Créer sa base et ses tables avec HeidiSQL

Premier contact avec un vrai SGBDR : on lance MySQL via Laragon, on se connecte avec HeidiSQL, on crée la base de données 'billetterie' et ses deux premières tables. Les clés étrangères suivent au cours suivant.

4TTR niveau

Tu as un MLD complet sur papier : trois tables, leurs colonnes, leurs types. Il est temps de le transposer dans une vraie base MySQL. Dans ce cours, on couvre la **base de données** et **deux tables sans relation entre elles** (`client` et `evenement`). Les clés étrangères (`ticket`) ont leur cours dédié juste après — c'est trop important pour le bâcler ici.

🎯 Objectifs

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

1. Démarrer le service MySQL depuis Laragon
2. Te connecter à ton serveur MySQL avec HeidiSQL
3. Créer une base de données
4. Créer une table avec colonnes, types et clé primaire
5. Insérer des lignes via l'interface
6. Lire le SQL généré par HeidiSQL

Partie 1 — Démarrer MySQL

Vérifier que Laragon/Xampp tourne

Lance **Laragon**. Tu vois une petite fenêtre avec des boutons : **Démarrer tout**, **Arrêter tout**, **Menu**…

Clique sur **Démarrer tout** (ou **Start All**). Deux services s'allument :

- **Apache** (le serveur web — utile plus tard pour phpMyAdmin)
- **MySQL** (le SGBDR — c'est lui qu'on attaque ici)

> 💡 Si tu vois deux pastilles vertes à côté des noms de services, c'est bon : le SERVEUR MYSQL

tourne et écoute sur le port `3306`. Rappelle-toi du cours 06 : sans ça, aucun client ne peut se connecter.

Vérification rapide

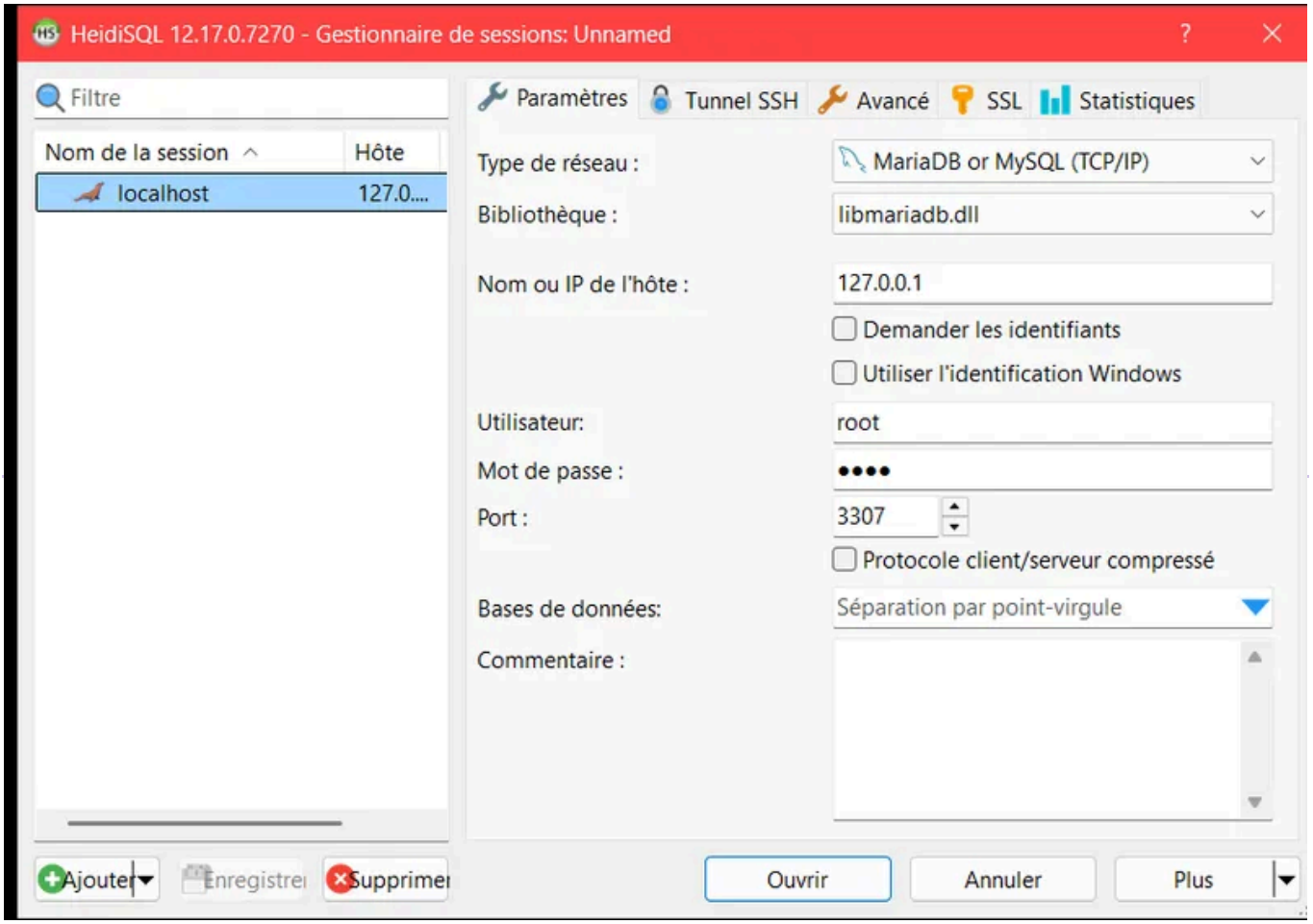
Dans le menu de Laragon, tu peux aller dans **MySQL → Version** : tu verras la version exacte de MariaDB ou MySQL installée. Pas critique, mais c'est un bon réflexe pour savoir avec quoi on travaille.

Partie 2 — Installer et lancer HeidiSQL

HeidiSQL est généralement **déjà installé avec Laragon**. Sinon, télécharge-le depuis heidisql.com (version *Installer*, gratuite).

Au premier lancement, tu arrives sur l'écran **Gestionnaire de sessions** : c'est là qu'on enregistre les connexions vers les serveurs MySQL.

Créer une session



Clique sur **Nouveau** en bas à gauche. Remplis :

Champ	Valeur	Pourquoi
Réseau	**MySQL (TCP/IP)**	On parle à MySQL en réseau, même si c'est local
Hôte / IP	`127.0.0.1` ou `localhost`	Le serveur tourne sur ta propre machine
Utilisateur	`root`	L'admin par défaut de MySQL
Mot de passe	*(vide)*	Laragon ne met pas de mot de passe par défaut
Port	`3306`	Le port standard de MySQL

Renomme la session en haut à gauche (par exemple `Laragon local`) puis clique **Ouvrir**.

> ⚠️ Le compte `root` sans mot de passe est acceptable **en local** pour apprendre. Sur un hébergement réel, c'est interdit — on aura un utilisateur dédié avec un mot de passe sérieux.

Tu es connecté(e)

À gauche : un arbre avec ton serveur et la liste des **bases existantes** (`information_schema`, `mysql`, `performance_schema`…). Ces bases sont **internes au SGBDR** — n'y touche pas. La tienne, on va la créer.

Partie 3 — Créer la base `billetterie`

Dans l'arbre de gauche, **clic droit sur le nom de la session** (la racine, pas une base existante) → **Créer nouveau** → **Base de données…**

Remplis :

Champ	Valeur
Nom	`billetterie`
Interclassement	`utf8mb4_unicode_ci`

Clique **OK**.

L'INTERCLASSEMENT

(*collation*) définit comment MySQL stocke et compare les caractères. `utf8mb4_unicode_ci` accepte tous les caractères Unicode (accents, emojis, alphabets non latins) et ignore la casse pour les comparaisons. C'est le bon choix par défaut en 2026 — c'est généralement l'option proposée d'office par HeidiSQL.

SI `UTF8MB4_UNICODE_CI` N'APPARAÎT PAS

dans la liste (cela dépend de la version de MariaDB/MySQL), prends dans l'ordre :

- `utf8mb4_unicode_520_ci` — équivalent, basé sur Unicode 5.20
- `utf8mb4_uca1400_ai_ci` (ou `uca1400_ai_ci`) — sur les MariaDB récentes, basé sur Unicode 14
- `utf8mb4_general_ci` — fallback de dernier recours, plus rapide mais moins précis (suffisant à ce stade)

L'essentiel est que le nom commence par `utf8mb4_` — pas `utf8_` tout court, qui ne stocke ni les emojis ni certains caractères asiatiques.

La base `billetterie` apparaît dans l'arbre — vide. On va la peupler.

Partie 4 — Créer la table `client`

Rappel du MLD :

```
client (
    id          INT  PK  AUTO_INCREMENT
    nom         VARCHAR(100)  NOT NULL
    prenom      VARCHAR(100)  NOT NULL
    email       VARCHAR(150)
    telephone   VARCHAR(20)
)
```

Lancer le formulaire

Clic droit sur la base `billetterie` → **Créer nouveau** → **Table**.

Une fenêtre de conception s'ouvre. En haut, le champ **Nom** : tape `client`.

Ajouter les colonnes

Dans le tableau central, chaque ligne représente une **colonne**. Clique **Ajouter** pour insérer une nouvelle colonne, puis remplis les champs :

Colonne 1 — `id` *(la plus importante)*

Champ	Valeur
Nom	`id`
Type de données	`INT`
Autoriser NULL	**décoché**
Défaut	*(rien)*
Cocher la case	`AUTO_INCREMENT`

Puis, à droite du tableau, ajoute un **index** de type **PRIMARY** sur la colonne `id` (onglet *Index* ou bouton dédié — l'emplacement varie selon la version).

💡

POURQUOI CES DEUX RÉGLAGES ?

- `AUTO_INCREMENT` dit à MySQL : *à chaque nouvelle ligne, attribue automatiquement le prochain entier disponible*. Tu n'as plus à inventer ou compter — la base s'occupe de générer un `id` unique pour chaque insertion. Sans ça, il faudrait fournir un `id` manuellement à chaque `INSERT`, avec le risque de doublons.
- **L'index** `PRIMARY` déclare que cette colonne est la **clé primaire** de la table. Trois conséquences techniques : MySQL refuse les doublons sur cette colonne, refuse les valeurs `NULL`, et crée un **index**

qui permet de retrouver n'importe quelle ligne par son `id` en une fraction de seconde, même avec des millions d'enregistrements.

En pratique, **les deux vont toujours ensemble** sur la colonne `id` d'une table : `AUTO_INCREMENT` garantit la génération automatique, `PRIMARY` garantit l'unicité et la rapidité d'accès. C'est ce qu'on a posé conceptuellement au cours 05 sur les identifiants — on le matérialise enfin ici.

Colonnes 2 à 5

Nom	Type	NULL
`nom`	`VARCHAR(100)`	non
`prenom`	`VARCHAR(100)`	non
`email`	`VARCHAR(150)`	oui
`telephone`	`VARCHAR(20)`	oui

Clique **Enregistrer**. La table `client` apparaît dans l'arbre.

Le SQL généré

En bas, HeidiSQL affiche **la requête SQL** qu'il vient d'envoyer au serveur :

```sql
CREATE TABLE `client` (
    `id`        INT          NOT NULL AUTO_INCREMENT,
    `nom`       VARCHAR(100) NOT NULL,
    `prenom`    VARCHAR(100) NOT NULL,
    `email`     VARCHAR(150) NULL,
    `telephone` VARCHAR(20)  NULL,
    PRIMARY KEY (`id`)
);
```

💡
REGARDE-LA ATTENTIVEMENT.

C'est exactement ce qu'on aurait tapé à la main dans un terminal `mysql`. HeidiSQL est juste un générateur de SQL — comprendre le SQL généré te rendra autonome.

Partie 5 — Créer la table `evenement`

Refais la même procédure pour la table `evenement` :

Nom	Type	NULL	Note
`id`	`INT`	non	PK, AUTO_INCREMENT
`titre`	`VARCHAR(200)`	non	
`date`	`DATETIME`	non	
`prix_base`	`DECIMAL(6,2)`	non	

Vérifie le SQL généré — il doit ressembler à :

```sql
CREATE TABLE `evenement` (
    `id`         INT           NOT NULL AUTO_INCREMENT,
    `titre`      VARCHAR(200)  NOT NULL,
    `date`       DATETIME      NOT NULL,
    `prix_base` DECIMAL(6,2)  NOT NULL,
    PRIMARY KEY (`id`)
);
```

Partie 6 — Insérer des données via l'interface

Sélectionne la table `client` dans l'arbre, puis va sur l'onglet **Données**. Tu vois un tableau vide.

Clique **Insérer une ligne** (en bas, ou bouton `+`). Remplis :

nom	prenom	email	telephone
Dupont	Marie	marie.dupont@email.be	0471 12 34 56

Ne remplis **pas** la colonne `id` — laisse `AUTO_INCREMENT` faire son travail. Valide (Enter ou bouton **Publier**).

Ajoute deux ou trois autres clients. Tu verras que les `id` se suivent : 1, 2, 3…

Fais pareil pour `evenement` :

titre	date	prix_base
Soirée Quiz Pop Culture	2026-10-03 20:00:00	12.00

titre	date	prix_base
Blind Test Années 90	2026-11-15 21:00:00	10.00
Quiz Cinéma	2026-12-05 20:30:00	15.00

> 💡 Le format `DATETIME` est `AAAA-MM-JJ HH:MM:SS`. HeidiSQL accepte aussi un petit calendrier graphique si tu cliques dans la cellule.

Partie 7 — Vérifier avec une requête SQL

Onglet **Requête** (en haut). Tape :

```sql
SELECT * FROM client;
```

Appuie sur **F9** ou clique **Exécuter**. Le résultat s'affiche en dessous : toutes les lignes de la table `client`.

```sql
SELECT id, titre, date FROM evenement WHERE prix_base < 15;
```

Tu apprendras `SELECT` en détail dans un cours dédié — c'est juste pour valider que tes données sont bien là.

🧪 Exercices

Exercice 1 — Créer une 3e table simple

Crée une table `salle` avec ces colonnes :

nom	type	NULL
`id`	INT	non (PK + AUTO_INCREMENT)
`nom`	VARCHAR(150)	non
`ville`	VARCHAR(100)	non

nom	type	NULL
`capacite`	INT	oui

Insère 3 salles.

Exercice 2 — Lire le SQL

Va dans la table `client`, fais clic droit → **Exporter en SQL** → **Vers le presse-papiers** (ou équivalent). Colle le résultat dans un fichier texte. Tu obtiens un **dump SQL** : c'est ce qui te permettra de transférer ta base ailleurs (et c'est exactement ce que phpMyAdmin produira aussi).

Exercice 3 — Ton projet

Sur ta machine, crée la base de données de **ton projet personnel** (celui de la séquence 4). Crée toutes ses tables **sauf** celles qui ont des clés étrangères. Insère quelques lignes dans chacune.

> ⚠️ Si toutes tes tables ont des FK, garde la table principale (celle dont les autres dépendent) pour ce cours, et fais les autres au cours suivant.

🧰 Petits réflexes utiles dans HeidiSQL

- **F5** : rafraîchir l'arbre des bases (utile si tu as créé quelque chose sans que ça apparaisse)
- **F9** : exécuter la requête SQL en cours
- **Ctrl + clic** sur une cellule : édition rapide
- **Onglet *Vue d'ensemble*** d'une table : un résumé des colonnes, indexes et taille
- **Bouton *Annuler*** : tant que tu n'as pas **Publié**, tes modifications de données ne sont pas envoyées au serveur

---

✍️ À retenir

- Avant tout : **MySQL doit tourner** dans Laragon (pastille verte).
- HeidiSQL se connecte avec `localhost`, user `root`, pas de mot de passe (en local seulement).
- Une **base de données** est créée à part — elle contient les tables.
- Pour chaque table : nom, colonnes avec leur type, et **clé primaire** `id INT AUTO_INCREMENT`.
- L'interface graphique est pratique mais **regarde toujours le SQL généré** — c'est lui la vérité.
- L'insertion de données ne nécessite **jamais** de remplir l'`id` à la main.

Suite

Tes deux tables `client` et `evenement` existent, mais **rien ne les relie**. Au prochain cours, on crée la table `ticket` avec ses **clés étrangères** — c'est elle qui matérialisera les relations du MCD dans la base.