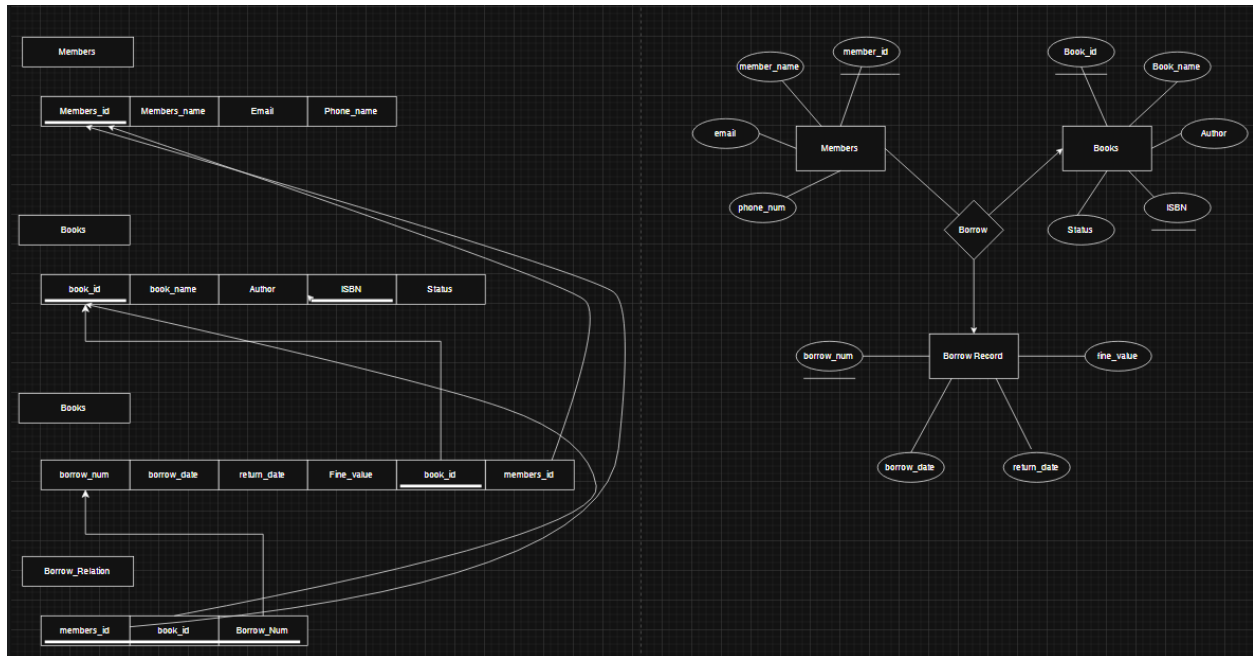


ER Diagram & Shema :



DDL Code :

```
CREATE DATABASE Library_Management_System;
```

```
GO
```

```
USE Library_Management_System;
```

```
GO
```

```
CREATE TABLE Members(
```

```
Member_ID INT PRIMARY KEY,
```

```
Member_Name NVARCHAR(20) NOT NULL,
```

```
Email NVARCHAR(50) NOT NULL UNIQUE,
```

```
Phone_Num NVARCHAR(15) NOT NULL,
```

```
);
```

```
CREATE TABLE Books(
```

```
Book_ID INT PRIMARY KEY,  
Book_name NVARCHAR(150) NOT NULL,  
Author NVARCHAR(50) NOT NULL,  
ISBN NVARCHAR(20) UNIQUE,  
Status NVARCHAR(20) DEFAULT 'Available'  
);
```

```
CREATE TABLE Borrow_Records(  
Borrow_Num INT PRIMARY KEY,  
Member_ID INT,  
Book_ID INT,  
Borrow_Date DATE NOT NULL,  
Return_Date DATE NOT NULL,  
Fine_Value DECIMAL(5,2) DEFAULT 0,  
FOREIGN KEY (Member_ID) REFERENCES Members(Member_ID),  
FOREIGN KEY (Book_ID) REFERENCES Books(Book_ID)  
);
```

```
INSERT INTO Members (Member_ID, Member_Name, Email, Phone_Num)  
VALUES ('1', 'Ziad', 'Ziad@gmail.com', '011'),  
      ('2', 'Zoz', 'zoz@gmail.com', '012');
```

```
INSERT INTO Books (Book_ID, Book_name, Author, ISBN)  
VALUES ('1', 'python', 'Devolper', '1234567890'),  
      ('2', 'c++', 'Dev2', '02134567'),
```

```
('3', 'Data Base', 'Dev5', '0218437');
```

```
GO
```

```
CREATE PROCEDURE Borrow_Book
```

```
    @Member_ID INT,
```

```
    @Book_ID INT
```

```
AS
```

```
BEGIN
```

```
    IF EXISTS (SELECT 1 FROM Books WHERE Book_ID = @Book_ID AND Status = 'Available')
```

```
    BEGIN
```

```
        INSERT INTO Borrow_Records (Member_ID, Book_ID, Borrow_Date)
```

```
        VALUES (@Member_ID, @Book_ID, GETDATE());
```

```
        UPDATE Books
```

```
        SET Status = 'Borrowed'
```

```
        WHERE Book_ID = @Book_ID;
```

```
        PRINT 'Book Borrowed';
```

```
    END
```

```
ELSE
```

```
BEGIN
```

```
    PRINT 'Book Is Not Available';
```

```
END
```

```
END;
```

GO

CREATE PROCEDURE Return_Book

 @Borrow_ID INT

AS

BEGIN

 DECLARE @Book_ID INT;

 DECLARE @Due_Date DATE;

 DECLARE @Fine DECIMAL(5,2);

 SELECT @Book_ID = Book_ID, @Due_Date = Borrow_Date FROM Borrow_Records
 WHERE @Borrow_ID = @Borrow_ID;

 SET @Fine = CASE

 WHEN DATEDIFF(DAY, @Due_Date, GETDATE()) > 14

 THEN (DATEDIFF(DAY, @Due_Date, GETDATE()) - 14) * 2

 ELSE 0

 END;

 UPDATE Borrow_Records

 SET Return_Date = GETDATE(), Fine_Value = @Fine

 WHERE @Borrow_ID = @Borrow_ID;

 UPDATE Books

 SET Status = 'Available'

 WHERE Book_ID = @Book_ID;

```
PRINT 'Book Returned' + CAST(@Fine AS NVARCHAR);  
END;
```

GUI :

Using tkinter

```
import pyodbc  
from tkinter import *  
from tkinter import messagebox, ttk  
  
try:  
    conn = pyodbc.connect(  
        'DRIVER=ODBC Driver 17 for SQL Server;' #This Driver Is For  
MicrosoftSQL Change It If You Use Another DBMS  
        'SERVER=DESKTOP-PMHMTSM;' #Change it According To Your Device Name  
        'DATABASE=Library_Management_System;'  
        'Trusted_Connection=yes;'  
    )  
  
    cursor = conn.cursor()  
except Exception as e:  
    messagebox.showerror("Database Connection Error", str(e))  
  
window = Tk()  
window.title("📖 Library Management System")  
window.geometry("800x600")  
  
def view_members():  
    try:  
        cursor.execute("SELECT * FROM Members")  
        records = cursor.fetchall()  
        display_records("Members", records)  
    except Exception as e:  
        messagebox.showerror("Error", str(e))  
  
def view_books():  
    try:
```

```

        cursor.execute("SELECT * FROM Books")
        records = cursor.fetchall()
        display_records("Books", records)
    except Exception as e:
        messagebox.showerror("Error", str(e))

def insert_member():
    member_id = member_id_entry.get()
    member_name = member_name_entry.get()
    email = email_entry.get()
    phone_num = phone_num_entry.get()
    try:
        cursor.execute(
            "INSERT INTO Members (Member_ID, Member_Name, Email, Phone_Num)
VALUES (?, ?, ?, ?)",
            (member_id, member_name, email, phone_num)
        )
        conn.commit()
        messagebox.showinfo("Success", "Member added successfully!")
    except Exception as e:
        messagebox.showerror("Error", str(e))

def insert_book():
    book_id = book_id_entry.get()
    book_name = book_name_entry.get()
    author = author_entry.get()
    isbn = isbn_entry.get()
    try:
        cursor.execute(
            "INSERT INTO Books (Book_ID, Book_name, Author, ISBN) VALUES
(?, ?, ?, ?)",
            (book_id, book_name, author, isbn)
        )
        conn.commit()
        messagebox.showinfo("Success", "Book added successfully!")
    except Exception as e:
        messagebox.showerror("Error", str(e))

def display_records(title, records):
    display_window = Toplevel(window)
    display_window.title(title)
    tree = ttk.Treeview(display_window, columns=(1, 2, 3, 4, 5), show="headings",
height=20)

```

```

tree.pack()
for i, column in enumerate(records[0] if records else range(5)):
    tree.heading(i+1, text=f"Column {i+1}")
for row in records:
    tree.insert('', 'end', values=row)

Label(window, text="Library Management System", font=("Arial", 24)).pack(pady=20)

frame = Frame(window)
frame.pack(pady=20)

Button(frame, text="View Members", command=view_members, width=20).grid(row=0,
column=0, padx=10)
Button(frame, text="View Books", command=view_books, width=20).grid(row=0,
column=1, padx=10)

Label(frame, text="Member ID").grid(row=1, column=0, pady=10)
member_id_entry = Entry(frame)
member_id_entry.grid(row=1, column=1, pady=10)

Label(frame, text="Member Name").grid(row=2, column=0, pady=10)
member_name_entry = Entry(frame)
member_name_entry.grid(row=2, column=1, pady=10)

Label(frame, text="Email").grid(row=3, column=0, pady=10)
email_entry = Entry(frame)
email_entry.grid(row=3, column=1, pady=10)

Label(frame, text="Phone Number").grid(row=4, column=0, pady=10)
phone_num_entry = Entry(frame)
phone_num_entry.grid(row=4, column=1, pady=10)

Button(frame, text="Insert Member", command=insert_member, width=20).grid(row=5,
column=0, pady=10)

Label(frame, text="Book ID").grid(row=6, column=0, pady=10)
book_id_entry = Entry(frame)
book_id_entry.grid(row=6, column=1, pady=10)

Label(frame, text="Book Name").grid(row=7, column=0, pady=10)

```

```
book_name_entry = Entry(frame)
book_name_entry.grid(row=7, column=1, pady=10)

Label(frame, text="Author").grid(row=8, column=0, pady=10)
author_entry = Entry(frame)
author_entry.grid(row=8, column=1, pady=10)

Label(frame, text="ISBN").grid(row=9, column=0, pady=10)
isbn_entry = Entry(frame)
isbn_entry.grid(row=9, column=1, pady=10)

Button(frame, text="Insert Book", command=insert_book, width=20).grid(row=10,
column=0, pady=10)

window.mainloop()

conn.close()
```