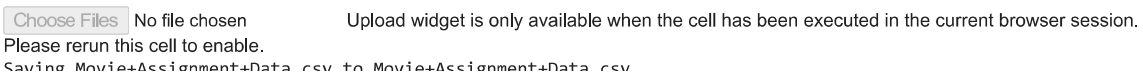
```
# Supress Warnings

import warnings
warnings.filterwarnings('ignore')


# Import the numpy and pandas packages

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`.

```
from google.colab import files
uploaded=files.upload()
```




```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
 # Write your code for importing the csv file here
movies = pd.read_csv("Movie+Assignment+Data.csv")
#movies.head()
print(movies.shape)
```

```
(5043, 28)
```

- Subtask 1.2: Inspect the dataframe

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

```
# Write your code for inspection here
print(movies.columns)
print(movies.shape)
print(movies.dtypes)
```

```
Index(['color', 'director_name', 'num_critic_for_reviews', 'duration',
       'director_facebook_likes', 'actor_3_facebook_likes', 'actor_2_name',
       'actor_1_facebook_likes', 'gross', 'genres', 'actor_1_name',
       'movie_title', 'num_voted_users', 'cast_total_facebook_likes',
       'actor_3_name', 'facenumber_in_poster', 'plot_keywords',
       'movie_imdb_link', 'num_user_for_reviews', 'language', 'country',
       'content_rating', 'budget', 'title_year', 'actor_2_facebook_likes',
       'imdb_score', 'aspect_ratio', 'movie_facebook_likes'],
      dtype='object')
(5043, 28)
color                         object
director_name                 object
num_critic_for_reviews       float64
duration                     float64
director_facebook_likes      float64
actor_3_facebook_likes       float64
actor_2_name                  object
actor_1_facebook_likes       float64
gross                        float64
genres                        object
actor_1_name                  object
movie_title                   object
num_voted_users                int64
cast_total_facebook_likes      int64
actor_3_name                  object
facenumber_in_poster         float64
plot_keywords                 object
movie_imdb_link               object
num_user_for_reviews         float64
```

```
language                    object
country                     object
content_rating              object
budget                     float64
title_year                 float64
actor_2_facebook_likes     float64
imdb_score                 float64
aspect_ratio               float64
movie_facebook_likes         int64
dtype: object
```

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.

```
# Write your code for column-wise null count here
null_counts = movies.isnull().sum(axis=0)

print(null_counts)
```

```
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
```

```
# Write your code for row-wise null count here
null_counts = movies.isnull().sum(axis=1)

print(null_counts)
```

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

```
# Write your code for column-wise null percentages here
percent_missing = movies.isnull().sum() * 100 / len(movies)

print(percent_missing)
```

```
color                       0.376760
director_name               2.062265
num_critic_for_reviews      0.991473
duration                    0.297442
director_facebook_likes     2.062265
actor_3_facebook_likes      0.456078
actor_2_name                0.257783
actor_1_facebook_likes      0.138806
gross                      17.529248
genres                      0.000000
actor_1_name                0.138806
movie_title                 0.000000
num_voted_users             0.000000
cast_total_facebook_likes   0.000000
actor_3_name                0.456078
facenumber_in_poster        0.257783
plot_keywords               3.033908
movie_imdb_link             0.000000
num_user_for_reviews        0.416419
language                    0.237954
country                     0.099147
content_rating              6.008328
budget                      9.756098
title_year                  2.141582
actor_2_facebook_likes      0.257783
imdb_score                  0.000000
aspect_ratio                6.523895
movie_facebook_likes        0.000000
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

```
# Write your code for dropping the columns here. It is advised to keep inspecting the dataframe after each set of operations
movies = movies.drop(['color',"director_facebook_likes","actor_1_facebook_likes","actor_2_facebook_likes","actor_3_facebook_likes","actor_2_n
print(movies.shape)
```

```
(5043, 13)
```

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

Now, 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.

```
# Write your code for dropping the rows here
perc = 5.0
min_count =  int(((100-perc)/100)*movies.shape[1] + 1)
movies_1 = movies.dropna( axis=0,
                    thresh=min_count)
print(movies_1.shape)
```

```
(3884, 13)
```

Subtask 2.4: Drop unecessary rows

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

```
# Write your code for dropping the rows here
movies_2 = movies_1[movies_1.isnull().sum(axis=1) < 6]
print(movies_2.shape)
```

```
(3884, 13)
```

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'`.

```
# Write your code for filling the NaN values in the 'language' column here
movies_2["language"] = movies_2["language"].replace(np.nan,"English")
movies_2["language"].isnull().sum()
```

```
0
```

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.

```
# Write your code for checking number of retained rows here
rows_retained = movies_2.shape[0]
print(rows_retained)
rows_retained_percent = (100*rows_retained)/5043
print(rows_retained_percent)
```

```
3884
77.01764822526275
```

Checkpoint 1: You might have noticed that we still have around `77%` of the rows!

Task 3: Data Analysis

Subtask 3.1: Change the unit of columns

Convert the unit of the `budget` and `gross` columns from `$` to `million $`.

```
# Write your code for unit conversion here
print(movies_2.budget[0])
movies_2["budget"] = movies_2["budget"].div(1000000)
movies_2["gross"] = movies_2["gross"].div(1000000)
print(movies_2.budget[0])
```

```
237000000.0
237.0
```

Subtask 3.2: Find the movies with highest 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`

```
# Write your code for creating the profit column here
movies_2["Profit"] = movies_2["gross"]-movies_2["budget"]
print(movies_2)
```

```
            director_name  num_critic_for_reviews       gross  \
0         James Cameron                   723.0  760.505847
```

```
1       Gore Verbinski                  302.0  309.404152
2           Sam Mendes                  602.0  200.074175
3     Christopher Nolan                 813.0  448.130642
5        Andrew Stanton                 462.0   73.058679
...                 ...                   ...         ...
5033      Shane Carruth                 143.0    0.424760
5034   Neill Dela Llana                  35.0    0.070071
5035   Robert Rodriguez                  56.0    2.040920
5037       Edward Burns                  14.0    0.004584
5042           Jon Gunn                  43.0    0.085222

                                genres     actor_1_name  \
0         Action|Adventure|Fantasy|Sci-Fi      CCH Pounder
1                Action|Adventure|Fantasy      Johnny Depp
2               Action|Adventure|Thriller  Christoph Waltz
3                         Action|Thriller        Tom Hardy
5                  Action|Adventure|Sci-Fi     Daryl Sabara
...                                   ...              ...
5033               Drama|Sci-Fi|Thriller    Shane Carruth
5034                             Thriller      Ian Gamazon
5035  Action|Crime|Drama|Romance|Thriller  Carlos Gallardo
5037                       Comedy|Drama      Kerry Bishé
5042                        Documentary      John August

                                  movie_title  num_voted_users  \
0                                      Avatar           886204
1     Pirates of the Caribbean: At World's End           471220
2                                    Spectre           275868
3                      The Dark Knight Rises          1144337
5                                  John Carter           212204
...                                        ...              ...
5033                                 Primer             72639
5034                                 Cavite               589
5035                              El Mariachi            52055
5037                                Newlyweds             1338
5042                        My Date with Drew             4285

      num_user_for_reviews language    budget  title_year  imdb_score  \
0                   3054.0  English  237.0000      2009.0         7.9
1                   1238.0  English  300.0000      2007.0         7.1
2                    994.0  English  245.0000      2015.0         6.8
3                   2701.0  English  250.0000      2012.0         8.5
5                    738.0  English  263.7000      2012.0         6.6
...                    ...      ...       ...         ...         ...
5033                 371.0  English    0.0070      2004.0         7.0
5034                  35.0  English    0.0070      2005.0         6.3
5035                 130.0  Spanish    0.0070      1992.0         6.9
5037                  14.0  English    0.0090      2011.0         6.4
5042                  84.0  English    0.0011      2004.0         6.6

      movie_facebook_likes      Profit
0                    33000  523.505847
1                        0    9.404152
2                    85000  -44.925825
3                   164000  198.130642
5                    24000 -190.641321
```

```
# Write your code for sorting the dataframe here
movies_3 = movies_2.sort_values(by='Profit', ascending=False)
print(movies_3)
```

```
           director_name  num_critic_for_reviews       gross  \
0         James Cameron                   723.0  760.505847
29      Colin Trevorrow                   644.0  652.177271
26        James Cameron                   315.0  658.672302
3024       George Lucas                   282.0  460.935665
3080  Steven Spielberg                    215.0  434.949459
...                  ...                   ...         ...
2334   Katsuhiro Ôtomo                   105.0    0.410388
2323    Hayao Miyazaki                   174.0    2.298191
3005      Lajos Koltai                    73.0    0.195888
3859     Chan-wook Park                   202.0    0.211667
2988       Joon-ho Bong                   363.0    2.201412

                                            genres         actor_1_name  \
0                      Action|Adventure|Fantasy|Sci-Fi          CCH Pounder
29                    Action|Adventure|Sci-Fi|Thriller  Bryce Dallas Howard
26                                     Drama|Romance    Leonardo DiCaprio
3024                   Action|Adventure|Fantasy|Sci-Fi        Harrison Ford
3080                                   Family|Sci-Fi         Henry Thomas
...                                              ...                  ...
2334  Action|Adventure|Animation|Family|Sci-Fi|Thriller     William Hootkins
2323                     Adventure|Animation|Fantasy         Minnie Driver
3005                                 Drama|Romance|War         Marcell Nagy
```

```
3859                                  Crime|Drama         Min-sik Choi
2988                       Comedy|Drama|Horror|Sci-Fi            Doona Bae

                            movie_title  num_voted_users  \
0                                Avatar           886204
29                      Jurassic World           418214
26                             Titanic           793059
3024  Star Wars: Episode IV - A New Hope           911097
3080         E.T. the Extra-Terrestrial           281842
...                                 ...              ...
2334                           Steamboy            13727
2323                  Princess Mononoke           221552
3005                           Fateless             5603
3859                      Lady Vengeance            53508
2988                            The Host            68883

      num_user_for_reviews   language        budget  title_year  imdb_score  \
0                   3054.0    English    237.000000      2009.0         7.9
29                  1290.0    English    150.000000      2015.0         7.0
26                  2528.0    English    200.000000      1997.0         7.7
3024                1470.0    English     11.000000      1977.0         8.7
3080                 515.0    English     10.500000      1982.0         7.9
...                    ...        ...           ...         ...         ...
2334                  79.0   Japanese   2127.519898      2004.0         6.9
2323                 570.0   Japanese   2400.000000      1997.0         8.4
3005                  45.0  Hungarian   2500.000000      2005.0         7.1
3859                 131.0     Korean   4200.000000      2005.0         7.7
2988                 279.0     Korean  12215.500000      2006.0         7.0

      movie_facebook_likes        Profit
0                    33000    523.505847
29                  150000    502.177271
26                   26000    458.672302
3024                 33000    449.935665
3080                 34000    424.449459
```

```
# Write your code to get the top 10 profiting movies here
Top10 = movies_3.head(10)
Top10
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	J
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Sta Episc A Ne
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	Te
17	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chris Hemsworth	Av
794	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chris Hemsworth	Av
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|Musical	Matthew Broderick	T
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|Sci-Fi	Natalie Portman	Sta Epi The P
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	Th

Subtask 3.3: Drop duplicate values

After you found out the top 10 profiting movies, you might have notice 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.`

```
# Write your code for dropping duplicate values here
display(movies_3.drop_duplicates())
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_tit
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Ava
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Juras Wo
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Tita
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wa Episode I A New Ho
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. Ext Terrest
...	...	...	...	...	...	
2334	Katsuhiro Ôtomo	105.0	0.410388	Action|Adventure|Animation|Family|Sci-Fi|Thriller	William Hootkins	Steamb
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantasy	Minnie Driver	Princ Monon
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|War	Marcell Nagy	Fatel
3859	Chan-wook Park	202.0	0.211667	Crime|Drama	Min-sik Choi	La Vengear
2988	Joon-ho Bong	363.0	2.201412	Comedy|Drama|Horror|Sci-Fi	Doona Bae	The H

3849 rows × 14 columns

```
# Write code for repeating subtask 2 here
display(Top10.drop_duplicates())
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	J
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Sta Episc A Ne
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	Te
17	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chris Hemsworth	Av
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|Musical	Matthew Broderick	T
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|Sci-Fi	Natalie Portman	Sta Epi The P
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	Th

Checkpoint 2: You might spot two movies directed by `James Cameron` in the list.

Subtask 3.4: Find IMDb Top 250

1. 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.
2. 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`.

```
# Write your code for extracting the top 250 movies as per the IMDb score here. Make sure that you store it in a new dataframe
# and name that dataframe as 'IMDb_Top_250
IMDb_Top_250=movies_3[movies_3['num_voted_users']>25000].sort_values(by='imdb_score',ascending=False).head(250)
IMDb_Top_250
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_titl
1937	Frank Darabont	199.0	28.341469	Crime|Drama	Morgan Freeman	Th Shawshar Redemptic
3466	Francis Ford Coppola	208.0	134.821952	Crime|Drama	Al Pacino	Th Godfath
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Da Knig
2837	Francis Ford Coppola	149.0	57.300000	Crime|Drama	Robert De Niro	Th Godfathe Part
1874	Steven Spielberg	174.0	96.067179	Biography|Drama|History	Liam Neeson	Schindler Li
...	...	...	...	...	...	
1171	Yimou Zhang	283.0	0.084961	Action|Adventure|History	Jet Li	He
1748	F. Gary Gray	349.0	161.029270	Biography|Crime|Drama|History|Music	Aldis Hodge	Straig Out Compto
788	Cameron Crowe	149.0	32.522352	Adventure|Comedy|Drama|Music	Philip Seymour Hoffman	Almo Famou
639	Michael Mann	209.0	28.965197	Biography|Drama|Thriller	Al Pacino	The Insid
525	Alfonso Cuarón	372.0	35.286428	Drama|Sci-Fi|Thriller	Charlie Hunnam	Children Me

250 rows × 14 columns

```
IMDb_Top_250['Rank']=IMDb_Top_250['imdb_score'].rank(method='first',ascending=False)
IMDb_Top_250
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_titl
1937	Frank Darabont	199.0	28.341469	Crime|Drama	Morgan Freeman	Th Shawshar Redemptic
3466	Francis Ford Coppola	208.0	134.821952	Crime|Drama	Al Pacino	Th Godfath
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Da Knig
2837	Francis Ford Coppola	149.0	57.300000	Crime|Drama	Robert De Niro	Th Godfathe Part
1874	Steven Spielberg	174.0	96.067179	Biography|Drama|History	Liam Neeson	Schindler Li
...	...	...	...	...	...	
1171	Yimou Zhang	283.0	0.084961	Action|Adventure|History	Jet Li	He

```
 # Write your code to extract top foreign language films from 'IMDb_Top_250' here
Top_Foreign_Lang_Film =IMDb_Top_250[IMDb_Top_250['language']!='English'] # Write your code to extract top foreign language films from 'IMDb_T
Top_Foreign_Lang_Film
```

Checkpoint 3: Can you spot `Veer-Zaara` in the dataframe?

Subtask 3.5: Find the best directors

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

4747 Akira Kurosawa 153.0 0.269061 Action|Adventure|Drama

```
# Write your code for extracting the top 10 directors here
top10director=movies.groupby('director_name').imdb_score.mean().sort_values(ascending=False)
top10director
```

```
director_name
John Blanchard        9.5
Mitchell Altieri      8.7
Sadyk Sher-Niyaz      8.7
Cary Bell             8.7
Mike Mayhall          8.6
                     ...
Georgia Hilton        2.2
Vondie Curtis-Hall    2.1
Frédéric Auburtin     2.0
A. Raven Cruz         1.9
Lawrence Kasanoff     1.7
Name: imdb_score, Length: 2398, dtype: float64
```

Checkpoint 4: No surprises that `Damien Chazelle` (director of Whiplash and La La Land) is in this list.

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.

1. 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` will be the same as `genre_1`.
2. Group the dataframe using `genre_1` as the primary column and `genre_2` as the secondary column.
3. 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`.

```
# Write your code for extracting the first two genres of each movie here
genre=movies_3.genres.str.split('|',expand=True).iloc[:,0:2]
genre.columns=['genre_1','genre_2']
genre.genre_2.fillna(genre.genre_1,inplace=True)
genre
```

	genre_1	genre_2
0	Action	Adventure
29	Action	Adventure
26	Drama	Romance
3024	Action	Adventure
3080	Family	Sci-Fi
...	...	...
2334	Action	Adventure
2323	Adventure	Animation
3005	Drama	Romance
3859	Crime	Drama
2988	Comedy	Drama

3884 rows × 2 columns

```
# Write your code for grouping the dataframe here
movies_by_segment =pd.concat([movies,genre],axis=1)
```

```
movies_by_segment
```

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

5043 rows × 15 columns

```
PropGenre=pd.DataFrame(movies_by_segment.gross.mean().sort_values(ascending=False) )  #Creating a dataframe
PropGenre[0:5]
```

Checkpoint 5: Well, as it turns out. `Family + Sci-Fi` is the most popular combo of genres out there!

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

1. 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.
2. Append the rows of all these dataframes and store them in a new dataframe named `Combined`.
3. Group the combined dataframe using the `actor_1_name` column.
4. Find the mean of the `num_critic_for_reviews` and `num_user_for_review` and identify the actors which have the highest mean.

```
# Write your code for creating three new dataframes here
Meryl_Streep =movies_3[movies_3['actor_1_name']=='Meryl Streep'] # Include all movies in which Meryl_Streep is the lead
Meryl_Streep
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	nu
1408	David Frankel	208.0	124.732962	Comedy|Drama|Romance	Meryl Streep	The Devil Wears Prada	
1575	Sydney Pollack	66.0	87.100000	Biography|Drama|Romance	Meryl Streep	Out of Africa	
1204	Nora Ephron	252.0	94.125426	Biography|Drama|Romance	Meryl Streep	Julie & Julia	
1618	David Frankel	234.0	63.536011	Comedy|Drama|Romance	Meryl Streep	Hope Springs	
410	Nancy Meyers	187.0	112.703470	Comedy|Drama|Romance	Meryl Streep	It's Complicated	
2781	Phyllida Lloyd	331.0	29.959436	Biography|Drama|History	Meryl Streep	The Iron Lady	
1925	Stephen Daldry	174.0	41.597830	Drama|Romance	Meryl Streep	The Hours	
3135	Robert Altman	211.0	20.338609	Comedy|Drama|Music	Meryl Streep	A Prairie Home Companion	

```
 # Include all movies in which Leo_Caprio is the lead
Leo_Caprio =movies_3[movies_3['actor_1_name']=='Leonardo DiCaprio'] # Include all movies in which Leo_Caprio is the lead
Leo_Caprio
```

```
 # Include all movies in which Brad_Pitt is the lead
Brad_Pitt =movies_3[movies_3['actor_1_name']=='Brad Pitt'] # Include all movies in which Brad_Pitt is the lead
Brad_Pitt
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_nam
400	Steven Soderbergh	186.0	183.405771	Crime|Thriller	Brad P
255	Doug Liman	233.0	186.336103	Action|Comedy|Crime|Romance|Thriller	Brad P
940	Neil Jordan	120.0	105.264608	Drama|Fantasy|Horror	Brad P
470	David Ayer	406.0	85.707116	Action|Drama|War	Brad P
254	Steven Soderbergh	198.0	125.531634	Crime|Thriller	Brad P
2204	Alejandro G. Iñárritu	285.0	34.300771	Drama	Brad P
2682	Andrew Dominik	414.0	14.938570	Crime|Thriller	Brad P
2898	Tony Scott	122.0	12.281500	Action|Crime|Drama|Romance|Thriller	Brad P
2333	Angelina Jolie Pitt	131.0	0.531009	Drama|Romance	Brad P
1490	Terrence Malick	584.0	13.303319	Drama|Fantasy	Brad P
101	David Fincher	362.0	127.490802	Drama|Fantasy|Romance	Brad P
683	David Fincher	315.0	37.023395	Drama	Brad P
1722	Andrew Dominik	273.0	3.904982	Biography|Crime|Drama|History|Western	Brad P
611	Jean-Jacques Annaud	76.0	37.901509	Adventure|Biography|Drama|History|War	Brad P
792	Patrick Gilmore	98.0	26.288320	Adventure|Animation|Comedy|Drama|Family|Fantas...	Brad P
147	Wolfgang Petersen	220.0	133.228348	Adventure	Brad P
382	Tony Scott	142.0	0.026871	Action|Crime|Thriller	Brad P

```
# Write your code for combining the three dataframes here
Combined=Meryl_Streep.append([Leo_Caprio,Brad_Pitt])
Combined
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_
1408	David Frankel	208.0	124.732962	Comedy|Drama|Romance	Meryl S
1575	Sydney Pollack	66.0	87.100000	Biography|Drama|Romance	Meryl S
1204	Nora Ephron	252.0	94.125426	Biography|Drama|Romance	Meryl S
1618	David Frankel	234.0	63.536011	Comedy|Drama|Romance	Meryl S
410	Nancy Meyers	187.0	112.703470	Comedy|Drama|Romance	Meryl S
2781	Phyllida Lloyd	331.0	29.959436	Biography|Drama|History	Meryl S
1925	Stephen Daldry	174.0	41.597830	Drama|Romance	Meryl S
3135	Robert Altman	211.0	20.338609	Comedy|Drama|Music	Meryl S
1106	Curtis Hanson	42.0	46.815748	Action|Adventure|Crime|Thriller	Meryl S
1674	Carl Franklin	64.0	23.209440	Drama	Meryl S
1483	Robert Redford	227.0	14.998070	Drama|Thriller|War	Meryl S
26	James Cameron	315.0	658.672302	Drama|Romance	Leor DiC
97	Christopher Nolan	642.0	292.568851	Action|Adventure|Sci-Fi|Thriller	Leor DiC
911	Steven Spielberg	194.0	164.435221	Biography|Crime|Drama	Leor DiC
296	Quentin Tarantino	765.0	162.804648	Drama|Western	Leor DiC
179	Alejandro G.	556.0	183.635922	Adventure|Drama|Thriller|Western	Leor

```
# Write your code for grouping the combined dataframe here
Combined
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_
1408	David Frankel	208.0	124.732962	Comedy|Drama|Romance	Meryl S
1575	Sydney Pollack	66.0	87.100000	Biography|Drama|Romance	Meryl S
1204	Nora Ephron	252.0	94.125426	Biography|Drama|Romance	Meryl S
1618	David Frankel	234.0	63.536011	Comedy|Drama|Romance	Meryl S
410	Nancy Meyers	187.0	112.703470	Comedy|Drama|Romance	Meryl S
2781	Phyllida Lloyd	331.0	29.959436	Biography|Drama|History	Meryl S
1925	Stephen Daldry	174.0	41.597830	Drama|Romance	Meryl S
3135	Robert Altman	211.0	20.338609	Comedy|Drama|Music	Meryl S

```
# Write the code for finding the mean of critic reviews and audience reviews here
Combined.groupby('actor_1_name').num_critic_for_reviews.mean()
```

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

```
Combined.num_user_for_reviews=Combined.num_user_for_reviews.astype('int')
Combined.num_user_for_reviews
```

```
1408     631
1575     200
1204     277
1618     178
410      214
2781     350
1925     660
3135     280
1106      69
1674     112
1483     298
26      2528
97      2803
911      667
296     1193
179     1188
452      964
361     2054
50       753
3476     753
2757     506
1422     244
308     1138
1453     279
257      799
2067      71
990      548
1114     414
1560     216
326     1166
641      263
307      657
400      845
255      798
940      406
470      701
254      627
2204     908
2682     369
2898     460
2333      61
1490     975
101      822
683     2968
1722     415
611      119
792       91
147     1694
382      361
Name: num_user_for_reviews, dtype: int64
```

```
Combined.groupby('actor_1_name').num_user_for_reviews.mean()
```

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

Checkpoint 6: `Leonardo` has aced both the lists!