Database Systems
Spring 2026
Section 2 Suggested Solution to Final Examination

[1] (a) For example:

```
SELECT DISTINCT f.facId,
       CONCAT(f.fname, ' ', f.lname) AS faculty,
       d.deptName AS department,
       COUNT(c.classId) AS `Number of classes taught`
FROM faculty AS f LEFT JOIN department AS d USING (deptCode)
       LEFT JOIN class AS c USING (facId)
GROUP BY f.facId, faculty, department
HAVING `Number of classes taught` >= 2
ORDER BY `Number of classes taught` DESC;
```

(b)

```
SELECT DISTINCT s.stuId,
       CONCAT(s.fname, ' ', s.lname) AS student,
       s.minor
FROM student AS s
WHERE s.major IS NULL
AND s.stuId NOT IN (SELECT DISTINCT stuId FROM enroll);
```

(c)

```
DELIMITER //
CREATE FUNCTION numCommonStudents(
    cid_1 INT,
    cid_2 INT) RETURNS INT
BEGIN
    DECLARE result INT;
    SELECT COUNT(DISTINCT e1.stuId) INTO result
    FROM enroll AS e1 INNER JOIN enroll AS e2 USING (stuId)
    WHERE e1.classId = cid_1
    AND e2.classId = cid_2;

    RETURN result;
END //
DELIMITER ;   RETURN count;

END //

DELIMITER ;
```

(2)

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

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

(k) F

(3)

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

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

(c) R(A,B,C,D) with {A->B, C->BD}
CK: [1] AC; prime: A, C;
Highest NF: 1NF; A->B, C->B and C->D violate 2NF.

(4) For R(A,B,C,D,E) {B->A, C->DE, BA->C, A->C}

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

(5) For example:

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

print('<h3>Department information</h3>')

#      SQL
query = '''
SELECT DISTINCT d.deptName,
    t1.n_majors,
    t1.majors,
    t2.n_minors,
    t2.minors
FROM department AS d,
     (SELECT COUNT(s.stuId) AS n_majors,
      GROUP_CONCAT(CONCAT(s.fname, ' ', s.lname) SEPARATOR ', ') AS majors
      FROM student AS s
      WHERE s.major = %s) AS t1,
     (SELECT COUNT(s.stuId) AS n_minors,
      GROUP_CONCAT(CONCAT(s.fname, ' ', s.lname) SEPARATOR ', ') AS minors
      FROM student AS s
      WHERE s.minor = %s) AS t2
WHERE d.deptCode = %s;
'''
```

```
cursor.execute(query, (str(dc), str(dc), str(dc)))
(department, n_majors, majors, n_minors, minors) = cursor.fetchone()
print(f'The department {department} ({dc}) has \n<ul><li>{n_majors} major students:
{majors}</li>')
print(f'<li>{n_minors} minor students: {minors}</li>\n</ul>\n')
```

(6) For example:

```
use toyu
db.student.find(
       {  "$and": [ {"advisor": {$ne: null}} ,
                    { "ach": {"$gte": 10}},
                    { "ach": {"$lte": 30}}] },
       {  "stuId": 1,
          "fname": 1,
          "lname": 1,
          "advisor": 1,
          "minor": {"$ifNull": ["$minor", "undeclared"]},
          "ach_credits": "$ach",
          "_id": 0    }
```

(7) (a) BCDE

Since, for,

1. There are exactly 2 candidate keys.
2. One of the candidate key is A.
3. B and E are prime attributes. C and D may or may not be prime attributes.
4. There are exactly 17 superkeys.

Only when BCDE is the second candidate can produce 17 superkeys. You do not really need (3) in your deduction.

(b) (i)

DormId -> DormName
DormName -> DormId
StudentId, Semester, Year -> DormId, Room

(ii) CK: {StudentId, Major, Semester, Year}
(iii) 1NF since StudentId, Semester, Year -> DormId, Room violates 2NF.

(c) The word "alienator" is a 9 character anagram of "relational". Other interesting answers will be considered.