Master 1 Humanités numériques – Bases de données

TD 3-4 : Interrogation de bases de données avec SQL

Cécile Favre – https://eric.univ-lyon2.fr

Préambule

1. Si elle n'est pas déjà créée, importer la base de données « univ » (cf. TD 1) à partir du fichier univ.sql. https://eric.univ-lyon2.fr/jdarmont/docs/univ.sql
2. Pour les questions suivantes, créer une requête par question en SQL (copier les différentes requêtes dans un document texte pour en garder la trace en reportant le numéro des questions).

Requêtes de base

01) Afficher la table ETUDIANT (tous les attributs) triée par ordre alphabétique du nom et du prénom de l'étudiant.e.

02) Afficher les nom et prénom des étudiant.es habitant Lyon.

03) Afficher les nom et prénom des étudiant.es habitant dans le Rhône (c'est à dire, celles et ceux dont le code postal commence par 69).

04) Afficher tous les étudiant.es (nom et prénom) dont le nom ne commence pas par M.

05) Afficher tous les étudiant.es (nom et prénom) dont le nom commence par B ou P.

06) Afficher tous les étudiant.es (nom, prénom et adresse) habitant dans une rue.

07) Afficher les épreuves (code, date) dont le lieu commence par S et a un 2 en huitième position.

08) Afficher la liste des étudiant.es (nom et prénom) habitant Bron ou Brignais.

9) Afficher les étudiant.es (nom, prénom et date de naissance) né.es avant le 01/01/1980.

10) Afficher les épreuves qui se sont déroulées entre le 20 et le 22 janvier 2013.

11) Afficher toutes les informations sur les étudiant.es habitant à Lyon et né.es avant 1980.

12) Afficher toutes les informations sur les étudiant.es habitant à Lyon ou né.es avant 1980.

13) Lister tous les étudiant.es qui, d'une part, habitent à Lyon et sont né.es avant 1980 ou, d'autre part, qui habitent à Bron ou Brignais.

14) Indiquer les codes des épreuves se déroulant dans un lieu dont l'appellation comporte un 2.

Requêtes de calcul, regroupements

15) Construire une requête indiquant pour chaque étudiant.e et chacune des épreuves sa note et un nouveau champ intitulé « NoteAugmentee » ajoutant 2 points à la note de départ.

16) Calculer la moyenne générale des notes obtenues (sans augmentation) par étudiant.e en précisant le numéro de l'étudiant.e.

17) Calculer la moyenne par épreuve en précisant le code de l'épreuve.

18) Calculer la moyenne générale par étudiant.e en n'affichant que les étudiant.es dont la moyenne obtenue est supérieure à 14.

19) Calculer par étudiant.e la moyenne en ECO en faisant la moyenne des notes par étudiant.e où le code des épreuves se limite à ECO1 ou ECO2.

20) Indiquer pour chaque épreuve le nombre d'étudiant.es l'ayant passée.

21) Pour chaque épreuve, déterminer les notes minimale et maximale.

22) Déterminer la note minimale et la note maximale parmi toutes les épreuves.

Requêtes de jointure

23) Afficher les nom, prénom, numéro d'étudiant.e, le nom de l'épreuve et sa date, l'intitulé de la matière ainsi que la note obtenue.

24) Afficher les nom, prénom des étudiant.es ayant fait une épreuve dans l'amphi Say.

25) Calculer la moyenne par épreuve en précisant le code de l'épreuve, le nom de l'épreuve et le lieu de l'épreuve.

26) Calculer la moyenne générale des notes obtenues par étudiant.e en précisant le numéro, le nom et le prénom de l'étudiant.e.

Correction

Requêtes de base

01)

```
SELECT * FROM etudiant
ORDER BY Nom, Prenom;
```

02)

```
SELECT Nom, Prenom
FROM etudiant
WHERE Ville = 'Lyon';
```

03)

```
SELECT Nom, Prenom
FROM etudiant
WHERE CP LIKE '69%';
```

04)

```
SELECT Nom, Prenom
FROM etudiant
WHERE Nom NOT LIKE ‘M%';
```

05)

```
SELECT Nom, Prenom
FROM etudiant
WHERE Nom LIKE 'B%'
OR Nom LIKE 'P%';
```

06)

```
SELECT Nom, Prenom, Rue,
CP, Ville FROM etudiant
WHERE Rue LIKE '%Rue%';
```

07)

```
SELECT CodeEpreuve,
Date FROM epreuve
WHERE Lieu LIKE 'S___2%';
```
(6 _ qui se suivent pour que le 2 soit en 8e position)

08)

```
SELECT Nom, Prenom
FROM etudiant
WHERE Ville IN ('Bron', 'Brignais');
```

09)

```
SELECT Nom, Prenom, DateNaiss
FROM etudiant
WHERE DateNaiss < '1980/01/01';
```

10)

```
SELECT *
FROM epreuve
WHERE Date BETWEEN '2013/01/20' AND '2013/01/22';
```

11)

```
SELECT *
FROM etudiant
WHERE Ville = 'Lyon'
AND DateNaiss < '1980/01/01';
```

12)

```
SELECT *
FROM etudiant
WHERE Ville = 'Lyon'
OR DateNaiss < '1980/01/01';
```

13)

```
SELECT * FROM etudiant
WHERE (Ville = 'Lyon' AND DateNaiss < '1980/01/01')
OR Ville IN ('Bron', 'Brignais');
```

14)

```
SELECT CodeEpreuve
FROM epreuve
WHERE Lieu LIKE '%2%';
```

Requêtes de calcul, regroupements

15)

```
SELECT NumEtu, CodeEpreuve, Ch_Note, Ch_Note + 2 AS NoteAugmentee
FROM passer;
```

16)

```
SELECT NumEtu, AVG(Ch_Note) AS MoyGenEtu
FROM passer
GROUP BY NumEtu;
```

17)

```
SELECT CodeEpreuve, AVG(Ch_Note) AS MoyEpreuve
FROM passer
GROUP BY CodeEpreuve;
```

Remarque : si on veut arrondir à 2 chiffres après la virgule, on peut utiliser la fonction ROUND(nombre_a_arrondir, nombre_chiffres_ap_virgule) :

```
SELECT CodeEpreuve, ROUND(AVG(Ch_Note), 2) AS MoyEpreuve
FROM passer
GROUP BY CodeEpreuve;
```

18)

```
SELECT NumEtu, AVG(Ch_Note) AS MoyGenEtu
FROM passer
GROUP BY NumEtu HAVING MoyGenEtu > 14;
```

19)

```
SELECT NumEtu, AVG(Ch_Note) AS MoyEco
FROM passer
WHERE CodeEpreuve IN ('ECO1', 'ECO2');
GROUP BY NumEtu ;
```

Remarque : Il n'y pas nécessairement 2 notes pour chaque étudiant.e dans l'expression de la requête.

20)

```
SELECT CodeEpreuve, COUNT(NumEtu) AS NbEtu
FROM passer
GROUP BY CodeEpreuve;
```

21)

```
SELECT CodeEpreuve, MIN(Ch_Note) AS NoteMini, MAX(Ch_Note) AS NoteMaxi
FROM passer
GROUP BY CodeEpreuve;
```

22)

```
SELECT MIN(Ch_Note) AS NoteMini, MAX(Ch_Note) AS NoteMaxi
FROM passer;
```

Requêtes de jointure

23)

Solution 1

```
SELECT      Nom, Prenom, etudiant.NumEtu, passer.CodeEpreuve, Date, Intitule,
            Ch_Note
FROM etudiant, passer, epreuve, matiere
WHERE etudiant.NumEtu = passer.NumEtu
 AND passer.CodeEpreuve = epreuve.CodeEpreuve
 AND epreuve.CodeMat = matiere.CodeMat;
```

Remarque : les prédicats de jointure sont indiqués dans la clause WHERE.

Solution 2

```
SELECT      Nom, Prenom, etudiant.NumEtu, passer.CodeEpreuve, Date, Intitule,
            Ch_Note
FROM ((etudiant INNER JOIN passer ON etudiant.NumEtu = passer.NumEtu)
INNER JOIN epreuve ON passer.CodeEpreuve = epreuve.CodeEpreuve)
INNER JOIN matiere ON epreuve.CodeMat = matiere.CodeMat;
```

Remarque : les jointures sont définies dans la clause FROM avec la syntaxe « INNER JOIN… ON » ; pas de clause WHERE ici puisque dans l'énoncé, il n'y a pas de sélection de données particulière.

Remarque pour les deux solutions : dans la clause SELECT il s'agit de désambiguïser les attributs qui sont présents dans deux tables en les préfixant par « nomtable. » (etudiant.NumEtu et passer.NumEtu, respectivement).

24)

Solution 1

```
SELECT DISTINCT Nom, Prenom
FROM etudiant, passer, epreuve
WHERE etudiant.NumEtu = passer.NumEtu
AND passer.CodeEpreuve = epreuve.CodeEpreuve
AND Lieu = 'Amphi Say';
```

Remarque : les prédicats de jointure sont indiqués dans la clause WHERE avec en plus le prédicat de sélection sur le lieu.

Solution 2

```
SELECT DISTINCT Nom, Prenom
FROM (etudiant INNER JOIN passer ON etudiant.NumEtu = passer.NumEtu)
INNER JOIN epreuve ON passer.CodeEpreuve = epreuve.CodeEpreuve
WHERE Lieu = 'Amphi Say';
```

Remarque : les jointures sont définies dans la clause FROM avec la syntaxe

« INNER JOIN… ON », la clause WHERE permet d'exprimer la sélection sur le lieu.

Remarque pour les deux solutions : dans la clause SELECT, le DISTINCT permet de dédoublonner les étudiant.es, compte-tenu du fait que l'on ne souhaite que les noms et prénoms. En effet, sinon, on aura les mêmes étudiant.es plusieurs fois, autant de fois qu'elles et ils ont participé à des épreuves dans l'amphi Say.

25)

Solution 1

```
SELECT epreuve.CodeEpreuve, Lieu, avg(Ch_Note) AS Moy_Epreuve
FROM passer, epreuve
WHERE passer.CodeEpreuve = epreuve.CodeEpreuve
GROUP BY epreuve.CodeEpreuve;
```

Solution 2

```
SELECT epreuve.CodeEpreuve, Lieu, avg(Ch_Note) AS Moy_Epreuve
FROM passer INNER JOIN epreuve ON passer.CodeEpreuve =
                                                    epreuve.CodeEpreuve
GROUP BY epreuve.CodeEpreuve;
```

Remarque pour les deux solutions : le mot clé AS permet de renommer l'intitulé de l'attribut, sinon la formule de calcul est utilisée par défaut, ici ce serait avg(Ch_Note). Il s'agit également de lever les ambiguïtés de noms d'attribut (y compris ici dans la clause GROUP BY).

26)

Solution 1

```
SELECT etudiant.NumEtu, Nom, Prenom, avg(Ch_Note) AS Moy_Gen
FROM etudiant, passer
WHERE etudiant.NumEtu = passer.NumEtu
GROUP BY etudiant.NumEtu;
```

Solution 2

```
SELECT etudiant.NumEtu, Nom, Prenom, avg(Ch_Note) AS Moy_Gen
FROM etudiant
INNER JOIN passer ON etudiant.NumEtu = passer.NumEtu
GROUP BY etudiant.NumEtu;
```

Remarque : pour toutes ces requêtes, il serait possible d'ajouter une clause de tri (ORDER BY) s'il était demandé dans l'énoncé d'ordonner les résultats.