DASC 5333 Database Systems for Data Science
CSCI 4333 Design of Database Systems
Spring 2023
Section 1 Suggested Solution to Final Examination

[1] (a) For example:

```
SELECT DISTINCT d.deptCode,
       d.deptName AS department,
       IFNULL(COUNT(DISTINCT s.stuId), 0) AS `number of enrolled majors`
FROM student AS s INNER JOIN enroll AS e USING (stuId)
       RIGHT JOIN department AS d ON (s.major = d.deptCode)
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 faculty AS f INNER JOIN student AS s ON (s.advisor = f.facId)
       INNER JOIN department AS d USING (deptCode)
WHERE d.schoolCode = 'CSE'
GROUP BY f.facId, faculty
HAVING `Number of advisees` >= 2;
```

(c)

```
WITH t1 AS
(SELECT DISTINCT classId, COUNT(stuId) AS enrollment
 FROM enroll
 GROUP BY classId),
t2 AS (SELECT MAX(enrollment) AS maxEnroll FROM t1)
SELECT  t1.classId,
       CONCAT(co.rubric, ' ', co.number) AS course,
       t1.enrollment
FROM t1 INNER JOIN class AS t USING (classId)
       INNER JOIN course AS co USING (courseId)
       INNER JOIN t2
WHERE t1.enrollment + 2 >= t2.maxEnroll
ORDER BY t1.enrollment DESC;
```

(2)

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

(3)

(a) R(A,B,C,D) with {B->D, C->D, D->A}

CK: [1] BC; prime: B, C; Highest NF: 1NF; B->D and C->D violate 2NF.

(b) R(A,B,C,D) with {B->AC, A->BD}

CK: [1] A, [2] B; prime: A, B; Highest NF: BCNF

(c) R(A,B,C,D) with {B->AC, A->BD, C->D}

CK: [1] A, [2] B; prime: A, B; Highest NF: 2NF; C->D violates 3NF.

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

(a) Canonical cover: {A->BCD, D->A, C->E} (not required)

Candidate Key: [1] A, [2] D; Prime attributes: A, D

(b) 2NF, as C->E violates 3NF

(c) R1(A,B,C,D) {A->BCD, D->A} in BCNF, and R2(C,E) {C->E} in BCNF

(5) For example:

```
print('<h3>Comparing 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, 'Undeclared') AS major,
    IFNULL(s.advisor, 'Not assigned') 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>')

print('</body></html>')
```

(6) For example:

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

(7) (a) BC

(b) For Tutoring(TutorId, TutorEMail, StudentId, StudentEMail, SubjectId, SubjectName):

(i) TutorId -> TutorEMail
StudentId -> StudentEMail
SubjectId -> SubjectName
SubjectName -> SubjectId

(ii) CK: (1) {TutorId, StudentId, SubjectId}, (2) {TutorId, StudentId, SubjectName}

(iii) 1NF since TutorId -> TutorEMail and StudentId -> StudentEMail violate 3NF.