

SQL Day 1 Assignments:

Assignment 1: Analyse a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram reflects proper normalization up to the third normal form.

Business Scenario:

A banking system needs to manage customer accounts, transactions, branches, and employees. Each customer can have one or more accounts, each account can have multiple transactions associated with it. Each branch employs several employees who may manage multiple accounts and transactions.

Entities:

1. Customer
2. Account
3. Transaction
4. Branch
5. Employee

Attributes:

1. Customer:
 - Customer_ID (Primary Key)
 - Name
 - Address
 - Phone_Number
 - Email
2. Account:
 - Account_ID (Primary Key)
 - Customer_ID (Foreign Key)

- Account_Type
- Balance
- Open_Date

3. Transaction:

- Transaction_ID (Primary Key)
- Account_ID (Foreign Key)
- Transaction_Type
- Amount
- Transaction_Date

4. Branch:

- Branch_ID (Primary Key)
- Branch_Name
- Location

5. Employee:

- Employee_ID (Primary Key)
- Branch_ID (Foreign Key)
- Name
- Position
- Salary

Relationships:

1. Customer - Account (One-to-Many):

- Each customer can have multiple accounts, but each account belongs to only one customer.

2. Account - Transaction (One-to-Many):

- Each account can have multiple transactions, but each transaction belongs to only one account.

3. Branch - Employee (One-to-Many):

- Each branch can have multiple employees, but each employee belongs to only one branch.

Cardinality:

1. Customer - Account:

- One customer can have multiple accounts (1:N).

2. Account - Transaction:

- One account can have multiple transactions (1:N).

3. Branch - Employee:

- One branch can have multiple employees (1:N).

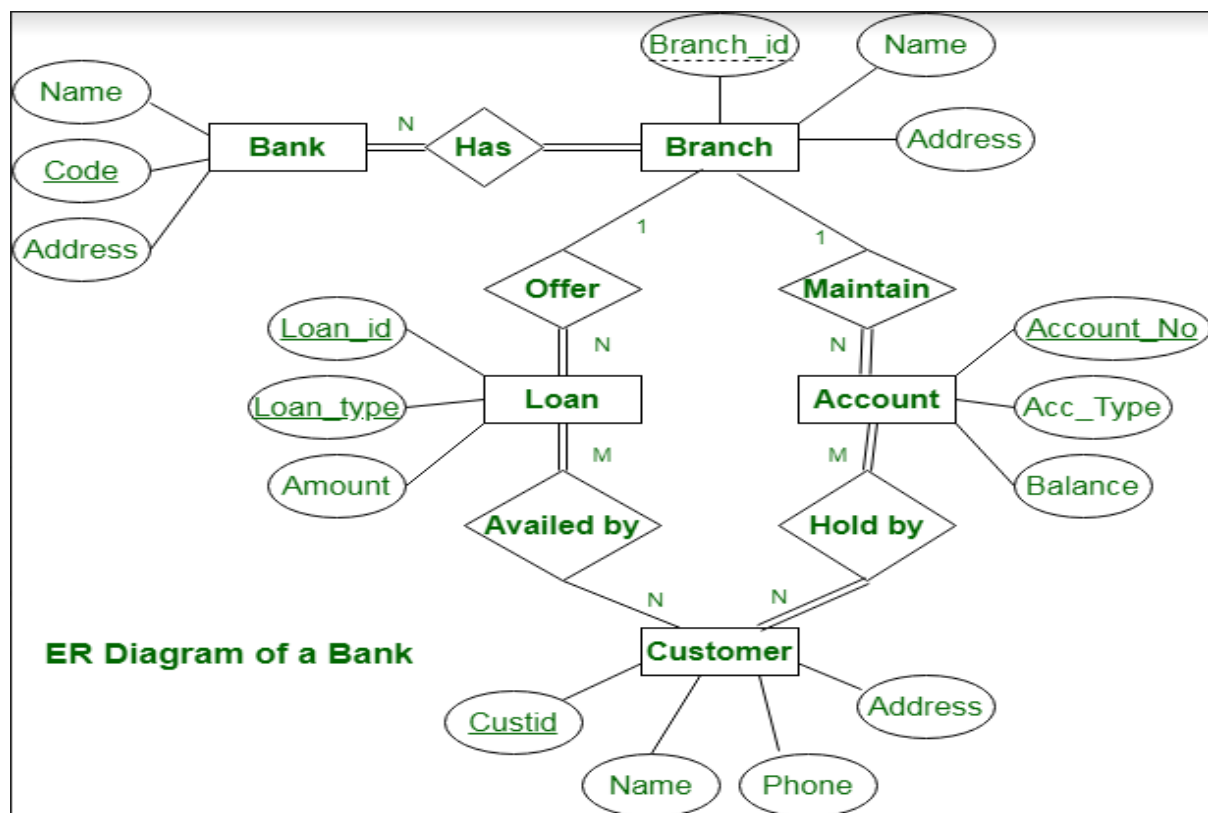
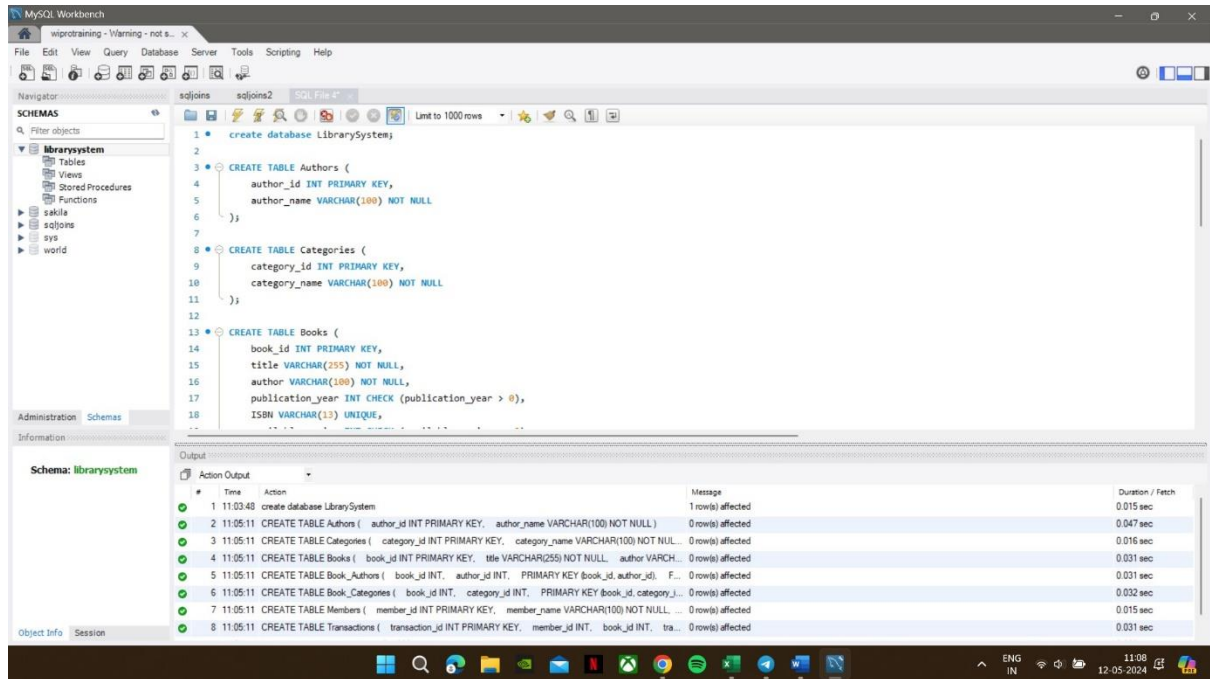


Figure 1/ER diagram for Banking System Business Model

Assignment 2: Design a database schema for a library system, including tables, fields, and constraints like NOT NULL, UNIQUE, and CHECK. Include primary and foreign keys to establish relationships between tables.

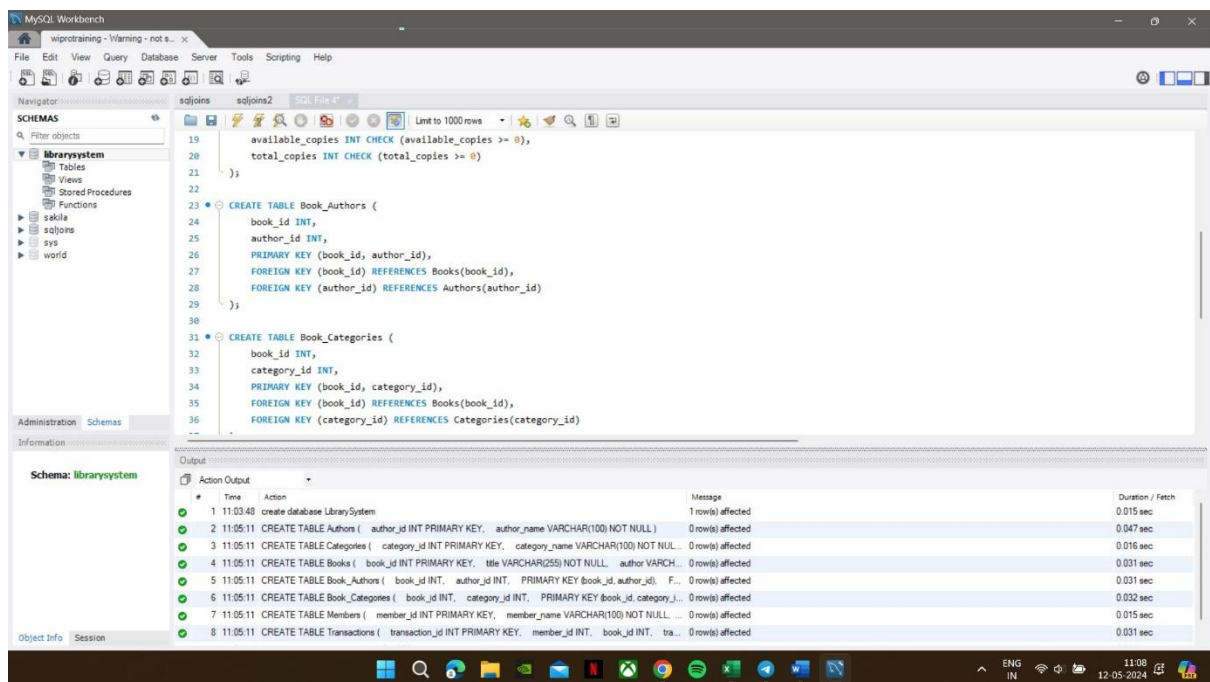
the SQL code to create the tables with the described schema and constraints:



The screenshot shows the MySQL Workbench interface with the 'SQL File Editor' open. The code defines the 'LibrarySystem' database and three tables: 'Authors', 'Categories', and 'Books'. The 'Authors' table has a primary key on 'author_id' and a 'VARCHAR(100)' field for 'author_name'. The 'Categories' table has a primary key on 'category_id' and a 'VARCHAR(100)' field for 'category_name'. The 'Books' table has a primary key on 'book_id', a 'VARCHAR(255)' field for 'title', a 'VARCHAR(100)' field for 'author', a 'CHECK' constraint for 'publication_year' (greater than 0), and a 'UNIQUE' constraint on the 'ISBN' field.

```
1 create database LibrarySystem;
2
3 CREATE TABLE Authors (
4   author_id INT PRIMARY KEY,
5   author_name VARCHAR(100) NOT NULL
6 );
7
8 CREATE TABLE Categories (
9   category_id INT PRIMARY KEY,
10  category_name VARCHAR(100) NOT NULL
11 );
12
13 CREATE TABLE Books (
14   book_id INT PRIMARY KEY,
15   title VARCHAR(255) NOT NULL,
16   author VARCHAR(100) NOT NULL,
17   publication_year INT CHECK (publication_year > 0),
18   ISBN VARCHAR(13) UNIQUE,
```

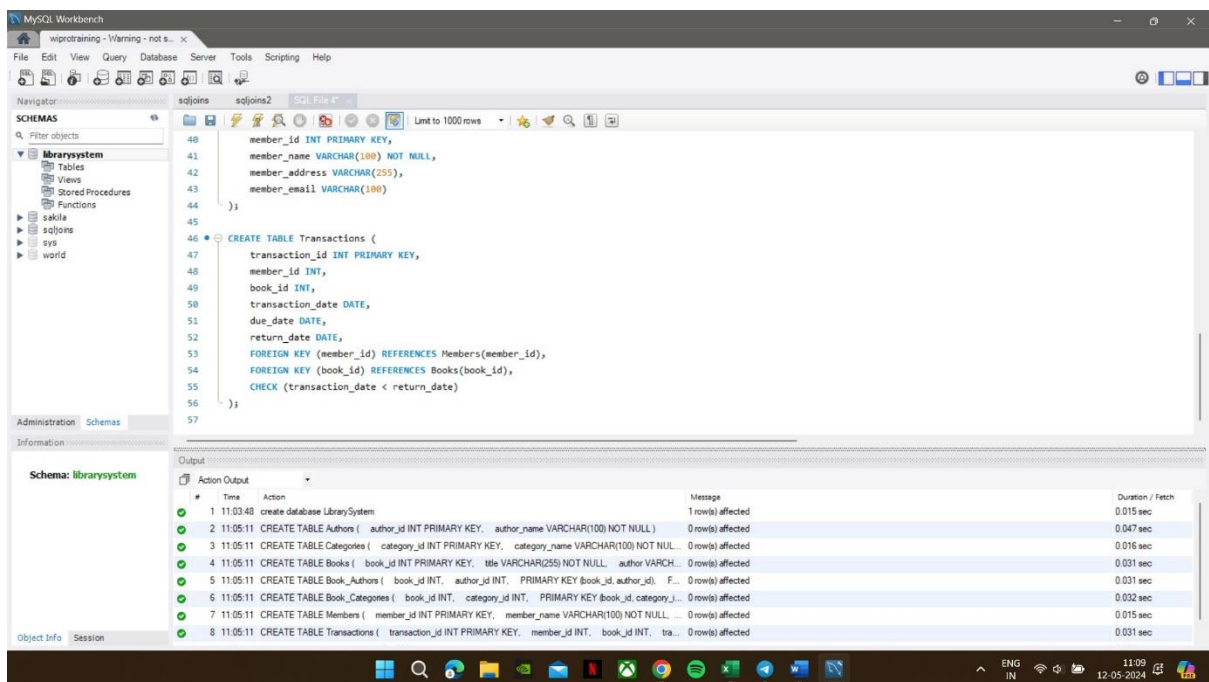
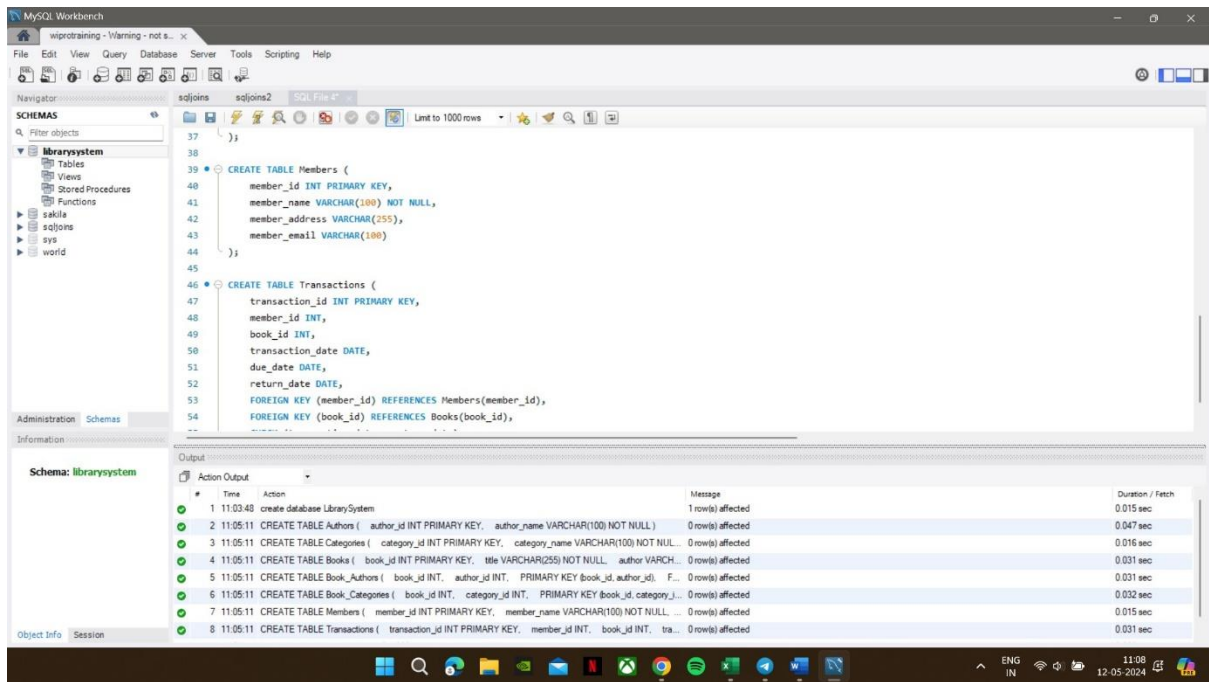
#	Time	Action	Message	Duration / Fetch
1	11:03:40	create database LibrarySystem	1 row(s) affected	0.015 sec
2	11:05:11	CREATE TABLE Authors (author_id INT PRIMARY KEY, author_name VARCHAR(100) NOT NULL)	0 row(s) affected	0.047 sec
3	11:05:11	CREATE TABLE Categories (category_id INT PRIMARY KEY, category_name VARCHAR(100) NOT NULL)	0 row(s) affected	0.016 sec
4	11:05:11	CREATE TABLE Books (book_id INT PRIMARY KEY, title VARCHAR(255) NOT NULL, author VARCHAR(100) NOT NULL, publication_year INT CHECK (publication_year > 0), ISBN VARCHAR(13) UNIQUE)	0 row(s) affected	0.031 sec
5	11:05:11	CREATE TABLE Book_Authors (book_id INT, author_id INT, PRIMARY KEY (book_id, author_id), FOREIGN KEY (book_id) REFERENCES Books(book_id), FOREIGN KEY (author_id) REFERENCES Authors(author_id))	0 row(s) affected	0.031 sec
6	11:05:11	CREATE TABLE Book_Categories (book_id INT, category_id INT, PRIMARY KEY (book_id, category_id), FOREIGN KEY (book_id) REFERENCES Books(book_id), FOREIGN KEY (category_id) REFERENCES Categories(category_id))	0 row(s) affected	0.032 sec
7	11:05:11	CREATE TABLE Members (member_id INT PRIMARY KEY, member_name VARCHAR(100) NOT NULL, member_address VARCHAR(255) NOT NULL)	0 row(s) affected	0.015 sec
8	11:05:11	CREATE TABLE Transactions (transaction_id INT PRIMARY KEY, member_id INT, book_id INT, transaction_date DATE, PRIMARY KEY (transaction_id, member_id, book_id), FOREIGN KEY (member_id) REFERENCES Members(member_id), FOREIGN KEY (book_id) REFERENCES Books(book_id))	0 row(s) affected	0.031 sec



The screenshot shows the continuation of the SQL code in the 'SQL File Editor'. It adds constraints and foreign keys to the 'Book_Authors' and 'Book_Categories' tables. The 'Book_Authors' table has a primary key on ('book_id', 'author_id'), a foreign key on 'book_id' referencing 'Books(book_id)', and a foreign key on 'author_id' referencing 'Authors(author_id)'. The 'Book_Categories' table has a primary key on ('book_id', 'category_id'), a foreign key on 'book_id' referencing 'Books(book_id)', and a foreign key on 'category_id' referencing 'Categories(category_id)'.

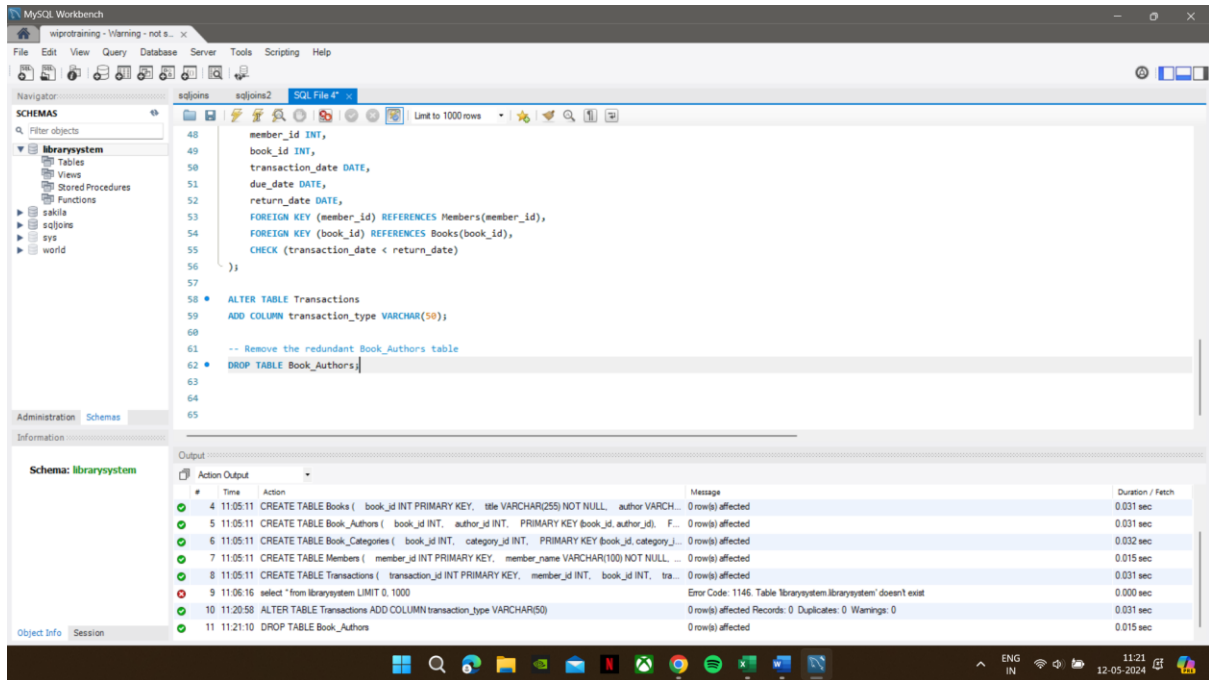
```
19   available_copies INT CHECK (available_copies >= 0),
20   total_copies INT CHECK (total_copies >= 0)
21 );
22
23 CREATE TABLE Book_Authors (
24   book_id INT,
25   author_id INT,
26   PRIMARY KEY (book_id, author_id),
27   FOREIGN KEY (book_id) REFERENCES Books(book_id),
28   FOREIGN KEY (author_id) REFERENCES Authors(author_id)
29 );
30
31 CREATE TABLE Book_Categories (
32   book_id INT,
33   category_id INT,
34   PRIMARY KEY (book_id, category_id),
35   FOREIGN KEY (book_id) REFERENCES Books(book_id),
36   FOREIGN KEY (category_id) REFERENCES Categories(category_id)
37 );
```

#	Time	Action	Message	Duration / Fetch
1	11:03:40	create database LibrarySystem	1 row(s) affected	0.015 sec
2	11:05:11	CREATE TABLE Authors (author_id INT PRIMARY KEY, author_name VARCHAR(100) NOT NULL)	0 row(s) affected	0.047 sec
3	11:05:11	CREATE TABLE Categories (category_id INT PRIMARY KEY, category_name VARCHAR(100) NOT NULL)	0 row(s) affected	0.016 sec
4	11:05:11	CREATE TABLE Books (book_id INT PRIMARY KEY, title VARCHAR(255) NOT NULL, author VARCHAR(100) NOT NULL, publication_year INT CHECK (publication_year > 0), ISBN VARCHAR(13) UNIQUE)	0 row(s) affected	0.031 sec
5	11:05:11	CREATE TABLE Book_Authors (book_id INT, author_id INT, PRIMARY KEY (book_id, author_id), FOREIGN KEY (book_id) REFERENCES Books(book_id), FOREIGN KEY (author_id) REFERENCES Authors(author_id))	0 row(s) affected	0.031 sec
6	11:05:11	CREATE TABLE Book_Categories (book_id INT, category_id INT, PRIMARY KEY (book_id, category_id), FOREIGN KEY (book_id) REFERENCES Books(book_id), FOREIGN KEY (category_id) REFERENCES Categories(category_id))	0 row(s) affected	0.032 sec
7	11:05:11	CREATE TABLE Members (member_id INT PRIMARY KEY, member_name VARCHAR(100) NOT NULL, member_address VARCHAR(255) NOT NULL)	0 row(s) affected	0.015 sec
8	11:05:11	CREATE TABLE Transactions (transaction_id INT PRIMARY KEY, member_id INT, book_id INT, transaction_date DATE, PRIMARY KEY (transaction_id, member_id, book_id), FOREIGN KEY (member_id) REFERENCES Members(member_id), FOREIGN KEY (book_id) REFERENCES Books(book_id))	0 row(s) affected	0.031 sec



This SQL above execution of code will create the necessary tables with the specified fields, constraints, primary keys, and foreign keys to establish the relationships between them as described.

Assignment 4: Write SQL statements to CREATE a new database and tables that reflect the library schema you designed earlier. Use ALTER statements to modify the table structures and DROP statements to remove a redundant table.



ALTER and DROP statements used above picture

These SQL statements will create a new database named **library_system**, create tables reflecting the library schema, modify the structure of the **Transactions** table by adding a new column, and remove the redundant **Book_Authors** table.

Assignment 5: Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDEX statement to remove the index and analyze the impact on query execution.

Let's demonstrate the creation of an index on a table and discuss its impact on query performance. We'll use a hypothetical scenario where we have a large table named **Books** and frequently query it based on the **title** column.

Step 1: Creating an Index:

We'll create an index on the **title** column of the **Books** table:

```
sql
```

```
-- Create an index on the 'title' column of the 'Books' table CREATE INDEX idx_title ON Books(title);
```

This statement creates an index named **idx_title** on the **title** column of the **Books** table.

Step 2: Query Performance Improvement:

By creating an index on the **title** column, the database management system (DBMS) creates a separate data structure (the index) that stores the values of the **title** column along with pointers to the corresponding rows in the **Books** table. When querying based on the **title** column, the DBMS can use this index to quickly locate the relevant rows, rather than scanning the entire table sequentially.

As a result, queries that involve filtering, sorting, or joining based on the **title** column will likely see significant performance improvements. The index allows the DBMS to perform index scans or index seeks, which are much faster operations compared to full table scans.

Step 3: Dropping the Index:

Now, let's drop the index we created:

```
sql
```

```
-- Drop the index on the 'title' column of the 'Books' table DROP INDEX idx_title ON Books;
```

Impact on Query Execution:

Without the index on the **title** column, queries that rely on filtering, sorting, or joining based on the **title** column may experience degraded performance. The DBMS will have to resort to full table scans, where it needs to examine every row in the table to find the desired data. This can be resource-intensive and time-consuming, especially for large tables.

In summary, creating an index on a column can significantly improve query performance by allowing the DBMS to quickly locate relevant rows. However, indexes come with some overhead in terms of storage space and maintenance, so they should be used judiciously based on the specific requirements of the application.