CS 121 Final Project Proposal

For the final project, you may choose to work with up to one other person. For partner-finding, use the #partner-search Discord channel to share your interests, any ideas, and preferences for working with a partner.

Student name(s): Amelia Whitworth, Jennie Chung

Student email(s): awhirwor@caltech.edu, jjchung@caltech.edu

DATABASE/APPLICATION OVERVIEW

In this proposal, you will be "pitching" your project, in which you have some freedom in choosing the domain of with a structured set of requirements that bring together the course material in a single project, from design to implementation to tuning. Keep in mind that unlike CS 121 assignments, you are free to publish/share your project, which can be useful for internship or job applications. In terms of scope, you should shoot for 8-12 hours on this project, though you are free to go above and beyond if you choose.

First, what type of database application are you planning on designing? Remember that the focus of this project is the database DDL and DML, but there will be **a small Python command-line application** component to motivate its use and give you practice applying everything this term in an interactive program. Don't worry too much about implementation at this step, and you can jot down a few ideas if you have more than one. Just think about something you would like to build "if you had the data" which could be simulated with a command-line program in Python. Your command-line program will start with a main menu with usage options for different features in your application. Staff are here to help you with scoping!

Here's a list of some application ideas to get you started. These encompass most of the applications within the scope of this project we anticipate students might be interested in, but if there's a different application that meets the requirements for your DB schema and implementation, you are welcome to ask!

Application ideas:

- Store simulations
    - Can auto-generate datasets, or find ones online (e.g. bookstore, Starbucks dataset, video games, Animal Crossing store dataset, Pokemart, etc.)
    - Online delivery simulation
    - Tables could include products, purchases, users, employees, carts, specials/promos, loyalty program, etc.
- Gameplay/leaderboard management
    - Trivia or Jeopardy! dataset

    - Wordle leaderboard, possible basic implementation of game (Kaggle dataset available)
    - Random Pokemon collecting/Pokedex management
        - Pokemon (Pokedex vs. collection), moves, types, collection, Pokemart
    - Players, game entities, categories, etc. for a game you implement or a public game score database
- Budgeting/utility application
    - User configuration of income/expenses
    - Set monthly categories, goals, limits
    - Add some gamification/achievements
    - Tables are harder for this one, unless you get particularly creative
- Messageboard, blog, or feed
    - Tables can include messages, users, admins, entry logs, event calendar, etc.
- Application system (e.g. applications for jobs, student clubs, research, adoption agency, course registration, etc.)
- Some creative idea with social media (most platforms have datasets published)
- Office Hour/other type of Queue simulator
- Stock market simulator

Proposed Database and Application Program Answer (3-4 sentences) :

We want to use a movie database for our project, likely with rating or user information. For our application, we could create a way for watchers/audiences to sort and filter the different works of media. For instance, we could allow users to find books with a certain actor before a certain year, the highest rated books of each genre, etc.

Next, where will you be getting your data? We will post a list of datasets to get you started, but you can find many here and here (look for ones in CSV file(s), which you will break into schemas similar to the Spotify assignment; we can also help students convert JSON to CSV if needed). You are also welcome to auto-generate your own datasets (e.g. with a Python script) and staff are more than happy to help you with the dataset-finding process, which can take longer than the rest of the starting design process.

Data set (general or specific) Answer:

We found a couple of datasets on kagle that we think would be interesting. Below are some of the ones we were thinking of using

https://github.com/malcolmosh/goodbooks-10k-extended

Who is the intended user base and what is their role? You will need to have at least one client usertype, and one admin. These different users may have different permissions granted for tables or procedural SQL.

***Client user(s) Answer*:**

We want the users to be able to specify criteria about the kinds of media that they are interested in. Then they will be able to use our databases to find media that suits their interests.

Admin user(s) Answer:

Be able to recommend works of media which will be consumed by users based on user preferences/specifications, to increase watching/reading activity.

REQUIRED FEATURES

The full specification of the project will outline the requirements of the project, but you remember you should aim to spend 8-12 hours (it replaces the final exam and A8), depending on how far you want to go with it. The time spent on finding or creating a dataset will vary the most. For the proposal, you will need to brainstorm the following (you can change your decisions later if needed).

E-R Model: There will be an ER model component, but it is not required for the proposal.

DDL: What are possible tables you would have for your database? Remember to think about your application (e.g. command-line functions for user prompts to select, add, update, and delete data, either as a client or an admin user). We encourage you to include the attributes and their types as well at this step, though not required. Include at least 4 tables in your response, as well as any design decisions you may run into in your schema breakdown.

Answer:

books(book_id, original_title, isbn, isbn_13, authors, original_publication_year, language_code)

- Relation to store information about each book.

books_details(book_id, description, pages, genre)

- Contains additional information about each book such as the book description, the number of pages in the book, and the genre.

ratings(user_id, book_id, rating)

- Will be used to store how a particular user rates a book. Can also be used to determine which books are most highly rated to give certain awards to or promote from the bookstore's perspective.

to_read(user_id, book_id)

- This will be a relation that individual users can update based on what's popular so that a bookstore or reading club can know what books have been recently highlighted

authors(last_name, first_name, book_id)

- Here we will break apart the author column to separate multiple authors on one book. This relation can be used to get information about all the books written by a certain author.

Queries: Part of the application will involve querying data in a way that would be useful for a client (e.g. searching for "Puzzle" games made in the last 5 years, ordered by year and price in a Video Game Store database, or finding all applicants who are pending for an adoption agency). Identify at least 3 queries that would make sense in a simple command-line application. In your answers, provide a brief description of the query or pseudocode, as well as the purpose in your application. These will likely be implemented as SQL queries within Python functions, wrapping your SQL queries (the methods of which will be taught in class). You are welcome (and encouraged) to add more, though not required.

Answers:

1. Get all the books made in a particular year, genre and language

```
SELECT original_title
        FROM books NATURAL JOIN book_details
        WHERE genre = "chosen genre" AND language = "chosen language"
        AND year = "chosen year";
```

2. Get the number of ratings for each author sorted by most rated to lowest rated

```
SELECT COUNT(*) as num_ratings

        FROM books NATURAL JOIN ratings

        GROUP BY name

        ORDER BY num_ratings
```

3. Get the highest-rated movie in each genre and year, and sort from oldest to newest

```
WITH
(
SELECT genre, year,  MAX(get_avg_rating(book_id)) as max_rating
        FROM books NATURAL JOIN ratings NATURAL JOIN book_details
        GROUP BY genre, year
        ORDER BY year;

)
AS max_ratings
SELECT *
        FROM max_ratings JOIN books ON (max_ratings.genre = book_details.genre AND
        max_ratings.year = books.year AND
        max_rating = media.rating);
```

Other possible queries

Get the average rating for each book

Procedural SQL: You will need to implement at least 1 each of a UDF, procedure, and trigger in your project. Identify at least one of each of these here. For each:

1. What is the name of the function/procedure/trigger?
2. Why is it motivated in your database? Remember for example that there are overheads for triggers, so they shouldn't be trivial.

Consider some examples in class/HW, such as logging DML queries, adding an extra layer of constraint-handling (e.g. the overdraft example from lecture), etc. For procedures, remember that these can be called in an application program written in a language like Python or Node.js, so these are especially useful to avoid writing queries in such application programs, *especially* SQL that performs **`INSERT/DELETE/UPDATE`** which you do not want to leave to an application user. Remember that you can also set permissions for different users to access tables and procedures. In your Python program, you can connect to your database as different users defined in your database (similar to the Node.js program El demo'd a while back).

Answers

UDF(s):

1. **Name: get_average_rating**
2. **Write a UDF that will get us the average rating for any book in the database. The UDF can take any book as the input and then it will find the average of all ratings of that book in the ratings relation. This can be used to identify what books are the most popular.**

Procedure(s):

1. **Name: get_highest_rated_seasonal_movie**
2. **This procedure would take the author name as an input and will add the highest average rated book by that author to the to_read relation. This procedure will make it possible to recommend the best books in the to_read relation**

Trigger(s):

1. **Name: update_average_ratings_per_genre(or other attribute of interest)**
2. **We will make a summary table of the most highly rated, most rated, how common is on the to read list. Then if a new rating is added the trigger will update these catagories**

Database Performance: At this point, you should have a rough feel for the shape and use of your queries and schemas. In the final project, you will need to add at least one index and show that it makes a performance benefit for some query(s). You don't need to identify what index(es) you choose right now (that comes with tuning) but you will need to briefly describe how and when you would go about choosing an index(es). Refer to the material on indexes if needed.

Performance Tuning Brainstorming:

Create an index for genre, since we expect users will often want to procure information based on this. For instance, in our queries section, we use genre to retrieve the highest rated books from each genre.

"STRETCH GOALS"

If you are particularly eager for a certain application (have your own start-up in mind?), it is easy to over-scope a final project, especially one that isn't a term-long project. You can list any stretch goals you might have "if you had the time" which staff can help give feedback on prioritizing.

Answer:
Want to create a book recommendation system based on user's "bookshelf," using a combination of python and SQL to get the necessary information.

POTENTIAL ROADBLOCKS

List any problems (at least one) you think could come up in the design and implementation of your database/application.

Answer: We anticipate the data cleaning part of this project will be quite substantial, and will probably have data-specific questions on cleanup decisions that are less obvious (e.g. when to include/exclude columns based on null values).
Also, we may run info problems when trying to parse the comma separated authors in books.csv to create the authors table.

OPEN QUESTIONS

Is there something you would like to learn how to implement in lecture? Any other questions or concerns? Is there anything the course staff can do to help accommodate these concerns?

Answer: We were wondering to what extent we are going to need to combine our SQL queries with python. Will this be covered in class next week?

Should we create a materialized view/view for the summary stats, or a new relation? We hope to use triggers for this, so mirroring Set 5 it seems like we should use a materialized view, but we don't know if this is completely necessary.

Have fun!