Question 1 :
```
SELECT idpatient
FROM MEDICAL
WHERE etat = "hernie discale"
```

Question 2 :
```
SELECT nom, prenom
FROM MEDICAL
JOIN MEDICAL ON PATIENT.id = MEDICAL.idpatient
WHERE etat = "spondylolisthesis"
```

Question 3 :
```
SELECT etat, COUNT(DISTINCT idpatient)
FROM MEDICAL
GROUP BY etat
```

Question 4 : Il est beaucoup plus rapide de parcourir avec un tableau avec numpy. Et comme le montre l'annexe il existe plein de fonction implémenté pour manier les tableaux de grande taille.
Ainsi Numpy est plus efficace pour traiter les tableaux de grande taille que les liste python.

Question 5 : Le tableau data serait une matrice de $N = 100\,000$ ligne et $n = 6$ colonnes. Chaque case contient une donné codé sur 32 bit soit 4 octet par case. Ainsi le tableau data utilisera $N \times n \times 4$ octet soit $2\,400\,000$ octet $= 2,4$ Mo.

Le tableau etat serait un vecteur à $N = 100\,000$ ligne chaque ligne contient une donné codé sur 8 bit soit 1 octet de mémoire utilisé par ligne soit 1 Mo utilisé en tout.

Ainsi il faut 2,5 Mo de mémoire pour pouvoir stocker le tableau data et etat.

Question 6 : (1)

```
def separationParGroupe(data, etat):
    e = [[], [], []]
    for i in range(len(etat)):
        e[etat[i]].append(data[:3])
    return e
```

Question 9 : $x_{normj} = \frac{x - x_{min}}{x_{max} - x_{min}}$ (1)

Question 10 : (2)

```
def min_max(X):
    min = ...
    max = X[0]
    for i in X:
        if i <= min:
            min = i
        if i >= max
            max = i
    return min, max
```

Question 11 : (4)

```
def distance(z, data):
    e = []
    for i in range(len(data)):
        s = 0
        for j in range(len(z)):
            s = (data[i][j] - z[j])^2
        dist = sqrt(s)
        e = e.append(dist)
    return e
```

Si on doit utiliser la technique de normalisation:

Question 11 :
```
def distance(z,data):
    l=[]
    for i in range(len z)
        l=l.append(min-max(data[:,i]+[z[i]])
    d=[]
    for i in range(len(data))
        s=0
        for j in range(len(z)):
            s=(((data[i][j]-l[j][0])/
(l[j][1]-l[j][0]))-((z[j]-l[j][0])/(l[j][1]-l[j][0]
))^2.
        dist=sqrt(s)
        d=d.append(dist)
    return d.
```

Question 12 : La partie 1 crée la liste T décrit dans le sujet et la trie.

La partie 2 crée une liste select de longueur nb où il place dans chaque case correspondant au nombre d'un état le nombre de voisin proche dans cette état. Par exemple la donné select[0] est le nombre de voisin proche qui ont un état normal dans notre exemple.

La partie 3 cherche le maximum de select et son indice qui correspond à l'état qui possèdent le plus de proche voisin. C'est ce qui renvoie.

Question 13:
La diagonale indique pour chaque etat
qui correspond au numéro des colonnes
Q. nombre de fois où cette état a été
l'état avec le plus de
proche voisin ... ... de datatest.
et dadta.

Question 14: La courbe obtenue nous
donne le pourcentage de réussite de
l'algorithme KNN. Cela nous indique
quand utilisant l'algorithme KNN on
obtient 70% et 75% de chance de bien
catégorisé un nouveau patient dans le
bon état.