In [1]:
```
# Importing libraries
import pandas as pd
import numpy as np

#importing datasets
import json
import csv

#importing visualization datasets
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#Loading the csv files on the directory
! ls *.csv
```

```
bom.movie_gross.csv
name.basics.csv
title.akas.csv
title.basics.csv
title.crew.csv
title.principals.csv
title.ratings.csv
tmdb.movies.csv
tn.movie_budgets.csv
```

In [3]:
```
# Loading the datasets onto the notebook
df1 = pd.read_csv("bom.movie_gross.csv")
df1
```

	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

Interpretation: This dataset has 3387 entries and 5 columns The information captured is the movie titles, the studios the production happened and the revenues made in different years

In [4]:
```
# Loading the datasets onto the notebook
df2 = pd.read_csv("name.basics.csv")
df2
```

	nconst	primary_name	birth_year	death_year	primary_profession	know
0	nm0061671	Mary Ellen Bauder	NaN	NaN	miscellaneous,production_manager,producer	tt0837562,tt2398241,tt084447
1	nm0061865	Joseph Bauer	NaN	NaN	composer,music_department,sound_department	tt0896534,tt6791238,tt028707
2	nm0062070	Bruce Baum	NaN	NaN	miscellaneous,actor,writer	tt1470654,tt0363631,tt010403
3	nm0062195	Axel Baumann	NaN	NaN	camera_department,cinematographer,art_department	tt0114371,tt2004304,tt161844
4	nm0062798	Pete Baxter	NaN	NaN	production_designer,art_department,set_decorator	tt0452644,tt0452692,tt345803
...	...	...	...	...	...	
606643	nm9990381	Susan Grobes	NaN	NaN	actress	
606644	nm9990690	Joo Yeon So	NaN	NaN	actress	tt909093
606645	nm9991320	Madeline Smith	NaN	NaN	actress	tt873443
606646	nm9991786	Michelle Modigliani	NaN	NaN	producer	

Interpretation: This dataset has 606648 entries and 6 columns The information captured is the names of professional involved, their individual professional contribution and what film's (title's) they are known for.

In [5]:
```
# Loading the datasets onto the notebook
df3 = pd.read_csv("title.akas.csv")
df3
```

Out[5]:

	title_id	ordering	title	region	language	types	attributes	is_original_title
0	tt0369610	10	Джурасик свят	BG	bg	NaN	NaN	0.0
1	tt0369610	11	Jurashikku warudo	JP	NaN	imdbDisplay	NaN	0.0
2	tt0369610	12	Jurassic World: O Mundo dos Dinossauros	BR	NaN	imdbDisplay	NaN	0.0
3	tt0369610	13	O Mundo dos Dinossauros	BR	NaN	NaN	short title	0.0
4	tt0369610	14	Jurassic World	FR	NaN	imdbDisplay	NaN	0.0
...	...	...	...	...	...	...	...	...
331698	tt9827784	2	Sayonara kuchibiru	NaN	NaN	original	NaN	1.0
331699	tt9827784	3	Farewell Song	XWW	en	imdbDisplay	NaN	0.0
331700	tt9880178	1	La atención	NaN	NaN	original	NaN	1.0
331701	tt9880178	2	La atención	ES	NaN	NaN	NaN	0.0
331702	tt9880178	3	The Attention	XWW	en	imdbDisplay	NaN	0.0

Interpretation: This dataset has 331703 entries and 8 columns The information captured is the titles, the film's language, the film's attributes and whether the film is an original title of not.

In [6]:
```
# Loading the datasets onto the notebook
df4 = pd.read_csv("title.basics.csv")
df4
```

Out[6]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	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	116.0	NaN

Interpretation: This dataset has 146144 entries and 6 columns The information captured is the original titles, the start year of sale, how long the film is and the film's genre.

In [7]:
```
# Loading the datasets onto the notebook
df5 = pd.read_csv("title.crew.csv")
df5
```

	tconst	directors	writers
0	tt0285252	nm0899854	nm0899854
1	tt0438973	NaN	nm0175726,nm1802864
2	tt0462036	nm1940585	nm1940585
3	tt0835418	nm0151540	nm0310087,nm0841532
4	tt0878654	nm0089502,nm2291498,nm2292011	nm0284943
...	...	...	...
146139	tt8999974	nm10122357	nm10122357
146140	tt9001390	nm6711477	nm6711477
146141	tt9001494	nm10123242,nm10123248	NaN
146142	tt9004986	nm4993825	nm4993825
146143	tt9010172	NaN	nm8352242

146144 rows × 3 columns

Interpretation: This dataset has 146144 entries and 3 columns The information captured is the directors and writers of the film title

In [8]:
```
# Loading the datasets onto the notebook
df6 = pd.read_csv("title.principals.csv")
df6
```

Out[8]:

	tconst	ordering	nconst	category	job	characters
0	tt0111414	1	nm0246005	actor	NaN	["The Man"]
1	tt0111414	2	nm0398271	director	NaN	NaN
2	tt0111414	3	nm3739909	producer	producer	NaN
3	tt0323808	10	nm0059247	editor	NaN	NaN
4	tt0323808	1	nm3579312	actress	NaN	["Beth Boothby"]
...	...	...	...	...	...	...
1028181	tt9692684	1	nm0186469	actor	NaN	["Ebenezer Scrooge"]
1028182	tt9692684	2	nm4929530	self	NaN	["Herself","Regan"]
1028183	tt9692684	3	nm10441594	director	NaN	NaN
1028184	tt9692684	4	nm6009913	writer	writer	NaN
1028185	tt9692684	5	nm10441595	producer	producer	NaN

1028186 rows × 6 columns

Interpretation: This dataset has 1028186 entries and 6 columns The information captured is the characters and their jobs in the production of the movies.

In [9]:
```
# Loading the datasets onto the notebook
df7 = pd.read_csv("title.ratings.csv")
df7
```

Out[9]:

	tconst	averagerating	numvotes
0	tt10356526	8.3	31
1	tt10384606	8.9	559
2	tt1042974	6.4	20
3	tt1043726	4.2	50352
4	tt1060240	6.5	21
...	...	...	...
73851	tt9805820	8.1	25
73852	tt9844256	7.5	24
73853	tt9851050	4.7	14
73854	tt9886934	7.0	5
73855	tt9894098	6.3	128

73856 rows × 3 columns

Interpretation: This table has 73856 entries with only 3 columns

- The columns point out the rating and the number of votes for each movie title

In [10]:
```
# Loading the datasets onto the notebook
df8 = pd.read_csv("tmdb.movies.csv")
df8
```

1	1	16, 10751]	10191	en	Dragon	28.734	2010-03-26	Dragon	7.7
2	2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8
3	3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9
4	4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3
...	...	...	...	...	...	...	...	...	...
26512	26512	[27, 18]	488143	en	Laboratory Conditions	0.600	2018-10-13	Laboratory Conditions	0.0
26513	26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.600	2018-05-01	_EXHIBIT_84xxx_	0.0
26514	26514	[14, 28, 12]	381231	en	The Last One	0.600	2018-10-01	The Last One	0.0
26515	26515	[10751, 12, 28]	366854	en	Trailer Made	0.600	2018-06-22	Trailer Made	0.0

Interpretation: This dataset has 26517 entries and 10 columns The information captured is the popularity of each movie, the release date of each and the vote count

In [11]:
```
# Loading the datasets onto the notebook
df9 = pd.read_csv("tn.movie_budgets.csv")
df9
```

Out[11]:

	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

Interpretation: This dataset has 5782 entries and 6 columns The information captured is the title of each movie, the release date of each, the production budget and the gross profit made from each

Basic Analysis: There are 9 different datasets with varying columns hence analysis of the same is essential to determine which ones can be merged easily

In [12]:
```
#Analysing the columns in the 8th dataset
print(df8.columns)
print()
```

```
Index(['Unnamed: 0', 'genre_ids', 'id', 'original_language', 'original_title',
       'popularity', 'release_date', 'title', 'vote_average', 'vote_count'],
      dtype='object')
```

In [13]:
```
#Analysing the columns in the 9th dataset
print(df9.columns)
```

```
Index(['id', 'release_date', 'movie', 'production_budget', 'domestic_gross',
       'worldwide_gross'],
      dtype='object')
```

In [14]:
```
#Renaming the title for proper alignment of the columns
df9.rename(columns={'movie':'title'}, inplace = True)
df9.head(1)
```

Out[14]:

	id	release_date	title	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279

In [15]:
```
#My resolve is to work with the 8th and 9th datasets since the columns are almost similar hence easier concate
#Merging datasets df8 and df9 and naming the output df89
df89 = df8.merge(df9, how = 'outer', on ='title', suffixes=('_f8', '_f9'))
df89.head(1)
```

Out[15]:

	Unnamed: 0	genre_ids	id_f8	original_language	original_title	popularity	release_date_f8	title	vote_average	vote_count	id_f9
0	0.0	[12, 14, 10751]	12444.0	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788.0	NaN

Merging of datasets df8 and df9 has been done along the column title. This has been done under the condition outer in order to retain the various columns that shall assist with analysis

In [16]:
```
# Subject dataset under review
df89
```

Out[16]:

	Unnamed: 0	genre_ids	id_f8	original_language	original_title	popularity	release_date_f8	title	vote_average	vote_count
0	0.0	[12, 14, 10751]	12444.0	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788.0
1	1.0	[14, 12, 16, 10751]	10191.0	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610.0
2	2.0	[12, 28, 878]	10138.0	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368.0
3	3.0	[16, 35, 10751]	862.0	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174.0
4	2473.0	[16, 35, 10751]	862.0	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174.0
...	...	...	...	...	...	...	...	...	...	...
30406	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Red 11	NaN	NaN
30407	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Following	NaN	NaN
30408	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Return to the Land of Wonders	NaN	NaN
30409	NaN	NaN	NaN	NaN	NaN	NaN	NaN	A Plague So Pleasant	NaN	NaN
30410	NaN	NaN	NaN	NaN	NaN	NaN	NaN	My Date With Drew	NaN	NaN

30411 rows × 15 columns

DATA PREVIEW

In [17]:
```
#Reviewing the basic data info in the columns of the df89 dataset
df89.info()
```

```
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         26606 non-null  float64
 1   genre_ids          26606 non-null  object 
 2   id_f8              26606 non-null  float64
 3   original_language  26606 non-null  object 
 4   original_title     26606 non-null  object 
 5   popularity         26606 non-null  float64
 6   release_date_f8    26606 non-null  object 
 7   title              30411 non-null  object 
 8   vote_average       26606 non-null  float64
 9   vote_count         26606 non-null  float64
 10  id_f9              6190 non-null   float64
 11  release_date_f9    6190 non-null   object 
 12  production_budget  6190 non-null   object 
 13  domestic_gross     6190 non-null   object 
 14  worldwide_gross    6190 non-null   object 
dtypes: float64(6), object(9)
memory usage: 3.7+ MB
```

In [18]:
```
#Checking out the contents of the individual columns
df89.describe()
```

Out[18]:

	Unnamed: 0	id_f8	popularity	vote_average	vote_count	id_f9
count	26606.000000	26606.000000	26606.000000	26606.000000	26606.000000	6190.000000
mean	13252.894234	294812.769601	3.155421	5.991220	199.429039	50.466236
std	7654.629576	153702.703736	4.407785	1.851133	977.486054	28.752526
min	0.000000	27.000000	0.600000	0.000000	1.000000	1.000000
25%	6625.250000	157803.250000	0.600000	5.000000	2.000000	26.000000
50%	13255.500000	309153.500000	1.376000	6.000000	5.000000	51.000000
75%	19882.750000	419438.250000	3.734000	7.000000	28.000000	75.000000
max	26516.000000	608444.000000	80.773000	10.000000	22186.000000	100.000000

In [19]:
```
#Data preview
df89.head()
```

Out[19]:

	Unnamed: 0	genre_ids	id_f8	original_language	original_title	popularity	release_date_f8	title	vote_average	vote_count	id_f9
0	0.0	[12, 14, 10751]	12444.0	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788.0	NaN
1	1.0	[14, 12, 16, 10751]	10191.0	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610.0	30.0
2	2.0	[12, 28, 878]	10138.0	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368.0	15.0
3	3.0	[16, 35, 10751]	862.0	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174.0	37.0
4	2473.0	[16, 35, 10751]	862.0	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174.0	37.0

In [20]:
```
#Data preview
df89.tail()
```

Out[20]:

	Unnamed: 0	genre_ids	id_f8	original_language	original_title	popularity	release_date_f8	title	vote_average	vote_count	id_
30406	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Red 11	NaN	NaN	78
30407	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Following	NaN	NaN	79
30408	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Return to the Land of Wonders	NaN	NaN	80
30409	NaN	NaN	NaN	NaN	NaN	NaN	NaN	A Plague So Pleasant	NaN	NaN	81
30410	NaN	NaN	NaN	NaN	NaN	NaN	NaN	My Date With Drew	NaN	NaN	82

In [21]:
```
#Data preview
df89.dtypes
```

Out[21]:
```
Unnamed: 0            float64
genre_ids              object
id_f8                 float64
original_language      object
original_title         object
popularity            float64
release_date_f8        object
title                  object
vote_average          float64
vote_count            float64
id_f9                 float64
release_date_f9        object
production_budget      object
domestic_gross         object
worldwide_gross        object
dtype: object
```

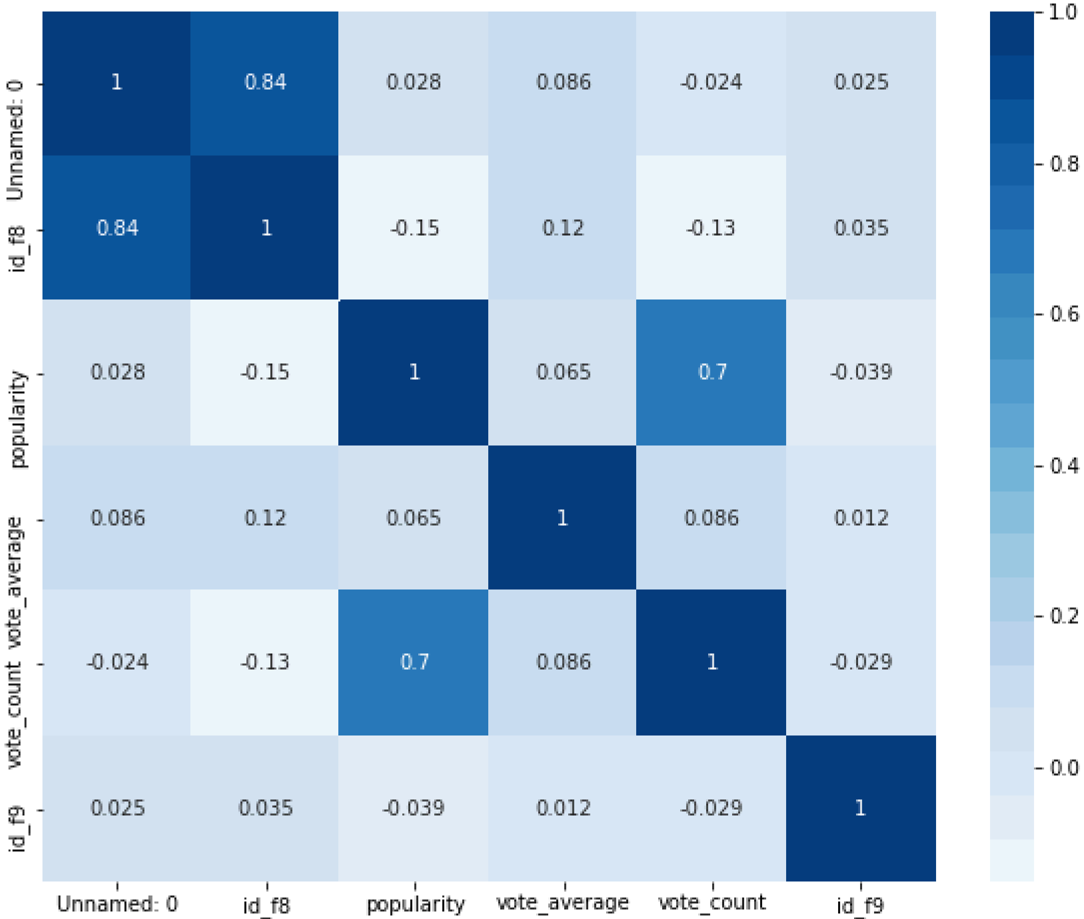
In [22]:
```
#Data preview
df89.columns
```

Out[22]:
```
Index(['Unnamed: 0', 'genre_ids', 'id_f8', 'original_language',
       'original_title', 'popularity', 'release_date_f8', 'title',
       'vote_average', 'vote_count', 'id_f9', 'release_date_f9',
       'production_budget', 'domestic_gross', 'worldwide_gross'],
      dtype='object')
```

In [23]:
```
#Checking for duplicates
df89.duplicated()# There are no duplicates
```

Out[23]:
```
0        False
1        False
2        False
3        False
4        False
         ...
30406    False
30407    False
30408    False
30409    False
30410    False
Length: 30411, dtype: bool
```

In [24]:
```
# To check useful values we can use correlation analysis to see which columns have high correlation
import seaborn as sns
plt.figure(figsize = (10,8))
dataplot = sns.heatmap(df89.corr(), cmap=sns.color_palette("Blues",20), annot = True)
#unnamed has the lowest correlation with other columns hence should be dropped
```



In [25]:
```
#Checking for missing values
df89.isnull().mean()*100
```

Out[25]:
```
Unnamed: 0           12.511920
genre_ids            12.511920
id_f8                12.511920
original_language    12.511920
original_title       12.511920
popularity           12.511920
release_date_f8      12.511920
title                 0.000000
vote_average         12.511920
vote_count           12.511920
id_f9                79.645523
release_date_f9      79.645523
production_budget    79.645523
domestic_gross       79.645523
worldwide_gross      79.645523
dtype: float64
```

In [26]:
```
#Dropping unnecessary columns
df89.dropna(subset = ['popularity', 'production_budget'])
```

Out[26]:

	Unnamed: 0	genre_ids	id_f8	original_language	original_title	popularity	release_date_f8	title	vote_average	vote_co
1	1.0	[14, 12, 16, 10751]	10191.0	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	761
2	2.0	[12, 28, 878]	10138.0	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	1236
3	3.0	[16, 35, 10751]	862.0	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	1017
4	2473.0	[16, 35, 10751]	862.0	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	1017
5	4.0	[28, 878, 12]	27205.0	en	Inception	27.920	2010-07-16	Inception	8.3	2218
...	...	...	...	...	...	...	...	...	...	
26192	26323.0	[]	509316.0	en	The Box	0.600	2018-03-04	The Box	8.0	
26193	26425.0	[10402]	509306.0	en	The Box	0.600	2018-03-04	The Box	6.0	
26235	26092.0	[35, 16]	546674.0	en	Enough	0.719	2018-03-22	Enough	8.7	
26441	26322.0	[]	513161.0	en	Undiscovered	0.600	2018-04-07	Undiscovered	8.0	
26599	26508.0	[16]	514492.0	en	Jaws	0.600	2018-05-29	Jaws	0.0	

2385 rows × 15 columns

In [27]:
```
#dropping the column 'Unnamed:0' since it is of no significance in my analysis
df89 = df89.drop(columns = ['Unnamed: 0'])
```

In [28]:
```
#checking the number of missing values
df89.original_language.isnull().sum()
```

Out[28]: 3805

In [29]:
```
#checking the total number of duplicated values
df89.duplicated().sum()
```

Out[29]: 1024

In [30]:
```
#Dropping the duplicates
df89 = df89.drop_duplicates()
df89
```

5	[illegible] 12]	27205.0	en	Inception	27.920	2010-07-16	Inception	8.3	22186.0	38.0
...	...	...	...	...	...	...	...	...	...	...
30406	NaN	NaN	NaN	NaN	NaN	NaN	Red 11	NaN	NaN	78.0
30407	NaN	NaN	NaN	NaN	NaN	NaN	Following	NaN	NaN	79.0
30408	NaN	NaN	NaN	NaN	NaN	NaN	Return to the Land of Wonders	NaN	NaN	80.0
30409	NaN	NaN	NaN	NaN	NaN	NaN	A Plague So Pleasant	NaN	NaN	81.0
30410	NaN	NaN	NaN	NaN	NaN	NaN	My Date With Drew	NaN	NaN	82.0

29387 rows × 14 columns

In [31]:
```
#The complete dataset but with missing values
df89
```

Out[31]:

	genre_ids	id_f8	original_language	original_title	popularity	release_date_f8	title	vote_average	vote_count	id_f9	rel
0	[12, 14, 10751]	12444.0	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788.0	NaN	
1	[14, 12, 16, 10751]	10191.0	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610.0	30.0	
2	[12, 28, 878]	10138.0	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368.0	15.0	
3	[16, 35, 10751]	862.0	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174.0	37.0	
5	[28, 878, 12]	27205.0	en	Inception	27.920	2010-07-16	Inception	8.3	22186.0	38.0	

In [32]:
```
#Checking  for duplicates
df89.duplicated().sum()
#No duplicates found
```

Out[32]: 0

In [33]:
```
#data preview
df89
```

5	[28, 878, 12]	27205.0	en	Inception	27.920	2010-07-16	Inception	8.3	22186.0	38.0
...	...	...	...	...	...	...	...	...	...	...
30406	NaN	NaN	NaN	NaN	NaN	NaN	Red 11	NaN	NaN	78.0
30407	NaN	NaN	NaN	NaN	NaN	NaN	Following	NaN	NaN	79.0
30408	NaN	NaN	NaN	NaN	NaN	NaN	Return to the Land of Wonders	NaN	NaN	80.0
30409	NaN	NaN	NaN	NaN	NaN	NaN	A Plague So Pleasant	NaN	NaN	81.0
30410	NaN	NaN	NaN	NaN	NaN	NaN	My Date With Drew	NaN	NaN	82.0

29387 rows × 14 columns

In [34]:
```
#There's is a huge chunk of missing values especially in the financials part of most films but dropping the co
df89.isna().mean()
```

Out[34]:
```
genre_ids            0.129479
id_f8                0.129479
original_language    0.129479
original_title       0.129479
popularity           0.129479
release_date_f8      0.129479
title                0.000000
vote_average         0.129479
vote_count           0.129479
id_f9                0.795284
release_date_f9      0.795284
production_budget    0.795284
domestic_gross       0.795284
worldwide_gross      0.795284
dtype: float64
```

In [35]:
```
#A preview of the financial components of my data set for easier analysis
finances = [
    'production_budget', 'domestic_gross',
    'worldwide_gross']
df89[finances].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29387 entries, 0 to 30410
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   production_budget  6016 non-null   object
 1   domestic_gross     6016 non-null   object
 2   worldwide_gross    6016 non-null   object
dtypes: object(3)
memory usage: 918.3+ KB
```

Observations:

There are inconsistencies in the foreign_gross and worldwide_gross columns in terms of values.

Convert the dtypes of the entries in production_budget, domestic_gross and worldwide_gross

In [36]:
```
#A preview of the financial components of my data set for easier analysis
df89[finances].head(3)
```

Out[36]:

	production_budget	domestic_gross	worldwide_gross
0	NaN	NaN	NaN
1	$165,000,000	$217,581,232	$494,870,992
2	$170,000,000	$312,433,331	$621,156,389

POPULARITY SCORES

In [37]:
```
#A preview of the popularity score of my data set for easier analysis
popularity_scores = [
'title', 'popularity', 'vote_average', 'vote_count'
]
df89[popularity_scores].head()
```

Out[37]:

	title	popularity	vote_average	vote_count
0	Harry Potter and the Deathly Hallows: Part 1	33.533	7.7	10788.0
1	How to Train Your Dragon	28.734	7.7	7610.0
2	Iron Man 2	28.515	6.8	12368.0
3	Toy Story	28.005	7.9	10174.0
5	Inception	27.920	8.3	22186.0

In [38]:
```
#A preview of the popularity score of my data set for easier analysis
df89[popularity_scores].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29387 entries, 0 to 30410
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   title         29387 non-null  object 
 1   popularity    25582 non-null  float64
 2   vote_average  25582 non-null  float64
 3   vote_count    25582 non-null  float64
dtypes: float64(3), object(1)
memory usage: 1.1+ MB
```

Popularity Scores Actionables

(i) The 3 columns, i.e. popularity, vote_average, vote_count, are related and they include a popularity score.

Guiding Questions:

1. Number of titles used
2. Original_title vs current title

3. Genre used
4. Production budget vs income earned(both domestic and worldwide gross)

1. NUMBER OF TITLES

In [39]:
```
print(df89.columns)


# unique titles
df89.title.nunique()
```

```
Index(['genre_ids', 'id_f8', 'original_language', 'original_title',
       'popularity', 'release_date_f8', 'title', 'vote_average', 'vote_count',
       'id_f9', 'release_date_f9', 'production_budget', 'domestic_gross',
       'worldwide_gross'],
      dtype='object')
```

Out[39]:
```
28462
```

2. Original Title

In [40]:
```
# number of original titles
original_ser = df89.groupby('original_title').original_title.count()#.to_frame()
Orig_Titles = pd.DataFrame(original_ser).rename(columns = {'original_title': 'Count'})
x = list(Orig_Titles.values)
x

Orig_Titles.values
```

Out[40]:
```
array([[1],
       [1],
       [1],
       ...,
       [1],
       [1],
       [1]], dtype=int64)
```

In [41]:
```
#Analysis of the movies that retained their original title compared to the income generated from their sale
df89[['title','original_title','production_budget', 'domestic_gross', 'worldwide_gross']].groupby('title').cou
```

Out[41]:

	original_title	production_budget	domestic_gross	worldwide_gross
title				
Home	21	21	21	21
The Gift	10	10	10	10
Beauty and the Beast	6	6	6	6
Robin Hood	6	6	6	6
Eden	5	5	5	5
The Box	5	5	5	5
Truth or Dare	5	5	5	5
Alice in Wonderland	4	4	4	4
Brothers	4	4	4	4
Carrie	4	4	4	4

From the above analysis, we can infer that retaining the original title of the movie will impact the sales positively since the film sales are higher as compared to those whose title was changed

3. GENRE

In [42]:
```
# Best performing genres by gross
print(df89.title.unique())

df89.groupby(['genre_ids', 'domestic_gross', 'worldwide_gross']
          ).title.value_counts().to_frame().sort_values(
        by=['domestic_gross'], ascending = False)[:10]
```

```
['Harry Potter and the Deathly Hallows: Part 1' 'How to Train Your Dragon'
 'Iron Man 2' ... 'Return to the Land of Wonders' 'A Plague So Pleasant'
 'My Date With Drew']
```

Out[42]:

				title
genre_ids	domestic_gross	worldwide_gross	title	
[12, 35, 10751, 14]	$99,215,042	$197,504,758	Christopher Robin	1
[28, 12]	$99,112,101	$250,700,000	Hercules	1
[28, 53]	$98,927,592	$172,878,928	Olympus Has Fallen	1
[28, 80, 35]	$98,780,042	$229,155,503	The Green Hornet	1
[35]	$98,711,404	$152,269,033	Date Night	1
[18, 37]	$977,772	$1,869,928	Meek's Cutoff	1
[18]	$970,816	$5,046,038	The Beaver	1
[12, 16, 35, 14]	$97,670,358	$141,344,255	Sausage Party	1
[18]	$96,962,694	$224,922,135	The Social Network	1
[18, 28]	$96,734	$96,734	Hardflip	1

My proposal is that Microsoft should incorporate various genres in their film production as the target audience is quite vast which shall increase the gross income

4. Original Language

In [43]:
```
# what are the unique languages in the dataset
print(df89.original_language.nunique())

df89.original_language.unique()
```

```
76
```

Out[43]:
```
array(['en', 'nl', 'zh', 'es', 'ja', 'sv', 'de', 'fr', 'cn', 'ru', 'it',
       'hi', 'tl', 'da', 'no', 'ko', 'fi', 'pl', 'te', 'hu', 'tr', 'ka',
       'pt', 'he', 'fa', 'th', 'cs', 'et', 'lt', 'sr', 'xx', 'bs', 'ar',
       'is', 'el', 'mr', 'hr', 'ro', 'uk', 'nb', 'hz', 'ca', 'cy', 'bg',
       'sl', 'lv', 'si', 'ab', 'ta', 'bo', 'id', 'gu', 'sq', 'bn', 'lo',
       'ne', 'kk', 'hy', 'ps', 'kn', 'vi', 'ku', 'ml', 'ur', 'mi', 'eu',
       'sn', 'ha', 'ky', 'yi', 'pa', 'xh', 'cr', 'sw', 'af', 'dz', nan],
      dtype=object)
```

My recommendation is that when engaging in movie production, Microsoft should incorporate the use of various languages in their films hence maximising on the audience reach out as this will increase the income earned.

In [44]:
```
#A preview of the films produced using different languages and the production budget compared to the income ea
df89[['title',  'genre_ids', 'original_language', 'production_budget', 'domestic_gross',
    'worldwide_gross']].groupby('original_language').count().sort_values(by=['worldwide_gross',
    'worldwide_gross'], ascending = False)[:15]
```

Out[44]:

original_language	title	genre_ids	production_budget	domestic_gross	worldwide_gross
en	22462	22462	2087	2087	2087
fr	484	484	24	24	24
es	439	439	16	16	16
ru	299	299	16	16	16
zh	175	175	11	11	11
hi	171	171	10	10	10
de	231	231	8	8	8
sv	66	66	4	4	4
ja	244	244	3	3	3
ko	92	92	3	3	3
ar	32	32	2	2	2
it	119	119	2	2	2
nl	46	46	2	2	2
no	48	48	2	2	2
pl	51	51	2	2	2

My proposal to microsoft is to consider producing a huge chunk of their films in the English language to reach a wider audience while incorporating other languages to serve the minority. This is is well depicted in the first axis where English movies have made the most sales when compared to the rest

VISUALIZATION:

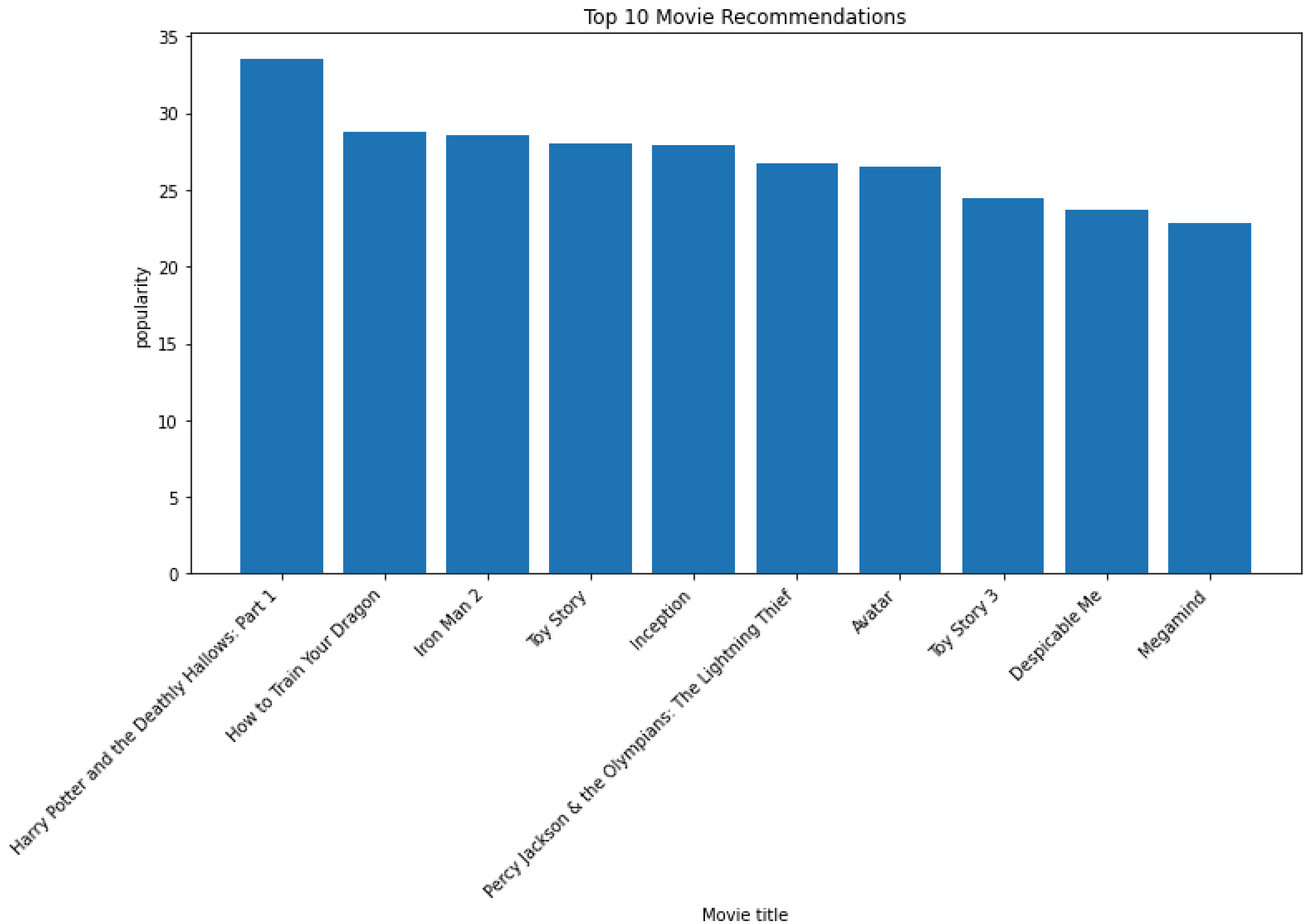
In [45]:
```
#Creating a list of the top 10 movies for easier analysis
x = df89.title[:10]
x
```

Out[45]:
```
0           Harry Potter and the Deathly Hallows: Part 1
1                                 How to Train Your Dragon
2                                               Iron Man 2
3                                                Toy Story
5                                                Inception
6      Percy Jackson & the Olympians: The Lightning T...
7                                                   Avatar
8                                              Toy Story 3
9                                            Despicable Me
10                                               Megamind
Name: title, dtype: object
```

In [46]:
```
#Creating a list of the popularity of the top 10 movies for easier analysis
y = df89.popularity[:10]
y
```

Out[46]:
```
0     33.533
1     28.734
2     28.515
3     28.005
5     27.920
6     26.691
7     26.526
8     24.445
9     23.673
10    22.855
Name: popularity, dtype: float64
```

In [47]:
```
#Creating a barchart to showcase the top 10 most popular movies in the industry
popularity= y
Movie_title = x
bar_chart_title = 'Top 10 Movie Recommendations'
bar_chart_count_label = 'Popularity'
bar_chart_series_label = 'Movie title'

plt.figure(figsize=(12, 6))
plt.bar(x, y)
plt.xlabel('Movie title')
plt.ylabel('popularity')
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better readability
plt.title(bar_chart_title )
plt.show()
```

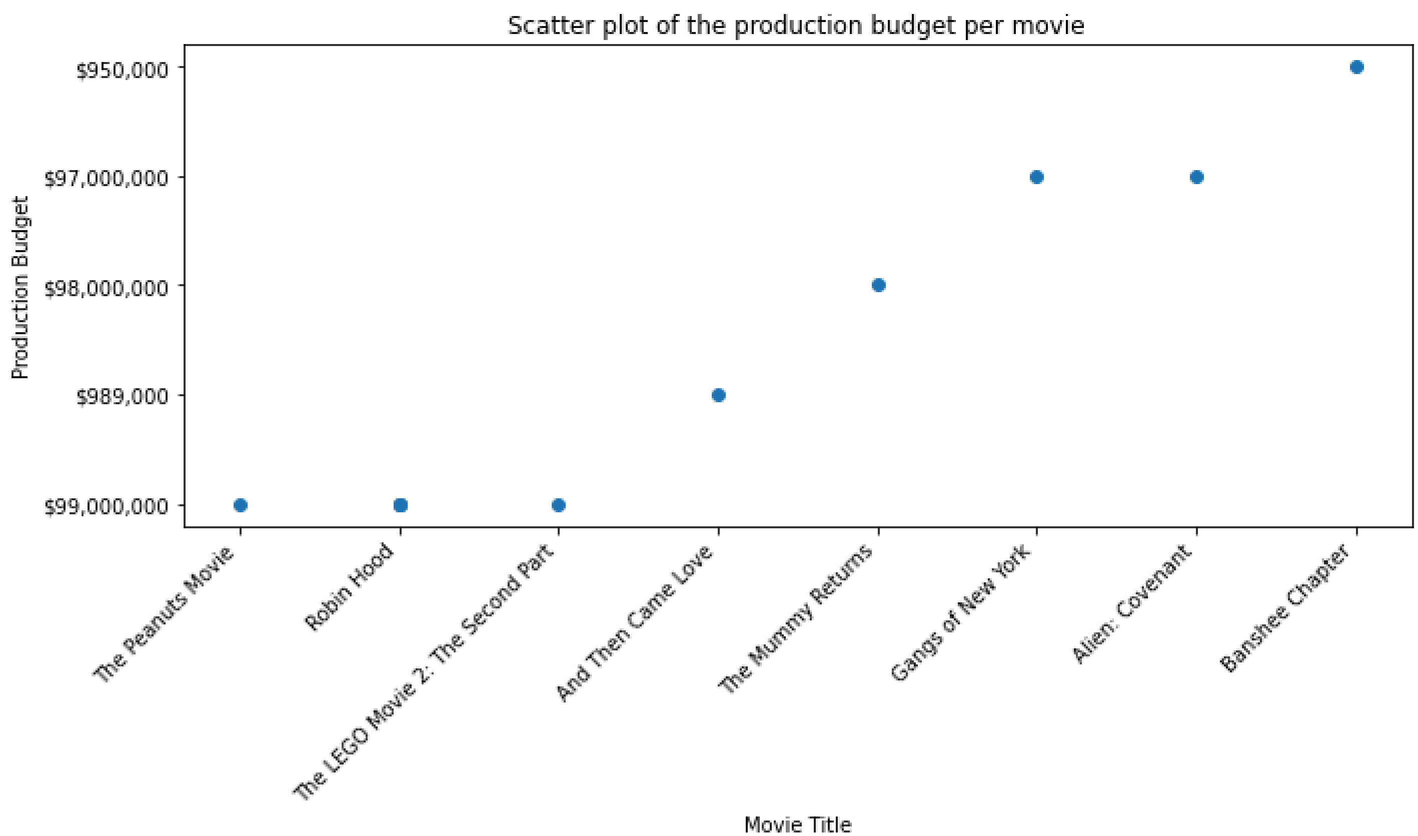


Recommendations:

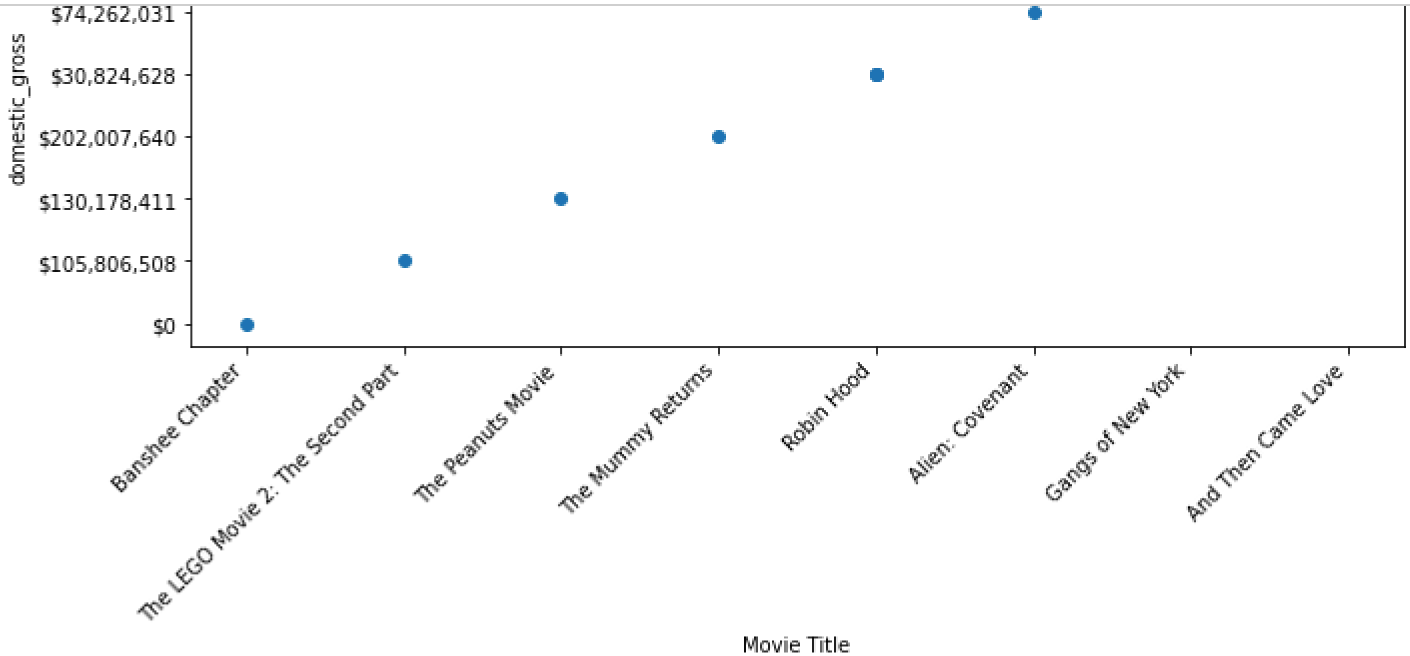
```
I propose that microsoft builds it's studio and produce films guided by the template used in creating
the following movies since they were the highest saught after in the industry and garnered a lot of p
opularity:
```

0 Harry Potter and the Deathly Hallows: Part 1 1 How to Train Your Dragon 2 Iron Man 2 3 Toy Story 5 Inception 6 Percy Jackson & the

In [48]:
```
#checking the production budget of the most expensive top 10 films
df89_sorted = df89.sort_values(by = 'production_budget', ascending = False)
top_10_production_budget = df89_sorted.head(10)

plt.figure(figsize=(10, 6))
plt.scatter(top_10_production_budget['title'], top_10_production_budget['production_budget'])
plt.title('Scatter plot of the production budget per movie')
plt.xlabel('Movie Title')
plt.ylabel('Production Budget')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



In [49]:
```
#checking the correlation between domestic gross of the top 10 films with the highest production budget as sho
df89_sorted1 = df89.sort_values(by = 'domestic_gross', ascending = False)
top_10_domestic_gross = df89_sorted.head(10)
top_10_domestic_gross_sorted = top_10_domestic_gross.sort_values(by='domestic_gross')

plt.figure(figsize=(10, 6))
plt.scatter(top_10_domestic_gross_sorted['title'], top_10_domestic_gross_sorted['domestic_gross'])
plt.title('Scatter plot of the domestic_gross per movie')
plt.xlabel('Movie Title')
plt.ylabel('domestic_gross')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



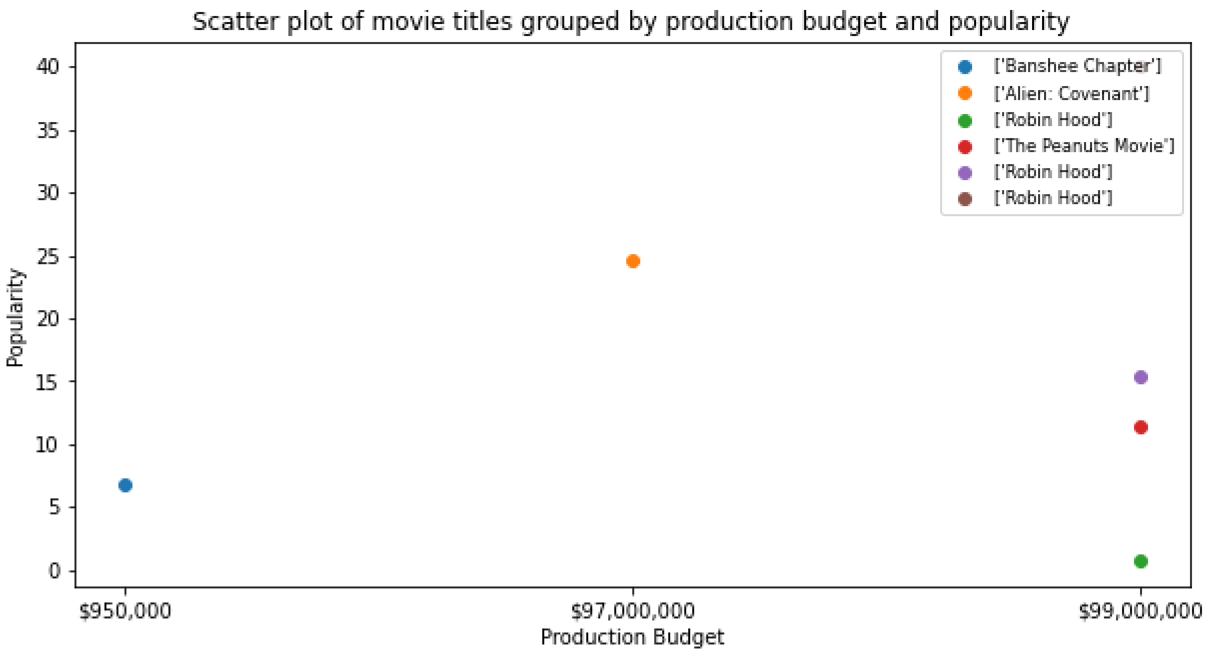
From the 2 scatter plots above, it has been clearly depicted how essential it is to invest well in movie production and ensure that the product is of the best quality as that will greatly boost the sale. This is especially in the case of films such as: The Mummy Returns whose production costed $98M but ended up generating $202M in profit together with the Peanut Movie whose budget was $99M but the sales were worth $130M. However, that is not always the case since other films ended up generating very little in come with some, barely covering the production costs hence leading to losses. This is well illustrated in the case of the film Banshee whose production cost was $950k but ended up not selling locally and Robin Hood whose production costed $99M but ended up generating $30M in local sales. That points out to the importance of considering other factors that may affect the sale of films which may include: 1. The target market, 2. Release timings(which should be strategic eg on weekends or holidays, 3. Runtime, 4. Production house or even 5. Directors used)

In [50]:
```
#Affirmation that the plotted info is correct
df89.loc[df89['title'].str.contains('Banshee Chapter')]
```

Out[50]:

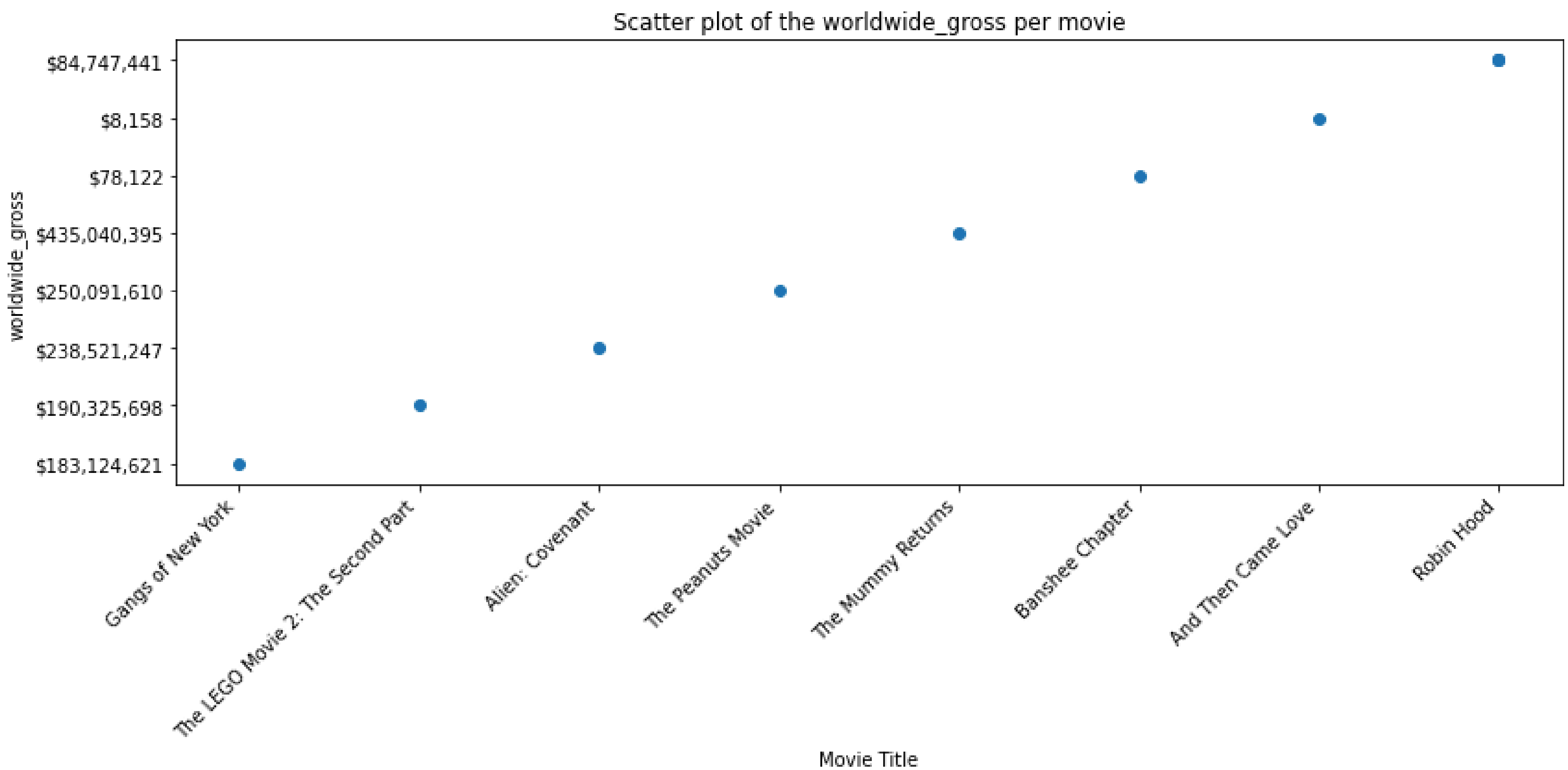
	genre_ids	id_f8	original_language	original_title	popularity	release_date_f8	title	vote_average	vote_count	id_f9	releas
8774	[27, 53]	207769.0	en	Banshee Chapter	6.872	2013-11-06	Banshee Chapter	5.6	135.0	30.0	Ja

In [51]:
```
#Popularity vs the production budget of the films
grouped_data = top_10_production_budget.groupby(['production_budget', 'popularity'])['title'].apply(list).rese
plt.figure(figsize=(10, 5))
for index, row in grouped_data.iterrows():
    production_budget = row['production_budget']
    popularity = row['popularity']
    titles = row['title']
    plt.scatter([production_budget] * len(titles), [popularity] * len(titles), label=titles)
plt.title('Scatter plot of movie titles grouped by production budget and popularity')
plt.xlabel('Production Budget')
plt.ylabel('Popularity')
# plt.tight_layout()
plt.legend(loc='upper right', fontsize='small')
plt.show()
```



From the above scatter plot, there's a clear demonstration of how popularity is a crucial factor in film production as it alludes to the tastes and preferences of the public. A lot was invested in the creation of Robin Hood which was clearly not in demand hence a huge loss was made. On the other hand, public opinion does not necessarily imply a favorable reaction to a movie as in the case of Alien:Covenant whose sales finally led to a loss. This is mostly in movie attendance which is a crude measure of public taste which may not translate to sales. It is therefore crucial that when it comes to production that Microsoft should not rely solely on public opinion since other factors might come into play

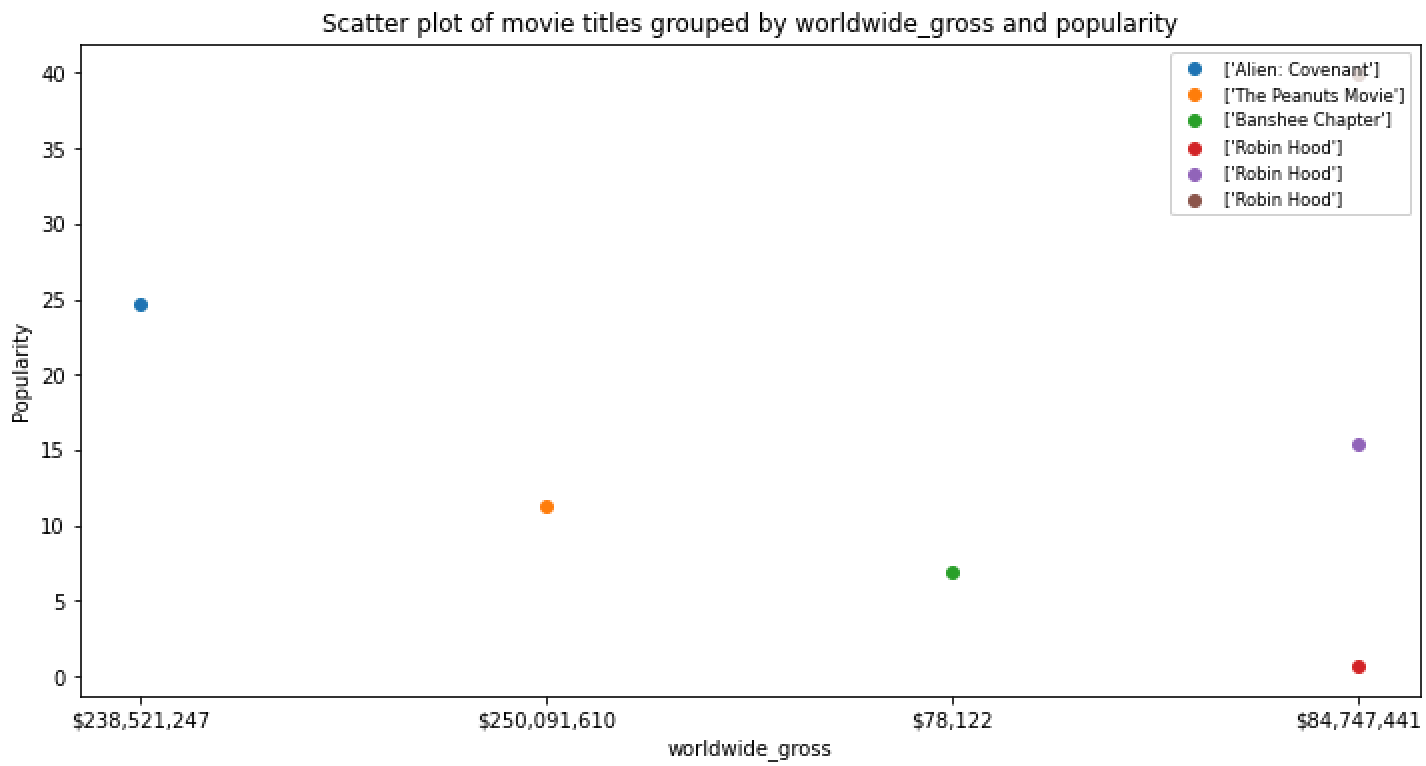
In [52]:
```
#checking the worldwide_gross of the top 10 films
df89_sorted1 = df89.sort_values(by = 'worldwide_gross', ascending = False)
top_10_worldwide_gross = df89_sorted.head(10)
top_10_worldwide_gross_sorted = top_10_worldwide_gross.sort_values(by='worldwide_gross')

plt.figure(figsize=(12, 6))
plt.scatter(top_10_worldwide_gross_sorted['title'], top_10_worldwide_gross_sorted['worldwide_gross'])
plt.title('Scatter plot of the worldwide_gross per movie')
plt.xlabel('Movie Title')
plt.ylabel('worldwide_gross')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



The above scatter plot has demonstrated how vast consumer preferences are hence the need for Microsoft to consider the international consumers in their production. For instance, Alien:Covenant did so well in the worldwide market and ended up

generating more than twice the budget of the production cost in sales. However, it failed terribly in the domestic market. More effort should be made in marketing the films worlwide for wider outreach. Other films such as The Mummy Returns ended up making quadruple the amount invested in the production hence the wider the target audience, the higher the income

In [53]:
```
grouped_data = top_10_worldwide_gross_sorted.groupby(['worldwide_gross', 'popularity'])['title'].apply(list).r
plt.figure(figsize=(12, 6))
for index, row in grouped_data.iterrows():
    worldwide_gross = row['worldwide_gross']
    popularity = row['popularity']
    titles = row['title']
    plt.scatter([worldwide_gross] * len(titles), [popularity] * len(titles), label=titles)
plt.title('Scatter plot of movie titles grouped by worldwide_gross and popularity')
plt.xlabel('worldwide_gross')
plt.ylabel('Popularity')
# plt.tight_layout()
plt.legend(loc='upper right', fontsize='small')
plt.show()
```



We can observe that high popularity definitely plays a factor in movie sales as in the case of Alien:Covenant. Therefore, Microsoft should invest a lot in film quality, especially by working on the storyline, acting, directing and marketing compaigns.