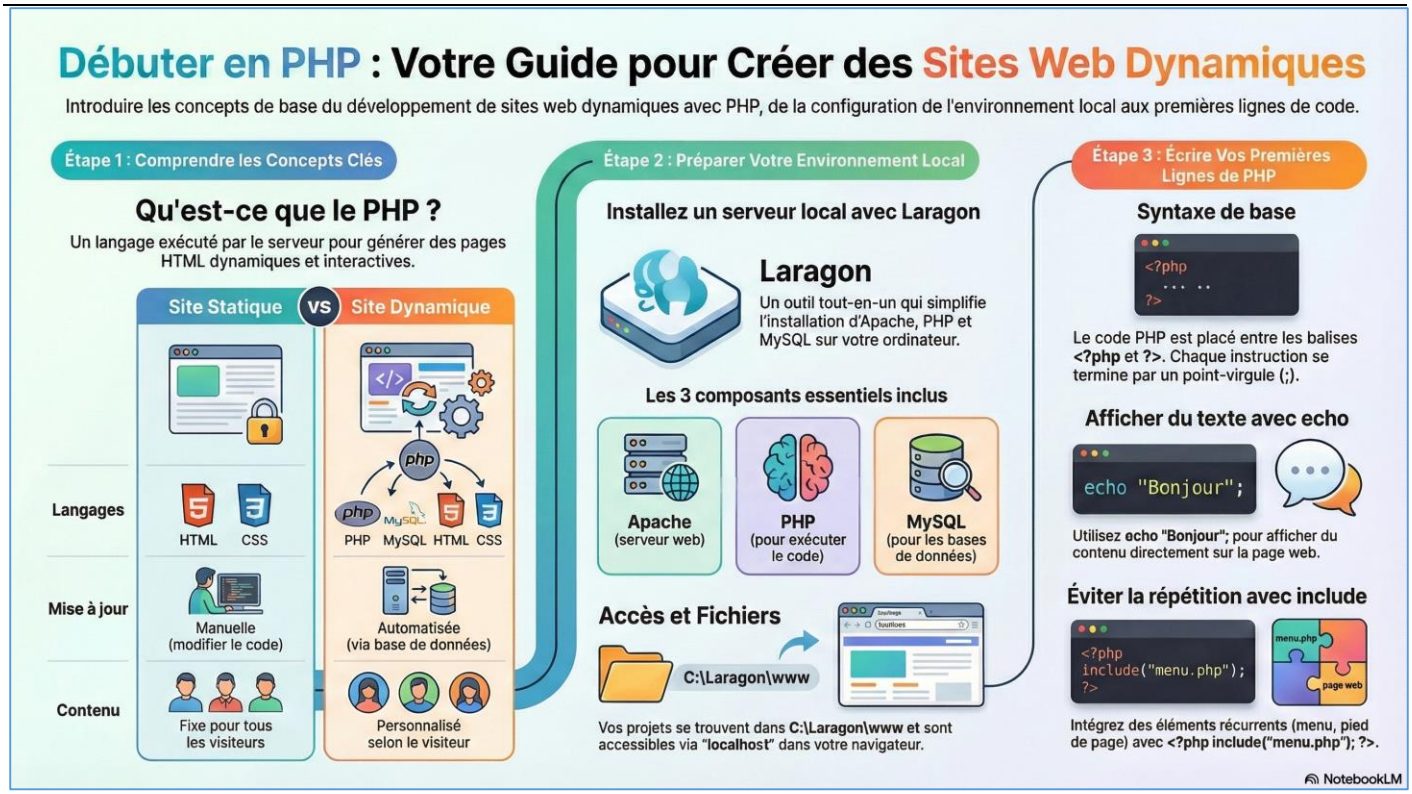
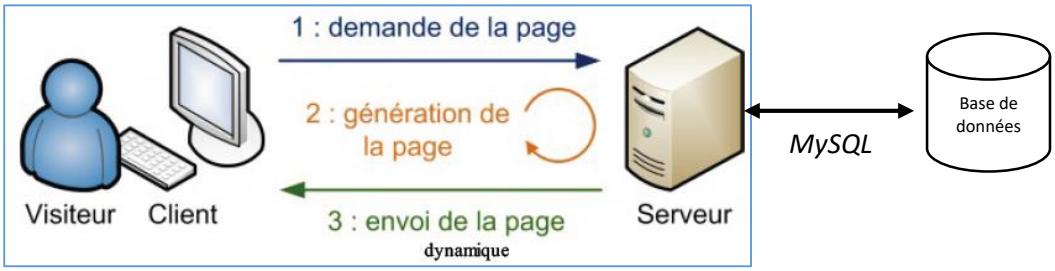
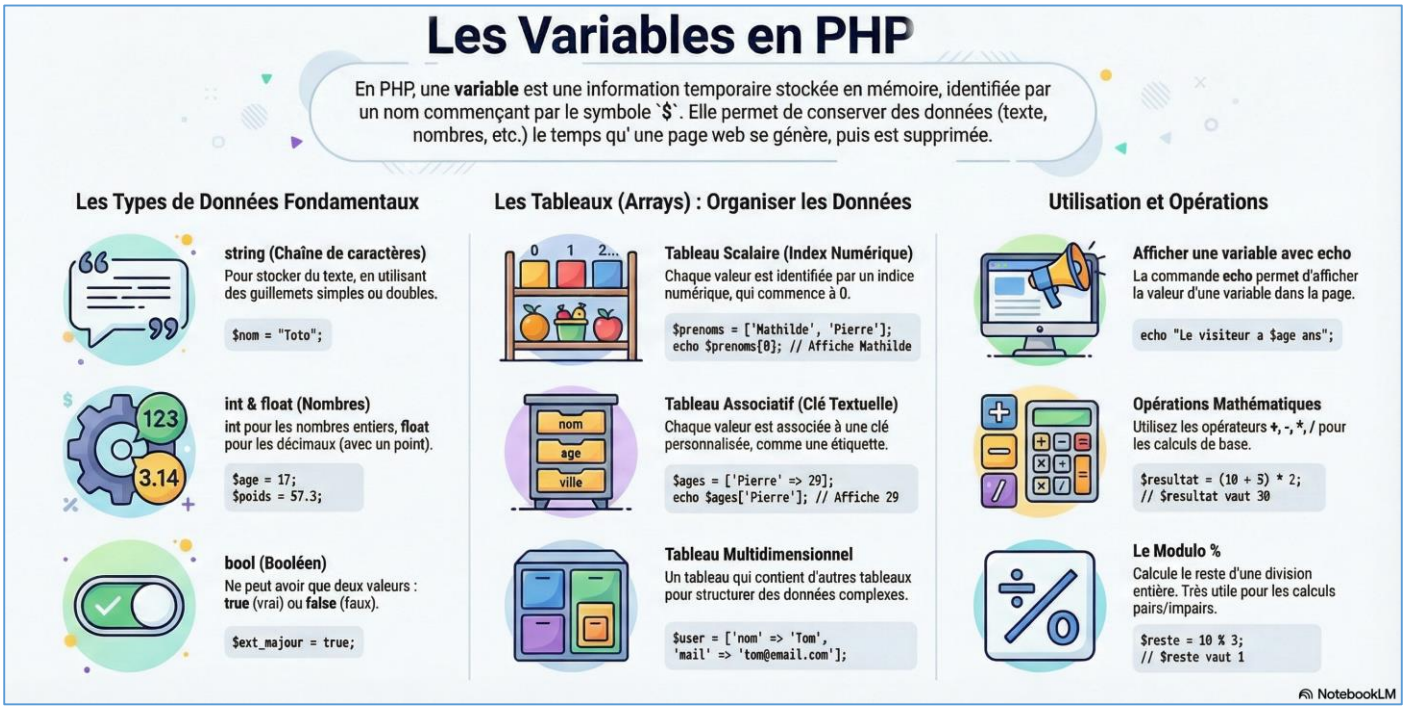
Cofinancé par l'Union européenne



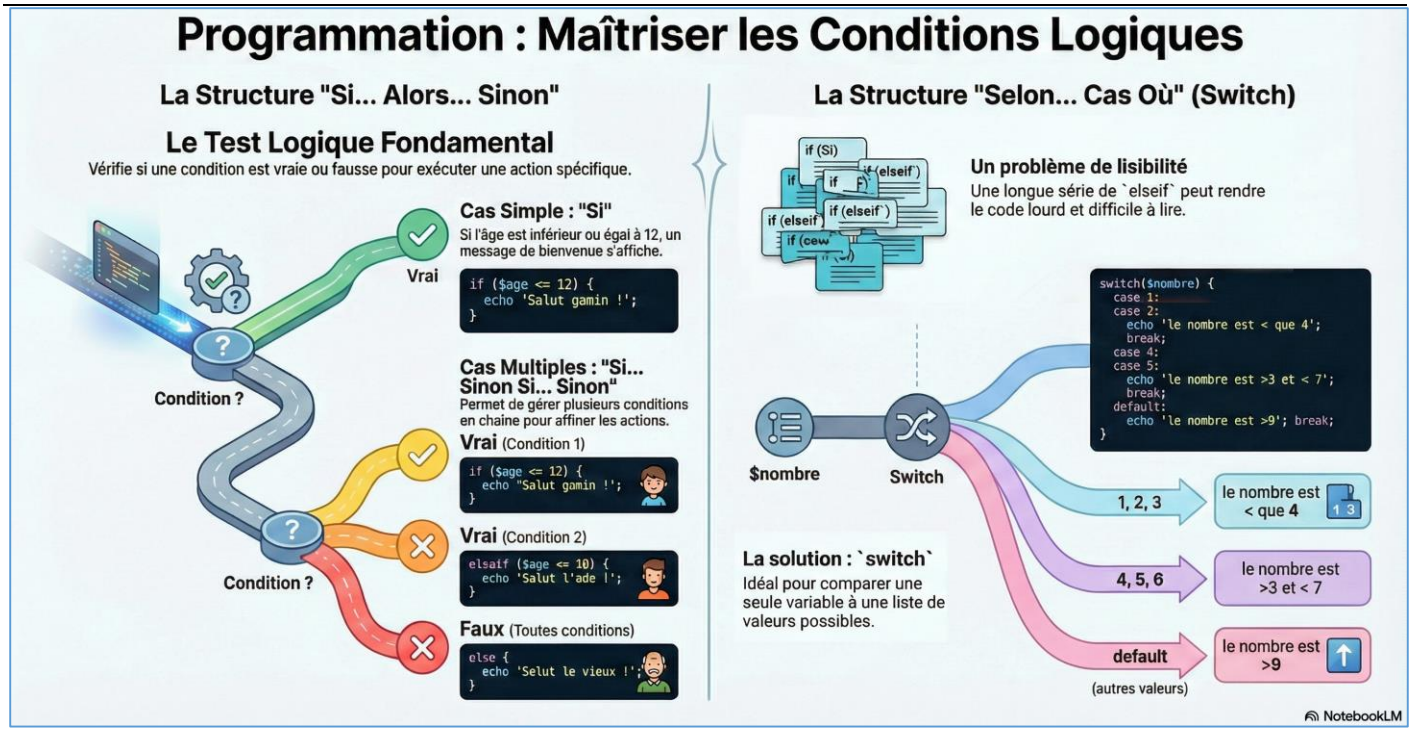

**Les sites dynamiques** :




Le **PHP** est un langage exécuté par le serveur.
Il permet de personnaliser la page en fonction du visiteur et/ou d'un contenu dans une base de données, d'effectuer des calculs, etc. Il génère une page HTML.



Cofinancé par l'Union européenne




**Conditionnelles**

```
if (      )
  {

  } else {

  }
```

```
if (      )
  {

  } elseif (     )
  {

  } else  {

  }
```

```
switch (     )
{
case 1 :
case 2 :
case 3 : …
          break;
default : …
}
```

Cofinancé par l'Union européenne

## Les répétitives

# Les Boucles en PHP : while, for & foreach

Les boucles (ou structures répétitives) sont essentielles pour exécuter un bloc d'instructions plusieurs fois en PHP. Voici les trois boucles fondamentales adaptées à des besoins spécifiques.

### La Boucle while

**Répète tant qu'une condition est vraie.**

Idéale lorsque le nombre d'itérations n'est pas connu à l'avance.


**Syntaxe de base**

Répète tant qu'une condition.

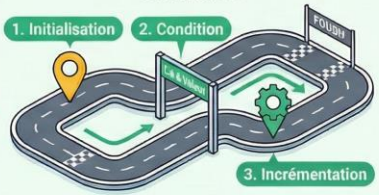
```
while ($chiffre <= 10) {
  ...
  $chiffre++;
}
```

**Exemple :** compter de 1 à 10.

### La Boucle for

**Répète pour un nombre de tours défini.**

Parfaite pour itérer avec un compteur que l'on maîtrise.




**Une structure en trois temps**

1. Initialisation, 2. Condition, 3. Incrémentation.

**Syntaxe de base**

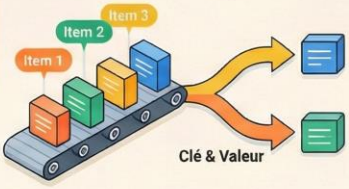
```
for ($i = 1; $i <= 100; $i++) {
  ...
}
```

**Exemple :** afficher 100 lignes.

### La Boucle foreach

**La spécialiste pour parcourir les tableaux.**

Simplifie la lecture de chaque élément d'un array.




**Deux façons de l'utiliser**

Récupérer juste la valeur **OU** récupérer la clé et la valeur.

**Syntaxe de base**

```
foreach ($tableau as $cle => $element) {
  ...
}
```

NotebookLM

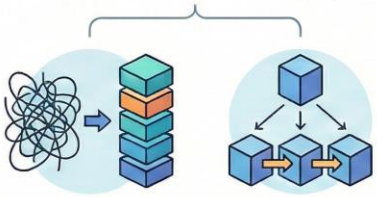
## Les fonctions

# Les Fonctions en PHP : Le Guide Essentiel

### Qu'est-ce qu'une fonction PHP ?

**Un bloc d'instructions réutilisable**

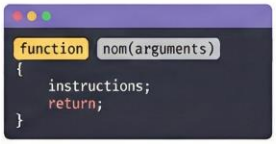
Elle effectue des actions spécifiques et peut retourner une valeur (nombre, texte, etc.).



**Les deux avantages principaux**

1. **Clarifier le code :** Rend le programme plus lisible.
2. **Réemployer le code :** Évite de réécrire les mêmes instructions.

### Créer et Utiliser une Fonction

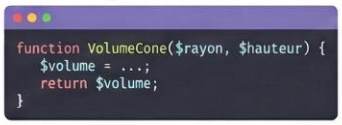


```
function nom(arguments)
{
    instructions;
    return;
}
```

**La Structure de Base (Syntaxe)**

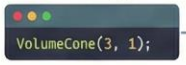
Utilisez le mot-clé `function`, donnez un nom, puis définissez les arguments entre parenthèses.

**Exemple de Déclaration**



```
function VolumeCone($rayon, $hauteur) {
   $volume = ...;
   return $volume;
}
```

**Comment l'appeler ?**



```
VolumeCone(3, 1);
```

Utilisez simplement son nom avec les valeurs des arguments : `VolumeCone(3, 1);`

### Exemples et Bonnes Pratiques


**Fonctions intégrées utiles**

PHP en possède de nombreuses pour la date (`date('Y')`) ou les textes (`strlen`, `strtoupper`).


**Bonne Pratique n°1 : Commentez**

Expliquez ce que fait votre fonction, ses arguments et ce qu'elle retourne.

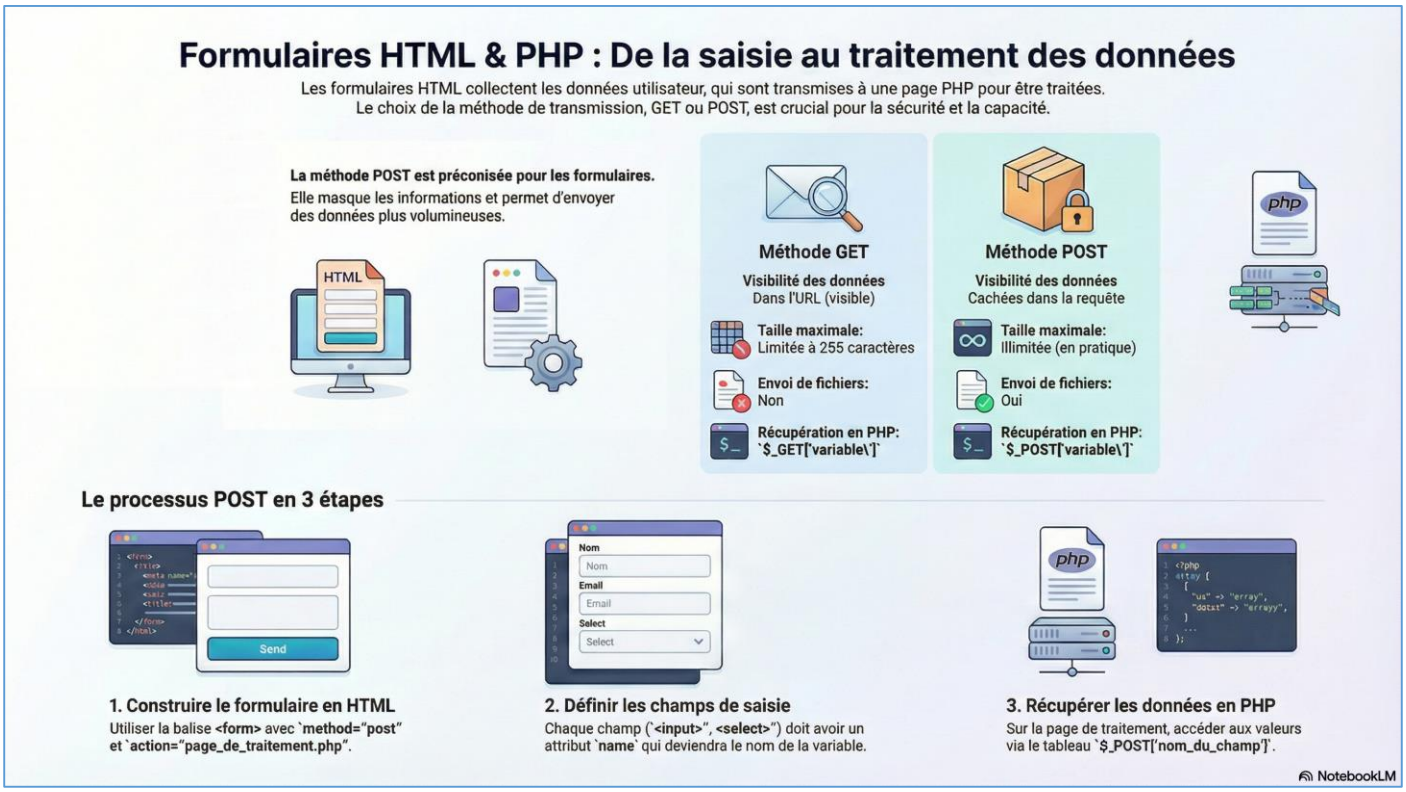
**Bonne Pratique n°2 : Organisez**

Regroupez vos fonctions un fichier séparé et incluez-le avec l'instruction `include`.

NotebookLM

Cofinancé par l'Union européenne

## Les formulaires




Méthode POST

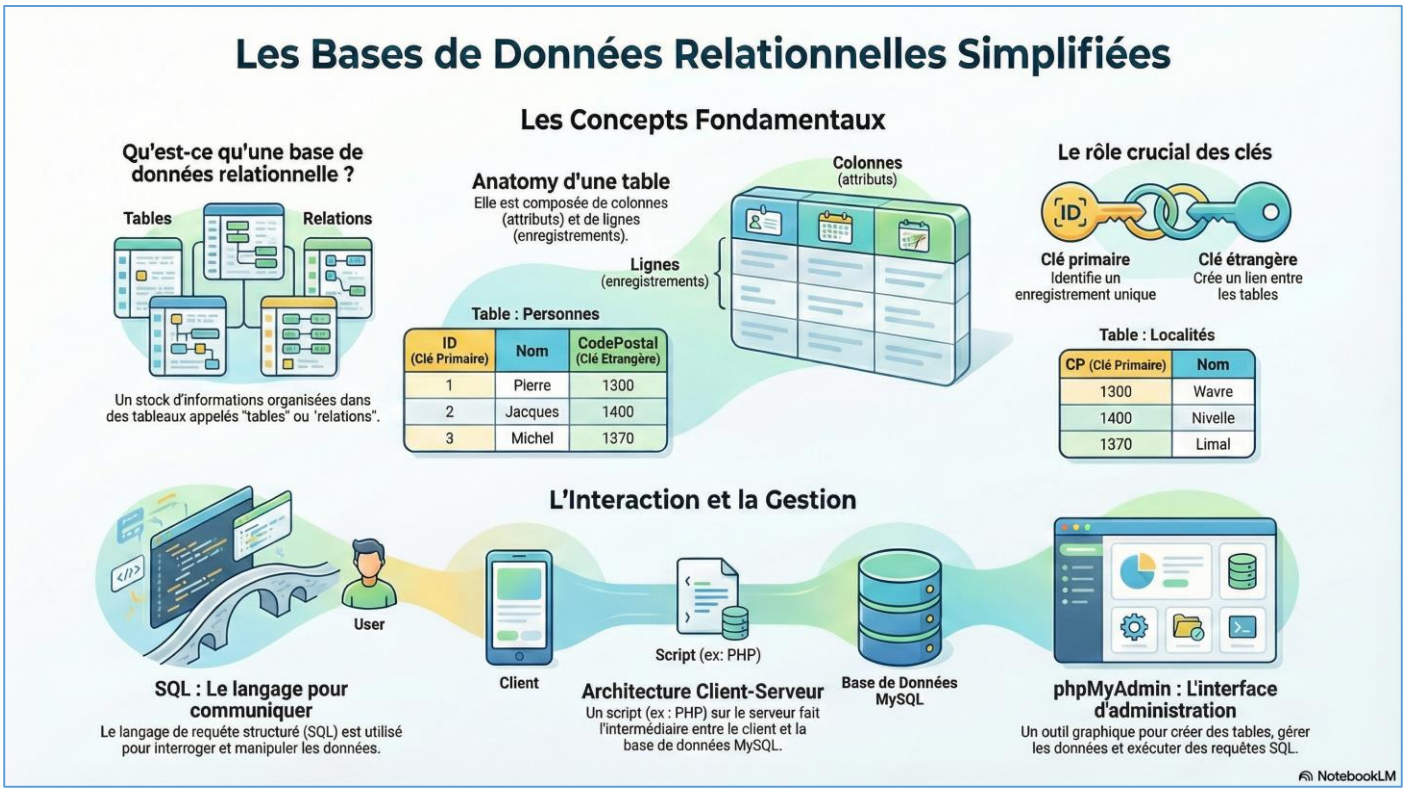
```
<form action="resultat.php" method="post">
Entrez votre prénom : <input type="text" name="prenom" />
<input type="submit" value="valider" />
</form>
```

**Récupération de la valeur d'un champ de formulaire :**

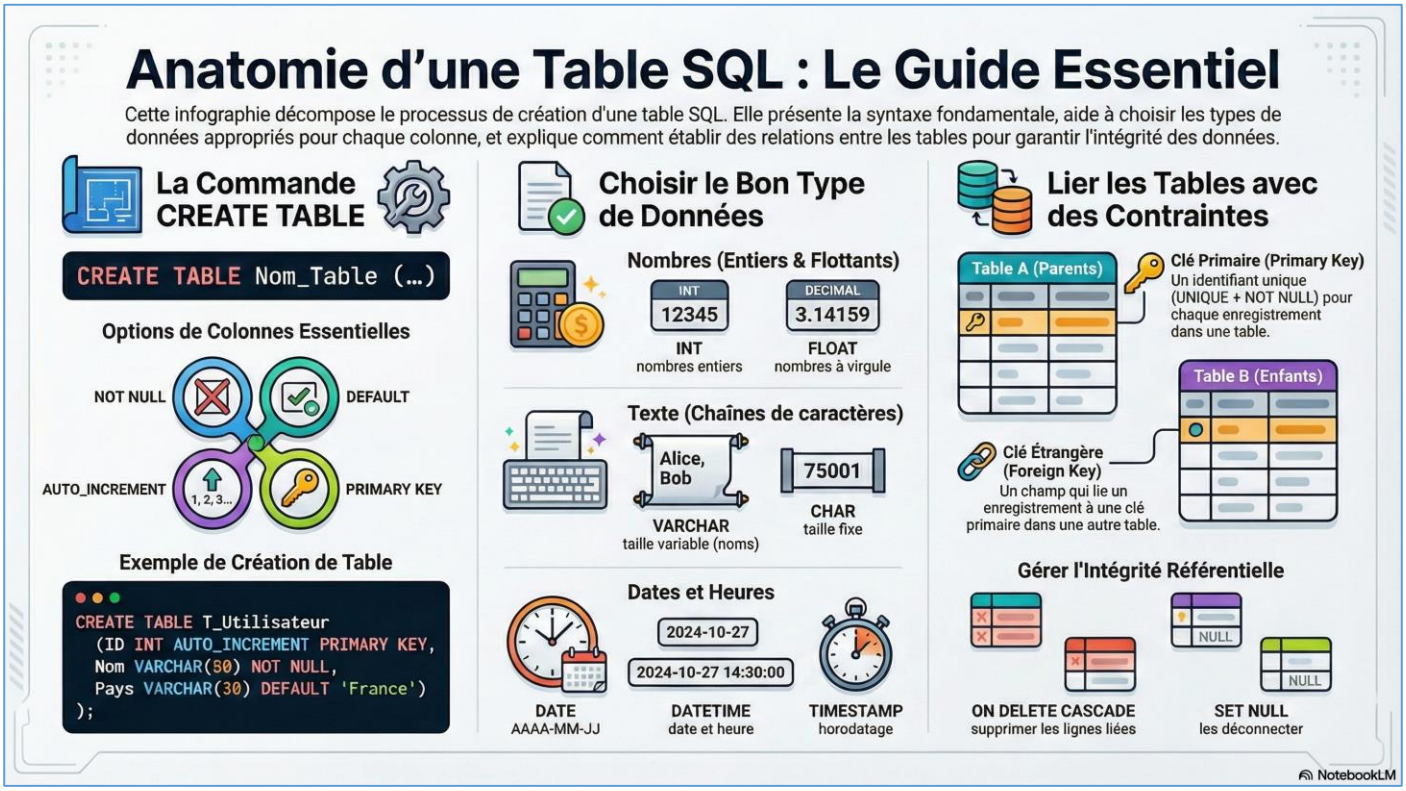
```
$_POST["prenom"]
```

Cofinancé par l'Union européenne

## Base de données

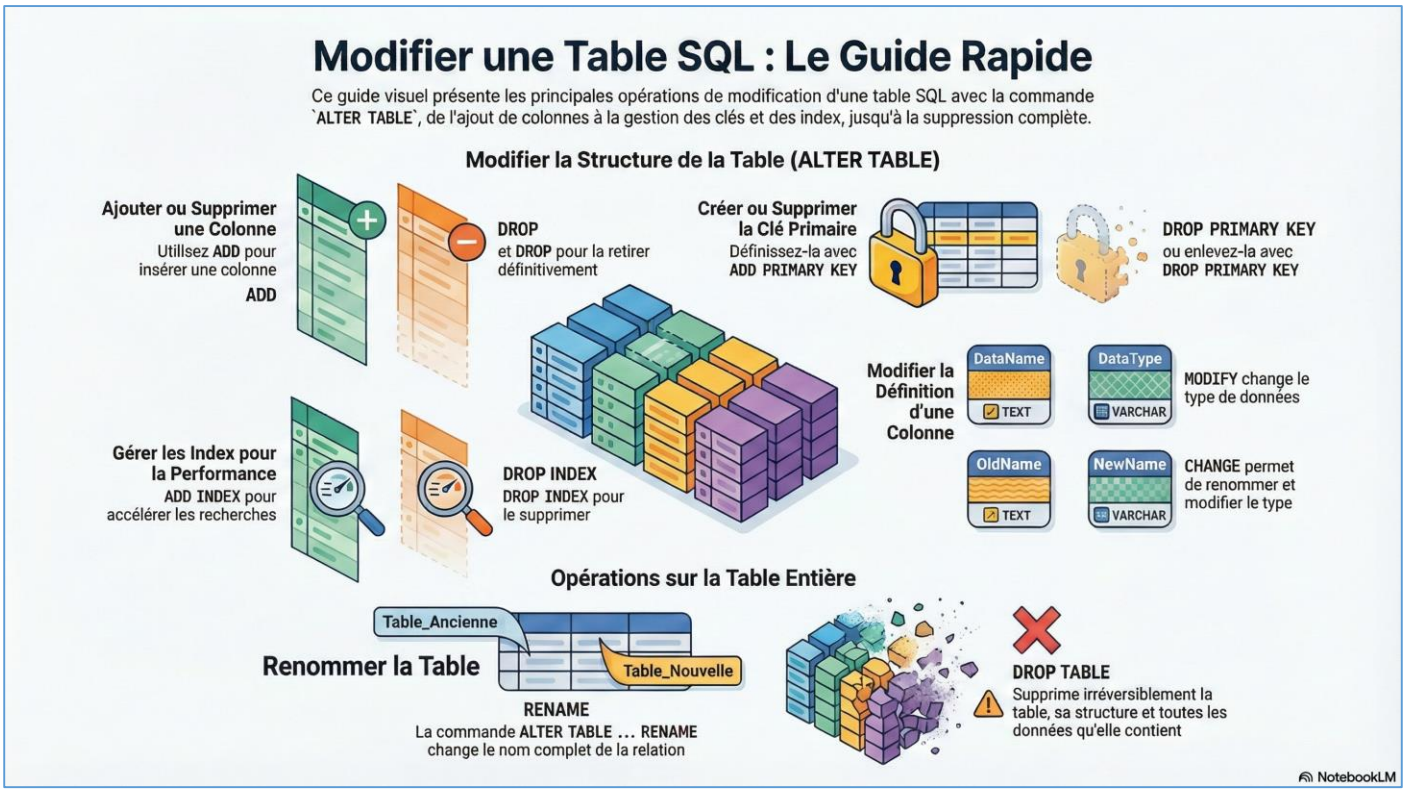



- Les instructions SQL ne sont pas sensibles à la casse

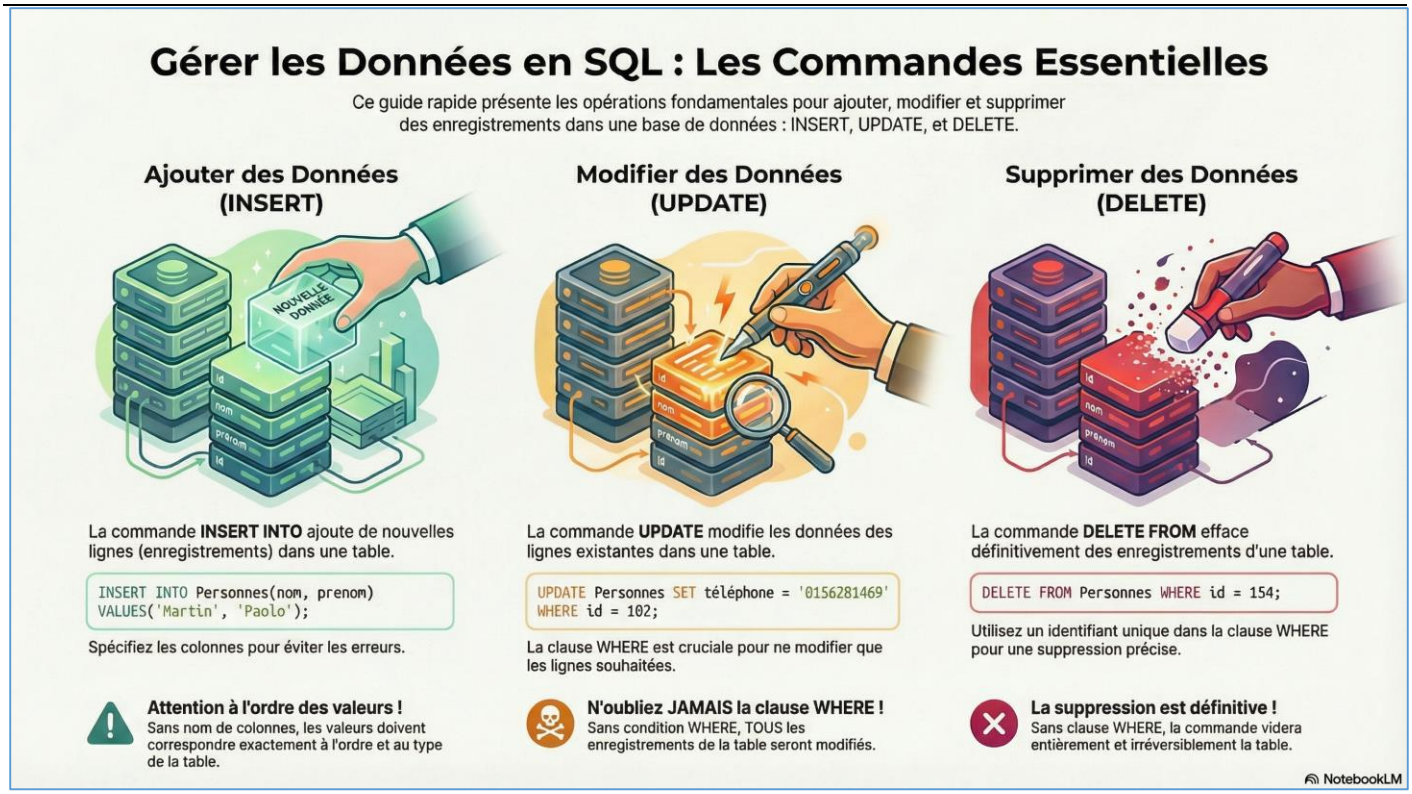


**RAPPEL – MEMENTO**
**5XBAK - Approche développement backend**

| | |
|---|---|
| Création d'une base de données | **CREATE DATABASE** |
| **Création d'une table** | **CREATE TABLE**(<br>…<br>)<br>**Champs et type de données**<br>Entiers **INT** / Flottants **FLOAT** / Chaine **VARCHAR TEXT** /Date et Heure **DATE DATETIME**/ Ensemble **ENUM** |
| **Création des relations entre les tables** ***Clé primaire (exemple ISBN, Reg Nat, ADN …)*** | **PRIMARY KEY** |
| **Attribut NOT NULL / UNIQUE**<br>**Valeur par défaut DEFAULT**<br>**INDEX ➔ recherches et les tris** | |
| **Clé étrangère** | **CONSTRAINT** fk **FOREIGN KEY** (fkey) **REFERENCES TABLE** (id) |
| **Mise à jour d'une base de données** | **ALTER TABLE … (modification)**<br>**DROP TABLE … (suppression)** |

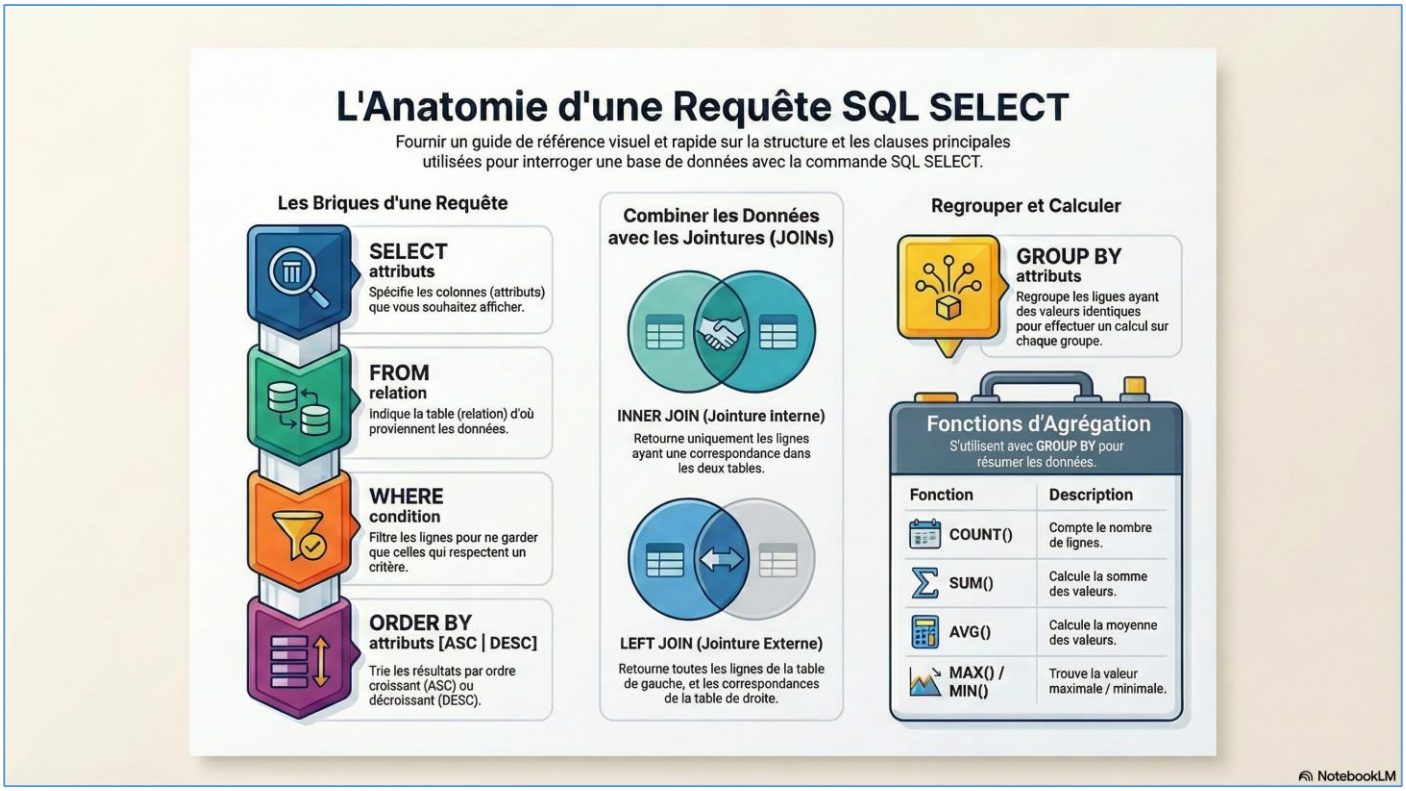


Cofinancé par l'Union européenne




| | |
|---|---|
| **Ajout de données** | **INSERT INTO TABLE (…) VALUES (…)**<br>*(attention à l'ordre des champs)* |
| **Modification de données** | **UPDATE TABLE SET** champ= ''…'' **WHERE** ID =… |
| **Suppression de données** | **DELETE FROM TABLE WHERE** ID =… |

## Interrogation de base de données



**RAPPEL – MEMENTO**
**5XBAK - Approche développement backend**

Cofinancé par l'Union européenne

## Requêtes de sélection

### Simple

```
SELECT attribut FROM table
```

### Avec filtre

```
SELECT attribut
FROM table
WHERE attribut = '' ... ''
```

### Avec tri

```
SELECT attribut
FROM table
ORDER BY attribut ASC
```

### Suppression des doublons

```
SELECT DISTINCT attribut FROM table
```

### Avec jointure

```
SELECT attribut
FROM table1, table2
WHERE table1.attribut = table2.attribut
```

## Multiple

### Requêtes imbriquées

```
SELECT * FROM table1
WHERE ID=( SELECT ID FROM table2  WHERE attribut = '' ... '')
```

### Requêtes sélection multi-tables

```
SELECT *
FROM table1,table2
WHERE table1.ID = table2.IDREF
AND attribut = '' ... ''
```

### Avec regroupement

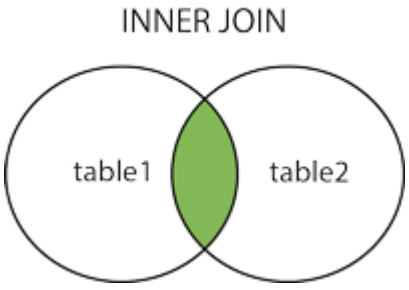
```
SELECT ...
FROM table
GROUP BY attribut
HAVING...
```

### Champs calculés/ Fonctions de MySQL

```
MIN(x), MAX(x), AVG(x), SUM(x), COUNT(x)...
```

### Les jointures internes

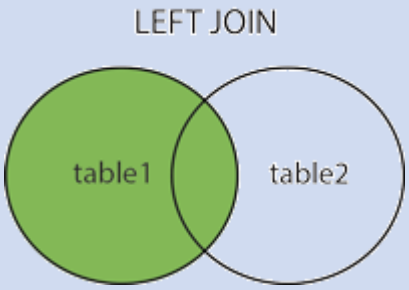
```
INNER JOIN
```




```
SELECT ...
FROM table1
INNER JOIN table2 ON table1.ID = table2.IDREF
```

### Les jointures externes

```
LEFT OUTER JOIN
```

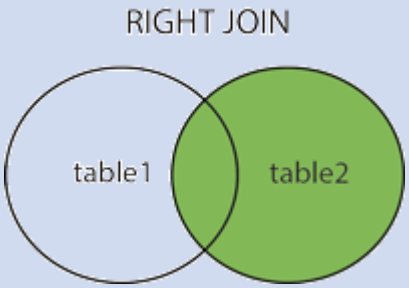



```
SELECT ...
```

Cofinancé par l'Union européenne

```
FROM table1
LEFT OUTER  JOIN table2 ON table1.ID = table2.IDREF
```

**RIGHT OUTER JOIN**




```
SELECT …
FROM table1
RIGHT OUTER  JOIN table2 ON table1.ID = table2.IDREF
```

## Requêtes ensemblistes

| | |
|---|---|
| **Union** | SELECT attribut FROM table1<br>UNION<br>SELECT attribut FROM table2 |
| **Intersection** | SELECT attribut FROM table1<br>WHERE attribut IN (SELECT attribut FROM table2) |
| **Différence** | SELECT attribut FROM table1<br>WHERE attribut NOT IN (SELECT attribut FROM table2) |
| **Produit cartésien** | SELECT * FROM table1,table2 |

Cofinancé par l'Union européenne

**PDO** est une **classe PHP** destinée à permettre à PHP de communiquer avec un serveur de données. (PDO : Php Data Object)

```php
<?php

// on se connecte à MySQL et on sélectionne la base
$connection = mysqli_connect('dns', 'utilisateur', 'motdepasse', 'basededonnees'[, 'port']);
?>
```

Pour **sélectionner des enregistrements**, nous utiliserons la méthode query($requete).

```php
<?php
// On établit la connexion
Include ('connexion.php');

// On envoie la requête
$select = $connection->query("SELECT * FROM t_contact");
?>
```

Affichage (SELECT)

```php
<?php
while( $enregistrement =mysqli_fetch_array($select))
{
  // Affichage des champs
  echo '<h1>' .  $enregistrement['Nom'] . ' ' . $enregistrement['Prenom'] .  '</h1>';
}
?>
```

```php
<?php
// Ajout de données
// On créé la requête
$req = "INSERT INTO table(texte) VALUES ('Du texte')";

// on envoie la requête
$res = $conn->query($req);
```

```php
// Mise à jour de données
// On créé la requête
$req = "UPDATE  FROM table WHERE id=XXX";

// on envoie la requête
$res = $conn->query($req);
```

```php
// Suppression de données
// On créé la requête
$req = "DELETE FROM table WHERE id=XXX";

// on envoie la requête
$res = $conn->query($req);
```