Lab 3 – SQL Data Definition & Manipulation

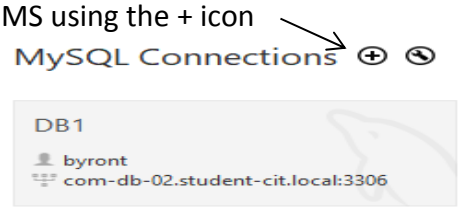
Learning outcome(s): The student should be able to

- Perform data definition: i.e define tables in SQL with properties including Keys (Primary, Composite, Candidate, Foreign), required fields(not null)
- Perform data manipulation i.e. SQL Select, Insert; Update ; Delete

Requirements: to be able to use a client utility, basic SELECT and have a database created.

Reminder: To Set up Connection to SQL Server

Find and run the MySQL Workbench 6.3 CE (GUI) client program to connect to a database server. You may find the connection is still in place from last week's lab, however you may need to set up a 'New Connection' to the DBMS using the + icon



To set up the connection again,

- Connection Name – you can name the connection any name, e.g. DB Lab, Your Firstname, etc.
- Identify the server (hostname): e.g. 157.190.43.7 or com-db-02.student-cit.local
- Security: your user name (Student ID)
- Click on Test Connection button, and enter the password – Spring2018
- Click OK, and then select the new connection added to your MySQL Connections as shown above

Assignment 1: Create a Database and Table

Part1: Create Database

- Execute the command to create a database and name it with your student Id: e.g.
  `Create database R00012345;` *Replace the sample StudentID with your own.*
  - See if there are any tables in that database, it should be empty: Call the lecturer to view.

Part 2: Create Tables

Overview:

You are required to create a set of tables. Once you created the tables, you must then insert data, update data and delete specific elements before finally dropping the actual tables.

SQL is free form and case insensitive for commands, so you can type SQL code in any way you wish. However, reserved words in the SQL language are **usually indicated** by putting them in capitals. Note, string (called varchar) values (literals) are case sensitive i.e. t <> T

Table 1 – Student:

```
CREATE TABLE     Student (
    Sid     varchar(10)     NOT NULL,
    Name        varchar(20),
    Login       varchar(20)     NOT NULL UNIQUE,
    Age     int,
    PRIMARY KEY (sid));
```

- Input a row into the table using the ***INSERT*** command.

```
INSERT INTO  student VALUES( 50000, 'Dave Murphy', 'Dave@cit.ie', 19 );
```

- **Use the Insert statement** for the following rows. Your final table should look like the following:

Sid	Name	Login	Age
50000	Dave Murphy	Dave@cit.ie	19
53688	Mary Murphy	Mary@cit.ie	20
53831	Joe Smith	Joe@cit.ie	21

- Display the table: you should know how to do this from Lab1.
- Insert the following data:

53688, David Moore, Dave@cit.ie, 22

  - Think about string (varchar) datatype literals and handle them correctly.
  - Does it insert correctly? Explain.

_____
_____
_____
_____

Table 2 – Enrolled:

You must now create a table named ENROLLED, which stores the information on modules and the grades that students have achieved for them.

```
CREATE TABLE enrolled (
mid     varchar(10)     NOT NULL,
mname       varchar(15)     NOT NULL,
    sid     varchar(10)     NOT NULL,
grade       char(1),
PRIMARY KEY (mid,sid),
FOREIGN KEY (sid) REFERENCES Student (sid) ON DELETE RESTRICT ON UPDATE CASCADE);
```

Note the foreign key controls on update and delete operations in the referenced table.

- Input the following data into the table using the SQL ***INSERT*** command.

```
'CS201','Database', 53831, 'A');
```

- Do the same for the remaining four rows in this table. Your final table should look like:

Mid	Mname	Sid	Grade
CS201	Database	53831	A
CS202	Graphics	53831	B
CS202	Graphics	53688	A

- Now try insert the following record into the ***ENROLLED*** table:

```
CS202, Database, 54000, A
```

  - Is the insertion carried out correctly?
- Examine the existing data in the tables.
- Display the table: type and run the following: **SELECT * FROM enrolled;**
- Display the student table: type and run the following: **SELECT * FROM student;**

Data integrity means data correctness. What would be wrong with the integrity of the data if the above Insert was allowed?

_____

_____

_____

- Go back and examine the code in the create table statements;
- **Identify what section of the statement might be responsible for the not allowing the insert?**

The SQL UPDATE statement allows us to modify (change) existing data in a table.

- To change all grades in the ENROLLED table to 'E', we do the following: Note without a WHERE clause this will update all rows.

```
UPDATE enrolled SET grade = 'E';
```

Note: Be careful, this multi-row operation may be controlled by the Workbench client *(Edit-Preferences-SQL Editor- De Select Safe mode).*

- Try to **Delete the student with student number: 53831**

```
DELETE FROM student WHERE sid = 53831;
```

- Does it work? **Do you understand why deletion is possible / not possible?**

Note: to delete a table completely you must DROP TABLE *table-name* e.g. drop table student. (Don't do this operation unless you really need to).

Assignment 2: Practice Exercises

1. In the create table statements above find how a composite key is defined.
2. Find the candidate key definition (i.e. an alternative to Primary).
3. We now want to define a composite candidate key on table student for name + age.
    a. What might the SQL syntax for this look like? *(You should have enough by using the answer to 1 + 2 to suggest the likely syntax).*

4. Design and implement a table to store data on an Employee.
    a. Input 2 rows.