1. SELECT idpatient
FROM MEDICAL
WHERE etat : 'hernie discale';

2. SELECT nom, prenom
FROM PATIENT
LEFT JOIN MEDICAL ON PATIENT.id = MEDICAL.idpatient
WHERE etat : 'spondylolisthesis';

3. SELECT etat, count (idpatient)
FROM MEDICAL
GROUP BY etat;

4. Cela permet de realiser des operations plus rapides même sur des tableau de grandes dimensions.

5. Pour le tableau, la quantité de memoire est
N × n × 32 : 100000 × 6 × 32 : 19 200 000 bits
: 1 900 000 octets : 1.4 Mo
Pour le vecteur d'état de santé : N × 1 : 100 000 octets
: 0.1 Mo
quantité memoire total : 1.5 Mo

6. def separationParGroupe (data, etat):
GPNormal = [] L = [ ]
GPHernie = []
GPSpondy = []
for i in range (len (etat)):
if etat [i] : 0 :
GPNormal.append (array (data [i, :]))
elif etat [i] : 1
GPHernie.append (array (data [i, :]))

```
        elif etat[i] == 2 :
            GPSpondy.append(array(data[i,:]))
    L.append(GPNormaP)
    L.append(GPKernic)
    L.append(GPSpondy)
    return L
```

9. $x_{norm_j} = \frac{x_j - \min(x)}{\max(x) - \min(x)}$

10.
```
def min_max(x):
    min = x[0]
    max = x[0]
    for i in range(len(x)):
        if x[i] > max:
            max = x[i]
        elif x[i] < min:
            min = x[i]
    return min, max
```

11.
```
def distance(z, data):
    L = []
    d = 0
    for i in range len(data):
        for j in range len(data[i,:]):
            d = d + (data[i,j] - z[j]) **2
        L.append(sqrt(d))
        d = 0
    return L
```

12. partie 1 : tri par ordre croissant distance entre le n-uplet à classer et un n-uplet connu
T, Pist, select sont des listes
ind est une variable inutile