In [1]:

```python
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```python
import numpy as np
import pandas as pd
```

Task 1: Reading and Inspection

Subtask 1.1: Import and read Import and read the movie database. Store it in a variable called movies.

In [3]:

```python
movies = pd.read_csv(r"C:\Users\dell\Downloads\Movie+Assignment+Data.csv")
movies
```

Out[3]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	4
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	1
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	2
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	
...	...	...	...	...	...	...	...	
5038	Color	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	
5039	Color	NaN	43.0	43.0	NaN	319.0	Valorie Curry	
5040	Color	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	
5041	Color	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	
5042	Color	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	

5043 rows × 28 columns

Subtask 1.2: Inspect the dataframe

Inspect the dataframe's columns, shapes, variable types etc.

In [4]:

```python
# Check the number of rows and columns in the dataframe

movies.shape
```

Out[4]:

```
(5043, 28)
```

In [5]:

```python
# Check the column-wise info of the dataframe

movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 28 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   color                      5024 non-null   object 
 1   director_name              4939 non-null   object 
 2   num_critic_for_reviews     4993 non-null   float64
 3   duration                   5028 non-null   float64
 4   director_facebook_likes    4939 non-null   float64
 5   actor_3_facebook_likes     5020 non-null   float64
 6   actor_2_name               5030 non-null   object 
 7   actor_1_facebook_likes     5036 non-null   float64
 8   gross                      4159 non-null   float64
 9   genres                     5043 non-null   object 
 10  actor_1_name               5036 non-null   object 
 11  movie_title                5043 non-null   object 
 12  num_voted_users            5043 non-null   int64  
 13  cast_total_facebook_likes  5043 non-null   int64  
 14  actor_3_name               5020 non-null   object 
 15  facenumber_in_poster       5030 non-null   float64
 16  plot_keywords              4890 non-null   object 
 17  movie_imdb_link            5043 non-null   object 
 18  num_user_for_reviews       5022 non-null   float64
 19  language                   5031 non-null   object 
 20  country                    5038 non-null   object 
 21  content_rating             4740 non-null   object 
 22  budget                     4551 non-null   float64
 23  title_year                 4935 non-null   float64
 24  actor_2_facebook_likes     5030 non-null   float64
 25  imdb_score                 5043 non-null   float64
 26  aspect_ratio               4714 non-null   float64
 27  movie_facebook_likes       5043 non-null   int64  
dtypes: float64(13), int64(3), object(12)
memory usage: 1.1+ MB
```

In [6]:

```
# Get a summary of the dataframe using 'describe()'

movies.describe()
```

Out[6]:

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	gross	num_voted
count	4993.000000	5028.000000	4939.000000	5020.000000	5036.000000	4.159000e+03	5.0430
mean	140.194272	107.201074	686.509212	645.009761	6560.047061	4.846841e+07	8.3668
std	121.601675	25.197441	2813.328607	1665.041728	15020.759120	6.845299e+07	1.3848
min	1.000000	7.000000	0.000000	0.000000	0.000000	1.620000e+02	5.0000
25%	50.000000	93.000000	7.000000	133.000000	614.000000	5.340988e+06	8.5935
50%	110.000000	103.000000	49.000000	371.500000	988.000000	2.551750e+07	3.4359
75%	195.000000	118.000000	194.500000	636.000000	11000.000000	6.230944e+07	9.6309
max	813.000000	511.000000	23000.000000	23000.000000	640000.000000	7.605058e+08	1.6897

Task 2: Cleaning the Data

Subtask 2.1: Inspect Null values

Find out the number of Null values in all the columns and rows. Also, find the percentage of Null values in each column. Round-off the percentages upto two decimal places.

In [7]:

```
# Get the column-wise Null count using 'is.null()' alongwith the 'sum()' function

movies.isnull().sum()
```

Out[7]:
```
color                         19
director_name                104
num_critic_for_reviews        50
duration                      15
director_facebook_likes      104
actor_3_facebook_likes        23
actor_2_name                  13
actor_1_facebook_likes         7
gross                        884
genres                         0
actor_1_name                   7
movie_title                    0
num_voted_users                0
cast_total_facebook_likes      0
actor_3_name                  23
facenumber_in_poster          13
plot_keywords                153
movie_imdb_link                0
num_user_for_reviews          21
language                      12
country                        5
content_rating               303
budget                       492
title_year                   108
actor_2_facebook_likes        13
imdb_score                     0
aspect_ratio                 329
movie_facebook_likes           0
dtype: int64
```

In [8]:
```
# Get the row-wise Null count the same way. This time just specify the axis as 1

movies.isnull().sum(axis=1)
```

Out[8]:
```
0        0
1        0
2        0
3        0
4       14
        ..
5038     4
5039     5
5040     4
5041     2
5042     0
Length: 5043, dtype: int64
```

In [9]:
```
# Get the percentages by dividing the sum obtained previously by the total length, multiplying it by 100 and ro
# two decimal places

round(100*(movies.isnull().sum()/len(movies.index)), 2)
```

Out[9]:
```
color                         0.38
director_name                 2.06
num_critic_for_reviews        0.99
duration                      0.30
director_facebook_likes       2.06
actor_3_facebook_likes        0.46
actor_2_name                  0.26
actor_1_facebook_likes        0.14
gross                        17.53
genres                        0.00
actor_1_name                  0.14
movie_title                   0.00
num_voted_users               0.00
cast_total_facebook_likes     0.00
actor_3_name                  0.46
facenumber_in_poster          0.26
plot_keywords                 3.03
movie_imdb_link               0.00
num_user_for_reviews          0.42
language                      0.24
country                       0.10
content_rating                6.01
budget                        9.76
title_year                    2.14
actor_2_facebook_likes        0.26
imdb_score                    0.00
aspect_ratio                  6.52
movie_facebook_likes          0.00
dtype: float64
```

Subtask 2.2: Drop unecessary columns For this assignment, you will mostly be analyzing the movies with respect to the ratings, gross collection, popularity of movies, etc. So many of the columns in this dataframe are not required. So it is advised to drop the following columns. color director_facebook_likes actor_1_facebook_likes actor_2_facebook_likes actor_3_facebook_likes actor_2_name cast_total_facebook_likes actor_3_name duration facenumber_in_poster content_rating country movie_imdb_link aspect_ratio plot_keywords

In [10]:
```
# Use the 'drop()' function to drop the unnecessary columns

movies = movies.drop(['color',
```

```python
                        'director_facebook_likes',
                        'actor_3_facebook_likes',
                        'actor_1_facebook_likes',
                        'cast_total_facebook_likes',
                        'actor_2_facebook_likes',
                        'duration',
                        'facenumber_in_poster',
                        'content_rating',
                        'country',
                        'movie_imdb_link',
                        'aspect_ratio',
                        'plot_keywords',
                        'actor_2_name',
                        'actor_3_name'],
                         axis = 1)
movies
```

Out[10]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_use
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	
4	Doug Walker	NaN	NaN	Documentary	Doug Walker	Star Wars: Episode VII - The Force Awakens ...	8	
...	...	...	...	...	...	...	...	
5038	Scott Smith	1.0	NaN	Comedy|Drama	Eric Mabius	Signed Sealed Delivered	629	
5039	NaN	43.0	NaN	Crime|Drama|Mystery|Thriller	Natalie Zea	The Following	73839	
5040	Benjamin Roberds	13.0	NaN	Drama|Horror|Thriller	Eva Boehnke	A Plague So Pleasant	38	
5041	Daniel Hsia	14.0	10443.0	Comedy|Drama|Romance	Alan Ruck	Shanghai Calling	1255	
5042	Jon Gunn	43.0	85222.0	Documentary	John August	My Date with Drew	4285	

5043 rows × 13 columns

In [11]:

```python
# Inspect the dataset. Notice only 13 columns are left.

movies.shape
```

Out[11]:

```
(5043, 13)
```

Subtask 2.3: Drop unecessary rows using columns with high NaN percentages

On inspection you might notice that some columns have large percentage (greater than 5%) of Null values. Drop all the rows which have Null values for such columns.

In [12]:

```python
# Inspecting the percentages of Null values again

round(100*(movies.isnull().sum()/len(movies.index)), 2)
```

Out[12]:
```
director_name              2.06
num_critic_for_reviews     0.99
gross                     17.53
genres                     0.00
actor_1_name               0.14
movie_title                0.00
num_voted_users            0.00
num_user_for_reviews       0.42
language                   0.24
budget                     9.76
title_year                 2.14
imdb_score                 0.00
movie_facebook_likes       0.00
dtype: float64
```

In [13]:
```
# Since 'gross' and 'budget' columns have large number of NaN values, drop all the rows with NaNs at this colum
# 'isnan' function of NumPy alongwith a negation '~'

movies = movies[~np.isnan(movies['gross'])]
movies = movies[~np.isnan(movies['budget'])]
movies
```

Out[13]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	nu
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	
...	...	...	...	...	...	...	...	
5033	Shane Carruth	143.0	424760.0	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	
5034	Neill Dela Llana	35.0	70071.0	Thriller	Ian Gamazon	Cavite	589	
5035	Robert Rodriguez	56.0	2040920.0	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	
5037	Edward Burns	14.0	4584.0	Comedy|Drama	Kerry Bishé	Newlyweds	1338	
5042	Jon Gunn	43.0	85222.0	Documentary	John August	My Date with Drew	4285	

3891 rows × 13 columns

In [14]:
```
# Inspecting the percentages of NaN

round(100*(movies.isnull().sum()/len(movies.index)), 2)
```

Out[14]:
```
director_name             0.00
num_critic_for_reviews    0.03
gross                     0.00
genres                    0.00
actor_1_name              0.08
movie_title               0.00
num_voted_users           0.00
num_user_for_reviews      0.00
language                  0.08
budget                    0.00
title_year                0.00
imdb_score                0.00
movie_facebook_likes      0.00
dtype: float64
```

Subtask 2.4: Drop unecessary rows

Some of the rows might have greater than five Null values. Such rows aren't of much use for the analysis and hence, should be removed.

In [15]:
```
# The rows for which the sum of Null is less than five are retained

movies = movies[movies.isnull().sum(axis=1) <= 5]
movies
```

Out[15]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	nu
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	
...	...	...	...	...	...	...	...	
5033	Shane Carruth	143.0	424760.0	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	
5034	Neill Dela Llana	35.0	70071.0	Thriller	Ian Gamazon	Cavite	589	
5035	Robert Rodriguez	56.0	2040920.0	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	
5037	Edward Burns	14.0	4584.0	Comedy|Drama	Kerry Bishé	Newlyweds	1338	
5042	Jon Gunn	43.0	85222.0	Documentary	John August	My Date with Drew	4285	

3891 rows × 13 columns

In [16]:

```
# Inspecting the percentages of NaN

round(100*(movies.isnull().sum()/len(movies.index)), 2)
```

Out[16]:

```
director_name             0.00
num_critic_for_reviews    0.03
gross                     0.00
genres                    0.00
actor_1_name              0.08
movie_title               0.00
num_voted_users           0.00
num_user_for_reviews      0.00
language                  0.08
budget                    0.00
title_year                0.00
imdb_score                0.00
movie_facebook_likes      0.00
dtype: float64
```

Subtask 2.5: Fill NaN values

You might notice that the language column has some NaN values. Here, on inspection, you will see that it is safe to replace all the missing values with 'English'.

In [17]:

```
# Inspect the language column of the dataset

movies['language'].describe()
```

Out[17]:

```
count        3888
unique         38
top       English
freq         3707
Name: language, dtype: object
```

In [18]:

```
# Fill the NaN values with 'English' since most of the movies are in the English language

movies.loc[pd.isnull(movies['language']), ['language']] = 'English'
movies
```

Out[18]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	nu
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	
...	...	...	...	...	...	...	...	
5033	Shane Carruth	143.0	424760.0	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	
5034	Neill Dela Llana	35.0	70071.0	Thriller	Ian Gamazon	Cavite	589	
5035	Robert Rodriguez	56.0	2040920.0	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	
5037	Edward Burns	14.0	4584.0	Comedy|Drama	Kerry Bishé	Newlyweds	1338	
5042	Jon Gunn	43.0	85222.0	Documentary	John August	My Date with Drew	4285	

3891 rows × 13 columns

In [19]:

```
# Inspecting the percentages of NaNs

round(100*(movies.isnull().sum()/len(movies.index)), 2)
```

Out[19]:

```
director_name             0.00
num_critic_for_reviews    0.03
gross                     0.00
genres                    0.00
actor_1_name              0.08
movie_title               0.00
num_voted_users           0.00
num_user_for_reviews      0.00
language                  0.00
budget                    0.00
title_year                0.00
imdb_score                0.00
movie_facebook_likes      0.00
dtype: float64
```

Subtask 2.6: Check the number of retained rows

You might notice that two of the columns viz. num_critic_for_reviews and actor_1_name have small percentages of NaN values left. You can let these columns as it is for now. Check the number and percentage of the rows retained after completing all the tasks above.

In [20]:

```
# Get the number of retained rows using 'len()'
# Get the percentage of retained rows by dividing the current number of rows with initial number of rows

print(len(movies.index))
print(len(movies.index)/5042)
```

```
3891
0.771717572391908
```

Task 3: Data Analysis

Subtask 3.1: Change the unit of columns Convert the unit of the budget and gross columns from *tomillion*.

In [21]:

```
# Divide the 'gross' and 'budget' columns by 1000000 to convert '$' to 'million $'

movies['gross'] = movies['gross']/1000000
movies['budget'] = movies['budget']/1000000
movies
```

Out[21]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	nur
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	
3	Christopher Nolan	813.0	448.130642	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	
...	...	...	...	...	...	...	...	
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	
5034	Neill Dela Llana	35.0	0.070071	Thriller	Ian Gamazon	Cavite	589	
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	
5037	Edward Burns	14.0	0.004584	Comedy|Drama	Kerry Bishé	Newlyweds	1338	
5042	Jon Gunn	43.0	0.085222	Documentary	John August	My Date with Drew	4285	

3891 rows × 13 columns

Subtask 3.2: Find the movies with highest profit

Create a new column called profit which contains the difference of the two columns: gross and budget. Sort the dataframe using the profit column as reference. Extract the top ten profiting movies in descending order and store them in a new dataframe - top10

In [22]:
```
# Create the new column named 'profit' by subtracting the 'budget' column from the 'gross' column

movies['profit'] = movies['gross'] - movies['budget']
movies
```

Out[22]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	nur
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	
3	Christopher Nolan	813.0	448.130642	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	
...	...	...	...	...	...	...	...	
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	
5034	Neill Dela Llana	35.0	0.070071	Thriller	Ian Gamazon	Cavite	589	
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	
5037	Edward Burns	14.0	0.004584	Comedy|Drama	Kerry Bishé	Newlyweds	1338	
5042	Jon Gunn	43.0	0.085222	Documentary	John August	My Date with Drew	4285	

3891 rows × 14 columns

In [23]:
```
# Sort the dataframe with the 'profit' column as reference using the 'sort_values' function. Make sure to set t
# 'ascending' to 'False'

movies = movies.sort_values(by = 'profit', ascending = False)
movies
```

Out[23]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	nu
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	
...	...	...	...	...	...	...	...	
2334	Katsuhiro Ôtomo	105.0	0.410388	Action|Adventure|Animation|Family|Sci-Fi|Thriller	William Hootkins	Steamboy	13727	
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantasy	Minnie Driver	Princess Mononoke	221552	
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|War	Marcell Nagy	Fateless	5603	
3859	Chan-wook Park	202.0	0.211667	Crime|Drama	Min-sik Choi	Lady Vengeance	53508	
2988	Joon-ho Bong	363.0	2.201412	Comedy|Drama|Horror|Sci-Fi	Doona Bae	The Host	68883	

3891 rows × 14 columns

In [24]:

```
# Get the top 10 profitable movies by using position based indexing. Specify the rows till 10 (0-9)

top10 = movies.iloc[:10, ]
top10
```

Out[24]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842
794	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chris Hemsworth	The Avengers	995415
17	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chris Hemsworth	The Avengers	995415
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|Musical	Matthew Broderick	The Lion King	644348
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|Sci-Fi	Natalie Portman	Star Wars: Episode I - The Phantom Menace	534658
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Dark Knight	1676169

Subtask 3.3: Drop duplicate values

After you found out the top 10 profiting movies, you might have noticed a duplicate value. So, it seems like the dataframe has duplicate values as well. Drop the duplicate values from the dataframe and repeat Subtask 3.2.

In [25]:

```
# Drop the duplicate values using 'drop_duplicates' function. All the columns for duplicate rows need to be dro
# the 'subset' argument is set to 'None'. The 'keep = first' indicates to retain the first row among the duplic
# 'inplace = True' performs the operation on the dataframe in place.

movies.drop_duplicates(subset = None, keep = 'first', inplace = True)
movies
```

Out[25]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	nu
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	
...	...	...	...	...	...	...	...	
2334	Katsuhiro Ôtomo	105.0	0.410388	Action|Adventure|Animation|Family|Sci-Fi|Thriller	William Hootkins	Steamboy	13727	
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantasy	Minnie Driver	Princess Mononoke	221552	
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|War	Marcell Nagy	Fateless	5603	
3859	Chan-wook Park	202.0	0.211667	Crime|Drama	Min-sik Choi	Lady Vengeance	53508	
2988	Joon-ho Bong	363.0	2.201412	Comedy|Drama|Horror|Sci-Fi	Doona Bae	The Host	68883	

3856 rows × 14 columns

In [26]:

```
# Get the top 10 profitable movies by using position based indexing. Specify the rows till 10 (0-9)

top10 = movies.iloc[:10, ]
top10
```

Out[26]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842
794	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chris Hemsworth	The Avengers	995415
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|Musical	Matthew Broderick	The Lion King	644348
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|Sci-Fi	Natalie Portman	Star Wars: Episode I - The Phantom Menace	534658
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Dark Knight	1676169
439	Gary Ross	673.0	407.999255	Adventure|Drama|Sci-Fi|Thriller	Jennifer Lawrence	The Hunger Games	701607

Subtask 3.4: Find IMDb Top 250

Create a new dataframe IMDb_Top_250 and store the top 250 movies with the highest IMDb Rating (corresponding to the column: imdb_score). Also make sure that for all of these movies, the num_voted_users is greater than 25,000. Also add a Rank column containing the values 1 to 250 indicating the ranks of the corresponding films. Extract all the movies in the IMDb_Top_250 dataframe which are not in the English language and store them in a new dataframe named Top_Foreign_Lang_Film.

In [27]:

```
# Sort the movies by IMDb score
# Retain the movies with 'num_voted_users' greater than 25000
# Use position based indexing to get the first 250 rows in the sorted dataframe
```

```python
# Create a new column rank which contains the rank from 1 to 250

IMDb_Top_250 = movies.sort_values(by = 'imdb_score', ascending = False)
IMDb_Top_250 = IMDb_Top_250.loc[IMDb_Top_250.num_voted_users > 25000]
IMDb_Top_250 = IMDb_Top_250.iloc[:250, ]
IMDb_Top_250['Rank'] = range(1,251)
IMDb_Top_250
```

Out[27]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_u
1937	Frank Darabont	199.0	28.341469	Crime|Drama	Morgan Freeman	The Shawshank Redemption	1689764	
3466	Francis Ford Coppola	208.0	134.821952	Crime|Drama	Al Pacino	The Godfather	1155770	
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Dark Knight	1676169	
2837	Francis Ford Coppola	149.0	57.300000	Crime|Drama	Robert De Niro	The Godfather: Part II	790926	
339	Peter Jackson	328.0	377.019252	Action|Adventure|Drama|Fantasy	Orlando Bloom	The Lord of the Rings: The Return of the King	1215718	
...	...	...	...	...	...	...	...	
788	Cameron Crowe	149.0	32.522352	Adventure|Comedy|Drama|Music	Philip Seymour Hoffman	Almost Famous	207287	
99	Peter Jackson	645.0	303.001229	Adventure|Fantasy	Aidan Turner	The Hobbit: An Unexpected Journey	637246	
1606	Nick Cassavetes	177.0	0.064286	Drama|Romance	Ryan Gosling	The Notebook	396396	
1735	James Mangold	291.0	119.518352	Biography|Drama|Music|Romance	Sandra Ellis Lafferty	Walk the Line	188637	
639	Michael Mann	209.0	28.965197	Biography|Drama|Thriller	Al Pacino	The Insider	133526	

250 rows × 15 columns

In [28]:

```python
# Get the non-English language films using conditional label based indexing

Top_Foreign_Lang_Film = IMDb_Top_250.loc[IMDb_Top_250['language'] != 'English']
Top_Foreign_Lang_Film
```

Out[28]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_vot
4498	Sergio Leone	181.0	6.100000	Western	Clint Eastwood	The Good, the Bad and the Ugly	
4029	Fernando Meirelles	214.0	7.563397	Crime|Drama	Alice Braga	City of God	
4747	Akira Kurosawa	153.0	0.269061	Action|Adventure|Drama	Takashi Shimura	Seven Samurai	
2373	Hayao Miyazaki	246.0	10.049886	Adventure|Animation|Family|Fantasy	Bunta Sugawara	Spirited Away	
4259	Florian Henckel von Donnersmarck	215.0	11.284657	Drama|Thriller	Sebastian Koch	The Lives of Others	
4921	Majid Majidi	46.0	0.925402	Drama|Family	Bahare Seddiqi	Children of Heaven	
4105	Chan-wook Park	305.0	2.181290	Drama|Mystery|Thriller	Min-sik Choi	Oldboy	
1298	Jean-Pierre Jeunet	242.0	33.201661	Comedy|Romance	Mathieu Kassovitz	Amélie	
1329	S.S. Rajamouli	44.0	6.498000	Action|Adventure|Drama|Fantasy|War	Tamannaah Bhatia	Baahubali: The Beginning	
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantasy	Minnie Driver	Princess Mononoke	
2970	Wolfgang Petersen	96.0	11.433134	Adventure|Drama|Thriller|War	Jürgen Prochnow	Das Boot	
4659	Asghar Farhadi	354.0	7.098492	Drama|Mystery	Shahab Hosseini	A Separation	
4033	Thomas Vinterberg	349.0	0.610968	Drama	Thomas Bo Larsen	The Hunt	
2829	Oliver Hirschbiegel	192.0	5.501940	Biography|Drama|History|War	Thomas Kretschmann	Downfall	

2734	Fritz Lang	260.0	0.026435	Drama|Sci-Fi	Brigitte Helm	Metropolis
3550	Denis Villeneuve	226.0	6.857096	Drama|Mystery|War	Lubna Azabal	Incendies
4000	Juan José Campanella	262.0	20.167424	Drama|Mystery|Thriller	Ricardo Darín	The Secret in Their Eyes
2047	Hayao Miyazaki	212.0	4.710455	Adventure|Animation|Family|Fantasy	Christian Bale	Howl's Moving Castle
2551	Guillermo del Toro	406.0	37.623143	Drama|Fantasy|War	Ivana Baquero	Pan's Labyrinth
3553	José Padilha	142.0	0.008060	Action|Crime|Drama|Thriller	Wagner Moura	Elite Squad
2914	Je-kyu Kang	86.0	1.110186	Action|Drama|War	Min-sik Choi	Tae Guk Gi: The Brotherhood of War
2830	Alejandro Amenábar	157.0	2.086345	Biography|Drama|Romance	Belén Rueda	The Sea Inside
4267	Alejandro G. Iñárritu	157.0	5.383834	Drama|Thriller	Adriana Barraza	Amores Perros
3423	Katsuhiro Ôtomo	150.0	0.439162	Action|Animation|Sci-Fi	Mitsuo Iwata	Akira
4461	Thomas Vinterberg	98.0	1.647780	Drama	Ulrich Thomsen	The Celebration
3344	Karan Johar	210.0	4.018695	Adventure|Drama|Thriller	Shah Rukh Khan	My Name Is Khan
4284	Ari Folman	231.0	2.283276	Animation|Biography|Documentary|Drama|History|War	Ari Folman	Waltz with Bashir
3456	Vincent Paronnaud	242.0	4.443403	Animation|Biography|Drama|War	Catherine Deneuve	Persepolis
4144	Walter Salles	71.0	5.595428	Drama	Fernanda Montenegro	Central Station
4897	Sergio Leone	122.0	3.500000	Action|Drama|Western	Clint Eastwood	A Fistful of Dollars
3677	Christophe Barratier	112.0	3.629758	Drama|Music	Jean-Baptiste Maunier	The Chorus
4640	Cristian Mungiu	233.0	1.185783	Drama	Anamaria Marinca	4 Months, 3 Weeks and 2 Days
4415	Fabián Bielinsky	94.0	1.221261	Crime|Drama|Thriller	Ricardo Darín	Nine Queens
2863	Clint Eastwood	251.0	13.753931	Drama|History|War	Yuki Matsuzaki	Letters from Iwo Jima
3510	Yash Chopra	29.0	2.921738	Drama|Musical|Romance	Shah Rukh Khan	Veer-Zaara
3264	Michael Haneke	447.0	0.225377	Drama|Romance	Isabelle Huppert	Amour

Subtask 3.5: Find the best directors

Group the dataframe using the director_name column. Find out the top 10 directors for whom the mean of imdb_score is the highest and store them in a new dataframe top10director.

In [30]:

```
# Create a pivot table using 'director_name' as index, 'imdb_score' as values, and 'mean' as aggfunc
# Sort the values by 'imdb_score'. Keep 'ascending' as 'False'
# Extract the top 10 from the dataframe created

# PS: If I had to find the worst 10 directors, I would have sorted the dataframe in an ascending order and agai

director = movies.pivot_table(values = 'imdb_score', index = 'director_name', aggfunc = 'mean')
director = director.sort_values(by = 'imdb_score', ascending = False)
director = director.iloc[:10, ]
director
```

Out[30]:

director_name	imdb_score
Charles Chaplin	8.600000
Tony Kaye	8.600000
Alfred Hitchcock	8.500000
Ron Fricke	8.500000
Damien Chazelle	8.500000
Majid Majidi	8.500000
Sergio Leone	8.433333
Christopher Nolan	8.425000
S.S. Rajamouli	8.400000
Marius A. Markevicius	8.400000

Subtask 3.6: Find popular genres

You might have noticed the genres column in the dataframe with all the genres of the movies seperated by a pipe (|). Out of all the movie genres, the first two are most significant for any film.

Extract the first two genres from the genres column and store them in two new columns: genre_1 and genre_2. Some of the movies might have only one genre. In such cases, extract the single genre into both the columns, i.e. for such movies the genre_2 should be the same as genre_1. Group the dataframe using genre_1 as the primary column and genre_2 as the secondary column. Find out the 5 most popular combo of genres by finding the mean of the gross values using the gross column and store them in a new dataframe named PopGenre.

In [31]:

```
# Split the elements of the 'genre' column at the pipe characters ('|') using str.split()
# Assign the first elements of the rows of 'genre' column to a new column named 'genre_1' using 'apply()' and '
# Some of the movies have only one genre. In such cases, assign the same genre to 'genre_2' as well

movies['genres'] = movies['genres'].str.split('|')
movies['genre_1'] = movies['genres'].apply(lambda x: x[0])
movies['genre_2'] = movies['genres'].apply(lambda x : x[1] if len(x) > 1 else x[0])
movies
```

Out[31]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	lar
0	James Cameron	723.0	760.505847	[Action, Adventure, Fantasy, Sci-Fi]	CCH Pounder	Avatar	886204	3054.0	
29	Colin Trevorrow	644.0	652.177271	[Action, Adventure, Sci-Fi, Thriller]	Bryce Dallas Howard	Jurassic World	418214	1290.0	
26	James Cameron	315.0	658.672302	[Drama, Romance]	Leonardo DiCaprio	Titanic	793059	2528.0	
3024	George Lucas	282.0	460.935665	[Action, Adventure, Fantasy, Sci-Fi]	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	1470.0	
3080	Steven Spielberg	215.0	434.949459	[Family, Sci-Fi]	Henry Thomas	E.T. the Extra-Terrestrial	281842	515.0	
...	...	...	...	...	...	...	...	...	
2334	Katsuhiro Ôtomo	105.0	0.410388	[Action, Adventure, Animation, Family, Sci-Fi,...	William Hootkins	Steamboy	13727	79.0	Ja
2323	Hayao Miyazaki	174.0	2.298191	[Adventure, Animation, Fantasy]	Minnie Driver	Princess Mononoke	221552	570.0	Ja
3005	Lajos Koltai	73.0	0.195888	[Drama, Romance, War]	Marcell Nagy	Fateless	5603	45.0	Hu
3859	Chan-wook Park	202.0	0.211667	[Crime, Drama]	Min-sik Choi	Lady Vengeance	53508	131.0	
2988	Joon-ho Bong	363.0	2.201412	[Comedy, Drama, Horror, Sci-Fi]	Doona Bae	The Host	68883	279.0	

3856 rows × 16 columns

In [32]:

```
# Group the dataframe using 'genre_1' as the primary column and 'genre_2' as secondary

movies_by_segment = movies.groupby(['genre_1', 'genre_2'])
```

In [33]:

```
# Create a new dataframe PopGenre which contains the 'mean' of the gross values of each combination of genres p
# Sort this dataframe using the 'gross' column and use index-based positioning to find out the five most popula

PopGenre = pd.DataFrame(movies_by_segment['gross'].mean()).sort_values(by = 'gross', ascending = False)
PopGenre.iloc[:5, ]
```

Out[33]:

genre_1	genre_2	gross
Family	**Sci-Fi**	434.949459
Adventure	**Sci-Fi**	228.627758
	Family	118.919540
	Animation	116.998550
Action	**Adventure**	109.595465

Subtask 3.7: Find the critic-favorite and audience-favorite actors

Create three new dataframes namely, Meryl_Streep, Leo_Caprio, and Brad_Pitt which contain the movies in which the actors: 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' are the lead actors. Use only the actor_1_name column for extraction. Also, make sure that you use the names 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' for the said extraction. Append the rows of all these dataframes and store them in a new dataframe named Combined. Group the combined dataframe using the actor_1_name column. Find the mean of the num_critic_for_reviews and num_user_for_review and identify the actors which have the highest mean.

In [34]:

```
# Create a new dataframe containing Meryl Streep movies in which she is the lead actor

Meryl_Streep = movies.loc[movies.actor_1_name == 'Meryl Streep']
Meryl_Streep
```

Out[34]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	lar
1408	David Frankel	208.0	124.732962	[Comedy, Drama, Romance]	Meryl Streep	The Devil Wears Prada	286178	631.0	
1575	Sydney Pollack	66.0	87.100000	[Biography, Drama, Romance]	Meryl Streep	Out of Africa	52339	200.0	
1204	Nora Ephron	252.0	94.125426	[Biography, Drama, Romance]	Meryl Streep	Julie & Julia	79264	277.0	
1618	David Frankel	234.0	63.536011	[Comedy, Drama, Romance]	Meryl Streep	Hope Springs	34258	178.0	
410	Nancy Meyers	187.0	112.703470	[Comedy, Drama, Romance]	Meryl Streep	It's Complicated	69860	214.0	
2781	Phyllida Lloyd	331.0	29.959436	[Biography, Drama, History]	Meryl Streep	The Iron Lady	82327	350.0	
1925	Stephen Daldry	174.0	41.597830	[Drama, Romance]	Meryl Streep	The Hours	102123	660.0	
3135	Robert Altman	211.0	20.338609	[Comedy, Drama, Music]	Meryl Streep	A Prairie Home Companion	19655	280.0	
1106	Curtis Hanson	42.0	46.815748	[Action, Adventure, Crime, Thriller]	Meryl Streep	The River Wild	32544	69.0	
1674	Carl Franklin	64.0	23.209440	[Drama]	Meryl Streep	One True Thing	9283	112.0	
1483	Robert Redford	227.0	14.998070	[Drama, Thriller, War]	Meryl Streep	Lions for Lambs	41170	298.0	

In [35]:

```
# Create a new dataframe containing Leonardo DiCaprio movies in which he is the lead actor

Leo_Caprio = movies.loc[movies.actor_1_name == 'Leonardo DiCaprio']
Leo_Caprio
```

Out[35]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	la
26	James Cameron	315.0	658.672302	[Drama, Romance]	Leonardo DiCaprio	Titanic	793059	2528.0	
97	Christopher Nolan	642.0	292.568851	[Action, Adventure, Sci-Fi, Thriller]	Leonardo DiCaprio	Inception	1468200	2803.0	
911	Steven Spielberg	194.0	164.435221	[Biography, Crime, Drama]	Leonardo DiCaprio	Catch Me If You Can	525801	667.0	
296	Quentin Tarantino	765.0	162.804648	[Drama, Western]	Leonardo DiCaprio	Django Unchained	955174	1193.0	
179	Alejandro G. Iñárritu	556.0	183.635922	[Adventure, Drama, Thriller, Western]	Leonardo DiCaprio	The Revenant	406020	1188.0	
452	Martin Scorsese	490.0	127.968405	[Mystery, Thriller]	Leonardo DiCaprio	Shutter Island	786092	964.0	
361	Martin Scorsese	352.0	132.373442	[Crime, Drama, Thriller]	Leonardo DiCaprio	The Departed	873649	2054.0	
50	Baz Luhrmann	490.0	144.812796	[Drama, Romance]	Leonardo DiCaprio	The Great Gatsby	362912	753.0	
3476	Baz Luhrmann	490.0	144.812796	[Drama, Romance]	Leonardo DiCaprio	The Great Gatsby	362933	753.0	
2757	Baz Luhrmann	106.0	46.338728	[Drama, Romance]	Leonardo DiCaprio	Romeo + Juliet	167750	506.0	
1422	Randall Wallace	83.0	56.876365	[Action, Adventure]	Leonardo DiCaprio	The Man in the Iron Mask	125219	244.0	
308	Martin Scorsese	606.0	116.866727	[Biography, Comedy, Crime, Drama]	Leonardo DiCaprio	The Wolf of Wall Street	780588	1138.0	
1453	Clint Eastwood	392.0	37.304950	[Biography, Crime, Drama]	Leonardo DiCaprio	J. Edgar	102728	279.0	
257	Martin Scorsese	267.0	102.608827	[Biography, Drama]	Leonardo DiCaprio	The Aviator	264318	799.0	
2067	Jerry Zaks	45.0	12.782508	[Drama]	Leonardo DiCaprio	Marvin's Room	20163	71.0	
990	Danny Boyle	118.0	39.778599	[Adventure, Drama, Thriller]	Leonardo DiCaprio	The Beach	176169	548.0	
1114	Sam Mendes	323.0	22.877808	[Drama, Romance]	Leonardo DiCaprio	Revolutionary Road	152591	414.0	
1560	Sam Raimi	63.0	18.636537	[Action, Thriller, Western]	Leonardo DiCaprio	The Quick and the Dead	69197	216.0	
326	Martin Scorsese	233.0	77.679638	[Crime, Drama]	Leonardo DiCaprio	Gangs of New York	314033	1166.0	
641	Ridley Scott	238.0	39.380442	[Action, Drama, Thriller]	Leonardo DiCaprio	Body of Lies	174248	263.0	
307	Edward Zwick	166.0	57.366262	[Adventure, Drama, Thriller]	Leonardo DiCaprio	Blood Diamond	400292	657.0	

In [36]:

```
# Create a new dataframe containing Brad Pitt movies in which he is the lead actor

Brad_Pitt = movies.loc[movies.actor_1_name == 'Brad Pitt']
Brad_Pitt
```

Out[36]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	l
400	Steven Soderbergh	186.0	183.405771	[Crime, Thriller]	Brad Pitt	Ocean's Eleven	402645	845.0	
255	Doug Liman	233.0	186.336103	[Action, Comedy, Crime, Romance, Thriller]	Brad Pitt	Mr. & Mrs. Smith	348861	798.0	
940	Neil Jordan	120.0	105.264608	[Drama, Fantasy, Horror]	Brad Pitt	Interview with the Vampire: The Vampire Chroni...	239752	406.0	
470	David Ayer	406.0	85.707116	[Action, Drama, War]	Brad Pitt	Fury	303185	701.0	
254	Steven Soderbergh	198.0	125.531634	[Crime, Thriller]	Brad Pitt	Ocean's Twelve	284852	627.0	
2204	Alejandro G. Iñárritu	285.0	34.300771	[Drama]	Brad Pitt	Babel	243799	908.0	
2682	Andrew Dominik	414.0	14.938570	[Crime, Thriller]	Brad Pitt	Killing Them Softly	111625	369.0	
2898	Tony Scott	122.0	12.281500	[Action, Crime, Drama, Romance, Thriller]	Brad Pitt	True Romance	163492	460.0	
2333	Angelina Jolie Pitt	131.0	0.531009	[Drama, Romance]	Brad Pitt	By the Sea	7976	61.0	
1490	Terrence Malick	584.0	13.303319	[Drama, Fantasy]	Brad Pitt	The Tree of Life	136367	975.0	
101	David Fincher	362.0	127.490802	[Drama, Fantasy, Romance]	Brad Pitt	The Curious Case of Benjamin Button	459346	822.0	
683	David Fincher	315.0	37.023395	[Drama]	Brad Pitt	Fight Club	1347461	2968.0	
1722	Andrew Dominik	273.0	3.904982	[Biography, Crime, Drama, History, Western]	Brad Pitt	The Assassination of Jesse James by the Coward...	136104	415.0	
611	Jean-Jacques Annaud	76.0	37.901509	[Adventure, Biography, Drama, History, War]	Brad Pitt	Seven Years in Tibet	96385	119.0	
792	Patrick Gilmore	98.0	26.288320	[Adventure, Animation, Comedy, Drama, Family, ...	Brad Pitt	Sinbad: Legend of the Seven Seas	36144	91.0	
147	Wolfgang Petersen	220.0	133.228348	[Adventure]	Brad Pitt	Troy	381672	1694.0	
382	Tony Scott	142.0	0.026871	[Action, Crime, Thriller]	Brad Pitt	Spy Game	121259	361.0	

In [37]:

```
# Combine the three dataframes using 'pd.concat()'

Combined = pd.concat([Meryl_Streep, Brad_Pitt, Leo_Caprio])
Combined
```

Out[37]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	l
1408	David Frankel	208.0	124.732962	[Comedy, Drama, Romance]	Meryl Streep	The Devil Wears Prada	286178	631.0	
1575	Sydney Pollack	66.0	87.100000	[Biography, Drama, Romance]	Meryl Streep	Out of Africa	52339	200.0	
1204	Nora Ephron	252.0	94.125426	[Biography, Drama, Romance]	Meryl Streep	Julie & Julia	79264	277.0	
1618	David Frankel	234.0	63.536011	[Comedy, Drama, Romance]	Meryl Streep	Hope Springs	34258	178.0	
410	Nancy Meyers	187.0	112.703470	[Comedy, Drama, Romance]	Meryl Streep	It's Complicated	69860	214.0	
				[Biography,					

2781	Phyllida Lloyd	331.0	29.959436	Drama, History]	Meryl Streep	The Iron Lady	82327	350.0
1925	Stephen Daldry	174.0	41.597830	[Drama, Romance]	Meryl Streep	The Hours	102123	660.0
3135	Robert Altman	211.0	20.338609	[Comedy, Drama, Music]	Meryl Streep	A Prairie Home Companion	19655	280.0
1106	Curtis Hanson	42.0	46.815748	[Action, Adventure, Crime, Thriller]	Meryl Streep	The River Wild	32544	69.0
1674	Carl Franklin	64.0	23.209440	[Drama]	Meryl Streep	One True Thing	9283	112.0
1483	Robert Redford	227.0	14.998070	[Drama, Thriller, War]	Meryl Streep	Lions for Lambs	41170	298.0
400	Steven Soderbergh	186.0	183.405771	[Crime, Thriller]	Brad Pitt	Ocean's Eleven	402645	845.0
255	Doug Liman	233.0	186.336103	[Action, Comedy, Crime, Romance, Thriller]	Brad Pitt	Mr. & Mrs. Smith	348861	798.0
940	Neil Jordan	120.0	105.264608	[Drama, Fantasy, Horror]	Brad Pitt	Interview with the Vampire: The Vampire Chroni...	239752	406.0
470	David Ayer	406.0	85.707116	[Action, Drama, War]	Brad Pitt	Fury	303185	701.0
254	Steven Soderbergh	198.0	125.531634	[Crime, Thriller]	Brad Pitt	Ocean's Twelve	284852	627.0
2204	Alejandro G. Iñárritu	285.0	34.300771	[Drama]	Brad Pitt	Babel	243799	908.0
2682	Andrew Dominik	414.0	14.938570	[Crime, Thriller]	Brad Pitt	Killing Them Softly	111625	369.0
2898	Tony Scott	122.0	12.281500	[Action, Crime, Drama, Romance, Thriller]	Brad Pitt	True Romance	163492	460.0
2333	Angelina Jolie Pitt	131.0	0.531009	[Drama, Romance]	Brad Pitt	By the Sea	7976	61.0
1490	Terrence Malick	584.0	13.303319	[Drama, Fantasy]	Brad Pitt	The Tree of Life	136367	975.0
101	David Fincher	362.0	127.490802	[Drama, Fantasy, Romance]	Brad Pitt	The Curious Case of Benjamin Button	459346	822.0
683	David Fincher	315.0	37.023395	[Drama]	Brad Pitt	Fight Club	1347461	2968.0
1722	Andrew Dominik	273.0	3.904982	[Biography, Crime, Drama, History, Western]	Brad Pitt	The Assassination of Jesse James by the Coward...	136104	415.0
611	Jean-Jacques Annaud	76.0	37.901509	[Adventure, Biography, Drama, History, War]	Brad Pitt	Seven Years in Tibet	96385	119.0
792	Patrick Gilmore	98.0	26.288320	[Adventure, Animation, Comedy, Drama, Family, ...	Brad Pitt	Sinbad: Legend of the Seven Seas	36144	91.0
147	Wolfgang Petersen	220.0	133.228348	[Adventure]	Brad Pitt	Troy	381672	1694.0
382	Tony Scott	142.0	0.026871	[Action, Crime, Thriller]	Brad Pitt	Spy Game	121259	361.0
26	James Cameron	315.0	658.672302	[Drama, Romance]	Leonardo DiCaprio	Titanic	793059	2528.0
97	Christopher Nolan	642.0	292.568851	[Action, Adventure, Sci-Fi, Thriller]	Leonardo DiCaprio	Inception	1468200	2803.0
911	Steven Spielberg	194.0	164.435221	[Biography, Crime, Drama]	Leonardo DiCaprio	Catch Me If You Can	525801	667.0

296	Quentin Tarantino	765.0	162.804648	[Drama, Western]	Leonardo DiCaprio	Django Unchained	955174	1193.0
179	Alejandro G. Iñárritu	556.0	183.635922	[Adventure, Drama, Thriller, Western]	Leonardo DiCaprio	The Revenant	406020	1188.0
452	Martin Scorsese	490.0	127.968405	[Mystery, Thriller]	Leonardo DiCaprio	Shutter Island	786092	964.0
361	Martin Scorsese	352.0	132.373442	[Crime, Drama, Thriller]	Leonardo DiCaprio	The Departed	873649	2054.0
50	Baz Luhrmann	490.0	144.812796	[Drama, Romance]	Leonardo DiCaprio	The Great Gatsby	362912	753.0
3476	Baz Luhrmann	490.0	144.812796	[Drama, Romance]	Leonardo DiCaprio	The Great Gatsby	362933	753.0
2757	Baz Luhrmann	106.0	46.338728	[Drama, Romance]	Leonardo DiCaprio	Romeo + Juliet	167750	506.0
1422	Randall Wallace	83.0	56.876365	[Action, Adventure]	Leonardo DiCaprio	The Man in the Iron Mask	125219	244.0
308	Martin Scorsese	606.0	116.866727	[Biography, Comedy, Crime, Drama]	Leonardo DiCaprio	The Wolf of Wall Street	780588	1138.0
1453	Clint Eastwood	392.0	37.304950	[Biography, Crime, Drama]	Leonardo DiCaprio	J. Edgar	102728	279.0
257	Martin Scorsese	267.0	102.608827	[Biography, Drama]	Leonardo DiCaprio	The Aviator	264318	799.0
2067	Jerry Zaks	45.0	12.782508	[Drama]	Leonardo DiCaprio	Marvin's Room	20163	71.0
990	Danny Boyle	118.0	39.778599	[Adventure, Drama, Thriller]	Leonardo DiCaprio	The Beach	176169	548.0
1114	Sam Mendes	323.0	22.877808	[Drama, Romance]	Leonardo DiCaprio	Revolutionary Road	152591	414.0
1560	Sam Raimi	63.0	18.636537	[Action, Thriller, Western]	Leonardo DiCaprio	The Quick and the Dead	69197	216.0
326	Martin Scorsese	233.0	77.679638	[Crime, Drama]	Leonardo DiCaprio	Gangs of New York	314033	1166.0
641	Ridley Scott	238.0	39.380442	[Action, Drama, Thriller]	Leonardo DiCaprio	Body of Lies	174248	263.0
307	Edward Zwick	166.0	57.366262	[Adventure, Drama, Thriller]	Leonardo DiCaprio	Blood Diamond	400292	657.0

In [38]:

```
# Group the dataframe by 'actor_1_name'

Combined_by_segment = Combined.groupby('actor_1_name')
```

In [39]:

```
# Remember that we had some null values for the column 'num_critic_for_reviews'. Make sure that none of these n
# present in the new dataframe - 'Combined' that we have created

Combined.isnull().sum()
```

Out[39]:

```
director_name             0
num_critic_for_reviews    0
gross                     0
genres                    0
actor_1_name              0
movie_title               0
num_voted_users           0
num_user_for_reviews      0
language                  0
budget                    0
title_year                0
imdb_score                0
movie_facebook_likes      0
profit                    0
genre_1                   0
genre_2                   0
dtype: int64
```

We are Good To GO!

In [40]:

```
# Find the mean of 'num_user_for_reviews' for each of the actor. Notice, Leonardo's is the highest

Combined_by_segment['num_user_for_reviews'].mean()
```

Out[40]:
```
actor_1_name
Brad Pitt            742.352941
Leonardo DiCaprio    914.476190
Meryl Streep         297.181818
Name: num_user_for_reviews, dtype: float64
```

In [41]:
```
# Find the mean of 'num_critic_for_reviews' for each of the actor. In this case as well, Leonardo is leading

Combined_by_segment['num_critic_for_reviews'].mean()
```

Out[41]:
```
actor_1_name
Brad Pitt            245.000000
Leonardo DiCaprio    330.190476
Meryl Streep         181.454545
Name: num_critic_for_reviews, dtype: float64
```

In [ ]:

Processing math: 100%