"* Student name: Joy Wangari\n",

```
"* Student pace: part time\n",
"* Scheduled project review date/time: 16/04/2023\n",
"* Instructor name: Noah Kandie\n",
"* Blog post URL:\n"
```

In [ ]:
```
import pandas as pd
```

In [2]:
```
import sqlite3
```

In [3]:
```
import matplotlib.pyplot as plt
```

In [4]:
```
import seaborn as sns
```

First Step is importing all important libraries

```
Then we load all the data sets
```

In [14]:
```
#Load the dataframes
box_office_m_df = pd.read_csv("C:/Users/user/Downloads/dt/bom.movie_gross.csv")
```

In [13]:
```
rotten_t_movies_df = pd.read_csv("C:/Users/user/Downloads/dt/rt.movie_info.tsv", delimiter=
```

In [12]:
```
rotten_t_reviews_df = pd.read_csv("C:/Users/user/Downloads/dt/rt.reviews.tsv", delimiter='\
```

In [11]:
```
tmdb_movies_df = pd.read_csv("C:/Users/user/Downloads/dt/tmdb.movies.csv")
```

In [15]:
```
t_numbers_budget_df = pd.read_csv("C:/Users/user/Downloads/dt/tn.movie_budgets.csv")
```

In [112]:

```
t_numbers_budget_df
```

Out[112]:

	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

In [216]:

```
t_numbers_budget_df.describe()
```

Out[216]:

	id
count	5782.000000
mean	50.372363
std	28.821076
min	1.000000
25%	25.000000
50%	50.000000
75%	75.000000
max	100.000000

In [23]:

```
t_numbers_budget_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   id                 5782 non-null   int64 
 1   release_date       5782 non-null   object
 2   movie              5782 non-null   object
 3   production_budget  5782 non-null   object
 4   domestic_gross     5782 non-null   object
 5   worldwide_gross    5782 non-null   object
dtypes: int64(1), object(5)
memory usage: 271.2+ KB
```

In [220]:
```
t_numbers_budget_df.isnull().sum()
#this shows there are no null rows
```

Out[220]:
```
id                   0
release_date         0
movie                0
production_budget    0
domestic_gross       0
worldwide_gross      0
dtype: int64
```

We need to find if there is any correlation between budget and gross income

In [24]:
```
# Grouping by Budget
t_numbers_grouped_by_budget = t_numbers_budget_df.groupby('production_budget').apply(lambda
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_2344\3882873739.py:2: FutureWarning: Not prepen
ding group keys to the result index of transform-like apply. In the future, the group keys
will be included in the index, regardless of whether the applied function returns a like-i
ndexed object.
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  t_numbers_grouped_by_budget = t_numbers_budget_df.groupby('production_budget').apply(lam
bda x: x[['movie', 'domestic_gross', 'worldwide_gross']])
```

In [225]:
```
print(t_numbers_grouped_by_budget.to_string(index=False))
```

```
                                                                     The Meg
$145,443,742    $529,530,715
                                                           Edge of Tomorrow
$100,206,256    $370,541,256
                                                             Captain Marvel
$426,525,952  $1,123,061,550
                                                            The Jungle Book
$364,001,123    $962,854,547
                                                                 Inside Out
$356,461,711    $854,235,992
                                                     Spider-Man: Homecoming
$334,201,140    $880,166,350
                                                              Suicide Squad
$325,100,054    $746,059,887
                                                                         Up
$293,004,164    $731,463,377
                                                                       Coco
$209,726,015    $798,008,101
                                                  Ralph Breaks The Internet
$201,091,711    $524,283,695
```

In [227]:
```
t_numbers_grouped_by_budget.describe()
```

Out[227]:

	movie	domestic_gross	worldwide_gross
count	5782	5782	5782
unique	5698	5164	5356
top	Halloween	$0	$0
freq	3	548	367

In [246]:
```
print(t_numbers_grouped_by_worldwide_gross.to_string(index=False))
```

```
$200,000,000   $233,921,534
                                             The Amazing Spider-Man 2
$200,000,000   $202,853,933
                                                               Cars 2
$200,000,000   $191,450,875
                                                         Tron: Legacy
$200,000,000   $172,062,763
                                                                 2012
$200,000,000   $166,112,167
                           Fantastic Beasts: The Crimes of Grindelwald
$200,000,000   $159,555,901
                                                 Terminator Salvation
$200,000,000   $125,322,469
                                                        Green Lantern
$200,000,000   $116,601,172
                                      Prince of Persia: Sands of Time
$200,000,000    $90,759,676
                                      Transformers: Dark of the Moon
$195,000,000   $352,390,543
                                                            The Mummy
```

In [228]:
```
t_numbers_grouped_by_worldwide_gross.describe()
```

Out[228]:

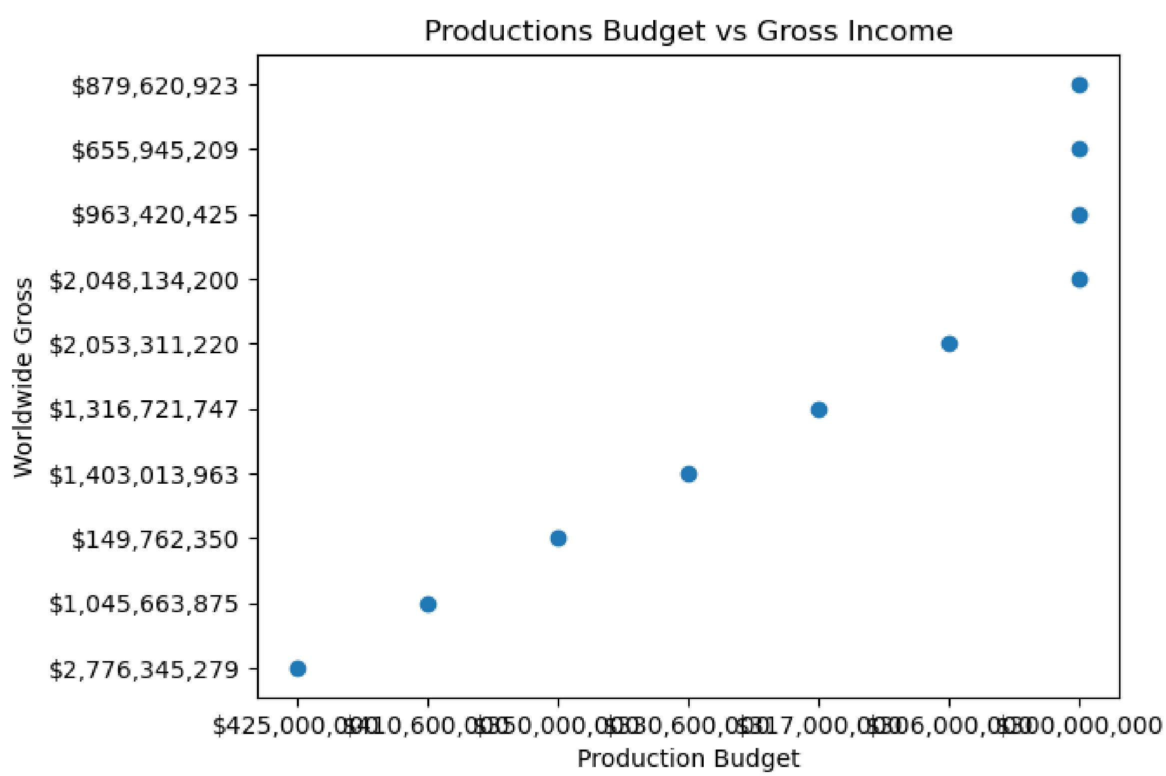
	movie	production_budget	domestic_gross
count	5782	5782	5782
unique	5698	509	5164
top	Halloween	$20,000,000	$0
freq	3	231	548

In [83]:
```
t_numbers_budget_df = t_numbers_budget_df.head(10)
t_numbers_budget_df
```

Out[83]:

	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
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	$306,000,000	$936,662,225	$2,053,311,220
6	7	Apr 27, 2018	Avengers: Infinity War	$300,000,000	$678,815,482	$2,048,134,200
7	8	May 24, 2007	Pirates of the Caribbean: At Worldâ□□s End	$300,000,000	$309,420,425	$963,420,425
8	9	Nov 17, 2017	Justice League	$300,000,000	$229,024,295	$655,945,209
9	10	Nov 6, 2015	Spectre	$300,000,000	$200,074,175	$879,620,923

In [85]:
```
x = t_numbers_budget_df['production_budget'].head(20)
plt.scatter(x, t_numbers_budget_df['worldwide_gross'])
plt.title("Productions Budget vs Gross Income")
plt.xlabel('Production Budget')
plt.ylabel('Worldwide Gross')
plt.show()
```



It is evident that they are directly proportional .The higher the budget, the higher the income

Let's explore the bom.movie_gross.csv data frame

In [37]:
```
# display box_office_movies
box_office_m_df.head(5)
```

Out[37]:

	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 [38]:
```
box_office_m_df.describe()
```

Out[38]:

	domestic_gross	year
count	3.359000e+03	3387.000000
mean	2.874585e+07	2013.958075
std	6.698250e+07	2.478141
min	1.000000e+02	2010.000000
25%	1.200000e+05	2012.000000
50%	1.400000e+06	2014.000000
75%	2.790000e+07	2016.000000
max	9.367000e+08	2018.000000

In [39]:
```
print(box_office_m_df.dtypes)
```

```
title              object
studio             object
domestic_gross    float64
foreign_gross      object
year                int64
dtype: object
```

To be able to use the foreign_gross we need to convert the data type

In [42]:
```
# Convert the worldwide_gross' columns to numeric types

box_office_m_df['foreign_gross'] = pd.to_numeric(box_office_m_df['foreign_gross'], errors='
```

In [43]:
```
print(box_office_m_df.dtypes)
```

```
title              object
studio             object
domestic_gross    float64
foreign_gross     float64
year                int64
dtype: object
```

In [44]:
```
box_office_m_df.isnull().sum()
```

Out[44]:
```
title                0
studio               5
domestic_gross      28
foreign_gross     1355
year                 0
dtype: int64
```

In [45]:
```
box_office_m_df["foreign_gross"].value_counts()
```

Out[45]:
```
1200000.0      23
1100000.0      14
1900000.0      12
4200000.0      12
2500000.0      11
               ..
96300000.0      1
138300000.0     1
63100000.0      1
118100000.0     1
30000.0         1
Name: foreign_gross, Length: 1199, dtype: int64
```

From the above it is safe to drop the rows with no foreign gross entry since it will skew my results

In [49]:
```
box_office_m_df = box_office_m_df.dropna()
box_office_m_df
```

Out[49]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000.0	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000.0	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000.0	2010
3	Inception	WB	292600000.0	535700000.0	2010
4	Shrek Forever After	P/DW	238700000.0	513900000.0	2010
...	...	...	...	...	...
3275	I Still See You	LGF	1400.0	1500000.0	2018
3286	The Catcher Was a Spy	IFC	725000.0	229000.0	2018
3309	Time Freak	Grindstone	10000.0	256000.0	2018
3342	Reign of Judges: Title of Liberty - Concept Short	Darin Southa	93200.0	5200.0	2018
3353	Antonio Lopez 1970: Sex Fashion & Disco	FM	43200.0	30000.0	2018

2002 rows × 5 columns

In [50]:
```
#checking for any remaining null values
box_office_m_df.isnull().sum()
```

Out[50]:
```
title             0
studio            0
domestic_gross    0
foreign_gross     0
year              0
dtype: int64
```

Next Step in EDA is to check for any duplicate values

In [52]:
```
box_office_m_df.duplicated().sum()
```

Out[52]: 0

In [56]:
```
box_office_m_df['studio'].unique()
```

Out[56]:
```
array(['BV', 'WB', 'P/DW', 'Sum.', 'Par.', 'Uni.', 'Fox', 'Wein.', 'Sony',
       'FoxS', 'SGem', 'WB (NL)', 'LGF', 'MBox', 'CL', 'W/Dim.', 'CBS',
       'Focus', 'MGM', 'Over.', 'Mira.', 'IFC', 'CJ', 'NM', 'SPC', 'ParV',
       'Gold.', 'JS', 'RAtt.', 'Magn.', 'Free', '3D', 'UTV', 'Rela.',
       'Zeit.', 'Anch.', 'PDA', 'Lorb.', 'App.', 'Drft.', 'Osci.', 'IW',
       'Rog.', 'Eros', 'Relbig.', 'Viv.', 'Hann.', 'Strand', 'NGE',
       'Scre.', 'Kino', 'Abr.', 'CZ', 'ATO', 'First', 'GK', 'FInd.',
       'NFC', 'TFC', 'Pala.', 'Imag.', 'NAV', 'Arth.', 'CLS', 'Mont.',
       'Olive', 'CGld', 'FOAK', 'IVP', 'Yash', 'ICir', 'WOW', 'FM', 'FD',
       'Vari.', 'TriS', 'ORF', 'IM', 'Elev.', 'Cohen', 'NeoC', 'Jan.',
       'MNE', 'Trib.', 'Vita.', 'Rocket', 'OMNI/FSR', 'KKM', 'Argo.',
       'Libre', 'FRun', 'P4', 'KC', 'MPFT', 'Icar.', 'AGF', 'NYer',
       'LG/S', 'WHE', 'WGUSA', 'MPI', 'RTWC', 'FIP', 'RF', 'KL', 'ArcEnt',
       'PalUni', 'EpicPics', 'EOne', 'AF', 'LD', 'TFA', 'WAMCR', 'PM&E',
       'A24', 'Distrib.', 'Imax', 'PH', 'Da.', 'E1', 'Shout!', 'SV', 'CE',
       'VPD', 'KE', 'Outs', 'HTR', 'DR', 'Ampl.', 'CP', 'BGP', 'Crnth',
       'LGP', 'EC', 'FUN', 'STX', 'BG', 'PFR', 'BST', 'FCW', 'U/P', 'UHE',
       'FR', 'Orch.', 'PBS', 'ITL', 'AR', 'JBG', 'BH Tilt', 'Zee', 'HC',
       'GrtIndia', 'PNT', 'Neon', 'Good Deed', 'ParC', 'Amazon', 'BBC',
       'Affirm', 'Annapurna', 'MOM', 'Studio 8', 'Global Road',
       'Trafalgar', 'ENTMP', 'Greenwich', 'Spanglish', 'Blue Fox',
       'Aviron', 'VE', 'Grindstone', 'Darin Southa'], dtype=object)
```

In [63]:
```
# group the data by studio and calculate the mean of domestic_gross
mean_domestic_gross = box_office_m_df.groupby('studio')['domestic_gross'].mean()
mean_domestic_gross
```

Out[63]:
```
studio
3D       6.100000e+06
A24      1.370825e+07
AF       5.775000e+05
AGF      1.580000e+04
AR       3.500000e+05
             ...
WOW      3.080000e+04
Wein.    2.133068e+07
Yash     3.745633e+06
Zee      1.100000e+06
Zeit.    3.458400e+05
Name: domestic_gross, Length: 172, dtype: float64
```

In [69]:
```
# group the data by studio and calculate the mean of domestic_gross
studio_foreign_gross = box_office_m_df.groupby('studio')['foreign_gross'].mean().apply(lamb
studio_foreign_gross
```

Out[69]:
```
studio
3D       $10M
A24      $13M
AF        $2M
AGF       $0M
AR       $58M
         ...
WOW       $0M
Wein.    $38M
Yash     $45M
Zee       $1M
Zeit.     $4M
Name: foreign_gross, Length: 172, dtype: object
```

In [127]:
```
#load data from sqlite instance
path = "C:/Users/user/Documents/dsc-phase-1-project-v2-4/im.db/im.db"
```

In [128]:
```
#connect to the db
conn = sqlite3.connect(path)
```

In [129]:
```
#create a cursor object for the db
cursor = conn.cursor()
```

In [130]:
```
# Execute an SQL query to fetch data from the movie_basics table
cursor.execute("SELECT movie_id, primary_title, genres, runtime_minutes FROM movie_basics WH
movie_basics_data = cursor.fetchall()
```

In [131]:
```
print(movie_basics_data)
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

In [132]:
```
# Execute an SQL query to fetch data from the movie_ratings table
cursor.execute("SELECT * FROM movie_ratings")
movie_ratings_data = cursor.fetchall()
```

In [133]:
```python
print(movie_ratings_data)
```

```
1437354', 5.5, 438), ('tt1438214', 5.0, 102), ('tt1440161', 6.3, 26441), ('tt1450651',
5.5, 74), ('tt1453262', 6.3, 111), ('tt1458408', 8.2, 17), ('tt1458730', 7.3, 28), ('t
t1460646', 3.2, 239), ('tt1483386', 7.2, 676), ('tt1486652', 5.0, 5), ('tt1489889', 6.
3, 138872), ('tt1490753', 6.8, 72), ('tt1490785', 5.8, 5658), ('tt1491603', 3.6, 138
8), ('tt1493816', 2.9, 30), ('tt1496374', 7.7, 10), ('tt1500694', 4.6, 121), ('tt15024
22', 6.5, 332), ('tt1506998', 5.7, 379), ('tt1510926', 5.9, 345), ('tt1511354', 5.6, 3
9), ('tt1511362', 8.0, 9), ('tt1515941', 8.1, 36), ('tt1517260', 5.9, 105633), ('tt151
7506', 5.2, 497), ('tt1519640', 6.1, 785), ('tt1520956', 4.5, 1420), ('tt1521223', 6.
9, 457), ('tt1526284', 5.6, 3793), ('tt1526616', 8.6, 7), ('tt1527721', 8.0, 15), ('tt
1529292', 6.4, 554), ('tt1531683', 8.5, 15), ('tt1533749', 6.9, 17777), ('tt1534834',
8.2, 10), ('tt1537485', 7.7, 10), ('tt1539146', 6.3, 21), ('tt1540995', 7.2, 17), ('tt
1543004', 6.2, 405), ('tt1544589', 8.0, 8), ('tt1546401', 5.1, 26), ('tt1546985', 7.2,
5), ('tt1550643', 5.6, 10), ('tt1550902', 7.0, 600), ('tt1555440', 6.4, 1097), ('tt156
3127', 7.2, 27), ('tt1563712', 6.5, 41), ('tt1565064', 7.7, 771), ('tt1566501', 6.6, 4
202), ('tt1567127', 5.5, 182), ('tt1567611', 7.1, 27), ('tt1570103', 7.4, 18), ('tt157
2169', 5.3, 43), ('tt1572501', 5.8, 502), ('tt1572769', 6.9, 2070), ('tt1578709', 4.3,
323), ('tt1579391', 6.5, 19), ('tt1582482', 5.9, 11), ('tt1582483', 5.3, 25), ('tt1582
567', 7.3, 6), ('tt1583279', 7.2, 22), ('tt1585660', 6.7, 23), ('tt1586001', 7.2, 596
9), ('tt1586516', 7.5, 67), ('tt1587220', 6.4, 18), ('tt1590193', 6.3, 83114), ('tt159
0231', 7.8, 9), ('tt1590970', 6.4, 159), ('tt1591123', 5.5, 217), ('tt1592583', 4.2, 3
```

In [134]:
```python
# Join the two tables based on the movie_id column
data = []
for row in movie_ratings_data:
    for row2 in movie_basics_data:
        if row[0] == row2[0]:
            data.append((row2[1], row2[2], row[1], row[2]))
            break
```

In [135]:
```python
data
```

Out[135]:
```
[('Laiye Je Yaarian', 'Romance', 8.3, 31),
 ('Borderless', 'Documentary', 8.9, 559),
 ('Vanquisher', 'Action,Adventure,Sci-Fi', 4.2, 148),
 ('Little Secret', 'Biography,Drama', 7.7, 1293),
 ('Dust Radio: A Film About Chris Whitley', 'Documentary,Drama', 8.2, 5),
 ('Zoolander 2', 'Comedy', 4.7, 59914),
 ('Killer Ink', 'Horror', 5.6, 64),
 ('Break Clause', 'Drama,Thriller', 8.0, 20),
 ('Lustrum', 'Documentary', 5.9, 14),
 ('The Little Prince', 'Action', 8.3, 6),
 ('Senses 3&4', 'Drama', 7.3, 7),
 ('Chopsticks', 'Comedy,Drama', 6.5, 1394),
 ('Q Ball', 'Documentary', 7.0, 15),
 ('Geceyarisi, Türkiye zamani', 'Crime,Drama', 4.3, 12),
 ('J Revolusi', 'Action', 5.5, 130),
 ('The 3rd Eye', 'Horror,Thriller', 5.0, 670),
 ('Spyder', 'Action,Thriller', 6.8, 7930),
 ('Mules', 'Documentary', 6.9, 117),
 ('Bloody Murder', 'Thriller', 3.7, 26),
```

In [123]:
```python
# Convert the data to a pandas DataFrame
im_db_df = pd.DataFrame(data, columns=['title', 'genres', 'rating', 'num_votes'])
```

In [124]:
```
im_db_df
```

Out[124]:

	title	genres	rating	num_votes
0	Laiye Je Yaarian	Romance	8.3	31
1	Borderless	Documentary	8.9	559
2	Vanquisher	Action,Adventure,Sci-Fi	4.2	148
3	Little Secret	Biography,Drama	7.7	1293
4	Dust Radio: A Film About Chris Whitley	Documentary,Drama	8.2	5
...	...	...	...	...
27135	Caisa	Documentary	8.1	25
27136	Code Geass: Lelouch of the Rebellion - Glorifi...	Action,Animation,Sci-Fi	7.5	24
27137	Sisters	Action,Drama	4.7	14
27138	The Projectionist	Documentary	7.0	5
27139	Sathru	Thriller	6.3	128

27140 rows × 4 columns

In [81]:
```
im_db_df.describe()
```

Out[81]:

	rating	num_votes
count	27140.000000	27140.000000
mean	6.411013	2509.276971
std	1.517132	20897.206511
min	1.000000	5.000000
25%	5.500000	14.000000
50%	6.600000	49.000000
75%	7.500000	259.000000
max	10.000000	820847.000000

In [82]:
```
#check for null values
im_db_df.isnull().sum()
```

Out[82]:
```
title          0
genres       244
rating         0
num_votes      0
dtype: int64
```

In [93]:
```
# Drop rows where genres column contains null values
clean_im_db_df = im_db_df.dropna(subset=['genres'])
```

In [94]:
```
#check for duplicated values
clean_im_db_df.duplicated().sum()
```

Out[94]:
```
0
```

In [95]:
```
#check the distribution of the rating value
print(clean_im_db_df['rating'].value_counts())
```

```
7.0     829
7.2     816
6.6     780
6.8     778
6.5     774
       ...
1.1      10
10.0     10
9.7      10
1.5       9
9.9       5
Name: rating, Length: 91, dtype: int64
```

In [96]:
```
#check the distribution of the num votes value
print(clean_im_db_df['num_votes'].value_counts())
```

```
6        1006
5         914
7         860
8         780
9         670
         ...
3338        1
5577        1
46265       1
3475        1
4057        1
Name: num_votes, Length: 3614, dtype: int64
```

In [ ]:
```
#Finding the average rating for each genre
```

In [114]:
```
# Group the data by genre and calculate the mean rating for each group
genre_ratings = clean_im_db_df.groupby('genres')['rating'].mean()
```

In [119]:
```
sorted_df = clean_im_db_df.sort_values(by='rating', ascending=False)
```

In [125]:
```
genre_ratings.head(10)
```

Out[125]:
```
genres
Action                          5.873016
Action,Adult,Comedy             3.400000
Action,Adventure                4.991667
Action,Adventure,Animation      6.579710
Action,Adventure,Biography      7.162500
Action,Adventure,Comedy         5.641026
Action,Adventure,Crime          5.568421
Action,Adventure,Documentary    7.800000
Action,Adventure,Drama          5.965152
Action,Adventure,Family         5.185714
Name: rating, dtype: float64
```

In [126]:
```
#Check out the top rated genres
sorted_df.head(10)
```

Out[126]:

	title	genres	rating	num_votes
18045	Fly High: Story of the Disc Dog	Documentary	10.0	7
20730	The Dark Knight: The Ballad of the N Word	Comedy,Drama	10.0	5
19038	Calamity Kevin	Adventure,Comedy	10.0	6
23911	Renegade	Documentary	10.0	20
10860	Pick It Up! - Ska in the '90s	Documentary	10.0	5
2476	Requiem voor een Boom	Documentary	10.0	5
20929	All Around Us	Documentary	10.0	6
16875	Exteriores: Mulheres Brasileiras na Diplomacia	Documentary	10.0	5
17533	Dog Days in the Heartland	Drama	10.0	5
23639	Ellis Island: The Making of a Master Race in A...	Documentary,History	10.0	6

In [127]:
```
#Check out the poorly rated genres
sorted_df.tail(10)
```

Out[127]:

	title	genres	rating	num_votes
8659	Roofied: The Lethal Dose	Drama	1.0	112
18569	Tachiiri kinshi Haittara shinu? Norowareta 5 hen	Horror	1.0	6
22879	Badang	Comedy,Fantasy	1.0	674
18672	Bloody Massacre	Drama,Horror,Thriller	1.0	22
9674	Jak se mori revizori	Comedy	1.0	5
5688	Desu foresuto kyofu no mori 5	Horror	1.0	230
22936	La Scelta Impossibile	Drama	1.0	5
11258	6 elementov vremeni	Adventure,Drama,Sci-Fi	1.0	19
25483	Cherry Blossoms	Drama	1.0	20
24187	Yes, Sir! 7	Comedy,Drama	1.0	96

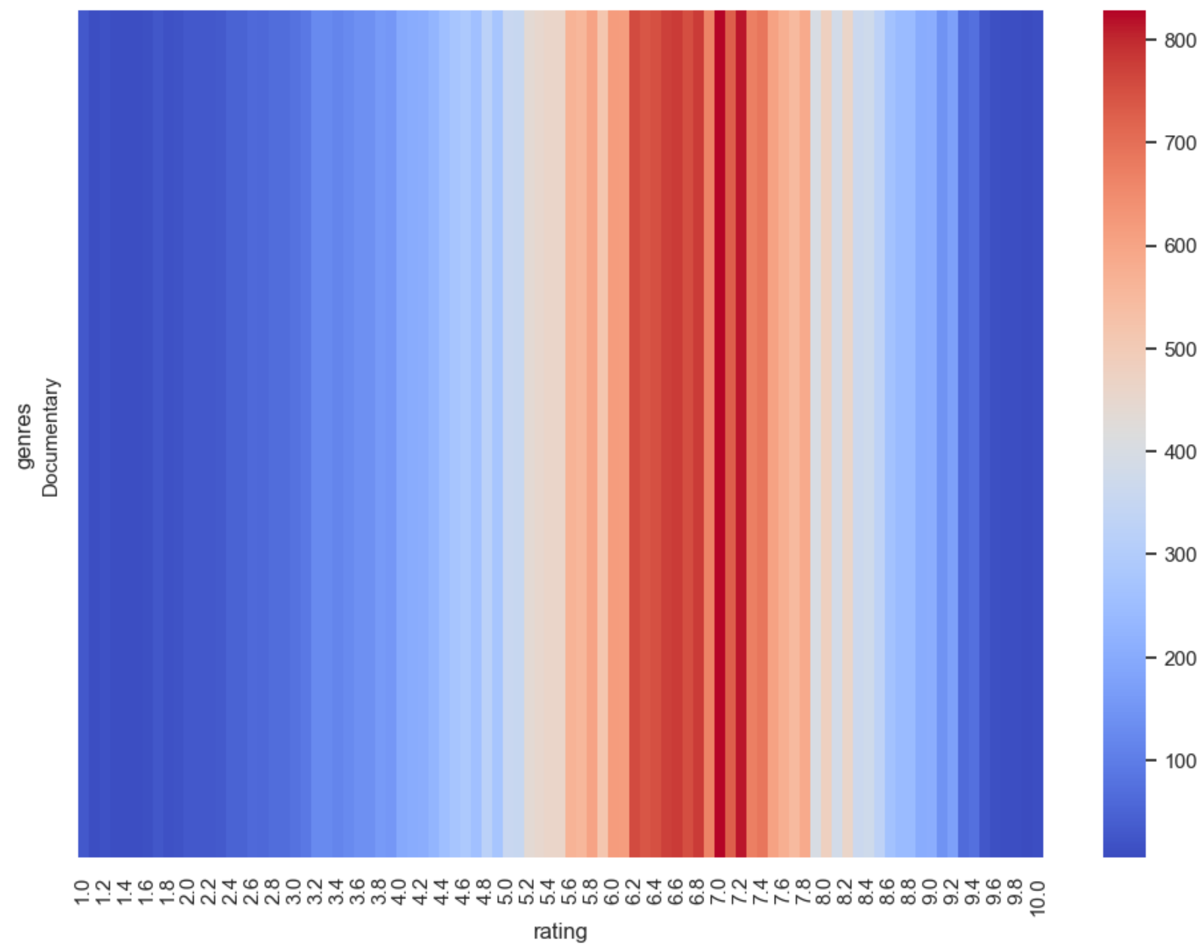
In [128]:
```
title_ratings = clean_im_db_df.groupby('title')['rating'].mean()
```

In [247]:
```
title_ratings
```

Out[247]:
```
genres
Action                          196.658730
Action,Adult,Comedy              28.000000
Action,Adventure               1218.333333
Action,Adventure,Animation    12604.188406
Action,Adventure,Biography     9508.500000
                                   ...
Thriller                        293.427941
Thriller,War                     10.000000
Thriller,Western                 30.000000
War                              81.500000
Western                         263.088235
Name: num_votes, Length: 692, dtype: float64
```

In [251]:
```
# Create a pivot table that groups the data by genre and rating
pivot_table = clean_im_db_df.pivot_table(index='genres', columns='rating', values='title', a

# Create the heatmap
sns.heatmap(pivot_table, cmap='coolwarm')
```

Out[251]: <AxesSubplot:xlabel='rating', ylabel='genres'>



In [131]:
```
genre_votes = clean_im_db_df.groupby('genres')['num_votes'].mean()
```

In [132]:
```
genre_votes
```

Out[132]:
```
genres
Action                         196.658730
Action,Adult,Comedy             28.000000
Action,Adventure              1218.333333
Action,Adventure,Animation   12604.188406
Action,Adventure,Biography    9508.500000
                                  ...
Thriller                       293.427941
Thriller,War                    10.000000
Thriller,Western                30.000000
War                             81.500000
Western                        263.088235
Name: num_votes, Length: 692, dtype: float64
```

In [134]:
```
rating_votes = clean_im_db_df.groupby('rating')['num_votes'].mean()
rating_votes
```

Out[134]:
```
rating
1.0       93.535714
1.1      150.500000
1.2      284.800000
1.3     3739.909091
1.4     1108.300000
           ...
9.6      362.461538
9.7      646.200000
9.8       15.090909
9.9       92.200000
10.0       7.000000
Name: num_votes, Length: 91, dtype: float64
```

In [175]:
```
rating_filter = clean_im_db_df['rating'] >= 10
#genres_filter = clean_im_db_df['']
combined_filter = rating_filter
filtered_data = clean_im_db_df[combined_filter]
filtered_data
```

Out[175]:

	title	genres	rating	num_votes
2476	Requiem voor een Boom	Documentary	10.0	5
10860	Pick It Up! - Ska in the '90s	Documentary	10.0	5
16875	Exteriores: Mulheres Brasileiras na Diplomacia	Documentary	10.0	5
17533	Dog Days in the Heartland	Documentary	10.0	5
18045	Fly High: Story of the Disc Dog	Documentary	10.0	7
19038	Calamity Kevin	Documentary	10.0	6
20730	The Dark Knight: The Ballad of the N Word	Documentary	10.0	5
20929	All Around Us	Documentary	10.0	6
23639	Ellis Island: The Making of a Master Race in A...	Documentary	10.0	6
23911	Renegade	Documentary	10.0	20

In [179]:

```
rating_filter2 = clean_im_db_df['rating'] >= 8
#genres_filter = clean_im_db_df['']
combined_filter2 = rating_filter
filtered_data2 = clean_im_db_df[combined_filter2]
filtered_data2
```

Out[179]:

	title	genres	rating	num_votes
2476	Requiem voor een Boom	Documentary	10.0	5
10860	Pick It Up! - Ska in the '90s	Documentary	10.0	5
16875	Exteriores: Mulheres Brasileiras na Diplomacia	Documentary	10.0	5
17533	Dog Days in the Heartland	Documentary	10.0	5
18045	Fly High: Story of the Disc Dog	Documentary	10.0	7
19038	Calamity Kevin	Documentary	10.0	6
20730	The Dark Knight: The Ballad of the N Word	Documentary	10.0	5
20929	All Around Us	Documentary	10.0	6
23639	Ellis Island: The Making of a Master Race in A...	Documentary	10.0	6
23911	Renegade	Documentary	10.0	20

In [222]:

```
print(clean_im_db_df['genres'].value_counts())
```

```
Documentary    26896
Name: genres, dtype: int64
```

In [176]:

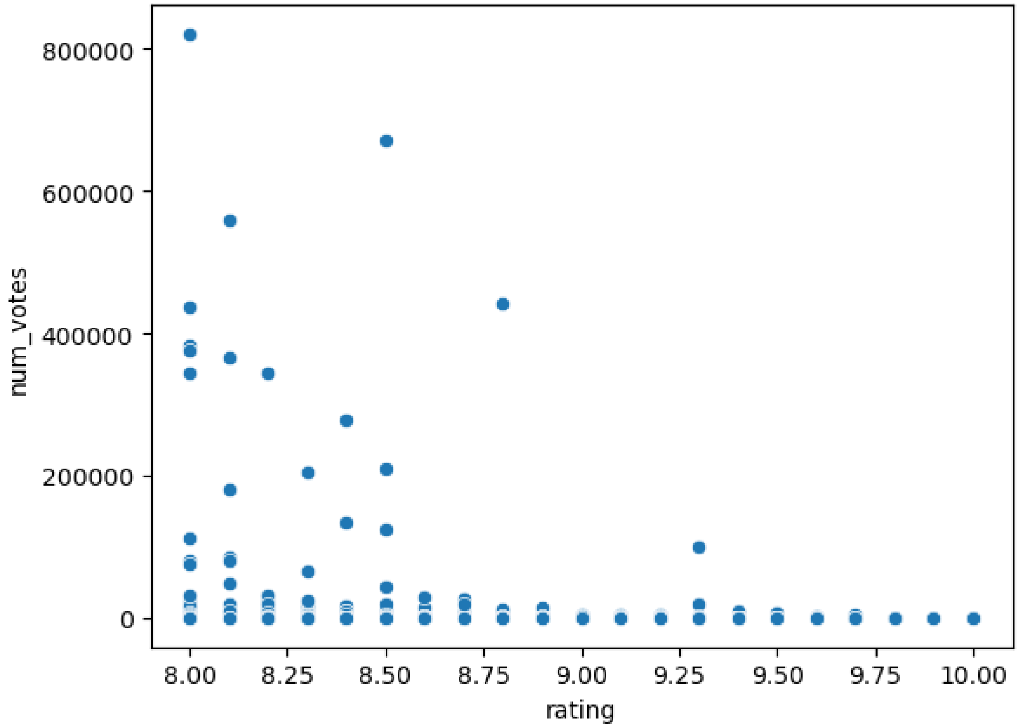
```
filtered_data.count()
```

Out[176]:

```
title        10
genres       10
rating       10
num_votes    10
dtype: int64
```

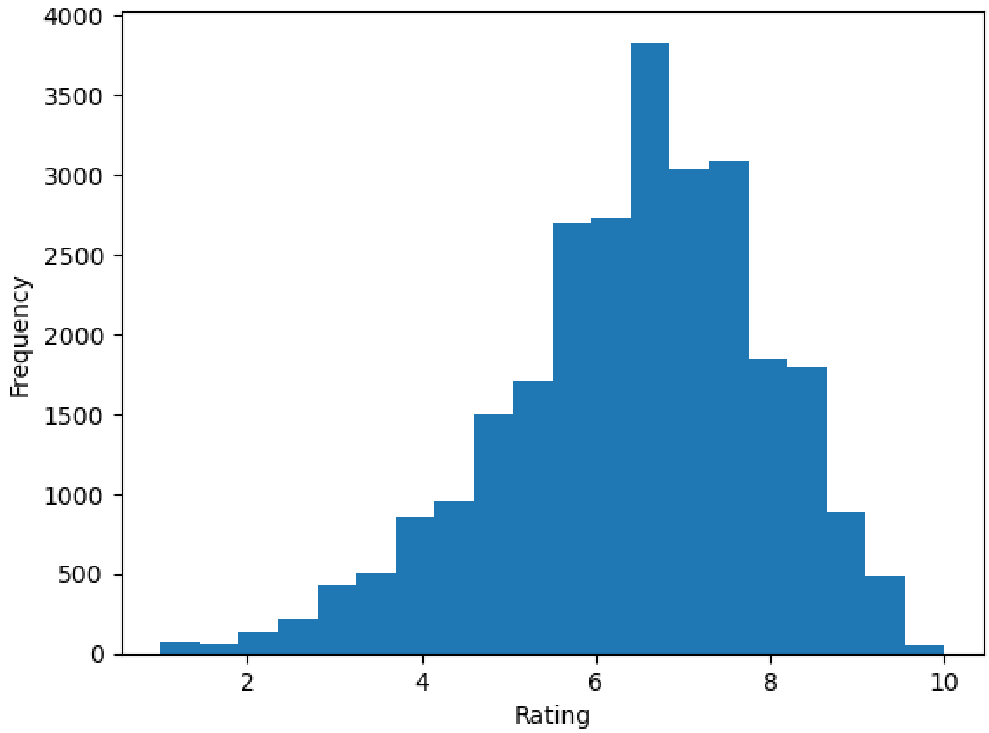
In [164]:

```
# Create a scatter plot
sns.scatterplot(data=filtered_data, x='rating', y='num_votes')

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



In [174]:

```
%matplotlib inline

# Plot a histogram of the 'rating' column in the original dataset
plt.hist(sorted_df['rating'], bins=20)
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```



In [ ]:

```
# Recommendations
# 1. Most top-rated movies to be in the Documentary genre group
# 2. Movies that had the most production_budget also had high worldwide gross returns
# 3. Horror movies had a high vote count but the ratings are poor. Other frequent genres wi
# 4. Movies with the multiple genres to have a good average rating.
# 5. Most Movies made 0$ both domestically and worldwide
```