***1.1.2.1Syntaxe***

**[** mots\_clés **]** : signifie en option

mot\_clé1 **|** mot\_clé2 : signifie mot\_clé1 **ou** mot\_clé2

***1.1.2.2Création d'une base de données***

*CREATE DATABASE nom\_base ;Exemple : CREATE DATABASE clients ;*

***1.1.2.3Connexion à une base***

*USE nom\_base ;ou CONNECT nom\_base ;Ex : USE clients ;ou CONNECT clients ;*

***1.1.2.4Création d'une table***

Correspond à la création d'une relation avec sa structure : attributs et domaines. On ne peut créer une table qu'à l'intérieur d'une base de données déjà existante.

*CREATE TABLE coordonnees (nom CHAR(20),*

*prenom CHAR(20),*

*adresse CHAR(40),*

*codepostal INTEGER,*

*ville CHAR(20));*

***1.1.2.5Modification d'une table***

*Ajouter une colonne : ALTER TABLE coordonnees ADD COLUMN telephone INTEGER;*

*Supprimer une colonne :ALTER TABLE coordonnees DROP COLUMN telephone;*

*Modifier une colonne : ALTER TABLE coordonnees MODIFY COLUMN prenom VARCHAR(20);*

***1.1.2.6Suppression d'une table***

*DROP TABLE coordonnees;*

**1.1.3Commandes de LID (Langage d’Interrogation de Données)**

***1.1.3.1Requête générique d'interrogation***

SELECT {expressions}

FROM {TABLES}

[WHERE {prédicats}

[GROUP BY {expressions} [HAVING {condition}] ] | [ORDER BY {expressions}] ] ;

***1.1.3.2Projection***

SELECT {liste\_colonnes} FROM {tables} ;

Exemple : *SELECT nom, prenom FROM coordonnees ;*

***1.1.3.3Union***

SELECT {…..} FROM table1 UNION [ALL] SELECT {….} FROM table2 ;

Restriction : Les 2 projections de la requête doivent posséder le même nombre de champs de données, de même type, dans le même ordre. Donne un résultat en supprimant les enregistrements en double, sauf si l'option ALL est activée.

***1.1.3.4Intersection***

SELECT {…..} FROM table1 INTERSECT SELECT {….} FROM table2 ;

Restriction : Les 2 projections de la requête doivent posséder le même nombre de champs de données, de même type, dans le même ordre.

Exemple : *SELECT nom, prenom FROM coordonnees INTERSECTSELECT name, surname FROM client ;*

MySQL ne propose pas encore cette commande, mais on peut la simuler avec :

SELECT [DISTINCT] value FROM table1 WHERE value IN ( SELECT value FROM table2 );

La commande DISTINCT évite d'afficher les doubles.

***1.1.3.5Restriction (selection)***

SELECT {liste\_colonnes}| SELECT COUNT(colonne)| SELECT SUM(colonne) FROM {tables} WHERE conditions […] ;

Exemple : *SELECT \* FROM coordonnees WHERE codepostal = 22100 AND ville = "dinan" ;*

***1.1.3.6Jointure***

SELECT {liste\_colonnes} FROM {table1, table2} WHERE table1.champ opérateur table2.champ ;

Exemple d’équi-jointure (opérateur =), appelée aussi jointure interne :

*SELECT nom, ville FROM coordonnees, commandes*

*WHERE ref\_produit = GT02 AND coordonnees.numclient = commandes.numclient ;*

***1.1.3.7Tri***

Le tri s'effectue à l'affichage. La base de donnée n'est pas modifiée.

Exemple : *SELECT nom, prenom FROM coordonnees ORDER BY nom ;*

**1.1.4Commande LMD (Langage de Manipulation de Données)**

***1.1.4.1Insertion d’un enregistrement (ligne d’une table)***

INSERT INTO table(colonne1, colonne2, …) VALUES ("valeur1", valeur2, …) ;

Les valeurs de type texte et date sont à déclarer entre " " ou ‘ ‘.

Exemple :*INSERT INTO coordonnees(nom, prenom, adresse, codepostal, ville)*

*VALUES ("Dupond", "Jean","8 bd de la République", 35400, "Saint-Malo") ;*

***1.1.4.2Modification d’enregistrements***

UPDATE table SET colonneX="valeur1" WHERE colonneY=valeur2 ;

Exemple : *UPDATE coordonnees SET telephone=0299211212 WHERE nom="Maupertuis" ;*

***1.1.4.3Suppression d'enregistrements***

DELETE FROM table WHERE colonneX = valeur1 [ [AND] | [OR] colonneY=valeur2 ] ;

Exemple : *DELETE FROM coordonnees WHERE nom="Martin" ;*

**1.1.5Commandes LCA (Langage de Contrôle d’Accès)**

•**GRANT** autorise certains utilisateurs à effectuer une certaine opération.

•**DENY** empêche certains utilisateurs à effectuer une certaine opération.

•**REVOKE** annule une commande de contrôle précédente.

•**UPDATE** autorise un utilisateur à mettre à jour des enregistrements.

•**READ** empêche un utilisateur de modifier la base de données, il peut juste lire les données.

•**DELETE** autorise un utilisateur à effacer des enregistrements.

***1.1.5.1Syntaxe***

GRANT <privilège> TO <utilisateurs>

<privilège> ::= <liste de privilège> ON {table} | <privilège sur la base>

**1.1.5.1.1Liste de privilèges**

•**alter** : droit de modifier la structure de la table

•**delete** : droit d'effacer des enregistrements

•**insert** : droit d'insérer des enregistrements

•**update** : droit de modifier des valeurs

•**select** : droits d'accès aux valeurs des enregistrements

•**all** : toutes les opérations précédentes

**1.1.5.1.2Privilèges sur la base de données**

•***Connect*** : accès à la base, mais interdit de créer et d'altérer les tables

•***ressource*** : connect + création de table

•***dba*** : administrateur : tous privilèges

**1.1.5.1.3Exemple**

*GRANT ALL ON clients TO Thierry*

***3.1Contraintes de colonnes***

Définie les contraintes par colonnes :

•**NOT NULL** ou **NULL** : interdit ou autorise l’insertion de valeur NULL pour cet attribut.

•**UNIQUE** : oblige chaque valeur d'une colonne à être différente. La colonne devient une clé secondaire de la table.

•**PRIMARY KEY** : désigne l’attribut comme clé primaire de la table. La clé primaire étant unique, cette contrainte ne peut apparaître que pour une colonne

•**CHECK (condition)** : vérifie lors de l’insertion de n-uplets que l’attribut réalise la condition.

•**DEFAULT** valeur : spécifie une valeur par défaut : constante numérique ou alphanumérique, **USER** nom utilisateur, **NULL**, **CURRENT\_DATE**, **CURRENT\_TIME**, CURRENT\_TIMESTAMP, etc...

|  |  |  |
| --- | --- | --- |
| **2Principaux types de donnéesType** | **Description** | **Norme** |
| CHAR(n) | Chaîne de caractères de longueur fixe n (1 ≤ n ≤ 255) | SQL 2 |
| VARCHAR(n) | Chaîne de caractères de longueur variable n=max (1 ≤ n ≤ X) | SQL 2 |
| BIGINT [unsigned] | Entier signé de 64 bits (-263 à +263-1) ou (0 à 264-1) | Courant |
| INTEGER [unsigned] | Entier signé de 32 bits (-231 à +231-1) ou (0 à 232-1) | SQL 2 |
| SMALLINT [unsigned] | Entier signé de 16 bits (-32768 à +32767) ou (0 à 65535) | SQL 2 |
| TINYINT [unsigned] | Entier signé sur 8 bits (-128 à +127) ou (0 à 255) | Courant |
| DOUBLE | Réels comportant 15 chiffres significatifs codés sur 8 octets | SQL 2 |
| FLOAT | Réels comportant 6 chiffres significatifs codés sur 4 octets. | SQL 2 |
| NUMERIC(n,[d]) | n = nb chiffres au total ; d = nb chiffres décimaux ; stocké sous forme  de chaîne de caractères | SQL 2 |
| DATE | Date sous la forme 16/07/99 | SQL 2 |
| TIME | Heure sous la forme 12:54:24.85 | SQL 2 |
| DATETIME | Date et Heure : AAAA-MM-JJ HH:MM:SS sur 8 octets | SQL 2 |
| YEAR(4) | Valeur de 1901 à 2155 | Courant |
| BIT(n) | Existe aussi BITVARYING(n) | SQL 2 |
| BOOLEAN | Valeur booléenne | SQL 3 |
| MONEY | Valeur monétaire | Courant |
| ENUM | Des valeurs dans un ensemble prédéfini | Courant |
| BLOB | Binary Large Objet (données binaires : images, son, vidéo) | Courant |

***Pour lancer mysql***

***Avant***  *Sudo mysql*

*mysql -u root -p*

*CREATE DATABASE gourmand;*

*connect table gournand ;*

*connect gourmand2;*

*show tables;*

*show databases;*

*describe client;*

*table client;*

*show table commande;*

***pour modifier dans la table***

*ALTER TABLE produit MODIFY COLUMN reference char(4) primary key;*

***sauvegarder***

*Linux>mysqldump -u root -p gourmand > gourmandAAAAMMJJ.sql*

***Importer*** *les enregistrements dans la base de données*

*Linux> mysql -u root -p gourmand < gourmand.sql*

***5.1Structure de la table "client"***

1.code\_c entier non signé, clé primaire

2.nomChaîne de caractères de longueur variable (35 max), non null

3.adresseChaîne de caractères de longueur variable (50 max)

4.cpChaîne de caractères de longueur fixe (5)

5.villeChaîne de caractères de longueur variable (25 max)

6.telephoneChaîne de caractères de longueur variable (16 max)

*CREATE TABLE client (code\_c INTEGER unsigned PRIMARY KEY, nom VARCHAR(35) not null, adresse VARCHAR(50), cp CHAR(5), ville VARCHAR(25), telephone VARCHAR(16));*

***5.2Structure de la table "vendeur"***

1.code\_v entier non signé, clé primaire

2.nomChaîne de caractères de longueur variable (35 max), non null

3.adresseChaîne de caractères de longueur variable (40 max)

4.cpChaîne de caractères de longueur fixe (5)

5.villeChaîne de caractères de longueur variable (30 max)

6.telephoneChaîne de caractères de longueur variable (16 max)

*CREATE TABLE vendeur (code\_v INTEGER PRIMARY KEY,*

*nom VARCHAR(35) NOT NULL,*

*adresse VARCHAR(40),*

*cp CHAR(5),*

*ville VARCHAR(30),*

*telephone VARCHAR(16));*

***5.3Structure de la table "commande"***

1.numero entier non signé, clé primaire, non null

2.#code\_ventier non signé, clé étrangère table vendeur

3.#code\_centier non signé, clé étrangère table client

4.date\_livraisondate

5.date\_commandedate

6.total\_htnumeric(9,2)

7.total\_tvanumeric(9,2)

8.etatbooléen

*CREATE TABLE commande (numero INTEGER PRIMARY KEY,*

*code\_v INTEGER UNSIGNED,*

*code\_c INTEGER UNSIGNED,*

*date\_livraison DATE,*

*date\_commande DATE,*

*total\_ht NUMERIC(9,2),*

*total\_tva NUMERIC(9,2),*

*etat CHAR(1),*

*FOREIGN KEY(code\_v) REFERENCES vendeur(code\_v),*

*FOREIGN KEY(code\_c) REFERENCES client(code\_c));*

*CREATE TABLE produit (reference CHAR(5), designation VARCHAR(35) not null, quantite INTEGER unsigned, descriptif VARCHAR(100), prix\_unitaire\_ht NUMERIC(9.2), stock INTEGER unsigned, poids\_piece INTEGER unsigned);*

*CREATE TABLE ligne\_commande*

*#numero*

*numero\_ligne*

*#reference table produit*

*quantite\_demandee*

*create table ligne commande (#numero INTEGER unsigned not null XXXX, numero\_ligne INTEGER not null, #reference char(4) not null XXXXtableproduit, quantite\_demandee INTEGER);*

***produit cartésien pour compter ….***

*SELECT COUNT(\*) FROM client, commande, ligne\_commande, produit, vendeur;*

***Requête SQL avec expression régulière***

Rechercher le nom des clients (nom, adresse, téléphone) dont la deuxième lettre est **A**, la quatrième **I** et la cinquième **N**.

Vous pourrez utiliser l'opérateur de comparaison LIKE expliqué page 189 de refman-5.0-fr.pdf

*SELECT nom, adresse, telephone FROM client WHERE nom LIKE "\_A\_IN%";*

***Jointure***

Rechercher les numéros des commandes passées par Mme RABIN.\*

*Selection du numero de la table commande, de la table client ou on passe par table commande du code\_c jointure table client du code\_c et une fois relié la table client au nom like RABIN*

*select numero from commande, client where commande.code\_c = client.code\_c and client.nom like "RABIN %";*

***6.6Requête SQL avec multiples jointures***

Rechercher la liste des articles (Désignation) achetés par Mme RABIN.\*

**SQL DISTINCT**

L’utilisation de la commande [SELECT](https://sql.sh/cours/select) en SQL permet de lire toutes les données d’une ou plusieurs colonnes. Cette commande peut potentiellement afficher des lignes en doubles. Pour éviter des redondances dans les résultats il faut simplement ajouter DISTINCT après le mot SELECT.

Selection (distinct pour éviter les doublons) designation de la table produit en passant par les tables ligne de commande, commande et client. Ou pour faire les jointures : table + ligne = autre table + meme ligne. Exemple : produit. Reference = ligne commande.reference and ligne commande. Numero= commande.numero and …. And une fois arrivé dans la table and nom like « RABIN% » ;

*SELECT DISTINCT designation FROM produit, ligne\_commande, commande, client WHERE*

*produit.reference = ligne\_commande.reference*

*AND ligne\_commande.numero = commande.numero*

*AND commande.code\_c = client.code\_c*

*AND nom LIKE "RABIN %";*

***6.7Requête SQL arithmétique avec jointure***

Rechercher le montant HT (total) des commandes de Mme RABIN, sachant que le montant HT des commandes inclut le transport.\*

*SELECT sum(total\_ht) FROM commande, client WHERE*

*commande.code\_c = client.code\_c*

*AND nom LIKE "RABIN %";*

***6.8Requête SQL arithmétique avec multiples jointures (1 point)***

Rechercher le Montant HT (total) des commandes de Mme RABIN, hors transport (calculé d'après le prix unitaire hors taxes des articles achetés).\*

*SELECT SUM(quantitee\_demandee\*prix\_unitaire\_HT) FROM ligne\_commande, produit, commande, client WHERE*

*produit.reference = ligne\_commande.reference*

*AND commande.numero = ligne\_commande.numero*

*AND commande.code\_c = client.code\_c*

*AND nom LIKE "RABIN %";*

***6.9Requête SQL arithmétique avec opérateur de comparaison***

Calculer le chiffre d'affaires total pour le mois d'octobre 2008. Vous pourrez utiliser l'opérateur de comparaison **BETWEEN** expliqué page 577 de refman-5.0-fr.pdf ou l'opérateur LIKE. Les **dates** sont exprimées au format **YYYY-MM-JJ**. (Voir page 552/53 et à partir de la page 557 de refman-5.0-fr.pdf).

*SELECT SUM(total\_ht) FROM commande*

*WHERE date\_commande BETWEEN '2008-10-01' AND '2008-10-31';*

***6.10Requête SQL arithmétique avec jointure et comparaison***

Calculer le chiffre d'affaires de chaque vendeur pour le mois d'octobre 2008 (Nom du vendeur, CA).

*SELECT nom, SUM(total\_ht) FROM commande, vendeur*

*WHERE commande.code\_v=vendeur.code\_v*

*AND date\_commande BETWEEN '2008-10-01' AND '2008-10-31'*

*GROUP BY nom;*

***6.11Requête SQL arithmétique avec groupement (1 point)***

Rechercher la liste des chiffres d'affaires quotidiens supérieurs à 300€ (date, CA journalier) d'après les dates de commande.

*SELECT date\_commande, SUM(total\_ht) FROM commande*

*GROUP BY date\_commande*

*HAVING SUM(total\_ht)>300;*

**SQL HAVING**

La condition HAVING en SQL est presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que SUM(), COUNT(), AVG(), MIN() ou MAX().

***6.12Requête SQL arithmétique, avec jointure, groupement et tri***

Rechercher la liste des produits figurant au moins dans deux commandes classée du plus sollicité au moins sollicité (désignation produit, nombre de fois que ce produit figure dans une commande).

*SELECT designation, COUNT(ligne\_commande.reference) FROM ligne\_commande, produit*

*WHERE produit.reference=ligne\_commande.reference*

*GROUP BY designation*

*HAVING COUNT(ligne\_commande.reference)>=2*

*ORDER BY count(ligne\_commande.reference) DESC;*

*calcul de tva*

*select sum(total\_ht) from commande where date\_commande between '2008-10-10' and '2008-10-13';*

*qui ont*

*création d'une sixième table*

*CREATE TABLE commission ( code\_commission INTEGER PRIMARY KEY, libelle VARCHAR(30), pourcentage INTEGER );*

**Cours pour insérer les index :**

describe Etablissement;

create index nomEtab\_Index on Etablissement(NomEtablissement\_Etablissement(10)ASC);

index est rentré mais il est en null alors qu’il faudrait qu’il soit NOTNULL. Donc il faut effacer l’index et modifier NULL en NOTNULL

drop index NomEtab\_Index ON Etablissement;

describe Etablissement;

ALTER TABLE Etablissement MODIFY COLUMN NomEtablissement\_Etablissement varchar(25) NOT NULL;

describe Etablissement;

create index nomEtab\_Index on Etablissement(NomEtablissement\_Etablissement(10)ASC);

show tables;

use CAVE;

show tables;

create view ListeCru as select \* from vins ;

create view ListeBuveurs as select nombuv, adresse from buveurs;

select nombuv from ListeBuveurs;

insert into buveurs(nombuv,adresse) values("Patrick","St-Malo");

select nombuv from buveurs;

insert into buveurs(nombuv,adresse) values ("Patrick", "St-Malo");

select nombuv from buveurs;

delete from buveurs where numbuv = 9;

select nombuv from buveurs;

show grants for'root'@'localhost';

create user'consulte'@'localhost'identified by 'bonjour';

grant select on SallesDeConcerts.\* TO'consulte'@'localhost';

show grants for 'consulte'@'localhost';

use information\_schema;

show tables;

describe user\_privileges;

select \* from user\_privileges;

TP

Pour sauvegarder

mysqldump -u root -p CAVE > Cave.sql20210121

more Cave.sql20210121 // afficher les infos du fichier

AOC

INSERT INTO regionviticole (nomregion, numregion) VALUES ('Bourgogne','1');

INSERT INTO regionviticole (nomregion, numregion) VALUES ('Vallee du Rhone','2');

INSERT INTO regionviticole (nomregion, numregion) VALUES ('Bordelais','3');

INSERT INTO regionviticole (nomregion, numregion) VALUES ('Alsace','4');

INSERT INTO regionviticole (nomregion, numregion) VALUES ('Champagne','5');

INSERT INTO aoc (nomaoc, regionviticole\_numregion) VALUES ('Charmes-chambertin', 1);

INSERT INTO aoc (nomaoc, regionviticole\_numregion) VALUES ('Chateauneuf-du-pape',2);

INSERT INTO aoc (nomaoc, regionviticole\_numregion) VALUES ('Médoc',3);

INSERT INTO aoc (nomaoc, regionviticole\_numregion) VALUES ('Pomerol',3);

INSERT INTO aoc (nomaoc, regionviticole\_numregion) VALUES ('Riesling',4);

INSERT INTO aoc (nomaoc, regionviticole\_numregion) VALUES ('Romanee-Conti',1);

INSERT INTO aoc (nomaoc, regionviticole\_numregion) VALUES ('Saint-emilion',3);

INSERT INTO aoc (nomaoc, regionviticole\_numregion) VALUES ('Sauternes',3);

INSERT INTO aoc (nomaoc, regionviticole\_numregion) VALUES ('Champagne',5);

INSERT INTO aoc (nomaoc, regionviticole\_numregion) VALUES ('suCeBOulish',5);

delete from aoc where numaoc = 10;

select nomaoc, nomregion from aoc, regionviticole where aoc.regionviticole\_numregion = regionviticole.numregion;

UPDATE VINS SET aoc\_numaoc = 3 WHERE cru = "Petrus";

update vins set aoc\_numaoc = 3 where cru = "Château Pontet-Canet";

update vins set aoc\_numaoc = 3 where cru = "Château La Gaffelière";

update vins set aoc\_numaoc = 3 where cru = "Château "Château d'Yquem";

update vins set aoc\_numaoc = 1 where cru = "Domaine René Bouvier";

update vins set aoc\_numaoc = 1 where cru = "Romanée-Conti";

update vins set aoc\_numaoc = 2 where cru = "Château de Beaucastel";

update vins set aoc\_numaoc = 4 where cru = "Clos Saint Landelin";

**Autre TP transaction**

use cave2;

SELECT \* FROM buveurs;

SHOW GLOBAL VARIABLES LIKE "autocommit";

SHOW SESSION VARIABLES LIKE "autocommit";

set AUTOCOMMIT = 0;

insert into buveurs(nombuv,adresse) values ("Tiago", "Rennes");

select\* from buveurs;

rollback;

select \* from aoc;

INSERT INTO aoc (nomaoc, regionviticole\_numregion) VALUES ('Champagne',5);

insert into vins (cru, millesime, degre, aoc\_numaoc) values ("Petrus", "2020", "18","1");

select \* from vins;

commit;

set AUTOCOMMIT = 0;

SAVEPOINT P1;

update abus set quantite = 2 where numbuv = 7 and numvin = 8;

select \* from abus;

rollback to savepoint P1;

**PDO**

sudo systemctl restart apache2

sudo systemctl status apache2

Sur Lubuntu, on peut ouvrir une fenêtre explorateur de fichier, en mode administrateur, à partir d'une fenêtre terminal, avec la commande : sudo pcmanfm-qt

more /etc/passwd

more /etc/group

2.4 Vérifier l’existence ou créer l’utilisateur apache2

USE mysql;

SHOW TABLES;

DESCRIBE user;

SELECT \* FROM user\G;

SELECT user, host, plugin, authentication\_string FROM mysql.user;

CREATE user "www-data"@"localhost" IDENTIFIED BY "apache2";

SELECT user, host, plugin, authentication\_string FROM mysql.user;

USE cave2;

SHOW PRIVILEGES;

GRANT SELECT, INSERT, UPDATE, DELETE ON cave2.\* TO "www-data"@"localhost";

SHOW GRANTS FOR 'www-data'@'localhost';

EXIT;

mysql -u www-data -p cave2

apache2

use cave2;

show tables;

select \* from vins;

sudo pcmanfm-qt

accès au dossier var / www / html / ………….

<http://localhost/caveIndex.htm>

**Révisions**

**sudo nano /etc/apt/apt.conf.d/proxy**

**mysql -u root -p**

**create database bar;**

**create table client (nom char(30), prenom char(30), age year);**

**create table vins (numVin int primary key not null auto\_increment, nomVin varchar(30) not null, annee year not null, degre int not null);**

**ALTER TABLE client add numClient int primary key;**

**create table serveur (numServeur int primary key, nomServeur char(30) not null, grade int not null);**

**insert into client (nom, prenom, age, telephone, adresse, cp, ville, numclient) value ('dupond', 'jean', '38', '0612202120', 'rueAugustin', '22100', 'quevert', '1');**

**insert into serveur (numServeur, nomServeur, grade) value ('3', 'aelia', '5');**

**insert into vins (numVin, nomVin, annee, degre) value ('1', 'sancerre', '2017','12');**

**delete from client where numClient = 1;**

**ALTER TABLE client add numVin int;**

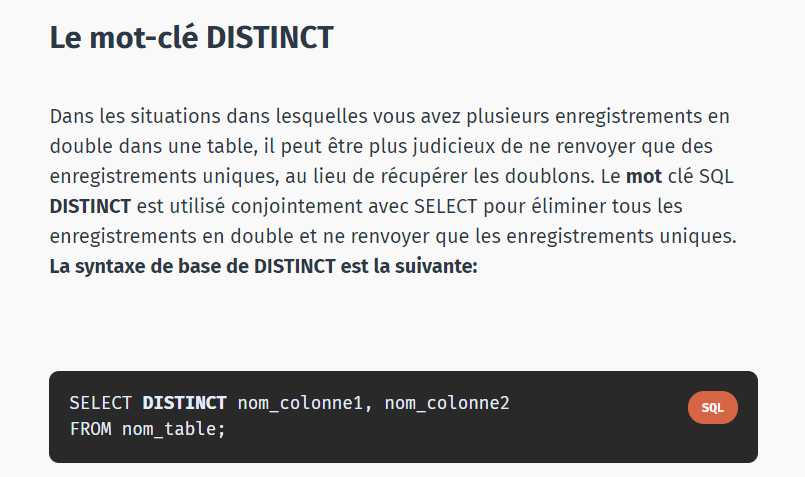
**UPDATE client SET nom = 'picard', prenom = 'jennifer', age = '38' WHERE numVin = 2;**

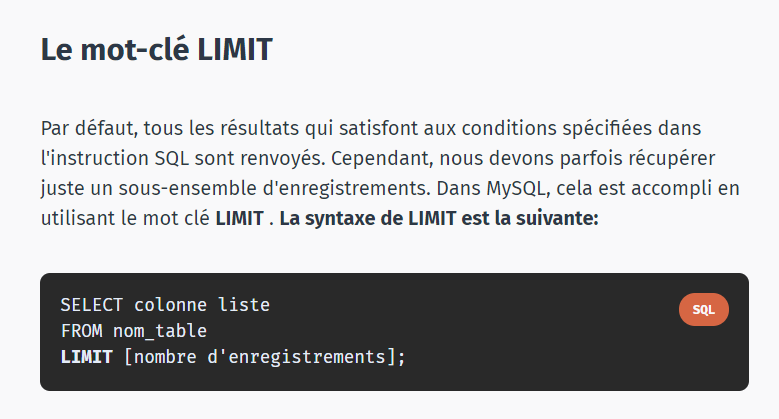
**select \* from client;**

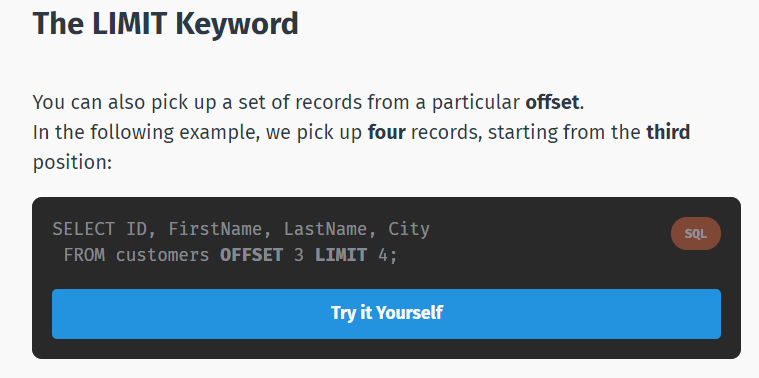
**UPDATE client SET numVin = '8' WHERE age = 36;**

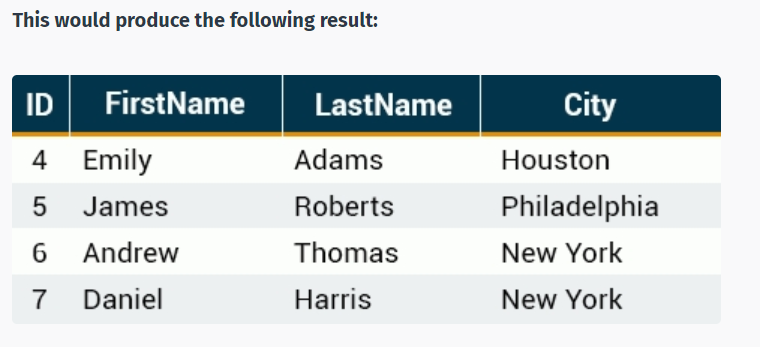
**select prenom from client where age = 38;**

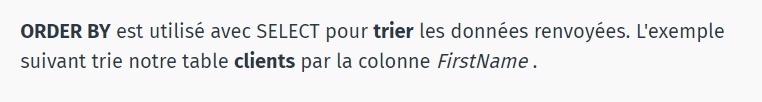
**select prenom,age from client where age<35;**

****

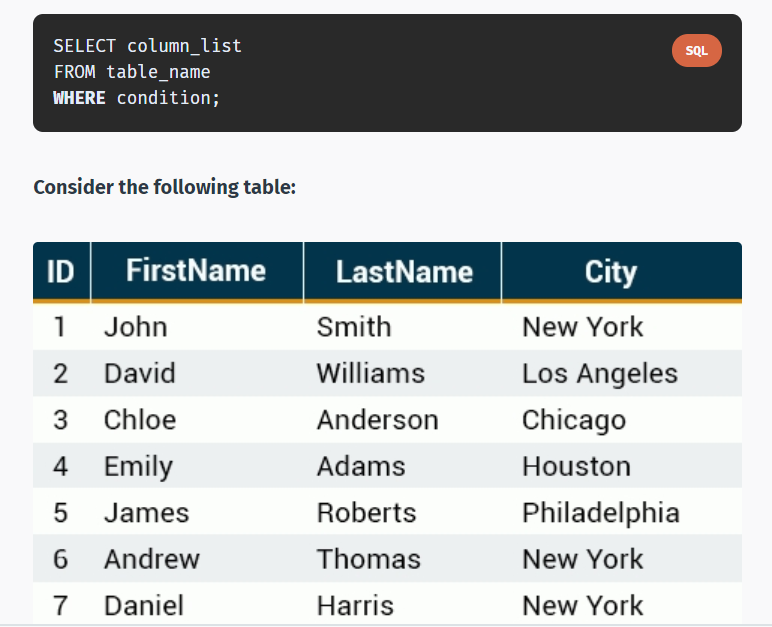
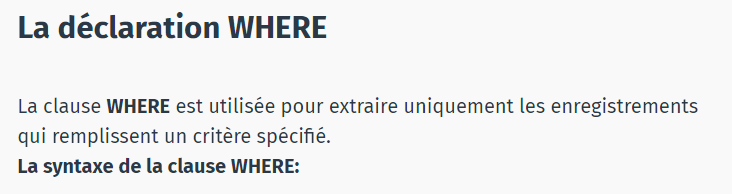
****

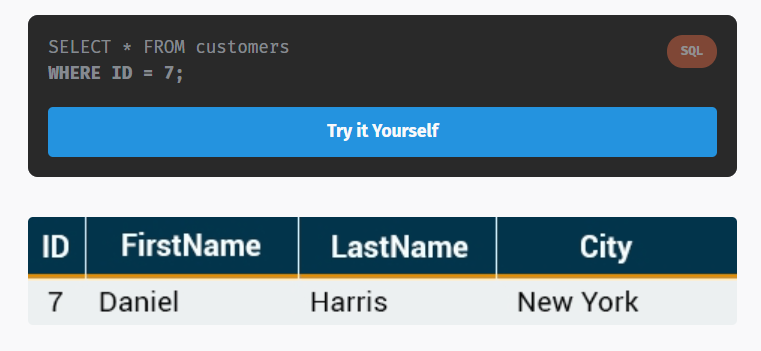
****

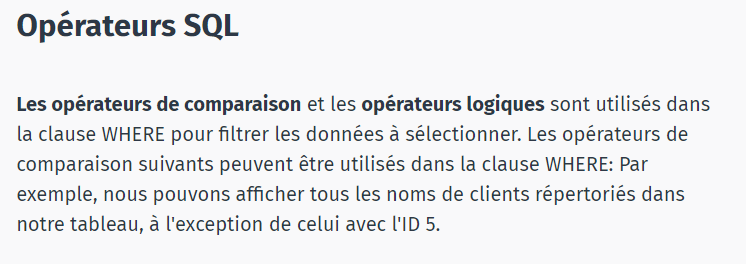
****

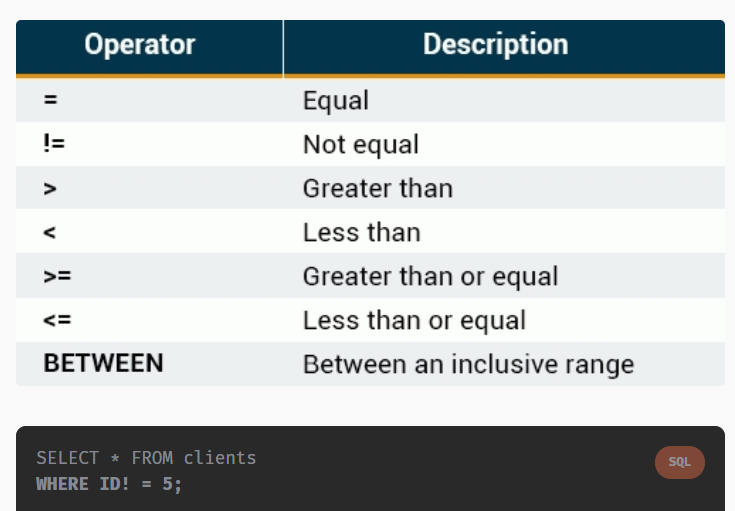
****

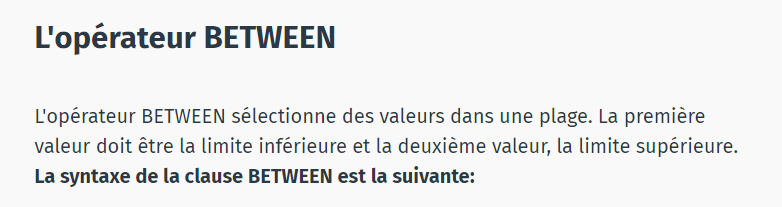
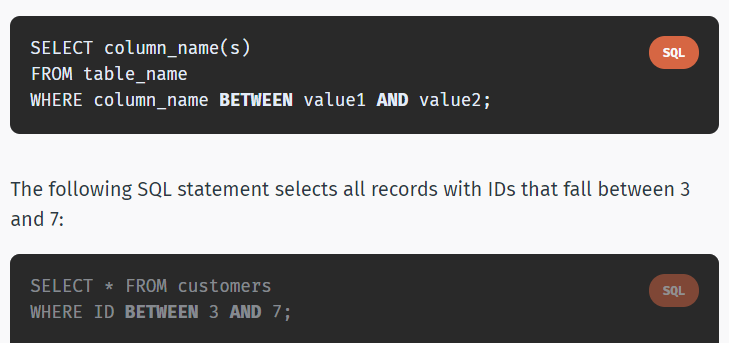
****

****

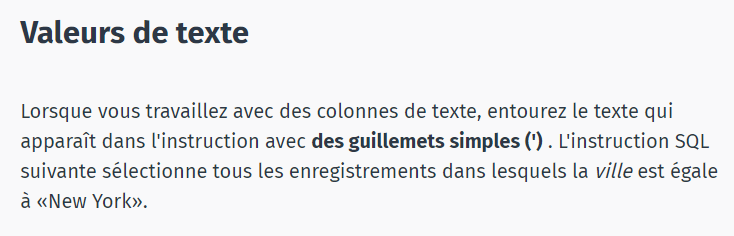
****

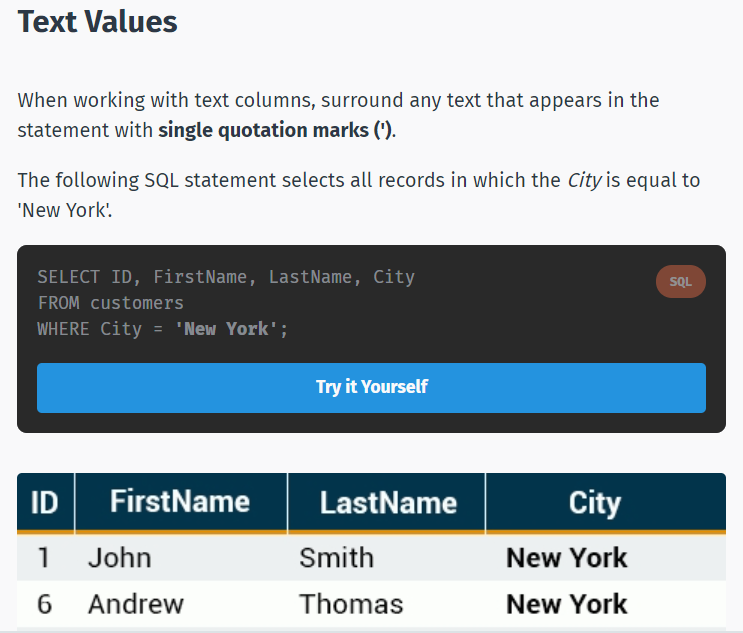
****

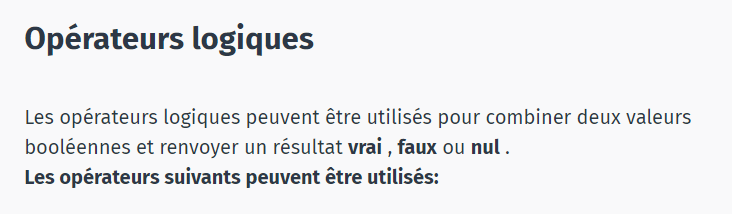
****

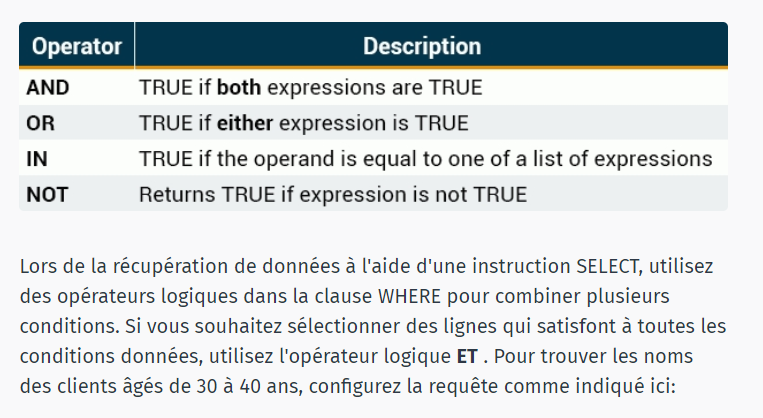
****

****

****

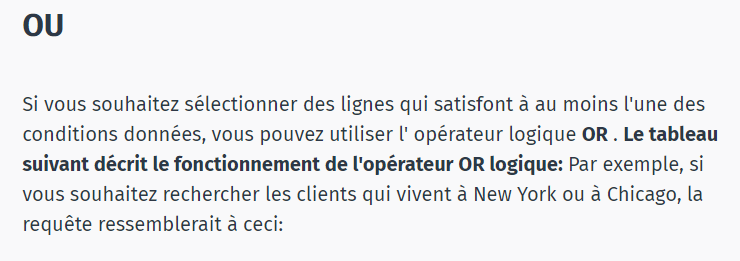
****

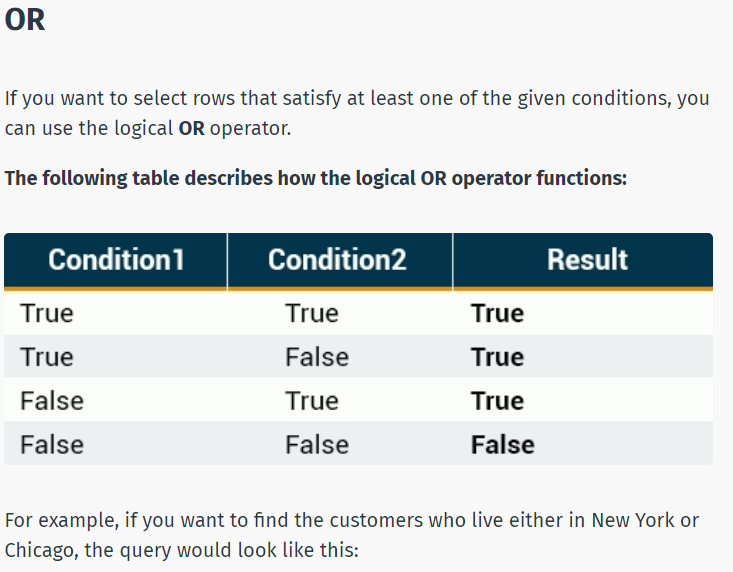
****

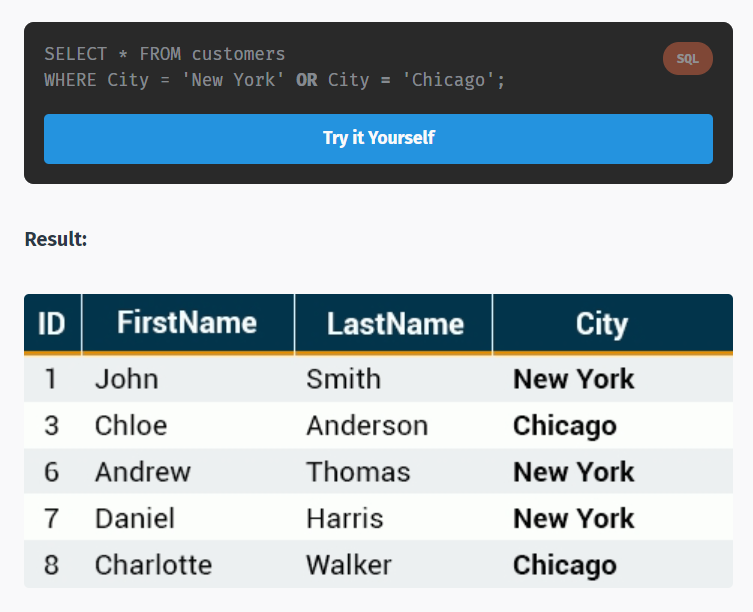
****

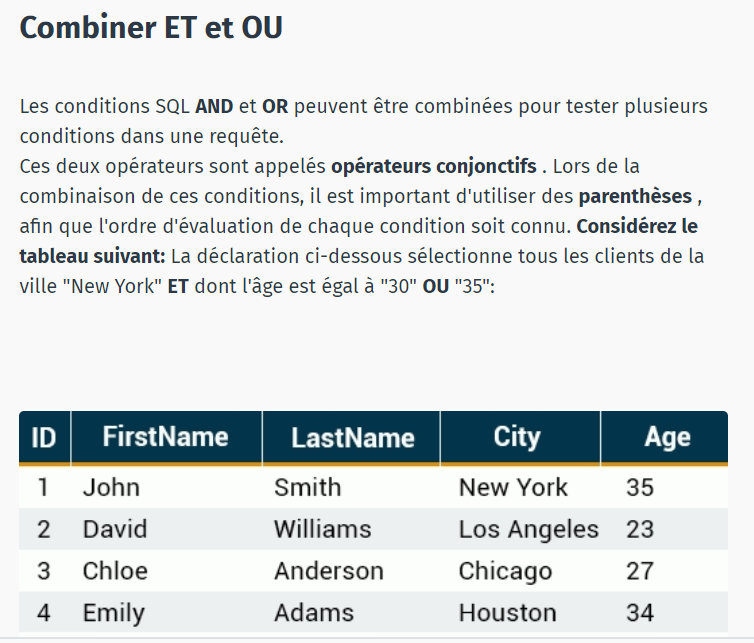
****

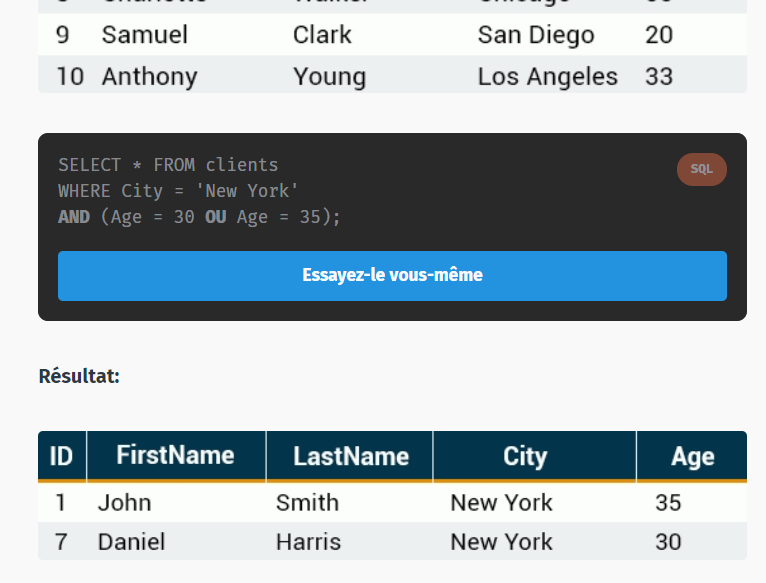
****

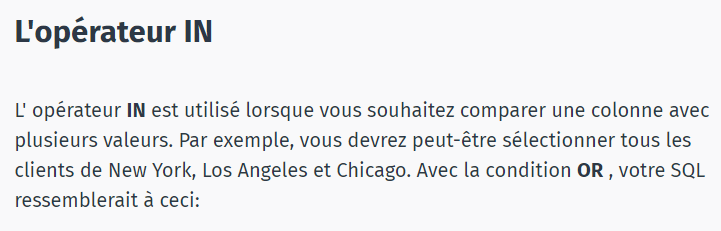
****

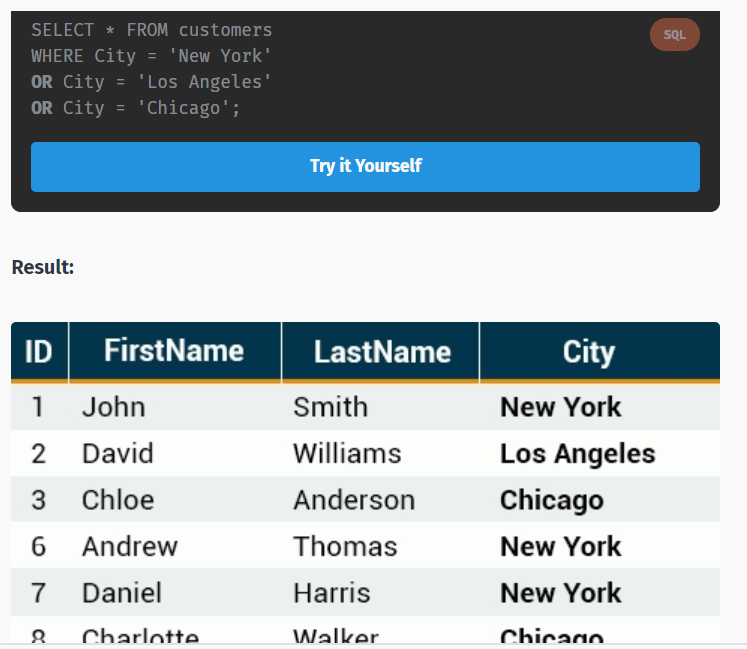
****

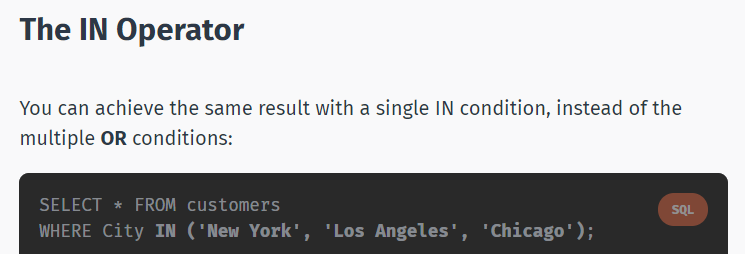
****

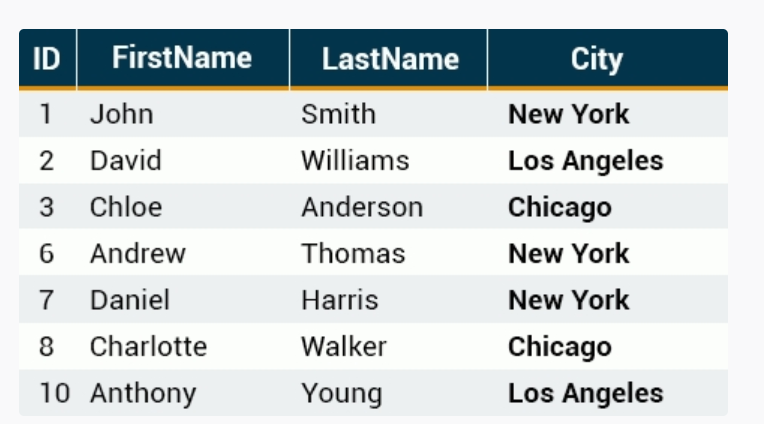
****

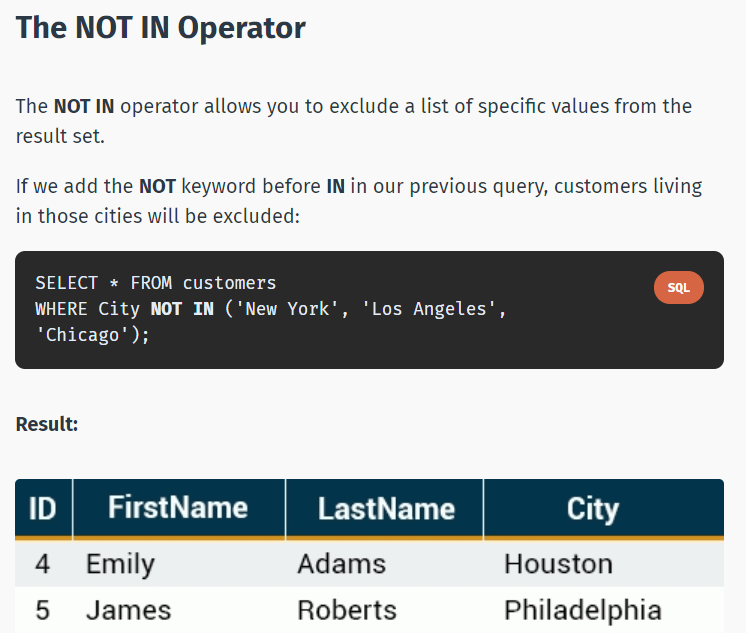
****

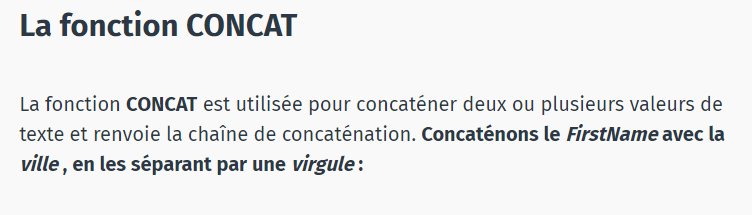
****

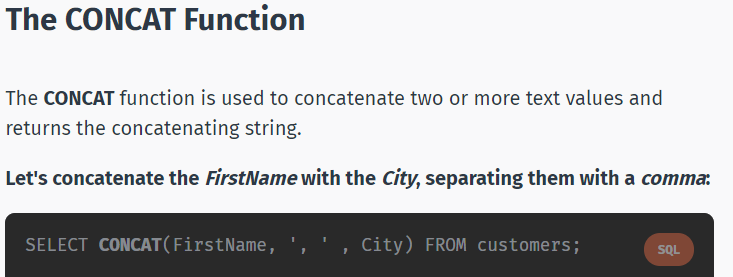
****

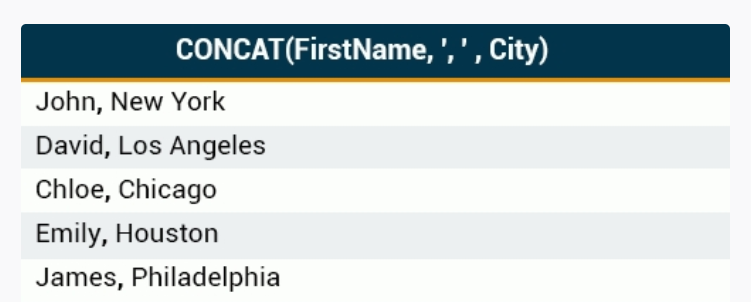
****

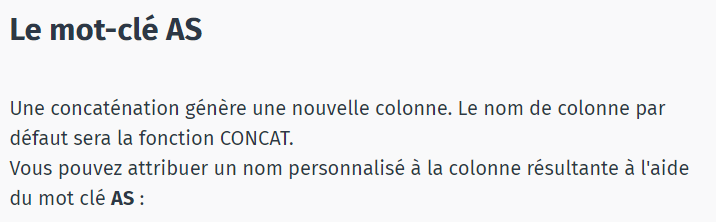
****

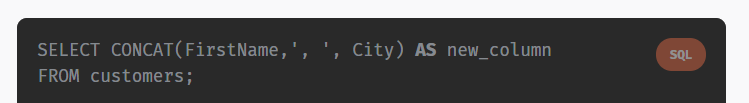
****

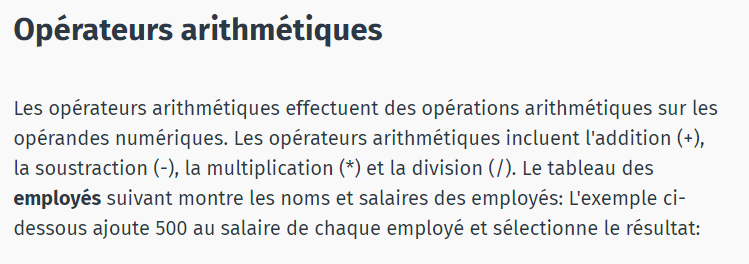
****

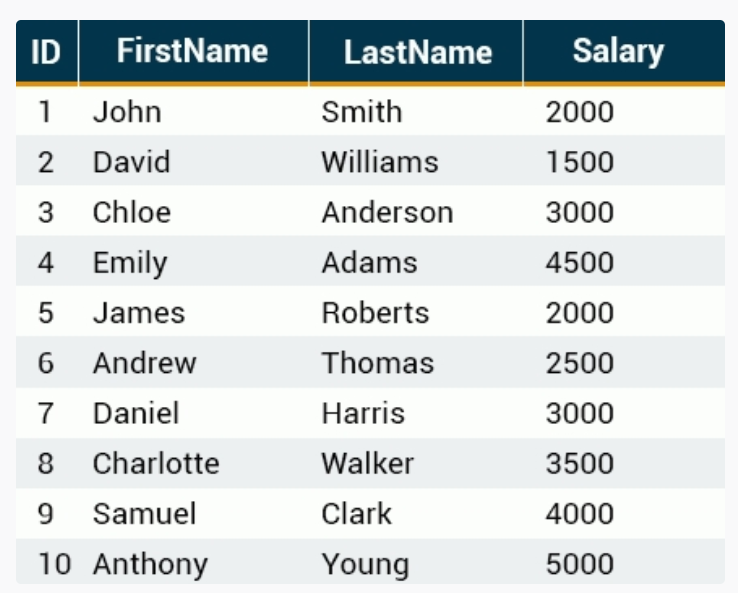
****

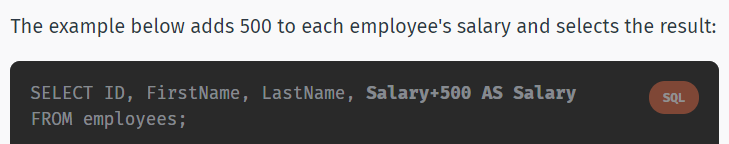
****

****

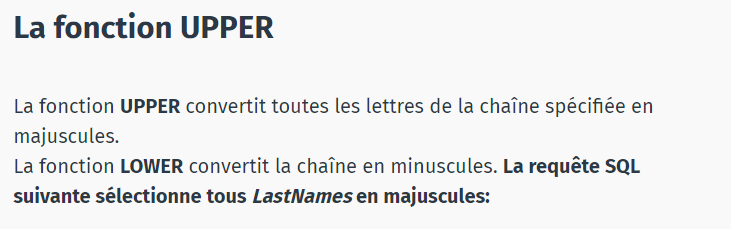
****

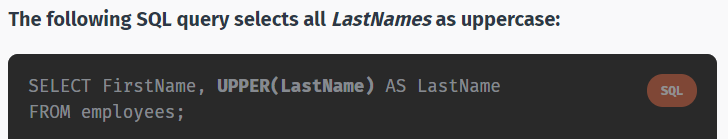
****

****

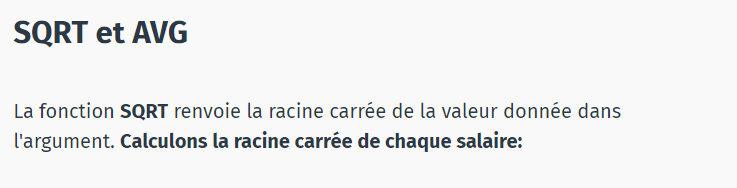
****

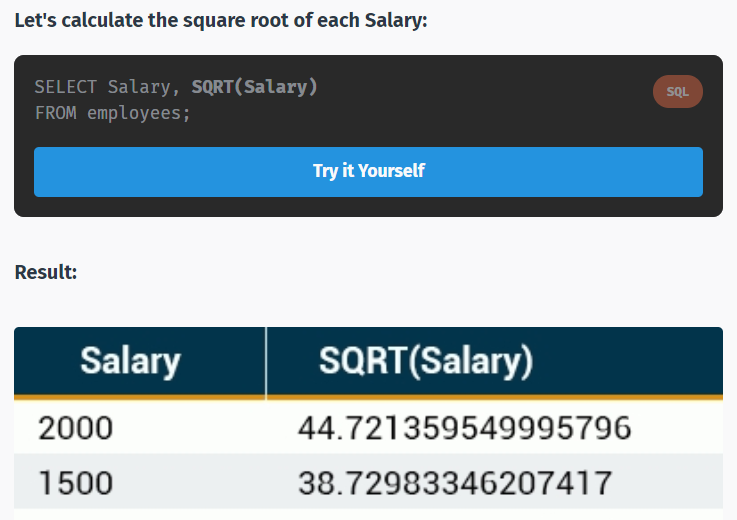
****

****

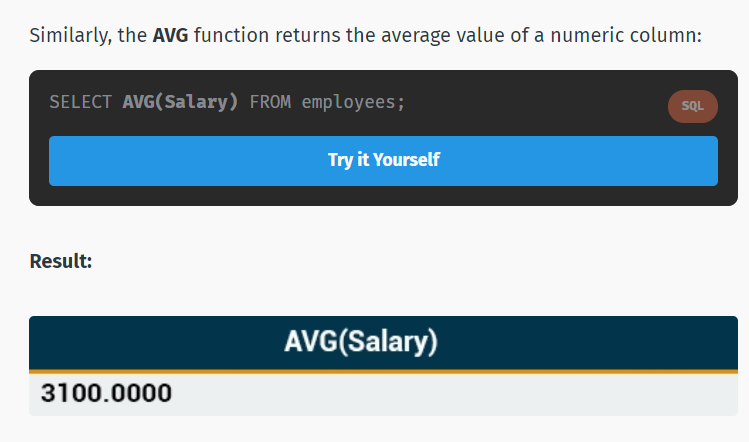
****

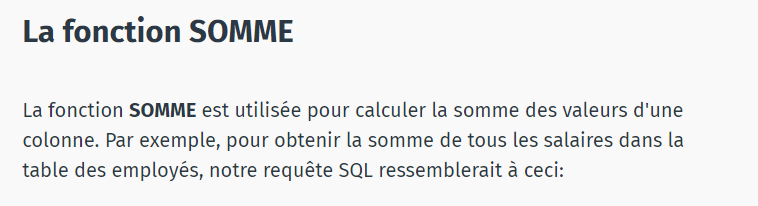
****

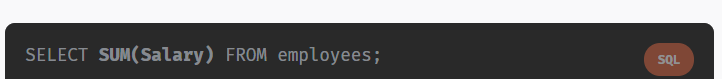
****

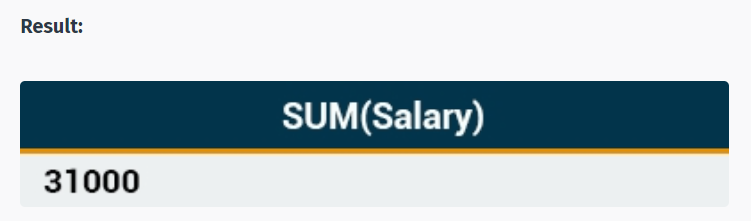
****

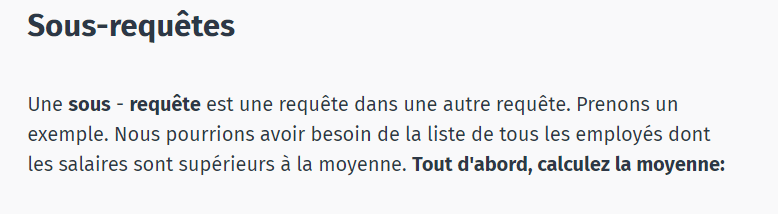
****

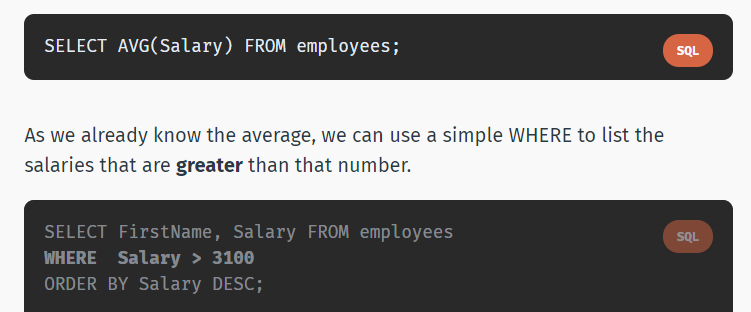
****

****

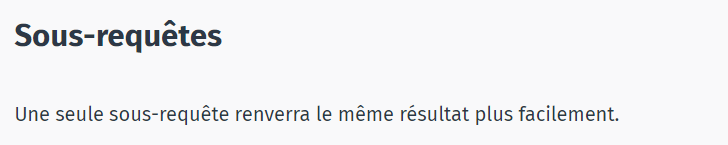
****

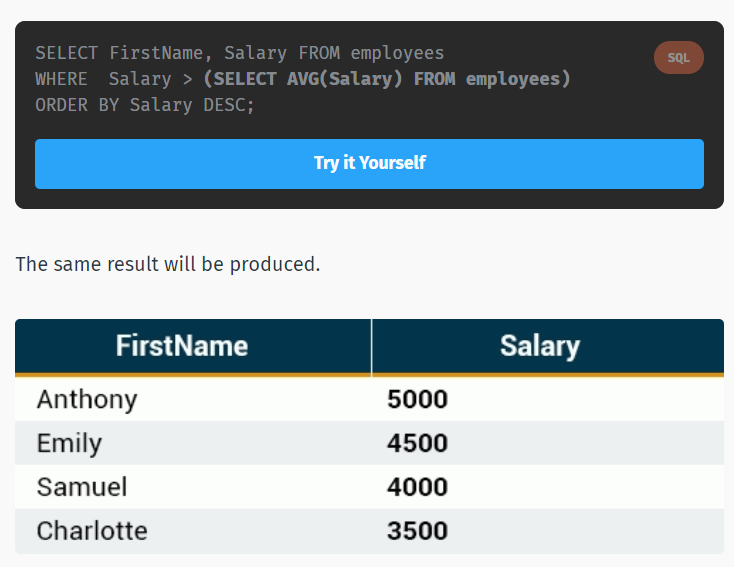
****

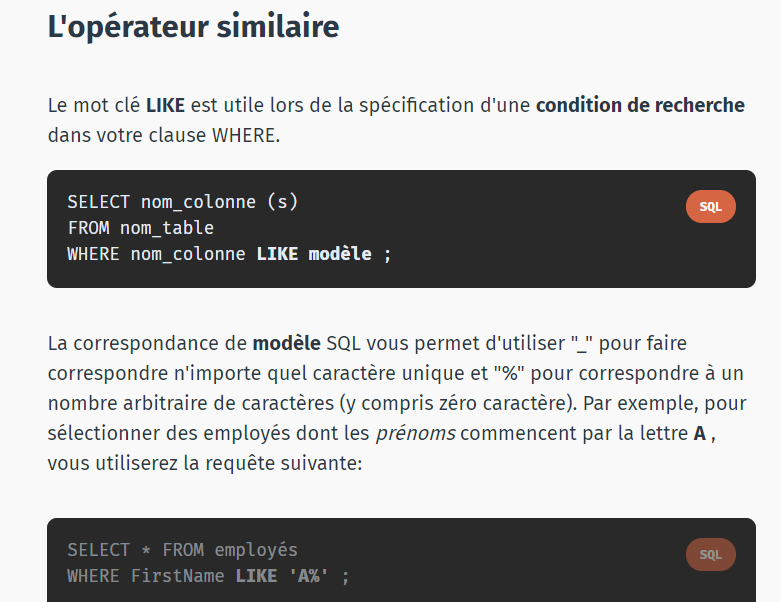
****

****

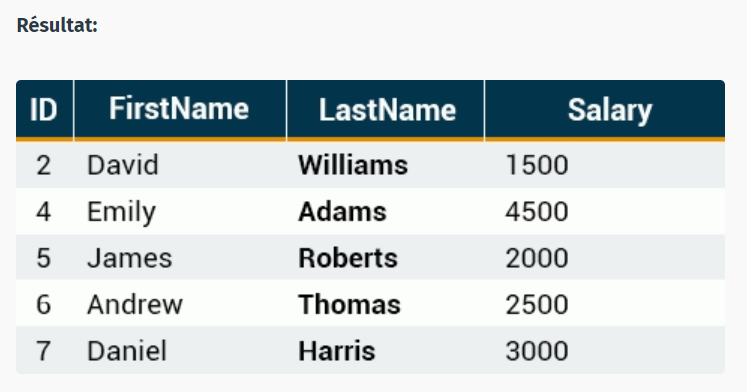
****

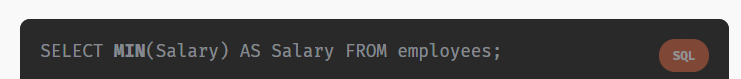
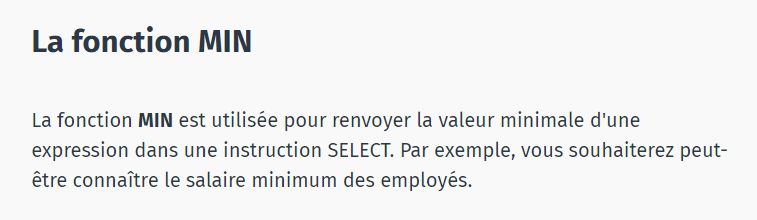
****

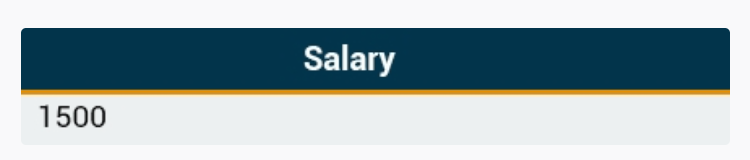
****

****

****

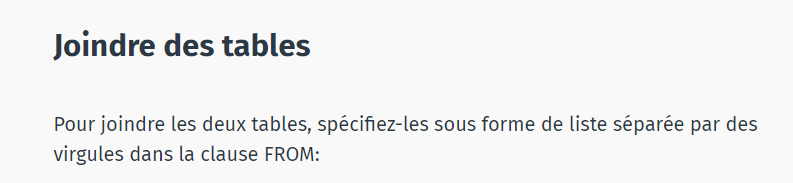
****

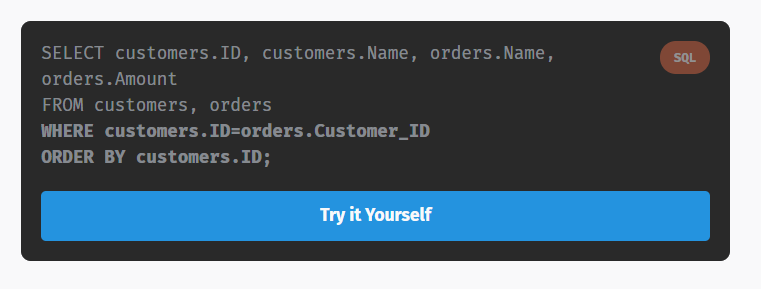
****

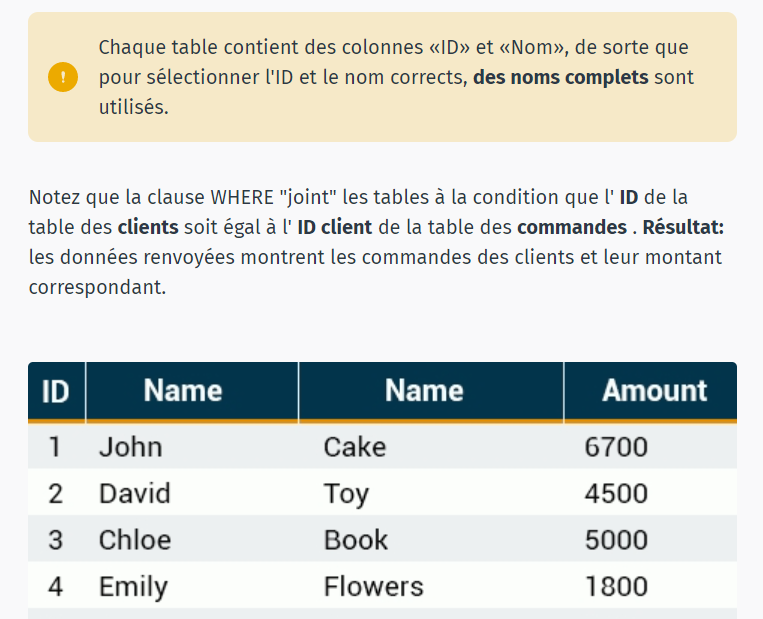
****

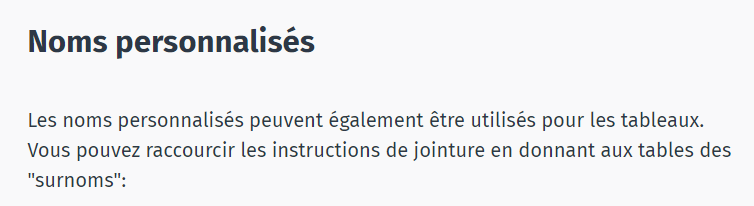
****

**SELECT \* FROM Appartments WHERE status = ‘not rented’ AND price > (SELECT AVG(price) FROM Appartments) ORDER BY price ;**

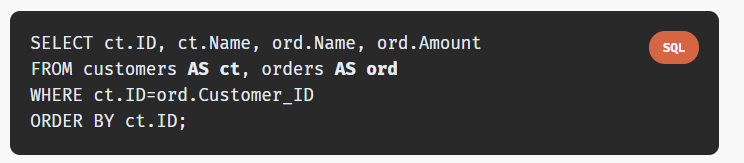
****

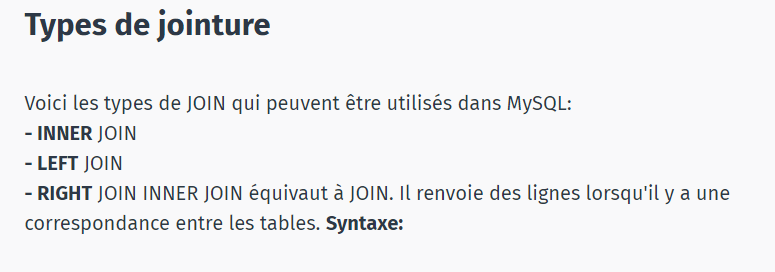
****

****

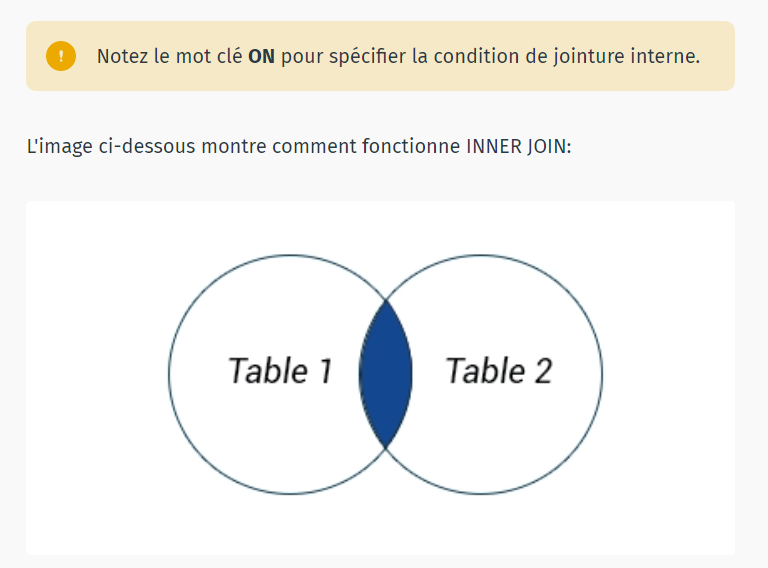
****

**Rappel : AS pour créer une nouvelle colonne**

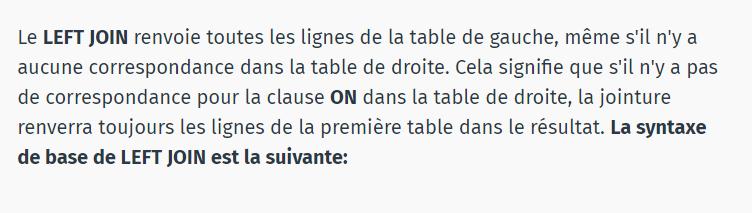
****

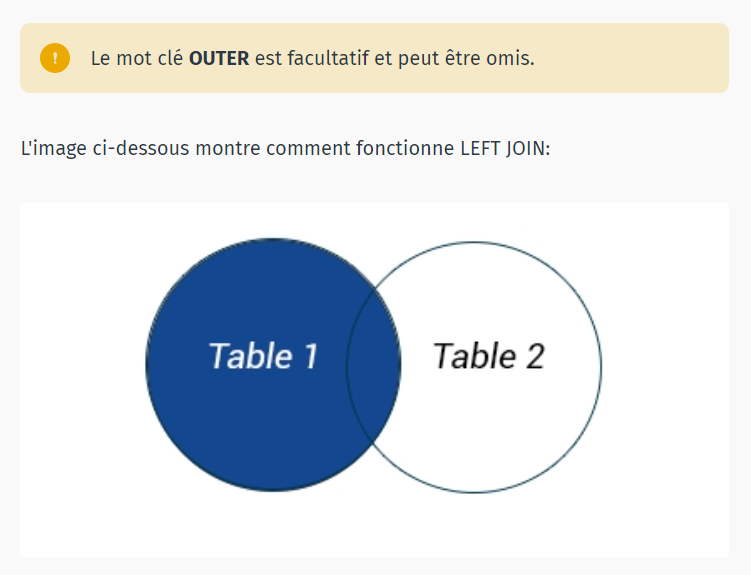
****

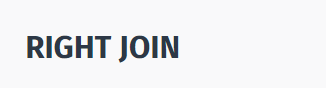
****

****

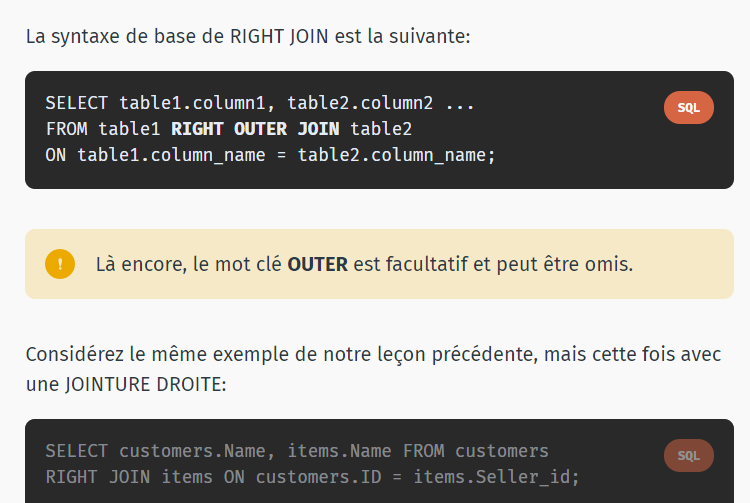
****

****

****

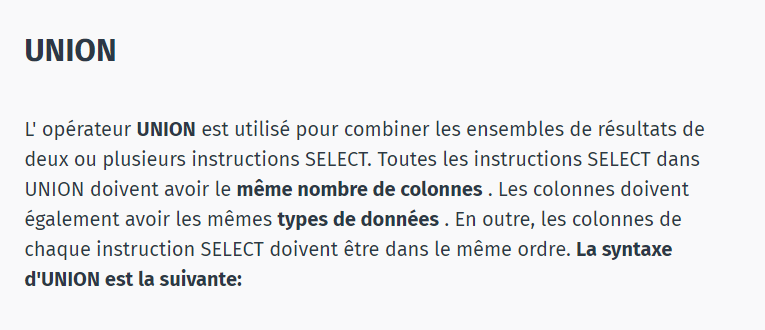
****

****

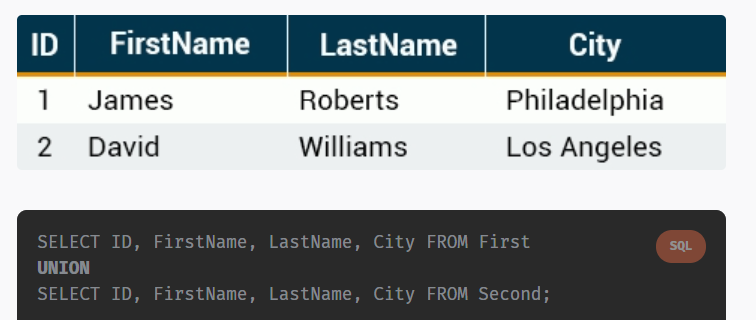
****

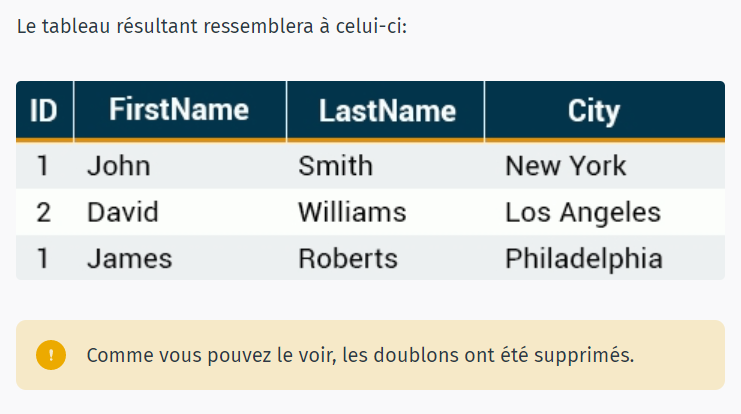
****

****

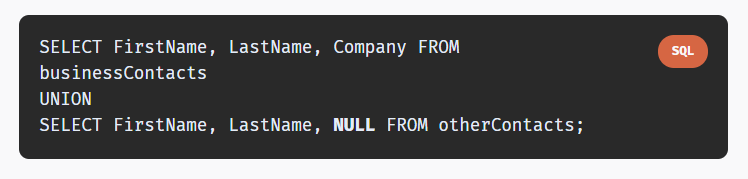
****

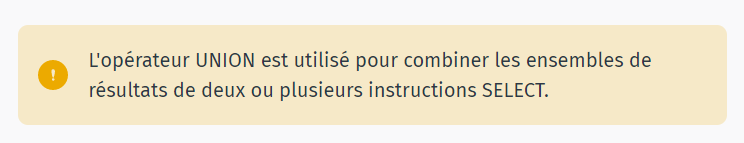
****

****

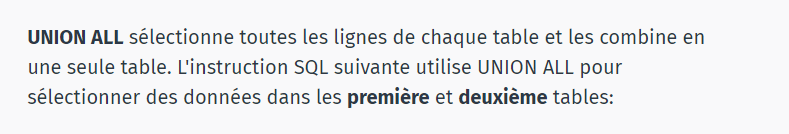
****

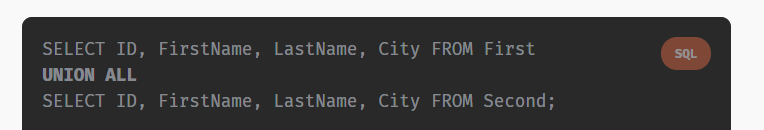
****

****

****

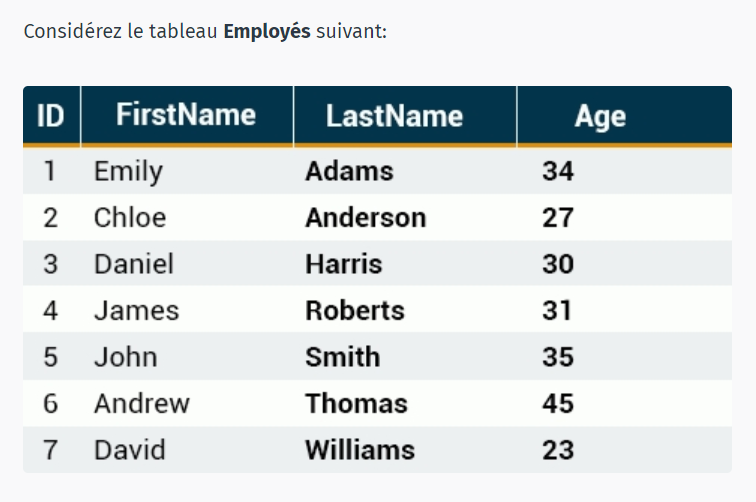
****

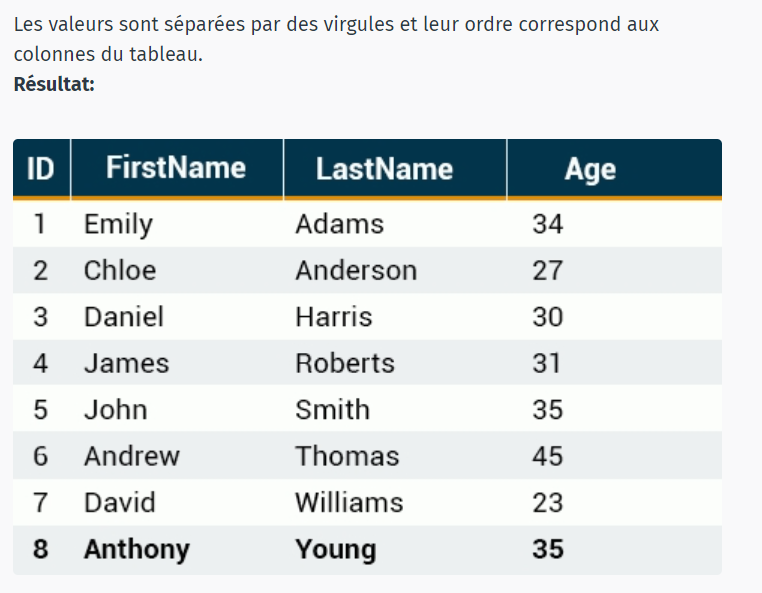
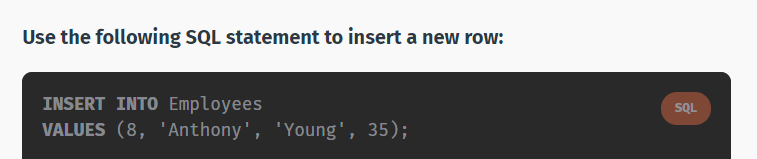
****

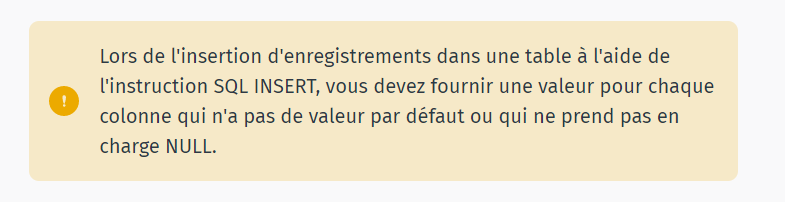
****

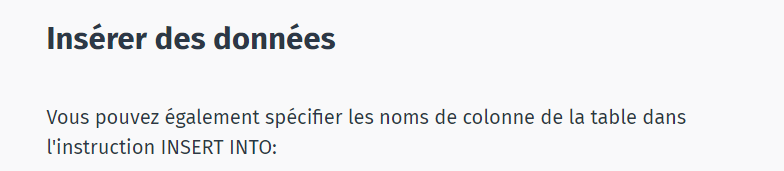
****

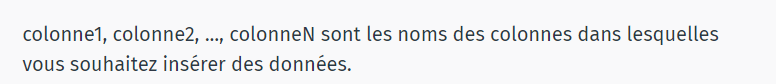
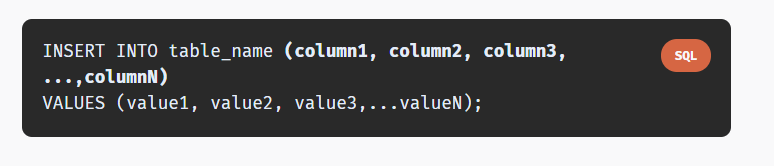
****

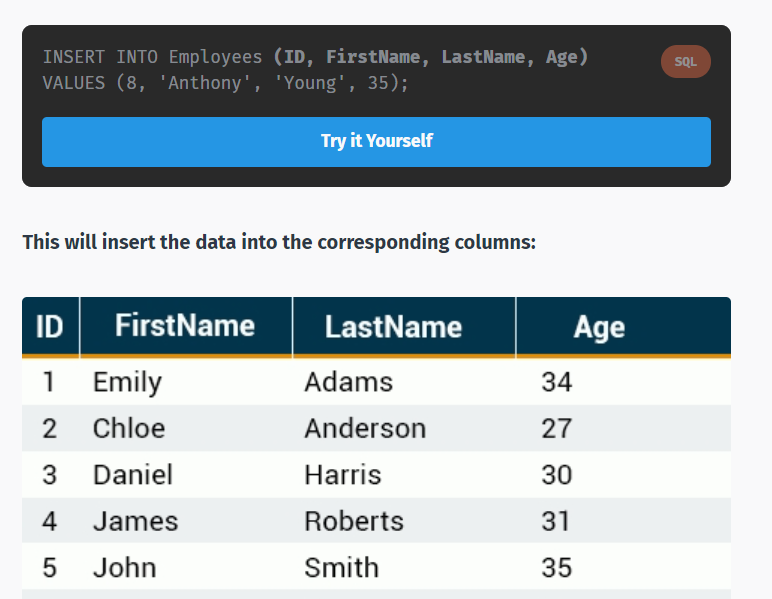
****

****

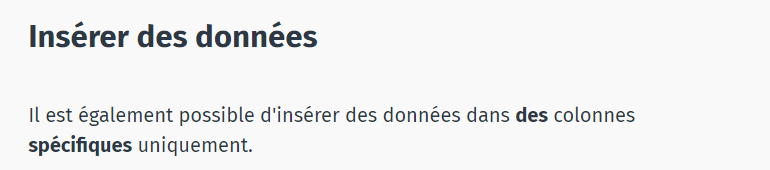
****

****

****

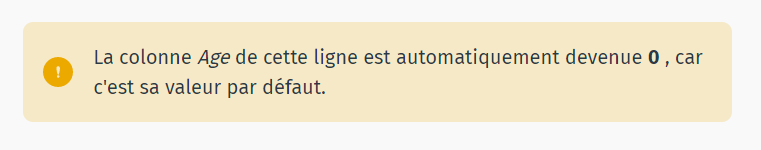
****

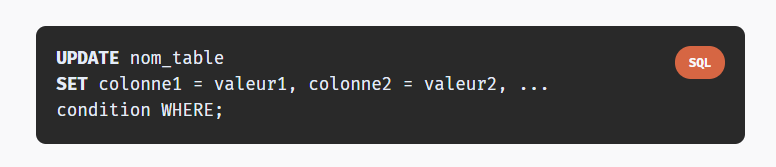
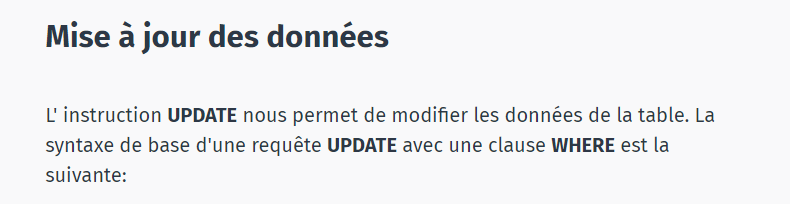
****

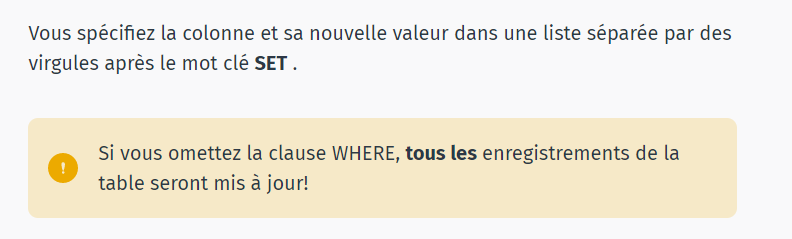
****

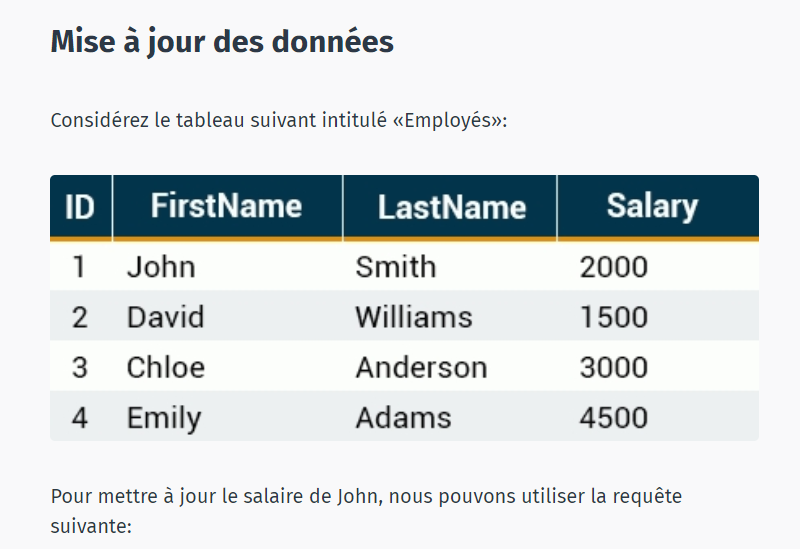
****

****

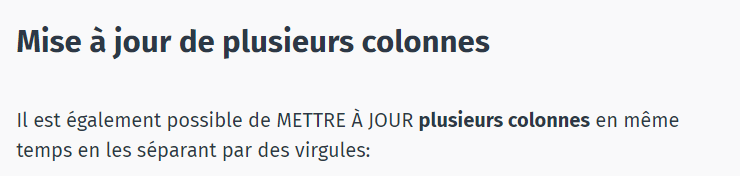
****

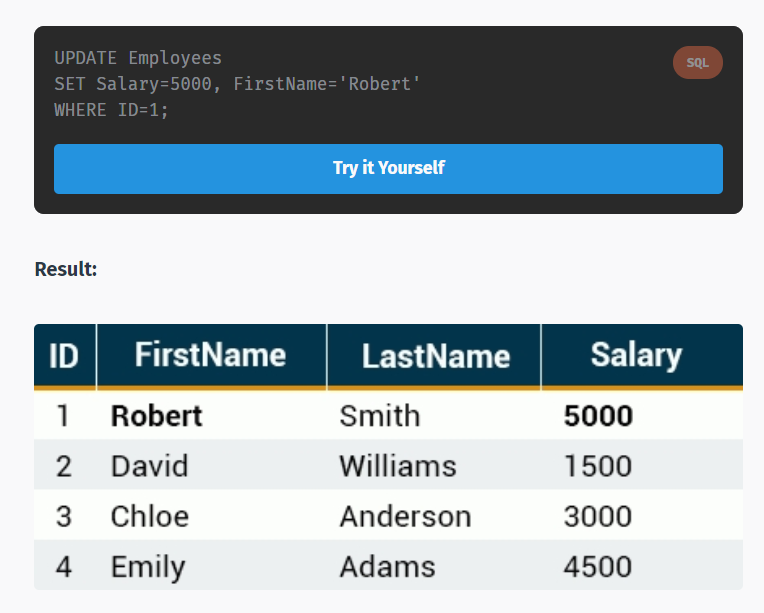
****

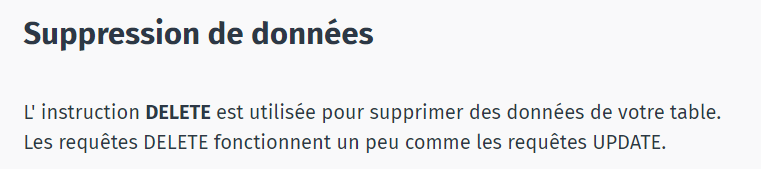
****

****

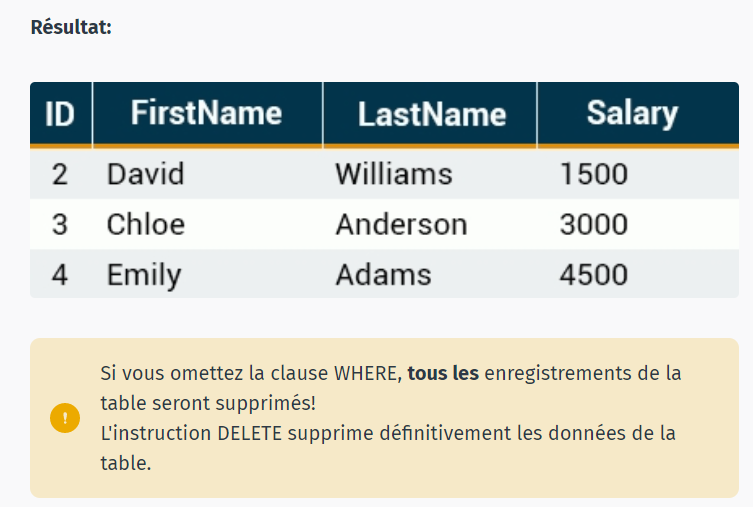
****

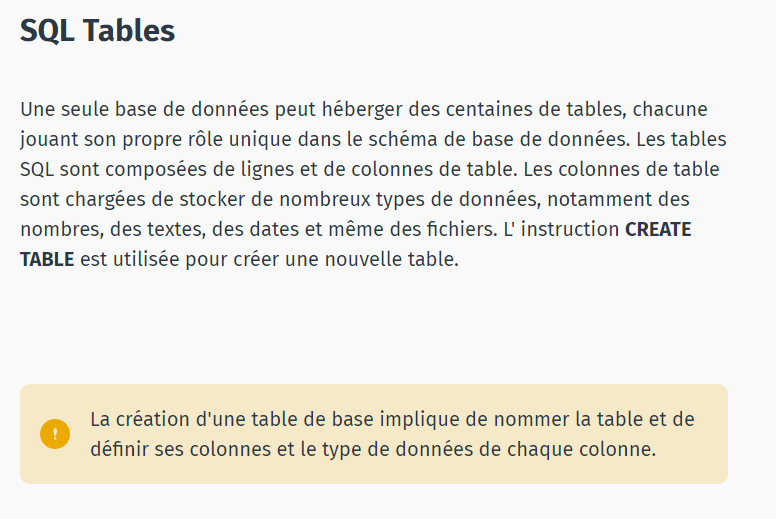
****

****

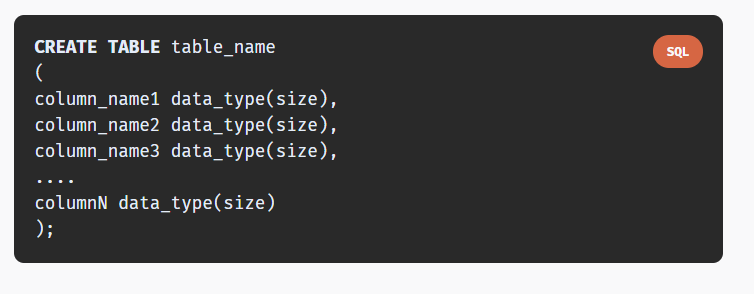
****

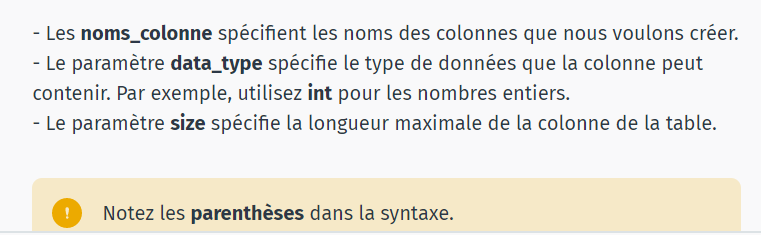
****

****

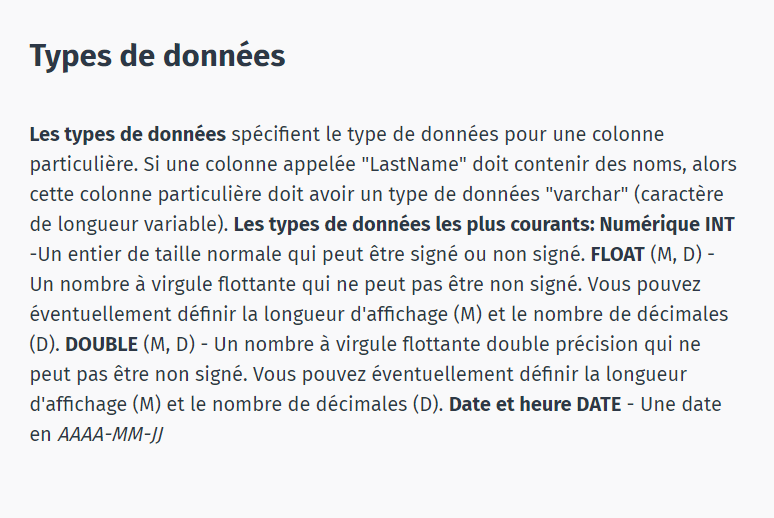
****

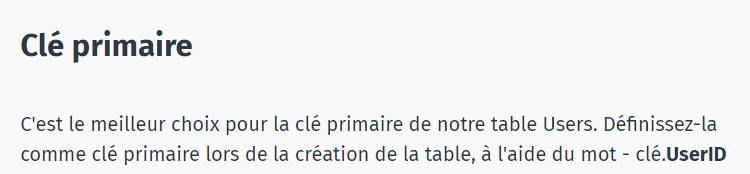
****

****

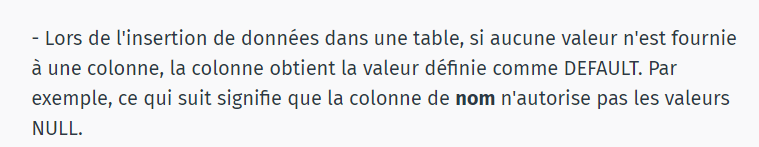
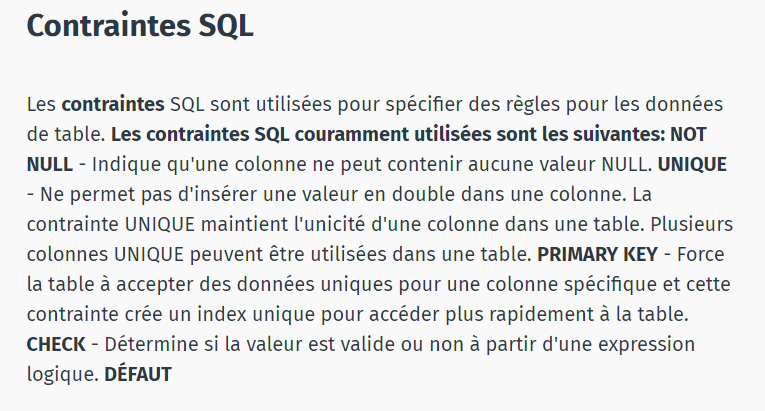
****

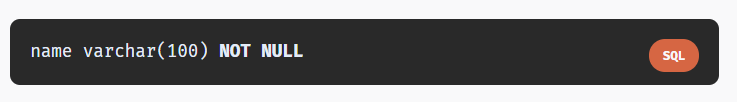
****

****

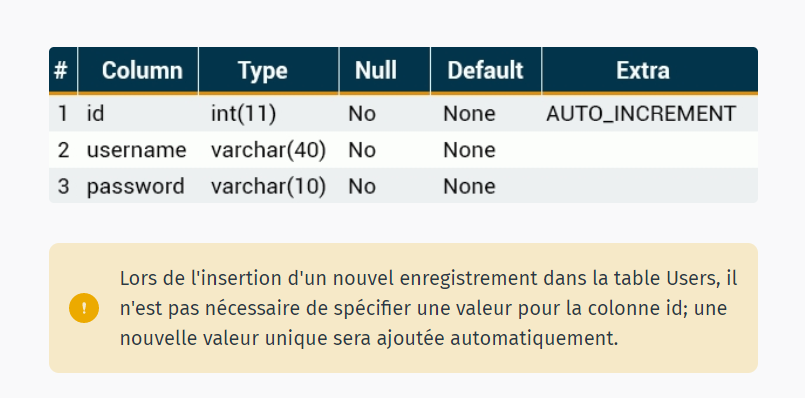
****

****

****

****

****

****

**Pousybook**

show tables ;

**changer le nom d’une colonne**

alter table livre change nombre\_pages nombre\_pages\_l int not null;

alter table livre change nom\_a nom\_al varchar(100) not null;

**Création de la table livre :**

create table livre (id int primary key auto\_increment, titre\_l varchar(100) not null, isbn\_l int not null, nombre\_pages int not null, edition\_l varchar(100) not null, annee\_edition\_l int not null, langue\_l varchar(50) not null, resume\_l varchar(255) not null, nom\_a varchar(100) not null, constraint fk\_nom\_a foreign key(nom\_a) references auteur(nom\_a));

**ajouter une contrainte de clé étrangère**

alter table livre add constraint fk\_nom\_a foreign key (nom\_a) references auteur (nom\_a);

**mysql -u root -p pouzybook**

insert into categorie (categorie\_c) value ("fanstastique");

insert into emprunt (dispo\_em, date\_em, delais\_em, id\_inscrit, id\_exemplaire) value ("oui","070321","45", "1","1");

insert into exemplaire (ref\_e, id\_inscrit) value ("47566","1");

insert into inscrit (date\_naissance\_i, nom\_i, prenom\_i, rue\_i, cp\_i, ville\_i, email\_i, tel\_i, mot\_de\_passe\_i) value ("160884","Picard","Jérôme", "13 rue Augustin Le Guen", "22100", "Quévert", "jey22@hotmail.fr", "0618300934", "bonjour");

insert into livre (titre\_l, isbn\_l, nombre\_pages\_l, edition\_l, langue\_l, nom\_a, id\_exemplaire, categorie\_c) values ("Le seigneur des anneaux", "45845", "282", "Folio", "anglais", "Tolkien", "1", "fantastique");

select titre\_l from livre;

insert into inscrit (date\_naissance\_i, nom\_i, prenom\_i, rue\_i, cp\_i, ville\_i, email\_i, tel\_i, mot\_de\_passe\_i) values ("19840816", "Gorge", "Jennifer", "11 rue Augustin Le Guen", "22101", "Dinan", "jeny35@orange.fr", "0630053456", "hello");

insert into inscrit (date\_naissance\_i, nom\_i, prenom\_i, rue\_i, cp\_i, ville\_i, email\_i, tel\_i, mot\_de\_passe\_i) value ("060789","Picard","David", "La Fourberie", "22100", "Taden", "david-22@hotmail.fr", "0618020102", "coucou");

insert into inscrit (date\_naissance\_i, nom\_i, prenom\_i, rue\_i, cp\_i, ville\_i, email\_i, tel\_i, mot\_de\_passe\_i) value ("020490","Brault","Typhaine", "La Fourberie", "22100", "Taden", "typhaine-22@hotmail.fr", "0610121402", "coucou2");

insert into inscrit (date\_naissance\_i, nom\_i, prenom\_i, rue\_i, cp\_i, ville\_i, email\_i, tel\_i, mot\_de\_passe\_i) value ("19520113","Picard","René", "l'Ecotay", "22100", "Brusvily", "renepicard@gmail.com", "0617101512", "garage");

insert into inscrit (date\_naissance\_i, nom\_i, prenom\_i, rue\_i, cp\_i, ville\_i, email\_i, tel\_i, mot\_de\_passe\_i) value ("19500503","Picard","Sylvie", "l'Ecotay", "22100", "Brusvily", "sylvie35@gmail.com", "0630021545", "bureau");

**mysql -u root -p pouzybook2**

insert into categorie (categorie\_c) values ("roman");

insert into exemplaire (users\_id) values ("3");

insert into langue (langue\_l) values ("italien");

insert into langue (langue\_l) values ("allemand");

insert into langue (langue\_l) values ("anglais");

insert into langue (langue\_l) values ("francais");

insert into livre (users\_id, langue\_id, exemplaire\_id, categorie\_id, titre\_l, nombre\_pages\_l, edition\_l, auteur) values ("3", "1", "7", "1", "Le seigneur des anneaux", "452", "Folio", "Tolkien");

**Procédures stockées**

use cave2;

SET GLOBAL log\_bin\_trust\_function\_creators = 1;

DELIMITER //

CREATE FUNCTION nbBuveur1vin (n SMALLINT) RETURNS SMALLINT

BEGIN

DECLARE resultat SMALLINT;

SELECT COUNT(DISTINCT numbuv) INTO resultat FROM abus WHERE quantite >= n;

RETURN resultat;

END //

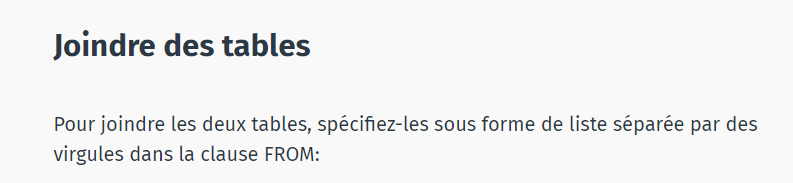
DELIMITER ;

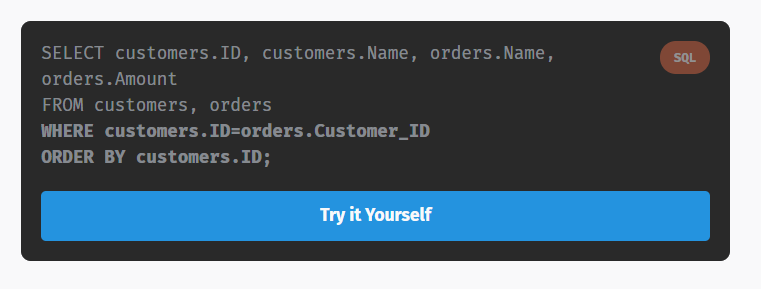
**Pour les appeler**

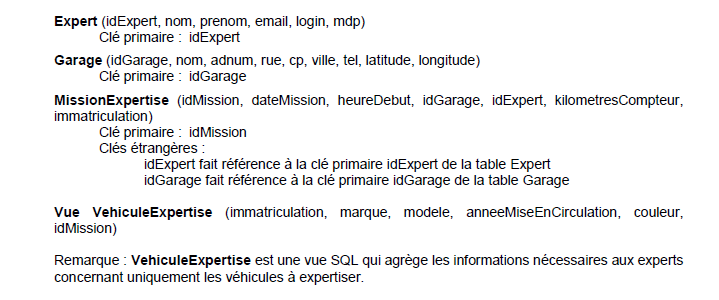
**Select pour les fonctions**

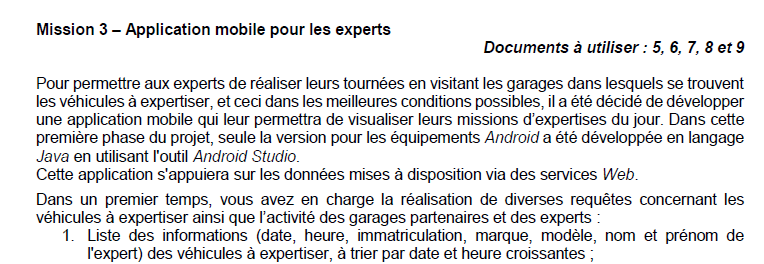
**Call pour les procédures**

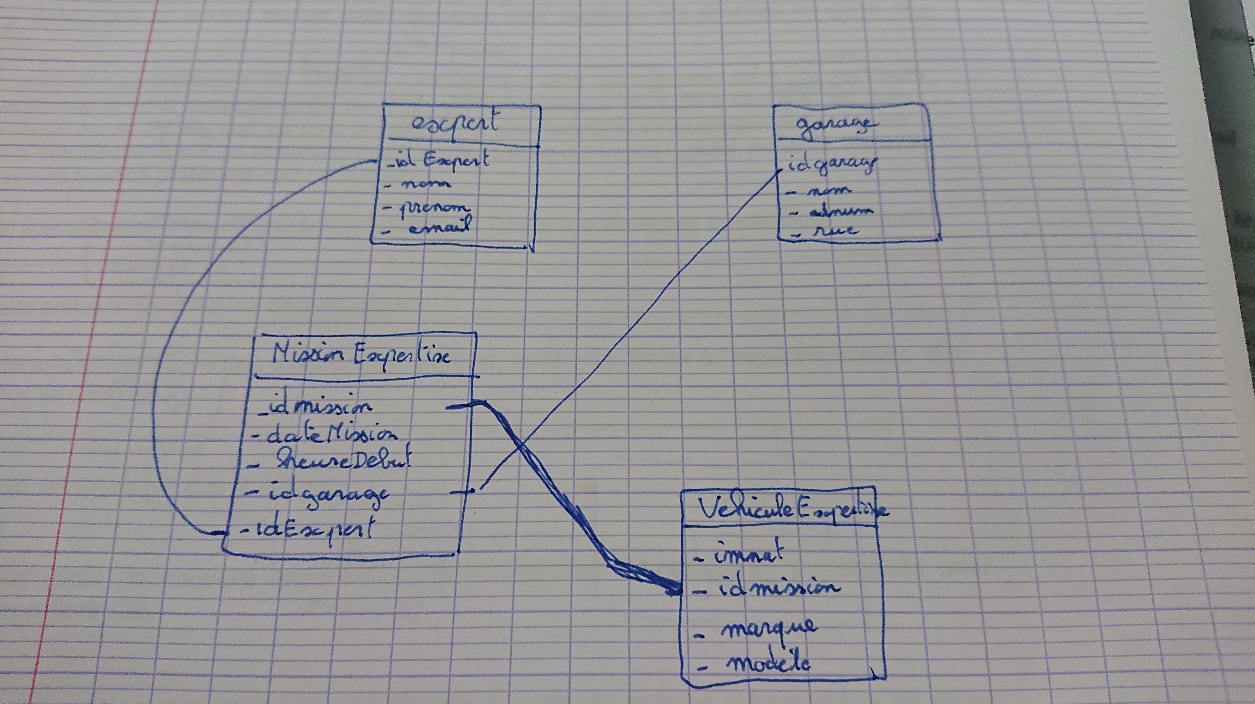
**Exercice Restiloc**

****

****







**Réponse :**

SELECT MissionExpertise.idmission MissionExpertise.dateMission MissionExpertise.heureDebut VehiculeExpertise.idmission VehiculeExpertise.marque VehiculeExpertise.immatriculation Vehiculeexpertise.modele Expert.nom Expert.prenom as journee FROM Expert MissionExpertise VehiculeExpertise WHERE VehiculeExpertise.idmission = MisionExpertise.idmission AND MissionExpertise.idExpert = Expert.idExpert ORDER by DateMission , heureDebut ASC ;

**ProcéduresStockées**

mysql -u root -p cave2

use cave2;

***C’est-à-dire que par défault, MySQL ne supporte pas les fonctions qui retournent pas un résultat déterministe (toujours le même) ou qui lisent des données avec SQL !***

**SET GLOBAL log\_bin\_trust\_function\_creators = 1;**

DELIMITER //

CREATE FUNCTION nbBuveur1vin (n SMALLINT) RETURNS SMALLINT

-> BEGIN

-> DECLARE resultat SMALLINT;

-> SELECT COUNT(DISTINCT numbuv) INTO resultat FROM abus WHERE quantite >= n;

-> RETURN resultat;

-> END //

DELIMITER ;

select nbBuveur1vin(1);

DELIMITER //

CREATE PROCEDURE sumQuantiteVin (IN nom VARCHAR(25), OUT resultat SMALLINT)

-> BEGIN

-> -- SELECT SUM(quantite) INTO resultat FROM abus, vins WHERE

-> -- vins.numvin = abus.numvin

-> -- and cru = nom;

-> SELECT SUM(quantite) INTO resultat FROM abus WHERE numvin IN (SELECT numvin FROM vins WHERE cru = 'nom');

-> END //

DELIMITER ;

**Triggers**

mysql -u root -p cave2

alter table vins add puHTbouteille integer;

describe vins;

select \* from vins;

update vins set puHTbouteille = 225 where numvins = 1;

update vins set puHTbouteille = 25 where numvins = 2;

……

update vins set puHTbouteille = 12 where numvins = 15;

alter table vins add puHTverre float;

// Trigger

**2.2**

DROP TRIGGER IF EXISTS majPrixVerre;

DELIMITER //

CREATE TRIGGER majPrixVerre

-> AFTER INSERT ON buveurs

-> FOR EACH ROW

-> BEGIN

-> UPDATE vins set puHTverre = puHTbouteille / 7;

-> END //

DELIMITER ;

show triggers;

select \* from vins;

insert into buveurs (nombuv,adresse) values ("Thierry","Dinan");

select \* from vins;

**2.3**

DROP TRIGGER IF EXISTS majPrixBouteille;

DELIMITER //

CREATE TRIGGER majPrixBouteille

BEFORE UPDATE ON buveurs

FOR EACH ROW

BEGIN

UPDATE vins set puHTbouteille = puHTbouteille \*1.05;

UPDATE vins set puHTverre = puHTbouteille / 7;

END //

DELIMITER ;

insert into buveurs (nombuv,adresse) values ("Thierry","Dinan");

select \* from vins;

**2.4**

DROP TRIGGER IF EXISTS verifPrixBouteille;

DELIMITER //

CREATE TRIGGER verifPrixBouteille

BEFORE INSERT ON vins

FOR EACH ROW

BEGIN

if new.puHTbouteille < 0 then

SIGNAL SQLSTATE '80000' SET MESSAGE\_TEXT= "le prix de la bouteille ne peut pas être negatif" ;

end if;

END //

DELIMITER ;

insert into vins (numvins,cru,millesime,degre,aoc\_numaoc,puHTbouteille,puHTverre) values ('16','Chinon','2018','12','4','-4.5','2');

**2.5**

DROP TRIGGER IF EXISTS verifPrixBouteille;

DELIMITER //

CREATE TRIGGER verifPrixBouteille

BEFORE INSERT ON vins

FOR EACH ROW

BEGIN

if new.puHTbouteille < 0 then

SIGNAL SQLSTATE '80000' SET MESSAGE\_TEXT= "le prix de la bouteille ne peut pas être negatif" ;

end if;

END //

DELIMITER ;

insert into vins (numvins,cru,millesime,degre,aoc\_numaoc,puHTbouteille,puHTverre) values ('16','Chinon','2018','12','4','-4.5','2');

**2.6**

DROP TRIGGER IF Exists verifPrixBouteille;

DELIMITER //

CREATE TRIGGER verifPrixBouteille

BEFORE INSERT ON vins

FOR EACH ROW

BEGIN

IF NEW.puHTBouteille<0 THEN

SET NEW.puHTBouteille=(-1\*(NEW.puHTBouteille));

END IF;

END //

DELIMITER ;

insert into vins (numvins,cru,millesime,degre,aoc\_numaoc,puHTbouteille,puHTverre) values ('16','Chinon','2018','12','4','-4.5','2');

**2.7**

alter table vins modify column puHTbouteille float unsigned;

alter table vins modify column puHTVerre float unsigned;

alter table vins modify column degre float unsigned;

show triggers\G;

drop trigger verifPrixBouteille;

DROP TRIGGER IF verifDegre;

DELIMITER //

CREATE TRIGGER verifPrixBouteille

BEFORE INSERT ON vins

FOR EACH ROW

BEGIN

IF NEW.degre>=100 THEN

SIGNAL SQLSTATE '80000' SET MESSAGE\_TEXT= "le volume d’alcool ne peut dépasser 100’ " ;

END IF;

END //

DELIMITER ;

insert into vins (numvins,cru,millesime,degre,aoc\_numaoc,puHTbouteille,puHTverre) values ('16','Chinon','2018','102','4','-4.5','2');

**2.8**

DELIMITER //

CREATE TRIGGER verifCruMillesime

BEFORE INSERT ON vins

FOR EACH ROW

BEGIN

DECLARE resultat SMALLINT ;

SELECT COUNT(\*) INTO resultat FROM vins WHERE cru = NEW.cru AND millesime = NEW.millesime ;

IF (resultat>0) THEN

SIGNAL SQLSTATE '80000' SET MESSAGE\_TEXT= "Ce cru et millesime existe déjà dans les vins " ;

END IF;

END //

DELIMITER ;

Ou

ALTER TABLE vins ADD CONSTRAINT cruMillesime UNIQUE (cru,millesime) ;

**2.9**

DELIMITER //

CREATE TRIGGER sommeQuantiteBues

BEFORE INSERT ON abus

FOR EACH ROW

BEGIN

SET@somme=@somme+New.quantite;

END //

DELIMITER ;

SHOW TRIGGERS\G

SELECT \* FROM abus ;

SELECT \* FROM buveurs ;

SET @somme=’’;

Insert into abus values(10,4,2),(10,3,2),(8,8,2) ;

Select @somme AS ‘total des quantités bues’ ;

SET @somme =’’ ;

Insert into abus values (8,4,1),(10,10,2) ;

Select @somme AS ‘Total des quantités bues’ ;

**2.10**

**Version 1**

DROP TRIGGER IF EXISTS verifDegre;

DELIMITER //

CREATE TRIGGER verifDegre

BEFORE INSERT ON vins

FOR EACH ROW

BEGIN

IF (NEW.degre >= 100 AND NEW.millesime >= YEAR(CURDATE())) THEN

SELECT 'Le volume en alcool ne peut dépasser 100 et le millesime ne peut depasser annee courante' into @tgerr;

SELECT 'a','a' into @tgerr;

ELSE IF NEW.degre >= 100 THEN

-- on garde un message d'erreur dans une variable

SELECT 'Le volume en alcool ne peut dépasser 100' into @tgerr;

-- on fait planter le trigger !!!

SELECT 'a','a' into @tgerr;

ELSE IF NEW.millesime >= YEAR(CURDATE()) THEN

SELECT 'Le millesime doit être inférieur à annee courante' into @tgerr;

SELECT 'a','a' into @tgerr;

END IF;

END IF;

END IF;

END //

DELIMITER ;

**Version 2**

DROP TRIGGER IF EXISTS verifDegre;

DELIMITER //

CREATE TRIGGER verifDegre

BEFORE INSERT ON vins

FOR EACH ROW

BEGIN

IF NEW.degre >= 100 AND NEW.millesime >= YEAR(CURDATE()) THEN

SELECT 'Le volume en alcool ne peut dépasser 100 et le millesime ne peut depasser annee courante' into @tgerr;

SELECT 'a','a' into @tgerr;

ELSEIF NEW.degre >= 100 THEN

-- on garde un message d'erreur dans une variable

SELECT 'Le volume en alcool ne peut dépasser 100' into @tgerr;

-- on fait planter le trigger !!!

SELECT 'a','a' into @tgerr;

ELSEIF NEW.millesime >= YEAR(CURDATE()) THEN

SELECT 'Le millesime doit être inférieur à annee courante' into @tgerr;

SELECT 'a','a' into @tgerr;

END IF;

END //

DELIMITER ;

TP HERITAGE

**Exemple :**

CREATE TABLE Orders (OrderID int NOT NULL, OrderNumber int NOT NULL, PersonID int, PRIMARY KEY (OrderID), FOREIGN KEY (PersonID) REFERENCES Persons(PersonID));

create table t\_vehicule\_vhc (vhc\_id int primary key auto\_increment unique, vhc\_immatriculation char(10));

alter table t\_vehicule\_vhc modify column vhc\_immatriculation char(10) UNIQUE not null;

CREATE TABLE t\_voiture\_vtr (vtr\_nombre\_place int, vhc\_id int primary key auto\_increment, FOREIGN KEY (vhc\_id) REFERENCES t\_vehicule\_vhc (vhc\_id));

CREATE TABLE t\_moto\_mto (mto\_cylindree float, vhc\_id int primary key, FOREIGN KEY (vhc\_id) REFERENCES t\_vehicule\_vhc (vhc\_id));

CREATE TABLE t\_camion\_cmn (cmn\_tonnage float, vhc\_id int primary key, FOREIGN KEY (vhc\_id) REFERENCES t\_vehicule\_vhc (vhc\_id));

DROP TRIGGER IF EXISTS verifImmatriculation;

DELIMITER //

CREATE TRIGGER verifImmatriculation

BEFORE INSERT ON t\_voiture\_vtr

FOR EACH ROW

BEGIN

if new.VHC\_ID not in (select VHC\_ID from T\_VEHICULE\_VHC)

THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT= "l’immatriculation n’existe pas";

end if;

END //

DELIMITER ;

**Correction :**

DROP TRIGGER IF EXISTS verifImmatriculation;

DELIMITER //

CREATE TRIGGER verifImmatriculation

BEFORE INSERT ON t\_voiture\_vtr

FOR EACH ROW

BEGIN

if NEW.vhc\_id not in (select vhc\_id from t\_vehicule\_vhc )

THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT= "le numéro d'immatriculation n'existe pas dans la classe t\_vehicule\_vhc";

end if;

if NEW.vhc\_id in (select vhc\_id from t\_voiture\_vtr union select vhc\_id from t\_moto\_mto)

THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT= "le numéro d'immatriculation existe dans table voiture ou moto";

end if;

END //

DELIMITER ;