Notes on ASSIGNMENT 3  
600.315/415 - Database Systems

• Part (g) of the Assignment can be found on the third page.

• You should use your username_hw2 database for development purposes. Also, make sure your procedure definitions run properly on dbase.

• You will be doing a lot of output that deals with numbers. To limit the number of decimal places, you can use the MySQL command `FORMAT`:

```
mysql> SELECT FORMAT(1,3);
+-------------------+
| FORMAT(1,3)       |
+-------------------+
| 1.000             |
+-------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT FORMAT(1.66666,2);
+-------------------+
| FORMAT(1.66666,2) |
+-------------------+
| 1.67              |
+-------------------+
1 row in set (0.00 sec)
```

You get the idea. It is a bit inconvenient since you will have to use `FORMAT` with every single numerical output, but the alternative is having too many decimal places in each numerical output, which makes your output a little unpleasant to look at, causing the grading staff severe headaches and -possibly- stomach pains.

• I have already prepared solutions for the Assignment and chose not to use any views (although the Assignment allows them). Whether you will define views or not is up to you. If you do create views, however, make sure you place all those definitions in your SQL file (preferably at the beginning of the file). Views were covered in the SQL lecture, and the MySQL documentation on views is available in the class folder in MySQL/mysql-views.pdf.

• In Part (d), notice that the computed weighted average should be one of the columns of the result table, as opposed to `ShowPercentages`, where the weighted average is displayed using a separate `SELECT` statement.

• In Part (e), your starting point should be `AllPercentages`. This means that `Stats` will be expecting a password parameter, just like `AllPercentages`. 
In case you are confused by the wording of Part (e), your output should consist of five tables, as follows:

```sql
mysql> call Stats('ImTheTA') //
```

```
<table>
<thead>
<tr>
<th>SSN</th>
<th>LName</th>
<th>FName</th>
<th>Section</th>
<th>HW1</th>
<th>...</th>
<th>FExam</th>
<th>CumAvg</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*** A BUNCH OF PERCENTAGE VALUES ***

```

8 rows in set (0.00 sec)

```
<table>
<thead>
<tr>
<th>Statistic</th>
<th>HW1</th>
<th>HW2a</th>
<th>HW2b</th>
<th>Midterm</th>
<th>HW3</th>
<th>FExam</th>
<th>CumAvg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>80.500</td>
<td>80.156</td>
<td>78.906</td>
<td>84.125</td>
<td>84.479</td>
<td>87.063</td>
<td>84.315</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1 row in set (0.00 sec)

```

```
<table>
<thead>
<tr>
<th>Statistic</th>
<th>HW1</th>
<th>HW2a</th>
<th>HW2b</th>
<th>Midterm</th>
<th>HW3</th>
<th>FExam</th>
<th>CumAvg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>29.00</td>
<td>38.75</td>
<td>37.50</td>
<td>55.00</td>
<td>50.83</td>
<td>53.00</td>
<td>51.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1 row in set (0.01 sec)

```

```
<table>
<thead>
<tr>
<th>Statistic</th>
<th>HW1</th>
<th>HW2a</th>
<th>HW2b</th>
<th>Midterm</th>
<th>HW3</th>
<th>FExam</th>
<th>CumAvg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1 row in set (0.01 sec)

```

```
<table>
<thead>
<tr>
<th>Statistic</th>
<th>HW1</th>
<th>HW2a</th>
<th>HW2b</th>
<th>Midterm</th>
<th>HW3</th>
<th>FExam</th>
<th>CumAvg</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1 row in set (0.02 sec)

Query OK, 0 rows affected (0.02 sec)
```

In case you decide to split up the information (for the extra credit) you would have ten tables instead of five (five tables for each section).
Part (g): Implementing Simple PHP/MySQL Interfaces

The PHP/MySQL examples can be found at: 

http://www.cs.jhu.edu/~ozaidan/DB_HW3/

- Write a simple MySQL stored procedure called *MyCalc* that takes two integers and decides which of them is larger, and returns its decision via an OUT parameter. For example, the calls *MyCalc(9,3,@z)*, *MyCalc(2,6,@z)* and *MyCalc(10,10,@z)* should set @z to +1, -1 and zero, respectively. This should be very simple. In fact, you can find something very similar in the *proc_examples.txt* file.

Now, implement an HTML interface (form) that takes in two integers from the user, and calls a PHP script (that you should also implement) that performs a procedure call to *MyCalc* using the two integers, and produces suitable output.

You don’t have to submit anything for this part. This is just to get you going...

- Implement another interface that calls a MySQL stored procedure. This time, the procedure should be called *UpdateMidterm* and should take three parameters: an access password, an SSN, and a new score. The procedure is similar to *ChangeScores* in (f), but is much simpler because it will always update the midterm score.

The HTML interface should take in three values for those parameters, just like in the previous part. The only output you are required to implement is something like “Update successful!” or “Update failed!”, and don’t worry about showing the old and new values as in Part (f).

The HTML and PHP files should be called *update.html* and *update.php*.

- Finally, implement an HTML interface that performs something similar to *AllRawScores* in Part (c). The interface takes a password from the user and produces a table of the raw scores. If the provided password is valid, the table should look exactly like your output in Part (c). Otherwise, the output should be a meaningful error message.

Notice that you will not be able to make a procedure call to *AllRawScores*, just like the list_courses example cannot make a procedure call to *listCourses*. However, we have seen that the solution is pretty straightforward. See the *list_courses_improved* example.

The HTML and PHP files should be called *list.html* and *list.php*.

Tip: One way of checking the validity of a password makes use of the *mysql_numrows* command, which is used like this: $num = mysql_numrows($result); You should think of how this would actually be used for password validation.

*Hint:* it would involve testing if a certain $num is equal to zero.

The definition for *UpdateMidterm* should be within the same SQL file containing procedure definitions from the previous parts of the Assignment. Also, include in your submission all the HTML and PHP code that you wrote as separate files. Therefore, your submission should have a total of five files: one SQL file for procedures/views definition/creation, two short HTML files for the interfaces and two PHP files that actually do the work.