Bases de Données Fall, 2024

TD7

1. Schema:

```
games (gamescore integer)
```

Each record contains a game score achieved in Tetris.

Chaque ligne contient un score de jeu obtenu dans Tetris.

Write a query to list the rank and the gamescore at that rank. Assume all game scores are distinct. Écrivez une requête pour lister le rang et le score de jeu à ce rang. Supposons que tous les scores de jeu sont distincts.

Example:

gamescore
53
11
99
45
59
46

(a) input

games	rank
53	4
11	1
99	6
45	2
59	5
46	3

(b) output

```
SELECT A.gamescore, count(*)
FROM games as A CROSS JOIN games as B
WHERE B.gamescore <= A.gamescore
GROUP BY A.gamescore</pre>
```

```
exam (time integer, questions integer)
```

Each record contains an hour, and the number of questions done during that hour. Chaque ligne contient une heure et le nombre de questions complétées pendant cette heure.

Write a query to list, for each hour, the **total** number of questions completed after that hour has passed. Écrivez une requête pour lister, pour chaque heure, le nombre total de questions complétées après l'expiration de cette heure.

Example:

time	questions
1	5
2	2
3	8
4	14
5	0
6	1

(c) input

time	questions
1	5
2	7
3	15
4	29
5	29
6	30

(d) output

```
SELECT E1.time, sum(E2.questions)
FROM exam as E1, exam as E2
WHERE E2.time <= E1.time
GROUP BY E1.time</pre>
```

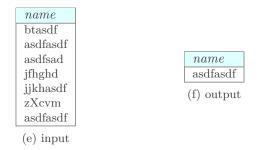
```
company (name text)
```

Each record contains the name of a person.

Chaque ligne contient le nom d'une personne.

Write a query to list the names that appear more than once. Écrivez une requête pour lister les noms qui apparaissent plus d'une fois.

Example:



```
SELECT name
FROM company
GROUP BY name
HAVING count(*) > 1
```

```
\mathbf{customer} \ (\mathrm{id} \ \mathrm{integer}, \ \underline{\mathsf{name}} \ \mathrm{text})
```

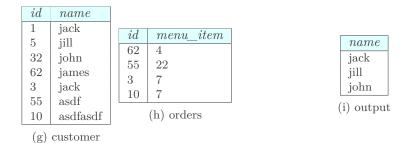
Each record contains the name of a person, and their unique id. Chaque ligne contient le nom d'une personne et son id unique.

```
orders (id integer, menu_item integer)
```

Each record contains the id and the menu item that that person has ordered. Chaque ligne contient le id et l'élément de menu que cette personne a commandé.

Write a query to list the names of the people that have not ordered anything. Écrivez une requête pour lister les noms des personnes qui n'ont rien commandé.

Example:



```
SELECT name
FROM customer
WHERE id NOT IN (SELECT id
FROM orders);

Second solution:

(SELECT name
FROM customer)

EXCEPT

(SELECT distinct name
FROM customer INNER JOIN orders USING (id))
```

```
temperature (day_no integer, temp integer)
```

Each record contains the day of the month (1-30) and its temperature. Chaque ligne contient le jour du mois (1-30) et sa température.

Write a query to list the days where the temperature is higher than that of the *previous* day. Écrivez une requête pour lister les jours où la température est supérieure à celle du jour *précédent*.

Example:

temp
27
30
31
30
28
29
27
32

(j) temperature

$d\epsilon$	iy_no
2	
3	
6	
8	
(k)	output

```
SELECT A.day_no
FROM temperature AS A CROSS JOIN temperature AS B
WHERE B.temp < A.temp AND B.day_no = A.day_no-1;

Solution 2:

SELECT A.day_no
FROM temperature AS A
WHERE A.temp >= (SELECT MAX(B.temp)
FROM temperature AS B
WHERE B.temp < A.temp
AND B.day_no = A.day_no-1);</pre>
```

```
server (id integer, device integer, login_date date)
```

Each record contains login student id, laptop number they used, and the date they logged in.
Chaque ligne contient l'identifiant de connexion de l'étudiant, le numéro de l'ordinateur portable qu'il a utilisé et la date à laquelle il s'est connecté.

Write a query to list, for each student, the first date they logged in. Écrivez une requête pour lister, pour chaque élève, la première date de sa connexion.

Example:

id	device	$login_date$
463	4	30-01-2023
422	1	20-01-2023
463	2	11-01-2023
834	6	16-01-2023
585	8	22-01-2023
936	1	23-01-2023
834	4	08-01-2023
463	3	21-01-2023

(l) server

id	$login_date$
463	11-01-2023
422	20-01-2023
834	08-01-2023
936	23-01-2023
585	22-01-2023

(m) output

```
SELECT id, min(login_date)
FROM server
GROUP BY id

SELECT A.id, A.login_date
FROM server AS A
WHERE A.login_date <= ALL (SELECT B.login_date
FROM server AS B
WHERE B.id = A.id)
```

```
server (id integer, device integer, login_date date)
```

Each record contains login student id, laptop number they used, and the date they logged in.
Chaque ligne contient l'identifiant de connexion de l'étudiant, le numéro de l'ordinateur portable qu'il a utilisé et la date à laquelle il s'est connecté.

Write a query to list, for each student, the device they used the first time they logged in. Écrivez une requête pour lister, pour chaque élève, l'appareil qu'il a utilisé la première fois qu'il s'est connecté.

Example:

id	device	login_date
463	4	30-01-2023
422	1	20-01-2023
463	2	11-01-2023
834	6	16-01-2023
585	8	22-01-2023
936	1	23-01-2023
834	4	08-01-2023
463	3	21-01-2023

(n)	server

id	device
463	2
422	1
834	4
936	1
585	8

(o) output

Solution.

Solution 2:

```
employee (id integer, name text, manager_id integer)
```

Each record contains employee id, name and the id of the manager.

Chaque ligne contient l'identifiant de l'employé, le nom et l'identifiant du responsable.

```
bonus (id integer, amount integer)
```

Each record contains employee id, and the bonus they received this year.

Chaque ligne contient l'identifiant de l'employé et la prime qu'il a reçue cette année.

Write a query to list the name and bonus of all employees who received a bonus of at most 1000. Écrivez une requête pour lister le nom et le bonus de tous les employés qui ont reçu un bonus d'au plus 1000.

Example:

id	name	$manager_id$
2	john	4
6	james	9
4	jamil	9
9	jean	null
7	joe	2

(p) employee

id	amount
4	500
9	2000

(q) bonus

name	prime
john	null
jamil	500
james	null
joe	null

(r) output

```
SELECT employee.name, amount AS prime
FROM employee LEFT OUTER JOIN bonus ON employee.id = bonus.id
WHERE bonus.amount <= 1000 OR amount IS null;</pre>
```

```
tree (node_id integer, parent_id integer)
```

Each record contains a binary tree node id, and the id of its parent.

Chaque ligne contient un nœud d'arbre binaire id et le id de son parent.

Write a query to list, for each node, if it is a 'Root' node, or an 'Inner' node or a 'Leaf' node. Écrivez une requête pour lister, pour chaque nœud, s'il s'agit d'un nœud « Racine », d'un nœud « Interne » ou d'un nœud « Feuille ».

Example:

$node_id$	$parent_id$	
1	null	
5	1	
7	1	
2	5	
3	7	
(s) tree		

(s) tree

$node_id$	type
1	Root
5	Inner
7	Inner
2	Leaf
3	Leaf

(t) output

Solution.

```
SELECT node_id, 'Root' AS type
FROM tree
WHERE parent_id IS null

UNION

SELECT node_id, 'Inner' AS type
FROM tree
WHERE node_id IN (SELECT parent_id FROM tree)
    AND parent_id IS NOT null

UNION

SELECT node_id, 'Leaf' AS type
FROM tree
WHERE node_id NOT IN (SELECT parent_id
FROM tree
WHERE parent_id IS NOT null)
```

Solution 2:

```
SELECT node_id,

CASE

WHEN parent_id IS null THEN 'Root'

WHEN parent_id IS NOT null

AND node_id IN (SELECT parent_id FROM tree) THEN 'Inner'

ELSE 'Leaf'

END AS type

FROM tree;
```