**Basic SQL**

1. Find the names of all the instructors from Biology department
2. Find the names of courses in Computer science department which have 3 credits
3. For the student with ID 12345 (or any other value), show all course\_id and title of all courses registered for by the student.
4. As above, but show the total number of credits for such courses (taken by that student). Don't display the tot\_creds value from the student table, you should use SQL aggregation on courses taken by the student.
5. As above, but display the total credits for each of the students, along with the ID of the student; don't bother about the name of the student. (Don't bother about students who have not registered for any course, they can be omitted)
6. Find the names of all students who have taken any Comp. Sci. course ever (there should be no duplicate names)
7. Display the IDs of all instructors who have never taught a couse (Notesad1) Oracle uses the keyword minus in place of except; (2) interpret "taught" as "taught or is scheduled to teach")
8. As above, but display the names of the instructors also, not just the IDs.
9. You need to create a movie database. Create three tables, one for actors(AID, name), one for movies(MID, title) and one for actor\_role(MID, AID, rolename). Use appropriate data types for each of the attributes, and add appropriate primary/foreign key constraints.
10. Insert data to the above tables (approx 3 to 6 rows in each table), including data for actor "Charlie Chaplin", and for yourself (using your roll number as ID).
11. Write a query to list all movies in which actor "Charlie Chaplin" has acted, along with the number of roles he had in that movie.
12. Write a query to list all actors who have not acted in any movie
13. List names of actors, along with titles of movies they have acted in. If they have not acted in any movie, show the movie title as null. (Do not use SQL outerjoin syntax here, write it from scratch.)

**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 which we covered in class (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. (This query is provided in the book/slides.)
9. Update the salary of each instructor to 10000 times the number of course sections they have taught.
10. Create your own query: define what you want to do in English, then write the query in SQL. Make it as difficult as you wish, the harder the better.