EDS THEORY ACTIVITY 1

Name: Rishikesh Suryawanshi

Div: CS8

Roll no: 31

PRN: 202401120073


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

To open the dataset

```
df = pd.read_csv('/content/drive/MyDrive/Dataset/imdb_top_1000.csv')
```

Displaying the data

```
df.head()
```

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre
0	https://m.mediaamazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	Drama
1	https://m.mediaamazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min	Crime, Drama
2	https://m.mediaamazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min	Action, Crime, Drama
3	https://m.mediaamazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	A	202 min	Crime, Drama

4	https://m.mediaamazon.com/images/M/MV5BMWU4N2...	12 Angry Men	1957	U	96 min	Crime, Drama

1. Find the total number of movies in the dataset.

```
total_movies = len(df)
print(total_movies)
```

```
1000
```

2. Find the average IMDB rating of all movies.

```
average_rating = df['IMDB_Rating'].mean()
print(average_rating)
```

```
7.949299999999999
```

3. Find the movie with the highest IMDB rating.

```
highest_rated_movie = df.loc[df['IMDB_Rating'].idxmax()]
print(highest_rated_movie)
```

```
Poster_Link      https://m.media-amazon.com/images/M/MV5BMDFkYT...
Series_Title                              The Shawshank Redemption
Released_Year                                                1994
Certificate                                                     A
Runtime                                                 142 min
Genre                                                     Drama
IMDB_Rating                                                 9.3
Overview         Two imprisoned men bond over a number of years...
Meta_score                                                 80.0
Director                                          Frank Darabont
Star1                                                Tim Robbins
Star2                                             Morgan Freeman
Star3                                                 Bob Gunton
Star4                                             William Sadler
No_of_Votes                                              2343110
Gross                                                28,341,469
Name: 0, dtype: object
```

4. Find the number of movies released after the year 2010.

```
df['Released_Year'] = pd.to_numeric(df['Released_Year'], errors='coerce')

movies_after_2010 = df[df['Released_Year'] > 2010]
print(len(movies_after_2010))
```

```
225
```

5. Find the count of movies for each genre.

```
genre_counts = df['Genre'].value_counts()
print(genre_counts)
```

```
Genre
Drama                          85
Drama, Romance                 37
Comedy, Drama                  35
Comedy, Drama, Romance         31
Action, Crime, Drama           30
..
Action, Adventure, Family       1
Action, Crime, Mystery          1
Animation, Drama, Romance       1
Drama, War, Western             1
Adventure, Comedy, War          1
Name: count, Length: 202, dtype: int64
```

6. Find the movie with the longest runtime.

```
df['Runtime'] = df['Runtime'].str.replace(' min', '').astype(float)
longest_runtime_movie = df.loc[df['Runtime'].idxmax()]
print(longest_runtime_movie)
```

```
Poster_Link      https://m.media-amazon.com/images/M/MV5BMTc5Nj...
Series_Title                                  Gangs of Wasseypur
Released_Year                                               2012
Certificate                                                    A
Runtime                                                    321.0
Genre                                       Action, Comedy, Crime
IMDB_Rating                                                  8.2
Overview         A clash between Sultan and Shahid Khan leads t...
Meta_score                                                  89.0
Director                                           Anurag Kashyap
Star1                                              Manoj Bajpayee
Star2                                                Richa Chadha
Star3                                           Nawazuddin Siddiqui
Star4                                              Tigmanshu Dhulia
No_of_Votes                                                82365
Gross                                                        NaN
Name: 140, dtype: object
```

7. Calculate the median gross collection of the movies.

```
df['Gross'] = df['Gross'].replace('[\$,M]', '', regex=True).astype(float)
median_gross = df['Gross'].median() print(median_gross)
```

```
23530892.0
```

8. Find the percentage of movies having a runtime greater than 120 minutes.

```
long_movies = df[df['Runtime'] > 120]
percentage_long_movies = (len(long_movies) / len(df)) * 100
print(percentage_long_movies)
```

```
47.699999999999996
```

9. List the top 5 directors with the most movies in the dataset.

```
top_directors = df['Director'].value_counts().head(5)
print(top_directors)
```

```
Director
Alfred Hitchcock    14
Steven Spielberg    13
Hayao Miyazaki      11
Akira Kurosawa      10
Martin Scorsese     10
Name: count, dtype: int64
```

10. Find the number of unique actors listed in the dataset.

```
unique_actors = pd.unique(df[['Star1', 'Star2', 'Star3', 'Star4']].values.ravel())
print(len(unique_actors))
```

```
2709
```

11. Find the average gross collection for movies directed by Christopher Nolan.

```
nolan_movies = df[df['Director'] == 'Christopher Nolan']
average_nolan_gross = nolan_movies['Gross'].mean()
print(average_nolan_gross)
```

```
242181763.25
```

12. Find how many movies have an IMDB rating greater than 8.5.

```
high_rating_movies = df[df['IMDB_Rating'] > 8.5]
print(len(high_rating_movies))
```

```
33
```

13. Find the total gross collection of all movies combined.

```
total_gross = df['Gross'].sum()
print(total_gross)
```

```
56536877976.0
```

14. List the movies which have both high ratings (above 8.5) and long runtime (above 150 minutes).

```
high_rating_long_movies = df[(df['IMDB_Rating'] > 8.5) & (df['Runtime'] > 150)]
print(high_rating_long_movies[['Series_Title', 'IMDB_Rating', 'Runtime']])
```

```
                                        Series_Title  IMDB_Rating  Runtime
1                                      The Godfather          9.2    175.0
2                                    The Dark Knight          9.0    152.0
3                                The Godfather: Part II          9.0
202.0
5                                 The Lord of the Rings: The Return of
the King          8.9    201.0
6                                       Pulp Fiction          8.9    154.0
7                                  Schindler's List          8.9
195.0
```

```
10  The Lord of the Rings: The Fellowship of the Ring          8.8    178.0
12                    Il buono, il brutto, il cattivo          8.8    161.0
13                    The Lord of the Rings: The Two Towers          8.7
                      179.0
18                                        Hamilton           8.6    160.0
20                                   Soorarai Pottru            8.6
                                     153.0
21                                   Interstellar           8.6    169.0
24                                  Saving Private Ryan           8.6
                                    169.0
25                                  The Green Mile           8.6    189.0
31                               Shichinin no samurai        8.6    207.0
```

15. Find the correlation between Gross collection and IMDB Rating.

```
correlation = df['Gross'].corr(df['IMDB_Rating'])
print(correlation)
```

```
0.09592277110132356
```

16. List all movies released before 1980.

```
old_movies = df[df['Released_Year'] < 1980]
print(old_movies[['Series_Title', 'Released_Year']])
```

```
                      Series_Title  Released_Year
    1                    The Godfather         1972.0
    3                     The Godfather: Part II         1974.0
    4                     12 Angry Men         1957.0
    12   Il buono, il brutto, il cattivo         1966.0
    17   One Flew Over the Cuckoo's Nest         1975.0
    ..                              ...            ...
    995                            Breakfast at Tiffany's         1961.0
    996                            Giant         1956.0
    997                            From Here to Eternity         1953.0
    998                            Lifeboat         1944.0
    999                            The 39 Steps         1935.0

    [275 rows x 2 columns]
```

17. Find the average runtime of movies per decade.

```
df['Runtime'] = pd.to_numeric(df['Runtime'], errors='coerce')

# Calculate decade df['Decade'] =
(df['Released_Year'] // 10) * 10

# Group by decade average_runtime_per_decade =
df.groupby('Decade')['Runtime'].mean()
print(average_runtime_per_decade)
```

```
Decade
    1920.0     86.272727
    1930.0    102.125000
    1940.0    109.800000
    1950.0    118.678571
    1960.0    126.452055
```

```
1970.0    122.736842
1980.0    121.224719
1990.0    123.613333
2000.0    123.607595
2010.0    127.756198
2020.0    126.666667
Name: Runtime, dtype: float64
```

18. List the top 10 movies based on their Gross collection.

```
top_gross_movies = df.sort_values(by='Gross', ascending=False).head(10)
print(top_gross_movies[['Series_Title', 'Gross']])
```

```
                                Series_Title        Gross
477  Star Wars: Episode VII - The Force Awakens  936662225.0
59                             Avengers: Endgame  858373000.0
623                                      Avatar  760507625.0
60                       Avengers: Infinity War  678815482.0
652                                     Titanic  659325379.0
357                                 The Avengers  623279547.0
891                                Incredibles 2  608581744.0
2                                 The Dark Knight  534858444.0
582                                     Rogue One  532177324.0
63                           The Dark Knight Rises  448139099.0
```

19. Find the most common certificate among the movies.

```
common_certificate = df['Certificate'].mode()[0]
print(common_certificate)
```

```
U
```

20. Find the average number of votes received by the movies.

```
average_votes = df['No_of_Votes'].mean()
print(average_votes)
```

```
273692.911
```

Start coding or generate with AI.