Feuille de travaux pratiques n° 5
Requêtes SQL (agrégation, regroupement et jointures avancées)

Exercice 5.1 — Projets

Analyse de l'existant

Un employé est identifié par son matricule et on mémorise son nom, son poste, son salaire, la prime qu'il reçoit et sa date d'embauche.
Chaque employé travaille dans un département (commercial, direction, production, etc.) et a ou non un autre employé pour supérieur.
On stocke pour chaque département son numéro identifiant, son nom et son lieu. Chaque projet est réalisé par un ou plusieurs employés et chaque employé participe à un ou plusieurs projets.
Pour chaque projet, on stocke son code identifiant et son nom.

Dictionnaire de données

Attribut	Type	Description
matr	entier(11)	Matricule de l'employé
nome	char(20)	Nom de l'employé
poste	char(10)	Poste de l'employé
dateemb	date	Date d'embauche de l'employé
salaire	réel(10,2)	Salaire en euros
prime	réel(10,2)	Prime en euros
sup	entier(11)	Matricule du supérieur
numd	entier(6)	Numéro identifiant un département
nomd	char(20)	Nom du département
lieu	char(10)	Lieu du département
codep	entier(11)	Code identifiant un projet
nomp	char(20)	Nom du projet
fonction	char(20)	Fonction de l'employé dans le projet

Schéma relationnel

```
emp(matr, nome, poste, dateemb, salaire, prime, sup, numd)
dept(numd, nomd, lieu)
projet(codep, nomp)
participe(matr, codep, fonction)
```

1. Récupérez le fichier `projets.sql` sur Moodle. Ouvrez ce fichier avec un éditeur de texte (**gedit** par exemple).

Création des tables et insertion des données

2. Lancez **DB Browser for SQLite**. Créez une base de données `projets.db`. Allez sur l'onglet **Exécuter le SQL**.
3. Copiez les requêtes de création des tables depuis le fichier sql et exécutez-les.

Requêtes avec regroupements et agrégations

4. Affichez pour chaque département, le numéro du département et le salaire maximal des employés de ce département. Utilisez une clause `GROUP BY` pour regrouper les employés par numéro de département (3 lignes).
5. Modifiez la requête précédente pour que soient affichés également pour chaque département, le salaire minimal, le salaire moyen et le nombre d'employés de ce département (3 lignes).
6. Modifiez la requête précédente afin d'afficher également pour chaque département le nom et le lieu du département (3 lignes).
7. Affichez pour chaque employé son matricule et son nom ainsi que le nombre de projets auxquels il participe (6 lignes).
8. Affichez pour chaque couple (numéro de département, code projet) le nombre d'employés de ce département qui participent à ce projet. Spécifiez plusieurs champs de regroupement dans la clause `GROUP BY` (5 lignes).
9. Moyenne des salaires des employés qui ont le même supérieur direct que l'employé de nom `'Berger'` (1520.0).

Requêtes avec condition de sélection sur le regroupement

10. Liste des départements qui comptent strictement plus de 3 employés. On veut afficher le numéro et le nom du département suivis du nombre d'employés. Utilisez une clause `HAVING` de sélection sur le regroupement (2 lignes).
11. Liste des employés qui participent à plusieurs (strictement plus de un) projets. Afficher le matricule et le nom de l'employé ainsi que le nombre de projets auxquels il participe (2 lignes).
12. Liste des départements dont la masse salariale totale (total des salaires des employés du département) dépasse 10000 € (2 lignes).

Sous requête dans la clause `HAVING`

13. **Remarque importante :** on vous demande ici d'écrire une requête qui utilise l'opérateur `ALL`. Mais ni `ALL`, ni `ANY` ne sont définis en SQLite. Donc votre requête ne sera pas acceptée par SQLite. Vous écrirez une autre requête acceptée par SQLite à la question suivante.

    Liste des employés qui participent au plus grand nombre de projets. La sous-requête doit renvoyer le nombre de projets pour chaque employé. Vous devez définir un regroupement des projets par employés (clause `GROUP BY`). La requête principale doit sélectionner un employé si son nombre de projets est supérieur ou égal à toutes les valeurs renvoyées par la sous-requête (opérateur `ALL`). Vous devez ici aussi définir un regroupement des projets par employé dans la clause `GROUP BY` et définir le critère de sélection dans la clause `HAVING` (2 lignes).
14. Récrivez la requête précédente sans l'opérateur `ALL`. Vous pourrez utiliser l'opérateur `NOT EXISTS`. Il ne doit exister aucun autre employé qui participe à un plus grand nombre de projets (2 lignes).
15. Liste des départements dont le salaire moyen est le plus élevé. Il ne doit exister aucun autre département dont la moyenne des salaires est strictement supérieure (10, finance, paris, 3266.667).

Exercice 5.2 — Bibliothèque

On reprend la base de données des bibliothèques, le fichier `biblio.db` est disponible sur Moodle.
Ouvrez cette base de données avec *DB Browser for SQLite*.

Schéma relationnel

```
Abonné(NumAbo, NomAbo, PrénomAbo, AdrAbo, DateAbo)
Livre(ISBN, Titre, Éditeur, Année)
Emprunt(NumEmp, DateEmp, DateRet, NumAbo, ISBN)
Auteur(NumAut, NomAut, PrénomAut)
Écrit(NumAut, ISBN)
```

Requêtes SQL

Écrivez des requêtes dans l'onglet *Exécuter le SQL* pour répondre aux questions suivantes.

Manipulation des dates avec SQLite

1. Affichez toutes les informations ainsi que la durée d'emprunt de chaque emprunt (13 lignes). Qu'observez-vous ?
2. En fait, par défaut, SQLite calcule uniquement la différence des années. Mais il existe la fonction SQLite `julianday` pour convertir une date en nombre de jours et faire les calculs qu'on veut.

   Remplacez `dateret - dateemp` par `julianday(dateret) - julianday(dateemp)` et observez le résultat (13 lignes).

   Remarque : cette syntaxe n'est pas le SQL standard, mais propre à SQLite.

   Avec d'autres SGBD, pour avoir ce qu'on s'attend à obtenir avec `dateret - dateemp`, on peut écrire `DATEDIFF(dateret, dateemp)`. Il faut lire la documentation selon le SGBD avec lequel vous travaillez.

   Par exemple, pour SQLite, vous pouvez consulter

   — https://sqlite.org/lang_datefunc.html pour les dates,

   — https://www.sqlite.org/lang_corefunc.html pour d'autres fonctions de base.
3. Affichez la durée moyenne des emprunts (27.2).
4. Affichez les informations des emprunts dont la durée est strictement plus grande que la moyenne (2 lignes).

Requêtes avec regroupements et agrégations

5. Affichez pour chaque numéro d'abonné le nombre d'emprunts qu'il a réalisé. Utilisez une clause `GROUP BY` pour regrouper les emprunts par numéro d'abonné (5 lignes).
6. Modifiez la requête précédente pour que soient affichés également le nom et le prénom de l'abonné ainsi que les dates de son premier et de son dernier emprunts (5 lignes).
7. Durée moyenne des emprunts de chaque abonné. Les abonnés doivent être affichés par ordre décroissant des moyennes des durées des emprunts (5 lignes).
8. Affichez pour chaque abonné et pour chaque livre qu'il a emprunté le nombre d'emprunts de ce livre par cet abonné. Affichez, le numéro et le nom de l'abonné ainsi que le numéro et le titre du livre. Utilisez plusieurs champs dans la clause `GROUP BY` (9 lignes).
9. Affichez pour chaque éditeur le nombre d'emprunts de livres édités par cet éditeur (3 lignes).

Requêtes avec condition de sélection sur le regroupement

10. Liste des abonnés qui ont réalisé strictement plus de 2 emprunts. Affichez le numéro et le nom de l'abonné suivis du nombre d'emprunts. Utilisez une clause `HAVING` de sélection sur le regroupement (3 lignes).
11. Liste des auteurs qui ont participé à l'écriture de plusieurs (strictement plus de un) livres. Afficher les numéro et nom de l'auteur ainsi que le nombre de livres (2 lignes).
12. Liste des livres qui ont été écrits par plusieurs (strictement plus de un) auteurs (1 ligne).
13. Liste des abonnés dont la durée maximale des emprunts est supérieure à 30 jours (2 lignes).

Sous requête dans la clause `HAVING`

14. Liste des abonnés qui ont réalisé le plus grand nombre d'emprunts (1 ligne – 103, 4).
    — La sous-requête doit renvoyer le nombre d'emprunts pour chaque abonné à partir d'un copie `E2` d'`Emprunt`. Vous devez définir un regroupement des emprunts par abonné (clause `GROUP BY`).
    — La requête principale doit sélectionner un abonné s'il n'existe pas d'abonné ayant un nombre d'emprunts strictement supérieur. Pour ce faire, vous devez ici aussi définir un regroupement des emprunts par abonné dans la clause `GROUP BY` et définir le critère de sélection dans la clause `HAVING`.
    — Dans la sous-requête, vous devez rajouter une clause `HAVING` stipulant que le nombre d'emprunts pour `E2` est supérieur au nombre d'emprunts de l'abonné de la requête principale.

Remarque importante : une autre solution en SQL standard serait d'utiliser l'opérateur `ALL`. Mais ni `ALL`, ni `ANY` ne sont définis en SQLite.

15. Liste des abonnés qui n'ont pas réalisé le plus grand nombre d'emprunts. Il doit exister au moins un abonné ayant réalisé plus d'emprunts (4 lignes).
16. Liste des livres dont la moyenne des durées d'emprunt est la plus élevée. Il ne doit exister aucun autre livre dont la durée moyenne des emprunts est supérieure (1 ligne).