Partie VI : Jointures

Exercice 2.6.1 – **Donner pour chaque cours le nom du professeur responsable ainsi que la section dont le professeur fait partie**

```
SELECT course_name, section_name, professor_name
FROM professor P, course C, section S
WHERE S.section_id = P.section_id
AND P.professor_id = C.professor_id
```

```
SELECT course_name, section_name, professor_name
FROM professor P JOIN course C
     ON P.professor_id = C.professor_id
     JOIN section S ON S.section_id = P.section_id
```

Exercice 2.6.2 – **Donner pour chaque section, l'id, le nom et le nom de son délégué. Classer les sections dans l'ordre inverse des id de section. Un délégué est un étudiant de la table « STUDENT »**

```
SELECT S.section_id, S.section_name, St.last_name
FROM section S, student St
WHERE S.delegate_id = St.student_id
ORDER BY S.section_id DESC
```

```
SELECT S.section_id, S.section_name, St.last_name
FROM section S JOIN student St
     ON S.delegate_id = St.student_id
ORDER BY S.section_id DESC
```

Exercice 2.6.3 – **Donner pour chaque section, le nom des professeurs qui en sont membre**

```
SELECT S.section_id, S.section_name, P.professor_name
FROM section S LEFT JOIN professor P
     ON S.section_id = P.section_id
ORDER BY S.section_id DESC
```

Exercice 2.6.4 – **Même objectif que la question 3 mais seuls les sections comportant au moins un professeur doivent être reprises**

```
SELECT S.section_id, S.section_name, P.professor_name
FROM section S JOIN professor P
     ON S.section_id = P.section_id
ORDER BY S.section_id DESC
```

Exercice 2.6.5 – **Donner à chaque étudiant ayant obtenu un résultat annuel supérieur ou égal à 12 son grade en fonction de son résultat annuel et sur base de la table grade. La liste doit être classée dans l'ordre alphabétique des grades attribués**

```
SELECT S.last_name, S.year_result, G.Grade
FROM Grade AS G, student AS S
WHERE (S.year_result BETWEEN G.Lower_bound AND G.Upper_bound)
      AND (S.year_result >= 12)
ORDER BY G.Grade
```

Exercice 2.6.6 – **Donner la liste des professeurs et la section à laquelle ils se rapportent ainsi que le(s) cour(s) (nom du cours et crédits) dont le professeur est responsable. La liste est triée par ordre décroissant des crédits attribués à un cours**

```
SELECT P.professor_name, section_name, C.course_name, C.course_ects
FROM professor AS P LEFT JOIN section AS S
     ON P.section_id = S.section_id,
     professor AS P1 LEFT JOIN course AS C
     ON P1.professor_id = C.professor_id
WHERE P1.professor_id = P.professor_id
ORDER BY C.course_ects DESC
```

```
SELECT P.professor_name, section_name, C.course_name, C.course_ects
FROM section AS S RIGHT JOIN professor AS P
     ON P.section_id = S.section_id
     LEFT JOIN course AS C
     ON P.professor_id = C.professor_id
ORDER BY C.course_ects DESC
```

Exercice 2.6.7 – **Donner pour chaque professeur son id et le total des crédits ECTS (« ECTS_TOT ») qui lui sont attribués. La liste proposée est triée par ordre décroissant de la somme des crédits alloués**

```
SELECT P.professor_id, sum(C.course_ects) AS ECTS_TOT
FROM professor AS P LEFT JOIN course AS C
     ON p.professor_id=c.professor_id
GROUP BY P.professor_id
ORDER BY sum(C.course_ects) DESC
```

Exercice 2.6.8 – **Donner la liste (nom et prénom) de l'ensemble des professeurs et des étudiants dont le nom est composé de plus de 8 lettres. Ajouter une colonne pour préciser la catégorie (S pour « STUDENT », P pour « PROFESSOR ») à laquelle appartient l'individu**

```sql
SELECT first_name, last_name, 'S' as "Catégorie"
FROM student
WHERE LEN(last_name) > 8

UNION

SELECT professor_surname, professor_name, 'P'
FROM professor
WHERE LEN(professor_name) > 8
```

Exercice 2.6.9 – **Donner l'id de chacune des sections qui n'ont pas de professeur attitré**

```sql
SELECT section_id FROM section

EXCEPT

SELECT section_id FROM professor
```

Partie VII : Requêtes imbriquées

Exercice 2.7.1 – **Donner la liste des étudiants (nom et prénom) qui font partie de la même section que mademoiselle « Roberts ». La liste doit être classée par ordre alphabétique sur le nom et mademoiselle « Roberts » ne doit pas apparaitre dans la liste**

```
SELECT last_name, first_name, section_id
FROM student
WHERE section_id = ( SELECT section_id
                     FROM student
                     WHERE last_name='Roberts' )
      AND last_name <> 'Roberts'
ORDER BY last_name
```

Exercice 2.7.2 – **Donner la liste des étudiants (nom, prénom et résultat) de l'ensemble des étudiants ayant obtenu un résultat annuel supérieur au double du résultat moyen pour l'ensemble des étudiants**

```
SELECT last_name, first_name, year_result
FROM student
WHERE year_result > 2 * ( SELECT AVG(year_result)
                          FROM student )
```

Exercice 2.7.3 – **Donner la liste de toutes les sections qui n'ont pas de professeur**

```
SELECT section_id, section_name
FROM section
WHERE section_id NOT IN ( SELECT DISTINCT section_id
                          FROM professor )
```

Exercice 2.7.4 – **Donner la liste des étudiants qui ont comme mois de naissance le mois correspondant à la date d'engagement du professeur « Giot ». Classer les étudiants par ordre de résultat annuel décroissant**

```
SELECT last_name, first_name,
       CONVERT(VARCHAR,birth_date,101) AS "Date de Naissance",
       year_result
FROM student
WHERE MONTH(birth_date) = ( SELECT DATEPART(MONTH,professor_hire_date)
                            FROM professor
                            WHERE professor_name = 'Giot' )
ORDER BY year_result DESC
```

Exercice 2.7.5 – **Donner la liste des étudiants qui ont obtenu le grade « TB » pour leur résultat annuel**

```
SELECT last_name, first_name, year_result
FROM student
WHERE year_result BETWEEN ( SELECT Lower_bound
                            FROM Grade
                            WHERE grade like 'TB' )
                  AND ( SELECT Upper_bound
                        FROM Grade
                        WHERE grade like 'TB' )
```

Exercice 2.7.6 – **Donner la liste des étudiants qui appartiennent à la section pour laquelle Mademoiselle « Marceau » est déléguée**

```sql
SELECT last_name, first_name, section_id
FROM student
WHERE section_id = ( SELECT section_id
                     FROM section
                     WHERE delegate_id = ( SELECT student_id
                                           FROM student
                                           WHERE last_name like 'Marceau' ) )
```

Exercice 2.7.7 – **Donner la liste des sections qui se composent de plus de quatre étudiants**

```sql
SELECT section_id, section_name
FROM section
WHERE section_id IN ( SELECT section_id
                      FROM student
                      GROUP BY section_id
                      HAVING COUNT(section_id) > 4 )
```

Exercice 2.7.8 – **Donner la liste des étudiants premiers de leur section en terme de résultat annuel et qui n'appartiennent pas aux sections dont le résultat moyen est inférieure à 10**

```sql
-- Moyenne par section
SELECT AVG(year_result)
FROM student
GROUP BY section_id

-- Sections dont le résultat moyen est inférieure à 10
SELECT section_id
FROM student
GROUP BY section_id
HAVING AVG(year_result) < 10

-- Étudiants qui ne sont pas dans les section
-- dont le résultat moyen est inférieure à 10
SELECT *
FROM student
WHERE section_id NOT IN (
    SELECT section_id
    FROM student
    GROUP BY section_id
    HAVING AVG(year_result) < 10
)

-- Meilleur résultat par section
SELECT section_id, MAX(year_result)
FROM student
GROUP BY section_id
```

```sql
-- Info du meilleur étudiant par section
SELECT last_name, first_name, section_id
FROM student s
WHERE year_result = (
    SELECT MAX(year_result)
    FROM student s1
    WHERE s.section_id = s1.section_id
    GROUP BY section_id
)

```

```sql
-- Combiné des deux parties - version finale
SELECT last_name, first_name, section_id
FROM student s
WHERE section_id NOT IN (
    SELECT section_id
    FROM student
    GROUP BY section_id
    HAVING AVG(year_result) < 10
) AND year_result = (
    SELECT MAX(year_result)
    FROM student s1
    WHERE s.section_id = s1.section_id
    GROUP BY section_id
)
```

Solution utilisant une jointure ainsi qu'une requête imbriquée dans la clause « FROM »

```
1  select last_name,first_name,S1.section_id
2  from student as S1 inner join (select section_id, AVG(year_result) as average
3                                   from student
4                                 group by section_id
5                                   having AVG(year_result) >= 10)
6              as S2 on S1.section_id = S2.section_id
7  where year_result = (select MAX(year_result)
8                         from student
9                        where section_id = S1.section_id)
```

Exercice 2.7.9 – **Donner la section qui possède la moyenne la plus élevée. Le résultat présente le numéro de section ainsi que sa moyenne**

```sql
-- 1. Trouver la moyenne pour chaque section
SELECT section_id, AVG(year_result) as 'Moyenne'
FROM student
GROUP BY section_id
HAVING AVG(year_result) = 17


-- 2. Trouver la moyenne la plus élevée parmi les moyennes de chaque section
SELECT MAX(Maximum)
FROM (SELECT section_id, AVG(year_result) as 'Maximum'
FROM student
GROUP BY section_id) AS Nouvelle_Table
```

```sql
-- 3. Trouver la section qui possède la moyennes la plus élevée
-- 4. Indiquer le numéro et la moyenne de la section qui possède la moyennes la plus élevée

SELECT section_id, AVG(year_result) as 'Moyenne'
FROM student
GROUP BY section_id
HAVING AVG(year_result) = (
SELECT MAX(Maximum)
FROM (
SELECT section_id, AVG(year_result) as 'Maximum'
FROM student
GROUP BY section_id) AS Nouvelle_Table
)
```

```sql
-- Méthode Avec WITH
WITH moy_par_sect (section_id, average) AS (
    SELECT section_id, AVG(year_result)
    FROM student
    GROUP BY section_id
)
SELECT section_id, average 'AVG'
FROM moy_par_sect mps
WHERE mps.average = (
    SELECT MAX(average)
    FROM moy_par_sect
);


-- Même résultat avec TOP
SELECT TOP(1) section_id, AVG(year_result)
FROM student
GROUP BY section_id
ORDER BY AVG(year_result) DESC
```