

# Sprint 08

Half Marathon Web

January 13, 2021



 **code connect**

# Contents

Engage . . . . .	2
Investigate . . . . .	3
Act: Task 00 > Database and User . . . . .	5
Act: Task 01 > Table and data . . . . .	6
Act: Task 02 > More tables! . . . . .	8
Act: Task 03 > Join it! . . . . .	10
Act: Task 04 > Selection . . . . .	11
Act: Task 05 > Find out the truth . . . . .	12
Act: Task 06 > PDO . . . . .	13
Act: Task 07 > Database Model . . . . .	14
Share . . . . .	15

# 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.

There are many ways to access a database from your PHP code with the help of extensions. We encourage you to get acquainted with all of them. But since you are beginners, we ask you to use only one way - via **PDO (PHP Data Objects)**, due to its versatility and general predictability.

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 tasks.

## BIG IDEA

SQL basics.

## ESSENTIAL QUESTION

How to make queries to a database?

## CHALLENGE

Connect to and operate with a database from a PHP script.



# 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 answers, 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` ?
- Which types of relations between entities do you know?
- 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` ?
- What is `PHP Data Objects` ?
- How to connect to MySQL via PDO in PHP?
- How to handle PDO connection errors in PHP?
- What is a transaction? How does PDO manage transactions?
- What are the benefits of querying to the database via transactions?

## GUIDING ACTIVITIES

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

- Learn the basics of SQL queries.
- Google the most common questions that are asked at interviews for the web developer position. Try to answer them.
- Make sure that you have installed MySQL on your computer.
- Find some test databases on the internet and try to make queries to them from the console.
- Read about how to integrate PDO in your PHP code.
- Read about PDO predefined constants.
- If you did all the above, then you are ready for a little bit of nerdy `humor`.
- Understand the information received, decide what is paramount for you and what is not.
- 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.

- Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
- Perform only those tasks that are given in the story.
- Submit only the specified files in the required directory and nothing else. In case you are allowed to submit any files to complete the task you should submit only useful files. Garbage shall not pass.
- Pay attention to what is allowed. Use of forbidden stuff is considered a cheat and your challenge will be failed.
- It is recommended to complete tasks according to the rules specified in the [PSR-12: Extended Coding Style](#).
- The solution will be checked and graded by students like you. [Peer-to-Peer learning](#).
- If you have any questions or don't understand something, ask other students or just Google it.

# Act: Task 00

## NAME

Database and User

## DIRECTORY

```
t00/
```

## 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.

Install MySQL:

1. Clone the repository <https://gitlab.unicode.world:8443/services/ubrew.git>.
2. Open the repository and follow the README provided there.
3. Then install MySQL with ubrew.

Make sure that the MySQL version on your host is equal to or greater than 8.0.0.

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 `unicode_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.

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

## SEE ALSO

SQL Tutorial



# Act: Task 01

## NAME

Table and data

## DIRECTORY

```
t01/
```

## 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
- `race` - a text field with a default value `human`
- `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.

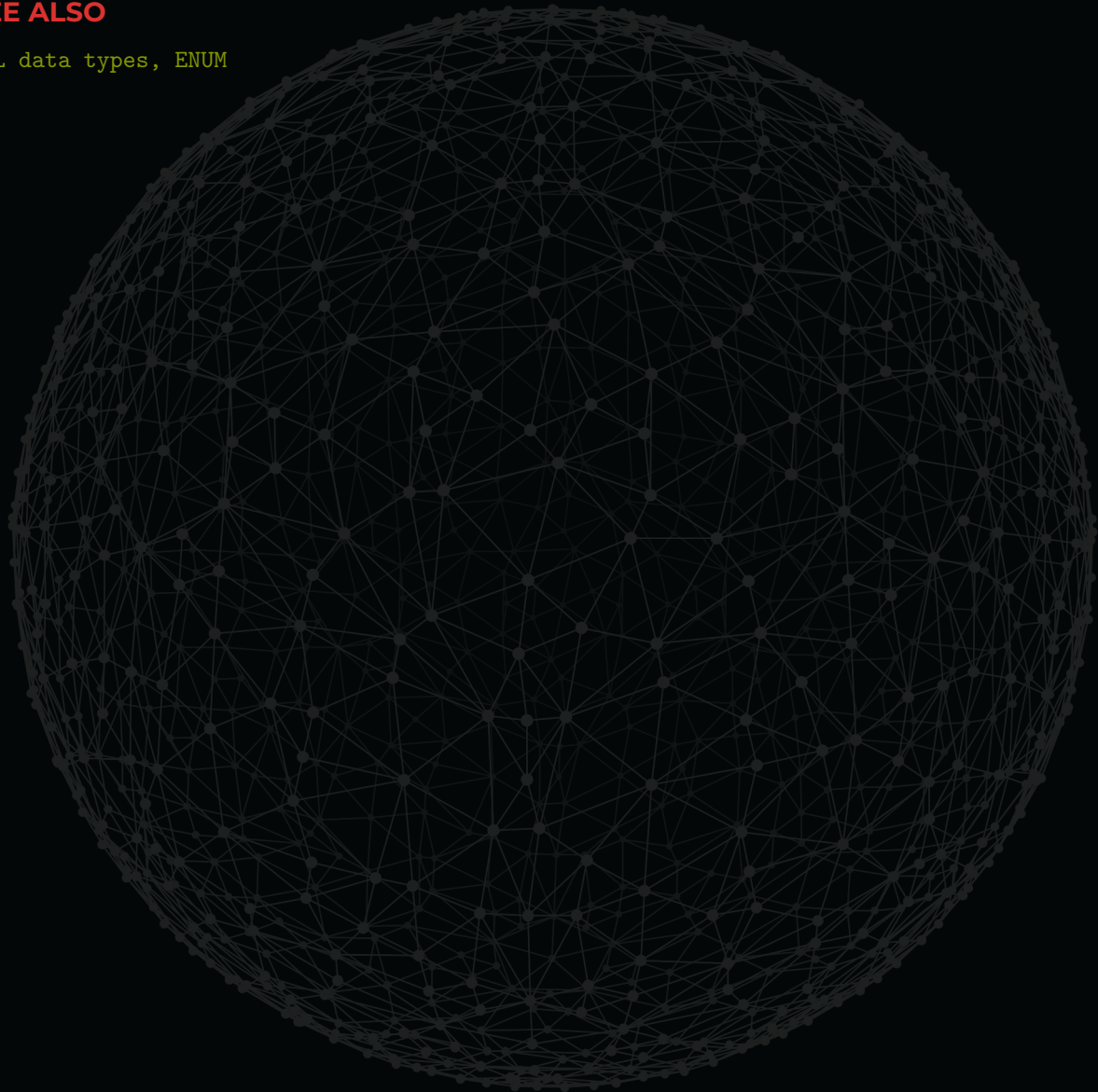
Create at least ten heroes from Marvel Earth-199999 or some other universe (even made up

by you). You can create any hero, but your table should contain at least two dps heroes and two non-human heroes.

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

## SEE ALSO

SQL data types, ENUM





# Act: Task 02

## NAME

More tables!

## DIRECTORY

t02/

## SUBMIT

extended.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`
- `hero_id` - the hero's id
- `name` - a text field with the name of the hero's power
- `points` - int, the number of power points
- `type` - a choice of type of power (can be `attack` or `defense`)

One hero can have a few powers or not have them at all. Anyway, it's always a good practice to limit the size of your fields. Use primary keys and define the behavior on item delete in all your tables.

You must also create a `racess` table and link it with your `heroes` table. One hero can have only one race, but there can be a lot of heroes with the same race.

The `racess` table will have such fields:

- `id`
- `hero_id` - the hero's id
- `name` - a text field

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 and link it with `heroes`:

- `id`
- `hero_id` - the hero's id

- `name` - a text field

Finally, add some values to the tables you created as well.

`Powers` should contain at least `bloody fist (attack, 110 points)` and `iron shield (defense, 200 points)`.

`Races` should contain at least `Human` and `Kree` races.

`Teams` should contain at least two teams: `Avengers` and `Hydra`.

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](#)

# Act: Task 03

## NAME

Join it!

## DIRECTORY

```
t03/
```

## 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



# Act: Task 04

## NAME

Selection

## DIRECTORY

```
t04/
```

## 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 `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

# Act: Task 05

## NAME

Find out the truth

## DIRECTORY

`find.sql`

## SUBMIT

`find_me.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 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.

# Act: Task 06

## NAME

PDO

## DIRECTORY

t06/

## SUBMIT

DatabaseConnection.php

## ALLOWED

PDO

## LEGEND

Well done! You have found the traitor. It is not so easy to write all these scripts and handle all the errors. Now we have an Asgardian technology called PDO (powerful destructor of the Odin), and it can help us to improve all our work!

## DESCRIPTION

Create a class `DatabaseConnection` that implements `constructor`, `destructor`, and `getConnectionStatus` methods. In the `constructor` method you should establish a connection with your database from previous tasks. How to do it? It's easier than it sounds. Here are some examples to help you solve this problem:

- `constructor` should look like this:  

```
public function __construct($host, $port, $username, $password, $database){}
```

and create a connection to the database
- `destructor` should close the connection
- `getConnectionStatus` should return `true` if the connection is established, otherwise `false`
- the class should contain only one public variable - `$connection`

To create connection you should use MySQL driver, make sure you have installed one (read PDO documentation).



# Act: Task 07

## NAME

Database Model

## DIRECTORY

t07/

## SUBMIT

DatabaseConnection.php, Heroes.php, Model.php

## ALLOWED

PDO

## 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

Create an abstract class `Model` in file `Model.php`. The class should have the following methods:

- `setTable()` (should be called in the constructor and be protected)
- `setConnection()` (should be called in the constructor)
- `find($id)`
- `delete()`
- `save()`

It will describe basic behavior of all your future models. After that, create your first model `Heroes` in file `Heroes.php`. This class should contain all the fields from your `heroes` table and should be an extension of the `Model` class. You can create public fields only if you can set a custom value of them. Other fields should be able to get methods.

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

# Share

## PUBLISHING

Last but not least, the final stage of your work is to publish it. This 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 challenge, and get a better understanding of both 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

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
- [Telegraph](#) - create a post that you can easily share on Telegram
- [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.