SQL : Langage d’interrogation de Données

Afpa - Stéphane PONTONNIER 2022

Plan

- Introduction
- Requêtes simples
  - SELECT … FROM …
  - LIMIT et OFFSET
  - DISTINCT
  - COUNT
  - WHERE
  - BETWEEN
  - IN
  - NOT
  - LIKE
  - IS
  - IFNULL
  - ORDER BY
  - GROUP BY
  - HAVING
  - UNION

Plan

- Jointures
  - Jointure implicite
  - Jointure explicite
    - JOIN … ON
    - LEFT JOIN … ON
    - RIGHT JOIN … ON

Plan

- Requêtes imbriquées
  - WHERE … IN … SELECT
  - WHERE … ANY … SELECT
  - WHERE EXISTS … SELECT
  - FROM … SELECT … AS
- Fonctions sur les chaînes de caractères

Introduction

SQL

SQL

- Langage de définition de données
- Langage de manipulation de données
- Langage de contrôle de données
- **Langage d'intérrogation de données**

Langage d'interrogation de données

- Langage qui permet de lire des données stockées dans la BD
- Utilisant des concepts connus en algèbre relationnel : projection, jointure, intersection, union, produit cartésien…

Introduction

Considérons la base de données poei **contenant les deux tables suivantes**

personne				
num	nom	prenom	salaire	ville
1	Cohen	Sophie	2000	Marseille
2	Leberre	Bernard	1500	Marseille
3	Benabar	Pierre	1800	Lyon
4	Hadad	Karim	2500	Paris
5	Wick	John	3000	Paris

num : la clé primaire

Introduction

vehicule				
immatriculation	marque	modele	annee	nump
100	Peugeot	5008	2018	5
200	Renault	clio	2000	4
300	Ford	fiesta	2010	1
400	Peugeot	106	2002	3
500	Citroen	C4	2015	4
600	Ford	Kuga	2019	
700	Fiat	punto	2008	5

immatriculation : la clé primaire
nump : clé étrangère

Introduction

SQL

Script de la création de la base de données. Dans un terminal (shell) :

```
CREATE DATABASE poei;
USE poei;

CREATE TABLE personne (
        num INT (3) PRIMARY KEY,
        nom VARCHAR (20),
        prenom VARCHAR (20),
        salaire INT(4),
        ville VARCHAR (20)
);
```

A vous de créer la table vehicule

Introduction

SQL

Script de la création de la base de données. Dans un terminal (shell) :

```
CREATE TABLE vehicule (
        immatriculation INT (3) PRIMARY KEY,
        marque VARCHAR (20),
        modele VARCHAR (20),
        annee INT(4),
        nump INT (3),
        CONSTRAINT fk_vehicule_personne FOREIGN KEY (nump) REFERENCES personne (num)
);
```

Introduction

SQL

Script d'insertion de données. Dans un terminal (shell) :

```
INSERT INTO personne VALUES
(1, 'Cohen', 'Sophie', 2000, 'Marseille'),
(2, 'Leberre', 'Bernard', 1500, 'Marseille'),
(3, 'Benabar', 'Pierre', 1800, 'Lyon'),
(4, 'Hadad', 'Karim', 2500, 'Paris'),
(5, 'Wick', 'John', 3000, 'Paris');

INSERT INTO vehicule VALUES
(100, 'Peugeot', '5008', 2018, 5),
(200, 'Renault', 'Clio', 2000, 4),
(300, 'Ford', 'Fiesta', 2010, 1),
(400, 'Peugeot', '106', 2002, 3),
(500, 'Citroen', 'C4', 2015, 4),
(600, 'Ford', 'Kuga', 2019, null),
(700, 'Fiat', 'Punto', 2008, 5);
```

Requêtes simples

SQL

Remarque

Une requête SQL de lecture est composée d'au moins deux clauses

Quelques clauses possibles

- **SELECT** : les colonnes à sélectionner
- **FROM** : les tables concernées
- **WHERE** : les conditions
- …

Requêtes simples

SQL : SELECT … FROM …

Pour sélectionner toutes les données de la table personne
SELECT * **FROM** personne;

Le résultat :

```
MariaDB [poei]> select * from personne;
+-----+---------+---------+---------+-----------+
| num | nom     | prenom  | salaire | ville     |
+-----+---------+---------+---------+-----------+
|   1 | Cohen   | Sophie  |    2000 | Marseille |
|   2 | Leberre | Bernard |    1500 | Marseille |
|   3 | Benabar | Pierre  |    1800 | Lyon      |
|   4 | Hadad   | Karim   |    2500 | Paris     |
|   5 | Wick    | John    |    3000 | Paris     |
+-----+---------+---------+---------+-----------+
5 rows in set (0.000 sec)
```

Requêtes simples

SQL : LIMIT & OFFSET

Et si on veut sélectionner que les 2 premières personnes
SELECT * **FROM** personne **LIMIT** 2;

Le résultat :

```
MariaDB [poei]> select * from personne limit 2;
+-----+---------+---------+---------+-----------+
| num | nom     | prenom  | salaire | ville     |
+-----+---------+---------+---------+-----------+
|   1 | Cohen   | Sophie  |    2000 | Marseille |
|   2 | Leberre | Bernard |    1500 | Marseille |
+-----+---------+---------+---------+-----------+
2 rows in set (0.000 sec)
```

Requêtes simples

SQL : LIMIT & OFFSET

Et si on veut sélectionner les 2 personnes suivantes
SELECT * **FROM** personne **LIMIT** 2 **OFFSET** 2;

Le résultat :

```
MariaDB [poei]> select * from personne limit 2 offset 2;
+-----+---------+--------+---------+-------+
| num | nom     | prenom | salaire | ville |
+-----+---------+--------+---------+-------+
|   3 | Benabar | Pierre |    1800 | Lyon  |
|   4 | Hadad   | Karim  |    2500 | Paris |
+-----+---------+--------+---------+-------+
2 rows in set (0.000 sec)
```

Requêtes simples

SQL : LIMIT & OFFSET

On peut aussi simplifier l'écriture précédente
SELECT * **FROM** personne **LIMIT** 2, 2;

Le résultat est le même :

```
MariaDB [poei]> select * from personne limit 2, 2;
+-----+---------+--------+---------+-------+
| num | nom     | prenom | salaire | ville |
+-----+---------+--------+---------+-------+
|   3 | Benabar | Pierre |    1800 | Lyon  |
|   4 | Hadad   | Karim  |    2500 | Paris |
+-----+---------+--------+---------+-------+
2 rows in set (0.000 sec)
```

Requêtes simples

SQL : LIMIT & OFFSET

Pour filtrer les colonnes (ici la ville de chaque personne)
SELECT ville **FROM** personne;

Le résultat :

```
MariaDB [poei]> select ville from personne;
+-----------+
| ville     |
+-----------+
| Marseille |
| Marseille |
| Lyon      |
| Paris     |
| Paris     |
+-----------+
5 rows in set (0.000 sec)
```

Comment faire pour supprimer les doublons ?

Requêtes simples

SQL : DISTINCT

Pour supprimer les doublons
SELECT DISTINCT(ville) **FROM** personne;

Le résultat :

```
MariaDB [poei]> select distinct(ville) from personne;
+-----------+
| ville     |
+-----------+
| Marseille |
| Lyon      |
| Paris     |
+-----------+
3 rows in set (0.001 sec)
```

Requêtes simples

SQL : COUNT

Pour compter le nombre de ville dans la table personne
SELECT COUNT(**DISTINCT**(ville)) **FROM** personne;

Le résultat :

```
MariaDB [poei]> SELECT COUNT(DISTINCT(ville)) FROM personne;
+------------------------+
| COUNT(DISTINCT(ville)) |
+------------------------+
|                      3 |
+------------------------+
1 row in set (0.002 sec)
```

Remarque

COUNT compte les tuples (les lignes) et pas les colonnes

Question

Comment remplacer le nom de la colonne COUNT(DISTINCT(ville)) **par un autre personnalisé ?**

Requêtes simples

SQL : COUNT

Pour modifier le nom d'une colonne dans l'affcihage du résultat
SELECT COUNT(**DISTINCT**(ville)) **AS** nombre_ville **FROM** personne;

Le résultat :

```
MariaDB [poei]> SELECT COUNT(DISTINCT(ville)) AS nombre_ville FROM personne;
+--------------+
| nombre_ville |
+--------------+
|            3 |
+--------------+
1 row in set (0.001 sec)
```

Requêtes simples

SQL : WHERE

Pour sélectionner les personnes dont la ville = ‘Marseille’
SELECT * **FROM** personne **WHERE** ville=’Marseille’;

Le résultat :

```
MariaDB [poei]> SELECT * FROM personne WHERE ville=’Marseille’;
+-----+---------+---------+---------+-----------+
| num | nom     | prenom  | salaire | ville     |
+-----+---------+---------+---------+-----------+
|   1 | Cohen   | Sophie  |    2000 | Marseille |
|   2 | Leberre | Bernard |    1500 | Marseille |
+-----+---------+---------+---------+-----------+
2 rows in set (0.000 sec)
```

Requêtes simples

SQL : WHERE

Il est possible de définir plusieurs conditions en utilisant les opérateurs logiques

- and
- or
- !

Requêtes simples

SQL : WHERE

Exercice 1

Ecrire une requête SQL qui permet de sélectionner les personnes qui habitent Marseille ou Lyon

Exercice 2

Ecrire une requête SQL qui permet de sélectionner les personnes dont le salaire est compris entre 2000 et 3000.

Requêtes simples

SQL : BETWEEN

Pour l'exercice 2, on peut aussi utiliser BETWEEN **et** AND

```
SELECT *
FROM personne
WHERE salaire BETWEEN 2000 AND 3000;
```

Le résultat :

```
MariaDB [poei]> SELECT *
    -> FROM personne
    -> WHERE salaire BETWEEN 2000 AND 3000;
+-----+-------+--------+---------+-----------+
| num | nom   | prenom | salaire | ville     |
+-----+-------+--------+---------+-----------+
|   1 | Cohen | Sophie |    2000 | Marseille |
|   4 | Hadad | Karim  |    2500 | Paris     |
|   5 | Wick  | John   |    3000 | Paris     |
+-----+-------+--------+---------+-----------+
3 rows in set (0.002 sec)
```

Requêtes simples

SQL : IN

Pour sélectionner les personnes ayant un salaire = 2000 ou 3000

```
SELECT *
FROM personne
WHERE salaire IN (2000, 3000);
```

Le résultat :

```
MariaDB [poei]> SELECT *
    -> FROM personne
    -> WHERE salaire IN (2000, 3000);
+-----+-------+--------+---------+-----------+
| num | nom   | prenom | salaire | ville     |
+-----+-------+--------+---------+-----------+
|   1 | Cohen | Sophie |    2000 | Marseille |
|   5 | Wick  | John   |    3000 | Paris     |
+-----+-------+--------+---------+-----------+
2 rows in set (0.000 sec)
```

Requêtes simples

SQL : IN

Exercice 3

Ecrire une requête SQL qui permet de sélectionner les personnes qui habitent Marseille et donc le salaire est soit inférieur à 2000 soit supérieur à 2500.

Requêtes simples

SQL : NOT

Pour l'exercice 3, on peut utiliser NOT **et** BETWEEN

```
SELECT *
FROM personne
WHERE ville='Marseille'
AND salaire NOT BETWEEN 2000 AND 2500;
```

Le résultat :

```
MariaDB [poei]> SELECT *
    -> FROM personne
    -> WHERE ville='Marseille'
    -> AND salaire NOT BETWEEN 2000 AND 2500;
+-----+---------+---------+---------+-----------+
| num | nom     | prenom  | salaire | ville     |
+-----+---------+---------+---------+-----------+
|   2 | Leberre | Bernard |    1500 | Marseille |
+-----+---------+---------+---------+-----------+
1 row in set (0.001 sec)
```

Requêtes simples

SQL : LIKE

Pour sélectionner les personnes dont le nom de la ville contient la lettre a

```
SELECT *
FROM personne
WHERE ville LIKE'%a%';
```

Le résultat :

```
MariaDB [poei]> SELECT *
    -> FROM personne
    -> WHERE ville LIKE'%a%';
+-----+---------+---------+---------+-----------+
| num | nom     | prenom  | salaire | ville     |
+-----+---------+---------+---------+-----------+
|   1 | Cohen   | Sophie  |    2000 | Marseille |
|   2 | Leberre | Bernard |    1500 | Marseille |
|   4 | Hadad   | Karim   |    2500 | Paris     |
|   5 | Wick    | John    |    3000 | Paris     |
+-----+---------+---------+---------+-----------+
4 rows in set (0.001 sec)
```

Requêtes simples

SQL : IS

Pour sélectionner les vehicules dont le numéro du propriétaire est non-nul

```
SELECT *
FROM vehicule
WHERE nump IS NOT NULL;
```

Le résultat :

```
MariaDB [poei]> SELECT *
    -> FROM vehicule
    -> WHERE nump IS NOT NULL;
+----------------+---------+--------+-------+------+
| immatriculation | marque  | modele | annee | nump |
+----------------+---------+--------+-------+------+
|             100 | Peugeot | 5008   |  2018 |    5 |
|             200 | Renault | Clio   |  2000 |    4 |
|             300 | Ford    | Fiesta |  2010 |    1 |
|             400 | Peugeot | 106    |  2002 |    3 |
|             500 | Citroen | C4     |  2015 |    4 |
|             700 | Fiat    | Punto  |  2008 |    5 |
+----------------+---------+--------+-------+------+
6 rows in set (0.000 sec)
```

Requêtes simples

SQL : IFNULL

Pour remplacer les valeurs nulles par une autre valeur, on peut utiliser la fonction IFNULL ()
SELECT marque, **IFNULL**(nump, ‘pas de propriétaire ’) **AS** proprietaire **FROM** vehicule;

Le résultat :

```
MariaDB [poei]> SELECT marque, IFNULL(nump, ‘pas de propriétaire ’) AS proprietaire
    -> FROM vehicule;
+---------+----------------------+
| marque  | proprietaire         |
+---------+----------------------+
| Peugeot | 5                    |
| Renault | 4                    |
| Ford    | 1                    |
| Peugeot | 3                    |
| Citroen | 4                    |
| Ford    | pas de propriétaire  |
| Fiat    | 5                    |
+---------+----------------------+
7 rows in set (0.000 sec)
```

Requêtes simples

SQL : ORDER BY

Pour ordonner le résultat selon le numéro du propriétaire
SELECT * **FROM** vehicule **WHERE** nump I**S NOT NULL ORDER BY** nump;

Le résultat :

```
MariaDB [poei]> SELECT * FROM vehicule WHERE nump IS NOT NULL ORDER BY nump;
+----------------+---------+--------+-------+------+
| immatriculation | marque  | modele | annee | nump |
+----------------+---------+--------+-------+------+
|             300 | Ford    | Fiesta |  2010 |    1 |
|             400 | Peugeot | 106    |  2002 |    3 |
|             200 | Renault | Clio   |  2000 |    4 |
|             500 | Citroen | C4     |  2015 |    4 |
|             100 | Peugeot | 5008   |  2018 |    5 |
|             700 | Fiat    | Punto  |  2008 |    5 |
+----------------+---------+--------+-------+------+
```

Requêtes simples

SQL : GROUP BY

Pour compter le nombre de véhicule pour chaque personne, il faut les regrouper et utiliser une fonction d'agrégation.
SELECT nump, **COUNT**(*) **AS** nombre_vehicule **FROM** vehicule **WHERE** nump **IS NOT NULL GROUP BY** nump;

Le résultat :

```
MariaDB [poei]> SELECT nump, COUNT(*) AS nombre_vehicule FROM vehicule WHERE nump IS NOT NULL GROUP BY nump;
+------+-----------------+
| nump | nombre_vehicule |
+------+-----------------+
|    1 |               1 |
|    3 |               1 |
|    4 |               2 |
|    5 |               2 |
+------+-----------------+
4 rows in set (0.002 sec)
```

Requêtes simples

SQL : HAVING

Pour filtrer les résultats de la fonction d'agrégation.
SELECT nump, **COUNT**(*) **AS** nombre_vehicule **FROM** vehicule **WHERE** nump **IS NOT NULL GROUP BY** nump **HAVING** nombre_vehicule > 1;

Le résultat :

```
MariaDB [poei]> SELECT nump, COUNT(*) AS nombre_vehicule FROM vehicule WHERE nump IS NOT NULL GROUP BY nump HAVING nombre_vehicule > 1;
+------+-----------------+
| nump | nombre_vehicule |
+------+-----------------+
|    4 |               2 |
|    5 |               2 |
+------+-----------------+
2 rows in set (0.003 sec)
```

Requêtes simples

SQL : HAVING

Fonctions d'agrégation

- MAX
- MIN
- COUNT
- SUM
- AVG

Remarques

- **Imbriquer les fonctions d'agrégation n'est pas possible**
- **DISTINCT n'est pas une fonction d'agrégation**
- **Pas d'espace entre la fonction d'agrégation et la parenthèse ouvrante (par exemple : max(...))**

Requêtes simples

SQL : HAVING

Exercice 4

Ecrire une requête SQL qui permet de compter la somme des salaires par ville

Exercice 5

Ecrire une requête SQL qui permet de sélectionner les numéros de personne qui ont un véhicule de marque *Renault* ou *Citroen*

Requêtes simples

SQL : UNION

Pour répondre à la question de l'exercice 5, on peut utiliser UNION

```
SELECT nump
FROM vehicule WHERE marque='Renault'
UNION
SELECT nump
FROM vehicule WHERE marque='Citroen';
```

```
MariaDB [poei]> SELECT nump
    -> FROM vehicule WHERE marque='Renault'
    -> UNION
    -> SELECT nump
    -> FROM vehicule WHERE marque='Citroen';
+------+
| nump |
+------+
|    4 |
+------+
1 row in set (0.002 sec)
```

Requêtes simples

SQL : UNION

Opérations sur les ensembles

- UNION
- INTERSECT
- EXCEPT
- MINUS

Remarques

Les trois dernières opérations ne fonctionnent pas avec MySQL. Il est cependant possible de les remplacer par les requêtes imbriquées.

Jointures

SQL

Remarques

- Pour les propriétaires des véhicules, on affichait chaque fois le numéro
- Pour l'utilisateur, il serait mieux de connaitre les noms et prénoms que le numéro
- Les noms et prénoms sont dans la table personne

Solution

Les jointures

Jointures

SQL

Deux types de jointures

- Implicite : sans le mot-clé JOIN
- Explicite : avec le mot-clé JOIN

Jointures

SQL

Pour sélectionner toutes les données de la table personne
SELECT nom, prenom, ville, marque, modele
FROM personne, vehicule
WHERE personne.num = vehicule.nump;

```
MariaDB [poei]> SELECT nom, prenom, ville, marque, modele
    -> FROM personne, vehicule
    -> WHERE personne.num = vehicule.nump;
+---------+--------+-----------+---------+--------+
| nom     | prenom | ville     | marque  | modele |
+---------+--------+-----------+---------+--------+
| Wick    | John   | Paris     | Peugeot | 5008   |
| Hadad   | Karim  | Paris     | Renault | Clio   |
| Cohen   | Sophie | Marseille | Ford    | Fiesta |
| Benabar | Pierre | Lyon      | Peugeot | 106    |
| Hadad   | Karim  | Paris     | Citroen | C4     |
| Wick    | John   | Paris     | Fiat    | Punto  |
+---------+--------+-----------+---------+--------+
6 rows in set (0.001 sec)
```

Jointures

SQL

Remarques

- **La jointure se fait, généralement, sur une colonne commune entre deux tables (clé primaire dans une première table qui est étrangère dans une deuxième)**
- **En cas d'ambiguïté, c'est-à-dire, si deux colonnes portent le même nom, il faut les préfixer par le nom de leurs tables respectives.**
- **Il est aussi possible de définir et d'utiliser des alias**
- **Les personnes qui n'ont pas de véhicules et les véhicules qui n'ont pas de propriétaires n'apparaissent pas dans le résultat.**

Jointures

SQL : JOINTURE IMPLICITE

On peut utiliser les alias

```
SELECT nom, prenom, ville, marque, modele
FROM personne p, vehicule v
WHERE p.num = v.nump;
```

```
MariaDB [poei]> SELECT nom, prenom, ville, marque, modele
    -> FROM personne p, vehicule v
    -> WHERE p.num = v.nump;
+---------+--------+-----------+---------+--------+
| nom     | prenom | ville     | marque  | modele |
+---------+--------+-----------+---------+--------+
| Wick    | John   | Paris     | Peugeot | 5008   |
| Hadad   | Karim  | Paris     | Renault | Clio   |
| Cohen   | Sophie | Marseille | Ford    | Fiesta |
| Benabar | Pierre | Lyon      | Peugeot | 106    |
| Hadad   | Karim  | Paris     | Citroen | C4     |
| Wick    | John   | Paris     | Fiat    | Punto  |
+---------+--------+-----------+---------+--------+
6 rows in set (0.001 sec)
```

Jointures

SQL : JOINTURE IMPLICITE

En l'absence d'ambiguïté, on peut écrire
SELECT nom, prenom, ville, marque, modele
FROM personne, vehicule
WHERE num = nump;

```
MariaDB [poei]> SELECT nom, prenom, ville, marque, modele
    -> FROM personne, vehicule
    -> WHERE num = nump;
+---------+--------+-----------+---------+--------+
| nom     | prenom | ville     | marque  | modele |
+---------+--------+-----------+---------+--------+
| Wick    | John   | Paris     | Peugeot | 5008   |
| Hadad   | Karim  | Paris     | Renault | Clio   |
| Cohen   | Sophie | Marseille | Ford    | Fiesta |
| Benabar | Pierre | Lyon      | Peugeot | 106    |
| Hadad   | Karim  | Paris     | Citroen | C4     |
| Wick    | John   | Paris     | Fiat    | Punto  |
+---------+--------+-----------+---------+--------+
6 rows in set (0.000 sec)
```

Jointures

SQL : JOINTURE EXPLICITE

Plusieurs syntaxes

- JOIN … ON : exactement comme la jointure implicite
- LEFT JOIN … ON : jointure gauche
- RIGHT JOIN … ON : jointure droite
- FULL JOIN … ON : jointures droite + gauche (ne fonctionne pas avec MySQL)

Jointures

SQL : JOINTURE EXPLICITE

On peut utiliser le mot-clé JOIN **et indiquer les colonnes sur lesquelles on fait la jointure**

```
SELECT nom, prenom, ville, marque, modele
FROM personne JOIN vehicule
ON num = nump;
```

```
MariaDB [poei]> SELECT nom, prenom, ville, marque, modele
    -> FROM personne JOIN vehicule
    -> ON num = nump;
+---------+--------+-----------+---------+--------+
| nom     | prenom | ville     | marque  | modele |
+---------+--------+-----------+---------+--------+
| Wick    | John   | Paris     | Peugeot | 5008   |
| Hadad   | Karim  | Paris     | Renault | Clio   |
| Cohen   | Sophie | Marseille | Ford    | Fiesta |
| Benabar | Pierre | Lyon      | Peugeot | 106    |
| Hadad   | Karim  | Paris     | Citroen | C4     |
| Wick    | John   | Paris     | Fiat    | Punto  |
+---------+--------+-----------+---------+--------+
6 rows in set (0.018 sec)
```

Jointures

SQL : JOINTURE EXPLICITE

Pour les personnes qui n'ont pas de véhicule, on peut écrire
SELECT nom, prenom, ville, marque, modele
FROM personne **LEFT JOIN** vehicule
ON num = nump;

```
MariaDB [poei]> SELECT nom, prenom, ville, marque, modele
    -> FROM personne LEFT JOIN vehicule
    -> ON num = nump;
+---------+---------+-----------+---------+--------+
| nom     | prenom  | ville     | marque  | modele |
+---------+---------+-----------+---------+--------+
| Cohen   | Sophie  | Marseille | Ford    | Fiesta |
| Leberre | Bernard | Marseille | NULL    | NULL   |
| Benabar | Pierre  | Lyon      | Peugeot | 106    |
| Hadad   | Karim   | Paris     | Renault | Clio   |
| Hadad   | Karim   | Paris     | Citroen | C4     |
| Wick    | John    | Paris     | Peugeot | 5008   |
| Wick    | John    | Paris     | Fiat    | Punto  |
+---------+---------+-----------+---------+--------+
7 rows in set (0.003 sec)
```

Jointures

SQL : JOINTURE EXPLICITE

Pour les véhicules qui n'ont pas de propriétaire, on peut écrire
SELECT nom, prenom, ville, marque, modele
FROM personne **RIGHT JOIN** vehicule
ON num = nump;

```
MariaDB [poei]> SELECT nom, prenom, ville, marque, modele
    -> FROM personne RIGHT JOIN vehicule
    -> ON num = nump;
+---------+--------+-----------+---------+--------+
| nom     | prenom | ville     | marque  | modele |
+---------+--------+-----------+---------+--------+
| Wick    | John   | Paris     | Peugeot | 5008   |
| Hadad   | Karim  | Paris     | Renault | Clio   |
| Cohen   | Sophie | Marseille | Ford    | Fiesta |
| Benabar | Pierre | Lyon      | Peugeot | 106    |
| Hadad   | Karim  | Paris     | Citroen | C4     |
| NULL    | NULL   | NULL      | Ford    | Kuga   |
| Wick    | John   | Paris     | Fiat    | Punto  |
+---------+--------+-----------+---------+--------+
7 rows in set (0.001 sec)
```

Jointures

SQL : JOINTURE EXPLICITE

Exercice 6

Ecrire une requête SQL qui permet d’afficher le résultat de la jointure entre personne et véhicule + les personnes n’ayant pas de voiture + les voitures n’ayant pas de propriétaire (**FULL JOIN**)

Requêtes imbriquées

SQL

Imbriquer les requêtes

Avoir une requête dans la clause d'une autre

Plusieurs niveaux d'imbrication

- WHERE
- FROM
- HAVING

Requêtes imbriquées

SQL : WHERE … IN … SELECT

Exercice 7

Ecrire une requête SQL qui permet de sélectionner les personnes qui ont à la fois un véhicule Fiat et un Peugeot

Requêtes imbriquées

SQL : WHERE … IN … SELECT

Pour répondre à la question de l'exercice 7, on peut utiliser les requêtes imbriquées

```
SELECT nom, prenom
FROM personne, vehicule WHERE nump = num
AND marque ='Fiat'
AND num IN (SELECT nump
FROM vehicule
WHERE marque = 'Peugeot');
```

```
MariaDB [poei]> SELECT nom, prenom
    -> FROM personne, vehicule WHERE nump = num
    -> AND marque ='Fiat'
    -> AND num IN (SELECT nump
    -> FROM vehicule
    -> WHERE marque = 'Peugeot');
+------+--------+
| nom  | prenom |
+------+--------+
| Wick | John   |
+------+--------+
1 row in set (0.005 sec)
```

Requêtes imbriquées

SQL : WHERE … ANY … SELECT

La même requête peut s'écrire avec ANY

```
SELECT nom, prenom
FROM personne, vehicule WHERE nump = num
AND marque ='Fiat'
AND num = ANY (SELECT nump
FROM vehicule
WHERE marque = 'Peugeot');
```

```
MariaDB [poei]> SELECT nom, prenom
    -> FROM personne, vehicule WHERE nump = num
    -> AND marque ='Fiat'
    -> AND num = ANY (SELECT nump
    -> FROM vehicule
    -> WHERE marque = 'Peugeot');
+------+--------+
| nom  | prenom |
+------+--------+
| Wick | John   |
+------+--------+
1 row in set (0.001 sec)
```

Requêtes imbriquées

SQL : WHERE … ANY … SELECT

Pour les requêtes imbriquées on peut utiliser

- IN
- ALL
- ANY

Avec ALL et IN, on peut utiliser les opérateurs de comparaison suivants

=, <, >, <>, !=,<=, >=

Requêtes imbriquées

SQL : WHERE … ANY … SELECT

Exercice 8

Ecrire une requête SQL qui permet de sélectionner les marques de voiture qui appartiennent à des personnes ayant deux véhicules et dont la ville = Paris.

Requêtes imbriquées

SQL : WHERE EXISTS … SELECT

La même requête peut être réécrite avec EXISTS **qui retourne un booléen**

```
SELECT nom, prenom
FROM personne, vehicule WHERE nump = num
AND marque =’Fiat’
AND EXISTS (SELECT nump
FROM vehicule
WHERE marque = ’Peugeot’);
```

```
MariaDB [poei]> SELECT nom, prenom
    -> FROM personne, vehicule WHERE nump = num
    -> AND marque =’Fiat’
    -> AND EXISTS (SELECT nump
    -> FROM vehicule
    -> WHERE marque = ’Peugeot’);
+------+--------+
| nom  | prenom |
+------+--------+
| Wick | John   |
+------+--------+
1 row in set (0.003 sec)
```

Requêtes imbriquées

SQL : WHERE EXISTS … SELECT

Pour les requêtes imbriquées, il est aussi possible d'utiliser la négation

- NOT IN
- NOT EXISTS

Requêtes imbriquées

SQL : FROM … SELECT … AS

On peut aussi imbriqué des requêtes dans la clause FROM
SELECT nom, prenom
FROM personne p, (**SELECT** nump **FROM** vehicule **WHERE** marque = “Peugeot”) **AS** peugeot
WHERE peugeot.nump = p.num;

```
MariaDB [poei]> SELECT nom, prenom
    -> FROM personne p, (SELECT nump FROM vehicule WHERE marque = "Peugeot") AS peugeot
    -> WHERE peugeot.nump = p.num;
+---------+--------+
| nom     | prenom |
+---------+--------+
| Wick    | John   |
| Benabar | Pierre |
+---------+--------+
2 rows in set (0.001 sec)
```

Requêtes imbriquées

SQL : FROM … SELECT … AS

Exercice 9

Ecrire une requête SQL qui permet de sélectionner les marques de voiture qui sont à la fois à Paris et à Lyon

Exercice 10

Ecrire une requête SQL qui permet de sélectionner les personnes qui ont une voiture Peugeot mais pas Fiat

Exercice 11

Ecrire une requête SQL qui permet de sélectionner les personnes qui ont le plus grand nombre de voitures

Fonctions sur les chaînes de caractères

SQL

Plusieurs fonctions prédéfinies

- **CONCAT**
- **TRIM**
- **LENGTH**
- **SUBSTRING**
- **UPPER**
- **LOWER**
- **…**

Fonctions sur les chaînes de caractères

SQL

Exemple avec CONCAT **et** UPPER

```
SELECT CONCAT (UPPER(nom), prenom)
        AS nom_complet
        FROM personne;
```

```
MariaDB [poei]> SELECT CONCAT (UPPER(nom), prenom) AS nom_complet
    -> FROM personne;
+----------------+
| nom_complet    |
+----------------+
| COHENSophie    |
| LEBERREBernard |
| BENABARPierre  |
| HADADKarim     |
| WICKJohn       |
+----------------+
5 rows in set (0.001 sec)
```