Netflix SVD Project

Team: Christopher, Frank, Ignacio, Lane

During this project, we will present four cases:

1. Recommending movies based on ratings from similar users that only rate certain genres using cosine similarity.
2. Recommending movies based on ratings from users that rate movies from different genres (and other features) and display movies from similar users using consine similarity.
3. Recommending movies based on similarity to movies by projecting the user movies into the feature space and back. This case is covered in the thesis.
4. Predicting the user's ratings based using only SVD.

1. Following up from the project slides, we recreated a similar matrix which has ratings split into three genres: Action, Comedy and Romance.

```
movieRatings = [
    3 2 1 0 0 0 0 0 0;
    1 2 3 0 0 0 0 0 0;
    4 4 4 0 0 0 0 0 0;
    0 0 0 2 2 2 0 0 0;
    0 0 0 4 1 3 0 0 0;
    0 0 0 0 0 0 2 4 5;
    0 0 0 0 0 0 1 5 5;
];
movieTitles = {'Aliens', 'John Wick', 'Top Gun: Maverick', 'Dumb and
Dumber', 'Superbad', 'The Hangover', 'Ever After', 'Romeo + Juliet', 'The
Notebook'};
movieGenres = {'Action', 'Action', 'Action', 'Comedy', 'Comedy', 'Comedy',
'Romance','Romance', 'Romance'};
```

Next we decompose the matrix using SVD. (the top main features will be the three genres so lets trim it to only keep that).

```
[U, S, V] = svd(movieRatings);
numGenresK = 3;
Uk = U(:, 1:numGenresK);
Sk = S(1:numGenresK, 1:numGenresK);
Vk = V(:, 1:numGenresK);
```

We will add new user Jane, with only one rating in her ratings vector.

```
janesRatings = [0 0 4 0 0 0 0 0 0];
```

We will project Jane's ratings vector to the genre space to get her genre strength vector.

```
janesGenreStrength = janesRatings * Vk;
```

Next we will do the same for every other user and compare their genre strength vector to Jane's. We calculate the similarity by calculating their cosine distance where a smaller distance represents a similar vector.

```
numUsers = size(movieRatings, 1);
cosineDistances = zeros(1, numUsers);

for i = 1:numUsers
    otherUserRatings = movieRatings(i, :);
    otherUserGenreStrength = otherUserRatings * Vk;

    dotProduct = dot(janesGenreStrength, otherUserGenreStrength);
    normProduct = norm(janesGenreStrength) * norm(otherUserGenreStrength);
    cosineSimilarity = dotProduct / normProduct;
    cosineDistances(i) = 1 - cosineSimilarity;
end
```

Next we find the cosine distances = 0 to decide which user has a similar genre taste to Jane. This is because the cosine distances will either be 0 or 1 since we trimmed the matrix to only have singular values for each genre.

```
similarUsers = find(cosineDistances == 0);
```

Next we will just recommend movies of one of the similar user's ratings and filter out ones they haven't rated and ones jane has.

```
recommendedMovies = find(movieRatings(similarUsers(1), :) > 0 &
janesRatings == 0);
```

Then display the recommended movies and genres.

```
disp('Janes movies:');
```

```
Janes movies:
```

```
disp(movieTitles(janesRatings ~= 0));
```

```
    {'Top Gun: Maverick'}
```

```
disp('Janes genres:');
```

```
Janes genres:
```

```
disp(unique(movieGenres(janesRatings ~= 0)));
```

```
    {'Action'}
```

```
disp('Recommended movies:');
```

```
Recommended movies:
```

```
disp(movieTitles(recommendedMovies));
```

```
    {'Aliens'}    {'John Wick'}
```

```
disp('Recommended genres:');
```

```
Recommended genres:
```

```
disp(unique(movieGenres(recommendedMovies)));
```

```
    {'Action'}
```

Setup for cases 2, 3, and 4 using real data:

Data was pulled from https://grouplens.org/datasets/movielens/ using the small data set of ~100K ratings to ~10K movies from ~600 users.

```
[movieRatings, movieGenres, movieIDToTitle, movieIDToGenres,
movieIDToIndex, uniqueMovieIDs, totalMovies, totalUsers, averageRatings,
moviesCSV] = ImportCSVData('ratings.csv', 'movies.csv');
```

```
Time to import and process data: 1.485 seconds.
Total users: 610
Total movies: 9742
Total ratings: 100836
```

Since this data is in comma delimited row form, we iterate through each row and store it in a matrix. However this data has gaps in movies, so to save processing time, we only store the ratings for movies present, but create maps to recall back to their original id which will be useful when fetching their title and genres. This prevents a matrix with columns for movies full of zeroes which slows down SVD process. Code for this import is a function below.

Next we perform Singular Value Decomposition on the movieRatings matrix which will be used for cases 2 and 3. We use the econ tag to perform economy-size decomposition. Since we are dealing with a big matrix and only need singular vectors that correspond to non-zero singular values. This is essentially trimming the data to only keep the most important parts able to reconstruct the matrix. This will reduce computation time. (~0.3 sec vs 2.3 sec)

```
tic;
[U, S, V] = svd(movieRatings, 'econ');
svdComputeTime = toc;
```

This produces the matrices U (user to features), S (feature strength), and V (movies to features).

Key Note: In case 1, this was split in a way that we could see genre strength. However, using real data (i.e. ratings to multiple movies and genres), it is no longer possible to say exactly what the S matrix represents. Instead we call it feature strength which could be genres, certain movie year(s), movies with certain actor(s), etc.

Even though we already did an economy-size decomposition, we can further reduce the dimensions. We will keep the top 200 features.

```
k = 200;
```

```
k = min(k, size(U, 2));
Uk = U(:, 1:k);
Sk = S(1:k, 1:k);
Vk = V(:, 1:k);
```

Next we will produce a new empty ratings vector for a new user.

```
userRatings = zeros(1, totalMovies);
```

We want to produce targeted ratings so that we can get good targeted recommendations. We will only rate movies that are from certain genres since that will be the easiest feature to target. If we had more maps of data, we could also target certain actors, certain movie years, etc.

```
numRatings = 5;

% List of available genres:
% Action, Adventure, Animation, Children, Comedy, Crime, Documentary,
Drama, Fantasy, Film-Noir, Horror, IMAX, Musical, Mystery, Romance, Sci-Fi,
Thriller, War, Western

targetGenres = {'Drama', 'Romance'};
%movieIdxWithGenres = FilterMoviesContainsGenres(moviesCSV, targetGenres,
movieIDToIndex);
movieIdxWithGenres = FilterMoviesOnlyGenres(moviesCSV, targetGenres,
movieIDToIndex);
randomMovieIdxWithGenres = randperm(length(movieIdxWithGenres), numRatings);
userMoviesToRate = movieIdxWithGenres(randomMovieIdxWithGenres);
```

We will set high ratings for random movies in the target genres.

```
userRatings(userMoviesToRate) = randi([4, 5], 1, numRatings);
```

2. Similar to case 1, we calculate the cosine distances between each user's feature strength just with real data.

```
tic;
% Project user's ratings into feature space
userFeatureStrength = userRatings * Vk;

cosineDistances = zeros(1, totalUsers);
for i = 1:totalUsers
    otherUserRatings = movieRatings(i, :);
    otherUserGenreStrength = otherUserRatings * Vk;

    dotProduct = dot(userFeatureStrength, otherUserGenreStrength);
    normProduct = norm(userFeatureStrength) * norm(otherUserGenreStrength);
    cosineSimilarity = dotProduct / normProduct;
    cosineDistances(i) = 1 - cosineSimilarity;
end
```

```
cosineDistanceComputeTime = toc;
```

Key Note: Now that users are rating multiple movies from multiple genres, our cosine distances will no longer just be 0 or 1. Instead they will be values between 0 and 1. This is because movies now have multiple features and not just genres (which is all we kept in case 1). However, like before, we want the ones closest to 0 since that still represents the most similar feature strength.

Next we trim and sort the cosine distances from low to high so that we only consider the first few users with similar feature strength.

```
topUsersToConsider = 3;
[~, sortedIndices] = sort(cosineDistances, 'ascend');
topUsers = sortedIndices(1:topUsersToConsider);
```

Lets prepare a vector to save the recommended movies.

```
ratingThreshold = 3.5;

nonRatedMovies = userRatings == 0;
recommendedMovies = zeros(1, totalMovies);
for i = 1:topUsersToConsider
    topUserIndex = topUsers(i);
    topUserRatings = movieRatings(topUserIndex, :);
    % Filter out movies that don't meet the rating threshold
    highlyRated = topUserRatings >= ratingThreshold;
    % Only keep the movies the user has not rated
    notRated = highlyRated & nonRatedMovies;
    % Update the recommendedMovie entries to 1
    recommendedMovies = recommendedMovies | notRated;
end
```

The recommended movies vector now represents 1 for movies that we should recommend (Logical vector). We convert this into movie indices.

```
recommendedMovieIndices = find(recommendedMovies == 1);
```

Lets randomize the order so we don't recommend only the first entries since the ratings go from smallest to largest movie ID. This step isn't necessary but prevents seeing same movies each time.

```
randomIndices = randperm(length(recommendedMovieIndices));
```

Now we can trim the list so we only show 20 random recommended movies.

```
numRecommendedMovies = 20;
recommendedMovieIndices =
recommendedMovieIndices(randomIndices(1:numRecommendedMovies));
```

We save this data, including the movies the user rated, in table format so that it is easy to display.

```
ratedMovieTitles = cell(numRatings, 1);
```

```
ratedMovieGenres = cell(numRatings, 1);
ratedMovieRatings = zeros(numRatings, 1);
for i = 1:numRatings
    movieIndex = userMoviesToRate(i);
    movieID = uniqueMovieIDs(movieIndex);
    ratedMovieTitles{i} = movieIDToTitle(movieID);
    ratedMovieGenres{i} = movieIDToGenres(movieID);
    ratedMovieRatings(i) = userRatings(userMoviesToRate(i));
end
ratedMovies = table(categorical(ratedMovieTitles),
categorical(ratedMovieGenres), ratedMovieRatings, 'VariableNames',
{'Movie', 'Genres', 'User Rating'});
```

```
recommendedMovieTitles = cell(numRecommendedMovies, 1);
recommendedMovieGenres = cell(numRecommendedMovies, 1);
recommendedAverageRatings = zeros(numRecommendedMovies, 1);
for i = 1:numRecommendedMovies
    movieIndex = recommendedMovieIndices(i);
    movieID = uniqueMovieIDs(movieIndex);
    averageRating = averageRatings(movieIndex);
    if isKey(movieIDToTitle, movieID)
        recommendedMovieTitles{i} = movieIDToTitle(movieID);
        recommendedMovieGenres{i} = movieIDToGenres(movieID);
    else
        recommendedMovieTitles{i} = num2str(movieID);
        recommendedMovieGenres{i} = 'N/A';
    end
    recommendedAverageRatings(i) = averageRating;
end
recommendedMovies = table(categorical(recommendedMovieTitles),
categorical(recommendedMovieGenres), recommendedAverageRatings,
'VariableNames', {'Movie', 'Genres', 'Average Rating'});
```

Lets go ahead and display all this information.

```
disp('Movies Rated by New User:');
```

```
Movies Rated by New User:
```

```
disp(ratedMovies);
```

```
                              Movie                                  Genres        User Rating
    ____________________________________________________________    _____________    ___________

    Shine (1996)                                                    Drama|Romance         5     
    Last Life in the Universe (Ruang rak noi nid mahasan) (2003)    Drama|Romance         5     
    Walk in the Clouds, A (1995)                                    Drama|Romance         4     
    Step Up (2006)                                                  Drama|Romance         4     
    You Can Count on Me (2000)                                      Drama|Romance         5     
```

```
disp([num2str(numRecommendedMovies), ' Random Movies by ',
num2str(topUsersToConsider),' Similar Users:']);
```

```
20 Random Movies by 3 Similar Users:
```

```
disp(recommendedMovies);
```

```
                  Movie                                   Genres                    Average Rating
    ______________________________________    ________________________________    ______________

    Who's Afraid of Virginia Woolf? (1966)    Drama                                    4.1
    Kundun (1997)                             Drama                                    3.3
    Easy Rider (1969)                         Adventure|Drama                          3.6
    Harold and Maude (1971)                   Comedy|Drama|Romance                     4.3
    Forrest Gump (1994)                       Comedy|Drama|Romance|War                 4.2
    Ghost and Mrs. Muir, The (1947)           Drama|Fantasy|Romance                    3.9
    Boot, Das (Boat, The) (1981)              Action|Drama|War                         4.2
    Hustler, The (1961)                       Drama                                    4.3
    Where the Money Is (2000)                 Comedy|Drama                             3.3
    Somewhere in Time (1980)                  Drama|Romance                            3.1
    Little Voice (1998)                       Comedy                                   3.9
    Exorcist, The (1973)                      Horror|Mystery                           3.8
    Mystic Pizza (1988)                       Comedy|Drama|Romance                     3.4
    Affair to Remember, An (1957)             Drama|Romance                            3.9
    Sabrina (1954)                            Comedy|Romance                           3.8
    Saving Private Ryan (1998)                Action|Drama|War                         4.1
    Manhattan (1979)                          Comedy|Drama|Romance                     4.1
    American Beauty (1999)                    Drama|Romance                            4.1
    Escape from New York (1981)               Action|Adventure|Sci-Fi|Thriller         3.4
    To Kill a Mockingbird (1962)              Drama                                    4.1
```

3. The third case is the initial method covered in the thesis where we project the user's movies into the feature space and then back into the movie space which will give movies that share the most features with the movies the user rated.

```
tic;
userFeatureStrength = userRatings * Vk;
userFeatureStrengthToMovies = userFeatureStrength * Vk';
userProjectComputeTime = toc;
```

We set the entries for movies the user already rated to negative infinity so that they end up at the bottom of the sorted list. (High values now represent movies with similar features)

```
userFeatureStrengthToMovies(userMoviesToRate) = -inf;
```

We can also add an average rating filter so we don't get movies below a rating threshold like case 2.

```
ratingThreshold = 3.5;

for i = 1:totalMovies
    averageRating = averageRatings(i);
    if averageRating < ratingThreshold
        userFeatureStrengthToMovies(i) = -inf;
    end
```

```
end
```

Next we sort them from highest to lowest strength and display the first 20 movies.

```
numRecommendedMovies = 20;
[~, recommendedMovieIndices] = sort(userFeatureStrengthToMovies, 'descend');
```

Lets update the recommendMovies table.

```
recommendedMovieTitles = cell(numRecommendedMovies, 1);
recommendedMovieGenres = cell(numRecommendedMovies, 1);
recommendedAverageRatings = zeros(numRecommendedMovies, 1);
for i = 1:numRecommendedMovies
    movieIndex = recommendedMovieIndices(i);
    movieID = uniqueMovieIDs(movieIndex);
    averageRating = averageRatings(movieIndex);
    if isKey(movieIDToTitle, movieID)
        recommendedMovieTitles{i} = movieIDToTitle(movieID);
        recommendedMovieGenres{i} = movieIDToGenres(movieID);
    else
        recommendedMovieTitles{i} = num2str(movieID);
        recommendedMovieGenres{i} = 'N/A';
    end
    recommendedAverageRatings(i) = averageRating;
end
recommendedMovies = table(categorical(recommendedMovieTitles),
categorical(recommendedMovieGenres), recommendedAverageRatings,
'VariableNames', {'Movie', 'Genres', 'Average Rating'});
```

We will display the new recommended movies and also re-display the movies rated movies so that we can compare.

```
disp('Movies Rated by New User:');
```

```
Movies Rated by New User:
```

```
disp(ratedMovies);
```

```
                                 Movie                                 Genres        User Rating
    ____________________________________________________________    _____________    ___________

    Shine (1996)                                                    Drama|Romance         5     
    Last Life in the Universe (Ruang rak noi nid mahasan) (2003)    Drama|Romance         5     
    Walk in the Clouds, A (1995)                                    Drama|Romance         4     
    Step Up (2006)                                                  Drama|Romance         4     
    You Can Count on Me (2000)                                      Drama|Romance         5     
```

```
disp(['Top ', num2str(numRecommendedMovies), ' Movies Recommended to New
User:']);
```

```
Top 20 Movies Recommended to New User:
```

```
disp(recommendedMovies);
```

```
                 Movie                                      Genres                     Average Rating
    __________________________________________    ___________________________________    ______________

    English Patient, The (1996)                   Drama|Romance|War                           3.7
    Out of Sight (1998)                           Comedy|Crime|Drama|Romance|Thriller         3.9
    Emma (1996)                                   Comedy|Drama|Romance                        3.9
    Secrets & Lies (1996)                         Drama                                       4.6
    Femme Nikita, La (Nikita) (1990)              Action|Crime|Romance|Thriller               4.1
    Roman Holiday (1953)                          Comedy|Drama|Romance                        4.1
    Circle of Friends (1995)                      Drama|Romance                               3.8
    Willy Wonka & the Chocolate Factory (1971)    Children|Comedy|Fantasy|Musical             3.9
    Sling Blade (1996)                            Drama                                       3.8
    What's Eating Gilbert Grape (1993)            Drama                                       3.8
    Dead Man Walking (1995)                       Crime|Drama                                 3.8
    Do the Right Thing (1989)                     Drama                                         4
    Trainspotting (1996)                          Comedy|Crime|Drama                            4
    Saving Grace (2000)                           Comedy                                      3.9
    Rear Window (1954)                            Mystery|Thriller                            4.3
    Sliding Doors (1998)                          Drama|Romance                               3.6
    Virgin Suicides, The (1999)                   Drama|Romance                               3.8
    Hustler, The (1961)                           Drama                                       4.3
    Romeo and Juliet (1968)                       Drama|Romance                               3.7
    Jacob's Ladder (1990)                         Horror|Mystery                              3.6
```

This approach presents more movies with similar genres since we are comparing movie feature strengths and not just users with similar feature strengths.

4. The last case applies SVD to the ratings matrix including the new users ratings added on at the end and reconstructs the matrix. This creates rating predictions for every user using learned patterns.

First let's add the new user's ratings to the end of the ratings matrix.

```
movieRatings = [movieRatings; userRatings];
```

Next we have to recompute the SVD matrices.

```
tic;
[U, S, V] = svd(movieRatings, 'econ');
Uk = U(:, 1:k);
Sk = S(1:k, 1:k);
Vk = V(:, 1:k);
```

Next we just reconstruct the matrix.

```
movieRatingsReconstructed = Uk * Sk * Vk';
reconstructedSVDComputeTime = toc;
```

Since we are only working with the new user, we will only pull the ratings for the last user added.

```
predictedUserRatings = movieRatingsReconstructed(end, :);
```

Next we will filter our movies the user has already rated. Similar to case 3, we set them to negative infinity so that they end up at the bottom of the list.

```
predictedUserRatings(userMoviesToRate) = -inf;
```

Then we just sort the predicted ratings and display the top 20 ratings.

```
numRecommendedMovies = 20;
[predictedUserRatings, recommendedMovieIndices] =
sort(predictedUserRatings, 'descend');
```

Lets update the recommendMovies table.

```
recommendedMovieTitles = cell(numRecommendedMovies, 1);
recommendedMovieGenres = cell(numRecommendedMovies, 1);
recommendedAverageRatings = zeros(numRecommendedMovies, 1);
for i = 1:numRecommendedMovies
    movieIndex = recommendedMovieIndices(i);
    movieID = uniqueMovieIDs(movieIndex);
    averageRating = averageRatings(movieIndex);
    if isKey(movieIDToTitle, movieID)
        recommendedMovieTitles{i} = movieIDToTitle(movieID);
        recommendedMovieGenres{i} = movieIDToGenres(movieID);
    else
        recommendedMovieTitles{i} = num2str(movieID);
        recommendedMovieGenres{i} = 'N/A';
    end
    recommendedAverageRatings(i) = averageRating;
end
recommendedMovies = table(categorical(recommendedMovieTitles),
categorical(recommendedMovieGenres), recommendedAverageRatings,
'VariableNames', {'Movie', 'Genres', 'Average Rating'});
```

We will display the new recommended movies and also re-display the movies rated movies so that we can compare.

```
disp('Movies Rated by New User:');
```

```
Movies Rated by New User:
```

```
disp(ratedMovies);
```

```
                                 Movie                                    Genres        User Rating
    ___________________________________________________________    _____________    ___________

    Shine (1996)                                                   Drama|Romance         5
    Last Life in the Universe (Ruang rak noi nid mahasan) (2003)   Drama|Romance         5
    Walk in the Clouds, A (1995)                                   Drama|Romance         4
    Step Up (2006)                                                 Drama|Romance         4
    You Can Count on Me (2000)                                     Drama|Romance         5
```

```
disp(['Top ', num2str(numRecommendedMovies), ' Movies Recommended to New
User:']);
```

```
Top 20 Movies Recommended to New User:
```

```
disp(recommendedMovies);
```

```
                     Movie                                       Genres                    Average Rating
    __________________________________________    ___________________________________    ______________

    English Patient, The (1996)                   Drama|Romance|War                           3.7
    Titanic (1997)                                Drama|Romance                               3.4
    Emma (1996)                                   Comedy|Drama|Romance                        3.9
    Out of Sight (1998)                           Comedy|Crime|Drama|Romance|Thriller         3.9
    Secrets & Lies (1996)                         Drama                                       4.6
    Femme Nikita, La (Nikita) (1990)              Action|Crime|Romance|Thriller               4.1
    Roman Holiday (1953)                          Comedy|Drama|Romance                        4.1
    Willy Wonka & the Chocolate Factory (1971)    Children|Comedy|Fantasy|Musical             3.9
    Circle of Friends (1995)                      Drama|Romance                               3.8
    Sling Blade (1996)                            Drama                                       3.8
    Mystic Pizza (1988)                           Comedy|Drama|Romance                        3.4
    What's Eating Gilbert Grape (1993)            Drama                                       3.8
    Dead Man Walking (1995)                       Crime|Drama                                 3.8
    Trainspotting (1996)                          Comedy|Crime|Drama                            4
    Do the Right Thing (1989)                     Drama                                         4
    Saving Grace (2000)                           Comedy                                      3.9
    Sliding Doors (1998)                          Drama|Romance                               3.6
    Rear Window (1954)                            Mystery|Thriller                            4.3
    Virgin Suicides, The (1999)                   Drama|Romance                               3.8
    Hustler, The (1961)                           Drama                                       4.3
```

Case 1

Pro: Features in Sk represent genres so it's easy to visualize.

Con: Unrealistic data for movie ratings.

Case 2

Pro: Easily finds users with similar genre/feature taste in movies

Con: May require more data manipulation when presenting a select amount of movies to avoid movies with a ton of ratings

Case 3

Pro: Does not require SVD recalculation like case 4 and is very quick

Con: Features will become "outdated" as new users are added / ratings are changed

Case 4

Pro: Able to easily predict ratings for all non-rated movies for all users

Con: Requires recalculation of SVD for each new user / change to ratings matrix

Another thing to try for yourself is to check what changes if you target movies with more than just the target genres. Uncomment line 54 above and comment line 55 to see how the results change!

There is also a way to work with larger CSV data sets by only importing a portion of it. Look at the import code to enable filtering on import. There is a larger data set you can get from here with ~33M ratings from ~331K users to 86K movies.

The end!

```
% Display time to calculate important sections of code
disp(['Time to process SVD: ', num2str(svdComputeTime), ' seconds.']);
```

```
Time to process SVD: 0.29121 seconds.
```

```
disp(['Time to process cosine distances: ',
num2str(cosineDistanceComputeTime), ' seconds.']);
```

```
Time to process cosine distances: 0.30758 seconds.
```

```
disp(['Time to process feature projection: ',
num2str(userProjectComputeTime), ' seconds.']);
```

```
Time to process feature projection: 0.0030591 seconds.
```

```
disp(['Time to process SVD and reconstruct: ',
num2str(reconstructedSVDComputeTime), ' seconds.']);
```

```
Time to process SVD and reconstruct: 0.32212 seconds.
```

Functions refactored out to improve readability:

These functions, other than maybe the cosine similarity functions are specifically tailored for this data.

Import CSV Data into Matrices

```
function [movieRatings, movieGenres, movieIDToTitle, movieIDToGenres,
movieIDToIndex, uniqueMovieIDs, totalMovies, totalUsers, averageRatings,
moviesCSV] = ImportCSVData(ratingsFile, moviesFile)

tic;
% Import ratings and movies CSV files which are comma delimited
ratingsCSV = readtable(ratingsFile, 'Delimiter', ',',
'PreserveVariableNames', true);
moviesCSV = readtable(moviesFile, 'Delimiter', ',',
'PreserveVariableNames', true);

% Filter data (useful for larger data sets) - not used currently
filterData = false;
if filterData
    minUserID = 0;
    maxUserID = 1000;
    minMovieID = 0;
    maxMovieID = 100000;

    filteredRatings = ratingsCSV.movieId >= minMovieID & ratingsCSV.movieId
<= maxMovieID & ratingsCSV.userId >= minUserID & ratingsCSV.userId <=
maxUserID;
    filteredMovies = moviesCSV.movieId >= minMovieID & moviesCSV.movieId <=
maxMovieID;
    ratingsCSV = ratingsCSV(filteredRatings, :);
    moviesCSV = moviesCSV(filteredMovies, :);
end

% Generate a list of unique genres
uniqueGenres = unique(split(join(string(unique(moviesCSV.genres)), '|'),
'|'));

% Create helper map functions to easily call back to data
movieIDToTitle = containers.Map(moviesCSV.movieId, moviesCSV.title);
movieIDToGenres = containers.Map(moviesCSV.movieId, moviesCSV.genres);
genreToIndex = containers.Map(uniqueGenres, 1:length(uniqueGenres));

% Create new maps for user and movie IDs to indices
uniqueUserIDs = unique(ratingsCSV.userId);
uniqueMovieIDs = unique([ratingsCSV.movieId; moviesCSV.movieId]);
userIDToIndex = containers.Map(uniqueUserIDs, 1:length(uniqueUserIDs));
movieIDToIndex = containers.Map(uniqueMovieIDs, 1:length(uniqueMovieIDs));

% Get the total amount of unique users, movies, and genres
totalUsers = length(uniqueUserIDs);
totalMovies = length(uniqueMovieIDs);
totalGenres = length(uniqueGenres);

% Prepare the movie ratings and movie genres matrices
movieRatings = zeros(totalUsers, totalMovies);
movieGenres = zeros(totalMovies, totalGenres);

% Populate the movie ratings matrix with the imported data
for i = 1:size(ratingsCSV, 1)
    userIndex = userIDToIndex(ratingsCSV.userId(i));
    movieIndex = movieIDToIndex(ratingsCSV.movieId(i));
    movieRatings(userIndex, movieIndex) = ratingsCSV.rating(i);
end

% Populate the movie genres matrix with the imported data
for i = 1:size(moviesCSV, 1)
    % Split the genres string into each genre
    movieGenresList = split(moviesCSV.genres{i}, '|');

    for j = 1:length(movieGenresList)
```

```
        % Get the index of the genre in the genres map
        genreIndex = genreToIndex(movieGenresList{j});

        % Set the corresponding entry in the matrix to 1
        % Where the movie index is the row and the genre index is the column
        movieIndex = movieIDToIndex(moviesCSV.movieId(i));
        movieGenres(movieIndex, genreIndex) = 1;
    end
end

% Not used in calculations above, just a visual on import
totalRatings = nnz(movieRatings);

% Calculate the average rating for each movie then round it to 1 decimal
% place
averageRatings = round(sum(movieRatings, 1) ./ sum(movieRatings ~= 0, 1),
1);

% Store time it took to import and process all the data
importComputeTime = toc;

% Display information about data set import
disp(['Time to import and process data: ', num2str(importComputeTime), '
seconds.']);
disp(['Total users: ', num2str(totalUsers)]);
disp(['Total movies: ', num2str(totalMovies)]);
disp(['Total ratings: ', num2str(totalRatings)]);

end
```

Filter Movies that Contain Target Genres:

```
function [filteredMovieIndices] = FilterMoviesContainsGenres(moviesCSV,
targetGenres, movieIDToIndex)
    % Start with all movies
    filteredMovies = moviesCSV;

    % Loop over each target genre
    for i = 1:length(targetGenres)
        % Filter movies that contain the current target genre
        genre = targetGenres{i};
        filteredMovies = filteredMovies(contains(filteredMovies.genres,
genre), :);
    end

    % Convert movie IDs to indices
    filteredMovieIndices = cell2mat(values(movieIDToIndex,
num2cell(filteredMovies.movieId)));
end
```

Filter Movies that Contain Only Target Genres:

```
function [filteredMovieIndices] = FilterMoviesOnlyGenres(moviesCSV,
targetGenres, movieIDToIndex)
    % Join the target genres into a single string with '|' as the separator
    targetGenresStr = strjoin(sort(targetGenres), '|');

    % Filter movies that match the target genres string
    filteredMovies = moviesCSV(matches(moviesCSV.genres,
targetGenresStr), :);

    % Convert movie IDs to indices
    filteredMovieIndices = cell2mat(values(movieIDToIndex,
num2cell(filteredMovies.movieId)));
end
```