

Hand in a hardcopy of your answers (printed or handwritten). Include your name, and be sure to clearly identify which problems you are answering and which homework the problems are from since problems from multiple homeworks will be collected at the same time. These problems will be collected on Monday 9/21 and Monday 9/28.

You may use the MySQL database to develop and test your queries, but be sure to still think through what each query is doing for yourself in order to practice and build your skills at reading and understanding SQL. Being able to reason through SQL is important for building up complex queries, and is essential for verifying that your queries are correct — they need to work in any case and not just for a particular instance of the database. (The example database provided is not guaranteed to be sufficient to catch all edge cases or reveal common logic bugs.) Instructions for accessing MySQL can be found on Canvas.

(database schema on the next page)

For these problems you'll be working with a grades database containing the following tables:

- **DEPARTMENT** — the academic departments offering courses (identified by a short code, e.g. 'CPSC', 'MATH').
- **COURSE** — the catalog of courses that can be offered, each belonging to a department, with a course number, title, and credit value. A course exists independently of any particular term or section.
- **INSTRUCTOR** — the instructors who teach courses, each affiliated with a department.
- **SECTION** — a specific offering of a course in a given term, taught by a particular instructor (e.g. "CPSC 124, section 1, Fall 2021, taught by Hu"). The A, B, C, etc columns store the points threshold for that letter grade.
- **STUDENT** — the students enrolled in the system, with basic identifying and contact information.
- **GRADECOMP** — the graded components (e.g. "exam1," "homework," "final") that make up a section's grading scheme, each with a maximum point value and a weight toward the final grade. Component names are only meaningful within their own section, not across sections.
- **GRADE** — the actual scores each student earned on each graded component of each section they took.

The database schema:

```
DEPARTMENT(name)
COURSE(courseid, dept, number, title, credits)
INSTRUCTOR(name, dept)
SECTION(sectionid, course, section, term, instructor, A, B, C, D, F)
STUDENT(id, firstname, lastname, email)
GRADECOMP(name, section, maxpts, weight)
GRADE(student, section, component, score)
```

The foreign keys: ($\rightarrow$ reads as "refers to")

- `COURSE.dept` $\rightarrow$ `DEPARTMENT.name`
- `INSTRUCTOR.dept` $\rightarrow$ `DEPARTMENT.name`
- `SECTION.course` $\rightarrow$ `COURSE.courseid`
- `SECTION.instructor` $\rightarrow$ `INSTRUCTOR.name`
- `GRADECOMP.section` $\rightarrow$ `SECTION.sectionid`
- `GRADE.student` $\rightarrow$ `STUDENT.id`
- `GRADE.(component, section)` $\rightarrow$ `GRADECOMP.(name, section)`

INSTRUCTOR

name	dept
Hu	CPSC
Eck	CPSC
King	MATH
Fietkiewicz	CPSC
Bell	MATH
Gassert	MATH

STUDENT

id	firstname	lastname	email
1	Alice	Adams	alice@x.edu
2	Bob	Baker	bob@x.edu
3	Carol	Clark	carol@x.edu
4	David	Diaz	david@x.edu
5	Ellen	Evans	ellen@x.edu
6	Frank	Foster	frank@x.edu

GRADECOMP

name	section	maxpts	weight
exam1	1101	100	20
exam2	1101	100	20
final	1101	100	30
homework	1101	100	10
final	1201	100	30
homework	1201	100	10
homework	1102	100	10
homework	1203	100	10

GRADE

student	section	component	score
1	1101	exam1	98
1	1101	final	97
2	1101	exam2	96
3	1101	final	100
4	1101	homework	70
4	1102	homework	65
4	1201	homework	75
4	1203	homework	60
5	1101	final	70
6	1201	final	95

COURSE

courseid	dept	number	title	credits
200	CPSC	220	Computer Architecture	1.00
300	MATH	130	Calculus I	1.00

SECTION

sectionid	course	section	term	instructor	A	B	C	D	F
1101	200	1	F21	Hu	90	80	70	60	0
1102	200	1	F22	Eck	90	80	70	60	0
1103	200	2	F21	Eck	90	80	70	60	0
1104	200	3	F21	King	90	80	70	60	0
1105	200	4	F21	King	90	80	70	60	0
1106	200	5	F21	King	90	80	70	60	0
1108	200	6	F21	Hu	90	80	70	60	0
1109	200	7	F22	Fietkiewicz	90	80	70	60	0
1201	300	1	F21	Bell	90	80	70	60	0
1202	300	1	F22	Bell	90	80	70	60	0
1203	300	2	F21	Bell	90	80	70	55	0
1206	300	4	F22	Bell	90	80	70	60	0
1204	300	3	F21	Gassert	90	80	70	60	0
1205	300	3	F22	Gassert	90	80	70	60	0
1207	300	4	F21	Gassert	90	80	70	50	0
1301	300	4	F22	King	90	80	70	60	0
1001	300	5	F21	NULL	90	80	70	60	0
1002	300	5	F22	NULL	90	80	70	60	0

Table 1: Data for questions 1–10.

For each of the following, (a) give the actual results (the table) that would be returned by the SQL query and (b) give a brief English description of the results of the query. Use the sample data given on the previous page for part (a), not what is in ex_grades!

1.

```
( SELECT student FROM GRADE WHERE component LIKE 'exam%' AND score >= 95 )
UNION
( SELECT student FROM GRADE WHERE component = 'final' AND score >= 95 )
```
2.

```
( SELECT student FROM GRADE WHERE component LIKE 'exam%' AND score >= 95 )
UNION ALL
( SELECT student FROM GRADE WHERE component = 'final' AND score >= 95 )
```
3.

```
( SELECT GRADE.student
  FROM GRADE JOIN SECTION ON GRADE.section = SECTION.sectionid
                JOIN COURSE ON SECTION.course = COURSE.courseid
  WHERE COURSE.dept = 'CPSC' )
INTERSECT
( SELECT GRADE.student
  FROM GRADE JOIN SECTION ON GRADE.section = SECTION.sectionid
                JOIN COURSE ON SECTION.course = COURSE.courseid
  WHERE COURSE.dept = 'MATH' )
```
4.

```
( SELECT A,B,C,D,F FROM SECTION WHERE instructor = 'Bell' )
INTERSECT ALL
( SELECT A,B,C,D,F FROM SECTION WHERE instructor = 'Gassert' )
```
5.

```
( SELECT instructor FROM SECTION WHERE term = 'F21' )
EXCEPT
( SELECT instructor FROM SECTION WHERE term = 'F22' )
```
6.

```
( SELECT SECTION.instructor
  FROM SECTION JOIN COURSE ON SECTION.course = COURSE.courseid
  WHERE COURSE.dept = 'CPSC' )
EXCEPT ALL
( SELECT SECTION.instructor
  FROM SECTION JOIN COURSE ON SECTION.course = COURSE.courseid
  WHERE COURSE.dept = 'MATH' )
```
7.

```
( SELECT course, instructor FROM SECTION WHERE term = 'F21' )
INTERSECT
( SELECT course, instructor FROM SECTION WHERE term = 'F22' )
```

```
8. ( ( SELECT student
      FROM GRADE
        JOIN SECTION ON GRADE.section = SECTION.sectionid
        JOIN COURSE ON SECTION.course = COURSE.courseid
        WHERE COURSE.dept = 'CPSC' )
  UNION
  ( SELECT student
    FROM GRADE
      JOIN SECTION ON GRADE.section = SECTION.sectionid
      JOIN COURSE ON SECTION.course = COURSE.courseid
      WHERE COURSE.dept = 'MATH' ) )
INTERSECT
( SELECT student FROM GRADE WHERE component = 'final' AND score > 90 )
```

9. Hint: INTERSECT has higher precedence than UNION (and EXCEPT).

```
( SELECT student
  FROM GRADE
    JOIN SECTION ON GRADE.section = SECTION.sectionid
    JOIN COURSE ON SECTION.course = COURSE.courseid
    WHERE COURSE.dept = 'CPSC' )
UNION
( SELECT student
  FROM GRADE
    JOIN SECTION ON GRADE.section = SECTION.sectionid
    JOIN COURSE ON SECTION.course = COURSE.courseid
    WHERE COURSE.dept = 'MATH' )
INTERSECT
( SELECT student FROM GRADE WHERE component = 'final' AND score > 90 )
```

(Comparing this query to the previous one illustrates why it is a good idea to always use parentheses when more than one set operation is present.)

```
10. ( SELECT firstname, lastname FROM STUDENT )
     UNION
     ( SELECT name, dept FROM INSTRUCTOR )
```

Write SQL for each of the following. Use set operations even if you can think of another way to write the query. Additionally use only the constructs discussed in class 9/4-9/16 (SQL basics, joins, aggregation and grouping, subqueries, set operations). In all cases it is fine to also give additional solutions without set operations or which use other constructs not discussed in class, just don't *only* give solutions without set operations or that use other constructs.

Sample data for testing queries is available in the `ex_grades` database in MySQL. Keep in mind, however, that what's in the database may not be sufficient to reveal every subtlety.

11. List all of the departments with affiliated faculty and/or courses.
12. List the names of students who scored 95 or higher on a midterm exam or final, along with the course(s) (department and number) and semester(s) when that occurred.
13. Which departments have an affiliated instructor, but did not have any of their courses offered in a section during F21?
14. What is the average score across every exam grade on record? (include both midterm exams — grade component names starting with “exam” — and finals)
15. List the names of instructors who have taught more CPSC sections than MATH sections, along with how many more CPSC sections they have taught.
16. Which CPSC sections were taught by an instructor who has never also taught a MATH section? List the section ID and instructor for each.

For each of the following, identify whether a set operation (`UNION/INTERSECT/EXCEPT`), a `JOIN` (inner, left, or cross), or a subquery (`IN/EXISTS/NOT EXISTS`) would be the most appropriate approach, according to the best-practices guidelines discussed in class. You do not need to write the SQL, just name the approach and briefly justify your choice.

17. List all of the departments with affiliated faculty and/or courses.
18. For each section, list the course's department alongside the department of the instructor who teaches it.
19. List the names of students who have earned a grade in some CPSC course.
20. Which instructors have never taught a section of a MATH course?
21. Which courses have been offered in both fall and spring terms?