DASC 5333 Database Systems for Data Science
Fall 2024
Section 1 Suggested Solution to Final Examination

[1] (a) For example:

```
SELECT DISTINCT s.stuId,
       CONCAT(s.fname, ' ', s.lname) AS student,
       COUNT(c.classId) AS `# enrolled Fall 2019 classes`
FROM student AS s LEFT JOIN enroll AS e ON (s.stuId = e.stuId)
       LEFT JOIN class AS c ON (e.classId = c.classId AND c.semester = 'Fall' AND
c.year = 2019)
GROUP BY s.stuId, student;
```

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

(3)

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

[b] R(A,B,C,D) with {C->AD, D->AC, B->C}; Canonical form: {C->AD, D->C, B->C};
Highest NF: 2NF; CK: [1] B; C->A violates 3NF

[c] R(A,B,C,D) with {C->AD, A->B, AB->C}; Canonical form: {C->AD, A->BC}
Highest NF: BCNF; CK: [1] A. [2] C

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

(a) Canonical cover: {A->BC, B->A, CD->E}
(b) Candidate Keys: [1] AD, [2] BD; Prime attributes: A, B, D
(c) 1NF, as A->C and CD ->E violate 3NF
(d) R1(A,B,C) {A->BC, B->A } in BCNF, R2(C,D,E) {CD->E} and R3(A,D) {} in BCNF.

(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</th>
<th>advisor name</th><th># classes enrolled</th>
</tr>
''')

#      SQL
query = '''
SELECT s.stuId AS sid,
       CONCAT(s.fName, ' ',s.lName) AS name,
       IFNULL(s.major, 'No major') AS major,
    IFNULL(CONCAT(f.fname, ' ', f.lname), 'No advisor') AS advisor,
       COUNT(e.classId) as numClasses
FROM student AS s LEFT JOIN enroll e ON (s.stuId = e.stuId)
       LEFT JOIN faculty AS f ON (s.advisor = f.facId)
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:

```
db.student.find(
   { "$and": [
             { "lname": {"$in": [ "Hawk", "Zico", "Johnson"]}},
             { "ach": {"$lte": 35}} ] },
   { "stuId": 1, "major":1, "minor": 1,
     "student": {"$concat": ["$fname", " ", "$lname"]},
     "ach credits": "$ach", "_id": 0 }
)

// or simply

db.student.find(
   { "lname": {"$in": [ "Hawk", "Zico", "Johnson"]},
        "ach": {"$lte": 35}  },
   { "stuId": 1, "major":1, "minor": 1,
     "student": {"$concat": ["$fname", " ", "$lname"]},
     "ach credits": "$ach", "_id": 0 }
))
```

(7) (a) The two CK: A and E.

This is because fact [2] indicates that C and D are not in any CK. Fact [3] indicates that B cannot be in any CK since D -> B (If BE is a CK, DE is also a CK, for example).

[i]

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

F3: TutorId, StudentId, SubjectId -> StartDate

[ii] [1] TutorId, StudentId, SubjectId

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