Cours SQL – Les Jointures avec MySQL

🔹 Pourquoi les jointures ?

Dans une base relationnelle bien conçue, les données sont réparties dans plusieurs tables. Pour les croiser, on utilise les jointures.

Exemple :
- Table utilisateurs
- Table commandes

Tu veux afficher les commandes avec le nom de l'utilisateur ? Il faut une jointure !

🔹 Les types de jointures dans MySQL

✅ INNER JOIN – La plus utilisée

Elle renvoie uniquement les lignes qui ont une correspondance dans les deux tables.

Exemple :
```
SELECT u.nom, c.date
FROM utilisateurs u
INNER JOIN commandes c ON u.id = c.utilisateur_id;
```

Résultat : uniquement les utilisateurs ayant passé au moins une commande.

✅ LEFT JOIN – Toutes les lignes de gauche

Renvoie toutes les lignes de la table de gauche, même si pas de correspondance dans la table de droite.

Exemple :
```
SELECT u.nom, c.date
FROM utilisateurs u
LEFT JOIN commandes c ON u.id = c.utilisateur_id;
```

Résultat : tous les utilisateurs, même ceux sans commande (valeurs NULL à droite).

✅ RIGHT JOIN – Toutes les lignes de droite

Renvoie toutes les lignes de la table de droite, même si pas de correspondance à gauche.

Exemple :
SELECT u.nom, c.date
FROM utilisateurs u
RIGHT JOIN commandes c ON u.id = c.utilisateur_id;

Résultat : toutes les commandes, même celles non associées à un utilisateur.

✅ FULL OUTER JOIN – Astuce pour MySQL

MySQL ne supporte pas nativement FULL OUTER JOIN.
Tu peux l'émuler ainsi :

SELECT u.nom, c.date
FROM utilisateurs u
LEFT JOIN commandes c ON u.id = c.utilisateur_id

UNION

SELECT u.nom, c.date
FROM utilisateurs u
RIGHT JOIN commandes c ON u.id = c.utilisateur_id;

🔹 Tables de liaison (relation plusieurs à plusieurs)

Exemple : un étudiant peut suivre plusieurs cours, et un cours peut avoir plusieurs étudiants.

Il faut une table de jointure :

```
CREATE TABLE etudiants (
  id INT PRIMARY KEY,
  nom VARCHAR(100)
);

CREATE TABLE cours (
```

```
  id INT PRIMARY KEY,
  titre VARCHAR(100)
);

CREATE TABLE inscriptions (
  etudiant_id INT,
  cours_id INT,
  PRIMARY KEY (etudiant_id, cours_id),
  FOREIGN KEY (etudiant_id) REFERENCES etudiants(id),
  FOREIGN KEY (cours_id) REFERENCES cours(id)
);
```

Requête pour voir les cours suivis par un étudiant :

```
SELECT e.nom, c.titre
FROM etudiants e
JOIN inscriptions i ON e.id = i.etudiant_id
JOIN cours c ON i.cours_id = c.id;
```

◆ Exemples pratiques

Voir les utilisateurs et leurs commandes (même ceux sans commande) :

```
SELECT u.nom, c.id AS commande_id
FROM utilisateurs u
LEFT JOIN commandes c ON u.id = c.utilisateur_id;
```

Compter le nombre de commandes par utilisateur :

```
SELECT u.nom, COUNT(c.id) AS nb_commandes
FROM utilisateurs u
LEFT JOIN commandes c ON u.id = c.utilisateur_id
GROUP BY u.nom;
```

◆ Cas particulier : SELF JOIN (auto-jointure)

Permet de comparer des lignes d'une même table.

Exemple :

```
SELECT a.nom AS employe, b.nom AS manager
FROM employes a
JOIN employes b ON a.manager_id = b.id;
```

◆ Résumé des JOIN (MySQL)

```
INNER JOIN       : Lignes correspondantes des deux tables
LEFT JOIN        : Toutes les lignes de gauche + correspondance (ou NULL)
RIGHT JOIN       : Toutes les lignes de droite + correspondance (ou NULL)
FULL OUTER JOIN  : Simulation avec UNION de LEFT et RIGHT JOIN
SELF JOIN        : Jointure sur la même table
```