**1. Collaborative Filtering: User-Based vs. Item-Based (4 Marks)**

a) Compare and contrast user-based collaborative filtering with item-based collaborative filtering.

User-based collaborative filtering finds similarities between users and recommends items that users with similar preferences have enjoyed. For example, if User A and User B have similar viewing histories, User A will get recommendations based on what User B has watched.

Item-based collaborative filtering instead looks at the similarity between items. It recommends items that are often co-rated or consumed together. For example, if many users who watched Movie X also liked Movie Y, then Movie Y is recommended when someone watches Movie X.

b) Discuss which approaches are more efficient in a large-scale system like Netflix.

For large-scale platforms like Netflix, item-based collaborative filtering is more efficient. This is because user-based methods require recalculating similarities every time new users join or existing users’ preferences change, which is expensive when larger scale. In contrast, item-based filtering works with a relatively stable set of movies and shows, allowing calculation in advance of item similarities, making it faster and more scalable for millions of users.

**2. Sparsity and Matrix Factorisation (2 Marks)**

a) Discuss matrix sparsity in collaborative filtering.

In collaborative filtering, the user–item rating matrix is usually very sparse because most users rate only a small fraction of all available items. For example, on Netflix, each user may watch and rate only dozens of movies out of thousands, leaving the majority of the matrix empty. This sparsity makes it difficult to find reliable similarities and generate accurate recommendations.

b) Explain how matrix factorisation helps address the sparsity issue.

Matrix factorisation reduces the large sparse rating matrix into two lower-dimensional matrices representing users and items in terms of latent features (e.g., genres, themes, or anything else). Techniques like Singular Value Decomposition (SVD) or Alternating Least Squares (ALS) fill in the missing values by learning these latent features. This way, even if a user has rated only a few movies, the system can still predict preferences by matching their latent factors with those movies, mitigating the sparsity problem.