PEYRON Romane MPSI 1

Q1. SELECT idpatient FROM MEDICAL WHERE etat = hernie discale ; ✓

Q2. SELECT nom, prenom FROM PATIENT WHERE idpatient = (SELECT idpatient FROM MEDICAL WHERE etat = spondylolisthésis); → Jointure

Q3. SELECT etat, COUNT(*) FROM MEDICAL WHERE
etat = ~~scoliose OR etat = spondylolisthésis~~ OR etat = hernie discale ;
→ Group By

Q4. Cette bibliothèque contient de nombreux outils mathématiques pour manipuler les tableaux numpy, de plus elle permet d'optimiser la mémoire mise en jeu.

Q5. Quantité de mémoire = $32 \times 6 \times N$ bits
$= \frac{32 \times 6 \times N}{8000000}$ Mo
$= 2,4\ M_o$ ✓

Q6.
```
def separationParGroupe(data, etat):
    etat_0=np.array()
    etat_1=np.array()
    etat_2=np.array()
    for i in range(N):
        if etat[i]=0:
            etat_0.append(data[i,j] for j in range(6))
        else if etat[i]=1:
            etat_1.append(data[i,j] for j in range(6))
            else if etat[i]=2:
                etat_2.append(data[i,j] for j in range(6))
    return [etat_0, etat_1, etat_2]
```
⚠ import numpy as np

Paladrain

Q8: Les diagrammes en diagonale présentent le nb de patients qui, dans un certain état ont tel ou tel caractéristique ou un attribut : par exemple on repère l'incidence de bassin la plus répandue chez les patients dans l'état : (0, 1 ou 2).
Chaque diagramme correspond à un attribut différent.

Les diagrammes en le côté permettent de repérer les différences de profil pour les états 0, 1 ou 2 ; si on voit un gros nuage de points, bien distinct d'un nuage de 'x' (hernie discale), on peut imaginer que la hernie discale influe (ou est influencée) sur (resp. par) ces paramètres.

Q7 TEST correspond à : != j

Q9. $x_{normj} = \frac{|x_j - min(X)|}{|min(X) - max(X)|}$

Q10. Complexité linéaire : on balaye le vecteur X :

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

Q11:
```
def distance(rj, data):
    Somme = 0
    L = len(data[1,:])
    Liste = []
    for i in range(N):
        for j in range L :
            Somme += abs(rj[j] - data[i,j])
        Liste.append(Somme)
    return Liste
```

Q12: Partie 1: T contient toutes les distances à rj, et l'identifiant du patient (numéro de la ligne dans data, et len(dist) ici) tri(T) permet juste de trier la liste T (triée par ordre croissant de distance à rj)

Partie 2: sélectionne les K plus proches voisins de rj (les K premières listes de 2 éléments de la liste triée T) en replaçant rj comme le plus proche de lui-même

select contient les "identifiants" des K patients les plus proches de rj, triés des plus proches aux moins proches