Exercice 1 : On considère la base de données contenant les tables ci-dessous (dont on donne un *extrait* des contenus)

Joueur

numero	nom	prenom	role
1	*Ruffier*	*Stéphane*	*gardien*
2	*Ghoulam*	*Faouzi*	*défenseur*
3	*Roux*	*Nolan*	*attaquant*
4	*Gameiro*	*Kevin*	*attaquant*
5	*Sirigu*	*Salvatore*	*gardien*
…	…	…	…

JoueDans

numJoueur	numClub
1	*1*
2	*1*
3	*2*
4	*3*
…	…

Club

numero	Nom	ville	anneeCreation
1	*ASSE*	*Saint-Etienne*	*1933*
2	*LOSC*	*Lille*	*1944*
3	*PSG*	*Paris*	*1970*

Spectateur

numero	nom	prenom	age
1	*Dupont*	*Pierre*	*24*
2	*Dupont*	*Jacques*	*46*
3	*Smith*	*Sophie*	*20*

Stade

numero	nom	ville	nbPlaces	etat
1	*Geoffroy Guichard*	*Saint-Etienne*	*35616*	*bon*
2	*Parc des Princes*	*Paris*	*47428*	*moyen*
3	*Stadium Nord*	*Lille*	*17963*	*mauvais*

Match

numero	numClub1	numClub2	numStade	date	nbSpectateur
1	*1*	*2*	*1*	*11/08/2012 21:00:00*	*23543*
2	*3*	*1*	*3*	*03/11/2012 17:00:00*	
…	…	…	…	…	…

Assiste

numSpectateur	numMatch
1	*2*
3	*1*
…	…

Selectionne

numJoueur	numMatch
1	*1*
3	*5*
…	…

Les phrases ci-dessous donnent la signification du premier tuple de chaque table.
Joueur : Le joueur numéro 1 s'appelle Stéphane Ruffier, il est gardien de but et a 26 ans.
Club : Le club numéro 1 s'appelle ASSE, il est localisé à Saint-Etienne et a été créé en 1933.
JoueDans : Le joueur numéro 1 joue dans le club numéro 1.
Spectateur : Le spectateur numéro 1 s'appelle Pierre Dupont et a 24 ans.
Stade : Le stade numéro 1 s'appelle « Geoffroy Guichard », il est localisé à Saint-Etienne, compte 35616 places et est aujourd'hui en bon état.
Match : Le club numéro 1 a rencontré le club numéro 2 dans le stade numéro 1 le 10 septembre 2011 à 19h et il y avait 35543 spectateurs.
Assiste : Le spectateur numéro 1 a assisté au match numéro 2.
Selectionne : Le joueur numéro 1 a été sélectionné pour jouer dans le match numéro 1.

Donnez les **instructions SQL** permettant de :

1. Créer les trois tables Joueur, Club, JoueDans sans préciser les clés primaires ni les contraintes d'intégrité référentielle ni aucune autre contrainte d'intégrité. On fera toutefois en sorte que les numéros de joueur et de club s'incrémentent automatiquement dans les tables Joueur et Club.

```
CREATE TABLE Joueur (
      numero TINYINT AUTO_INCREMENT ,
      nom VARCHAR(30) ,
      prenom VARCHAR(30) ,
      role VARCHAR(20) ) ;

CREATE TABLE Club (
      numero TINYINT AUTO_INCREMENT ,
      nom VARCHAR(30) ,
      ville VARCHAR(30) ,
      anneeCreation YEAR ) ;

CREATE TABLE JoueDans (
      numJoueur TINYINT ,
      numClub TINYINT ) ;
```

2. Définir les clés primaires qui vous semble les plus judicieuses pour chacune des trois tables créées à la question précédente.

```
ALTER TABLE Joueur ADD ( PRIMARY KEY (numero) ) ;
ALTER TABLE Club ADD ( PRIMARY KEY (numero) ) ;
ALTER TABLE JoueDans ADD ( PRIMARY KEY (numJoueur) ) ;
```
on aurait pu écrire ainsi également :
```
ALTER TABLE Joueur MODIFY ( numero PRIMARY KEY ) ;
ALTER TABLE Club MODIFY ( numero PRIMARY KEY ) ;
ALTER TABLE JoueDans MODIFY ( numJoueur PRIMARY KEY ) ;
```

Si on mémorise les diverses affectations des joueurs dans divers clubs au cours du temps, on pourrait avoir plusieurs fois le même numéro de joueur qui apparaitrait dans la colonne numJoueur, alors il faudrait définir la clé ainsi :

```
ALTER TABLE JoueDans ADD PRIMARY KEY (numJoueur, numClub) ;
```

3. Dans la (ou les) table(s) précédente(s) où cela vous semble utile, ajouter des contraintes d'intégrité référentielle.

Il n'y a que dans la table JoueDans qu'il est nécessaire d'ajouter des contraintes d'intégrité :

```
ALTER TABLE JoueDans ADD (
                        FOREIGN KEY (numJoueur) REFERENCES Joueur(numero) ,
                        FOREIGN KEY (numClub) REFERENCES Club(numero)
                          ) ;
ou (pas en MySQL)
ALTER TABLE JoueDans MODIFY (
                        numJoueur REFERENCES Joueur(numero) ,
                        numClub REFERENCES Club(numero)
                             ) ;
```

4. Ajouter un attribut «age» (de type INT) dans la table Joueur.

```
ALTER TABLE Joueur ADD ( age INT ) ;
```

5. Modifier le type de l'attribut «age» de la table Joueur pour qu'il soit TINYINT.

```
ALTER TABLE Joueur MODIFY ( age TINYINT ) ;
```

6. Ajouter une contrainte sur l'attribut «age» de la table Spectateur : ses valeurs doivent être comprises entre 5 et 100.

```
ALTER TABLE Spectateur MODIFY (age CHECK (age BETWEEN 5 AND 100) ) ;
```
ou
```
ALTER TABLE Spectateur ADD (CHECK (age BETWEEN 5 AND 100) ) ;
```

7. Supprimer l'attribut «etat» de la table Stade (en supposant la table créée).

```
ALTER TABLE Stade DROP COLUMN etat ;
```

8. Dans la table Joueur, ajouter un attribut «dateDeNaissance», permettant de mémoriser la date de naissance d'un joueur.

```
ALTER TABLE Joueur ADD (dateDeNaissance DATE) ;
```

9. Insérer, dans la table Stade, les valeurs de la première ligne indiquée ci-dessus.

```
INSERT INTO Stade VALUES (1,'Geaoffroy Guichard', 'Saint-Etienne', 35616, 'bon') ;
```

10. Insérer, dans la table Stade, un enregistrement contenant le nom et la ville d'un stade : 'Stadium des champions', 'Trifouillis les oies', le nombre de places et l'état n'étant pas connus.

```
INSERT INTO Stade (nom,ville) VALUES ('Stadium des champions', 'Trifouillis les oies') ;
```

11. Donner l'instruction permettant, dans la table Stade, de modifier l'état du Parc des Princes pour le faire passer à l'état « bon ».

```
UPDATE Stade SET etat='bon' WHERE nom='Parc des Princes' ;
```

12. Supprimer de la table Match tous les matchs ayant eu lieu en 2011.

```
DELETE FROM Match WHERE year(date)=2011
```

Exercice 2 *(création de tables)* : On considère les schémas, illustrés par les tables suivantes :

Etudiant				
NomE	**PrenomE**	**DateNaissanceE**	**AnneePremInscE**	**AnneeEtude**
Dupont	*Jean*	*23/12/1985*	*2003*	*3*
…	…	…	…	…

Cours			
NomC	**AnneeC**	**JourC**	**SalleC**
Bases de Données	*1*	*5*	*F101*
…	…	…	…

1. Donnez les instructions SQL permettant de :
   a. créer ces tables.

```
CREATE TABLE Etudiant (
      NomE VARCHAR(30) ,
      PrenomE VARCHAR(20) ,
      DateNaissanceE DATE ,
      AnneePremInscE    YEAR ,
      AnneeEtude TINYINT ) ;

CREATE TABLE Cours (
      NomC VARCHAR(40) ,
      AnneeC TINYINT ,
      JourC TINYINT ,
      SalleC VARCHAR(5) ) ;
```

   b. ajouter des informations sur l'adresse de chaque étudiant.

```
ALTER TABLE Etudiant
      ADD (
            NumRue INT ,
            NomRue VARCHAR(40)
            CodePostal INT ,
            NomVille VARCHAR(30) ) ;
```

   c. ajouter la contrainte permettant de s'assurer que l'année de première inscription est postérieure à 2000

```
ALTER TABLE Etudiant
      MODIFY (
            AnneePremInscE CHECK(AnneePremInscE > 2000) ) ;
```

   d. Supprimer l'année de première inscription.

```
ALTER TABLE Etudiant
      DROP COLUMN AnneePremInscE ;
```

   e. Renommer la table Etudiant en EtudiantsUJM.

```
ALTER TABLE Etudiant RENAME TO EtudiantsUJM ;
```

2. On souhaite créer une clé sur la relation *Etudiant*. On suppose qu'il n'y aura jamais deux étudiants de même nom et même prénom. Donnez une instruction SQL permettant de créer cette clé.

```
ALTER TABLE Etudiant ADD PRIMARY KEY (NomE, PrenomE)
```

3. On suppose maintenant qu'il est possible que deux étudiants de la même année d'étude aient le même nom, prénom, date de naissance et se soient inscrits pour la première fois la même année. Donnez une instruction SQL permettant de créer une clé dans ce contexte.

```
ALTER TABLE Etudiant
      ADD (IdEtudiant INT PRIMARY KEY AUTO_INCREMENT) ;
```

4. On souhaite créer une clé sur la table Cours. Donnez une instruction SQL permettant de créer celle-ci.

```
ALTER TABLE Cours ADD (IdCours INT PRIMARY KEY AUTO_INCREMENT) ;
```

Remarque à faire aux étudiants : on aurait pu prendre le couple (NomC, AnneeC) comme clé, mais il peut être possible d'avoir un même nom de cours dans une même année mais de filière différente. La solution proposée est donc ainsi valable dans tous les cas. C'est bien souvent qu'on introduit une clé qui à la forme d'un numéro de quelque chose.

5. On souhaite pouvoir mémoriser plusieurs prénoms pour chaque étudiant. Proposer une façon de représenter cela.

Etudiant				
IdEtudiant	**NomE**	**DateNaissanceE**	**AnneePremInscE**	**AnneeEtude**
1	*Dupont*	*23/12/1985*	*2003*	*3*
…	**…**	**…**	**…**	**…**

PrenomsEtudiant	
IdEtu	**PrenomE**
1	*Jean*
1	*Paul*
…	**…**

```
CREATE TABLE PrenomsEtudiant(
      IdEtu INT REFERENCES Etudiant(IdEtudiant) ,     (Pas possible avec MySQL)
      PrenomE VARCHAR(20) ) ;
```

autre façon d'écrire (notation obligatoire en MySQL) :

```
CREATE TABLE PrenomsEtudiant(
      IdEtu INT ,
      PrenomE VARCHAR(20) ,
      FOREIGN KEY (IdEtu) REFERENCES Etudiant(IdEtudiant)  ) ;
```

6. On souhaite maintenant ordonner les prénoms. Proposez une façon de représenter cela.

PrenomsEtudiant		
IdEtu	**Rang**	**PrenomE**
1	*1*	*Jean*
1	*2*	*Paul*
2	*1*	*Marie*
2	*2*	*Isabelle*
2	*3*	*Chloé*
…	**…**	**…**

```
CREATE TABLE PrenomsEtudiant (
      IdEtu INT REFERENCES Etudiant(IdEtudiant) ,     Pas possible avec MySQL
      Rang TINYINT ,
      PrenomE VARCHAR(20) ) ;
```

autre façon d'écrire (écriture obligatoire en MySQL) :

```
CREATE TABLE PrenomsEtudiant (
      IdEtu INT ,
      Rang TINYINT ,
      PrenomE VARCHAR(20) ,
      FOREIGN KEY (IdEtu) REFERENCES Etudiant(IdEtudiant) ) ;
```

7. A l'aide d'une nouvelle table, on voudrait maintenant stocker, pour chaque étudiant, la liste des cours qu'il suit. Donnez l'instruction SQL permettant de créer cette table en prenant soin de ne pas oublier la clé de celle-ci.

```
CREATE TABLE EtudiantSuitCours (
      NumEtudiant INT REFERENCES Etudiant (IdEtudiant) , Pas possible avec MySQL
      NumCours INT REFERENCES Cours (IdCours) ) ;        Pas possible avec MySQL
```

autre façon d'écrire (écriture obligatoire en MySQL) :

```
CREATE TABLE EtudiantSuitCours (
      NumEtudiant INT,
      NumCours INT ,
      FOREIGN KEY (NumEtudiant) REFERENCES Etudiant (IdEtudiant) ,
      FOREIGN KEY (NumCourst) REFERENCES Cours (IdCours) ) ;
```

8. Insérez les informations ci-dessus des tables *Etudiant* et *Cours*. Mémorisez le fait que Dupont suit le cours de Bases de Données.

```
INSERT INTO Etudiant VALUES ('Dupont', 'Jean', 1985/12/23, 2003, 3) ;
INSERT INTO Cours VALUES ('Bases de Données', 1, 5, 'F101') ;
INSERT INTO EtudiantSuitCours VALUES (1,  1) ;
```

On considère les tables ci-dessous.

Auto			
IdAuto	**Marque**	**Modèle**	**Puissance**
1	Citroen	C5	7
2	Peugeot	1007	4
3	Fiat	Punto	4
4	Opel	Meriva	6
5	Ford	Mondeo	7
6	Seat	Ibiza	6
7	Citroen	C3	4

PaysConstructeur	
Marque	**Pays**
Opel	Allemagne
Seat	Espagne
Citroen	France
Peugeot	France
Fiat	Italie
Ford	USA

InfoPays	
Pays	**Langue**
Allemagne	Allemand
Angleterre	Anglais
Espagne	Espagnol
France	Français
Italie	Italien
USA	Anglais

Donnez les instructions SQL permettant de créer ces tables. (pensez aux clés primaires et étrangères).
Donnez le résultat des requêtes ci-dessous. Traduisez chaque d'elles en langage naturel.

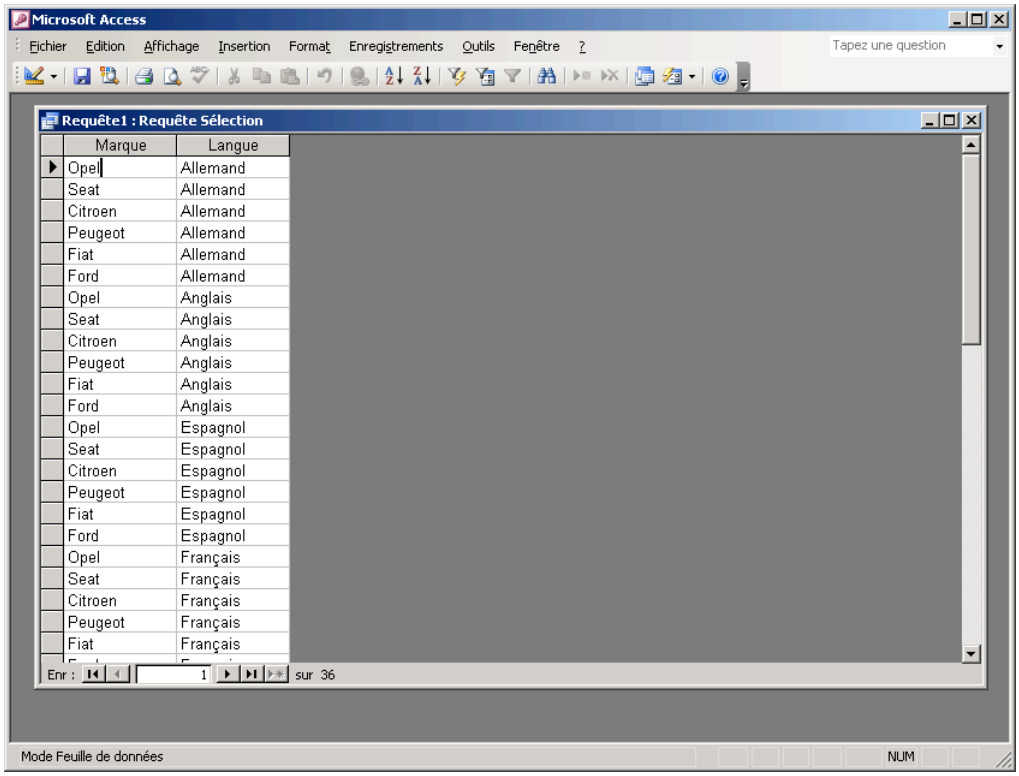
SELECT * **FROM** Auto ;
(Donnez tous les renseignements sur les enregistrements (lignes) de la table Auto)

SELECT Marque **FROM** Auto ;
(parler de **projection**)
(Donnez toutes les marques présentent dans la table Auto)

SELECT DISTINCT Marque **FROM** Auto ;
(Donnez toutes les marques présentent dans la table Auto, chaque marque n'apparaissant qu'une seule fois)

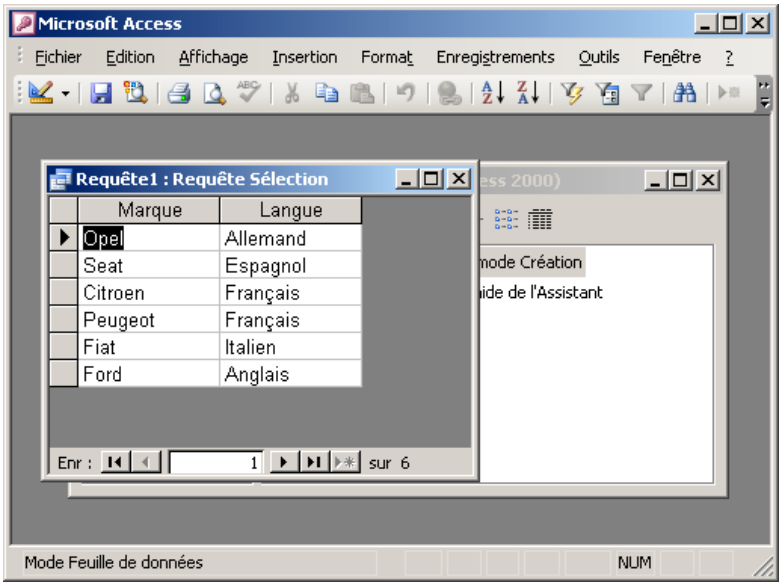
SELECT * **FROM** Auto **WHERE** Puissance = 4 ;
(parler de projection et **sélection**)
(Donnez tous les renseignements sur les voitures de 4 chevaux)

SELECT Marque, Langue **FROM** PaysConstructeur, InfoPays ;
(dire qu'il s'agit d'un **produit cartésien**)
(cette requête n'a pas un sens intelligible)

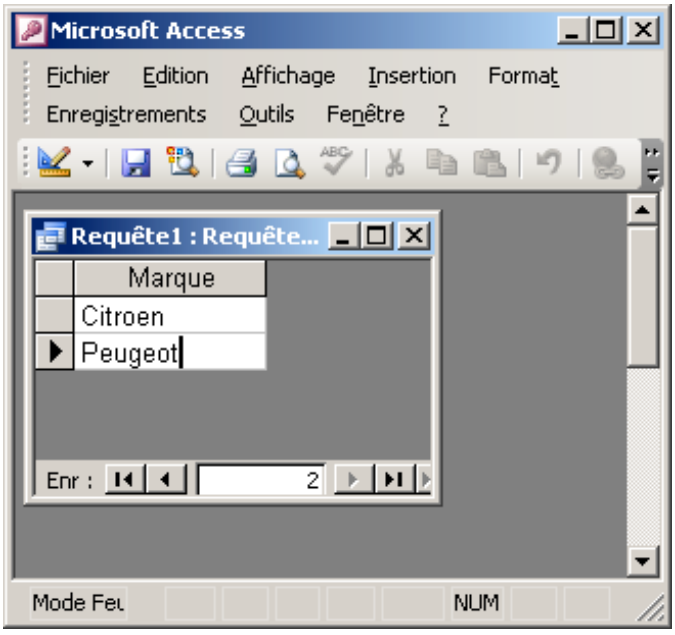



```
SELECT Marque, Langue
FROM PaysConstructeur, InfoPays
WHERE PaysConstructeur.Pays = InfoPays.Pays
```

(dire qu'il s'agit d'une **jointure**)
(Donnez toutes les marques de tous les constructeurs ainsi que les langues des pays d'origine de ceux-ci)

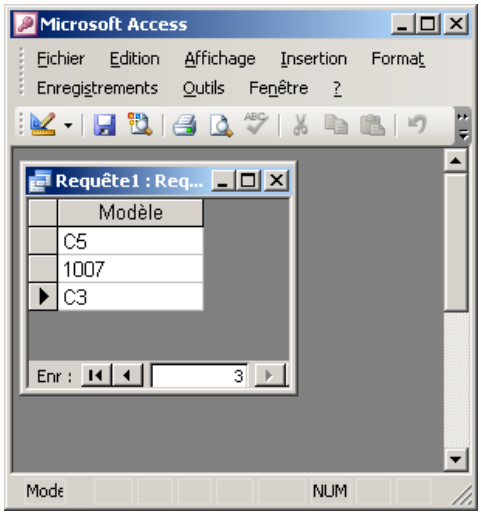


```
SELECT Marque
FROM PaysConstructeur, InfoPays
WHERE PaysConstructeur.Pays = InfoPays.Pays
AND Langue LIKE 'Français' ;
```
(dire qu'il s'agit d'une **jointure**)
(Donnez les marques des constructeurs des pays où l'on parle Français)



Ecrire une requête équivalente en utilisant des requêtes imbriquées.
Solution :
```
SELECT Marque
FROM PaysConstructeur
WHERE Pays IN (SELECT Pays
               FROM InfoPays
               WHERE Langue LIKE 'Français')
```
(Dire qu'il s'agit **également** d'une **jointure)**

```
SELECT Modèle
FROM Auto, PaysConstructeur, InfoPays  (bien expliquer pourquoi on
a besoin des 3 tables)
WHERE Auto.Marque = PaysConstructeur.Marque
  AND PaysConstructeur.Pays = InfoPays.Pays
  AND Langue LIKE 'Français'
```
(Donnez les modèles des marques des constructeurs des pays où l'on parle Français)



Ecrire une requête équivalente en utilisant des requêtes imbriquées.
Solution :

```
SELECT Modèle
FROM Auto
WHERE Marque IN
      (SELECT Marque
       FROM PaysConstructeur
       WHERE Pays IN
           (SELECT Pays
            FROM InfoPays
            WHERE Langue LIKE 'Français') ) ;
```

(Dire qu'il s'agit **également** d'une **jointure**)

Ecrire en SQL les requêtes suivantes :

- Quels sont les modèles d'une puissance supérieure ou égale à 6 CV ?

```
SELECT Modèle FROM Auto WHERE Puissance >= 6 ;
```

- Quelles sont les marques originaires de France ?

```
SELECT Marque FROM PaysConstructeur WHERE Pays LIKE 'France' ;
```

- Quels sont les pays où l'on parle anglais ?

```
SELECT Pays FROM InfoPays WHERE Langue LIKE 'Anglais' ;
```

- Quelles sont les puissances des voitures de constructeurs originaires de France ?

```
SELECT Puissance
FROM Auto, PaysConstructeur
WHERE Auto.Marque = PaysConstructeur.Marque
  AND PaysConstructeur.Pays = 'France' ;
```

- Quelles sont les puissances des voitures de constructeurs originaires de pays où l'on parle anglais ?

```
SELECT Puissance
FROM Auto, PaysConstructeur
WHERE Auto.Marque = PaysConstructeur.Marque
  AND PaysConstructeur.Pays = InfoPays.Pays
  AND InfoPays.Langue LIKE 'Anglais' ;
```