BESSON Matthys MPSI2.

DS09 informatique.

Question 1.

la requête SQL est la suivante :

```
SELECT idpatient from MEDICAL where
  etat = 'hernie discale' ;
```

✓

Question 2.

la requête SQL est :

```
SELECT nom, prénom from PATIENT
join MEDICAL on PATIENT.id = MEDICAL.idpatient
  where etat = 'spondylolisthésis' ;
```

✓

Question 3.

Different

```
SELECT etat, count(idpatient) from MEDICAL
  group by etat ;
```

Question 4.

La bibliothèque de calcul numérique Numpy est plus rapide que les autres bibliothèques quand les tableaux sont de grande taille.

Question 5.

on a $N = 100\,000$ lignes
$n = 6$ colonnes.
or d'après l'énoncé, les réels sont codés sur 32 bits
or 1 octet = 8 bits
donc 32 bits = 4 octets.

donc $N_{data} = N \times n \times 4$
$= 100\,000 \times 6 \times 4$
$= 2\,400\,000$ octets
$= 2,4$ Mo.

pour N_{etat} : 1 octet = 8 bits, les réels sont codés sur 8 bits.

$N_{etat} = N \times 1 \times \frac{1}{2}$ ← 1 seule colonne
← 1 octet
$= 100\,000$ octets
$= 0,1$ Mo.

D'où N total = N data + N état
= 2,4 + 0,1
= 2,5 Mo.

Il faut 2,5 Mo pour N = 100 000. ✓

Question 6. La fonction est la suivante :

```
def separationParGroupe(data, etat):
    sous_tableau = [[],[],[]]
    for j in range(len(data)):
        sous_tableau[etat[j]].append(data[j])
    Return sous_tableau
```

✓

etat[j] = 0, 1 ou 2

Avec cela va mettre dans la 1ère, 2e ou 3e liste.

Question 7.

ARG S1 = (n, n, (j+1)(i+1))

taille du tableau — j^e trié avec le décalage

ARG S2 = (groupe[i], groupe[j], marches = mach[k])

Il manque k

ARG S3 = data[i]

[:,i]

TEST = i ≠ j , il faut que i ≠ j pour enlever la requête (diagonale)

Question 8 :

Les diagrammes de la diagonale peuvent nous faire voir si les valeurs sont bien autour d'une valeur moyenne (fréquence de l'attribut).
Les diagrammes hors diagonale peuvent nous faire voir si on peut faire une régression et que toutes les valeurs correspondent ou s'il y a un problème.

Et corrélation

Question 9 :

D'après la méthode de normalisation :

$$x_{norm\,j} = \frac{x_j - \text{donn}(X)}{\max(X) - \min(X)}$$

car $\max_{norm}(X) = 1$
et $\min_{norm}(X) = 0$.

Donc on a bien normalisé.

Question 10 :

La fonction est la suivante :

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

Question 11.

La fonction est la suivante :

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

V

V

car la distance est $d = \sqrt{(b-c)^2}$

Question 12.

ligne 3 : crée une liste T.
ligne 4 : attribue la distance de la fonction d'avant dans dist.
ligne 5 : pour toutes les longueurs de dist
ligne 6 : on ajoute dans la liste T la liste la distance i associée à son numéro.
ligne 7 : trie par place la liste T.

Partie 2 :

Partie 3 :

Question 13.

Question 15.

```
def moyenne(x):
    a = 0
    for i in x:
        a = a + i
    Return a / len(x)
```

✓

```
def variance(x):
    b = 0
    for i in x:
        b = b + (i - (moyenne(x)))**2
    return b / len(x)
```

✓

Question 16.

```
def synthese(data, etat):
    liste = [[] for i in range len(data)]
    for j in range len(data):
        liste[j] = moyenne(data[j]), variance(data[j])
    return liste
```

Il faut une double boucle.

Question 17 :

def fonction gaussienne(a, mu, x) :

Question 18 :