TP 2 : Base de données des communes de la France métropolitaine **- Les Requêtes simples**

✖ Conversion d'un tableau en table pour une base de données :

- ouvrez avec le tableur Libre Office le tableau de données issu de l'IGN* nommé **tab_commune.ods**
- enregistrez ce tableau au format texte d'extension .csv** : **tab_commune.csv** et en veillant à prendre comme jeu de caractères : Europe occidentale (ISO-8859-1).

*Institut National Géographique / ** csv : comma-separated values : valeurs séparées par des virgules.

✖ Pour créer la base de données bdd_communes :

- lancez Firefox.
- lancez SQLite Manager à partir du menu *Outils* de Firefox.
- sélectionner *Nouvelle base de données* depuis le menu *Base de données* de SQLite Manager.
- donnez lui le nom **bdd_communes** puis OK, puis sélectionnez le dossier dans lequel va être créée la base.
- le nom de la nouvelle base de données apparaît en bleu dans la fenêtre de gauche (avec son extension .sqlite).
  Rem : cette fenêtre peut être éventuellement cachée si de largeur au minimum ; Ajustez alors sa largeur.
  Pour l'instant elle est vide car elle ne contient aucune table : ◢Tables(0)

- sélectionnez *Importer* depuis le menu *Base de données*, puis :
  - sélectionnez le fichier **tab_commune.csv**.
  - dès la sélection du fichier, un nom de table dans laquelle les données seront importées est défini. Ce nom est par défaut identique à celui de la table, conservez-le pour l'instant.
  - modifiez le *Codage des caractères* proposé par défaut et choisir ISO-8859-1
  - sous l'onglet CSV, cochez la case : *Première rangée contient les noms de colonnes*, puis OK.
  - un message d'avertissement apparaît, lisez-le, puis OK.
  - une nouvelle fenêtre apparaît : pour chaque variable définissez son type (cf. annexe) et prenez comme unique clé primaire la colonne "id", puis OK.
  - une fois l'importation des données achevée la table est créée. Fermez l'assistant.

- la table **tab_communes** a été crée. Elle apparaît sous la fenêtre **bdd_communes.sqlite** : on a ◢Tables(1)
- cliquez sur **tab_communes** puis sur l'onglet Structure et observez la commande SQL que vous avez générée par l'intermédiaire de l'assistant de création des tables. On retrouve :
  - pour chaque colonne (d'indice Column ID 0 à 17) le nom et le type des données : INTEGER, VARCHAR, FLOAT
  - la commande SQL qui a permis de créer la table : CREATE TABLE "tab_communes" ...
  - le nombre total d'enregistrements correspondant au nombre de lignes (No. of records)
  - la colonne d'indice 0 de nom id qui sert de clé primaire (cette clé permet d'assurer un référencement unique de chaque ligne de la table)

- cliquez avec le bouton droit sur le nom de la table **tab_communes** et choisissez *Renommez une table*. Prenez comme nouveau nom **communes**. Remarquez l'instruction SQL qui permet de renommer la table (ALTER ...)
- cliquez sur l'onglet *Parcourir & rechercher* pour visualiser le contenu de la table. Vous pouvez parcourir la table avec les flèches en bas de la page. Le nombre total de lignes (36613) figure également en bas de la page.

- cliquez sur l'onglet *Exécuter le SQL* puis entrez la commande SQL suivante : SELECT * FROM communes ;
  Le contenu de toute la base s'affiche : table de 18 colonnes et 36613 lignes, ainsi que te temps d'exécution de la requête (tout en bas à droite).
  Rem : pour saisir rapidement une instruction SQL, appuyez-vous sur les instructions prédéfinies sous les onglets *Select / Data Manipulation* .... qu'il n'y alors qu'à modifier et compléter.

Remarque :

Les commandes de manipulations des données (*Data Manipulation*) avec l'effacement, l'insertion, le remplacement des données peut être fait directement par l'intermédiaire de l'interface proposée sous l'onglet *Parcourir & rechercher* en double-cliquant sur les valeurs à modifier (une fenêtre d'édition permettant la modification des valeurs s'ouvre).
L'intérêt d'utiliser les commandes SQL pour manipuler les données est dont relativement limité. Il en est de même pour la création (CREATE) et renommage (ALTER) des tables et index ou leur suppression (DROP).
Par contre il n'en est pas de même pour faire des requêtes (interrogation des données selon des critères).
Seule la saisie d'instructions SQL pertinentes permet de créer les requêtes attendues.

✖ Répondez à chacune des questions suivantes en formulant puis exécutant une requête SQL adaptée :

- Quelles sont les caractéristiques de la communes qui a pour id 586 ? et son nom seul ?

```
SELECT * FROM communes WHERE id=586
SELECT nom_comm FROM communes WHERE id=586
```

- Affichez la liste des noms et du nombre d'habitants des préfectures de France dans l'ordre alphabétique, puis dans l'ordre inverse.

```
SELECT nom_comm, population FROM communes WHERE statut="Préfecture" ORDER BY nom_comm
SELECT nom_comm, population FROM communes WHERE statut="Préfecture" GROUP BY nom_comm
```

pour l'ordre inverse :

```
SELECT nom_comm, population FROM communes WHERE statut="Préfecture" ORDER BY nom_comm DESC
```

Rem : Un clic sur le titre de la colonne permet aussi d'inverser l'ordre.
Avec GROUP BY : pas de DESC possible, il ne s'agit pas d'un tri mais d'un regroupement !
Si les 2 clauses sont présentes : GROUP BY doit précéder ORDER BY

- Donnez le nom des sous-préfectures de la région Rhône-Alpes, classées par ordre alphabétique.

```
SELECT nom_comm FROM communes  WHERE statut="Sous-préfecture" AND  nom_region="RHONE-ALPES" ORDER BY nom_comm
```

- Dressez la liste des régions (la région ne doit apparaître qu'une seule fois).

```
SELECT DISTINCT nom_region FROM communes ORDER BY nom_region
```

Rem : DISTINCT : permet d'éliminer les doublons

ou :

```
SELECT nom_region FROM communes GROUP BY nom_region
```

- Affichez les 3 communes de la Savoie qui ont la plus grande superficie.
Indication : La clause **LIMIT n** permet de limiter la requête aux **n** premiers enregistrements.
Cette clause est à écrire en toute fin de requête : SELECT ... FROM ... LIMIT n

```
SELECT nom_comm,superficie FROM communes WHERE code_dept=73 ORDER BY superficie DESC LIMIT 3
```

- Affichez les 10 communes les plus peuplées dans le département de la Savoie.

```
SELECT nom_comm,population FROM communes WHERE code_dept=73 ORDER BY population DESC LIMIT 10
```

- Afficher les 10 moins peuplées.

```
SELECT nom_comm,population FROM communes WHERE code_dept=73 ORDER BY population DESC LIMIT 10
```

- Affichez le classement des communes entre la 7ème et 27ème position par leur nombre d'habitants.
Indication : L'utilisation combinée des clauses **LIMIT n** et **OFFSET m** permet de limiter la requête aux n premiers enregistrements mais comptés à partir de l'enregistrement m.
Ces clauses sont à écrire en toute fin de requête : SELECT ... FROM ... LIMIT n OFFSET m

```
SELECT nom_comm,population FROM communes ORDER BY population DESC LIMIT 20 OFFSET 7
```

- Donnez la liste des communes en France dont le nom contient "gren".

```
SELECT nom_comm FROM communes WHERE nom_comm LIKE "%gren%"
```

- La liste classée par ordre alphabétique des communes de Savoie dont le nom commence par "Saint".

```
SELECT nom_comm FROM communes WHERE code_dept=73 AND nom_comm LIKE "Saint%" ORDER BY nom_comm
```

- Donnez pour tous les départements (selon leur ordre décroissant) le nom des communes classées selon leur superficie croissante également.

```
SELECT nom_comm, superficie, nom_dept FROM communes ORDER BY nom_dept, superficie
```

- Donnez les noms des départements 12 à 18, classés par ordre alphabétique.

```
SELECT DISTINCT nom_dept, code_dept  FROM communes WHERE (12<=code_dept) AND (18>=code_dept) ORDER BY code_dept
```

- Donnez la liste des sous-préfectures classées par région et département.

```
SELECT nom_comm, nom_region, nom_dept FROM communes WHERE statut="Sous-préfecture" ORDER BY  nom_region, nom_dept,nom_comm
```

ou (mais moins bien !) :

```
SELECT nom_comm, nom_region, nom_dept FROM communes WHERE statut="Sous-préfecture" GROUP BY  nom_region, nom_dept, nom_comm
```

Rem 1 : 3 regroupements qui permettent d'obtenir le même résultat que la requête précédente
Rem 2 : ne pas oublier nom_comm dans clause GROUP BY !! sinon le regroupement ne se fait que sur les régions et départements et on a au plus une sous-préfecture qui s'affiche pour chaque département !

- Comptez les sous préfectures regroupées par région et département.

```
SELECT nom_region, nom_dept, COUNT(nom_comm) AS nombre_sous_préfecture_dans_le_département
FROM communes WHERE statut="Sous-préfecture" GROUP BY  nom_region, nom_dept
```

- Evaluez et affichez pour les communes de Savoie leur densité de population (nbre d'habitants par km^2).

```
SELECT
nom_comm,
population*1000 AS nombre_habitants,
superficie/100 AS superficie_en_km2,
(population*1000)/(superficie/100) AS densite
FROM communes WHERE code_dept=73
ORDER BY densite DESC
```

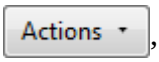
Pour les plus rapides

✖ Enregistrement du résultat d'une requête et traitement des données :

- Exécutez la requête suivante pour afficher pour chaque région de France leur densité (en habitants/km^2) :

```
SELECT nom_region,
SUM(population) AS nombre_hantitants_en_milliers,
0.01*SUM(superficie) AS superficie_en_km2,
1000*SUM(population)/(0.01*SUM(superficie)) AS densite
FROM communes GROUP BY nom_region
```

- Sélectionnez le résultat de cette requête qui apparaît sous la forme d'un tableau :
  - cliquez sur la première ligne du tableau pour la sélectionner
  - descendez tout en bas du tableau
  - appuyez sur la touche SHIFT ⇧
  - cliquez la dernière ligne : tout le tableau est sélectionné
  - cliquez enfin avec le bouton droit de la souris sur Copy Row(s) as CSV (MSExcel compatible) pour faire un **copié**
- Recopiez le contenu vers un tableur (Excel ou Calc de LibreOffice) :
  - lancez l'exécution du tableur
  - cliquez dans la case A1
  - faîtes un **collé** (on acceptera le jeu de caractères -Unicode- proposé par défaut)
- Vous pouvez consulter un exemple de traitement des données en ouvrant le fichier Calc suivant : population_par_region.ods
  Si vous avec une connaissance suffisante des tableurs, essayez d'obtenir les mêmes graphiques.

Rem : on aurait pu aussi, à partir de l'icône 

, enregistrer le résultat de la requête dans un fichier CSV.

✖ Exemples de requêtes sur la base de données en utilisant un script Python :

- copiez le fichier **commune.sqlite** directement sous la racine du disque dur.
- ouvrez le script **requetes_bdd_communes.py**
- exécutez le et observez l'affichage du résultat des 4 requêtes (sous la console ou sous la fenêtre graphique)
- saisissez les instructions nécessaires pour exécuter et afficher le résultat d'une requête simple de votre choix.

✖ Exécutez les requêtes suivantes, pour lister les villes (communes dont la population est supérieure à 2000 habitants) de la région Rhône-Alpes, et relevez leur temps d'exécution :

SELECT nom_comm,population,nom_region,code_reg FROM communes WHERE code_reg=82 AND population>=2	39ms
SELECT nom_comm,population,nom_region,code_reg FROM communes WHERE population>=2 AND code_reg=82	35ms

SELECT * FROM (SELECT nom_comm,population,nom_region,code_reg FROM communes) WHERE code_reg=82 AND population>=2	39ms
SELECT nom_comm,population,nom_region,code_reg FROM (SELECT * FROM communes WHERE code_reg=82 AND population>=2)	35 ms

Questions :
- l'ordre des termes (autour du AND) du prédicat logique de la clause WHERE a t-il de l'influence sur le temps d'évaluation de la requête ?
- l'ordre selon que l'on projette puis sélectionne ou l'inverse (sélection puis projection) a t-il de l'influence sur le temps d'évaluation de la requête ?

ANNEXE

Nom	Type	Description
id	INTEGER	clé primaire unique
code_comm	VARCHAR*	Il s'agit du code géographique de la commune permettant d'identifier la commune dans son département d'appartenance (texte de trois caractères).
insee_com	VARCHAR	Il s'agit d'un numéro de 5 caractères : le code du département suivi du code géographique de la commune. Pour les DOM, le 3ème caractère est commun au n° de département et au n° de commune.
nom_comm	VARCHAR	Nom de la commune (source INSEE). C'est un texte en majuscules non accentuées d'au plus 50 caractères.
statut	VARCHAR	Valeurs possibles : Capitale d'état / Préfecture de région / Préfecture / Sous-préfecture / Chef-lieu canton / Commune simple
x_chef_lieu	INTEGER	Abscisse du chef-lieu de la commune (en hectomètres) Rem : plus x est important, plus on est à l'est.
y_chef_lieu	INTEGER	Ordonnée du chef-lieu de la commune (en hectomètres). Rem : plus y est important, plus on est au nord.
x_centroid	INTEGER	Abscisse du centroïde de la commune (en hectomètres).
y_centroid	INTEGER	Ordonnée du centroïde de la commune (en hectomètres).
z_moyen	INTEGER	Altitude moyenne de la commune (en mètres).
superficie	INTEGER	Superficie de la commune en hectares. C'est la somme des surfaces des faces BD CARTO composant la commune (avant allégement géométrique et suppression des îles et enclaves).
population	FLOAT	Chiffre de population sans doubles comptes au dernier recensement, en milliers d'habitants, à une décimale. Pour Mayotte, ce chiffre provient du recensement de 1997.
code_cant	VARCHAR	Code géographique du canton auquel appartient la commune (texte de deux caractères).
code_arr	VARCHAR	Code géographique de l'arrondissement auquel appartient la commune (texte d'un caractère). Ce champ est vide pour les communes de Mayotte : il n'y a pas d'arrondissement dans cette collectivité départementale
code_dept	VARCHAR	Code géographique du département auquel appartient la commune (texte de deux caractères).
nom_dept	VARCHAR	Nom du département auquel appartient la commune. C'est un texte en majuscules non accentuées d'au plus 30 caractères.
code_reg	VARCHAR	Code géographique de la région à laquelle appartient la commune (texte de deux caractères).
nom_region	VARCHAR	Nom de la région à laquelle appartient la commune. C'est un texte en majuscules non accentuées d'au plus 30 caractères.

* VARCHAR : chaîne de caractères de taille variable et s'adaptant à la longueur du texte