

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 = 'spondylopathie';
```
3.

```
SELECT etat, count(idpatient)
FROM MEDICAL
GROUP BY etat;
```
4. Cela permet de réaliser des opérations pratiques même sur des tableaux de grandes dimensions.

5. Pour le tableau, la quantité de mémoire est

$$N \times n \times 32 = 100000 \times 6 \times 32 = 19\,200\,000 \text{ bits}$$

$$= 2\,400\,000 \text{ octets} = 2.4 \text{ Mo}$$

Pour le vecteur d'état de santé : $N \times 1 = 100\,000 \text{ octets}$
 $= 0.1 \text{ Mo}$

quantité mémoire totale : 2.5 Mo

6. `def separationParGroupe (data, etat):`

`GPNormap = [] L = []`

`GPHernie = []`

`GPSpondy = []`

`for i in range (len(etat)):`

`if etat[i] == 0:`

`GPNormap.append (array (data[i,:]))`

`elif etat[i] == 1`

`GPHernie.append (array (data[i,:]))`

elif elat [i]: 2:

GPSpondy.append(array(data[i,:]))

L.append(GPNormalP)

L.append(GPKernic)

L.append(GPSpondy)

return L

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

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(x, data):`
 `L = []`
 `x ← d = 0`
 `for i in range(len(data)):`
 `for j in range(len(data[i])):`
 `d = d + (data[i,j] - x[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 cellule