Final Project Submission

Please fill out:

- Student name: Vivian Watiri Wainaina
- Student pace: part time
- Scheduled project review date/time:
- Instructor name:
- Blog post URL:

OBJECTIVE:

To help microsoft decide on what movies to produce in their new movie studio:

To do this I plan to answer a few quesions:

1. Which studio is making the best returns both domestically and internationally and which ones are not?

2. Which movie genres have the best ratings and what genre has the worst ratings?

A). "bom.movie_gross.csv.gz"

In [1]:
```
#Loading necessary dictionaries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
```

In [2]:
```
# using pd.read_csv to read the "bom.movie_gross.csv.gz" into the dmovie_df
dmovie_df = pd.read_csv(r"C:\Users\ADMIN\OneDrive\Desktop\Moringa-Pharse1-project\dsc-p

dmovie_df.head()
```

Out[2]:

title	studio	domestic_gross	foreign_gross	year
Toy Story 3	BV	415000000.0	652000000	2010
Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
Inception	WB	292600000.0	535700000	2010
Shrek Forever After	P/DW	238700000.0	513900000	2010

In [3]:
```
pd.set_option('float_format', '{:.2f}'.format)
```

Data cleaning and Familiarising myself with the data

Familiarising

In [4]:
```
dmovie_df.shape
```

Out[4]:
```
(3387, 4)
```

In [5]:
```
dmovie_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3387 entries, Toy Story 3 to An Actor Prepares
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   studio          3382 non-null   object 
 1   domestic_gross  3359 non-null   float64
 2   foreign_gross   2037 non-null   object 
 3   year            3387 non-null   int64  
dtypes: float64(1), int64(1), object(2)
memory usage: 132.3+ KB
```

In [6]:
```
# Changing the "foreign_gross" column to float type to make it easier to statistically
dmovie_df["foreign_gross"].replace(',', '', regex = True, inplace = True)
dmovie_df["foreign_gross"] = dmovie_df["foreign_gross"].astype(float)
```

In [7]:
```
dmovie_df[["domestic_gross","foreign_gross"]].describe()
```

Out[7]:

	domestic_gross	foreign_gross
count	3359.00	2037.00
mean	28745845.07	74872810.15
std	66982498.24	137410600.84
min	100.00	600.00
25%	120000.00	3700000.00
50%	1400000.00	18700000.00
75%	27900000.00	74900000.00
max	936700000.00	960500000.00

Data cleaning

Columns missing data:

- studio = 6,

- domestic_gross = 28,
- foreign_gross = 1350,
- year = 0.

Deleting the "foreign_gross" column or rows (missing the "foreign_gross" data) would mean losing alot of data that I could otherwise use to conduct my analysis.

That is why I will use the fillna() method and fill the missing data with the mean.

In [8]:
```
dmovie_df.index
```

Out[8]:
```
Index(['Toy Story 3', 'Alice in Wonderland (2010)',
       'Harry Potter and the Deathly Hallows Part 1', 'Inception',
       'Shrek Forever After', 'The Twilight Saga: Eclipse', 'Iron Man 2',
       'Tangled', 'Despicable Me', 'How to Train Your Dragon',
       ...
       'Let Yourself Go', 'Hannah (2018)', 'Souvenir',
       'Furious (Legend of Kolovrat)', 'Beauty and the Dogs', 'The Quake',
       'Edward II (2018 re-release)', 'El Pacto', 'The Swan',
       'An Actor Prepares'],
      dtype='object', name='title', length=3387)
```

In [9]:
```
# checking for duplicates
duplicates = dmovie_df[dmovie_df.duplicated()]
print(len(duplicates))
duplicates
```

```
11
```

Out[9]:

title	studio	domestic_gross	foreign_gross	year
There Be Dragons	Gold.	1100000.00	nan	2011
About Cherry	IFC	3000.00	nan	2012
Perfect Sense	IFC	3000.00	nan	2012
Laggies	A24	1100000.00	nan	2014
Son of Saul	SPC	1800000.00	nan	2015
The Wolfpack	Magn.	1300000.00	nan	2015
Where Hope Grows	RAtt.	1200000.00	nan	2015
The Priests	CJ	185000.00	nan	2015
The Student and Mr. Henri	Distrib.	800.00	nan	2016
Better Watch Out	WGUSA	20400.00	nan	2017
Maria by Callas	SPC	1300000.00	nan	2018

From observing the above cell output we can see that there are no duplicates in the data

In [10]:
```
#Checking for extraneous values
for col in dmovie_df.columns:
    print(col, '\n', dmovie_df[col].value_counts(normalize=True).head(), '\n\n')
```

```
studio
 IFC     0.05
Uni.    0.04
WB      0.04
Fox     0.04
Magn.   0.04
Name: studio, dtype: float64


domestic_gross
 1100000.00   0.01
1000000.00   0.01
1300000.00   0.01
1200000.00   0.01
1400000.00   0.01
Name: domestic_gross, dtype: float64


foreign_gross
 1200000.00   0.01
1100000.00   0.01
1900000.00   0.01
4200000.00   0.01
2500000.00   0.01
Name: foreign_gross, dtype: float64


year
 2015   0.13
2016   0.13
2012   0.12
2011   0.12
2014   0.12
Name: year, dtype: float64
```

- There seems to be no other hidden missing data

In [11]:
```
round(dmovie_df["foreign_gross"].mean(),1)
```

Out[11]:
```
74872810.2
```

In [12]:
```
# filling the the missing data
dmovie_df["foreign_gross"].fillna(74872810.2, inplace = True)
```

In [13]:
```
# Ensuring that I have filled all the missing values in dmovie_df["foreign_gross"]
dmovie_df["foreign_gross"].isna()
```

Out[13]:
```
title
Toy Story 3                                     False
Alice in Wonderland (2010)                      False
Harry Potter and the Deathly Hallows Part 1     False
Inception                                       False
Shrek Forever After                             False
                                                ...
The Quake                                       False
Edward II (2018 re-release)                     False
El Pacto                                        False
The Swan                                        False
An Actor Prepares                               False
Name: foreign_gross, Length: 3387, dtype: bool
```

Since the "studio" and "domestic_gross" missing data is not really alot I will be deleting the rows that this data is missing from.

In [14]:
```
dmovie_df.dropna(axis = 0, subset =["domestic_gross","studio"], inplace = True)
```

In [15]:
```
# checking if there is still any missing data
dmovie_df.isna()
```

Out[15]:

title	studio	domestic_gross	foreign_gross	year
Toy Story 3	False	False	False	False
Alice in Wonderland (2010)	False	False	False	False
Harry Potter and the Deathly Hallows Part 1	False	False	False	False
Inception	False	False	False	False
Shrek Forever After	False	False	False	False
...	...	...	...	...
The Quake	False	False	False	False
Edward II (2018 re-release)	False	False	False	False
El Pacto	False	False	False	False
The Swan	False	False	False	False
An Actor Prepares	False	False	False	False

3356 rows × 4 columns

Data Analysis

Question

Which studio earned the most?

First I will be adding the "domestic_gross" and "foreign_gross" columns to allow me to see the total amount each movie grossed "total_gross".

In [16]:
```
# "Total_gross"
dmovie_df["total_gross"] = dmovie_df["domestic_gross"] + dmovie_df["foreign_gross"]
dmovie_df.tail()
```

Out[16]:

title	studio	domestic_gross	foreign_gross	year	total_gross
The Quake	Magn.	6200.00	74872810.20	2018	74879010.20
Edward II (2018 re-release)	FM	4800.00	74872810.20	2018	74877610.20
El Pacto	Sony	2500.00	74872810.20	2018	74875310.20

title	studio	domestic_gross	foreign_gross	year	total_gross
The Swan	Synergetic	2400.00	74872810.20	2018	74875210.20
An Actor Prepares	Grav.	1700.00	74872810.20	2018	74874510.20

In [17]:
```
#This shows us the highest to the lowest earning studio during the whole period
earnings = dmovie_df.groupby("studio")["total_gross"].sum().sort_values(ascending = Fal
earnings.head(10)
```

Out[17]:
```
studio
BV        44362629519.50
WB        31584677100.00
Fox       31155112216.40
Uni.      29981782622.00
Sony      22704410336.80
Par.      20073365368.40
WB (NL)   10409572809.20
LGF        9717674853.00
IFC        7848927897.60
Magn.      6633386025.20
Name: total_gross, dtype: float64
```

In [18]:
```
# this allows us how much each studio earned each year
yearly = dmovie_df.groupby(["year", "studio"])["total_gross"].sum()
yearly
```

Out[18]:
```
year  studio
2010  3D          16000000.00
      ATO          1040700.00
      Abr.        76031210.20
      Anch.       19714100.00
      App.         4700000.00
                     ...
2018  Vita.       75021810.20
      WB        5665245620.40
      WB (NL)   1055300000.00
      WGUSA     1096201340.80
      Yash         4230000.00
Name: total_gross, Length: 737, dtype: float64
```

In [19]:
```
# Looking for the movies with the most returns
dmovie_df.groupby("title",axis = 0)["total_gross"].max().sort_values(ascending = False)
```

Out[19]:
```
title
Marvel's The Avengers                            1518900000.00
Avengers: Age of Ultron                          1405400000.00
Black Panther                                    1347000000.00
Harry Potter and the Deathly Hallows Part 2      1341500000.00
Star Wars: The Last Jedi                         1332600000.00
                                                       ...
I'm Glad My Mother is Alive                        21900.00
The Thorn in the Heart                             17900.00
Cirkus Columbia                                    13000.00
Aurora                                             10800.00
To Die Like a Man                                   4900.00
Name: total_gross, Length: 3355, dtype: float64
```

In [20]:
```
# Analysing the studio with the most profits
bv = dmovie_df[dmovie_df["studio"] == "BV"]
```

```
bv.groupby("title",axis = 0)["total_gross"].max().sort_values(ascending = False)
```

Out[20]:
```
title
Marvel's The Avengers      1518900000.00
Avengers: Age of Ultron    1405400000.00
Black Panther              1347000000.00
Star Wars: The Last Jedi   1332600000.00
Frozen                     1276400000.00
                                  ...
Monkey Kingdom               17110999.00
Strange Magic                13600000.00
Queen of Katwe               10400000.00
The Fifth Estate              8600000.00
Waking Sleeping Beauty          84900.00
Name: total_gross, Length: 106, dtype: float64
```

Interesting facts

In [21]:
```
dmovie_df["studio"].mode()
```

Out[21]:
```
0    IFC
dtype: object
```

In [22]:
```
 dmovie_df["studio"].value_counts().head(10)
```

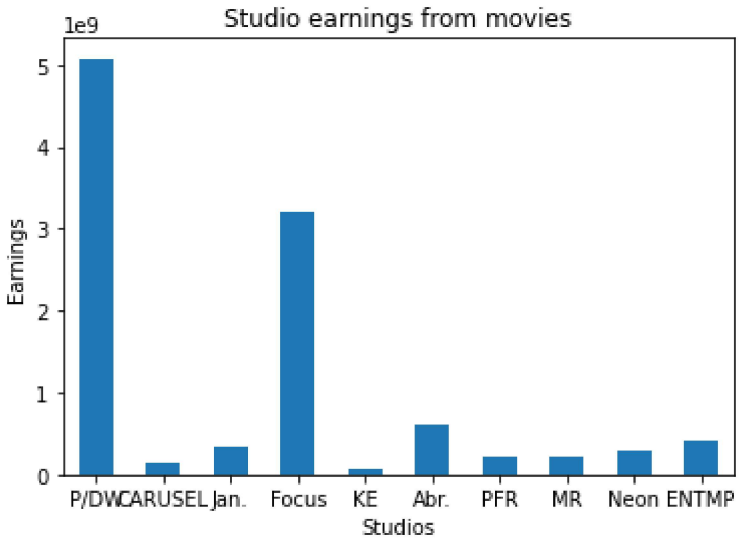
Out[22]:
```
IFC      166
Uni.     147
WB       140
Magn.    136
Fox      136
SPC      123
Sony     109
BV       106
LGF      102
Par.     101
Name: studio, dtype: int64
```

Visualization

In [23]:
```
# bar graph
earnings_sample = earnings.sample(n = 10)
ax = earnings_sample.plot.bar(x = "studio", y = "total_gross", rot = 0)

# labels and title
ax.set_xlabel("Studios")
ax.set_ylabel("Earnings")
ax.set_title("Studio earnings from movies")

plt.show()
```



The above visualization takes into account a small sample of the total earnings of each individual studio.

- Notice that each time that the code is run the graph takes a different sample of the data.

Conclusion

From the data I am able to learn that the BV studio earned the most revenue during this period and also that it had the highest grossing movie too. But I have also learnt that the IFC studio produce atleast 60 more movies than BV studio but earned alot less.

The mode finding is interesting.

It means that the studio that produced the most movies was not the one that had the highest earnings over the time period we are considering.

The value_count() function also revealed some interesting information since some of the highest earning studios did not even produce the most movies

B). "im.db" Data Set

In [24]:
```
#Loading the necessary environment and connecting the data set
import sqlite3
conn = sqlite3.connect(r"C:\Users\ADMIN\Downloads\dsc-phase-1-project-v2-4-master\dsc-p
```

In [25]:
```
#finding out the available tables
imovies_df = pd.read_sql("""SELECT name FROM sqlite_master WHERE type = 'table';""", co
imovies_df
```

Out[25]:

	name
0	movie_basics
1	directors
2	known_for

	name
3	movie_akas
4	movie_ratings
5	persons
6	principals
7	writers

In [26]:
```
# loading data from the "movie_basics" table
pd.read_sql("""SELECT * FROM movie_basics""",conn)
```

Out[26]:

	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
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.00	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	nan	Documentary
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	nan	Comedy
146142	tt9916730	6 Gunn	6 Gunn	2017	116.00	None
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	nan	Documentary

146144 rows × 6 columns

In [27]:
```
# loading data from the "movie_ratings" table
pd.read_sql("""SELECT * FROM movie_ratings""",conn)
```

Out[27]:

	movie_id	averagerating	numvotes
0	tt10356526	8.30	31
1	tt10384606	8.90	559
2	tt1042974	6.40	20

	movie_id	averagerating	numvotes
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

In [28]:
```
# filtering the ratings to see their distribution
pd.read_sql("""
SELECT *
FROM movie_ratings
WHERE numvotes <= 500;
""", conn)
```

Out[28]:

	movie_id	averagerating	numvotes
0	tt10356526	8.30	31
1	tt1042974	6.40	20
2	tt1060240	6.50	21
3	tt1069246	6.20	326
4	tt1156528	7.20	265
...	...	...	...
59971	tt9805820	8.10	25
59972	tt9844256	7.50	24
59973	tt9851050	4.70	14
59974	tt9886934	7.00	5
59975	tt9894098	6.30	128

59976 rows × 3 columns

In [29]:
```
# checking for missing data
pd.read_sql("""
SELECT *
FROM movie_ratings
 WHERE movie_id IS NULL;
 """, conn)
```

Out[29]:

movie_id	averagerating	numvotes

The "movie_ratings" data seems not to have missing data

Data Analysis

Question:

Which Genre has the best ratings?

In [30]:
```
# The genres with the best ratings
q = """
SELECT genres, averagerating, numvotes
FROM movie_basics
JOIN movie_ratings
    USING (movie_id)
WHERE numvotes >1000000;
"""
pd.read_sql_query(q, conn)
```

Out[30]:

	genres	averagerating	numvotes
0	Mystery,Thriller	8.10	1005960
1	Action,Thriller	8.40	1387769
2	Adventure,Drama,Sci-Fi	8.60	1299334
3	Drama,Western	8.40	1211405
4	Biography,Crime,Drama	8.20	1035358
5	Action,Adventure,Sci-Fi	8.10	1183655
6	Action,Adventure,Sci-Fi	8.80	1841066

In [31]:
```
# genre with the highest ratings
q = """
SELECT genres, MAX(averagerating), MAX(numvotes)
FROM movie_basics
JOIN movie_ratings
    USING (movie_id);
"""
pd.read_sql_query(q, conn)
```

Out[31]:

	genres	MAX(averagerating)	MAX(numvotes)
0	Action,Adventure,Sci-Fi	10.00	1841066

Analysing the combined data to see which genre has the best ratings

Also looking at the average number of votes:

- To ensure that the ratings are not just good with a really small sample of people.

Question:

Which genre has the least ratings and votes?

In [32]:
```
# movies genres with the least ratings
q = """
SELECT genres, averagerating, numvotes
FROM movie_basics
JOIN movie_ratings
    USING (movie_id)
WHERE numvotes <= 5
LIMIT 10;
"""
pd.read_sql_query(q, conn)
```

Out[32]:

	genres	averagerating	numvotes
0	Documentary	8.00	5
1	Documentary,Drama	8.20	5
2	Comedy,Music	7.80	5
3	Drama,Mystery,Thriller	1.40	5
4	Documentary	7.80	5
5	Drama	6.20	5
6	Documentary	8.80	5
7	Drama	7.00	5
8	Drama,Horror,Mystery	5.20	5
9	Documentary,Drama	8.60	5

In [33]:
```
q = """
SELECT genres, MIN(averagerating), Min(numvotes)
FROM movie_basics
JOIN movie_ratings
    USING (movie_id);
"""
pd.read_sql_query(q, conn)
```

Out[33]:

	genres	MIN(averagerating)	Min(numvotes)
0	Action,Drama	1.00	5

Visualize the above

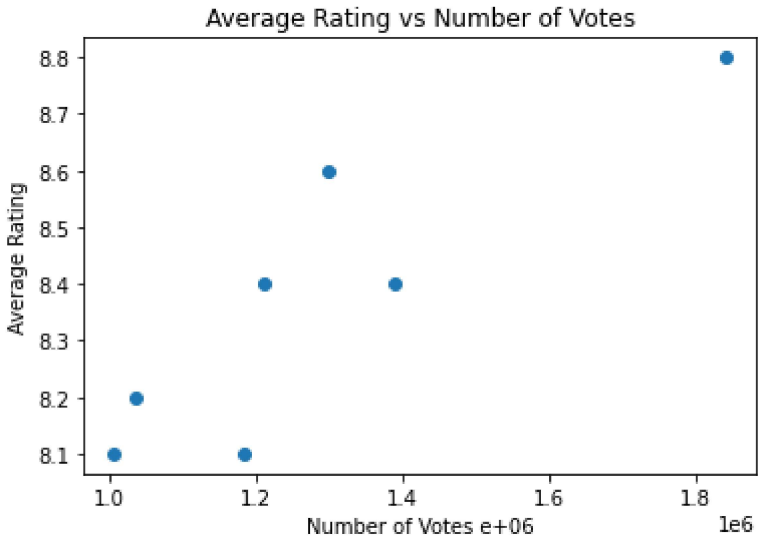
In [34]:
```
# run the query and store the results in a pandas dataframe
q = """
SELECT genres, averagerating, numvotes
FROM movie_basics
JOIN movie_ratings
    USING (movie_id)
```

```
WHERE numvotes >1000000;
"""
df = pd.read_sql_query(q, conn)

# create a scatter plot of averagerating vs numvotes
plt.scatter(df['numvotes'], df['averagerating'])

# add labels and title
plt.xlabel('Number of Votes e+06')
# This data here is over a million
plt.ylabel('Average Rating')
plt.title('Average Rating vs Number of Votes')

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



In [35]:

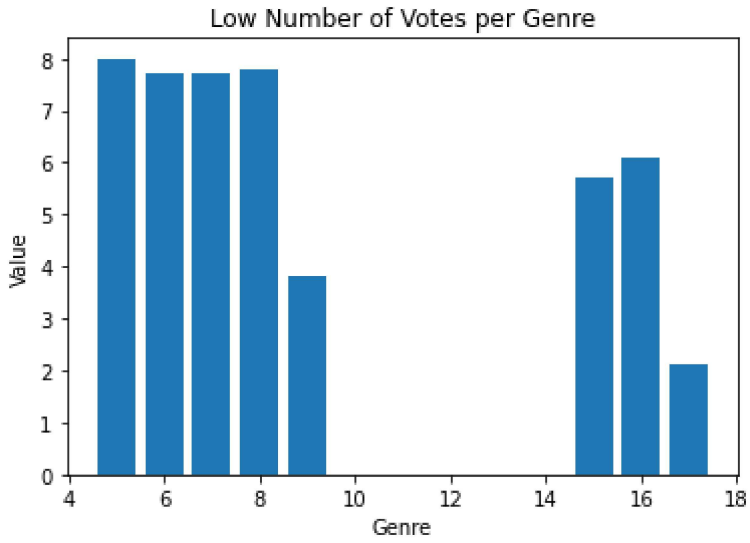
```
# run the query and store the results in a pandas dataframe

# run the query and store the results in a pandas dataframe
q = """
SELECT genres, averagerating, numvotes
FROM movie_basics
JOIN movie_ratings
    USING (movie_id)
WHERE numvotes < 20
LIMIT 10
"""
df = pd.read_sql_query(q, conn)

# create a bar chart of average rating and number of votes for each genre
plt.bar(df['numvotes'], df['averagerating'])

# add labels and title
plt.xlabel('Genre')
plt.ylabel('Value')
plt.title('Low Number of Votes per Genre')

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



Findings

1. The genre with the highest ratings and also with the highest number of votes is: ##### Action,Adventure,Sci-Fi

1. The genre with the lowest ratings is: ##### Action,Drama

Conclusion

- From the im.db data we learn that "Action,Adventure, & Sci-Fi" are the genres with the one of highest ratings.
    - This genres are some of the most well liked since the number of people voting is one of the highest.
    - I assume that a movie that has alot of people voting for it is a movie that has alot of people watching and therefore has good returns
- From the im.db data we learn that "Action & Drama" seem to be the ones with the least number of votes and the lowest ratings.
    - But we notice that the genre "Action" This shows that the movie genre is not the only determinant of success but rather just one of the factors.

DIsclaimer

Remember that I had to fill atleast 40% of the "foreign_gross" data with the mean. So this defintely has introduced a bias to my data.