PASCAL Tom
MPSI 1

DS d'Informatique

Q1) :

```
SELECT idpatient
FROM MEDICAL
WHERE etat = 'hernie discale'
```

✓

Q2)

```
SELECT nom, prenom
FROM PATIENT
WHERE id IN (SELECT idpatient
             FROM MEDICAL
             WHERE etat = 'spondylolisthésis')
```

La Jointure.

Q3)

```
SELECT COUNT(*)
FROM MEDICAL
GROUP BY etat
```

/

Q4) Numpy est plus rapide.

Q5)

Q6)

```
def separationParGroupe(data, etat):
    tab = [[],[],[]]
    for i in range(len(data)):
        if etat[i] = 0:
            tab[0].append(data[i])
        if etat[i] = 1:
            tab[1].append(data[i])
        if etat[i] = 2
            tab[2].append(data[i])
    return tab.
```

Q7) ARGS1 : n , n , n(i-1) + j
ARGS2 : groupes[k][:], groupes[k][j], marker = mark[k]
ARGS3 : data[:]
TEST : i != j

Q8) Les diagrammes diagonaux permettent de mesurer le taux d'apparition des symptômes.
Les diagrammes hors diagonale permettent de chercher un lien entre les symptômes selon la maladie observée.

Q9) $x_{norm_j} = \frac{x_j}{max(X) - min(X)}$

Q10)

```
def min_max(X):
    min = X[0]
    max = X[0]
    for i in range(len(X)):
        if X[i] < min:
```

```
        min = X[i]
    if X[i] > masc :
        masc = X[i]
return masc, min
```

Q11)

```
def distance(xy, data):
    list = [ ]
    for i in range(len(xy)):
        x = 0
```

Q12) La partie 1 crée une liste T contenant la distance et l'indice correspondant à chaque patient, triée par distances
La partie 2 crée une liste select qui donne la distance la plus courte pour chaque symptôme.
La partie 3 cherche le maximum de select pour donner l'état correspondant, stocké dans ind.

Q14) L'algorithme ne réussit jamais à plus de 75%, il n'est pas très efficace.

Q15)

```
def moyenne(x):
    n = len(x)
    a = 0
    for i in range(n):
        a = a + x[i]
    return a/n
```

```
def variance(x):
    m = moyenne(x)
    n = len(x)
    a = 0
    for i in range(n):
        a = a + (x[i] - m)**2
    return a/n
```

Q16)

```
def synthese(data, etat):
    l1 = len(data)
    l2 = len(etat)
    tab = []
    for i in range(l2):
        tab.append([[]]
    for k in range(l1):
        tab[etat[k]].append([moyenne(data[k]), variance(data[k])]
    return tab
```