Working with databases (MySql) - Practice Exercises

Question 1: Create a MySQL database by the name "**myDB**" and create a database user by the name "**myDBuser**" with a permissions to connect with the "**myDB**" database. Use the "mysql" module to create a connection with the newly created database. Display console message if the connection is successful or if it has an error.
Please find further instructions under the "Instructions for question 1" below.

Question 2: Here is a link to a document that contains the tables we need to create and convert the apple.com/iphones page into a dynamic page with a database. As you can see from the document, there are 5 tables that are needed (please scroll horizontally and vertically over the document to see all the 5 tables). Write a SQL query to create the apple.com tables inside of the "**myDB**" database you created above. Once you write the queries, use the "mysql" module to execute the queries on the database. Try both of these methods to initiate the execution of the queries:

- Include the execution code directly in the module to be executed as you run the app
- Use the Express module to receive requests. Configure your module in a way that it executes the queries when the "**/install**" URL is visited.

Please find further instructions under the "Instructions for question 2" below.

Question 3: Create an HTML file called, "**index.html**" with a form to populate the "products" table you created above.

- The form on the HTML page should send a POST request to a URL named "**/add-product**"
- Use Express to receive the POST request
- Use the body-parser module to parse the POST request sent to your Express server
- Write a SQL query to insert the data received from the HTML form into the "products" table

Please find further instructions under the "Instructions for question 3" below.

Hint on homework folder organization

- To be able to use a MySQL database, you should have MySQL software installed on your computer. Make sure that you have installed MAMP/WAMP or XAMPP as they come with MySql included in the package. If you have not downloaded and installed them yet, click only one of the links to download and install the MySQL database software:
    - For MAMP click here (Mac and Windows computers)
- Now, create a folder called "**MysqlPractice**" in your "**Evangadi**" folder
- In your "**MysqlPractice**" folder, create a module called "**app.js**"
- Make sure to always do the following when working on any Node projects
    - Open your project folder, the "**MysqlPractice**" folder in your case, in VSC and open your VSC terminal
    - Run "npm init" on your terminal to set up a new packages.json file
    - Install all contributed (third-party) node modules from NPM
    - Import all the modules your app depends on

Instructions for question 1

- To create your database and username, you will need to start your MAMP or WAMP application and to access your MySQL. Please make sure to remember the username and password you used when you create the database user.
- You will need a custom module (example, "**app.js"**) to write the code that connects your MySQL database to your MySQL database server
- Once you create your database, you will use Node to manipulate your database. However, for Node to be able to access your database, you need a MySQL driver, a module that creates a connection between Node and the MySQL database you created.

    - Make sure to install the "mysql" driver module from npm and import it in your "**app.js**" before writing a Node script to create a connection with the database you created.
    - You will need your database name, username and user password credentials to allow Node to access your database.
- Now, connect to the MySQL database server. It is a good practice to always display a message using console to make sure there is a successful connection to the MySQL database server from your Node.js application
- run your app (type the "node app.js" command)
- Make sure to always keep your server running after updating your app.js module with a new code
- It is highly recommended, especially for this week's practice exercise, that you code along while watching your class videos.

Instructions for question 2

- We need Express server to serve the table data upon a request with the URL of "**/install**". Therefore, before creating a server, make sure to install Express from NPM and import it into your "**app.js**" module.
- You will then write a SQL query to create tables from your Node app when the "**/install**" URL is visited.
    - Please watch out for syntax errors line by line. From past experiences, we have seen a lot of students struggling to create tables due to misspelling, missing a comma or semicolon.
- Make sure to execute the query you wrote to create your tables and show either an error message or a message showing creation of tables when the "**/install**" URL is visited.
- run your app. Make sure to always keep your server running
- Make sure your MySQL database is also on (meaning, your MAMP/WAMP needs to be started)

- Now run your localhost in your browser and visit the /install" URL. Make sure to also include the port number you provided for your Express server.

Instructions for question 3

- This question is basically asking you to write a query that adds all the information entered into the HTML form into your products table (in your database) when you run your localhost in your browser using the "**/addiphones**" URL
- You will need to include the correct URL address in your form's "action" attribute. Make sure this URL address includes the correct port number that matches the port number you used when you created your Express server
- Install the body-parser module from NPM and import it into your app. body-parser is a module that captures all the information entered in an HTML form and parses (separates them into easily processed units) them in an object form. Because the data from HTML forms are now in an object form, it will become easy to access them.
- Install the CORS module from NPM and import it into your app. Web browsers by default don't allow cross-origin resource sharing/CORS. CORS is a built-in browser security feature that protects unknown websites from accessing/using your website's resources. You need the CORS module installed to allow your Express server to have access to the form URL.
- Now you will write a SQL query to insert the data received from the HTML form into the "**products**" table
- Make sure to execute the query you wrote to insert the data. Please show either an error message or a message showing the insertion of form data into your tables when the "**/addiphones**" URL is visited.
- Open your HTML file in your browser, fill out the form and submit it.
- Now, go to your database. If there is no error, you will see the information from the HTML form inserted into the respective tables in your database.

Happy coding 😀