**Homework Assignment 2**

Due: 11:59PM April 7, 2023

1. Fill in the blanks.

(a) The (Data-definition language (DDL)) provides commands for defining relation schemas, deleting relations, and modifying relation schemas.

(b) The (Data-manipulation language (DML)) 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 (nonnull) and (unique).

(d) The (foreign-key) 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 (scalar subqueries).

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

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

2. 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.

WHERE LOWER(Department) LIKE '%sci%'

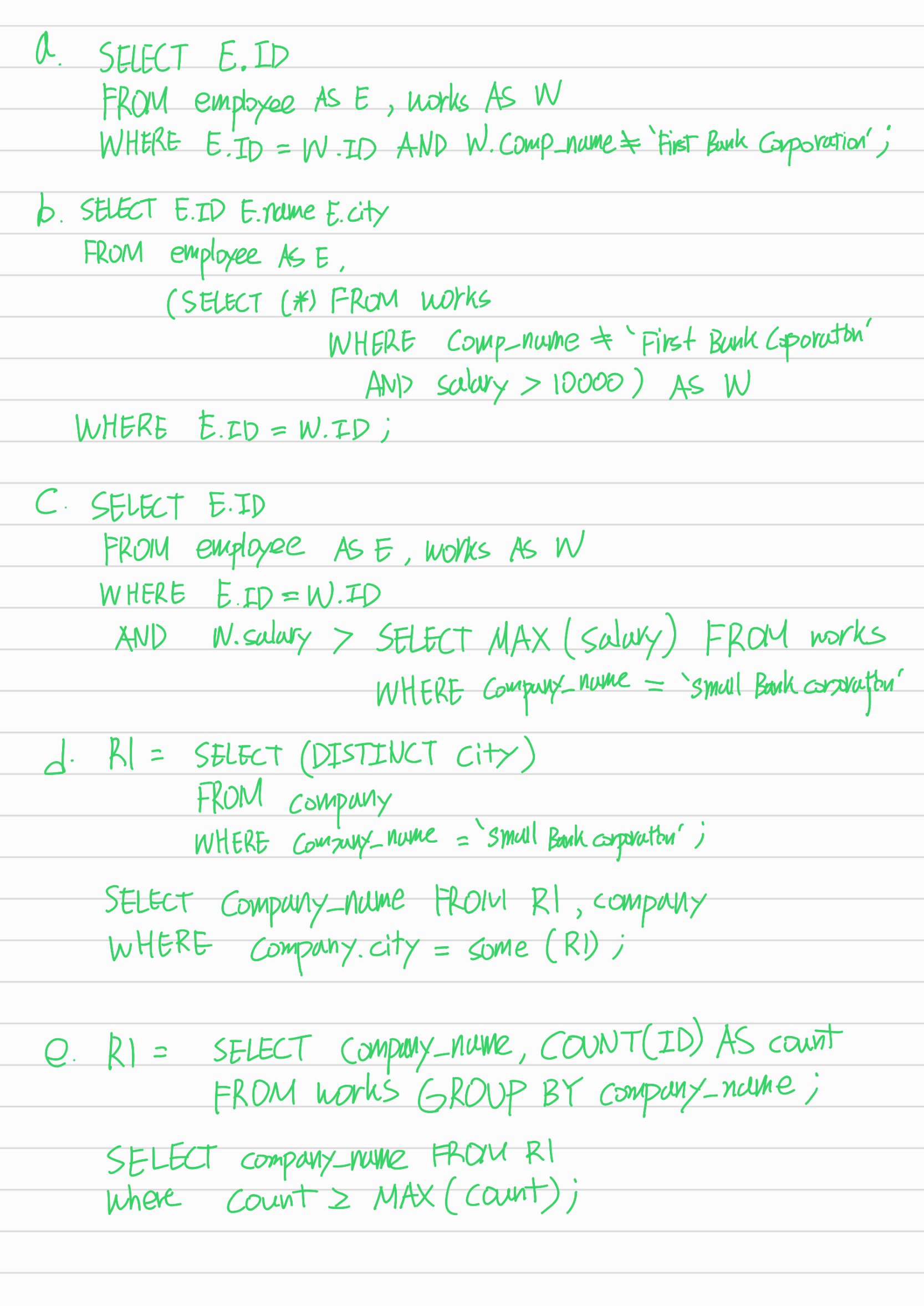
3. Show that, in SQL, <> ALL is identical to NOT IN

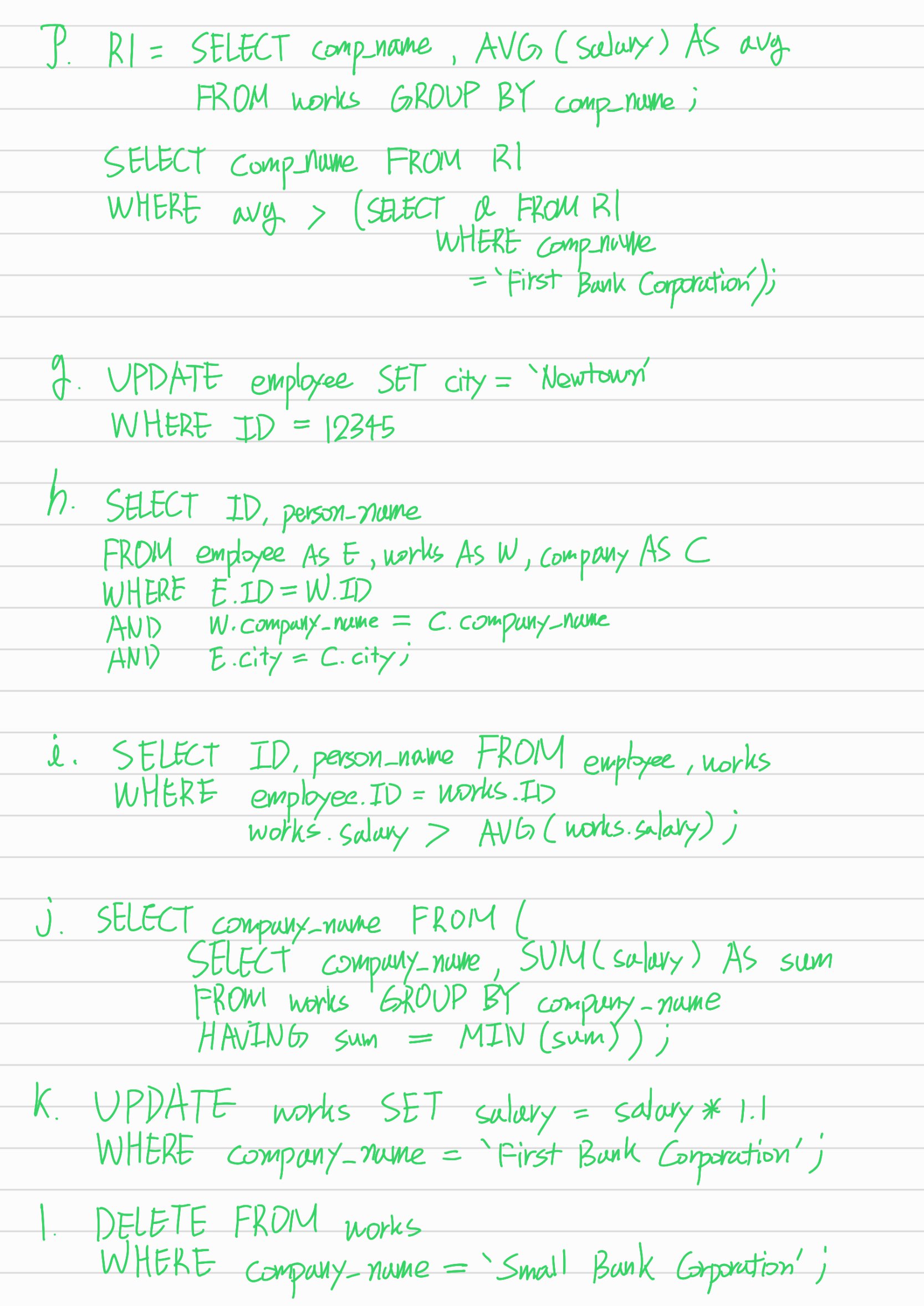
The <> ALL operator returns true if the value on the left side of the operator is not equal to any value returned by the subquery on the right side. Similarly, the NOT IN operator returns true if the value on the left side is not existed in the set of values returned by the subquery on the right side. Since, “not equal to any value” and “not existed in the set of values” are the equivalent propositions, these two operators are identical.

4. List two reasons why null values might be introduced into the database

One of the reasons why null values might be introduced is to denote absence of value in the database. For example, in the student relation, new students can be added but their department has not been assigned yet. In this example, null value may be useful to denote the absence of department which is a missing information.

Another reason is to represent the inapplicability of the attribute. For example, in a database containing a table of people with a column for spouse name, a null value can be used in this column for people who are not married yet.

5. Consider the relational database of Figure 3.19, where the primary keys are underlined. Given an expression in SQL for each of the following queries.



6. 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**

Answer:



SQL Query to obtain your answer:

use university;

select *count*(t.course\_id)   
from teaches as t, teaches as s   
where t.semester = "Fall" and   
 s.semester = "Spring" and   
 t.course\_id = s.course\_id;

**b**

Answer:



SQL Query to obtain your answer:

use university;

select *count*(distinct (title)) from course;

**c**

Answer:



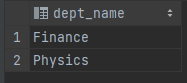
SQL Query to obtain your answer:

use university;

select dept\_name, *round*(*avg*(salary)/12) as avg\_montly\_salary   
from instructor   
group by dept\_name   
having dept\_name = "Cybernetics";

**d**

Answer:



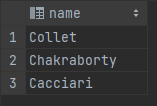
SQL Query to obtain your answer:

use university;

select dept\_name   
from department   
where budget > (select budget   
 from department   
 where dept\_name = "Psychology")   
Order By dept\_name;

**e**

Answer:



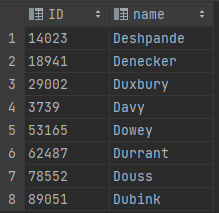
SQL Query to obtain your answer:

use university;

select name   
from student   
where dept\_name = "Geology" and   
 name like "C%";

**f**

Answer:



SQL Query to obtain your answer:

use university;

select s.ID, s.name   
from student as s   
where s.dept\_name = "History" and   
 s.name like "D%" and   
 (select *count*(t.course\_id)   
 from takes as t, course as c   
 where t.ID = s.ID and   
 c.title like "%Music%" and   
 t.course\_id = c.course\_id < 5);

g

Answer:



SQL Query to obtain your answer:

use university;

select \*   
from student   
where *length*(name) > 11 and   
 (dept\_name = "Physics" Or dept\_name = "Comp. Sci.");

h

Answer:



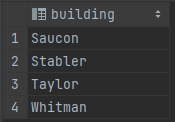
SQL Query to obtain your answer:

use university;

select *count*(ID)   
from student   
where dept\_name = "Comp. Sci." and   
 tot\_cred > (select *min*(tot\_cred)   
 from student   
 where dept\_name = "English");

i

Answer:



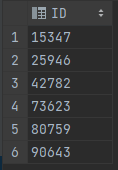
SQL Query to obtain your answer:

use university;

select b.building   
from (select building, *sum*(capacity) as sum   
 from classroom group by building having sum > 100) as b;

j

Answer:



SQL Query to obtain your answer:

use university;

select i.ID from instructor as i   
 where *exists*(select \* from teaches as t   
 where t.ID = i.ID and   
 year < 2003) and   
 not *exists*(select \* from teaches as t   
 where t.ID = i.ID and   
 year >= 2003);

k

Query:

use university;

select dept\_name, *count*(ID) as num\_students   
from student group by dept\_name   
 order by num\_students desc;

l

Answer:

use university;

select \* from course   
 where title in   
 (select title from course group by title

having *count*(title) = 1);

7. 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**

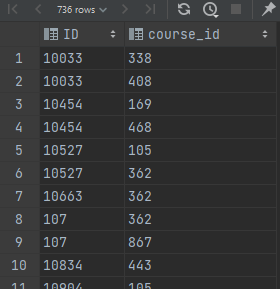
Query:

use university;

select course\_id, sec\_id, semester, year, *count*(ID) as num\_student   
from takes group by course\_id, sec\_id, semester, year;

B

Answer:



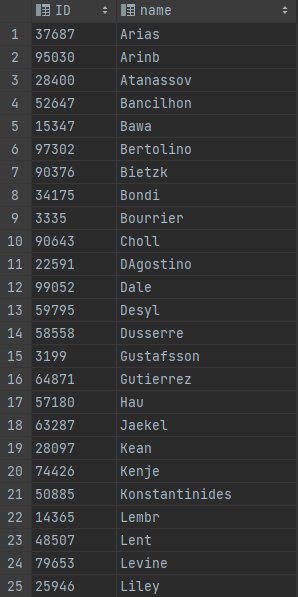
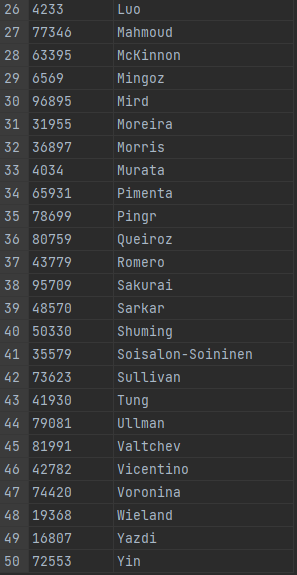
SQL Query to obtain your answer:

use university;

select ID, course\_id from takes group by ID, course\_id having *COUNT*(\*) >= 2;

C

Answer:

SQL Query to obtain your answer:

use university;

select i.ID, i.name from instructor as i where i.ID not in (select teach.ID from teaches as teach, takes as take  
 where teach.course\_id = take.course\_id and  
 teach.sec\_id = take.sec\_id and  
 teach.semester = take.year and  
 teach.year = take.year and  
 take.grade like "%A%")  
 order by i.name;

D

Answer:



SQL Query to obtain your answer:

use university;

select name  
from instructor as i  
where not *exists* ((select course\_id   
 from course as c   
 where c.dept\_name = i.dept\_name)   
 except  
 (select distinct course\_id   
 from teaches as t   
 where t.ID = i.ID))  
order by name desc;

E

The result of the query would not be zero if the instructor table contains any NULL values in the salary column. This is because the AVG function ignores NULL values when calculating the average, while

the SUM and COUNT functions do not.

For example, consider an instance of the instructor table with the following data:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **name** | **dept\_name** | **salary** |
| 1 | John | CS | 80000 |
| 2 | Jane | CS | NULL |

In this case, the result of the query would be calculated as follows:

The AVG(salary) would return 80000, because it only considers the non-NULL salary value for John.

The SUM(salary) would return 80000, because it sums all salary values including NULLs (which are treated as 0).

The COUNT(\*) would return 2, because it counts the number of records regardless of whether they contain NULLs.

Therefore, the result of the query would be 80000 - (80000/2) = 40000, which is not zero.