Project Report Submitted for
Database Management System (UCS310)

# Topic: General Library Management System

Submitted By

Rehatman Kaur
Shiven Khare
Hrishita Dalal

Batch: 2CO20
Submitted to
Dr. Rajendra Kumar Roul


## DEPARTMENT of COMPUTER SCIENCE and ENGINEERING

THAPAR INSTITUE OF ENGINEERING AND TECHNOLOGY
(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB, INDIA

**JAN – MAY 2024 (2324EVESEM 4th SEMESTER)**

# Index

# Problem Statement

Libraries serve as critical repositories of knowledge and information. However, manual handling of library operations, from book lending to tracking overdue returns, is both labor-intensive and error-prone. The goal of this project is to develop an automated Library Management System (LMS) that facilitates efficient library operations, enhances the user experience, and provides librarians with powerful tools to manage the inventory and user database.

In traditional library systems, many tasks are performed manually, including book cataloguing, circulation, and member management. This manual system leads to several issues:

- **Increased Error Rates:** Manual data entry and tracking increase the likelihood of errors.
- **Inefficient Resource Utilization:** Librarians spend a significant amount of time on administrative tasks which could be automated.
- **Limited Access to Information:** Members lack real-time access to book availability and their borrowing history.
- **Difficulty in Managing Overdue Returns:** Tracking and managing overdue books is challenging and often results in revenue loss due to uncollected fines.

# Introduction

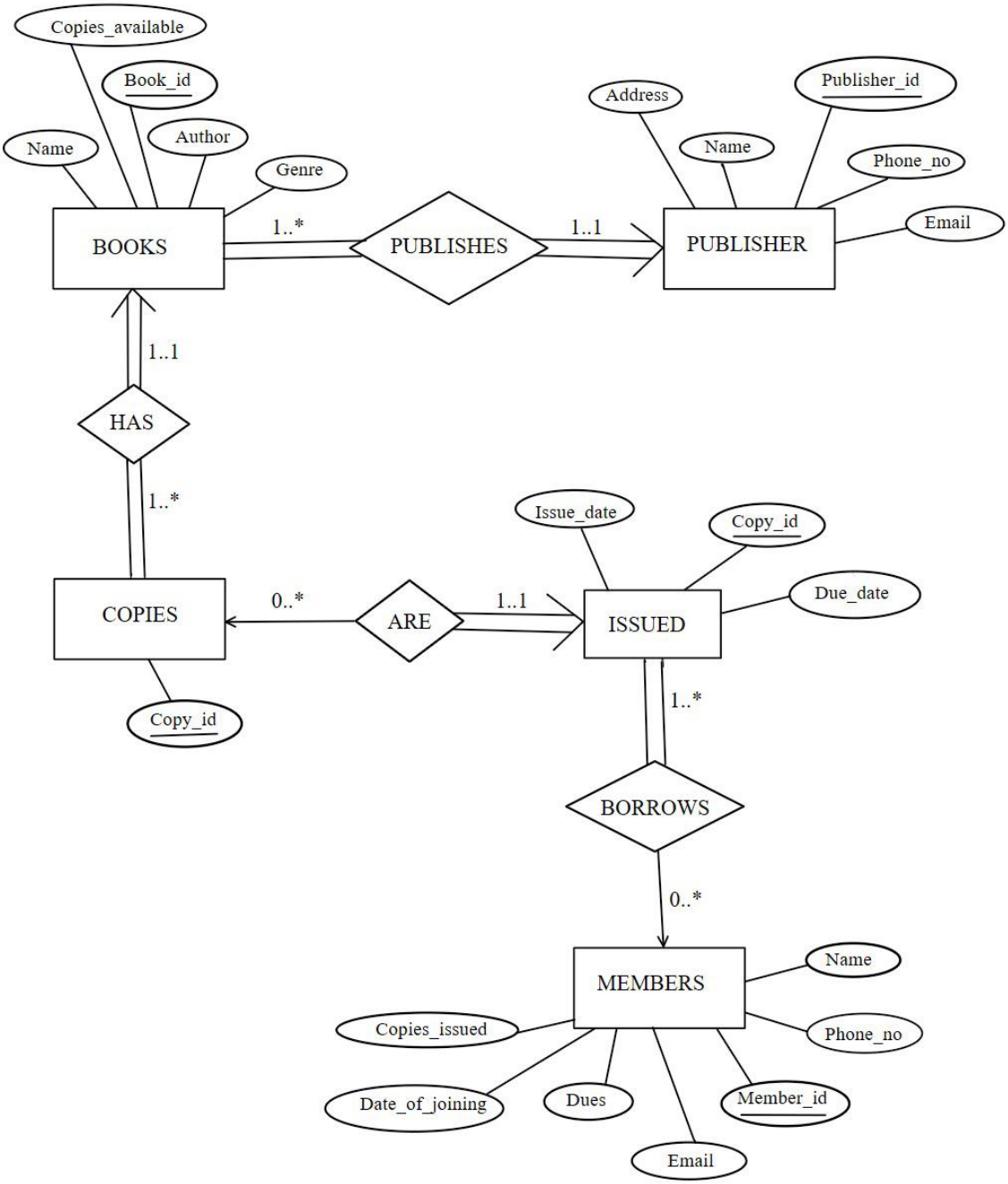
A library management system is a tool that manages and stores book information digitally, catering to the needs of its patrons. This system enables both library staff and patrons to keep a constant track of all books available in the library. It facilitates the search for desired books, making it essential for libraries to maintain a continuous check on the books issued and returned, as well as calculate fines. This digital approach reduces the chances of errors associated with manual operations by maintaining up-to-date records of issue dates, due dates, and returns, eliminating the need for manual tracking.

This system significantly reduces manual workload, enhancing the efficiency of library operations by minimizing errors in transaction details.

# Advantages

- **Paperless Management:** The system digitizes all book information, thus eliminating the need for paper records.
- **Continuous Updates:** Library staff can continuously update the system with new book arrivals and their availability, ensuring patrons do not need to physically visit the library for issuing purposes.
- **Systematic Organization:** Books are systematically categorized and organized within the system, allowing users to easily search and locate books.
- **Efficiency and Resource Conservation:** Automating library management saves significant human effort and resources, making library services more efficient for both staff and patrons.

# ER Diagram


Copies_available
Book_id
Author
Name
Genre
BOOKS
1..*
PUBLISHES
1..1
PUBLISHER
Address
Name
Publisher_id
Phone_no
Email
1..1
HAS
1..*
COPIES
0..*
ARE
1..1
ISSUED
Issue_date
Copy_id
Due_date
Copy_id
1..*
BORROWS
0..*
MEMBERS
Name
Phone_no
Copies_issued
Date_of_joining
Dues
Email
Member_id

# ER to Table

```
SQL> desc books;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOK_ID                                   NOT NULL NUMBER(6)
 NAME                                      NOT NULL VARCHAR2(50)
 AUTHOR                                             VARCHAR2(30)
 GENRE                                              VARCHAR2(30)
 COPIES_AVAILABLE                                   NUMBER(38)
 PUBLISHER_ID                              NOT NULL NUMBER(6)

SQL> desc members;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MEMBER_ID                                 NOT NULL NUMBER(6)
 NAME                                      NOT NULL VARCHAR2(30)
 BOOKS_ISSUED                                       NUMBER(38)
 DUES                                               FLOAT(126)
 DATE_OF_JOINING                           NOT NULL DATE
 PHONE_NO                                  NOT NULL NUMBER(10)
 EMAIL                                              VARCHAR2(40)

SQL> desc publisher;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PUBLISHER_ID                              NOT NULL NUMBER(6)
 NAME                                      NOT NULL VARCHAR2(30)
 PHONE_NO                                  NOT NULL NUMBER(10)
 EMAIL                                              VARCHAR2(40)
 ADDRESS                                   NOT NULL VARCHAR2(50)

SQL> desc copies;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COPY_ID                                   NOT NULL NUMBER(6)
 BOOK_ID                                   NOT NULL NUMBER(6)
 MEMBER_ID                                          NUMBER(6)
 ISSUE_DATE                                         DATE
 DUE_DATE                                           DATE

SQL> desc history;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COPY_ID                                   NOT NULL NUMBER(6)
 BOOK_ID                                   NOT NULL NUMBER(6)
 MEMBER_ID                                 NOT NULL NUMBER(6)
 ISSUE_DATE                                NOT NULL DATE
 RETURN_DATE                               NOT NULL DATE
```

History table is used to record information of all books issued/returned till date.

Primary Keys: Book_ID (Books), Member_ID (Members), Publisher_ID (Publisher), Copy_ID (Copies)

Foreign Keys:

- Books(Publisher_ID) references Publisher(Publisher_ID)
- Copies(Book_ID) references Books(Book_ID)
- Copies(Member_ID) references Members(Member_ID)

# Normalization

The analysis of the database schema represented in the ER diagram shows that it adheres to the first three normal forms, ensuring efficient and redundancy-free data management:

1. First Normal Form (1NF): Each table contains only atomic values with consistent data types. For example, the ‘books’ table with fields like ‘book_id’, ‘name’, and ‘author’ confirms adherence to 1NF.

2. Second Normal Form (2NF): All non-key attributes in each table are fully dependent on the primary key. In the ‘books’ table, attributes like ‘name’ and ‘author’ depend solely on ‘book_id’, indicating compliance with 2NF.

3. Third Normal Form (3NF): There are no transitive dependencies within the tables. For instance, ‘publisher_id’ in the ‘books’ table links directly to the ‘publisher’ table and depends only on ‘book_id’, not on any non-key attribute.

In conclusion, the schema effectively reaches up to the Third Normal Form (3NF), promoting data integrity and operational efficiency by ensuring that non-key attributes depend strictly on the primary keys.

# PL/SQL Codes

**--CREATION OF TABLES**

```
create table books(book_id number(6) constraint books_pk
primary key, name varchar(30) constraint books_name not null,
author varchar(30), genre varchar(30), copies_available int,
publisher_id number(6) constraint books_pubid not null,
constraint publisher_fk foreign key(publisher_id) references
publisher(publisher_id));

create table members(member_id number(6) constraint members_pk
primary key, name varchar(30) constraint members_name not
null, books_issued int, dues float, date_of_joining date
constraint members_date not null, phone_no number(10)
constraint members_phone not null, email varchar(40));

create table publisher(publisher_id number(6) constraint
publisher_pk primary key, name varchar(30) constraint
publisher_name not null, phone_no number(10) constraint
publisher_phone not null, email varchar(40), address
varchar(50) constraint publisher_address not null);

create table copies(copy_id number(6) constraint copies_pk
primary key, book_id number(6) constraint book_nn1 not null,
member_id number(6), issue_date date, due_date date,
constraint book_fk1 foreign key (book_id) references
books(book_id), constraint member_fk1 foreign key (member_id)
references members(member_id));

create table history(copy_id number(6) constraint copy_nn not
null, book_id number(6) constraint book_nn2 not null,
member_id number(6) constraint member_nn not null, issue_date
date constraint issue_nn not null, return_date date constraint
return_nn not null);
```

**-- TRIGGER FOR NO OPERATIONS ON SUNDAY**

```
create or replace trigger no_operations_on_sunday
before insert or update or delete on copies
begin
if to_char(sysdate, 'day')='Sunday' then
raise_application_error(-20001, 'No borrowing allowed on
sunday.');
end if;
end;
```

**-- TRGGER TO INSERT THE RETURNED BOOK INTO HISTORY AND UPDATE THE VALUES OF COPIES_AVAILABLE IN BOOKS TABLE AND BOOKS_ISSUED IN MEMBERS**

```
create or replace trigger book_returned
before delete on copies
for each row
begin
insert into history values(:old.copy_id, :old.book_id,
:old.member_id, :old.issue_date, sysdate);
update books set copies_available=copies_available+1 where
book_id=:old.book_id;
update members set books_issued=books_issued-1 where
member_id=:old.member_id;
end;
```

**-- TRGGER UPDATE THE VALUES OF COPIES_AVAILABLE IN BOOKS TABLE AND BOOKS_ISSUED IN MEMBERS WHEN A BOOK IS ISSUED**

```
create or replace trigger book_issued
after update of member_id on copies
for each row
begin
update books set copies_available=copies_available-1 where
book_id=:new.book_id;
update members set books_issued=books_issued+1 where
member_id=:new.member_id;
end;
```

**-- PROCEDURE TO DISPLAY THE BOOK DETAILS**

```
create or replace procedure display_books(id in number,
book_name in varchar) is
cursor c1 is select * from books where book_id=id or
name=book_name;
b_id books.book_id%type;
begin
select book_id into b_id from books where book_id=id or
name=book_name;
for rec in c1 loop
dbms_output.put_line('Book ID: ' || rec.book_id || chr(10) ||
'Name: ' || rec.name || chr(10) || 'Author: ' || rec.author ||
chr(10) || 'Genre: ' || rec.genre || chr(10) || 'Number of
Copies Available: ' || rec.copies_available);
end loop;
exception
when no_data_found then
dbms_output.put_line('Invalid ID or book not found.');
end;
```

**-- PROCEDURE TO DISPLAY THE MEMBER DETAILS**

```
create or replace procedure display_members(id in number,
member_name in varchar) is
cursor c1 is select * from members where member_id=id or
name=member_name;
m_id members.member_id%type;
begin
select member_id into m_id from members where member_id=id or
name=member_name;
for rec in c1 loop
dbms_output.put_line('Member ID: ' || rec.member_id || chr(10)
|| 'Name: ' || rec.name || chr(10) || 'Number of books
borrowed: ' || rec.books_issued || chr(10) || 'Dues: ' ||
rec.dues || chr(10) || 'Date of joining: ' ||
rec.date_of_joining || chr(10) || 'Phone number: ' ||
rec.phone_no || chr(10) || 'Email: ' || rec.email);
end loop;
exception
when no_data_found then
dbms_output.put_line('Invalid ID or member not found.');
end;
```

**-- PROCEDURE TO DISPLAY THE PUBLISHER DETAILS**

```
create or replace procedure display_publisher(id in number,
publisher_name in varchar) is
cursor c1 is select * from publisher where publisher_id=id or
name=publisher_name;
cursor c2 is select book_id, name from books where
publisher_id in (select publisher_id from publisher where
publisher_id=id or name=publisher_name);
p_id publisher.publisher_id%type;
begin
select publisher_id into p_id from publisher where
publisher_id=id or name=publisher_name;
for rec in c1 loop
dbms_output.put_line('Publisher ID: ' || rec.publisher_id ||
chr(10) || 'Name: ' || rec.name || chr(10) || 'Phone number: '
|| rec.phone_no || chr(10) || 'Email: ' || rec.email ||
chr(10) || 'Address: ' || rec.address || chr(10));
dbms_output.put_line('ID      BOOK');
for rec in c2 loop
dbms_output.put_line(rec.book_id || '  ' || rec.name);
end loop;
end loop;
exception
when no_data_found then
dbms_output.put_line('Invalid ID or publisher not found.');
end;
```

**-- PROCEDURE FOR ISSUING/BORROWING A BOOK**

```
create or replace procedure book_issued(c_id in number, m_id
in number) is
no_of_books members.books_issued%type;
id copies.copy_id%type;
fine members.dues%type;
begin
select member_id into id from copies where copy_id=c_id;
select dues into fine from members where member_id=m_id;
if fine>200 then
raise_application_error(-20003, 'Clear previous dues to issue
new book.');
if id is not null then
raise_application_error(-20002, 'Book already issued.');
end if;
select books_issued into no_of_books from members where
member_id=m_id;
if no_of_books<5 then
update copies set member_id=m_id, issue_date=sysdate,
due_date=sysdate+1 where copy_id=c_id;
else
dbms_output.put_line('Maximum limit(5) reached on book
borrowing.');
end if;
exception
when no_data_found then
dbms_output.put_line('Invalid ID.');
end;
```

**-- PROCEDURE FOR RETURNING A BOOK**

```
create or replace procedure book_returned(id in number) is
b_id history.book_id%type;
c_id copies.copy_id%type;
begin
select copy_id into c_id from copies where copy_id=id;
delete from copies where copy_id=c_id;
select book_id into b_id from history where copy_id=c_id;
insert into copies(copy_id, book_id) values (c_id, b_id);
exception
when no_data_found then
dbms_output.put_line('Invalid ID.');
end;
```

**-- PROCDEDURE TO CALL THE dues_calc FUNCTION EVERYDAY**

```
create or replace procedure calculate_dues is
cursor c1 is select member_id from members;
fine float;
begin
for rec in c1 loop
fine := dues_calc(rec.member_id);
update members set dues=fine where member_id=rec.member_id;
end loop;
end;
```

**-- PROCEDURE TO UPDATE DUES WHEN THE MEMBER PAYS FULL/PARTIAL AMOUNT**

```
create or replace procedure dues_paid(m_id in number, amount
in number) as
begin
update members set dues=dues-amount where member_id=m_id;
end;
```

**-- FUNCTION TO CALCULATE DUES**

```
create or replace function dues_calc(m_id in number) return
float is
cursor c1 is select due_date from copies where member_id=m_id;
days number;
fine float:=0;
begin
for rec in c1 loop
days:= round(sysdate - rec.due_date, 0);
if days>0 then
fine:=fine + days * 5;
end if;
end loop;
return fine;
end;
```

# Output Screenshots

## Creation of all the triggers, procedures and function as mentioned in the PL/SQL code:

-- Creation of trigger ‘no operation on Sunday’

```
SQL> create or replace trigger no_operations_on_sunday
  2  before insert or update or delete on copies
  3  begin
  4  if to_char(sysdate, 'day')='Sunday' then
  5  raise_application_error(-20001, 'No borrowing allowed on sunday.');
  6  end if;
  7  end;
  8  /

Trigger created.
```

-- Creation of trigger ‘book_returned’

```
  1  create or replace trigger book_returned
  2  before delete on copies
  3  for each row
  4  begin
  5  insert into history values(:old.copy_id, :old.book_id, :old.member_id, :old.issue_date, sysdate);
  6  update books set copies_available=copies_available+1 where book_id=:old.book_id;
  7  update members set books_issued=books_issued-1 where member_id=:old.member_id;
  8* end;
SQL> /

Trigger created.
```

-- Creation of trigger ‘book_issued’

```
  1  create or replace trigger book_issued
  2  after update of member_id on copies
  3  for each row
  4  begin
  5  update books set copies_available=copies_available-1 where book_id=:new.book_id;
  6  update members set books_issued=books_issued+1 where member_id=:new.member_id;
  7* end;
SQL> /

Trigger created.
```

-- Creation of procedure ‘display_books’

```
  1  create or replace procedure display_books(id in number, book_name in varchar) is
  2  cursor c1 is select * from books where book_id=id or name=book_name;
  3  b_id books.book_id%type;
  4  begin
  5  select book_id into b_id from books where book_id=id or name=book_name;
  6  for rec in c1 loop
  7  dbms_output.put_line('Book ID: ' || rec.book_id || chr(10) || 'Name: ' || rec.name
 || chr(10) || 'Author: ' || rec.author || chr(10) || 'Genre: ' || rec.genre || chr(10)
 || 'Number of Copies Available: ' || rec.copies_available);
  8  end loop;
  9  exception
 10  when no_data_found then
 11  dbms_output.put_line('Invalid ID or book not found.');
 12* end;
SQL> /

Procedure created.
```

-- Creation of procedure ‘display_members’

```
  1  create or replace procedure display_members(id in number, member_name in varchar)
is
  2  cursor c1 is select * from members where member_id=id or name=member_name;
  3  m_id members.member_id%type;
  4  begin
  5  select member_id into m_id from members where member_id=id or name=member_name;
  6  for rec in c1 loop
  7  dbms_output.put_line('Member ID: ' || rec.member_id || chr(10) || 'Name: ' || rec.
name || chr(10) || 'Number of books borrowed: ' || rec.books_issued || chr(10) || 'Dues
: ' || rec.dues || chr(10) || 'Date of joining: ' || rec.date_of_joining || chr(10) ||
'Phone number: ' || rec.phone_no || chr(10) || 'Email: ' || rec.email);
  8  end loop;
  9  exception
 10  when no_data_found then
 11  dbms_output.put_line('Invalid ID or member not found.');
 12* end;
SQL> /

Procedure created.
```

-- Creation of procedure ‘display_publishers’

```
  1  create or replace procedure display_publisher(id in number, publisher_name in varchar) is
  2  cursor c1 is select * from publisher where publisher_id=id or name=publisher_name;
  3  cursor c2 is select book_id, name from books where publisher_id in (select publisher_id from publisher
 where publisher_id=id or name=publisher_name);
  4  p_id publisher.publisher_id%type;
  5  begin
  6  select publisher_id into p_id from publisher where publisher_id=id or name=publisher_name;
  7  for rec in c1 loop
  8  dbms_output.put_line('Publisher ID: ' || rec.publisher_id || chr(10) || 'Name: ' || rec.name || chr(10
) || 'Phone number: ' || rec.phone_no || chr(10) || 'Email: ' || rec.email || chr(10) || 'Address: ' || rec
.address || chr(10));
  9  dbms_output.put_line('ID      BOOK');
 10  for rec in c2 loop
 11  dbms_output.put_line(rec.book_id || '  ' || rec.name);
 12  end loop;
 13  end loop;
 14  exception
 15  when no_data_found then
 16  dbms_output.put_line('Invalid ID or publisher not found.');
 17* end;
SQL> /

Procedure created.
```

-- Creation of procedure ‘book_issued’

```
 1  create or replace procedure book_issued(c_id in number, m_id in number) is
 2  no_of_books members.books_issued%type;
 3  id copies.copy_id%type;
 4  fine members.dues%type;
 5  begin
 6  select member_id into id from copies where copy_id=c_id;
 7  select dues into fine from members where member_id=m_id;
 8  if fine>200 then
 9  raise_application_error(-20003, 'Clear previous dues to issue new book.');
10  end if;
11  if id is not null then
12  raise_application_error(-20002, 'Book already issued.');
13  end if;
14  select books_issued into no_of_books from members where member_id=m_id;
15  if no_of_books<5 then
16  update copies set member_id=m_id, issue_date=sysdate, due_date=sysdate+1 where copy_id=c_id;
17  else
18  dbms_output.put_line('Maximum limit(5) reached on book borrowing.');
19  end if;
20  exception
21  when no_data_found then
22  dbms_output.put_line('Invalid ID.');
23* end;
SQL> /

Procedure created.
```

-- Creation of procedure ‘book_returned’

```
 1  create or replace procedure book_returned(id in number) is
 2  b_id history.book_id%type;
 3  c_id copies.copy_id%type;
 4  begin
 5  select copy_id into c_id from copies where copy_id=id;
 6  delete from copies where copy_id=c_id;
 7  select book_id into b_id from history where copy_id=c_id;
 8  insert into copies(copy_id, book_id) values (c_id, b_id);
 9  exception
10  when no_data_found then
11  dbms_output.put_line('Invalid ID.');
12* end;
SQL> /

Procedure created.
```

-- Creation of procedure ‘calculate_dues’

```
  1  create or replace procedure calculate_dues is
  2    cursor c1 is select member_id from members;
  3    fine float;
  4  begin
  5    for rec in c1 loop
  6      fine := dues_calc(rec.member_id);
  7      update members set dues=fine where member_id=rec.member_id;
  8    end loop;
  9* end;
SQL> /

Procedure created.
```

-- Creation of procedure ‘dues_paid’

```
SQL> create or replace procedure dues_paid(m_id in number, amount in number) as
  2  begin
  3  update members set dues = dues - amount where member_id = m_id;
  4  end;
  5  /

Procedure created.
```

-- Creation of function ‘dues_calc’

```
  1  create or replace function dues_calc(m_id in number) return float is
  2  cursor c1 is select due_date from copies where member_id=m_id;
  3  days number;
  4  fine float:=0;
  5  begin
  6  for rec in c1 loop
  7  days:=round(sysdate - rec.due_date,0);
  8  if days>0 then
  9  fine:=fine + days * 5;
 10  end if;
 11  end loop;
 12  return fine;
 13* end;
SQL> /

Function created.
```

## Records in the table after insertion and execution of the procedures and function:

-- Initial values in the table Publisher

```
SQL> select * from publisher;

PUBLISHER_ID NAME                           PHONE_NO EMAIL                                ADDRESS
------------ ------------------------------ -------- ------------------------------------ --------------------------------------------------
      101001 McGraw Hill                    1800103587 mcgrawhill@gmail.com                 Sector 62, NOIDA
      101002 Pearsons                       1800937593 pearsons@gmail.com                   Langford Gardens, Bangalore
      101003 Oracle Publications            1800294729 oracle@gmail.com                     Ananth Info Park, Hyderabad
```

-- Initial values in the table Books

```
SQL> select * from books;

   BOOK_ID NAME                                       AUTHOR                    GENRE                       COPIES_AVAILABLE PUBLISHER_ID
---------- ---------------------------------------- ------------------------- --------------------------- ---------------- ------------
    384000 Oracle PL/SQL Programming                 Steven Feuerstein         Educational (DBMS)                         5       101003
    439000 Building Python Programs                  Stuart Reges              Educational (Programming)                  5       101002
    823000 Fundamentals of Corporate Finance         Stephen A. Ross           Educational (Business)                     5       101001
    924000 Object Oriented Programming with C++      E. Balagurusamy           Educational (Programming)                  5       101001
```

-- Initial values in the table Copies

```
SQL> select * from copies;

   COPY_ID    BOOK_ID  MEMBER_ID ISSUE_DAT DUE_DATE
---------- ---------- ---------- --------- ---------
    384001     384000
    384002     384000
    384003     384000
    384004     384000
    384005     384000
    439001     439000
    439002     439000
    439003     439000
    439004     439000
    439005     439000
    823001     823000
    823002     823000
    823003     823000
    823004     823000
    823005     823000
    924001     924000
    924002     924000
    924003     924000
    924004     924000
    924005     924000

20 rows selected.
```

-- Initial values in the table Members

```
SQL> select * from members;

 MEMBER_ID NAME                           BOOKS_ISSUED       DUES DATE_OF_J   PHONE_NO EMAIL
---------- ------------------------------ ------------ ---------- --------- ---------- ----------------------------------------
    203852 Shiven                                    0          0 06-MAY-24 9284308293 skhare_be22@thapar.edu
    203863 Hrishita                                  0          0 06-MAY-24 8293803849 hdalal_be22@thapar.edu
    203730 Rehatman                                  0          0 06-MAY-24 7029248593 rkaur_be22@thapar.edu
```

-- Output after execution of 'display_books' procedure

```
SQL> execute display_books(null, 'Object Oriented Programming with C++');
Book ID: 924000
Name: Object Oriented Programming with C++
Author: E. Balagurusamy
Genre: Educational (Programming)
Number of Copies Available: 5

PL/SQL procedure successfully completed.
```

-- Output after execution of 'display_members' procedure

```
SQL> execute display_members(null, 'Hrishita');
Member ID: 203863
Name: Hrishita
Number of books borrowed:
Dues:
Date of joining: 06-MAY-24
Phone number: 8293803849
Email: hdalal_be22@thapar.edu

PL/SQL procedure successfully completed.
```

-- Output after execution of 'display_publisher' procedure

```
SQL> execute display_publisher(101001, null);
Publisher ID: 101001
Name: McGraw Hill
Phone number: 1800103587
Email: mcgrawhill@gmail.com
Address: Sector 62, NOIDA

ID      BOOK
823000  Fundamentals of Corporate Finance
924000  Object Oriented Programming with C++

PL/SQL procedure successfully completed.
```

-- Execution of ‘book_issued’ procedure

```
SQL> execute book_issued(439002, 203863);

PL/SQL procedure successfully completed.

SQL> execute book_issued(924003, 203730);

PL/SQL procedure successfully completed.
```

-- Execution of ‘book_issued’ procedure (We deliberately gave an invalid ID, thus to showcase the user-defined exception, i.e., ORA-20002)

```
SQL> execute book_issued(384004, 203863);
BEGIN book_issued(384004, 203863); END;

*
ERROR at line 1:
ORA-20002: Book already issued.
ORA-06512: at "COE203863.BOOK_ISSUED", line 7
ORA-06512: at line 1
```

-- Execution of ‘book_returned’ procedure

```
SQL> execute book_returned(384004);

PL/SQL procedure successfully completed.
```

-- Execution of ‘book_returned’ procedure (We deliberately gave an invalid ID, thus to showcase the error message ‘Invalid ID’)

```
SQL> execute book_returned(3820930);
Invalid ID.

PL/SQL procedure successfully completed.
```

-- Values in the table books and members after execution of the above procedures

```
SQL> select * from books;

   BOOK_ID NAME                                               AUTHOR                    GENRE                          COPIES_AVAILABLE PUBLISHER_ID
---------- -------------------------------------------------- ------------------------- ------------------------------ ---------------- ------------
    384000 Oracle PL/SQL Programming                          Steven Feuerstein         Educational (DBMS)                             4       101003
    439000 Building Python Programs                           Stuart Reges              Educational (Programming)                      3       101002
    823000 Fundamentals of Corporate Finance                  Stephen A. Ross           Educational (Business)                         5       101001
    924000 Object Oriented Programming with C++               E. Balagurusamy           Educational (Programming)                      4       101001

SQL> select * from members;

 MEMBER_ID NAME                           BOOKS_ISSUED       DUES DATE_OF_J   PHONE_NO EMAIL
---------- ------------------------------ ------------ ---------- --------- ---------- ----------------------------------------
    203852 Shiven                                    0          0 06-MAY-24 9284308293 skhare_be22@thapar.edu
    203863 Hrishita                                  2          0 06-MAY-24 8293803849 hdalal_be22@thapar.edu
    203730 Rehatman                                  2          0 06-MAY-24 7029248593 rkaur_be22@thapar.edu
```

-- Values in the table copies and history after execution of the above procedures

```
SQL> select * from history;

   COPY_ID    BOOK_ID  MEMBER_ID ISSUE_DAT RETURN_DA
---------- ---------- ---------- --------- ---------
    384004     384000     203852 06-MAY-24 06-MAY-24

SQL> select * from copies;

   COPY_ID    BOOK_ID  MEMBER_ID ISSUE_DAT DUE_DATE
---------- ---------- ---------- --------- ---------
    384001     384000     203863 06-MAY-24 07-MAY-24
    384002     384000
    384003     384000
    384005     384000
    439001     439000     203730 06-MAY-24 07-MAY-24
    439002     439000     203863 06-MAY-24 07-MAY-24
    439003     439000
    439004     439000
    439005     439000
    823001     823000
    823002     823000
    823003     823000
    823004     823000
    823005     823000
    924001     924000
    924002     924000
    924003     924000     203730 06-MAY-24 07-MAY-24
    924004     924000
    924005     924000
    384004     384000

20 rows selected.
```

-- Values in the table members after execution of the procedure calculate_dues

```
SQL> execute calculate_dues;

PL/SQL procedure successfully completed.

SQL> select * from members;

 MEMBER_ID NAME                           BOOKS_ISSUED       DUES DATE_OF_J   PHONE_NO EMAIL
---------- ------------------------------ ------------ ---------- --------- ---------- ----------------------
    203852 Shiven                                    0          0 06-MAY-24 9284308293 skhare_be22@thapar.edu
    203863 Hrishita                                  2         10 06-MAY-24 8293803849 hdalal_be22@thapar.edu
    203730 Rehatman                                  2         10 06-MAY-24 7029248593 rkaur_be22@thapar.edu
```

-- Values in the table members after execution of the procedure book_issued, if the dues exceed 200, a user-defined exception is raised

```
SQL> select * from members;

 MEMBER_ID NAME                           BOOKS_ISSUED       DUES DATE_OF_J   PHONE_NO EMAIL
---------- ------------------------------ ------------ ---------- --------- ---------- ----------------------
    203852 Shiven                                    1          0 06-MAY-24 9284308293 skhare_be22@thapar.edu
    203863 Hrishita                                  3        250 06-MAY-24 8293803849 hdalal_be22@thapar.edu
    203730 Rehatman                                  3         10 06-MAY-24 7029248593 rkaur_be22@thapar.edu

SQL> execute book_issued(924002, 203863);
BEGIN book_issued(924002, 203863); END;

*
ERROR at line 1:
ORA-20003: Clear previous dues to issue new book.
ORA-06512: at "COE203863.BOOK_ISSUED", line 9
ORA-06512: at line 1
```

-- Values in the table members after execution of the procedure dues_paid

```
SQL> execute dues_paid(203863, 10);

PL/SQL procedure successfully completed.

SQL> select * from members;

 MEMBER_ID NAME                           BOOKS_ISSUED       DUES DATE_OF_J   PHONE_NO EMAIL
---------- ------------------------------ ------------ ---------- --------- ---------- ----------------------
    203852 Shiven                                    1          0 06-MAY-24 9284308293 skhare_be22@thapar.edu
    203863 Hrishita                                  3         10 06-MAY-24 8293803849 hdalal_be22@thapar.edu
    203730 Rehatman                                  3         20 06-MAY-24 7029248593 rkaur_be22@thapar.edu
```

# Conclusion

The implementation of our Database Management System (DBMS) has significantly improved the efficiency and automation of library operations. Key processes, such as issuing and returning books, managing user accounts, and enforcing library policies, are now fully automated. Triggers within the system ensure compliance with operational policies, like no transactions on Sundays, and handle updates in real-time to book availability and user borrowing limits.

The system also features comprehensive procedures for reporting detailed information on books, members, and publishers, enhancing staff's ability to access and utilize data efficiently. Furthermore, automated dues calculations through tailored procedures and functions streamline financial transactions, promoting timely returns and maintaining financial discipline among borrowers.

Overall, this DBMS deployment has not only minimized manual intervention but has also increased the accuracy, reliability, and responsiveness of our library’s service offerings, significantly enhancing user satisfaction and operational control.