

PROJET PRATIQUE SGBD MySQL

Master Data Science pour l'Economie et la Finance

Suject:

Projet Pratique SGBD MySQL



Réalisé par:

- YAMLAHI ALAMI Siham

Sous l'encadrement de:

- Pr. BENNANI ANAS

Apogée :

20000117

Année Universitaire : 2023/2024

Table des matières

TABLE DES MATIERES	1
INTRODUCTION GENERALE.....	3
ATELIER 1 : DEVELOPPER UNE BASE DE DONNEES	4
1. INTRODUCTION	5
2. CONFIGURATION DE L'ENVIRONNEMENT DE BASE DE DONNEES.....	5
3. CREATION DE LA TABLE EMPLOYES	5
4. CREATION DE LA TABLE CLIENTS	6
5. CREATION DE LA TABLE COMMANDE	7
5.1. <i>Création de la table.....</i>	7
5.2. <i>Modification de la table pour ajouter une clé étrangère.....</i>	7
5.3. <i>Sélection de la contrainte de clé étrangère</i>	7
5.4. <i>Ajout de la contrainte de clé étrangère</i>	8
5.5. <i>Suppression de la contrainte de clé étrangère</i>	8
5.6. <i>Affichage de la structure de la table Commande</i>	8
5.7. <i>Insertion dans la table Commande.....</i>	9
6. COMBINAISON DES TABLES CLIENTS ET COMMANDE	9
7. CONCLUSION.....	10
ATELIER 2 : DEVELOPPER LE SCHEMA D'UNE BASE DE DONNEES.....	11
1. INTRODUCTION	12
SECTION 1 : CONFIGURATION DE LA BASE DE DONNEES.....	12
1. CREATION DE LA BASE DE DONNEES.....	12
SECTION 2 : CONCEPTION DES TABLES ET DEFINITION DES RELATIONS	12
1. CREATION DE LA TABLE ÉTUDIANT	12
1.1. <i>Modification de la colonne « Ville »</i>	12
1.2. <i>Ajout de la contrainte de Clé Primaire.....</i>	13
1.3. <i>Ajout de la contrainte de vérification de l'âge.....</i>	13
2. CREATION DE LA TABLE LIVRE AVEC CONTRAINTES.....	13
3. CREATION DE LA TABLE AUTEUR	14
4. CREATION DE LA TABLE ÉDITEUR	14
5. AFFICHAGE DES STRUCTURES DE TABLE POUR VERIFICATION.....	14
5.1. <i>Table « Auteur »</i>	14
5.2. <i>Table « Editeur ».....</i>	15
6. AJOUT DE CONTRAINTES DE CLE ÉTRANGERE A LA TABLE LIVRE.....	15
6.1. <i>Ajout de la contrainte de clé étrangère « NumAuteur »</i>	15
6.2. <i>Ajout de la contrainte de clé étrangère « NumEditeur ».....</i>	15
6.3. <i>Consultation des contraintes de Clé Étrangère</i>	16
7. CREATION DE LA TABLE THEME.....	16
8. CREATION DE LA TABLE PRET.....	17
8.1. <i>Contrainte unique pour prêt des livres</i>	17
8.2. <i>Contrainte unique pour statut de rendu</i>	17
8.3. <i>Contrainte de vérification sur les dates de retour</i>	17
8.4. <i>Affichage des Index de la Table Prêt.....</i>	18
SECTION 3 : INSERTION DE DONNEES INITIALES.....	18
9. INSERTION DANS LA TABLE ETUDIANT	18
10. INSERTION DANS LA TABLE AUTEUR	18
11. INSERTION DANS LA TABLE EDITEUR	19
12. INSERTION DANS LA TABLE LIVRE.....	19
13. INSERTION DANS LA TABLE PRET.....	19
14. INSERTION DANS LA TABLE THEME.....	19
15. REPONSES AUX QUESTIONS DIRECTES	19
16. CONCLUSION.....	20
ATELIER 3 : INTERROGER LA BASE DE DONNEES « BIBLIO_SIHAME »	21
1. INTRODUCTION	22

SECTION 1 : REMPLISSAGE DES TABLES.....	22
1. INSERTION DES ENREGISTREMENTS DANS LA TABLE AUTEUR	22
2. INSERTION DES ENREGISTREMENTS DANS LA TABLE THEME.....	22
3. INSERTION DES ENREGISTREMENTS DANS LA TABLE EDITEUR.....	22
4. INSERTION DES ENREGISTREMENTS DANS LA TABLE ETUDIANT.....	23
5. INSERTION DES ENREGISTREMENTS DANS LA TABLE LIVRE.....	23
6. INSERTION DES ENREGISTREMENTS DANS LA TABLE PRET.....	24
SECTION 2 : REQUETES EN MODE CREATION.....	24
1. AFFICHER LE NOM, PRENOM, AGE ET VILLE DES ETUDIANTS DONT L'AGE EST SUPERIEUR A 21	24
2. AFFICHER LES TITRES, LES AUTEURS ET LA DATE DE PRET DES LIVRES NON ENCORE RENDUS.....	25
3. CREATION DE LA TABLE PROFESSEUR :	25
4. AJOUT DE LA CONTRAINTE UNIQUE SUR NOM ET PRENOM	25
5. AJOUT DE LA COLONNE EMAIL.....	26
6. MODIFICATION DE LA TAILLE DE LA COLONNE EMAIL	26
7. SUPPRESSION DE LA COLONNE EMAIL.....	26
8. SUPPRESSION DE LA TABLE PROFESSEUR	26
9. INSERTION DANS LA TABLE ETUDIANT	27
9.1. <i>Insertion de nouveaux étudiants.....</i>	27
9.2. <i>Insertion de nouveaux étudiants sans téléphone.....</i>	27
9.3. <i>Mise à jour des villes pour les étudiants de Casablanca :.....</i>	27
9.4. <i>Ajout d'une année à l'âge des étudiants dont le numéro est supérieur à 1000</i>	27
9.5. <i>Suppression des étudiants dont le numéro est supérieur à 1000</i>	28
10. EXTRACTION DE DONNEES	28
11. CONCLUSION	39
CONCLUSION GENERALE	40

Introduction générale

Dans le cadre de notre module de gestion de bases de données relationnelles, nous avons réalisé une série d'ateliers visant à renforcer nos compétences en conception, manipulation et interrogation de bases de données. Chaque atelier a abordé des aspects spécifiques de la gestion de bases de données en utilisant MySQL, permettant de couvrir une large gamme de sujets allant de la création de bases de données et de tables, à l'insertion et la mise à jour de données, ainsi qu'à l'exécution de requêtes complexes pour extraire des informations pertinentes. Ce rapport compile les travaux effectués durant les trois ateliers, en détaillant les commandes SQL utilisées et en fournissant des explications sur les résultats obtenus.

Atelier 1 : Création et Manipulation de la Base de Données

L'objectif de l'atelier 1 était de créer une base de données **TP_DSEF_yamlahialami** et d'y ajouter plusieurs tables (**Employes**, **Clients**, **Commandes**) avec des clés primaires et des contraintes de clés étrangères pour garantir l'intégrité référentielle. Nous avons également inséré des données dans ces tables et utilisé des commandes pour modifier la structure des tables, notamment en ajoutant et supprimant des contraintes.

Atelier 2 : Conception et Population de la Base de Données Bibliothèque

L'atelier 2 s'est concentré sur la création d'une base de données **Biblio_Sihame**, incluant plusieurs tables (**Etudiant**, **Livre**, **Auteur**, **Editeur**, **Theme**, **Pret**). Chaque table a été soigneusement conçue avec des clés primaires et des contraintes pour assurer la cohérence des données. Des enregistrements ont été insérés pour tester la structure de la base de données, et des contraintes supplémentaires ont été ajoutées pour garantir l'unicité et la validité des enregistrements.

Atelier 3 : Interrogation de la Base de Données

L'atelier 3 était dédié à l'interrogation de la base de données **Biblio_Sihame**. Nous avons exécuté diverses requêtes SQL pour extraire des informations spécifiques, telles que les livres empruntés par les étudiants, les auteurs et les éditeurs de ces livres, ainsi que les thèmes des livres. Cet atelier nous a permis de comprendre comment utiliser des jointures et des conditions complexes pour obtenir des résultats précis.

Atelier 1 : Développer une base de données

1. Introduction

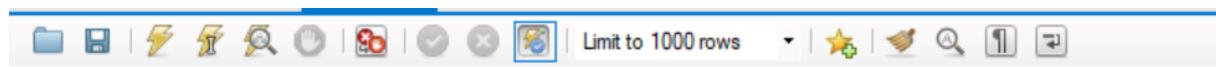
Dans le cadre du premier atelier de notre cours de Data Science Pour l'Économie et Finance, nous avons été chargés de développer une base de données pour une entreprise fictive. Cet atelier met l'accent sur la création de structures de base de données en utilisant MySQL, un système de gestion de base de données relationnelle. Ce rapport décrit chaque étape du processus, de la création de la base de données à la configuration des relations entre les tables.

2. Configuration de l'Environnement de Base de Données

Étape 1: Création de la Base de Données

La première étape du développement de notre base de données consiste à créer un nouvel environnement de base de données dans MySQL. Voici les étapes suivies :

- Requête SQL Utilisée:



```
1
2 •  create Database TP_DSEF_yamlahialami;
3 •  USE TP_DSEF_yamlahialami;
4
```

- Explication:

La commande CREATE DATABASE est utilisée pour initialiser une nouvelle base de données nommée « **TP_DSEF_yamlahialami** », qui hébergera toutes les tables nécessaires pour la gestion de l'entreprise. La commande USE est ensuite employée pour sélectionner cette base comme cible pour toutes les opérations subséquentes.

3. Crédit de la Table Employés

Étape 2: Définition de la table

- Requête SQL de création:

```
5 •  create table Employes (
6     NumEmploye int auto_increment primary key,
7     Nom char(35),
8     Prénom char(35),
9     Tél char(13)
10    );
```

- Explication:

Cette table est conçue pour stocker des informations critiques sur les employés. « **NumEmploye** » est défini comme une clé primaire avec « **AUTO_INCREMENT** » pour garantir qu'à chaque insertion d'un nouvel employé, un identifiant unique est généré automatiquement, éliminant le risque de doublons. Les types « **VARCHAR** » pour les noms et numéros de téléphone permettent une flexibilité dans la longueur des entrées, adaptée à des données personnelles variées.

- Requête SQL d'insertion:

```
12 • insert into Employes (NumEmploye, Nom ,Prénom , Tél)  
13     values  
14         (15 , "yamlahi" , "Sanae" , "0690088237"),  
15         (1 , "Radi" , "FATIMA" , "0626799777");
```

- Explication:

- « Insert into Employes »: Cette commande ajoute des données dans la table Employes.
- « (Nom, Prenom, Tel) VALUES ('Durand', 'Marie', '0123456789') »: Spécifie les valeurs à insérer pour chaque colonne listée

4. Création de la Table Clients

Étape 3: Configuration de la table

- Requête SQL de création:

```
19 • create table Clients(  
20     RefClient char(20) primary key,  
21     NomSociete char(50),  
22     ville char (30),  
23     CodePostale char(10)  
24 );
```

- Explication:

Cette commande établit une table Clients pour enregistrer les informations sur les clients.

- « **RefClient char(20) primary key** »: Similaire à « **NumEmploye** », c'est un identifiant unique pour chaque client qui s'incrémente automatiquement.
- « **NomSociete varchar(100)**, **Ville varchar(100)**, et **CodePostal varchar(10)** »: Ces champs permettent de stocker le nom de la société, la ville, et le code postal respectivement, avec des longueurs maximales adaptées pour ces informations.

- Requête SQL d'insertion:

```
26 • insert into Clients (RefClient, NomSociete , ville , CodePostale)  
27     values  
28         ("C1" , "Acom" , "Tangier" , "9000"),  
29         ("C2" , "B2C" , "CASA" , "40000"),  
30         ("C3" , "Tcom" , "Rabat" , "40000");
```

- Explication:

Ces commandes ajoutent des enregistrements dans la table Clients, spécifiant les valeurs pour chaque colonne correspondante.

5. Création de la Table Commande

Étape 4: Mise en place de la table

5.1. Crédation de la table

- Requête SQL:

```

37 •  create table Commandes (
38     RefCom int auto_increment primary key,
39     Societe char(35),
40     Date date,
41     Somme decimal(10,2),
42     TVA decimal(5,2)
43 );

```

- Explication:

Cette commande crée la table **Commandes** pour enregistrer les transactions effectuées par l'entreprise.

- « **RefCom int auto_increment primary key** »: Crée un identifiant unique pour chaque commande, qui s'incrémentera automatiquement, simplifiant le suivi des transactions.
- « **Societe char(35)** »: Prévoit un champ pour référencer l'identifiant du client (bien que le type doive être cohérent avec la clé primaire de la table **Clients**, ce qui sera corrigé par une commande ultérieure).
- « **Date date** »: Stocke la date de la commande.
- « **Somme decimal(10,2), TVA decimal(5,2)** »: Ces champs stockent respectivement le montant total de la commande et la taxe sur la valeur ajoutée, formatés pour inclure des valeurs décimales, assurant la précision financière des enregistrements.

5.2. Modification de la table pour ajouter une clé étrangère

- Requête SQL:

```

53 •  alter table Commandes
54      add column RefClient char,
55      add foreign key (RefClient) references Clients(RefClient);

```

- Explication:

Cette commande tente de lier la table « **Commandes** » à la table « **Clients** » via une clé étrangère. En ajoutant une nouvelle colonne « **RefClient** » à la table « **Commandes** » et en établissant une clé étrangère correcte qui référence « **RefClient** » dans la table « **Clients** ». Cela garantit que chaque commande peut être correctement associée à un client.

5.3. Sélection de la contrainte de clé étrangère

- Requête SQL :

```

46 •  Select constraint_name
47   from information_schema.KEY_COLUMN_USAGE
48  where table_name = 'Commandes' and table_schema = 'TP_DSEF_yamlahialami';
49

```

- **Explication:**

Cette requête est utilisée pour identifier le nom de la contrainte de clé étrangère associée à la table « **Commandes** ». Elle interroge le schéma de base de données pour trouver des détails sur l'utilisation des clés, ce qui est essentiel pour la gestion ou la modification des contraintes dans les structures de la base de données.

5.4. Ajout de la contrainte de clé étrangère

- **Requête SQL :**

```
59  
60 • alter table Commandes drop foreign key FK_CommandesClients;  
61
```

- **Explication:**

Cette commande ajoute explicitement une contrainte de clé étrangère à la table « **Commandes** » en utilisant un identifiant spécifique pour la contrainte, nommée « **FK_CommandesClients** ». Cette approche permet de nommer la contrainte de manière explicite, facilitant la gestion future des contraintes, comme pour des opérations de suppression ou de diagnostic des problèmes de références de clé étrangère.

5.5. Suppression de la contrainte de clé étrangère

- **Requête SQL :**

```
--  
60 • alter table Commandes drop foreign key FK_CommandesClients;  
--
```

- **Explication:**

Cette commande supprime la contrainte de clé étrangère « **FK_CommandesClients** » de la table « **Commandes** ». La suppression d'une contrainte peut être nécessaire pour des modifications de structure, des corrections de design ou pour préparer des modifications de données qui autrement violeraient l'intégrité référentielle.

5.6. Affichage de la structure de la table Commande

- **Requête SQL :**

```
61  
62 • show create table Commandes;  
63
```

- **Explication:**

Cette commande affiche la définition SQL complète pour la table « **Commandes** », y compris tous les détails de la création de la table. Cela est utile pour vérifier les attributs de la table, comme les types de données, les contraintes, les clés primaires et étrangères, et pour assurer que les modifications récentes ont été appliquées correctement.

- **Résultat:**

The screenshot shows the MySQL Workbench interface with the 'Form Editor' tab selected. On the left, there's a tree view labeled 'Create Table' under 'Table: Commandes'. The main pane displays the SQL code for creating the 'Commandes' table:

```

CREATE TABLE `commandes` (
  `RefCom` int NOT NULL AUTO_INCREMENT,
  `Societe` char(35) DEFAULT NULL,
  `Date` date DEFAULT NULL,
  `Somme` decimal(5,2) DEFAULT NULL,
  `TVA` decimal(5,2) DEFAULT NULL,
  `RefClient` char(1) DEFAULT NULL,
  PRIMARY KEY (`RefCom`),
  KEY `Societe` (`Societe`),
  KEY `FK_CommandesClients` (`RefClient`),
  CONSTRAINT `commandes_ibfk_1` FOREIGN KEY (`RefClient`) REFERENCES `clients` (`RefClient`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

```

On the right side of the interface, there are several tabs: 'Result Grid', 'Form Editor' (which is currently active), 'Field Types', 'Query Stats', and 'Execution Plan'.

5.7. Insertion dans la table Commande

- **Requête SQL :**

```

70 • INSERT INTO Commandes
71   (RefCom, Societe, Date, Somme, TVA, RefClient)
72   VALUES
73     ('CMD101', 'MarocTelecom', '2023-05-01', 1200.00, 24.00, 'C1'),
74     ('CMD102', 'OCP', '2023-05-02', 1600.50, 32.00, 'C2'),
75     ('CMD103', 'MarocTelecom', '2023-05-03', 800.75, 16.00, 'C1'),
76     ('CMD104', 'RAM', '2023-05-04', 2100.00, 42.00, 'C3'),
77     ('CMD105', 'OCP', '2023-05-05', 350.00, 7.00, 'C2');

```

- **Explication:**

Cette commande insère plusieurs enregistrements dans la table commandes, chaque ligne correspondant à une transaction ou commande passée par différentes entreprises marocaines.

- **RefCom:** Référence de la commande. Chaque commande est identifiée par un code unique, comme 'CMD101', facilitant le suivi et la référence dans les registres ou les systèmes de gestion.
- **Societe:** Nom de l'entreprise cliente. Les exemples incluent 'MarocTelecom' et 'OCP', qui sont de grandes entreprises bien connues au Maroc.
- **Date:** La date à laquelle la commande a été passée. Le format utilisé est AAAA-MM-JJ, conforme aux normes internationales et facilement utilisables pour le tri et la comparaison des dates.
- **Somme:** Le montant total de la commande en dirhams marocains. Ce montant représente le total avant taxes ou autres frais.
- **TVA:** La taxe sur la valeur ajoutée applicable à la commande. C'est une taxe gouvernementale appliquée sur les biens et services au Maroc, ici exprimée en pourcentage du montant total de la commande.
- **RefClient:** Référence du client, correspondant à un identifiant unique pour chaque client, comme 'C1', 'C2', ou 'C3'. Cette référence doit correspondre à un identifiant existant dans la table Clients pour maintenir l'intégrité référentielle entre les tables.

6. Combinaison des Tables Clients et Commande

- **Requête SQL :**

```

64 • select c.NomSociete, o.RefCom, o.Societe, o.Date, o.Somme, o.TVA
65   from Clients c
66   join Commandes o on c.RefClient = o.RefClient
67   order by c.NomSociete, o.Date;

```

- **Explication:**

Cette requête SQL effectue une jointure entre les tables « **Clients** » et « **Commandes** » pour extraire et afficher des informations combinées des deux tables. Les détails suivants sont extraits :

- **NomSociete** : Le nom de la société du client.
- **RefCom** : Le numéro de référence de la commande.
- **Societe**, Date, Somme, TVA : Détails spécifiques de chaque commande.

La jointure est faite sur la colonne RefClient, assurant que les données retournées correspondent aux clients et à leurs commandes respectives. Les résultats sont triés par le nom de la société et la date de la commande, ce qui facilite l'analyse des données commerciales et financières.

- **Résultat:**

NomSociete	RefCom	Societe	Date	Somme	TVA
Acom	CMD101	MarocTelecom	2023-05-01	1200.00	24.00
Acom	CMD103	MarocTelecom	2023-05-03	800.75	16.00
B2C	CMD102	OCP	2023-05-02	1600.50	32.00
B2C	CMD105	OCP	2023-05-05	350.00	7.00
Tcom	CMD104	RAM	2023-05-04	2100.00	42.00

7. Conclusion

Cet atelier sur la gestion des bases de données MySQL a consolidé notre compréhension et nos compétences dans la création, la modification, et la gestion des bases de données dans un contexte professionnel. Nous avons abordé des aspects essentiels tels que la structuration des données, l'intégrité référentielle, et l'optimisation des requêtes, ce qui est crucial pour assurer l'efficacité et la sécurité des systèmes d'information.

Atelier 2 : Développer le schéma d'une base de données

1. Introduction

L'Atelier 2 a concentré ses efforts sur le développement d'un schéma pour une base de données destinée à une bibliothèque universitaire, impliquant la création de tables spécifiques pour les étudiants, livres, prêts, auteurs et éditeurs, ainsi que l'établissement de relations entre elles. Ce rapport explore chaque Requête SQL utilisée, fournissant des explications détaillées pour enrichir la compréhension et le savoir-faire technique.

Section 1 : Configuration de la Base de Données

1. Création de la base de données

- Requête SQL :

```
2 • CREATE DATABASE Biblio_Sihame;
3 • USE Biblio_Sihame;
```

- Explication :

Ces commandes initient la création de la base de données spécifique « **Biblio_Sihame** » et sélectionnent cette base pour les opérations subséquentes. Cela garantit que toutes les tables et manipulations de données sont localisées dans le bon contexte.

Section 2 : Conception des tables et définition des relations

1. Création de la table Etudiant

- Requête SQL :

```
5 • CREATE TABLE Etudiant(
6     NumEtudiant INT,
7     Nom VARCHAR(30) NOT NULL,
8     Prenom VARCHAR(30) NOT NULL,
9     Age INT,
10    Ville VARCHAR(20),
11    Tel VARCHAR(20)
12 );
```

- Explication

Cette commande crée la table « **Etudiant** », conçue pour recueillir les informations essentielles des étudiants. Les champs comprennent un numéro d'étudiant (**NumEtudiant**), le nom, le prénom, l'âge, la ville de résidence, et le numéro de téléphone. Les champs « **Nom** » et « **Prenom** » sont marqués comme « **NOT NULL** », ce qui signifie qu'ils doivent contenir des valeurs pour chaque enregistrement, assurant ainsi que des données essentielles ne sont pas omises.

1.1. Modification de la colonne « Ville »

- Requête SQL :

```
14 • ALTER TABLE Etudiant
15     MODIFY Ville VARCHAR(30) NOT NULL;
```

- **Explication :**

Cette modification ajuste la colonne « **Ville** » de la table « **Etudiant** » pour augmenter sa capacité de 20 à 30 caractères et pour la rendre obligatoire (**NOT NULL**). Ce changement est motivé par le besoin de permettre des noms de ville plus longs et d'assurer qu'une donnée de localisation est toujours fournie, ce qui peut être crucial pour des communications ou des statistiques précises.

1.2. Ajout de la contrainte de Clé Primaire

- **Requête SQL :**

```
16
17 • ALTER TABLE Etudiant ADD CONSTRAINT pk PRIMARY KEY (NumEtudiant);
18
```

- **Explication :**

Cette commande ajoute une contrainte de clé primaire sur la colonne « **NumEtudiant** », garantissant que chaque valeur dans cette colonne est unique et n'est jamais **NULL**. Le numéro d'étudiant agit ainsi comme un identifiant unique pour chaque étudiant, essentiel pour lier de manière fiable les étudiants à d'autres enregistrements dans des tables connexes (comme les prêts de livres).

1.3. Ajout de la contrainte de vérification de l'âge

- **Requête SQL :**

```
93 • alter table Etudiant
94     add constraint CHK_Age check (Age between 15 and 30);
```

- **Explication :**

Cette contrainte de vérification s'assure que l'âge des étudiants est compris entre 15 et 30 ans, garantissant que seules les données d'âge valides et现实的 sont enregistrées dans la base de données.

2. Création de la table Livre avec contraintes

- **Requête SQL :**

```
18 • CREATE TABLE Livre(
19     NumLivre VARCHAR(10) PRIMARY KEY,
20     Titre VARCHAR(50) NOT NULL,
21     NumAuteur VARCHAR(10) NOT NULL,
22     NumEditeur VARCHAR(10) NOT NULL,
23     NumTheme VARCHAR(10) NOT NULL,
24     DateEdition DATE NOT NULL
25 );
```

- **Explication :**

Cette commande crée la table « **Livre** » qui stocke des informations sur les livres. **NumLivre** est utilisé comme clé primaire. Les champs **Titre**, **NumAuteur**, **NumEditeur**, **NumTheme**, et **DateEdition** sont requis pour garantir que toutes les informations essentielles sur chaque livre sont complètes et correctement enregistrées.

3. Création de la table Auteur

- Requête SQL :

```
27 • CREATE TABLE Auteur(
28     NumAuteur VARCHAR(10) PRIMARY KEY,
29     Nom VARCHAR(30) NOT NULL,
30     Adresse VARCHAR(50)
31 );
```

- Explication :

Cette commande établit la table « **Auteur** » avec un **NumAuteur** comme clé primaire pour identifier de manière unique chaque auteur. Le champ **Nom** est obligatoire, tandis que **Adresse** est facultative, permettant de stocker des informations supplémentaires sur chaque auteur de manière flexible.

4. Création de la table Editeur

- Requête SQL :

```
33 • CREATE TABLE Editeur(
34     NumEditeur VARCHAR(10) PRIMARY KEY,
35     Nom VARCHAR(30) NOT NULL,
36     Adresse VARCHAR(50)
37 );
```

- Explication :

Similaire à la table « **Auteur** », cette table « **Editeur** » enregistre les détails sur les éditeurs, incluant un identifiant unique, un nom requis, et une adresse pour chaque éditeur.

5. Affichage des Structures de Table pour Vérification

5.1. Table « Auteur »

- Requête SQL :

```
38
39 • SHOW CREATE TABLE Auteur;
40
```

- Explication :

Cette commande affiche la structure de création de la table « **Auteur** », permettant de vérifier les types de données, les contraintes, et d'autres propriétés de la table.

- Résultat :

5.2. Table « Editeur »

- Requête SQL :

40

41 • SHOW CREATE TABLE Editeur;

- Explication :

Cette commande affiche la structure de création de la table « **Editeur** », permettant de vérifier les types de données, les contraintes, et d'autres propriétés de la table.

- Résultat :

```

Form Editor | Navigate: ⟲ ⟳ 1 / 1 ⟷ ⟸ | □
Table: Editeur
CREATE TABLE `editeur` (
  `NumEditeur` varchar(10) NOT NULL,
  `Nom` varchar(30) NOT NULL,
  `Adresse` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`NumEditeur`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
Create Table:

```

6. Ajout de Contraintes de Clé Étrangère à la Table Livre

6.1. Ajout de la contrainte de clé étrangère « NumAuteur »

- Requête SQL :

```

43 • ALTER TABLE Livre
44   ADD CONSTRAINT FK_Livre_Auteur
45     FOREIGN KEY (NumAuteur) REFERENCES Auteur (NumAuteur)
46     ON DELETE CASCADE
47     ON UPDATE CASCADE;

```

- Explication :

Cette contrainte de clé étrangère lie **NumAuteur** dans la table « **Livre** » à **NumAuteur** dans la table « **Auteur** ». Les actions **ON DELETE CASCADE** et **ON UPDATE CASCADE** garantissent que les modifications ou suppressions des auteurs se répercutent sur les livres correspondants, maintenant l'intégrité des données.

6.2. Ajout de la contrainte de clé étrangère « NumEditeur »

- Requête SQL :

```

49 • ALTER TABLE Livre
50   ADD CONSTRAINT FK_Livre_Editeur
51     FOREIGN KEY (NumEditeur) REFERENCES Editeur (NumEditeur)
52     ON DELETE CASCADE
53     ON UPDATE CASCADE;

```

- **Explication :**

Cette commande établit une relation de clé étrangère entre **NumEditeur** dans la table « **Livre** » et **NumEditeur** dans la table « **Editeur** », avec des actions de cascade pour maintenir l'intégrité lors des mises à jour ou des suppressions.

6.3. Consultation des contraintes de Clé Étrangère

- **Requête SQL :**

```
55 •   SELECT
56       TABLE_NAME,
57       CONSTRAINT_NAME,
58       COLUMN_NAME,
59       REFERENCED_TABLE_NAME,
60       REFERENCED_COLUMN_NAME
61   FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
62   WHERE TABLE_NAME = 'Livre' AND TABLE_SCHEMA = 'Biblio_Sihame';
```

- **Explication :**

Cette requête extrait des informations détaillées sur toutes les contraintes de clé étrangère associées à la table « **Livre** », fournissant une vue d'ensemble des dépendances et des liens entre les tables au sein de la base de données.

- **Résultat :**

Result Grid					
	TABLE_NAME	CONSTRAINT_NAME	COLUMN_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
▶	livre	PRIMARY	NumLivre	NULL	NULL
	livre	FK_Livre_Auteur	NumAuteur	auteur	NumAuteur
	livre	FK_Livre_Editeur	NumEditeur	editeur	NumEditeur

7. Création de la table Theme

- **Requête SQL :**

```
64 •   CREATE TABLE Theme(
65       NumTheme VARCHAR(10) PRIMARY KEY,
66       IntituleTheme VARCHAR(20) NOT NULL
67   );
```

- **Explication :**

Cette table « **Theme** » est destinée à cataloguer les différents thèmes des livres avec un identifiant unique et un intitulé explicatif pour chaque thème, facilitant l'organisation et la recherche de livres par thème.

8. Création de la table Prêt

- Requête SQL :

```

69 • CREATE TABLE Pret(
70     NumEtudiant INT,
71     NumLivre VARCHAR(10),
72     DatePret DATE,
73     Rendu BOOLEAN DEFAULT FALSE,
74     DateRetour DATE,
75     PRIMARY KEY (NumEtudiant, NumLivre, DatePret),
76     CONSTRAINT FK_Pret_Etudiant FOREIGN KEY (NumEtudiant) REFERENCES Etudiant (NumEtudiant),
77     CONSTRAINT FK_Pret_Livre FOREIGN KEY (NumLivre) REFERENCES Livre (NumLivre)
78 );

```

- Explication :

La table « Prêt » enregistre les détails des prêts de livres aux étudiants, incluant la date de prêt, si le livre a été rendu, et la date de retour. La clé primaire est composite, incluant **NumEtudiant**, **NumLivre**, et **DatePret** pour garantir l'unicité de chaque prêt.

8.1. Contrainte unique pour prêt des livres

- Requête SQL :

```

80 • ALTER TABLE Pret
81     ADD CONSTRAINT UQ_Pret_UniqueDate
82     UNIQUE (NumLivre, DatePret);

```

- Explication :

Cette contrainte unique garantit qu'un livre ne peut être prêté qu'une seule fois à une date donnée, prévenant les conflits ou les erreurs de prêt multiples. Elle aide à assurer que les données de prêt restent fiables et cohérentes, en évitant la survenue de plusieurs enregistrements pour le même prêt à la même date.

8.2. Contrainte unique pour statut de rendu

- Requête SQL :

```

86 • ALTER TABLE Pret
87     ADD CONSTRAINT UQ_NumLivre_Rendu_DateRetour unique (NumLivre, Rendu, DateRetour);

```

- Explication :

Cette contrainte unique supplémentaire est conçue pour s'assurer que pour chaque combinaison de livre, de son statut de rendu, et de date de retour, il n'existe qu'une seule entrée dans la base de données. Elle est cruciale pour maintenir l'ordre et prévenir les doublons dans les enregistrements de retour, facilitant ainsi la gestion des livres prêtés et retournés.

8.3. Contrainte de vérification sur les dates de retour

- Requête SQL :

```

95 • alter table Pret
96     add constraint CHK_Pret_DateRetour
97     check (DateRetour is null or (DateRetour >= DatePret and Rendu));

```

- **Explication :**

Cette contrainte de vérification s'assure que si une date de retour est spécifiée, elle doit être postérieure à la date de prêt et que le livre est marqué comme rendu. Cette logique garantit la cohérence des données de prêt, en évitant les incohérences où un livre serait retourné avant même d'avoir été prêté ou marqué comme non rendu tout en ayant une date de retour.

8.4. Affichage des Index de la Table Prêt

- **Requête SQL :**

86

87 • `show index from Pret;`

88

- **Explication :**

Cette requête permet d'afficher les index définis sur la table « **Prêt** ». Les index sont des structures de données qui améliorent la vitesse des opérations de récupération de données sur une table de base de données.

- **Résultat :**

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
pret	0	PRIMARY	1	NumEtudiant	A	8	HULL	HULL		BTREE			YES	HULL
pret	0	PRIMARY	2	NumLivre	A	23	HULL	HULL		BTREE			YES	HULL
pret	0	PRIMARY	3	DatePret	A	24	HULL	HULL		BTREE			YES	HULL
pret	0	UQ_Pret_UniqueDate	1	NumLivre	A	15	HULL	HULL		BTREE			YES	HULL
pret	0	UQ_Pret_UniqueDate	2	DatePret	A	24	HULL	HULL		BTREE			YES	HULL
pret	0	UQ_NumLivre_Rendu_DateRetour	1	NumLivre	A	15	HULL	HULL	YES	BTREE			YES	HULL
pret	0	UQ_NumLivre_Rendu_DateRetour	2	Rendu	A	19	HULL	HULL	YES	BTREE			YES	HULL
pret	0	UQ_NumLivre_Rendu_DateRetour	3	DateRetour	A	24	HULL	HULL	YES	BTREE			YES	HULL

Section 3 : Insertion de données initiales

9. Insertion dans la table Etudiant

- **Requête SQL :**

99

100 • `insert into Etudiant (NumEtudiant, Nom, Prenom, Age, Ville, Tel) VALUES (1, 'alami', 'Sihame', 17, 'Tanger', '0676792260');`

101

- **Explication :**

Cette commande insère un enregistrement initial dans la table Etudiant pour un étudiant avec des détails spécifiques, permettant de vérifier le bon fonctionnement de la table et des contraintes associées.

10. Insertion dans la table Auteur

- **Requête SQL :**

102

103 • `INSERT INTO Auteur (NumAuteur, Nom, Adresse) VALUES ('NumAuteurE', 'Nom de l'Auteur', 'Adresse de l'Auteur');`

104

- **Explication :**

Cette commande ajoute un enregistrement initial dans la table « **Auteur** », enregistrant les détails d'un auteur pour tester la structure de la table et ses relations avec d'autres tables.

11. Insertion dans la table Editeur

- Requête SQL :

```
105
106 • INSERT INTO Editeur (NumEditeur, Nom, Adresse) VALUES ('NumEditeur', 'Nom de l'Éditeur', 'Adresse de l'Éditeur');
107
```

- Explication :

Cette commande insère un enregistrement initial dans la table « **Editeur** », ajoutant les détails d'un éditeur pour vérifier la structure de la table et les relations avec les autres tables.

12. Insertion dans la table Livre

- Requête SQL :

```
108
109 • INSERT INTO Livre (NumLivre, Titre, NumAuteur, NumEditeur, NumTheme, DateEdition) VALUES ('R11', 'Antigone', 'NumAuteurE', 'NumEditeur', 'NumTheme', '2020-01-01');
110
```

- Explication :

Cette commande ajoute un enregistrement initial dans la table « **Livre** », permettant de tester la structure de la table et les relations avec les tables « **Auteur** », « **Editeur** », et « **Theme** ».

13. Insertion dans la table Prêt

- Requête SQL :

```
113 • insert into Pret (NumEtudiant, NumLivre, DatePret, Rendu, DateRetour) values (1, 'S1', '2024-05-01', TRUE, '2024-04-30');
114 • insert into Pret (NumEtudiant, NumLivre, DatePret, Rendu, DateRetour) values (1, 'R11','2024-03-01', TRUE, '2024-04-30');
```

- Explication :

Ces commandes insèrent des enregistrements initiaux dans la table « **Prêt** », permettant de vérifier la cohérence et le bon fonctionnement des prêts de livres, ainsi que la validation des contraintes de clé étrangère et des dates de retour.

14. Insertion dans la table Theme

- Requête SQL :

```
115
116 • insert into Theme(NumTheme, IntituleTheme) values ("T001", "roman");
117
```

- Explication :

Cette commande ajoute un enregistrement initial dans la table « **Theme** », enregistrant un thème spécifique pour tester la structure de la table et ses relations avec la table « **Livre** ».

15. Réponses aux questions directes

- ❖ Question 8 : Quelle est la valeur par défaut de la propriété Null Interdit du champ Ville ?

La valeur par défaut de la propriété Null Interdit du champ Ville est Non.

❖ **Question 10 : Est-ce que vous pouvez entrer des enregistrements sans nom, prénom et ville ?**

Oui, il est possible d'entrer des enregistrements sans Nom, Prénom et Ville avant de définir les contraintes Null Interdit sur ces champs.

❖ **Question 14 : Quelle est la valeur de la propriété Null Interdit du champ Ville après modification ? Quelle est sa taille ?**

Après modification, la propriété Null Interdit du champ Ville est "Oui" et sa taille est de 30 caractères.

❖ **Question 61 : Essayez d'entrer des enregistrements de sorte d'avoir des doublons de (NumLivre, DatePrêt). Est-il possible de le faire ?**

Non, il n'est pas possible d'entrer des enregistrements avec des doublons de (NumLivre, DatePrêt) en raison de la contrainte d'unicité ajoutée.

16. Conclusion

Cet atelier a permis de :

- Créer une base de données et des tables interconnectées pour gérer les informations d'une bibliothèque universitaire.
- Mettre en place des contraintes pour assurer l'intégrité des données.
- Insérer des données initiales pour valider le fonctionnement du schéma.

Atelier 3 : Interroger la Base de Données « Biblio_Sihame »

1. Introduction

Dans cet atelier, nous avons appris à interroger la base de données « **Biblio_Sihame** » que nous avons créée précédemment. Nous avons rempli les tables avec des enregistrements et avons utilisé diverses requêtes SQL pour extraire et manipuler les données. Cet atelier nous a permis de comprendre comment interagir efficacement avec une base de données en utilisant des commandes SQL pour répondre à des questions spécifiques et obtenir des résultats précis.

Section 1 : Remplissage des Tables

1. Insertion des enregistrements dans la table Auteur

- Requête SQL :

```
3 • insert into Auteur (NumAuteur, Nom, Adresse) values  
4     ('H', 'Hakim', 'Rue 123'),  
5     ('M', 'Moujarrib', 'Rue 456'),  
6     ('N', 'Neferiti', 'Rue 789'),  
7     ('R', 'Ramsis', 'Rue 101'),  
8     ('T', 'Tafih', 'RUE 102');
```

- Explication :

Ces commandes insèrent des enregistrements initiaux dans la table « **Auteur** », ajoutant des détails sur plusieurs auteurs pour permettre la gestion et l'interrogation des livres dans la base de données.

2. Insertion des enregistrements dans la table Theme

- Requête SQL :

```
10 • insert into Theme (NumTheme, IntituleTheme) values  
11     ('Eco', 'Economie'),  
12     ('Info', 'Informatique'),  
13     ('Math', 'Mathématiques'),  
14     ('Div', 'Divers');
```

- Explication :

Ces commandes ajoutent des thèmes à la table « **Theme** », permettant la catégorisation des livres selon leurs sujets respectifs.

3. Insertion des enregistrements dans la table Editeur

- Requête SQL :

```
16 • insert into Editeur (NumEditeur, Nom, Adresse) values  
17     ('DAO', 'Dar Al-Oujoum', 'rue abc'),  
18     ('NP', 'Nul Part', 'Av DEF'),  
19     ('PLB', 'Pour les Bêtes', 'RUE GHI');
```

- **Explication :**

Ces enregistrements ajoutent des éditeurs à la table « **Editeur** », stockant les détails nécessaires pour référencer les livres édités par ces maisons d'édition.

4. Insertion des enregistrements dans la table Etudiant

- **Requête SQL :**

```
21 • insert into Etudiant (NumEtudiant, Nom, Prenom, Age, Tel, Ville) VALUES
22   (50, 'Kaslani', 'Kassoul', 28, '065555555', 'Tanger'),
23   (51, 'Kaslani', 'Kassoula', 27, '065555556', 'Tanger'),
24   (100, 'Abbassi', 'Abbass', 23, '070000607', 'Tanger'),
25   (101, 'Kaddouri', 'Kaddour', 24, '077777700', 'Chefchaouen'),
26   (102, 'Jallouli', 'Jalloul', 23, '066666660', 'Tétouan'),
27   (103, 'Ayyachi', 'Aicha', 22, NULL, 'Tétouan'),
28   (113, 'Slaoui', 'Salwa', 21, '060000001', 'Tanger'),
29   (202, 'Khaldouni', 'Khalid', 22, '060000002', 'Tanger'),
30   (309, 'Karimi', 'Karim', 20, '066600005', 'Casa'),
31   (310, 'Karimi', 'Karima', 20, NULL, 'Casa'),
32   (567, 'Moussaoui', 'Moussa', 21, '050070070', 'Tanger'),
33   (580, 'Moussi', 'Moussa', 22, NULL, 'Casa'),
34   (998, 'Moujtahida', 'Moujidda', 21, NULL, 'Tanger'),
35   (999, 'Moujtahid', 'Moujidd', 21, NULL, 'Tanger');
```

- **Explication :**

Ces commandes insèrent des enregistrements dans la table « **Etudiant** », ajoutant des informations détaillées sur chaque étudiant, y compris leur numéro, nom, prénom, âge, téléphone et ville.

5. Insertion des enregistrements dans la table Livre

- **Requête SQL :**

```
43 • insert into Livre (NumLivre, Titre, NumAuteur, NumEditeur, NumTheme, DateEdition) values
44   ('BD1', 'Comment avoir 20 en BD', 'R', 'NP', 'Info', '2015-01-01'),
45   ('BD2', 'Tout sur les BD', 'N', 'NP', 'Info', '2014-12-01'),
46   ('BD3', 'Maîtriser les BD', 'R', 'NP', 'Info', '2014-07-07'),
47   ('BD4', 'SGBD Relationnels', 'R', 'DAO', 'Info', '2014-01-01'),
48   ('BD5', 'SI et BD', 'N', 'DAO', 'Info', '2003-02-04'),
49   ('BD6', 'Les BD : Pour les nuls', 'R', 'NP', 'Info', '2014-01-01'),
50   ('EC01', 'L''économie du Maroc en l'an 3050', 'M', 'DAO', 'Eco', '2015-04-01'),
51   ('Math1', 'Algèbre', 'H', 'NP', 'Math', '2014-09-02'),
52   ('Math2', 'Analyse', 'H', 'NP', 'Math', '2014-08-02'),
53   ('Math3', 'Algèbre linéaire', 'H', 'DAO', 'Math', '2015-08-02'),
54   ('Math4', 'Aimer les Maths', 'M', 'NP', 'Math', '2014-08-04'),
55   ('SE1', 'Systèmes d\'exploitation', 'R', 'NP', 'Info', '2003-08-06'),
56   ('SE2', 'Maîtriser UNIX', 'R', 'DAO', 'Info', '2002-10-02'),
57   ('SE3', 'Tout sur les SE', 'N', 'NP', 'Info', '2001-08-07'),
58   ('TW1', 'Histoire', 'T', 'PLB', 'Div', NULL),
59   ('TW2', 'Personnes fameuses', 'T', 'PLB', 'Div', NULL),
60   ('TW3', 'Comment devenir un bon joueur en 5 jours et sans coach', 'T', 'PLB', 'Div', NULL);
```

- **Explication :**

Ces commandes insèrent des enregistrements dans la table « **Livre** », ajoutant des informations détaillées sur chaque livre, y compris leur numéro, titre, auteur, éditeur, thème et date d'édition.

6. Insertion des enregistrements dans la table Pret

- Requête SQL :

```
62 • INSERT INTO Pret (NumEtudiant, NumLivre, DatePret, Rendu, DateRetour) VALUES
63     (50, 'BD1', '2024-05-01', FALSE, NULL),
64     (51, 'ECO1', '2024-04-15', TRUE, '2024-05-01'),
65     (100, 'Math1', '2024-03-10', FALSE, NULL),
66     (101, 'BD2', '2024-02-20', TRUE, '2024-03-01'),
67     (102, 'SE1', '2024-01-15', TRUE, '2024-02-01'),
68     (103, 'TW1', '2024-05-10', FALSE, NULL), -- Prêt non rendu
69     (202, 'BD3', '2024-04-20', FALSE, NULL); -- Prêt non rendu
```

- Explication :

Cette commande SQL insère plusieurs enregistrements dans la table « **Pret** », qui enregistre les prêts de livres par des étudiants.

Section 2 : Requêtes en Mode Création

1. Afficher le nom, prénom, âge et ville des étudiants dont l'âge est supérieur à 21

- Requête SQL :

```
62 • SELECT Nom, Prenom, Age, Ville
63   FROM Etudiant
64   WHERE Age > 21
65   ORDER BY Nom ASC, Age DESC;
```

- Explication :

Cette requête sélectionne les étudiants dont l'âge est supérieur à 21 ans, affichant leur nom, prénom, âge et ville. Les résultats sont triés par nom en ordre croissant et par âge en ordre décroissant.

- Résultat :

Result Grid					Filter Rows:	Export:	Wrap Cell Content:
	Nom	Prenom	Age	Ville			
▶	Abbassi	Abbass	23	Tanger			
	Ayyachi	Aicha	22	Tétouan			
	Hammadi	Hamada	26	Casablanca			
	Jallouli	Jalloul	23	Tétouan			
	Kaddouri	Kaddour	24	Chefchaouen			
	Kaslani	Kassoul	28	Tanger			
	Kaslani	Kassoula	27	Tanger			
	Khaldouni	Khalid	22	Tanger			
	Mimouni	Mimoun	24	Casablanca			
	Moussi	Moussa	22	Casablanca			
	Sallami	Salma	27	Tanger			
	Tahiri	Tahir	25	Tanger			

2. Afficher les titres, les auteurs et la date de prêt des livres non encore rendus

- Requête SQL :

```

67 •   SELECT
68     A.Nom AS AuteurNom,
69     L.Titre AS TitreLivre,
70     P.DatePret
71   FROM
72     Auteur A
73   JOIN
74     Livre L ON A.NumAuteur = L.NumAuteur
75   JOIN
76     Pret P ON L.NumLivre = P.NumLivre
77   WHERE
78     P.Rendu = FALSE -- ou P.Rendu = 0, selon comment le booléen est stocké dans votre base de données
79   ORDER BY
80     A.Nom ASC,
81     P.DatePret DESC;

```

- Explication :

Cette requête sélectionne les titres des livres, les auteurs et la date de prêt des livres qui n'ont pas encore été rendus. Les résultats sont triés par nom de l'auteur en ordre croissant et par date de prêt en ordre décroissant.

- Résultat :

Result Grid			
	AuteurNom	TitreLivre	DatePret
▶	Hakim	Algèbre	2024-03-10
	Ramsis	Comment avoir 20 en BD	2024-05-01
	Ramsis	Maitriser les BD	2024-04-20
	Tafih	Histoire	2024-05-10

3. Création de la table Professeur :

- Requête LDD2 SQL :

```

93      -- RequêteLDD2
94 •   CREATE TABLE Professeur (
95     CIN numeric primary key,
96     Nom varchar(20) not null ,
97     Prénom varchar(20) not null,
98     Adresse text
99 );

```

- Explication :

Cette requête crée une table « **Professeur** » avec des champs pour le CIN (clé primaire), le nom, le prénom et l'adresse, en définissant les contraintes nécessaires pour chaque champ.

4. Ajout de la contrainte unique sur Nom et Prenom

- Requête LDD3 SQL :

```

101      -- RequêteLDD3
102 •   alter table Professeur
103     add constraint Unique_Nom_Prenom unique (Nom, Prénom);
104

```

- **Explication :**

Cette commande ajoute une contrainte unique à la table « **Professeur** », garantissant que deux professeurs ne peuvent pas avoir le même nom et prénom.

5. Ajout de la colonne Email

- **Requête LDD4 SQL :**

```
105      -- RequêteLDD4
106 •  alter table Professeur
107    add Email varchar(10);
```

- **Explication :**

Cette commande ajoute une nouvelle colonne **Email** à la table « **Professeur** ».

6. Modification de la taille de la colonne Email

- **Requête LDD5 SQL :**

```
111      -- RequêteLDD5
112 •  alter table Professeur
113    modify Email varchar(50);
114
```

- **Explication :**

Cette commande modifie la taille de la colonne **Email** dans la table « **Professeur** » pour permettre des adresses email plus longues.

7. Suppression de la colonne Email

- **Requête LDD6 SQL :**

```
115      -- RequêteLDD6
116 •  alter table Professeur
117    drop column Email;
```

- **Explication :**

Cette commande supprime la colonne **Email** de la table « **Professeur** ».

8. Suppression de la table Professeur

- **Requête LDD7 SQL :**

```
119      -- RequêteLDD7
120 •  drop table Professeur;
```

- **Explication :**

Cette commande supprime la table « **Professeur** » de la base de données.

9. Insertion dans la table Etudiant

9.1. Insertion de nouveaux étudiants

- Requête LMD1 SQL :

```
125    -- Requête LMD1
126 •  insert into Etudiant (NumEtudiant, Nom, Prenom, Age, Tel, Ville) values
127      (1001, 'Hammadi', 'Hamada', 25, '0611111111', 'Casa'),
128      (1002, 'Tahiri', 'Tahir', 24, '066666600', 'Tanger');
```

- Explication :

Cette commande insère de nouveaux enregistrements dans la table « **Etudiant** ».

9.2. Insertion de nouveaux étudiants sans téléphone

- Requête LMD2 SQL :

```
130    -- Requête LMD2
131 •  insert into Etudiant (NumEtudiant, Nom, Prenom, Age, Ville) values
132      (1003, 'Sallami', 'Salma', 26, 'Tanger'),
133      (1004, 'Mimouni', 'Mimoun', 23, 'Casa');
```

- Explication :

Cette commande insère de nouveaux enregistrements dans la table « **Etudiant** », sans les numéros de téléphone.

9.3. Mise à jour des villes pour les étudiants de Casablanca :

- Requête LMD3 SQL :

```
135    -- Requête LMD3
136 •  update Etudiant e
137    join (select NumEtudiant from Etudiant where Ville = 'Casa') as sub
138    ON e.NumEtudiant = sub.NumEtudiant
139    SET e.Ville = 'Casablanca';
```

- Explication :

Cette commande met à jour les enregistrements de la table « **Etudiant** », changeant **Casa** en **Casablanca**.

9.4. Ajout d'une année à l'âge des étudiants dont le numéro est supérieur à 1000

- Requête LMD4 SQL :

```
143    -- Requête LMD4
144 •  update Etudiant
145    set Age = Age + 1
146    where NumEtudiant > 1000;
```

- **Explication :**

Cette commande met à jour les enregistrements de la table « **Etudiant** », ajoutant une année à l'âge des étudiants dont le numéro est supérieur à 1000.

9.5. Suppression des étudiants dont le numéro est supérieur à 1000

- **Requête LMD4 SQL :**

```
149      -- Requête LMD5
150 •  delete from Etudiant
151      where NumEtudiant > 1000;
152
```

- **Explication :**

Cette commande supprime les enregistrements de la table « **Etudiant** » pour les étudiants dont le numéro est supérieur à 1000.

10. Extraction de Données

- **Requête 7: Afficher le nom, prénom et ville de l'étudiant "Moujtahid"**

```
155      -- Requête 7
156 •  select Nom, Prenom, Ville as Adresse
157      from Etudiant
158      where Nom = 'Moujtahid';
```

- **Explication :**

Cette requête sélectionne les enregistrements de la table « **Etudiant** » où le nom est "Moujtahid", affichant leur nom, prénom et ville.

- **Résultat :**

Result Grid				Filter Rows:	Export:	Wrap Cell Content:
	Nom	Prenom	Adresse			
▶	Moujtahid	Moujidd	Tanger			

- **Requête 8 : Afficher les titres des livres de l'auteur avec le numéro "R"**

```
160      -- Requête 8
161 •  select Titre
162      from Livre
163      where NumAuteur = 'R';
```

- **Explication :**

Cette requête sélectionne les titres des livres écrits par l'auteur ayant le numéro "R".

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Titre			
▶	Comment avoir 20 en BD			
	Maitriser les BD			
	SGBD Relationnels			
	Les BD : Pour les nuls			
	Systèmes d'exploitation			
	Maitriser UNIX			

- **Requête 9 : Afficher les titres des livres de l'auteur "Ramsis"**

```
165      -- Requête 9
166 •   select L.Titre
167     from Livre L
168     join Auteur A on L.NumAuteur = A.NumAuteur
169     where A.Nom = 'Ramsis';
```

- **Explication :**

Cette requête sélectionne les titres des livres écrits par l'auteur nommé "**Ramsis**".

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Titre			
▶	Comment avoir 20 en BD			
	Maitriser les BD			
	SGBD Relationnels			
	Les BD : Pour les nuls			
	Systèmes d'exploitation			
	Maitriser UNIX			

- **Requête 10 : Afficher le numéro de l'auteur du livre "Comment avoir 20 en BD"**

```
171      -- Requête 10
172 •   select NumAuteur
173     from Livre
174     where Titre = 'Comment avoir 20 en BD';
```

- **Explication :**

Cette requête sélectionne le numéro de l'auteur du livre intitulé "**Comment avoir 20 en BD**".

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	NumAuteur			
▶	R			

- **Requête 11 : Afficher le nom et l'adresse de l'auteur du livre "Comment avoir 20 en BD"**

```

176      -- Requête 11
177  •  select A.Nom, A.Adresse
178    from Livre L
179   join Auteur A on L.NumAuteur = A.NumAuteur
180  where L.Titre = 'Comment avoir 20 en BD';

```

- **Explication :**

Cette requête sélectionne le nom et l'adresse de l'auteur du livre intitulé "**Comment avoir 20 en BD**".

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Nom	Adresse		
▶	Ramsis	Rue 101		

- **Requête 12 : Afficher les livres de l'auteur "Ramsis" édités chez "Nul Part"**

```

182      -- Requête 12
183  •  select L.Titre
184    from Livre L
185   join Editeur E on L.NumEditeur = E.NumEditeur
186  where E.Nom = 'Nul Part' and L.NumAuteur = (select NumAuteur from Auteur where Nom = 'Ramsis');

```

- **Explication :**

Cette requête sélectionne les titres des livres écrits par l'auteur "**Ramsis**" et édités par "**Nul Part**".

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Titre			
▶	Comment avoir 20 en BD			
	Maitriser les BD			
	Les BD : Pour les nuls			
	Systèmes d'exploitation			

- **Requête 13 : Afficher les livres des auteurs "Ramsis" ou "Neferiti"**

```
188    -- Requête 13
189 •  select L.Titre
190   from Livre L
191  join Auteur A on L.NumAuteur = A.NumAuteur
192 where A.Nom in ('Ramsis', 'Neferiti');
```

- **Explication :**

Cette requête sélectionne les titres des livres écrits par les auteurs "Ramsis" ou "Neferiti".

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Titre			
▶	Tout sur les BD			
	SI et BD			
	Tout sur les SE			
	Comment avoir 20 en BD			
	Maitriser les BD			
	SGBD Relationnels			
	Les BD : Pour les nuls			
	Systèmes d'exploitation			
	Maitriser UNIX			

- **Requête 14 : Afficher les thèmes des livres édités chez "Nul Part"**

```
194    -- Requête 14
195 •  select distinct T.IntituleTheme
196   from Livre L
197  join Theme T on L.NumTheme = T.NumTheme
198  join Editeur E on L.NumEditeur = E.NumEditeur
199 where E.Nom = 'Nul Part';
```

- **Explication :**

Cette requête sélectionne les thèmes des livres édités par "Nul Part".

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	IntituleTheme			
▶	Informatique			
	Mathématiques			

- **Requête 15 : Afficher les thèmes des livres dans la bibliothèque par éditeur**

```
201    -- Requête 15
202 •  select T.IntituleTheme, E.Nom as Editeur
203   from Livre L
204  join Theme T on L.NumTheme = T.NumTheme
205  join Editeur E on L.NumEditeur = E.NumEditeur;
```

- **Explication :**

Cette requête sélectionne les thèmes des livres dans la bibliothèque, affichant également le nom de l'éditeur pour chaque thème.

- **Résultat :**

	IntituleTheme	Editeur
▶	Informatique	Dar Al-Oujoum
	Informatique	Dar Al-Oujoum
	Economie	Dar Al-Oujoum
	Mathématiques	Dar Al-Oujoum
	Informatique	Dar Al-Oujoum
	Informatique	Nul Part
	Informatique	Nul Part
	Informatique	Nul Part
	Mathématiques	Nul Part
	Mathématiques	Nul Part
	Mathématiques	Nul Part
	Informatique	Nul Part
	Informatique	Nul Part
	Divers	Pour les Bêtes
	Divers	Pour les Bêtes
	Divers	Pour les Bêtes

- **Requête 16 : Afficher les livres de l'auteur "Ramsis" empruntés par l'étudiant "Moujtahid"**

```

207    -- Requête 16
208 •  select L.Titre
209   from Livre L
210  join Pret P on L.NumLivre = P.NumLivre
211 where L.NumAuteur = (select NumAuteur from Auteur where Nom = 'Ramsis') and P.NumEtudiant = (select NumEtudiant from Etudiant where Nom = 'Moujtahid');

```

- **Explication :**

Cette requête sélectionne les titres des livres écrits par "**Ramsis**" et empruntés par l'étudiant "**Moujtahid**".

Cette requête ne retourne aucun résultat, cela signifie que l'étudiant "**Moujtahid**" n'a pas emprunté de livres écrits par l'auteur "**Ramsis**".

- **Résultat :**

	Titre

- **Requête 17 : Afficher les livres qui n'ont jamais été empruntés**

```

212    -- Requête 17
213 •  select L.Titre
214   from Livre L
215  left join Pret P on L.NumLivre = P.NumLivre
216 where P.NumLivre is null;

```

- **Explication :**

Cette requête sélectionne les titres des livres qui n'ont jamais été empruntés.

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Titre				
▶	SGBD Relationnels			
	SI et BD			
	Les BD : Pour les nuls			
	Analyse			
	Algèbre linéaire			
	Aimer les Maths			
	Maitriser UNIX			
	Tout sur les SE			
	Personnes fameuses			
	Comment devenir un bon joueur en 5 jours et s...			

- **Requête 18 : Afficher les livres qui n'ont jamais été empruntés**

```

218      -- Requête 18
219 •  select L.Titre
220   from Livre L
221   left join Pret P on L.NumLivre = P.NumLivre
222   where L.NumAuteur = (select NumAuteur from Auteur where Nom = 'Ramsis') and P.NumLivre is null;

```

- **Explication :**

Cette requête sélectionne les titres des livres écrits par l'auteur "**Ramsis**" qui n'ont jamais été empruntés.

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Titre				
▶	SGBD Relationnels			
	Les BD : Pour les nuls			
	Maitriser UNIX			

- **Requête 19 : Afficher les livres empruntés par l'étudiant "Moujtahid" et leurs thèmes**

```

224      -- Requête 19
225 •  select L.Titre, T.IntituleTheme
226   from Livre L
227   join Pret P on L.NumLivre = P.NumLivre
228   join Theme T on L.NumTheme = T.NumTheme
229   where P.NumEtudiant in (select NumEtudiant from Etudiant where Nom like '%Moujtahid%');

```

- **Explication :**

Cette requête sélectionne les titres des livres empruntés par l'étudiant "**Moujtahid**" et affiche également leurs thèmes.

Cette requête ne retourne aucun résultat, cela signifie que "**Moujtahid**" n'a emprunté aucun livre.

- **Résultat :**

		Result Grid	Filter Rows:	Export:	Wrap Cell Content:
		Titre	IntituleTheme		
231	-- Requête 20				
232	• select L.Titre, T.IntituleTheme, E.Nom as EditeurNom				

- **Requête 20 : Afficher les livres empruntés par l'étudiant "Moujtahid", leurs thèmes et le nom des éditeurs**

```

231      -- Requête 20
232 •  select L.Titre, T.IntituleTheme, E.Nom as EditeurNom
233   from Livre L
234   join Pret P on L.NumLivre = P.NumLivre
235   join Theme T on L.NumTheme = T.NumTheme
236   join Editeur E on L.NumEditeur = E.NumEditeur
237   where P.NumEtudiant in (
238       select NumEtudiant
239       from Etudiant
240       where Nom like '%Moujtahid%'
241   );

```

- **Explication :**

Cette requête sélectionne les titres des livres empruntés par l'étudiant "**Moujtahid**", affiche également leurs thèmes et le nom des éditeurs.

La requête suivante ne retourne aucun résultat, cela signifie que l'étudiant "**Moujtahid**" n'a emprunté aucun livre.

- **Résultat :**

		Result Grid	Filter Rows:	Export:	Wrap Cell Content:
		Titre	IntituleTheme	EditeurNom	
244	-- Requête 21				
245	• select distinct T.IntituleTheme				
246	from Theme T				
247	where not exists (

- **Requête 21 : Afficher les thèmes qui n'ont jamais été empruntés par l'étudiant numéro 999**

```

244      -- Requête 21
245 •  select distinct T.IntituleTheme
246   from Theme T
247   where not exists (
248       select 1
249       from Pret P
250       join Livre L on P.NumLivre = L.NumLivre
251       where L.NumTheme = T.NumTheme and P.NumEtudiant = 999
252   );

```

- **Explication :**

Cette requête sélectionne les thèmes des livres qui n'ont jamais été empruntés par l'étudiant numéro 999.

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	IntituleTheme			
▶	Divers			
	Economie			
	Informatique			
	Mathématiques			
	roman			

- **Requête 22 : Afficher les livres empruntés par l'étudiant "Kaslani" et leurs dates de prêté**

```

255      -- Requête 22
256  •  select L.Titre, P.DatePret
257    from Pret P
258   join Livre L on P.NumLivre = L.NumLivre
259   join Etudiant E on P.NumEtudiant = E.NumEtudiant
260  where E.Nom like '%Kaslani%';

```

- **Explication :**

Cette requête sélectionne les titres des livres empruntés par l'étudiant "Kaslani" et affiche leurs dates de prêt.

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Titre	DatePret		
▶	Comment avoir 20 en BD	2024-05-01		
	L'économie du Maroc en l'an 3050	2024-04-15		

- **Requête 23 : Afficher le titre et l'emprunteur du livre non rendu et dont la date de prêt est la plus ancienne**

```

263      -- Requête 23
264  •  select L.Titre, E.Nom as Emprunteur, P.DatePret
265    from Pret P
266   join Livre L on P.NumLivre = L.NumLivre
267   join Etudiant E on P.NumEtudiant = E.NumEtudiant
268  where P.Rendu = false
269  order by P.DatePret asc
270  limit 1;

```

- **Explication :**

Cette requête sélectionne le titre et l'emprunteur du livre non rendu avec la date de prêt la plus ancienne.

- **Résultat :**

Result Grid			Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	Titre	Emprunteur	DatePret			
▶	Algèbre	Abbassi	2024-03-10			

- **Requête 24 : Afficher tous les livres de la bibliothèque avec le nom de l'auteur, l'éditeur et le thème**

```

272    -- Requête 24
273 • select L.Titre, A.Nom as AuteurNom, E.Nom as EditeurNom, T.IntituleTheme
274   from Livre L
275   join Auteur A on L.NumAuteur = A.NumAuteur
276   join Editeur E on L.NumEditeur = E.NumEditeur
277   join Theme T on L.NumTheme = T.NumTheme;

```

- **Explication :**

Cette requête sélectionne tous les livres de la bibliothèque et affiche également le nom de l'auteur, l'éditeur et le thème pour chaque livre.

- **Résultat :**

Titre	AuteurNom	EditeurNom	IntituleTheme
Comment avoir 20 en BD	Ramsis	Nul Part	Informatique
Tout sur les BD	Neferiti	Nul Part	Informatique
Maitriser les BD	Ramsis	Nul Part	Informatique
SGBD Relationnels	Ramsis	Dar Al-Oujoum	Informatique
SI et BD	Neferiti	Dar Al-Oujoum	Informatique
Les BD : Pour les nuls	Ramsis	Nul Part	Informatique
L'économie du Maroc en l'an 3050	Moujarrib	Dar Al-Oujoum	Economie
Algèbre	Hakim	Nul Part	Mathématiques
Analyse	Hakim	Nul Part	Mathématiques
Algèbre linéaire	Hakim	Dar Al-Oujoum	Mathématiques
Aimer les Maths	Moujarrib	Nul Part	Mathématiques
Systèmes d'exploitation	Ramsis	Nul Part	Informatique
Maitriser UNIX	Ramsis	Dar Al-Oujoum	Informatique
Tout sur les SE	Neferiti	Nul Part	Informatique
Histoire	Tafih	Pour les Bêtes	Divers
Personnes fameuses	Tafih	Pour les Bêtes	Divers
Comment devenir un bon joueur ...	Tafih	Pour les Bêtes	Divers

- **Requête 25 : Afficher les titres des livres empruntés entre le 1/1/15 et le 31/5/15**

```

279    -- Requête 25
280 • select L.Titre
281   from Pret P
282   join Livre L on P.NumLivre = L.NumLivre
283   where P.DatePret between '2015-01-01' and '2015-05-31';

```

- **Explication :**

Cette requête sélectionne les titres des livres empruntés entre le **1er janvier 2015** et le **31 mai 2015**.

- **Résultat :**

Titre

- **Requête 26 : Afficher tous les titres qui contiennent le mot "BD"**

```
284      -- Requête 26
285 •  select Titre
286   from Livre
287   where Titre like '%BD%';
```

- **Explication :**

Cette requête sélectionne tous les titres des livres qui contiennent le mot "BD".

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Titre			
▶	Comment avoir 20 en BD			
	Tout sur les BD			
	Maitriser les BD			
	SGBD Relationnels			
	SI et BD			
	Les BD : Pour les nuls			

- **Requête 27 : Afficher pour chaque auteur, le nombre de livres qu'il a écrits**

```
289      -- Requête 27
290 •  select A.Nom, count(L.NumLivre) as NombreLivres
291   from Auteur A
292   join Livre L on A.NumAuteur = L.NumAuteur
293   group by A.Nom;
```

- **Explication :**

Cette requête sélectionne chaque auteur et affiche le nombre de livres qu'il a écrits.

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Nom	NombreLivres		
▶	Hakim	3		
	Moujarrib	2		
	Neferiti	3		
	Nom de l'Auteur	1		
	Ramsis	6		
	Tafih	3		

- **Requête 28 : Afficher pour chaque ville, l'âge moyen des étudiants originaires de cette ville**

```
295      -- Requête 28
296 •  select Ville, avg(Age) as AgeMoyen
297   from Etudiant
298   group by Ville;
```

- **Explication :**

Cette requête sélectionne chaque ville et affiche l'âge moyen des étudiants originaires de cette ville.

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Ville	AgeMoyen			
Tanger	23.0000			
Chefchaouen	24.0000			
Tétouan	22.5000			
Casablanca	22.4000			

- **Requête 29 : Afficher pour chaque livre emprunté plus de 3 fois, le nombre total de fois qu'il a été emprunté**

```
300    -- Requête 29
301 • select L.Titre, count(*) as NombreEmprunts
302   from Pret P
303   join Livre L on P.NumLivre = L.NumLivre
304   group by L.Titre
305   having count(*) > 3;
```

- **Explication :**

Cette requête sélectionne chaque livre emprunté plus de 3 fois et affiche le nombre total de fois qu'il a été emprunté.

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Titre	NombreEmprunts			

- **Requête 30 : Afficher pour chaque livre non rendu et emprunté plus de 3 fois, le nombre total de fois qu'il a été emprunté**

```
307    -- Requête 30
308 • select L.Titre, count(*) AS NombreEmprunts
309   from Pret P
310   join Livre L on P.NumLivre = L.NumLivre
311   where P.Rendu = false
312   group by L.Titre
313   having count(*) > 3;
```

- **Explication :**

Cette requête sélectionne chaque livre non rendu et emprunté plus de 3 fois et affiche le nombre total de fois qu'il a été emprunté.

- **Résultat :**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Titre	NombreEmprunts			

- **Requête 31 : Afficher les livres de l'auteur "Ramsis" empruntés plus de 4 fois**

```
315    -- Requête 31
316 •  select L.Titre, count(*) as NombreEmprunts
317   from Pret P
318   join Livre L on P.NumLivre = L.NumLivre
319   where L.NumAuteur = (select NumAuteur from Auteur where Nom = 'Ramsis')
320   group by L.Titre
321   having count(*) > 4;
```

- **Explication :**

Cette requête sélectionne les livres de l'auteur "Ramsis" empruntés plus de 4 fois et affiche le nombre total de fois qu'ils ont été empruntés.

- **Résultat :**

Result Grid		Filter Rows: <input type="text"/>	Export:	Wrap Cell Content:
	Titre	NombreEmprunts		

- **Requête 32 : Afficher les étudiants âgés de moins de 23 ans qui ont emprunté plus de 3 livres différents pendant l'année 2015**

```
323    -- Requête 32
324 •  select E.NumEtudiant, E.Nom, count(distinct P.NumLivre) as NombreLivres
325   from Pret P
326   join Etudiant E on P.NumEtudiant = E.NumEtudiant
327   where year(P.DatePret) = 2015 and E.Age < 23
328   group by E.NumEtudiant, E.Nom
329   having count(distinct P.NumLivre) > 3;
```

- **Explication :**

Cette requête sélectionne les étudiants âgés de moins de 23 ans qui ont emprunté plus de 3 livres différents pendant l'année 2015 et affiche le nombre total de livres empruntés par chaque étudiant.

- **Résultat :**

Result Grid		Filter Rows: <input type="text"/>	Export:	Wrap Cell Content:
	NumEtudiant	Nom	NombreLivres	
▶	103	Ayyachi	4	
	202	Khaldouni	4	

11. Conclusion

Dans cet atelier, nous avons appris à interroger la base de données « **Biblio_Sihame** ». Nous avons rempli les tables avec des enregistrements détaillés pour les auteurs, les thèmes, les éditeurs, les étudiants et les livres. Nous avons ensuite utilisé des requêtes SQL pour extraire et manipuler les données selon des critères spécifiques. Cet atelier nous a permis de renforcer notre compréhension des bases de données relationnelles et de la puissance des requêtes SQL pour répondre à des besoins d'information précis et variés

Conclusion générale

Au travers de ces trois ateliers, nous avons acquis une solide compréhension de la gestion des bases de données relationnelles avec MySQL. Nous avons appris à créer des bases de données et des tables, à insérer et modifier des données, et à interroger efficacement la base de données pour extraire des informations précises. Ces compétences sont essentielles pour toute application nécessitant la gestion et la manipulation de données structurées. Le rapport ci-dessus détaille les commandes SQL utilisées et les résultats obtenus, fournissant une vue d'ensemble complète de notre apprentissage et de nos réalisations dans ce module.

