Homework 4:
Development of a web application with Streamlit and MySQL

Goal

Create a web application in Python (Streamlit) that can interact with a MySQL database to perform queries based on user interactions.

Preliminary phase

- Start MySQL via Docker or XAMPP (see *Live Coding* and *Laboratory 6*)
- Import the database
- Create and start the Streamlit project (new project or from the *repository base* branch)
- Verify the connection to the db through the credentials
- ***Tip***: Use *st.write()* to investigate your results

Database Description

The database is named GYM and it is about the activities in a gym. It is described by the following logical schema (primary keys are underlined, foreign keys are in italic, and optional attributed are denoted with *):

TRAINER (SSN, Name, Surname, DateOfBirth, Email, PhoneNo*)

COURSE (CId, Name, Type, Level)

SCHEDULE (SSN, Day, StartTime, Duration, CId, GymRoom)

Use the phpMyAdmin interface to import the *createDB.sql* script found on the course's website and verify that it was successful. Use the interface to explore the database, available tables, and saved data.

Exercises

Create a multi-page application to display the main information contained in the database and allow the user to add new information. In particular:

1. Create a custom *Homepage* that leverages the basic markdown syntax (or Streamlit elements) to introduce the lab, the goal, and the student.

2. Create a page for viewing and filtering available courses. The page must be supported by two *metrics* to show the number of courses and distinct types available. The input widgets should be created in a way that the suggested options are based on the information already present in the database. The user must be able to view course information by filtering by multiple categories (i.e., *Type*) and must be able to specify the level range he or she is interested in.

In a separate expander, visualize lessons for the selected courses. In case of empty results, a related error/warning must be printed.

3. Create a page to view available instructors. The user must have the ability to filter by typing the instructor's last name and using a date range to choose by date of birth (***hint:*** use datetime.date() to set the date_input and pass the date as a string in the query). The view should not be an overall table, but divided item by item (create a dataframe and use *df.iterrows to print one row at a time, see Lab 6*). If the results are empty, an associated message should be displayed.

4. Create a page to display graphs about scheduled lessons. The page must contain two graphs: a Bar Chart that shows the number of lessons for each time slot and a Line Chart that shows the number of lessons scheduled according to the day of the week.

5. Create a page for the insertion of new courses through a suitable form. Use an input form that requires all the data necessary for the insertion of a new course in the database (CId, Name, Type, Level). Your application must verify that all fields are valued and that the value of the Level attribute is an integer between 1 and 4 (***hint:*** use *number_input*). In case of missing data, duplicate key or other errors, the application should generate an error message. If, on the other hand, the data entered is correct and the insertion operation is successful, a message of correct insertion must be displayed.

6. Create a form for inserting a new weekly lesson in the SCHEDULE table. The form must allow you to enter all the necessary fields (SSN, Day, StartTime, Duration, CId, GymRoom) related to the programming of a new lesson. The selection of the instructor must take place through a drop-down menu containing the SSN of the possible instructors generated by the content of the database table. Similarly, the selection of the course must also take place through a drop-down menu populated by the database. The other fields are fields populated manually by the user, using the most suitable widgets (e.g., slider for StartTime and Duration) or textual ones. The application verifies that the user does not try to include in the program lessons that last more than 60 minutes and that the day indicated is a day between Monday and Friday. The insertion of a new lesson in the program must be allowed and executed if and only if no other lessons are scheduled for the same course on the same day of the week (***hint:*** use input values to query and verify that there are no records). If the insertion request respects the constraints indicated and the insertion ends successfully, a message of correct insertion must be displayed, otherwise an error message must be notified (the error message must indicate the type of problem that caused the error).

All pages should be customized with text elements (using markdown or Streamlit's pre-set widgets) so that headings, subheadings, and paragraphs highlight what is being represented. In addition to generating the correct queries, to make the vision and the interface more intuitive and organized, the main layout elements must be used: expanders, columns, tabs.