EDS Theory Activity No. 1

Dataset – Movie review

Name – Jay Nimase

Class- CS6

Roll no- CS6-13

PRN- 202401100121

Lets assume Movie Review dataset that might contain the following columns:

- MovieID (int)
- Title (str)
- Genre (str)
- Year (int)
- Rating (float)
- Votes (int)
- Review (str)
- ReviewerID (int)

Problem statements-

1. Find the average rating of all movies.
   ```
   average_rating = df['Rating'].mean()
   ```

2. Count how many unique genres are present.
   ```
   unique_genres = df['Genre'].nunique()
   ```

3. List the top 5 highest-rated movies.
   ```
   top_movies = df.sort_values(by='Rating',
   ascending=False).head(5)
   ```

4. Find the number of movies released each year.
   ```
   movies_per_year = df['Year'].value_counts().sort_index()
   ```

5. Get the most reviewed movie.
   ```
   most_reviewed = df.groupby('Title')['Review'].count().idxmax()
   ```

6. Calculate the total number of votes per genre.
   ```
   votes_per_genre = df.groupby('Genre')['Votes'].sum()
   ```

7. Filter out movies with rating less than 3.0.
   ```
   low_rated = df[df['Rating'] < 3.0]
   ```

8. Check how many movies have the word "Love" in the title.
   ```
   love_movies = df[df['Title'].str.contains('Love', case=False)]
   count_love_movies = love_movies.shape[0]
   ```

9. Find the average rating for each genre.
   ```
   avg_rating_by_genre = df.groupby('Genre')['Rating'].mean()
   ```

10. Get a pivot table of average rating by year and genre.
    ```
    rating_pivot = df.pivot_table(values='Rating', index='Year', columns='Genre', aggfunc='mean')
    ```

11. Replace missing values in the "Rating" column with the column mean.
    ```
    df['Rating'] = df['Rating'].fillna(df['Rating'].mean())
    ```

12. Use NumPy to find the standard deviation of ratings.
    ```
    import numpy as np
    rating_std = np.std(df['Rating'].dropna())
    ```

13. Find which reviewer has given the highest average rating.
    ```
    top_reviewer = df.groupby('ReviewerID')['Rating'].mean().idxmax()
    ```

14. Extract year from the title if it contains it (e.g., "Titanic (1997)").

```
df['Extracted_Year'] = df['Title'].str.extract(r'\((\d{4})\)').astype('float')
```

15. Find correlation between votes and rating.

```
correlation = df[['Votes', 'Rating']].corr().loc['Votes', 'Rating']
```

16. List all movies released after 2015 with a rating above 4.0.

```
recent_high_rated = df[(df['Year'] > 2015) & (df['Rating'] > 4.0)]
```

17. Get the count of reviewers who gave more than 10 reviews.

```
active_reviewers = df['ReviewerID'].value_counts()
reviewer_count = (active_reviewers > 10).sum()
```

18. Use NumPy to normalize the rating column.

```
ratings = df['Rating'].values
normalized_ratings = (ratings - np.min(ratings)) / (np.max(ratings) - np.min(ratings))
df['Normalized_Rating'] = normalized_ratings
```

19. Calculate the average number of words in reviews.

```
df['Word_Count'] = df['Review'].fillna(").apply(lambda x: len(x.split()))
average_words = df['Word_Count'].mean()
```

20. Create a new column that categorizes ratings: "Low" (<3), "Medium" (3-4), "High" (>4).

```
df['Rating_Category'] = pd.cut(df['Rating'], bins=[0, 3, 4, 5], labels=['Low', 'Medium', 'High'])
```