CS 121 25wi Final Project Proposal

Due: Friday Feb. 14th 2024, 11:30PM

- *Note: Proposal deadline is exception to weekly Monday deadlines for sets, between A3 and A4. Try to submit by Thursday, but we will accept by Friday.* ***A4 is due on Tuesday Feb. 19th*** *due to Monday holiday but will be shorter with proposal overlap.*

Notable 25wi updates:

- app.py with function stubs is **required** in proposal this year
  - [02/10] El went through some student project ideas and will provide another version of template since we're simplifying client/admin breakdown this year
  - A summary of minimum requirements is added to this document (show_options() and starts to function stubs)
- Setup-proposal.sql (DDL) **required** in proposal
- Procedural SQL and DB Performance **both removed** from proposal but kept for reference
  - Triggers will be **optional** for the Final Project this year
- Other requirements may be reduced a bit to account for lost week during wildfires

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. There will be optional work time in lecture to find partners and ask about any questions with El around; we encourage you to attend to aim for finalizing a project idea, a partner, and any dataset/scoping questions you have. You are welcome to remove any suggestions/notes from this proposal in your proposal, as long as your required components are clearly in the order specified (we've highlighted text for you to answer in blue for ease).

Note: If working with another student, only one of you needs to submit the required files to CodePost, but ***both of you should have equal contributions; i.e. you should ideally work through this document together.*** *For ease of grading please have the other student submit a simple* **collaboration.txt** *file to their CodePost submission in the following format (failure to do so may result in deductions):*

Names: *<student1>, <student2>*

CodePost emails: *<email1>, <email2>*

For full credit, we are looking for a robust proposal demonstrating careful consideration of the motivation, design process, and implementation plan for your application; you should expect to spend 3-4 hours minimum on your proposal (finding a dataset may take the longest). For each part of this proposal that requires your response, you should have 2-3 sentences (bullet points are encouraged as well) justifying your answers. We strongly encourage you to include as much detail as you can in this proposal (this is your only assignment this week) and you can include any diagrams and SQL brainstorming (e.g. a start to your DDL) with your submission on CodePost. If you want to add any additional planning, you can include a diagram as a PDF/PNG/etc. and/or .sql files, though not required. You can also include a starter Python program with function stubs for a command-line implementation you will complete for the Final Project (without SQL integration).

Summary of Files to Submit (max 3MB, reach out to El if you have trouble with this limit, but usually compressing images and/or the PDF file using your system's viewer or an online compressor will work):

- **proposal.pdf (a copy of this document filled out)**
- ***additional diagrams/sketches/brainstorming not otherwise required; you can include these at the bottom of your proposal document or as additional files in your submission.***
- **setup_proposal.sql (start to your DDL)** - your final project submission will rename this appropriately
- **app.py (or app_client.py/app_admin.py) function stubs**

The advantage of adding these in your proposal is to get you started in advance, and also to get feedback from the staff on your design and implementation so far.

Student name(s): Jonathan Lin, Enoch Luk

Student email(s): jonathan@caltech.edu, eluk@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; you can find a template from last year here, which may be adjusted slightly before the Final Project is released, but gives you an idea of the breakdown. 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. **For students who took CS 132, you may alternatively use a web-based application which El can help with.** Staff are here to help you with scoping!

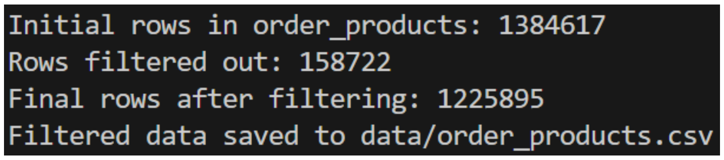
Proposed Database and Application Program Answer (3-4 sentences) :

The project we're aiming to create will help data analysts for supermarkets optimally stock their shelves - what and when to fill their shelves with. The database will contain info about past purchases, where past products were located in the physical store, each order and their components, etc. The command-line application would then act like a mediator between the SQL database and the terminal, abstracting away the sometimes complicated database code for a simpler interface. A client will be able to observe trends, popular products, etc about a given store. An admin has more functionality to do things like adding new orders, adding a new product, deleting old products, 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 or using ChatGPT similar to A2 Part D) 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:
https://www.kaggle.com/competitions/instacart-market-basket-analysis/data

We got our dataset from Kaggle at the above link. The dataset was too large, so we used the trim.py file in our project to remove the last 300,000 rows of the orders.csv file because it was too large (3.5 million rows). We also have a filter.py script that removes any rows in order_products.csv whose order_id doesn't exist in orders and product_id doesn't exist in products.csv. This is the result of running that script:



```
Initial rows in order_products: 1384617
Rows filtered out: 158722
Final rows after filtering: 1225895
Filtered data saved to data/order_products.csv
```

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

The client users are lower-level store managers or analysts who primarily interact with the system in a read-only capacity. Their role focuses on querying data for insights into product demand, customer purchasing patterns, and stock movement trends. They can analyze sales by product, aisle, and department, identify high-demand or slow-moving items, etc. However, they do not have permissions to modify inventory data or any other high-level actions.

Admin user(s) Answer:

The admin users are upper level (e.g. regional) store managers who oversee inventory management and supply chain decisions. They will input data and modify the SQL database when new batches of orders come in or other kinds of changes are made. Unlike client users, admins have full control over inventory adjustments and store-wide planning.

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).

DDL: What are possible tables you would have for your database? Include at least 4 tables in your response. 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). To make this step easier, thinking about the menu options your application provides, as well as the queries you might want to support, is helpful to narrow down the information you'll want to model.

1. Provide a general breakdown of your database schema and tables below, as well as any design decisions you may run into in your schema breakdown. You can provide any notes/questions that you would like us to be aware of for your proposed project.
2. Start your DDL with CREATE TABLE statements in the **setup_proposal.sql** file attached in your submission. We will not grade strictly on DDL, but your final submission should have documentation, etc. and we will prioritize evaluating your table breakdown, choice of attribute domains, and keys/relationships.

Answer:

Our proposed database schema is designed to support efficient tracking and analysis of supermarket transactions and inventory management. The schema consists of five primary tables: aisles, departments, products, orders, and order_products. The aisles table includes an aisle_id as its primary key and an aisle name to categorize products based on their physical location within the store. Similarly, the departments table contains a department_id and a department name, allowing products to be grouped into broader categories. The products table is central to the schema, holding details such as product_id, product_name, and foreign keys aisle_id and department_id that link each product to its specific aisle and department. The orders table records customer transaction data with fields like order_id (primary key), user_id (to track which customer placed the order), customer_order_number (indicating the sequence of a customer's orders), order_dow, order_hour_of_day, and days_since_prior_order to capture temporal trends and reorder intervals. Finally, the order_products table serves as a table linking orders and products, with a composite primary key comprising order_id and product_id, as well as fields for add_to_cart_order and reordered to track the sequence of product additions and whether the product was reordered. This design enforces referential integrity through foreign key constraints with cascading deletes, ensuring that related records are automatically maintained while providing a robust framework for both client and admin functionalities.

The schema supports queries that allow clients to analyze top-selling products per aisle, identify shopping trends by time and day, and generate reports on inventory depletion.

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.

Identify and display the top-selling products based on how many times each product was ordered (volume). This helps clients quickly see which products are most in demand.

Pseudocode:

```
SELECT product_id, product_name, COUNT(*) AS total_orders
FROM orders JOIN order_products USING (order_id)
    JOIN products USING (product_id)
GROUP BY product_id, product_name
ORDER BY total_orders DESC
LIMIT 10;
```

2

Determine which aisles in the supermarket receive the highest order volume. This information can be used to optimize store layout or marketing strategies.

Pseudocode:

```
SELECT aisle_id, aisle, COUNT(*) AS order_count
FROM orders JOIN order_products USING (order_id)
    JOIN products USING (product_id)
    JOIN aisles USING (aisle_id)
GROUP BY aisle_id, aisle
ORDER BY order_count DESC;
```

3.

Retrieve and display the order history for a specific customer. This lets a client view all past orders associated with a given customer ID, including product details and order dates.

Pseudocode:

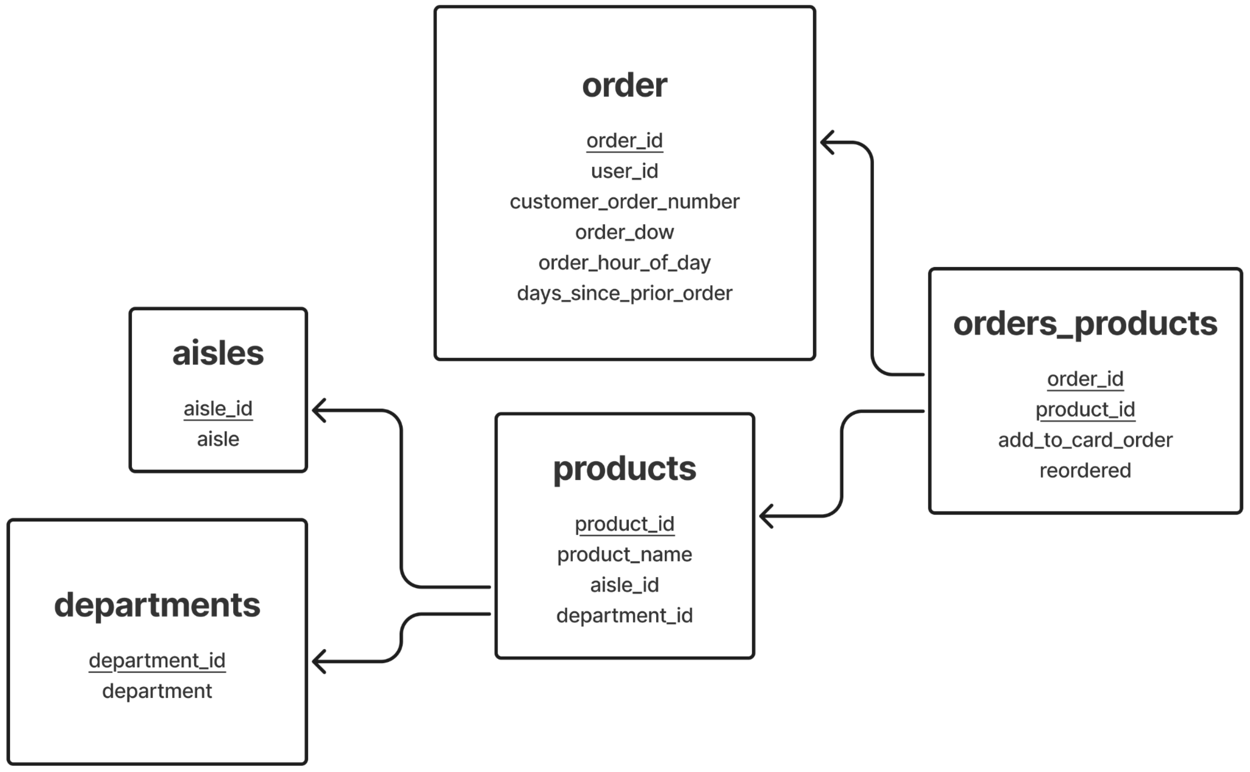
```
SELECT order_id, order_date, product_name, add_to_cart_order
  FROM orders JOIN order_products USING (order_id)
     JOIN products USING (product_id)
  WHERE user_id = customer_id;
```

E-R Model: There will be an ER model component, which we will cover in a few weeks. ER models are an essential part of database projects, and can vary in conventions. For the proposal, you will not need to complete formal ER diagrams, but you will need to provide 1) a visual representation of your database (a very useful strategy for the design and planning phase, especially when proposing a database model to collaborators with varying technical backgrounds) and 2) some visualization of one possible program flow for a client user from start to finish. For 2), you are encouraged to also include an example program flow for an admin user, though are not required for the proposal. You have flexibility in how you represent your database application here and both diagrams do not necessarily need to be separate, but you must:

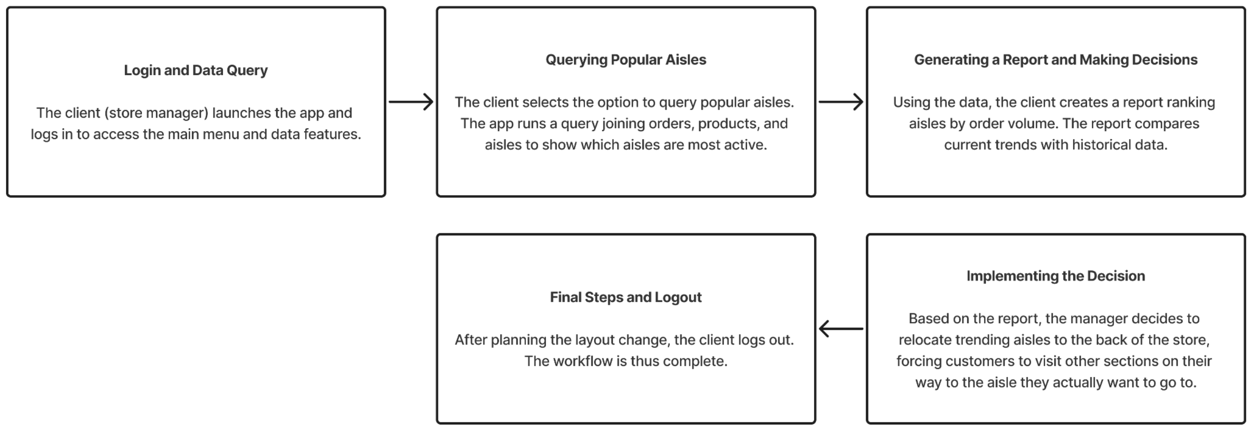
1) Have some representation of (all) proposed schemas and relationships from your DDL (with arrows for foreign keys between tables) that is interpretable.
    - Use annotations (or equivalent) for your proposed queries/client interaction such that someone (e.g. a reviewer) could identify which tables/attributes would be relevant to clients/users. Consider color-coding of keys, etc. to improve your proposed schema organization. Students have also previously added questions for design/implementation decisions they are uncertain about (e.g. "Should this *menu* table be broken down into *drinks* and *foods* or kept as *menu* with the category attribute?")
2) Include at least one visualization (e.g. flow-chart) of a runthrough of your client program from start to finish (it may help to think of your command-line client program as a simulation of a website that interacts with your database). The following is an example (a text-based version, though a visual flow chart is an improvement at the proposal stage to show the different possible edge cases):
    - User starts program (or visits website) -> creates an account after being prompted to login/create account (inserts into loyal customers table with inputted data, such as email) -> logs into created account if success (validation is done on database side) -> is presented a list of options -> inputs ('menu') for show all cafe products listed in '<n>: <product name> <price>' format, where <n> starts with 1 -> inputs 3 to add a Black Coffee to their cart -> confirms 'y' to finish adding to cart and submit order -> order is added to orders table with customer's id and product id -> gets a success

message and estimated wait time ('simulated') and is directed back to the main menu -> chooses to log out and finish the order

Schema Diagram (may also alternatively as diagrams.png or diagrams.pdf)



Example User Flowchart(s):


Login and Data Query
The client (store manager) launches the app and logs in to access the main menu and data features.
Querying Popular Aisles
The client selects the option to query popular aisles. The app runs a query joining orders, products, and aisles to show which aisles are most active.
Generating a Report and Making Decisions
Using the data, the client creates a report ranking aisles by order volume. The report compares current trends with historical data.
Implementing the Decision
Based on the report, the manager decides to relocate trending aisles to the back of the store, forcing customers to visit other sections on their way to the aisle they actually want to go to.
Final Steps and Logout
After planning the layout change, the client logs out. The workflow is thus complete.

app.py Client Program

Now that you have brainstormed your DDL and UI flowchart, you have a good start to draft your client application for the project. For the scope of the proposal, we want students to get started early with app.py, prioritizing "proof of concept" and modeling the possible UI flow. app.py will be the client program interfacing with your MySQL database, and we have provided a template to get you started.

For the proposal, your app.py should at minimum have:

- A list of 3 or more possible options for your program in show_options()
    - These will likely correspond to your diagrams and queries above
- Function stubs for each option/feature in your app.py
    - Use docstrings/comments to specify any arguments/input, and returns/output
    - You can use comments to show your current ideas for implementing the functionality
    - For input prompts, you are welcome to "mock" examples; this will make it easier to understand the intended flow for your application, and also easily translate to SQL when implementing the FInal Project requirements
    - Think about possible helper functions as you work through these
- If you choose to include both a client and admin application for your proposal, you can submit app_client.py and app_admin.py (there may be overlap with get_conn() and show_options(), with get_conn() having a separate username/password for admin privileges, covered in Wed. 02/12 slides)

You aren't expected to spend more than 30 minutes or so on this part when starting with the template, but should use to make as much progress early on before the Final Project later. Students can see some past demos in lectures/OH this week as well. Include a comment at the top of the program with any questions you have about your proposed client program and feedback you would like us to prioritize (we won't give in-depth feedback to this otherwise, knowing it's a draft subject to change).

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:

Would it be beneficial to precompute commonly queried queries? Sacrifice space complexity in exchange for lower time complexity.

"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:

If time permits, we would explore two advanced features to enhance store operations and customer insights. The first is aisle optimization to minimize travel distance, which leverages the order in which customers add products to their carts. By analyzing purchasing sequences, we can identify frequently bought-together items and determine optimal product placement to reduce unnecessary travel within the store or to force customers through unrelated aisles in hopes of them buying more things. This would help store managers rearrange aisles in a way that improves shopping convenience, reduces congestion in high-traffic areas, and increases sales. A potential approach involves clustering products that frequently appear together in a single transaction and using these patterns to generate data-driven aisle layout recommendations.

The second stretch goal is an AI-driven product recommendation system that transforms customer purchase history into latent features to uncover hidden relationships between products. Instead of relying on simple frequency-based recommendations, this system would analyze purchasing behavior to infer underlying shopping themes. For example, if a customer consistently buys organic produce and gluten-free items, the model could suggest other niche health-related products that align with their preferences. This could be applied at both the individual level (for personalized recommendations) and the store level (to identify trending product categories and improve inventory decisions).

Potential roadblocks

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

Answer:

One potential roadblock is scalability and performance issues, as querying large datasets with millions of orders could slow down response times. Without proper indexing or query optimization, retrieving trends on popular products or order frequency may take too long, making real-time insights impractical. Another challenge is handling missing or incomplete data, such as gaps in the days_since_prior_order field, which could impact the accuracy of demand forecasting. Imputation techniques or careful filtering may be needed to address this.

Managing user roles and security is another potential issue, as regional store managers will have different permissions than lower-level managers. Ensuring proper access control while preventing unauthorized modifications to inventory will require role-based authentication and careful SQL query design. Another challenge is ensuring data consistency when multiple managers interact with the system simultaneously. If one manager updates stock levels while another queries inventory, race conditions could cause inaccurate reporting. Using ACID-compliant transactions (like most common SQL applications) and proper isolation levels will help mitigate these risks.

Review

This year, we are adding a small component to get students feedback from their peers and to practice giving feedback on proposed projects. You do not need to spend a lot of time on feedback (but may earn up to 3 additional points for above-and-beyond effort, such as reviewing multiple student posts with thoughtful feedback), but for for full credit you must:

- Post your overview followed by one part of your proposal (e.g. diagrams or DDL) on #final-project-review and include at least one question you have you'd like feedback on
- Review one other posted project and provide 1-3 sentences of feedback that demonstrates at least 5 minutes of reflection of trade-offs and concepts you've learned so far. Feel free to bring in your own user experiences! For example, you may have experience inputting a long last name that a website has a size limit on when creating an account, and could note this when suggesting a modification to another student's DDL for representing names.

Link to your Discord post:
https://discordapp.com/channels/1325070896913846344/1339359506836230230/1339500739801776139

We're not Discord experts, so please let us know if this link doesn't work.

Link(s) to your Discord feedback:

https://discordapp.com/channels/1325070896913846344/1339359506836230230/1339505522864422932

Collaboration

For projects with partners, this section is required (if not, you can leave it out, or note that you are decided yet). Both students should provide a brief summary of their planned workload distribution, method(s) of collaboration, and 1-2 points you are most interested to do in the project. You may also clarify any concerns/confidence for your partner work here. Feel free to provide a paragraph or bullet points; we're looking for you to have thought this out and discussed your plan for collaboration at this step.

Jonathan Answer:
I will focus on the database schema design, ensuring that the tables, relationships, and constraints align with our application's needs. I will also implement SQL queries for retrieving sales trends, analyzing customer purchasing patterns, and optimizing restocking schedules. Additionally, I will work on the command-line interface for client users, ensuring that lower-level managers can efficiently query data and generate reports. I will also work on performance optimization, making sure our queries run efficiently on large datasets. If time permits, I would be interested in exploring the AI-driven recommendation system for analyzing customer purchasing behavior.

Enoch Answer:

I will be responsible for implementing role-based access control, ensuring that regional store managers (admins) and lower-level managers (clients) have the correct permissions. I will develop transaction handling mechanisms to maintain data consistency, particularly when stock levels are updated. Additionally, I will work on the command-line interface for client users, ensuring that lower-level managers can efficiently query data and generate reports.

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:

Content is great and we're learning a lot, but we feel that sometimes the sets take too much time and we learn too much content at once. We think that slowing the course down a little would allow us to synthesize and understand the content at a deeper level rather than trying to learn as many things relevant to relational databases as possible.

Have fun!

Optional Brainstorming/Sketches/Other Notes

This is an optional section you can provide any other brainstorming notes here that may help in the design phase of your database project (you may find it helpful to refer to the early design phase in your Final Project Reflection).
