UNIVERSITY OF
SCIENCE AND TECHNOLOGY
OF HANOI

FUNDAMENTAL OF DATABASES

Final Project Report

Library Management System

Group 15 – CSDS

Hanoi, VN, November 2023

Group info

- **Nguyễn Thị Vân (Leader) -** 22BI13459 - [vannt.22BI13459@usth.edu.vn]
- **Trần Trung Kiên -** 22BI13227 - [kientt.22BI13227@usth.edu.vn]
- **Nguyễn Bá Vinh -** 22BI13472 - [vinhnb.22BI13472@usth.edu.vn]
- **Nguyễn Minh Đức –** 22BI13091 – [ducnm.22bi13091@usth.edu.vn]
- **Trần Trung Hiếu –** 22BI13162 – [hieutt.22bi13162@usth.edu.vn]

Contents

1. Introduction

- What is a library management system?

  A library management system (LMS) is a software program developed to help library staff effectively manage a variety of tasks in a library and meet all customer needs. It has tools and features to organize, catalog, and track library resources such as books, multimedia, and digital content. Nowadays, books are typically found at libraries. Typical features of an LMS include resource reservations and online catalog access, along with services for categorizing materials, maintaining user accounts, and creating reports. It is basically an all-inclusive system that makes the library's general operation easier.

- Why do we need to have a library management system?

  A library management system is critical for a library's successful operation, ensuring that resources are well-maintained and easily available, and that users have a positive and efficient experience while communicating with the library's services. Because of the massive number of books, book management has become essential. A library management system will eliminate all manual effort, save time, and give detailed book information.

- How can we build a library management system using SQL?

  Creating an effective library management system with SQL requires a solid understanding of SQL and database management principles, as well as database design, query implementation, business logic, and interface development. This involves several steps:

  1. Define requirements: outline all the functions needed from users' views, both borrower and manager.
  2. Design database: using SQL, we can create a database schema. Then design tables for books, borrowers, staffs, … and created relationships between them using keys. The relationship between tables then will be illustrated by the ERD or the E/R table.
  3. Table creation: Using SQL commands to create tables based on what we have designed, then import the data needed.
  4. Implement tasks: from the given tasks, we can implement many commands or functions to solve the tasks. One task can be solved in many ways.
  5. Test and Optimize: this step involves testing the queries to check whether they are functioning correctly. On the other hand, we also want to optimize the performance of

the system by using many functions or procedures to enhance the speed and saving time for the user.

- Goals of this project:
  Knowing what a library management system is, why we need to have a LMS, and how to build a LMS, we want to create an effective library management system which can give solutions for many tasks from the user in 2 main groups: staff and borrowers. In this project, we also want to expand this system into managing a local or a city library but not restricted to the school or university library, with 1000 book data and 200 customer data.

- Data source
  The data for this project is created manually by our team using the tool from Google Sheets. Here is the link to the spreadsheet containing data.
  Link

2. User Requirements

One of the most important tasks in building an effective database is to gather all the user requirements, which are from the customers and the staff.

2.1. The Borrower

- Borrowers can borrow books from the library and must return them on time.
- Borrowers can change their personal information or provide enough information if they want to borrow reading materials from the library.
- Borrowers can know the information about books or time they need to return the books.
- Borrowers can also call for help from the staff if they need.

2.2. The Staffs

In our project, we consider the staff into 5 main positions. Each member of staff in a specific position has their own jobs and responsibilities, together with the amount of information they can access in the library management system.

- Manager: managing all staff and can have access to all information in the library management system. Including: books, staff, and borrower information.
- Library Assistant: helping the manager with the management tasks and can get access to all information about books and borrowers.

- Library Technician: dealing with technical problems which can be encountered in a library or from the borrower. They know the books and the borrower's information, especially the problems that borrowers can have.
- Archivist: working with book data only and they can manage or organize books in the library.
- Librarian: they do the main tasks in a library such as giving the book to the borrower, checking for the number of books remaining in the library, organizing borrower information and other tasks. They can get access to the book and the borrower's information.

2.3. Functions

In this part, we will build functions to solve the tasks given by 2 main groups: Customer and Staff. Some important functions are inserting and deleting customers, borrowdetails or books information. Then we will implement some other functions that are frequently used by these groups of people.

2.3.1. Functions for Customer

- Insert new customer information (important function).
- Insert new borrow detail information when there is a new customer borrow turn (important function).
- Finding books with specific author's information.
- Finding information of a book.
- Finding books published in a year with descending order of rating.
- Finding books with individual genre and for an assigned age group.
- Entering the problem (with problem ID) and receive the information from the staff for help.
- Checking for the exact deadline to return the book.

2.3.2. Functions for Staff

Position	Functions
Manager	• Finding books genre with the highest frequency of borrowed. • Counting the number of staff in each position.
Library Assistant	• Finding borrow details of a specific customer. • Delete a specific borrow detail (important function). • Delete a specific customer (important function).
Archivist	• Counting the number of books for each genre. • Insert a new book data (important function). • Delete a book data (important function).
Librarian	• Displaying information of borrowers who return books late. • Updating the current shelf information for corresponding books.

3. Entity Relationship Diagram

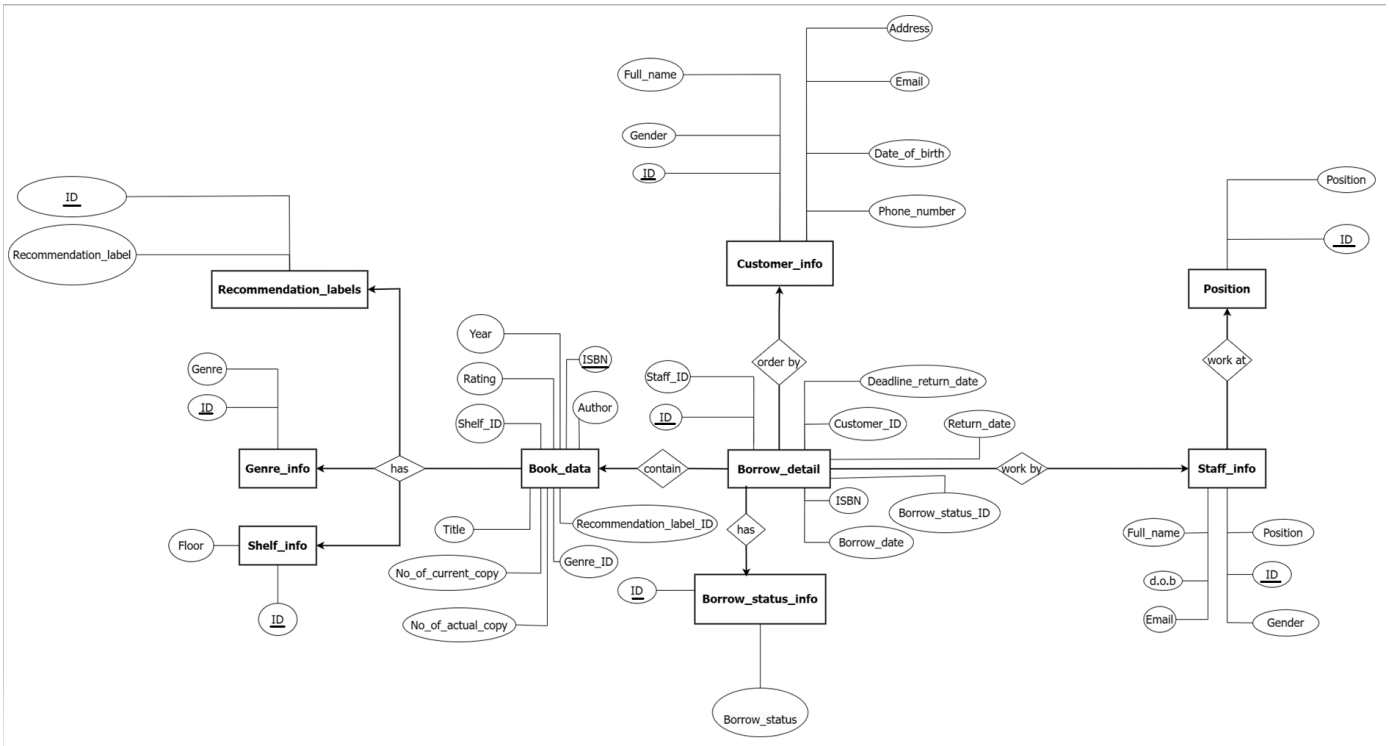


Figure 1: Entity Relationship Diagram

- Customer_info – Borrow_detail (Many to One): One customer can have many borrow turns (each turn is corresponding to a borrow ID) and one borrow detail can only be ordered by one customer.
- Staff_info – Borrow_detail (Many to One): One staff can work on many borrow details and one borrow detail is managed by one staff only.
- Position – Staff_info(Many to One): One position can have many staff but one staff can only have one position.
- Borrow_status_info – Borrow_detail (Many to One): One borrow detail can have only one borrow status but one borrow status can belong to many borrow details.
- Book_data – Borrow_detail (Many to One): One book data can belong to many borrow detail but one borrow detail can only have one book data.
- Recommendation_labels – Book_data (Many to One): One recommendation label can belong to many book data, but one book data can only have one recommendation label.
- Genre_info – Book_data (Many to One): One book genre can have many book data but one book data can only have one book genre.
- Shelf_info – Book data (Many to One): One shelf can contains many books but one book can only belong to one shelf.

4. Schema

There are 9 tables in our schema: borrow_detail, borrow_status_info, book_data, genre_info, recommendation_labels, shelf_info, staff_info, position and customer_info. Each table has data and relationships with other tables. In this part we will give a detailed description for each table.

Table	**Description**
Borrow_detail	Contains the information about the customer and information of books that they borrowed, with detail time and corresponding staff. Primary key: ID
Borrow_status_info	Contains the information of the borrow status: returned, have not returned or returned late. Primary key: ID
Book_data	Contains necessary information of 1000 books including ISBN, title, author, year published, gerne, rating, … Primary key: ISBN
Genre_info	Contains 26 book genres with corresponding ID. Primary key: ID
Recommendation_labels	Contains information of 5 age groups that books can be assigned to. Primary key: ID

Shelf_info	Contains information of 10 book shelfs. The ID is also the order of the bookshelf, and each bookshelf will be placed on a specific floor. Primary key: ID
Staff_info	Contains personal data of staff in the library, including ID, full name, gender, date of birth, contact information…. Primary key: ID
Position	Contains information about 5 positions that a staff can have, each position is assigned with an ID. Primary key: ID
Customer_info	Contains data of 200 customers, including ID, full name, date of birth, gender, contact information, … Primary key: ID

Although the primary key in most of our tables have the same name as ID, we ensure this will not affect the effectiveness and correctness of the system since whenever we want to refer to a specific key of specific table, we always use the dot notation (membership operator). The reason we want to keep this same name is because we have a lot of tables and if we give each primary key of each table a different name, this will make the process of implementation slower, thus, we want to optimize the time used for implementation and simplify our program.

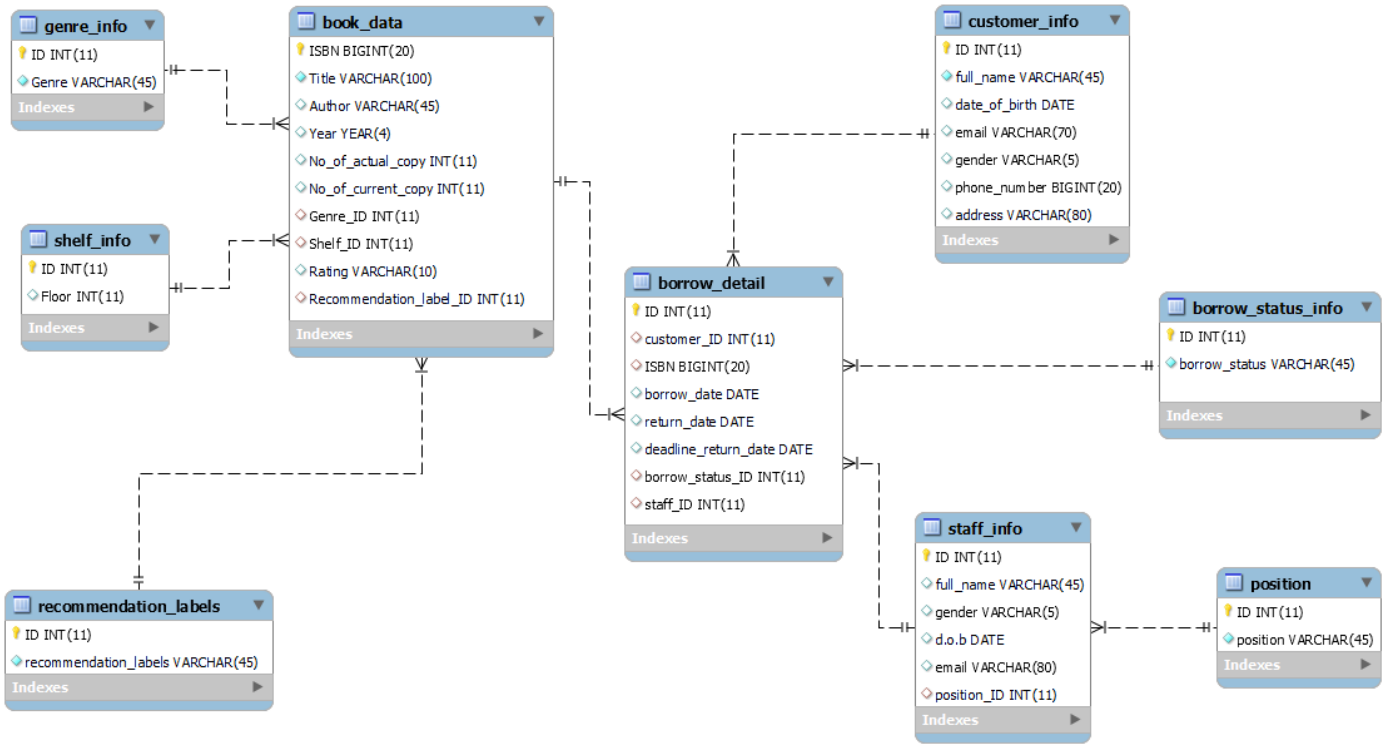


Figure 2: E/R Diagram Table

5. Project Implementation

5.1. Create Database and Tables Queries

- Create database:

```
DROP database if exists Librarymanagement;
CREATE DATABASE if not exists LibraryManagement;
USE LibraryManagement;
```

We need to drop the database if it exists first, since if we have created it before, we can create the same database again. After ensuring that our database does not exist, we can create a new one named “Librarymanagement”.

5.1.1. Book_data table

This table includes information of: ISBN, Title, Author, Year, No_of_actual_copy, No_of_current_copy, Genre_ID, Shelf_ID, Rating, Recommendation_label_ID of books.

Create table SQL code:

```
CREATE TABLE `book_data` (
  `ISBN` bigint(20) NOT NULL,
  `Title` varchar(100) NOT NULL,
  `Author` varchar(45) DEFAULT NULL,
  `Year` year(4) DEFAULT NULL,
  `No_of_actual_copy` int(11) DEFAULT NULL,
  `No_of_current_copy` int(11) DEFAULT NULL,
  `Genre_ID` int(11) DEFAULT NULL,
  `Shelf_ID` int(11) DEFAULT NULL,
  `Rating` varchar(10) DEFAULT NULL,
  `Recommendation_label_ID` int(11) DEFAULT NULL,
  PRIMARY KEY (`ISBN`));
```

Table information:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ISBN	BIGINT(20)	☑	☑	☐	☐	☐	☐	☐	☐	
Title	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
Author	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	
Year	YEAR(4)	☐	☐	☐	☐	☐	☐	☐	☐	
No_of_actual_copy	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	
No_of_current_copy	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	
Genre_ID	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	
Shelf_ID	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	
Rating	VARCHAR(10)	☐	☐	☐	☐	☐	☐	☐	☐	
Recommendation_label_ID	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	

5.1.2. Borrow_detail table

This table includes information of: ID, customer_ID, ISBN, borrow_date, return_date, deadline_return_date, borrow_status_ID, staff_ID of borrow details.

Create table SQL code:

```
CREATE TABLE `borrow_detail` (
`ID` int(11) NOT NULL,
`customer_ID` int(11) DEFAULT NULL,
`ISBN` bigint(20) DEFAULT NULL,
`borrow_date` date DEFAULT NULL,
`return_date` date DEFAULT NULL,
`deadline_return_date` date DEFAULT NULL,
`borrow_status_ID` int(11) DEFAULT NULL,
`staff_ID` int(11) DEFAULT NULL,
PRIMARY KEY (`ID`));
```

Table information:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
customer_ID	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	
ISBN	BIGINT(20)	☐	☐	☐	☐	☐	☐	☐	☐	
borrow_date	DATE	☐	☐	☐	☐	☐	☐	☐	☐	
return_date	DATE	☐	☐	☐	☐	☐	☐	☐	☐	
deadline_return_date	DATE	☐	☐	☐	☐	☐	☐	☐	☐	
borrow_status_ID	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	
staff_ID	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	

5.1.3. Borrow_status_info table

This table including information of: ID and borrow_status

Create table SQL code:

```
CREATE TABLE `borrow_status_info` (
`ID` int(11) NOT NULL,
`borrow_status` varchar(45) NOT NULL,
PRIMARY KEY (`ID`));
```

Table information:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
borrow_status	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	

5.1.4. Customer_info table

This table includes information of: ID, full_name, date_of_birth, email, gender, phone_number, address of customers.

Create table SQL code:

```
CREATE TABLE `customer_info` (
`ID` int(11) NOT NULL,
`full_name` varchar(45) NOT NULL,
`date_of_birth` date DEFAULT NULL,
`email` varchar(70) DEFAULT NULL,
`gender` varchar(5) DEFAULT NULL,
`phone_number` bigint(20) DEFAULT NULL,
`address` varchar(80) DEFAULT NULL,
PRIMARY KEY (`ID`));
```

Table information:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
full_name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
date_of_birth	DATE	☐	☐	☐	☐	☐	☐	☐	☐	
email	VARCHAR(70)	☐	☐	☐	☐	☐	☐	☐	☐	
gender	VARCHAR(5)	☐	☐	☐	☐	☐	☐	☐	☐	
phone_number	BIGINT(20)	☐	☐	☐	☐	☐	☐	☐	☐	
address	VARCHAR(80)	☐	☐	☐	☐	☐	☐	☐	☐	

5.1.5. Genre_info table

This table includes information of: ID and Genre of books.

Create table SQL code:

```
CREATE TABLE `genre_info` (
`ID` int(11) NOT NULL,
`Genre` varchar(45) NOT NULL,
PRIMARY KEY (`ID`));
```

Table information:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
Genre	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	

5.1.6. Position table

This table includes information of ID and position of staff.

Create table SQL code:

```
CREATE TABLE `position` (
`ID` int(11) NOT NULL,
`position` varchar(45) NOT NULL,
PRIMARY KEY (`ID`));
```

Table information:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
position	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	

5.1.7. Recommendation_labels table

This table includes information of: ID and recommendation_label of books.

Create table SQL code:

```
CREATE TABLE `recommendation_labels` (
`ID` int(11) NOT NULL,
`recommendation_label` varchar(45) NOT NULL,
PRIMARY KEY (`ID`));
```

Table information:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
recommendation_labels	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

5.1.8. Shelf_info table

This table includes information of: ID and Floor of shelfs contain books.

Create table SQL code:

```
CREATE TABLE `shelf_info` (
`ID` int(11) NOT NULL,
`Floor` int(11) NOT NULL,
PRIMARY KEY (`ID`));
```

Table information:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
Floor	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

5.1.9. Staff_info table

This table includes information of: ID, full_name, gender, d.o.b (date of birth), email, position_ID of each staff.

Create table SQL code:

```
CREATE TABLE `staff_info` (
`ID` int(11) NOT NULL,
`full_name` varchar(45) DEFAULT NULL,
`gender` varchar(5) DEFAULT NULL,
`d.o.b` date DEFAULT NULL,
`email` varchar(80) DEFAULT NULL,
`position_ID` int(11) DEFAULT NULL,
PRIMARY KEY (`ID`));
```

Table information:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
full_name	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	
gender	VARCHAR(5)	☐	☐	☐	☐	☐	☐	☐	☐	
d.o.b	DATE	☐	☐	☐	☐	☐	☐	☐	☐	
email	VARCHAR(80)	☐	☐	☐	☐	☐	☐	☐	☐	
position_ID	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	

5.2. Import data to tables Queries.

After creating tables, we import the data into the table by using INSERT query. This is an example for inserting data into the tables.

- Insert data into table staff_info:

```
1  INSERT INTO staff_info VALUE (1,"Jame Smith","M","1990-10-13","JameSmith@nationlib.edu.vn",3);
2  INSERT INTO staff_info VALUE (2,"Robert William ","M","1999-02-04","RobertWilliam@nationlib.edu.vn",5);
3  INSERT INTO staff_info VALUE (3,"John  Brown","M","1984-05-29","JohnBrown@nationlib.edu.vn",2);
4  INSERT INTO staff_info VALUE (4,"Michael Milles","M","1992-03-11","MichaelMilles@nationlib.edu.vn",5);
5  INSERT INTO staff_info VALUE (5,"David Johns","M","1996-10-31","DavidJohns@nationlib.edu.vn",5);
6  INSERT INTO staff_info VALUE (6,"Mary  Smith","F","1985-12-20","MarySmith@nationlib.edu.vn",1);
7  INSERT INTO staff_info VALUE (7,"Jenifer Garcia","F","2000-08-30","JeniferGarcia@nationlib.edu.vn",3);
8  INSERT INTO staff_info VALUE (8,"Susan  Swift","F","1990-08-08","SusanSwift@nationlib.edu.vn",4);
9  INSERT INTO staff_info VALUE (9,"Lisa Brown ","F","1993-07-13","LisaBrown@nationlib.edu.vn",4);
10 INSERT INTO staff_info VALUE (10,"Emily Thomson","F","1987-06-13","EmilyThomson@nationlib.edu.vn",5);
```

- Insert data into table recommendation_labels:

```
INSERT INTO recommendation_labels VALUE (301,"2 to 8");
INSERT INTO recommendation_labels VALUE (302,"7 to 9");
INSERT INTO recommendation_labels VALUE (303,"9 to 12");
INSERT INTO recommendation_labels VALUE (304,"13 to 18");
INSERT INTO recommendation_labels VALUE (305,"over 18");
```

Since the data importing process is long and repeated for almost every table, we only provide 2 examples here. The remaining process of importing data we have shown in the source code submitted.

- Set Foreign Key for each table:
  For this part, we write all the set foreign key queries in a separate file. This can help us alter or modify the foreign key more efficiently and save time.

  Example of setting Foreign Key for the borrow_detail table:

```
USE librarymanagement;

ALTER TABLE borrow_detail
ADD CONSTRAINT book_data_ISBN_ISBN
FOREIGN KEY (ISBN) REFERENCES book_data(ISBN);

ALTER TABLE borrow_detail
ADD CONSTRAINT customer_info_ID
FOREIGN KEY (customer_ID) REFERENCES customer_info(ID);

ALTER TABLE borrow_detail
ADD CONSTRAINT borrow_status_info_ID
FOREIGN KEY (borrow_status_ID) REFERENCES borrow_status_info(ID);

ALTER TABLE borrow_detail
ADD CONSTRAINT staff_info_ID
FOREIGN KEY (staff_ID) REFERENCES staff_info(ID);
```

5.3. Functions in the library management system

5.3.1. Queries for Customer

- Insert new customer information:
  Example: Insert a new customer with the following personal information

SQL code:

```
-- DROP PROCEDURE IF EXISTS insertNewCustomer;
DELIMITER //
CREATE PROCEDURE insertNewCustomer
(IN newID int(11), newFullName varchar(45), newDOB date, newEmail varchar(70), newGender varchar(5),phone_number bigint(20),address varchar(80))
BEGIN
INSERT INTO customer_info VALUE(newID,newFullName, newDOB, newEmail, newGender,phone_number,address);
SELECT *
FROM customer_info
WHERE
ID = newID;
END
// DELIMITER /
-- Example: Insert a new customer with the following information
CALL insertNewCustomer(201,"Taylor Swift","1990-01-01","TaylorSwift@gmail.com","F",0987654322,"21 Timber Street");
```

Result:

The new row with the information below will be added into the customer_info table

	ID	full_name	date_of_birth	email	gender	phone_number	address
▶	201	Taylor Swift	1990-01-01	TaylorSwift@gmail.com	F	987654322	21 Timber Street

The customer_info table after inserting new customer information

	ID	full_name	date_of_birth	email	gender	phone_number	address
	198	Archie Smith	1996-08-24	ArchieSmith@gmail.com	F	918212157	8785 Ellis Court
	199	Kit Thomas	1985-02-06	KitThomas@gmail.com	F	937880493	1636 Briarview Court
	200	Jonah Robert	2007-10-29	JonahRobert@gmail.com	F	985492716	12 Knox Street
	201	Taylor Swift	1990-01-01	TaylorSwift@gmail.com	F	987654322	21 Timber Street
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Insert new borrow detail information when there is a new customer borrow turn. Example: insert a new borrow detail information with the following data

SQL code:

```
-- DROP PROCEDURE IF EXISTS insertBorrowDetail;
DELIMITER //
CREATE PROCEDURE insertBorrowDetail
(IN newID int(11), newCustomerID int(11), newISBN bigint(20),
newBorrowDate date, newReturnDate date,newDeadline date, newBorrowStatusID int(11), newStaffID int(11))
BEGIN
INSERT INTO borrow_detail VALUE(newID,newCustomerID,newISBN,newBorrowDate,newReturnDate,newDeadline,newBorrowStatusID,newStaffID);
SELECT *
FROM borrow_detail
WHERE
ID = newID;
END
// DELIMITER /
-- Insert a new borrow detail with the following information
CALL insertBorrowDetail(201,35,1189893438,"2022-01-01","2022-01-03","2022-01-14",1,5);
```

Result:

After the procedure, this row of data will be added into the borrow_detail table

ID	customer_ID	ISBN	borrow_date	return_date	deadline_return_date	borrow_status_ID	staff_ID
201	35	1189893438	2022-01-01	2022-01-03	2022-01-14	1	5

The borrow_detail table after modifying with new information:

ID	customer_ID	ISBN	borrow_date	return_date	deadline_return_date	borrow_status_ID	staff_ID
194	38	9113221924	2022-11-29	2022-12-09	2022-12-13	2	4
195	59	2338548212	2022-12-08	2022-12-13	2022-12-22	2	5
196	78	2938095049	2022-12-13	0000-00-00	2022-12-27	1	10
197	4	4690069463	2022-02-09	2022-02-27	2022-02-23	3	2
198	6	5378861196	2022-07-28	2022-07-29	2022-08-11	2	4
199	43	1859059073	2022-05-26	2022-06-04	2022-06-09	2	5
200	46	3939134055	2022-03-21	2022-04-07	2022-04-04	3	10
201	35	1189893438	2022-01-01	2022-01-03	2022-01-14	1	5

- Finding books with specific author's information, for instance author's name "Dan Brown"

SQL code:

```
-- DROP PROCEDURE IF EXISTS FindAuthor
DELIMITER //
CREATE PROCEDURE FindAuthor (IN AuthorName varchar(255))
BEGIN
SELECT *
FROM book_data
WHERE Author = AuthorName;
END
// DELIMITER /
CALL FindAuthor("Dan Brown");
```

Result:

ISBN	Title	Author	Year	No_of_actual_copy	No_of_current_copy	Genre_ID	Shelf_ID	Rating	Recommendation_label_ID
2265831782	Blood Promise	Dan Brown	1998	1	15	16	2	1/5	302
5332178204	The Girl in the Spider's Web	Dan Brown	2017	8	17	18	7	3/5	305
5362863969	Before We Were Yours	Dan Brown	2002	6	15	21	2	3/5	304
9024903371	Crazy Rich Asians	Dan Brown	2013	9	11	26	2	1/5	302

- Finding information of a book name “Little Woman”

SQL code:

```
-- DROP PROCEDURE IF EXISTS FindBook
DELIMITER //
CREATE PROCEDURE FindBook (IN book_name varchar(255))
BEGIN
SELECT *
FROM book_data
WHERE Title = book_name;
END
// DELIMITER /
CALL FindBook("Little Women");
```

Result:

ISBN	Title	Author	Year	No_of_actual_copy	No_of_current_copy	Genre_ID	Shelf_ID	Rating	Recommendation_label_ID
9093434982	Little Women	Charlaine Harris	2003	6	13	1	4	1/5	305

- Finding books published in the year 2000 with descending order of rating.

SQL code:

```
-- DROP PROCEDURE IF EXIST FindBookRating
DELIMITER //
CREATE PROCEDURE FindBookRating (IN book_year int(5))
BEGIN
SELECT *
FROM book_data
WHERE Year = book_year
ORDER BY Rating DESC;
END
// DELIMITER /
CALL FindBookRating(2000);
```

Result:

ISBN	Title	Author	Year	No_of_actual_copy	No_of_current_copy	Genre_ID	Shelf_ID	Rating	Recommendation_label_ID
2882655412	Outliers	Stephen King	2000	7	16	14	5	5/5	304
9480667295	The Wide Window	Janet Evanovich	2000	7	12	5	10	5/5	301
3352131489	The Gunslinger	Marjane Satrapi	2000	2	20	22	8	5/5	304
4268042710	Artemis Fowl and the Lost Colony	Malcolm Gladwell	2000	7	13	8	1	5/5	302
4456378345	Lady Midnight	Meg Cabot	2000	8	15	12	4	5/5	304
4858276620	Atonement	David Sedaris	2000	7	15	7	1	5/5	304
8624303344	House Rules	Jim Butcher	2000	2	20	13	4	4/5	304
3337150874	Snow Crash	Nicholas Sparks	2000	2	16	23	6	4/5	302
3186350163	The Drawing of the Three	Lemony Snicket	2000	1	17	20	7	4/5	301
6268525991	Slammed	Dave Eggers	2000	9	15	12	6	3/5	301
9314744581	Wuthering Heights	Mark Danielewski	2000	9	11	15	10	3/5	303
8671385688	Anne of Green Gables	Sophie Kinsella	2000	5	13	10	1	3/5	303
9538530305	The Awakening	Jodi Picoult	2000	9	14	8	7	3/5	301
8931412133	Ready Player One	Michael Chabon	2000	3	19	6	6	2/5	303
8458699080	The Dark Tower	Lois Lowry	2000	2	19	18	7	2/5	305
6787815834	The Phantom Tollbooth	Margaret Atwood	2000	6	14	24	9	2/5	304
6188469811	Go Ask Alice	Jerry Spinelli	2000	1	19	8	3	2/5	301
9763712445	The Picture of Dorian Gray	Lemony Snicket	2000	10	15	13	9	2/5	304
7054397324	The Storied Life of A.J. Fikry	Lemony Snicket	2000	2	12	18	9	1/5	304
7048074976	Room	Anthony Bourdain	2000	5	11	3	6	1/5	303
5375923802	Are You There, Vodka? It's Me, ...	George R.R. M...	2000	10	19	6	1	1/5	304

- Finding books with individual genre and for an assigned age group: for instance, genre "Science & Technology" with age group 13 to 18 (id 304)

SQL code:

```
-- DROP PROCEDURE IF EXISTS FindGenreAge;
DELIMITER //
CREATE PROCEDURE FindGenreAge (IN book_genre varchar(255),age_group int(5))
BEGIN
SELECT Title,Author,Year,
b.Genre AS "Category",
c.recommendation_label AS "Age group",
No_of_actual_copy,No_of_current_copy,Rating
FROM book_data a
    INNER JOIN genre_info b
    ON a.Genre_ID = b.ID
    INNER JOIN recommendation_labels c
    ON a.Recommendation_label_ID = c.ID
WHERE
b.Genre = book_genre
AND
a.Recommendation_label_ID = age_group
ORDER BY Rating DESC;
END
// DELIMITER /
CALL FindGenreAge("Science  & Technology",304);
```

Result:

Title	Author	Year	Category	Age group	No_of_actual_copy	No_of_current_copy	Rating
1Q84	George Orwell	1949	Science & Technology	13 to 18	1	18	5/5
Passion	Gaston Leroux	1909	Science & Technology	13 to 18	3	15	4/5
The Invention of Wings	Nicholas Sparks	2010	Science & Technology	13 to 18	2	12	4/5
The Devil Wears Prada	Philippa Gregory	2005	Science & Technology	13 to 18	9	15	3/5
One Plus One	Brian Vaughan	2012	Science & Technology	13 to 18	3	16	3/5
Stardust	Samantha Young	2012	Science & Technology	13 to 18	5	13	3/5
Before We Were Yours	Dan Brown	2002	Science & Technology	13 to 18	6	15	3/5
Angela's Ashes	Neil Gaiman	2001	Science & Technology	13 to 18	4	19	2/5
Pet Sematary	Kate Atkinson	2013	Science & Technology	13 to 18	2	12	1/5

- Entering the problem (with problem ID) and receive the information from the staff for help.

Here are some problems with corresponding ID that a customer can have:

ID	Problem	Staff contact information
1	Personal information (wrong information, customer want to change information, …)	Library technician
2	Book problem (book is damage, lost book, …)	Librarian
3	Job and recruitment (customer want to find a job in the library, …)	Manager

For instance, our customers have some problems with personal information on the system.

SQL code:

```
-- DROP PROCEDURE IF EXISTS inputHelp;
DELIMITER //
CREATE PROCEDURE inputHelp (IN helpID int(11))
BEGIN

IF(helpID=1)
THEN
SELECT a.ID,a.full_name,a.email,b.position
FROM staff_info a
    INNER JOIN position b
    ON a.position_ID = b.ID
WHERE b.ID = 3;
END IF;
```

```
IF(helpID=2)
THEN
SELECT a.ID,a.full_name,a.email,b.position
FROM staff_info a
    INNER JOIN position b
    ON a.position_ID = b.ID
WHERE b.ID = 5;
END IF;

IF(helpID=3)
THEN
SELECT a.ID,a.full_name,a.email,b.position
FROM staff_info a
    INNER JOIN position b
    ON a.position_ID = b.ID
WHERE b.ID = 1;
END IF;
END;

// DELIMITER /
CALL inputHelp(1);
```

Result:

ID	full_name	email	position
1	Jame Smith	JameSmith@nationlib.edu.vn	Library Technician
7	Jenifer Garcia	JeniferGarcia@nationlib.edu.vn	Library Technician

Here is the email address of the staff that the customer can contact to help with the problem.

- Checking for the exact deadline to return the book. For instance, the customer with ID 2 wants to check for the deadline to return books.

SQL code:

```
-- DROP PROCEDURE IF EXISTS checkDeadline;
DELIMITER //
CREATE PROCEDURE checkDeadline (IN id int(11))
BEGIN
SELECT *
FROM borrow_detail
WHERE customer_ID = id ;
END
// DELIMITER /
CALL checkDeadline(2);
```

Result:

	ID	customer_ID	ISBN	borrow_date	return_date	deadline_return_date	borrow_status_ID	staff_ID
▶	87	2	9032762091	2022-08-28	2022-09-03	2022-09-11	2	5

Here the customer can see the deadline to return the book on time.

5.3.2. Queries for Staff

- Manager: counting the number of staff in each position.

SQL code:

```
SELECT b.position,COUNT(DISTINCT a.ID) AS "Number of staff"
FROM staff_info a
INNER JOIN position b
ON a.position_ID = b.ID
GROUP BY position_ID;
```

Result:

	position	Number of staff
▶	Manager	1
	Library Assistant	1
	Library Technician	2
	Archivist	2
	Librarian	4

Here the manager can see the detailed number of staff in each position.

- Manager: find the book category that is borrowed the most or find the frequency of borrowing books from each genre.

SQL code:

```
CREATE VIEW Book_genre AS
SELECT a.ISBN,b.Title,c.Genre
FROM borrow_detail a
INNER JOIN book_data b
ON a.ISBN = b.ISBN
INNER JOIN genre_info c
ON b.Genre_ID = c.ID;
SELECT *
FROM Book_genre;

SELECT genre_info.*,
(SELECT COUNT(*) FROM Book_genre WHERE Book_genre.Genre = genre_info.Genre) AS "Frequency"
FROM genre_info
ORDER BY Frequency DESC;
```

Result:

ID	Genre	Frequency
18	Travel	13
2	Science Fiction	13
8	LGBTQ+	12
16	Lifestyle	12
21	Science & Technology	11
7	Historical	11
23	Essay	10
4	Mystery	10
24	Humor	10
6	Thriller	9
13	Biography	8
12	Nonfiction	8
5	Horror	7
22	Religion	7
3	Action & Adventure	7
11	Children	7
15	Health	7
9	Self-help	6
25	Magazine	6
20	Humantities & Social...	6
10	Literary Fiction	6
1	Fantasty	5
14	Food & Drink	4
26	Art	3
19	Guide/How-to	2
17	Contemporary Fiction	0

Here the manager can see the frequency of borrowing of each book genre. As a result, they can find out which book genre is borrowed the most in the library.

- Library Assistant: find all the borrow detail of customer with ID 30

SQL code:

```
-- DROP PROCEDURE IF EXISTS FindCustomer
DELIMITER //
CREATE PROCEDURE FindCustomer (IN customer int(11))
BEGIN
SELECT a.*, b.ISBN,b.borrow_date,b.return_date,c.borrow_status,b.staff_ID
FROM customer_info a
INNER JOIN borrow_detail b
ON a.ID = b.customer_ID
INNER JOIN borrow_status_info c
ON b.borrow_status_ID = c.ID

WHERE a.ID = customer;
END
// DELIMITER /
CALL FindCustomer(30);
```

Result:

ID	full_name	date_of_birth	email	gender	phone_number	address	ISBN	borrow_date	return_date	borrow_status	staff_ID
30	Daphne Taylor	2007-12-11	DaphneTaylor@gmail.com	F	922619708	81 Seaton Place Northwest	7101958601	2022-12-04	2022-12-18	Returned	5
30	Daphne Taylor	2007-12-11	DaphneTaylor@gmail.com	F	922619708	81 Seaton Place Northwest	9538283096	2022-02-22	2022-03-02	Returned	4
30	Daphne Taylor	2007-12-11	DaphneTaylor@gmail.com	F	922619708	81 Seaton Place Northwest	6904079677	2022-05-25	2022-06-14	Returned late	2
30	Daphne Taylor	2007-12-11	DaphneTaylor@gmail.com	F	922619708	81 Seaton Place Northwest	4798484335	2022-09-14	2022-09-30	Returned late	5

Here, the library assistant can see the borrowing details of customer with ID 30, together with detailed information of that customer.

- Library Assistant: Delete a specific borrow detail.
  Example: delete a borrow detail with ID = 201

SQL code:

```
DELIMITER //
CREATE PROCEDURE deleteBorrowDetail(IN deleteID int(11))
BEGIN
DELETE FROM borrow_detail
WHERE ID = deleteID;
END
// DELIMITER /

-- Example: Delete a borrow detail with ID 201
CALL deleteBorrowDetail(201);
```

Result:

After running the procedure, the borrow detail with ID = 201 has been successfully deleted from the borrow_detail table.

The table after deletion, there is no borrow detail row of ID = 201

ID	customer_ID	ISBN	borrow_date	return_date	deadline_return_date	borrow_status_ID	staff_ID
197	4	4690069463	2022-02-09	2022-02-27	2022-02-23	3	2
198	6	5378861196	2022-07-28	2022-07-29	2022-08-11	2	4
199	43	1859059073	2022-05-26	2022-06-04	2022-06-09	2	5
200	46	3939134055	2022-03-21	2022-04-07	2022-04-04	3	10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Library Assistant: delete a customer.
  Example: delete a customer with the ID = 201

SQL code:

```
-- DROP PROCEDURE IF EXISTS deleteCustomer;
DELIMITER //
CREATE PROCEDURE deleteCustomer(IN deleteID int(11))
BEGIN
DELETE FROM customer_info
WHERE ID = deleteID;
END
// DELIMITER /
-- Example: Delete a customer with ID 201
CALL deleteCustomer(201);
```

Result:

After running the procedure, the customer with ID = 201 has been successfully delete from the customer_info table

The table after deletion, there is no customer with ID = 201;

ID	full_name	date_of_birth	email	gender	phone_number	address
197	Dylan Brown	2006-06-04	DylanBrown@gmail.com	F	940793466	58 North U.S.A Drive
198	Archie Smith	1996-08-24	ArchieSmith@gmail.com	F	918212157	8785 Ellis Court
199	Kit Thomas	1985-02-06	KitThomas@gmail.com	F	937880493	1636 Briarview Court
200	Jonah Robert	2007-10-29	JonahRobert@gmail.com	F	985492716	12 Knox Street
NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Archivist: Counting the number of books for each genre.

SQL code:

```
SELECT b.Genre, COUNT(Genre_ID) AS "Number of books"
FROM book_data a
INNER JOIN genre_info b
ON a.Genre_ID = b.ID
GROUP BY Genre_ID;
```

Result:

Genre	Number of books
Fantasty	26
Science Fiction	23
Action & Adventure	43
Mystery	36
Horror	36
Thriller	41
Historical	42
LGBTQ+	52
Self-help	39
Literary Fiction	33
Children	44
Nonfiction	39
Biography	40
Food & Drink	36
Health	40
Lifestyle	39
Contemporary Fi...	35
Travel	37
Guide/How-to	35
Humantities & So...	35
Science & Techn...	44
Religion	35
Essay	40
Humor	50
Art	43

Here the archivist can see all the number of books of each book genre in the library.

- Archivist: Insert a new book data

SQL code:

```
-- DROP PROCEDURE IF EXISTS insertNewBook;
DELIMITER //
CREATE PROCEDURE insertNewBook
(IN newISBN bigint(20),newTitle varchar(100), newAuthor varchar(45), newYear year(4), newNo_of_actual_copy int(11), newNo_of_current_copy int(11),
newGenre_ID int(11), newShelf_ID int(11), newRating varchar(10), newRecommendation_label_ID int(11))
BEGIN
INSERT INTO book_data VALUE(newISBN,newTitle, newAuthor, newYear, newNo_of_actual_copy, newNo_of_current_copy,
newGenre_ID , newShelf_ID , newRating , newRecommendation_label_ID );
SELECT *
FROM book_data
WHERE
ISBN = newISBN;
END
// DELIMITER /
-- Example: Insert a new book with the following information
CALL insertNewBook(1234567890,"New Book","New Author",1999,10,20,4,1,"5/5",305);
```

Result:

After running the procedure, the following column will be added into the table book_data:

ISBN	Title	Author	Year	No_of_actual_copy	No_of_current_copy	Genre_ID	Shelf_ID	Rating	Recommendation_label_ID
1234567890	New Book	New Author	1999	10	20	4	1	5/5	305

The book_data table after inserting new book data information:

ISBN	Title	Author	Year	No_of_actual_copy	No_of_current_copy	Genre_ID	Shelf_ID	Rating	Recommendation_label_ID
1235545313	The Ruins of Gorlan	L.M. Montgomery	1909	9	19	26	9	2/5	301
1234567890	New Book	New Author	1999	10	20	4	1	5/5	305
1228297094	The Lincoln Lawyer	Ally Condie	2012	6	12	18	8	2/5	305
1224407867	A Long Way Gone: Memo…	Khaled Hosseini	2007	4	20	24	1	4/5	304
1219528960	So Long, and Thanks for …	Dave Eggers	2013	1	11	6	6	5/5	301

- Archivist: Delete a book data with specific ISBN

SQL code:

```sql
-- DROP PROCEDURE IF EXISTS deleteBook;

DELIMITER //
CREATE PROCEDURE deleteBook(IN deleteID bigint(20))
BEGIN
DELETE FROM book_data
WHERE ISBN = deleteID;
END
// DELIMITER /

-- Example: Delete a book with ISBN 1234567890
CALL deleteBook(1234567890);
```

Result:

After running the procedure, the book with corresponding ISBN code has been deleted successfully from the book_data table:

- Librarian: displaying information of borrowers who return books late.

SQL code:

```sql
SELECT c.*,a.ISBN,b.borrow_status
FROM borrow_detail a
    INNER JOIN borrow_status_info b
    ON a.borrow_status_ID = b.ID
    INNER JOIN customer_info c
    ON a.customer_ID = c.ID
WHERE a.borrow_status_ID = 3
ORDER BY c.full_name;
```

Result:

ID	full_name	date_of_birth	email	gender	phone_number	address	ISBN	borrow_status
59	Adeline Antony	1994-06-27	AdelineAntony@gmail.com	F	936419222	125 John Street	6295529997	Returned late
13	Alice Antony	1983-01-23	AliceAntony@gmail.com	M	955950689	87 Horseshoe Drive	4181819668	Returned late
13	Alice Antony	1983-01-23	AliceAntony@gmail.com	M	955950689	87 Horseshoe Drive	6771460349	Returned late
20	Amelia Smith	2004-11-20	AmeliaSmith@gmail.com	M	916233813	22572 Toreador Drive	3541303144	Returned late
38	Arabella Holmes	2008-06-28	ArabellaHolmes@gmail.com	F	989779411	5928 West Mauna Loa Lane	9559972831	Returned late
38	Arabella Holmes	2008-06-28	ArabellaHolmes@gmail.com	F	989779411	5928 West Mauna Loa Lane	3383509355	Returned late
23	Aria Taylor	1983-12-01	AriaTaylor@gmail.com	M	914344722	5114 Greentree Drive	6825653845	Returned late
19	Astrid Brown	1997-10-24	AstridBrown@gmail.com	F	945496680	629 Debbie Drive	9962961560	Returned late
19	Astrid Brown	1997-10-24	AstridBrown@gmail.com	F	945496680	629 Debbie Drive	1180986033	Returned late
19	Astrid Brown	1997-10-24	AstridBrown@gmail.com	F	945496680	629 Debbie Drive	8626603457	Returned late
62	Athena Carter	2000-11-23	AthenaCarter@gmail.com	M	991175328	491 Arabian Way	7769368591	Returned late
64	Audrey Carter	2004-03-15	AudreyCarter@gmail.com	M	997479125	80 North East Street	9209482328	Returned late
9	Aurora Williams	1991-08-12	AuroraWilliams@gmail.com	M	959248837	5403 Illinois Avenue	7856146866	Returned late
9	Aurora Williams	1991-08-12	AuroraWilliams@gmail.com	M	959248837	5403 Illinois Avenue	1204407841	Returned late
3	Ava Brown	1997-05-22	AvaBrown@gmail.com	F	945596426	560 Penstock Drive	5323732223	Returned late
63	Cecilia Holmes	1998-02-09	CeciliaHolmes@gmail.com	M	981574445	12245 West 71st Place	2080060782	Returned late
79	Celeste Watson	2006-05-02	CelesteWatson@gmail.com	M	984482774	270 Chrissy's Court	3074408448	Returned late
22	Charlotte Robert	1991-02-04	CharlotteRobert@gmail.com	M	928570524	3729 East Mission Boulevard	1361714268	Returned late
22	Charlotte Robert	1991-02-04	CharlotteRobert@gmail.com	M	928570524	3729 East Mission Boulevard	2570802257	Returned late
61	Chloe Holmes	1998-08-03	ChloeHolmes@gmail.com	F	965334190	8376 Albacore Drive	3750840512	Returned late
68	Claire Robert	1991-06-22	ClaireRobert@gmail.com	M	921461248	132 Laurel Green Court	5673698454	Returned late
32	Clara Williams	1988-06-20	ClaraWilliams@gmail.com	F	999386743	7431 Candace Way	3538029428	Returned late
30	Daphne Taylor	2007-12-11	DaphneTaylor@gmail.com	F	922619708	81 Seaton Place Northwest	4798484335	Returned late
30	Daphne Taylor	2007-12-11	DaphneTaylor@gmail.com	F	922619708	81 Seaton Place Northwest	6904079677	Returned late
74	Delilah Thomas	2003-02-22	DelilahThomas@gmail.com	F	997906162	232 Maine Avenue	5338401237	Returned late
74	Delilah Thomas	2003-02-22	DelilahThomas@gmail.com	F	997906162	232 Maine Avenue	2376863907	Returned late

ID	full_name	date_of_birth	email	gender	phone_number	address	ISBN	borrow_status
74	Delilah Thomas	2003-02-22	DelilahThomas@gmail.com	F	997906162	232 Maine Avenue	2376863907	Returned late
75	Elena Robert	1998-02-09	ElenaRobert@gmail.com	F	950718187	1 Kempf Drive	9713125748	Returned late
73	Elizabeth Smith	2000-04-08	ElizabethSmith@gmail.com	F	940394353	700 Winston Place	3750840512	Returned late
24	Elodie Holmes	1988-02-22	ElodieHolmes@gmail.com	F	933560436	3466 Southview Avenue	6020124295	Returned late
1	Eloise Holmes	2008-09-09	EloiseHolmes@gmail.com	F	914645144	1745 T Street Southeast	9962961560	Returned late
1	Eloise Holmes	2008-09-09	EloiseHolmes@gmail.com	F	914645144	1745 T Street Southeast	2466235799	Returned late
6	Esme Robert	1994-01-08	EsmeRobert@gmail.com	M	987441414	18 Densmore Drive	4960195768	Returned late
51	Evangeline Tho...	1986-09-24	EvangelineThomas@gmail....	M	949718437	1508 Massachusetts Aven...	7073474321	Returned late
31	Evelyn Lewis	1982-03-21	EvelynLewis@gmail.com	F	928443991	1267 Martin Street	7407018079	Returned late
58	Florence Hill	2003-12-08	FlorenceHill@gmail.com	F	951668219	159 Downey Drive	9103210851	Returned late
18	Hazel Carter	1991-09-24	HazelCarter@gmail.com	F	937405338	5461 West Shades Valley ...	1800858125	Returned late
14	Iris Hall	1990-01-22	IrisHall@gmail.com	M	914761335	60 Desousa Drive	1869071256	Returned late
14	Iris Hall	1990-01-22	IrisHall@gmail.com	M	914761335	60 Desousa Drive	4960195768	Returned late
15	Ivy Holmes	1998-10-31	IvyHolmes@gmail.com	F	997464750	4 Old Colony Way	8282968625	Returned late
15	Ivy Holmes	1998-10-31	IvyHolmes@gmail.com	F	997464750	4 Old Colony Way	1365316311	Returned late
43	Josephine Smith	1981-11-25	JosephineSmith@gmail.com	F	986715211	388 East Main Street	3321937169	Returned late
12	Luna Hill	1980-05-22	LunaHill@gmail.com	M	948456219	6463 Vrain Street	1325271378	Returned late
12	Luna Hill	1980-05-22	LunaHill@gmail.com	M	948456219	6463 Vrain Street	8776228567	Returned late
12	Luna Hill	1980-05-22	LunaHill@gmail.com	M	948456219	6463 Vrain Street	5438955642	Returned late
46	Lyra Taylor	1985-01-06	LyraTaylor@gmail.com	F	978681288	308 Woodleaf Court	8682903531	Returned late
46	Lyra Taylor	1985-01-06	LyraTaylor@gmail.com	F	978681288	308 Woodleaf Court	8624303344	Returned late
46	Lyra Taylor	1985-01-06	LyraTaylor@gmail.com	F	978681288	308 Woodleaf Court	3939134055	Returned late
52	Mae Robert	2000-09-09	MaeRobert@gmail.com	F	915667214	5615 West Villa Maria Drive	8626603457	Returned late
52	Mae Robert	2000-09-09	MaeRobert@gmail.com	F	915667214	5615 West Villa Maria Drive	4986331197	Returned late
5	Maeve Thomas	1994-04-24	MaeveThomas@gmail.com	F	989169717	2721 Lindsay Avenue	3490546171	Returned late

Here the librarian can see the information of customers who return books late, with the order name from A to Z.

- Librarian: updating the current shelf information for corresponding books.

For instance, the librarian wants to update the book name “The Wind” to shelf number 6.

SQL code:

```
-- DROP PROCEDURE IF EXISTS updateShelf;
DELIMITER //
CREATE PROCEDURE updateShelf (IN bookName varchar(255),shelfNew int(5))
BEGIN
UPDATE book_data
SET Shelf_ID = shelfNew
WHERE Title = bookName;

SELECT ISBN,Title,Author,Year,Shelf_ID
FROM book_data
WHERE Title = bookName;
END;
// DELIMITER /
CALL updateShelf("The Wind", 6);
```

Result:

	ISBN	Title	Author	Year	Shelf_ID
▶	1397207043	The Wind	Emmuska Orczy	1905	6

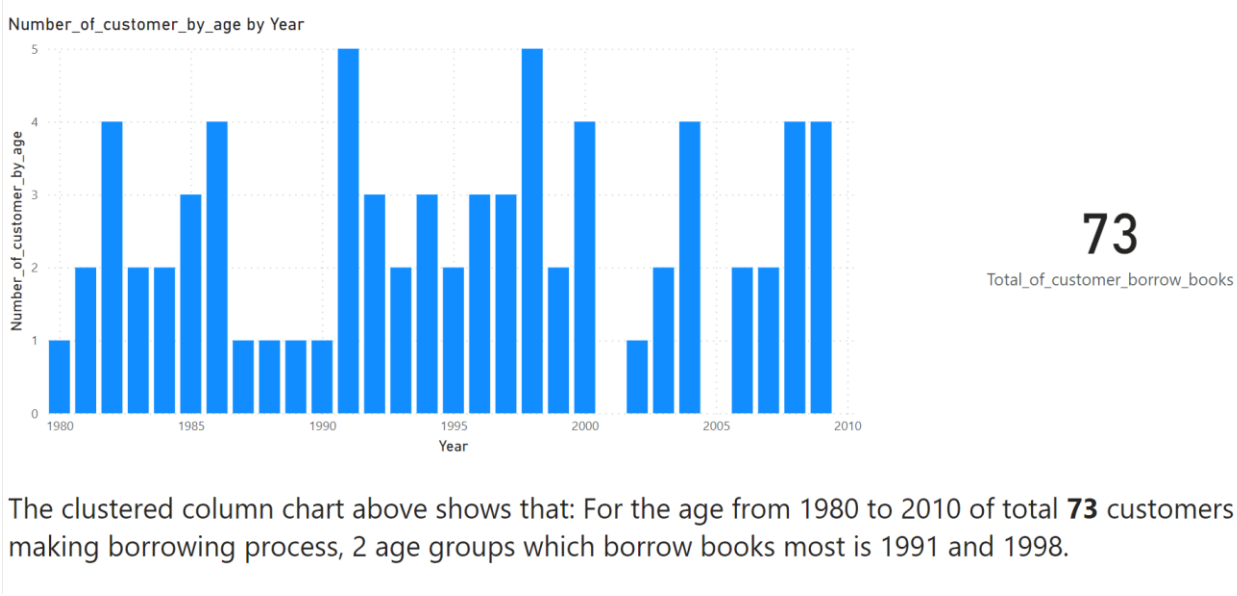
Here the librarian can see the new position of the book.

6. Visualization

In this part, we will illustrate some data using the *Power BI* tool. These visualizations can hep the manager or the staff for their job, especially in creating reports.

6.1. Customer visualization

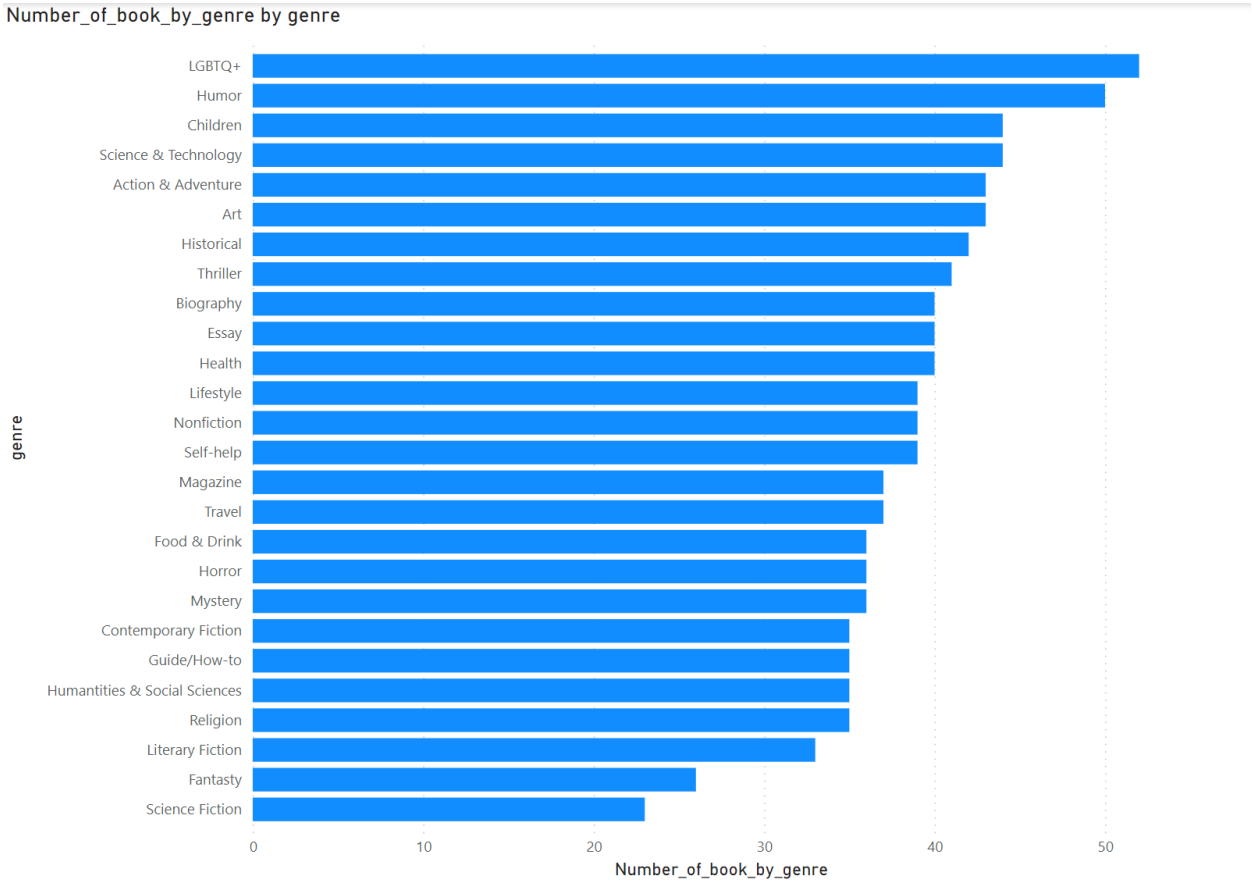
- Number of customers by age:



The clustered column chart above shows that: For the age from 1980 to 2010 of total **73** customers making borrowing process, 2 age groups which borrow books most is 1991 and 1998.

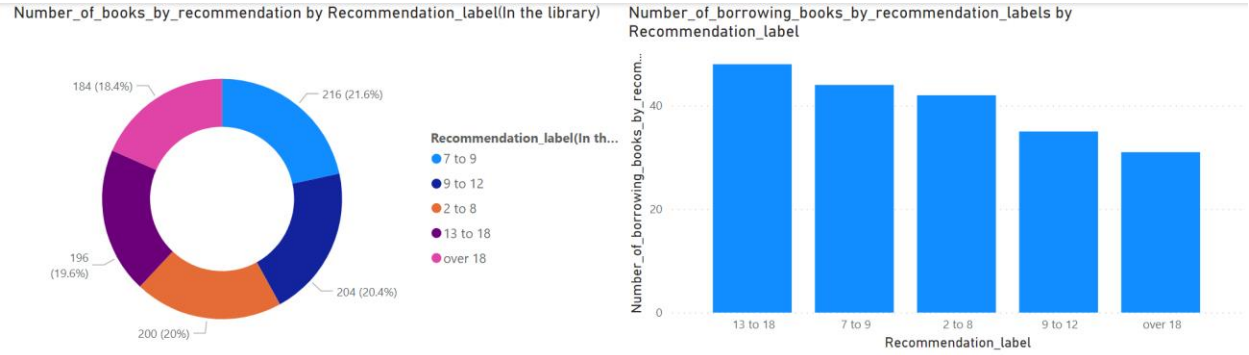
6.2. Book visualization

- Number of books by genre:



The favorite genre of books in library is **LGBTQ+** and **Humor**.While **Science Fiction** is the kind of genre having lowest number of books.

- Number of borrowed books by recommendation labels

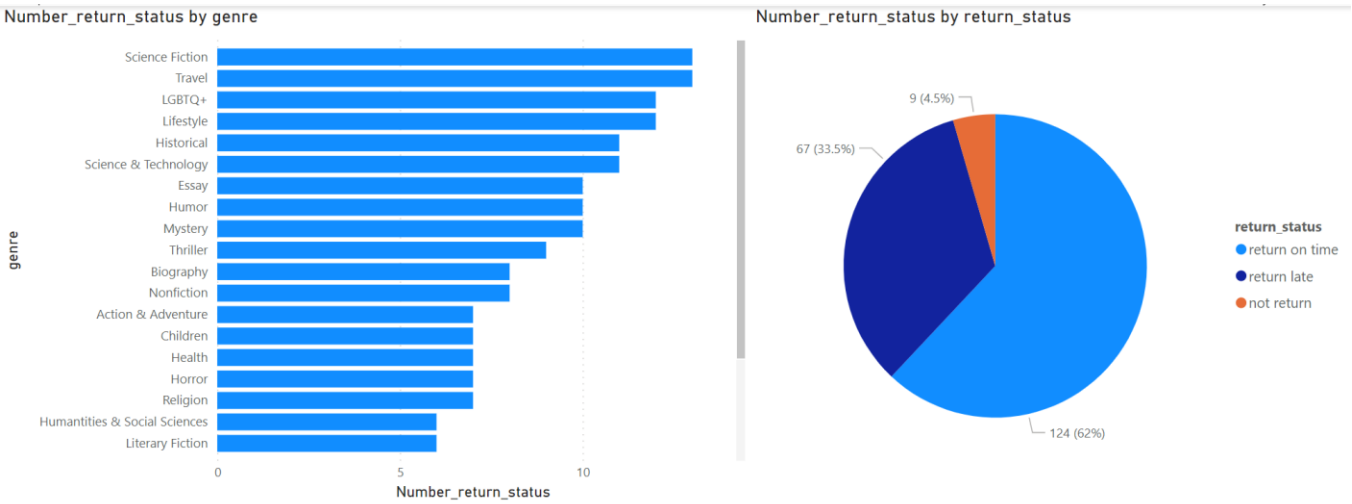


Following 2 chart above, there are some differences between the number of books **in library** and number of books which **is borrowed** by recommendation labels:

. The number of **(13 to 18)** books in the library just account for 19.6% total of books, but it's the type having highest number of borrowing.

. The number of **(9 to 12)** books take the second place with 20.4% in the library but it's the second type of book having lowest borrowers.

- Return status observation.



Comment:

. The pie chart show the rate of each return status:
- Return on time account the most percentage.
- Return late status need to be decrease.

. Science Fiction and Travel are 2 type of genre which is often returned late.

7. Reference

Book data source: Link

Address data source: Link

Book genre data source: Link

Recommendation label data source: Link