Library Management System

SUBMITTED TO:- DR. ABISHI CHOWDHURY

SUBMITTED BY:-

22BAI1084 SHAKTHIREKA KARTHIKEYAN

22BAI1109 M. KOPIKA

PROBLEM STATEMENT

The current library administration systems are inefficient and difficult to operate, which leads to mistakes and lengthy procedures. A modern library management system that manages transactions, fines, book data, and member information with ease is desperately needed. The system should have an easy-to-use interface, optimize processes, and make use of technologies like Python and MySQL for effective front-end development and data administration. The objective is to guarantee the efficient administration of library resources, increase operational efficiency, and improve user experience.

PROBLEM DESCRIPTION

Despite advances in technology, many libraries struggle with outdated and inefficient ways of managing their resources and services.

- Traditional library systems often lack the resources needed to effectively manage transactions, staffing, bibliographic information, and membership information, resulting in errors and inefficiencies
- Manual systems lead to time-consuming tasks, patron dissatisfaction, and disruption of library services.
- Ineffective bookkeeping: Searching for specific books, verifying availability, and maintaining credit accounts are all time-consuming and error-prone
- Limited access: Access to library materials during opening hours or remotely is not possible, hindering the user.
- Complicated loan and return processes: Manual book delivery and returns are slow and prone to human error, resulting in misplaced books and incorrect charges.

PROBLEM DESCRIPTION

- Lack of Data & Reporting: Reporting on book usage, membership growth and excessive penalties is difficult and unreliable due to lack of centralized data systems
- Limited user management: Manually registering and managing user accounts is tedious and lacks features for user self-service or personalized recommendations.

To meet these challenges, there is an urgent need for a comprehensive modern library management system (LMS) that addresses the shortcomings of existing systems. The main problem is the lack of an integrated and user-friendly system that effectively manages all aspects of library operations, including transactions, penalties, subscription processing and real-time updates

ALL POSSIBLE ASSUMPTIONS

- There is a strong entity Staff assuming the library employee staff having the attributes email id, staff id, name, gender, phone no, address, join date, position , salary etc.

- There is a strong entity book assuming the book in the library having attributes, book title, categories (having genre and language), publisher, multiple authors, having condition for book borrow or checkout, book id , edition copies and status

- There is a strong entity Member assuming the member of the library having attributes Member_ID,SSN, member name, age, gender, address, phone number, email id,join date and status of the member

- There is a strong entity Membership assuming the membership taken by the member of the library having attributes Member_ID, date, renewed/joined, amount and payment_status.

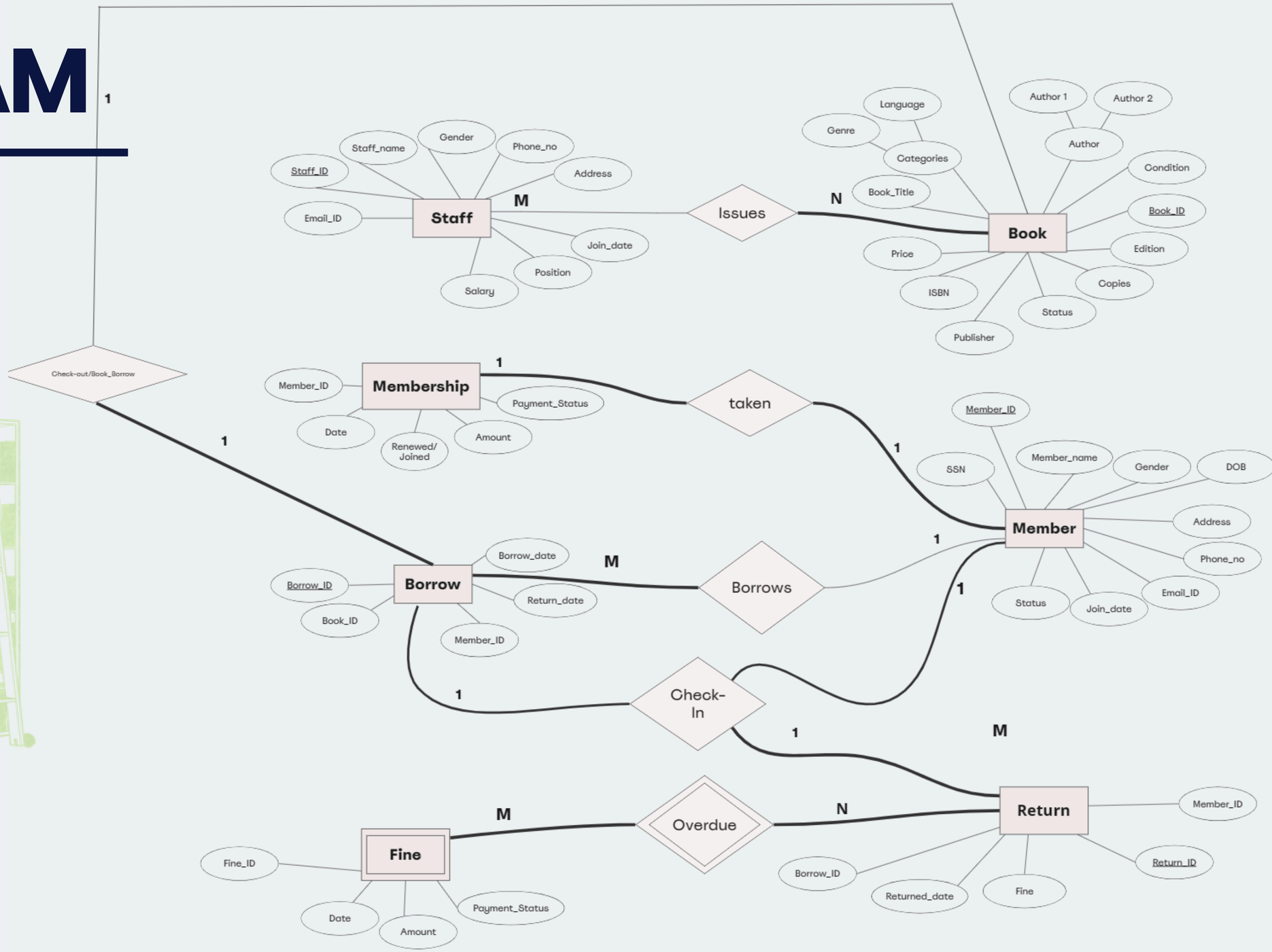
ALL POSSIBLE ASSUMPTIONS

- There is a strong entity **Borrow** assuming the books being borrowed by the member. It has attributes like borrow id, book id, member id, return date and borrow date.

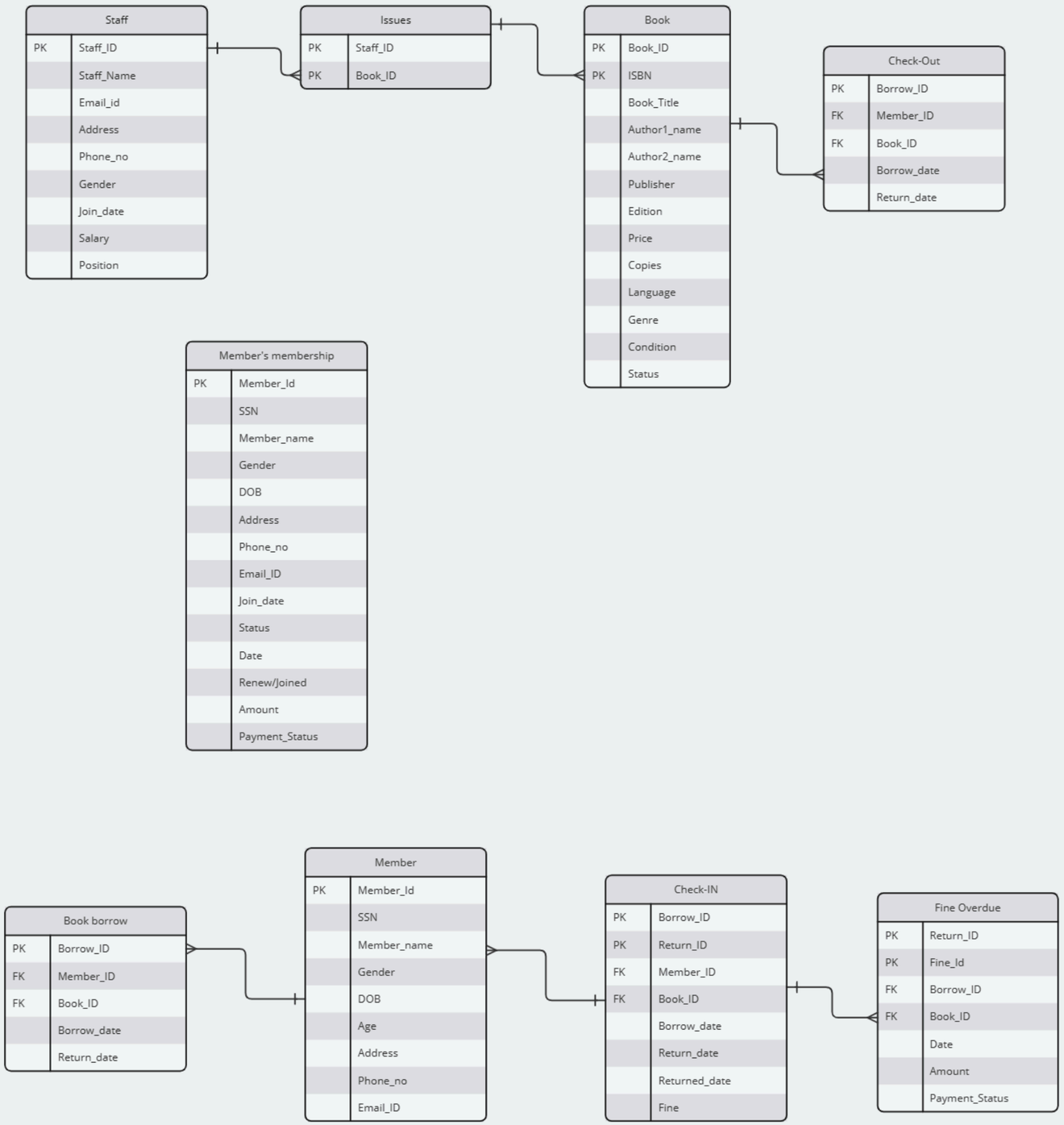
- There is a weak entity **Fine** assuming the fine to be paid by the members if the book is not returned. It has attributes fine id, Date, amount and payment status .

- There is a strong entity **Return** assuming the return details when a member returns a book and has attributes returned date(when the book is returned), fine, return_id, borrow_id and member_id date(ideally when the book is supposed to be returned) and return Id.

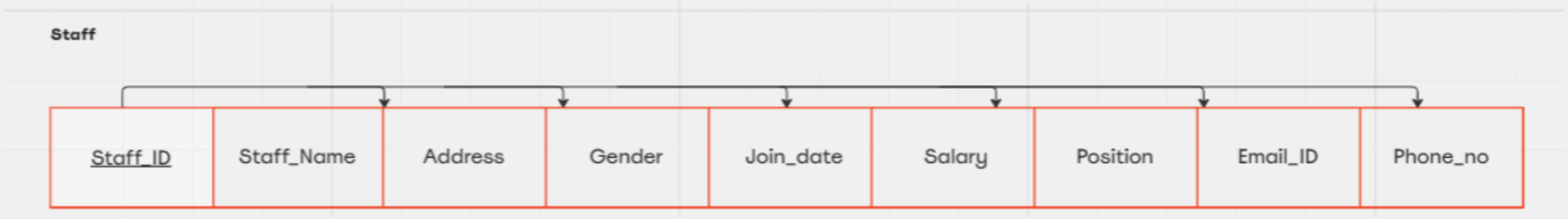
ER DIAGRAM


Staff
Staff_ID
Staff_name
Gender
Phone_no
Address
Email_ID
Join_date
Position
Salary
Issues
M
N
Book
Language
Genre
Categories
Book_Title
Author 1
Author 2
Author
Condition
Book_ID
Edition
Price
Copies
ISBN
Status
Publisher
Check-out/Book_Borrow
1
Membership
Member_ID
Payment_Status
Date
Renewed/ Joined
Amount
taken
Member
Member_ID
SSN
Member_name
Gender
DOB
Address
Phone_no
Email_ID
Status
Join_date
Borrow
Borrow_ID
Book_ID
Borrow_date
Return_date
Member_ID
Borrows
Check-In
Return
Member_ID
Return_ID
Borrow_ID
Returned_date
Fine
Overdue
Fine
Fine_ID
Date
Amount
Payment_Status

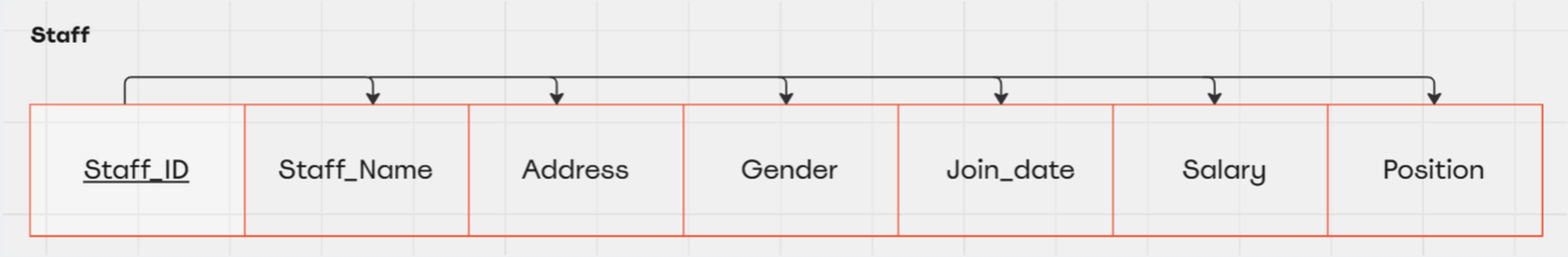
RELATIONAL MODEL



NORMALIZATION

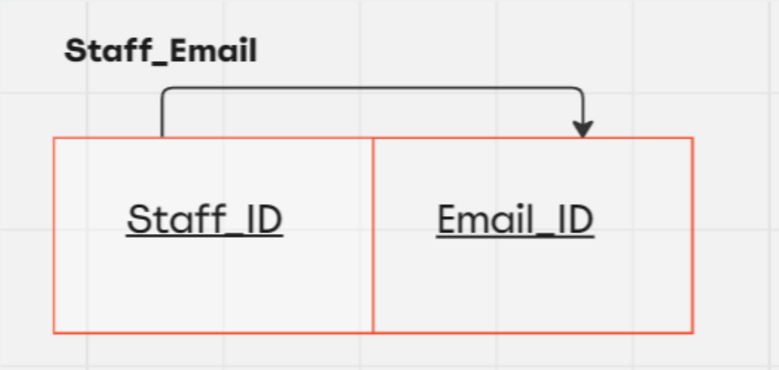


- The staff table has multivalued attributes and therefore it is not in 1NF so creating a seperate table for it so that it is in 1NF



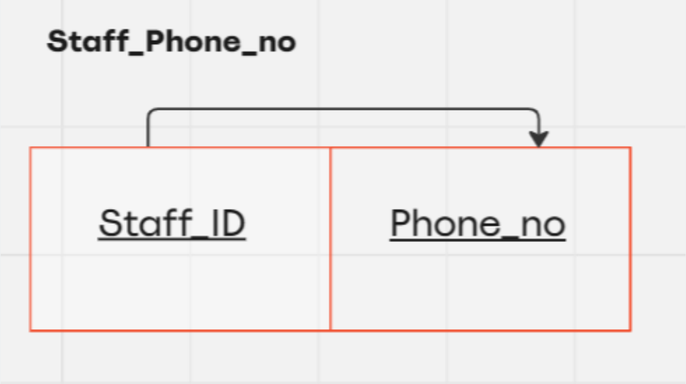
- This is normalized form since the staff table is in 1NF as it has no multivalued attribute
- The table is also in 2NF as it has no partial dependencies and non-key attributes fully functionally depend on the candidate key Staff_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

NORMALIZATION



- This is normalized form since the table is in 1NF as it has no multivalued attribute
- The table is also in 2NF as it has no partial dependencies and non-key attribute fully functionally depends on the candidate key Staff_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

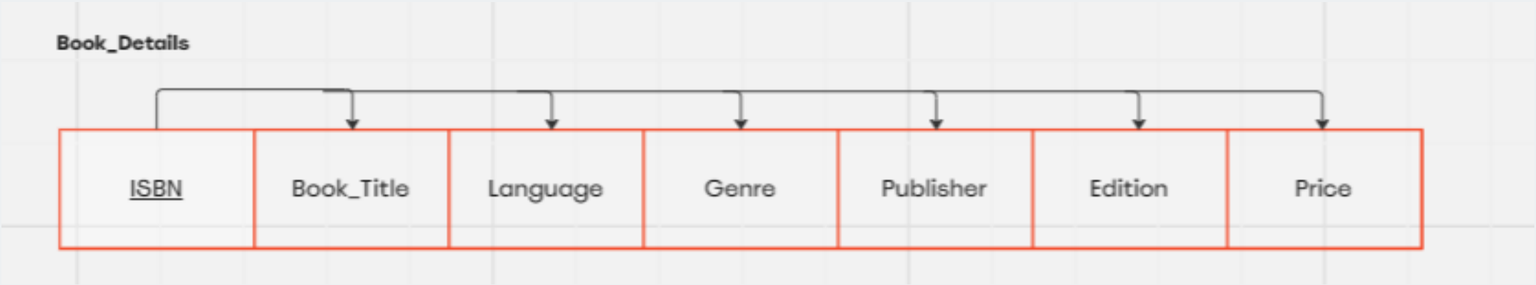
NORMALIZATION



- This is normalized form since the table is in 1NF as it has no multivalued attribute
- This table is also in 2NF as it has no partial dependencies and non-key attribute phone number fully functionally depends on the candidate key Staff_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

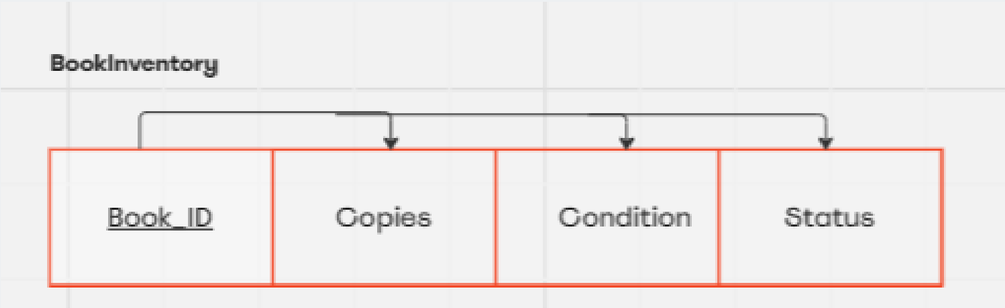
NORMALIZATION

- The book table has multivalued attributes and therefore it is not in 1NF so creating a seperate table for it so that it is in 1NF



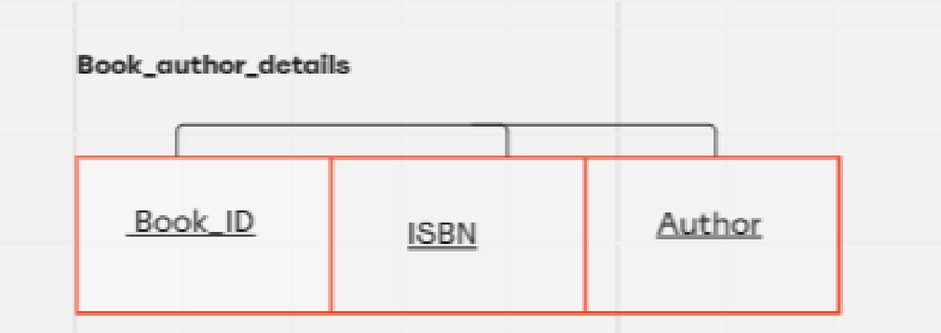
- This is normalized form since the book inventory table is in 1NF as it has no multivalued attribute
- The table is also in 2NF as it has no partial dependencies and non-key attributes fully functionally depend on the candidate key ISBN
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

NORMALIZATION



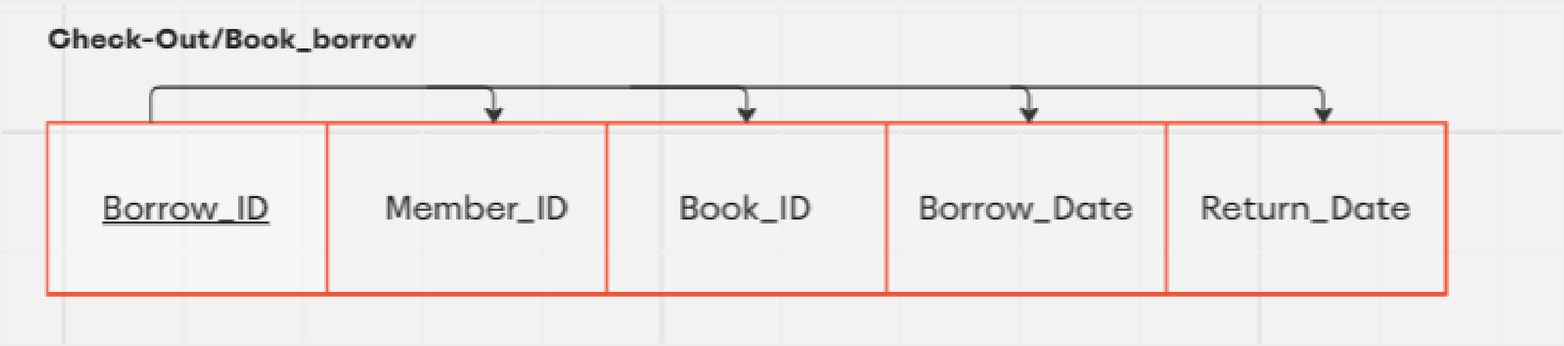
- This is normalized form since the table is in 1NF as it has no multivalued attribute
- This table is also in 2NF as it has no partial dependencies and non-key attributes copies, condition and status fully functionally depend on the candidate key Book_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

NORMALIZATION



- The book_author_details table is in normalized form since the table is in 1NF as it has no multivalued attribute
- This table is also in 2NF as it has no partial dependencies and non-key attributes fully functionally depend on the candidate key Book_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

NORMALIZATION

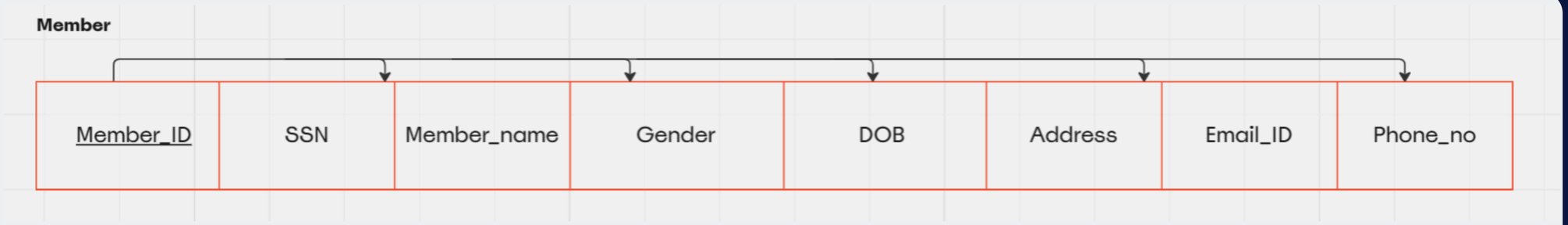


- This check-out or book_borrow table is in normalized form since the table is in 1NF as it has no multivalued attribute
- This table is also in 2NF as it has no partial dependencies and non-key attributes fully functionally depends on the candidate key Borrow_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

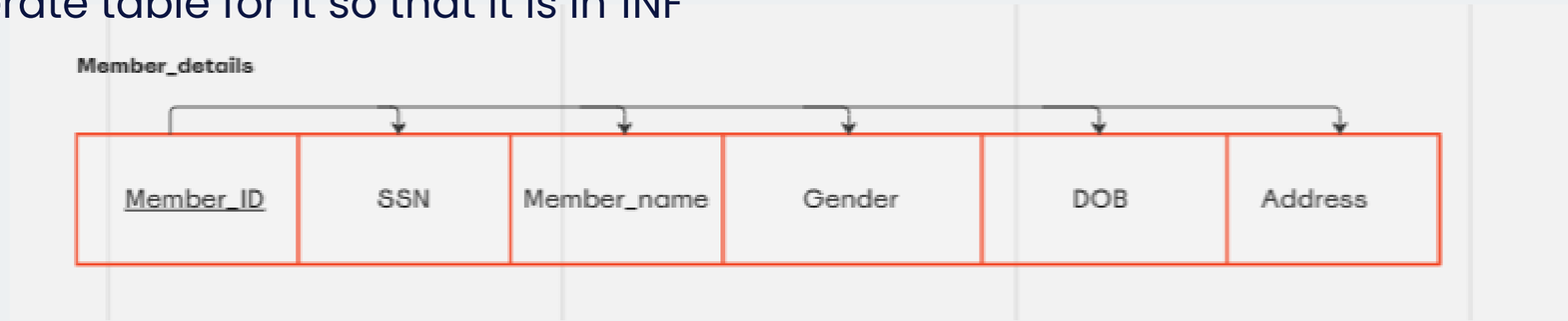
NORMALIZATION

- This table is in normalized form since the table is in 1NF as it has no multivalued attribute
- This table is also in 2NF as it has no partial dependencies and non-key attributes like SSN, member_name, gender, DOB, Address, Join_date, Status, Payment_status, Renew/Joined and Amount fully functionally depend on the candidate key Member_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

NORMALIZATION

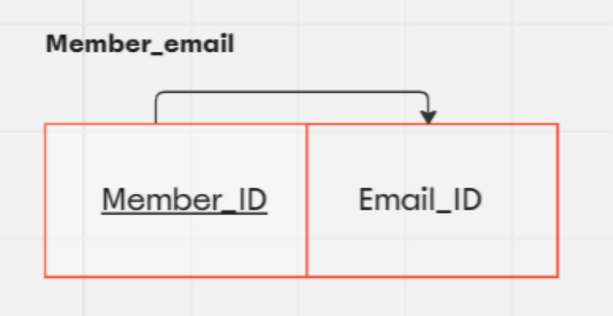


- The Member table has multivalued attributes and therefore it is not in 1NF so creating a seperate table for it so that it is in 1NF



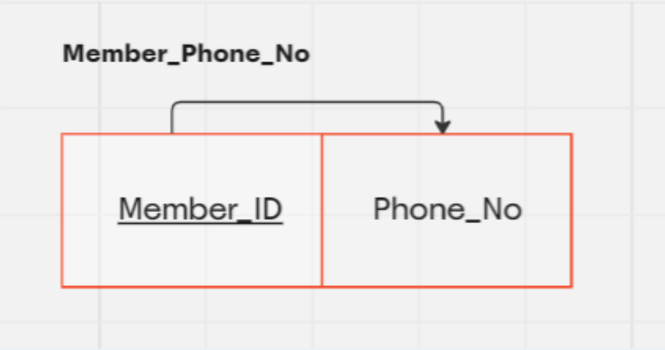
- This is normalized form since the table is in 1NF as it has no multivalued attribute
- The table is also in 2NF as it has no partial dependencies and non-key attributes fully functionally depend on the candidate key Member_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

NORMALIZATION



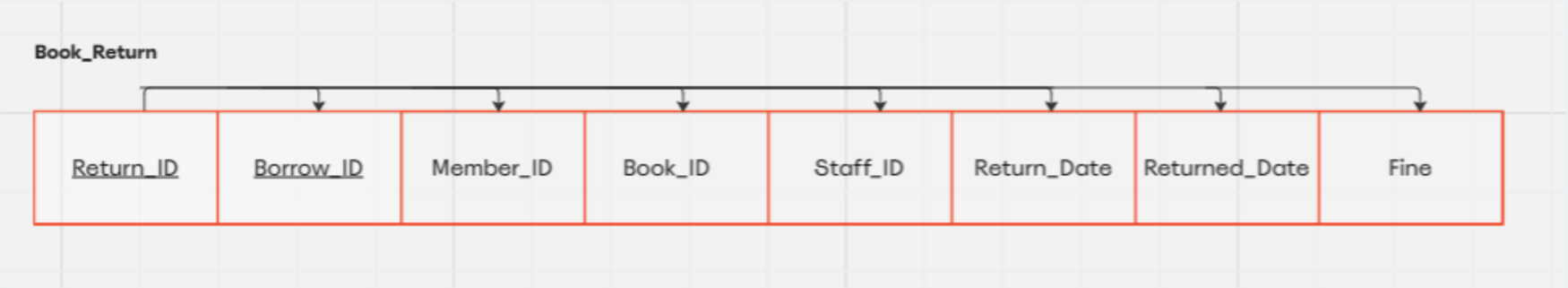
- This table is in normalized form since the table is in 1NF as it has no multivalued attribute
- This table is also in 2NF as it has no partial dependencies and non-key attribute email_ID fully functionally depends on the candidate key Member_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

NORMALIZATION



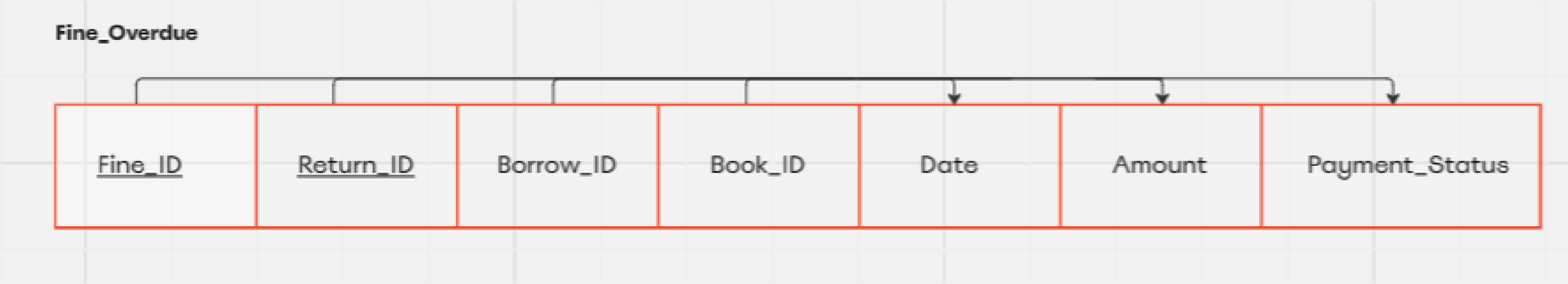
- This table is in normalized form since the table is in 1NF as it has no multivalued attribute
- This table is also in 2NF as it has no partial dependencies and non-key attribute Phone_No fully functionally depends on the candidate key Member_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

NORMALIZATION



- This book_return table is in normalized form since the table is in 1NF as it has no multivalued attributes
- This table is also in 2NF as it has no partial dependencies and non-key attributes fully functionally depend on the candidate keys Return_ID, Borrow_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

NORMALIZATION



- This Fine_Overdue table is in normalized form since the table is in 1NF as it has no multivalued attributes
- This table is also in 2NF as it has no partial dependencies and non-key attributes fully functionally depend on the candidate keys Return_ID, Fine_ID
- It also follows 3NF as it has no transitive dependencies.
- Finally the table is in BCNF as the LHS is a super key.

IMPLEMENTATION

Menu

Python 3.9.0 Shell

File Edit Shell Debug Options Window Help

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct  5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\kopika\Documents\VIT\Academics\SEM-4\Assignments and Practice\DBMS\Library managem
ent system.py
Required database and tables have been created.

---LIBRARY MENU---
1.Book details
2.Member details
3.Staff details
4.Book addition
5.Book deletion
6.Book deactivation
7.Member addition
8.Modify member details
9.Member deletion
10.Member cancellation
11.Renew Membership
12.Staff addition
13.Modify staff details
14.Staff deletion
15.Borrow book
16.Return book
17.Fine payment
18.Search
19.transaction details
0.Exit
```

IMPLEMENTATION

Adding a new book

Python 3.9.0 Shell

File Edit Shell Debug Options Window Help

```
7.Member addition
8.Modify member details
9.Member deletion
10.Member cancellation
11.Renew Membership
12.Staff addition
13.Modify staff details
14.Staff deletion
15.Borrow book
16.Return book
17.Fine payment
18.Search
19.transaction details
0.Exit

Enter the choice:4
Enter the ISBN:712345
Enter the book title:The Lord of the Rings
Enter the author names seperated by commas(,):JRR Tolkein
Enter the publisher's name:Bloomsbury Publications
Enter the edition year:2007
Enter the price:999
Enter the genre:Fantasy fiction
Enter the language:english
Enter the number of copies present:1
Enter whether Active/Condemned:active
Enter the status as Available/Checked Out:available
Book added
Enter more records?(y/n):
```

Ln: 42 Col: 25

IMPLEMENTATION

Displaying all the books

Python 3.9.0 Shell

File Edit Shell Debug Options Window Help

```
Enter the choice:1

BOOK DETAILS
a. To display in ascending order
b. To display in descending order
c. To display in the order as stored

Enter the choice: a
Book details:
ISBN: 987623
Title: Harry Potter and the philosophers stone
Language: english
Genre: fantasy fiction
Publisher: Penguin
Edition: 1999
Price: 499.00
Author: JK Rowling
Book ID: 10001
Copies: 2
Condition: active
Status: Checked Out

ISBN: 123456
Title: Higher engineering mathematics
Language: english
Genre: education
Publisher: Pearson
Edition: 2006
```

IMPLEMENTATION

Updating staff's details

```
Enter the choice:13
Enter the staff id of the staff to be modified:20001

a.Address
b.Contact
c.Email

Enter the choice:a
Enter the new address:Iyyatil Junction, Mahakavi G road, Kochi, Kerala
Address has been updated
```

Python 3.9.0 Shell

File Edit Shell Debug Options Window Help

```
Enter the choice:3

STAFF DETAILS
a. To display in ascending order
b. To display in descending order
c. To display in the order as stored

Enter the choice: b
Staff details:
Staff ID: 20001
Name: Anastasia
Address: Iyyatil Junction, Mahakavi G road, Kochi, Kerala
Gender: F
Join Date: 2024-04-21
Salary: 20000.00
Position: head librarian
Email: anasa@hotmail.com
Phone Number: 9886646536
```

IMPLEMENTATION

Deleting a member

Python 3.9.0 Shell

File Edit Shell Debug Options Window Help

```
Enter the choice:2

MEMBER DETAILS
a. To display in ascending order
b. To display in descending order
c. To display in the order as stored

Enter the choice: c
Member details:
Member ID: 30001
SSN: 128128
Name: Katrina
Gender: F
DOB: 2002-02-02
Address: Chennai
Payment Status: Successful
Join Date: 2024-04-21
Status: Active
Renew Joined: Renewed
Amount: 499.00
Email: katrina@gmail.com
Phone Number: 8053937373
```

Python 3.9.0 Shell

File Edit Shell Debug Options Window Help

```
Enter the choice:9
Enter the member ID of the member to be deleted:30001
Member deleted
```

IMPLEMENTATION

Borrowing a book

```
Enter the choice:15
Enter the book id of the book:10001
Enter the member id of the member:30003
Enter the staff id of the staff issuing the book:20002
Enter the return date:2024-05-01
Book borrowed successfully with the borrow id: 40005
```

Returning a book

```
Enter the choice:16
Enter the borrow id: 40005
Enter the staff id: 20001
Book returned successfully with no fines.
```

IMPLEMENTATION

Searching a book

```
18.Search
19.transaction details
0.Exit

Enter the choice:18

a.Search for a book
b.Search for a member
c.Search for a staff

Enter the choice: a
Enter the book id: 10004
Book details:
ISBN: 654321
Title: The Da Vinci Code
Language: english
Genre: Historical fiction
Publisher: Doubleday
Edition: 2005
Price: 699.00
Author: Dan Brown
Copies: 1
Condition: active
Status: available
```

IMPLEMENTATION

Book borrow history

```
*Python 3.9.0 Shell*
File  Edit  Shell  Debug  Options  Window  Help
14.Staff deletion
15.Borrow book
16.Return book
17.Fine payment
18.Search
19.transaction details
0.Exit

Enter the choice:19

a.Borrow details
b.Return details
c.Fine details

Enter the choice:a
40001 | 30002 | 10001 | 2024-04-21 | 2024-04-26
40002 | 30002 | 10002 | 2024-04-22 | 2024-04-29
40003 | 30003 | 10003 | 2024-04-23 | 2024-04-30
40004 | 30002 | 10002 | 2024-04-23 | 2024-04-30
40005 | 30003 | 10001 | 2024-04-23 | 2024-05-01
```

Book return history

```
*Python 3.9.0 Shell*
File  Edit  Shell  Debug  Options  Window  Help
15.Borrow book
16.Return book
17.Fine payment
18.Search
19.transaction details
0.Exit

Enter the choice:19

a.Borrow details
b.Return details
c.Fine details

Enter the choice:b
50001 | 40001 | None | 10001 | None | 2024-04-26 | 2024-04-22 | 0.00
50002 | 40001 | None | 10001 | None | 2024-04-26 | 2024-04-23 | 0.00
50003 | 40002 | None | 10002 | None | 2024-04-29 | 2024-04-23 | 0.00
50004 | 40003 | None | 10003 | None | 2024-04-30 | 2024-04-23 | 0.00
50005 | 40004 | None | 10002 | None | 2024-04-30 | 2024-04-23 | 0.00
50006 | 40005 | None | 10001 | None | 2024-05-01 | 2024-04-23 | 0.00
```

CONCLUSION

The development of a library management system using Python for the front end and MySQL for the backend has resulted in a robust and efficient solution. Leveraging Python's versatility and MySQL's relational database capabilities, the system offers seamless book management and database management functionalities. Through further designing of a user-friendly and attractive user interface, it will provide users and administrators with powerful tools for managing the library's resources. Moving forward, continued refinement and optimization will ensure scalability and adaptability to evolving needs. This project exemplifies the synergy between Python and MySQL in building practical and effective database-driven applications.

THANK YOU!