

# Exercices corrigés SQL

Machine Learning

Octobre 2024

## Table des matières

<b>1</b>	<b>Le schéma utilisé pour ces exercices</b>	<b>3</b>
<b>2</b>	<b>Exercices</b>	<b>4</b>
	Exercice 1 : Création des Tables . . . . .	4
	Exercice 2 : Insérer des Données . . . . .	5
	Exercice 3 : Sélectionner les Livres Disponibles . . . . .	6
	Exercice 4 : Trier les Livres par Date de Publication . . . . .	6
	Exercice 5 : Filtrer les Emprunts en Cours . . . . .	7
	Exercice 6 : Calculer la Durée d'un Emprunt . . . . .	7
	Exercice 7 : Jointure sur les Livres et les Auteurs . . . . .	7
	Exercice 8 : Filtrer les Emprunteurs qui n'ont pas encore rendu de livres . . . . .	8
	Exercice 9 : Nombre de Livres par Genre . . . . .	8
	Exercice 10 : Durée Moyenne d'Emprunt par Emprunteur . . . . .	9
	Exercice 11 : Jointure avec Emprunteurs, Livres, et Genres . . . . .	9
	Exercice 12 : Livres les Plus Empruntés . . . . .	9
	Exercice 13 : Nombre de Livres Empruntés par Emprunteur . . . . .	10
	Exercice 14 : Livres Jamais Empruntés . . . . .	10
	Exercice 15 : Nombre d'Emprunteurs par Auteur . . . . .	10

# 1 Le schéma utilisé pour ces exercices



Ici, chaque ligne de la table **Livres** correspond à un vrai livre physique. Donc un livre n'est emprunté qu'une fois à la fois.

## 2 Exercices

Attention, il n'y a pas qu'une seule bonne réponse.

### Exercice 1 : Création des Tables

#### Énoncé

Écrivez les requêtes SQL pour créer toutes les tables du schéma suivant :

- **Livres** (*LivreID*, Titre, AuteurID, GenreID, DatePublication, Disponible)
- **Auteurs** (*AuteurID*, Nom, Prénom, Pays)
- **Genres** (*GenreID*, NomGenre)
- **Emprunteurs** (*EmprunteurID*, Nom, Prénom, Email, Téléphone)
- **Emprunts** (*EmpruntID*, LivreID, EmprunteurID, DateEmprunt, DateRetourPrévue, DateRetourEffective)

Attention à bien respecter les types renseignés.

#### Correction

```
CREATE TABLE Auteurs (  
  AuteurID INT PRIMARY KEY,  
  Nom VARCHAR(100),  
  Prénom VARCHAR(100),  
  Pays VARCHAR(100)  
);
```

```
CREATE TABLE Genres (  
  GenreID INT PRIMARY KEY,  
  NomGenre VARCHAR(100)  
);
```

```
CREATE TABLE Livres (  
  LivreID INT PRIMARY KEY,  
  Titre VARCHAR(255),  
  AuteurID INT,  
  GenreID INT,  
  DatePublication DATE,  
  Disponible BOOLEAN,  
  FOREIGN KEY (AuteurID) REFERENCES Auteurs(AuteurID),  
  FOREIGN KEY (GenreID) REFERENCES Genres(GenreID)  
);
```

```
CREATE TABLE Emprunteurs (  
  EmprunteurID INT PRIMARY KEY,  
  Nom VARCHAR(100),  
  Prénom VARCHAR(100),  
  Email VARCHAR(150),  
  Téléphone VARCHAR(20)  
);
```

```
CREATE TABLE Emprunts (  
  EmpruntID INT PRIMARY KEY,  
  LivreID INT,  
  EmprunteurID INT,  
  DateEmprunt DATE,  
  DateRetourPrévue DATE,  
  DateRetourEffective DATE,  
  FOREIGN KEY (LivreID) REFERENCES Livres(LivreID),  
  FOREIGN KEY (EmprunteurID) REFERENCES Emprunteurs(EmprunteurID)
```

```
);
```

**Explication** : Chaque table est créée avec les colonnes et les clés primaires. Les clés étrangères sont également ajoutées pour établir les relations entre les tables.

## Exercice 2 : Insérer des Données

Utilisez ce code pour insérer les données :

```
INSERT INTO Auteurs (AuteurID, Nom, Prénom, Pays) VALUES
```

```
(1, 'Hugo', 'Victor', 'France'),
(2, 'Orwell', 'George', 'Royaume-Uni'),
(3, 'Asimov', 'Isaac', 'Russie'),
(4, 'Tolkien', 'J.R.R.', 'Royaume-Uni'),
(5, 'Austen', 'Jane', 'Royaume-Uni'),
(6, 'Dumas', 'Alexandre', 'France'),
(7, 'Bradbury', 'Ray', 'États-Unis'),
(8, 'Camus', 'Albert', 'France'),
(9, 'Verne', 'Jules', 'France'),
(10, 'Hemingway', 'Ernest', 'États-Unis');
```

```
INSERT INTO Genres (GenreID, NomGenre) VALUES
```

```
(1, 'Roman'),
(2, 'Science-fiction'),
(3, 'Fantasy'),
(4, 'Classique'),
(5, 'Philosophie'),
(6, 'Aventure'),
(7, 'Horreur'),
(8, 'Biographie');
```

```
INSERT INTO Livres (LivreID, Titre, AuteurID, GenreID, DatePublication,
    Disponible) VALUES
```

```
(1, 'Les Misérables', 1, 1, '1862-01-01', TRUE),
(2, '1984', 2, 2, '1949-06-08', FALSE),
(3, 'Fondation', 3, 2, '1951-01-01', TRUE),
(4, 'Le Seigneur des Anneaux', 4, 3, '1954-07-29', TRUE),
(5, 'Orgueil et Préjugés', 5, 4, '1813-01-28', TRUE),
(6, 'Le Comte de Monte-Cristo', 6, 6, '1844-08-28', TRUE),
(7, 'Fahrenheit 451', 7, 2, '1953-10-19', TRUE),
(8, 'L'Étranger', 8, 5, '1942-01-01', FALSE),
(9, 'Vingt mille lieues sous les mers', 9, 6, '1870-06-20', TRUE),
(10, 'Le Vieil Homme et la Mer', 10, 4, '1952-09-01', FALSE),
(11, 'Les Trois Mousquetaires', 6, 6, '1844-03-14', TRUE),
(12, 'Le Château', NULL, 4, '1926-01-01', TRUE);
```

```
INSERT INTO Emprunteurs (EmprunteurID, Nom, Prénom, Email, Téléphone) VALUES
```

```
(1, 'Dupont', 'Jean', 'jean.dupont@mail.com', '0601020304'),
(2, 'Martin', 'Lucie', 'lucie.martin@mail.com', '0602030405'),
(3, 'Bernard', 'Paul', 'paul.bernard@mail.com', '0603040506'),
(4, 'Durand', 'Sophie', 'sophie.durand@mail.com', '0604050607'),
(5, 'Lefevre', 'Antoine', NULL, '0605060708'),
(6, 'Roux', 'Marie', 'marie.roux@mail.com', '0606070809'),
(7, 'Moreau', 'Julie', 'julie.moreau@mail.com', '0607080910'),
(8, 'Petit', 'Nicolas', 'nicolas.petit@mail.com', '0608091011'),
(9, 'Girard', 'Laure', 'laure.girard@mail.com', '0609101112'),
(10, 'Andre', 'Thomas', 'thomas.andre@mail.com', NULL),
(11, 'Lam', 'Marc', 'marc.lam@mail.com', '0609101113')
;
```

```
INSERT INTO Emprunts (EmpruntID, LivreID, EmprunteurID, DateEmprunt, DateRetourPr
    évue, DateRetourEffective) VALUES
```

```
(1, 1, 1, '2024-10-10', '2024-10-17', NULL),
(2, 2, 2, '2024-10-11', '2024-10-18', '2024-10-13'),
(3, 3, 3, '2024-10-12', '2024-10-19', NULL),
(4, 4, 4, '2024-10-13', '2024-10-20', '2024-10-17'),
(5, 5, 5, '2024-10-14', '2024-10-21', NULL),
(6, 6, 6, '2024-10-15', '2024-10-22', '2024-10-20'),
(7, 7, 7, '2024-10-16', '2024-10-23', NULL),
(8, 8, 8, '2024-10-17', '2024-10-24', '2024-10-28'),
(9, 9, 9, '2024-10-18', '2024-10-25', NULL),
(10, 5, 10, '2024-10-19', '2024-10-26', NULL),
(11, 11, 1, '2024-10-20', '2024-10-27', '2024-10-25'),
(12, 7, 2, '2024-10-21', '2024-10-28', NULL),
(13, 8, 3, '2024-10-22', '2024-10-29', NULL),
(15, 1, 5, '2024-10-24', '2024-10-31', NULL),
(16, 4, 6, '2024-10-25', '2024-11-01', NULL),
(17, 9, 7, '2024-10-26', '2024-11-02', NULL);
```

### Correction

*Les données ont été insérées en utilisant les requêtes SQL fournies.*

### Exercice 3 : Sélectionner les Livres Disponibles

#### Énoncé

Écrivez une requête pour récupérer tous les livres disponibles.

### Correction

```
SELECT Titre FROM Livres WHERE Disponible = TRUE;
```

**Explication** : Cette requête sélectionne tous les titres des livres qui sont encore disponibles, en filtrant avec la colonne `Disponible`.

#### Variante avancée :

```
SELECT Titre,
       CASE
         WHEN Disponible = TRUE THEN 'Disponible'
         ELSE 'Non disponible'
       END AS Etat
FROM Livres;
```

### Exercice 4 : Trier les Livres par Date de Publication

#### Énoncé

Écrivez une requête pour récupérer les livres et les trier par date de publication, du plus ancien au plus récent.

### Correction

```
SELECT Titre, DatePublication FROM Livres ORDER BY DatePublication ASC;
```

**Explication** : Les livres sont triés en fonction de leur date de publication, du plus ancien au plus récent.

## Exercice 5 : Filtrer les Emprunts en Cours

### Énoncé

Écrivez une requête pour récupérer les emprunts dont la **DateRetourEffective** est encore NULL, ce qui signifie que le livre n'a pas encore été rendu.

### Correction

```
SELECT EmpruntID, LivreID, EmprunteurID, DateEmprunt
FROM Emprunts
WHERE DateRetourEffective IS NULL;
```

**Explication** : Cette requête récupère les emprunts où la date de retour effective est encore NULL, indiquant que le livre n'a pas encore été rendu.

## Exercice 6 : Calculer la Durée d'un Emprunt

### Énoncé

Écrivez une requête pour calculer la durée (en jours) entre la date d'emprunt et la date de retour effective pour chaque emprunt, et nommez cette nouvelle colonne **DureeEmprunt**.

Indice : utilisez la fonction **JULIANDAY** pour convertir la date en nombre de jours depuis une date particulière.

### Correction

```
SELECT EmpruntID,
       (JULIANDAY(DateRetourEffective) - JULIANDAY(DateEmprunt)) AS DureeEmprunt
FROM Emprunts
WHERE DateRetourEffective IS NOT NULL;
```

**Explication** : Cette requête calcule la durée (en jours) entre la date d'emprunt et la date de retour effective pour chaque emprunt, en créant une nouvelle colonne **DureeEmprunt**.

### Variante avancée :

```
SELECT EmpruntID,
       CASE
         WHEN DateRetourEffective IS NULL THEN 'Non retourné'
         ELSE CAST((JULIANDAY(DateRetourEffective) - JULIANDAY(DateEmprunt)) AS
                   TEXT)
       END AS DureeEmprunt
FROM Emprunts;
```

## Exercice 7 : Jointure sur les Livres et les Auteurs

### Énoncé

Écrivez une requête SQL pour afficher le titre des livres ainsi que le nom complet (Nom + Prénom) de leur auteur. Je veux afficher un livre même s'il n'a pas d'auteur connu, par contre je ne veux pas afficher les auteurs qui n'ont pas de livre dans la base.

### Correction

```
SELECT Livres.Titre, Auteurs.Nom, Auteurs.Prénom
FROM Livres
LEFT JOIN Auteurs ON Livres.AuteurID = Auteurs.AuteurID;
```

**Explication** : On utilise une jointure LEFT JOIN pour afficher les livres même si certains n'ont pas d'auteur associé. Les auteurs sans livre ne sont pas inclus.

**Variante avancée** :

```
SELECT L.Titre,  
       CASE  
         WHEN A.AuteurID IS NULL THEN 'Auteur inconnu'  
         ELSE A.Nom || ' ' || A.Prénom  
       END AS NomCompleet  
FROM Livres L  
LEFT JOIN Auteurs A ON L.AuteurID = A.AuteurID;
```

## Exercice 8 : Filtrer les Emprunteurs qui n'ont pas encore rendu de livres

### Énoncé

Utilisez une jointure pour afficher les informations des emprunteurs (Nom, Prénom, Email) qui n'ont pas encore rendu leurs livres (c'est-à-dire les emprunts où la date de retour effective est NULL).

### Correction

```
SELECT Emprunteurs.Nom, Emprunteurs.Prénom, Emprunteurs.Email FROM Emprunteurs  
LEFT JOIN Emprunts ON Emprunteurs.EmprunteurID = Emprunts.EmprunteurID  
WHERE Emprunts.DateRetourEffective IS NULL;
```

**Explication** : On utilise une jointure LEFT JOIN pour ne récupérer que les emprunteurs liés aux emprunts en cours (où la date de retour effective est encore NULL) et uniquement les emprunteurs pour lesquels il existe au moins un enregistrement dans la table Emprunts

**Variante avancée** qui affiche un emprunteur une seule fois :

```
SELECT DISTINCT E.Nom, E.Prénom,  
       CASE WHEN E.Email IS NULL THEN 'Non renseigné' ELSE E.Email END AS Email  
FROM Emprunts Em  
INNER JOIN Emprunteurs E ON Em.EmprunteurID = E.EmprunteurID  
WHERE Em.DateRetourEffective IS NULL;
```

## Exercice 9 : Nombre de Livres par Genre

### Énoncé

Écrivez une requête qui utilise une jointure pour afficher le nombre de livres par genre. Le résultat doit montrer le nom du genre et le nombre de livres associés.

### Correction

```
SELECT Genres.NomGenre, COUNT(Livres.LivreID) AS NombreLivres  
FROM Genres  
LEFT JOIN Livres ON Livres.GenreID = Genres.GenreID  
GROUP BY Genres.NomGenre;
```

**Explication** : La requête utilise GROUP BY pour compter le nombre de livres par genre. La jointure permet de relier les livres à leurs genres respectifs.



## Exercice 10 : Durée Moyenne d’Emprunt par Emprunteur

### Énoncé

Calculez la durée moyenne des emprunts pour chaque emprunteur, et affichez leur nom et prénom ainsi que la durée moyenne en jours.

### Correction

```
SELECT Emprunteurs.Nom, Emprunteurs.Prénom,  
       AVG(JULIANDAY(DateRetourEffective) - JULIANDAY(DateEmprunt)) AS  
         DureeMoyenne  
FROM Emprunts  
INNER JOIN Emprunteurs ON Emprunts.EmprunteurID = Emprunteurs.EmprunteurID  
WHERE DateRetourEffective IS NOT NULL  
GROUP BY Emprunteurs.EmprunteurID;
```

**Explication** : Cette requête calcule la durée moyenne des emprunts pour chaque emprunteur en utilisant AVG et en s’assurant que seuls les emprunts rendus (avec une `DateRetourEffective` non nulle) sont pris en compte.

## Exercice 11 : Jointure avec Emprunteurs, Livres, et Genres

### Énoncé

Affichez le nom et prénom de chaque emprunteur, le titre du livre emprunté et le genre de ce livre. Je veux voir tous les livres, même ceux qui n’ont pas été empruntés, ainsi que tous les emprunteurs, même s’ils n’ont pas encore emprunté de livre, et tous les genres, même s’il n’existe pas de livre pour ces genres.

### Correction

```
SELECT Emprunteurs.Nom, Emprunteurs.Prénom, Livres.Titre, Genres.NomGenre  
FROM Emprunteurs  
FULL OUTER JOIN Emprunts ON Emprunteurs.EmprunteurID = Emprunts.EmprunteurID  
FULL OUTER JOIN Livres ON Emprunts.LivreID = Livres.LivreID  
FULL OUTER JOIN Genres ON Livres.GenreID = Genres.GenreID;
```

**Explication** : Une jointure FULL OUTER JOIN permet d’afficher toutes les lignes des emprunteurs, livres et genres, même s’il n’y a pas de correspondance directe dans les autres tables.

## Exercice 12 : Livres les Plus Empruntés

### Énoncé

Écrivez une requête SQL pour trouver les trois livres les plus empruntés. Affichez leur titre et le nombre d’emprunts.  
Indice : ajoutez LIMIT 3 à la fin d’une requête pour ne garder que les 3 premières lignes.

### Correction

```
SELECT Livres.Titre, COUNT(Emprunts.EmpruntID) AS NombreEmprunts  
FROM Emprunts  
INNER JOIN Livres ON Emprunts.LivreID = Livres.LivreID  
GROUP BY Livres.LivreID  
ORDER BY NombreEmprunts DESC  
LIMIT 3;
```

**Explication** : Cette requête affiche les trois livres les plus empruntés en utilisant GROUP BY pour regrouper les emprunts par livre, et ORDER BY pour les classer par nombre d’emprunts en ordre décroissant.

### Exercice 13 : Nombre de Livres Empruntés par Emprunteur

#### Énoncé

Écrivez une requête SQL pour afficher le nombre total de livres empruntés par chaque emprunteur. Le résultat doit inclure les emprunteurs qui n'ont jamais emprunté de livre.

#### Correction

```
SELECT Emprunteurs.Nom, Emprunteurs.Prénom, COUNT(Emprunts.EmpruntID) AS  
       NombreLivres  
FROM Emprunteurs  
LEFT JOIN Emprunts ON Emprunteurs.EmprunteurID = Emprunts.EmprunteurID  
GROUP BY Emprunteurs.EmprunteurID;
```

**Explication** : La jointure `LEFT JOIN` permet d'inclure tous les emprunteurs, même ceux qui n'ont jamais emprunté de livre. On compte ensuite le nombre d'emprunts par emprunteur avec `COUNT`.

### Exercice 14 : Livres Jamais Empruntés

#### Énoncé

Écrivez une requête SQL pour afficher la liste des livres qui n'ont jamais été empruntés.

#### Correction

```
SELECT Livres.Titre  
FROM Livres  
LEFT JOIN Emprunts ON Livres.LivreID = Emprunts.LivreID  
WHERE Emprunts.EmpruntID IS NULL;
```

**Explication** : Cette requête utilise une jointure `LEFT JOIN` pour inclure tous les livres, puis filtre avec `WHERE` pour ne garder que ceux qui n'ont jamais été empruntés (c'est-à-dire ceux sans correspondance dans la table des emprunts).

### Exercice 15 : Nombre d'Emprunteurs par Auteur

#### Énoncé

Écrivez une requête SQL pour afficher le nombre total d'emprunteurs pour chaque auteur. Le résultat doit inclure les auteurs dont aucun livre n'a été emprunté.

#### Correction

```
SELECT A.Nom, A.Prénom, COUNT(DISTINCT E.EmprunteurID) AS NombreEmprunteurs  
FROM Auteurs A  
LEFT JOIN Livres L ON A.AuteurID = L.AuteurID  
LEFT JOIN Emprunts E ON L.LivreID = E.LivreID  
GROUP BY A.AuteurID;
```

**Explication** : On compte le nombre d'emprunteurs distincts pour chaque auteur en utilisant `COUNT(Emprunts.EmprunteurID)`, ce qui permet de comptabiliser chaque emprunteur une seule fois par auteur, même s'il a emprunté plusieurs livres de cet auteur. Les auteurs sans livres empruntés sont également inclus grâce à la jointure `LEFT JOIN`. Même si un auteur n'a aucun livre (ou aucun livre emprunté), il sera affiché avec un compte à 0.