Advanced SQL Lab Exercise 2

Use the University.sql file to do the exercise

Deliverables

You have to produce and submit a document that includes the following for each query:

1. the SQL query, and
2. the print screen of the execution screen that shows that you successfully ran the query.

Intermediate SQL

Using the university schema that you have, write the following queries. In some cases, you need to insert extra data to show the effect of a particular feature - this is indicated with the question. You should then show not only the query, but also the insert statements to add the required extra data.

1. Find the maximum and minimum enrollment across all sections, considering only sections that had some enrollment, don't worry about those that had no students taking that section
2. Find all sections that had the maximum enrollment (along with the enrollment), using a subquery.
3. As in in Q1, but now also include sections with no students taking them; the enrollment for such sections should be treated as 0. Do this in two different ways (and create require data for testing)
    1. Using a scalar subquery
    2. Using aggregation on a left outer join (use the SQL natural left outer join syntax)
4. Find all courses whose identifier starts with the string "CS-1"
5. Find instructors who have taught all the above courses
    1. Using the "not exists ... except ..." structure
    2. Using matching of counts (don't forget the distinct clause!).
6. Insert each instructor as a student, with tot_creds = 0, in the same department
7. Now delete all the newly added "students" above (note: already existing students who happened to have tot_creds = 0 should not get deleted)
8. Some of you may have noticed that the tot_creds value for students did not match the credits from courses they have taken. Write and execute query to update tot_creds based on the credits passed, to bring the database back to consistency.
9. Update the salary of each instructor to 10000 times the number of course sections they have taught.

Advanced SQL

In this assignment, you will write more complex SQL queries, using the university schema by default. Again, you have to add required data to test your queries, and must include the SQL statements to create the data along with your queries.

1. The university rules allow an F grade to be overridden by any pass grade (A, B, C, D). Now, create a view that lists information about all fail grades that have not been overridden (the view should contain all attributes from the takes relation).

2. Find all students who have 2 or more non-overridden F grades as per the takes relation, and list them along with the F grades.
3. Grades are mapped to a grade point as follows: A:10, B:8, C:6, D:4 and F:0. Create a table to store these mappings, and write a query to find the CPI(Cumulative Performance Index) of each student, using this table. Make sure students who have not got a non-null grade in any course are displayed with a CPI of null.
4. Find all rooms that have been assigned to more than one section at the same time. Display the rooms along with the assigned sections; I suggest you use a with clause or a view to simplify this query.
5. Create a view faculty showing only the ID, name, and department of instructors.
6. Create a view CSinstructors, showing all information about instructors from the Comp. Sci. department.
7. Insert appropriate tuple into each of the views faculty and CSinstructors, to see what updates your database allows on views; explain what happens.
8. Grant permission to one of your friends to view all data in your student relation.
9. Now grant permission to all users to see all data in your faculty view. Conversely, find a friend who has granted you permission on their faculty view, and execute a select query on that view.