**A Project on**

# LIBRARY MANAGEMENT SYSTEM

**TABLE OF CONTENTS:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE NO.** |
| 01. | Problem Statement | 3 |
| 02. | Entities and attributes | 3 |
| 03. | Functional Dependencies | 3 |
| 04 | Novelty | 4 |
| 05. | Relationships identified | 16 |
| 06. | Relations Diagram | 17 |
| 07. | Entities Relationship  Diagram | 18 |
| 08. | E-R to Relational Diagram | 19 |
| 09. | Schema Diagram | 18 |
| 10 | Normalisation | 17 |
| 11 | DBMS concepts used | 28 |
| 12 | Sample Demo | 35 |

**PROBLEM STATEMENT:**

The Library Management System optimizes resource handling by efficiently cataloging, tracking, and managing books and other materials. It streamlines processes like borrowing, returning, and reservation through a user-friendly interface, enhancing user accessibility. The project's objective is to develop an efficient backend system using Oracle SQL for the library management application.

**INITIAL ASSUMPTIONS:**

1. We have assumed that we have 7 relations in total for the given problem.
2. Books relation: This relation describes the attributes of each books in the library
3. Books\_orders relation: This relation contains the details about the orders placed for the books needed by the library.

Here we have assumed that one order can be placed from a single supplier only but may contain many books.

1. Member\_login relation: It contains the authentication details of the members
2. Members relation: It consists of the details about the members
3. Issues relation: This relation consists of the details about the books borrowed by the members
4. Renewals relation: This relation consists of the renewals taking place with respect to an IssueID
5. Reservation relation: This relation consists of the requests made by the users for the books that are not currently available to lend

**NOVELTY**

**(i) Separate user and admin interfaces:**

This application has distinct user and admin interfaces to cater to the different needs of the user and the admin.

The user interface provides the user with the following functionalities:

* Search for a particular book
* View current borrowed books and reserved books.
* Borrow/Reserve a book
* Renew a borrowed book
* Update user profile

The admin interface provides the admin with the following functionalities:

* View all current issues/reservations
* View details about a particular user
* View details of all procurements
* View details about a particular supplier

**(ii) Single page to borrow, renew, reserve and return:**

A single page shows book information on a particular book along with consideration of the member id. This page is used to borrow, renew, reserve, and return a book. Using a single page reduces confusion and increases ease of use thereby improving user experience.

**(iii) Reservation system:**

We have introduced a reservation system that is not so prevalent in many library management apps. This feature enables the user to reserve a book for a particular date which he/she will borrow on the day.

**(iv) Easy renewal system:**

Our application has an easy renewal system in which the user can renew the book with a single click. The app allows the user to renew the book up to 10 times and each renewal pushes the due date by a week. This reduces the hassle of returning and reborrowing the same book again and again every week.

**ENTITIES AND ATTRIBUTES:**

* Books\_registry (ISBN, Title, Author\_ID, Authorname, Genre,

Publisher, Year, Copies)

* Book\_Orders(Procurement\_ID,Supplier\_ID,ISBN,Suppliername,Supp lier\_mob,Order\_date,Recived\_date,Qty)
* Members(Mem\_ID, Name, Role, Dept, Email,Mem\_mobno)
* Member\_login(Mem\_id,Pwd)
* Issues (Issue\_ID, Mem\_ID, ISBN, Duedate, Issuedate, ReturnStatus, FineStatus)
* Renewals(RenewaID,Issue\_ID,NewDueDate,RenewedDate)
* Reservation( Reserve\_ID, Mem\_ID, ISBN, Reservedate,

ReservationStatus)

**FUNCTIONAL DEPENDENCIES:**

### 01) Books\_registry (ISBN, Title, Author\_ID, Authorname, Genre, Publisher, Year, Copies)

Books relation contains information about the books available in the library and the number of copies.

F= { ISBN→ Title, Authorname, Genre, Publisher, Year, Copies, Author\_Id→Authorname}

### STEP 1: Check if the singleton property is satisfied

{ISBN} →{Title}

{ISBN}→{AuthorID}

{ISBN} →{Authorname}

{ISBN} →{Publisher}

{ISBN} →{Genre}

{ISBN} →{Copies}

{ISBN} →{Year}

Therefore F1={

ISBN →Title,

ISBN→Author\_ID

ISBN→Authorname,

ISBN →Publisher,

ISBN→Genre,

ISBN →Copies,

ISBN →Year,

Author\_Id→Authorname

}

**STEP 2: Check for the existence of extraneous attributes in an FD**

//No extraneous attribute

### STEP 3: Check for the existence of redundant FDs

There is a transitive dependency among the FDS as follows:

ISBN→Author\_ID,

Author\_Id→Authorname

Thus ISBN→Authorname is redundant.

Therefore THE MINIMAL SET OF DEPENDENCIES,

F3={

ISBN →Title,

ISBN→Author\_ID,

ISBN →Publisher,

ISBN→Genre,

ISBN →Copies,

ISBN →Year,

Author\_Id→Authorname}

**CLOSURE:**

{ISBN+}={ISBN,Title,Author\_ID,Authorname,Year,Copies,Genre,Publisher} ● It is a super key and also a minimal super key.

● Hence, it is the primary key.

### 02) Book\_Orders (Procurement\_ID,Supplier\_ID,ISBN,Suppliername, Supplier\_mob,Order\_date,Recived\_date,Qty)

F= { Supplier\_ID→Suppliername,Supplier\_mob;

Procurement\_ID→Supplier\_ID,ISBN,Order\_date,Recived\_date,Suppliername,Supplier mobno;

Procurement\_ID,ISBN→Qty}

### STEP 1: Check if the singleton property is satisfied

{Procurement\_ID} →{SupplierID}

{Procurement\_ID}→{ISBN}

{Procurement\_ID} →{Order\_date}

{Procurement\_ID} →{Recived\_date}

{Supplier\_ID} →{Suppliername}

{Supplier\_ID} →{ Suppliermobno}

Therefore F1={

Procurement\_ID →SupplierID,

Procurement\_ID→ISBN,

Procurement\_ID →Order\_date,

Procurement\_ID →Recived\_date,

Procurement\_ID →Suppliername,

Procurement\_ID →Suppliermobno,

Procurement\_ID,ISBN →Quantity,

Supplier\_ID →Suppliername,

Supplier\_ID→Suppliermobno}

### STEP 2: Check for the existence of extraneous attributes in an FD

**Procurement\_ID,ISBN →Quantity**

* Let's consider the FD by removing procurement\_ID,

ISBN→Quantity

Implication is not satisfied hence this is not an extraneous attribute.

* Now let us consider the fd by removing ISBN,

Procurement\_ID→Quantity

This also doesn't satisfy hence this is not an extraneous attribute.

### STEP 3: Check for the existence of redundant FDs

There is a transitive dependency among the FDS as follows:

* Procurement\_ID→SupplierID, Supplier\_ID→Suppliername,

Thus, Procurement\_ID→Suppliername is redundant.

* Procurement\_ID→SupplierID, Supplier\_ID→Suppliermobno

Thus, Procurement\_ID→Suppliermobno is redundant.

Therefore THE MINIMAL SET OF DEPENDENCIES,

F3={

Procurement\_ID →SupplierID,

Procurement\_ID→ISBN,

Procurement\_ID →Order\_date,

Procurement\_ID →Recived\_date,

Procurement\_ID,ISBN →Quantity,

Supplier\_ID →Suppliername, Supplier\_ID→Suppliermobno}

**CLOSURE:**

{Procurement\_ID+}={SupplierID, Suppliername, Suppliermobno, ISBN, Order\_date, Received\_date,Quantity}

* It is a super key and also a minimal super key.
* Hence, it is the primary key.

**03) Members(Mem\_ID, Name, Role, Dept, Email,Mem\_mobno)**

F= {Memid→Name, Role, Dept, Email, Mem\_mobno}

### STEP 1: Check if the singleton property is satisfied

{Mem\_ID} →{Name}

{Mem\_ID}→{Role}

{Mem\_ID} →{Dept}

{Mem\_ID} →{Email}

{Mem\_ID} →{Mem\_mobno}

Therefore F1={

Mem\_ID→Name

Mem\_ID→Role

Mem\_ID →Dept

Mem\_ID →Email

Mem\_ID→Mem\_mobno

}

### STEP 2: Check for the existence of extraneous attributes in an FD

//No extraneous attribute

**STEP 3: Check for the existence of redundant FDs** There are no redundant FDs.

**CLOSURE:**

{Memid+}=Name, Role, Dept, Email, Mem\_mobno

* It is a super key and also a minimal super key.
* Hence, it is the primary key.

### 04) Member\_login(Mem\_id,Pwd)

F= {Memid→Pwd}

**STEP 1: Singleton property satisfied**

### STEP 2: Check for the existence of extraneous attributes in an FD

//No extraneous attribute

**STEP 3: Check for the existence of redundant FDs** There are no redundant FDs.

**CLOSURE:**

{Memid+}= Memid→Pwd

* It is a super key and also a minimal super key.
* Hence, it is the primary key.

**05) Issues (Issue\_ID, Mem\_ID, ISBN, Duedate, Issuedate, ReturnStatus, FineStatus)**

F= Issue\_ID → {Mem\_ID, ISBN, Duedate, Issuedate, ReturnStatus, FineStatus}

### STEP 1: Check if the singleton property is satisfied

{Issue\_ID } →{Mem\_ID} {Issue\_ID } →{ISBN}

{Issue\_ID } →{Duedate}

{Issue\_ID } →{Issuedate}

{Issue\_ID } →{ReturnStatus}

{Issue\_ID } →{FineStatus}

**STEP 2: Check for the existence of extraneous attributes in an FD**

//No extraneous attribute

### STEP 3: Check for the existence of redundant FDs

F3={Issue\_ID → Mem\_ID,

Issue\_ID → ISBN,

Issue\_ID → Duedate,

Issue\_ID → Issuedate,

Issue\_ID → ReturnStatus,

Issue\_ID → FineStatus}

* {Issue\_ID+}={Issue\_ID,ISBN, Duedate,Issuedate, ReturnStatus, FineStatus}
* {Issue\_ID+}={Issue\_ID,Mem\_ID,Duedate,Issuedate,ReturnStatus,FineStatus}
* {Issue\_ID+}={Issue\_ID,Mem\_ID,ISBN,Issuedate,ReturnStatus,FineStatus}
* {Issue\_ID+}={Issue\_ID,Mem\_ID,ISBN,Duedate,ReturnStatus,FineStatus} ● {Issue\_ID+}={Issue\_ID,Mem\_ID,ISBN,Duedate,Issuedate ,ReturnStatus}

//No redundant

Minimal set of dependencies=

{Issue\_ID → Mem\_ID,

Issue\_ID → ISBN,

Issue\_ID → Duedate,

Issue\_ID → Issuedate,

Issue\_ID → ReturnStatus,

Issue\_ID → FineStatus}

**CLOSURE:**

{Issue\_ID+}→ {Issue\_ID,Mem\_ID,ISBN,Duedate,Issuedate,ReturnStatus,FineStatus} ● It is a super key and also a minimal super key.

● Hence, it is the primary key.

**06) Renewals(RenewaID,Issue\_ID,NewDueDate,RenewedDate)**

F= {Renewal\_ID→ Issue\_ID, NewDueDate, RenewedDate}

### STEP 1: Check if the singleton property is satisfied

{Renewal\_ID} →{Issue\_ID}

{Renewal\_ID} →{NewDueDate}

{Renewal\_ID} →{RenewedDate}

**STEP 2: Check for the existence of extraneous attributes in an FD**

//No extraneous attribute

### STEP 3: Check for the existence of redundant FDs

F3={Renewal\_ID → Issue\_ID,Renewal\_ID → NewDueDate,Renewal\_ID →

RenewedDate}

{Renewal\_ID+}={Renewal\_ID,NewDueDate,RenewedDate}

{Renewal\_ID+}={Renewal\_ID,Issue\_ID,RenewedDate}

{Renewal\_ID+}={Renewal\_ID,Issue\_ID,NewDueDate}

//No redundant

Minimal set of dependencies=

{Renewal\_ID → Issue\_ID,Renewal\_ID → NewDueDate,Renewal\_ID → RenewedDate}

**CLOSURE:**

{Renewal\_ID+}→ {IRenewal\_ID,Issue\_ID, NewDueDate, Reservedate}

* It is a super key and also a minimal super key. ● Hence, it is the primary key.
* Reservation( Reserve\_ID, Mem\_ID, ISBN, Reservedate,ReservationStatus)

**07) Reservation( Reserve\_ID, Mem\_ID, ISBN, Reservedate, ReservationStatus)**

F= {Reserve\_ID→ Mem\_ID, ISBN, Reservedate,ReservationStatus}

### STEP 1: Check if the singleton property is satisfied

{Reserve\_ID} →{Mem\_ID}

{Reserve\_ID} →{ISBN}

{Reserve\_ID} →{Reservedate}

{Reserve\_ID} →{ReservationStatus}

**STEP 2: Check for the existence of extraneous attributes in an FD**

//No extraneous attribute

### STEP 3: Check for the existence of redundant FDs

F3={Reserve\_ID →Mem\_ID,

Reserve\_ID →ISBN,

Reserve\_ID →Reservedate,

Reserve\_ID →ReservationStatus}

{Reserve\_ID+}={Reserve\_ID,ISBN,Reservedate,ReservationStatus}

{Reserve\_ID+}={Reserve\_ID,Mem\_ID,Reservedate,ReservationStatus}

{Reserve\_ID+}={Reserve\_ID,Mem\_ID,ISBN,ReservationStatus}

{Reserve\_ID+}={Reserve\_ID,Mem\_ID,ISBN,Reservedate}

//No redundant

Minimal set of dependencies=

{Reserve\_ID →Mem\_ID,

Reserve\_ID →ISBN,

Reserve\_ID →Reservedate, Reserve\_ID →ReservationStatus}

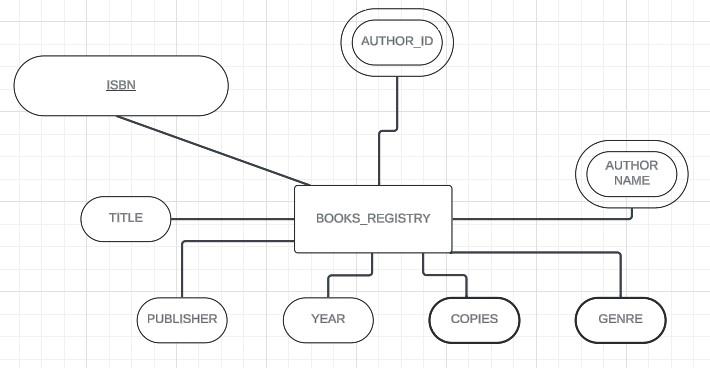
**CLOSURE:**

{Reserve\_ID+}→ {Reserve\_ID,Mem\_ID, ISBN, Reservedate,ReservationStatus} ● It is a super key and also a minimal super key.

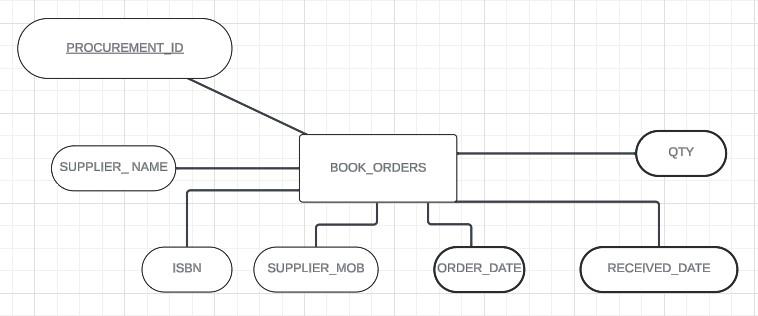
● Hence, it is the primary key.

**ENTITIES AND ATTRIBUTES DIAGRAMS:**

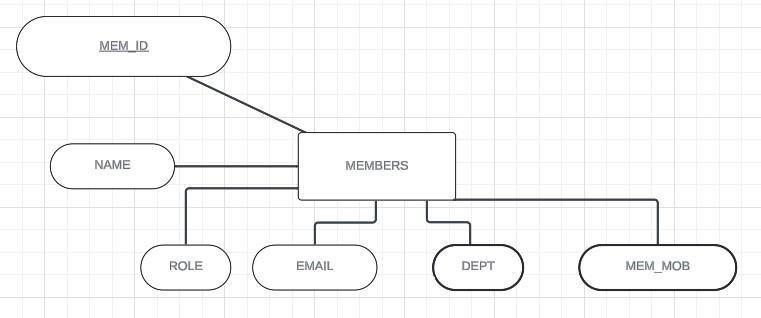
### 01)



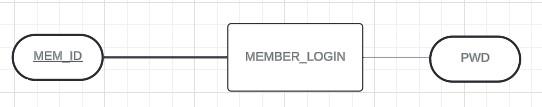
### 02)



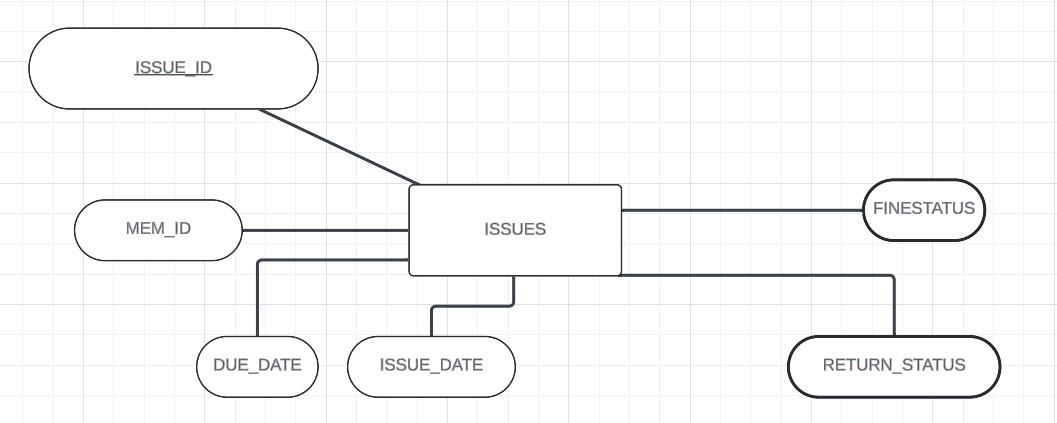
### 03)



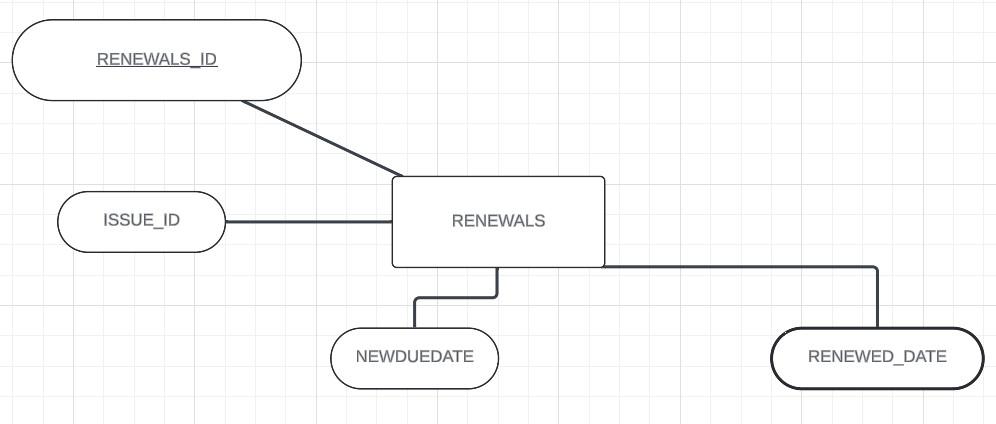
### 04)



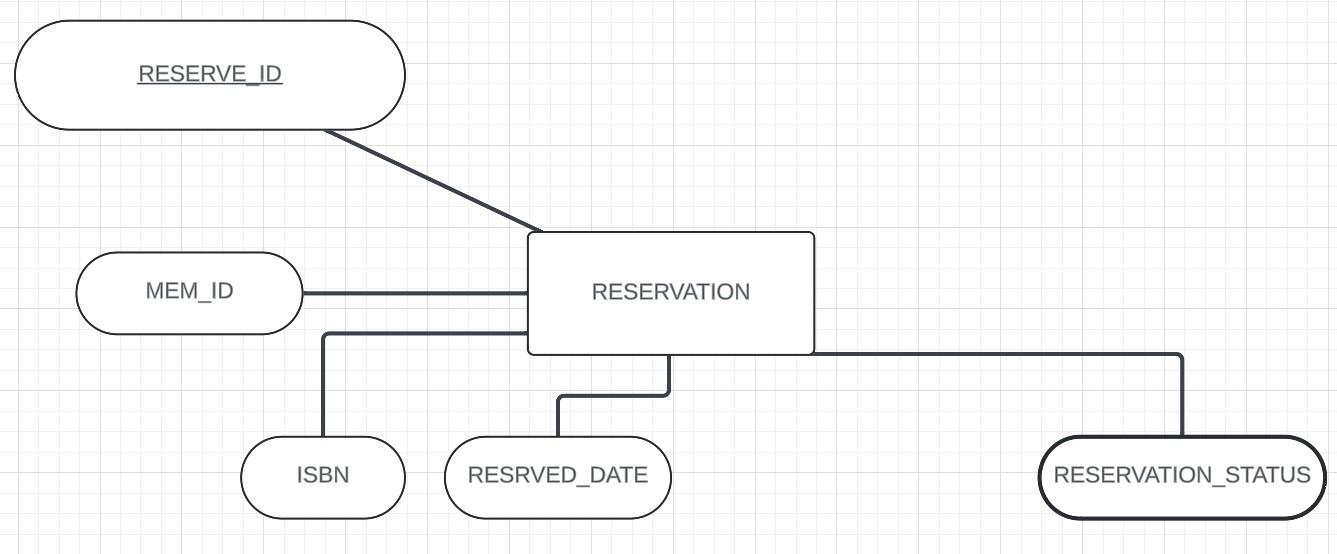
### 05)



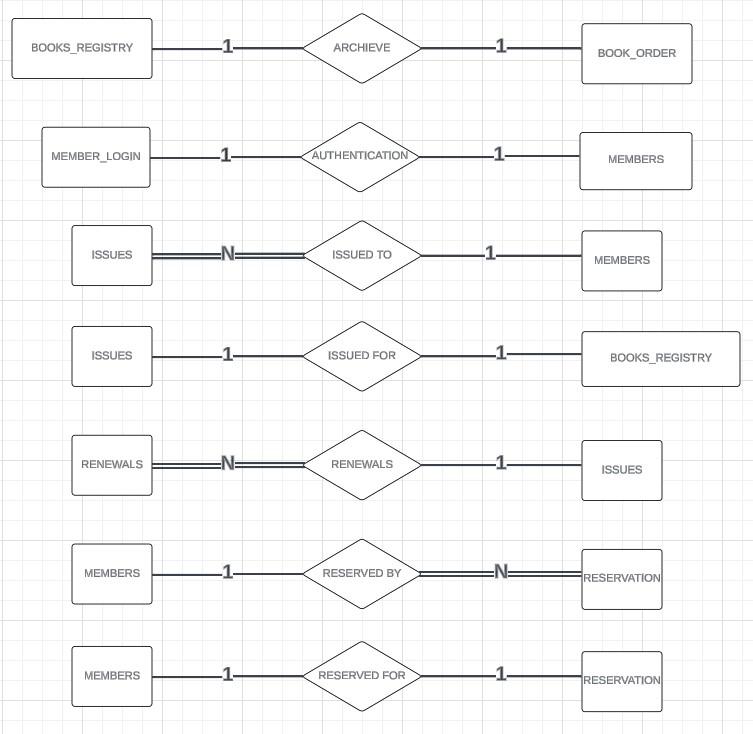
### 06)



### 07)

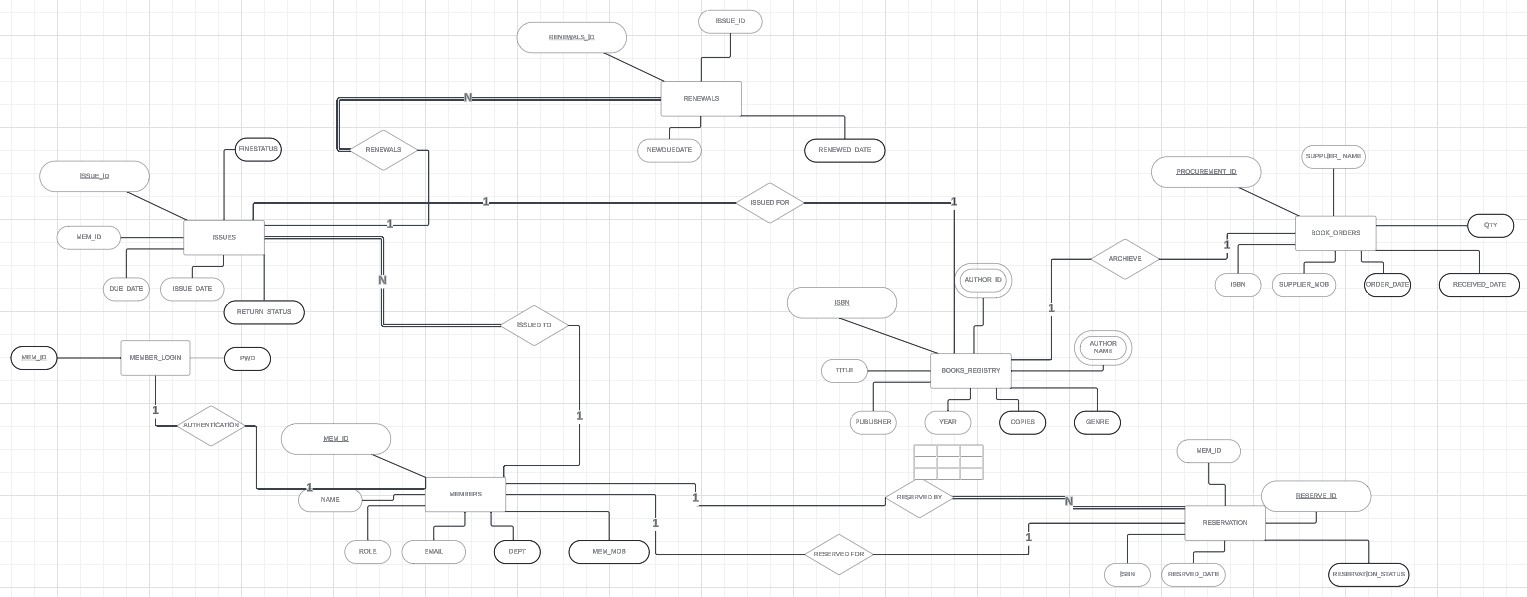


**RELATIONSHIPS:**



**E-R**

**DIAGRAM:**



**SCHEMA DIAGRAM:**

### NORMALIZATION

**1) Books\_registry**

**Canonical cover:**

**ISBN→Title**

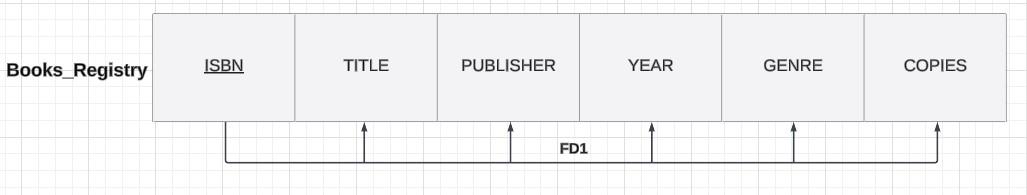
**ISBN→Publisher**

**ISBN→Year**

**ISBN→Copies**

**ISBN→Genre**

**Primary Key:** ISBN **FDS:**



**FD1:ISBN→Title, Publisher, Year, Copies, Genre**

**1NF: Already in 1NF since there are no multivalue attributes**

**2NF: Already in 2NF since there are no partial functional dependencies**

**3NF: Already in 3NF since there are no transitive dependencies**

**2)Entity:Member\_login Canonical cover: Mem\_ID→Pwd**

**Primary Key: Mem\_ID FDS:**

**FD1:Mem\_ID→Pwd**

**1NF: Already in 1NF since there is no multivalue attributes**

**2NF: Already in 2NF since there is no partial functional dependencies**

**3NF: Already in 3NF since there is no transitive dependencies**

**3)Entity:Members Canonical cover:**

**Mem\_ID→Name**

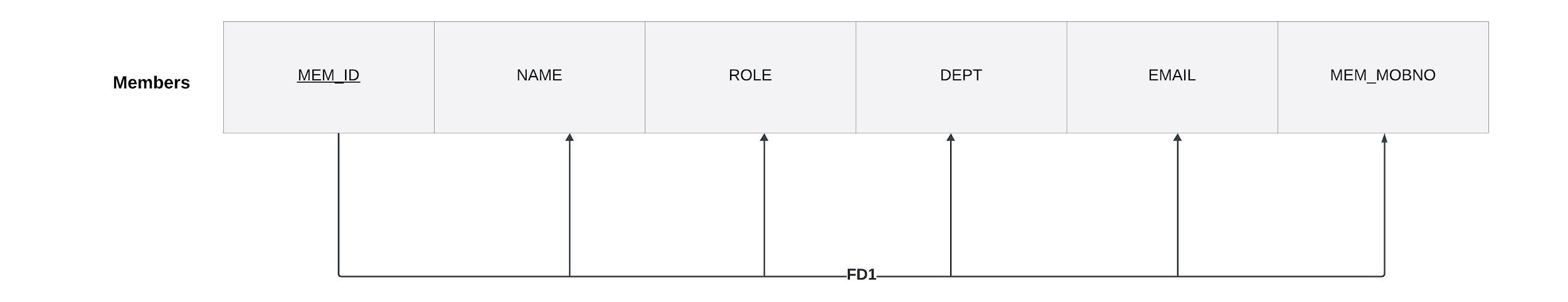
**Mem\_ID→Role**

**Mem\_ID→Dept**

**Mem\_ID→Email**

**Mem\_ID→Mem\_mobno**

**Primary Key: Mem\_ID FDS:**



**FD1:Mem\_ID→Name,Role,Dept,Email,Mem\_mobno**

**1NF: Already in 1NF since there is no multivalue attributes**

**2NF: Already in 2NF since there is no partial functional dependencies**

**3NF: Already in 3NF since there is no transitive dependencies**

**4)Entity:Issues Canonical cover:**

**Issue\_ID→Mem\_ID**

**Issue\_ID→ISBN**

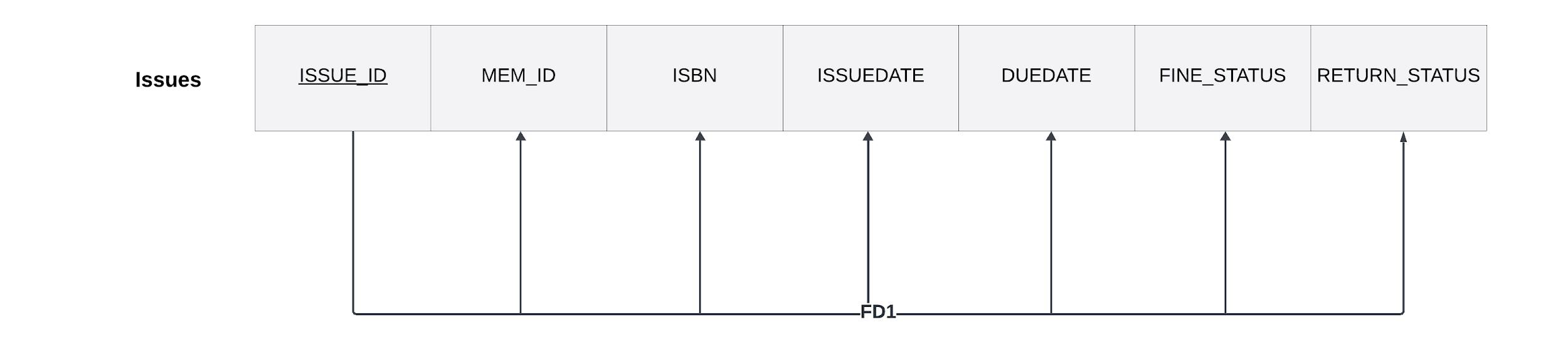
**Issue\_ID→Issuedate**

**Issue\_ID→Duedate**

**Issue\_ID→Returnstatus**

**Issue\_ID→Duestatus**

**Primary Key: Issue\_ID FDS:**



**FD1:Issue\_ID→Mem\_ID,ISBN,Issuedate,Duedate,Returnstatus,Duestatus**

**1NF: Already in 1NF since there is no multivalue attributes**

**2NF: Already in 2NF since there is no partial functional dependencies**

**3NF: Already in 3NF since there is no transitive dependencies**

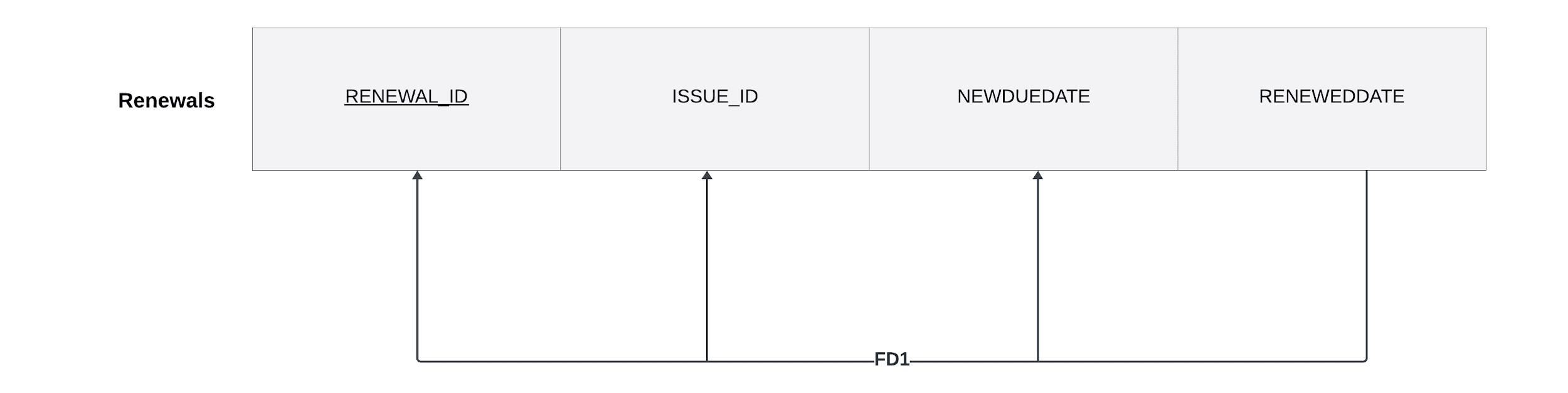
**5)Entity:Renewals Canonical cover:**

**Renewal\_ID→Issue\_ID**

**Renewal\_ID→Newduedate**

**Renewal\_ID→Reneweddate**

**Primary Key: Renewal\_ID FDS:**



**FD1:Renewal\_ID→Issue\_ID,Newduedate,Reneweddate**

**1NF: Already in 1NF since there is no multivalue attributes**

**2NF: Already in 2NF since there is no partial functional dependencies**

**3NF: Already in 3NF since there is no transitive dependencies**

**6)Entity:Reservations Canonical cover:**

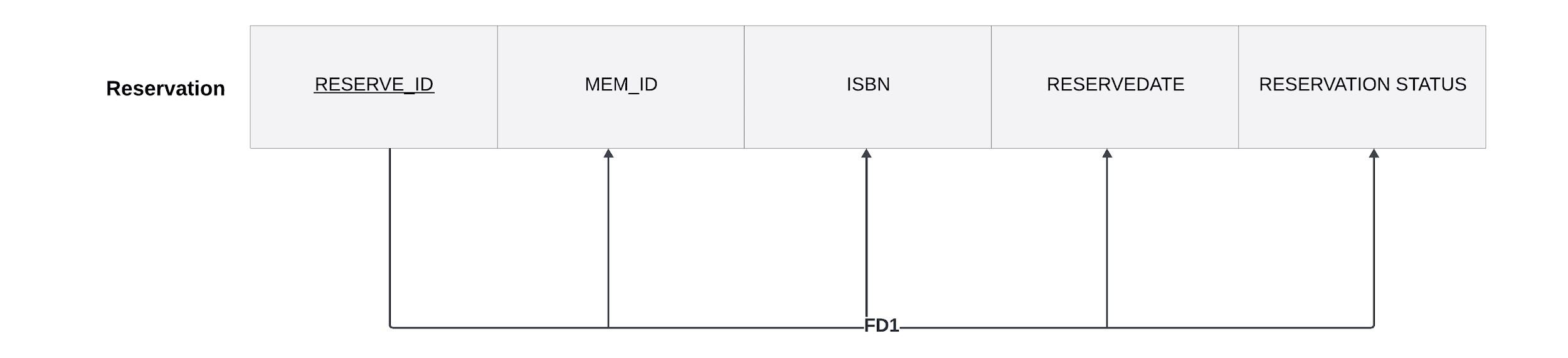
**Reservation\_ID→Mem\_ID**

**Reservation\_ID→ISBN**

**Reservation\_ID→Reservedate**

**Reservation\_ID→Reservationstatus**

**Primary Key:Reservation\_ID FDS:**



**FD1:Reservation\_ID→Mem\_ID,ISBN,Reserveddate,Reservationstatus**

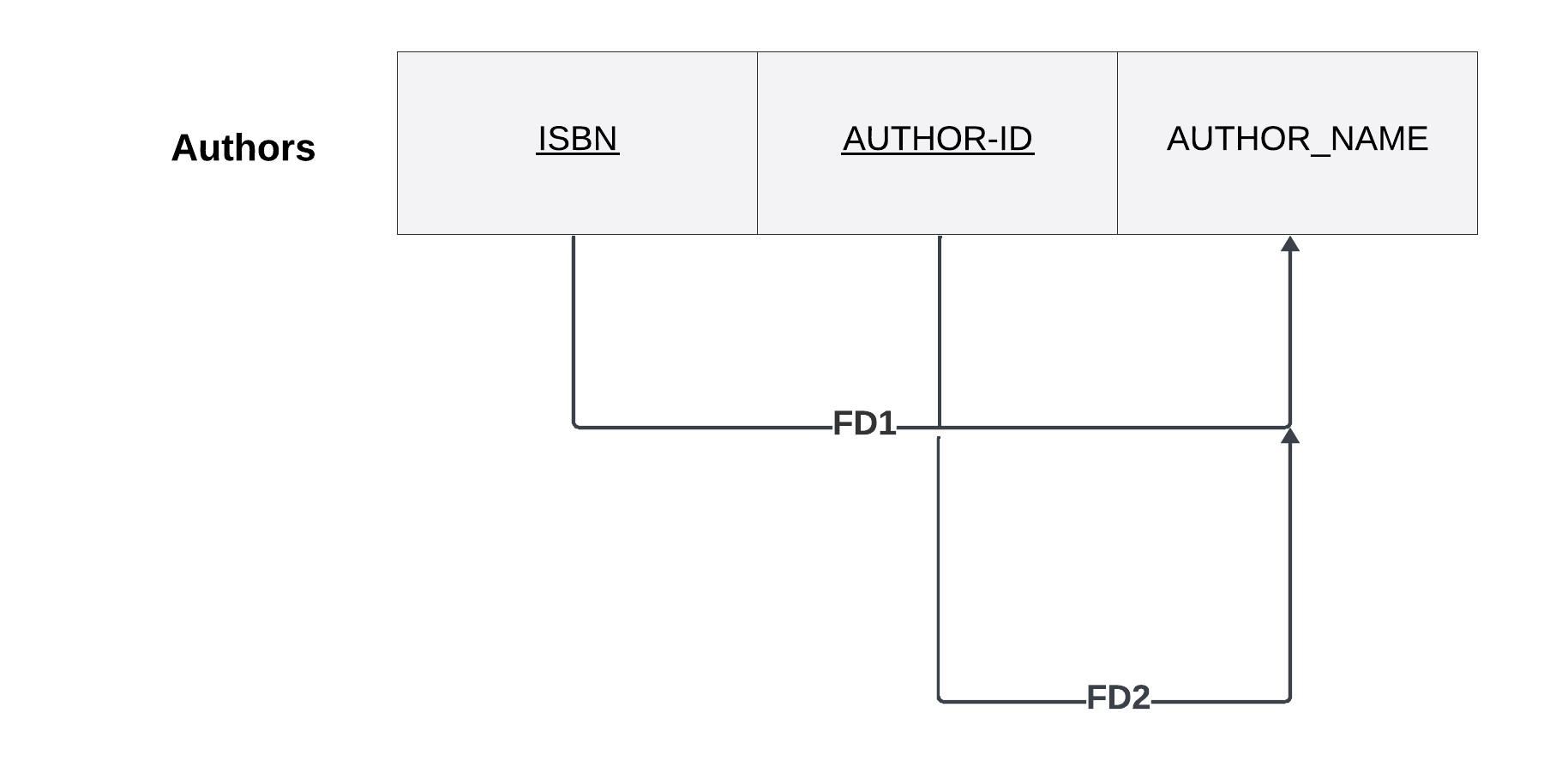
**1NF: Already in 1NF since there is no multivalue attributes**

**2NF: Already in 2NF since there is no partial functional dependencies**

**3NF: Already in 3NF since there is no transitive dependencies 7)Entity:Authors Canonical cover:**

**Auth\_ID→Authname**

**Primary Key: Issue\_ID,Auth\_ID FDS:**



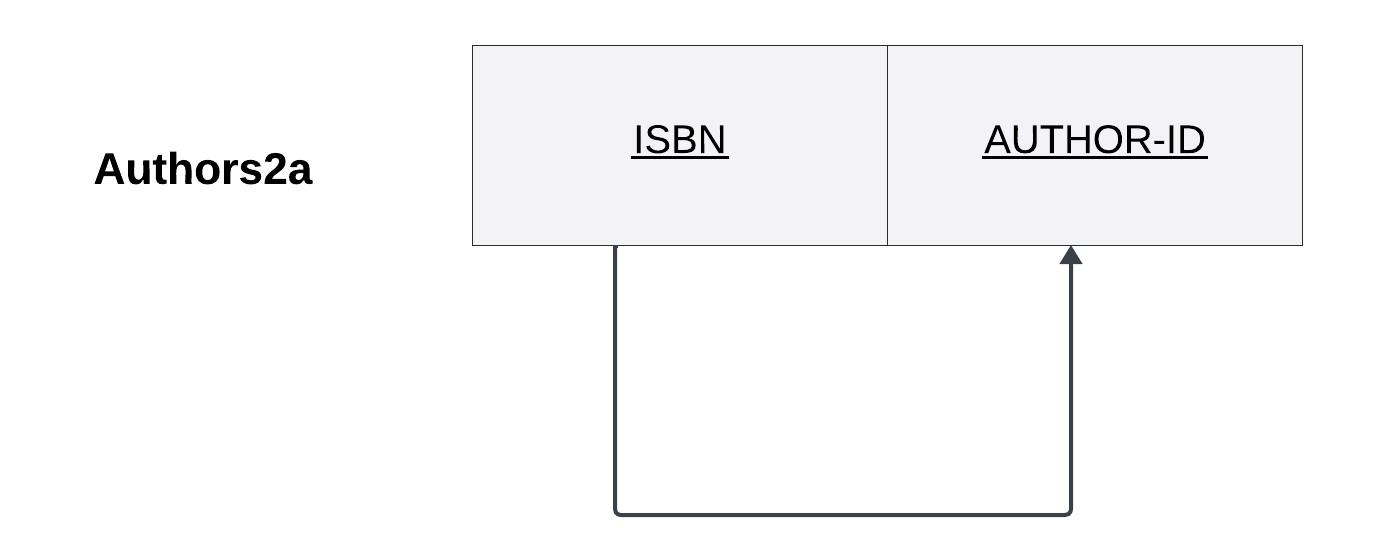
**FD1:Auth\_ID→Authname**

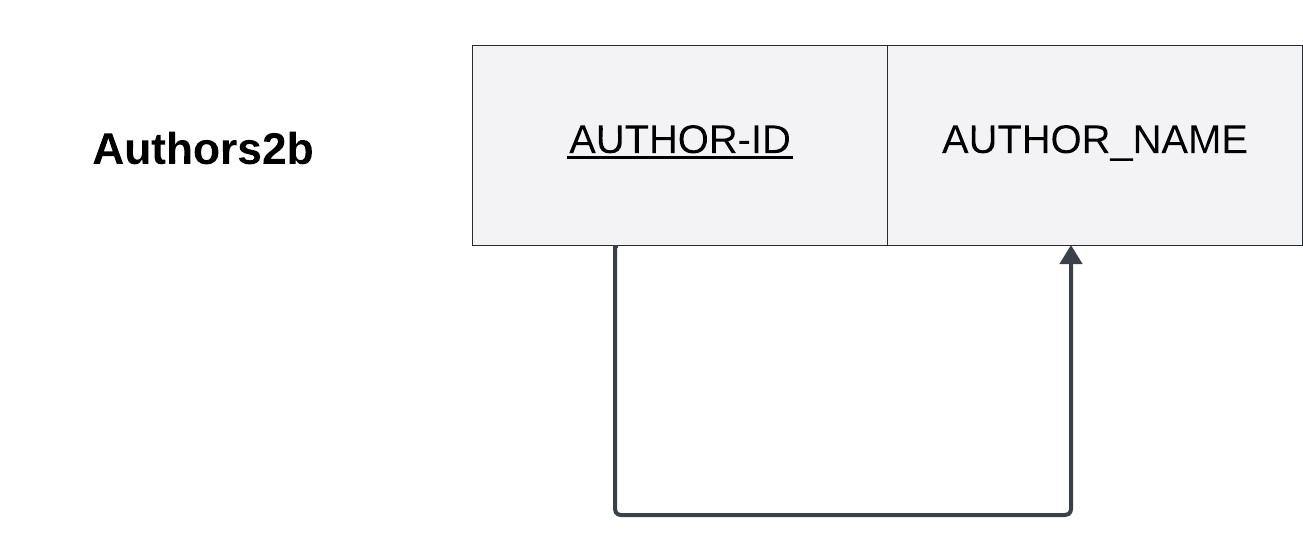
**1NF: Already in 1NF since there is no multivalue attributes**

**2NF: It is violating 2NF since there is a partial functional dependency**

**Auth\_ID→Authname**

**Thus we have to split the table into twos Authors2a and Authors2b as follows,**





**3NF: Already in 3NF since there is no transitive dependencies**

**8)Entity:Book\_orders Canonical cover:**

**Procurement\_ID→Supplier\_ID**

**Procurement\_ID→ISBN**

**Procurement\_ID→Order\_date**

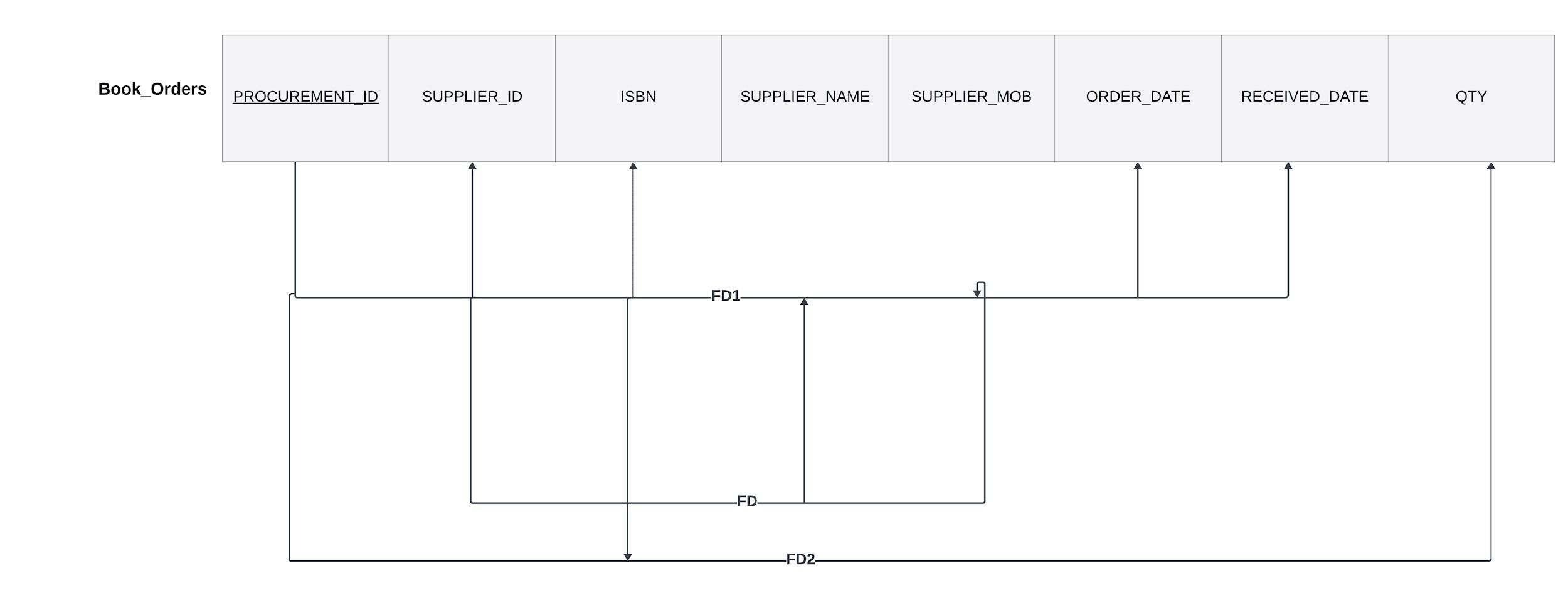
**Procurement\_ID→Recived\_date**

**Procurement\_ID,ISBN→Quantity**

**Supplier\_ID→Suppliername**

**Supplier\_ID→Suppliermobno**

**Primary Key: Procurement\_ID FDS:**

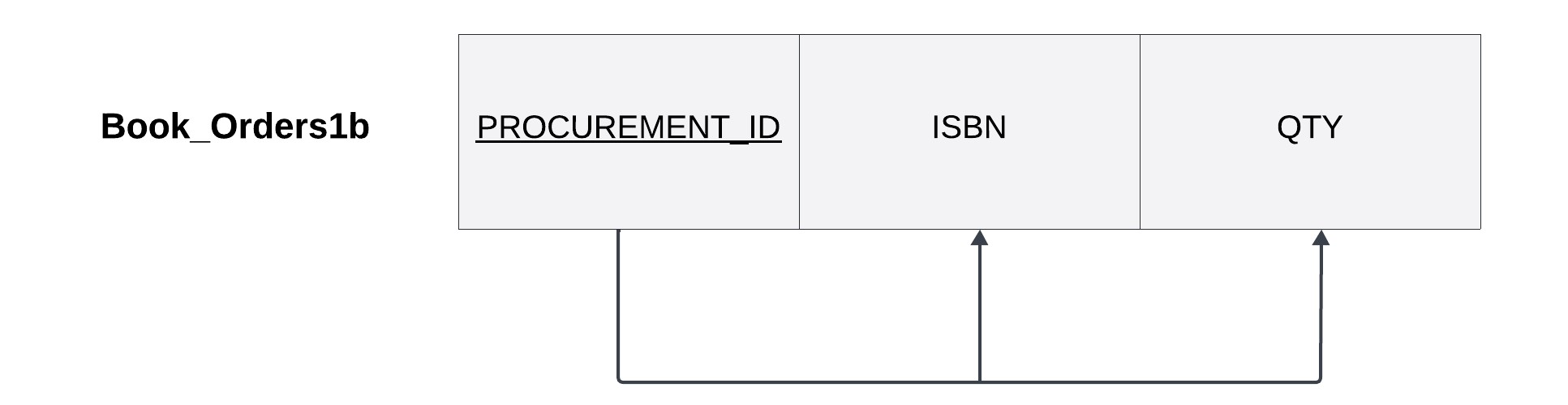
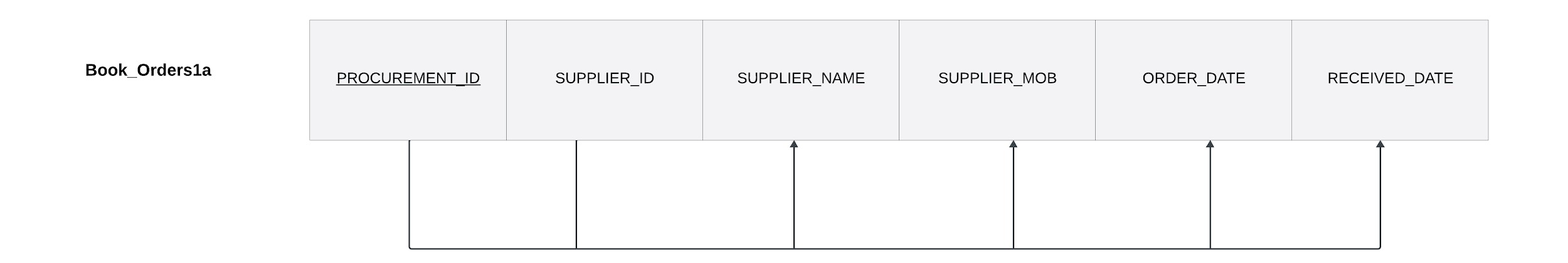


**FD1:Procurement\_ID→Supplier\_ID,ISBN,Order\_date,Recived\_date**

**FD2:Procurement\_ID,ISBN→Quantity**

**FD3:Supplier\_ID→Suppliername,Suppliermobno**

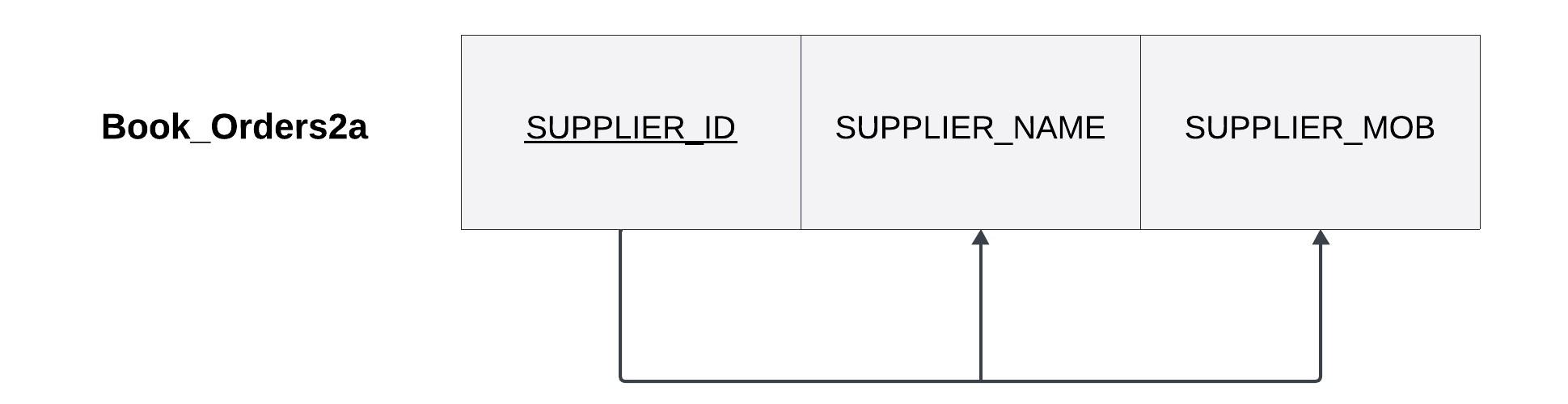
**1NF:ISBN is a multivalued attribute hence there is a 1NF violation but it is also a partial key so we can divide it as follows into Book\_orders1a, Book\_orders1b**

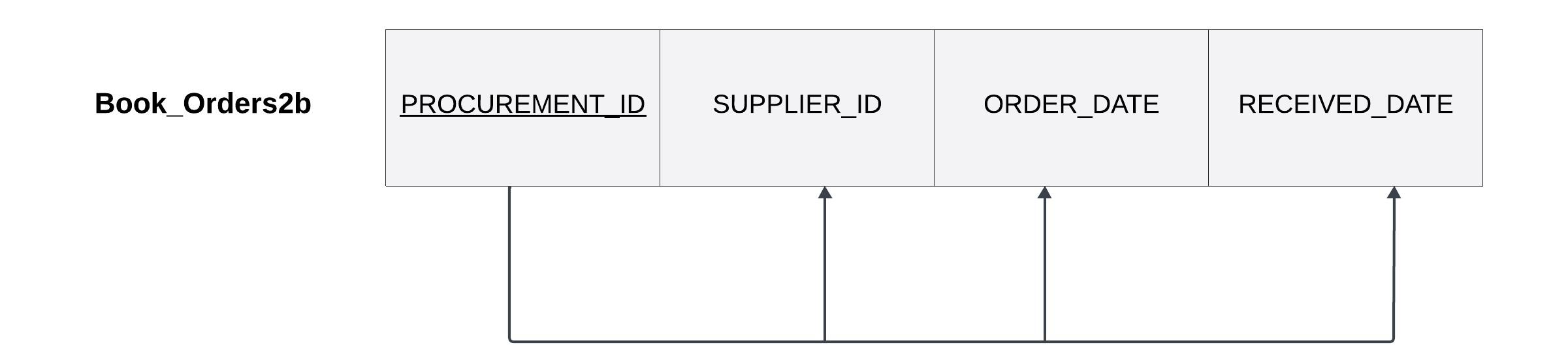


**2NF: Both the tables are in 2NF since there is no partial functional dependencies**

**3NF: Book\_orders1a is in 3NF violation since there is a transitive dependency with respect to the Supplier\_ID so we divide it into two tables as follows Book\_orders2a and**

**Book\_orders2b**





DBMS CONCEPTS USED

**1) TRIGGERS:**

**(i)Trigger\_Insert\_Reservation:**

CREATE OR REPLACE TRIGGER Trigger\_Insert\_Reservation

BEFORE INSERT ON Reservation

FOR EACH ROW

BEGIN

    IF :NEW.Reservation\_ID IS NOT NULL THEN

        :NEW.reserve\_status := 'PENDING';

    END IF;

END;

/

**Functionality:** Whenever a new reserve request is raised, the reserve status will default to ‘Pending’ and the reserve\_status will be changed to ‘Issued’ later. This trigger **Trigger\_Insert\_Reservation** sets **reserve\_status to pending for every new insertion** in the Reservation table.

**(ii) Trigger\_Insert\_Renewal:**

CREATE OR REPLACE TRIGGER Trigger\_Insert\_Renewal

BEFORE INSERT ON Renewals

FOR EACH ROW

BEGIN

    IF :NEW.Renew\_ID IS NOT NULL THEN

        :NEW.Newdue\_date := :NEW.Renewed\_date + 7;

    END IF;

END;

/

**Functionality:** Every time a user renews his /her borrowed book (that is reflected as a new entry in the renewals table), this trigger **Trigger\_Insert\_Renewal** updates the newdue\_date to Renewed\_date+7 (i.e., **upon renewal add a week to the due date**)

**(iii)Trigger\_Insert\_issues\_due\_date:**

rem: 'trigger insert before issue due\_date= issue\_date+7'

CREATE OR REPLACE TRIGGER Trigger\_Insert\_issues\_due\_date

BEFORE INSERT ON Issues

FOR EACH ROW

BEGIN

IF :NEW.Issue\_ID IS NOT NULL THEN

:NEW.Due\_Date := :NEW.Issue\_Date + 7;

:NEW.due\_status := 'No due';

:NEW.return\_status := 'Not Returned';

END IF;

END;

/

**Functionality:** Whenever a new user borrows a book, a new issue is made. This trigger **Trigger\_Insert\_issues\_due\_date** sets

* Due\_Date as 7 days more than the issued date (which will be the system date-SYSDATE at the time of issue creation, SYSDATE is used in the Netbeans frontend)
* Due\_status to ‘**No due**’ (default status)
* Return\_status to ‘**Not Returned**’

**2)FUNCTIONS:**

**(i) Book\_Stock:**

**CREATE OR REPLACE FUNCTION BOOK\_STOCK (**

**ISBN1 IN NUMBER**

**)**

**RETURN NUMBER**

**IS**

**issue\_count NUMBER;**

**book\_copies NUMBER;**

**BEGIN**

**-- Get the count of issued books**

**SELECT COUNT(Issue\_ID) INTO issue\_count**

**FROM ISSUES**

**WHERE ISBN = ISBN1;**

**-- Get the total number of book copies**

**SELECT COPIES INTO book\_copies**

**FROM Book\_Registry**

**WHERE ISBN = ISBN1;**

**-- Return the number of available copies**

**RETURN book\_copies - issue\_count;**

**END;**

**/**

**Functionality:** The function **Book\_Stock** is used to **return the number of copies in stock** of a particular book (identified by unique ISBN). This returns integer output by subtracting the number of issues done to a particular ISBN (from the Issues table) from the ‘copies’ value (total number of books of the given ISBN in the library’s possession) from the Book\_Registry table for the corresponding ISBN.

**3) PROCEDURES/FUNCTIONS:**

**(i)Check\_login**

CREATE OR REPLACE PROCEDURE check\_login(

    p\_mem\_id IN Mem\_login.mem\_id%type,

    p\_pwd IN Mem\_login.mem\_id%type,

    p\_result OUT VARCHAR2

) AS

    v\_count NUMBER;

BEGIN

    SELECT COUNT(\*)

    INTO v\_count

    FROM Mem\_login

    WHERE mem\_id = p\_mem\_id;

    IF v\_count = 0 THEN

        p\_result := 'not\_found';

    ELSE

        SELECT COUNT(\*)

        INTO v\_count

        FROM Mem\_login

        WHERE mem\_id = p\_mem\_id AND pwd = p\_pwd;

        IF v\_count = 0 THEN

            p\_result := 'wrong\_password';

        ELSE

            p\_result := 'login\_successful';

        END IF;

    END IF;

END;/

**Functionality:** This procedure checks the validity of the user trying to login to the system. **It verifies that the corresponding username and password** matched in the backend relation ‘Mem\_login’ which contains all the usernames and passwords of registered users.

**(ii)Book\_Status**

CREATE OR REPLACE PROCEDURE BOOK\_STATUS(

ISBN1 IN INT,

MEM\_ID1 IN VARCHAR2,

STATUS OUT VARCHAR2

) IS

stock NUMBER;

issues\_check VARCHAR2(1);

reserve\_check VARCHAR2(1);

BEGIN

stock := 0;

-- Handling for issues\_check

BEGIN

SELECT 'Y' INTO issues\_check

FROM Issues

WHERE ISBN = ISBN1 AND MEM\_ID = MEM\_ID1;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

issues\_check := NULL;

END;

-- Handling for reserve\_check

BEGIN

SELECT 'Y' INTO reserve\_check

FROM Reservation

WHERE ISBN = ISBN1 AND MEM\_ID = MEM\_ID1;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

reserve\_check := NULL;

END;

-- Calculate stock using the BOOK\_STOCK function

stock := BOOK\_STOCK(ISBN1);

-- Determine status based on checks

IF reserve\_check IS NOT NULL AND issues\_check IS NULL THEN

STATUS := 'RESERVED';

ELSIF issues\_check IS NOT NULL AND reserve\_check IS NULL THEN

STATUS := 'BORROWED';

ELSE

IF stock = 0 THEN

STATUS := 'UNAVAILABLE';

ELSE

STATUS := 'AVAILABLE';

END IF;

END IF;

END;

/

**Functionality:** The procedure Book\_Status is used to **classify each book** (also taking into consideration the member ID to one of the following categories

* Reserved
* Borrowed
* Available
* Unavailable

First, the procedure uses the member ID and the ISBN searched, and checks for records in the Reservation and Issues table for the particular member ID and ISBN pair to find if that book is either borrowed (record available in the issues table) or reserved ( record available in the reservation table) by the user already and the set the status accordingly to ‘**BORROWED**’ or ‘**RESERVED**’.

If the book is neither reserved nor borrowed, it calculates the number of books currently available with the library using the BOOK\_STOCK function and sets the status to ‘**AVAILABLE**’ or ‘**UNAVAILABLE**’.

**(iii) Random\_Check\_Issues:**

CREATE OR REPLACE PROCEDURE RANDOM\_CHECK\_ISSUES (

random\_value IN VARCHAR2,

value\_exists OUT INT

)

IS

BEGIN

SELECT COUNT(\*)

INTO value\_exists

FROM ISSUES

WHERE issue\_id= random\_value;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

value\_exists := 0;https://docs.google.com/document/d/1Dr9b5ydoWg6bQW2fpSu-vOABD76ILpum/edit?usp=sharing&ouid=103133928149497977141&rtpof=true&sd=true

WHEN OTHERS THEN

value\_exists := -1;

RAISE;

END RANDOM\_CHECK\_ISSUES;

/

**Functionality:** The procedure ‘Random\_check\_issues’ checks if the random Issue\_ID generated (in Netbeans Frontend) during the insertion of a new issue into the ‘Issues’ table is already present in the Issues table or not. This allows us to ensure unique Issue\_ID is generated **to avoid Entity Integrity Constraint** since Issue\_ID is a primary key.

**(ivi) Random\_Check\_Renewals:**

CREATE OR REPLACE PROCEDURE RANDOM\_CHECK\_RENEWALS (

random\_value IN VARCHAR2,

value\_exists OUT INT)

IS

BEGIN

SELECT COUNT(\*)

INTO value\_exists

FROM RENEWALS

WHERE renew\_id= random\_value;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

value\_exists := 0;

WHEN OTHERS THEN

value\_exists := -1;

RAISE;

END RANDOM\_CHECK\_RENEWALS;

/

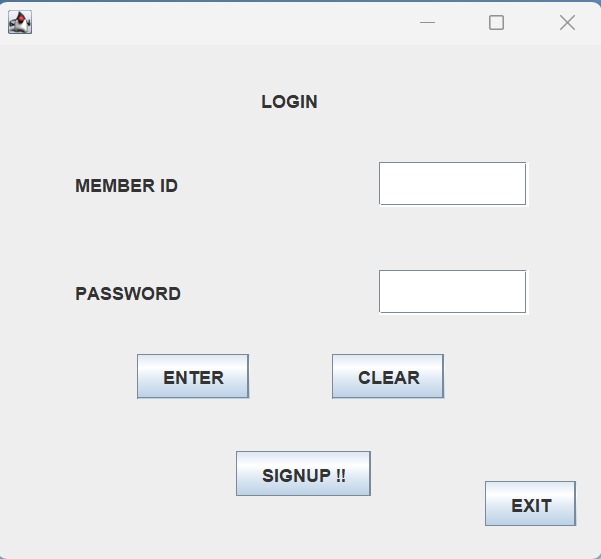
**Functionality:** The procedure ‘Random\_check\_renewals’ checks if the random Renew\_ID generated (in Netbeans Frontend) during the insertion of a new renewal into the ‘Renewals’ table is already present in the Renewals table or not. This allows us to ensure unique Renew\_ID is generated **to avoid Entity Integrity Constraint** since Renew\_ID is a primary key.

**Sample Demo:**

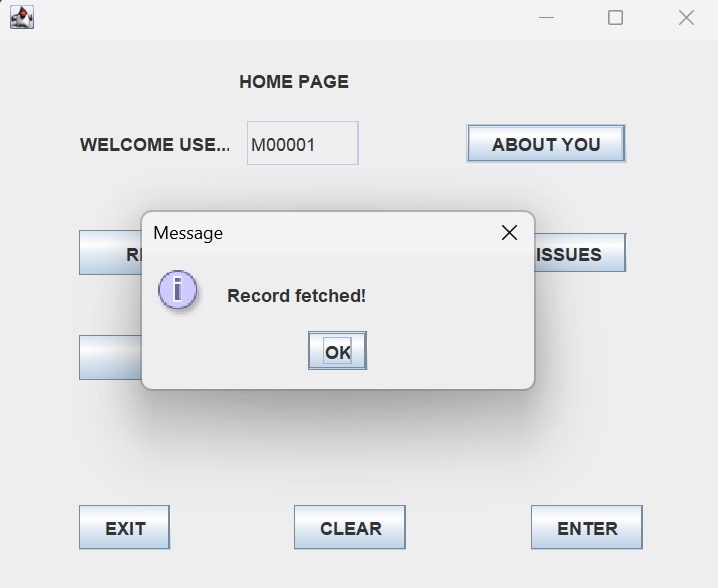
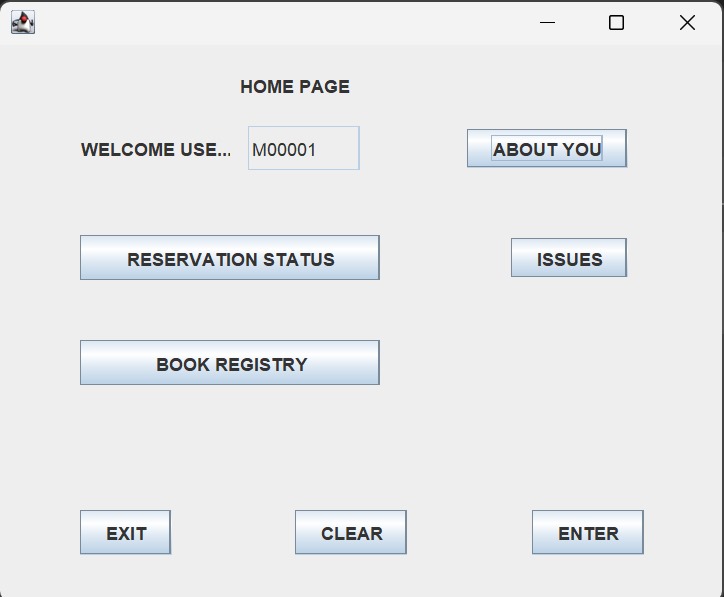
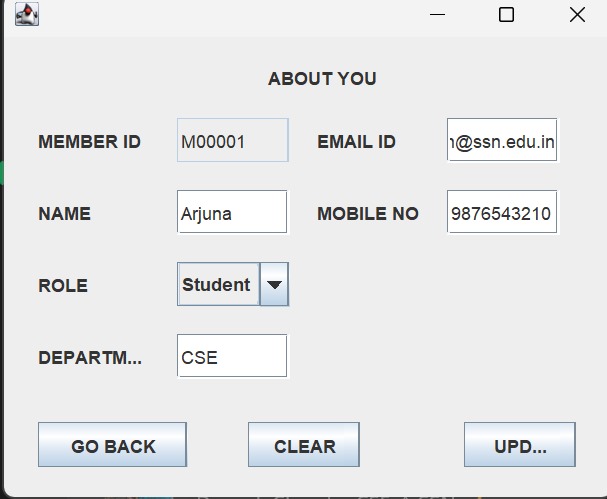
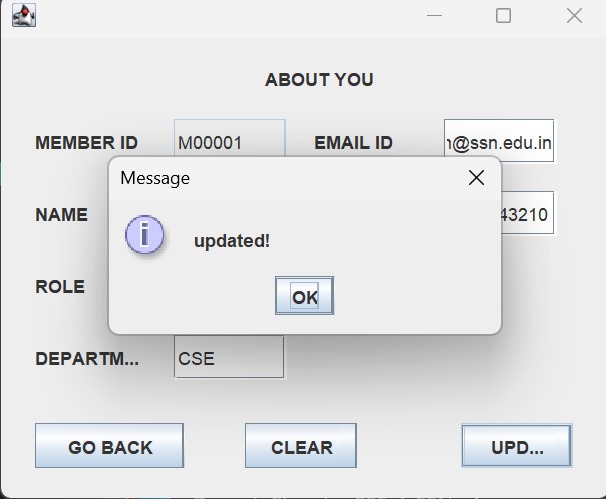
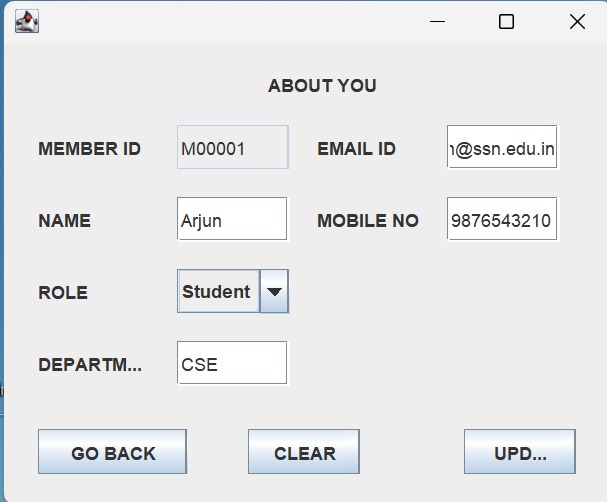
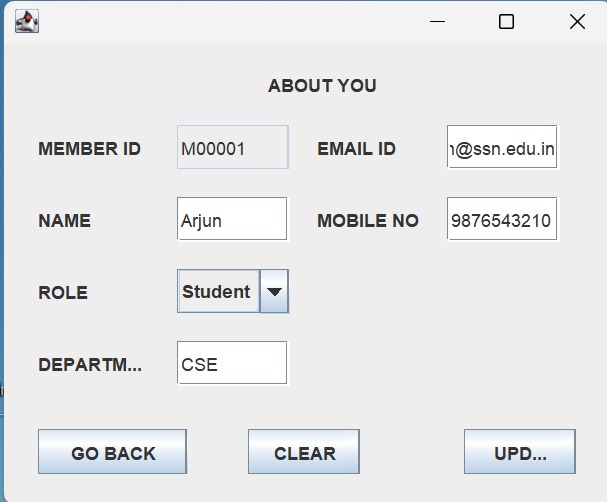
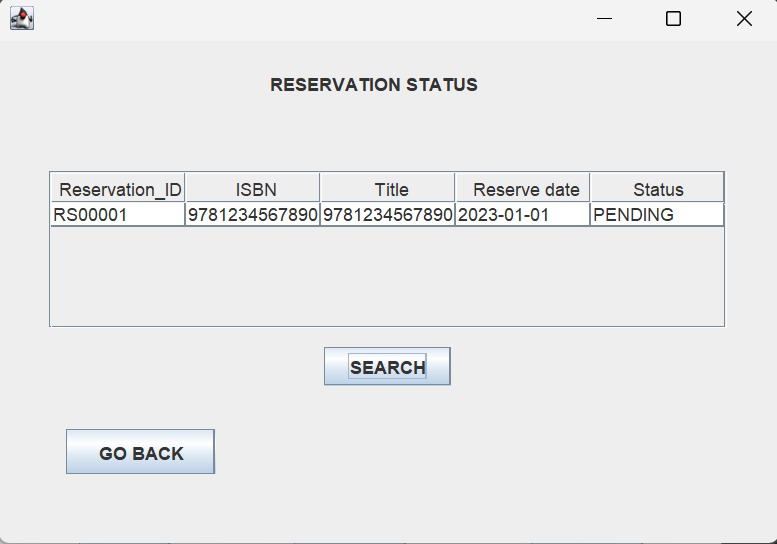
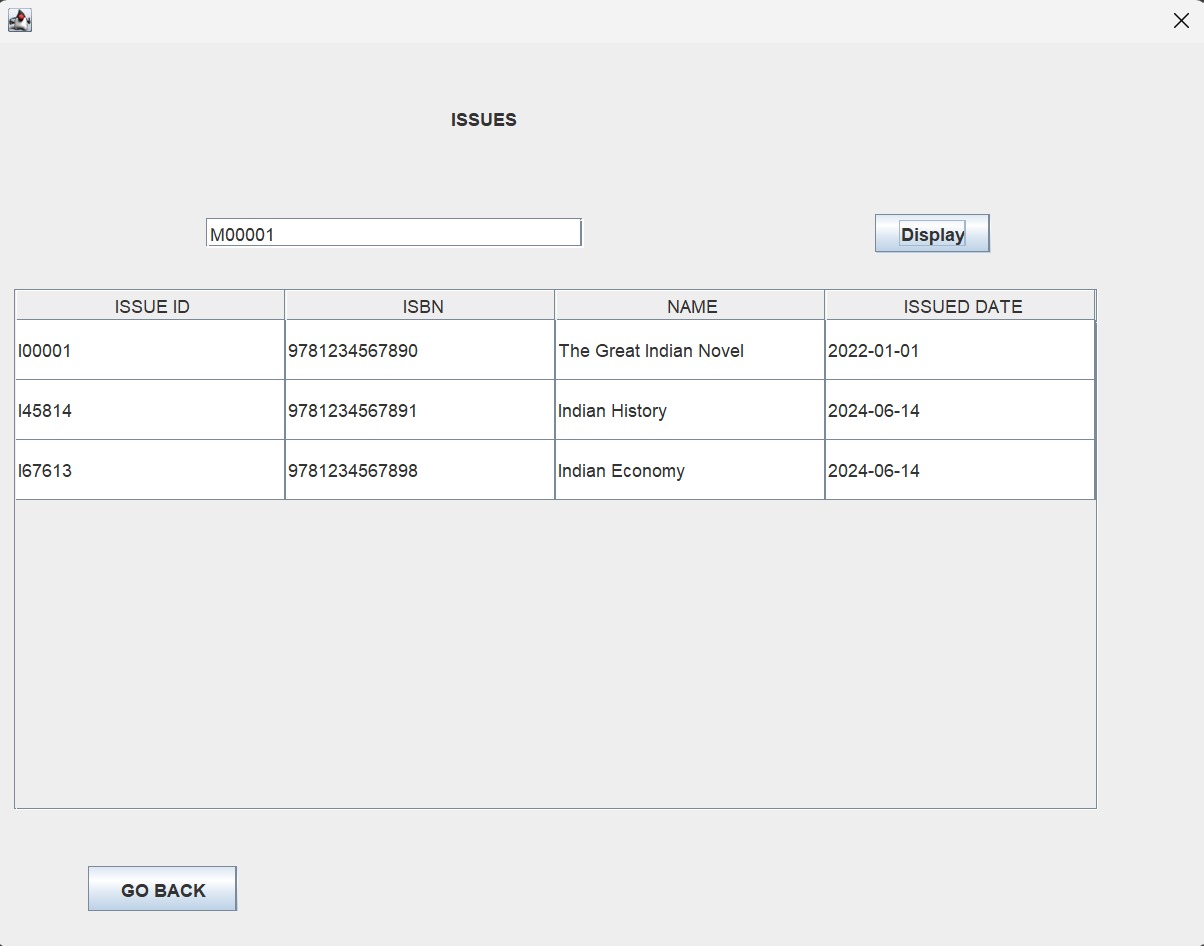
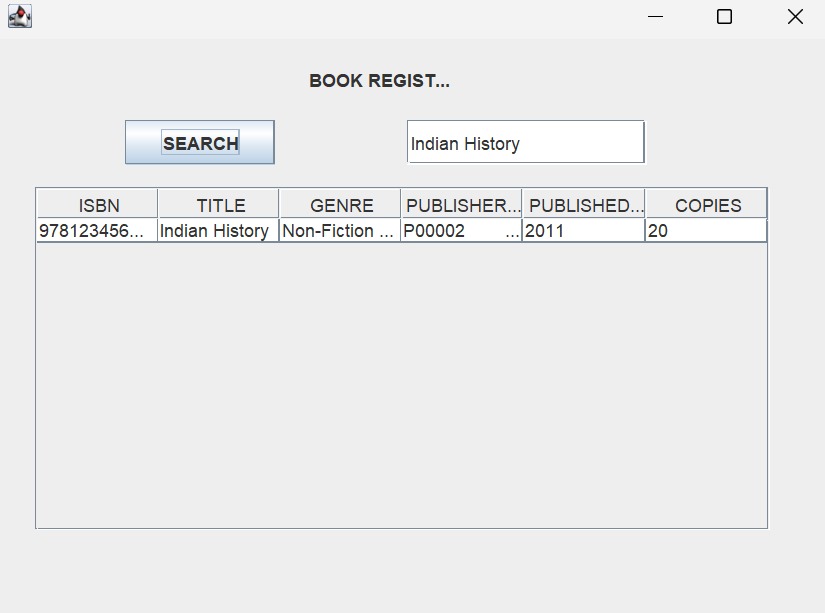
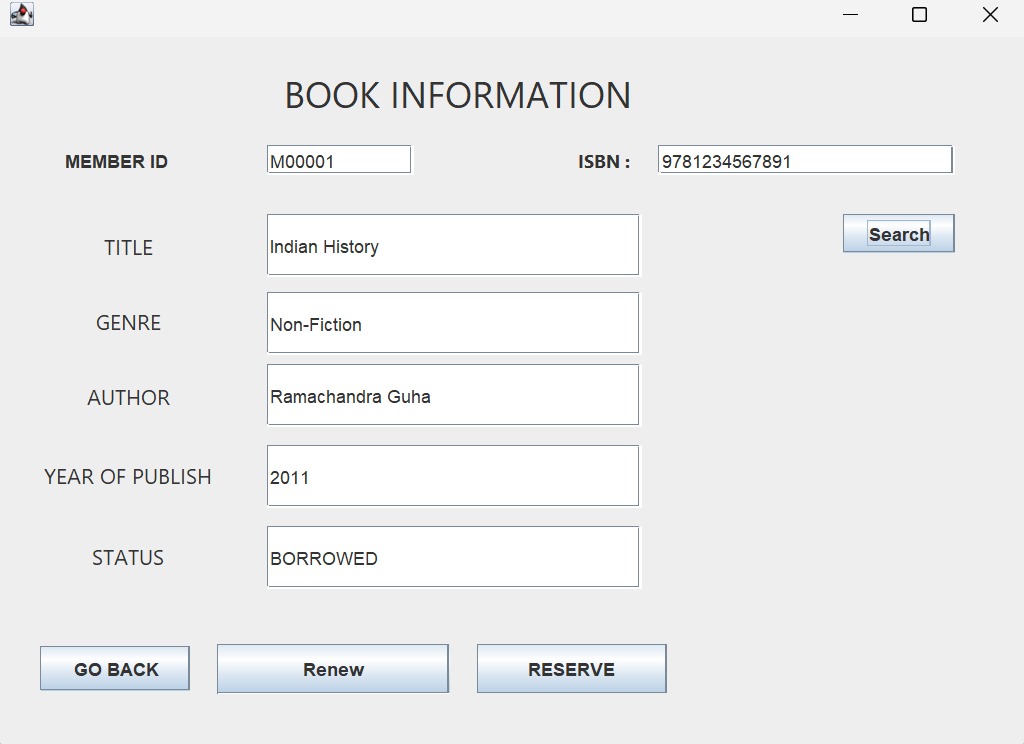
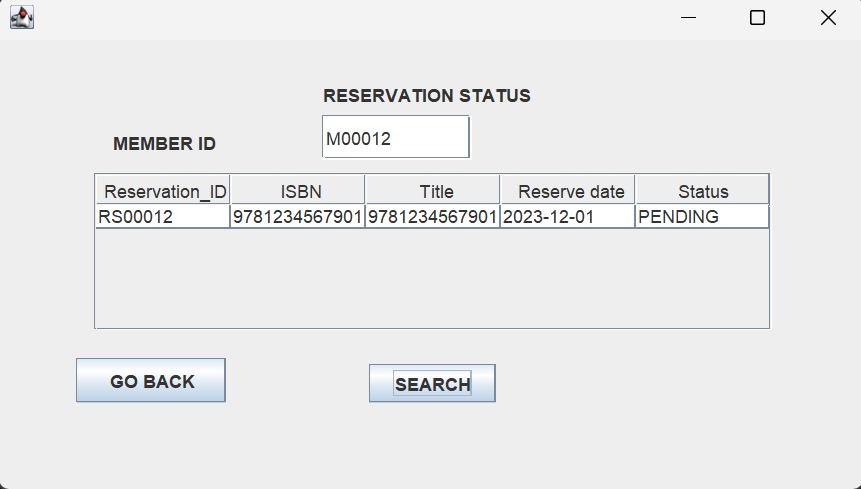
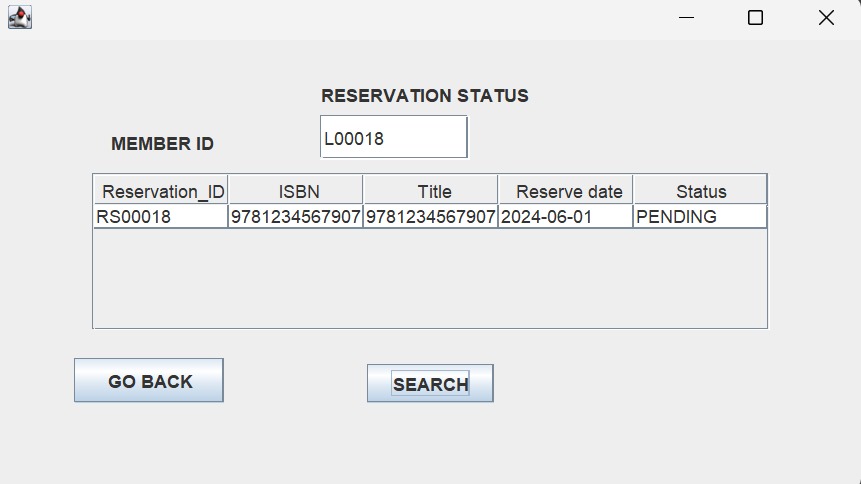
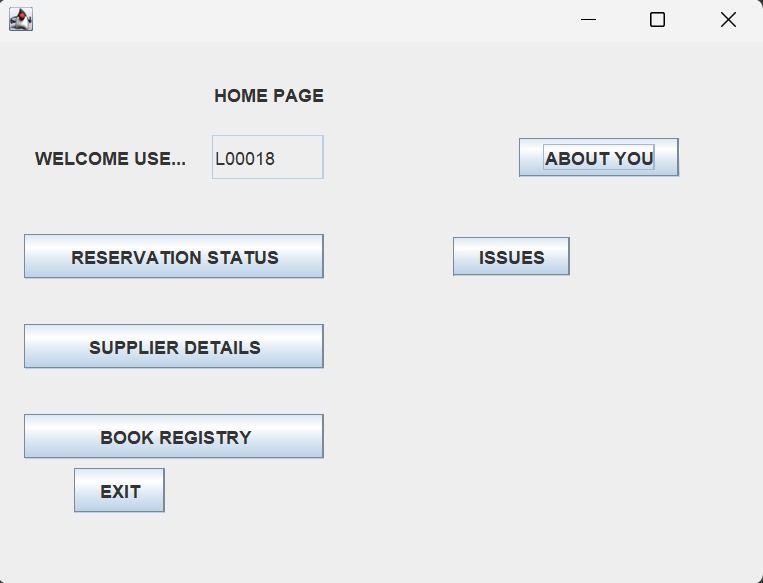
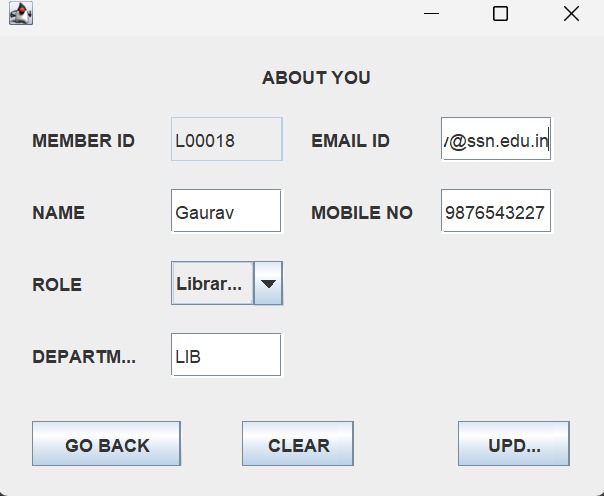
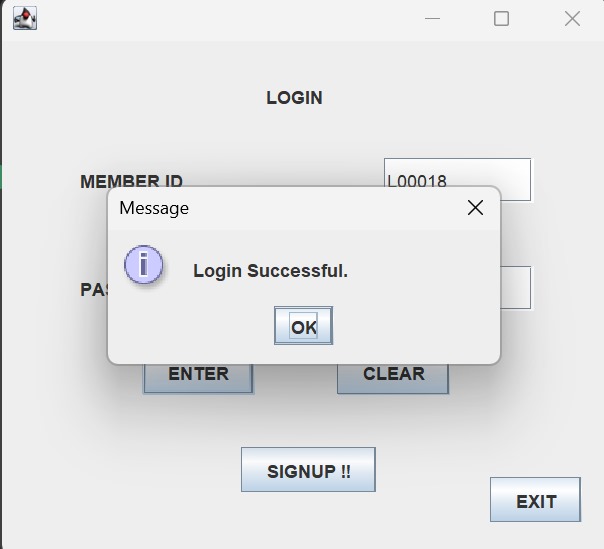
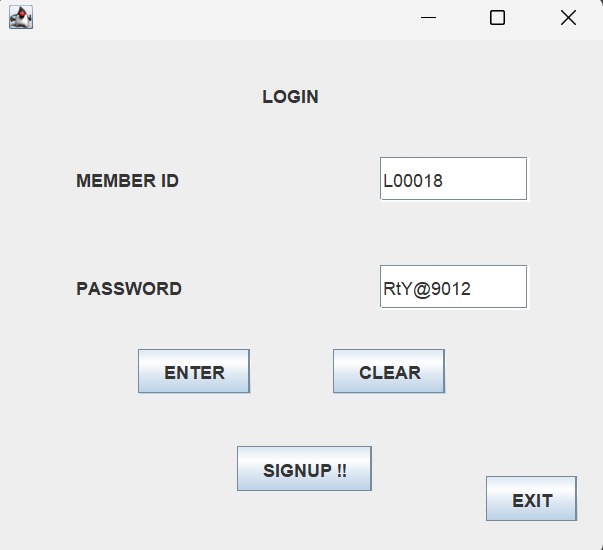
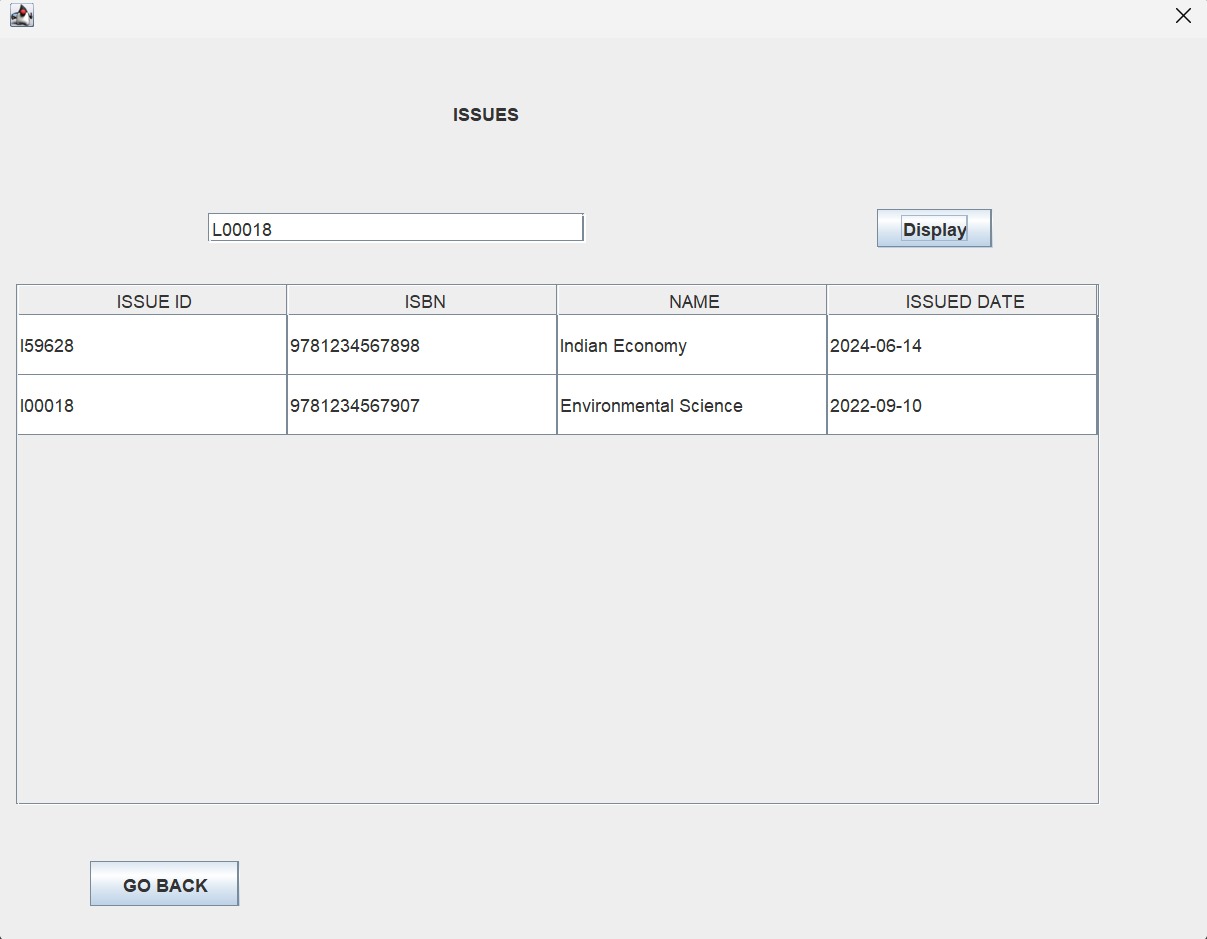
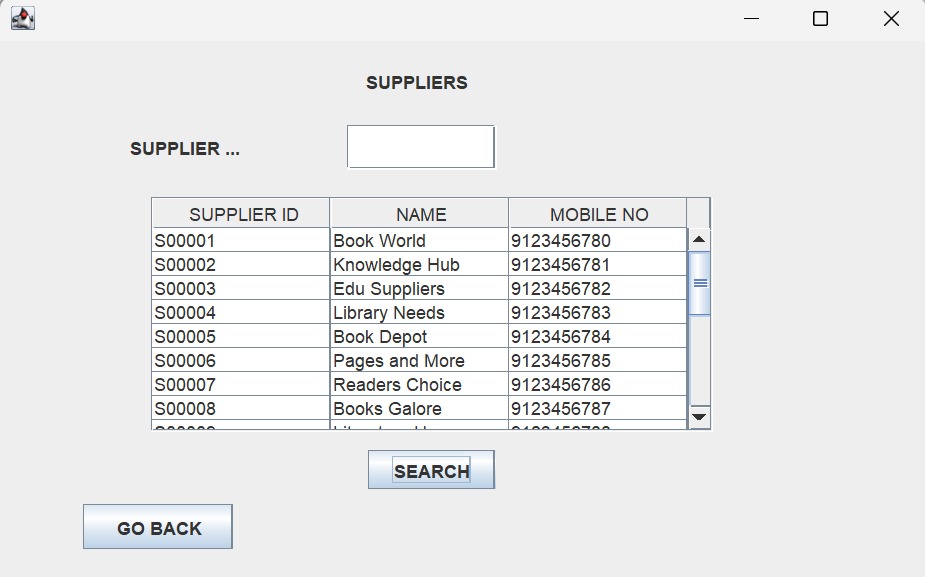
**Login page of the user:**

It permits the user to login based on their preventially saved credentials when they signedup for the first time .In our app we have two kinds of login one is the members of the library (the users) and then the librarian(admin).

Lets go through the member’s login first



**This is the homepage**

**This is the about you page:****This is the reservation page:****This is the issues page:****This is the books registry page to search for a book** **This is the searched and selected book information page** **This is the admin login page** **This is the supplier details page:****This is the page where the admin can search for the issues given to the users**