

# TP INFORMATIQUE 17

## Bases de données.

### 1. Initialisation du TP

#### 1.1 Installation de SQLite Manager


*SQLite Manager est une interface graphique permettant de visualiser une base de données et de lancer des requêtes SQL.*

Il s'agit en fait d'une extension (ou module complémentaire) du navigateur Firefox.

- Lancer Firefox
- Aller dans le menu **Outils:Modules complémentaires**.
- Dans le champ de recherche, chercher **sqlite Manager**, puis l'installer (*redémarrer Firefox pour que l'installation soit effective*).
- SQLite Manager se lance par le menu **Outils** de Firefox.

#### 1.2 Accès à la base de données

Nous allons travailler sur un fichier de base de données appelée `locations.sqlite`

- Télécharger ce fichier sur le site de classe `pcsi.lamartin.fr`, rubrique informatique.
- Enregistrer ce fichier **en local** (sur le bureau par exemple) <sup>1</sup>.
- Lancer SQLite Manager (si ce n'est déjà fait) et se *connecter* à la base de données (menu **Base de données** ou icône .

### 2. Exploitation de la base de données

#### 2.1 Structure de la base

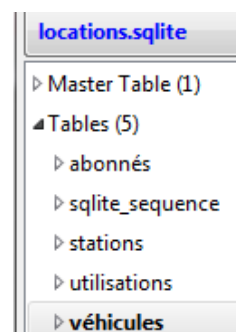
La base de données *locations.sqlite* reproduit de manière simplifiée un système de locations de voiture sur Lyon pour les mois de janvier et février 2013 :

- Des abonnés peuvent utiliser des véhicules au départ de stations. Le retour se fait dans la même station.
- Dans chaque station, il y a un ou plusieurs véhicules.
- Chaque utilisation donne lieu à une facturation liée au nombre de jours d'utilisation et au nombre de kilomètres parcourus.

1. Sur le menu latéral gauche, dérouler la rubrique **Tables**.

Après avoir cliqué sur une table, l'onglet **Parcourir et rechercher** permet de la visualiser.

Nous utiliserons les tables *abonnés*, *stations*, *utilisations*, *véhicules*.<sup>2</sup>



---

1. mais surtout pas dans votre dossier personnel sur le serveur, cela ne fonctionnerait pas.  
2. celle appelée *sqlite\_sequence* est apparue pour des raisons techniques mystérieuses...

2. A l'aide de l'onglet **Structure**, reproduire sur une feuille les schémas relationnels de chacune des tables. Au passage déterminer les domaines de chacun des attributs.
3. Entourer dans chaque schéma l'attribut jouant le rôle de clé primaire.  
Si besoin, comment les distinguer dans une requête SQL ?
4. Identifier aussi les attributs pouvant jouer le rôle de clé étrangère en les reliant d'une flèche à l'attribut auquel ils font référence.
5. Vu les conditions d'utilisation des véhicules, un attribut d'une des tables est en fait inutile. Lequel et pourquoi ?

## 2.2 Informations sur les abonnés (requêtes sur une seule table)

Les requêtes SQL s'effectuent via l'onglet **Exécuter le SQL**.  
Vous tapez votre requête puis l'exécutez en cliquant sur le bouton prévu à cet effet (ou par le raccourci clavier **Ctrl + ;**). Le résultat s'affiche en dessous.

### AVERTISSEMENT

SQLite Manager ne permet pas de sauvegarder les requêtes SQL effectuées.  
Il vous appartient de les copier-coller dans un fichier texte.

### A l'aide de requêtes SQL, donner les informations suivantes.

1. Afficher les nom et prénom de tous les abonnés.  
Même chose en triant les réponses par ordre alphabétique du nom (cf remarque suivante).  
*Remarque.* En fin de requête, le mot-clé **ORDER BY Att** permet d'ordonner les réponses suivant l'attribut **Att**. On peut encore rajouter **DESC** à la suite de l'attribut pour trier par ordre contraire.
2. Nom, prénom des abonnés résidant dans le 8<sup>e</sup>. Des femmes habitant dans le 8<sup>e</sup>.
3. Nombres respectifs d'hommes et de femmes.
4. Nombre d'abonnés par arrondissement de Lyon (on ne veut pas afficher les non-lyonnais).

## 2.3 Croisement des données (requêtes sur plusieurs tables)

A l'aide de requêtes SQL, donner les informations suivantes.

1. Afficher la liste des véhicules (de leurs plaques en fait) ainsi que leur station d'attache.  
Même liste classée par ordre alphabétique des stations.
2. L'identifiant de la station *Ambroise Courtois* puis les utilisations au départ de cette station (exceptionnellement 2 requêtes autorisées).
3. Les utilisations au départ de la station *Bellecour*.
4. Afficher le nombre total de km parcourus, de jours utilisés et la somme totale facturée pour le véhicule immatriculé VC-568-LK.
5. Afficher pour toutes les utilisations : le début, la fin, l'immatriculation du véhicule, le nom et le prénom de l'utilisateur.

*Indication : il est possible de faire plusieurs jointures en une requête.*

6. Afficher le nom et le prénom de tous les abonnés ayant utilisé le véhicule immatriculé CK-047-LC.
7. Afficher les nom et prénom de tous les abonnés ayant utilisé un véhicule basé à la Station *Gare Perrache* ainsi que les dates de début et fin des utilisations.
8. Afficher le nombre de locations, le nombre total de jours de locations ainsi que le nombre total de km parcourus par l'ensemble des femmes abonnées.
9. Afficher pour chaque abonné son nom, son prénom ainsi que la somme totale qui lui a été facturée. Classer par montant décroissant.
10. Afficher le nombre de locations et le nombre de jours de location par station (avec leur nom).
11. Afficher pour chaque nom de station sa **moyenne kilométrique par utilisation** ; classer par moyenne décroissante.
12. Afficher pour chaque nom de station sa **moyenne kilométrique par jour d'utilisation** ; classer par moyenne décroissante.

### 3. Pour aller plus loin

On propose le script Python suivant

```
# -*- coding: utf-8 -*-
import sqlite3

# la fonction
def requete(req):
    """ exécute une requete sur la bdd locations.sqlite

    retourne la liste des résultats
    """
    bdd = sqlite3.connect('locations.sqlite')
    cur = bdd.cursor()

    cur.execute(req)
    liste_res = cur.fetchall()

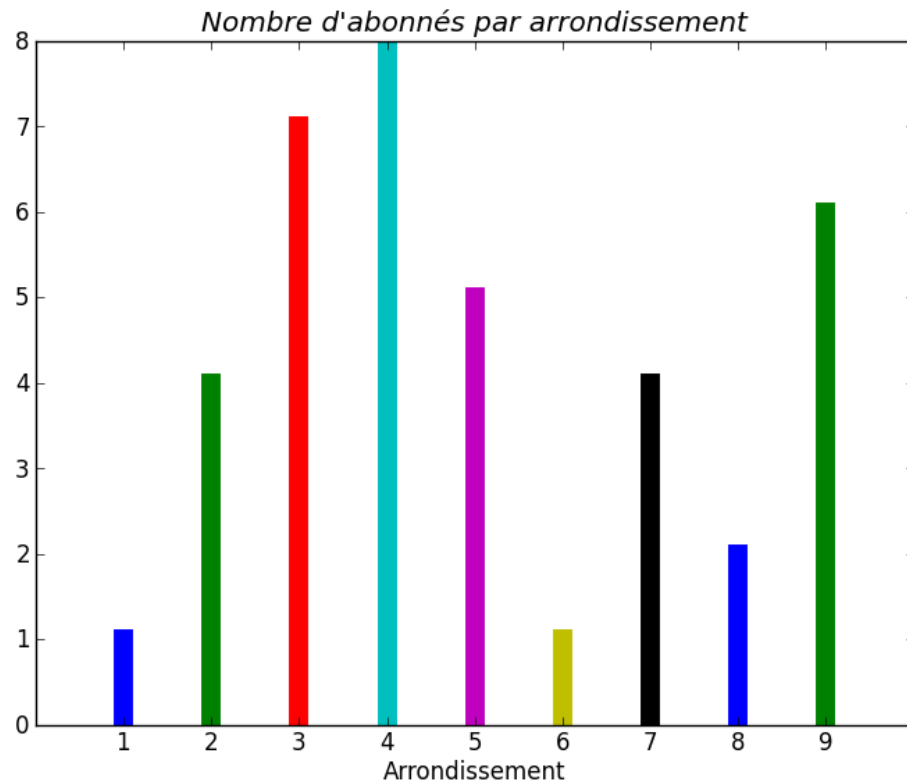
    cur.close()
    bdd.close()

    return(liste_res)

# un exemple d'utilisation
req = """ SELECT * FROM stations ; """
stations = requete(req)
print(stations)
```

1. Exécuter ce script et comprendre son fonctionnement.
2. Adapter ce script et le compléter pour tracer un diagramme en bâtons représentant le nombre d'abonnés par arrondissement lyonnais.

*Remarque.* Les bâtons ne sont que des segments verticaux ; vous pouvez les “épaissir” au moyen de l’option `linewidth=10` dans `plot`.



3. Représenter le chiffre d'affaires en fonction de l'arrondissement.
4. Représenter, toujours par arrondissement, le montant moyen dépensé par utilisateur.