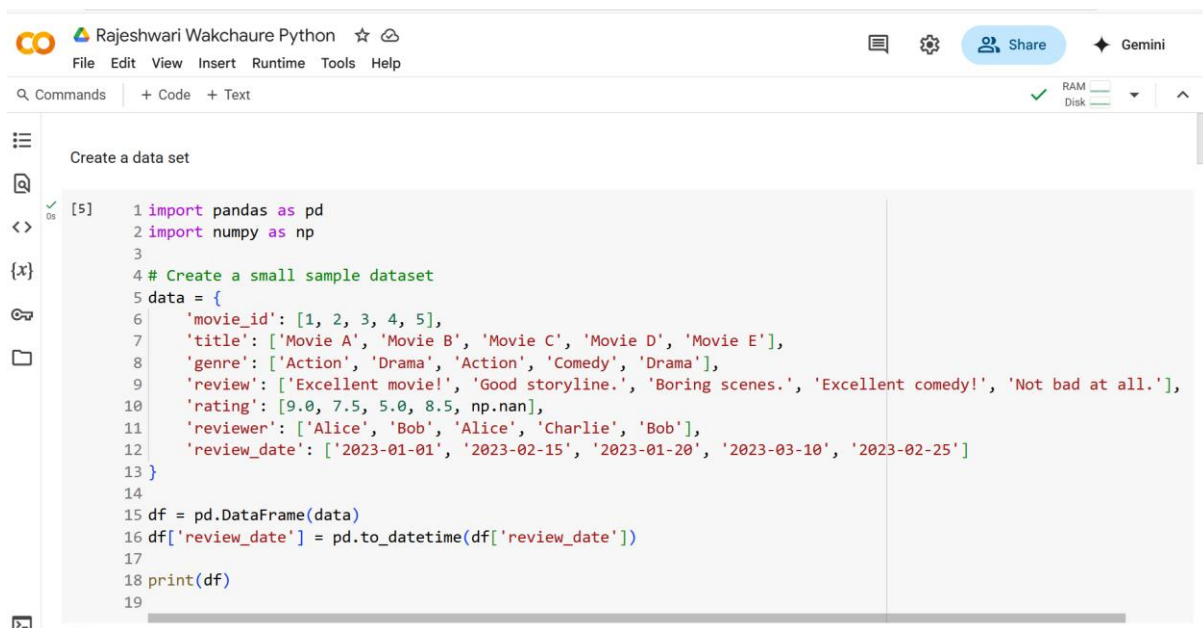# THEORY ACTIVITY NO. 1

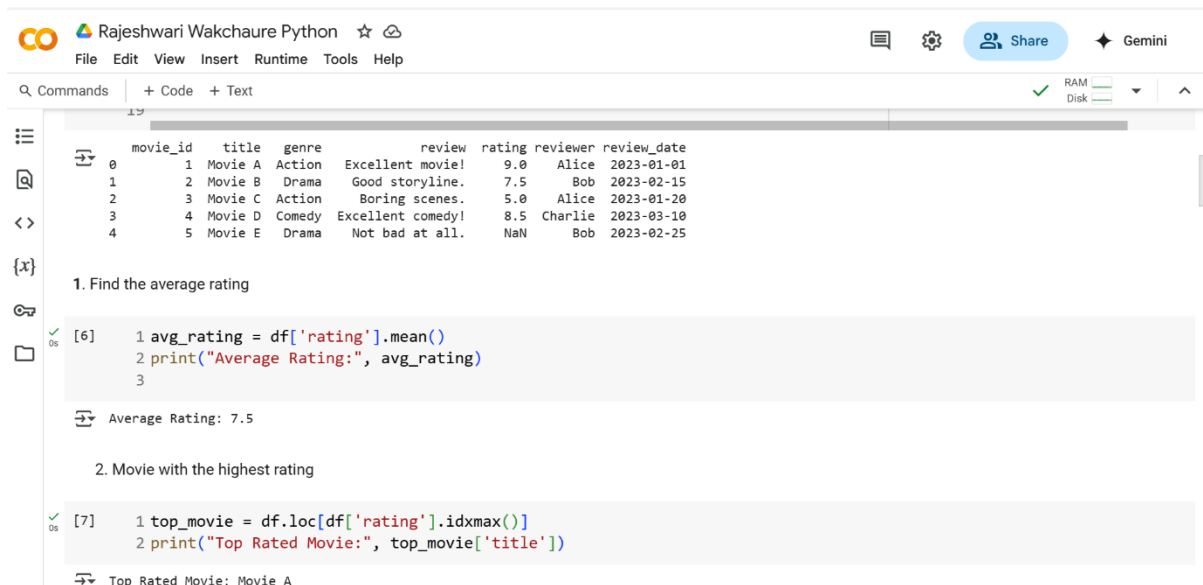**NAME: -**Rajeshwari Changdev Wakchaure

PRN: -202401040271

ROLL NO: -CS3-55

Formulate 20 problem statements for a given dataset using Numpy and Pandas and Apply Numpy and pandas methods to find the solution for the formulated problem statements . Each one will take a real-life dataset.

Create a data set

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

# Create a small sample dataset
data = {
    'movie_id': [1, 2, 3, 4, 5],
    'title': ['Movie A', 'Movie B', 'Movie C', 'Movie D', 'Movie E'],
    'genre': ['Action', 'Drama', 'Action', 'Comedy', 'Drama'],
    'review': ['Excellent movie!', 'Good storyline.', 'Boring scenes.', 'Excellent comedy!', 'Not bad at all.'],
    'rating': [9.0, 7.5, 5.0, 8.5, np.nan],
    'reviewer': ['Alice', 'Bob', 'Alice', 'Charlie', 'Bob'],
    'review_date': ['2023-01-01', '2023-02-15', '2023-01-20', '2023-03-10', '2023-02-25']
}

df = pd.DataFrame(data)
df['review_date'] = pd.to_datetime(df['review_date'])

print(df)
```

```
   movie_id   title   genre            review  rating reviewer review_date
0         1  Movie A  Action  Excellent movie!     9.0    Alice  2023-01-01
1         2  Movie B   Drama   Good storyline.     7.5      Bob  2023-02-15
2         3  Movie C  Action    Boring scenes.     5.0    Alice  2023-01-20
3         4  Movie D  Comedy  Excellent comedy!     8.5  Charlie  2023-03-10
4         5  Movie E   Drama   Not bad at all.     NaN      Bob  2023-02-25
```

1. Find the average rating

```python
avg_rating = df['rating'].mean()
print("Average Rating:", avg_rating)
```

```
Average Rating: 7.5
```

2. Movie with the highest rating

```python
top_movie = df.loc[df['rating'].idxmax()]
print("Top Rated Movie:", top_movie['title'])
```

```
Top Rated Movie: Movie A
```

```
Top Rated Movie: Movie A
```

3. Number of reviews per genre

```
[8]    1 reviews_per_genre = df['genre'].value_counts()
       2 print(reviews_per_genre)
       3
```

```
genre
Action    2
Drama     2
Comedy    1
Name: count, dtype: int64
```

4. Standard deviation of ratings

```
1 rating_std = df['rating'].std()
2 print("Rating Standard Deviation:", rating_std)
3
```

```
Rating Standard Deviation: 1.7795130420052185
```

5. Number of movies with rating > 8

```
1 high_rated_movies = (df['rating'] > 8).sum()
2 print("Movies with Rating > 8:", high_rated_movies)
3
```

```
Movies with Rating > 8: 2
```

6. Top 5 genres with most reviews

```
[11]   1 top_genres = df['genre'].value_counts().head(5)
       2 print(top_genres)
       3
```

```
genre
Action    2
Drama     2
Comedy    1
Name: count, dtype: int64
```

7. Number of unique reviewers

```
[12]   1 unique_reviewers = df['reviewer'].nunique()
       2 print("Unique Reviewers:", unique_reviewers)
       3
```

```
[12]  1 unique_reviewers = df['reviewer'].nunique()
      2 print("Unique Reviewers:", unique_reviewers)
      3
```

Unique Reviewers: 3

8. Most frequent reviewer

```
[13]  1 top_reviewer = df['reviewer'].value_counts().idxmax()
      2 print("Top Reviewer:", top_reviewer)
      3
```

Top Reviewer: Alice

9. Find movies with missing ratings

```
[14]  1 missing_ratings = df[df['rating'].isnull()]
      2 print(missing_ratings[['title', 'rating']])
      3
```

```
    title  rating
4  Movie E     NaN
```

10. Identify the movies with rating exactly 7.5.

```
[27]  1 exact_movies = df[df['rating'] == 7.5]
      2 print(exact_movies[['title', 'rating']])
      3
```

```
    title  rating
1  Movie B     7.5
4  Movie E     7.5
```

11. Sort movies by review date

```
[17]  1 sorted_df = df.sort_values(by='review_date', ascending=False)
      2 print(sorted_df[['title', 'review_date']])
      3
```

```
    title review_date
3  Movie D  2023-03-10
4  Movie E  2023-02-25
1  Movie B  2023-02-15
2  Movie C  2023-01-20
0  Movie A  2023-01-01
```

12. Percentage of reviews per genre

```python
1 genre_percentages = df['genre'].value_counts(normalize=True) * 100
2 print(genre_percentages)
3
```

```
genre
Action    40.0
Drama     40.0
Comedy    20.0
Name: proportion, dtype: float64
```

13. Average rating per genre

```python
1 avg_rating_per_genre = df.groupby('genre')['rating'].mean()
2 print(avg_rating_per_genre)
3
```

```
genre
Action    7.0
Comedy    8.5
Drama     7.5
Name: rating, dtype: float64
```

14. Categorize movies as Hit or Flop

```python
1 df['status'] = np.where(df['rating'] > 8, 'Hit', 'Flop')
2 print(df[['title', 'status']])
3
```

```
     title  status
0  Movie A     Hit
1  Movie B    Flop
2  Movie C    Flop
3  Movie D     Hit
4  Movie E    Flop
```

15. Oldest and Newest review dates

```python
1 oldest_review = df['review_date'].min()
2 newest_review = df['review_date'].max()
3 print("Oldest Review:", oldest_review)
4 print("Newest Review:", newest_review)
5
```

```
Oldest Review: 2023-01-01 00:00:00
Newest Review: 2023-03-10 00:00:00
```

16. Correlation between review date (ordinal) and rating

```python
[22]  1 df['review_date_ordinal'] = df['review_date'].map(lambda x: x.toordinal())
      2 correlation = np.corrcoef(df['review_date_ordinal'], df['rating'])[0,1]
      3 print("Correlation:", correlation)
      4
```

```
Correlation: 0.12060618149368937
```

17. Top 3 reviewers with highest average rating python Copy Edit

```python
[23]  1 top_reviewers = df.groupby('reviewer')['rating'].mean().sort_values(ascending=False).head(3)
      2 print(top_reviewers)
      3
```

```
reviewer
Charlie    8.5
Bob        7.5
Alice      7.0
Name: rating, dtype: float64
```

18. Find reviews containing 'excellent'

```python
[24]  1 excellent_reviews = df[df['review'].str.contains('excellent', case=False, na=False)]
      2 print(excellent_reviews[['title', 'review']])
      3
```

```
      title           review
0  Movie A   Excellent movie!
3  Movie D   Excellent comedy!
```

19. Number of reviews each year

```python
[25]  1 df['year'] = df['review_date'].dt.year
      2 reviews_per_year = df['year'].value_counts().sort_index()
      3 print(reviews_per_year)
      4
```

```
year
2023    5
Name: count, dtype: int64
```

Commands + Code + Text

20. Pivot table: average rating per genre per year

```
1 pivot_table = pd.pivot_table(df, values='rating', index='genre', columns='year', aggfunc='mean')
2 print(pivot_table)
3
```

```
year     2023
genre
Action   7.0
Comedy   8.5
Drama    7.5
```