Work Integrated Learning Programmes

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Assignment

Subject Name	**Database Design and Applications**
Subject Code	**SSZG518**
Topic Name	**Student Library Management System**

Version Number	Date	Author/Owner	Description of Change
1.1	29/09/2021	Group 5	First release

1. Team Members and their Responsibilities

Dominic Quazi	2021MT12390@wilp.bits-pilani.ac.in	Team lead, scope finalization, DB design and review, SQL query, Stored procedures and end to end implementation review
Jinendra Rathod	2020mt93223@wilp.bits-pilani.ac.in	ER and relational schema review, SQL Query formation and execution, RS document update,SQL query execution
Saquib	2021mt12266@wilp.bits-pilani.ac.in	Creation of Relational Schema Diagram , and Object Model / /UML diagram creation , ER diagram review , SQL query execution and testing.Running Stored Procedure testing.
Anishu	2021mt12453@wilp.bits-pilani.ac.in	ER review, Insert data in the table , SQL query execution (DDL)
Madhusudhanan	2021mt12430@wilp.bits-pilani.ac.in	ER review, Infrastructure setup , Schema creation in DB, SQL query execution.

Aparna Mondal	2021mt12168@wilp.bits-pilani.ac.in	DB use case specification , ER and relational schema review, Normalization, design and development of stored procedure . SQL query execution. RS document update.
Batra Ashutosh	2021mt12025@wilp.bits-pilani.ac.in	ERdiagram , Infrastructure setup , Table Schema creation in DB , SQL query execution,development and test of stored procedure, RS document update.
Nitish Kumar D	2021mt12457@wilp.bits-pilani.ac.in	SQL Query Creation for Statistics, SQL Query formation, Soft delete feature and testing, RS Document Update
Deepak Jain	2021mt12286@wilp.bits-pilani.ac.in	Relational database schema Update, SQL Query formation, RS Document Update

2. Problem Statement & Requirements Definition

STUDENT LIBRARY MANAGEMENT SYSTEM:

Often the library management system involves a lot of manual paperwork. Calculation of fines is in practice in several study institutions. This project will help the librarian as well as students to get the inventory information related to books. It helps students to check their profile and eligibility for borrowing books. It also helps to get rid of a lot of error prone human intervention which sometimes causes conflict as well.

This database system will have 2 views (Who will be the users) :

i) Librarian view.
ii) Student view

Functions & Features of Application :

A) Requirements from Librarian's perspective:

i) Membership management (Create / Delete/ Update).
ii) Search book inventory based on publisher name / author / title / edition.
iii) Issue the book to the student if available and update the inventory.
iv) Able to view the pending fine before issuing the book to students.
v) Able to soft delete book issual
v) Clear the fine against books received from the student.
vi) Return book to inventory if the pending fine is clear (if applicable).
vii) Able to get an overview of the current position of all books

B) Requirements from Students' perspective:

i) View book inventory (Generate library report)
ii) Generate student profile report (Generate user profile report)

- Student details
- Issued books along with return date and fine details
- Generate fine payment history to indicate all previous defaulter cases .

iii) Request for books based on publisher name/author/title/edition. (Generate issue report)

iii) Check the return date against each borrowed book.

iv) Check the fine corresponding to each book / total fine for all borrowed books.

v) Return book(s) to the library on time .

vi) Pay the fine against each book to enable the user to request for the book again . (Generate book return reports along with fine details)

vii) Able to get an overview of how all its members use the services provided.

Primarily there will be below mentioned output:

i) Generate library report.
ii) Generate user profile report.
iii) Generate issue report.
iv) Generate book return reports along with fine details .
v) Generate fine payment reports indicating full /partial payment .
vi) Generate fine payment history
vii) Generate statistics reports

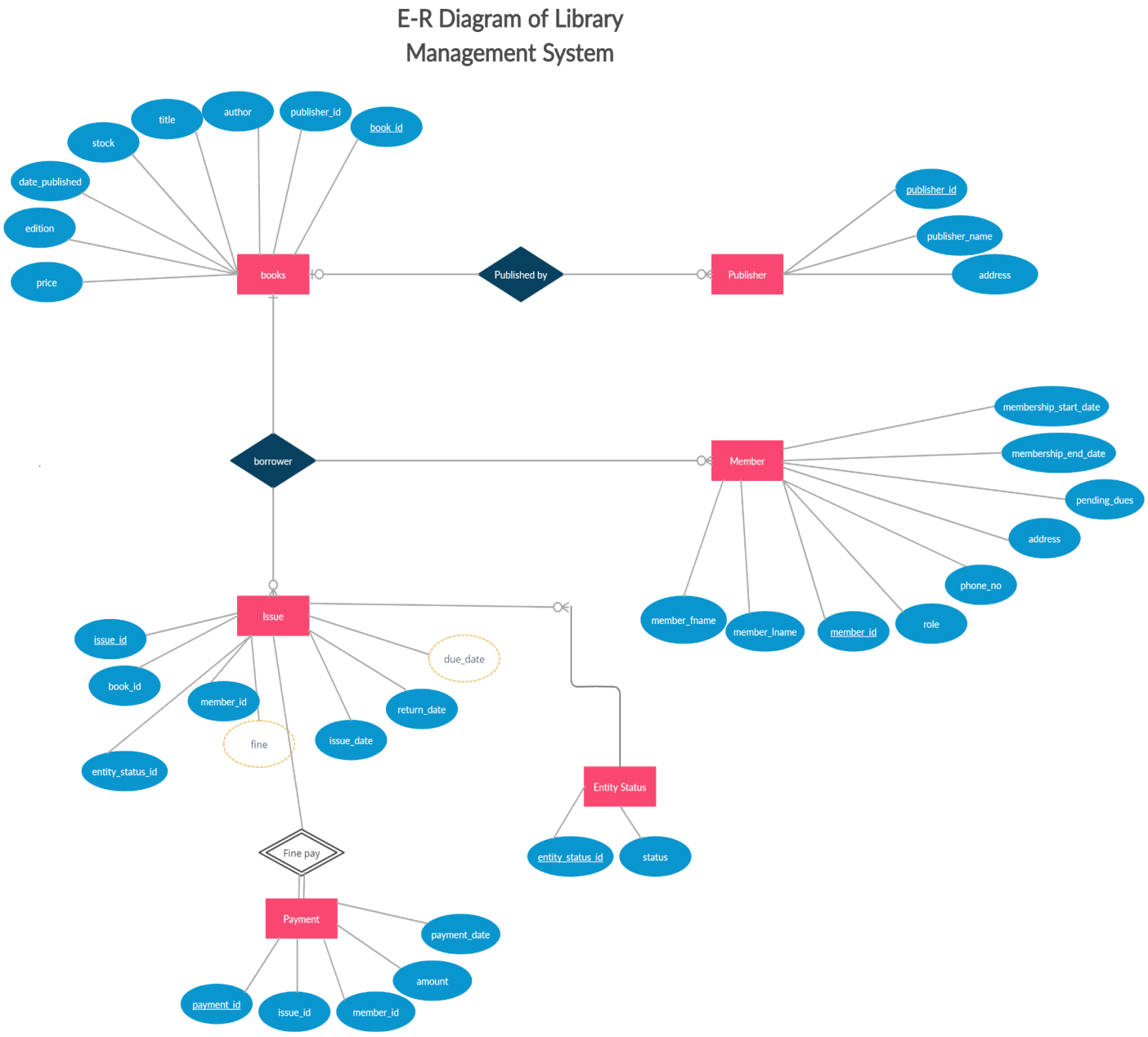
Assumption:
Book details are preloaded into the database system.
Payment for the fine will be made against each book. Also, partial payment is not accepted.

What are the benefits of this application:

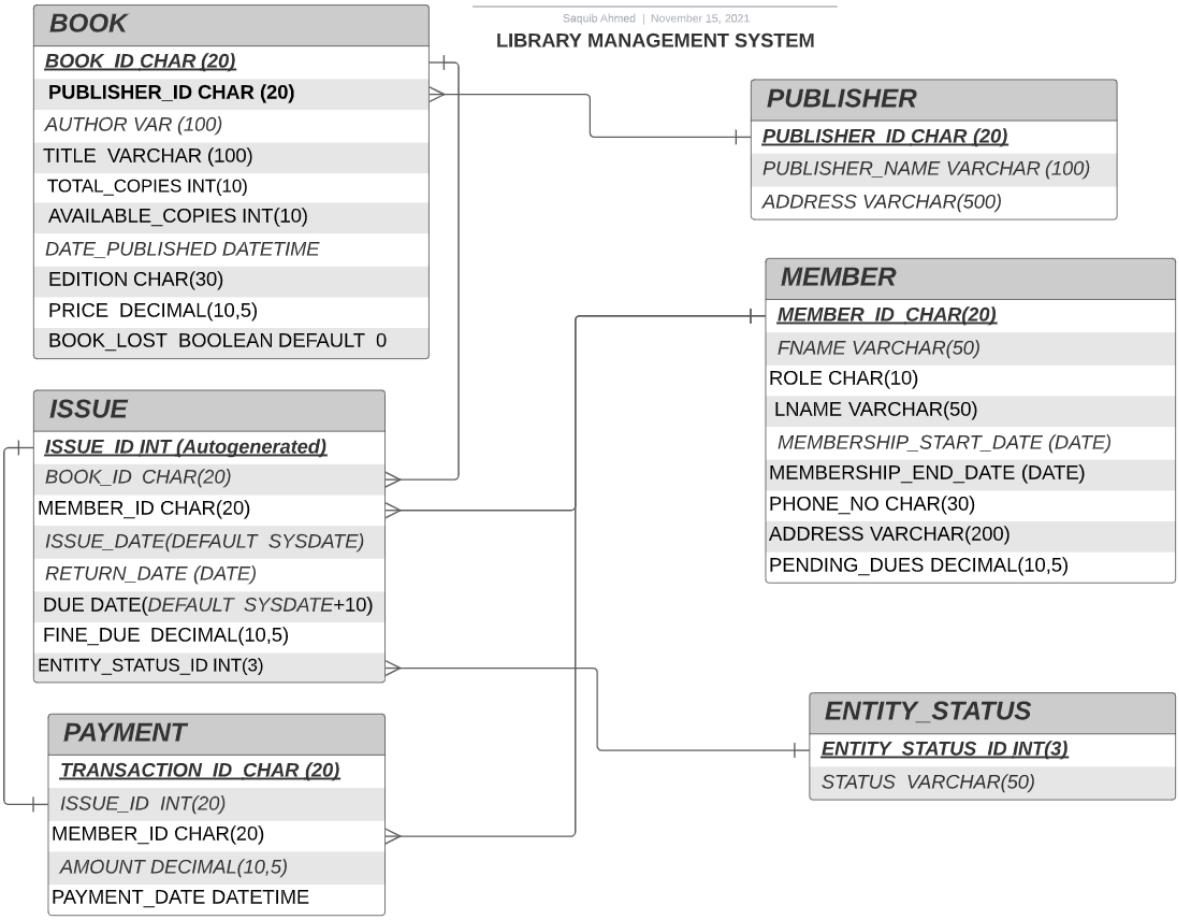
- There are lots of books and it is quite difficult to manage & also requires staff and time to manage it.One of the key benefits of using the Library management system you can easily,quickly carry out activities like inventory,serial control,book circulation,track information,Remove manual processes to issue books and maintain records.
- The library's executives Library Management system empowers the approved staff to keep legitimate records of all subtleties of the books like writer, release, duplicates, issue and return dates and different fines that are charged.Many students can access this application at the time.

Database would help student to check which books are available and if they are eligible to borrow books or not and also review return date and also keep track of fine.From Librarian point of view, they can check user profile and check book return date and based on it check fine and send fine reports,issue & return report.

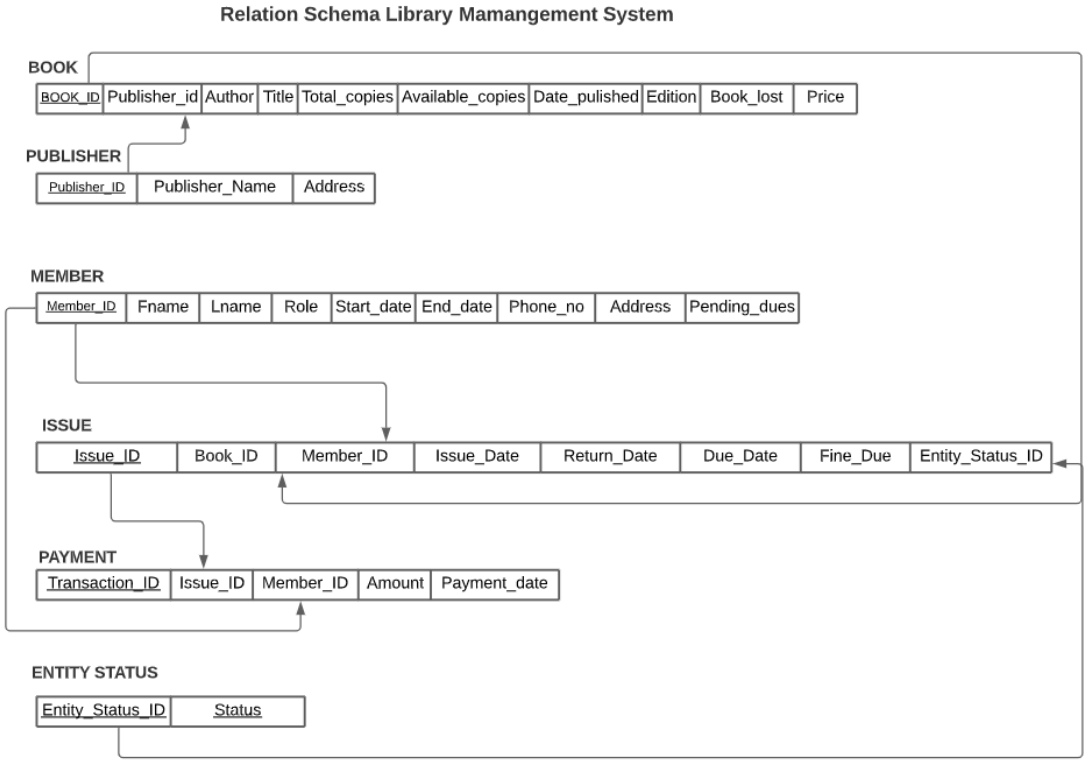
3. Entity Relationship Model



4. Object Model Diagram



5. Relational Database Schema



6. Normalization

Below are the verification points considered for normalization check:

1. 1NF : All attributes hold atomic values
2. 2NF: No partial functional dependency exists.
3. 3NF: No transitive functional dependency exists
4. BCNF: Relation is in 3 NF and in all FDs X->Y, X is a super key.

Normalisation steps performed :

i) The initial version of the books table was like below. We could identify the partial dependency as publisher ID which is a subset of the primary key of the books relation.It was violating 2NF rule which indicates no non-key attribute should be dependent on part of the key. Hence we decomposed the books relation into 2 relations : Book and Publisher to conform to the 2NF rule.

Relation name	Attribute names	Candidate key	Function dependency
Book	BOOK_Id, Publisher_id, Author, Title, Total copies, Available_copies, Date_published Edition , Book_lost Price	Book_Id	All non-key attributes Depends on the candidate key alone. And publisher_id is kept as foreign key referring to the publisher relation.
Publisher	Publisher_id, Publisher_Name, Address	Publisher_id	

We can find more description about functional dependencies for all relation below:

Relation	**Candidate Key**	**Functional Dependency**
book	Book_Id	Book_Id-> Author, Title, Total_copies, Available_copies, Date_published, Edition , Book_lost, Price Total_copies -> Available_copies

publisher	Publisher_id	Publisher_id ->Publisher_Name, Address
member	Member_ID	Member_ID -> Fname, Name, Role, Start_date, End_date, Address, Pending_dues
issue	Issue_ID	Issue_ID,Book_ID,Member_ID -> Issue_Date, Return_Date, Due_Date, Fine_Due
payment	Transaction_ID	Transaction_ID,Issue_ID, Member_ID -> Amount, Payment,date
entitystatus	Entitu_Status_ID	Entity_Status_ID -> Status

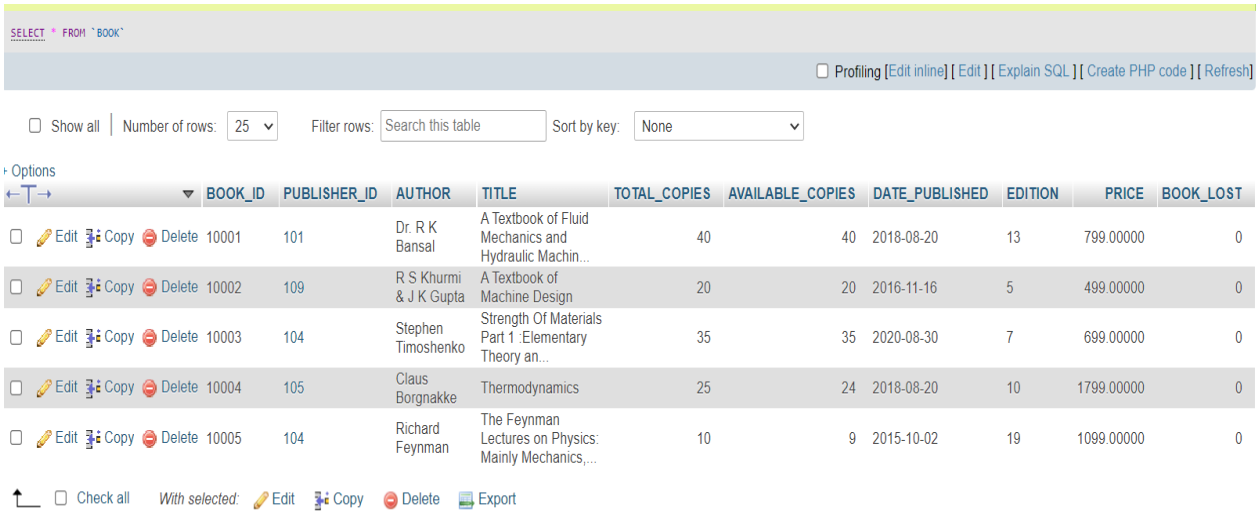
- *Books relation contains attributes which contains all atomic values . Also partial dependency is being removed after the initial round of discussions. There is no transitive dependency. All non key attributes depend on only superkey (Book_ID & publisher ID) Hence it conforms to all checks for 1NF, 2NF, 3NF . Since it contains one more functional dependency containing non-prime attributes depending on non-prime attributes, this one may not be conformed to BCNF .*
- *Publisher relation contains all atomic attributes. Also it does not contain any transitive / partial dependency . Also, all non-prime attributes are dependent on the super key of the table i.e Publisher_ID . Hence it conforms to 1NF, 2NF, 3NF as well as BCNF requirements.*
- *The Member relation also has a super key which is Member_ID. All other non-prime atomic attributes are dependent on this super key . Hence this table also conforms upto BCNF requirement as no transitive/ partial dependency is observed*
- *Issue relation contains member_ID, Book_ID as foreign key.When these are conjugated with Issue_ID , it becomes the super key of the relation . Also there are no multivalued attributes in this relation. All other attributes* Issue_Date, Return_Date, Due_Date, Fine_Due *depend on the super key strictly .Hence this conforms upto BCNF requirement.*
- *For transaction relation , transaction ID along with Member_ID and Issue_ID becomes the super key .*

- *For Entity status relation, this is a lookup table and Entity_Status_ID is a super key and the other attribute (Status) is dependent on this super key*

  But from the above explanations, we can realize that amongst all relation the highest normalisation form is 3NF. Hence this database conforms to 3NF .

7. Table Definitions and Data Contents

1. *<u>BOOK:</u>*



SELECT * FROM `BOOK`

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

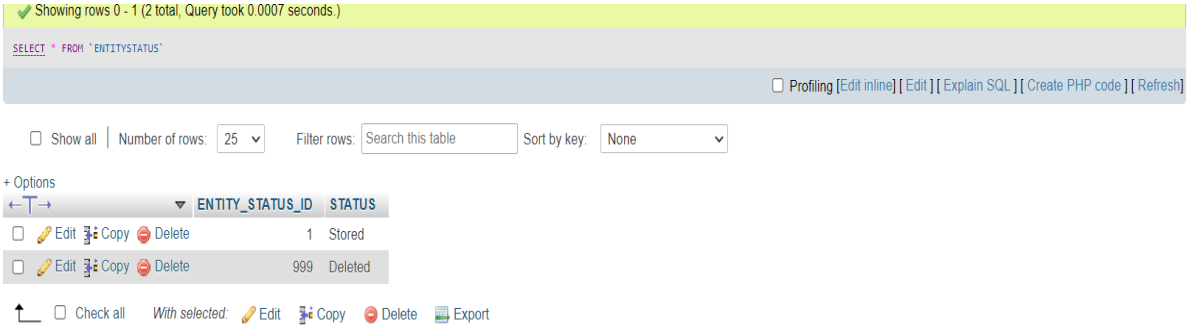
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Options

	BOOK_ID	PUBLISHER_ID	AUTHOR	TITLE	TOTAL_COPIES	AVAILABLE_COPIES	DATE_PUBLISHED	EDITION	PRICE	BOOK_LOST
Edit Copy Delete	10001	101	Dr. R K Bansal	A Textbook of Fluid Mechanics and Hydraulic Machin...	40	40	2018-08-20	13	799.00000	0
Edit Copy Delete	10002	109	R S Khurmi & J K Gupta	A Textbook of Machine Design	20	20	2016-11-16	5	499.00000	0
Edit Copy Delete	10003	104	Stephen Timoshenko	Strength Of Materials Part 1 :Elementary Theory an...	35	35	2020-08-30	7	699.00000	0
Edit Copy Delete	10004	105	Claus Borgnakke	Thermodynamics	25	24	2018-08-20	10	1799.00000	0
Edit Copy Delete	10005	104	Richard Feynman	The Feynman Lectures on Physics: Mainly Mechanics,...	10	9	2015-10-02	19	1099.00000	0

Check all With selected: Edit Copy Delete Export

2. *<u>ENTITYSTATUS:</u>*



Showing rows 0 - 1 (2 total, Query took 0.0007 seconds.)

SELECT * FROM `ENTITYSTATUS`

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

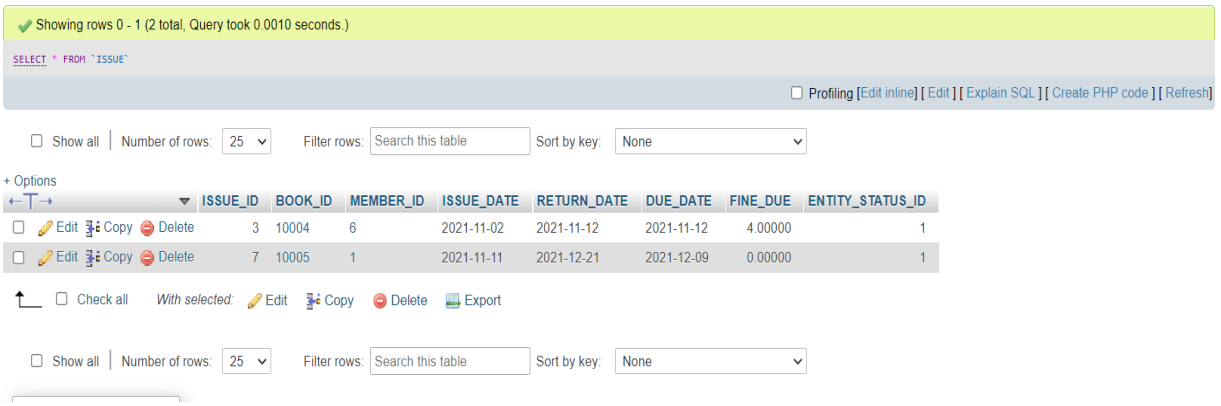
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

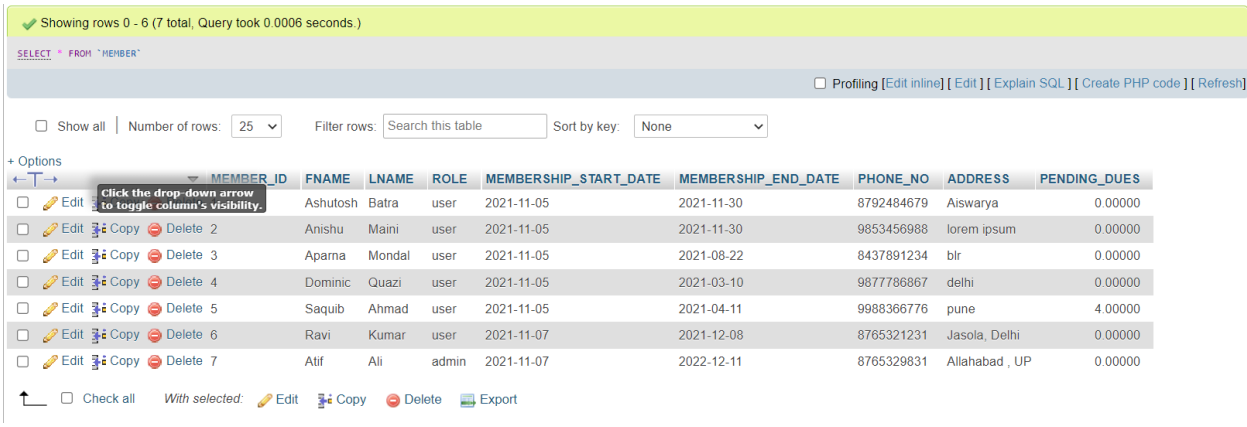
	ENTITY_STATUS_ID	STATUS
Edit Copy Delete	1	Stored
Edit Copy Delete	999	Deleted

Check all With selected: Edit Copy Delete Export

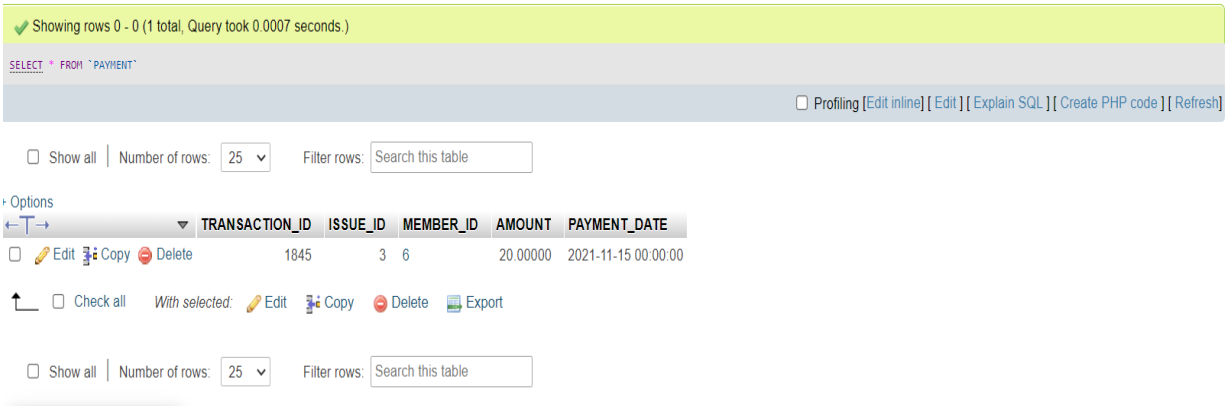
3. *<u>ISSUE:</u>*



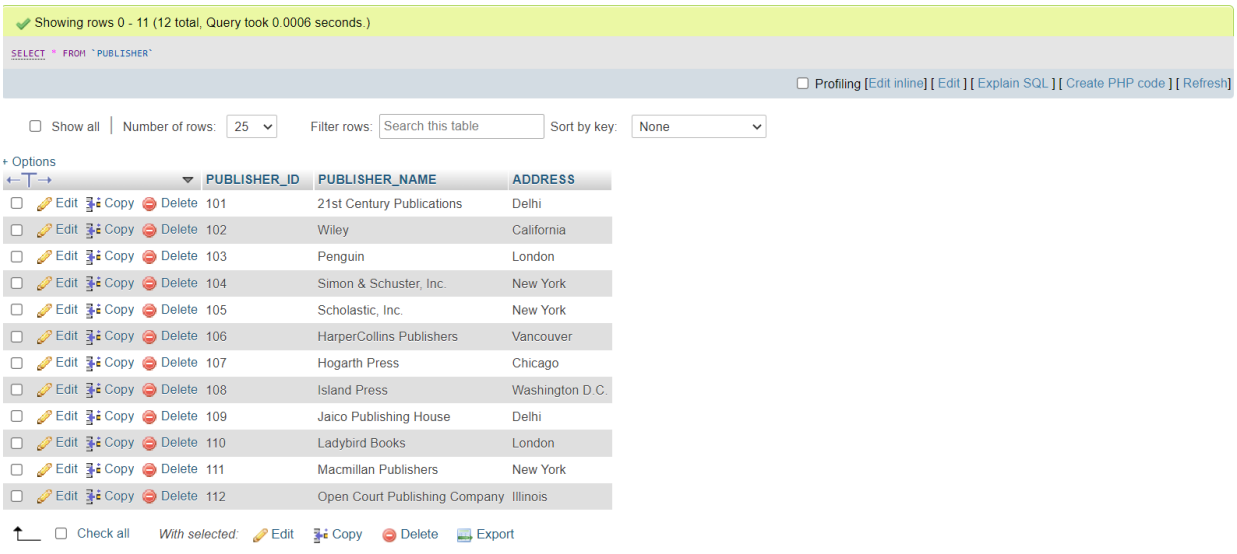
4. ***MEMBER:***



5. ***PAYMENT:***



6. PUBLISHER:



8. SQL Statements

DDL STATEMENTS

i) DB CREATION
MYSQL>**CREATE DATABASE IF NOT EXISTS** id17808854_lms;

ii) SHOW DB SCHEMA
MYSQL>**SHOW CREATE DATABASE** id17808854_lms;

iii) DISPLAY ALL DB
MYSQL>**SHOW DATABASES**;

iii) USE DB
MYSQL>**USE** id17808854_lms;

iv) CREATE PUBLISHER RELATION

```
CREATE TABLE IF NOT EXISTS publisher
  (
     publisher_id    CHAR(20) NOT NULL,
     publisher_name  VARCHAR(100) NOT NULL,
     address         VARCHAR(500),
     PRIMARY KEY(publisher_id)
  );
```

v) CREATE BOOK RELATION

```
CREATE TABLE IF NOT EXISTS book
```

```
  (
     book_id           CHAR(20) NOT NULL,
     publisher_id      CHAR(20) NOT NULL,
     author            VARCHAR(100) NOT NULL,
     title             VARCHAR(100) NOT NULL,
     total_copies      INT(10) NOT NULL,
     available_copies  INT(10) NOT NULL,
     date_published    DATE NOT NULL,
     edition           CHAR(30) NOT NULL,
     price             DECIMAL(10, 5) NOT NULL,
     book_lost         BOOLEAN NOT NULL DEFAULT 0,
     PRIMARY KEY(book_id),
     FOREIGN KEY (publisher_id)
REFERENCESpublisher(publisher_id)
  );
```

vi) CREATE MEMBER RELATION

```
CREATE TABLE IF NOT EXISTS member
  (
     member_id              CHAR(20) NOT NULL,
     fname                  VARCHAR(50) NOT NULL,
     lname                  VARCHAR(50),
     role                   CHAR(10),
     membership_start_date  DATE NOT NULL,
     membership_end_date    DATE NOT NULL,
     phone_no               CHAR(30),
     address                VARCHAR(200),
     pending_dues           DECIMAL(10, 5) NOT NULL DEFAULT
'0.00',
     PRIMARY KEY(member_id)
  );
```

vii) CREATE ISSUE RELATION

```
CREATE TABLEIF NOT EXISTS issue
(
      issue_id int NOT NULL auto_increment ,
      book_id char(20) NOT NULL,
      member_id char(20) NOT NULL,
      issue_date date NOT NULL DEFAULT CURRENT_TIMESTAMP,
      return_date date,
      due_date date NOT NULL DEFAULT date_add(CURRENT_TIMESTAMP,
      interval 10 day),
      fine_due decimal(10,5) NOT NULL DEFAULT '0.00',
      entity_status_id int NOT NULL DEFAULT '1',
```

```
    PRIMARY KEY(issue_id),
    FOREIGN KEY (book_id) REFERENCES book(book_id), FOREIGN
    KEY (entity_status_id) REFERENCES
    entitystatus(entity_status_id, FOREIGN KEY (member_id)
    REFERENCES member(member_id)
);
```

viii) CREATE PAYMENT RELATION

```
CREATE TABLE IF NOT EXISTS payment
  (
    transaction_id CHAR(20) NOT NULL,
    issue_id       CHAR(20) NOT NULL,
    member_id      CHAR(20) NOT NULL,
    amount         DECIMAL(10, 5) NOT NULL DEFAULT '0.00',
    payment_date   DATETIME NOT NULL,
    PRIMARY KEY(transaction_id),
    FOREIGN KEY (member_id) REFERENCES member(member_id)
  );
```

ix) CREATE ENTITY STATUS RELATION

```
CREATE TABLE IF NOT EXISTS entitystatus
  (
    entity_status_id INT(3) NOT NULL,
    status           VARCHAR(50) NOT NULL,
    PRIMARY KEY(entity_status_id)
  );
```

Initial Values:

INSERT into ENTITYSTATUS values (1,'Stored');

INSERT into ENTITYSTATUS values (999,'Deleted');

Use Case specific DATA RETRIEVAL STATEMENTS

a) Membership management (Create / Delete/ Update):

```
INSERT INTO `MEMBER` (`MEMBER_ID`, `FNAME`, `LNAME`, `ROLE`,
  `MEMBERSHIP_START_DATE`, `MEMBERSHIP_END_DATE`, `PHONE_NO`,
```

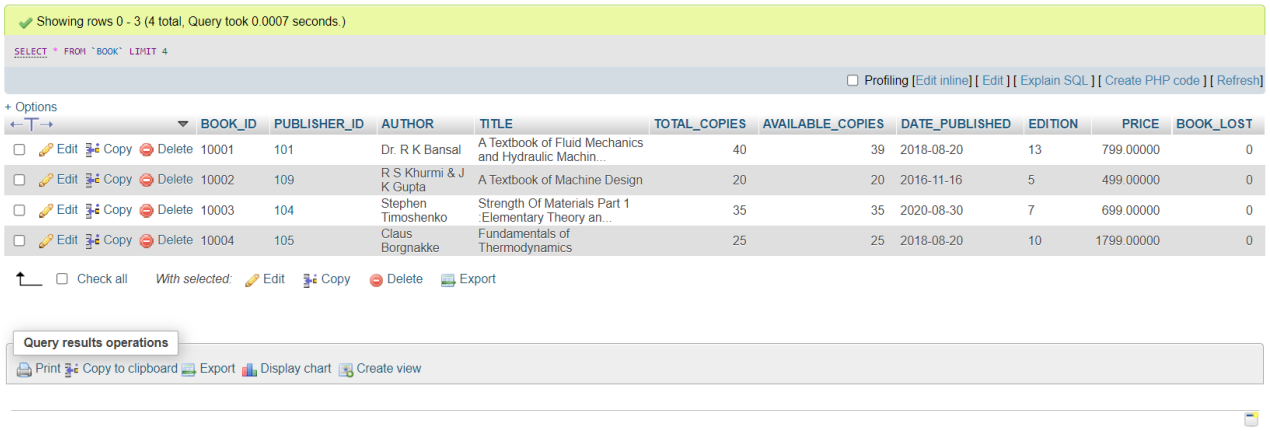
```
`ADDRESS`, `PENDING_DUES`) VALUES ('', 'Ravi ', 'Kumar', 'user', '2021-11-07',
'2021-12-08', '8765321231', 'Jasola, Delhi ', '0.00000');

INSERT INTO `MEMBER` (`MEMBER_ID`, `FNAME`, `LNAME`, `ROLE`,
`MEMBERSHIP_START_DATE`, `MEMBERSHIP_END_DATE`, `PHONE_NO`,
`ADDRESS`, `PENDING_DUES`) VALUES ('', Atif', Ali, admin, '2021-11-07',
'2021-12-08', '8765321231', 'Jasola, Delhi ', '0.00000');
```

Update member address:

```
UPDATE MEMBER set ADDRESS = 'Aiswarya' WHERE MEMBER_ID = 1
```

b) Search book in the inventory



Showing rows 0 - 3 (4 total, Query took 0.0007 seconds.)

```
SELECT * FROM `BOOK` LIMIT 4
```

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

+ Options

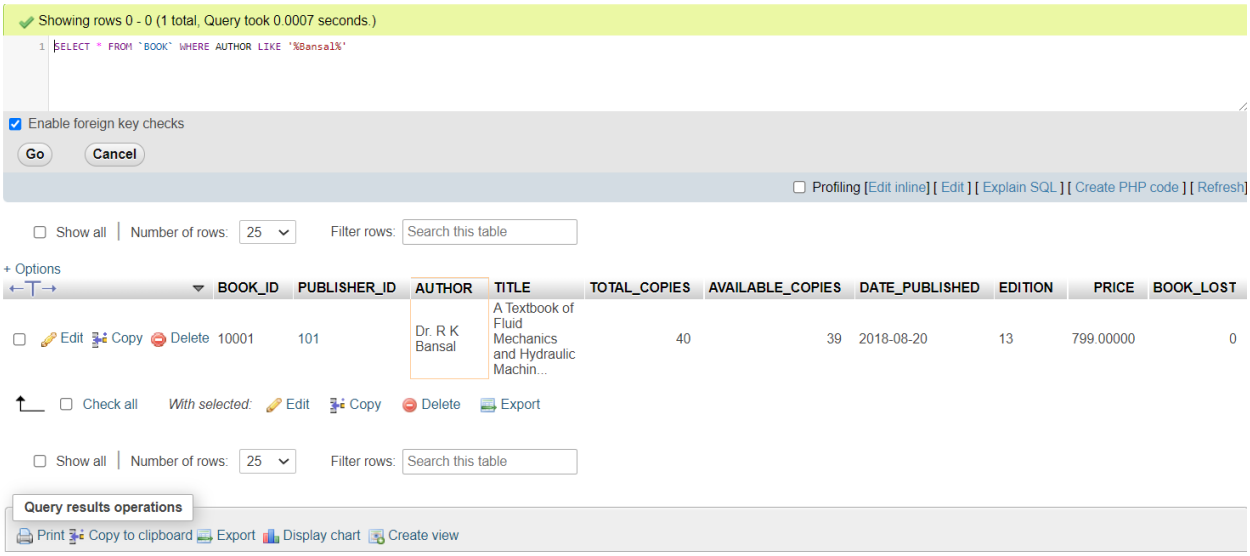
	BOOK_ID	PUBLISHER_ID	AUTHOR	TITLE	TOTAL_COPIES	AVAILABLE_COPIES	DATE_PUBLISHED	EDITION	PRICE	BOOK_LOST
Edit Copy Delete	10001	101	Dr. R K Bansal	A Textbook of Fluid Mechanics and Hydraulic Machin...	40	39	2018-08-20	13	799.00000	0
Edit Copy Delete	10002	109	R S Khurmi & J K Gupta	A Textbook of Machine Design	20	20	2016-11-16	5	499.00000	0
Edit Copy Delete	10003	104	Stephen Timoshenko	Strength Of Materials Part 1 :Elementary Theory an...	35	35	2020-08-30	7	699.00000	0
Edit Copy Delete	10004	105	Claus Borgnakke	Fundamentals of Thermodynamics	25	25	2018-08-20	10	1799.00000	0

Check all *With selected:* Edit Copy Delete Export

Query results operations

Print Copy to clipboard Export Display chart Create view

c) Search the book based on Author name



Showing rows 0 - 0 (1 total, Query took 0.0007 seconds.)

```
1 SELECT * FROM `BOOK` WHERE AUTHOR LIKE '%Bansal%'
```

Enable foreign key checks

Go Cancel

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

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	BOOK_ID	PUBLISHER_ID	AUTHOR	TITLE	TOTAL_COPIES	AVAILABLE_COPIES	DATE_PUBLISHED	EDITION	PRICE	BOOK_LOST
Edit Copy Delete	10001	101	Dr. R K Bansal	A Textbook of Fluid Mechanics and Hydraulic Machin...	40	39	2018-08-20	13	799.00000	0

Check all *With selected:* Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table

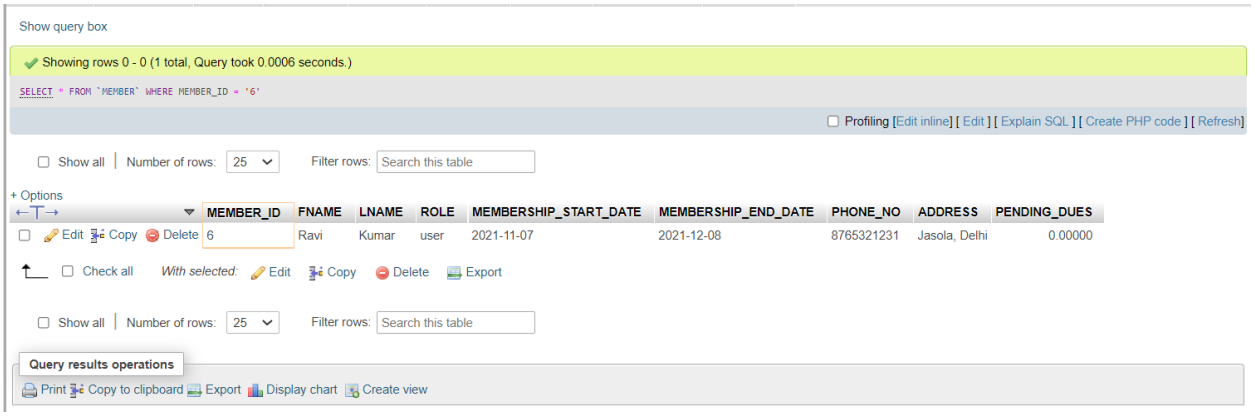
Query results operations

Print Copy to clipboard Export Display chart Create view

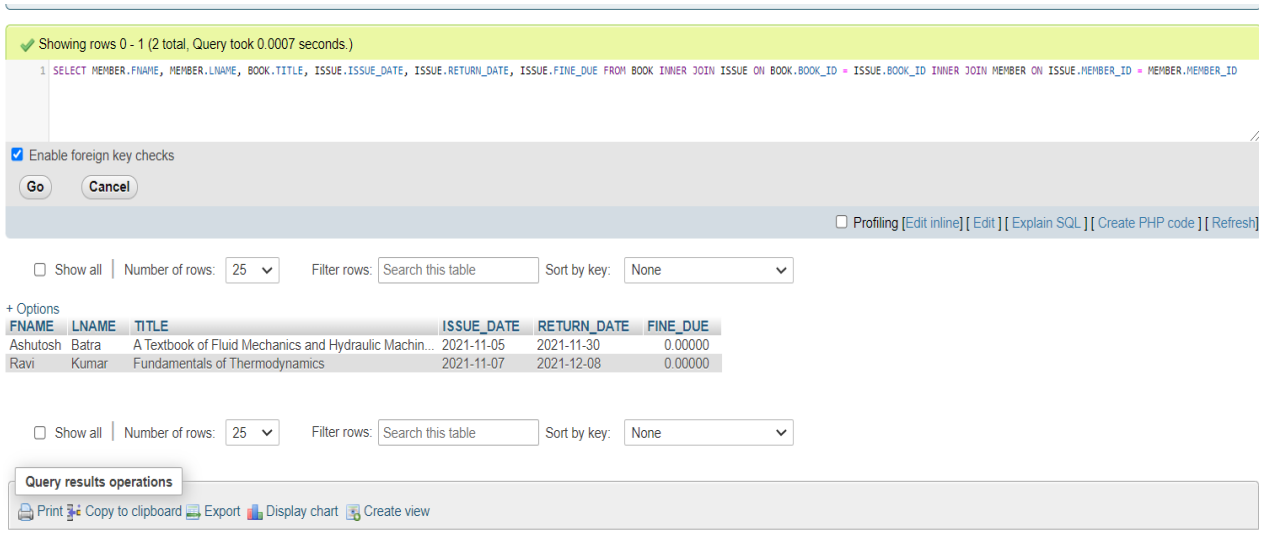
d) Issue the book to the student if available and update the inventory.

For example , user Ravi kumar requests with member ID 6 requests librarian to issue Thermodynamics book.

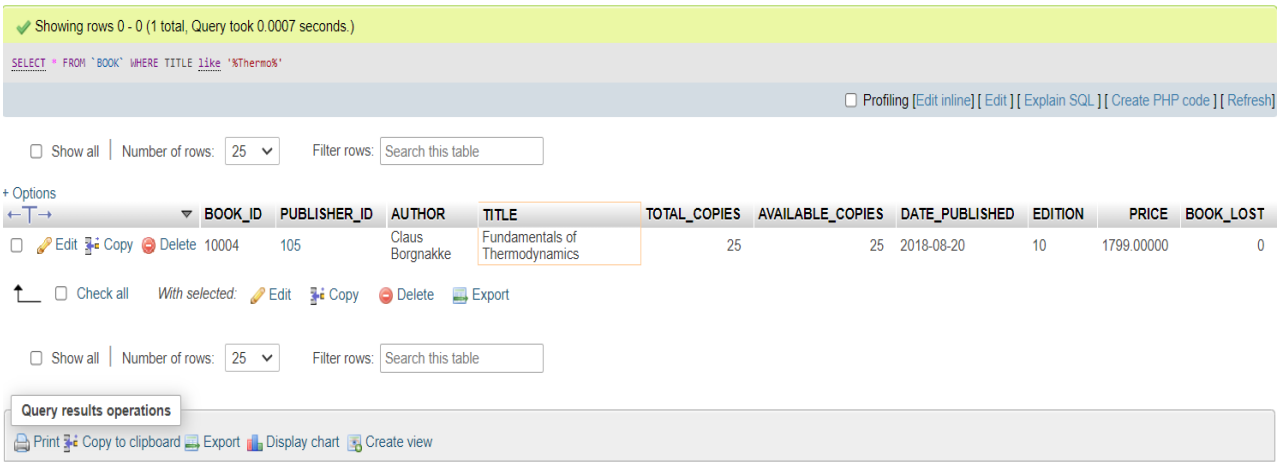
Retrieve member details with member ID



Admin view to see the issued copies for the member and their dues



Search the availability status based on book name



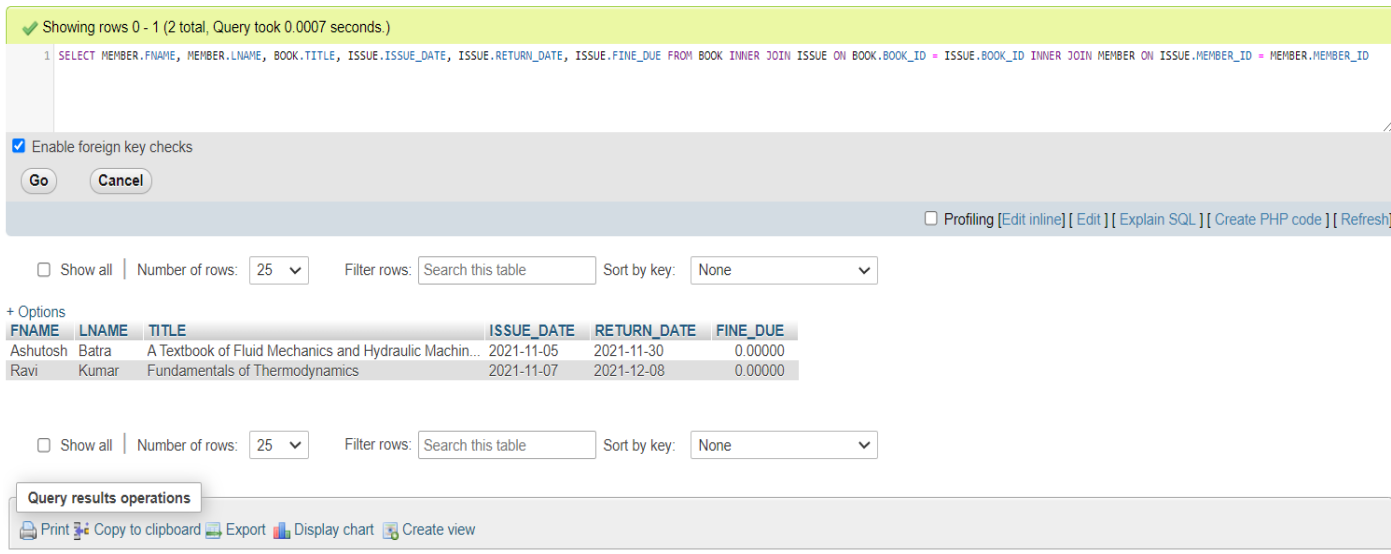
Stored procedure also written for this purpose :

GetAvailableCopiesFromTitle

Arguments : IN , Varchar (255)

```
BEGIN
    SELECT BOOK.AVAILABLE_COPIES FROM BOOK WHERE BOOK.TITLE LIKE
    CONCAT('%', BookName , '%');
END
```

Able to view the pending fine before issuing the book to students.

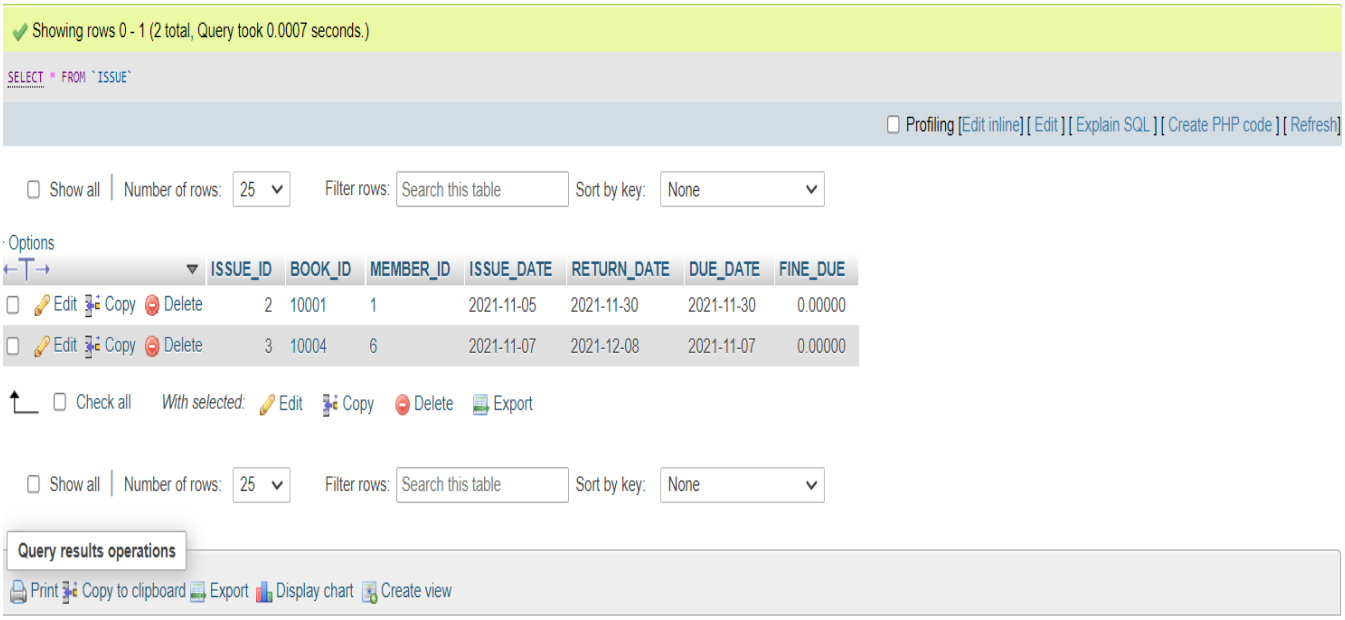


SQL command to INSERT into Issue table based on Member ID, Book Id

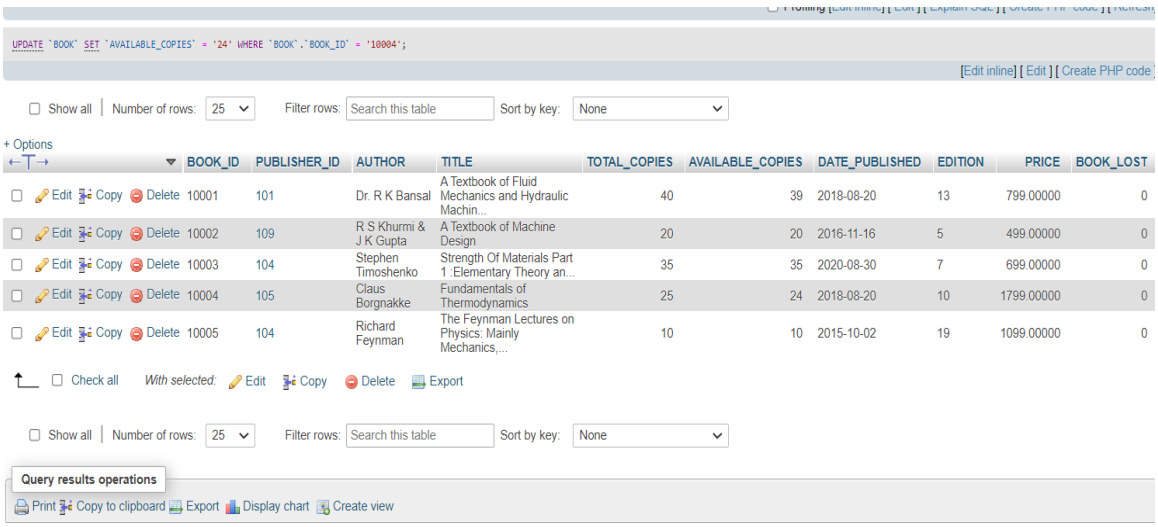
```
INSERT INTO `ISSUE` (`ISSUE_ID`, `BOOK_ID`, `MEMBER_ID`, `ISSUE_DATE`,
    `RETURN_DATE`, `DUE_DATE`, `FINE_DUE`) VALUES ('3', '10004', '6', '2021-11-07',
    '2021-12-08', '2021-11-07', '0.00000');
```

After the above SQL command , we can see the ISSUE table get populated with the required entries. The book with name matching thermodynamics subject was issued to Ravi Kumar. Also we can see that the number of available copies gets reduced by 1 .

SELECT * FROM `ISSUE`



Snapshot after a book issued to user (Thermodynamics book issues to Ravi kumar)



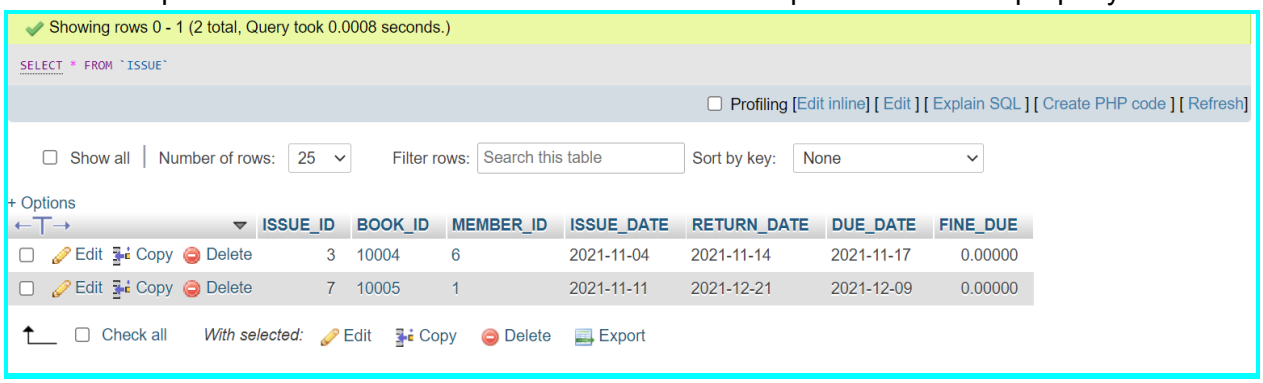
Procedure written to update issue table based on member ID and book name

Procedure name: IssueBook (Varchar, INT)

```
BEGIN
   DECLARE max_issueID INT;
   CALL `GetBookId`(BookName, @book_id);
SELECT MAX(ISSUE.ISSUE_ID) INTO max_issueID FROM ISSUE WHERE
ISSUE.ENTITY_STATUS_ID!=999;
   INSERT INTO `ISSUE` (`ISSUE_ID`, `BOOK_ID`, `MEMBER_ID`, `ISSUE_DATE`,
    `RETURN_DATE`, `DUE_DATE`, `FINE_DUE`) VALUES (max_issueID+1, @book_id,
    MemberID, CURRENT_DATE, CURRENT_DATE+10, CURRENT_DATE+10, '0.00000');
   UPDATE BOOK SET BOOK.AVAILABLE_COPIES = BOOK.AVAILABLE_COPIES - 1
    WHERE BOOK.BOOK_ID = @book_id;
   SELECT "Book Issued Successfully";
END
```

Snapshot of issue table and books table before calling this procedure:

Available copies can be examined to crosscheck if the issue procedure works properly.



Showing rows 0 - 1 (2 total, Query took 0.0008 seconds.)

```
SELECT * FROM `ISSUE`
```

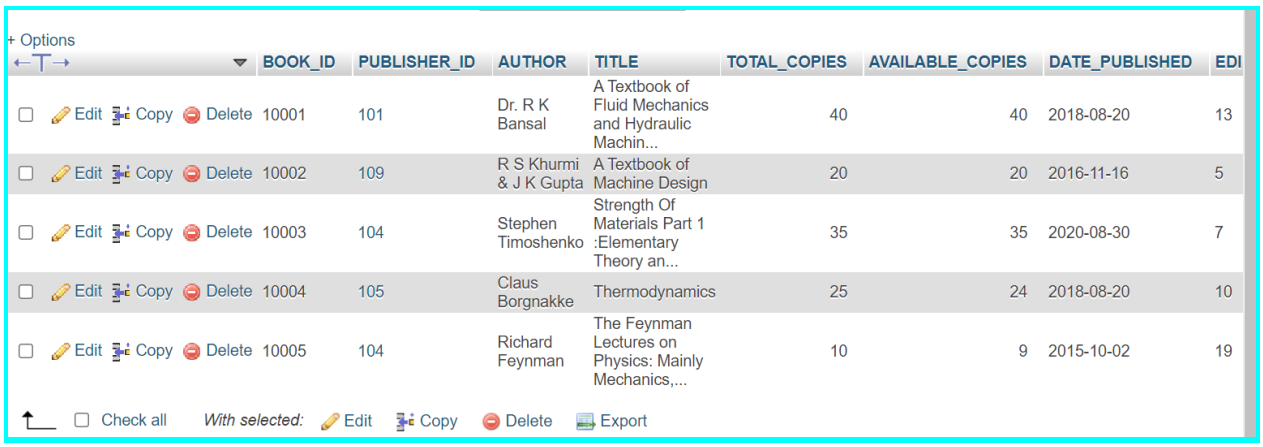
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

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

+ Options

	ISSUE_ID	BOOK_ID	MEMBER_ID	ISSUE_DATE	RETURN_DATE	DUE_DATE	FINE_DUE
Edit Copy Delete	3	10004	6	2021-11-04	2021-11-14	2021-11-17	0.00000
Edit Copy Delete	7	10005	1	2021-11-11	2021-12-21	2021-12-09	0.00000

Check all With selected: Edit Copy Delete Export



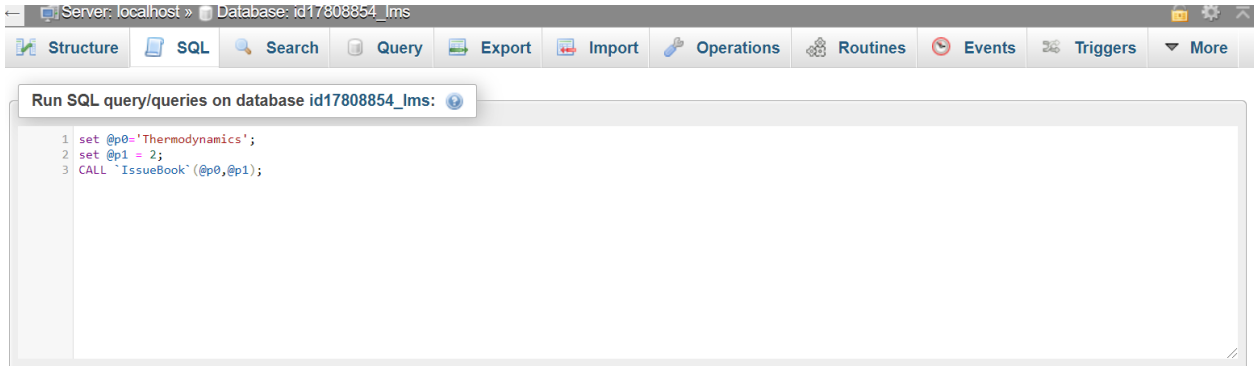
+ Options

	BOOK_ID	PUBLISHER_ID	AUTHOR	TITLE	TOTAL_COPIES	AVAILABLE_COPIES	DATE_PUBLISHED	EDI
Edit Copy Delete	10001	101	Dr. R K Bansal	A Textbook of Fluid Mechanics and Hydraulic Machin...	40	40	2018-08-20	13
Edit Copy Delete	10002	109	R S Khurmi & J K Gupta	A Textbook of Machine Design	20	20	2016-11-16	5
Edit Copy Delete	10003	104	Stephen Timoshenko	Strength Of Materials Part 1 :Elementary Theory an...	35	35	2020-08-30	7
Edit Copy Delete	10004	105	Claus Borgnakke	Thermodynamics	25	24	2018-08-20	10
Edit Copy Delete	10005	104	Richard Feynman	The Feynman Lectures on Physics: Mainly Mechanics,...	10	9	2015-10-02	19

Check all With selected: Edit Copy Delete Export

Issue procedure is called like below:

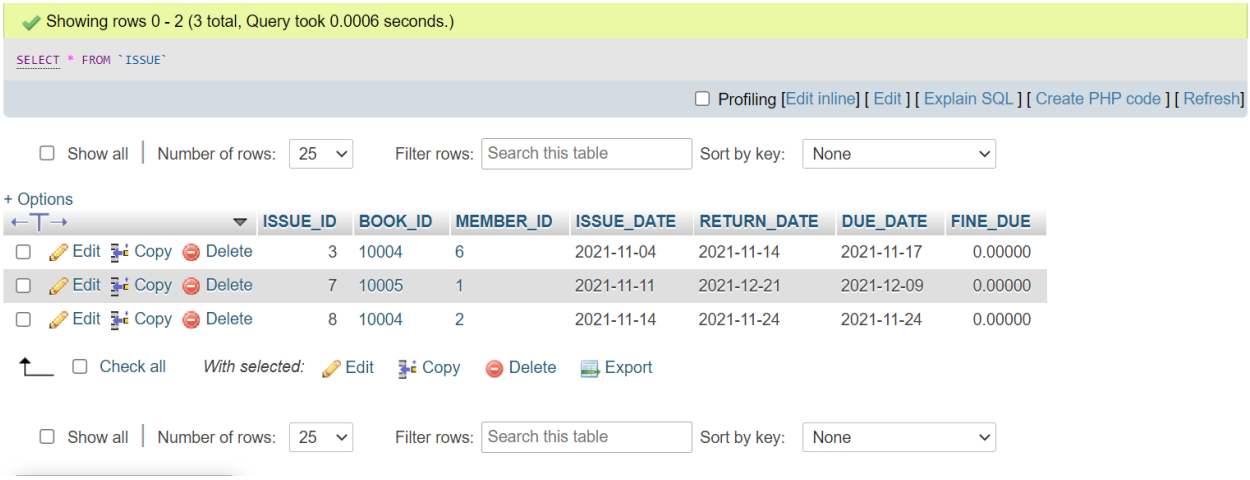
```
set @p0='Thermodynamics';
set @p1 = 2;
CALL `IssueBook`(@p0,@p1);
```



Output of the procedure execution:

Snapshot of Issue table after procedure is called :

We can notice that a new entry created with issueID = 8 as before inserting this entry max ISsue ID was 7 . Also the issue date is considered as today's date and return & due date is updated within 10 days from the issue date .



Showing rows 0 - 2 (3 total, Query took 0.0006 seconds.)

```
SELECT * FROM `ISSUE`
```

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

+ Options

	ISSUE_ID	BOOK_ID	MEMBER_ID	ISSUE_DATE	RETURN_DATE	DUE_DATE	FINE_DUE
Edit Copy Delete	3	10004	6	2021-11-04	2021-11-14	2021-11-17	0.00000
Edit Copy Delete	7	10005	1	2021-11-11	2021-12-21	2021-12-09	0.00000
Edit Copy Delete	8	10004	2	2021-11-14	2021-11-24	2021-11-24	0.00000

Check all With selected: Edit Copy Delete Export

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

Book table also got updated with the actual number of available copies of that particular book . Notice that thermodynamics named book available copies now became 23 from 24.

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

+ Options

	BOOK_ID	PUBLISHER_ID	AUTHOR	TITLE	TOTAL_COPIES	AVAILABLE_COPIES	DATE_PUBLISHED	ED
Edit Copy Delete	10001	101	Dr. R K Bansal	A Textbook of Fluid Mechanics and Hydraulic Machin...	40	40	2018-08-20	13
Edit Copy Delete	10002	109	R S Khurmi & J K Gupta	A Textbook of Machine Design	20	20	2016-11-16	5
Edit Copy Delete	10003	104	Stephen Timoshenko	Strength Of Materials Part 1 :Elementary Theory an...	35	35	2020-08-30	7
Edit Copy Delete	10004	105	Claus Borgnakke	Thermodynamics	25	23	2018-08-20	10
Edit Copy Delete	10005	104	Richard Feynman	The Feynman Lectures on Physics: Mainly Mechanics,...	10	9	2015-10-02	19

Check all With selected: Edit Copy Delete Export

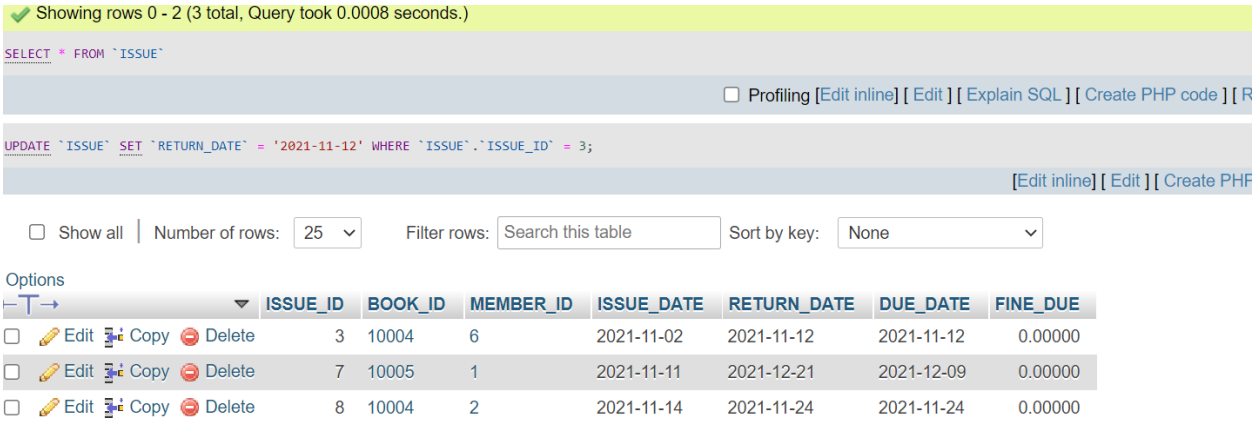
e) Return use case

Call updateFine procedure to update the fine amount for all issue based on due date : This method can be called by the librarian daily once he logs into the Library system.

updateFine ()

```
BEGIN
    SELECT * FROM ISSUE WHERE ISSUE.ENTITY_STATUS_ID!=999;
    UPDATE ISSUE
    SET ISSUE.FINE_DUE = (CURRENT_DATE - DUE_DATE) *10
    WHERE ISSUE.DUE_DATE < CURRENT_DATE AND WHERE
ISSUE.ENTITY_STATUS_ID!=999;
    SELECT * FROM ISSUE WHERE ISSUE.ENTITY_STATUS_ID!=999;
END
```

Snapshot of issue before calling this procedure :

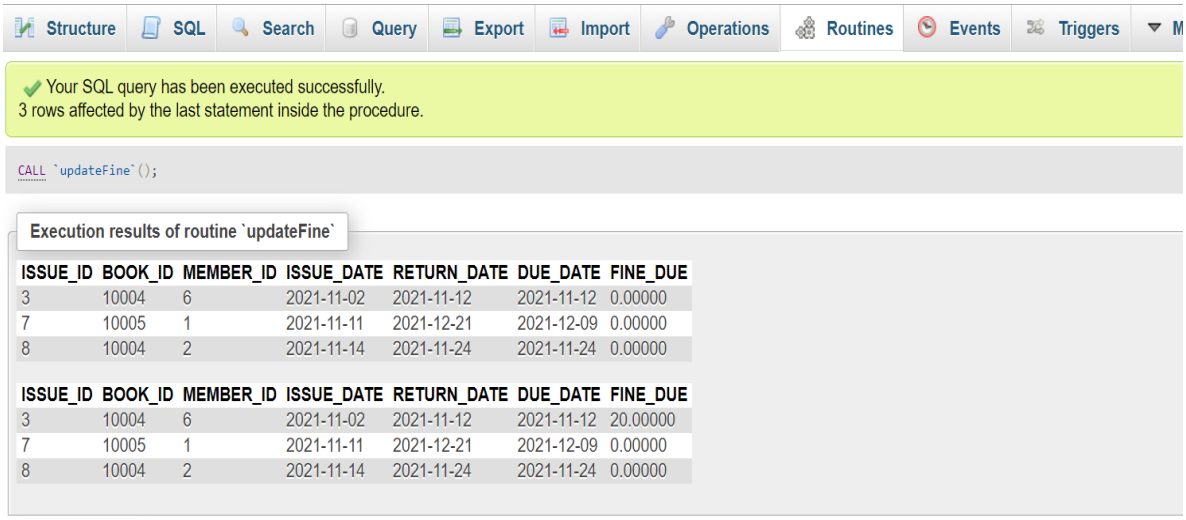


	ISSUE_ID	BOOK_ID	MEMBER_ID	ISSUE_DATE	RETURN_DATE	DUE_DATE	FINE_DUE
Edit Copy Delete	3	10004	6	2021-11-02	2021-11-12	2021-11-12	0.00000
Edit Copy Delete	7	10005	1	2021-11-11	2021-12-21	2021-12-09	0.00000
Edit Copy Delete	8	10004	2	2021-11-14	2021-11-24	2021-11-24	0.00000

Procedure is called like below :

“call updateFine”

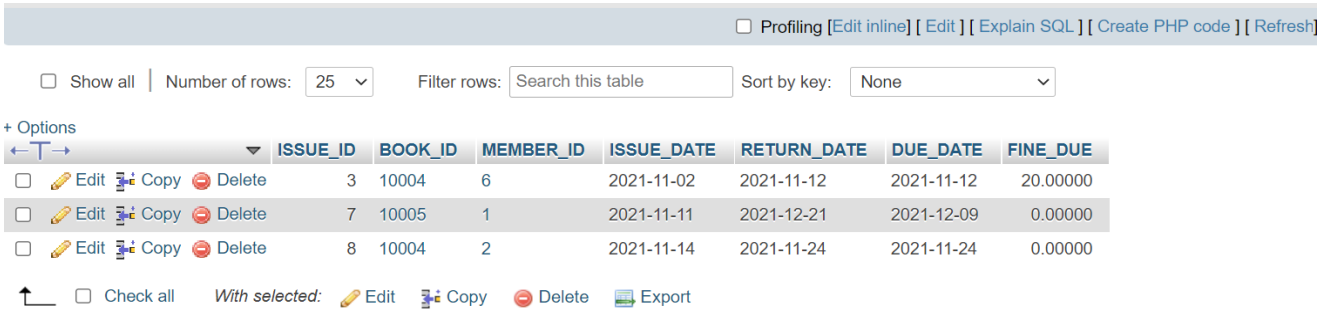
Output of the procedure:



ISSUE_ID	BOOK_ID	MEMBER_ID	ISSUE_DATE	RETURN_DATE	DUE_DATE	FINE_DUE
3	10004	6	2021-11-02	2021-11-12	2021-11-12	0.00000
7	10005	1	2021-11-11	2021-12-21	2021-12-09	0.00000
8	10004	2	2021-11-14	2021-11-24	2021-11-24	0.00000

ISSUE_ID	BOOK_ID	MEMBER_ID	ISSUE_DATE	RETURN_DATE	DUE_DATE	FINE_DUE
3	10004	6	2021-11-02	2021-11-12	2021-11-12	20.00000
7	10005	1	2021-11-11	2021-12-21	2021-12-09	0.00000
8	10004	2	2021-11-14	2021-11-24	2021-11-24	0.00000

Snapshot of issue after calling this procedure :



	ISSUE_ID	BOOK_ID	MEMBER_ID	ISSUE_DATE	RETURN_DATE	DUE_DATE	FINE_DUE
Edit Copy Delete	3	10004	6	2021-11-02	2021-11-12	2021-11-12	20.00000
Edit Copy Delete	7	10005	1	2021-11-11	2021-12-21	2021-12-09	0.00000
Edit Copy Delete	8	10004	2	2021-11-14	2021-11-24	2021-11-24	0.00000

- Validate member name based on member ID
- Get book ID based on book name
- Delete entry from issue table using member ID and book ID if fine = 0. Else prompt about fine due.
- Take the payment for that book ID as whole and then delete the entry from the issue table and update the available number of book copies .

Algorithm for Stored procedure

a) procedure returnBook(MemberID, BookID)
//SQL query to check fine
if fine > 0
return from procedure with the amount user has to pay
else
delete from issue table
increment the count of available copies

b) If return from procedure value indicates fine amount, pay the fine and again call return Book procedure.

// payFine (memberId, bookID, amount)

// SQL query to insert entry in payment table with the amount being passed.

// update fine to 0 in issue table

==========

GetBookId Procedure Creation:

```
BEGIN
        SELECT BOOK.BOOK_ID INTO book_id FROM BOOK WHERE BOOK.TITLE
LIKE CONCAT('%', book_name , '%');

END
```

GetBookId Procedure Parameter:

INPUT: book_name VARCHAR(255)
OUTPUT: book_id INT

GetBookId Procedure Call:

```
SET @book_name = ‘Theromodynamics’;
CALL `GetBookId`(@book_name);
SELECT @book_id;
```

ReturnBook Procedure Creation:

```
BEGIN
        DECLARE v_fine_due INT;
   CALL `GetBookId`(book_name, @book_id);
        SELECT ISSUE.FINE_DUE INTO v_fine_due FROM ISSUE WHERE
ISSUE.BOOK_ID=@book_id AND ISSUE.MEMBER_ID = member_id AND
ISSUE.ENTITY_STATUS_ID!=999;
   IF (v_fine_due != 0) THEN
        SELECT CONCAT('Pending due is ', v_fine_due);
        ELSE
                DELETE FROM ISSUE WHERE ISSUE.MEMBER_ID = member_id AND
ISSUE.BOOK_ID = @book_id;
      UPDATE BOOK SET BOOK.AVAILABLE_COPIES = BOOK.AVAILABLE_COPIES
+ 1 WHERE BOOK.BOOK_ID = @book_id;
      SELECT "Book Return Successfully";
        END IF;
END
```

Procedure Execution:

EXEC ReturnBook @member_id = '6', @book_id = '10004';

SET @member_id = '6';
SET @book_id = '10004';
CALL `ReturnBook` (@member_id , @book_id)

Case 1: If book returned within due date

Snapshot of Issue table before return of the book

+ Options

	ISSUE_ID	BOOK_ID	MEMBER_ID	ISSUE_DATE	RETURN_DATE	DUE_DATE	FINE_DUE
Edit Copy Delete	3	10004	6	2021-11-02	2021-11-12	2021-11-12	20.00000
Edit Copy Delete	7	10005	1	2021-11-11	2021-12-21	2021-12-09	0.00000
Edit Copy Delete	8	10004	2	2021-11-14	2021-11-24	2021-11-24	0.00000

Snapshot of the book table before return of the book :

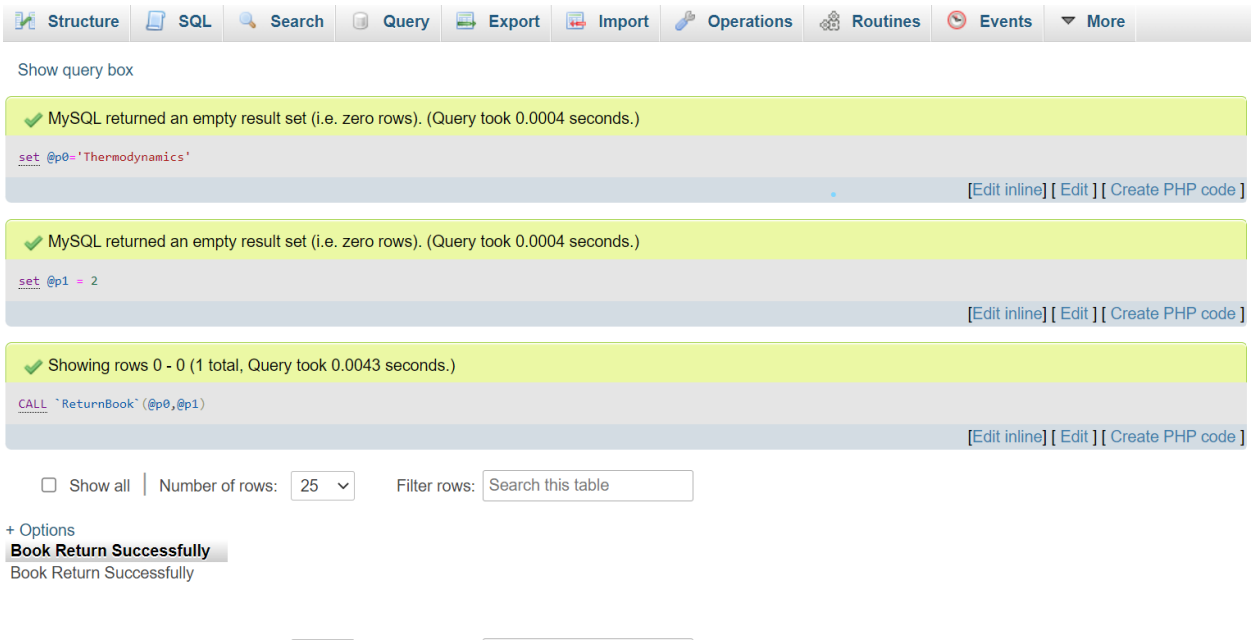
Edit Copy Delete	10001	101	Dr. R K Bansal	A Textbook of Fluid Mechanics and Hydraulic Machin...	40	40	2018-08-20	13
Edit Copy Delete	10002	109	R S Khurmi & J K Gupta	A Textbook of Machine Design	20	20	2016-11-16	5
Edit Copy Delete	10003	104	Stephen Timoshenko	Strength Of Materials Part 1 :Elementary Theory an...	35	35	2020-08-30	7
Edit Copy Delete	10004	105	Claus Borgnakke	Thermodynamics	25	23	2018-08-20	10
Edit Copy Delete	10005	104	Richard Feynman	The Feynman Lectures on Physics: Mainly Mechanics,...	10	9	2015-10-02	19

We can see that available copies for thermodynamics book is : 23 as it is being issued to 2 users . Suppose Member ID 2 wants to return this book as it may not be useful for him. As a first step, updateFine is calculated which indicates this particular user does not have any fine for this book . Hence this book can be sent back to the library immediately.

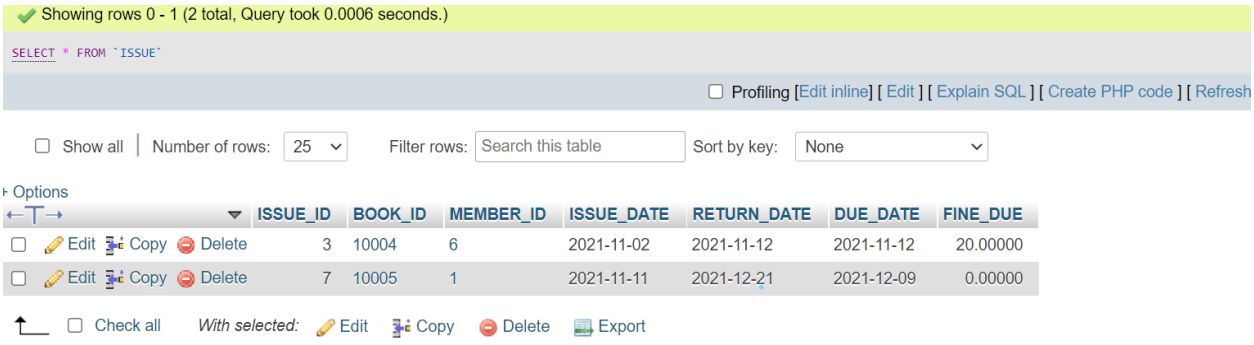
Procedure call ReturnBook

set @p0='Thermodynamics';
set @p1 = 2;
CALL `ReturnBook`(@p0,@p1);

Snapshot of the returnBook execution:

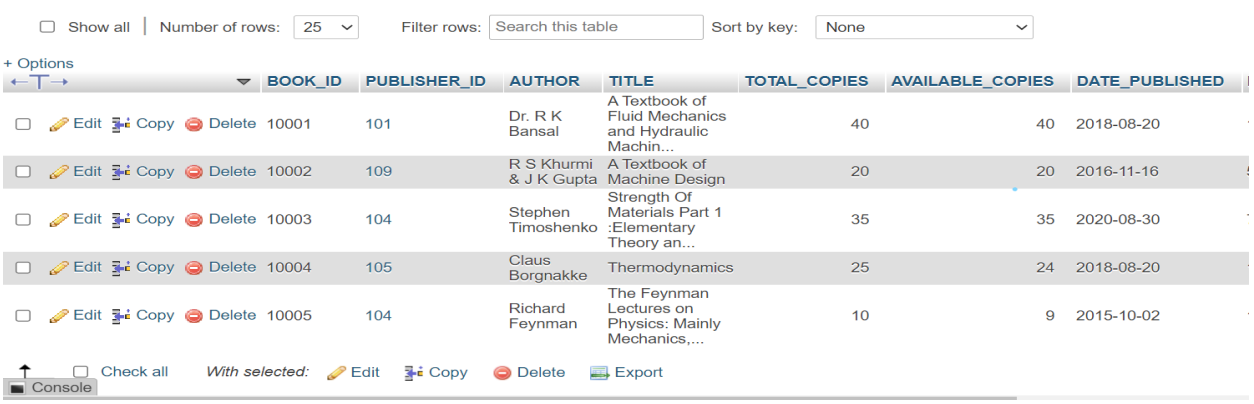


Snapshot of Issue Table after return :



Snapshot of Books Table after return :

We can see that the number of available copies incremented to 24 from 23 .

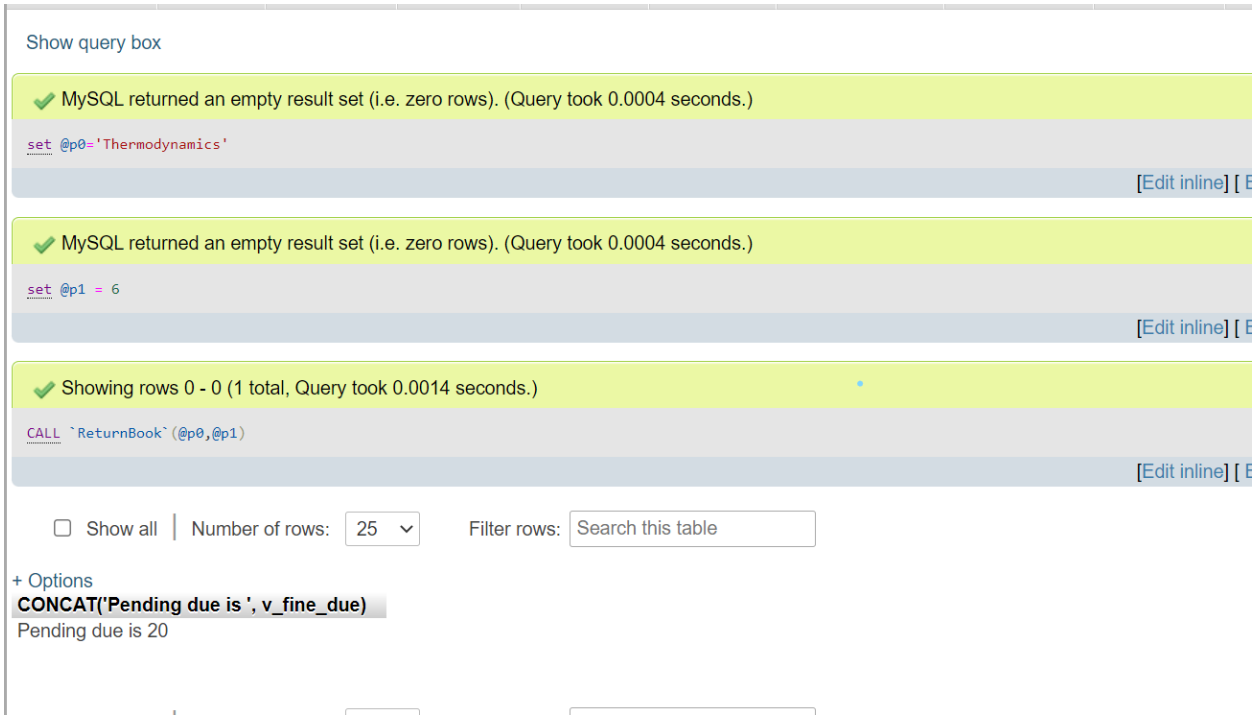


Case 2: If book returned beyond due date

Find out fine_due and intimate user

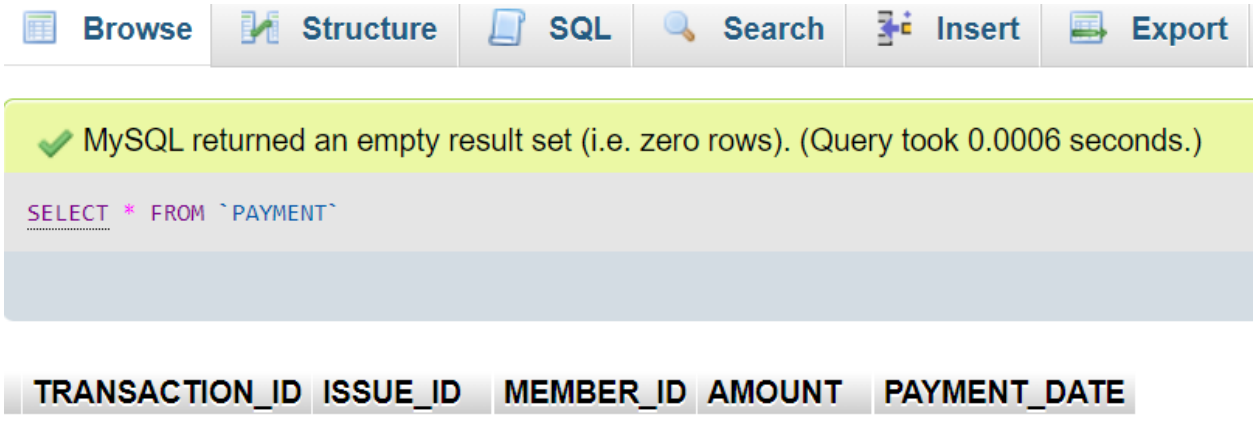
```
set @p0='Thermodynamics';
set @p1 = 6;
CALL `ReturnBook`(@p0,@p1);
```

And we can clearly see from the below output that if fine > 0, amount is being displayed in the below output.



Entry in the Payment table to be created with the amount passed for that issue.

Snapshot of payment table (before paying fine) :



TRANSACTION_ID	ISSUE_ID	MEMBER_ID	AMOUNT	PAYMENT_DATE

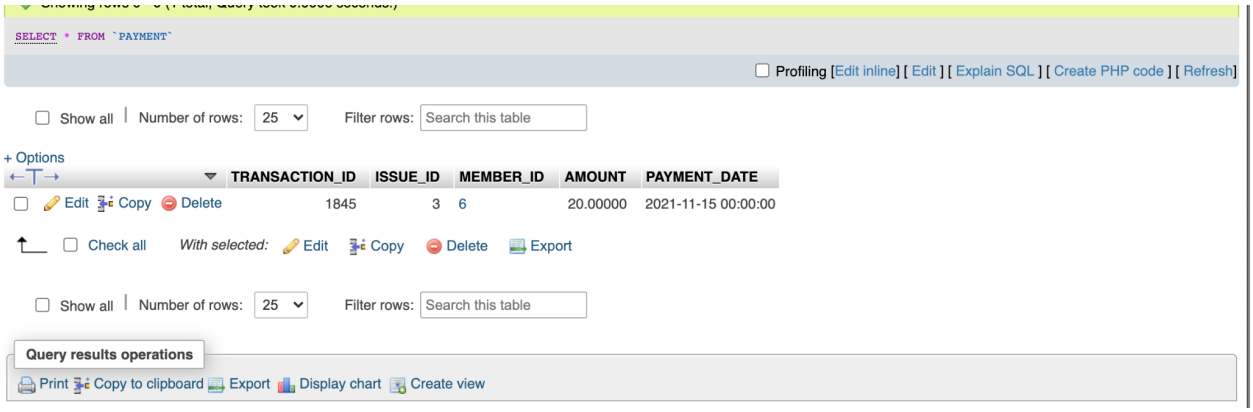
PayFine Procedure Creation

```
BEGIN
   INSERT INTO `PAYMENT` (`TRANSACTION_ID`, `ISSUE_ID`, `MEMBER_ID`,
`AMOUNT`, `PAYMENT_DATE`) VALUES (CONCAT(FLOOR(RAND()*12345),
issueID, memID, amount, CURRENT_DATE);
   UPDATE ISSUE SET ISSUE.FINE_DUE = 0 WHERE ISSUE.ISSUE_ID=issueID;
   SELECT * from ISSUE WHERE ISSUE.ENTITY_STATUS_ID!=999;
   SELECT * from PAYMENT;
END
```

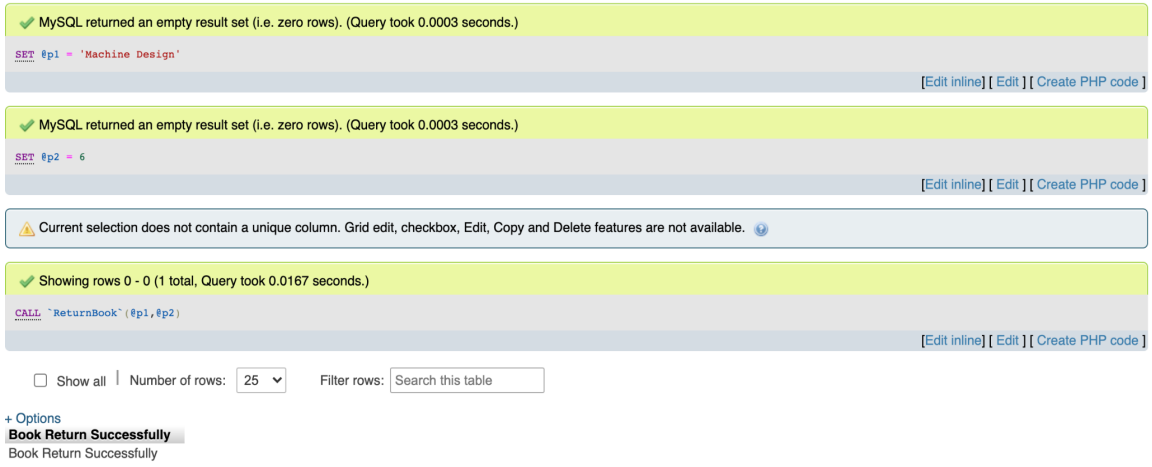
PayFine Procedure Execution

```
set @p0 =3;
set @p1 = 6;
set @p2 = 20;
CALL `PayFine`(@p0,@p1,@p2);
```

Snapshot of payments table after fine is paid of for particular issue ID



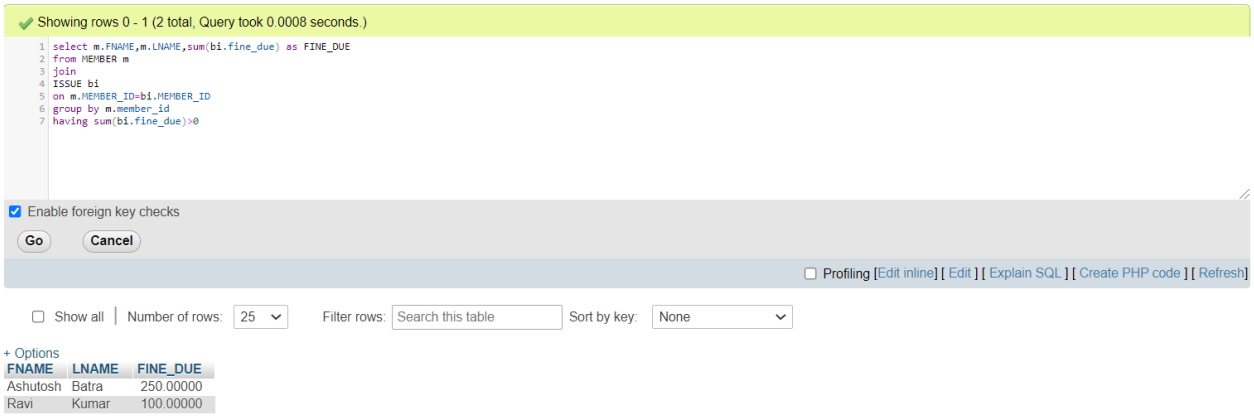
Call again returnBook procedure with bookname and member ID (explained already in case 1)



f) Few more bookkeeping Activities (Librarian)

Fine Details

```
select m.FNAME,m.LNAME,sum(bi.fine_due) as FINE_DUE
from MEMBER m
join
ISSUE bi
on m.MEMBER_ID=bi.MEMBER_ID
group by m.member_id
having sum(bi.fine_due)>0
```

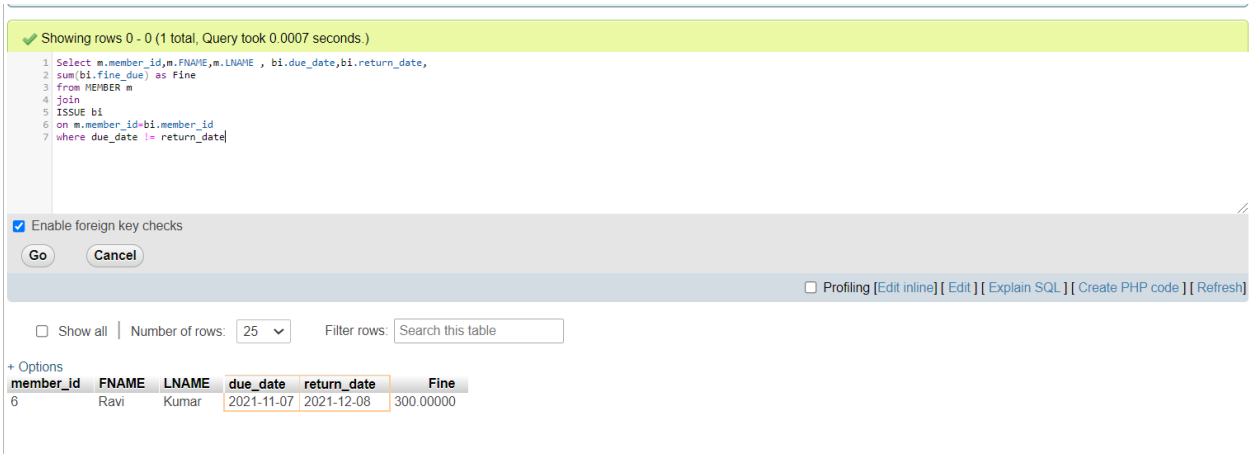


Update fine based on the return date and current date

Already explained in the previous section, the updateFine method does the same.

Members who have not returned books before due date

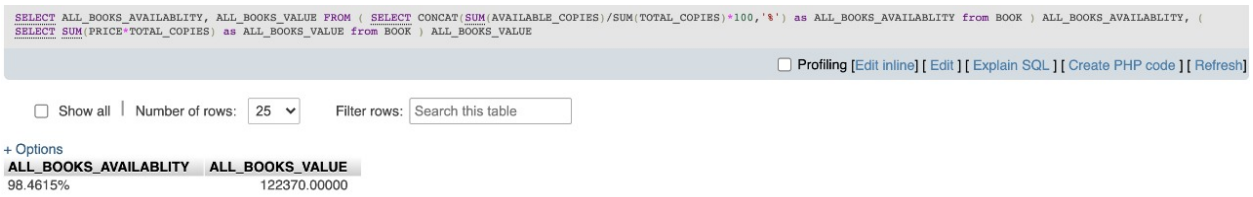
```
Select m.member_id,m.FNAME,m.LNAME , bi.due_date,bi.return_date,
sum(bi.fine_due) as Fine
from MEMBER m
join
ISSUE bi
on m.member_id=bi.member_id
where due_date != return_date
```



Generate fine payment history to indicate all previous defaulter cases .

Generate statistics about the books

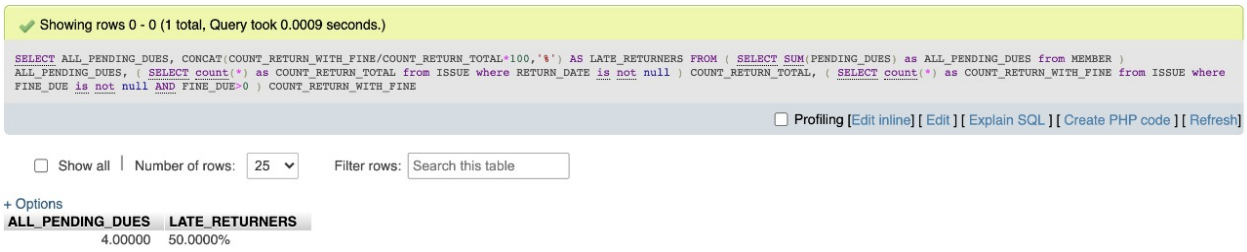
```
SELECT ALL_BOOKS_AVAILABLITY, ALL_BOOKS_VALUE
FROM
(   SELECT CONCAT(SUM(AVAILABLE_COPIES)/SUM(TOTAL_COPIES)*100,'%') as
ALL_BOOKS_AVAILABLITY from BOOK
) ALL_BOOKS_AVAILABLITY,
(   SELECT SUM(PRICE*TOTAL_COPIES) as ALL_BOOKS_VALUE from BOOK
) ALL_BOOKS_VALUE;
```



Generate statistics about the members

```
SELECT ALL_PENDING_DUES,
CONCAT(COUNT_RETURN_WITH_FINE/COUNT_RETURN_TOTAL*100,'%') AS
LATE_RETURNERS
FROM
(   SELECT SUM(PENDING_DUES) as ALL_PENDING_DUES from MEMBER
) ALL_PENDING_DUES,
(   SELECT count(*) as COUNT_RETURN_TOTAL from ISSUE where RETURN_DATE is
not null AND ISSUE.ENTITY_STATUS_ID!=999
```

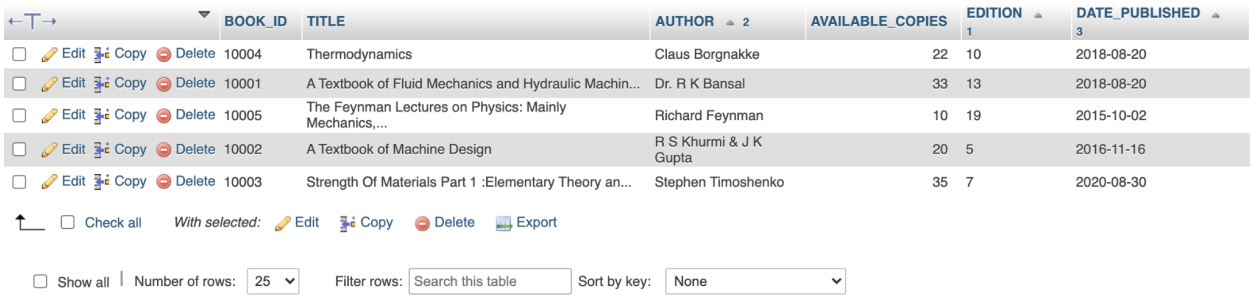
) COUNT_RETURN_TOTAL,
(SELECT count(*) as COUNT_RETURN_WITH_FINE from ISSUE where FINE_DUE is not null AND FINE_DUE>0 AND ISSUE.ENTITY_STATUS_ID!=999
) COUNT_RETURN_WITH_FINE;



g) General queries by student

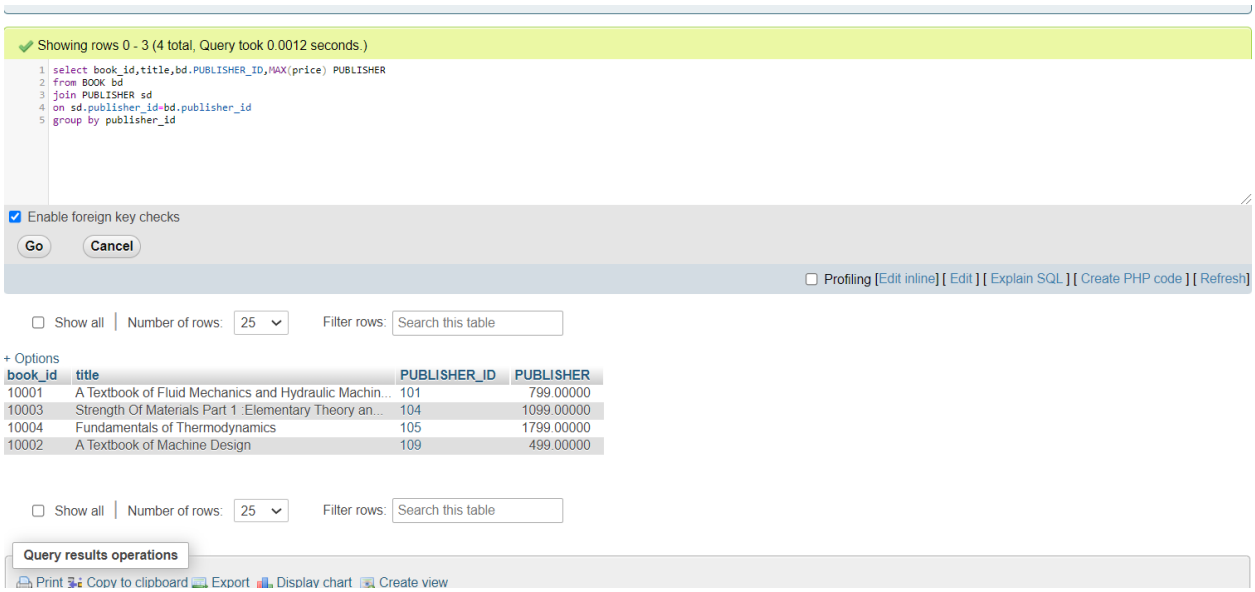
View book inventory (generate library report)

```
select BOOK_ID,TITLE,
AUTHOR,AVAILABLE_COPIES,EDITION,DATE_PUBLISHED
from BOOK
order by EDITION,AUTHOR,DATE_PUBLISHED ;
```



Get Price details for books along with publisher name:

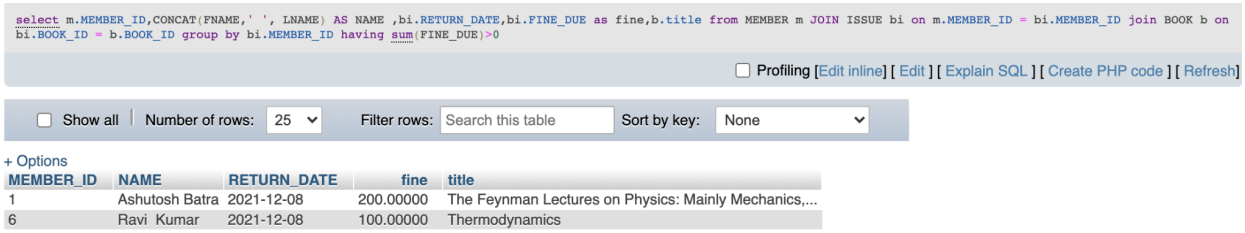
```
select book_id,title,bd.PUBLISHER_ID,MAX(price) as Price , sd.PUBLISHER_NAME
        PUBLISHER
from BOOK bd
join PUBLISHER sd
on sd.publisher_id=bd.publisher_id
group by publisher_id
```



Generate full student profile report

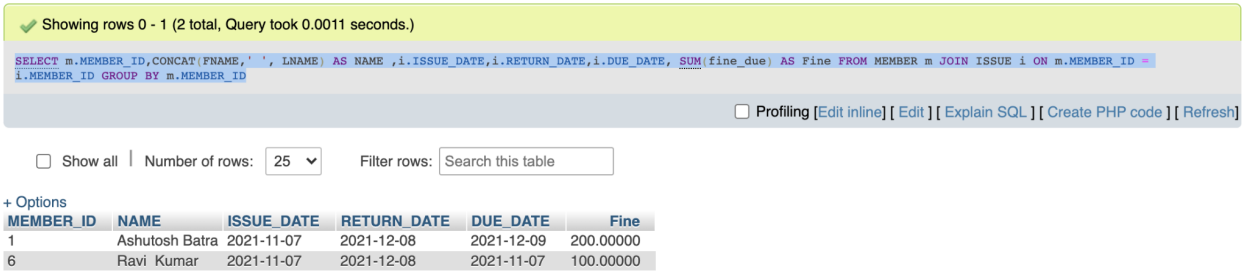
- **Student details**
- **Issued books along with return date and fine details**

```
select m.MEMBER_ID,CONCAT(FNAME,' ', LNAME) AS NAME
,bi.RETURN_DATE,bi.FINE_DUE as fine,b.title from MEMBER m JOIN ISSUE bi on
m.MEMBER_ID = bi.MEMBER_ID AND bi.ENTITY_STATUS_ID!=999 join BOOK b on
bi.BOOK_ID = b.BOOK_ID group by bi.MEMBER_ID having sum(FINE_DUE)>0
```



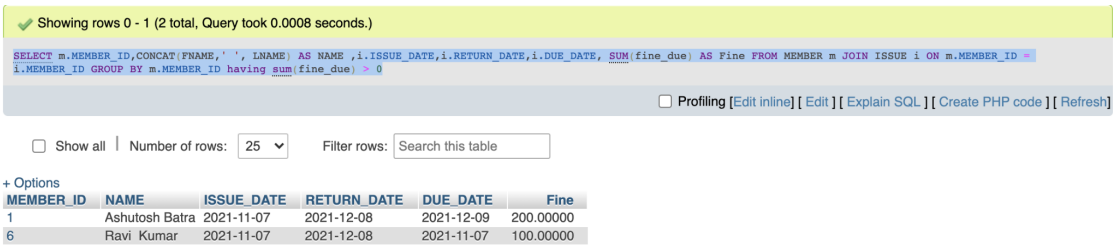
Generate my fine due report (total count of fine for all issued books)

```
SELECT m.MEMBER_ID,CONCAT(FNAME,' ', LNAME) AS NAME
,i.ISSUE_DATE,i.RETURN_DATE,i.DUE_DATE, SUM(fine_due) AS Fine FROM MEMBER m
JOIN ISSUE i ON m.MEMBER_ID = i.MEMBER_ID AND i.ENTITY_STATUS_ID!=999 GROUP
BY m.MEMBER_ID
```



Generate my fine due detailed report (all issued books where pending fine is present)

```
SELECT m.MEMBER_ID,CONCAT(FNAME,' ', LNAME) AS NAME
    ,i.ISSUE_DATE,i.RETURN_DATE,i.DUE_DATE, SUM(fine_due) AS Fine FROM MEMBER
    m JOIN ISSUE i ON m.MEMBER_ID = i.MEMBER_ID AND i.ENTITY_STATUS_ID!=999
    GROUP BY m.MEMBER_ID having sum(fine_due) > 0
```



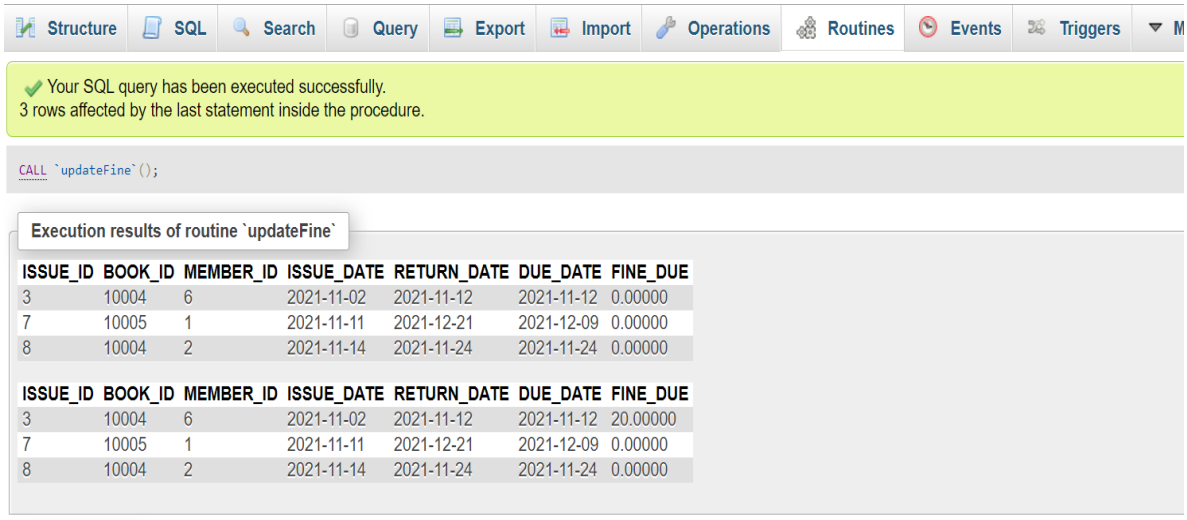
9. Stored Procedures/ Triggers:

i) procedure used by the librarian to update the fine amount based on the due date .

·Procedure name: updateFine

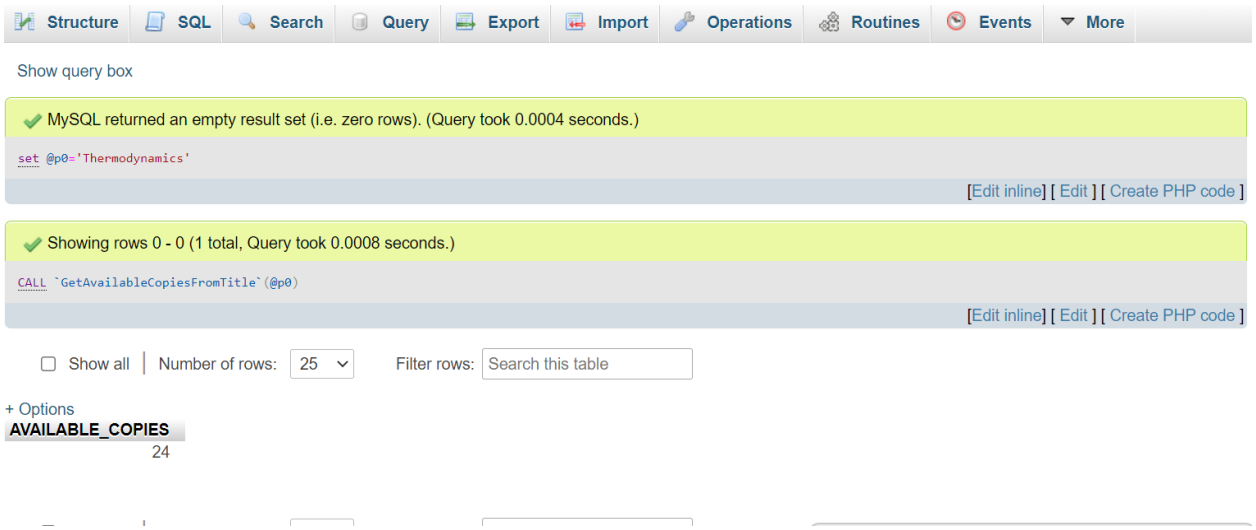
·Input Parameters: void

```
@BookName [nvarchar ]
BEGIN
SELECT * FROM ISSUE;
UPDATE ISSUE
SET ISSUE.FINE_DUE = (CURRENT_DATE - DUE_DATE) *10
WHERE ISSUE.DUE_DATE < CURRENT_DATE;
SELECT * FROM ISSUE;
END
```



ii) Search available book ID based on book name

Procedure name: GetAvailableCopiesFromTitle
Input Parameters: @book_name [Varchar]
BEGIN
SELECT BOOK.AVAILABLE_COPIES FROM BOOK WHERE
BOOK.TITLE LIKE CONCAT('%', BookName , '%');
END



iii) Get book details based on name.

Procedure name: GetBookId
Input Parameters: @book_name [Varchar]
BEGIN
SELECT BOOK.BOOK_ID INTO book_id FROM BOOK WHERE BOOK.TITLE
LIKE CONCAT('%', book_name , '%');

END

Output parameters: @book_id [INT]

This procedure is called from the Book Issue procedure .

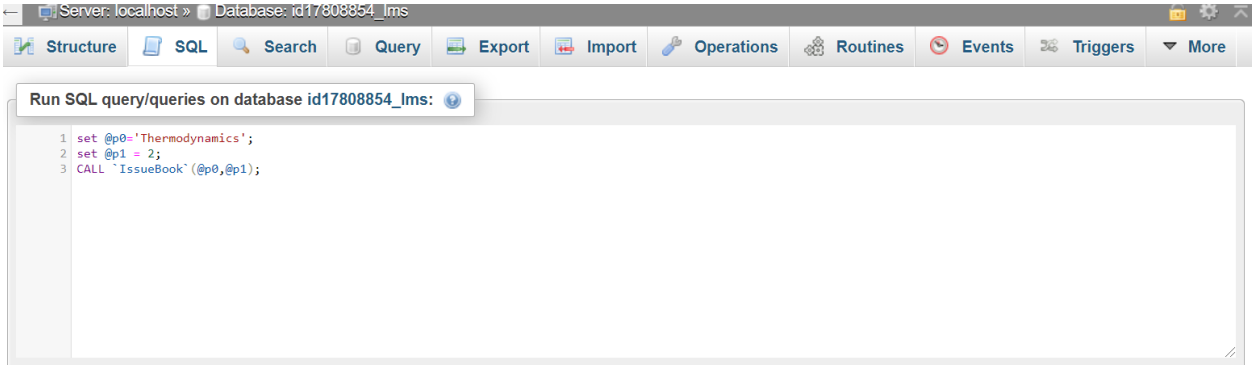
iv) Issue procedure based on book ID.

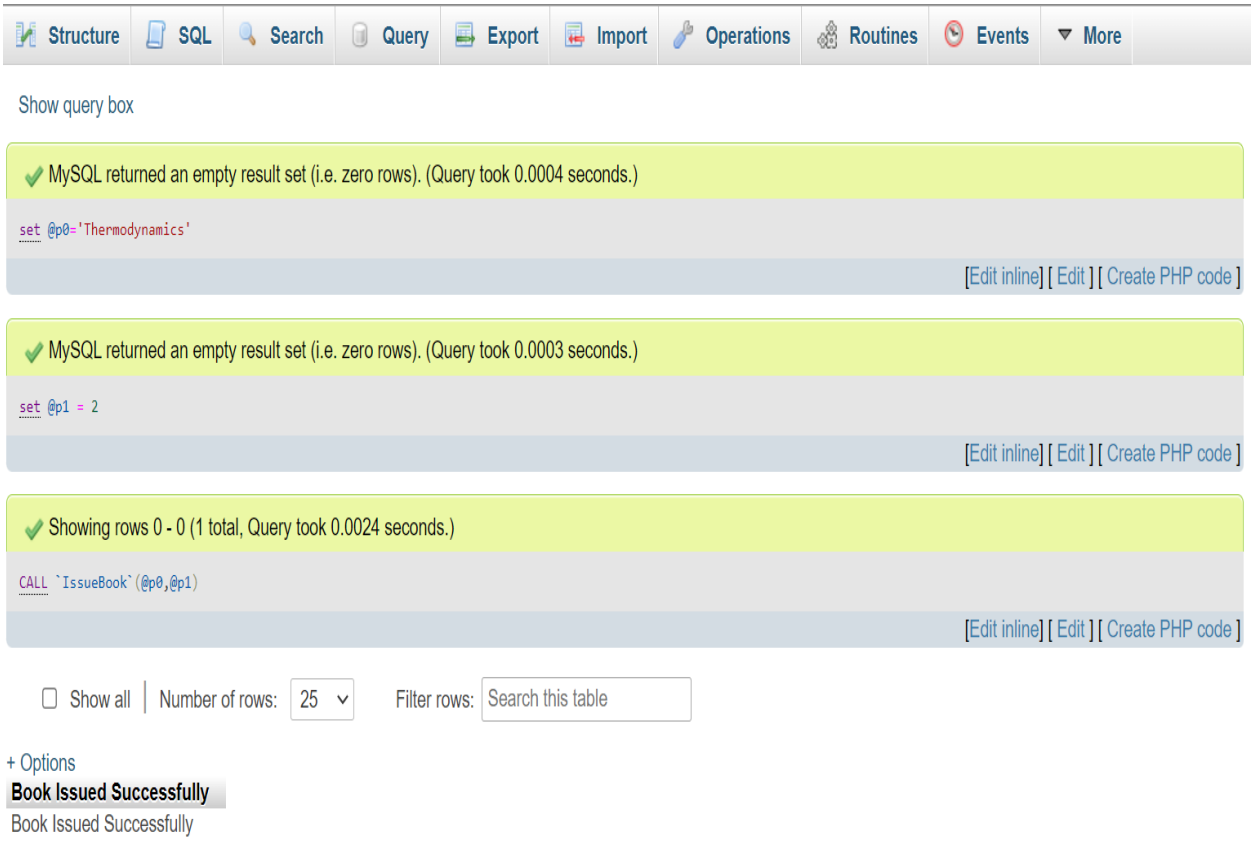
Procedure name: IssueBook

Input Parameters: @BookName [VARCHAR]

@MemberID[INT]

```
BEGIN
    DECLARE max_issueID INT;
    CALL `GetBookId`(BookName, @book_id);
    SELECT MAX(ISSUE.ISSUE_ID) INTO max_issueID FROM ISSUE WHERE
ISSUE.ENTITY_STATUS_ID!=999;
     INSERT INTO `ISSUE` (`ISSUE_ID`, `BOOK_ID`, `MEMBER_ID`, `ISSUE_DATE`,
`RETURN_DATE`, `DUE_DATE`, `FINE_DUE`) VALUES (max_issueID+1, @book_id,
MemberID, CURRENT_DATE, CURRENT_DATE+10, CURRENT_DATE+10, '0.00000');
    UPDATE BOOK SET BOOK.AVAILABLE_COPIES = BOOK.AVAILABLE_COPIES -
1 WHERE BOOK.BOOK_ID = @book_id;
     SELECT "Book Issued Successfully";
END
```





v) Return book based on book ID. Thus, the fine will be calculated based on return date and Issue date.

Procedure name: ReturnBook

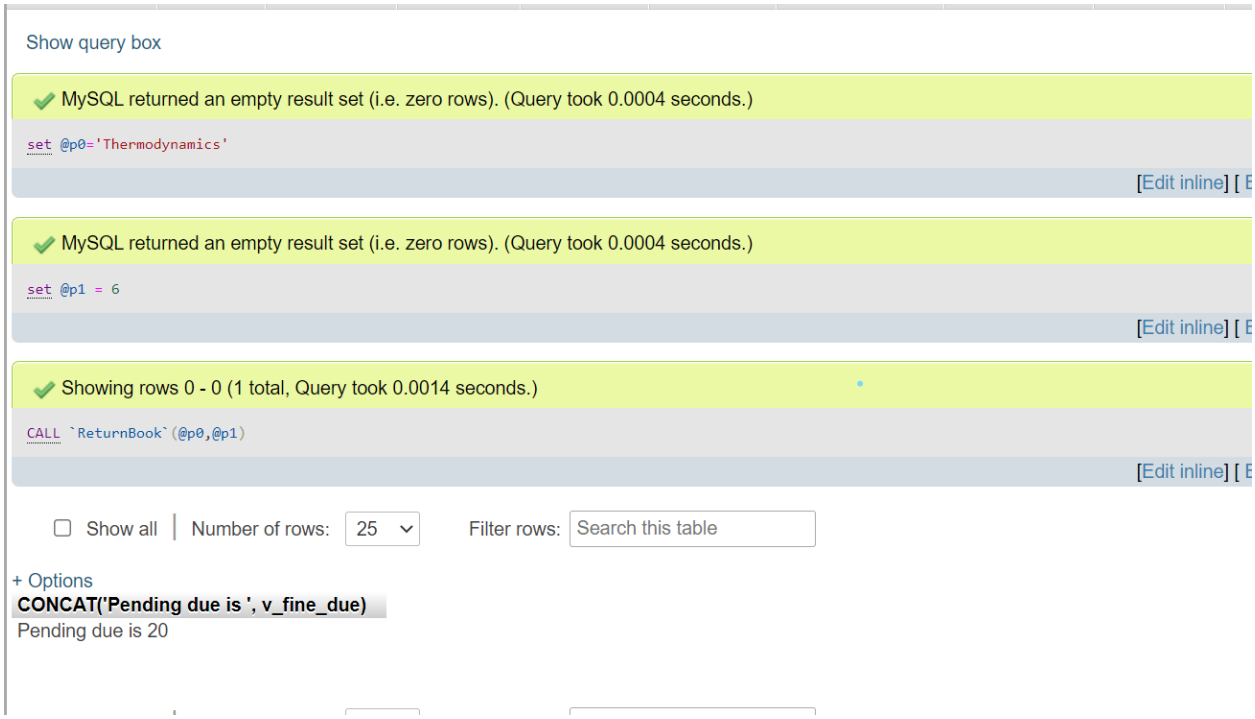
Input Parameters:

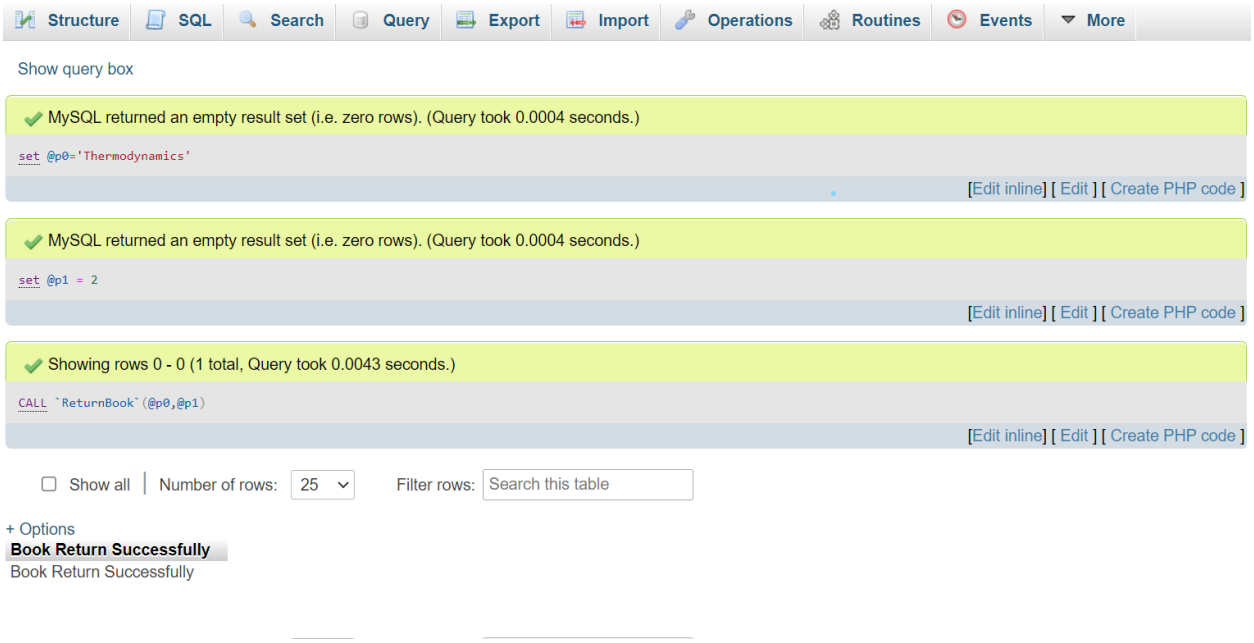
@book_name [INT]

@member_id[INT]

```
BEGIN
    DECLARE v_fine_due INT;
    CALL `GetBookId`(book_name, @book_id);
    SELECT ISSUE.FINE_DUE INTO v_fine_due FROM ISSUE WHERE
ISSUE.BOOK_ID=@book_id AND ISSUE.MEMBER_ID = member_id AND
ISSUE.ENTITY_STATUS_ID!=999;
    IF (v_fine_due != 0) THEN
   SELECT CONCAT('Pending due is ', v_fine_due);
   ELSE
        DELETE FROM ISSUE WHERE ISSUE.MEMBER_ID = member_id AND
   ISSUE.BOOK_ID = @book_id;
    UPDATE BOOK SET BOOK.AVAILABLE_COPIES = BOOK.AVAILABLE_COPIES +
   1 WHERE BOOK.BOOK_ID = @book_id;
```

```
    SELECT "Book Return Successfully";
  END IF;
END
```





vi) Pay fine based on book ID.

Procedure name: PayFine

Input Parameters:

@issueID [INT]

@memID[INT]
@amount[INT]

```
BEGIN
  INSERT INTO `PAYMENT` (`TRANSACTION_ID`, `ISSUE_ID`, `MEMBER_ID`,
`AMOUNT`, `PAYMENT_DATE`) VALUES (FLOOR(RAND()*12345), issueID, memID,
amount, CURRENT_DATE);
  UPDATE ISSUE SET ISSUE.FINE_DUE = 0 WHERE ISSUE.ISSUE_ID=issueID;
  SELECT * from ISSUE;
  SELECT * from PAYMENT;
END
```

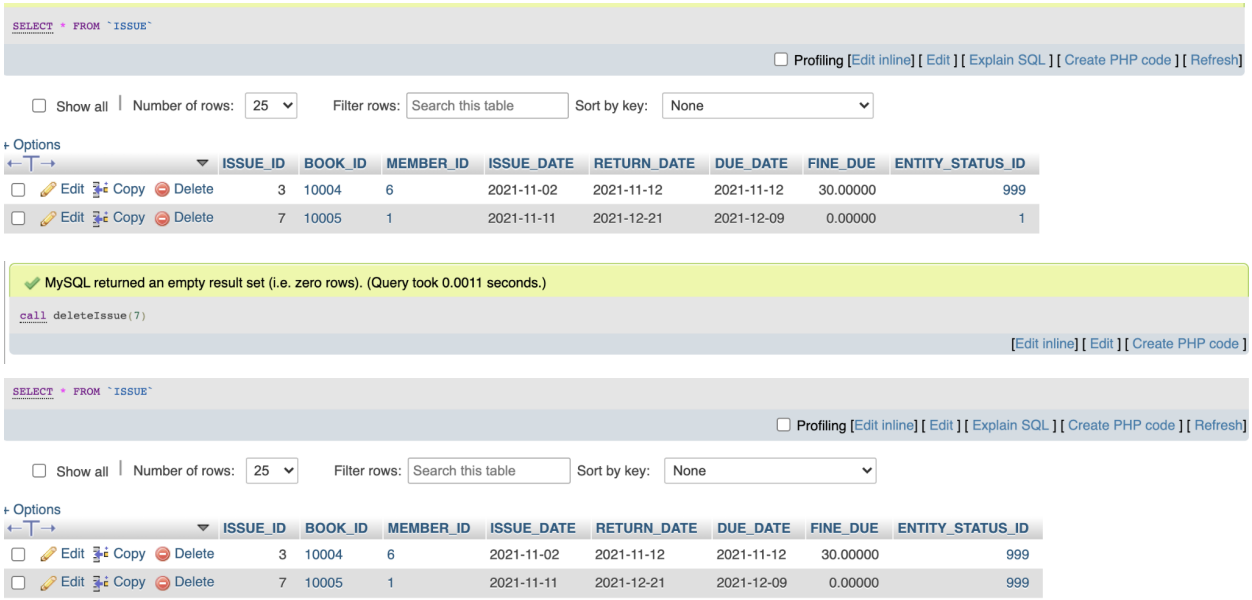
vii) Delete(Soft delete) issue

Procedure Name: deleteIssue ()

Input Parameters:

@issue_id_to_delete[NUMBER(38,0)]

```
BEGIN
    UPDATE ISSUE
    SET ISSUE.ENTITY_STATUS_ID = (SELECT ENTITY_STATUS_ID from
ENTITYSTATUS where STATUS = 'Deleted') where ISSUE.ISSUE_ID =
issue_id_to_delete;
 END
```



10. DB connection steps

Project Name: BITSLibrary

Login details :
DataBase Direct Login URL: https://databases-auth.000webhost.com/index.php
DB NAME: id17808854_lms
DB USER: id17808854_student
DB PASSWORD: Bits@lms2021

Steps to access SLMS DB - Student Library Management System
1.Login
2.Hover on “bitslibrary” project and click Manage Website button
3.Select “Tools” menu from the left side panel of “bitslibrary” manage page
4.Select “Datanase Manager” from the left side panel of “bitslibrary” manage page
5.Click on “Manage” button, present in the middle window and choose “PhpMyAdmin”
6.Elaboarate “id17808854_slms” you can see the created relations under the SLMS database.