BDW1 - TD 1
Des requêtes SQL

UCBL - Département Informatique de Lyon 1 – 2018

L'objectif de ce TD est de vous familiariser avec les premiers éléments de la syntaxe SQL.

Exercice 1 : Base MAGASINS

Soit le schéma de la relation MAGASINS suivant :
Magasin(idM, Nom, Categorie, Ville, Gerant)

idM	Nom	Categorie	Ville	Gerant
1	TASCORAMA	Bricolage	Bron	Yvan Declou
2	TASCORAMA	Bricolage	Rillieux-la-Pape	Eddy Visse
3	LECIO	Textile	Lyon	Ella Laclasse
4	SAPEY	Textile	Lyon	Côme Jamet
5	LOUBANGER	Informatique	Lyon	Yann Apadebeugue
6	VBH	Bricolage	Paris	Elvis Tourne

a) Exprimer en langage naturel ce que retournent les requêtes suivantes :

1.
```
SELECT Nom FROM Magasin;
```

2.
```
SELECT DISTINCT Categorie
FROM Magasin WHERE Ville = 'Lyon';
```

3.
```
SELECT *
FROM Magasin
WHERE Ville != 'Lyon' AND Categorie = 'Bricolage'
ORDER BY Nom;
```

4.
```
SELECT DISTINCT M1.idM, M2.idM
FROM Magasin M1 JOIN Magasin M2
        ON M1.Ville = M2.Ville AND M1.idM != M2.idM;
```

b) Quelle modification apporteriez-vous à la dernière requête pour éviter d'avoir les pairs symétriques dans le résultat (*par exemple :* (3,4) et (4,3)) ?

Exercice 2 : Base JOUEURS

Soient les schémas de la relation JOUEUR et PALMARES suivant :

JOUEUR(Nom, Prenom, AnNaiss, Nation, Taille, Poids)
PALMARES(#Nom, Annee, Titre)

Donner les requêtes SQL permettant de répondre aux besoins suivants :

1. Donner le nom et le prénom des joueurs.
2. Donner les joueurs nés avant 1987 (inclus) ou ceux faisant plus de 78kg (strict).
3. Donner le nom et le prénom des joueurs ayant eu un palmarès en 2014.
4. Donner le nom et le palmarès (Titre et année) des joueurs nés la même année que le joueur Emile Maitre.
5. Donner le nom des joueurs titrés en 2010 et 2014.

Exercice 3 : Base UNIVERSITE

Soit le schéma de la base de données UNIVERSITE suivant :

Etudiant (NumEt, NomEt, PrenomEt, Age, Ville, Statut)
UE (NumUE, NomUE , nbECTS)
Inscription (#NumEt, #NumUE, Semestre, Annee, NoteFinale)

La relation *Etudiant* donne pour chaque étudiant son numéro d'étudiant, son nom, son prénom et son âge, la ville où il habite et son statut (célibataire, marié(e) ...)
La relation *UE* donne pour chaque UE son numéro, son nom et le nombre d'ECTS associés.
La relation *Inscription* fait le lien entre les étudiants, les UE qu'ils suivent durant un semestre (S1 pour le semestre automne ou S2 pour le semestre printemps) d'une année pendant lesquelles ils ont suivi cette UE et la note finale (correspondant à la moyenne) qu'ils ont obtenue.

Donner les requêtes SQL permettant de répondre aux besoins suivants :

1. Donner le nom des UE dont on ne connaît pas le nombre d'ECTS associés.
2. Donner le numéro des UE dont le nom commence par 'BDW' ou contenant les caractères 'info'.
3. Donner le numéro des étudiants dont le nom a pour troisième caractère un 'M'.
4. Donner le nom de toutes les UE suivies par l'étudiant "Jean Vidaprendre". On précisera pour chaque UE, le nombre d'ECTS associés, ainsi que le semestre et l'année d'inscription. Le résultat sera trié par année décroissante, puis par semestre, puis par ordre lexicographique sur les noms d'UE.
5. Donner le nom des UE de 6 ECTS ouvertes (*i.e.* avec des inscrits) aux deux semestres de l'année universitaire 2017-2018.
6. Donner le nom et le prénom des étudiants ayant validé (i.e. note finale supérieure à 10) l'UE BDW1 au premier semestre de l'année universitaire 2014-2015. Le résultat sera trié par ordre lexicographique.

Exercice 4 : Base TRAIN

On considère le schéma de base de données suivant :

LIGNE *(NoLigne, VilleDep, VilleArr)*
ARRET *(#NoLigne, Rang, Ville)*
TRAFIC *(NoTrain, #NoLigne, NoJour)*
TRAIN *(#NoTrain, #NoWagon)*
WAGON *(NoWagon, TypeWagon, PoidsVide, Capacite, Etat)*

La relation LIGNE donne pour chaque ligne de train la gare de départ et la gare d'arrivée. La relation ARRET donne le rang et la ville de l'arrêt pour une ligne donnée. La relation TRAFIC associe les trains et les lignes parcourues par ces trains à un numéro de jour de semaine donné (1 pour lundi, 2 pour mardi...). La relation TRAIN liste les wagons qui composent un train. La relation WAGON donne les caractéristiques de chaque wagon (l'état peut être "libre" ou "occupé"). Le poids est exprimé en kg.

Exemple d'instance de base :

Pour LIGNE : {(1,'Lyon','Clermont-Ferrand'), (2,'Lyon', 'Marseille'), ... }
Pour ARRET : {(1,1,'Gannat'), (2,1,'Valence'), (2,2, 'Avignon'), ... }
Pour TRAFIC : {(1,1,1),(1,1,2),(1,1,3),(1,1,4), (1,1,5),(2,2,6),(2,2,7), ... }
Pour TRAIN : {(1,1), (1,3), (1,4), (1,5), (2,1), (2,10), ... }
Pour WAGON : {(1,'Passagers', NULL, 150, 'Libre'),
(2,'Marchandise', 12345, NULL, 'Occupé'), ... }

Donner les requêtes SQL permettant de répondre aux questions suivantes :

1. Donner la liste des gares traversées par la ligne 3. Le résultat sera renommé 'GareTraverseeParL3' et sera trié selon le rang de l'arrêt.
2. Donner les numéros des wagons faisant à vide entre 800kg et 2tonnes et de type 'Marchandise'.
3. Donner les wagons dont on ne connaît pas soit le poids à vide soit la capacité.
4. Donner les numéros des lignes partant de Lyon et pour lesquelles au moins un train circule le mercredi.
5. Donner les wagons de type 'Marchandise' de capacité inférieure strictement à celle du wagon 12.
6. Donner les couples de numéros de wagons qui sont libres, de même type, de même poids à vide mais de capacités différentes.
7. Donner le numéro des train qui sont composés d'au moins deux wagons de type 'Passagers' et qui partent de 'Dijon' ou arrivent à 'Valence'.