Goals:

Prepare your database management system.
Create a table and data.
Learn about the Result and the Response grids.
Homework Deliverables are due before the next class begins.

Important: When sending deliverables, please use a file of type **.txt** file or of type **.sql** (a text file with a different suffix) to send your code (do not send other file types, not rtf, not a docx.) This will help create consistency that will assist in grading in blackboard.

1.Set up mysql in your onyx environment

Follow the instructions in this document for setup instructions:

(We will use the Java instructions during a class homework in the future, *that section is not required at this time*.)

Set Up MySQL and Java

Note: It's very important that you create a named connection to MySQL using the port assigned to you (mentioned in the MySQL setup document.)

If you have any problems, the following notes may assist you, refer to this document in the future as needed:

Tip: While in the lab, you must "ssh" to onyx. (You can see that your command line may mention "node" in the server name in the prompt, if so be sure to create a secure shell to onyx.) The SSH instruction needs the "-Y" flag. (This is the "X11 forwarding" option that allows mysql-workbench to open a window in the local computer screen.)

```
ssh -Y myloginname@onyx.boisestate.edu
```

Tip: Some ssh terminals on your home computer do not enable X11 forwarding by default. You may need to "enable x11 forwarding" (to allow my-sql workbench to open as an "X11 display window" in your local computer), so: use the -Y option in the ssh command; or check the X11 forwarding option in your ssh application. (For example, in Putty there is an SSH setting that you will select for your connection profile to enable this.)

Note: Sometimes port issues arise, you will know it happened to you **if you cannot log into your database** connection in my-sql workbench using your password. If that happens, then you may have "lost your port." The instruction link above include a way to get a new port, but those instructions are not necessary to review until/unless you lose your port. (If that happens, the directions will ensure that you will not lose your database.)

Note: if you are working from your own computer, you may need to download and install an X11 display server (like xming) to allow the MySQL workbench to open a MySQL Workbench window in the local environment. If you find that you do need this, then the client will have to be downloaded *and running* for X11 display windows to open in your local computer. MobaXterm has and some other ssh terminals may have this built-in. Putty does not have this built in.

3. Create a database - Open a query window and run the following 2 lines of code (replace the strings for the database name, including the angle brackets> before running this):

```
Create database <mydatabasenamehere>;
USE <mydatabasenamehere>;
```

-- Note: Be sure to save all of the code you write into one or more files

4. Create the `Class` Table - copy the following code and execute it in your MySQL database query window:

```
Create table Class(ID int auto_increment
, Name varchar(50) NOT NULL
, Description varchar(200)
, Code varchar(10) UNIQUE
, MaximumStudents int DEFAULT 10
, primary key (id));
-- the "NOT NULL Constraint" makes this column be a required data element
-- the UNIQUE constraint makes sure each code will be distinct
-- The default constraint will ensure there is data in the column if you do not specify a
value
-- we will learn more about primary keys in the next class
```

5. Run this Unit test - copy the following INSERT statement and run it in your database query window. (Note: This is code will test the **`UNIQUE`** *Constraint*.) This test is to help you practice using the **response grid**.

Deliverable 1 : Send me the error message from the response grid. (use right click on the grid and choose "copy response", then paste the results into a text file)

```
-- run the following code by itself,
-- don't include the previous code when running these instructions
        Insert into Class(Name ,Code)
        Values ('Computer Science 310','CSHU310');
        Insert into Class(Name ,Code,  MaximumStudents)
        Values ('Computer Science HU 310','CSHU310',30);

-- This is a single line comment.
-- This is useful for holding a single-line of text that is not code.
```

6. Check out the data in your table: Copy the following SELECT statement and run it in your database query window. This is to practice running code and sending me the data from the result grid.

```
Select * from Class;
```

Deliverable 2: Send The result data. (Right Click in the Result grid and select this option: "copy row with names unquoted", paste the results into a text file.)

```
/*
This is a multi-line comment section.
When delivering results and responses, it will help to put non-code content into comments.
*/
```

7. Understanding the Constraints: Look at the table definition. Look at the result set. Look at the code you've run on the database. Consider the unit test results. Answer these questions (in 1 or two sentences each) you do not need to write any code:

Deliverable 3: Answer the following questions

- **Question 1:** What did the DEFAULT constraint on MaximumStudents do to the table data?
- **Question 2:** What did the UNIQUE constraint do when the second INSERT ran?
- **Question 3:** What did the auto_increment on the ID column do to the table data? Guess what you think will happen when more records are inserted.

- **Question 4:** *Based on the unit test example*, it possible to skip naming some columns of the table when writing and running an Insert instruction? Why do you think this is the case?