Database Systems
Spring 2026
Section 2 Final Examination

Last Name: _____ **First Name:** _____ **Student Id:** _____

Number: _____

Time allowed: *2 hours.* Total score: 100 points+ 4 Bonus. *Closed* book examination. Two information sheets (letter size, both sides) prepared by yourself are allowed. Answer all questions. Turn in everything: question and answer papers, information sheet and sketch papers. They will be stapled together.

Academic honesty policy will be followed strictly. Cheating will be pursued vigorously and will result in a failing grade of D or below, a permanent academic record, and possibly other more serious penalties.

Use the toyu db in the supplementary sheet for questions on SQL and Python.

(1) [24 points] Construct SQL statements for the following queries. Make sure that your answers generate the exact results, including column names and orders (if ordered).

(a) List all ids, names and department names of all faculty members who taught 2 or more classes. Include the number of classes taught and show the result in the descending order of the number of classes.

```
+-------+---------------+-----------------+--------------------------+
| facId | faculty       | department      | Number of classes taught |
+-------+---------------+-----------------+--------------------------+
|  1011 | Paul Smith    | Computer Science |                        3 |
|  1012 | Mary Tran     | Computer Science |                        2 |
|  1014 | Sharon Mannes | Computer Science |                        2 |
|  1018 | Art Allister  | Arts             |                        2 |
+-------+---------------+-----------------+--------------------------+
4 rows in set
```

(b) List the id, name and minor of every student who has not enrolled in any class and has not declared major in the following manner.

```
+--------+----------------+-------+
| stuId  | student        | minor |
+--------+----------------+-------+
| 100111 | Cathy Johanson | NULL  |
+--------+----------------+-------+
1 row in set
```

(c) Write a SQL stored function numCommonStudents to take two classId (type INT) and return the number of students who have enrolled in both classes. In the example below, both classes 10000 and 10001 have students 100000 and 100001.

```
MariaDB [toyu]> SELECT numCommonStudents(10000, 10001);
+---------------------------------+
| numCommonStudents(10000, 10001) |
+---------------------------------+
|                               2 |
+---------------------------------+
1 row in set (0.001 sec)
```

Your code:

```
DELIMITER //
CREATE FUNCTION numCommonStudents(
    cid_1 INT,
    cid_2 INT) RETURNS INT
BEGIN
…
```

(2) [20 points + 2 Bonus] True or False. *Circle* one choice or *clearly* write 'T' or 'F'.

(a) [T or F] Python is dynamically typed.

(b) [T or F] A useful technique in mitigating SQL injection is to conduct thorough user input validation.

(c) [T or F] If A+ = B+ = C+ in R(A,B,C), then A, B and C are all candidate keys.

(d) [T or F] The decomposition of R(A,B,C,D) {A->B, C->D} into R1(A,B,D) and R2(A,C) is lossless.

(e) [T or F] In SQL, a stored procedure cannot have any UPDATE statement.

(f) [T or F] ACID supports transaction atomicity.

(g) [T or F] In SQL, a trigger can be called directly by a stored function (even though it cannot be called by a stored procedure.)

(h) [T or F] A relation may have no superkey.

(i) [T or F] MongoDB is an example of a NoSQL DBMS.

(j) [T or F] If A->B, B->C, then AD->C.

(k) [T or F] (Bonus) SQL stands for Structured Query Language and the Sun rises in the West.

(3) [9 points] Short Questions. State the candidate keys and the highest normal forms of the following relations. Assume the relations are at least in 1NF.

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

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

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

(4) [9 points] Consider the relation R(A,B,C,D,E) {B->A, C->DE, BA->C, A->C}

(a) Provide a canonical cover.

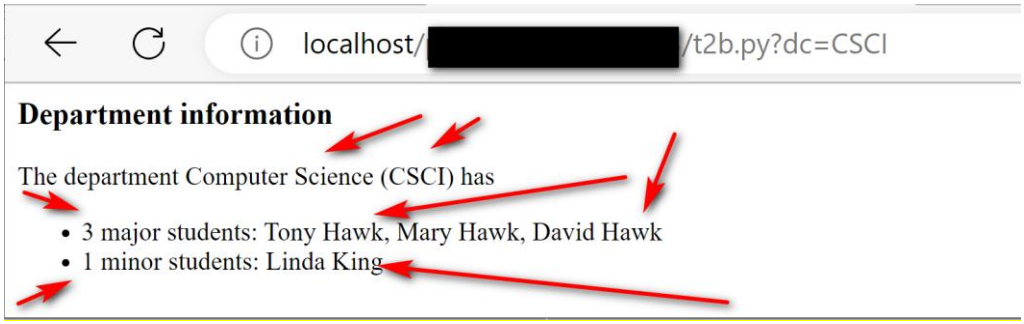
(b) Show all candidate keys.

(c) What is the highest normal form (up to BCNF)? Why?

(d) If it is not in BCNF, can you losslessly decompose R into component relations in BCNF while preserving functional dependencies? If yes, how? If no, why not?

(5) [16 points] Write a Python CGI program, t2b.py, to accept a HTTP Get parameter *dc* (department code) and display the department name, and the numbers and names of students majoring and minoring in it.

For example, for http://.../t2b.py?dc=CSCI, the following result specifies the required output:



There is no need for error checking of the user input parameter *dc*. There is no need for error checking of the input parameter. A skeleton for t2a.py is provided for you. Assume that dbconfig.py and dbconfig.ini are correctly set.

```
from dbconfig import *
import pymysql
import cgi

print("Content-Type: text/html;charset=utf-8")
print()
print ("<html>\n<head></head>\n<body>")

db = get_mysql_param()
cnx = pymysql.connect(user=db['user'], password=db['password'],
                      host=db['host'], database=db['database'])
cursor = cnx.cursor()

# your code here. Write in the back of the previous page if needed.

print('</body></html>')
cursor.close()
cnx.close()
quit()
```

(6) [10 points] Consider the collection ‘student’ in the db ‘toyu’ as stored in MongoDB:

```
[ { _id: ObjectId("63c19f66c1fb90601512c759"), stuId: 100000, fname: 'Tony',
       lname: 'Hawk', major: 'CSCI', minor: 'CINF', ach: 40, advisor: 1011 },
  { _id: ObjectId("63c19f66c1fb90601512c75a"), stuId: 100001, fname: 'Mary',
    lname: 'Hawk', major: 'CSCI', minor: 'CINF', ach: 35, advisor: 1011 },
  { _id: ObjectId("63c19f66c1fb90601512c75b"), stuId: 100002, fname: 'David',
    lname: 'Hawk', major: 'CSCI', minor: 'ITEC', ach: 66, advisor: 1012 },
  { _id: ObjectId("63c19f66c1fb90601512c75c"), stuId: 100003, fname: 'Catherine',
    lname: 'Lim', major: 'ITEC', minor: 'CINF', ach: 20, advisor: null },
  { _id: ObjectId("63c19f66c1fb90601512c75d"), stuId: 100004, fname: 'Larry',
    lname: 'Johnson',  major: 'ITEC', minor: null, ach: 66, advisor: 1017 },
  { _id: ObjectId("63c19f66c1fb90601512c75e"), stuId: 100005, fname: 'Linda',
    lname: 'Johnson', major: 'CINF', minor: 'ENGL', ach: 13, advisor: 1015 },
  { _id: ObjectId("63c19f66c1fb90601512c75f"), stuId: 100006, fname: 'Lillian',
    lname: 'Johnson', major: 'CINF', minor: 'ITEC', ach: 18, advisor: 1016 },
  { _id: ObjectId("63c19f66c1fb90601512c760"), stuId: 100007, fname: 'Ben',
    lname: 'Zico', major: null, minor: null, ach: 16, advisor: null },
  { _id: ObjectId("63c19f66c1fb90601512c761"), stuId: 100008, fname: 'Bill',
    lname: 'Ching', major: 'ARTS', minor: null, ach: 90, advisor: null },
  { _id: ObjectId("63c19f66c1fb90601512c762"), stuId: 100009, fname: 'Linda',
    lname: 'King', major: 'ARTS', minor: 'CSCI', ach: 125, advisor: 1018 },
  { _id: ObjectId("63c19f66c1fb90601512c763"), stuId: 100111, fname: 'Cathy',
    lname: 'Johanson', major: null, minor: null, ach: 0, advisor: 1018 }
]
```

Construct Mongosh query in JS to show the following information of all students with an advisor and with 10 to 30 ach credits: stuId, fname, lname, minor, and ach credits, in the following manner. Answer in the back of the previous page if needed.

Tip: MongoDB support null as a value. Furthermore, the expression: '"xyz": { $ifNull: ["$xyz", "not applicable"] }' returns the value of the field "xyz" is null. Otherwise, it returns "not applicable".

```
[
  {
    stuId: 100005,
    fname: 'Linda',
    lname: 'Johnson',
    advisor: 1015,
    minor: 'ENGL',
    ach_credits: 13
  },
  {
    stuId: 100006,
    fname: 'Lillian',
    lname: 'Johnson',
    advisor: 1016,
    minor: 'ITEC',
    ach_credits: 18
  }
]
```

(7) [12 points + 2 Bonus] (a) [3 points] Four facts are known for R(A,B,C,D,E):

1. There are exactly 2 candidate keys.
2. One of the candidate keys 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.

What is the second candidate key?

(b) [9 points] Consider the relation DormAssignment(StudentId, Major, DormId, DormName, Room, Semester, Year). StudentId is the id of a student with a major, which may be undeclared. A student may have double majors (e.g., S211). A dorm has a unique id, DormId, and an unique DormName. Dorm assignment is semester-based. A row in the table stores the room of the dorm assigned to a student in a particular semester and year. There are separate tables for storing information about students, majors, and dorms. An example of a portion of the table indicating that the student S101 was assigned to three different rooms in three semesters is shown below. Rooms can be shared by students. Make reasonable assumptions.

StudentId	Major	DormId	DormName	Room	Semester	Year
S101	CSCI	D118	Jones Hall	2101	Spring	2022
S101	CSCI	D118	Jones Hall	3313	Fall	2022
S101	CSCI	D31	Roberts Hall	1024	Spring	2023
S211	MATH	D31	Roberts Hall	1024	Spring	2023
S211	CENG	D31	Roberts Hall	1024	Spring	2023

(i) List the functional dependencies representing the specification above.

(ii) What are the candidate keys?

(iii) What is the highest normal form for the DormAssignment relation? Why?

(c) [Bonus: 2 points] What is interesting about the phrase "relational alienator"? Tip: it is not your instructor.