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 9/14 and Monday 9/21.

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)`

Sample data for testing other 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.

For each of the following, give a brief English description of the result of the query. Describe the result rather than the working of the query. For example, "names and salaries of employees working in department 5" rather than "first find the rows of `EMPLOYEE` where the department number is 5, then get the first name, last name, and salary".

1. ```
   SELECT COUNT(*)
   FROM SECTION
   WHERE term = 'F22'
   ```

2. ```
   SELECT dept, COUNT(*), COUNT(title)
   FROM COURSE
   GROUP BY dept
   ```

3. ```
   SELECT COUNT(DISTINCT SECTION.instructor)
   FROM SECTION JOIN COURSE ON SECTION.course = COURSE.courseid
   WHERE COURSE.dept = 'CPSC'
   ```

4. ```
   SELECT COURSE.dept, SECTION.term, SUM(COURSE.credits)
   FROM SECTION JOIN COURSE ON SECTION.course = COURSE.courseid
   GROUP BY COURSE.dept, SECTION.term
   ```

5. ```
   SELECT SECTION.sectionid, GRADE.component,
          AVG(GRADE.score), MIN(GRADE.score), MAX(GRADE.score)
   FROM GRADE JOIN SECTION ON GRADE.section = SECTION.sectionid
   WHERE SECTION.term = 'F21'
   GROUP BY SECTION.sectionid, GRADE.component
   ```

6. ```
   SELECT instructor, COUNT(*)
   FROM SECTION
   GROUP BY instructor
   HAVING COUNT(*) > 3
   ```

7. ```
   SELECT SECTION.instructor, SECTION.term,
          AVG(GRADE.score/GRADECOMP.maxpts*100)
   FROM GRADE
     JOIN SECTION ON GRADE.section = SECTION.sectionid
     JOIN GRADECOMP ON GRADE.component = GRADECOMP.name AND
                       GRADE.section = GRADECOMP.section
   WHERE GRADE.component = 'final'
   GROUP BY SECTION.instructor, SECTION.term
   HAVING AVG(GRADE.score/GRADECOMP.maxpts)*100 >= 70
   ```
   ```

For each of the pairs of queries in this section, decide whether the two queries always produce the same result set — and if not, briefly explain why.

8. (a)
```
SELECT dept, COUNT(*)
FROM COURSE
GROUP BY dept
```

   (b)
```
SELECT dept, COUNT(title)
FROM COURSE
GROUP BY dept
```

9. (a)
```
SELECT dept, COUNT(*)
FROM COURSE
GROUP BY dept
```

   (b)
```
SELECT dept, COUNT(courseid)
FROM COURSE
GROUP BY dept
```

10. (a)
```
SELECT section, COUNT(component)
FROM GRADE
GROUP BY section
```

    (b)
```
SELECT section, COUNT(DISTINCT component)
FROM GRADE
GROUP BY section
```

11. (a)
```
SELECT section, AVG(score)
FROM GRADE
WHERE score >= 60
GROUP BY section
```

    (b)
```
SELECT section, AVG(score)
FROM GRADE
GROUP BY section
HAVING AVG(score) >= 60
```

12. (a)
```
SELECT dept, AVG(credits)
FROM COURSE
GROUP BY dept
```

    (b)
```
SELECT dept, SUM(credits) / COUNT(*)
FROM COURSE
GROUP BY dept
```

Write SQL for each of the following. Use only the constructs discussed in class 9/4-9/11 (SQL basics, joins, aggregation and grouping) and follow best practices for the use of those constructs even if you could write the query another way.

13. How many students are in the database?
14. How many instructors are there in each department?
15. For each department, how many courses are offered — including any courses that have no department listed?
16. What are the lowest- and highest-numbered courses offered in each department?
17. How many different students have taken a MATH course?
18. Which students have earned grades in courses from more than one department?
19. For each student, how many different sections have they taken, and how many total grade entries do they have recorded across all components?
20. Which instructors have taught more than two different courses in the same term? Report the instructor, the term, and the number of courses taught for each.
21. For section 5137, what is each student's average percentage score across their graded components? Treat all the components as having equal weight, so the percentage score on a component is the score divided by the maximum points possible.
22. For section 5137, what is each student's overall weighted percentage grade? The weighted percentage takes into account that not all of the grade components count the same, and is computed as the sum of the percentage scores times their weights divided by the sum of the weights. (This is the standard formula for average when all of the weights are 1.)