Homework Assignment 4

Due: 11:59PM, May 08, 2025

1.(a)

(a) (Exercise 4.1) Consider the following SQL query that seeks to find a list of titles of all courses taught in Spring
2017 along with the name of the instructor.
SELECT *name*, *title*
FROM *instructor* **NATURAL JOIN** *teaches* **NATURAL JOIN** *section* **NATURAL JOIN** *course*
WHERE *semester* = 'Spring' **AND** *year* = 2017;
What is wrong with this query?

Answer 1(a).

(NATURAL JOIN) automatically performs equi-join between attributes with the same name. Semester and year attributes all exist in the instructor, teachers, section, and course tables, but the meanings may be different. In this case**, mis-matching** may occur **between attributes with different meanings**, resulting in incorrect results. Therefore, it is safe to explicitly use the (JOIN ...ON) or (JOIN ... USING) syntax in this situation.

1.(b)

(b) (Exercise 4.16) Write an SQL query using the university schema to find the ID of each student who has never
taken a course at the university. Do this using no subqueries and no set operations (use an outer join).

Answer 1(b).

```
SELECT A.ID
FROM student A
LEFT JOIN takes B
ON A.ID = B.ID
WHERE  B.ID is NULL;
```

1.(c)

(c) (Exercise 4.17) Express the following query in SQL using no subqueries and no set operations.
SELECT *ID*
FROM *student*
EXCEPT
SELECT *s_id*
FROM *advisor*
WHERE *i_ID* **IS NOT NULL**;

Answer 1(c).

```
SELECT ID
FROM student A
LEFT JOIN advisor B
ON A.ID = B.s_id
WHERE B.s_id is NULL
```

1.(d)

(d) (Exercise 4.20) Show how to define a view *tot_credits*(*year*, *num_credits*), giving the total number of credits
taken in each year.

Answer 1(d).

```
CREATE VIEW tot_credits(year, num_credits) AS
SELECT year, SUM(credits)
FROM takes NATURAL JOIN course
GROUP BY year;
```

1.(e)

(e) (Exercise 4.21) For the view that you have defined in the previous problem (Problem 4(d)), explain why the database system would not allow a tuple to be inserted into the database through this view.

Answer 1(e).

Using the aggregation classroom (SUM, GROUP BY) (the next operation is impossible because we do not know which original table column should be inserted during INSERT)

2(a).

(a) Write an SQL query that returns the number of distinct *countries* per *continent*.

Answer 2(a).

```
SELECT continent, COUNT(DISTINCT NAME)
FROM country
GROUP BY continent;
```

2(b).

(b) Write an SQL query that returns the number of distinct *countries* that use Language = 'English'.

Answer 2(b).

```
 SELECT COUNT(DISTINCT C.CODE)
FROM Country AS C
JOIN (
SELECT CountryCode
FROM CountryLanguage AS CL
WHERE CL.Language = "English"
) AS CL2 ON (C.code = CL2.countryCode);
```

2(c).

(c) Write an SQL query that returns the number of distinct *countries* that use Language = 'English' per *continent*.

Answer 2(c).

```
SELECT COUNT(DISTINCT NAME)
FROM country
JOIN countryLanguage ON (country.Code = countryLanguage.CountryCode)
WHERE CountryLanguage.Language = 'English'
GROUP BY Continent
```

2(d).

(d) Write an SQL query that lists distinct country names, within which one of the languages is used by more than 99 percent of time.

Answer 2(d).

```
SELECT DISTINCT c.NAME
FROM country AS C
JOIN (
SELECT CountryCode
FROM countryLanguage AS CL
WHERE CL.percentage >= 99.0
) AS CL2
ON (C.code = CL2.countryCode);
```

2(e).

(e) Write an SQL query that increase *populations* of *countries* (*country.Population*) whose *population* is over 100,000,000 by 1% and all others by 2%.

Answer 2(e).

```
UPDATE country
SET population = CASE
   WHEN population >100000000 THEN (population * 1.01)
ELSE (population * 1.02)
END;
```

2(f).

(f) Write an SQL query that modifies the table schema of *city* and adds a new attribute named *Remark* whose type is VARCHAR(512).

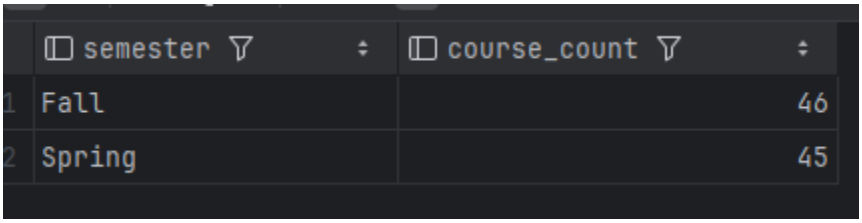
Answer 2(f).

```
ALTER TABLE city ADD COLUMN Remark VARCHAR(512);
```

3.

(a) Find the number of all courses offered in Fall and that of Spring, respectively.

Answer:



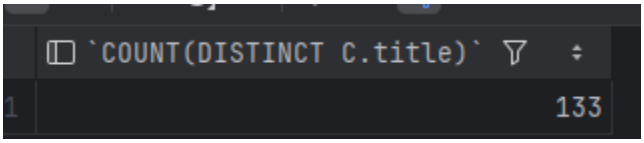
	semester	course_count
1	Fall	46
2	Spring	45

SQL Query to obtain your answer:

```
SELECT semester, COUNT(DISTINCT course_id) AS course_count
FROM teaches
WHERE semester IN ('Fall', 'Spring')
GROUP BY semester;
```

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

Answer:



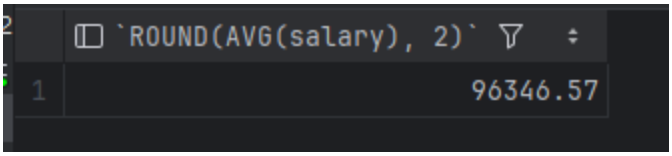
	`COUNT(DISTINCT C.title)`
1	133

SQL Query to obtain your answer:

```
SELECT COUNT(DISTINCT C.title)
FROM course AS C;
```

(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:

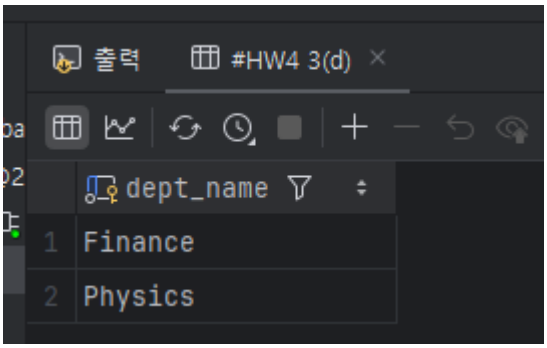


SQL Query to obtain your answer:

```
SELECT ROUND(AVG(salary), 2)
FROM instructor
WHERE dept_name = 'Cybernetics';
```

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

Answer:

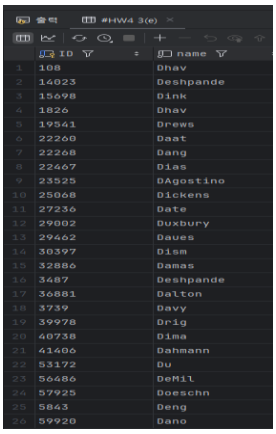


SQL Query to obtain your answer:

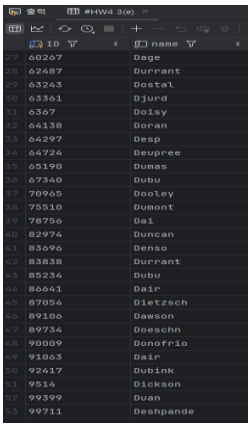
```
SELECT D.dept_name
FROM department AS D
        JOIN department AS P
            ON P.dept_name = 'Psychology' AND D.budget > P.budget
ORDER BY D.dept_name;
```

(e) Find the ID and name of each *History* student whose name begins with the letter 'D' and who has not taken at least three *Psychology* courses.

Answer:



	ID	name
1	108	Dhav
2	14023	Deshpande
3	15698	Dink
4	1826	Dhav
5	19541	Drews
6	22260	Daat
7	22268	Dang
8	22467	Dias
9	23525	DAgostino
10	25068	Dickens
11	27236	Date
12	29002	Duxbury
13	29462	Daves
14	30397	Dism
15	32886	Damas
16	3487	Deshpande
17	36881	Dalton
18	3739	Davy
19	39978	Drig
20	40738	Dima
21	41406	Dahmann
22	53172	Du
23	56486	DeMil
24	57925	Doeschn
25	5843	Deng
26	59920	Dano



	ID	name
27	60267	Dage
28	62487	Durrant
29	63243	Dostal
30	63361	Djurd
31	6367	Doisy
32	64138	Doran
33	64297	Desp
34	64724	Deupree
35	65190	Dumas
36	67340	Dubu
37	70965	Dooley
38	75510	Dumont
39	78756	Dai
40	82974	Duncan
41	83696	Denso
42	83838	Durrant
43	85234	Dubu
44	86641	Dair
45	87054	Dietzsch
46	89106	Dawson
47	89734	Doeschn
48	90009	Donofrio
49	91063	Dair
50	92417	Dubink
51	9514	Dickson
52	99399	Duan
53	99711	Deshpande

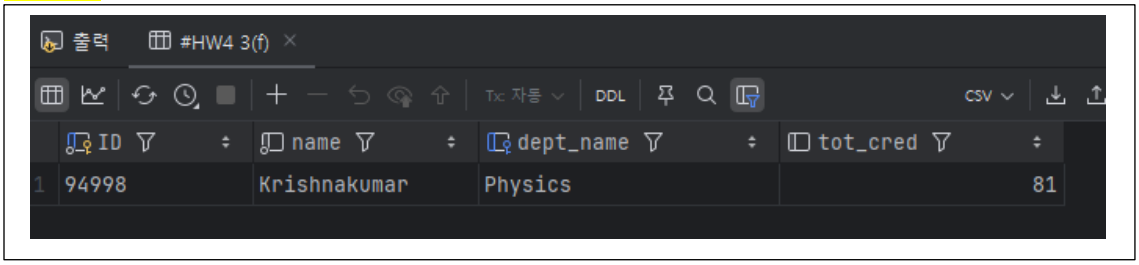
total: 53 students

SQL Query to obtain your answer:

```
 SELECT b.ID, b.name
FROM student AS b
LEFT JOIN (
    SELECT takes.ID, COUNT(*) AS takes_num
    FROM takes
    JOIN course ON takes.course_id = course.course_id
    WHERE course.dept_name = 'Psychology'
    GROUP BY takes.ID
) AS c
ON b.ID = c.ID
WHERE (c.takes_num < 3 OR c.takes_num IS NULL)
  AND b.name LIKE 'D%'
  AND b.dept_name = 'History';
```

(f) Find all *Physics* and *Comp. Sci.* students whose name is longer than 11 characters.

Answer:

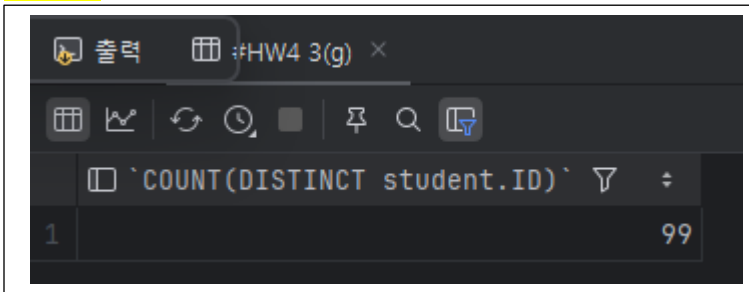


SQL Query to obtain your answer:

```
SELECT *
FROM student
WHERE (dept_name = 'Physics' OR  dept_name = 'Comp. Sci.') AND CHAR_LENGTH(name)
> 11 ;
```

(g) Find the number of *Comp. Sci.* student total credits greater than that of AT LEAST ONE student in the *English* department.

Answer:

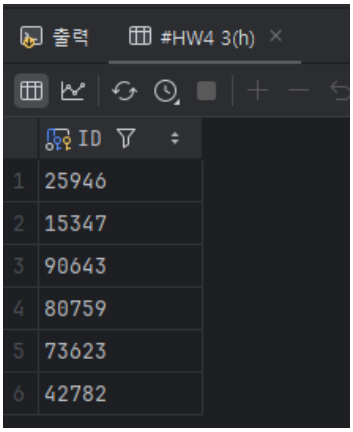


SQL Query to obtain your answer:

```
SELECT  COUNT(DISTINCT student.ID)
FROM student
WHERE student.dept_name = 'Comp. Sci.' AND
    student.tot_cred
> SOME (SELECT  student.tot_cred
      FROM student
      WHERE dept_name = 'English');
```

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

Answer:



SQL Query to obtain your answer:

```
SELECT A.ID
FROM teaches AS A
LEFT JOIN (SELECT * FROM teaches WHERE year > 2003) AS B ON A.ID = B.ID
WHERE B.id IS NULL;
```

4.

Given the following tables, evaluate the result of the following queries. Remember that the output of a query is a table.

Question 4(a),

(a) SELECT AGENT_NAME FROM AGENTS
WHERE COMMISSION > SOME(
SELECT COMMISSION FROM AGENTS
WHERE WORKING_AREA = 'Bangalore'
);

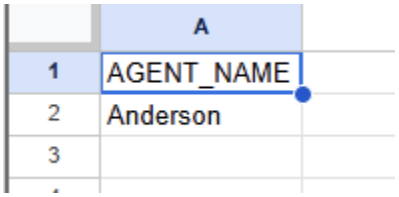
Answer 4(a)

	A
1	AGENT_NAME
2	Ivan, McDen

Question 4(b),

```
SELECT AGENT_NAME FROM AGENTS
WHERE COMMISSION IN(
SELECT COMMISSION FROM AGENTS
WHERE WORKING_AREA = 'Brisban'
);
```

Answer 4(b)

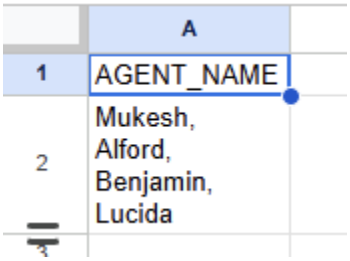


	A
1	AGENT_NAME
2	Anderson
3	

Question 4(c),

```
SELECT AGENT_NAME FROM AGENTS
WHERE COMMISSION < ALL(
SELECT COMMISSION FROM AGENTS
WHERE WORKING_AREA = 'London'
);
```

Answer 4(c)

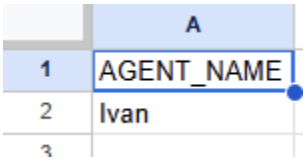


	A
1	AGENT_NAME
2	Mukesh, Alford, Benjamin, Lucida
3	

Question 4(d),

```
SELECT AGENT_NAME
FROM AGENTS AS A
WHERE WORKING_AREA = 'San Jose' AND
EXISTS (SELECT * FROM AGENTS AS B
WHERE COMMISSION < 0.14 AND A.AGENT_CODE = B.AGENT_CODE)
```

Answer 4(d)

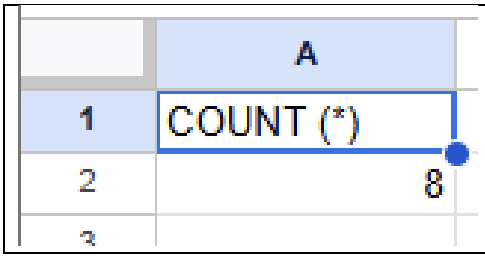


	A
1	AGENT_NAME
2	Ivan
3	

Question 4(e),

```
SELECT COUNT(*)
FROM AGENTS
NATURAL JOIN AGENTS_PHONE;
```

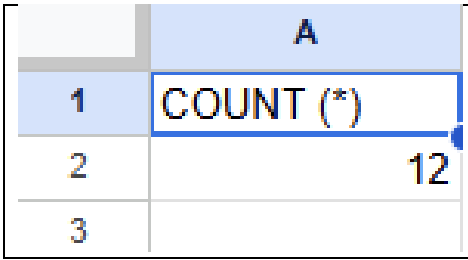
Answer 4(e)

	[reason] 2 + 2 +3+ 1 (A001, A003, A004, A010) Based on AGENTS_PHONE, the INNER JOIN value attribute is a table containing both elements

Question 4(f),

```
SELECT COUNT(*)
FROM AGENTS a
LEFT JOIN AGENTS_PHONE p
ON a.AGENT_CODE = p.AGENT_CODE;
```

Answer 4(f)

	[reason] OUTTER JOIN is for the purpose of maintaining DATA, Currently, the data of the AGENT a table is preserved. Since there are a total of 12 AGENT_CODE (A001~A012) in the AGENT table, 12 was output.

Question 4(g),

```
SELECT COUNT(*)
FROM AGENTS a
RIGHT JOIN AGENTS_PHONE p
ON a.AGENT_CODE = p.AGENT_CODE;
```

Answer 4(g)

	A
1	COUNT (*)
2	9
3	

[reason]
2 + 2 +3+ 1 + 1
(A001, A003, A004, A010, A013)

The INNER JOIN value + the AGENTS_PHONE records value declared RIGHT JOIN based on AGENTS_PHONE
(AGENT's attributes are treated as NULL), made table in the order of p-based attributes.