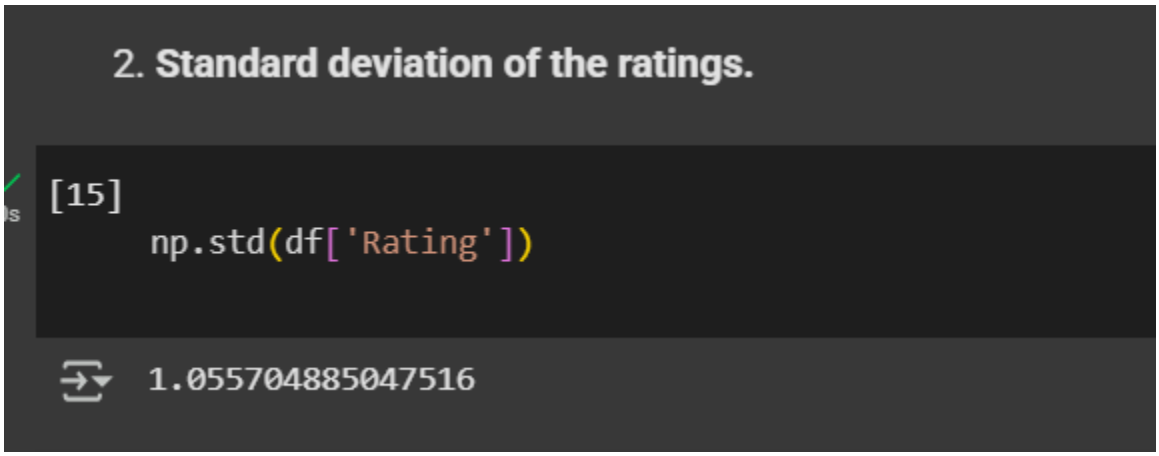
EDS THEORY ASSIGNMENT

NAME : NIKITA SHARAD PAWAR
ROLL NO : CS5-55
PRN : 202401100018
BATCH : CS53

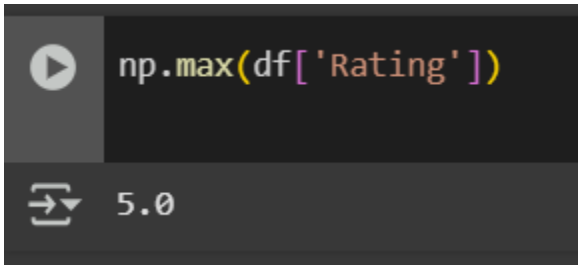
NUMPY BASED PROBLEMS

1. Find the average rating.

2. Find the standard deviation of the ratings.



3. Find the maximum rating given



4. Find the minimum rating given.

```
[8] np.min(df['Rating'])

0.5
```

5. How many ratings are above the average rating?

```
[9] np.sum(df['Rating'] > np.mean(df['Rating']))

np.int64(535671)
```

6. Calculate the variance of the ratings.

```
[10] np.var(df['Rating'])

1.1145128043131887
```

7. Find the 25th, 50th, and 75th percentiles of ratings.

```
[11] np.percentile(df['Rating'], [25, 50, 75])

array([3., 4., 4.])
```

8. Normalize the ratings (min-max scaling between 0 and 1).

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

	Rating
0	0.777778
1	0.111111
2	0.333333
3	1.000000
4	1.000000
...	...
1048570	0.777778
1048571	0.777778
1048572	1.000000
1048573	0.888889
1048574	0.666667

1048575 rows × 1 columns

dtype: float64

9. Create a boolean array where ratings are greater than 3

```
[13] np.array(df['Rating']) > 3
```

```
array([ True, False, False, ...,  True,  True,  True])
```

10. Sum of all ratings.

```
[14] np.sum(df['Rating'])

np.float64(3737353.5)
```

Pandas-Based Problems

11. Find the number of movies in the dataset.

```
PANDAS BASED PROBLEMS

[17] df['Title'].nunique()

87382
```

12. Find the movie with the highest rating.

```
[18] df.loc[df['Rating'].idxmax()]
```

	3
MovieId	4.0
Title	Waiting to Exhale (1995)
Genres	Comedy|Drama|Romance
ImdbId	114885.0
TmdbId	31357.0
Rating	5.0
Timestamp	944249077

dtype: object

13. Find all movies with a rating of 5.

```
[19] df[df['Rating'] == 5]
```

	MovieId	Title	Genres	ImdbId	TmdbId	Rating	Timestamp
3	4.0	Waiting to Exhale (1995)	Comedy|Drama|Romance	114885.0	31357.0	5.0	944249077
4	5.0	Father of the Bride Part II (1995)	Comedy	113041.0	11862.0	5.0	943228858
7	8.0	Tom and Huck (1995)	Adventure|Children	112302.0	45325.0	5.0	944248943
9	10.0	GoldenEye (1995)	Action|Adventure|Thriller	113189.0	710.0	5.0	944249008
11	12.0	Dracula: Dead and Loving It (1995)	Comedy|Horror	112896.0	12110.0	5.0	943228442
...	...	...	...	...	...	...	...
1048551	NaN	NaN	NaN	NaN	NaN	5.0	1607098167
1048561	NaN	NaN	NaN	NaN	NaN	5.0	1564778600
1048563	NaN	NaN	NaN	NaN	NaN	5.0	1607097746
1048568	NaN	NaN	NaN	NaN	NaN	5.0	1607097777
1048572	NaN	NaN	NaN	NaN	NaN	5.0	1607097533

154358 rows × 7 columns

14. Count the number of genres available (considering '|' as separator).

```
[20] genres = df['Genres'].str.split('|').explode()
     genres.nunique()
```

```
20
```

15. List top 5 most frequent genres.

```
[21] genres.value_counts().head(5)
```

Genres	count
Drama	34175
Comedy	23124
Thriller	11823
Romance	10369
Action	9668

dtype: int64

16. Find the average rating for each genre.

```
df_exploded = df.copy()
df_exploded['Genres'] = df_exploded['Genres'].str.split('|')
df_exploded = df_exploded.explode('Genres')
df_exploded.groupby('Genres')['Rating'].mean()
```

Genres	Rating
(no genres listed)	3.527119
Action	3.526893
Adventure	3.536283
Animation	3.592376
Children	3.517035
Comedy	3.520779
Crime	3.534977
Documentary	3.548329
Drama	3.532904
Fantasy	3.545573
Film-Noir	3.464589
Horror	3.557315
IMAX	3.466667
Musical	3.556185
Mystery	3.524670
Romance	3.533996
Sci-Fi	3.508355
Thriller	3.542671
War	3.535914
Western	3.499410

dtype: float64

17. How many movies were rated after 2000 (based on Timestamp)?

```
[24] from datetime import datetime

     df['Datetime'] = pd.to_datetime(df['Timestamp'], unit='s')
     (df['Datetime'].dt.year > 2000).sum()
```

```
np.int64(857938)
```

18. Create a new column categorizing ratings as 'High' (>=4) or 'Low' (<4).

```
df['Rating_Category'] = df['Rating'].apply(lambda x: 'High' if x >= 4 else 'Low')
df[['Title', 'Rating_Category']]
```

	Title	Rating_Category
0	Toy Story (1995)	High
1	Jumanji (1995)	Low
2	Grumpier Old Men (1995)	Low
3	Waiting to Exhale (1995)	High
4	Father of the Bride Part II (1995)	High
...	...	...
1048570	NaN	High
1048571	NaN	High
1048572	NaN	High
1048573	NaN	High
1048574	NaN	Low

1048575 rows × 2 columns

19. Find the average reviewer rating per release decade.

```
df['Release_Year'] = df['Title'].str.extract(r'\((\d{4})\)')
df['Decade'] = (df['Release_Year'].astype(float) // 10) * 10
df.groupby('Decade')['Rating'].mean()
```

Decade	Rating
1870.0	4.666667
1880.0	3.923077
1890.0	3.430348
1900.0	3.672241
1910.0	3.524485
1920.0	3.540411
1930.0	3.519676
1940.0	3.535326
1950.0	3.518385
1960.0	3.515740
1970.0	3.518531
1980.0	3.522788
1990.0	3.513755
2000.0	3.540739
2010.0	3.530085
2020.0	3.584272

20. Sort movies by Rating in descending order.

```
[27] df.sort_values('Rating', ascending=False)
```

	MovieId	Title	Genres	ImdbId	TmdbId	Rating	Timestamp	Datetime	Rating_Category	Release_Year	Decade
1048572	NaN	NaN	NaN	NaN	NaN	5.0	1607097533	2020-12-04 15:58:53	High	NaN	NaN
7	8.0	Tom and Huck (1995)	Adventure|Children	112302.0	45325.0	5.0	944248943	1999-12-03 19:22:23	High	1995	1990.0
1048551	NaN	NaN	NaN	NaN	NaN	5.0	1607098167	2020-12-04 16:09:27	High	NaN	NaN
11	12.0	Dracula: Dead and Loving It (1995)	Comedy|Horror	112896.0	12110.0	5.0	943228442	1999-11-21 23:54:02	High	1995	1990.0
3	4.0	Waiting to Exhale (1995)	Comedy|Drama|Romance	114885.0	31357.0	5.0	944249077	1999-12-03 19:24:37	High	1995	1990.0
...	...	...	...	...	...	...	...	...	...	...	...
82022	275761.0	Diorama (2022)	Comedy|Drama	10851466.0	967940.0	0.5	1308754650	2011-06-22 14:57:30	Low	2022	2020.0
639968	NaN	NaN	NaN	NaN	NaN	0.5	1372447940	2013-06-28 19:32:20	Low	NaN	NaN
1031673	NaN	NaN	NaN	NaN	NaN	0.5	1563909460	2019-07-23 19:17:40	Low	NaN	NaN
640011	NaN	NaN	NaN	NaN	NaN	0.5	1239503262	2009-04-12 02:27:42	Low	NaN	NaN
1031693	NaN	NaN	NaN	NaN	NaN	0.5	1563909988	2019-07-23 19:26:28	Low	NaN	NaN

1048575 rows × 11 columns