

Name : Sanket Kakde

Roll No : CS1-31

PRN : 202401040245

EDS Activity 1 : IMDB dataset

Problem statements :

- 1) Find the first number of rows and columns in the dataset.

```
# Load dataset
df = pd.read_csv('IMDB-Movie-Data.csv')

# 1. Find the number of rows and columns in the dataset.
shape = df.shape
print(shape)
```

(1000, 12)

- 2) Display the first 5 rows of the dataset.

```
[13] # 2. Display the first 5 rows of the dataset.
head_5 = df.head()
print(head_5)
```

| | Rank | Title | Genre | Description | Director |
|---|------|-------------------------|--------------------------|---|----------------------|
| 0 | 1 | Guardians of the Galaxy | Action,Adventure,Sci-Fi | A group of intergalactic criminals are forced... | James Gunn |
| 1 | 2 | Prometheus | Adventure,Mystery,Sci-Fi | Following clues to the origin of mankind, a te... | Ridley Scott |
| 2 | 3 | Split | Horror,Thriller | Three girls are kidnapped by a man with a diag... | M. Night Shyamalan |
| 3 | 4 | King | Animation,Comedy,Family | In a city of humanoid animals, a hustling thea... | Christophe Lourdelet |
| 4 | 5 | Suicide Squad | Action,Adventure,Fantasy | A secret government agency recruits some of th... | David Ayer |

| | Actors | Year | Runtime (Minutes) |
|---|--|------|-------------------|
| 0 | Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S... | 2014 | 121 |
| 1 | Noomi Rapace, Logan Marshall-Green, Michael Fa... | 2012 | 124 |
| 2 | James McAvoy, Anya Taylor-Joy, Haley Lu Richar... | 2016 | 117 |
| 3 | Matthew McConaughey, Reese Witherspoon, Seth Ma... | 2016 | 108 |
| 4 | Will Smith, Jared Leto, Margot Robbie, Viola D... | 2016 | 123 |

| | Rating | Votes | Revenue (Millions) | Metascore |
|---|--------|--------|--------------------|-----------|
| 0 | 8.1 | 757074 | 333.13 | 76.0 |
| 1 | 7.0 | 485820 | 126.46 | 65.0 |
| 2 | 7.3 | 157606 | 138.12 | 62.0 |
| 3 | 7.2 | 68545 | 270.32 | 59.0 |
| 4 | 6.2 | 393727 | 325.02 | 40.0 |

- 3) Find the list of unique genres in the dataset.

```
[14] # 3. Find the list of unique genres.
unique_genres = df['Genre'].unique()
print(unique_genres)
```

['Action,Adventure,Sci-Fi' 'Adventure,Mystery,Sci-Fi' 'Horror,Thriller'
'Animation,Comedy,Family' 'Action,Adventure,Fantasy' 'Comedy,Drama,Musical'
'Comedy' 'Action,Adventure,Biography' 'Adventure,Drama,Romance'
'Adventure,Family,Fantasy' 'Biography,Drama,History'
'Animation,Adventure,Comedy' 'Action,Comedy,Drama' 'Action,Thriller'
'Biography,Drama' 'Drama,Mystery,Sci-Fi' 'Adventure,Drama,Thriller'
'Drama' 'Crime,Drama,Horror' 'Action,Adventure,Drama' 'Drama,Thriller'
'Action,Adventure,Comedy' 'Action,Horror,Sci-Fi' 'Adventure,Drama,Sci-Fi'
'Action,Adventure,Western' 'Comedy,Drama' 'Horror'
'Adventure,Drama,Fantasy' 'Action,Crime,Thriller' 'Action,Crime,Drama'
'Adventure,Drama,History' 'Crime,Horror,Thriller' 'Drama,Romance'
'Comedy,Drama,Romance' 'Horror,Mystery,Thriller' 'Crime,Drama,Mystery'
'Drama,Romance,Thriller' 'Drama,History,Thriller' 'Action,Drama,Thriller'
'Drama,History' 'Action,Drama,Romance' 'Drama,Fantasy' 'Action,Sci-Fi'
'Adventure,Drama,War' 'Action,Comedy,Fantasy' 'Biography,Comedy,Crime'
'Crime,Drama' 'Comedy,Crime,Drama' 'Action,Comedy,Crime'
'Animation,Drama,Fantasy' 'Horror,Mystery,Sci-Fi'
'Drama,Mystery,Thriller' 'Crime,Drama,Thriller' 'Biography,Crime,Drama'
'Crime,Mystery,Thriller' 'Action,Horror,Thriller' 'Romance,Sci-Fi'
'Action,Fantasy,War' 'Action,Biography,Drama' 'Drama,Horror,Mystery'
'Adventure,Drama,Family' 'Adventure,Comedy,Romance' 'Action'
'Adventure,Crime,Mystery' 'Comedy,Family,Musical'
'Adventure,Comedy,Drama' 'Drama,Horror,Thriller' 'Drama,Musical'
'Mystery,Thriller' 'Mystery,Thriller,Western' 'Comedy,Family'
'Biography,Comedy,Drama' 'Drama,Western' 'Drama,Mystery,Romance'
'Action,Drama,Mystery' 'Action,Adventure,Crime'
'Adventure,Sci-Fi,Thriller' 'Action,Comedy,Mystery' 'Thriller,War'
'Action,Adventure,Thriller' 'Drama,Fantasy,Romance'
'Action,Drama,History' 'Animation,Adventure,Family' 'Adventure,Horror'
'Drama,Romance,Sci-Fi' 'Action,Adventure,Family' 'Action,Comedy'
'Comedy,Romance' 'Horror,Mystery' 'Drama,Family,Fantasy' 'Sci-Fi'
'Drama,War' 'Drama,Fantasy,Horror' 'Crime,Drama,History'
'Horror,Sci-Fi,Thriller' 'Action,Drama,Sport' 'Adventure,Biography,Drama']

4) Find the average rating of all movies.

```
[15] # 4. Find the average rating of all movies.
average_rating = df['Rating'].mean()
print(average_rating)

6.723199999999999
```

5) Find the movie with the highest revenue.

```
[18] # 5. Find the movie with the highest revenue.
highest_revenue_movie = df.loc[df['Revenue (Millions)'].idxmax()]
print(highest_revenue_movie)

Rank                    51
Title  Star Wars: Episode VII - The Force Awakens
Genre    Action,Adventure,Fantasy
Description  Three decades after the defeat of the Galactic...
Director    J.J. Abrams
Actors    Daisy Ridley, John Boyega, Oscar Isaac, Dommona...
Year        2015
Runtime (Minutes)      136
Rating                8.1
Votes             661608
Revenue (Millions)    936.63
Metascore             81.0
Name: 50, dtype: object
```

6) Find the movie with the lowest revenue.

```
[22] # 6. Find the movie with the lowest revenue.
lowest_revenue_movie = df.loc[df['Revenue (Millions)'].idxmin()]
print(lowest_revenue_movie)

Rank                    232
Title  A Kind of Murder
Genre    Crime,Drama,Thriller
Description  In 1960s New York, Walter Stackhouse is a succ...
Director    Andy Goddard
Actors    Patrick Wilson, Jessica Biel, Haley Bennett, V...
Year        2016
Runtime (Minutes)       95
Rating                5.2
Votes              3305
Revenue (Millions)      0.0
Metascore             50.0
Name: 231, dtype: object
```

7) Calculate the total revenue generated by all movies.

```
[23] # 7. Calculate the total revenue generated by all movies.
total_revenue = df['Revenue (Millions)'].sum()
print("\n7. Total revenue generated by all movies (in millions):", total_revenue)

7. Total revenue generated by all movies (in millions): 72337.95999999999
```

8) Find how many movies have a rating above 8.

```
[24] # 8. How many movies have a rating above 8?
movies_above_8 = df[df['Rating'] > 8].shape[0]
print("\n8. Number of movies with rating above 8:", movies_above_8)

8. Number of movies with rating above 8: 59
```

9) List the top 10 movies based on revenue.

```
[25] # 9. List the top 10 movies based on Revenue.
top_10_revenue = df.sort_values(by='Revenue (Millions)', ascending=False).head(10)
print("\n9. Top 10 movies based on revenue:\n", top_10_revenue[['Title', 'Revenue (Millions)']])

9. Top 10 movies based on revenue:
Rank  Title  Revenue (Millions)
50  Star Wars: Episode VII - The Force Awakens  936.63
87  Avatar  760.51
85  Jurassic World  652.18
76  The Avengers  623.28
54  The Dark Knight  533.32
12  Rogue One  532.17
119  Finding Dory  486.29
94  Avengers: Age of Ultron  458.99
124  The Dark Knight Rises  448.13
578  The Hunger Games: Catching Fire  424.65
```

10) Find how many movies were released each year.

```
[26] # 10. Find how many movies were released each year.
movies_per_year = df['Year'].value_counts()
print("\n10. Number of movies released each year:\n", movies_per_year)
```

10. Number of movies released each year:

| Year | count |
|------|-------|
| 2016 | 297 |
| 2015 | 127 |
| 2014 | 98 |
| 2013 | 91 |
| 2012 | 64 |
| 2011 | 63 |
| 2010 | 60 |
| 2007 | 53 |
| 2008 | 52 |
| 2009 | 51 |
| 2006 | 44 |

Name: count, dtype: int64

11) Find the average revenue per year.

```
[27] # 11. Find the average revenue per year.
avg_revenue_per_year = df.groupby('Year')['Revenue (Millions)'].mean()
print("\n11. Average revenue per year:\n", avg_revenue_per_year)
```

11. Average revenue per year:

| Year | Revenue (Millions) |
|------|--------------------|
| 2006 | 86.296667 |
| 2007 | 87.882245 |
| 2008 | 99.882745 |
| 2009 | 112.601277 |
| 2010 | 105.981579 |
| 2011 | 87.612258 |
| 2012 | 107.973281 |
| 2013 | 87.121818 |
| 2014 | 85.078723 |
| 2015 | 78.355044 |
| 2016 | 54.690976 |

Name: Revenue (Millions), dtype: float64

12) Find the average runtime of movies.

```
[28] # 12. Find the average runtime of movies.
avg_runtime = df['Runtime (Minutes)'].mean()
print("\n12. Average runtime of movies (in minutes):", avg_runtime)
```

12. Average runtime of movies (in minutes): 113.172

13) Find the director with the most movies.

```
[29] # 13. Find the director with the most movies.
most_movies_director = df['Director'].value_counts().idxmax()
print("\n13. Director with most movies:", most_movies_director)
```

13. Director with most movies: Ridley Scott

14) Find how many movies have the genre 'Action'.

```
[30] # 14. How many movies have the genre 'Action'?
action_movies_count = df['Genre'].str.contains('Action').sum()
print("\n14. Number of Action movies:", action_movies_count)
```

14. Number of Action movies: 303

15) Find the correlation between Rating and Revenue.

```
[31] # 15. Find the correlation between Rating and Revenue.
correlation_rating_revenue = df['Rating'].corr(df['Revenue (Millions)'])
print("\n15. Correlation between Rating and Revenue:", correlation_rating_revenue)
```

15. Correlation between Rating and Revenue: 0.21765389419105993

16) Find the percentage of movies with a rating greater than 7.

```
[32] # 16. Find the percentage of movies with a rating greater than 7.
percent_rating_above_7 = (df[df['Rating'] > 7].shape[0] / df.shape[0]) * 100
print("\n16. Percentage of movies with rating greater than 7:", percent_rating_above_7)
```

16. Percentage of movies with rating greater than 7: 39.900000000000006

17) Find the movie(s) with the longest runtime.

```
[33] # 17. Find the movie(s) with the longest runtime.
longest_runtime_movie = df.loc[df['Runtime (Minutes)'].idxmax()]
print("\n17. Movie with the longest runtime:\n", longest_runtime_movie)
```

17. Movie with the longest runtime:

| | |
|--------------------|--|
| Rank | 829 |
| Title | Grindhouse |
| Genre | Action,Horror,Thriller |
| Description | Quentin Tarantino and Robert Rodriguez's homage... |
| Director | Robert Rodriguez |
| Actors | Kurt Russell, Rose McGowan, Danny Trejo, Zoë Bell |
| Year | 2007 |
| Runtime (Minutes) | 191 |
| Rating | 7.6 |
| Votes | 168250 |
| Revenue (Millions) | 25.03 |
| Metascore | NaN |

Name: 828, dtype: object

18) Replace all NaN values in the Revenue column with the column mean.

```
[34] # 18. Replace all NaN values in Revenue with the column mean.
df['Revenue (Millions)'].fillna(df['Revenue (Millions)'].mean(), inplace=True)
print("\n18. Replaced all NaN values in Revenue with column mean.")
```

18. Replaced all NaN values in Revenue with column mean.

<ipython-input-34-94cd8776f94>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy. For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or 'df[col] = df[col].method(value)' instead, to perform the operation inplace on the original object.

```
df['Revenue (Millions)'].fillna(df['Revenue (Millions)'].mean(), inplace=True)
```

19) Group movies by Director and find the average rating of their movies.

```
[35] # 19. Group movies by Director and find the average rating of their movies.
avg_rating_per_director = df.groupby('Director')['Rating'].mean()
print("\n19. Average rating per director:\n", avg_rating_per_director)
```

19. Average rating per director:

| | |
|--------------------|------|
| Director | |
| Aamir Khan | 8.50 |
| Abdelatif Kechiche | 7.80 |
| Adam Leon | 6.50 |
| Adam McKay | 7.00 |
| Adam Shankman | 6.30 |
| ... | |
| Xavier Dolan | 7.55 |
| Yimou Zhang | 6.10 |
| Yorgos Lanthimos | 7.20 |
| Zack Snyder | 7.04 |
| Zackary Adler | 5.10 |

Name: Rating, Length: 644, dtype: float64

20) Find the number of missing values in each column.

```
[36] # 20. Find the number of missing values in each column.
missing_values = df.isnull().sum()
print("\n20. Number of missing values in each column:\n", missing_values)
```

20. Number of missing values in each column:

| | |
|--------------------|----|
| Rank | 0 |
| Title | 0 |
| Genre | 0 |
| Description | 0 |
| Director | 0 |
| Actors | 0 |
| Year | 0 |
| Runtime (Minutes) | 0 |
| Rating | 0 |
| Votes | 0 |
| Revenue (Millions) | 0 |
| Metascore | 64 |

dtype: int64

