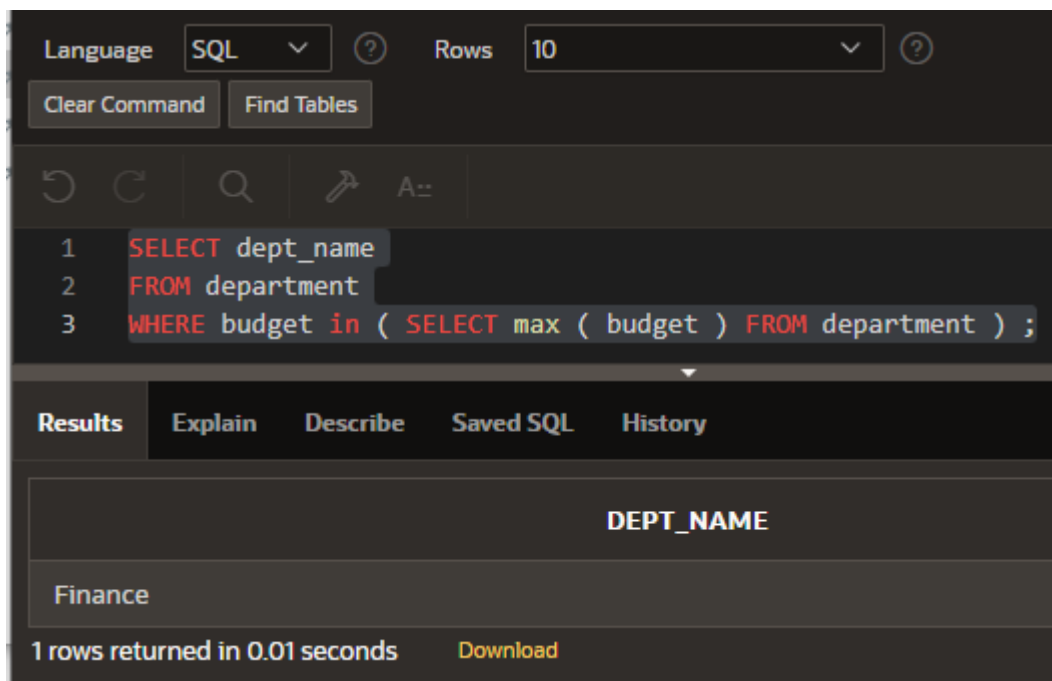


TP n° 2 : Requêtes, dépendances fonctionnelles et normalisation

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 ) ;
```



The screenshot shows a SQL query editor interface. At the top, there are controls for 'Language' (set to SQL) and 'Rows' (set to 10). Below these are buttons for 'Clear Command' and 'Find Tables'. The query editor contains the following SQL code:

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

Below the query editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, showing a table with the following structure:

DEPT_NAME
Finance

At the bottom, it indicates '1 rows returned in 0.01 seconds' and provides a 'Download' link.

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);
```

1	SELECT A.name, A.salary
2	FROM teacher A
3	WHERE A.salary > (SELECT AVG(B.salary) FROM teacher B);
4	

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

```

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;

```

1	SELECT teacher.name, student.name, COUNT(*)
2	FROM teacher
3	JOIN teaches ON teacher.id = teaches.id
4	JOIN takes ON takes.course_id = teaches.course_id
5	AND takes.sec_id = teaches.sec_id
6	AND takes.semester = teaches.semester
7	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.

```

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;

```


</

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

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

1SELECT * FROM teacher WHERE name LIKE 'E%';

Results

ExplainDescribeSaved SQLHistory

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

2 rows returned in 0.00 secondsDownload

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`

);

```
1  SELECT name
2  FROM teacher T1
3  WHERE 3 = (
4      SELECT COUNT(DISTINCT T2.salary)
5      FROM teacher T2
6      WHERE T2.salary > T1.salary
7  )
```

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;
```

<

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

```
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
);
```

The screenshot shows a SQL IDE interface. At the top, there's a toolbar with 'Language' set to 'SQL', 'Rows' set to '10', and buttons for 'Clear Command', 'Find Tables', 'Save', and 'Run'. Below the toolbar is a query editor with the following SQL code:

```
1 SELECT S.name
2 FROM student S
3 WHERE EXISTS (
4     SELECT 1
5     FROM takes T
6     WHERE T.id = S.id
7     AND T.semester = 'Fall'
```

Below the query editor is a results pane with tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing a table with one column named 'NAME'. The table contains three rows of data:

NAME
Zhang
Shankar
Peltier

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
);
```


1	SELECT S.name
2	FROM student S
3	WHERE EXISTS (
4	SELECT 1
5	FROM takes T
6	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).

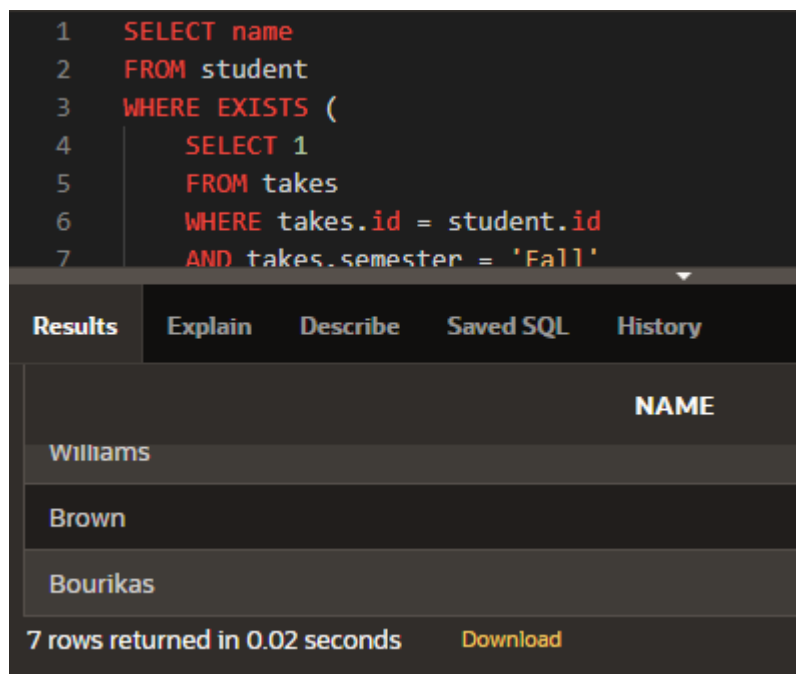
```
SELECT DISTINCT student.name
FROM takes
NATURAL JOIN student
WHERE takes.semester = 'Fall'
AND takes.year = 2009;
```

1	SELECT DISTINCT student.name
2	FROM takes
3	NATURAL JOIN student
4	WHERE takes.semester = 'Fall'
5	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
);
```



The screenshot shows a SQL query execution interface. The query is displayed in a dark-themed editor with line numbers 1 through 7. Below the editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, showing a table with one column named 'NAME'. The table contains three rows with the names 'Williams', 'Brown', and 'Bourikas'. At the bottom of the results section, it states '7 rows returned in 0.02 seconds' and provides a 'Download' link.

```
1  SELECT name
2  FROM student
3  WHERE EXISTS (
4      SELECT 1
5      FROM takes
6      WHERE takes.id = student.id
7      AND takes.semester = 'Fall'
```

NAME
Williams
Brown
Bourikas

7 rows returned in 0.02 seconds [Download](#)

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;
```

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 résultat par ordre décroissant

```

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;

```

1	SELECT teacher.name, COUNT(*) AS num_courses
2	FROM takes
3	INNER JOIN teaches ON takes.course_id = teaches.course_id
4	AND takes.sec_id = teaches.sec_id
5	AND takes.semester = teaches.semester
6	AND takes.year = teaches.year
7	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.

```
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

↶ ↷ 🔍 🔗 A:: ⚙️

```

1 SELECT teacher . name , count ( course_id )
2 FROM ( takes INNER JOIN teaches USING ( course_id , sec_id ,
3 semester , year ) )
4 RIGHT OUTER JOIN teacher ON teaches . id = teacher . id
5 GROUP BY teacher . name , teacher . id ORDER BY count ( course_id ) DESC ;
6

```

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.

```

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;

```

```

1  WITH mytakes AS (
2      SELECT id, course_id, sec_id, semester, year, grade
3      FROM takes
4      WHERE grade = 'A' -- Correction ici : 'A' est une chaîne de caractères
5  )

```

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 où un élève a suivi au moins deux cours dispensés par l'enseignant en question

```

1  SELECT teacher . name , student . name , count (*) FROM ( teacher NATURAL JOIN t
2  INNER JOIN
3  ( takes NATURAL JOIN student ) USING
4  [ course_id , sec_id , semester , year ] GROUP BY teacher . name , student . nam

```

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

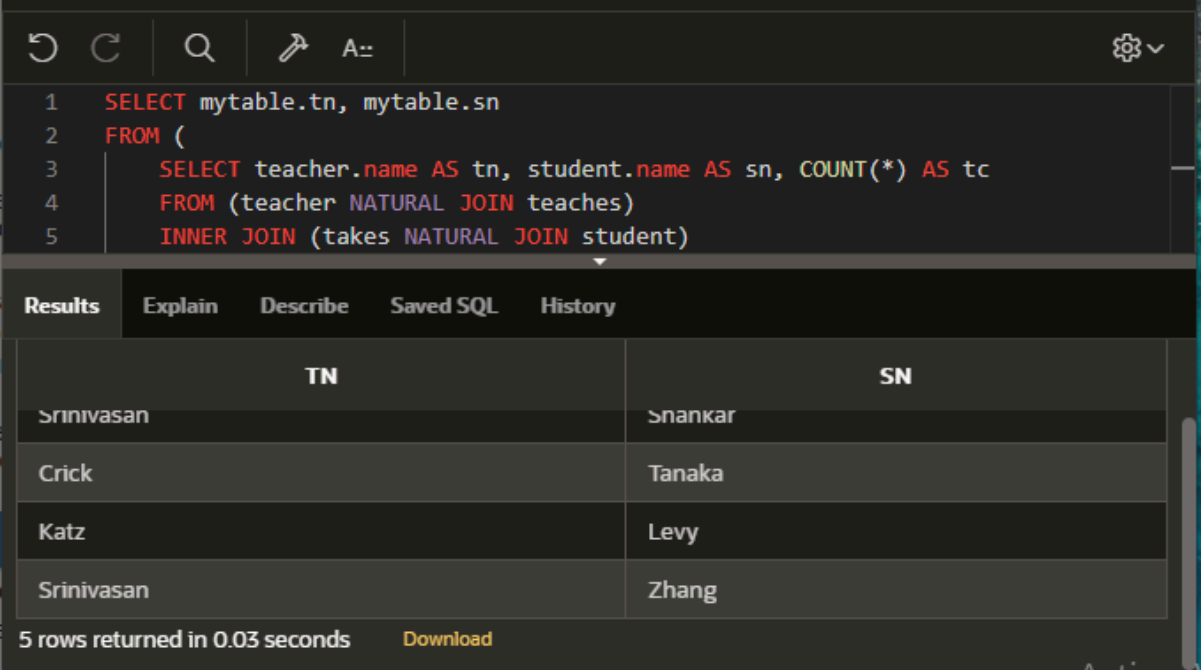
18. Afficher toutes les paires enseignant-élève où un élève a suivi au moins deux cours dispensés par l'enseignant en question.

```

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;
```



The screenshot shows a SQL IDE interface. At the top, there is a toolbar with icons for undo, redo, search, and a command prompt. Below the toolbar, a SQL query is entered in a text area. The query is as follows:

```
1 SELECT mytable.tn, mytable.sn
2 FROM (
3     SELECT teacher.name AS tn, student.name AS sn, COUNT(*) AS tc
4     FROM (teacher NATURAL JOIN teaches)
5     INNER JOIN (takes NATURAL JOIN student)
```

Below the query editor, there is a tabbed interface with the following tabs: Results, Explain, Describe, Saved SQL, and History. The 'Results' tab is currently selected, displaying a table with two columns: TN and SN. The table contains four rows of data:

TN	SN
Srinivasan	Snankar
Crick	Tanaka
Katz	Levy
Srinivasan	Zhang

At the bottom of the Results tab, it states '5 rows returned in 0.03 seconds' and provides a 'Download' link.