# Systèmes de gestion de bases de données
# Exercices sur une seule table (série 1)

Cours de Jacques THOORENS

## 1 Exercices de laboratoire (solutions)

1. Donner la liste des employés en citant Nom et Prénom (*Firstname*).

```
SELECT Last_name, First_name
FROM Employees;
```

2. Reprendre cette liste et la trier intelligemment.

```
SELECT Last_name, First_name
FROM Employees
ORDER BY Last_name, First_name;
SELECT Last_name, First_name
FROM Employees
ORDER BY 1,2;
SELECT Last_name as Nom, First_name as Prenom
FROM Employees
-- impossible sur SQLServer ou mySQL
ORDER BY Last_name, Prenom;
```

3. Donner la liste des employés, classés cette fois par départements.

```
SELECT DEPARTMENT_id, Last_name as Nom, First_name as Prenom
FROM Employees
ORDER BY Department_id,2,3;
```

4. Démontrer à l'aide de requêtes qu'il y a des homonymes dans les noms et dans les prénoms.

```
SELECT ALL Last_Name
FROM Employees;
SELECT DISTINCT Last_Name
FROM Employees;
```

   Il est possible de donner une solution automatique à ce problème [1].

5. Donnez la liste des employés par ordre d'ancienneté (du plus ancien au dernier engagé).

1. On utilise des fonctions statistiques :

```
-- Solution reelle (voir chapitre 6)
SELECT Count(all last_name),count(distinct last_name)
FROM Employees;
```

```
SELECT Hire_date, Last_name, First_name
FROM Employees
ORDER BY 1 asc, 2,3;
```

6. Donnez la liste des programmeurs (IT_PROG).

```
SELECT job_id,Last_name, First_name
FROM Employees
WHERE lower(job_id)='it_prog';
```

La fonction `lower()` qui transforme en minuscules nous dispense de savoir comment le code de la profession a été écrit précisément.

7. Donnez le prénom de l'employé nommé Gates.

```
SELECT Last_name, First_name
FROM Employees
WHERE upper(Last_name)='GATES';
```

Même précaution pour l'écriture du nom (cette fois, la fonction `upper()` transforme en majuscules.

8. Donnez la liste des employés dont le nom commence par B.

```
SELECT Last_name
FROM Employees
WHERE Last_name like 'B%';
```

9. Donnez la liste des emails
   - qui finissent par t,

```
SELECT email
FROM Employees
WHERE lower(email) like '%t';
```

   - qui contiennent t,

```
SELECT email
FROM Employees
WHERE lower(email) like '%t%';
```

   - qui contiennent au moins 2 t.

```
SELECT email
FROM Employees
WHERE lower(email) like '%t%t%';
```

10. Donnez la liste des employés
    - engagés en 1995,

```
SELECT Hire_date, Last_name, First_name
FROM Employees
WHERE hire_date like '__/__/95';
```

La fonction `to_char()` permet d'éviter tout problème en rapport avec la localisation.

```
SELECT Hire_date, Last_name, First_name
FROM Employees
WHERE to_char(hire_date,'DD/MM/YYYY') like '__/__/1995';
SELECT Hire_date, Last_name, First_name
FROM Employees
WHERE to_char(hire_date,'YYYY') = '1995';
```

- engagés après 1997

```
SELECT Hire_date, Last_name, First_name
FROM Employees
WHERE to_char(hire_date,'YYYYMMDD') > '19971231';
SELECT Hire_date, Last_name, First_name
FROM Employees
WHERE to_char(hire_date,'YYYY') > '1997';
```

- engagés entre 1996 et 1998

```
SELECT Hire_date, Last_name, First_name
FROM Employees
WHERE to_char(hire_date,'YYYY/MM/DD') BETWEEN '19960101' AND '19981231';
SELECT Hire_date, Last_name, First_name
FROM Employees
WHERE to_char(hire_date,'YYYY') BETWEEN '1996' AND '1998';
```

- engagés au mois d'avril

```
SELECT Hire_date, Last_name, First_name
FROM Employees
WHERE to_char(hire_date,'YYYY/MM/DD') LIKE '____/04/__';
SELECT Hire_date, Last_name, First_name
FROM Employees
WHERE to_char(hire_date,'MM') = '04';
```

11. A l'aide **de plusieurs requêtes**, déterminer les employés qui sont sous les ordres de Mr Raphaely.

```
SELECT employee_id
FROM Employees
WHERE upper(Last_name) = 'RAPHAELY';
-- C'est 114
SELECT Last_name, manager_id
FROM Employees
WHERE manager_id = 114;
```

Pour éviter le recours au Postit, on pourra faire un requête imbriquée [2].

12. Vérifier que tous les employés ont un département.

2. Voir chapitre 7 :

```
SELECT Last_name, manager_id
FROM Employees
WHERE manager_id = (SELECT employee_id
FROM Employees
WHERE upper(Last_name) = 'RAPHAELY');
```

```
SELECT Last_Name
FROM EMployees
WHERE Department_id is null;
```

Si la requête n'affiche rien, c'est que tous les employés ont un département.

# 2 Exercices à rendre sur feuille (Solutions)

1. Affichez le nom et le prénom de tous les étudiants (sans tri).

```
SELECT NOM, PRENOM FROM Etudiants;
```

2. Affichez le prénom de tous les étudiants (sans tri).

```
SELECT ALL PRENOM FROM Etudiants;
-- ou
SELECT PRENOM FROM Etudiants;
```

3. Affichez la liste de toutes les sections de l'école.

```
SELECT DISTINCT Section FROM Etudiants;
```

4. Afficher la liste des prénoms rencontrés dans l'école.

```
SELECT DISTINCT PRENOM FROM Etudiants;
```

5. Affichez le nom, les différents points obtenus et le total général de tous les étudiants du cours « Allemand 1 », triés par nom.

```
SELECT Nom, Interro1, Interro2, Examen,
(Interro1 +Interro2+Examen) AS TOTAL
FROM Etudiants
WHERE Section = 'Allemand␣1'
ORDER BY Nom;
```

6. Affichez le nom, les différents points obtenus et la moyenne de tous les étudiants du cours « Espagnol 3 ». Vous placerez les meilleurs moyennes en tête. En cas d'*ex aequo*, vous trierez sur le nom.

```
SELECT Nom, Interro1, Interro2, Examen,
(Interro1 +Interro2+Examen)/3 AS Moyenne
FROM Etudiants
WHERE Section = 'Espagnol␣3'
ORDER BY 5 DESC, Nom;
```

7. Faites la même chose pour tous les étudiants qui suivent un cours d'anglais, quelle que soit l'année.

```
SELECT Nom, Interro1, Interro2, Examen,
(Interro1 +Interro2+Examen)/3 AS Moyenne
FROM Etudiants
WHERE Section LIKE 'Anglais%'
ORDER BY 5 DESC, Nom;
```

8. Affichez le nom et la date de naissance de tous les étudiants nés après 1975, triés par âges décroissants.

```
SELECT Nom, DateNaiss
FROM Etudiants
WHERE DateNaiss > date '1975-12-31'
ORDER BY 2;
```

J'ai employé ici une méthode prudente pour noter la date. Utiliser le mot date puis la chaîne en mode japonnais.

9. Affichez les élèves d'allemand 2 qui ont réussi la première interrogation (au moins 5/10). Triez par mérite décroissant et par noms en cas d'*ex-aequo*.

```
SELECT Nom, Interro1
FROM Etudiants
WHERE Interro1 > 4 AND Section = 'Allemand␣2
ORDER␣BY␣2␣DESC,␣1;
```

10. Modifiez l'écriture de la requête pour ne pas utiliser de comparateur '>'.

```
-- Ceci répond à la question de manière plaisante
SELECT Nom, Interro1
FROM Etudiants
WHERE not Interro1 < 5 AND Section = 'Allemand␣2'
ORDER BY 2 DESC, 1;
-- Une solution plus originale
SELECT Nom, Interro1
FROM Etudiants
WHERE Interro1 BETWEEN 5 AND 10 AND Section = 'Allemand␣2'
ORDER BY 2 DESC, 1;
```

11. Écrivez une troisième version de cette requête, sachant que les notes sont toujours des entiers.

```
SELECT Nom, Interro1
FROM Etudiants
WHERE Interro1 IN (5,6,7,8,9,10) AND Section = 'Allemand␣2'
ORDER BY 2 DESC, 1;
```

12. Affichez tous les étudiants dont le nom commence par A.

```
SELECT Nom
FROM Etudiants
WHERE Nom LIKE 'A%';
```

13. Affichez les étudiants dont le nom comporte deux E au minimum.

```
SELECT Nom
FROM Etudiants
WHERE Nom LIKE '%E%E%';
```

14. Affichez tous les étudiants dont le nom commence par A, B, C ou D. Les noms ne sont pas des ensembles de lettres tirées au hasard.

```
SELECT Nom
FROM Etudiants
WHERE Nom < 'E'
ORDER BY 1;
```

15. Listez par classes, les étudiants qui ont entre 3 et 5 à l'examen, en commençant par les moins mauvais résultats et en triant les *ex aequo*.

```
SELECT Section, Nom, Examen
FROM Etudiants
WHERE Examen BETWEEN 3 AND 5
ORDER BY 1,3 DESC, 2;
```

16. Affichez la liste alphabétique de tous les étudiants de 1re année (sachant que ces classes portent des noms comportant un chiffre 1). Triez par classes et par noms.

```
SELECT Section, Nom
FROM Etudiants
WHERE Section LIKE '%1%' ORDER BY 1,2;
```

17. Affichez la liste des étudiants qui ont un échec à la deuxième interro . Même présentation.

```
SELECT Section, Nom, Examen
FROM Etudiants
Where Examen < 5
ORDER BY 1,2;
```

18. Affichez la liste des étudiants qui ont un échec à l'examen et à aucune des interrogations.

```
SELECT Section, Nom, Interro1, Interro2, Examen
FROM Etudiants
Where Examen < 5 AND Interro1 >= 5 AND Interro2 >= 5
ORDER BY 1,2;
```

19. Affichez la liste des étudiants qui suivent un cours n'ayant pas de rapport avec les langues (comptabilité et secrétariat) et n'ont pas d'échec au total.

```
SELECT DISTINCT Nom, Prenom, Section, Interro1+Interro2 + Examen AS TOTAL
FROM Etudiants
WHERE (SECTION LIKE 'Comptabilité%' OR Section LIKE 'Secrétariat%')
AND Interro1+Interro2 + Examen >= 15
ORDER BY 3,1,2;
```

20. Affichez la liste des étudiants qui ont plus à l'examen qu'à la moyenne des interros.

```
SELECT Section, NOM, Examen, (Interro1+ Interro2)/2 AS Moyenne
FROM Etudiants
WHERE EXAMEN > (INTERRO1 + Interro2)/2
ORDER BY 1, 3 DESC, 2;
```

21. Affichez la liste des étudiants qui ont plus de 23 au total des points.

```
SELECT Nom ,Interro1+Interro2+Examen
FROM  Etudiants
WHERE Interro1+Interro2+Examen > 23
ORDER BY 1;
```

22. Affichez la liste des étudiants qui sont nés en janvier (essayez de donner une solution indépendante des réglages linguistiques).

```
SELECT Section, Nom, DateNaiss
FROM Etudiants
WHERE DateNaissance LIKE '%/01/%'
ORDER BY 1,2;
```

Cette manière d'écrire la requête peut poser problème en fonction des réglages linguistiques. Voici une solution plus fiable :

```
SELECT Nom, DateNaiss
FROM Etudiants
WHERE TO_CHAR(DateNaiss, 'mm')=1
```