Bases de données (2/2) : écosystème et découverte de SQL

D'après un TP de Stéphane Gonnord .

Buts du TP

- Apprendre à créer/alimenter une base « à l'aide d'un outil interactif ».
- Pratiquer des requêtes avec fonctions d'agrégation.
- Faire des requêtes mixant agrégations et jointures plus ou moins élaborées.

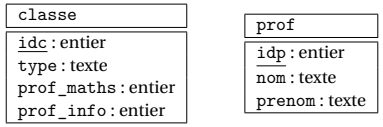
EXERCICE 1 Copier/Coller le répertoire du TP récupéré dans le répertoire `Donnees` de la classe. Tous les fichiers nécessaires (bases de données, fichiers textes, images, scripts ...) se trouvent dans le sous-répertoire `materiel` :

`cinema843.db` `cafes.csv` `titres_reserves.csv` `prenoms_paris.db` `notes_colle.db` `clubs.csv` `foot0.db`

1 Création et alimentation d'une base de données

EXERCICE 2 *Créons une base à la souris.*

1. Créer (depuis `sqliteman`) une base de données contenant deux tables de schémas relationnels suivants :



   Réfléchir aux liens entre les tables.

2. Entrer quelques lignes dans chacune de ces tables. Il y aura **entre autres** les tuples :

```
(841, 'PCSI', 17, 17)
(842, 'PCSI', 23, 69)
(843, 'PCSI', 15, 69)
(17, 'Vanderlynden', 'Laurent')
(23, 'Moynot', 'Olivier')
```

3. Commiter les changements, quitter `sqliteman`. Relancer `sqliteman` et ouvrir la base précédemment créée ; vérifier que tout le monde est où il faut !

EXERCICE 3 *Importons une table*

1. Ouvrir le fichier `titres_reserves.csv`, qui contient les 1000 titres les plus réservés dans les bibliothèques municipales à Paris en 2013. Observer la nature des informations, et en déduire un schéma relationnel pour inclure ces données dans une base de données.
2. Créer une base de données constituée d'une table répondant à ce schéma.
3. Importer les données du fichier `csv` dans cette table en choisissant le point-virgule comme séparateur de champ [1]. Fermer `sqliteman` puis l'ouvrir, et charger la base de données préalablement créée.
4. Exécuter la requête (en adaptant éventuellement le nom des attributs) :

```
SELECT auteurs,SUM(nombre) AS s
   FROM prets
   GROUP BY auteurs
   ORDER BY s DESC
```

   Observer le résultat (dans les 20 premiers titres), et pointer le(s) problème(s) : c'est typique d'une base de données mal conçue/pensée au départ.

1. Oui, vous allez avoir des petites erreurs de signalées ; passons ! Par contre, après l'import, il faut commiter...

2 Prénoms parisiens

La base `prenoms_paris.db` les prénoms enregistrés à l'état civil de Paris depuis 2004 jusqu'à 2013.

EXERCICE 4 *Ouvrons les yeux*
Donner le schéma relationnel de cette base (différents attributs des différentes tables).

EXERCICE 5 *De SQL vers le français.*
Pour chacune de requêtes SQL suivantes, donner la traduction « en français », et vérifier la vraisemblance du résultat depuis `sqliteman`.

```
SELECT DISTINCT prenom FROM enregistrement_etat_civil

SELECT SUM(nombre) FROM enregistrement_etat_civil

SELECT COUNT(DISTINCT prenom) FROM enregistrement_etat_civil

SELECT annee,SUM(nombre) FROM enregistrement_etat_civil GROUP BY annee

SELECT prenom,annee,SUM(nombre) as s
   FROM enregistrement_etat_civil
   GROUP BY prenom,annee
   HAVING s>=300 ORDER BY s DESC

SELECT prenom,SUM(nombre) as somme
   FROM enregistrement_etat_civil
   GROUP BY prenom
   HAVING somme>=2000 ORDER BY somme DESC
```

EXERCICE 6 *À vous de jouer.*

1. Combien de naissances de filles ont été enregistrées ?
2. Combien de fois votre prénom a-t-il été donné à Paris pendant les années concernées ?

   On donnera le résultat trié par années croissantes. Attention aux accents, pas toujours bien traités dans la base (le fichier initial était défaillant...).
3. Même question avec le prénom du professeur d'informatique.
4. Donner, pour chaque année, le nombre de prénoms différents qui ont été enregistrés.
5. Quels prénoms ont été donnés exactement 100 fois ?
6. Quel est le prénom qui a été le plus donné sur l'ensemble de la période ?
7. Quel est le prénom *féminin* qui a été le plus donné sur l'ensemble de la période ?

3 Notes de colles

La base de données `notes_colles.db` contient trois tables, décrivant des colles virtuelles, données par des agrégés de la promotion 1930 à des agrégés de la promotion 1950.

EXERCICE 7 *Qu'est-ce qu'on a ?*
Ouvrir cette base. Écrire son schéma relationnel. Comprendre le lien entre les différents attributs des différentes tables.

EXERCICE 8 *C'est parti*

1. Déterminer la liste des noms et prénoms des professeurs... et ceux des élèves.
2. Déterminer le nombre de « 20 » qui ont été attribués, ainsi que les notes majorées par 6.

EXERCICE 9 *Avec jointures*

1. Déterminer les notes de Jacques-Louis Lions (triées selon les semaines croissantes).

2. Refaire la même chose, avec cette fois le nom des colleurs.
3. Déterminer les quadruplets (élève, professeur, note, semaine) pour toutes les colles où la note était supérieure ou égale à 19.

EXERCICE 10 *Des agrégats (ou groupes)*

1. Déterminer la moyenne des notes de colle de Jacques-Louis Lions.
2. Déterminer la liste des couples (élève, moyenne).
3. Respirer lentement, puis écrire une requête permettant de calculer la moyenne des moyennes.
   Merci de ne pas tricher, et bien calculer une moyenne de moyennes, et non pas la moyenne globale des notes (qui est la même ici, tout le monde ayant eu 25 *colles !)*
4. Parmi tous les élèves, déterminer le nom de ceux ayant eu au moins 10 notes strictement sous la moyenne.
5. Déterminer les élèves ayant eu une moyenne de 14 ou plus.
6. Parmi tous les élèves, déterminer ceux ayant eu au moins 6 notes strictement supérieures à 18.

4 Retour au cinéma

Cet exercice porte sur la base de données `cinema843.db` déjà étudiée dans le TP 14. Il s'inspire d'un TP du *livre Bases de données : de la modélisation au SQL* de Laurent Audibert qui est édité chez Ellipses et d'un TP posé en 2013/2014 par M.Ridde en classe de 833.

EXERCICE 11 *Des jointures et des agrégats*

Charger la base de données `cinema843.db` dans `Sqliteman` et écrire des requêtes SQL permettant de répondre aux problèmes suivants :

1. Dresser la liste des nombres de films par genre.
2. Déterminer combien de films différents ont été projetés à l'Astoria.
3. Dresser la liste des acteurs (nom et prénom) en précisant pour chacun le nombre de films dans lesquels il a joué.
   La liste doit être triée par odre alphabétique croissant des noms.
4. Déterminer le nombre de films réalisés par chacun des réalisateurs.
5. Dresser la liste des nombres de projections par cinéma puis la liste des nombres de films projetés par cinéma.
6. Déterminer le nombre de films réalisés par le réalisateur qui en a réalisé le plus.
7. Déterminer les réalisateurs (possibilité d'ex aequo) ayant réalisé le plus de films.
8. Déterminer les genres de films (possibilité d'ex aequo) qui comptent le moins de films
9. Déterminer les cinémas qui ont projeté tous les films.
10. Déterminer les acteurs qu'il a été possible de voir dans tous les cinémas.
11. Déterminer le nombre de films réalisés par Lars von Trier.
12. Déterminer les acteurs qui ont joué dans tous les films de Lars von Trier.

5 Des matchs de foot

On veut créer une base de matchs de foot entre clubs européens prestigieux (le Réal, la Juve, Guingamp, etc.) sur quelques dizaines d'années. Disons de 1971 à 2014.

Les impatients peuvent passer directement au dernier exercice, où on fait des requêtes (absurdes) sur une telle base (absurde) !

EXERCICE 12 *Les clubs.*

1. Ouvrir le fichier `clubs.csv`, observer le contenu. Définir sur papier un schéma relationnel permettant de représenter les différents clubs. *On impose un attribut entier qui sera une clé primaire : cet attribut ne prend jamais deux fois la même valeur.*

2. À l'aide d'un script Python, créer une base de données, constitué d'une table `clubs`, et l'alimenter à l'aide du fichier `clubs.csv` qu'on lira ligne-à-ligne.
   On pourra s'inspirer du canevas ci-dessous :

```
def creation_base_clubs():
    base = sqlite3.connect("foot843.db") #connexion à la base
    curseur = base.cursor()  #création du curseur
    curseur.execute("DROP TABLE IF EXISTS clubs")
    curseur.execute("CREATE TABLE clubs (idc integer primary key, nom text, pays text)")
    source = open('clubs.csv','r')
    ....
    for ligne in source:
        .....
        #insertion dans la table
        curseur.execute("INSERT INTO clubs VALUES (?,?,?)",(idc,nom,pays))
        .....
    source.close() #fermeture du fichier texte source
    base.commit() #pour enregistrer les modifications
    base.close() #fermeture de la connexion à la base
```

EXERCICE 13 *Et maintenant, les matchs*

Un match est la donnée de deux équipes *distinctes*, deux scores, une année, un jour dans cette année.

1. Définir sur papier un schéma relationnel pour constituer une table représentant des matchs. Ajouter cette table (vierge, pour le moment) à la base de donnée en cours.

   Pour établir le nombre b de buts marqués par une équipe lors d'un match [2], on tire au hasard $x \in [0,1]$. Ensuite

   - si $x > 3/4$, b vaut 0 ;
   - si $5/12 < x \leqslant 3/4$, b vaut 1 ;
   - si $1/12 < x \leqslant 5/12$, b vaut 2 ;
   - sinon, on détermine le plus petit $k \geqslant 3$ tel que $x \leqslant \frac{1}{12}\left(1 - \frac{1}{4^{k-2}}\right)$, et b vaut alors k.

2. Écrire une fonction `buts` sans entrée, et retournant un entier déterminé selon les rêgles précédentes. Vérifier la vraisemblance en faisant 10^6 tirages et en observant les fréquences.
   Les probabilités théoriques d'obtenir 0, 1, 2, ... buts sont respectivement $\frac{1}{4}, \frac{1}{3}, \frac{1}{3}, \frac{1}{4^2}, \frac{1}{4^3}, \frac{1}{4^4}$, *etc* ...
3. Remplir la base en réalisant 100 matchs aléatoires pour chaque année entre 1971 et 2014.

La base `foot843.db` est un exemple de telle base générée aléatoirement.

EXERCICE 14 *Et maintenant, jouons un peu !*

1. Que signifie la requête suivante ?

```
SELECT c1.nom, sc1, c2.nom, sc2
   FROM clubs AS c1 JOIN matchs  JOIN clubs AS c2
      ON c1.idc = eq1 AND c2.idc=eq2
         WHERE c1.nom='Real Madrid' OR c2.nom='Real Madrid'
```

2. Déterminer les matchs où le Paris Saint-Germain a mis strictement plus de un but à l'extérieur sans gagner.
3. Combien de fois un club français a-t-il marqué au moins 4 buts contre un club italien ?
4. Concevoir quelques requêtes absurdes.

Ça y est : vous êtes prêts pour concevoir des feuilles de TP vous-mêmes !

2. on ne se soucie pas du fait de jouer à domicile ou à l'extérieur

Corrigé du TP 15 Base de Données n° 2

Chenevois-Jouhet-Junier

1 Exercice 3

Exo 3 : Tous les auteurs classés selon le nombre de réservations par ordre décroissant. L'auteur null (type None en Python) arrive en tête car le champ auteur n'était pas complété pour de nombreuses lignes.

```
SELECT auteurs, SUM(nombre) AS s
FROM prets
GROUP BY auteurs
ORDER BY s DESC
```

```
(None, 26449.0)
(Gloaguen, Philippe, 2027)
(Sobral, Patrick, 1487)
(Kirkman, Robert, 1392)
... (380 de plus)
```

2 Exercice 5

Exo 5 : Quels sont tous les prenoms distincts de la table ?

```
SELECT DISTINCT prenom
    FROM enregistrement_etat_civil
```

```
(Liz)
(Lohan)
(Lou)
... (2061 de plus)
```

Exo 5 : Nombre de prénoms enregistrés dans la table


```
SELECT SUM(nombre)
    FROM enregistrement_etat_civil
```

```
(311109)
```

Exo 5 : Nombre de prénoms distincts dans la table

```
SELECT COUNT(DISTINCT prenom )
    FROM enregistrement_etat_civil
```

```
(2064)
```

Exo 5 : Nombre de prenoms (ou naissances) enregistrés par année

```
SELECT annee, SUM(nombre)
    FROM enregistrement_etat_civil
    GROUP BY annee
```

```
(2004, 30780)
(2005, 30356)
(2006, 31043)
... (7 de plus)
```

Exo 5 : Quels sont les triplets (prénom,année, nombre de naissances) où le prenom a été enregistré plus de 300 fois lors de cette année, dans l'ordre décroissant des nombres de naissance ?

```
SELECT prenom,annee,SUM(nombre) as s
    FROM enregistrement_etat_civil
    GROUP BY prenom, annee
    HAVING s >= 300
    ORDER BY s DESC
```

```
(Gabriel, 2010, 398)
(Gabriel, 2013, 381)
(Gabriel, 2011, 374)
... (19 de plus)
```

Exo 5 : Quels sont les prenoms (et leur nombre d'enregistrement) qui ont été enregistrés plus de 2000 fois dans la table ?

```
SELECT prenom,SUM(nombre) as somme
   FROM enregistrement_etat_civil
   GROUP BY prenom
   HAVING somme >= 2000 ORDER BY somme DESC
```

```
(Gabriel, 3278)
(Raphaël, 2875)
(Louise, 2788)
... (14 de plus)
```

3 Exercice 6

Exo 6 : Combien de naissances de filles ont été enregistrées ?

```
SELECT SUM(nombre)
    FROM enregistrement_etat_civil
    WHERE sexe = 'F'
```

```
(149345)
```

Exo 6 : Combien de fois votre prénom a-t-il été donné à Paris pendant les années concernées en distinguant les caractères accentués ?

```
SELECT  prenom, SUM(nombre)
    FROM enregistrement_etat_civil
    WHERE prenom LIKE 'St%phane'
    GROUP BY prenom
```

```
(Stephane, 88)
(Stéphane, 6)
```

Exo 6 : Combien de fois votre prénom a-t-il été donné à Paris pendant les années concernées sans distinguer les caractères accentués ?

```
SELECT  SUM(nombre)
    FROM enregistrement_etat_civil
    WHERE prenom LIKE 'St%phane'
```

```
(94)
```


Exo 6 : Combien de fois le prénom du professeur d'informatique a-t-il été donné à Paris pendant les années concernées ?

```
SELECT  SUM(nombre)
    FROM enregistrement_etat_civil
    WHERE prenom LIKE 'Laurent'
```

```
(100)
```

Exo 6 : Donner, pour chaque année, le nombre de prénoms différents qui ont été enregistrés. La requete avec DISTINCT prenom donne un résultat différent car certains prénoms sont mixtes et peuvent etre donnés à des filles ou à des garçons

```
SELECT annee,COUNT(DISTINCT prenom)
    FROM enregistrement_etat_civil
    GROUP BY  annee
```

```
(2004, 1005)
(2005, 1021)
(2006, 1055)
... (7 de plus)
```

Exo 6 : Quels prénoms ont été donnés exactement 100 fois ?

```
SELECT prenom,SUM(nombre) as n
    FROM enregistrement_etat_civil
        GROUP BY  prenom
            HAVING n=100
```

```
(Coumba, 100)
(Cyril, 100)
(Eyal, 100)
(Laurent, 100)
(Mayeul, 100)
(Ombeline, 100)
(Youcef, 100)
```

Exo 6 : Quels prénoms ont été donnés exactement 100 fois ?

```
SELECT prenom
    FROM
     (SELECT  prenom, SUM(nombre) AS  somme
```

```
        FROM enregistrement_etat_civil
            GROUP BY prenom)
    WHERE somme = 100
```

```
(Coumba)
(Cyril)
(Eyal)
(Laurent)
(Mayeul)
(Ombeline)
(Youcef)
```

Exo 6 : Quel est le prénom qui a été le plus donné sur l'ensemble de la période ?

```
SELECT prenom,SUM(nombre) as n
FROM enregistrement_etat_civil
GROUP BY  prenom
HAVING n = (SELECT MAX(k) FROM
                (SELECT  SUM(nombre) as k
                    FROM enregistrement_etat_civil
                    GROUP BY  prenom
                )
            )
```

```
(Gabriel, 3278)
```

Exo 6 : Une autre requete sans sous-requete et plus performante.

```
SELECT prenom,SUM(nombre) as n
    FROM enregistrement_etat_civil
        GROUP BY  prenom
            ORDER BY n DESC
                LIMIT 0,1
```

```
(Gabriel, 3278)
```

Exo 6 : Quel est le prénom féminin qui a été le plus donné sur l'ensemble de la période ?

```
SELECT prenom, SUM(nombre) AS somme
    FROM enregistrement_etat_civil
    WHERE sexe= 'F'
    GROUP BY prenom
    HAVING somme =
```



```
    (SELECT MAX(somme)
        FROM (SELECT  SUM(nombre) AS somme
                FROM enregistrement_etat_civil
                WHERE sexe= 'F'
                GROUP BY prenom
                )
    )
```

```
(Louise, 2788)
```

Exo 6 : Quel est le prénom féminin qui a été le plus donné sur l'ensemble de la période ?

```
SELECT prenom,SUM(nombre) as n
    FROM enregistrement_etat_civil
        WHERE sexe='F'
            GROUP BY prenom
                ORDER BY n DESC
                    LIMIT 0,1
```

```
(Louise, 2788)
```

Exo 6 : Quel est le prénom féminin qui a été le plus donné sur l'ensemble de la période ? Autre technique en créant une vue pour stocker la table virtuelle sur laquelle on veut travailler. D'abord on supprime une vue du meme nom si elle existe.

```
DROP VIEW IF EXISTS feminins
```

Exo 6 : Ensuite on crée la vue.

```
CREATE  VIEW  feminins AS
    SELECT prenom, SUM(nombre) AS somme
    FROM enregistrement_etat_civil
    WHERE sexe= 'F'
    GROUP BY prenom
```

Exo 6 : Enfin on formule la requete déterminant le prenom avec le maximum d'occurences.

```
SELECT prenom, somme FROM feminins
    WHERE somme = (SELECT MAX(somme)
```

```
                        FROM feminins
                )
```

```
(Louise, 2788)
```

4 Exercice 8

Exo 8 : Déterminer la liste des noms et prénoms des professeurs.

```
SELECT nom, prenom
    FROM profs
```

```
(Théron, Pierre)
(Brun, Jules)
(Durix, Marcel)
... (7 de plus)
```

Exo 8 : Déterminer la liste des noms et prénoms des eleves.

```
SELECT nom, prenom
    FROM eleves
```

```
(Lions, Jacques-Louis)
(Laurent, Jean)
(Chabert, Gilles)
... (26 de plus)
```

Exo 8 : Déterminer le nombre de 20 qui ont été attribués

```
SELECT COUNT(*)
    FROM colles
        WHERE note=20
```

```
(44)
```

Exo 8 : Déterminer le nombre de notes majorées par 6.

```
SELECT COUNT(*)
    FROM colles
        WHERE note<=6
```

```
(42)
```


5 Exercice 9

Exo 9 : Notes de Jacques-Louis Lions (triées selon les semaines croissantes).

```
SELECT colles.semaine, colles.note
    FROM colles JOIN eleves
        ON  colles.eleve = eleves.ide
            WHERE  (eleves.nom='Lions' AND eleves.prenom='Jacques-Louis')
                ORDER BY semaine ASC
```

```
(1, 19)
(2, 10)
... (23 de plus)
```

Exo 9 : La même chose, avec cette fois le nom des colleurs, deux jointures et des renommages.

```
SELECT c.semaine, c.note, p.nom
    FROM colles AS c JOIN eleves  AS e JOIN profs AS p
        ON (c.eleve = e.ide AND c.prof = p.idp)
            WHERE  (e.nom='Lions' AND e.prenom='Jacques-Louis')
                ORDER BY semaine ASC
```

```
(1, 19, Théron)
(2, 10, Poix)
... (23 de plus)
```

Exo 9 : Déterminer les quadruplets (élève, professeur, note, semaine) pour toutes les colles où la note était supérieure ou égale à 19.

```
SELECT e.nom,e.prenom,p.nom,p.prenom,c.note,c.semaine
    FROM colles AS c JOIN eleves  AS e JOIN profs AS p
        ON (c.eleve = e.ide AND c.prof = p.idp)
            WHERE  c.note>=19
                ORDER BY semaine ASC
```

```
(Lions, Jacques-Louis, Théron, Pierre, 19, 1)
(Roumieu, , Gerbaud, Philippe, 19, 1)
... (94 de plus)
```

6 Exercice 10

Exo 10 : Déterminer la moyenne des notes de colle de Jacques-Louis Lions.

```
SELECT  AVG(colles.note)
    FROM colles JOIN eleves
        ON colles.eleve = eleves.ide
            WHERE (eleves.nom='Lions' AND eleves.prenom='Jacques-Louis')

(13.24)
```

Exo 10 : Déterminer la liste des couples (élève, moyenne).

```
SELECT  eleves.nom, eleves.prenom, AVG(colles.note) AS moyenne
    FROM colles JOIN eleves
        ON colles.eleve = eleves.ide
            GROUP BY colles.eleve
                ORDER BY moyenne DESC

(Donedu, , 14.8)
(Lannou, Jean, 14.6)
... (27 de plus)
```

Exo 10 : Calcul de la moyenne des moyennes d'élèves.

```
SELECT AVG(moyenne)
FROM (SELECT AVG(colles.note) AS moyenne
        FROM colles JOIN eleves
            ON colles.eleve = eleves.ide
                GROUP BY eleve
     )

(12.928275862068968)
```

Exo 10 : Parmi tous les élèves, déterminer le nom de ceux ayant eu au moins 10 notes strictement sous la moyenne.

```
SELECT e.nom, COUNT(c.note) AS n
    FROM colles AS c JOIN eleves  AS e
    ON (c.eleve = e.ide)
    WHERE c.note < (SELECT AVG(note)    FROM  colles)
    GROUP BY c.eleve
    HAVING n >= 10
    ORDER BY n DESC

(Chabert, 16)
(Blanchard, 16)
... (23 de plus)
```


Exo 10 : Déterminer les élèves ayant eu une moyenne de 14 ou plus.

```
SELECT eleves.nom, AVG(colles.note) AS moyenne
    FROM colles  JOIN eleves
    ON (colles.eleve = eleves.ide)
    GROUP BY colles.eleve
    HAVING moyenne >= 14
    ORDER BY moyenne DESC

(Donedu, 14.8)
(Lannou, 14.6)
(Guérindon, 14.16)
... (1 de plus)
```

Exo 10 : Parmi tous les élèves, déterminer ceux ayant eu au moins 6 notes strictement supérieures à 18.

```
SELECT eleves.nom, COUNT(colles.note) AS nombre
    FROM colles  JOIN eleves
    ON (colles.eleve = eleves.ide)
    WHERE colles.note > 18
    GROUP BY colles.eleve
    HAVING nombre >= 6
    ORDER BY nombre DESC

(Donedu, 9)
(Lannou, 7)
(Guérindon, 7)
```

7 Exercice 11

Exo 11 : Dresser la liste des nombres de films par genre.

```
SELECT genre, COUNT(*) FROM film
    GROUP BY genre

(Drame, 6)
(Epouvante, 1)
(Espionnage, 1)
(Policier, 2)
(Western, 1)
```

Exo 11 : Déterminer combien de films différents ont été projetés à l'Astoria.

```
SELECT    COUNT(DISTINCT projection.idf)
    FROM projection JOIN cinema
        ON projection.idc = cinema.idc
            WHERE cinema.nom = 'Astoria'

(2)
```

Exo 11 : Déterminer combien de films différents ont été projetés à l'Astoria.

```
SELECT COUNT(*) FROM
    (SELECT DISTINCT projection.idf FROM
        cinema JOIN projection
            ON cinema.idc = projection.idc
            WHERE cinema.nom = 'Astoria'
    )

(2)
```

Exo 11 : Dresser la liste des acteurs (nom et prénom) en précisant pour chacun le nombre de films dans lesquels il a joué. La liste doit etre triée par odre alphabétique croissant des noms.

```
SELECT personne.nom, personne.prenom, COUNT(*) AS nbfilm
    FROM personne JOIN jouer
        ON personne.idp = jouer.ida
            GROUP BY personne.nom, personne.prenom
                ORDER BY personne.nom

(Arquette, Rosanna, 1)
(Bettany, Paul, 1)
(Eastwood, Clint, 1)
(Hunter, Holly, 1)
(Irons, Jeremy, 1)
(Kidman, Nicole, 1)
... (8 de plus)
```

Exo 11 : Déterminer le nombre de films réalisés par chacun des réalisateurs.

```
SELECT personne.nom, personne.prenom, COUNT(*) AS nbfilm
    FROM personne JOIN film
        ON personne.idp = film.idr
            GROUP BY personne.nom, personne.prenom
```



```
(Cronenberg, David, 2)
(Eastwood, Clint, 3)
(Glen, John, 1)
(Mendes, Sam, 1)
(Tarantino, Quentin, 1)
(Wayne, John, 1)
... (1 de plus)
```

Exo 11 : Dresser la liste des nombres de projections par cinéma.

```
SELECT cinema.nom, COUNT(*) AS nbprojections
    FROM projection JOIN cinema
        ON projection.idc = cinema.idc
            GROUP BY cinema.nom

(Astoria, 3)
(Comedia, 3)
(Pathé, 5)
(UGC, 14)
```

Exo 11 : Dresser la liste des nombres de films projetés par cinéma.

```
SELECT cinema.nom, COUNT(DISTINCT projection.idf) AS nbfilm
    FROM projection JOIN cinema
        ON projection.idc = cinema.idc
            GROUP BY cinema.nom

(Astoria, 2)
(Comedia, 2)
(Pathé, 5)
(UGC, 11)
```

Exo 11 : Déterminer le nombre de films réalisés par le réalisateur qui en a réalisé le plus.

```
SELECT MAX(nbfilm)
    FROM (SELECT COUNT(*) AS nbfilm
        FROM personne JOIN film
            ON personne.idp = film.idr
                GROUP BY personne.idp)

(3)
```

Exo 11 : Déterminer les réalisateurs (possibilité d'ex aequo) ayant réalisé le plus de films.

```
SELECT personne.nom, personne.prenom, COUNT(*) AS nbfilm
    FROM personne JOIN film
        ON personne.idp = film.idr
            GROUP BY personne.idp
                HAVING nbfilm = (
                SELECT MAX(nbfilm)  FROM (
                    SELECT COUNT(*) AS nbfilm
                        FROM personne JOIN film
                            ON personne.idp = film.idr
                                GROUP BY personne.idp
                                )
                        )
```

```
(Eastwood, Clint, 3)
```

Exo 11 : Déterminer les genres de films (possibilité d'ex aequo) qui comptent le moins de films.

```
SELECT genre, COUNT(*) AS nbfilm
    FROM film
        GROUP BY genre
            HAVING nbfilm = (
                SELECT MIN(nbfilm) FROM (
                    SELECT COUNT(*) AS nbfilm
                        FROM film
                            GROUP BY genre)
                    )
```

```
(Epouvante, 1)
(Espionnage, 1)
(Western, 1)
```

Exo 11 : Déterminer les cinémas qui ont projeté tous les films.

```
SELECT cinema.nom, COUNT(DISTINCT projection.idf) AS nbfilm
    FROM projection JOIN cinema
        ON projection.idc = cinema.idc
            GROUP BY cinema.nom
                HAVING nbfilm = (
                    SELECT COUNT(*) FROM film
                        )
```

```
(UGC, 11)
```

Exo 11 : Déterminer les acteurs qu'il a été possible de voir dans tous les cinémas.


```
SELECT personne.nom, personne.prenom, COUNT(DISTINCT projection.idc) AS nbcinema
    FROM personne JOIN jouer ON personne.idp = jouer.ida
        JOIN projection ON jouer.idf = projection.idf
            GROUP BY personne.nom, personne.prenom
                HAVING nbcinema = (
                        SELECT COUNT(*) FROM cinema
                        )
```

```
(L. Jackson, Samuel, 4)
(Travolta, John, 4)
(Willis, Bruce, 4)
```

Exo 11 : Déterminer le nombre de films réalisés par Lars von Trier.

```
SELECT COUNT(*)
    FROM personne JOIN film ON personne.idp = film.idr
        WHERE personne.prenom = 'Lars' AND personne.nom = 'von Trier'
```

```
(2)
```

Exo 11 : Déterminer les acteurs qui ont joué dans tous les films de Lars von Trier.

```
SELECT personne.prenom, personne.nom
    FROM personne JOIN jouer ON personne.idp = jouer.ida
        JOIN film ON film.idf = jouer.idf
            WHERE film.idr = (
                SELECT idp FROM personne
                    WHERE personne.prenom = 'Lars'
                        AND personne.nom = 'von Trier'
                            )
                    GROUP BY personne.idp
                        HAVING COUNT(*) = (
                            SELECT COUNT(*)
                                FROM personne JOIN film
                                    ON personne.idp = film.idr
                                    WHERE personne.prenom = 'Lars'
                                        AND personne.nom = 'von Trier'
                                            )
```

```
(Stellan, Skarsgard)
```

Exo 11 : Déterminer les acteurs qui ont joué dans tous les films de Lars von Trier

```
SELECT personne.prenom, personne.nom
    FROM (
        SELECT film.idf AS i FROM
            personne JOIN film
                ON personne.idp = film.idr
                    WHERE personne.nom = 'von Trier'
                        AND personne.prenom = 'Lars'
        )
        JOIN jouer ON jouer.idf = i
            JOIN personne ON personne.idp = jouer.ida
                GROUP BY personne.idp
                    HAVING COUNT(*) = (
                      SELECT COUNT(*)
                      FROM  film JOIN personne
                        ON film.idr = personne.idp
                            WHERE personne.nom = 'von Trier'
                                AND personne.prenom = 'Lars'
                        )
```

(Stellan, Skarsgard)

8 Exercice 12

```
import sqlite3

def creation_base_clubs():
    #connexion à la base
    base = sqlite3.connect("foot843.db")
    #création du curseur
    curseur = base.cursor()
    #eexécution de requetes sql
    curseur.execute("DROP TABLE IF EXISTS clubs")
    curseur.execute(
    """CREATE TABLE clubs (idc integer primary key,
                           nom text,
                           pays text)"""
                          )
    source = open('clubs.csv','r')
    idc = 1
    for ligne in source:
       #récupération de la ligne dans le fichier csv
       nom , pays  = ligne.rstrip().split(';')
       #insertion dans la table
       curseur.execute("INSERT INTO clubs VALUES (?,?,?)",(idc,nom,pays))
       idc += 1
    source.close()
```



```
    #pour enregistrer les modifications dans la base
    base.commit()
    #fermeture de la connexion à la base
    base.close()

"""
# Exercice 13
"""

def buts():
    """générateur aléatoire du nombre de buts par match
    Probas théoriques d'obtenir 0,1,2, ... buts :
    1/4,1/3,1/3,1/4**2,1/4**3,1/4**4 etc ..
    """
    import random
    alea = random.random()
    if alea >= 3/4:
        return 0
    elif alea > 5/12:
        return 1
    elif alea > 1/12:
        return 2
    k = 3
    while alea > 1/12*(1-1/4**(k-2)) :
        k += 1
    return k

def test(fonctionbut,n):
    """Test du générateur aléatoire de buts par match"""
    nbuts = [buts() for i in range(n)]  #echantillon de buts
    nbuts.sort()  #tri de l'échantillon
    probas = [1/4,1/3,1/3] + [1/4**k for k in range(2, 51)]
    b = nbuts[0] #premier but de l'échantillon
    effectif = 0
    for e in nbuts[1:]:
        if e == b:
            #cumul des valeurs de buts successivement égales
            effectif += 1
        else:
            print('%s but(s)'%b,
            'fréquence observée = %.8f et attendue=%.8f'%(effectif/n, probas[b]))
            b = e  #nouvelle valeur de but dans l'échantillon
            effectif = 1

"""
>>> test(buts,10**6)
0 but(s), fréquence observée = 0.25021300 et attendue=0.25000000
1 but(s), fréquence observée = 0.33398400 et attendue=0.33333333
2 but(s), fréquence observée = 0.33311000 et attendue=0.33333333
3 but(s), fréquence observée = 0.06198900 et attendue=0.06250000
```

```
4 but(s), fréquence observée = 0.01556700 et attendue=0.01562500
5 but(s), fréquence observée = 0.00381400 et attendue=0.00390625
6 but(s), fréquence observée = 0.00097700 et attendue=0.00097656
.....
"""

def creation_base_matchs(nmatchs):
    """idem mais avec les identifiants de clubs idc
    à la place des noms de clubs"""
    #import du module random
    import random
    #connexion à la base
    base = sqlite3.connect('foot843.db')
    #création du curseur
    curseur = base.cursor()
    #eexécution de requetes sql
    curseur.execute("DROP TABLE IF EXISTS matchs")
    curseur.execute(
    """CREATE TABLE matchs (
    idm integer primary key,
    eq1 integer,
    sc1  integer,
    eq2 integer,
    sc2 integer,
    annee integer,
    jour integer)"""
                           )
    curseur.execute("SELECT idc FROM clubs")
    listeclubs = curseur.fetchall()
    nclubs = len(listeclubs)
    idm = 1
    #peuplement de la base avec 100 matchs
    #aléatoires entre 1971 et 2014
    for i in range(nmatchs):
       #annee aléatoire entre 1971 et 2014
       annee = random.randint(1971,2014)
       #jour aléatoire entre 1 et 365
       jour = random.randint(1,365)
       #tirage de l'équipe1
       indexequipe1 = random.randint(0,len(listeclubs)-1)
       equipe1 = listeclubs[indexequipe1][0]
       #on sort l'équipe 1 de la liste des clubs pour tirer l'équipe 2
       listeclubs.pop(indexequipe1)
       #tirage de l'équipe2
       equipe2 = listeclubs[random.randint(0,len(listeclubs)-1)][0]
       #on remet l'équipe 1 dans la liste des clubs
       listeclubs.append((equipe1,))
       #insertion dans la table
       curseur.execute('INSERT INTO matchs VALUES (?,?,?,?,?,?,?)',
       (idm,equipe1,buts(),equipe2,buts(),annee,jour))
```



```
       idm += 1
    #pour enregistrer les modifications réalisées dans la base
    base.commit()
    base.close()

""">>> creation_base_matchs(100)"""
```

9 Exercice 14

Exo 14 : Scores de tous les matchs disputés par le Real Madrid.

```
SELECT c1.nom, sc1, c2.nom, sc2
   FROM clubs AS c1 JOIN matchs  JOIN clubs AS c2
      ON c1.idc = eq1 AND c2.idc=eq2
         WHERE c1.nom='Real Madrid' OR c2.nom='Real Madrid'

(Real Madrid, 1, Juventus de Turin, 3)
(Bayern Munich, 1, Real Madrid, 1)
(Olympique de Marseille, 1, Real Madrid, 2)
(Paris Saint-Germain, 1, Real Madrid, 1)
... (162 de plus)
```

Exo 14 : Matchs où le Paris Saint-Germain a mis strictement plus de un but à l'extérieur sans gagner.

```
SELECT   c1.nom, sc1, c2.nom, sc2
    FROM clubs c1 JOIN matchs JOIN clubs c2
        ON c1.idc = eq1 AND c2.idc=eq2
            WHERE c2.nom='Paris Saint-Germain' AND sc2>1 AND sc1>sc2

(En Avant Guingamp, 3, Paris Saint-Germain, 2)
(Manchester City, 4, Paris Saint-Germain, 2)
(Real Madrid, 3, Paris Saint-Germain, 2)
(Milan AC, 3, Paris Saint-Germain, 2)
```

Exo 14 : Combien de fois un club français a-t-il marqué au moins 4 buts contre un club italien ?

```
SELECT   COUNT(*)
    FROM clubs AS c1 JOIN matchs JOIN clubs AS c2
        ON c1.idc = eq1 AND c2.idc=eq2
            WHERE (c1.pays='ITA'  AND c2.pays='FRA' AND sc2>=4)
                OR (c1.pays='FRA'  AND c2.pays='ITA' AND sc1>=4)

(2)
```