

**Base de données 2**

**SupNum**

# **Jointures et Groupe de Fonctions**

*Moussa Demba*

# Le Group by (rappel)

```
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 !

## Group functions

```
SELECT colonne(s)
FROM tables
WHERE condition
GROUP BY colonne(s)
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 sélectionner 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 functions

| 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 functions

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

Par ordre decroissant de jobcode.

| JOB | COUNT |
|-----|-------|
| 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)

| JOB | COUNT |
|-----|-------|
| AP  | 1     |
| CM  | 1     |
| CI  | 2     |
| IN  | 2     |

## Group functions

```
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 functions

```
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     |

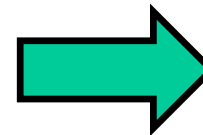


**arrondi à 2 décimales**

## Application: le fichier de spécialisation de supnum

- *Importez le fichier orientation.ods*
- *Afficher le nb d'étudiants par filiere*

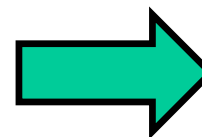
```
SELECT Orientation, count(*)  
FROM tabOrient  
group by Orientation  
order by count(*) DESC;
```



| Orientation | nb_etudiants |
|-------------|--------------|
| DSI         | 46           |
| RSS         | 23           |
| CNM         | 12           |

- *Afficher le nb d'étudiants par filiere qui ont plus de 20 étudiants*

```
SELECT Orientation, count(*)  
FROM tabOrient  
group by Orientation  
Having count(*)>20  
order by count(*) DESC;
```



| Orientation | nb_etudiants |
|-------------|--------------|
| DSI         | 46           |
| RSS         | 23           |

## Les jointures internes : NATURAL et INNER JOIN

On considère les 3 tables R1, R2 et R3:

| A     | B  |
|-------|----|
| -10.5 | 15 |
| -20.1 | 30 |

| B  | C    |
|----|------|
| 15 | toto |
| 40 | titi |

| A     | B  | C  |
|-------|----|----|
| 30.5  | 15 | 80 |
| 40.5  | 50 | 70 |
| -20.1 | 30 | 90 |

❖ `SELECT * FROM R1 NATURAL JOIN R2 ;`      OU BIEN      `SELECT * FROM R1 INNER JOIN R2 USING(B);`

*NATURAL JOIN est basée sur les noms des colonnes*

| b  | a      | c    |
|----|--------|------|
| 15 | -10.50 | toto |

*La colonne de jointure, ici B, n'est pas dupliquée*

❖ `SELECT * FROM R1 NATURAL JOIN R3 ;`      OU BIEN      `SELECT * FROM R1 INNER JOIN R3 USING(A, B);`

| a      | b  | c  |
|--------|----|----|
| -20.10 | 30 | 90 |

❖ `SELECT * FROM R1 INNER JOIN R3 USING(B);`

| b  | a      | a      | c  |
|----|--------|--------|----|
| 15 | -10.50 | 30.50  | 80 |
| 30 | -20.10 | -20.10 | 90 |

*Attention, la colonne A est dupliquée  
Car ce n'est pas une colonne de jointure*

## Les jointures internes : NATURAL et INNER JOIN (suite)

On considere les 3 tables

R1, R2 et R3:

R1

| A     | B  |
|-------|----|
| -10.5 | 15 |
| -20.1 | 30 |

R2

| B  | C    |
|----|------|
| 15 | toto |
| 40 | titi |

R3

| A     | B  | C  |
|-------|----|----|
| 30.5  | 15 | 80 |
| 40.5  | 50 | 70 |
| -20.1 | 30 | 90 |

❖ `SELECT * FROM R1 INNER JOIN R2 ON R1.B=R2.B;`

| a      | b  | b  | c    |
|--------|----|----|------|
| -10.50 | 15 | 15 | toto |



*La colonne de jointure, ici B, est dupliquée*

❖ `SELECT * FROM R1 INNER JOIN R3 ON R1.B=R3.B;`

| a      | b  | a      | b  | c  |
|--------|----|--------|----|----|
| -10.50 | 15 | 30.50  | 15 | 80 |
| -20.10 | 30 | -20.10 | 30 | 90 |



❖ Quel est le resultat de cette requete:

`SELECT * FROM R1 INNER JOIN R2 USING(B) INNER JOIN R3 USING(B);`

???

## Les jointures externes : Left et Right JOIN

On considere les 3 tables R1, R2 et R3:

*R1*

| A     | B  |
|-------|----|
| -10.5 | 15 |
| -20.1 | 30 |

*R2*

| B  | C    |
|----|------|
| 15 | toto |
| 40 | titi |

*R3*

| A     | B  | D  |
|-------|----|----|
| 30.5  | 15 | 80 |
| 40.5  | 50 | 70 |
| -20.1 | 30 | 90 |

❖ **SELECT A, R1.B, C FROM R1 LEFT JOIN R2 ON R1.B=R2.B;**

| a      | b  | b    | c    |
|--------|----|------|------|
| -10.50 | 15 | 15   | toto |
| -20.10 | 30 | NULL | NULL |

 R1

❖ **SELECT A, R1.B, C FROM R1 LEFT JOIN R2 USING(B);**

| b  | a      | c    |
|----|--------|------|
| 15 | -10.50 | toto |
| 30 | -20.10 | NULL |

❖ **SELECT A, R2.B, C FROM R1 RIGHT JOIN R2 USING(B);**

| b  | c    | a      |
|----|------|--------|
| 15 | toto | -10.50 |
| 40 | titi | NULL   |

 R2

## Les jointures externes : Left et Right JOIN

On considere les 3 tables R1, R2 et R3:

*R1*

| A     | B  |
|-------|----|
| -10.5 | 15 |
| -20.1 | 30 |

*R2*

| B  | C    |
|----|------|
| 15 | toto |
| 40 | titi |

*R3*

| A     | B  | D  |
|-------|----|----|
| 30.5  | 15 | 80 |
| 40.5  | 50 | 70 |
| -20.1 | 30 | 90 |

❖ **SELECT A,R2.B,C FROM R1 RIGHT JOIN R2 ON USING(B) ;**

| b  | c    | a      |
|----|------|--------|
| 15 | toto | -10.50 |
| 40 | titi | NULL   |

❖ **SELECT \* FROM R1 RIGHT JOIN R2 USING(B) WHERE A IS NOT NULL ;**

| b  | c    | a      |
|----|------|--------|
| 15 | toto | -10.50 |

## Important

**POUR EXECUTER UN FICHER .SQL DEPUIS L'INVITE DE COMMANDE DOS:**

On suppose qu'on a un fichier TP.SQL qui contient les commandes SQL:

d:/id2/tp2.sql

- **Oracle:** sql> @d:/id2/tp2.sql;
- **MySQL:** mysql> source d:/id2/tp2.sql;

La BD dans tp2.sql sera créée dans la BD locale.

# Les sous requetes: les opérateurs ANY, SOME et ALL

ANY : au moins 1 ...

SOME : au moins 1 ...

ALL : tout ...

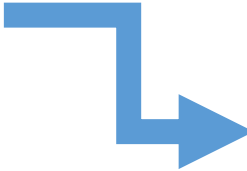
PAIE

| 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  |

```
SQL> SELECT NAME
      FROM PAIE
      WHERE JOBCODE= ANY (SELECT JOBCODE FROM
                          PAIE WHERE JOBCODE IN ('CI', 'IN')
                          );
```

## Remarques :

- l'opérateur ANY est équivalent à l'opérateur SOME.
- X > ANY (Ensemble)  
retourne vrai si X est > que n'importe quelle valeur de l'ensemble.



| name |      |
|------|------|
| Toto | Titi |
| Tutu | Jojo |
| Toto | Jojo |
| Jojo | Dada |
| Baba | Bubu |
| Papa | Nono |

# Les sous requetes: les opérateurs ANY, SOME et ALL

ANY : au moins 1 ...

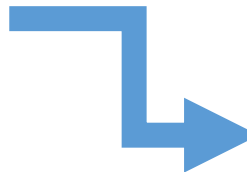
SOME : au moins 1 ...

ALL : tout ...

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  |

```
SQL> SELECT NAME,SALARY
FROM PAIE
WHERE SALARY > ANY (SELECT SALARY FROM
                    PAIE WHERE JOBCODE LIKE 'IN';
                    );
```



| name      | salary |
|-----------|--------|
| Toto Titi | 45000  |
| Tata Titi | 42000  |
| Tutu Jojo | 50000  |
| Baba Bubu | 48000  |
| Toto Jojo | 45000  |
| Papa Nono | 45000  |

## Les opérateurs ANY, SOME et ALL

Sélectionnez les paiements dont les salaires sont différents de 45.000, 40.000 et 48.000

```
SQL> SELECT *  
      FROM PAIE  
      WHERE SALARY != ALL (select salary where jobcode IN('IN', 'AP'));
```

!=ALL est equivalent a NOT IN

=ANY est équivalent à IN

*X =ANY ( A , B , C ) est la même chose que X IN ( A , B , C )*

## Les opérateurs ANY, SOME et ALL

**ALL:** Le résultat est vrai si toutes les valeurs répondent à la comparaison

```
SELECT NUMAD FROM RESULTATS
WHERE NOTE >ALL
      (SELECT NOTE FROM RESULTATS WHERE CODE_COURS ='KEG') ;
```

*On cherche un numero pour lequel sa note est superieure a toutes les notes du cours KEG*

Donne le resultat

| NUMAD |
|-------|
| 102   |

Resultats

| CODE_COURS | NUMAD | NOTE |
|------------|-------|------|
| KED        | 100   | 78   |
| KED        | 101   | 65   |
| KEG        | 100   | 88   |
| KEG        | 101   | 78   |
| KED        | 109   | 80   |
| KED        | 107   | 78   |
| KEG        | 107   | 55   |
| KED        | 102   | 90   |

```
SELECT NUMAD FROM RESULTATS
WHERE NOTE >ANY
      (SELECT NOTE FROM RESULTATS WHERE CODE_COURS ='KEG') ;
```

*On cherche un numero pour lequel sa note est superieure a au moins une des notes du cours KEG*

Donne le resultat

| NUMAD |
|-------|
| 102   |
| 100   |
| 109   |
| 100   |
| 101   |
| 107   |
| 101   |

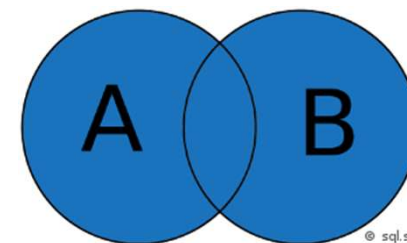
## UNION

C'est une commande qui permet de concaténer les résultats de 2 requêtes ou plus. Pour l'utiliser il est *nécessaire* que chacune des requêtes à concaténer retournes le même nombre de colonnes, avec les mêmes types de données et dans le même ordre.

**A savoir :** par défaut, les enregistrements exactement identiques ne seront pas répétés dans les résultats. Pour effectuer une union dans laquelle même les lignes dupliquées sont affichées il faut plutôt utiliser la commande UNION ALL.

La syntaxe pour unir les résultats de 2 tableaux sans afficher les doublons est la suivante:

```
SELECT * FROM A ...  
UNION  
SELECT * FROM B ...
```



```
mysql> SELECT * FROM PILOTE;
```

| PLNUM | PLNOM       | PLPRENOM   | VILLE    | SALAIRE |
|-------|-------------|------------|----------|---------|
| 1     | MIRANDA     | SERGE      | PARIS    | 26000   |
| 2     | LETHANH     | NAHN       | TOULOUSE | 21000   |
| 3     | TALADOIRE   | GILLES     | NICE     | 18000   |
| 4     | CHARBONNIER | ANNETTE    | PARIS    | 17000   |
| 5     | REY         | CHRISTOPHE | TOULOUSE | 19000   |
| 6     | CHARBONNIER | FABIEN     | PARIS    | 18000   |
| 7     | PENAUD      | PIERRE     | NICE     | 17000   |
| 8     | FOUILHOX    | PIERRE     | LYON     | 15000   |
| 9     | GANNAT      | CHRISTOPHE | NULL     | 18000   |
| 10    | GADAIX      | SONIA      | PARIS    | 20000   |

10 rows in set (0.00 sec)

```
mysql> SELECT * FROM avion;
```

| AUNUM | AUNOM    | CAPACITE | LOCALISATION |
|-------|----------|----------|--------------|
| 1     | A300     | 300      | NICE         |
| 2     | A310     | 300      | NICE         |
| 3     | B707     | 250      | PARIS        |
| 4     | A300     | 280      | LYON         |
| 5     | CONCORDE | 160      | NICE         |
| 6     | B747     | 460      | PARIS        |
| 7     | B707     | 250      | PARIS        |
| 8     | A310     | 300      | TOULOUSE     |
| 9     | MERCURE  | 180      | LYON         |
| 10    | CONCORDE | 160      | PARIS        |

```
mysql> SELECT * FROM UOL;
```

| UOLNUM | PLNUM | AUNUM | VILLEDEP | VILLEARR | HEUREDEP | HEUREARR |
|--------|-------|-------|----------|----------|----------|----------|
| 100    | 1     | 1     | NICE     | TOULOUSE | 1100     | 1230     |
| 101    | 1     | 8     | PARIS    | TOULOUSE | 1700     | 1830     |
| 102    | 2     | 1     | TOULOUSE | LYON     | 1400     | 1600     |
| 103    | 5     | 3     | TOULOUSE | LYON     | 1800     | 2000     |
| 104    | 9     | 1     | PARIS    | NICE     | 645      | 815      |
| 105    | 10    | 2     | LYON     | NICE     | 1100     | 1200     |
| 106    | 1     | 4     | PARIS    | LYON     | 800      | 900      |
| 107    | 8     | 4     | NICE     | PARIS    | 715      | 845      |
| 108    | 1     | 8     | NANTES   | LYON     | 900      | 1530     |
| 109    | 8     | 2     | NICE     | PARIS    | 1215     | 1345     |
| 110    | 9     | 2     | PARIS    | LYON     | 1500     | 1600     |
| 111    | 1     | 2     | LYON     | NANTES   | 1630     | 2000     |
| 112    | 4     | 5     | NICE     | LENS     | 1100     | 1400     |
| 113    | 3     | 5     | LENS     | PARIS    | 1500     | 1600     |
| 114    | 8     | 9     | PARIS    | TOULOUSE | 1700     | 1800     |
| 115    | 7     | 5     | PARIS    | TOULOUSE | 1800     | 1900     |

16 rows in set (0.00 sec)

```
mysql> SELECT * FROM PILOTE;
```

| PLNUM | PLNOM       | PLPRENOM   | VILLE    | SALAIRE |
|-------|-------------|------------|----------|---------|
| 1     | MIRANDA     | SERGE      | PARIS    | 26000   |
| 2     | LETHANH     | NAHN       | TOULOUSE | 21000   |
| 3     | TALADOIRE   | GILLES     | NICE     | 18000   |
| 4     | CHARBONNIER | ANNETTE    | PARIS    | 17000   |
| 5     | REY         | CHRISTOPHE | TOULOUSE | 19000   |
| 6     | CHARBONNIER | FABIEN     | PARIS    | 18000   |
| 7     | PENAUD      | PIERRE     | NICE     | 17000   |
| 8     | FOUILHOUX   | PIERRE     | LYON     | 15000   |
| 9     | GANNAT      | CHRISTOPHE | NULL     | 18000   |
| 10    | GADAIX      | SONIA      | PARIS    | 20000   |

Afficher les pilotes qui sont de paris ou nice.

```
mysql> SELECT * FROM PILOTE WHERE VILLE='PARIS' UNION SELECT * FROM PILOTE WHERE VILLE='NICE';
```

### Attention:

Pour utiliser la clause UNION, les expressions SELECT doivent avoir le meme nombre de colonnes

**Sans duplications des enregistrements**

UNION ALL → avec duplications des enregistrements, s'il en existe

## Le prédicat EXISTS

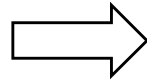
Il permet de tester si le résultat (tuples) d'une sous-requête est vide ou non

La clause EXISTS est suivie d'une sous-requête entre parenthèses, et *prend la valeur **vrai** s'il existe au moins une ligne satisfaisant les conditions de la sous-requête.*

- ❖ Pour tester si une valeur se trouve dans le résultat d'une requête, on utilise IN.
- ❖ Soit B une liste de valeurs renvoyées par une requête. Pour tester si une condition (par exemple  $A < B$ ) est vraie pour:
  - toutes les valeurs de B, on utilise ALL,
  - au moins une valeur de B, on utilise ANY.
- ❖ EXISTS vérifie si la sous-requête contient au moins une ligne.

## Le prédicat EXISTS

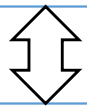
```
Select R1.*
From R1, R2
Where R1.b=R2.b;
```



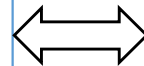
| a      | b  |
|--------|----|
| -10.50 | 15 |

| A     | B  |
|-------|----|
| -10.5 | 15 |
| -20.1 | 30 |

| B  | C    |
|----|------|
| 15 | toto |
| 40 | titi |



```
Select *
From R1
Where R1.b IN (Select R2.b From R2);
```



```
Select *
From R1
Where EXISTS (Select * From R2 Where R1.b=R2.b);
```

Soit  $E$  le resultat de la sous-requete:

- ✓  $b \text{ IN } E$  : retourne *true* si  $b \in E$  et *false* sinon
- ✓ **EXISTS**  $E$ : retourne *true* si  $E$  n'est pas vide et *false* sinon

- ✓  $b \text{ NOT IN } E$  : retourne *true* si  $b \notin E$  et *false* sinon
- ✓ **NOT EXISTS**  $E$ : retourne *true* si  $E$  est vide et *false* sinon

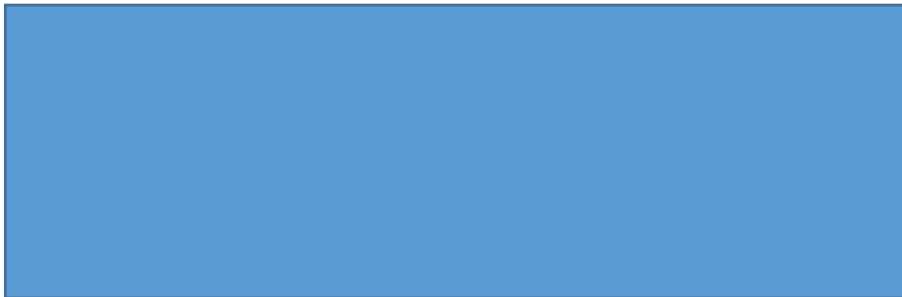
1. Liste des pilotes pilotant le meme avion que le pilote numero 4.



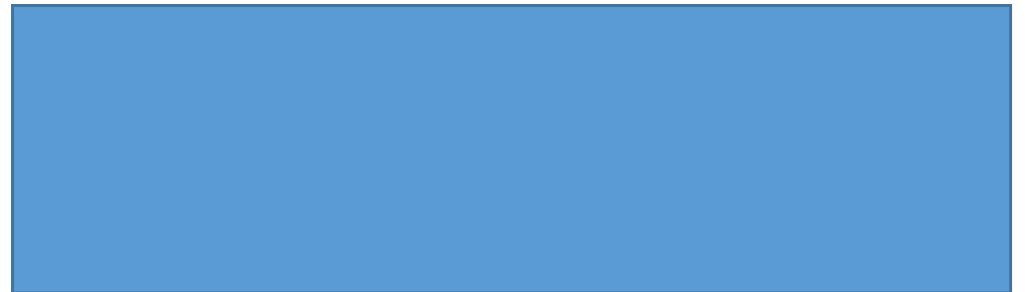
2. Liste des pilotes GAGNANT plus que tous les pilotes nicois.



3. Liste des pilotes ayant le MEME SALAIRE que le pilote 'PENAULD'.



4. Liste des pilotes ayant la MEME VILLE et le MEME SALAIRE que le pilote 'PENAULD'.



5. Liste des avions qui n'ont jamais volé.



6. Afficher les 3 pilotes les mieux payés par ordre décroissant des salaires.



7. Le numero et le nom de l'avion qui a volé le plus.

|  |
|--|
|  |
|--|

8. Créer une autre table pilote2 qui a la **même structure** que la table pilote.

|  |
|--|
|  |
|--|

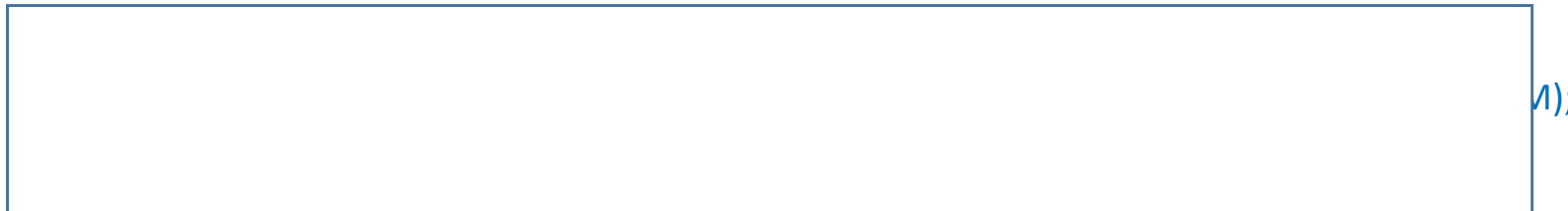
9. Insérer dans la table pilote2 tous les pilotes qui n'ont jamais volé, en utilisant une sous-requête;

|  |
|--|
|  |
|--|

10. Créer une autre table pilote3 qui contient les **memes données** que la table pilote, en utilisant une sous-requête;

A large, empty rectangular box with a thin blue border, intended for writing a SQL query to create a table named 'pilote3' using a subquery.

11. Supprimer tous les pilotes de la table pilote3 qui sont dans la table pilote2, en utilisant une sous-requête;

A large, empty rectangular box with a thin blue border, intended for writing a SQL query to delete records from 'pilote3' based on a subquery. The text '11);' is partially visible to the right of the box.

12. Que fait cette requete ?

```
UPDATE PILOTE3 SET salaire=(SELECT MAX(salaire) FROM PILOTE) WHERE plnum=9;
```



12. Que fait cette requete ?

```
DELETE FROM PILOTE3
```

```
WHERE not exists (SELECT 1 FROM PILOTE3 WHERE PILOTE3.plnum=VOL.plnum);
```



**FIN**