

Annick Lacroix

MySQL

Notes de Cours AFPA

PC ANNICK

2018-2019

MySQL

1- Table des matières

1- Table des matières	2
2- Aide-Mémoire	5
1- Description	7
2- Les avantages des bases de données :.....	7
1- Se connecter à sa base de données	8
2- Les commandes → CRUD (Create Read Update Delete)	9
A- Les types les plus utilisés en Caractères.....	9
B- Les attributs.....	10
C- Insérer les infos sur une table :	10
D- Insérer des informations dans une table.....	11
Exo.....	13
E- Insertion de valeurs dans une table donnée.....	14
F- Pour importer une base de donnée	16
G- Modifier la structure d'une table	16
Ajouter une colonne :	16
Supprimer une colonne :.....	16
Modifier une colonne	16
Changer le nom d'une colonne :.....	16
H- Trier les informations	17
Pour trier dans l'ordre inverse, il faut mettre :.....	17
Afficher le nombre d'éléments d'une colonne :	18
I- Modifier le contenu d'une colonne.....	18
J- Extraire des données.....	18
Extrait les données comprenant :.....	18
Sélectionner un intervalle de données :	18
Rechercher	19
Exemple :.....	19
Mettre à jour des données :	20
Delete des données :.....	20
Si on veut sélectionner parmi un critère ou un autre.....	21

3- Les Fonctions.....	21
A- Les concaténations éphémères (1)	21
Afficher seulement une partie d'une colonne :.....	22
Afficher la date :.....	23
4- Fonctions d'agrégation SQL	24
A- Liste des fonctions d'agrégation statistiques.....	24
B- Utilisation simple.....	24
C- Utilisation avec GROUP BY	24
5- Les concaténations éphémères (2).....	25
Afficher seulement une partie d'une colonne :.....	26
A- Afficher la date	27
B- Modification de l'affichage	27
C- Calculer une moyenne.....	28
D- Sélectionner selon une longueur.....	28
E- Trier	28
F- Sélectionner une valeur entre 2 intervalles	28
G- Différence entre 2 dates.....	29
Différence entre 2 dates	29
Nombre entre 2 intervalles.....	29
Ajouter ou soustraire des jours à une date	30
Trier par rapport à une date de naissance.....	30
H- Changer la casse du texte	30
Mettre en majuscule de façon permanente.....	30
Le mot limit	31
Delete seulement une partie d'une table.....	31
Réaliser un update mais partiel	31
Baisser le grade / chiffre donné d'une valeur choisie (ici le grade des hommes)	31
Ajout d'un jour à une date uniquement à une valeur donnée	31
Arrondir à un certain nombre de chiffres après la virgule :	31
Créer une colonne qui affiche un chiffre en décimal :.....	31
Requête dans une requête avec le mot in :.....	31
Afficher un jour précis de manière simple :.....	32

MySQL

6-	Les primary keys contenant plusieurs champs	32
A-	Select plusieurs tables en même temps :.....	32
B-	Créer une seconde table qui a les mêmes propriétés que la première :.....	32
C-	Enregistrer une requête pour éviter de la retaper :	33
D-	Pour voir quelle formule a été utilisée pour créer une view :	33
E-	Afficher toutes les données d'une colonne sans les redondances :.....	33
F-	Pour faire une recherche de quelque chose de null :.....	33
G-	Lier une table à plusieurs foreign keys :	34
H-	Faire une recherche grâce à une requête imbriquée :.....	34
7-	Les foreign keys :.....	35
8-	Les jointures	35
A-	Afficher toutes les données d'une colonne sans les redondances	36
B-	Pour faire une recherche de quelque chose de null	36
C-	Lier une table à plusieurs foreign keys :	36
D-	Faire une recherche grâce à une requête imbriquée.....	37
E-	Notes	38
9-	Tables fonctionnelles	39
A-	Syntaxe du texte d'entrée	39
	Principe	39
	Définir une entité	39
B-	Les attributs sont séparés par des virgules.....	39
	Définir une association	39
	Spécifier une mise en page	41
10-	EXERCICES SQL (sur base MySQL)	44
1-	Exercice1.....	44
11-	EXERCICES SQL (sur base MySQL)	55
2-	Exercice 2.....	55
12-	EXERCICES SQL (sur base MySQL)	60
3-	Exercice 3	60



Aide mémoire SQL

©sql.sh

Requêtes SQL

SELECT *	# Sélection des colonnes
FROM table	# Nom d'une ou plusieurs tables
WHERE condition	# Obtenir les résultats selon la condition
GROUP BY expression	# Grouper les tables en groupe
HAVING condition	# Condition sur un groupe
{ UNION INTERSECT EXCEPT }	# Unir plusieurs requêtes
ORDER BY expression	# Trier les résultats
LIMIT count	# Limiter à N enregistrements
OFFSET start	# Débuter à partir N enregistrement
SELECT * FROM table	
INNER JOIN table2 ON table.id = table2.id	# Jointures de 2 tables
SELECT LAST_INSERT_ID() as new	# Retourner l'ID du dernier INSERT
SELECT COUNT(*) FROM table	# Retourner le nombre de lignes
INSERT INTO table VALUES ('valeur 1', 'valeur 2', ...)	# Insérer un enregistrement
UPDATE table SET nom_colonne_1 = 'nouvelle valeur'	# Modifier un enregistrement
WHERE condition	
DELETE FROM table WHERE condition	# Supprimer un enregistrement

Types de données

Colonnes numériques	Colonnes de texte	Colonnes temporelles	Fonctions GROUP BY
TINYINT	CHAR	DATE	AVG
SMALLINT	VARCHAR	DATETIME	COUNT
MEDIUMINT	TINYTEXT, TINYBLOB	TIMESTAMP	MAX
INT, INTEGER	TEXT, BLOB	TIME	MIN
BIGINT	LONGTEXT, LONGBLOB	YEAR	SUM
FLOAT	ENUM		BIT_AND
DOUBLE PRECISION, REAL	SET		BIT_OR
DECIMAL			BIT_XOR

Fonctions

Maths	Dates et heures	Chaînes de caractères
ABS	ADDDATE	ASCII
ACOS	ADDTIME	BIN
ASIN	CONVERT_TZ	BIT_LENGTH
ATAN	CURDATE	CHAR_LENGTH
CEIL	CURRENT_DATE	CHAR
CEILING	CURTIME	CHARACTER_LENGTH
CONV	CURRENT_TIME	CONCAT_WS
COS	CURRENT_TIMESTAMP	CONCAT
COT	DATE_ADD	ELT
CRC32	DATE_FORMAT	EXPORT_SET
DEGREES	DATE_SUB	FIELD
EXP	DATE	FIND_IN_SET
FLOOR	DATEDIFF	FORMAT
LN	DAYNAME	HEX
LOG10	DAYOFMONTH, DAY	INSERT
LOG2	DAYOFWEEK	INSTR
LOG	DAYOFYEAR	LCASE
MOD	EXTRACT	LEFT
PI	FROM_DAYS	LENGTH
POW	FROM_UNIXTIME	LIKE
POWER	GET_FORMAT	LOAD_FILE
RADIANS	HOUR	LOCATE
RAND	LAST_DAY	LOWER
SIGN	LOCALTIME	LPAD
SIN	LOCALTIMESTAMP	LTRIM
SQRT	MAKEDATE	MAKE_SET
TAN	MAKETIME	
TRUNCATE	MICROSECOND	
	MINUTE	
	YEAR	
	YEARWEEK	

MySQL

1. PHP

1- Description

Leur intérêt premier est de stocker, mais elles sont aussi utiles pour extraire des informations au bon vouloir, trier des données, ect...

Les bases de données les plus connues sont :

SGBDR (système de gestion des bases de données relationnelles) :

- DB2(IBM)
- SQL Server (MS)
- Oracle DB (Oracle)
- MySQL (Sun Oracle) la plus utilisée
- Maria DB (fork de MySQL) deuxième plus utilisée
- Postgres (open source)
- Mongo DB.(base non SQL)

Les bases de données sont pensées comme des tables (de la même manière qu'un tableau excel).

Dans les bases SQL, toutes les lignes ont la même structure (un tableau où chaque colonne définit l'information relative à la ligne sur un sujet donné).

2- Les avantages des bases de données :

1) Intégration :

L'intégration permet d'avoir un seul fichier qui sert à plusieurs domaines pour avoir une plus grande flexibilité.

Par exemple, le service commercial et le service RH ont une base de données en commun afin de ne pas avoir à remplir deux fois les informations.

- L'indépendance entre données et traitements ;
- La duplication des données est réduite ;
- La base de données dote l'entreprise d'un contrôle centralisé de données opérationnelles qui représentent le capital important de l'entreprise ;
- L'ordre dans le stockage de données ;
- L'utilisation simultanée des données par différents utilisateurs.

2) Flexibilité

3) Disponibilité :

4) Sécurité :

Tolérance aux failles

Tolérances aux erreurs (responsabilité de l'admin)

Tolérance aux actes malveillants (rôle du développeur)

MySQL

Le dev de la BD doit gérer la base de données pour qu'elle soit la plus juste possible et doit avoir une tolérance aux erreurs.

Son code doit également empêcher les actes malveillants envers elle et donc avoir une tolérance aux actes malveillants.

Enfin, il faut faire attention aux pannes, ce qui est le job de l'admin réseau.

2. MySQL

Base de données dominante (100% d'utilisation en France, plus de 95% dans le monde) grâce à sa licence très permissive au niveau commerciale notamment, cette base de données est incontournable pour tout développeur web qui se respecte.

1- Se connecter à sa base de données

Démarrer le serveur de BDD Uwamp
(pour se connecter à sa base de données)
Dans le fichier de wamp, il faut aller dans :

Bin/
Database/
MySQL-x.y.z/
Bin/

Ensuite faire shift+clic droit => ouvrir une fenêtre de commandes ici.

Quand on rentre une commande, elle doit **IMPERATIVEMENT** se terminer par un ;

Pour se connecter à la BD via la console il faut taper :

mysql -u root(le nom d'utilisateur) -p

il faut ensuite rentrer son mdp (pour nous, root)

Ou ouvrir une fenêtre d'invite de commande en tapant **cmd** dans la barre de recherche windows :

Avec la fenêtre de commande : (avec powershell, indiquer le chemin .\mysql)

Ensute taper :

cd Desktop\UwAmpDW2M\bin\database\mysql-5.7.11\bin

on obtient :

C:\Users\62014-06-08\Desktop\UwAmpDW2M\bin\database\mysql-5.7.11\bin>

taper ensuite ceci → **mysql -u root -p <J**
 Taper le mot de passe : **root <J**

```
Microsoft Windows [version 10.0.17134.285]
(c) 2018 Microsoft Corporation. Tous droits réservés.

C:\Users\62014-06-08>cd Desktop\UwAmpDW2M\bin\database\mysql-5.7.11\bin

C:\Users\62014-06-08\Desktop\UwAmpDW2M\bin\database\mysql-5.7.11\bin>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.11 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

2- Les commandes → CRUD (Create Read Update Delete)

Select version();	pour savoir quelle version de database est utilisée
Show databases;	quelles sont les bdd présentent dans le serveur
Create database NOMBD;	créer une base de données
Use NOMBD;	utiliser la bdd en particulier
Show tables;	voir la liste des dossiers dans la bdd
Drop database NOMBD;	supprimer une bdd
Drop table NOMBD;	supprimer une table
Describe NOMTABLE;	permet d'avoir la description de la table
Create table if not exists tb1;	créer la table si elle n'existe pas
Drop table if exists tb1;	effacer la table si elle existe
Truncate table tb1;	vider la table sans la supprimer
Select * from tb1;	sélectionne les colonnes
Select col1 from tb1;	sélectionne la colonne spécifiée

A- Les types les plus utilisés en Caractères

Int : permet de rentrer des valeurs numériques.

Char(x) : permet de rentrer x caractères alpha numériques. Par exemple pour un char(4), si on ne met qu'une lettre, les trois espaces restants seront comblés par des espaces. C'est

plus utile pour les valeurs fixes, comme les années qui prendront toujours 4 espaces ou les codes postaux qui en prennent toujours 5. ***Le x peut aller jusqu'à 255 maximum.***

Varchar(x) : Contrairement au char(x), ici le reste ne sera pas comblé par des espaces. On peut utiliser x nombre de caractères au maximum. Si on met varchar(5) et qu'on ne met qu'une lettre, alors le serveur dira qu'il n'y a qu'un espace d'utilisé, ce qui permet d'économiser de la place dans les données. ***Le x peut aller jusqu'à 65 000.***

```
mysql> create table tb1 (
    -> num int,
    -> nom varchar(64),
    -> email varchar(64)
    -> );
```

⚠️ **On ne met pas de virgule après le dernier élément créé**, sinon la console va s'attendre à ce qu'on veuille en rajouter un. Ici, il n'y a pas de virgule après le mail.

B- Les attributs

Les attributs sont des facteurs qui vont être appliqués à des données entrées dans les tables. Si on ne précise rien, la donnée est null par défaut.

Les attributs les plus utilisés sont :

not null (il faut nécessairement rentrer quelque chose)

default a moins d'être modifiée, la valeur donnée sera la valeur par défaut.

primary key (est une sorte d'ID qui permet de distinguer une ligne de manière unique.)

unique (Utile par exemple pour les numéros de factures).

```
mysql> create table tb2(
    -> num int not null,
    -> ville varchar(24) default 'Lens',
    -> email varchar(24)
    -> );
Query OK, 0 rows affected (0.33 sec)

mysql> describe tb2
      -> ;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| num   | int(11)   | NO   |     | NULL    |       |
| ville | varchar(24)| YES  |     | Lens    |       |
| email | varchar(24)| YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

C- Insérer les infos sur une table :

Avant tout, vérifier la db sur laquelle on travaille avec :

Select database();

Ensuite il faut taper :

Insert into nomdetable(colonnex,colonnex,... selon ce qu'on veut insérer dans telle colonne)

Values(val1,val2,valx...);

MySQL

L'ordre est très important parce que la value1 ira se placer dans la première colonne appelée.

```
mysql> insert into tb2(num,email) values(25,"email@gmail.com");
Query OK, 1 row affected (0.03 sec)
```

Il est possible de ne pas préciser les colonnes. Mais dans ce cas elles seront toutes visées par défaut et il faudra donc intégrer *TOUTES* les informations.

D- Insérer des informations dans une table

```
Insert into tb1 (col1,col2,col3) (25,'prof@essai.com' ;
```

Values

```
(val1,val2,val3) ; Insert into tb2(ville)
```

```
Insert into tb2(num, mail)
```

Value

```
(‘Paris’) ;
```

Values

```
C:\Users\62014-06-08\Desktop\UwAmpDW2M\bin\database\mysql-5.7.11\bin>mysql.exe -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.11 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select version();
+-----+
| version() |
+-----+
| 5.7.11   |
+-----+
1 row in set (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bdd1 |
| mysql |
| performance_schema |
| phpmyadmin |
| sys |
| test1 |
| test2 |
+-----+
8 rows in set (0.02 sec)

mysql> create database zaza
      -> \c
mysql> create database zaza;
Query OK, 1 row affected (0.00 sec)

mysql> use zaza;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> show tables;
Empty set (0.00 sec)

mysql> drop database zaza;
Query OK, 0 rows affected (0.02 sec)
```

MySQL

Ici nous avons créé la base zaza, puis sommes allés dans la base voir les tables (il n'y en a pas)

Voir les tables d'une base : **show tables** ; (ici le contenu de la bdd1)

```
mysql> use database bdd1;
ERROR 1049 (42000): Unknown database 'database'
mysql> use bdd1;
Database changed
mysql> show tables;
+-----+
| Tables_in_bdd1 |
+-----+
| wp_commentmeta
| wp_comments
| wp_links
| wp_options
| wp_postmeta
| wp_posts
| wp_term_relationships
| wp_term_taxonomy
| wp_termmeta
| wp_terms
| wp_usermeta
| wp_users
+-----+
12 rows in set (0.00 sec)
```

Création de 2 bases de données : **Create Table nomtable**

```
C:\Users\62014-06-08\Desktop\UwAmpDW2M\bin\database\mysql-5.7.11\bin>mysql.exe -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.11 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bdd1 |
| essai |
| mysql |
| performance_schema |
| phpmyadmin |
| sys |
| test1 |
| test2 |
+-----+
9 rows in set (0.00 sec)
```

MySQL

```
mysql> use essai;
Database changed
mysql> create table tb1(
    -> num int,
    -> nom varchar(64),
    -> mail varchar(64)
    -> );
Query OK, 0 rows affected (0.20 sec)

mysql> show tables;
+-----+
| Tables_in_essai |
+-----+
| tb1           |
+-----+
1 row in set (0.00 sec)

mysql> create table tb2(
    -> num int not null,
    -> ville varchar(24) default 'lens',
    -> email varchar(24)
    -> );
Query OK, 0 rows affected (0.17 sec)

mysql> show tables;
+-----+
| Tables_in_essai |
+-----+
| tb1           |
| tb2           |
+-----+
2 rows in set (0.00 sec)
```

Description des bases de données avec **describe tb1 ;**

```
mysql> show tables;
+-----+
| Tables_in_essai |
+-----+
| tb1           |
| tb2           |
+-----+
2 rows in set (0.00 sec)

mysql> describe tb1;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| num   | int(11)   | YES  |     | NULL    |       |
| nom   | varchar(64)| YES  |     | NULL    |       |
| mail  | varchar(64)| YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> describe tb2;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| num   | int(11)   | NO   |     | NULL    |       |
| ville | varchar(24)| YES  |     | lens    |       |
| email | varchar(24)| YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Pour créer une requête affichant ‘bonjour, bienvenu dans le monde de SQL’. Le nom de la table étant HELLO :

```
mysql> select "Bonjour, Bienvenue dans le monde SQL";
+-----+
| Bonjour, Bienvenue dans le monde SQL |
+-----+
| Bonjour, Bienvenue dans le monde SQL |
+-----+
1 row in set (0.00 sec)

mysql> select "Bonjour, Bienvenue dans le monde SQL" as HELLO;
+-----+
| HELLO |
+-----+
| Bonjour, Bienvenue dans le monde SQL |
+-----+
```



Exo

Créer une base BD ‘DWWM’

Créer la table stagiaire

Id not null (id int not nul → nombre entier non nul)

Nom(48)

Prénom(48)

Email(64)

Tel(12)

On utilise **as** pour renommer la colonne éphémère

Si on ne peut pas écrire comme on le souhaite dans la fenêtre d’invit de commande (pas de retour en arrière, pas de copié/collé), on peut mettre les valeurs dans un fichier source externe ‘par exemple avec VS code) et l’insérer comme suit :

Source fichier.sql ; exécute toutes les instructions du fichier sql

(dans fichier sql) :

Ensuite insérer les valeurs dans la table :

Insert into stagiaire values

(1,’paul’,’paul@free.fr’),

(2,’Marie’,’Marie@gmail.com’) ;  ne pas oublier de mettre les caractères entre ‘....’

MySQL

```

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bdd1 |
| essai |
| mysql |
| performance_schema |
| phpmyadmin |
| sys |
| test1 |
| test2 |
+-----+
9 rows in set (0.00 sec)

mysql> create database DWMM;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bdd1 |
| dwmm |
| essai |
| mysql |
| performance_schema |
| phpmyadmin |
| sys |
| test1 |
| test2 |
+-----+
10 rows in set (0.00 sec)

mysql> use dwmm;
Database changed

mysql> create table stagiaire(
    -> id int not null,
    -> Nom varchar(48),
    -> Prénom varchar(48),
    -> Email varchar(64),
    -> Tel char(12)
    -> );
Query OK, 0 rows affected (0.24 sec)

mysql> show tables;
+-----+
| Tables_in_dwmm |
+-----+
| stagiaire |
+-----+
1 row in set (0.00 sec)

mysql> describe stagiaire;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id   | int(11)   | NO  |   | NULL    |       |
| Nom  | varchar(48)| YES |   | NULL    |       |
| Prénom | varchar(48)| YES |   | NULL    |       |
| Email | varchar(64) | YES |   | NULL    |       |
| Tel  | char(12)  | YES |   | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>

```

E- Insertion de valeurs dans une table donnée

```

mysql> insert into stagiaire (id,Nom,Prénom,Email,Tel)
    -> values
    -> (1,'Juszak','Rémy','RémyJuszak@free.fr',060102030405),
    -> (2,'Lacroix','Annick','annick@free.fr',0615010203);
Query OK, 2 rows affected (0.05 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> describe stagiaire
    -> \c
mysql> describe stagiaire;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id   | int(11)   | NO  |   | NULL    |       |
| Nom  | varchar(48)| YES |   | NULL    |       |
| Prénom | varchar(48)| YES |   | NULL    |       |
| Email | varchar(64) | YES |   | NULL    |       |
| Tel  | char(12)  | YES |   | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql> select * from stagiaire;
+---+---+---+---+---+
| id | Nom     | Prénom | Email           | Tel      |
+---+---+---+---+---+
| 1  | Juszak  | Rémy   | RémyJuszak@free.fr | 60102030405 |
| 2  | Lacroix | Annick | annick@free.fr   | 615010203  |
+---+---+---+---+---+
2 rows in set (0.00 sec)

```

⚠ La dernière ligne se termine par un ;

```
mysql> insert into stagiaire (id,Nom,Prénom,Email,Tel)
-> values
-> (1,'Jussak','Rémy','RémyJuszak@free.fr',0601020304), → ! mettre les caractères
entre ' '
-> (2,'Lacroix','Annick','annick@free.fr',0615010203); → ! mettre un ; à la dernière ligne
```

Un exemple de « vraie » base de donnée créée

```
mysql> CREATE TABLE stagiaire(id int not null,
-> nom varchar(48),
-> prenom varchar(48),
-> email varchar(64),
-> tel varchar(12)
-> );
Query OK, 0 rows affected (0.20 sec)

mysql> describe stagiaire;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id    | int(11) | NO   |     | NULL    |       |
| nom   | varchar(48)| YES  |     | NULL    |       |
| prenom | varchar(48)| YES  |     | NULL    |       |
| email  | varchar(64) | YES  |     | NULL    |       |
| tel    | varchar(12) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

mysql> describe stagiaire; → voir si la table est bien créée

```
mysql> insert into stagiaire(id,nom,prenom,email,tel)
-> value
-> (11,"TANCREZ","Antoine","antoine.tancrez@gmail.com","0642897208"),
-> (10,"ZZZZZ","Brian","brian@gmail.com","0689897562"),
-> (09,"YUSJEZ","François","François@gmail.com","0678452102"),
-> (08,"GFDYEJHDQ","Adrien","adrien@gmail.com","0656230285"),
-> (07,"GFDDPOQ","Kevin","kevin@gmail.com","0698023872"),
-> (06,"FRAHSOA","Remi","remi@gmail.com","0678257852"),
-> (05,"DRAFSJZZASFE","Annick","annick@gmail.com","0625566992"),
-> (04,"DRAFE","Alain","alain@gmail.com","0625566942"),
-> (03,"DJJDHGEBE","Quentin","quentin@gmail.com","0644110022"),
-> (02,"HETMANN","Maxime","Maxime@gmail.com","0622558899"),
-> (01,"HJIOAMKDJDGHGA","Donovan","donovan@gmail.com","0644556677"),
-> (00,"DJHDGHGA","Saada","saada@gmail.com","0688770033");
Query OK, 12 rows affected (0.03 sec)
Records: 12  Duplicates: 0  Warnings: 0
```

On peut intégrer plusieurs lignes en même temps, il suffit de séparer les différentes values par une virgule et mettre un point-virgule à la dernière.

id	nom	prenom	email	tel
11	TANCREZ	Antoine	antoine.tancrez@gmail.com	0642897208
10	ZZZZZ	Brian	brian@gmail.com	0689897562
9	YUSJEZ	François	François@gmail.com	0678452102
8	GFDYEJHDQ	Adrien	adrien@gmail.com	0656230285
7	GFDDPOQ	Kevin	kevin@gmail.com	0698023872
6	FRAHSOA	Remi	remi@gmail.com	0678257852
5	DRAFSJZZASFE	Annick	annick@gmail.com	0625566992
4	DRAFE	Alain	alain@gmail.com	0625566942
3	DJJDHGBE	Quentin	quentin@gmail.com	0644110022
2	HETMANN	Maxime	Maxime@gmail.com	0622558899
1	HJIOAMKDJDHGHA	Donovan	donovan@gmail.com	0644556677
0	DJHDGHGA	Saada	saada@gmail.com	0688770033

Pour connaître le nombre total de lignes dans une table : select count (*) from nomtable ;

F- Pour importer une base de donnée

On peut taper la base de données sur un fichier texte exactement de la même manière que ce qu'on aurait tapé sur la console.

Sur la console on rentre :

source sourcedufichier ;

Techniquement, on peut tout taper sur un éditeur de texte et l'importer sur la console, donc on peut même créer des bases de données, tables ect.

Il faut s'identifier avant d'importer le fichier.

G- Modifier la structure d'une table

Ajouter une colonne :

Alter table nomtable add col x type x ;

Supprimer une colonne :

Alter table nomtable drop col x

/ou drop column col x ;

Modifier une colonne :

Alter table nomtable /ou modify col x type x ;

Changer le nom d'une colonne :

Alter table nomtable change old-col new-col typex ; (**Alter table nomtable change anciennom nouveaunom typex ;**)

Pour écrire un commentaire :

On peut faire --text en commentaire et le commentaire fera toute la ligne.

On peut aussi utiliser dièse ou double dièse.

La meilleure solution reste de faire /* commentaire */

H- Trier les informations

On peut comme précisé plus haut afficher tout ou partie d'un tableau :

Select column from table ;

Mais on peut les trier par rapport à une colonne en particulier (même si celle-ci n'est pas affichée dans la sélection) :

Select colx from table order by coly ; on trie la colonne x par rapport aux données de la colonne y.

Si on ne précise pas, le tri se fera dans l'ordre croissant des valeurs.

Extraire des colonnes :

Select * from stagiaire ;	extrait tout
Select nom from stagiaire ;	extrait la colonne demandée
Select * from stagiaire limit 3	extraire que 3 lignes à partir du début

```
mysql> select * from stagiaire;
+---+-----+-----+-----+-----+
| id | Nom    | Prénom | Email           | Tel
+---+-----+-----+-----+-----+
| 1  | Juszak | Rémy   | RémyJuszak@free.fr | 60102030405 |
| 2  | Lacroix| Annick | annick@free.fr   | 615010203  |
+---+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select Nom from stagiaire;
+-----+
| Nom  |
+-----+
| Juszak |
| Lacroix |
+-----+
2 rows in set (0.00 sec)
```

Select * from stagiaire limit N ou limit N,M

Pour trier dans l'ordre inverse, il faut mettre :

Select colx from table order by coly *desc* ; à la fin de la ligne de commande.

```
mysql> select nom,prenom from tab_test order by age desc;
+-----+-----+
| nom  | prenom |
+-----+-----+
| tata | Michel Loebbe |
| tutu | Clovis |
| titi | Charlemagne |
| toto | jean-noel |
+-----+-----+
4 rows in set (0.00 sec)
```

MySQL

Pour regarder une base de données soit sur un nombre N ou M lignes à partir d'un nombre N :

Select * from table limit N ; Ici, on prendra les N premières lignes.

Select * from table limit N,M ; Ici, on affichera a partir de la ligne N M lignes.

Afficher le nombre d'éléments d'une colonne :

Pour afficher combien de lignes contient une colonne : Select count(col) from table ;

I- Modifier le contenu d'une colonne

```
mysql> select * from stagiaire;
+----+-----+-----+-----+
| id | Nom   | Prénom | Email        | Tel   |
+----+-----+-----+-----+
| 1  | Juszak | Rémy   | RémyJuszak@free.fr | 60102030405 |
| 2  | Lacroix | Annick | annick@free.fr   | 615010203  |
+----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Dans notre exemple, on change
le téléphone du prénom Rémy :

Update stagiaire

Set tel= « +33610203040 »

Where Prénom like « Rémy » ;

On utilise **select*from stagiaire** pour
vérifier la modification.

Quand il y a plusieurs données à changer,
on met une virgule derrière la ligne

contenant **set**, et on écrit à la suite :

Set tel= « +33610203040 », age=30,

J- Extraire des données

Extrait les données comprenant :

Select * from tbl where ville : « Paris » ;
Age<25 ;
Age>18 ;
Nom : »Paul »

```
mysql> update stagiaire
      -> set Tel="0601020304"
      -> where Prénom like "Rémy";
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from stagiaire
      -> ;
+----+-----+-----+-----+
| id | Nom   | Prénom | Email        | Tel   |
+----+-----+-----+-----+
| 1  | Juszak | Rémy   | RémyJuszak@free.fr | 0601020304 |
| 2  | Lacroix | Annick | annick@free.fr   | 615010203  |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> update stagiaire
      -> set Tel="0615010203"
      -> where id=2;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from stagiaire
      -> ;
+----+-----+-----+-----+
| id | Nom   | Prénom | Email        | Tel   |
+----+-----+-----+-----+
| 1  | Juszak | Rémy   | RémyJuszak@free.fr | 0601020304 |
| 2  | Lacroix | Annick | annick@free.fr   | 0615010203  |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Sélectionner un intervalle de données :

L'opérateur **BETWEEN** est utilisé dans une requête SQL pour sélectionner un intervalle de données dans une requête utilisant **WHERE**. L'intervalle peut être constitué de chaînes de

caractères, de nombres ou de dates. L'exemple le plus concret consiste par exemple à récupérer uniquement les enregistrements entre 2 dates définies.

L'utilisation de la commande BETWEEN s'effectue de la manière suivante :

```
SELECT * FROM table WHERE nom_colonne BETWEEN valeur1 AND valeur2 ;
```

La requête suivante retournera toutes les lignes dont la valeur de la colonne "nom_colonne" sera comprise entre **valeur1** et **valeur2**.

Rechercher

Like « _ »	recherche un mot indiqué.
Like «%_»	Recherche une expression finissant par ...
Like «_%»	recherche une expression commençant par...
Like «%_%»	recherche une expression comprenant...

Exemple :

Like « n% » recherche les expression commençant par n

Select * from stagiaire where prenom like « on » ;

va extraire tous les prénom avec un « on » à l'intérieur

Select * from table where colonne = « mot ou chiffre recherché » ;

On peut utiliser les signes < et > pour les chiffres.

Il est possible de faire :

Select * from table where colonne like « %donnée% » ;

Le like permet d'afficher tout ce qui contient ce qu'il y a dans les % même si ce n'est qu'une partie du mot qui sera retourné.

Sur MySQL :

Quand on met like donnée, le like vaudra le = . Dans la pratique, ce n'est pas utilisé car le = le fait déjà.

Si on met like donnée% alors ça affichera tout ce qui commence par la donnée.

Si on met like %donnée, alors ça affichera tout ce qui finira par la donnée.

Si on veut par exemple les données qui commencent par un a et qui contiennent un b dans le mot alors il faut mettre :

Like « a%b% » ;

Si on veut par exemple que les prénoms qui commencent par un a et qui ont un t en troisième position :

Like « a_t% » ; chaque caractère vide sera remplacé par un underscore.

MySQL

```
mysql> select * from stagiaire where nom="TANCREZ";
+----+-----+-----+-----+-----+
| id | nom      | prenom   | email           | tel    |
+----+-----+-----+-----+-----+
| 11 | TANCREZ  | Antoine  | antoine.tancrez@gmail.com | 0642897208 |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from stagiaire where prenom like "a%";
+----+-----+-----+-----+-----+
| id | nom      | prenom   | email           | tel    |
+----+-----+-----+-----+-----+
| 11 | TANCREZ  | Antoine  | antoine.tancrez@gmail.com | 0642897208 |
| 8  | GFDYEJHDQ | Adrien   | adrien@gmail.com  | 0656230285 |
| 5  | DRAFSJZZASFE | Annick  | annick@gmail.com | 0625566992 |
| 4  | DRAFE    | Alain    | alain@gmail.com  | 0625566942 |
+----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Mettre à jour des données :

Si on veut mettre à jour toutes les données d'une colonne pour donner la même donnée :

Update nomtable set colonne = « donnée » ;

Si on veut modifier les données d'une ligne en particulier ou alors cibler plus en détails :

Update nomtable set colonne = « nouvelle donnée » where colonne= « donnée » ;

```
mysql> update stagiaire set prenom="Leinhart" where prenom="Antoine";
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> select * from stagiaire;
+----+-----+-----+-----+-----+
| id | nom      | prenom   | email           | tel    |
+----+-----+-----+-----+-----+
| 11 | TANCREZ  | Leinhart | antoine.tancrez@gmail.com | 0642897208 |
+----+-----+-----+-----+-----+
```

Imaginons qu'on ait une liste de numéros de téléphones.

Si on veut update en remplaçant les lignes sans numéros par « non communiqué », il faut mettre :

Update nomtable set tel = « non communiqué » where tel= « » ;

Ici, les espaces vides sont ciblés par les guillemets vides.

Delete des données :

On utilisera toujours une condition pour ne pas supprimer toute une colonne. Il faut faire attention car

Delete from table ; fonctionne et supprime tout et videra le tableau.

En général on mettra :

Delete from table where colonne= « donnée » ;

Par exemple, si on veut enlever les lignes qui n'ont pas de numéro de téléphone on mettra :

Delete from table where tel= « » ;

Si on veut sélectionner parmi un critère ou un autre on peut faire

Select* from table where colonne in (« data1 », « data2 »,...) ;

Ici, ça sera les données qui sont soit le data1 ou le data2.

Select* from table where colonne not in (« data1 », « data2 »,...) ;

Si on veut sélectionner tout ce qui ne sera pas le data1 ou le data2.

On peut utiliser le mot **between** pour chercher entre deux valeurs ordonnées :

Select* from table where age between « 18 » and « 50 » ;

Ici, on va sélectionner tout ce qui est entre 18 et 50 compris.

mysql> select * from stagiaire where id between "5" and "9";				
id	nom	prenom	email	tel
9	YUSJEZ	François	François@gmail.com	0678452102
8	GFDYEJHDQ	Adrien	adrien@gmail.com	0656230285
7	GFDDPOQ	Kevin	kevin@gmail.com	0698023872
6	FRAHSOA	Remi	remi@gmail.com	0678257852
5	DRAFSJZZASFE	Annick	annick@gmail.com	0625566992

3- Les Fonctions

A- Les concaténations éphémères (1)

Elles servent par exemple à regrouper deux colonnes pour voir les données affichées ensemble (uniquement pour la requête).

Select Concat(col,col,col) from table;

Pour afficher les deux colonnes ensemble.

On peut renommer cette colonne éphémère avec un alias :

Select Concat(col,col,col) as « nom » from table;

On peut afficher les résultats d'une colonne tout en majuscule ou tout en minuscule :

Select ucase(col) from table ; Pour les majuscules.

Select lcase(col) from table ; Pour les minuscules.

```
mysql> select uc case(prenom) from stagiaire;
+-----+
| uc case(prenom) |
+-----+
| LEINHART      |
| BRIAN         |
| FRAN OIS       |
| ADRIEN        |
| KEVIN          |
| REMI           |
| ANNICK         |
| ALAIN          |
| QUENTIN        |
| MAXIME         |
| DONOVAN        |
+-----+
11 rows in set (0.00 sec)
```

Il est possible d'ajouter dans la concaténation des valeurs qui ne sont pas des colonnes. Il suffit de les mettre entre guillemets.

Aussi, l'alias « as » peut-être utilisé tout le temps.

```
mysql> select concat("[",nom,"=>",prenom,"]")as "Relations" from stagiaire;
+-----+
| Relations      |
+-----+
| [TANCREZ=>Leinhart] |
| [ZZZZZ=>Brian]    |
| [YUSJEZ=>Fran ois] |
| [GFDYEJHDQ=>Adrien] |
| [GFDDPOQ=>Kevin]   |
| [FRAHSOA=>Remi]    |
| [DRAFSJZZASFE=>Annick] |
| [DRAFE=>Alain]     |
| [DJJDHGBE=>Quentin] |
| [HETMANN=>Maxime]   |
| [HJIOAMKDJDGHGA=>Donovan] |
+-----+
11 rows in set (0.00 sec)
```

Ici, la colonne éphémère Relations contient les deux colonnes nom et prénom qui sont séparées par des caractères ajoutés pour l'affichage.

Afficher seulement une partie d'une colonne :

Il est possible d'afficher seulement une partie d'une colonne :

Select substring(col,1,5) from table ;

Ici, le premier chiffre détermine à partir de quel numéro de caractère on va afficher et le deuxième de combien de caractères on va afficher :

MySQL

```
mysql> select substring(prenom,2,4) from stagiaire;
+-----+
| substring(prenom,2,4) |
+-----+
| einh
| rian
| ranç
| drie
| evin
| emi
| nnic
| lain
| uent
| axim
| onov
+-----+
11 rows in set (0.00 sec)
```

Afficher la date :

```
Select date(now());
```

```
mysql> select date(now())as "aujourd'hui";
+-----+
| aujourd'hui |
+-----+
| 2019-02-01 |
+-----+
1 row in set (0.00 sec)
```

On peut modifier l'affichage :

Select date_format(now(), »%y-%M-%d %h-%i-%s ») as « today » ;

%y signifie l'année

%m signifie le mois. **%M** l'affiche en lettres.

%d signifie le jour

%h signifie l'heure

%i signifie les minutes

%s signifie les secondes

```
mysql> select date_format(now(), "%y-%M-%d %h-%i-%s") as "today";
+-----+
| today |
+-----+
| 19-February-01 11-20-42 |
+-----+
1 row in set (0.00 sec)
```

4- Fonctions d'agrégation SQL

Les fonctions d'agrégation dans le langage SQL permettent d'effectuer des opérations statistiques sur un ensemble d'enregistrement. Étant données que ces fonctions s'appliquent à plusieurs lignes en même temps, elles permettent des opérations qui servent à récupérer l'enregistrement le plus petit, le plus grand ou bien encore de déterminer la valeur moyenne sur plusieurs enregistrements.

A- Liste des fonctions d'agrégation statistiques

Les fonctions d'agrégation sont des fonctions idéales pour effectuer quelques statistiques de bases sur des tables. Les principales fonctions sont les suivantes :

<u>AVG()</u>	pour calculer la moyenne sur un ensemble d'enregistrement
<u>COUNT()</u>	pour compter le nombre d'enregistrement sur une table ou une colonne distincte
<u>MAX()</u>	pour récupérer la valeur maximum d'une colonne sur un ensemble de ligne. Cela s'applique à la fois pour des données numériques ou alphanumérique
<u>MIN()</u>	pour récupérer la valeur minimum de la même manière que MAX()
<u>SUM()</u>	pour calculer la somme sur un ensemble d'enregistrement

B- Utilisation simple

L'utilisation la plus générale consiste à utiliser la syntaxe suivante :

```
SELECT fonction(colonne) FROM table
```

La fonction **COUNT()** possède une subtilité. Pour compter le nombre total de ligne d'une table, il convient d'utiliser l'étoile "*" qui signifie que l'on cherche à compter le nombre d'enregistrement sur toutes les colonnes. La syntaxe serait alors la suivante :

```
SELECT COUNT(*) FROM table
```

```
mysql> select count(note1+note2+note3) as "moyenne classe" from tb1;
+-----+
| moyenne classe |
+-----+
|          12   |
+-----+
```

C- Utilisation avec GROUP BY

Toutes ces fonctions prennent tout leur sens lorsqu'elles sont utilisée avec la commande **GROUP BY** qui permet de filtrer les données sur une ou plusieurs colonnes. Imaginons une table qui contient tous les achats sur un site avec le montant de chaque achat pour chaque enregistrement. Pour obtenir le total des ventes par clients, il est possible d'exécuter la requête suivante :

```
SELECT client, SUM(tarif)
FROM achat
```

GROUP BY client

Le résultat sera le suivant :

client	SUM(tarif)
Pierre	262
Simon	47
Marie	38

5- Les concaténations éphémères (2)

Elles servent par exemple à regrouper deux colonnes pour voir les données affichées ensemble (uniquement pour la requête).

Select Concat(col,col,col) from table ; Pour afficher les deux colonnes ensemble.

On peut renommer cette colonne éphémère avec un alias :

Select Concat(col,col,col) as « nom » from table;

On peut **afficher les résultats d'une colonne tout en majuscule ou tout en minuscule** :

Select ucase(col) from table ; Pour les majuscules.

Select lcase(col) from table ; Pour les minuscules.

```
mysql> select ucase(prenom) from stagiaire;
+-----+
| ucase(prenom) |
+-----+
| LEINHART    |
| BRIAN       |
| FRAN OIS    |
| ADRIEN      |
| KEVIN       |
| REMI        |
| ANNICK     |
| ALAIN       |
| QUENTIN    |
| MAXIME     |
| DONOVAN    |
+-----+
11 rows in set (0.00 sec)
```

Il est possible d'ajouter dans la concaténation des valeurs qui ne sont pas des colonnes. Il suffit de les mettre entre guillemets.

Aussi, l'**alias « as »** peut-être utilisé tout le temps.

```
mysql> select concat("[",nom,"=>",prenom,"]")as "Relations" from stagiaire;
+-----+
| Relations           |
+-----+
| [TANCREZ==>Leinhart] |
| [ZZZZZ==>Brian]      |
| [YUSJEZ==>François]  |
| [GFDYEJHDQ==>Adrien] |
| [GFDDPOQ==>Kevin]    |
| [FRAHSOA==>Remi]     |
| [DRAFSJZZASFE==>Annick] |
| [DRAFE==>Alain]      |
| [DJJDHGEBE==>Quentin] |
| [HETMANN==>Maxime]   |
| [HJIOAMKDJHDGHGA==>Donovan] |
+-----+
11 rows in set (0.00 sec)
```

Ici, la colonne éphémère *Relations* contient les deux colonnes *nom* et *prénom* qui sont séparées par des caractères ajoutés pour l'affichage.

Afficher seulement une partie d'une colonne :

Il est possible d'afficher seulement une partie d'une colonne :

Select *substring(col,1,5)* from *table* ;

Ici, le premier chiffre détermine à partir de quel numéro de caractère on va afficher et le deuxième de combien de caractères on va afficher :

```
mysql> select substring(prenom,2,4) from stagiaire;
+-----+
| substring(prenom,2,4) |
+-----+
| einh                |
| rian                |
| ranç                |
| drie                |
| evin                |
| emi                 |
| nnic                |
| lain                |
| uent                |
| axim                |
| onov                |
+-----+
11 rows in set (0.00 sec)
```

A- Afficher la date :

Select date(now());

```
mysql> select date(now())as "aujourd'hui";
+-----+
| aujourd'hui |
+-----+
| 2019-02-01 |
+-----+
1 row in set (0.00 sec)
```

B- Modification de l'affichage

On peut modifier l'affichage :

Select date_format(now(), »%y-%M-%d %h-%i-%s ») as « today » ;

%y signifie l'année

%m signifie le mois. **%M** l'affiche en lettres.

%d signifie le jour

%h signifie l'heure

%i signifie les minutes

%s signifie les secondes

```
mysql> select date_format(now(), "%y-%M-%d %h-%i-%s") as "today";
+-----+
| today |
+-----+
| 19-February-01 11-20-42 |
+-----+
1 row in set (0.00 sec)
```

Afficher le nombre d'éléments d'une colonne :

Pour afficher combien de lignes contient une colonne :

Select count(col) from table ;

```
mysql> select concat(substring(nom,1,1)," ",substring(prenom,1,1)," ",note1) from note order by note1 ;
+-----+
| concat(substring(nom,1,1)," ",substring(prenom,1,1)," ",note1) |
+-----+
| L S 4
| N C 7
| M P 8
| D B 9
| C B 9
| L A 10
| M A 11
| D C 11
| L M 13
| D M 15
| F Y 17
| D P 18
+-----+
12 rows in set (0.00 sec)
```

Ici, on a mis des ***substrings*** dans le ***concat*** pour retirer que les initiales.

C- Calculer une moyenne

Calculer une moyenne :

```
mysql> select avg ((note1+note2+note3)/3) from note;
+-----+
| avg ((note1+note2+note3)/3) |
+-----+
|           12.25000000 |
+-----+
1 row in set (0.00 sec)
```

Select avg (calcul de la moyenne d'une personne) from table ;

D- Sélectionner selon une longueur

On utilise ***length***

```
mysql> select nom as "password faible" from note where length(password)<6;
+-----+
| password faible |
+-----+
| Marchand         |
| Martin           |
| Dalors           |
| Fergusson        |
| Lacour           |
| Dermand          |
| Cazanne          |
+-----+
7 rows in set (0.00 sec)
```

E- Trier

Utiliser le mot ***in*** pour trier :

```
mysql> select concat(nom," ",prenom," ",age) from note where age in(15,25,50);
+-----+
| concat(nom," ",prenom," ",age) |
+-----+
| Dupont Marc 25                |
| Laporte Marie 15              |
+-----+
2 rows in set (0.00 sec)
```

Le mot ***in*** est pratique pour sélectionner les éléments qui font partie de telle, telle ou telle condition. Au lieu de mettre plusieurs ***or*** d'affilé, ici il suffit juste de faire une liste des éléments que l'on veut voir rentrer en tant que condition.

F- Sélectionner une valeur entre 2 intervalles

Le mot ***between*** ou ***not between*** :

En utilisant le mot ***between***, on peut sélectionner tout ce qui est entre telle et telle valeur. Le ***not between*** permet d'exclure tout ce qu'il y a entre les valeurs :

```
mysql> select nom from note where age not between 25 and 45 and(note1<10 or note2<10 or note3<10);
+-----+
| nom   |
+-----+
| Laclasse |
| Cazanne |
+-----+
2 rows in set (0.00 sec)
```

Le mot **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().

```
mysql> select avg(note3) from note group by sexe having avg(note3)>12;
+-----+
| avg(note3) |
+-----+
| 13.0000 |
+-----+
1 row in set (0.00 sec)
```

G- Différence entre 2 dates

Une date est écrite de la façon suivante : '**2018-10-12**' pour le 12 octobre 2018

Différence entre 2 dates

Datediff :

Dans le langage SQL, la fonction **DATEDIFF()** permet de déterminer **l'intervalle entre 2 dates spécifiées**. La fonction est utilisée avec les systèmes MySQL et SQL Server, mais s'utilise différemment :

MySQL : la fonction prend 2 dates en paramètres et retourne le nombre de jours entre les 2 dates

SQL Server : la fonction prend 2 dates ainsi qu'un paramètre déterminant sous quel intervalle de distance temporelle le résultat doit être retourné (nombre de jours entre 2 dates, nombre d'années, nombre de semaines, nombre d'heures ...)

MySQL :

Sous MySQL, la fonction **DATEDIFF()** s'utilise dans une requête avec le format suivant :

Nombre entre 2 intervalles

SELECT DATEDIFF(date1, date2);

Les dates doivent être au format DATE (cf. **AAAA-MM-JJ**) ou DATETIME (cf. **AAAA-MM-JJ HH:MM:SS**). **Pour que le résultat soit positif il faut que la date1 soit plus récente que la date2.**

```
mysql> select datediff(now(),"1971-11-06");
+-----+
| datediff(now(),"1971-11-06") |
+-----+
|          17257 |
+-----+
1 row in set (0.00 sec)
```

Ajouter ou soustraire des jours à une date

Obtenir une date future ou passée en intervalles de jours :

SELECT DATE_ADD(NOW(),INTERVAL 5 DAY); # le futur

SELECT DATE_SUB(NOW(), INTERVAL 5 DAY); # le passé

```
mysql> select date_add(now(),interval 63 day);
+-----+
| date_add(now(),interval 63 day) |
+-----+
| 2019-04-08 10:43:00 |
+-----+
1 row in set (0.00 sec)
```

Trier par rapport à une date de naissance :

```
select * from soldats WHERE timestampdiff(year,date_naissance,now())>25;
```

Cette requête permet de trier les soldats par rapport à leur date de naissance pour afficher uniquement ceux qui ont plus de 25 ans.

le **year** permet de tester en années. Le premier paramètre définit la date que l'on veut tester et le deuxième est ceux par rapport à quoi on va tester. On veut que ce résultat dépasse les 25 ans.

H- Changer la casse du texte

Mettre en majuscule de façon permanente :

```
update soldats
set nom = ucase(nom);
```

Permet de modifier les noms de familles pour les mettre en majuscule de façon permanente.

Le mot limit :

```
select * from soldats order by date_naissance limit 2;
```

Permet d'afficher uniquement les deux soldats les plus âgés.

Delete seulement une partie d'une table :

```
delete from soldats order by date_naissance limit 2;
```

Cette ligne permet de delete uniquement les deux plus vieux soldats.

Réaliser un update mais partiel (une augmentation)

Il est possible de réaliser un update que de certaines données dans une colonne :

```
update soldats
set grade = grade + 1 where grade <5;
-- Cette commande permet d'update le grade de tout le monde sauf ceux qui ont
un grade égal à 5.
```

Baisser le grade / chiffre donné d'une valeur choisie (ici le grade des hommes)

```
update soldats
set grade = grade - 1 where grade =3 and sexe="h";
Ici, on baisse de 1 le grade uniquement des hommes qui ont un grade = 3.
```

Ajout d'un jour à une date uniquement à une valeur donnée

```
update soldats
set date_naissance= date_add(date_naissance,interval 1 day) where sexe="f" and
grade =4;
Ici, on ajoute un jour à la date de naissance uniquement aux femmes qui ont un
grade = 4.
```

Arrondir à un certain nombre de chiffres après la virgule :

On utilisera la commande round :

```
select round(avg(satellite),2) from planete_antoine;
```

Le chiffre après la virgule dans la commande définit combien de chiffres on veut afficher après la virgule.

Créer une colonne qui affiche un chiffre en décimal :

Le **int** affiche des chiffres mais uniquement entiers. Pour les décimaux, on utilisera décimal :

```
nomdecolonne decimal(16.2) not null,
```

Le premier numéro va définir le nombre de caractères max affichés (virgule comprise), le deuxième le nombre de chiffres après la virgule. On utilise le **.** (point) au lieu de la virgule.

Requête dans une requête avec le mot in :

Il est possible d'imbriquer des requêtes pour avoir des résultats plus précis :

```
select planete from planete_antoine where satellite in (select
max(satellite)from planete_antoine);
```

Le résultat retourné après le **in** sera retourné dans la requête principale.

On peut imbriquer les requêtes sans le mot **in**. Les parenthèses sont le plus important. Le mot **in** est important quand on recherche un résultat non fixe.

Afficher un jour précis de manière simple :

```
select dayname("2018-12-23") as "veille du réveillon";
```

Le jour va s'afficher avec le jour en lettres. C'est la même chose que %w.

6- Les primary keys contenant plusieurs champs

Dans le langage SQL la “**PRIMARY KEY**”, autrement la clé primaire, permet d’identifier chaque enregistrement dans une table de base de données. Chaque enregistrement de cette clé primaire doit être UNIQUE et ne doit pas contenir de valeur NULL.

La clé primaire est un index, chacune des tables ne peut contenir qu'une seule clé primaire, composée d'une ou plusieurs colonnes.

Il faut créer les colonnes de manière classique sans préciser primary key, et créer la primary key à la fin.

```
create table planete_antoine(
    Planete VARCHAR(15) NOT NULL,
    Distance int NOT NULL,
    Rotation VARCHAR(5) NOT NULL,
    Revolution VARCHAR(30) NOT NULL,
    Satellite int NOT NULL,
    PRIMARY KEY(planete,distance,rotation)
);
```

A- Select plusieurs tables en même temps :

Il faut utiliser le mot **union** qui va permettre de lier les deux tables. Evidemment, il faut que les champs des deux tables soient identiques (mêmes colonnes, ect).

```
select * from table1
UNION
select * from table2;
```

B- Crée une seconde table qui a les mêmes propriétés que la première :

Si on veut créer une table identique à une table déjà créée, il existe un moyen rapide évitant de tout retaper et ainsi permettant d'éviter les erreurs humaines :

```
create table2 like table1;
```

C- Enregistrer une requête pour éviter de la retaper :

Il est possible de créer une requête qui fonctionne comme un alias afin d'éviter de retaper toujours la même ligne grâce à la commande **create view**.

Ce qui est créé est considéré comme une table virtuelle. C'est-à-dire qu'elle apparaît comme une table, on peut la manipuler comme une table mais ce n'est pas une table. Elle n'est qu'une table par sa formule. Elle n'existe pas réellement et est juste créée à la demande.

```
-- Ici, le as sert à introduire la requête pour dire cette view signifie la
requête entre parenthèses.
create view nomdeview as (select * from planete_antoine where satellite>20);
```

Cette formule à pas mal d'utilités, notamment limiter l'accès à uniquement certaines infos pré-triées.

D- Pour voir quelle formule a été utilisée pour créer une view :

```
show create view test_view;
```

La formule n'est pas très standard et l'affichage qui va apparaître est normal.

E- Afficher toutes les données d'une colonne sans les redondances :

Il existe une commande pour tout afficher d'une colonne en ne prenant pas en compte les doublons :

```
select DISTINCT numclient from commande;
-- Cette commande permet d'afficher toutes les données d'une colonne sans
afficher les doublons. C'est très pratique
-- Pour voir par exemple combien de clients ont commandé, en ne les affichant
qu'une seule fois s'ils ont effectué plusieurs commandes.
```

```
select count(distinct(numvendeur)) from commande having count(numcmde)>=1;
-- Ici, on vient compter les vendeurs qui ont fait au moins une commande. Il
faut donc les distinguer pour éviter les doublons.
```

F- Pour faire une recherche de quelque chose de null :

Dans MySQL, donnée=null ne fonctionne pas. Il faut mettre *donnee is null* :

```
select nom,prenom,numservice from filialea
where numservice is null
UNION
select nom,prenom,numservice from filialeb
where numservice is null;
-- Ici, le is null signifie que l'on va rechercher uniquement les lignes qui
ont la colonne numservice de null.
--Evidemment, on fait deux fois la requête parce qu'on utilise le union pour
lier deux tables qui ont des colonnes identiques.
```

G- Lier une table à plusieurs foreign keys :

```

select concat(substring(prenom,1,1),".",nom),nom_service,matricule
from
filialea join voitures
on filialea.numvoiture=voitures.idvoiture
join services
on filialea.numservice=services.ids
UNION
select concat(substring(prenom,1,1),".",nom),nom_service,matricule
from
filialeb join voitures
on filialeb.numvoiture=voitures.idvoiture
join services
on filialeb.numservice=services.ids
;
-- Ici, on utilise deux clefs étrangères. Pour ce faire, on va join la table à
la première qui a la foreign key
-- préciser ce qui les relie, et faire la même chose sur la deuxième table
possédant la foreign key. Union permet de relier à une autre table identique
-- à la première , donc on peut comparer les deux tables.

```

H- Faire une recherche grâce à une requête imbriquée :

```

select nom_service,count(numservice)
from
(select nom,prenom,nom_service,numservice
from
filialea join services
on filialea.numservice = services.ids
union
select nom,prenom,nom_service,numservice
from
filialeb join services
on filialeb.numservice = services.ids)
as totaux
group by numservice
;

-- Ici, l'idée est de compter les personnes qui appartiennent à un service
d'une entreprise.
-- On va donc faire deux colonnes, la première pour avoir le numéro de service
et la seconde va afficher le nombre de personnes qui y travaillent.
-- Le from englobe l'union des deux tableaux qui vont chercher les
informations requises grâce à leur foreign key. Leur select doit
nécessairement appeler les colonnes du select principal pour avoir le bon
résultat.
-- Le as est obligatoire. Pour un gros from entre parenthèses comme celui-ci,
SQL demande de lui donner un alias.
-- On trie par numéro de service pour afficher chaque service.

```

7- Les foreign keys :

Une **foreign key** sert à mettre deux tables en relation. Si on doit mettre les données d'une colonne en relation avec les données d'un autre tableau, il suffira de faire un **foreign key** qui fera référence à une colonne donnée d'un autre tableau.

```
-- Par exemple: table 1:
create table1(
    nom VARCHAR(15) NOT NULL,
    prenom varchar(15) NOT NULL,
    age VARCHAR(3) NOT NULL,
    services int not null,
    FOREIGN KEY (services) REFERENCES table2(services_nom)
);
-- La table 2:
create table2(
    services_nom int,
    compagny varchar(30) not null
);

-- Ici, la colonne services de la table 1 doit être la même que service ID de
la table 2 Elle peut être nulle.
-- Elle ne peut pas être différente.
Dans la table 1, la colonne services est liée à services_id de la colonne 2.
```

8- Les jointures

On peut faire des recherches en corrélant plusieurs tables pour avoir une recherche qui va afficher les résultats provenant des deux tables dans une seule :

```
select nom,prenom,nom_services
  FROM
table1 join table1
  on services = nom_services
;
-- Ici, nom et prenom proviennent de la table 1. nom_services provient de
la table2.
-- On joint les deux tables grâce à la foreign key où le numéro de service est
égal à l'id du nom_services.
```

Il est possible d'ajouter le mot **left** pour prendre en compte les résultats où il n'y a pas de réponse mais un champ nul ou pas de lien dans la deuxième table :

```
select nom,prenom,nom_services
  FROM
table1 left join table1
```

```
on services = nom_services
;
```

Il est possible d'utiliser le mot ***right***. Dans ce cas, ce sont les données de la première table qui sont nulles dans la deuxième.

A- Afficher toutes les données d'une colonne sans les redondances

Il existe une commande pour tout afficher d'une colonne en ne prenant pas en compte les doublons :

```
select DISTINCT numclient from commande;
-- Cette commande permet d'afficher toutes les données d'une colonne sans
afficher les doublons. C'est très pratique
-- Pour voir par exemple combien de clients ont commandé, en ne les affichant
qu'une seule fois s'ils ont effectué
-- plusieurs commandes.
```

```
select count(distinct(numvendeur)) from commande having count(numcmde)>=1;
-- Ici, on vient compter les vendeurs qui ont fait au moins une commande. Il
faut donc les distinguer pour
-- éviter les doublons.
```

B- Pour faire une recherche de quelque chose de null

Dans MySQL, donnée=null ne fonctionne pas. Il faut mettre *donnée is null* :

```
select nom,prenom,numservice from filialea
where numservice is null
UNION
select nom,prenom,numservice from filialeb
where numservice is null;
-- Ici, le is null signifie que l'on va rechercher uniquement les lignes qui
ont la colonne
-- numservice de null.
--Evidemment, on fait deux fois la requête parce qu'on utilise le union pour
lier deux tables qui ont des colonnes identiques.
```

C- Lier une table à plusieurs foreign keys :

```
select concat(substring(prenom,1,1),".",nom),nom_service,matricule
from
filialea join voitures
on filialea.numvoiture=voitures.idvoiture
join services
on filialea.numservice=services.idservic
UNION
select concat(substring(prenom,1,1),".",nom),nom_service,matricule
from
filialeb join voitures
```

```

on filialeb.numvoiture=voitures.idvoiture
join services
on filialeb.numservice=services.ids
ervice
;
-- Ici, on utilise deux clefs étrangères. Pour ce faire, on va join la table à
la première qui a la foreign key
-- préciser ce qui les relie, et faire la même chose sur la deuxième table
possédant la foreign key. Union permet de relier à une autre table identique
-- à la première , donc on peut comparer les deux tables.

```

D- Faire une recherche grâce à une requête imbriquée

```

select nom_service,count(numservice)
from
(select nom,prenom,nom_service,numservice
from
filialea join services
on filialea.numservice = services.ids
union
select nom,prenom,nom_service,numservice
from
filialeb join services
on filialeb.numservice = services.ids)
as totaux
group by numservice
;

-- Ici, l'idée est de compter les personnes qui appartiennent à un service
d'une entreprise.
-- On va donc faire deux colonnes, la première pour avoir le numéro de service
et la seconde
-- va afficher le nombre de personnes qui y travaillent.
-- Le from englobe l'union des deux tableaux qui vont chercher les
informations requises grâce
-- à leur foreign key. Leur select doit nécessairement appeler les colonnes du
select principal
-- pour avoir le bon résultat.
-- Le as est obligatoire. Pour un gros from entre parenthèses comme celui-ci,
SQL demande de lui
-- donner un alias.
-- On trie par numéro de service pour afficher chaque service.

```

E- Notes

- `Select * from table 1
union (permet de réunir les 2 tables)
select * from table 2;`
 - `select * from t1 Intersect select * from t2;`
 Langage SQL
 pour récupérer les m^{es} données des t1 et t2
 Δ ne fonctionne pas sur MySQL.
 - Table 1 en attente de paiement
 Table 2 payé
 ↳ montrer que les paiements en attente
`Select * from t1 minus select * from t2;` t1 moins t2
 ↑ minus
 Langage SQL
 Δ ne fonctionne pas sur MySQL
 - create table t1 { }
`create table t2 like t1;`
 permet de copier les données et la structure de t1 pour t2
- Retour ex01 ; tableau note;**
- nom, prenom, age (note 1, note 2) note 3)
 - sexe : "M" order by "age"
 - = `select nom, prenom, age, concat(note1, ", ", note2, ", ", note3)`
`as nnn from note where sexe = "M" order by age;`
 - ↳ en faisant `= create view list 01 as (select...);`
 ça crée une requête identique à un alias

Pour ne pas ~~réécrire~~
 {retaper} les requêtes à chaque fois;

= `create view nom as (select (requete));`

↳ `Select * from list01;` → ne seront visibles que le sexe masculin
`Select nom from list01;` ne seront visibles que les hommes(noms)

- `describe tb1;` → décrit juste la structure de la table
- `Show create view tb1;`
 décrit en profondeur la structure de la table

9- Tables fonctionnelles

MOCODonline

A- Syntaxe du texte d'entrée

Principe

Chaque ligne constitue la définition d'une entité ou d'une association (« boîte » dans la suite).

Définir une entité

`nom entité: attribut 1, attribut 2, attribut 3, ...`

Un nom d'entité est séparé de ses attributs par un deux-points.

B- Les attributs sont séparés par des virgules.

Le premier attribut est par défaut l'identifiant de l'entité.

Définir une association

`nom association, 01 nom entité 1, 1N nom entité 2, ... : attribut 1, attribut 2, ...`

Un nom d'association est séparé de sa première patte par une virgule.

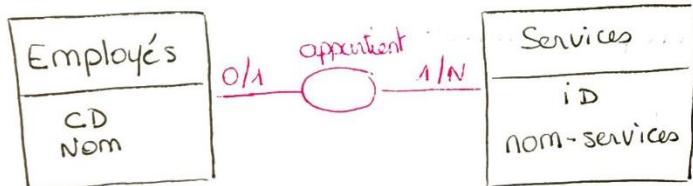
Les pattes d'une association sont séparées par des virgules.

Une patte consiste en un couple de cardinalités (01, 11, 0N, 1N) suivi du nom d'une entité.

La dernière patte d'une association est séparée de ses éventuels attributs par un deux-points.

Les attributs sont séparés par des virgules.

Deux pattes d'une même association peuvent aboutir à une même entité (association réflexive).

Tables fonctionnelles

- un employé appartient à au plus 1 seul service 0/1
- un service peut avoir plusieurs employés N

→ Create table personnes

id int primary key not null
auto-increment,
nom varchar(25) not null,
numservice int not null,

Foreign key (numservice) references service (id service); ①

create table services id int primary key not null auto-increment,
nom_services varchar(35) not null,

① (Foreign key (id-Parent) references parent(id))

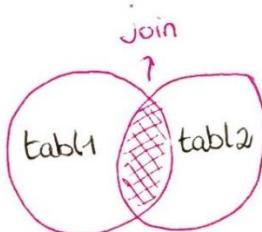
↳ clef étrangère pour éviter de modifier le tableau des données,

dont être équivalent à une valeur de la table
exemple = Service 1/2/3 ou null

ne peut pas être une autre valeur

Select nom, prenom, nom-service
from tabl1 join tabl2
permet de joindre/associer les 2 tables avec égalité
⇒ donnera le nom, le prénom et le nom du service

id numero service = id service



nom	prenom	nom service

left join → inclus toutes les données des 2 tableaux dans 1 tableau commun (de services)

right join → inclus toutes les données (ex: personnels)
inclus les services = null

Spécifier une mise en page

Les boîtes définies sur des lignes *consécutives* sont tracées sur une même rangée.

Un saut de ligne commence une nouvelle rangée.

Toutes les boîtes d'une même colonne sont alignées verticalement.

Une ligne réduite à un deux-points insère une boîte invisible.



Mocodo est un logiciel d'aide à l'enseignement et à la conception de bases de données relationnelles. En entrée, il prend une description textuelle des entités et associations du modèle conceptuel de données (MCD). En sortie, il produit son diagramme entité-association en SVG et son schéma relationnel (MLD) en SQL, LATEX, Markdown, etc.

Avec ce code :

```

<html>
<head>
<meta charset='utf-8'>
<style>
  
```

MySQL

```
#mld .relation { font-variant: small-caps; font-weight: bold }
#mld .primary { text-decoration: underline }
#mld .foreign { font-style: oblique }
#mld .normal { }

</style>
</head>
<body>
<div id='mld'>
<div>
  <span class='relation'>voitures</span> (
    <span class='primary'>idvoitures</span>, <!--relation primaire-->
    <span class='normal'>blabla</span>, <!--relation normale-->
    <span class='foreign'>idpersonnes</span> <!--relation étrangère-->
  )
</div>
<div>
  <span class='relation'>personnes</span> (
    <span class='primary'>idpersonnes</span>,
    <span class='normal'>blabla</span>
  )
</div>
<div>
  <span class='relation'>peut conduire</span> (
    <span class='foreign primary'>idvoitures</span>,
    <span class='foreign primary'>idpersonnes</span>,
    <span class='normal'>date début</span>,
    <span class='normal'>date fin</span>
  )
</div>
</div>
</body>
</html>
```

MySQL

10- EXERCICES SQL (sur base MySQL)

1- Exercice1

Données :

NUM	PRENOM	NOM	Sexe	PASSWORD	AGE	NOTE1	NOTE2	NOTE3
1	Marc	Dupont	M	df5sdfs	25	15	11	9
2	Paul	Marchand	M	sf5s	32	8	10	16
3	Marie	Laporte	F	klj8jkA5	15	13	17	11
4	Alex	Martin	M	frg16	23	11	12	16
5	Benoit	Dalors	M	12345	41	9	14	15
6	Yohann	Fergusson	M	Asde5	39	17	16	13
7	Phillipe	Dany	M	htyaz_8	19	19	18	12
8	Suzanne	Lacour	F	de4	43	4	11	16
9	Celine	Dermand	F	aqw85	27	11	16	13
10	Anne	Laclasse	F	ytrc9v6b	18	10	9	12
11	Charles	Nourri	M	i5f6z7q	30	7	11	6
12	Bernard	Cazanne	M	abc	52	9	15	9

Exercices :

- Ecrire une requête affichant 'Bonjour, bienvenue dans le monde de SQL'. Le nom de la colonne doit être Hello.
- Ecrire la requête qui permet de créer une base de données nommée DWWM-PRENOM (avec votre prénom).
- Ecrire la requête qui permet de créer une table 'Note' avec les données du tableau ci-dessus.
- Ecrire une requête affichant le nombre de stagiaires dans la table.
- Ecrire une requête affichant les données triées par nom.
- Ecrire une requête affichant les données triées par Age.
- Ecrire une requête affichant le nom et le prénom des stagiaires âgés de plus de 20 ans.
- Ecrire une requête affichant les prénoms et âges des stagiaires dont l'âge est entre 18 ans et 28 ans, triés par âge.
- Ecrire une requête affichant la liste des stagiaires dont le nom commence par 'D'.
- Ecrire une requête affichant la liste des stagiaires dont le prénom commence par un 'M', ou qui finit par un 'E'.
- Ecrire une requête affichant une liste de stagiaires contenant une seule colonne contenant le nom complet sous forme 'prénom NOM' et triée par âge
- Ecrire une requête affichant une liste de stagiaires contenant une seule colonne contenant le nom complet sous forme 'prénom N', où N est la première lettre du nom.

- 13.Ecrire une requête affichant une liste composée des initiales des stagiaires et de la note de l'examen N1 triée en ordre croissant
- 14.Ecrire une requête affichant une liste composée des initiales des stagiaires (A.A) et de la somme des notes sur 60 des examens triés en ordre décroissant.
- 15.Ecrire une requête affichant une liste de stagiaires composée d'une colonne 'Nom Prénom', d'une colonne avec la moyenne des 3 examens
- 16.Ecrire une requête affichant la moyenne de la classe.
- 17.Ecrire une requête affichant le nom et l'âge des stagiaires qui ont eu plus de 12 à l'examen N2 et dont le prénom contient la lettre 'N'.
- 18.Ecrire une requête affichant le nom des stagiaires qui ont eu moins de 10 à l'examen N3 et dont l'âge est inférieur à 25 ans ;
- 19.Ecrire une requête affichant une liste des noms des stagiaires dont le mot de passe a une longueur inférieure à 6. Le nom de cette colonne est 'Password Faible'
- 20.Ecrire une requête affichant une colonne contenant 'prénom(Age)' pour les stagiaires dont l'âge est soit 15, 25 ou 50 (Interdiction d'utiliser l'opérateur '=').
- 21.Ecrire une requête qui affiche la liste des stagiaires qui ont la moyenne dans toutes les matières.
- 22.Ecrire une requête qui liste les stagiaires qui ne sont pas dans la tranche 25-45 ans et qui n'ont pas la moyenne dans une des trois matières.
- 23.Ecrire une requête qui affiche la moyenne de tous les hommes sur l'examen 1 et la moyenne de toutes les femmes sur l'examen 1.
- 24.Ecrire une requête qui affiche la moyenne de tous les hommes sur l'examen 3 et la moyenne de toutes les femmes sur l'examen 3.
- 25.Ecrire une requête affichant la liste des moyennes par sexe pour les groupes ayant plus de 12 de moyenne à l'examen 3
- 26.Ecrire une requête affichant la date sous cette forme : '9999-99-99, 99 :99 :99'. Le nom de la colonne doit s'intituler « Maintenant ».
- 27.Ecrire une requête affichant le nombre de jours entre le 6 Novembre 1971 et aujourd'hui.
- 28.Ecrire une requête affichant la date qu'il sera dans 63 jours.
- 29.Ecrire une requête affichant la différence en jours entre le 1 avril 2019 et le 28 juin 2019.
- 30.Ecrire une requête affichant le nombre de jours depuis votre naissance.

MySQL

```
/* question 1 */
select "Bonjour, Bienvenue dans le monde SQL";
+-----+
| Bonjour, Bienvenue dans le monde SQL |
+-----+
| Bonjour, Bienvenue dans le monde SQL |
+-----+


select "Bonjour, Bienvenue dans le monde SQL" as HELLO;
+-----+
| HELLO           |
+-----+
| Bonjour, Bienvenue dans le monde SQL |
+-----+


/*question 2*/
> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| bdd1          |
| dwwm          |
| dwwm_annick   |
| essai          |
|                 |
| performance_schema |
| phpmyadmin     |
| sys            |
| test1          |
| test2          |
+-----+


/*question 3 -> erreur de nom j'ai mis tb1 au lieu de note*/
use dwwm_annick;
Database changed
> show tables;
+-----+
| Tables_in_dwwm_annick |
+-----+
| tb1                    |
+-----+


describe tb1;
+-----+-----+-----+-----+-----+-----+
| Field    | Type     | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
```

MySQL

NUM	<code>int(11)</code>	NO		<code>NULL</code>		
PRENOM	<code>varchar(64)</code>	YES		<code>NULL</code>		
NOM	<code>varchar(64)</code>	YES		<code>NULL</code>		
SEXE	<code>char(1)</code>	YES		<code>NULL</code>		
PASSWORD	<code>varchar(12)</code>	YES		<code>NULL</code>		
AGE	<code>int(11)</code>	NO		<code>NULL</code>		
NOTE1	<code>int(11)</code>	NO		<code>NULL</code>		
NOTE2	<code>int(11)</code>	NO		<code>NULL</code>		
NOTE3	<code>int(11)</code>	NO		<code>NULL</code>		

```
insert into tb1(NUM,PRENOM,NOM,SEXE,PASSWORD,AGE,NOTE1,NOTE2,NOTE3)
values
(1,'Marc','Dupont','M','df5sdfs',25,15,11,9),
(2,'Paul','Marchand','M','sf5s',32,8,10,16),
(3,'Marie','Laporte','F','klj8jkA5',15,16,17,11),
(4,'Alex','Martin','M','frg16',23,11,12,16),
(5,'Benoit','Dalors','M','12345',41,9,14,15),
(6,'Yohann','Fergusson','M','Asde5',39,17,16,13),
(7,'Philippe','Dany','M','htya_8',19,19,18,12),
(8,'Suzanne','Lacour','F','de4',43,4,11,16),
(9,'Celine','Dermand','F','aqw85',27,11,16,13),
(10,'Anne','Laclasse','F','ytrc9v6b',18,10,9,12),
(11,'Charles','Nourri','M','i5f6z7q',30,7,11,6),
(12,'Bernard','Cezanne','M','abc',52,9,15,9)
;
```

```
describe tb1;
```

Field	Type	Null	Key	Default	Extra
NUM	<code>int(11)</code>	NO		<code>NULL</code>	
PRENOM	<code>varchar(64)</code>	YES		<code>NULL</code>	
NOM	<code>varchar(64)</code>	YES		<code>NULL</code>	
SEXE	<code>char(1)</code>	YES		<code>NULL</code>	
PASSWORD	<code>varchar(12)</code>	YES		<code>NULL</code>	
AGE	<code>int(11)</code>	NO		<code>NULL</code>	
NOTE1	<code>int(11)</code>	NO		<code>NULL</code>	
NOTE2	<code>int(11)</code>	NO		<code>NULL</code>	
NOTE3	<code>int(11)</code>	NO		<code>NULL</code>	

```
select * from tb1;
```

NUM	PRENOM	NOM	SEXE	PASSWORD	AGE	NOTE1	NOTE2	NOTE3
1	Marc	Dupont	M	df5sdfs	25	15	11	9

MySQL

2	Paul	Marchand	M	sf5s	32	8	10	16
3	Marie	Laporte	F	klj8jkA5	15	16	17	11
4	Alex	Martin	M	frg16	23	11	12	16
5	Benoit	Dalors	M	12345	41	9	14	15
6	Yohann	Fergusson	M	Asde5	39	17	16	13
7	Philippe	Dany	M	htya_8	19	19	18	12
8	Suzanne	Lacour	F	de4	43	4	11	16
9	Celine	Dermand	F	aqw85	27	11	16	13
10	Anne	Laclasse	F	ytrc9v6b	18	10	9	12
11	Charles	Nourri	M	i5f6z7q	30	7	11	6
12	Bernard	Cezanne	M	abc	52	9	15	9


```
/*question 4*/
select count(*) from tb1;
```

+-----+
count(*)
+-----+
12
+-----+

1 row in set (0.00 sec)


```
/*question 5*/
select NOM from tb1;
```

+-----+
NOM
+-----+
Dupont
Marchand
Laporte
Martin
Dalors
Fergusson
Dany
Lacour
Dermand
Laclasse
Nourri
Cezanne
+-----+


```
/*question 6*/
select AGE from tb1;
```

+-----+
AGE
+-----+
25
32
15

MySQL

```

| 23 |
| 41 |
| 39 |
| 19 |
| 43 |
| 27 |
| 18 |
| 30 |
| 52 |
+-----+
/*question 7*/
select NOM,PRENOM from tb1 where age>20;
+-----+-----+
| NOM      | PRENOM   |
+-----+-----+
| Dupont   | Marc     |
| Marchand | Paul     |
| Martin    | Alex     |
| Dalors   | Benoit   |
| Fergusson | Yohann   |
| Lacour   | Suzanne  |
| Dermand   | Celine   |
| Nourri   | Charles  |
| Cezanne   | Bernard  |
+-----+-----+
/*question 8*/
select Nom,PRENOM from tb1 where age between 18 and 28;
+-----+-----+
| Nom      | PRENOM   |
+-----+-----+
| Dupont   | Marc     |
| Martin   | Alex     |
| Dany     | Philippe |
| Dermand   | Celine   |
| Laclasse | Anne     |
+-----+-----+
/*question 9*/
select * from tb1 where NOM like "D%";
```

NUM	PRENOM	NOM	SEXE	PASSWORD	AGE	NOTE1	NOTE2	NOTE3
1	Marc	Dupont	M	df5sdfs	25	15	11	9
5	Benoit	Dalors	M	12345	41	9	14	15
7	Philippe	Dany	M	htya_8	19	19	18	12
9	Celine	Dermand	F	aqw85	27	11	16	13

MySQL

```

/* question 10*/
select * from tb1 where prenom like "M%" or "%E" ;
+-----+-----+-----+-----+-----+-----+-----+-----+
| NUM | PRENOM | NOM      | SEXE | PASSWORD | AGE   | NOTE1  | NOTE2  | NOTE3  |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1   | Marc    | Dupont   | M    | df5sdfs  | 25   | 15     | 11     | 9      |
| 3   | Marie   | Laporte  | F    | klj8jkA5 | 15   | 16     | 17     | 11     |
+-----+-----+-----+-----+-----+-----+-----+-----+


/*question 11*/
select concat (PRENOM, ' ', NOM) as "nom complet" from tb1 where AGE;
+-----+
| nom complet |
+-----+
| Marc Dupont |
| Paul Marchand |
| Marie Laporte |
| Alex Martin |
| Benoit Dalors |
| Yohann Fergusson |
| Philippe Dany |
| Suzanne Lacour |
| Celine Dermand |
| Anne Laclasse |
| Charles Nourri |
| Bernard Cezanne |
+-----+


/*question 12*/
select concat (PRENOM, ' ',substring(NOM,1,1)) as "prenom n" from tb1;
+-----+
| prenom n |
+-----+
| Marc D   |
| Paul M   |
| Marie L   |
| Alex M   |
| Benoit D  |
| Yohann F  |
| Philippe D |
| Suzanne L  |
| Celine D  |
| Anne L   |
| Charles N  |
| Bernard C  |
+-----+


/*question 13*/

```

MySQL

```

select concat (substring(prenom,1,1), '.',substring(Nom,1,1), '-' ,note1) as
"examen n1" from tb1 order by note1;
+-----+
| examen n1 |
+-----+
| S.L-4      |
| C.N-7      |
| P.M-8      |
| B.D-9      |
| B.C-9      |
| A.L-10     |
| A.M-11     |
| C.D-11     |
| M.D-15     |
| M.L-16     |
| Y.F-17     |
| P.D-19     |
+-----+

/*question 14*/
select concat (substring(prenom,1,1), '.',substring(Nom,1,1), '-',
',note1+note2+note3) as "examen n1" from tb1 order by note1+note2+note3 desc;
+-----+
| examen n1 |
+-----+
| P.D-49    |
| Y.F-46    |
| M.L-44    |
| C.D-40    |
| A.M-39    |
| B.D-38    |
| M.D-35    |
| P.M-34    |
| B.C-33    |
| S.L-31    |
| A.L-31    |
| C.N-24    |
+-----+

/*question 15*/
select concat(substring(PRENOM,1,1),"." ,substring(NOM,1,1)) as
initiale,concat(NOTE1+NOTE2+NOTE3,"/60") from tb1 order by NOTE1 asc;
+-----+-----+
| initiale | concat(NOTE1+NOTE2+NOTE3,"/60") |
+-----+-----+
| S.L      | 31/60          |
| C.N      | 24/60          |
| P.M      | 34/60          |
| B.D      | 38/60          |
+-----+

```

MySQL

```
| B.C      | 33/60          |
| A.L      | 31/60          |
| A.M      | 39/60          |
| C.D      | 40/60          |
| M.D      | 35/60          |
| M.L      | 44/60          |
| Y.F      | 46/60          |
| P.D      | 49/60          |
+-----+-----+-----+  
  
/*question 16*/
select count(note1+note2+note3) as "moyenne classe" from tb1;
+-----+
| moyenne classe |
+-----+
|           12   |
+-----+  
  
/*question 17*/
select nom,age from tb1 where note2>12 and prenom like "%N%";
+-----+-----+
| nom      | age   |
+-----+-----+
| Dalors   | 41    |
| Fergusson | 39    |
| Dermand   | 27    |
| Cezanne   | 52    |
+-----+-----+  
  
/*question 18*/
select NOM from tb1 where NOTE3<10 and AGE<25;
Empty set (0.00 sec)  
  
/*question 19*/
select NOM as "Password faible" from tb1 where length(PASSWORD)<6;
+-----+
| Password faible |
+-----+
| Marchand        |
| Martin          |
| Dalors          |
| Fergusson       |
| Lacour          |
| Dermand         |
| Cezanne         |
+-----+  
  
/*question 20*/
```

MySQL

```
select concat(PRENOM,"(",AGE,")") from tb1 where AGE like 15 OR AGE like 25 OR AGE like 50;
+-----+
| concat(PRENOM,"(",AGE,")") |
+-----+
| Marc(25)                  |
| Marie(15)                 |
+-----+



/*question 21*/
select concat(PRENOM, " ",NOM) from tb1 where NOTE1>10 and NOTE2>10 and NOTE3>10;
+-----+
| concat(PRENOM, " ",NOM) |
+-----+
| Marie Laporte           |
| Alex Martin              |
| Yohann Fergusson         |
| Philippe Dany            |
| Celine Dermand           |
+-----+



/*question 22*/
select concat(PRENOM, " ",NOM) from tb1 where (AGE>45 or AGE<25) and (NOTE1<10 or NOTE2<10 or NOTE3<10);
+-----+
| concat(PRENOM, " ",NOM) |
+-----+
| Anne Laclasse           |
| Bernard Cezanne          |
+-----+



/* question 23*/
select avg(NOTE1) from tb1 group by Sexe desc;
+-----+
| avg(NOTE1) |
+-----+
| 11.8750   |
| 10.2500   |
+-----+



/* question 24*/
select avg(NOTE3) from tb1 group by Sexe desc;
+-----+
| avg(NOTE3) |
+-----+
| 12.0000   |
| 13.0000   |
+-----+
```

```

/*question 25*/
select avg((NOTE3+NOTE2+NOTE1)/3) from tb1 where NOTE3>12 group by Sexe desc;
+-----+
| avg((NOTE3+NOTE2+NOTE1)/3) |
+-----+
| 13.08332500 |
| 11.83330000 |
+-----+

/*question 26*/
select date_format(now(),"%Y-%M-%D, %H :%I :%S") as Maintenant;
+-----+
| Maintenant           |
+-----+
| 2019-February-6th, 14 :02 :11 |
+-----+

/*question 27*/
select datediff(now(),"1971-11-06");
+-----+
| datediff(now(),"1971-11-06") |
+-----+
| 17259 |
+-----+

/*question 28*/
select adddate(now(),63);
+-----+
| adddate(now(),63)   |
+-----+
| 2019-04-10 14:26:10 |
+-----+

/*question 29*/
select datediff("2019-04-1","2019-06-28");
+-----+
| datediff("2019-04-1","2019-06-28") |
+-----+
| -88 |
+-----+

/*question 30*/
select datediff("1997-1-3",now());
+-----+
| datediff("1997-1-3",now()) |
+-----+
| -8069 |
+-----+

```

11- EXERCICES SQL (sur base MySQL)

2- Exercice 2

Exercices SQL - série 2

1. Créer une base de données Dwwm2-prenom.
2. Créer une table soldats (nom, prenom, date_naissance, sexe, grade)
{grade : valeur de 1 à 5}
3. Insérer dans la table 16 personnes – (toutes les données seront en minuscules)
Essayer de panacher les sexes, les âges et les grades.
4. Créer une requête qui liste les soldats de sexe masculin
5. Créer une liste des soldats qui ont plus de 25 ans
6. Créer une requête qui liste le nom de tous les soldats en majuscule
7. Créer une requête qui transforme les noms des soldats en majuscule de façon permanente
8. Créer une requête qui efface les deux soldats les plus âgés
9. Créer une requête qui insère quatre nouveaux soldats
10. Créer une requête qui met le grade de tout le monde au niveau +1 sauf pour les grades 5
11. Créer une requête qui baisse le grade (-1) de tous les soldats masculins qui ont un grade de 3
12. Créer une requête qui ajuste la date de naissance à j+1 pour toutes les femmes soldats qui ont un grade 4

```
/* Question 1 */

create database Dwwm2_remy;
use Dwwm2_remy;
Database changed
```

```
/* Question 2 */

create table soldats(
    nom varchar (48),
    prenom varchar(48),
    date_naissance date,
    sexe char(1),
    grade int(5)
);
```

MySQL

```

/*Question 3 */
insert into soldats (nom,prenom,date_naissance,sexe,grade)
    value
        ("Aimone","Trentino",'1965-01-25',"F",1),
        ("Prudenzio","Esposito",'1941-06-20',"M",2),
        ("Berengario","Zito",'1946-12-11',"M",3),
        ("Maurilio","Capon",'1969-04-24',"M",4),
        ("Donata","Fallaci",'1974-08-17',"F",5),
        ("Domenica","Fiorentino",'1972-05-4',"F",1),

        ("Gioacchino","Marino",'1971-02-27',"M",2),
        ("Facondo","Mancini",'1983-03-24',"M",3),
        ("Pasqualina","Padovano",'1965-07-07',"F",4),
        ("Rolando","Giordano",'1999-08-19',"M",5),
        ("Ulisse","Mancini",'1997-07-07',"F",1),
        ("Amaranto","Genovese",'1974-12-30',"M",2),
        ("Raimondo","Esposito",'1956-12-19',"M",3),
        ("Ivone","Longo",'1971-07-31',"F",4),
        ("Chiara","Arcuri",'2002-05-01',"F",5),
        ("Cupido","Zito",'1984-09-24',"M",5)
    ;
Query OK, 16 rows affected (0.03 sec)
Records: 16  Duplicates: 0  Warnings: 0

```

/ Question 4 */*

```

select * from soldats where sexe like 'M%';
+-----+-----+-----+-----+-----+
| nom      | prenom   | date_naissance | sexe | grade |
+-----+-----+-----+-----+-----+
| Prudenzio | Esposito | 1941-06-20     | M    | 2    |
| Berengario | Zito     | 1946-12-11     | M    | 3    |
| Maurilio  | Capon    | 1969-04-24     | M    | 4    |
| Gioacchino | Marino   | 1971-02-27     | M    | 2    |
| Facondo    | Mancini  | 1983-03-24     | M    | 3    |
| Rolando    | Giordano | 1999-08-19     | M    | 5    |
| Amaranto   | Genovese | 1974-12-30     | M    | 2    |
| Raimondo   | Esposito | 1956-12-19     | M    | 3    |
| Cupido     | Zito     | 1984-09-24     | M    | 5    |
+-----+-----+-----+-----+-----+

```

/ Question 5 */*

```

select * from soldats where date_naissance <= All(select date_naissance from
soldats);
+-----+-----+-----+-----+-----+
| nom      | prenom   | date_naissance | sexe | grade |
+-----+-----+-----+-----+-----+

```

MySQL

```

| Prudenzio | Esposito | 1941-06-20      | M      |      2 |
+-----+-----+-----+-----+-----+
/* Question 6 */
select nom from soldats where upper (nom);

/* Question 7*/
select UPPER(nom) as UpperNom from soldats;

/* Question 8 */
select * from soldats order by date_naissance limit 2;

/* Affiche la liste des 2 soldats les plus agées */
+-----+-----+-----+-----+-----+
| nom      | prenom   | date_naissance | sexe | grade |
+-----+-----+-----+-----+-----+
| Prudenzio | Esposito | 1941-06-20      | M    |      2 |
| Berengario | Zito     | 1946-12-11      | M    |      3 |
+-----+-----+-----+-----+-----+

/* Supprime les 2 soldats les plus agées */
delete from soldats order by date_naissance limit 2;
Query OK, 2 rows affected (0.05 sec)

/* Verification */
select * from soldats ;
-> ;
+-----+-----+-----+-----+-----+
| nom      | prenom   | date_naissance | sexe | grade |
+-----+-----+-----+-----+-----+
| Aimone   | Trentino | 1965-01-25      | F    |      1 |
| Maurilio | Capon    | 1969-04-24      | M    |      4 |
| Donata   | Fallaci  | 1974-08-17      | F    |      5 |
| Domenica | Fiorentino | 1972-05-04      | F    |      1 |
| Gioacchino | Marino  | 1971-02-27      | M    |      2 |
| Facondo   | Mancini  | 1983-03-24      | M    |      3 |
| Pasqualina | Padovano | 1965-07-07      | F    |      4 |
| Rolando   | Giordano | 1999-08-19      | M    |      5 |
| Ulisse    | Mancini  | 1997-07-07      | F    |      1 |
| Amaranto  | Genovese | 1974-12-30      | M    |      2 |
| Raimondo  | Esposito | 1956-12-19      | M    |      3 |
| Ivone     | Longo    | 1971-07-31      | F    |      4 |
| Chiara    | Arcuri   | 2002-05-01      | F    |      5 |
| Cupido    | Zito     | 1984-09-24      | M    |      5 |
+-----+-----+-----+-----+-----+
14 rows in set (0.00 sec)

```

```

/* Question 9 */

/* Rajoutez 4 soldats */

insert into soldats (nom,prenom,date_naissance,sexe,grade)
    value
    ("Fosca","Calabrese",'2006-09-10','F',1),
    ("Casa","Papel",'2000-01-02','M',2),
    ("Antonio","Vega",'2010-05-22','M',3),
    ("Edmonda","Piazza",'2007-07-30','F',4);

/* Question 10 */

/* Rajoute +1 a la valeur de la colonnes Grade (sauf pour les grades 5) */
update soldats set grade = grade + 1 where grade <5;
Query OK, 14 rows affected (0.03 sec)
Rows matched: 14  Changed: 14  Warnings: 0

/* Vérification */
select * from soldats;
+-----+-----+-----+-----+-----+
| nom      | prenom     | date_naissance | sexe | grade |
+-----+-----+-----+-----+-----+
| Aimone   | Trentino   | 1965-01-25    | F    | 2    |
| Maurilio | Capon      | 1969-04-24    | M    | 5    |
| Donata   | Fallaci    | 1974-08-17    | F    | 5    |
| Domenica | Fiorentino | 1972-05-04    | F    | 2    |
| Gioacchino | Marino     | 1971-02-27    | M    | 3    |
| Facondo  | Mancini    | 1983-03-24    | M    | 4    |
| Pasqualina | Padovano   | 1965-07-07    | F    | 5    |
| Rolando  | Giordano   | 1999-08-19    | M    | 5    |
| Ulisse   | Mancini    | 1997-07-07    | F    | 2    |
| Amaranto | Genovese   | 1974-12-30    | M    | 3    |
| Raimondo | Esposito  | 1956-12-19    | M    | 4    |
| Ivone    | Longo      | 1971-07-31    | F    | 5    |
| Chiara   | Arcuri     | 2002-05-01    | F    | 5    |
| Cupido   | Zito       | 1984-09-24    | M    | 5    |
| Fosca    | Calabrese  | 2006-09-10    | F    | 2    |
| Casa     | Papel      | 2000-01-02    | M    | 3    |
| Antonio  | Vega       | 2010-05-22    | M    | 4    |
| Edmonda | Piazza    | 2007-07-30    | F    | 5    |
+-----+-----+-----+-----+-----+
18 rows in set (0.00 sec)

/* Question 11 */

-- Baisse le grade (-1) de tous les soldats masculins qui ont un grade de 3
update soldats set grade = grade -1 where sexe='M' and grade=3;
Query OK, 3 rows affected (0.01 sec)

```

MySQL

```
Rows matched: 3  Changed: 3  Warnings: 0
```

```
mysql> select * from soldats;
```

nom	prenom	date_naissance	sexe	grade
Aimone	Trentino	1965-01-25	F	2
Maurilio	Capon	1969-04-24	M	5
Donata	Fallaci	1974-08-17	F	5
Domenica	Fiorentino	1972-05-04	F	2
Gioacchino	Marino	1971-02-27	M	2
Facondo	Mancini	1983-03-24	M	4
Pasqualina	Padovano	1965-07-07	F	5
Rolando	Giordano	1999-08-19	M	5
Ulisse	Mancini	1997-07-07	F	2
Amaranto	Genovese	1974-12-30	M	2
Raimondo	Esposito	1956-12-19	M	4
Ivone	Longo	1971-07-31	F	5
Chiara	Arcuri	2002-05-01	F	5
Cupido	Zito	1984-09-24	M	5
Fosca	Calabrese	2006-09-10	F	2
Casa	Papel	2000-01-02	M	2
Antonio	Vega	2010-05-22	M	4
Edmonda	Piazza	2007-07-30	F	5

```
18 rows in set (0.00 sec)
```

```
/* Question 12 */
```

```
/* Ajoute +1 au jour de naissance a toutes les femmes de grade 4 (Aucun cas
présent ici mais la requête est validée)*/
```

```
update soldats set date_naissance = adddate(date_naissance, 1) where grade=4
and sexe ="F";
```

12- EXERCICES SQL (sur base MySQL)

3- Exercice 3

Exercices SQL - série 3

numcmd	numclient	numvendeur	montant	datecmd
25	101	51	112,2	05/10/2018
26	107	52	75,25	11/02/2019
27	105	53	63,25	12/01/2019
31	101	52	63,25	12/02/2019
45	111	52	45	11/01/2019
46	122	52	40	25/01/2019
47	117	55	80	30/01/2019
49	121	57	81,25	07/11/2018
50	123	51	50	06/11/2018
52	105	62	25	11/02/2019
53	112	66	25,39	02/02/2019
55	107	79	31	11/02/2019
59	119	62	9,8	09/02/2019
61	124	51	70	15/01/2019
62	123	54	63	31/01/2019

numvendeur	nomvendeur
51	etienne
52	mathieu
53	stéphane
54	tony
55	david
57	suzette
62	romain
66	sylvie
79	gilles

- 1/ créer une requête qui affiche le montant total de toutes les commandes.
- 2/ créer une requête qui affiche la moyenne de toutes les commandes.
- 3/ créer une requête qui affiche le nombre de commandes.
- 4/ créer une requête qui affiche le nombre de clients qui ont passé au moins une commande.
- 5/ quel est le nombre de vendeurs qui ont réalisé une ou plusieurs commandes.
- 6/ créer une requête qui affiche la liste des clients qui ont passé plus de 2 commandes
- 7/ créer une requête qui affiche la liste les vendeurs qui ont fait plus de 65 € de commandes.
- 8/ créer une requête qui affiche la plus grande vente par vendeur.

9/ créer une requête qui affiche la plus grande commande pour les clients dont le numéro varie entre 105 et 120.

10/ créer une requête qui affiche le montant des commandes passées le 11 février 2019.

11/ créer une requête qui liste un numéro de commande et le nom du vendeur qui l'a réalisée.

12/ créer une requête qui affiche le nom d'un vendeur et la date de sa dernière commande.

13/ créer une requête qui liste la moyenne de vente par vendeur.

Exercice page 1:

```
show databases;
+-----+
| Database      |
+-----+
| information_schema |
| bdd1          |
| dwwm          |
| dwwm2_annick   |
| dwwm_annick    |
| essai          |
| mysql          |
| performance_schema |
| phpmyadmin     |
| sys            |
| test1          |
| test2          |
+-----+

create database exo_3;
mysql> use exo_3;
Database changed
show tables;
create table commandes(
    -> numcmd int not null,
    -> numclient int not null,
    -> numvendeur int not null,
    -> montant decimal(16,2) not null,
    -> datecmd date
    -> );

show tables;
+-----+
| Tables_in_exo_3 |
+-----+
| commandes      |
+-----+
```

MySQL

```

describe commandes;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| numcmdde   | int(11)    | NO   |     | NULL    |       |
| numclient  | int(11)    | NO   |     | NULL    |       |
| numvendeur | int(11)    | NO   |     | NULL    |       |
| montant    | decimal(16,2)| NO  |     | NULL    |       |
| datecmd    | date       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+

insert into commandes (numcmdde,numclient,numvendeur,montant,datecmd)
-> values
-> (25,101,51,112.2,'2018-10-05'),
-> (26,107,52,75.25,'2019-02-11'),
-> (27,105,53,63.25,'2019-01-12'),
-> (31,101,52,63.25,'2019-02-12'),
-> (45,111,52,45,'2019-01-11'),
-> (46,122,52,40,'2019-01-25'),
-> (47,117,55,80,'2019-01-30'),
-> (49,121,57,81.25,'2018-11-07'),
-> (50,123,51,50,'2018-11-06'),
-> (52,105,62,25,'2019-02-11'),
-> (53,112,66,25.39,'2019-02-02'),
-> (55,107,79,31,'2019-02-11'),
-> (59,119,62,9.8,'2019-02-09'),
-> (61,124,51,70,'2019-01-15'),
-> (62,123,54,63,'2019-01-31');

select * from commandes;
+-----+-----+-----+-----+-----+
| numcmdde | numclient | numvendeur | montant | datecmd   |
+-----+-----+-----+-----+-----+
|      25 |        101 |         51 |   112.20 | 2018-10-05 |
|      26 |        107 |         52 |    75.25 | 2019-02-11 |
|      27 |        105 |         53 |    63.25 | 2019-01-12 |
|      31 |        101 |         52 |    63.25 | 2019-02-12 |
|      45 |        111 |         52 |    45.00 | 2019-01-11 |
|      46 |        122 |         52 |    40.00 | 2019-01-25 |
|      47 |        117 |         55 |    80.00 | 2019-01-30 |
|      49 |        121 |         57 |    81.25 | 2018-11-07 |
|      50 |        123 |         51 |    50.00 | 2018-11-06 |
|      52 |        105 |         62 |    25.00 | 2019-02-11 |
|      53 |        112 |         66 |   25.39 | 2019-02-02 |
|      55 |        107 |         79 |   31.00 | 2019-02-11 |
|      59 |        119 |         62 |    9.80 | 2019-02-09 |
|      61 |        124 |         51 |   70.00 | 2019-01-15 |
|      62 |        123 |         54 |   63.00 | 2019-01-31 |
+-----+-----+-----+-----+-----+

```

```

show tables;
+-----+
| Tables_in_exo_3 |
+-----+
| commandes      |
| vendeurs       |
+-----+

insert into vendeurs (numvendeur,nomvendeur)
    -> values
    -> (51,'étienne'),
    -> (52,'mathieu'),
    -> (53,'stéphane'),
    -> (54,'tony'),
    -> (55,'david'),
    -> (57,'suzette'),
    -> (62,'romain'),
    -> (66,'sylvie'),
    -> (79,'gilles');
select * from vendeurs;
+-----+
| numvendeur | nomvendeur |
+-----+
|      51 | étienne   |
|      52 | mathieu   |
|      53 | stéphane |
|      54 | tony       |
|      55 | david     |
|      57 | suzette   |
|      62 | romain   |
|      66 | sylvie   |
|      79 | gilles   |
+-----+

/*question 1*/
select sum(montant) from commandes;
+-----+
| sum(montant) |
+-----+
|    834.39 |
+-----+

/*question 2*/
select avg(montant)from commandes;
+-----+
| avg(montant) |
+-----+
|  55.626000 |
+-----+

```

MySQL

```
/*question 3*/
select count(numcmde) from commandes;
+-----+
| count(numcmde) |
+-----+
|          15   |
+-----+

/*question 4*/
select count(numcmde>1) from commandes ;
+-----+
| count(numcmde>1) |
+-----+
|          15   |
+-----+
```

MySQL

Personnes

Filiale A

idpersonnes	nom	prenom	email	numService	numVoiture
1	Pauland	Marc	p.marc@nimps.com	1	1
2	Taralou	Dimitri	z.tar.dim@zolka.com	3	1
3	Marchand	Aline	m.aline@duvent.fr	2	2
4	Zolak	Pierre	z.pierre@purple.com	1	1
5	Laporte	martine	l.martine@free.fr	NULL	NULL

Filiale B

idpersonnes	nom	prenom	email	numService	numVoiture
1	Marliax	Jean	m.jean@nloips.com	3	1
2	Zappy	Claudine	z.c.200@vayent.fr	1	2
3	Taralou	Dimitri	z.tar.dim@zolka.com	3	1
4	Zolak	Pierre	z.pierre@purple.com	1	1
5	Fenestry	Etienne	Fen_eti_fr@free.fr	NULL	1

Services

idservice	nom_service
1	compta
2	vente
3	support

Voitures

idvoitures	marque	matricule
1	Renault	951 KIO 45
2	Fiat	741 MIP 65

- 1/ créer une requête d'une colonne de la forme 'Prenom_NOM' triée par prénom pour la filiale A.
- 2/ créer une requête d'une colonne de la forme 'Prenom_NOM' triée par prénom pour la filiale B.
- 3/ créer une liste qui regroupe le nom et le prenom de toutes les personnes de toutes les filiales.
- 4/ créer une requête qui liste les personnes qui n'ont pas de service affecté.
- 5/ créer une requête qui liste les personnes de la filiale A qui ne travaillent pas pour la filiale B.
- 6/ créer une requête qui liste les personnes de la filiale A ou la filiale B mais pas pour les deux filiales.
- 7/ créer une requête qui liste les personnes des deux filiales : nom, prenom, nom du service.
- 8/ créer une requête qui liste : 'prenom.N', matricule de la voiture.
- 9/ créer une requête qui liste 'P.nom', nom du service, matricule de la voiture.
- 10/ créer une requête qui affiche le nombre de personnes affectés à chaque voiture.
- 11/ créer une requête qui affiche le nombre de personnes affectés à un service.

MySQL

Un Journaliste sportif souhaite établir des statistiques sur les blessures des joueurs de la ligue de Rugby.

Pour chaque joueur, en plus de l'état-civil, il souhaite avoir la trace :

- des blessures reçues (type et date de la blessure, en plus du match où elle a été infligée. Les matchs seront référencés dans une liste comportant le lieu, les dates).
- des postes occupé (avec les dates).
- de l'équipe de rattachement (avec les dates) durant sa carrière.

Etablir le MCD et le MLD.

Une personne souhaite créer un site consacré à ses auteurs préférés. Il doit donc tenir l'inventaire de ses livres, avec pour chacun d'eux le titre, l'auteur, l'éditeur et l'année de parution.

On traitera successivement de deux hypothèses :

la bibliothèque ne comprend aucune livre de différents auteurs.

la bibliothèque comprend des livres co-écrits par plusieurs auteurs.

Ecrire le MCD et le MLD dans ces deux cas.

.....

Correction à venir