Partie 4 : Introduction au langage SQL

Propriétaire	Ⓜ Marine
Étiquettes	

1. Qu'est-ce que le SQL ?

SQL (Structured Query Language) est un langage utilisé pour interagir avec les bases de données relationnelles.

Fonctionnalités principales :

- **Définir des structures de données** : Créer des tables, des index, etc.
- **Manipuler les données** : Ajouter, modifier, supprimer et interroger des données.
- **Contrôler l'accès** : Gérer les utilisateurs et leurs droits.

2. Les commandes SQL de base

2.1. Créer une table

```
CREATE TABLE Clients (
    ID_Client INT PRIMARY KEY,
    Nom VARCHAR(50),
    Email VARCHAR(100)
);
```

2.2. Insérer des données

```
INSERT INTO Clients (ID_Client, Nom, Email)
VALUES (1, 'Dupont', 'dupont@email.com');
```

2.3. Lire des données

```
SELECT * FROM Clients;
```

2.4. Mettre à jour des données

```
UPDATE Clients
SET Email = 'nouveau@email.com'
WHERE ID_Client = 1;
```

2.5. Supprimer des données

```
DELETE FROM Clients
WHERE ID_Client = 1;
```

3. Requêtes plus avancées

3.1. Relation entre tables

Afficher les commandes et les noms des clients :

```
SELECT Commandes.ID_Commande, Clients.Nom, Commandes.Montant
FROM Commandes
INNER JOIN Clients ON Commandes.ID_Client = Clients.ID_Client;
```

3.2. Filtrer les données

Afficher les clients avec un montant de commande supérieur à 50 (voir le cours sur les jointures)

```
SELECT Clients.Nom, Commandes.Montant
FROM Commandes
INNER JOIN Clients ON Commandes.ID_Client = Clients.ID_Client
WHERE Commandes.Montant > 50;
```

3.3. Fonctions d’agrégation

- `COUNT()` : compte le nombre de lignes
- `SUM()` : somme des valeurs
- `AVG()` : moyenne
- `MAX()` / `MIN()` : maximum / minimum

Exemple : afficher le total des montants de toutes les commandes :

```
SELECT SUM(Montant) AS Total_Commandes FROM Commandes;
```

Voir le cours sur les fonctions de calculs

3.4. GROUP BY

Permet de grouper les résultats selon une ou plusieurs colonnes.

Exemple : afficher le total des montants par client :

```
SELECT ID_Client, SUM(Montant) AS Total
FROM Commandes
GROUP BY ID_Client;
```

3.5. HAVING

`HAVING` permet de filtrer les résultats **après** un `GROUP BY`.

Exemple : clients ayant un total de commandes supérieur à 100 :

```
SELECT ID_Client, SUM(Montant) AS Total
FROM Commandes
GROUP BY ID_Client
HAVING SUM(Montant) > 100;
```

4. Bonnes pratiques avec SQL

- Utilisez des clés primaires et étrangères pour garantir la cohérence des données.

- Limitez l'accès aux données sensibles en gérant les permissions.
- Faites des sauvegardes régulières de vos bases de données.

Partie 4 : Exercices pratiques

Niveau débutant

1. **Afficher tous les livres disponibles dans la bibliothèque.**
2. **Lister les lecteurs vivant à Paris.**

1. **Afficher le titre et l'auteur de tous les livres.**

🟡 Niveau intermédiaire

1. **Lister les emprunts en cours avec les noms et prénoms des lecteurs.**

   (Indice : jointure entre `EMPRUNTS` *et* `LECTEURS`*, filtrer sur* `etat = 'EN COURS'`*)*
2. **Afficher le nombre de livres empruntés par chaque lecteur.**

   (Indice : `GROUP BY noLecteur`*, avec jointure possible avec* `LECTEURS`*)*
3. **Afficher les livres qui n'ont plus de stock (stock = 0).**

🔴 Niveau avancé

1. **Afficher pour chaque auteur, le nombre total de livres écrits.**

   (Indice : jointure entre `LIVRES` *et* `AUTEURS`*,* `GROUP BY` *auteur)*
2. **Lister les catégories ayant au moins 2 livres référencés.**

   (Indice : jointure avec `CATEGORIES`*,* `GROUP BY`*,* `HAVING COUNT(*) >= 2`*)*
3. **Afficher les livres qui ont été empruntés au moins deux fois.**

   (Indice : `GROUP BY refLivre`*,* `HAVING COUNT(*) >= 2`*)*
4. **Afficher les lecteurs qui ont fait au moins une réservation et une seule.**

   (Indice : table `RESERVATIONS`*,* `GROUP BY noLecteur`*,* `HAVING COUNT(*) = 1`*)*