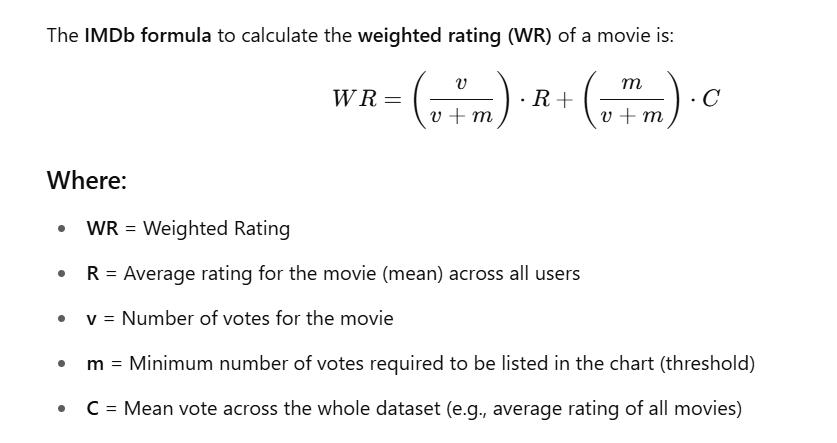
**Experiment 2:**

**Instructions for Students**

1. **Import Necessary Libraries**
   * Import pandas for data handling.
   * Import TfidfVectorizer from sklearn.feature\_extraction.text.
   * Import cosine\_similarity from sklearn.metrics.pairwise.
2. **Load the Dataset**
   * Load movies\_metadata.csv into a DataFrame using pd.read\_csv().
   * Display the first few rows using head() to understand the structure.
3. **Preprocessing & Filtering**
   * Compute the **mean vote average** from the dataset and store it as C.
   * Calculate the **90th percentile of vote\_count** and store it as m.
   * Filter the movies that have a vote count >= m and store them in a new DataFrame q\_movies.
   * Write a function that computes the weighted rating of each movie using IMDB Formula:
   * 
   * Add a new column called score to the q\_movies DataFrame, where each movie's score is calculated using the weighted\_rating() function.
   * Sort movies based on score calculated above, and Print the top 20 movies.
4. **Working with TF-IDF**

Content-Based Recommender

* Print overviews of the first 5 movies.
* Import TfIdfVectorizer from scikit-learn
* Use TfidfVectorizer to convert the movie descriptions (e.g., from overview column) into TF-IDF vectors. Remove all english stop words such as 'the', 'a'.
* Handle NaN values in text data (e.g., replace them with empty strings).
* Construct the required TF-IDF matrix by fitting and transforming the data
* Output the shape of tfidf\_matrix.
* Display an array mapping from feature integer indices to feature name. (Indices from 5000 till 5009).

1. **Compute Similarities**
   * Import linear kernel.
   * Calculate cosine similarity between the TF-IDF vectors of all movies.
   * Create a **reverse lookup map** from **movie titles to their index positions** in the metadata DataFrame. Now you can retrieve the index of a movie by its title. This index can then be used to fetch similarity scores, metadata, etc.
2. **Build the Recommender Function**
   * Create a function that:
     + Accepts a movie title.
     + Finds its index.
     + Fetches similarity scores. Creates pairs of (index, similarity\_score) and converts it into a list of tuples.
     + Sorts and returns the top 10 most similar movies.
3. **Test the Recommender**
   * Test your recommender function with a movie name like "The Dark Knight" and display the recommendations.