# Business Requirements Document (BRD)

# 📄 Business Requirements Document (BRD)

**Team members:** [

1- Mohammed Elsaeid Elantably _

2- Mohammed M. abo-raia _

3- Yamen Amr Ahmed _

4- Ahmed Ashraf Galal _

5- Mahmoud Hamed Mohamed _

6- مصطفي مجاهد مصطفي

]

**Date:** July 2025

---

## Project Title

**Library Management System Database Design**

---

## Project Objective

To design and implement a relational database for a library management system that tracks books, members, borrowing activities, and fines. The database must follow normalization principles to eliminate redundancy and ensure data integrity.

---

## Scope

The system will support the following:

- Store information about books available in the library
- Manage member registrations and details
- Track book borrowings and returns
- Calculate and record overdue fines

---

## Business Requirements

### 1. Book Management

The system must store the following book details:

- Book ID
- Title
- Author
- Genre
- Publication Year
- ISBN (must be unique)
- Copies Available

---

### 2. Member Management

Each library member record must include:

- Member ID
- Full Name
- Email (unique)
- Phone Number
- Address
- Membership Start Date

---

### 3. Borrowing Records

Each borrowing event must capture:

- Borrowing ID
- Member ID
- Book ID
- Borrow Date
- Due Date
- Return Date (nullable until returned)

---

#### 4. Fine Management

If a book is returned late, a fine record must be created with:

- Fine ID
- Borrowing ID
- Amount
- Paid Status (Yes/No)

---

## Business Constraints

#### Book Constraints

- ISBN must be unique
- Copies Available must be ≥ 0

#### Member Constraints

- Email must be unique
- Phone number must not exceed 15 characters

#### Borrowing Constraints

- Each borrowing must link to an existing Member and Book

- Due Date must be after Borrow Date
- Return Date must not be earlier than Borrow Date

**Fine Constraints**

- Amount must be > 0
- Fine must reference an existing BorrowingID

---

## Expected Outcomes

- A fully normalized and relational database schema
- SQL DDL scripts to create the structure
- Constraints and keys to enforce data integrity
- Ready for integration with a library management front-end system

---

## ✅ Test Your Understanding

Now that you've reviewed this case, you can try answering the following:

1. **List the main entities and their relationships in an ERD-style format.**
2. **Design the SQL DDL to create this schema. ✅**

3. **Add sample data for at least one book, one member, one borrowing, and one fine.✅**

4. **List one use case (e.g., return a book and apply a fine).✅**

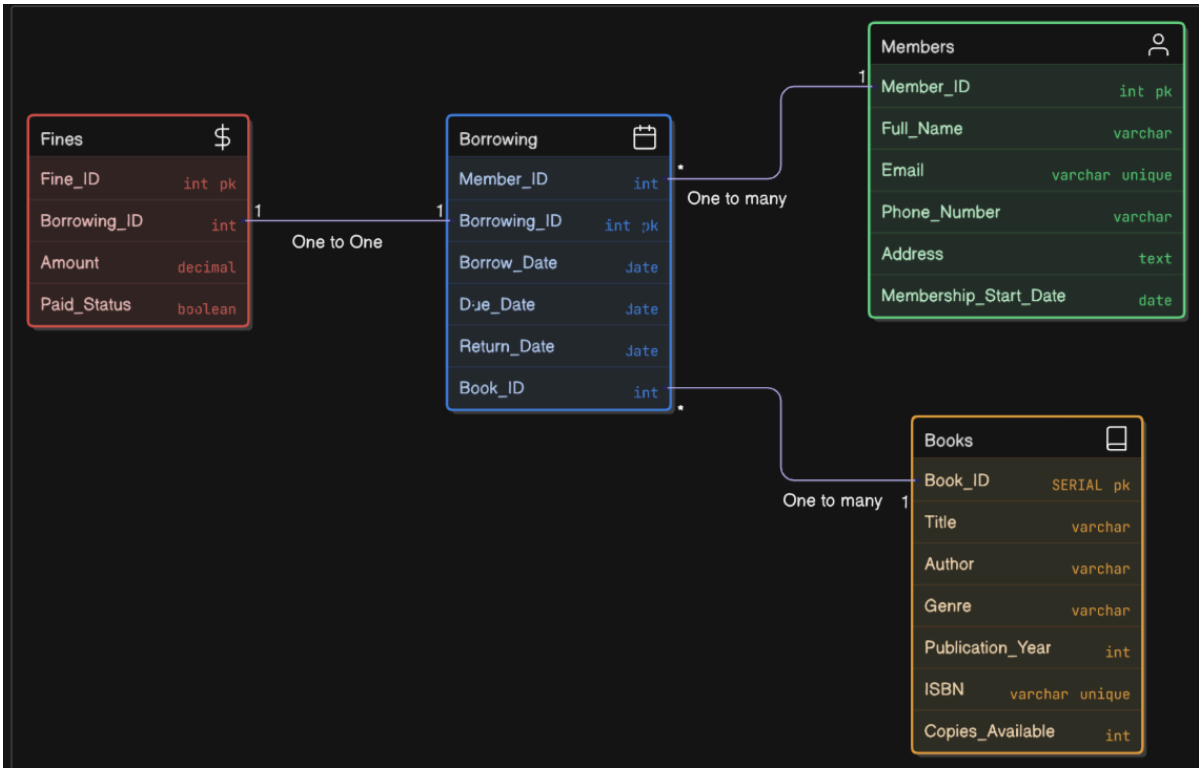
# ERD Diagram

```
Books {
  Book_ID int [pk]
  Title varchar(55)
  Author varchar(55)
  Genre varchar(50)
  Publication_Year int
  ISBN varchar(20) [unique]
  Copies_Available int [note: "Must be >= 0"]
}

Members {
  Member_ID int [pk]
  Full_Name varchar(255)
  Email varchar(255) [unique]
  Phone_Number varchar(15)
  Address text
  Membership_Start_Date date
}

Borrowing {
  Borrowing_ID int [pk]
  Member_ID int [ref: > Members.Member_ID]
  Book_ID int [ref: > Books.Book_ID]
  Borrow_Date date
  Due_Date date [note: "Must be after Borrow_Date"]
  Return_Date date [note: "NULL or >= Borrow_Date"]
}

Fines {
  Fine_ID int [pk]
  Borrowing_ID int [ref: > Borrowing.Borrowing_ID]
  Amount decimal(10,2) [note: "> 0"]
  Paid_Status boolean [note: "TRUE = Paid, FALSE = Unpaid"]
}
```


Members
Member_ID int pk
Full_Name varchar
Email varchar unique
Phone_Number varchar
Address text
Membership_Start_Date date
1
One to many
Fines
Fine_ID int pk
Borrowing_ID int
Amount decimal
Paid_Status boolean
1
One to One
1
Borrowing
Member_ID int
Borrowing_ID int pk
Borrow_Date date
Due_Date date
Return_Date date
Book_ID int
One to many
1
Books
Book_ID SERIAL pk
Title varchar
Author varchar
Genre varchar
Publication_Year int
ISBN varchar unique
Copies_Available int

# DDL SQL Code

```sql
CREATE DATABASE library_Management;

CREATE TABLE Books (
    Book_ID SERIAL PRIMARY KEY,
    Title VARCHAR(55) NOT NULL,
    Author VARCHAR(55) NOT NULL,
    Genre VARCHAR(50),
    Publication_Year INT,
    ISBN VARCHAR(20) UNIQUE NOT NULL,
    Copies_Available INT NOT NULL CHECK (Copies_Available >= 0)
);

CREATE TABLE Members (
    Member_ID SERIAL PRIMARY KEY,
    Full_Name VARCHAR(255) NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    Phone_Number VARCHAR(15) NOT NULL,
    Address TEXT,
    Membership_Start_Date DATE NOT NULL
   );

CREATE TABLE Borrowing (
    Borrowing_ID SERIAL PRIMARY KEY,
    Member_ID INTEGER REFERENCES Members(Member_ID),
    Book_ID INTEGER REFERENCES Books(Book_ID),
    Borrow_Date DATE NOT NULL,
    Due_Date DATE NOT NULL,
    Return_Date DATE,
    CHECK(Due_Date > Borrow_Date),
    CHECK(Return_Date IS NULL OR Return_Date >= Borrow_Date)
   );

CREATE TABLE Fines (
    Fine_ID SERIAL PRIMARY KEY,
    Borrowing_ID INTEGER REFERENCES Borrowing(Borrowing_ID),
    Amount DECIMAL(10, 2) NOT NULL CHECK (Amount > 0),
    Paid_Status BOOLEAN
   );
```

# Sample Data & DML Code

```sql
INSERT INTO Books (Title, Author, Genre, Publication_Year, ISBN,
Copies_Available)
VALUES ('1984', 'George Orwell', 'Dystopian', '1949', '9780451524935',
3);


INSERT INTO Members (Full_Name, Email, Phone_Number, Address,
Membership_Start_Date)
VALUES ('Alice Smith', 'alice@example.com', '1234567890', '123 Library
St.', '2025-07-01');


INSERT INTO Borrowing (Member_ID, Book_ID, Borrow_Date, Due_Date)
VALUES (1, 1, '2025-07-15', '2025-07-22');


INSERT INTO Fines (Borrowing_ID, Amount, Paid_Status)
VALUES (1, 5.00, FALSE);
```

# Use Case Document

```sql
SELECT *
FROM Borrowing
WHERE Borrowing_id = 1;

UPDATE Borrowing
SET Return_Date = CURRENT_DATE
WHERE Borrowing_id = 1;


SELECT Borrow_Date, Due_Date, CURRENT_DATE AS Return_Date
FROM Borrowing
WHERE Borrowing_id = 1;


SELECT GREATEST(0, CURRENT_DATE - Due_Date) AS Days_Overdue
FROM Borrowing
WHERE Borrowing_id = 1;

INSERT INTO Fines (Borrowing_id, Amount, Paid_Status)
VALUES (1, 12.00, FALSE);
```

# Library Management System — Process Summary Report

Library Management System — Process Summary Report

This project involved the design and implementation of a relational database for a library. The system manages books, library members, borrowing records, and fines for overdue returns. The design follows normalization principles to minimize redundancy and maintain data integrity.

The database includes four main tables:

Books: Stores details of each book, including title, author, genre, and available copies.

Members: Holds member registration data such as full name, email, and address.

Borrowing: Records when a member borrows a book, the due date, and the return date.

Fines: Tracks fines associated with late returns, including the amount and payment status.

The tables are linked through primary and foreign keys. Borrowings are tied to both members and books. If a return is late, a fine is created referencing the borrowing.

The SQL structure includes constraints like unique ISBNs and emails, CHECK conditions on date logic and amounts, and foreign keys for referential integrity.

To demonstrate functionality, sample data was added for one book, one member, one borrowing, and one fine.

In the example use case, a member borrows a book and returns it late. The system calculates the number of overdue days and inserts a fine. This showcases the integration between borrowing activity and the fine management process.

The system is now ready to be connected with a front-end interface for full library operations.