Ainpho

Vendredi 24 mars

Est-ce cruel (SQL)

Ainpho

Vous gérez une entreprise commerciale qui possède plusieurs *magasins* et *vend* différents *produits*. Tous les *magasins* n'ont pas tous les *produits*, mais on ne s'occupe pas ici de gestion de stocks. On a cependant dans la base de données une table qui indique quel *magasin vend* quel *produit (voir plus bas)*.
Vous disposez d'un fichier *clients* [1]. Ce fichier est formé de deux tables dans la base de données : l'une vous indique quels *produits aiment* les différents *clients*, et l'autre quels *magasins* ils fréquentent.
Une base de données contient trois tables très simples *(deux champs chacune)* :

Table	vend		frequente		aime	
Structure	Magasin	Produit	Client	Magasin	Client	Produit
Exemple	Rivoli	Boissons	Jean-Luc	Belleville	François	Costumes
	Rivoli	Livres	François	Elysées	Nathalie	Livres
	Elysées	Costumes	Nathalie	Belleville	Jean-Luc	Boissons
	Montparnasse	Boissons	Jean-Luc	Rivoli	Julie	Livres

Comme les champs de plusieurs tables portent logiquement les mêmes noms, quand vous ferez une jointure comme `ON vend JOIN frequente`, si vous utilisez un champ `Magasin` il faudra préciser si c'est `vend.Magasin` ou `frequente.Magasin` *(il se peut d'ailleurs que vous imposiez une sélection* `frequente.Magasin = vend.Magasin`*)*.
On rappelle qu'on peut joindre une table à elle même. Il faut alors utiliser un `alias` [2] pour chacune, afin de les distinguer.
Par exemple pour trouver les *clients* qui *fréquentent* les mêmes *magasins* que Serge :

```
SELECT F1.Client
FROM frequente as F1 JOIN frequente as F2
WHERE F1.Magasin = F2.Magasin AND F1.Client = 'Serge'
```

Règle d'écriture des requêtes *(même pour vous sur votre copie)* :
- cadre autour de la requête
- majuscules pour les mots de SQL tels que `SELECT`, `COUNT`, `ORDER BY`, `JOIN`
- passage à la ligne pour chaque composante de la requête `SELECT/FROM/WHERE/IN`
- indentation pour les sous-requêtes comme dans l'exemple ci dessous

```
SELECT Client
FROM frequenter
WHERE Client NOT IN
........(SELECT Client
.........FROM frequenter
.........WHERE Magasin NOT IN
..................(SELECT Magasin
..................FROM vend JOIN aime
..................WHERE aime.Client = frequenter.Client AND vend.Produit = aime.Produit))
```

Je ne vous demande pas ce que fait la requête encadrée ci dessus, c'est juste un exemple de mise en forme.

[1] à quoi vous croyez qu'elles servent les cartes de fidélité des magasins ? à vous faire plaisir, ou à pouvoir avoir un fichier qui informe sur les goûts de chacun pour savoir comment optimiser les mises en rayon ? (faut il vendre des surgelés dans le magasin Rivoli, faut il mettre le rayon vins à côté du rayon légumes...)

[2] comme pour `Python` quand avec Solène vous `importez numpy as np`

	points	question
A	1	Ecrivez une requête qui détermine les *magasins* que *fréquente* MARC.
B	2	Ecrivez une requête qui compte combien de *magasins vendent* des *livres*.
C	1	Ecrivez une requête qui donne la liste des *produits* du *magasin* BELLEVILLE, triée par ordre *alphabétique*.

D	2	Que fait la requête suivante :
		`SELECT Client`
		`FROM frequente`
		`WHERE Magasin IN`
		`....(SELECT Magasin`
		`.....FROM vend`
		`.....WHERE Produit IN ('Livres', 'Jouets'))`

E	3	Ecrivez une requête pour savoir tous les *produits* qu'on trouve dans le *magasin* RIVOLI. Déduisez une requête *(avec sous-requête si nécessaire)* pour savoir tout ce qu'on trouve à la fois dans le magasin RIVOLI et le magasin BELLEVILLE.

F	2	Que va chercher cette requête `SQL` :
		`SELECT frequente.Client`
		`FROM vend as V JOIN frequente as F JOIN aime as A`
		`WHERE A.Produit=V.Produit AND V.Magasin=F.Magasin AND A.Client=F.Client`

G	2	Que font
		`SELECT Produit`
		`FROM aime`
		`WHERE Client = 'Martine'`
		puis
		`SELECT magasin`
		`FROM vend`
		`WHERE produit IN`
		`....(SELECT produit`
		`.....FROM aime`
		`.....WHERE client = 'Martine')`

H	1	Inventez une question et donnez sa formulation `SQL`.
I	3	Ecrivez une requête qui détermine les *clients* qui vont dans tous les *magasins*.
J	2	Déterminez les *produits* que CLAUDE *aime* et qui sont *vendus* dans le magasin BELLEVILLE.

K	3	`SELECT Client`
		`FROM vend as V JOIN aime AS A JOIN frequente AS F`
		`WHERE V.Produit=A.Produit AND A.Client=F.Client AND F.Magasin=V.Magasin`
		`GROUP BY Client`
		`HAVING COUNT(Magasin)>2`

Finalement, il y a avanr ce tableau une requête qui servait d'exemple pour la mise en forme... je vous demande maintenant à quelle question elle répond. 3 pt.

MYSTÈRE — Ainpho

Voici un script `Python`. Que fait il ?[3] 4 pt.

[3] évidemment une réponse du type "*il prend un nombre qu'il appelle S et lui donne la valeur 0 puis un nombre qu'il appelle C... si n modulo k est égal à 0 il donne à Test la valeur 0...*" ne correspond pas à ce qu'on attend de vous, ce serait comme si à la question de Camille "*qui est Ulrich*" vous répondiez "*206 os, 32 dents, 21 kilos de muscles, 8 kilos de graisses, 5 litres de sang, dix sept kilos de viscères...*"

```
def mystere(A) :
....S = 0
....C = 0
....n = 1
....while S < A :
........n += 1
........Test = 1
........for k in range(2, n) :
............if n%k == 0 :
................Test = 0
........S += Test*n**(-1)
........C += Test
.....return(n, C)
```

Quelle sera la réponse à `mystere(1)` ? [1 pt.] Oserez vous lancer `mystere(5)`, pourquoi ? [2 pt.]

ALGÈBRE — Ainpho

Pour tout matrice A réelle de taille 3, on définit

- son polynôme caractéristique $\det(A - X.I_3)$ *(vérifiez que c'est un polynôme de degré 3)* [1 pt.]
- son spectre : ensemble des racines du polynôme caractéristique *(vérifiez que son cardinal est 1, 2 ou 3 et donnez un exemple où il vaut 3)* [1 pt.]
- ses sous-espaces propres $E_\lambda = \{U \in \mathbb{R}^3 \mid A.U = \lambda.U\}$ avec λ dans le spectre *(vérifiez que ce sont des espaces vectoriels)* [1 pt.]

Vérifiez que justement la dimension de E_λ est non nulle si et seulement si λ est bien une valeur propre de A *(=élément du spectre)*. [2 pt.]

Ecrivez un script `Python` qui prend en entrée la matrice A *(liste de listes de flottants)* et

- donne les quatre coefficients du polynôme caractéristique [1 pt.]
- trace le polynôme caractéristique entre -10 et 10 [2 pt.]
- indique le nombre de racines réelles [4 pt.]

NOMBRES — Ainpho

Un nombre de Tamain [4] est un nombre divisible par la somme de ses chiffres *(écriture décimale)* et ne contenant aucun 0 dans son écriture.

Par exemple | 117 | 126 | 1998 | mais pas 131 ni 2 016.

Ecrivez un script Python qui prend en entrée n et teste si il est de Tamain. [4 pt.]

Prouvez que ces nombres là le sont :

1 999 998
$123\underline{22\ldots2}$ sachant que dans la partie soulignée il y a 98 chiffres 2
9 999 999 999 999 125

[4 pt.]

M.P.S.I.2 2016 — 54 points — 2017 Charlemagne — ℵ Ainpho ℵ

Euler a "prouvé" que le n^{ieme} nombre premier est équivalent à $n.\ln(n)$. Ce fut prouvé par DeLaValléePoussin et Hadamard.

[4] "laisse moi devenir l'ombre de ton ombre, l'ombre de ta main, l'ombre de ton chien" (Jacques Brel, "*Ne me quitte pas*")

MPSI2 2016 2017 Lycée Charlemagne

Correction

Ainpho

La procédure mystère. Ainpho

Cette procédure prend en entrée un nombre `A`. Elle ne l'utilise pas directement.
Au mieux, elle utilise `A` dans un test de rupture de boucle.
Elle initialise ensuite des variables dont les noms sont `S` *(comme somme si l'on se fie à `S+=...`)* et `C` *(comme compteur si l'on note que `C` est incrémenté à chaque boucle d'une valeur `1` ou `0`)*. On initialise aussi un `n` qui va grimper pas à pas *(instruction `n+=1`)*.
A chaque tout, l'accumulateur `S` est incrémenté d'un nombre en `test/n`.
Qui est `test` ? C'est comme un booléen, puisqu'il ne peut prendre que les valeurs `0` et `1`.
Pour n donné, on met au départ le booléen à 1 et on prend un `k` qui va de `2` à `n-1`. Si `k` divise `n` *(c'est à dire si `n` admet un diviseur autre que `1` et lui même)*, on remet le `test` à `0`.
Bilan : après parcours de

```
Test = 1
for k in range(2, n) :
....if n%k == 0 :
.......Test = 0
```

on dispose d'un booléen qui vaut `1` si `n` n'a pas de diviseur et `0` sinon. C'est un test (peu efficace) de primalité.
Avec l'instruction

```
S += Test*N**(-1)
C += Test
```

on augment `S` de `1/n` si `n` est premier, sinon, on ne l'augmente pas. Et si n est premier, on incrémente le compteur d'une unité.
Bref, la somme `S` accumule les inverses de nombres premiers : $S_n = \sum_{\substack{p\leqslant n\\ p \text{ premier}}} \frac{1}{p}$.
Et elle le fait tant que la somme n'a pas dépassé `A`.
On retourne finalement le premier n tel que $\sum_{\substack{p\leqslant n\\ p \text{ premier}}} \frac{1}{p} \geqslant A$.

On cherche donc le premier indice pour lequel la somme des inverses des nombres premiers dépasse `A`. Et le compteur C indique combien il y a eu de nombres premiers utilisés.f.

Par exemple `mystere(1)` s'arrête quand $\frac{1}{2}+\frac{1}{3}+\frac{1}{5}+\frac{1}{7}+\frac{1}{11}+\ldots+\frac{1}{p}$ dépasse 1. C'est en fait vrai avec $\frac{1}{2}+\frac{1}{3}+\frac{1}{5}$. La réponse est donc `N=5` et `C=3`.

Mystère(5) veut que la somme dépasse 5. On se demande si c'est réalisable.
On sait déjà que la liste des inverses de tous les nombres grimpe en $\ln(n)$. Pour qu'elle dépasse 5, il faut déjà que n soit de l'ordre de e^5. Mais comme on ne garde qu'une partie des nombres entiers, la croissance est largement plus lente.
D'ailleurs, si la série des inverses des nombres premiers converge vers un nombre plus petit que 5, ce n'est pas la peine d'attendre, on ne dépassera jamais 5.
Soyons plus précis quoique un peu approximatif.
Si l'on se fie à Euler, le n^{ieme} nombre premier est équivalent à $n.\ln(n)$.
Prenons nous à supposer que le le n^{ieme} nombre premier soit exactement $n.\ln(n)$.
On doit alors regarder la convergence ou divergence de la série de terme général $1/n.\ln(n)$, et en cas de divergenec, on doit estimer sa vitesse.
Par comparaison série intégrale, on compare $\sum_{n=2}^{N} \frac{1}{n.\ln(n)}$ et $\int_1^N \frac{dt}{t.\ln(t)}$. On trouve une divergence à

la vitesse de $\ln(\ln(n))$.
On finira donc par dépasser toute valeur A imposée dès le début.
Mais ça risque de prendre du temps... Pour dépasser 5, on va attendre que N soit de l'ordre de e^{e^5}, nombre à 64 chiffres... Peu réaliste.

Nombres de Tamain. Ainpho

On prend l'entier `N`, on le fait fondre à coups de divisions par `10` et on garde à chaque fois le chiffre des unités

```
while NN > 0 :
....unite = NN%10
....NN = NN/10 #division euclidienne
```

Si le chiffre des unités est un `0` on sort brutalement par un `return(False)` [5] sinon, on l'accumule dans une somme `S` initialisée à `0`.

```
while NN > 0 :
....unite = NN%10
....if unite == 0 :
........return(False)
....S = S+unite #pas besoin de else, en cas de 0 on est déjà sorti
....NN = NN/10 #division euclidienne
```

Une fois qu'on a calculé la somme des chiffres, on teste `n%S==0`.
Au lieu de

```
if
n%S==0 :
....return(True)
else :
....return(False)
```

on préférera `return(n%S==0)` qui retourne directement le booléen évalué.
Attention, la procédure avec les `n%10` et `n/10` a fait fondre `n`, il faut en avoir conservé un double.

```
def Test(n) :
....S = 0
....N = n
....while N>0 :
........unite = N%10
........if unite == 0 :
............return(False)
........S = S+unite
....return(n%S==0)
```

Les nombres à tester sont du domaine des mathématiques.
Pour ce qui est de 1 999 998, la somme de ses chiffres vaut 54 c'est à dire 2.3^3.
On vérifie qu'il est divisible par 2. Il est divisible par 9 *(somme des chiffres)*.
Après division par 9 on trouve 222 222, encore divisible par 3.
Ou alors on pose directement la division : 37 037.

Qui est le nombre $22\ldots2$ avec 98 chiffres 2 ?

Déjà, 222 par exemple, c'est $2+2.10+2.100$. Donc, $222\ldots2$ c'est $2.\sum_{k=0}^{96} 10^k$, c'est à dire $2.\dfrac{10^{99}-1}{9}$.

On lui ajoute $10^{99}.123$: $N = 123.10^{99} + 2.\dfrac{10^{97}-1}{9}$.

[5] si vous vous obstinez à mettre des `print`, au lieu de `return` repassez un bac L, vous savez écrire, mais vous ne savez pas communiquer

La somme de ses chiffres vaut $1+2+3+98\times 2$ c'est à dire 202.
On voit déjà que notre nombre est pair *(il se termine par 2)*.
Reste à prouver que c'est un multiple de 101.
On envisage de poser la division euclidienne et d'y voir un motif.
On trouve vite 610011001100…. Vu le nombre total de chiffres, la division tombe juste.

Il n'y a pas mieux ? Si ! Notre nombre est $2.\dfrac{10^{100}-1}{9}+101.10^{99}$.
Le terme 101.10^{99} *est évidemment multiple de* 202.
Le nombre 101 *est premier, donc* $10^{100}-1$ *est multiple de* 101 *(Fermat).*

La somme des chiffres de 9 999 999 999 999 125 vaut $13\times 9+1+2+5$ ce qui fait 125.
Modulo 125, on le réduit, en sachant que 9 999 999 999 999 000 est un multiple de 1 000 donc de 125.
Pas besoinde poser la division, elle tombera juste.
Si vous calculez au lieu de réfléchir, voici la réponse : 79 999 999 999 993.

Algèbre linéaire. Ainpho

Questions de mathématiques. On se donne A de terme général a_i^k. On écrit $A-X.I_3=\begin{pmatrix} a_1^1 & a_1^2 & a_1^3 \\ a_2^1 & a_2^2 & a_2^3 \\ a_3^1 & a_3^2 & a_3^3 \end{pmatrix}-\begin{pmatrix} x & 0 & 0 \\ 0 & x & 0 \\ 0 & 0 & x \end{pmatrix}=\begin{pmatrix} a_1^1-x & a_1^2 & a_1^3 \\ a_2^1 & a_2^2-x & a_2^3 \\ a_3^1 & a_3^2 & a_3^3-x \end{pmatrix}$. On développe ce déterminant par rapport à la première colonne, on trie ensuite en fonction de X :

terme	$-X^3$	$+X^2.(a_1^1+a_2^2+a_3^3)$	$-X.(a_1^1.a_2^2-a_1^2.a_2^1+a_1^1.a_3^3-a_3^3.a_3^1+a_2^2.a_3^3-a_3^3.a_3^2)$	$+X.(\det(A))$
coefficient	-1	$a_1^1+a_2^2+a_3^3$	$-(a_1^1.a_2^2-a_1^2.a_2^1+a_1^1.a_3^3-a_3^3.a_3^1+a_2^2.a_3^3-a_3^3.a_3^2)$	$\det(A)$
remarque		$Tr(A)$	$Tr(Com(A))$	$det(A)$

Pour le terme sans X, inutile de se prendre la tête à le développer. C'est la valeur pour x égal à 0. C'est donc $\det(A)$.

On a bien un polynôme et il est vraiment de degré 3 : $\boxed{-X^3+Tr(A).X^2-Tr(Com(A)).A+\det(A)}$

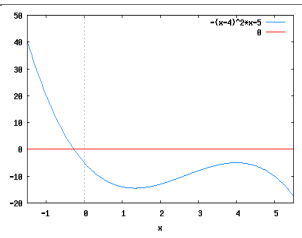
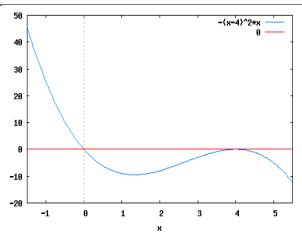
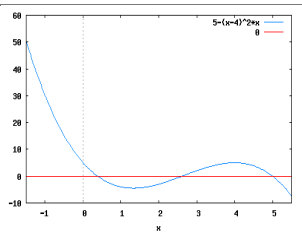
Pour perdre bêtement les points : je mets des $-X$ partout, alors que dans I_3 ils ne sont que sur la diagonale.

On se fixe λ et on étudie l'ensemble des vecteurs colonne U vérifiant $A.U=\lambda.U$. Si on a le nez collé sur le guidon, on écrit le système $\begin{pmatrix} a_1^1 & a_1^2 & a_1^3 \\ a_2^1 & a_2^2 & a_2^3 \\ a_3^1 & a_3^2 & a_3^3 \end{pmatrix}.\begin{pmatrix} x \\ y \\ z \end{pmatrix}=\lambda.\begin{pmatrix} x \\ y \\ z \end{pmatrix}$. Si on la carrément collé à la roue, on tente de le résoudre.
Si on fait des maths *(ce qui est quand même notre objectif souvent, même en I.P.T.)*, on cherche juste à prouver "sous-espace vectoriel de $\mathbb{R}^3$".
L'inclusion est dans la définition. Le vecteur nul vérifie $A.0_3=0_3=\lambda.0_3$.
Si on prend U et V vérifiant $A.U=\lambda.U$ et $A.V=\lambda.V$, on a sans effort $A.(U+V)=\lambda.U+\lambda.V=\lambda.(U+V)$ et aussi $A.(\alpha U)=\lambda.\alpha.U=\alpha.(\lambda.U)$. On a la stabilité.
Le faux mathématicien dit c'est évident, et quand on lui demande quelles formules il faut prouver (comme $A.(U+V)=\lambda.U+\lambda.V=\lambda.(U+V)$) il invente n'importe quoi. On le sanctionne évidemment.

Le degré du polynôme vaut 3. Et le polynôme est à coefficients réels. Il a au moins une racine par théorème des valeurs intermédiaires. Comme tout polynôme de degré 3.
La question était "*montrez qu'il n'en a pas plus que* 3", mais aussi "*montrez qu'il en a au moins une…*"

La locution "deux racines" est ambigüe mais peut correspondre au cas où il y a une racine double et

une racine simple.

	1	2	3
			
	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix}$	$P.\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}.P^{-1}$

Quel rapport entre les vecteurs vérifiant $A.U = \lambda.U$ et le déterminant de $A - \lambda.I_3$?

$A.U = \lambda.U$	$A.U - \lambda.U = 0_3$	$A.U - \lambda.I_3.U = 0_3$	$(A - \lambda.I_3)U = 0_3$

Le sous-espace est de dimension non nulle si et seulement si il existe au moins un vecteur non nul dedans.

Si la matrice $A - \lambda.I_3$ a un déterminant nul, elle est inversible, et en multipliant par son inverse, le système $(A - \lambda.I_3)U = 0_3$ conduit immédiatement à $U = (A - \lambda.I_3)^{-1}.0_3 = 0_3$. Il n'y a que le vecteur nul.

Si en revanche elle a un déterminant nul, le système admet une solution (et tous ses multiples), $E_\lambda$est au moins une droite.

On a bien une équivalence entre

λ est valeur propre	$\det(A - \lambda.I_3) = 0$	il y a au moins un vecteur non nul vérifiant $A.U = \lambda.U$

Pour ce qui est du script il faut calculer les trois coefficients.

Le mieux est de d'abord nommer simplement les coefficients de la matrice, puis de calculer

```
a, b, c = A[0][0], A[0][1], A[0][2]
aa, bb, cc = A[1][0], A[1][1], A[1][2]
aaa , bbb, ccc =A[2][0], A[2][1], A[2][2]
trace = a+bb+ccc
det = (a*bb*ccc-a*bbb*cc)+(b*cc*aaa-b*ccc*aa)+(c*aa*bbb-c*aaa*bb)
tracecom = (a*bb-aa*b)+(a*ccc-aaa*c)+(bb*ccc-bbb*cc)
```

Les parenthèses ne servent à rien sauf à s'y retrouver.

On crée ensuite la fonction appelée charpol *(polynôme caractéristique, Charpoly)*

```
def charpol(x) :
....return(-x*x*x+trace*x*x-tracecom*x+det)
```

Puis on utilise les modules de numpy auxquels je ne me suis jamais habitué.

Pour la fonction, les informaticiens préféreront

```
def charpol(x) :
....P = trace-x
....P = x*P-tracecom
....P = x*P+det
```

qui évite les puissances. C'est ce qu'on appelle une *"factorisation" de Hörner.*

Ensuite, pour savoir combien il y a de racines, il faut étudier les variations du polynôme caractéristique, c'est à dire le signe de $P'(X) = -3.X^2 + 2.Tr(A).X - Tr(Com(A))$.

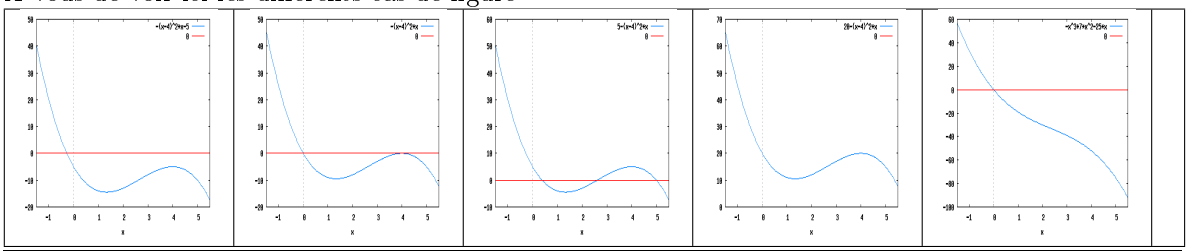
On calculera le discriminant.

Si il est négatif, impossible d'avoir plus d'une racine, P décroît sans cesse.

Si il est positif, on calcule les deux racines dans l'ordre, et on regarde les signes.

```
Delta = (2*trace)**2-12*tracecom
if Delta<0 :
....return(1)
r1 = (2*trace-sqrt(Delta))/6
r2 = (2*trace-sqrt(Delta))/6
y1 = charpol(r1)
y2 = charpol(r2)
if y1*y2 == 0 :
....return(2)
if y1*y2>0 :
....return(1)
return(3)
```

A vous de voir ici les différents cas de figure



- *Magasins* que *fréquente* MARC

Pas de difficulté.

```
SELECT Magasin
FROM frequente
WHERE Client = 'Marc'
```

- Nombre de *magasins* qui *vendent* des *livres*.

Pas de difficulté non plus.

```
SELECT COUNT(Magasin)
FROM vend
WHERE Produit LIKE 'Livre%' #par sécurité si on utilise ou non le pluriel
```

- *Produits vendus* au *magasin* BELLEVILLE, triés par ordre alphabétique

Toujours pas de difficulté.

```
SELECT Produit
FROM vend
WHERE Magasin = 'Belleville'
ORDER BY Produit
```

Une requête à déchiffrer en deux étapes :

- ```
  SELECT Magasin
  FROM vend
  WHERE Produit IN ('Livres', 'Jouets')
  ```

Les magasins où on vend des livres ou des jouets *(je sais, dans mes exemples, aucun ne le fait, mais je rappelle que je ne donne qu'un exemple de quatre ou cinq fiches sur quelques milliers...).*
  ```

- ```
  SELECT Client
  FROM frequente
  WHERE Magasin IN
  ....(SELECT Magasin
  .....FROM vend
  .....WHERE Produit IN ('Livres', 'Jouets'))
  ```

Les clients qui vont dans des magasins où on vend des livres et des jouets.

- Ce que l'on trouve à RIVOLI puis à RIVOLI et BELLEVILLE

En deux étapes. Déjà, ce qui se *vend* à RIVOLI.

```
SELECT Produit
FROM vend
WHERE Magasin = 'Rivoli'
```

Puis on regarde si on les trouve aussi à BELLEVILLE

```
SELECT Produit
FROM vend
WHERE Magasin = 'Belleville' AND Produit IN
....(SELECT Produit
.....FROM vend
.....WHERE Magasin = 'Rivoli')
```

On interprête une requête :

- ```
  SELECT frequente.Client
  FROM vend as V JOIN frequente as F JOIN aime as A
  WHERE A.Produit=V.Produit AND V.Magasin=F.Magasin AND A.Client=F.Client
  ```

On affiche une liste de clients.
Mais il y a une condition triple dans une grande jointure.
C'est simple :
le client va dans un magasin
ce magasin vend des produits
ce sont des produits que le client aime.

Bref, c'est la liste des clients qui vont dans un magasin qui vend les produits qu'ils aiment.

- ```
  SELECT Produit
  FROM aime
  WHERE Client = 'Martine'
  ```

On cherche les produits que Martine aime.

- ```
  SELECT magasin
  FROM vend
  WHERE produit IN
  ....(SELECT produit
  .....FROM aime
  .....WHERE client = 'Martine')
  ```

On donne la liste des magasins qui vendent les produits que Martine aime.
Bref, on indique à Martine où aller faire ses courses.

- *Clients* qui vont dans tous les *magasins*.

On va sélectionner des clients dans une table. Comme ce sont des clients qui vont quelquepart, c'est la table frequente.
Comment savoir qu'ils vont partout ?
On peut *compter* tous les *magasins* :

```
SELECT COUNT(DISTINCT Magasin)
FROM vend
```

On peut compter les *magasins* fréquentés par un *client* donné :

```
SELECT COUNT(DISTINCT Magasin)
FROM frequente
WHERE Client = '...'
```

On peut alors tester :

```
WHERE
....(SELECT COUNT(DISTINCT Magasin)
.....FROM frequente
.....WHERE Client = '...')
....=
....(SELECT COUNT(DISTINCT Magasin)
.....FROM vend)
```

On termine donc :

```
SELECT Client
FROM frequente AS F
WHERE
....(SELECT COUNT(DISTINCT Magasin)
.....FROM frequente
.....WHERE Client = F.Client)
....=
....(SELECT COUNT(DISTINCT Magasin)
.....FROM vend)
```

Dans les livres, j'ai trouvé ça :

```
SELECT Client
FROM frequente
WHERE Client NOT IN
....(SELECT F1.Client
.....FROM frequente AS F1, Frequente as F2
.....WHERE F1.Client NOT IN
.........(SELECT Client
..........FROM frequente
..........WHERE Magasin=F2.Magasin)
```

- Produits que CLAUDE aime et qui sont vendus à BELLEVILLE.

On va afficher les `produits`.
On va avoir besoin de deux tables : `vend` et `aime`, qu'on va jointurer.
On va mettre un jeu de conditions :

- le `client` de la table `aime` est CLAUDE
- le `produit` de la table `aime` est le même que de la table `vend`
- le `magasin` de la table `vend` est BELLEVILLE

```
SELECT vend.Produit
FROM vend JOIN aime
WHERE vend.magasin='Belleville' AND vend.Produit=aime.Produit AND
aime.Client='Claude'
```

- ```
  SELECT Client
  FROM vend as V JOIN aime AS A JOIN frequente AS F
  WHERE V.Produit=A.Produit AND A.Client=F.Client AND F.Magasin=V.Magasin
  GROUP BY Client
  HAVING COUNT(Magasin)>2
  ```

On cherche des clients qui vont dans des magasins où on vend des produits qu'ils aiment. On regroupe, et pour chaque client, on met une clause : il y a au moins deux magasins où il va.
C'est donc la liste des clients qui vont dans au moins deux magasins dans lesquels on vend des produits
  ```

qu'ils aiment.

- ```
  SELECT Client
  FROM frequenter
  WHERE Client NOT IN
  ........(SELECT Client
  .........FROM frequenter
  .........WHERE Magasin NOT IN
  ................(SELECT Magasin
  ................FROM vend JOIN aime
  ................WHERE aime.Client = frequenter.Client AND vend.Produit =
  aime.Produit))
  ```

Facile ![6] C'est la liste des clients qui ne vont que dans les magasins où il y a des produits qu'ils aiment.
Mais c'est de la logique, car on passe par le complémentaire :
les clients qui fréquentent au moins un magasin qui ne sert aucun des produits qu'ils aiment...
Aux concours, on n'ira jamais vous demander ça.
Mais c'est le genre de requête qui semble élémentaire à ceux qui conçoivent le sujet si ils pratiquent les bases de données.

Si vous avez aimé ou si vous voulez vous perfectionner avec des questions en plus :

```
SELECT Client
FROM Frequenter AS F
WHERE Client NOT IN
....(SELECT A.Client
.....FROM vend AS V JOIN aime AS A
.....WHERE V.Produit=A.Produit
.....AND A.Client NOT IN
.........(SELECT Client
..........FROM frequente
..........WHERE Magasin=V.Magasin))
```

C'est la liste des clients qui fréquentent tous les magasins qui vendent au moins un produit qu'ils aiment.
On passe par le complémentaire : les clients qui ne fréquentent pas au moins un magasin qui vend des produits qu'ils aiment.

[6] non, je plaisante
```