# Révisions Bases de données — Correction

## I. Requêtes sans jointure

**Question 1.** Combien d'habitants en France (en supposant que chaque habitant est rattaché à exactement une commune, ce qui est peut-être un postulat faux. . . ) ?

**Solution:**

```
SELECT SUM(population_2010) FROM communes
```

**Question 1bis.** Combien y-a-t-il de communes en France ? de communes ayant des noms différents ?

**Solution:**

```
SELECT COUNT(*), COUNT(DISTINCT nom) FROM communes
```

**Question 2.** Combien de communes dans le Calvados ?

**Solution:**

```
SELECT COUNT(*) FROM communes WHERE num_departement = 14
```

**Question 3.** Obtenez la liste des dix plus petites communes de France en terme de surface.

**Solution:**

```
SELECT nom, num_departement, surface FROM communes ORDER BY surface LIMIT 10
```

**Question 4.** Obtenez la liste des dix communes de France les plus peuplées.

**Solution:**

```
SELECT nom, num_departement, population_2010 FROM communes ORDER BY
population_2010  DESC LIMIT 10
```

**Question 5.** Quelles sont les 20 communes les plus densément peuplées du Calvados ?

**Solution:**

```
SELECT nom, population_2010/surface as densite FROM communes
WHERE num_departement = 14 ORDER BY densite DESC LIMIT 20
```

**Question 6.** Obtenez la liste des numéros des départements, et pour chaque département correspondant, le nombre de communes.

**Solution:**

```
SELECT num_departement, COUNT(*) AS nbcommunes  FROM communes
GROUP BY num_departement ORDER BY nbcommunes DESC
```

**Question 7.** Obtenez la liste des numéros des départements, et pour chaque département correspondant, la population totale. Ordonnez par population totale décroissante.

**Solution:**

```
SELECT num_departement, SUM(population_2010) AS population  FROM communes
GROUP BY num_departement ORDER BY population DESC
```

**Question 8.** Combien de communes françaises contiennent la lettre (« x » ou « X » dans leur nom ?

**Solution:**

```
SELECT COUNT(*)  FROM communes WHERE nom LIKE '%X%'
```

**Question 9.** Combien de communes françaises contiennent les cinq voyelles « a », « e », « i », « o », « u », « y » (en majuscule ou minuscule) dans leurs noms ?

**Solution:**

```
SELECT COUNT(*)  FROM communes WHERE nom LIKE '%a%' AND  nom LIKE '%e%'
AND  nom LIKE '%i%' AND  nom LIKE '%o%'  AND  nom LIKE '%u%'  AND  nom LIKE '%y%'
```

**Question 10.** Existe-t-elle une telle commune dans le Val-de-Marne ?

**Solution:**

```
SELECT nom FROM communes WHERE nom LIKE '%a%' AND  nom LIKE '%e%'
AND  nom LIKE '%i%' AND  nom LIKE '%o%'  AND  nom LIKE '%u%'  AND  nom LIKE '%y%'
AND num_departement = 94
```

**Question 11.** Quelles sont les 15 communes de France dont l'écart entre l'altitude la plus haute, et l'altitude la plus basse, est le plus élevé ?

**Solution:**

```
SELECT nom, zmax-zmin AS ecart FROM communes ORDER BY ecart DESC LIMIT 15
```

**Question 12.** Quelles sont les cinq communes du Val-de-Marne où la plus population a le plus augmenté entre 1999 et 2010 ? Répondez en augmentation absolue d'abord, puis en augmentation relative.

**Solution:**

```
SELECT nom, population_2010-population_1999 AS augmentation FROM communes
WHERE num_departement=94 ORDER BY augmentation DESC LIMIT 10
```

**Solution:**

```
SELECT nom, (1.0*population_2010-population_1999)/population_1999
AS augmentation_relative FROM communes WHERE num_departement=94
ORDER BY augmentation_relative DESC LIMIT 10
```

## II. Jointures à deux tables

**Question 13.** Obtenez la liste des départements (les noms), et pour chaque département correspondant, le nombre de communes. Vous ordonnerez par nombre de communes décroissant.

**Solution:**

```
SELECT d.nom, COUNT(*) as nbcommunes FROM communes AS c,departements AS d
WHERE c.num_departement=d.num_departement GROUP BY c.num_departement
ORDER BY nbcommunes DESC
```

**Question 14.** Obtenez la liste des départements des régions Auvergne et Aquitaine.

**Solution:**

```
SELECT r.nom  AS region, d.nom AS depart FROM regions AS r, departements AS d
WHERE r.num_region=d.num_region AND (r.nom='Auvergne' OR r.nom='Aquitaine')
ORDER BY region, depart
```

**Question 15.** Quelle est le population de chaque département (on veut les noms des départements en toutes lettres) ? Les départements seront classés par population décroissante.

**Solution:**

```
SELECT d.nom, c.num_departement, SUM(c.population_2010) AS population
FROM departements AS d, communes AS c WHERE d.num_departement=c.num_departement
GROUP BY d.num_departement ORDER BY population DESC
```

**Question 16.** Quelle est la densité de population de chaque département (on veut les noms des départements en toutes lettres) ?

**Solution:**

```
SELECT d.nom,  SUM(c.population_2010)/SUM(surface) AS densite FROM departements
AS d, communes AS c WHERE d.num_departement=c.num_departement
GROUP BY c.num_departement
```

**Question 17.** Obtenez la liste des régions de France, ainsi que le nombre de départements composant la région, rangée par nombre de départements décroissant.

**Solution:**

```
SELECT r.nom, COUNT(*) AS nbdepart FROM regions as r, departements as d
WHERE d.num_region = r.num_region GROUP BY r.num_region ORDER BY nbdepart DESC
```

On peut aussi afficher les départements de chaque région sur une seule ligne à l'aide de la fonction `GROUP_CONCAT` qui permet de regrouper les valeurs d'un groupe en une chaîne de caractère. Cela est utile pour regrouper des résultats en une seule ligne et éviter d'avoir autant de lignes qu'il y a de résultat dans ce groupe.

**Solution:**

```
SELECT r.nom  AS region, COUNT(*) AS nbdepart, GROUP_CONCAT(d.num_departement)
AS depart FROM regions AS r, departements AS d WHERE r.num_region=d.num_region
GROUP BY r.num_region ORDER BY nbdepart
```

**Question 18.** Obtenez la liste des départements (les noms) de plus de 1200000 habitants.

On utilise ici la clause `HAVING`, qui permet de sélectionner seulement certains groupes (un peu comme `WHERE` qui permet de sélectionner seulement certaines entrées).

**Solution:**

```
SELECT d.nom, SUM(population_2010) AS population FROM communes AS c,
departements AS d WHERE c.num_departement = d.num_departement
GROUP BY c.num_departement HAVING population > 1200000 ORDER BY population DESC
```

Voici un autre exemple d'utilisation de `HAVING`.

**Question 18 bis.** Donner la liste des départements (les noms) où il y a une ville de plus de 200000 habitants.

**Solution:**

```
SELECT d.nom, MAX(c.population_2010)  FROM communes AS c, departements AS d
WHERE c.num_departement = d.num_departement GROUP BY c.num_departement
HAVING MAX(c.population_2010)>200000
```

**Question 19.** Existe-t-il une commune de la région Aquitaine dont le nom contient les cinq voyelles « a », « e », « i », « o », « u », « y » (en majuscule ou minuscule) ?

**Solution:**

```
SELECT c.nom, c.num_departement FROM communes AS c, departements AS d
WHERE c.num_departement = d.num_departement AND d.num_region=2
AND c.nom LIKE '%a%' AND  c.nom LIKE '%e%'  AND c. nom LIKE '%i%'
AND  c.nom LIKE '%o%'  AND  c.nom LIKE '%u%'  AND c. nom LIKE '%y%'
```

**Question 19.1** Donnez toutes les communes ayant le même nombre d'habitants qu'une autre.

**Solution:**

```
SELECT c1.nom, GROUP_CONCAT(c2.nom), c1.population_2010 FROM communes as c1,
communes as c2 WHERE c1.population_2010=c2.population_2010
AND c1.nom <> c2.nom GROUP BY c1.nom ORDER BY c1.population_2010 DESC
```

## III. Jointures à trois tables

**Question 20.** Obtenez la liste des régions de France, et pour chaque région, la population totale, ainsi que le nom et le nombre d'habitants de la commune la plus peuplée.

**Solution:**

```
SELECT r.nom, SUM(c.population_2010) AS population,c.nom, MAX(c.population_2010)
FROM communes AS c , departements AS d, regions AS r
WHERE c.num_departement = d.num_departement AND d.num_region = r.num_region
GROUP BY r.num_region ORDER BY population DESC
```

**Question 21.** Obtenir la liste des régions (avec le nom), avec la population totale de chaque région.

**Solution:**

```
SELECT r.nom,SUM(c.population_2010) AS population FROM communes AS c,
departements AS d, regions as r WHERE c.num_departement = d.num_departement
AND d.num_region = r.num_region GROUP BY r.num_region ORDER BY r.nom
```

**Question 22.** Densité de population de chaque région ?

**Solution:**

```
SELECT r.nom,SUM(c.population_2010)/SUM(c.surface) AS densite FROM communes AS c,
departements AS d, regions as r WHERE c.num_departement = d.num_departement
AND d.num_region = r.num_region GROUP BY r.num_region ORDER BY densite DESC
```

**Question 23.** Quelles sont les 15 communes de France dont l'écart entre l'altitude la plus haute, et l'altitude la plus basse, est le plus élevé ? Cette fois, précisez le nom du département et le nom de la région pour chaque commune.

**Solution:**

```
SELECT c.nom, r.nom,d.nom,zmax-zmin AS ecart FROM communes AS c,
departements AS d, regions AS r WHERE c.num_departement = d.num_departement
AND d.num_region = r.num_region ORDER BY ecart DESC LIMIT 15
```

**Question 24.** Combien de communes en Basse-Normandie ?

**Solution:**

```
SELECT COUNT(c.nom) FROM communes AS c , departements AS d, regions AS r
WHERE c.num_departement = d.num_departement AND d.num_region = r.num_region
AND r.nom LIKE 'Basse Normandie'
```

## IV. Utilisation de sous-requêtes

**Question 25.** Obtenez (sans doublons) la liste des régions contenant des départements dont le nom commence par « c » ou « C » (vous devriez retrouver la Basse-Normandie, normalement !).

**Solution:**

```
SELECT r.nom FROM regions AS r WHERE EXISTS (SELECT * FROM departements AS d
WHERE r.num_region = d.num_region AND d.nom LIKE 'C%')
```

On pouvait aussi, plus simplement, utiliser le mot-clé `DISTINCT` :

**Solution:**

```
SELECT DISTINCT r.nom FROM departements as d, regions as r
WHERE d.num_region=r.num_region  AND d.nom LIKE 'C%'
```

**Question 26.** Obtenez (sans doublons) la liste des régions ne contenant pas de communes dont le nom contient les cinq voyelles.

**Solution:**

```
SELECT r.nom FROM regions AS r WHERE NOT EXISTS (SELECT * FROM communes AS c,
departements as d WHERE r.num_region = d.num_region
AND d.num_departement = c.num_departement AND c.nom LIKE '%A%'
AND  c.nom LIKE '%E%' AND c.nom LIKE   '%I%' AND c.nom LIKE  '%O%'
AND c.nom LIKE  '%U%' AND c.nom LIKE  '%Y%')
```

**Question 26.1** Donner toutes les communes ayant un nom identique à celui d'une autre commune.

Pour chaque commune de nom `c1.nom`, on peut tester si il existe des communes ayant même nom à l'aide d'une sous-requête.

**Solution:**

```
SELECT c1.nom,GROUP_CONCAT(c1.num_departement) FROM communes AS c1
WHERE EXISTS (SELECT c2.nom FROM communes AS c2
WHERE c2.nom = c1.nom AND c1.rowid  != c2.rowid) GROUP BY c1.nom
```

Mais cela est très lent, puisqu'il faut faire $N^2$ tests où $N$ est la taille de la table ($N \approx 37000$) ! La requête suivante est bien plus efficace :

**Solution:**

```
SELECT nom, num_departement FROM communes
WHERE nom IN (SELECT nom FROM communes GROUP BY nom HAVING COUNT(*)>1 )
ORDER BY nom, num_departement
```

Le défaut de la requête précédente est que, puisqu'une ne table ne peut pas contenir un nombre de colonnes variable suivant les lignes, chaque nom de commune apparaît autant de fois que le nombre de départements auxquels elle appartient.

La requête suivante permet d'afficher sur une seule ligne le nom des communes et tous les départements où l'on trouve une telle commune. Mais elle utilise la fonction `GROUP_CONCAT` qui n'est pas vraiment au programme.

**Solution:**

```
SELECT nom,COUNT(*),GROUP_CONCAT(num_departement) FROM communes
GROUP BY nom  HAVING (COUNT(nom)>1)
```

**Question 26.2** Nom des communes dont la population en 2010 est supérieure à la moyenne des populations des communes du même département.

**Solution:**

```
SELECT c2.nom, c2.population_2010, d2.nom, (SELECT AVG(c.population_2010)
FROM communes AS c, departements AS d WHERE c.num_departement=d.num_departement
AND c2.num_departement = c.num_departement GROUP BY d.num_departement) AS moyenne
FROM communes AS c2, departements as d2
WHERE d2.num_departement = c2.num_departement AND c2.population_2010 >=moyenne
ORDER BY moyenne DESC, c2.population_2010 DESC
```

## V. Un peu de dessin

**Question 27.** Représenter les communes de métropole (avec un numéro de département $\leqslant 95$) par des points de couleur fonction de l'altitude du point le plus haut dans la commune (points placés en fonction de longitude et latitude).

**Solution:**

```
import sqlite3
import matplotlib.pyplot as plt

Base = sqlite3.connect("geographie.sqlite")
res = Base.execute("SELECT longitude, latitude, zmax AS e FROM communes\
                    WHERE num_departement <= 95");

Lx=[]
Ly=[]
Lz=[]
```

```
for ligne in res:
    Lx.append(ligne[0])
    Ly.append(ligne[1])
    Lz.append(ligne[2])

plt.figure(1)
plt.scatter(Lx,Ly,c=Lz)

Base = sqlite3.connect("geographie.sqlite")
res = Base.execute("SELECT AVG(longitude), AVG(latitude), AVG(zmax) AS e FROM communes\
        WHERE num_departement <= 95 GROUP BY num_departement,canton");

Lx=[]
Ly=[]
Lz=[]

for ligne in res:
    Lx.append(ligne[0])
    Ly.append(ligne[1])
    Lz.append(ligne[2])

plt.figure(2)
plt.scatter(Lx,Ly,c=Lz)
plt.show()
```

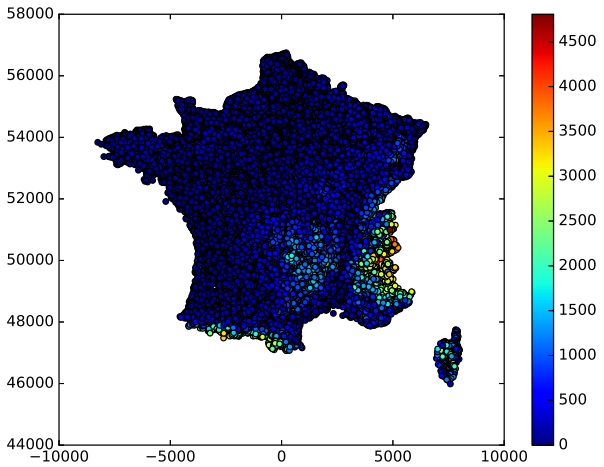



FIGURE 1 – $t = 0.5$

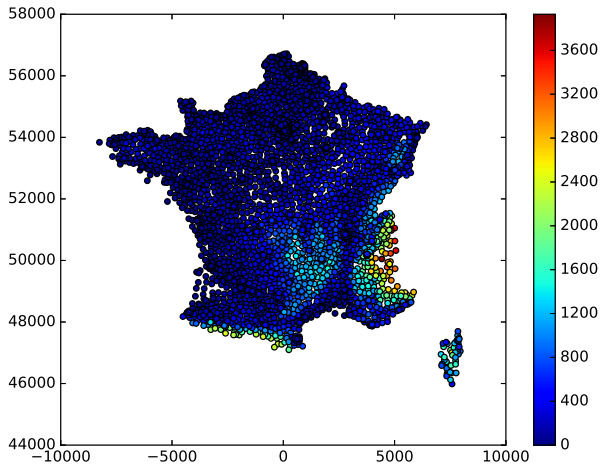



FIGURE 2 – $t = 0.2$

**Question 28.** Représenter par des points de couleur l'attractivité des cantons francais, représentée par l'augmentation relative du nom d'habitants entre 1999 et 2010 (points placés en fonction de longitude et lattitude).

**Solution:**

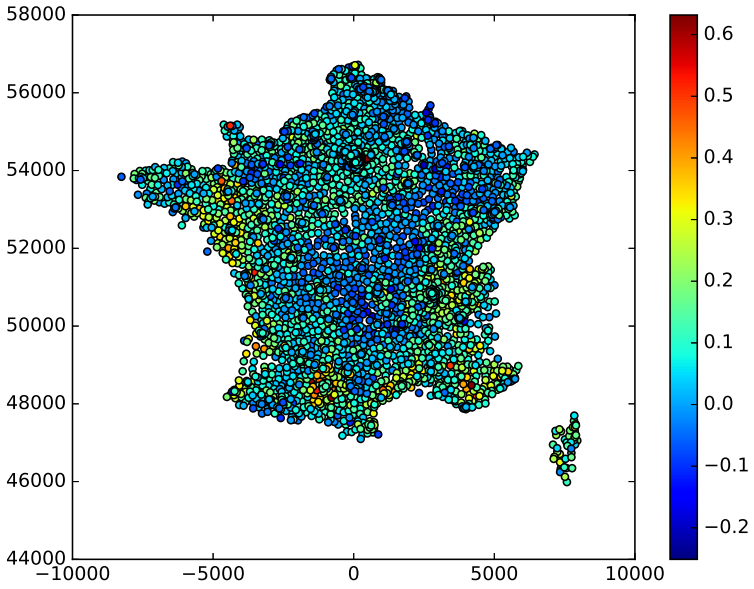
```
import sqlite3
import matplotlib.pyplot as plt
```

```
Base = sqlite3.connect("geographie.sqlite")
res = Base.execute("SELECT AVG(longitude), AVG(latitude),\
1.*SUM(population_2010)/SUM(population_1999) - 1\
     AS e FROM communes WHERE num_departement <= 95\
     GROUP BY num_departement,canton");

Lx=[]
Ly=[]
Lz=[]

for ligne in res:
    Lx.append(ligne[0])
    Ly.append(ligne[1])
    Lz.append(ligne[2])

plt.scatter(Lx,Ly,c=Lz)
plt.colorbar()
plt.show()
```




**Question 29.** Représenter par des points les 5000 plus grosses communes de métropole (points placés en fonction de longitude et lattidude).

**Solution:**

```
import sqlite3
import matplotlib.pyplot as plt

Base = sqlite3.connect("geographie.sqlite")
res = Base.execute("SELECT longitude, latitude, population_2010/surface\
        AS e FROM communes WHERE num_departement <= 95\
        ORDER by e DESC LIMIT 5000");
```

```
Lx=[]
Ly=[]
Lz=[]

for ligne in res:
    Lx.append(ligne[0])
    Ly.append(ligne[1])
    Lz.append(ligne[2])

plt.scatter(Lx,Ly,c=Lz)
plt.colorbar()
plt.show()
```