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

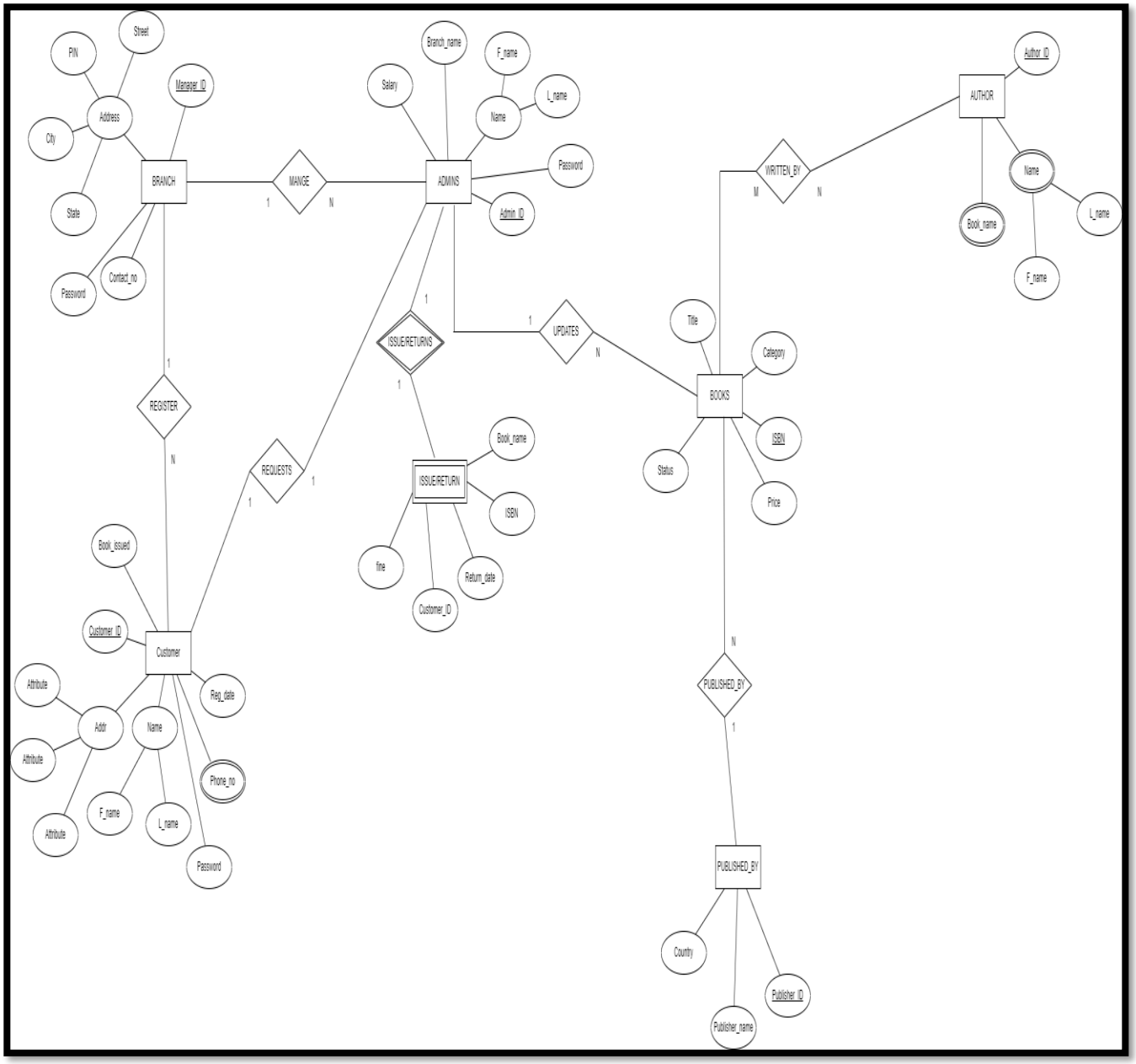
1. Short Description and Scope of the Project

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.

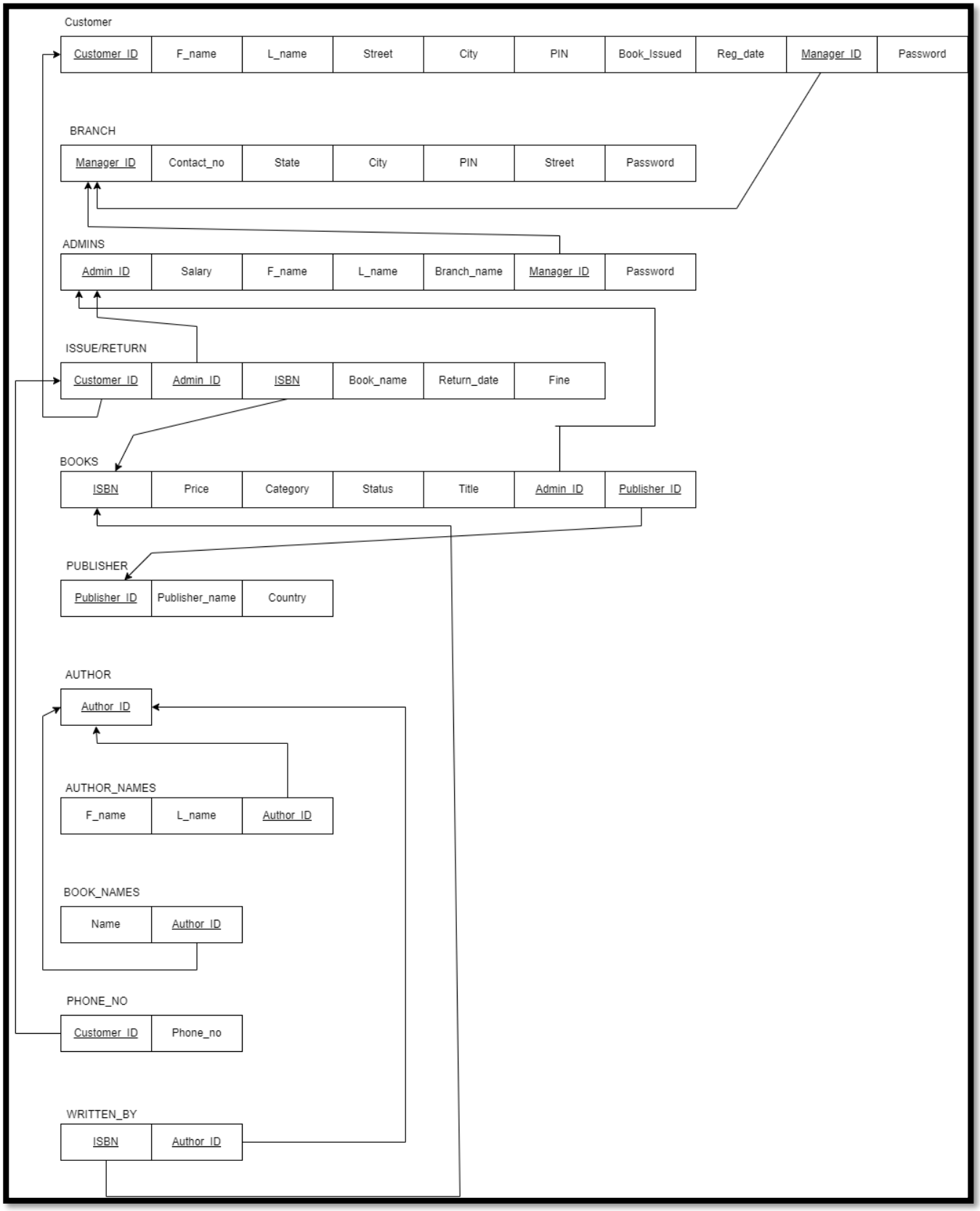
By using Python3(Streamlit) as frontend and keeping MySQL as the backend I am creating library database system. There will be several tables created inside the database which will represent each entity that I am going to create. These entities are related to each other by relationships and are mapped 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


Street
PIN
Address
City
State
Password
Contact_no
BRANCH
Manager_ID
MANGE
1
N
Branch_name
Salary
F_name
L_name
Name
Password
Admin_ID
ADMINS
1
ISSUE/RETURNS
1
UPDATES
1
N
REGISTER
1
N
REQUESTS
1
1
Book_name
ISSUE/RETURN
ISBN
fine
Return_date
Customer_ID
Book_issued
Customer_ID
Customer
Attribute
Addr
Attribute
Attribute
Name
F_name
L_name
Reg_date
Phone_no
Password
Author_ID
AUTHOR
Name
L_name
F_name
Book_name
WRITTEN_BY
M
N
Title
Category
BOOKS
ISBN
Status
Price
N
PUBLISHED_BY
1
PUBLISHED_BY
Country
Publisher_ID
Publisher_name

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, 'rohith', '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 theif', 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', 'udupi', 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),
('intoduction 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

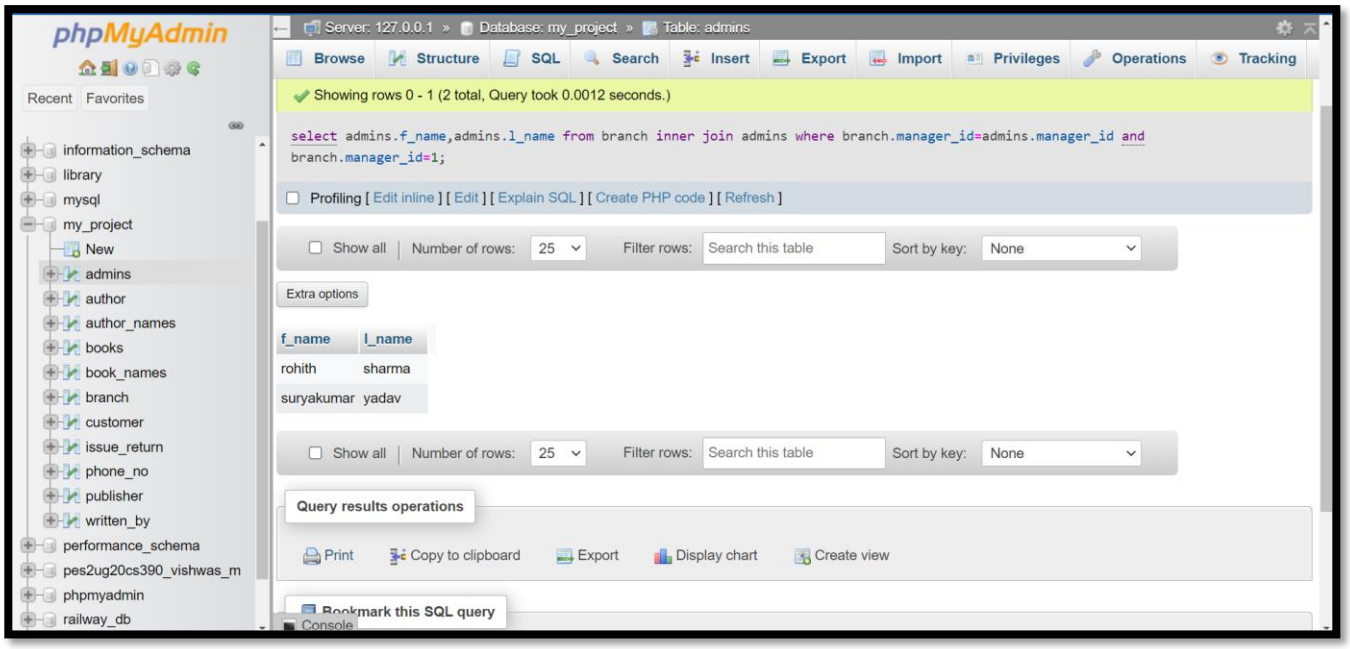
Showcase at least 4 join queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

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:

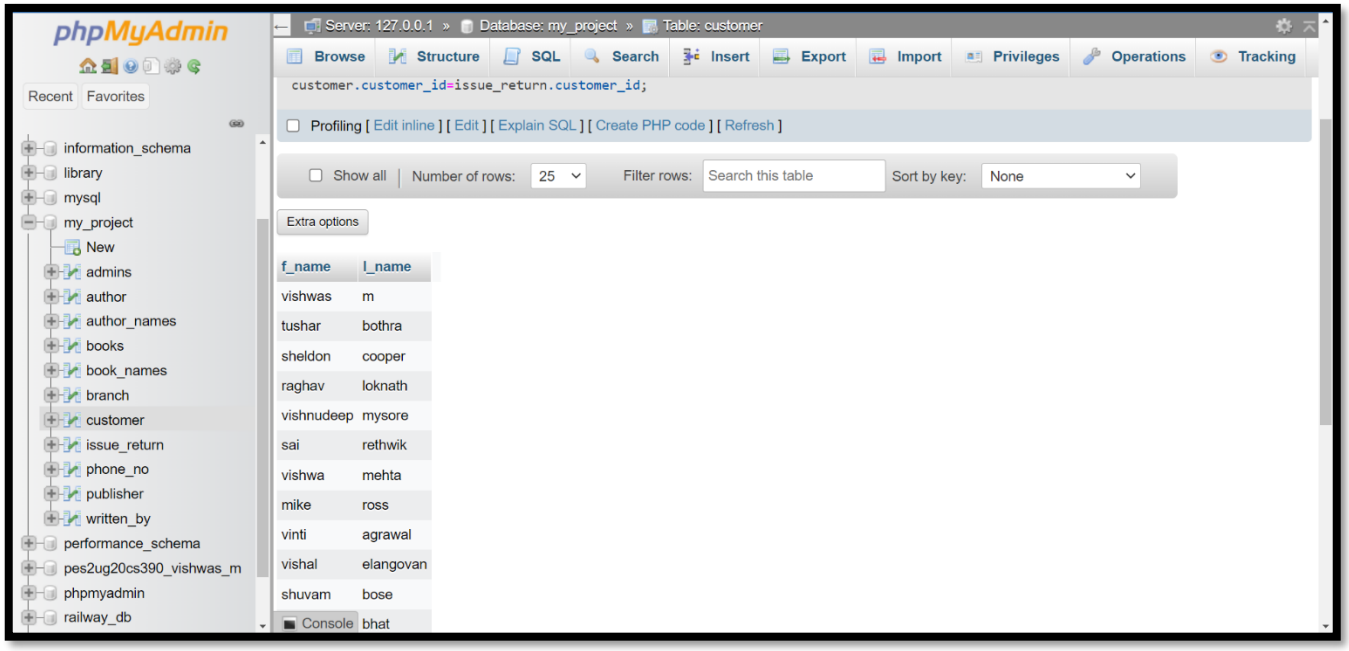


2) Customers who has issued one or more books in the library.

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:

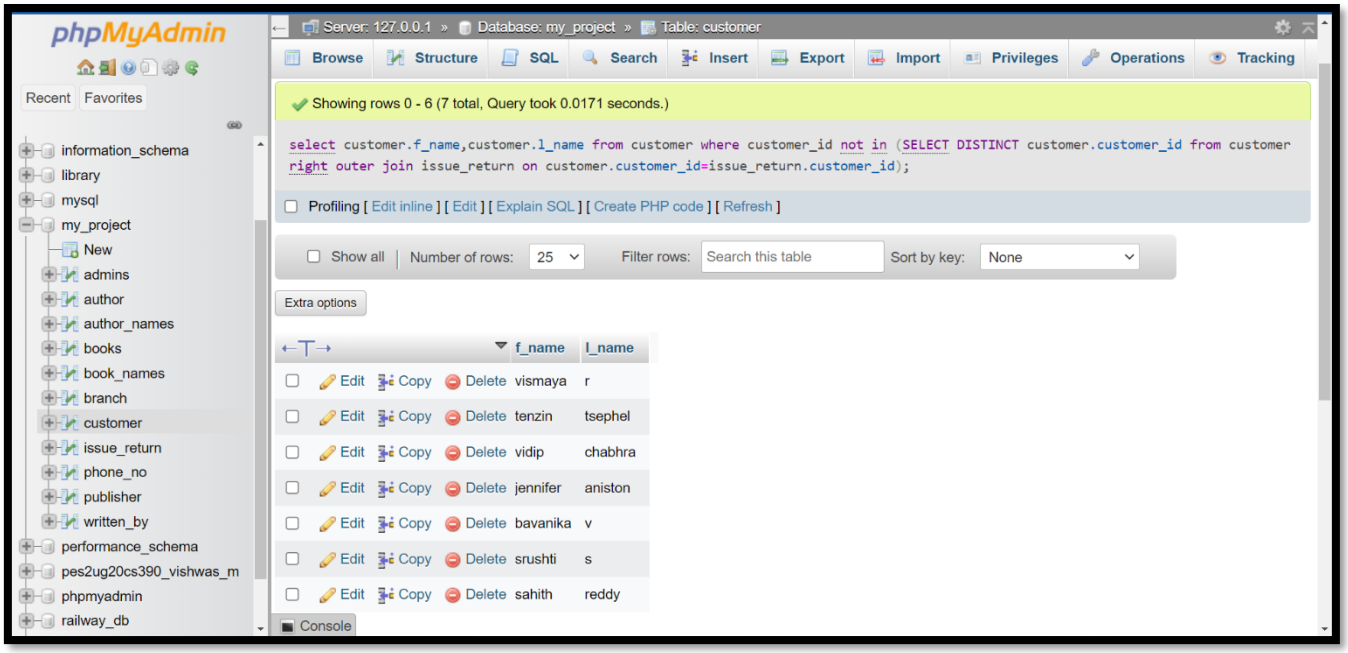


3) Customers who did not issue any of the books in the library.

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:

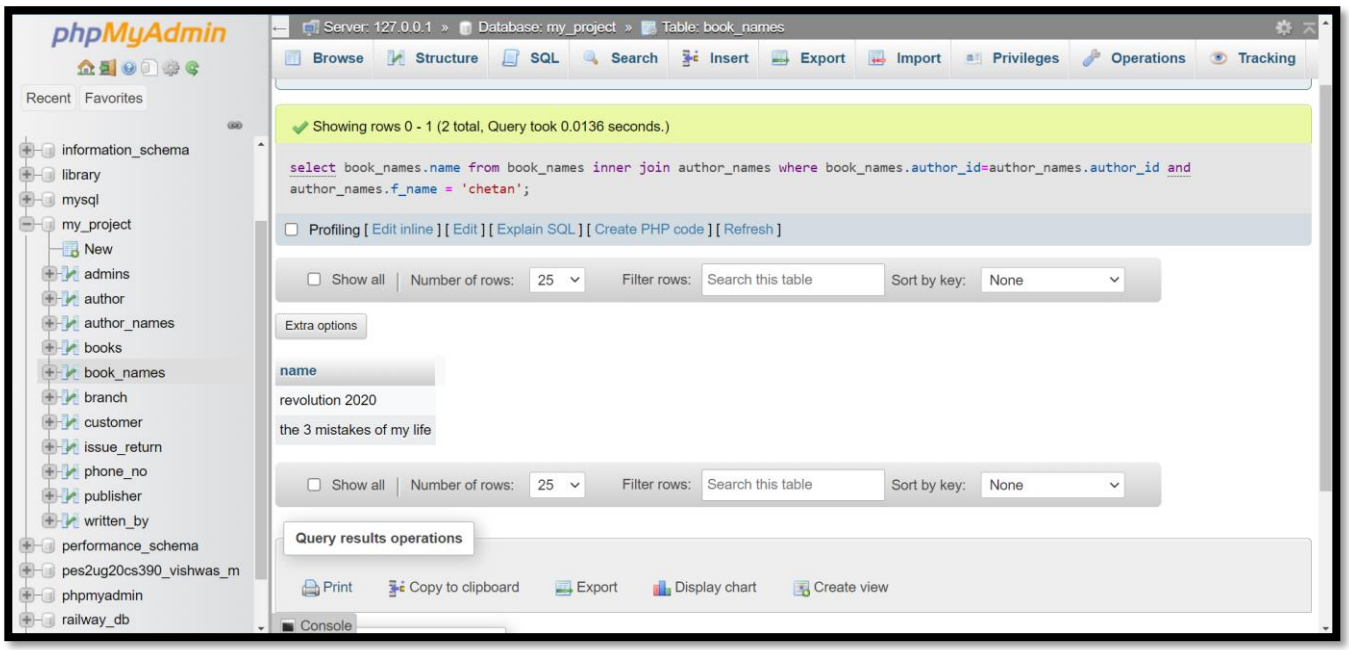


4) Display all the books written by the author whose first name is Chetan.

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:



7. Aggregate Functions

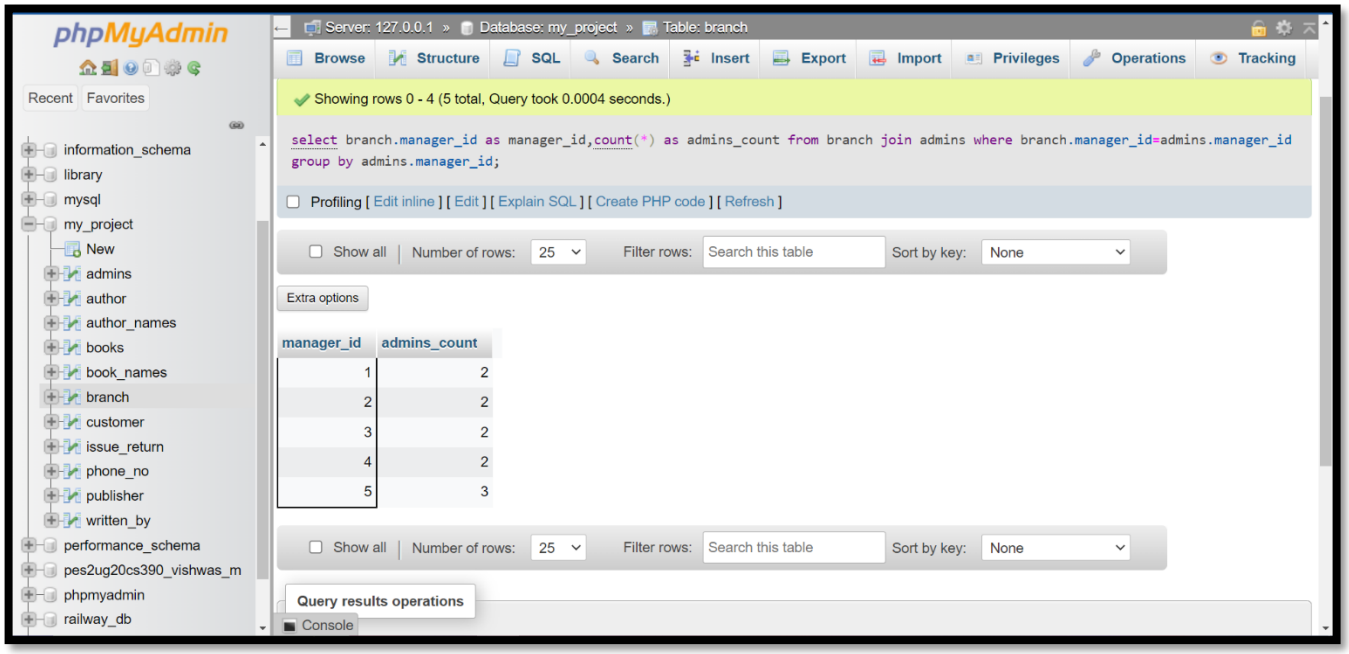
Showcase at least 4 Aggregate function queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

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:

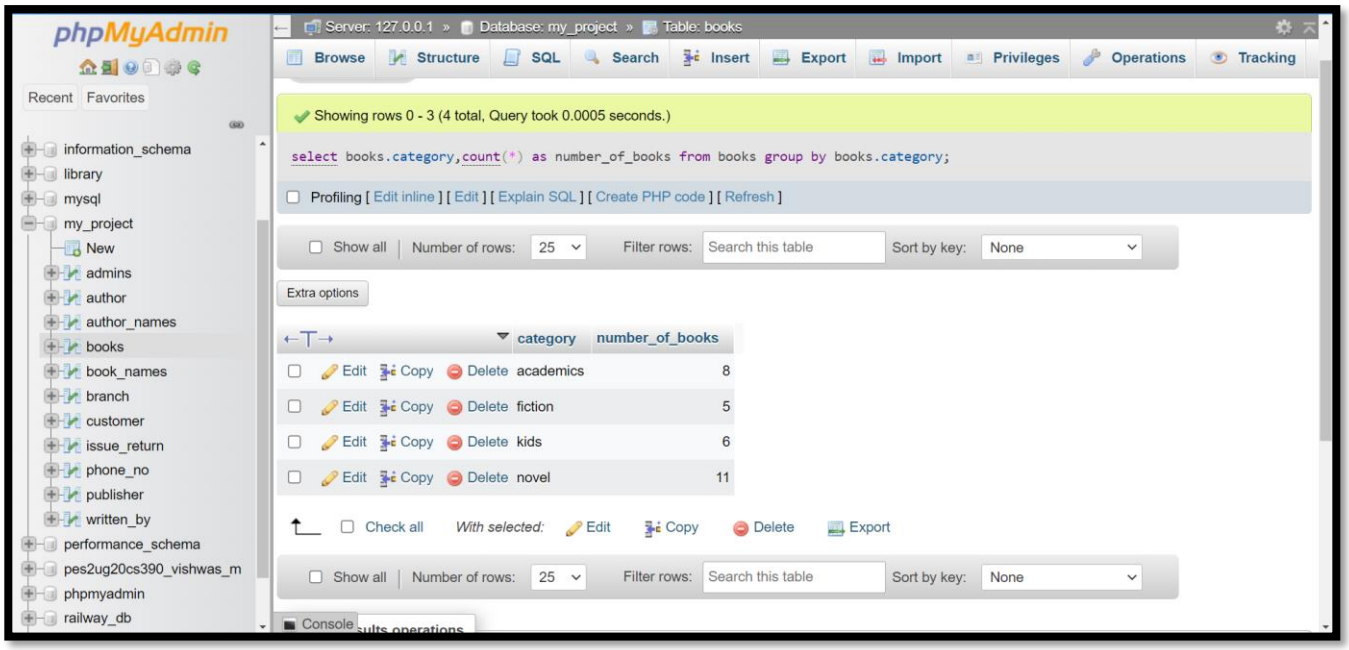


2) Find the number of books in each category of books.

Soln:

SQL Statement: select books.category,count(*) as number_of_books from books group by books.category

Output:



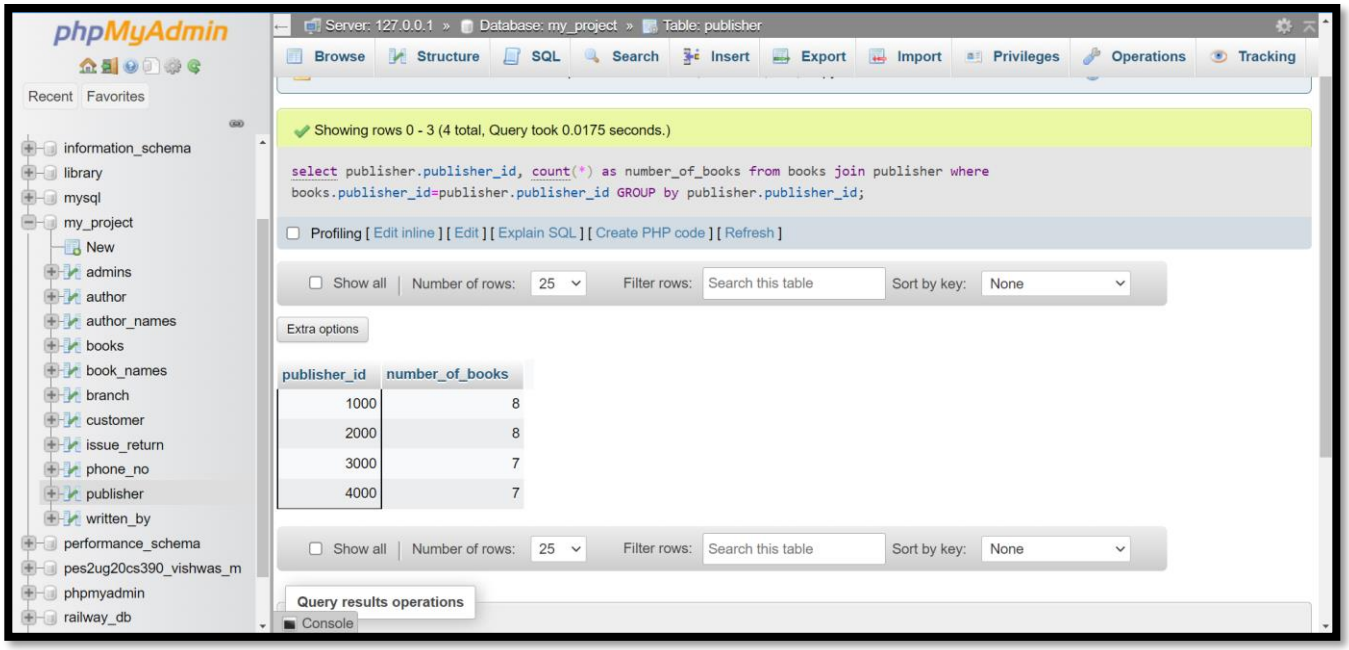
3) Find the number of books published by each publisher.

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

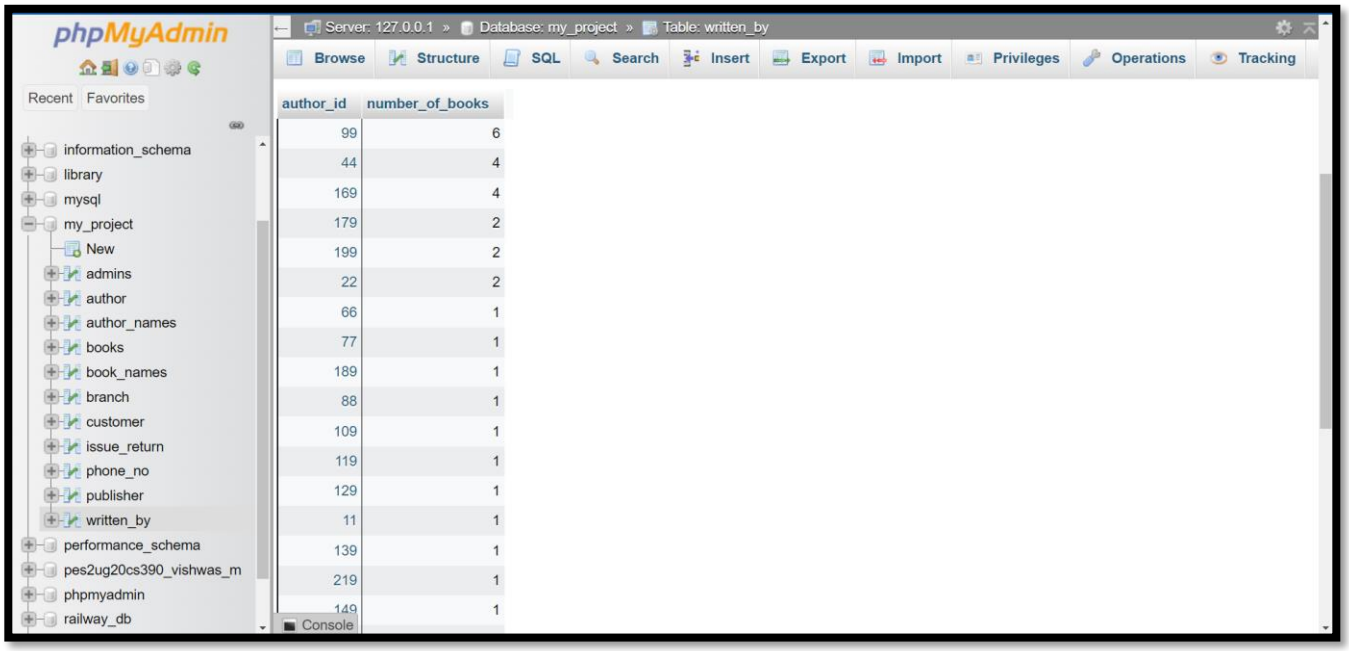


4) Find number of books written by each author.

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:



8. Set Operations

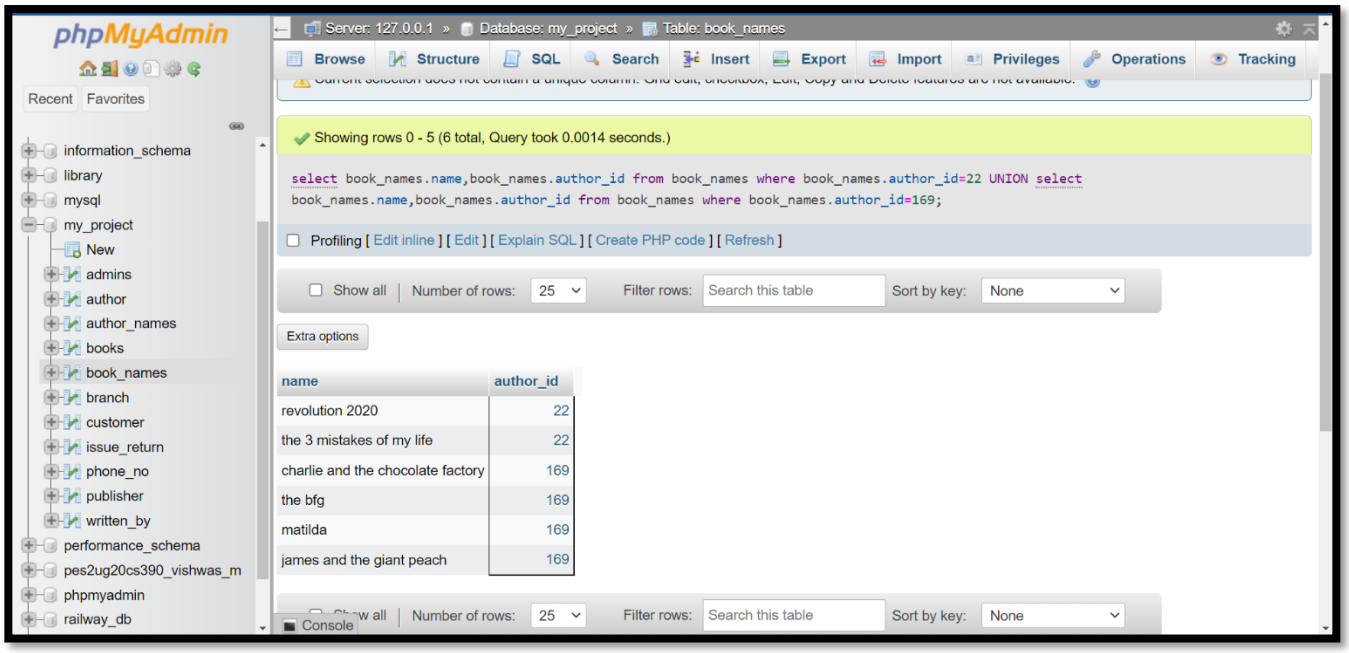
Showcase at least 4 Set Operations queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

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

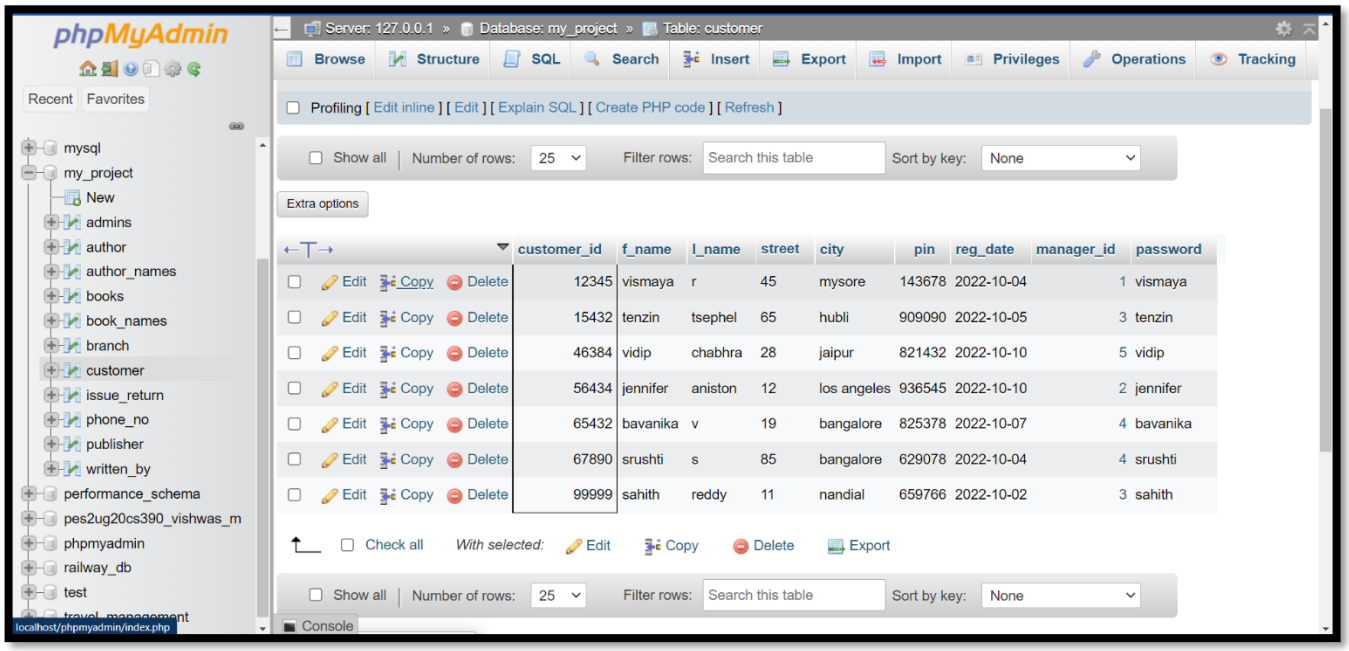
Output:



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)

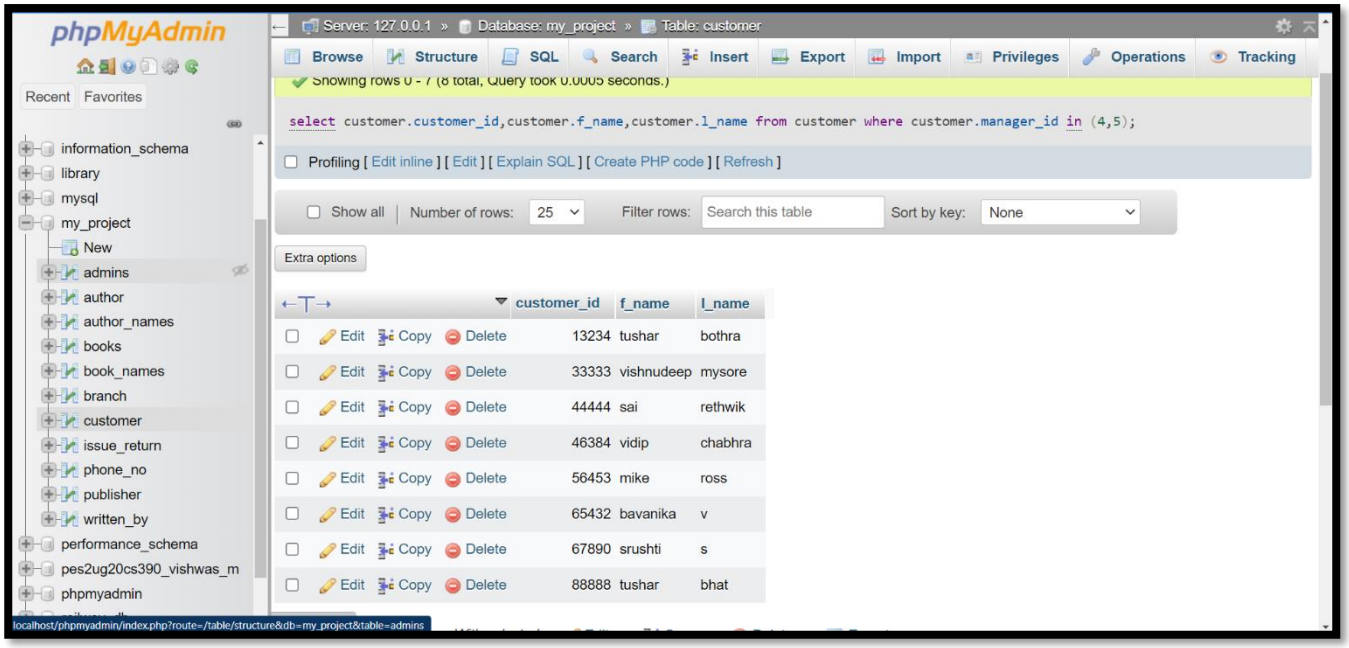


3) Find all the customers that come under the branch managers having manager_id 4 and 5.

Soln:

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

Output:

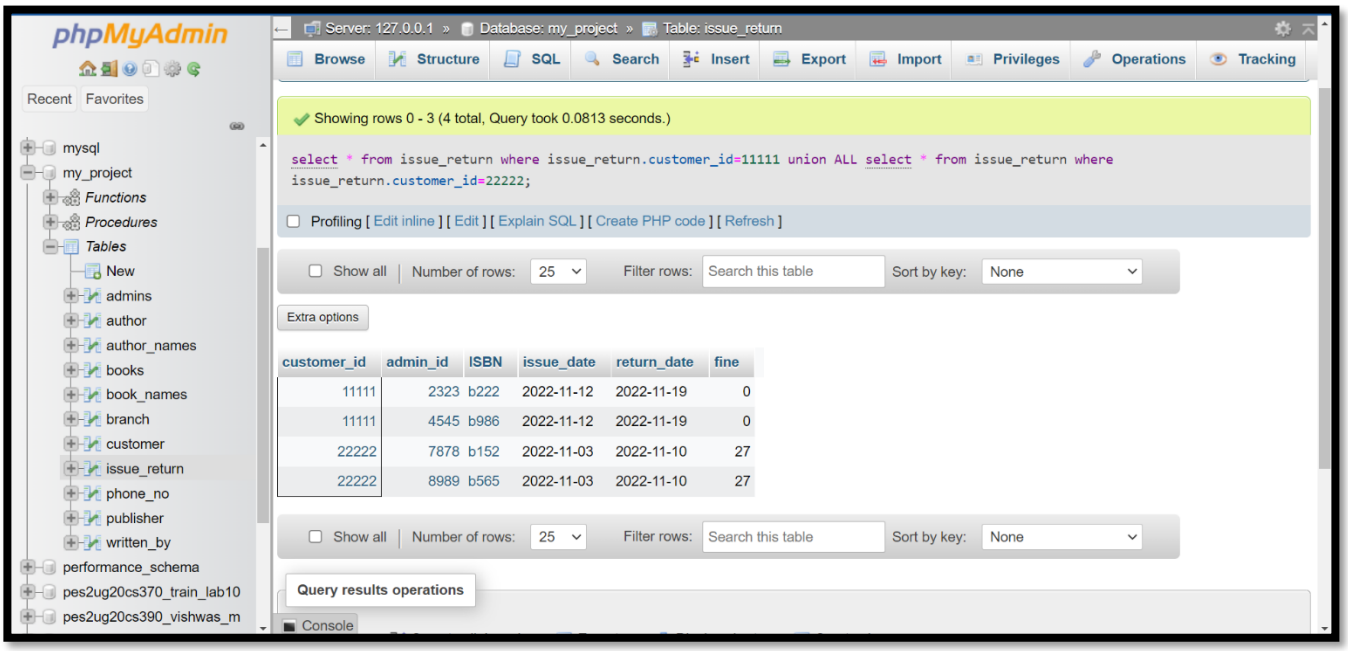


4) Find all the details regarding the books issued by the customer whose id's are 11111 and 22222.

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:

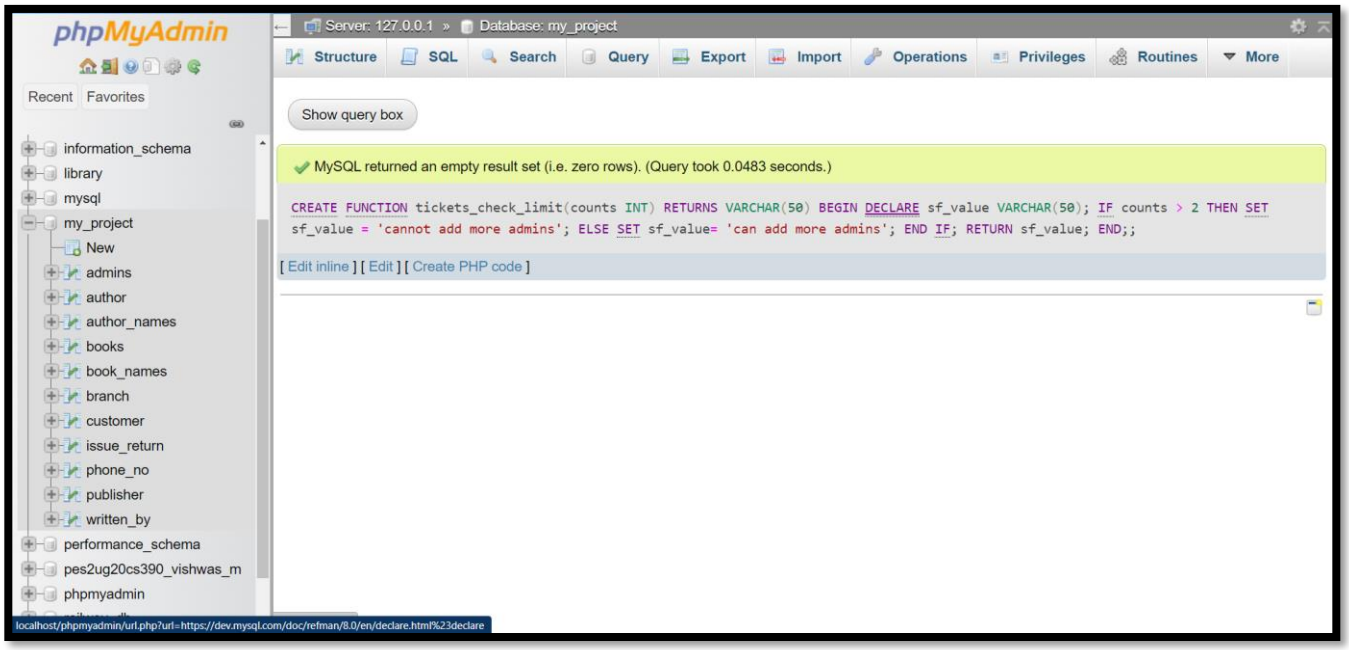


9. Functions and Procedures

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

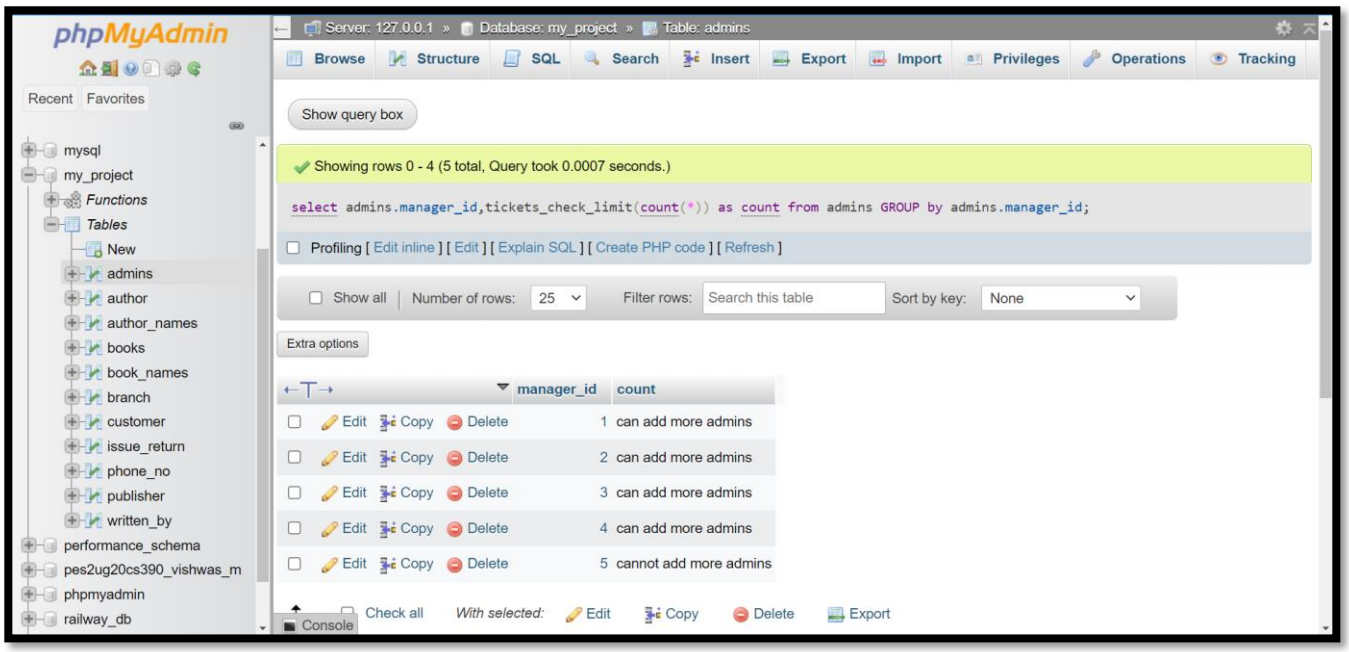
1) Create a function which checks whether the number of admins under a branch manager exceeds 2 or not. If exceeds more than 2 then we cannot add more admins under that manager else we can add more.

   Soln: Creating the function



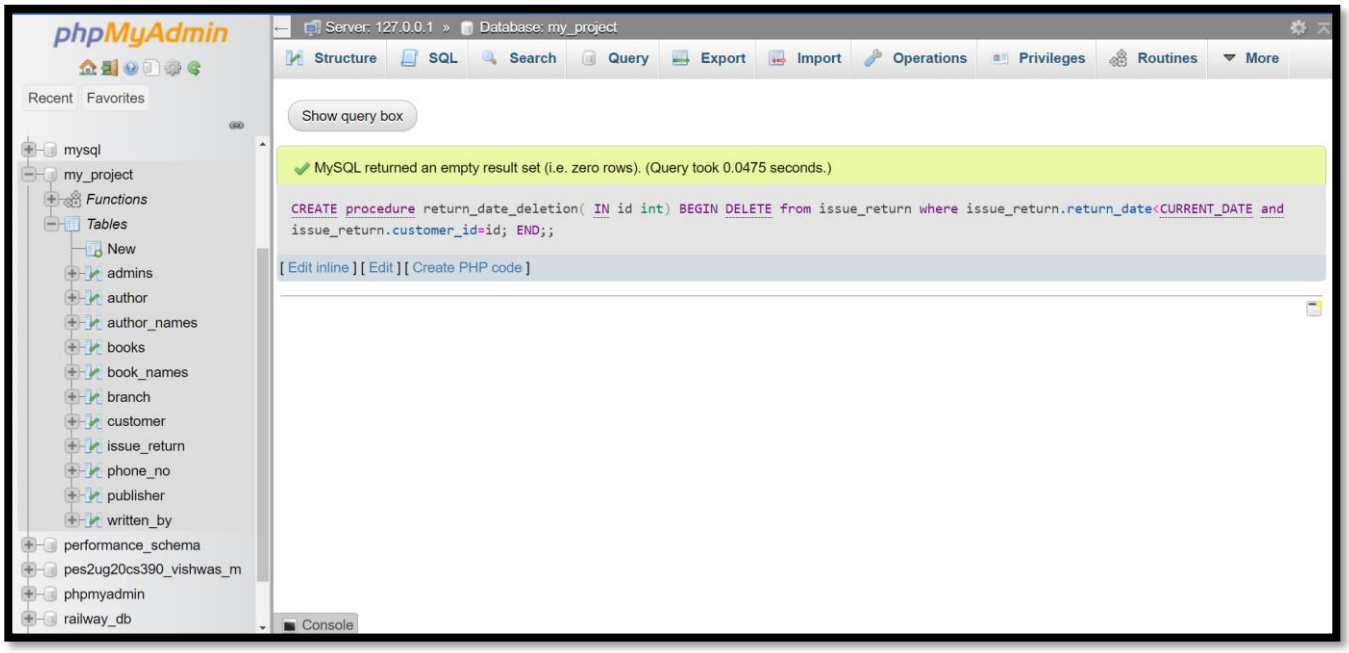
SQL QUERY: select admins.manager_id,tickets_check_limit(count(*)) as count from admins
GROUP by admins.manager_id

Output:

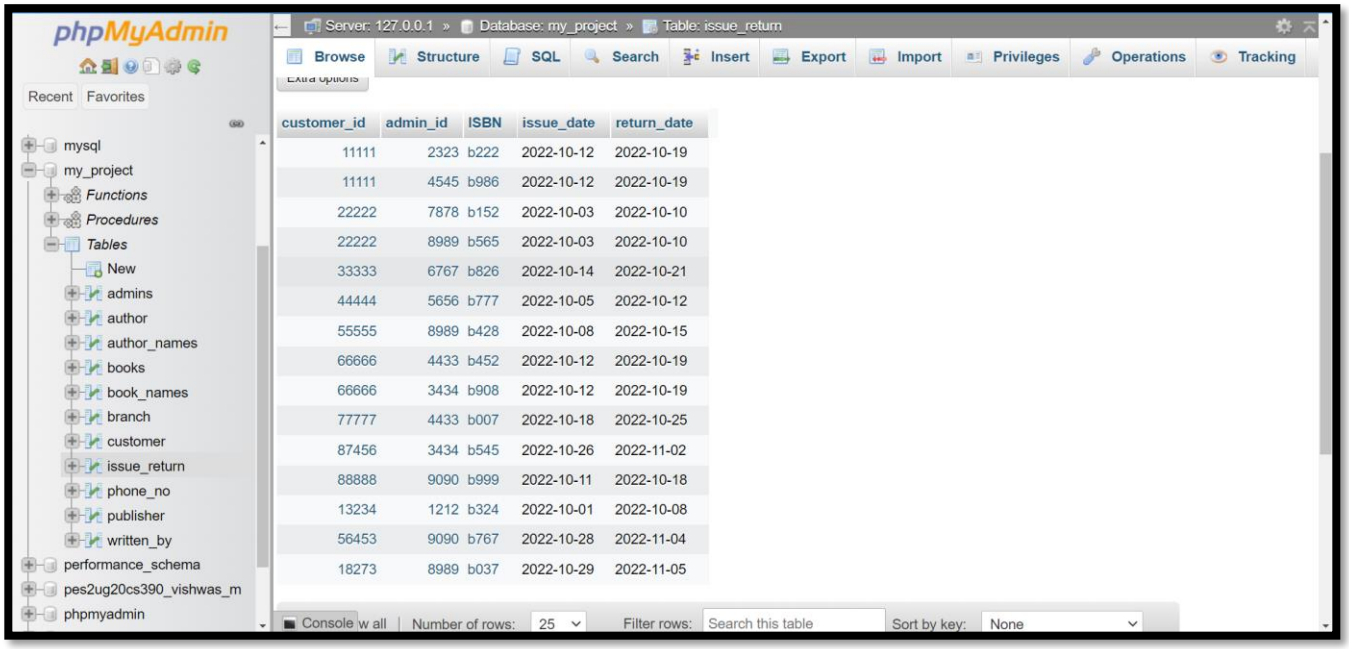


2) Delete the tuple whose due date has come to an end from the table using procedures.

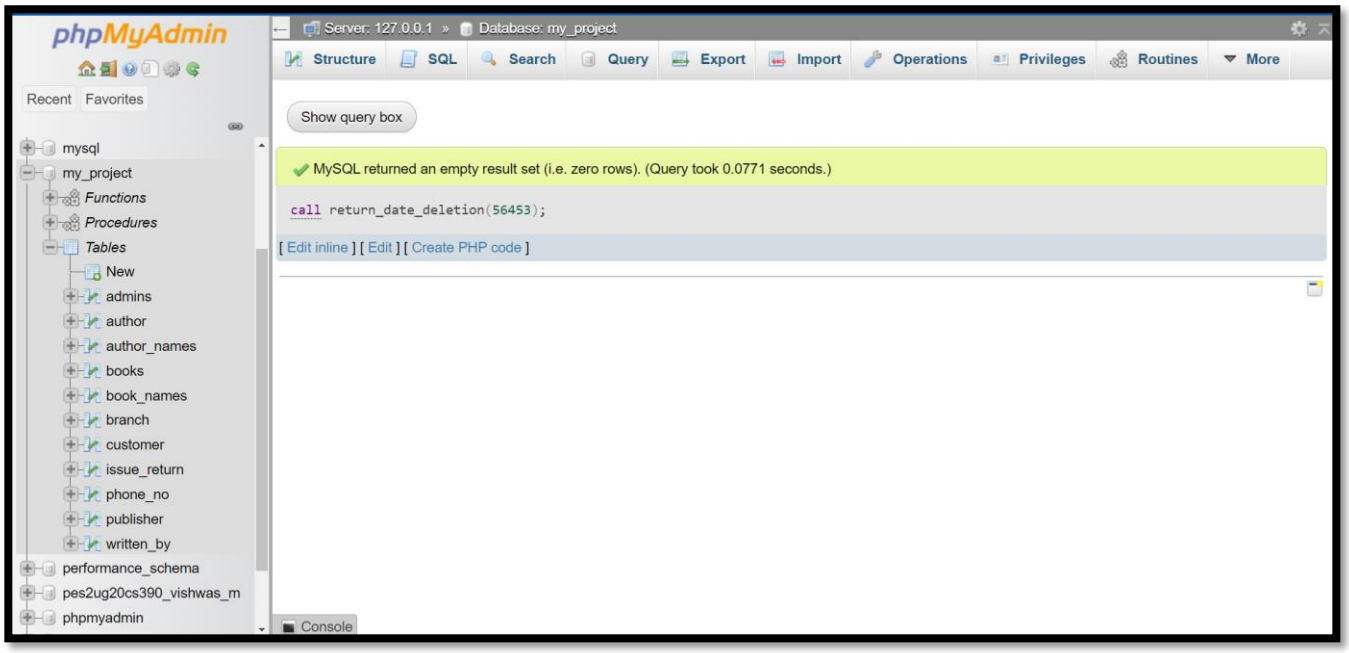
Soln:



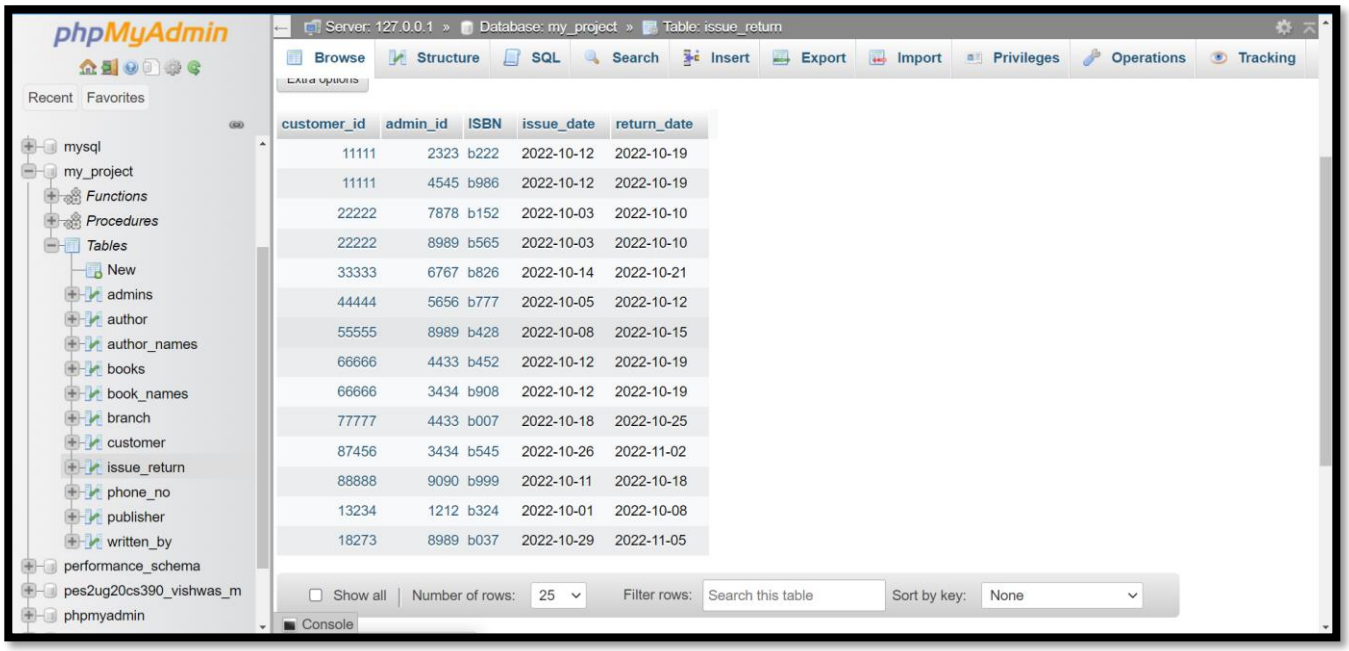
Initial table:



SQL Query: call return_date_deletion(56453);



After deletion:

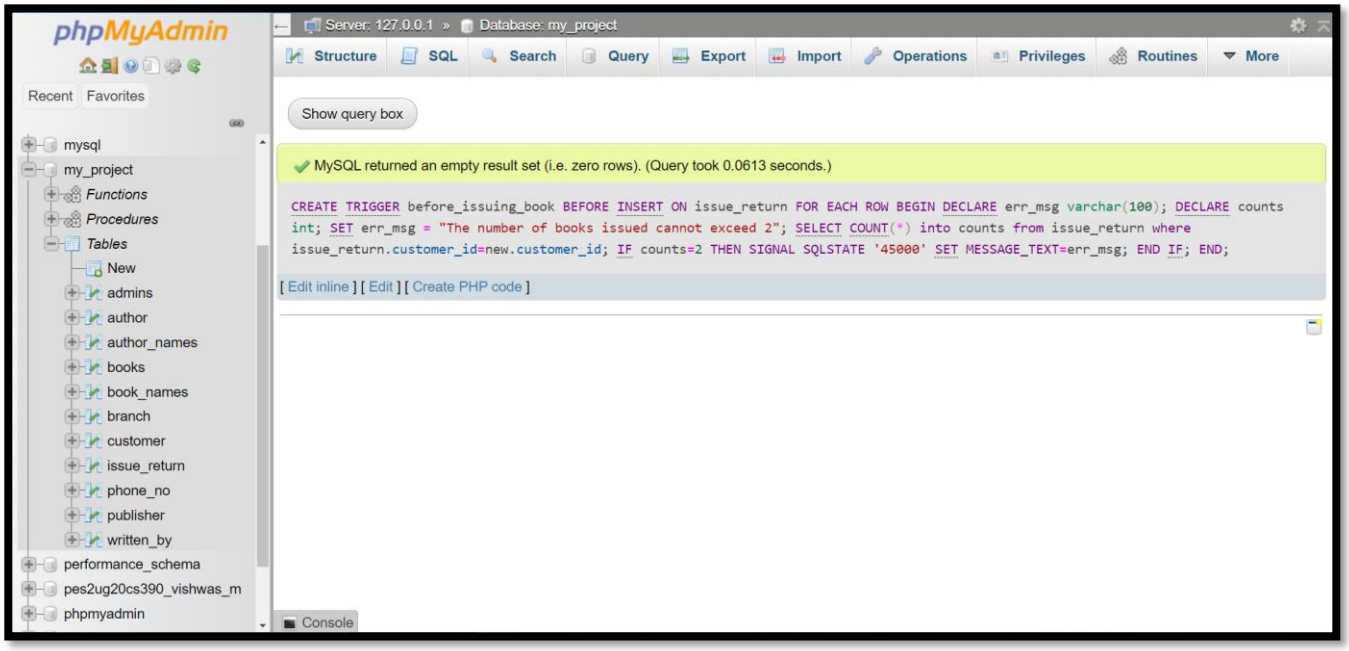


10. Triggers and Cursors

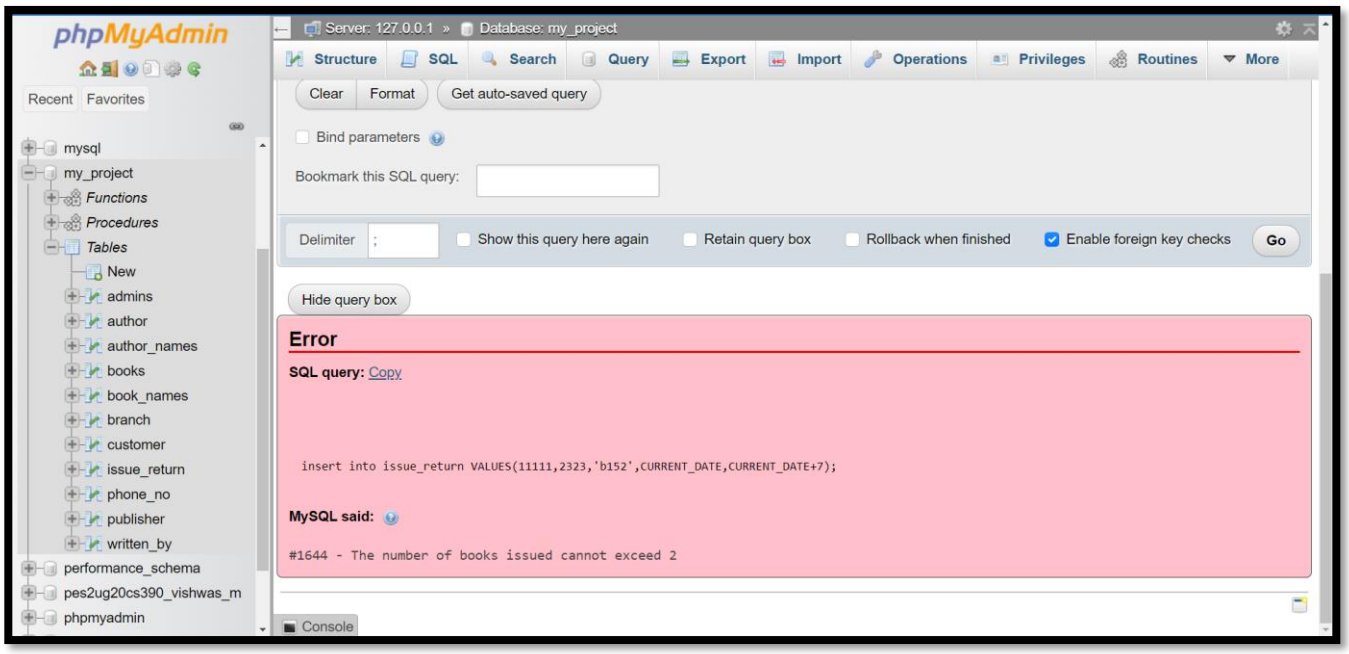
Create a Trigger and a Cursor. State the objective. Run and display the results.

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 :

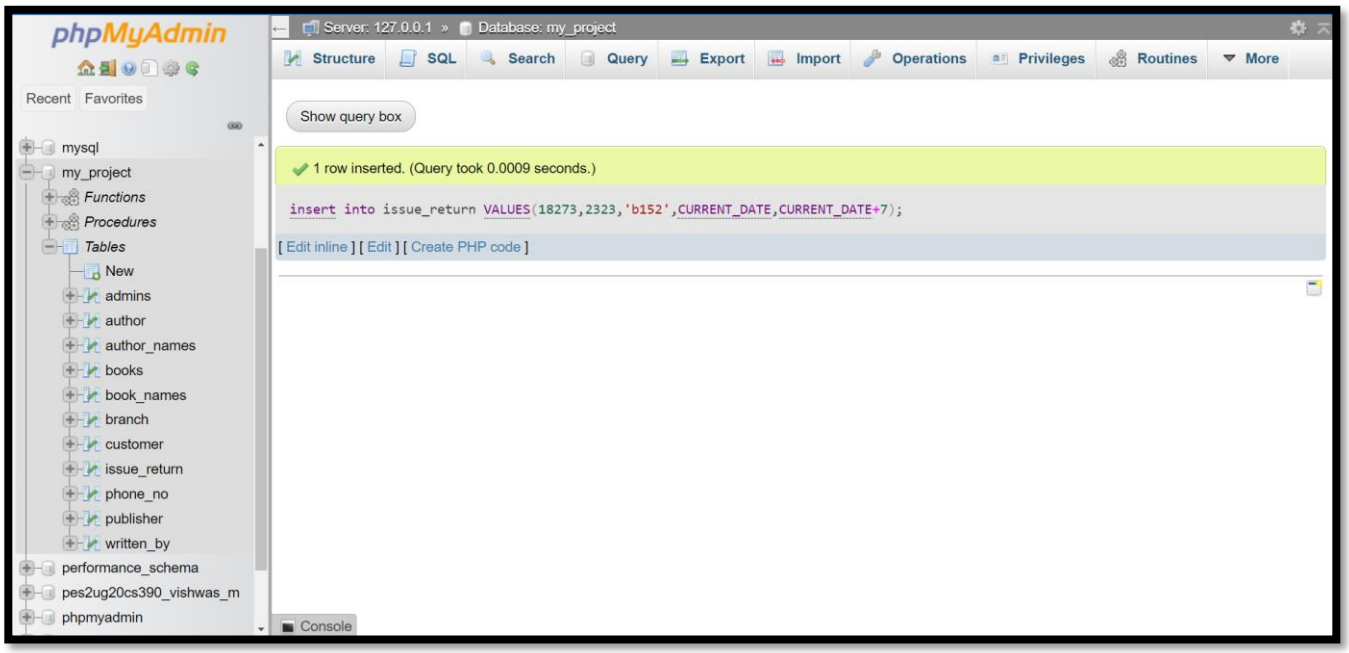


SQL QUERY: insert into issue_return VALUES (11111, 2323, 'b152', CURRENT_DATE,
CURRENT_DATE + 7)



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

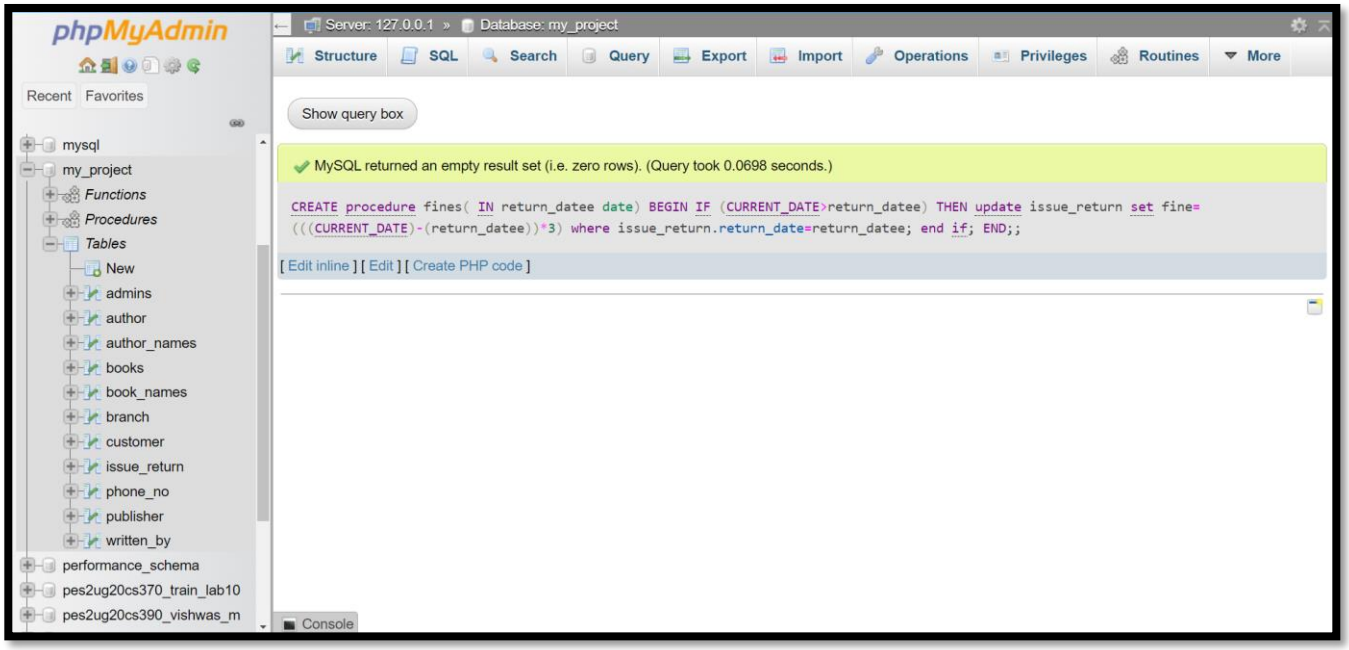
SQL QUERY: insert into issue_return VALUES (18273, 2323, 'b152', CURRENT_DATE,
CURRENT_DATE+7)



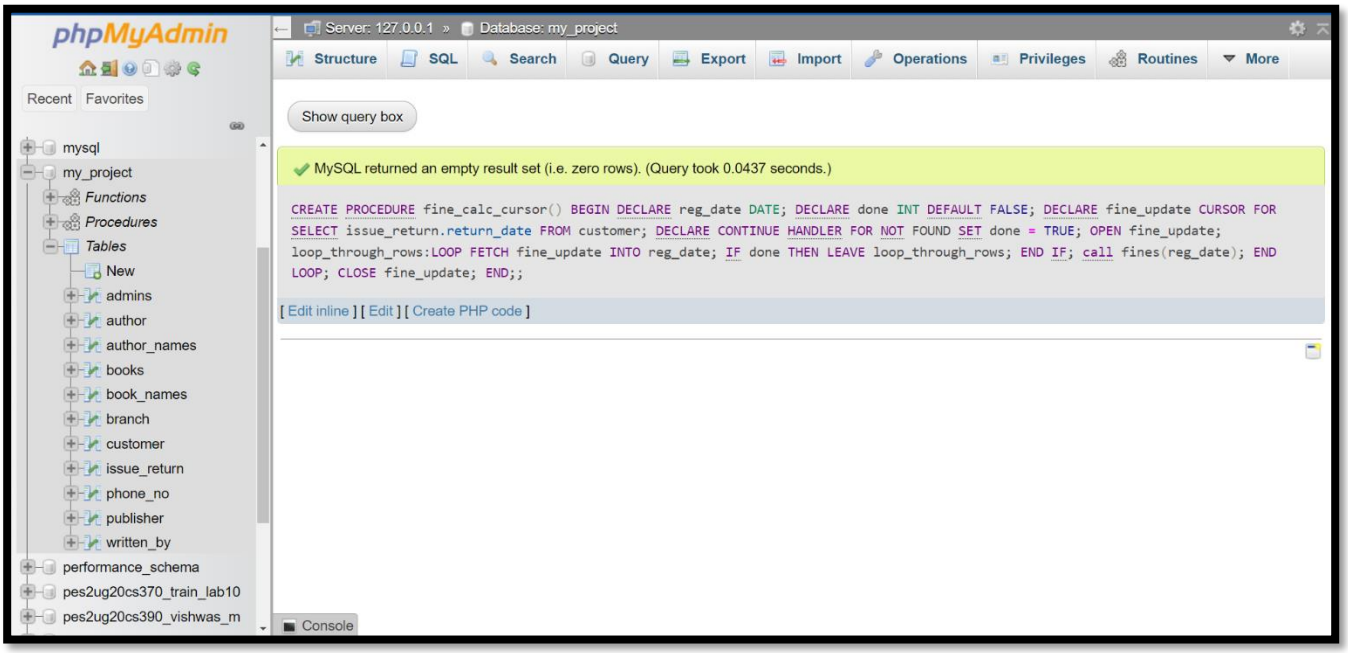
Error does not occur because customer having customer_id 18273 issued only one book and can still issue one more book.

2) Cursors :

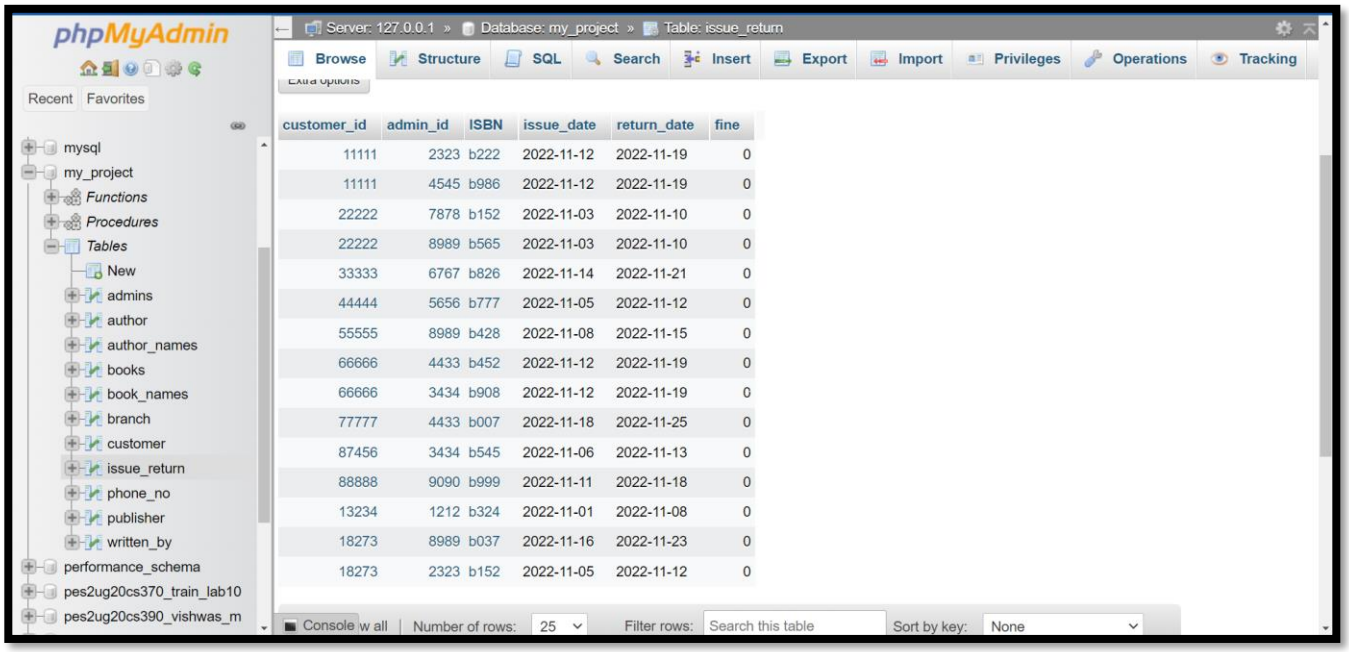
Procedure to update the Fine column in issue_return table



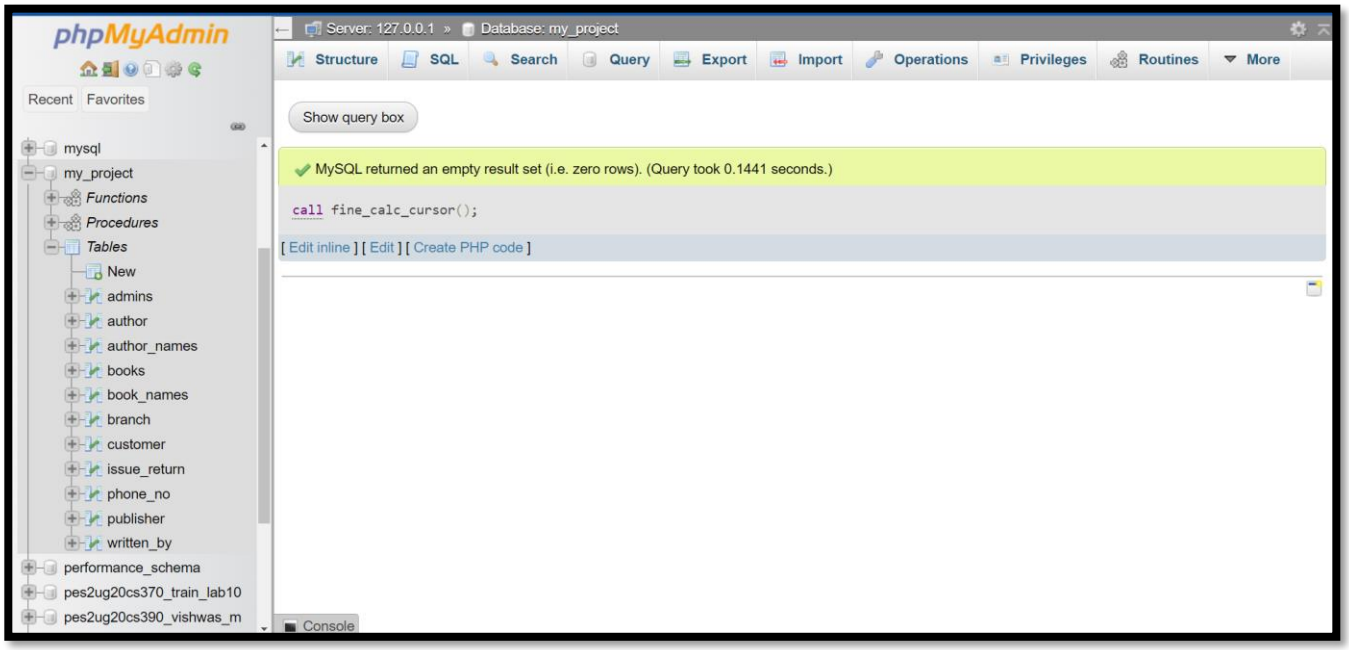
Cursor to call the above procedure:



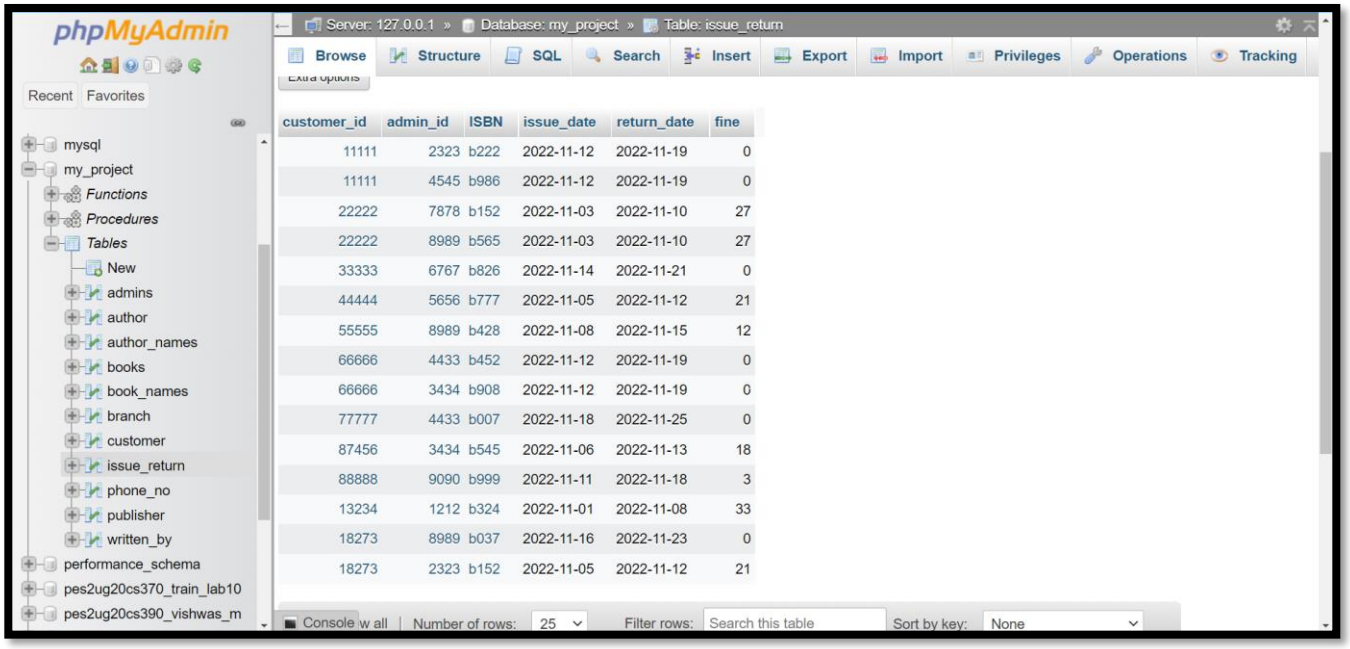
Before Update:



SQL command for calling the cursor: call fine_calc_cursor();



After Calling the Cursor:

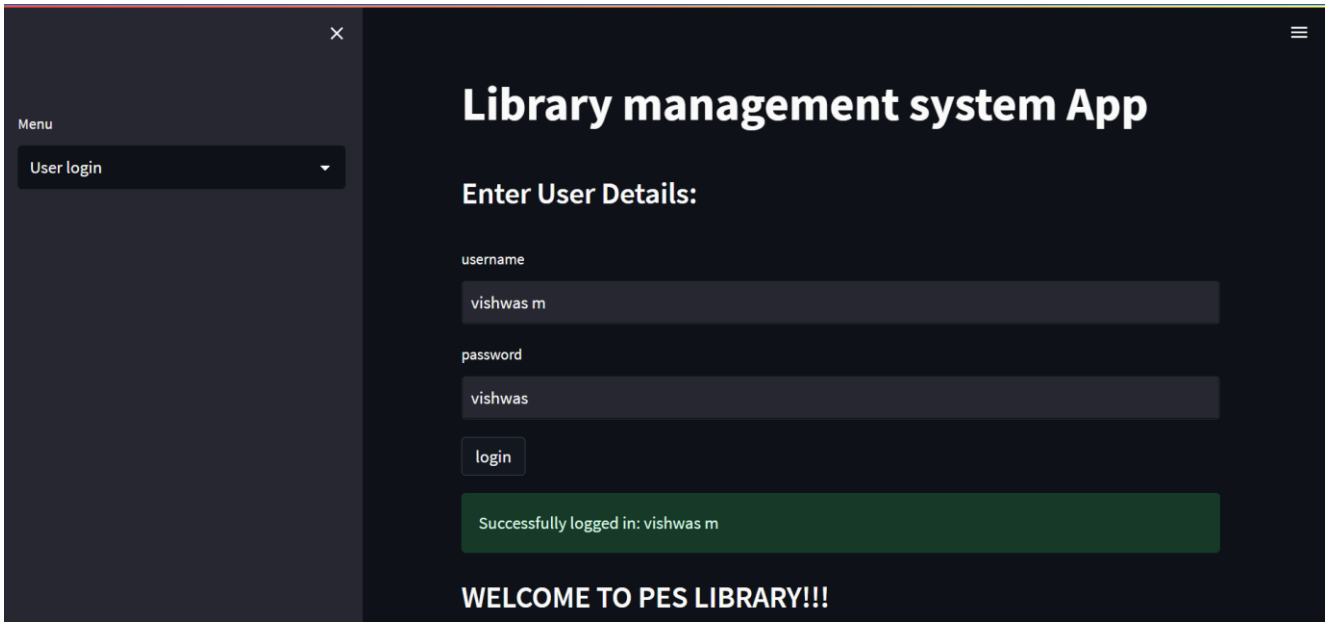


11. Developing a Frontend

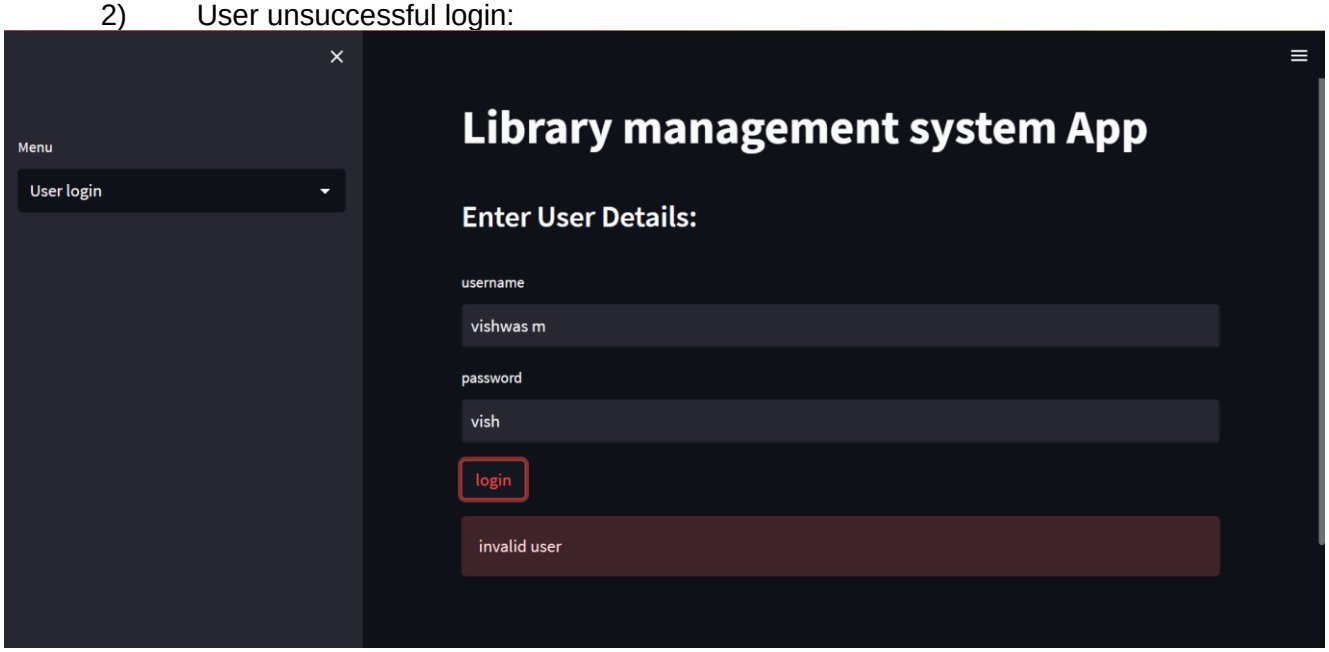
The frontend should support

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

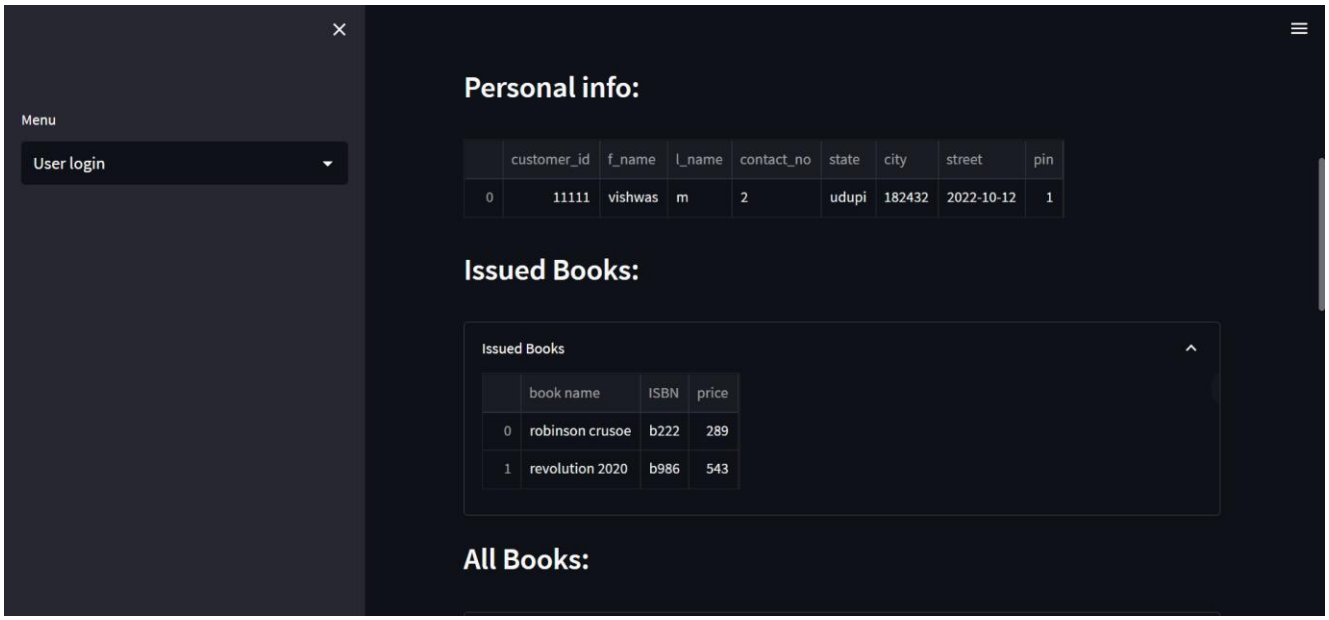
1) User successful login:



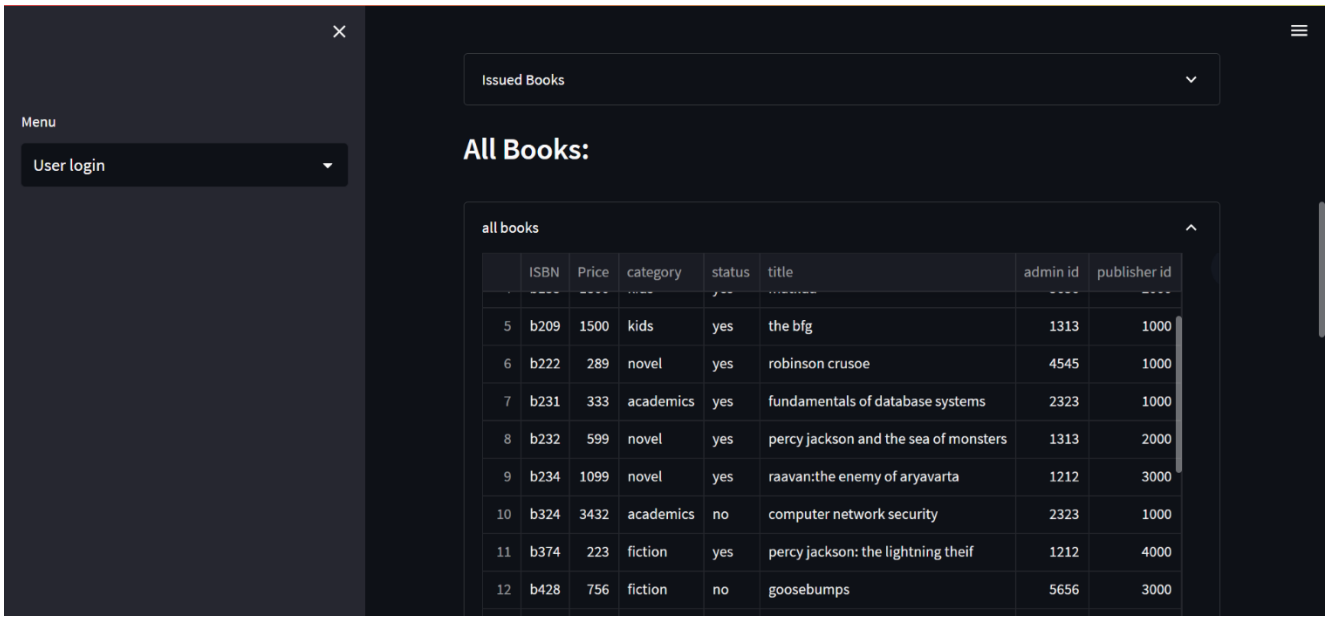
2) User unsuccessful login:



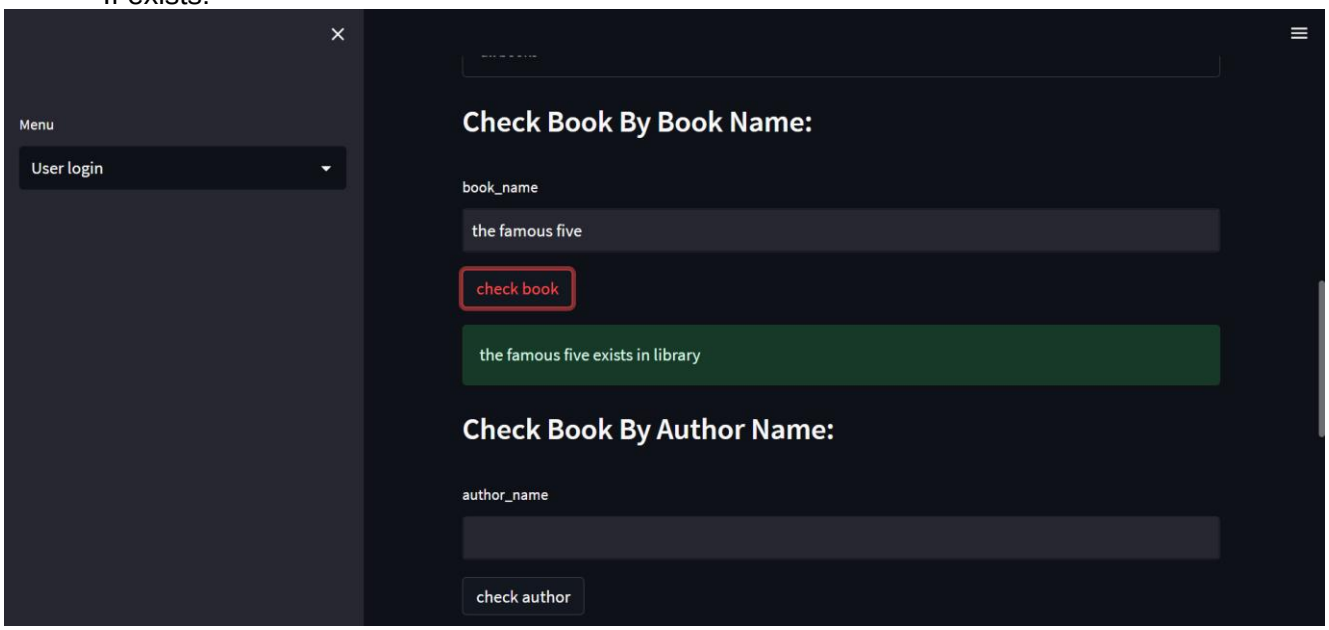
3) Displays personal info of the user and the books he issued from the library



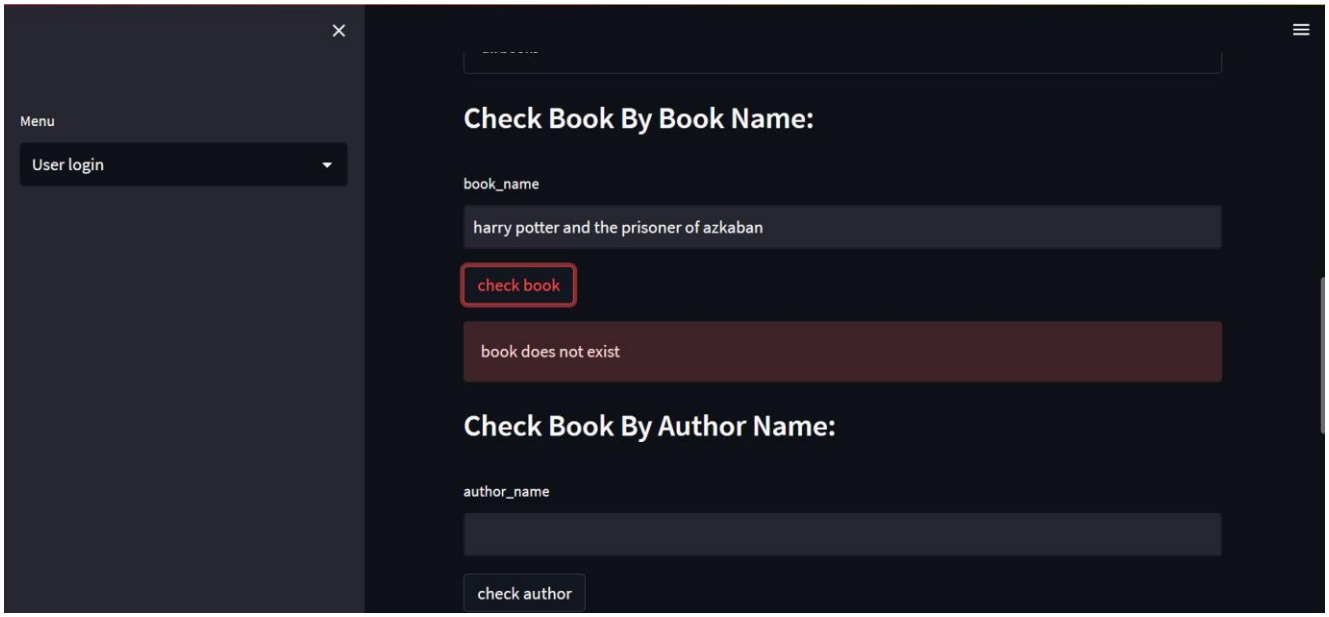
4) User can view all the books in the library



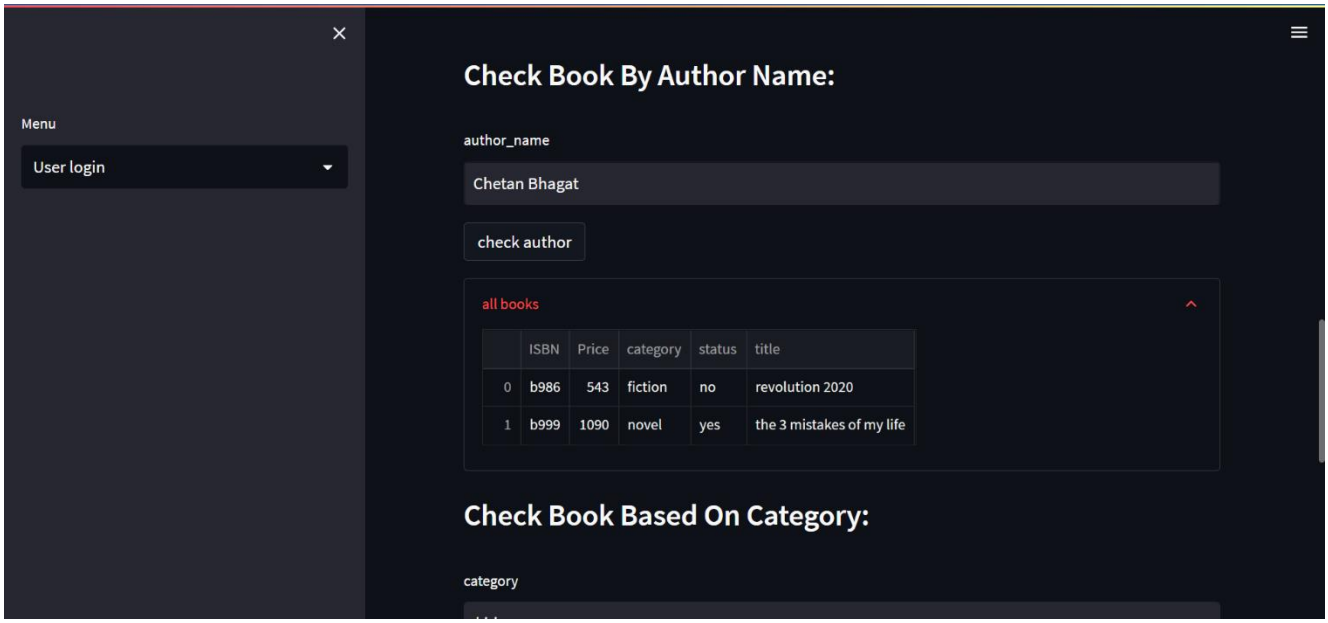
5) Check if book exists in library or not by searching the book name
If exists:



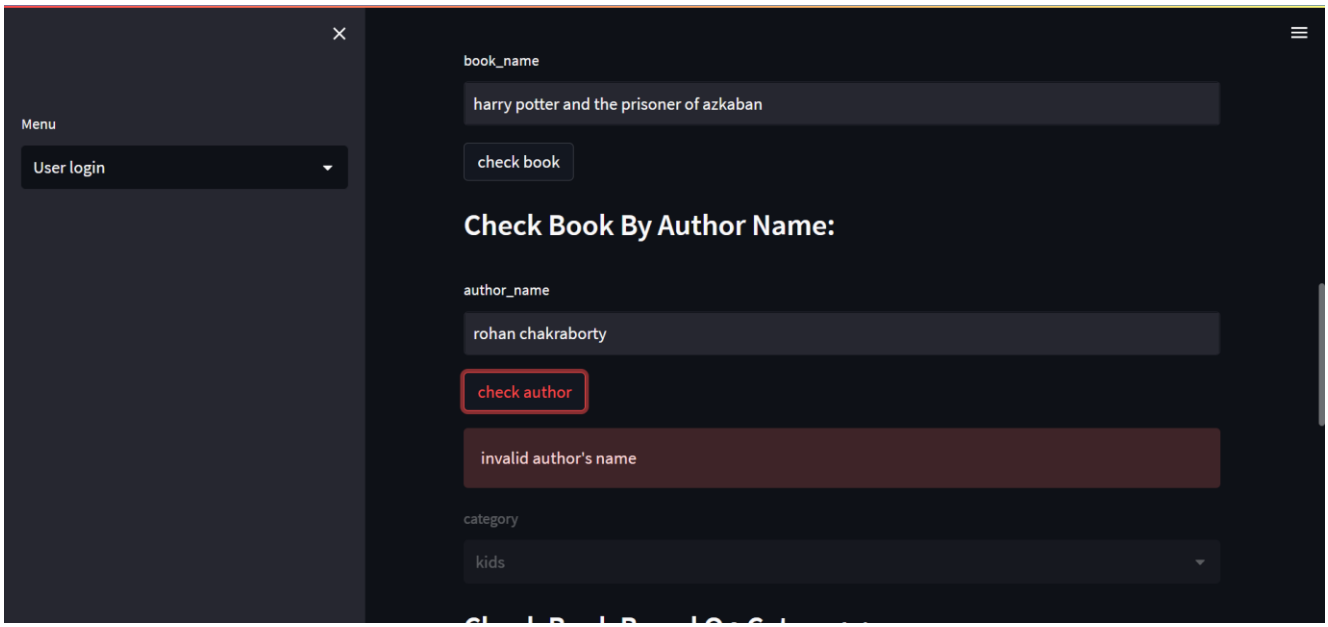
If not exists:



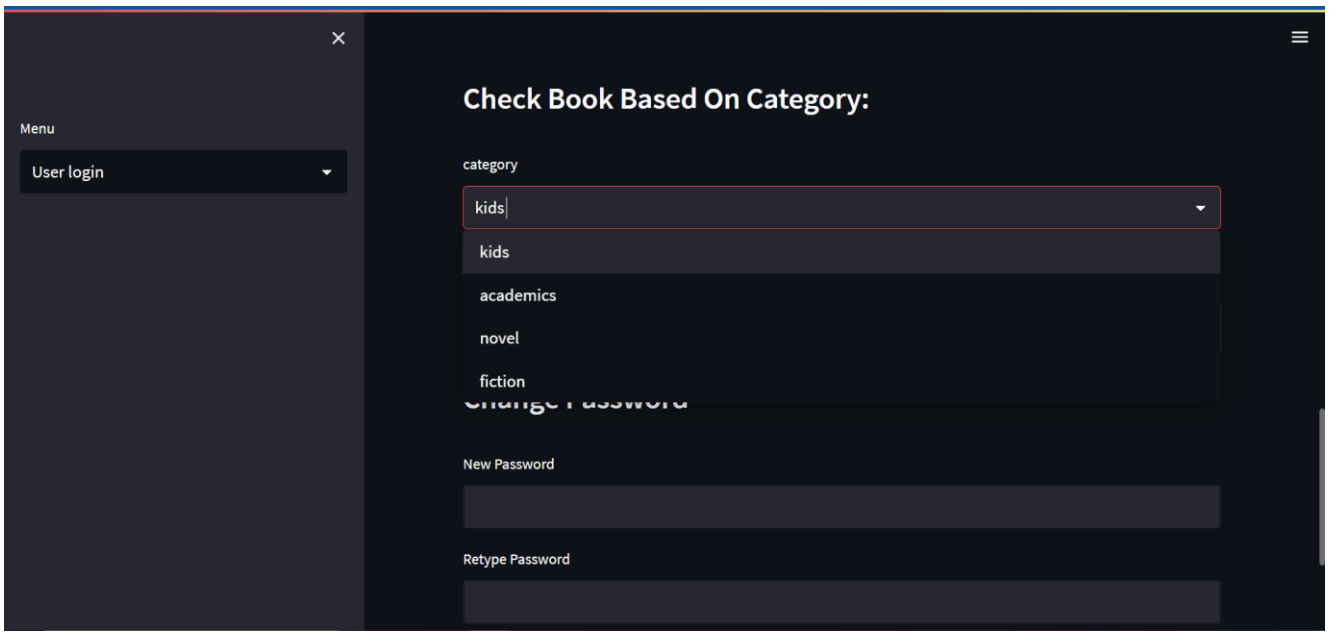
6) Check if book exists in library or not by searching the author's name
If exists:



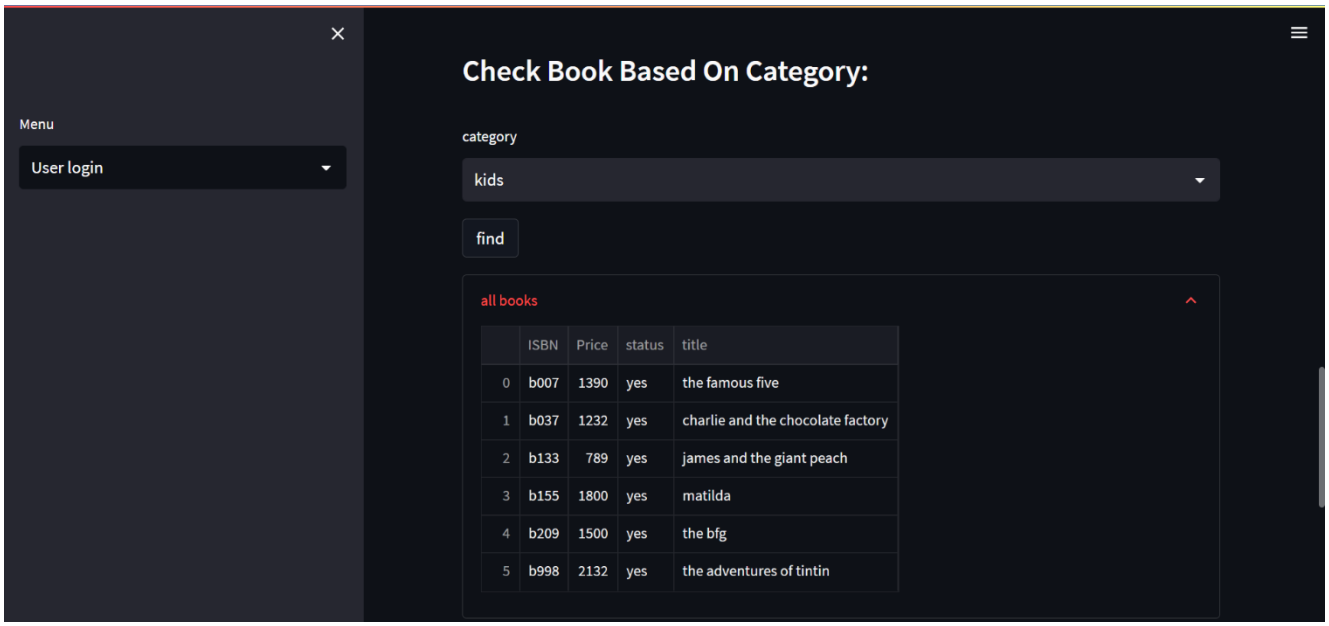
If author's books are not available in the library:



7) Check books by category:
Categories:



For kids:

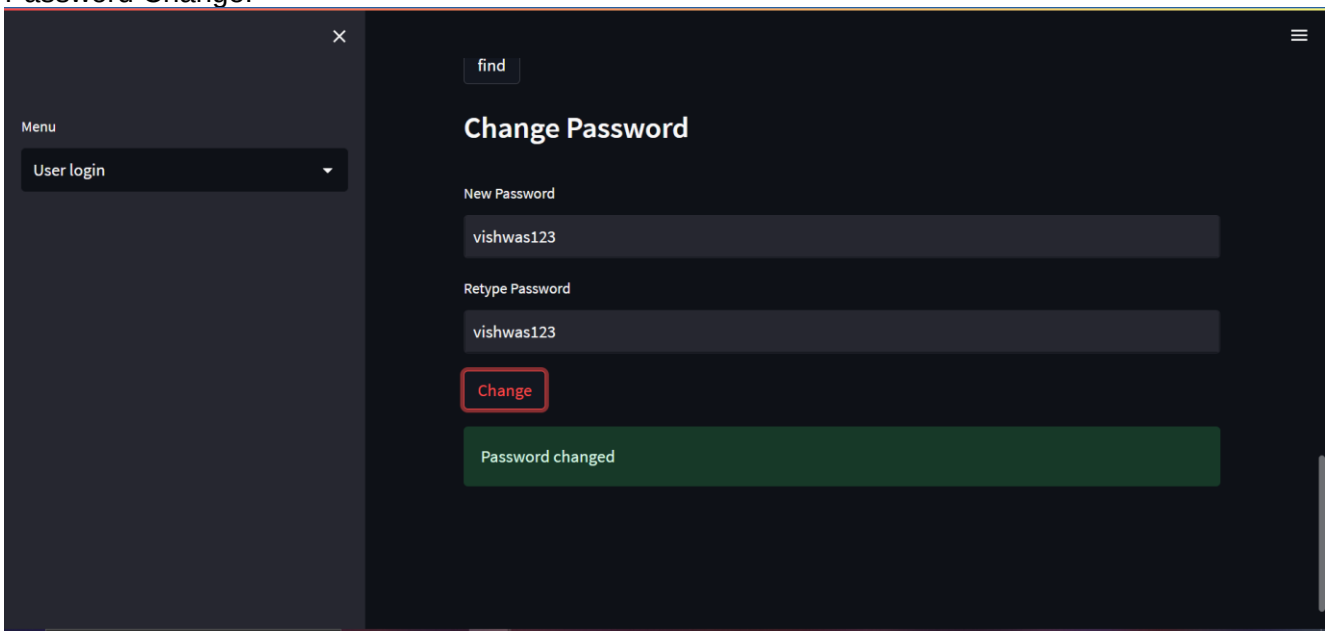


8) Change user password:

Initial password:

	customer_id	f_name	l_name	street	city	pin	reg_date	manager_id	password
Edit Copy Delete	11111	vishwas	m	2	udupi	182432	2022-10-12	1	vishwas

Password Change:

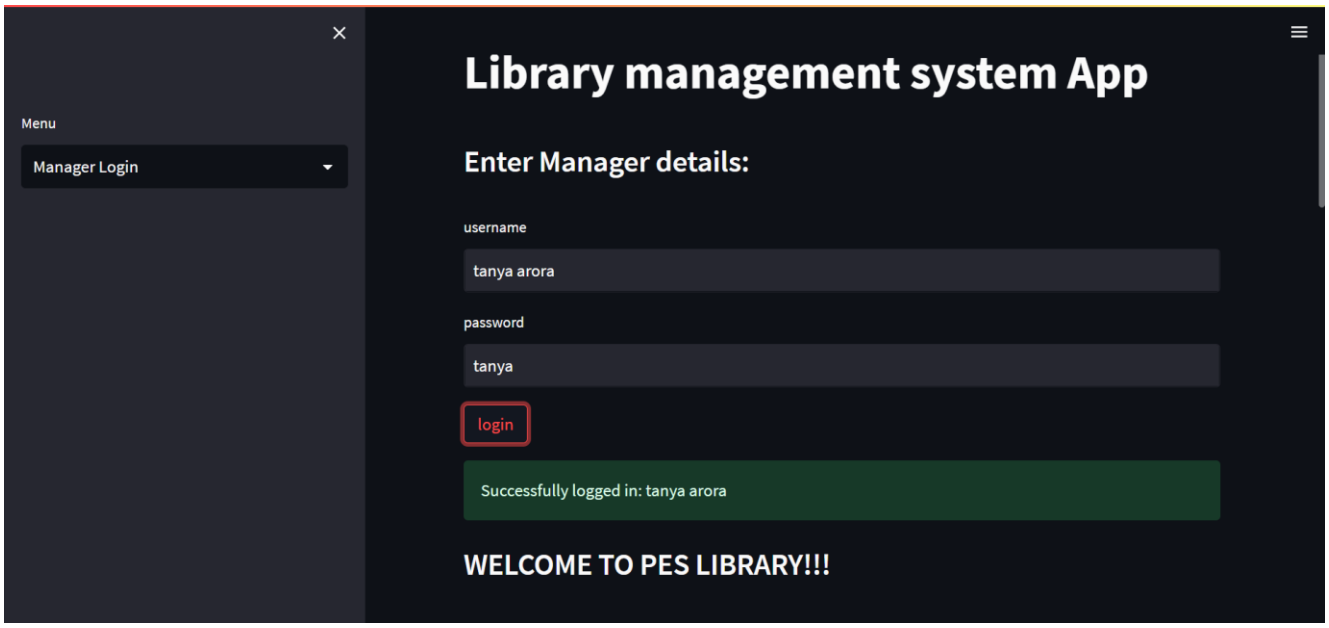


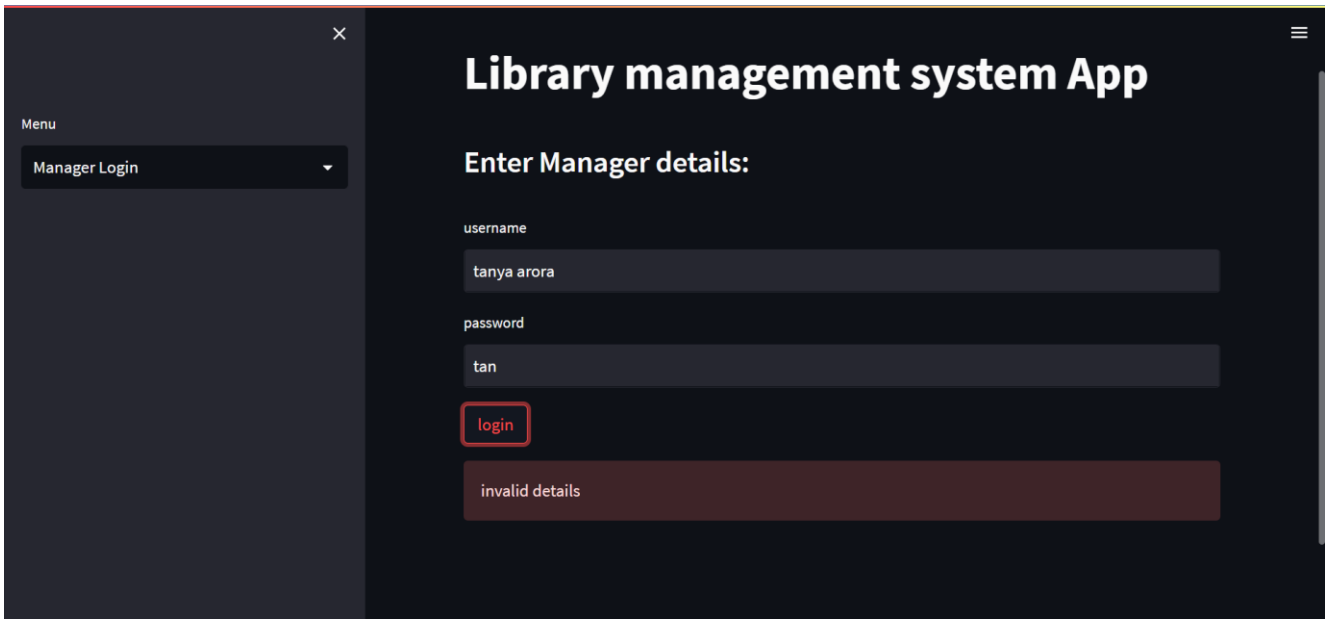
Updated Database:

	customer_id	f_name	l_name	street	city	pin	reg_date	manager_id	password
Edit Copy Delete	11111	vishwas	m	2	udupi	182432	2022-10-12	1	vishwas123

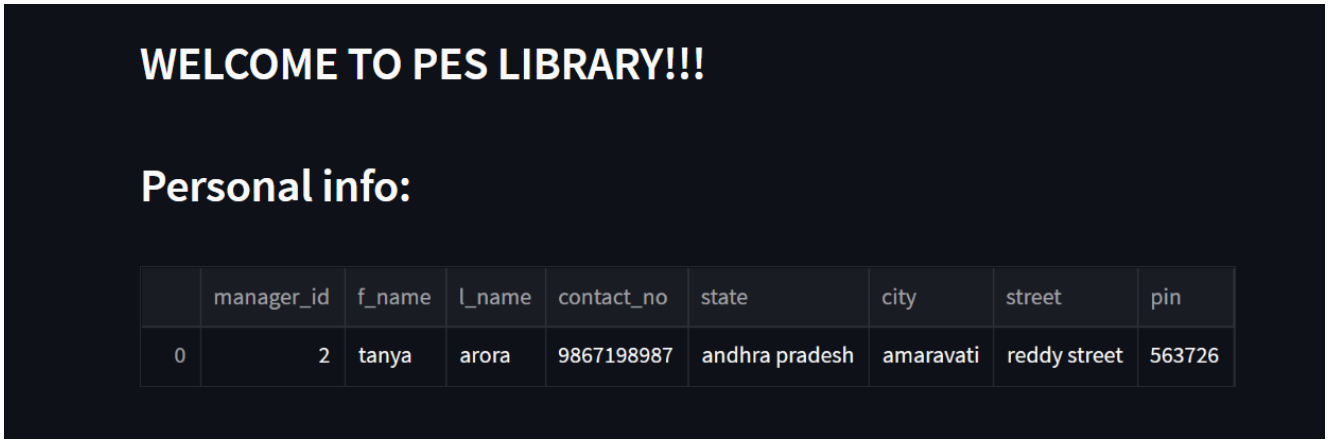
Branch Manager Page:

1) Manager login:

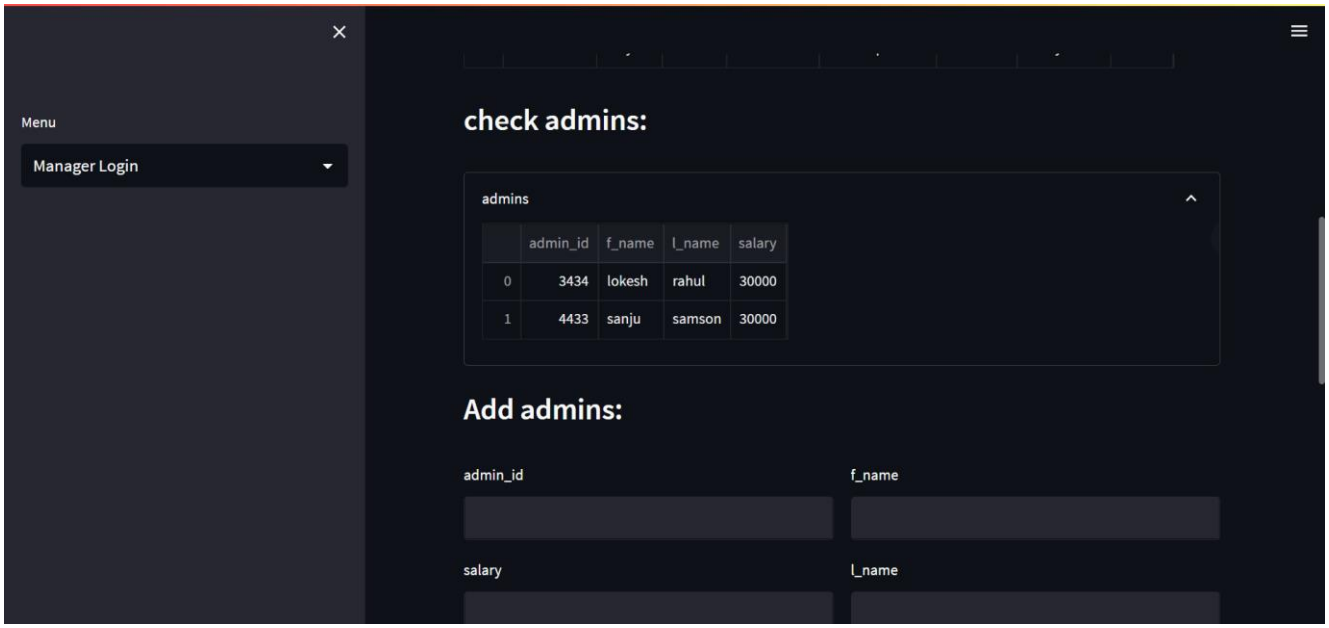




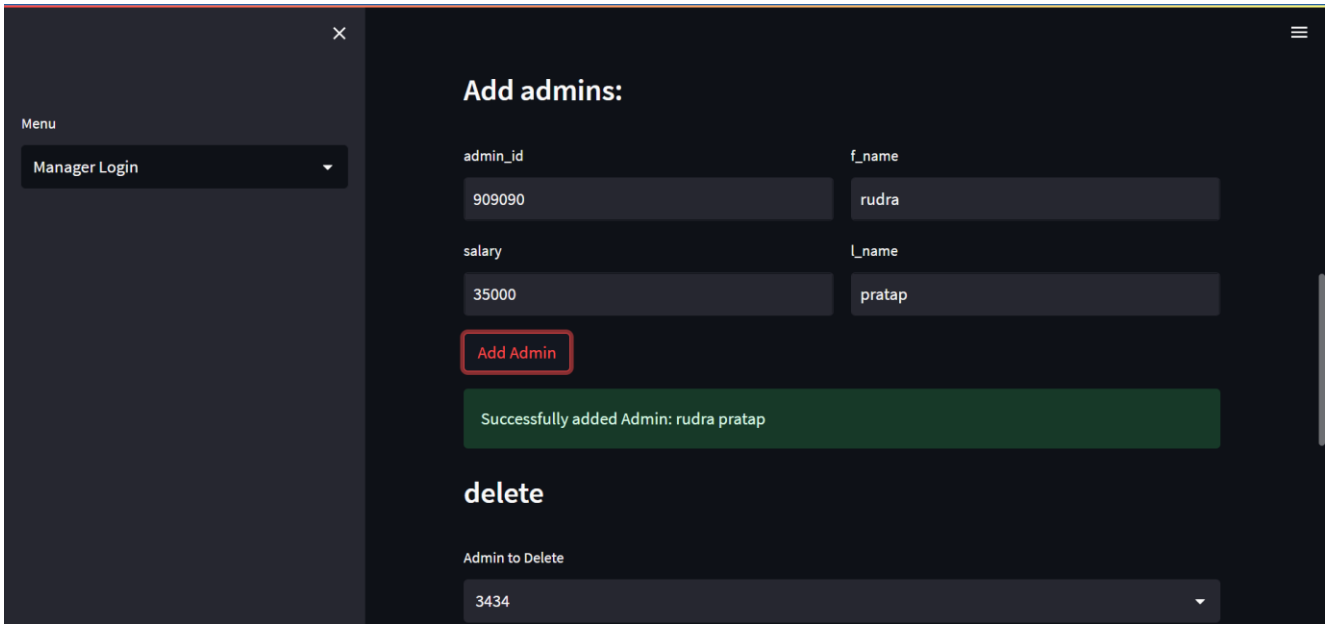
2) Displays personal info of manager:



3) Check all the admins under that particular manager:

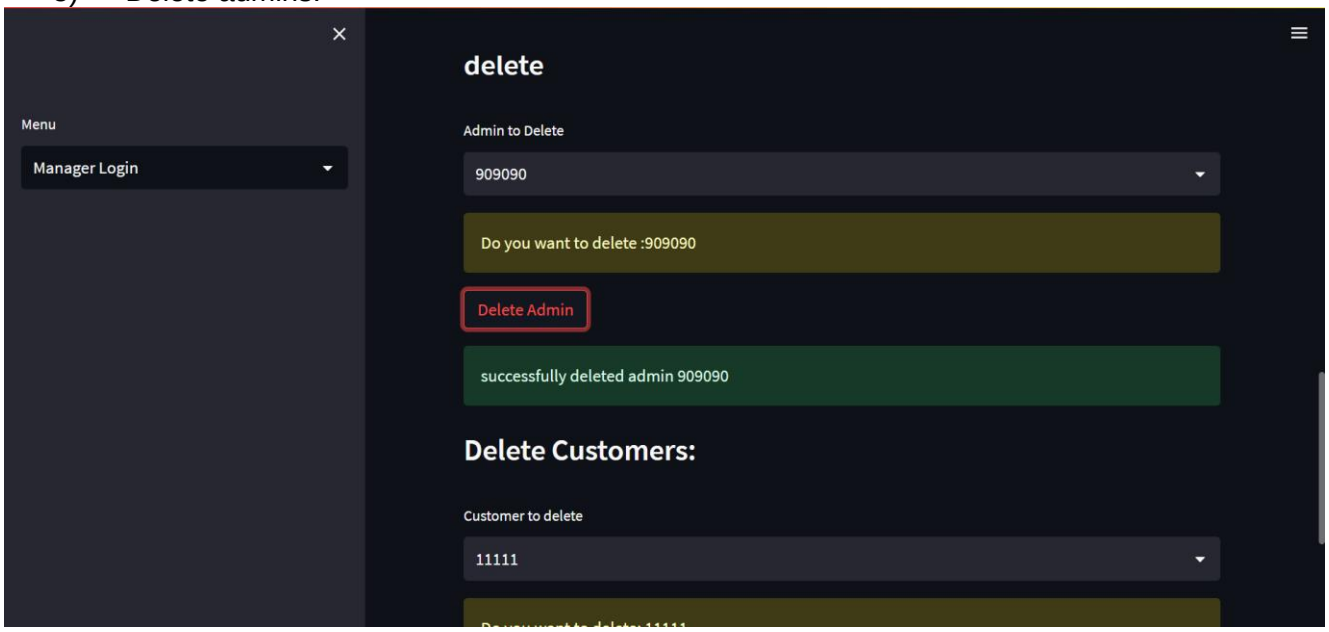


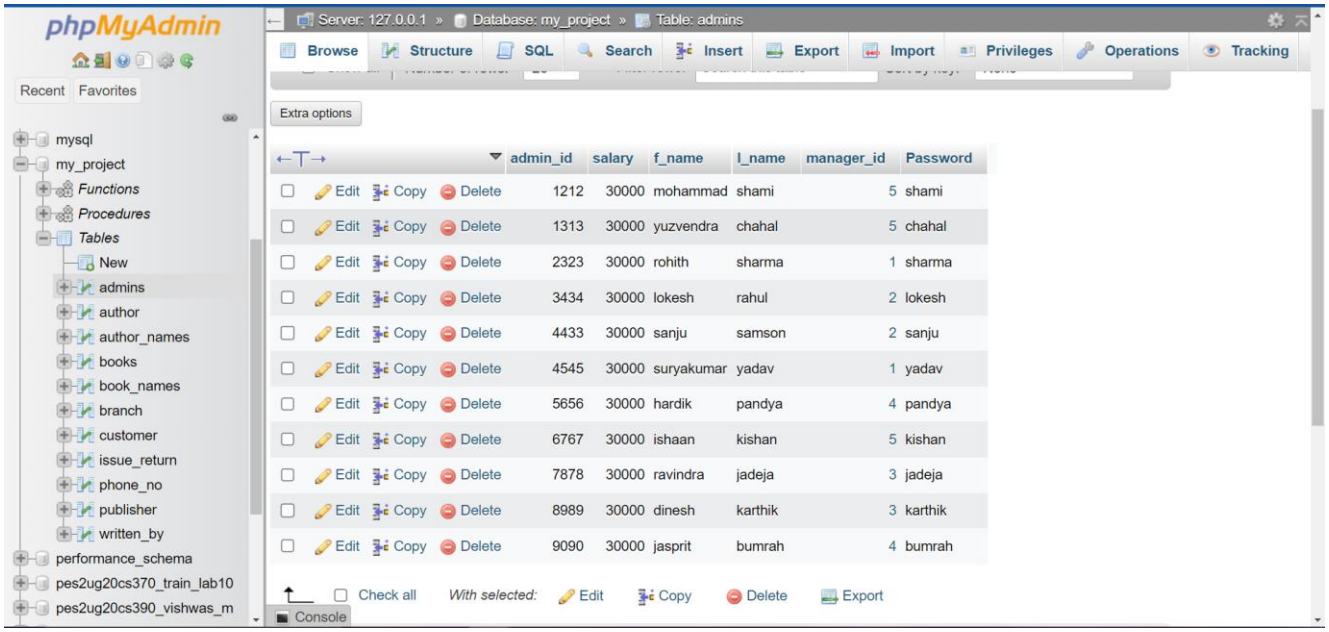
4) Add admins:



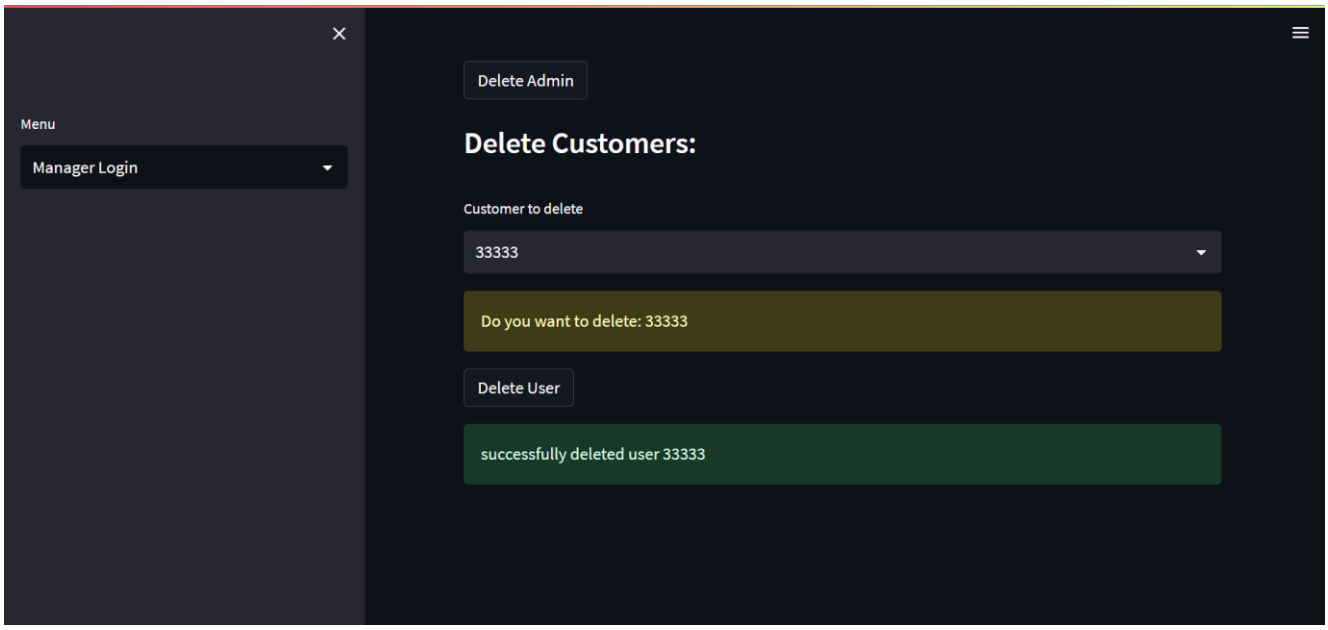
☐	Edit	Copy	Delete	9090	30000	jasprit	bumrah	4 bumrah
☐	Edit	Copy	Delete	909090	35000	rudra	pratap	2 rudra

5) Delete admins:



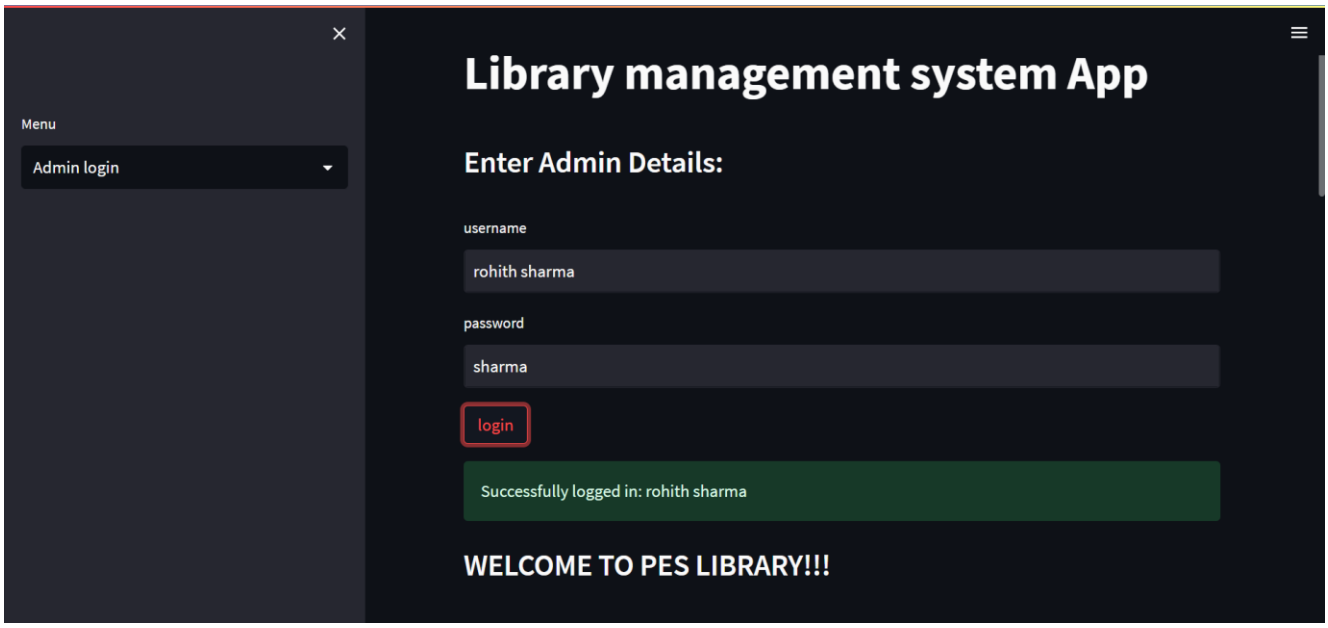


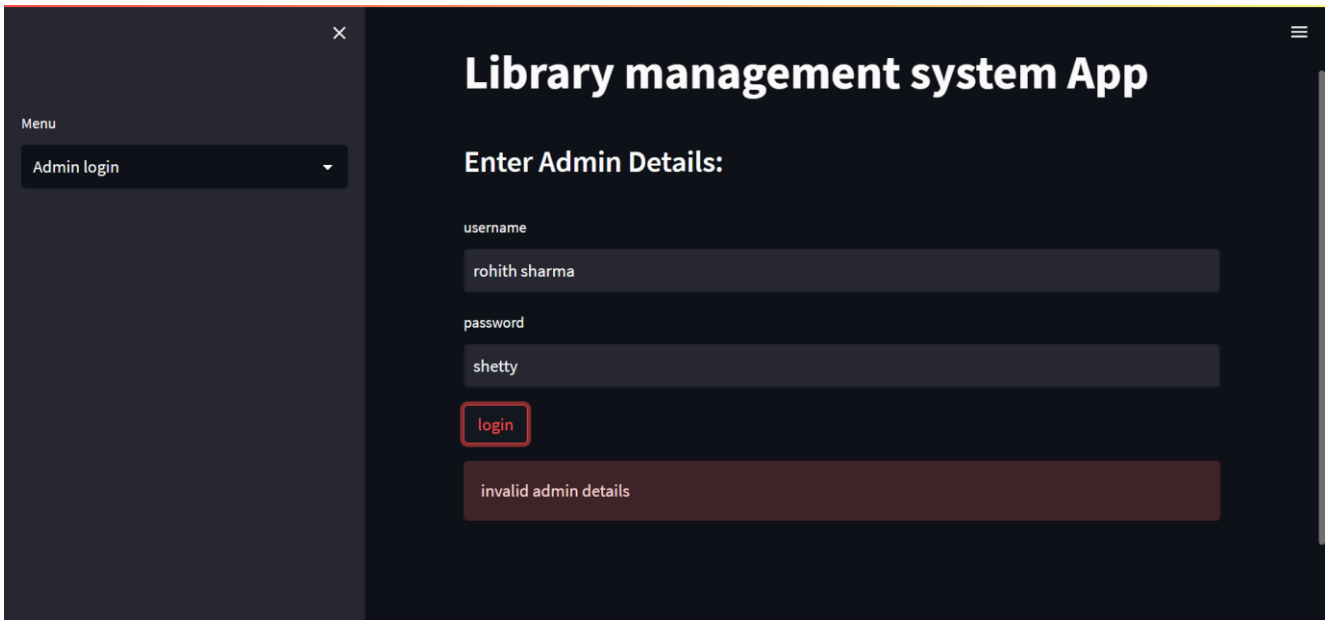
6) Delete customer:



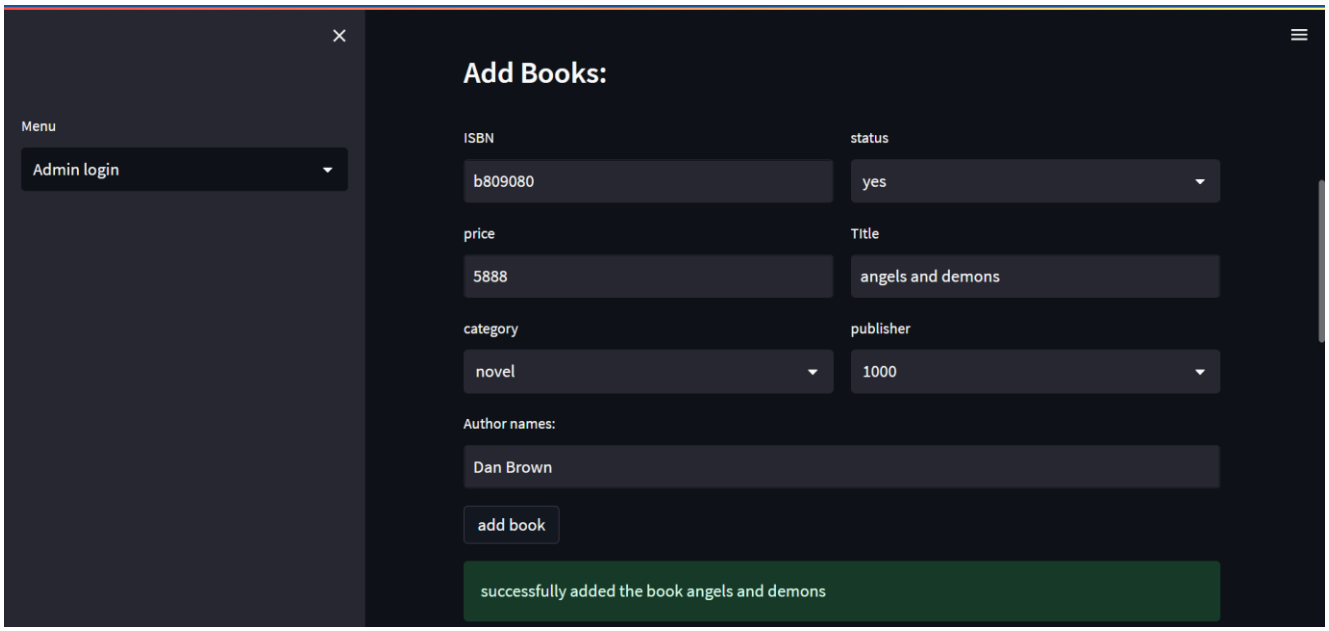
Admin Page:

1) Admin login:

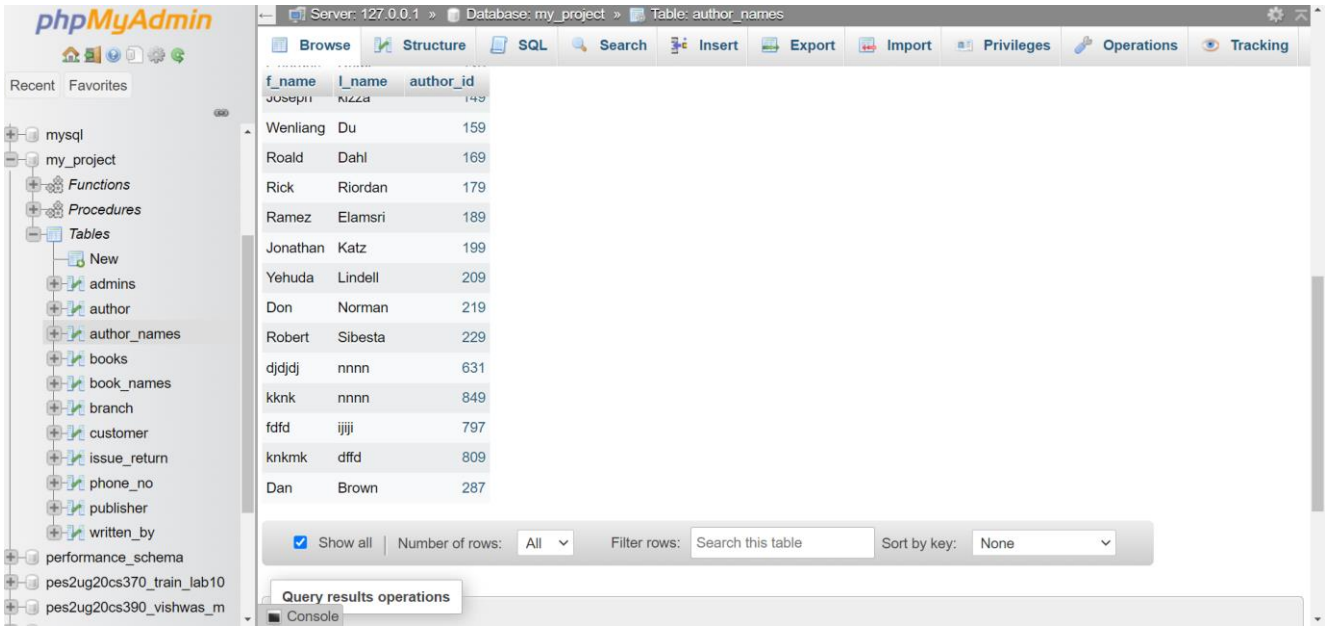


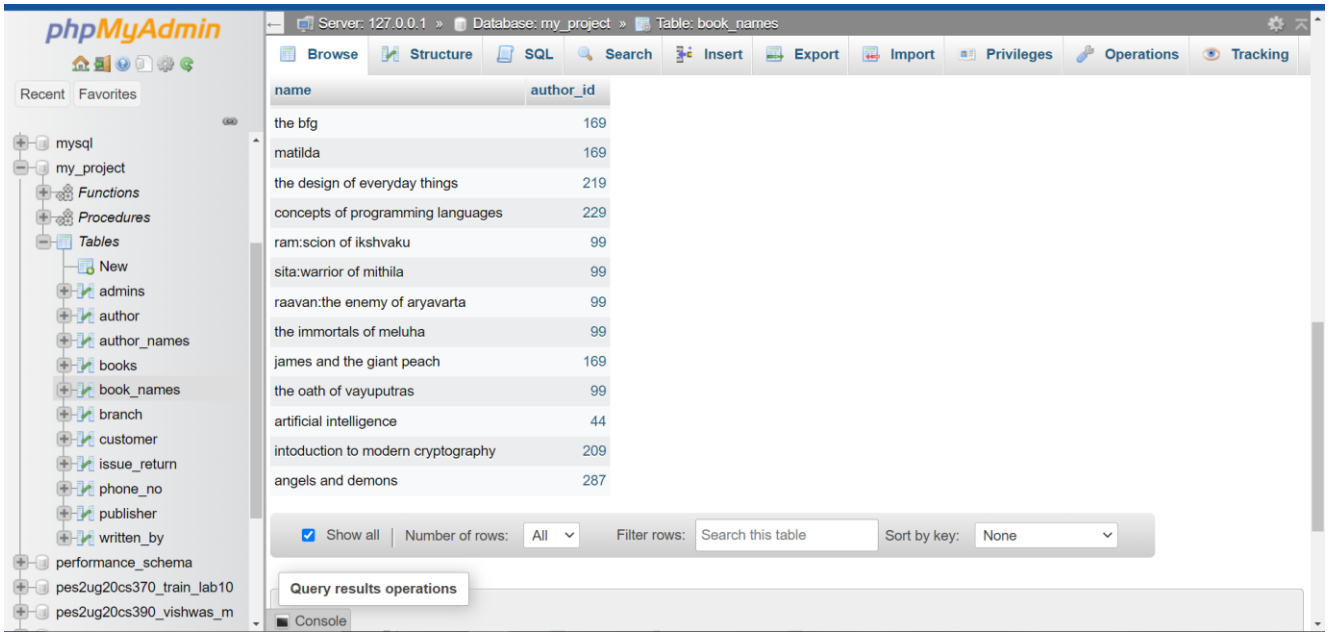


2) Adding Books to the Database:

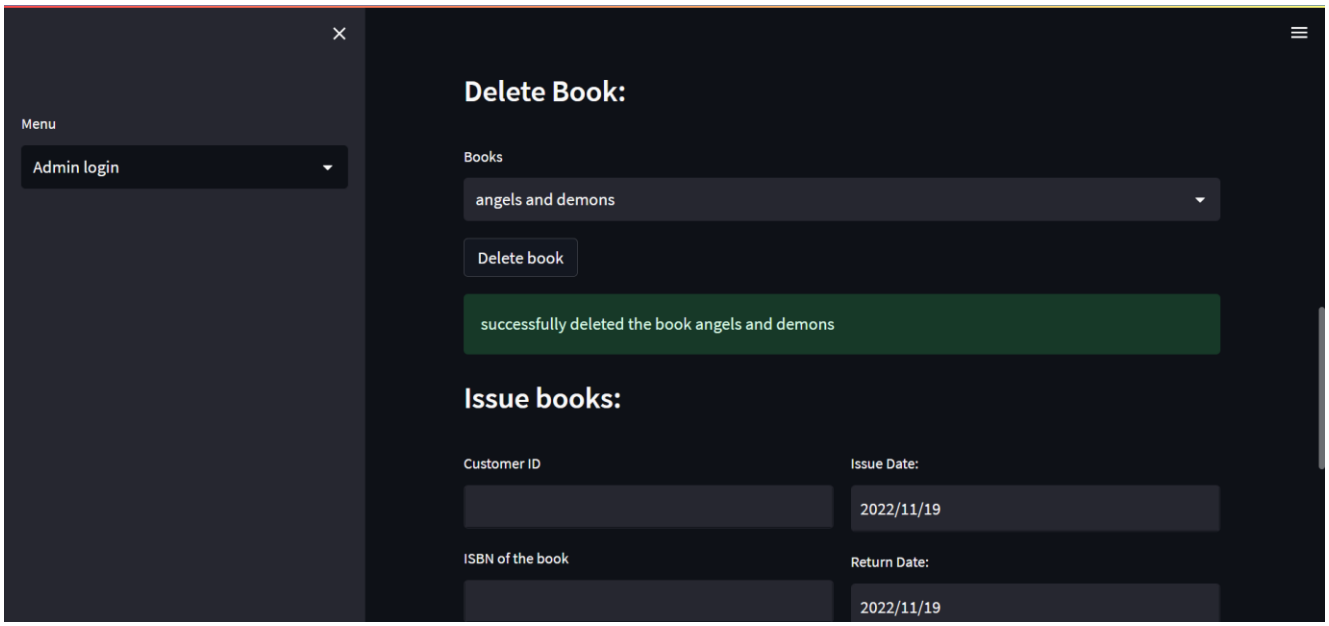


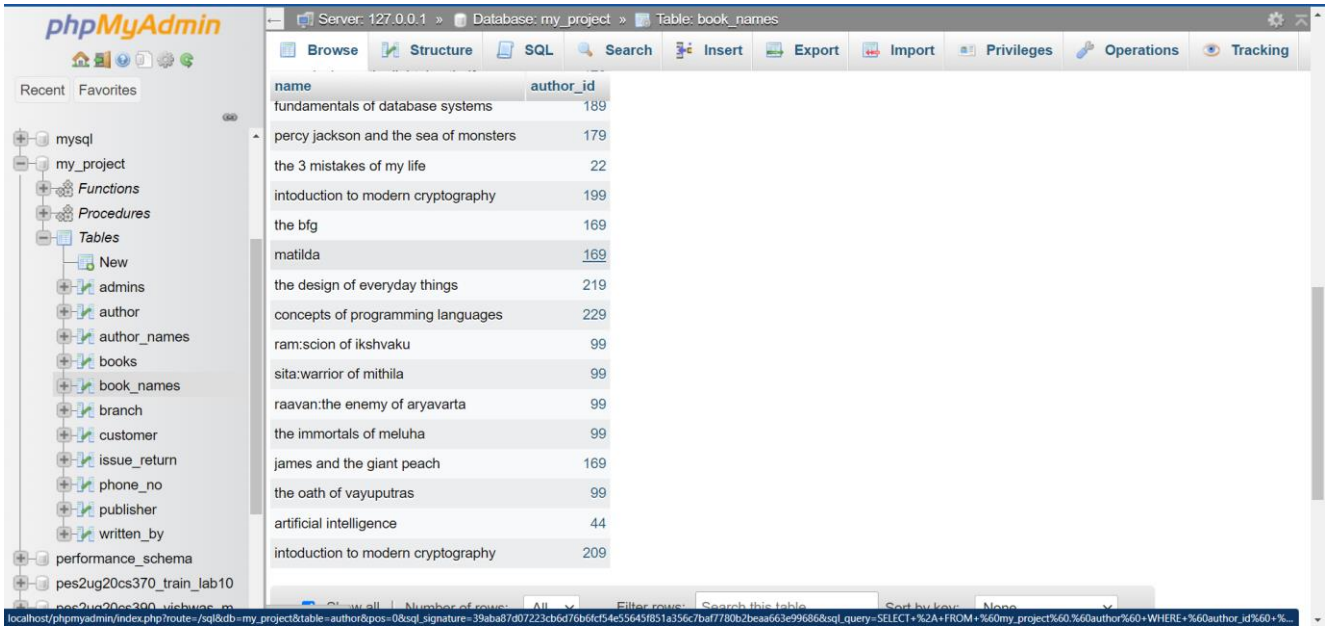
☐	Edit	Copy	Delete	b777	454	novel	yes	wuthering heights	6767	2000
☐	Edit	Copy	Delete	b809080	5888	novel	yes	angels and demons	2323	1000
☐	Edit	Copy	Delete	b826	222	fiction	no	harry potter and the chamber of secrets	4545	1000

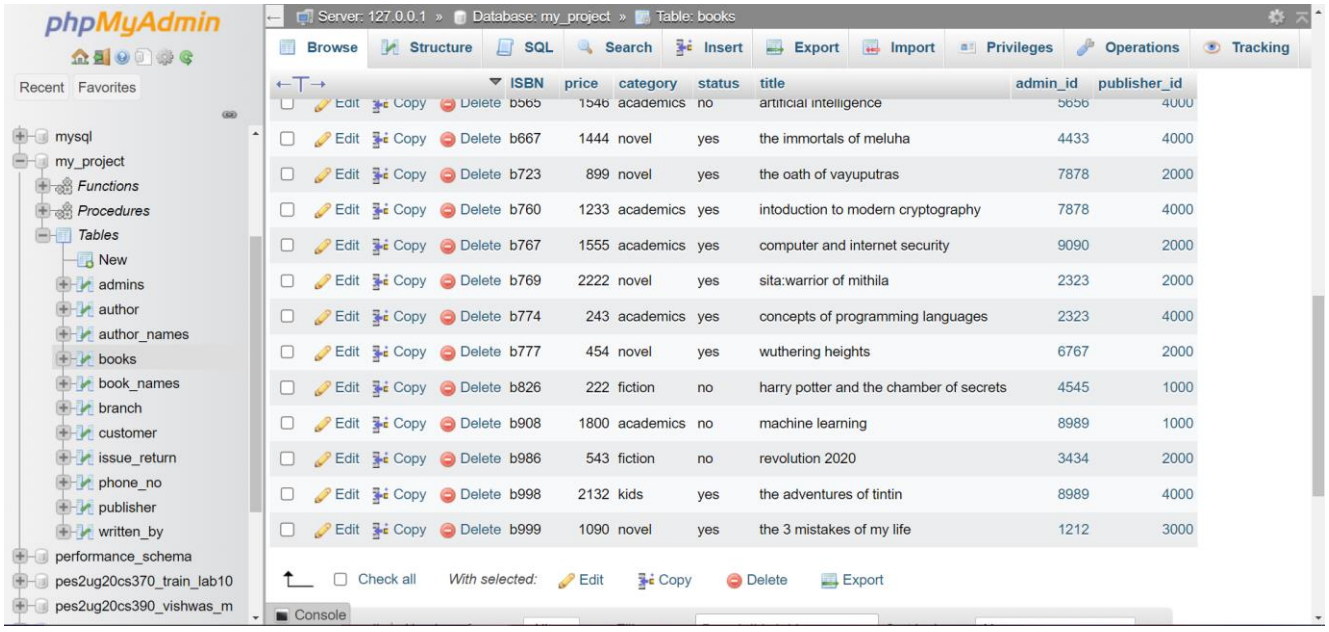




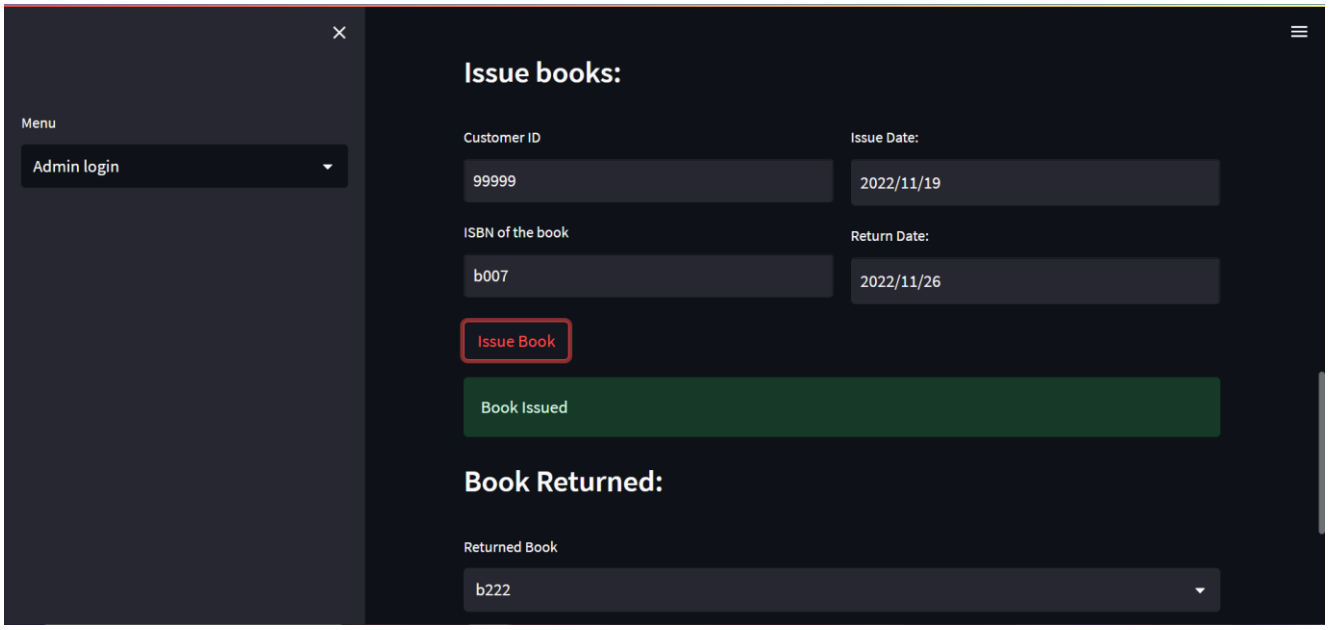
3) Delete a book from the library database:

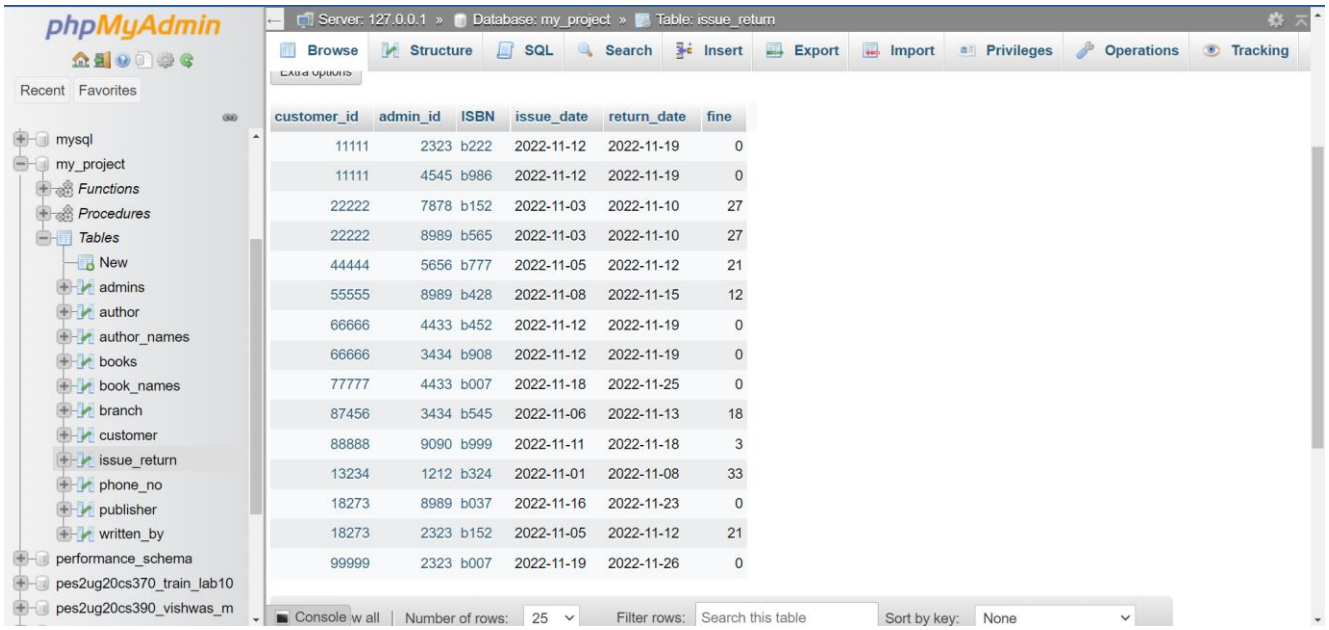



phpMyAdmin
Recent Favorites
mysql
my_project
Functions
Procedures
Tables
New
admins
author
author_names
books
book_names
branch
customer
issue_return
phone_no
publisher
written_by
performance_schema
pes2ug20cs370_train_lab10
Server: 127.0.0.1 » Database: my_project » Table: book_names
Browse Structure 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
intoduction 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

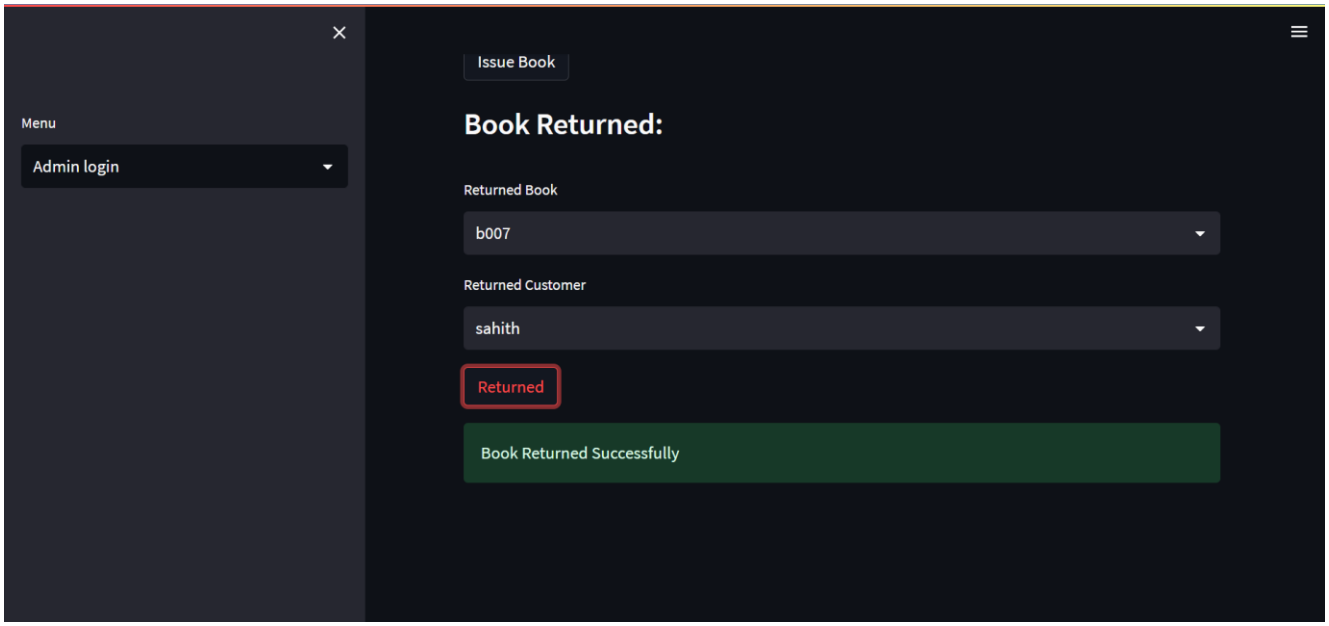

phpMyAdmin
Recent Favorites
mysql
my_project
Functions
Procedures
Tables
New
admins
author
author_names
books
book_names
branch
customer
issue_return
phone_no
publisher
written_by
performance_schema
pes2ug20cs370_train_lab10
pes2ug20cs390_vishwas_m
Server: 127.0.0.1 » Database: my_project » Table: books
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking
ISBN price category status title admin_id publisher_id
Edit Copy Delete b667 1444 novel yes the immortals of meluha 4433 4000
Edit Copy Delete b723 899 novel yes the oath of vayuputras 7878 2000
Edit Copy Delete b760 1233 academics yes intoduction to modern cryptography 7878 4000
Edit Copy Delete b767 1555 academics yes computer and internet security 9090 2000
Edit Copy Delete b769 2222 novel yes sita:warrior of mithila 2323 2000
Edit Copy Delete b774 243 academics yes concepts of programming languages 2323 4000
Edit Copy Delete b777 454 novel yes wuthering heights 6767 2000
Edit Copy Delete b826 222 fiction no harry potter and the chamber of secrets 4545 1000
Edit Copy Delete b908 1800 academics no machine learning 8989 1000
Edit Copy Delete b986 543 fiction no revolution 2020 3434 2000
Edit Copy Delete b998 2132 kids yes the adventures of tintin 8989 4000
Edit Copy Delete b999 1090 novel yes the 3 mistakes of my life 1212 3000
Check all With selected: Edit Copy Delete Export
Console

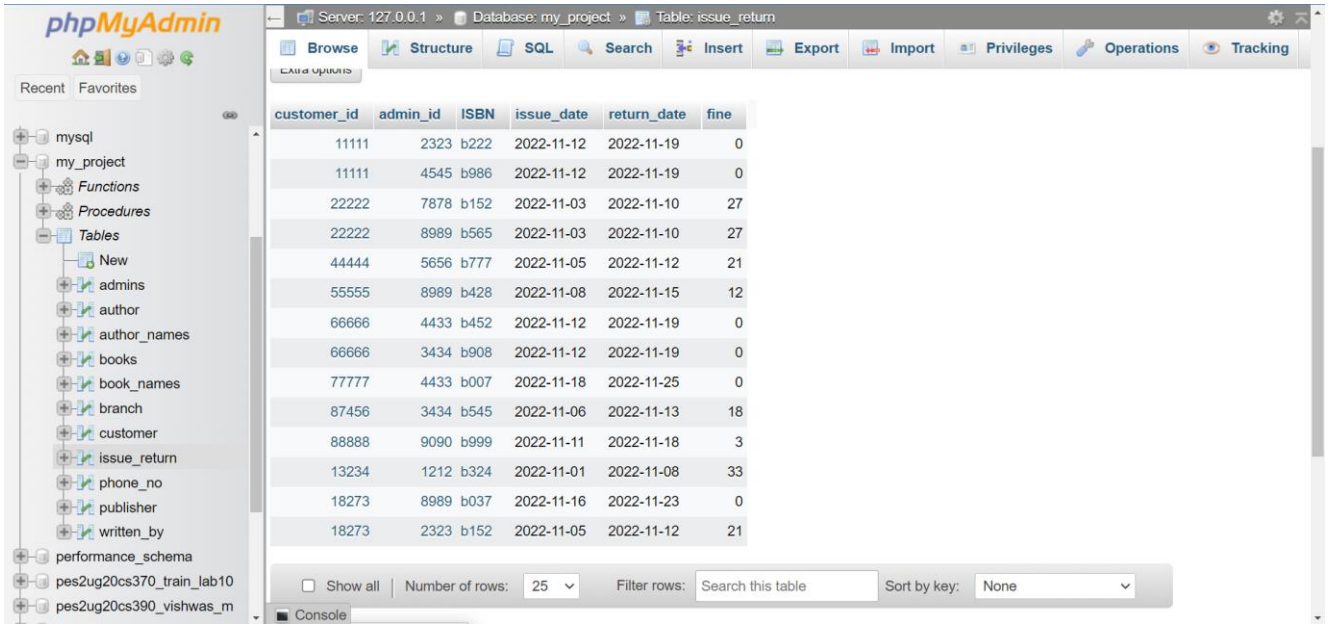
4) Issuing books to customers:



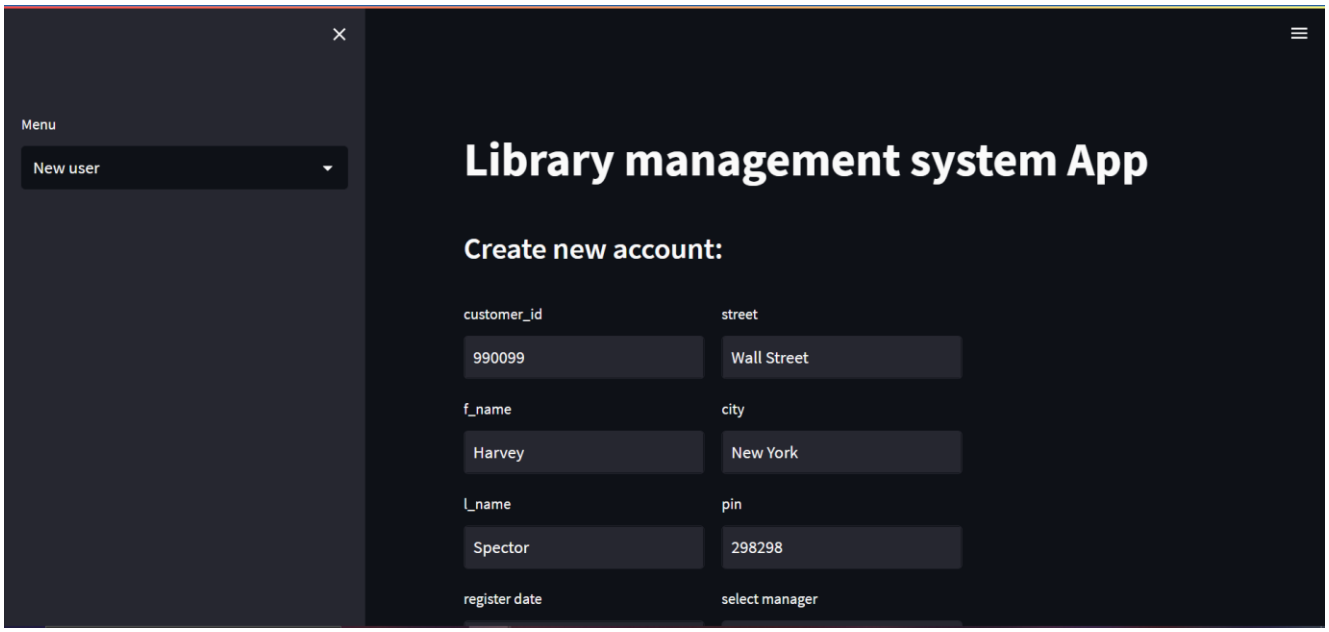


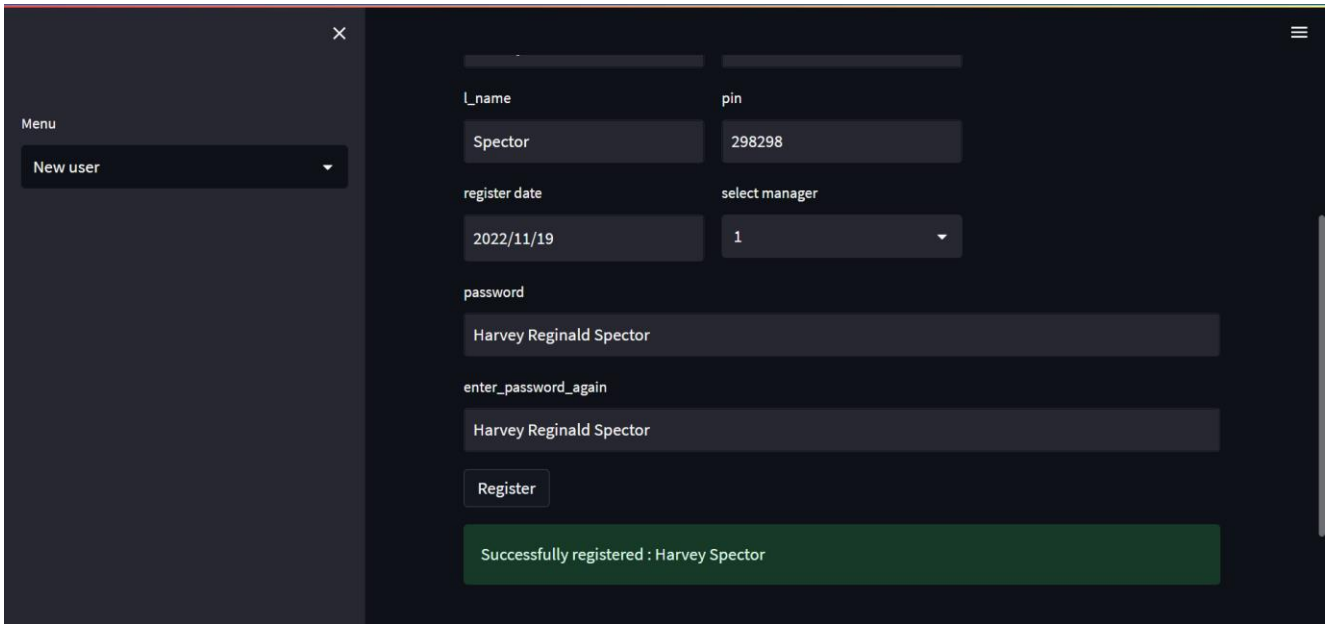
5) Returning Book back to the library:





Create New User:





☐	Edit Copy Delete	99999	sahith	reddy	11	nandyal	659766	2022-10-02	3	sahith
☐	Edit Copy Delete	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
        else:
```

```
            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,passw
ord)
                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,passw
ord):
    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.