Insérer des données en SQL (`INSERT`)

Pour enregistrer des données dans une table (comme des élèves, des livres ou des produits), tu dois les **insérer** (*insert*). C'est justement à ça que sert la commande SQL `INSERT`. Dans ce cours, tu vas apprendre comment fonctionne cette commande, comment l'écrire correctement, et comment éviter les erreurs les plus courantes. C'est une compétence de base essentielle pour travailler avec des bases de données.

Objectifs du cours

À la fin de cette leçon, tu seras capable de :

- Comprendre à quoi sert une requête `INSERT`
- Connaître la syntaxe de base pour ajouter des données dans une base MySQL
- Faire la différence entre un `INSERT` simple et un `INSERT` multiple
- Utiliser correctement les guillemets et les types de données
- Éviter les erreurs classiques lors de l'insertion

Introduction : c'est quoi une requête `INSERT` ?

Une requête `INSERT` sert à **ajouter de nouvelles lignes** (ou *enregistrements*) dans une table d'une base de données.

Imagine une table `Eleves` : Elle est vide au départ. Pour y mettre des données, on va utiliser `INSERT`.

💡 On n'utilise **pas** `INSERT` pour modifier ou supprimer des données déjà présentes. Pour ça, on utilise `UPDATE` ou `DELETE`.

Syntaxe de base en MySQL

Voici la **forme générale** d'une requête `INSERT` :

```sql
INSERT INTO nom_table (colonne1, colonne2, ...)
VALUES (valeur1, valeur2, ...);
```

Chaque valeur correspond à une colonne, dans l'ordre donné.

Exemple simple

Prenons cette table `Eleves` :

```sql
CREATE TABLE Eleves (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50),
    age INT,
    classe VARCHAR(10)
);
```

On veut ajouter un élève : Jean Dupont, 15 ans, en 4B.

```sql
INSERT INTO Eleves (nom, age, classe)
VALUES ('Jean Dupont', 15, '4B');
```

✅ Le champ `id` n'est **pas mentionné** car il est auto-incrémenté : MySQL s'en occupe tout seul.

Détails importants

Élément	Explication
`'Jean Dupont'`	Les chaînes de caractères doivent être entre **guillemets simples** `'`
`15`	Les nombres **n'ont pas** de guillemets
`'4B'`	Une classe est une chaîne de caractères : on met des guillemets

Ajouter plusieurs lignes en une seule requête

Pour insérer **plusieurs élèves** en une seule fois :

```sql
INSERT INTO Eleves (nom, age, classe)
VALUES
  ('Alice Martin', 14, '3A'),
  ('Tom Leroy', 16, '5C'),
  ('Sofia Van Damme', 15, '4B');
```

💡 Avantage : c'est plus rapide que faire 3 `INSERT` séparés.

Et si j'oublie une colonne ?

Si la colonne a une **valeur par défaut** ou autorise les valeurs `NULL`, ce n'est pas un problème.

Exemple :

```sql
INSERT INTO Eleves (nom, age)
VALUES ('Laura Dubois', 16);
```

Ici, `classe` sera mise à `NULL`.

⚠️ Mais si `classe` est définie comme `NOT NULL`, cette requête va échouer.

Résumé : 5 choses essentielles à retenir

1. `INSERT` ajoute **une ou plusieurs lignes** dans une table.
2. Il faut respecter **l'ordre** des colonnes indiqué.
3. Les **chaînes de caractères** sont entourées de `'guillemets simples'`.
4. Les colonnes `AUTO_INCREMENT` comme `id` peuvent être **ignorées**.
5. On peut insérer **plusieurs lignes** en une seule commande avec plusieurs `VALUES`.

Bonnes pratiques

- Toujours **vérifier les types de données** de chaque colonne
- Utiliser les **guillemets simples** pour les textes (et pas des doubles `"`)
- Tester une requête `INSERT` sur une **base de test** avant de l'appliquer en production
- Pour copier-coller un grand nombre d'inserts, pense à bien **formater les données**

À toi de jouer

Voici une table `Livres` :

```sql
CREATE TABLE Livres (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titre VARCHAR(100),
    auteur VARCHAR(50),
    annee INT
);
```

Question : écris une requête `INSERT` qui ajoute ce livre :

Titre : *Le Petit Prince* Auteur : *Antoine de Saint-Exupéry* Année : *1943*

Tu peux aussi essayer d'ajouter deux autres livres en une seule commande.