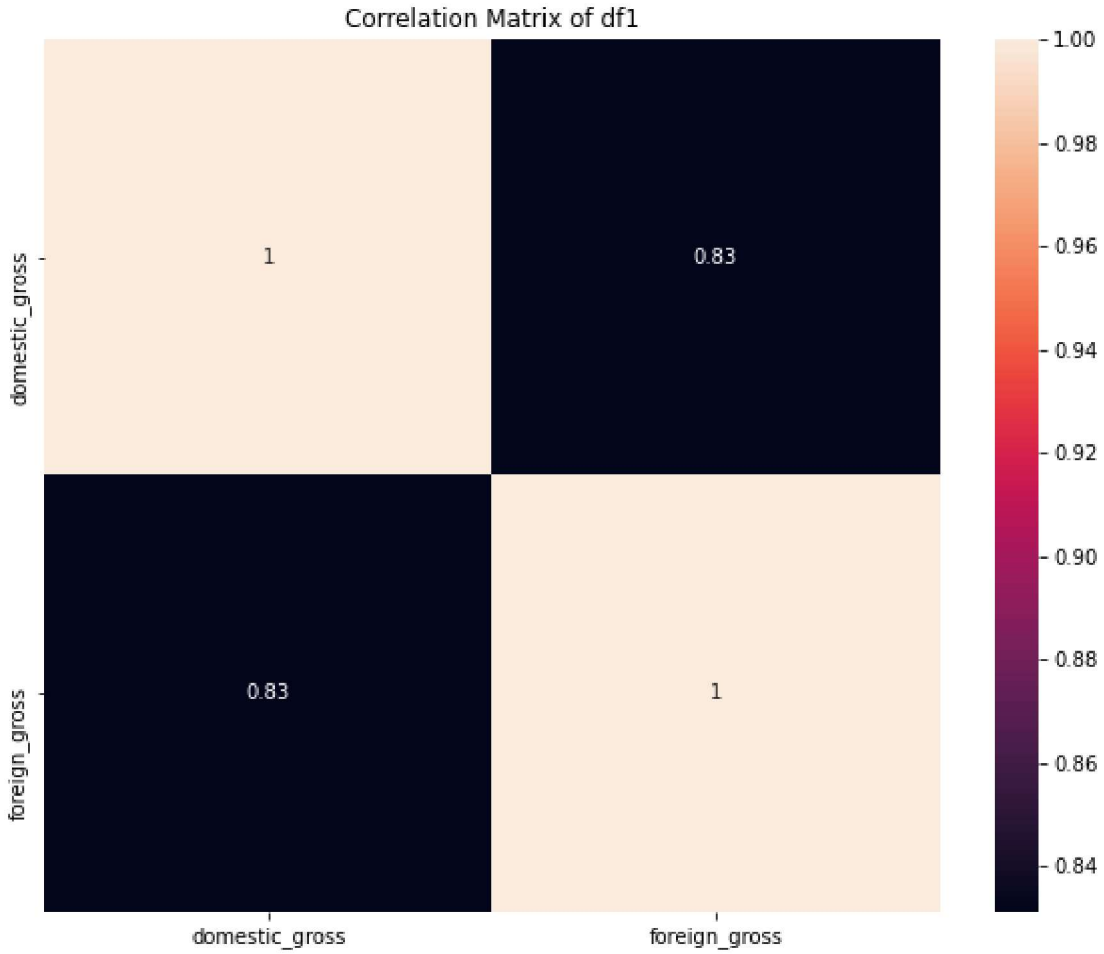
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2007 entries, 0 to 3353
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           2007 non-null   object 
 1   studio          2007 non-null   object 
 2   domestic_gross  2007 non-null   float64
 3   foreign_gross   2002 non-null   float64
 4   year            2007 non-null   object 
dtypes: float64(2), object(3)
memory usage: 94.1+ KB
```

Out[196]:

```
((2007, 5),
 Index(['title', 'studio', 'domestic_gross', 'foreign_gross', 'year'], dtype
='object'),
 None)
```

We need a heat map to create the correlation matrix of the numerical columns

In [197]:
```
# Plotting the correlation matrix

# Selecting the numerical variables for which you want to calculate the correl
numerical_vars = df1[['domestic_gross', 'foreign_gross']]

#Calculating the correlation matrix using the corr() method
correlation_matrix = numerical_vars.corr()

#Visualizing the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True)
plt.title('Correlation Matrix of df1')
plt.show();
```



The matrix shows both domestic and foreign_gross are nearly on the same level.This means that changes in trends affects both the domestic and foreign_gross almost simultaneously whether in a good way or ba way. This means Microsoft should not consider only the foreign_gross to earn them huge revenue but expect of the same results from your domestic_gross.

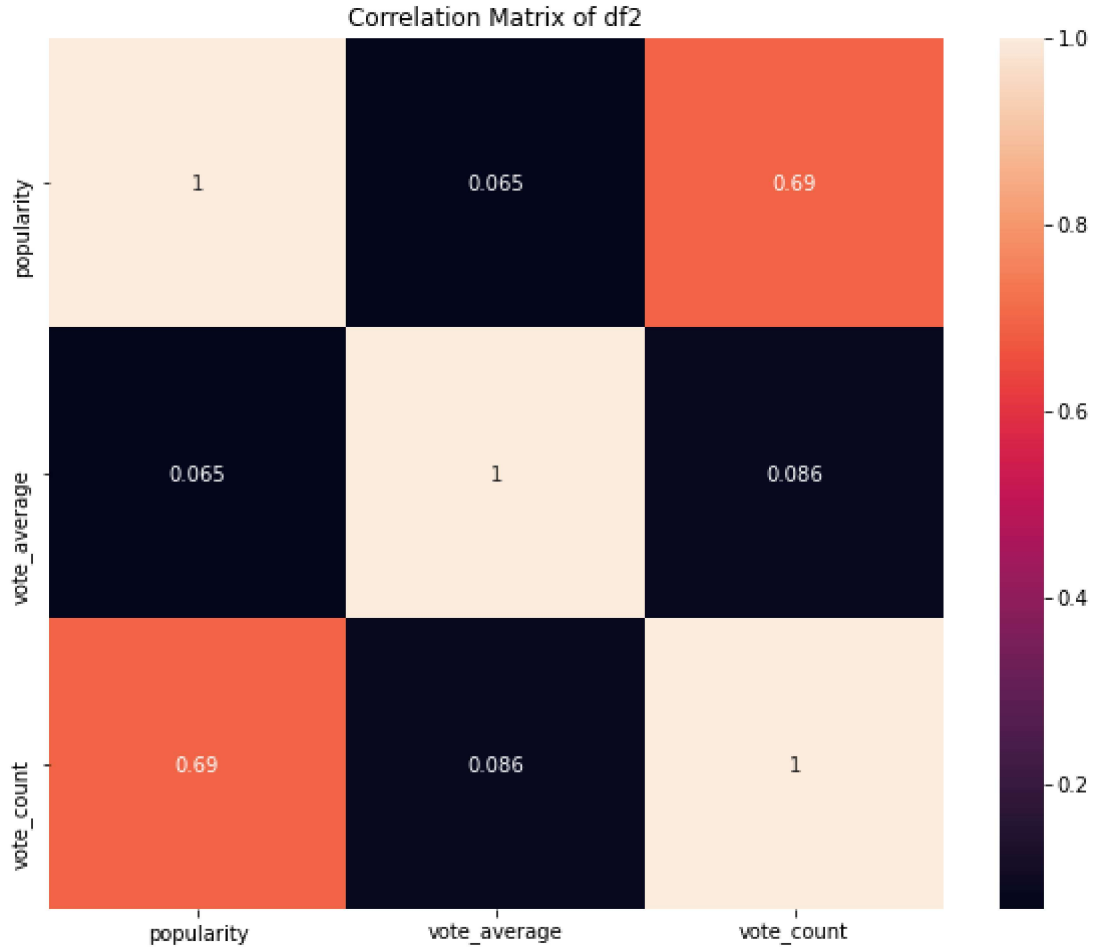
2nd data frame:

In [198]:
```
#Checking data
df2.head()
```

Out[198]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_av
0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	34	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	29	2010-03-26	How to Train Your Dragon	
2	[12, 28, 878]	10138	en	Iron Man 2	29	2010-05-07	Iron Man 2	
3	[16, 35, 10751]	862	en	Toy Story	28	1995-11-22	Toy Story	
4	[28, 878, 12]	27205	en	Inception	28	2010-07-16	Inception	

In [199]:
```
#  Plotting the correlation matrix
# Selecting the numerical variables for which you want to calculate the correl
numerical_vars = df2[['popularity', 'vote_average', 'vote_count']]

#Calculating the correlation matrix using the corr() method
correlation_matrix = numerical_vars.corr()

#Visualizing the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True)
plt.title('Correlation Matrix of df2')
plt.show();
```



The values of popularity and vote_count are almost on the same level meaning increase in number of viewers makes the company more popular hence Microsoft should think of ways to advertise their movies far and wide if they want to gain popularity to grow themselves as a company.

The multivariate analysis has given good examples of factors which correlate very well in the trends hence Microsoft should consider them keenly during their filmmaking business.Factors like `vote_count` and `Popularity` really a high correlation meaning equal same effects being brought by both factors.