

**INTRODUCTION AU LANGAGE SQL**

**LDD, LMD**

**Assuré par : Mr BERAIS Achour (Enseignant Développeur DBA )**

**Préparé par :** **DR. TOUAZI Fayçal (Maitre de conférences-Université Boumerdes)**

**Résumer :**

«Cette partie a pour but d’apprendre les principales commandes SQL avec le SGBD MySQL telles que : SELECT, FROM, WHERE, INSERT INTO, UPDATE, DELETE, etc. Chaque commande SQL sera présentée dans un chapitre dédié avec des exemples clairs, simples et concis.

»

**CHPITRE I : Introduction au langage SQL**

# Notions de base

## BASE DE DONNÉES

Une **base de données** est un **ensemble d'informations** (données) qui ont été stockées sur un support informatique de manière **organisée et structurée** afin de pouvoir facilement consulter et modifier leur contenu.

On peut prendre l'exemple d'une entreprise lambda avec un outil de facturation. Une base de données est fort utile pour stocker les clients et les informations liées à ces derniers comme les coordonnées et les contacts. Cette même base devra pouvoir stocker des devis, des factures, etc. en lien avec ces clients. Cet ensemble de données constitue une base de données. On peut faire le parallèle avec une boutique e-commerce. La base de données d'un tel site doit pouvoir stocker toutes les informations du catalogue produits ainsi que celles liées aux clients et aux commandes.

Avoir une base de données, c'est bien, pouvoir interagir avec, c'est beaucoup mieux. Dans l'absolu, on peut créer une base de données avec un ensemble de fichiers textes et même dans un tableau. Seulement, ces solutions ont des limites. En effet, il devient rapidement très complexe de faire des sélections précises de données ou de retrouver rapidement une information. C'est pour ces raisons que des Systèmes de Gestion de Bases de Données (SGBD), comme **MySQL**, ont été inventées.

## SQL

Pour langage de requêtes structuré est un langage déclaratif destiné à la manipulation de bases de données au sein des SGBD.

SQL est un langage déclaratif, il n'est donc pas a proprement parlé un langage de programmation, mais plutôt une interface standard pour accéder aux bases de données.

Il est composé de quatre sous-ensembles :

* **Le Langage de Définition de Données** (**LDD**, ou en anglais DDL, *Data Definition Language*) pour créer et supprimer des objets dans la base de données (tables, contraintes d'intégrité, vues, etc.).

Exemple de commandes : **CREATE** **DROP** **ALTER**

* **Le Langage de Manipulation de Données** (**LMD**, ou en anglais DML, *Data Manipulation Language*) pour la recherche, l'insertion, la mise à jour et la suppression de données. Le LMD est basé sur les opérateurs relationnels, auxquels sont ajoutés des fonctions de calcul d'agrégats et des instructions pour réaliser les opérations d'insertion, mise à jour et suppression.  
    
  Exemple de commandes : **INSERT** **UPDATE** **DELETE** **SELECT**
* **Le Langage de Contrôle de Données** (**LCD**, ou en anglais DCL, *Data Control Language*) pour gérer les droits sur les objets de la base (création des utilisateurs et affectation de leurs droits).

Exemple de commandes : **GRANT** **REVOKE**

* **Le Langage de Contrôle de Transaction (LCT**, ou en anglais TCL, *Transaction Control Language*) pour la gestion des transactions (validation ou annulation de modifications de données dans la BD)

Exemple de commandes : **COMMIT**  **ROLLBACK**

## SYSTÈME DE GESTION DE BASE DE DONNÉES (SGBD)

Un SGBD est un **logiciel permettant d'intéragir avec les informations d'une base de données**. On entend par intéragir, sélectionner, ajouter, modifier et supprimer des données de la base. On regroupe généralement ces opérations sous l'acronyme CRUD pour (Create, Read, Update and Delete). MySQL est bien évidemment un SGBD.

Nous venons de le voir, une base de données seule ne suffit pas. Nous avons donc préconisé d'héberger notre base de données au sein d'un SGBD. Pour utiliser MySQL, il est nécessaire d'utiliser un nouveau langage, à savoir le SQL.

On peut donc résumer :

* Les Systèmes de Gestion de Base de Données (SGBD) permettent de gérer les bases de données.
* MySQL est un SGBD.
* Le langage SQL est utilisé pour dialoguer avec MySQL.

Les fonctionnalités classiques d'un SGBD Relationnel :

1. Définition et manipulation de données (LDD et LMD ) :
2. La gestion de la cohérence de données
3. Des outils de gestion de BD repartis:
   * Accès à la BD sur des sites distants,
   * Possibilité d'avoir des données reparties sur plusieurs sites.
4. Outils d’administration, supervision et de cohérence de la BD
5. Gestion de la confidentialité
6. Possibilité d'avoir des vues (partie d'un BD qu'on attribue à un utilisateur)
7. Possibilité de donner des privilèges à un utilisateur ou à une classe d'utilisateurs
8. Gestion des contraintes d'intégrité: Il existe plusieurs types de contraintes:
9. Contraintes De Type Implicite
10. Contraintes De Présence :«Null» Ou «NotNull»
11. Contraintes De Référence
12. Des Outils De Développement D'application :
    * Générateur De Tableur,
    * Gestionnaire De Grille D'écran,
    * Générateur De Rapport ,Etc.…

## ORGANISATION D'UNE BASE DE DONNÉES

Une base de données MySQL est organisée avec différentes **tables**. Une base de données contient une ou plusieurs tables, dont les noms doivent être uniques au sein de la base de la données. Une table contient des **colonnes**. Les colonnes contiennent les données.

| Table : clients | | | | |
| --- | --- | --- | --- | --- |
| **id** | **Prénom** | **Nom** | **Email** | **ville** |
| 1 | Marine | Leroy | [mleroy@example.com](mailto:mleroy@example.com) | Paris |
| 2 | Jean | René | [jrene@example.com](mailto:jrene@example.com) | Lyon |
| 3 | Ted | Bundy | [tbundy@example.com](mailto:tbundy@example.com) | Miami |

Ci-dessus, la table clients représentée ici sous la forme d'un tableau, contient les **colonnes** suivantes :

* id
* prénom
* nom
* email
* ville

Les données insérées dans les tables sont représentées sous la forme de ligne (**tuple**). Dans notre exemple, la table clients contient 3 tuples.

# Le LDD (Langage de Définition de Données)

Il existe 3 commandes :

* + - CREATE
    - ALTER
    - DROP

## Création d’une base de données

CREATE DATABASEnombase

## Commande de création de tables « CREATE »

**Première forme de creation de table**

**Syntaxe:**

CREATE TABLE [IF NOT EXISTS] [nomBase.]nomTable

( colonne1 type1[NOT NULL | NULL] [DEFAULT valeur1]

[, colonne2 type2 [NOT NULL | NULL] [DEFAULT valeur2]]

[CONSTRAINT nomContrainte1 typeContrainte1] ...)

* IF NOT EXISTS : permet d’éviter qu’une erreur se produise si la table existe déjà (si c’est le cas, elle n’est aucunement affectée par la tentative de création).
* *nomBase* : (jusqu’à 64 caractères permis dans un nom de répertoire ou de fichier sauf« / », « \ » et « . ») s’il est omis, il sera assimilé à la base connectée.
* *nomTable* : mêmes limitations que pour le nom de la base.
* *colonnei typei* : nom d’une colonne (mêmes caractéristiques que pour les noms des tables) et son type (INTEGER, CHAR, DATE…). Nous verrons quels types sont disponibles sous MySQL. La directive DEFAULT fixe une valeur par défaut. La directive NOT NULL interdit que la valeur de la colonne soit nulle.
* *Nom Contrainte i type Contrainte i* : nom de la contrainte et son type (clé primaire,  
  clé étrangère, etc.) comme suit :

CONSTRAINT nomContrainte

UNIQUE (colonne1 [,colonne2]...)

PRIMARY KEY (colonne1 [,colonne2]...)

FOREIGN KEY (colonne1 [,colonne2]...)REFERENCES nomTablePere [(colonne1 [,colonne2]...)]

[ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}]

[ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]

CHECK (condition)

**Exemple :**

CREATETABLE Test(colonne DECIMAL(38,8));

CREATE TABLE bdsoutou.Compagnie

(comp CHAR(4),

nrue INTEGER(3),

rue CHAR(20),

ville CHAR(15) DEFAULT 'Paris',

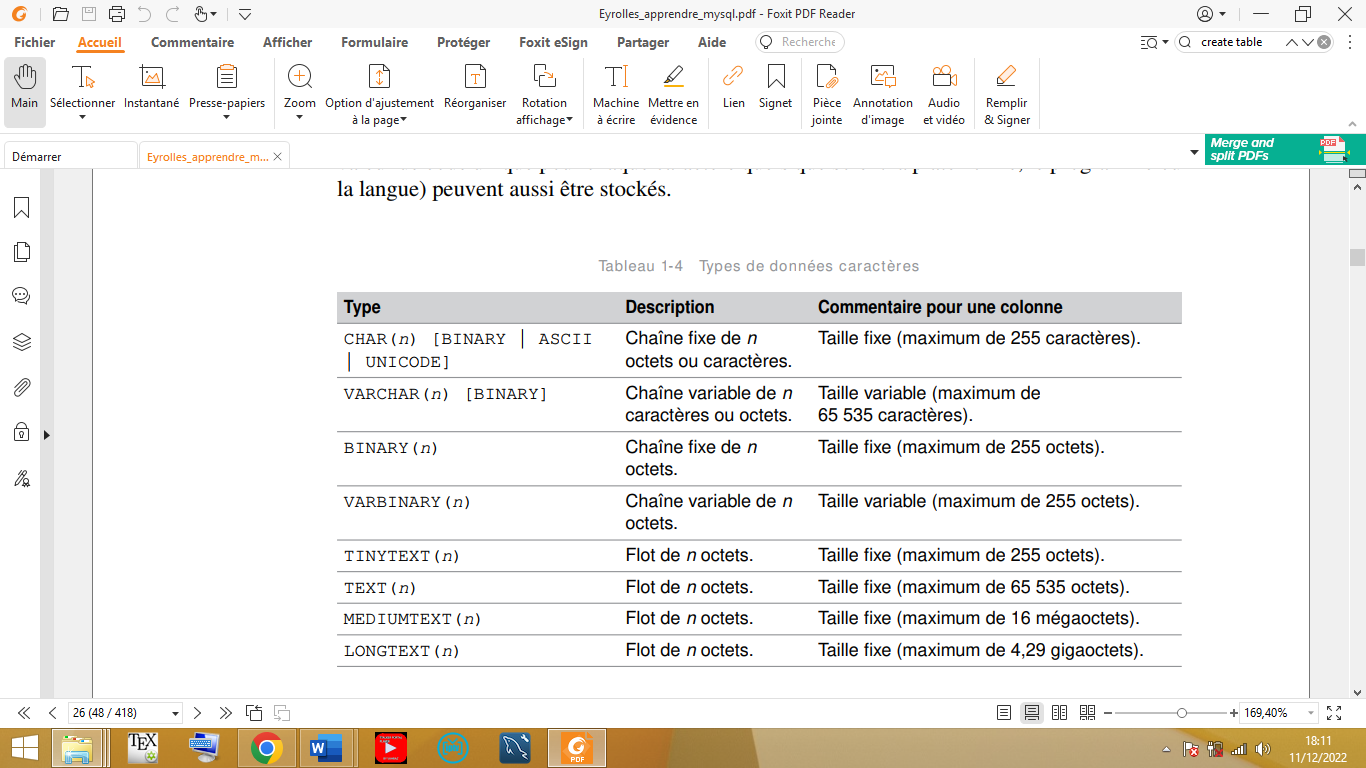
nomComp CHAR(15) NOT NULL

);

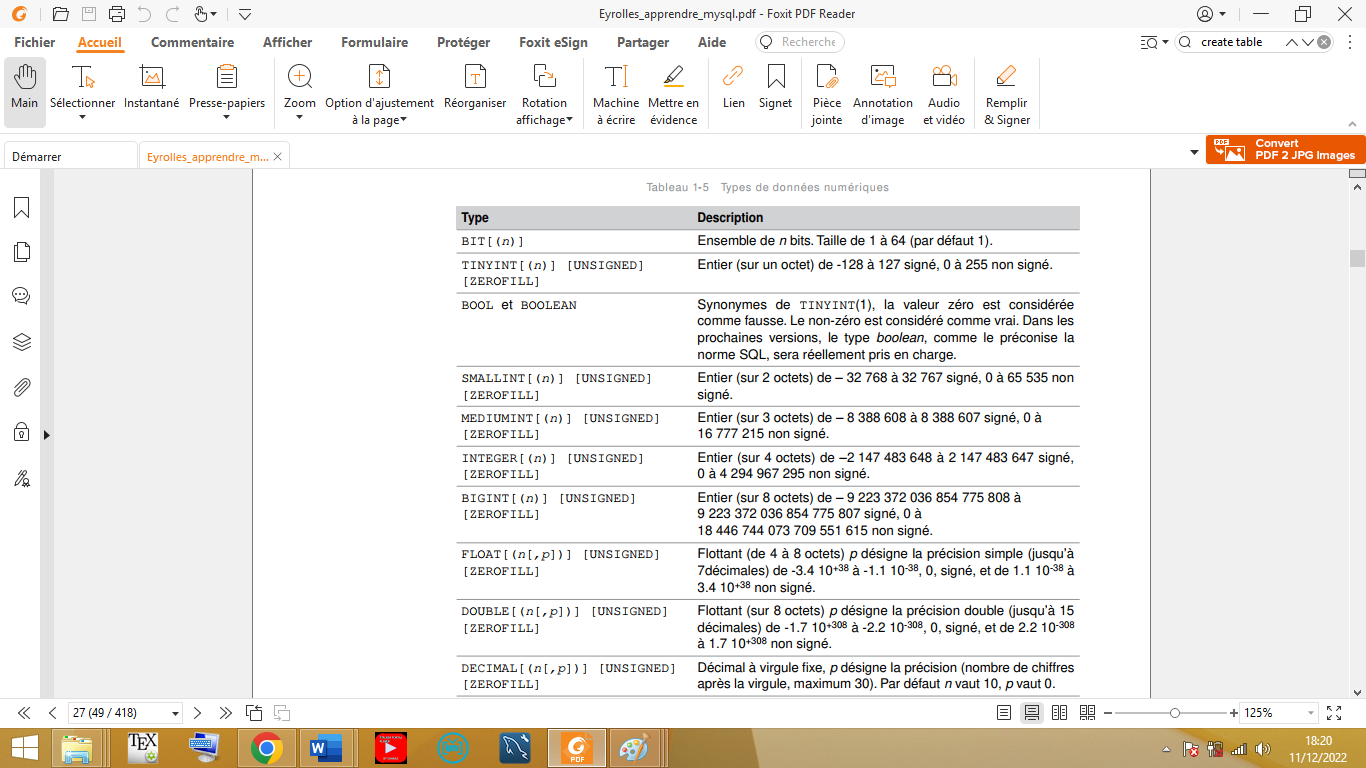
#### Rappel sur les types de données:

Les types les plus courants sont les suivants:

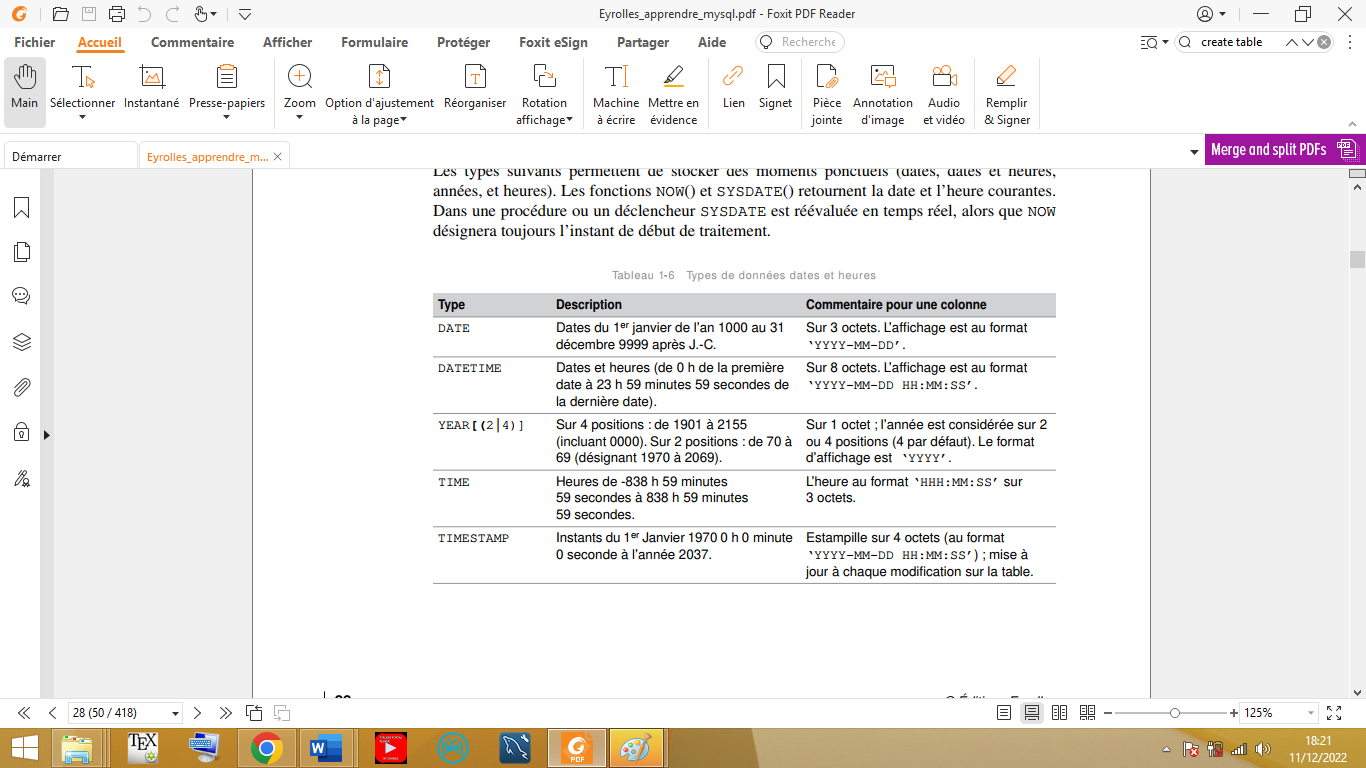
* + - * 1. **CHAINE DE CARACTERES**

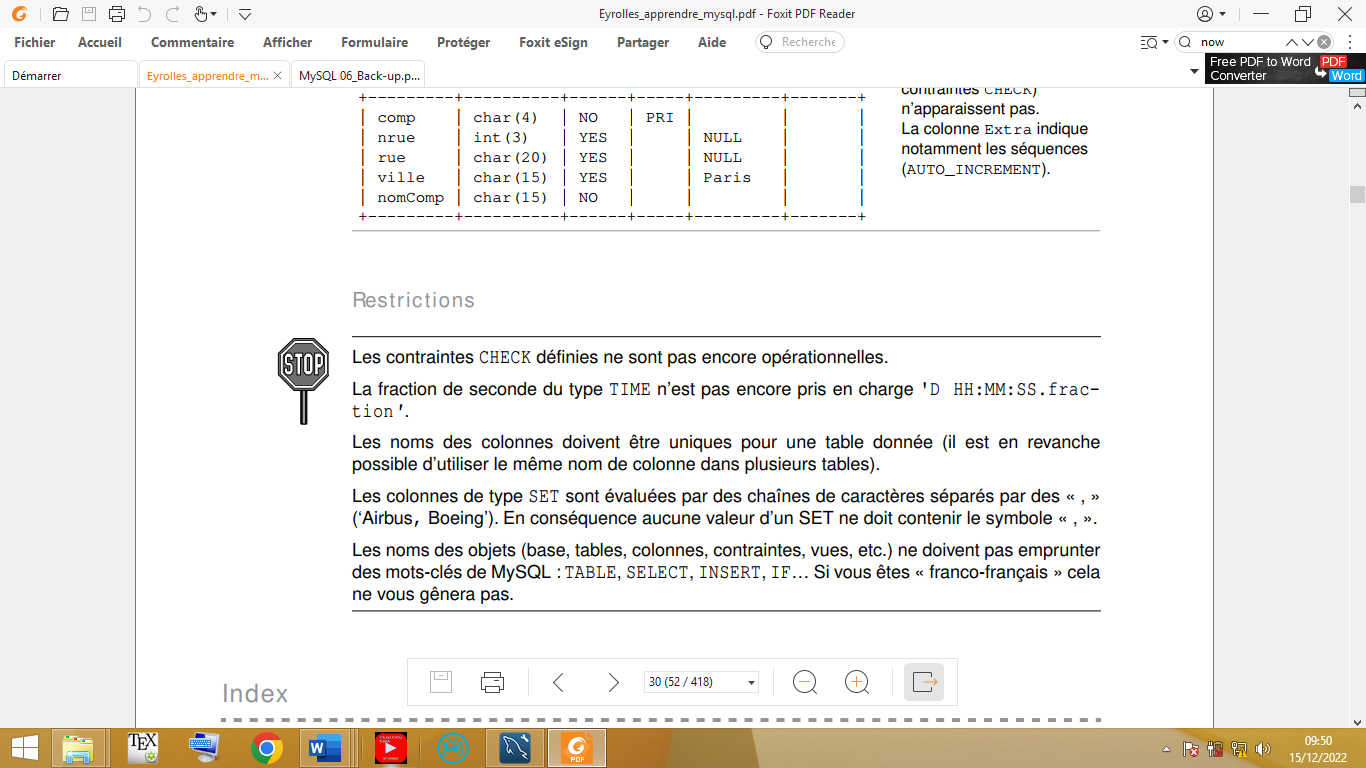


* + - * 1. **NOMBRE**



* + - * 1. **DATE**





Les types suivants permettent de stocker des moments ponctuels (dates, dates et heures,  
années, et heures). Les fonctions NOW() et SYSDATE() retournent la date et l’heure courantes.  
Dans une procédure ou un déclencheur SYSDATE est réévaluée en temps réel, alors que NOW désignera toujours l’instant de début de traitement.

**Exemple2**:de création de table

Soit la table **Id entité** simplifiée qui ne contient que le numéro d’identification, le nom ,le prénom ,la date de naissance, le nombre d’enfants ,le revenu et l'adresse d'un individu. De plus :

1. Le numéro d'identité est un nombre entier
2. Le nom et le prénom d'une personne ne doit pas dépasser 30 caractères .Le type de ces colonnes sera donc un char(30)ou un varchar(30)
3. L'adresse peut être longue
4. Le nombre d'enfants ne dépasse pas 2 chiffres, INT(2)
5. Enfin pour le revenu nous ne nous intéresserons pas aux centimes.

La commande pour la création de cette table est la suivante :

CREATE TABLE identite (

ident INT,

nom varchar(30),

prenom varchar(30),

date\_naissance date,

nbr\_enfants INT

revenu DECIMAL(12,0),

adresse varchar(200),

ville varchar(50),

codepostal DECIMAL(5),

) ;

**Deuxième forme de création detable**

Elle consiste à créer une table a l'aide d'une autre table (ou de plusieurs tables)

#### Syntaxe:

CREATE TABLE <Nom\_table>[(Liste de noms de colonnes avec éventuellement Null ou Not Null)]

AS < REQUÊTE > ;

**Rq**: une table peut donc être créer et alimenter à partir d'une requête.

**Exemple:**

CREATE TABLE Bons\_Clients

AS ( SELECT nom, adresse, val\_compte

FROM CLIENTS

WHERE val\_compte > 0) ;

## Commande de modification de tables (ALTER)

### Modification de la structure de tables

Elle consiste en l’ajout de colonnes.

**Syntaxe:**

ALTER TABLE <Nom\_table> ADD (définition\_d’une\_colonne) ;

**Exemple:**

ALTER TABLE Bons\_Clients ADD telChar(8));

### Modification du type des colonnes existantes

**Syntaxe:**

ALTER TABLE <nom table> Modify (définition des colonnes) ;

Dans la définition des colonnes, on peut redéfinir le type d’une colonne existante.

**Exemple:**

ALTER TABLE Bons\_Clients Modify tel char(10);

### Renommer une colonne colonnes existantes

**Syntaxe:**

ALTER TABLE <nom table> RENAME COLUMN <ancient nom>TO<nouveau nom>;

**Exemple :**

ALTER TABLE Bons\_Clients RENAME COLUMN tel TO Num\_Tel;

### Commande de suppression de tables (DROP):

**Syntaxe:**

DROP TABLE < Nom\_table> ;

**Exemple:**

DROP TABLE Bons\_clients ;

## Décrire la structure d’une table

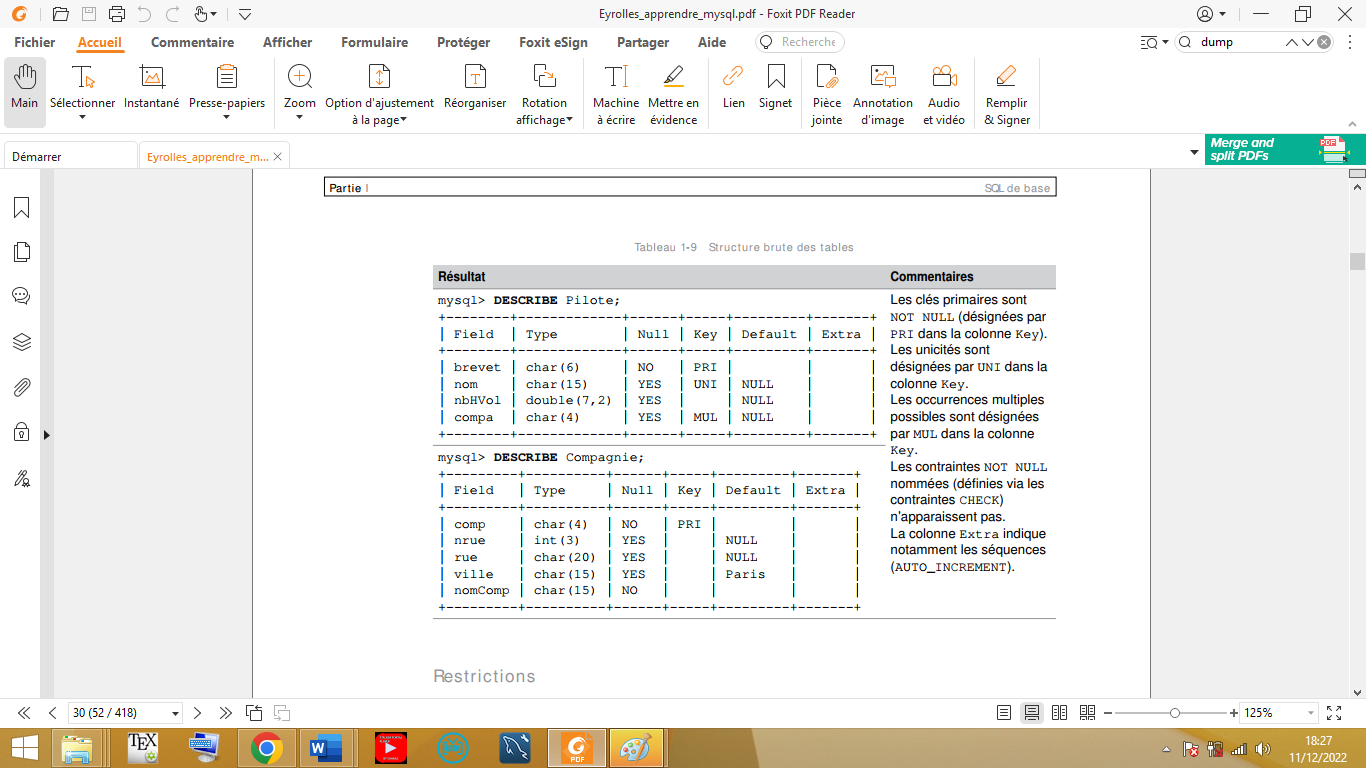
DESCRIBE (écriture autorisée DESC) est une commande qui vient de SQL\*Plus d’Oracle et  
qui a été reprise par MySQL. Elle permet d’extraire la structure brute d’une table ou d’une  
vue.

DESCRIBE[*nomBase*.] *nomTableouVue* [*colonne*];

**Exemple :**

DESCRIBEPilote;

DESCRIBECompagnie;



## Vider une table de ses tuples

La commande TRUNCATE permet de vider une table de tous ses tuples.

**Syntaxe :**

TRUNCATE<Table>;

**Exemple :**

TRUNCATEPilote;

# Le Langage de Manipulation de Données(LMD)

* SELECT
* INSERT
* UPDATE
* DELETE

## Insertion

**Syntaxe:**

NSERT INTO[nomBase.] { nomTable | nomVue }[(nomColonne,...)]

VALUES ( valeur\_i, valeur\_j, …, valeur\_n ) ;

**Exemple 1 :**

INSERT INTO clients

VALUES ('BEN', 'ALGER', 0 )

## Suppression

Il s’agit de supprimer des lignes dans une tables et non de supprimer la table

**Syntaxe:**

DELETE FROM <nomTable>[WHERE conditions] ;

**Exemple:**

DELETE FROM Pilote WHERE compa ='AF';

DELETE FROM Compagnie WHERE comp ='AF';

## Modification

Modifier le contenu d’une table et non sa structure!

**Syntaxe:**

UPDATE [nomBase.] nomTable

SET col\_name1=expr1 [, col\_name2=expr2 ...]

SET colonne1 = expression1 |(requête\_SELECT) | DEFAULT

[,colonne2 = expression2...]

[WHERE (condition)]

[ORDER BY listeColonnes]

[LIMIT nbreLimite]

* La clause SET actualise une colonne en lui affectant une expression (valeur, valeur par défaut, calcul ou résultat d’une requête).
* La condition du WHERE filtre les lignes à mettre à jour dans la table. Si aucune condition n’est précisée, tous les enregistrements seront actualisés. Si la condition ne filtre aucune ligne, aucune mise à jour ne sera réalisée.
* ORDER BY indique l’ordre de modification des colonnes.
* LIMIT spécifie le nombre maximum d’enregistrements à changer (par ordre de clé primaire croissante).

**Exemple :**

UPDATE Compagnie SET nrue = 50 WHERE comp = 'AN1';

## Consultation (ou Requête Simples)

**Syntaxe:**

SELECT [ { DISTINCT | DISTINCTROW } | ALL ] listeColonnes

FROM nomTable1 [,nomTable2]...

[ WHERE condition ]

[ clauseRegroupement ]

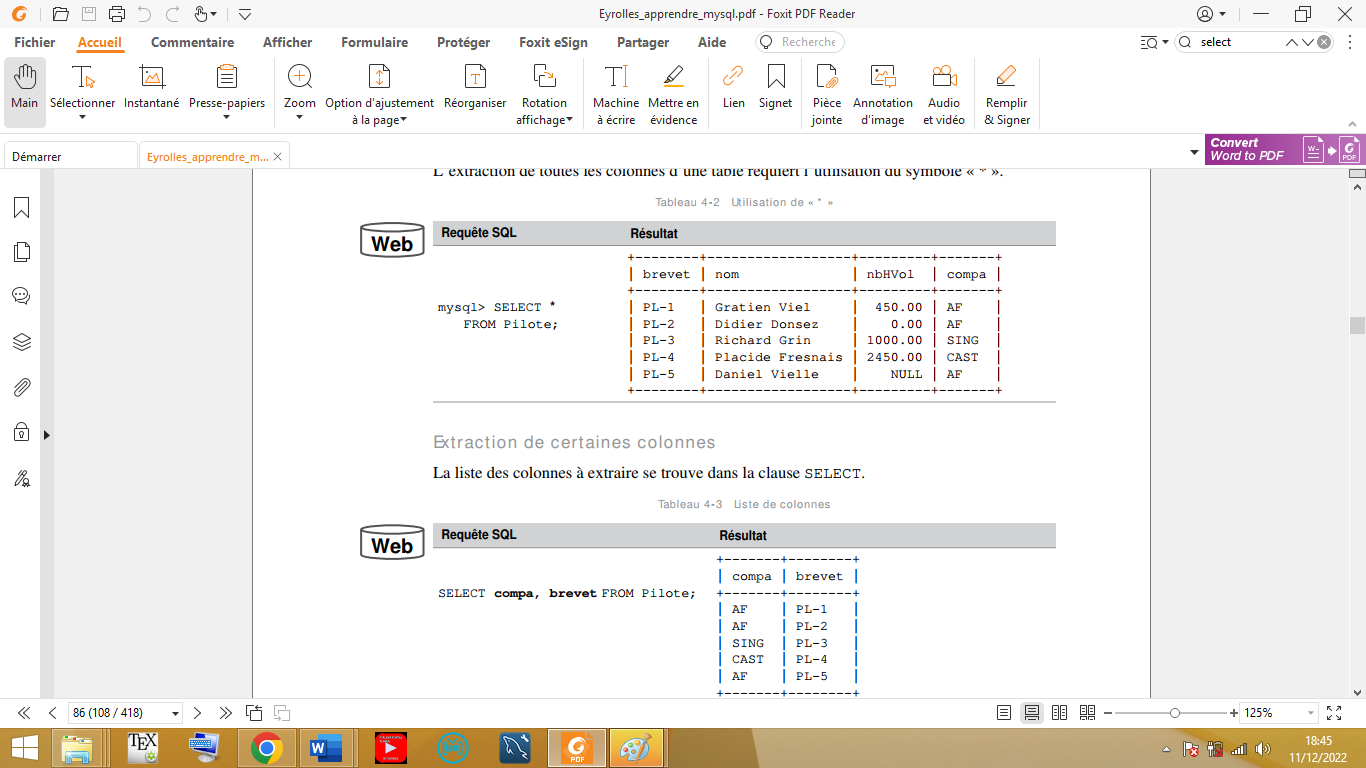
[ HAVING condition ]

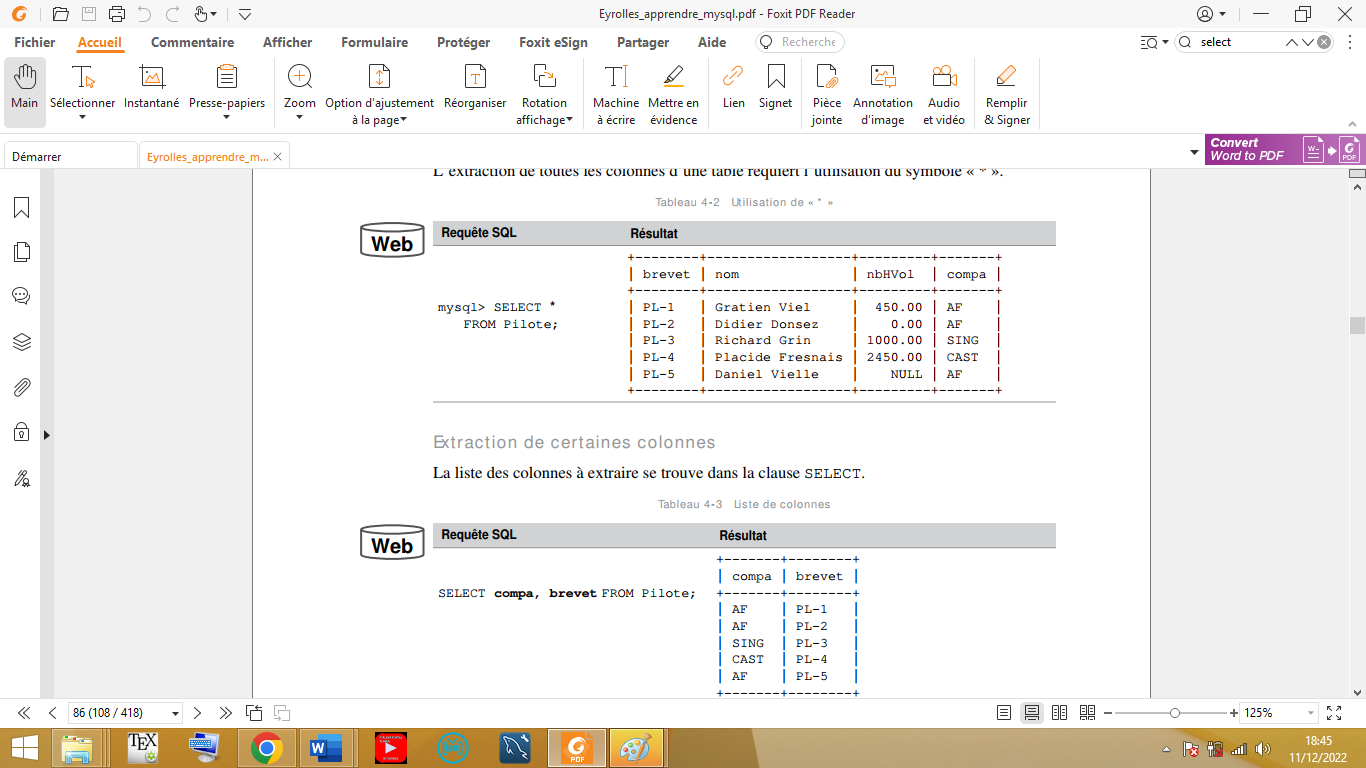
[ clauseOrdonnancement ]

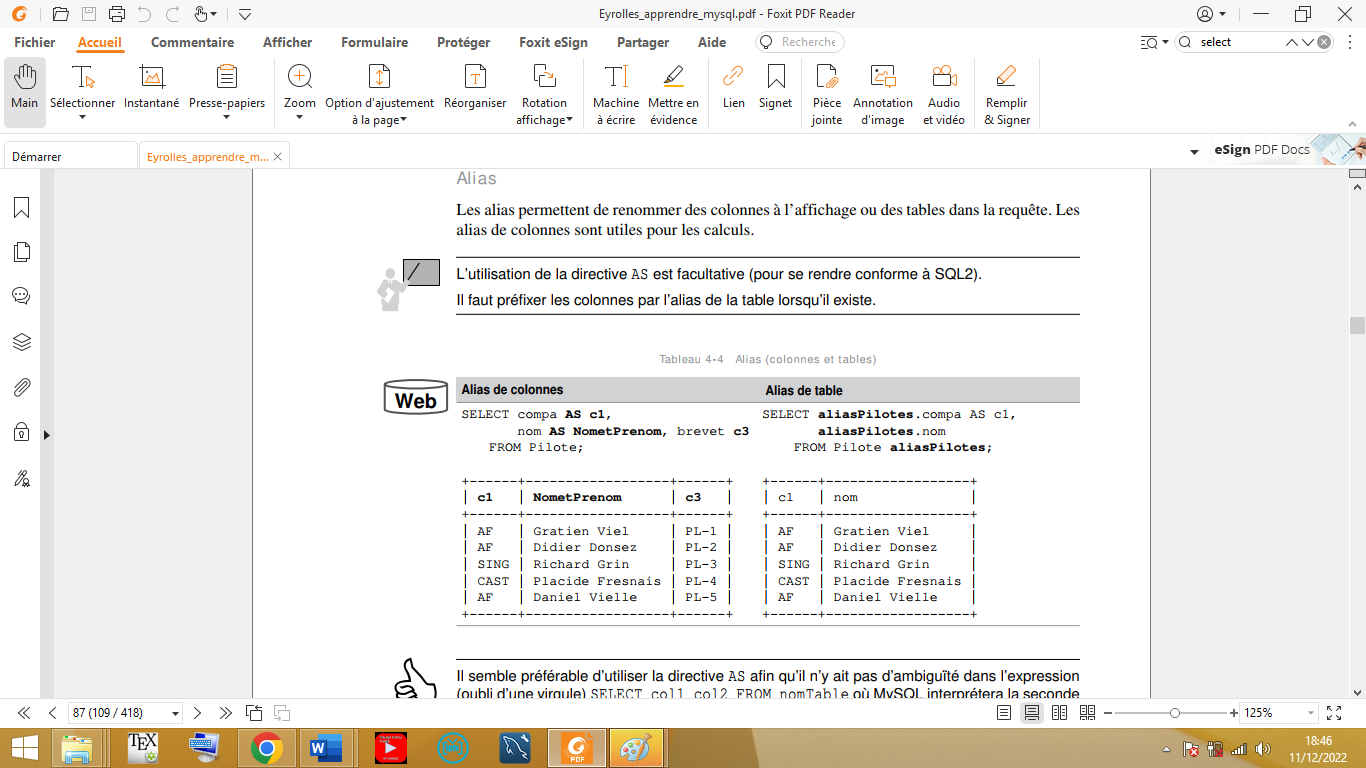
[ LIMIT [rangDépart,] nbLignes ] ;

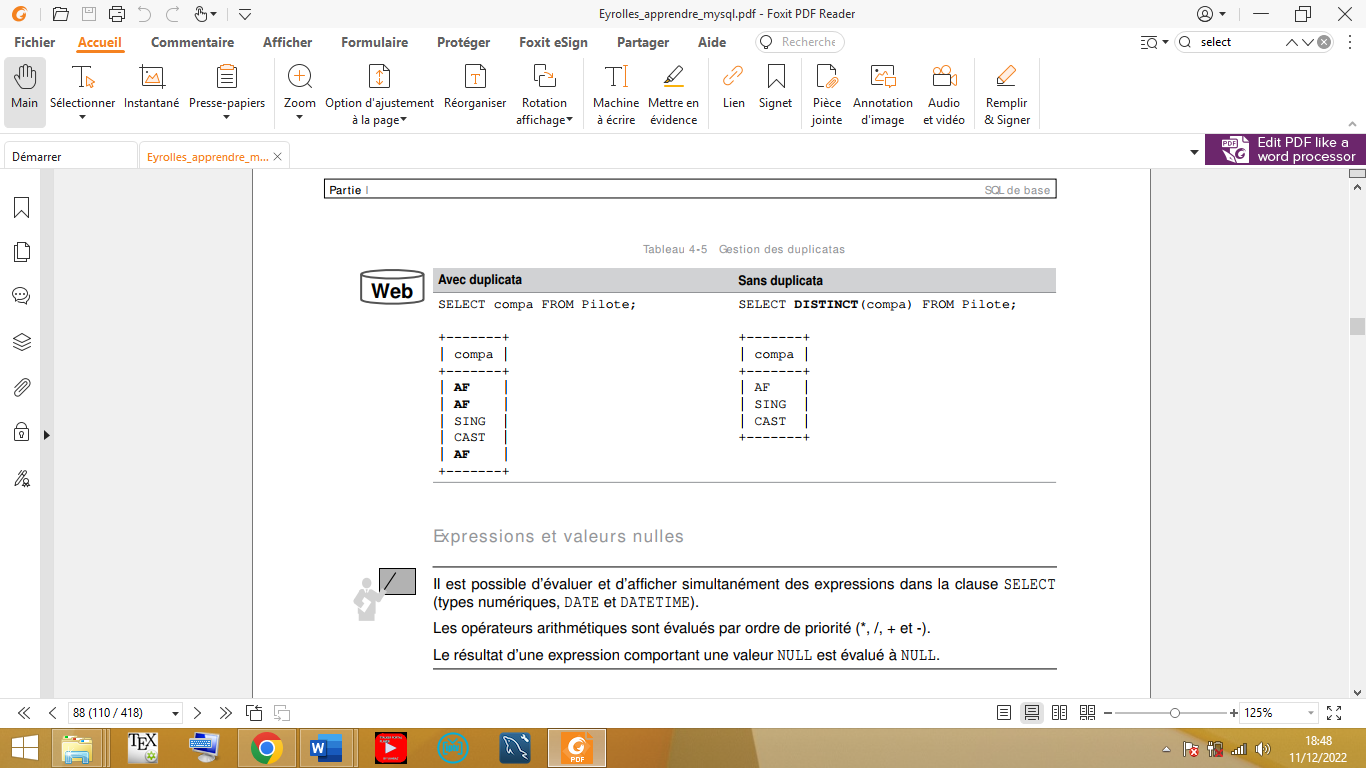
* DISTINCT et DISTINCTROW jouent le même rôle, à savoir ne pas inclure les duplicatas.
* ALL prend en compte les duplicatas (option par défaut).
* *listeColonnes* : { \* | *expression1* [[AS] *alias1* ] [, *expression2*[[AS] *alias2* ]...}
  + *\** : extrait toutes les colonnes de la table.
  + *expression* : nom de colonne, fonction SQL, constante ou calcul.
  + *alias* : renomme l’expression (nom valable pendant la durée de la requête).
* FROM désigne la table (qui porte un alias ou non) à interroger.
* Clause Ordonnancement : tri sur une ou plusieurs colonnes ou expressions.
* LIMIT pour limiter le nombre de lignes après résultat.

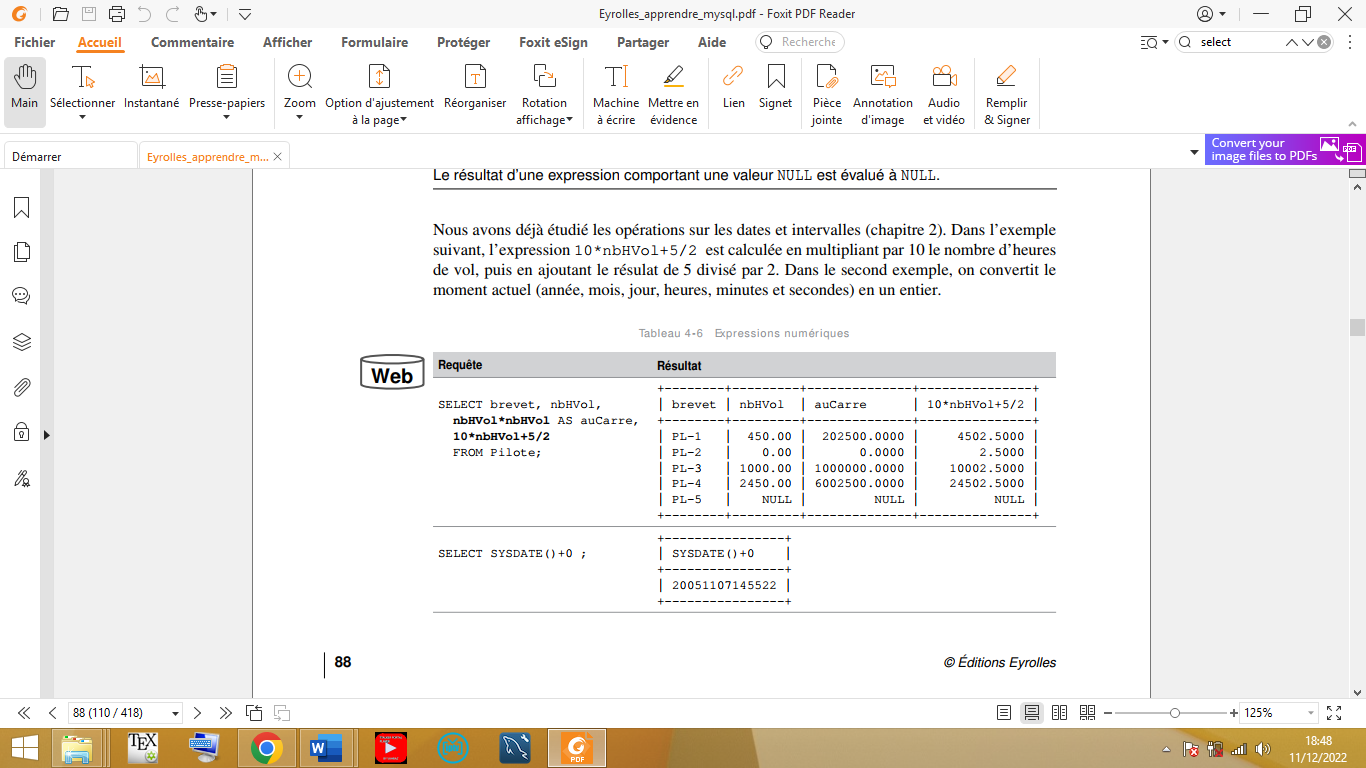
**Exemple :**











Nous avons aussi respectivement les opérateurs:

* + Appartenance à un intervalle:

Colonne[not]betweenexpression1andexpression2

* + Appartenance à une liste de valeurs:

Colonne[not]in(listedevaleurs)

* + De similitude à une chaine de caractère:

Colonne[not]like'chaine'

* + D'égalité ou de différence avec l'élément nul :

Colonneis[not]null

**Opérateurs logiques liant les conditions :**

Il est possible de définir dans les conditions de la clause WHERE plusieurs conditions .Pour lier les conditions entre elles, on utilise les opérateurs logiques OR et AND.

Voici quelques exemples utilisant des conditions de comparaisons:

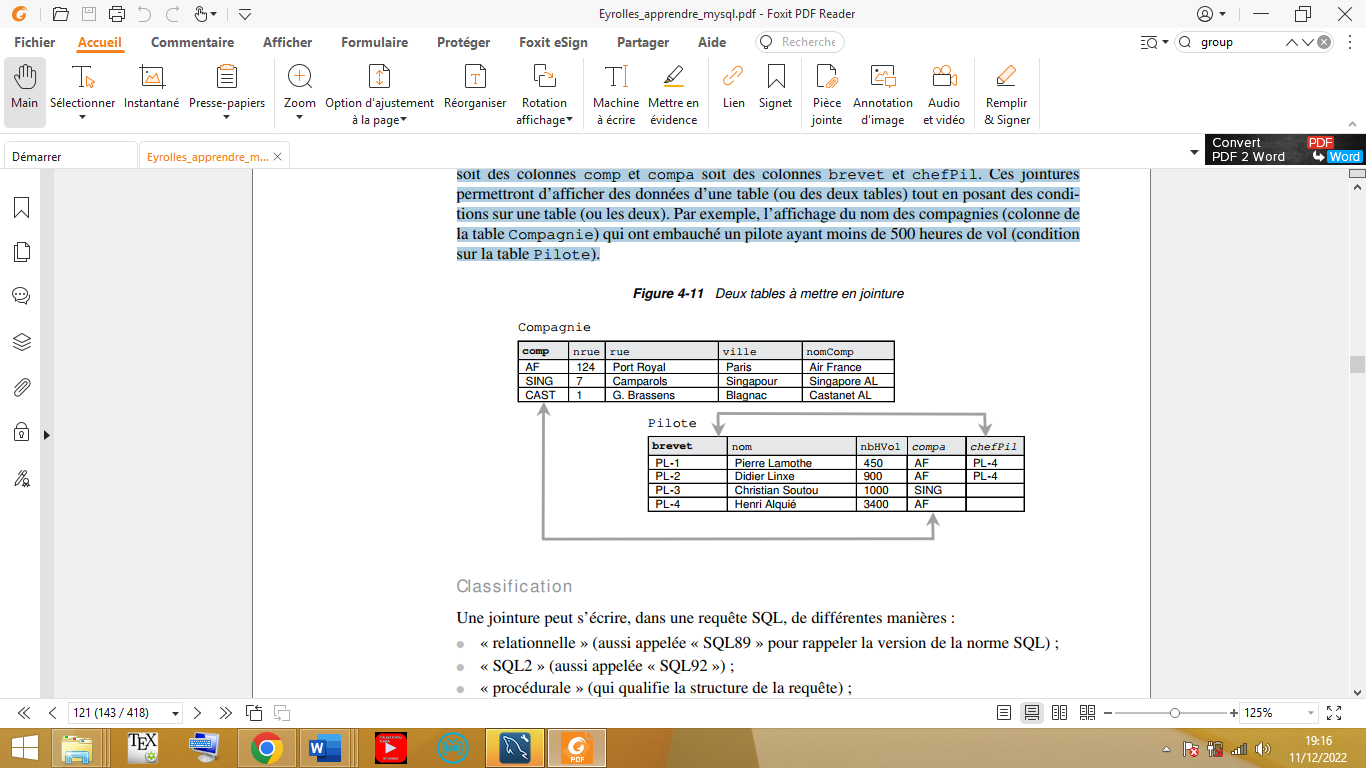
**Exemples:**

* + Lister les noms et adresses des fournisseurs ,ou l'adresse existe et ou le nom commence par 'D' ou bien la 2éme lettre du nom est 'A':
  + Sélectionner les dates de commandes comprise entre le 1er janvier 2022 et aujourd'hui:
  + Sélectionner les noms des clients dont les 2 premières lettres du noms ont 'DU',‘PR’ ou 'MO' et n'importe quelle lettre pour les suivantes:

## Jointures

Les jointures permettent d’extraire des données issues de plusieurs tables. Le processus de  
normalisation du modèle relationnel est basé sur la décomposition et a pour conséquence  
d’augmenter le nombre de tables d’un schéma. Ainsi, la majorité des requêtes utilisent des  
jointures nécessaires pour pouvoir extraire des données de tables distinctes.

En considérant les tables suivantes, les seules jointures logiques doivent se faire sur l’égalité  
soit des colonnes comp et compa soit des colonnes brevet et chefPil. Ces jointures  
permettront d’afficher des données d’une table (ou des deux tables) tout en posant des conditions sur une table (ou les deux). Par exemple, l’affichage du nom des compagnies (colonne de  
la table Compagnie) qui ont embauché un pilote ayant moins de 500 heures de vol (condition  
sur la table Pilote).



**Syntaxe :**

SELECT [ALL | DISTINCT | DISTINCTROW ] listeColonnes

FROM [nomBase.]nomTable1 [{ INNER | { LEFT | RIGHT } [OUTER] }]

JOIN [nomBase.]nomTable2{ ON condition | USING ( colonne1 [, colonne2]... )}

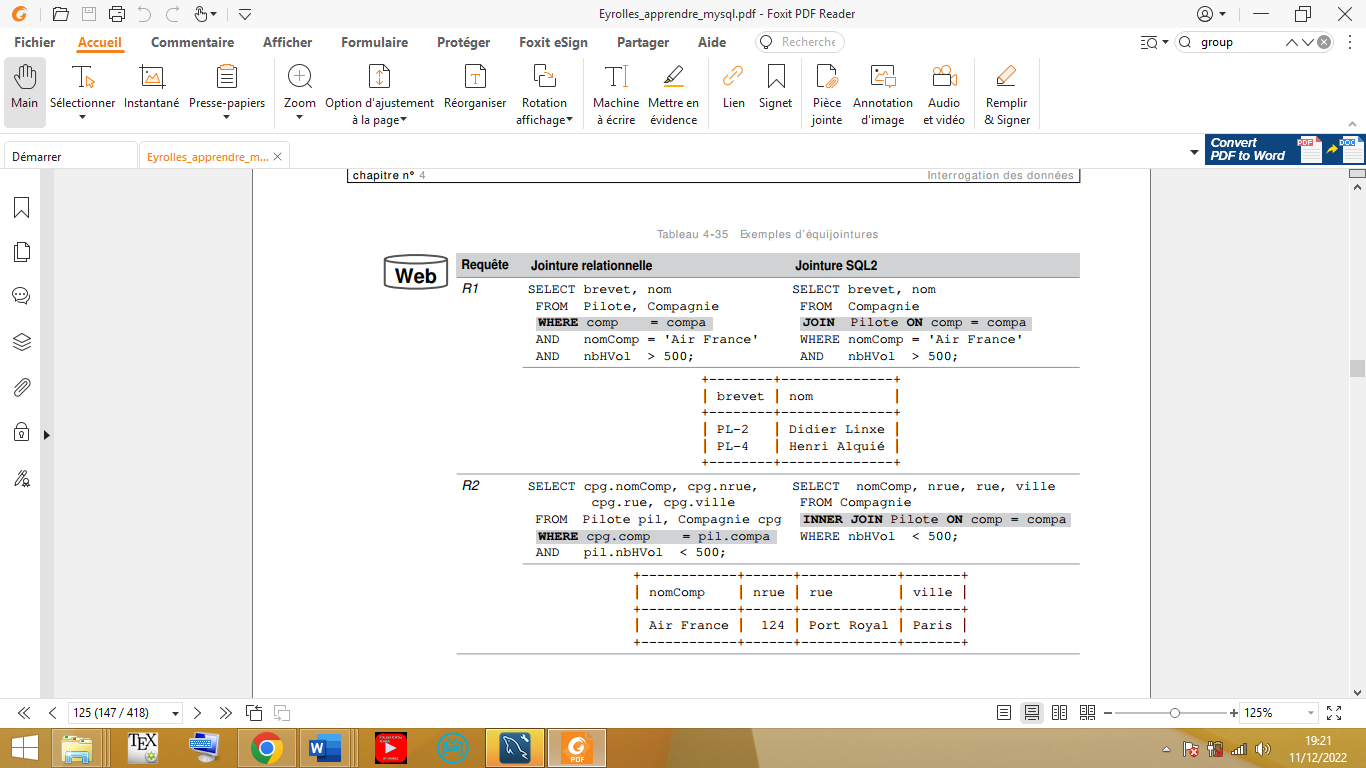
| { CROSS JOIN | NATURAL [{ LEFT | RIGHT } [OUTER] ]

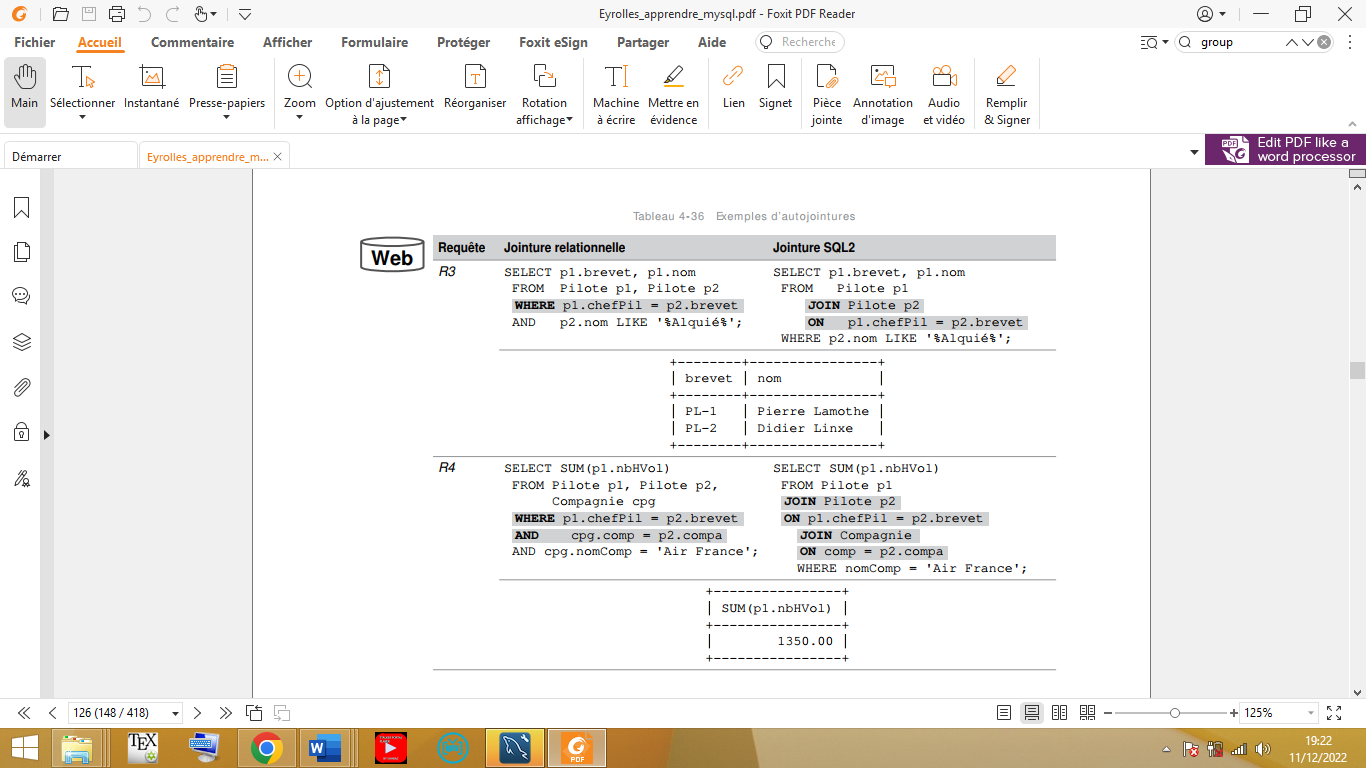
JOIN [nomBase.]nomTable2 } …

[ WHERE condition ];

Bien que, dans le vocabulaire courant, on ne parle que de « jointures » en fonction de la nature  
de l’opérateur utilisé dans la requête, de la clause de jointure et des tables concernées, on  
distingue :

* Les jointures internes (*inner joins*) :
  + L’équijointure (*equi join*) est la plus connue, elle utilise l’opérateur d’égalité dans la clause de jointure. La jointure naturelle est conditionnée en plus par le nom des colonnes. La non équijointure utilise l’opérateur d’inégalité dans la clause de jointure.
  + L’auto jointure (*self join*) est un cas particulier de l’équijointure, qui met en œuvre deuxfois la même table (des alias de tables permettront de distinguer les enregistrements entre eux).
* La jointure externe (*outer join*), la plus compliquée, qui favorise une table (dite  
  « dominante ») par rapport à l’autre (dite « subordonnée »). Les lignes de la table dominante sont retournées même si elles ne satisfont pas aux conditions de jointure.

**Exemple :**  




## Consultation avec sous-requêtes

Lesconditionsavecsous-requêtespermettentdecompareruneexpressionouunecolonneaurésultatd'une autre requête **Select** dite sous-requête.

La sous requête peut faire appel a une autre sous requête , etc.

Une condition de sous-requêtes peut être formulée selon l'une des possibilités suivantes :

* + - * + WHERE expression***opérateur-de-comparaison***

(SELECT...FROM...WHERE...condition…)

Où***opérateur-de-comparaison***est:=,>,<, <>ou!=(pourdifférent),<=,>=

* + - * + WHERE expression***opérateur-de-comparaison*ALL/ANY**

(SELECT...FROM...WHERE...condition…)

* + - * + WHERE expression **[NOT]IN**

(SELECT.....FROM.....WHERE...condition...)

* + - * + WHERE expression **[NOT]EXISTS**

(SELECT.....FROM.....WHERE...condition...)

**Exemple:**

SELECT Nom,Marchandise FROM FOURNISSEUR

WHERE Prix = ( SELECT MAX(Prix) FROM FOURNISSEUR

) ;

## Consultation avec operateurs ensemble liste

Les opérateurs ensemblistess ont:MINUX, INTERSECTetUNION

**Syntaxe:**

(Requête 1){UNION/ (Requête2)

INTERSECT/MINUS}

**Exemple**:Nomde touteslespersonnes delabasededonnées?

**Attention**:lalistedescolonnesde«requête1»et«requête2»doiventêtrelesmêmes(ennombreetentype).

## Regroupements

Cette section traite à la fois des regroupements de lignes (agrégats) et des fonctions de groupe  
(ou multilignes). Nous étudierons les parties surlignées de l’instruction SELECT suivante :

**Syntaxe :**

SELECT [ { DISTINCT | DISTINCTROW } | ALL ] listeColonnes

FROM nomTable

[ WHERE condition ]

[ clauseRegroupement ]

[ HAVING condition]

[clauseOrdonnancement ]

[ LIMIT [rangDépart,] nbLignes ] ;

* *listeColonnes* : peut inclure des expressions (présentes dans la clause de regroupement) ou des fonctions de groupe.
* *clauseRegroupement* : GROUP BY (*expression1*[,*expression2*]...) permet  
  de regrouper des lignes selon la valeur des expressions (colonnes, fonction, constante, calcul).
* HAVING *condition* : pour inclure ou exclure des lignes aux groupes (la condition ne  
  peut faire intervenir que des expressions du GROUP BY).
* *ClauseOrdonnancement* : déjà étudié (ORDER BY dans la section Projection/Ordonnancement).

### Fonctions de groupe

Le tableau suivant présente les principales fonctions. L’option DISTINCT évite les duplicatas  
(pris en compte sinon par défaut). À l’exception de COUNT, toutes les fonctions ignorent les  
valeurs NULL (il faudra utiliser IFNULL pour contrer cet effet).

**Syntaxe :**

SELECT col1[, col2…], fonction1Groupe(…)[,fonction2Groupe(…)…]

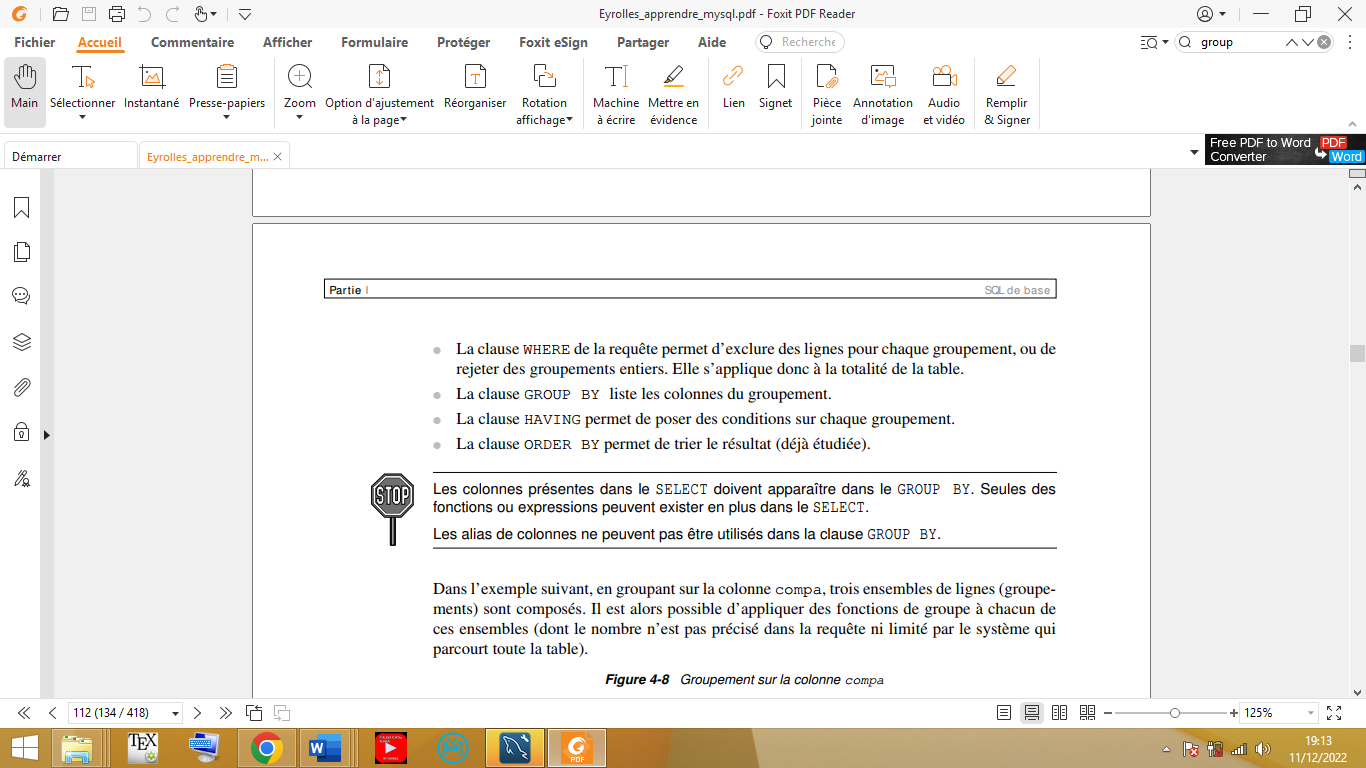
FROM nomTable

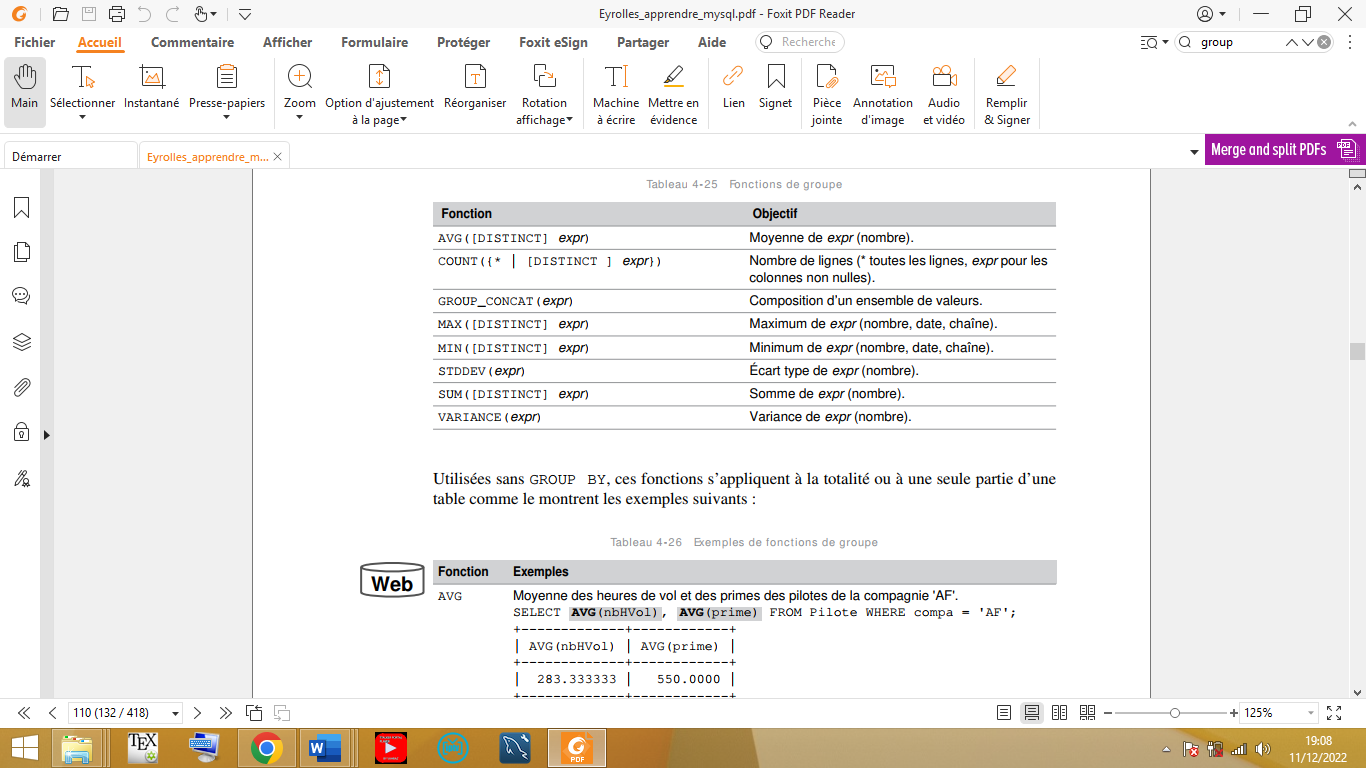
[WHERE condition ]

GROUP BY {*col1* | *expr1* | *position1*} [,{col2... }]  
[ HAVING *condition* ]

[ORDER BY {col1 | expr1 | position1} [ASC | DESC] [,{col1 ... }] ];

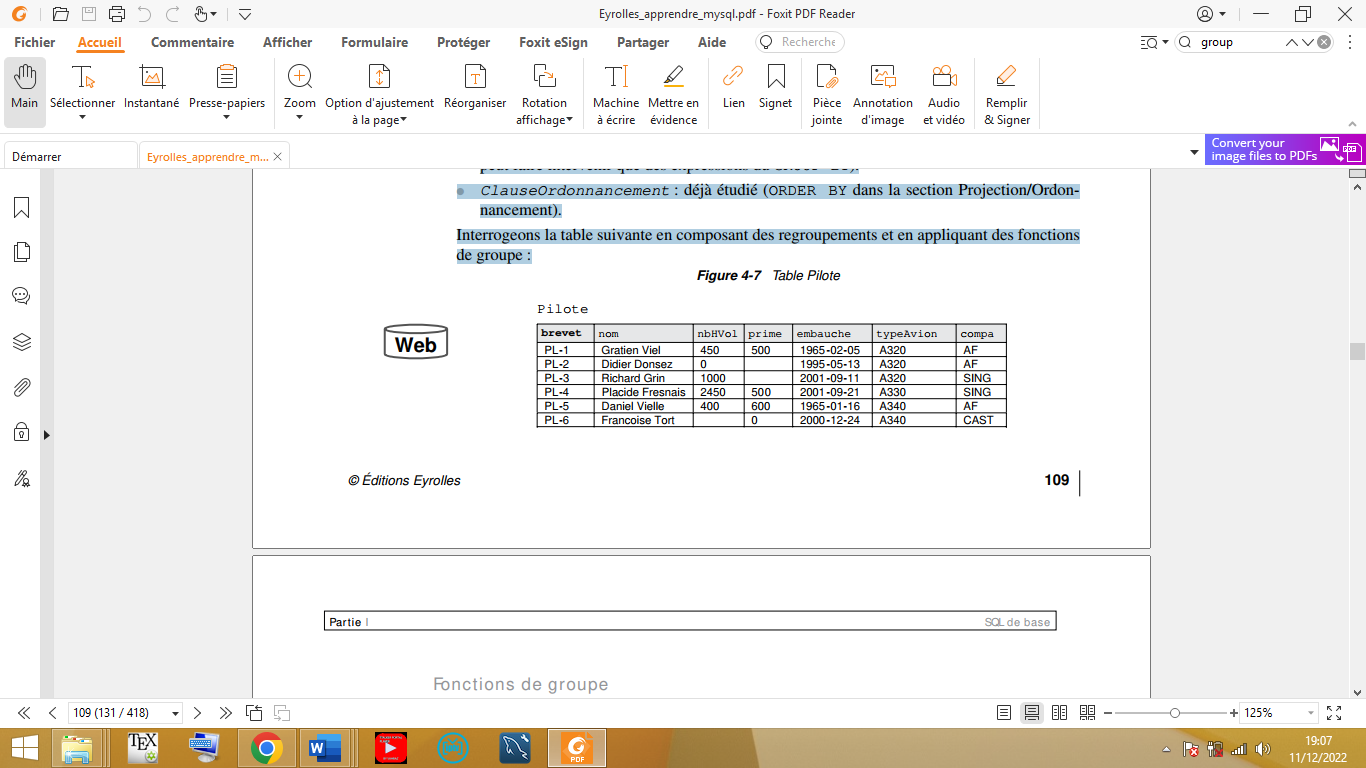
* La clause WHERE de la requête permet d’exclure des lignes pour chaque groupement, ou derejeter des groupements entiers. Elle s’applique donc à la totalité de la table.
* La clause GROUP BY liste les colonnes du groupement.
* La clause HAVING permet de poser des conditions sur chaque groupement.
* La clause ORDER BY permet de trier le résultat (déjà étudiée).

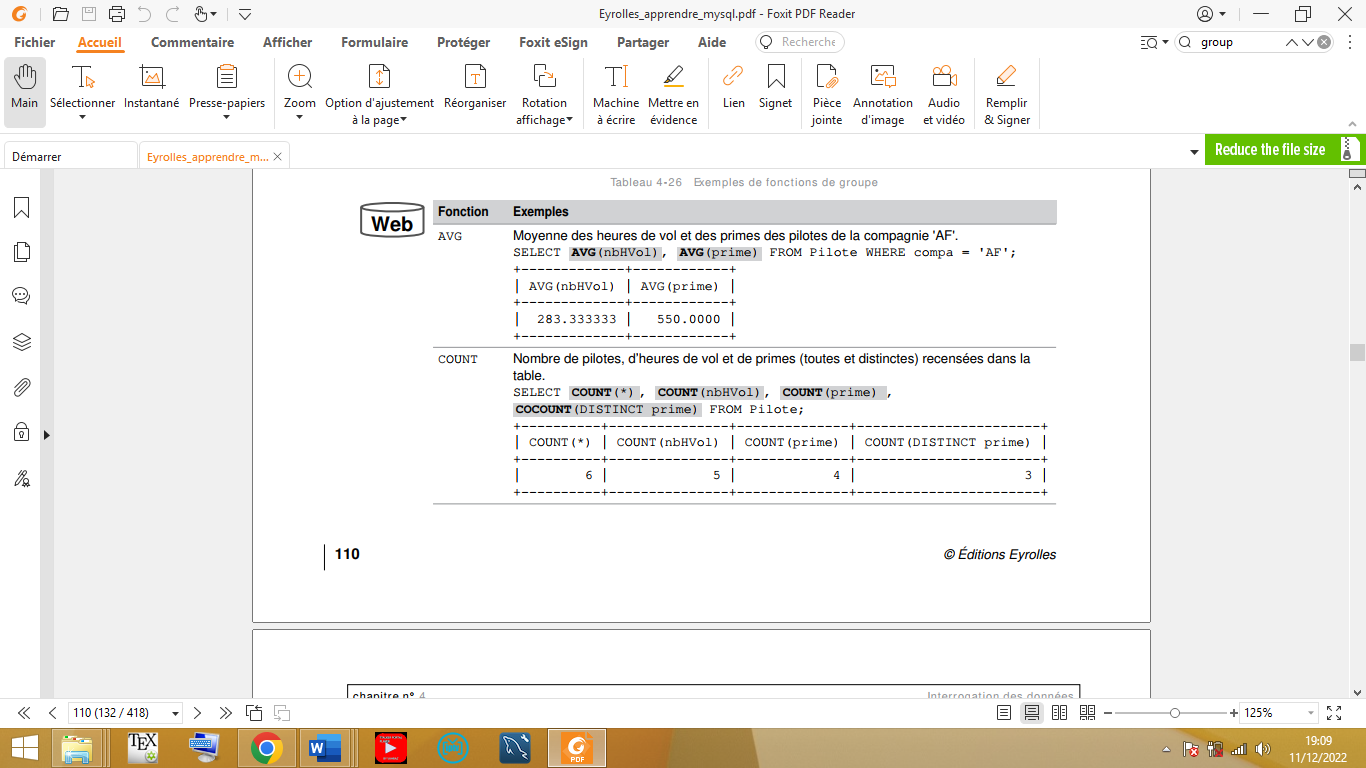


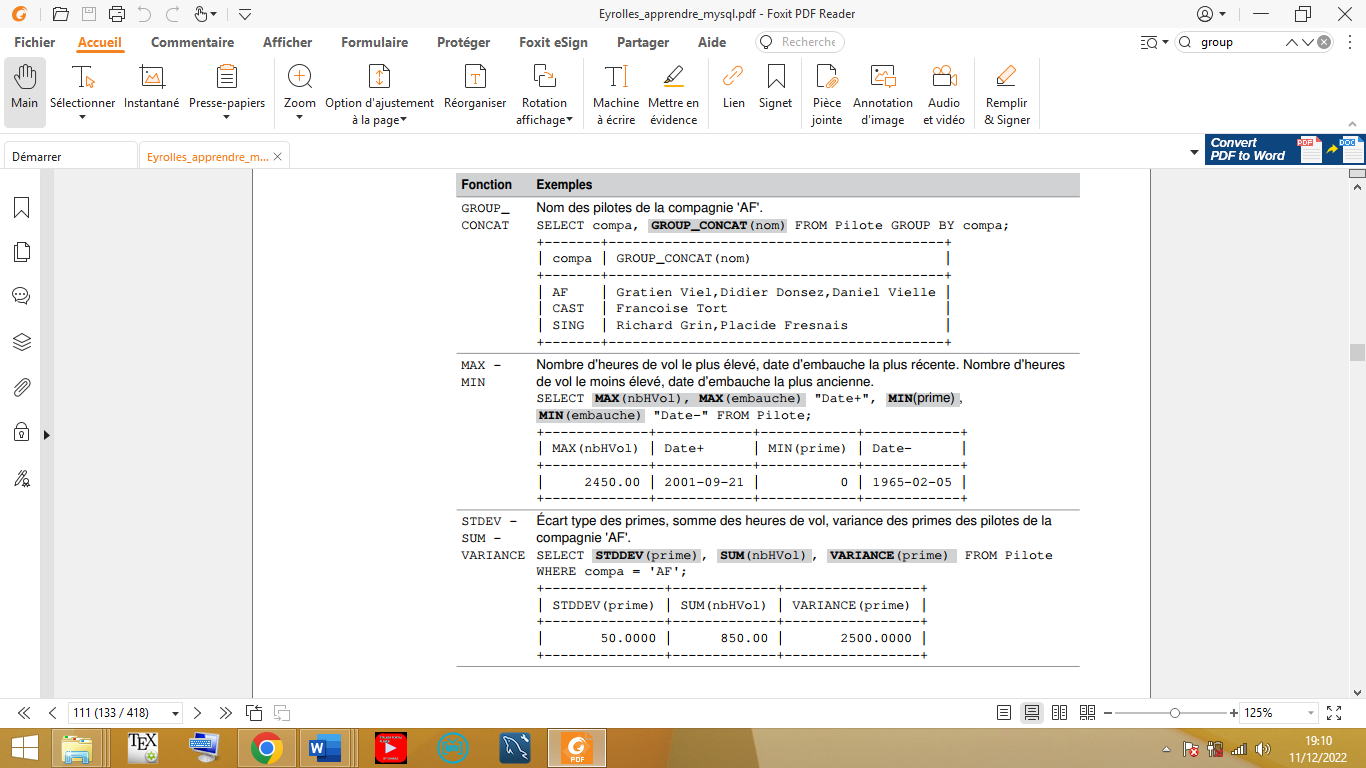


Utilisées sans GROUP BY, ces fonctions s’appliquent à la totalité ou à une seule partie d’une  
table comme le montrent les exemples suivants :

Soit la table Pilote comme suit :







# ANNEXE : LES FONCTIONS PREDEFINIES

Les fonctions prédéfinies deSQLsont:

1. LesFonctionsDeConversionEntreLesDifférentsTypesDeDonnées,
2. LesFonctionsD'agrégat,
3. LesFonctionsNumériques,
4. LesFonctionsDeChaînesDeCaractères,
5. LesFonctionsDeDate,
6. EtDesFonctionsUtilesPourL'environnementD'exploitation.

## LES FONCTIONS DE CONVERSION

La fonction CONVERT() convertit une valeur dans le type de données ou le jeu de caractères spécifié.

**Syntaxe :**

CONVERT(value, type)

* Valeur : La valeur à convertir
* type : Type de données vers lequel effectuer la conversion. Il peut s'agir de l'un des éléments suivants :

|  |  |
| --- | --- |
| Type | Description |
| DATE | Convertit la valeur en DATE. Format: "YYYY-MM-DD" |
| DATETIME | Convertit la valeur en DATETIME. Format: "YYYY-MM-DD HH:MM:SS" |
| DECIMAL | Convertit la valeur en DECIMAL. Utilisez les paramètres optionnels M et D pour spécifier le nombre maximum de chiffres (M) et le nombre de chiffres après la virgule (D). |
| TIME | Convertit la valeur en TIME. Format: "HH:MM:SS" |
| CHAR | Convertit la valeur en CHAR |
| NCHAR | Convertit la valeur en NCHAR (comme CHAR, mais produit une chaîne avec le jeu de caractères national) |
| SIGNED | Convertit la valeur en SIGNED (a signed 64-bit integer) |
| UNSIGNED | Convertit la valeur en UNSIGNED (an unsigned 64-bit integer) |
| BINARY | Convertit la valeur en BINARY (a binary string) |

## LES FONCTIONS NUMERIQUES

Cesfonctionstraitentlesdonnéesdetype INT :

* **abs(n):** donnela valeur absolueden
* **ceil(n):**  retournele pluspetit entiersupérieur ouégal àn
* **cos(n):** retournele cosinusde n,nétant expriméenradian
* **exp(n):** calculela valeurexponentielle den
* **floor(n):** retournelavaleurentièreden
* **Decimal(n):** calculelelogarithmenépérienden
* **mod(n):** retournelereste dela divisionentièredem parn
* **power(n,m):** retournela valeurden alapuissancem
* **round(n,[(m)]):** retournela valeur denarrondiàm positionsàdroitedu pointdécimal
* **sign(n):** retourne-1si nest <0,0si n=0, 1 si n>0
* **tan(n):** retournela tangenteden,nétantexprimé enradian
* **trunc(n):** retourne la valeur n tronquée a m positions décimales. Si m est négatif, latroncaturesefait avant lepointdécimal.

## LES FONCTIONS DE CHAINES DE CARACTERES

* **lower(char):** met enmajuscule machainechar
* **upper(char):** metenminusculeslachainechar
* **substr(char,m,n):** extraitdelachainecharunechainedecaractèresdelongueurnàpartir de m ieme caractère , le m ieme caractère étant compris danslachaineextraite
* **lenght(char):** donnela longueur delachainechar en caractères.

## LES FONCTIONS D'AGREGAT

MySQLoffreentreautreslesfonctionsdebasepourdéterminerdescalculsstatistiques;cesfonctionsnesontutilisables quedans la *liste\_de\_colonnes*d'unerequêteetdans le HAVING dela clauseWHERE.

* **count(\*):** Calculelenombredelignessatisfaisantlarequête,ycomprisleslignesnulls
* **count([DISTINCT/ALL]colonne):**
* Calcule le nombre de valeurs non null de la colonne ''colonne'' satisfaisantla requête ; avec l'option ALL qui est l'option par défaut, on compte touteslesvaleurs qui satisfont la requête
* **sum([DISTINCT/ALL]colonne):**CalculelasommedesélémentsdelacolonnedetypeINTquisatisfont la requête ;lacolonne peutêtre le produit de plusieurscolonnesdetypeINT

Exemple:

selectsum(prixunit)fromarticles ;

## FONCTIONSSTATISTIQUES

* **avg([DISTINCT/ALL]colonne):** fonctionnecommeSUM, mais calculelamoyenne.
* **max({DISTINCT/ALL]colonne):** calculele max des éléments delacolonne
* **min([DISTINCT/ALL]colonne):** calculeleminimum
* **stddev([DISTINCT/ALL]colonne):** calculel'écarttype
* **variance([DISTINCT/ALL]colonne):** calculelavariance

**Travaux pratiques**

**Description de la base de données ‘Concessionnaires deVoitures’:**

|  |  |  |
| --- | --- | --- |
| **FOURNISSEURS:** |  | **FOURNISSEURS-VOITURES-REVENDEURS**  NF NV NR Qté  -- -- -- ---  F1 V1 R1 2  F1 V1 R4 2  F2 V3 R1 4  F2 V3 R2 2  F2 V3 R3 2  F2 V3 R4 5  F2 V3 R5 6  F2 V3 R6 4  F2 V3 R7 8  F2 V5 R2 1  F3 V3 R1 2  F3 V4 R2 5  F4 V6 R3 3  F4 V6 R7 3  F5 V2 R2 2  F5 V2 R4 1  F5 V5 R5 5  F5 V5 R7 1  F5 V6 R2 2  F5 V1 R4 10  F5 V3 R4 12  F5 V4 R4 8  F5 V5 R4 4  F5 V6 R4 5 |
| **NF NomF** | **Code VilleF** |
| -- -------------- | ------- ---------------- |
| F1 BONEY | 20 LONDRES |
| F2 RAMIREZ | 10 PARIS |
| F3 BOUDU | 30 PARIS |
| F4 WILLIAM | 20 LONDRES |
| F5 DIETRICH | 30 COLOGNE |
| **REVENDEURS:** |  |
| **NR NomR** | **VilleR** |
| -- ------------------- | ----------- |
| R1 MENARDEAU | PARIS |
| R2 MACCIONE | ROME |
| R3 SCHULTZ | COLOGNE |
| R4 MUELLER | COLOGNE |
| R5 WADDLE | LONDRES |
| R6 OLAF | OSLO |
| R7 HIGGINS | LONDRES |
| **VOITURES:** |  |
| **NV NomV** | **Couleur Prix** |
| -- ------------------- | ---------------- -------- |
| V1 TESTAROSSA | ROUGE 12 |
| V2 ALPINE | VERT 17 |
| V3 ESPACE | BLEU 17 |
| V4 ESPACE | ROUGE 14 |
| V5 CLIO | BLEU 12 |
| V6 CORVETTE | ROUGE 29 |

**EcrirelesrequêtesSQLpermettantderépondreauxquestionssuivantes:**

* 1. Donneztouslesrenseignementssurtouslesrevendeurs.
  2. Donneztouslesrenseignementssurlesrevendeurshabitant Londres.
  3. Quelssontlesnumérosdesfournisseursfournissant desvoituresaurevendeur R1?
  4. Quelssontlesnumérosdesfournisseursqui fournissentdesvoituresV1aurevendeur R1?
  5. Quelssontlesnoms des revendeursdontl’undesfournisseursest F1?
  6. Quelssontles nomsdesrevendeurspourlesquelsaucunevoituren’est fournie?
  7. Quels sont les numéros des fournisseurs qui fournissent chacun simultanément des voiturespourlesrevendeursR1etR2?
  8. Quels sont les numéros des fournisseurs qui fournissent au moins une voiture rouge aurevendeurR1 ?
  9. Donnezlesnumérosdesvoituresfournies pourtouslesrevendeursdeLondres?
  10. Quelssontlesnumérosdesfournisseursqui fournissentdesvoituresrougespourdesrevendeurssituésà ParisetLondres?
  11. Quels sont les numéros des voitures dont les fournisseurs habitent la ville où est situé lerevendeurauquelces voitures sontdestinées?
  12. Donnez les numéros des revendeurs dont un fournisseur au moins n’habite pas la ville où estsitué ce revendeur.
  13. Quels sont les numéros des revendeurs pour lesquels aucune voiture rouge n’est fournie par unfournisseurhabitantLondres?
  14. Quels sont les numéros des fournisseurs n’habitant pas Paris mais dont le code est égal à celuid’unfournisseurhabitantParis?
  15. DonnezlesnumérosdesvoiturefournieschacuneàtouslesrevendeursdeLondres.
  16. Donnez les numéros des revendeurs qui proposent au moins un ensemble de voitures identiqueàceluifourniparF1.
  17. Donnez les numéros des revendeurs qui ne proposent que des voitures uniquement disponibleschezF1.

**Corrigé :**

**Q1 : Donnez tous les renseignements sur tous les revendeurs**

SELECT \* FROM `revend`

**Q2 : Donnez tous les renseignements sur les revendeurs habitant Londres**

SELECT \* FROM `revend` WHERE `VilleR`="LONDRES"

**Q3 : Quels sont les numéros des fournisseurs fournissant des voitures au revendeur R1**

SELECT NF FROM fvr WHERE NR = "R1"

**Q4 : Quels sont les numéros des fournisseurs qui fournissent des voitures V1 au revendeur R1 ?**

SELECT NF FROM `fvr` WHERE nv="V1" and NR="R1"

**Q5 :Quels sont les noms des revendeurs dont l’un des fournisseurs est F1 ?\***

SELECT NomR FROM fvr,revend WHERE fvr.nr=revend.nr and nf="F1"

**Q6 :Quels sont les noms des revendeurs pour lesquels aucune voiture n’est fournie ?**

SELECT distinct NomR FROM fvr,revend WHERE revend.nr not in (select nr from fvr)

**Q7 :Quels sont les numéros des fournisseurs qui fournissent chacun simultanément des voitures  
pour les revendeurs R1 et R2 ?**

SELECT distinct nf FROM `fvr` WHERE nf in (select nf from fvr where nr="R1") and nf in (select nf from fvr where nr="R2")

**Q8 : Quels sont les numéros des fournisseurs qui fournissent au moins une voiture rouge au  
revendeur R1 ?**

SELECT distinct nf FROM `fvr` WHERE nf in (select nf from fvr, voiture where fvr.nv=voiture.nv and couleur="ROUGE" and fvr.nr="R1")

**Q9 :Donnez les numéros des voitures fournies pour tous les revendeurs de Londres ?**

SELECT nv FROM `fvr`, revend WHERE fvr.nr=revend.nr and VilleR="LONDRES"

**Q10 : Quels sont les numéros des fournisseurs qui fournissent des voitures rouges pour des  
revendeurs situés à Paris et Londres ?**

SELECT distinct nf FROM `fvr`,revend,voiture WHERE fvr.nr=revend.nr and fvr.nv=voiture.nv and couleur="ROUGE" and VilleR="LONDRES" or VilleR="PARIS"

**Q11 :Quels sont les numéros des voitures dont les fournisseurs habitent la ville où est situé le  
revendeur auquel ces voitures sont destinées ?**

SELECT distinct nv FROM `fvr`,revend, fourn WHERE fvr.nf=fourn.nf and fvr.nr=revend.nr and VilleR=VilleF

**Q12 : Donnez les numéros des revendeurs dont un fournisseur au moins n’habite pas la ville où est  
situé ce revendeur.**

SELECT distinct nv FROM `fvr`,revend, fourn WHERE fvr.nf=fourn.nf and fvr.nr=revend.nr and VilleR !=VilleF

**Q13 :Quels sont les numéros des revendeurs pour lesquels aucune voiture rouge n’est fournie par un fournisseur habitant Londres**

SELECT nr FROM revend WHERE nr not in (select fvr.nr from fvr,voiture,fourn where fvr.nv=voiture.nv and fvr.nf=fourn.nf and couleur="ROUGE" and villef="LONDRES")

**Q14 : Quels sont les numéros des fournisseurs n’habitant pas Paris mais dont le code est égal à celuid’un fournisseur habitant Paris ?**

SELECT nf FROM fourn WHERE villef!="PARIS" and code in (select code from fourn where villeF="PARIS")

**Q15 : Donnez les numéros des voiture fournies chacune à tous les revendeurs de Londres**

select nv from voiture where not exists(

select \* from revend where villeR="LONDRES" and nr not in (

select nr from fvr where fvr.nr=revend.nr and fvr.nv = voiture.nv))

**Q16 : Donnez les numéros des revendeurs qui proposent au moins un ensemble de voitures identique à celui fourni par F1.**

select nr from revend where exists(

select \* from voiture where nv in (

select nv from fvr where fvr.nv=voiture.nv and fvr.nr = revend.nr and fvr.nf="F1"))

**Q17 : Donnez les numéros des revendeurs qui ne proposent que des voitures uniquement disponibles chez F1.**

select nr from revend where not exists(

select \* from voiture where nv in (

select nv from fvr where fvr.nv=voiture.nv and fvr.nr = revend.nr and fvr.nf!="F1"))