Database Systems
Fall 2025
Section 2 Suggested Solution to Final Examination

[1] (a) For example:

```
SELECT d.deptCode AS `Degree Program`,
       COUNT(s.stuId) AS `Number of majors`
FROM department AS d LEFT JOIN student AS s ON (d.deptCode = s.major)
GROUP BY d.deptCode
HAVING `Number of majors` <= 2
ORDER BY `Number of majors` ASC;
```

(b)

```
SELECT d.deptName AS department,
       COUNT(DISTINCT f.facId) AS `Number of faculty`,
       COUNT(DISTINCT s.stuId) AS `Number of majoring students`
FROM department AS d LEFT JOIN student AS s ON (d.deptCode = s.major)
       LEFT JOIN faculty AS f ON (d.deptCode = f.deptCode)
GROUP BY department;

-- Alternatively.
WITH f AS (SELECT deptCode, COUNT(facId) AS numFaculty FROM faculty GROUP BY
deptCode),
       m AS (SELECT major AS deptCode, COUNT(stuId) AS numMajor FROM student GROUP BY
deptCode)
SELECT d.deptName AS department,
       IFNULL(f.numFaculty, 0) AS `Number of faculty`,
       IFNULL(m.numMajor, 0) `Number of majoring students`
FROM department AS d LEFT JOIN f USING(deptCode)
       LEFT JOIN m USING(deptCode);
```

(c)

```
DELIMITER //

CREATE OR REPLACE FUNCTION n_enrolled_courses(
    sid INT,
    rub varchar(4))
RETURNS INT
READS SQL DATA
BEGIN
   DECLARE count INT DEFAULT 0;

   SELECT COUNT(DISTINCT c.courseId) INTO count
   FROM enroll AS e INNER JOIN class AS c ON (e.classId = c.classId)
        INNER JOIN course AS co ON (c.courseId = co.courseId
            AND co.rubric = rub)
   WHERE e.stuId = sid;

   RETURN count;
END //

DELIMITER ;
```

(2)

(a) F (b) T (c) F (d) F (e) T

(f) F (g) F (h) T (i) T (j) T

(k) F

(3)

(a) R(A,B,C,D) with {A->C, B->C, BC->AD}; Canonical cover (optional): A->C, B->AD
CK: [1] B; prime: B;
Highest NF: 2NF; C->A violates 3NF.

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

(c) R(A,B,C,D) with {A->B, B->C, C->D, D->AB}; Canonical cover (optional): A->B, B->C, C->D, D->A
CK: [1] A, [2] B, [3] C, [4] D; prime: A, B, C, D;
Highest NF: BCNF

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

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

(5) For example:

```
#      Get HTTP parameter: faculty id
form = cgi.FieldStorage()
fid = form.getfirst('fid')

#      SQL
query = '''
SELECT CONCAT(f.fName, ' ', f.LName) AS faculty,
    IFNULL (t1.n_classes, 0) AS n_classes,
    IFNULL (t2.advisees, '') AS advisees
FROM toyu.faculty AS f LEFT JOIN
     (SELECT facId, COUNT(classId) AS n_classes
      FROM toyu.class
      WHERE facId = %s
      GROUP BY facId) AS t1 ON (f.facId = t1.facId)
     LEFT JOIN
     (SELECT advisor AS facId,
        GROUP_CONCAT(CONCAT('<li>', fName, ' ', LName, '</li>')
        SEPARATOR '') AS advisees
      FROM toyu.student
      WHERE advisor = %s
      GROUP BY advisor) AS t2 ON (f.facId = t2.facId)
WHERE f.facId = %s
GROUP BY faculty, n_classes;
```

```
'''

cursor.execute(query, (fid, fid, fid))
(faculty, n_classes, advisees) = cursor.fetchone()
# Generate HTML code.
print('<h3>Faculty information</h3>')
print(f'Faculty Id# {fid} ({faculty}): instructor of {n_classes} classes; advises the
following students:\n<ol>\n{advisees}\n\</ol>')

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

(6) For example:

```
use toyu;

db.faculty.find(
       {  "rank": {"$in": ["Professor", "Assistant Professor"]},
             "deptCode": {"$in": ["CINF", "ITEC", "ARTS"] } },
       {  "facId": 1,
          "faculty": { $concat: ["$fname", " ", "$lname"] },
          "faculty rank": "$rank",
          "deptCode": 1,
          "_id": 0}
)
```

(7) (a) 1 CK: A. One can remove the facts [2] and [3] and draw the same conclusion.

(b) For Participation(StudentId, OrganizationId, RoleId, RoleName StartDate).

(i) StudentId, OrganizationId -> StartDate
RoleId -> RoleName
RoleName -> RoleId

(ii) CK: (1) { StudentId, OrganizationId, RoleId}, (2) { StudentId, OrganizationId, RoleName}

(iii) 1NF since StudentId, OrganizationId -> StartDate violates 2NF.