# I Dictionnaire et départements

(/0.5) 1. D'après les derniers relevés de l'INSEE, la population du Rhône est maintenant de 2 millions. Écrire une instruction pour mettre à jour `d`.

> Solution : `d[69] = 2.0`

(/1.5) 2. Écrire une fonction `somme(d)` renvoyant la somme des valeurs dans `d` (c'est-à-dire la population totale de tous les départements).

> Solution :
>
> ```
> def somme(d):
>     s = 0.0
>     for k in d:
>         s += d[k]
>     return s
> ```

(/2) 3. Écrire une fonction `min_population(d)` renvoyant le numéro de département le moins peuplé.

> Solution :
>
> ```
> def min_population(d):
>     n = 69 # on initialise avec un département quelconque
>     for k in d:
>         if d[k] < d[n]:
>             n = k
>     return n
> ```

# II Coupe du monde 1998

- equipe (nom, entraineur), qui contient par exemple ("France", "Aimé Jacquet")
- joueur (nom, age, nom_equipe), qui contient par exemple ("Zidane", 26, "France")
- match (id, nom_equipe1, nom_equipe2, stade, score_equipe1, score_equipe2), qui contient par exemple (42, "France", "Brésil", "Stade de France", 3, 0). On a toujours score_equipe1 $\geq$ score_equipe2.
- but (id_match, nom_joueur, minute), qui contient par exemple (42, "Zidane", 45)

(/1) 1. Écrire une requête SQL pour afficher le nom de l'entraîneur du Brésil.
▶ `SELECT entraineur FROM equipe WHERE nom = "Brésil";`

(/1) 2. Écrire une requête SQL pour afficher tous les joueurs de l'équipe de France.
▶ `SELECT nom FROM joueur WHERE nom_equipe = "France";`

(/1) 3. Écrire une requête SQL pour afficher tous les identifiants des matchs nuls.
▶ `SELECT id FROM match WHERE score_equipe1 = score_equipe2;`

(/1) 4. Écrire une requête SQL pour afficher l'âge moyen des joueurs français.
▶ `SELECT AVG(age) FROM joueur WHERE nom_equipe = "France";`

(/1) 5. Écrire une requête SQL pour afficher à quelle minute le but le plus rapide a été marqué.
▶ `SELECT MIN(minute) FROM but;`

(/1.5) 6. Écrire une requête SQL pour afficher le nombre total de buts marqués par la France.
▶ `SELECT COUNT(*) FROM joueur JOIN but ON nom = nom_joueur WHERE nom_equipe = "France";`

(/1.5) 7. Écrire une requête SQL pour afficher les noms des joueurs ayant marqué au moins 1 but au stade de France.
▶ `SELECT nom_joueur FROM but JOIN match ON id = id_match WHERE stade = "Stade de France";`

(/2) 8. Écrire une requête SQL pour afficher le nom du meilleur buteur (avec son nombre de buts).
▶ `SELECT nom_joueur, COUNT(*) FROM but GROUP BY nom_joueur ORDER BY COUNT(*) DESC LIMIT 1;`

## Prévention des collisions aériennes (CentraleSupélec)

**I.A** L'énoncé spécifie que l'on peut comparer une date et une heure avec une chaîne de caractères équivalente. De plus, l'opérateur d'agrégation permettant de compter le nombre d'enregistrements est `COUNT(*)`. On peut ainsi écrire :

```
SELECT COUNT(*) FROM vol WHERE jour='2016-05-02' AND heure<'12:00'
```

**I.B** Les informations à utiliser sont contenues dans deux tables : on doit afficher des identifiants de vol `id_vol` contenus dans la table `vol` et faire la sélection en particulier sur `ville`, dans la table `aeroport`. Il faut donc réaliser une jointure, c'est-à-dire l'assemblage des lignes correspondantes dans les deux tables. La condition de jointure est par conséquent `depart=id_aero`. On obtient :

```
SELECT id_vol FROM vol JOIN aeroport ON arrivee=id_aero
WHERE ville='Paris' AND jour='2016-05-02'
```

> Dès que l'on utilise deux tables dans la même requête, il y a un risque de nom identique pour deux colonnes. Ce n'est pas le cas ici, mais si les colonnes `id_vol` et `id_aero` s'appelaient toutes les deux `id`, alors le système ne pourrait pas comprendre la requête si on ne précise pas de quel `id` il s'agit. Il faudrait alors écrire respectivement `vol.id` et `aeroport.id`. Il est par ailleurs possible d'écrire ici `vol.id_vol`, mais cela n'apporte rien.

**I.C** Cette requête réalise une double jointure entre la table `vol` et deux fois la table `aeroport`, afin d'associer à chaque vol la ville et le pays des deux aéroports concernés. Elle sélectionne ensuite les vols du 2 mai 2016 allant d'un aéroport français à un autre aéroport français.

Cette requête liste les vols nationaux français du 2 mai 2016.

**I.D** Il est nécessaire d'utiliser deux fois la table `vol`. Comme dans l'exemple donné à la question précédente, il faut renommer les tables. Les conditions de jointure et de sélection sont ici exceptionnellement proches, au point que plusieurs solutions sont possibles en déplaçant les conditions du `ON` vers le `WHERE` ou inversement. Par exemple, en considérant que l'on joint les tables pour les vols ayant lieu le même jour au même niveau, puis que l'on sélectionne les vols ayant les mêmes points de départ et d'arrivée, on a

```
SELECT v1.id_vol AS id1, v2.id_vol AS id2
FROM vol AS v1 JOIN vol AS v2
ON v1.jour=v2.jour AND v1.niveau=v2.niveau
WHERE v1.depart=v2.arrivee AND v1.arrivee=v2.depart
AND v1.id_vol<v2.id_vol
```

> On aurait donc pu aussi joindre les tables à la condition d'aéroports correspondants, et réaliser la sélection sur les égalités de jour et de niveau. On pourrait même positionner les 4 égalités dans la condition de jointure.
>
> La dernière inégalité dans la condition de sélection sert à n'afficher qu'une seule fois chaque couple : sans elle, chaque couple de vols serait affiché deux fois, avec `id1` et `id2` permutant leurs valeurs. Demander cette condition supprime l'une des deux lignes. Les `id_vol` étant des chaînes de caractères, elles sont classiquement ordonnées suivant l'ordre lexicographique (ordre du dictionnaire, prenant en compte chiffres et lettres).