CSCI 4333 Design of Database Systems
Spring 2025
Section 1 Suggested Solution to Final Examination

[1] (a) For example:

```
SELECT DISTINCT d.deptCode,
       d.deptName AS department,
       COUNT(s.stuId) AS `# enrolled majors`
FROM department AS d LEFT JOIN student AS s ON (d.deptCode = s.major)
GROUP BY d.deptCode, department;
```

(b)

```
SELECT DISTINCT f.facId,
       CONCAT(f.fname, ' ', f.lname) AS faculty,
       COUNT(s.stuId) AS `Number of advisees`
FROM student AS s INNER JOIN faculty AS f ON (s.advisor = f.facId)
       INNER JOIN department AS d ON (f.deptCode = d.deptCode)
WHERE d.schoolCode = 'CSE'
GROUP BY f.facId, faculty
HAVING `Number of advisees` > 1;
```

(c)

```
WITH t1 AS
(SELECT facId
 FROM faculty AS f INNER JOIN class AS c USING (facId)
    INNER JOIN course AS co USING (CourseId)
 WHERE co.rubric = 'CSCI'
 GROUP BY facId
 HAVING COUNT(c.classId) >= 2)
SELECT f.facId,
    CONCAT(f.fname, ' ', f.lname) AS faculty,
    COUNT(s.stuId) AS `number of advisees`
FROM faculty AS f INNER JOIN t1 USING (facId)
    LEFT JOIN student AS s ON (f.facId = s.advisor)
GROUP BY f.facId, faculty;
```

(2)

(a)	F	(b)	T	(c)	F	(d)	F	(e)	F
(f)	F	(g)	T	(h)	T	(i)	F	(j)	F
(k)	F								

(3)

[a] R(A,B,C,D) with {B->D, C->D, D->A}; Canonical cover: same
CK: BC;
Highest NF: 1NF; B->D violates 2NF.

[b] R(A,B,C,D) with {B->AC, A->BD}; Canonical cover: same
CK: [1] A. [2] B
Highest NF: BCNF

[c] R(A,B,C,D) with {B->AC, A->BD, C->D}; Canonical cove: {B->AC, A->B, C->D}
CK: [1] A. [2] B;
Highest NF: 2NF; C -> D violates 3NF.

(4) R(A,B,C,D,E) {A->B, AB->CD, D->AC, C->E}

[a] Canonical Cover: {A->BCD, D->A, C->E}
[b] CK: [1] A, [2] D
[c] Highest NF: 2NF; as C->E violates 3NF.
[d] R1(A,B,C,D) { A->BCD, D->A}
R2(C,E) {C->E}

(5) For example:

```
#      Get HTTP parameters: the ids of two students to be compared.
form = cgi.FieldStorage()
sid1 = form.getfirst('sid1')
sid2 = form.getfirst('sid2')

print('<h3>Two students</h3>')
print('''
<table border='1'>
<tr><th>Id</th><th>Student</th><th>Major department</th>
<th>advisor facId</th><th># classes enrolled</th>
</tr>
''')

#      SQL
query = '''
SELECT s.stuId AS sid,
       CONCAT(s.fName, ' ',s.lName) AS name,
       IFNULL(d.deptName, 'No major') AS major,
    IFNULL(s.advisor, 'No advisor') AS advisor,
       COUNT(e.classId) as numClasses
FROM student AS s LEFT JOIN enroll e ON (s.stuId = e.stuId)
       LEFT JOIN department AS d ON (s.major = d.deptCode)
WHERE (s.stuId = %s OR s.stuId = %s)
GROUP BY sid, name, major, advisor;
'''
cursor.execute(query, (str(sid1), str(sid2)))
for (sid, name, major, advisor, n_classes) in cursor:
       print('    <tr><td>' + str(sid) +
                     '</td><td>' + name + '</td><td>' +
                     major + '</td><td>' + str(advisor) + '</td><td>' +
            str(n_classes) + '</td></tr>')
```

(6) For example:

```
use toyu;

db.student.find(
   { "$and": [
              { "$or": [ {"major": "CINF"}, {"minor": "CINF"}]},
              { "ach": {"$gte": 15}} ] },
   { "stuId": 1, "major":1, "minor": 1,
     "student": {"$concat": ["$fname", " ", "$lname"]},
     "ach credits": "$ach", "_id": 0 }
)

// or simply:
db.student.find(
   { "$or": [ {"major": "CINF"}, {"minor": "CINF"}],
      "ach": {"$gte": 15} },
   { "stuId": 1, "major":1, "minor": 1,
     "student": {"$concat": ["$fname", " ", "$lname"]},
     "ach credits": "$ach", "_id": 0 }
)
```

(7) (a) The second CK is BC.

Given facts [1] and [3], the potential second CK may be B, C or BC. Only having BC as the second CK can produce 20 SK.

(b)

[i]

F1: TutorId -> TutorEMail
F2: StudentId -> StudentEMail

F3: SubjectId -> SubjectName
F4: SubjectName -> SubjectId

[ii]

[1] TutorId, StudentId, SubjectId
[2] TutorId, StudentId, SubjectName

[iii] Highest NF: 1NF as F1 and F2 violates 2NF.