CSI2532 Project Winter 2021

Competition Leaderboard

Groups of 2 ± 1 students allowed for this project. The project is split into 4 deliverables throughout the semester for a total of 30% of your final grade.

You must submit a project to satisfy the requirements of the course.

Track all Artifacts

All code, all images, all content should be in your GitHub repository.

If you are using third-party tools to manage, for example, your ER diagram then you must include the raw source files (or if an online tool, then include an export from that tool and instructions on how to use it).

If some artifacts are derived (i.e. compiled) then include instructions on how to build those artifacts AND include a copy of them (as a backup should the build process not work).

Deliverables

Deliverable 1 (5%) Hello-World

Mark	Description
2.0	GitHub repository setup
2.5	ER model
2.5	Relational model / SQL schema
1.0	SQL examples to INSERT, UPDATE, SELECT and DELETE data
1.0	README.md contains all required information
1.0	Git usage (commit messages, all students involved)
/ 10	

We will focus on getting your project up and running, getting familiar with git, GitHub, your ER diagramming tool, relational modelling, writing some SQL and documenting everything in Markdown.

The leaderboard database should only model the athletes and include details such as their name, date of birth, and identified gender.

For this delivery you will be responsible for

- Creating your project team of 2 ± 1 persons
- Creating your GitHub Classroom repository
- Creating a (simplified) ER model
- Creating a (simplified) Relational model
- Implementing the SQL schema (based on your relational model) to create your database
- Providing SQL examples
- Documenting the above in a README.md

Your README.md should contain at a minimum

- List team members including names, and student numbers
- Include your ER model (as an image and links to any raw files)
- Include your relational model (as an image and links to any raw files)
- Instructions on how you created your ER and Relational model
- A reference to your SQL schema (and how to run it)
- Additional example queries to show INSERT, UPDATE, SELECT, DELETE on your database

Deliverable 2 (5%) DB Backed Application

Mark	Description
3.0	ER model
3.0	Relational model / SQL schema
1.0	Application (read-only)
1.0	SQL seed / examples to INSERT, UPDATE, SELECT and DELETE data
1.0	README.md contains all required information
1.0	Git usage (commit messages, all students involved)
/ 10	

SQL Migrations

Database models evolve. Here the focus is to help you manage the evolution of your database overtime using migrations.

Update your ER model / relational models from Deliverable 1 to include support for `competitions`. A competition has a `name`, `venue`, a date (e.g. `start_date`, `start_time` or maybe `start_date_time`). Competitions typically run over multiple days (so they should include the ability to track `duration`).

An `athlete` can `register` for any competition. For now, do not worry about the actual scheduled events, or how well an athlete performs at the competition. Our focus is on managing model changes.

Change your schema from Deliverable 1 to be a migration. Now instead of just editing your schema directly, create a migration script to evolve your database into the desired model of Deliverable 2.

Maintain the latest copy of your schema which should align with your ER and relational models. Note that applying all SQL migrations should be equivalent to that latest schema. Document how to run your migrations.

Also, include a SQL script to seed your database with sample data. This seed should be based on your latest schema.

Application

Databases are also rarely used in isolation. So we need an application that can create, read, write and update the information in our database.

In this deliverable, we will focus on READ-ONLY access in your application. You are free to provide other capabilities as that will be required in future deliverables.

Deliverable Summary

In summary, you will be responsible for

- Updating your ER model and relational model based on the changes above
- Creating a SQL migration to track changes to your database
- Maintaining the latest SQL schema of your database
- Providing a SQL seed to populate your database with sample data
- Updating your SQL examples
- Creating an application that can READ (and display) data from your database
- Updating your README.md

Your README.md should be updated to also contain at a minimum

- Updates to your README.md for any changes that might now be out of date (e.g. ER model)
- Instructions (and a reference) to running your SQL migrations
- A reference to your SQL seed (and how to run it)

The simplest tech stack for your application is probably PHP (which has a built-in webserver) and PostgreSQL. You are free to use other technologies like Java, Ruby, Python, Node, Elixir but you will be responsible for clearly documenting how to install your applications on a Linux environment (including Mac OSX).

This could be a great opportunity to learn new technologies but do not expect much support as the professor and TA might not be familiar with your chosen tech stack.

Deliverable 3 (12%) The Application

Mark	Description
2.0	ER model
2.0	Relational model / SQL schema
2.0	SQL seed / examples / migrations
2.0	Application
1.0	README.md contains all required information
1.0	Git usage (commit messages, all students involved)
/ 10	

Now that we have some experience with ER models and relational modelling, we have an environment to manage database changes and an application that can access our database; **let's build a competition leaderboard**.

MeFit hosts an annual fitness competition but is looking to expand by allowing partners to host their competitions. Those partners will be responsible for competition registrations, payments and other logistical details, but MeFit wants to maintain control over the competition's leaderboard that tracks the ranking of athletes.

MeFit has asked you to help them build a competition's leaderboard database and application as described below.

The leaderboard needs to manage athletes. An athlete has a name, email, date of birth and an identified gender. They are identified by a globally unique ID. Athletes can include any number of additional attributes such as (as an example only) the athlete's nationality.

For every partner, we need to know their company name, the address of their headquarters, a contact person including their email and phone number.

The competitions (there can be many) of each partner are within a competition year. For each competition we need to know the competition date (it can span multiple days), the maximum number of athletes (based on identified gender), number of events, the competition address, and a contact person (including an email and phone number). Note that some events have no maximums allowing as people to join as possible.

For each event in a competition, we need to know how it is scored. The two main ways to score an event are based on time (the faster the better), on the number of repetitions (the more the better), or the weight moved (also the more the better).

But, not all events can be completed in time requiring tie breakers. Tiebreakers are scored the same as the event can be scored. The tiebreaker can be based on time (the faster the better), on the number of repetitions (the more are better), or on the weight moved (also the more the better). Not all events have tie-breakers.

If there is still a tie after that, then the athletes tie and each is awarded the same rank for that event. The overall score of an athlete is the sum of their rank on each competition, and this time the lower the number the better.

A few examples are below.

Event	Score	Time Capped	Tie-Breaker
10k run	Time	N/A	None
As many burpees in 7 minutes	# Reps	N/A	None
21/15/9 of thrusters and pull-ups in 10 minutes	Time	# Reps	Time of last round
As many rounds of A, B, C in 10 minutes	# Reps	N/A	Time of last round

A competition's leaderboard shows the rank of each athlete on each event but can also show the overall rank of the athlete. Custom leaderboards can be created by filtering the athletes based on one or more of their attributes. For example, a leaderboard based on gender; or another based on nationality AND gender.

Athlete	Points	20.1	20.2	20.3	20.4	20.5
Dan Shrum (M / 18)	1 (10 points)	2 (12:40)	1 (784 reps)	1 (131 reps)	2 (211 reps 15:12)	4 (234 reps)
Audrey Begin (F / 18)	3 (38 points)	4 (13:51)	2 (682 reps)	17 (83 reps)	3 (201 reps 16:35)	12 (208 reps 11:08)
Kevin Sourapha (M / 18)	4 (43 points)	1 (12:30)	3 (609 reps)	5 (101 reps)	6 (187 reps)	28 (--)
Matthieu Desloges (M / 18)	2 (19 points)	3 (12:53)	4 (588 reps)	3 (111 reps)	8 (169 reps 12:51)	1 (16:40)
Taylor Stewart (M / 18)	10 (66 points)	49 (--)	5 (585 reps)	2 (119 reps)	5 (200 reps 19:17)	5 (232 reps)
Eleonor Buteau (F / 18)	6 (50 points)	7 (171 reps)	6 (578 reps)	20 (66 reps)	4 (200 reps 16:58)	13 (208 reps 11:35)
Andrew Forward (M / 40)	8 (60 points)	15 (149 reps)	7 (513 reps)	11 (90 reps 07:31)	20 (125 reps 11:30)	7 (228 reps 19:43)

MeFit admins can add new companies as sanctioned partners to host competitions. Using a secure token, partners can register athletes into a competition through online API / application.

Users can search the leaderboard for partners, competitions and leaderboard. A leaderboard can be searched for athletes.

We should be able to delete from our database the partners, competitions and events. We cannot have in the database information about an event without having in the database the information about the corresponding competition (i.e. the competition in which the event belongs too). In the same way, we cannot have in the database information about competitions without having in the database the information about the corresponding partner (i.e. the partner in which the competition belongs too).

Deliverable Summary

In summary, you will be responsible for

- ER model that corresponds to the above description

- Relational model based on your ER model
- Define all constraints to ensure the correctness of the database to be created according to your Relational model. These constraints include primary keys, referential integrity constraints, domain constraints and user-defined constraints.
- Implement the database based on the relational models and constraints you have defined (using migrations)
- Implement the necessary SQL modifications (using queries and triggers). Your database should allow insert, delete and update operations of data in your database according to the referential integrity constraints that you have defined.
- Seed your database with example partners, competitions, athletes and scores
- Build at least two database views
- Build an online application / API to interact with your database

Deliverable 4 (8%) Presentation

Mark	Description
3.0	Presentation
3.0	Application
2.0	SQL seed
1.0	ER model / Relational model (SQL schema)
1.0	README.md contains all required information
/ 10	

Let's wrap up the project. You will be given approximately 10 minutes to present your project. Specifics will be forthcoming once we know the number of groups.