School of Computer Science

COMP47470

Lab 3
Relational Database Management Systems (MySQL)

Teaching Assistant:	Leandro Batista de Almeida
Coordinator:	Dr Anthony Ventresque
Date:	Thursday 7th February, 2019
Total Number of Pages:	5

General Requirements/Resources

For this lab session you can either use the account created for you on `csserver.ucd.ie` (recommended) or you can install a MySQL server on your own laptop.

Part I
MySQL: a Relational Database Management System

Tutorials

- Basic MySQL Tutorial
- Tutorial from MySQL Documentation
- Tutorial from Vogella

References

- MySQL Reference Manual
- MySQL Documentation

1 Connection to a MySQL Client (command line)

The `mysql` command is a simple shell (command line interface) that supports interactive and non-interactive modes (i.e., you can start an online "session" or run commands offline). Here we assume you are connected interactively to the client application.

1. The command below shows the various options of the `mysql` command:
   ```
   $> mysql   help
   ```
2. There are various options to get connected to the client application, for instance:
   ```
   $> mysql -h localhost -u your-username -p
   ```
   use this one to get an interactive MySQL shell.
3. There is an example database called 'World Database' available for MySQL.

   The instructions on how to install the database are available at: https://dev.mysql.com/doc/world-setup/en/

   Remember to change the name of the database to your username in the sql scripts used to import this database.

2 Basic SQL Commands

Do not forget the semicolon after the SQL commands - or the system will expect the lines to continue.

1. The following command displays all the databases associated with a user:

   ```
   $> show databases;
   ```

2. Select the world database (remember to change the name 'world' to your username):

   ```
   $> use world;
   ```

3. Show the tables in the database currently selected:

   ```
   $> show tables;
   ```

4. Display the structure of a table:

   ```
   $> describe city;
   $> describe country;
   ```

3 Basic DDL (Data Definition Language) - Create a Database and a Table

1. The following two commands display the options associated with `create` and create a new database.

   ```
   $> help create database;
   $> create database yourLoginName;
   ```

2. Set the current database:

   ```
   $> use yourLoginName;
   ```

3. Create a new table in the current database:

   ```
   $> help create table;
   $> create table student (
   student_id smallint unsigned not null,
   name varchar(60) not null,
   dept_name varchar(50) not null,
   tot_cred smallint unsigned,
   age smallint unsigned,
   primary key (student_id)
   );
   $> describe student;
   ```

4 Basic DML (Data Manipulation Language) - Insert, Update, Delete

1. The following two commands display the options associated with `insert` and insert a new tuple in the table `student`.

```
$> help insert;
$> insert into student (student_id, name, dept_name, tot_cred, age)
values (1, 'Edgard Frank Codd', 'Computer Science', 10, 79);
```

2. Show all records:

```
$> select * from student;
```

3. Insert new records:

```
$> insert into student (student_id, name, dept_name, tot_cred, age)
values (2, 'James Nicholas Gray', 'Computer Science', 10, 63);
$> insert into student (student_id, name, dept_name, tot_cred, age)
values (3, 'Alan Mathison Turing', 'Mathematics', 10, 41);
$> insert into student (student_id, name, dept_name, tot_cred, age)
values (4, 'Claude Elwood Shannon', 'Electrical Engineering', 10, 84);
$> insert into student (student_id, name, dept_name, tot_cred, age)
values (5, 'Grace Brewster Murray Hopper', 'Computer Science', 10, 85);
```

4. Display the students of the School of Computer Science:

```
$> select * from student where dept_name = 'Computer Science';
```

5. Display the name of students whose age is less than 50:

```
$> select name from student where age < 50;
```

6. Get some info about the command `update`:

```
$> help update;
```

7. Update the department name of the student 3 (Alan Turing) to Computer Science:

```
$> update student set dept_name = 'Computer Science'
where student_id = 3;
$> select * from student where student_id = 3;
```

8. Get some info about the command `delete`:

```
$> help delete;
```

9. Delete the record for student_id = 4:

```
$> delete from student where student_id = 4;
$> select * from student;
```

5 Basic DML (Data Manipulation Language) - Retrieval of Records

The `select` command is used to retrieve records selected from one or more tables. The retrieved fields and records can be filtered and ordered, and numeric functions can be applied to the outputs.

1. Get some info about the command `select`::

```
$> help select;
```

2. Retrieve only the fields name and age from all records in the table students:

```
$> select name, age from student;
```

3. Order the records by a field or a set of fields:

```
$> select * from student order by name;
$> select * from student order by dept_name, name;
$> select * from student order by age desc;
```

4. Filter the records using a condition based on values of the fields:

```
$> select name, age from student where age >= 60;
$> select name, age from student where age >= 60 and age <= 50;
$> select * from student where dept_name =  Computer Science ;
```

5. Filter the records using parts of strings, with a penalty on performance:

```
$> select name from student where name like 'Allan%';
$> select name from student where name like '%Nicholas%';
```

6. Count how many records are in the table:

```
$> select count(*) from student;
```

7. Retrieve the average age from students:

```
$> select avg(age) from student;
```

8. Retrieve the sum of credits of all students, and from students of a department

```
$> select sum(tot_cred) from student;
$> select sum(tot_cred) from student where dept_name = 'Computer Science';
```

9. Group the results of a function by a field

```
$> select dept_name, sum(tot_cred) from student group by dept_name;
```

6 Joining information of two or more tables

The join operation is an important one in relational databases. When you join information of two or more tables, a field (or a set of fields) must be used as a pivot in the tables, to correlate the information. If the two tables have the same names for some of the fields, the name of the table must be used as a prefix to avoid ambiguities. MySQL supports inner and outer joins, as well left and right joins too. You can read about these types of joins in the documentation (see page 2 of this lab script). For this exercise, the World Database will be used.

1. Get some info about the command `join`:

```
$> help join;
```

2. Use the world database and display the structure of the tables and some of their content

```
$> use world;
$> show tables;
$> describe city;
$> describe country;
$> select * from city limit 10;
$> select * from country limit 10;
```

3. Display the name of the country, and the name of the city, for the first 10 records.

```
$> select country.name, city.name
from country
join city
where country.code = city.countrycode
limit 10;
```

4. Display the languages spoken in a country sorted by percentage. Try Ireland (code = 'IRL') and Brazil (code = 'BRA').

```
$> select name, continent, localname, language, isofficial, percentage
from country
join countrylanguage
where
code = countrycode and
code = 'IRL'
order by percentage desc
```

7 Bash Script for MySQL Server

As said in the first section of this part of the practical, there are two ways to access the MySQL client: interactive (what we've seen so far) or not. Two examples of the non-interactive mode are:

```
mysql -h localhost -u username -p -e "show databases"
mysql -h localhost -u username -p -D "word" -e "show tables"
```

Revisit some of the commands we've seen so far in this mode. See how to update a database, send queries, etc. directly from Bash commands/a Bash script.

8 Exercise

Get the "titanic" database from this url: http://jse.amstat.org/jse_data_archive.htm. Try to understand the dataset (check the "titanic.txt" file) and then create and populate a database (1 table is enough - e.g, use a Bash script that would read the csv file and create sql queries to insert values in the database). Answer the following questions:

- Describe how you created the structure of the database.
- Describe how you populated the database.
- How many passengers were there on the Titanic? (use a SQL query)
- How many passengers survived? (use a SQL query)
- What percentage of passengers survived? (use a (nested) SQL query)