```
# 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.

[ ] ↳ 46 cells hidden

Write code for repeating subtask 2 here\n"

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"

```
#1: Create a new column called profit which contains the difference of the two columns: gross and budget.
# Assuming your DataFrame is called 'movies_cleaned_deduplicated'

# We have ensure df_cleaned_deduplicated is defined (from the previous cell)
df_cleaned_deduplicated = df_cleaned.drop_duplicates().copy()  # Create an explicit copy using .copy()

# We have used .loc for assignment to avoid SettingWithCopyWarning
df_cleaned_deduplicated.loc[:, 'profit'] = df_cleaned_deduplicated['gross'] - df_cleaned_deduplicated['budget']# Write y
print(df_cleaned_deduplicated.loc[:, 'profit'])
```

```
0       523505847.0
1         9404152.0
2       -44925825.0
3       198130642.0
5      -190641321.0
           ...
5038           NaN
5039           NaN
5040           NaN
5041           NaN
5042       84122.0
Name: profit, Length: 4924, dtype: float64
```

```
#2: Sort the dataframe using the profit column as reference.

# Sort the DataFrame in descending order based on the 'profit' column
df_cleaned_deduplicated = df_cleaned_deduplicated.sort_values(by='profit', ascending=False)
print(df_cleaned_deduplicated)
```

```
       color     director_name  num_critic_for_reviews  duration  \
0      Color     James Cameron                   723.0     178.0
29     Color   Colin Trevorrow                   644.0     124.0
26     Color     James Cameron                   315.0     194.0
3024   Color      George Lucas                   282.0     125.0
3080   Color  Steven Spielberg                   215.0     120.0
...      ...               ...                     ...       ...
5036   Color   Anthony Vallone                    NaN      84.0
5038   Color       Scott Smith                    1.0      87.0
5039   Color               NaN                   43.0      43.0
5040   Color  Benjamin Roberds                   13.0      76.0
5041   Color       Daniel Hsia                   14.0     100.0

       director_facebook_likes  actor_3_facebook_likes      actor_2_name  \
0                          0.0                   855.0  Joel David Moore
29                       365.0                  1000.0       Judy Greer
26                         0.0                   794.0     Kate Winslet
3024                       0.0                   504.0     Peter Cushing
3080                   14000.0                   548.0       Dee Wallace
...                        ...                     ...              ...
5036                       2.0                     2.0    John Considine
5038                       2.0                   318.0     Daphne Zuniga
5039                       NaN                   319.0     Valorie Curry
5040                       0.0                     0.0     Maxwell Moody
5041                       0.0                   489.0     Daniel Henney
```

```
      actor_1_facebook_likes        gross                           genres  \
0                     1000.0  760505847.0   Action|Adventure|Fantasy|Sci-Fi
29                    3000.0  652177271.0  Action|Adventure|Sci-Fi|Thriller
26                   29000.0  658672302.0                     Drama|Romance
3024                 11000.0  460935665.0   Action|Adventure|Fantasy|Sci-Fi
3080                   861.0  434949459.0                     Family|Sci-Fi
...                      ...          ...                              ...
5036                    45.0          NaN                      Crime|Drama
5038                   637.0          NaN                     Comedy|Drama
5039                   841.0          NaN      Crime|Drama|Mystery|Thriller
5040                     0.0          NaN             Drama|Horror|Thriller
5041                   946.0      10443.0              Comedy|Drama|Romance

      ... language country  content_rating       budget title_year  \
0     ...  English     USA           PG-13  237000000.0     2009.0
29    ...  English     USA           PG-13  150000000.0     2015.0
26    ...  English     USA           PG-13  200000000.0     1997.0
3024  ...  English     USA              PG   11000000.0     1977.0
3080  ...  English     USA              PG   10500000.0     1982.0
...   ...      ...     ...             ...          ...        ...
5036  ...  English     USA           PG-13       3250.0     2005.0
5038  ...  English  Canada             NaN          NaN     2013.0
5039  ...  English     USA           TV-14          NaN        NaN
5040  ...  English     USA             NaN       1400.0     2013.0
5041  ...  English     USA           PG-13          NaN     2012.0

      actor_2_facebook_likes imdb_score aspect_ratio  movie_facebook_likes  \
0                      936.0        7.9         1.78                 33000
29                    2000.0        7.0         2.00                150000
26                   14000.0        7.7         2.35                 26000
3024                  1000.0        8.7         2.35                 33000
3080                   725.0        7.9         1.85                 34000
```

#3:Extract the top ten profiting movies in descending order and store them in a new dataframe - top10"

```
# Extract the top 10 profiting movies
top10_cleaned_deduplicated= df_cleaned_deduplicated.head(10)
print(top10_cleaned_deduplicated)
```

```
      color      director_name  num_critic_for_reviews  duration  \
0     Color      James Cameron                   723.0     178.0
29    Color    Colin Trevorrow                   644.0     124.0
26    Color      James Cameron                   315.0     194.0
3024  Color       George Lucas                   282.0     125.0
3080  Color   Steven Spielberg                   215.0     120.0
17    Color        Joss Whedon                   703.0     173.0
509   Color       Roger Allers                   186.0      73.0
240   Color       George Lucas                   320.0     136.0
66    Color  Christopher Nolan                   645.0     152.0
439   Color          Gary Ross                   673.0     142.0

      director_facebook_likes  actor_3_facebook_likes       actor_2_name  \
0                         0.0                   855.0   Joel David Moore
29                      365.0                  1000.0         Judy Greer
26                        0.0                   794.0       Kate Winslet
3024                      0.0                   504.0      Peter Cushing
3080                  14000.0                   548.0        Dee Wallace
17                        0.0                 19000.0  Robert Downey Jr.
509                      28.0                   847.0        Nathan Lane
240                       0.0                  1000.0        Liam Neeson
66                    22000.0                 11000.0       Heath Ledger
439                     378.0                   575.0    Josh Hutcherson

      actor_1_facebook_likes        gross  \
0                     1000.0  760505847.0
29                    3000.0  652177271.0
26                   29000.0  658672302.0
3024                 11000.0  460935665.0
3080                   861.0  434949459.0
17                   26000.0  623279547.0
509                   2000.0  422783777.0
240                  20000.0  474544677.0
66                   23000.0  533316061.0
439                  34000.0  407999255.0

                                  genres  ... language country  \
0             Action|Adventure|Fantasy|Sci-Fi  ...  English     USA
29           Action|Adventure|Sci-Fi|Thriller  ...  English     USA
26                           Drama|Romance  ...  English     USA
3024          Action|Adventure|Fantasy|Sci-Fi  ...  English     USA
3080                         Family|Sci-Fi  ...  English     USA
17                   Action|Adventure|Sci-Fi  ...  English     USA
509   Adventure|Animation|Drama|Family|Musical  ...  English     USA
240           Action|Adventure|Fantasy|Sci-Fi  ...  English     USA
66                Action|Crime|Drama|Thriller  ...  English     USA
439           Adventure|Drama|Sci-Fi|Thriller  ...  English     USA

      content_rating       budget title_year  actor_2_facebook_likes  \
```

```
0             PG-13  237000000.0     2009.0                936.0
29            PG-13  150000000.0     2015.0               2000.0
26            PG-13  200000000.0     1997.0              14000.0
3024             PG   11000000.0     1977.0               1000.0
3080             PG   10500000.0     1982.0                725.0
17            PG-13  220000000.0     2012.0              21000.0
509               G   45000000.0     1994.0                886.0
240              PG  115000000.0     1999.0              14000.0
66            PG-13  185000000.0     2008.0              13000.0
```

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

Answer :Yes there are two movies directed by James Cameron. 0 = Action|Adventure|Fantasy| Sci-Fi

26 = Drama|Romance

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

```
#1: Write your code for extracting the top 250 movies as per the IMDb score here. Make sure that you store it in a new
# and name that dataframe as 'IMDb_Top_250'
top_250_movies = movies_cleaned.sort_values(by='imdb_score', ascending=False).head(250) #Fixed the indentation and appl
top_250_movies = top_250_movies[top_250_movies['num_voted_users'] > 25000]
top_250_movies['Rank'] = range(1,len(top_250_movies) +1)
IMDb_Top_250 = top_250_movies
print(IMDb_Top_250)
```

```
                color         director_name  num_critic_for_reviews  \
1937            Color        Frank Darabont                   199.0
3466            Color  Francis Ford Coppola                   208.0
66              Color     Christopher Nolan                   645.0
2837            Color  Francis Ford Coppola                   149.0
339             Color         Peter Jackson                   328.0
...               ...                   ...                     ...
2619            Color        John Carpenter                   318.0
2493   Black and White           Yimou Zhang                   283.0
2492            Color        John Carpenter                   318.0
353             Color         John Lasseter                   191.0
1807            Color           John Landis                   125.0

      duration  director_facebook_likes  actor_3_facebook_likes  \
1937     142.0                      0.0                   461.0
3466     175.0                      0.0                  3000.0
66       152.0                  22000.0                 11000.0
2837     220.0                      0.0                  3000.0
339      192.0                      0.0                   416.0
...        ...                      ...                     ...
2619     101.0                      0.0                   598.0
2493      80.0                    611.0                   576.0
2492     101.0                      0.0                   598.0
353       82.0                    487.0                   967.0
1807     148.0                    644.0                   326.0

             actor_2_name  actor_1_facebook_likes        gross  \
1937       Jeffrey DeMunn                 11000.0   28341469.0
3466        Marlon Brando                 14000.0  134821952.0
66           Heath Ledger                 23000.0  533316061.0
2837            Al Pacino                 22000.0   57300000.0
339            Billy Boyd                  5000.0  377019252.0
...                   ...                     ...          ...
2619     Donald Pleasence                  2000.0   47000000.0
2493  Tony Chiu Wai Leung                  5000.0      84961.0
2492     Donald Pleasence                  2000.0   47000000.0
353     John Ratzenberger                 15000.0  245823397.0
1807      Aretha Franklin                  1000.0   54200000.0

                                      genres  ...       budget title_year  \
1937                            Crime|Drama  ...   25000000.0     1994.0
3466                            Crime|Drama  ...    6000000.0     1972.0
66                Action|Crime|Drama|Thriller  ...  185000000.0     2008.0
2837                            Crime|Drama  ...   13000000.0     1974.0
339             Action|Adventure|Drama|Fantasy  ...   94000000.0     2003.0
...                                        ...  ...          ...        ...
2619                         Horror|Thriller  ...     300000.0     1978.0
2493                  Action|Adventure|History  ...   31000000.0     2002.0
2492                         Horror|Thriller  ...     300000.0     1978.0
353   Adventure|Animation|Comedy|Family|Fantasy  ...   90000000.0     1999.0
```

```
1807                 Action|Comedy|Crime|Music  ...   27000000.0     1980.0

      actor_2_facebook_likes  imdb_score aspect_ratio  movie_facebook_likes  \
1937                   745.0         9.3         1.85                108000
3466                 10000.0         9.2         1.85                 43000
66                   13000.0         9.0         2.35                 37000
2837                 14000.0         9.0         1.85                 14000
339                    857.0         8.9         2.35                 16000
```

```
#2: Top_Foreign_Lang_Film = # 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']
print(Top_Foreign_Lang_Film)
```

```
                color                    director_name  \
4498            Color                     Sergio Leone
4747  Black and White                    Akira Kurosawa
4029            Color                Fernando Meirelles
2373            Color                    Hayao Miyazaki
4921            Color                      Majid Majidi
4259            Color  Florian Henckel von Donnersmarck
4659            Color                    Asghar Farhadi
2323            Color                    Hayao Miyazaki
2970            Color                 Wolfgang Petersen
4105            Color                    Chan-wook Park
1298  Black and White                Jean-Pierre Jeunet
4033            Color                 Thomas Vinterberg
2829            Color               Oliver Hirschbiegel
2734  Black and White                         Fritz Lang
3550            Color                   Denis Villeneuve
2551            Color                Guillermo del Toro
4000            Color              Juan José Campanella
2047            Color                    Hayao Miyazaki
2830            Color                 Alejandro Amenábar
4267            Color             Alejandro G. Iñárritu
2914            Color                       Je-kyu Kang
3423            Color                   Katsuhiro Ôtomo
3553            Color                      José Padilha
4461            Color                 Thomas Vinterberg
3456            Color                 Vincent Paronnaud
3344            Color                       Karan Johar
4897            Color                     Sergio Leone
4144            Color                    Walter Salles
4284            Color                       Ari Folman
3677            Color              Christophe Barratier
1171  Black and White                       Yimou Zhang
2863            Color                    Clint Eastwood
2605            Color                           Ang Lee
2493  Black and White                       Yimou Zhang

      num_critic_for_reviews  duration  director_facebook_likes  \
4498                   181.0     142.0                      0.0
4747                   153.0     202.0                      0.0
4029                   214.0     135.0                    353.0
2373                   246.0     125.0                   6000.0
4921                    46.0      89.0                    373.0
4259                   215.0     137.0                    207.0
4659                   354.0     123.0                      0.0
2323                   174.0     134.0                   6000.0
2970                    96.0     293.0                    249.0
4105                   305.0     120.0                      0.0
1298                   242.0     122.0                      0.0
4033                   349.0     115.0                    346.0
2829                   192.0     178.0                    101.0
2734                   260.0     145.0                    756.0
3550                   226.0     139.0                    777.0
2551                   406.0     112.0                      0.0
4000                   262.0     129.0                    195.0
2047                   212.0     119.0                   6000.0
2830                   157.0     125.0                    448.0
4267                   157.0     115.0                      0.0
2914                    86.0     148.0                     16.0
```

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

```
if('veer zara is in  Top_Foreign_Lang_Film'):
  print('Veer Zara is among 250 IMdB top 250 movies')
else:
  print('not found')
```

```
Veer Zara is among 250 IMdB top 250 movies
```

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.

```
import pandas as pd

# Load the dataset
file_path = "Movie+Assignment+Data.csv"  # Update with the correct file path
df = pd.read_csv(file_path)

# Remove rows with more than 5 NaN values
df_cleaned = df.dropna(thresh=len(df.columns) - 5)

# Remove duplicate rows
df_cleaned_deduplicated = df_cleaned.drop_duplicates()

# Find the top 10 directors with the highest mean IMDb score
top10director = (
    df_cleaned_deduplicated.groupby("director_name")["imdb_score"]
    .mean()
    .nlargest(10)
    .reset_index()
)

# Display the top 10 directors
print(top10director)
```

```
        director_name  imdb_score
0          Cary Bell       8.700
1   Sadyk Sher-Niyaz       8.700
2    Charles Chaplin       8.600
3    Damien Chazelle       8.500
4       Majid Majidi       8.500
5         Raja Menon       8.500
6         Ron Fricke       8.500
7       Sergio Leone       8.475
8          Tony Kaye       8.450
9  Christopher Nolan       8.425
```

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

```
movies_by_segment = []# Write your code for grouping the dataframe here"


# Extract the first two genres using str.split and expand=True
movies_by_segment = []# Write your code for grouping the dataframe here"


# Extract the first two genres using str.split and expand=True
#'|', n=1: This part splits the string based on the pipe symbol ('|') with a maximum of 1 split (to get the first two g
# expand=True: This creates separate columns for the split results.
movies_cleaned[['genre_1', 'genre_2']] = movies_cleaned['genres'].str.split('|', n=1, expand=True)

# Fill NaN values in genre_2 with genre_1 for movies with only one genre
# fillna(movies_cleaned['genre_1']): It fills any missing values (NaN) in the 'genre_2' column with the corresponding v
# This ensures that movies with only one genre have that genre in both 'genre_1' and 'genre_2'.
```

```
movies_cleaned['genre_2'] = movies_cleaned['genre_2'].fillna(movies_cleaned['genre_1'])
print(movies_cleaned['genre_2']) # Changed 'genere_2' to 'genre_2'
```

```
0          Adventure|Fantasy|Sci-Fi
1                 Adventure|Fantasy
2                Adventure|Thriller
3                          Thriller
5                   Adventure|Sci-Fi
                      ...
5026                   Music|Romance
5027                           Drama
5033                  Sci-Fi|Thriller
5035    Crime|Drama|Romance|Thriller
5042                     Documentary
Name: genre_2, Length: 3784, dtype: object
```

```
PopGenre = None # Write your code for getting the 5 most popular combo of genres here


# Group the dataframe by genre_1 and genre_2
movies_by_genre = movies_cleaned.groupby(['genre_1', 'genre_2'])

# Calculate the mean gross for each genre combination, sort, and get the top 5
PopGenre = movies_by_genre['gross'].mean().sort_values(ascending=False).head(5).reset_index()

# Rename columns for clarity
PopGenre.columns = ['genre_1', 'genre_2', 'mean_gross']
# Group the dataframe by genre_1 and genre_2
movies_by_genre = movies_cleaned.groupby(['genre_1', 'genre_2'])

# Calculate the mean gross for each genre combination, sort, and get the top 5
PopGenre = movies_by_genre['gross'].mean().sort_values(ascending=False).head(5).reset_index()

# Rename columns for clarity
PopGenre.columns = ['genre_1', 'genre_2', 'mean_gross']
print(PopGenre) # Changed 'PopGenere' to 'PopGenre'
```

```
      genre_1                              genre_2    mean_gross
0      Family                               Sci-Fi  4.349495e+08
1   Adventure        Animation|Drama|Family|Musical  4.227838e+08
2   Adventure  Animation|Comedy|Drama|Family|Fantasy  3.564544e+08
3      Action   Biography|Drama|History|Thriller|War  3.501236e+08
4      Action               Adventure|Fantasy|Sci-Fi  2.966848e+08
```

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

Answer : Yes Family+ Sci-Fi is the most popular combo of genres.

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.1:

```
[] # Write your code for creating three new dataframes here

Meryl_Streep = movies_cleaned[movies_cleaned['actor_1_name'] == 'Meryl Streep']
Leo_Caprio = movies_cleaned[movies_cleaned['actor_1_name'] == 'Leonardo DiCaprio']
Brad_Pitt = movies_cleaned[movies_cleaned['actor_1_name'] == 'Brad Pitt']


 # Include all movies in which Meryl_Streep is the lead"
 # Include all movies in which Meryl_Streep is the lead
Meryl_Streep = movies_cleaned[movies_cleaned['actor_1_name'] == 'Meryl Streep']
print(Meryl_Streep['movie_title'])  # Access the 'movie_title' column using bracket notation
```

```
410            It's Complicated
1106             The River Wild
1204              Julie & Julia
1408       The Devil Wears Prada
1483             Lions for Lambs
1575               Out of Africa
```

```
1618               Hope Springs
1674             One True Thing
1925                  The Hours
2781              The Iron Lady
3135    A Prairie Home Companion
Name: movie_title, dtype: object
```

```
# Include all movies in which Leo_Caprio is the lead
Leo_Caprio = movies_cleaned[movies_cleaned['actor_1_name'] == 'Leonardo DiCaprio']
print(Leo_Caprio ['movie_title'])  # Access the 'movie_title' column using bracket notation
```

```
26                      Titanic
50              The Great Gatsby
97                     Inception
179                 The Revenant
257                  The Aviator
296             Django Unchained
307                Blood Diamond
308      The Wolf of Wall Street
326            Gangs of New York
361                 The Departed
452                Shutter Island
641                  Body of Lies
911           Catch Me If You Can
990                      The Beach
1114           Revolutionary Road
1422    The Man in the Iron Mask
1453                     J. Edgar
1560      The Quick and the Dead
2067                Marvin's Room
2757               Romeo + Juliet
3476            The Great Gatsby
Name: movie_title, dtype: object
```

```
Brad_Pitt =[] # Include all movies in which Brad_Pitt is the lead
# Include all movies in which Brad_Pitt is the lead
Brad_Pitt = movies_cleaned[movies_cleaned['actor_1_name'] == 'Brad_Pitt ']
print(Brad_Pitt ['movie_title'])  # Access the 'movie_title' column using bracket notation
```

```
Series([], Name: movie_title, dtype: object)
```

```
# Write your code for creating three new dataframes here

Meryl_Streep = movies_cleaned[movies_cleaned['actor_1_name'] == 'Meryl Streep']
Leo_Caprio = movies_cleaned[movies_cleaned['actor_1_name'] == 'Leonardo DiCaprio']
Brad_Pitt = movies_cleaned[movies_cleaned['actor_1_name'] == 'Brad Pitt']
# Write your code for grouping the combined dataframe here
Combined = pd.concat([Meryl_Streep, Leo_Caprio, Brad_Pitt]) #Concatenate the three dataframes
actor_group = Combined.groupby('actor_1_name')  #Group by 'actor_1_name'
print(actor_group)
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7ebce0834490>
```

```
 # Write the code for finding the mean of critic reviews and audience reviews here

 # Write the code for finding the mean of critic reviews and audience reviews here
# Write your code for creating three new dataframes here

Meryl_Streep = movies_cleaned[movies_cleaned['actor_1_name'] == 'Meryl Streep']
Leo_Caprio = movies_cleaned[movies_cleaned['actor_1_name'] == 'Leonardo DiCaprio']
Brad_Pitt = movies_cleaned[movies_cleaned['actor_1_name'] == 'Brad Pitt']

# Write your code for grouping the combined dataframe here
Combined = pd.concat([Meryl_Streep, Leo_Caprio, Brad_Pitt]) #Concatenate the three dataframes
actor_group = Combined.groupby('actor_1_name')  #Group by 'actor_1_name'


# Calculate the mean of critic and audience reviews for each actor
critic_reviews_mean = actor_group['num_critic_for_reviews'].mean()
audience_reviews_mean = actor_group['num_user_for_reviews'].mean()

# Print the results
print("Mean Critic Reviews:")
print(critic_reviews_mean)
print("\nMean Audience Reviews:")
print(audience_reviews_mean)
```

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

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

Double-click (or enter) to edit

Checkpoint 6: Leonardo has aced both the lists

Yes, Leonardo has aced in both the lists.