Préliminaires

Ouvrez Google Chrome puis installer l'extension SQLite Manager. Une fois cette extension installée, vous pouvez directement glisser-déposer la base de données voulue dans l'onglet correspondant puis les requêtes se tapent directement dans le cadre jaune contenant l'aide. Pour passer à la ligne utiliser SHIFT + Entrée et utiliser Entrée pour exécuter la requête voulue.

Rappel sur les requêtes

La structure principale est la suivante :

```
SELECT [DISTINCT]..., [COUNT/SUM...]
FROM ... [JOIN .. ON ..]
WHERE...
[GROUP BY... HAVING ...]
[ORDER BY...]
```

Méthodologie pour les requêtes simples

1. Identifier dans la question tous les champs (ou attributs, ou colonnes) qui doivent être affichés ou sur lesquels portent des conditions.
   Toutes les données à afficher/retourner doivent figurer dans la clause SELECT.
2. Lister la ou les tables qui contiennent les champs identifiés, construire la clause FROM contenant éventuellement une ou plusieurs jointures.
3. Écrire, si besoin la clause WHERE qui permet de filtrer les données en imposant des conditions sur un ou plusieurs champs. On peut utiliser les connecteurs logiques AND, OR ou la négation OR pour les conditions complexes.
4. La clause ORDER BY permet d'ordonner les lignes retournée en les triant suivant une ou des colonnes.

Écriture des jointures

La base utilisée comme exemple ici est décrite dans le document "Revisions-SQL.pdf" joint à ce TD.

1. La première chose à remarquer est qu'il y a des correspondances entre tables. Nous en identifions 2 : les colonnes `groupe` des tables `eleves` et `planning`, et ensuite la colonne `creneau_id` de la table `planning` qui fait référence à `id` de la table `creneaux`
2. Imaginons que nous souhaitons obtenir des données provenant des tables `eleves` et `planning`. La clause `FROM` pourrait alors être

   ```
   FROM eleves JOIN planning ON eleves.groupe = planning.groupe
   ```

   Remarquons que nous devons utiliser des noms qualifiés (de la forme *table.colonne*) pour éviter lambiguïté sur le nom `groupe`.
   Avec des alias de tables on peut par exemple obtenir

   ```
   FROM eleves AS e JOIN planning AS p ON e.groupe = p.groupe
   ```

Conseil : utiliser un fichier texte pour sauvegarder les requêtes qui fonctionnent.

I Explorer la base de données

Exercice 1
Lister les attributs (ou colonnes) des tables `countries` et `regions`. Identifier la clé primaire puis donner pour chaque table un autre attribut unique.

Exercice 2
Trouver les liens permettant de faire des jointures entre les 3 tables de cette base.

II Première requêtes

Exercice 3
Trouver tous les aéroport se situant en France (le code iso est "FR"), puis tous les grands aéroports allemands (code iso "DE").

Exercice 4
Donner les noms, latitudes et longitudes des aéroports sud-américains (trouver le code de contient dans la base).

Exercice 5
En utilisant la commande `SELECT DISTINCT`, trouver tous les types possibles pour les aéroports.
Combien y-a-t-il de bases d'hydravion ? On utilisera la fonction `COUNT(*)`.

Exercice 6
Donner de nombre d'aéroports dont la latitude est dans $[0, 30]$.

Exercice 7
Donner le nombre de pays dont le code de pays commence par un A.
Indication : les relations $\leqslant$ et $\geqslant$ entre chaînes de caractères se basent sur l'ordre alphabétique.

III Quelques jointures

Exercice 8
Trouver tous les aéroports de Tanzanie, en utilisant une condition sur le champ `name` de la table `countries` (le nom en question est “Tanzania”).

Exercice 9
Trouver tous les aéroports (via une jointure) dans la région dont le nom est “Ohio”.

Exercice 10
Donner le nom du pays, de la région et la ville pour chaque base d'hydravion.

Exercice 11
Plus délicat. Donner les noms des aéroports, noms des pays et villes pour les couples d'aéroports de même latitude. On veut obtenir deux colonnes pour les noms d'aéroports, deux pour les pays, deux pour les villes.

IV Grouper les données

Pour rappel, lorsque l'on utilise une fonction dagrégation, on peut regrouper les données suivant un attribut (ou colonne).

```
SELECT count(*) c, ...
FROM ...
WHERE ...
GROUP BY nom_colonne
```

Dans ce cas, les lignes partageant la même valeur dans la colonne citée seront regroupées et comptée ensemble. On obtient alors, dans la table résultat, une ligne par compte effectué c'est à dire une ligne par valeur différente dans la colonne nommée `nom_colonne`.

Exercice 12
Trouver le nombre d'aéroport par pays. On donnera ce nombre, plus le code iso du pays.

Exercice 13
Trouver, pour les pays en possédant, le nombre de de base d'hydravion en donnant en plus le nom du pays (et pas seulement son code).

Exercice 14
Donner la liste des noms de régions et le nombre d'aéroport par région, classée par ordre de nombres d'aéroports décroissant. On pourra utiliser `ORDER BY ... DESC`

Exercice 15
Donner pour chaque pays (affiché par son nom), la latitude et longitude moyenne de ses aéroports. On pourra utiliser `avg(colonne)`.

V Conditions calculées

Le résultat du calcul d'un fonction d'agrégation ne peut pas être utilisé directement dans une clause `WHERE`. Pour donner des conditions sur une colonne `count`, par exemple, on doit énoncer cette condition dans une clause `HAVING`

```
SELECT count(*) c, ...
FROM ...
WHERE ...
GROUP BY nom_colonne
HAVING c>=...
```

par exemple.

Exercice 16
Donner les pays qui possèdent entre 0 et 10 aéroports.

Exercice 17
Donner le nombre moyen d'aéroport par pays (en affichant le nom du pays, encore). Indication : sous-requête.
En déduire les pays qui possèdent moins d'aéroports que la moyenne.