- EXERCICE 1 -

La base de données utilisée dans cet exercice liste des informations sur des villes et pays du monde entier. Elle comprend trois tables :

- La table "country" contient les pays du monde, identifiés par un code à trois lettres ainsi que de multiples informations sur ces pays, la capitale de chaque pays étant un entier correspondant à l'identifiant de cette ville.
  `country` = ((Code : TEXT), (Name : TEXT), (Continent : TEXT), (SurfaceArea : INT), (Population : INT), (LifeExp : INT), (GNP : INT), (Capital : INT))
- La table "city" recense quelques grandes villes mondiales. A chaque ville est attribué un identifiant unique, le code du pays auquel elles appartiennent et d'autres informations.
  `city` = ((ID : INT), (Name : TEXT), (CountryCode : TEXT), (Population : INT))
- La table "countryLanguage" contient des informations sur les langues parlées dans chaque pays et le pourcentage de la population qui les parlent.
  `countryLanguage` = ((CountryCode : INT), (Language : TEXT), (Percentage : INT))

L'utilisation de l'aide mémoire SQL donné en annexe est utile pour répondre à certaines questions.

1. Identifier les clés primaires des tables "country" et "city" et justifier que le couple (CountryCode, Language) constitue une clé primaire de la table "countryLanguage".

   .....
   .....
   .....

2. Établir le schéma relationnel de cette base de données, en faisant apparaitre les clés étrangères.

country							
Code	**Name**	**Continent**	**SurfaceArea**	**Population**	**LifeExp**	**GNP**	**Capital**

city			
ID	**Name**	**CountryCode**	**Population**

countryLanguage		
CountryCode	**Language**	**Percentage**

3. Déterminer le nom des villes et leur population de la table "city" dont la population est supérieure à 1000000.

   .....
   .....

4. Déterminer le nombre de villes de la table "city".

   .....
   .....

5. Lister les villes de noms différents dans la table "city".

   .....
   .....

6. Déterminer le nombre de noms de villes différents dans la table "city".

   .....
   .....

7. Déterminer le noms des pays de la table "country" situées en "Europe" classés par ordre décroissant de population.

   .....
   .....

8. Déterminer la population du pays le plus peuplé d'Afrique ("Africa").

   .....
   .....

9. Déterminer le nom du pays le plus peuplé d'Afrique ("Africa"), en réutilisant la requête précédente.

   .....
   .....

10. Déterminer l'espérance de vie moyenne des pays d'Europe ayant un PNB (GNP dans la table) supérieur à 1000000.

    .....
    .....

11. Déterminer le PNB par habitant des pays d'Europe.

    .....
    .....

12. Déterminer la superficie de l'Afrique.

    .....
    .....

13. Déterminer la liste des villes d'Europe avec leur pays d'appartenance.

    .....
    .....

14. Déterminer le nombre de langues parlées en Europe.

    .....
    .....

15. Déterminer la liste des noms des pays où plus d'un quart de la population parle français ("French").

    .....
    .....

16. Déterminer la liste des noms des pays Européens dont la population est inférieure à la moyenne mondiale.

    .....
    .....

17. Déterminer le nom des pays où l'on parle l'anglais et le français.

    .....
    .....

18. Déterminer le nom des pays où l'on parle l'anglais ou le français.

    .....
    .....

19. Déterminer le nom des pays où l'on parle le français mais pas l'anglais.

    .....
    .....

20. Déterminer le nombre de villes de France dont la population dépasse la population moyenne des villes de France.

    .....
    .....

21. Déterminer le pourcentage de pays d'Europe pour lesquels le PNB par habitant est supérieur à la moyenne des pays de l'Europe.

    .....
    .....

AIDE-MÉMOIRE SQL

Des requêtes

`CREATE TABLE table`
Crée la table nommée table
`SELECT col1, col2, ... FROM table;`
Renvoie un tableau formé des colonnes nommées col1, col2 de la table nommée table. On peut utiliser * pour sélectionner toutes les colonnes et aussi la clause WHERE, en fin de requête, pour ne sélectionner que certains enregistrements.
`INSERT INTO table(col1, col2, col3, ...) VALUES (val1, val2, val3, ...);`
Ajoute un nouvel enregistrement dans la table nommée table en affectant aux champs col1, col2, col3, ... les valeurs val1, val2, val3, ... Les champs des colonnes absentes, qui ne sont pas de type auto-incrémenté, prennent la valeur NULL.
`UPDATE table SET col1 = val1, col2 = val2, ... WHERE condition`
Met à jour les colonnes col1, col2, ... de la table nommée table pour les enregistrements vérifiant la condition.
`SELECT col1, col2,... FROM table1 INNER JOIN table2 ON colA=colB...;`
Réalisation d'une jointure. Chaque colonne de col1, col2,.. appartient à l'une des deux tables, $colA$ appartient à une table et $colB$ à l'autre.
`SELECT ... FROM table1 ... UNION SELECT ... FROM table2 ...;`
Combine les résultats de plusieurs sélections d'enregistrements dans un unique tableau, chaque enregistrement n'étant sélectionné qu'une seule fois même s'il est présent dans les deux tables.
`SELECT ... FROM table1 ... INTERSECT SELECT ... FROM table2 ...;`
Cette commande permet donc de sélectionner les enregistrements communs à deux sélections.
`SELECT ... FROM table1 ... EXCEPT SELECT ... FROM table2 ...;`
Cette requête permet de sélectionner les résultats de table1 sans inclure les enregistrements de la table1 qui sont aussi dans la table2.

Des commandes et opérateurs

`WHERE` - Commande qui dans une requête permet de sélectionner les enregistrements d'une table qui respectent une condition.
`ORDER BY col ASC` - Permet d'obtenir un résultat d'une requête où les enregistrements sont triés dans l'ordre croissant suivant les valeurs du champ col. Il est possible de trier les données sur une ou plusieurs colonnes, par ordre croissant (ASC) ou décroissant (DESC).
`DISTINCT` - Placée après SELECT dans une requête, permet de ne sélectionner que des enregistrements distincts deux à deux.
`AND, OR, NOT` - Opérateurs logiques pour les clauses WHERE et ON.

Des fonctions d'agrégation

`MAX` - Retourne la valeur maximale d'une colonne dans un ensemble d'enregistrement. La fonction peut s'appliquer à des données numériques ou alphanumériques.
`MIN` - Retourne la valeur minimale d'une colonne d'un ensemble d'enregistrements. La fonction peut s'appliquer à des données numériques ou alphanumériques.
`SUM` - Retourne la somme des valeurs d'une colonne dans un ensemble d'enregistrements.
`AVG` - Retourne la moyenne des valeurs d'une colonne dans un ensemble d'enregistrements.
`COUNT` - Retourne le nombre d'enregistrement d'un ensemble d'enregistrements.