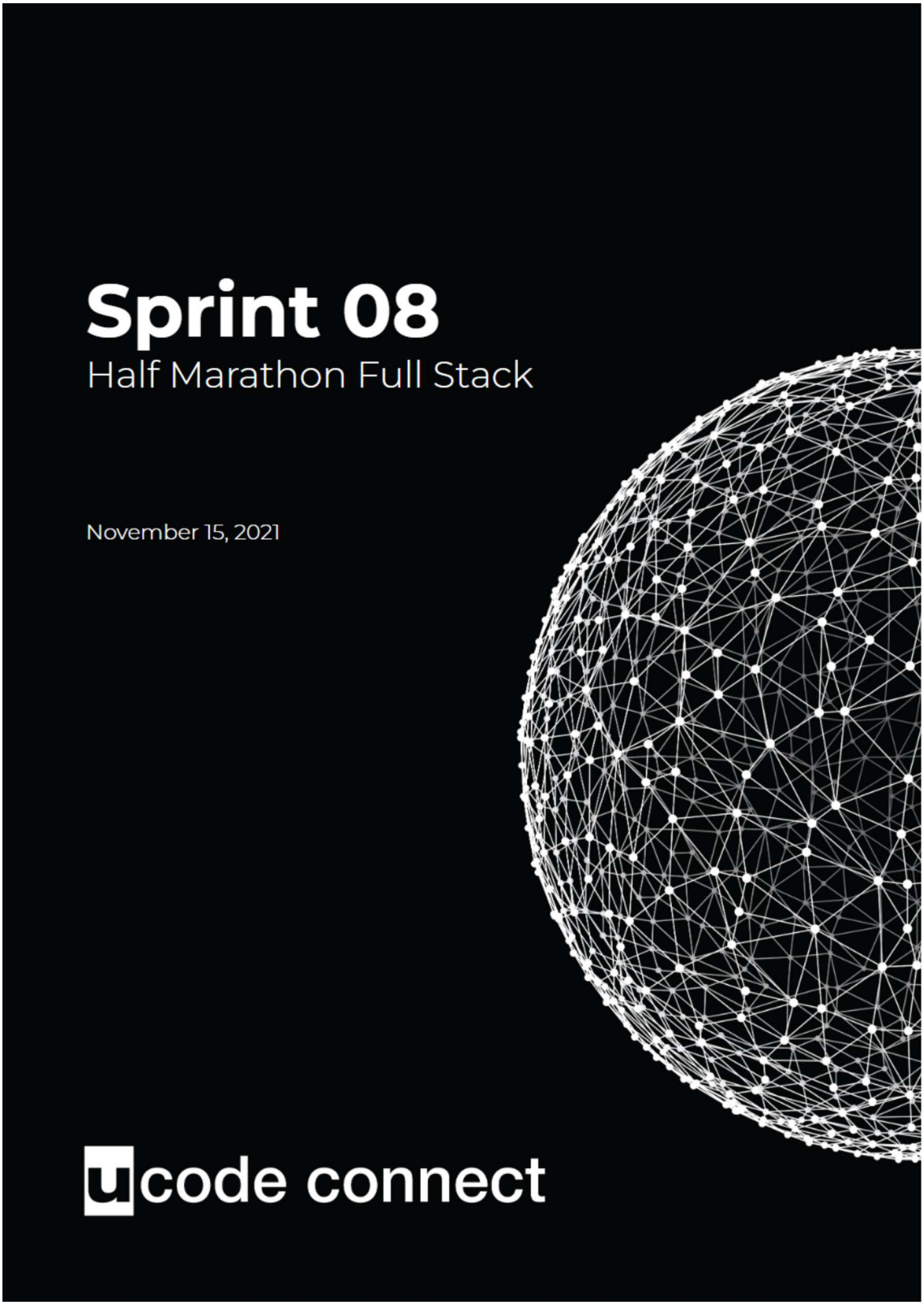
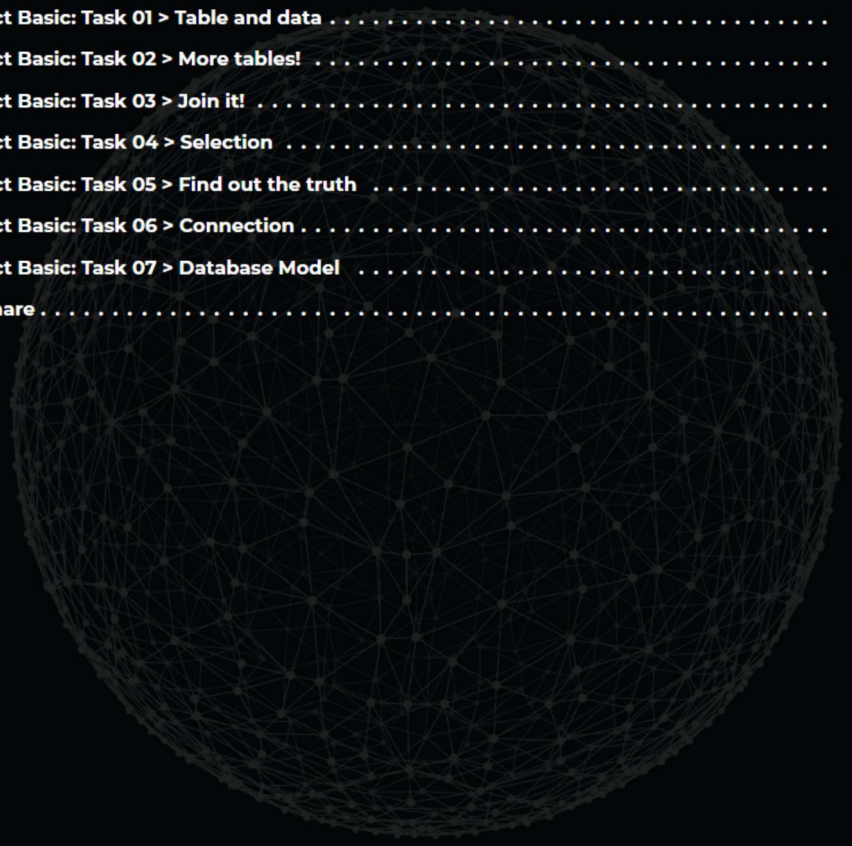

Sprint 08
Half Marathon Full Stack
November 15, 2021
ucode connect

Contents



ucode connect

Engage

DESCRIPTION

You almost did it! Two more Sprints and your young fighter course will be over!

Today you will get acquainted with SQL basics. SQL (which stands for Structured Query Language) is used to interact with databases. A database usually represents the synthesis of two elements - a data set and a database management system (DBMS).

No web service works without a database, because, regardless of its purpose, it still represents some set of information which needs to be stored somewhere and somehow. And DBMS is responsible for where and how exactly this data is stored.

DBMS consists of an integrated set of computer software that allows users to interact with one or more databases, and provides access to all of the data contained in the database. In this Sprint, your task is to understand how to operate with databases via SQL queries.

The knowledge of SQL basics is crucial for backend development on any programming language. So, take this Sprint responsibly, be curious, and do even more than mentioned in the tasks.

BIG IDEA

Databases.

ESSENTIAL QUESTION

How to deal with a database?

CHALLENGE

Connect to and operate a MySQL database using Node.js.

ucode connect

Sprint 08 | Half Marathon Full Stack **> 2**

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find solutions, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What is the difference between `SQL` and `MySQL`?
- What is the difference between `relational databases` and `non-relational databases`?
- What is a `primary key`?
- What is `auto-increment` in SQL?
- Are `NULL` values the same as that of zero or a blank space?
- What is the difference between `BETWEEN` and `IN` operators in SQL?
- What is the difference between a `primary key` and `unique constraints`?
- How to connect MySQL and Node.js?
- What is a transaction (in connection to databases)?
- What are the benefits of querying to the database via transactions?

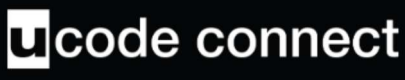
GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Read this article about relations between entities.
- Learn the basics of SQL queries.
- Install MySQL.
  If you are working with a ucode iMac and there is no MySQL on your computer install it via MSC.
  If you are working on a personal computer, install MySQL using official docs.
  Make sure that the MySQL version on your host is equal to or greater than 8.0.0.
- Find some test databases on the internet and try to make queries to them from the console.
- Read about how to connect to a database using Node.js.
- If you've done all of the above, then you are ready for some nerdy humor.
- Clone your git repository issued on the challenge page in the LMS.
- Brainstorm with other students and find more than one solution.
- Try to implement your thoughts in code.

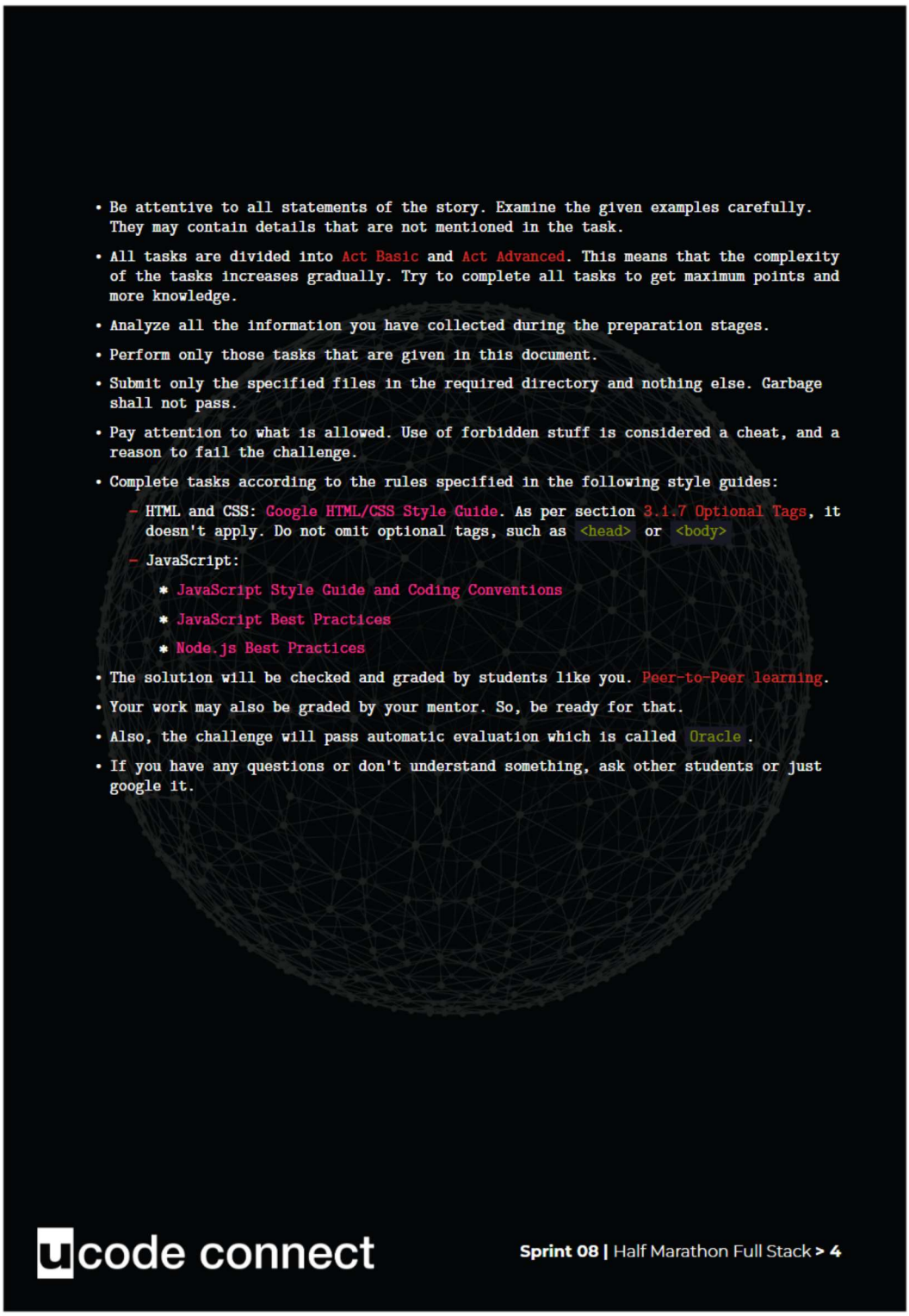
ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.



Sprint 08 | Half Marathon Full Stack > 3

- Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
- All tasks are divided into Act Basic and Act Advanced. This means that the complexity of the tasks increases gradually. Try to complete all tasks to get maximum points and more knowledge.
- Analyze all the information you have collected during the preparation stages.
- Perform only those tasks that are given in this document.
- Submit only the specified files in the required directory and nothing else. Garbage shall not pass.
- Pay attention to what is allowed. Use of forbidden stuff is considered a cheat, and a reason to fail the challenge.
- Complete tasks according to the rules specified in the following style guides:
  - HTML and CSS: Google HTML/CSS Style Guide. As per section 3.1.7 Optional Tags, it doesn't apply. Do not omit optional tags, such as `<head>` or `<body>`
  - JavaScript:
    * JavaScript Style Guide and Coding Conventions
    * JavaScript Best Practices
    * Node.js Best Practices
- The solution will be checked and graded by students like you. Peer-to-Peer learning.
- Your work may also be graded by your mentor. So, be ready for that.
- Also, the challenge will pass automatic evaluation which is called Oracle .
- If you have any questions or don't understand something, ask other students or just google it.



Act Basic: Task 00

NAME

Database and User

DIRECTORY

`t00_database_and_user/`

SUBMIT

`db.sql`

ALLOWED

SQL

DESCRIPTION

This is the very beginning of your MySQL journey! Make yourself comfortable and enjoy the ride!

You should get access to a database server. In our case, you need to access a MySQL server as a root user.

Every database starts with creation, so you need to create a database and a user to work with it. Write a SQL query to create a database called `ucode_web`. Create a user with all privileges on the new database and set the password `securepass`. The user should have access from the localhost and the username should be the same as your login.
Save all queries in the `db.sql` file.
And use `mysql -u root < db.sql` to run the script.

You must use this database and the created user in all of the following tasks.

SEE ALSO

SQL Tutorial

ucode connect

Sprint 08 | Half Marathon Full Stack **> 5**

Act Basic: Task 01

NAME

Table and data

DIRECTORY

`t01_table_and_data/`

SUBMIT

`table.sql, data.sql`

ALLOWED

SQL

LEGEND

Take a breath and clear your mind, you are going to create the Universe!

DESCRIPTION

Create a table in your database and add some heroes to it.
Create a table and name it heroes. Every hero can have a few characteristics. In our case, every hero should have a name, description, class_role, and id. All your heroes should have a unique name, so people don't get confused. Heroes can play different roles.

By analogy with role-playing computer games, the role can be one of these:

- tankman - this player takes damage instead of the group, protecting the others from attacks
- healer - this player heals others when they are hurt or take damage, thus keeping party members alive in order to tank or kill the enemies
- dps - (damage per second) - the role of DPS is to kill the enemies

The role is up to you and you can change it any time.

Your table structure will be the following:

- id - an auto incremented int value that should be a primary key
- name - a unique text field limited by 30 characters
- description - a text field
- class_role - a field with three possible values: `tankman`, `healer`, or `dps`

All the fields must not be NULL. You should handle the cases when the table already exists (here and in future tasks as well).

Submit the MySQL script in the table.sql file.

Create at least ten heroes from Marvel Earth-199999 or some other universe (even made up by you). You can create any hero, but your table should contain at least two dps heroes and two non-human heroes.

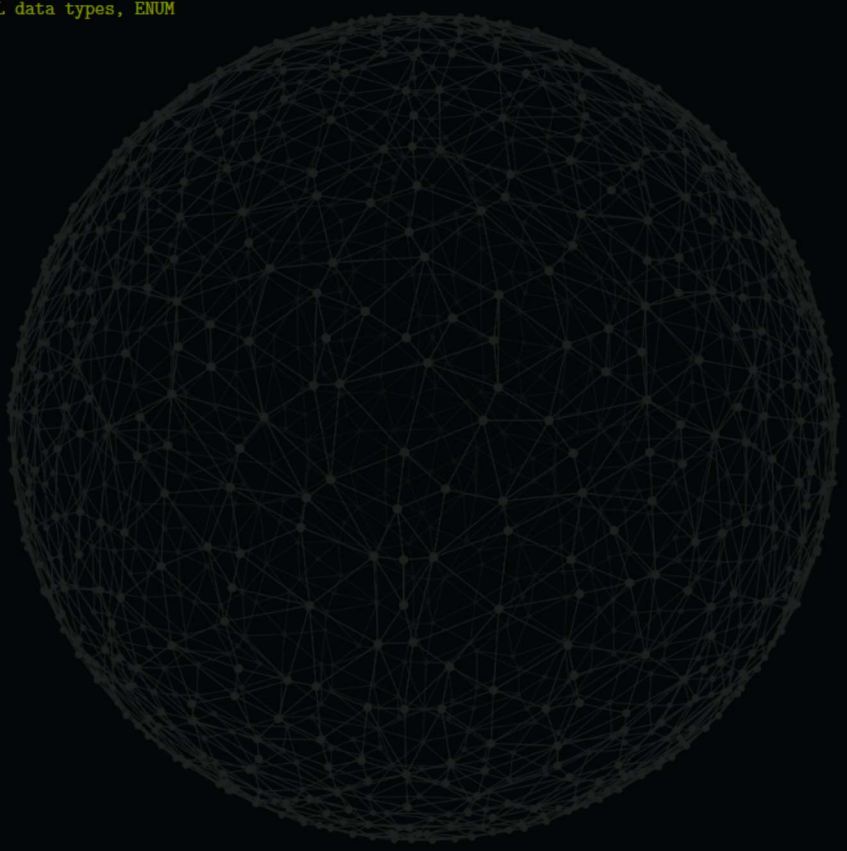
ucode connect

Sprint 08 | Half Marathon Full Stack **> 6**

Submit the MySQL script for data input in the data.sql file.

SEE ALSO

SQL data types, ENUM



ucode connect

Sprint 08 | Half Marathon Full Stack **> 7**

Act Basic: Task 02

NAME

More tables!

DIRECTORY

`t02_more_tables/`

SUBMIT

`extended.sql, data.sql`

ALLOWED

SQL

LEGEND

A hero without powers is not a hero, Nick Fury doesn't count. And what's a hero without a mighty team?

DESCRIPTION

In this task, we invite you to extend your database and create additional tables with some relationships.
Create a `powers` table with such fields:

- `id`
- `name` - a text field with the name of the hero's power
- `type` - a text field with the type of the power (can be `attack` or `defense`)

One hero can have a few powers or not have them at all, and a few heros can have the same power. So create a many-to-many relationship.

Link it to your heroes by creating a join table `heroes_powers` with fields:

- `hero_id`
- `power_id`
- `power_points` - int, the number of power points

And do not forget to limit the size of your fields. Use primary keys and specify delete actions in all your tables.

You must also create a `races` table. Add a `race_id` field to the `heroes` table. One hero can have only one race, but there can be a lot of heroes with the same race.
The `races` table will have such fields:

- `id`
- `name` - a text field

ucode connect

Sprint 08 | Half Marathon Full Stack **> 8**

What's a hero without a great team? Let's create some! Pay attention that some heroes like extra work, so they can be members of several teams simultaneously.
Create a `teams` table:

- `id`
- `name` - a text field

Link it to your heroes by creating a join table `heroes_teams` with fields:

- `hero_id`
- `team_id`

Finally, add some values to the tables you created as well. `Powers` should contain at least the following powers:

- `bloody fist`, type - `attack`, points - `110`
- `iron shield`, type - `defense`, points - `200` In addition, you can make up your own powers and assign them as either `attack` or `defense`.

`Races` should contain at least `Human` and `Kree` races. You may also add other races (existing or made up).
`Teams` should contain at least two teams: `Avengers` and `Hydra`. Feel free to add other teams (existing or made up).

At least one hero should have an iron shield, two heroes should be non-humans, and there should be at least one hero in both of the Avengers and the Hydra teams.

Submit the MySQL script for table creation and data input in the extended.sql and data.sql files respectively.

SEE ALSO

Database relationships

ucode connect

Sprint 08 | Half Marathon Full Stack **> 9**

Act Basic: Task 03

NAME

Join it!

DIRECTORY

`t03_join_it/`

SUBMIT

`join.sql`

ALLOWED

SQL

LEGEND

Nick Fury requests a report about heroes and their activity. The Universe is in danger, and he needs to know what heroes can save us all. Help him to find it out.

DESCRIPTION

Create a few queries according to the requests:

- get all the heroes with their team name. Should be displayed as NULL if the hero has no team
- get all the heroes with their powers. All the powers should be in the list, even if nobody has them
- get the heroes with their powers and teams. Only heroes with both power and team should be displayed

The WHERE conditions are not allowed in this task.

SEE ALSO

Join types

ucode connect

Sprint 08 | Half Marathon Full Stack **> 10**

Act Basic: Task 04

NAME

Selection

DIRECTORY

`t04_selection/`

SUBMIT

`avengers.sql`

ALLOWED

SQL

LEGEND

We have great heroes with such strong abilities, but how to select a leader of each team? And which team is stronger? Let's find out it!

DESCRIPTION

Create a few selections from the ucode_web database:

- select the most powerful hero from the database. Power is a sum of all skills (attack and defense). If a few heroes have equal power, the hero with the smallest id should be selected
- select the weakest hero. A weak hero is the one who has the smallest defense (sum of all defense skills)
- select all the Avengers, except the double-agent, and sort them by the total power in descending order
- get total power of both the Avengers and the Hydra teams. The most powerful team should be the last in order

ucode connect

Sprint 08 | Half Marathon Full Stack > **11**

Act Basic: Task 05

NAME

Find out the truth

DIRECTORY

`t05_find_out_the_truth/`

SUBMIT

`find.sql`

ALLOWED

SQL

LEGEND

There is a traitor in the Avengers team. They have been searching for this traitor for the last few months, and they are not even nearly close to finding the truth. Help them in their search. You should activate your superpower with SQL scripts!

DESCRIPTION

Create only one `select` to find the hero by such parameters:

- hero is in two or more teams at the same time
- hero is not human
- hero's name contains the letter 'a' (lowercase)
- hero can be tank or healer
- if there is more than one hero - select the one with the lowest id

Make sure you have enough data in the database to test your query.

ucode connect

Sprint 08 | Half Marathon Full Stack **> 12**

Act Basic: Task 06

NAME

Connection

DIRECTORY

`t06_connection/`

SUBMIT

`config.json, db.js`

ALLOWED

MySQL, JS

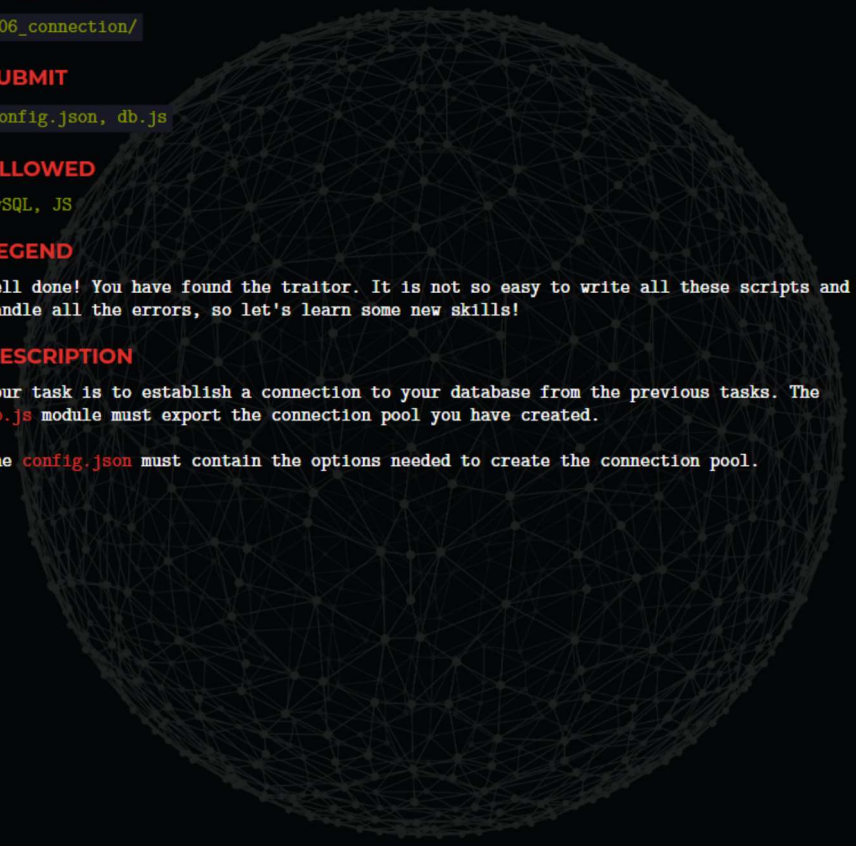
LEGEND

Well done! You have found the traitor. It is not so easy to write all these scripts and handle all the errors, so let's learn some new skills!

DESCRIPTION

Your task is to establish a connection to your database from the previous tasks. The `db.js` module must export the connection pool you have created.

The `config.json` must contain the options needed to create the connection pool.



ucode connect

Sprint 08 | Half Marathon Full Stack **> 13**

Act Basic: Task 07

NAME

Database Model

DIRECTORY

`t07_database_model/`

SUBMIT

`config.json, db.js, model.js, models/hero.js`

ALLOWED

MySQL, JS

LEGEND

As you can see, there are a lot of things you can improve. Now Nick Fury needs to improve heroes management. He gets really tired of writing all these queries, so if you don't want to do it for him you'd better find a way to make it easier.

DESCRIPTION

In order to connect to your database, use the `db.js` module you have created for the previous task.

Create a class `Model` in the file `model.js`. The class should have the following methods:

- a constructor that sets the attributes given to it
- `find(id)`
- `delete()`
- `save()`

It will describe basic behavior of all your future models. After that, create your first model `Hero` in the file `hero.js`. This class must contain all the fields from your `heroes` table and must be an extension of the `Model` class.

Implement the following features:

- `find(id)` - the method should select an object from the database and set all the object's values as properties of the class
- `delete()` - the method should delete an object by id from the database. It's a good idea to check if an id exists before deletion
- `save()` - the method should implement `INSERT` and `UPDATE` logic. You should define which type to use depending on the situation

Both `Model` and `Hero` modules must export their respective classes.

ucode connect

Sprint 08 | Half Marathon Full Stack **> 14**

Share

PUBLISHING

Last but not least, the final stage of your work is to publish it. It allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the task and better understand your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio (macOS)
- ScreenToGif - screen, webcam, and sketchboard recorder with an integrated editor (Windows)

Examples of ways to share your experience:

- Facebook - create and share a post that will inspire your friends
- YouTube - upload an exciting video
- GitHub - share and describe your solution
- Instagram - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use #ucode and #CBLWorld on social media.