

DSO9 informatique.

Question 1.

la requête SQL est la suivante :

```
SELECT idpatient from MEDICAL where  
etat = 'herpès à viscéral';
```



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.

on d'après l'énoncé, les réels sont codés sur 32 bits

on  $1 \text{ octet} = 8 \text{ bits}$

donc  $32 \text{ bits} = 4 \text{ octets}$ .

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



pour  $N_{\text{vect}}$ :  $1 \text{ octet} = 8 \text{ bits}$ , les réels sont codés sur 8 bits.

$$\begin{aligned}N_{\text{vect}} &= N \times 1 \times 1 \xleftarrow{1 \text{ seule colonne}} 1 \text{ octet} \\ &= 100\,000 \text{ octets} \\ &= 0,1 \text{ Mo.}\end{aligned}$$





$$\begin{aligned}
 \text{Au } N_{\text{total}} &= N_{\text{data}} + N_{\text{etat}} \\
 &= 2,4 + 0,2 \\
 &= 2,5 \text{ Mo.}
 \end{aligned}$$

Il faut 2,5 Mo pour  $N = 100\,000$ . ✓

Question 6. la fonction est la suivante :

```

def separation Par Groupe (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  
 car cela va mettre  
 dans la 1<sup>ère</sup>, 2<sup>e</sup> ou 3<sup>e</sup> liste.

✓

Question 7.

ARG51 = (n, n, ~~(j+2)(i+2)~~)

taille du tableau      je crée avec le découpage

ARG52 = (groupe[i], groupe[j], marker - mark[k])

Il manque k

ARG53 = data[i]

[i,i]

TEST = i != j, il faut que  $i \neq j$  pour  
 continuer 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 à un même problème.

Et corrélation

### Question 9.

D'après la méthode de normalisation :

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

$$\text{car } \max_{norm}(X) = 1$$

$$\text{et } \min_{norm}(X) = 0.$$

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, lista):

    a = [ ] for i in range(len z)

        for j in range(len z):

            a[i] = np.sqrt((z[i] - lista[j])<sup>2</sup>)

    return a

Car la distance de  $a = \sqrt{(b - c)^2}$

### Question 12.

ligne 3: crée une liste T

ligne 4: attribue la distance à 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: on place la liste T

~~ligne~~ Partie 2:

Partie 3:

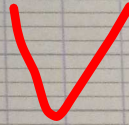


Question 13.



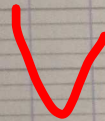
### Question 15.

```
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, moy, x):

Question 18.