Practice Exercise 2

In this assignment, you will try to find some interesting insights into a few movies released between 1916 and 2016, using Python. You will have to download a movie dataset, write Python code to explore the data, gain insights into the movies, actors, directors, and collections, and submit the code.

Some tips before starting the assignment

1. Identify the task to be performed correctly, and only then proceed to write the required code. Don't perform any incorrect analysis or look for information that isn't required for the assignment.
2. In some cases, the variable names have already been assigned, and you just need to write code against them. In other cases, the names to be given are mentioned in the instructions. We strongly advise you to use the mentioned names only.
3. Always keep inspecting your data frame after you have performed a particular set of operations.
4. There are some checkpoints given in the IPython notebook provided. They're just useful pieces of information you can use to check if the result you have obtained after performing a particular task is correct or not.
5. Note that you will be asked to refer to documentation for solving some of the questions. That is done on purpose for you to learn new commands and also how to use the documentation.

In [1]:
```
# 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`.

In [2]:
```
movies = pd.read_csv("Movies.csv")
movies
```

Out[2]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gr
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	7605058
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	3094041
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	2000741
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	4481306
4	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	640.0	730586
...	...	...	...	...	...	...	...	...	
3848	Color	Shane Carruth	143.0	77.0	291.0	8.0	David Sullivan	291.0	4247
3849	Color	Neill Dela Llana	35.0	80.0	0.0	0.0	Edgar Tancangco	0.0	700
3850	Color	Robert Rodriguez	56.0	81.0	0.0	6.0	Peter Marquardt	121.0	20409
3851	Color	Edward Burns	14.0	95.0	0.0	133.0	Caitlin FitzGerald	296.0	45
3852	Color	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	86.0	852

3853 rows × 28 columns

Subtask 1.2: Inspect the dataframe

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

Question 1: How many rows and columns are present in the dataframe?

- (3821, 26)
- (3879, 28)

- (3853, 28)
- (3866, 26)

In [3]:
```
movies.shape
```

Out[3]:
```
(3853, 28)
```

Question 2: How many columns have null values present in them? Try writing a code for this instead of counting them manually.

- 3
- 6
- 9
- 12

In [4]:
```
len(movies.columns[movies.isna().any()])
```

Out[4]:
```
12
```

Task 2: Cleaning the Data

Subtask 2.1: 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 [5]:
```
column_list = ["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"]
movies1 = movies.drop(column_list,axis=1)
```

In [6]:
```
movies1
```

Out[6]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_revi
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	30
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	12

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_revi
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	9
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	27
4	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	7
...	...	...	...	...	...	...	...	
3848	Shane Carruth	143.0	424760.0	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	3
3849	Neill Dela Llana	35.0	70071.0	Thriller	Ian Gamazon	Cavite	589	
3850	Robert Rodriguez	56.0	2040920.0	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	1
3851	Edward Burns	14.0	4584.0	Comedy|Drama	Kerry Bishé	Newlyweds	1338	
3852	Jon Gunn	43.0	85222.0	Documentary	John August	My Date with Drew	4285	

3853 rows × 13 columns

Question 3: What is the count of columns in the new dataframe?

- 10
- 13
- 15
- 17

In [7]:
```
len(movies1.columns)
```

Out[7]:
```
13
```

Subtask 2.2: Inspect Null values

As you have seen above, there are null values in multiple columns of the dataframe 'movies'. Find out the percentage of null values in each column of the dataframe 'movies'.

In [8]:
```
movies1.isna().any()
```

Out[8]:
```
director_name              False
num_critic_for_reviews      True
gross                      False
genres                     False
actor_1_name               False
movie_title                False
num_voted_users            False
num_user_for_reviews       False
language                    True
budget                     False
title_year                 False
imdb_score                 False
movie_facebook_likes       False
dtype: bool
```

Question 4: Which column has the highest percentage of null values?

- language
- genres
- num_critic_for_reviews
- imdb_score

In [9]:
```
movies1.isna().mean()
```

Out[9]:
```
director_name              0.000000
num_critic_for_reviews     0.000260
gross                      0.000000
genres                     0.000000
actor_1_name               0.000000
movie_title                0.000000
num_voted_users            0.000000
num_user_for_reviews       0.000000
language                   0.000779
budget                     0.000000
```

```
title_year                0.000000
imdb_score                0.000000
movie_facebook_likes      0.000000
dtype: float64
```

Subtask 2.3: 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 [10]:
```
movies1["language"] = movies1["language"].fillna("English")
movies1.isna().any()
```

Out[10]:
```
director_name              False
num_critic_for_reviews      True
gross                      False
genres                     False
actor_1_name               False
movie_title                False
num_voted_users            False
num_user_for_reviews       False
language                   False
budget                     False
title_year                 False
imdb_score                 False
movie_facebook_likes       False
dtype: bool
```

Question 5: What is the count of movies made in English language after replacing the NaN values with English?

- 3670
- 3674
- 3668
- 3672

In [11]:
```
len(movies1["language"][movies1["language"]=="English"]) # Alternative 1
```

Out[11]:
```
3674
```

In [12]:
```
(movies1.language == 'English').sum() # Alternative 2
```

Out[12]:
```
3674
```

Task 3: Data Analysis

Subtask 3.1: Change the unit of columns

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

In [13]:
```
movies1[['budget','gross']] = movies1[['budget','gross']]/1000000
```

In [14]:
```
movies1
```

Out[14]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_revie
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	305
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	123
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	99
3	Christopher Nolan	813.0	448.130642	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	270
4	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	73
...	...	...	...	...	...	...	...	
3848	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	37

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_revie
3849	Neill Dela Llana	35.0	0.070071	Thriller	Ian Gamazon	Cavite	589	3
3850	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	13
3851	Edward Burns	14.0	0.004584	Comedy|Drama	Kerry Bishé	Newlyweds	1338	1
3852	Jon Gunn	43.0	0.085222	Documentary	John August	My Date with Drew	4285	8

3853 rows × 13 columns

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. (Find which command can be used here to sort entries from the documentation)
3. Extract the top ten profiting movies in descending order and store them in a new dataframe - `top10`

In [15]:
```
movies1['profit']=movies1['gross']-movies1['budget']
```

In [16]:
```
top10 = movies1.sort_values('profit',ascending=False).head(10)
```

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

Question 6: Which movie is ranked 5th from the top in the list obtained?

- E.T. the Extra-Terrestrial
- The Avengers
- The Dark Knight
- Titanic

In [17]:
```
top10.iloc[4]["movie_title"]
```

Out[17]:
```
'E.T. the Extra-Terrestrial\xa0'
```

Subtask 3.3: 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.

In [18]:
```
movies1.head(0)
```

Out[18]:

director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	language	budget	title_year	in

In [19]:
```
ordered_imdb_score = (movies1.sort_values('imdb_score',ascending=False))
IMDb_Top_250 = ordered_imdb_score[ordered_imdb_score['num_voted_users']>25000][:250]
len(IMDb_Top_250)
```

Out[19]:
```
250
```

In [20]:
```
IMDb_Top_250
```

Out[20]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_f
1795	Frank Darabont	199.0	28.341469	Crime|Drama	Morgan Freeman	The Shawshank Redemption	1689764	
3016	Francis Ford Coppola	208.0	134.821952	Crime|Drama	Al Pacino	The Godfather	1155770	
2543	Francis Ford Coppola	149.0	57.300000	Crime|Drama	Robert De Niro	The Godfather: Part II	790926	
64	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Dark Knight	1676169	

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_f
325	Peter Jackson	328.0	377.019252	Action|Adventure|Drama|Fantasy	Orlando Bloom	The Lord of the Rings: The Return of the King	1215718	
...	...	...	...	...	...	...	...	
2708	David O. Russell	410.0	93.571803	Biography|Drama|Sport	Christian Bale	The Fighter	275869	
22	Peter Jackson	509.0	258.355354	Adventure|Fantasy	Aidan Turner	The Hobbit: The Desolation of Smaug	483540	
1612	James Mangold	291.0	119.518352	Biography|Drama|Music|Romance	Sandra Ellis Lafferty	Walk the Line	188637	
3237	Duncan Jones	415.0	5.009677	Drama|Mystery|Sci-Fi	Kevin Spacey	Moon	260607	
874	Andrew Adamson	212.0	267.652016	Adventure|Animation|Comedy|Family|Fantasy	Kathleen Freeman	Shrek	467113	

250 rows × 14 columns

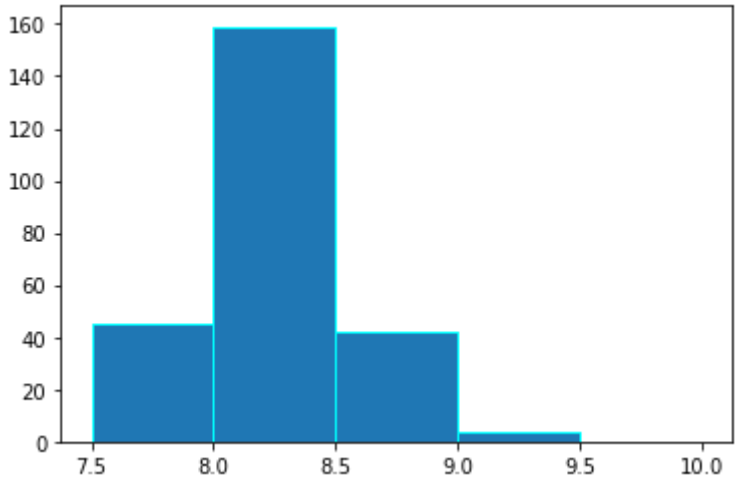
Question 7: Suppose movies are divided into 5 buckets based on the IMDb ratings:

- 7.5 to 8
- 8 to 8.5
- 8.5 to 9
- 9 to 9.5
- 9.5 to 10

Which bucket holds the maximum number of movies from *IMDb_Top_250*?

Alternative 1

In [21]:
```
import matplotlib.pyplot as plt
plt.hist(IMDb_Top_250['imdb_score'], bins = 5, range = (7.5,10), edgecolor = 'cyan')
plt.show()
```



Alternative 2

In [22]:
```
def bucketing(x):
    if 7.5<=x<8:
        return "7.5 to 8"
    elif 8<=x<8.5:
        return "8 to 8.5"
    elif 8.5<=x<9:
        return "8.5 to 9"
    elif 9<=x<9.5:
        return "9 to 9.5"
    elif 9.5<=x<=10:
        return "9.5 to 10"
IMDb_Top_250["buckets"] = IMDb_Top_250["imdb_score"].apply(bucketing)
```

In [23]:
```
IMDb_Top_250[["imdb_score","buckets"]].groupby("buckets")["imdb_score"].count()
```

Out[23]:
```
buckets
7.5 to 8     45
8 to 8.5    159
```

```
8.5 to 9     42
9 to 9.5      4
Name: imdb_score, dtype: int64
```

Subtask 3.4: 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.

In [24]:
```
movies1.head(0)
```

Out[24]:

director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	language	budget	title_year	in

In [25]:
```
# Write your code for creating three new dataframes here
Meryl_Streep = movies1[movies1["actor_1_name"]=="Meryl Streep"]
```

In [26]:
```
Leo_Caprio = movies1[movies1["actor_1_name"]=="Leonardo DiCaprio"]
```

In [27]:
```
Brad_Pitt = movies1[movies1["actor_1_name"]=="Brad Pitt"]
```

In [28]:
```
len(Meryl_Streep)
```

Out[28]:
```
11
```

In [29]:
```
len(Leo_Caprio)
```

Out[29]:
```
21
```

In [30]:
```
len(Brad_Pitt)
```

Out[30]:
```
17
```

In [31]:
```
Combined = Meryl_Streep.append(Leo_Caprio).append(Brad_Pitt)
```

In [32]:
```
Combined.groupby("actor_1_name").head(0)
```

Out[32]:

director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	language	budget	title_year	in

Alternative 1

In [33]:
```
Combined.groupby("actor_1_name").mean()[["num_critic_for_reviews","num_user_for_reviews"]]
```

Out[33]:

actor_1_name	num_critic_for_reviews	num_user_for_reviews
Brad Pitt	245.000000	742.352941
Leonardo DiCaprio	330.190476	914.476190
Meryl Streep	181.454545	297.181818

Alternative 2

In [34]:
```
Combined.pivot_table(index="actor_1_name",values=["num_critic_for_reviews","num_user_for_reviews"],aggfunc="mean")
```

Out[34]:

actor_1_name	num_critic_for_reviews	num_user_for_reviews
Brad Pitt	245.000000	742.352941
Leonardo DiCaprio	330.190476	914.476190

actor_1_name	num_critic_for_reviews	num_user_for_reviews
Meryl Streep	181.454545	297.181818

Question 8: Which actor is highest rated among the three actors according to the user reviews?

- Meryl Streep
- Leonardo DiCaprio
- Brad Pitt

refer above output table

Answer:-

Leonardo DiCaprio

Question 9: Which actor is highest rated among the three actors according to the critics?

- **Meryl Streep**
- **Leonardo DiCaprio**
- **Brad Pitt**

refer above output table

Answer:-

Leonardo DiCaprio
