# **COMP-1701 - Transferring Data to Databases**

## **Data Science & Machine Learning (DSML) - RRC Polytech**

Module B.1 - Database CLI - Connection, Server Commands and CREATE DATABASE

Any MySQL installation comes bundled with a command-prompt/terminal-based CLI (Command-Line Interface), which is also a pseudonym for what we call a Client. Upon connection to the database server, we are basically fulfilling a Client-Server relationship architecture and can make requests from the MySQL Client to the Database Server. The CLI is sufficient to do everything that you need to within.

WHEN COPYING & PASTING CODE, COMMANDS, OR QUERIES

Always select text from the end of the query after the semi-colon; and select up to and including the starting of the query. This is always how you should copy code or data, from the end to the start, and make sure you aren't copying any extra spaces or lines before or after.

Connecting

When you went thru the install, if you recall, we added a password for the root account. To initially login, you need to specify the root account and that password when connecting to the CLI. Start a PowerShell or Terminal window and type in the following:

Text

Description automatically generated

If you've successfully logged in you will get the following Welcome message:

Graphical user interface, text, application

Description automatically generated

SEMI-COLONS [ ; ]

CLI **SQL Commands/Queries** require a semi-colon at the end [ ; ] and in the CLI, you will need to hit the [**ENTER**] key to execute it.

If you forget to type in the semi-colon and you've already hit [**ENTER**], on the resulting line, just enter in a semi-colon, and hit [**ENTER**] again, then your command/query will execute.

COMMENTS /\* \*/ vs. -- vs. #

Just as when you are coding, you need to add comments to the commands you are creating, so you can pass these SQL scripts you create to another database developer.

-- Single Line comment, I recommend you use this  
-- format always, it is for a single-line comment  
-- and when selecting SQL queries to run, these   
-- won't interfere with the following comment  
-- style, -- should be followed by a space

# not universally supported, default is to use --   
  
/\* this is a multi-line comment, best used at the start of your documents, to identify, your name, student number, course, etc. and temporarily within your code, to block out entire sections of code, while testing scripts, this comment style should be used very sparingly \*/

-- below is what should be up at the top of your  
-- cheatsheet and boxstore scripts  
/\*   
\*\* Name:  
\*\* Assignment:  
\*\* Date:   
\*\* History: YYYY-MM-DD  
\*\* Description and list of changes  
\*\* - change 1  
\*\* - change 2  
\*\* - change n  
\*\*  
\*\* YYYY-MM-DD  
\*\* Latest description and list of changes  
\*\* - latest change n  
\*\*/

SHOW DATABASES; | SHOW SCHEMAS;

To show you a listing of the databases pre-installed type in:

SHOW DATABASES;  
  
-- or can use:  
*SHOW SCHEMAS;*

These are system databases or system schemas, that have different functions:

Graphical user interface, text

Description automatically generated

* **information\_schema** is where all databases, tables, columns, keys, indexes, and permissions are stored, for each database hosted on the server
* **mysql** contains database server related information
* **performance\_schema** and **sys** are used for benchmarking performance

USE <database\_name>;

To start working in a database, in this case, we will jump into information\_schema, type in USE information\_schema; and hit [**ENTER**]

Graphical user interface

Description automatically generated

You will note that the **MariaDB[(none)]** changed the **(none)** part to have **information\_schema** and it also mentions **Database changed**, which indicates the command was successful, and you can directly work in this database now.

**SELECT DATABASE();** will return the current database's name in a query view, with **DATABASE()** as the column name and the current database as its only row (note: can be useful to use when working in testing, staging, and production database servers, if some query results aren't making sense, it might be because you using the wrong database).

Graphical user interface, text, application

Description automatically generated

SHOW TABLES;

Now that you are in a database, you many want to see its listing of tables, which you can do by running the **SHOW TABLES;** command. As you can see, the **information\_schema** database has several tables within, most of which you will use unbeknownst to yourself. Once you begin creating your databases, these tables will get filled up with the structure of your database architecture.

Text

Description automatically generated

DESCRIBE <table\_name>;

Run the following in your command-line (just copy and paste it over to your CLI)

DESCRIBE collations;

This command outlines the structure of the **collations** table; then hit [**ENTER**] and you will get the following results:

A screenshot of a computer

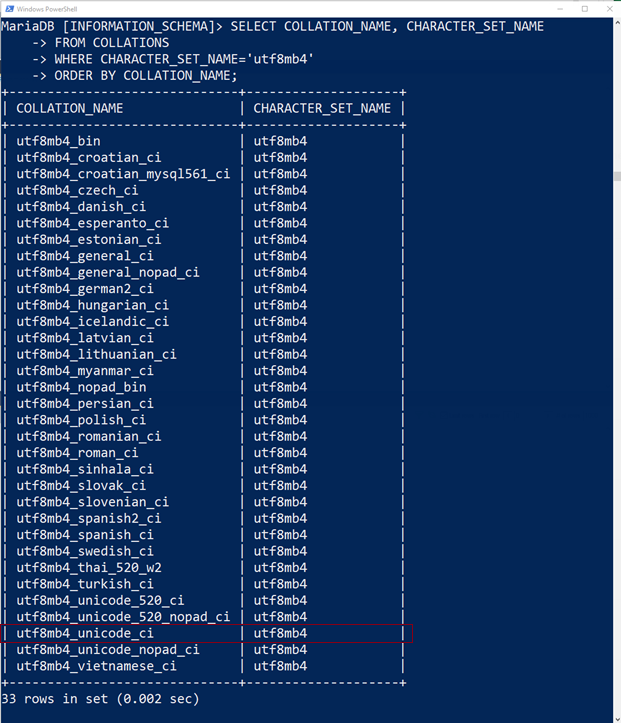
Description automatically generated with medium confidence

**Field**, which depicts the listing of "**Column Names**" in the *SCHEMATA* table and their co-attributes of:

* **Type**, refers to the:
* **Data Type** needed for the field, such as characters (any visual keyboard keys), integers (numbers, with no decimals), or longer variable length text (or characters, or what we call a string, like a sentence)
* **Length** (or size) of the field , and
* **Unsigned**, for when it is an integer column, this would indicate the column can only have positive numbers starting from 0, then 1, 2, 3...etc. No -1 or smaller.
* **Null**, refers to whether the field can contain a Null value (Yes) or Not Null (NO), if the Null column indicates NO, that means the rows must have a value within. Think of Null as nothing.
* **Key**, refers to whether the column is a key column (not relevant for this faux "table"
* **Default** value, so if Null (ie: nothing) is inserted to the row when creating the record, then the Default value will kick in.
* **Extra** column, which denotes any extra information stored by the column.

Now let's run our first actual query, and see what data is inside:

SELECT collation\_name, character\_set\_name  
FROM collations;  
WHERE character\_set\_name='utf8mb4'  
ORDER BY collation\_name;



You will notice that these rows are just filtering down the table, to values of **'utf8mb4'** and this particular character set is the latest and most compatible of the **UTF-8** standard that **MariaDB** has available for use.

When creating our first database, we will use the **'utf8mb4'** as our character set and **'utf8mb4\_unicode\_ci'** as our collation.

CREATE DATABASE & CHARSET/COLLATE

Activity:

Within this course, we are building up a database from the very beginning, so the first database will be your **f**irst and **l**ast name initials in lowercase, an underscore '\_', your student id , followed by an '\_' underscore character, and the text 'boxstore'.

So, like for **M**ichael **J**ordan, his initials are mj and with student id of **0232323**  it would be **mj\_0232323\_boxstore**

Now that you have your database name, and the collation and character set from earlier, please note the syntax of the CREATE DATABASE  command:

-- replace database\_name with your database name  
CREATE DATABASE *database\_name*  
CHARSET='utf8mb4'  
COLLATE='utf8mb4\_unicode\_ci';

 ...and this will create the database you will be working in.

If you have made an error, just retype it in, exactly as written above, of course replacing the ***your\_database\_name*** with the database name I've asked you to put in.

SQL files you submit should use this exact name with .sql as the file extension.

DROP DATABASE

If you made an error in the naming convention specified, you will need to run a:

DROP DATABASE IF EXISTS *database\_name*;

...and then re-run your CREATE DATABASE command.

When running an export on the structure of the database, it will usually create 2 statements for the database itself. The DROP statement with an IF EXISTS parameter, and then a CREATE statement with IF NOT EXISTS. Therefore if you select both DROP and CREATE, it will DROP & CREATE the database. If you select just the CREATE command, it will only recreate the database, if the database is not yet created.  
  
**DROP/CREATE DATABASE syntax**

-- DROP DATABASE IF EXISTS *database\_name*;  
CREATE DATABASE IF NOT EXISTS *database\_name*  
CHARSET='utf8mb4'  
COLLATE='utf8mb4\_unicode\_ci';

Once you are complete, you can run a USE database\_name command to get into it.

-- DROP DATABASE IF EXISTS *database\_name*;  
CREATE DATABASE IF NOT EXISTS *database\_name*  
CHARSET='utf8mb4'  
COLLATE='utf8mb4\_unicode\_ci';

USE *database\_name*;

All of these commands, syntax, and notes, should be copied over to your \*\_cheatsheet.sql script, including the example header used to document your file (name, date, history, etc.)

Files required for every submission:

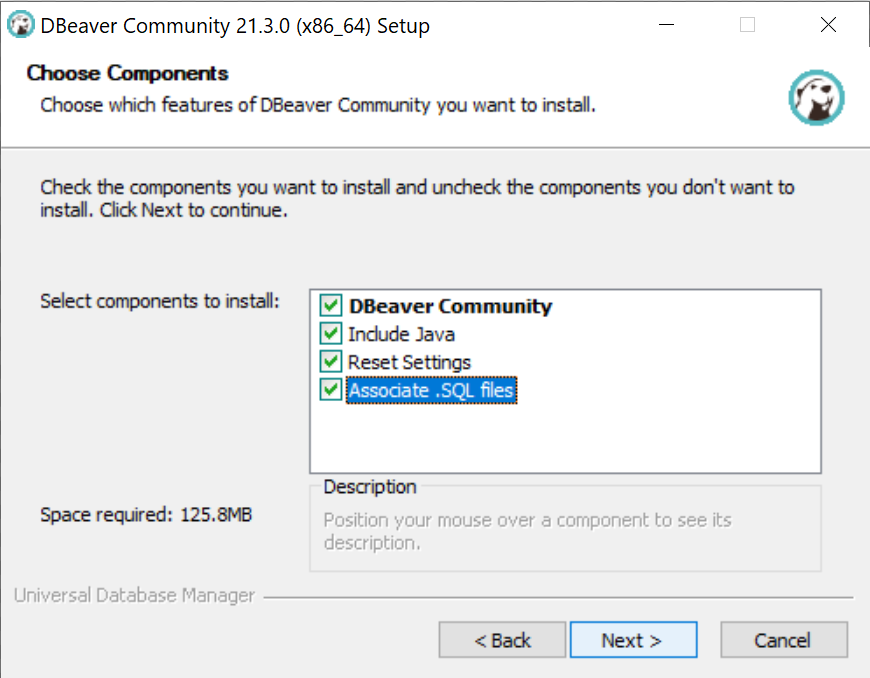
Copy the commands you have used, to 2 files called:

* ***FL*\_*STUDENTID*\_boxstore.sql** & ***FL\_STUDENTID*\_cheatsheet.sql**
* **FL** are the initials of your first and last name (and possibly middle or hyphenated last names, whichever are your normal initials you use are)
* **Student ID** is of course, your student ID
* **\_boxstore.sql** will be the descriptor for your electronics/box store
* **\_cheatsheet.sql** will be your syntax, as in a study guide and place to access any SQL syntax easily, throughout the course
* all of the commands and directory locations etc, from the beginning should be copied into this \_cheatsheet.sql file.
* **filenaming** should always be LOWERCASE.

Both files will be required submissions, at each phase of our database build/assessment. We will go over this now.

And when asked to, submit them to your Dropbox.

For next class, find and install **DBeaver**, it is a SQL GUI, which will give us better visibility for our data and database builds. From within DBeaver, this is where we will be using our FL\_STUDENTID\_boxstore.sql file. Has installs for both Windows and Mac. **Download 64-bit versions**. And check off all options during the install…for this screen.



When working, open your cheatsheet in a text editor, such as Notepad++, and open your working boxstore script in DBeaver. If you have the windows accessible and visible, you can copy syntax from your cheatsheet to your boxstore script and adjust within the boxstore script for the build portion of the course. You don't necessarily need to *Associate .SQL files* to only open in DBeaver to note.

The boxstore script should only have commands that builds the database, tables, etc. If it is just syntax, and not working code, that should be only in your cheatsheet script.

Additionally, it is in your best interest to create a COMP-1701\_Transferring\_Data\_to\_Databases folder, in your Users\Documents\RRC\Term folder. If you don't know how to manage files, we can do this together. Folders you should create are:

Users\Documents\RRC\2023\_Fall\

COMP-1701\_Transferring\_Data\_to\_Databases\

lecturefilesfromlearn.docx

FL\_STUDENTID\_cheatsheet.sql

\_scripts\

FL\_STUDENTID\_boxstore.sql

…and other project SQL scripts and data files

## 