LaSalle College
Montréal

Project 4: 10%

Course Identification

Name of program – Code:	
Course title:	**DATABASE II**
Course number:	420-BD2-AS
Group:	07438
Teacher’s name:	Jean-François Parent
Duration:	Extended
Semester:	Fall 2021

Student Identification

Name: _____ Student number: _____

Date: _____ Result: _____

☐ I declare that this is an original work, and that I credited all content sources of which I am not the author (online and printed, images, graphics, films, etc.), in the required quotation and citation style for this work.

Standard of the Evaluated Competency
Statement of the evaluated competency – Code
Use a database management system – 00Q7
Evaluated elements of the competency 00Q7
1. Create the database. *2. Formulate read requests, insertion requests, modification requests and deletion requests.* *3. Ensure data confidentiality and consistency.* *4. Program automated data processing operations.* *5. Save and restore the database.*

Instructions

- Create a database (schema) according to the requirements.
- It is the teacher's responsibility to identify language errors. If such errors are found, teachers may apply a penalty of up to 20% of the grade (IPEL – Article 5.7).
- Plagiarism, attempts at plagiarism or complicity in plagiarism during a summative evaluation results in a mark of zero (0). In the case of recidivism, in the same course or in another course, the student will be given a grade of '0' for the course in question. (IPEL – Article 5.16)
- Submit dates on Omnivox must be respected.

- Please consult the correction grid at the end of document for more details.

TOTAL: 100 POINTS

Because this work must be done at home, if we think that the answers are not yours, the department reserves the right to complete your evaluation with a virtual meeting to verify that you attained the required competencies.

Projet 4

Directives

Note: This is **NOT** a team work

1. For objects names, only use the recommended characters (A-Z (objects names must begin by a letter), 0-9 and the underscore (_).

2. Each field must have the correct data type. The size must be big enough to store all required data, but not too big to use too much space on the disk.

3. Unless specified otherwise, each field must contain data and must not accept NULL values and/or empty strings ('').

4. Specify a default value where you can do it.

5. Specify constraints where you can do it to prevent users to enter unwanted data.

6. Each table must have a primary key which identifies each row with a unique value (GUID).

7. You must create indexes for each field used in a search, for each foreign key, and for each column which must be unique.

8. Each SQL query must use good indentation (for example, no query on a single line).

9. Write comments to describe what the SQL code does.

10. For each table and object, you must write the revision history. It must contains the date, your name, your student number and the modifications made. For tables, enter this information in the comments of the primary key. For the other objects (stored procedures, functions, etc.) you must inscribe these informations in the comments. For example:

Table:

```
2023-10-29    Bill Torvalds (1244556)    Created the customers table.
2023-10-31    Bill Torvalds (1244556)    Added the firstname column.
```

Stored procedure:

```
--Revision history:
--2023-11-01    Bill Torvalds (1244556)    Created the procedure customers_select.
--2023-11-03    Bill Torvalds (1244556)    Fixed the ORDER BY bug.
```

11. All the newly created tables must include the 2 fields for the date and time of creation/modification (like in project 1).

Projet description

This project must re-use the database created in previous project. Marks will be awarded only for the new elements. It is thus not mandatory to fix the errors made in previous projects unless these elements are required in project 3 (in other words, the new SQL code must not crash).

Improve the database created in the previous projects by implementing the following requirements:

Cursor (15pt)

- Create a **stored procedure** which receives a student ID and which will read all the results (reports) for a student, sorted by date. For each exam display the cummulative average which changed during the semester. For example:

Date	**Firstname**	**Lastname**	**Course**	**Teacher**	**Result**	**Cumulative average**
2023-09-15	Juliana	Tryhard	Math	Anne Day	40%	40%
2023-09-15	Juliana	Tryhard	English	Ken Smith	70%	55%
2023-09-15	Juliana	Tryhard	History	Simon Perry	85%	65%

The course name comes from the courses table so you need to add a foreign key in the reports table.

Note that the column **headers** (in bold) are not required in the stored procedure but are used to show you which fields to display (so you can use your existing column names).

The procedure should a display on the screen the number of completed courses (ex: The student completed 3 courses)

Regular expressions (20pt)

- Create a trigger for the INSERT of the courses table which will perform all the following verifications with **regular expressions** (case-insentitive):
  - If a course name contains exactly the text 'Math', replace the name of the course with 'Mathematics' in the table.
  - If a course begins with 'eng', for example 'eng advanced', replace eng with English (thus the new course name will be 'English advanced'.
  - If a course contains the text 'history' or 'geography' or 'science' save the name of the course in uppercase.
  - If a course contains the text 'self defense' or 'self-defense' write the first letter in uppercase (Self defense or Self-defense).

- If a course contains 5 digits in a row save the course name in lowercase.

- If a course contains the text 'Cancel' or 'Cancelled', block the INSERT by generating an error.

Triggers (11pt)

- Create a trigger for the DELETE of the reports. Block the deletion by generating an error and save this event in the modifications table because no one is supposed to delete reports marks. Use 'Deletion' for the name of the modified field.

- Create a trigger for the INSERT of the modifications. If the field name contains the word ‘Deletion’ save it in uppercase ('DELETION') to help spot this change in the table.

Functions and procedures (38pt)

Create **stored procedures** and **functions** to perform the following tasks:

To calculate the averages, do not create a new field in the reports/students tables but use aggregate functions.

- Create a **function** to display on the screen (DBMS_OUTPUT) all the students (lastname, firstname, student number) who have an average of 80% or more (sort by student number).

- Create a **function** which returns (RETURN) the student ID of the student with the highest average. This function must also return (with a output (OUT) parameter) the student ID of the student with the lowest average.

- Create a **function** which receives a teacher ID. This function must do a SELECT (pipelined) to display the name of all students of this teacher (firstname, lastname, student number, sorted by lastname and firstname).

- Create a **procedure** which displays on the screen (DBMS_OUTPUT) all the teachers who gave 10 courses or more. Use an aggregate function to count the rows in the reports table.

- Create a **procedure** which receives a teacher ID and which returns, with two output (OUT) parameters, the highest **result** in the table with the name of the **course**. Of course you can (as usual) use any extra variable to achieve the desired result.

Permissions (15pt)

You must create a user with your own **firstname** (utilisez votre prénom) and a role named **website** to grant permissions to use (but not modify) your database. Create a script named **permissions_*xxxxxxxx*.sql** (replace xxxxxxxx with your own student number) to perform the following tasks:

- Connect with a user who can create roles.
- Create a Oracle user that will be used by the website.
- Create a role named website.
- Add the new user to the new role.
- GRANT to the website **role** (not to the user) the permissions to read and write in the database but only with the created procedures, functions and types. Do not give permissions directly on the tables because it would be a big security hole in the database security.
- Connect with the newly created user.
- Once connected as this user, call at least one object (procedure, function, etc.) to make sure the permissions were granted properly.

Scripts (1pt)

When your database is ready, export it into a text file (.sql). The filename must contain at least your lastname and the date when the backup was made.

This exported file only creates the objects and not the database. You must thus add manually the SQL code to create and use a database (schema) which has the same name than your database. Use the password “123”. The script must create all the objects (database/schema, tables, procedures, etc.) so test it properly.

Your script must NOT delete any existing database/schema if it already contains object(s). Your script must be error-free.

Submit the two files (DB creation + permissions script) on Omnivox.