

Introductory MySQL Commands

Principles of Databases (CS 365)

Fall 2020

UTF-8 Character Set Conflicts

- Use UTF-8 character sets whenever possible

MySQL Configuration File

- On macOS, add *my.cnf* to the */etc* folder.
- In Windows *my.cnf* may be called *my.ini* and could be in one of many places. Read the official documentation from *dev.mysql.com* at <https://dev.mysql.com/doc/refman/5.7/en/option-files.html>

Logging in

Use the *mysql* command to log in. This command says, “Log in to MySQL as user (*-u*) *root* and tell the CLI to request my password (*-p*).”

```
mysql -u root -p
```

Logging in

You can also close the space between ***-u*** and ***root***, as follows:

```
mysql -uroot -p
```

Logging in

You can also append the password to the **-p** option. (No space character.) For example, if my password were ***password***, I could log in as follows:

```
mysql -u root -ppassword
```

or

```
mysql -uroot -ppassword
```

Logging in

Appending the password to the **-p** option is insecure, as the password would sit as a plain text entry in your CLI's history file.

In ***bash***, for example, you'd find the password in ***.bash_history***. You could clear it (and the rest of your history) with the **-c** flag to the ***history*** command:

```
history -c
```

Logging in

The more secure option is to have MySQL request your password via your CLI.

```
mysql -u root -p
```


Exiting MySQL

Similar to exiting your CLI, exiting MySQL is simply...

EXIT

Warnings

If an error is generated, you can see the latest warning with

SHOW WARNINGS;

Checking the Status of the Database

You can view some important information, such as current user and database, IP address, and character set configurations, using the simple ***STATUS*** command:

STATUS

Creating a Database

Let's create a database called ***users*** with a default and collation character set of UTF-8.

```
CREATE DATABASE `users` DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;
```

Note: This doesn't place focus on the new database; it simply creates it.

Creating a Database | Placing Focus

To work with a database, you need to focus on it by using the ***USE*** command. Let's focus on the ***users*** database:

USE users

If you now run ***STATUS***, you'll see, below ***Connection id:***,
Current database: users

Add a User to the Database with a Password

Let's create a user called ***the-user*** whose password is ***the-password***.

```
CREATE USER 'the-user'@'localhost' IDENTIFIED BY 'the-password';
```

Provide a User Access to the Database

Let's now grant ***the-user*** all privileges to *all* the tables under the ***users*** database

```
GRANT ALL PRIVILEGES ON users.* to 'the-user'@'localhost';
```

Logging into the Database with the New User

```
mysql -u the-user -p
```


Show Databases

You can see the databases to which you have access with the ***SHOW*** command:

SHOW DATABASES;

Create a Table

```
CREATE TABLE students (  
    first_name VARCHAR(20) NOT NULL,  
    last_name VARCHAR(20) NOT NULL  
);
```

Note: Both are set to **NOT NULL**, meaning that an entry into the *students* table can only happen when both values are present. What happens when you try to defeat the **NOT NULL** rule?

Flush the Contents of a Table

Flushing the contents of a table means that MySQL will drop the tables, then recreate them without any entries.

TRUNCATE TABLE students;

Drop/Delete a Table

Let's delete the *students* table.

DROP TABLE students;

Note: This isn't the same as ***TRUNCATE***, which flushes the tuples in the table, but doesn't delete the table.

Insert a Single Record in a Table (CREATE)

```
INSERT INTO students  
  (first_name, last_name)  
VALUES  
  ("Edward", 'Bobward');
```

Insert Multiple Records into a Table (CREATE)

```
INSERT INTO students  
  (first_name, last_name)  
VALUES  
  ("Edward", 'Bobward'),  
  ("Ed", 'Bob'),  
  ("Frank", "Einstein"),  
  ("Johnny", "Rotten");
```

Read All Records from a Table (READ)

```
SELECT * FROM students;
```

Read All Records from a Table with a Matching Clause (READ)

Let's get all students whose first name is Frank.

```
SELECT * FROM students WHERE first_name LIKE "Frank";
```


Read All Records from a Table that Start with a String (READ)

Let's get all students whose first name starts with "ed".

```
SELECT * FROM students WHERE first_name LIKE "Ed%";
```

or for a more case-insensitive search:

```
SELECT * FROM students WHERE UPPER(first_name) LIKE UPPER("ed%");
```

Read All Records from a Table that End with a String (READ)

```
SELECT * FROM students WHERE last_name LIKE "%Bob";
```

or for a more case-insensitive search:

```
SELECT * FROM students WHERE UPPER(last_name) LIKE UPPER("%bob");
```

Read All Records from a Table's Column (READ)

Let's get all *first_names* from the *students* table.

```
SELECT first_name FROM students;
```

Read All Records from a Table's Column (READ)

Or *last_names*.

```
SELECT last_name FROM students;
```

Read All Records from a Table in Reverse Order (READ)

```
SELECT last_name, first_name FROM students;
```

Describe the Fields/Columns in a Table

There are at least 3 different ways to describe the structure of a table.

SHOW COLUMNS FROM students;

DESC students;

DESCRIBE students;

Update (UPDATE)

Let's change Frank's first name to Albert:

```
UPDATE students SET first_name="Albert" WHERE first_name="Frank";
```

Remove (DELETE)

Let's remove Johnny, who's no longer a student:

```
DELETE FROM students WHERE first_name="Johnny";
```


