



MySQL Week 4 Exercises

Background

We have been developing a menu-driven application that demonstrates how to perform CRUD (Create, Read, Update, Read) operations on a DIY project database. Thus far, we have learned how to create a connection to a MySQL database and how to insert records into a table. In this section, we will apply our knowledge of querying data to list all projects without project details, and to list a single project with all details.

Objectives

In these exercises, you will expand the menu application to list all projects (name and ID). Then, you will write code to select a project to edit. This will involve returning a selected project along with all project details. This will further our pursuit of implementing CRUD operations on database tables.

In these exercises, you will:

- Hone your SQL query skills by writing SQL statements to fetch a `List of Project` records.
- Learn how to perform multiple queries in a single transaction.
- Write an inner join to fetch `category` rows related to a `project` row.
- Use an `Optional` to either return a `project` record or to throw a custom `Exception`.
- Practice writing Lambda expressions both to list the projects and to throw a custom `Exception` from an `Optional`.

Important

In the exercises below, you will see this icon: .

This means to make sure that you include this functionality in your video showcase.

Also important: you should take the variable names and method names as suggestions. They're good suggestions, but if you want to deviate from them, feel free to do so. However, don't go crazy and change `listProjects()` to `emptyMyBankAccountByBuyingAJeep()`. You should follow Java best practices. Method names should describe what the method does in the interest of self-documentation.



MySQL Week 4 Exercises



Instructions

URL to GitHub Repository: <https://github.com/blg12/week8assignment.git>

URL to Public Link of your Video: <https://youtu.be/qIBFrioC1pM>

Instructions :

1. Follow the [Exercises](#) below to complete this assignment.

- In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed.
- Create a new repository on GitHub for this week's assignment and push your completed code to this dedicated repo, including your entire Maven Project Directory (e.g., mysql-java) and any .sql files that you create. In addition, screenshot your ERD and push the screenshot to your GitHub repo.
- Include the functionality into your Video when you see: 
- Create a video showcasing your work:
 - In this video: record and present your project verbally while showing the results of the working project. Don't forget to include the requested functionality, indicated by: 
 - Easy way to Create a video: Start a meeting in Zoom, share your screen, open Eclipse with the code and your Console window, start recording & record yourself describing and running the program showing the results.
 - Your video should be a maximum of 5 minutes.
 - Upload your video with a public link.
 - Easy way to Create a Public Video Link: Upload your video recording to YouTube with a public link.

2. In addition, please include the following in your Coding Assignment Document:

- The URL for this week's GitHub repository.
- The URL of the public link of your video.

3. Save the Coding Assignment Document as a .pdf and do the following:

- Push the .pdf to the GitHub repo for this week.
 - Upload the .pdf to the LMS in your Coding Assignment Submission.
-



MySQL Week 4 Exercises

Exercises

Complete these exercises as directed. If you get hopelessly stuck, please see the "Solutions" section below.

In these exercises, you will often be told to call a method prior to creating it. This is a good approach. You set up the return type by assigning it to a variable and setting up the parameters. Then, Eclipse can correctly create the method.

List projects

In this section, you will add code to return and print a list of projects. Several application methods will call this method to let the user select a project from a list. To get the list of projects, you will add the menu option and method in the menu class, then you will add a method in the service class. The DAO class will perform the actual work of fetching the list using JDBC method calls.

Modifications to menu app

In this section you will add another option to the list of available options. Then you will add another case to the `switch` method along with the method to call the service to retrieve the list of projects.

In this section, you will be working in `ProjectsApp.java`.

1. Add this line to the list of operations at the top of `ProjectsApp.java`: `"2) List projects"`.
2. Add case 2 to the switch statement in `processUserSelection()`. In the case, call method `listProjects()`. Don't forget to add the `break` statement.
3. Have Eclipse create the method `listProjects()`. It should take no parameters and should return nothing. In the method:
 - a. Create a variable to hold a `List of Projects` named `projects`. Assign the variable the results of a method call to `projectService.fetchAllProjects()`.
 - b. Print `"\nProjects: "` (without quotes) to the console.
 - c. For each `Project`, print the ID and name separated by `": "`. Indent each line with a couple of spaces.
 - d. At this point, the method should look like this:

```
private void listProjects() {  
    List<Project> projects = projectService.fetchAllProjects();  
  
    System.out.println("\nProjects:");  
  
    projects.forEach(project -> System.out  
        .println("    " + project.getProjectId()  
            + ": " + project.getProjectName()));  
}
```