TP

6

BASES DE DONNÉES

Télécharger le fichier `villes` qui se trouve à l'adresse suivante :

`http://pellerin.xyz/pc/villes.sqlite`

La structure de cette base de données est la suivante :

`departement`

Nom	Type de données
numdep	char (3)
nom	char (50)
cheflieu	CHAR (70)
numreg	integer

`region`

Nom	Type de données
numreg	integer
nom	char (50)
cheflieu	CHAR (70)

`ville`

Nom	Type de données	Primary Key
idVille	integer	🔑
numdep	VARCHAR (3)	
nom	VARCHAR (100)	
codepostal	INTEGER	
codeINSEE	INTEGER	
numcanton	INTEGER	
pop2010	INTEGER	
pop1999	INTEGER	
surface	DECIMAL	
longitude	DECIMAL	
latitude	DECIMAL	
zmin	INTEGER	
zmax	INTEGER	

I - Quelques requêtes sans jointure

Dans ces requêtes, on manipulera uniquement les numéros des départements, pas leur nom. Par défaut les chiffres de populations seront ceux de 2010.

1. Donner la surface de Paris.
2. Donner la liste des communes de l'Aisne (02) de moins de 50 habitants dans l'ordre croissant du nombre d'habitants.
3. Donner la liste des 10 communes françaises les plus étendues en surface, avec leur numéro de département. Même question en se limitant à la métropole (numéro de département < 96). Et les cinq suivantes ?

4. Combien y a-t-il d'habitants dans l'ensemble des communes du département numéro 31 ?
5. Combien y a-t-il de communes dans le département numéro 31 ?
6. Donner la liste des 10 départements qui contiennent le plus de communes. On donnera juste le numéro du département et le nombre de communes.

II - Requêtes avec jointure

Dans ces requêtes on manipulera les noms des départements et des régions.

7. Donner la liste des 10 départements qui contiennent le plus de communes. On donnera cette fois le nom du département et le nombre de communes.
8. Donner la liste des 10 départements les moins peuplés (en considérant que la population est la somme des habitants par commune, comptage de 2010) : encore une fois on demande le nom.
9. Quelle était la liste des chefs-lieux de départements de l'ancienne région (2010) Picardie ?
10. Donner la liste des départements (on veut les noms) de plus de 1,2 million d'habitants.
11. Donner les 10 départements dont les chefs-lieux sont les plus peuplés. Même chose avec les 10 moins peuplés. Vérifier la cohérence de vos résultats.

III - Requêtes avec double jointure

12. Indiquer la liste des 10 communes les plus peuplées, avec leur région.
13. Quelle est la densité de population de chaque région (faire une liste alphabétique) ?

IV - Pour continuer à s'entrainer...

Écrire une requête en langage SQL :

14. qui donne la surface de Toulouse ;
15. qui donne le nom du département de la ville de Bordeaux ;
16. qui détermine la plus grande surface d'une ville de la métropole ;
17. qui donne les nom et surface des villes de plus grande et plus petite surfaces de la métropole ;
    (la technique utilisée pour cette question peut vous resservir par la suite)
18. qui donne le nom et le département (par son nom) de la ville de plus petite surface de la métropole ;
19. qui classe par surface décroissante les villes Bordeaux, Lyon, Paris, Toulouse ;
20. qui donne le numéro de département de la Guadeloupe ;
21. qui détermine la surface de la Guadeloupe ;
22. qui donne le nom de la ville la plus peuplée de Martinique (population 2010) ;
23. (plus difficile) qui donne le nom de la deuxième ville la plus peuplée de Martinique (et uniquement celui-ci)
24. qui détermine la liste des département dont au moins le nom d'une commune comporte 'CLERMONT' ;
    (on pourra utiliser l'opérateur de comparaison de chaînes LIKE : LIKE "D%ONT" renverra TRUE pour toute chaîne de caractères commençant par D, finissant par ONT(% remplace n'importe quelle séquence de caractères : DUPONT,DUMONT..)
25. qui donne les villes de métropole de plus de 8000 habitants (population 2010) situées en totalité à plus de 1000 m d'altitude ;
26. qui donne le nom, la population et l'altitude maximale de la ville de code postal 74400 ;
27. qui détermine l'altitude minimale et l'altitude maximale de la Haute-Savoie ;
28. qui détermine le nombre de villes du départenemnt 95 situées en totalité à plus de 100 m d'altitude ;
29. qui détermine le nombre de villes du département du Val-d'Oise situées en totalité à plus de 100 m d'altitude ;
30. qui donne les noms des départements de métropole dont les villes ont en moyenne une altitude minimale supérieure ou égale à 500 m.

31. qui donne les noms des départements de métropole situés en totalité à plus de 200 m d'altitude.

32. dont la réponse est le nombre de départements de métropole situés en totalité à plus de 200 m d'altitude ;

33. qui détermine l'altitude maximale de la région Corse ;

34. (plus difficile) qui détermine l'altitude du point culminant de Corse ainsi que le nom de(s) commune(s) sur lequel il se trouve ;

35. qui donne la liste des départements de métropole ayant au moins une commune située en totalité à plus de 1000 m d'altitude, avec leur nombre de telles communes, en les classant à partir du département en ayant le plus ;

36. (plus difficile) qui donne la liste des départements de métropole ayant au moins une commune située en totalité à plus de 1000 m d'altitude, avec leur pourcentage de telles communes, en les classant à partir du département en ayant le plus grand pourcentage.

V - Exploiter des résultats avec Python

V.1 - Utilisation du module sqlite3

- Il faut d'abord l'importer :

```
import sqlite3 as sql
```

- Puis ouvrir une connexion avec la base : par exemple

```
Base = sql.connect('C://Users/sebas/Documents/PC-info-2025-2026/TP-06/villes.sqlite')
```

Vous pouvez récupérer le chemin en allant sur le fichier de la base et en cliquant sur Propriétés, changer éventuellement ensuite les \ en des /.

- Pour effectuer une requête

```
req = ''SELECT numreg,nom FROM region''
result = Base.execute(req)
```

- On peut alors récupérer les différents enregistrements par un for, et les attributs successifs par enreg[0], enreg[1]...

```
for enreg in result:
  print("Numéro : ",enreg[0],"-",enreg[1])
```

- Pour un tracé : on peut utiliser `scatter` pour créer des points de couleur variable en fonction d'une liste T de valeurs.

```
import matplotlib.pyplot as plt
plt.axis('equal') #meme echelle en x et y
plt.scatter(X,Y,c=T)
```

V.2 - Mise en œuvre

Pour rester en métropole on se limitera systématiquement à des numéros de département inférieurs à 95. On passera de la longitude à la coordonnée x en multipliant par cos $\frac{2\pi}{9}$ (ceci approche bien la projection de Mercator pour la zone correspondant à la France).
Pour un meilleur résultat, on pourra prendre `np.log10(T)` (question 1) ou `np.log10(T+2)` (questions 2 et 3) à la place place de T pour définir c.

1. Placer les 5000 communes les plus peuplées sur un dessin à partir de leur longitude et leur latitude.

2. Représenter l'altitude maximale de chaque canton. On représentera donc un canton par un seul point, en faisant la moyenne de la longitude et celle de la latitude. On prendra pour T la liste des altitudes.

3. Représenter les accroissements de population 1999-2010 de chaque canton, en pourcentage. On représente là aussi un canton par un seul point.

Solutions des requêtes

1. Donner la surface de Paris.

```sql
  SELECT surface
    FROM ville
   WHERE nom = 'PARIS'
```

105.4

2. Donner la liste des communes de l'Aisne (02) de moins de 60 habitants.

```sql
SELECT nom
  FROM ville
 WHERE numdep = 2 AND
       pop2010 < 50
 ORDER BY pop2010
```

17 communes de Tannières (17 habitants) à Gibercourt (45 hab) en passant par Eparcy (44 hab)

3. Donner la liste des 10 communes françaises les plus étendues en surface, avec leur numéro de département. Même question en se limitant à la métropole (numéro de département < 96)?

```sql
  SELECT nom,
       numdep
    FROM ville
   ORDER BY surface DESC
   LIMIT 10
```

MARIPASOULA 973
REGINA 973
CAMOPI 973
MANA 973
SAINT-ELIE 973
SAINT-LAURENT-DU-MARONI 973
SAUL 973
ROURA 973
IRACOUBO 973
POMPIDOU PAPA ICHTON 973

```sql
SELECT nom,
       numdep
  FROM ville
 WHERE numdep < 96
 ORDER BY surface DESC
 LIMIT 10
```

ARLES 13
SAINTES-MARIES-DE-LA-MER 13
LARUNS 64
MARSEILLE 13
SAINT-MARTIN-DE-CRAU 13
LACANAU 33
SAINT-PAUL-SUR-UBAYE 4
SARTENE 2A
NEVACHE 5
HOURTIN 33

```sql
SELECT nom,
       numdep
  FROM ville
 WHERE numdep < 96
 ORDER BY surface DESC
 LIMIT 5
 OFFSET 10
```

```
AIX-EN-PROVENCE  13
CALENZANA        2B
HAGUENAU         67
BRESSUIRE        79
LA TESTE-DE-BUCH 33
```

4. Combien y a-t-il d'habitants dans l'ensemble des communes de Haute-Garonne ?

```sql
SELECT SUM(pop2010)
  FROM ville
 WHERE numdep = 31
```

1243641

5. Combien y a-t-il de communes en Haute-Garonne (département numéro 31) ?

```sql
SELECT COUNT( * )
  FROM ville
 WHERE numdep = 31
```

589

6. Donner la liste des 10 départements qui contiennent le plus de communes. On donnera juste le numéro du département et le nombre de communes.

```sql
SELECT numdep,
       COUNT( * ) AS nbCommunes
  FROM ville
 GROUP BY numdep
 ORDER BY nbCommunes DESC
 LIMIT 10
```

```
62 895
2 816
80 782
76 745
57 730
14 706
21 706
60 693
27 675
59 650
```

7. Donner la liste des 10 départements qui contiennent le plus de communes. On donnera cette fois le nom du département et le nombre de communes.

```sql
SELECT departement.nom, COUNT( * ) AS nbCommunes
  FROM ville
       JOIN
       departement ON ville.numdep = departement.numdep
 GROUP BY departement.numdep
 ORDER BY nbCommunes DESC
 LIMIT 10
```

```
nom              nbCommunes
PAS-DE-CALAIS    895
AISNE            816
SOMME            782
SEINE-MARITIME   745
MOSELLE          730
CALVADOS         706
COTE-D'OR        706
OISE             693
EURE             675
NORD             650
```

8. Donner la liste des 10 départements les moins peuplés (en considérant que la population est la somme des habitants par commune, comptage de 2010) : encore une fois on demande le nom.
Pour varier les possibilités de rédaction des requêtes, on utilise ici des alias mais cela n'a rien d'indispensable

```
SELECT d.nom,
       SUM(pop2010) AS popdep
  FROM ville v
       JOIN
       departement d ON v.numdep = d.numdep
 GROUP BY d.numdep
 ORDER BY popdep
 LIMIT 10
```

```
nom                       popdep
ST PIERRE ET MIQUELON       6080
LOZERE                     77082
CREUSE                    123029
HAUTES-ALPES              136971
TERRITOIRE DE BELFORT      142911
CORSE-DU-SUD              143600
CANTAL                    148162
ARIEGE                    152038
ALPES-DE-HAUTE-PROVENCE   160149
HAUTE-CORSE               166093
```

9. Quelle était la liste des chefs-lieux de départements de l'ancienne région (2010) Picardie ?

```
SELECT d.cheflieu
  FROM departement d
       JOIN
       region r ON d.numreg = r.numreg
 WHERE r.nom = 'PICARDIE'
```

```
cheflieu
AMIENS
BEAUVAIS
LAON
```

10. Donner la liste des départements (on veut les noms) de plus de 1,2 million d'habitants.

```
SELECT d.nom
  FROM ville v
       JOIN
       departement d ON v.numdep = d.numdep
 GROUP BY d.numdep
HAVING SUM(pop2010) > 1200000
```

```
 nom
BOUCHES-DU-RHONE
HAUTE-GARONNE
GIRONDE
ISERE
LOIRE-ATLANTIQUE
NORD
PAS-DE-CALAIS
RHONE
PARIS
SEINE-MARITIME
SEINE-ET-MARNE
YVELINES
ESSONNE
HAUTS-DE-SEINE
SEINE-SAINT-DENIS
VAL-DE-MARNE
```

11. Donner les 10 départements dont les chefs-lieux sont les plus peuplés. Même chose avec les 10 moins peuplés. Vérifier la cohérence de vos résultats.

```
SELECT departement.nom
  FROM ville JOIN departement
       ON departement.numdep = ville.numdep
  WHERE departement.cheflieu = ville.nom
 ORDER BY v.pop2010 DESC
 LIMIT 10
```

nom
PARIS
BOUCHES-DU-RHONE
RHONE
HAUTE-GARONNE
ALPES-MARITIMES
LOIRE-ATLANTIQUE
BAS-RHIN
HERAULT
GIRONDE
NORD

```
SELECT d.nom
  FROM ville AS v
       JOIN
       departement AS d ON d.numdep = v.numdep AND
                           d.cheflieu = v.nom
 ORDER BY v.pop2010 ASC
 LIMIT 10
```

nom
ARDECHE
ARIEGE
GUADELOUPE
LOZERE
CREUSE
MAYOTTE
CORREZE
HAUTE-SAONE
MEUSE
ALPES-DE-HAUTE-PROVENCE

12. Indiquer la liste des 10 communes les plus peuplées, avec leur région.

```
  SELECT v.nom,
       r.nom
  FROM ville AS v
       JOIN
       departement AS d ON v.numdep = d.numdep
       JOIN
       region AS r ON d.numreg = r.numreg
 ORDER BY v.pop2010 DESC
 LIMIT 10
```

nom nom :1
PARIS ILE-DE-FRANCE
MARSEILLE PROVENCE-ALPES-COTE D'AZUR
LYON RHONE-ALPES
TOULOUSE MIDI-PYRENEES
NICE PROVENCE-ALPES-COTE D'AZUR
NANTES PAYS DE LA LOIRE
STRASBOURG ALSACE
MONTPELLIER LANGUEDOC-ROUSSILLON
BORDEAUX AQUITAINE
LILLE NORD-PAS-DE-CALAIS

13. Quelle est la densité de population de chaque région (faire une liste alphabétique)?

```
SELECT region.nom,
       SUM(pop2010) / SUM(surface) AS densite
  FROM ville
       JOIN
       departement ON ville.numdep = departement.numdep
       JOIN
       region ON departement.numreg = region.numreg
 GROUP BY region.numreg
 ORDER BY region.nom ASC
```

```
nom densite
ALSACE 222.9036738243035
AQUITAINE 78.29576054764256
AUVERGNE 51.79689761499
BASSE-NORMANDIE 83.77208442395725
BOURGOGNE 51.99534797713632
```

Pour continuer à s'entrainer, écrire une requête en langage SQL :

14. qui donne la surface de Toulouse;

```
SELECT surface
FROM ville
WHERE nom = 'TOULOUSE'
```

```
118.3
```

15. qui donne le nom du département de la ville de Bordeaux;

```
SELECT d.nom
FROM ville v
     JOIN
     departement d ON v.numdep = d.numdep
WHERE v.nom = 'BORDEAUX'
```

```
GIRONDE
```

16. qui détermine la plus grande surface d'une ville de la métropole;

```
SELECT max(surface)
FROM ville
WHERE numdep < 96
```

```
758.93
```

17. qui donne les nom et surface des villes de plus grande et plus petite surfaces de la métropole;
(la technique utilisée pour cette question peut vous resservir par la suite)

```
SELECT nom, surface
FROM ville
WHERE surface = (SELECT max(surface) FROM ville WHERE numdep < 96)
      OR
      surface = (SELECT min(surface) FROM ville WHERE numdep < 96)
```

```
ARLES 758.93
CASTELMORON-D'ALBRET 0.04
```

18. qui donne le nom et le département (par son nom) de la ville de plus petite surface de la métropole;

```
SELECT v.nom, d.nom
FROM ville v
     JOIN
     departement d ON v.numdep = d.numdep
WHERE d.numdep < 96 AND
ORDER BY v.surface ASC
LIMIT 1
```

ou

```
SELECT v.nom, d.nom
FROM ville v
     JOIN
     departement d ON v.numdep = d.numdep
WHERE d.numdep < 96 AND
       v.surface = (SELECT min(surface) FROM ville WHERE numdep < 96)
```

CASTELMORON-D'ALBRET GIRONDE

19. qui classe par surface décroissante les villes Bordeaux, Lyon, Paris, Toulouse ;

```
SELECT nom
  FROM ville
  WHERE nom='BORDEAUX' OR nom='LYON' OR nom='PARIS' OR nom='TOULOUSE'
  ORDER BY surface DESC
```

ou (mais IN n'est pas censé être au programme)

```
SELECT nom
  FROM ville
  WHERE nom IN ('BORDEAUX', 'LYON', 'PARIS', 'TOULOUSE')
  ORDER BY surface DESC
```

TOULOUSE
PARIS
BORDEAUX
LYON

20. qui donne le numéro de département de la Guadeloupe ;

```
SELECT numdep
FROM departement
WHERE nom = 'GUADELOUPE'
```

971

21. qui détermine la surface de la Guadeloupe ;

```
SELECT sum(surface)
FROM ville v
     JOIN
     departement d ON v.numdep = d.numdep
GROUP BY d.nom
HAVING d.nom = 'GUADELOUPE'
```

1692.2

22. qui donne le nom de la ville la plus peuplée de Martinique (population 2010) ;

```
SELECT v.nom
FROM ville v
     JOIN
     departement d ON v.numdep = d.numdep
WHERE d.nom = 'MARTINIQUE'
ORDER BY v.pop2010 DESC
LIMIT 1
```

ou

```
SELECT v.nom
FROM ville v
     JOIN
     departement d ON v.numdep = d.numdep
WHERE d.nom = 'MARTINIQUE' AND
       pop2010 = (SELECT max(pop2010)
                  FROM ville v
                       JOIN
                       departement d ON v.numdep = d.numdep
                  WHERE d.nom = 'MARTINIQUE')
```

FORT-DE-FRANCE

23. (plus difficile) qui donne le nom de la deuxième ville la plus peuplée de Martinique (et uniquement celui-ci)

```
SELECT v.nom
FROM ville v
     JOIN
     departement d ON v.numdep = d.numdep
WHERE d.nom = 'MARTINIQUE'
ORDER BY v.pop2010 DESC
LIMIT 1
OFFSET 1
```

ou

```
SELECT v.nom
FROM ville v
     JOIN
     departement d ON v.numdep = d.numdep
WHERE d.nom = 'MARTINIQUE' AND
       pop2010 = (SELECT max(pop2010)
                        FROM ville v
                             JOIN
                             departement d ON v.numdep = d.numdep
                       WHERE d.nom = 'MARTINIQUE' AND
                             pop2010 < (SELECT max(pop2010)
                                               FROM ville v
                                                    JOIN
                                                    departement d ON v.numdep = d.numdep
                                              WHERE d.nom = 'MARTINIQUE'))
```

LAMENTIN

24. qui détermine la liste des département dont au moins le nom d'une commune comporte 'CLERMONT' ; (on pourra utiliser l'opérateur de comparaison de chaînes LIKE : LIKE "D%ONT" renverra TRUE pour toute chaîne de caractères commençant par D, finissant par ONT(% remplace n'importe quelle séquence de caractères : DUPONT,DUMONT..)

```
SELECT DISTINCT (d.nom)
  FROM departement d
       JOIN
       ville v ON d.numdep = v.numdep
 WHERE v.nom LIKE '%CLERMONT%'
```

AISNE
ARIEGE
AUDE
CHARENTE-MARITIME
DORDOGNE
HAUTE-GARONNE
GERS
HERAULT
ISERE
LANDES
LOT-ET-GARONNE
MEUSE
OISE
PUY-DE-DOME
SARTHE
HAUTE-SAVOIE

25. qui donne les villes de métropole de plus de 8000 habitants (population 2010) situées en totalité à plus de 1000 m d'altitude ;

```
SELECT nom
FROM ville
WHERE pop2010 >= 8000 AND zmin >= 1000 AND  numdep < 96
```

BRIANCON

26. qui donne le nom, la population et l'altitude maximale de la ville de code postal 74400 ;

```
SELECT nom, pop2010, zmax
FROM ville
WHERE codepostal = 74400
```

CHAMONIX-MONT-BLANC 8972 4807

27. qui détermine l'altitude minimale et l'altitude maximale de la Haute-Savoie ;

```
SELECT min(zmin), max(zmax)
FROM ville v
       JOIN
       departement d ON d.numdep = v.numdep
 WHERE d.nom = 'HAUTE-SAVOIE'
```

250 4807

28. qui détermine le nombre de villes du département 95 situées en totalité à plus de 100 m d'altitude ;

```
SELECT count( * )
FROM ville
WHERE numdep = 95 AND zmin >= 100
```

15

29. qui détermine le nombre de villes du département du Val-d'Oise situées en totalité à plus de 100 m d'altitude ;

```
SELECT count(*)
FROM ville v
     JOIN
     departement d ON v.numdep = d.numdep
WHERE d.nom = 'VAL-D''OISE' AND zmin >= 100
```

15

30. qui donne les noms des départements de métropole dont les villes ont en moyenne une altitude minimale supérieure ou égale à 500 m.

```
SELECT d.nom
FROM ville v
     JOIN
     departement d ON d.numdep = v.numdep
WHERE d.numdep < 96
GROUP BY d.numdep
HAVING avg(zmin) >= 500
```

CANTAL
ALPES-DE-HAUTE-PROVENCE
HAUTE-LOIRE
LOZERE
HAUTES-ALPES
HAUTE-SAVOIE

31. qui donne les noms des départements de métropole situés en totalité à plus de 200 m d'altitude.

```
SELECT d.nom
FROM ville v
     JOIN
     departement d ON d.numdep = v.numdep
WHERE d.numdep < 96
GROUP BY d.numdep
HAVING min(zmin) >= 200
```

```
ALPES-DE-HAUTE-PROVENCE
HAUTE-LOIRE
HAUTES-ALPES
PUY-DE-DOME
SAVOIE
HAUTE-SAVOIE
VOSGES
TERRITOIRE DE BELFORT
```

32. dont la réponse est le nombre de départements de métropole situés en totalité à plus de 200 m d'altitude ;

```sql
SELECT count( * )
  FROM (SELECT d.nom
              FROM ville v
                   JOIN
                   departement d ON d.numdep = v.numdep
             WHERE d.numdep <= 95
             GROUP BY d.numdep
            HAVING min(zmin) >= 200)
```

```
8
```

33. qui détermine l'altitude maximale de la région Corse ;

```sql
SELECT max(zmax)
FROM ville v
     JOIN
     departement d ON v.numdep = d.numdep
     JOIN
     region r ON d.numreg = r.numreg
WHERE r.nom = 'CORSE'
```

```
2706
```

34. (plus difficile) qui détermine l'altitude du point culminant de Corse ainsi que le nom de(s) commune(s) sur lequel il se trouve ;

```sql
SELECT v.zmax,v.znom
FROM ville v
     JOIN
     departement d ON v.numdep = d.numdep
     JOIN
     region r ON d.numreg = r.numreg
WHERE r.nom = 'CORSE' AND
       zmax = (SELECT max(zmax)
                      FROM ville v
                           JOIN
                           departement d ON v.numdep = d.numdep
                           JOIN
                           region r ON d.numreg = r.numreg
                     WHERE r.nom = 'CORSE')
```

```
2706 ASCO
2706 LOZZI
```

35. qui donne la liste des départements de métropole ayant au moins une commune située en totalité à plus de 1000 m d'altitude, avec leur nombre de telles communes, en les classant à partir du département en ayant le plus ;

```sql
SELECT d.nom,
       count( * ) AS nb1000
  FROM ville v
       JOIN
       departement d ON d.numdep = v.numdep
 WHERE d.numdep <= 95 AND
       v.zmin >= 1000
 GROUP BY d.numdep
 ORDER BY nb1000 DESC
```

```
HAUTES-ALPES 46
PYRENEES-ORIENTALES 38
LOZERE 35
ALPES-DE-HAUTE-PROVENCE 20
SAVOIE 20
HAUTE-GARONNE 12
HAUTE-LOIRE 11
CANTAL 9
HAUTES-PYRENEES 9
ISERE 8
ARDECHE 8
ALPES-MARITIMES 6
ARIEGE 5
HAUTE-SAVOIE 4
DOUBS 3
JURA 3
AUDE 2
DROME 1
PUY-DE-DOME 1
```

36. (plus difficile) qui donne la liste des départements de métropole ayant au moins une commune située en totalité à plus de 1000 m d'altitude, avec leur pourcentage de telles communes, en les classant à partir du département en ayant le plus grand pourcentage.

```sql
SELECT inter1.nomdep,
       100 * inter1.nb1000 / inter2.nbtotal AS pourcentage1000
  FROM ( SELECT d.nom AS nomdep,
                count( * ) AS nb1000
           FROM ville v
                JOIN
                departement d ON d.numdep = v.numdep
          WHERE d.numdep <= 95 AND
                v.zmin >= 1000
          GROUP BY d.numdep )  AS inter1
       JOIN
       ( SELECT d.nom,
                count( * ) AS nbtotal
           FROM ville v
                JOIN
                departement d ON d.numdep = v.numdep
          WHERE d.numdep <= 95
          GROUP BY d.numdep ) AS inter2
 GROUP BY inter1.nomdep
 ORDER BY pourcentage1000 DESC
```

```
HAUTES-ALPES 24
PYRENEES-ORIENTALES 20
LOZERE 18
ALPES-DE-HAUTE-PROVENCE 10
SAVOIE 10
HAUTE-GARONNE 6
HAUTE-LOIRE 5
ARDECHE 4
CANTAL 4
HAUTES-PYRENEES 4
ISERE 4
ALPES-MARITIMES 3
ARIEGE 2
HAUTE-SAVOIE 2
AUDE 1
DOUBS 1
JURA 1
DROME 0
PUY-DE-DOME 0
```

Utilisation du module sqlite3

```
import sqlite3 as sql

import matplotlib.pyplot as plt
import numpy as np

Base = sql.connect('C://Users/spellerin/PCe/TP-06/villes.sqlite')

def dessin(requete):
    result=Base.execute(requete)
    Long=[]
    Lat=[]
    L=[]
    for enreg in result:
        Long.append(enreg[0]*np.cos(2*np.pi/9))
        Lat.append(enreg[1])
        L.append(enreg[2])
    Long=np.array(Long)
    Lat=np.array(Lat)
    L=np.array(L)
    plt.figure()
    plt.axis('equal') #meme echelle en x et y
    plt.scatter(Long,Lat,c=np.log10(L+2))
    plt.show()

ex1 = "SELECT longitude,latitude, pop2010 FROM ville WHERE numdep < 96 ORDER BY pop2010 DESC LIMIT
    0, 5000"
ex2 = "SELECT AVG(longitude),AVG(latitude),MAX(zmax) FROM ville WHERE numdep < 96 GROUP BY numdep,
    numcanton"
ex3 = "SELECT AVG(longitude),AVG(latitude),100*(SUM(pop2010 - pop1999)+0.0) / SUM(pop1999) FROM
    ville WHERE numdep < 96 GROUP BY numdep,numcanton"
dessin(ex1)
dessin(ex2)
dessin(ex3)
```

On obtient les trois graphiques suivants