The reason the recommendations in the function differ from our original notebook code is due to a few key differences in logic and a common indexing mismatch.

Here are the three main reasons for the discrepancy:

1. Recommendation Pool (Search Scope)

- **In our notebook:** We limited the recommendations to *only* movies that had been watched by the **top 5 most similar users**. This is a very small pool of movies.
- **In the function:** The function looks at **every movie in the entire database** that the user hasn't seen yet. It then predicts what the user would rate those movies.

2. Selection Criteria (Threshold vs. Ranking)

- **In our notebook:** We selected all movies from our small pool that had a predicted rating **>=1** (if value >= 1).
- **In the function:** It sorts the predictions and picks the **Top 5 (or N) absolute highest** predicted ratings. It doesn't use a fixed cutoff like 1.

3. Handling of Movie IDs

In our notebook, datama.columns contains the actual Movie IDs (e.g., 1, 2, 3...).

- When we do input_user_pred[re], We are accessing the column named re.
- In my function, I ensure that the mapping between the matrix index and the Movie ID is preserved, but since our search pool was restricted to the "similar users' list," the results will naturally be different.

The function looks at **every movie in the entire database** that the user hasn't seen yet instead of top five most similar users.

It sorts the predictions and picks the **Top 5 (or N) absolute highest** predicted ratings. It doesn't use a fixed cutoff like 1.

Ensured that the mapping between the matrix index and the Movie ID is preserved