Exercice 1 On dispose d'une base de données **db.sqlite3**. Ouvrir Spyder le temps de traiter le premier exercice.

1. Sur une feuille, représenter le schéma de la base, ainsi que les liens entre champs qui nous servirons à effectuer des jointures.
2. Trouver le nombre de notes présentes dans la base.
3. Trouver le numéro des devoir de type "DS" pour lesquels il y a au moins un élève ayant eu la note 20. (On pourra dans un premier temps aller chercher à la main l'id de ce type de devoir)
4. Trouver le nom, le numéro (des devoirs) et les élèves ayant eu la note 8.
5. Trouver la moyenne des notes de "Colle physique".
6. Donner les numéros des "Interro" dont la moyenne est supérieure ou égale à 5.
   INDICATION : la condition porte sur une donnée que l'on calcule, la clause `WHERE` ne convient pas, il faudra utiliser `HAVING`. Nous devons donc grouper les données.

Exercice 2 Le module *query.py* nous permet d'exécuter des requêtes sur notre base de données depuis l'interpréteur python.

```
import query
query.req('SELECT note, eleve, devoir_id FROM notes where note=20')
```

La valeur de retour est une liste de résultats : le premier résultat est la liste des noms de colonnes, les suivants sont les listes correspondant aux lignes de la table retournée. Ainsi la valeur de retour est une liste de liste. On pourra utiliser la syntaxe

```
L[1:]
```

(où `L` est une liste) pour récupérer les éléments d'indices $1, 2, \ldots$ d'une liste, c'est à dire tous sauf le premier.
Pour convertir un nombre en chaîne, on pourra utiliser **str** et si l'opération inverse est nécessaire l'une des fonctions **float, int**.

1. Créer une fonction python **repartition(devoir_id)** qui prend l'id d'un devoir en argument et affiche l'histogramme de répartition des notes.
   On utilisera la fonction **plt.hist(liste, bins=10)** où liste est la liste des données à représenter (pour nous la liste des notes du devoir, qu'il faudra convertir en nombres) et bins le nombre de barres voulues.
2. Créer une fonction **evolution(eleve, type_id)** qui trace la courbe des notes obtenues en fonction du numéro du devoir, pour un élève fixé et un type de devoir fixé.
   Rappel : la fonction **plt.plot(X, Y)** trace la courbe des points dont les abscisses sont la liste X et les ordonnées Y.
3. Même question (evolution2), mais cette fois le type de devoir est donné par la chaîne de caractère qui est son nom.
4. Créer une fonction **stats(L)** qui prend une liste de nombres et renvoie la liste [mini, maxi, moyenne, ecart_type] des éléments statistiques de cette liste.
   Rappel : l'écart type est la racine carrée de la variance, c'est à dire que pour des nombres $x_1, \ldots, x_n$ dont la moyenne est m, il vaut $\sqrt{\left(\frac{1}{n}\sum_{k=1}^{n} x_k^2\right) - m^2}$.
5. Créer une fonction **bilan(eleve, nom_type, numero)** qui analyse les résultat de l'élève donné pour ce devoir. Le but est de retourner une chaîne de caractères de la forme :

   ```
   ''Note de l'élève 4 pour le DS 5 : 12.0
   Note minimale : 0.0
   Note maximale : 20.0
   Moyenne : 9.47
   Ecart type: 6.21
   ''
   ```

   Pour arrondir les nombres flottant, on pourra utiliser **round**. Le caractère pour passer à la ligne est "$\backslash n$".
6. Ecrire une fonction **tri_devoir(eleve)** qui affiche deux listes de noms de devoirs (séparées par des virgules) : celle pour laquelle la note de l'élève est supérieure ou égale à la moyenne du devoir, et la liste des autres.
   On attend une sortie de la forme :

   ```
   ''Au dessus de la moyenne : DS 1, Interro 4, .....
   En dessous : DS 2, ...
   ''
   ```

   Pour améliorer la sortie on peut présenter les résultats sous forme de liste à puces.
7. Même question, mais pour les devoirs se situant dans ou hors de l'intervalle $[m - ec, m + ec]$ où m est la moyenne et ec l'écart type.