

Quelques commandes SQL pour MySQL

Notre base de données dans l'exemple se nommera **VENTES** avec comme tables **CLIENTS** (Code_Cli, Nom_Cli, Adr_Cli, Ville_Cli, Tel_Cli), **PRODUITS** (Code_Pts, Nom_Pts, Quanti_Pts, Prix_Pts) et **COMMANDES** (Code_Com, Code_Pts, Code_Cli, Quanti_Com, Date_Com) :

1) Création d'une Base de Données :

```
CREATE DATABASE `ventes`
```

2) Suppression d'une Base de Données :

```
DROP DATABASE `ventes`
```

3) Création d'une Table :

```
CREATE TABLE `produits` (`Code_Pts` BIGINT NOT NULL auto_increment,  
`Nom_Pts` VARCHAR(100) NOT NULL, `Quanti` INT(6) NOT NULL, `Prix`  
DECIMAL(5,2) NOT NULL unsigned, PRIMARY KEY (`Code_Pts`))
```

4) Vider une Table :

```
TRUNCATE TABLE `clients`
```

5) Modifications d'une Table :

a) Renommée une table :

```
ALTER TABLE `produits` RENAME `Liste_Produits`
```

b) Ajout d'un index :

```
ALTER TABLE `produits` ADD INDEX (`Nom_Pts`)
```

c) Suppression d'un index :

```
ALTER TABLE `produits` DROP INDEX (`Nom_Pts`)
```

d) Ajout d'un élément pour la recherche en FULLTEXT :

```
ALTER TABLE `produits` ADD FULLTEXT (`Nom_Pts`)
```

e) Activé ou désactivé une clé :

```
ALTER TABLE `produits` DISABLE KEYS
```

```
ALTER TABLE `produits` ENABLE KEYS
```

f) Ajout d'un champ :

```
ALTER TABLE `produits` ADD `Description_Pts` VARCHAR(200) NOT NULL
```

g) Suppression d'un champ :

```
ALTER TABLE `produits` DROP `Description_Pts`
```

h) Modification d'un champ :

```
ALTER TABLE `produits` CHANGE `Quanti` `Quanti` INT(8) NOT NULL
```

i) Modification du nom d'un ou plusieurs champs :

```
SELECT `Quanti` AS `Quanti_Pts`, `Prix` AS `Prix_Pts` FROM `produits`
```

6) Utilisations d'une Table :

a) Afficher tout les Enregistrements suivant un ordre ascendant :

```
SELECT * FROM `clients` ORDER BY `Code_Cli` ASC
```

b) Afficher une liste de 30 Enregistrements à partir de la ligne 10 suivant un ordre descendant :

```
SELECT * FROM `produits` LIMIT 10, 30 ORDER BY `Code_Pts` DESC
```

c) Afficher une liste d'Enregistrement Multi-Tables regroupés par numéro client :

```
SELECT `commandes.Code_Com`, `clients.Code_Cli`, `produits.Code_Pts`,  
`produits.Prix_Pts` FROM `produits`, `commandes`, `clients` GROUP BY  
`Code_Cli`
```

d) Ajout d'un ou de plusieurs Enregistrements :

```
INSERT INTO `clients` (`Code_Cli`, `Nom_Cli`, `Adr_Cli`, `Ville_Cli`, `Pays_Cli`,  
`Tel_Cli`) VALUES ('1', 'nom client1', 'rue des fleurs', 'Montréal', 'Canada', '012-  
345-6789'), ('2', 'nom client2', 'avenue de paris', 'Nice', 'France', '01-23-45-67-  
89')
```

e) Suppression d'un ou de plusieurs Enregistrements :

```
DELETE FROM `clients` WHERE `Nom_Cli`='nom client3'  
DELETE FROM `clients` WHERE `Ville_Cli` LIKE 'Paris'
```

f) Modification d'un ou de plusieurs Enregistrements :

```
UPDATE [Low Priority] [Ignore] `nom de table` SET `Champ1`='infos1',  
`Champ2`='infos2', `Champ3`='infos3' WHERE `Id`=40
```

```
UPDATE `clients` SET `Adr_Cli`='225 avenue de Paris', `Ville_Cli`='Marseille'  
WHERE `Code_Cli`=2 AND `Nom_Cli`='nom client2'
```

```
UPDATE `clients` SET `Adr_Cli`='225 avenue de Paris', `Ville_Cli`='Marseille'  
WHERE `Code_Cli`=2 OR `Nom_Cli`='nom client2'
```

```
UPDATE `clients` SET `Ville_Cli`='Paris' WHERE `Adr_Cli` != 'rue' LIMIT 5 <==  
(c'est le nombre maximal d'enregistrement qui seront affectés par la mise à jour)
```

7) Traitements de données :

a) Having : affichage conditionnel de données lors d'un affichage par groupement

```
SELECT `Code_Cli` FROM `commandes`, `clients` ORDER BY `clients.Ville_Cli`  
ASC HAVING SUM(`Quanti_Com`) > 250
```

b) Between : affichage de données en fonction d'un intervalle

```
SELECT `Nom_Pts` FROM `produits` WHERE `Code_Pts` BETWEEN 20 AND  
50
```

c) Concaténation de données ou de plusieurs champs d'un Enregistrement :

```
SELECT CONCAT('Produit n° ', `Code_Pts`) FROM `produits`
```

- d) Length() : Longueur d'une chaîne de caractères
`SELECT LENGTH(`Nom_Pts`) FROM `produits``
- e) Left(chaîne,n) : Les n premiers caractères d'une chaîne de caractères
`SELECT LEFT(`Nom_Pts`, 6) FROM `produits``
- f) Substring(chaîne,p,n) : Les N caractères d'une chaîne de caractères selon une position
`SELECT SUBSTRING(`Nom_Pts`,3,7) FROM `produits``
- 8) Les Fonctions de calcul :
- a) Count() : Le nombre total des clients
`SELECT COUNT(`Code_Cli`) FROM `clients``
- b) Sum() : Quantité totale des produits en stock
`SELECT SUM(`Quanti_Pts`) FROM `produits``
- c) Min() : Le produit qui a le stock le moins élevé parmi tous les produits
`SELECT MIN(`Quanti_Pts`) FROM `produits``
- d) Max() : Le produit qui a le stock le plus élevé parmi tous les produits
`SELECT MAX(`Quanti_Pts`) FROM `produits``
- e) Avg() : Quantité moyenne des produits en stock
`SELECT AVG(`Quanti_Pts`) FROM `produits``
- 9) Voir les informations sur la base de données et la Table :
- a) Voir le statut d'une base de données :
`SHOW DATABASE `ventes``
- b) Voir le statut d'une table :
`SHOW TABLE FROM `clients``
- c) Voir la structure d'une table :
`DESCRIBE `produits``
- d) Voir les indexes d'une table :
`SHOW INDEX FROM `clients``
- e) Voir le statut d'une table :
`SHOW TABLE STATUS FROM `clients``
- 10) La recherche en FULLTEXT :
- a) Indexation FULLTEXT :
`ALTER TABLE `clients` ADD `Description_Cli` LONGTEXT NOT NULL`

`ALTER TABLE `clients` ADD FULLTEXT(`Description_Cli`)`
- b) La recherche simple :

```
SELECT * FROM `clients` WHERE MATCH (`Nom_Cli`, `Adr_Cli`, `Ville_Cli`,  
`Pays_Cli`, `Tel_Cli`, `Description_Cli`) AGAINST ('Chaîne de caractères')
```

Attention :

- les mots de moins de 3 lettres sont ignorés.
- plus le apparaît peu, plus il a du poids d'importance.

c) La recherche booléennes :

+

-

"" : phrase ou mot exact

*

~ : diminue l'importance du mot au niveau du symbole

> : augmente l'importance du mot

< : diminue l'importance du mot

```
SELECT `Nom_Cli`, `Adr_Cli`, `Ville_Cli`, `Pays_Cli`, `Tel_Cli` FROM `clients`  
WHERE MATCH (`Description_Cli`) AGAINST ('_Mot 1+ Mot 2' IN Boolean  
Mode)
```

11) Autres :

a) Création d'une Table à partir d'une autre table :

```
CREATE TABLE Nouvelle_Table AS SELECT [* ou Col1, Col2, ...] FROM  
Nom_Table
```

```
CREATE TABLE `clients_2005` AS SELECT * FROM `clients`
```

b) Les opérateurs de comparaison :

= : test de l'égalité

> : test de la supériorité

< : test de l'infériorité

!= ou <> : différence

c) Les opérateurs logiques :

AND

NOT

OR

d) Initialiser une session SQL :

Connexion une base de donnée :

```
CONNECT id_utilisateur@nom_de_la_base_de_donnée
```

Fermeture de la connexion à la base de donnée :

```
DISCONNECT ou EXIT
```