
CO1019 - Databases and Web Technologies

MySQL Exercises

In this exercise, you will learn to use MySQL database (<http://www.mysql.com/>) and how to create a simple relational database. This exercise should be completed within **2 lab sessions**. On the first lab you are asked to work on pages 1 to 5. On the second lab session (the week after), you are asked to work on pages 6 to 8. We advise that you spend some of your own time preparing things before the lab-time and completing all tasks afterwards. Your completed exercise is to be presented to the lab tutors. The exercise will generally cover connecting to your MySQL database, and practicing the basic SQL commands as taught in the theory lectures.

1 - Connecting via MySQL Workbench

Your MySQL account should already have been set-up. So, before starting the exercise, please make sure that you have your user-id and password, and successfully able to log on to the localhost MySQL server in your Linux Mint machine. You can also do this using Microsoft Windows.

From the main menu, open *MySQL Workbench* as shown in the screenshot on the right.



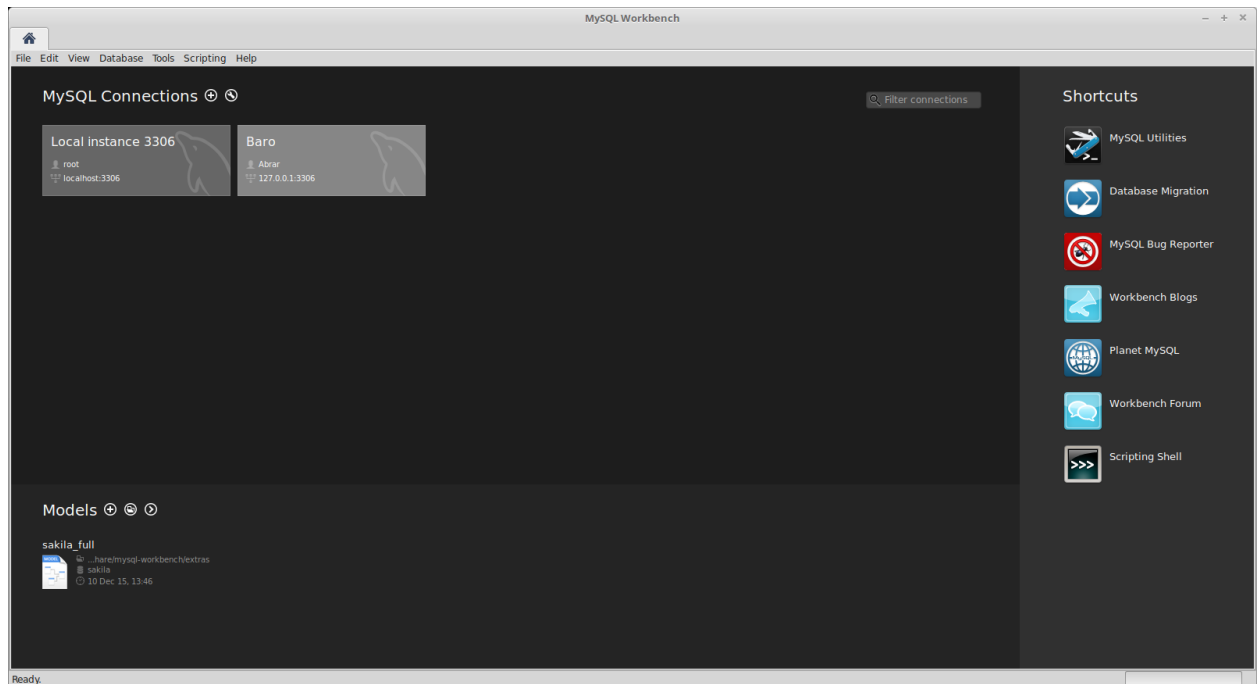
Select your account, NOT the root, and enter your password (if not saved)

The Host is: `mysql.mcscw3.le.ac.uk`

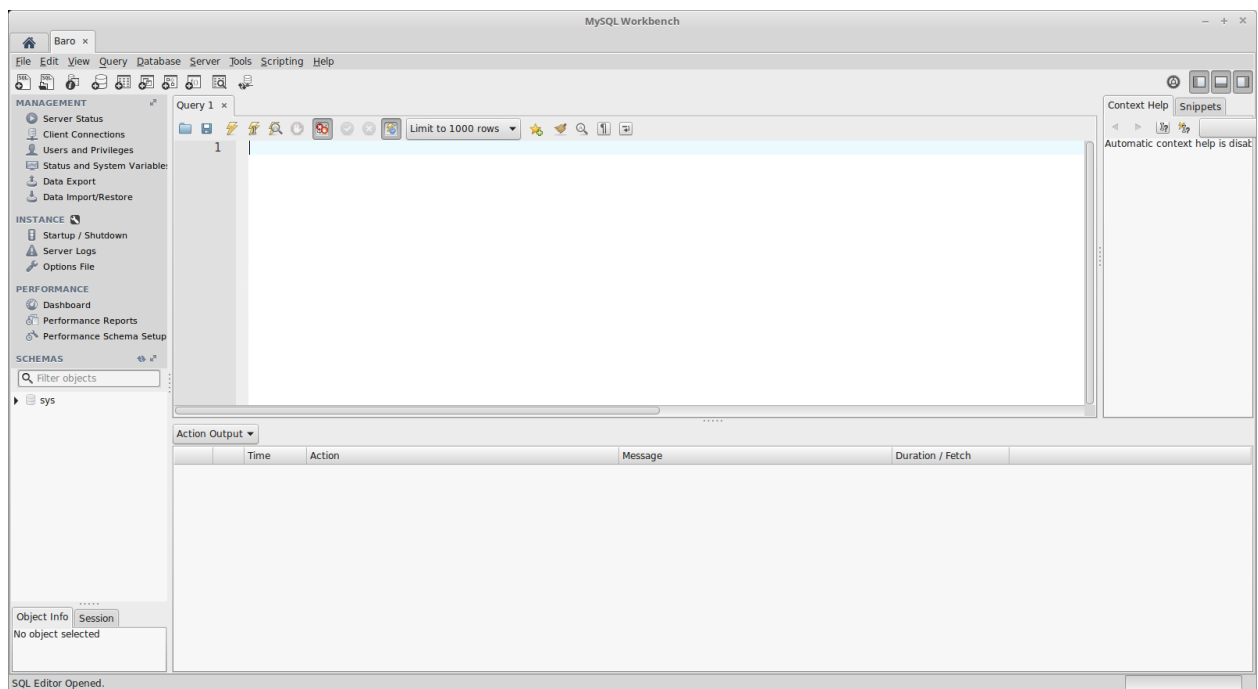
Username: `<username>`

Password: is saved under: `~/.my.cnf` in your linux account

And the DB (schema) name is: `<username>`

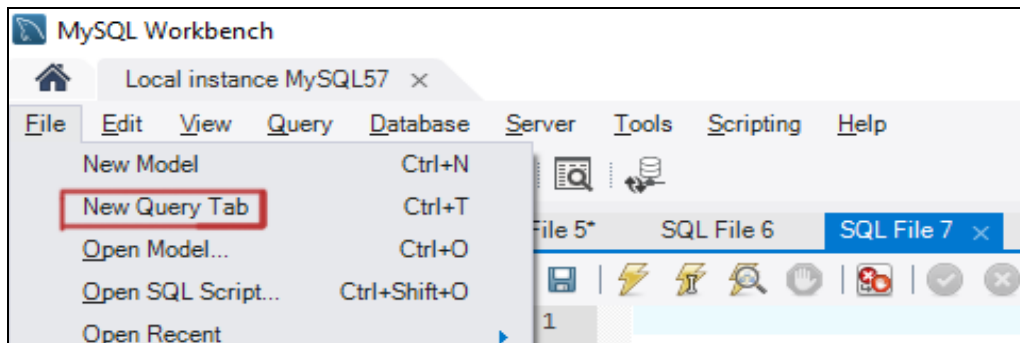


If you successfully logged in, your interface should look like the following screenshot



2 - Creating database schema and simple queries on one table

Start MySQL Workbench, click *File* from the top menu → *New Query Tab* (or Ctrl + T):

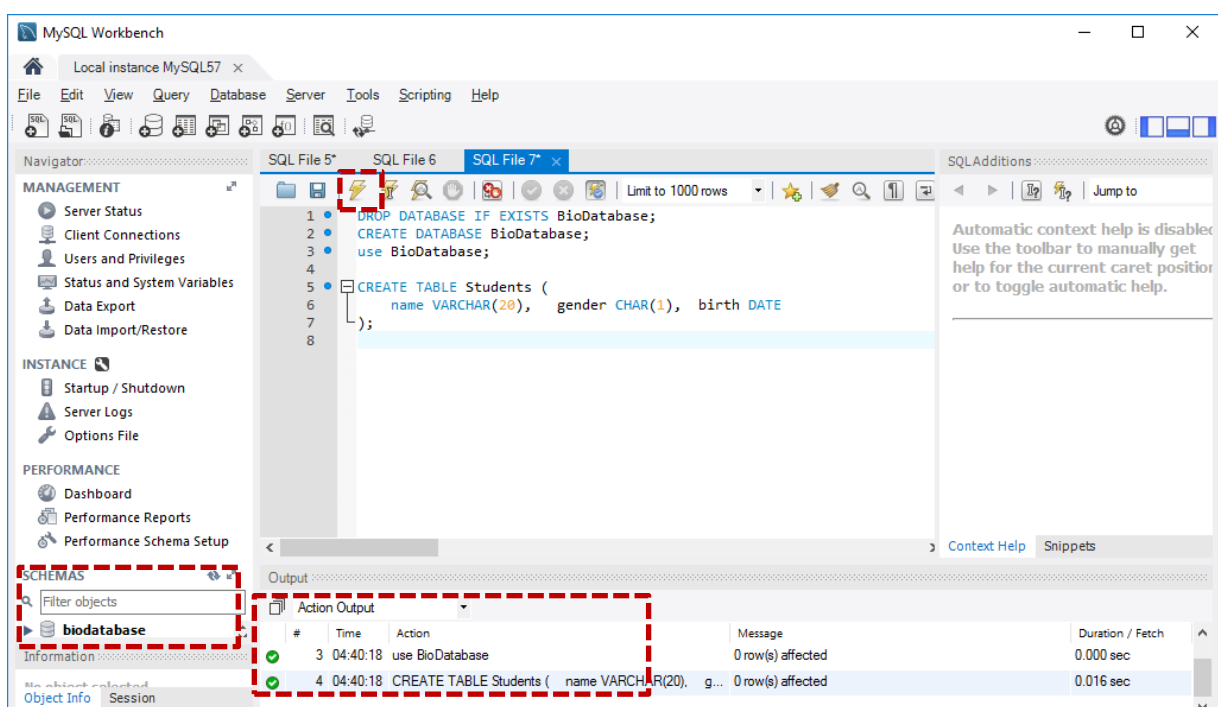


Copy/past the following statement into the generated new SQL file box, let's call it (*SQL File-1*) , and then execute it by (Ctrl + Enter) or from *Query* tap → *Execute Current Statement*.

```
DROP DATABASE IF EXISTS BioDatabase;
CREATE DATABASE BioDatabase;
use BioDatabase;

CREATE TABLE Students (
    name VARCHAR(20), gender CHAR(1), birth DATE
);
```

You should see something like the following screenshot.



To confirm the creation of *Students* table, try to execute the following statement, ideally in another *Query Tap* and check the results.

```
show tables;
```

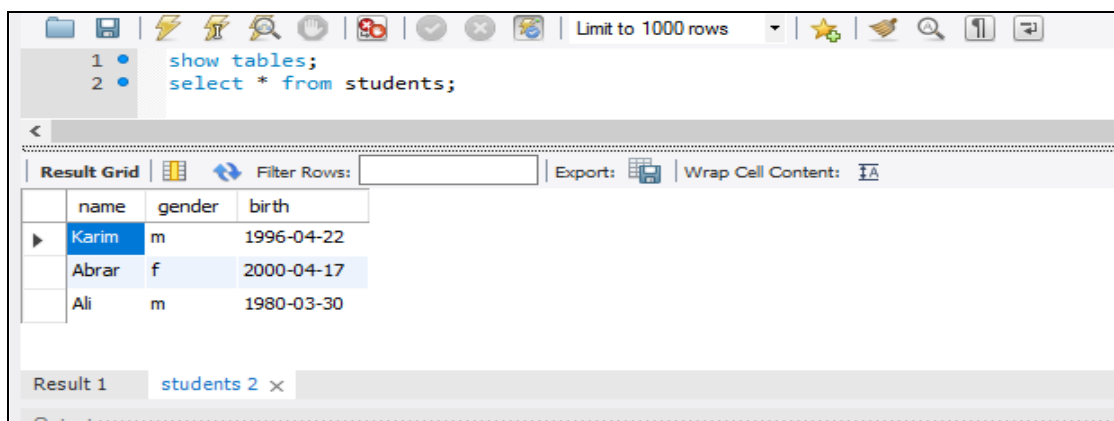
Modify *SQL File 1* that was created previously by appending the following statements:

```
DROP DATABASE IF EXISTS BioDatabase;  
CREATE DATABASE BioDatabase;  
use BioDatabase;  
  
CREATE TABLE Students (  
    name VARCHAR(20), gender CHAR(1), birth DATE  
);  
  
Insert into Students values  
('Karim','m','1996-04-22'),  
('Abrar','f','2000-04-17'),  
('Ali','m','1980-03-30');
```

To display inserted data into *Students* table, execute the following SELECT statement in also a different *Query Tap* and then check the results.

```
Select * from Students;
```

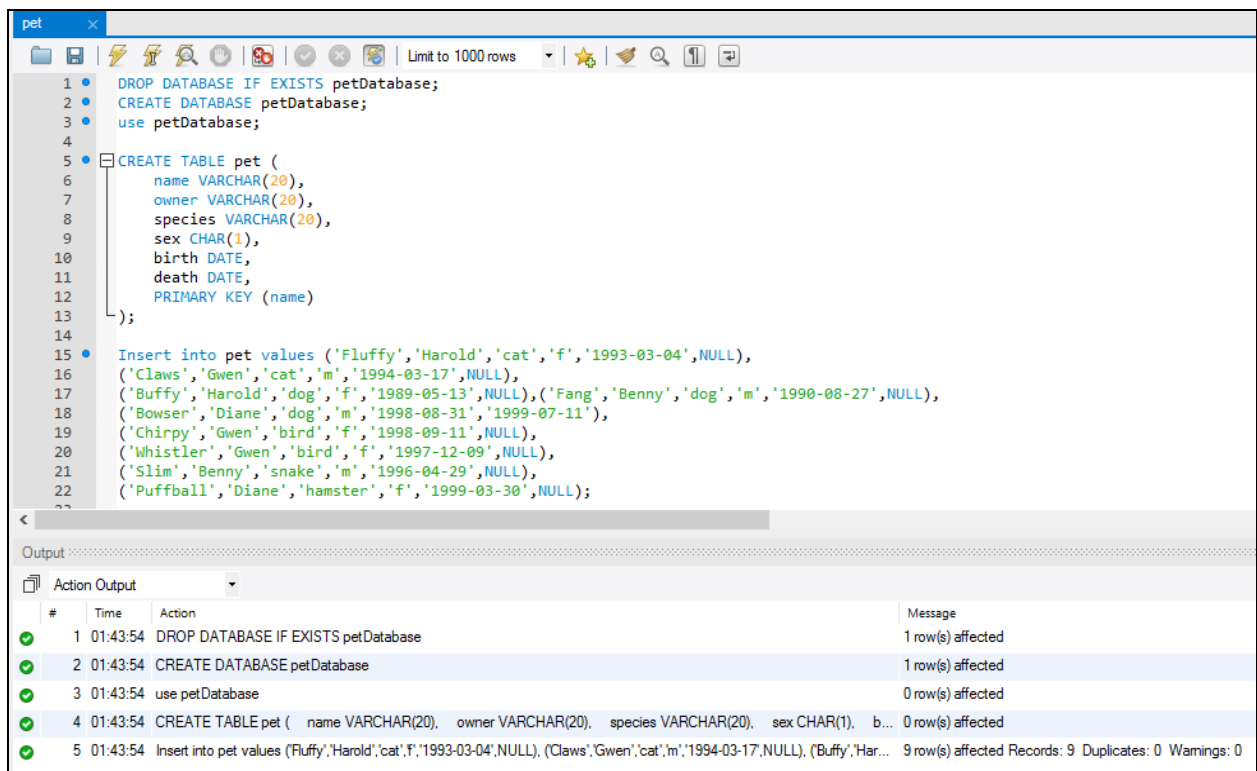
To save *SQL File 1* script, select *File* → *Save As* and name it, e.g., *MyFirst.sql*



3 - Practicing different queries upon the data.

There is a file called *pet.sql* you should download and place in a directory. Open MySQL Workbench and log in as shown previously. From the top menu, click on *File* → *Open SQL Script*, and then navigate to select *pet.sql* that you downloaded.

Execute *pet.sql* script. In doing so it will create a table called *pet* inside your database, see screenshot below.



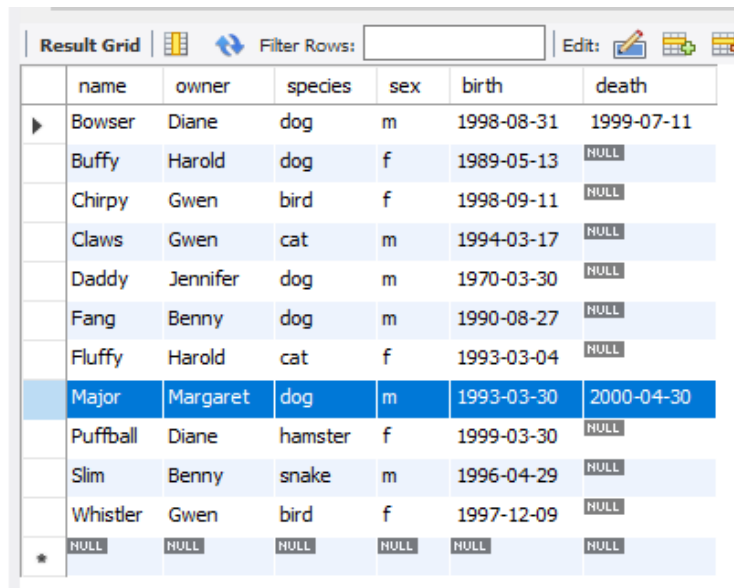
Try to add an extra two rows to the table. Ensure your database is selected and in the query box enter the following statements to add an extra two rows to your *pet* table:

```

INSERT INTO pet VALUES
('Daddy','Jennifer','dog','m','1970-03-30',NULL),
('Major','Margaret','cat','m','1993-03-30','2000-04-30');

```

Now you should try to use the *UPDATE* and *DELETE* commands. For example, modify the *pet* table so that *Major* is a *dog*. Query the table to see the results of your update, which should be similar to the following screenshot.



	name	owner	species	sex	birth	death
►	Bowser	Diane	dog	m	1998-08-31	1999-07-11
	Buffy	Harold	dog	f	1989-05-13	NULL
	Chirpy	Gwen	bird	f	1998-09-11	NULL
	Claws	Gwen	cat	m	1994-03-17	NULL
	Daddy	Jennifer	dog	m	1970-03-30	NULL
	Fang	Benny	dog	m	1990-08-27	NULL
	Fluffy	Harold	cat	f	1993-03-04	NULL
	Major	Margaret	dog	m	1993-03-30	2000-04-30
	Puffball	Diane	hamster	f	1999-03-30	NULL
	Slim	Benny	snake	m	1996-04-29	NULL
	Whistler	Gwen	bird	f	1997-12-09	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

You are now going to write some statements to show some specific data of your *pet* table:

1. Create a *VIEW* which shows the numbers of pets that have died. Use an appropriate column label (e.g. '*deaths*').
2. Show all details of live pets that are born in 1993 or 1998 and are female.
3. Show name, birth date, and species of pets sorted by name (descending order).
4. Count how many pets there are of each gender. The result of the query should be displayed as a two-column table, the columns being gender and number.

4 - Inter-table relationships with advance commands

There is a file called *relationships.sql* you should download and place in a directory. Open MySQL Workbench and log in as shown previously. From the top menu, click on *File → Open SQL Script*, and then navigate to select *relationships.sql* that you downloaded.

```
DROP TABLE IF EXISTS User;
DROP TABLE IF EXISTS Proteins;
DROP TABLE IF EXISTS Alloc;

CREATE TABLE User (
  UID INT,
  FName VARCHAR(20),
  LName VARCHAR(20),
  Password VARCHAR(20),
  PRIMARY KEY (UID)
);

CREATE TABLE Proteins (
  PID INT,
  FullName VARCHAR(20),
  PRIMARY KEY (PID)
);

CREATE TABLE Alloc (
  UID INT,
  PID INT,
  FOREIGN KEY (UID) REFERENCES User (UID),
  FOREIGN KEY (PID) REFERENCES Proteins (PID),
  PRIMARY KEY (UID , PID)
);

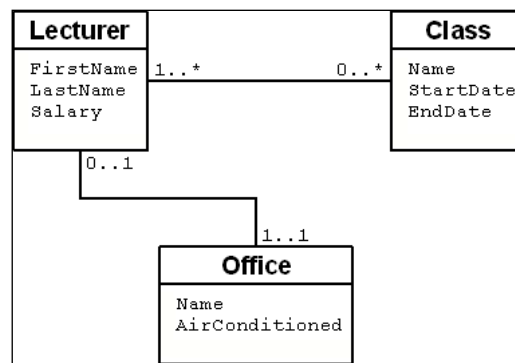
/* add your insert statements */
```

Use *relationships.sql* script to practice the following tasks:

1. Create a database called *myDB* for the tables (*User*, *Proteins*, *Alloc*) and select it as the default schema.
2. Populate the database with at least 4 records in each table (i.e. write 4 *INSERT INTO* queries for each table). You should choose sensible values for the records, in particular the *Proteins* table.

3. Check your records in each table by running a *SELECT * FROM* query on each table.
4. Write a *VIEW* with *SELECT* query to list the first names and last names of the user along with the full Name of the protein they are associated with.
5. Can you think of any other ways to do step 4?
6. What kind of relationship exists between *User* and *Protein*? Draw an ER diagram.

5 - Implementing an ER Diagram



Consider the above ER diagram:

1. Create the Lecturer and Office tables first, ensuring that an office can be associated with 0 to 1 lecturers, while a Lecturer must always be associated with 1 office. This means offices do not need lecturers, but lecturers need offices.
2. Create the Class table and ensure a many-to-many relationship.