Final Project Submission

Please fill out:

- Student name:Lee Kimaita
- Student pace: Part time
- Scheduled project review date/time:
- Instructor name: Noah Kandie
- Blog post URL: https://github.com/LKimaita/dsc-phase-1-project-v2-4/upload/master (https://github.com/LKimaita/dsc-phase-1-project-v2-4/upload/master)

Project Overview

The film industry is among a multi-billion dollar venture by numerous companies. Microsoft company has deemed it essential to ensure that they derive meaningful data from the available data from currently available video content in a bid to create a new studio. This analysis has put into use exploratory data to generate insights for the business stakeholders. Data has been cleaned and analysed.

Business Problem

Microsoft company seeks to create a competitive advantage in the film industry by establishing an ultra-modern, state of the art and futuristic movie studio. It seeks to be the giant and most preferred movie studio of choice by entertainment seekers with minimal competition from other key film players. Thus, we look at the various best film genres in the industry to help draw insights on the most profitable and marketable venture.

Data Understanding

Microsoft does not currently have any data relating to the multi-billion dollar industry. The available data from the industry has the best film genres that the company can venture into. The dataset available includes; genres, budgets and grossings (domestic and worldwide).

Importing Packages & Libraries

In [104]:
```
# Libraries to use

import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sqlite3
import warnings
warnings.filterwarnings("ignore")
```

Data from the budget

In [105]:
```
# importing movie budget data
tn_movies = pd.read_csv("tn.movie_budgets.csv" , index_col = 0)
tn_movies
```

Out[105]:

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

5782 rows × 5 columns

Cleaning budget data

In this section, we look for either duplicates or missing values. since data is available, we can assume that there is no missing data and choose to keep the first row or eliminate duplicates.

In [106]:
```
# identifying any missing data
tn_movies .isna().sum()
```

Out[106]:
```
release_date         0
movie                0
production_budget    0
domestic_gross       0
worldwide_gross      0
dtype: int64
```

In [107]:
```
# identify any duplicates

tn_movies .duplicated().sum()
```

Out[107]:
```
0
```

In [108]:
```
tn_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5782 entries, 1 to 82
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   release_date       5782 non-null   object
 1   movie              5782 non-null   object
 2   production_budget  5782 non-null   object
 3   domestic_gross     5782 non-null   object
 4   worldwide_gross    5782 non-null   object
dtypes: object(5)
memory usage: 271.0+ KB
```

In [109]:
```
# choosing to eliminate signage ($ and ,) by converting to float

tn_movies ['production_budget'] = tn_movies ['production_budget'].str.rep
tn_movies['domestic_gross']= tn_movies['domestic_gross'].str.replace('$',
tn_movies['worldwide_gross']= tn_movies['worldwide_gross'].str.replace('$
tn_movies .head()
```

Out[109]:

id	release_date	movie	production_budget	domestic_gross	worldwide_gross
1	Dec 18, 2009	Avatar	425000000.0	760507625.0	2.776345e+09
2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09
3	Jun 7, 2019	Dark Phoenix	350000000.0	42762350.0	1.497624e+08
4	May 1, 2015	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09
5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09

In [110]:

```
#dropping column
tn_movies.drop(["release_date"], axis = 1 , inplace = True)
```

In [111]:

```
# creating return investment column
tn_movies ["return_investment"]= ((tn_movies['domestic_gross'] + tn_movie
```

In [112]:

```
# sorting data and removing outliers
tn_movies_sorted = tn_movies.sort_values(by = 'return_investment', ascendi
tn_movies_sorted
```

Out[112]:

id	movie	production_budget	domestic_gross	worldwide_gross	return_investment
93	Paranormal Activity	450000.0	107918810.0	194183034.0	670.34
7	The Blair Witch Project	600000.0	140539099.0	248300000.0	647.07
80	The Gallows	100000.0	22764410.0	41656474.0	643.21
74	El Mariachi	7000.0	2040920.0	2041928.0	582.26
14	Mad Max	200000.0	8750000.0	99750000.0	541.50
10	Super Size Me	65000.0	11529368.0	22233808.0	518.43
47	Bambi	858000.0	102797000.0	268000000.0	431.16
16	The Brothers McMullen	50000.0	10426506.0	10426506.0	416.06
66	The Texas Chainsaw Massacre	140000.0	26572439.0	26572439.0	378.61
77	Night of the Living Dead	114000.0	12087064.0	30087064.0	368.95
37	Halloween	325000.0	47000000.0	70000000.0	359.00
11	Rocky	1000000.0	117235147.0	225000000.0	341.24
82	My Date With Drew	1100.0	181041.0	181041.0	328.17
73	American Graffiti	777000.0	115000000.0	140000000.0	327.19
43	Clerks	27000.0	3073428.0	3894240.0	257.06
18	Snow White and the Seven Dwarfs	1488000.0	184925486.0	184925486.0	247.56
58	Billy Jack	800000.0	98000000.0	98000000.0	244.00
47	In the Company of Men	25000.0	2883661.0	2883661.0	229.69
8	Napoleon Dynamite	400000.0	44540956.0	46122713.0	225.66

Bom movies

In [113]:
```
bom_movie = pd.read_csv('ZippedData/bom.movie_gross.csv.gz')
bom_movie
```

Out[113]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

3387 rows × 5 columns

Data cleaning for the bom movies gross data shall also encompass identifying duplicates and missing values.

In [114]:
```
#importing bom gross data
bom_movies = pd.read_csv("bom.movie_gross.csv")
```

In [115]:
```
bom_movies.head()
```

Out[115]:

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

In [116]:
```
bom_movies.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
```

missing values

In [117]:
```
bom_movies.isna().sum()
```

Out[117]:
```
title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64
```

In [118]:
```
#identifying  duplicates
bom_movies.duplicated().sum()
```

Out[118]:
```
0
```

In [119]:
```
# dropping foreign gross column
bom_movies.drop(["foreign_gross"], axis =1 , inplace = True)
```

In [120]:
```
bom_movies
```

Out[120]:

	title	studio	domestic_gross	year
0	Toy Story 3	BV	415000000.0	2010
1	Alice in Wonderland (2010)	BV	334200000.0	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	2010
3	Inception	WB	292600000.0	2010
4	Shrek Forever After	P/DW	238700000.0	2010
...	...	...	...	...
3382	The Quake	Magn.	6200.0	2018
3383	Edward II (2018 re-release)	FM	4800.0	2018
3384	El Pacto	Sony	2500.0	2018
3385	The Swan	Synergetic	2400.0	2018
3386	An Actor Prepares	Grav.	1700.0	2018

3387 rows × 4 columns

In [121]:
```
# drop the rows that have missing studio values
bom_movies.dropna(subset =["studio"], axis = 0 , inplace = True)
```

In [122]:
```
# check presence of missing values
bom_movies.isna().sum()
```

Out[122]:
```
title              0
studio             0
domestic_gross    26
year               0
dtype: int64
```

In [123]:
```
# eliminate rows that miss the domestic gross values
bom_movies.dropna(subset =["domestic_gross"], axis = 0 , inplace = True)
```

In [124]:
```
bom_movies.isna().sum()
```

Out[124]:
```
title             0
studio            0
domestic_gross    0
year              0
dtype: int64
```

In [125]:
```
# changing the date format
bom_movies["year"] = pd.to_datetime(bom_movies["year"])
```

In [126]:
```
bom_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3356 entries, 0 to 3386
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   title           3356 non-null   object        
 1   studio          3356 non-null   object        
 2   domestic_gross  3356 non-null   float64       
 3   year            3356 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 131.1+ KB
```

IMDB

We first define the connection

In [127]:
```
#defining conn
conn = sqlite3.connect('im.db')
```

In [128]:
```
imdb_data2 = pd.read_sql(""" select * from movie_basics""" , conn)
imdb_data2
```

Out[128]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	g
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,[
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,[
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	[
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,[
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fa
...	...	...	...	...	...	
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	[
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Docum
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Co
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Docum

146144 rows × 6 columns

In [129]:
```
imdb_data2 = pd.read_sql(""" select * from movie_basics""" , conn)
imdb_data2
```

Out[129]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	g
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,[
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,[
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	[
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,[
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fa
...	...	...	...	...	...	
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	[
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Docum
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Cc
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Docum

146144 rows × 6 columns

In [130]:

```
# Joining Tables
imdb_data =pd.read_sql(""" SELECT primary_title, start_year, genres, avera
FROM movie_basics AS MB
JOIN movie_ratings AS MR
ON MB.movie_id = MR.movie_id
WHERE numvotes > 1000000 AND averagerating BETWEEN 6.8 AND 9.2
ORDER BY averagerating DESC
LIMIT 50; """, conn)

imdb_data.head()
```

Out[130]:

	primary_title	start_year	genres	averagerating	numvotes
0	Inception	2010	Action,Adventure,Sci-Fi	8.8	1841066
1	Interstellar	2014	Adventure,Drama,Sci-Fi	8.6	1299334
2	The Dark Knight Rises	2012	Action,Thriller	8.4	1387769
3	Django Unchained	2012	Drama,Western	8.4	1211405
4	The Wolf of Wall Street	2013	Biography,Crime,Drama	8.2	1035358

In [131]:

```
grouped = imdb_data.groupby('genres')
grouped.get_group('Action,Adventure,Sci-Fi')
```

Out[131]:

	primary_title	start_year	genres	averagerating	numvotes
0	Inception	2010	Action,Adventure,Sci-Fi	8.8	1841066
6	The Avengers	2012	Action,Adventure,Sci-Fi	8.1	1183655

In [132]:

```
genres_mean_sorted = pd.DataFrame(imdb_data.groupby("genres")["numvotes"].
genres_mean_sorted
```

Out[132]:

genres	numvotes
Action,Adventure,Sci-Fi	1512360.5
Action,Thriller	1387769.0
Adventure,Drama,Sci-Fi	1299334.0
Drama,Western	1211405.0
Biography,Crime,Drama	1035358.0
Mystery,Thriller	1005960.0

In [133]:

```
genres_mean_sorted = pd.DataFrame(imdb_data.groupby("genres")["numvotes"].
genres_mean_sorted
```

Out[133]:

genres	numvotes
Action,Adventure,Sci-Fi	1512360.5
Action,Thriller	1387769.0
Adventure,Drama,Sci-Fi	1299334.0
Drama,Western	1211405.0
Biography,Crime,Drama	1035358.0
Mystery,Thriller	1005960.0

ANALYSIS

Studio and Domestic Gross

From the data cleaning undertaken from the given dataset, we now undertake to data analysis. From the analysis, Microsoft can determine its closest competitors in their newly established film studio. This is because we can identify the studios producing better films in the industry. Later, we get to plot our dataset using histograms that show the top performing studios based on their average domestic gross.

In [134]:

```
# grouping data by studio and domestic gross
bom_movies_grouped = bom_movies.groupby('studio')["domestic_gross"].mean()
```

In [135]:
```
# df dataframe
movies_grouped = pd.DataFrame(bom_movies_grouped)
movies_grouped
```

Out[135]:

	domestic_gross
studio	
3D	6.100000e+06
A23	8.210000e+04
A24	6.616208e+06
ADC	1.241000e+05
AF	3.571500e+05
...	...
XL	2.290000e+05
YFG	1.100000e+06
Yash	2.433185e+06
Zee	1.100000e+06
Zeit.	3.539688e+05

255 rows × 1 columns

In [136]:
```
# sorting grouped data
Grouped_movies = movies_grouped.sort_values(by = ["domestic_gross"], ascen
```

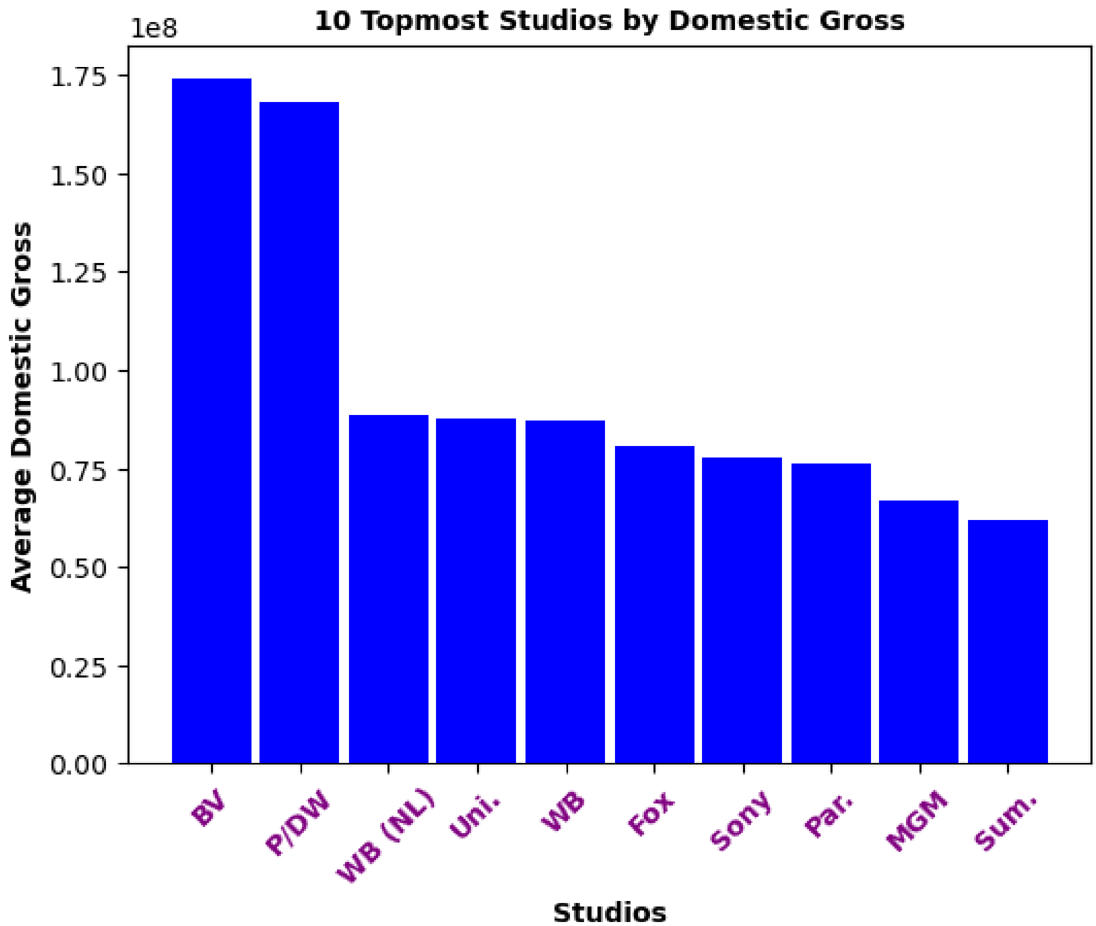
In [137]:
```
topmost_studios = Grouped_movies.head(10)
topmost_studios
```

Out[137]:

	domestic_gross
studio	
BV	1.737644e+08
P/DW	1.682900e+08
WB (NL)	8.879333e+07
Uni.	8.777138e+07
WB	8.691461e+07
Fox	8.051103e+07
Sony	7.761177e+07
Par.	7.609773e+07
MGM	6.666667e+07
Sum.	6.212473e+07

In [138]:

```
# plotting histogram
plt.bar(topmost_studios["domestic_gross"].index, topmost_studios["domestic
plt.xticks(rotation = 45 , fontsize = 10 , fontweight = "bold" , color = '
plt.xlabel("Studios" , fontsize = 10 , fontweight = "bold")
plt.ylabel("Average Domestic Gross", fontsize = 10 , fontweight = "bold")
plt.title("10 Topmost Studios by Domestic Gross", fontsize = 10 , fontweig
plt.gcf().set_size_inches = "9 ,8"
plt.show()
```



EXPLANATION

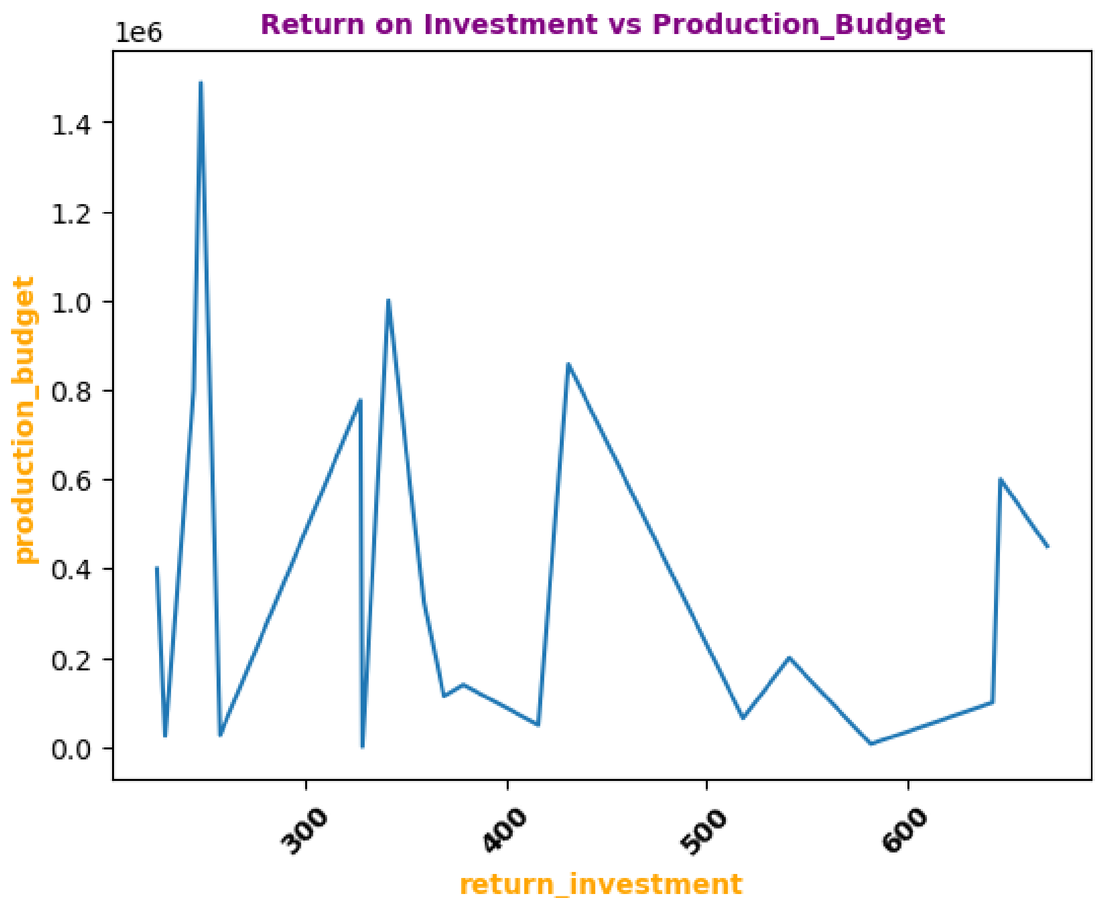
From the analysis, the BV studio ranks the highest in average domestic grosswhich is at 1.73M while the least grossing studio is sum. Most studios averaged between 0.75-0.85m. Given Microsoft's reputation as a big market play, its main competition will be from BV and P/DW which are the two top grossing in terms of their domestic averages. It is therefore imperative that the company undertakes to benchmark from both companies either in production, marketting choice of movie stars among other film aspects.

Return Investments and Production Budget

Every company wishes to have returns on their investments at all costs. We get to analyse the budget data. As a stakeholder the preferred variables would be comparing the costs incurred in the production process with the return on investment to ascertain profit margin levels. Further, we wish to identify whether there exists some linear relationship between the ROI and the production budget

In [139]:

```
#plotting bar graph
x = tn_movies_sorted['return_investment']
y = tn_movies_sorted['production_budget']
plt.plot(x,y)
plt.xticks(rotation = 45 , fontsize = 10 , fontweight = "bold" )
plt.xlabel("return_investment" , fontsize = 10 , fontweight = "bold" ,  co
plt.ylabel("production_budget", fontsize = 10 , fontweight = "bold" ,  co
plt.title("Return on Investment vs Production_Budget", fontsize = 10 , fo
plt.gcf().set_size_inches = "9 ,8"
plt.show()
```



Explanatory remarks

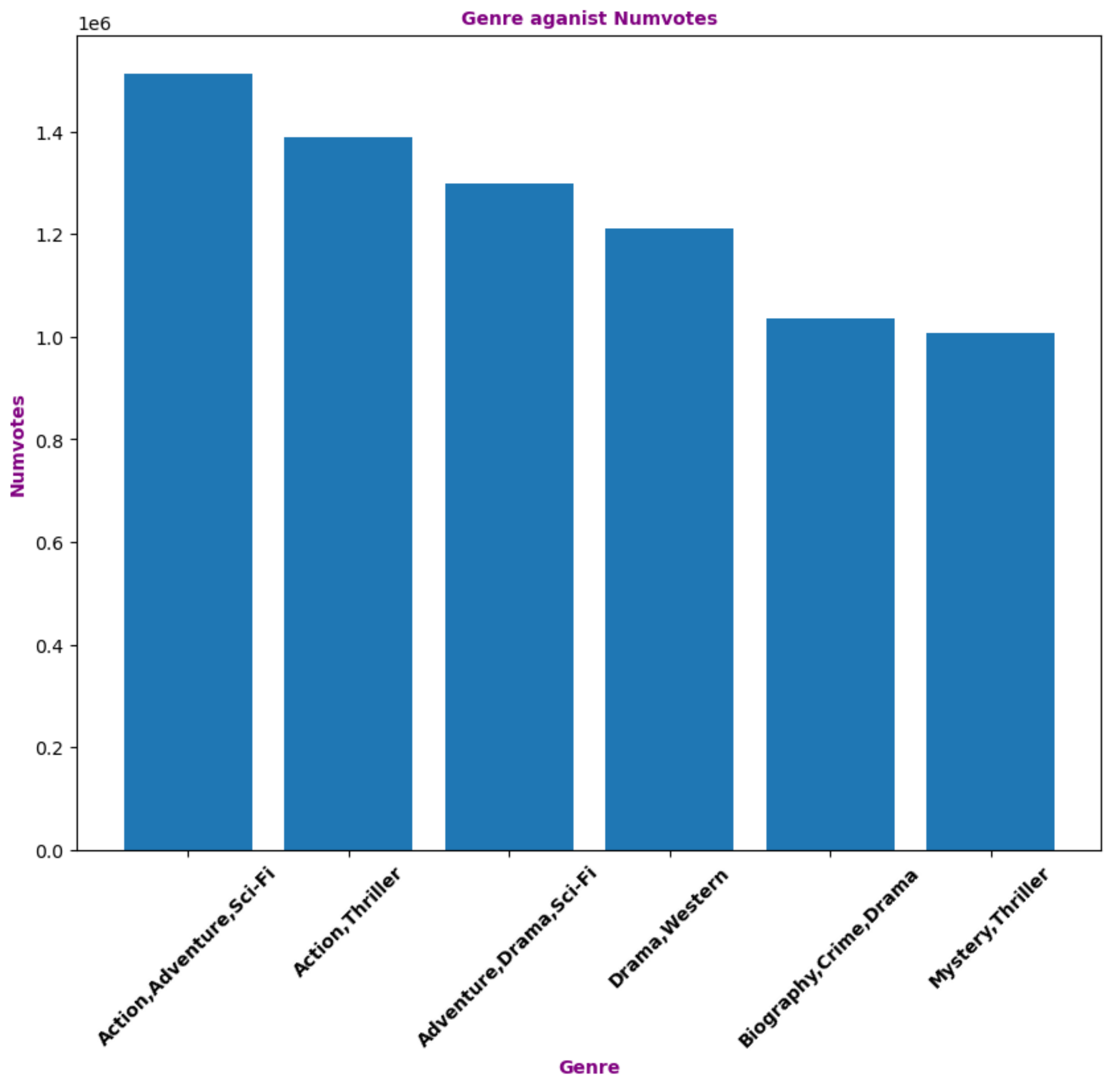
There is clearly no linear relationship between profits made from the sale of the films and the producion budget. Therefore, the company should source and strategize on ways of ensuring the utmost success of the movies rather than being wasteful on a larger production budget.

Genre and Numvotes

The dataset available is grouped by the number of votes and by the genre. This means we can do an analysis based on which genre had the most votes. We therefore get to plot the genres against the votes garnered.

In [140]:

```
#plotting the graph
plt.figure(figsize=(10, 8))
y = genres_mean_sorted["numvotes"]
plt.bar(y.index, y.values)
plt.xticks(rotation = 45 , fontsize = 10 , fontweight = "bold")
plt.xlabel("Genre" , fontsize = 10 , fontweight = "bold", color = "purple'
plt.ylabel("Numvotes", fontsize = 10 , fontweight = "bold" ,  color = "pur
plt.title("Genre aganist Numvotes", fontsize = 10 , fontweight = "bold" ,
plt.gcf().set_size_inches = "9 ,8"
plt.show()
```



EXPLANATION

Action,adventure,sci-fi received the highest number of votes as per the plot. The genre had the highest votes adding up to 151,260. The runners up position was for the Action and thriller which had 138,7769 streams. The least favorite genre was the and thriller with 1005960 votes.The company may consider venturing into producing the top two genres in the film making industry.

CONCLUSION

In conclusion, Microsoft should consider the following recommendations:

Invest in the production of action,adventure,sci-fi movies and or Action and thriller movie genres

Benchmark with the top two production heavyweights in the industry (BV and P/DW) on how to rake in on profits and success

Develop and implement strategies that ensure production budgets are low while maximizing onprofits

Ensure that they protect the property rights against piracy

In [ ]: