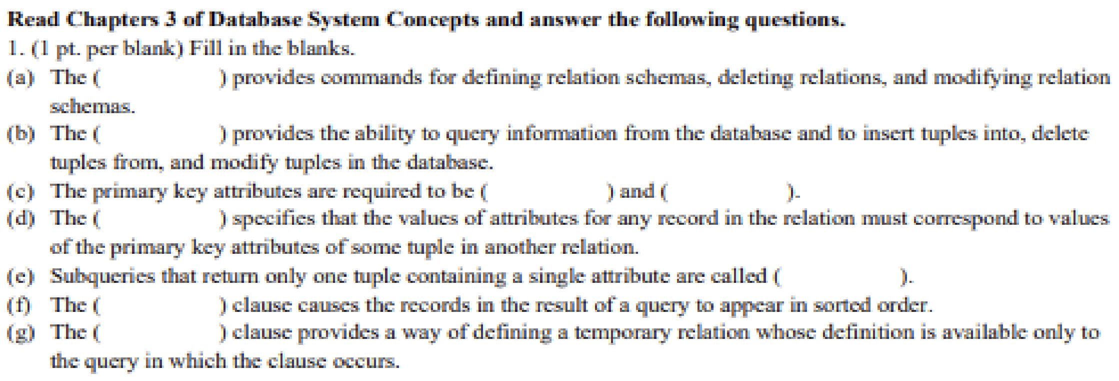
Homework Assignment 2

Due: 11:59PM April 10, 2025

1.



Read Chapters 3 of Database System Concepts and answer the following questions.

1. (1 pt. per blank) Fill in the blanks.

(a) The ( ) provides commands for defining relation schemas, deleting relations, and modifying relation schemas.

(b) The ( ) provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.

(c) The primary key attributes are required to be ( ) and ( ).

(d) The ( ) specifies that the values of attributes for any record in the relation must correspond to values of the primary key attributes of some tuple in another relation.

(e) Subqueries that return only one tuple containing a single attribute are called ( ).

(f) The ( ) clause causes the records in the result of a query to appear in sorted order.

(g) The ( ) clause provides a way of defining a temporary relation whose definition is available only to the query in which the clause occurs.

Answer 1.

a. DDL
b. DML
c. Unique, not null
d. Foreign key
e. Scalar subquery
f. ORDER BY
g. WITH

2. Short-answer questions.

(a) (Exercise problem 3.6) The SQL **LIKE** operator is case sensitive (in most systems), but the **LOWER**() function on strings can be used to perform case-insensitive matching. Show how to write a query that finds departments whose names contain the string "sci" as a substring, regardless of the case.

(b) (Exercise problem 3.20) Show that, in SQL, **<> ALL** is identical to **NOT IN**.

(c) (Exercise problem 3.19) List two reasons why null values might be introduced into the database.

Answer 2

a.

```
 SELECT dept_name
FROM department
 WHERE LOWER(name) LIKE '%sci%';
```

b.

<> ALL checks if a value is different from all values in a list. This is the same as NOT IN, which returns true if the value is not in the list. So, both expressions work the same in most cases.

c.

The data is not known yet (e.g., phone number not given).
The data does not apply (e.g., no middle name).

3. (2 pt. each; based on Exercise problems 3.9, 3.10, 3.16, and 3.17) Consider the relational database of Figure 3.19, where the primary keys are underlined. Given an expression in SQL for each of the following

(a) Find the ID of each employee who does not work for "First Bank Corporation".

SQL Query:

```
SELECT ID
FROM works
WHERE company_name <> 'First Bank Corporation';
```

(b) Find the ID, name, and city of residence of each employee who works for "First Bank Corporation" and earns
more than $10,000.

SQL Query:

```
SELECT e.ID, e.person_name, e.city
FROM employee e, works w
WHERE e.ID = w.ID
  AND w.company_name = 'First Bank Corporation'
  AND w.salary > 10000;
```

(c) Find the ID of each employee who earns more than every employee of "Small Bank Corporation".

SQL Query:

```
SELECT ID
FROM works
WHERE salary > ALL (
  SELECT salary
  FROM works
  WHERE company_name = 'Small Bank Corporation'
);
```

(d) Assume that companies may be located in several cities. Find the name of each company that is located in every city in which "Small Bank Corporation" is located. Your query should run on MySQL.

SQL Query:

```
SELECT DISTINCT c1.company_name
FROM company c1
WHERE NOT EXISTS (
  SELECT city
  FROM company
  WHERE company_name = 'Small Bank Corporation'
  AND city NOT IN (
    SELECT city
    FROM company c2
    WHERE c1.company_name = c2.company_name
  )
`
```

(e) Find the name of the company that has the most employees (or companies, if there is a tie).

SQL Query:

```
SELECT company_name
FROM works
GROUP BY company_name
HAVING COUNT(*) >= ALL (
  SELECT COUNT(*)
  FROM works
  GROUP BY company_name
);
```

(f) Find the name of each company whose employees earn a higher salary on average, than the average salary at“First Bank Corporation”.

SQL Query:

```
SELECT company_name
FROM works
GROUP BY company_name
HAVING AVG(salary) > (
  SELECT AVG(salary)
  FROM works
  WHERE company_name = 'First Bank Corporation'
);
```

(g) Modify the database so that the employee whose ID is ‘12345’ now lives in a city called “Newtown”.

SQL Query:

```
UPDATE employee
SET city = 'Newtown'
WHERE id = '12345';
```

(h) Find ID and name of employee who lives in the same city as the location of the company for which the employee works.

SQL Query:

```
SELECT e.ID, e.person_name
FROM employee e
JOIN works w ON e.ID = w.ID
JOIN company c ON w.company_name = c.company_name
WHERE e.city = c.city;
```

(i) Find ID and name of each employee who earns more than the average salary of all employees of her or his company.

SQL Query:

```
SELECT e.ID, e.person_name
FROM employee e
JOIN works w1 ON e.ID = w1.ID
WHERE w1.salary > (
  SELECT AVG(w2.salary)
  FROM works w2
  WHERE w1.company_name = w2.company_name
);
```

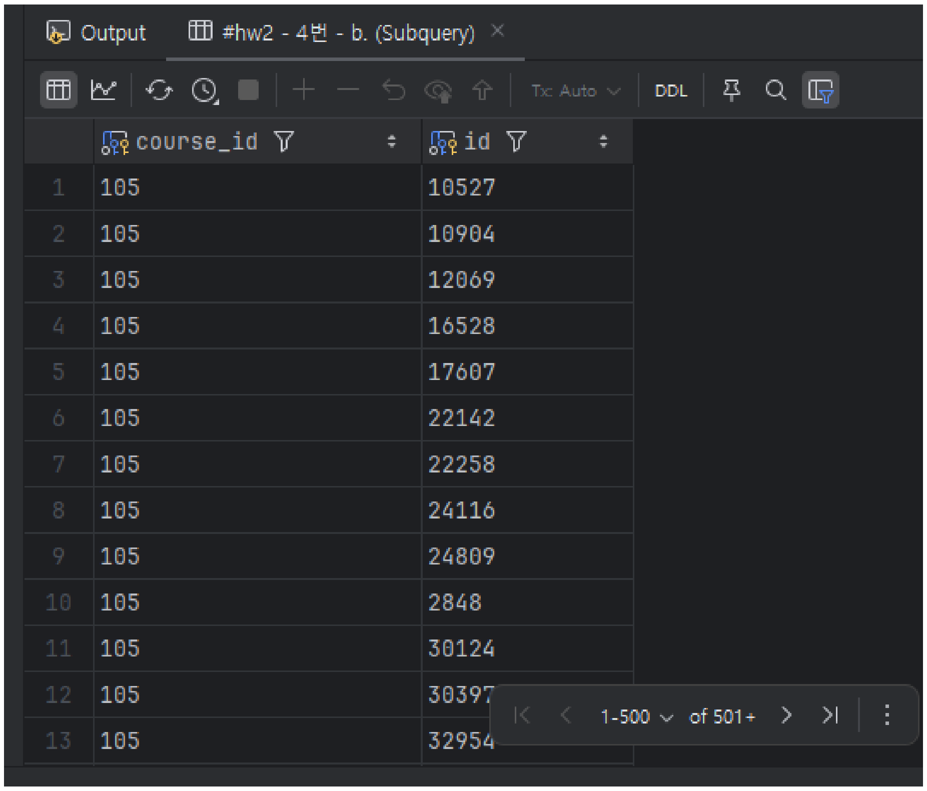
4. (3 pt. each) Find the answers to the following questions and provide the SQL queries showing how you find

them. All queries should be complete to obtain the listed answers solely by themselves.
(a) Write a query that **lists up all classes** that have been open in the university, together **with the number of students** who were in each class. More specifically, enumerate all the *course IDs, section IDs, years, and semesters*, along with the *number of students who took each of the classes*.
Hint: you may want to come up with a result that starts as below
Answer 4(a)

```
 SELECT course_id, sec_id, semester, year, (SELECT COUNT(takes.course_id)
                                    FROM takes
                                    WHERE section.course_id = takes.course_id  AND
section.sec_id = takes.sec_id AND section.semester = takes.semester) AS num_students
FROM section
ORDER BY course_id;
```

(b) (Exercise problem 3.26) For each student who has retaken a course at least twice (*i.e.*, the student has taken the course at least three times), show the course ID and the student's ID. Please display your results in order of course ID and do not display duplicate rows.
Answer 4(b):

[Query Result]



	course_id	id
1	105	10527
2	105	10904
3	105	12069
4	105	16528
5	105	17607
6	105	22142
7	105	22258
8	105	24116
9	105	24809
10	105	2848
11	105	30124
12	105	30397
13	105	32954

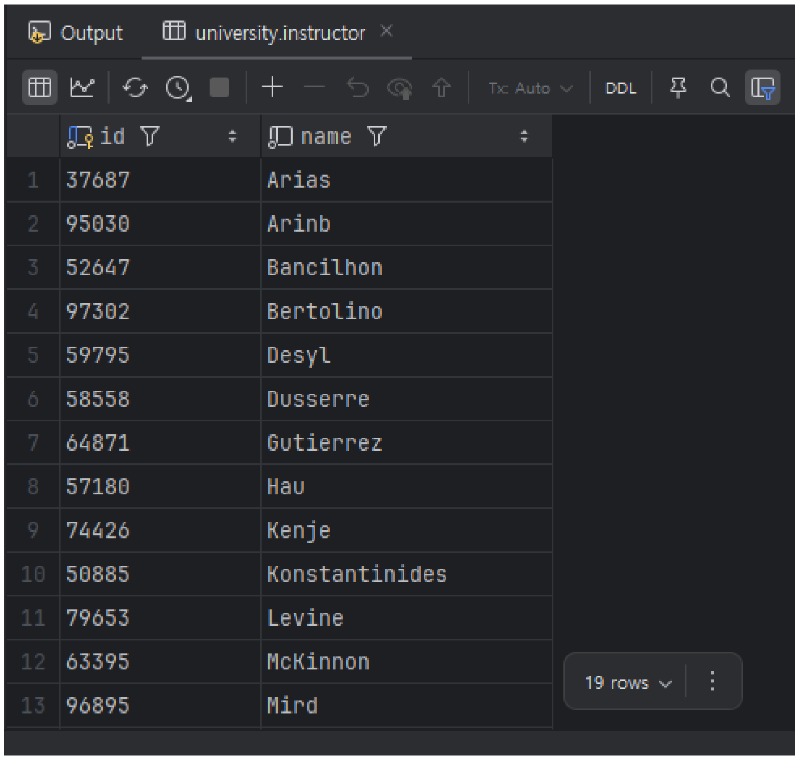
[SQL query answer]

```
SELECT DISTINCT
D.course_id, D.id
FROM (SELECT id,
course_id,
COUNT(course_id) AS
num_take
      FROM takes
      GROUP BY id,
course_id
      ) AS D
WHERE D.num_take >= 2
ORDER BY D.course_id ;
```

(c) (Based on exercise problem 3.31) Find the ID and name of each instructor who has never given an A grade in

any course s/he has taught. Order result by name.
Answer 4(c):

[Query Result]



Output | university.instructor

	id	name
1	37687	Arias
2	95030	Arinb
3	52647	Bancilhon
4	97302	Bertolino
5	59795	Desyl
6	58558	Dusserre
7	64871	Gutierrez
8	57180	Hau
9	74426	Kenje
10	50885	Konstantinides
11	79653	Levine
12	63395	McKinnon
13	96895	Mird

19 rows

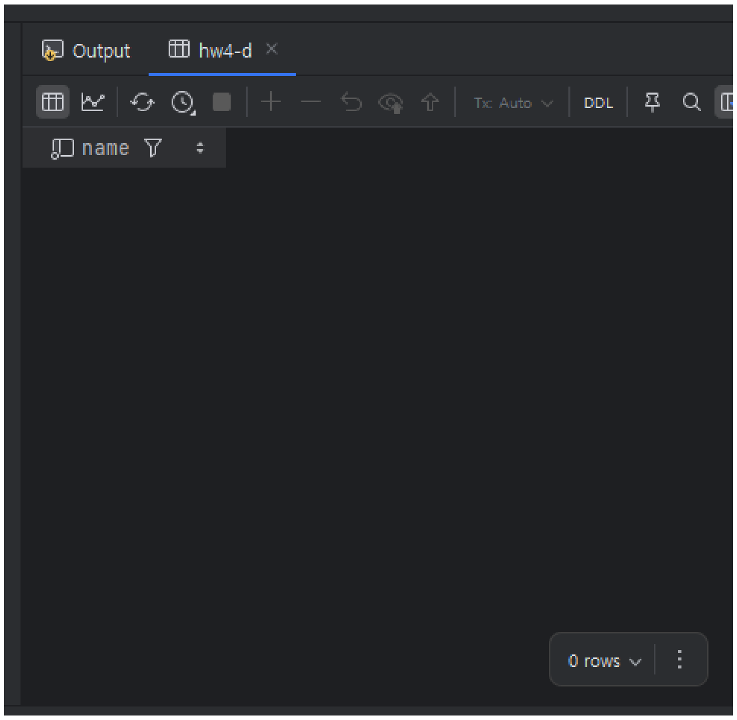
[SQL query answer]

```
SELECT instructor.id,
instructor.name
FROM instructor
WHERE NOT EXISTS (
    SELECT 1
    FROM teaches
            JOIN takes ON
teaches.course_id = takes.course_id
    WHERE teaches.id = instructor.id
      AND takes.grade LIKE '%A%'
)
ORDER BY instructor.name;
```

(d) (Based on exercise problem 3.28) Find the names of the instructors who teach every course taught in his/her

department. Order result in reverse alphabetical order.
Answer 4(d):

[Query Result]	**[SQL query answer]**

 Output hw4-d name 0 rows	SELECT DISTINCT s.name
	FROM instructor AS s
	WHERE NOT *EXISTS*(SELECT course_id
	FROM course
	WHERE course.dept_name = s.dept_name AND course_id NOT IN (
	SELECT t.course_id
	FROM teaches AS t
	WHERE t.ID = s.ID
	)
	);

(e) (Exercise problem 3.30) Consider the following SQL query on the university schema:
SELECT *AVG*(salary) - (*SUM*(salary)/*COUNT*(*))
FROM instructor;
We might expect that the result of this query is zero since the average of a set of numbers is defined to be the sum of the numbers divided by the number of numbers. Indeed, this is true for the example *instructor* relation in Figure 2.1. However, there are other possible instances of that relation for which the result would NOT be zero. Give one such instance, and explain why the results would not be zero.

Answer 4(e)

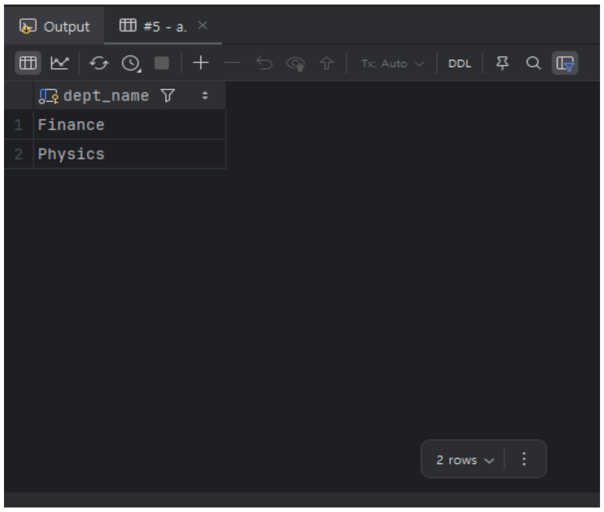
If some instructors have NULL in their salary, AVG(salary) and SUM(salary) ignore those rows, but COUNT(*) includes them. This causes SUM(salary)/COUNT(*) to be smaller than AVG(salary), so the result is not zero.

5. Launch and access the MySQL databases distributed with the class Docker image. Below uses the "**university**" database (NOT *university_small*), which shares the same schemas with the database used in the lectures but contains a larger set of data records collected within a different period of time.

(3 pt. each) Find the answers to the following questions and provide the SQL queries showing how you find them. All queries should be complete to obtain the listed answers solely by themselves.

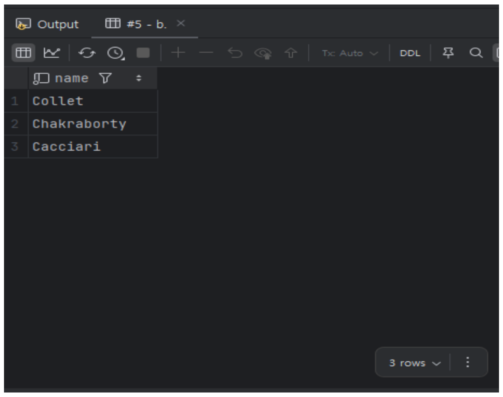
(a) Find the names of departments whose budget is higher than that of *Psychology*. List them in alphabetic order.
Answer

[Query Result]	**[SQL query answer]**

 dept_name
1 Finance
2 Physics
2 rows
	SELECT s.dept_name
	FROM department AS s,
	(SELECT budget
	FROM department
	WHERE dept_name = 'Psychology')
	AS p
	WHERE s.budget > p.budget
	ORDER BY s.dept_name;

(b) List the names of the students in the *Geology* department whose name starting with 'C'.
Answer:

[Query Result]	**[SQL query answer]**

 name
1 Collet
2 Chakraborty
3 Cacciari
3 rows
	SELECT name
	FROM student
	WHERE (dept_name = 'Geology' AND name LIKE 'C%');

(c) Write a query that counts the **number of students for each department** and sort the results in **descending order of the student counts**. *Hint: the head of the query result looks like the following:*

Answer:

[SQL query answer]

```
SELECT dept_name, COUNT(*) AS num_students
FROM student
GROUP BY dept_name
ORDER BY num_students DESC ;
```

(d) (Exercise problem 3.22) Rewrite the WHERE clause
WHERE UNIQUE (SELECT title FROM course)
without using the UNIQUE construct.

Answer:

[SQL query answer]

```
WHERE (
  SELECT COUNT(DISTINCT title)
  FROM course
) = 1
```