# Library Management System

Submitted for

**DATABASE MANAGEMENT SYSTEM (UCS310)**

**Mini Project**

Submitted by:


| | |
|---|---|
| **Yashvi Kumar** | **102203430** |
| **Geetansh Mohindru** | **102203718** |
| **Shrey Dhar Dubey** | **102383011** |
| **Iipsita Devgan** | **102203408** |

**BE Second Year Batch – 2CO10**


Submitted to:

**Ms. Nishu Mehta**


**Computer Science and Engineering Department**

**Thapar Institute of Engineering & Technology, Patiala**


***EVEN SEM (Jan-May 2024)***

# TABLE OF CONTENTS

# PROBLEM STATEMENT

The Library Management System (LMS) project use SQL and PL/SQL components: Cursors for result iteration, Triggers to enforce rules, Procedures, and Functions for common tasks, and Exception Handling for error management.
These elements ensure efficient management of book locating, issuing, record members within the system.

# EXPLANATION OF THE PROJECT

Introducing our advanced library management system, prioritizing efficient book location by floor and shelf, seamless book issuance and return processes, and waitlist status checks. Additionally, we offer an exclusive book club for college members, accessible only via school ID and password authentication. With membership tiers tailored to varying borrowing needs and meticulous fine calculation methods, we promote fairness and responsible borrowing habits. Automatic updates to our book table provide members with real-time availability information, enhancing their overall user experience.

1. Our library accommodates three tiers of membership:
    a) Monthly members are entitled to borrow up to four books.
    b) Yearly members, with a borrowing limit of two books.
    c) Lifetime members who can borrow up to six books.
2. Our system streamlines book retrieval by enabling users to specify floor and shelf numbers for precise locations within the library, enhancing overall efficiency and user experience.
3. In accordance with our fine policy, overdue items incur a charge of 5/- per day, encouraging timely returns and responsible borrowing habits.
4. Our exclusive book club offers avid readers a curated selection of literature.
5. Exclusively for college members, access requires authentication via College ID and password, ensuring a community of passionate readers with tailored content and discussions.

# FUNCTIONALITIES

**Issue Books:**

- User Authentication: Implement authentication for members to access the system.
- Book Availability Check: Enable users to verify book availability before issuing.
- Borrowing Limits: Set borrowing limits based on membership tier.
- Fine Calculation: Develop a system to calculate fines for overdue books.
- Transaction Records: Maintain records of book issuances and returns for tracking.

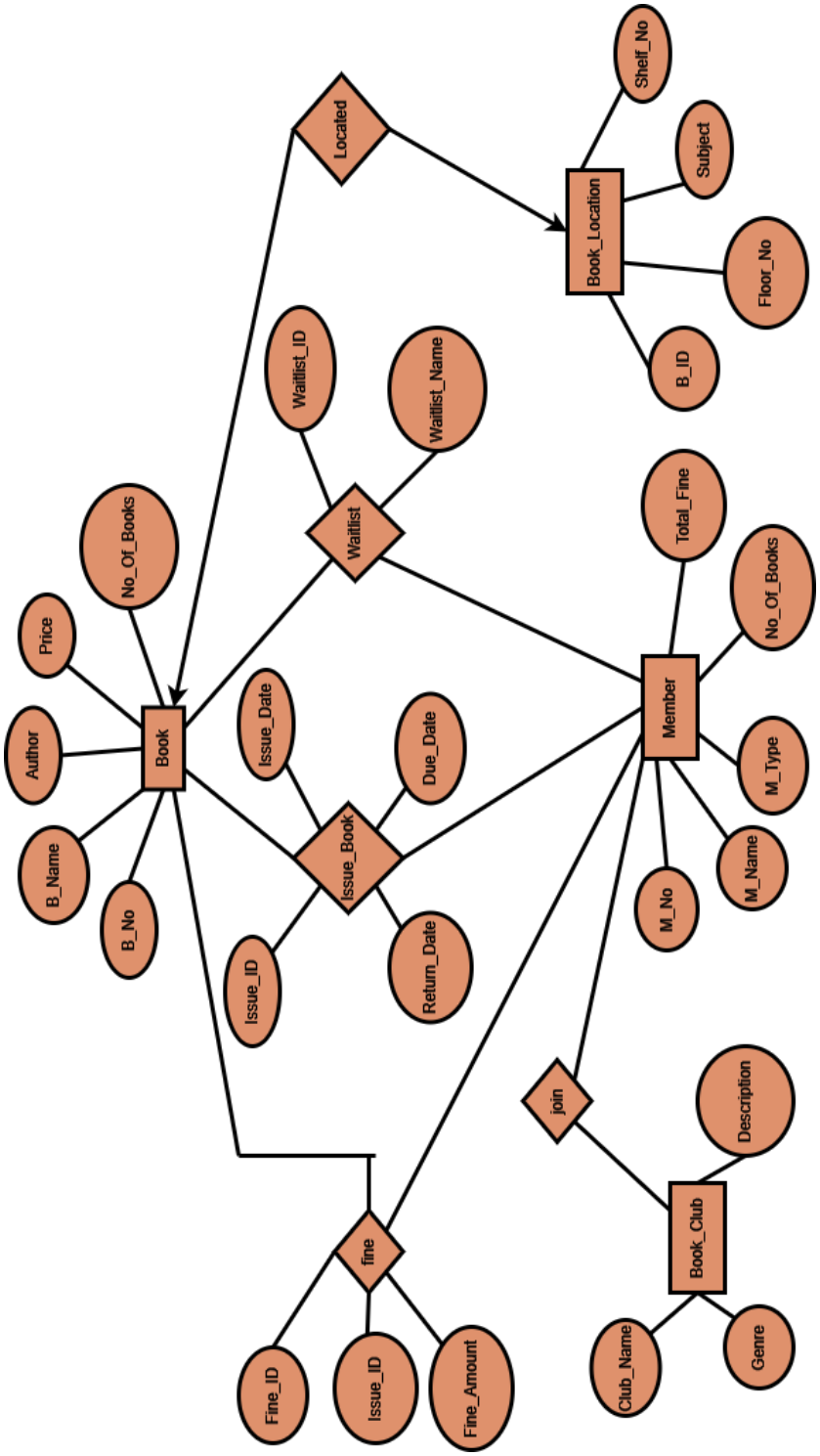
**Locating Books:**

- Floor and Shelf Information: Store and retrieve book locations by floor and shelf numbers.
- Search Functionality: Provide users with a search feature to find books by title, author, or category.
- Real-time Availability: Update availability status in real-time to reflect issued and available books.

**Book Club:**

- Membership Authentication: Authenticate college members for access to the book club.
- Curated Selection: Offer a curated selection of literature tailored to avid readers.
- Discussion Forums: Provide a platform for members to engage in book discussions and share recommendations.
- Event Management: Organize book-related events such as author talks, book signings, and reading challenges.
- Resource Sharing: Allow members to share reviews, annotations, and additional resources related to club selections.

# ER DIAGRAM


Located
Shelf_No
Subject
Book_Location
Floor_No
B_ID
Waitlist_ID
Waitlist_Name
Waitlist
No_Of_Books
Price
Author
Book
B_Name
B_No
Issue_Date
Due_Date
Issue_Book
Issue_ID
Return_Date
Total_Fine
No_Of_Books
Member
M_Type
M_Name
M_No
join
Description
Book_Club
Club_Name
Genre
fine
Fine_ID
Issue_ID
Fine_Amount

# ER DIAGRAM TO TABLE

| MEMBER | M_NO | M_NAME | M_TYPE | NO_OF_BOOKS | TOT_FINE |
|---|---|---|---|---|---|

| FINE | M_NO | B_NO | FINE_ID | ISSUE_ID | FINE_AMOUNT |
|---|---|---|---|---|---|

| BOOK | B_NO | B_NAME | AUTHOR | PRICE | NO_OF_BOOKS |
|---|---|---|---|---|---|

| ISSUE_BOOK | M_NO | B_NO | ISSUE_ID | ISSUE_DATE | RETURN_DATE | DUE_DATE |
|---|---|---|---|---|---|---|

| WAITLIST | M_NO | B_NO | WAITLIST_ID | WAITLIST_DATE |
|---|---|---|---|---|

| BOOK_CLUB | CLUB_NAME | GENRE | DESCRIPTION |
|---|---|---|---|

| JOIN | M_NO | CLUB_NAME |
|---|---|---|

| BOOK_LOCATION | B_ID | B_NO | FLOOR_NO | SUBJECT | SHELF_NO |
|---|---|---|---|---|---|

# CODE FOR CREATION OF TABLES AND PROCEDURES

```
CREATE TABLE MEMBER (
M_NO VARCHAR2(20) PRIMARY KEY,
M_NAME VARCHAR2(20) NOT NULL,
M_TYPE VARCHAR2(20),
NO_OF_BOOKS NUMBER(4),
TOT_FINE NUMBER(10,2) -- Changed data type to accommodate larger numbers
);
CREATE TABLE BOOK (
B_NO VARCHAR2(20) PRIMARY KEY,
B_NAME VARCHAR2(50) NOT NULL, -- Increased length to accommodate longer book
names
AUTHOR VARCHAR2(50), -- Increased length to accommodate longer author names
PRICE NUMBER(10,2), -- Changed data type to represent currency
NO_OF_BOOKS NUMBER(4)
);
CREATE TABLE ISSUE_BOOK (
B_NO VARCHAR2(20),
M_NO VARCHAR2(20),
ISSUE_DATE DATE,
DUE_DATE DATE,
RETURN_DATE DATE,
CONSTRAINT BID_FKEY FOREIGN KEY (B_NO) REFERENCES BOOK(B_NO),
CONSTRAINT MID_FKEY FOREIGN KEY (M_NO) REFERENCES MEMBER(M_NO)
);
CREATE TABLE TRANSACTION_HISTORY (
ISSUE_ID VARCHAR2(20) PRIMARY KEY, -- Added primary key
B_NO VARCHAR2(20),
M_NO VARCHAR2(20),
ISSUE_DATE DATE,
DUE_DATE DATE,
RETURN_DATE DATE,
CONSTRAINT BID_FKEY1 FOREIGN KEY (B_NO) REFERENCES BOOK(B_NO),
CONSTRAINT MID_FKEY1 FOREIGN KEY (M_NO) REFERENCES MEMBER(M_NO)
);
CREATE TABLE WAITLIST (
WAITLIST_ID VARCHAR2(20) PRIMARY KEY,
B_NO VARCHAR2(20),
```

```
M_NO VARCHAR2(20),
WAITLIST_DATE DATE,
CONSTRAINT WAITLIST_BID_FKEY FOREIGN KEY (B_NO) REFERENCES BOOK(B_NO),
CONSTRAINT WAITLIST_MID_FKEY FOREIGN KEY (M_NO) REFERENCES MEMBER(M_NO)
);
CREATE TABLE BOOK_CLUB (
CLUB_NAME VARCHAR2(50) PRIMARY KEY,
DESCRIPTION VARCHAR2(200),
GENRE VARCHAR2(50)
);
CREATE TABLE BOOK_LOCATION (
B_ID VARCHAR2(20) PRIMARY KEY,
B_NO VARCHAR(20) UNIQUE,
FLOOR_NO NUMBER(2),
SHELF_NO NUMBER(2),
SUBJECT VARCHAR2(50),
CONSTRAINT FK_BOOK_LOCATION FOREIGN KEY (B_NO) REFERENCES BOOK(B_NO)
);
CREATE TABLE FINE (
FINE_ID VARCHAR2(20) PRIMARY KEY,
M_NO VARCHAR2(20),
B_NO VARCHAR2(20),
ISSUE_ID VARCHAR2(20),
FINE_AMOUNT NUMBER(10, 2),
CONSTRAINT FK_FINE_MEMBER FOREIGN KEY (M_NO) REFERENCES MEMBER(M_NO),
CONSTRAINT FK_FINE_BOOK FOREIGN KEY (B_NO) REFERENCES BOOK(B_NO),
CONSTRAINT FK_FINE_TRANSACTION FOREIGN KEY (ISSUE_ID) REFERENCES
TRANSACTION_HISTORY(ISSUE_ID) -- Corrected reference to TRANSACTION_HISTORY
);
CREATE TABLE JOIN_TABLE (
M_NO VARCHAR2(20),
CLUB_NAME VARCHAR2(50),
CONSTRAINT JOIN_PK PRIMARY KEY (M_NO, CLUB_NAME),
CONSTRAINT JOIN_FK1 FOREIGN KEY (M_NO) REFERENCES MEMBER(M_NO),
CONSTRAINT JOIN_FK2 FOREIGN KEY (CLUB_NAME) REFERENCES
BOOK_CLUB(CLUB_NAME)
);
INSERT INTO MEMBER VALUES('1','DEEPESH','M',2,NULL);
INSERT INTO MEMBER VALUES('2','PRIYANSH','L',0,NULL);
INSERT INTO MEMBER VALUES('3','AKASH','Y',2,NULL);
INSERT INTO MEMBER VALUES('4','SWATI','M',4,NULL);
INSERT INTO MEMBER VALUES('5','BOSS','L',1,NULL);
INSERT INTO MEMBER VALUES('6','PRATIKHYA','Y',1,NULL);
```

```
INSERT INTO MEMBER VALUES('7','DHRUTI','L',2,NULL);
INSERT INTO BOOK VALUES('B1','YES YOU CAN WIN!','GAREY V','200',2);
INSERT INTO BOOK VALUES('B2','MIDNIGHT LIBRARY','MATT HAIG','470',3);
INSERT INTO BOOK VALUES('B3','HOW I MET UR MOTHER?','BARNEY SINSTON','500',5);
INSERT INTO BOOK VALUES('B4','CORPORATE CHANKYA','MIRAL','170',5);
INSERT INTO BOOK VALUES('B5','LIFE AT EDGE','TDP','650',0);
INSERT INTO BOOK VALUES('B6','VIVEK GEETA!','KABIR','180',2);
INSERT INTO ISSUE_BOOK VALUES('B4','7','01-MAY-20','05-MAY-20','07-MAY-20');
INSERT INTO ISSUE_BOOK VALUES('B3','5','01-MAY-20','05-MAY-20',NULL);
INSERT INTO ISSUE_BOOK VALUES('B3','2','07-MAY-20','14-MAY-20',NULL);
INSERT INTO ISSUE_BOOK VALUES('B6','3','07-MAY-20','14-MAY-20',NULL);
INSERT INTO ISSUE_BOOK VALUES('B1','1','07-MAY-20','14-MAY-20',NULL);
INSERT INTO TRANSACTION_HISTORY (ISSUE_ID, B_NO, M_NO, ISSUE_DATE, DUE_DATE,
RETURN_DATE)
VALUES ('1', 'B1', '1', TO_DATE('07-MAY-20', 'DD-MON-YY'), TO_DATE('14-MAY-
20', 'DD-MON-YY'),
NULL);
INSERT INTO TRANSACTION_HISTORY (ISSUE_ID, B_NO, M_NO, ISSUE_DATE, DUE_DATE,
RETURN_DATE)
VALUES ('2', 'B3', '2', TO_DATE('07-MAY-20', 'DD-MON-YY'), TO_DATE('14-MAY-
20', 'DD-MON-YY'),
NULL);
INSERT INTO TRANSACTION_HISTORY (ISSUE_ID, B_NO, M_NO, ISSUE_DATE, DUE_DATE,
RETURN_DATE)
VALUES ('3', 'B6', '3', TO_DATE('07-MAY-20', 'DD-MON-YY'), TO_DATE('14-MAY-
20', 'DD-MON-YY'),
NULL);
INSERT INTO WAITLIST (WAITLIST_ID, B_NO, M_NO, WAITLIST_DATE)
VALUES ('W1', 'B5', '1', SYSDATE);
INSERT INTO WAITLIST (WAITLIST_ID, B_NO, M_NO, WAITLIST_DATE)
VALUES ('W2', 'B2', '2', SYSDATE);
INSERT INTO WAITLIST (WAITLIST_ID, B_NO, M_NO, WAITLIST_DATE)
VALUES ('W3', 'B4', '3', SYSDATE);
INSERT INTO BOOK_CLUB (CLUB_NAME, DESCRIPTION, GENRE) VALUES ('Bookworms', 'A
club for
avid readers who love discussing books and sharing recommendations.',
'Various');
INSERT INTO BOOK_CLUB (CLUB_NAME, DESCRIPTION, GENRE) VALUES ('Literature
Lovers',
'Dedicated to exploring classic and contemporary literature from around the
world.', 'Literature');
INSERT INTO BOOK_CLUB (CLUB_NAME, DESCRIPTION, GENRE) VALUES ('Fiction
Fanatics', 'A
```

```
club for fans of fiction novels, including sci-fi, fantasy, and thrillers.',
'Fiction');
INSERT INTO BOOK_CLUB (CLUB_NAME, DESCRIPTION, GENRE) VALUES ('Mystery
Readers', 'For
those who enjoy solving puzzles and exploring the world of mystery novels.',
'Mystery');
INSERT INTO BOOK_CLUB (CLUB_NAME, DESCRIPTION, GENRE) VALUES ('Sci-Fi
Enthusiasts',
'Exploring the realms of science fiction and speculative fiction.', 'Sci-Fi');
INSERT INTO BOOK_CLUB (CLUB_NAME, DESCRIPTION, GENRE) VALUES ('Self-Help
Seekers', 'A
club dedicated to personal growth, self-improvement, and motivation.', 'Self-
Help');
INSERT INTO BOOK_CLUB (CLUB_NAME, DESCRIPTION, GENRE) VALUES ('History Buffs',
'Exploring the past through historical fiction, non-fiction, and
biographies.', 'History');
INSERT INTO BOOK_LOCATION (B_ID,B_NO, FLOOR_NO, SHELF_NO, SUBJECT)
VALUES (101,'B1', 1, 2, 'Self-help');
INSERT INTO BOOK_LOCATION (B_ID,B_NO, FLOOR_NO, SHELF_NO, SUBJECT)
VALUES (102,'B2', 2, 3, 'Fiction');
INSERT INTO BOOK_LOCATION (B_ID,B_NO, FLOOR_NO, SHELF_NO, SUBJECT)
VALUES (103,'B3', 3, 1, 'Comedy');
INSERT INTO BOOK_LOCATION (B_ID,B_NO, FLOOR_NO, SHELF_NO, SUBJECT)
VALUES (104,'B4', 1, 3, 'Business');
INSERT INTO BOOK_LOCATION (B_ID,B_NO, FLOOR_NO, SHELF_NO, SUBJECT)
VALUES (105,'B5', 2, 1, 'Science');
INSERT INTO BOOK_LOCATION (B_ID,B_NO, FLOOR_NO, SHELF_NO, SUBJECT)
VALUES (106,'B6', 3, 2, 'Philosophy');
INSERT INTO FINE (FINE_ID, M_NO, B_NO, ISSUE_ID, FINE_AMOUNT)
VALUES ('1', '1', 'B1', '1', 50.00);
INSERT INTO FINE (FINE_ID, M_NO, B_NO, ISSUE_ID, FINE_AMOUNT)
VALUES ('2', '2', 'B3', '2', 75.00);
INSERT INTO FINE (FINE_ID, M_NO, B_NO, ISSUE_ID, FINE_AMOUNT)
VALUES ('3', '3', 'B6', '3', 60.00);
INSERT INTO JOIN_TABLE (M_NO, CLUB_NAME) VALUES ('1', 'Bookworms');
INSERT INTO JOIN_TABLE (M_NO, CLUB_NAME) VALUES ('2', 'Literature Lovers');
INSERT INTO JOIN_TABLE (M_NO, CLUB_NAME) VALUES ('3', 'Fiction Fanatics');
INSERT INTO JOIN_TABLE (M_NO, CLUB_NAME) VALUES ('4', 'Mystery Readers');
INSERT INTO JOIN_TABLE (M_NO, CLUB_NAME) VALUES ('5', 'Sci-Fi Enthusiasts');
INSERT INTO JOIN_TABLE (M_NO, CLUB_NAME) VALUES ('6', 'Self-Help Seekers');
INSERT INTO JOIN_TABLE (M_NO, CLUB_NAME) VALUES ('7', 'History Buffs');
SELECT * FROM MEMBER;
SELECT * FROM BOOK;
```

```
SELECT * FROM ISSUE_BOOK;
SELECT * FROM TRANSACTION_HISTORY;
SELECT * FROM WAITLIST;
SELECT * FROM BOOK_CLUB;
SELECT * FROM BOOK_LOCATION;
SELECT * FROM FINE;
SELECT * FROM JOIN_TABLE;
CREATE SEQUENCE WAITLIST_SEQ START WITH 1;
- -Book issue
CREATE OR REPLACE PROCEDURE INSERT1(BOOK_ID VARCHAR2, MEM_ID NUMBER) IS
A BOOLEAN DEFAULT FALSE;
B BOOLEAN DEFAULT FALSE;
C BOOLEAN DEFAULT FALSE;
D BOOLEAN DEFAULT FALSE;
MEP NUMBER(4); -- validate whether member exists in database or not
TEP NUMBER(4); -- validate whether book exists in database or not
SEP NUMBER(4); -- check if member has already issued same book and not
returned
BEP NUMBER(4); -- check if book is available in library or not
MB NUMBER(4); -- check if member has exceeded the borrowing limit by
membership type
DAT VARCHAR2(10);
TYP VARCHAR2(10);
EXPIRY_DATE DATE;
DDATE DATE;
BEGIN
-- (A)BOOK NO SHOULD BE VALID FROM BOOK TABLE OR HANDLE EXCEPTION
SELECT COUNT(*) INTO TEP FROM BOOK WHERE B_NO = BOOK_ID;
IF TEP = 1 THEN
dbms_output.put_line('THIS BOOK ' || BOOK_ID || ' EXISTS IN LIBRARY.');
ELSE
dbms_output.put_line('THIS BOOK ' || BOOK_ID || ' DOES NOT EXIST IN
LIBRARY.');
END IF;
-- (B)MEMBER NO SHOULD BE VALID FROM MEMBER TABLE OR HANDLE EXCEPTION
SELECT COUNT(*) INTO MEP FROM MEMBER WHERE M_NO = MEM_ID;
IF MEP = 1 THEN
dbms_output.put_line('THE USER ' || MEM_ID || ' IS FROM THE CLUB.');
ELSE
dbms_output.put_line('THE USER ' || MEM_ID || ' IS NOT FROM THE CLUB.');
END IF;
-- (C)THE SAME MEMBER CAN'T BORROW THE SAME WITHOUT RETURNING IT.
SELECT COUNT(*) INTO SEP FROM ISSUE_BOOK WHERE B_NO = BOOK_ID AND M_NO =
```

```
MEM_ID AND RETURN_DATE IS NULL;
IF SEP = 1 THEN
dbms_output.put_line('ISSUING BOOK TO MEMBER ' || MEM_ID || '.');
ELSE
dbms_output.put_line('THE USER ALREADY HAS THIS BOOK.');
END IF;
-- (D)IF THE DUE DATE CROSSING THE EXPIRY DATE OF THE MEMBER THEN DON'T ISSUE
THE BOOK.
SELECT M_TYPE INTO TYP FROM MEMBER WHERE M_NO = MEM_ID;
EXPIRY_DATE := ADD_MONTHS(TRUNC(SYSDATE, 'MONTH'), 1);
DDATE := TRUNC(SYSDATE, 'YEAR');
IF TYP = 'M' THEN
IF EXPIRY_DATE < SYSDATE + 7 THEN
A := TRUE;
dbms_output.put_line('YOUR MEMBERSHIP EXPIRY DATE ' || EXPIRY_DATE || ' IS
BEFORE
DUE DATE ' || SYSDATE + 7 || '.');
END IF;
ELSIF TYP = 'Y' THEN
IF EXPIRY_DATE < SYSDATE + 7 THEN
A := TRUE;
dbms_output.put_line('YOUR MEMBERSHIP EXPIRY DATE ' || EXPIRY_DATE || ' IS
BEFORE
DUE DATE ' || SYSDATE + 7 || '.');
END IF;
ELSIF TYP = 'L' THEN
dbms_output.put_line('YOU HAVE LIFETIME MEMBERSHIP.');
END IF;
SELECT NO_OF_BOOKS INTO MB FROM MEMBER WHERE M_NO = MEM_ID;
IF TYP = 'M' THEN
IF MB >= 4 THEN
B := TRUE;
dbms_output.put_line('YOU HAVE REACHED MONTHLY BORROW LIMIT OF 4 BOOKS.');
END IF;
ELSIF TYP = 'Y' THEN
IF MB >= 2 THEN
B := TRUE;
dbms_output.put_line('YOU HAVE REACHED YEARLY BORROW LIMIT OF 2 BOOKS.');
END IF;
ELSIF TYP = 'L' THEN
IF MB >= 6 THEN
B := TRUE;
dbms_output.put_line('YOU HAVE REACHED LIFETIME BORROW LIMIT OF 6 BOOKS.');
```

```
END IF;
END IF;
-- (F)IF THE STOCK OF THE BOOK IS NOT AVAILABLE THEN TRAP THE ERROR.
SELECT NO_OF_BOOKS INTO BEP FROM BOOK WHERE B_NO = BOOK_ID;
IF BEP >= 1 THEN
D := TRUE;
dbms_output.put_line('THE BOOK IS AVAILABLE IN THE LIBRARY.');
END IF;
-- (G)IF ALL VALIDATIONS ARE FULFILLED, THEN ENTER INTO ISSUE_BOOK TABLE
-- BOOKNO.,MEMNO. ISSUE WILL BY SYSDATE AND DUE_DATE IS SYSDATE+7, RETURN DATE
IS NULL & FINE IS NULL.
IF (TEP IS NOT NULL AND MEP IS NOT NULL AND B IS NOT NULL AND A IS NOT NULL
AND D IS
NOT NULL AND C IS NOT NULL) THEN
INSERT INTO ISSUE_BOOK VALUES(BOOK_ID, MEM_ID, SYSDATE, SYSDATE + 7, NULL);
dbms_output.put_line('ITS WORKING');
END IF;
SELECT TO_CHAR(SYSDATE, 'DY') INTO DAT FROM DUAL;
IF DAT = 'SUN' THEN
dbms_output.put_line('IT IS ' || TO_CHAR(SYSDATE, 'DAY') || ' SO CANNOT ISSUE
THE BOOK.');
ELSIF DAT = 'SAT' THEN
dbms_output.put_line('IT IS ' || TO_CHAR(SYSDATE, 'DAY') || ' SO CANNOT ISSUE
THE BOOK.');
ELSE
C := TRUE;
dbms_output.put_line('IT IS ' || TO_CHAR(SYSDATE, 'DAY') || ' SO CAN ISSUE
BOOK.');
END IF;
END;
/
CREATE OR REPLACE PROCEDURE RETURNBOOK(BOOK_ID VARCHAR2, MEM_ID NUMBER) IS
FINE NUMBER(20);
MEMID NUMBER(20);
RETRN_DATE DATE NOT NULL := TO_DATE('07-MAY-20', 'DD-MON-YY');
DAT VARCHAR2(5);
DD DATE;
BEGIN
-- (A)RETURN THE BOOK IF THE MEMBER HAS BORROWED THE BOOK, CHECK IN THE
EXISTENCE ISSUE_BOOK TABLE.
SELECT M_NO INTO MEMID FROM ISSUE_BOOK WHERE M_NO = MEM_ID AND B_NO =
BOOK_ID;
-- (B)UPDATE RETURN DATE WITH CURRENT DATE & CALCULATE THE AMOUNT OF FINE.
```

```
UPDATE ISSUE_BOOK SET RETURN_DATE = TO_DATE('07-MAY-20', 'DD-MON-YY') WHERE
B_NO = BOOK_ID AND M_NO = MEM_ID;
SELECT DUE_DATE INTO DD FROM ISSUE_BOOK WHERE B_NO = BOOK_ID AND M_NO =
MEM_ID;
FINE := (DD - RETRN_DATE) * 5;
dbms_output.put_line('FINE IS ' || fine);
UPDATE MEMBER SET TOT_FINE = FINE WHERE M_NO = MEM_ID;
-- (D)NO RETURN ON SATURDAY & SUNDAY.
SELECT TO_CHAR(SYSDATE, 'DY') INTO DAT FROM DUAL;
IF DAT = 'SUN' THEN
dbms_output.put_line('IT IS ' || TO_CHAR(SYSDATE, 'DAY') || ' SO YOU CANNOT
RETURN
BOOK.');
END IF;
IF DAT = 'SAT' THEN
dbms_output.put_line('IT IS ' || TO_CHAR(SYSDATE, 'DAY') || ' SO YOU CANNOT
RETURN
BOOK.');
END IF;
-- (E)UPON RETURNING THE BOOK DELETE THE INFORMATION FROM ISSUE_BOOK & MOVE
TO TRANSACTION_HISTORY TABLE.
-- USED USING TRIGGER
-- (F)CREATE TRANSACTION_HISTORY AS THAT OF ISSUE_BOOK TABLE TO RECORD OLD
DATA.
EXCEPTION
WHEN NO_DATA_FOUND THEN
dbms_output.put_line('THERE IS NO BOOK ISSUED TO THIS MEMBER');
END;
/
CREATE OR REPLACE PROCEDURE RECOMMEND_BOOKS_BY_GENRE(GENRE VARCHAR2)
IS
BEGIN
FOR REC IN (SELECT * FROM BOOK WHERE GENRE = GENRE)
LOOP
DBMS_OUTPUT.PUT_LINE('Recommended Book: ' || REC.B_NAME || ' by ' ||
REC.AUTHOR);
END LOOP;
END RECOMMEND_BOOKS_BY_GENRE;
/
CREATE OR REPLACE PROCEDURE ADD_TO_WAITLIST(BOOK_ID VARCHAR2, MEM_ID
VARCHAR2)
IS
WAITLIST_ID VARCHAR2(20);
```

```
BEGIN
SELECT 'W' || WAITLIST_SEQ.NEXTVAL INTO WAITLIST_ID FROM DUAL;
INSERT INTO WAITLIST (WAITLIST_ID, B_NO, M_NO, WAITLIST_DATE)
VALUES (WAITLIST_ID, BOOK_ID, MEM_ID, SYSDATE);
DBMS_OUTPUT.PUT_LINE('Added to waitlist. Waitlist ID: ' || WAITLIST_ID);
END ADD_TO_WAITLIST;
/
CREATE OR REPLACE PROCEDURE RECOMMEND_BOOKS_BY_TITLE(TITLE VARCHAR2)
IS
BEGIN
FOR REC IN (SELECT * FROM BOOK WHERE B_NAME = TITLE)
LOOP
DBMS_OUTPUT.PUT_LINE('Recommended Book: ' || REC.B_NAME || ' by ' ||
REC.AUTHOR);
END LOOP;
END RECOMMEND_BOOKS_BY_TITLE;
/
-- TRIGGER FOR UPDATING BOOKS ON ISSUE & RETURN
/*------------- TRIGGER TO AUTOMATICALLY INCREMENT & DECREMENT THE NO_OF_BOOKS
FROM MEMBER & BOOK TABLE UPON ISSUE & RETURN ----------------*/
CREATE OR REPLACE TRIGGER INCR_TRIGGER
AFTER INSERT OR UPDATE ON ISSUE_BOOK
FOR EACH ROW
BEGIN
IF INSERTING THEN
UPDATE BOOK
SET NO_OF_BOOKS = NO_OF_BOOKS - 1
WHERE B_NO = :NEW.B_NO;
UPDATE MEMBER
SET NO_OF_BOOKS = NO_OF_BOOKS + 1
WHERE M_NO = :NEW.M_NO;
ELSIF UPDATING THEN
UPDATE BOOK
SET NO_OF_BOOKS = NO_OF_BOOKS + 1
WHERE B_NO = :OLD.B_NO;
UPDATE MEMBER
SET NO_OF_BOOKS = NO_OF_BOOKS - 1
WHERE M_NO = :OLD.M_NO;
END IF;
END;
/
-- TRIGGER FOR DELETING DATA FROM ISSUE_BOOK & MOVE IT TO TRANSACTION HISTORY
TABLE
```

```
/*------------- TRIGGER TO MOVE ISSUE_BOOK DATA INTO TRANSACTION_HISTORY TABLE
UPON DELETION ----------------*/
CREATE OR REPLACE TRIGGER MOVE_TRIGGER
BEFORE DELETE ON ISSUE_BOOK
FOR EACH ROW
BEGIN
INSERT INTO TRANSACTION_HISTORY (B_NO, M_NO, ISSUE_DATE, DUE_DATE,
RETURN_DATE)
VALUES (:OLD.B_NO, :OLD.M_NO, :OLD.ISSUE_DATE, :OLD.DUE_DATE,
:OLD.RETURN_DATE);
END;
/
```

**SQL Worksheet**

```
1  CREATE TABLE MEMBER (
2      M_NO VARCHAR2(20) PRIMARY KEY,
3      M_NAME VARCHAR2(20) NOT NULL,
4      M_TYPE VARCHAR2(20),
5      NO_OF_BOOKS NUMBER(4),
6      TOT_FINE NUMBER(10,2) -- Changed data type to accommodate larger numbers
7  );
8
9  CREATE TABLE BOOK (
10     B_NO VARCHAR2(20) PRIMARY KEY,
```

```
Table created.

Table created.

Table created.

Table created.

Table created.

Table created.
```

**SQL Worksheet**

```
1  CREATE TABLE MEMBER (
2      M_NO VARCHAR2(20) PRIMARY KEY,
3      M_NAME VARCHAR2(20) NOT NULL,
4      M_TYPE VARCHAR2(20),
5      NO_OF_BOOKS NUMBER(4),
```

| M_NO | M_NAME | M_TYPE | NO_OF_BOOKS | TOT_FINE |
|---|---|---|---|---|
| 1 | DEEPESH | M | 2 | - |
| 2 | PRIYANSH | L | 0 | - |
| 3 | AKASH | Y | 2 | - |
| 4 | SWATI | M | 4 | - |
| 5 | BOSS | L | 1 | - |
| 6 | PRATIKHYA | Y | 1 | - |
| 7 | DHRUTI | L | 2 | - |

**SQL Worksheet**

```
CREATE TABLE BOOK (
    B_NO VARCHAR2(20) PRIMARY KEY,
    B_NAME VARCHAR2(50) NOT NULL, -- Increased length to accommodate longer book names
    AUTHOR VARCHAR2(50), -- Increased length to accommodate longer author names
    PRICE NUMBER(10,2), -- Changed data type to represent currency
```

7 rows selected.

| B_NO | B_NAME | AUTHOR | PRICE | NO_OF_BOOKS |
|---|---|---|---|---|
| B1 | YES YOU CAN WIN! | GAREY V | 200 | 2 |
| B2 | MIDNIGHT LIBRARY | MATT HAIG | 470 | 3 |
| B3 | HOW I MET UR MOTHER? | BARNEY SINSTON | 500 | 5 |
| B4 | CORPORATE CHANKYA | MIRAL | 170 | 5 |
| B5 | LIFE AT EDGE | TDP | 650 | 0 |
| B6 | VIVEK GEETA! | KABIR | 180 | 2 |

Download CSV

**SQL Worksheet**

```
CREATE TABLE ISSUE_BOOK (
    B_NO VARCHAR2(20),
    M_NO VARCHAR2(20),
    ISSUE_DATE DATE,
    DUE_DATE DATE,
```

6 rows selected.

| B_NO | M_NO | ISSUE_DATE | DUE_DATE | RETURN_DATE |
|---|---|---|---|---|
| B4 | 7 | 01-MAY-20 | 05-MAY-20 | 07-MAY-20 |
| B3 | 5 | 01-MAY-20 | 05-MAY-20 | - |
| B3 | 2 | 07-MAY-20 | 14-MAY-20 | - |
| B6 | 3 | 07-MAY-20 | 14-MAY-20 | - |
| B1 | 1 | 07-MAY-20 | 14-MAY-20 | - |

Download CSV

5 rows selected

**SQL Worksheet**

```
27  CREATE TABLE TRANSACTION_HISTORY (
28      ISSUE_ID VARCHAR2(20) PRIMARY KEY, -- Added primary key
29      B_NO VARCHAR2(20),
30      M_NO VARCHAR2(20),
31      ISSUE_DATE DATE,
32      DUE_DATE DATE,
33      RETURN_DATE DATE,
```

5 rows selected.

| ISSUE_ID | B_NO | M_NO | ISSUE_DATE | DUE_DATE | RETURN_DATE |
|---|---|---|---|---|---|
| 1 | B1 | 1 | 07-MAY-20 | 14-MAY-20 | - |
| 2 | B3 | 2 | 07-MAY-20 | 14-MAY-20 | - |
| 3 | B6 | 3 | 07-MAY-20 | 14-MAY-20 | - |

Download CSV

3 rows selected.

**SQL Worksheet**

```
38  CREATE TABLE WAITLIST (
39      WAITLIST_ID VARCHAR2(20) PRIMARY KEY,
40      B_NO VARCHAR2(20),
41      M_NO VARCHAR2(20),
42      WAITLIST_DATE DATE,
43      CONSTRAINT WAITLIST_BID_FKEY FOREIGN KEY (B_NO) REFERENCES BOOK(B_NO),
44      CONSTRAINT WAITLIST_MID_FKEY FOREIGN KEY (M_NO) REFERENCES MEMBER(M_NO)
45  );
46
```

Download CSV

3 rows selected.

| WAITLIST_ID | B_NO | M_NO | WAITLIST_DATE |
|---|---|---|---|
| W1 | B5 | 1 | 05-MAY-24 |
| W2 | B2 | 2 | 05-MAY-24 |
| W3 | B4 | 3 | 05-MAY-24 |

**SQL Worksheet**

```
47  CREATE TABLE BOOK_CLUB (
48      CLUB_NAME VARCHAR2(50) PRIMARY KEY,
49      DESCRIPTION VARCHAR2(200),
50      GENRE VARCHAR2(50)
51  );
```

| CLUB_NAME | DESCRIPTION | GENRE |
|---|---|---|
| Bookworms | A club for avid readers who love discussing books and sharing recommendations. | Various |
| Literature Lovers | Dedicated to exploring classic and contemporary literature from around the world. | Literature |
| Fiction Fanatics | A club for fans of fiction novels, including sci-fi, fantasy, and thrillers. | Fiction |
| Mystery Readers | For those who enjoy solving puzzles and exploring the world of mystery novels. | Mystery |
| Sci-Fi Enthusiasts | Exploring the realms of science fiction and speculative fiction. | Sci-Fi |
| Self-Help Seekers | A club dedicated to personal growth, self-improvement, and motivation. | Self-Help |
| History Buffs | Exploring the past through historical fiction, non-fiction, and biographies. | History |

Download CSV

**SQL Worksheet**

```
CREATE TABLE BOOK_LOCATION (
    B_ID VARCHAR2(20) PRIMARY KEY,
    B_NO VARCHAR(20) UNIQUE,
    FLOOR_NO NUMBER(2),
    SHELF_NO NUMBER(2),
```

7 rows selected.

| B_ID | B_NO | FLOOR_NO | SHELF_NO | SUBJECT |
|---|---|---|---|---|
| 101 | B1 | 1 | 2 | Self-help |
| 102 | B2 | 2 | 3 | Fiction |
| 103 | B3 | 3 | 1 | Comedy |
| 104 | B4 | 1 | 3 | Business |
| 105 | B5 | 2 | 1 | Science |
| 106 | B6 | 3 | 2 | Philosophy |

**SQL Worksheet**

```
CREATE TABLE FINE (
    FINE_ID VARCHAR2(20) PRIMARY KEY,
    M_NO VARCHAR2(20),
    B_NO VARCHAR2(20),
    ISSUE_ID VARCHAR2(20),
    FINE_AMOUNT NUMBER(10, 2),
```

Download CSV

6 rows selected.

| FINE_ID | M_NO | B_NO | ISSUE_ID | FINE_AMOUNT |
|---|---|---|---|---|
| 1 | 1 | B1 | 1 | 50 |
| 2 | 2 | B3 | 2 | 75 |
| 3 | 3 | B6 | 3 | 60 |

Download CSV

3 rows selected

**SQL Worksheet**

```
CREATE TABLE JOIN_TABLE (
    M_NO VARCHAR2(20),
    CLUB_NAME VARCHAR2(50),
    CONSTRAINT JOIN_PK PRIMARY KEY (M_NO, CLUB_NAME),
    CONSTRAINT JOIN_FK1 FOREIGN KEY (M_NO) REFERENCES MEMBER(M_NO),
```

| M_NO | CLUB_NAME |
|---|---|
| 1 | Bookworms |
| 2 | Literature Lovers |
| 3 | Fiction Fanatics |
| 4 | Mystery Readers |
| 5 | Sci-Fi Enthusiasts |
| 6 | Self-Help Seekers |
| 7 | History Buffs |

**SQL Worksheet**

```
259
260 CREATE OR REPLACE PROCEDURE RETURNBOOK(BOOK_ID VARCHAR2, MEM_ID NUMBER) IS
261     FINE NUMBER(20);
262     MEMID NUMBER(20);
263     RETRN_DATE DATE NOT NULL := TO_DATE('07-MAY-20', 'DD-MON-YY');
264     DAT VARCHAR2(5);
265     DD DATE;
266 BEGIN
267     -- (A)RETURN THE BOOK IF THE MEMBER HAS BORROWED THE BOOK, CHECK IN THE EXISTENCE ISSUE_BOOK TABLE.
268     SELECT M_NO INTO MEMID FROM ISSUE_BOOK WHERE M_NO = MEM_ID AND B_NO = BOOK_ID;
```

```
Procedure created.

Procedure created.

Procedure created.

Procedure created.

Procedure created.

Trigger created.

Trigger created.
```

# NORMALIZATION

## Relation and Entity Analysis:

**MEMBER:**

1NF: The MEMBER table (M_NO, M_NAME, M_TYPE, NO_OF_BOOKS, TOT_FINE) adheres to 1NF.
2NF: Decomposition into MEMBER_INFO (M_NO, M_NAME, M_TYPE) and MEMBER_RECORD (M_NO, NO_OF_BOOKS, TOT_FINE) achieves 2NF.
3NF: The decomposed tables already comply with 3NF. FINE:
1NF: The provided table (M_NO, B_NO, FINE_ID, ISSUE_ID, FINE_AMT) satisfies 1NF requirements.
2NF: The table (M_NO, FINE_ID, ISSUE_ID, FINE_AMT) likely adheres to 2NF as well.
3NF: No transitive dependencies. BOOK:
1NF: The BOOK table (B_NO, B_NAME, AUTHOR, PRICE, NO_OF_BOOKS) complies with 1NF.
2NF & 3NF: The table already adheres to both 2NF and 3NF. ISSUE_BOOK:
1NF: The table (M_NO, B_NO, ISSUE_ID, ISSUE_DATE, RETURN_DATE, DUE_DATE) satisfies 1NF. BCNF:
ISSUE Table: (M_NO, B_NO, ISSUE_ID, ISSUE_DATE, DUE_DATE)
RETURN Table: (ISSUE_ID, RETURN_DATE) WAITLIST:
1NF: The WAITLIST table (M_NO, B_NO, WAITLIST_ID, WAITLIST_DATE) adheres to 1NF.
2NF & 3NF: The table follows both 2NF and 3NF.

**BOOK_CLUB:**

1NF: The BOOK_CLUB table (CLUB_NAME, GENRE, DESCRIPTION) satisfies 1NF.
2NF & 3NF: The table already adheres to both 2NF and 3NF. JOIN:
1NF: The JOIN table (M_NO, CLUB_NAME) satisfies 1NF. 2NF & 3NF: The table likely follows both 2NF and 3NF.

**BOOK_LOCATION:**

1NF: The original LOCATION table (B_ID, B_NO, FLOOR_NO, SUBJECT, SHELF_NO) adheres to 1NF.
2NF: The table likely follows 2NF with B_ID (assuming it uniquely identifies a location).
3NF: The table violate 3NF if SUBJECT solely depends on SHELF_NO.
LOCATION (B_ID, B_NO, FLOOR_NO, SHELF_NO)
SUBJECT (B_ID, B_NO, SUBJECT)

# APPLICATION

Highlighting the real-life applications of Library Management Systems:

- University Libraries: Ensures fair access to academic resources among students, faculty, and staff.
- Corporate Libraries: Supports professional development by managing book resources for employees.
- Digital Libraries: Handles e-book loans, maintains digital resource data, and tracks user interactions seamlessly.
- Community Centers: Fosters reading cultures through organized lending programs.
- Research Institutions: Maximizes access to scholarly materials via resource sharing and interlibrary loans.
- Nonprofit Organizations: Efficiently manages resource libraries, aiding knowledge dissemination in support of their missions.
- Government Libraries: Helps government agencies maintain information resources, ensuring compliance with standards.

These applications underscore the system's versatility, meeting diverse needs across settings and user groups.