**Homework Assignment 2**

Due: 11:59PM April 7, 2023

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

1. Data-definition language (DDL)
2. Data-manipulation language (DML)
3. Unique, never changed
4. foreign key references
5. scalar subqueries
6. order by
7. with

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

* SELECT \* FROM department WHERE LOWER(dept\_name) LIKE LOWER('%sci\_%') SELECT \* FROM department WHERE LOWER(dept\_name) LIKE LOWER('%sci\_%')

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

* The ALL statement returns data if all elements of a given subquery satisfy the condition. therefore, <>ALL returns data if at least one element of the subquery does not satisfy the condition. And The NOT IN statement returns data if not included in the given subquery. if we use <>ALL with equal (=) operator, it means any of element in the subquery are not equal to the data. so, in that case, the meaning of both statement is the same. (예시필요함?)

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

1. Null values need for a value to put in its place if the data does not be available or may be unknown at the time of data entry.
2. it also need if data is transformed from one data type to another, null values may be introduced if the data cannot be accurately converted.

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.

텍스트이(가) 표시된 사진

자동 생성된 설명

a. Find the ID of each employee who does not work for “First Bank Corporation.”

* SQL Query: SELECT *ID* FROM *employee* e INNER JOIN *work* w ON e.*ID* = w.*ID* 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 *ID, person\_name, city* FROM *employee* e

INNER JOIN *work* w ON e.*ID* = w.*ID*

WHERE *company\_name* = ‘First Bank Corporation’ AND salary > 10000;

c. Find the ID of each employee who earns more than every employee of “Small Bank Corporation.”

* SQL Query:

SELECT *ID* FROM *employee* e

INNER JOIN *work* w ON e.*ID* = w.*ID*

ORDER BY salary DESC

LIMIT 1 ;

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:

WITH target (city) AS (SELECT city FROM company WHERE company\_name = ‘Small Bank Corporation’)

SELECT company\_name FROM company AS c, target AS t WHERE c. city = t. city;

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

* SQL Query:

WITH c(company\_name, count) AS

(SELECT w.company \_name, count(\*) FROM employees e

INNER JOIN work w on e. ID = w.ID

GROUP BY w.company \_name)

SELECT company\_name FROM c ORDER BY count DESC LIMIT 1;

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:

WITH c(avg) AS

(SELECT AVG(salary) FROM work

GROUP BY company \_name

HAVING company\_name = ‘First Bank Corporation’)

SELECT company\_name FROM (SELECT company\_name, AVG(salary) AS avg\_salary FROM works GROUP BY company\_name) AS cc, c WHERE cc.avg\_salary > c.avg;

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’

WEHRE 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:

WITH emp(ID, name, comp, city) AS

(SELECT *ID, name, company\_name, city* FROM *employee* e

INNER JOIN *work* w ON e.*ID* = w.*ID*)

SELECT *ID, name, city* FROM emp e

INNER JOIN *company* c ON e.*comp* = 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:

WITH c(cname, avg) AS

(SELECT company\_name, AVG(salary) FROM work

GROUP BY company \_name

HAVING company\_name = ‘First Bank Corporation’)

SELECT *ID, person\_name* FROM *employee* e

INNER JOIN *work* w ON e.*ID* = w.*ID*, c

WHERE company\_name = cname AND salary > avg

j. Find the company that has the smallest payroll (sum of all salary in a company)

* SQL Query:

WITH c(cname, sum) AS

(SELECT company\_name, SUM(salary) FROM work

GROUP BY company \_name)

SELECT company\_name FROM c ORDER BY sum LIMIT 1;

k. Given all employees of “First Bank Corporation” a 10 percent raise

* SQL Query:

UPDATE works

SET salary = salary\*1.1

WHERE company\_name = ‘First Bank Corporation’

l. Delete all tuples in the works relation for employees of “Small Bank Corporation”

* SQL Query:

DELETE FROM works WHERE company\_name = ‘Small Bank Corporation’;

WHERE e.

6. (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 number of all courses offered in Fall and that of Spring, respectively.

* answer: 51, 49
* SQL Query to obtain your answer:

WITH falls(cnt) AS

(SELECT COUNT(\*) FROM section WHERE semester = 'Fall')

SELECT AVG(cnt) AS FALL, COUNT(\*) AS SPRING

FROM section, falls WHERE semester = 'Spring'

b. How many unique course names (titles) are among the courses offered by the university?

* answer: 133
* SQL Query to obtain your answer:

SELECT COUNT(DISTINCT title) FROM course

c. What is the average monthly salary of the instructors in the Cybernetics department? Round the answers at the second decimal place, if necessary.

* answer: 96346.57
* SQL Query to obtain your answer:

SELECT ROUND(AVG(salary), 2) from instructor

GROUP BY dept\_name

HAVING dept\_name = 'Cybernetics'

d. Find the names of departments whose budget is higher than that of Psychology. List them in alphabetic order

* answer: Finance, Physics
* SQL Query to obtain your answer:

WITH psybud(budget) AS

(SELECT budget FROM department WHERE dept\_name = 'Psychology')

SELECT \* FROM department AS d, psybud WHERE d.budget > psybud.budget

ORDER BY dept\_name

e. List the names of the students in the Geology department whose name starting with ‘C’

* answer: Collet, Chakraborty, Cacciari
* SQL Query to obtain your answer:

SELECT \* FROM student WHERE dept\_name = 'Geology' AND name LIKE 'C%'

f. (Exercise problem 3.29) Find the ID and name of each History student whose name begins with the letter ‘D’ and who has not taken at least five Music courses

* answer:

3739,Davy

14023,Deshpande

78552,Douss

53165,Dowey

89051,Dubink

29002,Duxbury

* SQL Query to obtain your answer:

WITH music\_std(ID, music\_count) AS

(SELECT ID, COUNT(\*)

FROM (select ID, title from takes

inner join course c on takes.course\_id = c.course\_id

)as m

where m.title like '%music%'

GROUP BY ID)

SELECT s.ID, name FROM student s

inner join music\_std m on s.ID = m.ID

WHERE dept\_name = 'History' AND name LIKE 'D%'

AND music\_count < 5;

g. Find all Physics and Comp. Sci. students whose name is longer than 11 characters.

* answer: Rosenkrantz, Albuquerque, Srivastava, Westervelt, Krishnakumar
* SQL Query to obtain your answer:

SELECT \* FROM student

WHERE dept\_name IN ('Comp. Sci.', 'Physics') AND name LIKE '\_\_\_\_\_\_\_\_\_\_%'

h. Find the number of Comp. Sci. students total credits greater than that of AT LEAST ONE student in the English departments. Find all Physics and Comp. Sci. students whose name is longer than 11 characters.

* answer: Rosenkrantz, Albuquerque
* SQL Query to obtain your answer:

SELECT \* FROM student

WHERE dept\_name = 'Comp. Sci.'

AND tot\_cred > SOME(

SELECT tot\_cred FROM student WHERE dept\_name = 'English')

AND name LIKE '\_\_\_\_\_\_\_\_\_\_\_%';

i. Which of the university buildings can accommodate more than 100 people? Hint: see the sum of classroom capacities

* answer: Saucon, Stabler, Taylor, Whitman
* SQL Query to obtain your answer:

SELECT building FROM classroom

GROUP BY building

HAVING SUM(capacity) >= 100

j. Find all instructor IDs who had taught until 2003 but had not taught after 2003. Hint: Attribute teaches.ID is the instructor ID.

* answer:99052,90643,81991,80759,79081,77346,74420,73623,6569,43779,42782,41930,36897,34175,3199,28400,28097,25946,22591,19368,15347
* SQL Query to obtain your answer:

SELECT DISTINCT i.ID FROM instructor i

INNER JOIN teaches t on i.ID = t.ID WHERE year <= 2003

k. 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:

* Query (you do not need to submit your query result):

SELECT dept\_name, COUNT(\*) FROM student GROUP BY dept\_name

ORDER BY COUNT(\*) DESC

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

* answer:

SELECT title

FROM course

GROUP BY title

HAVING COUNT(title) = 1

7. (4 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

* Query (you do not need to submit your query result):

WITH num\_std(id, sem, yer, cnt) AS(

SELECT course\_id, semester, year, COUNT(distinct ID) FROM takes GROUP BY course\_id, year, semester)

SELECT course\_id, sec\_id, semester, year, cnt AS num\_student

FROM section s

INNER JOIN num\_std n ON s.course\_id = n.id AND s.semester = n.sem AND s.year = n.yer

ORDER BY course\_id, year DESC;

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:

362,16480

362,16969

362,27236

362,39925

362,39978

362,44881

362,49611

362,5414

362,69581

362,9993

* SQL Query to obtain your answer:

SELECT course\_id, ID

FROM takes GROUP BY ID, course\_id

HAVING COUNT(\*) >= 3

ORDER BY 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: no instructor did like that.
* SQL Query to obtain your answer: SELECT ID, name from

(SELECT i.ID, name, course\_id FROM teaches t

INNER JOIN instructor i ON t.ID = i.ID) as inst,

(SELECT course\_id FROM takes

WHERE course\_id NOT IN(

SELECT course\_id

FROM takes

WHERE grade LIKE 'A%'

* )) as t
* WHERE inst.course\_id = t.course\_id

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: DAgostino
* SQL Query to obtain your answer:

SELECT name FROM

(SELECT dept\_name, count(\*) AS cnt FROM course GROUP BY dept\_name) AS dept,

(SELECT name, dept\_name, COUNT(\*) AS cnt FROM teaches t

INNER JOIN instructor i ON t.ID = i.ID

GROUP BY name, dept\_name) AS inst

WHERE inst.dept\_name = dept.dept\_name AND inst.cnt = dept.cnt

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:
* SQL Query to obtain your answer: