

**PES UNIVERSITY**  
**Department of Computer Science**  
**& Engineering**



**DBMS - UE20CS301**

**Mini Project**

**Library Management System**

**Submitted to:**  
Dr. Geetha D  
Associate Professor

**Submitted By:**  
Name: Vishwas M  
SRN: PES2UG20CS390  
Semester: 5  
Section: F

## Table of Contents

<b>Sl.No</b>	<b>Title</b>	<b>Page No</b>
1	Short Description and Scope of the Project	3
2	Entity Relationship Diagram	4
3	Relational Schema	5
4	DDL Statements-Building the Database	6
5	Populating the Database	8
6	Join Queries	14
7	Attribute Functions	17
8	Set Operations	20
9	Functions and Procedures	23
10	Triggers and Cursors	26
11	Frontend Application	31
12	Conclusion	63
13	References	64

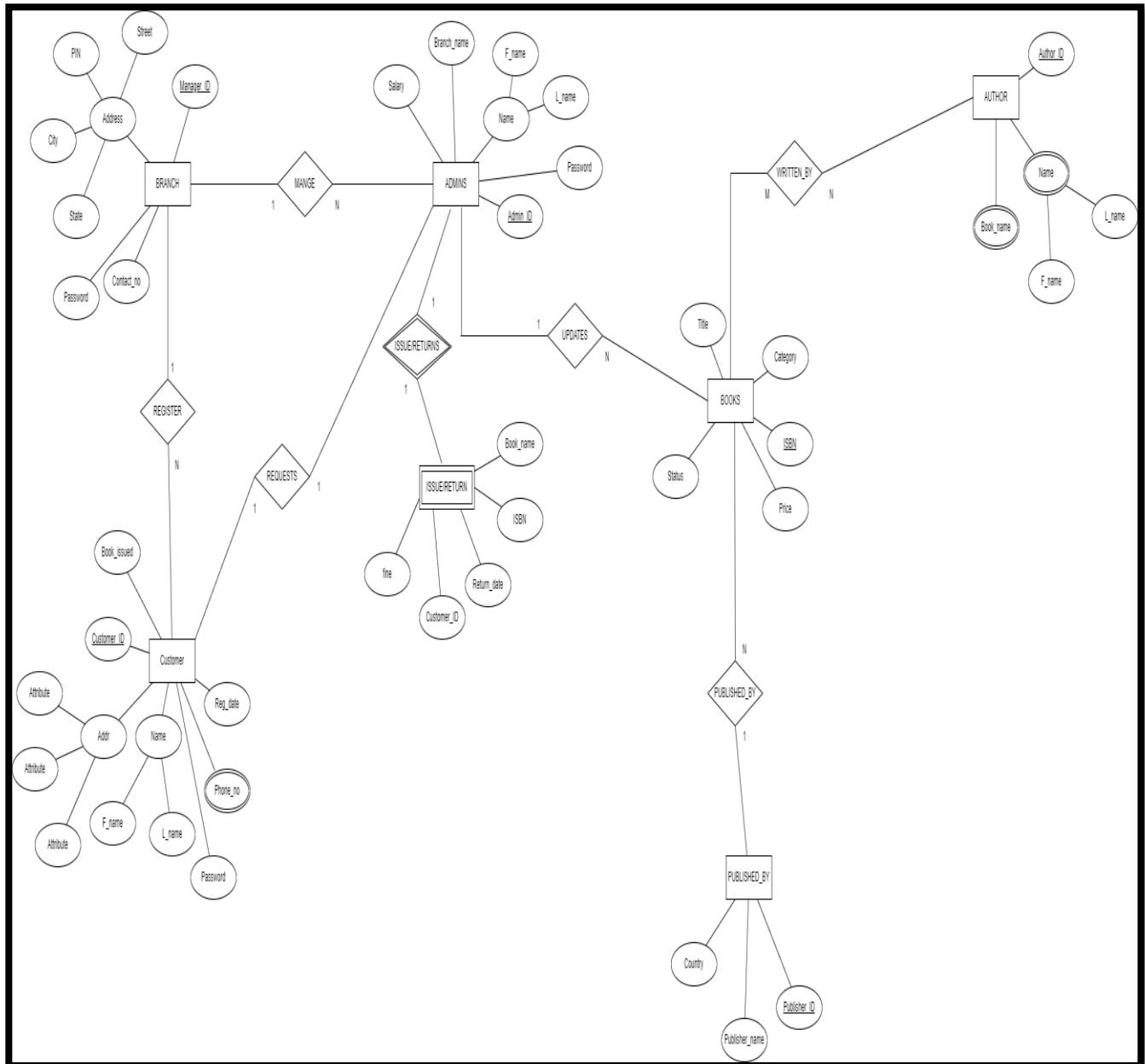
## **1. Short Description and Scope of the Project**

Library Management project is a computer-based system. It reduces human made errors and increases the efficiency. The main objective of this library management system project is to reduce human efforts and time. The maintenance of the records is made easy and all the records are stored in the SQL database which can be retrieved easily after words. Suitable navigation control is given in all the forms to navigate through the records in this project.

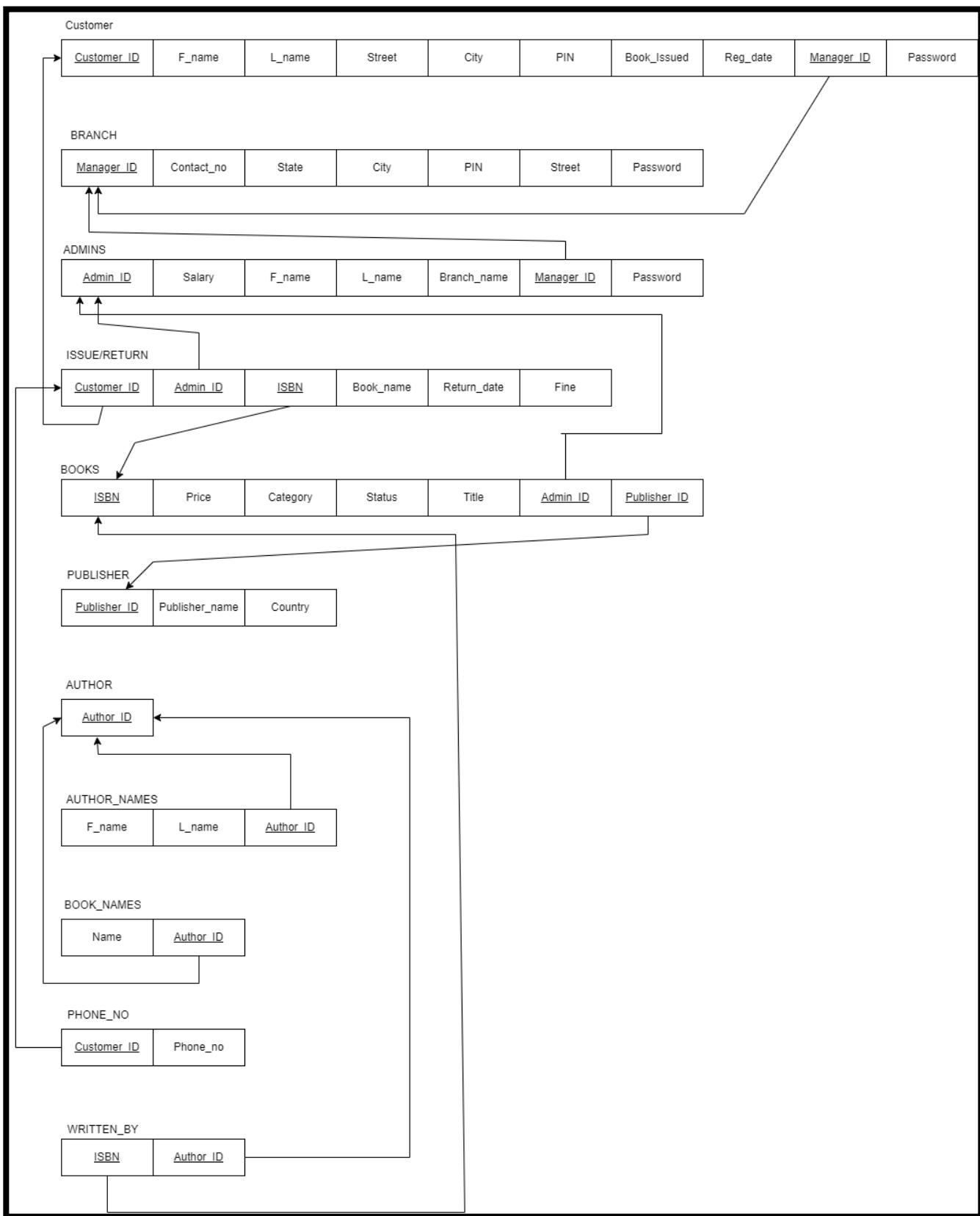
By using Python3(Streamlit) as frontend and keeping MySQL as the backend I am creating library database system. There will be several tables created inside the database which will represent each entity that I am going to create. These entities are related to each other by relationships and are mapped by cardinality ratios. The primary keys and the foreign keys are related to each other wherever it is necessary. All the constraints that a database should follow and the constraints that are applicable to the users are also are maintained properly.

If the number of records is very large then user has to simply type in the search string and he will get the results immediately. The editing is also made simpler. The user has to just type in the required field and press the update button to update the required field. The Books and members are given a unique id no. So that they can be accessed without any error. Our main objective of this project is to get the correct data about a particular student and books available in the library. Even the library admin can access the information about the user and can check about the books issued by him/her. Librarian can also see the information about the issue and due date of a particular book issued by the user.

## 2. ER Diagram



### 3. Relational Schema



## 4. DDL statements - Building the database

### 1) BRANCH:

```
CREATE table branch( manager_id int, contact_no bigint, state  
varchar(255), city varchar(255) ,street varchar(255), pin int,password  
varchar(20), primary key(manager_id));
```

### 2) CUSTOMER:

```
Create table customer ( customer_id int, f_name varchar(255), l_name  
varchar(255), street varchar(255), city varchar(255), pin int, reg_date  
date, manager_id int, password varchar(20), primary key(customer_id),  
foreign key(manager_id) references branch(manager_id));
```

### 3) ADMIN:

```
Create table admin (admin_id int, salary int, f_name varchar(255), l_name  
varchar(255), manager_id int,password varchar(20), primary  
key(admin_id), foreign key(manager_id) references branch(manager_id));
```

### 4) PUBLISHER:

```
Create table publisher (publisher_id int, publisher_name varchar(255),  
country varchar(255), primary key(publisher_id));
```

### 5) BOOKS:

```
Create table books ( ISBN varchar(255), price int, category varchar(255),  
status varchar(255), title varchar(255), admin_id int, publisher_id int,  
primary key(ISBN), foreign key(admin_id) references admin(admin_id),  
foreign key(publisher_id) references publisher(publisher_id));
```

### 6) ISSUE\_RETURN:

```
Create table issue_return (customer_id int, admin_id int, ISBN  
varchar(255), issue_date date, return_date date, fine int, foreign  
key(customer_id) references customer(customer_id),foreign  
key(admin_id) references admin(admin_id), foreign key(ISBN) references  
books(ISBN));
```

**7) AUTHOR:**

Create table author (author\_id int, primary key(author\_id));

**8) AUTHOR\_NAMES:**

Create table author\_names (f\_name varchar(255), l\_name varchar(255), author\_id int, foreign key(author\_id) references author(author\_id));

**9) BOOK\_NAMES:**

Create table book\_names (name varchar(255), author\_id int, foreign key(author\_id) references author(author\_id));

**10) PHONE\_NO:**

Create table phone\_no (customer\_id int, number int, foreign key(customer\_id) references customer(customer\_id));

**11) WRITTEN\_BY:**

Create table written\_by (ISBN varchar(255), author\_id int, foreign key(ISBN) references books(ISBN), foreign key(author\_id) references author(author\_id));

## 5. Populating the Database

1)

```
INSERT INTO admins (`admin_id`, `salary`, `f_name`, `l_name`,  
`manager_id`) VALUES  
(1212, 30000, 'mohammad', 'shami', 5, 'shami'),  
(1313, 30000, 'yuzvendra', 'chahal', 5, 'chahal'),  
(2323, 30000, 'rohit', 'sharma', 1, 'sharma'),  
(3434, 30000, 'lokesh', 'rahul', 2, 'rahul'),  
(4433, 30000, 'avesh', 'khan', 2, 'khan'),  
(4545, 30000, 'suryakumar', 'yadav', 1, 'yadav'),  
(5656, 30000, 'hardik', 'pandya', 4, 'pandya'),  
(6767, 30000, 'ishaan', 'kishan', 5, 'kishan'),  
(7878, 30000, 'ravindra', 'jadeja', 3, 'jadeja'),  
(8989, 30000, 'dinesh', 'karthik', 3, 'karthik'),  
(9090, 30000, 'jasprit', 'bumrah', 4, 'bumrah');
```

2)

```
INSERT INTO author (`author_id`) VALUES  
(11),(22),(33),(44),(55),(66),(77),(88),(99),(109),(119),(129),(139),(149),(159)  
(169),(179),(189),(199),(209),(219),(229),(239),(249),(259),(269),(279),(289),  
(299), (309);
```

3)

```
INSERT INTO books (`ISBN`, `price`, `category`, `status`, `title`,  
`admin_id`, `publisher_id`) VALUES  
('b007', 1390, 'kids', 'yes', 'the famous five', 7878, 2000),  
('b037', 1232, 'kids', 'yes', 'charlie and the chocolate factory', 4545, 3000),  
('b133', 789, 'kids', 'yes', 'james and the giant peach', 9090, 1000),  
('b152', 1000, 'novel', 'yes', 'The lord of the rings', 2323, 3000),  
('b155', 1800, 'kids', 'yes', 'matilda', 5656, 2000),  
('b209', 1500, 'kids', 'yes', 'the bfg', 1313, 1000),  
('b222', 289, 'novel', 'yes', 'robinson crusoe', 4545, 1000),
```

('b231', 333, 'academics', 'yes', 'fundamentals of database systems', 2323, 1000),  
('b232', 599, 'novel', 'yes', 'percy jackson and the sea of monsters', 1313, 2000),  
('b234', 1099, 'novel', 'yes', 'raavan:the enemy of aryavarta', 1212, 3000),  
('b324', 3432, 'academics', 'no', 'computer network security', 2323, 1000),  
('b374', 223, 'fiction', 'yes', 'percy jackson: the lightning thief', 1212, 4000),  
('b428', 756, 'fiction', 'no', 'goosebumps', 5656, 3000),  
('b436', 1543, 'novel', 'yes', 'ram:scion of ikshvaku', 1212, 1000),  
('b452', 476, 'fiction', 'yes', 'The secret of nagas', 2323, 4000),  
('b521', 999, 'academics', 'yes', 'the design of everyday things', 4433, 3000),  
('b545', 1234, 'novel', 'no', 'the great gatsby', 6767, 3000),  
('b565', 1546, 'academics', 'no', 'artificial intelligence', 5656, 4000),  
('b667', 1444, 'novel', 'yes', 'the immortals of meluha', 4433, 4000),  
('b723', 899, 'novel', 'yes', 'the oath of vayuputras', 7878, 2000),  
('b760', 1233, 'academics', 'yes', 'intoduction to modern cryptography', 7878, 4000),  
('b767', 1555, 'academics', 'yes', 'computer and internet security', 9090, 2000),  
('b769', 2222, 'novel', 'yes', 'sita:warrior of mithila', 2323, 2000),  
('b774', 243, 'academics', 'yes', 'concepts of programming languages', 2323, 4000),  
('b777', 454, 'novel', 'yes', 'wuthering heights', 6767, 2000),  
('b826', 222, 'fiction', 'no', 'harry potter and the chamber of secrets', 4545, 1000),  
('b908', 1800, 'academics', 'no', 'machine learning', 8989, 1000),  
('b986', 543, 'fiction', 'no', 'revolution 2020', 3434, 2000),  
('b998', 2132, 'kids', 'yes', 'the adventures of tintin', 8989, 4000),  
('b999', 1090, 'novel', 'yes', 'the 3 mistakes of my life', 1212, 3000);

**4)**

```
INSERT INTO branch (`manager_id`, `contact_no`, `state`, `city`, `street`,  
`pin`) VALUES  
(1, 9736457263, 'karnataka', 'bangalore', 'church street', 635487),  
(2, 9867198987, 'andhra pradesh', 'amaravati', 'reddy street', 563726),  
(3, 9487255664, 'tamil nadu', 'chennai', 'amma street', 978878),  
(4, 9188769897, 'maharashtra', 'mumbai', 'linking road', 263874),  
(5, 9235547563, 'gujarat', 'ahmedabad', 'gandhi road', 162534);
```

**5)**

```
INSERT INTO customer (`customer_id`, `f_name`, `l_name`, `street`, `city`,  
`pin`, `reg_date`, `manager_id`) VALUES  
(11111, 'vishwas', 'm', '2', 'udipi', 182432, '2022-10-12', 1, 'vishwas'),  
(12345, 'vismaya', 'r', '45', 'mysore', 143678, '2022-10-04', 1, 'vismaya'),  
(13234, 'tushar', 'bothra', '44', 'mumbai', 369061, '2022-10-06', 5, 'tushar'),  
(15432, 'tenzin', 'tsephel', '65', 'hubli', 909090, '2022-10-05', 3, 'tenzin'),  
(18273, 'sheldon', 'cooper', '46', 'texas', 756735, '2022-10-12', 3, 'sheldon'),  
(22222, 'raghav', 'loknath', '6', 'salem', 543234, '2022-10-03', 3, 'raghav'),  
(33333, 'vishnudeep', 'mysore', '45', 'mysore', 567543, '2022-10-15', 5,  
'vishnudeep'),  
(44444, 'sai', 'rethwik', '7', 'marathalli', 248554, '2022-10-16', 4, 'sai'),  
(46384, 'vidip', 'chabhra', '28', 'jaipur', 821432, '2022-10-10', 5, 'vidip'),  
(55555, 'vishwa', 'mehta', '1', 'haralur', 268543, '2022-10-11', 3, 'vishwa'),  
(56434, 'jennifer', 'aniston', '12', 'los angeles', 936545, '2022-10-10', 2,  
'jennifer'),  
(56453, 'mike', 'ross', '41', 'new york', 152434, '2022-10-05', 4, 'mike'),  
(65432, 'bavanika', 'v', '19', 'bangalore', 825378, '2022-10-07', 4,  
'bavanika'),  
(66666, 'vinti', 'agrawal', '34', 'nepal', 275325, '0000-00-00', 2, 'vinti'),  
(67890, 'srushti', 's', '85', 'bangalore', 629078, '2022-10-04', 4, 'srushti'),  
(77777, 'vishal', 'elangovan', '67', 'chennai', 873456, '2022-10-09', 2,  
'vishal'),  
(87456, 'shuvam', 'bose', '73', 'begur', 456234, '2022-10-14', 2, 'shuvam'),  
(88888, 'tushar', 'bhat', '98', 'mangalore', 813456, '2022-10-09', 4, 'tushar'),  
(99999, 'sahith', 'reddy', '11', 'nandyal', 659766, '2022-10-02', 3, 'sahith');
```

6)

```
INSERT INTO issue_return (`customer_id`, `admin_id`, `ISBN`,  
    `issue_date`, `return_date`, `fine`) VALUES  
(11111, 2323, 'b222', '2022-10-12', '2022-10-19',0),  
(11111, 4545, 'b986', '2022-10-12', '2022-10-19',0),  
(22222, 7878, 'b152', '2022-10-03', '2022-10-10',0),  
(22222, 8989, 'b565', '2022-10-03', '2022-10-10',0),  
(33333, 6767, 'b826', '2022-10-14', '2022-10-21',0),  
(44444, 5656, 'b777', '2022-10-05', '2022-10-12',0),  
(55555, 8989, 'b428', '2022-10-08', '2022-10-15',0),  
(66666, 4433, 'b452', '2022-10-12', '2022-10-19',0),  
(66666, 3434, 'b908', '2022-10-12', '2022-10-19',0),  
(77777, 4433, 'b007', '2022-10-18', '2022-10-25',0),  
(87456, 3434, 'b545', '2022-10-26', '2022-11-02',0),  
(88888, 9090, 'b999', '2022-10-11', '2022-10-18',0),  
(13234, 1212, 'b324', '2022-10-01', '2022-10-08',0),  
(56453, 9090, 'b767', '2022-10-28', '2022-11-04',0),  
    (18273, 8989, 'b037', '2022-10-29', '2022-11-05',0);
```

7)

```
INSERT INTO phone_no (`customer_id`, `number`) VALUES  
(11111, 2147483647),(22222, 2147483647),(33333, 2147483647),  
(44444, 2147483647),(55555, 2147483647),(66666, 2147483647),  
(77777, 2147483647),(88888, 2147483647),(99999, 2147483647),  
(12345, 2147483647),(67890, 2147483647),(13234, 2147483647),  
(15432, 2147483647),(65432, 2147483647),(87456, 2147483647),  
(46384, 2147483647),(56453, 2147483647),(18273, 2147483647),  
(56434, 2147483647),(11111, 2147483647),(22222, 2147483647);
```

**8)**

```
INSERT INTO publisher (`publisher_id`, `publisher_name`, `country`)
VALUES
(1000, 'pearson', 'england'),(2000, 'macmillan', 'usa'),
(3000, 'puffin books', 'united kingdom'),
(4000, 'penguin random house', 'india');
```

**9)**

```
INSERT INTO author_names (`f_name`, `l_name`, `author_id`) VALUES
('Daniel', 'Defoe', 11),('Chetan ', 'Bhagat', 22),('J.R.R', 'Tolkien', 33),
('Peter', 'Norwig', 44),('Stuart', 'Russel', 55),('J K', 'Rowling', 66),
('Emily', 'Bronte', 77),('R L', 'Stine', 88),('Amish', 'Tripathi', 99),
('Tom', 'Mitchell', 109),('Enid', 'Blyton', 119),('Scott', 'Fitzgerald', 129),
('Georges', 'Remi', 139),('Joseph', 'kizza', 149),('Wenliang ', 'Du', 159),
('Roald ', 'Dahl', 169),('Rick', 'Riordan', 179),('Ramez', 'Elamsri', 189),
('Jonathan', 'Katz', 199),('Yehuda', 'Lindell', 209),('Don', 'Norman', 219),
('Robert', 'Sibesta', 229);
```

**10)**

```
INSERT INTO book_names (`name`, `author_id`) VALUES
('robinson crusoe', 11),('revolution 2020', 22),('The lord of the rings', 33),
('artificial intelligence', 44),('harry potter and the chamber of secrets', 66),
('wuthering heights', 77),('goosebumps', 88),('The secret of nagas', 99),
('machine learning', 109),('the famous five', 119),('the great gatsby', 129),
('the adventures of tintin', 139),('computer network security', 149),
('computer and internet security', 159),('charlie and the chocolate factory',
169),('percy jackson: the lightning theif', 179),
('fundamentals of database systems', 189),
('percy jackson and the sea of monsters', 179),
('the 3 mistakes of my life', 22),
('introduction to modern cryptography', 199),('the bfg', 169),
('matilda', 169),('the design of everyday things', 219),
```

('concepts of programming languages', 229),('ram:scion of ikshvaku', 99),  
('sita:warrior of mithila', 99),('raavan:the enemy of aryavarta', 99),  
('the immortals of meluha', 99),('james and the giant peach', 169),  
('the oath of vayuputras', 99),('artificial intelligence', 44),  
('intoduction to modern cryptography', 209);

11)

```
INSERT INTO written_by (`ISBN`, `author_id`) VALUES  
('b222', 11),('b986', 22),('b152', 33),('b565', 44),('b826', 66),  
('b777', 77),('b428', 88),('b452', 99),('b908', 109),('b007', 119),  
('b545', 129),('b998', 139),('b324', 149),('b767', 159),('b037', 169),  
('b374', 179),('b231', 189),('b232', 179),('b999', 22),('b760', 199),  
('b209', 169),('b155', 169),('b521', 219),('b774', 229),('b436', 99),  
('b769', 99),('b234', 99),('b667', 99),('b133', 169),('b723', 99),  
('b565', 44),('b760', 209);
```

## 6. Join Queries

Showcase at least 4 join queries

Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

- 1) Find all the admins that work under the manager having ID=1.

Sol :

SQL Statement: select admins.f\_name,admins.l\_name from branch  
inner join admins where branch.manager\_id=admins.manager\_id and  
branch.manager\_id=1

Output:

The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. The left sidebar lists various databases and tables. The main area is titled 'Table: admins' and displays the results of the following SQL query:

```
select admins.f_name,admins.l_name from branch inner join admins where branch.manager_id=admins.manager_id and branch.manager_id=1;
```

The results are shown in a table with columns 'f\_name' and 'l\_name'. There are two rows of data:

f_name	l_name
rohit	sharma
suryakumar	yadav

Below the table are several operation buttons: Print, Copy to clipboard, Export, Display chart, Create view, and a 'Bookmark this SQL query' link.

2) Customers who has issued one or more books in the library.

Soln:

SQL Statement: `SELECT DISTINCT customer.f_name, customer.l_name  
from customer natural join issue_return where  
customer.customer_id=issue_return.customer_id`

Output:

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** my\_project
- Table:** customer
- Query:** `customer.customer_id=issue_return.customer_id;`
- Results:** A table with 15 rows, showing columns `f_name` and `l_name`. The data is as follows:

f_name	l_name
vishwas	m
tushar	bothra
sheldon	cooper
raghav	loknath
vishnudeep	mysore
sai	rethwik
vishwa	mehta
mike	ross
vinti	agrawal
vishal	elangovan
shuvam	bose
bhat	

3) Customers who did not issue any of the books in the library.

Soln:

SQL Statement: `select customer.f_name, customer.l_name  
from customer where customer_id not in (SELECT DISTINCT  
customer.customer_id from customer right outer join issue_return on  
customer.customer_id=issue_return.customer_id)`

## Output:

The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. The left sidebar lists tables such as information\_schema, library, mysql, my\_project (containing New, admins, author, author\_names, books, book\_names, branch, customer, issue\_return, phone\_no, publisher, written\_by), performance\_schema, pes2ug20cs390\_vishwas\_m, phpmyadmin, and railway\_db. The current table is 'customer'. The SQL query executed is:

```
select customer.f_name, customer.l_name from customer where customer_id not in (SELECT DISTINCT customer.customer_id from customer right outer join issue_return on customer.customer_id=issue_return.customer_id);
```

The results show 6 rows:

f_name	l_name
vismaya	r
tenzin	tsephel
vidip	chabhra
jennifer	aniston
bavanika	v
srushti	s
sahith	reddy

4) Display all the books written by the author whose first name is Chetan.

Soln:

SQL statement: select book\_names.name from book\_names inner join author\_names where book\_names.author\_id=author\_names.author\_id and author\_names.f\_name = 'chetan'

Output:

The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. The left sidebar lists tables such as information\_schema, library, mysql, my\_project (containing New, admins, author, author\_names, books, book\_names, branch, customer, issue\_return, phone\_no, publisher, written\_by), performance\_schema, pes2ug20cs390\_vishwas\_m, phpmyadmin, and railway\_db. The current table is 'book\_names'. The SQL query executed is:

```
select book_names.name from book_names inner join author_names where book_names.author_id=author_names.author_id and author_names.f_name = 'chetan';
```

The results show 2 rows:

name
revolution 2020
the 3 mistakes of my life

## 7. Aggregate Functions

Showcase at least 4 Aggregate function queries

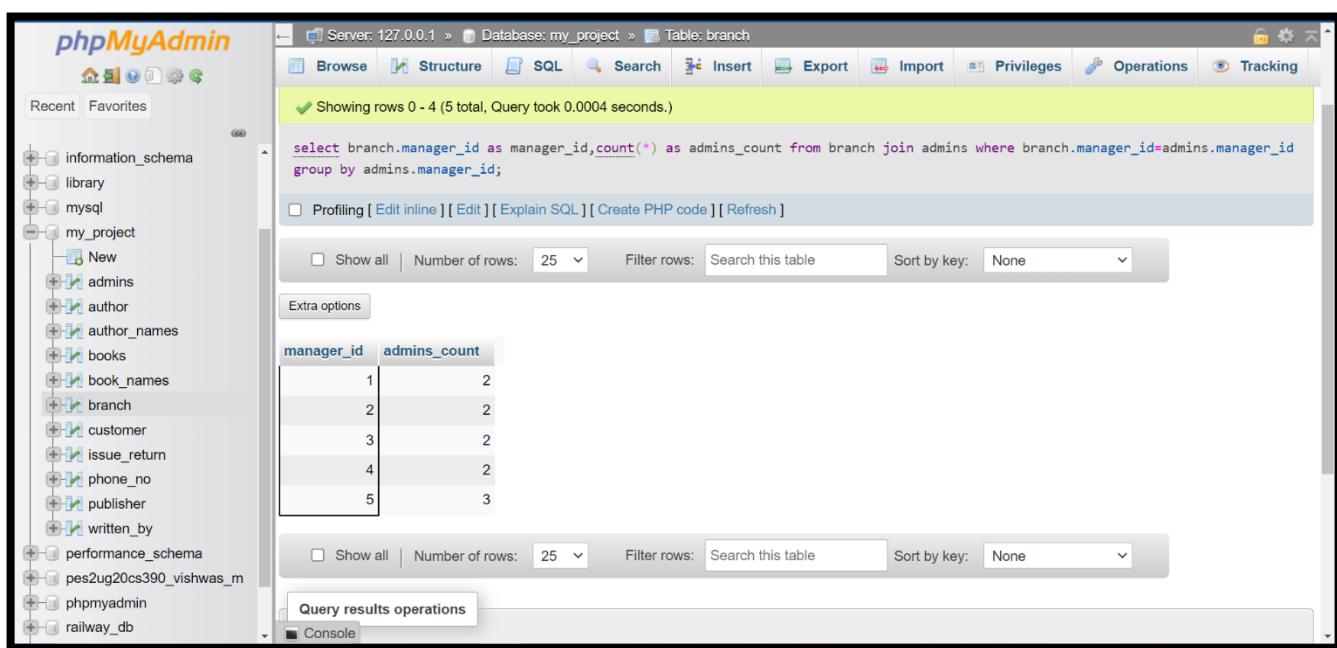
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

- 1) Find the number of admins working under each branch managers in the library.

Soln:

SQL Statement: select branch.manager\_id as manager\_id, count(\*) as admins\_count from branch join admins where branch.manager\_id=admins.manager\_id group by admins.manager\_id

Output:



The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. The left sidebar lists various tables: information\_schema, library, mysql, my\_project (with New, admins, author, author\_names, books, book\_names, branch, customer, issue\_return, phone\_no, publisher, written\_by), performance\_schema, pes2ug20cs390\_vishwas\_m, phpmyadmin, and railway\_db. The main panel is titled 'Table: branch' and displays the results of the following SQL query:

```
select branch.manager_id as manager_id, count(*) as admins_count from branch join admins where branch.manager_id=admins.manager_id group by admins.manager_id;
```

The results are shown in a table with two columns: 'manager\_id' and 'admins\_count'. The data is:

manager_id	admins_count
1	2
2	2
3	2
4	2
5	3

2) Find the number of books in each category of books.

Soln:

SQL Statement: select books.category,count(\*) as number\_of\_books from books group by books.category

Output:

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** my\_project
- Table:** books
- Query Result:** Showing rows 0 - 3 (4 total, Query took 0.0005 seconds.)
- SQL Query:** select books.category, count(\*) as number\_of\_books from books group by books.category;
- Table Headers:** category, number\_of\_books
- Data Rows:**

category	number_of_books
academics	8
fiction	5
kids	6
novel	11
- Action Buttons:** Edit, Copy, Delete for each row; Check all, With selected: Edit, Copy, Delete, Export.

3) Find the number of books published by each publisher.

Soln:

SQL Statement: select publisher.publisher\_id, count(\*) as number\_of\_books from books join publisher where books.publisher\_id=publisher.publisher\_id GROUP by publisher.publisher\_id;

Output:

d

phpMyAdmin

Server: 127.0.0.1 » Database: my\_project » Table: publisher

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking

Showing rows 0 - 3 (4 total, Query took 0.0175 seconds.)

```
select publisher.publisher_id, count(*) as number_of_books from books join publisher where books.publisher_id=publisher.publisher_id GROUP by publisher.publisher_id;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

publisher_id	number_of_books
1000	8
2000	8
3000	7
4000	7

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations Console

- 4) Find number of books written by each author.

Soln:

SQL statement: select distinct written\_by.author\_id, count(\*) as number\_of\_books from book\_names join written\_by where book\_names.author\_id = written\_by.author\_id GROUP by book\_names.name order by count(\*) DESC;

Output:

phpMyAdmin

Server: 127.0.0.1 » Database: my\_project » Table: written\_by

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking

author_id	number_of_books
99	6
44	4
169	4
179	2
199	2
22	2
66	1
77	1
189	1
88	1
109	1
119	1
129	1
11	1
139	1
219	1
149	1

Console

## 8. Set Operations

Showcase at least 4 Set Operations queries

Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

- 1) Find all the books written by authors having author id 22 and 169.

Soln:

SQL Statement: select book\_names.name,book\_names.author\_id from book\_names where book\_names.author\_id=22  
UNION  
select book\_names.name,book\_names.author\_id from book\_names where book\_names.author\_id=169

Output:

The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. The left sidebar lists various tables, including 'book\_names', which is currently selected. The main area displays the results of a UNION query:

```
select book_names.name,book_names.author_id from book_names where book_names.author_id=22 UNION select book_names.name,book_names.author_id from book_names where book_names.author_id=169;
```

The results table shows the following data:

name	author_id
revolution 2020	22
the 3 mistakes of my life	22
charlie and the chocolate factory	169
the bfg	169
matilda	169
james and the giant peach	169

- 2) Find all the customers who haven't issued any books from the library using set operations.

Sol:

SQL Statement: `SELECT * from customer where not EXISTS (select * FROM issue_return  
where customer.customer_id = issue_return.customer_id)`

- 3) Find all the customers that come under the branch managers having manager\_id 4 and 5.

Soln:

SQL Statement: `select customer.customer_id, customer.f_name, customer.l_name from customer where customer.manager_id in (4,5)`

Output:

- 4) Find all the details regarding the books issued by the customer whose id's are 11111 and 22222.

Soln:

SQL Statement: select \* from issue\_return where issue\_return.customer\_id=11111  
union ALL  
select \* from issue\_return where issue\_return.customer\_id=22222

Output:

The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. The left sidebar lists various tables: mysql, Functions, Procedures, Tables (including New, admins, author, author\_names, books, book\_names, branch, customer, issue\_return, phone\_no, publisher, written\_by), performance\_schema, pes2ug20cs370\_train\_lab10, and pes2ug20cs390\_vishwas\_m. The 'Tables' section is expanded. The main area shows the results of a query run against the 'issue\_return' table. The query is:

```
select * from issue_return where issue_return.customer_id=11111 union ALL select * from issue_return where issue_return.customer_id=22222;
```

The results show four rows of data:

customer_id	admin_id	ISBN	issue_date	return_date	fine
11111	2323	b222	2022-11-12	2022-11-19	0
11111	4545	b986	2022-11-12	2022-11-19	0
22222	7878	b152	2022-11-03	2022-11-10	27
22222	8989	b565	2022-11-03	2022-11-10	27

## 9. Functions and Procedures

Create a Function and Procedure. State the objective of the function / Procedure. Run and display the results.

- 1) Create a function which checks whether the number of admins under a branch manager exceeds 2 or not. If exceeds more than 2 then we cannot add more admins under that manager else we can add more.

Soln: Creating the function

The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. In the left sidebar, there's a tree view of tables: information\_schema, library, mysql, my\_project (which contains New, admins, author, author\_names, books, book\_names, branch, customer, issue\_return, phone\_no, publisher, written\_by), performance\_schema, pes2ug20cs390\_vishwas\_m, and phpmyadmin. The main query editor window has a green status bar at the top stating 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0483 seconds.)'. Below it is the SQL code for creating a function:

```
CREATE FUNCTION tickets_check_limit(counts INT) RETURNS VARCHAR(50) BEGIN DECLARE sf_value VARCHAR(50); IF counts > 2 THEN SET sf_value = 'cannot add more admins'; ELSE SET sf_value= 'can add more admins'; END IF; RETURN sf_value; END;
```

Below the code are three buttons: [Edit inline], [Edit], and [Create PHP code].

SQL QUERY: select admins.manager\_id,tickets\_check\_limit(count(\*)) as count from admins  
GROUP by admins.manager\_id

Output:

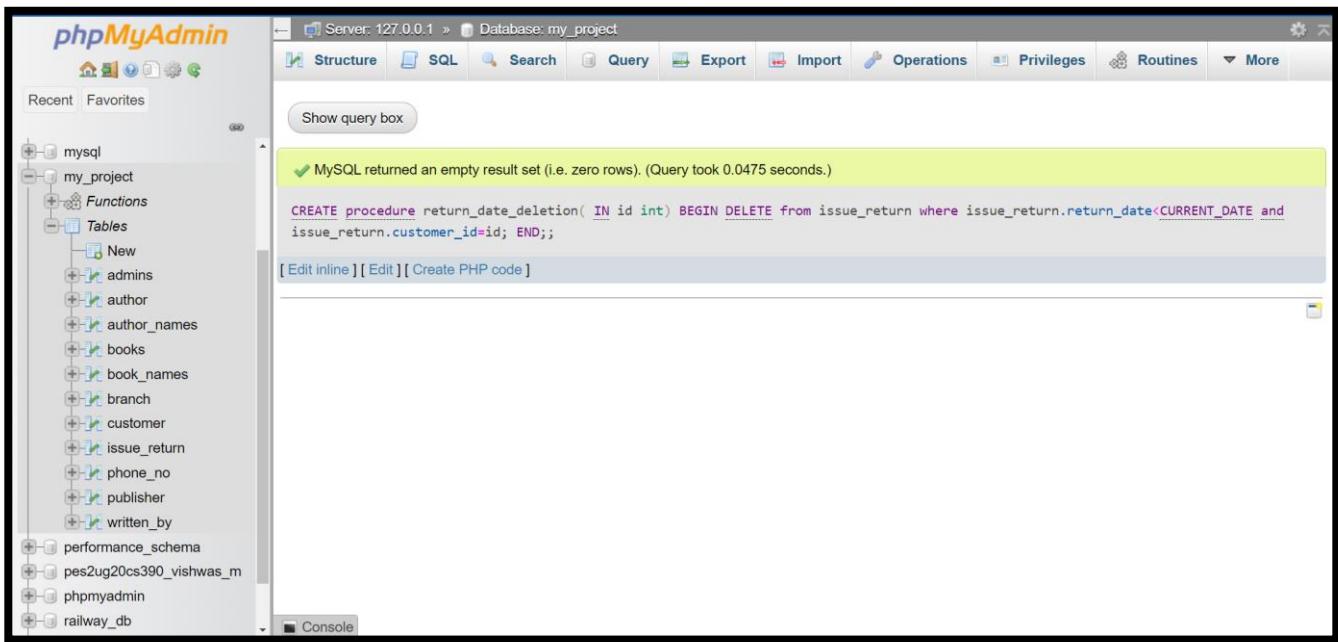
The screenshot shows the phpMyAdmin interface for the 'my\_project' database, specifically for the 'admins' table. The left sidebar shows the same tree structure as before. The main area displays the results of the previously run SQL query:

manager_id	count
1	can add more admins
2	can add more admins
3	can add more admins
4	can add more admins
5	cannot add more admins

At the bottom of the results table, there are several buttons: Edit, Copy, Delete, Check all, With selected:, Edit, Copy, Delete, and Export.

2) Delete the tuple whose due date has come to an end from the table using procedures.

Soln:

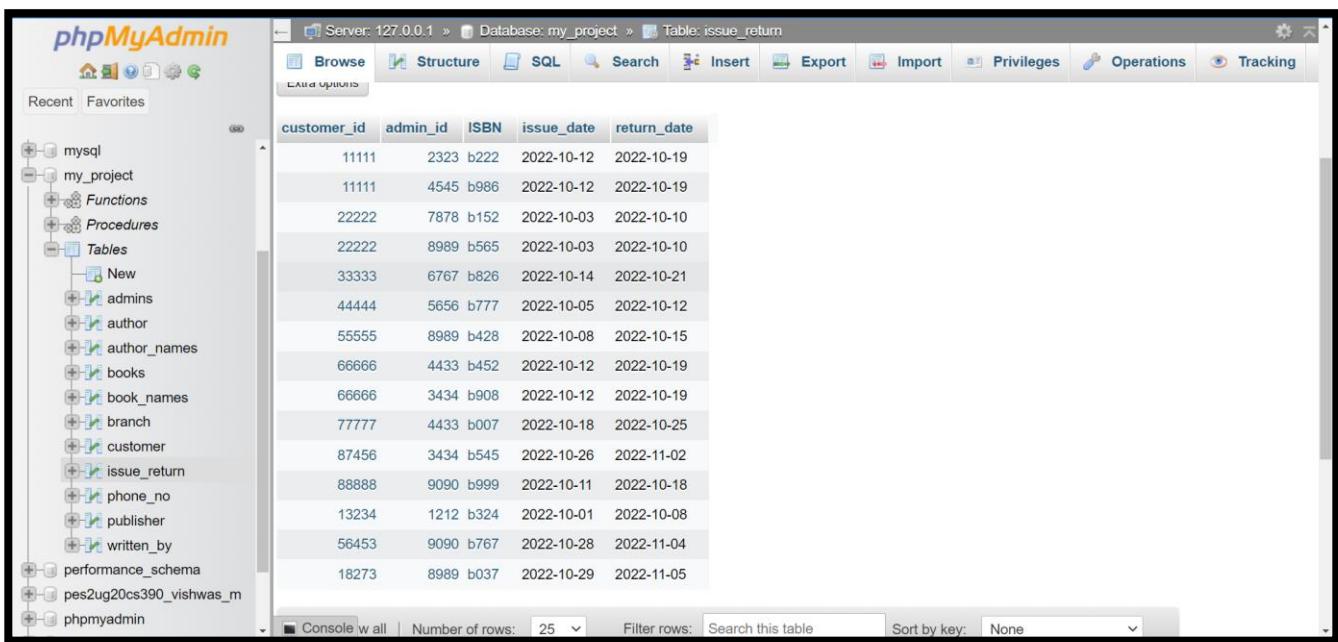


The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. In the left sidebar, under 'Tables', there are several tables listed: admins, author, author\_names, books, book\_names, branch, customer, issue\_return, phone\_no, publisher, and written\_by. In the main query editor area, a MySQL procedure is being created:

```
CREATE procedure return_date_deletion( _IN id int) BEGIN DELETE from issue_return where issue_return.return_date<CURRENT_DATE and issue_return.customer_id=id; END;
```

The status bar at the bottom indicates: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0475 seconds.)

Initial table:



The screenshot shows the 'issue\_return' table in the 'my\_project' database. The table has columns: customer\_id, admin\_id, ISBN, issue\_date, and return\_date. The data is as follows:

	customer_id	admin_id	ISBN	issue_date	return_date
1	11111	2323	b222	2022-10-12	2022-10-19
2	11111	4545	b986	2022-10-12	2022-10-19
3	22222	7878	b152	2022-10-03	2022-10-10
4	22222	8989	b565	2022-10-03	2022-10-10
5	33333	6767	b826	2022-10-14	2022-10-21
6	44444	5656	b777	2022-10-05	2022-10-12
7	55555	8989	b428	2022-10-08	2022-10-15
8	66666	4433	b452	2022-10-12	2022-10-19
9	66666	3434	b908	2022-10-12	2022-10-19
10	77777	4433	b007	2022-10-18	2022-10-25
11	87456	3434	b545	2022-10-26	2022-11-02
12	88888	9090	b999	2022-10-11	2022-10-18
13	13234	1212	b324	2022-10-01	2022-10-08
14	56453	9090	b767	2022-10-28	2022-11-04
15	18273	8989	b037	2022-10-29	2022-11-05

SQL Query: call return\_date\_deletion(56453);

The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. In the left sidebar, there is a tree view of tables: mysql, my\_project, Functions, Procedures, Tables, New, admins, author, author\_names, books, book\_names, branch, customer, issue\_return, phone\_no, publisher, written\_by, performance\_schema, pes2ug20cs390\_vishwas\_m, and phpmyadmin. The 'Tables' node is expanded. In the main area, the SQL tab is active, displaying the query: 'call return\_date\_deletion(56453);'. A green status bar at the top indicates: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0771 seconds.)'. Below the status bar, there are three buttons: [Edit inline], [Edit], and [Create PHP code].

After deletion:

The screenshot shows the phpMyAdmin interface for the 'my\_project' database, specifically the 'issue\_return' table. The left sidebar shows the same tree structure as the previous screenshot. The 'Tables' node is expanded, and 'issue\_return' is selected. The main area displays the contents of the 'issue\_return' table in a grid format. The columns are labeled: customer\_id, admin\_id, ISBN, issue\_date, and return\_date. The data consists of 18 rows, each representing a record of an item issued and its return date. The rows are as follows:

customer_id	admin_id	ISBN	issue_date	return_date
11111	2323	b222	2022-10-12	2022-10-19
11111	4545	b986	2022-10-12	2022-10-19
22222	7878	b152	2022-10-03	2022-10-10
22222	8989	b565	2022-10-03	2022-10-10
33333	6767	b826	2022-10-14	2022-10-21
44444	5656	b777	2022-10-05	2022-10-12
55555	8989	b428	2022-10-08	2022-10-15
66666	4433	b452	2022-10-12	2022-10-19
66666	3434	b908	2022-10-12	2022-10-19
77777	4433	b007	2022-10-18	2022-10-25
87456	3434	b545	2022-10-26	2022-11-02
88888	9090	b999	2022-10-11	2022-10-18
13234	1212	b324	2022-10-01	2022-10-08
18273	8989	b037	2022-10-29	2022-11-05

## 10. Triggers and Cursors

Create a Trigger and a Cursor. State the objective. Run and display the results.

- 1) A customer can issue maximum of 2 books from the library. Before inserting, check the number of books issued by the user and if and only if the number of books issued is less than or equal to only then the book should be issued else should give an error stating the reason. Use triggers to perform the above objective.

Creation of trigger :

The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. The left sidebar displays the database structure with tables like 'admins', 'author', 'book', etc. The main area shows a SQL query window with the following code:

```
CREATE TRIGGER before_issuing_book BEFORE INSERT ON issue_return FOR EACH ROW BEGIN DECLARE err_msg varchar(100); DECLARE counts int; SET err_msg = "The number of books issued cannot exceed 2"; SELECT COUNT(*) into counts from issue_return where issue_return.customer_id=new.customer_id; IF counts=2 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT=err_msg; END IF; END;
```

The query was executed successfully, as indicated by the message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0613 seconds.)". There are also links to 'Edit inline', 'Edit', and 'Create PHP code'.

SQL QUERY: insert into issue\_return VALUES ( 11111, 2323, 'b152', CURRENT\_DATE, CURRENT\_DATE + 7 )

The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. The left sidebar lists tables such as 'admins', 'author', 'books', 'book\_names', 'branch', 'customer', 'issue\_return', 'phone\_no', 'publisher', and 'written\_by'. The main area contains a SQL query editor with the following content:

```
insert into issue_return VALUES(11111,2323,'b152',CURRENT_DATE,CURRENT_DATE+7);
```

An error message is displayed in a red box:

**MySQL said:** #1644 - The number of books issued cannot exceed 2

Error occurs because customer having customer\_id =11111 already issued 2 books.

SQL QUERY: insert into issue\_return VALUES ( 18273, 2323, 'b152', CURRENT\_DATE, CURRENT\_DATE+7 )

The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. The left sidebar lists tables such as 'admins', 'author', 'books', 'book\_names', 'branch', 'customer', 'issue\_return', 'phone\_no', 'publisher', and 'written\_by'. The main area contains a SQL query editor with the following content:

```
insert into issue_return VALUES(18273,2323,'b152',CURRENT_DATE,CURRENT_DATE+7);
```

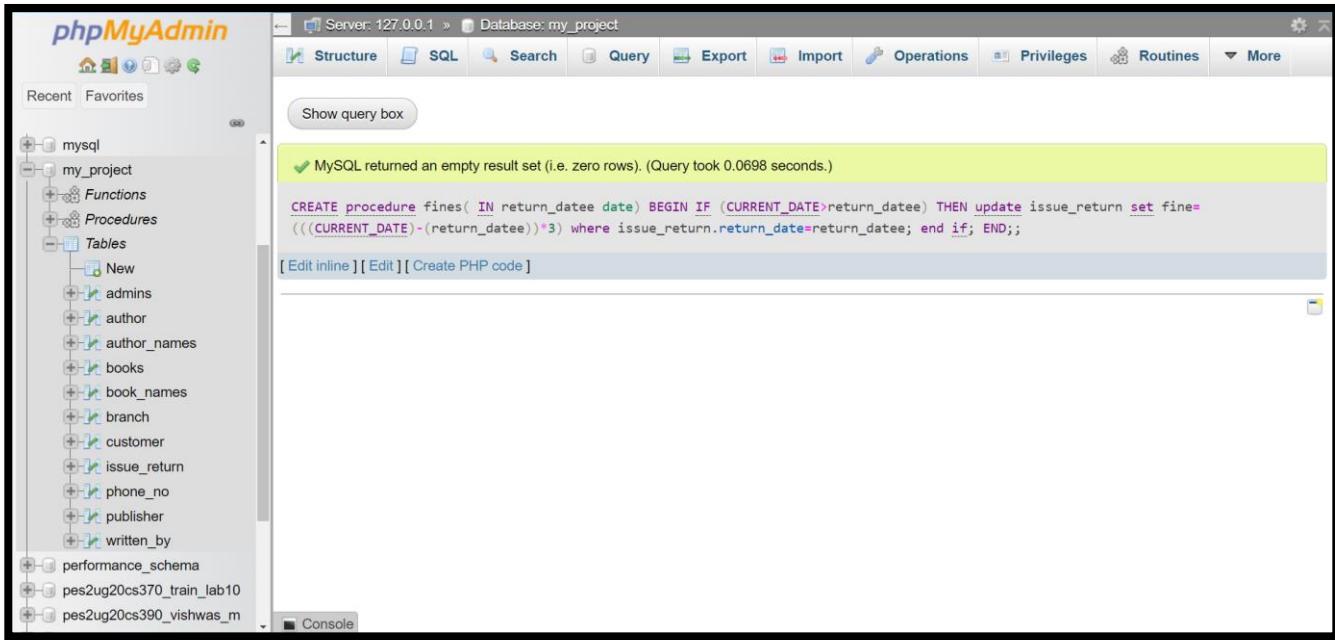
A green success message is displayed in a yellow box:

✓ 1 row inserted. (Query took 0.0009 seconds.)

Error does not occur because customer having customer\_id 18273 issued only one book and can still issue one more book.

## 2) Cursors :

Procedure to update the Fine column in issue\_return table

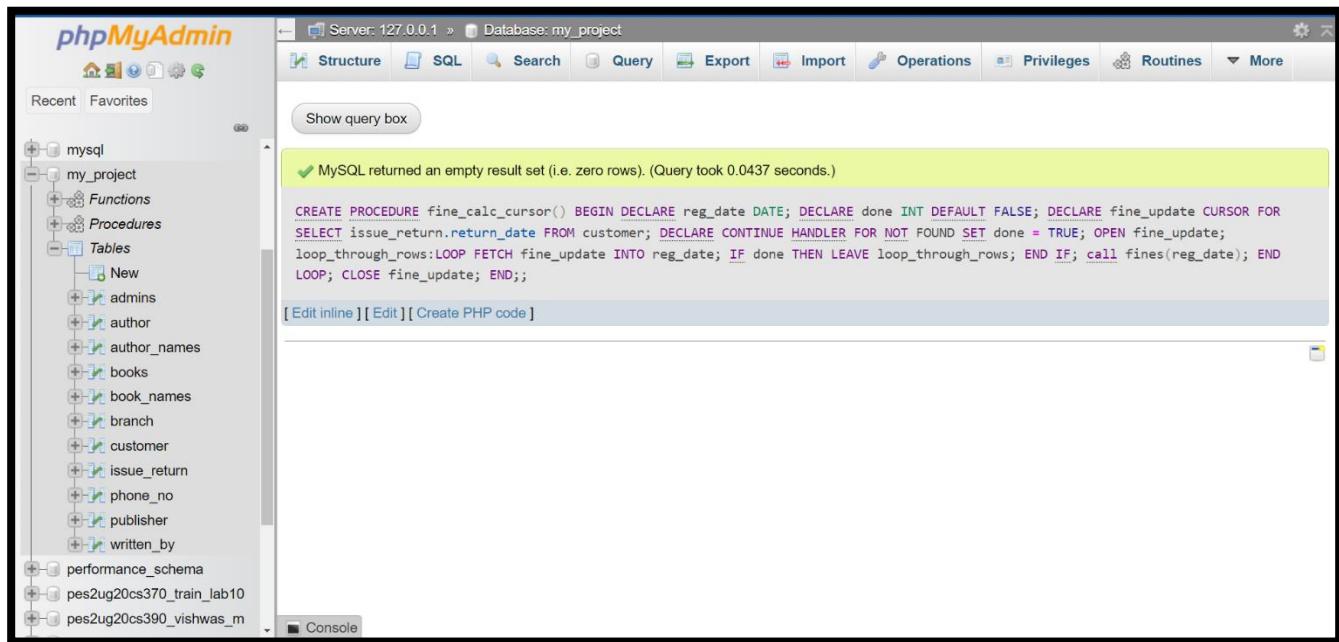


The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. The left sidebar lists various tables and their structures. In the main query editor area, a MySQL procedure named 'fines' is being created. The code is as follows:

```
CREATE procedure fines( IN return_datee date) BEGIN IF (CURRENT_DATE>return_datee) THEN update issue_return set fine=(((CURRENT_DATE)-(return_datee))*3) where issue_return.return_date=return_datee; end if; END;;
```

The status bar at the top indicates: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0698 seconds.)

Cursor to call the above procedure:



The screenshot shows the phpMyAdmin interface for the same database 'my\_project'. The left sidebar shows the same list of tables. In the main query editor area, a MySQL procedure named 'fine\_calc\_cursor' is being created. The code is as follows:

```
CREATE PROCEDURE fine_calc_cursor() BEGIN DECLARE reg_date DATE; DECLARE done INT DEFAULT FALSE; DECLARE fine_update CURSOR FOR SELECT issue_return.return_date FROM customer; DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE; OPEN fine_update; loop_through_rows:LOOP FETCH fine_update INTO reg_date; IF done THEN LEAVE loop_through_rows; END IF; call fines(reg_date); END LOOP; CLOSE fine_update; END;;
```

The status bar at the top indicates: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0437 seconds.)

Before Update:

The screenshot shows the phpMyAdmin interface for the 'my\_project' database. The left sidebar lists various tables, including 'admins', 'author', 'books', 'book\_names', 'branch', 'customer', 'issue\_return', 'phone\_no', 'publisher', and 'written\_by'. The 'issue\_return' table is selected in the main area, displaying 18 rows of data. The columns are labeled: customer\_id, admin\_id, ISBN, issue\_date, return\_date, and fine. The data shows various entries with dates ranging from 2022-11-03 to 2022-11-19 and fines mostly set to 0.

customer_id	admin_id	ISBN	issue_date	return_date	fine
11111	2323	b222	2022-11-12	2022-11-19	0
11111	4545	b986	2022-11-12	2022-11-19	0
22222	7878	b152	2022-11-03	2022-11-10	0
22222	8989	b565	2022-11-03	2022-11-10	0
33333	6767	b826	2022-11-14	2022-11-21	0
44444	5656	b777	2022-11-05	2022-11-12	0
55555	8989	b428	2022-11-08	2022-11-15	0
66666	4433	b452	2022-11-12	2022-11-19	0
66666	3434	b908	2022-11-12	2022-11-19	0
77777	4433	b007	2022-11-18	2022-11-25	0
87456	3434	b545	2022-11-06	2022-11-13	0
88888	9090	b999	2022-11-11	2022-11-18	0
13234	1212	b324	2022-11-01	2022-11-08	0
18273	8989	b037	2022-11-16	2022-11-23	0
18273	2323	b152	2022-11-05	2022-11-12	0

SQL command for calling the cursor: call fine\_calc\_cursor();

The screenshot shows the phpMyAdmin interface after executing the SQL query 'call fine\_calc\_cursor();'. The message bar at the top indicates that MySQL returned an empty result set (0.1441 seconds). The query editor shows the executed command.

```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.1441 seconds.)  
call fine_calc_cursor();
```

## After Calling the Cursor:

The screenshot shows the phpMyAdmin interface for a database named 'my\_project'. The left sidebar lists various tables: mysql, my\_project, Functions, Procedures, Tables (including New, admins, author, author\_names, books, book\_names, branch, customer, issue\_return, phone\_no, publisher, written\_by), performance\_schema, pes2ug20cs370\_train\_lab10, and pes2ug20cs390\_vishwas\_m. The 'issue\_return' table is selected and displayed in the main pane. The table has columns: customer\_id, admin\_id, ISBN, issue\_date, return\_date, and fine. The data shows 15 rows of issued and returned book information. The bottom of the screen shows navigation links like 'Console w all', 'Number of rows: 25', 'Filter rows: Search this table', and 'Sort by key: None'.

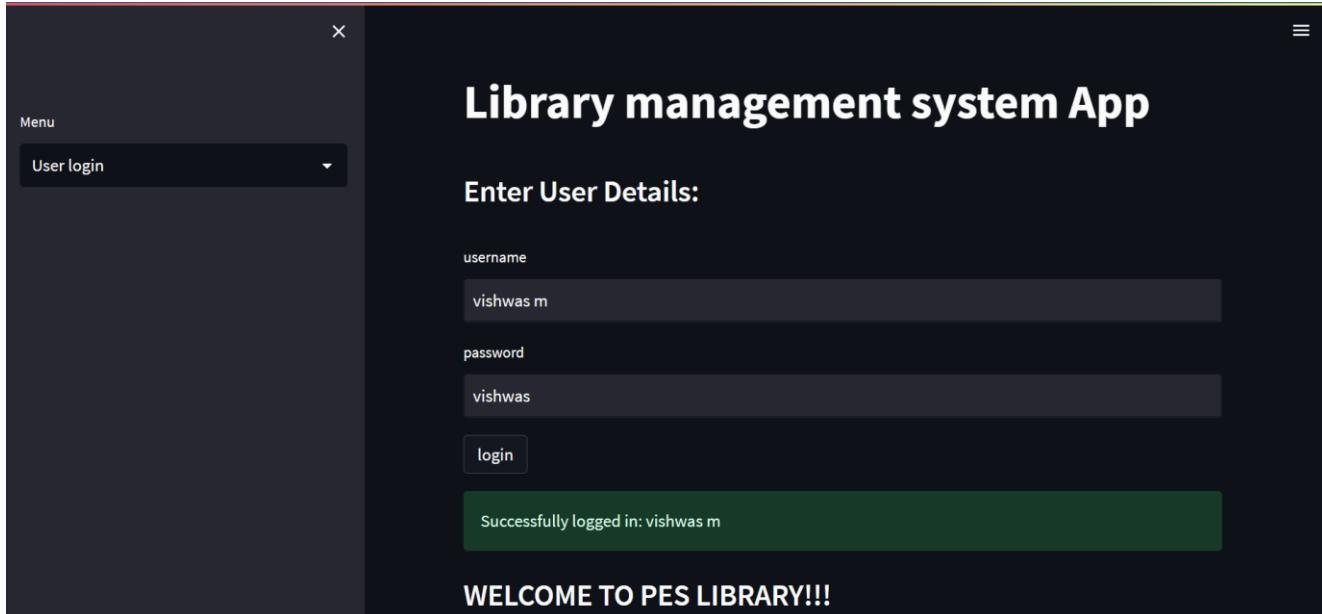
customer_id	admin_id	ISBN	issue_date	return_date	fine
11111	2323	b222	2022-11-12	2022-11-19	0
11111	4545	b986	2022-11-12	2022-11-19	0
22222	7878	b152	2022-11-03	2022-11-10	27
22222	8989	b565	2022-11-03	2022-11-10	27
33333	6767	b826	2022-11-14	2022-11-21	0
44444	5656	b777	2022-11-05	2022-11-12	21
55555	8989	b428	2022-11-08	2022-11-15	12
66666	4433	b452	2022-11-12	2022-11-19	0
66666	3434	b908	2022-11-12	2022-11-19	0
77777	4433	b007	2022-11-18	2022-11-25	0
87456	3434	b545	2022-11-06	2022-11-13	18
88888	9090	b999	2022-11-11	2022-11-18	3
13234	1212	b324	2022-11-01	2022-11-08	33
18273	8989	b037	2022-11-16	2022-11-23	0
18273	2323	b152	2022-11-05	2022-11-12	21

## 11. Developing a Frontend

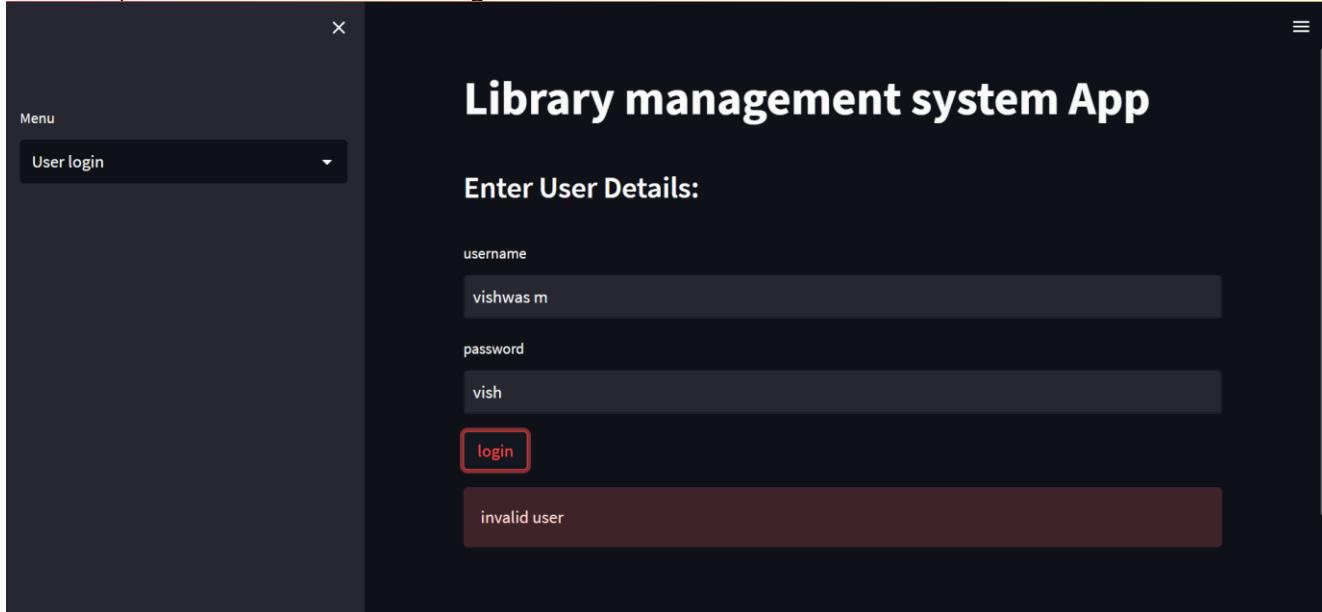
The frontend should support

1. Addition, Modification and Deletion of records from any chosen table
2. There should be an window to accept and run any SQL statement and display the result

1) User successful login:



2) User unsuccessful login:



- 3) Displays personal info of the user and the books he issued from the library

The screenshot shows a dark-themed application interface. On the left, there is a sidebar with a 'Menu' section and a 'User login' dropdown. The main content area has three sections: 'Personal info:' with a table showing a single row of user details; 'Issued Books:' with a table showing two rows of issued books; and 'All Books:' which is currently inactive.

**Personal info:**

	customer_id	f_name	l_name	contact_no	state	city	street	pin
0	11111	vishwas	m	2	udupi	182432	2022-10-12	1

**Issued Books:**

	book name	ISBN	price
0	robinson crusoe	b222	289
1	revolution 2020	b986	543

**All Books:**

- 4) User can view all the books in the library

The screenshot shows a dark-themed application interface. On the left, there is a sidebar with a 'Menu' section and a 'User login' dropdown. The main content area has two sections: 'Issued Books' (which is inactive) and 'All Books:' (which is active). The 'All Books:' section displays a table of books with 12 rows.

**All Books:**

	ISBN	Price	category	status	title	admin id	publisher id
5	b209	1500	kids	yes	the bfg	1313	1000
6	b222	289	novel	yes	robinson crusoe	4545	1000
7	b231	333	academics	yes	fundamentals of database systems	2323	1000
8	b232	599	novel	yes	percy jackson and the sea of monsters	1313	2000
9	b234	1099	novel	yes	raavan:the enemy of aryavarta	1212	3000
10	b324	3432	academics	no	computer network security	2323	1000
11	b374	223	fiction	yes	percy jackson: the lightning thief	1212	4000
12	b428	756	fiction	no	goosebumps	5656	3000

- 5) Check if book exists in library or not by searching the book name  
If exists:

The screenshot shows a dark-themed user interface. On the left, there is a sidebar with a "Menu" button and a "User login" dropdown. The main content area has a header "Check Book By Book Name:". Below it is a form field labeled "book\_name" containing the text "the famous five". A red-bordered "check book" button is positioned below the input field. A green success message box at the bottom contains the text "the famous five exists in library".

If not exists:

The screenshot shows a dark-themed user interface. On the left, there is a sidebar with a "Menu" button and a "User login" dropdown. The main content area has a header "Check Book By Book Name:". Below it is a form field labeled "book\_name" containing the text "harry potter and the prisoner of azkaban". A red-bordered "check book" button is positioned below the input field. A red error message box at the bottom contains the text "book does not exist".

- 6) Check if book exists in library or not by searching the author's name  
If exists:

The screenshot shows a dark-themed web application interface. On the left, there is a sidebar with a "Menu" section containing a "User login" button. The main content area has a title "Check Book By Author Name:". Below it, a form field labeled "author\_name" contains the value "Chetan Bhagat". A "check author" button is present. To the right, a table titled "all books" displays two rows of data:

	ISBN	Price	category	status	title
0	b986	543	fiction	no	revolution 2020
1	b999	1090	novel	yes	the 3 mistakes of my life

Below this, another section titled "Check Book Based On Category:" is visible, featuring a "category" dropdown menu.

If author's books are not available in the library:

The screenshot shows a similar dark-themed web application interface. The sidebar and main search fields are identical to the previous screenshot. However, the search results for "rohan chakraborty" show an error message: "invalid author's name" is displayed in a red-bordered box. The "category" dropdown menu is also visible at the bottom.

7) Check books by category:

Categories:

The screenshot shows a dark-themed web application interface. On the left, there is a vertical sidebar with a "Menu" button at the top and a "User login" button below it. The main content area has a title "Check Book Based On Category:". Below the title is a "category" dropdown menu. The dropdown is open, showing four options: "kids", "academics", "novel", and "fiction".

For kids:

The screenshot shows the same application interface after selecting the "kids" category. The main content area now displays a table titled "all books" with the following data:

	ISBN	Price	status	title
0	b007	1390	yes	the famous five
1	b037	1232	yes	charlie and the chocolate factory
2	b133	789	yes	james and the giant peach
3	b155	1800	yes	matilda
4	b209	1500	yes	the bfg
5	b998	2132	yes	the adventures of tintin

8) Change user password:

Initial password:

<input type="button" value="← T →"/>	<input type="button" value="▼"/>	customer_id	f_name	I_name	street	city	pin	reg_date	manager_id	password
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	11111	vishwas	m	2	udupi	182432	2022-10-12

Password Change:

Change Password

New Password  
vishwas123

Retype Password  
vishwas123

Change

Password changed

Updated Database:

<input type="button" value="← T →"/>	<input type="button" value="▼"/>	customer_id	f_name	I_name	street	city	pin	reg_date	manager_id	password
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	11111	vishwas	m	2	udupi	182432	2022-10-12

Branch Manager Page:

1) Manager login:

The screenshot shows the 'Library management system App' interface. On the left, there is a dark sidebar with a 'Menu' section containing a 'Manager Login' button. The main content area has a title 'Library management system App' and a sub-section 'Enter Manager details:'. It contains two input fields: 'username' with the value 'tanya arora' and 'password' with the value 'tanya'. Below these is a red-bordered 'login' button. A green success message at the bottom states 'Successfully logged in: tanya arora'. At the very bottom, a white banner says 'WELCOME TO PES LIBRARY!!!'.

This screenshot shows the same application interface as the previous one, but with different results. The 'username' field still contains 'tanya arora' and the 'password' field contains 'tan'. When the 'login' button is clicked, a brown error message box appears at the bottom with the text 'invalid details'.

2) Displays personal info of manager:

**WELCOME TO PES LIBRARY!!!**

**Personal info:**

	manager_id	f_name	l_name	contact_no	state	city	street	pin
0	2	tanya	arora	9867198987	andhra pradesh	amaravati	reddy street	563726

3) Check all the admins under that particular manager:

**check admins:**

	admin_id	f_name	l_name	salary
0	3434	lokesh	rahul	30000
1	4433	sanju	samson	30000

**Add admins:**

admin_id	f_name
salary	l_name

4) Add admins:

The screenshot shows a dark-themed application window titled "Add admins:". It contains two input fields: "admin\_id" with value "909090" and "f\_name" with value "rudra". Below these is another pair of fields: "salary" with value "35000" and "l\_name" with value "pratap". A red-bordered "Add Admin" button is visible. A green success message at the bottom states "Successfully added Admin: rudra pratap".

**delete**

Admin to Delete

3434

<input type="checkbox"/>		Edit		Copy		Delete	9090	30000	jasprit	bumrah	4	bumrah
<input type="checkbox"/>		Edit		Copy		Delete	909090	35000	rudra	pratap	2	rudra

5) Delete admins:

The screenshot shows a dark-themed application window titled "delete". It displays a single item in a dropdown menu: "909090". A green message box asks "Do you want to delete :909090". Below it is a red-bordered "Delete Admin" button. A green success message at the bottom states "successfully deleted admin 909090".

**Delete Customers:**

Customer to delete

11111

phpMyAdmin

Server: 127.0.0.1 » Database: my\_project » Table: admins

	admin_id	salary	f_name	l_name	manager_id	Password
<input type="checkbox"/>	1212	30000	mohammad	shami	5	shami
<input type="checkbox"/>	1313	30000	yuzvendra	chahal	5	chahal
<input type="checkbox"/>	2323	30000	rohit	sharma	1	sharma
<input type="checkbox"/>	3434	30000	lokesh	rahul	2	lokesh
<input type="checkbox"/>	4433	30000	sanju	samson	2	sanju
<input type="checkbox"/>	4545	30000	suryakumar	yadav	1	yadav
<input type="checkbox"/>	5656	30000	hardik	pandya	4	pandya
<input type="checkbox"/>	6767	30000	ishaan	kishan	5	kishan
<input type="checkbox"/>	7878	30000	ravindra	jadeja	3	jadeja
<input type="checkbox"/>	8989	30000	dinesh	Karthik	3	Karthik
<input type="checkbox"/>	9090	30000	Jasprit	Bumrah	4	Bumrah

Extra options:  Check all    With selected:  Edit  Copy  Delete  Export

## 6) Delete customer:

Menu

Manager Login

Delete Admin

Delete Customers:

Customer to delete

33333

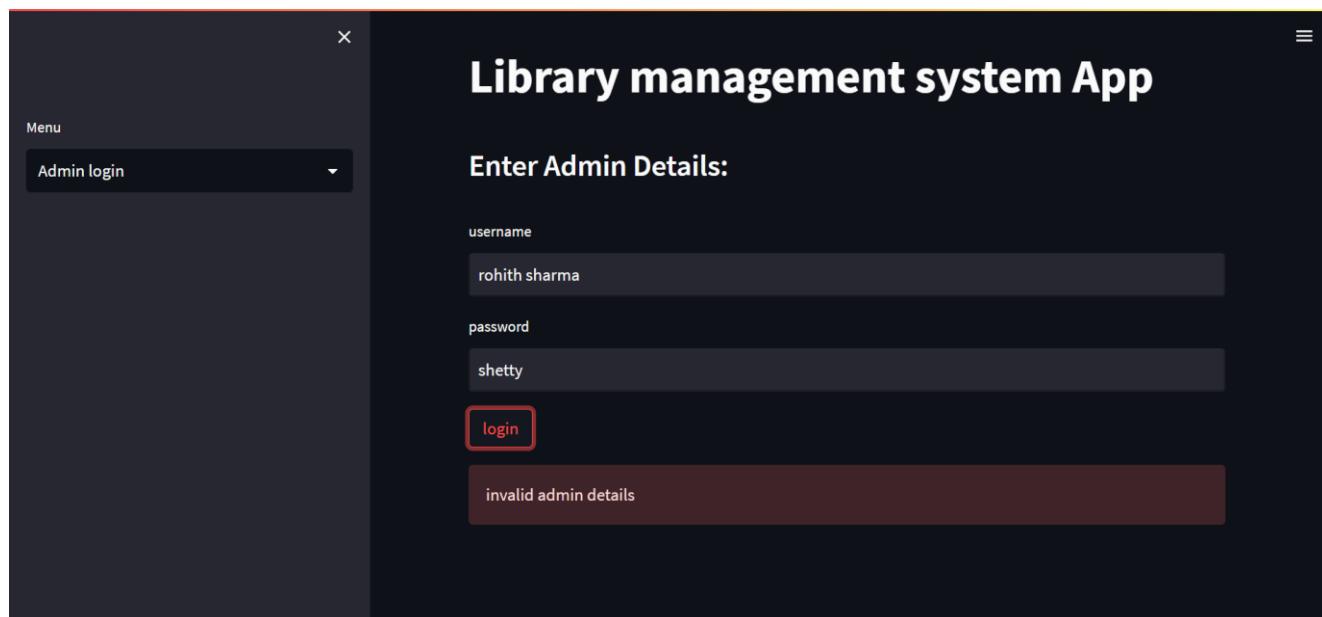
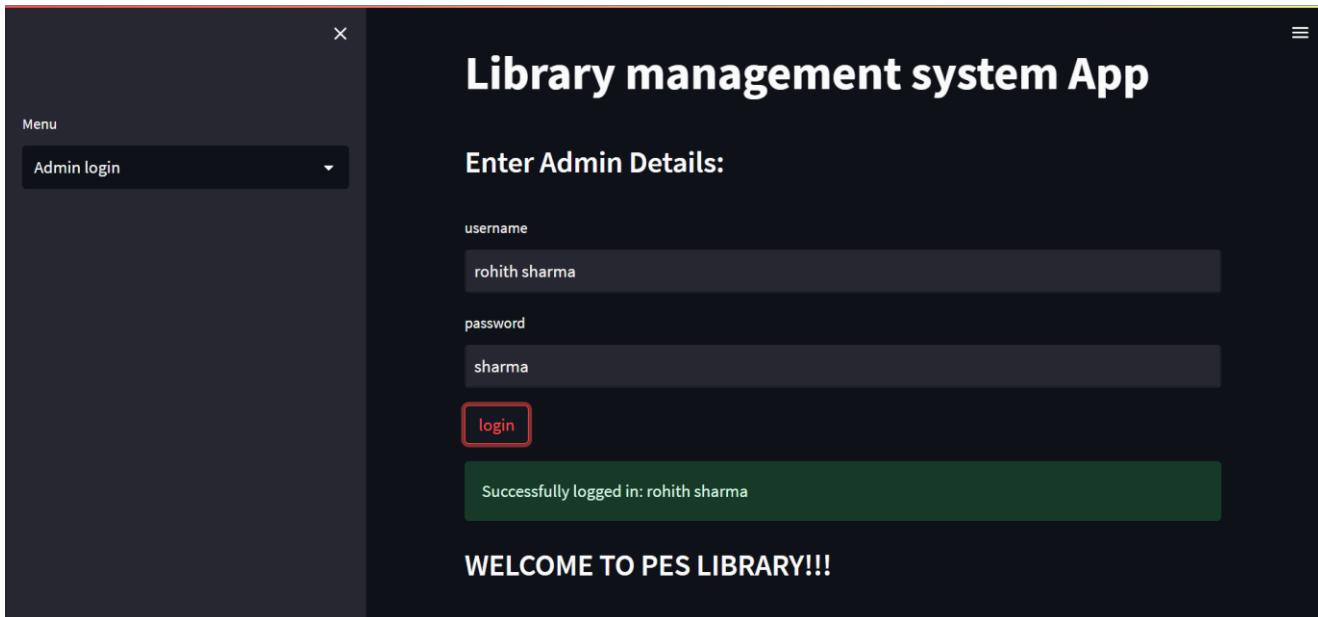
Do you want to delete: 33333

Delete User

successfully deleted user 33333

Admin Page:

1) Admin login:



## 2) Adding Books to the Database:

Add Books:

ISBN	status
b809080	yes
price	Title
5888	angels and demons
category	publisher
novel	1000
Author names:	
Dan Brown	
<input type="button" value="add book"/>	
successfully added the book angels and demons	

<input type="checkbox"/>	Edit	Copy	Delete	b777	454	novel	yes	wuthering heights	6767	2000
<input type="checkbox"/>	Edit	Copy	Delete	b809080	5888	novel	yes	angels and demons	2323	1000
<input type="checkbox"/>	Edit	Copy	Delete	b826	222	fiction	no	harry potter and the chamber of secrets	4545	1000

phpMyAdmin

Server: 127.0.0.1 » Database: my\_project » Table: author\_names

f_name	l_name	author_id
joseph	nizza	149
Wenliang	Du	159
Roald	Dahl	169
Rick	Riordan	179
Ramez	Elamsri	189
Jonathan	Katz	199
Yehuda	Lindell	209
Don	Norman	219
Robert	Sibesta	229
djdj	nnnn	631
kknk	nnnn	849
ffff	ijji	797
knkmk	dffd	809
Dan	Brown	287

Show all Number of rows: All Filter rows: Search this table Sort by key: None

Query results operations Console

phpMyAdmin

Server: 127.0.0.1 » Database: my\_project » Table: book\_names

name	author_id
the bfg	169
matilda	169
the design of everyday things	219
concepts of programming languages	229
ram:scion of ikshvaku	99
sita:warrior of mithila	99
raavan:the enemy of aryavarta	99
the immortals of meluha	99
james and the giant peach	169
the oath of vayuputras	99
artificial intelligence	44
intoduction to modern cryptography	209
angels and demons	287

Show all | Number of rows: All | Filter rows: Search this table | Sort by key: None

Query results operations

Console

### 3) Delete a book from the library database:

Menu

Admin login

Delete Book:

Books

angels and demons

Delete book

successfully deleted the book angels and demons

Issue books:

Customer ID: \_\_\_\_\_ Issue Date: 2022/11/19

ISBN of the book: \_\_\_\_\_ Return Date: 2022/11/19

**phpMyAdmin**

Recent Favorites

Server: 127.0.0.1 » Database: my\_project » Table: book\_names

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking

name author\_id

fundamentals of database systems	189
percy jackson and the sea of monsters	179
the 3 mistakes of my life	22
introduction to modern cryptography	199
the bfg	169
matilda	169
the design of everyday things	219
concepts of programming languages	229
ram:scion of ikshvaku	99
sita:warrior of mithila	99
raavan:the enemy of aryavarta	99
the immortals of meluha	99
james and the giant peach	169
the oath of vayupratas	99
artificial intelligence	44
introduction to modern cryptography	209

Show all | Number of rows: All | Filter rows: Search this table Sort by key None

localhost/phpmyadmin/index.php?route=/sql&db=my\_project&table=author&pos=0&sql\_signature=39aba87d07223cb6d7b6fc54e55645f851a356c7ba7780b2beaa663e9966&sql\_query=SELECT+%2A+FROM+my\_project%60%60author%60+WHERE+author\_id%60%60+%

**phpMyAdmin**

Recent Favorites

Server: 127.0.0.1 » Database: my\_project » Table: books

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking

ISBN price category status title admin\_id publisher\_id

b555	1546	academics	no	artificial intelligence	5656	4000
b667	1444	novel	yes	the immortals of meluha	4433	4000
b723	899	novel	yes	the oath of vayupratas	7878	2000
b760	1233	academics	yes	introduction to modern cryptography	7878	4000
b767	1555	academics	yes	computer and internet security	9090	2000
b769	2222	novel	yes	sita:warrior of mithila	2323	2000
b774	243	academics	yes	concepts of programming languages	2323	4000
b777	454	novel	yes	wuthering heights	6767	2000
b826	222	fiction	no	harry potter and the chamber of secrets	4545	1000
b908	1800	academics	no	machine learning	8989	1000
b986	543	fiction	no	revolution 2020	3434	2000
b998	2132	kids	yes	the adventures of tintin	8989	4000
b999	1090	novel	yes	the 3 mistakes of my life	1212	3000

Check all With selected: Edit Copy Delete Export

Console

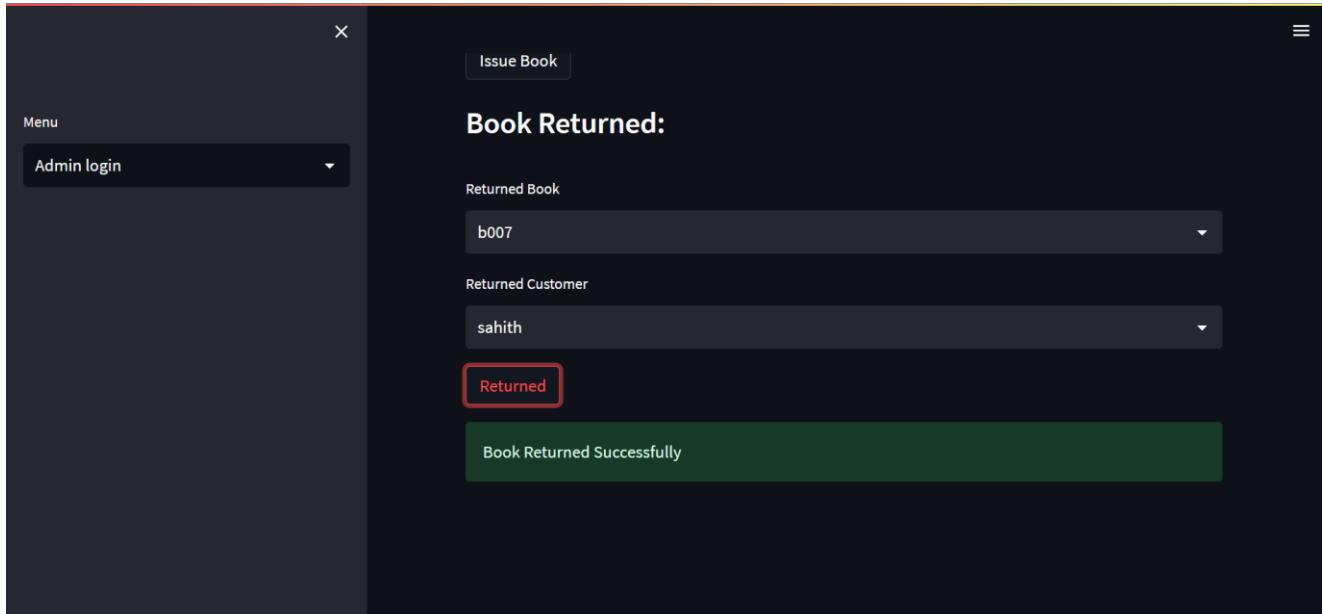
## 4) Issuing books to customers:

The screenshot shows a dark-themed web application interface. On the left, there is a sidebar with a "Menu" section containing a dropdown menu with "Admin login". The main content area has a title "Issue books:". It contains two sets of input fields: "Customer ID" (99999) and "Issue Date" (2022/11/19), and "ISBN of the book" (b007) and "Return Date" (2022/11/26). Below these fields is a red-bordered button labeled "Issue Book". Underneath the button, a green bar displays the message "Book Issued".

The screenshot shows the "issue\_return" table from the "my\_project" database in phpMyAdmin. The table has columns: customer\_id, admin\_id, ISBN, issue\_date, return\_date, and fine. The data is as follows:

customer_id	admin_id	ISBN	issue_date	return_date	fine
11111	2323	b222	2022-11-12	2022-11-19	0
11111	4545	b986	2022-11-12	2022-11-19	0
22222	7878	b152	2022-11-03	2022-11-10	27
22222	8989	b565	2022-11-03	2022-11-10	27
44444	5656	b777	2022-11-05	2022-11-12	21
55555	8989	b428	2022-11-08	2022-11-15	12
66666	4433	b452	2022-11-12	2022-11-19	0
66666	3434	b908	2022-11-12	2022-11-19	0
77777	4433	b007	2022-11-18	2022-11-25	0
87456	3434	b545	2022-11-06	2022-11-13	18
88888	9090	b999	2022-11-11	2022-11-18	3
13234	1212	b324	2022-11-01	2022-11-08	33
18273	8989	b037	2022-11-16	2022-11-23	0
18273	2323	b152	2022-11-05	2022-11-12	21
99999	2323	b007	2022-11-19	2022-11-26	0

## 5) Returning Book back to the library:



The screenshot shows the 'issue\_return' table in the 'my\_project' database via phpMyAdmin. The table has columns: customer\_id, admin\_id, ISBN, issue\_date, return\_date, and fine. The data shows various entries of books being issued and returned, along with their respective dates and fines.

customer_id	admin_id	ISBN	issue_date	return_date	fine
11111	2323	b222	2022-11-12	2022-11-19	0
11111	4545	b986	2022-11-12	2022-11-19	0
22222	7878	b152	2022-11-03	2022-11-10	27
22222	8989	b565	2022-11-03	2022-11-10	27
44444	5656	b777	2022-11-05	2022-11-12	21
55555	8989	b428	2022-11-08	2022-11-15	12
66666	4433	b452	2022-11-12	2022-11-19	0
66666	3434	b908	2022-11-12	2022-11-19	0
77777	4433	b007	2022-11-18	2022-11-25	0
87456	3434	b545	2022-11-06	2022-11-13	18
88888	9090	b999	2022-11-11	2022-11-18	3
13234	1212	b324	2022-11-01	2022-11-08	33
18273	8989	b037	2022-11-16	2022-11-23	0
18273	2323	b152	2022-11-05	2022-11-12	21

# Create New User:

The screenshot shows the 'Library management system App' interface. On the left, a dark sidebar labeled 'Menu' has a 'New user' option selected. The main area is titled 'Library management system App' and contains the heading 'Create new account:'. Below this, there are two rows of input fields. The first row contains 'customer\_id' (990099) and 'street' (Wall Street). The second row contains 'f\_name' (Harvey) and 'city' (New York). There are also fields for 'l\_name' (Spector), 'pin' (298298), 'register date' (2022/11/19), and 'select manager' (a dropdown menu showing the value '1').

This screenshot shows the continuation of the user creation process. The 'New user' option is still selected in the sidebar. The main form now includes additional fields: 'l\_name' (Spector), 'pin' (298298), 'register date' (2022/11/19), 'select manager' (dropdown showing '1'), 'password' (Harvey Reginald Spector), and 'enter\_password\_again' (Harvey Reginald Spector). A 'Register' button is present. A green success message at the bottom states 'Successfully registered : Harvey Spector'. At the very bottom of the screen, a table displays a list of users with columns: checkbox, Edit, Copy, Delete, ID, Name, Last Name, Register Date, and Manager. The first user listed is 'sahith' with ID 99999, and the second user listed is 'Harvey Spector' with ID 990099.

<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	99999	sahith	reddy	11	nandyal	659766	2022-10-02	3	sahith
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	990099	Harvey	Spector	Wall Street	New York	298298	2022-11-19	1	Harvey Reginald Spector

# Frontend Code:

## Creating a Database:

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password=""
)
c = mydb.cursor()

c.execute("CREATE DATABASE pycharm_lab10_assignment_pes2ug20cs390")
```

## App.py:

```
import streamlit as st
import mysql.connector

from user_login_create import create
from user_login_delete import delete
from manager_login_create import manager_login
from add_new_user import add_user
from admin_login_page import admin_login
from user_login_read import read

def main():
    st.title("Library management system App")
    menu = ["User login", "Manager Login", "Admin login", "New user"]
    choice = st.sidebar.selectbox("Menu", menu)

    if choice == "User login":
        st.subheader("Enter User Details:")
        create()

    elif choice == "Manager Login":
        st.subheader("Enter Manager details: ")
        manager_login()

    elif choice == "Admin login":
        st.subheader("Enter Admin Details: ")
        admin_login()
```

```

    elif choice == "New user":
        st.subheader("Create new account: ")
        add_user()

    else:
        st.subheader("About tasks")

if __name__ == '__main__':
    main()

```

## user\_login\_create.py:

```

import streamlit as st
import pandas as pd
from user_login_read import read
from user_login_database import
check_login,check_issued_books,get_customer_id,display_all_books,get_book_by_name,get
_author_by_name,get_book_by_category,customer_info,user_change_password

def callback():
    st.session_state.button_clicked=True
def create():
    if 'button_clicked' not in st.session_state:
        st.session_state.login=False
    username = st.text_input("username").split(' ')
    password = st.text_input("password")
    if st.button("login",on_click=callback) or st.session_state.button_clicked:
        if check_login(username[0],username[1],password):
            st.success("Successfully logged in: {}".format(username[0]+
'+username[1])))
            st.subheader('\t\tWELCOME TO PES LIBRARY!!!')
            st.subheader('Personal info:')
            customer_id = get_customer_id(username[0], username[1])
            result2 = customer_info(username[0], username[1],customer_id)
            df3 = pd.DataFrame(result2,columns=['customer_id','f_name', 'l_name',
'contact_no', 'state', 'city', 'street','pin'])
            st.dataframe(df3)
            read(username[0],username[1])
            st.subheader('Check Book By Book Name:')
            book = st.text_input("book_name")
            if st.button('check book'):
                if get_book_by_name(book):
                    st.success("{} exists in library".format(book))

```

```
        else:
            st.error("book does not exist")
    st.subheader('Check Book By Author Name:')
    author = st.text_input("author_name").split(' ')
    if st.button('check author'):
        try:
            result3= get_author_by_name(author[0],author[1])

            if result3:
                df1 = pd.DataFrame(result3, columns=['ISBN', 'Price',
'category', 'status', 'title'])
                with st.expander("all books"):
                    st.dataframe(df1)
            else:
                st.error('invalid author\'s name')
        except IndexError:
            st.error("Enter full name of the author")
    st.subheader('Check Book Based On Category:')
    category =
st.selectbox("category",["kids","academics","novel","fiction"])
    if st.button('find'):
        result3 = get_book_by_category(category)
        df1 = pd.DataFrame(result3, columns=['ISBN', 'Price','status',
'title'])
        with st.expander("all books"):
            st.dataframe(df1)
    st.subheader("Change Password")
    password = st.text_input('New Password')
    new_password = st.text_input('Retype Password')
    if st.button('Change'):
        if password == new_password:
            user_change_password(password, customer_id)
            st.success("Password changed")
        else:
            st.error("error occurred :(")
    else:
        st.error('invalid user')
```

## user\_login\_read.py:

```
import pandas as pd
import streamlit as st
from user_login_database import get_customer_id,display_all_books,check_issued_books

def read(username1,username2):
    customer_id = get_customer_id(username1, username2)
    result = check_issued_books(customer_id)
    df = pd.DataFrame(result, columns=['book name', 'ISBN', 'price'])
    st.subheader('Issued Books:')
    with st.expander("Issued Books"):
        st.dataframe(df)
    result1 = display_all_books()
    st.subheader('All Books:')
    df1 = pd.DataFrame(result1, columns=['ISBN', 'Price', 'category', 'status',
'title', 'admin id', 'publisher id'])
    with st.expander("all books"):
        st.dataframe(df1)
```

## user\_login\_database.py:

```
# pip install mysql-connector-python
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="my_project"
)
c = mydb.cursor()

def customer_info(username1,username2,customer_id):
    c.execute('select customer_id,f_name,l_name,street,city,pin,reg_date,manager_id
from `customer` where f_name="{}" and l_name="{}" and
customer_id="{}"'.format(username1, username2,customer_id))
    data = c.fetchall()
    return data

def check_login(username1,username2,password):
    c.execute('SELECT f_name,l_name, password from `customer` where f_name=%s and
l_name=%s and password=%s',
              (username1.username2.password))
```

```
data = c.fetchall()
try:
    if data[0][2]==password and username1==data[0][0] and username2==data[0][1]:
        return 1
    else:
        return 0
except IndexError:
    return 0
def get_customer_id(username1,username2):
    c.execute('select customer_id from `customer` where f_name=%s and
l_name=%s',(username1,username2))
    data = c.fetchone()
    return data[0]
def check_issued_books(customer_id):
    c.execute('select books.title,issue_return.ISBN,books.price from `issue_return`'
INNER join `books` on issue_return.ISBN=books.ISBN where
customer_id="{}''.format(customer_id))
    data=c.fetchall()
    return data
def display_all_books():
    c.execute('SELECT * FROM `books` ')
    data = c.fetchall()
    return data

def get_book_by_name(book_name):
    c.execute('Select title from `books` where title="{}''.format(book_name)')
    data = c.fetchall()
    if data:
        return 1
    else:
        return 0
def get_author_by_name(f_name,l_name):
    c.execute('select ISBN,price,category,status,title from `books` where books.title
in (Select name from `book_names` natural join `author_names` where
author_names.f_name=%s and author_names.l_name=%s)',(f_name,l_name))
    data = c.fetchall()
    return data

def get_book_by_category(category):
    c.execute('select ISBN,price,status,title from `books` where
    books.category="{}''.format(category))
    data = c.fetchall()
    return data
```

```

c.execute('update `customer` set password="{}" where
          customer_id="{}"'.format(password, customer_id))
mydb.commit()

```

## manager\_login\_delete.py:

```

import streamlit as st
import pandas as pd
from manager_login_database import
manager_login_check,find_manager_id,admins_check,manager_info,add_admin,view_only_adm
in_id,delete_admin,view_only_cust_id,delete_user

def callback():
    st.session_state.button_clicked=True

def manager_login():
    if 'button_clicked' not in st.session_state:
        st.session_state.login=False
    username = st.text_input("username").split(' ')
    password = st.text_input("password")
    if st.button("login",on_click=callback) or st.session_state.button_clicked:
        if manager_login_check(username[0],username[1],password):
            st.success("Successfully logged in: {}".format(username[0]+
'+username[1])))
                st.subheader('\t\tWELCOME TO PES LIBRARY!!!')
                st.subheader('Personal info:')
                manager_id = find_manager_id(username[0], username[1])
                result2 = manager_info(username[0],username[1],manager_id)
                df3 =
pd.DataFrame(result2,columns=['manager_id','f_name','l_name','contact_no','state','ci
ty','street','pin'])
                st.dataframe(df3)
                st.subheader("check admins: ")
                result3 = admins_check(manager_id)
                if result3:
                    df1 = pd.DataFrame(result3,
columns=['admin_id','f_name','l_name','salary'])
                    with st.expander("admins"):
                        st.dataframe(df1)
                else:
                    st.error("admins dosen't exist")

```

st.subheader('Add admins:')

```
col1,col2 =st.columns(2)
with col1:
    admin_id = st.text_input("admin_id")
    salary = st.text_input("salary")
with col2:
    f_name = st.text_input("f_name")
    l_name = st.text_input("l_name")
if st.button("Add Admin"):
    add_admin(admin_id,salary,f_name,l_name,manager_id)
    st.success("Successfully added Admin: {}".format(f_name+' '+l_name))

st.subheader('delete')
result4 = [i[0] for i in view_only_admin_id(manager_id)]
selected_admin = st.selectbox("Admin to Delete", result4)
st.warning("Do you want to delete :{}".format(selected_admin))
if st.button("Delete Admin"):
    delete_admin(selected_admin)
    st.success("successfully deleted admin {}".format(selected_admin))

st.subheader('Delete Customers:')
result5 = [i[0] for i in view_only_cust_id()]
selected_user = st.selectbox('Customer to delete',result5)
st.warning("Do you want to delete: {}".format(selected_user))
if st.button("Delete User"):
    delete_user(selected_user)
    st.success("successfully deleted user {}".format(selected_user))

else:
    st.error('invalid details ')
```

## manager\_login\_database.py:

```
import mysql.connector
import random
import streamlit as st
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="my_project"
)
c = mydb.cursor()

def manager_info(username1,username2,customer_id):
    c.execute('select manager_id,f_name,l_name,contact_no,state,city,street,pin from `branch` where f_name="{}" and l_name="{}" and manager_id="{}"'.format(username1,
username2,customer_id))
    data = c.fetchall()
    return data

def manager_login_check(username1,username2,password):
    c.execute('SELECT f_name,l_name, password from `branch` where f_name=%s and
l_name=%s and password=%s',
              (username1, username2, password))
    data = c.fetchall()
    try:
        if data[0][2] == password and username1==data[0][0] and
username2==data[0][1]:
            return 1
        else:
            return 0
    except IndexError:
        return 0

def find_manager_id(f_name,l_name):
    c.execute('SELECT manager_id from `branch` where f_name=%s and l_name=%s',
              (f_name,l_name))
    data=c.fetchall()
    return data[0][0]

def admins_check(cust_id):
    c.execute('select admins.admin_id,admins.f_name,admins.l_name,admins.salary from `branch` inner join `admins` where branch.manager_id=admins.manager_id and
branch.manager_id="{}"'.format(cust_id))
    data=c.fetchall()
    return data
```

```

def add_admin(admin_id,salary,f_name,l_name,manager_id):
    c.execute('INSERT INTO admins(admin_id,salary,f_name,l_name,manager_id,password)
VALUES ("{}", "{}", "{}", '
        '"{}", "{}", "{}")'.format(admin_id,salary,f_name,l_name,manager_id,f_name))
    mydb.commit()

def view_only_admin_id(man_id):
    c.execute('select admin_id from `admins` where manager_id="{}"'.format(man_id))
    data = c.fetchall()
    return data

def delete_admin(admin_id):
    c.execute('select admin_id from `admins` where admin_id!="{}"'.format(admin_id))
    data = c.fetchall()
    c.execute('SET foreign_key_checks = 0')
    c.execute('update `books` set admin_id="{}" where admin_id = '
    "{}".format(random.choice(data[0]),admin_id))
    c.execute('update `issue_return` set admin_id="{}" where admin_id = '
    "{}".format(random.choice(data[0]), admin_id))
    c.execute('delete from admins where admin_id="{}"'.format(admin_id))
    c.execute('SET foreign_key_checks = 1')
    mydb.commit()

def view_only_cust_id():
    c.execute('select customer_id from customer')
    data=c.fetchall()
    return data

def delete_user(cust_id):
    c.execute('SET foreign_key_checks = 0')
    c.execute('delete from issue_return where customer_id="{}"'.format(cust_id))
    c.execute('delete from customer where customer_id="{}"'.format(cust_id))
    c.execute('SET foreign_key_checks = 1')
    mydb.commit()

```

## admin\_login\_page.py:

```
import streamlit as st

from admin_login_database import
admin_login_check,view_publisher_id,admin_id,add_book,get_all_books,delete_book,book_
issue,get_books_assigned_admin,get_customer_got_assigned_by_admin,book_returned

def callback():
    st.session_state.button_clicked=True

def admin_login():
    if 'button_clicked' not in st.session_state:
        st.session_state.login=False
    username = st.text_input("username").split(' ')
    password = st.text_input("password")
    if st.button("login",on_click=callback) or st.session_state.button_clicked:
        if admin_login_check(username[0], username[1], password):
            st.success("Successfully logged in: {}".format(username[0] + ' ' +
username[1]))
            st.subheader('\t\tWELCOME TO PES LIBRARY!!!')
            st.subheader('Add Books:')
            col1,col2 =st.columns(2)
            pub =  [i[0] for i in view_publisher_id()]
            admin = admin_id(username[0],username[1])
            with col1:
                ISBN = st.text_input("ISBN")
                price = st.text_input("price")
                category =
st.selectbox("category",['novel','fiction','academics','kids'])
            with col2:
                status = st.selectbox("status",["yes","no"])
                title = st.text_input("Title")
                publisher_id = st.selectbox('publisher',pub)
                Authors = st.text_input('Author names:').split(",")
                if st.button("add book"):
                    if
add_book(ISBN,price,category,status,title,admin,publisher_id,Authors):
                        st.success("successfully added the book {}".format(title))
                    else:
                        st.error("failed to update:")
            st.subheader('Delete Book:')
            result = get_all_books(admin)
            selected_book = st.selectbox('Books',result)
            if st.button("Delete book"):
                delete_book(selected_book)
```

```

        st.success("successfully deleted the book {}".format(title))
    st.subheader('Issue books:')
    col1,col2 = st.columns(2)
    with col1:
        customer_id = st.text_input('Customer ID')
        ISBN1 = st.text_input('ISBN of the book')
    with col2:
        issue_date = st.date_input('Issue Date:')
        return_date = st.date_input('Return Date:')
    if st.button('Issue Book'):
        book_issue(customer_id,ISBN1,issue_date,return_date,admin)
        st.success("Book Issued")
    st.subheader('Book Returned:')
    returned = get_books_assigned_admin(admin)
    returned_book = st.selectbox('Returned Book',returned)
    ret_cust = get_customer_got_assigned_by_admin(admin,returned_book)
    returned_customer = st.selectbox('Returned Customer',ret_cust)
    if st.button("Returned "):
        book_returned(returned_book,admin,returned_customer)
        st.success('Book Returned Successfully')
    else:
        st.error('invalid admin details ')

```

## admin\_login\_database.py:

```

import mysql.connector
import random
import streamlit as st
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="my_project"
)
c = mydb.cursor()

def admin_login_check(username1,username2,password):
    c.execute('SELECT f_name,l_name, password from `admins` where f_name="{}" and l_name="{}" and password="{}"'.format(username1, username2, password))
    data = c.fetchall()
    try:
        if data[0][2] == password and username1==data[0][0] and username2==data[0][1]:
            return 1
    except:
        return 0

```

```

        return 0
    except IndexError:
        return 0

def admin_info(username1,username2,customer_id):
    c.execute('select manager_id,f_name,l_name,contact_no,state,city,street,pin from `branch` where f_name="{}" and l_name="{}" and manager_id="{}"'.format(username1,username2,customer_id))
    data = c.fetchall()
    return data

def admin_id(username1,username2):
    c.execute('select admin_id from admins where f_name="{}" and l_name="{}"'.format(username1,username2))
    data = c.fetchall()
    return data[0][0]

def view_publisher_id():
    c.execute('select publisher_id from publisher')
    data = c.fetchall()
    return data

def add_book(ISBN,price,category,status,title,admin,publisher_id,authors):
    c.execute('select * from books where ISBN="{}"'.format(ISBN))
    data = c.fetchall()
    l=[]
    if data ==l:
        c.execute('Insert into books(ISBN,price,category,status,title,admin_id,publisher_id) values ("{}","{}","{}","{}","{}","{}","{}")'.format(ISBN, price, category, status, title, admin, publisher_id))
        for i in range(len(authors)):
            data1 = random.randint(220,1000)
            l.append(data1)
            c.execute('insert into author(author_id) values("{}")'.format(data1))

            for i in range(0,len(authors)):
                u = authors[i].split(" ")
                c.execute('insert into author_names(f_name,l_name,author_id) values ("{}","{}","{}")'.format(u[0],u[1],l[i]))
                for i in range(0,len(authors)):
                    c.execute('insert into book_names(name,author_id) values ("{}","{}")'.format(title,l[i]))

mydb.commit()
return 1

```

```

        else:

            return 0

def get_all_books(admin):
    c.execute('select title from `books` where admin_id="{}"'.format(admin))
    data=c.fetchall()
    l=[]
    for i in range(0,len(data)):
        l.append(data[i][0])
    return l

def delete_book(book):
    c.execute('delete from books where title="{}"'.format(book))
    c.execute('delete from book_names where name="{}"'.format(book))
    mydb.commit()

def book_issue(customer_id,ISBN1,issue_date,return_date,admin):
    c.execute('SET foreign_key_checks = 0')
    c.execute('insert into
issue_return(customer_id,admin_id,ISBN,issue_date,return_date)
values("{}", "{}", "{}", "{}", "{}")'.format(customer_id,admin,ISBN1,issue_date,return_da
te))
    c.execute('SET foreign_key_checks = 1')
    mydb.commit()

def get_books_assigned_admin(admin):
    c.execute('select ISBN from issue_return where admin_id="{}"'.format(admin))
    data=c.fetchall()
    l = []
    for i in range(0, len(data)):
        l.append(data[i][0])
    return l

def get_customer_got_assigned_by_admin(admin,returned):
    c.execute('select customer_id from issue_return where admin_id="{}" and
ISBN="{}"'.format(admin,returned))
    data = c.fetchall()
    l = []
    for i in range(0, len(data)):
        l.append(data[i][0])
    s=[]
    for i in range(0,len(l)):
        c.execute('select f_name from customer where
customer_id="{}"'.format(l[i]))
        data=c.fetchall()
        s.append(data[0][0])
    return s

```

```

def book_returned(ISBN,admin,customer):
    c.execute('SET foreign_key_checks = 0')
    c.execute('select customer_id from customer where f_name="{}"'.format(customer))
    data=c.fetchall()
    c.execute('delete from issue_return where customer_id="{}" and admin_id="{}" and ISBN="{}"'.format(data[0][0],admin,ISBN))
    c.execute('SET foreign_key_checks = 1')
    mydb.commit()

```

## add\_new\_user.py:

```

import streamlit as st
import pandas as pd
from add_new_user_database import register_new_user
def add_user():
    col1, col2,col3 = st.columns(3)
    with col1:
        customer_id = st.text_input("customer_id")
        f_name = st.text_input("f_name")
        l_name = st.text_input("l_name")
        reg_date = st.date_input("register date")
    with col2:
        street = st.text_input("street")
        city = st.text_input("city")
        pin = st.text_input("pin")
        manager_id = st.selectbox("select manager",["1","2","3","4","5"])
    password = st.text_input("password")
    check_password = st.text_input("enter_password_again")
    if st.button("Register"):
        try:
            if password==check_password:
                register_new_user(customer_id,f_name,l_name,street,city,pin,reg_date,manager_id,password)
                st.success("Successfully registered : {}".format(f_name + ' ' + l_name))
        except :
            st.error('invalid details')

```

## add\_new\_user\_database.py:

```
import mysql.connector
import streamlit as st
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="my_project"
)
c = mydb.cursor()

def
register_new_user(customer_id,f_name,l_name,street,city,pin,reg_date,manager_id,password):
    c.execute('INSERT INTO
customer(customer_id,f_name,l_name,street,city,pin,reg_date,manager_id,password)
VALUES ("{}", "{}", "{}", "{}", "{}", "{}")'.format(customer_id,f_name,l_name,street,city,pin,reg
_date,manager_id,password))
    mydb.commit()
```

## **12.Conclusion**

**I have achieved all the objectives of this project. I made the library management system easy and convenient to user, admin and branch managers. All the CRUD operations are made very easy for all the end users which was one of the main objectives of this project.**

## **References(in any)**

- 1) DBMS lab manuals, slides and notes provided by PES UNIVERSITY.