

Chap 3

Introduction aux Bases de données

Introduction au SQL (MySQL)

TC Sup'Num

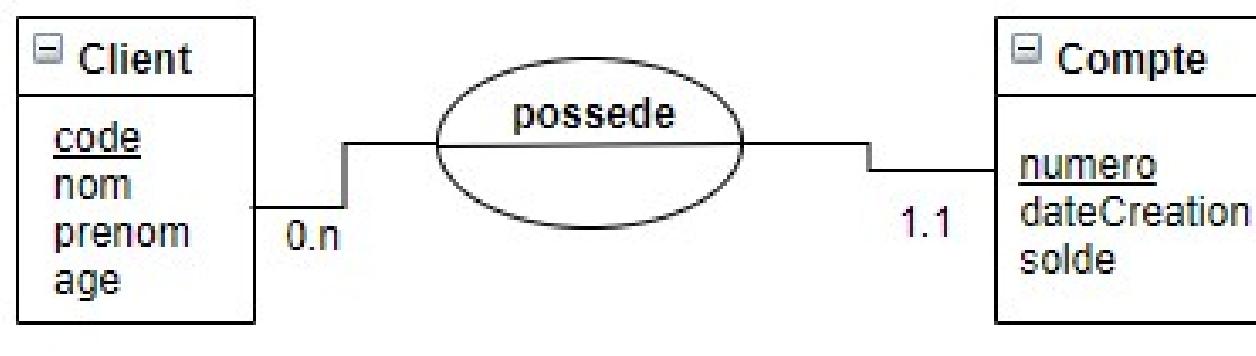
Exemple

❖ Expression des besoins

Créer une BD pour la gestion des comptes client d'une banque. Une personne peut avoir plusieurs comptes et un compte appartient à une seule personne.



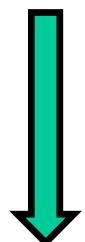
❖ Conception



❖ Schéma Relationnel

Client(code, nom, prenom, age)

Compte(numero, codeClient#, dateCreation, solde)



❖ Création de la BD

Création des tables: Create table

Client(code, nom, prenom, age)

Compte(numero, codeClient#, dateCreation, solde)

```
CREATE TABLE Client( code INTEGER(5),  
                      nom VARCHAR(20),  
                      prenom VARCHAR(20),  
                      age INTEGER(2),  
                      primary Key(code)  
);
```

```
CREATE TABLE Compte( numero INTEGER(10) primary key,  
                      codeClient INTEGER(5) not Null,  
                      dateOuverture DATE,  
                      solde INTEGER,  
                      foreign key(codeClient ) references Client(code)  
);
```

Insertion des données: Insert into

- On peut insérer pour toutes les colonnes :

```
INSERT INTO Client(code, nom, prenom, age)  
VALUES (21001, 'Ali', 'Ahmed', 18);
```

```
INSERT INTO Client(code, nom, prenom, age)  
VALUES (21002, 'Ahmed', 'Moussa', 20);
```

```
INSERT INTO Client(code, nom, prenom, age)  
VALUES (21003, 'Fatma', 'Sidi', 18);
```

```
INSERT INTO Client  
VALUES (21004, 'Fati', 'Bah', 21);
```

The diagram illustrates the insertion of five rows into a 'Client' table. Four separate INSERT statements are shown on the left, each adding a new row to the table on the right. The table has columns: code, nom, prenom, and age. The rows are: (21001, Ali, Ahmed, 18), (21002, Ahmed, Moussa, 20), (21003, Fatma, Sidi, 18), (21004, Fati, Bah, 21), and (21005, Toto, NULL, NULL). A large brace on the left groups all four INSERT statements, and another brace on the right groups all five rows in the table.

code	nom	prenom	age
21001	Ali	Ahmed	18
21002	Ahmed	Moussa	20
21003	Fatma	Sidi	18
21004	Fati	Bah	21
21005	Toto	NULL	NULL

- On peut insérer pour certaines colonnes :

```
INSERT INTO Client(code, nom)  
VALUES (21005, 'Toto');
```

Modification des données : Update

Pour modifier l'age de Toto à 25 ans,
on utilise la commande UPDATE.

code	nom	prenom	age
21001	Ali	Ahmed	18
21002	Ahmed	Moussa	20
21003	Fatma	Sidi	18
21004	Fati	Bah	21
21005	Toto	NULL	NULL



Update client

Set age=25

Where code=21005;

Que fait cette commande ?

UPDATE client

SET age=age+1;

code	nom	prenom	age
21001	Ali	Ahmed	18
21002	Ahmed	Moussa	20
21003	Fatma	Sidi	18
21004	Fati	Bah	21
21005	Toto	NULL	25



Suppression des données/table: Delete - Drop

Pour supprimer une ou plusieurs lignes,
on utilise la commande DELETE.

Par exemple, supprimer le client numero 21005:

code	nom	prenom	age
21001	Ali	Ahmed	18
21002	Ahmed	Moussa	20
21003	Fatma	Sidi	18
21004	Fati	Bah	21
21005	Toto	NULL	25



**Delete from client
where code=21005;**

code	nom	prenom	age
21001	Ali	Ahmed	18
21002	Ahmed	Moussa	20
21003	Fatma	Sidi	18
21004	Fati	Bah	21

Quel sera le resultat cette commande ?

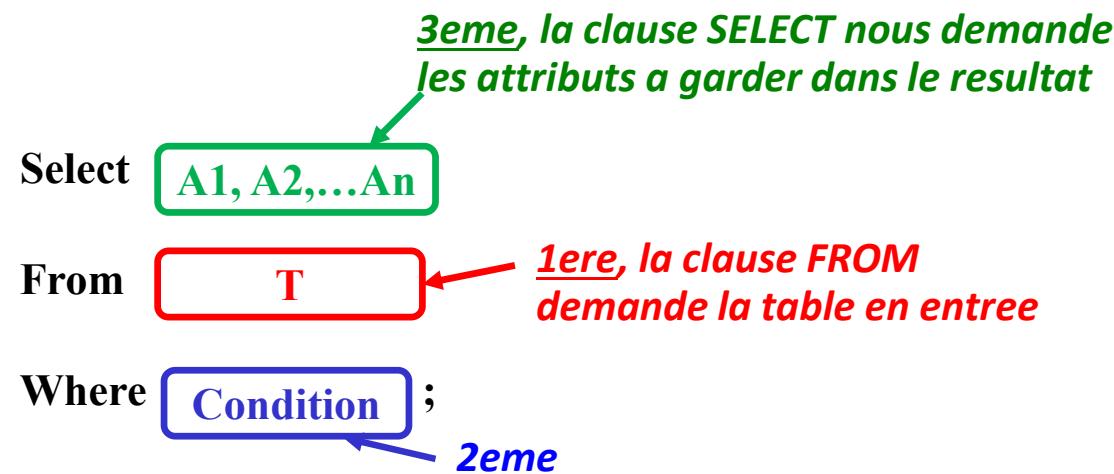
`DELETE FROM client;`

Pour supprimer une table (et donc les données), on utilise la commande DROP.

`DROP table client;`

Extraction de données: Select

A partir d'une seule table



Selection: Example

Compte

<u>numero</u>	code	DateOuverture	solde
102	21001	10/22/2019	500
102	21001	10/29/2009	200
104	21003	10/29/2020	1000
105	21002	11/2/2021	10000

Select * From Compte ;

resultat

<u>numero</u>	code	DateOuverture	solde
102	21001	10/22/2019	500
102	21001	10/29/2009	200
104	21003	10/29/2020	1000
105	21002	11/2/2021	10000

Example (suite)

Compte

<u>numero</u>	code	DateOuverture	solde
102	21001	10/22/2019	500
102	21001	10/29/2009	200
104	21003	10/29/2020	1000
105	21002	11/2/2021	10000

Select * From Compte

Where solde<1000;

output

solde< 1000?
oui!

<u>numero</u>	code	DateOuverture	solde
102	21001	10/22/2019	500

Example (suite)

compte

<u>numero</u>	code	DateOuverture	solde
102	21001	10/22/2019	500
103	21001	10/29/2009	200
104	21003	10/29/2020	1000
105	21002	11/2/2021	10000

solde < 1000 ?

oui!

Select * from compte
Where solde < 1000;

output

<u>numero</u>	code	DateOuverture	solde
102	21001	10/22/2019	500
103	21001	10/29/2009	200

Example (suite)

compte

<u>numero</u>	code	DateOuverture	solde
102	21001	10/22/2019	500
102	21001	10/29/2009	200
104	21003	10/29/2020	1000
105	21002	11/2/2021	10000



solde < 1000?

NON!

A ignorer

Select * from Account
Where solde < 1000;

output

<u>numero</u>	code	DateOuverture	solde
102	21001	10/22/2019	500
102	21001	10/29/2009	200

Example (suite)

compte

<u>numero</u>	code	DateOuverture	solde
102	21001	10/22/2019	500
103	21001	10/29/2009	200
104	21003	10/29/2020	1000
105	21002	11/2/2021	10000

solde < 1000?

NON!

A ignorer

Select * from Compte

Where solde < 1000;

output

<u>numero</u>	code	DateOuverture	solde
102	21001	10/22/2019	500
103	21001	10/29/2009	200

Example (suite)

compte

<u>numero</u>	code	DateOuverture	solde
102	1	10/22/09	500
103	2	10/29/09	200
104	3	10/29/09	1000
105	4	11/2/09	10000

Select num, solde
from Compte
Where solde<1000;

solde< 1000?
YES!

WHERE

SELECT

~~| <u>numero</u> | code | DateOuverture | solde |
|---------------|------|---------------|-------|
| 102 | 1 | 10/22/09 | 500 |
| 103 | 2 | 10/29/09 | 200 |~~

Resultat intermediaire

<u>numero</u>	solde
102	500
103	200

Resultat final

Example

compte

<u>numero</u>	code	solde	Type
101	Moussa	700	checking
102	Khaled	2000	checking
103	Azeiz	500	savings
104	Moussa	1200	checking
105	Sultan	8000	checking

```
SELECT *  
FROM compte  
WHERE Type = "checking" AND proprietaire="Moussa";
```

Que retourne cette requête?

Example

compte

<u>num</u>	proprietaire	solde	Type
101	Moussa	700	checking
102	Khaled	2000	checking
103	Azeiz	500	savings
104	Moussa	1200	checking
105	Sultan	8000	checking

```
SELECT *
FROM compte
WHERE Type = "checking" AND proprietaire="Moussa";
```

resultat:

<u>num</u>	proprietaire	solde	Type
101	Moussa	700	Checking
104	Moussa	1200	checking

Example

Compte

<u>numero</u>	proprietaire	solde	Type
101	Moussa	700	checking
102	Khaled	2000	checking
103	Azeiz	500	savings
104	Moussa	1200	checking
105	Sultan	8000	checking

```
SELECT proprietaire, Type  
FROM compte  
WHERE Type= "checking" AND solde<=400;
```

resultat

proprietaire	Type

Example

Compte

<u>num</u>	proprietaire	Solde	Type
101	Moussa	700	checking
102	Khaled	2000	checking
103	Azeiz	500	savings
104	Moussa	1200	checking
105	Sultan	8000	checking

```
SELECT proprietaire, Type  
FROM Compte  
WHERE proprietaire= "Moussa" OR solde<=600;
```

Resultat?

Example 5 cont.

Compte

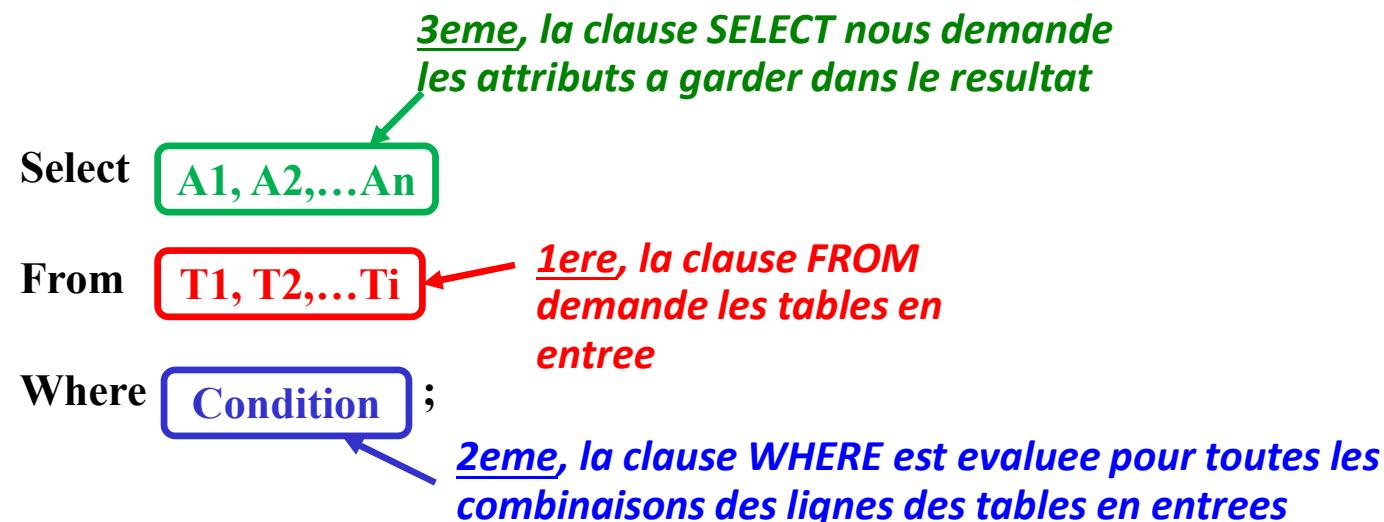
<u>num</u>	proprietaire	solde	Type
101	Moussa	700	checking
102	Khaled	2000	checking
103	Azeiz	500.00	savings
104	Moussa	1200	checking
105	Sultan	8000	checking

```
SELECT proprietaire, Type  
FROM Compte  
WHERE proprietaire= "Moussa" OR  
      Balance<=600;
```

proprietaire	Type
Moussa	checking
Azeiz	savings
Moussa	checking

Extraction de données: SELECT

A partir de plusieurs tables



Exemple

Depts(deptno, dname)
Emps(empno, ename, addr, deptno#)

Emps

<u>empno</u>	<u>ename</u>	<u>addr</u>	<u>deptno</u>
7499	Ali	...	30
7654	Moussa	...	30
7698	Salah	...	30
7839	Azeiz	...	10
7844	Fahd	...	30
7986	sultan	...	50

Depts

<u>deptno</u>	<u>dname</u>
30	SALES
10	ACCOUNTING
50	SUPPORT

6 relations

Q1. afficher les noms et addresses des employés:

SELECT ename, addr FROM Emps;

QUESTIONS-REPONSES

Q2. Lister les noms et addresses des employés qui travaillent dans le département numero 30:

```
SELECT ename, addr FROM Emps  
where deptno=30;
```

Q3. Calculer le nombre d'employés:

```
SELECT count(*) FROM Emps;
```

Q4. Calculer le nombre d'employés dans le département numéro 30:

```
SELECT count(*) FROM Emps  
Where deptno=30;
```

QUESTIONS-REPONSES

Q5. Afficher le nom du département dans lequel travail Ali:

```
SELECT dname FROM Emps, Depts  
WHERE ename = 'Ali' and Emps.depno=Depts.deptno;
```

Q6. Afficher les noms des employés du département de SALES:

```
SELECT ename FROM Emps, Depts  
WHERE dname = 'SALES' and  
Emps.depno=Depts.deptno;
```

Q7. Pour chaque employé, afficher son nom et le nom de son département

```
SELECT ename, dname FROM Emps, Depts  
Where Emps.depno=Depts.deptno;
```

ENAME	DNAME
-----	-----
Ali	SALES
Moussa	SALES
Salah	SALES
Azeiz	ACCOUNTING
Fahd	SALES
Sultan	SUPPORT

Les Groupes de Function

Les groupes de fonctions produisent des résultats basés sur un groupe/ensemble de lignes.

SQL> SELECT * FROM Paie;

**SELECT COUNT(*)
FROM Paie;**

ID	NAME	JOBCODE	STARTDATE	SALARY	BONUS
1111	Toto Titi	CI	15-JAN-97	45000	1000
2222	Jojo Dada	IN	25-SEP-92	40000	1500
3333	Soso Tata	AP	05-FEB-00	25000	500
4444	Tata Titi	CM	03-JUL-97	42000	2000
5555	Tutu Jojo	CI	30-OCT-92	50000	2000
6666	Baba Bubu	IN	18-AUG-94	48000	2000
7777	Toto Jojo	CI	05-NOV-99	45000	
8888	Papa Nono	IN	12-DEC-98	45000	2000

COUNT(*)

COUNT compte les lignes et ignore les valeurs NULL.

ID	NAME	JOBCODE	STARTDATE	SALARY	BONUS
1111	Toto Titi	CI	15-JAN-97	45000	1000
2222	Jojo Dada	IN	25-SEP-92	40000	1500
3333	Soso Tata	AP	05-FEB-00	25000	500
4444	Tata Titi	CM	03-JUL-97	42000	2000
5555	Tutu Jojo	CI	30-OCT-92	50000	2000
6666	Baba Bubu	IN	18-AUG-94	48000	2000
7777	Toto Jojo	CI	05-NOV-99	45000	
8888	Papa Nono	IN	12-DEC-98	45000	2000

```
SQL> SELECT COUNT(name)  
      FROM Paie;
```

COUNT (NAME)

8

COUNT(name) compte les lignes avec name non NULL.

```
SQL> SELECT COUNT(bonus)  
      FROM Paie;
```

COUNT(bonus) compte les lignes avec bonus non NULL.

.

COUNT (BONUS)

7

Group Function

```
SQL> SELECT ID, SALARY FROM Paie;
```



ID	SALARY
1111	45000
2222	40000
3333	25000
4444	42000
5555	50000
6666	48000
7777	45000
8888	45000

```
SQL> SELECT COUNT(salary)  
      FROM Paie;
```

COUNT (SALARY)

8

```
SQL> SELECT COUNT(DISTINCT salary)  
      FROM Paie;
```

COUNT (SALARY)

6

"Nombre de salaires différents"

Group Function

Dans cette SUM, la valeur NULL est ignorée. Le total est 11000.

```
SQL> SELECT SUM(bonus)  
      FROM Paie;
```

ID	BONUS
1111	1000
2222	1500
3333	500
4444	2000
5555	2000
6666	2000
7777	
8888	2000

```
SQL> SELECT ID, BONUS FROM Paie;
```

ID	BONUS
1111	1000
2222	1500
3333	500
4444	2000
5555	2000
6666	2000
7777	
8888	2000

11000

Group Function

```
SQL> SELECT SUM(bonus) , AVG(bonus)  
      FROM Paie;
```

SUM(BONUS)	AVG(BONUS)
-----	-----
11000	1571.4286

AVG(bonus)=SUM(bonus) / 7

```
SQL> SELECT ID , BONUS FROM Paie;
```

ID	BONUS
1111	1000
2222	1500
3333	500
4444	2000
5555	2000
6666	2000
7777	
8888	2000

Attention!

```
SQL> SELECT MIN(salary) , MAX(salary)  
      FROM Paie;
```

MIN (SALARY)	MAX (SALARY)
-----	-----
25000	50000

Cette expression affiche le plus petit et le plus grand salaires de la table Paie.

```
SQL> SELECT * FROM Paie;
```

ID	NAME	JOBCODE	STARTDATE	SALARY	BONUS
1111	Toto Titi	CI	15-JAN-97	45000	1000
2222	Jojo Dada	IN	25-SEP-92	40000	1500
3333	Soso Tata	AP	05-FEB-00	25000	500
4444	Tata Titi	CM	03-JUL-97	42000	2000
5555	Tutu Jojo	CI	30-OCT-92	50000	2000
6666	Baba Bubu	IN	18-AUG-94	48000	2000
7777	Toto Jojo	CI	05-NOV-99	45000	
8888	Papa Nono	IN	12-DEC-98	45000	2000

Group Function

```
SQL> SELECT MIN(bonus) , MAX(bonus)  
      FROM Paie;
```

MIN (BONUS)	MAX (BONUS)
500	2000

La valeur NULL est ignorée

```
SQL> SELECT * FROM Paie;
```

ID	NAME	JOBCODE	STARTDATE	SALARY	BONUS
1111	Toto Titi	CI	15-JAN-97	45000	1000
2222	Jojo Dada	IN	25-SEP-92	40000	1500
3333	Soso Tata	AP	05-FEB-00	25000	500
4444	Tata Titi	CM	03-JUL-97	42000	2000
5555	Tutu Jojo	CI	30-OCT-92	50000	2000
6666	Baba Bubu	IN	18-AUG-94	48000	2000
7777	Toto Jojo	CI	05-NOV-99	45000	
8888	Papa Nono	IN	12-DEC-98	45000	2000

Le Group by

```
SQL> SELECT * FROM Paie;
```

ID	NAME	JOBCODE	STARTDATE	SALARY	BONUS
1111	Toto Titi	CI	15-JAN-97	45000	1000
2222	Jojo Dada	IN	25-SEP-92	40000	1500
3333	Soso Tata	AP	05-FEB-00	25000	500
4444	Tata Titi	CM	03-JUL-97	42000	2000
5555	Tutu Jojo	CI	30-OCT-92	50000	2000
6666	Baba Bubu	IN	18-AUG-94	48000	2000
7777	Toto Jojo	CI	05-NOV-99	45000	3000
8888	Papa Nono	IN	12-DEC-98	45000	2000

Partitionnement: Group By

- Principe
 - partitionnement horizontal d'une relation, selon les valeurs d'un attribut ou groupe d'attributs qui est spécifié dans la clause GROUP BY
 - la relation est (logiquement) fragmentée en groupes de lignes, où tous les tuples de chaque groupe ont la même valeur pour l'attribut (ou le groupe d'attributs) de partitionnement
- Restrictions sur les groupes
 - application possible d'un critère de restriction sur les groupes obtenus
 - clause HAVING

Exemple: Afficher le nombre d'employés par code de travail (jobcode) ?

ID	NAME	JOBCODE	STARTDATE	SALARY	BONUS
1111	Toto Titi	CI	15-JAN-97	45000	1000
2222	Jojo Dada	IN	25-SEP-92	40000	1500
3333	Soso Tata	AP	05-FEB-00	25000	500
4444	Tata Titi	CM	03-JUL-97	42000	2000
5555	Tutu Jojo	CI	30-OCT-92	50000	2000
6666	Baba Bubu	IN	18-AUG-94	48000	2000
7777	Toto Jojo	CI	05-NOV-99	45000	3000
8888	Papa Nono	IN	12-DEC-98	45000	2000

JOBCODE	COUNT (ID)
-----	-----
AP	1
CI	3
CM	1
IN	3

Il faut utiliser une fonction qui permet de regrouper les lignes par **jobcode** et puis les compter !

```
SELECT colonnes
FROM tables
WHERE conditions
GROUP BY colonnes
HAVING condition
ORDER BY colonne(s);
```

Dans cet exemple, on affiche le nombre d'employés par code de travail (jobcode). Pour ce faire on a besoin de regrouper les employés par jobcode. Pour grouper par jobcode, on a besoin de selectionner le jobcode.

On a mis count(ID) pour compter le nombre d'employés

```
SQL> SELECT jobcode, count(ID)
      FROM Paie
      GROUP BY jobcode;
```

JOBCODE	COUNT (ID)
AP	1
CI	3
CM	1
IN	3

```
SQL> SELECT jobcode, count(*)
      FROM Paie
      GROUP BY jobcode;
```

JOBCODE	COUNT (*)
AP	1
CI	3
CM	1
IN	3

Group by

ID	NAME	JOBCODE	STARTDATE	SALARY	BONUS
1111	Toto Titi	CI	15-JAN-97	45000	1000
2222	Jojo Dada	IN	25-SEP-92	40000	1500
3333	Soso Tata	AP	05-FEB-00	25000	500
4444	Tata Titi	CM	03-JUL-97	42000	2000
5555	Tutu Jojo	CI	30-OCT-92	50000	2000
6666	Baba Bubu	IN	18-AUG-94	48000	2000
7777	Toto Jojo	CI	05-NOV-99	45000	
8888	Papa Nono	IN	12-DEC-98	45000	2000

```
SQL> SELECT jobcode, COUNT(name)
  FROM Paie
 WHERE salary <= 45000
 GROUP BY jobcode;
```

JOBCODE	COUNT (NAME)
-----	-----

AP	1
----	---

CI	2
----	---

CM	1
----	---

IN	2
----	---

Contrairement à l'exemple précédent, ici on considère seulement les employés dont les salaires sont ≤ 45000 . Donc les employés 5555 et 6666 sont exclus.

Group by

```
SQL> SELECT jobcode, COUNT(name)
   FROM Paie
 WHERE salary <= 45000
 GROUP BY jobcode
 ORDER BY jobcode desc;
```

Par ordre décroissant de jobcode.

JOBCODE	COUNT (NAME)
IN	2
CM	1
CI	2
AP	1

```
SQL> SELECT jobcode, COUNT(name)
   FROM Paie
 WHERE salary <= 45000
 GROUP BY jobcode
 ORDER BY COUNT(name);
```

Par ordre croissant de count(name)

JOBCODE	COUNT (NAME)
AP	1
CM	1
CI	2
IN	2

Group by

```
SQL> SELECT jobcode, COUNT(name)
  FROM Paie
 WHERE jobcode != 'IN'
 GROUP BY jobcode;
```

JOBCODE	COUNT (NAME)
-----	-----
AP	1
CI	3
CM	1

ID	NAME	JOBCODE
1111	Toto Titi	CI
2222	Jojo Dada	IN
3333	Soso Tata	AP
4444	Tata Titi	CM
5555	Tutu Jojo	CI
6666	Baba Bubu	IN
7777	Toto Jojo	CI
8888	Papa Nono	IN

on affiche le nombre d'employés par code de travail (jobcode) excepté de ceux du code 'IN'.

Group by

```
SQL> SELECT jobcode, MIN(salary), MAX(salary)  
      FROM Paie  
     GROUP BY jobcode;
```

JOBCODE	MIN(SALARY)	MAX(SALARY)
AP	25000	25000
CI	45000	50000
CM	42000	42000
IN	40000	48000

Affiche pour chaque code de travail (jobcode) le plus petit et le plus grand salaires.

JOBCODE	SALARY
CI	45000
IN	40000
AP	25000
CM	42000
CI	50000
IN	48000
CI	45000
IN	45000

```
SQL> SELECT jobcode, AVG(salary)  
      FROM Paie  
     GROUP BY jobcode;
```

JOBCODE	AVG(SALARY)
AP	25000
CI	46666.667
CM	42000
IN	44333.333

Donne le salaire moyen pour chaque jobcode.

"... uniquement si ce jobcode concerne plus d'un employé"

```
SQL> SELECT jobcode, AVG(salary)
   FROM Paie
  GROUP BY jobcode;
 HAVING COUNT(*)>1
```

JOBCODE	AVG(SALARY)
--	-----
CI	46666.667
IN	44333.333



JOBCODE	SALARY
CI	45000
IN	40000
AP	25000
CM	42000
CI	50000
IN	48000
CI	45000
IN	45000

```
SQL> SELECT jobcode, ROUND(AVG(salary),2)
   FROM Paie
  GROUP BY jobcode;
 HAVING COUNT(*)>1
```

JOBCODE	AVG(SALARY)
--	-----
CI	46666.66
IN	44333.33

arrondi à 2 décimales



FIN