Database Systems
Spring 2026
Section 1 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.major
FROM student AS s
WHERE s.minor IS NULL
AND s.stuId NOT IN (SELECT DISTINCT stuId FROM enroll);
```

(c)

```
DELIMITER //
CREATE FUNCTION numCommonClasses(
    sid_1 INT,
    sid_2 INT) RETURNS INT
BEGIN
    DECLARE result INT;
    SELECT COUNT(DISTINCT e1.classId) INTO result
    FROM enroll AS e1 INNER JOIN enroll AS e2 USING (classId)
    WHERE e1.stuId = sid_1
    AND e2.stuId = sid_2;

    RETURN result;
END //
DELIMITER ;
```

(2)

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

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

(k) T

(3)

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

(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, B->AC, AB->D}: canonical cover: {A->BCD, B->A}
CK: [1] A, [2] B; prime: A, B;
Highest NF: BCNF

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

(a) Canonical cover: {B->AD, D->E}
(b) Candidate Key: [1] BC; Prime attributes: B, C
(c) 1NF, as B->A and B->D violate 2NF
(d) R1(A,B,D) {B->AD} in BCNF, R2(D,E) {D->E} in BCNF and R3(B,C) {} 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_faculty
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(f.facId) AS n_faculty
      FROM faculty AS f
      WHERE f.deptCode = %s) AS t2
WHERE d.deptCode = %s;
'''
cursor.execute(query, (str(dc), str(dc), str(dc)))
(department, n_majors, majors, n_faculty) = cursor.fetchone()
```

```
print(f'The department {department} ({dc}) has \n<ul><li>{n_majors} major students:
{majors}</li><li>{n_faculty} faculty members</li>\n</ul>\n')
```

(6) For example:

```
use toyu
db.student.find(
       {  "$and": [ {"advisor": {$ne: null}} ,
                    { "ach": {"$gte": 15}},
                    { "ach": {"$lte": 35}}] },
       {  "stuId": 1,
          "fname": 1,
          "lname": 1,
          "advisor": 1,
          "minor": {"$ifNull": ["$minor", "undeclared"]},
          "ach_credits": "$ach",
          "_id": 0}
)

//  or
db.student.find(
       {  "advisor": {$ne: null} ,
          "ach": {"$gte": 15, "$lte": 35}},
       {  "stuId": 1,
          "fname": 1,
          "lname": 1,
          "advisor": 1,
          "minor": {"$ifNull": ["$minor", "undeclared"]},
          "ach_credits": "$ach",
          "_id": 0}
)
```

(7) (a) BE

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 20 superkeys.

[1] to [3] imply that the second candidate key must contain B and E, such as BE, BEC, BED and BECD. However, only when BE is the second candidate key, there are 20 superkeys.

(b) (i)

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

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

(c) The words "modeling" and "goldmine" are anagrams. Other interesting answers will be considered.