ISTE.230.600 - Introduction to Database and Data Modeling

**Database Development Lifecycle Phases Project**

*Book Nook Library*

Mouza Alameri | 776006459

Suhaila Alfalasi| 764005288

Rania Kanaan | 769007321

Dr. Zainab Al-Zanbouri

Fall 2024

November 24, 2024

Group 4

**Manifesto**

| **Student** | **ID** | **Contribution** |
| --- | --- | --- |
| Mouza Alameri | 776006459 | Formating & Design  Transposition  Normalization |
| Suhaila Alfalasi    Rania Kanaan | 764005288  769007321 | Business Scenario  Entities and Relationships  ER Diagram  SQL code |

**Table of content**

[**PHASE I | Conceptual and Logical Design 4**](#_mefk4eijix5m)

[**1. Define Business Scenario 4**](#_hchc6exk9ets)

[1.1 Business Scenario Explanation 4](#_9yyr2hy2a5q9)

[1.2 Core User Requirements 6](#_y2otihr8ux2p)

[**2. Entities and Relationships 7**](#_2xx40a7jrrre)

[2.1 List of Entities and Attributes 7](#_8dq1yvl99sim)

[2.2 List of Relationships Between Entities 9](#_af6tmgbdkupb)

[**3. E-R Diagram 10**](#_y5bdsze8nkex)

[**4. Transposition 11**](#_dyql1wrlum7e)

[**5. Normalization 15**](#_polfnizeqrfb)

[**PHASE II | Physical Design 28**](#_kbioxe9h74oa)

[**1. Database Creation 28**](#_cwv8x8q9p5l7)

[**2. Tables and Relationships** 28](#_rgajc0ph31ah)

[2.2 Create the Publisher Table: 29](#_489o17g18ur)

[2.3 Create the Book Table: 29](#_ocoh2lf69hsx)

[2.4 Create the MembershipType Table: 29](#_d5ogvzei642s)

[2.5 Create the Member Table: 30](#_xjapf2qgr9o2)

[2.6 Create the Loan Table: 31](#_qabto4xhf58l)

[2.7 Create the Fine Table: 31](#_3cj7aao6akmi)

[**3. Sample Data Insertion 32**](#_g6kydea7kmnc)

[3.1 Insert Data into Author: 32](#_uj9rppggzebf)

[3.2 Insert Data into Publisher: 32](#_sgddkqcjlhpc)

[3.3 Insert Data into Book: 32](#_w92i7vrkcfsg)

[3.4 Insert Data into MembershipType: 33](#_myp5fwsivhob)

[3.5 Insert Data into Member: 33](#_e9ws6jgm1vtn)

[3.6 Insert Data into Loan: 33](#_6lty201lkjov)

[3.7 Insert Data into Fine: 34](#_4x76v88067i1)

[**4. Query Examples 34**](#_9adm3oqz5fsu)

[4.1 View All Overdue Loans: 34](#_ffcn1gkwgg7y)

[4.2 View Fine Details for Overdue Loans: 35](#_d7a4pugr57gc)

[4.3 Check Books Availability: 36](#_dbf8yb77p694)

[4.4 Update on the book: 37](#_2lr733v4j37i)

[4.5 ALTER TABLE for the author: 38](#_egdu136hefyk)

[4.6 Describe the attributes: 40](#_30ngo5pa4fuc)

[4.7 Delete from Book: 45](#_cp6aonymv3tw)

# 

# PHASE I | Conceptual and Logical Design

## [**1. Define Business Scenario**](#_9vsyoh7rk62i)

### [**1.1 Business Scenario Explanation**](#_5qo7lykk2aiv)

Book Nook Library is one among the greatest libraries in Dubai which wants to expand and provide more variety of books to all ages. Book Nook Library is currently trying to improve the library’s performance by trying to offer a variety of services and for this purpose and in that case it has developed the concept of implementing a new library management system. This system will assist in the arrangement of large volumes of books classified in the various genres such as: fiction, non-fiction, academic, litecture, children’s books, etc. The library needs to implement services that fit all ages categories such as Children and Young People’s Services, Computing Services, Local History, and Reading Groups and Workshops. Book Nook Library is facing issues with:

* Managing an expanding book inventory.
* Handling member registrations and renewals.
* Tracking loans, returns, and overdue fines.
* Managing special events and workshops.

The primary objective of the Book Nook library is to develop a Library Management System which will assist in establishing a centralized database to provide information on books, members, loans, events and fines.

Book Nook Library Scenario Overview:

The Library Operation:

* The library will start providing a variety of different genres of books that will be available for members to borrow or access within the library.
* Members can choose from different types of memberships such as, family membership, student membership, premium membership, etc) with each that provide different privileges access.
* The members of the library can borrow books from the library, but they have a limit to the amount of books they can borrow. They are as well responsible for the time they are supposed to return.
* Library plans to host some interesting events that include reading sessions, book clubs, workshops, etc, that will require for the people to register beforehand.
* The library maintains an organized system to monitor book availability, loan statuses, member activity, and overdue fines.

Database Objectives:

* The database will centralize and organize information related to books, members, loans, events, and fines.
* It will facilitate efficient management of member enrollments, book loans, event registrations, and fine calculations.
* All employees of the library will have the access to the database which they will have the ability to input, retrieve, and update the data.

The main purpose of this database is to assist the employees of the library in a simple way in order to enable them to do the same work as they have been doing in an efficient manner make necessary improvements to the members services and well as make administration improvements as well.

### **1.2 Core User Requirements**

The Library Management System functionalities that it must support:

* **Member Management:** The library staff should register anyone who is interested in becoming a member in the Book Nook library program. The staff should record members personal details like full names, contact details as well as type of membership.
* **Book Inventory Management:** The staff members should be able to add new books, update details of the book such as the author name, genre, publication date, and as well be able to manage available copies.
* **Loan Management:** Track book loans, including due dates, return dates, and current status (borrowed/returned).
* **Fine Calculation and Management:** Automatically calculate overdue fees if a book is returned late and track fine payments.
* **Search and Reporting:** Help to search books, members and loan records and generate reports for library activities such as popular books, active members etc.
* **Inventory Alerts:** Notifying the staff when book availability has reached a certain minimum so that they restock it on time.

## **2. Entities and Relationships**

### **2.1 List of Entities and Attributes**

Book: Information about each book available at Book Nook Library.

* Attributes:
  + book\_id (Primary Key): Unique identifier for each book.
  + title: Title of the book.
  + author\_id: Foreign Key linking to Author.
  + genre: Genre or category of the book.
  + publication\_year: Year the book was published.
  + publisher\_id: Foreign Key linking to Publisher.
  + quantity\_available: Number of copies currently available.

Author: Information about each book’s author.

* Attributes:
  + author\_id (Primary Key): Unique identifier for each author.
  + name: Author's full name.
  + bio: Brief biography or description of the author.

Publisher: Details about publishers of books in the library.

* Attributes:
  + publisher\_id (Primary Key): Unique identifier for each publisher.
  + name: Publisher’s name.
  + address: Publisher’s address.

Member: Information about each Book Nook Library member.

* Attributes:
  + member\_id (Primary Key): Unique identifier for each member.
  + name: Full name of the member.
  + address: Member’s address.
  + email: Member’s email address.
  + membership\_type\_id: Foreign Key linking to MembershipType.

Loan: Records of loan transactions for borrowing books.

* Attributes:
  + loan\_id (Primary Key): Unique identifier for each loan.
  + book\_id (Foreign Key): Links to Book.
  + member\_id (Foreign Key): Links to Member.
  + loan\_date: Date the book was borrowed.
  + due\_date: Due date for the book’s return.
  + return\_date: Actual date of return, if applicable.

Fine: Records overdue fees for late returns.

* Attributes:
  + fine\_id (Primary Key): Unique identifier for each fine.
  + loan\_id (Foreign Key): Links to Loan.
  + amount: Amount charged as a fine.
  + date\_issued: Date the fine was issued.

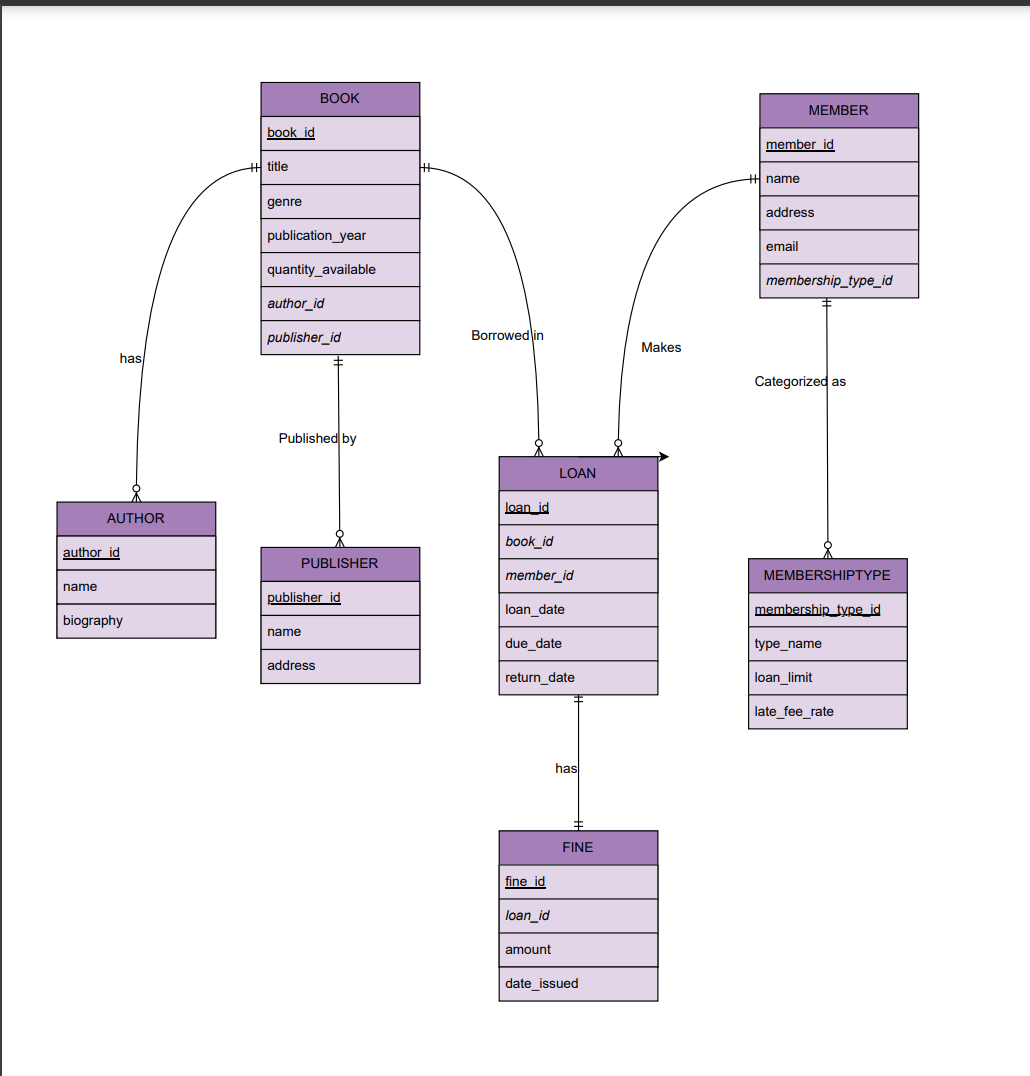
MembershipType: Types of memberships available with associated privileges.

* Attributes:
  + membership\_type\_id (Primary Key): Unique identifier for each type.
  + type\_name: Name of the membership type (e.g., Regular, Premium).
  + loan\_limit: Maximum books a member can borrow at one time.
  + late\_fee\_rate: Daily rate charged for overdue books.

### **2.2 List of Relationships Between Entities**

* Book - Author: Many-to-One. Each book has one author, but an author can write multiple books.
* Book - Publisher: Many-to-One. Each book is associated with one publisher, but a publisher can publish multiple books.
* Member - Loan: One-to-Many. Each member can have multiple loans, but each loan links to only one member.
* Book - Loan: One-to-Many. A book can have multiple loan records, each linked to a single transaction.
* Loan - Fine: One-to-One. A loan may have one associated fine if overdue, and each fine is linked to a single loan.
* Member - MembershipType: Many-to-One. Each member has one membership type, but each membership type can apply to multiple members.

## **3. E-R Diagram**



## 4. Transposition

Step 1: list all entity names as relations names in hierarchical order.

BOOK()

AUTHOR()

PUBLISHER()

MEMBER()

MEMBERSHIPTYPE()

LOAN()

FINE()

Step 2 : copy the attributes of entity which becomes attributes of the relation.

BOOK(book\_id, title, genre, publication\_year, quantity\_available, author\_id, publisher\_id.)

AUTHOR(author\_id, name, biography)

PUBLISHER(publisher\_id, name, address)

MEMBER(member\_id, name, address, email, membership\_type\_id)

MEMBERSHIPTYPE(membership\_type\_id, type\_name, loan\_limit, late\_fee\_rate)

LOAN(loan\_id, book\_id, member\_id, loan\_date, due\_date, return\_date)

FINE(fine\_id, loan\_id, amount, date\_issued, date\_paid)

Step 3: Define a primary key and forigen | Note: not confirmed until 1NF. PK: Denoted by an underline , FK : denoted by italizeled.

BOOK(book\_id, title, genre, publication\_year, quantity\_available, *author\_id, publisher\_id.*)

AUTHOR(author\_id, name, biography)

PUBLISHER(publisher\_id, name, address)

MEMBER(member\_id, name, address, email, *membership\_type\_id*)

MEMBERSHIPTYPE(membership\_type\_id, type\_name, loan\_limit, late\_fee\_rate,*member\_id*)

LOAN(loan\_id, *book\_id, member\_id,* loan\_date, due\_date, return\_date)

FINE(fine\_id, *loan\_id*, amount, date\_issued, date\_paid)

Step 4: examine cardinalities

Our diagrams have,

1:1 cardinality (mandatory)

Based on our logic we placed the loan pk in fine as fk to ensure referential integrity. Every fine is tied to an existing loan.

It retains everything it has

1:N cardinality

Move the pk from one side to be the fk in many side.

BOOK(book\_id, title, genre, publication\_year, quantity\_available, *author\_id, publisher\_id.*)

AUTHOR(author\_id, name, biography,*book\_id*)

PUBLISHER(publisher\_id, name, address,*book\_id*)

MEMBER(member\_id, name, address, email, *membership\_type\_id*)

MEMBERSHIPTYPE(membership\_type\_id, type\_name, loan\_limit, late\_fee\_rate, *member\_id*)

LOAN(loan\_id, *book\_id, member\_id,* loan\_date, due\_date, return\_date)

FINE(fine\_id, *loan\_id*, amount, date\_issued, date\_paid)

Step 5 : add a reference for the foreign key.

BOOK(book\_id, title, genre, publication\_year, quantity\_available, *author\_id, publisher\_id.*)

BOOK(author\_id) mei AUTHOR(author\_id)

BOOK(publisher\_id) mei PUBLISHER(publisher\_id)

AUTHOR(author\_id, name, biography,*book\_id*)

AUTHOR(book\_id) mei BOOK(book\_id)

PUBLISHER(publisher\_id, name, address,*book\_id*)

PUBLISHER(book\_id) mei BOOK(book\_id)

MEMBER(member\_id, name, address, email, *membership\_type\_id*)

MEMBER(membership\_type\_id) mei MEMBERSHIPTYPE( membership\_type\_id)

MEMBERSHIPTYPE(membership\_type\_id, type\_name, loan\_limit, late\_fee\_rate, *member\_id*)

MEMBERSHIPTYPE(member\_id) mei MEMBER(member\_id)

LOAN(loan\_id, *book\_id, member\_id,* loan\_date, due\_date, return\_date)

LOAN(member\_id) mei MEMBER(member\_id)

FINE(fine\_id, *loan\_id*, amount, date\_issued, date\_paid)

FINE(loan\_id) mei LOAN(loan\_id)

| *Final Transposition :*  BOOK(book\_id, title, genre, publication\_year, quantity\_available, *author\_id, publisher\_id.*)  BOOK(author\_id) mei AUTHOR(author\_id)  BOOK(publisher\_id) mei PUBLISHER(publisher\_id)  AUTHOR(author\_id, name, biography,*book\_id*)  AUTHOR(book\_id) mei BOOK(book\_id)  PUBLISHER(publisher\_id, name, address,*book\_id*)  PUBLISHER(book\_id) mei BOOK(book\_id)  MEMBER(member\_id, name, address, email, *membership\_type\_id*)  MEMBER(membership\_type\_id) mei MEMBERSHIPTYPE( membership\_type\_id)  MEMBERSHIPTYPE(membership\_type\_id, type\_name, loan\_limit, late\_fee\_rate, *member\_id*)  MEMBERSHIPTYPE(member\_id) mei MEMBER(member\_id)  LOAN(loan\_id, *book\_id, member\_id,* loan\_date, due\_date, return\_date)  LOAN(member\_id) mei MEMBER(member\_id)  FINE(fine\_id, *loan\_id*, amount, date\_issued, date\_paid)  FINE(loan\_id) mei LOAN(loan\_id) |
| --- |

# 

## 5. Normalization

1. BOOK(book\_id, title, genre, publication\_year, quantity\_available, *author\_id, publisher\_id.*)

Functional Dependencies :

| **book\_id → title, genre, publication\_year, quantity\_available, author\_id, publisher\_id**  Book\_id uniquely identifies a book.  **author\_id, title → book\_id, genre, publication\_year, quantity\_available, publisher\_id**  Author\_id and title create a unique combination of identifying a book.  **publisher\_id, title → book\_id, genre, publication\_year, quantity\_available, author\_id**  Publisher\_id and title also uniquely identify a book.  **book\_id → quantity\_available**  The quantity of books available is specified by book id |
| --- |

Candidate keys :

| * Book\_id | simple candidate key * (author\_id, title) | Composite candidate key * (publisher\_id, title) | Composite candidate key |
| --- |

“Proposed” Primary Keys:

| Book\_id |
| --- |

1NF Through 3NF:

| **1NF :** No atomic no repeating groups  **2NF :** no partial dependency, and follows the first normal form  **3NF :** it has a transitive dependency  **BOOK\_DETAILS(book\_id, title, genre, publication\_year)**  Functional Dependency: book\_id → title, genre, publication\_year  **AUTHOR\_BOOK(author\_id, title, book\_id)**  Functional Dependency: author\_id, title → book\_id  **PUBLISHER\_BOOK(publisher\_id, title, book\_id)**  Functional Dependency: publisher\_id, title → book\_id  **BOOK\_AVAILABILITY(book\_id, quantity\_available)**  Functional Dependency: book\_id → quantity\_available |
| --- |

1. AUTHOR(author\_id, name, biography,*book\_id*)

Functional Dependencies :

| **author\_id → name, biography**  The author uniquely identifies an author.  **book\_id → author\_id**  Each book\_id is written by an author.  **book\_id → name, biography**  book\_id indirectly identifies name and biography. |
| --- |

Candidate keys :

| author\_id  book\_id |
| --- |

“Proposed” Primary Keys:

| Author\_id  Why not book\_id?  *Using book\_id as a primary key would cause redundancy* |
| --- |

(1NF) Through Boyce-Codd Normal Form (BCNF):

| 1NF: no atomic or repeating values  2NF: no partial dependency  3NF: there are no transitive dependencies  book\_id → author\_id and author\_id → name, biography creates a **transitive dependency**:  book\_id indirectly determines name and biography.  **AUTHOR(author\_id, name, biography)**  **BOOK\_AUTHOR(book\_id, author\_id)** |
| --- |

Candidate keys :

| Book\_id:  publisher\_id: |
| --- |

“Proposed” Primary Keys:

| Book\_id |
| --- |

(1NF) Through Boyce-Codd Normal Form (BCNF):

| 1NF : yes, atomic non repeating  2NF : no, partial dependencies exist  **PUBLISHER(publisher\_id, name, address)**  **BOOK\_PUBLISHER(book\_id, publisher\_id)**  3NF : no, transitive dependency exists  **PUBLISHER(publisher\_id, name, address)**   * Functional Dependency: publisher\_id → name, address   **BOOK\_PUBLISHER(book\_id, publisher\_id)**  Functional Dependency: book\_id → publisher\_id |
| --- |

1. PUBLISHER(publisher\_id, name, address,*book\_id*)

Functional Dependencies :

| **Publisher\_id→name,address**  **name→publisher\_id,address**  **address→publisher\_id,name**  **book\_id→publisher\_id,name,address**  **publisher\_id,book\_id→name,address** |
| --- |

Candidate keys :

| book\_id |
| --- |

“Proposed” Primary Keys:

| book\_id |
| --- |

(1NF) Through Boyce-Codd Normal Form (BCNF):

| 1NF: no, atomic  2NF: Yes, partial dependencies  book\_id→publisher\_id, name,address  Publisher\_id→name,address  BOOK(book\_id,publisher\_id)  PUBLISHER(publisher\_id,name,address)  No transitive dependenicies |
| --- |

1. MEMBER(member\_id, name, address, email, *membership\_type\_id*)

Functional Dependencies :

| **member\_id → name, address, email, membership\_type\_id**  member\_id uniquely identifies a member |
| --- |

Candidate keys :

| member\_id |
| --- |

“Proposed” Primary Keys:

| member\_id |
| --- |

(1NF) Through Boyce-Codd Normal Form (BCNF):

| 1NF : Yes , atomic  2NF: Yes , no partial dependencies  3NF: no, transitive dependencies exist  ***MEMBER(member\_id, name, address, email, membership\_type\_id)***  *Functional Dependency: member\_id → name, address, email, membership\_type\_id*  ***MEMBERSHIP\_TYPE(membership\_type\_id, membership\_type\_details)***  *Functional Dependency: membership\_type\_id → membership\_type\_details* |
| --- |

1. MEMBERSHIPTYPE(membership\_type\_id, type\_name, loan\_limit, late\_fee\_rate, *member\_id*)

Functional Dependencies :

| **membership\_type\_id → type\_name, loan\_limit, late\_fee\_rate**  **member\_id → membership\_type\_id**  **member\_id → type\_name, loan\_limit, late\_fee\_rate (via transitivity)** |
| --- |

Candidate keys :

| **member\_id** is a candidate key because it uniquely determines all attributes |
| --- |

“Proposed” Primary Keys:

| Member\_id |
| --- |

(1NF) Through Boyce-Codd Normal Form (BCNF):

| **1NF :** No atomic no repeating groups  **2NF :** no partial dependency, and follows the first normal form  **3NF :**it has a transitive dependency   * **member\_id → membership\_type\_id and membership\_type\_id → type\_name, loan\_limit, late\_fee\_rat**   **MEMBERSHIPTYPE(membership\_type\_id, type\_name, loan\_limit, late\_fee\_rate)**  **MEMBER(member\_id, membership\_type\_id)** |
| --- |

#### 

6. LOAN(loan\_id, *book\_id, member\_id,* loan\_date, due\_date, return\_date)

Functional Dependencies :

| **loan\_id → book\_id, member\_id, loan\_date, due\_date, return\_date**  **book\_id, member\_id, loan\_date → loan\_id, due\_date, return\_date** |
| --- |

Candidate keys :

| **loan\_id** (simple candidate key)  **book\_id, member\_id, loan\_date** (composite candidate key) |
| --- |

“Proposed” Primary Keys:

| loan\_id |
| --- |

(1NF) Through Boyce-Codd Normal Form (BCNF):

| **1NF :** No atomic no repeating groups  **2NF :** no partial dependency, and follows the first normal form  **3NF :** it has a transitive dependency   * **loan\_id → book\_id, member\_id, loan\_date → due\_date, return\_date**:   **LOAN\_DETAILS(loan\_id, book\_id, member\_id, loan\_date)**  **LOAN\_SCHEDULE(loan\_id, due\_date, return\_date)** |
| --- |

7. FINE(fine\_id, *loan\_id*, amount, date\_issued, date\_paid)

Functional Dependencies :

| **fine\_id → loan\_id, amount, date\_issued, date\_paid**  **loan\_id → amount, date\_issued**  **fine\_id → loan\_id → amount, date\_issued (transitivity)**  **fine\_id → date\_paid** |
| --- |

Candidate keys :

| * **Fine\_id**   loan\_id determines amount and date\_issued, it does not determine date\_paid, which depends on fine\_id. Therefore, loan\_id cannot uniquely identify all attributes in the relation. |
| --- |

“Proposed” Primary Keys:

| fine\_id |
| --- |

(1NF) Through Boyce-Codd Normal Form (BCNF):

| **1NF :** No atomic no repeating groups  **2NF :** no partial dependency, and follows the first normal form  **3NF :**it has a transitive dependency   * **fine\_id → loan\_id → amount, date\_issued** * **FINE(fine\_id, loan\_id, date\_paid)** * **LOAN\_FINE\_DETAILS(loan\_id, amount, date\_issued)** |
| --- |

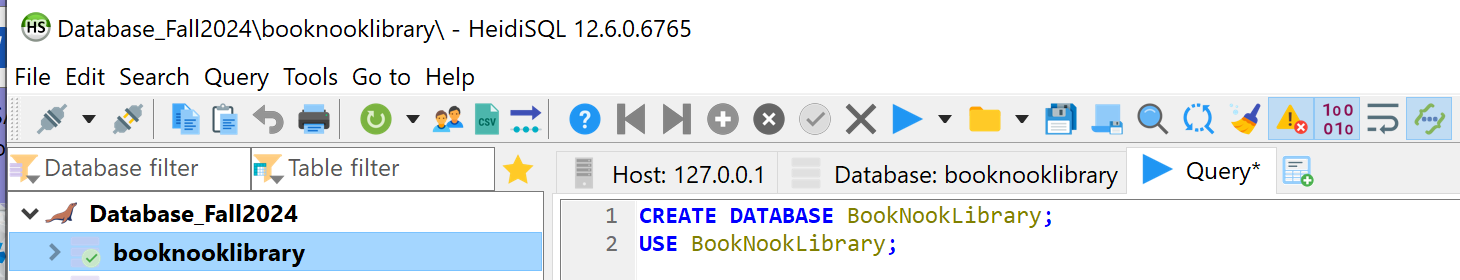
# 

# 

# PHASE II | Physical Design

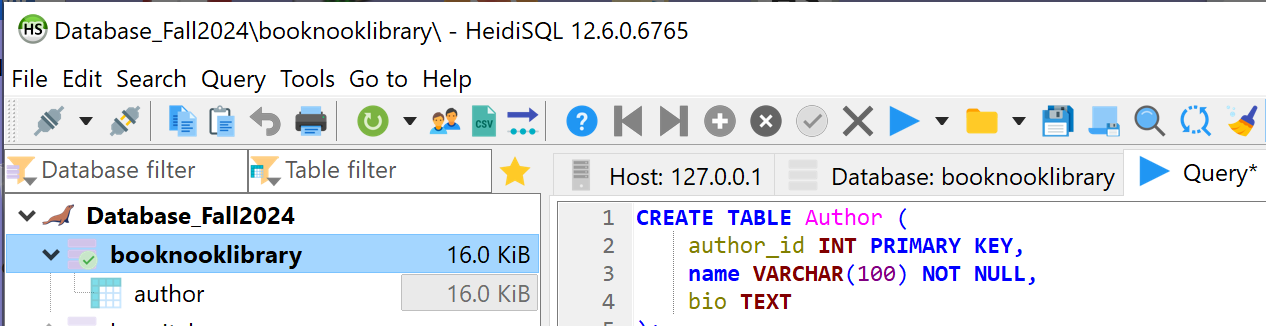
## 1. **Database Creation**

| CREATE DATABASE BookNookLibrary; USE BookNookLibrary; |
| --- |



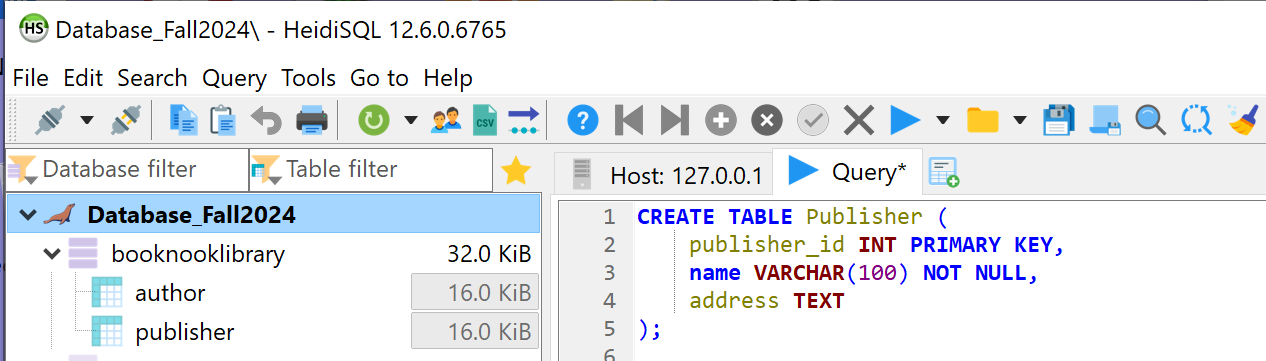
## 2. **Tables and Relationships**

| **CREATE TABLE Author (  author\_id INT PRIMARY KEY,  name VARCHAR(100) NOT NULL,  bio TEXT );** |
| --- |

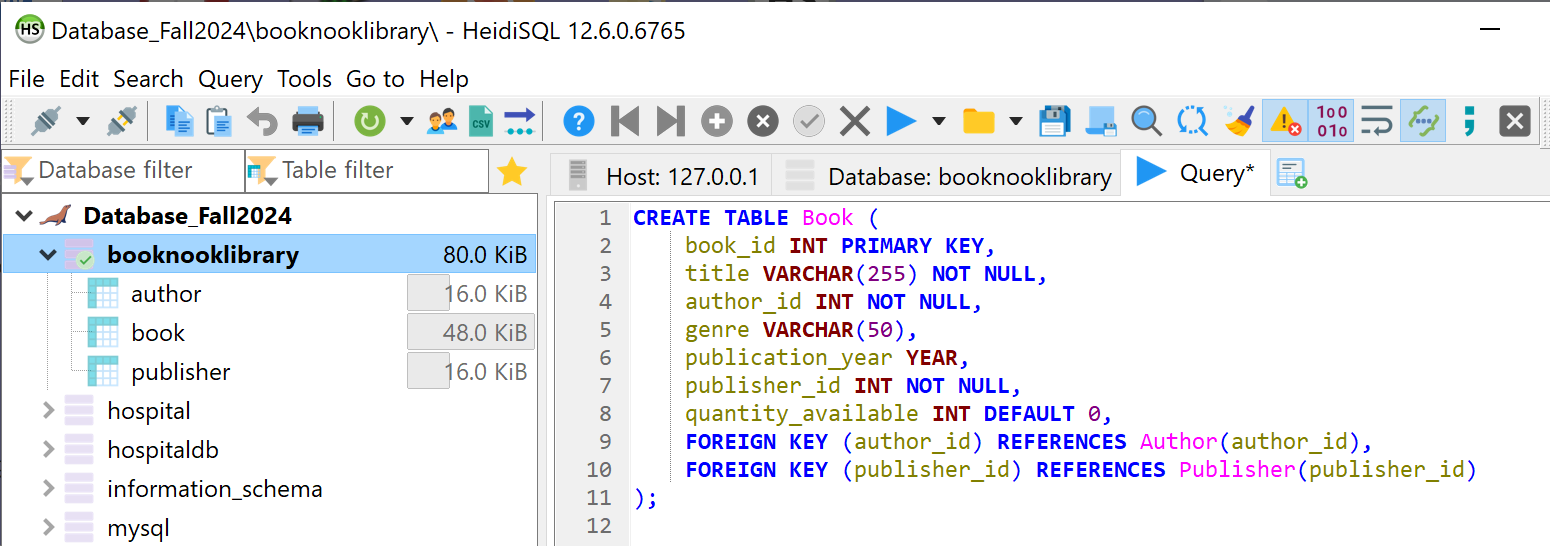
****

### **2.2 Create the Publisher Table:**

| **CREATE TABLE Publisher (  publisher\_id INT PRIMARY KEY,  name VARCHAR(100) NOT NULL,  address TEXT );** |
| --- |

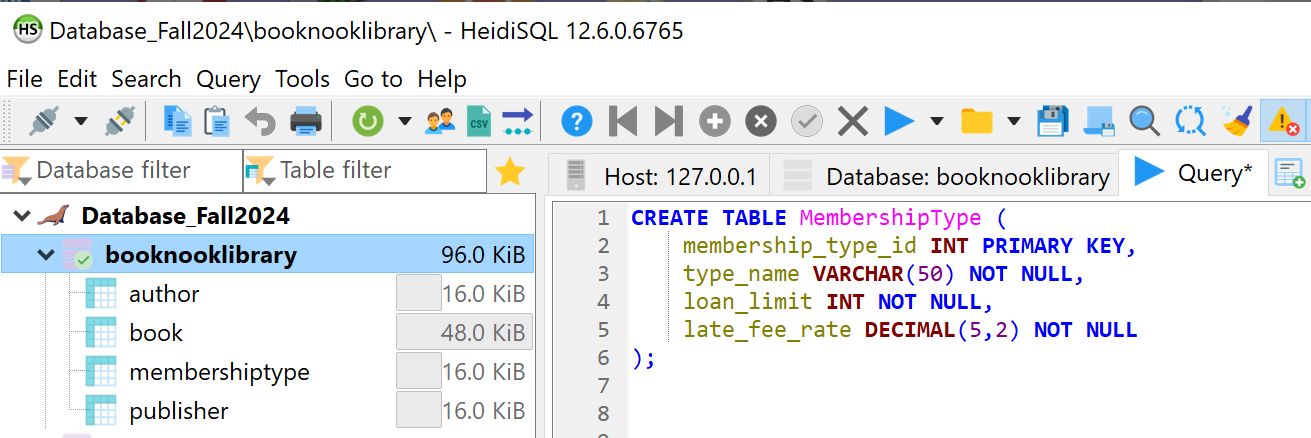
****

### **2.3 Create the Book Table:**

****

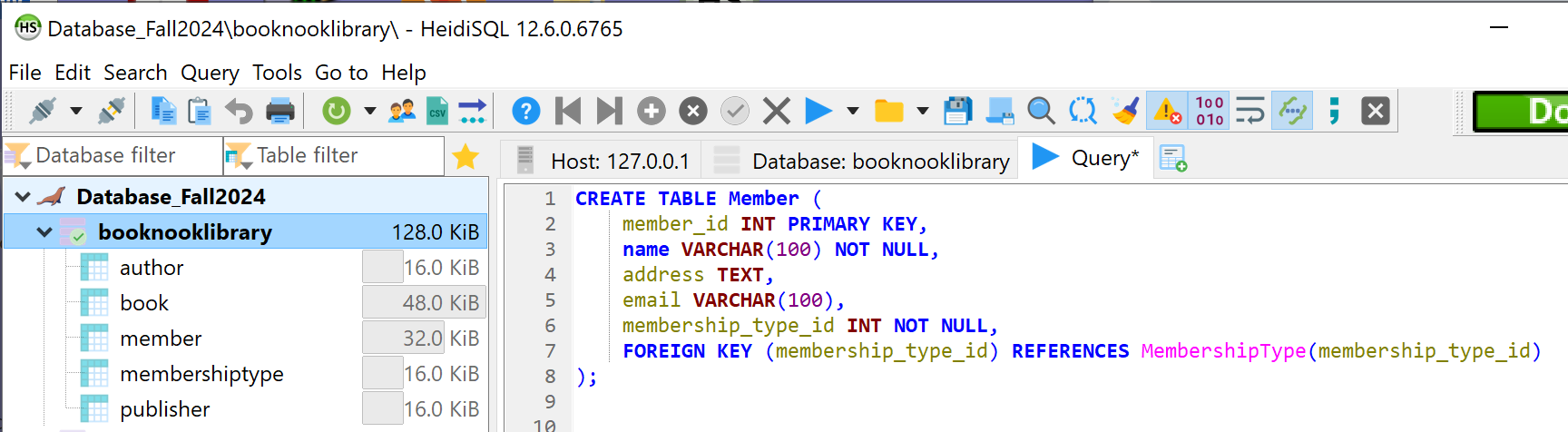
### **2.4 Create the MembershipType Table:**

| **CREATE TABLE MembershipType (  membership\_type\_id INT PRIMARY KEY,  type\_name VARCHAR(50) NOT NULL,  loan\_limit INT NOT NULL,  late\_fee\_rate DECIMAL(5,2) NOT NULL );** |
| --- |

****

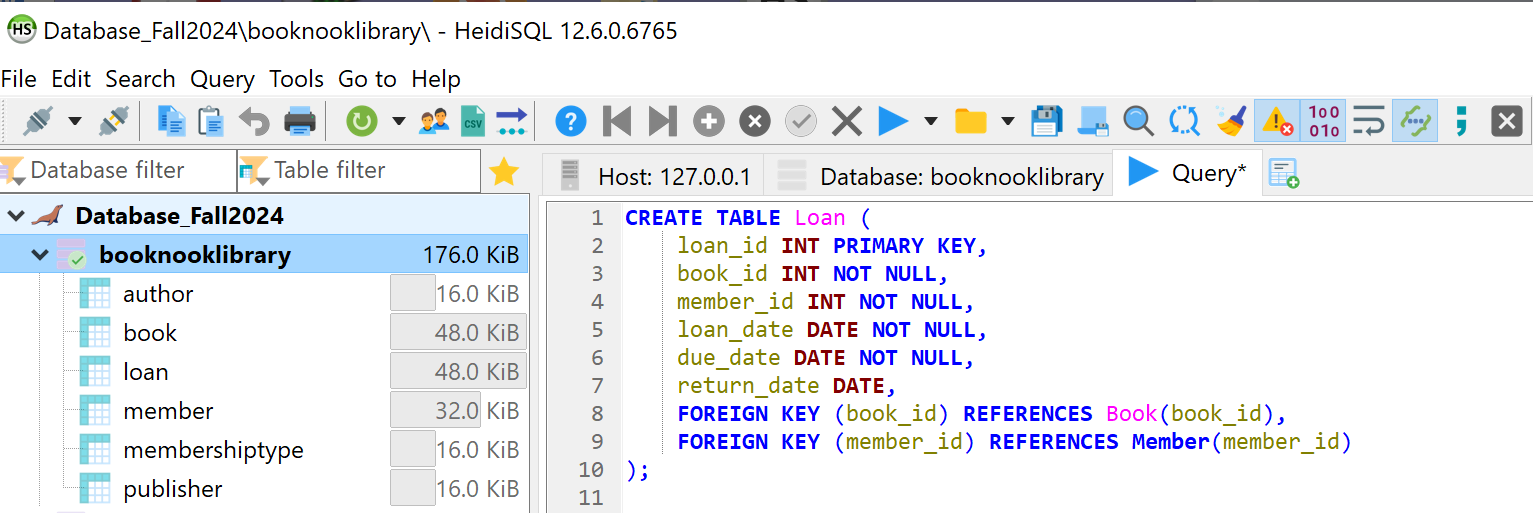
### **2.5 Create the Member Table:**

| **CREATE TABLE Member (  member\_id INT PRIMARY KEY,  name VARCHAR(100) NOT NULL,  address TEXT,  email VARCHAR(100),  membership\_type\_id INT NOT NULL,  FOREIGN KEY (membership\_type\_id) REFERENCES MembershipType(membership\_type\_id) );** |
| --- |

****

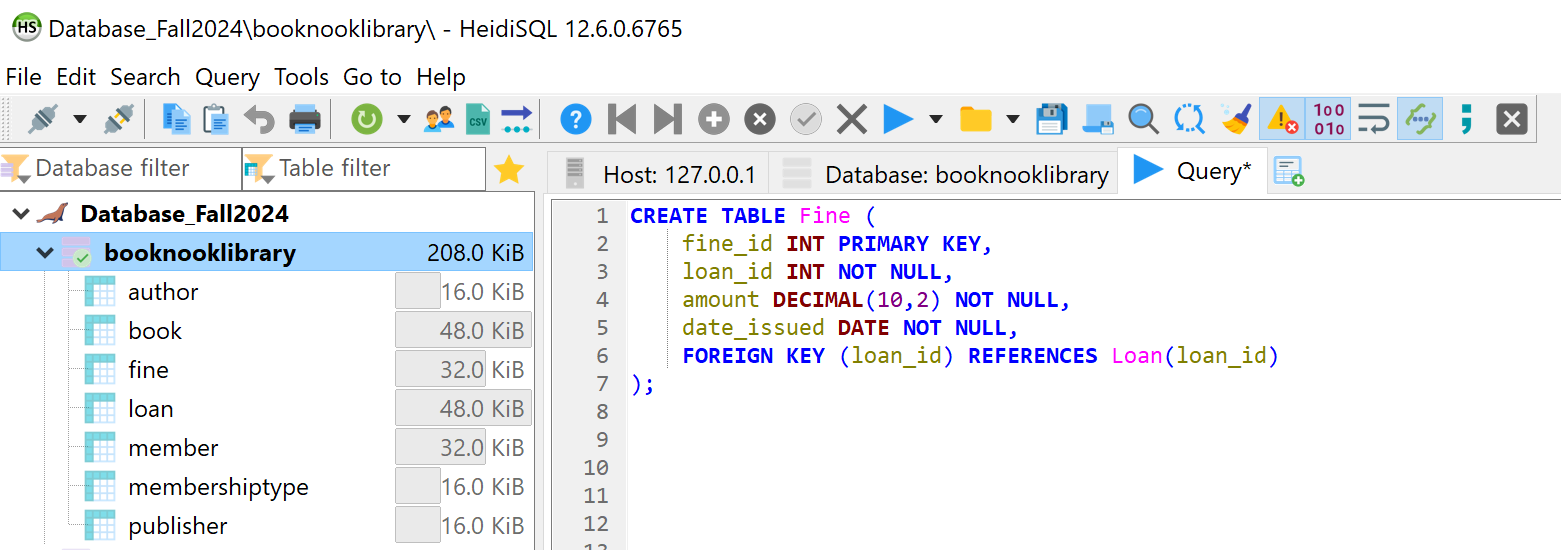
### **2.6 Create the Loan Table:**

| **CREATE TABLE Loan (  loan\_id INT PRIMARY KEY,  book\_id INT NOT NULL,  member\_id INT NOT NULL,  loan\_date DATE NOT NULL,  due\_date DATE NOT NULL,  return\_date DATE,  FOREIGN KEY (book\_id) REFERENCES Book(book\_id),  FOREIGN KEY (member\_id) REFERENCES Member(member\_id) );** |
| --- |

****

### **2.7 Create the Fine Table:**

| **CREATE TABLE Fine (  fine\_id INT PRIMARY KEY,  loan\_id INT NOT NULL,  amount DECIMAL(10,2) NOT NULL,  date\_issued DATE NOT NULL,  FOREIGN KEY (loan\_id) REFERENCES Loan(loan\_id) );** |
| --- |

****

## **3. Sample Data Insertion**

### **3.1 Insert Data into Author:**

| **INSERT INTO Author (author\_id, name, bio) VALUES (1, 'J.K. Rowling', 'Author of the Harry Potter series'), (2, 'George Orwell', 'Author of 1984 and Animal Farm');** |
| --- |

### **3.2 Insert Data into Publisher:**

| **INSERT INTO Publisher (publisher\_id, name, address) VALUES (1, 'Bloomsbury', 'London, UK'), (2, 'Secker & Warburg', 'London, UK');** |
| --- |

### **3.3 Insert Data into Book:**

| **INSERT INTO Book (book\_id, title, author\_id, genre, publication\_year, publisher\_id, quantity\_available) VALUES (1, 'Harry Potter and the Philosopher\'s Stone', 1, 'Fantasy', 1997, 1, 5), (2, '1984', 2, 'Dystopian', 1949, 2, 3), (3, 'The Hobbit', 3, 'Fantasy', 1937, 3, 7), (4, 'To Kill a Mockingbird', 4, 'Classic', 1960, 4, 4), (5, 'Pride and Prejudice', 5, 'Romance', 1813, 5, 6);** |
| --- |

### **3.4 Insert Data into MembershipType:**

| **INSERT INTO MembershipType (membership\_type\_id, type\_name, loan\_limit, late\_fee\_rate) VALUES (1, 'Regular', 5, 0.50), (2, 'Premium', 10, 0.25);** |
| --- |

### **3.5 Insert Data into Member:**

| **INSERT INTO Member (member\_id, name, address, email, membership\_type\_id) VALUES (1, 'Alice Smith', '123 Main St', 'alice@example.com', 1), (2, 'Bob Johnson', '456 Elm St', 'bob@example.com', 2);** |
| --- |

### **3.6 Insert Data into Loan:**

| **INSERT INTO Loan (loan\_id, book\_id, member\_id, loan\_date, due\_date, return\_date) VALUES (1, 1, 1, '2024-11-01', '2024-11-15', NULL), (2, 2, 2, '2024-11-05', '2024-11-20', NULL);** |
| --- |

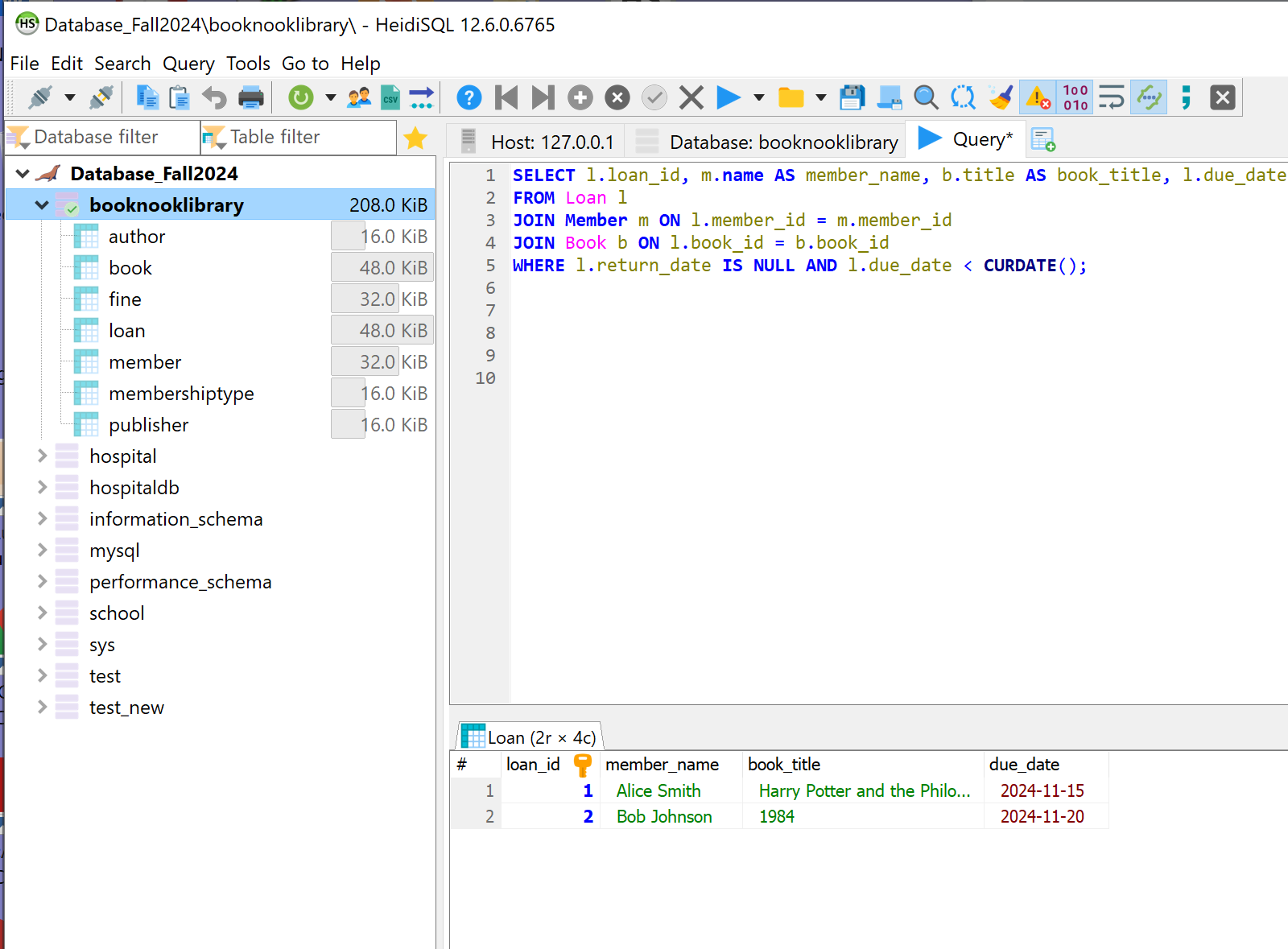
### **3.7 Insert Data into Fine:**

| **INSERT INTO Fine (fine\_id, loan\_id, amount, date\_issued) VALUES (1, 1, 2.50, '2024-11-17');** |
| --- |

## 4. Query Examples

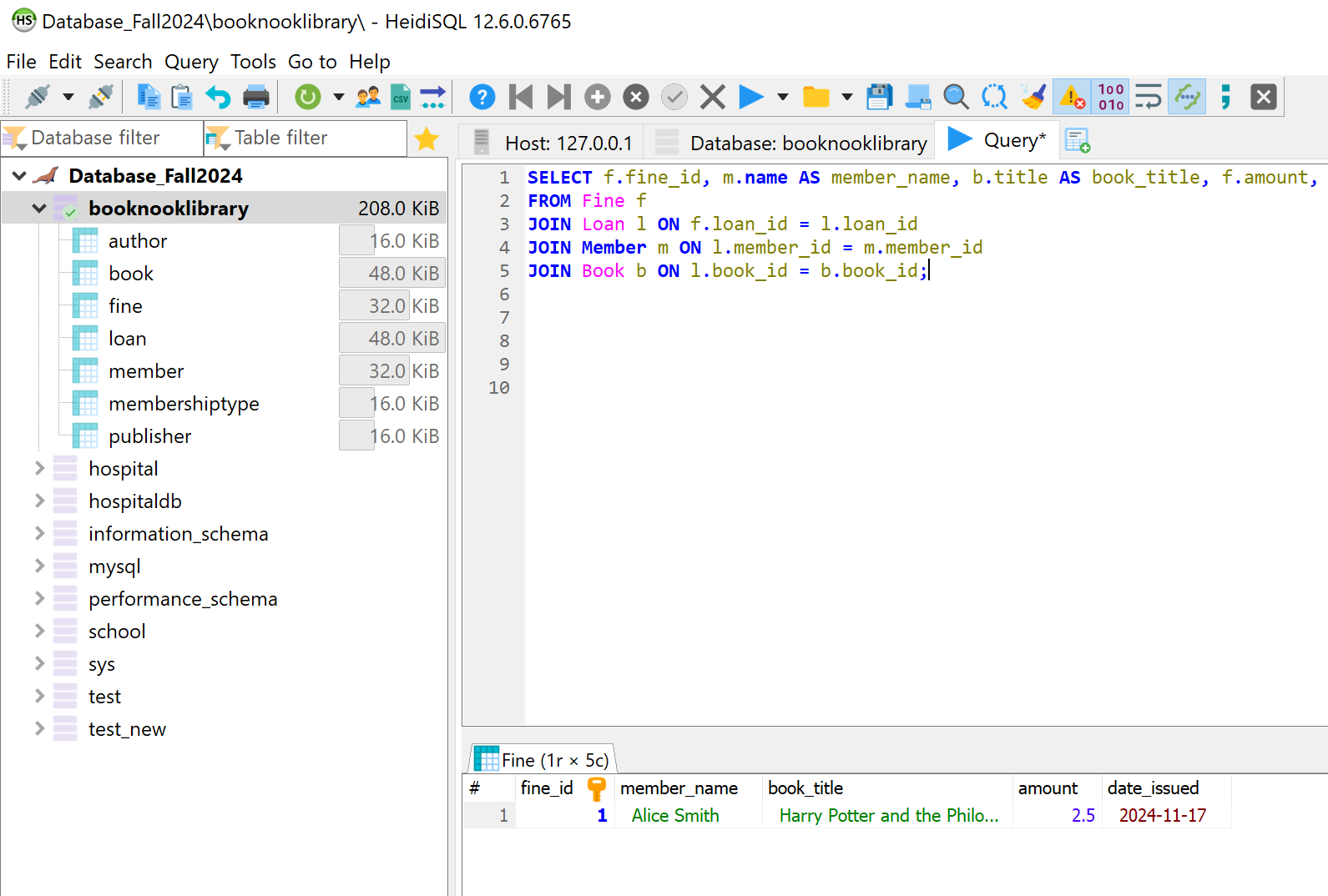
### **4.1 View All Overdue Loans:**

| **SELECT l.loan\_id, m.name AS member\_name, b.title AS book\_title, l.due\_date FROM Loan l JOIN Member m ON l.member\_id = m.member\_id JOIN Book b ON l.book\_id = b.book\_id WHERE l.return\_date IS NULL AND l.due\_date < CURDATE();** |
| --- |

****

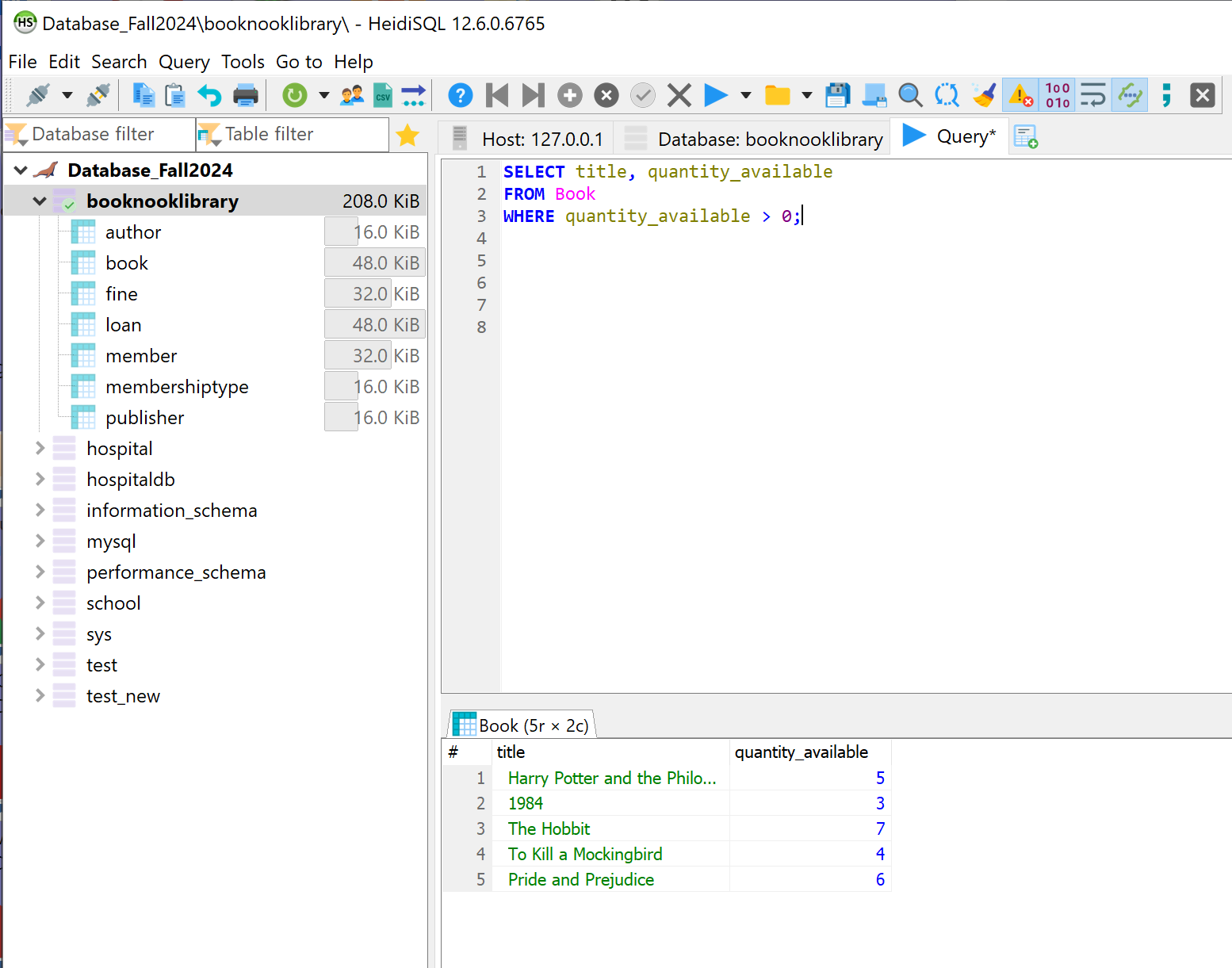
### **4.2 View Fine Details for Overdue Loans:**

| **SELECT f.fine\_id, m.name AS member\_name, b.title AS book\_title, f.amount, f.date\_issued FROM Fine f JOIN Loan l ON f.loan\_id = l.loan\_id JOIN Member m ON l.member\_id = m.member\_id JOIN Book b ON l.book\_id = b.book\_id;** |
| --- |

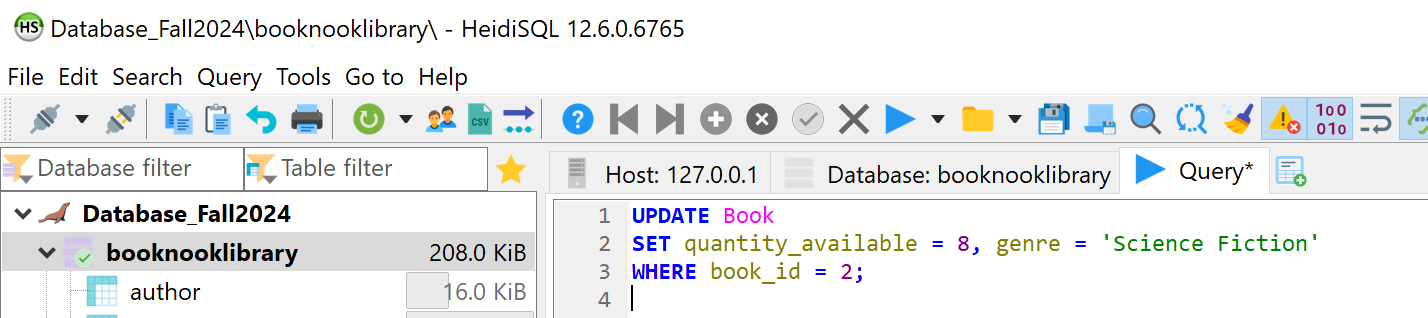
****

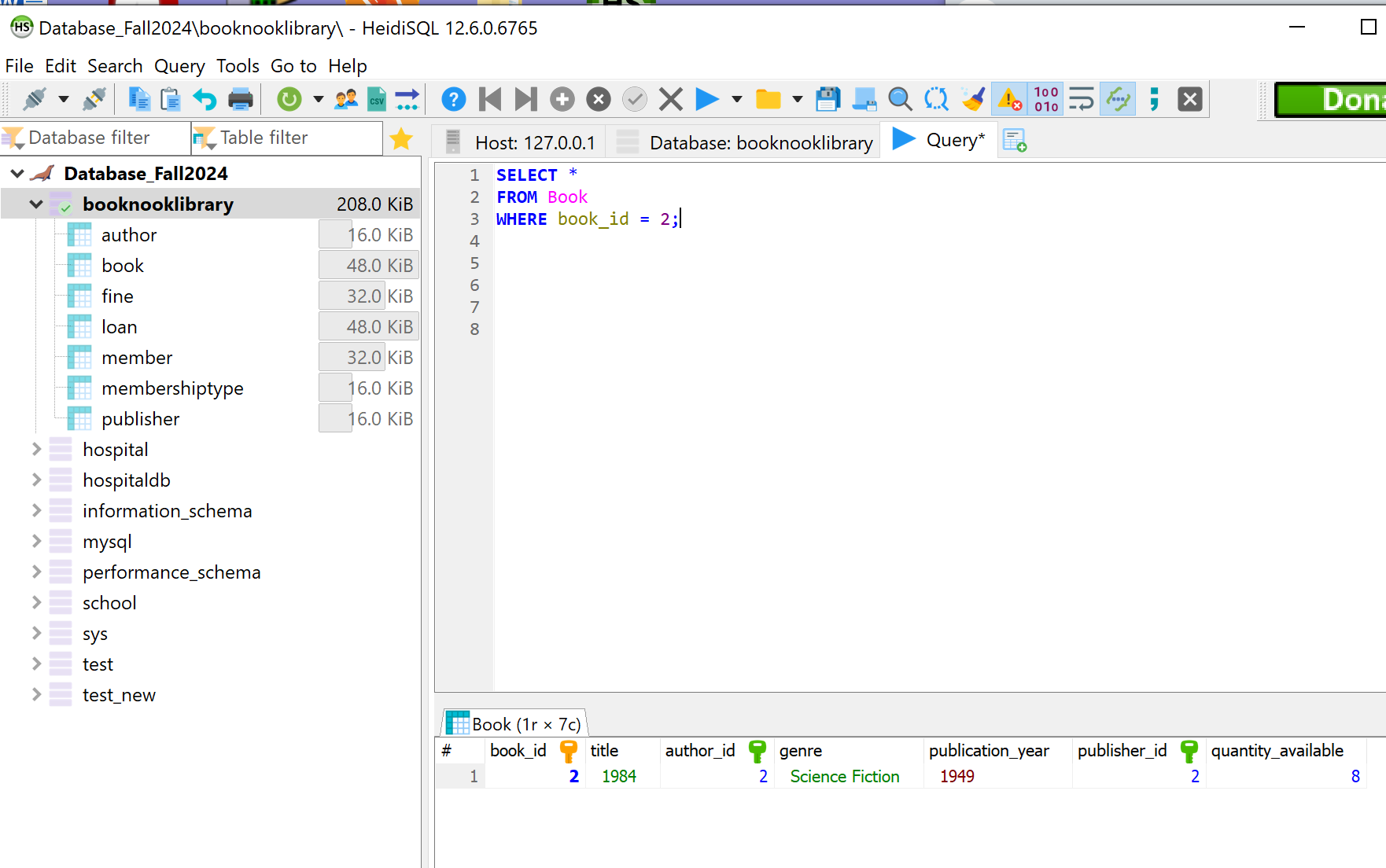
### **4.3 Check Books Availability:**

| **SELECT title, quantity\_available FROM Book WHERE quantity\_available > 0;** |
| --- |

****

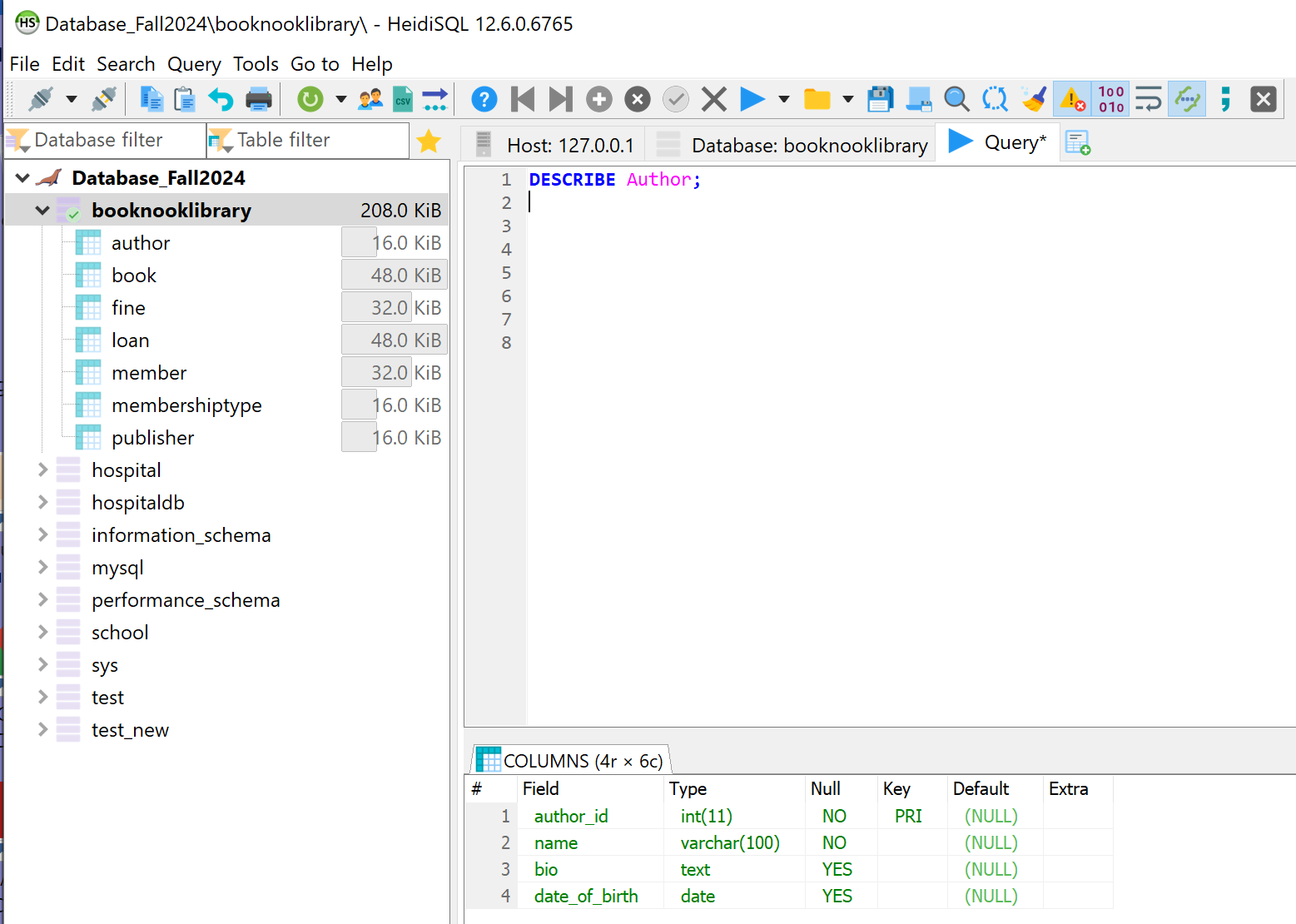
### **4.4 Update on the book:**

****

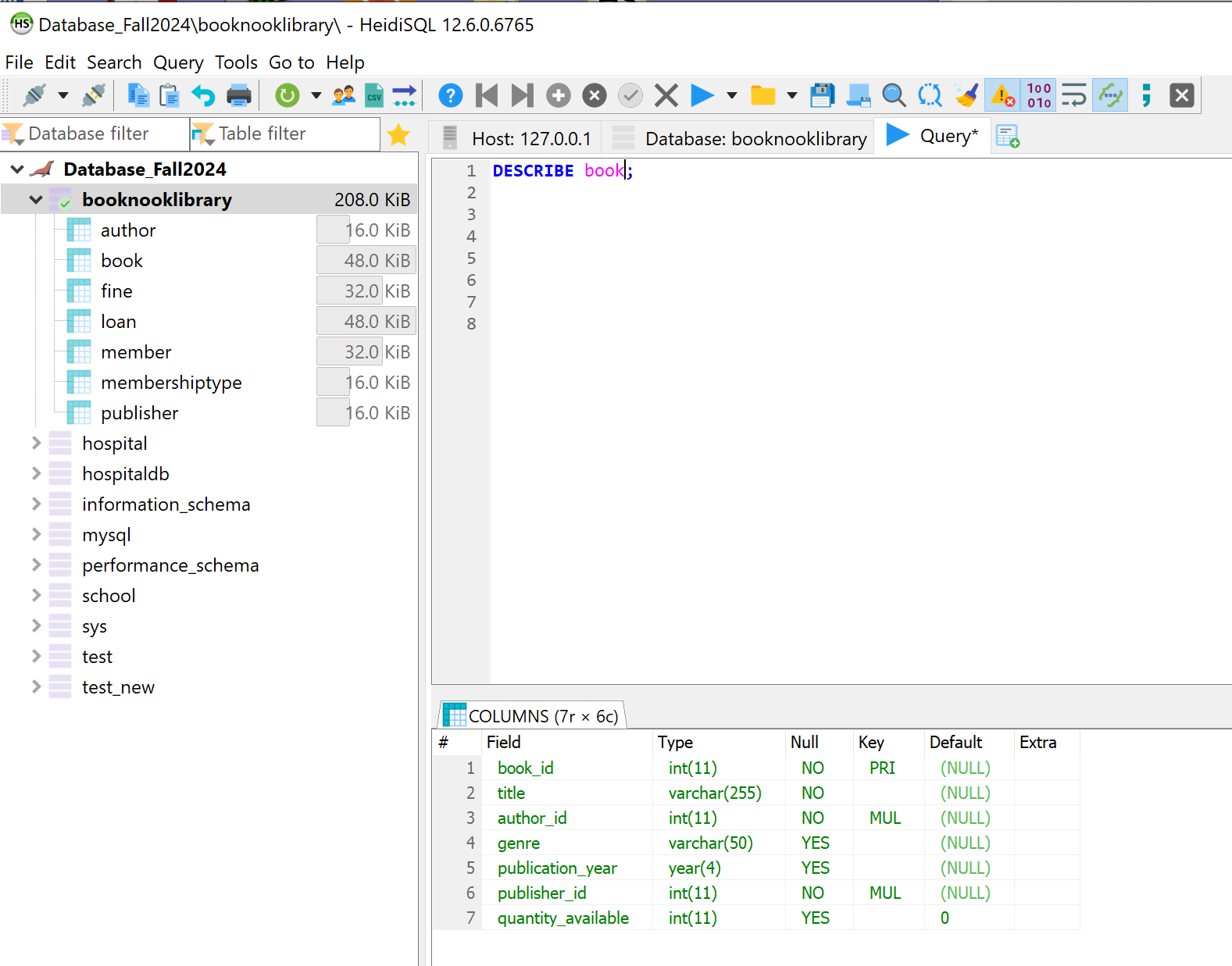
****

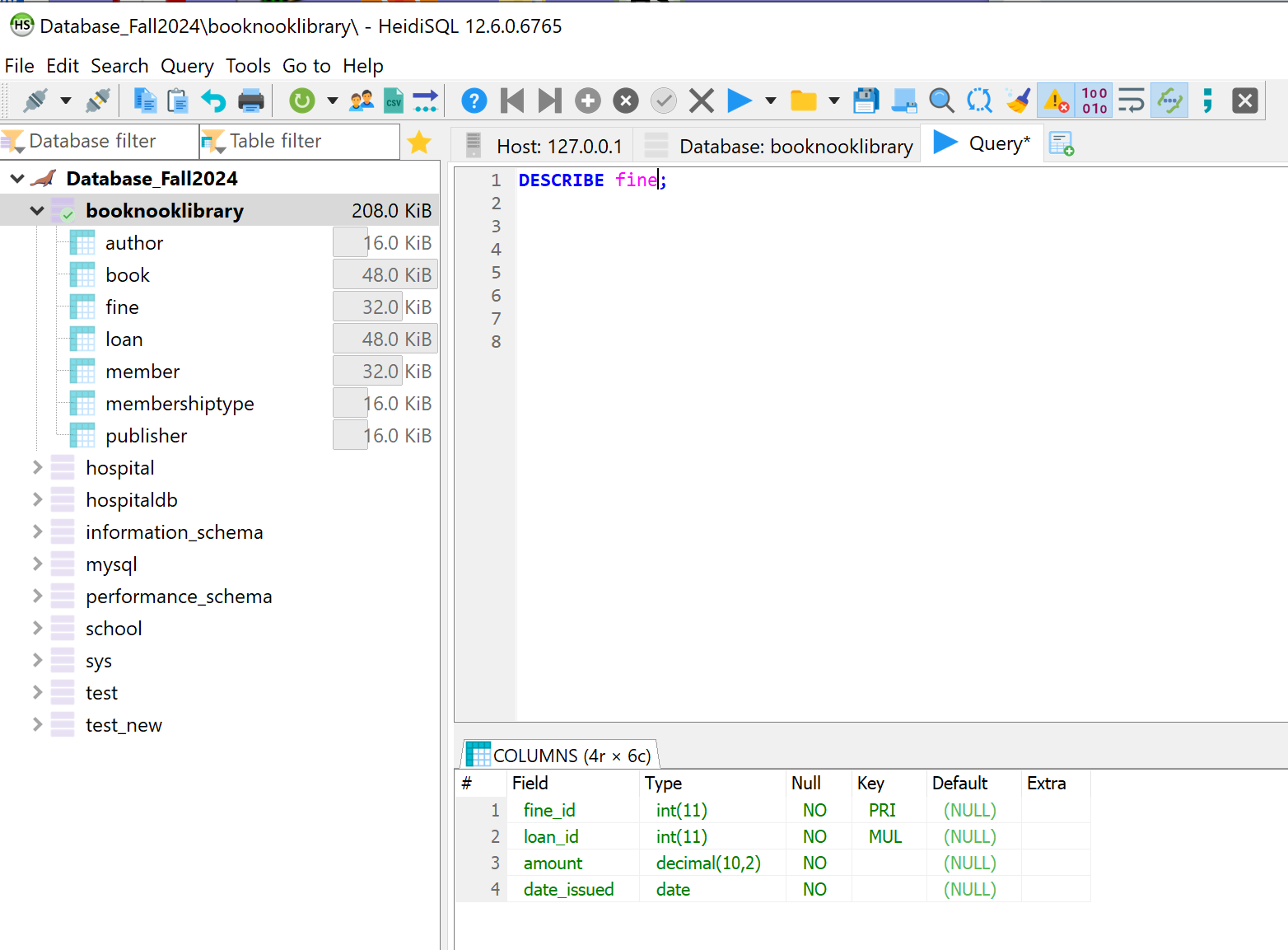
### **4.5 ALTER TABLE for the author:**

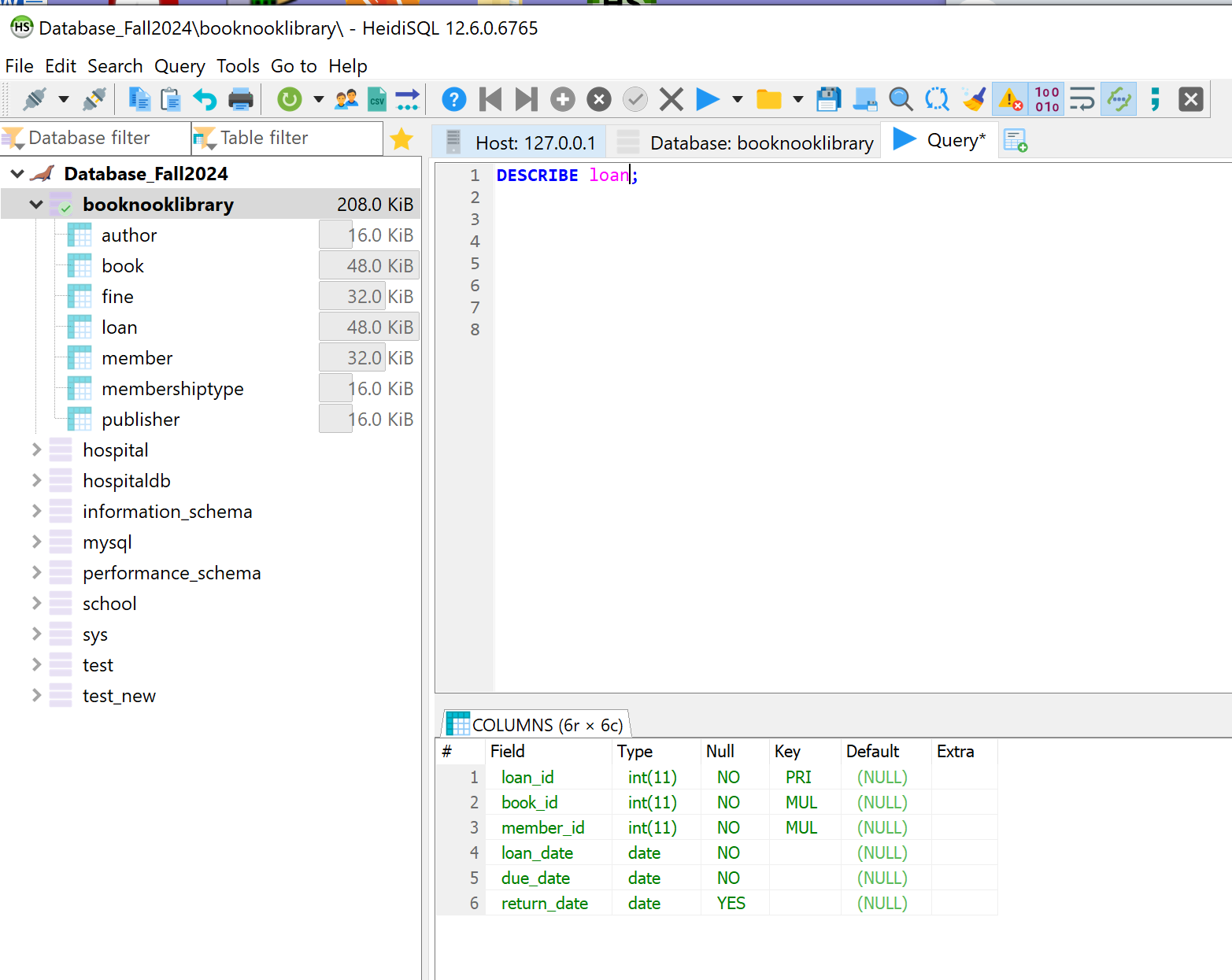
| ALTER TABLE Author ADD COLUMN date\_of\_birth DATE; |
| --- |

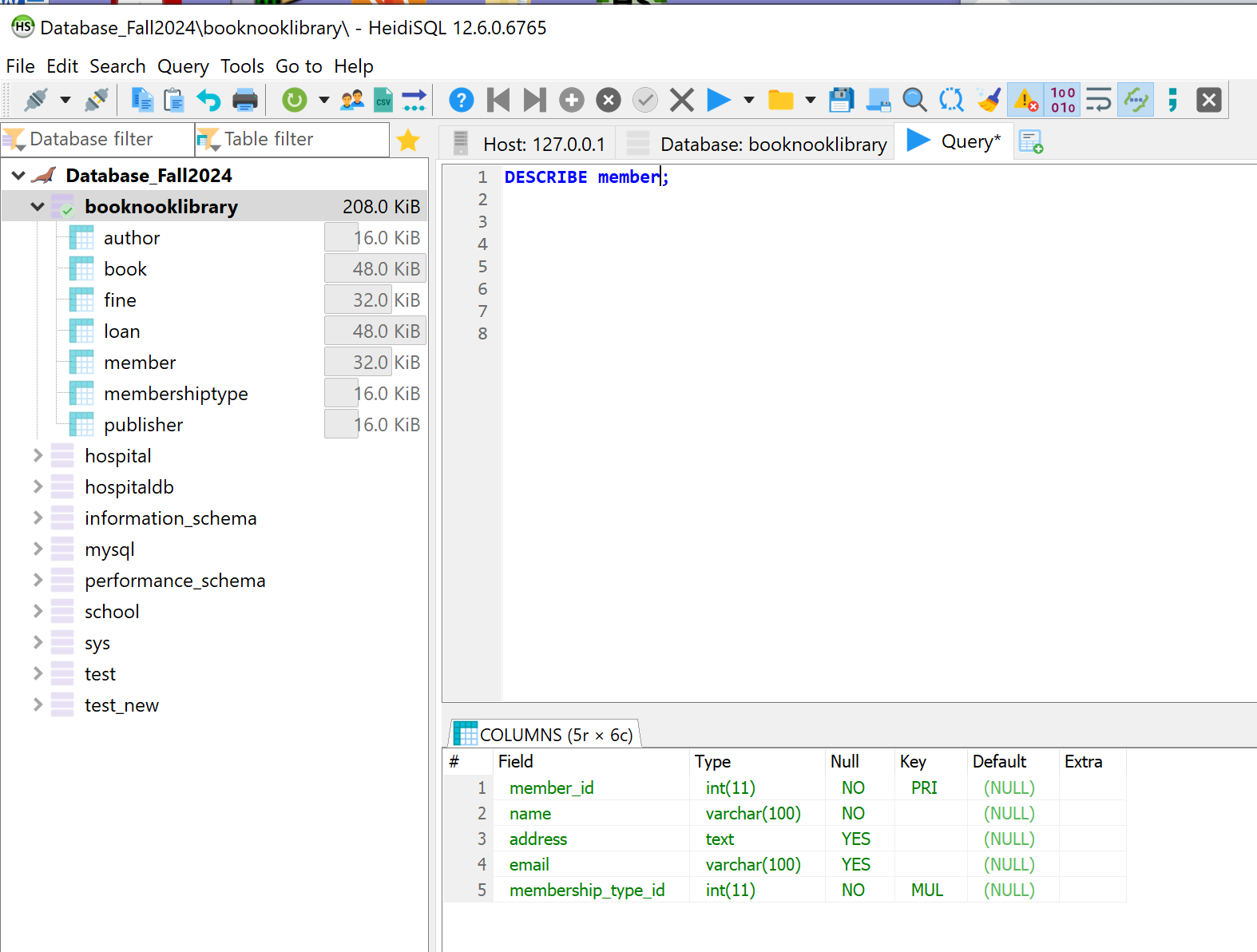
****

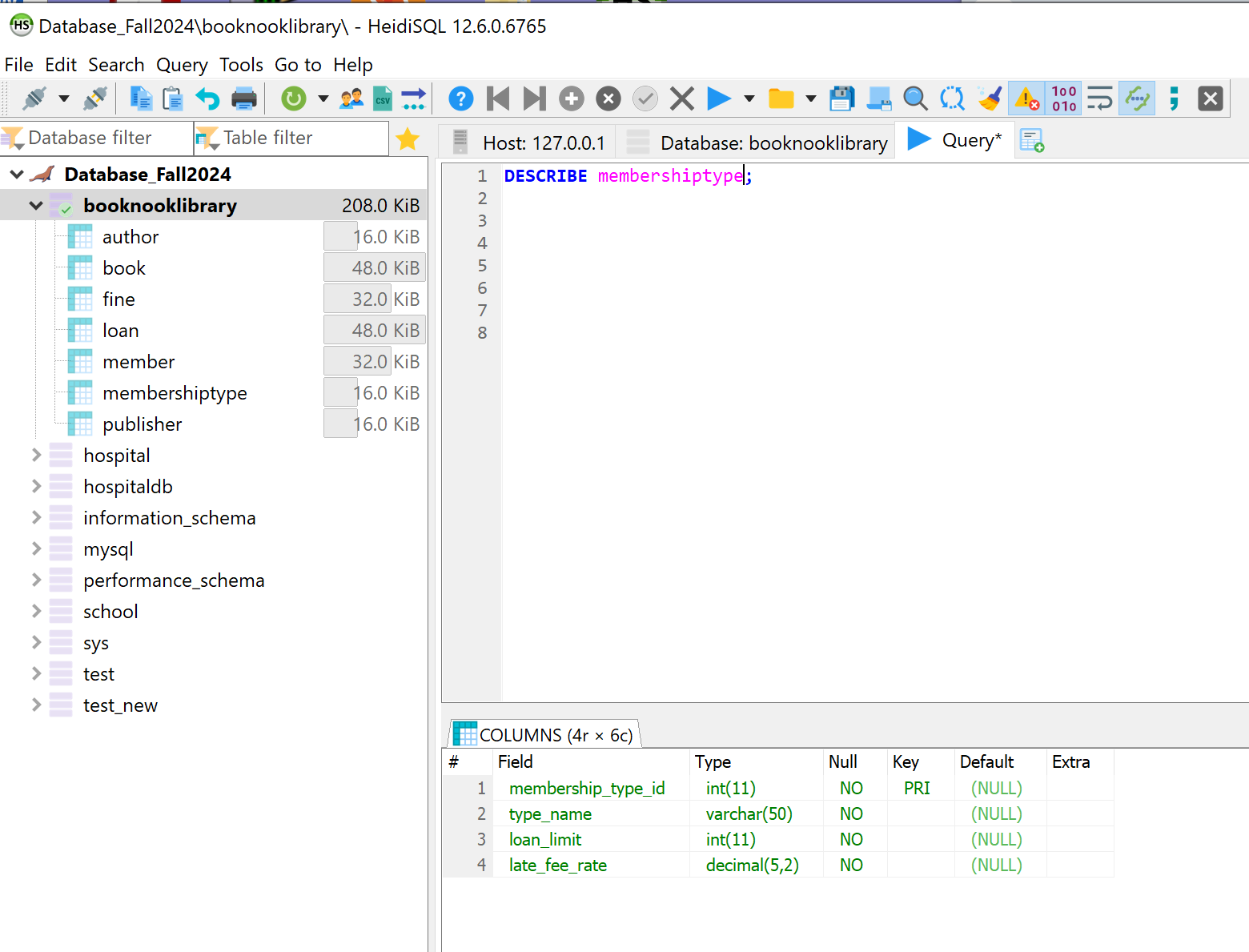
### **4.6 Describe the attributes:**

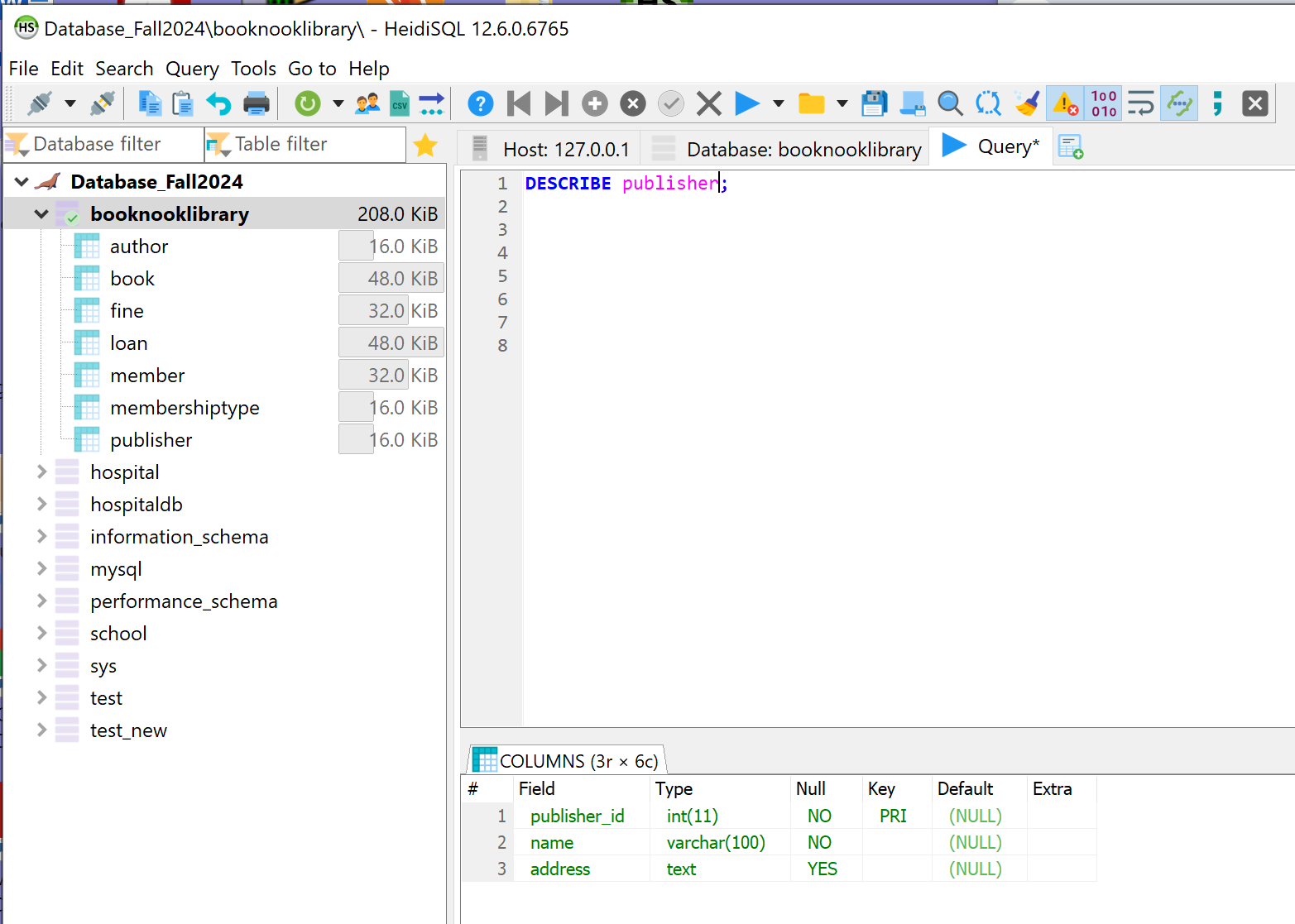
****

****

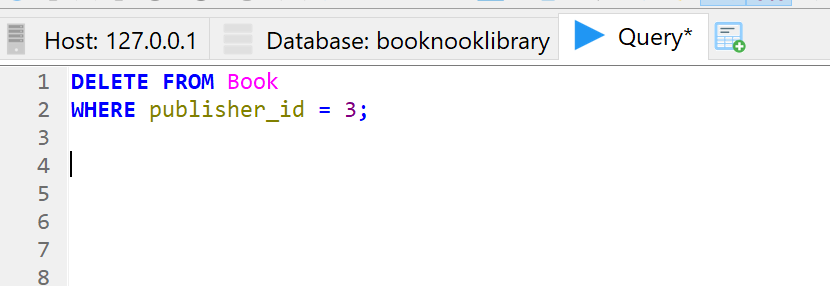
****

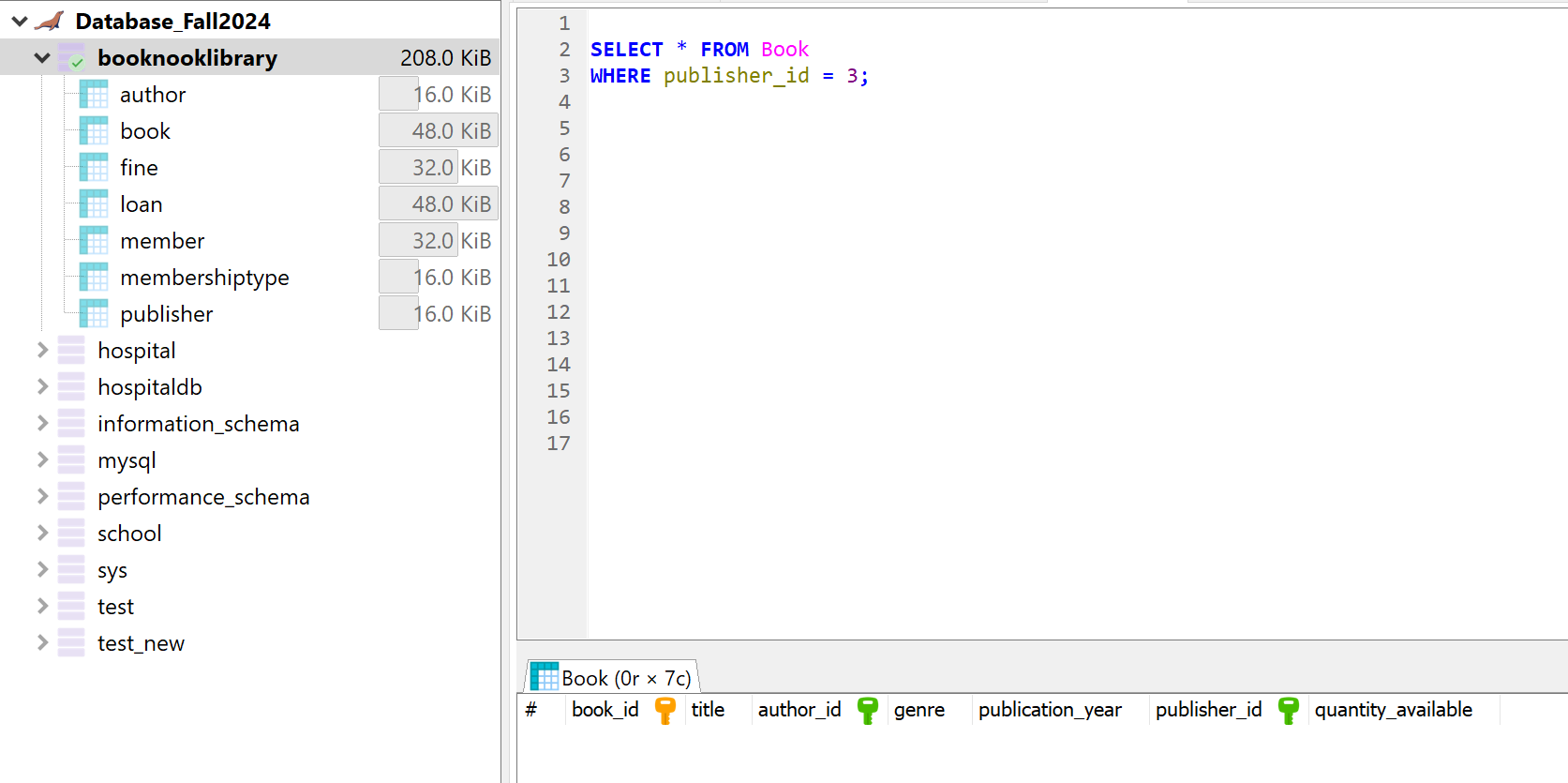
****

****

****

### **4.7 Delete from Book:**

****

****