# Final Project

## Step 3: 12 points

### *A. Create Database and Tables:*

Translate the relational schema you developed in Step 2 into an actual database. Use MySQL DBMS and the SQL language to create your database and the corresponding tables. Place the constraints (e.g., default values, foreign keys) you find appropriate - **Important notes:**

1- Your database, as discussed in Step2 and because of the limited time, should have 4 entity-sets and 4 relationship-sets - these are the minimum requirements. You, as the designer and developer, may decide that your database requires more entity-sets and relationship-sets.
2- In step2, you submitted an ER diagram and corresponding relational schema – as the design process is iterative, you can add/remove/update the schema to fulfill the implementation phase.
3- In step2, you submitted 8 queries - as we discussed more SQL operators, you can update/modify the queries.

### *B. Prepare and Insert Records:*

Create a simple "***Interactive***" Java application (a terminal-based, GUI-based, or web-based) that connects to the created database to insert tuples and run queries. Prepare the data tuples/rows to be inserted in your database as follows:

**1.** You can ***(Recommended option)*** populate your database with real data. Search online for data (usually in .csv, .txt, or .excel format) and use them, ***through your Java app***, to build the insert queries to fill the different tables. You may not find a data set that fully cover all attributes of all tables, such data is expected to cover only the important attributes.

**2.** If you don't have real data to use or you couldn't find a data set that fully cover all attributes of all tables (as discussed above), your Java app should generate random data for the uncovered attributes - The generated data should make sense as much as possible (e.g., don't generate negative ages for employees or students with GPA > 4.0). Take the following ***Football Games*** table from a football database as an example:

| GameID | Team Name | Opponent Team | Match location | Match date | Result |
|---|---|---|---|---|---|

To prepare data tuples/rows for this table, you can search on the internet and find real data that covers the Team Name, Opponent Team, and Match Date attributes. In this case, your Java application should then randomly generate data for rest of attributes (GameID, Match location, and Result). The GameID can be an integer that autoincrements every time you insert a record, the Result attribute can be a random integer from 0 to 10, the Match location can be a random choice from a list like (Paris, London, Cairo, Moscow, Dubai).

**3.** If finding real data from external online resources or generating meaningful random data use Java will be a difficult problem, you may use online tools (e.g., https://www.mockaroo.com/) to generate random data into files, then ***use Java*** to read these files, line by line, and build the corresponding insert queries.

- From the tuples you prepared from the previous point, your Java app can now populate (via insert SQL command) the database with these tuples. Each main table in your design is expected (***if possible***) to hold at least 500 tuples.

### *C. Run Queries and Print Results:*

Your Java app then handles user interactions. Display a nice name for your application along with a short welcome message descripting the goal of your database. Your application should display the different queries you prepared in step2 (query number and short description) and ask for user's input (user enters 8 and clicks enter, your application executes the 8th query). In order for the application to display the result, the user should select one of the following options:

- *Option1 - "Display the results here!"* In this case, your application will display the results the way you find appropriate on terminal.
- *Option2 – "Save the results to a file!"* In this case, your application will save the results to a file on the desktop, use the file name you find appropriate that makes each file unique (e.g., the query number + the time).

After the application executes a query, the app views the menu again for the user to select another query. You should allow the user to exit your application. The user should not write SQL query through your terminal to be executed.

**Submission:**

**1.** Directly to the folder titled Step 3 under the D2L Assignments tab, submit the .java file(s) you developed and one PDF structured as follows:
   1. On the first page, place you name and a short description of your idea (from Step1)
   2. On the second page, mention the type of data you used:
      i. did you use real data (yes or no), which tables and attributes are covered with such real data set.
      ii. did you use java to generate random data, which tables and attributes are covered with such random data.
      iii. did you use online tools to generate random data, which tables and attributes are covered with such data set.
   3. On the third page, one screenshot of your running application to show if you developed a terminal, GUI, or web application along with the menu of queries for user interaction. **Do not take screenshots with a camera.**
   4. On the following page(s), for each of the 8 queries, take screenshot(s) for the SQL query you developed as well as the output. **Do not take screenshots with a camera.**
   5. On the last page, list the edits/modifications/adjustments you made (if any) to the ER, Relational Schemas, or Queries you submitted in step2.

**2.** This is an **individual assignment** -- Cheating/plagiarism will be checked and will receive zero.

**3.** The assignment is due 10:00pm – **December 6th**. You can submit your assignment, within 24 hours after this due date, to be graded out of 75% of the assignment's grade. After this grace period your late submission will not be accepted.