TP 20 : Bases de données

Repris d'un TP de Jean-Baptise Bianquis, du Lycée du Parc, et d'une base de donnée transmise entre profs de prépa de génération en génération.
Certaines questions utilisent des choses que nous n'avons pas encore vues (ou en tout cas pas en détail) en cours. N'hésitez pas à consulter le cours et à demander de l'aide si besoin.
Nous allons travailler avec une base de données des communes de France, organisée en trois tables dont le schéma est donné ci-dessous :

```
communes : (id : text, dep : text, nom : text, pop : integer)
departements : (id : text, reg : integer, nom : text)
regions : (id : integer, nom : text)
```

Les régions récemment fusionnées sont encore séparées dans cette base de données.

Exercice 1 – Mise en place

Pour ceux qui sont sur leurs ordis perso
Je vous conseille d'aller au plus simple. Plusieurs sites permettent d'importer une base de données au format `.sql` ou `.db` et d'exécuter des requêtes dessus. Une de ces bases simple à utiliser : `https://tchou.github.io/sqlsb/`. Dans l'archive, je vous fournis un fichier d'extension `.db` qui contient la même base de donnée que celle sur les PCs de la H210, au format SQLite.
Si vous voulez exactement le même cadre que sur ces ordis, il faut :

- Installer `postgresql`, et les extensions `SQLTools` et son driver PostgreSQL pour VSCode / Codium.
- importer la BDD dans une nouvelle base de donnée à l'aide de `psql`.
- créer un utilisateur avec les droits d'accès à cette base,
- utiliser l'extension pour se connecter à cette base avec cet utilisateur.

Je pourrais vous donner un document explicatif si vous avez envie. Sinon, `https://tchou.github.io/sqlsb/` marche très bien.

Pour ceux sur les PC du lycée
Pour le premier groupe, les autres devraient pouvoir profiter d'une connexion déjà paramétrée !

1. Lancez Codium, qui doit maintenant avoir une nouvelle extension apparaissant sur la barre de gauche nommée `SQLTools`.
2. Sélectionnez *Add New Connection*, choisissez `PostgreSQL` et remplissez les informations comme suis :
   - Connection name : cdr
   - Connect using : Server and Port
   - Server address : 100.116.38.5
   - Port : 5432
   - Database : cdr
   - Username : eleve
   - Use password : password
   - Password : h210lesage
3. Sélectionnez *Save Connection*, puis *Connect Now* s'il n'y a pas d'erreur apparente.
4. Un fichier `.sql` devrait s'ouvrir dans Codium. Enregistrez-le dans votre dossier personnel et vous pouvez commencer !
5. Quand votre requête est écrite, lancez là avec le petit bouton qui est apparu dans l'éditeur *Run on active connection*. Vous pouvez aussi paramétrer un raccourci clavier pour l'action *SQLTools Connection : Run current query* qui ne va pas lancer toutes les requêtes d'un fichier mais seulement celle où se trouve votre curseur (*Ctrl* + *Maj* + *P* puis on cherche le nom de l'action pour la trouver, et on clique sur la roue crantée pour paramétrer le raccourci clavier).

Petit rappel : on termine une requête par un simple ;. Par exemple, on écrira :

```sql
SELECT * FROM communes;
SELECT * FROM departements;
```

et *Run on active connection* lancera les deux requêtes et affichera leurs résultats dans deux onglets différents.

Exercice 2 – Structure de la base de données

1. Identifier, si elle existe, la clé primaire de chacune des tables.
2. Identifier les relations entre les différentes tables (c'est-à-dire les éventuelles clés étrangères).
3. Y a-t-il quelque chose de surprenant dans les types des colonnes (texte, entier…)? Pourquoi ce choix a-t-il été fait (allez voir en Corse…)?

Exercice 3 – Requêtes simples

Dans chacune des questions suivantes, écrire une requête affichant exactement les informations demandées.

1. La population de Lyon.
   Attention : le nom des communes (et des départements) est écrit dans la base avec la première lettre en majuscule et le reste en minuscules : `'Lyon'` et pas `'lyon'` ni `'LYON'`.
2. Le nombre, la population totale et la population moyenne des communes françaises.
3. Les noms et populations des dix communes les plus peuplées de France.
4. Le nombre de communes de plus de cinquante mille habitants en France.
5. Le nombre de noms de commune *distincts*.
6. Donner la liste des communes ayant moins d'habitants que de caractères dans leur nom.
   La fonction `length` *permet d'obtenir la longueur d'une chaîne de caractères.*
7. Les noms et populations des dix communes les moins peuplées de la Meuse.
   Question subsidiaire extra-informatique : expliquer ce que l'on observe.
8. Les départements français, classés par nombre de communes décroissant.
9. Les régions françaises, classées par population décroissante.

Exercice 4 – Encore des requêtes

1. Quelle région contient le plus grand nombre de communes de moins de 100 habitants (et combien de telles communes contient-elle)?
2. Quel est en France le nom de commune le plus fréquent?
3. Y a-t-il dans cette base de données deux communes portant le même nom *dans le même département?*
   Attention : un attribut apparaissant dans la clause `SELECT` doit obligatoirement apparaître dans la clause `HAVING`, sauf si on lui applique une fonction d'agrégation. Par exemple, on pourra écrire :

   ```sql
   SELECT a, COUNT(b)
   FROM t
   GROUP BY a
   ```

   mais on ne pourra pas écrire :

   ```sql
   SELECT a, COUNT(b), c
   FROM t
   GROUP BY a
   -- il faut écrire GROUP BY a, c qui va regrouper les enregistrements selon les
    ↪ valeurs du couple (a, c)
   ```

4. Parmi les communes ayant au moins un homonyme, laquelle est la plus peuplée ? On donnera en même temps la somme des populations de tous ses homonymes.
5. Donner les noms des départements n'ayant aucune commune de moins de 20 habitants. On écrira une requête utilisant `EXCEPT` et une ne l'utilisant pas.

Exercice 5 – Sous-requêtes et auto-jointure

1. Déterminer (**sans l'exécuter**) ce que renvoie la requête suivante :

```sql
SELECT nom, pop, dep
FROM communes AS c1
WHERE pop =
  (SELECT MAX(pop) FROM communes AS c2
  WHERE c1.dep = c2.dep);
```

2. En oubliant le fait que les SGBD optimisent vos requêtes, quel problème risque-t-on d'avoir en l'exécutant ?
3. Compléter la requête suivante pour qu'elle renvoie le même résultat :

```sql
SELECT c.nom, c.pop, c.dep
FROM communes AS c
JOIN
  (SELECT MAX(pop) as maxp, dep
  FROM communes
  GROUP BY dep)
  AS d
ON ..........
WHERE .......;
```

4. En appliquant la même technique qu'à la question précédente, déterminer, pour chaque département, la proportion de la population totale vivant dans la commune la plus peuplée. On classera par proportion décroissante.
   En SQL, `n / p` *renvoie le quotient de la division euclidienne si* n *et* p *sont des entiers, comme en C. On peut écrire* `1.0 * n / p` *si l'on veut une division flottante, comme en C.*
   Les deux premières lignes de ce que l'on souhaite obtenir :

nomC	nomD	pop	popT	ratio
Paris	Paris	2243833	2243833	1
Ajaccio	Corse-du-Sud	65542	143600	0.45642...