Assignment #2 – Programming in SQL: PL/SQL
Due: Wednesday, February 21st, at start of class

Use the Grade Book database created in Assignment #1 to perform the following tasks:

1. Write a PL/SQL stored procedure that finds the student with the highest overall score (weighted) in each section (of every course). If there is more than one student tied for the highest average, the procedure should return all of the students with that score. The results should be returned in a PL/SQL table of records, where each record has the fields: `course number`, `section number`, `term`, `course title`, `student ID`, `student name`, and `overall score`. Also, write an anonymous PL/SQL block that calls the stored procedure and prints the results to the standard output.
   HINT: It might help to put both the table of records and the stored procedure in a PACKAGE – then have the anonymous block refer to the table of records in the package.

2. Write a PL/SQL stored procedure that takes as input (parameters) the term and section number for a particular course section and process updates to the student scores for this specified class. The updates to the scores are assumed to be in a table (which you create) called `GRADE_UPDATES` with three columns: `compname`, `sid`, and `change`. The change column is a positive or negative number indicating the change to be made to the current component score. If the term or section number of the course in invalid, the procedure should simply return (and do nothing else). Otherwise, the procedure should process each of the updates. If the component name for the student ID is invalid, or if the updated score is outside the range of allowed values (that is, less than zero or more than `maxpoints`), the procedure should send an error message to the standard output and continue processing the remaining updates in the `GRADE_UPDATES` table. Write a PL/SQL anonymous block to test the procedure.

3. Write a trigger that fires when a row is deleted from the `enrolls` table. The trigger should record all of the dropped student's scores in a temporary table called `DELETED_SCORES`, and cascade the deletes (in the trigger – not changing the table structure to include `CASCADE ON DELETE`) to ensure that the referential integrity constraints are maintained.

To Turn In:

For each of the three tasks above, turn in a printout with **both** the SQL statement(s) **and** sufficient output to demonstrate its functioning on your database (e.g., table listings before and after the statement, as well as the output of the queries).

In addition, submit the source code on Blackboard (using the link in the Assignment folder). You should submit the code from (a) the two stored procedures, the two anonymous blocks, and the trigger.