## Exercise 1: Building a Movie Recommender System using Unsupervised Learning

The purpose of this exercise is to develop a simple movie recommender system using unsupervised learning techniques. You will be working with the MovieLens dataset, which is one of the most common datasets used when implementing and testing recommender systems.

**Dataset:**
MovieLens | GroupLens
It is advisable to use : the one in the **recommended for education and development section**. However, feel free to explore any of the variations you have been presented with.

**Steps:**

1. Data Preprocessing:
   - Load the MovieLens data into a pandas DataFrame.
   - Handle any missing data: check for missing values and fill them appropriately.
   - Conduct exploratory data analysis: understand the dataset, look for patterns, outliers, etc.

2. Data Transformation:
   - Transform the data to prepare it for the unsupervised learning model. This can involve creating a matrix where each row corresponds to a user, each column corresponds to a different movie, and each entry corresponds to a user's rating for a specific movie.

3. Model Creation:
   - Use an unsupervised learning technique such as K-means clustering or hierarchical clustering to segment users into groups. The idea is that users in the same cluster have similar movie preferences.
   - Experiment with different numbers of clusters. Determine the optimal number of clusters by using techniques such as the Elbow method.

4. Recommender System:
   - Create a function that takes a user ID and the number of recommendations as inputs and outputs movie recommendations for that user.
   - This function can work by finding the cluster that a user belongs to, and then recommending the highest-rated movies from that cluster that the user hasn't already rated.

5. Evaluation:
   - Although this is an unsupervised task, we can still evaluate the performance of our recommender system. One common way to do this is by splitting the data into a training set and a test set. We can pretend we don't know some of the ratings in the test set, predict them using our recommender system, and then compare our predictions to the actual values.
   - Calculate metrics such as RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) to evaluate the performance of your recommender system.

6. Reflection:
   - Write a short report summarizing your findings, discussing any challenges you faced, and suggesting potential improvements or future directions for this work.

## Exercise 2: Creating a Music Recommender System using Unsupervised Learning

For this exercise, you will develop a music recommendation system using unsupervised learning techniques. You'll be working with the Last.fm dataset, which includes data about music preferences from Last.fm.

**Dataset:**
:: Music Recommendation Datasets ::

**Steps:**

1. Data Preprocessing:
   - Load the data into a Pandas DataFrame.
   - Handle any missing data and conduct some exploratory data analysis.

2. Data Transformation:
   - Transform the data into a matrix format where rows correspond to users, columns correspond to songs or artists, and entries correspond to the number of times a user has listened to a song or artist.

3. Model Creation:
   - Implement an unsupervised learning technique, such as K-means clustering, to group users based on their music preferences.
   - Experiment with different numbers of clusters, and use a method like the Elbow method to determine the optimal number of clusters.

4. Recommender System:
   - Create a function that takes a user ID and the number of recommendations as input, and outputs music recommendations for that user.
   - This function should find the cluster that the user belongs to, and recommend songs or artists from that cluster that the user hasn't listened to yet.

5. Evaluation:
   - Split your data into training and test sets and use this to evaluate the performance of your recommender system. Compare your system's recommendations to the actual music the users in the test set listened to. **See exercise 1, no. 5 for more details**

6. Reflection:
   - Write a brief report about your findings. Discuss any challenges you faced, as well as potential improvements and future directions for this project.

## Exercise 3: Building a Book Recommender System with Unsupervised Learning

In this exercise, you will be working with the Book-Crossing dataset, which includes data on book ratings.

**Dataset:**
Book-Crossing Dataset

Use the CSV dump.

**Steps:**

1. Data Preprocessing:
   - Load the data into a Pandas DataFrame.
   - Handle any missing data and conduct exploratory data analysis to understand the dataset.

2. Data Transformation:
   - Transform the data into a matrix where each row corresponds to a user, each column corresponds to a different book, and each entry corresponds to a user's rating for a particular book.

3. Model Creation:
   - Apply an unsupervised learning technique like hierarchical clustering or K-means clustering to group users into clusters based on their book preferences.
   - Experiment with different numbers of clusters, and find the optimal number using the Elbow method or another technique.

4. Recommender System:
   - Develop a function that takes a user ID and the number of recommendations as input and provides book recommendations for that user.
   - The function should find the cluster the user belongs to and recommend highly-rated books from that cluster that the user hasn't already rated.

5. Evaluation:
   - Split the data into a training set and a test set to evaluate your recommender system. Predict ratings for the test set based on the recommendations from your system and compare them to the actual ratings. **See exercise 1, no. 5 for more details**

6. Reflection:
   - Write a short report discussing your findings, any difficulties you encountered, potential improvements, and future directions for this project.

**Please Note:** These exercises provide a basic introduction to building a recommender system using unsupervised learning. However, it's worth noting that many advanced recommender systems often use a combination of different techniques, including content-based filtering, collaborative filtering, and deep learning.