Database Systems
Project 1: Writing SQL Queries in Oracle
(Due: October 31, 2016, at the start of the class)

The following normalized tables from the Student Registration System (some tables have been simplified) will be used in this project:

Students(B#, firstname, lastname, status, gpa, email, bdate, deptname)
Courses(dept_code, course#, title)
Course_credit(course#, credits)
Classes(classid, dept_code, course#, sect#, year, semester, limit, class_size, faculty_B#)
Faculty(B#, firstname, lastname, rank, office, email, phone#, deptname)
Enrollments(B#, classid, lgrade)
Grades(lgrade, ngrade)

As a general clarification, we assume that no student takes the same course (including different sections of the same course) more than once. If you have questions about these tables, please contact the instructor for clarification.

1. Preparation

Save the sql script file proj1_tables_script16.txt as proj1_tables_script16.sql in your bingsuns account.

To run the above script from your Oracle account, use

SQL> start proj1_tables_script16

Then check whether all tables are created correctly in your Oracle account.

2. Project Requirements

There are 20 statements (see Section 3 below) in this project and you are asked to write one or more SQL query for each statement. Your query should take into consideration that the tuples currently in the database may change. In other words, your query must be correct for any valid state of every table. It is very important that you understand each query statement correctly. If you have any doubt about the correct interpretation of a statement, please ask the instructor for clarification by posting your questions in the Project 1 thread in the discussion forum on blackboard. For this project, **it is required that your queries do not return un-necessary duplicate results while keeping necessary duplicate results.** For this project, identical results corresponding to different entities are considered to be necessary duplicates while identical results corresponding to the same entity are considered un-necessary duplicates. The following example illustrates necessary and un-necessary duplicates results. Consider query "select first name from students;". If two students have the same first name "Lisa", then displaying "Lisa" twice is considered necessary but displaying "Lisa" more than twice is considered un-necessary. The query (or queries) for each statement is worth 5 points. Keeping un-necessary duplicates or not keeping necessary duplicates will result in one point deduction. No other partial credit will be given in general.

It is suggested that you first test each query individually and save each query in a different file (with extension .sql). After all queries have been tested to your satisfaction, you can run all the queries in a sequence and save the entire session in a spool file. Suppose you have saved your queries in files query1.sql, ..., query20.sql. Follow the steps below to generate the spool file after you have logged on Oracle:

```
SQL> set echo on
SQL> spool project1.txt
SQL> start query1
.......
SQL> start query20
SQL> spool off
```

To prepare your submission of Project 1, follow the steps below:

(1) **Edit file project1.txt** by **adding your name** at the top of the file and **adding the following honesty statement**: "I have done this assignment completely on my own. I have not copied it, nor have I given my solution to anyone else. I understand that if I am involved in plagiarism or cheating I will have to sign an official form that I have cheated and that this form will be stored in my official university record. I also understand that I will receive a grade of 0 for the involved assignment for my first offense and that I will receive a grade of "F" for the course for any additional offense." No other changes to the file are permitted.

(2) Print the edit file project1.txt, **sign your name** next to the above statement. **Your printout must list each SQL query followed by the result of executing the query.**

(3) Hand in the hardcopy printout at the start of the class on October 31, 2016.

3. Query Statements

The following are the 20 query statements:

1. Find the dept_code, course# and title of each course that was offered in the Spring semester of 2016. In the output, dept_code and course# of each course should be concatenated under a new column header course_id.

2. Find the B#, first name, last name and GPA of each student who has taken at least one CS course and at least one math course.

3. Find the dept_code and course# of each course that was not offered in the Fall semester of 2015.

4. Find the B#, firstname, lastname and GPA of each graduate student who has received a grade below B (i.e., B-, C+, C, C-, D, F) at least one course he/she has taken.

5. Find the firstname and lastname of each student who has never received an A for any course he/she has taken. In the output, firstname and lastname of each student should be concatenated with a space in between under a new column header name.

6. Find student B# and faculty B# pairs (named student_B# and faculty_B#, respectively) such that the student has taken a course from the faculty but they are not from the same department.

7. Find the dept_code and course# of each course that has been offered the most number of times (each record in the classes table corresponds to a course offering). Note that it is possible that more than one course may satisfy this query condition; in this case, all such courses should be retrieved.

8. Find the classid, dept_code and course# of each class offered in Spring 2016 that is not full and for each such class, also list the number of seats available (computed by limit – class_size) under the header "seats_available".

9. Find every student (all attributes are needed) who has taken more than 3 classes. Write a query with a correlated subquery and write another query with an uncorrelated subquery.

10. Find every class (all attributes are needed) that is offered by the CS department in the Spring semester of 2016 and has less than 3 students enrolled. For this query, the use of the class_size information from the classes table is not permitted.

11. Find the B# and first name of every student who has taken all 400-level CS courses. **Here we are referring to courses, not classes.**

12. Find the title of each course that has been taken by student B001 but not by student B002.

13. Find the first name of every student who has taken at least one course that has been taken by student B005. Note that here we are talking about taking the same course, not just the same class. (Clearly, student B005 satisfies the condition and his first name should be included in the output.)

14. Find the dept_code and course# of each course that has two or more classes in the same semester of the same year. The query should also show the semester and year information for each qualified course.

15. Find the B# and firstname of each student who has received at least one highest grade in one of the classes he/she has taken. Suppose all possible grades are (A, A-, B+, B, B-, C+, C, C-, D, F, I). Note that the highest grade given to students in a class is not necessarily A. (Hint: You may need to utilize the number grade ngrade for this query.)

16. List the dept_code, course# and title of each course that has been taken by student B003. For each such course, also list the grade the student received. If no grade has been assigned for a course, output "to be assigned" as the grade information for the course.

17. Find the dept_code, course# and title of each course whose title contains "systems" and that has been taken by all students whose GPA is higher than 3.25. Note that even though a qualified course is required to be taken by all students whose GPA is higher than 3.25, it may also be taken by some students whose GPA is not higher than 3.25.

18. Find the B#, firstname and the total number of credits that have been earned by each student. If a student has not taken any course, he/she is assumed to have earned zero credits and zero credits should be reported for such a student. Don't count the credits when no grade is given for a class.

19. Find the average total number of credits earned by students (the average is over all students and only one value should be computed) who have taken at least one course. Don't count the credits when no grade is given for a class.

20. Compute the GPA for each student from the student's number grades (ngrade) for all the courses he/she has taken (ignore the GPA values already in the students table). The GPA of a student is computed by dividing the sum of his/her number grades by the number of classes he/she has taken and received a non-null number grade. If a student has not received any non-null grade yet, the student's GPA will be null. For each student, the B# of the student and the computed GPA (name column head as cgpa) should be displayed. Display the results in descending non-null GPA values.