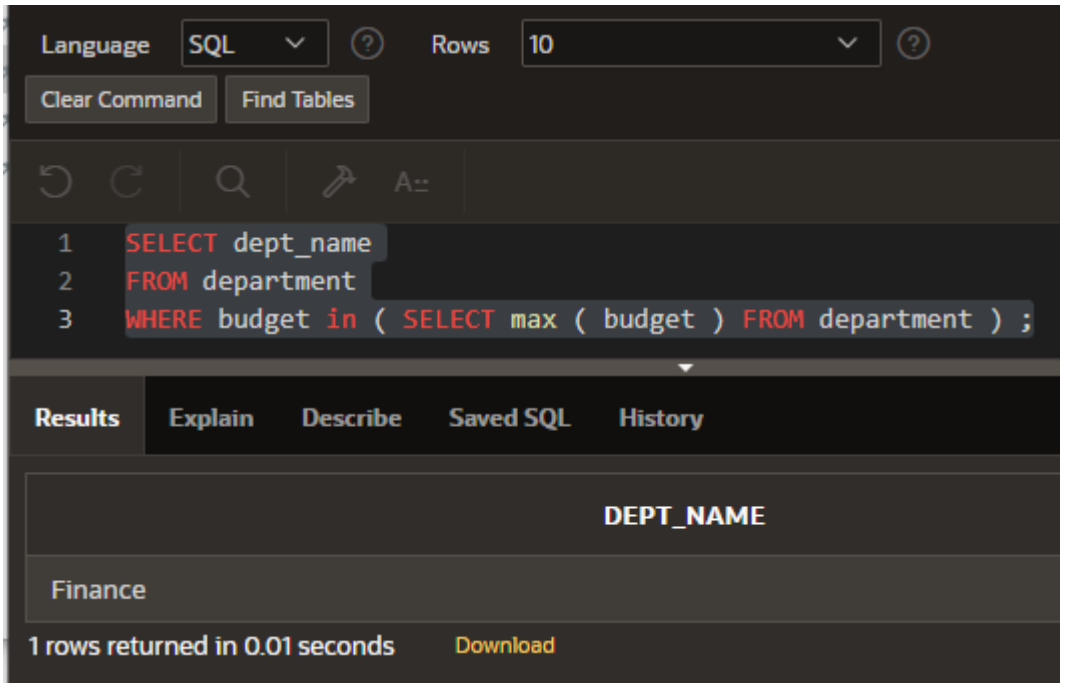
TP n° 2 : Requêtes, dépendances fonctionnelles et normalisation

Exercice 1 :

1. Afficher le nom du département qui a le budget le plus élevé.

```
SELECT dept_name
FROM department
WHERE budget in ( SELECT max ( budget ) FROM department ) ;
```



2. Afficher les salaires et les noms des enseignants qui gagnent plus que le salaire moyen

```
SELECT A.name, A.salary
FROM teacher A
WHERE A.salary > (SELECT AVG(B.salary) FROM teacher B);
```

```sql
SELECT A.name, A.salary
FROM teacher A
WHERE A.salary > (SELECT AVG(B.salary) FROM teacher B);
```

Results | Explain | Describe | Saved SQL | History

NAME	SALARY
Wu	90000
Einstein	95000
Gold	87000
Katz	75000
Singh	80000

3. Pour chaque enseignant, afficher tous les étudiants qui ont suivi plus de deux cours dispensés par cet enseignant ainsi que le nombre total de cours suivis par chaque étudiant, en utilisant la clause HAVING

```sql
SELECT teacher.name, student.name, COUNT(*)
FROM teacher
JOIN teaches ON teacher.id = 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.year = teaches.year
JOIN student ON student.id = takes.id
GROUP BY teacher.name, student.name
HAVING COUNT(*) >= 2;
```

```sql
SELECT teacher.name, student.name, COUNT(*)
FROM teacher
JOIN teaches ON teacher.id = 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.year = teaches.year
```

Results | Explain | Describe | Saved SQL | History

NAME	NAME	COUNT(*)
Katz	Levy	2
Srinivasan	Zhang	2

5 rows returned in 0.06 seconds Download

4. Pour chaque enseignant, afficher tous les étudiants qui ont suivi plus de deux cours dispensés par cet enseignant ainsi que le nombre total de cours suivis par chaque étudiant, sans utiliser la clause HAVING.

```sql
SELECT T.teachername, T.studentname, T.totalcount
FROM (
   SELECT teacher.name AS teachername, student.name AS studentname, COUNT(*) AS totalcount
   FROM teacher
   JOIN teaches ON teacher.id = 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.year = teaches.year
   JOIN student ON student.id = takes.id
   GROUP BY teacher.name, student.name
) T
WHERE T.totalcount >= 2
ORDER BY T.teachername;
```

```sql
SELECT T.teachername, T.studentname, T.totalcount
FROM (
    SELECT teacher.name AS teachername, student.name AS studentname, COUNT(*) AS
    FROM teacher
    JOIN teaches ON teacher.id = teaches.id
```

Results | Explain | Describe | Saved SQL | History

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

5. Afficher les identifiants et les noms des étudiants qui n'ont pas suivi de cours avant 2010.

```sql
(SELECT student.id, student.name
 FROM student)
EXCEPT
(SELECT student.id, student.name
 FROM student
 JOIN takes ON takes.id = student.id
 WHERE takes.year < 2009);
```

```
(SELECT student.id, student.name
 FROM student)
EXCEPT
(SELECT student.id, student.name
 FROM student
```

Results | Explain | Describe | Saved SQL | History

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

6. Afficher tous les enseignants dont les noms commencent par E.

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

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

Results | Explain | Describe | Saved SQL | History

ID	NAME	DEPT_NAME	SALARY
22222	Einstein	Physics	95000
32343	El Said	History	60000

2 rows returned in 0.00 seconds Download

7. Afficher les salaires et les noms des enseignants qui perçoivent le quatrième salaire le plus élevé

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

);

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

Results | Explain | Describe | Saved SQL | History

NAME
Gold

1 rows returned in 0.01 seconds Download

8. Afficher les noms et les salaires des trois enseignants qui perçoivent les salaires les moins élevés.
Les afficher par ordre décroissant.

```
SELECT T1.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;
```

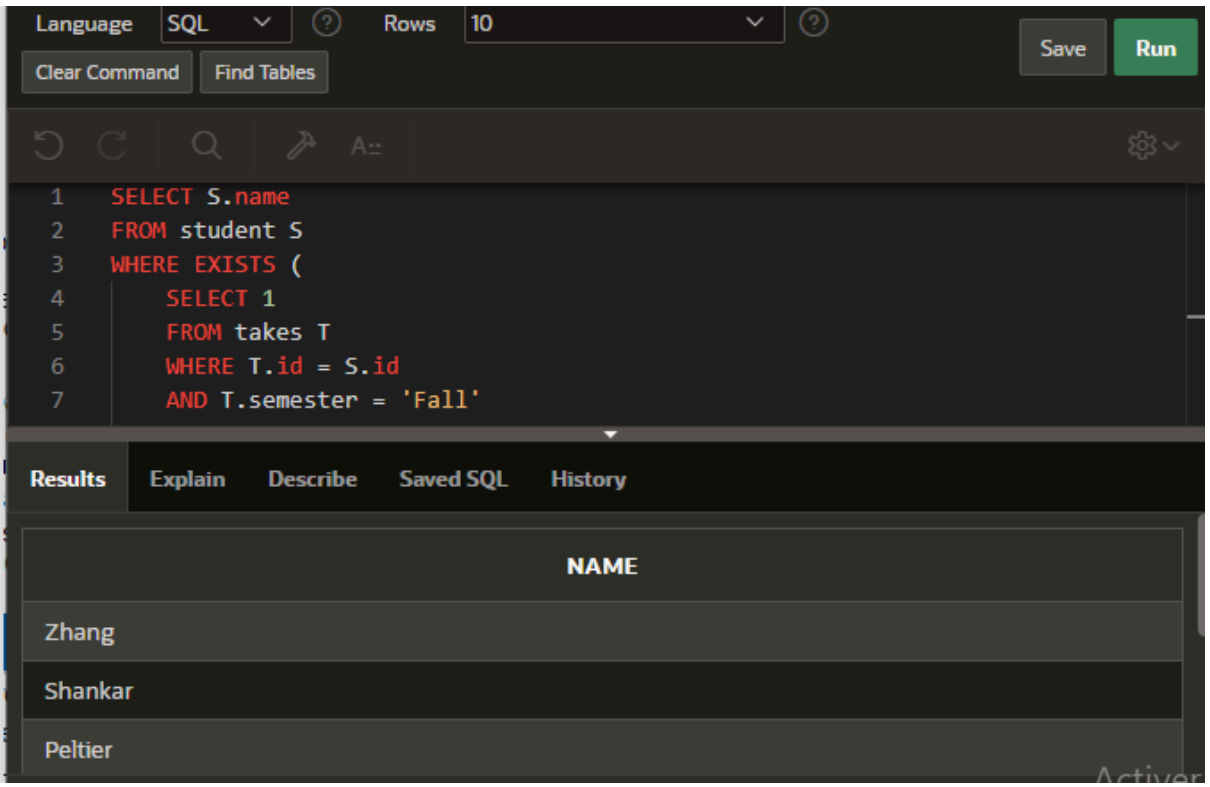
```sql
SELECT T1.name, T1.salary
FROM teacher T1
WHERE 2 >= (
    SELECT COUNT(DISTINCT T2.salary)
    FROM teacher T2
    WHERE T2.salary < T1.salary
)
```

Results | Explain | Describe | Saved SQL | History

NAME	SALARY
Mozart	40000
El Said	60000
Califieri	62000

9. Afficher les noms des étudiants qui ont suivi un cours en automne 2009, en utilisant la clause IN.

```sql
SELECT S.name
FROM student S
WHERE EXISTS (
   SELECT 1
   FROM takes T
   WHERE T.id = S.id
   AND T.semester = 'Fall'
   AND T.year = 2009
);
```



10. Afficher les noms des étudiants qui ont suivi un cours en automne 2009, en utilisant la clause SOME.

```
SELECT S.name
FROM student S
WHERE EXISTS (
   SELECT 1
   FROM takes T
   WHERE T.id = S.id
   AND T.semester = 'Fall'
   AND T.year = 2009
);
```

```sql
SELECT S.name
FROM student S
WHERE EXISTS (
    SELECT 1
    FROM takes T
    WHERE T.id = S.id
```

Results | Explain | Describe | Saved SQL | History

NAME
Zhang
Shankar
Peltier
Levy

11. Afficher les noms des étudiants qui ont suivi un cours en automne 2009, en utilisant la jointure naturelle (NATURAL INNER JOIN).

```sql
SELECT DISTINCT student.name
FROM takes
NATURAL JOIN student
WHERE takes.semester = 'Fall'
AND takes.year = 2009;
```

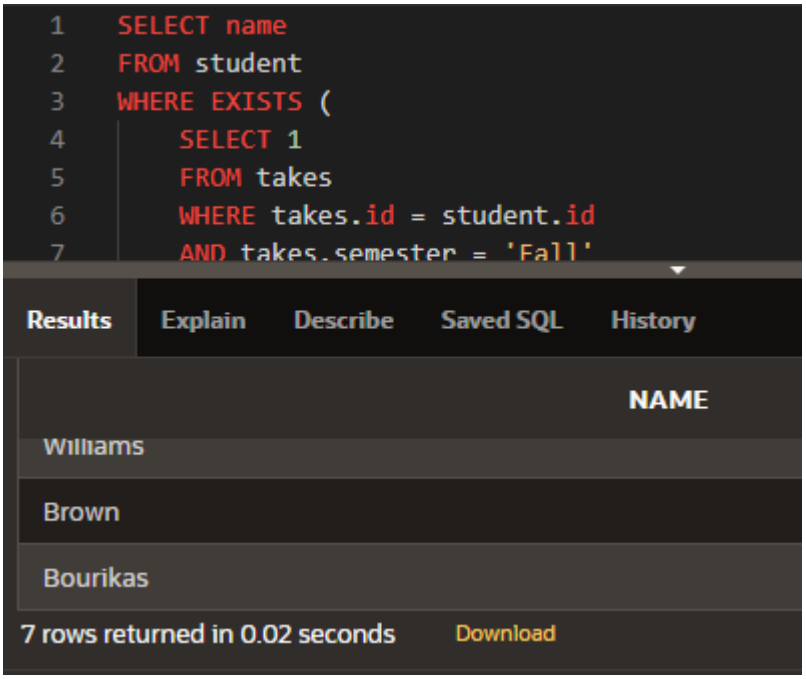
```sql
SELECT DISTINCT student.name
FROM takes
NATURAL JOIN student
WHERE takes.semester = 'Fall'
AND takes.year = 2009;
```

Results | Explain | Describe | Saved SQL | History

NAME
Brown
Zhang
Levy
Bourikas

12. Afficher les noms des étudiants qui ont suivi un cours en automne 2009, en utilisant la clause EXISTS.

```
SELECT name
FROM student
WHERE EXISTS (
   SELECT 1
   FROM takes
   WHERE takes.id = student.id
   AND takes.semester = 'Fall'
   AND takes.year = 2009
);
```



13. Afficher toutes les paires des étudiants qui ont suivi au moins un cours ensemble.

```
SELECT t.ID AS teacher_id, t.name AS teacher_name, COUNT(*) AS total_students
FROM teacher t
JOIN teaches te ON t.ID = te.ID
JOIN section sec ON te.course_id = sec.course_id AND te.sec_id = sec.sec_id
JOIN takes tk ON sec.course_id = tk.course_id AND sec.sec_id = tk.sec_id
GROUP BY t.ID, t.name
ORDER BY total_students ASC;
```

```sql
SELECT t.ID AS teacher_id, t.name AS teacher_name, COUNT(*) AS total_students
FROM teacher t
JOIN teaches te ON t.ID = te.ID
JOIN section sec ON te.course_id = sec.course_id AND te.sec_id = sec.sec_id
JOIN takes tk ON sec.course_id = tk.course_id AND sec.sec_id = tk.sec_id
```

Results | Explain | Describe | Saved SQL | History

TEACHER_ID	TEACHER_NAME	TOTAL_STUDENTS
15151	Mozart	1
32343	El Said	1
12121	Wu	1
98345	Kim	1

14. Afficher pour chaque enseignant, qui a effectivement assuré un cours, le nombre total d'étudiants qui ont suivi ses cours. Si un étudiant a suivi deux cours différents avec le même enseignant, on le compte deux fois. Trier le r´esultat par ordre décroissant

```sql
SELECT teacher.name, COUNT(*) AS num_courses
FROM takes
INNER JOIN teaches ON takes.course_id = teaches.course_id
            AND takes.sec_id = teaches.sec_id
            AND takes.semester = teaches.semester
            AND takes.year = teaches.year
INNER JOIN teacher ON teaches.id = teacher.id
GROUP BY teacher.name, teacher.id
ORDER BY num_courses DESC;
```

```sql
SELECT teacher.name, COUNT(*) AS num_courses
FROM takes
INNER JOIN teaches ON takes.course_id = teaches.course_id
                   AND takes.sec_id = teaches.sec_id
                   AND takes.semester = teaches.semester
                   AND takes.year = teaches.year
INNER JOIN teacher ON teaches.id = teacher.id
```

Results | Explain | Describe | Saved SQL | History

NAME	NUM_COURSES
Mozart	1
Wu	1
Kim	1

9 rows returned in 0.07 seconds Download

15. Afficher pour chaque enseignant, même s'il n'a pas assuré de cours, le nombre total d'étudiants qui ont suivi ses cours. Si un étudiant a suivi deux fois un cours avec le même enseignant, on le compte deux fois. Trier le résultat par ordre décroissant.

```sql
SELECT teacher . name , count ( course_id )
FROM ( takes INNER JOIN teaches USING ( course_id , sec_id ,
semester , year ) )
RIGHT OUTER JOIN teacher ON teaches . id = teacher . id
GROUP BY teacher . name , teacher . id ORDER BY count ( course_id ) DESC ;
```

Language SQL Rows 10 Save Run

Clear Command Find Tables

```sql
SELECT teacher . name , count ( course_id )
FROM ( takes INNER JOIN teaches USING ( course_id , sec_id ,
semester , year ) )
RIGHT OUTER JOIN teacher ON teaches . id = teacher . id
GROUP BY teacher . name , teacher . id ORDER BY count ( course_id ) DESC ;
```

Results Explain Describe Saved SQL History

NAME	COUNT(COURSE_ID)
Mozart	1
Gold	0

More than 10 rows available. Increase rows selector to view more rows.

16. Pour chaque enseignant, afficher le nombre total de grades A qu'il a attribué.

```sql
WITH mytakes AS (
   SELECT id, course_id, sec_id, semester, year, grade
   FROM takes
   WHERE grade = 'A'
)
SELECT teacher.name, COUNT(teaches.course_id) AS total_courses
FROM mytakes
JOIN teaches
ON mytakes.course_id = teaches.course_id
AND mytakes.sec_id = teaches.sec_id
AND mytakes.semester = teaches.semester
AND mytakes.year = teaches.year
RIGHT OUTER JOIN teacher
ON teaches.id = teacher.id
GROUP BY teacher.name, teacher.id
ORDER BY total_courses DESC;
```

```sql
WITH mytakes AS (
    SELECT id, course_id, sec_id, semester, year, grade
    FROM takes
    WHERE grade = 'A' -- Correction ici : 'A' est une chaîne de caractères
)
```

Results | Explain | Describe | Saved SQL | History

NAME	TOTAL_COURSES
Srinivasan	4
Brandt	2
Crick	1
Califieri	0

17. Afficher toutes les paires enseignants-élèves ou un élève a suivi le cours de l'enseignant, ainsi que le nombre de fois que cet élève a suivi un cours dispensé par cet enseignant. 18. Afficher toutes les paires enseignant-´élève o`u un ´el`eve a suivi au moins deux cours dispensé par l'enseignant en question

```sql
SELECT teacher . name , student . name , count (*) FROM ( teacher NATURAL JOIN t
INNER JOIN
( takes NATURAL JOIN student ) USING
( course_id , sec_id , semester , year ) GROUP BY teacher . name , student . nam
```

Results | Explain | Describe | Saved SQL | History

NAME	NAME	COUNT(*)
Brandt	Shankar	1
Wu	Chavez	1
El Said	Brandt	1
Einstein	Peltier	1

18. Afficher toutes les paires enseignant-´el`élève ou un`élève a suivi au moins deux cours dispensés par l'enseignant en question.

```sql
SELECT mytable.tn, mytable.sn
FROM (
   SELECT teacher.name AS tn, student.name AS sn, COUNT(*) AS tc
   FROM (teacher NATURAL JOIN teaches)
   INNER JOIN (takes NATURAL JOIN student)
   USING (course_id, sec_id, semester, year)
```

```
    GROUP BY teacher.name, student.name
) mytable
WHERE mytable.tc >= 2;
```

```sql
SELECT mytable.tn, mytable.sn
FROM (
    SELECT teacher.name AS tn, student.name AS sn, COUNT(*) AS tc
    FROM (teacher NATURAL JOIN teaches)
    INNER JOIN (takes NATURAL JOIN student)
```

Results | Explain | Describe | Saved SQL | History

TN	SN
Srinivasan	Shankar
Crick	Tanaka
Katz	Levy
Srinivasan	Zhang

5 rows returned in 0.03 seconds Download