Final Project Submission

Please fill out:

- Student name: DIANA MBUVI
- Student pace: PART TIME

import pandas as pd import numpy as np import matplotlib.pyplot as plt %matplotlib inline

Importing the data

UNDERSTANDING THE DATA

In [251]:
```
df=pd.read_csv('C:/Users/Lenovo/Desktop/Phase_one_project/dsc-phase-1-project/zippedData/tmdb.movies.csv', in
df

#reading the data from a csv file
```

Out[251]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count
0	[12, 14, 10751]	12444	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
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610
2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174
4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186
...	...	...	...	...	...	...	...	...	...
26512	[27, 18]	488143	en	Laboratory Conditions	0.600	2018-10-13	Laboratory Conditions	0.0	1
26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.600	2018-05-01	_EXHIBIT_84xxx_	0.0	1
26514	[14, 28, 12]	381231	en	The Last One	0.600	2018-10-01	The Last One	0.0	1
26515	[10751, 12, 28]	366854	en	Trailer Made	0.600	2018-06-22	Trailer Made	0.0	1
26516	[53, 27]	309885	en	The Church	0.600	2018-10-05	The Church	0.0	1

26517 rows × 9 columns

In [252]:
```
df1=pd.read_csv('C:/Users/Lenovo/Desktop/Phase_one_project/dsc-phase-1-project/zippedData/tn.movie_budgets.csv
df1

#reading the data from a csv file
```

Out[252]:

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

In [253]:
```
df.columns #getting the columns titles from the data
```

Out[253]:
```
Index(['genre_ids', 'id', 'original_language', 'original_title', 'popularity',
       'release_date', 'title', 'vote_average', 'vote_count'],
      dtype='object')
```

In [254]:
```
df1.columns #getting the columns from the second data set
```

Out[254]:
```
Index(['release_date', 'movie', 'production_budget', 'domestic_gross',
       'worldwide_gross'],
      dtype='object')
```

In [255]:
```
# Convert 'release_date' to datetime type
df['release_date'] = pd.to_datetime(df['release_date'])
df1['release_date'] = pd.to_datetime(df1['release_date'])

# Merge the dataframes based on 'release_date' and 'original_title'
df2= pd.merge(df, df1, left_on=['release_date', 'original_title'], right_on=['release_date', 'movie'], how='i
```
```
# Display the merged dataframe
df2
```

Out[255]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	movie	produ
0	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	How to Train Your Dragon	
1	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368	Iron Man 2	
2	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174	Toy Story	
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174	Toy Story	
4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186	Inception	
...	...	...	...	...	...	...	...	...	...	...	
1396	[53, 18, 27]	510284	en	Braid	5.972	2019-02-01	Braid	6.4	18	Braid	
1397	[18, 10752]	514407	en	Indivisible	5.599	2018-10-26	Indivisible	5.4	13	Indivisible	
1398	[18, 28, 80]	547590	en	El Chicano	5.274	2019-05-03	El Chicano	9.0	1	El Chicano	
1399	[18, 35, 28, 80]	506971	ur	Teefa in Trouble	4.486	2018-07-20	Teefa in Trouble	7.6	11	Teefa in Trouble	
1400	[28, 12, 16]	332718	en	Bilal: A New Breed of Hero	2.707	2018-02-02	Bilal: A New Breed of Hero	6.8	54	Bilal: A New Breed of Hero	

1401 rows × 13 columns

In [256]:
```
df2.head() #getting the first 5 rows of the data
```

Out[256]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	movie	production_
0	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	How to Train Your Dragon	$165.
1	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368	Iron Man 2	$170.
2	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174	Toy Story	$30.
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174	Toy Story	$30.
4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186	Inception	$160.

In [257]:
```
df2.info() #gives information about the data set
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1401 entries, 0 to 1400
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   genre_ids          1401 non-null   object
 1   id                 1401 non-null   int64
 2   original_language  1401 non-null   object
 3   original_title     1401 non-null   object
 4   popularity         1401 non-null   float64
 5   release_date       1401 non-null   datetime64[ns]
 6   title              1401 non-null   object
 7   vote_average       1401 non-null   float64
 8   vote_count         1401 non-null   int64
 9   movie              1401 non-null   object
 10  production_budget  1401 non-null   object
 11  domestic_gross     1401 non-null   object
 12  worldwide_gross    1401 non-null   object
dtypes: datetime64[ns](1), float64(2), int64(2), object(8)
memory usage: 153.2+ KB
```

In [258]:
```
df2.dtypes #getting the type of data
```

Out[258]:
```
genre_ids                    object
id                            int64
original_language            object
original_title               object
popularity                  float64
release_date         datetime64[ns]
title                        object
vote_average                float64
vote_count                    int64
movie                        object
production_budget            object
domestic_gross               object
worldwide_gross              object
dtype: object
```

In [259]:
```
df2.describe() #describes the data
```

Out[259]:

	id	popularity	vote_average	vote_count
count	1401.000000	1401.000000	1401.000000	1401.000000
mean	202685.153462	13.346337	6.346824	2243.757316
std	138350.904263	8.051220	0.865988	2993.835538
min	95.000000	0.600000	3.000000	1.000000
25%	68728.000000	8.459000	5.800000	373.000000
50%	205584.000000	11.546000	6.400000	1097.000000
75%	323676.000000	16.302000	6.900000	2817.000000
max	547590.000000	80.773000	9.000000	22186.000000

In [260]:
```
df2.isna().sum() #gives the number of empty values per column
```

Out[260]:
```
genre_ids            0
id                   0
original_language    0
original_title       0
popularity           0
release_date         0
title                0
vote_average         0
vote_count           0
movie                0
production_budget    0
domestic_gross       0
worldwide_gross      0
dtype: int64
```

There were no missing values in the columns.

In [261]:
```
df2.isna().sum().sum() #gives number of empty values in entires data
```

Out[261]:
```
0
```

There were no missing values.

In [262]:
```
duplicate_rows= df2[df2.duplicated()] #checking for duplicated rows
duplicate_rows
```

Out[262]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	movie	pr
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174	Toy Story	
86	[18, 10749]	46705	en	Blue Valentine	8.994	2010-12-29	Blue Valentine	6.9	1677	Blue Valentine	
112	[35, 18]	46829	en	Barney's Version	7.357	2011-01-14	Barney's Version	7.2	210	Barney's Version	
131	[18]	59728	en	The 5th Quarter	2.142	2011-03-25	The 5th Quarter	4.7	15	The 5th Quarter	
145	[18, 36, 10752]	387	de	Das Boot	16.554	1982-02-10	Das Boot	8.1	981	Das Boot	
...	...	...	...	...	...	...	...	...	...	...	
1248	[35, 18, 878]	301337	en	Downsizing	10.682	2017-12-22	Downsizing	5.1	1887	Downsizing	
1257	[18, 12]	407890	en	Lean on Pete	9.307	2018-04-06	Lean on Pete	6.9	133	Lean on Pete	
1263	[18]	300687	en	Same Kind of Different as Me	8.756	2017-10-20	Same Kind of Different as Me	6.8	103	Same Kind of Different as Me	
1266	[28, 35]	398177	en	Just Getting Started	8.459	2017-12-08	Just Getting Started	4.9	94	Just Getting Started	
1272	[18]	392982	en	Marshall	7.879	2017-10-13	Marshall	7.3	257	Marshall	

130 rows × 13 columns

In [263]:
```
df2=df2.drop_duplicates() #displays data after dropping duplicated rows
df2
```

Out[263]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	movie	produ
0	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	How to Train Your Dragon	
1	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368	Iron Man 2	
2	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174	Toy Story	
4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186	Inception	
5	[28, 12, 14, 878]	19995	en	Avatar	26.526	2009-12-18	Avatar	7.4	18676	Avatar	
...	...	...	...	...	...	...	...	...	...	...	
1396	[53, 18, 27]	510284	en	Braid	5.972	2019-02-01	Braid	6.4	18	Braid	
1397	[18, 10752]	514407	en	Indivisible	5.599	2018-10-26	Indivisible	5.4	13	Indivisible	
1398	[18, 28, 80]	547590	en	El Chicano	5.274	2019-05-03	El Chicano	9.0	1	El Chicano	
1399	[18, 35, 28, 80]	506971	ur	Teefa in Trouble	4.486	2018-07-20	Teefa in Trouble	7.6	11	Teefa in Trouble	
1400	[28, 12, 16]	332718	en	Bilal: A New Breed of Hero	2.707	2018-02-02	Bilal: A New Breed of Hero	6.8	54	Bilal: A New Breed of Hero	

1271 rows × 13 columns

In [264]:
```
vote_count = df2['vote_count'].max() #checking the highest number of votes
vote_count
```

Out[264]: 22186

In [265]:
```
average_vote_count = df2['vote_count'].mean() #getting the mean of the vote count
average_vote_count
```

Out[265]: 2164.369000786782

In [266]:
```
mode_vote_count = df2['vote_count'].mode() #getting the mode of the vote count
mode_vote_count
```

Out[266]:
```
0    6
Name: vote_count, dtype: int64
```

In [267]:
```
median_vote_count = df2['vote_count'].median() #getting the median of the vote count
median_vote_count
```

Out[267]: 1058.0

In [268]:
```
df2 = df2[df2['vote_count'] >= 1058] #dropping the rows with an average vote count less than 1058
df2.reset_index(drop=True, inplace=True)
df2
```

Out[268]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	movie	produc
0	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	How to Train Your Dragon	$
1	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368	Iron Man 2	$
2	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174	Toy Story	
3	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186	Inception	$
4	[28, 12, 14, 878]	19995	en	Avatar	26.526	2009-12-18	Avatar	7.4	18676	Avatar	$
...	...	...	...	...	...	...	...	...	...	...	
631	[9648, 27, 53]	406563	en	Insidious: The Last Key	16.017	2018-01-05	Insidious: The Last Key	6.1	1306	Insidious: The Last Key	
632	[35, 18, 10749]	449176	en	Love, Simon	15.608	2018-03-16	Love, Simon	8.2	3165	Love, Simon	
633	[53, 27]	460019	en	Truth or Dare	14.354	2018-04-13	Truth or Dare	6.0	2005	Truth or Dare	
634	[10752, 18, 36, 28]	429351	en	12 Strong	13.183	2018-01-19	12 Strong	5.6	1312	12 Strong	
635	[12, 878, 10751, 14]	407451	en	A Wrinkle in Time	12.529	2018-03-09	A Wrinkle in Time	5.0	1073	A Wrinkle in Time	$

636 rows × 13 columns

In [269]:
```
vote_average = df2['vote_average'].max() #the highest vote average
vote_average
```

Out[269]: 8.4

In [270]:
```
df2_sort = df2.sort_values(by=['vote_average'], ascending=True) #sorting to get the least vote_average
df2_sort
```

Out[270]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	movie	produ
93	[35]	71880	en	Jack and Jill	11.277	2011-11-11	Jack and Jill	4.1	1124	Jack and Jill	
420	[53]	241251	en	The Boy Next Door	7.062	2015-01-23	The Boy Next Door	4.4	1382	The Boy Next Door	
375	[28, 12, 878]	166424	en	Fantastic Four	16.360	2015-08-07	Fantastic Four	4.4	3837	Fantastic Four	
233	[35]	87818	en	Movie 43	12.813	2013-01-25	Movie 43	4.4	1332	Movie 43	
15	[28, 12, 10751, 14]	10196	en	The Last Airbender	16.595	2010-07-01	The Last Airbender	4.6	2143	The Last Airbender	
...	...	...	...	...	...	...	...	...	...	...	
590	[18, 35]	490132	en	Green Book	36.284	2018-11-16	Green Book	8.3	3499	Green Book	
579	[12, 28, 14]	299536	en	Avengers: Infinity War	80.773	2018-04-27	Avengers: Infinity War	8.3	13948	Avengers: Infinity War	
3	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186	Inception	
270	[18, 10402]	244786	en	Whiplash	28.784	2014-10-10	Whiplash	8.4	7908	Whiplash	
131	[18, 80]	311	en	Once Upon a Time in America	17.717	1984-06-01	Once Upon a Time in America	8.4	2243	Once Upon a Time in America	

636 rows × 13 columns

In [271]:
```
df2_sort = df2.sort_values(by=['vote_average'], ascending=False) #sorting values from the highest vote average
df2_sort
```

Out[271]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	movie	produ
131	[18, 80]	311	en	Once Upon a Time in America	17.717	1984-06-01	Once Upon a Time in America	8.4	2243	Once Upon a Time in America	
270	[18, 10402]	244786	en	Whiplash	28.784	2014-10-10	Whiplash	8.4	7908	Whiplash	
579	[12, 28, 14]	299536	en	Avengers: Infinity War	80.773	2018-04-27	Avengers: Infinity War	8.3	13948	Avengers: Infinity War	
3	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186	Inception	
590	[18, 35]	490132	en	Green Book	36.284	2018-11-16	Green Book	8.3	3499	Green Book	
...	...	...	...	...	...	...	...	...	...	...	
15	[28, 12, 10751, 14]	10196	en	The Last Airbender	16.595	2010-07-01	The Last Airbender	4.6	2143	The Last Airbender	
375	[28, 12, 878]	166424	en	Fantastic Four	16.360	2015-08-07	Fantastic Four	4.4	3837	Fantastic Four	
420	[53]	241251	en	The Boy Next Door	7.062	2015-01-23	The Boy Next Door	4.4	1382	The Boy Next Door	
233	[35]	87818	en	Movie 43	12.813	2013-01-25	Movie 43	4.4	1332	Movie 43	
93	[35]	71880	en	Jack and Jill	11.277	2011-11-11	Jack and Jill	4.1	1124	Jack and Jill	

636 rows × 13 columns

In [272]:
```
# Convert columns to numeric for calculations
df2['worldwide_gross'] = pd.to_numeric(df2['worldwide_gross'].str.replace('$', '').str.replace(',', ''))
df2['production_budget'] = pd.to_numeric(df2['production_budget'].str.replace('$', '').str.replace(',', ''))
df2['domestic_gross'] = pd.to_numeric(df2['domestic_gross'].str.replace('$', '').str.replace(',', ''))

# df2['production_budget'] = pd.to_numeric(df2['production_budget'].str.replace('$', '')
# Calculate domestic ROI
df2['domestic_roi'] = ((df2['domestic_gross'] - df2['production_budget']) / df2['production_budget']) * 100

# Calculate worldwide ROI
df2['worldwide_roi'] = ((df2['worldwide_gross'] - df2['production_budget']) / df2['production_budget']) * 100

# Display the dataset with the calculated ROI
df2[['original_title', 'production_budget', 'worldwide_gross', 'worldwide_roi', 'domestic_roi']]
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14264\1942295363.py:2: FutureWarning: The default value of rege
x will change from True to False in a future version. In addition, single character regular expressions will
*not* be treated as literal strings when regex=True.
  df2['worldwide_gross'] = pd.to_numeric(df2['worldwide_gross'].str.replace('$', '').str.replace(',', ''))
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14264\1942295363.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df2['worldwide_gross'] = pd.to_numeric(df2['worldwide_gross'].str.replace('$', '').str.replace(',', ''))
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14264\1942295363.py:3: FutureWarning: The default value of rege
x will change from True to False in a future version. In addition, single character regular expressions will
*not* be treated as literal strings when regex=True.
  df2['production_budget'] = pd.to_numeric(df2['production_budget'].str.replace('$', '').str.replace(',',
''))
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14264\1942295363.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df2['production_budget'] = pd.to_numeric(df2['production_budget'].str.replace('$', '').str.replace(',',
''))
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14264\1942295363.py:4: FutureWarning: The default value of rege
x will change from True to False in a future version. In addition, single character regular expressions will
*not* be treated as literal strings when regex=True.
  df2['domestic_gross'] = pd.to_numeric(df2['domestic_gross'].str.replace('$', '').str.replace(',', ''))
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14264\1942295363.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df2['domestic_gross'] = pd.to_numeric(df2['domestic_gross'].str.replace('$', '').str.replace(',', ''))
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14264\1942295363.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df2['domestic_roi'] = ((df2['domestic_gross'] - df2['production_budget']) / df2['production_budget']) * 10
0
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14264\1942295363.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df2['worldwide_roi'] = ((df2['worldwide_gross'] - df2['production_budget']) / df2['production_budget']) *
100
```

Out[272]:

	original_title	production_budget	worldwide_gross	worldwide_roi	domestic_roi
0	How to Train Your Dragon	165000000	494870992	199.921813	31.867413
1	Iron Man 2	170000000	621156389	265.386111	83.784312
2	Toy Story	30000000	364545516	1115.151720	539.320777
3	Inception	160000000	835524642	422.202901	82.860122
4	Avatar	425000000	2776345279	553.257713	78.942971
...	...	...	...	...	...
631	Insidious: The Last Key	10000000	167885588	1578.855880	577.453300
632	Love, Simon	10000000	65520633	555.206330	308.263410
633	Truth or Dare	3500000	95127344	2617.924114	1083.171857
634	12 Strong	35000000	71118378	103.195366	30.913466
635	A Wrinkle in Time	103000000	133401882	29.516390	-2.447953

636 rows × 5 columns

In [273]:
```
df2 #checking the data
```

Out[273]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	movie	produc
0	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	How to Train Your Dragon	
1	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368	Iron Man 2	
2	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174	Toy Story	
3	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186	Inception	
4	[28, 12, 14, 878]	19995	en	Avatar	26.526	2009-12-18	Avatar	7.4	18676	Avatar	
...	...	...	...	...	...	...	...	...	...	...	
631	[9648, 27, 53]	406563	en	Insidious: The Last Key	16.017	2018-01-05	Insidious: The Last Key	6.1	1306	Insidious: The Last Key	
632	[35, 18, 10749]	449176	en	Love, Simon	15.608	2018-03-16	Love, Simon	8.2	3165	Love, Simon	
633	[53, 27]	460019	en	Truth or Dare	14.354	2018-04-13	Truth or Dare	6.0	2005	Truth or Dare	
634	[10752, 18, 36, 28]	429351	en	12 Strong	13.183	2018-01-19	12 Strong	5.6	1312	12 Strong	
635	[12, 878, 10751, 14]	407451	en	A Wrinkle in Time	12.529	2018-03-09	A Wrinkle in Time	5.0	1073	A Wrinkle in Time	

636 rows × 15 columns

In [274]:
```
# Sort the DataFrame based on the 'vote_average' column in ascending order
df2_sorted = df2.sort_values(by='vote_average')
df2=df2_sorted
df2
```

Out[274]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	movie	produ
93	[35]	71880	en	Jack and Jill	11.277	2011-11-11	Jack and Jill	4.1	1124	Jack and Jill	
420	[53]	241251	en	The Boy Next Door	7.062	2015-01-23	The Boy Next Door	4.4	1382	The Boy Next Door	
375	[28, 12, 878]	166424	en	Fantastic Four	16.360	2015-08-07	Fantastic Four	4.4	3837	Fantastic Four	
233	[35]	87818	en	Movie 43	12.813	2013-01-25	Movie 43	4.4	1332	Movie 43	
15	[28, 12, 10751, 14]	10196	en	The Last Airbender	16.595	2010-07-01	The Last Airbender	4.6	2143	The Last Airbender	
...	...	...	...	...	...	...	...	...	...	...	
590	[18, 35]	490132	en	Green Book	36.284	2018-11-16	Green Book	8.3	3499	Green Book	
579	[12, 28, 14]	299536	en	Avengers: Infinity War	80.773	2018-04-27	Avengers: Infinity War	8.3	13948	Avengers: Infinity War	
3	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186	Inception	
270	[18, 10402]	244786	en	Whiplash	28.784	2014-10-10	Whiplash	8.4	7908	Whiplash	
131	[18, 80]	311	en	Once Upon a Time in America	17.717	1984-06-01	Once Upon a Time in America	8.4	2243	Once Upon a Time in America	

636 rows × 15 columns

In [275]:
```
sort_popularity = df2.sort_values(by=['popularity'], ascending=True) #sorting the data by popularity
sort_popularity
```

Out[275]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	movie	prod
259	[35]	116741	en	The Internship	0.600	2013-06-07	The Internship	6.2	2631	The Internship	
51	[28, 12, 27, 878]	35791	en	Resident Evil: Afterlife	0.667	2010-09-10	Resident Evil: Afterlife	5.9	2119	Resident Evil: Afterlife	
578	[28, 27, 878]	173897	en	Resident Evil: The Final Chapter	0.844	2017-01-27	Resident Evil: The Final Chapter	5.9	1859	Resident Evil: The Final Chapter	
183	[28, 27, 878]	71679	en	Resident Evil: Retribution	1.325	2012-09-14	Resident Evil: Retribution	5.7	2321	Resident Evil: Retribution	
115	[28, 878, 12]	49538	en	X-Men: First Class	1.447	2011-06-03	X-Men: First Class	7.2	8211	X-Men: First Class	
...	...	...	...	...	...	...	...	...	...	...	
262	[28, 878, 12]	118340	en	Guardians of the Galaxy	49.606	2014-08-01	Guardians of the Galaxy	7.9	17958	Guardians of the Galaxy	
116	[878, 28, 12]	24428	en	The Avengers	50.289	2012-05-04	The Avengers	7.6	19673	The Avengers	
261	[28, 12, 14]	122917	en	The Hobbit: The Battle of the Five Armies	53.783	2014-12-17	The Hobbit: The Battle of the Five Armies	7.3	8392	The Hobbit: The Battle of the Five Armies	
260	[28, 53]	245891	en	John Wick	78.123	2014-10-24	John Wick	7.2	10081	John Wick	
579	[12, 28, 14]	299536	en	Avengers: Infinity War	80.773	2018-04-27	Avengers: Infinity War	8.3	13948	Avengers: Infinity War	

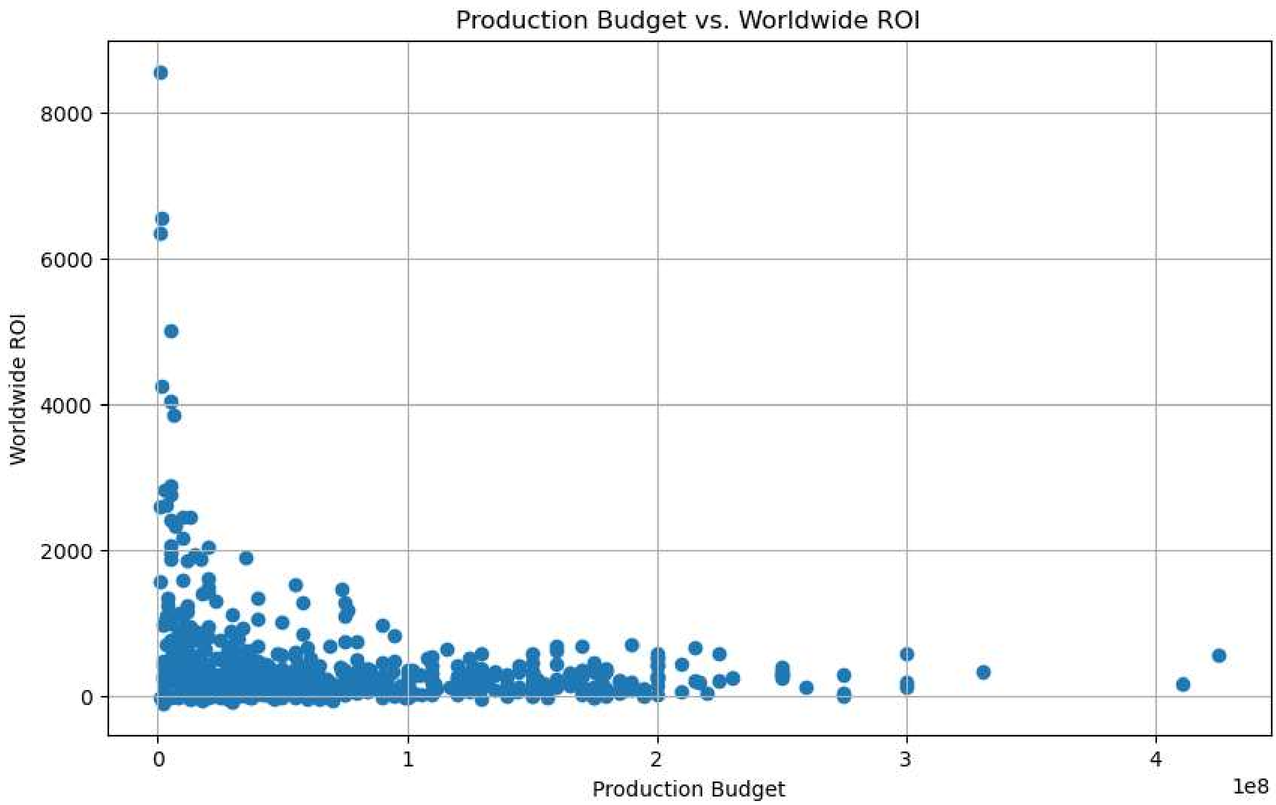
636 rows × 15 columns

DATA VISUALIZATION

The graphical representation of the data. Scatter plots were used for this analysis.

SCATTER PLOT

In [276]:
```
plt.figure(figsize=(10, 6))
plt.scatter(production_budget, worldwide_roi, marker='o')
plt.title('Production Budget vs. Worldwide ROI')
plt.xlabel('Production Budget')
plt.ylabel('Worldwide ROI')
plt.grid(True)
plt.show()
```

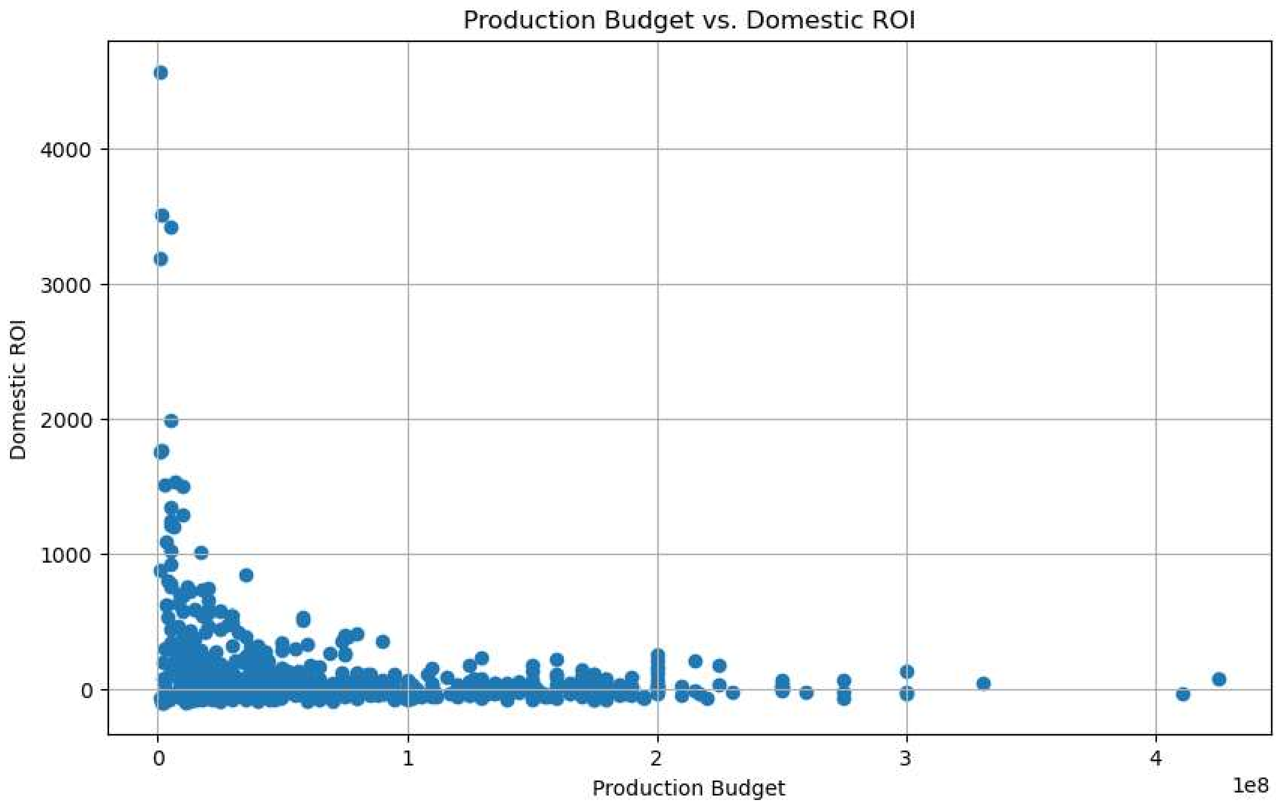


This scatter plot shows a positive correlation between the two variables, suggesting that films with higher production budgets tend to have higher worldwide ROI. However, there is also a lot of variation in the data, suggesting that other factors also influence worldwide ROI

In [277]:
```
# Extract the 'domestic_roi' and 'production_budget' columns
domestic_roi = df2['domestic_roi']
production_budget = df2['production_budget']

# Create a scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(production_budget, domestic_roi, marker='o')
plt.title('Production Budget vs. Domestic ROI')
plt.xlabel('Production Budget')
plt.ylabel('Domestic ROI')
plt.grid(True)

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

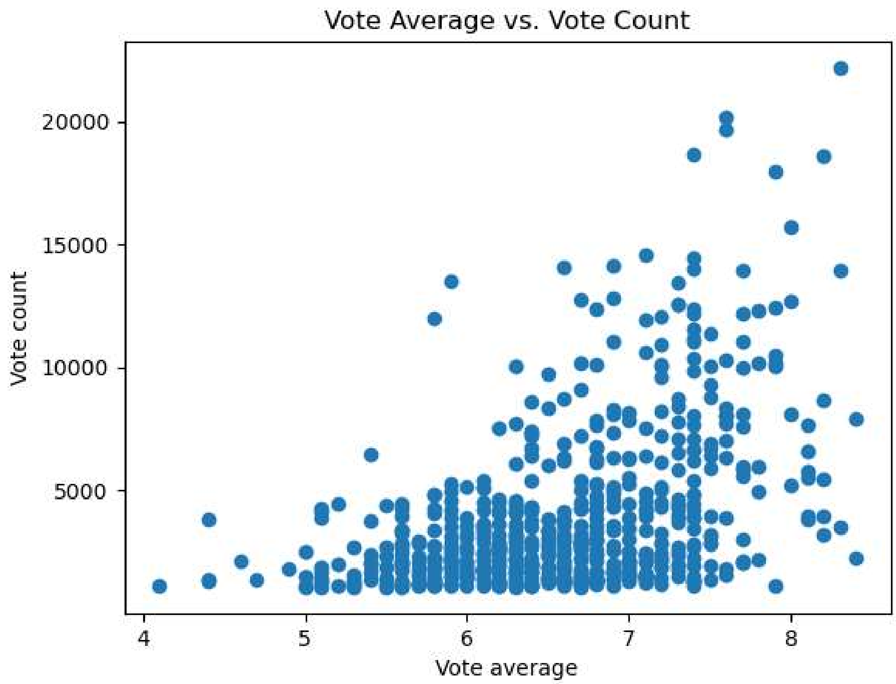


In [278]:
```
vote_average = df2['vote_average']
vote_count = df2['vote_count']

# Create a scatter plot
plt.scatter(vote_average, vote_count, marker='o')

# Add labels and a title
plt.xlabel('Vote average')
plt.ylabel('Vote count')
plt.title('Vote Average vs. Vote Count')

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



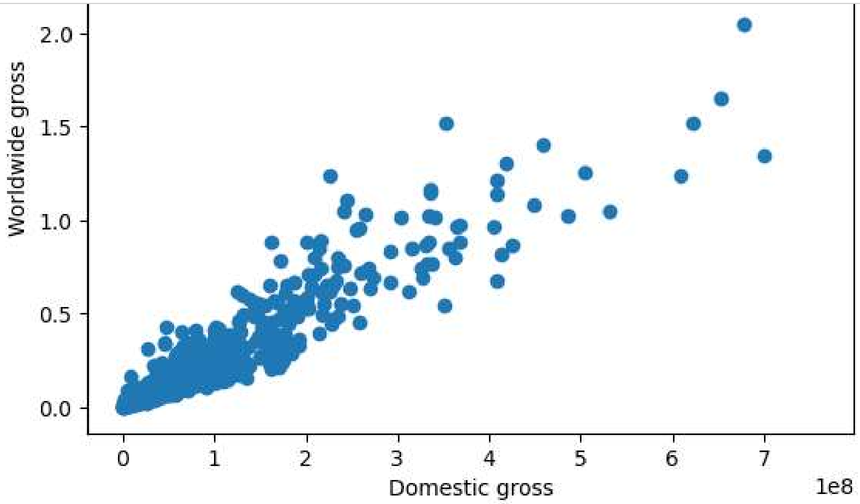
This scatter plot shows a positive correlation between the two variables, suggesting that films with higher vote averages tend to also have higher vote counts. However, there are some outliers, such as films that were less popular with the general public but were popular with a small group of critics.

In [279]:
```
domestic_gross = df2['domestic_gross']
worldwide_gross = df2['worldwide_gross']

# Create a scatter plot
plt.scatter(domestic_gross, worldwide_gross, marker='o')

# Add labels and a title
plt.xlabel('Domestic gross')
plt.ylabel('Worldwide gross')
plt.title('Domestic Gross vs. Worldwide Gross')

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



This scatter plot shows a strong correlation between the two variables, suggesting that films with higher domestic gross tend to also have higher worldwide gross. However, there are some outliers, such as films that were more popular in international markets than in the domestic market.

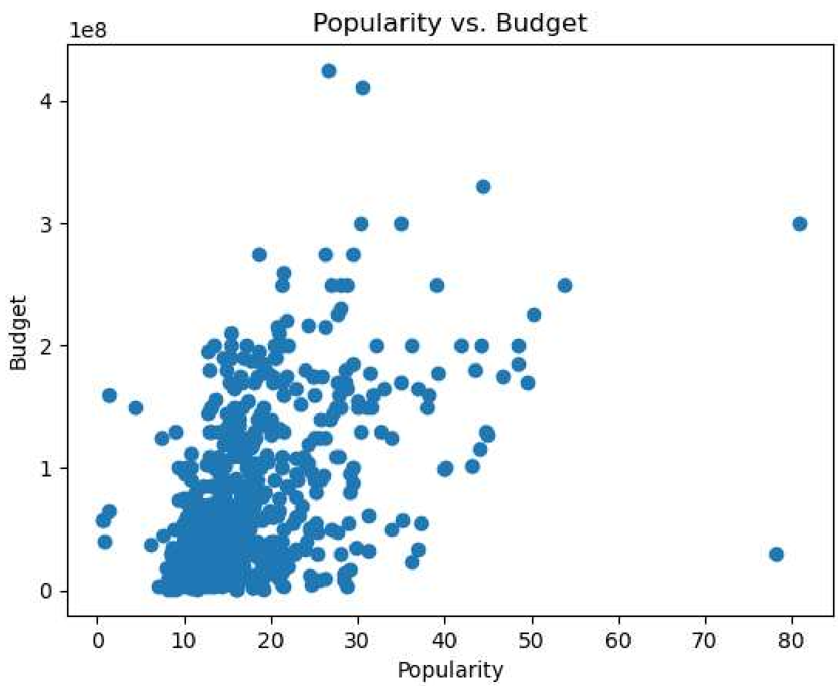
In [280]:

```
# Extract the popularity and budget variables
popularity = df2['popularity']
budget = df2['production_budget']

# Create a scatter plot
plt.scatter(popularity, budget, marker='o')

# Add labels and a title
plt.xlabel('Popularity')
plt.ylabel('Budget')
plt.title('Popularity vs. Budget')

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



The scatter plot shows that movies with a higher production budget are more popular