Bases de Données Fall, 2024

TD 5

The cut-and-paste code from this pdf file will work directly on the computer with `postgreSQL`, in case you want to try these queries.

Here are the tables we used in class:

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

(a) course

id	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

(b) teacher

id	*name*	*dept_name*	*tot_cred*
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

(c) student

course_id	*sec_id*	*semester*	*year*	*building*	*rn*	*time_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

(d) section

id	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

(e) teaches

id	*course_id*	*sec_id*	*semester*	*year*	*grade*
00128	CS-101	1	Fall	2009	A
00128	CS-347	1	Fall	2009	A-
12345	CS-101	1	Fall	2009	C
12345	CS-190	2	Spring	2009	A
12345	CS-315	1	Spring	2010	A
12345	CS-347	1	Fall	2009	A
19991	HIS-351	1	Spring	2010	B
23121	FIN-201	1	Spring	2010	C+
44553	PHY-101	1	Fall	2009	B-
45678	CS-101	1	Fall	2009	F
45678	CS-101	1	Spring	2010	B+
45678	CS-319	1	Spring	2010	B
54321	CS-101	1	Fall	2009	A-
54321	CS-190	2	Spring	2009	B+
55739	MU-199	1	Spring	2010	A-
76543	CS-101	1	Fall	2009	A
76543	CS-319	2	Spring	2010	A
76653	EE-181	1	Spring	2009	C
98765	CS-101	1	Fall	2009	C-
98765	CS-315	1	Spring	2010	B
98988	BIO-101	1	Summer	2009	A
98988	BIO-301	1	Summer	2010	

(f) takes

dept_name	*building*	*budget*
Biology	Watson	90000.00
Comp. Sci.	Taylor	100000.00
Elec. Eng.	Taylor	85000.00
Finance	Painter	120000.00
History	Painter	50000.00
Music	Packard	80000.00
Physics	Watson	70000.00

(g) department

1. Remember the query we did in class :

```
WITH dept_total (dept_name, value) AS
        (SELECT dept_name, sum(salary)
         FROM teacher
         GROUP BY dept_name),

 dept_total_avg(value) as
        (SELECT avg(value)
         FROM dept_total)

SELECT dept_name
FROM dept_total, dept_total_avg
WHERE dept_total.value >= dept_total_avg.value;
```

Now write this query *without* using **WITH**.

Écrivez maintenant cette requête *sans* utiliser **WITH**.

We give three solutions, two proposed by students in `TD`.

```
SELECT DISTINCT T1.dept_name
FROM teacher as T1
WHERE
        (SELECT sum(salary)
         FROM teacher as T2
         WHERE T2.dept_name = T1.dept_name)
                >=
        (SELECT avg(s)
         FROM
                (SELECT sum(salary) as s
                 FROM teacher as T3
                 GROUP BY T3.dept_name) as T4
        );
```

```
SELECT dept_name
FROM teacher
GROUP BY dept_name
HAVING sum(salary) >=
                        (SELECT avg(T1.m)
                         FROM (SELECT sum(T2.salary) as m
                                FROM teacher as T2
                                GROUP BY T2.dept_name) as T1
                        )
```

```
SELECT dept_name
FROM
        (SELECT dept_name, sum(salary) as value
         FROM teacher
         GROUP BY dept_name) as A
WHERE A.value >=
     (
          SELECT avg(value)
          FROM (SELECT dept_name, sum(salary) as value
                 FROM teacher
                 GROUP BY dept_name) as B
     );
```

dept_name
Comp. Sci.
Finance
Physics

2. Find the name of the teacher with the 4-th highest salary. Assume the salaries are distinct.

   Trouvez le nom de l'enseignant ayant le salaire le plus élevé au 4e rang. Supposons que les salaires soient distincts.

```
SELECT name
FROM teacher as T1
WHERE 3 = (
        SELECT COUNT (T2.salary)
        FROM teacher as T2
        WHERE T2.salary > T1.salary
        );
```

```
SELECT T.name
FROM (SELECT t1.name, count(*) as nb
      FROM teacher AS t1, teacher AS t2
      WHERE t1.salary < t2.salary
      GROUP BY t1.name) AS T
WHERE nb = 3;
```

```
SELECT t1.name
FROM teacher AS t1, teacher AS t2
WHERE t1.salary < t2.salary
GROUP BY t1.name
HAVING count(*) = 3;
```

name
Gold

3. Find the `names` and `salaries` of the teachers among the top 4 salaries, sorted by decreasing salary. Assume the salaries are distinct.

   Trouvez les `noms` et les `salaires` des enseignants parmi les 4 salaires les plus élevés, classés par salaire décroissant. Supposons que les salaires soient distincts.

```
SELECT T1. name , T1.salary
FROM teacher as T1
WHERE 3 >= (
        SELECT COUNT (T2.salary)
        FROM teacher as T2
        WHERE T2.salary > T1.salary)
ORDER BY T1.salary DESC;
```

name	*salary*
Einstein	95000.00
Brandt	92000.00
Wu	90000.00
Gold	87000.00

4. Find the `names` and `salaries` of the teachers among the 3 lowest salaries, sorted by increasing salary. Assume the salaries are distinct.

   Trouvez les `noms` et les `salaires` des enseignants parmi les 3 salaires les plus bas, classés par salaire croissant. Supposons que les salaires soient distincts.

```
SELECT T1. name, T1.salary
FROM teacher as T1
WHERE 2 >= (
        SELECT COUNT (T2.salary)
        FROM teacher as T2
        WHERE T2.salary < T1.salary)
ORDER BY T1.salary ASC;
```

name	*salary*
Mozart	40000.00
El Said	60000.00
Califieri	62000.00

5. Find the `ids` and `names` of all students who have not taken any course before 2010.

   Trouvez les `ids` et `names` de tous les étudiants qui n'ont suivi aucun cours avant 2010.

```
(SELECT student.id, student.name
FROM student)
EXCEPT
(SELECT student.id, student.name
FROM student, takes
WHERE takes.id = student.id and takes.year < 2010)
```

id	*name*
55739	Sanchez
19991	Brandt
23121	Chavez
70557	Snow

6. Find the names of all students who have taken a class in Fall 2009, using **IN** statement.

   Trouvez les noms de tous les étudiants qui ont suivi un cours à l'automne 2009, en utilisant l'instruction **IN**.

```sql
SELECT S.name
FROM student as S
WHERE ('Fall', 2009) IN  (SELECT semester, year
                          FROM takes
                          WHERE takes.id = S.id);
```

name
Zhang
Shankar
Peltier
Levy
Williams
Brown
Bourikas

7. Find the names of all students who have taken a class in Fall 2009, using **SOME** statement.

   Trouvez les noms de tous les étudiants qui ont suivi un cours à l'automne 2009, en utilisant l'instruction **SOME**.

```
SELECT S.name
FROM student as S
WHERE ('Fall', 2009) = SOME (SELECT semester, year
                             FROM takes
                             WHERE takes.id = S.id);
```

name
Zhang
Shankar
Peltier
Levy
Williams
Brown
Bourikas

8. Find the names of all students who have taken a class in Fall 2009, using **EXISTS** statement.

   Recherchez les noms de tous les étudiants qui ont suivi un cours à l'automne 2009, en utilisant l'instruction **EXISTS**.

```
SELECT name
FROM student
WHERE EXISTS (SELECT *
                FROM takes
                WHERE takes.id = student.id AND
                      semester = 'Fall' AND
                      year = 2009);
```

name
Shankar
Bourikas
Brown
Levy
Peltier
Zhang
Williams

9. Find all the teachers whose names begin with ‘E’.

   Trouvez tous les enseignants dont les noms commencent par ‘E’.

```sql
SELECT * FROM teacher WHERE name LIKE 'E%';
```

id	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00

Here is the schema for a new database:

Employee Database

employee(<u>name</u>, street, city)
works(<u>name</u>, company_name, salary)
company(<u>company_name</u>, city)
manages(<u>name</u>, manager_name)

10. Find all employees who earn more than the average salary of all employees of their company.

    Trouvez tous les employés qui gagnent plus que le salaire moyen de tous les employés de leur entreprise.

```sql
SELECT W1.name
FROM works AS W1
WHERE W1.salary >
        (SELECT  avg (salary)
         FROM works as W2
         WHERE W2.company_name = W1.company_name);
```

11. Find the company that has the smallest total salary of employees, using **ALL** statement.

    Trouvez l'entreprise qui a le plus petit salaire total d'employés, en utilisant l'instruction **ALL**.

```
SELECT W1.company_name
FROM works as W1
GROUP BY company_name
HAVING sum (salary) <= ALL ( SELECT sum (salary)
                             FROM works as W2
                             GROUP BY company_name
                            );
```