# 1. Création et mise en place d'une BDD

## Création d'une base de données

```
CREATE DATABASE NomBDD;
```

## Se Placer Sur une Base de données

```
USE NomBDD;
```

## Créer une table

```
CREATE TABLE NomTable (
NomChamp1 varchar(50),
NomChamp2 int
);
```

## Définir une clé primaire

### Première possibilité :

```
CREATE TABLE NomTable (
id int,
nom varchar(200)
CONSTRAINT PK_NomContrainte PRIMARY KEY (id)
);
```

### Seconde possibilité :

```
CREATE TABLE NomTable(
id int PRIMARY KEY,
nom varchar(200)
);
```

## Ajouter une clé primaire a une colonne existante

```
ALTER TABLE latable
ADD CONSTRAINT PK_NomContrainte
PRIMARY KEY (NomColonne);
```

## Définir une clé étrangère

```
CREATE TABLE NomTable (
id_premieretable int,
nom varchar(100),
id_AutreTable int
FOREIGN KEY id_AutreTable REFERENCES NomAutreTable(id_premieretable) #Sans lui
indiquer de nom
);
```

## Ajouter une clé étrangère a une colonne existante

```
ALTER TABLE latable
ADD constraint FK_Nomdelacleetrangere
FOREIGN KEY (lacolonne)
REFERENCES lautretable(lautrecolonne);
```

## Modifier une clé étrangère

```
ALTER TABLE latable DROP
CONSTRAINT FK_Nomdelacleetrangere
GO
ALTER TABLE latable
ADD constraint FK_NomdelaNouvellecleetrangere
FOREIGN KEY (lacolonne)
REFERENCES lautretable(lautrecolonne)
GO
```

## Supprimer une clé étrangère

```
ALTER TABLE latable DROP
CONSTRAINT FK_Nomdelacleetrangere;
```

## Supprimer une table

```
DROP TABLE NomTable;
```

## Vider une table

```
TRUNCATE TABLE NomTable;
```

## Supprimer une colonne

```
ALTER TABLE NomTable
DROP COLUMN NomCOlonne;
```

## Modifier une colonne presente dans une table

```
ALTER TABLE latable
ALTER COLUMN lacolonne VARCHAR(30);
```

## Ajouter une colonne a une table

```
ALTER TABLE NomTable
ADD NomChamps VARCHAR(30);
```

### Renommer une colonne

```
#MYSQL 5.6.x 5.7.X
ALTER TABLE infos CHANGE AncienNom NouveauNom [TYPE];

#MYSQL 8.X
ALTER TABLE NomTable
RENAME COLUMN AncienNom TO NouveauNom;
```

### Renommer une table

```
ALTER TABLE NomTable
RENAME TO NouveauNomTable;
```

### Insérer des valeurs dans une table

```
INSERT INTO NomTable
VALUES ('texte', 10), ('texte2', 36);
```

### Modifier des champs

```
#Le WHERE contient la condition qui permet de choisir dans quelles lignes se
fera la modification
UPDATE NomTable SET NomChamp='TexteModifié' WHERE NomChamp='Texte';
```

### Mettre a jour une colonne en utilisant une sous requêtes

```
UPDATE latable
set lacolonne = unalias.instruction1
FROM (
SELECT instruction1
from UNETABLE
)as unalias
```

### Supprimer des champs

```
#Le WHERE contient la condition qui permet de choisir quelles lignes seront
supprimées
DELETE FROM NomTable WHERE NomChamp='Texte';
```

# 2. Sélection et tri de données

### Sélection simple de toutes les données d'une table

```
SELECT * FROM NomTable;
```

## Sélection de données avec conditions

```
SELECT *
FROM NomTable
WHERE NomChamp='Texte';
```

## Eviter les doublons dans un résultat

```
SELECT DISTINCT NomChamp
FROM NomTable
WHERE NomChamp='Texte';
```

## Sélectionner un intervalle

```
SELECT *
FROM NomTable
WHERE
    NomChamp BETWEEN 5 AND 15;
```

## Jointure entre tables

### Joindre deux tables avec des id qui correspondent

```
SELECT
    colonne1,
    colonne2
FROM unetable
    INNER JOIN deuxiemetable ON unetable.id = deuxiemetable.id
```

### Joindre deux tables avec des id qui ne correspondent pas

```
SELECT
    colonne1,
    colonne2
FROM unetable
    LEFT JOIN deuxiemetable ON unetable.id = deuxiemetable.id # ou right JOIN

/* LEFT JOIN = LISTER tous les résultats de la table de gauche (première table
appelé) (left=gauche)
quand il n'ay pas de correspondance dans la table de droite */

/* RIGHT JOIN = LISTER tous les résultats de la table de droite(deuxième table
appelé) ( right = droite)
quand il n'y a pas de correspondance avec la table de gauche */
```

## Les alias

```
SELECT
    nomChamp AS AlliasNomChamp
FROM NomTable;
#Vous pouvez utiliser AS "Nomalias" | AS Nomalias | "Nomalias" | Nomalias
```

## Fonctions d'agrégation

### Sélectionner un maximum ou un minimum (fonctionne avec les Chaine de caractères)

```
SELECT
    MIN(NomChamp) AS Minimum,
    MAX(NomChamp) AS Maximum
FROM NomTable;
```

### Faire une moyenne

```
SELECT
    AVG(NomChamp) AS Moyenne
FROM NomTable;
```

### Faire une somme

```
SELECT
    SUM(NomChamp) AS Somme
FROM NomTable;
```

### Compter des lignes

```
SELECT
    COUNT(*) AS Nb_de_Lignes
FROM NomTable;
```

## Regrouper les résultats d'une fonction d'agrégation avec GROUP BY

```
SELECT
    count(*) AS Total,
    NomChamp FROM NomTable
GROUP BY NomChamp;
```

## Utiliser des fonctions d'agrégation en conditions de requête avec HAVING

```
SELECT
    NomChamp,
    SUM(NomChamp2)
FROM NomTable
GROUP BY NomChamp
HAVING
    SUM(NomChamp2) > 30;
```

# 3. Vue/Fonction/Procédure/Trigger

# Vue

## Créer une Vue

```
CREATE VIEW NomdelaVue AS
   SELECT instruction
   FROM NomTable;
```

## Appeler la vue

```
SELECT * FROM NomdeleVue;
```

# Fonctions

## Créer une fonction (scalaire)

```
CREATE FUNCTION
   nomdelafonction(@champs sontype(),...)
   RETURNS untype
       AS
       BEGIN /* requete ou calcul*/
       RETURN /* le résultat du calcul*/
       END
```

## Appeler la Fonction:

```
SELECT dbo.nomdelafonction(parametre);
```

# Procédure

## Créer une procédure

```
CREATE PROCEDURE nomdelaprocedure /* (@champs type(),...) */
AS
BEGIN
# la requête / instructions
END
```

## Exécuter la procédure:

```
EXEC nomdelaprocedure /*'parametre'*/;
```

# Trigger

## Créer un Trigger

```
CREATE TRIGGER nomdutrigger
ON NomTable
AFTER /* FOR | INSTEAD OF */ INSERT/*DELETE,UPDATE*/
AS
BEGIN
/*Debut de Requete*/
END
```