VSY LIBRARY

LIBRARY MANAGEMENT SYSTEM

UCS2411 – Database Lab

PROJECT REPORT

Submitted By

Vidisha Desai 3122 22 5001 154

V Yuktha 3122 22 5001 166

J M Sunil Sairaj 3122 22 5001 144

Department of Computer Science and Engineering

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Kalavakkam – 603110

December 2023

INDEX

ASSUMPTIONS MADE FOR THE LIBRARY MANAGEMENT SYSTEMS:

The Library management system in this project is based on the following assumptions and criteria:

> The book is identified by book_id and there can be multiple coies of a book with same book_id which is also kept track of, then there is title, author_id of the authors who have written the book, publisher_id of the publisher, and isbn.

> The author is uniquely identified by author_id and also has name, dob, nationality , awards received.

> The awards and also stored in the data bases which is uniquely identified by an anward_no and also keeps track of the book_id for which the award is given for, th author_id for whom the award is given for and the year_awarded.

> Each member of the library has his own member_id and fine_id (which will created at the same time as member id) , and also contact, date of registration.

> The user can borrow books from the library where it comes with a unique borrow_id, associated with book_id, date of issue , due date_and the number of days past due which is calculated by sys-date - due_date. the user can borrow 5 books at max at a time. (and must return all before taking any other books).

> Fines are maintained in the data base associated with the brrow_id , member_id , fine_id and the fine amount is calculated as no_of _days past due * 0.1* book_price.

> Members can make reservation for a particular book at any branch which is identified by a unique reservation_no , associated wih the book_id of the book reserved, member_id and the date of reservation. The reservation made must be only valid till 2 weeks after the reservation is made.

ENTITIES AND THEIR ATTRIBUTES:

1.We are developing an application that helps managing the library more efficiently to make it more accessible for both the workers, supplier, and the member who has membership of the library.

2. Entity **Books**
The attribute of entity Books has attributes like:
A. **Book_id**: An unique id for each individual book of a distinct publisher. (Which is same for all the copies of the same book)

B. **Publisher_id** : this is an unique ID given for each individual publishing company.

C. **Author_id** : An unique id is given by the library for every individual author who has published a book so far.

D. **Title** : his attribute is the name of the book

E. **Book_price**: the book price is also considered as an attribute in this Entity

F. **Copies_available**: the copies available of a particular book of particular publication is determined.

G. **ISBN**: this is a unique number given to publication companies which changes at a period of time.

Fds{ book_id --> publisher_id, author_id, Title, copies_available, ISBN
Book_id, ISBN --> Book_price
}

3. Entity **authors**:

A. **Author_id** : this is a unique id given to the individual authors to avoid the situations where two authors have the same author name.

B. **Author_name**: the name of the author

C. **country:** the nationality of the author

D. **Author_ mail**: the email-id of the author

E. **Author_dob**: the date of birth of the author.

FdS { Author_id --> Author_name,country, Author_mail, Author_dob

Author_mail-> author_name
}

4. Entity **Fines** has the following attributes:

A. **Fine_id**: it is a unique id for determining the fine of every individual member which initially sums upto zero then if any number offline generated on a particular member then the fine amount is added to the particular fine_id

B. **Fine_amount**: it is an derived attribute calculated on number of days past due * 10% of book_price. And then added to the final fine amount.

C. **Member_id**: unique id given to a member at the time of registering for the membership.

FDs: { Fine_id → Fine amount, Member_id;
}

5. Entity **Publishers** has following attributes:

A. **Publisher_ID**: unique id given to each publishing company.
B. **Name**: it is the name of each publisher that has published the books available in the library.
C. **Contact**: contact number of the publishers
D. **ISBN**: a unique number to identify the publisher and the year of publication.
E. **Published_yr**: the year of book publication

Fds:
Publisher_ID -> Name, contact
Publisher_ID, publisher_Yr-> ISBN

6. Attributes of entity **Borrowed_Books** includes attributes:

A. **Borrowed_id**: a new record is made every time a user takes a book after returning previous books.
B. **Book_id**: the book that has been borrowed
C. **Date_borrowed**: date of borrowing the book
D. **Due_date**: the book has to be returned by 2 weeks after the date of borrowing.
E. **Returned_date**: the date when the borrower returns the book. Based on the difference between the return date and due date the fines are calculated.
F. **Member_ID**: member who has borrowed the book.

Fds:{ Borrowed_id -> Member_ID, Date_borrowed, Due_date, Returned_date
Member_ID, book_id ->Borrowed_id}

7. Entity **Members** includes following attributes:

A.**Member_id** : A unique id given to each member at the time of registration, only with a member id an individual can borrow books from the library.

B. **Fine_id**: a unique fine id is generated along with the member_id for each member the corresponding fine amount to a fine id is initially 0.

C. **Borrow_id** : this unique borrow id is generated along with the member id for each member and under a borrow id a person can borrow a maximum of 5 books at a time.

D. **Member_name**: name of the individual registered for the membership.

E. **Member_DOB**: date of birth of the individual registered for the membership.

F. **contact_phonem**: phone number of the individual.

G. **Date_of_registration**: the date of registration for the membership.

F. **expiry_date**: expiry date of member_ship.

FDs { member_id -> Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry date;
Fine_id, Member_id → Borrow_id.}

8. Entity **Awards** has the following attributes:

A. **Award_number**: A particular number given to the award as there are only countable awards for literature.

B. **Award_name**: name of the award.

C. **Author_id**: author who has received the particular award.

D. **Title:** The title of the book for which the award has been given.

E. **year**: The year in which the award was awarded.

FDs { Author_id → Award_number, Award_name, Year
Title → Award_number, Award_name, Year}

9. Entity **Reservations**:

A. **Reservation_number**: a unique reservation number is generated for each member who wants to reserve certain copies of books before hand.

B. **Member_id**: member who makes the reservation.

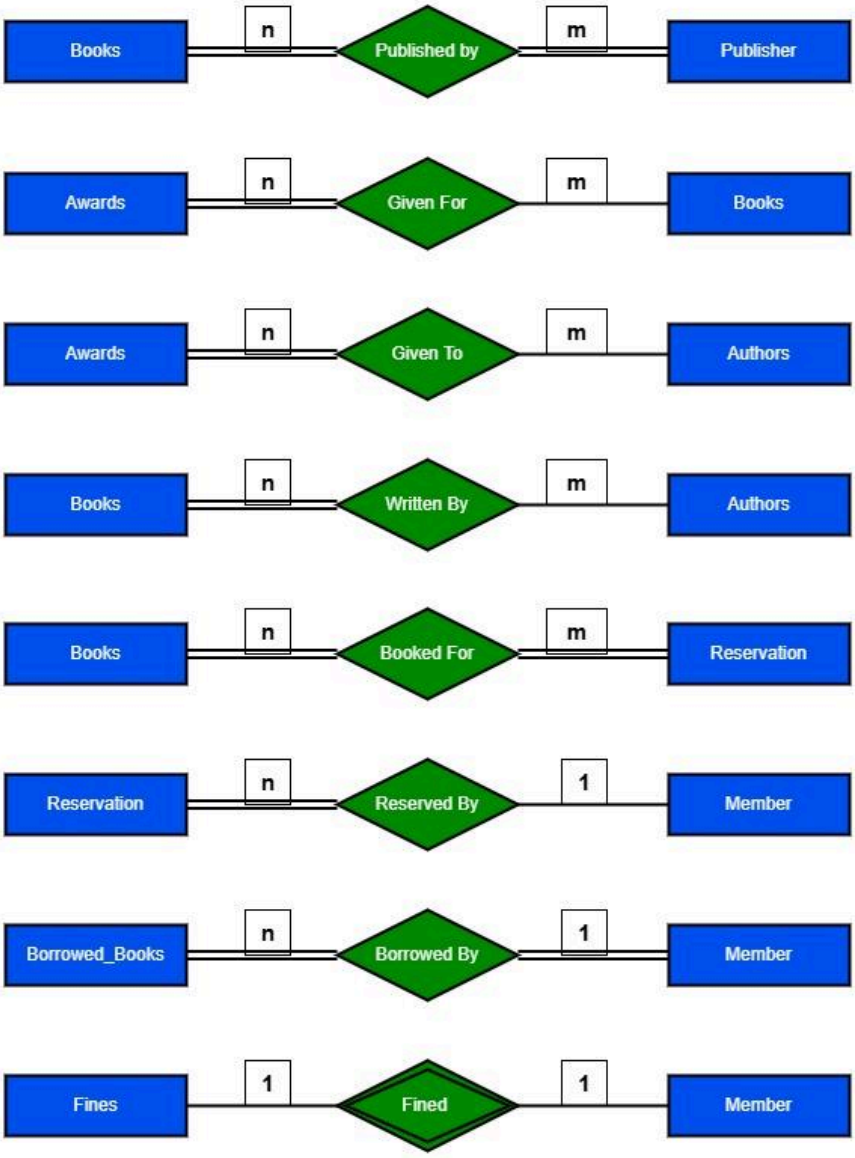
C. **date_of_reservation:** date in which the reservation was made.

D. **Book_id**: the book on which the reservation had been made

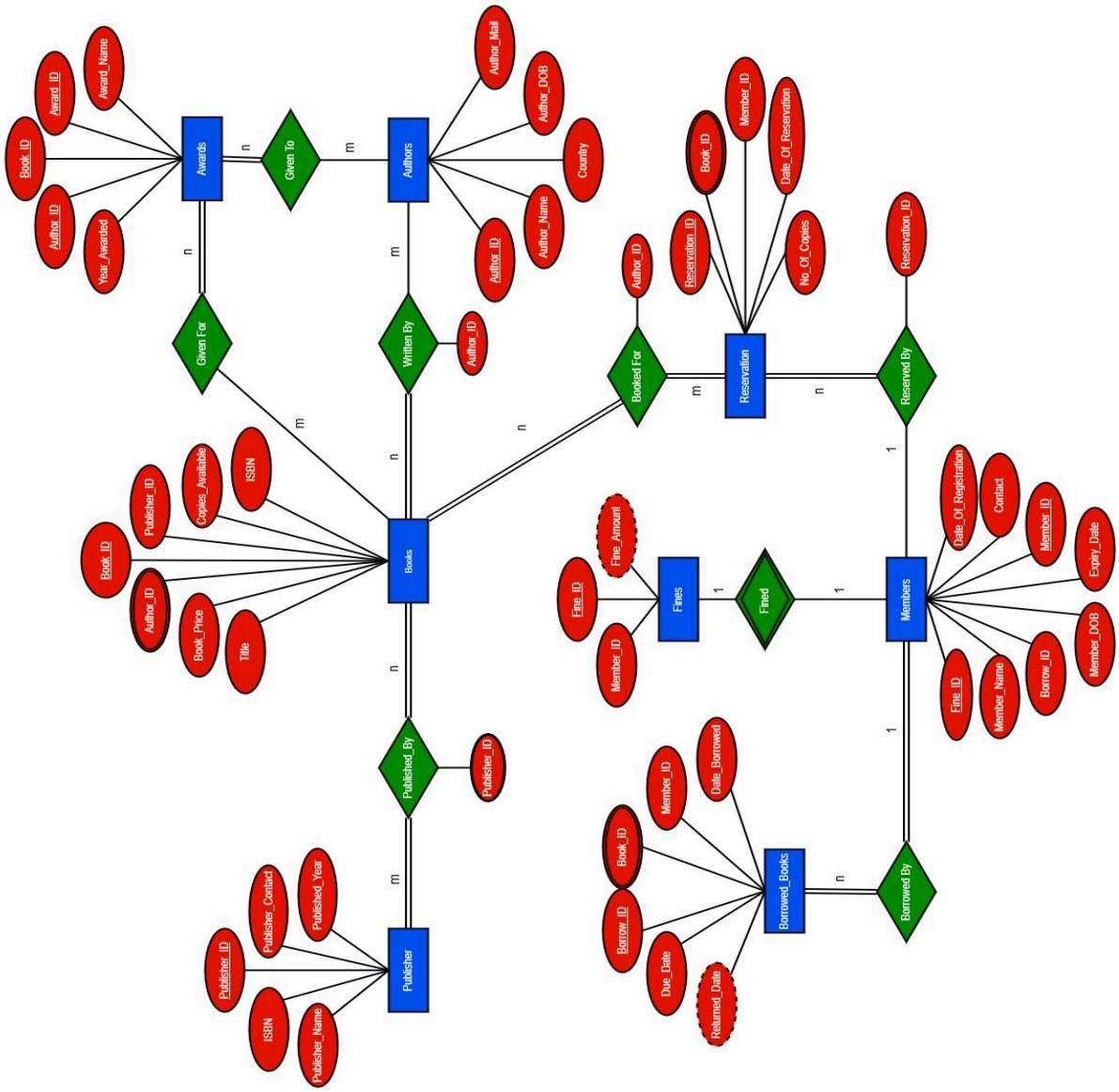
E. **no_copies**: no.of copies of the book wanted

FDs { Member_id, Book_id → Reservation_number, No_copies
Book_id → Reservation_number, Member_id, Date_of_reservation, No_copies }

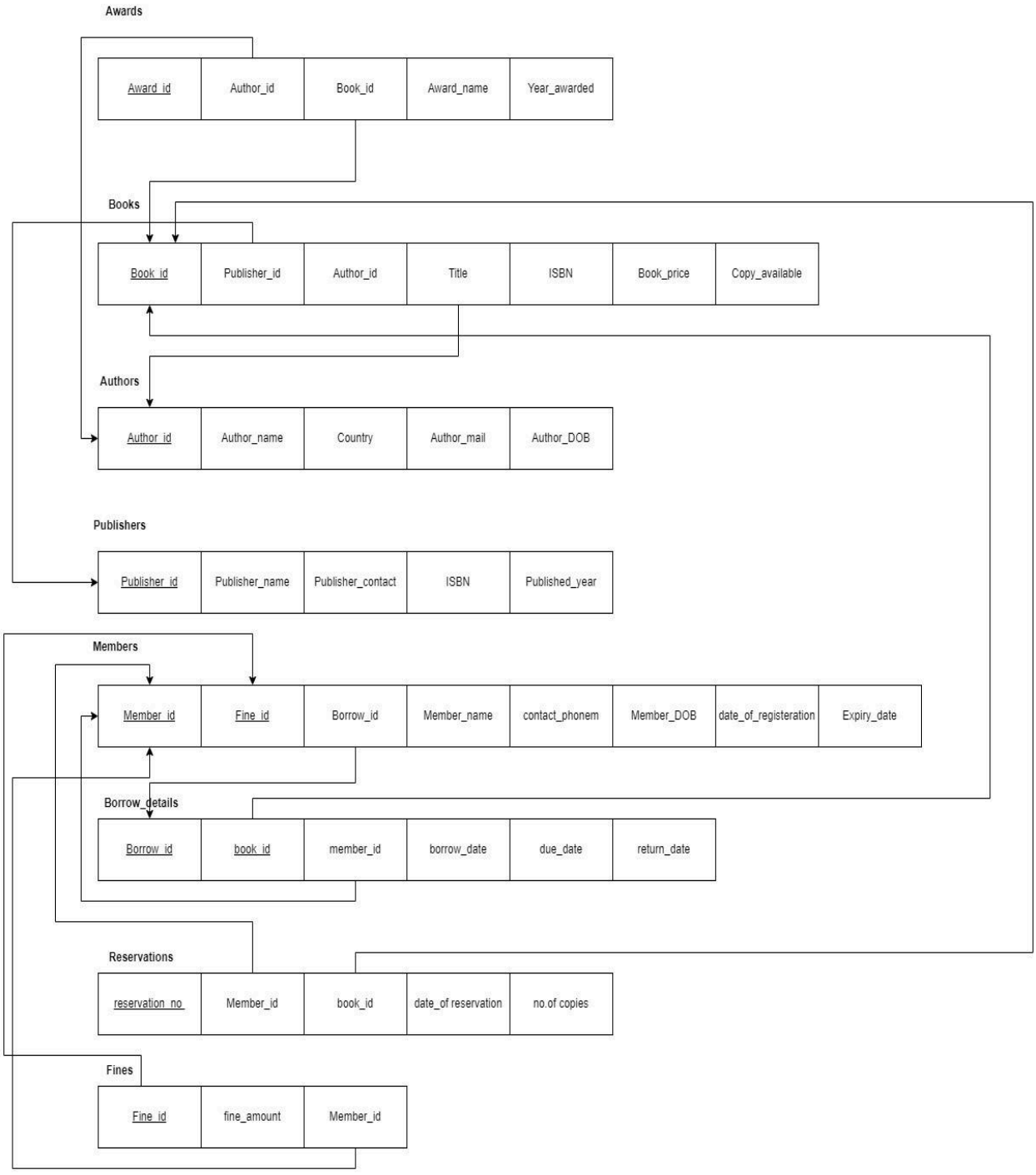
Relations


Books
n
Published by
m
Publisher
Awards
n
Given For
m
Books
Awards
n
Given To
m
Authors
Books
n
Written By
m
Authors
Books
n
Booked For
m
Reservation
Reservation
n
Reserved By
1
Member
Borrowed_Books
n
Borrowed By
1
Member
Fines
1
Fined
1
Member

ENTITY RELATION DIAGRAM:



SCHEMA WITHOUT NORMALIZATION:



NORMALIZING ALL THE FD'S OF RELATIONS:

1.Books (Book_id, Publisher_id , Author_id, Title, Book_price, Copies_available, ISBN)

Fds{ book_id --> publisher_id, author_id, Title, copies_available, ISBN
Title, ISBN --> Book_price
}

(i) Irreducible set:

For: Title, ISBN --> Book_price
Assume:
Title is extraneous :
ISBN -> Book_price :
{ISBN}+ = {ISBN, Book_price}

ISBN is extraneous :
Title -> Book_price :
{Title }+ = {Title , Book_price}
Therefore: no extraneous attributes.

Considering fd:
{Title, ISBN}+ = {Title, ISBN , Book_price}
Without considering the fd:
{Title, ISBN}+ = {Title, ISBN }

book_id --> publisher_id:

Considering fd:
{book_id}+ = {book_id , publisher_id, author_id, Title, copies_available,ISBN,Book_price}
Without considering the fd:
{book_id}+ = {book_id , author_id, Title, copies_available, ISBN,Book_price}

The closure without considering and with consideration of fd: **book_id --> publisher_id, is not the same hence not redundant.**

book_id --> author_id:

Considering fd:
{book_id}+ = {book_id , publisher_id, author_id, Title, copies_available,ISBN,Book_price}
Without considering the fd:
{book_id}+ = {book_id , publisher_id, Title, copies_available, ISBN,Book_price}

The closure without considering and with consideration of fd: **book_id --> author_id is not the same and hence not redundant.**

book_id --> Title:

Considering fd:
{book_id}+ = {book_id , publisher_id, author_id, Title, copies_available,ISBN,Book_price}
Without considering the fd:
{book_id}+ = {book_id , publisher_id, author_id, copies_available,ISBN}

The closure without considering and with consideration of fd: **book_id --> Title, is not the same hence not redundant.**

book_id --> copies_available:

Considering fd:
{book_id}+ = {book_id , publisher_id, author_id, Title, copies_available,ISBN,Book_price}
Without considering the fd:
{book_id}+ = {book_id , publisher_id, author_id, Title,ISBN,Book_price}

The closure without considering and with consideration of fd: **book_id --> copies_available, is not the same hence not redundant.**

book_id --> ISBN:

Considering fd:
{book_id}+ = {book_id , publisher_id, author_id, Title, copies_available,ISBN,Book_price}
Without considering the fd:
{book_id}+ = book_id , publisher_id, author_id, Title, copies_available}

The closure without considering and with consideration of fd: **book_id --> ISBN, is not the same hence not redundant.**

Irreducible fd's:
book_id --> publisher_id, author_id, Title, copies_available, ISBN
Title, ISBN --> Book_price

(ii) The relation is in 1NF
The relation is already in 1NF since all attributes are atomic (i.e., indivisible).

(iii) Candidate Key:
Closure set:
{Book_id, Publisher_id , Author_id, Title, Book_price, Copies_available, ISBN **}->{**Book_id, Publisher_id , Author_id, Title, Book_price, Copies_available, ISBN **}**

By considering book_id --> publisher_id, author_id, Title, copies_available, ISBN

Closure set:

{Book_id, Book_price**}->{**Book_id, Publisher_id , Author_id, Title, Book_price, Copies_available, ISBN **}**

By considering Title, ISBN --> Book_price

Closure set:

{Book_id**}->{**Book_id, Publisher_id , Author_id, Title, Book_price, Copies_available, ISBN **}**

So the candidate key is Book_id
Prime Attribute (PA) : Book_id
Non Prime Attribute (NPA) : Publisher_id , Author_id, Title, Book_price, Copies_available, ISBN

(iv) 2NF:

If the table should be in 2NF then there should be no partial dependencies in the table. A partial dependency is when a proper subset of candidate keys determines non-prime attributes.

The FDs are:

book_id --> publisher_id, author_id, Title, copies_available, ISBN
Title, ISBN --> Book_price

Since the candidate key here doesn't have any proper subset, by default it is in 2NF.

(v) 3NF:

A relation is in 3NF if there exists no transitive dependencies.
A transitive dependency is when non-prime attributes determine some other non-prime attributes.

The FDs are:

book_id --> publisher_id, author_id, Title, copies_available, ISBN
Title, ISBN --> Book_price

Here Title, ISBN are NPA and the fd: Title, ISBN --> Book_price has Transitive dependency . As 1 fd violates need to decompose into two relations

The relations after decomposing are:

Book_details (Book_id, Publisher_id , Author_id, Title, Copies_available, ISBN)
Candidate key: Book_id

Bost_cost (Title, ISBN, Book_price)

Candidate key : Title,ISBN

In both the realtions, there is no partial dependency and transitive dependency they are now in 2nf and 3 nf.

vi)BCNF:

Since all the relations FDs have superkey as their LHS , the relations are in BCNF.
Therefore the final relations for the Books is: Book_details and Book_cost

==========

2. Relation authors (Author_id , Author_name, Country, Author_ mail, Author_dob)

Fd's { Author_id --> Author_name,country, Author_mail, Author_dob
Author_mail-> author_name
}

(i) Irreducible set:

Since no two elements in the LHS, we directly reduce the attributes in RHS.

Author_id --> Author_name:

Considering fd:
{Author_id}+ = {Author_id , Author_name, Country, Author_ mail, Author_dob}
Without considering the fd:
{Author_id}+ = {Author_id , Country, Author_ mail, Author_dob}

The closure without considering and with consideration of fd: **Author_id --> Author_name ,is not the same hence not redundant.**

Author_id --> country:

Considering fd:
{Author_id}+ = {Author_id , Author_name, Country, Author_ mail, Author_dob}
Without considering the fd:
{Author_id}+ = {Author_id , Author_name, Author_ mail, Author_dob}

The closure without considering and with consideration of fd: **Author_id --> country, is not the same hence not redundant.**

Author_id --> Author_mail:

Considering fd:
{Author_id}+ = {Author_id , Author_name, Country, Author_ mail, Author_dob}

Without considering the fd:
{Author_id}+ = {Author_id , Author_name, Countryl, Author_dob}

The closure without considering and with consideration of fd: **Author_id --> Author_mail is not the same and hence not redundant.**

Author_id --> Author_dob:

Considering fd:
{Author_id}+ = {Author_id , Author_name, Country, Author_ mail, Author_dob}
Without considering the fd:
{Author_id}+ = {Author_id , Author_name, Country, Author_ mail}

The closure without considering and with consideration of fd: **Author_id --> Author_dob, is not the same hence not redundant.**

Irreducible fd's:
Author_id → Author_name country, Author_mail, Author_dob

(ii) The relation is in 1NF
The relation is already in 1NF since all attributes are atomic (i.e., indivisible).

(iii) Candidate Key:
Closure set:
{Author_id , Author_name, Country, Author_ mail, Author_dob **}->{**Author_id , Author_name, Country, Author_ mail, Author_dob **}**

By considering Author_id → country, Author_mail, Author_dob
Closure set:
{Author_id**}->{**Author_id , Author_name, Country, Author_ mail, Author_dob **}**

So the candidate key is Author_id
Prime Attribute (PA) : Author_id
Non Prime Attribute (NPA) : Author_name, Country, Author_ mail, Author_dob

(iv) 2NF:

If the table should be in 2NF then there should be no partial dependencies in the table.

A partial dependency is when a proper subset of candidate keys determines non-prime attributes.

The FDs are:

Author_id → country, Author_mail, Author_dob

Since the candidate key here doesn't have any proper subset, by default it is in 2NF.

(v) 3NF:

A relation is in 3NF if there exists no transitive dependencies.
A transitive dependency is when non-prime attributes determine some other non-prime attributes.

The FDs are:

Author_id → author_name, country, Author_mail, Author_dob

Here no fd has Transitive dependency . Hence The relation **Author (Author_id , Author_name, Country, Author_ mail, Author_dob)** is in 2nf and 3 nf.

vi)BCNF:
Since all the relations FDs have superkey as their LHS , the relations are in BCNF.
Therefore the final relations for the Authors is: Author

==========

3. Relation Fines (Fine_id, Fine_amount, Member_id)

FDs:
1. Fine_id → Fine_amount
2. Fine_id → Member_id

(i) Irreducible Set of Functional Dependencies:

To find the irreducible set, we check if we can remove any attributes from the right-hand side (RHS) of the FDs without changing the closure.

1. Fine_id → Fine_amount
- **Considering FD:**
  {Fine_id}+ = {Fine_id, Fine_amount, Member_id}
- **Without considering the FD:**
  {Fine_id}+ = {Fine_id, Member_id}
- The closure changes without the FD **Fine_id → Fine_amount**, hence it is not redundant.

2. Fine_id → Member_id
- **Considering FD:**
  {Fine_id}+ = {Fine_id, Fine_amount, Member_id}
- **Without considering the FD:**
  {Fine_id}+ = {Fine_id, Fine_amount}
- The closure changes without the FD **Fine_id → Member_id,** hence it is not redundant.

Therefore, the irreducible set of FDs remains:
- Fine_id → Fine_amount
- Fine_id → Member_id

(ii) First Normal Form (1NF):

The relation is in 1NF since all attributes are atomic.

(iii) Candidate Key:

To find the candidate key, we determine the closure of the potential keys.

1. Fine_id:
- **Closure set:**
  {Fine_id}+ = {Fine_id, Fine_amount, Member_id}

Since the closure of {Fine_id} includes all attributes in the relation, Fine_id is a candidate key.

Candidate Key: Fine_id

Prime Attribute (PA): Fine_id

Non-Prime Attributes (NPA): Fine_amount, Member_id

(iv) Second Normal Form (2NF):

A relation is in 2NF if it is in 1NF and there are no partial dependencies.

Since the candidate key (Fine_id) does not have any proper subsets, the relation is in 2NF by default.

(v) Third Normal Form (3NF):

A relation is in 3NF if it is in 2NF and there are no transitive dependencies.

Checking the FDs:
- Fine_id → Fine_amount
- Fine_id → Member_id

There are no transitive dependencies because there are no non-prime attributes determining other non-prime attributes.

Thus, the relation is in 3NF.

(vi) Boyce-Codd Normal Form (BCNF):

A relation is in BCNF if for every FD (X → Y), X is a superkey.

Given FDs:
- Fine_id → Fine_amount (Fine_id is a superkey)
- Fine_id → Member_id (Fine_id is a superkey)

Since all FDs have a superkey on the left-hand side (LHS), the relation is in BCNF.

Final Normalized Relation:

The final relation "Fines" remains:
- Fines(Fine_id, Fine_amount, Member_id)

It is in 1NF, 2NF, 3NF, and BCNF.

==========

4. Publisher (Publisher_ID, Publisher_Name, Publisher_contact, ISBN, Published_yr)

Fds { Publisher_ID -> Publisher_Name, Publisher_contact
Publisher_ID, Publisher_Yr-> ISBN }

(i) Irreducible set:

For: Publisher_ID, Publisher_Yr-> ISBN
Assume:
Publisher_ID is extraneous :
Publisher_Yr-> ISBN
{Publisher_Yr}+ = {ISBN, Publisher_Yr} (but doesn't include published_id)

Publisher_Yr is extraneous :
Publisher_id -> ISBN :

{Publisher_id }+ = {Publisher_id , ISBN, Publisher_Name, Publisher_contact}
(but doesn't include published_yr)

Therefore: no extraneous attributes.

Publisher_ID, Publisher_Yr-> ISBN

Considering fd:
{Publisher_ID, Publisher_Yr}+ = {Publisher_id , ISBN, Publisher_Name, Publisher_contact, Publisher_Yr}
Without considering the fd:
{Publisher_ID, Publisher_Yr}+ = {Publisher_id , Publisher_Name, Publisher_contact,Publisher_Yr}

The closure without considering and with consideration of fd: **Publisher_ID, Publisher_Yr-> ISBN, is not the same hence not redundant.**

Publisher_ID --> Publisher_Name:

Considering fd:
{Publisher_ID}+ = {Publisher_id, Publisher_Name, Publisher_contact}
Without considering the fd:
{Publisher_ID}+ = {Publisher_id, Publisher_contact}

The closure without considering and with consideration of fd: **Publisher_ID --> Publisher_Name, is not the same hence not redundant.**

Publisher_ID --> Publisher_contact:

Considering fd:
{Publisher_ID}+ = {Publisher_id, Publisher_Name, Publisher_contact}
Without considering the fd:
{Publisher_ID}+ = {Publisher_id, Publisher_Name}

The closure without considering and with consideration of fd: **Publisher_ID --> Publisher_contact is not the same and hence not redundant.**

Irreducible fd's:
Publisher_ID -> Publisher_Name, Publisher_contact
Publisher_ID, Publisher_Yr-> ISBN

(ii) The relation is in 1NF
(iii) Candidate Key:

Closure set:
{Publisher_ID , Publisher_Name, Publisher_contact , Publisher_Yr ,ISBN **}->{**Publisher_ID Publisher_Name, Publisher_contact , Publisher_Yr ,ISBN **}**

By considering Publisher_ID -> Publisher_Name, Publisher_contact

Closure set:

{Publisher_ID, Publisher_Yr ,ISBN **}->{**Publisher_ID Publisher_Name, Publisher_contact , Publisher_Yr ,ISBN **}**

By considering Publisher_ID, Publisher_Yr-> ISBN

Closure set:

{Publisher_ID, Publisher_Yr **}->{**Publisher_ID Publisher_Name, Publisher_contact , Publisher_Yr ,ISBN **}**

So the candidate key is : Publisher_ID, Publisher_Yr
Prime Attribute (PA) : Publisher_ID, Publisher_Yr
Non Prime Attribute (NPA) : Publisher_Name, Publisher_contact ,ISBN

(iv) 2NF:

If the table should be in 2NF then there should be no partial dependencies in the table.
A partial dependency is when a proper subset of candidate keys determines non-prime attributes.

The FDs are:

Publisher_ID -> Publisher_Name, Publisher_contact
Publisher_ID, Publisher_Yr-> ISBN

Fd 1 violates the partial dependency, and decomposed into two relations :
Publisher_details (Publisher_id, Publisher_Name, Publisher_Contact)
Candidate key : Publisher_id
Publisher_ISBN (Publisher_id, Publisher_yr, ISBN)
Candidate key: Publisher_id, Publisher_yr

Both publisher_details and publisher_ISBN are not having any proper subset that violates partial dependency, hence in 2NF

(v) 3NF:

A relation is in 3NF if there exists no transitive dependencies.
A transitive dependency is when non-prime attributes determine some other non-prime attributes.

For relation Publisher_details (Publisher_id, Publisher_Name, Publisher_Contact)
Candidate key : Publisher_id
The FDs are:

Publisher_ID -> Publisher_Name, Publisher_contact
Here, there is no transitive dependency. Hence in 3NF.

For relation Publisher_ISBN (Publisher_id, Publisher_yr, ISBN)
Candidate key: Publisher_id, Publisher_yr
The FDs are:
Publisher_ID, Publisher_Yr-> ISBN

Here, there is no transitive dependency. Hence in 3NF.

vi)BCNF:
Since all the relations FDs have superkey as their LHS , the relations are in BCNF.
Therefore the final relations for the Publisher is: Publisher_details and Publisher_ISBN

==========

5. Borrowed_Books (Borrowed_id, Book_id, Date_borrowed, Due_date, Returned_date, Member_ID)

FDs :
1. Borrowed_id → Member_ID, Date_borrowed, Due_date, Returned_date
2. Member_ID, Book_id → Borrowed_id

(i) Irreducible Set of Functional Dependencies:

To find the irreducible set, we check if we can remove any attributes from the right-hand side (RHS) of the FDs without changing the closure.

1. Borrowed_id → Member_ID, Date_borrowed, Due_date, Returned_date
- Considering FD:
{Borrowed_id}+ = {Borrowed_id, Member_ID, Date_borrowed, Due_date, Returned_date, Book_id}
- **Without considering the FD:**
{Borrowed_id}+ = {Borrowed_id, Book_id}
- The closure changes without the FD Borrowed_id → Member_ID, Date_borrowed, Due_date, Returned_date, hence it is not redundant.

2. Member_ID, Book_id → Borrowed_id
- Considering FD:
{Member_ID, Book_id}+ = {Member_ID, Book_id, Borrowed_id, Date_borrowed, Due_date, Returned_date}
- **Without considering the FD:**
{Member_ID, Book_id}+ = {Member_ID, Book_id}

- The closure changes without the FD Member_ID, Book_id → Borrowed_id, hence it is not redundant.

Therefore, the irreducible set of FDs remains:
- Borrowed_id → Member_ID, Date_borrowed, Due_date, Returned_date
- Member_ID, Book_id → Borrowed_id

(ii) First Normal Form (1NF):

The relation is in 1NF since all attributes are atomic.

(iii) Candidate Key:

To find the candidate key, we determine the closure of the potential keys.

1. Borrowed_id:
- **Closure set:**
{Borrowed_id}+ = {Borrowed_id, Member_ID, Date_borrowed, Due_date, Returned_date, Book_id}

Since the closure of {Borrowed_id} includes all attributes in the relation, Borrowed_id is a candidate key.

2. Member_ID, Book_id:
- **Closure set:**
{Member_ID, Book_id}+ = {Member_ID, Book_id, Borrowed_id, Date_borrowed, Due_date, Returned_date}

Since the closure of {Member_ID, Book_id} includes all attributes in the relation, Member_ID and Book_id together form a candidate key.

Candidate Keys:Borrowed_id, Member_ID, Book_id

Prime Attributes (PAs): Borrowed_id, Member_ID, Book_id

Non-Prime Attributes (NPAs):Date_borrowed, Due_date, Returned_date

(iv) Second Normal Form (2NF):

A relation is in 2NF if it is in 1NF and there are no partial dependencies.

Given the FDs:
- Borrowed_id → Member_ID, Date_borrowed, Due_date, Returned_date
- Member_ID, Book_id → Borrowed_id

Since Borrowed_id is a single attribute and the second FD has a combination of Member_ID and Book_id, there are no partial dependencies. Thus, the relation is in 2NF.

(v) Third Normal Form (3NF):

A relation is in 3NF if it is in 2NF and there are no transitive dependencies.

Given the FDs:
- Borrowed_id → Member_ID, Date_borrowed, Due_date, Returned_date
- Member_ID, Book_id → Borrowed_id

There are no transitive dependencies because there are no non-prime attributes determining other non-prime attributes.

Thus, the relation is in 3NF.

(vi) Boyce-Codd Normal Form (BCNF):

A relation is in BCNF if for every FD (X → Y), X is a superkey.

Given the FDs:
- Borrowed_id → Member_ID, Date_borrowed, Due_date, Returned_date (Borrowed_id is a superkey)
- Member_ID, Book_id → Borrowed_id (Member_ID, Book_id is a superkey)

Since all FDs have a superkey on the left-hand side (LHS), the relation is in BCNF.

Final Normalized Relation:

The final relation "Borrowed_Books" remains:
- Borrowed_Books(Borrowed_id, Book_id, Date_borrowed, Due_date, Returned_date, Member_ID)

It is in 1NF, 2NF, 3NF, and BCNF.

==========.=====

6. Members(Member_id, Fine_id, Borrow_id, Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date)

FDs :

1. Member_id → Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date
2. Fine_id, Member_id → Borrow_id

(i) Irreducible Set of Functional Dependencies:

To find the irreducible set, we check if we can remove any attributes from the right-hand side (RHS) of the FDs without changing the closure.

1. Member_id → Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date
- **Considering FD:**
  {Member_id}+ = {Member_id, Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date, Fine_id, Borrow_id}
- **Without considering the FD:**
  {Member_id}+ = {Member_id, Fine_id, Borrow_id}
- The closure changes without the FD Member_id → Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date, hence it is not redundant.

2. Fine_id, Member_id → Borrow_id
- **Considering FD:**
  {Fine_id, Member_id}+ = {Fine_id, Member_id, Borrow_id, Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date}
- **Without considering the FD:**
  {Fine_id, Member_id}+ = {Fine_id, Member_id}
- The closure changes without the FD Fine_id, Member_id → Borrow_id, hence it is not redundant.

Therefore, the irreducible set of FDs remains:
- Member_id → Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date
- Fine_id, Member_id → Borrow_id

(ii) First Normal Form (1NF):

The relation is in 1NF since all attributes are atomic.

(iii) Candidate Key:

To find the candidate key, we determine the closure of the potential keys.

1. Member_id:
- **Closure set**:
  {Member_id}+ = {Member_id, Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date, Fine_id, Borrow_id}

Since the closure of {Member_id} includes all attributes in the relation, Member_id is a candidate key.

2. Fine_id, Member_id:
- Closure set:
{Fine_id, Member_id}+ = {Fine_id, Member_id, Borrow_id, Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date}

Since the closure of {Fine_id, Member_id} includes all attributes in the relation, Fine_id and Member_id together form a candidate key.

Candidate Keys: Member_id, Fine_id

Prime Attributes (PAs): Member_id, Fine_id

Non-Prime Attributes (NPAs): Borrow_id, Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date

(iv) Second Normal Form (2NF):

A relation is in 2NF if it is in 1NF and there are no partial dependencies.

Given the FDs:
- Member_id → Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date
- Fine_id, Member_id → Borrow_id

Since Member_id is a single attribute and the second FD has a combination of Fine_id and Member_id, there are no partial dependencies. Thus, the relation is in 2NF.

(v) Third Normal Form (3NF):

A relation is in 3NF if it is in 2NF and there are no transitive dependencies.

Given the FDs:
- Member_id → Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date
- Fine_id, Member_id → Borrow_id

There are no transitive dependencies because there are no non-prime attributes determining other non-prime attributes.

Thus, the relation is in 3NF.

(vi) Boyce-Codd Normal Form (BCNF):

A relation is in BCNF if for every FD (X → Y), X is a superkey.

Given the FDs:
- Member_id → Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date (Member_id is a superkey)
- Fine_id, Member_id → Borrow_id (Fine_id, Member_id is a superkey)

Since all FDs have a superkey on the left-hand side (LHS), the relation is in BCNF.

Final Normalized Relation:

The final relation "Members" remains:
- Members(Member_id, Fine_id, Borrow_id, Member_name, Member_DOB, Contact_phonem, Date_of_registration, expiry_date)

It is in 1NF, 2NF, 3NF, and BCNF.

==========

7. Reservations(Reservation_number, Member_id, Date_of_reservation, Book_id, No_copies)

FDs:
1. Member_id, Book_id → Reservation_number, No_copies
2. Book_id → Reservation_number, Member_id, Date_of_reservation, No_copies

(i) Irreducible Set of Functional Dependencies:

The given FDs appear to be irreducible since they have multiple attributes on the left-hand side (LHS) and determine multiple attributes on the right-hand side (RHS). However, it's always a good practice to verify their irreducibility.

Irreducible FDs:
1. Member_id, Book_id → Reservation_number, No_copies
2. Book_id → Reservation_number, Member_id, Date_of_reservation, No_copies

(ii) First Normal Form (1NF):

The relation is already in 1NF since all attributes are atomic (i.e., indivisible).

(iii) Candidate Key:

To find the candidate key, we determine the closure of the potential keys.

1. Member_id, Book_id:
- Closure set:
{Member_id, Book_id}+ = {Member_id, Book_id, Reservation_number, No_copies, Date_of_reservation}

Since the closure of {Member_id, Book_id} includes all attributes in the relation, {Member_id, Book_id} is a candidate key.

Candidate Key: Member_id, Book_id

Prime Attributes (PAs): Member_id, Book_id

Non-Prime Attributes (NPAs): Reservation_number, Date_of_reservation, No_copies

(iv) Second Normal Form (2NF):

A relation is in 2NF if it is in 1NF and there are no partial dependencies.

Partial Dependencies:
There are no partial dependencies since the candidate key is composed of two attributes.

The relation is already in 2NF.

(v) Third Normal Form (3NF):

A relation is in 3NF if it is in 2NF and there are no transitive dependencies.

Transitive Dependencies:
There are no transitive dependencies since all non-prime attributes are directly dependent on the candidate key.

The relation is already in 3NF.

(vi) Boyce-Codd Normal Form (BCNF):

A relation is in BCNF if for every FD (X → Y), X is a superkey.

Checking for BCNF:
- The FDs involve the candidate key (Member_id, Book_id), so the relation is in BCNF.

The relation is already in BCNF.

Final Normalized Relation:

The relation "Reservations" is in 1NF, 2NF, 3NF, and BCNF.

Final Relation:
- Reservations(Reservation_number, Member_id, Date_of_reservation, Book_id, No_copies)

==========

8. Awards (Award_id, Award_name, Book_id, Author_id,Year_awarded)

Fd's : { Award_ID→ Book_ID,Author_ID,Award_Name,Year
Award_ID,Book_ID→Author_ID }

(i) Irreducible set:

For: Award_ID,Book_ID→ Author_ID
Assume:
Award_ID is extraneous :
Book_ID→-> Author_ID:
{ISBN}+ = {Book_ID→, Author_ID}

Book_ID→is extraneous :
Award_ID-> Author_ID:
{Award_ID }+ = {Award_ID , Author_ID, Book_ID, Award_Name,Year}

Therefore: Book_ID is an extraneous attribute in Award_ID,Book_ID→Author_ID .

Decomposing fd's:
Award_ID-> Author_ID

Final fd: Award_ID→ Book_ID,Author_ID,Award_Name,Year

Award_ID→ Book_ID :

Considering fd:
{Award_ID}+ = {Award_ID, Book_ID,Author_ID,Award_Name,Year }
Without considering the fd:
{Award_ID}+ = {Award_ID,Author_ID,Award_Name,Year }

The closure without considering and with consideration of fd: **Award_ID→ Book_ID, is not the same hence not redundant.**

Award_ID→ Author_ID:

Considering fd:
{Award_ID}+ = {Award_ID, Book_ID,Author_ID,Award_Name,Year }
Without considering the fd:
{Award_ID}+ = {Award_ID, Book_ID,Award_Name,Year }

The closure without considering and with consideration of fd: **Award_ID→ Author_ID is not the same and hence not redundant.**

Award_ID→ Award_Name:

Considering fd:
{Award_ID}+ = {Award_ID, Book_ID,Author_ID,Award_Name,Year }
Without considering the fd:
{Award_ID}+ = {Award_ID, Book_ID,Author_ID,Year }

The closure without considering and with consideration of fd: **Award_ID→ Award_Name , is not the same hence not redundant.**

Award_ID→ Year:

Considering fd:
{Award_ID}+ = {Award_ID, Book_ID,Author_ID,Award_Name,Year }
Without considering the fd:
{Award_ID}+ = {Award_ID, Book_ID,Author_ID,Award_Name }

The closure without considering and with consideration of fd: **Award_ID→ Year, is not the same hence not redundant.**

Irreducible fd's:
Award_ID→ Book_ID,Author_ID,Award_Name,Year

(ii) The relation is in 1NF

(iii) Candidate Key:
Closure set:
{ Award_ID, Book_ID,Author_ID,Award_Name,Year **}**+ = **{** Award_ID, Book_ID,Author_ID,Award_Name,Year **}**

By considering book_id --> Award_ID→ Book_ID,Author_ID,Award_Name,Year
Closure set:
{ Award_ID **}**+ = **{** Award_ID, Book_ID,Author_ID,Award_Name,Year **}**

So the candidate key is : Award_id
Prime Attribute (PA) : Award_id
Non Prime Attribute (NPA) : Book_ID,Author_ID,Award_Name,Year

(iv) 2NF:

If the table should be in 2NF then there should be no partial dependencies in the table.
A partial dependency is when a proper subset of candidate keys determines non-prime attributes.

The FDs are:

Award_ID→ Book_ID,Author_ID,Award_Name,Year

Since the candidate key here doesn't have any proper subset, by default it is in 2NF.

(v) 3NF:

A relation is in 3NF if there exists no transitive dependencies.
A transitive dependency is when non-prime attributes determine some other non-prime attributes.

The FDs are:

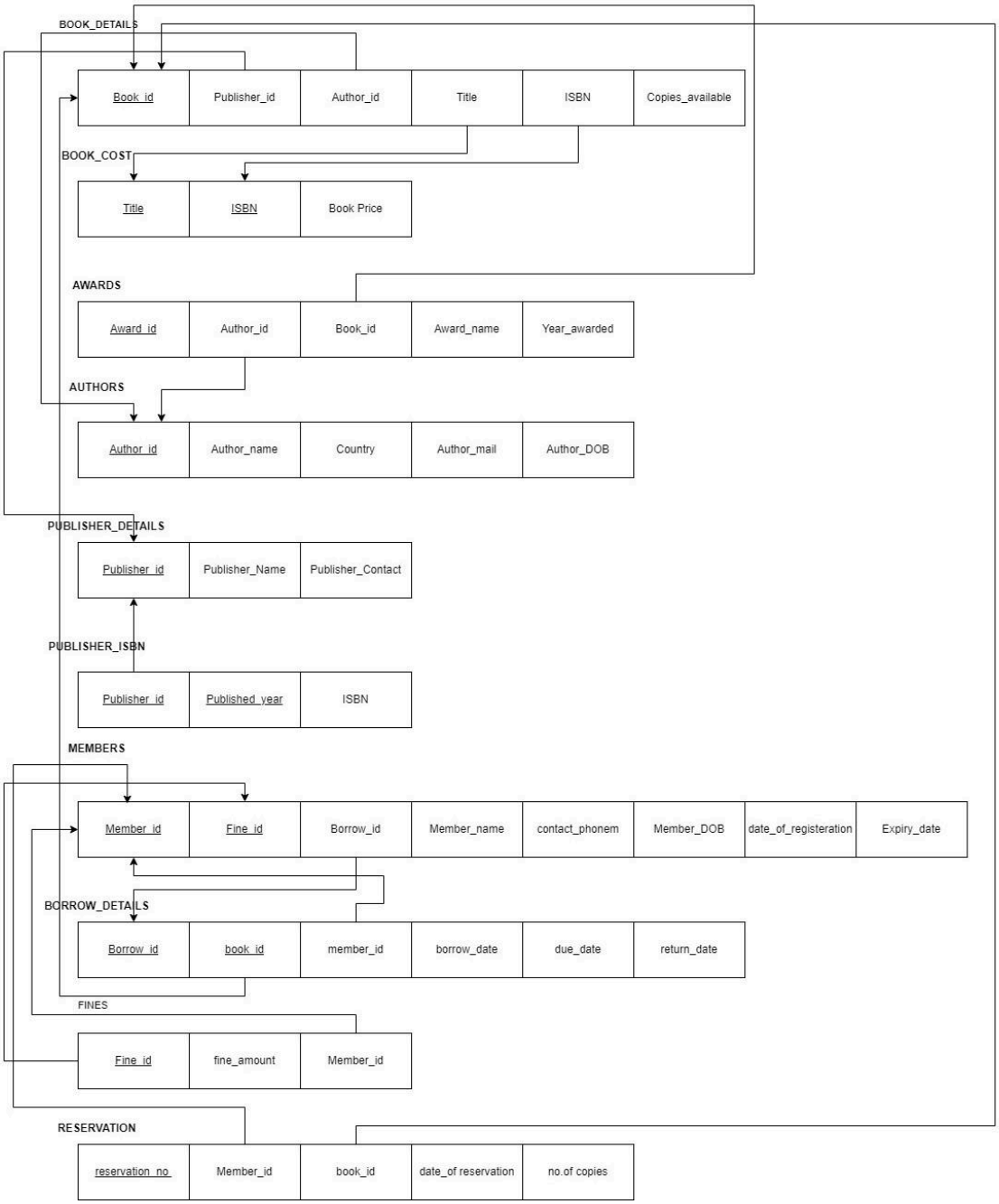
Award_ID→ Book_ID,Author_ID,Award_Name,Year

Here the fd has no Transitive dependency .**Hence is in 3NF.**

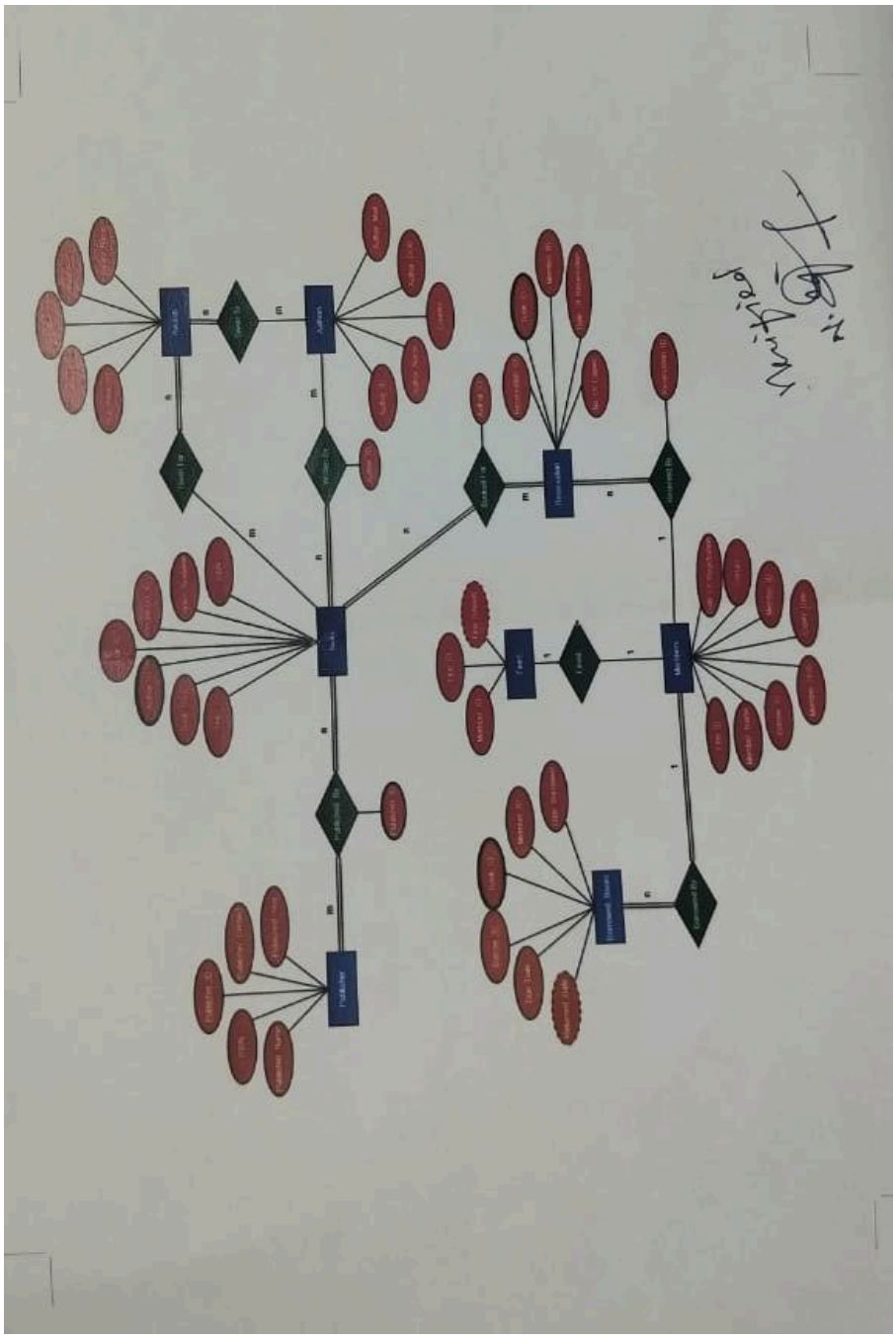
vi)BCNF:
Since all the relations FDs have superkey as their LHS , the relations are in BCNF.
Therefore the final relations for the Book_Awards is: Book_details and Book_cost

==========

SCHEMA AFTER NORMALIZATION:



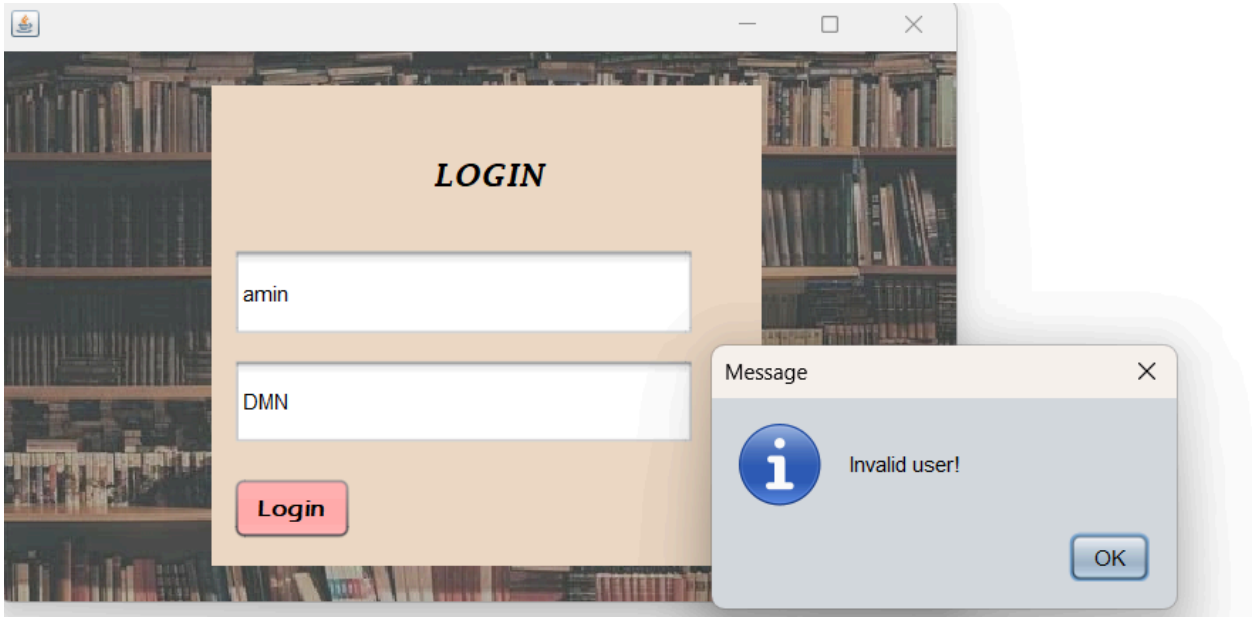
VERIFIED ER DIAGRAM:



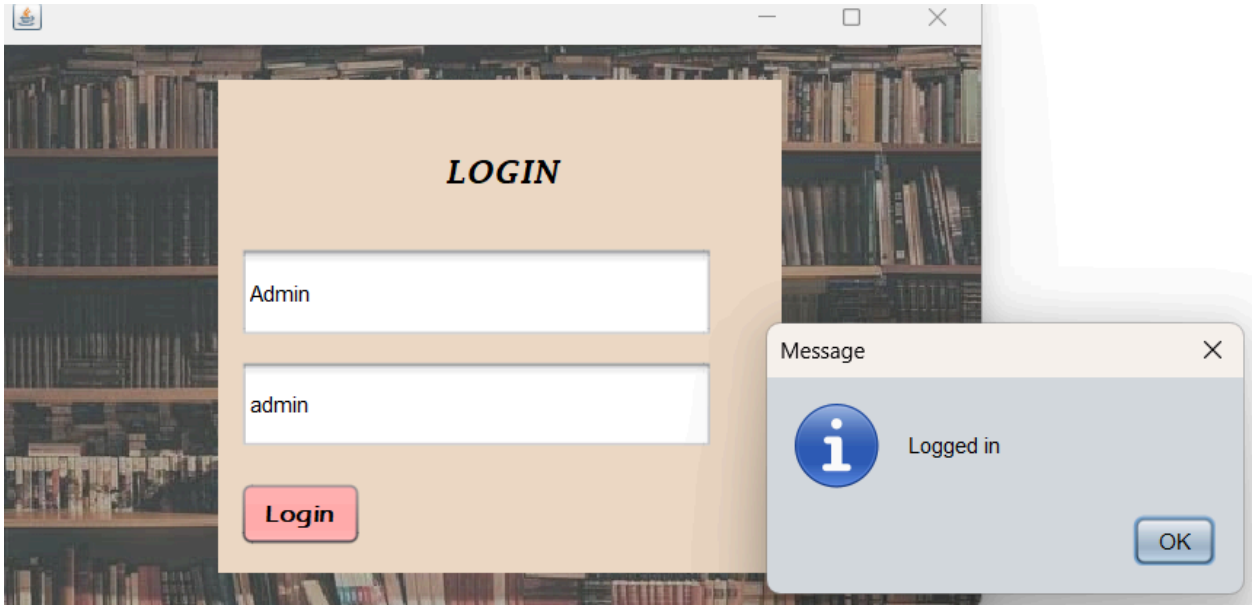
WORKFLOW

1)LOGIN

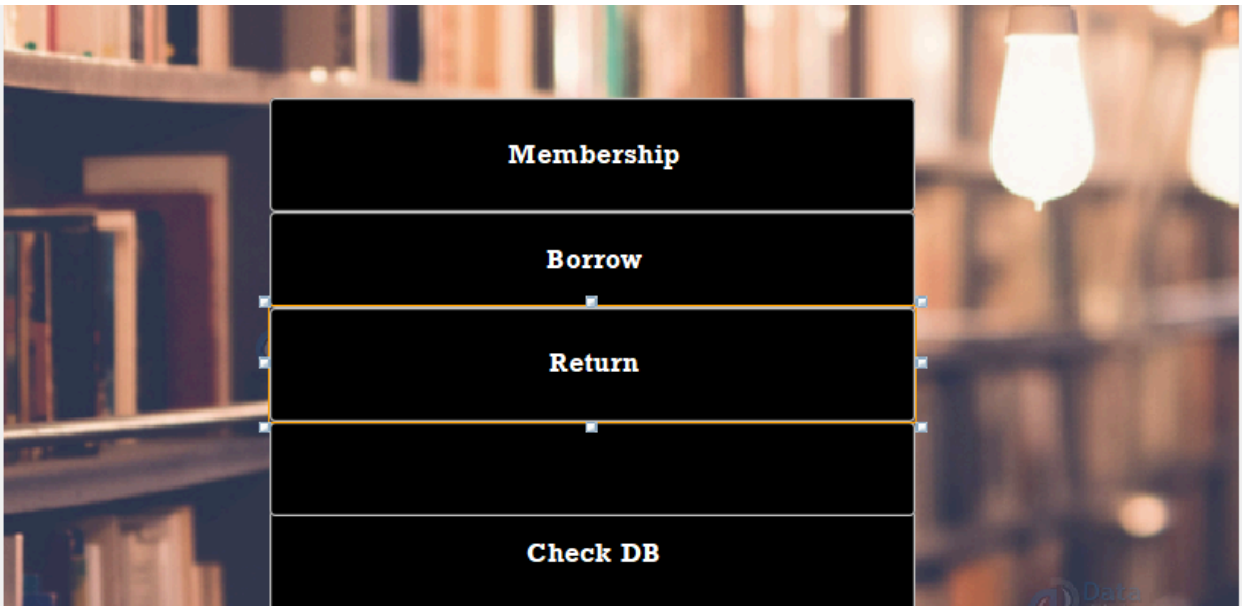
-> Invalid User



-> Valid User

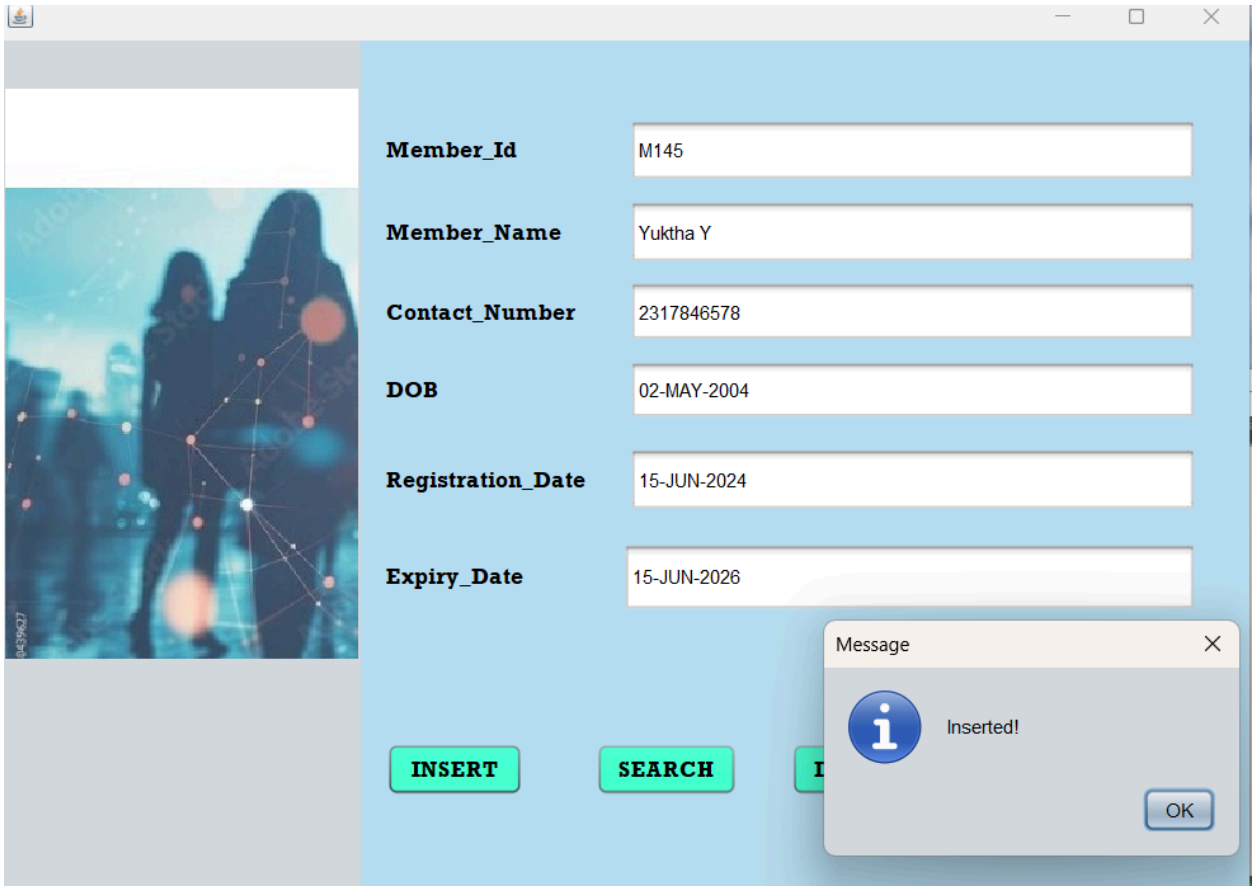


2)NAVIGATION



->MEMBERSHIP PAGE

3)INSERT



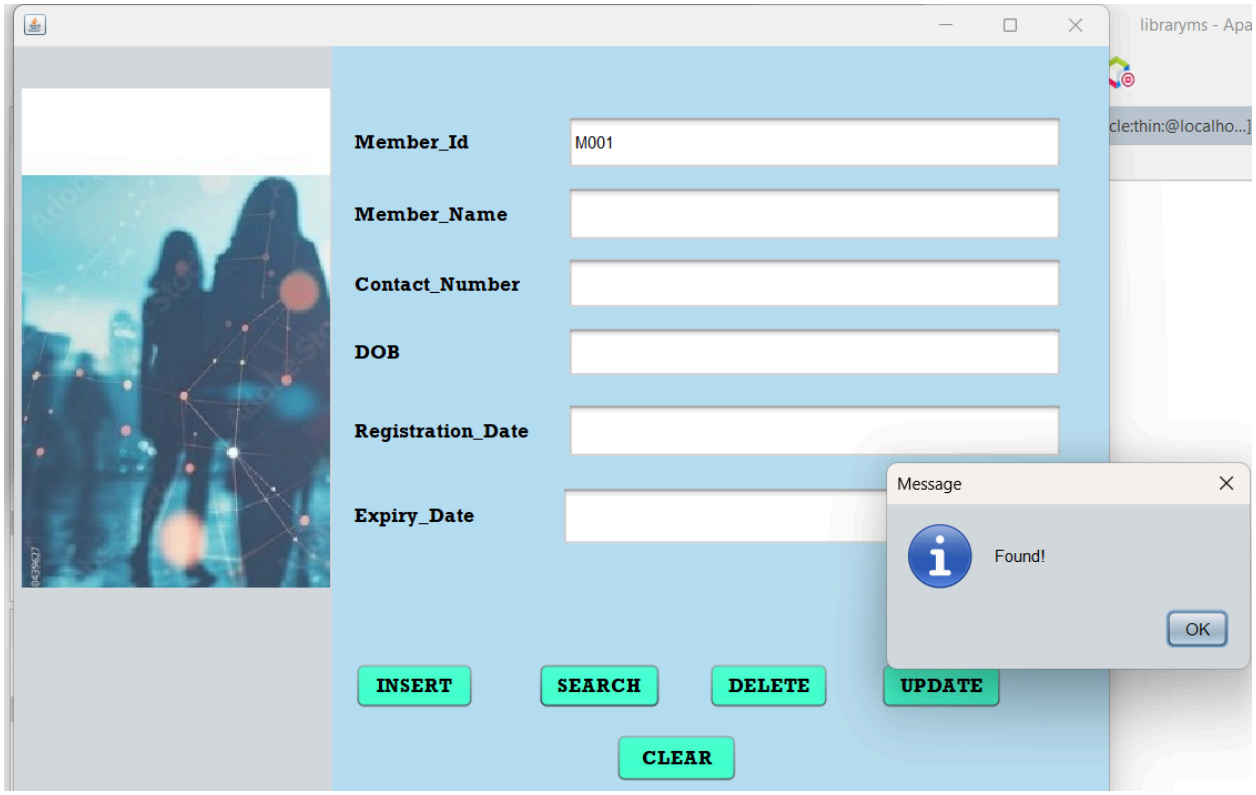

```
SQL> select *from members;

MEMBE FINE_ BORRO
----- ----- -----
MEMBERNAME
--------------------------------------------------------------------------------
MEMBERCONTACT_N MEMBER_DO MEMBER_RE MEMBER_EX
--------------- --------- --------- ---------
M145  FM145 BM145
Yuktha Y
2317846578      02-MAY-04 15-JUN-24 15-JUN-26
```

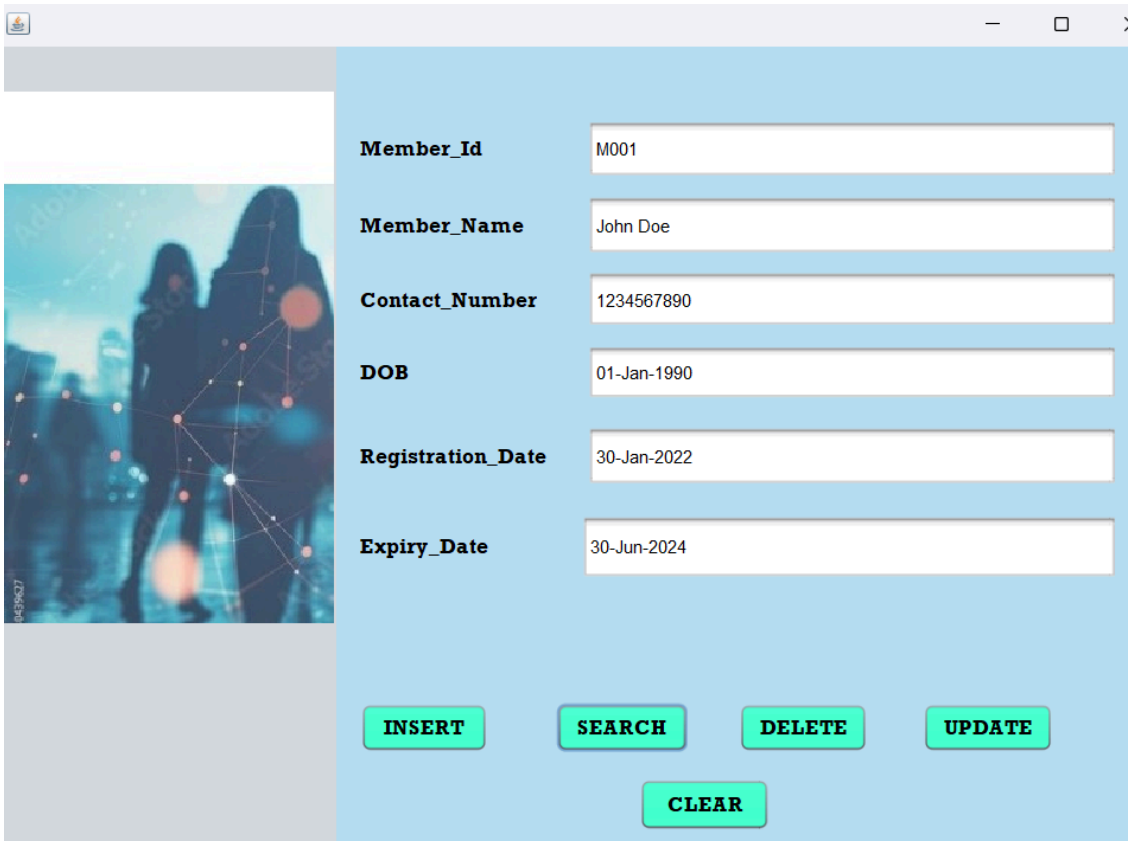
SELECT * FROM SCOTT.MEMBE... ×

Max. rows: 100 | Fetched Rows: 6 | Matching Rows:

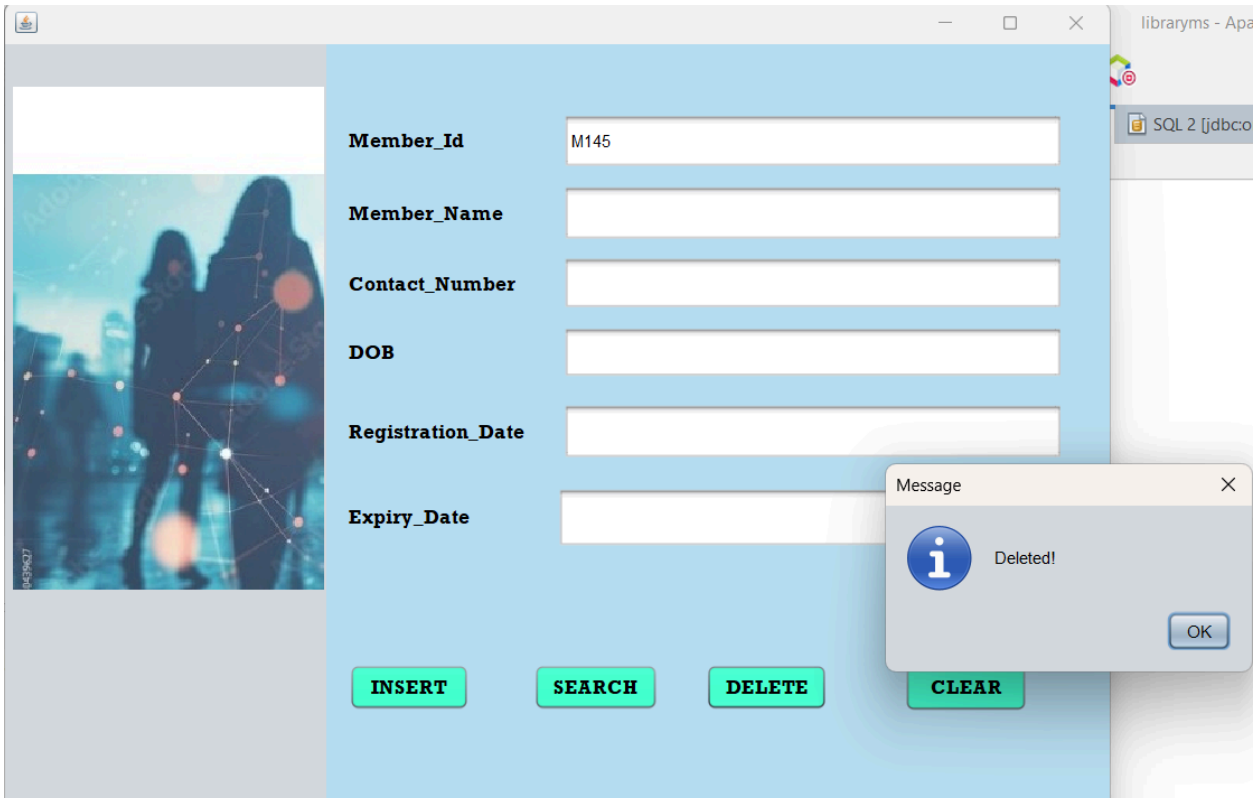
#	MEMBER_ID	FINE_ID	BORROW_ID	MEMBERNAME	MEMBERCONTACT_NUMBER	MEMBER_DOB	MEMBER_REGISTRATION	MEMBER_EXPIRY_DATE
1	M009	FM009	BM009	Yuktha	4769087652	2004-05-09 00:0...	2024-05-09 00:00:00.000	2026-05-09 00:00:00.000
2	M001	FM001	BM001	John Doe	1234567890	1990-01-01 00:0...	2022-01-30 00:00:00.000	2024-06-30 00:00:00.000
3	M002	FM002	BM002	Jane Smith	9876543210	1985-05-15 00:0...	2021-07-30 00:00:00.000	2023-12-31 00:00:00.000
4	M003	FM003	BM003	Michael Johnson	5551234567	1995-09-20 00:0...	2023-01-15 00:00:00.000	2025-01-15 00:00:00.000
5	M004	FM004	BM004	Emily Brown	3339876543	1988-03-10 00:0...	2020-08-31 00:00:00.000	2022-08-31 00:00:00.000
6	M005	FM005	BM005	David Lee	7775554444	1992-11-28 00:0...	2021-11-30 00:00:00.000	2023-11-30 00:00:00.000

4)SEARCH


Member_Id
M001
Member_Name
Contact_Number
DOB
Registration_Date
Expiry_Date
Message
Found!
OK
INSERT
SEARCH
DELETE
UPDATE
CLEAR


Member_Id
M001
Member_Name
John Doe
Contact_Number
1234567890
DOB
01-Jan-1990
Registration_Date
30-Jan-2022
Expiry_Date
30-Jun-2024
INSERT
SEARCH
DELETE
UPDATE
CLEAR

5) DELETE



SELECT * FROM SCOTT.MEMBE... ×

Max. rows: 100 Fetched Rows: 5 Matching Rows:

#	MEMBER_ID	FINE_ID	BORROW_ID	MEMBERNAME	MEMBERCONTACT_NUMBER	MEMBER_DOB	MEMBER_REGISTRATION	MEMBER_EXPIRY_DATE
1	M001	FM001	BM001	John Doe	1234567890	1990-01-01 00:0...	2022-01-30 00:00:00.000	2024-06-30 00:00:00.000
2	M002	FM002	BM002	Jane Smith	9876543210	1985-05-15 00:0...	2021-07-30 00:00:00.000	2023-12-31 00:00:00.000
3	M003	FM003	BM003	Michael Johnson	5551234567	1995-09-20 00:0...	2023-01-15 00:00:00.000	2025-01-15 00:00:00.000
4	M004	FM004	BM004	Emily Brown	3339876543	1988-03-10 00:0...	2020-08-31 00:00:00.000	2022-08-31 00:00:00.000
5	M005	FM005	BM005	David Lee	7775554444	1992-11-28 00:0...	2021-11-30 00:00:00.000	2023-11-30 00:00:00.000

6) UPDATE

->Before Update

SELECT * FROM SCOTT.MEMBE... ×

Max. rows: 100 Fetched Rows: 7 Matching Rows:

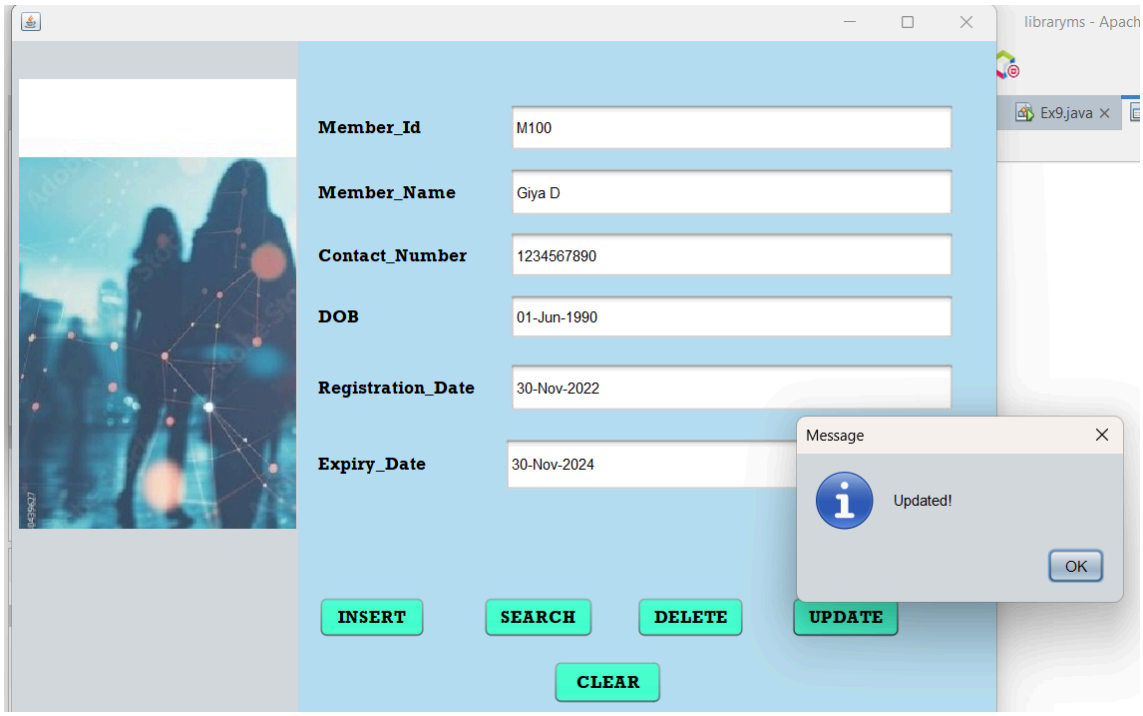
#	MEMBER_ID	FINE_ID	BORROW_ID	MEMBERNAME	MEMBERCONTACT_NUMBER	MEMBER_DOB	MEMBER_REGISTRATION	MEMBER_EXPIRY_DATE
1	M009	FM009	BM009	Yuktha	4769087652	2004-05-09 00:0...	2024-05-09 00:00:00.000	2026-05-09 00:00:00.000
2	M100	FM100	BM100	Hiya Doe	1234567890	1990-06-01 00:0...	2022-11-30 00:00:00.000	2024-11-30 00:00:00.000
3	M001	FM001	BM001	John Doe	1234567890	1990-01-01 00:0...	2022-01-30 00:00:00.000	2024-06-30 00:00:00.000
4	M002	FM002	BM002	Jane Smith	9876543210	1985-05-15 00:0...	2021-07-30 00:00:00.000	2023-12-31 00:00:00.000
5	M003	FM003	BM003	Michael Johnson	5551234567	1995-09-20 00:0...	2023-01-15 00:00:00.000	2025-01-15 00:00:00.000
6	M004	FM004	BM004	Emily Brown	3339876543	1988-03-10 00:0...	2020-08-31 00:00:00.000	2022-08-31 00:00:00.000
7	M005	FM005	BM005	David Lee	7775554444	1992-11-28 00:0...	2021-11-30 00:00:00.000	2023-11-30 00:00:00.000

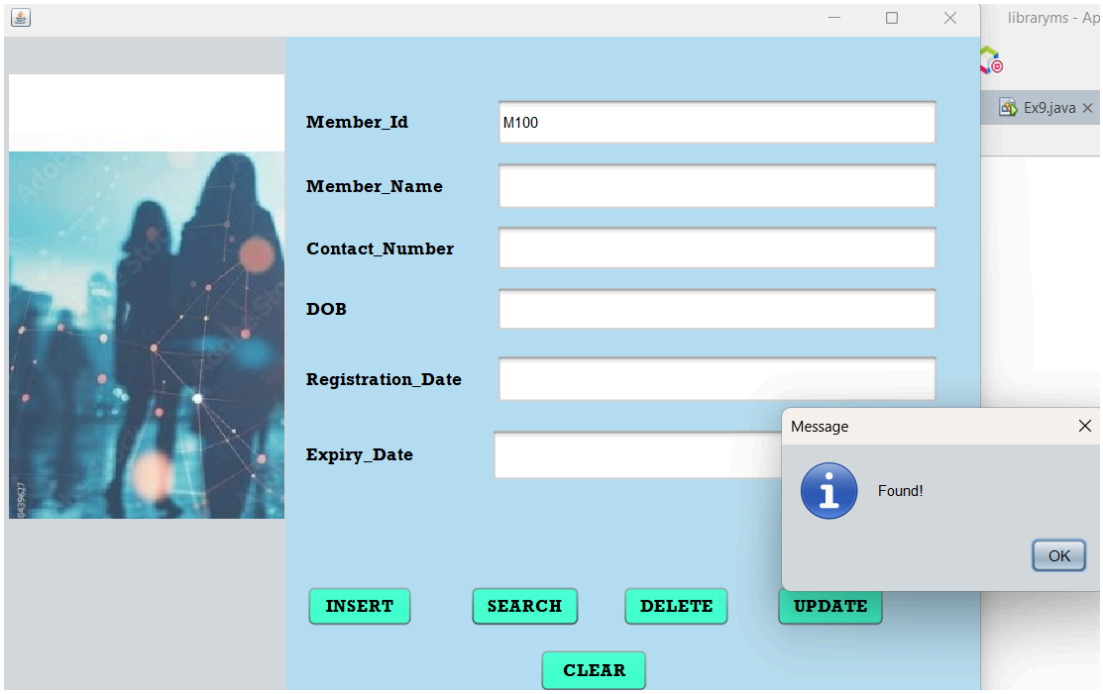
->After Update

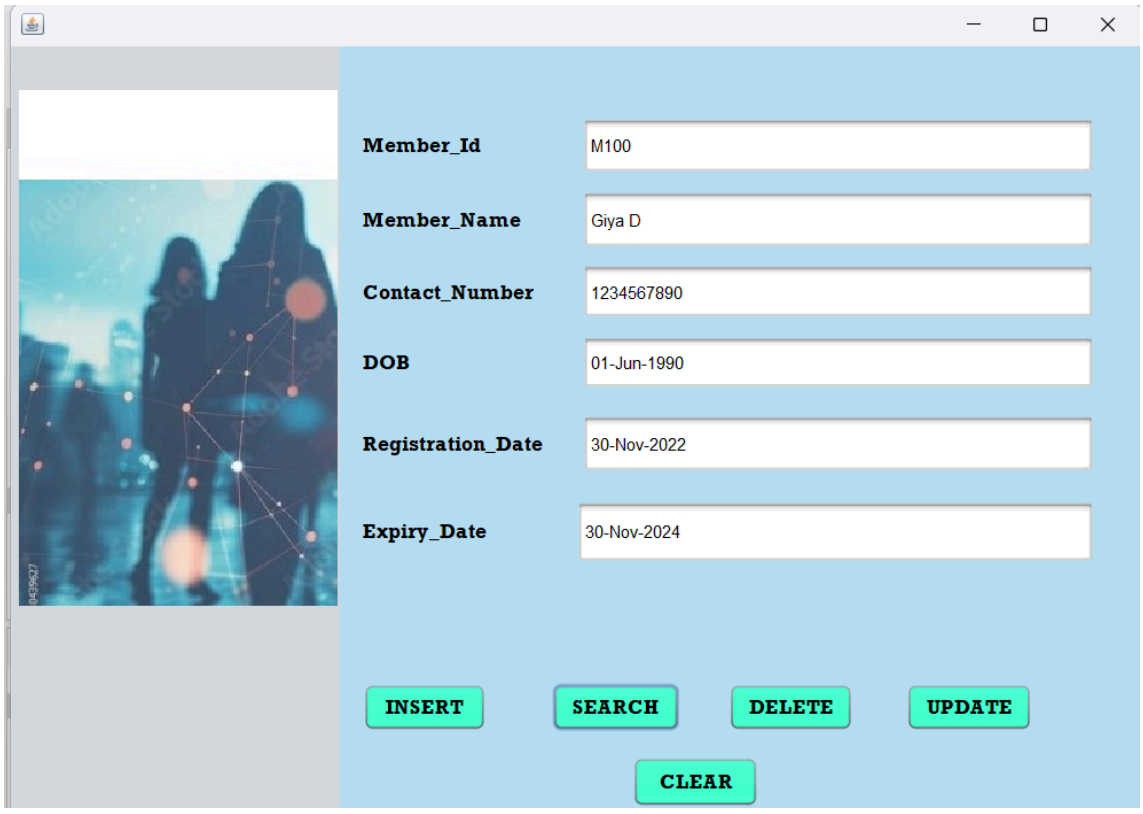
SELECT * FROM SCOTT.MEMBE... ×

Max. rows: 100 | Fetched Rows: 7 | Matching Rows:

#	MEMBER_ID	FINE_ID	BORROW_ID	MEMBERNAME	MEMBERCONTACT_NUMBER	MEMBER_DOB	MEMBER_REGISTRATION	MEMBER_EXPIRY_DATE
1	M009	FM009	BM009	Yuktha	4769087652	2004-05-09 00:0...	2024-05-09 00:00:00.000	2026-05-09 00:00:00.000
2	M100	FM100	BM100	Giya D	1234567890	1990-06-01 00:0...	2022-11-30 00:00:00.000	2024-11-30 00:00:00.000
3	M001	FM001	BM001	John Doe	1234567890	1990-01-01 00:0...	2022-01-30 00:00:00.000	2024-06-30 00:00:00.000
4	M002	FM002	BM002	Jane Smith	9876543210	1985-05-15 00:0...	2021-07-30 00:00:00.000	2023-12-31 00:00:00.000
5	M003	FM003	BM003	Michael Johnson	5551234567	1995-09-20 00:0...	2023-01-15 00:00:00.000	2025-01-15 00:00:00.000
6	M004	FM004	BM004	Emily Brown	3339876543	1988-03-10 00:0...	2020-08-31 00:00:00.000	2022-08-31 00:00:00.000
7	M005	FM005	BM005	David Lee	7775554444	1992-11-28 00:0...	2021-11-30 00:00:00.000	2023-11-30 00:00:00.000

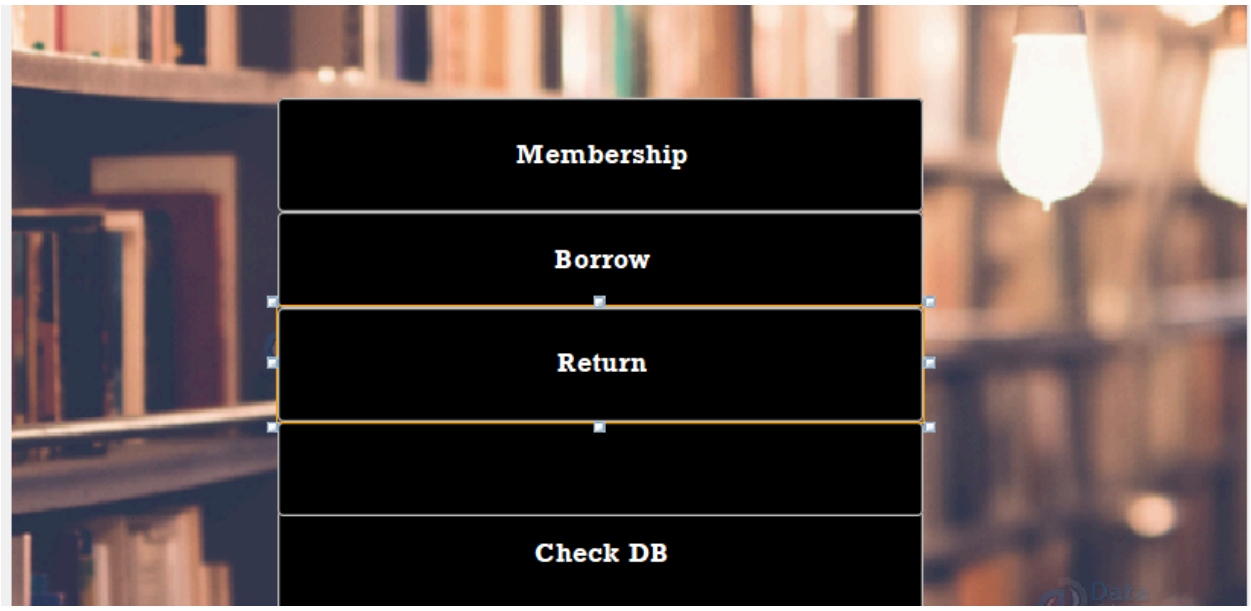



Member_Id
M100
Member_Name
Contact_Number
DOB
Registration_Date
Expiry_Date
Message
Found!
OK
INSERT
SEARCH
DELETE
UPDATE
CLEAR


Member_Id
M100
Member_Name
Giya D
Contact_Number
1234567890
DOB
01-Jun-1990
Registration_Date
30-Nov-2022
Expiry_Date
30-Nov-2024
INSERT
SEARCH
DELETE
UPDATE
CLEAR

7) CLEAR

->Navigating to Borrow Page



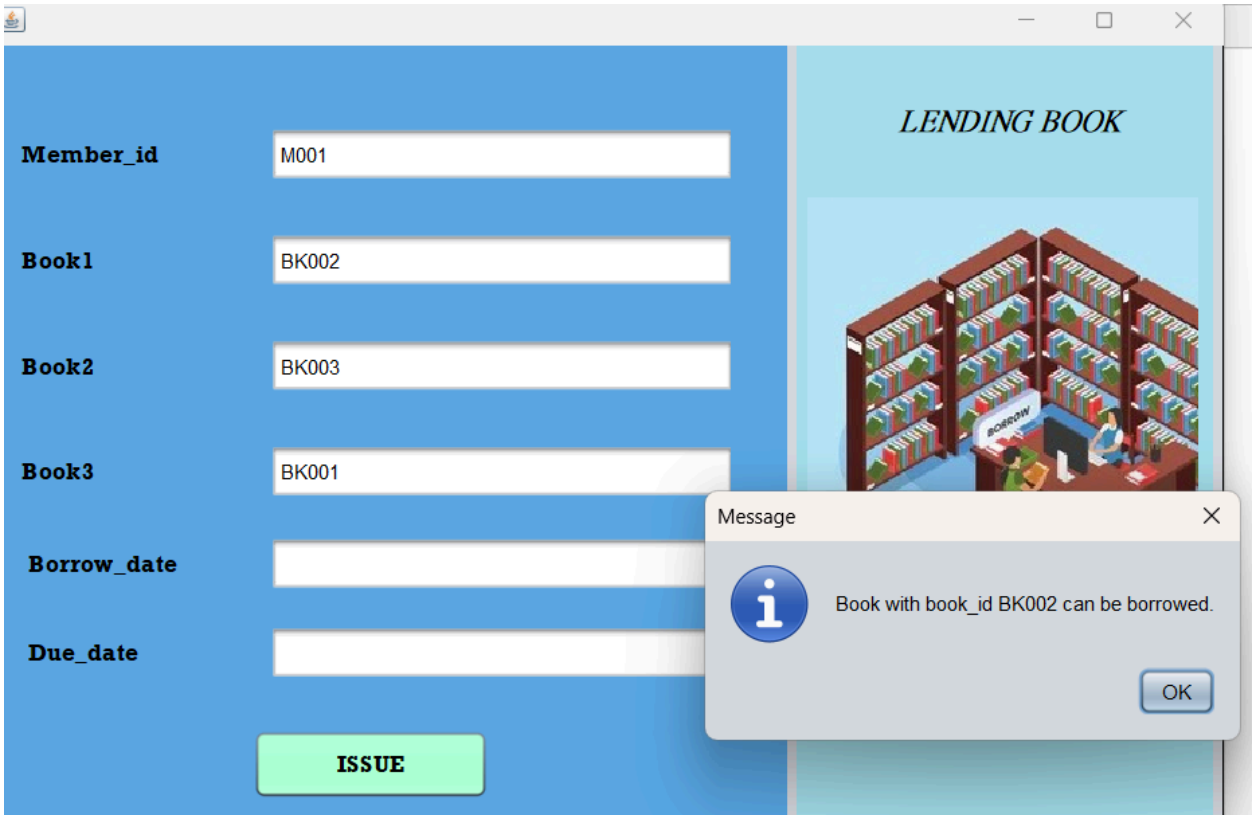
Before Borrowing:

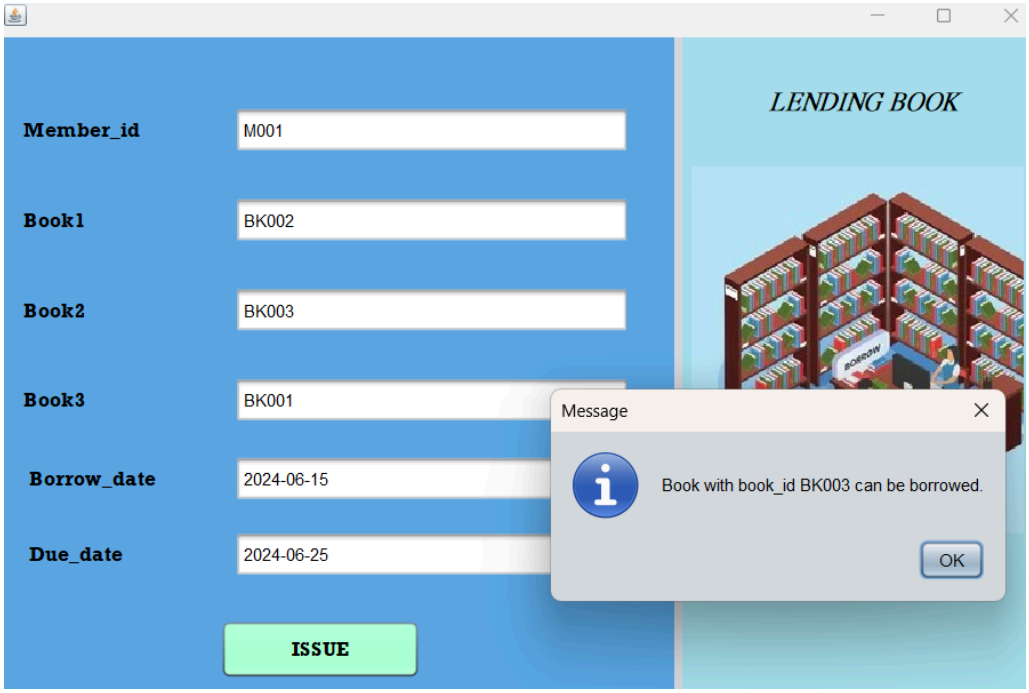
SELECT * FROM SCOTT.BORRO... ×

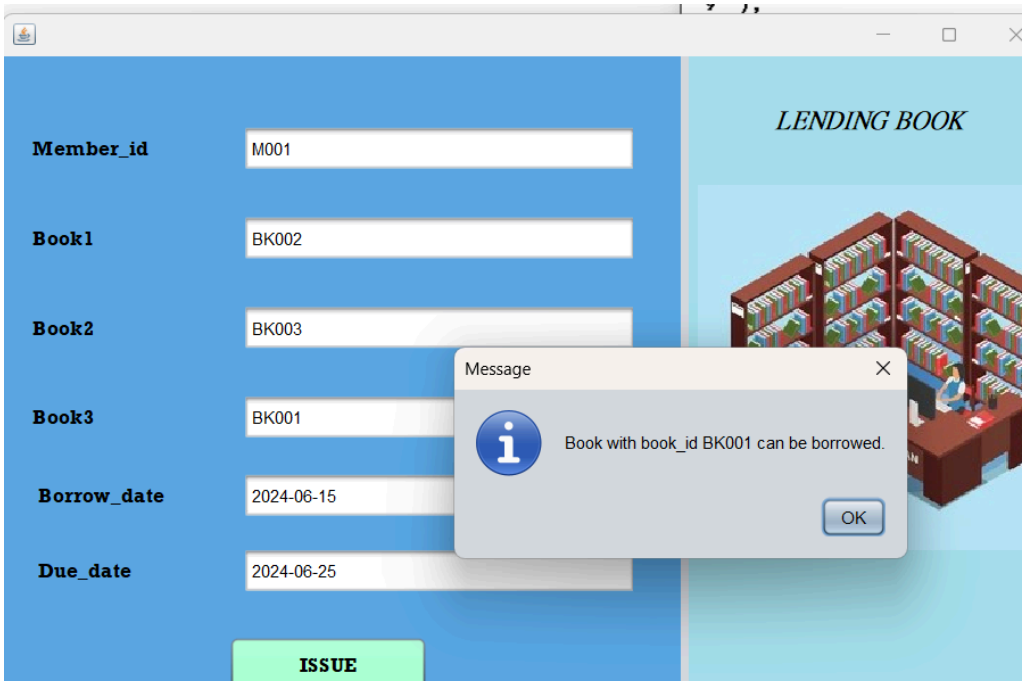
Max. rows: 100 Fetched Rows: 5

#	BORROW_ID	BOOK_ID	BORROW_DATE	DUE_DATE	RETURN_DATE	MEMBER_ID
1	BM001	BK001	2024-06-01 00:00:0..	2024-06-10 ...	2024-06-10 00:00:0..	M001
2	BM002	BK002	2024-05-20 00:00:0..	2024-05-30 ...	2024-06-05 00:00:0..	M002
3	BM004	BK004	2024-06-10 00:00:0..	2024-06-20 ...	2024-06-20 00:00:0..	M004
4	BM005	BK005	2024-06-05 00:00:0..	2024-06-15 ...	2024-06-15 00:00:0..	M005
5	BM003	BK003	2024-06-15 00:00:0..	2024-06-25 ...	2024-06-26 00:00:0..	M003

Borrowing:




LENDING BOOK
Member_id
M001
Book1
BK002
Book2
BK003
Book3
BK001
Borrow_date
2024-06-15
Due_date
2024-06-25
ISSUE
Message
Book with book_id BK003 can be borrowed.
OK


LENDING BOOK
Member_id
M001
Book1
BK002
Book2
BK003
Book3
BK001
Borrow_date
2024-06-15
Due_date
2024-06-25
ISSUE
Message
Book with book_id BK001 can be borrowed.
OK

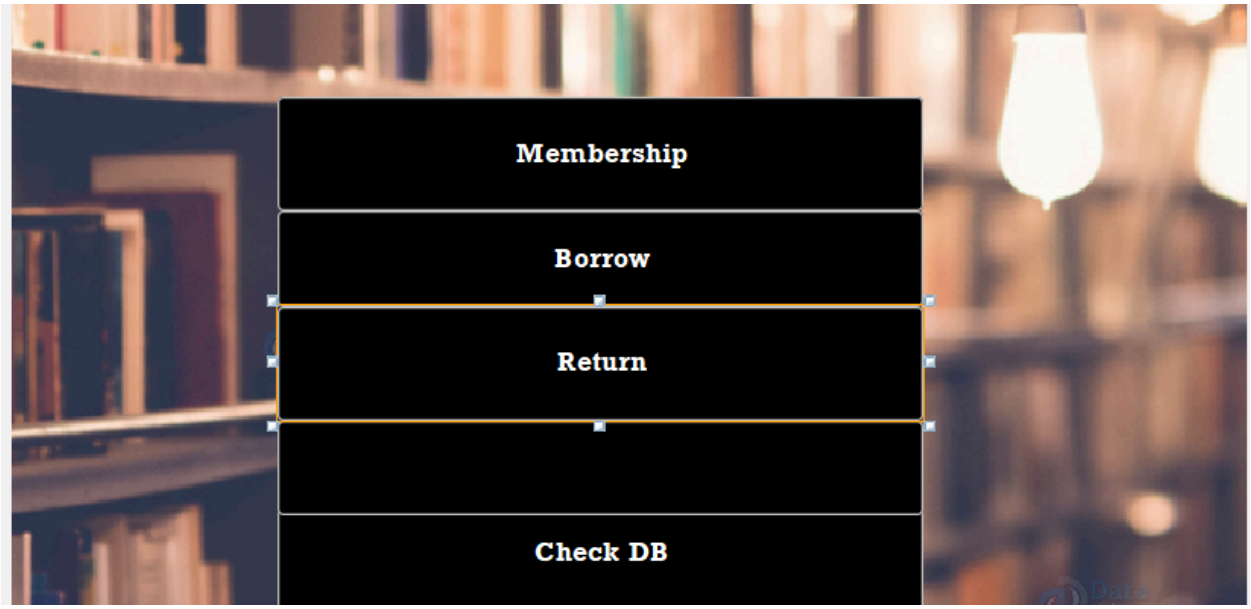
After Borrowing

SELECT * FROM SCOTT.BORRO... ×

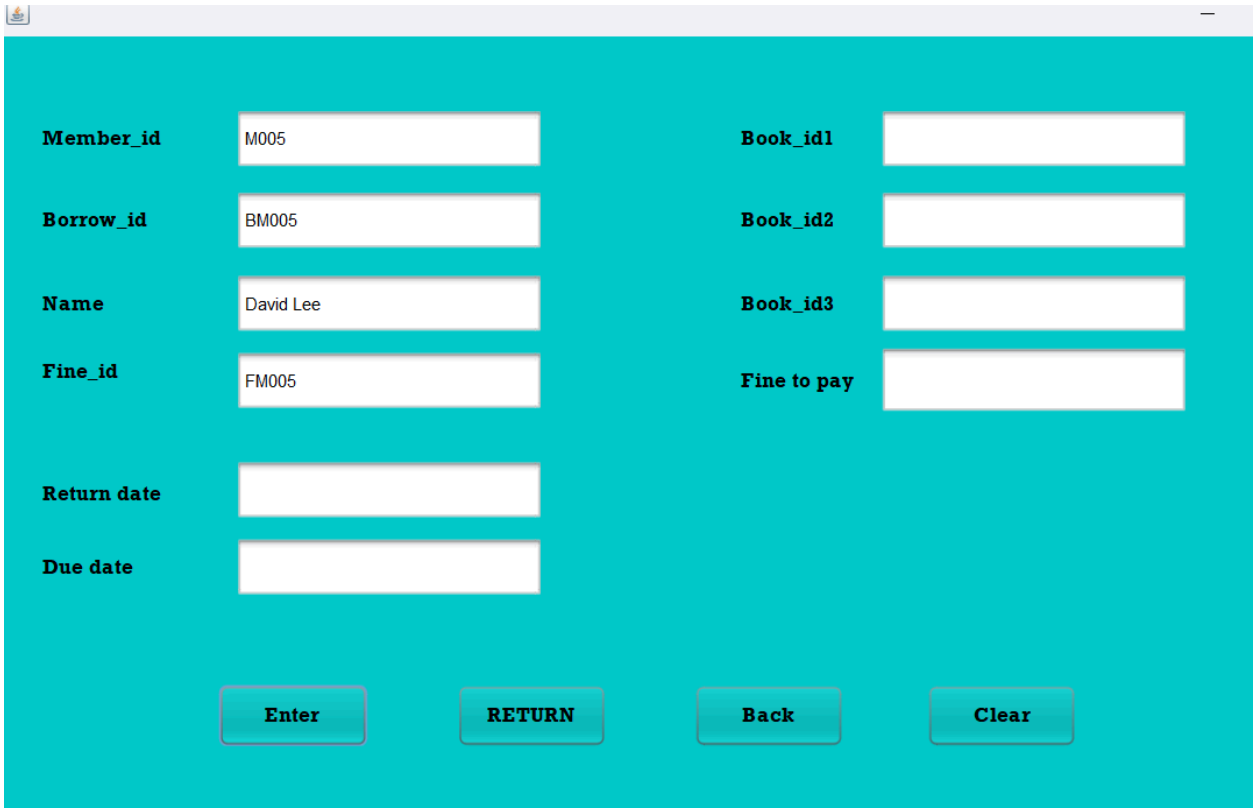
Max. rows: 100 Fetched Rows: 8

#	BORROW_ID	BOOK_ID	BORROW_DATE	DUE_DATE	RETURN_DATE	MEMBER_ID
1	BM001	BK002	2024-06-15 00:00:0..	<NULL>	2024-06-25 00:00:0..	M001
2	BM002	BK003	2024-06-15 00:00:0..	<NULL>	2024-06-25 00:00:0..	M001
3	BM003	BK001	2024-06-15 00:00:0..	<NULL>	2024-06-25 00:00:0..	M001
4	BM001	BK001	2024-06-01 00:00:0..	2024-06-10 ...	2024-06-10 00:00:0..	M001
5	BM002	BK002	2024-05-20 00:00:0..	2024-05-30 ...	2024-06-05 00:00:0..	M002
6	BM004	BK004	2024-06-10 00:00:0..	2024-06-20 ...	2024-06-20 00:00:0..	M004
7	BM005	BK005	2024-06-05 00:00:0..	2024-06-15 ...	2024-06-15 00:00:0..	M005
8	BM003	BK003	2024-06-15 00:00:0..	2024-06-25 ...	2024-06-26 00:00:0..	M003

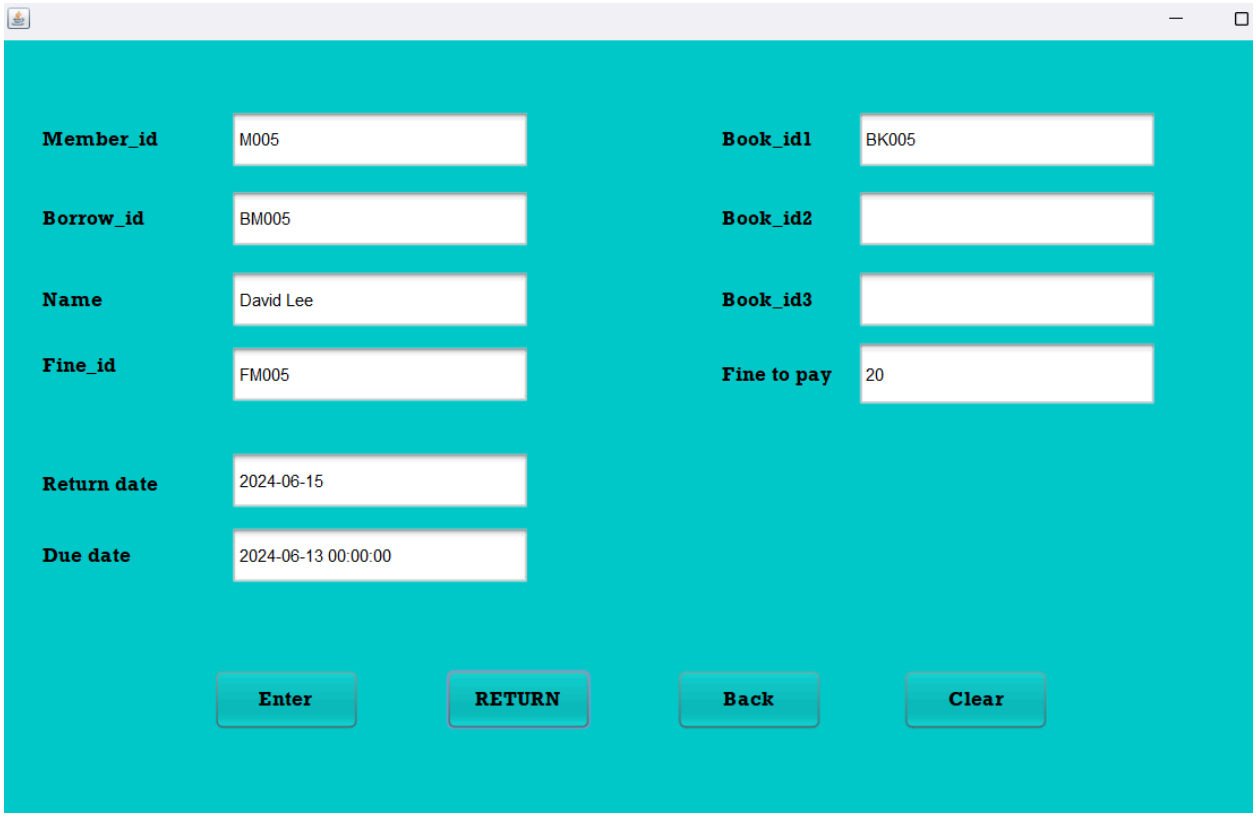
->Navigating to Check DB



RETURN:

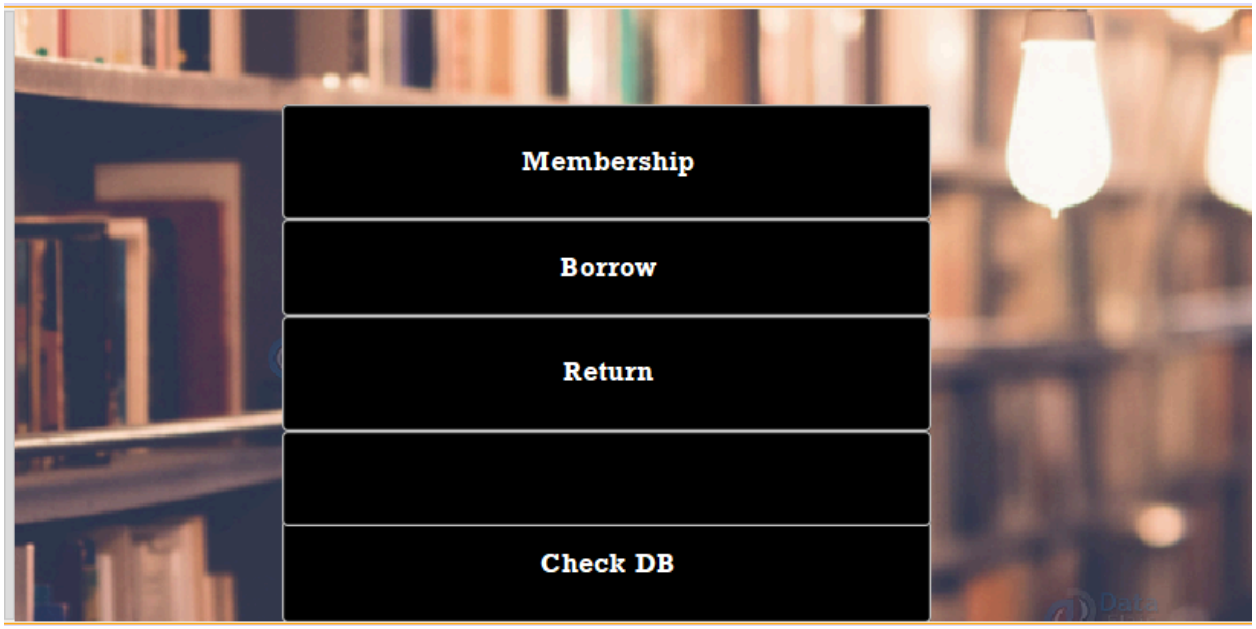


After Clicking enter the Name, fine id and the borrowed book_id s and their borrow date is displayed.

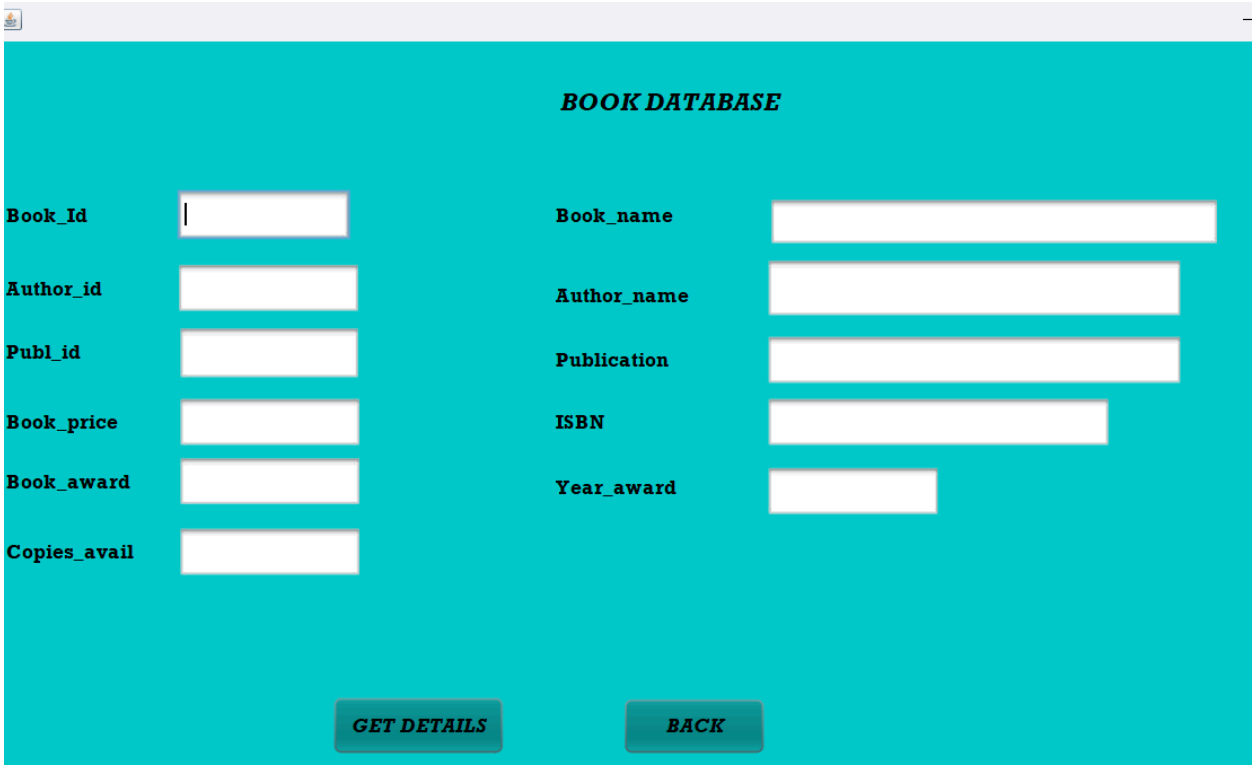


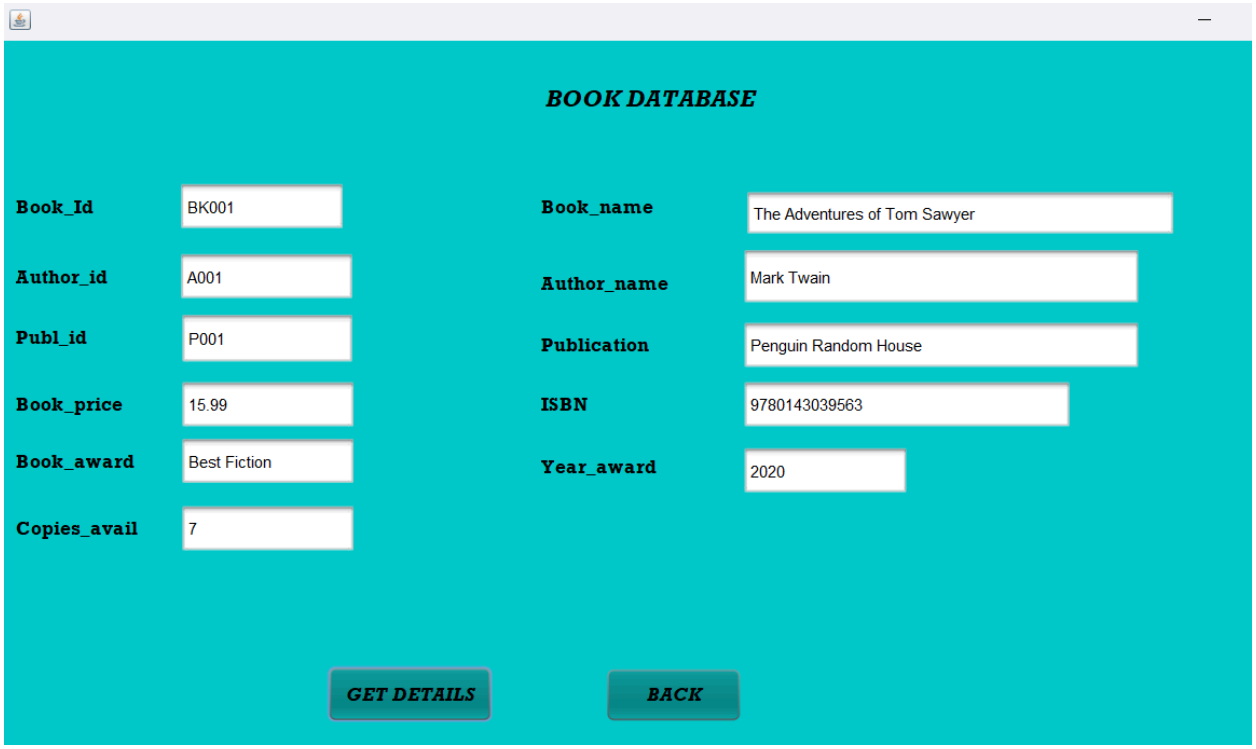
Now after entering the return date and clicking on the return button the fine is calculated and displayed

NAVIGATION:



Checking Book Database




BOOK DATABASE
Book_Id
BK001
Author_id
A001
Publ_id
P001
Book_price
15.99
Book_award
Best Fiction
Copies_avail
7
Book_name
The Adventures of Tom Sawyer
Author_name
Mark Twain
Publication
Penguin Random House
ISBN
9780143039563
Year_award
2020
GET DETAILS
BACK

TRIGGERS OR PROCEDURES USED:

1) REM: trigger for creating borrow id and fine id along with memberid

```
CREATE OR REPLACE TRIGGER trg_create_borrowid_fineid
BEFORE INSERT ON members  -- Change to BEFORE INSERT for setting values before insert
FOR EACH ROW
BEGIN
   -- Generate borrow_id and fine_id based on member_id
   :NEW.borrow_id := 'b' || :NEW.member_id;
   :NEW.fine_id := 'f' || :NEW.member_id;
END;
/
```

2) REM: trigger for borrowing books

```
CREATE OR REPLACE TRIGGER trg_borrow_book
BEFORE INSERT ON borrows
FOR EACH ROW
DECLARE
   v_copies NUMBER;
BEGIN
   -- Check if book_id has available copies

   -- If copies are available, insert into borrows and update copies_available
   IF v_copies > 0 THEN
      :NEW.borrow_date := SYSDATE;
      :NEW.return_date := SYSDATE + 10;

   ELSE
      -- Display a message that book is not available
      DBMS_OUTPUT.PUT_LINE('Book with book_id ' || :NEW.book_id || ' is not available now.');
   END IF;
END;
/
```

3) REM: trigger to update the fines

```
CREATE OR REPLACE TRIGGER UPDATE_FINE_AMOUNT_TRIGGER
AFTER UPDATE OF return_date ON borrow_details
```

```
FOR EACH ROW
DECLARE
   v_due_date   DATE;
   v_book_price NUMBER;
   v_days_late  NUMBER;
   fine_amount_n NUMBER;
BEGIN
   -- Retrieve the due date and book price for the corresponding book_id
   SELECT  book_price
   INTO v_book_price
   FROM book_details
   WHERE book_id = :NEW.book_id;

   SELECT due_date
   INTO v_due_date
   FROM borrow_details
   WHERE borrow_id = :NEW.borrow_id;
   -- Calculate the number of days late if return_date is greater than due_date
   IF :NEW.return_date > v_due_date THEN
      v_days_late := TRUNC(:NEW.return_date - v_due_date); -- Calculate whole days
   ELSE
      v_days_late := 0; -- No fine if return_date is not greater than due_date
   END IF;

   -- Calculate the fine amount based on days late and book price
   fine_amount_n := v_days_late * v_book_price; -- Use := for assignment

   -- Update the fine table with the calculated fine amount
   UPDATE fines
   SET fine_amount = fine_amount_n
   WHERE fine_id = :NEW.borrow_id;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      NULL; -- Handle exception if book_id not found (optional)
   WHEN OTHERS THEN
      NULL; -- Handle other exceptions (optional)
END;
/
```

4) REM: to update the due date to borrow date +14

```
CREATE OR REPLACE TRIGGER insert_borrow_details_trigger
AFTER INSERT ON borrow_details
```

```
FOR EACH ROW
DECLARE
   due_date DATE;
BEGIN
   -- Calculate due_date as sysdate + 14 days
   due_date := SYSDATE + 14;

   -- to Update borrow_details table with calculated due_date
   UPDATE borrow_details
   SET due_date = due_date
   WHERE borrow_id = :NEW.borrow_id;

   -- Reduce no_of_copies in book_details table for the corresponding book_id
   UPDATE book_details
   SET copies_available = copies_available - 1
   WHERE book_id = :NEW.book_id;
END;
/
```

Operations that are unique to our project

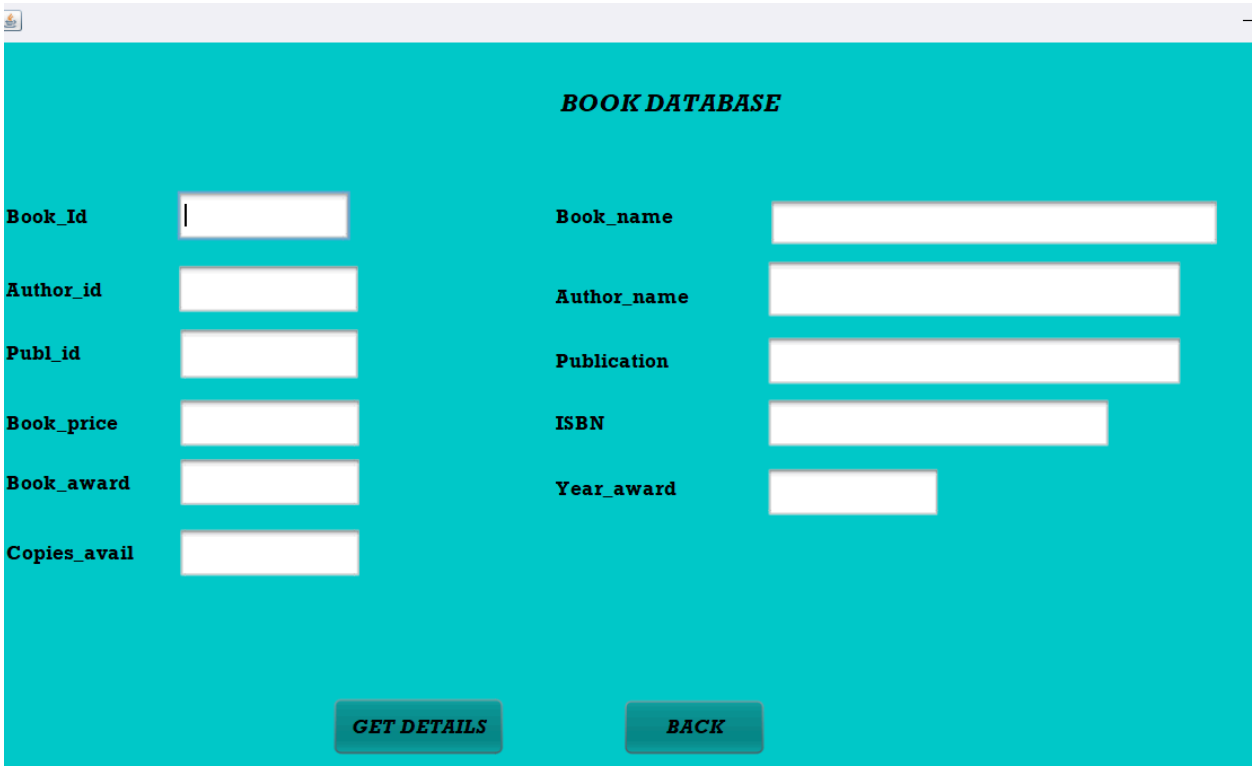
1) Book database:

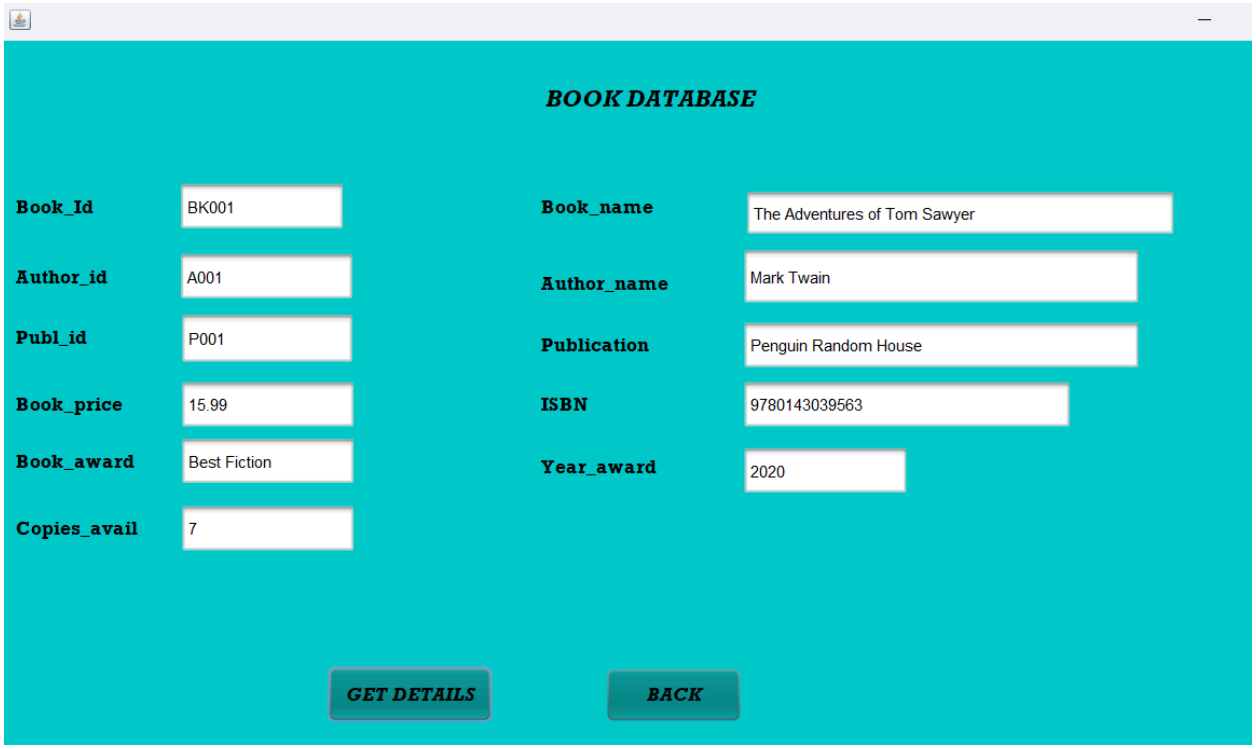
CODE:

```
private void get_detailsActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String bookId = book_id.getText().trim();

    try {
        //con = DatabaseConnection.getConnection();
        con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl",
"scott", "tiger");

        // Query to fetch book details
        String query = "SELECT * FROM book_details_ WHERE book_id = ?";
        ps = con.prepareStatement(query);
        ps.setString(1, bookId);
        rs = ps.executeQuery();

        if (rs.next()) {
            // Display book details
            book_name.setText(rs.getString("book_name"));
            author_id.setText(rs.getString("author_id"));
            publisher_id.setText(rs.getString("publisher_id"));
            book_price.setText(rs.getString("book_price"));
            book_awards.setText(rs.getString("book_awards"));
            COPIES_AVAILABLE.setText(rs.getString("COPIES_AVAILABLE"));
            year_awarded.setText(rs.getString("year_awarded"));
            isbn.setText(rs.getString("isbn"));

            // Fetch publisher details based on publisher_id
            String publisherQuery = "SELECT publisher_name FROM publisher_details
WHERE publisher_id = ?";
            ps = con.prepareStatement(publisherQuery);
            ps.setString(1, rs.getString("publisher_id"));
            ResultSet publisherRs = ps.executeQuery();

            if (publisherRs.next()) {
                publisher_name.setText(publisherRs.getString("publisher_name"));
```

```
            }

            publisherRs.close();
        } else {
            JOptionPane.showMessageDialog(this, "Book ID not found!");
        }
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this, "Database error: " + ex.getMessage());
    } finally {
        try {
            if (rs != null) rs.close();
            if (ps != null) ps.close();
            if (con != null) con.close();
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Error closing database connection: " +
ex.getMessage());
        }
    }
    }

    private void backActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        this.dispose();

    // Open Dashboard window
    Database dashboard = new Database();
    dashboard.setVisible(true);
    }
```

SCREENSHOT:


BOOK DATABASE
Book_Id
Author_id
Publ_id
Book_price
Book_award
Copies_avail
Book_name
Author_name
Publication
ISBN
Year_award
GET DETAILS
BACK


BOOK DATABASE
Book_Id
BK001
Author_id
A001
Publ_id
P001
Book_price
15.99
Book_award
Best Fiction
Copies_avail
7
Book_name
The Adventures of Tom Sawyer
Author_name
Mark Twain
Publication
Penguin Random House
ISBN
9780143039563
Year_award
2020
GET DETAILS
BACK

2) Calculation of fine :

CODE:

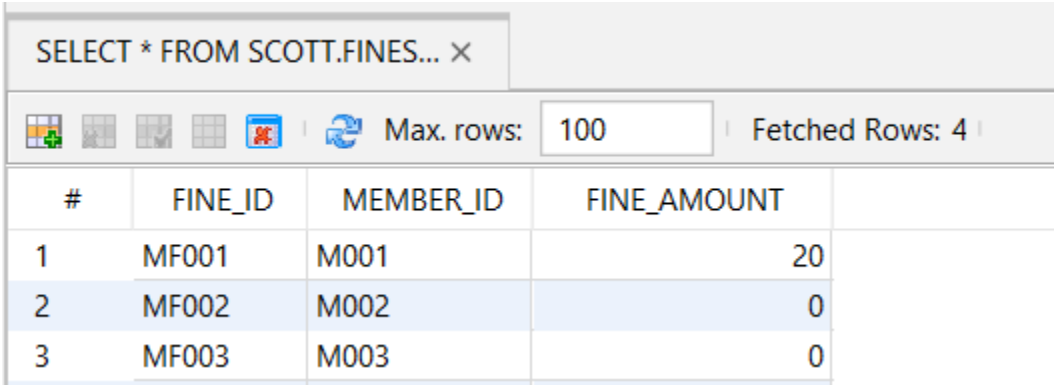
(TRigger)

```
CREATE OR REPLACE TRIGGER UPDATE_FINE_AMOUNT_TRIGGER
AFTER UPDATE OF return_date ON borrow_details
FOR EACH ROW
DECLARE
   v_due_date   DATE;
   v_book_price NUMBER;
   v_days_late  NUMBER;
   fine_amount_n NUMBER;
BEGIN
   -- Retrieve the due date and book price for the corresponding book_id
   SELECT  book_price
   INTO v_book_price
   FROM book_details
   WHERE book_id = :NEW.book_id;

   SELECT due_date
   INTO v_due_date
   FROM borrow_details
   WHERE borrow_id = :NEW.borrow_id;
   -- Calculate the number of days late if return_date is greater than due_date
   IF :NEW.return_date > v_due_date THEN
      v_days_late := TRUNC(:NEW.return_date - v_due_date); -- Calculate whole days
   ELSE
      v_days_late := 0; -- No fine if return_date is not greater than due_date
   END IF;

   -- Calculate the fine amount based on days late and book price
   fine_amount_n := v_days_late * v_book_price; -- Use := for assignment

   -- Update the fine table with the calculated fine amount
   UPDATE fines
   SET fine_amount = fine_amount_n
   WHERE fine_id = :NEW.borrow_id;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      NULL; -- Handle exception if book_id not found (optional)
   WHEN OTHERS THEN
```

```
    NULL; -- Handle other exceptions (optional)
END;
/
```

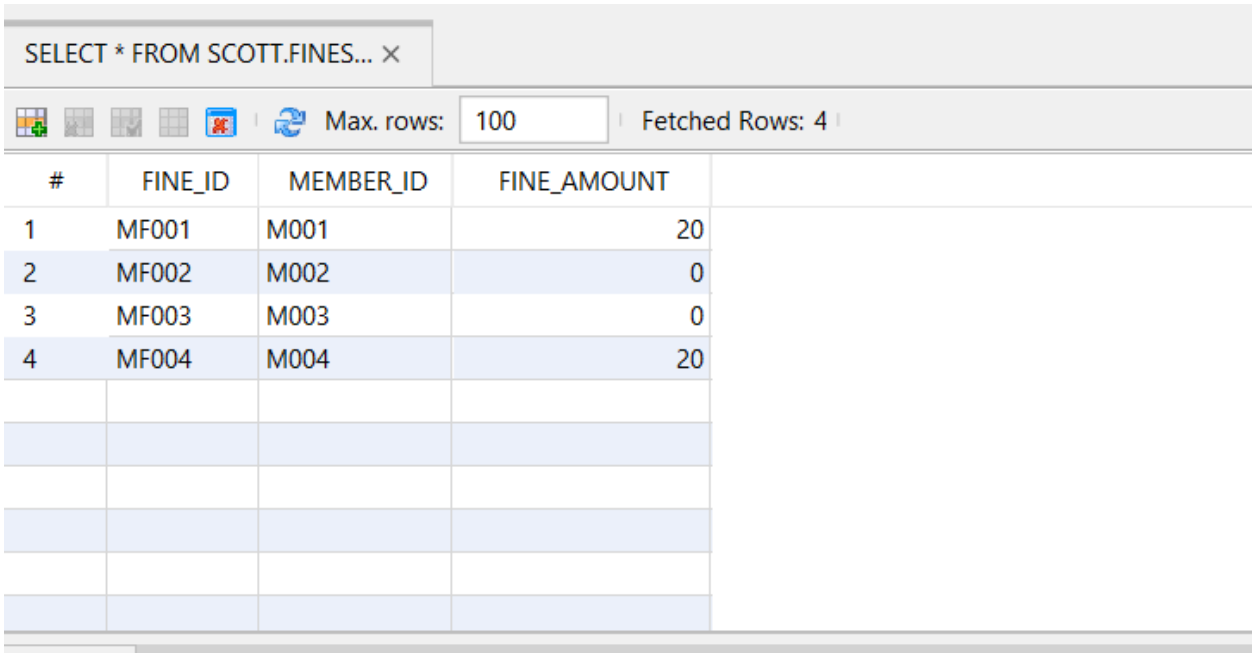
SCREENSHOT:

Before any trigger:



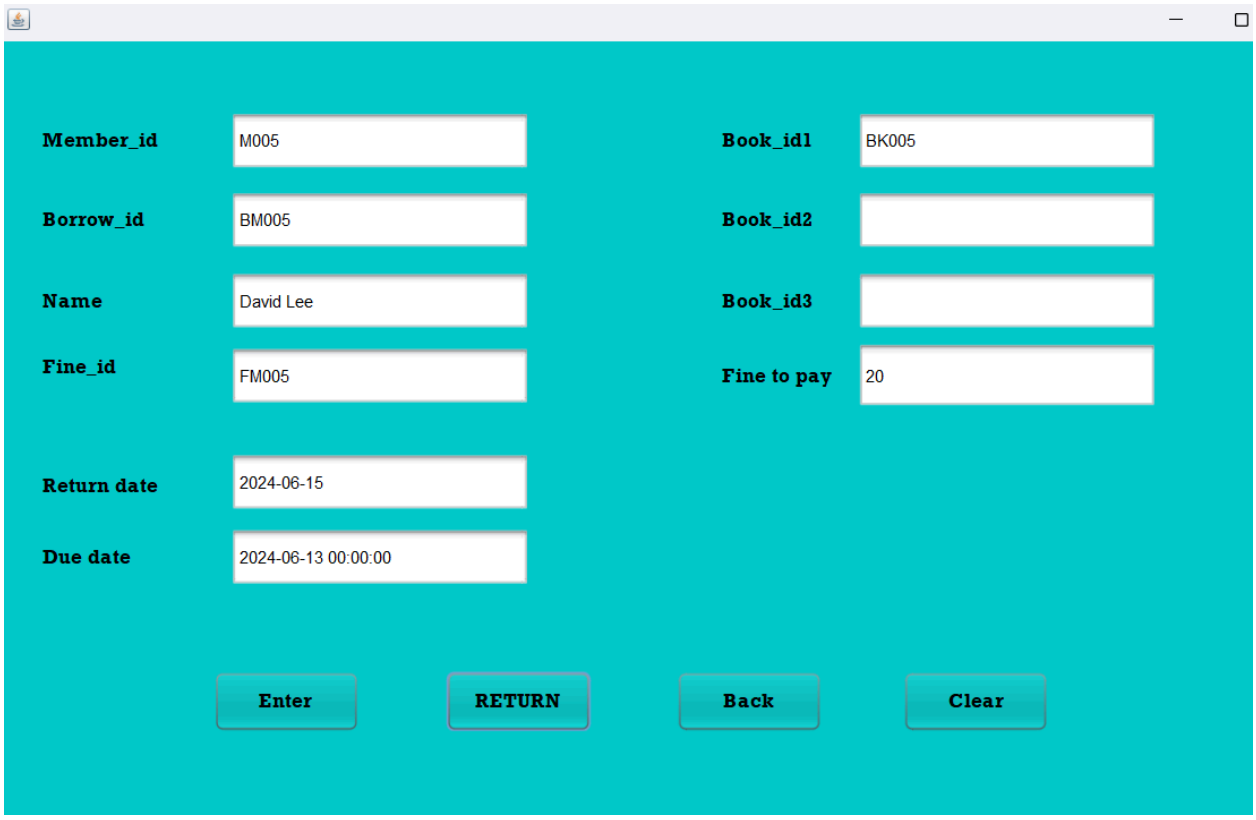
#	FINE_ID	MEMBER_ID	FINE_AMOUNT
1	MF001	M001	20
2	MF002	M002	0
3	MF003	M003	0

After trigger



#	FINE_ID	MEMBER_ID	FINE_AMOUNT
1	MF001	M001	20
2	MF002	M002	0
3	MF003	M003	0
4	MF004	M004	20

Calculated fine:



NOVELTY:

1. **On borrowing the book on a certain date the due date in the database is updated to the date 2 weeks from the borrow date this was done using triggers.**

Code:

```
        CREATE OR REPLACE TRIGGER insert_borrow_details_trigger
AFTER INSERT ON borrow_details
FOR EACH ROW
DECLARE
    due_date DATE;
BEGIN
    -- Calculate due_date as sysdate + 14 days
    due_date := SYSDATE + 14;


    -- Update borrow_details table with calculated due_date
    UPDATE borrow_details
    SET due_date = due_date
    WHERE borrow_id = :NEW.borrow_id;


    -- Reduce no_of_copies in book_details table for the corresponding
book_id
    UPDATE book_details
    SET copies_available = copies_available - 1
    WHERE book_id = :NEW.book_id;
END;
/
```

2. **On Updating the Return date while returning the book the fine amount is calculated and displayed as the number of days late (i.e return_date - due_date) multiplied by 10% of book_price and add the total fines.**

Code:

```
CREATE OR REPLACE TRIGGER UPDATE_FINE_AMOUNT_TRIGGER
AFTER UPDATE OF return_date ON borrow_details
FOR EACH ROW
DECLARE
    v_due_date   DATE;
    v_book_price NUMBER;
    v_days_late  NUMBER;
    fine_amount_n NUMBER;
BEGIN
    -- Retrieve the due date and book price for the corresponding
book_id
    SELECT  book_price
    INTO v_book_price
    FROM book_details
    WHERE book_id = :NEW.book_id;


    SELECT due_date
    INTO v_due_date
    FROM borrow_details
    WHERE borrow_id = :NEW.borrow_id;
    -- Calculate the number of days late if return_date is greater
than due_date
    IF :NEW.return_date > v_due_date THEN
        v_days_late := TRUNC(:NEW.return_date - v_due_date); --
Calculate whole days
    ELSE
        v_days_late := 0; -- No fine if return_date is not greater
than due_date
    END IF;


    -- Calculate the fine amount based on days late and book price
    fine_amount_n := v_days_late * v_book_price*0.1; -- Use := for
assignment
    fine_amount_n += fine_amount_n;
    -- Update the fine table with the calculated fine amount
```

```
    UPDATE fines
    SET fine_amount = fine_amount_n
    WHERE fine_id = :NEW.borrow_id;


EXCEPTION
    WHEN NO_DATA_FOUND THEN
        NULL; -- Handle exception if book_id not found
    WHEN OTHERS THEN
        NULL; -- Handle other exceptions
END;
/
```