Prompt

Imagine you were building some kind of app to do something, what kind of queries would you need to be able perform the functions of the app? What kind of information would be stored? What would the database schema look like to be able to support these queries?

Submission link: https://forms.gle/iZ2SxzKpAS3cP1rf6
Submission deadline: Midnight after class (Aug 31, 11:59pm)

Note: if you see an error mentioning “data loss prevention”, make sure you’re not signed in to a school / work gmail account. In the worst case scenario you can submit via slack to avoid going past the deadline.

What we expect

A ~5 minute presentation covering the following:

- Overview of the subject matter / app: who is it for (e.g. students, people getting vaccinated, people who love cycling)? What are the different types of users, if multiple? (e.g. viewer vs administrator) What does it do?
- Overview of the schema: what are all the tables? What are the primary keys and the most important fields? What are the foreign key relationships between the tables?
    - A diagram could help here
- Overview of queries:
    - What are the most important queries our app does?

Tips on choosing a subject:

- It's fine to be creative here - you can choose something more practical or business related, a problem you've encountered at work, something related to a hobby you might have (e.g. plant growth tracker, recipe book, workout app), etc.

Example Idea: A vaccination tracker database.

- We have two types of users: patients (who sign up to get vaccinated), schedulers (who manage appointment dates and locations), and vaccinators (who perform vaccinations).
- We’ll store the data in the following tables:
    - **locations**: List of vaccination centers and cities
    - **appointment_slots**: List of appointment slots for each vaccination center
    - **patients**: Patients who have registered for a shot

    - **appointments**: Relationship table that assigns a patient to a slot and marks completion
    - **consents**: Mark that patients have read the documentation and consent to be vaccinated
- Patients can:
    - Check vaccination centres within a city.
        - *("SELECT * from vaxx_locations WHERE...")*
    - Register for an appointment with their name, e-mail, phone number, home address.
        - *("INSERT INTO patients (...) ....")*
    - Check whether they've been assigned an appointment yet
        - *"Select from appointments where patient_id="*
    - "Cancel" a future appointment if they've been assigned one
        - *"Delete from appointments where patient_id= and date > now()"*
- Schedulers can:
    - Find all patients that don't have appointments scheduled, showing those who registered earlier first
        - *("SELECT patient_name from patients join appointments where … order by ...")*
    - For an unscheduled patient, ting th all vaccination centres that have appointment slots open
    - Look at a report of vaccinations by age group
        - *"Select * from appointments join patients … group by …"*
    - Look at a report of vaccination count per week and vaccine type
        - *"CREATE VIEW "vaccinations_by_week" as select … from … group by …"*
- Vaccinators can:
    - Check that a patient has consented
    - Mark a patient as being vaccinated at a specific date and with a specific vaccine brand
        - *"Update appointments set vaccination_date=..., vaccine_type=..."*

Criteria

Every bullet point below gets a grade from 0 to 3, and the results are added up:

0 - You didn't do it

1 - You tried, even if not complete

2 - You did it and it works / was satisfactory

3 - (bonus) You went clearly above and beyond what was required

- Presentation: approx 5 minutes, you can reuse content from your project registration
    - Explanation of app use case, what is the target audience, who are the users
    - Diagram of schema, auto-generated or created by hand
    - Brief overview of schema, fields, relationships (can mention most relevant bits)
    - Explanation of app "features" and the query that is run for each feature
- Schema
    - Schema and sample data delivered in a single .SQL file that we can import or paste into dbfiddle and run.
    - Has minimum 4 tables
    - Has a schema design that makes sense for your app's use case.
    - Types set appropriately
    - Primary and foreign key constraints set appropriately
- Queries
    - Sample queries delivered in a separate .SQL file that we can copy/paste to run one by one, once schema is imported.
    - Comments included above each query explaining what it’s for
    - Has minimum 5 queries, enough to cover "features" of the app
    - Query complexity: At least 2 of the queries are more than just the simplest kind of "select x from y;" - could use joins, groupbys, functions etc.
    - "Advanced" features: At least one query using a subquery, a view (that uses multiple tables), a stored procedure, a trigger, or a feature you looked up on your own that we didn't learn in class.