IMDb Movie Assignment

You have the data for the 100 top-rated movies from the past decade along with various pieces of information about the movie, its actors, and the voters who have rated these movies online. In this assignment, you will try to find some interesting insights into these movies and their voters, using Python.

Task 1: Reading the data

- ### Subtask 1.1: Read the Movies Data.

Read the movies data file provided and store it in a dataframe `movies`.

In [1]:
```
import pandas as pd
df = pd.read_csv('IMDB_dataset.csv')
df
```

Out[1]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1_facebook_likes	actor_2_facebook_likes
0	La La Land	2016	30000000	151101803	Ryan Gosling	Emma Stone	Amiée Conn	14000	19000.0
1	Zootopia	2016	150000000	341268248	Ginnifer Goodwin	Jason Bateman	Idris Elba	2800	28000.0
2	Lion	2016	12000000	51738905	Dev Patel	Nicole Kidman	Rooney Mara	33000	96000.0
3	Arrival	2016	47000000	100546139	Amy Adams	Jeremy Renner	Forest Whitaker	35000	5300.0
4	Manchester by the Sea	2016	9000000	47695371	Casey Affleck	Michelle Williams	Kyle Chandler	518	71000.0
...	...	...	...	...	...	...	...	...	...
95	Whiplash	2014	3300000	13092000	J.K. Simmons	Melissa Benoist	Chris Mulkey	24000	970.0
96	Before Midnight	2013	3000000	8114507	Seamus Davey-Fitzpatrick	Ariane Labed	Athina Rachel Tsangari	140	63.0
97	Star Wars: Episode VII - The Force Awakens	2015	245000000	936662225	Doug Walker	Rob Walker	0	131	12.0
98	Harry Potter and the Deathly Hallows: Part I	2010	150000000	296347721	Rupert Grint	Toby Jones	Alfred Enoch	10000	2000.0
99	Tucker and Dale vs Evil	2010	5000000	223838	Katrina Bowden	Tyler Labine	Chelan Simmons	948	779.0

100 rows × 62 columns

- ### Subtask 1.2: Inspect the Dataframe

Inspect the dataframe for dimensions, null-values, and summary of different numeric columns.

In [2]:
```
# Check the number of rows and columns in the data
df.shape
```

Out[2]:
```
(100, 62)
```

In [3]:
```
#check data types of each column
df.dtypes
```

Out[3]:
```
Title              object
title_year          int64
budget              int64
Gross               int64
actor_1_name       object
                    ...
Votes1000         float64
```

```
VotesUS           float64
VotesnUS          float64
content_rating     object
Country            object
Length: 62, dtype: object
```

In [4]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 62 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Title                   100 non-null    object 
 1   title_year              100 non-null    int64  
 2   budget                  100 non-null    int64  
 3   Gross                   100 non-null    int64  
 4   actor_1_name            100 non-null    object 
 5   actor_2_name            100 non-null    object 
 6   actor_3_name            100 non-null    object 
 7   actor_1_facebook_likes  100 non-null    int64  
 8   actor_2_facebook_likes  99 non-null     float64
 9   actor_3_facebook_likes  98 non-null     float64
 10  IMDb_rating             100 non-null    float64
 11  genre_1                 100 non-null    object 
 12  genre_2                 97 non-null     object 
 13  genre_3                 74 non-null     object 
 14  MetaCritic              95 non-null     float64
 15  Runtime                 100 non-null    int64  
 16  CVotes10                100 non-null    int64  
 17  CVotes09                100 non-null    int64  
 18  CVotes08                100 non-null    int64  
 19  CVotes07                100 non-null    int64  
 20  CVotes06                100 non-null    int64  
 21  CVotes05                100 non-null    int64  
 22  CVotes04                100 non-null    int64  
 23  CVotes03                100 non-null    int64  
 24  CVotes02                100 non-null    int64  
 25  CVotes01                100 non-null    int64  
 26  CVotesMale              100 non-null    int64  
 27  CVotesFemale            100 non-null    int64  
 28  CVotesU18               100 non-null    int64  
 29  CVotesU18M              100 non-null    int64  
 30  CVotesU18F              100 non-null    int64  
 31  CVotes1829              100 non-null    int64  
 32  CVotes1829M             100 non-null    int64  
 33  CVotes1829F             100 non-null    int64  
 34  CVotes3044              100 non-null    int64  
 35  CVotes3044M             100 non-null    int64  
 36  CVotes3044F             100 non-null    int64  
 37  CVotes45A               100 non-null    int64  
 38  CVotes45AM              100 non-null    int64  
 39  CVotes45AF              100 non-null    int64  
 40  CVotes1000              100 non-null    int64  
 41  CVotesUS                100 non-null    int64  
 42  CVotesnUS               100 non-null    int64  
 43  VotesM                  100 non-null    float64
 44  VotesF                  100 non-null    float64
 45  VotesU18                100 non-null    float64
 46  VotesU18M               100 non-null    float64
 47  VotesU18F               100 non-null    float64
 48  Votes1829               100 non-null    float64
 49  Votes1829M              100 non-null    float64
 50  Votes1829F              100 non-null    float64
 51  Votes3044               100 non-null    float64
 52  Votes3044M              100 non-null    float64
 53  Votes3044F              100 non-null    float64
 54  Votes45A                100 non-null    float64
 55  Votes45AM               100 non-null    float64
 56  Votes45AF               100 non-null    float64
 57  Votes1000               100 non-null    float64
 58  VotesUS                 100 non-null    float64
 59  VotesnUS                100 non-null    float64
 60  content_rating          100 non-null    object 
 61  Country                 100 non-null    object 
dtypes: float64(21), int64(32), object(9)
memory usage: 48.6+ KB
```

In [5]:

In [5]:

```
#check for null values
df.isnull().sum()
```

Out[5]:

```
Title              0
title_year         0
budget             0
Gross              0
actor_1_name       0
                  ..
Votes1000          0
VotesUS            0
VotesnUS           0
content_rating     0
Country            0
Length: 62, dtype: int64
```

In [6]:

```
# Check the column-wise info of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 62 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Title                   100 non-null    object
 1   title_year              100 non-null    int64
 2   budget                  100 non-null    int64
 3   Gross                   100 non-null    int64
 4   actor_1_name            100 non-null    object
 5   actor_2_name            100 non-null    object
 6   actor_3_name            100 non-null    object
 7   actor_1_facebook_likes  100 non-null    int64
 8   actor_2_facebook_likes  99 non-null     float64
 9   actor_3_facebook_likes  98 non-null     float64
 10  IMDb_rating             100 non-null    float64
 11  genre_1                 100 non-null    object
 12  genre_2                 97 non-null     object
 13  genre_3                 74 non-null     object
 14  MetaCritic              95 non-null     float64
 15  Runtime                 100 non-null    int64
 16  CVotes10                100 non-null    int64
 17  CVotes09                100 non-null    int64
 18  CVotes08                100 non-null    int64
 19  CVotes07                100 non-null    int64
 20  CVotes06                100 non-null    int64
 21  CVotes05                100 non-null    int64
 22  CVotes04                100 non-null    int64
 23  CVotes03                100 non-null    int64
 24  CVotes02                100 non-null    int64
 25  CVotes01                100 non-null    int64
 26  CVotesMale              100 non-null    int64
 27  CVotesFemale            100 non-null    int64
 28  CVotesU18               100 non-null    int64
 29  CVotesU18M              100 non-null    int64
 30  CVotesU18F              100 non-null    int64
 31  CVotes1829              100 non-null    int64
 32  CVotes1829M             100 non-null    int64
 33  CVotes1829F             100 non-null    int64
 34  CVotes3044              100 non-null    int64
 35  CVotes3044M             100 non-null    int64
 36  CVotes3044F             100 non-null    int64
 37  CVotes45A               100 non-null    int64
 38  CVotes45AM              100 non-null    int64
 39  CVotes45AF              100 non-null    int64
 40  CVotes1000              100 non-null    int64
 41  CVotesUS                100 non-null    int64
 42  CVotesnUS               100 non-null    int64
 43  VotesM                  100 non-null    float64
 44  VotesF                  100 non-null    float64
 45  VotesU18                100 non-null    float64
 46  VotesU18M               100 non-null    float64
 47  VotesU18F               100 non-null    float64
 48  Votes1829               100 non-null    float64
 49  Votes1829M              100 non-null    float64
 50  Votes1829F              100 non-null    float64
 51  Votes3044               100 non-null    float64
 52  Votes3044M              100 non-null    float64
 53  Votes3044F              100 non-null    float64
 54  Votes45A                100 non-null    float64
 55  Votes45AM               100 non-null    float64
```

```
 56  Votes45AF              100 non-null    float64
 57  Votes1000              100 non-null    float64
 58  VotesUS                100 non-null    float64
 59  VotesnUS               100 non-null    float64
 60  content_rating         100 non-null    object
 61  Country                100 non-null    object
dtypes: float64(21), int64(32), object(9)
memory usage: 48.6+ KB
```

In [7]:

```
# Check the summary for the numeric columns
df.describe()
```

Out[7]:

	title_year	budget	Gross	actor_1_facebook_likes	actor_2_facebook_likes	actor_3_facebook_likes	IMDb_rating	MetaCriti
count	100.000000	1.000000e+02	1.000000e+02	100.000000	99.000000	98.000000	100.000000	95.00000
mean	2012.820000	7.838400e+07	1.468679e+08	13407.270000	7377.303030	3002.153061	7.883000	78.25263
std	1.919491	7.445295e+07	1.454004e+08	10649.037862	13471.568216	6940.301133	0.247433	9.12206
min	2010.000000	3.000000e+06	2.238380e+05	39.000000	12.000000	0.000000	7.500000	62.00000
25%	2011.000000	1.575000e+07	4.199752e+07	1000.000000	580.000000	319.750000	7.700000	72.00000
50%	2013.000000	4.225000e+07	1.070266e+08	13000.000000	1000.000000	626.500000	7.800000	78.00000
75%	2014.000000	1.500000e+08	2.107548e+08	20000.000000	11000.000000	1000.000000	8.100000	83.50000
max	2016.000000	2.600000e+08	9.366622e+08	35000.000000	96000.000000	46000.000000	8.800000	100.00000

8 rows × 53 columns

In [8]:

```
# Check the summary for the numeric and caretorical columns
```

Task 2: Data Analysis

Now that we have loaded the dataset and inspected it, we see that most of the data is in place. As of now, no data cleaning is required, so let's start with some data manipulation, analysis, and visualisation to get various insights about the data.

- ### Subtask 2.1: Reduce those Digits!

These numbers in the `budget` and `gross` are too big, compromising its readability. Let's convert the unit of the `budget` and `gross` columns from `$` to `million $` first.

In [5]:

```
df.Gross
```

Out[5]:

```
0     151101803
1     341268248
2      51738905
3     100546139
4      47695371
        ...
95     13092000
96      8114507
97    936662225
98    296347721
99       223838
Name: Gross, Length: 100, dtype: int64
```

In [6]:

```
# Divide the 'gross' and 'budget' columns by 1000000 to convert '$' to 'million $'
df['Gross_m_$']=df.Gross/1000000
```

In [7]:

```
df
```

Out[7]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1_facebook_likes	actor_2_facebook_likes
0	La La Land	2016	30000000	151101803	Ryan Gosling	Emma Stone	Amiée Conn	14000	19000.0
1	Zootopia	2016	150000000	341268248	Ginnifer Goodwin	Jason Bateman	Idris Elba	2800	28000.0
2	Lion	2016	12000000	51738905	Dev Patel	Nicole Kidman	Rooney Mara	33000	96000.0

3	Arrival	2016	47000000	100546139	Amy Adams	Jeremy Renner	Forest Whitaker	35000	5300.0
4	Manchester by the Sea	2016	9000000	47695371	Casey Affleck	Michelle Williams	Kyle Chandler	518	71000.0
...	...	...	...	...	...	...	...	...	...
95	Whiplash	2014	3300000	13092000	J.K. Simmons	Melissa Benoist	Chris Mulkey	24000	970.0
96	Before Midnight	2013	3000000	8114507	Seamus Davey-Fitzpatrick	Ariane Labed	Athina Rachel Tsangari	140	63.0
97	Star Wars: Episode VII - The Force Awakens	2015	245000000	936662225	Doug Walker	Rob Walker	0	131	12.0
98	Harry Potter and the Deathly Hallows: Part I	2010	150000000	296347721	Rupert Grint	Toby Jones	Alfred Enoch	10000	2000.0
99	Tucker and Dale vs Evil	2010	5000000	223838	Katrina Bowden	Tyler Labine	Chelan Simmons	948	779.0

100 rows × 63 columns

In [10]:

```
# Who is having highest rating in year 2016
df[df.title_year==2016].IMDb_rating.max()
```

Out[10]:

```
8.2
```

In [11]:

```
# Which movie has Highest budget till date?
df[df.budget==df.budget.max()]['Title']
```

Out[11]:

```
7    Tangled
Name: Title, dtype: object
```

In [12]:

```
# Average budget spent in 2013
df[df.title_year==2013].budget.mean()
```

Out[12]:

```
60588235.294117644
```

In [11]:

```
# How many Action movies Scarlett Johansson did?
(df.actor_3_name=='Scarlett Johansson']) | (df.actor_3_name=='Scarlett Johansson'])|(df.actor_3_name=='Scarlett J
```

Out[11]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1_facebook_likes	actor_2_facebook_likes	act
11	The Avengers	2012	220000000	623279547	Chris Hemsworth	Robert Downey Jr.	Scarlett Johansson	26000	21000.0	

1 rows × 63 columns

In [14]:

```
# Plot top 10 genres
df.head(10)
```

Out[14]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1_facebook_likes	actor_2_facebook_likes	ac
0	La La Land	2016	30000000	151101803	Ryan Gosling	Emma Stone	Amiée Conn	14000	19000.0	
1	Zootopia	2016	150000000	341268248	Ginnifer Goodwin	Jason Bateman	Idris Elba	2800	28000.0	
2	Lion	2016	12000000	51738905	Dev Patel	Nicole Kidman	Rooney Mara	33000	96000.0	
3	Arrival	2016	47000000	100546139	Amy Adams	Jeremy Renner	Forest Whitaker	35000	5300.0	
4	Manchester by the Sea	2016	9000000	47695371	Casey Affleck	Michelle Williams	Kyle Chandler	518	71000.0	
5	Hell or High Water	2016	12000000	27007844	Chris Pine	Jeff Bridges	Ben Foster	19000	12000.0	

6	Doctor Strange	2016	165000000	232641920	Benedict Cumberbatch	Chiwetel Ejiofor	Rachel McAdams	19000	NaN
7	Tangled	2010	260000000	200807262	Brad Garrett	Donna Murphy	M.C. Gainey	799	553.0
8	The Dark Knight Rises	2012	250000000	448130642	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	27000	23000.0
9	Captain America: Civil War	2016	250000000	407197282	Robert Downey Jr.	Scarlett Johansson	Chris Evans	21000	19000.0

10 rows × 62 columns

In [15]:

```
# List out movies with Sci-fi genre
df[(df.genre_1=="Sci-Fi")|(df.genre_2=="Sci-Fi")|(df.genre_3 =="Sci-Fi")]['Title']
```

Out[15]:

```
3                                  Arrival
9               Captain America: Civil War
11                            The Avengers
14              X-Men: Days of Future Past
15                 Star Trek Into Darkness
17                       Edge of Tomorrow
19                 Guardians of the Galaxy
20     Captain America: The Winter Soldier
26                           Interstellar
27                              Inception
28                     X-Men: First Class
29                     Mad Max: Fury Road
33                            The Martian
34                                Gravity
37          Rise of the Planet of the Apes
67                                    Her
80                             Ex Machina
Name: Title, dtype: object
```

In [16]:

```
# What is maximum rating of Action movies
df[(df.genre_1=="Action") |(df.genre_2=="Action") |(df.genre_3=="Action")].IMDb_rating.max()
```

Out[16]: 8.8

In [17]:

```
#List out X man series (means all movies under name - Xman)
df[(df.Title=='X-Men: Days of Future Past') | (df.Title=='X-Men: First Class')]
```

Out[17]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1_facebook_likes	actor_2_facebook_likes	actor_
14	X-Men: Days of Future Past	2014	200000000	233914986	Jennifer Lawrence	Peter Dinklage	Hugh Jackman	34000	22000.0	
28	X-Men: First Class	2011	160000000	146405371	Jennifer Lawrence	Michael Fassbender	Oliver Platt	34000	13000.0	

2 rows × 62 columns

In [18]:

```
# List out actors and actress played role in movie 127 Hours
df[df["Title"]=="127 Hours"]
```

Out[18]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1_facebook_likes	actor_2_facebook_likes	actor_3_
73	127 Hours	2010	18000000	18329466	James Franco	Treat Williams	Kate Burton	11000	642.0	

1 rows × 62 columns

- ### Subtask 2.2: Let's Talk Profit!

1. Create a new column called `profit` which contains the difference of the two columns: `gross` and `budget`.
2. Sort the dataframe using the `profit` column as reference.
3. Extract the top ten profiting movies in descending order and store them in a new dataframe - `top10`.
4. Plot a scatter between the columns `budget` and `profit` and write a few words on what you observed.
5. Extract the movies with a negative profit and store them in a new dataframe - `neg_profit`

In [19]:

```
# Create the new column named 'profit' by subtracting the 'budget' column from the 'gross' column
df['profit']=df['budget']-df['Gross']
df
```

Out[19]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1_facebook_likes	actor_2_facebook_likes	a
0	La La Land	2016	30000000	151101803	Ryan Gosling	Emma Stone	Amiée Conn	14000	19000.0	
1	Zootopia	2016	150000000	341268248	Ginnifer Goodwin	Jason Bateman	Idris Elba	2800	28000.0	
2	Lion	2016	12000000	51738905	Dev Patel	Nicole Kidman	Rooney Mara	33000	96000.0	
3	Arrival	2016	47000000	100546139	Amy Adams	Jeremy Renner	Forest Whitaker	35000	5300.0	
4	Manchester by the Sea	2016	9000000	47695371	Casey Affleck	Michelle Williams	Kyle Chandler	518	71000.0	
...	...	...	...	...	...	...	...	...	...	
95	Whiplash	2014	3300000	13092000	J.K. Simmons	Melissa Benoist	Chris Mulkey	24000	970.0	
96	Before Midnight	2013	3000000	8114507	Seamus Davey-Fitzpatrick	Ariane Labed	Athina Rachel Tsangari	140	63.0	
97	Star Wars: Episode VII - The Force Awakens	2015	245000000	936662225	Doug Walker	Rob Walker	0	131	12.0	
98	Harry Potter and the Deathly Hallows: Part I	2010	150000000	296347721	Rupert Grint	Toby Jones	Alfred Enoch	10000	2000.0	
99	Tucker and Dale vs Evil	2010	5000000	223838	Katrina Bowden	Tyler Labine	Chelan Simmons	948	779.0	

100 rows × 63 columns

In [20]:

```
# Sort the dataframe with the 'profit' column as reference using the 'sort_values' function. Make sure to set the
#'ascending' to 'False'
df.sort_values(["Title","profit"],axis=0,ascending=True,inplace=True)
df
```

Out[20]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1_facebook_likes	actor_2_facebook_likes	act
69	12 Years a Slave	2013	20000000	56667870	QuvenzhanÃ© Wallis	Scoot McNairy	Taran Killam	2000	660.0	
73	127 Hours	2010	18000000	18329466	James Franco	Treat Williams	Kate Burton	11000	642.0	
91	50/50	2011	8000000	34963967	Joseph Gordon-Levitt	Anna Kendrick	Bryce Dallas Howard	23000	10000.0	
88	About Time	2013	12000000	15294553	Tom Hughes	Tom Hollander	Lindsay Duncan	565	555.0	
89	Amour	2012	8900000	225377	Isabelle Huppert	Emmanuelle Riva	Jean-Louis Trintignant	678	432.0	
...	...	...	...	...	...	...	...	...	...	
95	Whiplash	2014	3300000	13092000	J.K. Simmons	Melissa Benoist	Chris Mulkey	24000	970.0	
25	Wreck-It Ralph	2012	165000000	189412677	Jack McBrayer	Sarah Silverman	Joe Lo Truglio	975	931.0	
14	X-Men: Days of Future Past	2014	200000000	233914986	Jennifer Lawrence	Peter Dinklage	Hugh Jackman	34000	22000.0	
28	X-Men: First Class	2011	160000000	146405371	Jennifer Lawrence	Michael Fassbender	Oliver Platt	34000	13000.0	
1	Zootopia	2016	150000000	341268248	Ginnifer Goodwin	Jason Bateman	Idris Elba	2800	28000.0	

100 rows × 63 columns

In [21]:
```
# Get the top 10 profitable movies by using position based indexing. Specify the rows till 10 (0-9)
df.head(10)
```

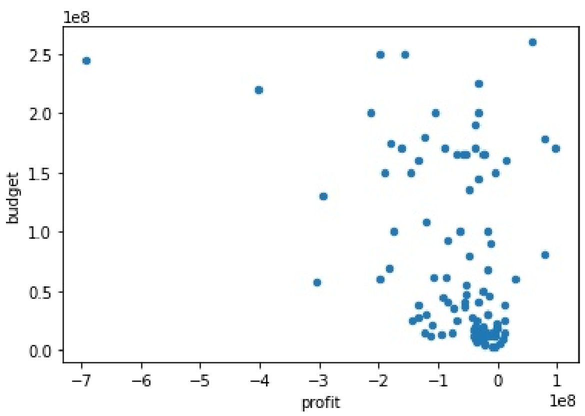
Out[21]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1_facebook_likes	actor_2_facebook_likes
69	12 Years a Slave	2013	20000000	56667870	QuvenzhanÃ© Wallis	Scoot McNairy	Taran Killam	2000	660.0
73	127 Hours	2010	18000000	18329466	James Franco	Treat Williams	Kate Burton	11000	642.0
91	50/50	2011	8000000	34963967	Joseph Gordon-Levitt	Anna Kendrick	Bryce Dallas Howard	23000	10000.0
88	About Time	2013	12000000	15294553	Tom Hughes	Tom Hollander	Lindsay Duncan	565	555.0
89	Amour	2012	8900000	225377	Isabelle Huppert	Emmanuelle Riva	Jean-Louis Trintignant	678	432.0
51	Argo	2012	44500000	136019448	Clea DuVall	Scoot McNairy	Tate Donovan	1000	660.0
3	Arrival	2016	47000000	100546139	Amy Adams	Jeremy Renner	Forest Whitaker	35000	5300.0
96	Before Midnight	2013	3000000	8114507	Seamus Davey-Fitzpatrick	Ariane Labed	Athina Rachel Tsangari	140	63.0
23	Big Hero 6	2014	165000000	222487711	Damon Wayans Jr.	Daniel Henney	Abraham Benrubi	756	719.0
72	Birdman or (The Unexpected Virtue of Ignorance)	2014	18000000	42335698	Emma Stone	Naomi Watts	Merritt Wever	15000	6000.0

10 rows × 63 columns

In [22]:
```
#Plot profit vs budget
import matplotlib.pyplot as plt
df.plot.scatter(x='profit',y='budget')
```

Out[22]: <AxesSubplot:xlabel='profit', ylabel='budget'>



My Observations: Movies with higher budgets are not necessarily profitable

The dataset contains the 100 best performing movies from the year 2010 to 2016. However, the scatter plot tells a different story. You can notice that there are some movies with negative profit. Although good movies do incur losses, but there appear to be quite a few movie with losses. What can be the reason behind this? Lets have a closer look at this by finding the movies with negative profit.

In [27]:
```
#Find the movies with negative profit
df[df['profit'] < 0]['Title']
```

Out[27]:
```
69              12 Years a Slave
73                     127 Hours
91                         50/50
88                    About Time
51                          Argo
```

```
                  ...
55                     True Grit
95                      Whiplash
25                Wreck-It Ralph
14    X-Men: Days of Future Past
1                       Zootopia
Name: Title, Length: 89, dtype: object
```

In [24]:

```
# Create the dataframe df_by_genre which will contain genre info only
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js