NAME : ABHIJEET SATISH BEDWAL
PRN : 202401090061
DIV.: ME.   BATCH : ME4
ROLL NO.: ME76
SUB : EDS THEORY ACTIVITY NO. : 01

Problem Statements and Solutions for MovieLens Latest Dataset using NumPy and Pandas

Dataset Overview

The MovieLens dataset contains movie ratings, user information, and movie metadata. We will use the following files:
- movies.csv: Contains movieId, title, and genres.
- ratings.csv: Contains userId, movieId, rating, and timestamp.
- tags.csv: Contains userId, movieId, tag, and timestamp.
- links.csv: Contains movieId, imdbId, and tmdbId.

Problem Statements and Solutions

1. Load and Inspect the Dataset
Problem: Load the movies.csv, ratings.csv, tags.csv, and links.csv files into Pandas DataFrames and display the first 5 rows of each.

Solution:

```
import pandas as pd

movies = pd.read_csv('movies.csv')
ratings = pd.read_csv('ratings.csv')
tags = pd.read_csv('tags.csv')
links = pd.read_csv('links.csv')

print(movies.head())
print(ratings.head())
print(tags.head())
print(links.head())
```

---

2. Check for Missing Values
Problem:Determine if there are any missing values in the `movies`, `ratings`, `tags`, and `links` datasets.

Solution:

```
print(movies.isnull().sum())
print(ratings.isnull().sum())
print(tags.isnull().sum())
print(links.isnull().sum())
```

---

3. Find the Number of Unique Users
Problem:Calculate the number of unique users who have rated movies.

Solution:
```
unique_users = ratings['userId'].nunique()
print(f"Number of unique users: {unique_users}")
```

---

4. Find the Number of Unique Movies
Problem: Calculate the number of unique movies in the dataset.

Solution:
```
unique_movies = movies['movieId'].nunique()
print(f"Number of unique movies: {unique_movies}")
```

---

5. Find the Average Movie Rating
Problem:Compute the average rating across all movies.

Solution:

```
avg_rating = ratings['rating'].mean()
print(f"Average movie rating: {avg_rating:.2f}")
```

---

6. Find the Most Rated Movie
Problem: Identify the movie with the highest number of ratings.

Solution:

```
most_rated = ratings['movieId'].value_counts().idxmax()
movie_title = movies[movies['movieId'] == most_rated]['title'].values[0]
print(f"Most rated movie: {movie_title}")
```

---

7. Find the Highest-Rated Movie (Avg Rating ≥ 4.5)
Problem: Find movies with an average rating of at least 4.5 (with a minimum of 50 ratings).

Solution:

```
movie_ratings = ratings.groupby('movieId')['rating'].agg(['mean', 'count'])
high_rated = movie_ratings[(movie_ratings['mean'] >= 4.5) & (movie_ratings['count'] >= 50)]
high_rated_movies = pd.merge(high_rated, movies, on='movieId')
print(high_rated_movies[['title', 'mean']].sort_values('mean', ascending=False))
```

---

8. Find the Most Popular Genres
Problem: Determine the most common movie genres.

Solution:

```
genres = movies['genres'].str.split('|', expand=True).stack().value_counts()
print("Most common genres:")
print(genres.head(10))
```

---

9. Find the Year with the Most Movie Releases
Problem: Extract the year from movie titles and find the year with the most releases.

Solution:

```
movies['year'] = movies['title'].str.extract(r'\((\d{4})\)')
year_counts = movies['year'].value_counts()
```

```
print(f"Year with most releases: {year_counts.idxmax()} ({year_counts.max()} movies)")
```

---

10. Find the User Who Rated the Most Movies
Problem:Identify the user who has rated the highest number of movies.

Solution:

```
active_user = ratings['userId'].value_counts().idxmax()
print(f"Most active user (ID): {active_user}")
```

---

11. Find the Distribution of Ratings
Problem:Plot the distribution of movie ratings (histogram).

Solution:

```
import matplotlib.pyplot as plt
ratings['rating'].hist(bins=10)
plt.title("Distribution of Movie Ratings")
plt.xlabel("Rating")
plt.ylabel("Frequency")
plt.show()
```

---

12. Find the Correlation Between Rating Count and Average Rating
Problem: Check if movies with more ratings tend to have higher average ratings.

Solution:

```
movie_stats = ratings.groupby('movieId')['rating'].agg(['mean', 'count'])
correlation = movie_stats['mean'].corr(movie_stats['count'])
print(f"Correlation between rating count and average rating: {correlation:.2f}")
```

---

13. Find the Most Common Tags
Problem: Identify the most frequently used tags.

Solution:

```
common_tags = tags['tag'].value_counts().head(10)
print("Most common tags:")
print(common_tags)
```

---

14. Find the Longest Movie Title
Problem: Determine the movie with the longest title.

Solution:

```
movies['title_length'] = movies['title'].str.len()
longest_title = movies.loc[movies['title_length'].idxmax(), 'title']
print(f"Longest movie title: {longest_title}")
```

---

15. Find the Average Rating per Genre
Problem: Compute the average rating for each genre.

Solution:

```
genre_ratings = pd.merge(movies, ratings, on='movieId')
genre_ratings = genre_ratings.explode('genres')
avg_genre_rating = genre_ratings.groupby('genres')['rating'].mean().sort_values(ascending=False)
print(avg_genre_rating)
```

---

16. Find the Most Prolific Taggers
Problem: Identify users who have applied the most tags.

Solution:

```
prolific_taggers = tags['userId'].value_counts().head(5)
print("Most prolific taggers:")
print(prolific_taggers)
```

---

17. Find the Average Number of Ratings per User
Problem: Calculate the average number of ratings given by each user.

Solution:

```
avg_ratings_per_user = ratings.groupby('userId')['movieId'].count().mean()
print(f"Average ratings per user: {avg_ratings_per_user:.2f}")
```

---

18. Find the Most Rated Movie in Each Genre
Problem: For each genre, find the movie with the highest number of ratings.

Solution:

```
genre_movies = pd.merge(movies, ratings, on='movieId')
genre_movies = genre_movies.explode('genres')
most_rated_per_genre = genre_movies.groupby('genres').apply(lambda x:
x.loc[x['movieId'].value_counts().idxmax()])
print(most_rated_per_genre[['genres', 'title']])
```

---

19. Find the Number of Movies per Year
Problem: Count the number of movies released each year.

Solution:

```
movies_per_year = movies['year'].value_counts().sort_index()
print("Movies per year:")
print(movies_per_year)
```

---

20. Find the Most Rated Movie in the Last Decade
Problem: Identify the most rated movie released in the last 10 years.

Solution:

```
recent_movies = movies[movies['year'].astype(float) >= (2023 - 10)]
recent_ratings = pd.merge(recent_movies, ratings, on='movieId')
most_rated_recent = recent_ratings['movieId'].value_counts().idxmax()
movie_title = movies[movies['movieId'] == most_rated_recent]['title'].values[0]
print(f"Most rated recent movie: {movie_title}")
```

Tab 2