TP2 : Requêtes, dépéndances fonctionnelles et normalisation

Exercice 1 :

Q1) SELECT dept_name
FROM department
WHERE budget IN (
SELECT MAX(budget)
FROM department
);

Réponse : Finance

Q2)

SELECT Name_A.TEACHER_NAME, Name_A.salary
FROM TEACHER Name_A
WHERE Name_A.salary > (SELECT AVG(B.salary) FROM TEACHER B);

TEACHER_NAME	SALARY
Wu	90000
Einstein	95000
Gold	87000
Katz	75000
Singh	80000
Brandt	92000
Kim	80000

7 rows returned in 0.02 seconds Download

Remarque : Le **FROM Teacher AS B** ne marche qu'en PL/SQL d'Oracle, mais pas dans APEX Oracle.

Q3)

SELECT teacher.teacher_name, student.student_name, COUNT(*)
FROM teacher, student, takes, teaches
WHERE teacher.teacher_id = teaches.teaches_id
AND student.student_id = takes.takes_id
AND takes.course_id = teaches.course_id
AND takes.sec_id = teaches.sec_id
AND takes.semester = teaches.semester

```
AND takes.takes_year = teaches.teaches_year
GROUP BY teacher.teacher_name, student.student_name
HAVING COUNT(*) >= 2;
```

TEACHER_NAME	STUDENT_NAME	COUNT(*)
Srinivasan	Bourikas	2
Srinivasan	Shankar	3
Crick	Tanaka	2
Katz	Levy	2
Srinivasan	Zhang	2

Q4)

```
SELECT T.teachername, T.studentname, T.totalcount
FROM (
   SELECT
      teacher.teacher_name AS teachername,
      student.student_name AS studentname,
      COUNT(*) AS totalcount
   FROM
      teacher
      JOIN teaches ON teacher.teacher_id = teaches.teaches_id
      JOIN takes ON takes.course_id = teaches.course_id
         AND takes.sec_id = teaches.sec_id
         AND takes.semester = teaches.semester
         AND takes.takes_year = teaches.teaches_year
      JOIN student ON student.student_id = takes.takes_id
   GROUP BY
      teacher.teacher_name, student.student_name
) T
WHERE T.totalcount >= 2
ORDER BY T.teachername;
```

TEACHERNAME	STUDENTNAME	TOTALCOUNT
Crick	Tanaka	2
Katz	Levy	2
Srinivasan	Bourikas	2
Srinivasan	Shankar	3
Srinivasan	Zhang	2

Q5)

```
SELECT student.student_id, student.student_name
FROM student
MINUS
SELECT student.student_id, student.student_name
FROM student
JOIN takes ON takes.takes_id = student.student_id
WHERE takes.takes_year < 2009;
```

STUDENT_ID	STUDENT_NAME
00128	Zhang
12345	Shankar
19991	Brandt
23121	Chavez
44553	Peltier
45678	Levy
54321	Williams
55739	Sanchez
70557	Snow
76543	Brown
76653	Aoi
98765	Bourikas
98988	Tanaka

Q6) SELECT * FROM teacher WHERE teacher_name LIKE 'E%' ;

TEACHER_ID	TEACHER_NAME	DEPT_NAME	SALARY
22222	Einstein	Physics	95000
32343	El Said	History	60000

Q7) SELECT teacher_name

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

Réponse : Gold

Q8) SELECT T1.teacher_name, T1.salary
FROM teacher T1
WHERE 2 >= (
SELECT COUNT(DISTINCT T2.salary)
FROM teacher T2
WHERE T2.salary < T1.salary
)
ORDER BY T1.salary ASC;

TEACHER_NAME	SALARY
Mozart	40000
El Said	60000
Califieri	62000

Q9) SELECT S.student_name
FROM student S
WHERE ('Fall', 2009) IN (
SELECT takes.semester, takes.takes_year
FROM takes
WHERE takes.takes_id = S.student_id
);

STUDENT_NAME
Zhang
Shankar
Peltier
Levy
Williams
Brown
Bourikas

Q10) SELECT S.student_name
FROM student S
WHERE EXISTS (
SELECT 1
FROM takes
WHERE takes.takes_id = S.student_id
AND takes.semester = 'Fall'
AND takes.takes_year = 2009
);

STUDENT_NAME
Zhang
Shankar
Peltier
Levy
Williams
Brown
Bourikas

Q11) SELECT DISTINCT student.student_name
FROM takes
NATURAL JOIN student
WHERE takes.semester = 'Fall'
AND takes.takes_year = 2009;

STUDENT_NAME
Brandt
Sanchez
Brown
Aoi
Zhang
Levy
Snow
Bourikas
Shankar
Chavez
Peltier
Tanaka
Williams

Q12) SELECT student.student_name
FROM student
WHERE EXISTS (
SELECT 1
FROM takes
WHERE takes.takes_id = student.student_id
AND takes.semester = 'Fall'
AND takes.takes_year = 2009
);

STUDENT_NAME
Zhang
Shankar
Peltier
Levy
Williams
Brown
Bourikas

Q13) SELECT Name_A.student_name, Name_B.student_name
FROM student Name_A
JOIN takes TA ON Name_A.student_id = TA.takes_id
JOIN student Name_B ON Name_B.student_id = TA.takes_id
JOIN takes TB ON Name_B.student_id = TB.takes_id
WHERE Name_A.student_id <> Name_B.student_id
AND TA.course_id = TB.course_id
AND TA.sec_id = TB.sec_id
AND TA.semester = TB.semester
AND TA.takes_year = TB.takes_year
AND Name_A.student_name < Name_B.student_name
ORDER BY Name_A.student_name, Name_B.student_name;

Réponse : No Data Found

Exercice 2 :

Cas 1 : R(A,B,C) et F={A → B;B → C}

Dans cette relation, A détermine B et B détermine C, ce qui implique que A détermine également C par transitivité. Par conséquent, A est la clé primaire de la relation. La relation est en 1ère forme normale (1NF) car chaque attribut contient des valeurs atomiques. En ce qui concerne la 2ème forme normale (2NF), la relation est valide car chaque attribut non-clé dépend entièrement de la clé primaire A. Cependant, la relation n'est pas en 3ème forme normale (3NF), car il existe une dépendance transitive A → B → C, ce qui viole la règle de la 3NF. De plus, elle n'est pas en forme normale de Boyce-Codd (BCNF), car la dépendance B → C viole la condition de BCNF, étant donné que B n'est pas une superclé. Pour atteindre la 3NF, la relation doit être décomposée en deux relations : R1(A,B) et R2(B,C), respectivement en 3NF.

Cas 2 : R(A,B,C) et F={A → C;A → B}

Dans ce cas, les dépendances fonctionnelles montrent que A détermine à la fois B et C, ce qui fait de A la clé primaire de la relation. La relation est en 1NF, car chaque attribut contient des valeurs atomiques. Puisque A est la clé primaire et que tous les autres attributs dépendent pleinement de cette clé, la relation est également en 2NF. Il n’y a aucune dépendance transitive, ce qui fait que la relation est en 3NF. Enfin, comme chaque dépendance fonctionnelle repose sur la clé primaire A, la relation est aussi en forme normale de Boyce-Codd (BCNF). En résumé, cette relation est déjà en BCNF.

Cas 3 : R(A,B,C) et F={A,B → C;C → B}

Ici, la dépendance A,B → C montre que la combinaison de A et B détermine C, tandis que C → B suggère que C détermine B. Cela implique que C peut être une clé candidate. Cependant, il existe une dépendance partielle entre C et B, ce qui viole la 2NF. Par conséquent, la relation n'est pas en 2NF, et donc pas en 3NF ni en BCNF. Pour corriger cela et atteindre la 2NF, la relation doit être décomposée en deux relations : R1(A,B,C), avec la dépendance A,B → C, et R2(C,B), avec la dépendance C → B. Ces deux relations respectent la 2NF.