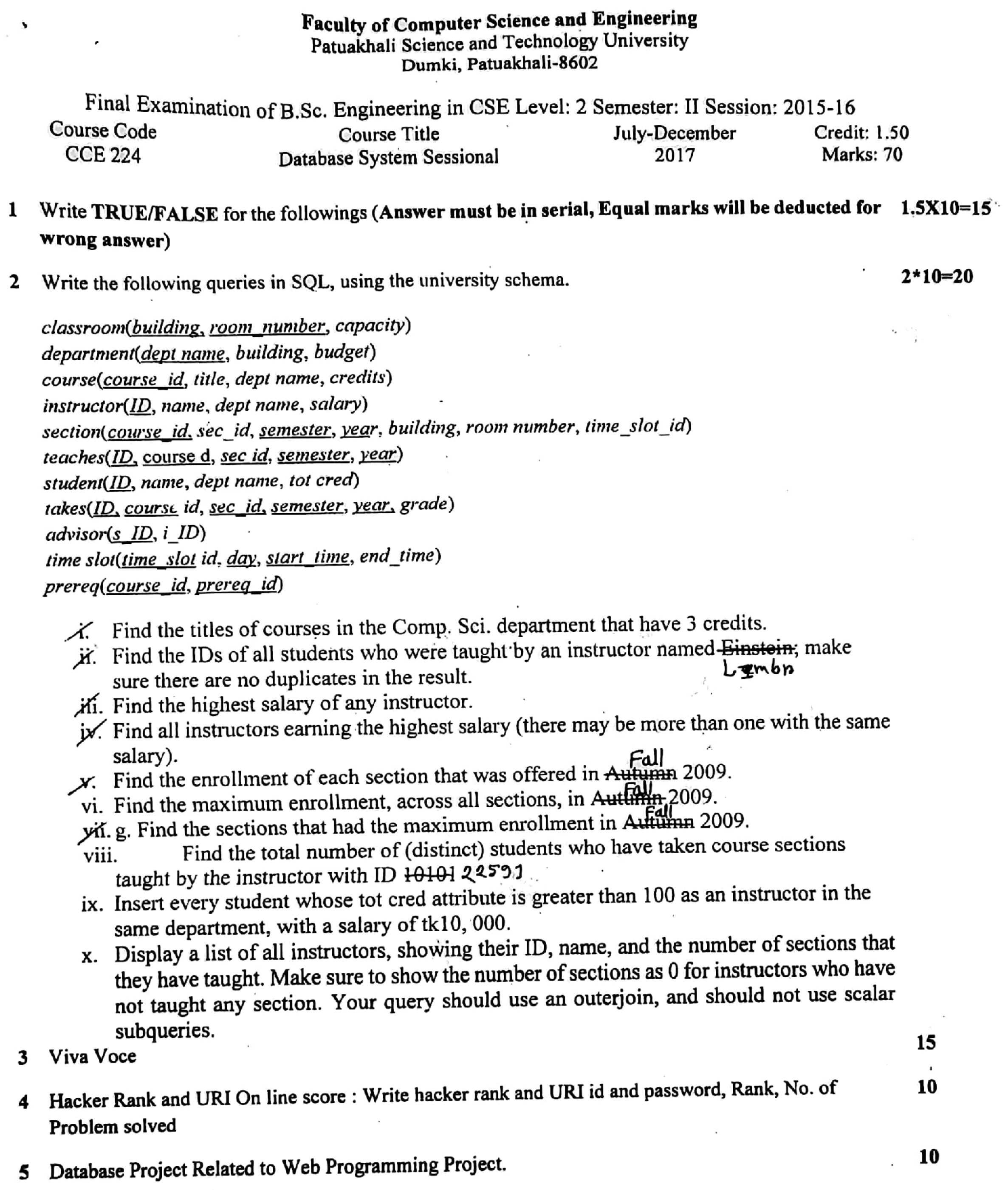
Database System Sessional (CCE 224) - SQL Queries



Faculty of Computer Science and Engineering
Patuakhali Science and Technology University
Dumki, Patuakhali-8602

Final Examination of B.Sc. Engineering in CSE Level: 2 Semester: II Session: 2015-16

Course Code	Course Title	July-December	Credit: 1.50
CCE 224	Database System Sessional	2017	Marks: 70

1 Write **TRUE/FALSE** for the followings (**Answer must be in serial, Equal marks will be deducted for wrong answer)** **1.5X10=15**

2 Write the following queries in SQL, using the university schema. **2*10=20**

classroom(<u>building</u>, <u>room_number</u>, capacity)
department(<u>dept name</u>, building, budget)
course(<u>course_id</u>, title, dept name, credits)
instructor(<u>ID</u>, name, dept name, salary)
section(<u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, building, room number, time_slot_id)
teaches(<u>ID</u>, <u>course d</u>, <u>sec id</u>, <u>semester</u>, <u>year</u>)
student(<u>ID</u>, name, dept name, tot cred)
takes(<u>ID</u>, <u>course id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, grade)
advisor(<u>s_ID</u>, i_ID)
time slot(<u>time_slot id</u>, <u>day</u>, <u>start_time</u>, end_time)
prereq(<u>course_id</u>, <u>prereq_id</u>)

i. Find the titles of courses in the Comp. Sci. department that have 3 credits.
ii. Find the IDs of all students who were taught by an instructor named ~~Einstein~~ Lambo; make sure there are no duplicates in the result.
iii. Find the highest salary of any instructor.
iv. Find all instructors earning the highest salary (there may be more than one with the same salary).
v. Find the enrollment of each section that was offered in ~~Autumn~~ Fall 2009.
vi. Find the maximum enrollment, across all sections, in ~~Autumn~~ Fall 2009.
vii. g. Find the sections that had the maximum enrollment in ~~Autumn~~ Fall 2009.
viii. Find the total number of (distinct) students who have taken course sections taught by the instructor with ID ~~10101~~ 22591.
ix. Insert every student whose tot cred attribute is greater than 100 as an instructor in the same department, with a salary of tk10, 000.
x. Display a list of all instructors, showing their ID, name, and the number of sections that they have taught. Make sure to show the number of sections as 0 for instructors who have not taught any section. Your query should use an outerjoin, and should not use scalar subqueries.

3 Viva Voce **15**

4 Hacker Rank and URI On line score : Write hacker rank and URI id and password, Rank, No. of Problem solved **10**

5 Database Project Related to Web Programming Project. **10**

Schema reminder:

- `classroom(building, room_number, capacity)`
- `department(dept_name, building, budget)`
- `course(course_id, title, dept_name, credits)`
- `instructor(ID, name, dept_name, salary)`
- `section(course_id, sec_id, semester, year, building, room_number, time_slot_id)`
- `teaches(ID, course_id, sec_id, semester, year)`
- `student(ID, name, dept_name, tot_cred)`
- `takes(ID, course_id, sec_id, semester, year, grade)`
- `advisor(s_ID, i_ID)`
- `time_slot(time_slot_id, day, start_time, end_time)`
- `prereq(course_id, prereq_id)`

Questions and Answers

1. Titles of courses in the Comp. Sci. department that have 3 credits:

```
SELECT title
FROM course
WHERE dept_name = 'Comp. Sci.' AND credits = 3;
```

2. IDs of all students who were taught by an instructor named 'Einstein' (no duplicates):

```
SELECT DISTINCT t.ID
FROM takes t
JOIN teaches te ON t.course_id = te.course_id AND t.sec_id = te.sec_id
   AND t.semester = te.semester AND t.year = te.year
JOIN instructor i ON te.ID = i.ID
WHERE i.name = 'Einstein';
```

3. The highest salary of any instructor:

```
SELECT MAX(salary) AS highest_salary
FROM instructor;
```

4. All instructors earning the highest salary:

```
SELECT *
FROM instructor
WHERE salary = (SELECT MAX(salary) FROM instructor);
```

5. The enrollment of each section that was offered in Autumn 2009:

```
SELECT course_id, sec_id, COUNT(ID) AS enrollment
FROM takes
WHERE semester = 'Autumn' AND year = 2009
GROUP BY course_id, sec_id;
```

6. Maximum enrollment, across all sections, in Autumn 2009:

```
SELECT MAX(student_count) AS max_enrollment
FROM (
   SELECT COUNT(ID) AS student_count
   FROM takes
   WHERE semester = 'Autumn' AND year = 2009
   GROUP BY course_id, sec_id
) AS sub;
```

7. Sections that had the maximum enrollment in Autumn 2009:

```
SELECT course_id, sec_id
FROM takes
WHERE semester = 'Autumn' AND year = 2009
GROUP BY course_id, sec_id
HAVING COUNT(ID) = (
   SELECT MAX(student_count)
   FROM (
      SELECT COUNT(ID) AS student_count
      FROM takes
      WHERE semester = 'Autumn' AND year = 2009
      GROUP BY course_id, sec_id
   ) AS sub
);
```

8. Total number of distinct students taught by the instructor with ID '10101':

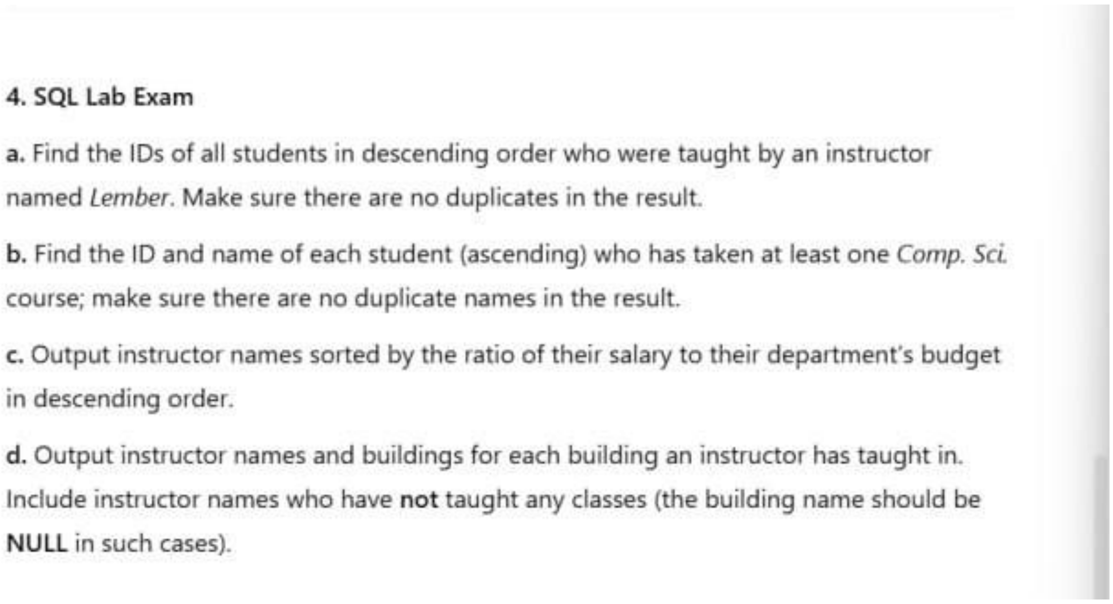
```
SELECT COUNT(DISTINCT t.ID) AS total_students
FROM takes t
JOIN teaches te ON t.course_id = te.course_id AND t.sec_id = te.sec_id
   AND t.semester = te.semester AND t.year = te.year
WHERE te.ID = '10101';
```

9. Insert students with tot_cred > 100 as instructors with salary 10000:

```
INSERT INTO instructor(ID, name, dept_name, salary)
SELECT ID, name, dept_name, 10000
FROM student
WHERE tot_cred > 100;
```

10. All instructors with their ID, name, and number of sections taught (including 0):

```
SELECT i.ID, i.name, COUNT(te.course_id) AS num_sections
FROM instructor i
LEFT JOIN teaches te ON i.ID = te.ID
GROUP BY i.ID, i.name;
```



4. SQL Lab Exam

a. Find the IDs of all students in descending order who were taught by an instructor named *Lember*. Make sure there are no duplicates in the result.

b. Find the ID and name of each student (ascending) who has taken at least one *Comp. Sci.* course; make sure there are no duplicate names in the result.

c. Output instructor names sorted by the ratio of their salary to their department's budget in descending order.

d. Output instructor names and buildings for each building an instructor has taught in. Include instructor names who have **not** taught any classes (the building name should be **NULL** in such cases).

1. IDs of students in descending order who were taught by instructor named 'Lember':

```
SELECT DISTINCT t.ID
FROM takes t
JOIN teaches te ON t.course_id = te.course_id AND t.sec_id = te.sec_id
   AND t.semester = te.semester AND t.year = te.year
JOIN instructor i ON te.ID = i.ID
WHERE i.name = 'Lember'
ORDER BY t.ID DESC;
```

2. ID and name of each student (ascending) who has taken at least one Comp. Sci. course:

```
SELECT DISTINCT s.ID, s.name
FROM student s
JOIN takes t ON s.ID = t.ID
JOIN course c ON t.course_id = c.course_id
WHERE c.dept_name = 'Comp. Sci.'
ORDER BY s.ID ASC;
```

3. Instructor names sorted by the ratio of their salary to department's budget (descending):

```
SELECT i.name
FROM instructor i
JOIN department d ON i.dept_name = d.dept_name
ORDER BY (i.salary * 1.0) / d.budget DESC;
```

4. Instructor names and buildings for each building they have taught in (NULL if none):

```
SELECT i.name, b.building
FROM instructor i
LEFT JOIN teaches te ON i.ID = te.ID
LEFT JOIN section s ON te.course_id = s.course_id AND te.sec_id = s.sec_id
   AND te.semester = s.semester AND te.year = s.year
LEFT JOIN classroom b ON s.building = b.building
GROUP BY i.name, b.building;
```