PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

B.TECH. (ECE)

IV SEMESTER

UE20CS301 – Database Management Systems (Minors)

Mini-Project Report

on

Title

SUBMITTED BY

Name	SRN
ANUMULA BALAJI	PES1UG21EC052
S G S S DHANVI	PES1UG21EC104
GAUTHAM MENON	PES1UG21EC101

January – May 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BENGALURU – 560100, KARNATAKA, INDIA

CONTENTS

INTRODUCTION

A library is a collection of organized information and resources which is made accessible to a well-defined community for borrowing or reference's sake. The collection of the resources and information are provided in digital or physical format in either a building/room or in a virtual space or even both. Library resources and collections may include newspapers, books, films, prints, maps, CDs, tapes, videotapes, microform, database etc.

The main aim of this system is to develop a new programmed system that will conveying ever lasting solution to the manual base operations and to make available a channel through which staff can maintain the record easily and customers can access the information about the library at whatever place they might find themselves.

The Library Management System allows the user to store the book details and the customer details. The system is strong enough to withstand regressive yearly operations under conditions where the database is maintained and cleared over a certain time span. The implementation of the system in the organization will considerably reduce data entry time and provide readily calculated reports.

OBJECTIVE: It keeps track of all the information about the books in the library, their cost, status, and total number of books available in the library. The user will find it easy in this automated system rather than using the manual writing system. The system contains a database where all the information will be

DATA TYPES

1. Integer: one optional sign character (+ or -) followed by at least one digit (0-9). Leading and trailing blanks are ignored. No other character is allowed.

2. Varchar: It is used to store alpha numeric characters. In this data type we can set the maximum number of characters up to 8000 ranges by defaults SQL server will set the size to 50 characters range.

3. Date: The DATE data type accepts date values. No parameters are required when declaring a DATE data type. Date values should be specified in the form: YYYY-MM-DD. However, Point Base will also accept single digits entries for month and day values.

4. Time: The TIME data type accepts time values. No parameters are required when declaring a TIME data type. Date values should be specified in the form: HH:MM: SS. An optional fractional value can be used to represent nanoseconds.

DATA REQUIREMENTS

ENTITIES

- BRANCH
- EMPLOYEE
- CUSTOMER
- ISSUE STATUS
- RETURN STATUS
- BOOKS

ATTRIBUTES

- BRANCH
  - Manager_id
  - Branch_id
  - Address
    - Branch_h_no
    - Street
    - City
    - State
    - Zipcode
  - Contact No
- CUSTOMER
  - Customer_id
  - Books_issued
  - Name
  - Address
  - Registration_date

- ISSUE STATUS
  - Issue_book_name
  - Issue_id
  - Issue_date
  - ISBN
  - Customer_id

- Employee
  - Employee_id
  - Branch_id
  - Name
  - Position
  - Phone_No
- RETURN STATUS
  - Return_id
  - Return_date
  - Customer_id
  - Return_book_name
  - ISBN
- BOOKS
  - ISBN
  - Title
  - Category
  - Rental_price
  - Author
  - Publisher
  - Status

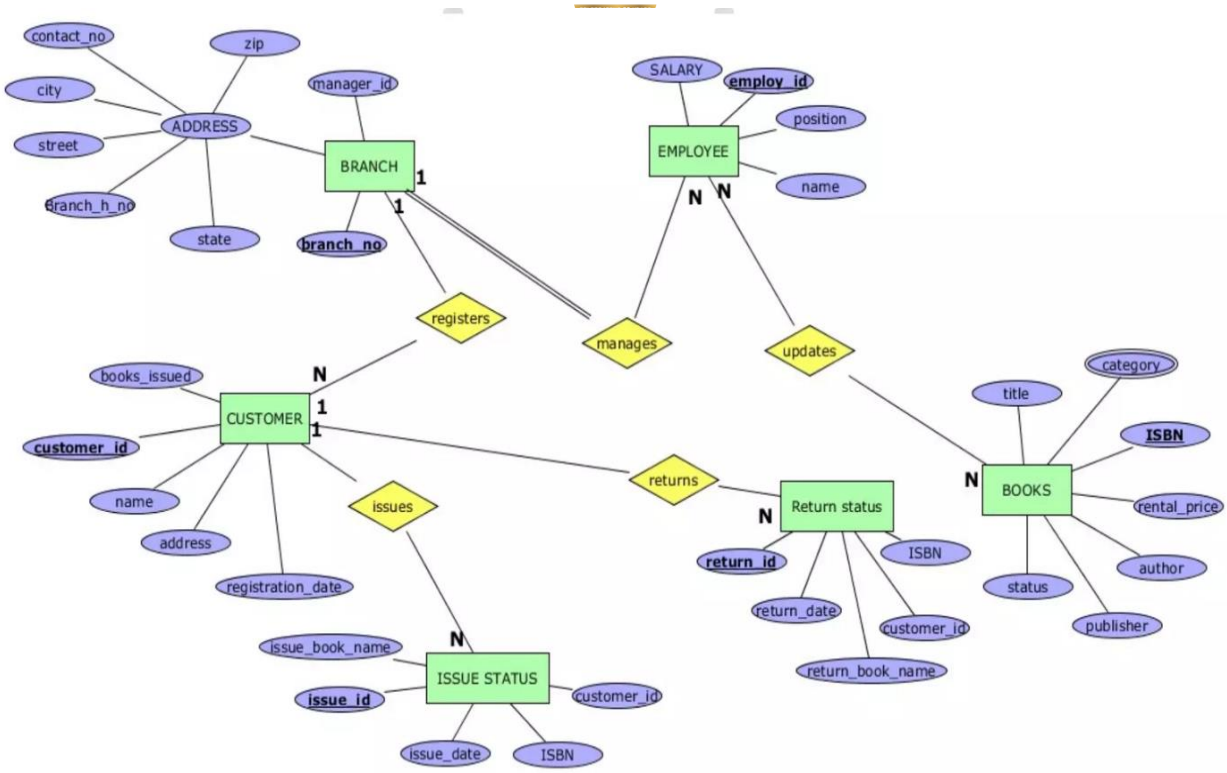
RELATIONSHIPS – CARDINALITY

MANAGER manages the BRANCH	(1-N)
CUSTOMER registers in the respective BRANCH	(N-1)
CUSTOMER issues the BOOKS	(1-N)
CUSTOMER returns the BOOKS	(N-1)
EMPLOYEE updates the BOOKS	(N-N)

ENTITY RELATIONSHIP DIAGRAM (ER DIAGRAM)

Entity Relationship Diagram is used in modern database software engineering to illustrate logical structure of database. It is a relational schema database modelling method used to Model a system and approach. This approach is commonly used in database design. The diagram created using this method is called ER-diagram.

The ER-diagram depicts the various relationships among entities, considering each object as entity. Entity is represented as rectangular shape and relationship represented as diamond shape. It depicts the relationship between data objects. The ER-diagram is the notation used to conduct data modelling activity.

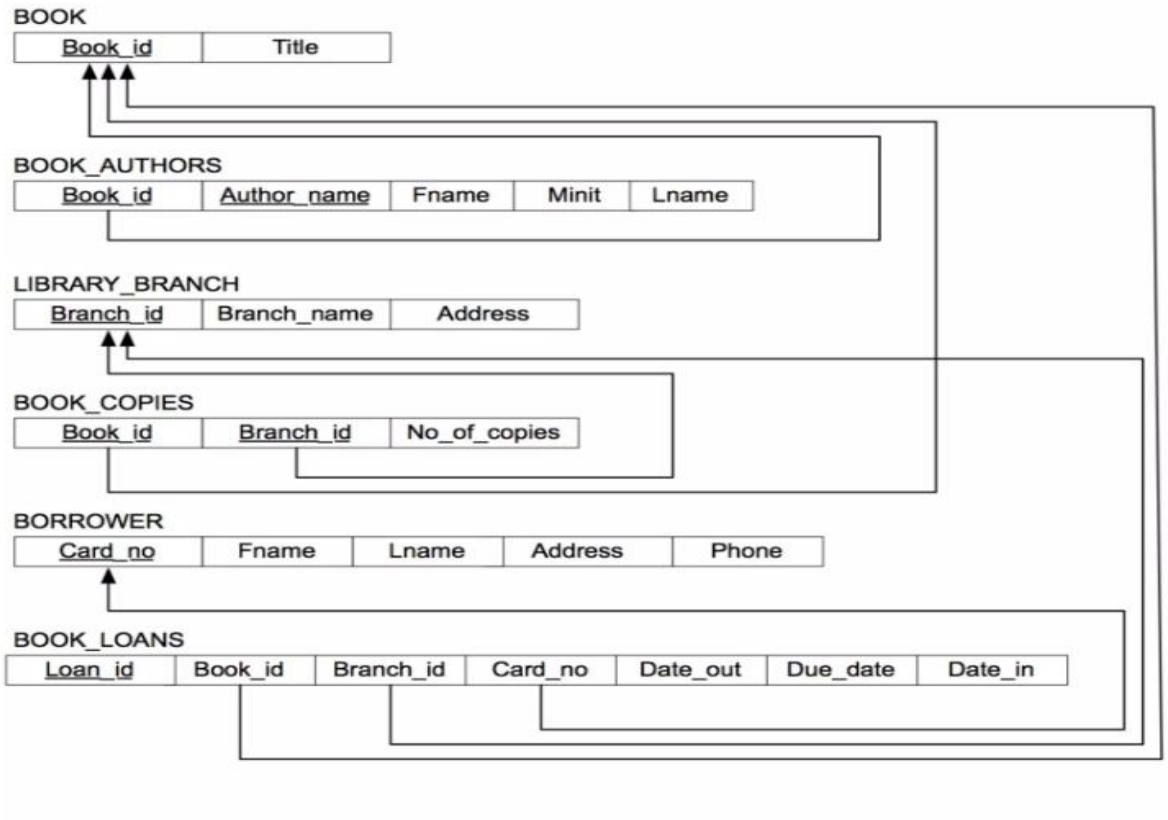


SCHEMA DIAGRAM

A schema is the structure behind data organization. It is a visual representation of how different table relationships enable the schema's underlying mission business rules for which the database is created. Database schema defines its entities and the relationship among them.

It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

Schema diagrams have an important function because they force database developers to transpose ideas to paper. This provides an overview of the entire database, while facilitating future database administrator work.



CREATING DATABASE USING MySQL

```
CREATE TABLE Branch (
  BranchID INT NOT NULL AUTO_INCREMENT,
  Name VARCHAR(255) NOT NULL,
  Address VARCHAR(255) NOT NULL,
  Phone VARCHAR(20) NOT NULL,
  PRIMARY KEY (BranchID)
);

CREATE TABLE Employee (
  EmployeeID INT NOT NULL AUTO_INCREMENT,
  Name VARCHAR(255) NOT NULL,
  Address VARCHAR(255) NOT NULL,
```

```
  Phone VARCHAR(20) NOT NULL,
  BranchID INT NOT NULL,
  PRIMARY KEY (EmployeeID),
  FOREIGN KEY (BranchID) REFERENCES Branch(BranchID)
);

CREATE TABLE Customer (
  CustomerID INT NOT NULL AUTO_INCREMENT,
  Name VARCHAR(255) NOT NULL,
  Address VARCHAR(255) NOT NULL,
  Phone VARCHAR(20) NOT NULL,
  PRIMARY KEY (CustomerID)
);

CREATE TABLE Issue_Status (
  IssueStatusID INT NOT NULL AUTO_INCREMENT,
  Name VARCHAR(255) NOT NULL,
  PRIMARY KEY (IssueStatusID)
);

CREATE TABLE Return_Status (
  ReturnStatusID INT NOT NULL AUTO_INCREMENT,
  Name VARCHAR(255) NOT NULL,
  PRIMARY KEY (ReturnStatusID)
);

CREATE TABLE Books (
  BookID INT NOT NULL AUTO_INCREMENT,
  Title VARCHAR(255) NOT NULL,
  Author VARCHAR(255) NOT NULL,
  Publisher VARCHAR(255) NOT NULL,
  ISBN VARCHAR(20) NOT NULL,
  BranchID INT NOT NULL,
  PRIMARY KEY (BookID),
  FOREIGN KEY (BranchID) REFERENCES Branch(BranchID)
);

CREATE TABLE Issue (
  IssueID INT NOT NULL AUTO_INCREMENT,
```

```
  BookID INT NOT NULL,
  CustomerID INT NOT NULL,
  EmployeeID INT NOT NULL,
  IssueDate DATE NOT NULL,
  DueDate DATE NOT NULL,
  ReturnDate DATE,
  IssueStatusID INT NOT NULL,
  PRIMARY KEY (IssueID),
  FOREIGN KEY (BookID) REFERENCES Books(BookID),
  FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
  FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
  FOREIGN KEY (IssueStatusID) REFERENCES Issue_Status(IssueStatusID)
);

CREATE TABLE Return_ (
  ReturnID INT NOT NULL AUTO_INCREMENT,
  IssueID INT NOT NULL,
  ReturnDate DATE NOT NULL,
  ReturnStatusID INT NOT NULL,
  PRIMARY KEY (ReturnID),
  FOREIGN KEY (IssueID) REFERENCES Issue(IssueID),
  FOREIGN KEY (ReturnStatusID) REFERENCES
Return_Status(ReturnStatusID)
);
```

```
+----------------+
| Tables_in_proj |
+----------------+
| books          |
| branch         |
| customer       |
| employee       |
| issue          |
| issue_status   |
| return_        |
| return_status  |
+----------------+
```

```
MariaDB [proj]> desc books;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| BookID    | int(11)     | NO   | PRI | NULL    | auto_increment |
| Title     | varchar(255)| NO   |     | NULL    |                |
| Author    | varchar(255)| NO   |     | NULL    |                |
| Publisher | varchar(255)| NO   |     | NULL    |                |
| ISBN      | varchar(20) | NO   |     | NULL    |                |
| BranchID  | int(11)     | NO   | MUL | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
6 rows in set (0.010 sec)
```

```
MariaDB [proj]> desc branch;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| BranchID | int(11)     | NO   | PRI | NULL    | auto_increment |
| Name     | varchar(255)| NO   |     | NULL    |                |
| Address  | varchar(255)| NO   |     | NULL    |                |
| Phone    | varchar(20) | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.009 sec)
```

```
MariaDB [proj]> desc customer;
+------------+-------------+------+-----+---------+----------------
| Field      | Type        | Null | Key | Default | Extra
+------------+-------------+------+-----+---------+----------------
| CustomerID | int(11)     | NO   | PRI | NULL    | auto_increment
| Name       | varchar(255)| NO   |     | NULL    |
| Address    | varchar(255)| NO   |     | NULL    |
| Phone      | varchar(20) | NO   |     | NULL    |
+------------+-------------+------+-----+---------+----------------
```


```
MariaDB [proj]> desc employee;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| EmployeeID | int(11)      | NO   | PRI | NULL    | auto_increment |
| Name       | varchar(255) | NO   |     | NULL    |                |
| Address    | varchar(255) | NO   |     | NULL    |                |
| Phone      | varchar(20)  | NO   |     | NULL    |                |
| BranchID   | int(11)      | NO   | MUL | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
5 rows in set (0.009 sec)
```

```
MariaDB [proj]> desc issue_status;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| IssueStatusID | int(11)      | NO   | PRI | NULL    | auto_increment |
| Name          | varchar(255) | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
2 rows in set (0.008 sec)
```

```
MariaDB [proj]> desc issue;
+---------------+---------+------+-----+---------+----------------+
| Field         | Type    | Null | Key | Default | Extra          |
+---------------+---------+------+-----+---------+----------------+
| IssueID       | int(11) | NO   | PRI | NULL    | auto_increment |
| BookID        | int(11) | NO   | MUL | NULL    |                |
| CustomerID    | int(11) | NO   | MUL | NULL    |                |
| EmployeeID    | int(11) | NO   | MUL | NULL    |                |
| IssueDate     | date    | NO   |     | NULL    |                |
| DueDate       | date    | NO   |     | NULL    |                |
| ReturnDate    | date    | YES  |     | NULL    |                |
| IssueStatusID | int(11) | NO   | MUL | NULL    |                |
+---------------+---------+------+-----+---------+----------------+
8 rows in set (0.011 sec)
```

```
MariaDB [proj]> desc return_status;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| ReturnStatusID | int(11)      | NO   | PRI | NULL    | auto_increment |
| Name           | varchar(255) | NO   |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
2 rows in set (0.008 sec)
```

```
MariaDB [proj]> desc return_;
+----------------+---------+------+-----+---------+----------------+
| Field          | Type    | Null | Key | Default | Extra          |
+----------------+---------+------+-----+---------+----------------+
| ReturnID       | int(11) | NO   | PRI | NULL    | auto_increment |
| IssueID        | int(11) | NO   | MUL | NULL    |                |
| ReturnDate     | date    | NO   |     | NULL    |                |
| ReturnStatusID | int(11) | NO   | MUL | NULL    |                |
+----------------+---------+------+-----+---------+----------------+
4 rows in set (0.010 sec)
```

```
INSERT INTO Branch (Name, Address, Phone) VALUES
  ('Main Library', '123 Main St, Anytown, USA', '555-1234'),
  ('North Branch', '456 North St, Anytown, USA', '555-5678');

INSERT INTO Employee (Name, Address, Phone, BranchID) VALUES
  ('John Smith', '456 South St, Anytown, USA', '555-2345', 1),
  ('Jane Doe', '789 East St, Anytown, USA', '555-6789', 2);

INSERT INTO Customer (Name, Address, Phone) VALUES
  ('Bob Johnson', '1234 West St, Anytown, USA', '555-3456'),
  ('Sally Jones', '5678 South St, Anytown, USA', '555-7890');

INSERT INTO Issue_Status (Name) VALUES
  ('Issued'),
  ('Overdue'),
  ('Returned');

INSERT INTO Return_Status (Name) VALUES
  ('On Time'),
  ('Late');

INSERT INTO Books (Title, Author, Publisher, ISBN, BranchID) VALUES
  ('The Great Gatsby', 'F. Scott Fitzgerald', 'Scribner', '978-0743273565', 1),
  ('To Kill a Mockingbird', 'Harper Lee', 'J. B. Lippincott & Co.', '978-
0446310789', 2);

INSERT INTO Issue (BookID, CustomerID, EmployeeID, IssueDate,
DueDate, ReturnDate, IssueStatusID) VALUES
  (1, 1, 1, '2023-05-01', '2023-06-01', NULL, 1),
  (2, 2, 2, '2023-05-15', '2023-06-15', NULL, 1);

INSERT INTO Return_ (IssueID, ReturnDate, ReturnStatusID) VALUES
  (1, NULL, NULL),
  (2, NULL, NULL);
```


```
MariaDB [proj]> select *from books;
+--------+---------------------+--------------------+----------------------+---------------+----------+
| BookID | Title               | Author             | Publisher            | ISBN          | BranchID |
+--------+---------------------+--------------------+----------------------+---------------+----------+
|      1 | The Great Gatsby    | F. Scott Fitzgerald | Scribner             | 978-0743273565 |        1 |
|      2 | To Kill a Mockingbird | Harper Lee         | J. B. Lippincott & Co. | 978-0446310789 |        2 |
+--------+---------------------+--------------------+----------------------+---------------+----------+
2 rows in set (0.000 sec)
```

```
MariaDB [proj]> select *from branch;
+----------+--------------+---------------------------+----------+
| BranchID | Name         | Address                   | Phone    |
+----------+--------------+---------------------------+----------+
|        1 | Main Library | 123 Main St, Anytown, USA | 555-1234 |
|        2 | North Branch | 456 North St, Anytown, USA | 555-5678 |
+----------+--------------+---------------------------+----------+
2 rows in set (0.000 sec)
```

```
MariaDB [proj]> select *from customer;
+------------+-------------+---------------------------+----------+
| CustomerID | Name        | Address                   | Phone    |
+------------+-------------+---------------------------+----------+
|          1 | Bob Johnson | 1234 West St, Anytown, USA | 555-3456 |
|          2 | Sally Jones | 5678 South St, Anytown, USA | 555-7890 |
+------------+-------------+---------------------------+----------+
2 rows in set (0.000 sec)
```

```
MariaDB [proj]> select *from employee;
+------------+------------+---------------------------+----------+----------+
| EmployeeID | Name       | Address                   | Phone    | BranchID |
+------------+------------+---------------------------+----------+----------+
|          1 | John Smith | 456 South St, Anytown, USA | 555-2345 |        1 |
|          2 | Jane Doe   | 789 East St, Anytown, USA | 555-6789 |        2 |
+------------+------------+---------------------------+----------+----------+
2 rows in set (0.000 sec)
```

TEST QUERIES:

1. Retrieve the names of all employees and their associated branch names:

```
MariaDB [proj]> select employee.name, branch.name as branchname from employee inner join branch on employee.branchid = branch.branchi
d;
+------------+-------------+
| name       | branchname  |
+------------+-------------+
| John Smith | Main Library |
| Jane Doe   | North Branch |
+------------+-------------+
2 rows in set (0.001 sec)
```

2. Retrieve the names of all customers who have issued a book:

```
MariaDB [proj]> select distinct customer.name from customer inner join issue on customer.customerid = issue.customerid;
+-------------+
| name        |
+-------------+
| Bob Johnson |
| Sally Jones |
+-------------+
2 rows in set (0.001 sec)
```

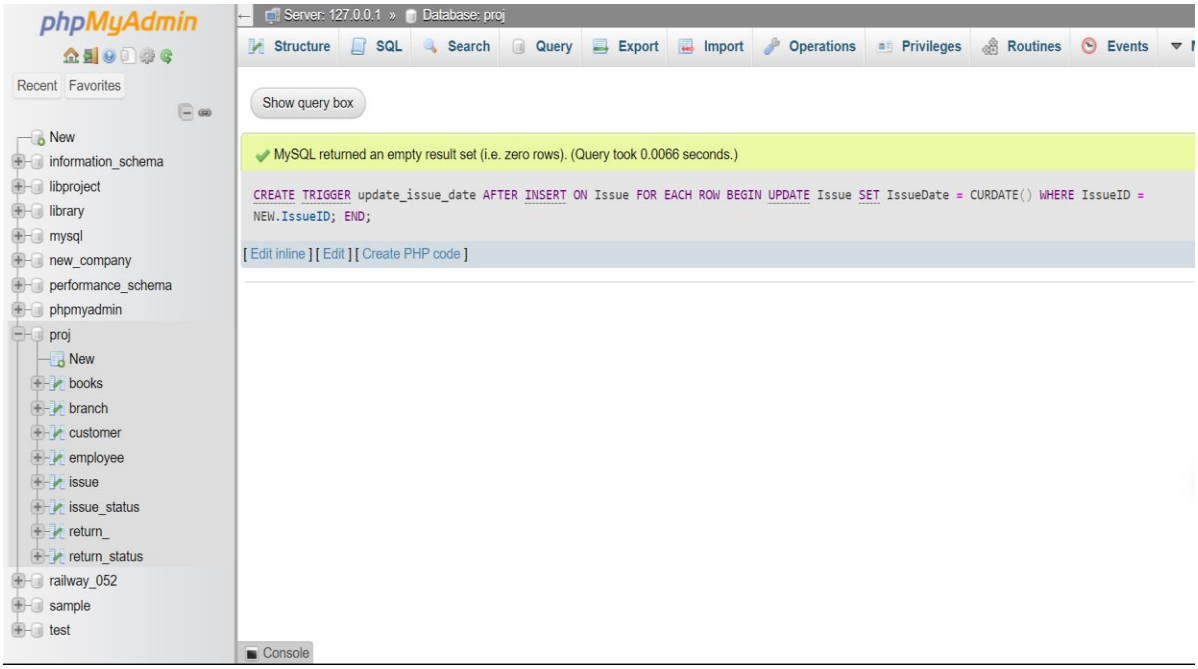
3. Retrieve the titles of all books issued by a specific branch:

```
MariaDB [proj]> select books.title from books inner join issue on books.bookid = issue.bookid where books.branchid = 1;
+-----------------+
| title           |
+-----------------+
| The Great Gatsby |
+-----------------+
1 row in set (0.001 sec)
```

4. Retrieve the number of books issued by each branch:

```
MariaDB [proj]> select branch.name , count(*) as numbooksissued from branch inner join employee on branch.branchid = employee.branchi
d inner join issue on employee.employeeid = issue.employeeid group by branch.name;
+-------------+---------------+
| name        | numbooksissued |
+-------------+---------------+
| Main Library |             1 |
| North Branch |             1 |
+-------------+---------------+
2 rows in set (0.001 sec)
```

5. Trigger to update the `ReturnDate` column in the `Issue` table when a book is returned:



6. stored procedure that retrieves all books that have been issued but not yet returned:

```
DELIMITER $$
CREATE PROCEDURE GetIssuedBooks()
BEGIN
  SELECT Books.Title, Customer.Name AS CustomerName, Employee.Name
AS EmployeeName, Issue.IssueDate, Issue.DueDate
  FROM Books
  INNER JOIN Issue ON Books.BookID = Issue.BookID
  INNER JOIN Customer ON Issue.CustomerID = Customer.CustomerID
  INNER JOIN Employee ON Issue.EmployeeID = Employee.EmployeeID
  WHERE Issue.ReturnDate IS NULL;
END $$
DELIMITER ;
```

```
MariaDB [proj]> call getissuedbooks();
+----------------------+--------------+--------------+------------+------------+
| Title                | CustomerName | EmployeeName | IssueDate  | DueDate    |
+----------------------+--------------+--------------+------------+------------+
| The Great Gatsby     | Bob Johnson  | John Smith   | 2023-05-01 | 2023-06-01 |
| To Kill a Mockingbird | Sally Jones  | Jane Doe     | 2023-05-15 | 2023-06-15 |
+----------------------+--------------+--------------+------------+------------+
2 rows in set (0.001 sec)
```

7. Function to calculate the total number of books issued by an employee:

```
DELIMITER $$
CREATE FUNCTION GetNumBooksIssuedByEmployee(employee_id INT)
RETURNS INT
BEGIN
  DECLARE num_books INT;
  SELECT COUNT(*) INTO num_books FROM Issue WHERE EmployeeID =
employee_id;
  RETURN num_books;
END$$;
DELIMITER ;
```

8. Running a stream lit.

PES UNIVERSITY

```python
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 26 13:16:31 2023

@author: admin
"""

import mysql.connector
import streamlit as st
import pandas as pd

db = mysql.connector.connect(
    host="localhost",
    port="3306",
    user="root",
    password="",
    database="pro"
)

cursor = db.cursor()

'''
#Library managment system
PES1UG21EC101 - Gautham
PES1UG21EC0   - Balaji
PES1UG21EC104 - Dhanvi
'''

st.title("Book Details")

cursor.execute("SELECT * FROM books")
st.dataframe(pd.DataFrame(cursor.fetchall(), columns=( "ISBN", " book_title","category","rental_price"," status"," author","publisher")))

st.subheader("Add a new book")
ISBN = st.number_input("ISBN",key="ISBN_create")
book_title = st.text_input("Title",key="book_title_create")
category = st.text_input("Category",key="category_create")
rental_price = st.number_input("Rental_Price",key="rental_price_create")
status = st.text_input("Status",key="status_create")
author = st.text_input("Author",key="author_create")
publisher=st.text_input("Publisher",key="publisher_create")
if st.button("Add book"):
    cursor.execute("INSERT INTO books VALUES (%s, %s, %s, %s, %s, %s,%s)", (ISBN,book_title,category,rental_price, status, author,publisher
    db.commit()
    st.success("book added successfully")
st.subheader("Update a book")

ISBN = st.number_input("ISBN",key="ISBN_update")
book_title = st.text_input("Title",key="book_title_update")
category = st.text_input("Category",key="category_update")
rental_price = st.number_input("Rental_Price",key="rental_price_update")
status = st.text_input("Status",key="status_update")
author = st.text_input("Author",key="author_update")
publisher=st.text_input("Publisher",key="publisher_update")

if st.button("Update book"):
    cursor.execute("UPDATE books SET book_title = %s, category = %s, rental_price = %s, status = %s, author = %s, publisher =%s WHERE ISBN
    db.commit()
    st.success("book updated successfully")
st.subheader("Delete a book")
train_no = st.number_input("ISBN",key="ISBN_delete")
if st.button("Delete book"):
    cursor.execute("DELETE FROM books WHERE ISBN = %s", (ISBN,))
    db.commit()
    st.success("book removed successfully")
```

```
67    cursor.close()
68    db.close()
```

9. FUNCTION

```
DELIMITER $$
CREATE FUNCTION func5(ISBN INT)
RETURNS INT
BEGIN
   DECLARE no_of_copies INT;
   DECLARE no_issued INT;

   SELECT no_of_copies INTO no_of_copies FROM books;
   SELECT no_issued INTO no_issued FROM customer;

   IF no_issued >= 1 THEN
      UPDATE books SET no_of_copies = no_of_copies - no_issued WHERE ISBN = ISBN;
   ELSEIF no_of_copies = 0 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot issue a book as availability status is NULL';
      UPDATE books SET status = 'un-available' WHERE ISBN = ISBN;
   END IF;

   RETURN no_of_copies;
END $$
DELIMITER ;
```

CONCLUSION

- SQL Database management application is very well used uin the modern world in organizing and manipulating a database.

- Though SQL doesn’t have the GUI interface like Microsoft access is having and they all manage the database comfortably.

- Depending on the user or users, if an organization has multiple users, then they should go for SQL server-based application.

- This project shows how to create tables in SQL and how to create simple data manipulation language and data definition language with how to execute them.

- It also shows how the relationships are established with the concepts of primary and foreign key within a table.

- Lastly, the project shows how queries are created in SQL server, queries like the create, view, update, alter, etc.

REFERENCES

- http://people.cs.pitt.edu/~chang/156/03ERmodel.html
- http://www.academia.edu/13780884/Database_system_for_library_management_system
- https://lbsitbytes2010.wordpress.com/2013/09/21/er-diagram-of-library-management-rno15s5cs2/
- https://www.slideshare.net/fiu025/library-management-32343393?next_slideshow=1
- http://www.c-sharpcorner.com/UploadFile/ea3ed6/database-design-for-library-management-system
- http://stackoverflo.com/question/17641134/what-is-diiferent-between-er-diagram-and-database-schema