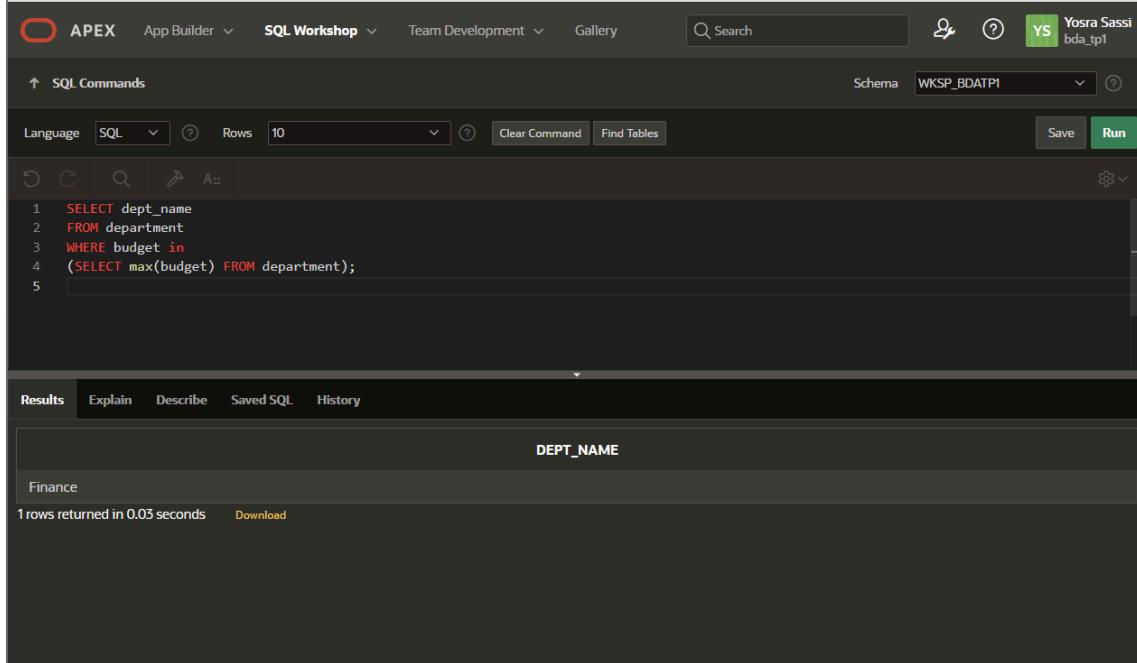


Exercice 1 – Requêtes SQL

Dans cet exercice, nous appliquons des requêtes SQL afin d'extraire des informations pertinentes à partir d'une base de données universitaire.

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

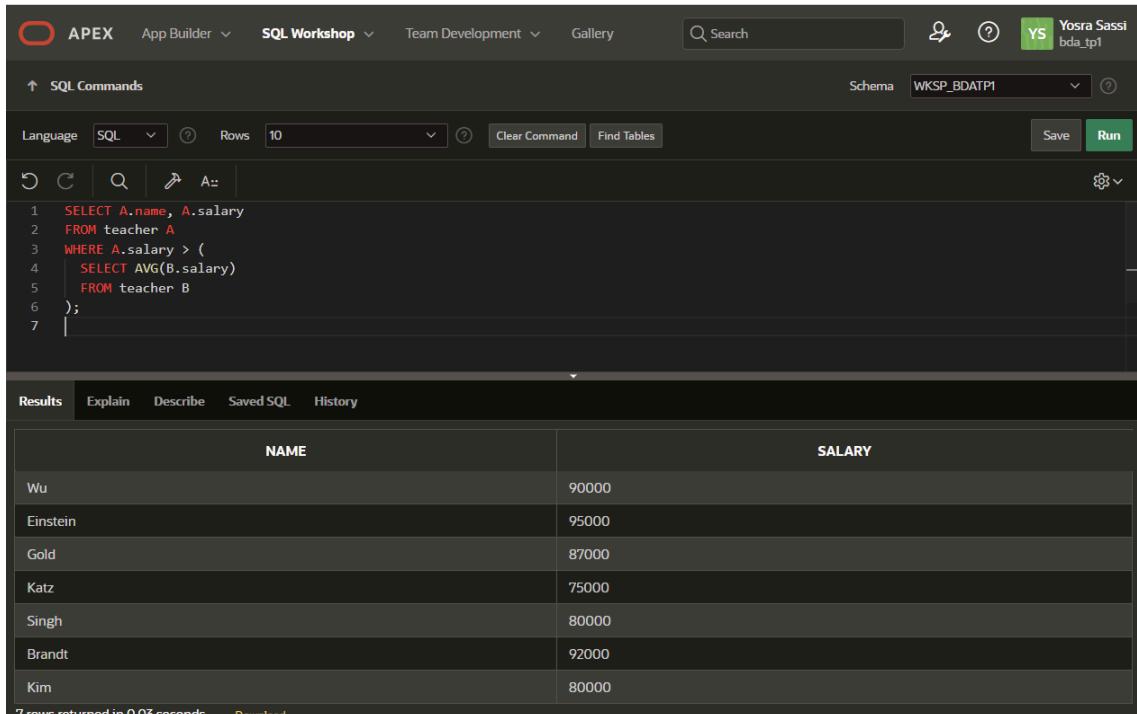


```
SELECT dept_name
FROM department
WHERE budget IN
(SELECT max(budget) FROM department);
```

DEPT_NAME
Finance

1 rows returned in 0.03 seconds [Download](#)

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
);
```

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

7 rows returned in 0.03 seconds [Download](#)

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.

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```

1 SELECT teacher.name, student.name, COUNT(*)
2 FROM teacher, student, takes, teaches
3 WHERE teacher.id = teaches.id
4 AND student.id = takes.id AND takes.course_id = teaches.course_id AND takes.sec_id = teaches.sec_id
5 AND takes.semester = teaches.semester AND takes.year = teaches.year
6 GROUP BY teacher.name, student.name
7 HAVING COUNT(*) >= 2;
8

```

The results table displays the following data:

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

5 rows returned in 0.00 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.

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```

1 SELECT T.teachername, T.studentname, T.totalcount
2 FROM (
3   SELECT teacher.name AS teachername, student.name AS studentname, COUNT(*) AS totalcount
4   FROM teacher, student, takes, teaches
5   WHERE teacher.id = teaches.id AND student.id = takes.id AND takes.course_id = teaches.course_id
6   AND takes.sec_id = teaches.sec_id AND takes.semester = teaches.semester AND takes.year = teaches.year
7   GROUP BY teacher.name, student.name
8 ) T
9 WHERE T.totalcount >= 2
10 ORDER BY T.teachername;
11

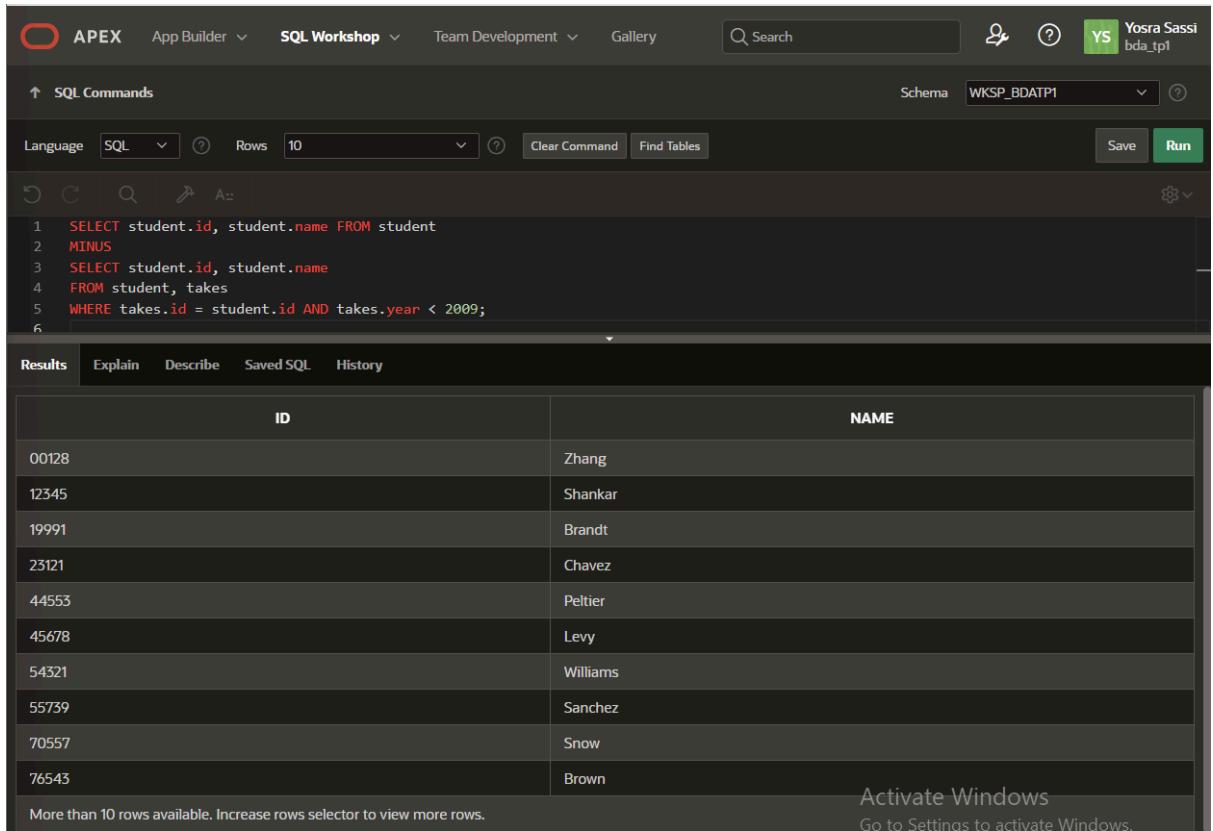
```

The results table displays the following data:

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

5 rows returned in 0.03 seconds [Download](#)

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



```

APEX SQL Workshop
Schema: WKSP_BDATPI

SQL Commands
Language: SQL Rows: 10
SELECT student.id, student.name FROM student
MINUS
SELECT student.id, student.name
FROM student, takes
WHERE takes.id = student.id AND takes.year < 2009;

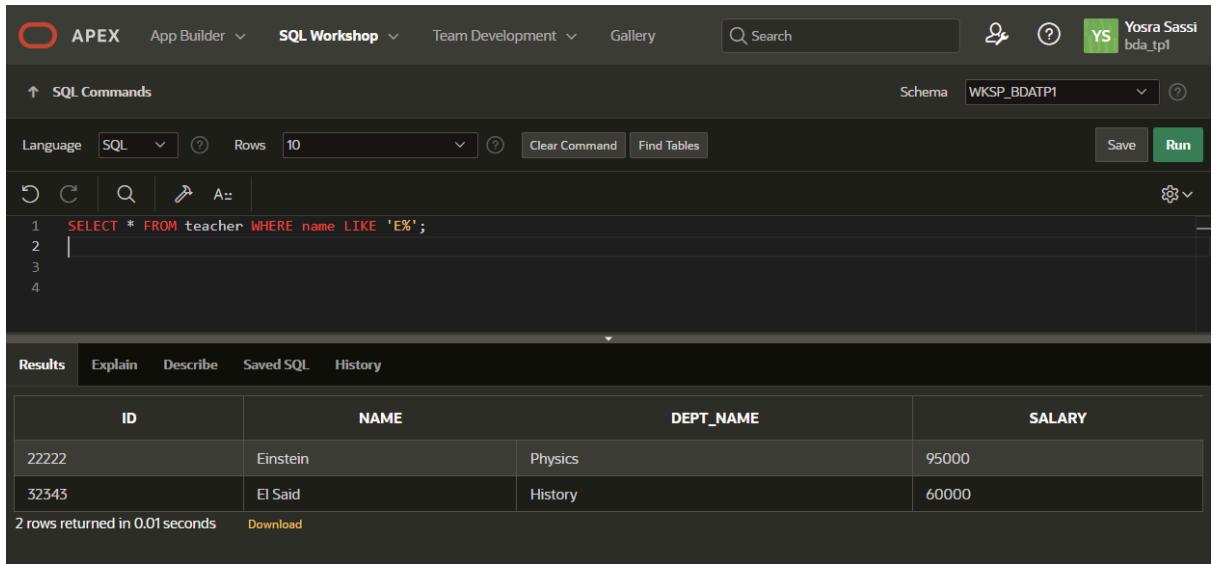
```

Results

ID	NAME
00128	Zhang
12345	Shankar
19991	Brandt
23121	Chavez
44553	Peltier
45678	Levy
54321	Williams
55739	Sanchez
70557	Snow
76543	Brown

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

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



```

APEX SQL Workshop
Schema: WKSP_BDATPI

SQL Commands
Language: SQL Rows: 10
SELECT * FROM teacher WHERE name LIKE 'E%';

```

Results

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

2 rows returned in 0.01 seconds [Download](#)

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

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands pane contains the following query:

```

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 );
8

```

The Results pane displays the output:

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.

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands pane contains the following query:

```

1 SELECT T1. name , T1.salary FROM teacher T1
2 WHERE 2 >= (
3   SELECT COUNT (DISTINCT T2.salary) FROM teacher T2
4   WHERE T2.salary < T1.salary)
5   ORDER BY T1.salary ASC;
6
7

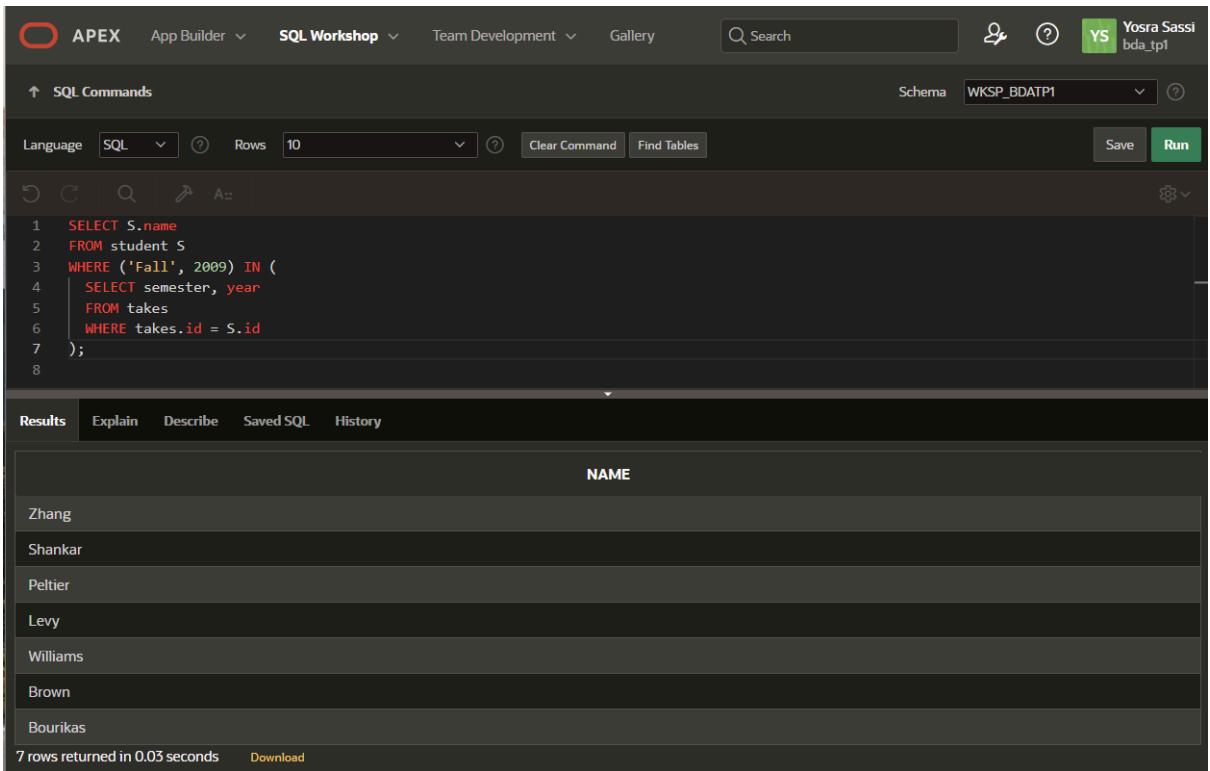
```

The Results pane displays the output:

NAME	SALARY
Mozart	40000
El Said	60000
Califieri	62000

3 rows returned in 0.01 seconds [Download](#)

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



The screenshot shows the Oracle SQL Workshop interface. The query window contains the following SQL code:

```

1 SELECT S.name
2 FROM student S
3 WHERE ('Fall', 2009) IN (
4   SELECT semester, year
5   FROM takes
6   WHERE takes.id = S.id
7 );
8

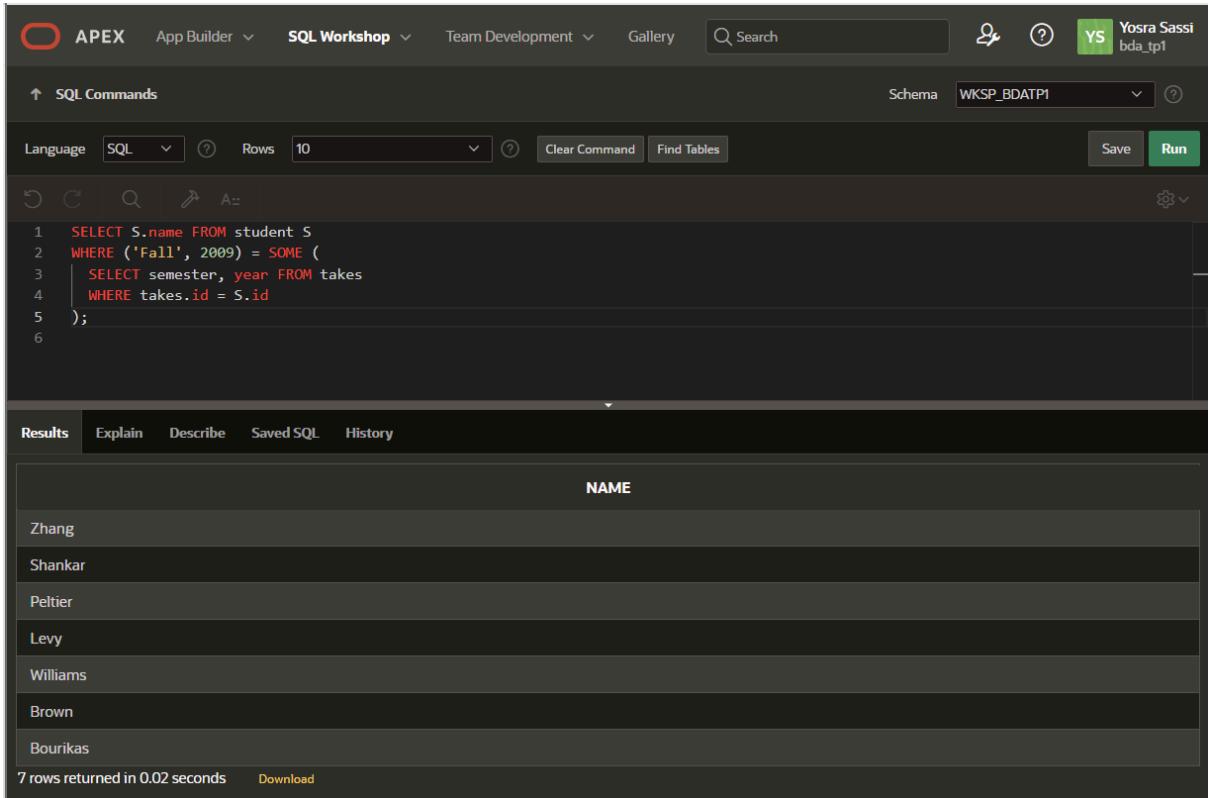
```

The results pane displays the following table:

NAME
Zhang
Shankar
Peltier
Levy
Williams
Brown
Bourikas

7 rows returned in 0.05 seconds [Download](#)

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



The screenshot shows the Oracle SQL Workshop interface. The query window contains the following SQL code:

```

1 SELECT S.name FROM student S
2 WHERE ('Fall', 2009) = SOME (
3   SELECT semester, year FROM takes
4   WHERE takes.id = S.id
5 );
6

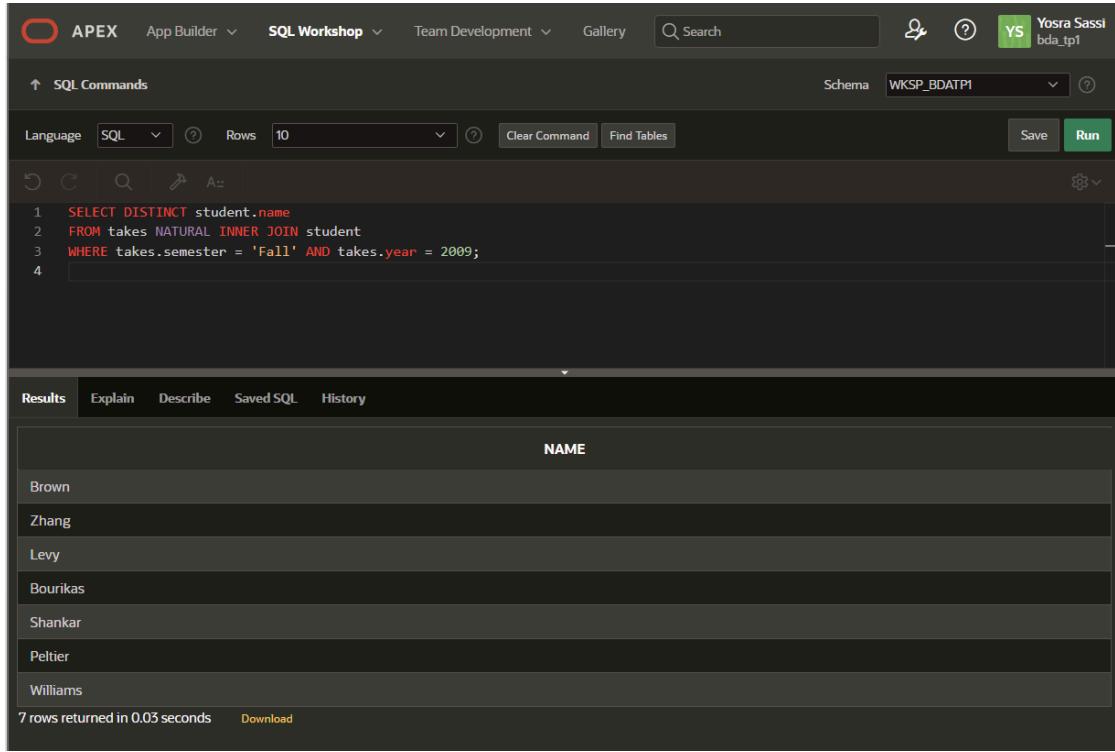
```

The results pane displays the following table:

NAME
Zhang
Shankar
Peltier
Levy
Williams
Brown
Bourikas

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

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



The screenshot shows the Oracle SQL Workshop interface. The schema is set to WKSP_BDATP1. The SQL command entered is:

```

1 SELECT DISTINCT student.name
2 FROM takes NATURAL INNER JOIN student
3 WHERE takes.semester = 'Fall' AND takes.year = 2009;
4

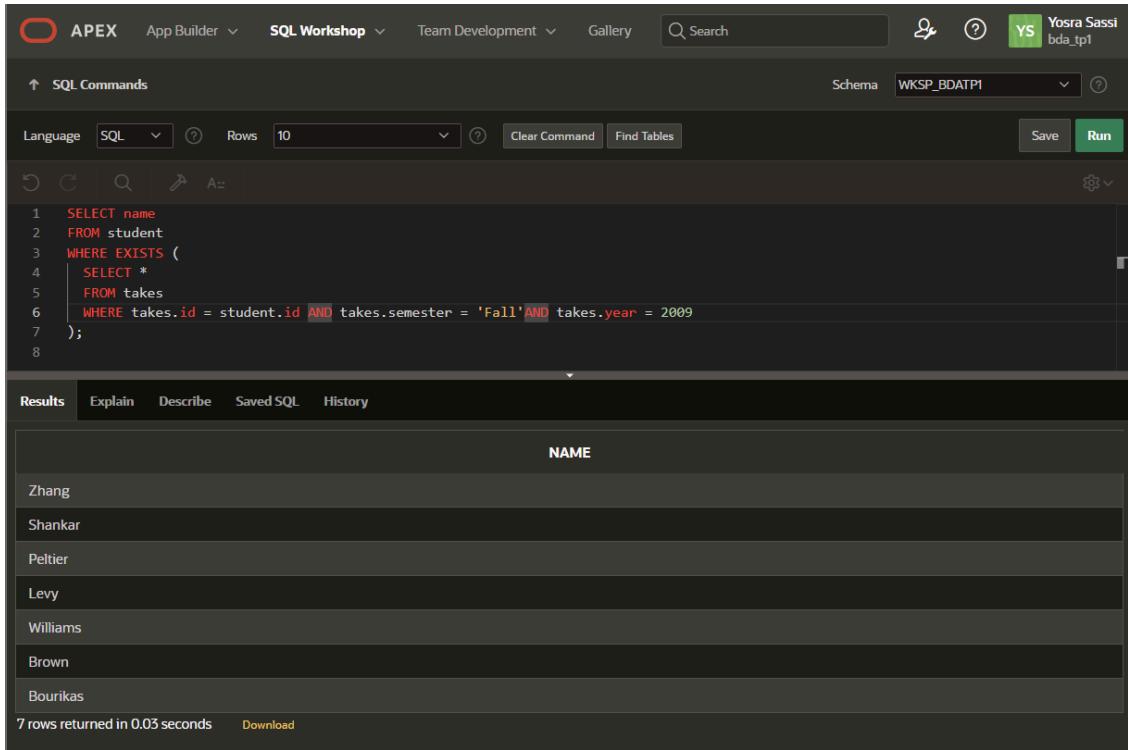
```

The results table displays the names of students who took courses in Fall 2009:

NAME
Brown
Zhang
Levy
Bourikas
Shankar
Peltier
Williams

7 rows returned in 0.03 seconds [Download](#)

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



The screenshot shows the Oracle SQL Workshop interface. The schema is set to WKSP_BDATP1. The SQL command entered is:

```

1 SELECT name
2 FROM student
3 WHERE EXISTS (
4   SELECT *
5   FROM takes
6   WHERE takes.id = student.id AND takes.semester = 'Fall' AND takes.year = 2009
7 );
8

```

The results table displays the names of students who took courses in Fall 2009:

NAME
Zhang
Shankar
Peltier
Levy
Williams
Brown
Bourikas

7 rows returned in 0.03 seconds [Download](#)

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

The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user profile are also at the top. The main area is titled "SQL Commands". The code input field contains the following query:

```

1 SELECT A.name, B.name
2 FROM (SELECT student.id, student.name, takes.course_id, takes.sec_id, takes.semester, takes.year FROM student
3       JOIN takes ON student.id = takes.id) A,
4       (SELECT student.id, student.name, takes.course_id, takes.sec_id, takes.semester, takes.year FROM student
5       JOIN takes ON student.id = takes.id) B
6 WHERE A.course_id = B.course_id AND A.sec_id = B.sec_id AND A.semester = B.semester
7       AND A.year = B.year AND A.id < B.id AND A.name < B.name
8 GROUP BY A.name, B.name
9 HAVING COUNT(*) >= 1;
10

```

Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing a table with two columns: NAME and NAME. The data is as follows:

NAME	NAME
Levy	Williams
Brown	Shankar
Bourikas	Brown
Levy	Zhang
Brown	Williams
Bourikas	Williams
Bourikas	Zhang
Shankar	Williams
Brown	Zhang
Williams	Zhang

Below the table, a message says "More than 10 rows available. Increase rows selector to view more rows." and "10 rows returned in 0.05 seconds". There is a "Download" link.

14. Afficher pour chaque enseignant, qui a effectivement assuré un cours, le nombre total d'étudiant 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.

The screenshot shows the Oracle SQL Workshop interface. The top navigation and user profile are identical to the previous screenshot. The main area is titled "SQL Commands". The code input field contains the following query:

```

1 SELECT teacher.name, COUNT(*) FROM takes
2 JOIN teaches ON takes.course_id = teaches.course_id AND takes.sec_id = teaches.sec_id
3           AND takes.semester = teaches.semester AND takes.year = teaches.year
4 JOIN teacher ON teaches.id = teacher.id
5 GROUP BY teacher.name, teacher.id
6 ORDER BY COUNT(*) DESC;
7

```

Results Explain Describe Saved SQL History

NAME	NAME
Levy	Williams
Brown	Shankar
Bourikas	Brown
Levy	Zhang
Brown	Williams
Bourikas	Williams
Bourikas	Zhang
Shankar	Williams
Brown	Zhang
Williams	Zhang

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

10 rows returned in 0.05 seconds [Download](#)

15. Afficher pour chaque enseignant, même s'il n'a pas assuré de cours, le nombre total d'étudiant 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.

APEX App Builder SQL Workshop Team Development Gallery Search Schema WKSP_BDATPI Yosra Sassi bda_tp1

↑ SQL Commands

Language SQL Rows 10 Clear Command Find Tables Save Run

```

1 SELECT teacher.name, COUNT(teaches.course_id)
2 FROM takes
3 JOIN teaches ON takes.course_id = teaches.course_id AND takes.sec_id = teaches.sec_id
4 AND takes.semester = teaches.semester AND takes.year = teaches.year
5 RIGHT OUTER JOIN teacher ON teaches.id = teacher.id
6 GROUP BY teacher.name, teacher.id
7 ORDER BY COUNT(teaches.course_id) DESC;
8

```

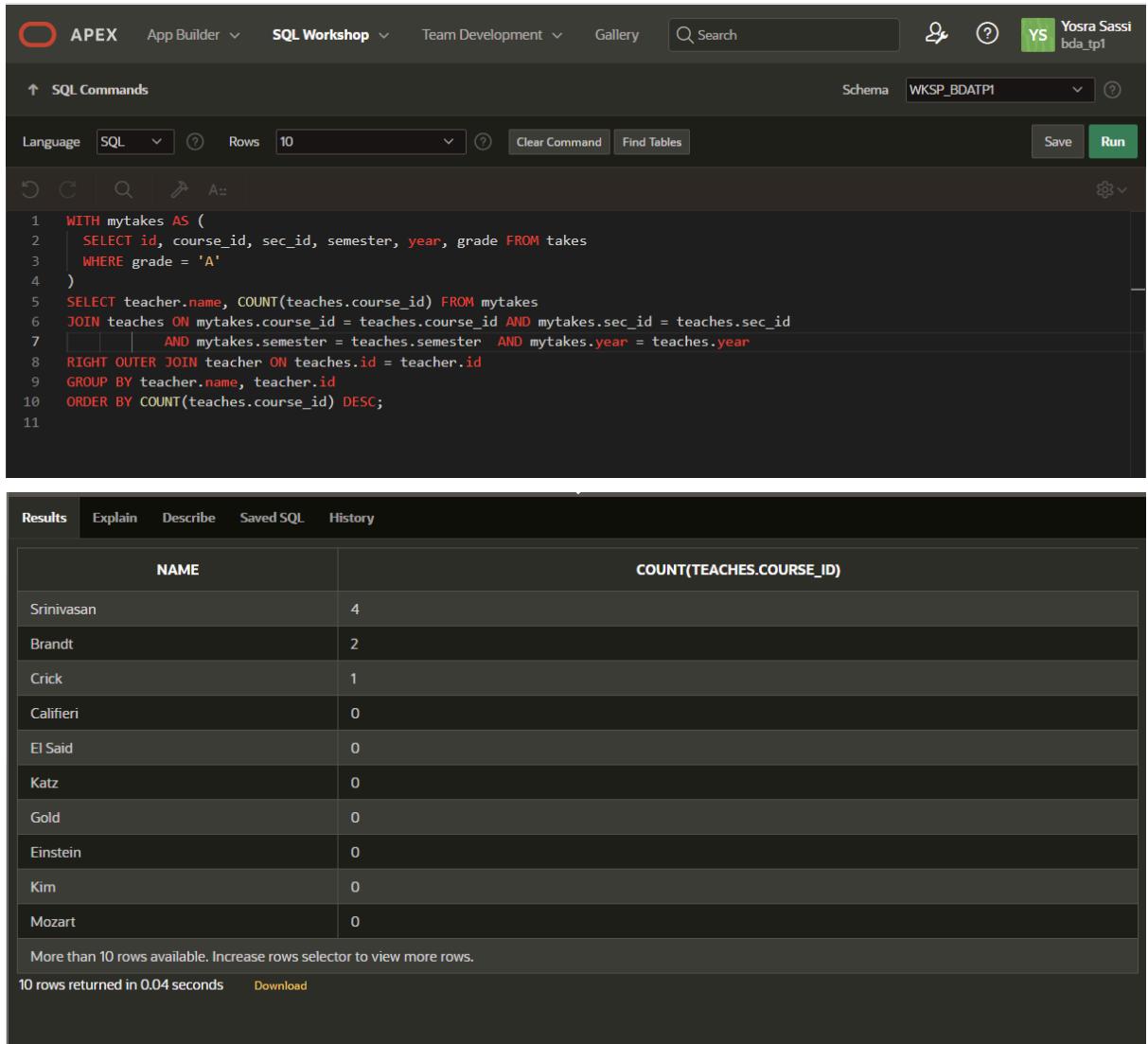
Results Explain Describe Saved SQL History

NAME	COUNT(TEACHES.COURSE_ID)
Srinivasan	10
Brandt	3
Katz	2
Crick	2
El Said	1
Einstein	1
Kim	1
Wu	1
Mozart	1
Gold	0

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

10 rows returned in 0.08 seconds [Download](#)

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



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', 'Gallery', a search bar, and a user profile for 'Yosra Sassi bda_tp1'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_BDATP1'. Below the command input field, there are buttons for 'Save' and 'Run'. The SQL code is as follows:

```

1 WITH mytakes AS (
2   SELECT id, course_id, sec_id, semester, year, grade FROM takes
3   WHERE grade = 'A'
4 )
5 SELECT teacher.name, COUNT(teaches.course_id) FROM mytakes
6 JOIN teaches ON mytakes.course_id = teaches.course_id AND mytakes.sec_id = teaches.sec_id
7 AND mytakes.semester = teaches.semester AND mytakes.year = teaches.year
8 RIGHT OUTER JOIN teacher ON teaches.id = teacher.id
9 GROUP BY teacher.name, teacher.id
10 ORDER BY COUNT(teaches.course_id) DESC;
11

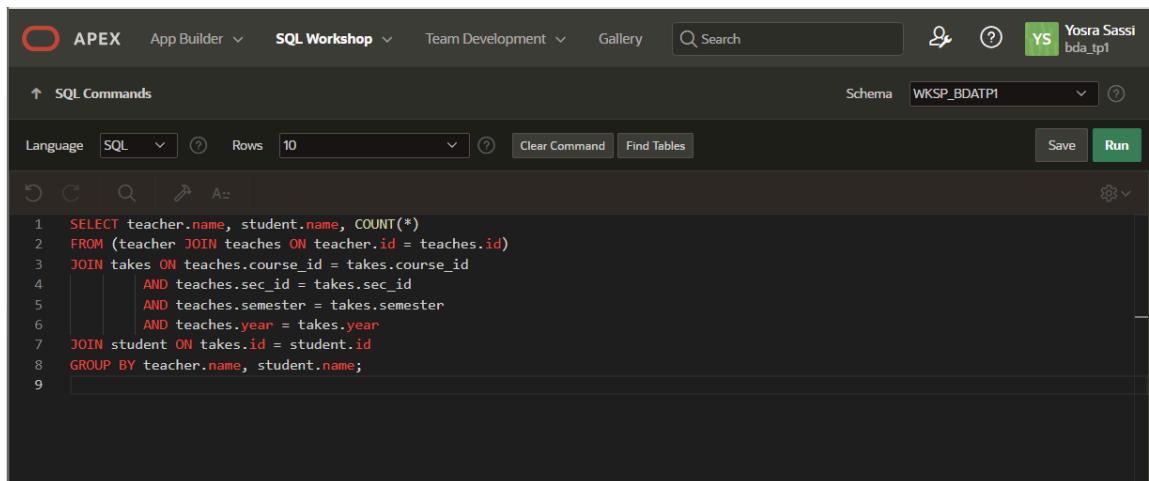
```

The results section displays a table with two columns: 'NAME' and 'COUNT(TEACHES.COURSE_ID)'. The data is as follows:

NAME	COUNT(TEACHES.COURSE_ID)
Srinivasan	4
Brandt	2
Crick	1
Califieri	0
El Said	0
Katz	0
Gold	0
Einstein	0
Kim	0
Mozart	0

Text at the bottom of the results pane indicates: "More than 10 rows available. Increase rows selector to view more rows." and "10 rows returned in 0.04 seconds".

17. Afficher toutes les paires enseignant-élèves où 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.



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', 'Gallery', a search bar, and a user profile for 'Yosra Sassi bda_tp1'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_BDATP1'. Below the command input field, there are buttons for 'Save' and 'Run'. The SQL code is as follows:

```

1 SELECT teacher.name, student.name, COUNT(*)
2 FROM (teacher JOIN teaches ON teacher.id = teaches.id)
3 JOIN takes ON teaches.course_id = takes.course_id
4 AND teaches.sec_id = takes.sec_id
5 AND teaches.semester = takes.semester
6 AND teaches.year = takes.year
7 JOIN student ON takes.id = student.id
8 GROUP BY teacher.name, student.name;
9

```

Results Explain Describe Saved SQL History

NAME	NAME	COUNT(*)
Brandt	Shankar	1
Wu	Chavez	1
El Said	Brandt	1
Einstein	Peltier	1
Brandt	Brown	1
Srinivasan	Bourikas	2
Mozart	Sanchez	1
Kim	Aoi	1
Srinivasan	Shankar	3
Brandt	Williams	1

More than 10 rows available. Increase rows selector to view more rows.
10 rows returned in 0.01 seconds [Download](#)

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

APEX App Builder SQL Workshop Team Development Gallery Search Yosra Sassi bda_tp1

↑ SQL Commands Schema WKSP_BDATP1

Language SQL Rows 10 Clear Command Find Tables Save Run

```

1  SELECT mytable.tn, mytable.sn
2  FROM ( SELECT teacher.name tn, student.name sn, COUNT(*) tc
3    FROM teacher JOIN teaches ON teacher.id = teaches.id
4    JOIN takes ON teaches.course_id = takes.course_id
5    AND teaches.sec_id = takes.sec_id AND teaches.semester = takes.semester | AND teaches.year = takes.year
6    JOIN student ON takes.id = student.id
7    GROUP BY teacher.name, student.name
8  ) mytable
9  WHERE tc >= 2;
10

```

Results Explain Describe Saved SQL History

TN	SN
Srinivasan	Bourikas
Srinivasan	Shankar
Crick	Tanaka
Katz	Levy
Srinivasan	Zhang

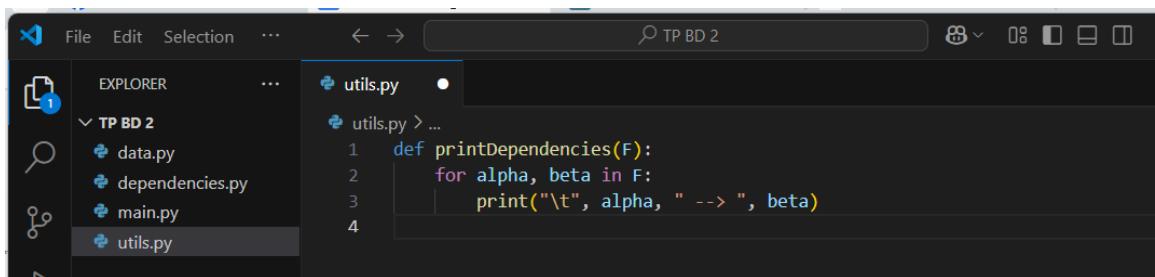
5 rows returned in 0.04 seconds [Download](#)

Exercice 2 – Requêtes SQL

Cet exercice consiste à manipuler des dépendances fonctionnelles, tester des propriétés comme super-clé et clé candidate, vérifier si une relation est en BCNF, et appliquer une **décomposition** en BCNF en Python.

Fonctions disponibles :

- Affichage des dépendances et relations :



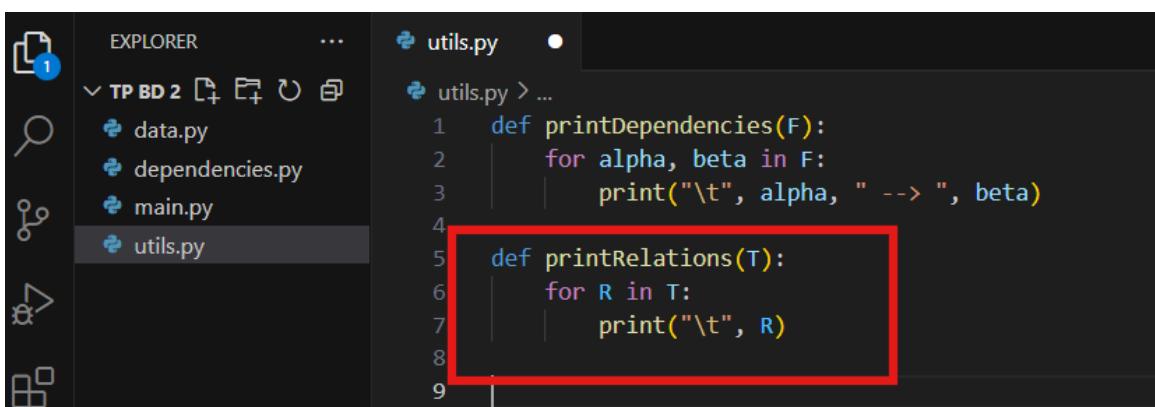
```

File Edit Selection ...
EXPLORER ...
TP BD 2 ...
data.py
dependencies.py
main.py
utils.py

utils.py ...
1 def printDependencies(F):
2     for alpha, beta in F:
3         print("\t", alpha, " --> ", beta)
4

```

- Affichage des relations :



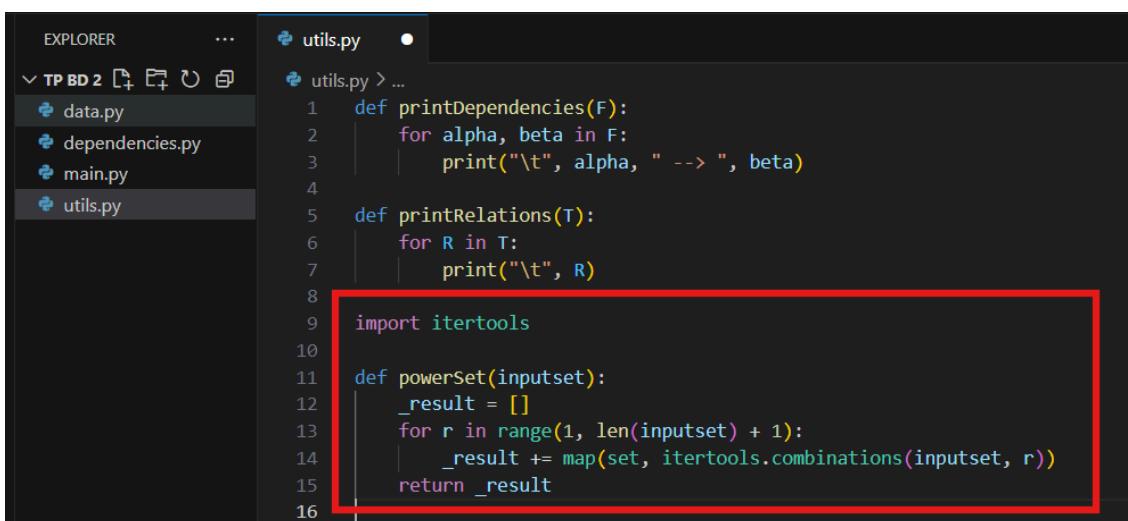
```

File Edit Selection ...
EXPLORER ...
TP BD 2 ...
data.py
dependencies.py
main.py
utils.py

utils.py ...
1 def printDependencies(F):
2     for alpha, beta in F:
3         print("\t", alpha, " --> ", beta)
4
5 def printRelations(T):
6     for R in T:
7         print("\t", R)
8
9

```

- Génération de tous les sous-ensembles (power set) :



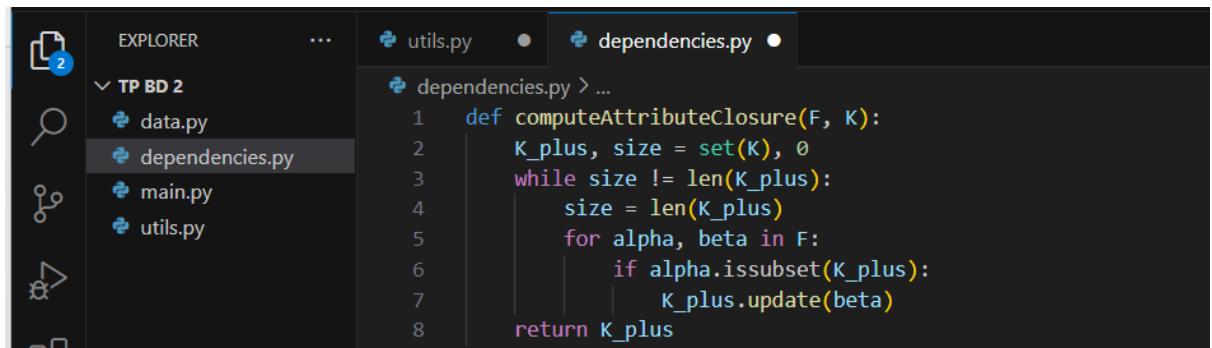
```

File Edit Selection ...
EXPLORER ...
TP BD 2 ...
data.py
dependencies.py
main.py
utils.py

utils.py ...
1 def printDependencies(F):
2     for alpha, beta in F:
3         print("\t", alpha, " --> ", beta)
4
5 def printRelations(T):
6     for R in T:
7         print("\t", R)
8
9 import itertools
10
11 def powerSet(inputset):
12     _result = []
13     for r in range(1, len(inputset) + 1):
14         _result += map(set, itertools.combinations(inputset, r))
15
16

```

4. Calcul de la fermeture d'un ensemble d'attributs :



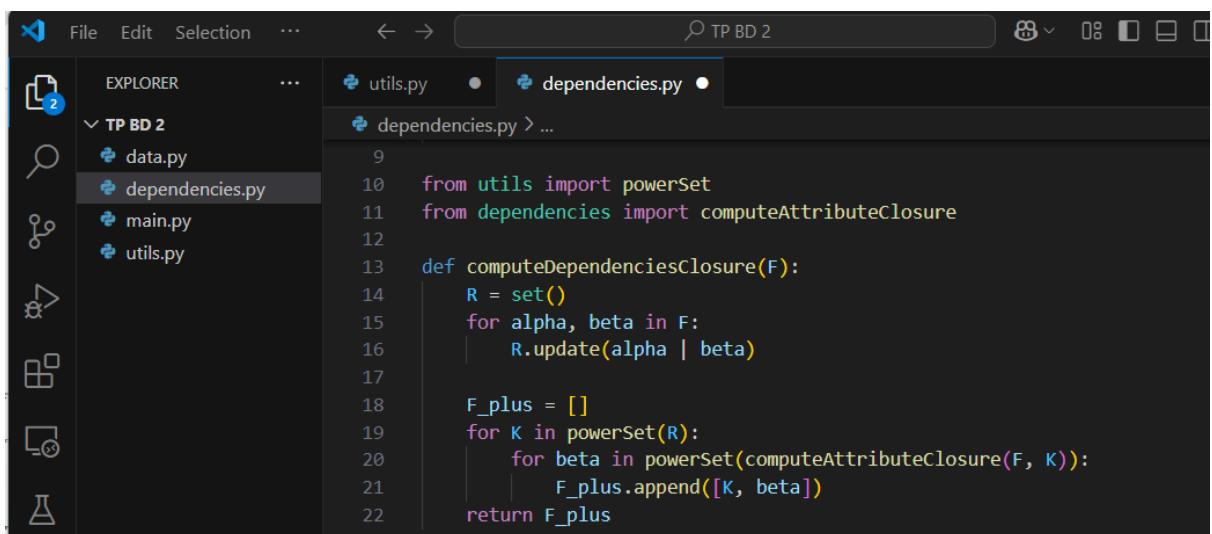
```

EXPLORER ... utils.py dependencies.py
TP BD 2
data.py
dependencies.py > ...
main.py
utils.py

1 def computeAttributeClosure(F, K):
2     K_plus, size = set(K), 0
3     while size != len(K_plus):
4         size = len(K_plus)
5         for alpha, beta in F:
6             if alpha.issubset(K_plus):
7                 K_plus.update(beta)
8
return K_plus

```

5. Calcul de la fermeture d'un ensemble de dépendances :



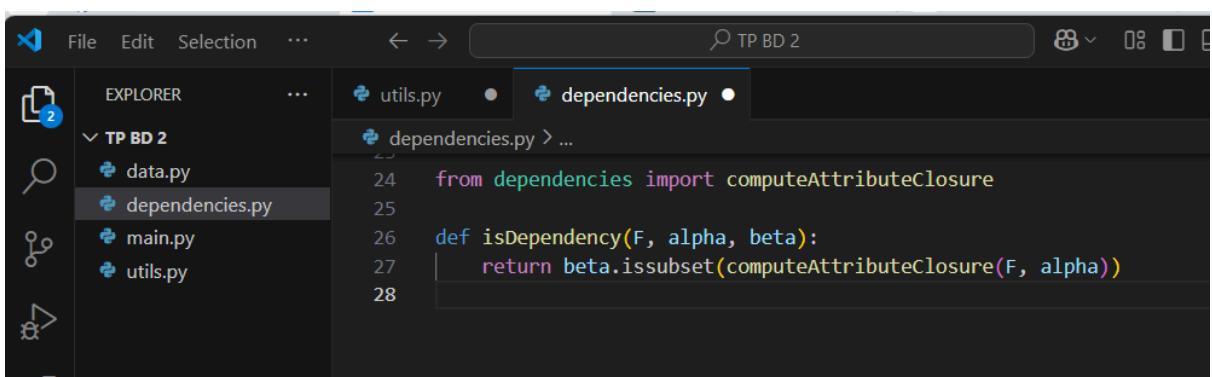
```

File Edit Selection ... ← → TP BD 2
EXPLORER ... utils.py dependencies.py
TP BD 2
data.py
dependencies.py > ...
main.py
utils.py

9
10 from utils import powerSet
11 from dependencies import computeAttributeClosure
12
13 def computeDependenciesClosure(F):
14     R = set()
15     for alpha, beta in F:
16         R.update(alpha | beta)
17
18     F_plus = []
19     for K in powerSet(R):
20         for beta in powerSet(computeAttributeClosure(F, K)):
21             F_plus.append([K, beta])
22
return F_plus

```

6. Vérification d'une dépendance déduite ($\alpha \rightarrow \beta ?$) :



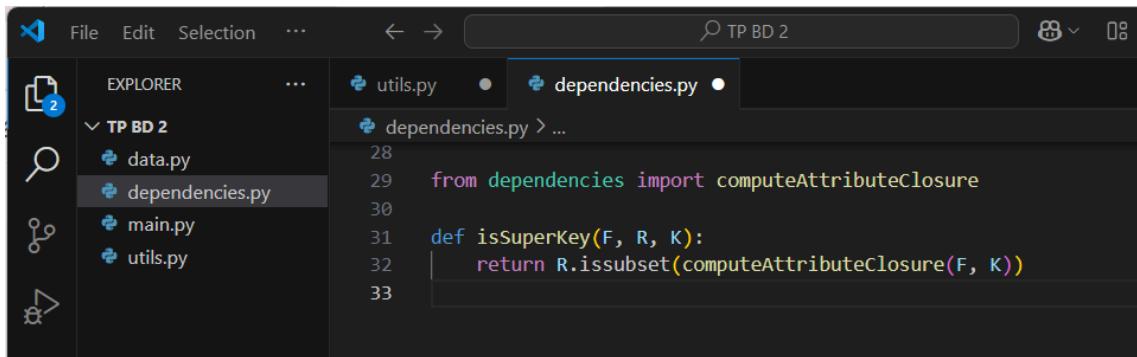
```

File Edit Selection ... ← → TP BD 2
EXPLORER ... utils.py dependencies.py
TP BD 2
data.py
dependencies.py > ...
main.py
utils.py

24 from dependencies import computeAttributeClosure
25
26 def isDependency(F, alpha, beta):
27     return beta.issubset(computeAttributeClosure(F, alpha))
28

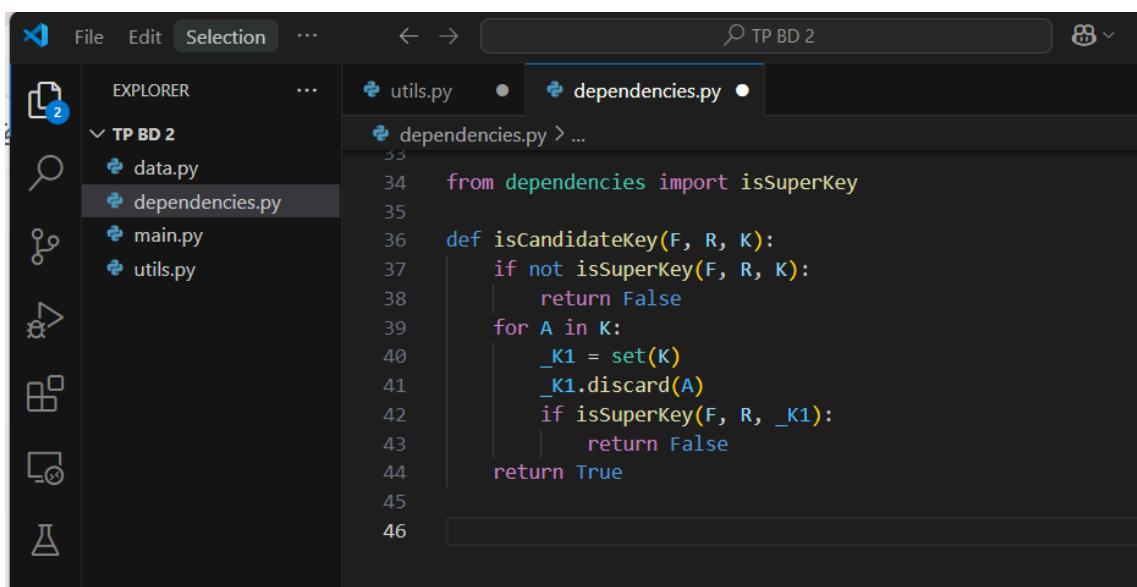
```

7. Vérification si un ensemble est une super-clé :



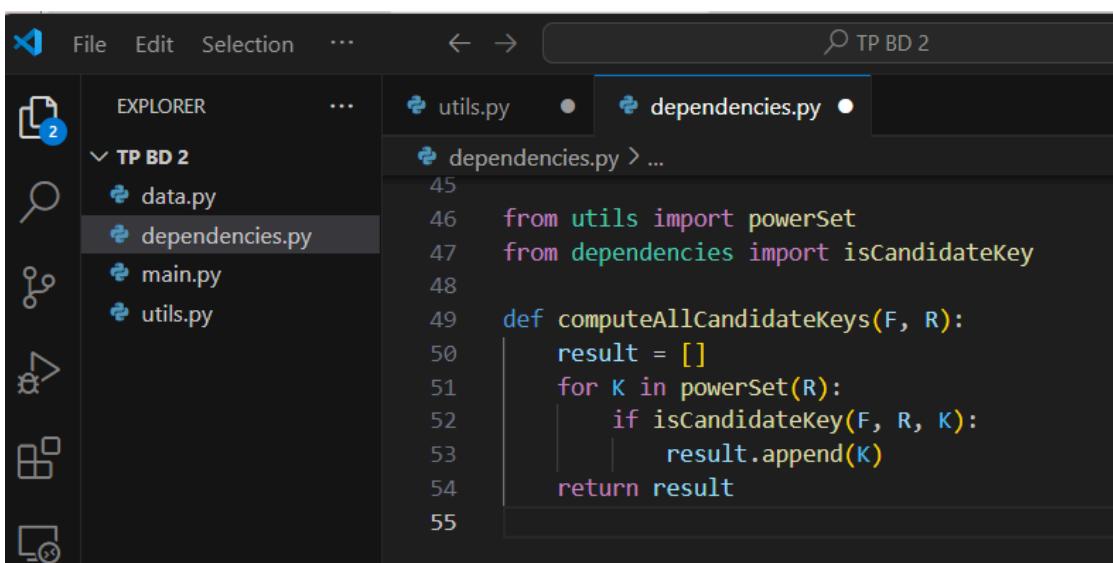
```
File Edit Selection ... ← → ⌂ TP BD 2
EXPLORER ... utils.py ● dependencies.py ●
TP BD 2
data.py
dependencies.py > ...
28
29     from dependencies import computeAttributeClosure
30
31 def isSuperKey(F, R, K):
32     return R.issubset(computeAttributeClosure(F, K))
33
```

8. Vérification si un ensemble est une clé candidate :



```
File Edit Selection ... ← → ⌂ TP BD 2
EXPLORER ... utils.py ● dependencies.py ●
TP BD 2
data.py
dependencies.py > ...
33
34     from dependencies import isSuperKey
35
36 def isCandidateKey(F, R, K):
37     if not isSuperKey(F, R, K):
38         return False
39     for A in K:
40         _K1 = set(K)
41         _K1.discard(A)
42         if isSuperKey(F, R, _K1):
43             return False
44     return True
45
46
```

9. Calcul de toutes les clés candidates :



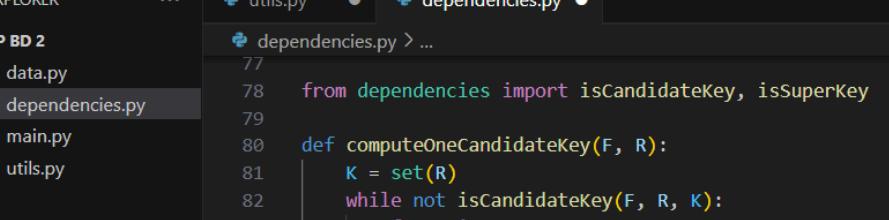
```
File Edit Selection ... ← → ⌂ TP BD 2
EXPLORER ... utils.py ● dependencies.py ●
TP BD 2
data.py
dependencies.py > ...
45
46     from utils import powerSet
47     from dependencies import isCandidateKey
48
49 def computeAllCandidateKeys(F, R):
50     result = []
51     for K in powerSet(R):
52         if isCandidateKey(F, R, K):
53             result.append(K)
54     return result
55
```

10. Calcul de toutes les super-clés :

```
from utils import powerset
from dependencies import isSuperKey

def computeAllSuperKeys(F, R):
    result = []
    for K in powerSet(R):
        if isSuperKey(F, R, K):
            result.append(K)
    return result
```

11. Calcul d'une seule clé candidate :



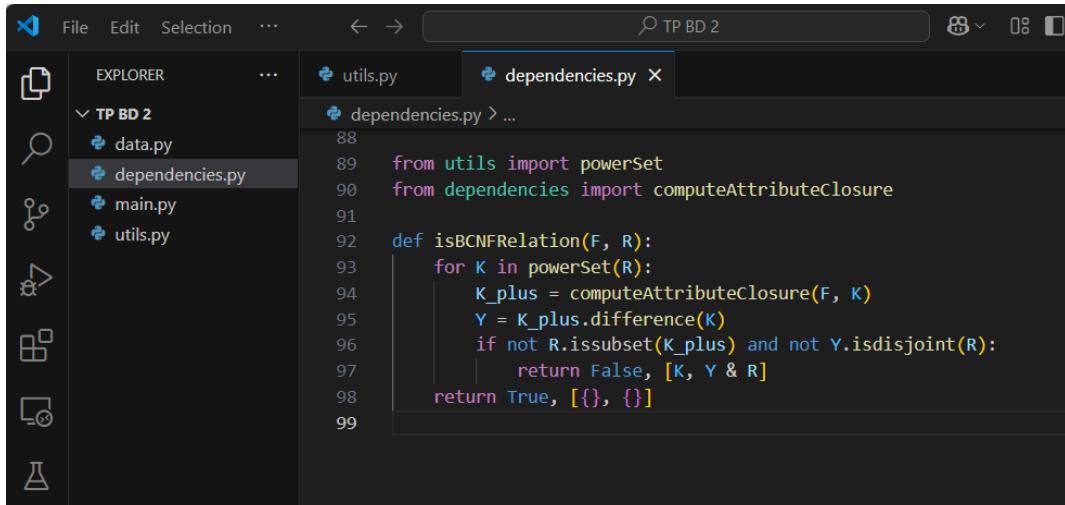
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a tree view with a folder named "TP BD 2" containing files: data.py, dependencies.py, main.py, and utils.py. The file "dependencies.py" is currently selected.
- Code Editor (Right):** Displays the content of the "dependencies.py" file. The code defines a function `computeOneCandidateKey` that iterates over a set `K` and removes elements that are superkeys of the current element `A`. The code uses Python syntax with imports from the "dependencies" module.

```
from dependencies import isCandidateKey, isSuperKey

def computeOneCandidateKey(F, R):
    K = set(R)
    while not isCandidateKey(F, R, K):
        for A in K:
            if isSuperKey(F, R, K.difference({A})):
                K.remove(A)
        break
    return K
```

12. Vérification si une relation est en BCNF :



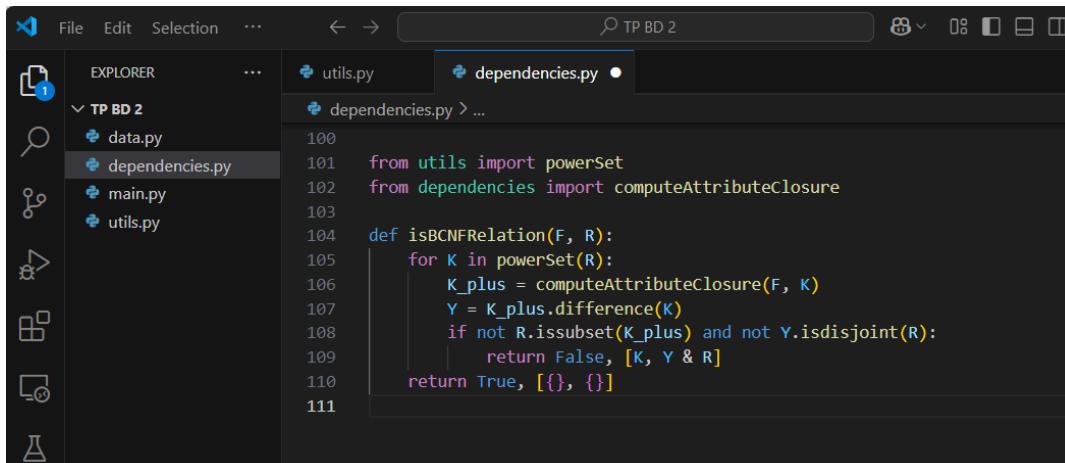
```

File Edit Selection ...
EXPLORER ...
TP BD 2
data.py
dependencies.py > ...
main.py
utils.py

dependencies.py > ...
88
89     from utils import powerset
90     from dependencies import computeAttributeClosure
91
92     def isBCNFRelation(F, R):
93         for K in powerSet(R):
94             K_plus = computeAttributeClosure(F, K)
95             Y = K_plus.difference(K)
96             if not R.issubset(K_plus) and not Y.isdisjoint(R):
97                 return False, [K, Y & R]
98
99     return True, [{}], [{}]

```

13. Vérification si un schéma complet est en BCNF :



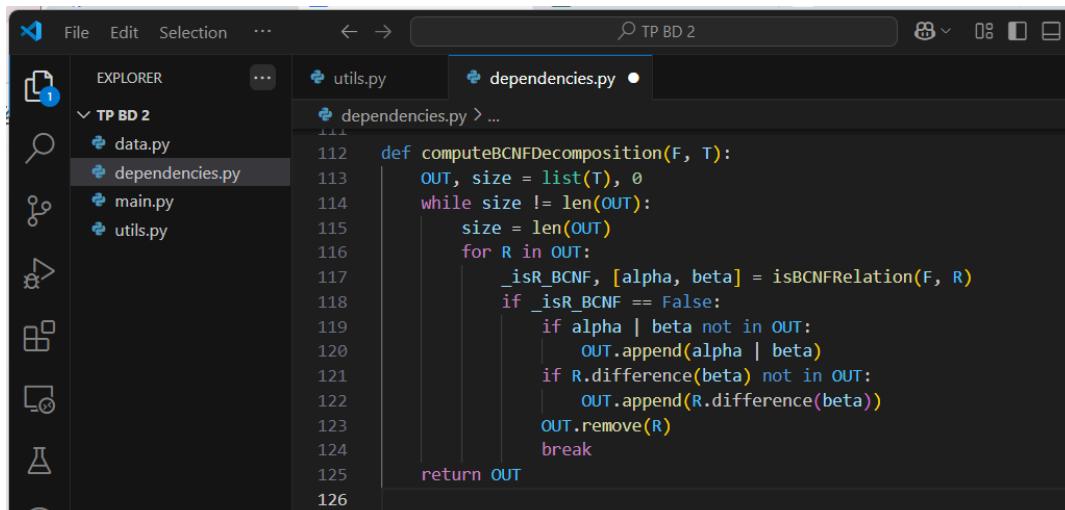
```

File Edit Selection ...
EXPLORER ...
TP BD 2
data.py
dependencies.py > ...
main.py
utils.py

dependencies.py > ...
100
101     from utils import powerSet
102     from dependencies import computeAttributeClosure
103
104     def isBCNFRelation(F, R):
105         for K in powerSet(R):
106             K_plus = computeAttributeClosure(F, K)
107             Y = K_plus.difference(K)
108             if not R.issubset(K_plus) and not Y.isdisjoint(R):
109                 return False, [K, Y & R]
110
111     return True, [{}], [{}]

```

14. Algorithme de décomposition d'un schéma en BCNF :



```

File Edit Selection ...
EXPLORER ...
TP BD 2
data.py
dependencies.py > ...
main.py
utils.py

dependencies.py > ...
112     def computeBCNFDecomposition(F, T):
113         OUT, size = list(T), 0
114         while size != len(OUT):
115             size = len(OUT)
116             for R in OUT:
117                 _isR_BCNF, [alpha, beta] = isBCNFRelation(F, R)
118                 if _isR_BCNF == False:
119                     if alpha | beta not in OUT:
120                         OUT.append(alpha | beta)
121                     if R.difference(beta) not in OUT:
122                         OUT.append(R.difference(beta))
123                     OUT.remove(R)
124             return OUT
125
126

```

Script de test

Fichier : data.py : **Ce fichier sert de base de données locale** pour toutes les fonctions.

```

EXPLORER      ...
TP BD 2
data.py
dependencies.py
main.py
utils.py

utils.py    dependencies.py    data.py    main.py

1 # data.py
2
3 myrelations = [
4     {'A', 'B', 'C', 'G', 'H', 'I'},
5     {'X', 'Y'}
6 ]
7
8 mydependencies = [
9     [{ 'A'}, { 'B'}],
10    [{ 'A'}, { 'C'}],
11    [{ 'C', 'G'}, { 'H'}],
12    [{ 'C', 'G'}, { 'I'}],
13    [{ 'B'}, { 'H'}]
14 ]
15

```

Fichier : main.py : Ce fichier aide à **afficher** et **manipuler** facilement les structures de données.

```

File Edit Selection ...
TP BD 2
utils.py    dependencies.py    data.py    main.py

main.py
1 # Importation des données de test : relations et dépendances
2 from data import myrelations, mydependencies
3
4 # Importation des fonctions d'affichage
5 from utils import printDependencies, printRelations
6
7 # Importation des fonctions de traitement logique
8 from dependencies import (
9     isSuperKey,
10    computeAllCandidateKeys,
11    isBCNFRrelation,
12    computeBCNFDecomposition
13 )
14
15 # ▶ Affiche la liste des relations définies (chaque ensemble d'attributs)
16 print("Relations :")
17 printRelations(myrelations)
18 # ▶ Affiche la liste des dépendances fonctionnelles sous forme A --> B
19 print("Dépendances :")
20 printDependencies(mydependencies)
21 # ▶ Test si l'ensemble {'A'} est une super-clé dans la première relation
22 print("\nEst-ce que {'A'} est une super-clé ?")
23 print(isSuperKey(mydependencies, myrelations[0], {'A'}))
24 # ▶ Calcule et affiche toutes les clés candidates de la première relation
25 print("\nClés candidates :")
26 print(computeAllCandidateKeys(mydependencies, myrelations[0]))
27 # ▶ Vérifie si la première relation est en forme normale de Boyce-Codd (BCNF)
28 print("\nRelation en BCNF ?")
29 print(isBCNFRrelation(mydependencies, myrelations[0]))
30 # ▶ Applique la décomposition BCNF sur toutes les relations
31 print("\nDécomposition BCNF :")
32 print(computeBCNFDecomposition(mydependencies, myrelations))
33

```

Étape finale : Lancer le test

Dans ton terminal, exécute le fichier : **python main.py**

```
● PS C:\Users\Lenovo\Desktop\Sup Galilée - Cours\Base de données\TP BD 2> python main.py
Relations :
    {'A', 'H', 'C', 'I', 'B', 'G'}
    {'Y', 'X'}
Dépendances :
    {'A'} --> {'B'}
    {'A'} --> {'C'}
    {'C', 'G'} --> {'H'}
    {'C', 'G'} --> {'I'}
    {'B'} --> {'H'}

Est-ce que {'A'} est une super-clé ?
False

Clés candidates :
[{'A', 'G'}]

Relation en BCNF ?
(False, [{'A'}, {'H', 'C', 'B'}])

Décomposition BCNF :
[['Y', 'X'], {'I', 'A', 'G'}, {'H', 'B'}, {'A', 'C', 'B'}]
○ PS C:\Users\Lenovo\Desktop\Sup Galilée - Cours\Base de données\TP BD 2> ]
```

→ Les deux relations affichées représentent des schémas relationnels sur lesquels s'appliquent les dépendances fonctionnelles listées.

→ Le test effectué montre que l'attribut {A} n'est pas une super-clé, car il ne permet pas à lui seul de déterminer tous les attributs de la relation.