1) SELECT idpatient FROM MEDICAL WHERE etat = hernie discale

2) SELECT nom, prenom FROM
(SELECT * FROM PATIENT JOIN MEDICAL ON
PATIENT.id = MEDICAL.idpatient WHERE etat =
spondylolisthesis)

3) SELECT etat COUNT(id_patient) FROM MEDICAL
GROUP BY etat

4) Numpy permet une meilleur lisibilité des tableaux et une manipulation plus facile ce qui permet les opérations.

5) il y a 6 x 100 000 case dans le tableau avec dans chacun un entier codé sur 32 bits soit 32 x 6 x 100000 bits au total pour le tableau. Le vecteur contient 100 000 valeurs codé sur 8 bit donc 100 000 x 8 bit pour le vecteur
donc 6 x 100 000 x 32 x 100 000 x 8 = 2 0 000 000
il faut donc 20000 000 bits pour tout stocké soit 2,5 Mo
car 8 bit = 1 octet et 1 Mo = 1 000 000 octet

6)

```
def separationParGroupe(data, etat):
    liste1 = []
    liste2 = []
    liste3 = []
    for i in range(len(data)):
        if etat[i]==0 :
```

15) def moyenne (x):

```
            liste1.append data[i]
        else:
            if stat[i]==2:
                liste2.appad data[i]
            else:
                liste3.appad data[i]
    return([liste1, liste2, liste3])
```

7) ARGS1 : n, n , i*n+j+1

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

si $x_j = x_{max}$ $x_{normj} = 1$

si $x_j = x_{min}$ $x_{normj} = 0$ la chaîne est donc logique.

10) def min_max(x):

```
    n = length(x)
    min = X[0]
    max = X[0]
    for i in range n:
        if min > X[i]:
            min = X[i]
        if max < X[i]:
            max = X[i]

    return (min, max)
```

11) def distance(z, data)

1a) La partie 1 permet de trier les distances entre chaque n-uplet tout en gardant en mémoire la plus marginale.

La partie 2 crée une liste relative de tel sorte qui permet d'associer à chaque n-uplet son rang.

La partie 3 permet de trouver la position du n-uplet dans sa distance et la plus éloigné

T est la liste qui associe la distance du n-uplet à sa position dist la liste des distance de tout les n-uplet
relat le rang (par ordre croissant) des distances des n-uplet
ind l'indice du n-uplet qui a la distance la plus grande

2)

1a) l'algorithme est plutôt efficace mais pas totalement fiable.

15) def moyenne (x):

```
    n = length[x]
    cont = 0
    for i in range n:
        cont += x[i]
    return (cont/n)
```

```
def variance (x) =
    n = length[n]
    R = moyenne (x)
    ind = 0
    for i in range n:
        cont += (x[i] - R*R)
    return (ind / n)
```

16) def synthèse (data, etat):