Corrigé

Bases de données

1. Interrogations usuelles de la base de données

Question 1.

a) Donner la liste des dix communes les plus vastes de France, classées par ordre décroissant de taille, en précisant pour chacune d'elles le nom du département où elles se situent.

```
SELECT c.nom, d.nom
  FROM communes c JOIN depts d ON c.dep_id = d.id
    ORDER BY c.superficie DESC LIMIT 10
```

b) Quelle est la commune de plus de 10 000 habitants la plus haute de France ? Dans quel département se situe-t-elle ?

```
SELECT c.nom, d.nom
  FROM communes c JOIN depts d ON c.dep_id = d.id
    WHERE c.population > 10
  ORDER BY c.altitude DESC LIMIT 1
```

Il s'agit de la ville de Briançon, dans les Hautes-Alpes.

c) Quel est le chef-lieu de département le moins peuplé de France ?

```
SELECT d.nom, c.nom, c.population
  FROM depts d JOIN communes c ON d.chef_lieu = c.id
  ORDER BY c.population ASC LIMIT 1
```

Il s'agit de Privas, chef-lieu de l'Ardèche, avec 8 500 habitants.

d) Quelle est la superficie moyenne (en hectares) des communes corses ? Rappelons que la Corse est constituée de deux départements de codes administratifs 2A et 2B.

```
SELECT AVG(c.superficie)
  FROM communes c JOIN depts d ON c.dep_id = d.id
    WHERE d.code = '2A' OR d.code = '2B'
```

La superficie moyenne des communes corses est de 2 432,8 hectares.

e) En considérant que la superficie d'un département est égal à la somme des superficies des communes qu'il abrite, calculer la superficie moyenne (en km^2) des départements métropolitains.

```
SELECT 0.01*AVG(s)
  FROM (SELECT SUM(c.superficie) s
          FROM communes c JOIN depts d ON c.dep_id = d.id
            GROUP BY d.id)
```

La superficie moyenne d'un département est de 5 711 km^2.

f) Quel est le département le plus vaste ?

```
SELECT d.nom, SUM(c.superficie)*0.01 s
  FROM communes c JOIN depts d ON c.dep_id = d.id
    GROUP BY d.id
  ORDER BY s DESC LIMIT 1
```

Il s'agit du département de la Gironde, avec 10 147 km^2.

g) Parmi les communes de plus de 1 000 habitants, quel est celle dont la densité est la plus faible ? (c'est-à-dire dont le nombre d'habitants par mètre carré est le plus petit). Dans quel département se trouve-t-elle ? On exprimera sa densité en nombre d'habitants par kilomètre carré.

```
SELECT c.nom, d.nom, c.population / c.superficie * 100000 s
  FROM communes c JOIN depts d ON c.dep_id = d.id
    WHERE c.population > 1
  ORDER BY s ASC LIMIT 1
```

Il s'agit de la commune de Laruns, dans les Pyrénées-Atlantiques (5,2 habitant par km^2).

h) Existe-t-il dans le Pas-de-Calais (code administratif 62) deux communes ayant même superficie et situées à la même altitude ?

```
SELECT c1.nom, c2.nom, c1.superficie , c1.altitude
  FROM communes c1 JOIN depts d ON c1.dep_id = d.id , communes c2
    WHERE d.code = '62' AND c1.dep_id = c2.dep_id
                        AND c1.superficie = c2.superficie
                        AND c1.altitude = c2.altitude
                        AND c1.nom < c2.nom
```

On trouve trois couples de communes : Bois-Bernard et Marant, Fresnes-les-Montauban et Hamblain-les-prés, Gouy-Servins et Huclier.

i) Enfin, déterminer le code postal attribué au nombre le plus élevé de communes différentes (on en précisera le nombre).

```
SELECT postal , COUNT(*) c
  FROM communes
    GROUP BY postal
  ORDER BY c DESC LIMIT 1
```

Le code postal 51 300 est partagé par 46 communes.

2. Interaction avec la base de données

Question 2. On définit les fonctions :

```
def chef_lieu(s):
    cur.execute("SELECT c.nom FROM communes c JOIN depts d ON c.id = d.chef_lieu
                 WHERE d.code='{}'".format(s))
    result = cur.fetchall()[0][0]
    return result
```

```
def prefecture(s):
    cur.execute("SELECT d.code FROM depts d JOIN communes c ON d.id = c.dep_id
                 WHERE c.nom='{}'".format(s))
    result = [r[0] for r in cur.fetchall()]
    return [chef_lieu(x) for x in result]
```

Le nom de Saint-Colombe est partagé par 14 communes en France ; les préfectures correspondantes sont :

```
>>> prefecture('SAINTE-COLOMBE')
'GAP', 'ANGOULEME', 'LA ROCHELLE', 'DIJON', 'BESANCON', 'BORDEAUX', 'RENNES',
'MONT-DE-MARSAN', 'CAHORS', 'SAINT-LO', 'LYON', 'ROUEN', 'MELUN', 'AUXERRE'
```

Question 3. On définit la fonction :

```
def code_postal_inverse(n):
    cur.execute("SELECT c.nom, d.nom FROM communes c JOIN depts d ON c.dep_id = d.id
                 WHERE c.postal LIKE '%{}%'".format(str(n)))
    result = cur.fetchall()
    if len(result) == 0:
        return None
    return result[0][1], [r[0] for r in result]
```

Le code postal 42440 est partagé par huit communes de la Loire :

```
>>> code_postal_inverse(42440)
('LOIRE', ['CERVIERES', 'LA CHAMBA', 'LA CHAMBONIE', 'NOIRETABLE', 'SAINT-JEAN-LA-VETRE',
'SAINT-JULIEN-LA-VETRE', 'SAINT-PRIEST-LA-VETRE', 'LES SALLES'])
```

Le code postal 08320 est commun à quatre communes des Ardennes :

```
>>> code_postal_inverse('08320')
('ARDENNES', ['AUBRIVES', 'HIERGES', 'VIREUX-MOLHAIN', 'VIREUX-WALLERAND'])
```

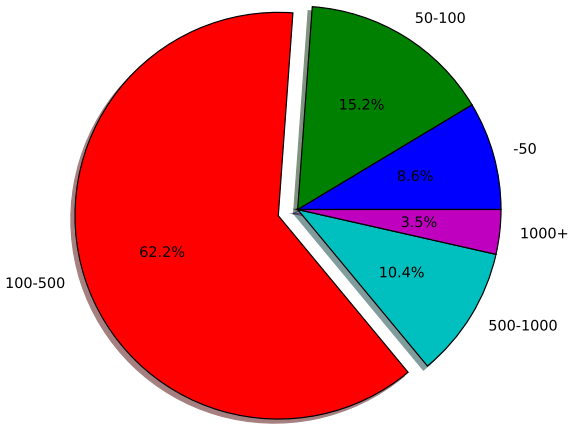
Deux erreurs sont à éviter : les codes postaux débutant par un 0 doivent être introduits sous la forme d'une chaîne de caractères, et il faut tenir compte que certaines grandes villes, Nantes par exemple, peuvent posséder plusieurs codes postaux :

```
SELECT postal FROM communes WHERE nom = 'NANTES'
'44000/44100/44200/44300'
```

Question 4. On réalise le script ci-dessous :

```
data = []
cur.execute("SELECT count(*) FROM communes WHERE altitude < 50")
data.append(cur.fetchall()[0][0])
cur.execute("SELECT count(*) FROM communes WHERE altitude BETWEEN 50 and 100")
data.append(cur.fetchall()[0][0])
cur.execute("SELECT count(*) FROM communes WHERE altitude BETWEEN 100 and 500")
data.append(cur.fetchall()[0][0])
cur.execute("SELECT count(*) FROM communes WHERE altitude BETWEEN 500 and 1000")
data.append(cur.fetchall()[0][0])
cur.execute("SELECT count(*) FROM communes WHERE altitude > 1000")
data.append(cur.fetchall()[0][0])

label = ['-50', '50-100', '100-500', '500-1000', '1000+']
plt.pie(data, explode=(0, 0, 0.1, 0, 0), labels=label, autopct='%1.1f%%', shadow=True)
plt.axis('equal')
plt.show()
```



3. Tracé de contour

Question 5. On définit la fonction :

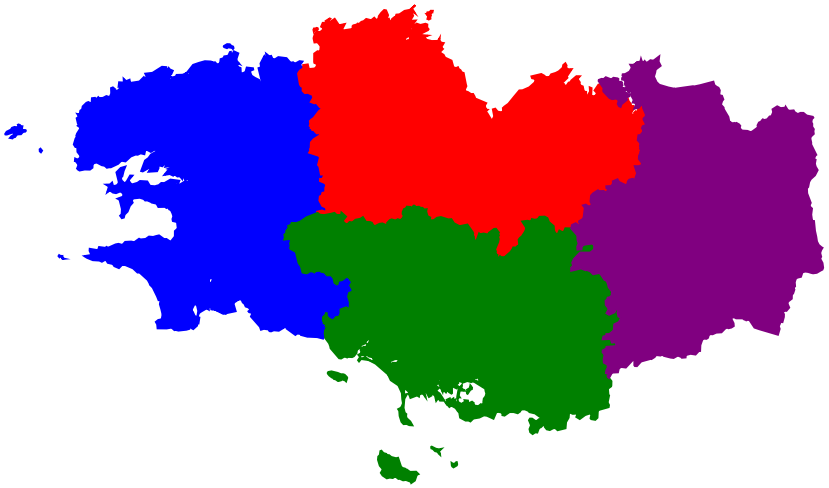
```
def polygone(c, i):
    cur.execute("SELECT p.x, p.y FROM pointsdep p JOIN depts d ON p.iddept = d.id
                 WHERE d.code='{}' and p.poly={} ORDER BY p.ordre ASC".format(c, i))
    result = cur.fetchall()
    return [r[0] for r in result], [r[1] for r in result]
```

Il faut ensuite, pour un département donné, déterminer les numéros des polygones qui lui sont associés puis, pour chacun d'eux, effectuer le tracé dans la couleur choisie.

```
def dessine_departement(c, couleur='blue'):
    cur.execute("SELECT DISTINCT poly FROM pointsdep p JOIN depts d ON p.iddept = d.id
                 WHERE d.code='{}'".format(c))
    poly = [r[0] for r in cur.fetchall()]
    for i in poly:
        x, y = polygone(c, i)
        plt.fill(x, y, color=couleur)
```

Le script suivant permet de dessiner les départements bretons :

```
dessine_departement('29')
dessine_departement('22', couleur='red')
dessine_departement('56', couleur='green')
dessine_departement('35', couleur='purple')
plt.axes().set_aspect(1.5)
plt.show()
```



Question 6. Pour obtenir la carte qui figure page suivante, j'ai utilisé le script :

```
cur.execute("SELECT d.code, SUM(c.population) FROM depts d JOIN communes c ON d.id = c.dep_id
             GROUP BY d.id")
result = cur.fetchall()
mini = min([r[1] for r in result])
maxi = max([r[1] for r in result])
rouge = np.array([1, 0, 0], dtype=float)
jaune = np.array([1, 1, 0], dtype=float)
for r in result:
    t = (r[1]-mini)/(maxi-mini)
    c = (1-t)*jaune + t*rouge
    dessine_departement(r[0], couleur=c)
plt.axes().set_aspect(1.5)
plt.show()
```

On obtient ainsi un dégradé allant du jaune (pour les départements les moins peuplés) au rouge (pour les départements les plus peuplés).