Design Decisions

TA feedback: How to quantify 'popular' that was the focus of some questions, and to expand upon the questions. Every attribute that is included should be required. Divisions of relations should be done to reduce redundancy.

Decisions based on feedback:

- Popular can/will be quantified based on multiple attributes, namely their rating, score (which can be filtered with having a minimum scoredBy value), members, or possible averaging or aggregating (based on some criteria for reviewers/users) review scores or the number of favorites from users. By deciding on using these to quantify, and in combination with the original reluctance for the requirement of attributes rank and popularity (both rankings for score and members, respectively), rank and popularity were excluded.
- Most of the schema was decided to have already been good, splitting up animes + their information from anime + genres, users + their information from users + their favorite animes, and reviews + their information from reviews + their categorical scores, in order to eliminate data redundancy in storing anime, user, and review data, while also listing animes and their genres, users and their favorites, and reviews and their categorical scores, in a queryable manner.

Additional decisions:

- Dropping the studio attribute of the anime relation, because of multiple factors
    - Because multiple studios can collaborate on an anime, the factor of how these studios contributed could vary, so the fact a studio participated on the anime may not be the best barometer for the studio's influence on the anime's performance
    - Did not want to either: have the primary key of the anime relation be (aID, studio), or have a new relation for (aID, studio) pairs (+ studios are not confirmed to have 100% unique names)
- Chose to use user ID number instead of username as uID despite usernames also being unique identifiers, because of many wildly inappropriate/offensive usernames, not suitable for work/school

Cleaning Process

Imported anime_filtered.csv (used the filtered dataset that have already removed rows with missing data for birth dates, location, and gender)

- Relation - Anime:
    - Selected *anime_id, title, type, episodes, source, rating, score, scored_by, members, favorites, studio, genre* columns
    - Dropped rows with "na" value in any columns
    - Dropped rows with value "None" in the *rating* column
    - Renamed columns *anime_id* to *aID* and *score_by* to *scoreBy*
    - Removed the *genre* column
    - Wrote this dataframe to **anime.csv** without the index column.
- Relation - Genre:

        - Selected *anime_id*, *genre* columns from the <u>anime</u> relation
        - Extracted only the first part of the ratings for readability (e.g. from “G - All Ages” to “G”)
        - Split the long string of genre into a list of genres for each *aID, then s*eparated the list of genres for each aID into separate rows with one aID and one genre per row
        - Wrote this dataframe to **genre.csv** without the index column.

Imported users_filtered.csv

- Relation - MALUser:
    - Selected *username*, *user_id*, *gender*, *birth_date* (taking username for joining the other tables in a later step) columns
    - Renamed columns *user_id* to *uID*, *birth_date* to *dateOfBirth*.
    - Dropped the row with a NULL username.
    - Dropped rows with invalid dates or NULL values for *dateOfBirth.*
    - Wrote this dataframe to **maluser.csv** without the index column.

Imported profiles.csv

- Relation - Favorites:
    - Selected *profile*, *favorites_anime*
    - Renamed column *profile* to *uID* and *favorite_anime* to *aID*
    - Converted the *favorites_anime* values into a list of strings, then broke down the list into individual rows of *uID* and *aID* pairs
    - Removed rows with Null values
    - Wrote this dataframe to **favorites.csv** without the index column.

Imported review_data.csv

- Relation - Review:
    - Selected *uid*, *profile*, *anime_uid*, *score*, *scores*
    - Renamed column *uid* to *rID* as *uid* in the dataset represents rating IDs, *profile* to *uID*, *anime_uid* to *aID*
    - Removed *scores* column
    - Dropped duplicated reviews
    - Filtered out rows with *overallScore* not within 0 to 10
    - Wrote this dataframe to **review.csv** without the index column.
- Relation - Scores:
    - Selected *rID, scores* from <u>review</u> relation
    - Separated *scores* into individual rows for each category and their score, naming the columns *category, score*
    - Filtered out rows with *score* not within 0 to 10
    - Wrote this dataframe to **scores.csv** without the index column.

Attached is the file used to clean the data, data_cleaning.py, in case anything was unclear.